Unsupervised Graph Outlier Detection: Problem Revisit, New Insight, and Superior Method

Yihong Huang[†], Liping Wang[†*], Fan Zhang[‡], Xuemin Lin[§]

[†]*East China Normal University,* [‡]*Guangzhou University,* [§]*Shanghai Jiao Tong University*

hyh957947142@gmail.com, lipingwang@sei.ecnu.edu.cn

zhangf@gzhu.edu.cn, lxue@cse.unsw.edu.au

***Abstract*—A large number of studies on Graph Outlier Detection (GOD) have emerged in recent years due to its wide applications, in which Unsupervised Node Outlier Detection (UNOD) on attributed networks is an important area. UNOD focuses on detecting two kinds of typical outliers in graphs: the structural outlier and the contextual outlier. Most existing works conduct experiments based on datasets with injected outliers. However, we find that the most widely-used outlier injection approach has a serious data leakage issue. By only utilizing such data leakage, a simple approach can achieve state-of-the-art performance in detecting outliers. In addition, we observe that existing algorithms have a performance drop with the mitigated data leakage issue. The other major issue is on balanced detection performance between the two types of outliers, which has not been considered by existing studies.**

In this paper, we analyze the cause of the data leakage issue in depth since the injection approach is a building block to advance UNOD. Moreover, we devise a novel variance-based model to detect structural outliers, which outperforms existing algorithms significantly and is more robust at kinds of injection settings. On top of this, we propose a new framework, Variance-based Graph Outlier Detection (VGOD), which combines our variance-based model and attribute reconstruction model to detect outliers in a balanced way. Finally, we conduct extensive experiments to demonstrate the effectiveness and efficiency of VGOD. The results on 5 real-world datasets validate that VGOD achieves not only the best performance in detecting outliers but also a balanced detection performance between structural and contextual outliers.

***Index Terms*—Graph Outlier Detection; Graph Neural Network; Unsupervised Graph learning; Attributed Networks**

I. Introduction

Graph Outlier Detection [1] (GOD, a.k.a. graph anomaly detection) is a fundamental graph mining task. It has various applications in high-impact domains and complex systems, such as financial fraudster identification [2]. The detection objects of GOD can be classified into different levels like node, edge, community, and graph [3]. For example, detecting abnormal users in a social network is the node-level GOD task while detecting abnormal molecules can be regarded as a graph-level GOD task.

Due to the high cost or unavailability of manually labeling the ground truth outliers, a large number of existing GOD approaches are carried out in an unsupervised manner [4, 5], which aims to detect the instances that significantly deviate from the majority of instances in graphs [6]. Attributed networks (a.k.a. attributed graphs) are a powerful data representation for many real-world complex systems (e.g. a social network with user profiles), in which entities can be represented as nodes with their attribute information; the interaction or relationship between entities can be represented as edges [7]. In recent years, the study of Unsupervised Node Outlier Detection (UNOD) on attributed networks has been blooming due to its wide applications [3, 8, 9]. Different from traditional global outlier detection and time series outlier detection, it defines two new typical types of outliers on attributed networks, namely, structural outlier and contextual outlier [4].

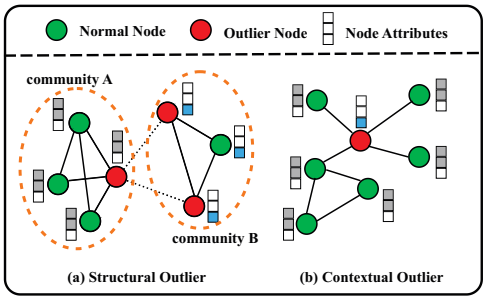


Fig. 1. An example of structural and contextual outliers in UNOD.

In Fig 1, there is a toy example for these two kinds of basic outliers. Particularly, structural outliers are those nodes structurally connected to different communities, i.e., their structural neighborhood is inconsistent. In other words, a structural outlier has normal attributes while it may have several abnormal links. For example, those people from different communities but have a strong connection with each other can be regarded as structural outliers. As shown in Fig 1(a), there are two communities outlined with orange circles and structural outliers are those nodes that have abnormal links to other communities. On the other hand, a contextual outlier has a consistent neighborhood structure while its attributes are corrupted, noisy, or significantly different from its neighbor nodes' attributes. For example, in Fig 1(b), suppose that the node in red is a football player while the nodes in green are music teachers. In this case, the node in red is regarded as a contextual outlier since it has a vast difference from

* Liping Wang is the corresponding author.

2375-026X/23/$31.00 ©2023 IEEE
DOI 10.1109/ICDE55515.2023.00197

its neighbors. In the real world, datasets are much more complicated than this toy example and it is difficult to measure the degree of inconsistency among nodes.

There have been various methods proposed to solve UNOD [9]. They can be roughly divided into two categories, namely, non-deep and deep-learning-based methods. Non-deep methods usually leverage traditional machine learning methods such as matrix factorization [10], density-based clustering [11], and relational learning [12] to encode the graph information and detect outliers. However, these methods fail to address the computational challenge with high-dimension data [13]. With the rapid prevalence of Graph Neural Networks (GNNs) [14], more and more methods are based on deep learning [15] and GNNs nowadays [3]. For example, DOMINANT [4] employs two GNN autoencoders to reconstruct the attribute information and structure information. According to the results reported in PyGOD [9], most deep-learning-based methods have a much better performance than non-deep methods in detecting injected outliers. To unify the outlier injection process, PyGOD [9] adopts the outlier injection approach from [4] as the standard injection approach.

Challenge. Although the recent deep-learning-based methods have achieved an excellent performance in UNOD, we find that the most widely-used outlier injection approach, which is adopted by [4, 9, 16, 17, 18, 19, 20, 21, 22, 23, 24], will cause a serious data leakage issue. Here, we refer to the data leakage [25] in machine learning, which means the information strongly associated with the labels is leaked to the training dataset. After employing this approach to inject outliers, structural outliers will have a larger node degree than the average while attribute vectors of contextual outliers will have a larger L2-norm (a.k.a. Euclidean norm) than expected. As a result, a simple solution only utilizing node degree or L2-norm of attribute vectors as the outlier score to detect the corresponding type of outliers can acquire a quite satisfying performance as shown in Fig 2. The metric of AUC [26] is adopted here to measure the detection performance. Under such a serious data leakage issue in injected datasets, most existing algorithms cannot have a better performance than the simple solution. In addition, it is observed that existing algorithms have a performance drop in varied injection settings, in which the data leakage issue caused by the current injection approach is alleviated. Therefore, it is urgent to find out the cause of data leakage and reduce its impact. On the other hand, it is also necessary to exploit an effective UNOD algorithm, which has a superior performance and is robust to the data leakage issue. Moreover, the balance between structural and contextual outliers detection performance is little considered in existing works [9]. An algorithm with unbalanced detection may only have detection ability for a certain type of outliers. It is found that existing algorithms focus more on contextual outliers than structural outliers when detecting them. To gain more feasible algorithms, comprehensive metrics for balance evaluation should be devised.

Our Solution. In this paper, we are devoted to analyzing the cause of data leakage and devising a superior outlier detection method for UNOD to achieve better performance in both the current injection setting and varied injection settings.

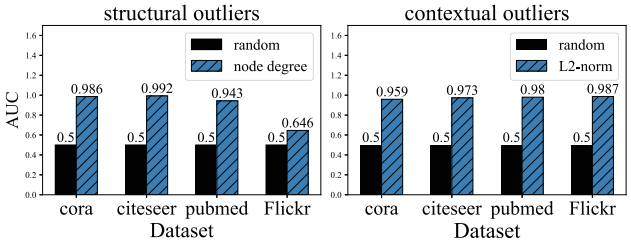


Fig. 2. After injecting outliers in four datasets, node degree is employed to detect structural outliers while L2-norm of attribute vector is employed to detect contextual outliers. Both of them, compared to the random detector, achieve unexpectedly high scores.

In particular, we propose a novel variance-based model to detect structural outliers, which adopts the variance of representations of neighbor nodes to detect structural outliers. To the best of our knowledge, it is the first time to employ the neighbor variance to detect outliers. Existing algorithms are based on either reconstruction of adjacency matrix [4, 18] or contrastive learning [16, 23] to detect structural outliers. According to the definition, the essence of a structural outlier is its inconsistent neighbors that come from different communities. However, existing algorithms cannot directly capture this essence. In this case, we devise a deep graph model to measure the inconsistency among neighbors by the variance of latent representations of neighbors, which captures the essence and gives a better utility for detection. On top of this, a new framework, Variance-based Graph Outlier Detection (VGOD), is also proposed to detect two types of outliers with a variance-based model and an attribute reconstruction model. To address the balance issue, we separately train two models to avoid overtraining and normalize two types of outlier scores to eliminate the scale difference. To evaluate the detection balance on two outlier types, we introduce a new metric to measure the gap in the performance score. The experiments are conducted on 5 real-world datasets and the results demonstrate that VGOD achieves the best detection performance.

Contribution. Our major contributions are as follows.

1) To the best of our knowledge, we are the first to identify the data leakage issue in the most widely-used outlier injection approach.
2) We analyze the cause of data leakage in depth, give suggestions for the future design of the outlier injection approach, and purpose a new approach for injecting structural outliers.
3) We propose a novel variance-based model and a new VGOD framework, which outperforms existing algorithms in detecting outliers and alleviates the issue of balanced detection.
4) Extensive experiments are conducted to demonstrate that our approach achieves the best detection performance and the detection balance.

II. Related Work

A. Graph Neural Network

GNNs [14] are a group of neural network models which utilize the graph structure for network representation learning and various tasks. Among GNNs, GCN [27] is one of the most influential models, which extends the convolutional operation in sequence or grid data to graph-structured data. Furthermore, to aggregate messages from neighbors more flexibly, GAT [28] introduces an attention mechanism to learn the importance of each neighbor node. On the other hand, GraphSage [29] adopts the sampling-based method to aggregate the neighbor information to work in large-scale graphs. From a topological learning perspective, GIN [30] is a more expressive model than GCN and can achieve the same discriminative power as the 1-WL graph isomorphism test [31]. In our proposed framework, GNN plays a vital role in the network embedding representation of nodes. Generally, the GNN module in our framework can be set to any type of the above-mentioned GNNs.

B. Unsupervised Node Outlier Detection on Attributed Networks

UNOD on attributed networks has attracted considerable research interest in recent years due to its wide application in complex systems. Radar [32] utilizes the residuals of attribute information and its coherence graph structure for outlier detection. ANOMALOUS [33] conducts attribute selection and outlier detection jointly based on CUR decomposition and residual analysis. However, these methods have computation limitations in high-dimension attributes due to their shallow mechanisms.

Quite a few studies based on the deep-learning technique have emerged recently [3]. Dominant [4] builds deep autoencoders on top of GCN layers to reconstruct the adjacency and attribute matrices. AnomalyDAE [18] employs dual autoencoder architecture with cross-modality interactions and the attention mechanism to reconstruct the adjacency and attribute matrices. CoLA [16] and SL-GAD [23] perform the UNOD task via contrastive self-supervised learning and random walk to embed nodes. AEGIS [17] studies UNOD in the inductive setting by utilizing generative adversarial ideas to generate potential outliers. DONE [34] employs deep unsupervised autoencoders to generate the network embedding which eliminates the effects of outliers at the same time. CONAD [19] adopts four data augmentation strategies and contrastive learning for outlier detection. GUIDE [21] replaces adjacency reconstruction with higher-order structure reconstruction to detect structural outliers. Under the manner of outlier injection, all these above deep methods show superior performance than non-deep methods in detecting these two types of outliers. To evaluate UNOD algorithms, PyGOD [9] adopts the most widely-used outlier injection approach from [4] as the standard injection method and provides unified benchmarks for UNOD, which facilitates fairness for comparing different methods.

Current UNOD methods have achieved an excellent performance. However, as demonstrated in Fig 2, the widely-used outlier injection approach exists a data leakage issue. To our surprise, simply using the combination of L2-norm and node degree to detect outliers can achieve state-of-the-art performance. Therefore, our work focuses on analyzing the cause of the data leakage issue and designing a superior method. In addition, as mentioned in [9] that no current method has a balanced detection performance on two outlier types, we also consider the balance issue in our method.

III. Preliminary

In this section, we formally present some concepts which are used throughout this paper and define the problem. We use lowercase letters (e.g. a), bold lowercase letters (e.g. $\boldsymbol{x}$), uppercase letters (e.g. X), and calligraphic fonts (e.g. $\mathcal{V}$) to denote scalars, vectors, matrices, and sets, respectively.

A. Graph Neural Network

GNNs stack L layers of message-passing layers. Each layer performs a message passing through the given graph structure. After the initial node feature $\boldsymbol{h_0} \in \mathbb{R}^{d_0}$ is transformed by L layers, the vector representation $\boldsymbol{h_L} \in \mathbb{R}^{d_L}$ is learned for each node v. Most message-passing layers can be expressed using the following rule:

$$\boldsymbol{h_v^{(l)}} = \sigma(\Psi^{(l)}(AGG(\{\Phi^{(l)}(\boldsymbol{h_u^{(l-1)}}), u \in \mathcal{N}_v \cup \{v\}\}))) \quad (1)$$

where $\sigma(\cdot)$ is the active function, $\Psi^{(l)}(\cdot)$ and $\Phi^{(l)}(\cdot)$ denote differentiable functions such as Multi-Layer Perceptrons (MLP). $AGG(\cdot)$ denotes a differentiable, permutation invariant function (e.g. sum, mean, max) and $\mathcal{N}_v$ denotes node v 's direct linked neighbors.

Here, we introduce three commonly used GNNs, namely GCN, GAT, and GIN.

Graph Convolution Network (GCN) [27] is the most widely-used GNN module, which adopts the propagation rule:

$$H^{(l+1)} = \sigma(\hat{A}H^{(l)}W^{(l)}) \quad (2)$$

where $\hat{A}$ is the symmetric normalized adjacency matrix, $H^{(l)}$ is the l^{th} hidden layer node representation, and $W^{(l)}$ is the parameters in the l^{th} hidden layer.

Graph Attention Network (GAT) [28] flexibly aggregates messages from neighbors with calculated weight α_{ij} (vs. average weight adopted by GCN) of each edge $\langle i, j \rangle$ as

$$\alpha_{ij} = \frac{exp(LeakyReLU(\boldsymbol{a}^T[W\boldsymbol{h_i}||W\boldsymbol{h_j}]))}{\sum_{k \in \mathcal{N}_i} exp(LeakyReLU(\boldsymbol{a}^T[W\boldsymbol{h_i}||W\boldsymbol{h_k}]))} \quad (3)$$

where $\boldsymbol{a}$ and W are the learnable weights. Layer mask (l) is omitted for simplicity.

Graph Isomorphism Network (GIN) [30] is the expressively more powerful GNN model, which follows the rule to propagate messages as

$$H^{(l)} = \sigma(\Psi^{(l)}(A + (1+\epsilon) \cdot I)H^{(l-1)}) \quad (4)$$

where ϵ can be a fixed or learnable scalar parameter, and I and A is the identity matrix and adjacency matrix, respectively.

B. Unsupervised Node Outlier Detection on Attributed Networks

Definition 1 (Attributed Network)**.** *An attributed network can be denoted as $\mathcal{G} = (\mathcal{V}, \mathcal{E}, X)$, where $\mathcal{V} = \{v_1, v_2, ..., v_n\}$ is the set of nodes ($|\mathcal{V}| = n$), $\mathcal{E}$ is the set of edges ($|\mathcal{E}| = m$), and $X \in \mathbb{R}^{n \times d}$ is the attribute matrix. The i^{th} row vector $\boldsymbol{x_i} \in \mathbb{R}^d$ of the attribute matrix denotes the attribute information of the i^{th} node. Node i's direct neighbors can be denoted as $\mathcal{N}_i$.*

With the aforementioned notations, the outlier detection problem on attributed network can be formally stated as a ranking problem.

Definition 2 (Outlier Detection on Attributed networks)**.** *Given an attributed network $\mathcal{G} = (\mathcal{V}, \mathcal{E}, X)$, the goal is to learn an outlier score function $f(\cdot)$ to calculate the outlier score $o_i = f(v_i)$ for each node. The higher the outlier score o_i is, the i^{th} node is more likely to be a structural outlier or a contextual outlier. By ranking all the nodes with their outlier scores, the abnormal nodes can be detected according to their ranking.*

In this paper, we consider the setting of unsupervised node outlier detection (UNOD), which is generally adopted by previous works. In this setting, none of labels of nodes is given in the training phase.

IV. Data Leakage issue Analysis

As shown in Fig 2, the current widely-used outlier injection approach exists a serious data leakage issue. In this section, we analyze the data leakage issue in detail. For these two types of outliers, we first introduce the outlier injection approach from [4]. Next, we theoretically analyze the cause of data leakage and give our suggestions for a better design of the outlier injection approach.

A. Structural Outlier

1) Injection Approach: The structural outliers are acquired by disturbing the topological structure of the graph. In a clique, nodes are fully connected. The intuition is that nodes in a clique should have a strong correlation with each other. Based on this, the outlier assumption is that a clique formed by unrelated nodes is structurally abnormal. The process of structural outlier injection is as followed. The first step is to specify the clique size q and the number of cliques p. Next, for each clique, q random nodes are chosen from the set of normal nodes and made fully connected as structural outliers. Therefore, total $p \times q$ structural outliers will be injected into the dataset. In previous works, the clique size q is fixed to 15 for all datasets, and the value of p is set according to the size of the dataset.

2) Cause Analysis: It is obvious that the chosen structural outliers will have a higher node degree since additional edges are added to them. Table I shows that none of the three citation networks (Cora, Citeseer, PubMed) have an average node degree greater than 3. However, due to the above injection approach, all the outliers will have a node degree of at least more than 15 (i.e. q) in previous work.

B. Contextual Outlier

1) Injection Approach: The contextual outliers are acquired by disturbing the attributes of nodes. The injection process is as followed. Firstly, total $p \times q$ normal nodes will be chosen as contextual outliers, which have the same number as structural outliers. Next, for each chosen outlier node v_i, another k nodes $\{v_{c1}, v_{c2}, ..., v_{ck}\}$ are randomly sampled from $\mathcal{V}$ as a candidate set $\mathcal{V}_c$. For each v_{ci} in $\mathcal{V}_c$, the Euclidean distance between the attribute vector $\boldsymbol{x_{ci}}$ of v_{ci} and $\boldsymbol{x_i}$ of v_i will be calculated. The attribute vector $\boldsymbol{x_{ci}}$ with the largest $\|\boldsymbol{x_{ci}} - \boldsymbol{x_i}\|_2$ will be used to replace $\boldsymbol{x_i}$ as the new attribute vector of v_i. The size of candidate set k is set to 50, ensuring that the disturbance amplitude is large enough.

2) Cause Analysis: To ensure a large enough disturbance of attributes, the above injection approach changes the $\boldsymbol{x_i}$ to $\boldsymbol{x_{ci}}$ with the largest $\|\boldsymbol{x_{ci}} - \boldsymbol{x_i}\|_2$. However, such a strategy will lead to the L2-norm of the final chosen $\boldsymbol{x_{ci}}$ (i.e. $\|\boldsymbol{x_{ci}}\|_2$) being more likely to be large. We make the following assumptions.

Assumption 1. *Suppose both $\boldsymbol{x_{ci}} \in \mathbb{R}^d$ and $\boldsymbol{x_i} \in \mathbb{R}^d$ are independently sampled from attribute matrix X. The rank of matrix X is greater than* 1.

Assumption 2. *For $\boldsymbol{x_{ci}} \sim X$, $\boldsymbol{x_{ci}} = \|\boldsymbol{x_{ci}}\|_2 \cdot \vec{e_{ci}}$, where $\|\boldsymbol{x_{ci}}\|_2$ and $\vec{e_{ci}}$ are the modulo and direction of $\boldsymbol{x_{ci}}$, respectively. We assumed that $\|\boldsymbol{x_{ci}}\|_2$ and $\vec{e_{ci}}$ are independently distributed.*

We use $P_r(x)$ to denote the possibility of x, then we have the following theorem.

Theorem 1. $P_r(\|\boldsymbol{x_{ci}} - \boldsymbol{x_i}\|_2 > \|\boldsymbol{x_{cj}} - \boldsymbol{x_i}\|_2 \Rightarrow \|\boldsymbol{x_{ci}}\|_2 > \|\boldsymbol{x_{cj}}\|_2) > 0.5$

Proof. We define $D(\boldsymbol{x_{ci}}, \boldsymbol{x_i}) = \|\boldsymbol{x_{ci}} - \boldsymbol{x_i}\|_2$. For notational convenience, we use $\boldsymbol{s}$ to refer to $\boldsymbol{x_{ci}}$ and $\boldsymbol{t}$ to refer to $\boldsymbol{x_i}$ Please note that both $\boldsymbol{s}$ and $\boldsymbol{t}$ are independently sampled from $X \in \mathbb{R}^{n \times d}$.

$$\begin{aligned} D^2(\boldsymbol{s}, \boldsymbol{t}) &= \sum_i^d (\boldsymbol{s}_i - \boldsymbol{t}_i)^2 \\ &= \sum_i^d \boldsymbol{s}_i^2 - 2\sum_i^d \boldsymbol{s}_i \boldsymbol{t}_i + \sum_i^d \boldsymbol{t}_i^2 \\ &= \|\boldsymbol{s}\|_2^2 - 2\|\boldsymbol{s}\|_2 \|\boldsymbol{t}\|_2 cos\alpha + \|\boldsymbol{t}\|_2^2 \\ &= f(\|\boldsymbol{s}\|_2) \end{aligned}$$

where α is the angle between vector $\boldsymbol{s}$ and $\boldsymbol{t}$, and $\|\boldsymbol{s}\|_2$ is the modulo of $\boldsymbol{s}$. From the above Equation, we can regard $D^2(\boldsymbol{s}, \boldsymbol{t})$ as a quadratic function $f(\cdot)$ of $\|\boldsymbol{s}\|_2$. Particularly, $\|\boldsymbol{s}\|_2 = \|\boldsymbol{t}\|_2 cos\alpha$ is the symmetry axis for $f(\|\boldsymbol{s}\|_2)$. According to the properties of a quadratic function in one variable, the function is monotonic on both sides of the symmetry axis. Therefore,

$$\begin{aligned} if \quad \|\boldsymbol{s}\|_2 > \|\boldsymbol{t}\|_2 cos\alpha \Rightarrow (f(\|\boldsymbol{s}\|_2) \uparrow \quad \Rightarrow \|\boldsymbol{s}\|_2 \uparrow) \\ if \quad \|\boldsymbol{s}\|_2 < \|\boldsymbol{t}\|_2 cos\alpha \Rightarrow (f(\|\boldsymbol{s}\|_2) \uparrow \quad \Rightarrow \|\boldsymbol{s}\|_2 \downarrow) \end{aligned}$$

where $\uparrow$ means increase and $\downarrow$ means decrease. In this case, we can draw the following conclusions.

$$\begin{aligned} P_r(\|\boldsymbol{s}\|_2 > \|\boldsymbol{t}\|_2 cos\alpha) &= P_r(f(\|\boldsymbol{s}\|_2) \uparrow \quad \Rightarrow \|\boldsymbol{s}\|_2 \uparrow) \\ &= P_r(D^2(\boldsymbol{s}, \boldsymbol{t}) \uparrow \quad \Rightarrow \|\boldsymbol{s}\|_2 \uparrow) \end{aligned}$$

$$= P_r(\|\boldsymbol{x_{ci}} - \boldsymbol{x_i}\|_2 > \|\boldsymbol{x_{cj}} - \boldsymbol{x_i}\|_2 \Rightarrow \|\boldsymbol{x_{ci}}\|_2 > \|\boldsymbol{x_{cj}}\|_2)$$

Since $\boldsymbol{s}$ and $\boldsymbol{t}$ are independently sampled from attribute matrix X, we can draw

$$P_r(\|\boldsymbol{s}\|_2 > \|\boldsymbol{t}\|_2) = 0.5.$$

Due to the assumption that the rank of X is greater than 1, the angle between $\boldsymbol{s}$ and $\boldsymbol{t}$ does not always equal zero. Therefore, $P_r(cos\alpha \equiv 1) < 1$. Note that $cos\alpha \leq 1$. Finally, we draw the following

$$P_r(\|\boldsymbol{s}\|_2 > \|\boldsymbol{t}\|_2 cos\alpha) > 0.5$$

which means

$$P_r(\|\boldsymbol{x_{ci}} - \boldsymbol{x_i}\|_2 > \|\boldsymbol{x_{cj}} - \boldsymbol{x_i}\|_2 \Rightarrow \|\boldsymbol{x_{ci}}\|_2 > \|\boldsymbol{x_{cj}}\|_2) > 0.5$$

□

Fig 2 verifies our analysis that only utilizing the L2-norm of attribute vectors of nodes can achieve nearly 0.98 AUC score for all these four datasets when $k = 50$.

Further, we vary the parameter k of the above injection approach. As k is set smaller, the data leakage issue is mitigated, which is shown in the left part of Fig 3, indicating the large k is the main cause for the serious data leakage issue. In the right part of Fig 3, we also replace the Euclidean distance by cosine distance in the injection approach. At this time, not all datasets have a data leakage issue when k becomes large. Therefore, Euclidean distance is also a key cause for data leakage.

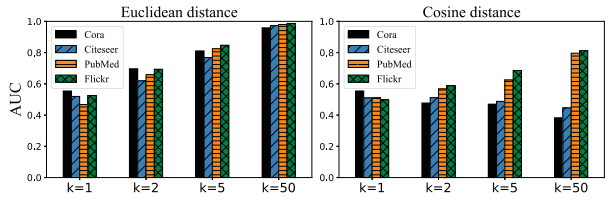


Fig. 3. AUC of L2-norm for contextual outliers injection with varying parameter of k (size of candidate set) and different distance measurements.

C. Suggestion

Due to the serious data leakage issue, it is hard to figure out whether the outlier detection algorithm has an effect on detection or potentially exploits the leaked information of labels. According to our experiment in Section VI-B, a simple baseline only using the leaked information outperforms existing deep-learning-based solutions that need a long time to train. This can be explained from two aspects:

1) The data leakage issue caused by the current injection approach is too serious, which results in little space for these algorithms to improve.
2) Existing outlier detection algorithms are not effective enough.

To mitigate data leakage caused by the current injection approach, we give the following suggestions on designing the new injection approach, better experiment, and parameter setting.

Suggestions for experiments and outlier injection. We give the suggestion for experiments and a new idea for outlier injection:

- Firstly, the data leakage issue should be examined. If it is hard to avoid data leakage, then the leaked information should be compared to see the exact improvement.
- Secondly, some datasets contain category labels for the node classification task. It is natural to think nodes with different labels come from different communities. Can we design a better injection approach based on this?

Suggestions for structural outliers injection. The size of the structural outlier clique is much larger than the average node degree of graph, which leads to node degree being a signal for structural outliers. Therefore, we can set the injection clique size q smaller for the current injection approach.

On the other hand, in real world, higher node degrees are not a signal for structural outliers. For example, a famous person has many friends, but these friends are all in his or her community circle, so this person is not a structural outlier. Therefore, to keep the distribution of node degree, replacing edges can be considered for a new injection approach.

Suggestions for contextual outliers injection. We have analyzed that the size of candidate set k and Euclidean distance are two key factors to cause data leakage. Based on this, we give these suggestions:

- Firstly, simply setting k smaller can mitigate the data leakage for the current injection approach.
- Secondly, replacing Euclidean distance with other distance measurements can be tried, such as cosine distance, shortest path distance, and so on.

D. Summary

In summary, the current widely-used outlier injection approach will cause the data leakage issue, both in structural and contextual outlier injection. We also give some suggestions for designing a new injection approach, better experiment, and parameter setting. Some of suggestions are applied in our experiment to evaluate the effectiveness of existing outlier detection solutions as well as our solution.

V. METHODOLOGY

In this section, we are going to illustrate our proposed framework VGOD in detail. Since current UNOD algorithms cannot outperform the simple baseline which only utilizes data leakage information, we propose our new framework VGOD, which combines a novel variance-based model and attribute reconstruction model. Specifically, the former model is for detecting structural outliers and the latter model is for detecting contextual outliers. Then we standardize the outlier scores outputted by two models and add them to get the final score. Fig 4 presents the whole architecture of VGOD framework.

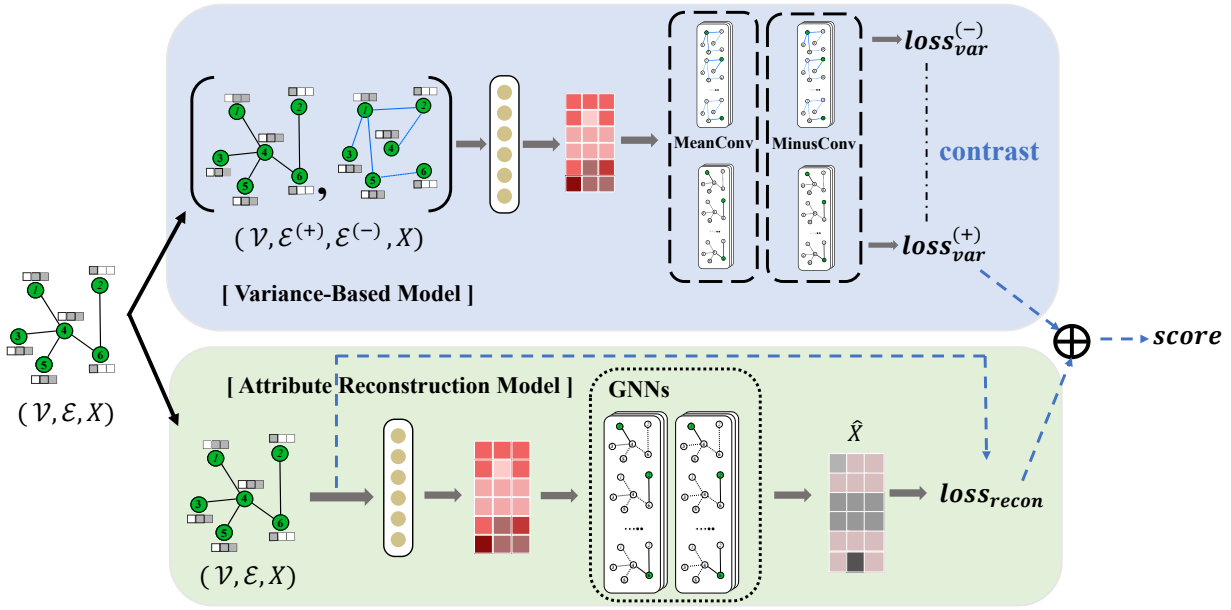


Fig. 4. The overview of our proposed unsupervised node-level graph outlier detection framework VGOD. For a given attributed network $\mathcal{G}$, the variance-based model and attribute reconstruction model are employed to calculate the structural and contextual outlier score, respectively. The final score is the sum of two standardized scores. In the variance-based model (VBM), we use a negative edge sampling technique to generate a corresponding negative edge set $\mathcal{E}^{(-)}$ per epoch which has the same number of edges as $\mathcal{E}$. VBM is trained by the contrastive learning of $\mathcal{E}$ and $\mathcal{E}^{(-)}$.

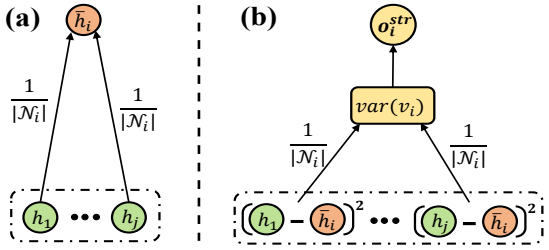


Fig. 5. (a) The MeanConv Layer. (b) The MinusConv Layer. We calculate the variance of neighbor nodes' low-dimension latent representation by (a) and (b).

A. Variance-based Model

In order to effectively detect structural outliers, we propose a novel Variance-Based Model (VBM). To the best of our knowledge, this is the first time to utilize the variance of neighbors to detect outliers. Neighbor variance measures the consistency of neighbor nodes. The bigger the variance, the less consistency it implies. In addition, our VBM has no bias on nodes with larger node degrees.

Feature Transformation. Before calculation of neighbor variance, we conduct the feature transformation $f_\theta(\cdot)$ for the original attribute matrix X and get the low-dimension hidden representation matrix H of nodes:

$$H = f_\theta(X) \tag{5}$$

where $f_\theta(\cdot)$ denotes a neural network, such as $MLP(\cdot)$. The i^{th} row vector $\boldsymbol{h_i}$ of the hidden representation matrix H denotes the latent representation of the i^{th} node. In our experiment, we implement it with a linear transformation and L2-normalization as :

$$\begin{aligned} \hat{H} &= XW + \boldsymbol{b} \\ \boldsymbol{h_i} &= \frac{\hat{\boldsymbol{h}}_i}{\|\hat{\boldsymbol{h}}_i\|_2} \end{aligned} \tag{6}$$

where $W \in \mathbb{R}^{d \times d_h}$ and $\boldsymbol{b} \in \mathbb{R}^{d_h}$ are the learnable parameters, d and d_h are the input dimension and hidden dimension of representation, respectively.

Neighbor Variance. In order to capture the consistency of neighbor nodes of a given node v_i, we calculate the variance of attribute vectors of neighbor nodes for v_i:

$$\overline{\boldsymbol{h}}_i = \frac{1}{|\mathcal{N}_i|} \sum_{j \in \mathcal{N}_i} \boldsymbol{h_j} \tag{7}$$

$$var(v_i) = \frac{1}{|\mathcal{N}_i|} \sum_{j \in \mathcal{N}_i} (\boldsymbol{h_j} - \overline{\boldsymbol{h}}_i)^2 \tag{8}$$

$$o_i^{str} = loss_{var}(v_i) = \|var(v_i)\|_1 \tag{9}$$

where $\overline{\boldsymbol{h}}_i$ is the average of hidden representations of neighbor nodes of v_i. The L1-norm of $var(v_i)$ is applied as structural outlier score o_i^{str} for node v_i. Since all components of the vector $var(v_i)$ are greater than 0, the L1 norm calculation is to simply sum components of the $var(v_i)$. In order to efficiently calculate variance for each node, we implement the calculation of variance based on the message-passing scheme [35] and design two message-passing layers without parameters, namely MeanConv and MinusConv, as illustrated in Fig 5. Concretely, MeanConv is employed to calculate Eq.

(7) and MinusConv is used to calculate Eq. (8) as well as Eq. (9).

Train. In order to train VBM to learn the representation that a normal node has a low variance while a structural outlier has a high variance, the train objective for VBM can be formally defined as below:

$$\min_{\theta} \mathbb{E}_{v_i \sim \mathcal{V}}[loss_{var}(v_i) - \frac{1}{|\mathcal{V}^{-\mathcal{N}_i}|} \sum_{j \notin \mathcal{N}_i} (\boldsymbol{h_j} - \frac{1}{|\mathcal{V}^{-\mathcal{N}_i}|} \sum_{u \notin \mathcal{N}_i} \boldsymbol{h_u})^2] \tag{10}$$

where $\mathcal{V}^{-\mathcal{N}_i} = \mathcal{V} \setminus \mathcal{N}_i$ is the non-neighbor node set of v_i.

We minimize the neighbor variance of a node while maximizing the variance of hidden representations of all the non-neighbor nodes. In this case, the model will avoid generating the same hidden representations for all nodes. However, it is too expensive to maximize the variance of all non-neighbor nodes every time. In this case, we apply negative edge sampling each epoch to generate a network $\mathcal{G}^{(-)}$ whose edge set $\mathcal{E}^{(-)}$ has the same number of edges as $\mathcal{E}$.

Definition 3 (negative edge set)**.** *For a given attributed network $\mathcal{G} = \{\mathcal{V}, \mathcal{E}, X\}$, if $\mathcal{E}^{(-)}$ is the negative edge set of $\mathcal{G}$, then $\langle u, v\rangle \in \mathcal{E}^{(-)} \Rightarrow \langle u, v\rangle \notin \mathcal{E}$, $\forall\ u, v \in \mathcal{V}$.*

Definition 4 (negative network)**.** *For a given attibute network $\mathcal{G} = \{\mathcal{V}, \mathcal{E}, X\}$, we define the negative network $\mathcal{G}^{(-)} = \{\mathcal{V}, \mathcal{E}^{(-)}, X\}$, where $\mathcal{E}^{(-)}$ is the negative edge set of $\mathcal{G}$.*

Therefore, we can utilize such a negative graph to maximize the variance of unrelated nodes. In other words, we randomly sample the same number of negative neighbors for each node v_i and maximize the neighbor variance calculated by these negative neighbors. Instead of maximizing the variance of all non-neighbor nodes, fewer nodes are required for computation by using negative sampling, which greatly saves time and space.

Therefore, for each node v_i, it has the "related vs unrelated" neighbor nodes pair, and corresponding $loss_{var}(v_i)^{(+)}$ and $loss_{var}(v_i)^{(-)}$ can be calculated respectively. The contrastive learning of neighbor nodes pair can be formalized as:

$$\begin{aligned} loss_{var}^{(+)}(v_i) &= \|var(v_i, \mathcal{G})\|_1 \\ loss_{var}^{(-)}(v_i) &= \|var(v_i, \mathcal{G}^{(-)})\|_1 \\ loss^{str}(v_i) &= loss_{var}^{(+)}(v_i) - loss_{var}^{(-)}(v_i) \end{aligned} \tag{11}$$

where $var(v_i, \mathcal{G})$ and $var(v_i, \mathcal{G}^{(-)})$ means we calculate the neighbor variance based on network $\mathcal{G}$ and $\mathcal{G}^{(-)}$, respectively.

Finally, we minimize the above $loss^{str}$ for all nodes in $\mathcal{V}$ as:

$$\min_{\theta} \mathbb{E}_{v \sim \mathcal{V}} loss^{str}(v) \tag{12}$$

Thus, trained VBM can output a larger neighbor variance score for nodes with unrelated neighbor nodes and a relatively small score for nodes with related neighbor nodes. Consequently, we can utilize VBM to detect structural outliers.

Can neighbor variance help detect contextual outliers? We employ a simple technique, self-loop edge, to make neighbor variance have an effect on detecting contextual outliers besides structural outliers. In specific, self-loop edges are added to all nodes as

$$\hat{\mathcal{N}}_i = \mathcal{N}_i \cup \{v_i\}, \forall v_i \in \mathcal{V} \tag{13}$$

where $\mathcal{N}_i$ is the neighbor set of node v_i. As the attributes of a contextual outlier are significantly different from its neighbors, neighbor variance of it would be increased greatly after adding the self-loop neighbor when $|\mathcal{N}_i|$ (i.e. node degree of node v_i) is small. This technique is optional and we employ it when the average node degree of graph is small. Our experiment in Section VI-E5 studies the effect of this technique.

B. Attribute Reconstruction Model

We employ attribute reconstruction in the detection of contextual outliers. Our attribute reconstruction Model (ARM) is flexible that any popular GNN model can be used as the backbone to reconstruct the attributes of nodes.

Feature Transformation. Similar to VBM, we first transform the original attribute matrix X to the low-dimension feature representation matrix $Z^{(0)}$ as:

$$\begin{aligned} \hat{Z} &= XW' + \boldsymbol{b'} \\ \boldsymbol{z_i^{(0)}} &= \frac{\boldsymbol{\hat{z}_i}}{\|\boldsymbol{\hat{z}_i}\|_2} \end{aligned} \tag{14}$$

where W' and $\boldsymbol{b'}$ are the learning parameters, $\boldsymbol{z_i^{(0)}}$ is the i^{th} row vector of $Z^{(0)}$.

GNN Layers. Then we employ L GNN layers to transform $Z^{(0)}$ to $Z^{(L)}$ to fully absorb the message from neighbor nodes. The l^{th} GNN Layer can be formalized as:

$$Z^{(l)} = GNN^{(l)}(Z^{(l-1)}, \mathcal{G}) \tag{15}$$

where l^{th} operator $GNN^{(l)}(\cdot)$ can be implemented by any popular GNN model like GCN [27], GAT [28], GIN [30], and so on.

Feature Retransformation. Finally, we retransform the $Z^{(L)}$ to $\hat{X}$, where $\hat{X} \in \mathbb{R}^{|\mathcal{V}| \times d}$ has the same shape as original attributes matrix X.

$$\hat{X} = Z^{(L)}\hat{W} + \hat{\boldsymbol{b}} \tag{16}$$

where $\hat{X}$ is the reconstruction of the attribute matrix, $\hat{W}$ and $\hat{\boldsymbol{b}}$ are the weight and bias parameters. Thus we can use the reconstruction attribute matrix $\hat{X}$ to calculate the reconstruction error, which is denoted as

$$o_i^{attr} = loss^{recon}(v_i) = \|\boldsymbol{\hat{x}_i} - \boldsymbol{x_i}\|^2 \tag{17}$$

$$\min_{\theta} \mathbb{E}_{v \sim \mathcal{V}} loss^{recon}(v) \tag{18}$$

By minimizing the above objective, trained ARM can detect contextual outliers.

C. Outlier Detection

As mentioned in [9], current UNOD algorithms fail to have balanced performance on two outlier types. During the training stage, the previous practice that combines contextual and structural loss with a fixed weight fails to balance the optimization of model parameters. Similarly, during the inference stage, combing the contextual and structural score

with a fixed weight fails to achieve a balanced detection performance. Therefore, we separately train our VBM and ARM with different epochs to avoid unbalanced optimization.

Score combination. After both VBM and ARM are well-trained, we employ the mean-std normalization on two types of outlier scores outputted by two models and sum scores to get the final score, which can be formalized as:

$$\begin{aligned} \widehat{o}_i^{str} &= \frac{o_i^{str} - \mu(\mathcal{O}^{str})}{std(\mathcal{O}^{str})} \\ \widehat{o}_i^{attr} &= \frac{o_i^{attr} - \mu(\mathcal{O}^{attr})}{std(\mathcal{O}^{attr})} \\ o_i &= \widehat{o}_i^{str} + \widehat{o}_i^{attr} \end{aligned} \tag{19}$$

where $\mathcal{O}^{str}$ and $\mathcal{O}^{attr}$ denote the set of structural outlier scores and contextual outlier scores, respectively. $\mu(\cdot)$ denotes the mean function, $std(\cdot)$ denotes standard deviation function.

By adopting Eq. (19) as the final outlier score, our model can have a more balanced performance in detecting two types of outliers during the inference stage. The overall procedure of our VGOD framework is described in Algorithm 1.

Algorithm 1 The overall procedure of VGOD framework

Input: Attributed Network: $\mathcal{G} = (\mathcal{V}, \mathcal{E}, X)$, Training epochs for VBM: $Epoch_{VBM}$, Training epochs for ARM: $Epoch_{ARM}$

Output: Well-trained VBM and ARM, outlier scores $\mathcal{O}$

// Training phase
for $i \in 1, 2, ..., Epoch_{VBM}$ **do**
  Generate the negative network $\mathcal{G}^{(-)} = (\mathcal{V}, \mathcal{E}^{(-)}, X)$ by negative edge sampling.
  Compute the neighbor variance of nodes in $\mathcal{G}^{(+)}$ and $\mathcal{G}^{(-)}$ via Eq. (6)-(9).
  Update VBM with the loss function via Eq. (11).
end for
for $i \in 1, 2, ..., Epoch_{ARM}$ **do**
  Compute the reconstruction node attributes via Eq. (14)-(16).
  Update ARM with the loss function via Eq. (17).
end for
// Inference phase
Compute the $\mathcal{O}^{str}$ and $\mathcal{O}^{attr}$ via VBM and ARM, respectively.
Compute the final outlier scores $\mathcal{O}$ via Eq. (19).
return VBM, ARM, $\mathcal{O}$

D. Complexity Analysis

The complexity is mainly bounded by message-passing layers. For simplicity, the number of layers and the number of dimensions are considered constant. The space and time complexity both are $O(|\mathcal{E}| + |\mathcal{V}|)$. In addition, we are only using GNN layers and two linear layers to build our model. Since there are a large number of research on extending GNNs to larger networks, we can make use of various mini-batch training techniques such as [29, 36, 37] to extend our model in a large-scale network without much effort.

TABLE I
DATASETS FOR UNOD EXPERIMENTS.

Dataset	#nodes	#edges	#attrs	#avg_deg	#outliers*	%outlier*
Cora	2,706	5,429	1,433	2.01	150	5.5%
Citeseer	3,327	4,732	3,703	1.42	150	4.5%
PubMed	19,717	44,338	500	2.25	600	3.0%
Flickr	7,575	239,738	12,407	31.65	450	5.9%
Weibo	8,405	407,963	64	48.5	868	10.3%

*#outlier and %outlier are only for UNOD Experiments.

VI. EXPERIMENT

In this section, we conduct experiments to illustrate the effectiveness of our proposed framework VGOD. Firstly, we describe the experiment settings including datasets, baselines, evaluation metrics, and computing infrastructures. Then, we conduct the Unsupervised Node Outlier Detection (UNOD) experiment to validate the effectiveness of our framework. Next, we conduct two structural outlier detection experiments under different injection parameters and a new injection approach, respectively. Finally, we make further analysis for our approach, including efficiency and ablation study. Our code is available at https://github.com/goldenNormal/vgod-github.

A. Experiment settings

1) Datasets: We evaluate the proposed framework on five real-world datasets for UNOD on attributed networks, including four widely-used benchmark datasets with injected outliers and one dataset with labeled outliers. These datasets, shown in Table I, include three citation networks[1] (Cora, Citeseer, PubMed) and two social networks (Flickr[2] and Weibo[3]). Only the Weibo dataset contains labeled outliers.

2) Baselines: We compare our proposed framework VGOD with the recent five deep-learning-based SOTA models. These baselines are summarized in Table II. Column time complexity indicates the time complexity of inference. For simplicity, the number of layers and the number of dimensions are considered as constant. If the model outputs more than one score for outliers like VGOD, then we consider that it has the feature of score combination. If the hyperparameters of the model (e.g., the number of layer units) are not coupled to the number of nodes or edges of the training graph, then we regard it can perform inductive inference, which means a trained model can be directly used for detecting outliers on a new graph with the same attribute schema. Since none of labels is given in the training phase, our experiments are conducted in the transductive setting, which is consistent with existing unsupervised outlier detection works.

Due to data leakage in the outlier injection approach, we also design a simple baseline for comparison and evaluation, which only utilizes the leaked information (i.e. node degree and L2-norm of attribute vectors). We name it DegNorm.

DegNorm adopts node degree as structural outlier score while L2-norm of attribute vectors of nodes are adopted as

[1]https://linqs.org/datasets/
[2]https://github.com/mengzaiqiao/CAN
[3]https://github.com/zhao-tong/Graph-Anomaly-Loss/tree/master/data

TABLE II
THE COMPARISON OF BASELINES.

Baseline	Time Complexity	Contrastive Learning	Reconstruction	Score Combination	Inductive Inference
Dominant [4]	$O(\|\mathcal{E}\|+\|\mathcal{V}\|^2)$	×	✓	✓	✓
AnamalyDAE [18]	$O(\|\mathcal{E}\|+\|\mathcal{V}\|^2)$	×	✓	✓	×
DONE [34]	$O(\|\mathcal{V}\|K)$[1]	×	✓	✓	✓
CoLA [16]	$O(c\|\mathcal{V}\|R(c+\delta))$[2]	✓	×	×	✓
CONAD [19]	$O(\|\mathcal{E}\|+\|\mathcal{V}\|^2)$	✓	✓	✓	✓
VGOD (Ours)	$O(\|\mathcal{E}\|+\|\mathcal{V}\|)$	✓	✓	✓	✓

[1] K denotes the number of sampling neighbors for each node

[2] R denotes the number of sampling rounds, c denotes the number of nodes within the local subgraph, and δ denotes the average degree of the network.

contextual outlier score. The mean-std normalization is applied to two scores. The final outlier score is the sum of these two scores which have been normalized. The calculation of o_i^{str} and o_i^{attr} can be formalized as:

$$\begin{aligned} o_i^{str} &= |\mathcal{N}_i| \\ o_i^{attr} &= \|\boldsymbol{x_i}\|_2 \end{aligned} \tag{20}$$

where $\mathcal{N}_i$ is the neighbor node set of node v_i, $\boldsymbol{x_i}$ is the attribute vector of node v_i.

3) Evaluation Metrics: We use *Area Under receiver operating characteristic Curve* (AUC) to measure. In specific, AUC evaluates the degree of alignment between the outlier score and the ground truth label under varying thresholds:

$$AUC = \frac{1}{|\mathcal{V}^+||\mathcal{V}^-|} \sum_{v_i^+ \in \mathcal{V}^+} \sum_{v_j^- \in \mathcal{V}^-} (\mathbb{I}(f(v_i^+) < f(v_j^-))) \tag{21}$$

where $\mathcal{V}$, $\mathcal{V}^-$, and $\mathcal{V}^+ = \mathcal{V} \setminus \mathcal{V}^-$ are the set of all nodes, the set of all outlier nodes, and the set of all normal nodes respectively, $\mathbb{I}(\cdot)$ is the indicator function and $f(v_i)$ is the outlier score of node v_i given by one outlier detector. To explore the utility of the model for different outliers, we extend the concept of AUC. Generally, $AUC(\mathcal{V}_L, \mathcal{O})$ means using $\mathcal{V}_L$ as the set of outliers to be detected, $\mathcal{O}$ as the outlier scores to calculate the AUC. In other words, $\mathcal{V}_L$ defines the outlier labels. Particularly, $AUC = AUC(\mathcal{V}^-, \mathcal{O})$. In addition, if a model can output the structural and contextual outlier scores like our VGOD, then $AUC(\mathcal{V}^-, \mathcal{O}^{str})$ and $AUC(\mathcal{V}^-, \mathcal{O}^{attr})$ can be calculated.

We also propose $AucGap$ to evaluate the balanced detection performance for different types of outliers, which can be formalized as below:

$$AucGap = max\{\frac{AUC(\mathcal{V}^{str}, \mathcal{O})}{AUC(\mathcal{V}^{attr}, \mathcal{O})}, \frac{AUC(\mathcal{V}^{attr}, \mathcal{O})}{AUC(\mathcal{V}^{str}, \mathcal{O})}\} \tag{22}$$

where $\mathcal{V}^{str}$ and $\mathcal{V}^{attr}$ are structural outliers set and contextual outliers set, respectively. $AucGap$ aims to calculate the gap between the model's AUC score for two types of outliers. The lower the $AucGap$ is, the more balanced detection performance it indicates.

4) Computing Infrastructures: Our proposed learning framework is implemented using PyTorch 1.11.1 and PyTorch Geometric 2.1.0. All experiments are conducted on a computer with Ubuntu 16.04 OS, i7-9750H CPU, and a Tesla V100 (32GB memory) GPU.

B. Unsupervised Node Outlier Detection

We first conduct the UNOD experiment to verify the effectiveness of our proposed framework. UNOD experiment hereinafter refers to this experiment.

1) Injection Setting: We adopt the most widely-used outlier injection approach as mentioned in Section IV-A1 and Section IV-B1. We keep the same injection parameter setting with [4, 16, 20, 21, 22, 23] to have a fair comparison (i.e., $q = 15$, $k = 50$ for all datasets and $p = 5, 5, 20, 15$ for Cora, Citeseer, PubMed, Flickr, respectively). The statistics of these datasets are demonstrated in Table I. Only Weibo contains the labeled outliers while other datasets contain injected outliers. Note that $AucGap$ can only be calculated on these injected datasets.

2) Parameter Setting: For each algorithm, we run 5 times and calculate the average score. For our proposed framework VGOD, we fix the embedding dimension to 128 for both the Variance-Based Model (VBM) and Attribute Reconstruction Model (ARM). We set the learning rate to 0.005 for all injected datasets and 0.01 for Weibo. Two layers of GAT are adopted as the GNN module in ARM and the row-normalization to the attribute vectors is applied in Weibo. We employ self-loop edge technique in Cora, Citeseer, PubMed, and Weibo for VGOD. We directly run the code in [16] to inject outliers. For all baselines, we adopt the default parameter setting in their code except the number of training epochs. We stop training their model as long as their AUC score reaches its peak. In this case, the performance can be promised to be better or equal to the performance of their default parameter setting. For our approach, we train ARM 100 epochs and VBM 10 epochs for all datasets since it has already significantly outperformed baselines in a fixed number of training epochs. In fact, our two models require fewer epochs to converge. Adam optimizer is employed to train our models. We adopt the AUC score of Weibo published in [9] for the baseline Dominant, AnomalyDAE, DONE, and CONAD.

3) Result Analysis: The AUC scores and $AucGap$ scores are shown in Table IV and Table III. $AUC(V^{str}, \mathcal{O})$ and $AUC(V^{attr}, \mathcal{O})$ are listed in the column of *str* and *context*. The best score is marked in bold while the second best is underlined. The AUC scores lower than 0.7 are red and italicized in Table III. According to the results, we have the following observations:

- Our proposed framework VGOD achieves the highest AUC score for all datasets while achieving the overall

TABLE III
AUCGAP OF UNOD EXPERIMENT.

Model	Cora			Citeseer			PubMed			Flickr		
	AucGap	str	context	*AucGap*	str	context	*AucGap*	str	context	*AucGap*	str	context
Dominant	1.312	*0.696*	0.913	1.165	0.755	0.880	1.652	*0.600*	0.990	2.029	*0.486*	0.986
AnomalyDAE	1.161	0.895	0.771	1.070	0.864	0.808	1.118	0.933	0.834	1.860	*0.521*	0.969
DONE	1.217	0.922	0.758	**1.016**	0.872	0.886	1.217	0.836	0.687	1.557	*0.578*	0.900
CoLA	1.127	0.943	0.837	1.188	0.953	0.802	1.054	0.954	0.905	1.395	*0.622*	0.868
CONAD	1.877	*0.513*	0.964	2.236	*0.434*	0.972	2.417	*0.404*	0.976	2.066	*0.478*	0.987
DegNorm	1.132	0.936	0.827	1.116	0.979	0.877	1.093	0.861	0.941	1.822	*0.527*	0.960
VGOD	**1.072**	0.970	0.905	1.026	0.986	0.961	**1.021**	0.962	0.983	**1.066**	0.838	0.893

TABLE IV
AUC FOR UNOD EXPERIMENT.

Model	Cora	Citeseer	PubMed	Flickr	Weibo
Dominant	0.8134	0.8250	0.7999	0.7440	0.925*
AnomalyDAE	0.8433	0.8441	0.8898	0.7524	0.928*
DONE	0.8498	0.8800	0.7664	0.7482	0.887*
CoLA	0.8790	0.8861	0.9214	0.7530	0.748
CONAD	0.7456	0.7078	0.6930	0.7395	0.927*
DegNorm	0.8928	0.9385	0.9074	0.7515	0.893
VGOD	**0.9503**	**0.9845**	**0.9813**	**0.8773**	**0.9765**

* denotes the result reported in [9]

highest $AucGap$ among all datasets. There are several reasons for such performance. Firstly, our variance-based model significantly improves the ability to detect structural outliers. Secondly, we separately train two models to prevent each component from being over-trained. Thirdly, we adopt mean-std normalization to eliminate the scale difference between the two scores which gives a more balanced detection performance.

- DegNorm also achieves SOTA performance compared to other baselines.
- In Table III, it is observed that all baselines can not have a good detection performance on structural outliers in the Flickr dataset and achieve a poorly balanced detection.

Though the $AucGap$ of VGOD in Citeseer is slightly lower than DONE, its detection performance is already balanced. Moreover, both $AUC(V^{str}, \mathcal{O})$ and $AUC(V^{attr}, \mathcal{O})$ of VGOD are much higher than that of DONE.

C. Structural Outlier Detection under different injection parameters

Further, we conduct the structural outlier detection experiment with varied injection parameters to explore the effectiveness of our variance-based model (VBM) in depth.

1) Injection Setting: We vary the parameter q of injected clique size of structural outliers to $Q = \{3, 5, 10, 15\}$. For each dataset $\mathcal{D}_i$, we inject 4 groups of structural outliers $\{\mathcal{V}^{q=3}, \mathcal{V}^{q=5}, \mathcal{V}^{q=10}, \mathcal{V}^{q=15}\}$ into $\mathcal{D}_i$. Each group has the same number of outliers, which is set to 2% of the total number of nodes, i.e. $|\mathcal{V}^{q=Q_i}| = 2\% \cdot |\mathcal{V}|$. The outlier set $\mathcal{V}^{-}$ is the union of 4 groups of structural outliers set. We report the $AUC(\mathcal{V}^{-}, \mathcal{O}^{str})$ in Table V and the AUC score of each group $AUC(V^{q=Q_i}, \mathcal{O}^{str})$ is shown in Fig 6. Note that $\mathcal{O}^{str}$ of VGOD is the output of VBM. The injected outliers are all structural outliers.

2) Parameter Setting: We keep the same parameter setting for VBM and other baselines as the UNOD experiment except that we train baselines and VBM until their AUC scores reach the peak. Since we fail to get a reasonable result for CONAD, we do not list the result of it. We also evaluate the performance of simple baseline **Deg**, which only utilizes the node degree as an outlier score for comparison. For all other baselines, if their model outputs multiple scores (e.g., o_i, o_i^{str}, o_i^{attr}), we adopt the score with the highest AUC as its structural score.

TABLE V
AUC FOR STRUCTURAL OUTLIER DETECTION UNDER DIFFERENT INJECTION PARAMETERS

Model	Cora	Citeser	PubMed	Flickr
Dominant	0.9227	0.9467	0.8878	0.5715
AnomalyDAE	0.9127	0.9219	0.8968	0.6253
DONE	0.9034	0.8985	0.8868	0.5516
CoLA	0.8073	0.8919	0.8698	0.5712
Deg	0.9467	0.9541	0.9333	0.5671
VBM	**0.9815**	**0.9816**	**0.9893**	**0.8003**

3) Result Analysis: According to results in Table V and Fig 6, we have the following observations:

- VBM achieves the best AUC score for all datasets in Table V. In addition, VBM has a huge performance gain in Flickr.
- As shown in Fig 6, when the clique size is reduced, the performance of VBM declines the least compared to other baselines. Therefore, the performance of VBM is the most robust to varied injection settings.
- Deg that directly utilizes a node's degree outperforms other baselines in Cora, Citeseer, and PubMed.

D. Structural Outlier Detection under a new injection approach

In this subsection, we design a new approach to inject structural outliers without data leakage. We conduct the following experiment for evaluation.

1) Injection approach: Since all these datasets have category labels for the node classification task, it is natural to think that the nodes with different labels are from different communities. In our opinion, structural outliers do not necessarily form clusters. We generate structural outliers by replacing their

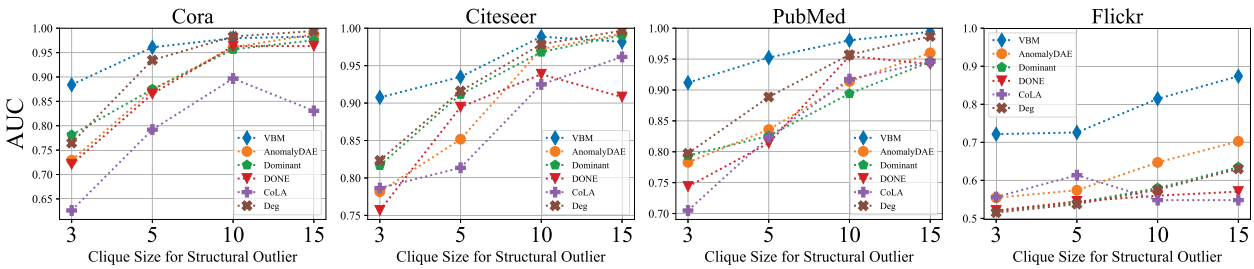


Fig. 6. Comparison of the detection performance with the varying clique size parameter. Each polyline represents the AUC values of a model on several groups of structural outliers with different clique sizes.

original neighbors (both in and out) with nodes uniformly sampled from other communities. In this manner, the degree distribution of all outliers is not changed. The number of outliers is set as 10% of the number of nodes. The injected outliers are all structural outliers.

TABLE VI
AUC FOR STRUCTURAL OUTLIER DETECTION UNDER A NEW INJECTION APPROACH

	Cora	Citeseer	PubMed	Flickr
Dominant	0.838	0.770	0.853	0.917
AnomalyDAE	0.770	0.673	0.566	0.898
DONE	0.762	0.664	0.659	0.541
CoLA	0.658	0.743	0.752	0.632
CONAD	0.793	0.770	0.779	0.495
VBM	**0.935**	**0.907**	**0.858**	**0.958**

2) Result Analysis: We keep the same parameter setting for VBM and all baselines as the experiment in Section VI-C. Table VI lists the $AUC(\mathcal{V}^{-}, \mathcal{O}^{str})$. Our VBM is still the most effective model, which outperforms others with a significant gap. This further verifies the effectiveness of neighbor variance to detect structural outliers.

E. Further Analysis

In this subsection, we make further analysis of our proposed framework.

1) Efficiency of model inference: We calculate the time for each model to use the CPU for training and inference at the setting of UNOD experiment. The training time per epoch of all models (in seconds) is shown in Fig 7. In Table VII, we list the inference time in seconds. The inference time of the model is roughly the same as the training time per epoch, except for CoLA. For all datasets, our VGOD framework completes inference in a relatively short time. For datasets with a large number of nodes, such as PubMed, our model takes significantly less time than other models due to the linear relationship to the number of nodes. Since CoLA requires multiple rounds of sampling for inference, its computational cost is much higher than other models.

2) Effect of the number of epochs for VBM: We investigate the AUC variation trend of VBM during training. As shown in Fig 8, VBM shows a high AUC score at the beginning, and the AUC score reaches the peak after only a few epochs of training. Afterward, as the training progresses, the AUC score slowly decreases due to overfitting. Different group of structural outliers shows a similar trend while the group of smaller clique size shows a later overfitting time point.

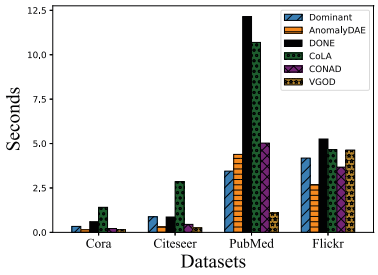


Fig. 7. Training time per epoch in seconds.

TABLE VII
INFERENCE TIME OF MODELS (IN SECONDS).

Model	Cora	Citeseer	PubMed	Flickr
Dominant	0.102	0.235	3.021	4.183
AnomalyDAE	0.147	0.303	4.390	2.493
DONE	0.604	0.865	12.147	5.256
CoLA	413	752	3266	910
CONAD	0.093	0.201	2.823	**1.379**
VGOD	**0.088**	**0.145**	**0.874**	3.899

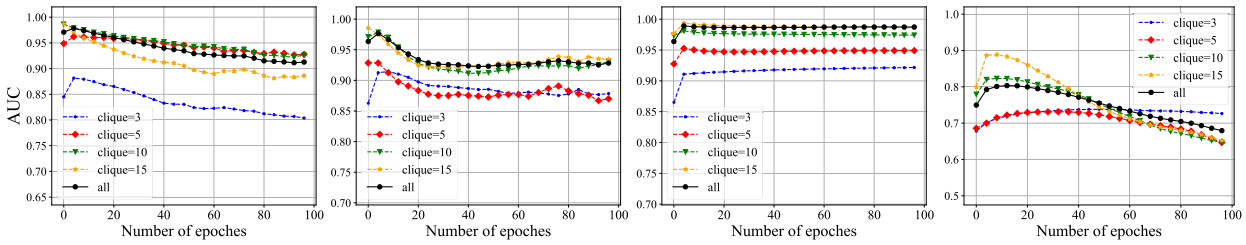
3) Effect of different GNN Layers for ARM: We investigate the effect of different GNN layers in ARM. We replace different GNN layers in the UNOD experiment for research. Table VIII and Table IX show the AUC and $AucGap$ respectively on four datasets. It is observed that GAT outperforms other GNNs significantly on the Weibo. For other datasets, their AUC and $AucGap$ scores are comparable.

TABLE VIII
AUC VALUES COMPARISON FOR DIFFERENT GNN LAYERS.

Model	Cora	Citeseer	PubMed	Flickr	weibo
VGOD (GIN)	0.9503	0.9845	0.9801	0.8773	0.9093
VGOD (GCN)	**0.9566**	0.9867	0.9802	0.8735	0.9154
VGOD (GAT)	0.9560	**0.9868**	**0.9813**	**0.8835**	**0.9765**

TABLE IX
AUCGAP VALUES COMPARISON FOR DIFFERENT GNN LAYERS.

Model	Cora	Citeseer	PubMed	Flickr
VGOD (GIN)	1.0716	**1.0261**	1.0215	**1.0655**
VGOD (GCN)	**1.0637**	1.0278	1.0214	1.0713
VGOD (GAT)	1.0680	1.0268	**1.0211**	1.0672

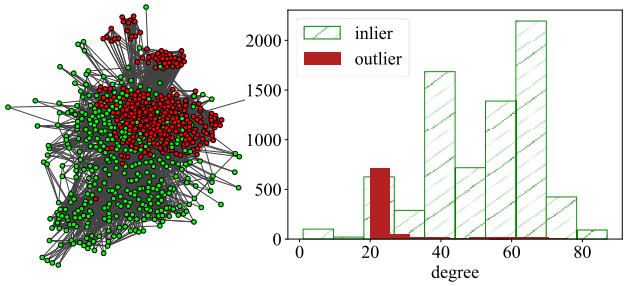


(a) Cora (b) Citeseer (c) PubMed (c) Flickr

Fig. 8. AUC variation trend of the variance-based model during the training. Each polyline represents a group of structural outliers with a certain clique size.

4) Labeled outlier study: We make the analysis on the Weibo dataset. We compare VGOD with the second best baseline AnomalyDAE in Table X. It reveals that the main reason for VGOD's superior performance is its improvement in structural outlier detection. It is shown in Fig 9(b) that outliers do not have a higher node degree distribution. In addition, we find that attribute vectors of outliers are more diverse, as the variance of attribute vectors among all outliers is 425.0 and that of the inliers is 11.95.

From Fig 9(a), we find that both inliers (green points) and outliers (red points) are quite cohesive. The homophily [38] of the whole graph is 0.75. Note that a random graph has a homophily of 0. In this case, these outliers, which differ greatly from each other, are connected closely, forming clusters of structural outliers. Therefore, it is easily detected by the neighbor variance of VGOD. There are also a lot of clusters formed by inliers. They are not regarded as outliers since their attributes are close.



(a) (b)

Fig. 9. (a) A subgraph in Weibo. The nodes in red denote the outliers. (b) Node degree distribution of Weibo.

TABLE X
AUC DETAIL IN THE WEIBO

	AUC	$AUC(\mathcal{V}^-, \mathcal{O}^{str})$	$AUC(\mathcal{V}^-, \mathcal{O}^{attr})$
VGOD	0.977	0.922	0.926
AnomalyDAE	0.925	0.796	0.925

5) Effect of the self-loop edge: We study the effect of the self-loop edge technique, which enables neighbor variance to detect contextual outliers besides structural outliers. In the first step, we study the effect of variance-based model to detect contextual outliers. We inject only contextual outliers into the datasets, using the same injection parameters as Section VI-B1. In Table XI and Table XII, "w/ SL" means employing the self-loop edge technique. The result in Table XI shows that neighbor variance with this simple technique indeed has an effect on detecting contextual outliers, especially in those citation networks with small node degrees.

TABLE XI
AUC OF VBM TO DETECT CONTEXTUAL OUTLIERS

	Cora	Citeseer	PubMed	Flickr
VBM	0.5026	0.5128	0.4883	0.4725
VBM w/ SL	0.7978 ↑	0.8567 ↑	0.8364 ↑	0.6463 ↑

TABLE XII
AUC OF VGOD IN ABLATION OF SELF-LOOP EDGE

	Cora	Citeseer	PubMed	Flickr	Weibo
VGOD	0.8911	0.9485	0.9592	0.8773	0.9707
VGOD w/ SL	0.9503 ↑	0.9845 ↑	0.9813 ↑	0.8313	0.9765 ↑

In the second step, we do the ablation study of this technique under the UNOD experiment. Table XII demonstrates that self-loop edge also greatly improves the detection performance of VGOD in those citation networks due to the extra utilities of neighbor variance on contextual outlier detection.

VII. CONCLUSION

In this paper, we revisit the problem of unsupervised node outlier detection. Firstly, we find that the current outlier injection approach exists a serious data leakage issue and make a theoretical analysis in depth. Secondly, we propose a new framework, which consists of a novel variance-based model and a more general attribute reconstruction model to detect two types of outliers. Our model successfully outperforms all previous SOTA models with the best outlier detection performance and the detection balance.

We believe our insight into the data leakage issue will lead to better outlier injection approaches and UNOD algorithms. Moreover, the concept of neighbor variance may also exhibit great potential in other research areas such as graph mining and graph representation learning in the future.

ACKNOWLEDGEMENT

This work is supported by the National Key R&D Program of China under grant 2021ZD0114501.

REFERENCES

[1] L. Akoglu, H. Tong, and D. Koutra, "Graph based anomaly detection and description: a survey," *Data mining and knowledge discovery*, vol. 29, no. 3, pp. 626–688, 2015.

[2] Y. Dou, Z. Liu, L. Sun, Y. Deng, H. Peng, and P. S. Yu, "Enhancing graph neural network-based fraud detectors against camouflaged fraudsters," in *Proceedings of the 29th ACM International Conference on Information & Knowledge Management*, 2020, pp. 315–324.

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

[4] K. Ding, J. Li, R. Bhanushali, and H. Liu, "Deep anomaly detection on attributed networks," in *Proceedings of the 2019 SIAM International Conference on Data Mining*. SIAM, 2019, pp. 594–602.

[5] L. Gutiérrez-Gómez, A. Bovet, and J.-C. Delvenne, "Multi-scale anomaly detection on attributed networks," in *Proceedings of the AAAI conference on artificial intelligence*, vol. 34, no. 01, 2020, pp. 678–685.

[6] G. Pang, C. Shen, L. Cao, and A. V. D. Hengel, "Deep learning for anomaly detection: A review," *ACM Computing Surveys (CSUR)*, vol. 54, no. 2, pp. 1–38, 2021.

[7] C. Hou, S. He, and K. Tang, "Rosane: Robust and scalable attributed network embedding for sparse networks," *Neurocomputing*, vol. 409, pp. 231–243, 2020.

[8] L. Akoglu, H. Tong, and D. Koutra, "Graph based anomaly detection and description: a survey," *Data mining and knowledge discovery*, vol. 29, no. 3, pp. 626–688, 2015.

[9] K. Liu, Y. Dou, Y. Zhao, X. Ding, X. Hu, R. Zhang, K. Ding, C. Chen, H. Peng, K. Shu *et al.*, "Bond: Benchmarking unsupervised outlier node detection on static attributed graphs," in *Thirty-sixth Conference on Neural Information Processing Systems Datasets and Benchmarks Track*, 2022.

[10] H. Tong and C.-Y. Lin, "Non-negative residual matrix factorization with application to graph anomaly detection," in *Proceedings of the 2011 SIAM International Conference on Data Mining*. SIAM, 2011, pp. 143–153.

[11] D. Chakrabarti, "Autopart: Parameter-free graph partitioning and outlier detection," in *European conference on principles of data mining and knowledge discovery*. Springer, 2004, pp. 112–124.

[12] D. Koutra, T.-Y. Ke, U. Kang, D. H. P. Chau, H.-K. K. Pao, and C. Faloutsos, "Unifying guilt-by-association approaches: Theorems and fast algorithms," in *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*. Springer, 2011, pp. 245–260.

[13] S. Thudumu, P. Branch, J. Jin, and J. J. Singh, "A comprehensive survey of anomaly detection techniques for high dimensional big data," *Journal of Big Data*, vol. 7, no. 1, pp. 1–30, 2020.

[14] Z. Wu, S. Pan, F. Chen, G. Long, C. Zhang, and S. Y. Philip, "A comprehensive survey on graph neural networks," *IEEE transactions on neural networks and learning systems*, vol. 32, no. 1, pp. 4–24, 2020.

[15] S. Pouyanfar, S. Sadiq, Y. Yan, H. Tian, Y. Tao, M. P. Reyes, M.-L. Shyu, S.-C. Chen, and S. S. Iyengar, "A survey on deep learning: Algorithms, techniques, and applications," *ACM Computing Surveys (CSUR)*, vol. 51, no. 5, pp. 1–36, 2018.

[16] Y. Liu, Z. Li, S. Pan, C. Gong, C. Zhou, and G. Karypis, "Anomaly detection on attributed networks via contrastive self-supervised learning," *IEEE transactions on neural networks and learning systems*, vol. 33, no. 6, pp. 2378–2392, 2021.

[17] K. Ding, J. Li, N. Agarwal, and H. Liu, "Inductive anomaly detection on attributed networks," in *Proceedings of the Twenty-Ninth International Conference on International Joint Conferences on Artificial Intelligence*, 2021, pp. 1288–1294.

[18] H. Fan, F. Zhang, and Z. Li, "Anomalydae: Dual autoencoder for anomaly detection on attributed networks," in *ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2020, pp. 5685–5689.

[19] Z. Xu, X. Huang, Y. Zhao, Y. Dong, and J. Li, "Contrastive attributed network anomaly detection with data augmentation," in *Pacific-Asia Conference on Knowledge Discovery and Data Mining*. Springer, 2022, pp. 444–457.

[20] K. Ding, J. Li, and H. Liu, "Interactive anomaly detection on attributed networks," in *Proceedings of the twelfth ACM international conference on web search and data mining*, 2019, pp. 357–365.

[21] X. Yuan, N. Zhou, S. Yu, H. Huang, Z. Chen, and F. Xia, "Higher-order structure based anomaly detection on attributed networks," in *2021 IEEE International Conference on Big Data (Big Data)*. IEEE, 2021, pp. 2691–2700.

[22] M. Jin, Y. Liu, Y. Zheng, L. Chi, Y.-F. Li, and S. Pan, "Anemone: graph anomaly detection with multi-scale contrastive learning," in *Proceedings of the 30th ACM International Conference on Information & Knowledge Management*, 2021, pp. 3122–3126.

[23] Y. Zheng, M. Jin, Y. Liu, L. Chi, K. T. Phan, and Y.-P. P. Chen, "Generative and contrastive self-supervised learning for graph anomaly detection," *IEEE Transactions on Knowledge and Data Engineering*, 2021.

[24] F. Liu, X. Ma, J. Wu, J. Yang, S. Xue, A. Beheshti, C. Zhou, H. Peng, Q. Z. Sheng, and C. C. Aggarwal, "Dagad: Data augmentation for graph anomaly detection," *arXiv preprint arXiv:2210.09766*, 2022.

[25] S. Kaufman, S. Rosset, C. Perlich, and O. Stitelman, "Leakage in data mining: Formulation, detection, and avoidance," *ACM Transactions on Knowledge Discovery from Data (TKDD)*, vol. 6, no. 4, pp. 1–21, 2012.

[26] D. W. Hosmer, S. Lemeshow, and R. Sturdivant, "Area under the receiver operating characteristic curve," *Applied Logistic Regression. Third ed: Wiley*, pp. 173–182, 2013.

[27] T. N. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," *arXiv preprint arXiv:1609.02907*, 2016.

[28] P. Velickovic, G. Cucurull, A. Casanova, A. Romero, P. Lio, and Y. Bengio, "Graph attention networks," *stat*, vol. 1050, p. 20, 2017.

[29] W. Hamilton, Z. Ying, and J. Leskovec, "Inductive representation learning on large graphs," *Advances in neural information processing systems*, vol. 30, 2017.

[30] K. X. W. H. J. Leskovec and S. Jegelka, "How powerful are graph neural networks," *ICLR. Keyulu Xu Weihua Hu Jure Leskovec and Stefanie Jegelka*, 2019.

[31] B. Weisfeiler and A. Leman, "The reduction of a graph to canonical form and the algebra which appears therein," *NTI, Series*, vol. 2, no. 9, pp. 12–16, 1968.

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

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

[34] S. Bandyopadhyay, S. V. Vivek, and M. Murty, "Outlier resistant unsupervised deep architectures for attributed network embedding," in *Proceedings of the 13th international conference on web search and data mining*, 2020, pp. 25–33.

[35] J. Gilmer, S. S. Schoenholz, P. F. Riley, O. Vinyals, and G. E. Dahl, "Neural message passing for quantum chemistry," in *International conference on machine learning*. PMLR, 2017, pp. 1263–1272.

[36] W.-L. Chiang, X. Liu, S. Si, Y. Li, S. Bengio, and C.-J. Hsieh,

"Cluster-gcn: An efficient algorithm for training deep and large graph convolutional networks," in *Proceedings of the 25th ACM SIGKDD international conference on knowledge discovery & data mining*, 2019, pp. 257–266.

[37] H. Zeng, M. Zhang, Y. Xia, A. Srivastava, A. Malevich, R. Kannan, V. Prasanna, L. Jin, and R. Chen, "Decoupling the depth and scope of graph neural networks," *Advances in Neural Information Processing Systems*, vol. 34, pp. 19 665–19 679, 2021.

[38] D. Lim, F. Hohne, X. Li, S. L. Huang, V. Gupta, O. Bhalerao, and S. N. Lim, "Large scale learning on non-homophilous graphs: New benchmarks and strong simple methods," *Advances in Neural Information Processing Systems*, vol. 34, pp. 20 887–20 902, 2021.