# UniGAD: Unifying Multi-level Graph Anomaly Detection

**Yiqing Lin**[1*]**, Jianheng Tang**[2,3]**, Chenyi Zi**[3]**, H.Vicky Zhao**[1]**, Yuan Yao**[2]**, Jia Li**[2,3†]

[1]Tsinghua University
[2]Hong Kong University of Science and Technology
[3]Hong Kong University of Science and Technology (Guangzhou)
`linyq20@mails.tsinghua.edu.cn, jtangbf@connect.ust.hk,`
`barristanzi666@gmail.com, vzhao@tsinghua.edu.cn, {yuany,jialee}@ust.hk`

## Abstract

Graph Anomaly Detection (GAD) aims to identify uncommon, deviated, or suspicious objects within graph-structured data. Existing methods generally focus on a single graph object type (node, edge, graph, etc.) and often overlook the inherent connections among different object types of graph anomalies. For instance, a money laundering transaction might involve an abnormal account and the broader community it interacts with. To address this, we present UniGAD, the first unified framework for detecting anomalies at node, edge, and graph levels jointly. Specifically, we develop the Maximum Rayleigh Quotient Subgraph Sampler (MRQSampler) that unifies multi-level formats by transferring objects at each level into graph-level tasks on subgraphs. We theoretically prove that MRQSampler maximizes the accumulated spectral energy of subgraphs (i.e., the Rayleigh quotient) to preserve the most significant anomaly information. To further unify multi-level training, we introduce a novel GraphStitch Network to integrate information across different levels, adjust the amount of sharing required at each level, and harmonize conflicting training goals. Comprehensive experiments show that UniGAD outperforms both existing GAD methods specialized for a single task and graph prompt-based approaches for multiple tasks, while also providing robust zero-shot task transferability. All codes can be found at https://github.com/lllyyq1121/UniGAD.

## 1 Introduction

Graph Anomaly Detection (GAD) involves identifying a minority of uncommon graph objects that significantly deviate from the majority within graph-structured data [17, 2]. These anomalies can manifest as abnormal nodes, unusual relationships, irregular substructures within the graph, or entire graphs that deviate significantly from others. GAD has many practical applications in various contexts, including the identification of bots and fake news on social media [3, 1, 4, 30], detection of sensor faults and internet invasions in IoT networks [8, 14], and prevention of fraudsters and money laundering activities in transaction networks [19, 55]. The mainstream GAD models originate from the Graph Neural Networks (GNNs), which have recently gained popularity for mining graph data [52, 24, 57, 16]. To address the specific challenges of graph anomalies such as label imbalance [34, 31], relation camouflage [12, 38], and feature heterophily [50, 15], numerous adaptations of standard GNNs have been proposed [9, 67, 13, 40, 39, 10, 53, 41, 62].

However, existing GAD approaches typically focus on a single type of graph object, such as node-level or graph-level anomaly detection, often overlooking the inherent correlations between different types

---

[*]Work done as a visiting student at Hong Kong University of Science and Technology.
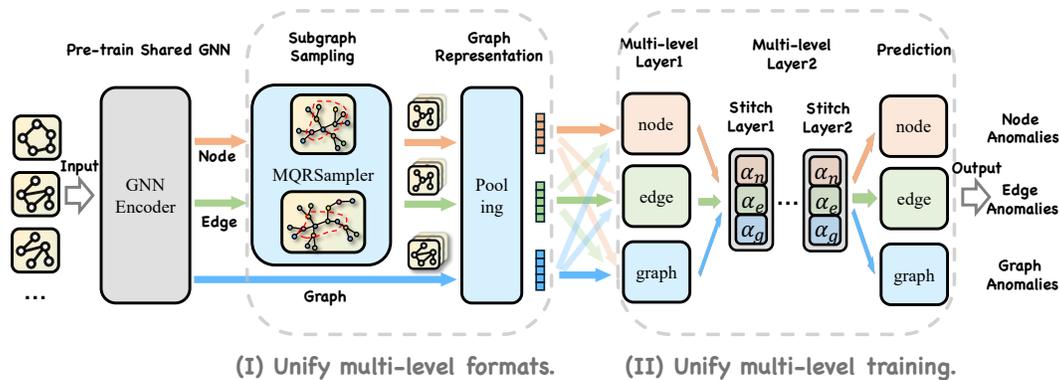[†]Corresponding Author.

Figure 1: The overall framework of UniGAD.

of objects in graph-structured data. For example, a money laundering transaction might involve both an abnormal account and the broader community it interacts with, while the specific cancer of a cell is determined by particular proteins or protein complexes within the cell. Although some unsupervised methods fuse information from nodes, edges, and subgraphs through reconstruction [27, 10, 47] or contrastive pre-training [58, 13, 36], they are still limited to single-level label supervision or prediction. There is a need for a unified approach that considers these correlations information across different levels and performs multi-level anomaly detection.

To design a unified model for addressing multi-level GAD, we identify two key challenges:

**1.** *How to unify multi-level formats?* Addressing node-level, edge-level, and graph-level tasks uniformly is challenging due to their inherent differences. Some recent works provide insights into unifying these tasks through the use of large language models (LLMs) or prompt tuning. While some methods leverage the generalization capability of LLMs [32, 54, 29] on text-attributed graphs, such semantic information is often unavailable in anomaly detection scenarios due to privacy concerns. On the other hand, graph prompt learning methods [48, 37, 61] design induced $k$-hop graphs to transform node or edge levels into graph-level tasks. Nevertheless, their sampling strategies are not specifically tailored to anomaly data, resulting in inappropriate node selections that 'erase' critical anomaly information. This oversight can severely impact the effectiveness of anomaly detection.

**2.** *How to unify multi-level training?* Training a single model for multi-level tasks involves various influencing factors, such as transferring information between different levels and achieving a balanced training of these level tasks. There is limited research on multi-task learning in the graph learning domain. Efforts like ParetoGNN [23] employ multiple self-supervised learning objectives (e.g., similarity, mutual information) as independent tasks, but these are insufficient for managing multi-level supervision. A comprehensive approach is needed to effectively integrate and balance the training of different level tasks in multi-level GAD.

In this paper, we propose UniGAD, a unified GAD model that leverages the transferability of information across node-level, edge-level, and graph-level tasks. To address the first challenge, we develop a novel subgraph sampler, **MRQSampler**, that maximizes accumulated spectral energy (i.e., the Rayleigh quotient) in the sampled subgraph with theoretical guarantee, ensuring that the sampled subgraphs contain the most critical anomaly information from nodes and edges. For the second challenge, we introduce the **GraphStitch** Network, which unifies multi-level training by integrating separate but identical networks for nodes, edges, and graphs into a unified multi-level model. This is achieved using a novel GraphStitch Unit that facilitates information sharing across different levels while maintaining the effectiveness of individual tasks. We perform comprehensive experiments on 14 GAD datasets and compare 17 state-of-the-art methods covering both node-level and graph-level GAD techniques, as well as prompt-based general multi-task graph learning methods. Results show that UniGAD achieves superior performance and offers robust zero-shot transferability across different tasks.

## 2 Related Work and Preliminaries

**Graph Anomaly Detection.** Leveraging deep learning techniques in GAD has led to significant advancements and a wide range of applications [3, 14, 1, 4, 19, 65], thoroughly reviewed in a comprehensive survey [43]. Node-level anomaly detection, the most prevalent scenario in GAD, has witnessed numerous adaptations and improvements in graph neural networks (GNNs) aimed at

enhancing performance from either a spatial [34, 38, 25] or spectral [28, 50, 15] perspective. Despite these advancements, recent benchmarks such as BOND [33] for unsupervised settings and GADBench [49] for supervised settings reveal that no single model excels across all datasets, highlighting the need for model selection tailored to specific datasets and task characteristics. For graph-level anomaly detection, various methodologies have been proposed, including transformation learning [66], knowledge distillation [42], and evolutionary mapping [44]. SIGNET [35] employs information bottleneck to generate informative subgraphs for explaining graph-level anomalies, while Rayleigh Quotient GNN [11] explores the spectral properties of anomalous graphs. Although both node-level and graph-level anomaly detection are rapidly evolving fields, to the best of our knowledge, there is no existing model that supports the joint detection of both node-level and graph-level anomalies.

**Multi-task Learning on Graphs.** Multi-task learning involves training a model to handle multiple tasks simultaneously, utilizing shared representations and relationships within the graph to enhance performance across all tasks. Recently, techniques such as graph prompt-based approaches and large language model (LLM)-based approaches have shown promise in this area. Prompt frameworks [69] like GraphPrompt [37], All-in-One [48], PRODIGY [22], MultiGPrompt [61], and SGL-PT [68] are designed to address a wide array of graph tasks. These approaches transform tasks at other levels into graph-level tasks by leveraging induced graphs. The All-in-One framework enhances connectivity by adding links between the prompt graph and the original graph, whereas GraphPrompt inserts the prompt token into graph nodes through element-wise multiplication. On the other hand, LLM-based frameworks [32, 54, 29, 6, 51] utilize the power of LLMs to learn from different levels, but they require graphs with text attributes or descriptions, which are not applicable in most anomaly detection scenarios. Additionally, some multi-task GNN efforts [23] focus on multiple self-supervised specific objectives (such as similarity and mutual information) as independent tasks, which are not suitable for unifying GAD with multi-level label supervision and prediction.

**Notation.** Let $\mathcal{G} = \{\mathcal{V}, \mathcal{E}, \boldsymbol{X}\}$ denote a connected undirected graph, where $\mathcal{V} = \{v_1, v_2, ..., v_N\}$ is the set of $N$ nodes, $\mathcal{E} = \{e_{ij}\}$ is the set of edges, and $\boldsymbol{X} \in \mathbb{R}^{n \times F}$ is node features. Let $\boldsymbol{A}$ be the corresponding adjacency matrix, $\boldsymbol{D}$ be the degree matrix with $\boldsymbol{D}_{ii} = \sum_j \boldsymbol{A}_{ij}$. Laplacian matrix $\boldsymbol{L}$ is then defined as $\boldsymbol{D} - \boldsymbol{A}$ (regular) or as $\boldsymbol{I} - \boldsymbol{D}^{-\frac{1}{2}} \boldsymbol{A} \boldsymbol{D}^{-\frac{1}{2}}$ (normalized), where $\boldsymbol{I}$ is an identity matrix. The Laplacian matrix is a symmetric matrix and can be eigen-decomposed as $\boldsymbol{L} = \boldsymbol{U} \boldsymbol{\Lambda} \boldsymbol{U}^T$, where the diagonal matrix $\boldsymbol{\Lambda}$ consists of real eigenvalues (graph spectrum). Besides, we define the subgraph as $\mathcal{G}_i$ centered on node $v_i$ and our sampled subgraph for node $v_i$ as $\mathcal{S}_i$.

**Problem Formulation.** The multi-level graph anomaly detection problem introduces a more universal challenge compared to traditional single-level approaches, described as follows:

**Definition 2.1** (Multi-level GAD). *Given a training set $\mathcal{T}r(\mathcal{N}, \mathcal{E}, \mathcal{G})$ containing nodes, edges, and graphs with arbitrary labels at any of these levels, the goal is to train a unified model to predict anomalies in a test set $\mathcal{T}e(\mathcal{N}, \mathcal{E}, \mathcal{G})$, which also contains arbitrary labels at any of these levels.*

Note that our approach does not require the presence of labels at all three levels simultaneously. It is feasible to have labels at one or more levels. Our proposed model aims to leverage the transferability of information across different levels to enhance its predictive capability.

# 3 Methodology

This section details the proposed model UniGAD for multi-level GAD, comprising a GNN encoder, MRQSampler, and GraphStitch Network, as shown in Fig. 1. Firstly, a shared pre-trained unsupervised GNN encoder is utilized to learn a more generalized node representation. To unify multi-level formats, the MRQSampler employs spectral sampling to extract subgraphs that contain the highest amount of anomalous information from nodes and edges, thus converting tasks at all three levels into graph-level tasks (Sec. 3.1). To unify multi-level training, the GraphStitch Network integrates information from different levels, adjusts the amount of sharing required at each level, and harmonizes conflicting training goals. (Sec. 3.2).

## 3.1 Spectral Subgraph Sampler for Unifying Multi-level Formats

In this subsection, we present the Maximum Rayleigh Quotient Subgraph Sampler (MRQSampler), the core module of our unified framework. By sampling subgraphs of nodes or edges, we transform node-level and edge-level tasks into graph-level tasks. Our sampler optimizes these subgraphs to maximize the Rayleigh quotient, ensuring that the sampled subgraphs retain a higher concentration of anomaly information.
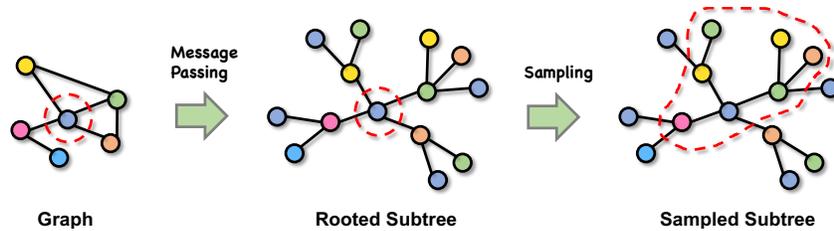
Figure 2: Message passing in GNNs and rooted subtree sampling.

### 3.1.1 Analysis of the Subgraph Sampling

**What is a suitable subgraph for GAD?** Existing methods on selecting subgraphs for target nodes or edges often use straightforward approaches like $r$-ego or $k$-hop subgraphs [48]. However, the size of the subgraph is critical for classification outcomes. If the subgraph is too large, it includes too many irrelevant nodes, while if it is too small, it may not align effectively with graph-level tasks.

To measure anomaly information in a subgraph, recent studies [50, 11] have identified a 'right-shift' phenomenon in the spectral energy distribution, moving from low to higher frequencies. This accumulated spectral energy can be quantified by the Rayleigh quotient [20]:

$$RQ(\boldsymbol{x}, \boldsymbol{L}) = \frac{\boldsymbol{x}^T \boldsymbol{L} \boldsymbol{x}}{\boldsymbol{x}^T \boldsymbol{x}} = \frac{\sum_{(i,j) \in \mathcal{E}} A_{ij}(x_j - x_i)^2}{\sum_{i \in \mathcal{V}} x_i^2}. \tag{1}$$

The following lemma [50] illustrates the relationship between the Rayleigh quotient $RQ(\boldsymbol{x}, \boldsymbol{L})$ and anomaly information:

**Lemma 1** (Tang, 2022). *Rayleigh quotient $RQ(\boldsymbol{x}, \boldsymbol{L})$, i.e. the accumulated spectral energy of the graph signal, is monotonically increasing with the anomaly degree.*

Thus, for any node $v_i$, our sampling objective is to identify the induced subgraph with the highest Rayleigh quotient containing the most anomaly information.

**Where to Sample Subgraph From?** To preserve the properties of target nodes, it is essential to sample subgraphs centered around these nodes, capturing key surrounding nodes. The most intuitive methods are $r$-ego graphs or $k$-hop graphs. However, considering the message-passing mechanisms of most GNNs [24, 56, 16], a classical work [57] provides valuable insight:

**Lemma 2** (Xu, 2018). *A GNN recursively updates each node's feature vector through its rooted subtree structures to capture the network structure and features of surrounding nodes.*

As shown in Fig. 2, the message-passing process of GNNs suggests that a rooted subtree centered on the target node is more consistent with the GNN's architecture. Therefore, we sample subgraphs from these rooted subtree structures. The remaining question is: ***How to implement subgraph sampling based on the above?*** To address this, we introduce a novel MRQSampler in the next subsection.

### 3.1.2 Maximum Rayleigh Quotient Subgraph Sampler (MRQSampler)

Building on the motivation in Section 3.1.1, our approach involves sampling subgraphs for each node starting from the rooted subtree with the node as its root. The target node is always included. We then select the subtree with the maximum Rayleigh quotient from all possible subtrees as the representative subgraph for that node to ensure it contain the maximum anomaly information. We formulate this as the following optimization problem:

$$\begin{aligned} \mathcal{S}^{\star} = \underset{\mathcal{S} \subseteq \mathcal{G}}{\arg\max} \quad & \frac{\sum_{(p,q) \in \mathcal{E}_{\mathcal{S}}} (x_p - x_q)^2}{\sum_{p \in \mathcal{S}} x_p^2}, \\ \text{s.t.} \quad & v \in \mathcal{S}, \\ & \forall v_p \in \mathcal{S}, \ (v, v_p) \text{ is accessible.} \end{aligned} \tag{2}$$

where $\mathcal{G}$ represents $k$-depth rooted subtree from $v$, and $\mathcal{S}$ is a possible subgraph from $\mathcal{G}$. The first constraint ensures the target node is included, and the second constraint ensures message passability. Generally, similar selecting subgraphs in this manner is considered an NP-Hard problem [59]. However, leveraging the properties of trees, we propose an algorithm to solve **the optimal solution**.
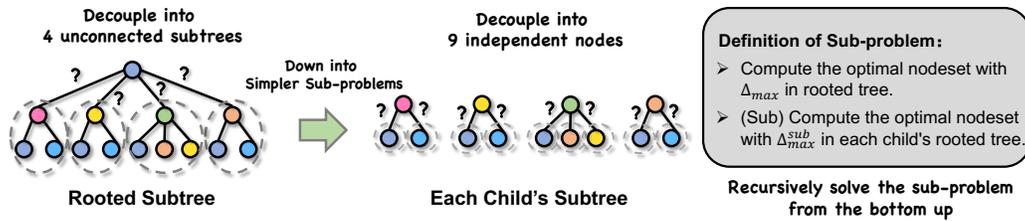
Figure 3: MRQSampler: (i) Derive the condition (Theorem 2) satisfied with the optimal subtree. (ii) Decompose the problem into simpler sub-problems by recursing through the tree depth to solve the optimal subtree with the dynamic programming (DP) algorithm.

We first determine the conditions that increase a subgraph's Rayleigh quotient when adding a node, presented in the following theorem:

**Theorem 1.** *For a graph $\mathcal{G}$, let one of its subgraphs be $\mathcal{S}$, and let its Rayleigh quotient be $RQ(\mathcal{S})$. If a new node $v_{new} \in \mathcal{G} - \mathcal{S}$ is added to $\mathcal{S}$, the Rayleigh quotient $RQ(\mathcal{S})$ will increase if and only if:*

$$\Delta(v_{new}) = \frac{\sum_{v_r \in \mathcal{S}}(x_{new} - x_r)^2}{x_{new}^2} > RQ(\mathcal{S}). \tag{3}$$

The proof of Theorem 1 can be found in Appendix A.1. We can extend this theorem from a single new node $v_{new}$ to a new node set $\mathcal{V}_{new}$, leading to the following corollary:

**Corollary 1.** *For a graph $\mathcal{G}$, let one of its subgraphs be $\mathcal{S}$, and let its Rayleigh quotient be $RQ(\mathcal{S})$. If a new nodeset $\mathcal{V}new \subset \mathcal{G} - \mathcal{S}$ is added to $\mathcal{S}$, the Rayleigh quotient $RQ(\mathcal{S})$ will increase if and only if:*

$$\Delta(\mathcal{V}_{new}) = \frac{\sum_{(i,r) \in \mathcal{E}_{\mathcal{S}+\mathcal{V}_{new}}}(x_{new_i} - x_r)^2 + \sum_{(i,j) \in \mathcal{E}_{\mathcal{V}_{new}}}(x_{new_i} - x_{new_j})^2}{\sum_{v_{new} \in \mathcal{V}_{new}} x_{new}^2} > RQ(\mathcal{S}). \tag{4}$$

The proof details are also in Appendix A.2. While the above analysis can indeed increase the Rayleigh quotient of the sampled subgraph, the sampling order may cause the results to fall into a local optimum, which may not guarantee a globally optimal solution. To identify the nodes that must be sampled in the optimal subgraph, we present the following theorem:

**Theorem 2.** *For a graph $\mathcal{G}$, let one of its subgraph be $\mathcal{S}$, the $\mathcal{S}^*$ be its final optimal subgraph, and $\mathcal{S} \subset \mathcal{S}^*$. For a new **connected** nodeset $\tilde{\mathcal{V}}_{new} \cap S = \emptyset$, it is contained in $\mathcal{S}^*$ when it satisfies:*

$$\Delta_{max}(\tilde{\mathcal{V}}_{new}) = \max_{\tilde{\mathcal{V}}_{new} \subseteq \mathcal{G}-\mathcal{S}} \Delta(\tilde{\mathcal{V}}_{new}), \text{ and } \Delta_{max}(\tilde{\mathcal{V}}_{new}) > RQ(\mathcal{S}). \tag{5}$$

We refer readers to Appendix A.3 for the rigorous proof. Through the above analysis, we derive the conditions of the nodeset contained in the optimal subtree (Theorem 2). When $\tilde{\mathcal{V}}_{new}$ satisfies Eq. (5), it always increases the Rayleigh quotient based on the current subgraph, ensuring that $\mathcal{V}_{new}$ is contained in the optimal solution. Thus, we decouple the problem of finding the subgraph with the maximum Rayleigh quotient into a process of finding the maximum $\Delta_{max}(\mathcal{V}_{new})$ each time, until adding any node/node set fails to increase the $RQ(\mathcal{S})$. Following this, we design a dynamic programming (DP) algorithm to ensure the optimal subset satisfies these conditions.

**MRQSampler Algorithm.** We introduce the Maximum Rayleigh Quotient Subgraph Sampler (MRQSampler), which uses dynamic programming (DP) to find the optimal solution. We break down the computation for the central node into sub-problems, storing the results of sub-problems to avoid redundant computations in future calculations. For a rooted subtree with the target node (edge) as the root, its children are unconnected to each other. In Fig. 3, we consider a 2-depth subtree and summarize our algorithm as follows:

- **Stage 1:** We recursively compute and store the maximum $\Delta(\tilde{\mathcal{V}}_{new})$ for each subtree, which can be down into simpler sub-problems similar to the previous one and calculates each layer in the tree recursively from the bottom up.
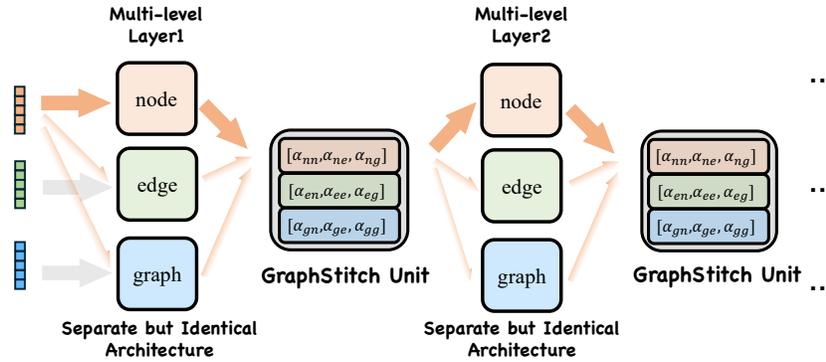
136124

Figure 4: GraphStitch network structure in UniGAD. Node level is highlighted.

- **Stage 2:** Based on Theorem 2, we iteratively select the descendant with the maximum $\Delta_{max}(\tilde{\mathcal{V}}_{new})$ (within its own rooted subtree) of the target node and the currently selected nodeset, until the conditions of Theorem 2 are no longer satisfied, i.e., when the Rayleigh quotient of the sampled subgraph no longer increases.

For efficiency, this approach can obtain the subgraph with the maximum Rayleigh quotient of the target node/edge's rooted subtree while reducing the algorithmic complexity to $O(N \log N)$. It can be further accelerated in parallel since the computation for different nodes is independent. Additionally, the sampling process only needs to be computed once in training and inference processes, minimally impacting model efficiency. For the detailed pseudocode of the algorithm, please refer to Appendix B. Note that we use mean pooling for entire graphs, but for subgraphs, we use weighted pooling to highlight central nodes/edges, with an exponential decay based on the number of hops to the central nodes/edges. This method transforms node-level and edge-level tasks into graph-level tasks, ensuring that the most anomaly information is retained in the sampled subgraphs.

### 3.2 GraphStitch Network for Unifying Multi-level Training

After obtaining graph representations, training them together through a fully connected layer can negatively impact individual levels due to the inherent differences across different-level anomalies. This can result in mediocre performance at all levels. A key challenge, therefore, is to facilitate information transfer between multi-levels without compromising single-level effectiveness. Inspired by work in the computer vision field [45], we introduce the novel GraphStitch Network to jointly consider multi-level representations.

Specifically, we train separate but identical networks for each level and use the GraphStitch unit to combine these networks into a multi-level network, managing the degree of sharing required at different levels. This approach aims to maintain single-level effectiveness while enhancing multi-level information transfer. The network structure is illustrated in Fig. 4.

To elaborate, we denote $\mathbf{e}_N$, $\mathbf{e}_E$, and $\mathbf{e}_G$ as the embeddings for nodes, edges, and graphs, respectively. The node embedding $\mathbf{e}_N = (\mathbf{e}_{nn}, \mathbf{e}_{ne}, \mathbf{e}_{ng})^\top$ consists of outputs from three separate but identically structured networks specialized for nodes, edges, and graphs. Similarly, the edge and graph embeddings are represented as $\mathbf{e}_E = (\mathbf{e}_{en}, \mathbf{e}_{ee}, \mathbf{e}_{eg})^\top$ and $\mathbf{e}_G = (\mathbf{e}_{gn}, \mathbf{e}_{ge}, \mathbf{e}_{gg})^\top$.

We define a GraphStitch operation as follows:

$$(\tilde{\mathbf{e}}_N, \tilde{\mathbf{e}}_E, \tilde{\mathbf{e}}_G) = diag \left[ \begin{pmatrix} \alpha_{nn} & \alpha_{ne} & \alpha_{ng} \\ \alpha_{en} & \alpha_{ee} & \alpha_{eg} \\ \alpha_{gn} & \alpha_{ge} & \alpha_{gg} \end{pmatrix} (\mathbf{e}_N, \mathbf{e}_E, \mathbf{e}_G) \right]. \tag{6}$$

The sharing of representations is achieved by learning a linear combination of the outputs from the three networks. This linear combination is parameterized using learnable $\alpha$. In particular, when training data lacks a certain level, the influence of that level on other levels is defined as zero during training but still retains the influence of other levels on this level. In this way, it allows the labels for training and testing to be arbitrary. Besides, if all the cross terms ($\alpha_{ne}, \alpha_{ng}, \alpha_{en}, \alpha_{eg}, \alpha_{gn}, \alpha_{ge}$) are equal to 0 means that training the three networks jointly is equivalent to training them independently. Finally, the embeddings for nodes, edges, and graphs are fed into three independent multi-layer perceptrons (MLPs) to compute the abnormal probabilities $p_i^{\mathcal{N}}$, $p_i^{\mathcal{E}}$, and $p_i^{\mathcal{G}}$, respectively.

In addition to the GraphStitch structure, UniGAD optimizes the loss functions for multi-level tasks. Specifically, the gradients of each level task's loss may conflict in direction or magnitude, potentially causing negative effects and resulting in worse performance compared to learning single-level tasks individually. Therefore, UniGAD uses a multi-level weighted cross-entropy loss for training:

$$\mathcal{L} = \sum_{\{\mathcal{N},\mathcal{E},\mathcal{G}\}} \sum_i \beta^{\{\mathcal{N},\mathcal{E},\mathcal{G}\}} \left[ \gamma y_i^{\{\mathcal{N},\mathcal{E},\mathcal{G}\}} \log \left( p_i^{\{\mathcal{N},\mathcal{E},\mathcal{G}\}} \right) + \left( 1 - y_i^{\{\mathcal{N},\mathcal{E},\mathcal{G}\}} \right) \log \left( 1 - p_i^{\{\mathcal{N},\mathcal{E},\mathcal{G}\}} \right) \right]. \quad (7)$$

where $\gamma$ is the ratio of anomaly labels ($y_i = 1$) to normal labels ($y_i = 0$), and $\beta^{\{\mathcal{N},\mathcal{E},\mathcal{G}\}}$ are adaptive weights for different tasks. We adopt a 'Gradient Surgery' approach [60] to adjust the $\beta^{\{\mathcal{N},\mathcal{E},\mathcal{G}\}}$, altering the gradients by projecting each onto the normal plane of the others. This prevents the interfering components of the gradients from affecting the network and minimizes interference among different-level GAD tasks. In this way, UniGAD ensures that each level remains relatively independent while facilitating cross-passing of relevant information between multi-level tasks.

## 4 Experiments

In this section, we conduct experiments to evaluate our UniGAD with node-level, edge-level, and graph-level tasks by answering the following questions: **Q1:** How effective is UniGAD in unifying in multi-level anomaly detection? **Q2:** Can UniGAD transfer information across different levels in zero-shot learning? **Q3:** What are the contributions of the modular design in the UniGAD model? **Q4:** How do the time and space efficiencies of UniGAD compare to those of other methods?

### 4.1 Experimental Setup

**Datasets.** We consider a total of 14 datasets, including both single-graph datasets and multi-graph datasets. 7 single-graph datasets are used to evaluate the performance of unifying node-level and edge-level tasks: Reddit, Weibo, Amazon, Yelp, Tolokers, and Questions, T-finance from the work [49], which contain node-level anomaly labels. For edge anomaly labels, we generated them according to a specific anomaly probability following the formula $P_{\text{anom}}^{(i,j)} = avg(P_{\text{anom}}^i, P_{\text{anom}}^j)$. And 7 multi-graph datasets are used to validate the performance of unifying node-level and graph-level tasks, including BM-MN, BM-MS, BM-MT, MUTAG, MNIST0, MNIST1, and T-Group. The first six datasets are from [35], containing both node anomaly labels and graph anomaly labels. Moreover, we release a real-world large-scale social group dataset T-Group, combining the data (graph anomaly labels) in [26]. For its node anomaly labels, we assume that if a node appears in 3 malicious social groups, we consider it a malicious node. Statistical data for these datasets can be found in Table 1, including the percentage of training data, the number of graphs, edges, nodes, feature dimensions, and the proportions of abnormal nodes, edges, and graphs ($\text{Nodes}_{ab}$, $\text{Edges}_{ab}$, and $\text{Graphs}_{ab}$).

Table 1: Detailed statistics of the datasets used in our experiments.

| Dataset | Train% | # Graphs | # Edges | # Nodes | # Dims | $\text{Nodes}_{ab}$ | $\text{Edges}_{ab}$ | $\text{Graphs}_{ab}$ |
|---|---|---|---|---|---|---|---|---|
| Reddit | 40% | 1 | 168,016 | 10,984 | 64 | 3.33% | 2.72% | / |
| Weibo | 40% | 1 | 416,368 | 8,405 | 400 | 10.33% | 5.71% | / |
| Amazon | 70% | 1 | 8,847,096 | 11,944 | 25 | 6.87% | 2.49% | / |
| Yelp | 70% | 1 | 7,739,912 | 45,954 | 32 | 14.53% | 13.89% | / |
| Tolokers | 50% | 1 | 530,758 | 11,758 | 10 | 21.82% | 33.44% | / |
| Questions | 50% | 1 | 202,461 | 48,921 | 301 | 2.98% | 7.50% | / |
| T-Finance | 40% | 1 | 21,222,543 | 39,357 | 10 | 4.58% | 2.77% | / |
| BM-MN | 40% | 700 | 40,032 | 12,911 | 1 | 48.91% | / | 14.29% |
| BM-MS | 40% | 700 | 30,238 | 9,829 | 1 | 31.99% | / | 14.29% |
| BM-MT | 40% | 700 | 32,042 | 10,147 | 1 | 34.49% | / | 14.29% |
| MUTAG | 40% | 2,951 | 179,732 | 88,926 | 14 | 4.81% | / | 34.40% |
| MNIST0 | 10% | 70,000 | 41,334,380 | 4,939,668 | 5 | 35.46% | / | 9.86% |
| MNIST1 | 10% | 70,000 | 41,334,380 | 4,939,668 | 5 | 35.46% | / | 11.25% |
| T-Group | 40% | 37,402 | 93,367,082 | 11,015,616 | 10 | 0.64% | / | 4.26% |

**Baselines.** To comprehensively compare with traditional single-level tasks, we consider nine representative node-level methods: GCN [24], GIN [57], GraphSAGE [16], SGC [56], GAT [52], BernNet [18], PNA [7], AMNet [5], and BWGNN [50]. Given the limited work on edge anomalies, we adapt a concatenated strategy [64] from link prediction, resulting in nine corresponding edge-level methods: GCNE, GINE, GSAGEE, SGCE, GATE, BernE, PNAE, AME, and BWE. For graph-level

Table 2: Comparison of unified performance (AUROC) at both node and edge levels with different single-level methods, multi-task methods, and our proposed method.

| | Dataset | Reddit | | Weibo | | Amazon | | Yelp | | Tolokers | | Questions | | T-Finance | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | Task-level | Node | Edge | Node | Edge | Node | Edge | Node | Edge | Node | Edge | Node | Edge | Node | Edge |
| Node-Level | GCN | 62.60 | / | 97.97 | / | 82.37 | / | 57.62 | / | 75.21 | / | 70.15 | / | 90.70 | / |
| | GIN | 65.59 | / | 95.64 | / | 92.17 | / | 74.46 | / | 75.15 | / | 69.13 | / | 86.43 | / |
| | GraphSAGE | 62.25 | / | 94.45 | / | 84.53 | / | 82.12 | / | 79.74 | / | 72.47 | / | 78.16 | / |
| | SGC | 52.12 | / | 97.71 | / | 80.24 | / | 53.03 | / | 69.51 | / | 70.59 | / | 74.21 | / |
| | GAT | 65.87 | / | 94.40 | / | 96.24 | / | 77.40 | / | 78.90 | / | 71.38 | / | 90.60 | / |
| | BernNet | 66.68 | / | 93.93 | / | 96.62 | / | 81.48 | / | 76.68 | / | 70.28 | / | 92.37 | / |
| | PNA | 65.28 | / | 97.43 | / | 81.41 | / | 71.81 | / | 75.82 | / | 71.78 | / | 68.17 | / |
| | AMNet | 68.31 | / | 94.17 | / | 97.31 | / | 81.42 | / | 76.67 | / | 68.63 | / | 93.58 | / |
| | BWGNN | 64.65 | / | 97.42 | / | 97.80 | / | 83.11 | / | 80.51 | / | 70.25 | / | 96.03 | / |
| Edge-level | GCNE | / | 63.10 | / | 99.03 | / | 78.63 | / | 57.80 | / | 73.59 | / | 79.05 | / | 87.63 |
| | GINE | / | 67.36 | / | 98.09 | / | 79.74 | / | 67.58 | / | 69.27 | / | 80.75 | / | 79.05 |
| | GSAGEE | / | **67.52** | / | 98.67 | / | 78.92 | / | 73.30 | / | **76.98** | / | **87.51** | / | 77.14 |
| | SGCE | / | 53.36 | / | 98.55 | / | 76.41 | / | 52.02 | / | 70.59 | / | 74.24 | / | 69.01 |
| | GATE | / | 67.07 | / | 97.92 | / | 90.20 | / | 72.96 | / | 71.92 | / | 81.64 | / | 83.09 |
| | BernE | / | 65.57 | / | 97.87 | / | 89.60 | / | 73.93 | / | 73.39 | / | 84.78 | / | 87.80 |
| | PNAE | / | 64.15 | / | 99.10 | / | 75.71 | / | 67.98 | / | 75.09 | / | 84.05 | / | 83.91 |
| | AME | / | 66.73 | / | 97.08 | / | 89.36 | / | 73.69 | / | 71.99 | / | 84.93 | / | 86.19 |
| | BWE | / | 67.39 | / | 98.93 | / | 91.61 | / | 75.63 | / | 75.66 | / | 85.00 | / | 92.27 |
| Multi-task | GraphPrompt-U | 50.03 | 49.78 | 55.29 | 50.71 | 50.01 | 50.96 | 49.83 | 49.56 | 51.24 | 49.66 | 55.16 | 50.01 | OOT | OOT |
| | All-in-One-U | 51.35 | 54.10 | 48.61 | 52.63 | 56.11 | 54.80 | 49.77 | 49.13 | 50.41 | 49.29 | 51.49 | 64.24 | OOT | OOT |
| UniGAD (Ours) | UniGAD - GCN | **71.65** | 65.46 | 99.02 | **99.13** | 82.92 | 80.04 | 63.22 | 61.74 | 77.26 | 72.89 | **73.92** | 74.72 | 95.68 | 93.75 |
| | UniGAD - BWG | 64.42 | 53.60 | **99.07** | 99.10 | **97.84** | **92.18** | **86.23** | **79.05** | **80.62** | 74.85 | 70.97 | 73.45 | **96.49** | **94.32** |

anomaly detection, we consider six state-of-the-art methods: OCGIN [66], OCGTL [46], GLocalKD [42], iGAD [63], GmapAD [44], and RQGNN [11]. Additionally, to compare multi-task models, we include two recent multi-task graph prompt methods: GraphPrompt [37] and All-in-One [48]. While these methods were not originally proposed for joint multi-task training, we adapt their ideas and develop multi-task versions for our comparison, GraphPrompt-U and All-in-One-U, whose modifications were limited to the data preprocessing component to accommodate the simultaneous handling of multiple object types (node/edge or node/graph) within induced graphs.

**Implementations.** We evaluate three metrics: AUROC (Sec. 4), Macro F1-score and AUPRC (Appendix E). For each result, we conduct 5 runs and report the mean results. In UniGAD, we choose two backbone GNN encoders: GCN [24] and BWGNN [50]. We use a shared graph pre-training method, GraphMAE [21], to obtain a more generalized node representation. For multi-dimensional feature vectors, we normalize all feature dimensions and then take the norm (1-norm in our case) to obtain a composite feature for each node, allowing us to identify the most anomalous nodes in MRQSampler based on this comprehensive feature. To avoid data leakage, for single-graph datasets, edges between the training set and the testing set are not considered; for multi-graph datasets, the training set and the testing set consist of different graphs and their nodes. More details on the implementation can be found in the Appendix C.

## 4.2 Multi-Level Performance Comparison (RQ1)

To compare the performance of multi-level anomaly detection, we conduct experiments under two settings. For the single-graph datasets, we compare the performance of unified training on node-level and edge-level data. For the multi-graph datasets, we compare the performance of unified training on node-level and graph-level data.

**Node-level and edge-level jointly.** We first evaluate the performance of unified training on node-level and edge-level data. We compare UniGAD against three groups of GNN models mentioned above: node-level models, edge-level models, and multi-task graph learning methods. Table 2 reports the AUROC of each model on six datasets, with the best result on each dataset highlighted in boldface. Overall, we find that UniGAD achieves state-of-the-art performance in nearly all scenarios. UniGAD outperforms single-level specialized models, indicating that unified training with UniGAD leverages information from other levels to enhance the performance of individual tasks. Multi-task approaches (GraphPrompt-U and All-in-One-U) tend to negatively impact multi-task performance, potentially because they are unable to effectively handle different types of anomaly label supervision. Meanwhile, UniGAD is designed for a multi-task setting, the performance on a single level might be slightly compromised to ensure the model performs well across all tasks in some datasets.

**Node-level and graph-level jointly.** We then evaluate the unified training of node-level and graph-level tasks under similar settings. Table 3 shows the results, and UniGAD achieves state-of-the-art performance in nearly all scenarios. Our observations are as follows. First, there is a multi-level synergy in UniGAD, where strong performance in one task benefits the performance of other tasks.

Table 3: Comparison of unified performance (AUROC) at both node and graph levels with different single-level methods, multi-task methods, and our proposed method.

| Dataset Task-level | BM-MN Node | BM-MN Graph | BM-MS Node | BM-MS Graph | BM-MT Node | BM-MT Graph | MUTAG Node | MUTAG Graph | MNIST0 Node | MNIST0 Graph | MNIST1 Node | MNIST1 Graph | T-Group Node | T-Group Graph |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **Node-level** | | | | | | | | | | | | | | |
| GCN | 86.31 | / | 90.17 | / | 92.30 | / | 99.38 | / | 94.10 | / | 93.84 | / | 91.81 | / |
| GIN | 56.73 | / | 50.41 | / | 54.90 | / | 99.39 | / | 93.55 | / | 93.49 | / | 61.51 | / |
| GraphSAGE | 50.00 | / | 50.00 | / | 49.95 | / | 99.26 | / | 99.99 | / | 99.99 | / | 64.15 | / |
| SGC | 50.27 | / | 50.87 | / | 49.44 | / | 89.19 | / | 86.97 | / | 86.97 | / | 82.55 | / |
| GAT | 58.47 | / | 62.52 | / | 65.72 | / | 99.42 | / | 99.90 | / | 99.99 | / | 78.17 | / |
| BernNet | 60.06 | / | 65.58 | / | 59.18 | / | 98.97 | / | 99.99 | / | 99.99 | / | 93.85 | / |
| PNA | 72.96 | / | 55.19 | / | 75.61 | / | 98.76 | / | 99.80 | / | 99.87 | / | 55.66 | / |
| BWGNN | 93.05 | / | 87.22 | / | 88.97 | / | 99.50 | / | 99.99 | / | 99.99 | / | 94.81 | / |
| **Graph-level** | | | | | | | | | | | | | | |
| OCGIN | / | 98.46 | / | 81.97 | / | 58.05 | / | 89.50 | / | 57.24 | / | 86.15 | / | 64.53 |
| OCGTL | / | 98.48 | / | 83.17 | / | 59.99 | / | 92.19 | / | 59.35 | / | 93.45 | / | 46.77 |
| GLocalKD | / | 92.36 | / | 77.25 | / | 53.23 | / | 72.77 | / | 66.69 | / | 57.42 | / | 78.53 |
| iGAD | / | 91.68 | / | 96.68 | / | 99.14 | / | 96.28 | / | 98.93 | / | 99.50 | / | 64.44 |
| GmapAD | / | 50.00 | / | 50.00 | / | 50.00 | / | 50.00 | / | OOM | / | OOM | / | OOM |
| RQGNN | / | 98.79 | / | 97.98 | / | 99.83 | / | 96.41 | / | 96.62 | / | 95.57 | / | 73.90 |
| **Multi-task** | | | | | | | | | | | | | | |
| GraphPrompt-U | 51.59 | 46.85 | 50.54 | 48.67 | 51.42 | 49.38 | 97.08 | 68.23 | 81.16 | 83.88 | 81.37 | 6.16 | 47.40 | 50.81 |
| All-in-One-U | 67.87 | 3.21 | 54.70 | 19.42 | 69.70 | 45.89 | 50.63 | 48.98 | OOT | OOT | OOT | OOT | OOT | OOT |
| **UniGAD (Ours)** | | | | | | | | | | | | | | |
| UniGAD - GCN | 99.75 | 94.29 | 99.60 | 99.67 | 99.63 | 99.99 | 99.50 | 96.33 | 97.93 | 98.99 | 98.11 | 99.59 | 95.57 | 88.73 |
| UniGAD - BWG | 92.60 | 68.74 | 93.30 | 68.55 | 90.76 | 56.01 | 99.54 | 96.73 | 99.99 | 99.61 | 99.99 | 99.98 | 96.19 | 88.78 |

Table 4: Zero-shot transferability (AUROC) at node and edge levels.

| Methods | Reddit N→E | Reddit E→N | Weibo N→E | Weibo E→N | Amazon N→E | Amazon E→N | Yelp N→E | Yelp E→N | Tolokers N→E | Tolokers E→N | Questions N→E | Questions E→N | T-Finance N→E | T-Finance E→N |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| GraphPrompt-U | 54.06 | 47.43 | 57.03 | 42.85 | 49.76 | 50.26 | 49.97 | 49.94 | 48.56 | 51.08 | 54.26 | 51.97 | OOT | OOT |
| All-in-One-U | 49.23 | 49.93 | 52.22 | 54.30 | 52.61 | 42.35 | 49.48 | 44.50 | 48.34 | 50.22 | 49.83 | 51.97 | OOT | OOT |
| UniGAD - GCN | 59.67 | 59.46 | 98.31 | 98.59 | 76.20 | 82.38 | 58.28 | 60.92 | 71.45 | 73.35 | 69.54 | 65.37 | 91.63 | 90.17 |
| UniGAD - BWG | 53.32 | 57.63 | 94.71 | 96.87 | 82.64 | 96.41 | 75.56 | 84.08 | 74.04 | 78.49 | 71.02 | 62.72 | 93.60 | 95.68 |

For example, in MNIST-0 and MNIST-1, compared to other graph-level GAD methods, UniGAD significantly boosts graph-level performance by leveraging strong node-level results. Second, Uni-GAD performs better on large graphs, likely because graph structure plays a more significant role in smaller datasets. However, the backbones of UniGAD (GCN, BWGNN) are primarily node-level models, which may not effectively encode graph-level structural information. This limitation's impact diminishes in large-scale graph datasets. Besides, methods like All-in-One-U often run out of time (OOT) with large datasets because they redundantly learn the same node representations across different subgraphs, making processing impractically slow, especially for large graph-level datasets like T-Group. UniGAD addresses this issue by using a shared GNN encoder across tasks, avoiding redundant learning and enhancing efficiency.

## 4.3 The Transferability in Zero-Shot Learning (RQ2)

To assess the transfer capability of UniGAD, we explore zero-shot learning scenarios where labels for a given level have never been exposed during training, as shown in Tables 4 and 5. In these experiments, UniGAD is trained solely with labels from alternative levels. The notation $N \rightarrow E$ indicates using node labels to infer edge labels, with analogous notations for other label transfers. Our findings indicate that in zero-shot scenarios, UniGAD outperforms existing multi-task prompt learning methods. Moreover, the classification performance of UniGAD under zero-shot transfer learning even surpasses some of the leading baselines in supervised settings on Yelp and BM-MS. It highlights the superior transfer capability of UniGAD across various GAD tasks.

## 4.4 Ablation Study (RQ3)

To investigate the contribution of each module in UniGAD, we present the ablation study results in Table 6. For the sampler module, we compare the results without subgraph sampling (w/o GS.), using a simple sampler with all 2-hop neighbors (w 2hop.), and using random sampling (w RS.). For the Graph-Stitch module, we replace it with a unified MLP (w/o ST.). The results indicate that both the subgraph sampler (SG.) and the GraphStitch (ST.) modules enhance the overall performance of UniGAD. Additionally,

Table 6: Performance of UniGAD and its variants.

| Metrics Task-level | BM-MS AUROC node | BM-MS AUROC graph | BM-MS Macro F1 node | BM-MS Macro F1 graph | Reddit AUROC node | Reddit AUROC edge | Reddit Macro F1 node | Reddit Macro F1 edge |
|---|---|---|---|---|---|---|---|---|
| w/o GS. | 97.13 | 98.99 | 80.35 | 95.79 | 68.69 | 66.06 | 53.83 | 52.78 |
| w 2hop. | 97.49 | 99.94 | 67.29 | 84.20 | 67.53 | 63.62 | 51.77 | 50.69 |
| w RS. | 93.85 | 84.92 | 85.88 | 72.21 | 65.32 | 61.85 | 52.32 | 51.03 |
| w/o ST. | 99.94 | 95.51 | 99.47 | 84.91 | 67.74 | 65.92 | 54.35 | 52.52 |
| UniGAD | 99.60 | 99.67 | 99.57 | 95.86 | 71.65 | 65.46 | 56.70 | 53.80 |

Table 5: Zero-shot transferability (AUROC) at node and graph levels.

| Methods | BM-MN | | BM-MS | | BM-MT | | MUTAG | | MNIST0 | | T-Group | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | N→G | G→N | N→G | G→N | N→G | G→N | N→G | G→N | N→G | G→N | N→G | G→N |
| GraphPrompt-U | 50.60 | 51.57 | 51.97 | 46.95 | 46.62 | 48.06 | 59.62 | 64.26 | 83.98 | **88.06** | 58.28 | 58.35 |
| All-in-One-U | **94.39** | 65.69 | 52.63 | 40.88 | 44.86 | 34.27 | 61.63 | 36.13 | OOT | OOT | OOT | OOT |
| UniGAD - GCN | 72.82 | **87.63** | **81.49** | **90.83** | **62.85** | **79.26** | **72.79** | **88.53** | **85.24** | 70.57 | **86.86** | **75.89** |
| UniGAD - BWG | 64.61 | 57.56 | 65.33 | 51.34 | 55.78 | 53.41 | 66.92 | 87.03 | 74.23 | 63.70 | 86.81 | 64.81 |



(a) Comparing the execution time.
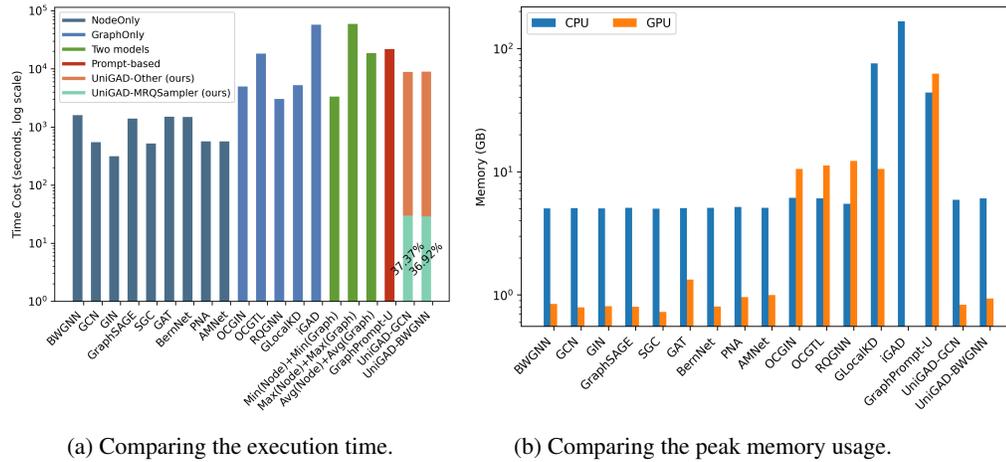


(b) Comparing the peak memory usage.

Figure 5: The evaluation of time and space efficiency metrics. We highlight the percentage of total execution time spent by MRQSampler.

inappropriate subgraph sampling may perform worse than no subgraph sampling, likely due to the loss of anomalous information during the sampling process.

## 4.5 Efficiency Analysis (RQ4)

we conduct a comprehensive evaluation of both time and space efficiency on the large-scale, real-world T-Group dataset. To provide a more straightforward comparison between single-task and multi-task baselines, we calculate the average, minimum, and maximum for combinations of single-task node-level and graph-level models, and compare these with multi-task models. The results, as shown in Fig. 5 (a), indicate that in terms of execution time, our method is slower than the combination of the fastest single-level models but faster than the average of the combination. Regarding peak memory usage, Fig. 5 (b) demonstrates that graph-level models consume significantly more memory than node-level models. Our method maintains memory consumption comparable to node-level models and substantially lower than both graph-level GAD models and prompt-based methods.

## 5 Conclusion

This paper presents UniGAD, a unified graph anomaly detection framework that jointly addresses anomalies at the node, edge, and graph levels. The model integrates two novel components: the MRQSampler and the GraphStitch network. MRQSampler maximizes spectral energy to ensure subgraphs capture critical anomaly information, addressing the challenge of unifying different graph object formats. The GraphStitch Network unifies multi-level training by using identical networks for nodes, edges, and graphs, facilitated by the GraphStitch Unit for effective information sharing. Our thorough evaluations across 14 GAD datasets, including two real-world large-scale datasets (T-Finance and T-Group), and comparisons with 17 graph learning methods show that UniGAD not only surpasses existing models in various tasks but also exhibits strong zero-shot transferability capabilities. A limitation of our paper is that the GNN encoder primarily focuses on node-level embeddings, which may result in lost information about the graph structure. We leave the exploration of multi-level tasks pre-training in the future works.

## Acknowledgments and Disclosure of Funding

## References

[1] Esma Aïmeur, Sabrine Amri, and Gilles Brassard. Fake news, disinformation and misinformation in social media: a review. *Social Network Analysis and Mining*, 13(1):30, 2023.

[2] Leman Akoglu, Hanghang Tong, and Danai Koutra. Graph based anomaly detection and description: a survey. *Data mining and knowledge discovery*, 29(3):626–688, 2015.

[3] Ketan Anand, Jay Kumar, and Kunal Anand. Anomaly detection in online social network: A survey. In *2017 International Conference on Inventive Communication and Computational Technologies (ICICCT)*, pages 456–459. IEEE, 2017.

[4] Alessandro Bondielli and Francesco Marcelloni. A survey on fake news and rumour detection techniques. *Information Sciences*, 497:38–55, 2019.

[5] Ziwei Chai, Siqi You, Yang Yang, Shiliang Pu, Jiarong Xu, Haoyang Cai, and Weihao Jiang. Can abnormality be detected by graph neural networks? In *IJCAI*, 2022.

[6] Nuo Chen, Yuhan Li, Jianheng Tang, and Jia Li. Graphwiz: An instruction-following language model for graph computational problems. In *Proceedings of the 30th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pages 353–364, 2024.

[7] Gabriele Corso, Luca Cavalleri, Dominique Beaini, Pietro Liò, and Petar Veličković. Principal neighbourhood aggregation for graph nets. *Advances in Neural Information Processing Systems*, 33:13260–13271, 2020.

[8] Kelton AP Da Costa, João P Papa, Celso O Lisboa, Roberto Munoz, and Victor Hugo C de Albuquerque. Internet of things: A survey on machine learning-based intrusion detection approaches. *Computer Networks*, 151:147–157, 2019.

[9] Ailin Deng and Bryan Hooi. Graph neural network-based anomaly detection in multivariate time series. In *Proceedings of the AAAI conference on artificial intelligence*, pages 4027–4035, 2021.

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

[11] Xiangyu Dong, Xingyi Zhang, and Sibo Wang. Rayleigh quotient graph neural networks for graph-level anomaly detection. *arXiv preprint arXiv:2310.02861*, 2023.

[12] Yingtong Dou, Zhiwei Liu, Li Sun, Yutong Deng, Hao Peng, and Philip S Yu. Enhancing graph neural network-based fraud detectors against camouflaged fraudsters. In *CIKM*, pages 315–324, 2020.

[13] Jingcan Duan, Siwei Wang, Pei Zhang, En Zhu, Jingtao Hu, Hu Jin, Yue Liu, and Zhibin Dong. Graph anomaly detection via multi-scale contrastive learning networks with augmented view. *arXiv preprint arXiv:2212.00535*, 2022.

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

[15] Yuan Gao, Xiang Wang, Xiangnan He, Zhenguang Liu, Huamin Feng, and Yongdong Zhang. Addressing heterophily in graph anomaly detection: A perspective of graph spectrum. In *Proceedings of the ACM Web Conference*, 2023.

[16] William L Hamilton, Rex Ying, and Jure Leskovec. Inductive representation learning on large graphs. In *NeurIPS*, 2017.

https://doi.org/10.52202/079017-4324

[17] Jiawei Han, Micheline Kamber, and Jian Pei. *Data Mining: Concepts and Techniques, 3rd edition*. Morgan Kaufmann, 2011.

[18] Mingguo He, Zhewei Wei, Hongteng Xu, et al. Bernnet: Learning arbitrary graph spectral filters via bernstein approximation. *Advances in Neural Information Processing Systems*, 34:14239–14251, 2021.

[19] Waleed Hilal, S Andrew Gadsden, and John Yawney. Financial fraud: a review of anomaly detection techniques and recent advances. *Expert systems With applications*, 193:116429, 2022.

[20] Roger A Horn and Charles R Johnson. *Matrix analysis*. Cambridge university press, 2012.

[21] Zhenyu Hou, Xiao Liu, Yukuo Cen, Yuxiao Dong, Hongxia Yang, Chunjie Wang, and Jie Tang. Graphmae: Self-supervised masked graph autoencoders. In *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pages 594–604, 2022.

[22] Qian Huang, Hongyu Ren, Peng Chen, Gregor Kržmanc, Daniel Zeng, Percy S Liang, and Jure Leskovec. Prodigy: Enabling in-context learning over graphs. *Advances in Neural Information Processing Systems*, 36, 2024.

[23] Mingxuan Ju, Tong Zhao, Qianlong Wen, Wenhao Yu, Neil Shah, Yanfang Ye, and Chuxu Zhang. Multi-task self-supervised graph neural networks enable stronger task generalization. *arXiv preprint arXiv:2210.02016*, 2022.

[24] Thomas N. Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. In *ICLR*, 2017.

[25] Ao Li, Zhou Qin, Runshi Liu, Yiqun Yang, and Dong Li. Spam review detection with graph convolutional networks. In *Proceedings of the 28th ACM International Conference on Information and Knowledge Management*, pages 2703–2711, 2019.

[26] Jia Li, Yu Rong, Hong Cheng, Helen Meng, Wenbing Huang, and Junzhou Huang. Semi-supervised graph classification: A hierarchical graph perspective. In *The World Wide Web Conference*, pages 972–982, 2019.

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

[28] Yuening Li, Xiao Huang, Jundong Li, Mengnan Du, and Na Zou. Specae: Spectral autoencoder for anomaly detection in attributed networks. In *CIKM*, pages 2233–2236, 2019.

[29] Yuhan Li, Zhixun Li, Peisong Wang, Jia Li, Xiangguo Sun, Hong Cheng, and Jeffrey Xu Yu. A survey of graph meets large language model: Progress and future directions. *arXiv preprint arXiv:2311.12399*, 2023.

[30] Yiqing Lin and H Vicky Zhao. Maximum entropy attack on decision fusion with herding behaviors. *IEEE Signal Processing Letters*, 2024.

[31] Fanzhen Liu, Xiaoxiao Ma, Jia Wu, Jian Yang, Shan Xue, Amin Beheshti, Chuan Zhou, Hao Peng, Quan Z Sheng, and Charu C Aggarwal. Dagad: Data augmentation for graph anomaly detection. *arXiv preprint arXiv:2210.09766*, 2022.

[32] Hao Liu, Jiarui Feng, Lecheng Kong, Ningyue Liang, Dacheng Tao, Yixin Chen, and Muhan Zhang. One for all: Towards training one graph model for all classification tasks. *arXiv preprint arXiv:2310.00149*, 2023.

[33] Kay Liu, Yingtong Dou, Yue Zhao, Xueying Ding, Xiyang Hu, Ruitong Zhang, Kaize Ding, Canyu Chen, Hao Peng, Kai Shu, Lichao Sun, Jundong Li, George H Chen, Zhihao Jia, and Philip S Yu. Bond: Benchmarking unsupervised outlier node detection on static attributed graphs. In *Advances in Neural Information Processing Systems*, volume 35, 2022.

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

[35] Yixin Liu, Kaize Ding, Qinghua Lu, Fuyi Li, Leo Yu Zhang, and Shirui Pan. Towards self-interpretable graph-level anomaly detection. *Advances in Neural Information Processing Systems*, 36, 2024.

[36] Yixin Liu, Zhao Li, Shirui Pan, Chen Gong, Chuan Zhou, and George Karypis. Anomaly detection on attributed networks via contrastive self-supervised learning. *IEEE transactions on neural networks and learning systems*, 2021.

[37] Zemin Liu, Xingtong Yu, Yuan Fang, and Xinming Zhang. Graphprompt: Unifying Pre-Training and Downstream Tasks for Graph Neural Networks. In *The Web Conference*, pages 417–428, 2023.

[38] Zhiwei Liu, Yingtong Dou, Philip S Yu, Yutong Deng, and Hao Peng. Alleviating the inconsistency problem of applying graph neural network to fraud detection. In *SIGIR*, pages 1569–1572, 2020.

[39] Zhiyuan Liu, Chunjie Cao, and Jingzhang Sun. Mul-gad: a semi-supervised graph anomaly detection framework via aggregating multi-view information. *arXiv preprint arXiv:2212.05478*, 2022.

[40] Zhiyuan Liu, Chunjie Cao, Fangjian Tao, and Jingzhang Sun. Revisiting graph contrastive learning for anomaly detection. *arXiv preprint arXiv:2305.02496*, 2023.

[41] Ziqi Liu, Chaochao Chen, Longfei Li, Jun Zhou, Xiaolong Li, Le Song, and Yuan Qi. Geniepath: Graph neural networks with adaptive receptive paths. In *Proceedings of the AAAI Conference on Artificial Intelligence*, pages 4424–4431, 2019.

[42] Rongrong Ma, Guansong Pang, Ling Chen, and Anton van den Hengel. Deep graph-level anomaly detection by glocal knowledge distillation. In *Proceedings of the fifteenth ACM international conference on web search and data mining*, pages 704–714, 2022.

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

[44] Xiaoxiao Ma, Jia Wu, Jian Yang, and Quan Z Sheng. Towards graph-level anomaly detection via deep evolutionary mapping. In *Proceedings of the 29th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pages 1631–1642, 2023.

[45] Ishan Misra, Abhinav Shrivastava, Abhinav Gupta, and Martial Hebert. Cross-stitch networks for multi-task learning. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3994–4003, 2016.

[46] Chen Qiu, Marius Kloft, Stephan Mandt, and Maja Rudolph. Raising the bar in graph-level anomaly detection. *arXiv preprint arXiv:2205.13845*, 2022.

[47] Amit Roy, Juan Shu, Jia Li, Carl Yang, Olivier Elshocht, Jeroen Smeets, and Pan Li. Gad-nr: Graph anomaly detection via neighborhood reconstruction. In *Proceedings of the 17th ACM International Conference on Web Search and Data Mining*, pages 576–585, 2024.

[48] Xiangguo Sun, Hong Cheng, Jia Li, Bo Liu, and Jihong Guan. All in one: Multi-task prompting for graph neural networks. In *Proceedings of the 29th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pages 2120–2131, 2023.

[49] Jianheng Tang, Fengrui Hua, Ziqi Gao, Peilin Zhao, and Jia Li. Gadbench: Revisiting and benchmarking supervised graph anomaly detection. *Advances in Neural Information Processing Systems*, 36, 2024.

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

[51] Jianheng Tang, Qifan Zhang, Yuhan Li, and Jia Li. Grapharena: Benchmarking large language models on graph computational problems. *arXiv preprint arXiv:2407.00379*, 2024.

[52] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Lio, and Yoshua Bengio. Graph attention networks. In *ICLR*, 2017.

[53] Daixin Wang, Jianbin Lin, Peng Cui, Quanhui Jia, Zhen Wang, Yanming Fang, Quan Yu, Jun Zhou, Shuang Yang, and Yuan Qi. A semi-supervised graph attentive network for financial fraud detection. In *ICDM*, pages 598–607. IEEE, 2019.

[54] Jianing Wang, Junda Wu, Yupeng Hou, Yao Liu, Ming Gao, and Julian McAuley. Instructgraph: Boosting large language models via graph-centric instruction tuning and preference alignment. *arXiv preprint arXiv:2402.08785*, 2024.

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

[56] Felix Wu, Amauri Souza, Tianyi Zhang, Christopher Fifty, Tao Yu, and Kilian Weinberger. Simplifying graph convolutional networks. In *ICML*, pages 6861–6871, 2019.

[57] Keyulu Xu, Weihua Hu, Jure Leskovec, and Stefanie Jegelka. How powerful are graph neural networks? *ICLR*, 2019.

[58] Zhiming Xu, Xiao Huang, Yue Zhao, Yushun Dong, and Jundong Li. Contrastive attributed network anomaly detection with data augmentation. In *Proceedings of the PAKDD*, 2022.

[59] Kuo Yang, Zhengyang Zhou, Xu Wang, Pengkun Wang, Limin Li, and Yang Wang. Raye-sub: Countering subgraph degradation via perfect reconstruction.

[60] Tianhe Yu, Saurabh Kumar, Abhishek Gupta, Sergey Levine, Karol Hausman, and Chelsea Finn. Gradient surgery for multi-task learning. *Advances in Neural Information Processing Systems*, 33:5824–5836, 2020.

[61] Xingtong Yu, Chang Zhou, Yuan Fang, and Xinming Zhang. Multigprompt for multi-task pre-training and prompting on graphs. *arXiv preprint arXiv:2312.03731*, 2023.

[62] Ge Zhang, Jia Wu, Jian Yang, Amin Beheshti, Shan Xue, Chuan Zhou, and Quan Z Sheng. Fraudre: Fraud detection dual-resistant to graph inconsistency and imbalance. In *2021 IEEE International Conference on Data Mining (ICDM)*, pages 867–876. IEEE, 2021.

[63] Ge Zhang, Zhenyu Yang, Jia Wu, Jian Yang, Shan Xue, Hao Peng, Jianlin Su, Chuan Zhou, Quan Z Sheng, Leman Akoglu, et al. Dual-discriminative graph neural network for imbalanced graph-level anomaly detection. *Advances in Neural Information Processing Systems*, 35:24144–24157, 2022.

[64] Muhan Zhang. Graph neural networks: link prediction. *Graph Neural Networks: Foundations, Frontiers, and Applications*, pages 195–223, 2022.

[65] Haihong Zhao, Bo Yang, Jiaxu Cui, Qianli Xing, Jiaxing Shen, Fujin Zhu, and Jiannong Cao. Effective fault scenario identification for communication networks via knowledge-enhanced graph neural networks. *IEEE Transactions on Mobile Computing*, 23(4):3243–3258, 2023.

[66] Lingxiao Zhao and Leman Akoglu. On using classification datasets to evaluate graph outlier detection: Peculiar observations and new insights. *Big Data*, 11(3):151–180, 2023.

[67] Li Zheng, Zhenpeng Li, Jian Li, Zhao Li, and Jun Gao. Addgraph: Anomaly detection in dynamic graph using attention-based temporal gcn. In *IJCAI*, pages 4419–4425, 2019.

[68] Yun Zhu, Jianhao Guo, and Siliang Tang. Sgl-pt: A strong graph learner with graph prompt tuning. *arXiv preprint arXiv:2302.12449*, 2023.

[69] Chenyi Zi, Haihong Zhao, Xiangguo Sun, Yiqing Lin, Hong Cheng, and Jia Li. Prog: A graph prompt learning benchmark. *arXiv preprint arXiv:2406.05346*, 2024.

# A Proofs

## A.1 The proof of Theorem 1

*Proof.* For a new node $v_{new}$, let $\mathcal{S}'$ be the subgraph after the addition of $v_{new}$. Based on the fact that the definition of Rayleigh quotient $RQ(\mathcal{S}) = \frac{\sum_{(p,q)\in\mathcal{E}_\mathcal{S}}(x_p - x_q)^2}{\sum_{p\in\mathcal{S}} x_p^2}$, it need to satisfy the following condition in order to increase the Rayleigh quotient $RQ(\mathcal{S}') > RQ(\mathcal{S})$:

$$\frac{\sum_{(p,q)\in\mathcal{E}}(x_p - x_q)^2 + \sum_{v_r\in\mathcal{S}}(x_{new} - x_r)^2}{\sum_{v_r\in\mathcal{S}} x_r^2 + x_{new}^2} > \frac{\sum_{(p,q)\in\mathcal{E}}(x_p - x_q)^2}{\sum_{v_r\in\mathcal{S}} x_r^2}, \tag{A.1}$$

where $\sum_{v_r\in\mathcal{S}}(x_{new} - x_r)^2$ represents the sum of the feature difference between the new node $v_{new}$ and the connecting edge of the node $v$ in the subgraph $\mathcal{S}$. It is worth noting that these edges are present in the original graph $\mathcal{G}$. Since both the numerator and denominator are positive numbers, the Eq. (A.1) can be transformed into:

$$\left[\sum_{(p,q)\in\mathcal{E}}(x_p - x_q)^2 + \sum_{p\in\mathcal{S}}(x_{new} - x_p)^2\right]\sum_{v_r\in\mathcal{S}} x_r^2 > \sum_{(p,q)\in\mathcal{E}}(x_p - x_q)^2(\sum_{v_r\in\mathcal{S}} x_r^2 + x_{new}^2), \tag{A.2}$$

which can be obviously simplified to:

$$\sum_{v_r\in\mathcal{S}}(x_{new} - x_r)^2 \sum_{v_r\in\mathcal{S}} x_r^2 > x_{new}^2 \sum_{(p,q)\in\mathcal{E}}(x_p - x_q)^2, \tag{A.3}$$

We move the term with the new node to the same side of the equation and rearrange the Eq. (A.3), and we obtain:

$$\frac{\sum_{v_r\in\mathcal{S}}(x_{new} - x_r)^2}{x_{new}^2} > \frac{\sum_{(p,q)\in\mathcal{E}_\mathcal{S}}(x_p - x_q)^2}{\sum_{p\in\mathcal{S}} x_p^2}. \tag{A.4}$$

Note that $RQ(\mathcal{S}) = \frac{\sum_{(p,q)\in\mathcal{E}_\mathcal{S}}(x_p - x_q)^2}{\sum_{p\in\mathcal{S}} x_p^2}$, we denote $\Delta(v_{new}) = \frac{\sum_{v_r\in\mathcal{S}}(x_{new} - x_r)^2}{x_{new}^2}$ and we then obtain the Theorem 1 in Section 3.1.2. $\qquad\square$

## A.2 The proof of Corollary 1

*Proof.* Similar to the proof of Theorem 1, for a new nodeset $v_{new_i} \in \mathcal{V}_{new}$, let $\mathcal{S}'$ be the subgraph after the addition of $\mathcal{V}_{new}$ and it also needs to satisfy $RQ(\mathcal{S}') > RQ(\mathcal{S})$, which is expanded as:

$$\frac{\sum_{(p,q)\in\mathcal{E}}(x_p - x_q)^2 + \sum_{v_{new_i}\in\mathcal{V}_{new_i}}\sum_{v_r\in\mathcal{S}}(x_{new_i} - x_r)^2 + \sum_{(i,j)\in\mathcal{E}_{new}}(x_{new_i} - x_{new_j})^2}{\sum_{v_r\in\mathcal{S}} x_r^2 + \sum_{v_{new_i}\in\mathcal{V}_{new}} x_{new_i}^2}$$
$$> \frac{\sum_{(p,q)\in\mathcal{E}}(x_p - x_q)^2}{\sum_{v_r\in\mathcal{S}} x_r^2}, \tag{A.5}$$

where $\sum_{v_{new_i}\in\mathcal{V}_{new_i}}\sum_{v_r\in\mathcal{S}}(x_{new_i} - x_r)^2$ represents the sum of the feature difference between the newly added nodeset $\mathcal{V}_{new}$ and the connecting edge of the subgraph $\mathcal{S}$. $\sum_{v_{new_i}\in\mathcal{V}_{new}} x_{new_i}^2$ represents the internal sum of the newly added nodeset $\mathcal{V}_{new}$. Similar to the proof of Theorem 1, this formula can be transformed into:

$$\left[\sum_{v_{new_i}\in\mathcal{V}_{new_i}}\sum_{v_r\in\mathcal{S}}(x_{new_i} - x_r)^2 + \sum_{(i,j)\in\mathcal{E}_{new}}(x_{new_i} - x_{new_j})^2\right]\sum_{v_r\in\mathcal{S}} x_r^2 > \sum_{(p,q)\in\mathcal{E}}(x_p - x_q)^2 \sum_{v_{new_i}\in\mathcal{V}_{new}} x_{new_i}^2, \tag{A.6}$$

Rearranging the Eq. (A.6), we get:

$$\frac{\sum_{v_{new}\in\mathcal{V}_{new}}\sum_{v_r\in\mathcal{S}}(x_{new} - x_r)^2 + \sum_{(i,j)\in\mathcal{E}_{\mathcal{V}_{new}}}(x_{new_i} - x_{new_j})^2}{\sum_{v_{new}\in\mathcal{V}_{new}} x_{new}^2} > \frac{\sum_{(p,q)\in\mathcal{E}}(x_p - x_q)^2}{\sum_{v_r\in\mathcal{S}} x_r^2}. \tag{A.7}$$

which is the same as Corollary 1 in Section 3.1.2.

$\qquad\square$

### A.3 The proof of Theorem 2

*Proof.* We define the nodeset $\mathcal{V}^*_{new}$ has the highest $\Delta_{max}(\mathcal{V}_{new})$ and $\Delta_{max}(\mathcal{V}_{new}) > RQ(\mathcal{S})$. To prove that the $\mathcal{V}^*_{new}$ is contained in the optimal subgraph $\mathcal{S}^*$, we give the proof by contradiction. Assume that the negation of the statement is true, so there does not exist $\mathcal{V}^*_{new}$ in $\mathcal{S}^*$. We will discuss the issues based on two scenarios.

In the first scenario, we assume that the current subgraph $\mathcal{S}$ is already the optimal solution. According to Corollary 1, we find that adding $\mathcal{V}^*_{new}$ can increase $RQ(S)$ since it satisfies the condition $\Delta_{max}(\mathcal{V}_{new}) > RQ(\mathcal{S})$. Therefore, it is obvious that the current set $\mathcal{S}$ is not the optimal solution.

In the other scenario, we assume that there is another nodeset $\mathcal{V}'_{new}$ ( $\mathcal{V}'_{new} \cap \mathcal{S}^* = \emptyset$), which together with the current subgraph $\mathcal{S} + \mathcal{V}'_{new}$ forms the optimal solution. According to the corollary 1, we have

$$\Delta_{max}(\mathcal{V}^*_{new}) = \frac{\sum_{\mathcal{V}^*_{new}} \sum_{\mathcal{S}} (x^*_{new} - x_r)^2 + \sum_{\mathcal{E}_{\mathcal{V}^*_{new}}} (x^*_{new_i} - x^*_{new_j})^2}{\sum_{\mathcal{V}^*_{new}} {x^*_{new}}^2}, \tag{A.8}$$

and it satisfies:

$$\begin{cases} \Delta_{max}(\mathcal{V}^*_{new}) > \frac{\sum_{(p,q)\in\mathcal{E}} (x_p - x_q)^2}{\sum_{v_r \in \mathcal{S}} x_r^2}, \\ \Delta_{max}(\mathcal{V}^*_{new}) > \frac{\sum_{\mathcal{V}'_{new}} \sum_{\mathcal{S}} (x'_{new} - x_r)^2 + \sum_{\mathcal{E}_{\mathcal{V}'_{new}}} (x_{new'_i} - x_{new'_j})^2}{\sum_{\mathcal{V}'_{new}} {x'_{new}}^2}, \forall \mathcal{V}'_{new} \subseteq \mathcal{G} - \mathcal{S}. \end{cases} \tag{A.9}$$

To continue with the proof, we present a useful inequality first.

**Lemma 3** (Dan's Favorite Inequality). *Let $a_1, ..., a_n$ and $b_1, ..., b_n$ be positive numbers. Then*

$$\min_i \frac{a_i}{b_i} \le \frac{\sum_i a_i}{\sum_i b_i} \le \max_i \frac{a_i}{b_i}. \tag{A.10}$$

*Proof.* Here we give a classical proof, we have

$$\sum_i a_i = \sum_i b_i \left( \frac{a_i}{b_i} \right) \le \sum_i b_i \left( \max_j \frac{a_j}{b_j} \right) = \left( \max_j \frac{a_j}{b_j} \right) \sum_i b_i, \tag{A.11}$$

So,

$$\frac{\sum a_i}{\sum b_i} \le \max_j \frac{a_j}{b_j}, \tag{A.12}$$

One can similarly prove

$$\frac{\sum a_i}{\sum b_i} \ge \min_j \frac{a_j}{b_j}. \tag{A.13}$$

$\square$

Combining Lemma 3 and Eq. (A.9), we obtain the following inequality.

$$\Delta_{max}(\mathcal{V}^*_{new}) >$$
$$\frac{\sum_{(p,q)\in\mathcal{E}} (x_p - x_q)^2 + \sum_{\mathcal{V}'_{new}} \sum_{\mathcal{S}} (x'_{new} - x_r)^2 + \sum_{\mathcal{E}_{\mathcal{V}'_{new}}} (x'_{new_i} - x'_{new_j})^2}{\sum_{v_r \in \mathcal{S}} x_r^2 + \sum_{\mathcal{V}'_{new}} {x'_{new}}^2}, \forall \mathcal{V}'_{new} \subseteq \mathcal{G} - \mathcal{S}. \tag{A.14}$$

Analyzing the above equation reveals that the right side of the formula is $RQ(\mathcal{V}'_{new} + \mathcal{S})$. That is, for any $\mathcal{V}'_{new}$, adding $\mathcal{V}^*_{new}$ still makes $RQ(\mathcal{V}'_{new} + \mathcal{S})$ increasing according to the Corollary 1, which contradicts the assumption that $RQ(\mathcal{V}'_{new} + \mathcal{S})$ is the optimal solution.

$$\Delta_{max}(\mathcal{V}_{new}) = \max_{\mathcal{V}_{new} \subseteq \mathcal{G}-\mathcal{S}} \Delta(\mathcal{V}_{new}), \text{ and } \Delta_{max}(\mathcal{V}_{new}) > RQ(\mathcal{S}). \tag{A.15}$$

However, identifying the maximum $\Delta_{max}(\mathcal{V}_{new})$ from the $\mathcal{V}_{new} \subseteq \mathcal{G} - \mathcal{S}$ is still a NP-hard problem. We consider relaxing any nodesets $\mathcal{V}'_{new}$ to any **connected** nodesets $\mathcal{V}^c_{new}$. Any nodesets can be decomposed into several disconnected smaller nodesets, that is, $\mathcal{V}'_{new} = \mathcal{V}^{c1}_{new} \cup \mathcal{V}^{c2}_{new} \cup ...$. Since there are no edges connecting these nodesets, the following decomposition formula can be derived.

$$\begin{cases} \sum_{\mathcal{V}'_{new}} \sum_{\mathcal{S}} (x'_{new} - x_r)^2 = \sum_{\mathcal{V}^{c1}_{new}} \sum_{\mathcal{S}} (x^{c1}_{new} - x_r)^2 + \sum_{\mathcal{V}^{c2}_{new}} \sum_{\mathcal{S}} (x^{c2}_{new} - x_r)^2 + \dots, \\ \sum_{\mathcal{E}_{\mathcal{V}'_{new}}} (x'_{new_i} - x'_{new_j})^2 = \sum_{\mathcal{E}_{\mathcal{V}^{c1}_{new}}} (x^{c1}_{new_i} - x^{c1}_{new_j})^2 + \sum_{\mathcal{E}_{\mathcal{V}^{c2}_{new}}} (x^{c2}_{new_i} - x^{c2}_{new_j})^2 + \dots, \\ \sum_{\mathcal{V}'_{new}} {x'_{new}}^2 = \sum_{\mathcal{V}^{c1}_{new}} {x^{c1}_{new}}^2 + \sum_{\mathcal{V}^{c2}_{new}} {x^{c2}_{new}}^2 + \dots. \end{cases}$$

(A.16)

Considering the condition that maximizes the Rayleigh quotient of any connected $\mathcal{V}^{ci}_{new}$,

$$\Delta_{max}(\mathcal{V}^*_{new}) > \Delta_{max}(\mathcal{V}^{ci}_{new}) = \frac{\sum_{\mathcal{V}^{ci}_{new}} \sum_{\mathcal{S}} (x^{ci}_{new} - x_r)^2 + \sum_{\mathcal{E}_{\mathcal{V}^{ci}_{new}}} (x^{ci}_{new_i} - x^{ci}_{new_j})^2}{\sum_{\mathcal{V}^{ci}_{new}} {x^{ci}_{new}}^2}. \quad \text{(A.17)}$$

According to Lemma 3, Eq. (A.14) is still satisfied. Therefore, we derive that $\mathcal{V}^*_{new}$ is contained in the optimal solution.

$\square$

# B The Pseudocode of MRQSampler Algorithm

We give the pseudocode of MRQSampler in Algorithm 1, which illustrates the algorithm for finding the subgraph with the target node that maximizes the Rayleigh Quotient. In section 3.1.2, we give a diagram of the sampling range of 2-hop and 1-hop cases. For the completeness of the theory, we give the complete algorithm for arbitrary $k$-hop cases in the pseudocode form.

For node $r$, we focus on the $k$-hop spanning tree $T$ with $r$ as the root node. And for any node $v$ in $T$ except for the root $r$, $\Delta_{max}[v]$ is defined as:

$$\Delta_{max}[v] := \max_{\mathcal{S} \subseteq \mathcal{T}_v} \frac{(x_i - x_p)^2 + \sum_{(i,j) \subseteq \mathcal{E}_{\mathcal{S}}} (x_i - x_j)^2}{\sum_{i \subseteq \mathcal{S}} x_i^2}. \quad \text{(A.18)}$$

where $\mathcal{S} \subseteq \mathcal{T}_v$ are the connected subgraphs of the subtree $T_v$ with $v$ as the root node, and $p$ is the parent node of the node $v$. As described in the Section 3.1.2, we break the computation into two steps:

- **Stage 1**: Compute and store the maximum $\Delta_{max}[v]$ for subtrees rooted with each node $v$ except for the root $r$, which is performed by recursively calling the function `GetMaxDeltas` in Algorithm 1.

- **Stage 2**: Use these memorized results to compute the optimal Rayleigh Quotient $RQ$ and its corresponding subgraph, which is performed by the function `MRQSampler`.

In **Stage 1**, the first thing we need to know is how we get $\Delta_{max}[v]$. Similar to the analysis of the Theorem 2, we can also obtain the condition that the nodeset is in the final optimal subgraph with largest $\Delta_{max}[v]$:

$$\Delta_{max}[v_{new}] = \max_{\{\tilde{v}_{new}\}} \Delta(\tilde{\mathcal{V}}_{new}), \text{ and } \Delta_{max}[v_{new}] > \Delta[v]. \quad \text{(A.19)}$$

This process is similar to finding the maximum $RQ$. In other words, we keep retrieving the unselected descendants with maximum $\Delta_{max}[v_{new}]$, and then check whether its $\Delta_{max}[v_{new}]$ exceeds the current $\Delta[v]$. If it does, the inclusion of the optimal subgraph with it can increase the current $\Delta[v]$, otherwise, it can no longer be increased by adding any descendants and the maximum $\Delta_{max}[v]$ is reached.

---

**algorithm 1** MRQSampler

---

**Globals:** $r$ : the original root of the tree; $i$ : an arbitrary node; $x[i]$ : the node $i$'s feature; $T_r[i]$ : an array that stores the child nodes of node $i$ in the tree; $\delta[i] \leftarrow \{\Delta_{max}[i], a_i, b_i, N, I\}$: an array of structures that stores the maximum $\Delta_{max}[i]$ achievable by any connected subgraph $\mathcal{V}$ containing the node $i$ within the subtree rooted at $i$ and $a_i, b_i$ stores the numerator and denominator of $\Delta_{max}[i]$. $N$ is the optimal selected nodes, $I$ is the inferior candidates,

1: # The function for computing $\Delta_{max}[i]$ in STAGE I
2: **Input:** $v \leftarrow$ root of the current subtree; $p \leftarrow$ parent of $v$
3: **Output:** $\delta[i] \leftarrow$ structure array with $\Delta_{max}[i]$ and correlated variables
4: **function** GETMAXDELTAS($v, p$)
5:     $N, I, U \leftarrow \{\}$         ▷ Optimal selected nodes, Inferior candidates, Un-selected children of $v$
6:     $Q \leftarrow$ SortedSet()                            ▷ Candidates queue
7:     $a_v \leftarrow (x[v] - x[p])^2$              ▷ Initialize the numerator of the maximum $\Delta_{max}[v]$
8:     $b_v \leftarrow x[v]^2$                    ▷ Initialize the denominator of the maximum $\Delta_{max}[v]$
9:     $\Delta_{max}[v] \leftarrow a_v/b_v$            ▷ Initialize the maximum $\Delta_{max}[v]$ for current sub-tree
10:     **for** $c$ in $T[v]$ **do**
11:        $\delta[c] \leftarrow$ GetMaxDeltas($c, v$)           ▷ Result of the subtree rooted with child $c$
12:        $Q$.insert($[c, \delta[c]]$)
13:        $U$.insert($c$)
14:     **while** $Q$.size() $\neq 0$ **do**
15:        $c, \delta[c] \leftarrow Q$.pop_largest() ▷ Retrieve the candidate $c$ with $\Delta_{max}[c]$ and structure $\delta[c]$
16:        **if** $\Delta_{max}[v] > (a_v + \delta[c].a_c)/(b_v + \delta[c].b_c)$ **then**    ▷ Optimization criterion of $\Delta_{max}[v]$
17:           Break
18:        $U$.remove_if_exist($c$)
19:        $a_v \leftarrow a_v + \delta[c].a_c$                           ▷ Update the maximum $\Delta_{max}[v]$
20:        $b_v \leftarrow b_v + \delta[c].b_c$
21:        $\Delta_{max}[v] \leftarrow a_v/b_v$
22:        $Q$.insert($\delta[c].I$)                         ▷ Activate the inferior candidates
23:        $N$.insert($\delta[c].N$)
24:     $I \leftarrow Q$                       ▷ The remaining candidates are the inferior ones
25:     $I$.insert($U$)                  ▷ Add the un-selected children to the inferior set
26:     $\delta[v] \leftarrow \{\Delta_{max}[v], a_v, b_v, N, I\}$                 ▷ Memorise the results
27:     **return** $\delta[v]$

28:

29: # The main function of MRQSampler in STAGE II
30: **Input:** $r \leftarrow$ the original root of the tree (target sampling node)
31: **Output:** $RQ_{max} \leftarrow$ maximum $RQ$; $N \leftarrow$ optimal sampling nodeset
32: **function** MRQSAMPLER($r$)
33:     $a_{RQ} \leftarrow 0$                                 ▷ Initialize the numerator of the $RQ_{max}$
34:     $b_{RQ} \leftarrow x[r]^2$                           ▷ Initialize the denominator of the $RQ_{max}$
35:     $RQ_{max} \leftarrow a_{RQ}/b_{RQ}$
36:     $Q \leftarrow$ SortedSet()
37:     **for** $c$ in $T[r]$ **do**
38:        $\delta[c] \leftarrow$ GetMaxDeltas(c, $r$)          ▷ Recursively calculate the $\Delta_{max}$ in Stage 1
39:        $Q$.insert($[c, \delta[c]]$)
40:     **while** $Q$.size() $\neq 0$ **do**
41:        $c, \delta[c] \leftarrow Q$.pop_largest()
42:        **if** $RQ_{max} > (a_{RQ} + \delta[c].a_c)/(b_{RQ} + \delta[c].b_c)$ **then** ▷ Optimization criterion of the RQ
43:           Break
44:        $a_{RQ} \leftarrow a_{RQ} + \delta[c].a_c$                              ▷ Update the result
45:        $b_{RQ} \leftarrow b_{RQ} + \delta[c].b_c$
46:        $RQ_{max} \leftarrow a_{RQ}/b_{RQ}$
47:        $Q$.insert($\delta[c].I$)                         ▷ Activate the inferior candidates
48:        $N$.insert($\delta[c].N$)                      ▷ Update the selected nodeset
49:     **return** $\{RQ_{max}, N\}$

---

For ease of computation, we store the optimal $\Delta_{max}[v]$ by its numerator $a_v$ and denominator $b_v$ (line 7-9). Next, we recursively calculate the result for each subtree rooted by every child $c$ of the current root $v$ (line 11). To simplify complexity in the following steps, we store these results in a sorted container (e.g. binary search tree) $Q$ (line 12). Next, we keep retrieving the subgraph with the highest $\Delta_{max}[c]$ from $Q$ and compute whether the $\Delta_{max}[c]$ increases after adding it to the current solution (line 15-17). According to Eq. (A.19), we obtain the optimal $\Delta_{max}[v]$ for the current subtree with root $v$ when the candidate cannot make $\Delta_{max}[v]$ larger. Moreover, sets from subtrees that are not optimal for $v$ may still be selected at higher levels. Therefore, we also need to keep track of those inferior subtrees and re-consider them when other subtrees that connect to them are merged into the solution (line 24-25). Note that subtrees in $I$ are only considered as candidates when the optimal subgraph with root $v$ is selected at a higher layer (the "activation" in line 22).

In **Stage 2**, the overall routine for obtaining $RQ_{max}$ is very similar to the one in `GetMaxDeltas`, except that the initial value is set to $RQ_{max} = \frac{0}{x[r]^2}$ since the root $r$ has no parent node. In other words, the algorithmic logic of the two functions in Stage 1 and Stage 2 is similar. Stage 2 can be regarded as a special case of Stage 1 without a parent node, utilizing the implementation of memoization from Stage 1.

Assuming that the k-hop spanning tree $T$ has $K$ nodes, the time complexity of Algorithm $\mathcal{O}(KlogK)$, since we will at worst examine each edge and sort them. Notice that the computation is irrelevant between different nodes, it can be further accelerated by simultaneously processing multiple nodes. In practice, we observe that the optimal choice of k-hop is typically <= 2, and thus the recursive computation can be unrolled thus further improving the efficiency.

# C    Implementation Details

**Node-level Baselines.**    **GCN** (Graph Convolutional Network [24]) leverages convolution operations to propagate information from nodes to their neighbors. **SGC** (Simple Graph Convolution [56]) further simplifies GCN by removing non-linearities and collapsing weight matrices between consecutive layers to improve efficiency. **GIN** (Graph Isomorphism Network [57]) captures graph structures by generating identical embeddings for structurally identical graphs, ensuring invariance to node label permutations. **GraphSAGE** (Graph Sample and AggregatE [16]) generates node embeddings through sampling and aggregating features from local neighborhoods, supporting inductive learning. **GAT** (Graph Attention Networks [52]) incorporates an attention mechanism to assign varying importance levels to different nodes during neighborhood aggregation, focusing on the most informative parts. **PNA** (Principle Neighbor Aggregation [7]) combines multiple aggregators with degree-scalers for effective neighborhood aggregation. **AMNet** (Adaptive Multi-frequency Graph Neural Network [5]) captures both low and high-frequency signals by stacking multiple BernNets [18], adaptively combining signals of different frequencies. **BWGNN** (Beta Wavelet Graph Neural Network [50]) employs the Beta kernel to tackle higher frequency anomalies with flexible band-pass filters.

**Graph-level Baselines.**    **OCGIN** [66] is a one-class graph-level anomaly detector based on a graph isomorphism network that addresses performance fluctuations in general graph classification methods. **OCGTL** [46] combines deep one-class classification with graph transformation learning. **GlocalKD** learns rich global and local normal pattern information by joint distillation of graph and node representations. **iGAD** [63] employs an attribute-aware GNN and a substructure-aware deep random walk kernel to achieve dual-discriminative capability for anomalous attributes and substructures. **GmapAD** [44] proposes an explainable graph mapping approach that projects graphs into a latent space for effective anomaly detection. **RQGNN** [11] identifies differences in the spectral energy distributions between anomalous and normal graphs.

**Multi-task Baselines.**    **GraphPrompt** [37] learns different task-specific prompt vectors for each task, which are added to the graph-level representations by element-wise multiplication. **All-in-One** [48] treats an extra subgraph as a prompt and merges it with the original graph by cross links.

**Hardware Specifications.**    Our experiments were mainly carried out on a Linux server equipped with dual AMD EPYC 7763 64-core CPU processor, 256GB RAM, and an NVIDIA RTX 4090 GPU with 24GB memory. Some of the extremely large datasets, such as T-Finance, and certain memory-intensive baselines were implemented on the NVIDIA 8*A800 GPUs. We mark the results as OOT (Out of Time) if the model training exceeds 2 days. For some large datasets, methods with GPU memory requirements exceeding 80GB were marked as OOM (Out of Memory), such as iGAD

and GmapAD, and were attempted to be run on the CPU. iGAD was successfully completed, but GmapAD still encountered a timeout issue.

**Metrics.** We utilize three widely used metrics to evaluate the performance of all methods: **F1-macro**, **AUROC** and **AUPRC**. **F1-macro** is the unweighted mean of the F1-scores for the two classes, disregarding the imbalance ratio between normal and anomaly labels. **AUROC** represents the area under the Receiver Operating Characteristic Curve. **AUPRC** represents the area under the Precision-Recall Curve, emphasizing model performance on imbalanced datasets by focusing on the trade-off between precision and recall.

**Hyperparameter Tuning.** As described in Section 3.2, we first use an unsupervised model based on GraphMAE [21] to learn the general representation of the input features. The hyperparameters for this step are set to the default values from the official GraphMAE implementation, with 50 training epochs. Table 7 lists all the hyperparameters used in our model along with their corresponding search spaces. During training, we conduct a **grid search** to identify the model that achieves the highest AUROC score on the validation set. Finally, we evaluate the selected model on the test sets and report the performance metrics.

Table 7: Hyperparameters search space for UniGAD.

| Hyperparameter | Distribution |
|---|---|
| learning rate | $\text{Range}(5^{-4}, 10^{-2})$ |
| activation | [ReLU, LeakyReLU, Tanh] |
| hidden dimension | [16,32,64] |
| MRQSampler tree depth | [1,2] |
| GraphStitch Network layer | [1,2,3]$\times$2 |
| epochs | [100, 200, 300, 400, 500] |

## D   Limitations and Impacts

Since the GNN encoder we employ mainly focuses on node-level features, the learned representations may not be perfectly suited for edge and graph level tasks. Therefore, we leave the exploration of how to integrate multiple tasks in the pre-training phase to future work. As a generalized graph anomaly detection model, our work will be helpful in detecting classical graph anomaly applications, such as financial fraud, cybercrime, etc. On the other hand, an error in the recognition result may put normal groups or behaviors into anomalies, causing disturbance for the normal users in the graph network.

## E   Additional Experimental Results

We also provide the results of all experiments under the F1-macro and AUPRC evaluation metrics. Similar to the arrangement in the main text, for F1-macro, we show the results of multi-level performance comparison under F1-macro metric in Table 8 and Table 9. The results of Zero-Shot Comparison under F1-macro metric are in Table 10 and Table 11. For AUPRC, we show the results of multi-level performance comparison under AUPRC metric in Table 12 and Table 13. The results of Zero-Shot Comparison under AUPRC metric are in Table 14 and Table 15. It can be observed from these tables that similar conclusions can be drawn as with the AUROC results in Section 4. UniGAD demonstrates superior performance across most datasets, regardless of whether unified or zero-shot performance is evaluated.

Table 8: Comparison of unified performance (F1-macro) at both node and edge levels with different single-level methods, multi-task methods, and our proposed method.

| Dataset Task-level | Reddit Node | Edge | Weibo Node | Edge | Amazon Node | Edge | Yelp Node | Edge | Tolokers Node | Edge | Questions Node | Edge | T-Finance Node | Edge |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **Node-level** | | | | | | | | | | | | | | |
| GCN | 38.81 | / | 92.90 | / | 62.45 | / | 42.84 | / | 58.64 | / | 48.74 | / | 70.61 | / |
| GIN | 27.22 | / | 92.01 | / | 72.21 | / | 58.78 | / | 59.40 | / | 49.95 | / | 76.81 | / |
| GraphSAGE | 28.30 | / | 85.05 | / | 64.05 | / | 64.20 | / | 63.71 | / | 51.07 | / | 62.63 | / |
| SGC | 12.42 | / | 90.49 | / | 57.43 | / | 47.26 | / | 54.48 | / | 48.59 | / | 56.69 | / |
| GAT | 31.28 | / | 92.46 | / | 87.96 | / | 57.56 | / | 62.68 | / | 47.28 | / | 69.11 | / |
| BernNet | 46.27 | / | 89.59 | / | 89.16 | / | 61.42 | / | 58.73 | / | 47.59 | / | 70.52 | / |
| PNA | 14.08 | / | 88.93 | / | 61.84 | / | 52.52 | / | 58.49 | / | 46.38 | / | 27.69 | / |
| AMNet | 45.14 | / | 89.04 | / | 89.67 | / | 58.75 | / | 58.47 | / | 49.90 | / | 74.31 | / |
| BWGNN | 42.25 | / | 86.94 | / | 90.49 | / | 64.89 | / | 63.43 | / | 52.09 | / | 80.57 | / |
| **Edge-level** | | | | | | | | | | | | | | |
| GCNE | / | 42.97 | / | 92.32 | / | 47.99 | / | 42.32 | / | 63.15 | / | 58.40 | / | 79.07 |
| GINE | / | 36.89 | / | 91.54 | / | 47.82 | / | 53.14 | / | 60.54 | / | 56.96 | / | 73.40 |
| SAGEE | / | 11.94 | / | 89.59 | / | 57.79 | / | 57.87 | / | 65.95 | / | **74.15** | / | 67.12 |
| SGCE | / | 41.08 | / | 86.70 | / | 55.97 | / | 45.13 | / | 56.50 | / | 52.07 | / | 64.51 |
| GATE | / | 40.65 | / | 90.53 | / | 64.69 | / | 49.99 | / | 61.43 | / | 62.80 | / | 68.75 |
| BernE | / | 39.85 | / | 92.15 | / | 69.12 | / | 59.76 | / | 61.53 | / | 67.32 | / | 63.16 |
| PNAE | / | 23.03 | / | 92.30 | / | 49.27 | / | 52.94 | / | 64.98 | / | 65.39 | / | 65.74 |
| AME | / | 41.11 | / | 87.04 | / | 66.27 | / | 57.09 | / | 61.42 | / | 66.74 | / | 57.45 |
| BWE | / | 45.36 | / | 91.72 | / | 67.56 | / | 59.30 | / | 65.09 | / | 66.28 | / | 70.88 |
| **Multi-task** | | | | | | | | | | | | | | |
| GraphPrompt-U | 31.23 | 38.54 | 50.64 | 46.53 | 40.93 | 35.95 | 40.90 | 42.94 | 48.26 | 48.34 | 39.43 | 44.61 | OOT | OOT |
| All-in-One-U | 49.12 | 2.41 | 51.23 | 48.65 | 48.67 | 2.45 | 14.43 | 46.29 | 50.17 | 47.90 | 48.81 | 33.29 | OOT | OOT |
| **UniGAD (Ours)** | | | | | | | | | | | | | | |
| UniGAD - GCN | **56.70** | **53.80** | **95.75** | **94.29** | 69.39 | 59.12 | 58.23 | 56.76 | 65.20 | 64.55 | 58.06 | 57.77 | 84.92 | 84.08 |
| UniGAD - BWG | 54.08 | 51.44 | 95.35 | 94.22 | **91.33** | **73.59** | **70.16** | **63.57** | **68.15** | **66.20** | **59.45** | 57.54 | **89.75** | **84.90** |

Table 9: Comparison of unified performance (F1-macro) at both node and graph levels with different single-level methods, multi-task methods, and our proposed method.

| Dataset Task-level | BM-MN Node | Graph | BM-MS Node | Graph | BM-MT Node | Graph | MUTAG Node | Graph | MNIST0 Node | Graph | MNIST1 Node | Graph | T-Group Node | Graph |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **Node-level** | | | | | | | | | | | | | | |
| GCN | 68.25 | / | 77.77 | / | 69.72 | / | 90.41 | / | 92.03 | / | 91.95 | / | 49.50 | / |
| GIN | 32.96 | / | 24.25 | / | 25.69 | / | 92.33 | / | 88.88 | / | 88.04 | / | 49.24 | / |
| GraphSAGE | 32.96 | / | 24.25 | / | 25.69 | / | 88.87 | / | 99.99 | / | 99.99 | / | 50.77 | / |
| SGC | 32.96 | / | 24.33 | / | 25.72 | / | 54.95 | / | 49.04 | / | 82.70 | / | 49.04 | / |
| GAT | 32.96 | / | 24.25 | / | 25.69 | / | 92.07 | / | 99.94 | / | 99.96 | / | 50.29 | / |
| BernNet | 35.04 | / | 51.71 | / | 30.26 | / | 86.76 | / | 99.99 | / | 99.99 | / | 52.84 | / |
| PNA | 32.96 | / | 24.25 | / | 25.69 | / | 87.49 | / | 98.83 | / | 99.27 | / | 49.88 | / |
| BWGNN | 82.48 | / | 75.22 | / | 76.19 | / | 92.75 | / | 99.99 | / | 99.99 | / | 51.81 | / |
| **Graph-level** | | | | | | | | | | | | | | |
| OCGIN | / | 46.15 | / | 46.15 | / | 46.11 | / | 39.78 | / | 47.79 | / | 49.74 | / | 48.91 |
| OCGTL | / | 46.15 | / | 46.15 | / | 46.15 | / | 39.62 | / | 47.41 | / | 47.02 | / | 48.91 |
| GLocalKD | / | 12.50 | / | 12.50 | / | 12.50 | / | 25.59 | / | 8.98 | / | 10.11 | / | 4.09 |
| iGAD | / | 68.29 | / | 81.59 | / | 89.89 | / | 89.78 | / | 87.73 | / | 95.04 | / | 46.51 |
| GmapAD | / | 46.15 | / | 46.15 | / | 46.15 | / | 75.48 | / | OOM | / | OOM | / | OOM |
| RQGNN | / | **95.46** | / | 93.02 | / | **97.56** | / | 89.39 | / | 93.42 | / | 96.99 | / | 48.91 |
| **Multi-task** | | | | | | | | | | | | | | |
| GraphPrompt-U | 36.95 | 45.55 | 37.46 | 12.86 | 47.92 | 46.01 | 83.08 | 45.70 | 80.66 | 52.39 | 80.49 | 28.25 | 50.77 | 49.78 |
| All-in-One-U | 34.98 | 12.86 | 21.24 | 20.38 | 41.36 | 12.86 | 38.71 | 25.71 | OOT | OOT | OOT | OOT | OOT | OOT |
| **UniGAD (Ours)** | | | | | | | | | | | | | | |
| UniGAD - GCN | **99.20** | 83.62 | **99.57** | **95.86** | **96.18** | 70.50 | **93.33** | **90.00** | 92.17 | 93.38 | 92.49 | 97.23 | 64.98 | 77.04 |
| UniGAD - BWG | 87.91 | 55.89 | 83.79 | 61.67 | 82.53 | 51.36 | 93.07 | 89.19 | **99.99** | **95.54** | **99.99** | **97.60** | **68.69** | **78.09** |

Table 10: Zero-shot transferability (F1-macro) at node and edge levels.

| Methods | Reddit N→E | E→N | Weibo N→E | E→N | Amazon N→E | E→N | Yelp N→E | E→N | Tolokers N→E | E→N | Questions N→E | E→N | T-Finance N→E | E→N |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| GraphPrompt-U | 29.87 | 41.00 | 50.48 | 47.06 | 42.19 | 33.22 | 44.77 | 41.44 | 47.77 | 43.72 | 47.63 | 39.00 | OOT | OOT |
| All-in-One-U | 2.38 | 49.17 | 36.82 | 51.58 | 12.84 | 23.6 | 12.22 | 46.04 | 33.39 | 49.83 | 33.08 | 49.54 | OOT | OOT |
| UniGAD - GCN | **50.61** | **50.58** | **94.44** | **94.29** | 56.70 | 61.88 | 51.29 | 53.36 | 61.14 | 57.14 | 52.01 | 52.26 | 80.85 | 69.21 |
| UniGAD - BWG | 49.79 | 50.04 | 92.11 | 93.62 | **68.47** | **85.07** | **63.69** | **71.10** | **65.46** | **65.57** | **55.81** | **53.44** | **83.35** | **87.61** |

Table 11: Zero-shot transferability (F1-macro) at node and graph levels.

| Methods | BM-MN N→G | G→N | BM-MS N→G | G→N | BM-MT N→G | G→N | MUTAG N→G | G→N | MNIST0 N→G | G→N | T-Group N→G | G→N |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| GraphPrompt-U | 12.86 | 34.98 | 20.59 | 42.78 | 46.01 | 43.85 | 42.15 | 27.73 | 26.16 | 26.75 | 48.47 | 43.64 |
| All-in-One-U | 12.86 | 34.98 | 46.01 | 41.25 | 12.86 | 22.74 | 39.53 | 48.80 | OOT | OOT | OOT | OOT |
| UniGAD - GCN | **53.43** | **83.88** | **57.84** | **74.63** | **52.78** | **54.87** | 39.89 | **77.47** | **65.78** | **63.75** | **66.47** | **49.22** |
| UniGAD - BWG | 46.15 | 59.27 | 46.15 | 52.27 | 46.22 | 39.54 | **45.58** | 66.32 | 44.56 | 60.84 | 63.92 | 48.24 |

Table 12: Comparison of unified performance (AUPRC) at both node and edge levels with different single-level methods, multi-task methods, and our proposed method.

| | Dataset Task-level | Reddit Node | Edge | Weibo Node | Edge | Amazon Node | Edge | Yelp Node | Edge | Tolokers Node | Edge | Questions Node | Edge | T-Finance Node | Edge |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Node-level | GCN | 5.84 | / | 94.55 | / | 36.21 | / | 20.58 | / | 43.68 | / | 12.20 | / | 70.81 | / |
| | GIN | 5.89 | / | 91.28 | / | 75.74 | / | 33.09 | / | 39.71 | / | 12.79 | / | 61.79 | / |
| | GraphSAGE | 5.44 | / | 84.97 | / | 55.38 | / | 45.27 | / | 48.90 | / | 16.72 | / | 19.62 | / |
| | SGC | 4.27 | / | 90.70 | / | 33.48 | / | 16.13 | / | 36.90 | / | 9.90 | / | 30.35 | / |
| | GAT | 6.15 | / | 90.18 | / | 85.61 | / | 37.51 | / | 46.01 | / | 15.82 | / | 54.70 | / |
| | BernNet | 7.34 | / | 88.39 | / | 87.08 | / | 46.22 | / | 43.04 | / | 15.11 | / | 67.65 | / |
| | PNA | 5.71 | / | 93.13 | / | 37.28 | / | 27.34 | / | 42.98 | / | 11.57 | / | 23.07 | / |
| | AMNet | 7.52 | / | 88.59 | / | 87.26 | / | 46.18 | / | 43.22 | / | 14.39 | / | 74.73 | / |
| | BWGNN | 6.41 | / | 92.81 | / | **88.57** | / | 50.32 | / | 49.44 | / | 16.21 | / | 84.94 | / |
| Edge-level | GCNE | / | 4.68 | / | 94.56 | / | 16.59 | / | 22.68 | / | 55.15 | / | 27.83 | / | 62.12 |
| | GINE | / | 5.01 | / | 91.84 | / | 25.08 | / | 28.28 | / | 46.03 | / | 27.34 | / | 52.01 |
| | SAGEE | / | 5.31 | / | 91.07 | / | 20.28 | / | 34.22 | / | **60.44** | / | **50.49** | / | 18.79 |
| | SGCE | / | 3.00 | / | 89.04 | / | 14.54 | / | 17.37 | / | 51.41 | / | 17.84 | / | 30.22 |
| | GATE | / | 5.32 | / | 86.61 | / | 40.30 | / | 31.96 | / | 51.03 | / | 32.70 | / | 33.47 |
| | BernE | / | 4.89 | / | 91.34 | / | 39.83 | / | 32.70 | / | 55.19 | / | 41.52 | / | 45.01 |
| | PNAE | / | 4.51 | / | 95.24 | / | 16.03 | / | 28.28 | / | 57.46 | / | 37.61 | / | 54.13 |
| | AME | / | 5.00 | / | 87.03 | / | 39.07 | / | 32.46 | / | 53.53 | / | 40.88 | / | 43.70 |
| | BWE | / | 5.26 | / | 93.08 | / | 38.83 | / | 35.33 | / | 58.42 | / | 42.72 | / | 68.13 |
| Multi-task | GraphPrompt-U | 3.60 | 2.92 | 17.22 | 7.31 | 6.62 | 2.64 | 12.41 | 13.63 | 22.19 | 33.52 | 3.25 | 5.22 | OOT | OOT |
| | All-in-One-U | 4.07 | 2.93 | 6.41 | 5.18 | 1.02 | 3.13 | 46.10 | 32.16 | 21.64 | 32.16 | 2.57 | 4.09 | OOT | OOT |
| UniGAD (Ours) | UniGAD - GCN | **9.73** | **5.82** | **96.79** | **95.65** | 38.06 | 15.53 | **61.00** | **40.90** | 46.38 | 54.05 | 15.58 | 15.96 | 75.30 | 69.90 |
| | UniGAD - BWG | 5.19 | 3.29 | 96.54 | 93.66 | 87.28 | **42.01** | 27.42 | 24.65 | 50.80 | 56.89 | **17.35** | 19.34 | **85.34** | **74.37** |

Table 13: Comparison of unified performance (AUPRC) at both node and graph levels with different single-level methods, multi-task methods, and our proposed method.

| | Dataset Task-level | BM-MN Node | Graph | BM-MS Node | Graph | BM-MT Node | Graph | MUTAG Node | Graph | MNIST0 Node | Graph | MNIST1 Node | Graph | T-Group Node | Graph |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Node-level | GCN | 84.82 | / | 78.23 | / | 83.12 | / | 82.17 | / | 91.24 | / | 91.29 | / | 8.78 | / |
| | GIN | 52.80 | / | 32.44 | / | 36.98 | / | 81.91 | / | 87.62 | / | 87.33 | / | 1.65 | / |
| | GraphSAGE | 49.17 | / | 32.01 | / | 34.55 | / | 80.20 | / | 99.93 | / | 99.94 | / | 5.79 | / |
| | SGC | 51.73 | / | 31.24 | / | 36.42 | / | 34.32 | / | 82.69 | / | 82.66 | / | 3.93 | / |
| | GAT | 54.33 | / | 40.83 | / | 44.23 | / | 82.44 | / | 99.40 | / | 99.90 | / | 6.56 | / |
| | BernNet | 58.11 | / | 38.34 | / | 42.79 | / | 72.17 | / | 99.99 | / | 99.99 | / | 13.51 | / |
| | PNA | 72.16 | / | 38.32 | / | 58.97 | / | 70.16 | / | 98.48 | / | 98.50 | / | 1.03 | / |
| | BWGNN | 91.85 | / | 70.10 | / | 78.53 | / | 84.33 | / | 99.99 | / | 99.99 | / | 16.30 | / |
| Graph-level | OCGIN | / | 89.40 | / | 48.80 | / | 41.14 | / | 31.02 | / | 12.99 | / | 18.09 | / | 4.46 |
| | OCGTL | / | 76.72 | / | 46.13 | / | 41.38 | / | 33.87 | / | 9.94 | / | 11.27 | / | 4.30 |
| | GLocalKD | / | 7.71 | / | 9.05 | / | 17.39 | / | 23.01 | / | 6.96 | / | 13.49 | / | 2.51 |
| | iGAD | / | 68.36 | / | 74.57 | / | 84.66 | / | 91.07 | / | 94.79 | / | 97.98 | / | 5.92 |
| | GmapAD | / | 14.29 | / | 14.29 | / | 14.29 | / | 60.96 | / | OOM | / | OOM | / | OOM |
| | RQGNN | / | **99.32** | / | 97.60 | / | 99.36 | / | 91.27 | / | 97.62 | / | 98.39 | / | 7.98 |
| Multi-task | GraphPrompt-U | 43.87 | 15.15 | 26.15 | 14.76 | 27.78 | 14.83 | 70.41 | 60.70 | 82.89 | 36.25 | 83.30 | 5.97 | 1.06 | 4.25 |
| | All-in-One-U | 57.75 | 8.58 | 35.75 | 9.16 | 25.13 | 20.23 | 5.86 | 33.09 | OOT | OOT | OOT | OOT | OOT | OOT |
| UniGAD (Ours) | UniGAD - GCN | **99.63** | 73.54 | **99.91** | **98.39** | **99.73** | **99.99** | 86.60 | 91.66 | 95.94 | 94.86 | 96.38 | 98.36 | 21.53 | 44.95 |
| | UniGAD - BWG | 91.19 | 23.83 | 85.81 | 30.89 | 84.93 | 14.74 | **87.15** | **92.00** | 99.99 | 97.92 | 99.99 | **98.60** | 31.31 | 55.64 |

Table 14: Zero-shot transferability (AUPRC) at node and edge levels.

| Methods | Reddit N→E | E→N | Weibo N→E | E→N | Amazon N→E | E→N | Yelp N→E | E→N | Tolokers N→E | E→N | Questions N→E | E→N | T-Finance N→E | E→N |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| GraphPrompt-U | 2.80 | 3.00 | 7.33 | 16.50 | 2.37 | 9.83 | 13.69 | 13.26 | 34.14 | 21.59 | 5.49 | 3.57 | OOT | OOT |
| All-in-One-U | 2.99 | 4.15 | 5.37 | 6.53 | 10.20 | 3.13 | 14.43 | 13.49 | 31.77 | 21.79 | 4.11 | 3.04 | OOT | OOT |
| UniGAD - GCN | **3.89** | **5.44** | 93.35 | 95.57 | 10.86 | 29.44 | 22.00 | 24.98 | 51.50 | 40.51 | 12.44 | **6.76** | 69.67 | 69.74 |
| UniGAD - BWG | 3.11 | 4.09 | 86.47 | 93.21 | **28.07** | **78.56** | **35.99** | **55.04** | 54.26 | 46.36 | **13.89** | 5.80 | **70.61** | **81.02** |

Table 15: Zero-shot transferability (AUPRC) at node and graph levels.

| Methods | BM-MN N→G | G→N | BM-MS N→G | G→N | BM-MT N→G | G→N | MUTAG N→G | G→N | MNIST1 N→G | G→N | T-Group N→G | G→N |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| GraphPrompt-U | 13.87 | 48.25 | 14.51 | 26.88 | 13.17 | 26.90 | **63.44** | 46.84 | 5.95 | 22.23 | 5.04 | 1.15 |
| All-in-One-U | **78.62** | 73.54 | 9.24 | 32.03 | 11.94 | 24.38 | 28.38 | 9.60 | OOT | OOT | OOT | OOT |
| UniGAD - GCN | 34.05 | **86.00** | **42.41** | **81.74** | **24.26** | **60.50** | 40.63 | **52.67** | **9.22** | 25.59 | **35.02** | **6.97** |
| UniGAD - BWG | 21.25 | 54.09 | 26.62 | 31.35 | 16.86 | 38.62 | 38.64 | 27.56 | 7.84 | **32.94** | 27.15 | 4.30 |

# NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: We confirm that the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope.

   Guidelines:

   - The answer NA means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A No or NA answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: We discuss the limitations of our work in Appendix D.

   Guidelines:

   - The answer NA means that the paper has no limitation while the answer No means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory Assumptions and Proofs**

Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

Answer: [Yes]

Justification: We provide a complete proof of our theoretical results (Theorem 1, Corollary 1, Theorem 2) in Appendix A.1.

Guidelines:

- The answer NA means that the paper does not include theoretical results.

- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.

- All assumptions should be clearly stated or referenced in the statement of any theorems.

- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.

- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.

- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental Result Reproducibility**

Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

Answer: [Yes]

Justification: We provide a link to the code in the abstract and include detailed implementation information in Appendix C to enhance reproducibility.

Guidelines:

- The answer NA means that the paper does not include experiments.

- If the paper includes experiments, a No answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.

- If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.

- Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.

- While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example

    (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.

    (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.

(c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).

(d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

Answer: [Yes]

Justification: We make the code of our model and newly released T-Group dataset open-sourced at `https://anonymous.4open.science/r/UniGAD-A087/`. Other datasets are used only publicly available datasets as stated in Section 4.

Guidelines:

- The answer NA means that paper does not include experiments requiring code.

- Please see the NeurIPS code and data submission guidelines (`https://nips.cc/public/guides/CodeSubmissionPolicy`) for more details.

- While we encourage the release of code and data, we understand that this might not be possible, so "No" is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).

- The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://nips.cc/public/guides/CodeSubmissionPolicy`) for more details.

- The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.

- The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.

- At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).

- Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental Setting/Details**

Question: Does the paper specify all the training and test details (e.g., data splits, hyper-parameters, how they were chosen, type of optimizer, etc.) necessary to understand the results?

Answer: [Yes]

Justification: Please refer to Appendix C for the experiment implementation details.

Guidelines:

- The answer NA means that the paper does not include experiments.

- The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.

- The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment Statistical Significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [Yes]

   Justification: All of our experimental results come from the mean of 5 randomized trials.

   Guidelines:

   - The answer NA means that the paper does not include experiments.

   - The authors should answer "Yes" if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.

   - The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).

   - The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)

   - The assumptions made should be given (e.g., Normally distributed errors).

   - It should be clear whether the error bar is the standard deviation or the standard error of the mean.

   - It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.

   - For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g. negative error rates).

   - If error bars are reported in tables or plots, The authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments Compute Resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: Please refer to Appendix C for the computation resources.

   Guidelines:

   - The answer NA means that the paper does not include experiments.

   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.

   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.

   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code Of Ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics https://neurips.cc/public/EthicsGuidelines?

Answer: [Yes]

Justification: We confirmed the research conducted in the paper conform with the NeurIPS Code of Ethics.

Guidelines:

- The answer NA means that the authors have not reviewed the NeurIPS Code of Ethics.

- If the authors answer No, they should explain the special circumstances that require a deviation from the Code of Ethics.

- The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader Impacts**

Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

Answer: [Yes]

Justification: Please refer to Appendix D for the relative discussion.

Guidelines:

- The answer NA means that there is no societal impact of the work performed.

- If the authors answer NA or No, they should explain why their work has no societal impact or why the paper does not address societal impact.

- Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.

- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.

- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.

- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pretrained language models, image generators, or scraped datasets)?

Answer: [NA]

Justification: The paper poses no such risks.

Guidelines:

- The answer NA means that the paper poses no such risks.

- Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring

that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.

- Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.

- We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

Answer: [Yes]

Justification: We properly credit all referenced baseline works and datasets in Section 4 and Appendix C.

Guidelines:

- The answer NA means that the paper does not use existing assets.

- The authors should cite the original paper that produced the code package or dataset.

- The authors should state which version of the asset is used and, if possible, include a URL.

- The name of the license (e.g., CC-BY 4.0) should be included for each asset.

- For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.

- If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, paperswithcode.com/datasets has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.

- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.

- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New Assets**

Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

Answer: [Yes]

Justification: We provide the code via the anonymous link in the abstract, which will be open-sourced under the MIT license. Comprehensive documentation is included with the assets to ensure ease of use and understanding.

Guidelines:

- The answer NA means that the paper does not release new assets.

- Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.

- The paper should discuss whether and how consent was obtained from people whose asset is used.

- At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and Research with Human Subjects**

Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

Answer: [NA]

Justification: The paper does not involve crowdsourcing nor research with human subjects.

Guidelines:

- The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
- Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
- According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional Review Board (IRB) Approvals or Equivalent for Research with Human Subjects**

Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

Answer: [NA]

Justification: The paper does not involve crowdsourcing nor research with human subjects.

Guidelines:

- The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
- Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
- We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
- For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.