DeNoise: Learning Robust Graph Representations for Unsupervised Graph-Level Anomaly Detection

Qingfeng Chen, Haojin Zeng, Jingyi Jie, Shichao Zhang, and Debo Cheng

Abstract—With the rapid growth of graph-structured data in critical domains, unsupervised graph-level anomaly detection (UGAD) has become a pivotal task. UGAD seeks to identify entire graphs that deviate from normal behavioral patterns. However, most Graph Neural Network (GNN) approaches implicitly assume that the training set is clean, containing only normal graphs, which is rarely true in practice. Even modest contamination by anomalous graphs can distort learned representations and sharply degrade performance. To address this challenge, we propose DeNoise, a robust UGAD framework explicitly designed for contaminated training data. It jointly optimizes a graph-level encoder, an attribute decoder, and a structure decoder via an adversarial objective to learn noise-resistant embeddings. Further, DeNoise introduces an encoder anchor-alignment denoising mechanism that fuses high-information node embeddings from normal graphs into all graph embeddings, improving representation quality while suppressing anomaly interference. A contrastive learning component then compacts normal graph embeddings and repels anomalous ones in the latent space. Extensive experiments on eight real-world datasets demonstrate that DeNoise consistently learns reliable graph-level representations under varying noise intensities and significantly outperforms state-of-the-art UGAD baselines.

Index Terms—Graph-level anomaly detection, Unsupervised learning, Noise interference, Data enhancement.

I. INTRODUCTION

ANOMALY detection in graph-structured data is crucial in domains such as social networks, cybersecurity, and bioinformatics [1]–[3]. At the graph level, graph-level anomaly detection (GAD) flags entire graphs whose structural or attribute patterns deviate markedly from typical behavior [4]–[6]. Traditional GLAD pipelines largely rely on supervised learning, training graph neural network classifiers on labeled corpora to distinguish normal from anomalous graphs [7], [8]. Methods that emphasize expressive representation learning, such as Graph Isomorphism Networks, can further boost performance when labels are plentiful [9]. However, assembling high-quality graph-level labels is costly and slow, often requiring experts to inspect entire subgraphs or full graphs. Label distributions are typically long-tailed, with few positive (anomalous) examples and substantial concept drift over time.

Q. Chen and H. Zeng are with the School of Computer, Electronics and Information, Guangxi University, Nanning 530004, China (e-mail: qingfeng@gxu.edu.cn).

J. Jie and D. Cheng are with the School of Computer Science and Technology, Hainan University, Haikou 570228, China (e-mail: chengd@hainanu.edu.cn).

S. Zhang is with the Guangxi Key Laboratory of Multi-Source Information Mining & Security, Guangxi Normal University, Guilin 541004, China.

(Corresponding author: D. Cheng & J. Li; e-mail: chengd@hainanu.edu.cn &Jiuyong.Li@unisa.edu.au).
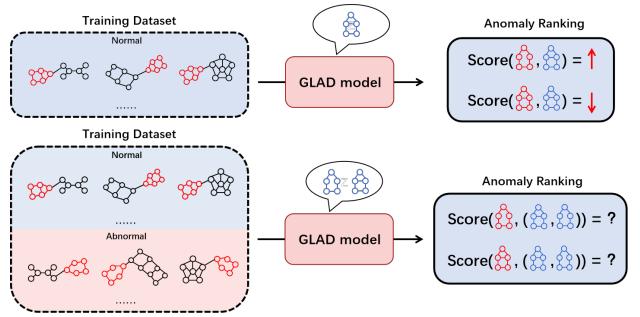
Fig. 1. Illustration of graph-level anomaly detection (GLAD). The labels above the GLAD model denote behavioral patterns learned during training. The $\text{Score}(\cdot)$ function quantifies the deviation of a new sample from these learned patterns, yielding an anomaly score (higher is more abnormal).

Consequently, supervised detectors trained on limited, static labels tend to overfit specific anomaly types and degrade when confronted with unseen or evolving patterns [10], [11].

Unsupervised graph-level anomaly detection (UGAD) has emerged as a highly promising alternative and has garnered widespread attention [12], [13]. The core idea of UGAD is to develop models that can identify anomalies without relying on labeled data. This is usually achieved by training models on datasets that are assumed to contain only normal graphs, thereby enabling the models to learn the patterns and characteristics of normal behavior. As shown in Figure 1, traditional UGAD models are trained on datasets containing only normal graphs, learning the behavior patterns and features of normal samples. When evaluating new samples, those with behaviors close to the learned model receive lower anomaly scores, while samples inconsistent with the learned behavior patterns are assigned higher anomaly scores. However, a major limitation of these methods is that it is often unrealistic to guarantee that the training set is completely free of anomalies in real-world scenarios. Even a small proportion of anomalous samples in the training data can significantly degrade model performance, as the model may inadvertently incorporate abnormal patterns into its learned representation of normal behavior. Consequently, the evaluation of new samples during the testing phase may fail, leading to unreliable anomaly detection outcomes.

More critically, existing UGAD methods often lack mechanisms to detect or mitigate the influence of anomalous samples during training. For instance, models like OCGIN [13] and OCGTL [12] rely on one-class classification objectives that assume all training data belong to a single "normal" distribution. If anomalies are present, these models tend to fit a broader

and more distorted distribution, reducing their sensitivity to true anomalies at test time. Similarly, reconstruction-based methods such as GLADPro [14] and MUSE [15] implicitly assume that anomalies will exhibit higher reconstruction errors. However, if anomalies are present during training, the model may learn to reconstruct them accurately, thereby diminishing their distinguishability.

Contrastive learning-based approaches like GLocalKD [16] and CVTGAD [17] suffer from the same issue: they treat all training graphs as positive samples, which allows anomalous patterns to be reinforced in the learned embeddings. SIGNET [18], despite introducing a multi-view subgraph information bottleneck to enhance interpretability, still assumes that the training set is clean. It selects representative subgraphs based on mutual information maximization, but when anomalous graphs are included, these "representative" subgraphs may in fact encode anomalous patterns. As a result, the model's explanation capability becomes unreliable, and the anomaly scoring mechanism may assign low scores to anomalies that resemble the learned subgraph patterns. These limitations highlight a fundamental brittleness in current UGAD paradigms: they are not robust to label noise or data contamination, which are common in real-world applications.

To address this challenge, we propose DeNoise, a novel framework for robust UGAD explicitly designed for training sets that may contain a nontrivial fraction of anomalous samples. DeNoise learns noise-resistant embeddings via a min–max (adversarial) objective that jointly optimizes a graph-level encoder together with attribute and structure decoders. We first build a reconstruction model to capture the latent distribution shared across graphs; its encoder also acts as a discriminator to perform a preliminary separation of likely normal versus anomalous graphs. Building on this, we introduce an encoder anchor-alignment denoising mechanism that fuses high-information node embeddings from the separated normal graphs into all graph embeddings, improving representation quality and suppressing anomaly interference. Finally, a contrastive learning stage compacts normal graph embeddings while pushing anomalous embeddings away in the latent space. Together, these components enhance robustness to contamination and improve detection accuracy. Overall, the contributions of this paper are as follows:

- **Problem**: We provide (to our knowledge) the first systematic study of how contaminated training sets impact mainstream UGAD models.
- **Method**: We propose DeNoise, a truly unsupervised, contamination-robust UGAD model that combines adversarial reconstruction, encoder anchor-alignment denoising, and contrastive separation, removing the clean-set assumption that limits prior work.
- **Experiments**: We conducted extensive experiments on eight real-world graph datasets with varying levels of noise to demonstrate that DeNoise effectively learns robust and reliable graph-level representations, consistently achieving state-of-the-art (SOTA) performance across all benchmarks.

II. RELATED WORK

A. Graph-level Anomaly Detection

In recent years, UGAD has emerged as a pivotal and rapidly advancing topic within the GNN community [19]–[21]. Nine representative baselines, including OCGIN [13], OCGTL [12], GLocalKD [16], GOOD-D [22], SIGNET [18], HIMNet [23], CVTGAD [17], MUSE [15], and GLADPro [14], collectively establish a dominant technical paradigm: "pre-training on normal data → anomaly scoring." In this pipeline, a GNN encoder first learns self-supervised or contrastive representations exclusively from normal graphs, after which anomalies are quantified via reconstruction error, cross-view consistency, or knowledge-distillation residuals, with AUROC serving as the primary evaluation metric. Although these methods differ in their regularization terms, memory modules, hyperbolic mutual information, or multi-view statistical strategies, they all tacitly assume that the training set is absolutely uncontaminated. Consequently, the current notion of "unsupervised" is effectively semi-supervised: models are trained solely on normal graphs, while anomalous graphs are encountered only at test time. Once anomalous graphs infiltrate the training set, all baselines exhibit a precipitous performance drop, revealing a fundamental lack of robustness.

B. Graph Contrastive Learning

Graph Contrastive Learning (GCL), a key branch of graph-based self-supervised learning, has gained remarkable advantages in unsupervised graph-representation tasks by maximizing the mutual information between semantically consistent samples across different views. In recent years, GCL has been rapidly adapted to graph-level anomaly detection, giving rise to several representative innovations: GLocalKD [16] was the first to integrate a global–local knowledge-distillation mechanism into GCL, enabling the model to capture global anomaly patterns while retaining sensitivity to local structural changes; GOOD-D [22] proposed a hierarchical contrastive framework that leverages multi-level semantic cues for unsupervised out-of-distribution (OOD) detection; SIGNET [18] pioneered the introduction of a hypergraph view into GCL and, by maximizing the mutual information between bottleneck subgraphs in dual views, significantly improved the interpretability of graph-level anomaly detection; finally, CVTGAD [17] embedded a cross-view Transformer into GCL, using multi-view interactions to substantially enhance detection stability in unsupervised settings.

C. Graph Data Augmentation

Graph data augmentation aims to enlarge the effective support of the training distribution via controlled perturbations on either topology or node attributes, thereby mitigating overfitting and enhancing robustness to unobserved distributions. Recent studies pursue two main avenues: (1) stochastic or adversarial perturbation, DropEdge [24] randomly removes edges to reduce message redundancy, while FLAG [25] injects adversarial noise into node features to craft hard negatives and strengthen feature invariance; (2) semantic interpolation,

M-Mixup [26] convexly combines representations of graphs from distinct classes to generate hybrid-semantic samples, and SMART [27] further proposes a differentiable interpolation scheme that reconciles structural heterogeneity when fusing graphs.

III. PRELIMINARIES

A. Notations

A graph is denoted by the tuple $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \boldsymbol{X})$, where $\mathcal{V} = \{v_1, \ldots, v_n\}$ is the set of n nodes, $\mathcal{E} \subseteq \mathcal{V} \times \mathcal{V}$ is the set of edges, and $\boldsymbol{X} = \left[X_1^\top; \ldots; X_n^\top \right] \in \mathbb{R}^{n \times d}$ is the node feature matrix in which the i-th row $X_i \in \mathbb{R}^d$ represents the d-dimensional attributes of node v_i. The topological structure of $\mathcal{G}$ is encoded by an adjacency matrix $\boldsymbol{A} \in \{0,1\}^{n \times n}$ with $\boldsymbol{A}_{i,j} = 1$ if $(v_i, v_j) \in \mathcal{E}$ and $\boldsymbol{A}_{i,j} = 0$ otherwise.

B. Problem Definition

In the task of unsupervised anomaly detection, the objective is to learn an anomaly scoring function $\Phi : \mathbb{G} \to \mathbb{R}$, which assigns a numerical score to each graph reflecting its likelihood of being anomalous. These scores are subsequently used to detect graphs that exhibit significant deviations from typical patterns.

However, in the existing mainstream unsupervised anomaly detection settings, it is commonly assumed that all samples in the training sample set $\mathbb{G} = \{\mathcal{G}_1, \ldots, \mathcal{G}_m\}$ are normal samples. This assumption enables the model to more effectively learn the features and patterns of normal data, so that during testing, samples whose behavior deviates significantly from the learned normal patterns receive higher anomaly scores. While this approach can be effective under ideal conditions, real-world datasets often contain anomalous samples in the training set. As a result, the model may inadvertently learn representations that incorporate anomalous patterns, leading to a significant degradation in detection performance.

To address this practical limitation, this paper investigates unsupervised anomaly detection in settings where the training data may be contaminated with anomalies. Specifically, we introduce $\beta \cdot m$ anomalous samples into the original graph set $\mathbb{G}$, assumed to consist solely of normal graphs, to construct a new set $\mathbb{G}'$. In this target scenario, our goal is to develop an interference-resistant unsupervised anomaly detection model $F(\boldsymbol{X}, \boldsymbol{A})$ that can learn a robust anomaly scoring function capable of accurately evaluating anomaly scores under varying noise intensities (e.g., $\beta = 0.1, 0.2, 0.3$).

IV. THE PROPOSED DENOISE MODEL

This section presents DeNoise, a robust graph-based anomaly detection framework, whose overall architecture is illustrated in Figure 2. DeNoise employs an adversarial training paradigm to jointly optimize three components: the graph-level latent encoder E_G, the attribute decoder D_F, and the structure decoder D_S. The adversarial objective guides E_G to learn high-quality embeddings that are resilient to the noise introduced by anomalous samples.

The framework begins by constructing a reconstruction model that leverages both structural and attribute information to capture the shared latent distribution among samples. Exploiting the imbalance between normal and anomalous samples, DeNoise performs an initial separation of graphs based on a majority consensus principle: graphs that closely resemble the majority are identified as normal, while those that deviate significantly are flagged as potentially anomalous.

To enhance the quality of latent representations and suppress anomalous interference, the framework extracts representative node embeddings from normal graphs and integrates them with the latent embeddings of each graph. Additionally, DeNoise incorporates a contrastive learning strategy, using sampled normal and anomalous graphs to refine the latent space: normal graphs are encouraged to form tight clusters, while anomalous graphs are pushed farther away. This design ensures that, even when the training set contains both normal and anomalous graphs, DeNoise can maintain low reconstruction errors for normal samples and significantly higher errors for anomalous ones, thereby enabling reliable anomaly detection.

A. Constructing the Discriminator and Reconstruction Model

In Step 1 of Figure 2, we construct a reconstruction model whose primary objective is twofold: (1) to train a discriminator that provides a reliable foundation for graph embedding representations used in subsequent denoising, and (2) to serve as a critical reconstruction reference for anomaly assessment in the third phase. Drawing inspiration from self-supervised learning paradigms, this stage enhances the model's robustness to variations in graph structure by applying random perturbations to the input graph, such as randomly dropping a subset of edges. This strategy improves the model's ability to capture patterns characteristic of normal graphs.

In particular, the reconstruction model employs a GNN encoder to map the perturbed graph $\mathcal{G}' = (\boldsymbol{X}, \boldsymbol{A}')$ into a low-dimensional node embedding $Z_{node} \in \mathbb{R}^{n \times d}$. This embedding incorporates both node attribute information and high-order structural dependencies. The original graph's adjacency matrix and node feature matrix are then reconstructed through two separate decoders: structural reconstruction is performed via an inner product followed by a sigmoid activation function to predict edge existence probabilities, while feature reconstruction is achieved using graph convolutional layers to recover node attributes. The process is formalized as:

$$
\begin{aligned}
\tilde{\boldsymbol{A}} &= \text{Perturb}(A), \ \ Z_{node} = \text{E}_{\text{G}}(\tilde{A}, X) \\
\hat{\boldsymbol{A}} &= \sigma\left(HH^T\right), \ H = \text{D}_{\text{S}}(Z_{node}), \ \hat{X} = \text{D}_{\text{F}}(Z_{node})
\end{aligned}
\tag{1}
$$

where $\text{Perturb}(\cdot)$ denotes the stochastic edge dropout operation, E_{G} represents the graph neural network encoder, D_{S} and D_{F} denote the structure and feature decoders, respectively, σ is the sigmoid activation function, and H is the intermediate latent representation output by the structure decoder.

To effectively identify behavioral patterns within the training set, the reconstruction model is optimized by minimizing losses associated with both node features and adjacency matrices. For node features, we adopt the cosine similarity loss as introduced in [15], [28], which ensures the reconstructed
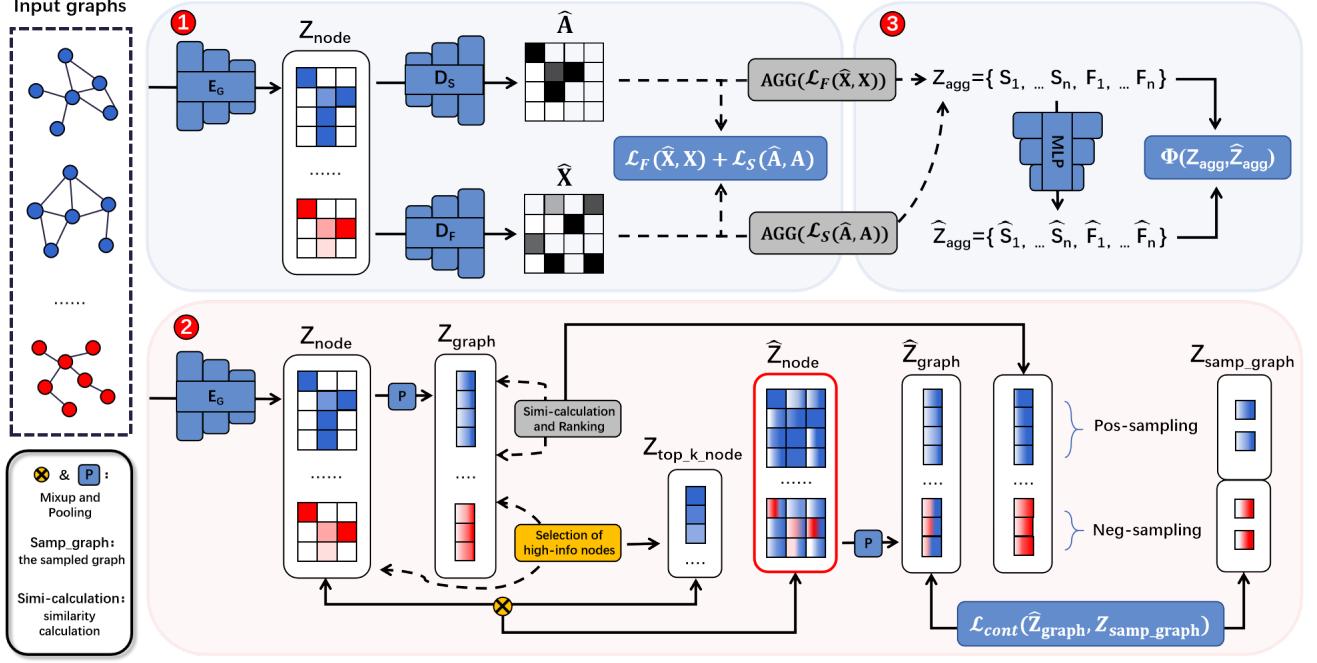
Fig. 2. The DeNoise framework comprises three key components: (1) the establishment of a discriminator and a reconstruction model, (2) a noise reduction phase applied to the encoder, and (3) a multidimensional anomaly assessment module. In the model illustration, node color intensity intuitively reflects the amount of information contained in each node: blue represents nodes from normal graphs, red indicates anomalous nodes, and white signifies nodes with low information content. The terms S_i and F_i denote the structural and attribute reconstruction errors, respectively, which are aggregated using different functions.

features align closely with the original ones. For adjacency matrix reconstruction, we use the binary cross-entropy (BCE) loss following [15], [29]. These losses are defined as:

$$
\mathcal{L}_F(X, \hat{X}) = \frac{1}{n} \sum_{i=1}^{n} \left(1 - \frac{X_i^\top \hat{X}_i}{\|X_i\|\|\hat{X}_i\|} \right), \ n = |V|
$$

$$
\mathcal{L}_S(A, \hat{A}) = -\frac{1}{n^2} \sum_{i,j} \Big(\omega A_{i,j} \log \hat{A}_{i,j} +
$$

$$
(1 - A_{i,j}) \log(1 - \hat{A}_{i,j}) \Big), \quad (2)
$$

$$
\omega = \left(\frac{\sum_{i,j} A_{i,j}}{\sum_{i,j}(1 - A_{i,j})} \right)^\tau, \ n = |V|
$$

where ω is a weighting coefficient designed to mitigate the class imbalance inherent in sparse adjacency matrices, and τ is a hyperparameter that scales the contribution of this weight. The functions $\mathcal{L}_F(\cdot, \cdot)$ and $\mathcal{L}_S(\cdot, \cdot)$ represent the reconstruction losses for node features and adjacency matrices, respectively.

Importantly, at this stage, the encoder also functions as a discriminator. Specifically, a graph-level similarity mechanism is employed to compare the embedding of the current graph with those of other graphs in the training set. If the embedding of a graph deviates significantly from the majority, it is preliminarily identified as a potential anomaly; otherwise, it is regarded as normal. The process is formalized as:

$$
Z_{graph} = \text{Readout}(Z_{node}) = \frac{1}{n} \sum_{i=1}^{n} z_i^{node}, \ n = |V|
$$

$$
\eta(z_{graph}) = \frac{1}{|\mathcal{D}|} \sum_{graph' \in \mathcal{D}} \text{sim}\left(z_{graph}, z_{graph'}\right)
$$

$$
= \frac{1}{|\mathcal{D}|} \sum_{graph' \in \mathcal{D}} \frac{(z_{graph})^T z_{graph'}}{\|z_{graph}\| \|z_{graph'}\|} \quad (3)
$$

$$
\hat{Y}_G = \begin{cases} 0, & \text{if } \eta(z_{graph}) \geq \tau_\alpha \\ 1, & \text{if } \eta(z_{graph}) < \tau_\alpha \end{cases},
$$

$$
\tau_\alpha = \text{Quantile}_\alpha \left(\left\{ \eta\left(z_{graph}^k\right) \right\}_{k=1}^{M} \right) \quad (4)
$$

where $\text{Readout}(\cdot)$ aggregates node-level embeddings to a graph-level representation, $\eta(\cdot)$ measures the similarity between a graph and the rest of the dataset $\mathcal{D}$ in the embedding space, and $\text{sim}(\cdot, \cdot)$ denotes cosine similarity. A higher value of η indicates a greater likelihood of the graph being normal. $\text{Quantile}_\alpha(\cdot)$ computes the α-quantile over the similarity scores, and $\hat{Y}_G$ is the preliminary anomaly label assigned to graph G (0: normal, 1: anomalous).

Finally, the reconstruction errors computed during this phase provide essential information for the anomaly scoring function defined in Stage 3. By integrating both structural and attribute reconstruction errors, the model produces a unified anomaly score, enabling accurate and reliable GLAD.

B. Encoder Anchor-Alignment Denoising

In Step 2 of Figure 2, the objective is to guide the embeddings generated by the encoder to align more closely with those of normal graphs, while distancing them from anomalous graph embeddings. The central motivation is to enhance the model's ability to capture the characteristics of normal graphs and to mitigate the interference caused by anomalous samples, thereby improving anomaly detection accuracy.

we begin by selecting high-information node embeddings from normal graphs identified by the discriminator in Phase 1 (as indicated by "Selection of high-info nodes" in Figure 2). These selected embeddings are then integrated into the node embeddings of all graphs. This strategy reinforces feature representations that are characteristic of normal graphs, while suppressing the influence of anomalous features. The highly informative embeddings, which resemble typical node features from normal graphs, help improve overall embedding quality and bring the representations of normal graphs closer to the expected distribution. This also encourages embeddings of anomalous graphs to converge toward the normal distribution, thereby reducing their deviation in feature space.

Let $\mathcal{G}_{\text{norm}} = \left\{ G_1, G_2, \ldots, G_N \mid \hat{Y}_{G_i} = 0 \right\}$ denote the set of graphs determined as normal by the discriminator. The high-information nodes are selected through the following steps:

$$I\left(z_{j,G_i}^{node}\right) = \frac{1}{|\mathcal{G}_{\text{norm}}|} \sum_{G' \in \mathcal{G}_{\text{norm}}} \frac{(z_{j,G_i}^{node})^T Z_{G'}^{gragh}}{\left\| z_{j,G_i}^{node} \right\| \left\| Z_{G'}^{gragh} \right\|}$$

$$Z_{top\_k\_node} = \left\{ z_{j,G_i}^{node} \mid I\left(z_{j,G_i}^{node}\right) \in \text{Top}_k\left(I\left(z_{j,G_i}^{node}\right)\right) \right\} \quad (5)$$

where z_{j,G_i}^{node} is the embedding of the j-th node in graph G_i, $Z_{G'}^{\text{graph}}$ is the graph-level embedding of G', and $I(\cdot)$ computes the information content of a node. The function $\text{Top}\,k(I(\cdot))$ selects the top k node embeddings with the highest information scores, forming the set $Z_{top\_k\_node}$.

To address potential dimensional mismatches during node embedding fusion, we adopt an adaptive strategy that performs a mixup operation. In particular, we compute a transformation matrix T based on feature similarity to align the dimensions of the embeddings, followed by a linear interpolation:

$$T = Z_{node} Z_{top\_k\_node}^T, \quad \tilde{Z}_{top\_k\_node} = T Z_{top\_k\_node}$$
$$\hat{Z}_{node} = \lambda Z_{node} + (1 - \lambda) \tilde{Z}_{top\_k\_node} \quad (6)$$

where λ is the fusion coefficient, used to control the intensity of the integration.

At the graph level, to address the issue of imbalanced data distribution between anomalous and normal graphs, we employ a method of equal sampling from regions of high and low similarity between graphs, labeling the corresponding samples as normal and anomalous graphs, respectively. Through contrastive learning, we optimize the embeddings that incorporate node information from normal graphs. This strategy is intended to guide the embeddings generated by the encoder to more closely align with those of normal graphs, while also pulling the embeddings of anomalous graphs into the feature space of normal graphs, thereby reducing the anomalous behavior of anomalous graphs in the feature space

and achieving a denoising effect. This process can be represented as follows:

$$\mathcal{G}_{\text{norm\_sample}} = \{G_1^+, G_2^+, \ldots, G_K^+ \mid sim_{gg'} > \tau_{\beta_1}\},$$
$$\mathcal{G}_{\text{anom\_sample}} = \{G_1^-, G_2^-, \ldots, G_K^- \mid sim_{gg'} < \tau_{(1-\beta_2)}\}, \quad (7)$$
$$sim_{gg'} = \text{sim}\left(z_{graph}, z_{graph'}\right)$$

$$\mathcal{L}_{\text{cont}} = \frac{1}{M} \sum_{i=1}^{M} \log \frac{\ell_i^+}{\ell_i^+ + \ell_i^-},$$
$$\ell_i^+ = \sum_{j=1}^{K} \exp\left(\text{sim}\left(\hat{Z}_i^{graph}, Z_j^{graph^+}\right)/\tau\right), \quad (8)$$
$$\ell_i^- = \sum_{j=1}^{K} \exp\left(\text{sim}\left(\hat{Z}_i^{graph}, Z_j^{graph^-}\right)/\tau\right)$$

where τ_{β_i} is consistent with Equation 4, representing the β_i-upper quantile of the empirical distribution of η calculated on dataset $\mathcal{D}$, where $\hat{Z}_i^{graph}$ represents the graph-level embedding representation of the i-th graph that incorporates high-quality node information from normal graphs, and $Z_j^{graph^+}$ is the graph-level embedding representation of the sampled normal graph G_j^+, $Z_j^{graph^-}$ is the graph-level embedding representation of the sampled abnormal graph G_j^-, and τ is the temperature hyperparameter.

By employing these two strategies at different hierarchical levels, this study ensures that even when the training set is contaminated with varying degrees of noise, the model can still maintain a lower reconstruction error for normal graphs and a higher reconstruction error for anomalous graphs during the inference process, thereby effectively identifying anomalous graphs.

C. Adversarial Alternating Training Strategy for Stages One and Two

In our work, we propose an adversarial alternating training strategy to optimize graph neural network models, which is divided into two stages aimed at ensuring the effective reconstruction of normal graphs and mitigating the negative impact of anomalous graphs on model performance.

During the initial training phase, our primary objective is to minimize the overall reconstruction loss of the training data. Although the training set may contain anomalous data, which could potentially lead the model to erroneously learn these anomalous behaviors, the model can still learn and reconstruct the dominant behaviors of normal graphs due to the significantly larger quantity of normal graphs compared to anomalous ones. To this end, we define two types of reconstruction losses (the specific details are shown in Equation 2): $\mathcal{L}_F(\cdot, \cdot)$ for node features and $\mathcal{L}_S(\cdot, \cdot)$ for adjacency matrices, with the optimization target for this stage being as follows:

$$\min_{\theta} \mathcal{L}_{\text{recon}} = \mathcal{L}_F\left(X, \hat{X}\right) + \mathcal{L}_S(A, \hat{A}) \quad (9)$$

where θ represents the parameters of the encoder E_G and decoders D_F and D_S, X and $\hat{X}$ denote the original and reconstructed node feature matrices, respectively, and A and $\hat{A}$ represent the original and reconstructed adjacency matrices.

The goal of the second stage is to filter out any anomalous data behaviors that may have been learned during the first phase. We integrate high-information node embeddings from normal graphs identified by the discriminator into the graph representations produced by the encoder. This process aims to make the fused embeddings closer to the embeddings of normal graphs, thereby pushing the behaviors of anomalous graphs further away in the feature space, causing them to be "forgotten" during the reconstruction process. This can be achieved through the following optimization objective (the details of the contrastive loss are shown in Equations 7, 8):

$$\max_{\phi} \mathcal{L}_{\text{sim}} = \mathcal{L}_{\text{cont}} \left(\hat{Z}^{graph}, Z^{graph^+}, Z^{graph^-} \right) \quad (10)$$

where ϕ represents the parameters of the encoder E_G, $\hat{Z}^{graph}$ denotes the graph embeddings that have incorporated high-information node embeddings from normal graphs, Z^{graph^+} represents the graph embeddings of sampled normal graphs, and Z^{graph^-} represents the graph embeddings of sampled anomalous graphs.

We integrate the optimization objectives of these two stages into a single min-max problem to implement adversarial alternating training:

$$\min_{\theta} \max_{\phi} \left\{ \mathcal{L}_{\text{recon}} + w \mathcal{L}_{\text{sim}} \right\} \quad (11)$$

where w is a hyperparameter used to balance the importance of the two objectives. The first objective is to minimize the reconstruction loss, ensuring that the model can capture the dominant behavior patterns of the majority of normal data. The second objective is to maximize the similarity with embeddings of normal graphs while minimizing the similarity with embeddings of anomalous graphs, thereby filtering out anomalous behaviors.

Through this adversarial alternating training strategy, the model preserves the reconstruction quality of normal graph data while effectively filtering out the influence of anomalous graphs on the reconstruction process, thereby achieving a noise-resistant effect.

D. Multidimensional Anomaly Scoring Module

During the initial two stages, we have meticulously enhanced the model's capacity for reconstruction and its efficacy in discerning anomalous activities. In the phase designated as Step 3 in Figure 2, our ambition is to perform a thorough evaluation of the graph's reconstruction discrepancies through a multidimensional aggregation technique, thereby facilitating more accurate anomaly detection.

Concretely, we harness the reconstruction errors acquired in the first stage, as delineated in Equation 2, and construct a multidimensional reconstruction error vector Z_{agg} using a variety of aggregation methodologies. This procedure is inspired by the work presented in [15], wherein we aggregate the node reconstruction errors and the adjacency matrix reconstruction errors using both mean and standard deviation methods to form a four-dimensional vector, represented as follows:

$$Z_{\text{agg}} = \{ \text{mean}(\mathcal{L}_F(X, \hat{X})), \ \text{mean}(\mathcal{L}_S(A, \hat{A})),$$
$$\text{std}(\mathcal{L}_F(X, \hat{X})), \ \text{std}(\mathcal{L}_S(A, \hat{A})) \} \quad (12)$$

Then we employ the constructed reconstruction error vector Z_{agg} to calculate the reconstruction error and utilize the values of the multidimensional reconstruction error vector as anomaly scores. The specific process is outlined below:

$$\hat{Z}_{agg} = \text{MLP}\left(Z_{agg} \right)$$
$$\text{score}(G_i) = \Phi\left(\hat{Z}_{agg}, Z_{agg} \right)$$
$$= \frac{1}{n} \sum_{i=1}^{n} \left(\left(\hat{Z}_{agg}^{(i)} - Z_{agg}^{(i)} \right)^2 / \sigma_i \right), \quad (13)$$
$$\sigma_j^2 = \frac{1}{N} \sum_{i=1}^{N} \left(Z_{agg,j}^{(i)} - \mu_j \right)^2, \mu_j = \frac{1}{N} \sum_{i=1}^{N} Z_{agg,j}^{(i)}$$

where μ_j represents the mean of the j-th feature within the aggregated reconstruction error vector Z_{agg}, and σ_j denotes the corresponding standard deviation, the anomaly score for the i-th sample is denoted by $\text{score}(G_i)$. The total number of samples is given by N, and the dimensionality of the reconstruction error vector is denoted by n. Additionally, $Z_{agg}^{(i)}$ and $\hat{Z}_{agg}^{(i)}$ correspond to the i-th element of the vectors Z_{agg} and $\hat{Z}_{agg}$, respectively.

By conducting a comprehensive assessment of the reconstruction errors from multiple perspectives, the model is enabled to consider the data holistically, thereby significantly enhancing the performance of anomaly detection.

E. Complexity Analysis

We analyze the time complexity of DeNoise with respect to the number of graphs N, average nodes per graph n, average edges m, and hidden dimension d. The GNN encoder performs L-layer message passing over nodes and edges, incurring $\mathcal{O}\left(N(Lmd + Lnd + nd^2)\right)$. Attribute and structure decoders reconstruct node features and adjacency via inner-product and GCN-style operations, adding $\mathcal{O}\left(N(nd^2 + n^2d)\right)$. Anchor-alignment denoising selects top-k high-information nodes and applies mixup across all graphs, contributing $\mathcal{O}\left(N(knd + k^2d)\right)$. Contrastive learning samples K positive/negative graph pairs and computes Info-NCE loss, yielding $\mathcal{O}(NKd)$. Multidimensional anomaly scoring aggregates four reconstruction statistics and feeds them through a small MLP, costing $\mathcal{O}(N \cdot 4d)$. Neglecting the smaller terms, the overall per-epoch complexity is $\mathcal{O}(NL(md + nd + d^2) + Nn^2d + NKd)$.

V. EXPERIMENTS

In this section, we systematically evaluate the performance of DeNoise on eight real-world datasets and under noisy scenarios through extensive experimental studies. We aim to address the following research questions:

- RQ1: How effective is DeNoise under different levels of noisy scenarios?
- RQ2: What are the contributions of the core designs in DeNoise?
- RQ3: How do each of the core hyperparameters influence the performance of DeNoise?
- RQ4: How does the embedding of the DeNoise model adaptively adjust during the denoising process?

TABLE I
THE STATISTICS OF DATASETS.

Dataset	COX2	DHFR	AIDS	PROTEINS_full	ENZYMES	DD	IMDB-BINARY	PROTEINS
Graphs	467	756	2000	1113	600	1178	1000	1113
Avg. Nodes	41.2	42.4	15.7	39.1	32.6	284.3	19.8	39.1
Avg. Edges	86.9	89.1	32.4	145.6	124.3	1431.3	193.1	145.6
Node Attr.	35	53	38	3	3	89	0	3

A. Experimental Setup

1) Datasets: In this study, we selected eight benchmark datasets [30] spanning biological molecules, enzyme classification, disease-related networks, and social networks to comprehensively evaluate the anomaly detection capabilities of our model. Detailed descriptions of each dataset are presented in Table 1.

2) Training and evaluation: In accordance with the methodology outlined in [16], [22], this study defines the minority or labeled anomalous samples as the anomaly class, while the remaining samples are categorized as the normal class. On this basis, the normal graph samples are divided into training, validation, and test sets in the proportions of 80%, 10%, and 10%, respectively. Meanwhile, from the anomalous graph samples, 5% are randomly sampled for the validation set and another 5% for the test set. For the noise scenario, in the training set, anomalous samples are randomly sampled at the proportions of 10%, 20%, and 30% of the number of normal graph samples and added to the training set to simulate varying degrees of noise interference. To ensure the stability and reliability of the results, for each noise proportion scenario, five independent trials are conducted, each with different data splits and model initializations. Finally, the model performance is comprehensively and objectively evaluated based on the average AUC (Area Under the Curve) and its standard deviation.

3) Baselines: In this study, to comprehensively evaluate the performance of our proposed model, we carefully selected nine representative baseline models for comparative analysis. These baseline models include GLADPro [14], an interpretable graph-level anomaly detection method based on prototype learning and the information bottleneck principle; MUSE [15], a graph-level anomaly detection method based on multi-dimensional aggregation reconstruction error; CVTGAD [17], which employs a simplified Transformer architecture with cross-view attention for graph-level anomaly detection; HimNet [23], a graph-level anomaly detection method implemented through hierarchical memory networks; SIGNET [18], a self-explaining graph-level anomaly detection method based on multi-view subgraph information bottleneck; GOOD-D [22], an out-of-distribution (OOD) detection method using hierarchical contrastive learning; GLocalKD [16], a graph-level anomaly detection method through global and local knowledge distillation; OCGTL [12], a graph anomaly detection method combining deep one-class classification and graph transformation learning; and OCGIN [13], a one-class graph anomaly detection method by optimizing the Support
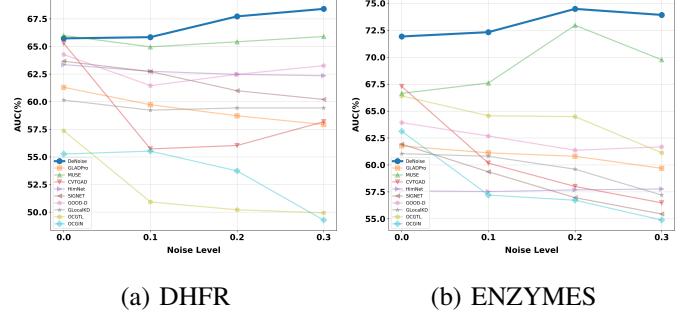

(a) DHFR   (b) ENZYMES

Fig. 3. Performance comparison under different noise levels on (a) DHFR and (b) ENZYMES datasets. The AUC scores (%) of various baseline methods are plotted against increasing levels of injected noise. The proposed method (blue line) consistently outperforms competing approaches and shows strong robustness to noise, while most baselines experience performance degradation as noise levels increase.

Vector Data Description (SVDD) objective. Through comparative analysis with these baseline models, we are able to more comprehensively assess the performance of our proposed model in the task of graph-level anomaly detection.

4) Implementation Details: The DeNoise model was implemented using PyTorch Geometric (PyG) version 2.6.1 and PyTorch version 2.1.1. All experiments were conducted on a GeForce RTX 3090 GPU with 24GB of memory.

B. Performance Comparison (RQ1)

To evaluate the noise resistance of DeNoise, we conducted experiments in which the proportion of anomalous samples in the training set was systematically varied, with β ranging from 0.1 to 0.3. The results, summarized in Table 2, yield the following key observations: 1) under the condition where the training set contains anomalous samples, DeNoise achieved SOTA performance on 8 datasets. As the number of anomalous samples increased, the performance of some baseline methods (e.g., SIGNET) deteriorated sharply, while the performance gap between DeNoise and these baseline methods widened. This highlights DeNoise's robust noise resistance and its ability to achieve true unsupervised learning. 2) under the original assumption (i.e., the training set contains only normal samples), DeNoise outperformed other baseline methods on 6 datasets and ranked second on the remaining 2 datasets. This consistent performance highlights the effectiveness of the encoder's anchor alignment denoising strategy, which not only suppresses the influence of anomalous samples but also enhances the quality of embeddings for normal graphs. We attribute this robustness to DeNoise's ability to integrate high-
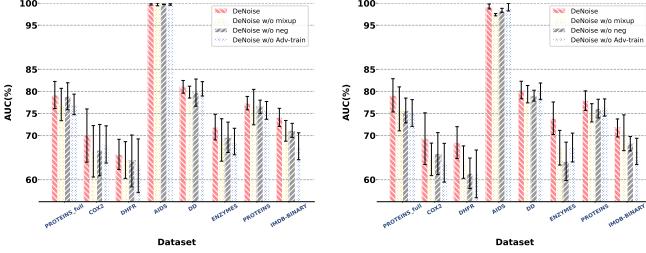
TABLE II
PERFORMANCE COMPARISON UNDER DIFFERENT NOISE CONDITIONS. THE BEST AND SECOND-BEST MODELS ARE HIGHLIGHTED IN BOLD AND UNDERLINED, RESPECTIVELY.

Method	COX2				DHFR			
	$\beta = 0.0$	$\beta = 0.1$	$\beta = 0.2$	$\beta = 0.3$	$\beta = 0.0$	$\beta = 0.1$	$\beta = 0.2$	$\beta = 0.3$
OCGIN	57.37 ± 10.57	56.16 ± 11.60	50.81 ± 9.58	57.98 ± 12.14	55.26 ± 12.08	55.52 ± 12.66	53.73 ± 11.09	49.28 ± 9.50
OCGTL	57.17 ± 10.20	54.95 ± 7.79	54.95 ± 10.87	54.75 ± 11.92	57.36 ± 16.84	50.92 ± 15.11	50.21 ± 15.47	49.93 ± 14.69
GLocalKD	63.13 ± 12.22	58.89 ± 15.62	58.48 ± 15.36	55.96 ± 18.77	60.14 ± 8.44	59.23 ± 7.79	59.43 ± 6.99	59.43 ± 7.70
GOOD-D	62.42 ± 8.44	60.00 ± 7.58	60.71 ± 7.36	60.61 ± 4.64	64.27 ± 8.21	61.45 ± 5.70	62.47 ± 5.88	63.26 ± 7.12
SIGNET	**71.01 ± 6.86**	<u>68.69 ± 7.97</u>	66.97 ± 8.82	64.95 ± 10.80	63.66 ± 12.51	62.72 ± 4.52	60.99 ± 9.29	60.20 ± 5.60
HimNet	69.49 ± 9.75	68.48 ± 10.49	67.27 ± 10.07	67.78 ± 11.39	63.35 ± 11.29	62.75 ± 11.21	62.47 ± 11.07	62.35 ± 11.05
CVTGAD	64.02 ± 13.07	63.40 ± 12.17	63.10 ± 12.20	61.60 ± 12.29	65.28 ± 9.87	55.72 ± 0.34	56.03 ± 0.57	58.18 ± 1.93
MUSE	67.47 ± 12.20	67.47 ± 7.09	<u>67.68 ± 11.88</u>	<u>67.88 ± 7.52</u>	**65.99 ± 6.13**	<u>64.96 ± 7.86</u>	<u>65.42 ± 5.65</u>	<u>65.90 ± 10.14</u>
GLADPro	66.87 ± 13.33	62.02 ± 8.82	61.11 ± 9.39	59.90 ± 8.42	61.30 ± 8.65	59.72 ± 5.24	58.72 ± 6.60	57.93 ± 3.69
DeNoise	<u>70.00 ± 6.03</u>	**68.99 ± 5.24**	**68.38 ± 15.76**	**69.29 ± 8.86**	<u>65.73 ± 3.42</u>	**65.84 ± 6.75**	**67.72 ± 5.95**	**68.40 ± 8.61**

Method	AIDS				PROTEINS_full			
	$\beta = 0.0$	$\beta = 0.1$	$\beta = 0.2$	$\beta = 0.3$	$\beta = 0.0$	$\beta = 0.1$	$\beta = 0.2$	$\beta = 0.3$
OCGIN	92.02 ± 2.52	91.38 ± 2.51	90.33 ± 3.16	84.32 ± 0.96	64.50 ± 5.53	63.41 ± 5.74	63.30 ± 4.79	62.57 ± 6.20
OCGTL	**99.94 ± 0.10**	99.63 ± 0.44	97.94 ± 2.26	89.01 ± 4.86	67.66 ± 3.45	60.30 ± 5.72	59.45 ± 7.78	59.06 ± 6.92
GLocalKD	97.76 ± 0.73	97.39 ± 0.26	96.60 ± 0.15	95.61 ± 0.82	69.51 ± 7.01	69.19 ± 6.84	68.18 ± 7.11	68.90 ± 7.41
GOOD-D	97.67 ± 0.67	96.69 ± 1.91	96.95 ± 2.36	95.60 ± 3.02	74.06 ± 1.75	71.85 ± 2.02	71.64 ± 1.64	70.13 ± 1.70
SIGNET	96.46 ± 0.84	69.59 ± 6.93	64.97 ± 7.13	64.12 ± 6.52	72.18 ± 1.12	70.91 ± 1.42	67.93 ± 1.90	66.05 ± 8.00
HimNet	99.74 ± 0.24	99.53 ± 0.31	99.41 ± 0.15	98.32 ± 0.53	73.44 ± 4.53	73.37 ± 4.57	73.32 ± 4.65	73.37 ± 4.60
CVTGAD	99.34 ± 0.90	<u>99.68 ± 0.05</u>	99.27 ± 0.11	98.81 ± 1.15	75.10 ± 3.51	74.58 ± 4.06	73.93 ± 4.05	73.84 ± 4.57
MUSE	99.71 ± 0.27	99.52 ± 4.21	<u>99.48 ± 2.44</u>	<u>99.26 ± 0.70</u>	77.09 ± 4.22	<u>76.96 ± 4.18</u>	<u>76.27 ± 3.91</u>	<u>77.13 ± 3.64</u>
GLADPro	97.71 ± 1.84	97.44 ± 2.07	96.46 ± 1.86	95.49 ± 2.76	<u>77.16 ± 4.40</u>	75.57 ± 6.32	74.42 ± 5.93	74.60 ± 4.35
DeNoise	<u>99.81 ± 0.22</u>	**99.72 ± 0.20**	**99.68 ± 0.28**	**99.87 ± 0.17**	**79.19 ± 3.08**	**79.62 ± 2.42**	**79.11 ± 4.04**	**79.12 ± 3.77**

Method	ENZYMES				DD			
	$\beta = 0.0$	$\beta = 0.1$	$\beta = 0.2$	$\beta = 0.3$	$\beta = 0.0$	$\beta = 0.1$	$\beta = 0.2$	$\beta = 0.3$
OCGIN	63.12 ± 5.97	57.20 ± 6.93	56.72 ± 5.54	54.88 ± 6.44	69.49 ± 6.02	69.63 ± 6.26	69.40 ± 6.37	69.86 ± 6.06
OCGTL	66.40 ± 5.33	64.56 ± 5.97	64.48 ± 3.75	61.12 ± 4.47	79.45 ± 5.03	79.10 ± 4.11	78.93 ± 4.09	78.77 ± 4.31
GLocalKD	61.04 ± 5.02	60.80 ± 8.84	59.60 ± 7.16	57.20 ± 8.03	80.10 ± 3.55	<u>80.09 ± 4.55</u>	<u>80.08 ± 4.50</u>	<u>80.01 ± 3.54</u>
GOOD-D	63.92 ± 6.70	62.68 ± 8.49	61.36 ± 9.34	61.68 ± 5.85	74.50 ± 2.61	72.06 ± 2.41	72.31 ± 3.64	68.57 ± 1.02
SIGNET	61.92 ± 9.22	59.36 ± 8.31	56.96 ± 7.14	55.44 ± 7.83	71.13 ± 4.06	66.40 ± 2.37	60.61 ± 4.64	59.62 ± 6.46
HimNet	57.60 ± 7.85	57.52 ± 8.72	57.68 ± 8.71	57.76 ± 7.72	<u>80.59 ± 3.97</u>	79.40 ± 2.77	79.39 ± 2.88	79.49 ± 2.89
CVTGAD	<u>67.28 ± 8.38</u>	60.16 ± 8.59	58.00 ± 5.18	56.48 ± 6.89	79.70 ± 5.25	75.81 ± 6.71	72.51 ± 2.84	69.54 ± 1.7
MUSE	66.64 ± 8.90	<u>67.60 ± 8.61</u>	<u>72.96 ± 6.31</u>	<u>69.76 ± 6.34</u>	80.42 ± 1.56	79.27 ± 1.78	79.47 ± 1.83	79.79 ± 1.74
GLADPro	61.76 ± 7.54	61.12 ± 7.13	60.80 ± 6.80	59.68 ± 5.97	75.99 ± 6.37	74.30 ± 7.54	73.95 ± 5.28	73.19 ± 8.07
DeNoise	**71.92 ± 6.90**	**72.32 ± 5.33**	**74.48 ± 4.98**	**73.92 ± 4.69**	**81.05 ± 1.44**	**81.05 ± 1.23**	**81.05 ± 2.01**	**80.32 ± 1.99**

Method	IMDB-BINARY				PROTEINS			
	$\beta = 0.0$	$\beta = 0.1$	$\beta = 0.2$	$\beta = 0.3$	$\beta = 0.0$	$\beta = 0.1$	$\beta = 0.2$	$\beta = 0.3$
OCGIN	69.10 ± 7.03	63.57 ± 12.16	63.31 ± 15.97	61.21 ± 8.35	67.35 ± 3.52	64.27 ± 4.37	62.92 ± 4.26	60.93 ± 10.69
OCGTL	64.44 ± 5.46	63.34 ± 5.29	59.32 ± 9.89	58.07 ± 10.91	65.75 ± 4.49	64.19 ± 2.78	60.97 ± 4.57	57.47 ± 6.58
GLocalKD	53.50 ± 4.51	53.76 ± 5.21	53.97 ± 5.47	52.41 ± 5.05	73.21 ± 4.30	73.20 ± 4.31	73.21 ± 4.34	73.24 ± 4.27
GOOD-D	65.88 ± 6.99	65.24 ± 4.76	66.46 ± 6.27	66.40 ± 6.41	76.01 ± 0.89	72.32 ± 2.56	71.84 ± 1.86	71.96 ± 2.61
SIGNET	72.22 ± 3.71	72.08 ± 3.73	70.59 ± 6.59	69.34 ± 5.44	71.80 ± 6.80	72.16 ± 7.25	72.15 ± 7.13	71.99 ± 7.02
HimNet	64.92 ± 7.10	64.86 ± 7.01	64.76 ± 7.13	64.65 ± 6.99	75.96 ± 5.50	75.86 ± 3.50	75.76 ± 3.14	75.52 ± 5.51
CVTGAD	71.60 ± 4.57	<u>73.29 ± 0.73</u>	<u>72.77 ± 1.29</u>	68.56 ± 3.01	74.11 ± 3.33	69.12 ± 5.15	76.28 ± 0.26	75.72 ± 1.45
MUSE	<u>74.04 ± 3.58</u>	72.09 ± 3.83	71.62 ± 2.95	69.93 ± 3.27	<u>76.74 ± 1.75</u>	<u>77.33 ± 2.58</u>	<u>77.56 ± 3.29</u>	<u>77.52 ± 2.95</u>
GLADPro	73.89 ± 4.69	71.15 ± 5.39	70.97 ± 6.04	<u>70.98 ± 5.88</u>	75.13 ± 8.35	73.08 ± 5.54	72.86 ± 7.19	72.67 ± 8.23
DeNoise	**74.12 ± 2.05**	**74.98 ± 3.18**	**73.85 ± 3.56**	**71.72 ± 2.04**	**77.35 ± 1.52**	**78.24 ± 2.38**	**78.24 ± 2.31**	**77.91 ± 2.22**

information-content node features from normal graphs into the latent representations of all graphs, thereby reinforcing the learning of normal patterns and improving overall anomaly detection accuracy.

Notably, as shown in Figure 3, the DeNoise method exhibits an unusual yet favorable trend: its performance improves with increasing noise levels on certain datasets (such as DHFR and ENZYMES). In contrast, other methods generally experience a decline in performance when confronted with increasing noise. We attribute this counterintuitive phenomenon to DeNoise's ability to propagate high-information node representations from normal (i.e., positive) samples across all graphs, including anomalous ones. As the proportion of anomalous samples in the training set increases, these anomalous graphs are effectively transformed through the integration of representative features from normal graphs. This feature infusion process enhances the quality of their embeddings, thereby improving the model's generalization ability. As a result, rather than being hindered by the presence of additional anomalies, the model benefits from the increased diversity in the training data,

(a) Experiments on $\beta = 0.0$. (b) Experiments on $\beta = 0.3$.

Fig. 4. Performance comparison of DeNoise and its variants under different noise conditions. (a) Experiments conducted on clean datasets ($\beta = 0.0$). (b) Experiments conducted under noisy conditions with 30% anomalous samples in the training set ($\beta = 0.3$).

leading to a gradual improvement in detection performance under higher noise levels.

C. Ablation Study (RQ2)

To verify the contributions of the individual components and key designs in DeNoise, we conducted experiments on various variants of DeNoise, with the results shown in Figure 4. Specifically, "DeNoise w/o mixup" refers to the variant that does not integrate high-quality normal sample nodes; "DeNoise w/o neg" refers to the variant that does not use negative samples for optimization during the sampling process; "DeNoise w/o Adv-train" refers to the variant that does not perform adversarial alternating training but instead trains Equations 9 and 10 jointly. We evaluated each variant under two conditions: a clean scenario (i.e., training set contains only normal samples) and a noisy scenario with $\beta = 0.3$ (i.e., 30% anomalous samples in the training set).

From the results in Figure 4, we draw the following conclusions: 1) the importance of incorporating high-quality normal sample nodes: The performance of "DeNoise w/o mixup" in the noisy scenario is lower than that of DeNoise, indicating that relying solely on sampling operations to optimize embeddings is insufficient to achieve noise resistance, thereby significantly affecting the final anomaly detection performance. In the clean scenario, the performance of this variant is also lower than that of DeNoise, further proving that incorporating high-quality positive sample node information can effectively enhance the quality of embeddings, thereby improving anomaly detection performance. 2) the necessity of negative sample sampling: In the clean scenario, the performance of "DeNoise w/o neg" is generally better than the other two variants; however, its performance drops significantly in the noisy scenario. This phenomenon indicates that in the noisy scenario, sampling negative samples and keeping them at a distance from the learned embeddings is crucial for enhancing noise resistance. 3) the effect of adversarial training: The performance of "DeNoise w/o Adv-train" is weaker than that of DeNoise in both the clean and noisy scenarios. We posit that this phenomenon may potentially stem from the training imbalance between the first and second stages. In the first stage, the model learns the behavior patterns of both normal and abnormal samples, while the second stage optimizes on

this basis. If these two stages are combined into joint training, it may lead to the model's inability to effectively enhance the quality of embeddings, thereby affecting overall performance.

D. Parameter Study (RQ3)

a) Fusion Coefficient λ and Selection Coefficient k: In Equation (5), we select the top k nodes with the highest information content. In Equation (6), we incorporate the embeddings of these k high-information nodes into other nodes according to the fusion coefficient λ. These two strategies are generally used jointly to enhance the quality of the embeddings. To thoroughly investigate the specific impacts of these two parameters on model performance, we conducted extensive experiments under the condition of noise intensity $\beta = 0.3$.

For the selection coefficient k, we evaluated three values: 512, 256, and 128. For the fusion coefficient λ, which is sampled from a uniform distribution in the implementation, we considered five intervals: [0.8, 1.0], [0.7, 0.9], [0.4, 0.6], [0.1, 0.3], and [0.0, 1.0].

As shown in Figure 5, both hyperparameters significantly influence model performance across different datasets: For the selection coefficient k, its impact on model performance varies depending on the dataset. In datasets such as AIDS, IMDB-BINARY, and PROTEINS, the model performs better when $k = 512$. This suggests that in these datasets, selecting a larger number of high-information nodes can better preserve key information, thereby enhancing the model's embedding of graph structures. As a result, the model can more accurately capture the features of the graph in subsequent tasks, leading to better performance. However, in the COX2 dataset, the model achieves optimal performance when $k = 128$. This indicates that over-reliance on a large number of high-information nodes does not necessarily lead to significant performance improvements. The possible reason is that the graph structure characteristics of this dataset cause some redundancy among certain high-information nodes, or its key information is not entirely concentrated in the largest number of high-information nodes. It is observed that different datasets have different distributions of key information, and the appropriate k value should be selected based on the specific dataset.

The fusion coefficient λ determines the intensity of the integration of high-information node embeddings. When λ is larger, the new embedding is closer to the original features; when λ is smaller, the new embedding is closer to the high-information node embedding. The selection of the range of values for the fusion coefficient λ is relatively more complex for different datasets compared to k, and it needs to be adjusted in combination with the specific dataset and k value. Overall, the selection of the selection coefficient k and the fusion coefficient λ should take into account the structural characteristics of the dataset and the specific needs of the model to achieve the best model performance.

b) The number of samples K and the quantiles β_1 and β_2: In Equation (7), we sorted the graph embeddings based on the similarity between graphs and sampled an equal number of samples from both the high-similarity and low-similarity
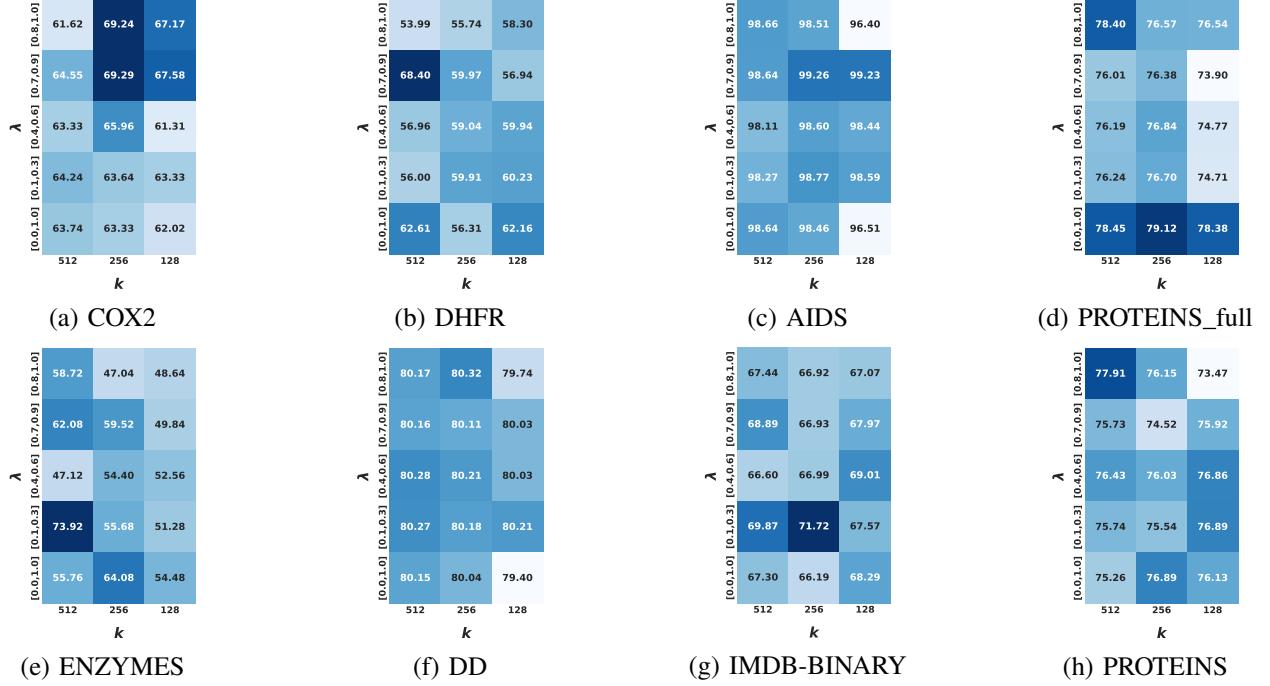
Fig. 5. Hyperparameter analysis of λ and k was conducted on eight datasets. Each heatmap displays the AUC score (%) achieved under different λ and k combinations, and darker colors indicate better performance.
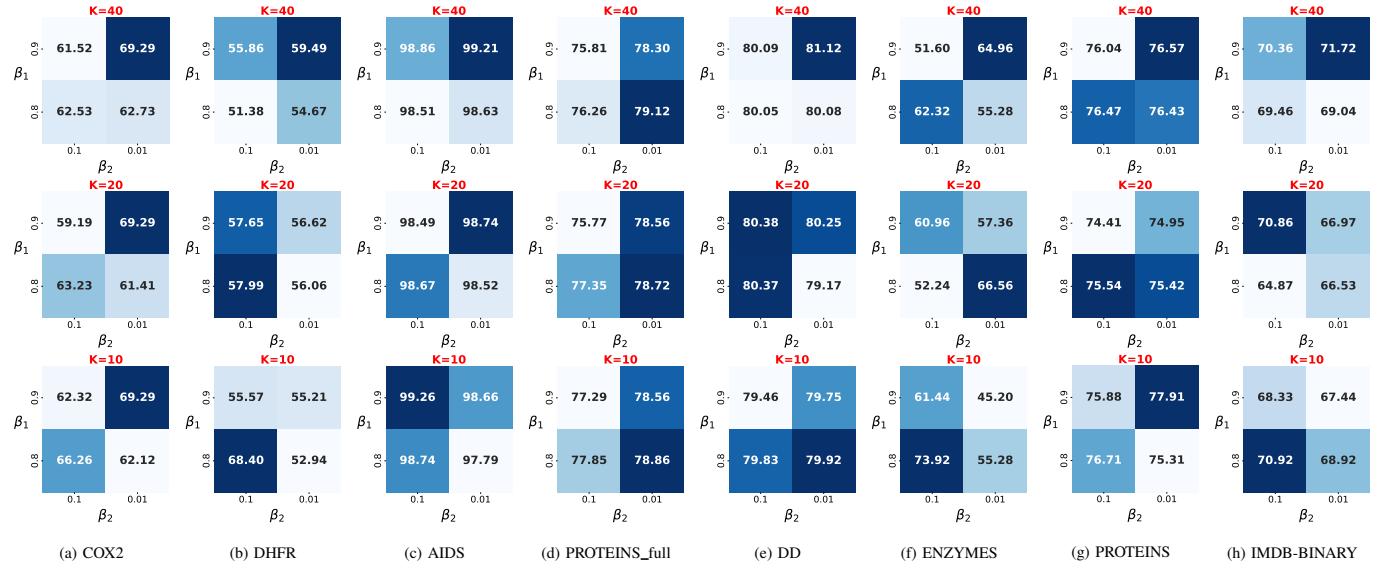

Fig. 6. Hyper-parameter analysis of K, β_1, and β_2 across eight datasets. Each heatmap shows AUC scores (%) under different combinations of β_1 and β_2 for three settings of $K \in \{10, 20, 40\}$. Higher AUC values indicate better anomaly detection performance.

regions. To investigate the impact of the choice of sampling regions and the number of samples K on model performance, we set K to 40, 20, and 10, respectively, and defined the high-similarity region by setting the quantile β_1 to 0.9 or 0.8, while defining the low-similarity region by setting the quantile β_2 to 0.1 or 0.01. On this basis, we conducted experiments with different parameter combinations under the scenario of noise intensity $\beta = 0.3$ to evaluate their impact on model performance.

As shown in Figure 6, across eight datasets the values of K, β_1 and β_2 exert a pronounced influence. Specifically, the effect of K differs between similarity regions. In high-

similarity regions (delimited by larger β_1), a larger K (e.g. 40) generally sustains higher performance, indicating that more samples help the model capture common patterns. Conversely, in low-similarity regions (delimited by smaller β_2), small K degrades performance, whereas larger K mitigates this drop and yields more stable results.

Datasets also vary in sensitivity to these changes. In EN-ZYMES, for instance, overly large K hurts performance, presumably because low-quality samples dilute the normal pattern. On DD and PROTEINS, a moderate K (e.g., 20) achieves the best balance: the model learns core normal features from high-similarity graphs while still acquiring suf-
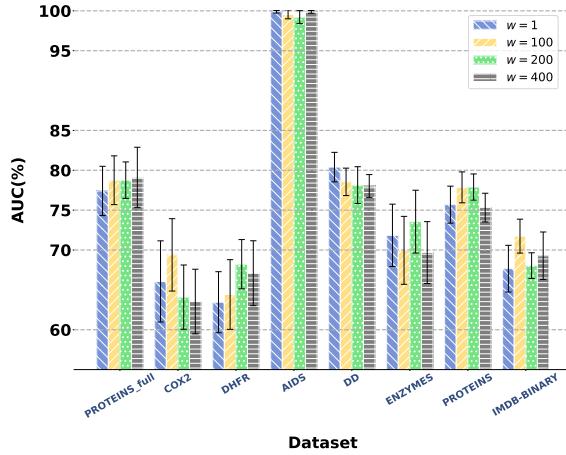
Fig. 7. Effect of hyper-parameter w on AUC performance across multiple datasets.

ficient anomalous information from low-similarity ones. AIDS remains stable in high-similarity regions even when K is reduced, whereas COX2 and DHFR exhibit large fluctuations in low-similarity regions where small K markedly degrades accuracy. These discrepancies are closely related to each dataset's distribution, noise level and graph complexity. PRO-TEINS_full shows similar trends, corroborating the generality of our observations.

In summary, judicious selection of K, β_1 and β_2 is crucial for strong performance in both similarity regions. Large K helps capture shared characteristics in high-similarity regions, while in low-similarity regions it can, to a limited extent, improve discrimination by supplying more anomalies; yet the gain is often unstable because of higher uncertainty and stronger noise. Therefore, parameters should be adapted dynamically to the characteristics of the concrete dataset in real applications.

c) The intensity weight w of adversarial training: In Equation 11, we optimize the model using adversarial alternating training, where the weight w is used to control the strength of the encoder's denoising. To investigate the impact of the adversarial training strength weight w on model performance, we set w to 1, 100, 200, and 400, respectively, and conducted extensive experiments under the condition of noise strength $\beta = 0.3$.

The experimental results in Figure 7 indicate that the hyperparameter w has a significant impact on model performance. The overall trend demonstrates that as w increases, the model's performance first improves and then declines on most datasets, suggesting the existence of an optimal w value. On the AIDS dataset, the model's performance is almost unaffected by w and remains at a high level. However, on the DHFR, PROTEINS_full, and PROTEINS datasets, the best performance is achieved when $w = 200$, indicating that moderate adversarial training strength can most effectively enhance the model's anomaly detection capability. An excessively high w (e.g., 400) leads to performance degradation on some datasets, possibly because the overly strong adversarial training intensity makes it difficult for the model to capture

the true data distribution in the first stage, thereby affecting its generalization ability. An excessively low w (e.g., 1) may prevent the model from fully learning the anomaly patterns, thus affecting detection performance. Therefore, selecting an appropriate w value is crucial for enhancing model performance and needs to be adjusted according to the characteristics of the dataset and the noise level to achieve the best denoising and learning effects.

E. Visualization (RQ4)

In Figure 8, we employed the t-SNE method to visualize the embedding changes of the DeNoise model during the denoising process. In the early stages of training (step 50), normal samples (represented by green circles in the training set and blue crosses in the test set) began to show a separation trend from abnormal samples (represented by black circles in the training set and red crosses in the test set) in the embedding space. However, due to the interference of noise, there were still some overlapping areas between the two, and some normal samples (blue crosses) had embeddings similar to abnormal samples, mixing into the cluster of abnormal samples. By step 100, the distribution of abnormal samples gradually moved away from the distribution area of normal samples and became more aggregated. This phenomenon indicates that the model began to more effectively distinguish between normal and abnormal samples, with the initial effectiveness of the denoising process becoming evident. Further, at step 150, the normal samples (blue crosses) that were previously mistakenly embedded into the abnormal sample cluster were correctly identified and repositioned back to the distribution area of normal samples. By the final stage of training (step 200), the distribution of normal and abnormal samples became extremely clear, with almost no overlap, and all abnormal samples in the test set (red crosses) were accurately separated. This demonstrates that the DeNoise model has reached a relatively ideal state in the denoising process, capable of efficiently and accurately distinguishing between normal and abnormal samples.

VI. CONCLUSION

In this paper, we delve into the truly unsupervised graph-level anomaly detection problem for the first time, breaking through the limitation of traditional methods that rely on the assumption that the training set contains only normal data. This assumption is hard to meet in the real world, and research on how to get rid of it is still relatively scarce. Through experimental verification, we find that existing methods based on the assumption that the training set contains only normal data generally suffer from unstable performance and decreased accuracy when facing noisy scenarios (i.e., the training set is contaminated with anomalous samples). In response to this situation, we propose a noise-resistant method named DeNoise. This method first trains a high-quality reconstruction model and uses it as a discriminator. With the embeddings generated by the encoder, DeNoise can separate the embeddings of normal samples and extract node embeddings with high information content from them. Subsequently, these node

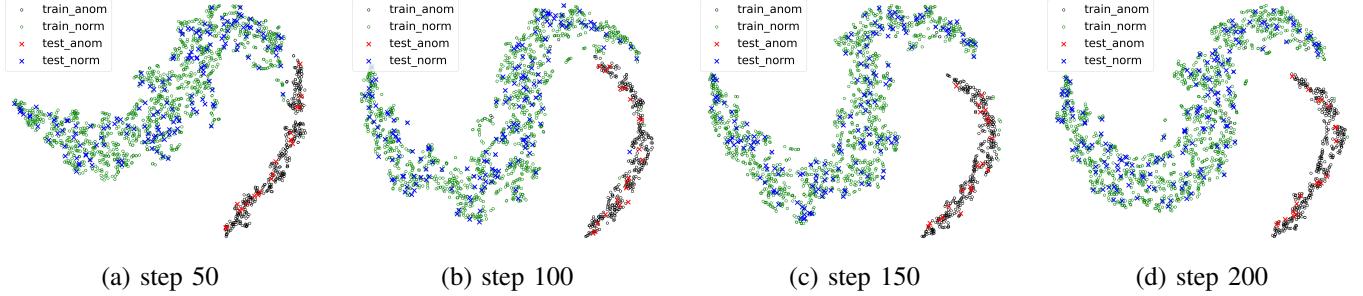(a) step 50   (b) step 100   (c) step 150   (d) step 200

Fig. 8. Progressive Denoising Visualization of Graph-Level Representations on the AIDS Dataset. We apply t-SNE to the graph embeddings obtained at training steps 50, 100, 150 and 200, projecting their evolution throughout the denoising process. Green circles: normal training graphs; black circles: anomalous training graphs; blue crosses: normal test graphs; red crosses: anomalous test graphs.

embeddings are infused into each graph embedding, and the model is optimized with the sampled positive and negative pairs to achieve a denoising effect. Extensive experimental results show that DeNoise can effectively learn high-quality embedding representations under different noise scenarios (with different proportions of anomalous samples mixed in the training set) and maintain stable anomaly detection performance.

REFERENCES

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

[2] K. Xu, W. Hu, J. Leskovec, and S. Jegelka, "How powerful are graph neural networks?" *arXiv preprint arXiv:1810.00826*, 2018.

[3] S. Zhang, P. Xi, M. Jiang, G. Zhang, and D. Cheng, "Latent representation learning for attributed graph anomaly detection," *ACM Transactions on Knowledge Discovery from Data*, vol. 19, no. 7, pp. 1–22, 2025.

[4] V. Petar, C. Guillem, C. Arantxa, R. Adriana, L. Pietro, and B. Yoshua, "Graph attention networks," in *International conference on learning representations*, vol. 8, 2018.

[5] B. Jiang, Z. Zhang, D. Lin, J. Tang, and B. Luo, "Semi-supervised learning with graph learning-convolutional networks," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2019, pp. 11 313–11 320.

[6] X. Ma, J. Wu, J. Yang, and Q. 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*, 2023, pp. 1631–1642.

[7] G. Zhang, Z. Yang, J. Wu, J. Yang, S. Xue, H. Peng, J. Su, C. Zhou, Q. Z. Sheng, L. Akoglu *et al.*, "Dual-discriminative graph neural network for imbalanced graph-level anomaly detection," *Advances in Neural Information Processing Systems*, vol. 35, pp. 24 144–24 157, 2022.

[8] Q. Chen, J. Deng, D. Cheng, J. Li, and L. Liu, "Multi-view debiasing representation learning for recommender systems," *Information Processing & Management*, vol. 63, no. 2, p. 104429, 2026.

[9] L. He, D. Cheng, G. Zhang, and S. Zhang, "Leveraging long-range nodes in multi-view graph contrastive learning," *Information Fusion*, vol. 122, no. C, 2025.

[10] P. Xi, D. Cheng, G. Lu, Z. Deng, G. Zhang, and S. Zhang, "Identifying local useful information for attribute graph anomaly detection," *Neurocomputing*, vol. 617, p. 128900, 2025.

[11] P. Xi, D. Cheng, Z. Deng, G. Zhang, and S. Zhang, "Lragad: Local information recognition for attribute graph anomaly detection," in *2023 IEEE 35th International Conference on Tools with Artificial Intelligence (ICTAI)*. IEEE, 2023, pp. 997–1001.

[12] C. Qiu, M. Kloft, S. Mandt, and M. Rudolph, "Raising the bar in graph-level anomaly detection," *arXiv preprint arXiv:2205.13845*, 2022.

[13] L. Zhao and L. Akoglu, "On using classification datasets to evaluate graph outlier detection: Peculiar observations and new insights," *Big Data*, vol. 11, no. 3, pp. 151–180, 2023.

[14] Z. Yang, G. Zhang, J. Wu, J. Yang, S. Xue, A. Beheshti, H. Peng, and Q. Z. Sheng, "Global interpretable graph-level anomaly detection via prototype," in *Proceedings of the 31st ACM SIGKDD Conference on Knowledge Discovery and Data Mining V. 2*, 2025, pp. 3586–3597.

[15] S. Kim, S. Y. Lee, F. Bu, S. Kang, K. Kim, J. Yoo, and K. Shin, "Rethinking reconstruction-based graph-level anomaly detection: limitations and a simple remedy," *Advances in Neural Information Processing Systems*, vol. 37, pp. 95 931–95 962, 2024.

[16] R. Ma, G. Pang, L. Chen, and A. 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*, 2022, pp. 704–714.

[17] J. Li, Q. Xing, Q. Wang, and Y. Chang, "Cvtgad: Simplified transformer with cross-view attention for unsupervised graph-level anomaly detection," in *Joint European conference on machine learning and knowledge discovery in databases*. Springer, 2023, pp. 185–200.

[18] Y. Liu, K. Ding, Q. Lu, F. Li, L. Y. Zhang, and S. Pan, "Towards self-interpretable graph-level anomaly detection," *Advances in Neural Information Processing Systems*, vol. 36, pp. 8975–8987, 2023.

[19] Z. Yang, G. Zhang, J. Wu, J. Yang, H. Peng, and P. Liò, "Learning from graph-graph relationship: a new perspective on graph-level anomaly detection," *IEEE Transactions on Knowledge and Data Engineering*, 2025.

[20] Z. Li, S. Liang, J. Shi, and M. van Leeuwen, "Cross-domain graph level anomaly detection," *IEEE Transactions on Knowledge and Data Engineering*, 2024.

[21] F. Feizi, H. Rahmani, A. Hosseinnia, and A. Bagheri, "Graph-guard: A framework for heterogeneous graph anomaly detection using supervised and unsupervised techniques," in *2024 10th International Conference on Web Research (ICWR)*. IEEE, 2024, pp. 125–129.

[22] Y. Liu, K. Ding, H. Liu, and S. Pan, "Good-d: On unsupervised graph out-of-distribution detection," in *Proceedings of the sixteenth ACM international conference on web search and data mining*, 2023, pp. 339–347.

[23] C. Niu, G. Pang, and L. Chen, "Graph-level anomaly detection via hierarchical memory networks," in *Joint European conference on machine learning and knowledge discovery in databases*. Springer, 2023, pp. 201–218.

[24] Y. Rong, W. Huang, T. Xu, and J. Huang, "Dropedge: Towards deep graph convolutional networks on node classification," *arXiv preprint arXiv:1907.10903*, 2019.

[25] K. Kong, G. Li, M. Ding, Z. Wu, C. Zhu, B. Ghanem, G. Taylor, and T. Goldstein, "Robust optimization as data augmentation for large-scale graphs," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2022, pp. 60–69.

[26] Y. Wang, W. Wang, Y. Liang, Y. Cai, and B. Hooi, "Mixup for node and graph classification," in *Proceedings of the Web Conference 2021*, 2021, pp. 3663–3674.

[27] Y. Liu, L. Huang, B. Cao, X. Li, F. Giunchiglia, X. Feng, and R. Guan, "A simple but effective approach for unsupervised few-shot graph classification," in *Proceedings of the ACM Web Conference 2024*, 2024, pp. 4249–4259.

[28] Z. Hou, X. Liu, Y. Cen, Y. Dong, H. Yang, C. Wang, and J. Tang, "Graphmae: Self-supervised masked graph autoencoders," in *Proceedings of the 28th ACM SIGKDD conference on knowledge discovery and data mining*, 2022, pp. 594–604.

[29] T. N. Kipf and M. Welling, "Variational graph auto-encoders," *arXiv preprint arXiv:1611.07308*, 2016.

[30] C. Morris, N. M. Kriege, F. Bause, K. Kersting, P. Mutzel, and M. Neumann, "Tudataset: A collection of benchmark datasets for learning with graphs," *arXiv preprint arXiv:2007.08663*, 2020.