Trustworthy anomaly detection for smart manufacturing

Li, Z.

Chapter 7

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

Authors: **Zhong Li**, Yuhang Wang, Matthijs van Leeuwen
Manuscript submitted to Data Mining and Knowledge Discovery.

Abstract

Self-supervised learning (SSL) is an emerging paradigm that exploits supervisory signals generated from the data itself, and many recent studies have leveraged SSL to conduct graph anomaly detection. However, we empirically found 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. 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", 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. 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.

7.1 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 [241]. GAD has numerous successful applications, e.g., in finance fraud detection [271], fake news detection [272], system fault diagnosis [232], and network intrusion detection [273]. 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 [98]. 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 labeled data. However, the process of labeling 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 [274], and recent studies have shown its usefulness for node anomaly detection [275, 276, 277, 278, 279, 280, 281, 282].

Graph SSL can be roughly divided into *generative*, *contrastive*, and *predictive* methods [283]. First, *generative* methods such as DOMINANT [284], GUIDE [279], and AnomalyDAE [275] aim to detect graph anomalies by reconstructing ('generating') the adjacency matrix and/or the node attribute matrix. Next, *contrastive* methods such as CoLA [278], ANEMONE [277], GRADATE [285], and Sub-CR [286] 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 [276] 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 [287]. 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)

C3. How to combine the contrast losses at different levels? (i.e., how to set their combination weights?)

Further, a recent study [276] 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 [288, 289, 290] 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 labeled 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 labeled data, corroborating the findings in [290].

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 [291], implying that the optimal SSL strategy is in general data-dependent [288, 289]. Therefore, utilizing a unified and pre-fixed combination weights and/or HPs in SSL strategies for all graphs can result in sub-optimal performance.

In the latter case, practitioners pick the optimal combination weights and other hyperparameter values following a 'hyperparameters sensitivity analysis' using labeled 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 [292, 293]. Specifically, label information should never be used (whether implicitly or explicitly) in an unsupervised learning scenario. As shown in Figure 7.1, label leakage leads to huge overestimation of the model's performance, which is also corroborated in [294] by comparing the max and average performance with different hyperparameter configurations (cf. Appendix 7.8 for more details).

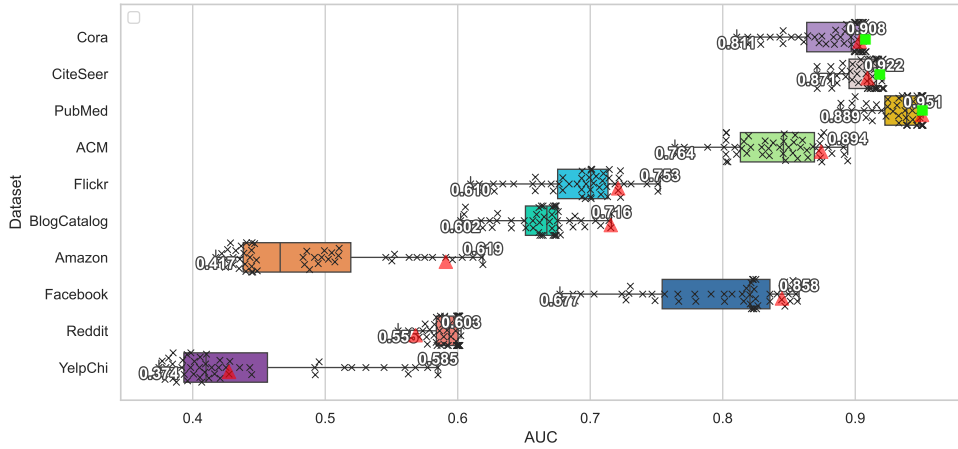

Figure 7.1: Large performance variations (here measured by AUC) over different hyperparameter configurations for ANEMONE [277] on various benchmark datasets. Using labeled data and only reporting the best possible performance leads to severe overestimation of model performance. For instance, the green squares on Cora, CiteSeer, and PubMed are reported by [277] (the other datasets were not used). Similar results are observed for other algorithms (see Appendix 7.8 for details). The red triangles represent the results obtained by our internal evaluation strategy, showing its potential for automating truly unsupervised anomaly detection.

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 [295] aims to automatically combine several predefined graph augmentations via learning a

sampling distribution, where the augmentations themselves are not learnable. Meanwhile, AD-GCL [296] uses learnable edge dropping augmentation and AutoGCL [297] 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, all these automated graph augmentation methods are agnostic to the downstream tasks, making the learned graph embeddings sub-optimal 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 [298], 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 analyze the internal evaluation strategy to prove why it is effective and empirically demonstrate this.

Overall, our main contributions can be summarized 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 10 benchmark datasets, demonstrating the effectiveness of our approach.

7.2 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.

7.2.1 Anomaly Detection on Attributed Graphs

Early methods for node anomaly detection in static attributed graphs, such as AMEN [299], Radar [300], and Anomalous [301], 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 [284], AnomalyDAE [275], and GUIDE [279], 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.

7.2.2 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 [274]. It has achieved promising performance on typical graph mining tasks such as representation learning [302] and graph classification [303]. [278] first applied SSL to the GAD problem. Their proposed method CoLA performs single scale comparison (node-subgraph) for anomaly detection. However, ANEMONE [277] 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 [285] and M-MAG [304] combines various multi-contrast objectives, namely node-node, node-subgraph, and subgraph-subgraph contrasts for node anomaly detection. To achieve better performance, SL-GAD [276] combines multi-view contrastive learning and generative attribute regression, while Sub-CR [286] combines multi-view contrastive learning and graph autoencoder. Finally, CONAD [280] considers both contrastive learning and generative reconstruction for better node anomaly detection.

7.2.3 Automated Self-Supervised Learning

Seminal work on *automated data augmentation for images* [305, 306] was followed by work improving [306] via faster searching mechanisms [307, 308, 309] or advanced

optimization methods [310, 311, 312].

In the context of *automated data augmentation for graphs*, related work exists on graph representation learning [313, 296, 314, 287, 297, 295], node classification [315, 316], and graph-level classification [317, 318, 297]. For example, JOAO [295] learns the sampling distribution of a set of predefined graph augmentations. AD-GCL [296] designs a learnable edge dropping augmentation and employs adversarial training strategy, and AutoGCL [297] proposes a learnable graph view generator that learns a probability distribution over the node-level augmentations. Further, [317] augment graph data samples, while [318] perturb the representation vector. However, these methods focus on other typical graph learning tasks and it is unclear how to use them for unsupervised GAD.

7.2.4 Automated Anomaly Detection

Recent studies [319, 320, 321, 322] pointed out that unsupervised anomaly detection methods tend to be highly sensitive to the values of their hyperparameters (HPs). For example, [319] show that a 10x performance difference is observed for LOF [173] by changing the number of nearest neighbors. Even more, [321] indicate that deep anomaly detection methods suffer more from such HP sensitivity issues. Concretely, [322] demonstrate that RAE [323] 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; [320] present an overview. Inspired by [320, 322], we subdivide existing approaches into two main categories:

- *Supervised evaluation* methods which require ground-truth labels although anomaly detection algorithms are unsupervised. Methods include PyODDS [324], TODS [325], AutoOD [326], and AutoAD [327];
- *Unsupervised evaluation* methods which do not require ground-truth labels. They include
  - randomly selecting an HP configuration;
  - selecting an HP configuration via an internal evaluation strategy [328, 329, 330, 331];
  - averaging the outputs of a set of randomly selected HP configurations [332];

– meta-learning based methods [333, 334, 322].

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

7.3 Problem Statement

We utilize 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 7.1 (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 anomaly 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 optimization problem for unsupervised graph anomaly detection (dubbed HPO for GAD):

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 $\{\boldsymbol{\lambda}_1, ..., \boldsymbol{\lambda}_k, ..., \boldsymbol{\lambda}_K\}$, where $\boldsymbol{\lambda}_k \in \mathbf{\Lambda}_k$ can be discrete or continuous, we aim to find*

$$\underset{\boldsymbol{\lambda}_1\in\mathbf{\Lambda}_1,...,\lambda_k\in\mathbf{\Lambda}_k,...,\lambda_K\in\mathbf{\Lambda}_K}{\arg\max} \; \mathit{Metric}\left[f(\boldsymbol{\lambda}_1, ..., \boldsymbol{\lambda}_k, ..., \boldsymbol{\lambda}_K; \mathcal{G})\right], \tag{7.1}$$

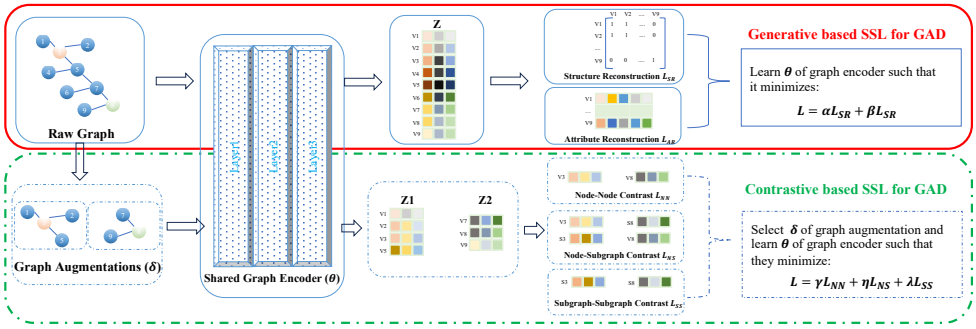

Figure 7.2: Self-supervised learning based graph anomaly detection methods can be subdivided into *generative based* methods and *contrastive based* methods. A *generative based* method generally involves *graph structure reconstruction* and *node attributes reconstruction*. A *contrastive based* method usually consists of a *graph augmentation module* and a *contrastive learning module*.

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

7.4 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.

7.4.1 Existing SSL for "Unsupervised" GAD

Figure 7.2 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 utilizes one augmentation function $f(\boldsymbol{\delta})$ to produce augmented samples, which usually include 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.

7.4.2 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 7.1).

Due to space constraints and to enhance readability, we revisit three representative SSL-based GAD algorithms in the main paper, including a contrastive method: ANEMONE [277], a generative method: AnomalyDAE [275], and a combined contrastive and generative method: SL-GAD [276].

Revisiting ANEMONE

ANEMONE [277] is 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 [335] 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.

Revisiting AnomalyDAE

AnomalyDAE [275] is 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 [336] 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 optimization 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.

Revisiting SL-GAD

SL-GAD [276] is 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 [335] 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 minimize 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 optimization 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.

Other SSL-based GAD methods

Due to space constraints, the analyses of other SSL-based GAD methods, including CoLA [278] , GRADATE [285], Sub-CR [286], CONAD [280], DOMINANT [284], GUIDE [279], and GAAN [337], are given in Appendix 7.8. 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 [277], AnomalyDAE [275], and SL-GAD [276].

7.4.3 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 Chapter 7.6 for experiment settings).

Table 7.1: 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 marked as 'UNF' denote persistent underfitting of algorithms, even after reaching the maximum allowed training epochs (e.g., loss values change by less than 10^{-2} after 400 epochs). 'NAN' indicates execution errors caused by excessive NaN values; these cases are excluded from further analysis. Refer to Chapter 7.6 for details on the experimental setup.

	Cora	CiteSeer	PubMed	ACM	Flickr	BlogCatalog	Amazon	Facebook	Reddit	YelpChi	**Average**
CoLA	1.1%	1.7%	1.6%	4.2%	3.3%	4.7%	18.0%	3.4%	2.9%	31.9%	7.3%
ANEMONE	8.9%	6.6%	6.3%	11.3%	16.9%	16.8%	32.7%	23.9%	8.9%	37.7%	17.0%
GRADATE	6.9%	14.1%	OOM	OOM	OOR	OOR	5.9%	22.9%	OOR	OOR	12.5%
SL-GAD	17.4%	16.2%	19.5%	17.7%	16.3%	23.4%	25.4%	47.8%	21.8%	OOR	22.8%
Sub-CR	15.1%	8.3%	OOM	OOM	9.8%	6.3%	28.6%	20.3%	OOM	OOM	14.7%
CONAD	5.8%	7.0%	2.3%	UNF	OOM	OOM	17.3%	27.3%	9.7%	40.7%	15.7%
DOMINANT	5.1%	6.0%	1.9%	UNF	UNF	UNF	12.4%	19.1%	8.4%	34.5%	12.5%
A-DAE	19.1%	25.3%	23.8%	20.8%	23.6%	14.3%	48.1%	64.9%	NAN	NAN	30.0%
GUIDE	4.8%	4.8%	1.8%	UNF	8.5%	UNF	11.5%	18.6%	8.0%	34.4%	11.6%
GAAN	28.1%	30.0%	30.3%	25.6%	10.1%	7.2%	13.1%	72.6%	11.9%	11.5%	24.0%

As shown in Figure 7.1, for a typical run with different hyperparameter configurations, the performance of ANEMONE [277] can vary strongly on each of the ten datasets. Other SSL-based GAD algorithms exhibit similar behavior; extensive results and analysis are deferred to Appendix 7.8 for space reasons.

For an in-depth yet compact analysis, Table 7.1 presents average results over five independent runs when varying SSL-related hyperparameter values. Specifically, CoLA [278], GUIDE [279], DOMINANT [284], GRADATE [285], and Sub-CR [286] demonstrate moderate performance variations (namely between 7.3% and 14.7% on average). Meanwhile, CONAD [280], ANEMONE [277], SL-GAD [276], GAAN [337], and AnomalyDAE [275] suffer from large performance variations (namely ranging from 15.7% to 30.0% on average). From Chapter 7.4.2 and Appendix 7.8, 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.

7.5 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 7.1—should not use any ground-truth label information, simply because this is not available in a truly unsupervised setting. We therefore propose to utilize 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. Chapter 7.5.2 describes a straightforward approach using discretization and grid search that works well in practice, as shown in the next section.

7.5.1 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.*

Contrast Score Margin

The metric that we use is Contrast Score Margin [338], which was introduced before but not for graph anomaly detection, and is defined as

$$T(f) = \frac{\hat{\mu}_{\mathbf{O}} - \hat{\mu}_{\mathbf{I}}}{\sqrt{\frac{1}{k}(\hat{\delta}^2_{\mathbf{O}} + \hat{\delta}^2_{\mathbf{I}})}}, \tag{7.2}$$

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.

Analysis

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

Theorem 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 $\mathbf{y} \in \mathbf{X} \backslash \mathbf{O}$, we have $f(\mathbf{x}) < f(\mathbf{y})$, then we may obtain $\mu_{\mathbf{O}} - \mu_{\mathbf{I}} < 0$ or $\mu_{\mathbf{O}} - \mu_{\mathbf{I}} \approx 0$. □

Improvements and Remarks

In practice we observed that Equation 7.2 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 behaviors as anomalies). Therefore, we propose to modify Equation 7.2 as follows:

$$T(f) = \frac{\hat{\mu}_{\mathbf{O}} - \tilde{\mu}_{\mathbf{I}}}{\sqrt{\hat{\delta}^2_{\mathbf{O}} + \tilde{\delta}^2_{\mathbf{I}}}}, \tag{7.3}$$

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 7.2. This is because the true labels are not accessible, and thus we utilize the pseudo-labels to identify the top-k anomalous and the top-k normal objects. However, the pseudo-labels of the top-k "pseudo-normal" objects may not be reliable due to the two facts stated above.

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 (please refer to Appendix 7.8 for empirical evidence of this).

7.5.2 Discretization and Grid Search

To find the optimal hyperparameter configuration, we first perform discretization of the continuous search space and then conduct grid search. The corresponding pseudo-code is provided in Algorithm 6, with a detailed explanation presented below.

Discretization of Continuous Search Space (Lines 1–2). To make the overall search process feasible, we discretize the hyperparameter space. Assume we are given a GAD algorithm $f(\cdot)$ with its set of hyperparameters $\boldsymbol{\lambda} \in \boldsymbol{\Lambda}$. Without loss of generality, we assume there are L different hyperparameters and let $\boldsymbol{\lambda} = \{\lambda^{(1)}, \lambda^{(2)}, \ldots, \lambda^{(L)}\}$,

Algorithm 6 Grid Search for Anomaly Detector Hyperparameter Optimization

Input: Graph anomaly detection algorithm $f(\cdot)$, graph $\mathcal{G}$, hyperparameter domains $\mathbf{\Lambda} = \{\mathbf{\Lambda}^{(1)}, \ldots, \mathbf{\Lambda}^{(L)}\}$, internal evaluation function $T(\cdot)$
Output: Best hyperparameter configuration $\boldsymbol{\lambda}_{\text{best}}$
1: Discretize each continuous domain $\mathbf{\Lambda}^{(l)}$ into a finite set if necessary
2: Generate hyperparameter search set $\boldsymbol{\lambda}_{\text{search}} = \{\boldsymbol{\lambda}_1, \ldots, \boldsymbol{\lambda}_M\}$ where $M = \prod_{l=1}^{L} |\mathbf{\Lambda}^{(l)}|$
3: Initialize best score $t_{\text{best}} \leftarrow -\infty$ and best configuration $\boldsymbol{\lambda}_{\text{best}} \leftarrow \emptyset$
4: **for** each $\boldsymbol{\lambda}_m \in \boldsymbol{\lambda}_{\text{search}}$ **do**
5: Compute anomaly scores $\boldsymbol{s}_m(\mathcal{G}) = f(\boldsymbol{\lambda}_m; \mathcal{G})$
6: Compute evaluation score $t_m(\mathcal{G}) = T(\boldsymbol{s}_m(\mathcal{G}))$
7: **if** $t_m(\mathcal{G}) > t_{\text{best}}$ **then**
8: Update $t_{\text{best}} \leftarrow t_m(\mathcal{G})$
9: Update $\boldsymbol{\lambda}_{\text{best}} \leftarrow \boldsymbol{\lambda}_m$
10: **end if**
11: **end for**
12: **return** $\boldsymbol{\lambda}_{\text{best}}$

where each $\lambda^{(l)} \in \mathbf{\Lambda^{(l)}}$ for $l = 1, 2, \ldots, L$. If a hyperparameter domain $\mathbf{\Lambda^{(l)}}$ is continuous, we discretize it into a finite set of values (with cardinality $|\mathbf{\Lambda^{(l)}}|$). This results in M possible hyperparameter configurations, represented by the set $\boldsymbol{\lambda}_{\text{search}} = \{\boldsymbol{\lambda}_1, \ldots, \boldsymbol{\lambda}_m, \ldots, \boldsymbol{\lambda}_M\}$, where $\boldsymbol{\lambda}_m = \{\lambda_m^{(1)}, \lambda_m^{(2)}, \cdots, \lambda_m^{(L)}\}$ and $M = \prod_{l=1}^{L} |\mathbf{\Lambda^{(l)}}|$.

Grid Search (Lines 3–11). Once the hyperparameter search space is discretized, we apply grid search to evaluate each configuration. For each hyperparameter configuration $\boldsymbol{\lambda}_m \in \boldsymbol{\lambda}_{\text{search}}$, we run the GAD algorithm $f(\boldsymbol{\lambda}_m)$ on the given graph $\mathcal{G}$ to produce a vector of anomaly scores $\boldsymbol{s}_m(\mathcal{G}) = f(\boldsymbol{\lambda}_m; \mathcal{G})$. These scores are evaluated using an internal unsupervised performance metric $T(\cdot)$ (with Equation 7.3) to yield a final score $t_m(\mathcal{G}) = T(\boldsymbol{s}_m(\mathcal{G}))$. The configuration that maximizes $T(\cdot)$ is selected as the optimal values of hyperparameters.

Note that more advanced strategies than grid search, such as SMBO-based optimization [339], could be employed (see Appendix 7.8 for an example). However, these methods often introduce additional hyperparameters (whose tuning may be non-trivial), which contradicts our goal of automated anomaly detection.

7.6 Experiments

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

RQ1 How sensitive are existing SSL-based GAD methods to the values of their hy-

Table 7.2: Summary of datasets: anomalies in Cora, CiteSeer, PubWeb, ACM, BlogCatalog, and Flickr are synthetically injected following established methods [277, 278], while Amazon, Facebook, Reddit, and YelpChi contain real-world anomalies.

Dataset	#Nodes	#Edges	#Attributes	#Anomalies
Cora [340]	2708	11060	1433	138(5.1%)
CiteSeer [340]	3327	4732	3703	150(4.5%)
PubMed [340]	19717	44338	500	150(2.5%)
ACM [341]	16484	71980	8337	600(3.6%)
BlogCataLog [342]	5196	171743	8189	300(5.8%)
Flickr [342]	7575	239738	12407	450(5.9%)
Amazon [343]	10244	175608	25	693(6.7%)
Facebook [344]	1081	55104	576	27(2.5%)
Reddit [345]	10984	168016	64	366(3.3%)
YelpChi [346]	24741	49315	32	1217(4.9%)

perparameters?

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.

7.6.1 Datasets

We use three popular citation networks, namely Cora, Citeseer, and Pubmed [340] 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 [277] and CoLA [278] to inject structure and contextual anomalies. Note that [294] have slightly modified this injection procedure. Following [347], we also consider four commonly-used graph datasets with real anomalies: Amazon [343], Facebook [344], Reddit [345], and YelpChi [346]. The resulting datasets are summarized in Table 7.2.

7.6.2 Baselines

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

- Generative methods: DOMINANT [284], AnomalyDAE [275], GUIDE [279], GAAN [337];
- Contrastive methods (and some also generative): CoLA [278], ANEMONE [277], GRADATE [285], SL-GAD [276], Sub-CR [286], CONAD [280].

Particularly, the SSL-related HPs for each GAD algorithm and their discretized search spaces are given in Table 7.6 in the Appendix. These GAD methods are further summarized in Table 7.7 in the Appendix.

7.6.3 Evaluation Metrics

To evaluate the effectiveness of various GAD algorithms, we utilize the ROC-AUC metric [348] (*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:

$$\frac{\max(AUC) - \min(AUC)}{\max(AUC)}, \tag{7.4}$$

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

$$\frac{\mathrm{CSM}(AUC) - \min(AUC)}{\min(AUC)}, \tag{7.5}$$

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

$$\frac{\mathrm{CSM}(AUC) - \mathrm{median}(AUC)}{\mathrm{median}(AUC)}, \tag{7.6}$$

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.

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

$$\frac{\text{CSM}(AUC) - \max(AUC)}{\max(AUC)}, \tag{7.7}$$

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.

7.6.4 Software and Hardware

All algorithms are implemented in Python 3.8 (using PyTorch [109] and PyTorch Geometric [110] 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 GitHub[1].

7.6.5 Results and Analysis

Table 7.3: Performance gain over minimal AUC defined as $\frac{\text{CSM}(AUC)-\min(AUC)}{\min(AUC)}$. Results are averaged on five independent runs. CSM is contrast score margin defined in Equation 7.3, while OOM, OOR, UNF, and NAN convey the same meanings as in Table 7.1.

	Cora	CiteSeer	PubMed	ACM	Flickr	BlogCatalog	Amazon	Facebook	Reddit	YelpChi	**Average**
CoLA	0.5%	1.2%	1.6%	2.8%	2.2%	4.7%	22%	1.8%	1.5%	2.5%	4.1%
ANEMONE	8.6%	5.9%	6.6%	6.8%	15.8%	19.7%	44.7%	28.1%	4.0%	11.9%	15.2%
GRADATE	4.0%	14.3%	OOM	OOM	OOR	OOR	4.3%	29.7%	OOR	OOR	13.1%
SL-GAD	21.2%	19.1%	23.7%	21.3%	18.2%	30.4%	13.0%	16.3%	15.8%	OOR	19.9%
Sub-CR	16.2%	4.3%	OOM	OOM	4.3%	2.4%	19.2%	25.3%	OOM	OOM	12.0%
CONAD	5.4%	2.3%	2.1%	UNF	OOM	OOM	6.5%	18.3%	2.4%	24.3%	8.8%
DOMINANT	5.2%	1.3%	1.8%	UNF	UNF	UNF	13.7%	14.9%	0.8%	28.0%	9.4%
A-DAE	11.3%	4.6%	11.9%	5.6%	30.8%	32.2%	67.3%	114.6%	NAN	NAN	34.8%
GUIDE	5.0%	1.2%	1.8%	UNF	0.1%	UNF	9.4%	14.3%	3.8%	28.8%	8.1%
GAAN	7.7%	34.1%	43.6%	5.6%	1.3%	6.6%	12.4%	77.9%	0.4%	14.5%	20.4%

We answer the research questions as follows:

RQ1: Sensitivity of SSL-based GAD methods to HPs

The results are summarized in Table 7.1 for five independent runs. Typical runs are depicted in Figure 7.1 and in Figures 7.5-7.13 in Appendix 7.8. We briefly analyzed the

[1] `https://github.com/ZhongLIFR/AutoGAD2024`

Table 7.4: Performance gain over median AUC defined as $\frac{\text{CSM}(AUC)-\text{median}(AUC)}{\text{median}(AUC)}$. Results are averaged on five independent runs. CSM is contrast score margin defined in Equation 7.3, while OOM, OOR, UNF, and NAN convey the same meanings as in Table 7.1. For enhanced readability, cells are color-coded based on their values, as specified in the legend.

Dark Orange	Light Orange	Light Green	Dark Green	Grey
$(-\infty, -5.0\%]$	$(-5.0\%, 0.0\%)$	$[0.0\%, 5.0\%)$	$[5.0\%, +\infty)$	Excluded

	Cora	CiteSeer	PubMed	ACM	Flickr	BlogCatalog	Amazon	Facebook	Reddit	YelpChi	**Average**
CoLA	-0.1%	0.1%	0.2%	-0.7%	0.6%	2.0%	15.4%	0.1%	-0.2%	-3.3%	1.4%
ANEMONE	0.3%	1.0%	1.5%	-0.8%	2.0%	7.2%	26.4%	3.5%	-2.4%	1.6%	4.0%
GRADATE	-0.6%	4.0%	OOM	OOM	OOR	OOR	0.7%	18.3%	OOR	OOR	5.6%
SL-GAD	3.3%	3.7%	4.8%	4.3%	2.8%	5.0%	-1.4%	-31.6%	-2.0%	OOR	-1.2%
Sub-CR	-1.6%	-0.4%	OOM	OOM	-3.2%	-2.2%	2.6%	9.8%	OOM	OOM	0.8%
CONAD	4.0%	1.2%	1.5%	UNF	OOM	OOM	-3.7%	2.5%	-3.1%	1.5%	0.6%
DOMINANT	3.7%	0.5%	1.3%	UNF	UNF	UNF	4.4%	-1.8%	-3.0%	4.8%	1.4%
A-DAE	2.9%	-3.0%	-2.9%	-4.1%	0.9%	-3.4%	2.6%	0.7%	NAN	NAN	-0.8%
GUIDE	3.8%	0.6%	1.5%	UNF	-0.3%	UNF	2.1%	-2.3%	0.2%	5.3%	1.4%
GAAN	2.8%	27.5%	35.6%	3.5%	0.7%	4.7%	-2.4%	-45.6%	-1.3%	-0.5%	2.5%

Table 7.5: Performance gain over maximal AUC defined as $\frac{\text{CSM}(AUC)-\max(AUC)}{\max(AUC)}$. Results are averaged on five independent runs. CSM is contrast score margin defined in Equation 7.3, while OOM, OOR, and NAN convey the same meanings as in Table 7.1.

	Cora	CiteSeer	PubMed	ACM	Flickr	BlogCatalog	Amazon	Facebook	Reddit	YelpChi	**Average**
CoLA	-0.6%	-0.5%	-0.1%	-1.5%	-1.3%	-0.3%	0%	-1.7%	-1.5%	-30.2%	-3.8%
ANEMONE	-1.1%	-1.1%	-0.2%	-5.4%	-3.9%	-0.4%	-2.6%	-2.6%	-5.3%	-30.3%	-5.3%
GRADATE	-3.2%	-1.9%	OOM	OOM	OOR	OOR	-1.8%	0.0%	OOR	OOR	-1.7%
SL-GAD	-0.4%	-0.3%	-0.4%	-0.4%	-1.2%	-0.2%	-15.8%	-39.4%	-9.4%	OOR	-7.5%
Sub-CR	-3.8%	-4.5%	OOM	OOM	-5.9%	-4.1%	-15.3%	-0.2%	OOM	OOM	-5.6%
CONAD	-0.8%	-4.9%	-0.2%	UNF	OOM	OOM	-11.9%	-14.0%	-7.6%	-26.3%	-9.3%
DOMINANT	-0.2%	-4.8%	-0.1%	UNF	UNF	UNF	-0.6%	-7.1%	-7.8%	-16.2%	-5.3%
A-DAE	-10.0%	-21.8%	-14.6%	-16.4%	-0.1%	-4.8%	-18.5%	-26.3%	NAN	NAN	-14.1%
GUIDE	0%	-3.6%	0%	UNF	-8.4%	UNF	-3.1%	-7.1 %	-4.6%	-15.4%	-5.3%
GAAN	-22.6%	-6.7%	0%	-21.6%	-8.9%	-1.2%	-2.6%	-53.7%	-11.6%	-0.9%	-13.0%

results in Chapter 7.4.3; more detailed analyses are given in Appendix 7.8. To recall, five out of ten algorithms show moderate performance variations, while the remaining five algorithms demonstrate large performance variations when the values of SSL-related HPs are varied. In other words, SSL-based GAD methods are (sometimes highly) sensitive to hyperparameter values.

RQ2: Effectiveness of AutoGAD in tuning SSL-related HPs

The results are summarized in Tables 7.3, 7.4 and 7.5 for five independent runs, while Figure 7.1 and Figures 7.5-7.13 depict typical runs. We have the following main observations:

1) From Table 7.3, one can see that *AutoGAD* can result in moderate *performance gain over minimal AUC* (namely between 4.1% and 13.1% on average) for CoLA,

GUIDE, CONAD, DOMINANT, Sub-CR, and GRADATE. Recall that five of these algorithms (including CoLA, GUIDE, DOMINANT, GRADATE, and Sub-CR) exhibit moderate performance variations, ranging from 7.3% to 14.7% on average. Moreover, *AutoGAD* leads to large *performance gain over minimal AUC* (namely between 6.8% and 24.1% on average) for the remaining four algorithms, which suffer from large performance variations (namely between 15.1% and 34.8% 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 Table 7.4, one can see that *AutoGAD* can result in positive *performance gain over median AUC* in 8 out 10 algorithms (ranging from 0.6% to 5.6% 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 GRADATE [285], ANEMONE [277], and GAAN [337] are 5.6%, 4.0%, and 2.5% respectively, which shows that *AutoGAD* is highly effective for these methods.

3) From Table 7.5, one can see that *AutoGAD* can result in *performance gain over max AUC* larger than -10% in 8 out 10 algorithms, implying that the HP values selected by *AutoGAD* can achieve performances that are comparable to optimal performances. For instance, the *performance gains over max AUC* for GRADATE and SL-GAD are -1.7% and -7.5% respectively, which shows that *AutoGAD* is highly effective for these methods while they show moderate or large performance variations (12.5% and 22.8% respectively).

4) Following the above observations, we check the details in Figure 7.13 for SL-GAD, Figure 7.11 for GRADATE, and Figure 7.1 for ANEMONE. For SL-GAD and GRADATE, *AutoGAD* often selects HP values better than 90% of randomly selected HPs values on most datasets. For ANEMONE, the HP values selected by *AutoGAD* often outperform 75% of randomly selected HP values.

Sensitivity Analysis

Sensitivity to k. 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

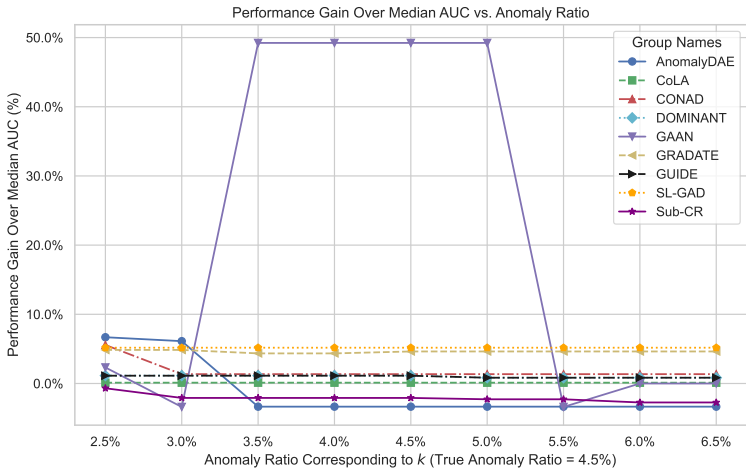

Figure 7.3: Sensitivity analysis of k (for our proposed AutoGAD) on dataset CiteSeer with all investigated SSL-based GAD algorithms. It can be seen that AutoGAD remains stable as long as k is not drastically distant from the actual anomaly ratio (namely 4.5%) all for SSL-based GAD algorithms.

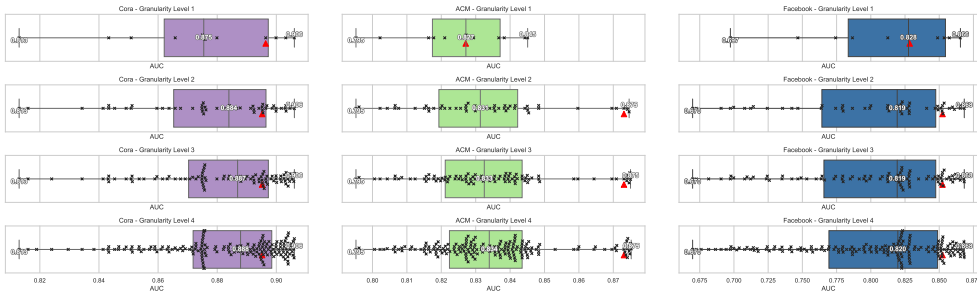

Figure 7.4: Performance of AutoGAD across different granularity levels of search grids using ANEMONE on the Cora, ACM, and Facebook datasets. Similar trends were observed for other anomaly detectors and datasets, which are omitted for brevity.

that aligns with real-world anomaly detection tasks where some prior knowledge about the frequency of anomalies is often available.

As shown in Figure 7.3, 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.

Sensitivity to the Granularity of the Search Grid. Acknowledging the significance of search space granularity in the performance of AutoGAD, we conduct a sensitivity analysis by varying the granularity levels of the search grids in grid search. Figure 7.4 presents representative results using ANEMONE [277] on the Cora, ACM, and Facebook datasets with four levels of search granularity, as follows:

- Granularity Level 1: $\alpha \in \{0, 0.2, 0.4, 0.6, 0.8, 1\}$, $K \in \{2, 4\}$;
- Granularity Level 2: $\alpha \in \{0, 0.01, 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 0.99, 1\}$, $K = \{2, 3, 4, 5\}$;
- Granularity Level 3: $\alpha \in \{0, 0.01, 0.05, 0.1, 0.15, 0.2, 0.25, 0.3, 0.35, 0.4, 0.45, 0.5, 0.55, 0.6, 0.65, 0.7, 0.75, 0.8, 0.85, 0.9, 0.95, 0.99, 1\}, K \in \{2, 3, 4, 5\}$;
- Granularity Level 4: $\alpha \in \{0, 0.01, 0.025, 0.05, 0.075, 0.1, 0.125, 0.15, 0.175, 0.2, 0.225, 0.25, 0.275, 0.3, 0.325, 0.35, 0.375, 0.4, 0.425, 0.45, 0.475, 0.5, 0.525, 0.55, 0.575, 0.6, 0.625, 0.65, 0.675, 0.7, 0.725, 0.75, 0.775, 0.8, 0.825, 0.85, 0.875, 0.9, 0.925, 0.95, 0.975, 0.99, 1\}, K \in \{2, 3, 4, 5, 6, 7\}$.

The results indicate that finer search grids tend to improve the performance of AutoGAD. This is expected, as the optimal value achievable in a finer search grid cannot be worse than that in a coarser grid. Similar observations were made for other anomaly detection methods and datasets, which are omitted here for brevity.

7.7 Alternative Strategies and Discussion

Internal evaluation strategies aim to assess the quality of a model based solely on internal information, without relying on external information such as ground-truth labels. Internal information can typically be derived from two sources: 1) the input samples, such as feature values of instances in tabular data or node attributes in graph

data; or 2) the anomaly scores generated by an anomaly detection model. Beyond the Contrast Score Margin [338] discussed in this paper, additional internal evaluation strategies exist for unsupervised model selection in anomaly detection. According to [298], these strategies can be categorized as *stand-alone* or *consensus-based* internal evaluation strategies; we will next discuss each category.

7.7.1 Stand-alone Internal Evaluation Strategies

Stand-alone strategies rely solely on input samples or individual anomaly detection methods (or models with specific HP configurations in our setting) and their output anomaly scores. Key methods include:

- **IROES** [349, 330] quantifies the separability of each input sample, assuming that a good anomaly detection model assigns high anomaly scores to highly separable samples. However, separability scores are defined only for tabular data, making extension to graph data non-trivial. Additionally, computing separability scores is computationally expensive, posing challenges for large datasets.
- **Mass-Volume and Excess-Mass** [328] use statistical tools to measure the quality of an anomaly scoring function. These methods operate on the raw input samples rather than anomaly scores and assume that anomalies occur in the distribution's tail. However, they are restricted to tabular data and are not applicable to graph data.
- **Clustering Validation Metrics** [350] assume that an anomaly detector divides input samples into two clusters: abnormal and normal. Clustering validation metrics, such as the Xie-Beni index [351], are then used to evaluate performance. While clustering coefficients on graphs could be analogous [352], these metrics are computationally expensive, particularly for large datasets.

7.7.2 Consensus-based Internal Evaluation Strategies

Consensus-based strategies assess the agreement among multiple anomaly detection models (or the same model with varying HP configurations in our setting). Key methods include:

- **UDR** [353] assumes that good HP configurations yield consistent results under different random initializations, while poor configurations do not. [298] repur-

posed UDR to select among heterogeneous anomaly detectors, assuming that good detectors produce consistent results across HP configurations.

- **Model Centrality** [354] hypothesizes that good models are close to the optimal model and thus to each other.
- **Model Centrality by HITS** [355] follows a similar hypothesis but employs a different computation approach.
- **Unsupervised Anomaly Detection Ensembling** [298] infers pseudo anomaly labels by aggregating outputs from a predefined subset of good models. However, this method is less feasible in our setting as there is no such pre-defined good models.

Two challenges remain when utilizing these strategies in our setting: 1) validating the underlying assumptions, which often lack theoretical justification, and 2) addressing their computational expenses, as consensus-based methods require pairwise comparisons. In contrast, Contrast Score Margin is computationally efficient, as it operates on anomaly scores rather than on raw data points and it avoids pairwise comparisons.

7.7.3 Discussion and Future Work

Although [298] demonstrated that many internal evaluation strategies perform suboptimally for selecting heterogeneous anomaly detectors, we hypothesize that some can be valuable for hyperparameter tuning within a single anomaly detection model. However, this is beyond the scope of this paper and is left for future work. The primary objectives of this paper are twofold:

- We highlight flaws in existing studies on using SSL for unsupervised graph anomaly detection. Specifically, we:
    1. Review these studies, showing that most tune HPs arbitrarily or selectively.
    2. Demonstrate empirically, through extensive experiments, that these methods are highly sensitive to HP settings. Consequently, we argue that these methods may suffer from label information leakage under unsupervised learning settings, leading to overstated performance in practical scenarios where label-based tuning is inaccessible.

- We propose an initial solution to these issues by utilizing and improving the Contrast Score Margin. This internal evaluation metric was selected for two reasons:

    1. It operates on anomaly scores rather than on raw data points and avoids pairwise computations, making it computationally efficient and suitable for large datasets.
    2. Theoretical guarantees for its properties are provided by Theorem 1, which may not hold for other internal evaluation strategies.

This paper does not aim to provide a perfect solution to the issues mentioned above. Instead, our goal is to spark interest in the research community to address these challenges. Unlike [298], we do not aim to conduct a comprehensive review and evaluation of internal evaluation strategies for SSL-based graph anomaly detection, as this requires significant computational resources and in-depth analysis. Nevertheless, we aim to explore this direction in future work by considering and potentially repurposing the internal evaluation strategies reviewed in [298]. We have described a more advanced search strategy than grid search, namely SMBO-based optimization [339], in Appendix 7.8, without experimental evaluation. This is because this method introduces additional hyperparameters and their tuning is non-trivial, contradicting our goal of automated anomaly detection. Other advanced hyper-parameter tuning methods [356, 357, 358] to speed up the search are possible, and we leave their explorations for future work.

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

Appendix A: Pitfalls in Existing Methods (Full Analysis)

A.1 CoLA

Particularly, CoLA [278] 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 [335] 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.

A.2 ANEMONE

ANEMONE [277] 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 [335] 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.

A.3 GRADATE

GRADATE [285] 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 in-

formation 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.

A.4 SL-GAD

Different from CoLA, ANEMONE and GRADATE, SL-GAD [276] 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 [335] 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 minimize 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 optimization 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.

A.5 Sub-CR

Similar to SL-GAD, Sub-CR [286] also 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—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 [335] to generate a sub-graph with a fixed size K in subgraph sampling. Besides, they apply Persnonalized PageRank to power the graph diffusion [359], 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 maximizes 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 per-

ceptron with *PReLU* activation function, aiming to reconstruct the attributes of the target node based on the attributes of neighboring 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 optimization 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.

A.6 CONAD

Similar to SL-GAD and Sub-CR, CONAD [280] also 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 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 neighbor 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 neighbor 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 [336] 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 optimization 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

A.7 DOMINANT

DOMINANT [284] is arguably the first work that utilizes *generative-based* framework and GNNs to perform unsupervised anomaly detection on attribute graphs.

Generative Framework They first employ GCN [360] 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 optimization 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.

A.8 AnomalyDAE

Similar to DOMINANT, AnomalyDAE [275] 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 [336] 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 optimization 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.

A.9 GUIDE

Similar to AnomalyDAE, GUIDE [279] 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 [361] 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 optimization 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 optimized 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 $D = 4$.

A.10 GAAN

GAAN [337] combines the *generative-based* framework and GAN [362] 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 neighboring 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 utilizes 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 optimization 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.

Appendix B: 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 [337] is provided in Figure 7.5, 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 utilizes different HP configurations on dataset CiteSeer (namely by changing the HP α from 0.5 to 0). Moreover, the results for CoLA [278] is provided in Figure 7.6. Compared to GAAN, CoLA is less sensitive to the setting of HPs, while we can still see moderate 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 7.7 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 7.8 and 7.13. 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.

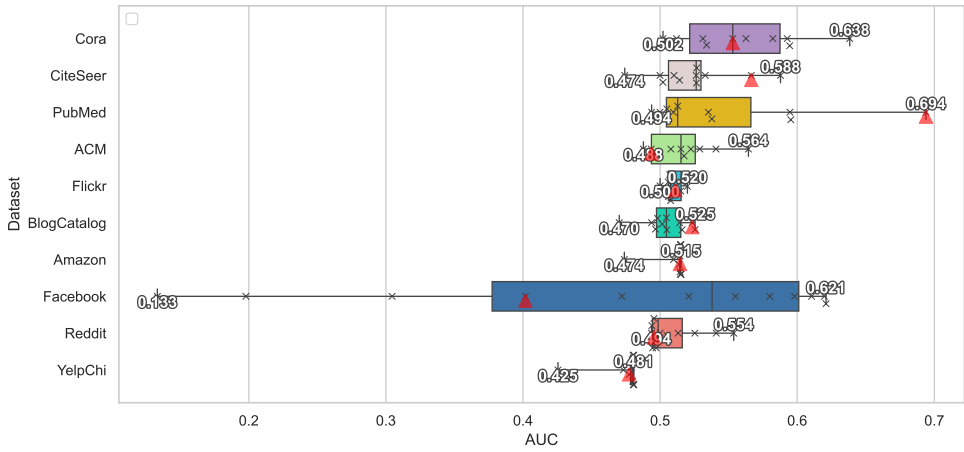

Figure 7.5: Performance variations over different HP configurations for GAAN [337] on different benchmark datasets.

As shown in Figure 7.9, CONAD shows similar behaviors 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 7.10, GRADATE in Figure 7.11, and Sub-CR in Figure 7.12 is similar and conveys the same issues.

Appendix C: Similar Observations in Other Papers

[294] 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 by comparing the average AUC vs max AUC:

- **Radar** [300] 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, 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** [301] 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** [284] 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

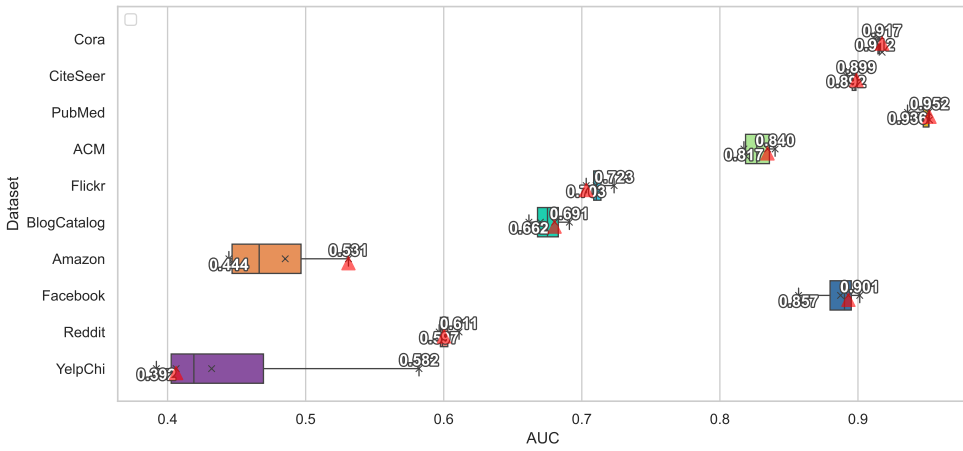

Figure 7.6: Performance variations over different HP configurations for CoLA [278] on different benchmark datasets.

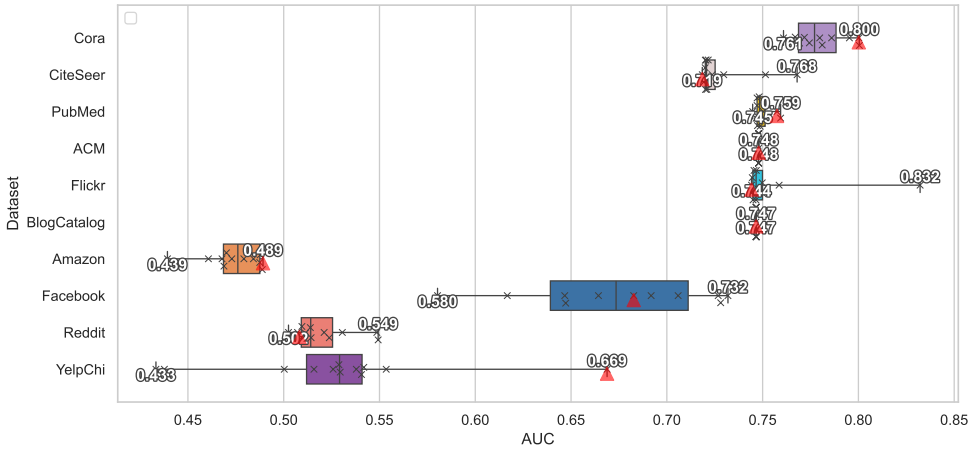

Figure 7.7: Performance variations over different HP configurations for DOMINANT [284] on different benchmark datasets.

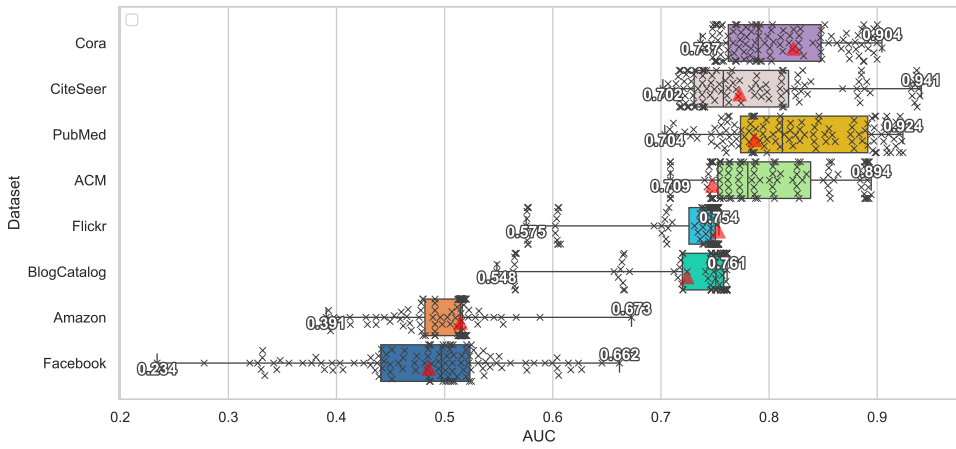

Figure 7.8: Performance variations over different HP configurations for AnomalyDAE [275] on different benchmark datasets.

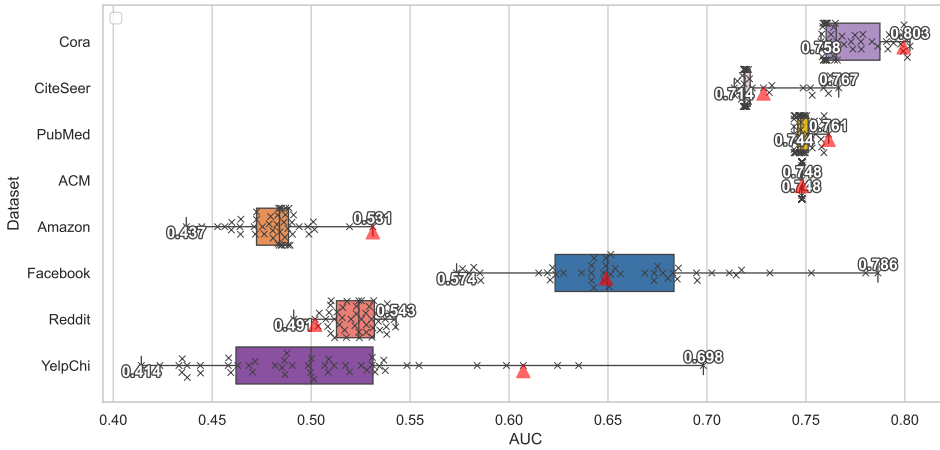

Figure 7.9: Performance variations over different HP configurations for CONAD [280] on different benchmark datasets.

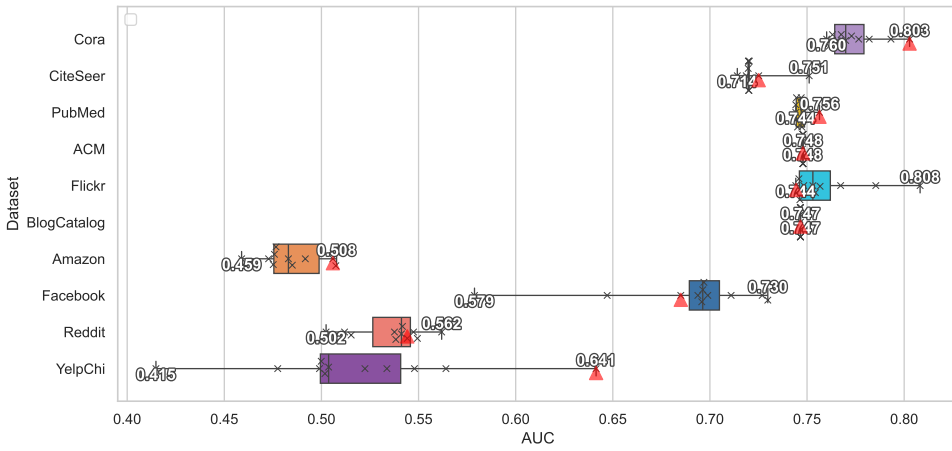

Figure 7.10: Performance variations over different HP configurations for GUIDE [279] on different benchmark datasets.

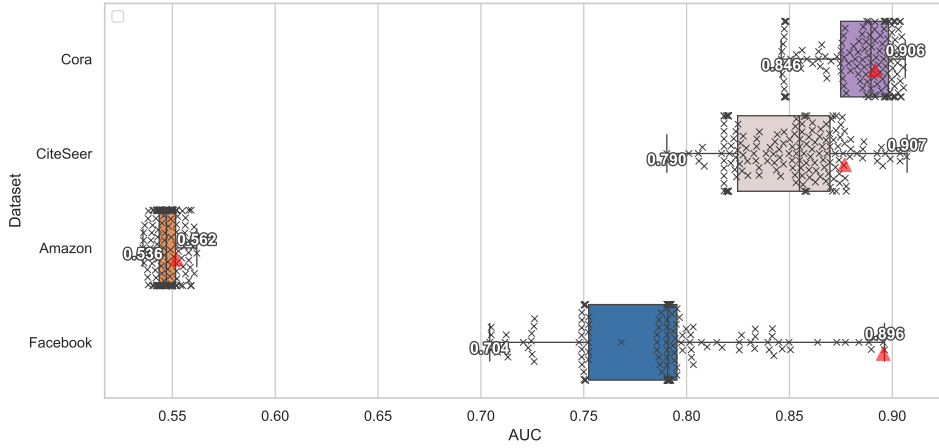

Figure 7.11: Performance variations over different HP configurations for GRADATE [285] on different benchmark datasets.

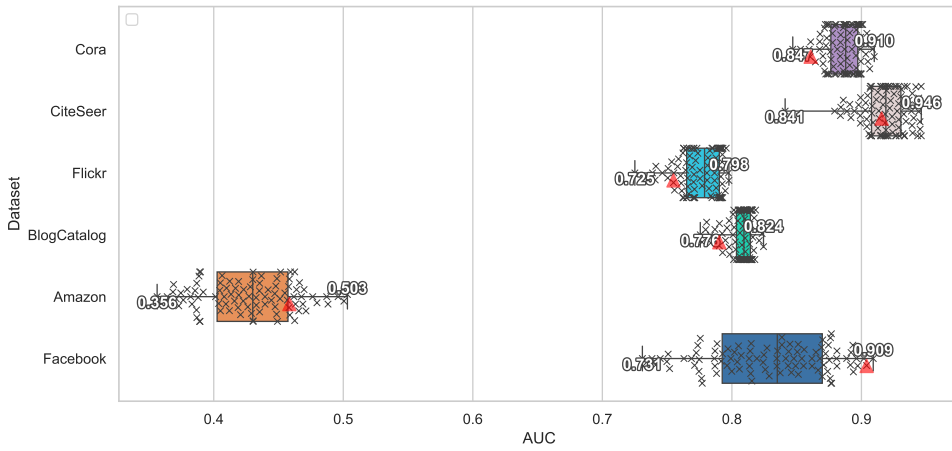

Figure 7.12: Performance variations over different HP configurations for Sub-CR [286] on different benchmark datasets.

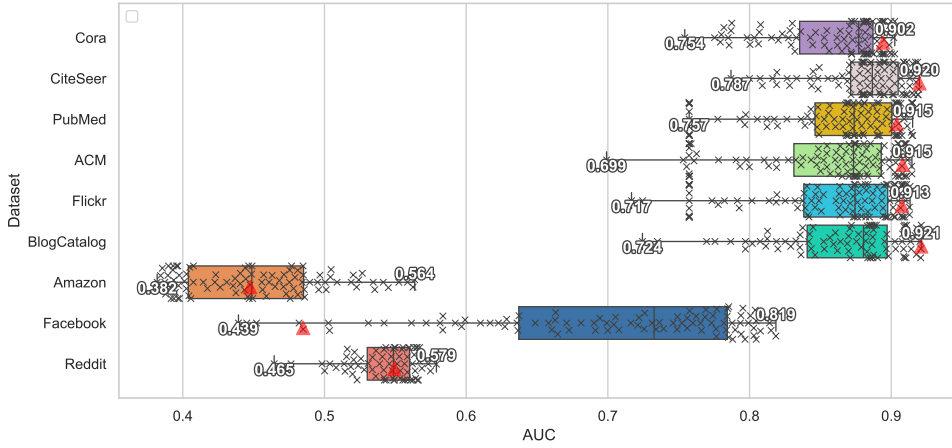

Figure 7.13: Performance variations over different HP configurations for SL-GAD [276] on different benchmark datasets.

on Books, 0.56 VS 0.56 on Reddit, 0.47 VS 0.55 on Disney)

- **AnomalyDAE** [275] 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** [279] 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** [280] 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).

Appendix D: Summary of existing SSL-based graph anomaly detection methods

Existing SSL-based graph anomaly detection methods are summarized in Table 7.7, 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.

Appendix E: Search Space Approximation based on SMBO

Performance Surrogate Functions

Although discretization 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 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) [363] is one popular choice for $g(\cdot)$. Based on these *performance*

Table 7.6: SSL-related HPs for different algorithms, where "**Range**" indicates the tested values in grid search.

Algo	HPs	Range
CoLA [278]	K	{2, 3, 4, 5}
ANEMONE [277]	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 [285]	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 [276]	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 [286]	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 [286]	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 [284]	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 [275]	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 [279]	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 [337]	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}

Table 7.7: Summary of existing SSL-based graph anomaly detection methods.

Method	Venue	Datasets	SSL methods	Hyperparameters	Code
CoLA [278]	TNNLS'21	Cora, Citeseer, Pubmed, BlogCatalog, Flickr, ACM, ogbn-arxiv	Node-Sub CL	Random walk length (K)	Github, PyGOD
ANEMONE [277]	CIKM'21	Cora, Citeseer, PubMed	Node-Node CL, Node-Sub CL	Ego-Net size (K), Combination weights	Github
GRADATE [285]	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 [276]	TKDE'21	Cora, Citeseer, PubMed, ACM, Flickr, BlogCataLog	Node-Sub CL, Attribute Recon	Random walk length (K), Combination weights	Github
Sub-CR [286]	IJCAI'22	Cora, Citeseer, PubMed, Flickr, BlogCataLog	Node-Sub CL, Attribute Recon	Random walk length (K), Teleport probability α, Combination weights	Github
CONAD [280]	PAKDD'22	Amazon, Flickr, Enron, Facebook, Twitter	Node-Sub CL, Attribute Recon, Structure Recon	Augmentation sampling probabilities, combination weights	PyGOD, Github
DOMINANT [284]	ICDM'19	ACM, Flickr, BlogCataLog	Attribute Recon, Structure Recon	Combination weight	PyGOD
AnomalyDAE [275]	ICASSP'20	ACM, Flickr, BlogCataLog	Attribute Recon, Structure Recon	Penalty HPs, Combination weights	PyGOD
GUIDE [279]	BigData'21	Cora, Citation, Pubmed, ACM, DBLP	Attribute Recon, Structure Recon	Combination weight	PyGOD
GAAN [337]	CIKM'20	ACM, Flickr, BlogCataLog	Attribute Recon, Discr Loss	Combination weight	PyGOD

surrogate functions, we can identify promising HPs without running experiments on all possible HPs, which will be illustrated in next subsection.

SMBO-based Optimization

Particularly, we leverage Sequential Model-based Optimization (SMBO) [339] to iteratively and efficiently identify promising HP configurations to evaluate, and finally output the optimal one as follows. Similar idea is also explored in [364].

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 utilize $g(\cdot)$ 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 discretization. Next, we utilize 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) [339] as the *acquisition function* since it has shown prominent performances in many studies [322]. Under the mild Gaussian assumption, the EI value of HP setting $\boldsymbol{\lambda}_j$ has the following closed-form expression:

$$EI(g(\boldsymbol{\lambda}_j)) = [\phi(\hat{\eta}_j) + \hat{\eta}_j \cdot \Phi(\hat{\eta}_j)]\,\sigma_j, \tag{7.8}$$

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:

$$\boldsymbol{\lambda}^* = \underset{\boldsymbol{\lambda}_j \in \boldsymbol{\lambda}_{sample}}{\arg\max}\ h(g(\boldsymbol{\lambda}_j)), \tag{7.9}$$

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 7.3 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 $\boldsymbol{\lambda}_{eval}$.

Appendix F: AutoGAD for Selecting Heterogeneous Anomaly Detectors

To evaluate the effectiveness of AutoGAD in selecting heterogeneous anomaly detectors, we compute the Pearson Correlation Coefficient between the highest improved CSM scores (based on Eq. 7.3) and the corresponding AUC scores for all anomaly detectors on each individual dataset.

As shown in Figure 7.14, the results reveal that AutoGAD's CSM score does not effectively predict the true performance (AUC) of heterogeneous anomaly detectors. Specifically, on the Cora dataset, the Pearson correlation is very weak (0.070), indicating almost no relationship between the CSM score and AUC. On the Amazon dataset, the correlation is negative (-0.488), suggesting that higher CSM scores are, in fact, associated with lower AUC values in many cases. This weak or inverse correlation demonstrates that AutoGAD's scoring mechanism may not be suitable for selecting the

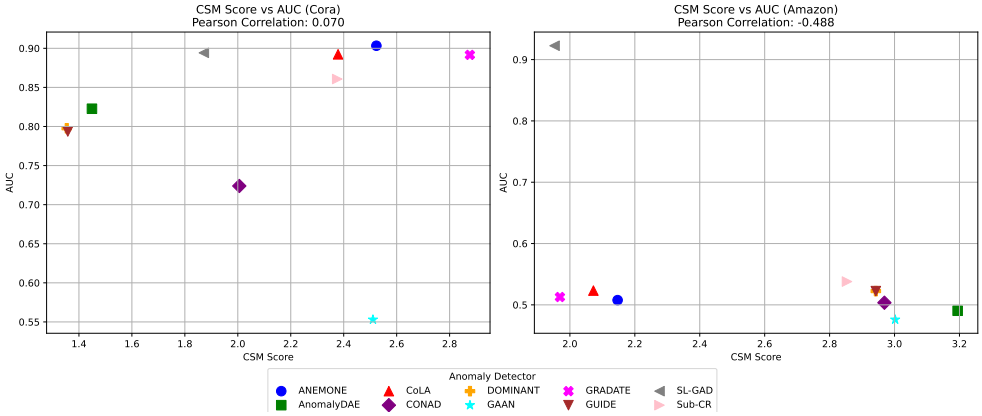

Figure 7.14: Performance of AutoGAD in selecting heterogeneous anomaly detectors on selected datasets (results on other datasets are similar and thus omitted).

best-performing anomaly detectors, as it fails to consistently align with true detector performance. Notably, detectors such as SL-GAD, which achieve high AUC, do not consistently receive high CSM scores, further underscoring the discrepancy. In summary, these findings suggest that AutoGAD's current approach to ranking anomaly detectors is unreliable and may require significant revisions to improve its predictive accuracy.