FRACAUG: FRACTIONAL AUGMENTATION BOOST GRAPH-LEVEL ANOMALY DETECTION UNDER LIMITED SUPERVISION

Anonymous authors
Paper under double-blind review

ABSTRACT

Graph-level anomaly detection (GAD) is critical in diverse domains such as drug discovery, yet high labeling costs and dataset imbalance hamper the performance of Graph Neural Networks (GNNs). To address these issues, we propose FracAug, an innovative plug-in augmentation framework that enhances GNNs by generating semantically consistent graph variants and pseudo-labeling with mutual verification. Unlike previous heuristic methods, FracAug learns semantics within given graphs and synthesizes fractional variants, guided by a novel weighted distance-aware margin loss. This captures multi-scale topology to generate diverse, semantic-preserving graphs unaffected by data imbalance. Then, FracAug utilizes predictions from both original and augmented graphs to pseudo-label unlabeled data, iteratively expanding the training set. As a model-agnostic module compatible with various GNNs, FracAug demonstrates remarkable universality and efficacy: experiments across 14 GNNs on 12 real-world datasets show consistent gains, boosting average AUROC, AUPRC, and F1-score by up to 5.72%, 7.23%, and 4.18%, respectively.

1 INTRODUCTION

Graph-structured data is pivotal in real applications ranging from drug discovery to anomaly identification among proteins (Zhang et al., 2022). While Graph Neural Networks (GNNs) excel at modeling topological and feature-based patterns through message-passing, their effectiveness in Graph-level Anomaly Detection (GAD)—distinguishing anomalous graphs from normal ones—is hindered by two key challenges: limited supervision and extreme class imbalance, as demonstrated in Section 5. Specifically, anomalies represent rare instances, exacerbating data imbalance and restricting the availability of labeled training samples (Chen et al., 2024; Dong et al., 2024). While data augmentation techniques have revolutionized computer vision (Zhang et al., 2023) by generating synthetic labels through rotations or crops, their adaptation to graph domains presents unique challenges. Unlike images, graphs inhabit non-Euclidean space where seemingly minor structural modifications (e.g., edge removal) risk distorting semantic properties and violating the label-invariant assumption—a critical constraint in GAD's challenging setting of limited supervision and inherent class imbalance.

Existing graph-level augmentation methods, such as MAA (Yoo et al., 2022), often employ heuristic modifications without considering data properties, leading to compromised semantics or insufficient diversity in GAD tasks. Consequently, their direct application may underperform vanilla GAD models. We attribute this gap to three key issues: (1) the absence of semantic-preserving augmentation strategies, (2) inadequate handling of imbalance, and (3) ineffective utilization of unlabeled data.

To address these challenges, we introduce FracAug, a novel plug-in augmentation framework that generates semantic-preserving graph variants and pseudo-labels for unlabeled graphs to train GNNs for GAD. Our key innovation leverages the fractional power of adjacency matrices, which encodes multi-scale topological relationships. By computing polynomials of various fractional graphs, guided by weighted distance-aware margin loss, FracAug introduces controlled structural variations while ensuring semantic consistency with the original graph's label, independent of the underlying data distribution. Afterward, a given GNN will produce predictions for both original and synthetic samples, enabling FracAug to employ a mutual verification mechanism for pseudo-labeling unlabeled graphs,

thereby iteratively expanding the training set. This approach not only mitigates supervision scarcity but also enhances model robustness against class imbalance.

In summary, our contributions are as follows:

- We present FracAug, the first augmentation framework designed for GAD that maintains effectiveness under the dual constraints of limited supervision and imbalanced distribution
- FracAug operates as a model-agnostic plug-in augmentation framework compatible with 14 GNNs without architectural modifications, facilitating seamless integration into existing models.
- Extensive experiments on 12 real-world datasets demonstrate that FracAug enhances performance across diverse GNNs, significantly outperforming existing graph augmentation approaches.

2 Related Work

Graph Classification. Generalized GNNs, such as GCN (Kipf & Welling, 2017), GraphSAGE (Hamilton et al., 2017), GAT (Velickovic et al., 2018), and GIN (Xu et al., 2019), excel at learning graph representations through neighborhood aggregation. Recent advances include LRGNN (Wei et al., 2023), which captures long-range dependencies with stacking GNNs, and GRDL (Wang & Fan, 2024), which achieves state-of-the-art (SOTA) performance by learning representation distributions of graphs. However, these representative GNNs are not specifically designed for GAD tasks. While they can capture certain topological or feature-based patterns, their performance degrades under data imbalance and limited supervision.

Graph-level Anomaly Detection. Recognizing the challenges underlying GAD tasks, researchers have introduced specialized approaches to address them. For instance, iGAD (Zhang et al., 2022) introduces dual-discriminative kernels guided by a point mutual information-based loss function to better capture graph anomalies. Later, by mapping anomalies and normal graphs to separate areas based on adjusted candidate nodes, GmapAD (Ma et al., 2023a) shows advanced performance. Moreover, RQGNN (Dong et al., 2024) leverages the Rayleigh Quotient to detect graph anomalies effectively within spectral space. Recently, UniGAD (Lin et al., 2024) combines different levels of graph anomaly detection to capture comprehensive information to enhance the detection accuracy. Although these specialized frameworks show promising performance in addressing data imbalance challenges, they still present inferior results due to the limited supervision issue.

Graph-level Augmentation. To address the scarcity of labeled examples, researchers also develop diverse augmentation techniques for graph-level tasks. For example, MAA (Yoo et al., 2022) proposes two separate methods, NodeSam and SubMix, to generate synthetic samples by heuristic structure modification. Besides, GLA (Yue et al., 2022) generates the latent representations as the augmented graphs during the training phase. Subsequently, GMixup (Han et al., 2022) and FGWMixup (Ma et al., 2023b) interpolate graphs or features linearly to mix normal and anomalous samples for producing novel samples. Nevertheless, they fail to produce semantic-preserving samples when dealing with imbalanced data with limited supervision, resulting in unsatisfactory performance.

In contrast, FracAug diverges by leveraging the fractional power of adjacency matrices, a mathematically grounded operation that preserves semantics within graphs while introducing multi-scale structural variations. The incorporation of weighted distance-aware margin loss further enables FracAug to adapt to the imbalanced scenario. Furthermore, its pseudo-labeling mechanism explicitly addresses the limited supervision constraint. As a plug-in module, FracAug overcomes above limitations in GNNs and graph-level augmentation methods without modifying GNN architectures, enabling given GNN models to learn discriminative features for GAD tasks even with sparse labels.

3 Preliminaries

Notation. Let $G = (\boldsymbol{A}, \boldsymbol{X})$ denote an undirected graph with n nodes and m edges, where $\boldsymbol{A} \in \mathbb{R}^{n \times n}$ is the adjacency matrix and $\boldsymbol{X} \in \mathbb{R}^{n \times F}$ is the node feature matrix. $\boldsymbol{A}_{ij} = 1$ if an edge exists between node i and j, and $\boldsymbol{A}_{ij} = 0$ otherwise. $\boldsymbol{D}$ is the diagonal degree matrix of $\boldsymbol{A}$, and the normalized adjacency matrix can be defined as $\tilde{\boldsymbol{A}} = \boldsymbol{D}^{-\frac{1}{2}}\boldsymbol{A}\boldsymbol{D}^{-\frac{1}{2}}$ correspondingly. For a given matrix $\boldsymbol{M}$, $\boldsymbol{M}^{\alpha}$ stands for the α-th power of matrix $\boldsymbol{M}$, where $\alpha \geq 0$. When $\boldsymbol{M}$ can be eigendecomposed,

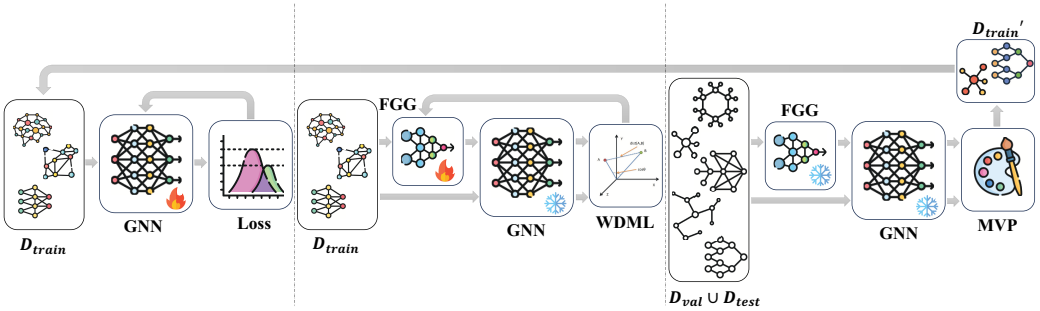


Figure 1: Overview of FracAug.

$\boldsymbol{M}^{\alpha} = \boldsymbol{U}\boldsymbol{\Lambda}^{\alpha}\boldsymbol{V}$, where $\boldsymbol{U}, \boldsymbol{V} \in \mathbb{C}^{n\times n}$ are unitary matrices and $\boldsymbol{\Lambda} \in \mathbb{R}^{n\times n}$ is a diagonal matrix composed of the eigenvalues of $\boldsymbol{M}$.

Continuous Semantic Space. Given a graph G with adjacency matrix $\boldsymbol{A}$ and a graph signal $\boldsymbol{x} \in \mathbb{R}^F$, the semantic space of G is defined as a subspace $\mathcal{S} \subseteq \mathbb{R}^F$ generated by the set of vectors obtained through the application of powers of $\boldsymbol{A}$ to $\boldsymbol{x}$. Unlike previous approaches that rely on discrete semantics constrained by integer powers, i.e., $\{\boldsymbol{A}^t\boldsymbol{x}|t \in \mathbb{N}\}$, our continuous semantic space formulation, $\mathcal{S} = span\{\boldsymbol{A}^t\boldsymbol{x}|t \in \mathbb{N}\}$, captures the underlying continuous semantic manifold of the graph, which enables us to synthesize novel semantic-preserving graph instances, shown in Section 4.

Graph-level Anomaly Detection. In this work, we focus on enhancing GAD performance under limited supervision. Given a training set with k labeled samples, $\mathcal{D}_{train} = \{(G_1, y_1), (G_2, y_2), ..., (G_k, y_k)\}$, the goal of GAD is to train a model that classifies unseen graphs as normal or anomalous. In real deployment, there are two main challenges in GAD. Firstly, $\mathcal{D}_{all} = \mathcal{D}_{train} \cup \mathcal{D}_{val} \cup \mathcal{D}_{test}$ contains N_0 normal graphs and N_1 anomalous graphs, where $N_0 \gg N_1$, leading to severe imbalanced problem. Secondly, only limited labeled graphs are accessible during training, i.e., $k \ll N_0 + N_1$, resulting in the limited supervision issue. Therefore, the key to enhancing the ability of GNNs on real-world GAD tasks is to address these two challenges simultaneously.

Graph-level Augmentation. Graph-level augmentation has been proven effective in improving the performance of GNNs on graph-level tasks. Graph generation and pseudo-labeling are the most common ways to conduct graph-level augmentation:

- Graph generation: This strategy maps the graph $G \in \mathcal{D}_{train}$ to a new graph G', i.e., $(G, y) \mapsto (G', y)$. The generated graph should have a semantic meaning similar to that of the original G.
- Pseudo-labeling: This approach leverages a GNN trained on $\mathcal{D}_{train}$ to classify samples from $\mathcal{D}_{val} \cup \mathcal{D}_{test}$ and assign pseudo-labels to samples with high confidence under a certain criterion.

By combining graph generation and pseudo-labeling techniques while tackling the imbalanced issue, FracAug effectively boosts GNN performance for GAD under limited supervision.

4 METHOD

4.1 OVERVIEW

Our proposed FracAug consists of three key components: (1) **Fractional Graph Generator (FGG)** in Section 4.2 captures the inherent semantics of graphs, enabling the synthesis of fractional variants that maintain semantic consistency with originals, as we demonstrate theoretically. (2) **Weighted Distance-Aware Margin Loss (WDML)** in Section 4.3 addresses data imbalance to guide FGG, employing distance-based margins to position synthetic graphs near original counterparts while ensuring distinctiveness. (3) **Mutual Verification Pseudo-Labeler (MVP)** in Section 4.4 minimizes pseudo-labeling errors through mutual verification of predictions from original and synthetic graphs, facilitating reliable and iterative training set expansion.

Figure 1 illustrates the pipeline of our FracAug. Initially, we warm up a given GNN to establish a preliminary semantic understanding of the GAD task. Then, we freeze the GNN parameters and utilize its outputs to train the FGG with WDML. The trained FGG then generates fractional graph variants, and the GNN predicts on both original and synthetic graphs to pseudo-label data within the validation and test sets using MVP, which are subsequently incorporated into the original training set.

Finally, we train the GNN using the new training set and continue the above process until both the GNN and our FracAug framework reach reasonable capability.

4.2 Fractional Graph Generator

Flexible Eigengraph Combinations. The fractional power of the adjacency matrix, $\boldsymbol{A}^\alpha$, where $\alpha \geq 0$, serves as the mathematical foundation of our framework due to its unique properties. Unlike integer powers of $\boldsymbol{A}$, which only capture discrete-step neighborhood aggregations, fractional powers enable continuous interpolation of graph structures, providing fine-grained control over topological variations. Crucially, $\boldsymbol{A}^\alpha$ can be expressed as a combination of eigengraphs derived from eigendecompositions. For an undirected graph G with symmetric adjacency matrix $\boldsymbol{A}$, we can decompose $\boldsymbol{A}^\alpha$ as:

$$\boldsymbol{A}^\alpha = \boldsymbol{U}\boldsymbol{\Lambda}^\alpha \boldsymbol{U}^T = \sum_{i=1}^{n} \lambda_i^\alpha \boldsymbol{u}_i \boldsymbol{u}_i^T,$$

where $\boldsymbol{\Lambda}$ is the diagonal eigenvalue matrix containing $\{\lambda_i\}_{i=1}^n$ in a descending order, $\boldsymbol{U}$ is the eigenvector matrix formed by $\{\boldsymbol{u}_i\}_{i=1}^n$, and $\boldsymbol{u}_i\boldsymbol{u}_i^T$ is the i-th eigengraph. It reveals two key advantages:

- Multi-scale Structure Adaptation: Fractional powers enable tunable control over spectral components via α, where lower values $(\alpha < 1)$ emphasize homophilic graph signals (low-frequency eigengraphs), while higher values $(\alpha > 1)$ accentuate heterophilic graph signals (high-frequency eigengraphs) (Yan et al., 2023). This adaptive reweighting preserves the hierarchical topology while generating augmented graphs, signaling structural anomalies for detection.
- Semantic-preserving Combination: By combining eigengraphs, FracAug preserves semantic-critical structures (targeting spectral deviations linked to anomalies (Dong et al., 2024)), ensuring that generated graphs retain the original semantics.

Semantic Preservation. Prior studies, such as GIN (Xu et al., 2019), rely on integer powers of adjacency matrices, limiting them to discrete semantic preservation. In contrast, we prove that for any $\alpha \geq 0$, $\boldsymbol{A}^\alpha \boldsymbol{x}$ resides in the original semantic space. Moreover, we further derive a theoretical boundary to quantify differences between the original and fractional graphs, detailed in Appendix A.

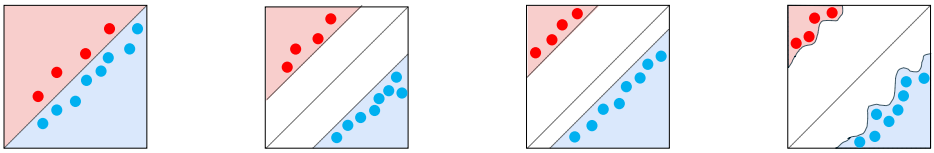
Theorem 1. *Given a polynomial function $p(\cdot;\boldsymbol{\theta})$ parameterized by $\boldsymbol{\theta}$, for any $\alpha \geq 0$, there exists $\boldsymbol{\theta}^*$ such that $\boldsymbol{A}^\alpha \approx p(\boldsymbol{A};\boldsymbol{\theta}^*) = \sum_{t=0}^{T} \boldsymbol{\theta}_t^* \boldsymbol{A}^t, T \in \mathbb{N}$. With proper parameter $\boldsymbol{\theta}^*$, the difference of them is bounded by $\beta e^{-\gamma T}$, where $\beta, \gamma > 0$ depend on the eigenvalues of $\boldsymbol{A}$. Since $\boldsymbol{A}^\alpha$ can be represented as a polynomial combination of $\{\boldsymbol{A}^t\}_{t\in\mathbb{N}}$, $\boldsymbol{A}^\alpha\boldsymbol{x}$ lies in $\mathcal{S}$ of the original graph as $T \to \infty$.*

Theorem 1 ensures that fractional graphs preserve the original semantic space while encoding multi-scale semantics. The continuous parameter α spans all possible semantic variations within this space, enabling rich and comprehensive augmentation. Besides, previous approaches such as MAA (Yoo et al., 2022) leverage heuristic perturbation techniques for generating synthetic graphs, which may result in useful substitutes near the semantic space of the original graph. Theorem 2 formally bridges these perturbation-based approaches with our fractional graph augmentation, revealing their shared theoretical foundations. The complete proof is provided in Appendix A.

Theorem 2. *Let the structural perturbation on a graph be a perturbation matrix $\boldsymbol{P}$ added to the original graph, so that any graph generated by a structural perturbation method can be expressed as $\boldsymbol{A} + \boldsymbol{P}$. Then, we can derive $||\boldsymbol{A}^\alpha - (\boldsymbol{A} + \boldsymbol{P})|| \leq c||\boldsymbol{P}|| + \max_i |\lambda_i - \lambda_i^\alpha|$, where c depends on α and the spectral gap of $\boldsymbol{P}$, and λ_i is the i-th eigenvalue of $\boldsymbol{A} + \boldsymbol{P}$. Thus, by choosing an appropriate α, $\boldsymbol{A}^\alpha$ can approximate any graph generated by structural perturbation methods.*

Based on Theorem 2, we observe that for a suitably chosen α, the fractional graph can approximate any sample generated by perturbation-based framework, demonstrating its generalization capability.

Fractional Graph Generation. Building on the above analysis, we conclude that fractional graphs offer powerful augmentation capabilities. However, directly deriving the fractional power of the adjacency matrix can be computationally prohibitive and may yield invalid results for non-semi-definite adjacency matrix. Thus, a transformation function $h(\cdot)$ is applied to the adjacency matrix to ensure valid fractional powers while preserving structural integrity (Yan et al., 2023). Specifically, instead of adding self-loops before normalization of the adjacency matrix, we introduce them after



(a) Softmax (b) LMCL (c) LDAM (d) WDML

Figure 2: Decision boundaries of different margin losses.

normalization and rescale the matrix, so the resulting adjacency matrix can be defined as:

$$\hat{\boldsymbol{A}} = h(\tilde{\boldsymbol{A}}) = \frac{1}{2}(\boldsymbol{I} + \tilde{\boldsymbol{A}}),$$

which is still a normalized adjacency matrix. Since all eigenvalues of $\tilde{\boldsymbol{A}}$ lie within $[-1, 1]$, the corresponding eigenvalues of $\hat{\boldsymbol{A}}$ fall within $[0, 1]$. Therefore, $h(\cdot)$ transforms $\boldsymbol{A}$ into a positive semi-definite matrix $\hat{\boldsymbol{A}}$, which allows the design of FGG.

Moreover, to mitigate computational costs for large graphs, we precompute the eigendecomposition (EVD) using the Arnoldi method (Lehoucq et al., 1998), retaining only the top-k_l largest and top-k_s smallest eigenpairs. Denote $\hat{\boldsymbol{\Lambda}}_{k_l}, \hat{\boldsymbol{\Lambda}}_{k_s}$ as diagonal matrices of the top-k_l largest and top-k_s smallest eigenvalues, $\boldsymbol{U}_{k_l} = \boldsymbol{U}[:, 0:k_l], \boldsymbol{U}_{k_s} = \boldsymbol{U}[:, n-k_s:n]$ as the corresponding matrices of eigenvectors, and the generated graph of $G(\boldsymbol{A}, \boldsymbol{X})$ as $G'(\boldsymbol{A}', \boldsymbol{X})$, FGG can be formulated as:

$$g(\boldsymbol{A}, k, H) = \sum_{h=1}^{H} \omega_h \boldsymbol{U}_k \hat{\boldsymbol{\Lambda}}_k^{\alpha_h} \boldsymbol{U}_k^T,$$

$$\boldsymbol{A}' = \text{FGG}(\boldsymbol{A}, k_l, k_s, H_l, H_s) = \omega g(\boldsymbol{A}, k_l, H_l) + (1-\omega) g(\boldsymbol{A}, k_s, H_s),$$

where $\sum_{h=1}^{H} \omega_h = 1$, ω are learnable coefficients, and α_h is h-th learnable fractional power of the matrix. By combining multiple fractional graphs with tunable weights, our generated graphs can capture comprehensive information while preserving semantics. Although the anomalous properties are well-preserved in graphs from FGG based on previous analysis, inherent data imbalance risks biasing FGG training. To counteract this, in Section 4.3, we design WDML to guide FGG training.

4.3 Weighted Distance-Aware Margin Loss

Revisiting Margin Loss. To better separate the semantic spaces of different graphs and enable FGG to generate high-quality fractional graphs robust to class imbalance, we introduce a novel margin loss function. Before illustrating the details of WDML, we first reexamine representative margin losses, with comprehensive empirical validation provided in Appendix H. Formally, margin loss based on cross-entropy can be defined as:

$$L = -\frac{1}{N} \sum_{i=1}^{N} \log \frac{e^{\boldsymbol{s}_{\boldsymbol{y}_i} - m}}{e^{\boldsymbol{s}_{\boldsymbol{y}_i} - m} + \sum_{j=1, j \neq \boldsymbol{y}_i}^{C} e^{\boldsymbol{s}_j}}, \tag{1}$$

where N is the number of the samples, $\boldsymbol{s}$ represents the normalized logits predicted by a given GNN, $\boldsymbol{y}_i$ is the ground truth label of i-th sample, m is the margin that determines the decision boundary, and C is the number of classes.

As shown in Figure 2 (a), when setting margin m to 0, the margin loss is degraded to a cross-entropy loss, which lacks explicit mechanisms to separate classes in complex scenarios. Another margin loss is LMCL (Wang et al., 2018) with m as a hyperparameter. Figure 2 (b) describes the result of setting $m > 0$, enforcing better inter-class separation. However, this uniform margin shifts the decision boundaries of different classes by the same value, which fails to detect the unique class-specific properties. Afterward, LDAM (Cao et al., 2019) in Figure 2 (c) tackles the issues by setting class-specific margin m_c for c-th class so that the decision boundaries can accommodate scenarios where classes require distinct margins. Existing margin losses typically employ fixed margins, which prove suboptimal for GAD where sample-specific semantic variations exist. To address this, we propose WDML, which assigns dynamic margins based on the intrinsic distance of each synthetic sample and its original graph with a weight according to its class. Figure 2 (d) describes the adaptive decision boundary of WDML.

Margin Loss Based on Sample-Specific Distance. For the i-th training graph G_i and its counterpart G_i' generated by FGG, we extract graph-level embeddings $\boldsymbol{o}_i$ and $\boldsymbol{o}_i'$ via a given GNN. Then, our distance-aware margin can be defined as:

$$m_i = \frac{1 - \cos(\boldsymbol{o}_i, \boldsymbol{o}_i')}{2}, \tag{2}$$

where cos represents cosine similarity. Substituting m in Equation 1 with the sample-specific margin m_i yields a distance-aware margin loss. By computing angular distances in Equation 2, this loss shifts the semantic space away from decision boundaries by a margin m_i, ensuring generated samples retain the original label with high confidence. To further address class imbalance, WDML incorporates weights based on class frequency:

$$L_{\text{WDML}} = -\sum_{i=1}^{N} \frac{1}{N_{\boldsymbol{y}_i}} \log \frac{e^{\boldsymbol{s}_{\boldsymbol{y}_i} - m_i}}{e^{\boldsymbol{s}_{\boldsymbol{y}_i} - m_i} + \sum_{j=1, j\neq \boldsymbol{y}_i}^{C} e^{\boldsymbol{s}_j}},$$

where $N_{\boldsymbol{y}_i}$ is the number of samples in class $\boldsymbol{y}_i$. With the assistance of WDML, FGG can generate fractional graphs effectively without being biased by the imbalanced distribution of labels. To further boost the performance by data augmentation, we design MVP to combine graph generation and pseudo-labeling techniques, whose details will be elaborated in Section 4.4.

4.4 Mutual Verification Pseudo-Labeler

Insight on Mutual Verification. Prior pseudo-labeling methods for related tasks, such as ConsisGAD (Chen et al., 2024), only rely on confidences from original samples, prone to high errors under low supervision (Dong et al., 2025). Therefore, we first investigate how mutual verification mitigates the error rates compared to single-view methods, theoretically. The proof is detailed in the Appendix A.

Proposition 1. *For a given GNN, assume its prediction error rates for original graphs and corresponding fractional graphs are both δ. The correlation coefficient between the errors is denoted as ρ. Then, when mutual verification is used, compared to single-view methods, the reduction factor of the error rate and its variance can be up to $\delta + \rho\delta(1-\delta)$ and ρ, respectively.*

Proposition 1 demonstrates that the mutual verification mechanism leverages semantic consistency between original graphs and their fractional counterparts to enhance pseudo-labeling reliability. Building on this, we design MVP based on the agreement between predicted labels for the original and synthetic samples, as detailed below.

High-Quality Pseudo-Label Prediction. Based on the above analysis, MVP assigns a pseudo-label $\hat{y}_i$ to the i-th sample in the validation or test set if and only if:

$$\hat{y}_i = \begin{cases} 0, & p_i \leq \tau_n \wedge p_i' \leq \tau_n, \\ 1, & p_i \geq \tau_a \wedge p_i' \geq \tau_a, \end{cases}$$

where p_i, p_i' represent the anomaly probabilities of the i-th original sample and its fractional counterpart, respectively, and τ_n, τ_a denote the confidence thresholds of a sample being normal/anomalous. For any given GNN, we iteratively incorporate high-confidence pseudo-labeled samples from the validation and test sets into the training set, further mitigating the limited supervision issue.

The core innovation of our mutual verification framework lies in leveraging the semantic consistency between original and fractional graphs to generate high-confidence pseudo-labels. This mechanism addresses the scarcity of labeled anomalies by iteratively expanding the training set with reliable samples, guided by theoretical guarantees of robustness.

In summary, our proposed FracAug combines FGG, WDML, and MVP to generate fractional graphs and pseudo-label samples to boost the performance of GNNs on GAD tasks under limited supervision. The experiments in Section 5 further validate our theoretical analysis in Section 4.

5 Experiments

5.1 Experimental Setup

Datasets. We evaluate FracAug on 12 real-world datasets, including MCF-7, MOLT-4, PC-3, SW-620, NCI-H23, OVCAR-8, P388, SF-295, SN12C, UACC257, PROTEINS_full and DBLP_v1.

Table 1: Average AUROC, AUPRC, and F1-score on 12 datasets with multiple runs, using graph classification models as baselines, where the white columns represent vanilla models and the gray ones represent models augmented by FracAug.

Datasets	Metrics	GCN	+FA	SAGE	+FA	GAT	+FA	GIN	+FA	LRGNN	+FA	GRDL	+FA
MCF-7	AUROC	0.5753	**0.5840**	0.5801	**0.5930**	0.5885	**0.6058**	0.5867	**0.5976**	0.5467	**0.6117**	0.5867	**0.6197**
	AUPRC	0.3035	**0.3073**	0.3316	**0.3340**	0.3678	**0.3814**	0.2830	**0.2971**	0.2951	**0.3309**	0.3038	**0.3469**
	F1-score	0.4982	**0.5074**	0.4798	**0.4963**	0.4573	**0.4687**	0.5366	**0.5421**	0.4660	**0.5290**	0.5147	**0.5254**
MOLT-4	AUROC	0.5531	**0.5650**	0.5326	**0.5727**	0.5403	**0.5721**	0.5733	**0.5854**	0.5495	**0.5851**	0.5858	**0.5924**
	AUPRC	0.3183	**0.3217**	0.1907	**0.2602**	0.3456	**0.3667**	0.2830	**0.3001**	0.3047	**0.3175**	0.3018	**0.3101**
	F1-score	0.4501	**0.4626**	0.5166	**0.5277**	0.4096	**0.4313**	0.5072	**0.5103**	0.4570	**0.4939**	0.5091	**0.5117**
PC-3	AUROC	0.5697	**0.5863**	0.5986	**0.6119**	0.5707	**0.5865**	0.5969	**0.6119**	0.5690	**0.6102**	0.6044	**0.6202**
	AUPRC	0.2740	**0.2837**	0.3154	**0.3248**	0.3562	**0.3626**	0.2797	**0.2893**	0.2320	**0.3038**	0.3381	**0.3459**
	F1-score	0.4745	**0.4876**	0.4751	**0.4841**	0.4036	**0.4166**	0.5063	**0.5205**	**0.5103**	0.5024	0.4615	**0.4750**
SW-620	AUROC	0.5662	**0.5839**	0.5800	**0.5968**	0.5633	**0.5870**	0.5938	**0.6004**	0.5758	**0.5946**	0.6005	**0.6046**
	AUPRC	0.3134	**0.3229**	**0.3401**	0.3260	0.2481	**0.2587**	0.2776	**0.2813**	**0.3506**	0.3386	**0.2908**	0.2901
	F1-score	0.4406	**0.4541**	0.4339	**0.4678**	0.4923	**0.5187**	0.5090	**0.5155**	0.4190	**0.4536**	0.5059	**0.5135**
NCI-H23	AUROC	0.5811	**0.5864**	0.5765	**0.6105**	0.5777	**0.6084**	0.5897	**0.5968**	0.6002	**0.6315**	0.6161	**0.6271**
	AUPRC	0.2777	**0.2777**	0.3197	**0.3207**	**0.2966**	0.2945	0.2566	**0.2659**	0.2830	**0.3205**	0.3034	**0.3069**
	F1-score	0.4751	**0.4819**	0.4333	**0.4740**	0.4548	**0.4961**	0.5059	**0.5073**	0.4987	**0.5055**	0.4983	**0.5112**
OVCAR-8	AUROC	0.5692	**0.5809**	0.5763	**0.5836**	0.5396	**0.5784**	0.5935	**0.5963**	0.5628	**0.5984**	0.6230	**0.6311**
	AUPRC	0.3240	**0.3263**	0.3391	**0.3423**	0.2010	**0.2401**	0.2573	**0.2612**	0.3106	**0.3290**	0.3042	**0.3129**
	F1-score	0.4216	**0.4334**	0.4163	**0.4221**	0.4855	**0.5060**	0.5118	**0.5123**	0.4260	**0.4518**	0.5087	**0.5119**
P388	AUROC	0.5171	**0.5896**	0.5820	**0.6277**	0.4964	**0.5758**	0.5565	**0.5913**	0.5546	**0.6316**	0.5500	**0.5852**
	AUPRC	0.3488	**0.3926**	**0.3569**	0.3540	0.2045	**0.2134**	0.2850	**0.3309**	0.2880	**0.2953**	0.2318	**0.2859**
	F1-score	0.3428	**0.3886**	0.4138	**0.4741**	0.4478	**0.5481**	0.4468	**0.4491**	0.4430	**0.5496**	0.4808	**0.4814**
SF-295	AUROC	0.5730	**0.5813**	0.5858	**0.6057**	0.5960	**0.6171**	0.5844	**0.6076**	0.5903	**0.6185**	0.6156	**0.6349**
	AUPRC	0.3290	**0.3308**	0.3161	**0.3222**	0.2652	**0.2708**	0.2766	**0.2832**	**0.3000**	0.2972	0.2796	**0.3115**
	F1-score	0.4199	**0.4279**	0.4463	**0.4652**	0.5065	**0.5389**	0.4803	**0.5047**	0.4669	**0.5068**	**0.5221**	0.5173
SN12C	AUROC	0.5624	**0.5818**	0.5705	**0.6030**	0.5863	**0.6020**	0.5995	**0.6079**	0.5973	**0.6104**	0.6061	**0.6211**
	AUPRC	0.2812	**0.2981**	0.2859	**0.3133**	0.2468	**0.2585**	0.2696	**0.2746**	0.2729	**0.2888**	0.2803	**0.2875**
	F1-score	0.4463	**0.4546**	0.4514	**0.4670**	0.5058	**0.5191**	0.5030	**0.5110**	0.4978	**0.5012**	0.5026	**0.5183**
UACC257	AUROC	0.5660	**0.5831**	0.6006	**0.6132**	0.5890	**0.6174**	0.5877	**0.6015**	0.6020	**0.6189**	0.6155	**0.6340**
	AUPRC	0.3334	**0.3509**	**0.3360**	0.3337	**0.3493**	0.3389	0.2480	**0.2598**	0.3047	**0.3215**	0.2942	**0.3051**
	F1-score	0.3921	**0.3954**	0.4289	**0.4456**	0.4031	**0.4455**	0.4906	**0.4983**	0.4585	**0.4631**	0.4843	**0.4990**
PROTEINS_full	AUROC	0.6186	**0.6259**	0.5942	**0.6310**	0.6157	**0.6836**	0.5799	**0.6174**	0.6434	**0.6503**	0.5895	**0.5987**
	AUPRC	0.6325	**0.6404**	0.6086	**0.6516**	0.6350	**0.7005**	0.6259	**0.6358**	0.6603	**0.6722**	0.6015	**0.6077**
	F1-score	0.6199	**0.6273**	0.5909	**0.6289**	0.6158	**0.6859**	0.5679	**0.6175**	0.6431	**0.6469**	0.5856	**0.5962**
DBLP_v1	AUROC	0.7866	**0.7973**	0.6218	**0.6825**	0.6119	**0.6885**	0.6231	**0.8044**	0.7922	**0.8006**	0.8089	**0.8222**
	AUPRC	0.8462	**0.8515**	0.7133	**0.7769**	0.7507	**0.7796**	0.7201	**0.8626**	0.8485	**0.8537**	0.8671	**0.8716**
	F1-score	0.7854	**0.7974**	0.6161	**0.6805**	0.5782	**0.6868**	0.5996	**0.8028**	0.7919	**0.8007**	0.8071	**0.8220**

These datasets are obtained from TUDataset[1], and their detailed statistics are listed in Appendix B. We randomly divide each dataset into 1%/1%/98% for $\mathcal{D}_{train}$/$\mathcal{D}_{val}$/$\mathcal{D}_{test}$ to simulate the limited supervision scenario in real applications.

Baselines. We integrate our FracAug with 10 distinct GNNs, including generalized graph classification models and specialized GAD models, to demonstrate its broad applicability. Besides, to further confirm the usefulness of FracAug, we compare FracAug against 4 SOTA graph-level augmentation frameworks based on their original vanilla models.

- Graph Classification: GCN (Kipf & Welling, 2017), GraphSAGE (Hamilton et al., 2017), GAT (Velickovic et al., 2018), GIN (Xu et al., 2019), LRGNN (Wei et al., 2023), and GRDL (Wang & Fan, 2024).
- Graph-level Anomaly Detection: iGAD (Zhang et al., 2022), GmapAD (Ma et al., 2023a), RQGNN (Dong et al., 2024), and UniGAD (Lin et al., 2024).
- Graph-level Augmentation: MAA (Yoo et al., 2022), GLA (Yue et al., 2022), GMixup (Han et al., 2022), and FGWMixup (Ma et al., 2023b).

Experimental Settings. To ensure fair evaluation, we standardize evaluations by: (1) sourcing all baseline code from GitHub and replacing loss functions with weighted version to mitigate class imbalance; (2) using authors' recommended hyperparameters for baselines, while optimizing FracAug's hyperparameters via grid search to maximize the summed AUROC/AUPRC/F1-score on validation sets. Complete configurations are detailed in Appendix E.

[1]https://chrsmrrs.github.io/datasets/docs/datasets/

Table 2: Average AUROC, AUPRC, and F1-score on 12 datasets with multiple runs, using GAD models as baselines, where the white columns represent vanilla models and the gray ones represent models augmented by FracAug.

Datasets	Metrics	iGAD	+FA	GmapAD	+FA	RQGNN	+FA	UniGAD	+FA
MCF-7	AUROC	0.5670	**0.5756**	0.5159	**0.5342**	0.5522	**0.5709**	0.5380	**0.5480**
	AUPRC	0.3402	**0.3525**	0.3521	**0.3577**	0.2379	**0.2648**	0.2405	**0.2527**
	F1-score	0.4546	**0.4549**	0.3776	**0.3961**	0.5609	**0.5804**	0.4959	**0.5008**
MOLT-4	AUROC	0.5562	**0.5573**	0.5100	**0.5374**	0.5576	**0.5696**	0.5353	**0.5445**
	AUPRC	0.3088	**0.3092**	0.2704	**0.2827**	0.2302	**0.2468**	0.2111	**0.2241**
	F1-score	0.4621	**0.4636**	0.4354	**0.4592**	0.5644	**0.5718**	0.5079	**0.5125**
PC-3	AUROC	0.5526	**0.5674**	0.5112	**0.5266**	0.5618	**0.6043**	0.5496	**0.5559**
	AUPRC	0.1887	**0.2159**	0.2999	**0.3094**	0.2370	**0.2663**	0.3875	**0.3889**
	F1-score	0.5217	**0.5229**	0.3845	**0.3938**	0.5721	**0.5972**	0.3461	**0.3542**
SW-620	AUROC	0.5641	**0.5776**	0.5279	**0.5362**	0.5428	**0.5692**	0.5427	**0.5688**
	AUPRC	0.3332	**0.3701**	**0.3506**	0.3479	0.1936	**0.2219**	0.2527	**0.2585**
	F1-score	**0.4207**	0.4029	0.3605	**0.3734**	0.5530	**0.5654**	0.4600	**0.4903**
NCI-H23	AUROC	0.5689	**0.5721**	0.5289	**0.5489**	0.5704	**0.6061**	0.5694	**0.5860**
	AUPRC	0.2267	**0.2288**	0.3274	**0.3401**	0.2166	**0.2500**	0.2733	**0.2756**
	F1-score	0.5039	**0.5066**	0.3710	**0.3827**	0.5770	**0.5820**	0.4645	**0.4831**
OVCAR-8	AUROC	0.5609	**0.5685**	0.5209	**0.5243**	0.5549	**0.5773**	0.5360	**0.5445**
	AUPRC	0.2319	**0.2325**	**0.2863**	0.2836	0.1933	**0.2216**	**0.3196**	0.3112
	F1-score	0.4880	**0.4991**	0.3992	**0.4054**	0.5618	**0.5794**	0.3873	**0.4043**
P388	AUROC	0.5143	**0.5300**	0.4782	**0.5057**	0.5952	**0.6108**	0.5104	**0.5167**
	AUPRC	0.1923	**0.1939**	0.2478	**0.2599**	0.2484	**0.2650**	0.1679	**0.1748**
	F1-score	0.4669	**0.4843**	0.3894	**0.4099**	0.5879	**0.5883**	0.4781	**0.4812**
SF-295	AUROC	0.5811	**0.5815**	0.5414	**0.5535**	0.5582	**0.5902**	0.5439	**0.5730**
	AUPRC	0.2666	**0.2821**	0.3066	**0.3070**	0.2141	**0.2342**	**0.3000**	0.2846
	F1-score	**0.4836**	0.4705	0.4030	**0.4167**	0.5719	**0.5847**	0.4117	**0.4582**
SN12C	AUROC	0.5522	**0.5537**	0.5343	**0.5441**	0.5597	**0.6038**	0.5433	**0.5497**
	AUPRC	0.1817	**0.1858**	0.3262	**0.3315**	0.1927	**0.2442**	**0.2028**	0.1961
	F1-score	**0.5151**	0.5144	0.3754	**0.3818**	0.5648	**0.5826**	0.4851	**0.4986**
UACC257	AUROC	0.5697	**0.5748**	0.5394	**0.5597**	0.5528	**0.5692**	0.5710	**0.5832**
	AUPRC	0.1906	**0.1970**	0.2927	**0.3004**	0.1601	**0.1885**	0.2510	**0.2642**
	F1-score	0.5201	**0.5224**	0.3986	**0.4144**	0.5522	**0.5678**	0.4676	**0.4710**
PROTEINS_full	AUROC	0.5976	**0.6206**	0.5041	**0.6289**	0.5641	**0.6365**	0.6173	**0.6212**
	AUPRC	0.6200	**0.6333**	0.5169	**0.6436**	0.5673	**0.6563**	0.6295	**0.6338**
	F1-score	0.5960	**0.6211**	0.5020	**0.6299**	0.5600	**0.6310**	0.6178	**0.6223**
DBLP_v1	AUROC	0.7755	**0.7909**	0.4975	**0.5045**	0.8065	**0.8082**	0.7601	**0.7965**
	AUPRC	0.8377	**0.8473**	0.6242	**0.6548**	0.8584	**0.8598**	0.8346	**0.8509**
	F1-score	0.7749	**0.7910**	0.4968	**0.5021**	0.8060	**0.8079**	0.7549	**0.7966**

5.2 Experimental Results

We evaluate the performance of FracAug on 6 graph classification models and 4 GAD models. Tables 1 and 2 report the AUROC, AUPRC, and F1-score on 12 datasets. Besides, we also compare FracAug with 4 graph-level augmentation methods on their vanilla models, as shown in Table 3. The best performance of each model is highlighted in boldface. To sum up, FracAug effectively boosts the performance of GNNs and outperforms almost all baselines on these real-world datasets. Next, we provide our detailed observations.

Augmentation for Graph Classification Models. We analyze 4 generalized GNNs (GCN, GraphSAGE, GAT, and GIN) and 2 recent models (LRGNN and GRDL) under limited supervision conditions. While generalized GNNs, due to architectural simplicity, struggle to capture nuanced anomaly patterns in GAD tasks, FracAug boosts their performance across most datasets as shown in Table 1, validating its augmentation efficacy. Surprisingly, LRGNN and GRDL initially underperform simpler GNNs in some cases, likely hindered by label scarcity, but regain competitiveness when integrated with FracAug, highlighting FracAug's adaptability to advanced architectures.

Augmentation for Graph-level Anomaly Detection Models. Specialized GAD models (iGAD, GmapAD, RQGNN, and UniGAD) exploit task-specific properties but falter under limited supervision due to insufficient generalization capability. Notably, these task-specific architectures may underperform even basic GNNs in low-label regimes as presented in Table 2, emphasizing FracAug's

Table 3: Average AUROC, AUPRC, and F1-score on 12 datasets with multiple runs, using graph-level augmentation models as baselines, where the white columns represent vanilla models and their own augmentation method, while the gray ones represent vanilla models augmented by FracAug.

Datasets	Metrics	MAAv	NodeSam	SubMix	+FA	GLAv	GLA	+FA	GMixupv	GMixup	+FA	FGWMixupv	FGWMixup	+FA
MCF-7	AUROC	0.5496	**0.5727**	0.5428	0.5695	0.5797	0.5735	**0.6068**	0.5730	0.5581	**0.5935**	0.5731	0.5502	**0.5828**
	AUPRC	0.2346	0.2445	0.2224	**0.2455**	0.2558	0.2456	**0.2944**	0.2767	0.2864	**0.3081**	0.2720	0.2130	**0.2945**
	F1-score	0.5179	0.5635	0.5512	**0.5721**	0.5607	0.5609	**0.5793**	0.5186	0.4879	**0.5220**	0.5585	0.5283	**0.5971**
MOLT-4	AUROC	0.5506	0.5391	0.5113	**0.5663**	0.5585	0.5578	**0.5792**	0.5637	0.5547	**0.5771**	0.5477	0.5308	**0.6067**
	AUPRC	0.2203	0.1914	0.1796	**0.2356**	0.2177	0.2159	**0.2511**	0.2411	0.2176	**0.2559**	0.1994	0.1939	**0.3104**
	F1-score	0.5595	0.5392	0.5048	**0.5658**	0.5540	0.5519	**0.5733**	0.5306	0.5368	**0.5416**	**0.5446**	0.5115	0.5366
PC-3	AUROC	0.5688	0.5805	0.5284	**0.5821**	0.6207	0.5938	**0.6221**	0.5705	0.5646	**0.5782**	0.5490	0.5442	**0.6107**
	AUPRC	0.2112	0.2233	0.1854	**0.2385**	0.2770	0.2386	**0.2790**	0.3506	0.3456	**0.3587**	0.2028	0.2108	**0.2623**
	F1-score	0.5698	0.5685	0.5321	**0.5848**	0.5650	0.5569	**0.5686**	0.4085	0.4058	**0.4101**	0.5027	0.4914	**0.5633**
SW-620	AUROC	0.5577	0.5776	0.5232	**0.5834**	0.5822	0.5799	**0.5936**	0.5839	0.5722	**0.5987**	0.5265	0.5395	**0.6183**
	AUPRC	0.2026	0.2205	0.1958	**0.2279**	0.2315	0.2258	**0.2455**	0.2599	0.2511	**0.2728**	0.1340	0.1777	**0.2759**
	F1-score	0.5641	0.5477	0.5273	**0.5676**	0.5422	0.5473	**0.5786**	0.5111	0.5015	**0.5220**	0.5283	0.5113	**0.5729**
NCI-H23	AUROC	0.5792	0.5634	0.5323	**0.5979**	0.5773	0.5762	**0.6194**	0.5912	0.5647	**0.6014**	0.5658	0.5760	**0.6422**
	AUPRC	0.2273	0.1983	0.2017	**0.2407**	0.2082	0.2188	**0.2680**	0.2587	0.2139	**0.2672**	0.1986	0.2345	**0.3105**
	F1-score	0.5821	0.5625	0.5445	**0.5835**	0.5659	0.5779	**0.5914**	0.5058	0.5087	**0.5134**	**0.5561**	0.5066	0.5358
OVCAR-8	AUROC	0.5507	0.5494	0.5278	**0.5726**	0.5911	0.5859	**0.6049**	0.5786	0.5725	**0.6024**	0.5696	0.5713	**0.6317**
	AUPRC	0.1775	0.1633	0.1665	**0.2132**	0.2346	0.2196	**0.2447**	0.2764	0.2994	**0.3072**	0.2696	0.2401	**0.3060**
	F1-score	0.5461	0.5408	0.5337	**0.5749**	0.5350	0.5548	**0.5728**	0.4733	0.4469	**0.4766**	0.4831	0.4950	**0.5211**
P388	AUROC	0.5500	0.5069	0.5057	**0.5720**	0.5622	0.5816	**0.6057**	0.5469	0.5265	**0.5647**	0.5480	0.5409	**0.5729**
	AUPRC	0.1958	0.1985	0.1127	**0.2229**	0.2157	0.2264	**0.2632**	0.1694	0.1536	**0.1957**	0.2056	0.1951	**0.2362**
	F1-score	0.5520	0.5000	0.4987	**0.5746**	0.5372	0.5766	**0.5925**	0.5315	0.5078	**0.5373**	0.5421	0.5282	**0.5798**
SF-295	AUROC	0.5649	0.5579	0.5292	**0.5753**	0.5954	0.6060	**0.6197**	0.5665	0.5687	**0.6040**	0.5893	0.5981	**0.6459**
	AUPRC	0.2114	0.1939	0.2218	**0.2252**	0.2316	0.2451	**0.2648**	0.2004	0.2061	**0.2509**	0.2331	0.2585	**0.3017**
	F1-score	0.5736	0.5644	0.5406	**0.5820**	0.5643	0.5702	**0.5855**	0.5245	0.5205	**0.5371**	0.5300	0.5161	**0.5623**
SN12C	AUROC	0.5509	0.5639	0.5160	**0.5795**	0.5715	0.6003	**0.6141**	0.5713	0.5336	**0.5984**	0.5831	0.5693	**0.6314**
	AUPRC	0.1726	0.1845	0.1966	**0.2164**	0.2048	0.2344	**0.2528**	0.2163	0.2047	**0.2524**	0.2382	0.2252	**0.2929**
	F1-score	0.5538	0.5405	0.5190	**0.5770**	0.5644	0.5590	**0.5655**	0.5136	0.4920	**0.5195**	0.5085	0.5002	**0.5326**
UACC257	AUROC	0.5623	0.5023	0.5211	**0.5947**	0.6159	0.6198	**0.6327**	0.5853	0.5843	**0.6209**	0.5535	0.5632	**0.6368**
	AUPRC	0.1805	0.0559	0.0942	**0.2210**	0.2755	0.2745	**0.2978**	0.2365	0.2285	**0.2963**	0.1534	0.2172	**0.2718**
	F1-score	0.5541	0.5005	0.5214	**0.5784**	0.5033	**0.5131**	0.5050	0.4993	**0.5042**	0.4898	0.5470	0.4830	**0.5571**
PROTEINS_full	AUROC	0.6009	0.6083	0.4998	**0.6217**	0.5652	0.5325	**0.6249**	0.5411	0.5132	**0.6097**	0.5078	0.5259	**0.6082**
	AUPRC	0.6183	0.6294	0.6186	**0.6366**	0.6053	0.5343	**0.6476**	0.5810	0.5057	**0.6244**	0.5300	**0.6934**	0.6333
	F1-score	0.6015	0.6039	0.4316	**0.6214**	0.5577	0.5291	**0.6227**	0.5348	0.5025	**0.6102**	0.4909	0.3809	**0.6031**
DBLP_v1	AUROC	0.6446	0.6608	0.6205	**0.6822**	0.7040	0.6402	**0.7222**	0.7939	0.7885	**0.7994**	0.7865	0.7772	**0.7989**
	AUPRC	0.7689	0.7868	**0.7947**	0.7816	0.7882	0.7450	**0.8085**	0.8503	0.8471	**0.8563**	0.8461	0.8377	**0.8549**
	F1-score	0.6252	0.6408	0.6291	**0.6778**	0.7029	0.6147	**0.7177**	0.7937	0.7878	**0.7985**	0.7856	0.7772	**0.7981**

effectiveness. Our framework universally elevates their performance by compensating for supervision scarcity, validating its versatility across model paradigms.

Comparison with Graph-level Augmentation Frameworks. To further prove the effectiveness of FracAug, we compare it against leading graph-level augmentation frameworks, including MAA, GLA, GMixup, and FGWMixup. In Table 3, we denote their corresponding vanilla models as MAAv, GLAv, GMixupv, and FGWMixupv, respectively. Such a setting will preserve the ability of those augmentation frameworks. Nevertheless, as we can see, the augmentation methods fail to generalize effectively to GAD tasks under limited supervision—the performance of the vanilla models may drop after the augmentation. In contrast, our FracAug can boost all vanilla models across real-world datasets, which demonstrates the usefulness of FracAug.

Beyond these primary results, we provide extensive additional analyses: complexity analysis, hyper-parameter analysis, ablation study, loss function comparison, performance with more training data, and the learned parameters on different datasets in Appendix D, F, G, H, I, and J, separately. These supplementary analyses can further demonstrate the effectiveness of our proposed FracAug.

6 Conclusion

In this paper, we investigate the efficacy of leveraging fractional graph variants for data augmentation in GAD under limited supervision scenarios. Based on the analysis, we design a model-agnostic plug-in augmentation framework, FracAug, which includes three key components: FGG, WDML, and MVP. FGG with WDML captures semantics from original samples and then generates semantic-preserving fractional graphs during model training, unaffected by the imbalanced data distribution, while MVP employs mutual verification to enhance pseudo-labeling reliability, iteratively expanding the training set. Comprehensive experiments demonstrate that FracAug not only effectively improves the performance of any given GNN but also significantly outperforms other graph-level augmentation methods, demonstrating the effectiveness of our method.

REFERENCES

Kaidi Cao, Colin Wei, Adrien Gaidon, Nikos Aréchiga, and Tengyu Ma. Learning imbalanced datasets with label-distribution-aware margin loss. In *NeurIPS*, pp. 1565–1576, 2019.

Nan Chen, Zemin Liu, Bryan Hooi, Bingsheng He, Rizal Fathony, Jun Hu, and Jia Chen. Consistency training with learnable data augmentation for graph anomaly detection with limited supervision. In *The Twelfth International Conference on Learning Representations, ICLR 2024, Vienna, Austria, May 7-11, 2024*, 2024.

Paul D Dobson and Andrew J Doig. Distinguishing enzyme structures from non-enzymes without alignments. *Journal of molecular biology*, 330:771–783, 2003.

Xiangyu Dong, Xingyi Zhang, and Sibo Wang. Rayleigh quotient graph neural networks for graph-level anomaly detection. In *ICLR*, 2024.

Xiangyu Dong, Xingyi Zhang, Lei Chen, Mingxuan Yuan, and Sibo Wang. Spacegnn: Multi-space graph neural network for node anomaly detection with extremely limited labels. In *ICLR*, 2025.

William L. Hamilton, Zhitao Ying, and Jure Leskovec. Inductive representation learning on large graphs. In *NeurIPS*, pp. 1024–1034, 2017.

Xiaotian Han, Zhimeng Jiang, Ninghao Liu, and Xia Hu. G-mixup: Graph data augmentation for graph classification. In *ICML*, pp. 8230–8248, 2022.

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

Richard B. Lehoucq, Danny C. Sorensen, and Chao Yang. *ARPACK users' guide - solution of large-scale eigenvalue problems with implicitly restarted Arnoldi methods*. SIAM, 1998.

Yiqing Lin, Jianheng Tang, Chenyi Zi, H. Vicky Zhao, Yuan Yao, and Jia Li. Unigad: Unifying multi-level graph anomaly detection. In *NeurIPS*, pp. 136120–136148, 2024.

Xiaoxiao Ma, Jia Wu, Jian Yang, and Quan Z. Sheng. Towards graph-level anomaly detection via deep evolutionary mapping. In *KDD*, pp. 1631–1642, 2023a.

Xinyu Ma, Xu Chu, Yasha Wang, Yang Lin, Junfeng Zhao, Liantao Ma, and Wenwu Zhu. Fused gromov-wasserstein graph mixup for graph-level classifications. In *NeurIPS*, pp. 15252–15276, 2023b.

Christopher Morris, Nils M. Kriege, Franka Bause, Kristian Kersting, Petra Mutzel, and Marion Neumann. Tudataset: A collection of benchmark datasets for learning with graphs. In *ICML 2020 Workshop on Graph Representation Learning and Beyond*, 2020.

Shirui Pan, Xingquan Zhu, Chengqi Zhang, and Philip S. Yu. Graph stream classification using labeled and unlabeled graphs. In *ICDE*, pp. 398–409, 2013.

Lloyd N Trefethen. *Approximation theory and approximation practice, extended edition*. SIAM, 2019.

Petar Velickovic, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. Graph attention networks. In *ICLR*, 2018.

Hao Wang, Yitong Wang, Zheng Zhou, Xing Ji, Dihong Gong, Jingchao Zhou, Zhifeng Li, and Wei Liu. Cosface: Large margin cosine loss for deep face recognition. In *CVPR*, pp. 5265–5274, 2018.

Zixiao Wang and Jicong Fan. Graph classification via reference distribution learning: Theory and practice. In *NeurIPS*, pp. 137698–137740, 2024.

Lanning Wei, Zhiqiang He, Huan Zhao, and Quanming Yao. Search to capture long-range dependency with stacking gnns for graph classification. In *WWW*, pp. 588–598, 2023.

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

Yuchen Yan, Yuzhong Chen, Huiyuan Chen, Minghua Xu, Mahashweta Das, Hao Yang, and Hanghang Tong. From trainable negative depth to edge heterophily in graphs. In *NeurIPS*, pp. 70162–70178, 2023.

Jaemin Yoo, Sooyeon Shim, and U Kang. Model-agnostic augmentation for accurate graph classification. In *WWW*, pp. 1281–1291, 2022.

Han Yue, Chunhui Zhang, Chuxu Zhang, and Hongfu Liu. Label-invariant augmentation for semi-supervised graph classification. In *NeurIPS*, pp. 29350–29361, 2022.

Ge Zhang, Zhenyu Yang, Jia Wu, Jian Yang, Shan Xue, Hao Peng, Jianlin Su, Chuan Zhou, Quan Z. Sheng, Leman Akoglu, and Charu C. Aggarwal. Dual-discriminative graph neural network for imbalanced graph-level anomaly detection. In *NeurIPS*, pp. 24144–24157, 2022.

Lingrui Zhang, Shuheng Zhang, Guoyang Xie, Jiaqi Liu, Hua Yan, Jinbao Wang, Feng Zheng, and Yaochu Jin. What makes a good data augmentation for few-shot unsupervised image anomaly detection? In *CVPR*, pp. 4345–4354, 2023.

A PROOFS

Proof of Theorem 1. To derive the approximation of $\boldsymbol{A}^\alpha$ and the corresponding error bound, we first consider a function for real numbers, i.e., $f(x) = x^\alpha$ defined on an interval $[a, b] \subset (0, +\infty)$. To satisfy the requirement of Chebyshev series approximation, we map $[a, b]$ to the standard Chebyshev interval $[-1, 1]$ via the linear transformation:

$$x = \frac{2}{b-a}(x' - \frac{b+a}{2}),$$

where $x' \in [a, b]$ maps to $x \in [-1, 1]$. Then we further define:

$$\tilde{f}(x) = (\frac{(b-a)x + (b+a)}{2})^\alpha,$$

which can be approximated using Chebyshev series approximation as:

$$\tilde{f}(x) \approx p_T(x) = \sum_{t=0}^{T} c_t P_t(x),$$

where $P_t(x)$ is the t-th Chebyshev polynomial, and the c_t are the Chebyshev coefficients. Specifically, we can find the coefficients c_t through the application of an inner product:

$$\int_{-1}^{+1} \frac{P_m(x)\tilde{f}(x)}{\sqrt{1-x^2}} dx = \sum_{t=0}^{\infty} c_t \int_{-1}^{+1} \frac{P_m(x)P_t(x)}{\sqrt{1-x^2}} dx.$$

On the interval $[-1, 1]$, we have:

$$\int_{-1}^{+1} \frac{P_m(x)P_t(x)}{\sqrt{1-x^2}} dx = \begin{cases} 0, & m \neq t, \\ \pi, & m = t = 0, \\ \frac{\pi}{2}, & m = t \neq 0, \end{cases}$$

so we can derive:

$$c_t = \begin{cases} \frac{1}{\pi} \int_{-1}^{+1} \frac{P_t(x)\tilde{f}(x)}{\sqrt{1-x^2}} dx, & t = 0, \\ \frac{2}{\pi} \int_{-1}^{+1} \frac{P_t(x)\tilde{f}(x)}{\sqrt{1-x^2}} dx, & t \neq 0. \end{cases}$$

Afterward, to obtain the error bound of the approximation, we leverage the following Theorem:

Theorem 3. *(Theorems 8.1 and 8.2 from previous work (Trefethen, 2019)) Let a function $f(x)$ analytic in $[-1, 1]$ be analytically continuable to open Bernstein ellipse $\boldsymbol{E}_p$, where it satisfies $|f(x)| \leq M$ for some M, then for each $t \geq 0$, its Chebyshev approximation $p_T(x)$ satisfies $||f(x) - p_T(x)|| \leq \frac{4M\rho^{-T}}{\rho-1}$, where ρ depends on the distance from $[-1, 1]$ to the nearest singularity of $f(x)$.*

The function $f(x) = x^\alpha$ has a branch point at $x = 0$. For $[a, b] \subset (0, +\infty)$, the mapped function $\tilde{f}(x)$ is analytic in a Bernstein ellipse $\boldsymbol{E}_p$, excluding $x = 0$. Therefore, $\tilde{f}(x)$ satisfies Theorem 3, so we can have:

$$||\tilde{f}(x) - p_T(x)|| \leq \frac{4M\rho^{-T}}{\rho - 1} = \beta e^{-\gamma T},$$

where $\beta = \frac{4M}{\rho-1}$ and $\gamma = \ln \rho$.

Similarly, we can directly apply the function to $\boldsymbol{A}$ with eigenvalues in $[\lambda_{min}, \lambda_{max}] \subset (0, +\infty)$, then we can conclude:

$$||\boldsymbol{A}^\alpha - p_T(\boldsymbol{A})|| \leq \beta e^{-\gamma T},$$

where β, γ is derived from $[\lambda_{min}, \lambda_{max}]$. □

Proof of Theorem 2. Using the Dunford-Taylor integral, for a contour Γ enclosing the spectra of $\boldsymbol{A}$ and $\boldsymbol{A}+\boldsymbol{P}$, we have:

$$\boldsymbol{A}^\alpha = \frac{1}{2\pi i}\int_\Gamma x^\alpha (x\boldsymbol{I}-\boldsymbol{A})^{-1}dx,$$
$$(\boldsymbol{A}+\boldsymbol{P})^\alpha = \frac{1}{2\pi i}\int_\Gamma x^\alpha (x\boldsymbol{I}-(\boldsymbol{A}+\boldsymbol{P}))^{-1}dx.$$

Then we subtract the two integrals:

$$\boldsymbol{A}^\alpha - (\boldsymbol{A}+\boldsymbol{P})^\alpha = \frac{1}{2\pi i}\int_\Gamma x^\alpha [(x\boldsymbol{I}-\boldsymbol{A})^{-1} - (x\boldsymbol{I}-(\boldsymbol{A}+\boldsymbol{P}))^{-1}]dx.$$

After applying the resolvent identity, we can have:

$$(x\boldsymbol{I}-\boldsymbol{A})^{-1} - (x\boldsymbol{I}-(\boldsymbol{A}+\boldsymbol{P}))^{-1} = (x\boldsymbol{I}-\boldsymbol{A})^{-1}\boldsymbol{P}(x\boldsymbol{I}-(\boldsymbol{A}+\boldsymbol{P}))^{-1}.$$

By substituting back into the integral, we have:

$$\boldsymbol{A}^\alpha - (\boldsymbol{A}+\boldsymbol{P})^\alpha = \frac{1}{2\pi i}\int_\Gamma x^\alpha (x\boldsymbol{I}-\boldsymbol{A})^{-1}\boldsymbol{P}(x\boldsymbol{I}-(\boldsymbol{A}+\boldsymbol{P}))^{-1}dx.$$

Take the operator norm and apply submultiplicativity:

$$||\boldsymbol{A}^\alpha - (\boldsymbol{A}+\boldsymbol{P})^\alpha|| \le \frac{1}{2\pi}\int_\Gamma |x^\alpha|||(x\boldsymbol{I}-\boldsymbol{A})^{-1}||||\boldsymbol{P}||||(x\boldsymbol{I}-(\boldsymbol{A}+\boldsymbol{P}))^{-1}|||dx|.$$

If we choose Γ to be a contour at distance $d>0$ from the spectra of $\boldsymbol{A}$, we can have:

$$(x\boldsymbol{I}-\boldsymbol{A})^{-1} = \boldsymbol{U}(x\boldsymbol{I}-\boldsymbol{\Lambda})^{-1}\boldsymbol{U}^T,$$

where $\boldsymbol{A}=\boldsymbol{U}\boldsymbol{\Lambda}\boldsymbol{U}^T$ is the eigendecomposition of $\boldsymbol{A}$ and the corresponding norm is:

$$||(x\boldsymbol{I}-\boldsymbol{A})^{-1}|| = ||(x\boldsymbol{I}-\boldsymbol{\Lambda})^{-1}|| = \max_{\lambda\in\sigma(\boldsymbol{A})}\frac{1}{|x-\lambda|} = \frac{1}{\text{dist}(x,\sigma(\boldsymbol{A}))} = \frac{1}{d},$$

where $\sigma(\boldsymbol{A})$ is the spectrum of $\boldsymbol{A}$ and dist$(\cdot)$ is the distance function.

For a small perturbation $||\boldsymbol{P}||$, the spectrum of $\boldsymbol{A}+\boldsymbol{P}$ will lie in a neighborhood of the spectrum of $\boldsymbol{A}$. Specifically, for any eigenvalue λ' of $\boldsymbol{A}+\boldsymbol{P}$, there exists an eigenvalue λ of $\boldsymbol{A}$ such that $|\lambda'-\lambda|\le||\boldsymbol{P}||$, which implies:

$$\text{dist}(x,\sigma(\boldsymbol{A}+\boldsymbol{P})) \ge \text{dist}(x,\sigma(\boldsymbol{A})) - ||\boldsymbol{P}||.$$

Then for $x\notin\sigma(\boldsymbol{A}+\boldsymbol{P})$, we use the Neumann series:

$$(x\boldsymbol{I}-(\boldsymbol{A}+\boldsymbol{P}))^{-1} = (x\boldsymbol{I}-\boldsymbol{A})^{-1}\sum_{i=0}^{+\infty}[\boldsymbol{P}(x\boldsymbol{I}-\boldsymbol{A})^{-1}]^i,$$

which converges if $||\boldsymbol{P}(x\boldsymbol{I}-\boldsymbol{A})^{-1}||<1$.

Afterward, we take its norm and apply submultiplicativity:

$$\begin{aligned}||(x\boldsymbol{I}-(\boldsymbol{A}+\boldsymbol{P}))^{-1}|| &\le \frac{||(x\boldsymbol{I}-\boldsymbol{A})^{-1}||}{1-||\boldsymbol{P}||||(x\boldsymbol{I}-\boldsymbol{A})^{-1}||}\\ &= \frac{1}{\text{dist}(x,\sigma(\boldsymbol{A}))-||\boldsymbol{P}||}\\ &\le \frac{1}{d}.\end{aligned}$$

Then let $M=\max_{x\in\Gamma}|x^\alpha|$ and L$(\cdot)$ be the length function, we can have:

$$||\boldsymbol{A}^\alpha - (\boldsymbol{A}+\boldsymbol{P})^\alpha|| \le \frac{1}{2\pi d^2}M||\boldsymbol{P}||\text{L}(\Gamma)$$

Table 4: Statistics of 12 real-world datasets, where n_n is the number of normal graphs, n_a is the number of anomalous graphs, $h = \frac{n_a}{n_n+n_a}$ is the anomalous ratio, $\bar{n}$ is the average number of nodes, $\bar{m}$ is the average number of edges, and F is the number of attributes.

Dataset	MCF-7	MOLT-4	PC-3	SW-620	NCI-H23	OVCAR-8	P388	SF-295	SN12C	UACC257	PROTEINS_full	DBLP_v1
n_n	25476	36625	25941	38122	38296	38437	39174	38246	38049	38345	663	9926
n_a	2294	3140	1568	2410	2057	2079	2298	2025	1955	1643	450	9530
h	0.0826	0.079	0.057	0.0595	0.051	0.0513	0.0554	0.0503	0.0489	0.0411	0.4043	0.4898
$\bar{n}$	26.4	26.1	26.36	26.06	26.07	26.08	22.11	26.06	26.08	262.09	39.06	10.48
$\bar{m}$	28.53	28.14	28.49	28.09	28.1	28.11	23.56	28.09	28.11	28.13	72.82	19.65
F	46	64	45	65	65	65	72	65	65	64	3	41325

Define $c = \frac{ML(\Gamma)}{2\pi d^2}$, yielding $||\boldsymbol{A}^\alpha - (\boldsymbol{A}+\boldsymbol{P})^\alpha|| \leq c||\boldsymbol{P}||$.

Besides, for a generated adjacency matrix from the perturbation method, it can be diagonalized, so we can have:

$$(\boldsymbol{A}+\boldsymbol{P}) - (\boldsymbol{A}+\boldsymbol{P})^\alpha = \boldsymbol{V}(\boldsymbol{\Sigma} - \boldsymbol{\Sigma}^\alpha)\boldsymbol{V}^T,$$

where $\boldsymbol{A}+\boldsymbol{P} = \boldsymbol{V}\boldsymbol{\Sigma}\boldsymbol{V}^T$ and $\boldsymbol{\Sigma}$ is a diagonal matrix composed of $(\lambda_1, \lambda_2, \cdots, \lambda_n)$. Take the norm, we can get:

$$||(\boldsymbol{A}+\boldsymbol{P}) - (\boldsymbol{A}+\boldsymbol{P})^\alpha|| = \max_i |\lambda_i - \lambda_i^\alpha|.$$

Finally, by applying the submultiplicativity, we can conclude:

$$||\boldsymbol{A}^\alpha - (\boldsymbol{A}+\boldsymbol{P})|| \leq c||\boldsymbol{P}|| + \max_i |\lambda_i - \lambda_i^\alpha|,$$

where c depends on α and the spectral gap of $\boldsymbol{P}$, and λ_i is the i-th eigenvalue of $\boldsymbol{A}+\boldsymbol{P}$. □

Proof of Proposition 1. Assume the two prediction error rates for original graphs and corresponding fractional graphs are two Bernoulli variables with mean δ, and the correlation of the errors is ρ. Then we have the joint error rate:

$$\mathbb{P}(\text{Both wrong}) = \delta^2 + \rho\delta(1-\delta).$$

Since the original error rate is δ, the mutual verification will lower the error with the reduction factor $\delta + \rho\delta(1-\delta)$.

According to the above analysis, the error rate of mutual verification is $p = \delta^2 + \rho\delta(1-\delta)$. Assuming it is also a Bernoulli variable, the variance can be calculated as:

$$v = (\delta^2 + \rho\delta(1-\delta))(1 - \delta^2 - \rho\delta(1-\delta)).$$

For a small error rate δ, we can approximate it as $v = \rho\delta(1-\rho\delta)$. Therefore, the reduction factor of variance is close to ρ. □

B Datasets and Baselines

Datasets. The datasets used in our experiments are collected by TUDataset (Morris et al., 2020). Specifically, MCF-7, MOLT-4, PC-3, SW-620, NCI-H23, OVCAR-8, P388, SF-295, SN12C, and UACC257 are small-molecule datasets from PubChem [2], which provide information on the biological activities of small molecules. In these datasets, nodes represent atoms within chemical compounds, while edges indicate the chemical bonds connecting pairs of atoms. Each dataset corresponds to a specific type of cancer screening, with outcomes classified as either active or inactive. We consider inactive chemical compounds as normal graphs and active compounds as anomalous graphs. Furthermore, the attributes are derived from node labels using one-hot encoding.

Besides, PROTEINS_full is a typical bioinformatics-related dataset (Dobson & Doig, 2003), which processes several proteins represented as graphs. In this dataset, nodes and edges are formulated in a similar way to small-molecule datasets from PubChem. This dataset aims to classify enzymes and non-enzymes, which are denoted as normal and anomalous graphs, respectively.

[2] https://pubchem.ncbi.nlm.nih.gov/

Beyond the above datasets, we also conduct experiments on DBLP_v1 (Pan et al., 2013), which consists of bibliography data in computer science. Each record in DBLP_v1 is associated with a number of attributes such as abstract, authors, year, venue, title, and reference ID. Since the dimension of the attributes is high, we first utilize EVD to lower the dimension to 16. In this dataset, nodes denote papers, while edges represent reference relations between papers. The classification task is to predict whether a paper belongs to the CVPR (computer vision and pattern recognition) or DBDM (database and data mining) conferences, which are seen as normal and anomalous, respectively.

Baselines. The first group is graph classification models:

- GCN (Kipf & Welling, 2017): a GNN that uses a convolution function on a graph to propagate information within the neighborhood of nodes;
- GraphSAGE (Hamilton et al., 2017): a GNN that leverages a sampling technique to aggregate features from the neighborhood.
- GAT (Velickovic et al., 2018): a GNN that adopts an attention mechanism within the neighborhood of each node;
- GIN (Xu et al., 2019): a GNN that follows graph isomorphism to capture the properties of a graph.
- LRGNN (Wei et al., 2023): a GNN stacking multiple GNNs to extract the long-range dependencies;
- GRDL (Wang & Fan, 2024): a GNN treating node embeddings as a discrete distribution, enabling direct classification without global pooling.

The second group is GAD models:

- iGAD (Zhang et al., 2022): a GNN with a substructure-aware component to capture properties of anomalous graphs.
- GmapAD (Ma et al., 2023a): a GNN mapping graphs into a latent space where anomalies can be effectively detected;
- RQGNN (Dong et al., 2024): a GNN using Rayleigh Quotient to obtain information from both spectral and spatial spaces.
- UniGAD (Lin et al., 2024): a GNN that unifies different levels of graph-related tasks.

The third group is graph-level augmentation frameworks:

- MAA (Yoo et al., 2022): a framework using node split and merge, and subgraph mix to augment graphs heuristically;
- GLA (Yue et al., 2022): a framework augmenting data in the representation space from the most difficult direction while keeping the label of augmented data the same as the original samples;
- GMixup (Han et al., 2022): a framework that interpolates graphons of different classes in the Euclidean space to get mixed graphons;
- FGWMixup (Ma et al., 2023b): a framework that seeks a midpoint of source graphs in the Fused Gromov-Wasserstein metric space to interpolate graphons of different classes.

C Algorithm

Algorithm 1: Preprocess

Input: $\mathcal{D}, k_l, k_s$

for G *in* $\mathcal{D}$ **do**

$G.\boldsymbol{A} \leftarrow \frac{1}{2}(\boldsymbol{I} + G.\boldsymbol{D}^{-\frac{1}{2}} * G.\boldsymbol{A} * G.\boldsymbol{D}^{-\frac{1}{2}})$;

$G.\boldsymbol{U}_l, G.\boldsymbol{\Lambda}_l \leftarrow \text{EVD}(G.\boldsymbol{A}, k_l)$;

$G.\boldsymbol{U}_s, G.\boldsymbol{\Lambda}_s \leftarrow \text{EVD}(G.\boldsymbol{A}, k_s)$;

D Complexity Analysis

For the Preprocess function, we first analyze the time complexity of the matrix multiplication. Since we utilize sparse matrices to conduct the experiment, the time complexity of the multiplication is

Algorithm 2: FGG

Input: $\mathcal{D}, H_l, H_s$
Output: $\mathcal{D}'$
for G *in* $\mathcal{D}$ **do**
  for $i = 0$ *to* H_l **do**
    $G_l \leftarrow G_l + \boldsymbol{\omega}_l[i] * G.\boldsymbol{U}_l * G.\boldsymbol{\Lambda}_l^{\boldsymbol{\alpha}_l[i]} * G.\boldsymbol{U}_l^T$;
  for $i = 0$ *to* H_l **do**
    $G_s \leftarrow G_s + \boldsymbol{\omega}_s[i] * G.\boldsymbol{U}_s * G.\boldsymbol{\Lambda}_s^{\boldsymbol{\alpha}_s[i]} * G.\boldsymbol{U}_s^T$;
  $G' \leftarrow \omega * G_l + (1 - \omega) * G_s$;
  $\mathcal{D}' \leftarrow \mathcal{D}' \cup G'$;
Return $\mathcal{D}'$;

Algorithm 3: WDML

Input: $f, \mathcal{D}, \mathcal{D}'$
for G, G' *in* $\mathcal{D}, \mathcal{D}'$ **do**
  $\boldsymbol{s}, \boldsymbol{o} \leftarrow f(G)$;
  $\boldsymbol{s}', \boldsymbol{o}' \leftarrow f(G')$;
  $m \leftarrow \frac{1-\cos(\boldsymbol{o},\boldsymbol{o}')}{2}$;
  $L_{\text{WDML}} \leftarrow L_{\text{WDML}} + -\frac{1}{N_{G.y}} \log \frac{e^{\boldsymbol{s}[G.y]-m}}{e^{\boldsymbol{s}[G.y]-m}+e^{\boldsymbol{s}[1-G.y]}}$;
L_{WDML}.backward();

Algorithm 4: MVP

Input: $f, \mathcal{D}, \mathcal{D}', \tau_n, \tau_a$
Output: $\mathcal{D}''$
for G, G' *in* $\mathcal{D}, \mathcal{D}'$ **do**
  $\boldsymbol{s}, \boldsymbol{o} \leftarrow f(G)$;
  $\boldsymbol{s}', \boldsymbol{o}' \leftarrow f(G')$;
  if $\boldsymbol{s}[0] < \tau_n \wedge \boldsymbol{s}'[0] \leq \tau_n$ **then**
    $G.y \leftarrow 0$;
    $\mathcal{D}'' \leftarrow \mathcal{D}'' \cup G$;
  else if $\boldsymbol{s}[1] < \tau_a \wedge \boldsymbol{s}'[1] \leq \tau_a$ **then**
    $G.y \leftarrow 1$;
    $\mathcal{D}'' \leftarrow \mathcal{D}'' \cup G$;
Return $\mathcal{D}''$;

Algorithm 5: FracAug

Input: $f, \mathcal{D}_{train}, \mathcal{D}_{val}, \mathcal{D}_{test}, H_l, H_s, k_l, k_s, e_{warmup}, e_{aug}, \tau_n, \tau_a$
Preprocess($\mathcal{D}_{train} \cup \mathcal{D}_{val} \cup \mathcal{D}_{test}, k_l, k_s$);
$D'_{train} \leftarrow D_{train}$;
for $e = 0$ *to* e_f **do**
  if $e > e_{warmup} \wedge e\%e_{aug} == 0$ **then**
    for $e' = 0$ *to* e_{FGG} **do**
      $\mathcal{D}_{temp} \leftarrow$ FGG($\mathcal{D}_{train}, H_l, H_s$);
      WDML($f, \mathcal{D}, \mathcal{D}_{temp}$);
    $\mathcal{D}_{temp} \leftarrow$ FGG($\mathcal{D}_{val} \cup \mathcal{D}_{test}, H_l, H_s$);
    $\mathcal{D}_{temp} \leftarrow$ MVP($f, \mathcal{D}_{val} \cup \mathcal{D}_{test}, \mathcal{D}_{temp}, \tau_n, \tau_a$);
    $D'_{train} \leftarrow D_{train} \cup \mathcal{D}_{temp}$;
  train(f, D'_{train}));

Table 5: Comparison of average running time.

Datasets	NSv+FA	NodeSam	SubMix	GLAv+FA	GLA	GMixupv+FA	GMixup	FGWMixupv+FA	FGWMixup
MCF-7	92.18+58.56	1282.84	876.39	92.00+73.57	1493.55	90.54+107.51	51.64	92.29+106.76	553.90
MOLT-4	133.10+83.70	1352.07	881.00	128.18+113.70	1924.35	131.30+150.27	64.55	129.84+176.65	855.30
PC-3	91.26+70.05	1249.30	879.76	90.55+64.84	1484.25	90.46+110.26	51.52	90.70+98.88	613.19
SW-620	133.80+102.96	1248.39	875.52	136.75+99.17	2076.36	133.42+158.47	75.68	133.50+153.30	873.41
NCI-H23	130.20+108.73	1301.09	883.09	130.63+93.86	2091.18	131.54+134.20	63.68	131.07+143.53	896.41
OVCAR-8	131.48+98.46	1351.17	890.68	132.62+102.36	2077.41	132.07+168.57	61.41	131.23+130.47	986.01
P388	120.73+103.09	1258.91	884.15	120.43+90.74	2140.75	122.53+145.56	60.47	121.12+135.97	965.61
SF-295	130.92+84.81	1359.48	879.52	131.26+110.89	2063.71	130.77+168.45	66.25	130.43+150.38	915.04
SN12C	129.79+102.16	1334.82	914.87	129.62+87.23	2044.62	130.21+148.75	68.58	128.95+134.45	933.55
UACC257	128.63+84.19	1387.70	930.43	128.39+88.05	2053.46	129.04+138.45	58.83	129.65+142.53	891.95
PROTEINS_full	9.03+6.67	206.50	202.65	8.50+8.86	69.30	8.62+7.98	4.58	8.84+7.63	30.25
DBLP_v1	38.86+62.06	1335.42	897.90	36.99+61.50	1123.57	39.22+99.74	30.51	39.15+105.83	155.67

$O(\text{nnz}(G.\boldsymbol{D}^{-\frac{1}{2}}) * \text{nnz}(G.\boldsymbol{A}) + \text{nnz}(G.\boldsymbol{D}^{-\frac{1}{2}} * G.\boldsymbol{A}) * \text{nnz}(G.\boldsymbol{D}^{-\frac{1}{2}})$, where nnz means non-zero entries of the matrix. Then, by adopting EVD to only keep the top-k_l largest and top-k_s smallest eigenvalues, the time complexity can be $O(n * (k_l^2 + k_s^2) + m * (k_l + k_s))$, where n, m is the number of nodes/edges. As shown in Algorithm 5, Preprocess can be called before the training process, and thus it won't burden the training or inference of our FracAug.

Then, we analyze the time complexity of FGG for each graph. As presented in Algorithm 2, we perform sparse matrix multiplication for every sample H_l and H_s times. Besides, since $G.\boldsymbol{\Lambda}$ is a diagonal matrix, the time complexity of multiplying $G.\boldsymbol{\Lambda}$ is the same as that of multiplying $G.\boldsymbol{\Lambda}^\alpha$. Therefore, the total time complexity of FGG is $O(\text{FGG}) = O(\text{nnz}(G.\boldsymbol{U}_l) * k_l + \text{nnz}(G.\boldsymbol{U}_l * G.\boldsymbol{\Lambda}_l) * \text{nnz}(G.\boldsymbol{U}_l^T) + \text{nnz}(G.\boldsymbol{U}_s) * k_s + \text{nnz}(G.\boldsymbol{U}_s * G.\boldsymbol{\Lambda}_s) * \text{nnz}(G.\boldsymbol{U}_s^T))$.

Next, we analyze the time complexity of WDML in Algorithm 3. Assuming that we only have one sample in $\mathcal{D}_{train}$, then the time complexity of WDML is $O(\text{WDML}) = O(d)$, where d is the dimension of the generated graph embedding $\boldsymbol{o}$.

Moreover, as shown in Algorithm 4, in MVP, we only need to see if the probability predicted by the given GNN satisfies the criterion, so the time complexity for MVP is $O(\text{MVP}) = O(1)$.

Finally, in Algorithm 5, we combine all the time complexities together within one training epoch of the given GNN f, assuming the time complexity of f for each sample is $O(f)$, then we have the total complexity as $O(e_{\text{FGG}} * (O(\text{FGG}) + O(\text{WDML}) + O(f)) * N_{train} + (O(\text{FGG}) + O(\text{MVP}) + O(f)) * (N_{val} + N_{test})$, where N_{val}, N_{test} represent the number of samples in $\mathcal{D}_{val}$ and $\mathcal{D}_{test}$, respectively.

In practice, we set e_{FGG} to 10 and e_{aug} to 25, which can reduce the computational cost, and FGG can still converge. According to the final complexity, we can see the dominant factor within each epoch is $O(e_{FGG} * O(f) * N_{train} + O(f) * (N_{val} + N_{test}))$. For such a factor, we need to calculate it in total $\frac{e_f - e_{warmup}}{e_{aug}} * e_{FGG}$ times, which is much less than the original training epoch of f. Hence, the increase in time complexity will not be the limitation of our FracAug in real applications.

In Table 5, we present a detailed runtime comparison between FracAug and several leading graph augmentation methods. For each technique, we decompose the total computational cost into a one-time preprocessing phase, performed once per dataset, and the subsequent training time measured over multiple epochs. While some baselines require repeated feature perturbations or costly online sampling at every iteration, FracAug's eigenvalue decomposition is only performed during preprocessing. As a result, the per-epoch training overhead of FracAug remains on par with, or even below, that of competing approaches, despite leveraging additional spectral information to boost anomaly detection performance.

Crucially, these efficiency gains do not come at the expense of detection performance. Across all datasets and baseline comparisons, FracAug consistently delivers state-of-the-art AUROC, AUPRC, and F1-score results while maintaining competitive total runtimes. By amortizing the heavier spectral computations over the entire training cycle and by implementing optimized matrix operations, FracAug strikes an effective balance between computational tractability and augmentation quality. This combination of speed and performance underscores the practical value of our method: practitioners can readily adopt FracAug for real graph applications without incurring prohibitive time costs.

Table 6: Hyperparameters of 12 datasets based on GIN.

Datasets	MCF-7	MOLT-4	PC-3	SW-620	NCI-H23	OVCAR-8	P388	SF-295	SN12C	UACC257	PROTEINS_full	DBLP_v1
k_l	4	4	4	4	3	3	4	3	4	4	3	4
H_l	4	3	3	3	3	3	4	4	3	4	4	4
k_s	3	4	3	3	4	3	4	3	3	4	3	3
H_s	4	4	3	3	3	3	4	3	3	4	4	3
e_{warmup}	50	25	50	50	25	25	50	50	25	50	50	25

Table 7: Varying k_l-k_s-H_l-H_s on different datasets based on GIN.

Datasets	PROTEINS_full			DBLP_v1		
k_l-k_s-H_l-H_s	AUROC	AUPRC	F1-score	AUROC	AUPRC	F1-score
3-3-3-3	0.6174	0.6298	0.6187	0.7950	0.8514	0.7947
3-3-3-4	0.6141	0.6327	0.6142	0.7972	0.8572	0.7955
3-3-4-3	0.6103	0.6291	0.6104	0.7995	0.8546	0.7992
3-3-4-4	0.6174	0.6358	0.6175	0.7925	0.8486	0.7925
3-4-3-3	0.6174	0.6298	0.6187	0.7950	0.8514	0.7947
3-4-3-4	0.6082	0.6304	0.6072	0.7972	0.8572	0.7955
3-4-4-3	0.6103	0.6291	0.6104	0.7995	0.8546	0.7992
3-4-4-4	0.6094	0.6210	0.6104	0.7982	0.8524	0.7982
4-3-3-3	0.6174	0.6298	0.6187	0.7885	0.8509	0.7867
4-3-3-4	0.6124	0.6284	0.6133	0.7972	0.8538	0.7967
4-3-4-3	0.6161	0.6295	0.6174	0.8044	0.8626	0.8028
4-3-4-4	0.6138	0.6316	0.6142	0.8007	0.8570	0.8000
4-4-3-3	0.6174	0.6298	0.6187	0.7972	0.8579	0.7953
4-4-3-4	0.6161	0.6295	0.6174	0.7994	0.8560	0.7987
4-4-4-3	0.6108	0.6334	0.6095	0.8015	0.8617	0.7996
4-4-4-4	0.6094	0.6210	0.6104	0.8008	0.8577	0.7999

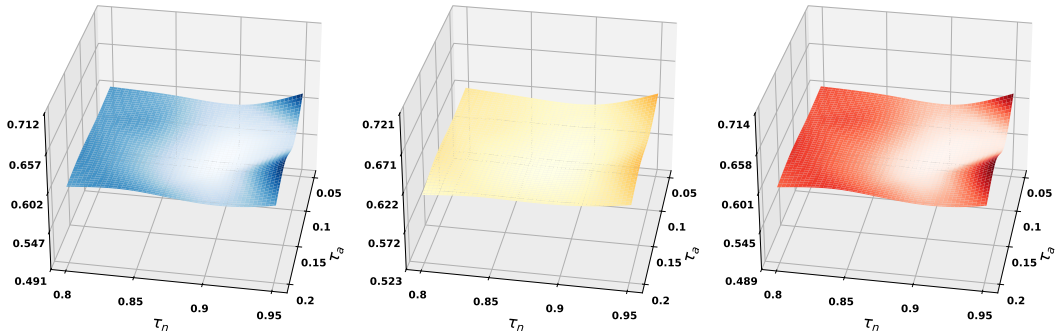
E Experimental Settings

Table 6 provides a comprehensive list of our hyperparameters based on GIN. We use grid search to train FracAug, which yields the best sum of AUROC, AUPRC, and F1-score on the validation set, and report the corresponding test performance. Specifically, k_l, H_l, k_s, H_s range from the set $\{3, 4\}$, and the number of warmup epoch for GNN is selected from $\{25, 50\}$, to reduce the cost of search. In the next Section F, we further analyze the influence of k_l, H_l, k_s, H_s on the AUROC, AUPRC, and F1-score of different datasets based on GIN. As for the experimental environment, we conduct all the experiments on an NVIDIA Quadro RTX 8000 for a fair comparison.

F Hyperarameter Analysis

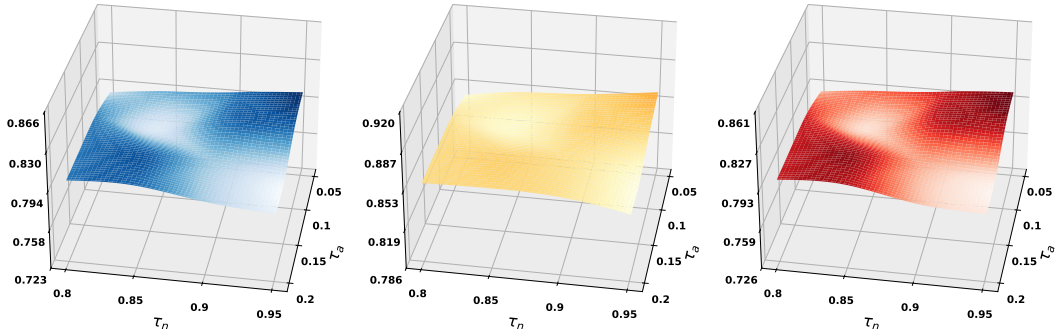
Table 7 presents a systematic exploration of FracAug's performance, measured in AUROC, AUPRC, and F1-score, when varying k_l, k_s, H_l, H_s between 3 and 4. Specifically, k_l, k_s denote the top-k_l largest and top-k_s eigenvalues generated by EVD, while H_l, H_s denote the number of learnable fractional powers of the matrix for the largest and smallest eigenvalues. By sweeping each of these four parameters, we generate a compact grid of 16 configurations. As detailed in Table 6, each evaluation metric prefers a slightly different quadruple of (k_l, k_s, H_l, H_s), but more importantly, Table 7 reveals that the detection performance barely wavers across all measured combinations. This robustness not only validates the spectral augmentation strategy at the heart of FracAug but also suggests that practitioners can avoid laborious hyperparameter sweeps without sacrificing anomaly detection performance.

In a parallel study, Figures 3 and 4 examine the sensitivity of FracAug to the pseudo-labeling thresholds τ_n and τ_a. Here, τ_n specifies the percentile above which a node is considered "normal," and τ_a the percentile below which it is flagged as "anomalous." By varying τ_n from 0.8 to 0.95 and τ_a from 0.05 to 0.2, we again explore 16 threshold pairs on each dataset, logging the resulting AUROC, AUPRC, and F1-score for every pair. Remarkably, all three metrics remain essentially



(a) AUROC (b) AUPRC (c) F1-score

Figure 3: Varying τ_a and τ_n for PROTEINS_full based on GIN.



(a) AUROC (b) AUPRC (c) F1-score

Figure 4: Varying τ_a and τ_n for DBLP_v1 based on GIN.

flat throughout this entire range, indicating that FracAug's pseudo-labeling module is forgiving of moderate threshold choices. Leveraging this newfound stability, we adopt the pair $\tau_n = 0.05$ and $\tau_a = 0.95$ as our default across all datasets, thereby slashing the computational overhead of threshold tuning without meaningfully affecting detection performance.

G Ablation Study

To examine the effectiveness of each component in FracAug, we conduct ablation study on 12 datasets based on GIN, which is shown in Table 8. Specifically, the gray column represents the performance of GIN with FracAug, w/o largest and w/o smallest denotes removing the fractional graphs generated by top-k_l largest and top-k_s smallest eigenvalues, respectively, w/o WDML means replacing our proposed WDML with weighted cross-entropy loss, and w/o MVP pseudo-labels samples in $\mathcal{D}_{val} \cup \mathcal{D}_{test}$ using only the predicted probability of original graphs. As shown in Table 8, FracAug consistently outperforms its variants, w/o largest, w/o smallest, w/o WDML, and w/o MVP, which demonstrates the benefits of these components.

H Margin Loss Comparison

Next, we investigate the performance of FracAug with different margin loss as stated in Section 4.3, by conducting comparison on 12 datasets based on GIN. As shown in Table 9, FracAug consistently outperforms its variants, Softmax, LMCL, and LDAM. Such a phenomenon demonstrates that sample-specific decision boundaries can be more effective than fixed margins for GAD tasks, aligning with our analysis in Section 4.3.

Table 8: Ablation study.

Datasets	Metrics	GIN	+FA	w/o largest	w/o smallest	w/o WDML	w/o MVP
MCF-7	AUROC	0.5867	0.5976	0.5848	0.5889	0.5860	0.5835
	AUPRC	0.2830	0.2971	0.2842	0.2882	0.2813	0.2790
	F1-score	0.5366	0.5421	0.5317	0.5351	0.5372	0.5350
MOLT-4	AUROC	0.5733	0.5854	0.5770	0.5760	0.5772	0.5754
	AUPRC	0.2830	0.3001	0.2974	0.2862	0.2981	0.2881
	F1-score	0.5072	0.5103	0.5001	0.5085	0.4998	0.5060
PC-3	AUROC	0.5969	0.6119	0.5963	0.6018	0.6026	0.5975
	AUPRC	0.2797	0.2893	0.2797	0.2815	0.2806	0.2780
	F1-score	0.5063	0.5205	0.5055	0.5124	0.5145	0.5092
SW-620	AUROC	0.5938	0.6004	0.5941	0.5938	0.5949	0.5930
	AUPRC	0.2776	0.2813	0.2737	0.2778	0.2800	0.2697
	F1-score	0.5090	0.5155	0.5132	0.5089	0.5082	0.5155
NCI-H23	AUROC	0.5897	0.5968	0.5893	0.5911	0.5866	0.5925
	AUPRC	0.2566	0.2659	0.2564	0.2633	0.2614	0.2656
	F1-score	0.5059	0.5073	0.5054	0.5013	0.4968	0.5013
OVCAR-8	AUROC	0.5935	0.5963	0.5911	0.5918	0.5911	0.5904
	AUPRC	0.2573	0.2612	0.2579	0.2565	0.2573	0.2605
	F1-score	0.5118	0.5123	0.5074	0.5100	0.5081	0.5038
P388	AUROC	0.5565	0.5913	0.5620	0.5708	0.5730	0.5599
	AUPRC	0.2850	0.3309	0.2917	0.3050	0.3074	0.3014
	F1-score	0.4468	0.4491	0.4482	0.4470	0.4470	0.4371
SF-295	AUROC	0.5844	0.6076	0.5971	0.5949	0.5920	0.5996
	AUPRC	0.2766	0.2832	0.2723	0.2716	0.2653	0.2790
	F1-score	0.4803	0.5047	0.5001	0.4975	0.4994	0.4973
SN12C	AUROC	0.5995	0.6079	0.5993	0.5963	0.5910	0.5990
	AUPRC	0.2696	0.2746	0.2685	0.2666	0.2632	0.2675
	F1-score	0.5030	0.5110	0.5041	0.5013	0.4971	0.5046
UACC257	AUROC	0.5877	0.6015	0.5939	0.5854	0.5938	0.5914
	AUPRC	0.2480	0.2598	0.2517	0.2527	0.2585	0.2586
	F1-score	0.4906	0.4983	0.4956	0.4835	0.4890	0.4858
PROTEINS_full	AUROC	0.5799	0.6174	0.5986	0.5842	0.5990	0.5854
	AUPRC	0.6259	0.6358	0.6111	0.5988	0.6205	0.6042
	F1-score	0.5679	0.6175	0.5995	0.5848	0.5976	0.5857
DBLP_v1	AUROC	0.6231	0.8044	0.7615	0.7828	0.7762	0.7628
	AUPRC	0.7201	0.8626	0.8301	0.8411	0.8374	0.8294
	F1-score	0.5996	0.8028	0.7594	0.7829	0.7760	0.7619

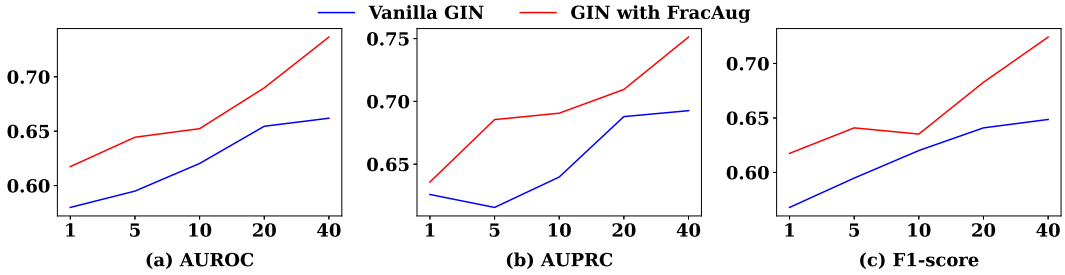


Figure 5: Varying training size (%) for PROTEINS_full.

I Performance with More Training Data

To further demonstrate the superior ability of our proposed FracAug, we conduct experiments on PROTEINS_full and DBLP_v1, varying by the training size (%), as shown in Figures 5 and 6. As we can see, the red lines, which represent the performance of GIN with FracAug, are always on top of the figures, which demonstrates that with more training data, our FracAug can still boost the performance of the given baseline consistently in terms of AUROC, AUPRC, and F1-score. To sum up, such experiments further prove that our FracAug is a more general and practical framework.

Table 9: Comparison of different margin loss.

Datasets	Metrics	GIN	+FA	Softmax	LMCL	LDAM
MCF-7	AUC	0.5867	0.5976	0.5844	0.5880	0.5844
	AUPRC	0.2830	0.2971	0.2781	0.2847	0.2796
	MF1	0.5366	0.5421	0.5378	0.5372	0.5360
MOLT-4	AUC	0.5733	0.5854	0.5760	0.5797	0.5751
	AUPRC	0.2830	0.3001	0.2969	0.2952	0.2857
	MF1	0.5072	0.5103	0.4991	0.5059	0.5076
PC-3	AUC	0.5969	0.6119	0.6021	0.6037	0.6001
	AUPRC	0.2797	0.2893	0.2809	0.2778	0.2769
	MF1	0.5063	0.5205	0.5134	0.5195	0.5144
SW-620	AUC	0.5938	0.6004	0.5947	0.5936	0.5931
	AUPRC	0.2776	0.2813	0.2779	0.2768	0.2720
	MF1	0.5090	0.5155	0.5100	0.5092	0.5133
NCI-H23	AUC	0.5897	0.5968	0.5913	0.5896	0.5887
	AUPRC	0.2566	0.2659	0.2650	0.2612	0.2622
	MF1	0.5059	0.5073	0.5001	0.5012	0.4990
OVCAR-8	AUC	0.5935	0.5963	0.5905	0.5890	0.5932
	AUPRC	0.2573	0.2612	0.2557	0.2570	0.2579
	MF1	0.5118	0.5123	0.5087	0.5051	0.5107
P388	AUC	0.5565	0.5913	0.5864	0.5653	0.5602
	AUPRC	0.2850	0.3309	0.3265	0.3080	0.2901
	MF1	0.4468	0.4491	0.4465	0.4377	0.4469
SF-295	AUC	0.5844	0.6076	0.5943	0.5898	0.5955
	AUPRC	0.2766	0.2832	0.2664	0.2712	0.2715
	MF1	0.4803	0.5047	0.5017	0.4909	0.4985
SN12C	AUC	0.5995	0.6079	0.5914	0.5955	0.6004
	AUPRC	0.2696	0.2746	0.2661	0.2601	0.2707
	MF1	0.5030	0.5110	0.4950	0.5068	0.5034
UACC257	AUC	0.5877	0.6015	0.5905	0.5899	0.5935
	AUPRC	0.2480	0.2598	0.2584	0.2581	0.2557
	MF1	0.4906	0.4983	0.4849	0.4844	0.4912
PROTEINS_full	AUC	0.5799	0.6174	0.6002	0.5937	0.6051
	AUPRC	0.6259	0.6358	0.6235	0.6078	0.6231
	MF1	0.5679	0.6175	0.5987	0.5945	0.6051
DBLP_v1	AUC	0.6231	0.8044	0.7776	0.7834	0.7874
	AUPRC	0.7201	0.8626	0.8383	0.8463	0.8502
	MF1	0.5996	0.8028	0.7774	0.7817	0.7855

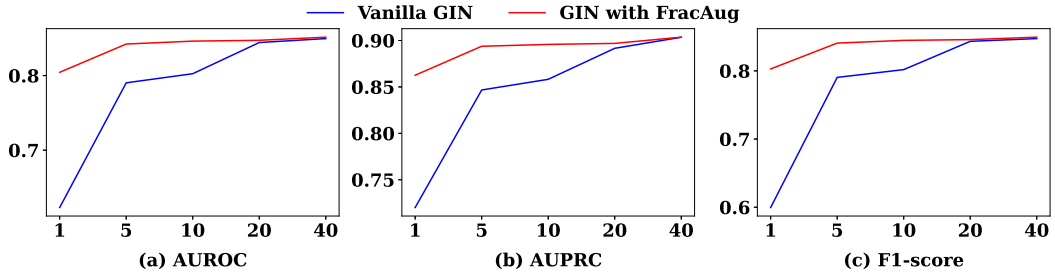


Figure 6: Varying training size (%) for DBLP_v1.

J Learned Parameters

Here, we present the learned parameters based on GIN, $\boldsymbol{\alpha}_l, \boldsymbol{\omega}_l, \boldsymbol{\alpha}_s, \boldsymbol{\omega}_s$, and $\boldsymbol{\omega}$ in Table 10. Specifically, $\boldsymbol{\alpha}_l, \boldsymbol{\alpha}_s$ represent the learned fractional powers for the largest and smallest eigenvalues, respectively, $\boldsymbol{\omega}_l, \boldsymbol{\omega}_s$ denote the corresponding learned coefficients for the fractional graphs, and $\boldsymbol{\omega}$ means the balanced coefficient between the group of fractional graphs from the largest and smallest eigenvalues. Recap from Section 4.2, we utilize H_l and H_s to control the number of combined fractional graphs from different eigenvalues, so the number of entries in $\boldsymbol{\alpha}_l, \boldsymbol{\omega}_l$ is equal to H_l and that of $\boldsymbol{\alpha}_s, \boldsymbol{\omega}_s$ is the same as H_s. Since we only consider the largest and smallest eigenvalue groups, the size of $\boldsymbol{\omega}$ should

Table 10: Learned parameters

Datasets	MCF-7	MOLT-4	PC-3	SW-620	NCI-H23	OVCAR-8	P388	SF-295	SN12C	UACC257	PROTEINS_full	DBLP_v1
α_l	0.9062	1.4187	1.0862	2.186	0.4991	0.8986	2.8806	0.8845	2.7404	1.6178	1.1407	1.8663
	2.0063	2.2813	1.9282	1.5108	0.3535	0.1262	2.9620	2.9038	0.5877	1.9339	0.6885	1.1374
	1.4770	1.7911	1.9928	1.8073	2.3672	1.1480	2.3129	2.3939	1.0219	2.4874	2.5432	1.1843
	1.4269	-	-	-	-	-	1.2674	2.4786	-	2.2634	1.4364	2.8254
ω_l	0.2921	0.2655	0.3450	0.2667	0.2793	0.3476	0.1639	0.3051	0.2385	0.1165	0.3842	0.1685
	0.3108	0.3101	0.3349	0.5068	0.4612	0.3615	0.3654	0.1747	0.3833	0.2806	0.1535	0.1714
	0.1549	0.4244	0.3202	0.2266	0.2595	0.2909	0.2728	0.2145	0.3782	0.3028	0.2927	0.4004
	0.2422	-	-	-	-	-	0.1979	0.3057	-	0.3000	0.1696	0.2597
α_s	2.3128	1.9621	2.8048	1.5579	2.4828	0.5030	1.5344	1.6912	2.9331	0.8959	1.1290	1.7201
	2.1539	0.4087	1.1722	1.8088	2.1366	1.4235	1.552	1.9896	1.5150	1.9589	0.1467	2.8101
	0.7485	2.9168	1.6098	0.4474	2.2668	2.6830	2.0318	1.8741	2.9352	2.8844	0.5492	2.0935
	2.5362	0.2449	-	-	-	-	2.7135	-	-	0.1148	0.0124	-
ω_s	0.3042	0.2821	0.3910	0.3756	0.3292	0.3384	0.1696	0.4052	0.4554	0.2916	0.2407	0.2517
	0.2569	0.2112	0.2750	0.4245	0.3630	0.3436	0.4001	0.2386	0.2343	0.2539	0.3285	0.4758
	0.2764	0.1519	0.3340	0.1999	0.3078	0.3180	0.2027	0.3562	0.3103	0.3122	0.2524	0.2725
	0.1625	0.3548	-	-	-	-	0.2277	-	-	0.1423	0.1785	-
ω	0.4634	0.6795	0.6481	0.5579	0.5263	0.4541	0.4408	0.5589	0.5724	0.5309	0.7134	0.5174
	0.5366	0.3205	0.3519	0.4421	0.4737	0.5459	0.5592	0.4411	0.4276	0.4691	0.2866	0.4826

be 2. Note that, according to Table 6, the optimal H_l, H_s varies by datasets, so the number of entries in Table 10 will also be different, where "-" represents no such entry in the vector. As shown in Table 10, the learned powers vary within the range of $(0, 3)$, which aligns with the parameter settings of common GNNs, as deeper GNNs may result in over-smoothing and shallower GNNs will lead to under-fitting. This experiment further shows the rationality of our proposed FracAug.