KEGOD: Kernel-enhanced Latent Substructure Learning for Graph Out-Of-Distribution Detection

Yifan Wang
University of International Business and Economics
Beijing, China
yifanwang@uibe.edu.cn

Haodong Zhang
Northeastern University
Shenyang, China
2110496@stu.neu.edu.cn

Zhiping Xiao
University of Washington
Seattle, WA, USA
patxiao@uw.edu

Yusheng Zhao*
Peking University
Beijing, China
yusheng.zhao@stu.pku.edu.cn

Siyu Yi
Sichuan University
Chengdu, China
siyuyi@scu.edu.cn

Nan Yin
Hong Kong University of Science and Technology
Hong Kong, China
yinnan8911@gmail.com

Xinwang Liu
National University of Defense Technology
Changsha, China
xinwangliu@nudt.edu.cn

Ming Zhang*
Peking University
Beijing, China
mzhang_cs@pku.edu.cn

Wei Ju†
Sichuan University
Chengdu, China
juwei@scu.edu.cn

Abstract

Out-of-Distribution (OOD) detection, which seeks to identify samples deviating from the In-Distribution (ID) training distribution at test time, is crucial for building robust machine learning systems. While extensive efforts have been made for Euclidean data, OOD detection on graph-structured data remains relatively underexplored. On the one hand, the specific properties of a graph may be attributed to its substructures. On the other hand, acquiring labeled data for graph learning is typically time-consuming and labor-intensive. Toward this end, in this paper, we propose a novel kernel-enhanced graph substructure learning framework termed KEGOD for unsupervised graph OOD detection. Specifically, we introduce a learnable graph generator to construct the augmented graph view that preserves distinguishable structure information. Then, for both the input graph and augmented view, a graph neural network (GNN) branch and a graph kernel (GK) branch are incorporated to explore graph latent patterns. By performing multi-branch concordance learning on the extracted graph patterns, our KEGOD captures complementary ID structural semantics in both implicit and explicit manners, enabling reliable detection of OOD graphs through semantic inconsistency. Finally, we build a self-adaptive training mechanism to automatically control diverse sensitivities of the graph patterns. Experimental results on several public graph datasets reveal the superiority of our KEGOD. Our code is available at https://github.com/jamesyifan/KEGOD.

CCS Concepts

• **Mathematics of computing** → **Graph algorithms**; • **Computing methodologies** → **Neural networks**; **Kernel methods**; **Anomaly detection**.

Keywords

Out-Of-Distribution Detection, Graph Neural Networks, Graph Kernels, Concordance Learning

ACM Reference Format:
Yifan Wang, Haodong Zhang, Zhiping Xiao, Yusheng Zhao, Siyu Yi, Nan Yin, Xinwang Liu, Ming Zhang, and Wei Ju. 2026. KEGOD: Kernel-enhanced Latent Substructure Learning for Graph Out-Of-Distribution Detection. In *Proceedings of the ACM Web Conference 2026 (WWW '26), April 13–17, 2026, Dubai, United Arab Emirates.* ACM, New York, NY, USA, 11 pages. https://doi.org/10.1145/3774904.3792465

1 Introduction

Graphs serve as an essential tool for representing structured and relational data, and are ubiquitous in numerous application domains [44, 45], e.g., biological networks [2], molecule graphs [9] and recommender system [46], etc. Graph neural networks (GNNs) [17, 41, 47], which extract relevant features from graphs by exploiting the powerful representational capabilities of deep learning, are frequently applied in open-world scenarios. However, these GNN algorithms often struggle to handle Out-Of-Distribution (OOD) graph inputs originating from a distribution not encountered during the training phase. This underscores the critical necessity for a reliable and effective graph learning system that can reliably classify In-Distribution (ID) graph data while proficiently detecting unknown Out-Of-Distribution (OOD) inputs during inference.

*National Key Laboratory for Multimedia Information Processing, School of Computer Science, Beijing Key Laboratory of Software and Hardware Cooperative Artificial Intelligence Systems, Peking University.
†Corresponding author.

WWW '26, Dubai, United Arab Emirates

ACM ISBN 979-8-4007-2307-0/2026/04
https://doi.org/10.1145/3774904.3792465

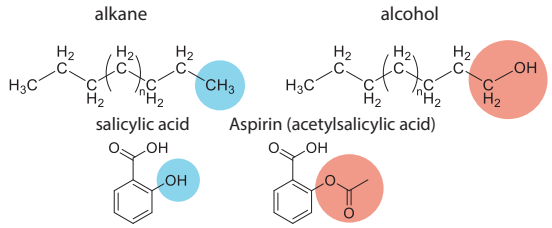


Figure 1: The difference between alcohol and alkane relies on the hydroxyl group. And modifying a substructure of the salicylic acid can turn it into acetylsalicylic acid (Aspirin).

As a fundamental research problem in machine learning, OOD detection has become an important research hot spot when it comes to real-world applications, such as medical diagnosis [4] and autonomous driving [6]. However, most works focus on natural language processing [53] domains or computer vision [20, 32], with substantially less exploration in the field of graph-structured data. A pertinent research area is graph anomaly detection, which typically focuses on identifying malicious graph samples within safety-critical systems [25, 52]. In contrast, OOD samples originate from a distribution that the model has not encountered during training, and graph anomaly detection can be seen as a specialized task of graph OOD detection. Recently, several pioneering studies have begun to study graph OOD detection. GOOD-D [23], AAGOD [8] and GOODAT [43] develop graph OOD detection models by training from scratch, post-hoc and test-time, respectively.

Despite the versatility of these graph OOD detection methods, the problem remains challenging, primarily due to the following two reasons: **(i) Huge space of graph OOD samples**. The latent patterns of ID graphs may be violated in various forms, such as feature-level variation (e.g., descriptive features of nodes) and structure-deviated samples (e.g., graph structure), which makes the space of OOD samples typically vast. Among them, the substructure of a graph typically determines the properties of the graph. For example, molecular toxicity can be induced by pharmacophores consisting of specific substructures [39]. Moreover, as shown in Figure 1, the differentiation between alcohol and alkane often relies on the presence of hydrogen and oxygen atoms at the end of molecules. Also, by changing only a substructure of the salicylic acid molecule, we can turn it into acetylsalicylic acid, which is a very effective type of medicine, well-known as Aspirin, further explaining the importance of the substructure in the graph. **(ii) Scarcity of class labels and OOD samples**. Most existing works assume the availability of both ID and OOD sample labels. This assumption is strong and can be impractical in real-world scenarios, particularly fine-grained annotations, as they are costly and labor-intensive to obtain. Consequently, it naturally raises a meaningful question: *Can we effectively detect OOD graphs by explicitly exploring latent graph patterns within unlabeled ID data?*

Recent advancements in self-supervised learning have demonstrated the effectiveness of concordance learning, especially contrastive learning, on unlabeled data [5, 37, 48, 54]. The method derives a robust inductive bias by encouraging concordance among multiple views of the same graph, thereby capturing intrinsic structural and semantic patterns, while enforcing distinction from other graph instances to enhance discriminative representation learning. However, such representations may not be optimal for OOD detection, which necessitates distinguishing between ID and OOD graph inputs, rather than solely capturing distribution variance within ID samples [38]. Specifically, for graph-structured input, the prevailing graph contrastive learning often employs stochastic augmentation techniques (e.g., node/edge dropping, feature perturbation and graph diffusion) to generate diverse views, which may inadvertently disrupt the graph's intrinsic structural and semantic integrity [34], potentially resulting in the synthesis of views that deviate from the ID manifold and resemble OOD graph samples.

In this paper, we propose a joint **K**ernel-**e**nhanced **G**raph-**O**ut-of-**D**istribution detection method (**KEGOD** for abbreviation). Our main concept is to capture the latent patterns shared among training ID graphs through both implicit and explicit manners to preserve the most representative structures for effective OOD graph detection. Specifically, we develop a learnable graph view generator that captures the essential information required for effective graph self-supervised learning, avoiding the introduction of detrimental perturbations. Then, for both the original input graph and the generated view, we incorporate coupled branches to holistically extract topological information. On the one hand, a GNN branch leverages the message-passing paradigm to implicitly extract latent patterns of the graph. On the other hand, a GK branch explores common random walk sequences to compare graph samples with learnable filters, which explicitly incorporate latent patterns of the graph into representations. Next, we conduct multi-branch graph concordance learning to promote the agreement of branches for two different views, therefore learning high-quality latent patterns of ID graph samples. Moreover, we equip each branch of concordance learning with an adaptive learning loss to automatically control its contribution. Finally, for each test graph sample, the branch disagreement between two different views can be exploited as an indicative OOD scoring function for the task. In a nutshell, the main contributions of this paper are as follows:

- *Conceptual:* We propose KEGOD, a self-supervised framework for effective graph-level OOD detection via the exploration of explainable latent graph patterns.
- *Methodological:* We introduce a graph view generator to preserve representative structural information of ID graphs, and incorporate both a GNN and a GK branch to implicitly and explicitly mine latent patterns from complementary perspectives.
- *Experimental:* Extensive experiments on a range of public datasets are conducted to evaluate our KEGOD. The results show the efficiency and outstanding explainability of our framework.

2 Problem Definition & Preliminaries

Definition 1 (Graph). *We define a graph as $G = (\mathcal{V}, \mathcal{E}, \boldsymbol{X})$, where $\mathcal{V}$ and $\mathcal{E} \in \mathcal{V} \times \mathcal{V}$ here denote the node and edge set, respectively. The feature matrix of the node set is captured by $\boldsymbol{X} \in \mathbb{R}^{|\mathcal{V}| \times d'}$, where each row, i.e., $\boldsymbol{x}_u \in \mathbb{R}^{d'}$, corresponds to the feature vector of node u, and d' is the dimension of the node feature. The structure information of the graph $\mathcal{G}$ can be characterized by an adjacency matrix $\boldsymbol{A} \in \mathbb{R}^{|\mathcal{V}| \times |\mathcal{V}|}$, where each entry of the matrix, i.e., $a_{uv} = 1$ if an edge $(u, v) \in \mathcal{E}$ exists and $a_{uv} = 0$ otherwise.*

Definition 2 (Graph-level OOD Detection). *Given an ID graph dataset $\mathcal{D}^{in} = \{G_1^{in}, \cdots G_{N_1}^{in}\}$, we assume there is an OOD*

graph dataset $\mathcal{D}^{out} = \{G_1^{out}, \cdots G_{N_2}^{out}\}$. Given a graph G drawn either from a certain in-distribution $\mathbb{P}^{in}$ or an OOD distribution $\mathbb{P}^{out}$, the fundamental goal of graph-level OOD detection is to correctly distinguish its source distribution (i.e., $\mathbb{P}^{in}$ or $\mathbb{P}^{out}$). In this paper, we aim to learn an explainable model $f(\cdot) : G \to (s, G')$. The model not only generates an OOD detection score $s \in \mathbb{R}$ for the graph G (with a larger s indicating a higher probability that G is from $\mathbb{P}^{out}$), but also provides corresponding explanations $G' \in \mathbb{G}$ that is able to clarify why G is identified as an ID/OOD sample. In practice, due to the limited availability of OOD samples, $f(\cdot)$ is only trained on the ID dataset $\mathcal{D}^{in}_{train} \subset \mathcal{D}^{in}$ while evaluated on the test set comprising both $\mathcal{D}^{in}_{test} \subset \mathcal{D}^{in}$ and $\mathcal{D}^{out}_{test} \subset \mathcal{D}^{out}$ ($\mathcal{D}^{in}_{test} \cap \mathcal{D}^{in}_{train} = \emptyset$).

Preliminary 1 (Graph Kernels). *Graph kernels (GKs) serve as positive semidefinite kernel functions, extending the applicability of the entire family of kernel-based learning methods on graph-structured data. The basic concept behind the majority of graph kernels is to decompose graphs into high-order substructures and quantify graph similarities through kernel functions. Formally, given two graphs $G = (\mathcal{V}, \mathcal{E}, \boldsymbol{X})$ and $G' = (\mathcal{V}', \mathcal{E}', \boldsymbol{X}')$, the similarity between them can be measured by the graph kernel $k(G, G')$, defined as:*

$$k(G, G') = \sum_{u \in \mathcal{V}} \sum_{u' \in \mathcal{V}'} k_{base}\left(f_G(u), f_{G'}(u')\right), \tag{1}$$

where $k_{base}(\cdot)$ is a base kernel (i.e., an inner product in a Hilbert space) comparing the substructures centered at nodes u and u'. The mapping function $f_G(\cdot)$ encodes the counts of substructures present in G, such as shortest paths [1], subtrees [35], graphlets [36].

3 The Proposed Model

The basic idea of our KEGOD is to explore the latent patterns within the ID graphs both implicitly and explicitly in the training phase to identify the OOD graph. The objective is to maximize the agreement between graphs and their generated representative graph views in a data-driven manner to identify deviations from learned ID patterns. The overall framework can be shown in Figure 2.

3.1 Learnable Graph View Generator

To preserve important structural and semantic information shared by ID graph data, we adopt a GNN-based generator to produce augmented graph views while avoiding perturbations. Simultaneously, we identify relevant node features using a feature masking transformation. Overall, we formulate a graph view generator $g(\cdot) = \{g_\phi(\cdot), g_m(\cdot)\}$ to generate an augmented view $\widetilde{G}$, where ϕ is the learnable parameter of the structure generator $g_\phi(\cdot)$, and $g_m(\cdot)$ corresponds to the feature masking transformation.

Given the assumption that both node feature matrix $\boldsymbol{X}$ and original graph structure information $\boldsymbol{A}$ contribute to the existence of edges, the generator integrates both $\boldsymbol{X}$ and $\boldsymbol{A}$ into the embedding network and employs GNN layers following a message-passing scheme. We can get the refined edge weight of $(u, v) \in \mathcal{E}$ with updated node embedding $\boldsymbol{h}_u^{(L)}$ at (L)-th layer:

$$\begin{aligned} \omega_{uv} &= \text{MLP}([\boldsymbol{h}_u^{(L)}; \boldsymbol{h}_v^{(L)}]), \\ \boldsymbol{h}_u^{(l)} &= g_\phi^{(l)}\left(\boldsymbol{h}_u^{(l-1)}, \{\boldsymbol{h}_v^{(l-1)}\}_{v \in \mathcal{N}(u)}, \boldsymbol{A}\right) \\ &= \text{GNN}_\phi^{(l)}\left(\boldsymbol{h}_u^{(l-1)}, \{\boldsymbol{h}_v^{(l-1)}\}_{v \in \mathcal{N}(u)}, \boldsymbol{A}\right), \end{aligned} \tag{2}$$

where $\text{GNN}_\phi^{(l)}(\cdot)$ denotes the (l)-th GNN layer. For each generator, we initialize the feature of (0)-th layer as matrix $\boldsymbol{X}$. We relate the edge weight ω_{uv} with a random variable $p_{uv} \sim \text{Bernoulli}(w_{uv})$, where the edge (u, v) is kept if $p_{uv} = 1$ and dropped otherwise, resulting in an augmented structure $\widetilde{\boldsymbol{A}}$. For differentiable optimization of the view generator $g_\phi^{(l)}(\cdot)$, we approximate the discrete sampling process by relaxing p_{uv} to a continuous variable in $[0, 1]$ with the Gumbel-Max reparametrization trick [13], formulated as:

$$p_{uv} = \text{Sigmoid}\left(\frac{\log \sigma - \log(1 - \sigma) + \omega_{uv}}{\tau}\right), \tag{3}$$

where τ is the temperature hyper-parameter and $\sigma \sim \text{Uniform}(0, 1)$. Note that as $\tau \to 0$, p_{uv} approaches binary.

Apart from edge-dropping for graph structure learning, we introduce feature masking to inject controlled noise into the node attributes. Specifically, given a feature matrix $\boldsymbol{X}$, we can get the mask $\boldsymbol{m} \in \{0, 1\}^{d'}$, where each entry is independently sampled from a Bernoulli distribution with masking probability p_x. Formally, the masked feature matrix of node set can be defined as:

$$\widetilde{\boldsymbol{X}} = g_m(\boldsymbol{X}) = [\boldsymbol{x}_1 \odot \boldsymbol{m}, \ldots, \boldsymbol{x}_{|\mathcal{V}|} \odot \boldsymbol{m}], \tag{4}$$

where $\boldsymbol{x}_u$ refer to the feature of node u and $\widetilde{\boldsymbol{X}}$ can be employed as the augmented feature matrix, $g_m(\cdot)$ is the masking function.

Moreover, to avoid a trivial solution where the generator learns to produce an augmented view that is almost identical to the input graph, we apply the following restriction to regularize the ratio of edges being dropped in each graph:

$$\mathcal{L}_{reg} = \sum_{(u,v) \in \mathcal{E}} \omega_{uv} / |\mathcal{E}|. \tag{5}$$

3.2 Dual Complementary Branch for Graph Exploration

To effectively capture latent patterns within graphs from complementary views, we integrate the two branches, i.e., the implicit GNN branch and explicit GK branch, for graph-level OOD detection.

Implicit Graph Neural Network Branch. The message-passing mechanism lies at the core of GNNs, enabling each node to refine its representation by aggregating features from its local neighborhood. Through iterative propagation across layers, GNNs effectively encode both structural dependencies and node attributes, facilitating the modeling of intricate topological patterns. We utilize a GNN to implicitly extract graph topical information. Take the original graph input as an example, the node representation of each node $u \in \mathcal{V}$ at the layer l is updated as:

$$\boldsymbol{h}_u^{(l)} = \text{GNN}_\theta^{(l)}\left(\boldsymbol{h}_u^{(l-1)}, \{\boldsymbol{h}_v^{(l-1)}\}_{v \in \mathcal{N}(u)}, \boldsymbol{A}\right), \tag{6}$$

where $\mathcal{N}(u)$ represents the neighborhood of u. Note that the feature of (0)-th layer can be initialized as $\boldsymbol{X}$ or $\widetilde{\boldsymbol{X}}$ for the original graph input and generated graph view, respectively. Eventually, by applying a readout function over the node embeddings from the final L-th layer. Formally, it can be:

$$\boldsymbol{h}_G^{gnn} = \text{READOUT}\left(\{\boldsymbol{h}_u^L\}_{u \in \mathcal{V}}\}\right), \tag{7}$$

where $\boldsymbol{h}_G^{gnn}$ denotes the implicit representation of the graph G.

Explicit Graph Kernel Branch. However, it remains challenging to explore rich high-order substructures (i.e., paths, rings) for

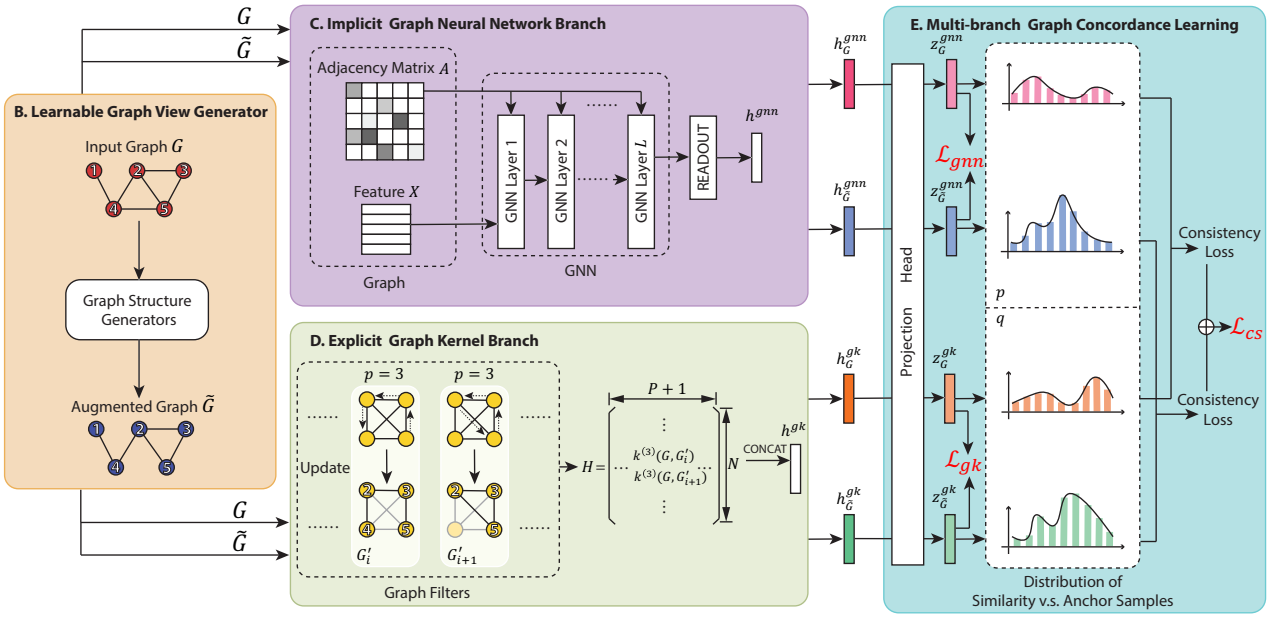


Figure 2: The overall framework of the proposed KEGOD, which feeds the original graph and its generated view into two branches (i.e., implicit GNN branch and explicit graph kernel branch). We leverage a self-supervised graph concordance learning to adaptively identify OOD graphs in a data-driven manner.

GNNs since the methods simply capture structural information via message-passing paradigm. Consequently, there is an anticipation to explicitly explore graph topology as a complement. Toward this end, we introduce a random walk GK to explicitly explore graph topology [29, 42, 51]. Specifically, given the graph $G = (\mathcal{V}, \mathcal{E}, \boldsymbol{X})$ and a graph filter $G' = (\mathcal{V}', \mathcal{E}', \boldsymbol{X}')$, we construct their direct product graph $G_\times = \{\mathcal{V}_\times, \mathcal{E}_\times\}$, where $\mathcal{V}_\times = \{(u, u') : u \in \mathcal{V} \wedge u' \in \mathcal{V}'\}$ and $\mathcal{E}_\times = \{\{(u, u'), (v, v')\} : (u, v) \in \mathcal{E} \wedge (u', v') \in \mathcal{E}'\}$. Thus, a simultaneous walk on both graph G and G' can be interpreted as a random walk on $G_\times$. The adjacency matrix of $G_\times$, i.e., $\boldsymbol{A}_\times$, can be used to determine the number of matching walks under the assumption of uniform starting and stopping probability distributions over the nodes of G and G'. In this manner, the p-step ($p \in \mathbb{N}$) random walk kernel between G and G' can be defined as:

$$k^{(p)}(G, G') = \sum_{i=1}^{|\mathcal{V}_\times|} \sum_{j=1}^{|\mathcal{V}_\times|} [\boldsymbol{A}_\times^p]_{ij}. \tag{8}$$

Furthermore, to incorporate node attribute information into the kernel computation, we associate each graph filter with a trainable node feature matrix $\boldsymbol{X}' \in \mathbb{R}^{|\mathcal{V}'| \times d'}$. In this way, we can define $\boldsymbol{S} = \boldsymbol{X}'\boldsymbol{X}'^{\mathrm{T}} \in \mathbb{R}^{|\mathcal{V}'| \times |\mathcal{V}|}$, where each entry quantifies the alignment between node features of two graphs. Then the matrix $\boldsymbol{S}$ is subsequently flattened into a vector $\boldsymbol{s}$ (i.e., $\boldsymbol{s} = flat(\boldsymbol{S}), \boldsymbol{s} \in \mathbb{R}^{|\mathcal{V}'| \cdot |\mathcal{V}|}$). We compute the kernel that considers both the number of common walks and node features between the two graphs as:

$$k^{(p)}(G, G') = \sum_{i=1}^{|\mathcal{V}_\times|} \sum_{j=1}^{|\mathcal{V}_\times|} \left[(\boldsymbol{s}\boldsymbol{s}^{\mathrm{T}}) \odot \boldsymbol{A}_\times^p\right]_{ij} = \boldsymbol{s}^{\mathrm{T}} \boldsymbol{A}_\times^p \boldsymbol{s}. \tag{9}$$

where $\odot$ denotes the Hadamard product between two matrices. For each walk length $p \in \mathcal{P} = \{0, \ldots, P\}$ and each graph filter $G'_i \in \{G'_1, \ldots, G'_N\}$, we compute the corresponding kernel value to encode the similarity and construct a kernel response matrix $\boldsymbol{H} \in \mathbb{R}^{N \times (P+1)}$, where each entry $\boldsymbol{H}_{ij} = k^{(j-1)}(G, G'_i)$. We then flatten matrix $\boldsymbol{H}$ into a vector and pass it through a fully connected layer to obtain the graph-level representation $\boldsymbol{h}_G^{\mathrm{gk}}$ as:

$$\boldsymbol{h}_G^{gk} = flat(\boldsymbol{H}) = \mathrm{CONCAT}\big(\boldsymbol{H}_{:j} | j \in \mathcal{P}\big). \tag{10}$$

3.3 Multi-branch Graph Concordance Learning

Given the graph G_i and its augmented view $\widetilde{G}_i$, we extract their representations from coupled branches, i.e., $\boldsymbol{h}_{G_i}^{gnn}, \boldsymbol{h}_{\widetilde{G}_i}^{gnn}$ for GNN and $\boldsymbol{h}_{G_i}^{kb}, \boldsymbol{h}_{\widetilde{G}_i}^{kb}$ for GK branch. Then, we formalize a multi-branch graph concordance learning framework, which performs intra-branch concordance learning within both the GNN and GK branches, and inter-branch alignment across them.

Implicit Branch Concordance. Since GNNs encode graph latent patterns into learned representations via the implicit message-passing framework, the GNN branch contrast is employed to find the implicit patterns shared by the ID graph samples. To this end, we maximize the agreement between the learned graph representations of the two views. Specifically, we first map $\boldsymbol{h}_{G_i}^{gnn}$ and $\boldsymbol{h}_{\widetilde{G}_i}^{gnn}$ into GNN-space embeddings $\boldsymbol{z}_{G_i}^{gnn}$ and $\boldsymbol{z}_{\widetilde{G}_i}^{gnn}$ with MLP-based projection head. The InfoNCE loss is adopted to maximize the latent pattern agreement under the implicit GNN branch, defined as:

$$\begin{aligned} \mathcal{L}_{gnn} &= \frac{1}{2|\mathcal{B}|} \sum_{G_i \in \mathcal{B}} \left[\ell(\boldsymbol{z}_{G_i}^{gnn}, \boldsymbol{z}_{\widetilde{G}_i}^{gnn}) + \ell(\boldsymbol{z}_{\widetilde{G}_i}^{gnn}, \boldsymbol{z}_{G_i}^{gnn})\right], \\ \ell(\boldsymbol{z}_{G_i}^{gnn}, \boldsymbol{z}_{\widetilde{G}_i}^{gnn}) &= -\log \frac{\psi(\boldsymbol{z}_{G_i}^{gnn}, \boldsymbol{z}_{\widetilde{G}_i}^{gnn})}{\sum_{G_j \in \mathcal{B} \setminus G_i} \psi(\boldsymbol{z}_{G_i}^{gnn}, \boldsymbol{z}_{G_j}^{gnn})}, \end{aligned} \tag{11}$$

where $\psi(\cdot,\cdot)$ is implemented as the exponential temperature-scaled cosine similarity, namely, $\psi(\cdot,\cdot) = \exp(\cos(\cdot,\cdot)/\tau)$ with temperature parameter τ. And $\mathcal{B}$ is the training batch containing multiple ID graph samples.

Explicit Branch Concordance. As a representative type of GKs, the random walk kernel adopted in our framework quantifies the similarity between graphs by enumerating shared random walk sequences. Thus, the explicit branch concordance focuses on exploring random walk kernel-based patterns shared among ID graph samples to inherently capture high-order substructural patterns. Similar to the implicit branch concordance, the learned graph representations $\boldsymbol{h}^{gk}_{G_i}$ and $\boldsymbol{h}^{gk}_{\widetilde{G}_i}$ are projected into GK-space embeddings $z^{gk}_{G_i}$ and $z^{gk}_{\widetilde{G}_i}$ with MLP-based projection head. The explicit branch concordance learning loss can be:

$$\mathcal{L}_{gk} = \frac{1}{2|\mathcal{B}|}\sum_{G_i\in\mathcal{B}}\Big[\ell(z^{gk}_{G_i}, z^{gk}_{\widetilde{G}_i}) + \ell(z^{gk}_{\widetilde{G}_i}, z^{gk}_{G_i})\Big],$$
$$\ell(z^{gk}_{G_i}, z^{gk}_{\widetilde{G}_i}) = -\log\frac{\psi(z^{gk}_{G_i}, z^{gk}_{\widetilde{G}_i})}{\sum_{G_j\in\mathcal{B}\setminus G_i}\psi(z^{gk}_{G_i}, z^{gk}_{G_j})}. \tag{12}$$

Cross-branch Concordance. As the two branches learn graph latent patterns from complementary views, we achieve concordance between their representations for collaborative integration. However, discrepancies may arise in the derived representation spaces, and simple alignment may be sub-optimal. To address this, we enhance each graph sample by comparing its similarities to other samples within the respective embedding spaces of both branches.

Specifically, we first construct a memory bank by randomly sampling a set of anchor graphs $\{G_{a1},\ldots,G_{aT}\}$, and storing them for reference. Then, we get the projected embedding of sampled graphs under two branches, namely, $\{z^{gnn}_{G_{at}}\}^T_{t=1}$ and $\{z^{gk}_{G_{at}}\}^T_{t=1}$. Note that maintaining a sufficiently large and diverse set of anchor graph samples is crucial for adequately capturing the intrinsic variability of the ID graphs. However, processing such large samples at once is computationally costly. To circumvent this, we implement a dynamically updated memory bank, a queue storing anchor embeddings from recent mini-batches. Formally, given implicit branch projection $z^{gnn}_{G_i}$ (and explicit GK branch projection $z^{gk}_{G_i}$), the pairwise similarity between the embedding and all anchor embeddings $\{z^{gnn}_{G_{at}}\}^T_{t=1}$ (and $\{z^{gk}_{G_{at}}\}^T_{t=1}$) can be formulated as:

$$p^i_t = \frac{\psi(z^{gnn}_{G_i}, z^{gnn}_{G_{at}})}{\sum^T_{t'=1}\psi(z^{gnn}_{G_i}, z^{gnn}_{G_{at'}})}, q^i_t = \frac{\psi(z^{gk}_{G_i}, z^{gk}_{G_{at}})}{\sum^T_{t'=1}\psi(z^{gk}_{G_i}, z^{gk}_{G_{at'}})}. \tag{13}$$

For each G_i, we encourage the consistency between the two derived probability distributions, i.e., $\boldsymbol{p}^i = [p^i_1,\ldots,p^i_T]$ and $\boldsymbol{q}^i = [q^i_1,\ldots,q^i_T]$. The concordance loss can be defined as:

$$\ell(z^{gnn}_{G_i}, z^{gk}_{G_i}) = \frac{1}{2|\mathcal{B}|}\sum_{G_i\in\mathcal{B}}\big[D_{\mathrm{KL}}(\boldsymbol{p}^i||\boldsymbol{q}^i) + D_{\mathrm{KL}}(\boldsymbol{q}^i||\boldsymbol{p}^i)\big], \tag{14}$$

where $D_{\mathrm{KL}}(\cdot||\cdot)$ is the Kullback-Leibler (KL) Divergence. Similarly, we can also measure the consistency loss for the generated graph view, and the cross-branch contrast loss can be defined as:

$$\mathcal{L}_{cs} = \ell(z^{gnn}_{G_i}, z^{gk}_{G_i}) + \ell(z^{gnn}_{\widetilde{G}_i}, z^{gk}_{\widetilde{G}_i}). \tag{15}$$

3.4 Adaptive Training and OOD Scoring

By simply adding three different concordance losses, we can train a graph OOD detection model. This model allows us to generate OOD scores for all the test graph samples by examining the predicted errors of the concordance losses, namely, $s^{gnn}_{G_i}, s^{gk}_{G_i}, s^{cs}_{G_i}$. And the ID graph sample is expected to have a lower predicted error, indicating that its latent patterns closely resemble the learned ones.

However, treating the three loss components with equal importance overlooks the varying sensitivities of latent graph patterns, potentially resulting in sub-optimal model performance [23]. On the one hand, individual ID graph datasets often manifest distinct and dataset-specific structural regularities, which correspond to different trade-off weights among loss terms. On the other hand, manually adjusting the trade-off weights between the three concordance losses is not trivial, particularly under unsupervised settings. To achieve this, we utilize the standard deviation of the predicted errors as a surrogate for task uncertainty, thereby allowing the model to automatically calibrate the relative importance of each loss. Concretely, the adaptive loss function is formulated as:

$$\mathcal{L} = (\sigma_{gnn})^\alpha\mathcal{L}_{gnn} + (\sigma_{gk})^\alpha\mathcal{L}_{gk} + (\sigma_{cs})^\alpha\mathcal{L}_{cs} - \beta\mathcal{L}_{reg}, \tag{16}$$

where $\sigma_{gnn}, \sigma_{gk}, \sigma_{cs}$ are the standard deviations of the predicted errors, $\alpha \geq 0$ denotes a hyper-parameter regulating the degree of self-adaptiveness. The final adaptation loss penalizes the loss term associated with a larger deviation, enabling the model to effectively focus on capturing shared implicit and explicit graph patterns.

During the testing phase, we apply z-score normalization based on the means and standard deviations of the predicted errors from the training graph samples to ensure balance, and the final OOD score can be defined as:

$$s_{G_j} = \frac{s^{gnn}_{G_j} - \mu_{gnn}}{\sigma_{gnn}} + \frac{s^{gk}_{G_j} - \mu_{gk}}{\sigma_{gk}} + \frac{s^{cs}_{G_j} - \mu_{cs}}{\sigma_{cs}}, \tag{17}$$

where G_j is the test graph sample, μ_{gnn}, μ_{gk}, and μ_{cs} are the means of the corresponding predicted errors under different concordance losses. By normalizing each predicted error with ID graph samples, we can highlight OOD graph samples with different latent patterns, making them easy to be detected. The detailed learning procedure of our proposed KEGOD is illustrated in Algorithm 1 in the Appendix A.

4 Experiment

4.1 Experimental Settings

Datasets. We choose 10 pairs of datasets from two popular graph data benchmarks (i.e., TU datasets [27] and OGB [12]). The dataset includes 8 molecular graph pairs (i.e., BZR vs. COX2, PTC-MR vs. MUTAG, AIDS vs. DHFR, Tox21 vs. SIDER, FreeSolv vs. ToxCast, BBBP vs. BACE, ClinTox vs. LIPO and Esol vs. MUV), alongside one pair from the bioinformatics domain (ENZYMES vs. PROTEINS) and one from social networks (IMDB-M vs. IMDB-B).

Baselines. To evaluate the performance of our proposed KEGOD, we adopt the following three classes of baselines for comparison: **(A) Graph kernel detectors:** We utilize Weisfeiler-Lehman kernel (WL) [35] and propagation kernel (PK) [28] as the chosen kernels. Additionally, the local outlier factor (LOF) [3], one-class SVM (OCSVM) [26], and isolation forest (iF) [22] are employed as detectors. **(B) Graph concordance detectors:** We adopt two

Table 1: Experimental results of graph OOD detection (AUC in %, mean ± std). The best and runner-up results are marked in bold and underline, respectively.

ID dataset	BZR	PTC-MR	AIDS	ENZYMES	IMDB-M	Tox21	FreeSolv	BBBP	ClinTox	Esol
OOD dataset	COX2	MUTAG	DHFR	PROTEIN	IMDB-B	SIDER	ToxCast	BACE	LIPO	MUV
PK-LOF	42.22±8.39	51.04±6.04	50.15±3.29	50.47±2.87	48.03±2.53	51.33±1.81	49.16±3.70	53.10±2.07	50.00±2.17	50.82±1.48
PK-OCSVM	42.55±8.26	49.71±6.58	50.17±3.30	50.46±2.78	48.07±2.41	51.33±1.81	48.82±3.29	53.05±2.10	50.06±2.19	51.00±1.33
PK-iF	51.46±1.62	54.29±4.33	51.10±1.43	51.67±2.69	50.67±2.47	49.87±0.82	52.28±1.87	51.47±1.33	50.81±1.10	50.85±3.51
WL-LOF	48.99±6.20	53.31±8.98	50.77±2.87	52.66±2.47	52.28±4.50	51.92±1.58	51.47±4.23	52.80±1.91	51.29±3.40	51.26±1.31
WL-OCSVM	49.16±4.51	53.31±7.57	50.98±2.71	51.77±2.21	51.38±2.39	51.08±1.46	50.38±3.81	52.85±2.00	50.77±3.69	50.97±1.65
WL-iF	50.24±2.49	51.43±2.02	50.10±0.44	51.17±2.01	51.07±2.25	50.25±0.96	52.60±2.38	50.78±0.75	50.41±2.17	50.61±1.96
InfoGraph-iF	63.17±9.74	51.43±5.19	93.10±1.35	60.00±1.83	58.73±1.96	56.28±0.81	56.92±1.69	53.68±2.90	48.51±1.87	54.16±5.14
InfoGraph-MD	86.14±6.77	50.79±8.49	69.02±11.67	55.25±3.51	81.38±1.14	59.97±2.06	58.05±5.46	70.49±4.63	48.12±5.72	77.57±1.69
GraphCL-iF	60.00±3.81	50.86±4.30	92.90±1.21	61.33±2.27	59.67±1.65	56.81±0.97	55.55±2.71	59.41±3.58	47.84±0.92	62.12±4.01
GraphCL-MD	83.64±6.00	73.03±2.38	93.75±2.13	52.87±6.11	79.09±2.73	58.30±1.52	60.31±5.24	75.72±1.54	51.58±3.64	78.73±1.40
OCGIN	76.66±4.17	80.38±6.84	86.01±6.59	57.65±2.96	67.93±3.86	46.09±1.66	59.60±4.78	61.21±8.12	49.13±4.13	54.04±5.50
GLocalKD	75.75±5.99	70.63±3.54	93.67±1.24	57.18±2.03	78.25±4.35	66.28±0.98	64.82±3.31	73.15±1.26	55.71±3.81	86.83±2.35
GOOD-D	94.99±2.25	81.21±2.65	99.07±0.40	61.84±1.94	79.94±1.09	66.50±1.35	80.13±3.43	82.91±2.58	69.18±3.61	91.52±0.70
HGOE	95.00±2.70	82.06±1.63	99.28±0.34	64.44±2.19	**81.74±2.25**	68.24±0.60	**82.89±2.33**	83.46±1.79	70.09±1.52	92.64±2.44
CGOD	95.47±3.85	81.42±2.04	98.17±0.28	61.52±1.62	79.02±3.80	69.10±1.58	81.72±1.07	81.75±1.43	65.13±2.61	89.68±3.02
KEGOD	**96.20±0.70**	**86.53±1.70**	**99.40±0.01**	**64.73±1.54**	81.50±1.85	**70.56±1.27**	81.92±2.16	**83.76±1.69**	**76.28±0.73**	**93.08±1.35**

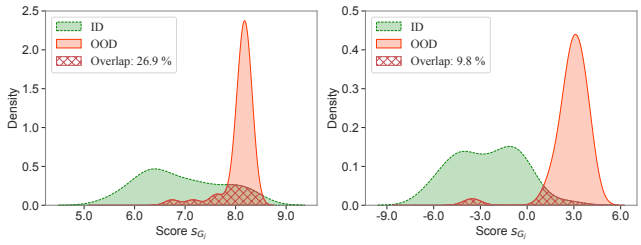
representative graph contrastive learning methods, namely InfoGraph [37] and GraphCL [48]. In addition to the iF detector, we also incorporate the Mahalanobis distance-based (MD) detector [32], which has demonstrated effectiveness in OOD data detection. **(C) Graph OOD detectors:** We also compare our method with recent graph OOD/anomaly detection frameworks (i.e., OCGIN [52], GLocalKD [25], GOOD-D [23], HGOE [15], and CGOD [21]), where the graph-level representation and corresponding detection mechanism are jointly optimized through end-to-end training.

Evaluation Protocol. To evaluate the Graph OOD detection methods, we adopt a widely adopted OOD detection metric, namely Area Under the receiver operating characteristic Curve (AUC). Following the recent works [23], for each dataset, 90% of the ID graph samples are selected as the training set, while the remaining 10% of the ID graph samples and the same number of OOD graph samples are combined as the test set. The results are computed by the mean AUC with standard variation after five repeated runs.

Implementation Details. We utilize GIN [47] as backbone to parameterize both the graph view generator and implicit GNN branch, consisting of three convolution layers and one sum-pooling layer. For our explicit GK branch, we empirically utilize 16 hidden graphs of fixed size (10 nodes each), and constrain the random walk process to a maximum length of $P = 3$. The hidden dimension d is set to 16 for both branches. The hyper-parameter α is tuned amongst $[0.2, 0.8]$ for adaptive training and OOD scoring, τ is set to 1, and p_x is tuned amongst $[0, 1.0]$. We optimize KEGOD with Adam optimizer, the learning rate is initially set to 0.001 and the number of epochs is set to 2000.

4.2 Performance Comparison

We compare our KEGOD with other competitive graph OOD detection baselines, as shown in Table 1. From the comparison results, the insights can be derived as follows: **(i)** Compared to graph kernel detectors, graph concordance learning methods generally exhibit superior performance. **(ii)** Graph OOD detectors, which detect OOD graphs in an end-to-end manner, generally outperform two-stage



(a) Before Adaptive Training **(b) After Adaptive Training**

Figure 3: Scoring distributions on BZR-COX2 with/without adaptive training. Graphs w.r.t. high (low) scores are identified as ID (OOD) samples, and KEGOD enlarges the separation between ID and OOD graphs.

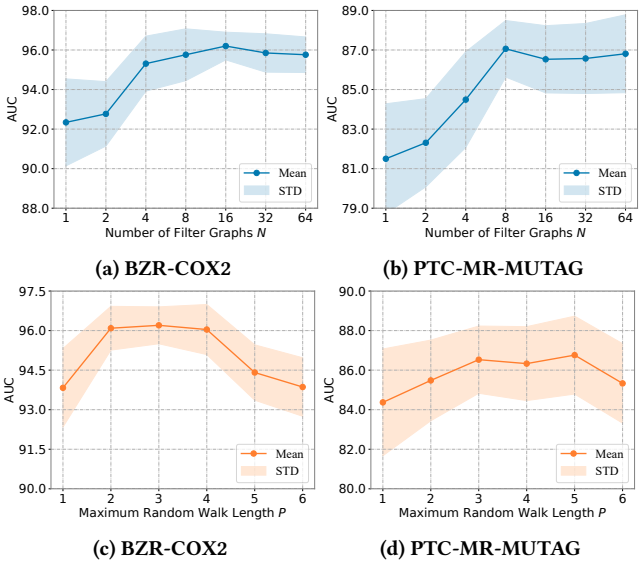
methods in terms of both accuracy and efficiency. **(iii)** Overall, our proposed KEGOD achieves very competitive results on all 10 datasets with an average rank of 1.3. From the results, we deem that the improvement is not only from exploring the implicit and explicit latent substructure of the graph, but also from considering the multi-branch concordance learning with adaptive training. Additionally, we provide the comparison experiments on anomaly detection in Appendix B.

4.3 Ablation Studies

To deeply understand KEGOD, we conduct ablation studies over the key components. Particularly, we introduce a few model variants from two aspects: **(i)** multi-branch graph concordance learning and **(ii)** graph-structure learning. For **(i)**, we conduct experiments on all possible combinations of these concordance losses, along with a variant model that considers all of these concordance losses but treats each component equally (w/o Adaptive Training) to validate the effectiveness of each component. For **(ii)**, we investigate the impact on different types of graph view generation strategies (i.e., Attentive, MLP, GNN). We focus on the former here while dedicating the analysis of the latter in the Appendix C.

Table 2: Ablation study results of KEGOD and its variants (AUC in %, mean ± std).

$\mathcal{L}_{gnn}$	$\mathcal{L}_{gk}$	$\mathcal{L}_{cs}$	BZR	PTC-MR	AIDS	ENZYMES	IMDB-M	Tox21	FreeSolv	BBBP	ClinTox	Esol
			COX2	MUTAG	DHFR	PROTEIN	IMDB-B	SIDER	ToxCast	BACE	LIPO	MUV
✓	-	-	90.24±3.67	82.60±3.15	97.81±0.19	60.44±2.08	74.83±1.88	67.39±1.97	79.34±2.53	81.21±1.29	72.92±2.15	90.49±1.30
-	✓	-	90.63±2.31	82.38±2.08	97.78±0.71	59.81±2.43	74.08±2.04	66.92±1.23	78.57±1.67	80.42±1.59	73.24±2.48	90.92±1.98
-	-	✓	91.45±3.46	80.14±3.38	98.66±0.21	60.61±3.57	73.58±2.69	68.17±1.22	78.44±2.33	78.20±3.34	73.38±1.42	90.45±2.60
✓	✓	-	91.54±1.47	83.92±2.78	98.98±0.01	61.39±2.51	75.82±2.93	69.83±1.31	80.34±4.21	82.92±2.19	75.58±2.16	92.90±1.63
✓	-	✓	93.01±2.54	84.18±4.10	98.84±0.22	61.51±4.17	**84.00±1.37**	69.47±0.87	80.39±2.60	82.30±4.54	74.86±2.70	91.37±2.45
-	✓	✓	94.47±0.70	83.91±0.80	98.65±0.08	63.47±2.08	77.10±1.81	69.79±0.90	79.05±0.60	81.25±2.31	74.69±2.00	92.07±2.30
w/o Adaptive Training			93.63±2.00	85.04±2.83	97.39±1.06	62.91±1.15	77.42±2.44	69.03±1.35	80.91±2.95	83.07±0.88	75.57±1.17	92.71±1.39
KEGOD			**96.20±0.70**	**86.53±1.70**	**99.40±0.01**	**64.73±1.54**	81.50±1.85	**70.56±1.27**	**81.92±2.56**	**83.76±1.69**	**76.28±0.73**	**93.08±1.35**



(a) BZR-COX2 (b) PTC-MR-MUTAG

(c) BZR-COX2 (d) PTC-MR-MUTAG

Figure 4: AUC w.r.t. number of filter graphs N (top) and maximum random walk length P (bottom) in four datasets.

Effect of Multi-branch Concordance Learning. The compared results on all 10 datasets are shown in Table 2 and we make the following observations. **(i)** Different concordance learning strategies contribute significantly to the OOD detection task. Meanwhile, compared with the individual concordance learning strategy, integrating them at two branches generally leads to better performance. This highlights the importance of leveraging complementary information from multiple perspectives. **(ii)** Compared with adaptive learning, directly adding three concordance losses may lead to suboptimal performance. The result is also consistent with the scoring distribution comparison in Figure 3, where we can observe that adaptive training indeed enlarges the distribution gap between the ID and OOD graphs for better detection.

4.4 Parameter Sensitivity

We also examine the sensitivity of KEGOD to varying numbers of filter graphs, different maximum random walk lengths and diverse self-adaptiveness strengths. We focus on the first two discussions here while dedicating the latter in the Appendix D.

Effect of the Number of Filter Graphs. To analyze whether KEGOD could benefit from explicit GK branch, we vary the number of filter graphs N in $\{1, 2, 4, 8, 16, 32, 64\}$ while keeping all other hyper-parameters fixed. Figure 4 (top) summarizes the experimental results and it can be observed that as the value of N increases, KEGOD achieves better performance, especially when the value is small. The reason for this improvement is that increasing the number of filter graphs would help to extract high-order structure information. However, when N becomes too large (i.e., $N > 16$), the improvement becomes marginal. Therefore, to strike a balance between performance and computational complexity, we set N to 16 in our model. More evidence is provided in Figure 9 in the Appendix.

Effect of the Maximum Random Walk Length. We also vary the maximum random walk length P in the range of $[1, 6]$ while fixing all other hyper-parameters. As shown in Figure 4 (bottom), under the setting that KEGOD with 16 filter graphs and each graph has 10 nodes, the performance exhibits an initial increase followed by a decline. This may be that an excessively small P would restrict topological exploration space while a large P introduces instability and fails to extract distinguishable substructures for OOD detection.

4.5 Visualization and Case Study

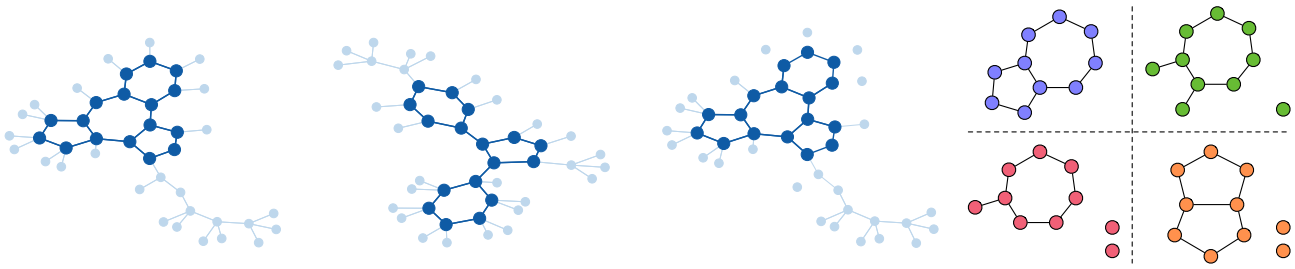
We conduct two qualitative analyses to study how KEGOD facilitates graph OOD detection, including the case study of the learned graph substructures and the visualization of the learned graph representations under two branches.

Case Study. To further understand KEGOD, Figure 5 shows the ID and OOD graph on the BZR-COX2 dataset and the corresponding learned graph views and substructures. We observe that the ID graph exhibits rings with common edges in the backbone, whereas rings in the OOD graph are independent. Consequently, the learned graph views and substructures on BZR-COX2 emphasize the graph backbone rather than side chains. Overall, the graph views and substructures highlight key positions conducive to OOD detection, thereby enlarging the difference between ID and OOD graphs.

Visualization. Figure 6 visualizes the learned representations under different branches via t-distributed stochastic neighbor embedding (t-SNE) [40] on the BZR-COX2 dataset. It can be observed that representations of ID/OOD graphs fail to clearly separate from each other when only trained via implicit or explicit concordance learning (yellow and purple dots). While representations that are trained via multi-branch concordance are more separable, which demonstrates the effectiveness of our graph OOD detection method.

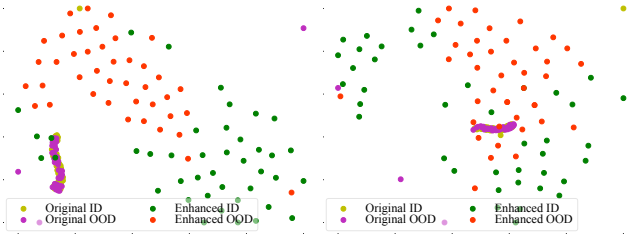
5 Related Work

Graph Neural Networks & Graph Kernels. Thanks to the development of deep learning, GNNs provide a category of powerful models designed for handling graph-structured data [14, 18, 33]. The fundamental idea behind GNNs is to capture information within



(a) ID (b) OOD (c) Augmented ID View (d) Learned Substructures

Figure 5: Visualization of ID/OOD graphs on the BZR-COX2 dataset, along with the corresponding augmented graph view and the learned substructures. We only draw the edge with the learned weight over 0.5.



(a) Implicit Branch (b) Explicit Branch

Figure 6: Visualization of learned representations under different branches on BZR-COX2 dataset. The ID and OOD graph representations (yellow v.s. purple) become more separable after the multi-branch enhancement (green v.s. red).

the graph by learning the relationships between nodes. GNNs define a series of layers on the graph, where each layer updates the node representations by considering information from neighboring nodes [41, 47]. This information propagation enables nodes to aggregate global information from the entire graph, allowing the model to understand and leverage the underlying structure [7]. Unlike GNNs, which learn representations through iterative information propagation, Graph Kernels (GKs) operate by directly computing a similarity measure between graphs [42]. Common GK methods include the random walk kernel, which measures the similarity between graphs through random walk statistics [16], and the graphlet kernel, which considers the distribution of small subgraph patterns [36]. The Weisfeiler-Lehman kernel applies an iterative labeling process to nodes, capturing the graph's structural information [35]. Our proposed KEGOD robustly identifies anomalies or deviations in OOD graphs by explicitly and implicitly exploring latent substructures.

Graph Out-of-distribution Detection. The goal of OOD detection is to distinguish samples at test time that significantly deviate from the training distribution. Depending on the availability of labels during the training phase, existing OOD detection methods can be divided into two categories, i.e., supervised [10, 19, 20] and unsupervised methods [24, 30–32, 53]. Due to the high cost and huge space required for label annotation [32], we focus on the unsupervised OOD detection task in this paper. Recently, a few studies have concentrated on graph-level OOD detection. GOOD-D [23] utilizes hierarchical contrastive learning to detect OOD graphs. SEGO [11] further introduces a coding tree form as an anchor view for the detection. GOODAT [43] implements a test-time method by extracting informative subgraphs related to the predicted pseudo-labels on the test set. HGOE [15] integrates realistic graph data from an external distribution while synthesizing internal outliers within ID data. Different from existing methods implicitly modeling graph topology, our proposed KEGOD also considers structural information explicitly via GKs.

6 Conclusion

In this paper, we propose a kernel-enhanced latent substructure learning framework termed KEGOD for graph OOD detection. Specifically, we propose a learnable graph generator to produce augmented graph views that preserve important structural and semantic information. Then, KEGOD is featured by two branches, i.e., a GNN branch and a GK branch, which explore graph information in implicit and explicit manners, respectively. Furthermore, we integrate the two branches into a multi-branch concordance learning framework where implicit and explicit branch concordance aim to get the OOD scores via mutual agreement between the graph and its augmented view, whereas cross-branch concordance strives to get the OOD scores via branch discrepancy. Finally, a self-adaptive strategy is employed to adjust the trade-off between learning objectives and learned OOD scores. Extensive experiments on several real-world benchmarks demonstrate the superior efficacy of our proposed KEGOD. In future work, we aim to integrate large language models [49, 50] into our KEGOD and apply the model to other realistic graph-related downstream applications.

Acknowledgments

This paper is partially supported by the National Key Research and Development Program of China with Grant No. 2023YFC3341203, the Fundamental Research Funds for the Central Universities in UIBE (Grant No. 23QN02), the Humanities and Social Sciences Research Fund of the Ministry of Education of China under Grant 25YJCZH275, the National Natural Science Foundation of China under Grant 62276002, 62306014 and 12501344, the Postdoctoral Fellowship Program (Grade A) of CPSF under Grant BX20250376 and BX20240239, the Sichuan Science and Technology Program under Grant 2025ZNSFSC1506 and 2025ZNSFSC0808, the Fundamental Research Funds for the Central Universities under Grant 1082204112K97, and the Sichuan University Interdisciplinary Innovation Fund.

References

[1] Karsten M Borgwardt and Hans-Peter Kriegel. 2005. Shortest-path kernels on graphs. In *Proceedings of the International Conference on Data Mining*. 8–pp.

[2] Karsten M Borgwardt, Cheng Soon Ong, Stefan Schönauer, SVN Vishwanathan, Alex J Smola, and Hans-Peter Kriegel. 2005. Protein function prediction via graph kernels. *Bioinformatics* 21 (2005), i47–i56.

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

[4] Rich Caruana, Yin Lou, Johannes Gehrke, Paul Koch, Marc Sturm, and Noemie Elhadad. 2015. Intelligible models for healthcare: Predicting pneumonia risk and hospital 30-day readmission. In *Proceedings of the International ACM SIGKDD Conference on Knowledge Discovery & Data Mining*. 1721–1730.

[5] Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. 2020. A simple framework for contrastive learning of visual representations. In *Proceedings of the International Conference on Machine Learning*. 1597–1607.

[6] Kevin Eykholt, Ivan Evtimov, Earlence Fernandes, Bo Li, Amir Rahmati, Chaowei Xiao, Atul Prakash, Tadayoshi Kohno, and Dawn Song. 2018. Robust physical-world attacks on deep learning visual classification. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 1625–1634.

[7] Justin Gilmer, Samuel S Schoenholz, Patrick F Riley, Oriol Vinyals, and George E Dahl. 2017. Neural message passing for quantum chemistry. In *Proceedings of the International Conference on Machine Learning*. 1263–1272.

[8] Yuxin Guo, Cheng Yang, Yuluo Chen, Jixi Liu, Chuan Shi, and Junping Du. 2023. A Data-centric Framework to Endow Graph Neural Networks with Out-Of-Distribution Detection Ability. In *Proceedings of the International ACM SIGKDD Conference on Knowledge Discovery & Data Mining*. 638––648.

[9] Zhongkai Hao, Chengqiang Lu, Zhenya Huang, Hao Wang, Zheyuan Hu, Qi Liu, Enhong Chen, and Cheekong Lee. 2020. ASGN: An active semi-supervised graph neural network for molecular property prediction. In *Proceedings of the International ACM SIGKDD Conference on Knowledge Discovery & Data Mining*. 731–752.

[10] Dan Hendrycks and Kevin Gimpel. 2017. A baseline for detecting misclassified and out-of-distribution examples in neural networks. In *Proceedings of the International Conference on Learning Representations*.

[11] Yue Hou, He Zhu, Ruomei Liu, Yingke Su, Jinxiang Xia, Junran Wu, and Ke Xu. 2025. Structural Entropy Guided Unsupervised Graph Out-Of-Distribution Detection. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 39. 17258–17266.

[12] Weihua Hu, Matthias Fey, Marinka Zitnik, Yuxiao Dong, Hongyu Ren, Bowen Liu, Michele Catasta, and Jure Leskovec. 2020. Open graph benchmark: Datasets for machine learning on graphs. In *Proceedings of the Conference on Neural Information Processing Systems*. 22118–22133.

[13] Eric Jang, Shixiang Gu, and Ben Poole. 2017. Categorical reparameterization with gumbel-softmax. (2017).

[14] Wei Ju, Zheng Fang, Yiyang Gu, Zequn Liu, Qingqing Long, Ziyue Qiao, Yifang Qin, Jianhao Shen, Fang Sun, Zhiping Xiao, et al. 2024. A comprehensive survey on deep graph representation learning. *Neural Networks* 173 (2024), 106207.

[15] He Junwei, Qianqian Xu, Yangbangyan Jiang, Zitai Wang, Yuchen Sun, and Qingming Huang. 2024. HGOE: Hybrid External and Internal Graph Outlier Exposure for Graph Out-of-Distribution Detection. 1544–1553.

[16] Hisashi Kashima, Koji Tsuda, and Akihiro Inokuchi. 2003. Marginalized kernels between labeled graphs. In *Proceedings of the International Conference on Machine Learning*. 321–328.

[17] Thomas N Kipf and Max Welling. 2017. Semi-supervised classification with graph convolutional networks. In *Proceedings of the International Conference on Learning Representations*.

[18] Bingheng Li, Erlin Pan, and Zhao Kang. 2024. Pc-conv: Unifying homophily and heterophily with two-fold filtering. In *Proceedings of the AAAI Conference on Artificial Intelligence*. 13437–13445.

[19] Zenan Li, Qitian Wu, Fan Nie, and Junchi Yan. 2022. Graphde: A generative framework for debiased learning and out-of-distribution detection on graphs. In *Proceedings of the Conference on Neural Information Processing Systems*. 30277–30290.

[20] Shiyu Liang, Yixuan Li, and Rayadurgam Srikant. 2018. Enhancing the reliability of out-of-distribution image detection in neural networks. In *Proceedings of the International Conference on Learning Representations*.

[21] Xixun Lin, Yanan Cao, Nan Sun, Lixin Zou, Chuan Zhou, Peng Zhang, Shuai Zhang, Ge Zhang, and Jia Wu. 2025. Conformal graph-level out-of-distribution detection with adaptive data augmentation. In *Proceedings of the Web Conference*. 4755–4765.

[22] Fei Tony Liu, Kai Ming Ting, and Zhi-Hua Zhou. 2008. Isolation forest. In *Proceedings of the International Conference on Data Mining*. 413–422.

[23] Yixin Liu, Kaize Ding, Huan Liu, and Shirui Pan. 2023. Good-d: On unsupervised graph out-of-distribution detection. In *Proceedings of the International ACM Conference on Web Search & Data Mining*. 339–347.

[24] Xuexiong Luo, Jia Wu, Jian Yang, Shan Xue, Hao Peng, Chuan Zhou, Hongyang Chen, Zhao Li, and Quan Z Sheng. 2022. Deep graph level anomaly detection with contrastive learning. *Scientific Reports* 12, 1 (2022), 19867.

[25] Rongrong Ma, Guansong Pang, Ling Chen, and Anton van den Hengel. 2022. Deep graph-level anomaly detection by glocal knowledge distillation. In *Proceedings of the International ACM Conference on Web Search & Data Mining*. 704–714.

[26] Larry M Manevitz and Malik Yousef. 2001. One-class SVMs for document classification. *Journal of Machine Learning Research* 2, Dec (2001), 139–154.

[27] Christopher Morris, Nils M Kriege, Franka Bause, Kristian Kersting, Petra Mutzel, and Marion Neumann. 2020. Tudataset: A collection of benchmark datasets for learning with graphs. *arXiv preprint arXiv:2007.08663* (2020).

[28] Marion Neumann, Roman Garnett, Christian Bauckhage, and Kristian Kersting. 2016. Propagation kernels: efficient graph kernels from propagated information. *Machine learning* 102 (2016), 209–245.

[29] Giannis Nikolentzos and Michalis Vazirgiannis. 2020. Random walk graph neural networks. In *Proceedings of the Conference on Neural Information Processing Systems*. 16211–16222.

[30] Jie Ren, Peter J Liu, Emily Fertig, Jasper Snoek, Ryan Poplin, Mark Depristo, Joshua Dillon, and Balaji Lakshminarayanan. 2019. Likelihood ratios for out-of-distribution detection. In *Proceedings of the Conference on Neural Information Processing Systems*. 14707–14718.

[31] Lukas Ruff, Robert Vandermeulen, Nico Goernitz, Lucas Deecke, Shoaib Ahmed Siddiqui, Alexander Binder, Emmanuel Müller, and Marius Kloft. 2018. Deep one-class classification. In *Proceedings of the International Conference on Machine Learning*. 4393–4402.

[32] Vikash Sehwag, Mung Chiang, and Prateek Mittal. 2021. Ssd: A unified framework for self-supervised outlier detection. In *Proceedings of the International Conference on Learning Representations*.

[33] Zhixiang Shen and Zhao Kang. 2025. When heterophily meets heterogeneous graphs: Latent graphs guided unsupervised representation learning. *IEEE Transactions on Neural Networks and Learning Systems* 36, 6 (2025), 10283–10296.

[34] Zhixiang Shen, Shuo Wang, and Zhao Kang. 2024. Beyond redundancy: Information-aware unsupervised multiplex graph structure learning. In *Proceedings of the Conference on Neural Information Processing Systems*. 31629–31658.

[35] Nino Shervashidze, Pascal Schweitzer, Erik Jan Van Leeuwen, Kurt Mehlhorn, and Karsten M Borgwardt. 2011. Weisfeiler-lehman graph kernels. *Journal of Machine Learning Research* 12, 9 (2011), 2539–2561.

[36] Nino Shervashidze, SVN Vishwanathan, Tobias Petri, Kurt Mehlhorn, and Karsten Borgwardt. 2009. Efficient graphlet kernels for large graph comparison. In *Proceedings of the International Conference on Artificial Intelligence and Statistics*. 488–495.

[37] Fan-Yun Sun, Jordan Hoffmann, Vikas Verma, and Jian Tang. 2019. Infograph: Unsupervised and semi-supervised graph-level representation learning via mutual information maximization. In *Proceedings of the International Conference on Learning Representations*.

[38] Jihoon Tack, Sangwoo Mo, Jongheon Jeong, and Jinwoo Shin. 2020. Csi: Novelty detection via contrastive learning on distributionally shifted instances. In *Proceedings of the Conference on Neural Information Processing Systems*. 11839–11852.

[39] Ruo-Chun Tzeng and Shan-Hung Wu. 2019. Distributed, egocentric representations of graphs for detecting critical structures. In *Proceedings of the International Conference on Machine Learning*. 6354–6362.

[40] Laurens Van der Maaten and Geoffrey Hinton. 2008. Visualizing data using t-SNE. *Journal of Machine Learning Research* 9, 11 (2008).

[41] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Lio, and Yoshua Bengio. 2018. Graph attention networks. In *Proceedings of the International Conference on Learning Representations*.

[42] S Vichy N Vishwanathan, Nicol N Schraudolph, Risi Kondor, and Karsten M Borgwardt. 2010. Graph kernels. *Journal of Machine Learning Research* 11 (2010), 1201–1242.

[43] Luzhi Wang, Dongxiao He, He Zhang, Yixin Liu, Wenjie Wang, Shirui Pan, Di Jin, and Tat-Seng Chua. 2024. GOODAT: Towards Test-Time Graph Out-of-Distribution Detection. In *Proceedings of the AAAI Conference on Artificial Intelligence*. 15537–15545.

[44] Shuo Wang, Shunyang Huang, Jinghui Yuan, Zhixiang Shen, and Zhao Kang. 2025. Cooperation of Experts: Fusing Heterogeneous Information with Large Margin. In *Proceedings of the International Conference on Machine Learning*.

[45] Shuo Wang, Bokui Wang, Zhixiang Shen, Boyan Deng, and Zhao Kang. 2025. Multi-domain graph foundation models: Robust knowledge transfer via topology alignment. In *Proceedings of the International Conference on Machine Learning*.

[46] Yifan Wang, Suyao Tang, Yuntong Lei, Weiping Song, Sheng Wang, and Ming Zhang. 2020. Disenhan: Disentangled heterogeneous graph attention network for recommendation. In *Proceedings of the International Conference on Information and Knowledge Management*. 1605–1614.

[47] Keyulu Xu, Weihua Hu, Jure Leskovec, and Stefanie Jegelka. 2019. How powerful are graph neural networks?. In *Proceedings of the International Conference on Learning Representations*.

[48] Yuning You, Tianlong Chen, Yongduo Sui, Ting Chen, Zhangyang Wang, and Yang Shen. 2020. Graph contrastive learning with augmentations. In *Proceedings*

of the Conference on Neural Information Processing Systems. 5812–5823.
[49] Ye Yuan, Chengwu Liu, Jingyang Yuan, Gongbo Sun, Siqi Li, and Ming Zhang. 2024. A hybrid RAG system with comprehensive enhancement on complex reasoning. *arXiv preprint arXiv:2408.05141* (2024).
[50] Ye Yuan, Kexin Tang, Jianhao Shen, Ming Zhang, and Chenguang Wang. 2024. Measuring Social Norms of Large Language Models. *Findings of the Association for Computational Linguistics: NAACL*, 650–699.
[51] Ge Zhang, Zhenyu Yang, Jia Wu, Jian Yang, Shan Xue, Hao Peng, Jianlin Su, Chuan Zhou, Quan Z Sheng, Leman Akoglu, et al. 2022. Dual-discriminative graph neural network for imbalanced graph-level anomaly detection. In *Proceedings of the Conference on Neural Information Processing Systems*. 24144–24157.
[52] Lingxiao Zhao and Leman Akoglu. 2023. On using classification datasets to evaluate graph outlier detection: Peculiar observations and new insights. *Big Data* 11, 3 (2023), 151–180.
[53] Wenxuan Zhou, Fangyu Liu, and Muhao Chen. 2021. Contrastive out-of-distribution detection for pretrained transformers. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*. 1100—-1111.
[54] Yanqiao Zhu, Yichen Xu, Feng Yu, Qiang Liu, Shu Wu, and Liang Wang. 2021. Graph contrastive learning with adaptive augmentation. In *Proceedings of the Web Conference*. 2069–2080.

A Training Details

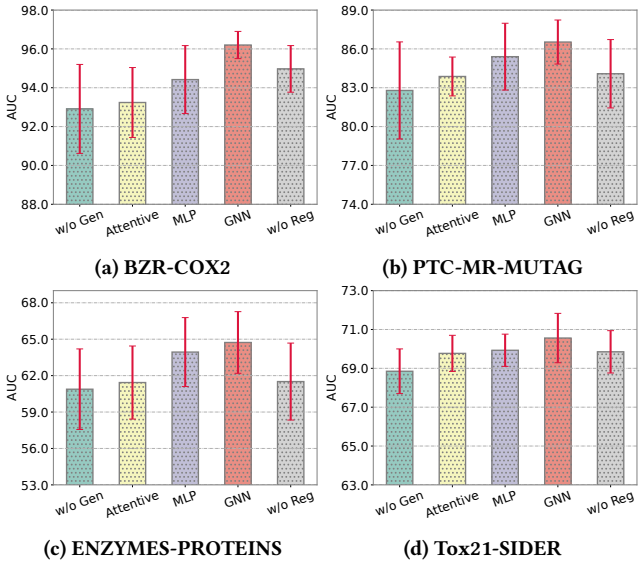
Algorithm 1: Kernel-based Graph OOD Detection

Input: Training set of graph samples $\mathcal{D} = \mathcal{D}^{in} \cup \mathcal{D}^{out}$, Test graph sample G_j
Output: KEGOD with learned parameters, G_j's OOD score s_{G_j}.

Sample T graph samples from $\mathcal{D}$ to construct anchor set as the memory bank. ;
while *not convergence* **do**
  Sample minibatch graph samples $\mathcal{B}$. ;
  Graph augmentation for $G_i \in \mathcal{B}$ with learnable graph view generator to get $\widetilde{G}_i$ and $\mathcal{L}_{reg}$. ;
  /* Eq. 6 */
  Obtain implicit graph representation $\boldsymbol{h}^{gnn}_{G_i}$ and $\boldsymbol{h}^{gnn}_{\widetilde{G}_i}$ under implicit GNN branch. ;
  /* Eq. 8- 10 */
  Obtain explicit graph representation $\boldsymbol{h}^{gk}_{G_i}$ and $\boldsymbol{h}^{gk}_{\widetilde{G}_i}$ under explicit GK branch. ;
  /* Eq. 11- 15 */
  Compute concordance loss term $\mathcal{L}_{gnn}, \mathcal{L}_{gk}, \mathcal{L}_{cs}$. ;
  /* Eq. 16 */
  Optimize framework via adaptive training. ;
end
/* Eq. 17 */
Calculate OOD score s_{G_j} with z-score normalization. ;
return Learned parameters for KEGOD, s_{G_j}. ;

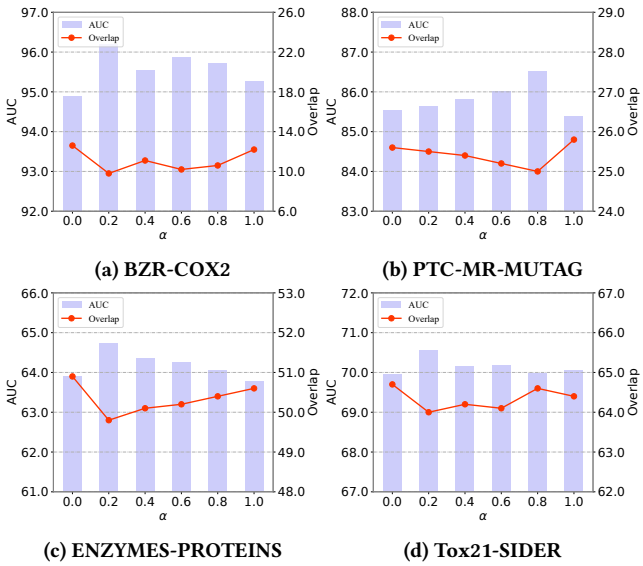
B Anomaly Detection

To investigate whether our proposed KEGOD can generalize to anomaly detection settings. we also conduct anomaly detection experiments on 15 datasets following the previous work [23, 25]. The results are shown in Table 3, we can observe that our proposed method also achieves competitive performance in anomaly detection settings. We attribute the performance improvement to the following two factors.



(a) BZR-COX2 (b) PTC-MR-MUTAG
(c) ENZYMES-PROTEINS (d) Tox21-SIDER

Figure 7: AUC w.r.t. different graph structure learning methods in four datasets.



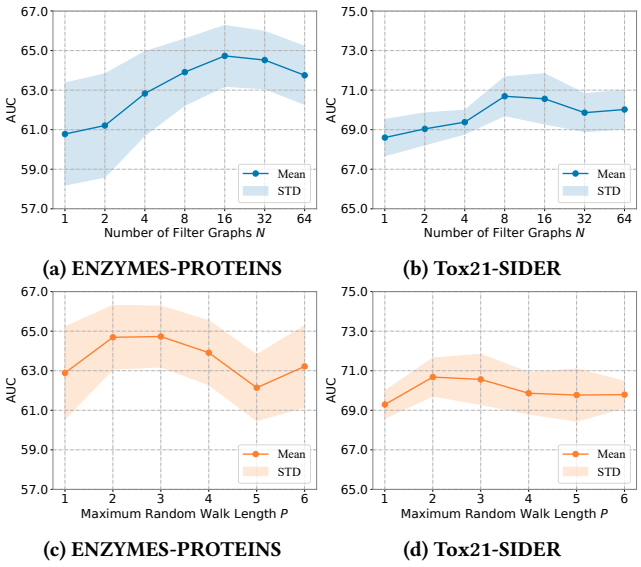
(a) BZR-COX2 (b) PTC-MR-MUTAG
(c) ENZYMES-PROTEINS (d) Tox21-SIDER

Figure 8: AUC and overlap score w.r.t. different self-adaptivenes strength α in four datasets.

- The graph view generator produces task-specific augmented graph views in a learnable manner. Our proposed generator can adaptively preserve crucial structural and semantic information while avoiding unnecessary perturbations under various task settings for the different datasets.
- The effectiveness of capturing shared latent patterns shared by the ID graphs. We investigate latent patterns through both implicit and explicit approaches, significantly enhancing the discriminability of the graph.

Table 3: Experimental results of anomaly detection (AUC in %, mean ± std). The best and runner-up results are marked in bold and underline, respectively.

Method	PK-OCSVM	PK-iF	WL-OCSVM	WL-iF	InfoGraph-iF	GraphCL-iF	OCGIN	GLocalKD	GOOD-D	HGOE	KEGOD
PROTEINS-full	50.49±4.92	60.70±2.55	51.35±4.35	61.36±2.54	57.47±3.03	60.18±2.53	70.89±2.44	77.30±5.15	71.97±3.86	73.13±0.46	**77.48**±2.27
ENZYMES	53.67±2.66	51.30±2.01	55.24±2.66	51.60±3.81	53.80±4.50	53.60±4.88	58.75±5.98	61.39±8.81	63.90±3.69	67.28±0.99	**67.63**±1.67
AIDS	50.79±4.30	51.84±2.87	50.12±3.43	61.13±0.71	70.19±5.03	79.72±3.98	78.16±3.05	93.27±4.19	97.28±0.69	97.84±0.55	**98.27**±2.21
DHFR	47.91±3.76	52.11±3.96	50.24±3.13	50.29±2.77	52.68±3.21	51.10±2.35	49.23±3.05	56.71±3.57	62.67±3.11	64.39±0.68	**64.85**±2.36
BZR	46.85±5.31	55.32±6.18	50.56±5.87	52.46±3.30	63.31±8.52	60.24±5.37	65.91±1.47	69.42±7.78	75.16±5.15	**80.54**±1.35	79.60±4.56
COX2	50.27±7.91	50.05±2.06	49.86±7.43	50.27±0.34	53.36±8.86	52.01±3.17	53.58±5.05	59.37±12.67	62.65±8.14	69.52±2.68	**74.35**±2.77
DD	48.30±3.98	71.32±2.41	47.99±4.09	70.31±1.09	55.80±1.77	59.32±3.92	72.27±1.83	**80.12**±5.24	73.25±3.19	76.95±2.24	78.94±4.00
NCI1	49.90±1.18	50.58±1.38	50.63±1.22	50.74±1.70	50.10±0.87	49.88±0.53	71.98±1.21	68.48±2.39	61.12±2.21	65.82±1.43	**72.25**±1.93
IMDB-B	50.75±3.10	50.80±3.17	54.08±5.19	50.20±0.40	56.50±3.58	56.50±4.90	60.19±8.90	52.09±3.41	65.88±0.75	69.82±1.37	**71.60**±2.42
REDDIT-B	45.68±2.24	46.72±3.42	49.31±2.33	48.26±0.32	68.50±5.56	71.80±4.38	75.93±8.65	77.85±2.62	88.67±1.24	**89.41**±1.21	88.04±1.18
COLLAB	49.59±2.24	50.49±1.72	52.60±2.56	50.69±0.32	46.27±0.73	47.61±1.29	60.70±2.97	52.94±0.85	72.08±0.90	74.24±0.73	**75.47**±0.97
HSE	57.02±8.42	56.87±10.51	62.72±10.13	53.02±5.12	53.56±3.98	51.18±2.71	64.84±4.70	59.48±1.44	69.65±2.14	**74.50**±3.73	71.99±2.50
MMP	46.65±6.31	50.06±3.73	55.24±3.26	52.68±3.34	54.59±2.01	54.54±1.86	71.23±0.16	67.84±0.59	70.51±1.56	71.94±0.54	**73.28**±1.00
p53	46.74±4.88	50.69±2.02	54.59±4.46	50.85±2.16	52.66±1.95	53.29±2.32	58.50±0.37	64.20±0.81	62.99±1.55	64.70±1.16	**65.27**±1.71
PPAR-gamma	53.94±6.94	45.51±2.58	57.91±6.13	49.60±0.22	51.40±2.53	50.30±1.56	71.19±4.28	64.59±0.67	67.34±1.71	71.92±4.17	**72.06**±2.92



(a) ENZYMES-PROTEINS (b) Tox21-SIDER

(c) ENZYMES-PROTEINS (d) Tox21-SIDER

Figure 9: AUC w.r.t. number of filter graphs N (top) and maximum random walk length P (bottom) in four datasets.

C Ablation Study

Effect of Graph Structure Learning. In KEGOD, we construct augmented graph views via the proposed learnable graph generators. To thoroughly investigate whether graph OOD detection can benefit from our proposed graph structure learning approach, we select three different types of graph view generation strategies (i.e., Attentive, MLP, GNN) and two ablation variants: one replaces the learnable augmentations with pre-defined graph augmentations (w/o Gen) and one removes the dropped edge regularization (w/o Reg) from GNN generators. The comparison results are shown in Figure 7 and we have following observations:

- The performance decreases when either using pre-defined graph augmentations or removing the dropped edge regularization, which indicates that learnable graph generators could preserve more distinguishable shared ID graph patterns for the downstream OOD detection task.
- Moreover, the GNN generator consistently outperforms the other two graph view generation strategies (Attentive, MLP). This demonstrates the reason why we chose the GNN generator as the backbone of the framework.

D Parameter Analysis

Effect of the Self-adaptiveness Strength. To further investigate how the various self-adaptiveness strengths affect adaptive training performance, we evaluate the model performance with varying α in the range of $[0, 1]$. The AUC and overlap area w.r.t. different selections of α are shown in Figure 8. We notice a negative correlation between the overlap area and the model performance. The smaller the overlap between the ID and OOD graphs, the higher the AUC of the model. Furthermore, we observe that the AUC value remains stable across different values of α, except for a slight drop when $\alpha = 0$. This further indicates the significance of the self-adaptive mechanism for our loss function, allowing the model to better align with the task-specific requirements.