Towards Automated Self-Supervised Learning for Truly Unsupervised Graph Anomaly Detection

Anonymous Author(s)

ABSTRACT

Self-supervised learning (SSL) is an emerging paradigm that exploits supervisory signals generated from the data itself, eliminating the need for human-annotated labels. This approach is especially beneficial for unsupervised tasks such as graph anomaly detection, where labeled anomalies are scarce or expensive to obtain. As a result, many recent studies have leveraged SSL to conduct graph anomaly detection, yielding outstanding detection performance. Our empirical findings, however, reveal that three important factors can substantially impact detection performance across datasets: 1) the specific SSL strategy employed; 2) the tuning of the strategy's hyperparameters; and 3) the allocation of combination weights when using multiple strategies. We find that most SSL-based graph anomaly detection methods circumvent these issues by arbitrarily or selectively (i.e., guided by label information) choosing SSL strategies, hyperparameter settings, and combination weights. While an arbitrary choice may lead to subpar performance, using label information in an unsupervised setting is label information leakage and leads to severe overestimation of a method's performance. Leakage has been criticized as "one of the top ten data mining mistakes" [25, 47], yet many recent studies on SSL-based graph anomaly detection have been using label information to select hyperparameters. To mitigate this issue, we propose to use an internal evaluation strategy (with theoretical analysis) to select hyperparameters in SSL for unsupervised anomaly detection. Our method does not need any ground-truth label information, which we consider a critical step towards to achieving truly unsupervised graph anomaly detection. We perform extensive experiments using 10 recent SSL-based graph anomaly detection algorithms on various benchmark datasets, demonstrating both the prior issues with hyperparameter selection and the effectiveness of our proposed strategy.

CCS CONCEPTS

• **Information systems** → **Data Mining, Anomaly Detection**.

KEYWORDS

Graph Anomaly Detection, Self-Supervised Learning, Automated Machine Learning, Graph Neural Networks, Label Leakage

ACM Reference Format:

Anonymous Author(s). 2024. Towards Automated Self-Supervised Learning for Truly Unsupervised Graph Anomaly Detection. In *Proceedings of SIGKDD 2024 (Submitted to KDD '24).* ACM, New York, NY, USA, 19 pages. https://doi.org/XXXXXXX.XXXXXXX

Submitted to KDD '24, August 25–29, 2024, Barcelona, Spain

ACM ISBN 978-1-4503-XXXX-X/18/06…$15.00
https://doi.org/XXXXXXX.XXXXXXX

1 INTRODUCTION

Graph anomaly detection (GAD) refers to the tasks of identifying anomalous graph objects—such as nodes, edges or sub-graphs—in an individual graph, or identifying anomalous graphs from a set of graphs [44]. GAD has numerous successful applications, e.g., in finance fraud detection [46], fake news detection [63], system fault diagnosis [34], and network intrusion detection [14]. In this paper, we focus on unsupervised node anomaly detection on static attributed graphs, namely identifying which nodes in a static attributed graph are anomalous. Recently, Graph Neural Networks (GNNs) have become prevalent in detecting node anomalies in graphs and have shown promising performance [26]. Specifically, GNNs can learn an embedding for each node by considering both the node attributes and the graph topological information, enabling them to capture and exploit complex patterns for anomaly detection.

Like with other neural networks, the high performance of GNNs is typically achieved at the cost of a substantial volume of labelled data. However, the process of labelling graphs is often a laborious and time-consuming effort, necessitating domain-specific expertise. For these reasons, GAD is preferably tackled in an unsupervised manner, without relying on any ground-truth labels. Self-supervised learning (SSL) has emerged as a promising unsupervised learning technique on graphs [39], and recent studies have shown its usefulness for node anomaly detection [3, 12, 22, 36, 40, 64, 70, 85].

Graph SSL can be roughly divided into *generative*, *contrastive*, and *predictive* methods [61]. First, *generative* methods such as DOMINANT [9], GUIDE [70], and AnomalyDAE [12] aim to detect graph anomalies by reconstructing ('generating') the adjacency matrix and/or the node attribute matrix. Next, *contrastive* methods such as CoLA [40], ANEMONE [22], GRADATE [11], and Sub-CR [75] train a graph encoder to pull positive pairs closer while pushing negative pairs away in the embedding space. The nodes with relatively large contrastive loss values are deemed anomalies. Finally, *predictive* methods such as SL-GAD [85] try to predict node properties using its local context (e.g., a subgraph), and nodes with large prediction errors are considered anomalies.

Contrastive learning is arguably the most successful SSL strategy for graphs [62]. Most contrastive graph learning methods consist of two main modules: 1) a *data augmentation module* that generates augmented data by operations such as edge dropping, node attribute masking, node addition, subgraph sampling, and/or graph diffusion. The augmented view of an instance is generally regarded as a positive pair with the original instance; and 2) a *contrastive learning module* that contrasts positive pairs (and often involves negative pairs) at different levels, such as node-node contrast, node-subgraph contrast, and subgraph-subgraph contrast.

Although SSL-based graph anomaly detection has been successful, using it in practice is often not straightforward. The most important reason for this is that most methods require a large number of choices to be made, leading to three challenges: **C1. How should we select appropriate data augmentation functions?**; **C2. How should we choose appropriate values for hyperparameters (HPs) of a given augmentation function?** (e.g., subgraph size in a subgraph sampling function, or the proportion of edges to drop in an edge dropping function); and **C3. How to combine the contrast losses at different levels?** (i.e., how to set their combination weights?). Further, a recent study [85] shows that combining multiple SSL strategies for GAD can achieve better performance than using a single SSL strategy. This leads to the fourth challenge: **C4. How should we combine different SSL strategies?** (i.e., how to set the combination weights of different SSL loss functions?).

Previous work [4, 67, 69] showed that the choice of SSL strategie(s) and hyperparameter values can strongly impact performance. In a *supervised* setting, these choices can be systematically and rigorously made by using separate labelled data for validation. In an *unsupervised setting* such as anomaly detection, however, one should assume that no labels are available even for hyperparameter tuning. In our extensive literature study, we found that existing SSL-based GAD methods typically either 1) arbitrarily choose settings or 2) do use labelled data, corroborating the findings in [67].

In the former case, practitioners typically heuristically select an augmentation function (C1) and fix its associated HPs (C2) across all datasets, and set the combination weights all equal to 1 or other fixed values (for C3 and Q4). Although this approach is not flawed, it is likely to result in suboptimal detection performance: graphs from different domains usually have different properties [79], implying that the optimal SSL strategy is in general data-dependent [4, 69].

In the latter case, practitioners pick the optimal combination weights and other hyperparameter values following a 'hyperparameters sensitivity analysis' using labelled data. By using ground-truth labels on test data to check model performance with different hyperparameter values and using that to select the best model, however, *label leakage* occurs. That is, information about the target of a data mining problem is used for learning/selecting model, while this information should not be legitimately accessible for learning purposes [25, 47]. Specifically, label information should never be used (whether implicitly or explicitly) in an unsupervised learning scenario. As shown in Figure 1, label leakage leads to huge overestimation of the model's performance, which is also corroborated in [38] by comparing the max and average performance with different hyperparameter configurations.

The reason that hyperparameter values are often chosen either arbitrarily or using label information is probably that it is challenging to construct an internal evaluation strategy for anomaly detection without using any labels. There have been some research efforts aimed at automating graph SSL though. For instance, JOAO [68] aims to automatically combine several predefined graph augmentations via learning a sampling distribution, where the augmentations themselves are not learnable. Meanwhile, AD-GCL [55] uses learnable edge dropping augmentation and AutoGCL [66] proposes a learnable graph view generator that learns a probability distribution over node-level augmentations, which can well preserve the semantic labels of graphs for graph-level tasks. However,

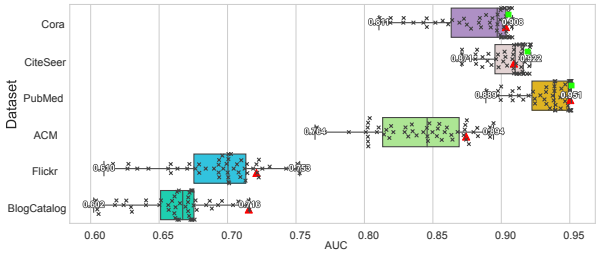

Figure 1: Large performance variations (here measured by AUC) over different hyperparameter configurations for ANEMONE [22] on various benchmark datasets. Using label data and only reporting the best possible performance leads to severe overestimation of model performance. For instance, the green squares on Cora, CiteSeer, and PubMed are reported by [22] (the other datasets were not used). Similar results are observed for other algorithms (see Appendix C for details). The red triangles represent the results obtained by our internal evaluation strategy, showing its potential for automating truly unsupervised anomaly detection.

all these automated graph augmentation methods are agnostic to the downstream tasks, making the learned graph embeddings suboptimal for a specific downstream task, namely anomaly detection in our case. Additionally, these methods are specifically designed for certain SSL frameworks, and it is non-trivial (if at all possible) to extend them to the general SSL framework. Moreover, these automated SSL strategies are computationally expensive, rendering them impractical in real-world applications.

As an initial step towards mitigating this long-standing but neglected issue, we propose a lightweight and plug-and-play approach dubbed *AutoGAD*, to automate SSL for truly unsupervised graph anomaly detection. Specifically, *AutoGAD* leverages a so-called internal evaluation strategy [43], without relying on any ground-truth labels (whether explicitly or implicitly), to select optimal combination weights and/or SSL-specific hyperparameter values. Moreover, we theoretically analyse the internal evaluation strategy to prove why it is effective and empirically demonstrate this.

Overall, our main contributions can be summarised as follows:

- We raise renewed awareness to the *label information leakage* issue, which is critical but often overlooked in the unsupervised GAD field;
- Although there exists a plethora of graph SSL methods and GAD approaches, we are the first to investigate automated SSL specifically for unsupervised GAD;
- We propose a lightweight, plug-and-play approach to automate SSL for truly unsupervised GAD and provide a theoretical analysis;
- Extensive experiments are conducted using 10 state-of-the-art SSL-based GAD algorithms on 6 benchmark datasets, demonstrating the effectiveness of our approach.

2 RELATED WORK

Our work is related to node anomaly detection on static attributed graphs, self-supervised learning for graph anomaly detection, automated self-supervised learning, and automated anomaly detection.

2.1 Anomaly Detection on Attributed Graphs

Early methods for node anomaly detection in static attributed graphs, such as AMEN [50], Radar [29], and Anomalous [49], are not based on deep learning. These methods work well on low-dimensional attributed graphs, but their performance is limited on complex graphs with high-dimensional node attributes.

Recently, deep learning-based methods, including DOMINANT [9], AnomalyDAE [12], and GUIDE [70], have been proposed for GAD. These methods usually employ graph autoencoders to encode nodes followed by decoders to reconstruct the adjacency matrix and/or node attributes. As a result, nodes with large reconstruction errors are considered anomalies. Despite their superior performance to non-deep learning methods, these reconstruction-based methods still suffer from sub-optimal performance, as reconstruction is a generic unsupervised learning objective. Besides, these methods require the full attribute and adjacency matrices as model input, making them unsuitable or even impossible for large graphs.

2.2 Self-Supervised Learning for Graph Anomaly Detection

Graph SSL aims to learn a model by using supervision signals generated from the graph itself, without relying on human-annotated labels [39]. It has achieved promising performance on typical graph mining tasks such as representation learning [21] and graph classification [73]. Liu et al. [40] first applied SSL to the GAD problem. Their proposed method CoLA performs single scale comparison (node-subgraph) for anomaly detection. However, ANEMONE [22] argues that modeling the relationships in a single contrastive perspective leads to limited capability of capturing complex anomalous patterns. Hence, they propose additional node-node contrast. Additionally, GRADATE [11] and M-MAG [41] combines various multi-contrast objectives, namely node-node, node-subgraph, and subgraph-subgraph contrasts for node anomaly detection. To achieve better performance, SL-GAD [85] combines multi-view contrastive learning and generative attribute regression, while Sub-CR [75] combines multi-view contrastive learning and graph autoencoder. Finally, CONAD [64] considers both contrastive learning and generative reconstruction for better node anomaly detection.

2.3 Automated Self-Supervised Learning

Seminal work on *automated data augmentation for images* [6, 52] was followed by work improving [6] via faster searching mechanisms [7, 20, 35] or advanced optimization methods [19, 32, 77].

In the context of *automated data augmentation for graphs*, related work exists on graph representation learning [18, 23, 55, 62, 66, 68], node classification [54, 78], and graph-level classification [42, 66, 71]. For example, JOAO [68] learns the sampling distribution of a set of predefined graph augmentations. AD-GCL [55] designs a learnable edge dropping augmentation and employs adversarial training strategy, and AutoGCL [66] proposes a learnable graph view generator that learns a probability distribution over the node-level augmentations. Further, [42] augments graph data samples, while [71] perturbs the representation vector. However, these methods focus on other typical graph learning tasks and it is unclear how to use them for unsupervised GAD.

2.4 Automated Anomaly Detection

Recent studies [1, 10, 80, 82] pointed out that unsupervised anomaly detection methods tend to be highly sensitive to the values of their hyperparameters (HPs). For example, [82] shows that a 10x performance difference is observed for LOF [2] by changing the number of nearest neighbours. Even more, [10] indicates that deep anomaly detection methods suffer more from such HP sensitivity issues. Concretely, [80] demonstrates that RAE [86] exhibits a 37x performance difference with different HPs configurations.

To tackle this issue, automated HP tuning and model selection for unsupervised anomaly detection has received increasing but insufficient attention; Bahri et al. [1] present an overview. Inspired by [1, 80], we subdivide existing approaches into two categories: 1) *supervised evaluation* methods which require ground-truth labels although anomaly detection algorithms are unsupervised. Methods include PyODDS [33], TODS [28], AutoOD [31], and AutoAD [30]; 2) *unsupervised evaluation* methods which do not require ground-truth labels. They include 2.1) randomly selecting an HP configuration; 2.2) selecting an HP configuration via an internal evaluation strategy [15, 45, 51, 81]; 2.3) averaging the outputs of a set of randomly selected HP configurations [59]; and 2.4) meta-learning based methods [74, 80, 83]. However, existing automated anomaly detection methods are primarily designed for non-graph data.

3 PROBLEM STATEMENT

We utilise lowercase letters, bold lowercase letters, uppercase letters, and calligraphic fonts to represent scalars (x), vectors ($\mathbf{x}$), matrices ($\mathbf{X}$), and sets ($\mathcal{X}$), respectively.

Definition 3.1 (Attributed Graph). We denote an attributed graph as $\mathcal{G} = \{\mathcal{V}, \mathcal{E}, \mathbf{X}\}$, where $\mathcal{V} = \{v_1, ..., v_n\}$ is the set of nodes. Besides, $\mathcal{E} = \{e_{ij}\}_{i,j \in \{1,...,n\}}$ is the set of edges, where $e_{ij} = 1$ if there exists an edge between v_i and v_j and $e_{ij} = 0$ otherwise. Moreover, $\mathbf{X} \in \mathbb{R}^{n \times d}$ represents the node attribute matrix, where the i-th row vector $\mathbf{x}_i$ means the node attribute of v_i.

Formally, we consider unsupervised node detection on attributed graphs (dubbed GAD hereafter), which is defined as follows:

Problem 1 (Node Anomaly Detection on Attributed Graph). *Given an attributed graph as $\mathcal{G} = \{\mathcal{V}, \mathcal{E}, \mathbf{X}\}$, we aim to learn an anomaly scoring function $f(\cdot)$ that assigns an anomaly score $s = f(v_i)$ to each node v_i, with a higher score representing a higher degree of being anomalous. Next, the anomaly scores are used to rank the nodes such that the top-k nodes can be considered as anomalies.*

In this paper, we consider the *transductive unsupervised anomaly detection* setting: the graph containing both normal and abnormal nodes are given at the training stage. Node labels are not accessible during the training stage and they are only used for performance evaluation. Importantly, the labels of nodes are **not** (and should not be) used for HP tuning under this unsupervised setting.

Formally, we consider the hyperparameter optimisation problem for unsupervised graph anomaly detection (dubbed HPO for GAD):

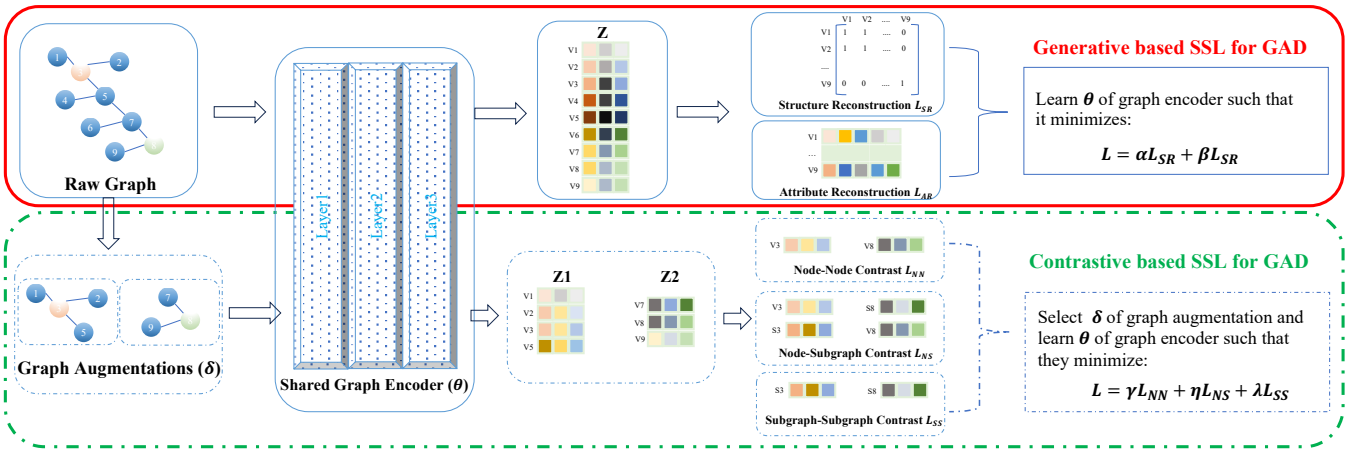

Figure 2: Self-supervised learning based graph anomaly detection methods can be subdivided into *generative based* methods and *contrastive based* methods. *Generative based* methods generally involves *graph structure reconstruction* and *node attributes reconstruction*. *Contrastive based* methods usually consists of a *graph augmentation module* and a *contrastive learning module*.

Problem 2 (HPO for GAD). *Given a graph $\mathcal{G}$ without labels and a graph anomaly detection algorithm $f(\cdot)$ with hyperparameter space $\mathbf{\Lambda}$, we aim to identify a hyperparameter configuration $\boldsymbol{\lambda} \in \mathbf{\Lambda}$ such that the resulting model $f(\boldsymbol{\lambda})$ can achieve the best performance on $\mathcal{G}$. I.e., suppose $\boldsymbol{\lambda}$ consists of K different hyperparameters $\{\lambda_1, ..., \lambda_k, ..., \lambda_K\}$, where $\lambda_k \in \mathbf{\Lambda}_k$ can be discrete or continuous, we then aim to find*

$$\underset{\lambda_1 \in \mathbf{\Lambda}_1, ..., \lambda_k \in \mathbf{\Lambda}_k, ..., \lambda_K \in \mathbf{\Lambda}_K}{\arg\max} \; Metric\left[f(\lambda_1, ..., \lambda_k, ..., \lambda_K; \mathcal{G})\right], \quad (1)$$

where $Metric[\cdot]$ is a given performance metric.

4 SSL FOR UNSUPERVISED GAD

In this section, we first revisit existing self-supervised learning methods for "unsupervised" graph anomaly detection, followed by an analysis and experiments to showcase pitfalls in existing studies.

4.1 Existing SSL for "Unsupervised" GAD

Figure 2 shows how existing SSL based GAD methods can be divided into *generative* methods and *contrastive* methods.

That is, a *generative* method usually consists of two individual SSL tasks, namely 1.1) *structure reconstruction* that aims to reconstruct the adjacency matrix, and 1.2) *attribute reconstruction* that aims to reconstruct the node attribute matrix. On this basis, the attribute reconstruction error and the structure reconstruction error are combined to obtain an anomaly score, where higher reconstruction error indicates a higher degree of anomalousness.

Meanwhile, a *contrastive* method often consists of two modules: 2.1) *data augmentation* module, and 2.2) *contrastive learning* module. First, for each target node, the *data augmentation* module utilises one augmentation function $f(\boldsymbol{\delta})$ to produce augmented samples, which usually includes positive samples and negative samples. The scenario of using multiple augmentation functions can be obtained in a similar way. Second, three contrastive perspectives can be applied to contrast positive pairs and negative pairs: 2.2.1) *node-node contrast* that contrasts node embedding with node embedding, and 2.2.2) *node-subgraph contrast* that contrasts node embedding with subgraph embedding, and 2.2.3) *subgraph-subgraph contrast* that contrasts subgraph embedding with subgraph embedding.

4.2 Pitfalls in Existing Methods

In this subsection, we revisit existing SSL-based unsupervised GAD methods by checking the following three aspects for each method:

- Which SSL framework does the method employ: *generative*, *contrastive*, or both?
- How many SSL-specific hyperparameters are involved? (E.g., combination weights and others.)
- How are values for key SSL hyperparameters chosen? (E.g., the ratio of node attribute masking or dropping edges, and the combination weights of multiple loss functions?)

By doing so, we point out that these studies have noticeable pitfalls. More importantly, we perform experiments to show that the high performance that these methods claim to achieve is often strongly overestimated due to label leakage issues (cf. Table 1).

Due to space constraints, we revisit three representative SSL-based GAD algorithms in the main paper: a contrastive method (ANEMONE [22]), a generative method (AnomalyDAE [12]), and a combined contrastive and generative method (SL-GAD [85]).

4.2.1 ANEMONE [22]. A *contrastive* method for unsupervised GAD.

Graph Augmentation Module A single graph augmentation operation is used, namely Random Ego-Nets generation with a fixed size K. Specifically, taking the target node as the center, they employ RWR [57] to generate two different subgraphs as ego-nets with a fixed size K. This results in one critical HP, namely K.

Contrast Learning Module Two contrast perspectives are considered: 1) node-node contrast between the embedding of a masked target node within the ego-net and the embedding of the original node, leading to loss term $\mathcal{L}_{NN}$, and 2) node-subgraph contrast within each view, leading to loss term $\mathcal{L}_{NS}$. These loss terms are combined as $\mathcal{L} = (1-\alpha)\mathcal{L}_{NN} + \alpha\mathcal{L}_{NS}$, where $\alpha \in [0, 1]$ is the trade-off HP, giving one more critical HP, namely α.

HPs Sensitivity & Tuning By using ground-truth label information, they heuristically set α to 0.8, 0.6, 0.8 on Cora, CiterSeer, and PubMed respectively, and report the corresponding results. The setting of K is not studied, and is set to 4 for all datasets.

4.2.2 AnomalyDAE [12]. A *generative* method using autoencoders (based on GNNs) for unsupervised GAD.

Generative Framework AnomalyDAE consists of two components: 1) an attribute autoencoder to reconstruct the node attributes, where the encoder consists of two non-linear feature transform layers and the decoder is simply a dot product operation. This leads to the loss term $\mathcal{L}_A$, and $\mathcal{L}_A$ is associated with a penalty HP η; and 2) a structure autoencoder to reconstruct the structure, where the encoder is based on GAT [58] and the decoder is a dot product operation followed by a *sigmoid* function. This leads to the loss term $\mathcal{L}_S$, and $\mathcal{L}_S$ is associated with a penalty HP θ.

Their overall optimisation objective is then defined as $\mathcal{L} = \alpha\mathcal{L}_S + (1-\alpha)\mathcal{L}_A$, where $\alpha \in (0, 1)$ balances the two objectives.

HPs Sensitivity & Tuning The paper finds that the AUC usually increases first and then decreases with the increase of α. However, the specific value of α on each dataset is selected using label information. The HPs (α, η, θ) are heuristically set as $(0.7, 5, 40)$, $(0.9, 8, 90)$, $(0.7, 8, 10)$ on BlogCatalog, Flickr, and ACM respectively.

4.2.3 SL-GAD [85]. An unsupervised GAD method that combines both contrastive and generative objectives.

Contrastive Framework—Data Augmentation Module The method uses a single graph augmentation operation, namely Random Ego-Nets generation with a fixed size K. Specifically, taking the target node as the center, RWR [57] is used to generate two different subgraphs as ego-nets with a fixed size K, where K controls the radius of the surrounding contexts. This gives one critical HP for graph augmentation, namely K.

Contrastive Framework—Contrast Learning Module The Multi-View Contrastive Learning module compares the similarity between a node embedding and the embedding of sampled sub-graphs in augmented views (namely node-subgraph contrast), leading to loss terms $\mathcal{L}_{con,1}$ and $\mathcal{L}_{con,2}$. Combining those leads to contrastive objective $\mathcal{L}_{con} = \frac{1}{2}(\mathcal{L}_{con,1} + \mathcal{L}_{con,2})$.

Generative Framework The Generative Attribute Regression module reconstructs node attributes, with the aim to achieve node-level discrimination. Specifically, they minimise the Mean Square Error between the target node's original and reconstructed attributes in augmented views, leading to loss terms $\mathcal{L}_{gen,1}$ and $\mathcal{L}_{gen,2}$. Combining those with equal weights leads to generative objective $\mathcal{L}_{gen} = \frac{1}{2}(\mathcal{L}_{gen,1} + \mathcal{L}_{gen,2})$.

The overall optimisation objective is then defined as $\mathcal{L} = \alpha\mathcal{L}_{con} + \beta\mathcal{L}_{gen}$, where $\alpha, \beta \in (0, 1]$ are trade-off HPs to balance the importance of the two SSL objectives.

HPs Sensitivity & Tuning The authors conducted a sensitive analysis and found that: 1) the performance first increases and then decreases with the increase of K. For efficiency considerations, they heuristically set the sampled subgraph size $K = 4$ for all datasets; 2) they heuristically fix $\alpha = 1$ for all datasets as they found that this achieves good performance on most datasets (with the help of label information); and 3) the selection of β is highly dependent on the specific dataset. Hence, they "fine-tune" the value of β for each dataset via selecting β from $\{0.2, 0.4, 0.6, 0.8, 1.0\}$ using labels.

4.2.4 Other SSL-based GAD methods. Due to space constraints, the analyses of other SSL-based GAD methods, including CoLA [40] , GRADATE [11], Sub-CR [75], CONAD [64], DOMINANT [9], GUIDE [70], and GAAN [5], are given in Appendix B. These methods are all representatives of recent advancements in using SSL to conduct unsupervised graph anomaly detection, and have yielded outstanding detection performance. Likewise, however, these methods also exhibit pitfalls with regard to hyperparameter tuning, similar to those of ANEMONE [22], AnomalyDAE [12], and SL-GAD [85].

4.3 Sensitivity Analysis

After revisiting recent SSL-based unsupervised GAD methods, we now empirically investigate their sensitivity to SSL-related HPs in a systematic way. More concretely, we report their performance variations in terms of RUC-AUC values under different hyperparameter configurations (see Section 6 for experiment settings).

As shown in Figure 1, for a typical run with different hyperparameter configurations, the performance of ANEMONE [22] can vary strongly on each of the six datasets. Other SSL-based GAD algorithms exhibit similar behavior; extensive results and analysis are deferred to Appendix C for space reasons.

For an in-depth yet compact analysis, Table 1 presents average results over five independent runs when varying SSL-related hyperparameter values. Specifically, CoLA [40], CONAD [64], DOMINANT [9] and GUIDE [70] demonstrate moderate performance variations (namely between 2.8% and 5.0% on average). Meanwhile, ANEMONE [22], GRADATE [11], SL-GAD [85], Sub-CR [75], AnomalyDAE [12], and GAAN [5] suffer from large performance variations (namely between 9.9% and 23.4% on average). From Subsection 4.2 and Appendix B, we see that the results reported in existing papers are often obtained by manually tuned HPs (in a post-hoc way with label information), thereby leading to strongly overestimated performance for real-world applications where labels are not accessible. To mitigate this severe issue, we propose *AutoGAD*, a method for automating hyperparameter selection in SSL for GAD and achieving truly unsupervised graph anomaly detection. Importantly, *AutoGAD* does not need any ground-truth labels.

5 AUTOGAD: USING INTERNAL EVALUATION TO AUTOMATE SSL FOR GAD

Our proposed approach, called AutoGAD, consists of two parts: 1) an unsupervised performance metric, and 2) an effective search method. Importantly, and as mentioned before, the chosen performance metric—denoted $Metric[\cdot]$ in Equation 1—should not use any ground-truth label information, simply because this is not available in a truly unsupervised setting. We therefore propose to utilise an internal evaluation strategy, which will be elucidated later. Next, given the impracticality of evaluating an infinite number of configurations for continuous hyperparameter domains, another challenge is the efficient exploration of the search space. Section 5.2 describes a straightforward approach using discretisation and grid search that works well in practice, as shown in the next section.

Table 1: Performance variation, quantified as $\frac{\max(AUC)-\min(AUC)}{\max(AUC)}$, under different hyperparameter settings on six benchmark datasets. Results are the mean of five independent runs, each with a unique random seed. OOM means out of memory, while OOR indicates that runtime exceeded a 7-day limit for a single trial. Cells highlighted in gray indicate persistent underfitting of algorithms despite reaching the maximum number of allowed training epochs (e.g., their loss values change by only 10^{-2} after 400 epochs); we exclude these cells from further analysis. Please refer to Section 6 for the experiment setup.

Dataset	CoLA	ANEMONE	GRADATE	SL-GAD	Sub-CR	CONAD	DOMINANT	A-DAE	GUIDE	GAAN
Cora [53]	1.1%	8.9%	6.9%	17.4%	15.1%	5.8%	5.1%	19.1%	4.8%	26.7%
CiteSeer [53]	1.7%	6.6%	14.1%	16.2%	8.3%	7.0%	6.0%	25.3%	4.8%	19.5%
PubMed [53]	1.6%	6.3%	OOM	19.5%	OOM	2.3%	1.9%	23.8%	1.8%	17.2%
ACM [56]	4.2%	11.3%	OOM	17.7%	OOM	0.2%	0%	20.8%	0%	23.0%
Flickr [72]	3.3%	16.9%	OOR	16.3%	9.8%	OOM	10.2%	23.6%	8.5%	16.1%
BlogCataLog [72]	4.7%	16.8%	OOR	23.4%	6.3%	OOM	0%	14.3%	0%	14.9%
Average	2.8%	11.1%	10.5%	18.4%	9.9%	5.0%	4.3%	23.4%	3.8%	19.5%

5.1 Internal Evaluation Strategy

The intuition behind the internal evaluation strategy that we use is to measure the similarity of anomaly scores within the same predicted anomaly class and the dissimilarity between anomaly scores across different predicted classes (i.e., 'anomaly' or 'no anomaly'). As we will prove later, optimizing the resulting measure is equivalent to simultaneously minimizing the false positive rate and the false negative rate. In this way, we aim to evaluate and optimize the performance of the anomaly detector under different SSL configurations *without having to rely on any ground-truth labels.*

5.1.1 Contrast Score Margin. The metric that we use is Contrast Score Margin [65], which was introduced before but not for graph anomaly detection, and is defined as

$$T(f) = \frac{\hat{\mu}_{\mathbf{O}} - \hat{\mu}_{\mathbf{I}}}{\sqrt{\frac{1}{k}(\hat{\delta}^2_{\mathbf{O}} + \hat{\delta}^2_{\mathbf{I}})}}, \tag{2}$$

where $\hat{\mu}_{\mathbf{O}}$ and $\hat{\delta}^2_{\mathbf{O}}$ denote the average and variance of the anomaly scores of the k predicted anomalous objects ($\hat{\mathbf{O}}$), respectively. Moreover, $\hat{\mu}_{\mathbf{I}}$ and $\hat{\delta}^2_{\mathbf{I}}$ represent the average and variance of the anomaly scores of the k predicted normal objects ($\hat{\mathbf{I}}$) with the highest scores, respectively. Intuitively, the metric focuses on the k predicted normal objects that are most similar to the k predicted anomalous objects, and aims to measure the margin of the anomaly scores between them. It only takes linear time with respect to N to compute.

5.1.2 Analysis. We now analyze why the internal evaluation metric Contrast Score Margin should work for our purposes.

Theorem 5.1 (Minimizing False Positives and Negatives). *For an anomaly detector $f(\cdot)$ on dataset $\mathbf{X}$, assume the anomaly scores of the top k true anomalies ($\mathbf{O}$) have the expected value $\mu_{\mathbf{O}}$ and variance $\delta^2_{\mathbf{O}}$, and the anomaly scores of the top k true normal objects with the highest anomaly scores ($\mathbf{I}$) have the expected value $\mu_{\mathbf{I}}$ and variance $\delta^2_{\mathbf{I}}$, then maximizing T is equal to simultaneously minimizing the false positive rate and the false negative rate .*

Proof. According to *Cantelli's inequality*, which makes no assumptions on specific probability distributions, on the one hand, for $\mathbf{x} \in \mathbf{O}$ we have $P(f(\mathbf{x}) \leq \mu_{\mathbf{O}} - \alpha) \leq \frac{\delta^2_{\mathbf{O}}}{\delta^2_{\mathbf{O}}+\alpha^2}$, where $\alpha \geq 0$ is a small constant chosen based on a desired bound on the false negative. By replacing $\alpha = a\delta_{\mathbf{O}}$, we have $P(f(\mathbf{x}) \leq \mu_{\mathbf{O}} - a\delta_{\mathbf{O}}) \leq \frac{1}{1+a^2}$, which is the False Negative Bound. In other words, $f(x)$ has a maximum probability of $\frac{1}{1+a^2}$ to be less than $\mu_{\mathbf{O}} - a\delta_{\mathbf{O}}$.

On the other hand, for $\mathbf{y} \in \mathbf{I}$ we have $P(f(\mathbf{y}) \geq \mu_{\mathbf{I}} + \beta) \leq \frac{\delta^2_{\mathbf{I}}}{\delta^2_{\mathbf{I}}+\beta^2}$, where $\beta \geq 0$ is a small constant chosen based on a desired bound on the false positive. By replacing $\beta = b\delta_{\mathbf{I}}$, we have $P(f(\mathbf{y}) \geq \mu_{\mathbf{I}} + b\delta_{\mathbf{I}}) \leq \frac{1}{1+b^2}$, which is the False Positive Bound. In other words, $f(y)$ has a maximum probability of $\frac{1}{1+b^2}$ to be larger than $\mu_{\mathbf{I}} + b\delta_{\mathbf{I}}$.

Furthermore, $(\mu_{\mathbf{O}} - a\delta_{\mathbf{O}}) - (\mu_{\mathbf{I}} + b\delta_{\mathbf{I}}) = (\mu_{\mathbf{O}} - \mu_{\mathbf{I}}) - (b\delta_{\mathbf{I}} + a\delta_{\mathbf{O}})$. Hence, to ensure a small false positive rate and a small false negative rate, we want $\mu_{\mathbf{O}} - \mu_{\mathbf{I}}$ to be as large as possible while $b\delta_{\mathbf{O}} + a\delta_{\mathbf{I}}$ as small as possible. In fact, this is equivalent to optimize the Contrast Score Margin, i.e.,

$$T(f) = \frac{\mu_{\mathbf{O}} - \mu_{\mathbf{I}}}{\sqrt{\frac{1}{k}(\delta^2_{\mathbf{O}} + \delta^2_{\mathbf{I}})}}$$

Note that if an anomaly detector $f(\cdot)$ produces a perfect anomaly detection result, i.e., for any $\mathbf{x} \in \mathbf{O}$ and any $\mathbf{y} \in \mathbf{X} \setminus \mathbf{O}$, we have $f(\mathbf{x}) > f(\mathbf{y})$, then we will obtain $\mu_{\mathbf{O}} - \mu_{\mathbf{I}} > 0$. In another extreme, if $f(\cdot)$ produces a poor anomaly detection result, i.e., for all $\mathbf{x} \in \mathbf{O}$ and any $\mathbf{y} \in \mathbf{X} \setminus \mathbf{O}$, we have $f(\mathbf{x}) < f(\mathbf{y})$, then we will obtain $\mu_{\mathbf{O}} - \mu_{\mathbf{I}} < 0$. Meanwhile, if an anomaly detector $f(\cdot)$ produces a random result, i.e., for some $\mathbf{x} \in \mathbf{O}$ and any $\mathbf{y} \in \mathbf{X} \setminus \mathbf{O}$, we have $f(\mathbf{x}) < f(\mathbf{y})$, then we may obtain $\mu_{\mathbf{O}} - \mu_{\mathbf{I}} < 0$ or $\mu_{\mathbf{O}} - \mu_{\mathbf{I}} \approx 0$. □

5.1.3 Improvements and Remarks. In practice we observed that Equation 2 is not always stable. Possible reasons are that 1) the proportion of anomalies is usually very small (namely less than 5% in most datasets); and 2) the exact number of anomalies is generally not known (even for a dataset with injected anomalies, there may exist some natural samples that exhibit similar behaviours as anomalies). Therefore, we propose to modify Equation 2 as follows:

$$T(f) = \frac{\hat{\mu}_{\mathbf{O}} - \tilde{\mu}_{\mathbf{I}}}{\sqrt{\hat{\delta}^2_{\mathbf{O}} + \tilde{\delta}^2_{\mathbf{I}}}}, \tag{3}$$

where $\hat{\mu}_{\mathbf{O}}$ and $\hat{\delta}^2_{\mathbf{O}}$ denote the average and variance of the anomaly scores of the k predicted anomalous objects, respectively. Importantly, $\tilde{\mu}_{\mathbf{I}}$ and $\tilde{\delta}^2_{\mathbf{I}}$ represent the average and variance of the anomaly

scores of the remaining $n-k$ objects, respectively. This change should lead to more stable performance compared to using anomaly scores of the top-k predicted normal objects in Equation 2.

Moreover, to ensure the effectiveness of this internal evaluation strategy, we have to make sure that: 1) we use the same algorithm with different hyperparameter configurations; and 2) the scales of the loss values are approximately the same when combining multiple loss functions in the same algorithm. In other words, we should not directly use the strategy to select among different heterogeneous anomaly detection algorithms.

5.2 Discretisation and Grid Search

We first perform discretization of the continuous search space and then conduct grid search. More advanced strategies, such as SMBO-based optimisation [24] (see Appendix F), could be leveraged. However, using these advanced methods will often introduce additional hyperparameters (the tuning of which may be non-trivial) and may lead to increased runtimes, violating our intentions.

We discretize the hyperparameter space to make the overall search tractable. We assume given a GAD algorithm $f(\cdot)$ and its HPs $\boldsymbol{\lambda} \in \boldsymbol{\Lambda}$. Then, without loss of generality, we suppose $\boldsymbol{\lambda} = \{\lambda_1, \lambda_2, ..., \lambda_K\}$, where $\lambda_k \in \boldsymbol{\Lambda_k}$ for $k = 1, 2, ..., K$. Particularly, $\boldsymbol{\Lambda_k}$ is called a HP domain and it can be discrete or continuous. We discretize $\boldsymbol{\Lambda_k}$ into a discrete domain if it is originally continuous. In this way, we can get M different cross-products of $\{\lambda_1, \lambda_2, ..., \lambda_K\}$, dubbed $\boldsymbol{\lambda}_{search} = \{\boldsymbol{\lambda}_1, ..., \boldsymbol{\lambda}_M\}$ with $M = \prod_{k=1}^{K} |\boldsymbol{\Lambda_k}|$. We employ $f(\boldsymbol{\lambda}_m)$ to denote the detector $f(\cdot)$ with the m-th HP setting $\boldsymbol{\lambda}_m \in \boldsymbol{\lambda}_{search}$. Furthermore, by applying $f(\boldsymbol{\lambda}_m)$ on graph $\mathcal{G}$, we can obtain a vector of anomaly scores $\boldsymbol{s}_m(\mathcal{G}) = f(\boldsymbol{\lambda}_m; \mathcal{G})$. By inputting $\boldsymbol{s}_m(\mathcal{G})$ into Equation 3, we can yield a score $t_m(\mathcal{G}) = T(\boldsymbol{s}_m(\mathcal{G}))$ by using the internal evaluation function $T(\cdot)$ as an unsupervised performance metric on the obtained anomaly scores vector.

After obtaining the discretized search space, we apply grid search to find the hyperparameter configuration that maximizes $T(\cdot)$.

6 EXPERIMENTS

We aim to answer the following research questions (RQ):

- **RQ1** How sensitive are existing SSL-based GAD methods to the values of their hyperparameters?
- **RQ2** How effective is *AutoGAD* in tuning SSL-related hyperparameter values for these methods?

We describe the experiment settings, including the datasets, baselines, evaluation metrics, and software and hardware used, which is followed by the experiment results and their interpretation.

6.1 Datasets

We use three popular citation networks, namely Cora, Citeseer, and Pubmed [53] with injected anomalies, one social network Flickr (less homophily) with injected anomalies, ACM as well as BlogCataLog with injected anomalies. Particularly, we follow the methods used by ANEMONE [22] and CoLA [40] to inject structure and contextual anomalies. Note that [38] has slightly modified this injection procedure. The resulting datasets are summarized in Table 2.

Table 2: Summary of datasets, where anomalies are synthetically injected by following existing methods [22, 40].

Dataset	#Nodes	#Edges	#Attributes	#Anomalies
Cora [53]	2708	11060	1433	138(5.1%)
CiteSeer [53]	3327	4732	3703	150(4.5%)
PubMed [53]	19717	44338	500	150(2.5%)
ACM [56]	16484	71980	8337	600(3.6%)
BlogCataLog [72]	5196	171743	8189	300(5.8%)
Flickr [72]	7575	239738	12407	450(5.9%)

6.2 Baselines

We study the performance of the following SSL-based graph anomaly detection methods:

- Generative methods: DOMINANT [9], AnomalyDAE [12], GUIDE [70], GAAN [5];
- Contrastive methods (and some also generative): CoLA [40], ANEMONE [22], GRADATE [11], SL-GAD [85], Sub-CR [75], CONAD [64].

Particularly, the SSL-related HPs for each GAD algorithm and their discretized search spaces are given in Table 4 in the Appendix. These GAD methods are further summarized in Table 5 in the Appendix.

6.3 Evaluation Metrics

To evaluate the effectiveness of various GAD algorithms, we utilize the ROC-AUC metric [17] (*AUC* for short hereinafter), where a value approaching 1 denotes the best possible performance.

Moreover, to quantify the performance variation of an individual GAD method under different SSL-related HP configurations, we define the following *performance variation* metric:

$$\frac{\max(AUC) - \min(AUC)}{\max(AUC)}, \tag{4}$$

where $\max(AUC)$ and $\min(AUC)$ represent the maximum and minimum of achieved *AUC* values for the evaluated GAD algorithm with different configurations, respectively. Hence, the smaller this value is, the less sensitive the algorithm is to SSL-related HPs.

Further, we define the *performance gain over minimal AUC* as

$$\frac{\mathrm{CSM}(AUC) - \min(AUC)}{\min(AUC)}, \tag{5}$$

where $\mathrm{CSM}(AUC)$ indicates the *AUC* value obtained for the evaluated GAD algorithm when configured with the HPs selected using the Contrast Score Margin. This metric can quantify the effectiveness of our strategy relative to the worst case hyperparameter setting. Next, we define *performance gain over median AUC* as

$$\frac{\mathrm{CSM}(AUC) - \mathrm{median}(AUC)}{\mathrm{median}(AUC)}, \tag{6}$$

where $\mathrm{median}(AUC)$ represents the median of the obtained *AUC* values for the GAD algorithm with different configurations. Thus, if the value of this metric is positive, the GAD algorithm configured with our selected HPs can at least outperform its counterparts configured with 50% of the other sampled hyperparameter values.

Table 3: Performance gain over minimal AUC defined as $\frac{\mathbf{CSM}(AUC)-\min(AUC)}{\min(AUC)}$ (first seven rows), and performance gain over median AUC defined as $\frac{\mathbf{CSM}(AUC)-\mathbf{median}(AUC)}{\mathbf{median}(AUC)}$ (last seven rows). Results are averaged on five independent runs. CSM is contrast score margin defined in Equation 3, while OOM, OOR and gray cells convey the same concepts as in Table 1.

Dataset	CoLA	ANEMONE	GRADATE	SL-GAD	Sub-CR	CONAD	DOMINANT	A-DAE	GUIDE	GAAN
Cora [53]	0.5%	8.6%	4.0%	21.2%	16.2%	5.4%	5.2%	11.3%	5.0%	36.3%
CiteSeer [53]	1.2%	5.9%	14.3%	19.1%	4.3%	2.3%	1.3%	4.6%	1.2%	23.5%
PubMed [53]	1.6%	6.6%	OOM	23.7%	OOM	2.1%	1.8%	11.9%	1.8%	21.6%
ACM [56]	2.8%	6.8%	OOM	21.3%	OOM	0.1%	0%	5.6%	0%	27.2%
Flickr [72]	2.2%	15.8%	OOR	18.2%	4.3%	OOM	0%	30.8%	0.1%	19.4%
BlogCataLog [72]	4.7%	19.7%	OOR	30.4%	2.4%	OOM	0%	32.2%	0%	16.7%
Average	2.2%	10.6%	9.1%	22.3%	6.8%	3.3%	2.8%	16.0%	2.7%	24.1%
Cora [53]	-0.1%	0.3%	-0.6%	3.3%	-1.6%	4.0%	3.7%	2.9%	3.8%	1.3%
CiteSeer [53]	0.1%	1.0%	4.0%	3.7%	-0.4%	1.2%	0.5%	-3.0%	0.6%	1.9%
PubMed [53]	0.2%	1.5%	OOM	4.8%	OOM	1.5%	1.3%	-2.9%	1.5%	7.7%
ACM [56]	-0.7%	-0.8%	OOM	4.3%	OOM	0.1%	0%	-4.1%	0%	-0.6%
Flickr [72]	0.6%	2.0%	OOR	2.8%	-3.2%	OOM	-0.3%	0.9%	0%	1.3%
BlogCataLog [72]	2.0%	7.2%	OOR	5.0%	-2.2%	OOM	0%	-3.4%	0%	3.5%
Average	0.3%	1.9%	1.7%	4.0%	-2.1%	1.5%	1.8%	-1.6%	2.0%	2.5%

6.4 Software and Hardware

All algorithms are implemented in Python 3.8 (using PyTorch [48] and PyTorch Geometric [13] libraries when applicable) and ran on workstations equipped with AMD EPYC7453 CPUs (with 64GB RAM) and/or Nvidia RTX4090 GPUs (with 24.0 GB video memory). All code and datasets are available on Anonymous GitHub[1].

6.5 Results and Analysis

We answer the research questions as follows:

6.5.1 ***RQ1*** *(Sensitivity of SSL-based GAD methods to HPs).* The results are summarized in Table 1 for five independent runs. Typical runs are depicted in Figure 1 and in Figures 3-11 in Appendix C. We briefly analyzed the results in Subsection 4.3; more detailed analyses are given in Appendix C. To recall, four out of ten algorithms show moderate performance variations, while the remaining six algorithms demonstrate large performance variations when the values of SSL-related HPs are varied. In other words, SSL-based GAD methods are (highly) sensitive to hyperparameter values.

6.5.2 ***RQ2*** *(Effectiveness of AutoGAD in tuning SSL-related HPs).* The results are summarized in Table 3 for five independent runs, while Figure 1 and Figures 3-11 depict typical runs. We have the following main observations:

1) From the first seven rows in Table 3, one can see that *AutoGAD* can result in moderate *performance gain over minimal AUC* (namely between 2.2% and 3.3% on average) for CoLA, CONAD, DOMINANT, and GUIDE. Recall that these four algorithms exhibit moderate performance variations, ranging from 2.8% to 5.0% on average. Moreover, *AutoGAD* leads to large *performance gain over minimal AUC* (namely between 6.8% and 24.1% on average) for the remaining 6 algorithms, which suffer from large performance variations (namely between 9.9% and 23.4% on average). Overall, *AutoGAD* is substantially better than the worst case, i.e., when one happens to select the HP values that give the smallest *AUC* value;
2) From the last seven rows in Table 3, one can see that *AutoGAD* can result in positive *performance gain over median AUC* in 8 out 10 algorithms (ranging from 0.3% to 4.0% on average), implying that the HP values selected by *AutoGAD* are better than at least 50% of randomly selected HP values. Particularly, the *performance gains over median AUC* for SL-GAD [85], GAAN [5], GUIDE [70], and ANEMONE [22] are 4.0%, 2.5%, 2.0%, 1.9% respectively, which shows that *AutoGAD* is highly effective for these methods.
3) Following the second observation, we check the details in Figure 11 for SL-GAD, Figure 3 for GAAN, Figure 8 for GUIDE, and Figure 1 for ANEMONE. For SL-GAD, GAAN and GUIDE, *AutoGAD* often selects HP values better than 90% of randomly selected HPs values. For ANEMONE, the HP values selected by *AutoGAD* often outperform 75% of randomly selected HP values.

7 CONCLUSIONS

SSL has received much attention in recent years, and many recent studies have explored SSL to perform unsupervised GAD. However, we found that most existing studies tune hyperparameters arbitrarily or selectively (i.e., guided by labels), and our empirical findings reveal that most methods are highly sensitive to hyperparameter settings. Using label information to tune hyperparameters in an unsupervised setting, however, is label information leakage and leads to severe overestimation of model performance. To mitigate this issue, we introduce AutoGAD, the first automated hyperparameter selection method for SSL-based unsupervised GAD. Extensive experiments demonstrate the effectiveness of our proposed strategy. Overall, we aim to raise awareness to the label information leakage

[1] https://anonymous.4open.science/r/AutoGAD-E519

issue in the unsupervised GAD field, and AutoGAD provides a first step towards achieving truly unsupervised SSL-based GAD.

REFERENCES

[1] Maroua Bahri, Flavia Salutari, Andrian Putina, and Mauro Sozio. 2022. AutoML: state of the art with a focus on anomaly detection, challenges, and research directions. *International Journal of Data Science and Analytics* 14, 2 (2022), 113–126.

[2] Markus M Breunig, Hans-Peter Kriegel, Raymond T Ng, and Jörg Sander. 2000. LOF: identifying density-based local outliers. In *Proceedings of the 2000 ACM SIGMOD international conference on Management of data*. 93–104.

[3] Bo Chen, Jing Zhang, Xiaokang Zhang, Yuxiao Dong, Jian Song, Peng Zhang, Kaibo Xu, Evgeny Kharlamov, and Jie Tang. 2022. Gccad: Graph contrastive learning for anomaly detection. *IEEE Transactions on Knowledge and Data Engineering* (2022).

[4] Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. 2020. A simple framework for contrastive learning of visual representations. In *International conference on machine learning*. PMLR, 1597–1607.

[5] Zhenxing Chen, Bo Liu, Meiqing Wang, Peng Dai, Jun Lv, and Liefeng Bo. 2020. Generative adversarial attributed network anomaly detection. In *Proceedings of the 29th ACM International Conference on Information & Knowledge Management*. 1989–1992.

[6] Ekin D Cubuk, Barret Zoph, Dandelion Mane, Vijay Vasudevan, and Quoc V Le. 2018. Autoaugment: Learning augmentation policies from data. *arXiv preprint arXiv:1805.09501* (2018).

[7] Ekin D Cubuk, Barret Zoph, Jonathon Shlens, and Quoc V Le. 2020. Randaugment: Practical automated data augmentation with a reduced search space. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition workshops*. 702–703.

[8] Kaize Ding, Jundong Li, Nitin Agarwal, and Huan Liu. 2021. Inductive anomaly detection on attributed networks. In *Proceedings of the twenty-ninth international conference on international joint conferences on artificial intelligence*. 1288–1294.

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

[10] Xueying Ding, Lingxiao Zhao, and Leman Akoglu. 2022. Hyperparameter sensitivity in deep outlier detection: Analysis and a scalable hyper-ensemble solution. *Advances in Neural Information Processing Systems* 35 (2022), 9603–9616.

[11] Jingcan Duan, Siwei Wang, Pei Zhang, En Zhu, Jingtao Hu, Hu Jin, Yue Liu, and Zhibin Dong. 2023. Graph anomaly detection via multi-scale contrastive learning networks with augmented view. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 37. 7459–7467.

[12] Haoyi Fan, Fengbin Zhang, and Zuoyong Li. 2020. Anomalydae: Dual autoencoder for anomaly detection on attributed networks. In *ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 5685–5689.

[13] Matthias Fey and Jan Eric Lenssen. 2019. Fast graph representation learning with PyTorch Geometric. *arXiv preprint arXiv:1903.02428* (2019).

[14] Pedro Garcia-Teodoro, Jesus Diaz-Verdejo, Gabriel Maciá-Fernández, and Enrique Vázquez. 2009. Anomaly-based network intrusion detection: Techniques, systems and challenges. *computers & security* 28, 1-2 (2009), 18–28.

[15] Nicolas Goix. 2016. How to evaluate the quality of unsupervised anomaly detection algorithms? *arXiv preprint arXiv:1607.01152* (2016).

[16] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. 2014. Generative adversarial nets. *Advances in neural information processing systems* 27 (2014).

[17] James A Hanley and Barbara J McNeil. 1982. The meaning and use of the area under a receiver operating characteristic (ROC) curve. *Radiology* 143, 1 (1982), 29–36.

[18] Kaveh Hassani and Amir Hosein Khasahmadi. 2022. Learning graph augmentations to learn graph representations. *arXiv preprint arXiv:2201.09830* (2022).

[19] Ryuichiro Hataya, Jan Zdenek, Kazuki Yoshizoe, and Hideki Nakayama. 2020. Faster autoaugment: Learning augmentation strategies using backpropagation. In *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part XXV 16*. Springer, 1–16.

[20] Daniel Ho, Eric Liang, Xi Chen, Ion Stoica, and Pieter Abbeel. 2019. Population based augmentation: Efficient learning of augmentation policy schedules. In *International conference on machine learning*. PMLR, 2731–2741.

[21] Yizhu Jiao, Yun Xiong, Jiawei Zhang, Yao Zhang, Tianqi Zhang, and Yangyong Zhu. 2020. Sub-graph contrast for scalable self-supervised graph representation learning. In *2020 IEEE international conference on data mining (ICDM)*. IEEE, 222–231.

[22] Ming Jin, Yixin Liu, Yu Zheng, Lianhua Chi, Yuan-Fang Li, and Shirui Pan. 2021. Anemone: Graph anomaly detection with multi-scale contrastive learning. In *Proceedings of the 30th ACM International Conference on Information & Knowledge Management*. 3122–3126.

[23] Wei Jin, Xiaorui Liu, Xiangyu Zhao, Yao Ma, Neil Shah, and Jiliang Tang. 2021. Automated self-supervised learning for graphs. *arXiv preprint arXiv:2106.05470* (2021).

[24] Donald R Jones, Matthias Schonlau, and William J Welch. 1998. Efficient global optimization of expensive black-box functions. *Journal of Global optimization* 13 (1998), 455–492.

[25] Shachar Kaufman, Saharon Rosset, Claudia Perlich, and Ori Stitelman. 2012. Leakage in data mining: Formulation, detection, and avoidance. *ACM Transactions on Knowledge Discovery from Data (TKDD)* 6, 4 (2012), 1–21.

[26] Hwan Kim, Byung Suk Lee, Won-Yong Shin, and Sungsu Lim. 2022. Graph anomaly detection with graph neural networks: Current status and challenges. *IEEE Access* (2022).

[27] Thomas N Kipf and Max Welling. 2016. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907* (2016).

[28] Kwei-Herng Lai, Daochen Zha, Guanchu Wang, Junjie Xu, Yue Zhao, Devesh Kumar, Yile Chen, Purav Zumkhawaka, Minyang Wan, Diego Martinez, et al. 2021. Tods: An automated time series outlier detection system. In *Proceedings of the aaai conference on artificial intelligence*, Vol. 35. 16060–16062.

[29] Jundong Li, Harsh Dani, Xia Hu, and Huan Liu. 2017. Radar: Residual analysis for anomaly detection in attributed networks.. In *IJCAI*, Vol. 17. 2152–2158.

[30] Yuening Li, Zhengzhang Chen, Daochen Zha, Kaixiong Zhou, Haifeng Jin, Haifeng Chen, and Xia Hu. 2021. Automated anomaly detection via curiosity-guided search and self-imitation learning. *IEEE Transactions on Neural Networks and Learning Systems* 33, 6 (2021), 2365–2377.

[31] Yuening Li, Zhengzhang Chen, Daochen Zha, Kaixiong Zhou, Haifeng Jin, Haifeng Chen, and Xia Hu. 2021. Autood: Neural architecture search for outlier detection. In *2021 IEEE 37th International Conference on Data Engineering (ICDE)*. IEEE, 2117–2122.

[32] Yonggang Li, Guosheng Hu, Yongtao Wang, Timothy Hospedales, Neil M Robertson, and Yongxin Yang. 2020. Differentiable automatic data augmentation. In *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part XXII 16*. Springer, 580–595.

[33] Yuening Li, Daochen Zha, Praveen Venugopal, Na Zou, and Xia Hu. 2020. Pyodds: An end-to-end outlier detection system with automated machine learning. In *Companion Proceedings of the Web Conference 2020*. 153–157.

[34] Zhong Li, Jiayang Shi, and Matthijs van Leeuwen. 2023. Graph Neural Network based Log Anomaly Detection and Explanation. *arXiv preprint arXiv:2307.00527* (2023).

[35] Sungbin Lim, Ildoo Kim, Taesup Kim, Chiheon Kim, and Sungwoong Kim. 2019. Fast autoaugment. *Advances in Neural Information Processing Systems* 32 (2019).

[36] Fanzhen Liu, Xiaoxiao Ma, Jia Wu, Jian Yang, Shan Xue, Amin Beheshti, Chuan Zhou, Hao Peng, Quan Z Sheng, and Charu C Aggarwal. 2022. Dagad: Data augmentation for graph anomaly detection. In *2022 IEEE International Conference on Data Mining (ICDM)*. IEEE, 259–268.

[37] Hanxiao Liu, Karen Simonyan, and Yiming Yang. 2018. Darts: Differentiable architecture search. *arXiv preprint arXiv:1806.09055* (2018).

[38] Kay Liu, Yingtong Dou, Yue Zhao, Xueying Ding, Xiyang Hu, Ruitong Zhang, Kaize Ding, Canyu Chen, Hao Peng, Kai Shu, et al. 2022. Bond: Benchmarking unsupervised outlier node detection on static attributed graphs. *Advances in Neural Information Processing Systems* 35 (2022), 27021–27035.

[39] Yixin Liu, Ming Jin, Shirui Pan, Chuan Zhou, Yu Zheng, Feng Xia, and S Yu Philip. 2022. Graph self-supervised learning: A survey. *IEEE Transactions on Knowledge and Data Engineering* 35, 6 (2022), 5879–5900.

[40] Yixin Liu, Zhao Li, Shirui Pan, Chen Gong, Chuan Zhou, and George Karypis. 2021. Anomaly detection on attributed networks via contrastive self-supervised learning. *IEEE transactions on neural networks and learning systems* 33, 6 (2021), 2378–2392.

[41] Zhiyuan Liu, Chunjie Cao, Fangjian Tao, and Jingzhang Sun. 2023. Revisiting Graph Contrastive Learning for Anomaly Detection. *arXiv preprint arXiv:2305.02496* (2023).

[42] Youzhi Luo, Michael McThrow, Wing Yee Au, Tao Komikado, Kanji Uchino, Koji Maruhashi, and Shuiwang Ji. 2022. Automated data augmentations for graph classification. *arXiv preprint arXiv:2202.13248* (2022).

[43] Martin Q Ma, Yue Zhao, Xiaorong Zhang, and Leman Akoglu. 2023. The need for unsupervised outlier model selection: A review and evaluation of internal evaluation strategies. *ACM SIGKDD Explorations Newsletter* 25, 1 (2023).

[44] Xiaoxiao Ma, Jia Wu, Shan Xue, Jian Yang, Chuan Zhou, Quan Z Sheng, Hui Xiong, and Leman Akoglu. 2021. A comprehensive survey on graph anomaly detection with deep learning. *IEEE Transactions on Knowledge and Data Engineering* (2021).

[45] Henrique O Marques, Ricardo JGB Campello, Jörg Sander, and Arthur Zimek. 2020. Internal evaluation of unsupervised outlier detection. *ACM Transactions on Knowledge Discovery from Data (TKDD)* 14, 4 (2020), 1–42.

[46] Soroor Motie and Bijan Raahemi. 2023. Financial fraud detection using graph neural networks: A systematic review. *Expert Systems with Applications* (2023), 122156.

[47] Robert Nisbet, John Elder, and Gary D Miner. 2009. *Handbook of statistical analysis and data mining applications*. Academic press.

[48] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. 2019. Pytorch: An imperative style, high-performance deep learning library. *Advances in neural information processing systems* 32 (2019).
[49] Zhen Peng, Minnan Luo, Jundong Li, Huan Liu, Qinghua Zheng, et al. 2018. ANOMALOUS: A Joint Modeling Approach for Anomaly Detection on Attributed Networks.. In *IJCAI*. 3513–3519.
[50] Bryan Perozzi and Leman Akoglu. 2016. Scalable anomaly ranking of attributed neighborhoods. In *Proceedings of the 2016 SIAM International Conference on Data Mining*. SIAM, 207–215.
[51] Andrian Putina, Maroua Bahri, Flavia Salutari, and Mauro Sozio. 2022. AutoAD: an Automated Framework for Unsupervised Anomaly Detectio.. In *2022 IEEE 9th International Conference on Data Science and Advanced Analytics (DSAA)*. IEEE, 1–10.
[52] Alexander J Ratner, Henry Ehrenberg, Zeshan Hussain, Jared Dunnmon, and Christopher Ré. 2017. Learning to compose domain-specific transformations for data augmentation. *Advances in neural information processing systems* 30 (2017).
[53] Prithviraj Sen, Galileo Namata, Mustafa Bilgic, Lise Getoor, Brian Galligher, and Tina Eliassi-Rad. 2008. Collective classification in network data. *AI magazine* 29, 3 (2008), 93–93.
[54] Junwei Sun, Bai Wang, and Bin Wu. 2021. Automated graph representation learning for node classification. In *2021 International Joint Conference on Neural Networks (IJCNN)*. IEEE, 1–7.
[55] Susheel Suresh, Pan Li, Cong Hao, and Jennifer Neville. 2021. Adversarial graph augmentation to improve graph contrastive learning. *Advances in Neural Information Processing Systems* 34 (2021), 15920–15933.
[56] Jie Tang, Jing Zhang, Limin Yao, Juanzi Li, Li Zhang, and Zhong Su. 2008. Arnetminer: extraction and mining of academic social networks. In *Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining*. 990–998.
[57] Hanghang Tong, Christos Faloutsos, and Jia-Yu Pan. 2006. Fast random walk with restart and its applications. In *Sixth international conference on data mining (ICDM'06)*. IEEE, 613–622.
[58] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Lio, and Yoshua Bengio. 2017. Graph attention networks. *arXiv preprint arXiv:1710.10903* (2017).
[59] Florian Wenzel, Jasper Snoek, Dustin Tran, and Rodolphe Jenatton. 2020. Hyperparameter ensembles for robustness and uncertainty quantification. *Advances in Neural Information Processing Systems* 33 (2020), 6514–6527.
[60] Christopher Williams and Carl Rasmussen. 1995. Gaussian processes for regression. *Advances in neural information processing systems* 8 (1995).
[61] Lirong Wu, Haitao Lin, Cheng Tan, Zhangyang Gao, and Stan Z Li. 2021. Self-supervised learning on graphs: Contrastive, generative, or predictive. *IEEE Transactions on Knowledge and Data Engineering* (2021).
[62] Yaochen Xie, Zhao Xu, Jingtun Zhang, Zhengyang Wang, and Shuiwang Ji. 2022. Self-supervised learning of graph neural networks: A unified review. *IEEE transactions on pattern analysis and machine intelligence* 45, 2 (2022), 2412–2429.
[63] Weizhi Xu, Junfei Wu, Qiang Liu, Shu Wu, and Liang Wang. 2022. Evidence-aware fake news detection with graph neural networks. In *Proceedings of the ACM Web Conference 2022*. 2501–2510.
[64] Zhiming Xu, Xiao Huang, Yue Zhao, Yushun Dong, and Jundong Li. 2022. Contrastive attributed network anomaly detection with data augmentation. In *Pacific-Asia Conference on Knowledge Discovery and Data Mining*. Springer, 444–457.
[65] Zekun Xu, Deovrat Kakde, and Arin Chaudhuri. 2019. Automatic hyperparameter tuning method for local outlier factor, with applications to anomaly detection. In *2019 IEEE International Conference on Big Data (Big Data)*. IEEE, 4201–4207.
[66] Yihang Yin, Qingzhong Wang, Siyu Huang, Haoyi Xiong, and Xiang Zhang. 2022. Autogcl: Automated graph contrastive learning via learnable view generators. In *Proceedings of the AAAI conference on artificial intelligence*, Vol. 36. 8892–8900.
[67] Jaemin Yoo, Yue Zhao, Lingxiao Zhao, and Leman Akoglu. 2023. DSV: An Alignment Validation Loss for Self-supervised Outlier Model Selection. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*. Springer, 254–269.
[68] Yuning You, Tianlong Chen, Yang Shen, and Zhangyang Wang. 2021. Graph contrastive learning automated. In *International Conference on Machine Learning*. PMLR, 12121–12132.
[69] Yuning You, Tianlong Chen, Yongduo Sui, Ting Chen, Zhangyang Wang, and Yang Shen. 2020. Graph contrastive learning with augmentations. *Advances in neural information processing systems* 33 (2020), 5812–5823.
[70] Xu Yuan, Na Zhou, Shuo Yu, Huafei Huang, Zhikui Chen, and Feng Xia. 2021. Higher-order structure based anomaly detection on attributed networks. In *2021 IEEE International Conference on Big Data (Big Data)*. IEEE, 2691–2700.
[71] Han Yue, Chunhui Zhang, Chuxu Zhang, and Hongfu Liu. 2022. Label-invariant augmentation for semi-supervised graph classification. *Advances in Neural Information Processing Systems* 35 (2022), 29350–29361.
[72] Hanqing Zeng, Hongkuan Zhou, Ajitesh Srivastava, Rajgopal Kannan, and Viktor Prasanna. 2019. Graphsaint: Graph sampling based inductive learning method. *arXiv preprint arXiv:1907.04931* (2019).
[73] Jiaqi Zeng and Pengtao Xie. 2021. Contrastive self-supervised learning for graph classification. In *Proceedings of the AAAI conference on Artificial Intelligence*, Vol. 35. 10824–10832.
[74] Daochen Zha, Kwei-Herng Lai, Mingyang Wan, and Xia Hu. 2020. Meta-AAD: Active anomaly detection with deep reinforcement learning. In *2020 IEEE International Conference on Data Mining (ICDM)*. IEEE, 771–780.
[75] Jiaqiang Zhang, Senzhang Wang, and Songcan Chen. 2022. Reconstruction enhanced multi-view contrastive learning for anomaly detection on attributed networks. *arXiv preprint arXiv:2205.04816* (2022).
[76] Mengchun Zhang, Maryam Qamar, Taegoo Kang, Yuna Jung, Chenshuang Zhang, Sung-Ho Bae, and Chaoning Zhang. 2023. A survey on graph diffusion models: Generative ai in science for molecule, protein and material. *arXiv preprint arXiv:2304.01565* (2023).
[77] Xinyu Zhang, Qiang Wang, Jian Zhang, and Zhao Zhong. 2019. Adversarial autoaugment. *arXiv preprint arXiv:1912.11188* (2019).
[78] Tong Zhao, Yozen Liu, Leonardo Neves, Oliver Woodford, Meng Jiang, and Neil Shah. 2021. Data augmentation for graph neural networks. In *Proceedings of the aaai conference on artificial intelligence*, Vol. 35. 11015–11023.
[79] Tong Zhao, Xianfeng Tang, Danqing Zhang, Haoming Jiang, Nikhil Rao, Yiwei Song, Pallav Agrawal, Karthik Subbian, Bing Yin, and Meng Jiang. 2022. Autogda: Automated graph data augmentation for node classification. In *Learning on Graphs Conference*. PMLR, 32–1.
[80] Yue Zhao and Leman Akoglu. 2022. Towards unsupervised hpo for outlier detection. *arXiv preprint arXiv:2208.11727* (2022).
[81] Yue Zhao, Zain Nasrullah, Maciej K Hryniewicki, and Zheng Li. 2019. LSCP: Locally selective combination in parallel outlier ensembles. In *Proceedings of the 2019 SIAM International Conference on Data Mining*. SIAM, 585–593.
[82] Yue Zhao, Ryan Rossi, and Leman Akoglu. 2021. Automatic unsupervised outlier model selection. *Advances in Neural Information Processing Systems* 34 (2021), 4489–4502.
[83] Yue Zhao, Ryan A Rossi, and Leman Akoglu. 2020. Automating outlier detection via meta-learning. *arXiv preprint arXiv:2009.10606* (2020).
[84] Yue Zhao, Sean Zhang, and Leman Akoglu. 2022. Toward unsupervised outlier model selection. In *2022 IEEE International Conference on Data Mining (ICDM)*. IEEE, 773–782.
[85] Yu Zheng, Ming Jin, Yixin Liu, Lianhua Chi, Khoa T Phan, and Yi-Ping Phoebe Chen. 2021. Generative and contrastive self-supervised learning for graph anomaly detection. *IEEE Transactions on Knowledge and Data Engineering* (2021).
[86] Chong Zhou and Randy C Paffenroth. 2017. Anomaly detection with robust deep autoencoders. In *Proceedings of the 23rd ACM SIGKDD international conference on knowledge discovery and data mining*. 665–674.

A MORE RELATED WORK

Our work is closely related to node anomaly detection on static attributed graphs, self-supervised learning for graph anomaly detection, automated self-supervised learning, and automated anomaly detection.

A.1 Anomaly Detection on Attributed Graphs

Early studies on node anomaly detection in static attributed graphs usually employ non-deep learning methods to detect anomalies. Specifically, AMEN [50] leverages ego-network information of each node to detect anomalies; Radar [29] identifies anomalies by considering the residuals of node attributes information and its coherence with topological information; Anomalous [49] further combines residual analysis of node attributes and CUR decomposition of adjacency matrix to detect anomalies. These methods have achieved success on low-dimensional attributed graphs, while their performance is limited on graphs with high-dimensional node attributes and complex structures.

Recently, deep learning-based methods have been proposed for GAD. These methods usually employ graph autoencoders to encode nodes followed by decoders to reconstruct the adjacency matrix and/or node attributes. As a result, nodes with large reconstruction errors are considered anomalies. For instance, DOMINANT [9] utilises GCN as the autoencoder to encode and reconstruct the attribute matrix and the adjacency matrix, and the anomaly

score is defined as the weighted sum of reconstruction errors of attribute and structure. Meanwhile, AnomalyDAE [12] and GUIDE [70] follow similar principles. Despite their superior performance to non-deep learning methods, these reconstruction-based methods still suffer from sub-optimal performance as reconstruction is generic unsupervised learning objective where the fineness of anomaly detection objective is not fully considered. Besides, these methods require the full attribute and adjacency matrices as model input, making it unsuitable or even impossible for large graphs.

A.2 Self-Supervised Learning for Graph Anomaly Detection

Graph SSL aims to learn a model by using self-generated supervision signals from the graph itself, without relying on human-annotated labels [39]. It has achieved promising performance on typical graph mining tasks such as representation learning [21] and graph classification [73]. Particularly, Liu et al. [40] start applying SSL to tackle the GAD problem. Their proposed method CoLA performs single scale comparison (node-subgraph) for anomaly detection. However, ANEMONE [22] argues that modeling the relationships in a single contrastive perspective leads to limited capability of capturing complex anomalous patterns. Hence, they propose additional node-node contrast. Additionally, GRADATE [11] and M-MAG [41] combines various multi-contrast objectives, namely node-node, node-subgraph, and subgraph-subgraph contrasts for node anomaly detection. Meanwhile, for achieving better performance, SL-GAD [85] combines multi-view contrastive learning and generative attribute regression, while Sub-CR [75] combines multi-view contrastive learning and graph autoencoder. In addition, CONAD [64] also considers both contrastive learning and generative reconstruction for better node anomaly detection.

A.3 Automated Self-Supervised Learning

A.3.1 Automated Data Augmentations on Images. The seminal work on automated data augmentations for images is proposed by [6, 52], while the following works try to improve [6] via faster searching mechanism [7, 20, 35] or advanced optimization methods including Faster-AA [19], DADA [32] (these two studies followed the DARTS [37]) and [77].

A.3.2 Automated Data Augmentations on Graphs. There exist related studies on automated data augmentations for graph representation learning [18, 23, 55, 62, 66, 68], node classification [54, 78], and graph-level classification [42, 66, 71]. Specifically, JOAO [68] learns the sampling distribution of a set of predefined graph augmentations (namely how to combine them), where the augmentations themselves are not learnable. Meanwhile, AD-GCL [55] design a learnable edge dropping augmentation and employ adversarial training strategy, while the node-level augmentations are not studied and the semantic label preserving capability is not guaranteed. Besides, AutoGCL [66] propose a learnable graph view generator that learns a probability distribution over the node-level augmentations, which can well preserve the semantic labels of graphs. Moreover, [42] augments the graph data samples while [71] perturbs the representation vector.

However, these methods only focus on typical graph learning tasks such as node classification and representation learning, and it is unclear how to perform automated SSL for unsupervised GAD task.

A.4 Automated Anomaly Detection

Recently, some studies [1, 10, 80, 82] point out that unsupervised anomaly detection methods tend to be highly sensitive to the setting of HPs. For example, [82] shows that 10x performance difference is observed for LOF [2] by simply changing the number of nearest neighbours in some datasets. Moreover, [10] indicates that deep anomaly detection methods suffer more from these HPs sensitivity issues. Concretely, [80] demonstrates that RAE [86] exhibits 37x performance difference with different HPs configurations in some datasets.

To tackle this issue, automated HP tuning and model selection for unsupervised anomaly detection has received increasing but insufficient attention. Concretely, Bahri et al. [1] present an overview of autoML strategies designed for unsupervised anomaly detection. Inspired by [1, 80], we can subdivide existing automated unsupervised detection approaches into two categories.

A.4.1 Supervised Evaluation. Although the involved anomaly detection algorithms are unsupervised, their evaluation requires ground-truth labels when performing HP tuning and model selection. For instance, given a new dataset, [33] proposes a pipeline PyODDS to search for an optimal anomaly detection algorithm along with its best configuration for tabular data. However, they utilise a supervised metric, *F1-score* that is based on the ground-truth labels, to compare different algorithms and configurations. Similarly, TODS [28] searches for an optimal algorithm and its best configuration for time series anomaly detection. Again, they employ *F1-score* as the evaluation metric. Meanwhile, AutoOD [31] develops a reinforcement learning-based approach that leverages Recurrent Neural Network controller, with the aim to select optimal neural network model to detect anomalies on image data. However, they utilise a supervised metric *ROC* to select models. Additionally, AutoAD [30] utilises meta-learning to find an optimal neural network model for anomaly detection, wherein they formulate the search process as a reinforcement learning problem. To find the optimal HPs values, they leverage the detection accuracy in the validation set as the reward, which requires ground-truth labels.

A.4.2 Unsupervised Evaluation. The evaluation does not require ground-truth labels when performing HP tuning and model selection. They roughly include 1) randomly selecting an HP configuration, 2) selecting an HP configuration via internal evaluation strategy [15, 45], 3) averaging the outputs of a set of randomly selected HP configurations [59], and 4) meta-learning based methods. Specifically, give a dataset, LSCP [81] initialises the algorithm with different configurations, and defines the so-called local region by consensus of the nearest training instances. On this basis, they select the optimal configuration by measuring the similarity between the anomaly scores associated with each configuration and the pseudo ground truth (which is obtained by taking the average or maximum of anomaly scores across all base detectors). Moreover, [45] presents internal evaluation strategy for unsupervised anomaly detection, which is however costly to compute. Meanwhile, MetaOD [83] leverages meta-learning to select an optimal algorithm along

with it best HPs configuration. Specifically, meta-learning methods stand on the historical performance (in terms of ground-truth labels) of anomaly detection algorithms on accessible datasets. Similarly, Meta-AAD [74] trains the meta-learner using deep reinforcement learning, with the aim to select an optimal detection model for a new dataset. Similarly, HPOD [80] employs meta-learning to enable SMBO [24] for efficiently optimizing the HPs of unsupervised anomaly detection methods. Particularly, SMBO employs a sampling strategy to efficiently identify promising HP configurations. AutoAD [51] consider joint algorithm selection and HP optimisation for unsupervised anomaly detection. Particularly, their method computes the SSE, Variance or Kurtosis of instances after removing the top K predicted anomalies as internal quality measure.

However, these automated anomaly detection methods are primarily designed for non-graph data.

B PITFALLS IN EXISTING METHODS (FULL ANALYSIS)

B.0.1 CoLA [40]. Particularly, CoLA is the first *contrastive-based* framework for unsupervised GAD. The design of its *data augmentation* module and *contrast learning* module is as follows.

Data Augmentation Module They consider one type of data augmentation, subgraph sampling, to obtain local augmented view for each node. Particularly, they employ RWR [57] to generate a sub-graph with a fixed size K in subgraph sampling, resulting in one critical HP in graph augmentation, namely K.

Contrast Learning Module They consider a single contrast aspect, namely node-subgraph contrast between the embedding of the target node and the aggregated embedding of its local sug-graph, without resulting in any HPs.

HPs Sensitivity & Tuning They conducted sensitive analysis and found that the selection of subgraph size K is dependent on the specific dataset. The AUC performance usually increases first and then decreases with the increasing of K. However, for efficiency and robustness consideration, they heuristically set the sampled subgraph size $K = 4$ for all datasets.

B.0.2 ANEMONE [22]. This is a *contrastive-based* framework for unsupervised GAD. They argue that modeling the relationships in a single contrastive perspective leads to limited capability of capturing complex anomalous patterns, and thus propose additional contrast perspectives as follows.

Graph Augmentation Module They consider a single graph augmentation operation, namely Random Ego-Nets generation with a fixed size K. Specifically, taking the target node as the center, they employ RWR [57] to generate two different subgraphs as ego-nets with a fixed size K. Overall, they result in one critical HP in graph augmentation, namely K.

Contrast Learning Module They consider two contrast perspectives: 1) node-node contrast between the embedding of masked target node within ego-net and the embedding of the original node, leading to loss term $\mathcal{L}_{NN}$, and 2) node-subgraph contrast within each view, leading to loss term $\mathcal{L}_{NS}$. On this basis, they combine these loss terms as

$$\mathcal{L} = (1-\alpha)\mathcal{L}_{NN} + \alpha\mathcal{L}_{NS}$$

where $\alpha \in [0, 1]$ is the trade-off HP. Hence, they result in one critical HP in graph contrast, namely α.

HPs Sensitivity & Tuning In their ablation studies: 1) by using ground-truth label information, they heuristically set α as $0.8, 0.6, 0.8$ on Cora, CiterSeer and PubMed respectively, and report the corresponding results; and 2) the setting of K was not studied, and it is set to 4 for all datasets.

B.0.3 GRADATE [11] . This is also a *contrastive-based* framework. They argue that subgraph-subgraph contrast is also critical in detecting graph anomalies, and design it as follows.

Data Augmentation Module They consider a single graph augmentation operation, namely Edge Modification that removes edges in the adjacency matrix as well as add the same number of edges. Concretely, they fix a proportion P, and then uniformly and randomly sample $\frac{P \cdot M}{2}$ edges from a total of M edges to remove. Meanwhile, $\frac{P \cdot M}{2}$ edges are added into the adjacency matrix. Overall, they result in one critical HP in graph augmentation, namely P.

Contrast Learning Module They consider three contrast aspects: 1) node-node contrast within each view, leading to loss term $\mathcal{L}_{NN}$), 2) node-subgraph contrast within each view, leading to loss term $\mathcal{L}_{NS}$, and 3) subgraph-subgraph contrast between original view and augmented view, leading to loss term $\mathcal{L}_{SS}$. On this basis, they combine these loss terms as

$$\mathcal{L} = (1-\beta)\mathcal{L}_{NN} + \beta\mathcal{L}_{NS} + \gamma\mathcal{L}_{SS},$$

where $\beta, \gamma \in (0, 1)$ are trade-off HPs. More, $\mathcal{L}_{NN} = \alpha\mathcal{L}_{NN,1} + (1-\alpha)\mathcal{L}_{NN,2}$, and $\mathcal{L}_{NS} = \alpha\mathcal{L}_{NS,1} + (1-\alpha)\mathcal{L}_{NS,2}$, with $\mathcal{L}_{NN,1}$ and $\mathcal{L}_{NN,2}$ being the loss term in the first and second views respectively. Overall, they result in three critical HPs in graph contrast, namely the combination weights α, β, γ.

HPs Sensitivity & Tuning In their ablation studies, 1) they compared four different graph augmentation strategies, including Gaussian Noise Feature, Feature Masking, Graph Diffusion, and Edge Modification, and they found that Edge Modification performs the best across different datasets (with ground-truth labels on test data to measure the performance); 2) with the help of ground-truth label information on test data, they heuristically set (α, β) as $(0.9, 0.3)$, $(0.1, 0.7)$, $(0.7, 0.1)$, $(0.9, 0.3)$, $(0.7, 0.5)$, $(0.5, 0.5)$ on EAT, WebKB, UAT, Cora, UAI2010, and Citation respectively; 3) similarly, they set $\gamma = 1$ for all datasets; and 4) they also heuristically set $P = 0.2$ for all datasets.

B.0.4 SL-GAD [85]. Different from CoLA, ANEMONE and GRADATE, SL-GAD combines the *contrastive-based* framework and the *generative-based* framework for unsupervised GAD.

First, the design of the *contrastive-based* framework is as follows.

Contrastive Framework—Data Augmentation Module They consider a single graph augmentation operation, namely Random Ego-Nets generation with a fixed size K. Specifically, taking the target node as the center, they employ RWR [57] to generate two different subgraphs as ego-nets with a fixed size K, where K controls the radius of the surrounding contexts. Overall, they result in one critical HP in graph augmentation, namely K. Particularly, they indicate that other augmentation strategies such as attribute masking and edge modification may introduce extra anomalies, while random ego-nets and graph diffusion can augment data without changing the underlying graph semantic information.

Contrastive Framework—Contrast Learning Module They introduce a Multi-View Contrastive Learning module that compare the similarity between node embedding and embedding of sampled sub-graphs in augmented views (namely node-subgraph contrast), leading to two loss terms $\mathcal{L}_{con,1}$ and $\mathcal{L}_{con,2}$ corresponding to two augmented views, respectively. On this basis, they obtain the contrastive objective $\mathcal{L}_{con} = \frac{1}{2}(\mathcal{L}_{con,1} + \mathcal{L}_{con,2})$, which combines the two loss terms with equal weights.

Second, the *generative-based* framework is designed as follows.

Generative Framework They introduce a Generative Attribute Regression module that reconstructs node attributes, with the aim to achieve node-level discrimination, where the encoder is a GCN and the decoder is another GCN. Specifically, they minimise the Mean Square Error between the target node's original and reconstructed attributes in augmented views, leading to two loss terms $\mathcal{L}_{gen,1}$ and $\mathcal{L}_{gen,2}$ corresponding to two augmented views, respectively. Then they combine them with equal weights, leading to the generative objective $\mathcal{L}_{gen} = \frac{1}{2}(\mathcal{L}_{gen,1} + \mathcal{L}_{gen,2})$.

At last, their final optimisation objective is defined as follows:

$$\mathcal{L} = \alpha\mathcal{L}_{con} + \beta\mathcal{L}_{gen},$$

where $\alpha, \beta \in (0, 1]$ are trade-off HPs to balance the importance of two SSL objectives.

HPs Sensitivity & Tuning They conducted sensitive analysis and found that: 1) the performance first increases and then decreases with the increasing of K. For efficiency consideration, they heuristically set the sampled subgraph size $K = 4$ for all datasets; 2) they heuristically fix $\alpha = 1$ for all datasets as they found that this achieves good performance on most datasets (with the help of label information); and 3) the selection of β is high dependent on the specific dataset. Hence, they "fine-tune" the value of β for each dataset via selecting β from $\{0.2, 0.4, 0.6, 0.8, 1.0\}$ with labels.

B.0.5 Sub-CR [75]. Similar to SL-GAD, they also combine the *contrastive-based* framework and the *generative-based* framework for unsupervised GAD.

First, the design of the *contrastive-based* framework is as follows.

Contrastive Framework—Contrast Learning Module They consider two types of data augmentation: 1) subgraph sampling to obtain local augmented views for each node (so-called local view subgraph), 2) graph diffusion plus subgraph sampling (in a sequential order) to obtain global augmented views for each node (so-called global view subgraph). Particularly, they employ RWR [57] to generate a sub-graph with a fixed size K in subgraph sampling. Besides, they apply Persnonalized PageRank to power the graph diffusion [76], wherein the teleport probability α needs to be determined. Overall, they result in two critical HPs in graph augmentation, namely K and α.

Contrastive Framework—Contrast Learning Module This module consists of: 1) intra-view contrastive learning that maximises the agreement between the node and its sub-graph level representations in the local view (with loss term $\mathbf{L}_{intra,1}$), and the agreement between the node and its sub-graph level representations in the global view (with loss term $\mathbf{L}_{intra,2}$), where they combine the local view and global view loss terms with equal weights to obtain the intra-view loss term $\mathbf{L}_{intra} = \mathbf{L}_{intra,1} + \mathbf{L}_{intra,2}$; and 2) inter-view contrastive learning that makes closer the discriminative scores of node-subgraph pairs in local view and global view, leading to the loss term $\mathbf{L}_{inter}$. On this basis, they combine the intra-view loss term and inter-view loss term with equal weights to obtain the multi-view contrastive learning loss term $\mathbf{L}_{con} = \mathbf{L}_{intra} + \mathbf{L}_{inter}$.

Second, the *generative-based* framework is designed as follows.

Generative Framework They introduce a masked Autoencoder-based Reconstruction module, where the encoder is a GCN and the decoder is a multilayer perceptron with *PReLU* activation function, aiming to reconstruct the attributes of the target node based on the attributes of neighbouring nodes in the local view (with loss term $\mathbf{L}_{res,1}$), and in the global view (with loss term $\mathbf{L}_{res,2}$). Next, they combine the local view and global view loss terms with equal weights to obtain the overall reconstruction loss term $\mathbf{L}_{res} = \mathbf{L}_{res,1} + \mathbf{L}_{res,2}$ for each node.

At last, their final optimisation objective is defined as follows:

$$\mathcal{L} = \mathcal{L}_{con} + \gamma\mathcal{L}_{res},$$

where $\gamma \in (0, 1]$ is the trade-off HP to balance the importance of two different SSL objectives.

HPs Sensitivity & Tuning They conducted sensitive analysis and found that: 1) the selection of K is dependent on the specific dataset. However, for efficiency and performance consideration, they heuristically set the sampled subgraph size $K = 4$ for all datasets; 2) they did not discuss the setting of teleport probability α; and 3) they claim that most datasets are not sensitive to the value of γ when $\gamma > 0.4$. Hence, they heuristically set $\gamma = 0.6$ for Cora, Citeseer, Flickr, and BlogCatalog while $\gamma = 0.4$ for PubMed with the help of label information.

B.0.6 CONAD [64]. Similar to SL-GAD and Sub-CR, they also combine the *contrastive-based* framework and the *generative-based* framework for unsupervised GAD.

First, the design of the *contrastive-based* framework is as follows.

Contrastive Framework—Data Augmentation Module They consider four different types of data augmentations, with each type of data augmentation operation corresponding to a specific type of node anomaly. They include 1) edge adding augmentation that connects a node with many other non-connected nodes (structure - high degree), 2) edge removing augmentation that removes most edges of a node (structure - outlying); 3) attribute replacement augmentation that replaces the target node's attributes with another dissimilar node's attributes (attribute - deviated), and 4) attribute scaling augmentation that scales the target node's attributes to much larger or smaller values (attribute - disproportionate); This leads to four HPs p_1, p_2, p_3, p_4, which represent the sampling probability of each augmentation strategy. Moreover, the rate r of augmented anomalies (namely modified nodes) is also a HP.

Contrastive Framework—Contrast Learning Module They consider two different contrast strategies: 1) Siamese contrast $\mathcal{L}_{SC} = \sum_{i\in\mathbf{NM}} d(\mathbf{z}_i, \hat{\mathbf{z}}_i) + \sum_{j\in\mathbf{MM}} max\{0, m - d(\mathbf{z}_j, \hat{\mathbf{z}}_j)\}$ where $d(\mathbf{z}_i, \hat{\mathbf{z}}_i)$ is the distance between embeddings of node i in the original view and in the augmented view. **MM** and **NM** mean the node is modified or non-modified, respectively; 2) Triplet contrast $\mathcal{L}_{TC} = \sum max\{0, m - [d(\mathbf{z}_i, \hat{\mathbf{z}}_j) - d(\mathbf{z}_i, \mathbf{z}_j)]\}$ where $d(\mathbf{z}_i, \mathbf{z}_j)$ is the distance between embeddings of node i and its neighbour j in the original view, and $d(\mathbf{z}_i, \hat{\mathbf{z}}_j)$ is the distance between embeddings of node i in

the original view and its neighbour j in the augmented view. Particularly, the contrastive loss term $\mathcal{L}_{Contr} = \mathcal{L}_{SC}$ or $\mathcal{L}_{Contr} = \mathcal{L}_{TC}$. This module contains a HP, namely the margin m.

Second, the *generative-based* framework is designed as follows.

Generative Framework This framework consists of two components: 1) an attribute autoencoder to reconstruct the node attributes, where the encoder is a GAT [58] and the decoder is another GAT. This leads to the loss term L_A; and 2) a structure autoencoder to reconstruct the structure, where the encoder is a GAT and the decoder is a dot product operation followed by a *sigmoid* function (namely $sigmoid(\mathbf{z}^t\mathbf{z})$). This leads to the loss term L_S. Combining these two loss terms leads to a loss term $L_{Recon} = \lambda L_A + (1-\lambda)L_S$, where $\lambda \in (0,1)$ is a trade-off HP to balance the two reconstruction errors. Unlike SL-GAD and Sub-CR, CONAD requires the whole adjacency matrix and node attribute matrix as input, and thus it can reconstruct the graph structure, making it unsuitable to large graphs. In contrast, SL-GAD and Sub-CR only require subgraphs as inputs, and thus are unable to perform structure reconstruction while being scalable.

At last, the final optimisation objective is defined as follows:

$$\mathcal{L} = \eta\mathcal{L}_{Contr} + (1-\eta)\mathcal{L}_{Recon},$$

where $\eta \in (0,1)$ is the trade-off HP to balance the importance of two SSL objectives.

HPs Sensitivity & Tuning They did not perform sensitivity analysis over the HPs. Instead, 1) They heuristically set the ration of augmented anomalies $r = 0.1$ and $r = 0.2$ for small and large datasets, respectively; 2) The sampling probability of each augmentation strategy is set to $p_i = 0.25$ for $i \in \{1,2,3,4\}$; 3) They heuristically set the margin $m = 0.5$ for all datasets; and 4) They heuristically set the trade-off hyper-parameters $\lambda = 0.9$ and $\eta = 0.7$ for all datasets

B.0.7 DOMINANT [9]. This is arguably the first work that utilises *generative-based* framework and GNNs to perform unsupervised anomaly detection on attribute graphs.

Generative Framework They first employ GCN [27] to obtain node embeddings. Next, they construct two decoders: 1) an attribute decoder, which consists of another GCN, to reconstruct the node attributes, leading to the loss term L_A, and 2) a structure decoder, which is a dot product operation followed by a *sigmoid* function (namely $sigmoid(\mathbf{z}^t\mathbf{z})$), to reconstruct topological structures, leading to the loss term L_S.

At last, their final optimisation objective is defined as follows:

$$\mathcal{L} = \alpha\mathcal{L}_A + (1-\alpha)\mathcal{L}_S,$$

where $\alpha \in (0,1)$ is the trade-off HP to balance the importance of two objectives.

HPs Sensitivity & Tuning Specifically, they found that the AUC performance usually increases first and then decreases with the increasing of α. However, the specific value of α on each dataset is heuristically selected with the help of labels. The HP α is selected from $[0.4, 0.7]$, $[0.4, 0.7]$, $[0.5, 0.8]$ on BlogCatalog, Flickr, and ACM respectively.

B.0.8 AnomalyDAE [12]. Similar to DOMINANT, AnomalyDAE leverages *generative-based* framework and autoencoders (based on GNNs) to perform unsupervised GAD.

Generative Framework AnomalyDAE consists of two components: 1) an attribute autoencoder to reconstruct the node attributes, where the encoder consists of two non-linear feature transform layers and the decoder is simply a dot product operation. This leads to the loss term L_A, and L_A is associated with a penalty HP $\eta > 1$); and 2) a structure autoencoder to reconstruct the structures, where the encoder is based GAT [58] and the decoder is a dot product operation followed by a *sigmoid* function (namely $sigmoid(\mathbf{z}^t\mathbf{z})$). This leads to the loss term L_S, and L_S is associated with a penalty HP $\theta > 1$.

At last, their final optimisation objective is defined as follows:

$$\mathcal{L} = \alpha\mathcal{L}_S + (1-\alpha)\mathcal{L}_A,$$

where $\alpha \in (0,1)$ is the trade-off HP to balance the importance of two objectives.

HPs Sensitivity & Tuning Specifically, they found that the AUC performance usually increases first and then decreases with the increasing of α. However, the specific value of α on each dataset is selected using label information. The HPs (α, η, θ) are heuristically set as $(0.7, 5, 40)$, $(0.9, 8, 90)$, $(0.7, 8, 10)$ on BlogCatalog, Flickr, and ACM respectively.

B.0.9 GUIDE [70]. Similar to AnomalyDAE, GUIDE leverages *generative-based* framework and autoencoders (based on GNNs) to perform unsupervised GAD. Particularly, they consider reconstructing the high-order structures.

Generative Framework GUIDE consists of two components: 1) an attribute autoencoder to reconstruct the node attributes, where the encoder is a GCN and the decoder is another GCN. This leads to the loss term L_A; and 2) a structure autoencoder to reconstruct the high-order structures, where the encoder is a graph node attention network based on [8] and the decoder is another graph node attention layer. This leads to the loss term L_S. Moreover, structure matrix is composed of node motif degrees, which leads to a HP, namely the degree of motifs D.

At last, their final optimisation objective is defined as follows:

$$\mathcal{L} = \alpha\mathcal{L}_A + (1-\alpha)\mathcal{L}_S,$$

where $\alpha \in (0,1)$ is the trade-off HP to balance the importance of two SSL objectives.

HPs Sensitivity & Tuning They mention that the HPs are optimised via a parameter sensitivity analysis experiment for each dataset. Specifically, they found that: 1) the AUC performance usually increases first and then decreases with the increasing of α, and most datasets can achieve a good performance when $0.1 < \alpha < 0.3$. However, the specific value of α on each dataset is selected using labels; and 2) they heuristically set the degree of motifs as $D = 4$.

B.0.10 GAAN [5]. They combine the *generative-based* framework and GAN [16] for unsupervised GAD. Particularly, GAN can be considered as a special case of *contrastive-based* framework.

Contrastive Framework—Data Augmentation Module GAAN employs GAN, which consists of a generator and a discriminator, to generate adversarial samples as augmented views, without involving any HPs.

Contrastive Framework—Contrastive Learning Module For each target node, GAAN computes the sum of cross-entropy losses of its 1-hop neighbouring nodes (where the edge is considered

as from real distribution by the discriminator) as anomaly score, leading to a loss term $\mathcal{L}_D$. In particular, this discriminator loss can be regarded as contrastive loss, and it considers both node attributes and graph structures.

Generative Framework GAAN utilises the generator to reconstruct the node attribute, and employs the reconstruction error to compute anomaly score, leading to a loss term $\mathcal{L}_G$.

At last, their final optimisation objective is defined as

$$\mathcal{L} = \alpha\mathcal{L}_G + (1-\alpha)\mathcal{L}_D,$$

where $\alpha \in [0, 1]$ is the trade-off HP to balance the importance of two objectives

HPs Sensitivity & Tuning Specifically, they found that the AUC performance usually increases first and then decreases with the increasing of α. However, the specific value of α on each dataset is selected using label information. The HP α is heuristically set as 0.2, 0.3, 0.1 on BlogCatalog, Flickr, and ACM respectively.

C PERFORMANCE VARIATIONS UNDER DIFFERENT HP SETTINGS

In this section, we present a comprehensive analysis of the performance exhibited by various semi-supervised learning (SSL) based graph anomaly detection techniques. This evaluation encompasses an extensive array of hyperparameter (HP) configurations and is conducted across multiple benchmark datasets.

Specifically, the results for GAAN [5] is provided in Figure 3, from which we can see huge performance variations under different HP settings. For example, the AUC value can vary from 0.474 to 0.747 if one utilises different HP configurations on dataset CiteSeer (namely by changing the HP α from 0.5 to 0). Moreover, the results for CoLA [40] is provided in Figure 4. Compared to GAAN, CoLA is less sensitive to the setting of HPs, while we can still see mediate performance variations on some datasets (e.g., from 0.693 to 0.733 on Flickr, and from 0.767 to 0.795 on ACM). Besides, Figure 5 shows that DOMINANT is also sensitive to HPs except for the cases where the algorithm is largely underfitted (i.e., on ACM, Flickr and BlogCatalog the loss values change only by 10^{-2} after 400 epochs of training).

Particularly, AnomalyDAE and SL-GAD are very sensitive to HPs as shown in Figures 6 and 11. For example, the performance of AnomalyDAE ranges from 0.702 to 0.941 on CiteSeer, and the performance of SL-GAD vary from 0.787 to 0.920. As shown in Figure 7, CONAD shows similar behaviours except for the cases where CONAD is largely underfitted (namely on ACM) or suffers from OOM errors (namely on Flickr and BlogCatalog). The analysis for GUIDE in Figure 8, GRADATE in Figure 9, and Sub-CR in Figure 10 is similar and conveys the same issues.

D SIMILAR OBSERVATIONS IN OTHER PAPERS

Liu et al. [38] conduct a comprehensive benchmark for unsupervised graph anomaly detection. From their results (note that their experiment setting is slightly different from ours), we can have similar observations as follows:

- **Radar** [29] is not sensitive to hyper-parameters (0.65 VS 0.66 on Cora, 0.99 VS 0.99 on Weibo, 0.55 VS 0.57 on Reddit,

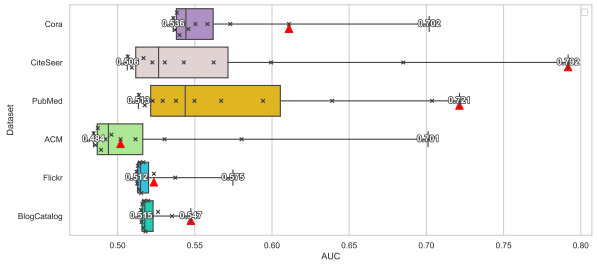

Figure 3: Performance variations over different HP configurations for GAAN [5] on different benchmark datasets.

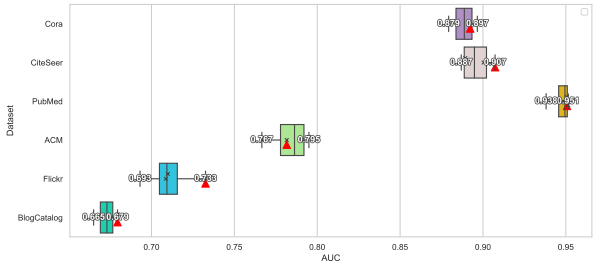

Figure 4: Performance variations over different HP configurations for CoLA [40] on different benchmark datasets.

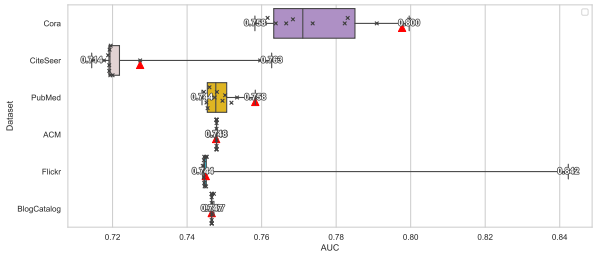

Figure 5: Performance variations over different HP configurations for DOMINANT [9] on different benchmark datasets.

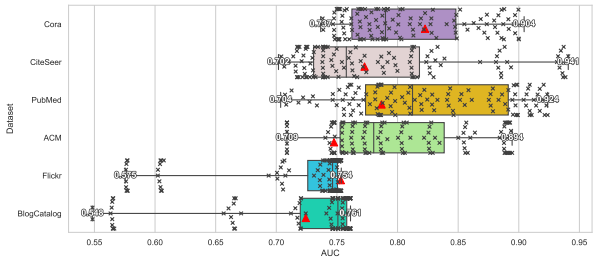

Figure 6: Performance variations over different HP configurations for AnomalyDAE [12] on different benchmark datasets.

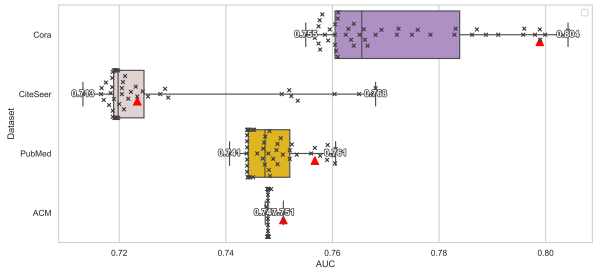

Figure 7: Performance variations over different HP configurations for CONAD [64] on different benchmark datasets.

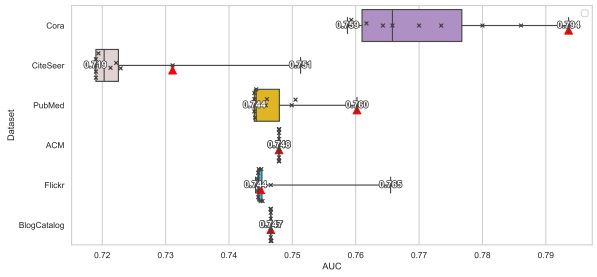

Figure 8: Performance variations over different HP configurations for GUIDE [70] on different benchmark datasets.

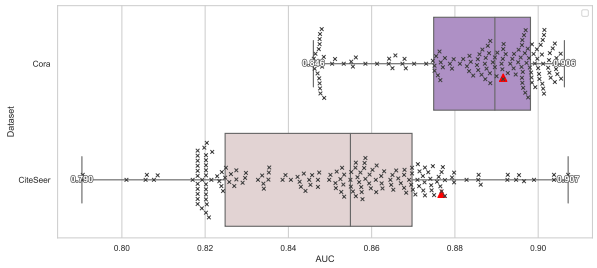

Figure 9: Performance variations over different HP configurations for GRADATE [11] on different benchmark datasets.

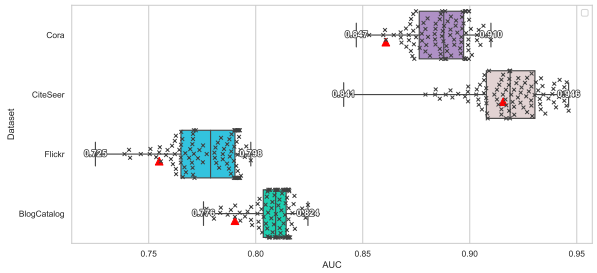

Figure 10: Performance variations over different HP configurations for Sub-CR [75] on different benchmark datasets.

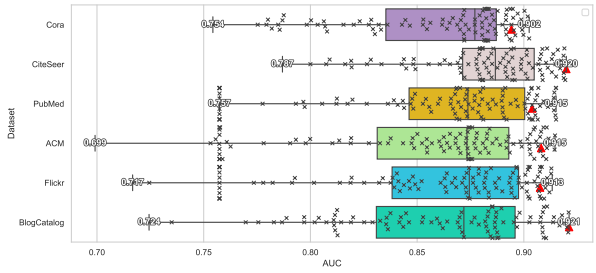

Figure 11: Performance variations over different HP configurations for SL-GAD [85] on different benchmark datasets.

0.52 VS 0.52 on Disney, 0.53 VS 0.53 on Books), but it will suffer from OOM errors for large graphs;

- **ANOMALOUS** [49] is very sensitive to hyper-parameters on some datasets (0.55 VS 0.68 on Cora, 0.99 VS 0.99 on Weibo, 0.55 VS 0.60 on Reddit, 0.52 VS 0.52 on Disney, 0.53 VS 0.53 on Books), and it will suffer from OOM errors for large graphs;
- **DOMINANT** [9] is very sensitive to hyper-parameters on some datasets (0.83 VS 0.84 on Cora, 0.76 VS 0.85 on Flickr, 0.85 VS 0.93 on Weibo, 0.50 VS 0.58 on Books, 0.56 VS 0.56 on Reddit, 0.47 VS 0.55 on Disney)
- **AnomalyDAE** [12] is very sensitive to hyper-parameters on some datasets (0.83 VS 0.85 on Cora, 0.86 VS 0.91 on Amazon, 0.66 VS 0.70 on Flickr, 0.91 VS 0.93 on Weibo, 0.56 VS 0.56 on Reddit, 0.49 VS 0.55 on Disney, 0.54 VS 0.69 on Books);
- **GUIDE** [70] is very sensitive to hyper-parameters on some datasets (0.39 VS 0.53 on Disney, 0.52 VS 0.63 on Books, 0.75 VS 0.78 on Cora), and it will suffer from OOM errors on large graph (including Amazon, Flickr, Weibo, Reddit). It needs much time and memory for training as it employs a graph motif counting algorithm to extract structural information;
- **CONAD** [64] is very sensitive to hyper-parameters on some datasets (0.79 VS 0.84 on Cora, 0.81 VS 0.82 on Amazon, 0.65 VS 0.67 on Flickr, 0.85 VS 0.93 on Weibo, 0.56 VS 0.56 on Reddit, 0.48 VS 0.53 on Disney, 0.52 VS 0.63 on Books).

E SUMMARY OF EXISTING SSL-BASED GRAPH ANOMALY DETECTION METHODS

Existing SSL-based graph anomaly detection methods are summarised in Table 5, which includes the datasets used to test, the core principles of SSL techniques, the involved hyper-parameters (only SSL related ones), and their public implementations.

F SEARCH SPACE APPROXIMATION BASED ON SMBO

F.1 Performance Surrogate Functions

Although discretisation of continuous domains can largely reduce the search space, it is still computationally prohibitive to search the full discretized HP space when the number of HPs is large. Therefore, we learn a regressor $g(\cdot)$ which aims to to learn the

Table 4: SSL-related HPs for different algorithms, where "Range" indicates the tested values in grid search.

Algo	HPs	Range
CoLA [40]	K	{2, 3, 4, 5}
ANEMONE [22]	K	{2, 3, 4, 5}
	α	{0, 0.01, 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 0.99, 1}
GRADATE [11]	P	{0.20}
	α	{0.9}
	β	{0, 0.01, 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 0.99, 1}
	γ	{0, 0.01, 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 0.99, 1}
SL-GAD [85]	K	{2, 3, 4, 5, 6, 7, 8, 9}
	α	{0.01, 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 0.99, 1}
	β	{0.6}
Sub-CR [75]	K	{2, 3, 4, 5, 6, 7, 8, 9}
	α	{0.01}
	γ	{0.01, 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 0.99, 1}
CONAD [75]	r	{0.10}
	$p1$	{0.25}
	$p2$	{0.25}
	$p3$	{0.25}
	$p4$	{0.25}
	m	{0.5}
	λ	{0.01, 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 0.99, 1}
	η	{0.01, 0.5, 0.99, 1}
DOMINANT [9]	α	{0.01, 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 0.99, 1}
AnomalyDAE [12]	α	{0.01, 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 0.99, 1}
	η	{1, 2, 3, 4, 5, 6, 7, 8, 9, 10}
	θ	{10}
GUIDE [70]	D	{4}
	α	{0.01, 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 0.99}
GAAN [5]	α	{0, 0.01, 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 0.99, 1}

mapping from HP settings onto the performance metric (namely the domain of $T(\cdot)$). Note that $g(\cdot)$ should be different for different combinations of graph and graph anomaly detector $[\mathcal{G}, f(\cdot)]$, and we call these functions *performance surrogate functions*. Gaussian Process (GP) [60] is one popular choice for $g(\cdot)$. Based on these *performance surrogate functions*, we can identify promising HPs without running experiments on all possible HPs, which will be illustrated in next subsection.

F.2 SMBO-based Optimisation

Particularly, we leverage Sequential Model-based Optimisation (SMBO [24]) to iteratively and efficiently identify promising HP configurations to evaluate, and finally output the optimal one as follows. Similar idea is also explored in [84].

Initialization Specifically, we first randomly sample a small number of HPs $\boldsymbol{\lambda}_{eval} = \{\boldsymbol{\lambda}_1, \boldsymbol{\lambda}_2, ..., \boldsymbol{\lambda}_J\}$ with $J \ll M$. Second, for each HP, we compute its unsupervised performance metric score $t(\mathcal{G})$, leading to pairs $\{(\boldsymbol{\lambda}_1, t_1(\mathcal{G})), (\boldsymbol{\lambda}_2, t_2(\mathcal{G})),..., (\boldsymbol{\lambda}_J, t_J(\mathcal{G}))\}$. Third, we employ these pairs to train a specific *performance surrogate function* $g(\cdot)$.

Iteration For each iteration, we leverage $g(\cdot)$ to predict the performance for a sampled HP $\boldsymbol{\lambda}_j$, denoted as $\eta_j = g(\boldsymbol{\lambda}_j)$. Moreover, we also utilise $g(\cdot)$ to predict the uncertainty around the prediction of $\boldsymbol{\lambda}_j$, denoted as $\sigma_j = \sigma[g(\boldsymbol{\lambda}_l|\boldsymbol{\lambda}_l \in \boldsymbol{\lambda}_{sample})]$. Note that $\boldsymbol{\lambda}_{sample}$ is different from $\boldsymbol{\lambda}_{eval}$, and it is a finite number of HPs that is randomly sampled from the full HP space before discretisation. Next, we utilise a so-called *acquisition function* $h(\cdot)$, which can make a trade-off between predicted performance and uncertainty, to select the most promising HP to evaluate. Particularly, we leverage Expected Improvement (EI) [24] as the *acquisition function* since it has shown prominent performances in many studies [80]. Under the mild Gaussian assumption, the EI value of HP setting $\boldsymbol{\lambda}_j$ has the following closed-form expression:

$$EI(g(\boldsymbol{\lambda}_j)) = \left[\phi(\hat{\eta}_j) + \hat{\eta}_j \cdot \Phi(\hat{\eta}_j)\right] \sigma_j, \tag{7}$$

where $\hat{\eta}_j = \frac{\eta_j - \eta^*_{eval}}{\sigma_j}$ if $\sigma_j > 0$ and $\hat{\eta}_j = 0$ otherwise. Moreover, $\phi(\cdot)$ and $\Phi(\cdot)$ denote the probability density function and the cumulative distribution function of standard Gaussian distribution, respectively. In addition, η^*_{eval} is the highest prediction performance on $\boldsymbol{\lambda}_{eval}$ so far. For each iteration, the most promising HP can be obtained as follows:

$$\boldsymbol{\lambda}^* = \underset{\boldsymbol{\lambda}_j \in \boldsymbol{\lambda}_{sample}}{\arg\max}\ h(g(\boldsymbol{\lambda}_j)), \tag{8}$$

where $g(\cdot) = g^{(current)}(\cdot)$ is the surrogate function in the current iteration, which can output the most promising HP $\boldsymbol{\lambda}^*$ to evaluate. On this basis, we apply $f(\boldsymbol{\lambda}^*)$ on graph $\mathcal{G}$ to obtain a vector of anomaly scores $\boldsymbol{s}^*$, followed by inputting $\boldsymbol{s}^*$ into Equation 3 to obtain the performance metric score t^*. At last, we update the evaluation HP set as $\boldsymbol{\lambda}_{eval} = \boldsymbol{\lambda}_{eval} \cup \boldsymbol{\lambda}^*$, and retrain $g(\cdot)$ with the updated pairs $\{(\boldsymbol{\lambda}_1, t_1(\mathcal{G})), (\boldsymbol{\lambda}_2, t_2(\mathcal{G})),..., (\boldsymbol{\lambda}_J, t_J(\mathcal{G}))\}..., (\boldsymbol{\lambda}^*, t^*)\}$. Additionally, we update η^*_{eval} using the updated $\boldsymbol{\lambda}_{eval}$.

G ADDITIONAL RESULTS

G.1 Performance Gain Over Max AUC

Furthermore, we define *performance gain over max AUC* as

$$\frac{\text{CSM}(AUC) - \max(AUC)}{\max(AUC)}, \tag{9}$$

where $\max(AUC)$ represents the maximum of the obtained *AUC* values for the GAD algorithm with different configurations. Thus, if the value of this metric is close to zero, the GAD algorithm configured with our selected HPs can approximately achieve the best possible performance.

From Table 6, one can see that *AutoGAD* can result in *performance gain over max AUC* larger than −2.5% in 8 out 10 algorithms, implying that the HP values selected by *AutoGAD* can achieve

Table 5: Summary of existing SSL-based graph anomaly detection methods.

Method	Venue	Datasets	SSL methods	Hyperparameters	Code
CoLA [40]	TNNLS'21	Cora, Citeseer, Pubmed, BlogCatalog, Flickr, ACM, ogbn-arxiv	Node-Sub CL	Random walk length (K)	Github, PyGOD
ANEMONE [22]	CIKM'21	Cora, Citeseer, PubMed	Node-Node CL, Node-Sub CL	Ego-Net size (K), Combination weights	Github
GRADATE [11]	AAAI'23	EAT, WebKB, UAT, Cora, UAI2010, Citation	Node-Node CL, Node-Sub CL, Sub-Sub CL	Proportion of modified edges (P), Combination weights	Github
SL-GAD [85]	TKDE'21	Cora, Citeseer, PubMed, ACM, Flickr, BlogCataLog	Node-Sub CL, Attribute Recon	Random walk length (K), Combination weights	Github
Sub-CR [75]	IJCAI'22	Cora, Citeseer, PubMed, Flickr, BlogCataLog	Node-Sub CL, Attribute Recon	Random walk length (K), Teleport probability α, Combination weights	Github
CONAD [64]	PAKDD'22	Amazon, Flickr, Enron, Facebook, Twitter	Node-Sub CL, Attribute Recon, Structure Recon	Augmentation sampling probabilities, combination weights	PyGOD, Github
DOMINANT [9]	ICDM'19	ACM, Flickr, BlogCataLog	Attribute Recon, Structure Recon	Combination weight	PyGOD
AnomalyDAE [12]	ICASSP'20	ACM, Flickr, BlogCataLog	Attribute Recon, Structure Recon	Penalty HPs, Combination weights	PyGOD
GUIDE [70]	BigData'21	Cora, Citation, Pubmed, ACM, DBLP	Attribute Recon, Structure Recon	Combination weight	PyGOD
GAAN [5]	CIKM'20	ACM, Flickr, BlogCataLog	Attribute Recon, Discr Loss	Combination weight	PyGOD

Table 6: Performance gain over max AUC defined as $\frac{\mathbf{CSM}(AUC)-\mathbf{max}(AUC)}{\mathbf{max}(AUC)}$. Results are averaged on five independent runs. CSM is contrast score margin defined in Equation 3, while OOM, OOR and gray cells convey the same concepts as in Table 1.

Dataset	CoLA	ANEMONE	GRADATE	SL-GAD	Sub-CR	CONAD	DOMINANT	A-DAE	GUIDE	GAAN
Cora [53]	-0.6%	-1.1%	-3.2%	-0.4%	-3.8%	-0.8%	-0.2%	-10.0%	0%	-2.7%
CiteSeer [53]	-0.5%	-1.1%	-1.9%	-0.3%	-4.5%	-4.9%	-4.8%	-21.8%	-3.6%	-0.9%
PubMed [53]	-0.1%	-0.2%	OOM	-0.4%	OOM	-0.2%	-0.1%	-14.6%	0%	0%
ACM [56]	-1.5%	-5.4%	OOM	-0.4%	OOM	0%	0%	-16.4%	0%	-2.8%
Flickr [72]	-1.3%	-3.9%	OOR	-1.2%	-5.9%	OOM	-10.2%	-0.1%	-8.4%	-0.5%
BlogCataLog [72]	-0.3%	-0.4%	OOR	-0.2%	-4.1%	OOM	0%	-4.8%	0%	-0.3%
Average	-0.7%	-2.0%	-2.5%	-0.5%	-4.6%	-2.0%	-1.7%	-11.3%	-1.2%	-1.2%

performances that are very close to optimal performances. Particularly, the *performance gains over max AUC* for SL-GAD [85] and GAAN [5] are -0.5% and -1.2% respectively, which shows that *AutoGAD* is highly effective for these methods while they show large performance variations (18.4% and 19.5% respectively).

G.2 The Setting of k and Sensitivity Analysis

The selection of the value of k in our experiments acknowledges the varying anomaly ratios across different datasets, implying that k should ideally differ to reflect the unique characteristics of each dataset. We operated under the assumption that the anomaly ratio within a dataset is approximately known, a premise that aligns with real-world anomaly detection tasks where some prior knowledge about the frequency of anomalies is often available.

As shown in Figure 12, we conducted a sensitivity analysis on k to assess the stability of AutoGAD against deviations from the true anomaly ratio. The findings from this analysis indicate that the effectiveness of AutoGAD remains stable as long as k is not drastically distant from the actual anomaly ratio, reinforcing the practical applicability of our approach even when exact anomaly proportions are not precisely determined.

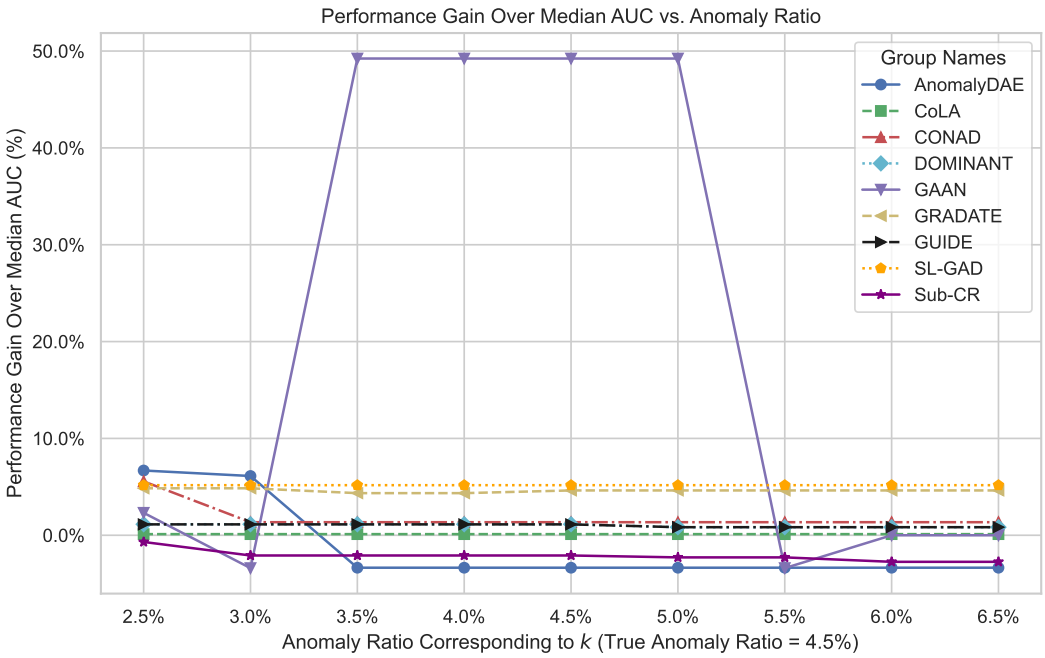

Figure 12: Sensitivity analysis of k (for our proposed AutoGAD) on dataset CiteSeer with all investigated SSL-based GAD algorithms. It can be seen that AutoGAD remains stable as long as k is not drastically distant from the actual anomaly ratio (namely 4.5%) all for SSL-based GAD algorithms.