

# SGOOD: Substructure-enhanced Graph-Level Out-of-Distribution Detection

Zhihao Ding  
tommy-zh.ding@connect.polyu.hk  
The Hong Kong Polytechnic  
University

Jieming Shi\*  
jieming.shi@polyu.edu.hk  
The Hong Kong Polytechnic  
University

Shiqi Shen  
shiqishen@tencent.com  
Tencent Inc.

Xuequn Shang  
shang@nwpu.edu.cn  
The Northwestern Polytechnical  
University

Jiannong Cao  
csjcao@comp.polyu.edu.hk  
The Hong Kong Polytechnic  
University

Zhipeng Wang  
markrocwang@tencent.com  
Tencent Inc.

Zhi Gong  
davidgong@tencent.com  
Tencent Inc.

## ABSTRACT

Graph-level representation learning is important in a wide range of applications. Existing graph-level models are generally built on i.i.d. assumption for both training and testing graphs. However, in an open world, models can encounter out-of-distribution (OOD) testing graphs that are from different distributions unknown during training. A trustworthy model should be able to detect OOD graphs to avoid unreliable predictions, while producing accurate in-distribution (ID) predictions. To achieve this, we present SGOOD, a novel graph-level OOD detection framework. We find that substructure differences commonly exist between ID and OOD graphs, and design SGOOD with a series of techniques to encode task-agnostic substructures for effective OOD detection. Specifically, we build a super graph of substructures for every graph, and develop a two-level graph encoding pipeline that works on both original graphs and super graphs to obtain substructure-enhanced graph representations. We then devise substructure-preserving graph augmentation techniques to further capture more substructure semantics of ID graphs. Extensive experiments against 11 competitors on numerous graph datasets demonstrate the superiority of SGOOD, often surpassing existing methods by a significant margin. The code is available at <https://anonymous.4open.science/r/SGOOD-0958>.

## KEYWORDS

Out-of-distribution Detection, Trustworthy Model, Reliability, Graph Classification

\*Corresponding Author.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than the author(s) must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from [permissions@acm.org](mailto:permissions@acm.org).  
CIKM '24, October 21 – 25, 2024, Boise, Idaho, USA

© 2018 Copyright held by the owner/author(s). Publication rights licensed to ACM.  
ACM ISBN 978-1-4503-XXXX-X/18/06  
<https://doi.org/XXXXXXXX.XXXXXXX>

## ACM Reference Format:

Zhihao Ding, Jieming Shi, Shiqi Shen, Xuequn Shang, Jiannong Cao, Zhipeng Wang, and Zhi Gong. 2018. SGOOD: Substructure-enhanced Graph-Level Out-of-Distribution Detection. In *Proceedings of International Conference on Information and Knowledge Management (CIKM '24)*. ACM, New York, NY, USA, 11 pages. <https://doi.org/XXXXXXXX.XXXXXXX>

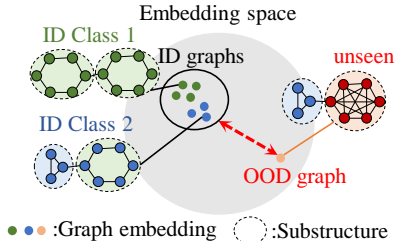
## 1 INTRODUCTION

Graphs are widely used to represent complex structured data, *e.g.*, chemical compounds, proteins, and social networks. Graph-level representation learning, which extracts meaningful representations of these graphs, is crucial for applications in biochemistry [16, 29] and social network analysis [7, 30]. Existing graph-level learning models are based on the closed-world assumption, in which testing graphs encountered at deployment are drawn i.i.d. from the same distribution as the training graph data. However, in reality, the models are actually in an *open world*, where test graphs can come from different, previously unseen distributions, making them out-of-distribution (OOD) *w.r.t.* in-distribution (ID) training graphs [20, 21, 38]. Consequently, the models trained solely by ID graphs tend to make incorrect predictions on OOD data [13], which raises reliability concerns in safety-critical applications, *e.g.*, drug discovery [1]. A trustworthy graph-level learning model should be capable of identifying OOD test graphs to avoid unreliable predictions.

While initial efforts have been made to explore graph-level OOD detection [10, 21, 24], these methods primarily rely on message-passing graph neural networks (GNNs) [12, 17] to first get node representations and then generate graph-level representations solely based on these nodes. Substructure patterns, which are high-level graph structures and contain rich graph semantics [38, 44], have yet to be leveraged for graph-level OOD detection. Intuitively, an OOD detector that can distinguish both node-level structures and substructure patterns would be more effective. However, leveraging substructures for OOD detection is challenging due to the absence of OOD graphs during training, making it difficult to determine in advance which substructure patterns should be learned for identifying OOD samples. Existing substructure learning methods, *e.g.*, hierarchical GNNs [9, 18, 39] and subgraph GNNs [42, 46], typically learn *task-specific* substructures that are tailored to discriminate

**Table 1: The percentage of OOD graphs with substructures never appeared in ID graphs.**

Data	ENZYMES	IMDB-M	IMDB-B	BACE	BBBP	DrugOOD
	58.9%	14.0%	8.5%	50.0%	44.6%	77.3%

**Figure 1: Substructure-enhanced graph-level OOD detection**

between labeled classes in the training set. We argue that such task-specific substructures are not sufficient for OOD detection. Instead, the detector should be able to learn a diverse set of substructure patterns from ID training graphs, including *task-agnostic* substructures not associated with specific classification tasks. As illustrated in Figure 1, the two ID graphs can be classified into different classes based on the presence of a task-specific triangle substructure (in blue). However, to identify the OOD graph, we need to compare its substructures to the task-agnostic ones such as the 6-node cycle (in green), to detect the irregular substructure (in orange). Therefore, effectively leveraging substructures for graph-level OOD detection requires considering task-agnostic information [32], a capability lacking in existing OOD detection methods.

In this paper, we develop SGOOD, a novel framework that explicitly encodes task-agnostic substructures and their relationships into effective representations for graph-level OOD detection. The design of SGOOD is supported by empirical findings that demonstrate the crucial role of task-agnostic substructures in distinguishing between ID and OOD graphs. Given a dataset of graphs (see Table 2 for data statistics), we apply community detection [3] to extract task-agnostic substructures. The substructures are task-agnostic since the adopted community detection is independent of specific learning tasks. Then the percentage of OOD graphs with substructures that never appeared in ID graphs per dataset is reported in Table 1. Observe that such percentage values are high, more than 44% in 4 out of 6 datasets. The results validate that task-agnostic substructures can reveal differences between ID and OOD graphs. As illustrated in Figure 1, if a method can preserve the task-agnostic substructures of ID graphs into embeddings, OOD graphs with unseen substructures will have embeddings distant from those of ID graphs, making them easy to detect.

Therefore, we design a series of techniques in SGOOD to effectively encode task-agnostic substructures and generate substructure-enhanced graph representations for graph-level OOD detection. Specifically, we first build a super graph  $\mathcal{G}_i$  of substructures for every graph  $G_i$  to obtain task-agnostic substructures and their relationships. Then, a two-level graph encoding pipeline is designed to work on  $G_i$  and  $\mathcal{G}_i$  in sequence to learn expressive substructure-enhanced graph representations. We prove that SGOOD with the pipeline is strictly more expressive than 1&2-WL, which theoretically justifies the power of preserving substructure patterns for

OOD detection. Moreover, to capture more information about task-agnostic substructures in training ID graphs, we design substructure-preserving graph augmentation techniques, which utilize the super graph of substructures to ensure that the substructures in a graph are modified as a whole. At test time, given a graph  $G_i$  and its super graph  $\mathcal{G}_i$ , our OOD detector obtains the graph-level representations of both, which are then used for OOD score estimation. Extensive experiments compare SGOOD with 11 baselines across 8 real-world graph datasets with various OOD types. SGOOD consistently outperforms existing methods, for example, achieving a 9.58% absolute improvement in AUROC over the runner-up baseline on the IMDB-M dataset. In summary, our contributions are:

- We present SGOOD, a leading method that highlights the importance of task-agnostic substructures and effectively leverages them to enhance graph-level OOD detection.
- We design a two-level graph encoding pipeline by leveraging a super graph of substructures, empowering SGOOD to learn graph representations enhanced with substructures.
- We further develop substructure-preserving graph augmentations via super graphs of substructures, to strengthen SGOOD’s ability in distinguishing OOD graphs.
- Extensive experiments demonstrate the superiority of SGOOD for graph-level OOD detection, achieving significant improvements over existing methods across multiple datasets.

## 2 PRELIMINARIES

We consider graph-level classification, which aims to classify a collection of graphs into different classes. Let  $G_i = (V_i, E_i)$  be a graph, where  $V_i$  and  $E_i$  are node set and edge set, respectively. Let  $\mathbf{x}_u \in \mathbb{R}^c$  denote the attribute vector of node  $u \in V_i$  in graph  $G_i$ . Denote  $\mathcal{X}$  as the in-distribution (ID) graph space and let  $\mathcal{Y} = \{1, 2, \dots, C\}$  be the label space. In graph-level classification, the training set  $D_{tr}^{in} = \{(G_i, y_i)\}_{i=1}^n$  is drawn i.i.d. from the joint data distribution  $P_{\mathcal{X}\mathcal{Y}}$ . Every graph sample in  $D_{tr}^{in}$  contains a graph  $G_i$  with label  $y_i$ . Let  $f$  be a learning model trained by the training set  $D_{tr}^{in}$ , and  $f$  is deployed to predict the label of a testing graph.

**Graph-level Out-Of-Distribution Detection.** At test time, graph-level OOD detection can be treated as a task to decide whether a testing graph  $G_i$  in testing set  $D_{test}$  is from the ID  $P_{\mathcal{X}}$  or from other irrelevant distributions (*i.e.*, OOD). A typical way for OOD detection is to develop an OOD detector by leveraging the representations generated from the classification model  $f$  that is trained via ID training graphs in  $D_{tr}^{in}$ . Specifically, the OOD detector has a scoring function  $S(G_i)$  for every testing graph  $G_i \in D_{test}$ . Testing graphs with low scores  $S(G_i)$  are regarded as ID, while the graphs with high scores are OOD. As stated in [26], a score threshold  $\lambda$  is typically set so that a high fraction of ID data (*e.g.*, 95%) is correctly classified.

## 3 THE SGOOD METHOD

**Solution Overview.** The main goal of SGOOD is to effectively encode task-agnostic substructures and their relationships into representations for graph-level OOD detection. As illustrated in Figure 2, SGOOD generates substructure-enhanced graph representations and further improves representation quality by substructure-preserving graph augmentations. Given a graph  $G_i$ , we first build

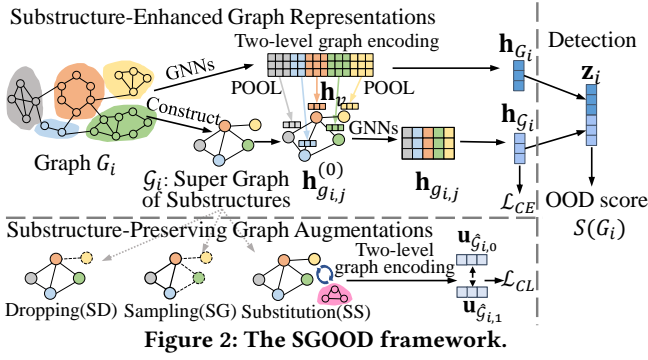


Figure 2: The SGOOD framework.

its super graph  $\mathcal{G}_i$  of task-agnostic substructures, in which a super node represents a substructure in  $G_i$  and edges connect super nodes by following the substructure connectivity in graph  $G_i$ . A two-level graph encoding pipeline is designed over both  $G_i$  and  $\mathcal{G}_i$  for learning graph-level representations that are enhanced by substructures. Furthermore, substructure-preserving graph augmentations are designed to preserve more information on task-agnostic substructures, making OOD graphs with unseen substructure patterns easier to detect. Specifically, given a graph  $G_i$ , we augment it by first performing dropping, sampling, and substitution on its super graph  $\mathcal{G}_i$  and then mapping the changes to  $G_i$  accordingly. In this way, the substructures in  $G_i$  are modified as a whole. SGOOD is trained using a combination of classification loss  $\mathcal{L}_{CE}$  and contrastive loss  $\mathcal{L}_{CL}$ . In test time, given a test graph  $G_i$ , we first obtain graph-level representations of both  $G_i$  and its super graph  $\mathcal{G}_i$ , concatenate and normalize the representations to compute the OOD score  $S(G_i)$ .

### 3.1 Substructure-Enhanced Graph Encoding

Given a graph  $G_i$ , we first describe how to get its super graph  $\mathcal{G}_i$  of task-agnostic substructures, and then present a two-level graph encoding pipeline to generate substructure-enhanced graph representations.

As SGOOD utilizes task-agnostic substructures, we treat substructure detection as a pre-processing step, and it is not our focus on how to detect substructures. There exist off-the-shelf methods [4, 6] to detect task-agnostic substructures. By default, we use modularity-based community detection [3]. We also test other subgraph detection methods and find that the modularity-based substructures are effective in SGOOD, as validated in Table 6.

**Super Graph of Substructures.** Let a substructure  $g_{i,j}$  of graph  $G_i$  be a connected subgraph of  $G_i$ . Specifically, a subgraph  $g_{i,j} = (V_{i,j}, E_{i,j})$  is a substructure of  $G_i = (V_i, E_i)$  iff  $V_{i,j} \subseteq V_i, E_{i,j} \subseteq E_i$ , and  $g_{i,j}$  is connected. The substructures  $\{g_{i,j}\}_{j=1}^{n_i}$  of a graph  $G_i$  satisfy the following properties: (i) the node sets of substructures are non-overlapping, (ii) the union of the nodes in all substructures is the node set of  $G_i$ , and (iii) every substructure is a connected subgraph of  $G_i$ . Then we construct the super graph  $\mathcal{G}_i$  by regarding every substructure  $g_{i,j}$  as a super node in  $\mathcal{G}_i$ , and connect super nodes by inserting edges via Definition 3.1. Super graph  $\mathcal{G}_i$  is a higher-order view depicting the relationships between the substructures of a graph  $G$ . We also add self-loops in super graph  $\mathcal{G}_i$ .

**Definition 3.1** (A Super Graph of Substructures). A super graph of substructures constructed from the input graph  $G_i = (V_i, E_i)$  is

denoted as  $\mathcal{G}_i = (\mathcal{V}_i, \mathcal{E}_i)$ , where every super node  $g_{i,j}$  in node set  $\mathcal{V}_i = \{g_{i,j}\}_{j=1}^{n_i}$  represents a substructure of  $G_i$ , and every edge in  $\mathcal{E}_i$  connecting two super nodes, and the edge set  $\mathcal{E}_i = \{(g_{i,j}, g_{i,k}) | \exists u \in V_{i,j} \wedge v \in V_{i,k}, (u, v) \in E_i\}$ .

**Two-level Graph Encoding.** Given a graph  $G_i$  and its super graph  $\mathcal{G}_i$ , we present a two-level graph encoding pipeline, as shown in Figure 2. The idea is that, in addition to learning representations over  $G_i$ , we further utilize the super graph  $\mathcal{G}_i$  to encode substructure information into graph representations, to better preserve distinguishable substructure patterns for effective OOD detection. The two-level graph encoding first adopts GNNs to learn node representations with initial features over graph  $G_i$ . For every node  $v \in V_i$ , its representation  $\mathbf{h}_v^{(l+1)}$  at  $(l+1)$ -layer is obtained by Eq. (1). Different GNNs have different aggregation and combination functions  $f_{AGGR}$ ,  $f_{COMB}$ . By default, we adopt Graph Isomorphism Network (GIN) [36] as the backbone. The GIN for  $G_i$  has  $L_1$  layers.

$$\mathbf{h}_v^{(l+1)} = f_{COMB}^{(l+1)}(\mathbf{h}_v^{(l)}, f_{AGGR}^{(l+1)}(\mathbf{h}_u^{(l)}, u \in N_{G_i}(v))), \quad (1)$$

where  $\mathbf{h}_v^{(l)} \in \mathbb{R}^d$  is the intermediate representation of node  $v$  from the  $l$ -th layer GNNs with hidden dimension  $d$ ,  $f_{AGGR}^{(l+1)}$  is the function that aggregates node features from  $v$ 's neighborhood  $N_{G_i}(v)$  in graph  $G_i$ ,  $f_{COMB}^{(l+1)}$  is the function that updates node  $v$ 's representation by combining the representations of its neighbors with its own, and initially  $\mathbf{h}_v^{(0)} = \mathbf{x}_v$ .

Next, we obtain the representations of substructures  $g_{i,j}$  in  $G_i$  by leveraging the node representations above. As shown in Eq. (2), given a node  $v$ , we first concatenate all representations  $\mathbf{h}_v^{(l)}$  for  $l = 1, \dots, L_1$  using  $f_{CAT}$  to get  $\mathbf{h}_v$  that preserves multi-scale semantics. Then, for a substructure  $g_{i,j}$  of graph  $G_i$ , we obtain the substructure representation  $\mathbf{h}_{g_{i,j}}^{(0)}$  by integrating  $\mathbf{h}_v$  of all  $v$  in  $g_{i,j}$  via DeepSet pooling [43]  $f_{POOL}$  in Eq. (2).

$$\mathbf{h}_{g_{i,j}}^{(0)} = f_{POOL}(\{\mathbf{h}_v | v \in V_{i,j}\}), \mathbf{h}_v = f_{CAT}(\{\mathbf{h}_v^{(l)}\}_{l=1}^{L_1}) \quad (2)$$

Note that  $\mathbf{h}_{g_{i,j}}^{(0)}$  only considers the nodes inside substructure  $g_{i,j}$  and the original graph topology  $G_i$ . To further consider the relationships depicted in the super graph  $\mathcal{G}_i$  of substructures, we regard  $\mathbf{h}_{g_{i,j}}^{(0)}$  as the initial features of super node  $g_{i,j}$  in  $\mathcal{G}_i$ , and employ a  $L_2$ -layer GIN over  $\mathcal{G}_i$  to learn substructure-enhanced graph representations by Eq. (3) and (4).

$$\mathbf{h}_{g_{i,j}}^{(l+1)} = f_{COMB}^{(l+1)}(\mathbf{h}_{g_{i,j}}^{(l)}, f_{AGGR}^{(l+1)}(\mathbf{h}_{g_{i,k}}, g_{i,k} \in N_{\mathcal{G}_i}(g_{i,j}))), \quad (3)$$

where  $N_{\mathcal{G}_i}(g_{i,j})$  is the neighbors of super node  $g_{i,j}$  in  $\mathcal{G}_i$ .

Lastly, in Eq. (4), we get the final representation  $\mathbf{h}_{g_{i,j}}$  of every super node  $g_{i,j}$  by concatenating the representation of  $g_{i,j}$  in every layer of the  $L_2$ -layer GIN, and obtain the graph representation  $\mathbf{h}_{\mathcal{G}_i}$  by readout  $f_{OUT}$  that is sum pooling.

$$\mathbf{h}_{\mathcal{G}_i} = f_{OUT}(\{\mathbf{h}_{g_{i,j}} | g_{i,j} \in \mathcal{V}_i\}), \mathbf{h}_{g_{i,j}} = f_{CAT}(\{\mathbf{h}_{g_{i,j}}^{(l)}\}_{l=0}^{L_2}) \quad (4)$$

Remark that the representation  $\mathbf{h}_{\mathcal{G}_i}$  of super graph  $\mathcal{G}_i$  of graph  $G_i$  is used to train the loss  $\mathcal{L}_{CE}$  for classification. Meanwhile, as explained shortly, for OOD detection during testing, we further consider another representation of graph  $G_i$  obtained by aggregating node representations. Existing studies, such as hierarchical pooling

[9, 18, 39] and subgraph GNNs [42, 46], learn task-specific substructures for the graph classification task. On the other hand, we leverage task-agnostic substructures. We conduct experiments to demonstrate that our SGOOD is more effective than these methods for graph-level OOD detection.

### 3.2 Substructure-Preserving Augmentations

Intuitively, if more information about task-agnostic substructures in training ID graphs is preserved, it is easier to distinguish OOD graphs with unseen substructure patterns. To achieve this, we design substructure-preserving graph augmentations by leveraging the super graph  $\mathcal{G}_i$  of graph  $G_i$ , to improve the performance further. However, it is challenging to achieve this. Substructures with subtle differences have different semantics. It is important to keep the substructures of a graph intact while performing augmentations. Common augmentation techniques like edge permutation and node dropping directly on graphs  $G_i$  may unexpectedly destroy meaningful substructures, and hamper OOD detection effectiveness.

To tackle the issue, we first perform augmentations on the super graph  $\mathcal{G}_i$  by regarding substructures as atomic nodes, and then map the augmentations to the original graph  $G_i$  with modifications over substructures as a whole. Specifically, we propose three substructure-level graph augmentations below, namely *substructure dropping (SD)*, *super graph sampling (SG)*, and *substructure substitution (SS)*. The default augmentation ratio is set to 0.3.

- *Substructure Dropping (SD)*. Given a graph  $G_i$  with its super graph  $\mathcal{G}_i$ , a fraction of super nodes in  $\mathcal{G}_i$  (i.e., the corresponding substructures in  $G_i$ ) are discarded randomly. Remark that selected substructures are dropped as a whole.
- *Super Graph Sampling (SG)*. In the super graph  $\mathcal{G}_i$ , we start from a random node, sample a fixed-size subgraph in  $\mathcal{G}_i$ , and drop the rest nodes and edges. The changes are mapped to  $G_i$  accordingly. Depth-first search is chosen as the sampling strategy [40].
- *Substructure Substitution (SS)*. Given a graph  $G_i$  in class  $c$  with super graph  $\mathcal{G}_i$  of substructures, we randomly substitute a fraction of nodes in  $\mathcal{G}_i$  (i.e., substructures in  $G_i$ ) with other substructures from the graphs of the same class  $c$ . To avoid drastic semantic change of the whole graph, only super nodes with degree one (excluding self-loops) in  $\mathcal{G}_i$  take part in the substitution.

### 3.3 Model Training and OOD Scoring

**3.3.1 Two-stage model training.** We adopt a cross-entropy loss  $\mathcal{L}_{CE}$  for classification. After getting representation  $\mathbf{h}_{\mathcal{G}_i}$  for the super graph  $\mathcal{G}_i$  of graph  $G_i$ , we apply a linear transformation to get prediction logits  $\hat{y}_i$ , evaluated against the ground-truth class label  $y_i$  to get  $\mathcal{L}_{CE}$  by Eq. (5) for a mini-batch of  $B$  training graphs.

$$\mathcal{L}_{CE} = -\frac{1}{B} \sum_{i=1}^B \sum_{c=1}^C \mathbb{1}(y_i = c) \log(\hat{y}_{i,c}) \quad (5)$$

Then we adopt the substructure-preserving graph augmentations in Section 3.2 to get contrastive loss  $\mathcal{L}_{CL}$ . Specifically, given a mini-batch of  $B$  training graphs  $\{G_i\}_{i=1}^B$  and their super graphs  $\{\mathcal{G}_i\}_{i=1}^B$ , we transform the super graphs to get  $\widehat{\mathcal{G}}_{i,0} = \mathcal{T}_0(\mathcal{G}_i)$  and  $\widehat{\mathcal{G}}_{i,1} = \mathcal{T}_1(\mathcal{G}_i)$ , where  $\mathcal{T}_0$  and  $\mathcal{T}_1$  are two augmentations chosen among  $\mathcal{A} = \{I, SD, SG, SS\}$ , where  $I$  indicates no augmentation. Graph  $G_i$  is transformed accordingly via  $\mathcal{T}_0$  and  $\mathcal{T}_1$  to obtain  $\widehat{G}_{i,0}$

and  $\widehat{G}_{i,1}$  respectively. Then, the representations  $\mathbf{h}_{\widehat{\mathcal{G}}_{i,0}}$  and  $\mathbf{h}_{\widehat{\mathcal{G}}_{i,1}}$  of the two augmented super graphs can be calculated by applying Eq.(1)-(4). We transform  $\mathbf{h}_{\widehat{\mathcal{G}}_{i,0}}$  and  $\mathbf{h}_{\widehat{\mathcal{G}}_{i,1}}$  by a shared projection head  $\psi(\cdot)$ , which is a 2-layer MLP, followed by  $l_2$ -normalization, to obtain  $\mathbf{u}_{\widehat{\mathcal{G}}_{i,0}} = \psi(\mathbf{h}_{\widehat{\mathcal{G}}_{i,0}}) / \|\psi(\mathbf{h}_{\widehat{\mathcal{G}}_{i,0}})\|$  and  $\mathbf{u}_{\widehat{\mathcal{G}}_{i,1}} = \psi(\mathbf{h}_{\widehat{\mathcal{G}}_{i,1}}) / \|\psi(\mathbf{h}_{\widehat{\mathcal{G}}_{i,1}})\|$ , respectively. We get  $\mathcal{L}_{CL}$  by

$$\mathcal{L}_{CL} = \frac{1}{2B} \sum_{i=1}^B \sum_{a \in \{0,1\}} -\log \frac{\exp(\mathbf{u}_{\widehat{\mathcal{G}}_{i,a}}^\top \mathbf{u}_{\widehat{\mathcal{G}}_{i,1-a}} / \tau)}{\sum_{j=1}^B \mathbb{1}(j \neq i) \sum_{k \in \{0,1\}} \exp(\mathbf{u}_{\widehat{\mathcal{G}}_{i,a}}^\top \mathbf{u}_{\widehat{\mathcal{G}}_{j,1-k}} / \tau)}, \quad (6)$$

where  $\tau$  is a temperature parameter.

The overall training loss is the combination of  $\mathcal{L}_{CE}$  and  $\mathcal{L}_{CL}$ , weighted by  $\alpha$ :

$$\mathcal{L} = \mathcal{L}_{CE} + \alpha \mathcal{L}_{CL}. \quad (7)$$

The training procedure of SGOOD consists of two stages. In the first pre-training stage, the parameters are solely updated by minimizing  $\mathcal{L}_{CL}$  for  $T_{PT}$  epochs. In the second stage, the parameters are fine-tuned under the combined overall loss  $\mathcal{L}$  for  $T_{PT}$  epochs. This training procedure achieves better performance than directly training  $\mathcal{L}$ , as shown in Figure 4 when pretraining  $T_{PT}$  is 0.

**3.3.2 Graph-level OOD scoring.** Recall that the main goal of SGOOD is to let the representations of ID data and OOD data to be distant from each other. Given a testing graph  $G_i \in D_{test}$ , we use the standard Mahalanobis distance [19] to quantify its OOD score. If  $G_i$  is with large Mahalanobis distance from the ID training data, it tends to be OOD.

As shown in Figure 2, in addition to the representation  $\mathbf{h}_{\mathcal{G}_i}$  of the super graph  $\mathcal{G}_i$  of a testing graph  $G_i$ , SGOOD also aggregates the node representations of  $G_i$  to get  $\mathbf{h}_{G_i} = f_{OUT}(\{\mathbf{h}_v | v \in V_i\})$ . Representations  $\mathbf{h}_{\mathcal{G}_i}$  and  $\mathbf{h}_{G_i}$  are concatenated together to estimate the OOD score  $S(G_i)$ :

$$S(G_i) = \max_{c \in [C]} (\mathbf{z}_i - \boldsymbol{\mu}_c)^\top \widehat{\Sigma}^{-1} (\mathbf{z}_i - \boldsymbol{\mu}_c), \mathbf{z}_i = \frac{f_{CAT}(\mathbf{h}_{\mathcal{G}_i}, \mathbf{h}_{G_i})}{\|f_{CAT}(\mathbf{h}_{\mathcal{G}_i}, \mathbf{h}_{G_i})\|_2}, \quad (8)$$

where  $[C] = \{1, \dots, C\}$ ,  $\boldsymbol{\mu}_c$  and  $\widehat{\Sigma}$  are the class centroid for class  $c$  and covariance matrix of training ID graphs, respectively.

### 3.4 Theoretical Analysis

We analyze the expressive power of SGOOD (using message-passing GNNs as the backbone) in comparison to 1&2-WL, a key tool for evaluating the expressivity of GNNs [36]. In Proposition 3.2, we demonstrate that SGOOD is more expressive than 1&2-WL, enabling it to distinguish structural patterns beyond the capability of 1&2-WL, and consequently, message-passing GNNs. The analysis, together with our empirical findings in Section 1, explains the power of SGOOD for graph-level OOD detection.

**Proposition 3.2.** *When the GNNs adopted in SGOOD are with sufficient number of layers, and the  $f_{POOL}$  function in Eq.(2) and  $f_{OUT}$  function in Eq.(4) are injective, then SGOOD is strictly more expressive than 1&2-WL.*

**PROOF.** We first prove that SGOOD is at least as powerful as 1&2-WL in Lemma 3.3. Then, we prove that SGOOD can distinguish 2-regular graphs that 1&2-WL cannot distinguish in Lemma 3.5.

Combining these two Lemmas, we prove that SGOOD is strictly more expressive than 1&2-WL.  $\square$

**Lemma 3.3.** *For graphs  $G_1 = (V_1, E_1)$  and  $G_2 = (V_2, E_2)$  identified as non-isomorphic by 1&2-WL, SGOOD projects them into different representations  $\mathbf{h}_{\mathcal{G}_1}$  and  $\mathbf{h}_{\mathcal{G}_2}$  in Eq. (4).*

**PROOF.** Let  $\mathcal{H}_1^G = \{\mathbf{h}_v | v \in V_1\}$  and  $\mathcal{H}_2^G = \{\mathbf{h}_v | v \in V_2\}$  be the multisets of node representations of  $G_1$  and  $G_2$  generated by GIN in Eq. (2), respectively. Let  $\mathcal{G}_1 = (\mathcal{V}_1, \mathcal{E}_1)$  and  $\mathcal{G}_2 = (\mathcal{V}_2, \mathcal{E}_2)$  be the super graphs of  $G_1$  and  $G_2$  respectively. We consider two cases: (1)  $|\mathcal{V}_1| \neq |\mathcal{V}_2|$ , (2)  $|\mathcal{V}_1| = |\mathcal{V}_2|$ .

For case (1),  $\mathcal{G}_1$  and  $\mathcal{G}_2$  are two graphs with different number of nodes. Thus,  $\mathcal{G}_1$  and  $\mathcal{G}_2$  can be easily determined as non-isomorphic by 1&2-WL. It is proved that GIN with sufficient number of layers and all injective functions is as powerful as 1&2-WL [36]. As GIN is adopted in SGOOD as GNN backbone with sufficient number of layers and  $f_{\text{OUT}}$  function in Eq.(4) is injective, representations  $\mathbf{h}_{\mathcal{G}_1}$  and  $\mathbf{h}_{\mathcal{G}_2}$  generated by SGOOD are different.

For case (2), let  $|\mathcal{V}_1| = |\mathcal{V}_2| = K$ ,  $\mathcal{H}_1^G = \{\mathbf{h}_{g_{1,j}}^{(0)} | g_{1,j} \in \mathcal{V}_1\}$  and  $\mathcal{H}_2^G = \{\mathbf{h}_{g_{2,j}}^{(0)} | g_{2,j} \in \mathcal{V}_2\}$  be the multisets of initial node representations of  $\mathcal{G}_1$  and  $\mathcal{G}_2$  calculated by Eq.(2), respectively. Using GIN with sufficient number of layers, we get  $\mathcal{H}_1^G \neq \mathcal{H}_2^G$  [36]. As stated in Section 3.1, the substructures  $\{g_{i,j}\}_{j=1}^{n_i}$  of a graph  $G_i$  satisfy the following properties: (i) the substructures are non-overlapping, (ii) the union of nodes in all substructures is the node set of  $G_i$ . Thus,  $\{\{\mathbf{h}_v | v \in g_{1,j}\}\}_{j=1}^K$  (resp.  $\{\{\mathbf{h}_v | v \in g_{2,j}\}\}_{j=1}^K$ ) is a partition of  $\mathcal{H}_1^G$  (resp.  $\mathcal{H}_2^G$ ). Then, we have  $\{\{\mathbf{h}_v | v \in g_{1,j}\}\}_{j=1}^K \neq \{\{\mathbf{h}_v | v \in g_{2,j}\}\}_{j=1}^K$ . As  $f_{\text{POOL}}$  function in Eq.(2) is injective, we have  $\{f_{\text{POOL}}(\{\mathbf{h}_v | v \in g_{1,j}\})\}_{j=1}^K \neq \{f_{\text{POOL}}(\{\mathbf{h}_v | v \in g_{2,j}\})\}_{j=1}^K$ , that is  $\mathcal{H}_1^{\mathcal{G}} \neq \mathcal{H}_2^{\mathcal{G}}$ . As GIN and  $f_{\text{OUT}}$  function in Eq.(4) are both injective, we derive that  $\mathbf{h}_{\mathcal{G}_1}$  and  $\mathbf{h}_{\mathcal{G}_2}$  generated on  $\mathcal{H}_1^{\mathcal{G}}$  and  $\mathcal{H}_2^{\mathcal{G}}$  are different.

Combining cases (1) and (2), we prove Lemma 3.3.  $\square$

Next, we prove that SGOOD can distinguish  $n$ -node 2-regular graphs that 1&2-WL cannot distinguish in Lemma 3.5. Before that, we first give the definition of 2-regular graphs. Note that we only consider undirected graphs in this paper.

**Definition 3.4** (2-regular graph). A graph is said to be regular of degree 2 if all local degrees are 2.

**Lemma 3.5.** *Given two non-isomorphic  $n$ -node 2-regular graphs  $G_1$  and  $G_2$  that 1&2-WL cannot distinguish, SGOOD projects them into different graph representations  $\mathbf{h}_{\mathcal{G}_1}$  and  $\mathbf{h}_{\mathcal{G}_2}$  in Eq. (4).*

**PROOF.** Based on the definition of a 2-regular graph, we can say that  $G_1$  and  $G_2$  consist of one or more disconnected cycles. Let  $r_1$  and  $r_2$  be the number of cycles in  $G_1$  and  $G_2$ . Let  $\mathcal{G}_1 = (\mathcal{V}_1, \mathcal{E}_1)$  and  $\mathcal{G}_2 = (\mathcal{V}_2, \mathcal{E}_2)$  be the constructed super graphs of  $G_1$  and  $G_2$ . We consider two cases: (1)  $r_1 \neq 1 \wedge r_2 \neq 1$ , (2)  $(r_1 = 1 \wedge r_2 \neq 1) \vee (r_1 \neq 1 \wedge r_2 = 1)$ .

For case (1),  $G_1$  and  $G_2$  consist of disconnected circles. As  $\mathcal{G}_1$  and  $\mathcal{G}_2$  are constructed by modularity-based community detection method [3], in Lemma 3.4 of [2], for graphs of disconnected circles, there is always a clustering with maximum modularity, in which each cluster consists of a connected subgraph. As a result,  $\forall g_{1,j} \in \mathcal{V}_1$  is a circle in  $G_1$ , and  $|\mathcal{V}_1| = r_1$ . Similarly,  $\forall g_{2,j} \in \mathcal{V}_2$  is a circle

**Table 2: Data Statistics.**

Dataset	OOD Type	# Class	# ID Train	# ID Val	# ID Test	# OOD Test
ENZYMES [27]	Unseen Classes	6	480	60	60	60
IMDB-M [27]	Unseen Classes	3	1200	150	150	150
IMDB-B [27]	Unseen Classes	2	800	100	100	100
REDDIT-12K [37]	Unseen Classes	11	6997	875	875	875
BACE [35]	Scaffold	2	968	121	121	121
BBBP [35]	Scaffold	2	1303	164	164	164
DrugOOD [15]	Protein Target	2	800	100	100	100
HIV [35]	Scaffold	2	26319	3291	3291	3291

in  $G_2$ , and  $|\mathcal{V}_2| = r_2$ . Let  $\mathcal{N}_1 = \{|\mathcal{V}_{1,j}|\}_{j=1}^{r_1}$  and  $\mathcal{N}_2 = \{|\mathcal{V}_{2,j}|\}_{j=1}^{r_2}$ . Since  $G_1$  and  $G_2$  are non-isomorphic, we have  $\exists n_{1,j} \in \mathcal{N}_1 : \forall n_{2,j} \in \mathcal{N}_2, n_{1,j} \neq n_{2,j}$ . As a result, we have  $\mathcal{N}_1 \neq \mathcal{N}_2$ . Then, we have  $\{\{\mathbf{h}_v | v \in \mathcal{V}_{1,j}\}\}_{j=1}^{r_1} \neq \{\{\mathbf{h}_v | v \in \mathcal{V}_{2,j}\}\}_{j=1}^{r_2}$ . As  $f_{\text{POOL}}$  function in Eq.(2) is injective, we have  $\mathcal{H}_1^{\mathcal{G}} = \{f_{\text{POOL}}(\{\mathbf{h}_v | v \in g_{1,j}\})\}_{j=1}^{r_1}$ ,  $\mathcal{H}_2^{\mathcal{G}} = \{f_{\text{POOL}}(\{\mathbf{h}_v | v \in g_{2,j}\})\}_{j=1}^{r_2}$ , and  $\mathcal{H}_1^{\mathcal{G}} \neq \mathcal{H}_2^{\mathcal{G}}$ . As shown in the proof of Lemma 3.3, we have  $\mathbf{h}_{\mathcal{G}_1}$  and  $\mathbf{h}_{\mathcal{G}_2}$  generated on  $\mathcal{H}_1^{\mathcal{G}}$  and  $\mathcal{H}_2^{\mathcal{G}}$  are different.

For case (2), we consider  $r_1 = 1 \wedge r_2 \neq 1$ , and the proof when  $r_2 = 1 \wedge r_1 \neq 1$  is similar.  $G_1$  consists of one single circle, and  $G_2$  consists of  $r_2$  disconnected circles. For  $G_2$  and  $\mathcal{G}_2$ ,  $\forall g_{2,j} \in \mathcal{V}_2$  is a circle in  $G_2$ , and  $|\mathcal{V}_2| = r_2$  following the conclusion in case (1). For  $G_1$  and  $\mathcal{G}_1$ , we consider two cases: (i)  $|\mathcal{V}_1| = r_1 = 1$ , and (ii)  $|\mathcal{V}_1| > 1$ . For case (i),  $\mathcal{V}_1 = \{g_{1,1}\}$ , where  $g_{1,1} = G_1$ . Let  $\mathcal{N}_1 = \{|\mathcal{V}_{1,j}|\}_{j=1}^{r_1} = \{|\mathcal{V}_{1,1}|\}$  and  $\mathcal{N}_2 = \{|\mathcal{V}_{2,j}|\}_{j=1}^{r_2}$ . As  $|\mathcal{N}_1| \neq |\mathcal{N}_2|$ , we have  $\mathcal{N}_1 \neq \mathcal{N}_2$ . Similar to case (1), we conclude that graph representations  $\mathbf{h}_{\mathcal{G}_1}$  and  $\mathbf{h}_{\mathcal{G}_2}$  generated on  $\mathcal{H}_1^{\mathcal{G}}$  and  $\mathcal{H}_2^{\mathcal{G}}$  are different. For case (ii),  $\mathcal{V}_1 = \{g_{1,j}\}_{j=1}^{r_1}$ , where  $\forall g_{1,j} \in \mathcal{V}_1$  is a chain and two nearby chain are connected in  $\mathcal{G}_1$ . In other words,  $\mathcal{G}_1$  is a  $|\mathcal{V}_1|$ -circle while  $\mathcal{G}_2$  consists of  $|\mathcal{V}_2|$  isolated nodes. Thus,  $\mathcal{G}_1$  and  $\mathcal{G}_2$  can be distinguished as non-isomorphic by 1&2-WL. By [36], when we encode  $\mathcal{G}_1$  and  $\mathcal{G}_2$  by Eq. (3) with sufficient layers of GIN, and generate  $\mathbf{h}_{\mathcal{G}_1}$  and  $\mathbf{h}_{\mathcal{G}_2}$  by Eq. (4), where  $f_{\text{OUT}}$  is injective,  $\mathbf{h}_{\mathcal{G}_1}$  and  $\mathbf{h}_{\mathcal{G}_2}$  are different. Combining cases (i) and (ii), we prove SGOOD generates different  $\mathbf{h}_{\mathcal{G}_1}$  and  $\mathbf{h}_{\mathcal{G}_2}$  for  $G_1$  and  $G_2$  in case (2).

Combining cases (1) and (2), we prove Lemma 3.5.  $\square$

## 4 EXPERIMENTS

We evaluate SGOOD in graph-level OOD detection against 11 baseline methods across 8 real-world datasets.

### 4.1 Experimental Setup

**4.1.1 Datasets.** We adopt real-world datasets that encompass diverse types of OOD graphs, as listed in Table 2. These datasets are curated from mainstream graph classification benchmarks [15, 27, 35] into OOD detection scenarios. The OOD graph data is generated following [21, 24]. All ID graphs are randomly split into training, validation, and testing with ratio 8:1:1, following the settings of standard graph classification [14, 27]. The testing set consists of the same number of ID graphs and OOD graphs.

**Table 3: Overall OOD detection performance by AUROC, AUPR, and FPR95 in percentage % (mean  $\pm$  std).  $\uparrow$  indicates larger values are better and vice versa. **Bold:** best. Underline: runner-up.**

Method	ENZYMES			IMDB-M			IMDB-B			REDDIT-12K		
	AUROC $\uparrow$	AUPR $\uparrow$	FPR95 $\downarrow$	AUROC $\uparrow$	AUPR $\uparrow$	FPR95 $\downarrow$	AUROC $\uparrow$	AUPR $\uparrow$	FPR95 $\downarrow$	AUROC $\uparrow$	AUPR $\uparrow$	FPR95 $\downarrow$
MSP	61.34 $\pm$ 3.79	61.65 $\pm$ 6.64	89.67 $\pm$ 2.26	42.75 $\pm$ 1.52	51.04 $\pm$ 1.93	95.73 $\pm$ 1.63	58.13 $\pm$ 2.31	59.63 $\pm$ 1.22	91.40 $\pm$ 4.16	50.63 $\pm$ 0.87	48.60 $\pm$ 1.08	95.95 $\pm$ 1.25
Energy	54.69 $\pm$ 9.18	56.90 $\pm$ 8.85	89.33 $\pm$ 3.55	24.50 $\pm$ 19.73	37.26 $\pm$ 11.78	96.40 $\pm$ 2.25	49.58 $\pm$ 17.76	59.03 $\pm$ 13.06	92.80 $\pm$ 3.55	55.10 $\pm$ 0.48	56.52 $\pm$ 0.78	97.19 $\pm$ 0.58
ODIN	63.70 $\pm$ 2.70	65.72 $\pm$ 4.77	92.66 $\pm$ 3.26	40.12 $\pm$ 2.96	50.08 $\pm$ 2.44	96.66 $\pm$ 1.03	58.25 $\pm$ 2.94	61.36 $\pm$ 0.49	92.20 $\pm$ 2.92	51.74 $\pm$ 2.03	54.53 $\pm$ 1.26	96.45 $\pm$ 0.73
MAH	67.37 $\pm$ 3.67	63.81 $\pm$ 2.15	83.33 $\pm$ 9.60	<u>69.26<math>\pm</math>3.67</u>	63.64 $\pm$ 2.14	60.93 $\pm$ 9.06	76.77 $\pm$ 4.37	76.88 $\pm$ 6.30	<u>81.40<math>\pm</math>7.14</u>	<u>72.68<math>\pm</math>0.87</u>	<u>74.47<math>\pm</math>0.48</u>	<u>80.75<math>\pm</math>2.05</u>
GNNSafe	56.85 $\pm$ 8.91	56.13 $\pm$ 8.26	97.00 $\pm$ 3.71	21.93 $\pm$ 1.76	36.88 $\pm$ 1.68	95.46 $\pm$ 1.42	70.49 $\pm$ 14.80	75.67 $\pm$ 15.71	87.80 $\pm$ 5.81	51.68 $\pm$ 0.08	53.97 $\pm$ 0.52	95.59 $\pm$ 2.80
GraphDE	61.35 $\pm$ 3.99	66.26 $\pm$ 2.98	99.00 $\pm$ 0.81	66.87 $\pm$ 4.25	62.60 $\pm$ 4.47	93.06 $\pm$ 8.24	26.91 $\pm$ 3.35	42.73 $\pm$ 2.06	100.00 $\pm$ 0.00	59.40 $\pm$ 0.18	63.06 $\pm$ 0.30	81.82 $\pm$ 0.01
GOOD-D	67.21 $\pm$ 6.41	64.86 $\pm$ 6.32	82.33 $\pm$ 8.31	61.89 $\pm$ 4.87	<u>66.91<math>\pm</math>7.60</u>	95.20 $\pm$ 4.55	52.58 $\pm$ 10.21	55.69 $\pm$ 10.56	99.20 $\pm$ 1.00	56.11 $\pm$ 0.10	59.56 $\pm$ 0.16	93.67 $\pm$ 0.34
AAGOD	69.25 $\pm$ 4.65	65.02 $\pm$ 4.41	82.78 $\pm$ 2.83	70.76 $\pm$ 5.48	68.15 $\pm$ 4.45	81.56 $\pm$ 22.32	72.51 $\pm$ 1.11	67.86 $\pm$ 4.79	86.33 $\pm$ 4.03	60.25 $\pm$ 2.16	61.44 $\pm$ 1.61	92.53 $\pm$ 1.55
OCGIN	68.11 $\pm$ 4.61	68.90 $\pm$ 4.19	89.67 $\pm$ 3.70	47.51 $\pm$ 9.47	50.76 $\pm$ 4.53	98.27 $\pm$ 17.70	60.78 $\pm$ 5.21	57.80 $\pm$ 5.10	8780 $\pm$ 9.15	59.33 $\pm$ 1.26	60.02 $\pm$ 1.88	90.00 $\pm$ 2.01
GLocalKD	71.46 $\pm$ 3.21	64.93 $\pm$ 4.44	<u>78.67<math>\pm</math>6.37</u>	19.82 $\pm$ 1.57	35.39 $\pm$ 0.49	98.27 $\pm$ 1.13	<u>79.39<math>\pm</math>4.71</u>	<b>85.56<math>\pm</math>3.33</b>	87.40 $\pm$ 5.42	49.60 $\pm$ 1.06	51.75 $\pm$ 0.72	97.60 $\pm$ 0.35
OGGTL	<u>73.22<math>\pm</math>1.83</u>	<b>73.61<math>\pm</math>3.19</b>	82.33 $\pm$ 2.70	54.07 $\pm$ 12.93	58.20 $\pm$ 7.86	86.40 $\pm$ 6.49	37.39 $\pm$ 18.82	47.11 $\pm$ 14.06	98.80 $\pm$ 2.40	51.62 $\pm$ 0.019	53.33 $\pm$ 0.01	96.79 $\pm$ 0.06
SGOOD	<b>74.40<math>\pm</math>1.42</b>	<u>72.53<math>\pm</math>2.51</u>	<b>73.66<math>\pm</math>7.03</b>	<b>78.84<math>\pm</math>2.00</b>	<b>72.54<math>\pm</math>3.21</b>	<b>45.46<math>\pm</math>6.62</b>	<b>80.41<math>\pm</math>3.16</b>	<u>83.49<math>\pm</math>3.59</u>	<b>81.20<math>\pm</math>2.28</b>	<b>74.95<math>\pm</math>0.79</b>	<b>74.93<math>\pm</math>0.93</b>	<b>75.17<math>\pm</math>2.72</b>
Method	BACE			BBBP			DrugOOD			HIV		
	AUROC $\uparrow$	AUPR $\uparrow$	FPR95 $\downarrow$	AUROC $\uparrow$	AUPR $\uparrow$	FPR95 $\downarrow$	AUROC $\uparrow$	AUPR $\uparrow$	FPR95 $\downarrow$	AUROC $\uparrow$	AUPR $\uparrow$	FPR95 $\downarrow$
MSP	46.34 $\pm$ 6.10	48.65 $\pm$ 3.08	97.02 $\pm$ 2.18	57.37 $\pm$ 4.28	56.84 $\pm$ 3.36	94.63 $\pm$ 2.26	52.86 $\pm$ 5.26	54.49 $\pm$ 4.33	98.80 $\pm$ 0.01	50.75 $\pm$ 1.88	50.49 $\pm$ 0.91	95.52 $\pm$ 0.50
Energy	46.05 $\pm$ 6.66	49.68 $\pm$ 4.16	97.36 $\pm$ 2.92	56.56 $\pm$ 4.16	55.74 $\pm$ 2.78	92.68 $\pm$ 2.62	52.81 $\pm$ 5.36	54.98 $\pm$ 4.36	98.20 $\pm$ 1.16	50.97 $\pm$ 2.13	50.49 $\pm$ 0.91	95.50 $\pm$ 0.59
ODIN	45.51 $\pm$ 3.85	48.28 $\pm$ 3.76	97.02 $\pm$ 1.53	54.78 $\pm$ 3.46	54.63 $\pm$ 3.69	96.34 $\pm$ 1.80	51.09 $\pm$ 3.79	52.70 $\pm$ 2.66	99.00 $\pm$ 1.09	50.16 $\pm$ 0.73	49.95 $\pm$ 0.58	94.60 $\pm$ 1.07
MAH	73.78 $\pm$ 1.97	75.33 $\pm$ 2.32	86.78 $\pm$ 6.32	53.77 $\pm$ 4.27	52.57 $\pm$ 3.81	93.29 $\pm$ 2.51	66.90 $\pm$ 4.14	64.30 $\pm$ 4.43	81.60 $\pm$ 4.58	58.10 $\pm$ 3.60	57.18 $\pm$ 3.18	91.89 $\pm$ 1.32
GNNSafe	47.61 $\pm$ 7.50	51.52 $\pm$ 5.91	98.18 $\pm$ 2.05	47.04 $\pm$ 2.40	51.52 $\pm$ 5.90	98.41 $\pm$ 0.99	50.44 $\pm$ 0.57	51.14 $\pm$ 0.30	96.01 $\pm$ 0.33	50.98 $\pm$ 6.82	55.13 $\pm$ 6.81	96.01 $\pm$ 0.33
GraphDE	47.32 $\pm$ 1.52	51.1 $\pm$ 2.57	94.21 $\pm$ 4.58	50.88 $\pm$ 2.78	51.47 $\pm$ 3.84	94.63 $\pm$ 2.34	60.19 $\pm$ 4.32	62.59 $\pm$ 2.47	88.80 $\pm$ 5.60	52.38 $\pm$ 1.86	54.14 $\pm$ 3.21	94.89 $\pm$ 0.84
GOOD-D	70.42 $\pm$ 2.22	73.21 $\pm$ 3.34	88.26 $\pm$ 1.78	54.15 $\pm$ 1.10	58.58 $\pm$ 1.93	99.39 $\pm$ 0.41	60.52 $\pm$ 3.33	63.09 $\pm$ 2.54	98.40 $\pm$ 1.27	59.69 $\pm$ 0.62	57.10 $\pm$ 1.14	92.03 $\pm$ 0.61
AAGOD	71.41 $\pm$ 2.37	71.82 $\pm$ 1.72	90.63 $\pm$ 1.95	58.16 $\pm$ 1.54	59.35 $\pm$ 1.32	93.5 $\pm$ 0.76	60.29 $\pm$ 3.23	66.2 $\pm$ 3.36	95.33 $\pm$ 0.47	55.72 $\pm$ 0.69	54.29 $\pm$ 0.51	92.2 $\pm$ 0.3
OCGIN	59.71 $\pm$ 5.20	61.43 $\pm$ 5.18	93.39 $\pm$ 4.44	47.78 $\pm$ 5.72	47.27 $\pm$ 2.98	94.76 $\pm$ 2.70	57.95 $\pm$ 5.80	59.50 $\pm$ 7.00	94.20 $\pm$ 3.12	54.06 $\pm$ 0.47	52.14 $\pm$ 0.26	92.81 $\pm$ 1.01
GLocalKD	45.34 $\pm$ 2.11	55.39 $\pm$ 2.35	98.68 $\pm$ 1.11	43.77 $\pm$ 2.23	45.84 $\pm$ 1.20	98.29 $\pm$ 1.00	45.72 $\pm$ 0.97	50.90 $\pm$ 3.33	100.00 $\pm$ 0.00	46.81 $\pm$ 2.90	46.95 $\pm$ 2.01	97.05 $\pm$ 0.19
OGGTL	<u>80.84<math>\pm</math>2.00</u>	<u>79.93<math>\pm</math>1.26</u>	<u>66.44<math>\pm</math>8.89</u>	58.73 $\pm$ 2.19	<b>60.47<math>\pm</math>1.38</b>	<u>91.46<math>\pm</math>2.21</u>	<u>67.59<math>\pm</math>7.93</u>	<u>70.90<math>\pm</math>5.80</u>	83.00 $\pm$ 11.22	51.78 $\pm$ 0.19	53.71 $\pm$ 2.02	96.41 $\pm$ 0.05
SGOOD	<b>84.39<math>\pm</math>2.73</b>	<b>83.32<math>\pm</math>2.49</b>	<b>64.13<math>\pm</math>4.83</b>	<b>61.25<math>\pm</math>1.60</b>	<u>59.36<math>\pm</math>2.39</u>	<b>88.04<math>\pm</math>3.44</b>	<b>73.15<math>\pm</math>4.48</b>	<b>73.25<math>\pm</math>4.49</b>	<b>67.40<math>\pm</math>5.16</b>	<b>60.82<math>\pm</math>0.75</b>	<b>59.99<math>\pm</math>0.69</b>	<b>90.39<math>\pm</math>1.04</b>

- **ENZYMES** comprises protein networks representing enzymes. ‘Non-enzymes’ protein networks from the PROTEINS dataset [27] are introduced as OOD.
- **IMDB-M** includes social networks categorized into three classes. We designate social networks from the IMDB-B dataset [27], labeled with ‘Action’, as OOD graphs, noting that the ‘Action’ class is absent in IMDB-M.
- **IMDB-B** regards graphs labeled as ‘Comedy’ and ‘Sci-Fi’ in IMDB-M as OOD graphs, as these classes are absent in IMDB-B.
- **REDDIT-12K** is a large-scale social network dataset [37]. Graphs in the dataset REDDIT-BINARY [37], which are social networks with classes different from REDDIT-12K, are introduced as OOD.
- **BACE** consists of molecules for property prediction. In our adaptation for OOD detection, we use the training set of the original BACE as ID graphs. OOD graphs are introduced by incorporating graphs from the provided test set, wherein molecules exhibit scaffolds distinct from those present in training set.
- **BBBP** is constructed similarly to BACE. OOD graphs have molecular scaffolds different from ID graphs.
- **DrugOOD** is a curated OOD dataset consisting of various molecular graphs. We use the provided curator to generate both ID and OOD graphs that have different protein targets.
- **HIV** is a large-scale molecular graph dataset [35]. Graphs with scaffolds different from ID graphs are regarded as OOD.
- **General OOD detection methods**, including MSP [13], Energy [23], ODIN [22], and MAH [19]. MSP, Energy, and ODIN estimate OOD scores directly from classification logits at test time. MSP uses the maximum softmax score as OOD score while Energy uses energy function. ODIN combines temperature scaling with gradient-based input perturbations to enlarge the differences between OOD and ID samples. MAH measures Mahalanobis distance between test samples and ID training data.
- **Graph-level OOD detection methods**, including GNNSafe [33], GraphDE [21], GOOD-D [24] and AAGOD [10]. GNNSafe incorporates GNNs in the energy model and detects OOD samples using energy scores. In our paper, we use graph labels to directly run the basic version of GNNSafe from its Section 3.1 [33]. GraphDE is a probabilistic model-based approach developed for debiased learning and OOD detection in graph data. GOOD-D is an unsupervised OOD detection method that adopts contrastive learning to capture latent patterns of ID graphs. AAGOD is a post-hoc framework that adopts an adaptive amplifier to enlarge the gap between OOD and ID graphs.
- **Graph-level anomaly detection methods**, including OCGIN [45], OCGTL [28], and GLocalKD [25]. OCGIN combines deep one-class classification with GIN [36] to detect outlier graphs at test time. OCGTL develops a one-class objective for graph anomaly detection. GLocalKD leverages knowledge distillation to detect both local and global graph anomalies.

4.1.2 *Baselines.* We compare with 11 competitors in 3 categories.



**Table 4: ID graph classification performance measured by average ID ACC (in percentage %). / indicates that ID ACC is not applicable for unsupervised methods.**

Method	ENZYMES	IMDB-M	IMDB-B	REDDIT-12K	BACE	BBBP	HIV	DrugOOD
MSP	37.33	48.27	69.80	48.91	80.83	87.44	96.62	79.20
Energy	37.33	48.27	69.80	48.91	80.83	87.44	96.62	79.20
ODIN	37.33	48.27	69.80	48.91	80.83	87.44	96.62	79.20
MAH	37.33	48.27	69.80	48.91	80.83	87.44	96.62	79.20
GNNSafe	17.66	30.13	50.20	27.42	56.69	79.14	96.58	64.40
GraphDE	46.00	37.86	69.80	40.68	77.68	88.90	96.20	77.00
GOOD-D	/	/	/	/	/	/	/	/
AAGOD	/	/	/	/	/	/	/	/
SGOOD	<b>48.66</b>	<b>48.66</b>	<b>71.60</b>	<b>51.82</b>	80.33	<b>89.14</b>	<b>96.66</b>	<b>79.40</b>

**4.1.3 Evaluation and Implementation.** Following [24, 33], all methods are trained using ID training set and evaluated their OOD detection performance and ID classification performance in the test set. Hyperparameters are tuned using ID graphs from the validation set. All methods are evaluated five times on each dataset, and the reported performance metrics are based on the mean and standard deviation results on the test set. We use three commonly used metrics AUROC, AUPR and FPR95 for OOD detection evaluation [13, 33]. For the classification performance in ID graphs, we use Accuracy (ID ACC). Remark that the priority of the graph-level OOD detection task is to accurately identify OOD graphs, instead of improving the ID ACC. For SGOOD, we set the number of layers  $L_1 = 3$  and  $L_2 = 2$ , and dimension  $d$  as 16. We set batch size 128. Training consists of 100 epochs for pre-training ( $T_{PT}$ ) and 500 epochs for fine-tuning ( $T_{FT}$ ). We experiment with learning rates in the range  $\{0.01, 0.001, 0.0001\}$  for the initial stage, and set the learning rate to 0.001 and  $\alpha$  to 0.1 for the refinement stage. For all baselines, we use the suggested parameters from their papers or obtained by grid search. Experiments are conducted on a Linux server equipped with an Nvidia RTX 3090 GPU card.

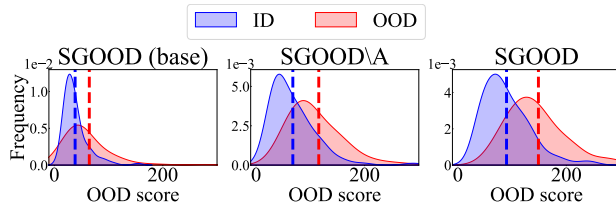
## 4.2 Overall Performance

**OOD detection performance.** Table 3 reports the overall graph-level OOD detection performance of all methods by AUROC, AUPR and FPR95 metrics on all datasets, by mean and standard deviation values. Observe that SGOOD consistently achieves superior OOD detection effectiveness under most settings. For instance, on IMDB-M, SGOOD has AUROC 78.84%, which indicates 9.58% absolute improvement over the best competitor with AUROC 69.26%. As another example on BACE molecule dataset, the AUROC of SGOOD is 84.39%, while the runner-up achieves AUROC 80.84%. The overall results in Table 3 show that SGOOD effectively encode task-agnostic substructures into expressive representations for graph-level OOD detection, validating the power of our technical designs.

**ID graph classification performance.** Table 4 reports the performance on ID graph classification of all methods by Accuracy (ID ACC). The results of graph-level anomaly detection methods are not reported as ID ACC is not applicable. SGOOD achieves the best ID ACC on 7 out of 8 datasets. For instance, on ENZYMES, SGOOD has ID ACC 48.66%, while the ID ACC of the best competitor GraphDE is 46.00%, indicating a relative improvement of 5.8%. On REDDIT-12K, a large-scale dataset with 11 ID classes, SGOOD has ID ACC

**Table 5: Ablation AUROC (%)**

Method	ENZYMES	IMDB-M	IMDB-B	BACE	BBBP	DrugOOD
Best baseline	71.46	69.26	79.39	73.78	57.37	57.37
SGOOD (base)	67.38	69.26	76.80	73.78	53.77	66.90
SGOOD\A	<u>73.60</u>	<u>75.22</u>	77.80	<u>75.96</u>	<u>57.84</u>	<u>68.80</u>
SGOOD	<b>74.41</b>	<b>78.84</b>	<b>80.42</b>	<b>84.40</b>	<b>61.25</b>	<b>73.16</b>

**Figure 3: ID and OOD score distributions, with the dotted line indicating the mean of ID/OOD scores.****Table 6: Comparison between different substructure detection methods by AUROC (%).**

SGOOD	ENZYMES	IMDB-M	IMDB-B	BACE	BBBP	DrugOOD
w.o. substructures	67.38	69.26	76.8	73.78	53.77	66.90
Modularity	<b>74.41</b>	<b>78.84</b>	<b>80.42</b>	<b>84.40</b>	<b>61.25</b>	<b>73.16</b>
Graclus	<u>71.12</u>	74.64	<u>78.86</u>	<u>79.54</u>	56.62	67.94
LP	68.09	<u>75.48</u>	78.46	76.63	54.90	<u>68.95</u>
BRICS	/	/	/	78.39	<u>60.18</u>	64.78

51.82%, outperforming the best competitor by a relative improvement of 5.9%. The results indicate that leveraging substructures benefits both OOD detection and graph classification.

## 4.3 Model Analysis

**Ablation Study.** In Table 5, SGOOD\A is SGOOD that ablates all augmentations in Section 3.2, *i.e.*,  $\alpha=0$  in Eq. (7); SGOOD (base) further ablates all substructure-related representations in Section 3.1. In Table 5, first observe that, from SGOOD (base) to SGOOD\A and then to the complete version SGOOD, the performance gradually increases on all datasets, validating the effectiveness of all proposed techniques. Second, SGOOD\A already surpasses the best baseline performance on most datasets, which demonstrates the effect of the techniques in Section 3.1, without the augmentation techniques in Section 3.2. With the help of the substructure-preserving graph augmentations, SGOOD pushes the performance further higher. In Figure 3, we visualize the ID and OOD score distributions of SGOOD (base), SGOOD\A and SGOOD on DrugOOD, with their mean scores shown in dotted lines. Clearly, we are obtaining more separable OOD scores against ID data from left to right in Figure 3, which demonstrates that our techniques in SGOOD can learn distinguishable representations for ID and OOD graphs.

**Effect of Task-agnostic Substructure Detection Methods.** As mentioned, SGOOD is orthogonal to specific substructure extraction methods. Here in SGOOD, we evaluate several commonly-used methods to extract substructures, including Graclus [6], label propagation (LP) [4], and BRICS [5]. BRICS uses chemistry knowledge for extraction. In Table 6, SGOOD with different substructure detection methods are all better than SGOOD w.o. substructures, and SGOOD

**Table 7: Comparing with subgraph-aware models AUROC (%). Bold: best. Underline: runner-up.**

Method	ENZYMES	IMDB-M	IMDB-B	BACE	BBBP	DrugOOD
SAG	70.40	76.50	77.30	<u>76.90</u>	<u>58.90</u>	65.80
TopK	70.20	<u>76.80</u>	77.20	74.20	54.90	58.50
DiffPool	73.30	75.90	78.00	76.50	57.50	70.60
NGNN	70.30	71.20	76.60	71.20	52.60	<b>75.60</b>
GNN-AK <sup>+</sup>	68.50	73.50	77.20	70.90	54.30	63.00
SGOOD	<b>74.41</b>	<b>78.84</b>	<b>80.42</b>	<b>84.40</b>	<b>61.25</b>	<u>73.16</u>

with Modularity is the best. The results validate the effectiveness of SGOOD that leverages substructures for OOD detection.

**Comparison with Task-specific Subgraph Models.** We then compare SGOOD directly with subgraph GNN models, including three hierarchical pooling methods (SAG [18], TopK [9], DiffPool [39]) and two subgraph GNNs (NGNN [42] and GNN-AK<sup>+</sup> [46]). Note that these methods are not specifically designed for graph-level OOD detection. At test time, we extract the graph representations generated by these methods and use Mahalanobis distance as OOD score. Table 7 reports the AUROC results. SGOOD performs best on 5 out of 6 datasets and is the top-2 on DrugOOD. This validates the effectiveness of our substructure-related techniques in Section 3 for graph-level OOD detection.

**Effect of Augmentations.** We evaluate the augmentations (SD, SG, and SS) in Section 3.2, with conventional graph augmentations that are not substructure-preserving, including edge perturbation (EP), attribute masking (AM), node dropping (ND), and subgraph sampling (SA). Table 8 reports the results, AM is not applicable on IMDB-M and IMDB-B since they do not have node attributes. Observe that our SD, SG, and SS are the top-3 ranked techniques for graph-level OOD detection, validating the effectiveness of the proposed substructure-preserving graph augmentations. In Appendix Figure 7, we also visualize the improvements of all pairwise combinations of our augmentation techniques.

**Performance under Different Backbones.** We evaluate SGOOD and competitors when changing the GIN backbone to GCN [17] and GraphSage [12]. Table 9 reports the results. With GCN backbone, compared with the baselines, SGOOD consistently achieves the best scores; with GraphSage backbone, SGOOD is the best on BACE, BBBP, DrugOOD, and the second best on other datasets. The results validate the robustness of SGOOD to different backbones.

**Model efficiency.** We compare the training time per epoch in seconds of all methods, with results in Table 10. Compared with other graph-level OOD detection competitors, including GraphDE, GOOD-D, and AAGOD, SGOOD requires less time to train. Compared with all methods, including the methods originally designed for image data, SGOOD requires moderate time for training. Considering together the time cost in Table 10 and the effectiveness in Table 3, we can conclude that SGOOD is effective and efficient for graph-level OOD detection,

#### 4.4 Parameter Sensitivity

In this section, we evaluate the performance of SGOOD under varying hyperparameters to test its robustness and sensitivity to parameter settings.

**Table 8: Comparing with different augmentations by AUROC.**

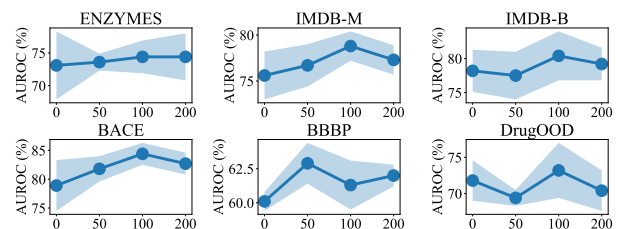
	ENZYMES	IMDB-M	IMDB-B	BACE	BBBP	DrugOOD	Avg. Rank
EP	74.28	76.50	78.44	78.75	58.24	71.32	4.67
AM	72.44	/	/	77.28	59.68	71.27	5.25
ND	73.11	77.09	78.40	78.79	58.59	69.48	4.83
SA	72.12	76.76	79.25	77.13	57.84	<b>72.66</b>	4.83
SD	<b>74.77</b>	<b>78.15</b>	79.54	82.00	59.76	72.65	<b>1.83</b>
SG	72.74	77.98	78.97	82.24	59.58	71.97	3.33
SS	74.27	76.20	<b>80.50</b>	<b>83.53</b>	<b>63.53</b>	71.94	2.67

**Table 9: Performance with different backbones by AUROC (%). Bold: best. Underline: runner-up.**

Backbone	Method	ENZYMES	IMDB-M	IMDB-B	BACE	BBBP	DrugOOD
GCN	MAH	<u>70.04</u>	<u>71.27</u>	53.46	72.68	54.97	66.01
	GraphDE	61.40	68.44	29.13	53.24	52.50	56.61
	GOOD-D	41.96	61.71	59.53	72.52	<u>58.91</u>	61.79
	OCGIN	64.35	57.46	<u>64.08</u>	67.54	51.23	59.30
	SGOOD	<b>71.26</b>	<b>73.52</b>	<b>65.91</b>	<b>83.42</b>	<b>62.76</b>	<b>72.52</b>
GraphSage	MAH	68.07	48.06	43.63	<u>73.60</u>	53.88	64.55
	GraphDE	61.37	<b>69.65</b>	28.28	53.24	52.50	56.66
	GOOD-D	45.55	57.02	23.90	73.15	<u>56.85</u>	61.57
	OCGIN	<b>71.75</b>	36.86	<b>71.44</b>	57.47	46.65	63.82
	SGOOD	<u>70.21</u>	<u>68.63</u>	<u>61.59</u>	<b>82.22</b>	<b>59.50</b>	<b>68.60</b>

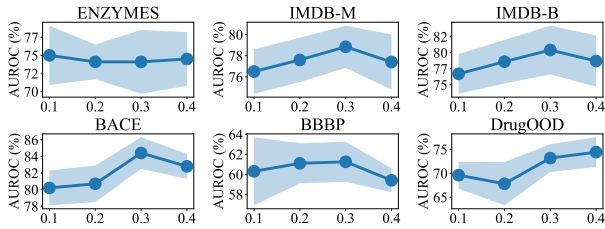
**Table 10: Training time per epoch of all methods on all datasets by seconds (s).**

Method	ENZYMES	IMDB-M	IMDB-B	REDDIT-12K	BACE	BBBP	HIV	DrugOOD
MSP	0.119	0.077	0.090	0.890	0.053	0.055	2.740	0.078
Energy	0.119	0.077	0.090	0.890	0.053	0.055	2.740	0.078
ODIN	0.119	0.077	0.090	0.890	0.053	0.055	2.740	0.078
MAH	0.119	0.077	0.090	0.890	0.053	0.055	2.740	0.078
GraphDE	1.692	1.175	1.392	176.400	0.950	0.696	43.770	1.020
GOOD-D	0.257	0.171	0.197	17.550	0.157	0.095	5.160	0.230
AAGOD	0.234	0.269	0.404	5.927	0.380	0.681	14.000	0.310
OCGIN	0.123	0.079	0.086	1.650	0.075	0.044	2.900	0.099
GLocalKD	0.072	0.203	0.054	142.000	0.052	0.035	4.220	0.067
SGOOD	0.161	0.138	0.137	0.980	0.085	0.058	3.970	0.124

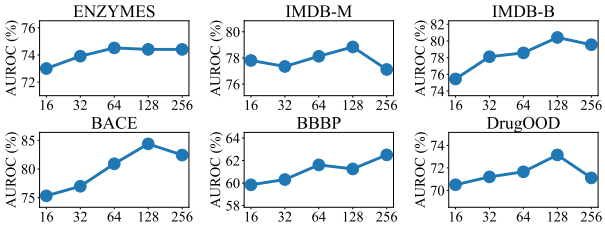
**Figure 4: OOD detection performance of SGOOD by AUROC (%) when the number of pretraining epochs  $T_{PT}$  varies from 0 to 200, with colored area representing standard deviation.**

**Varying pretraining epochs  $T_{PT}$ .** We conduct experiments to study the effect of pretraining epochs  $T_{PT}$  from 0 to 200. As shown in Figure 4, compared to SGOOD without first-stage pretraining ( $T_{PT} = 0$ ), pretraining improves SGOOD’s performance. We also found that excessive pretraining can sometimes have negative effects. For example, when  $T_{PT} = 200$ , SGOOD’s performance decrease on all datasets except ENZYMES. We speculate the reason is

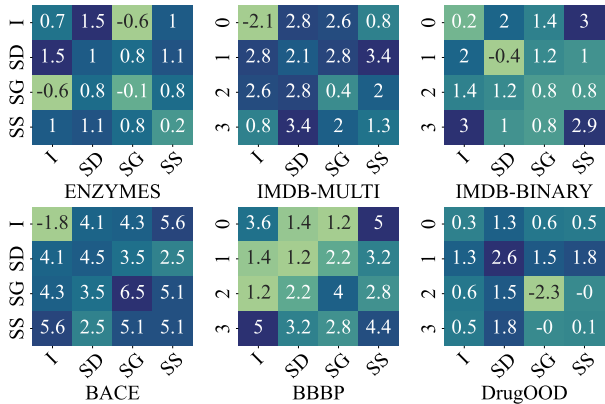




**Figure 5: OOD detection results of SGOOD by AUROC (%) when the weight of the contrastive loss  $\alpha$  varies from 0 to 1, with the colored area representing standard deviation.**



**Figure 6: OOD detection results of SGOOD by AUROC (%) when the batch size  $B$  varies from 16 to 256.**



**Figure 7: AUROC gain (%) of SGOOD compared with SGOOD\A without graph augmentations.**

that excessive pretraining makes task-agnostic information dominate, with a negative impact on the SGOOD’s ability to learn from class labels. As  $T_{PT} = 100$  generally leads to competitive performance across all datasets, we set the default value of  $T_{PT}$  to 100.

**Varying the weight of contrastive loss  $\alpha$ .** We vary  $\alpha$  from 0 to 1 to study the effect. As shown in Figure 5, compared to SGOOD fine-tuned solely by  $\mathcal{L}_{CE}$  (i.e.,  $\alpha = 0$ ), fine-tuning SGOOD with both  $\mathcal{L}_{CE}$  and  $\mathcal{L}_{CL}$  generally leads to better performance. As  $\alpha = 0.1$  usually leads to competitive performance across all datasets, we set the default value of  $\alpha$  to 0.1 in SGOOD.

**Varying number of negative samples in  $\mathcal{L}_{CL}$ .** We conduct experiments to study the effect of the number of negative samples used in contrastive loss  $\mathcal{L}_{CL}$  (Eq.(6)). Following the established convention in graph contrastive learning [40], pairs of augmented graphs originating from the same graph are treated as positive pairs,

**Table 11: Varying  $L_1$  and  $L_2$  in SGOOD (AUROC).**

$L_1$	$L_2$	ENZYMES	IMDB-M	IMDB-B	BBBP	BACE	DrugOOD
4	1	74.00	77.13	<b>81.00</b>	80.43	62.00	71.17
3	2	<b>74.41</b>	<b>78.84</b>	80.42	<b>84.40</b>	61.25	<b>73.16</b>
2	3	73.63	76.03	79.05	80.34	<b>62.26</b>	69.12
1	4	74.22	77.83	76.79	76.62	61.08	68.01

while pairs generated from different graphs within the batch are considered negative pairs. In such a way, in a  $B$ -size batch, for every  $G_i$ , it will have  $2B-2$  negative samples, as shown in the denominator of Eq.(6). Apparently the number of negative samples is related to batch size  $B$ . We vary  $B$  from 16 to 256 to evaluate sensitivity of SGOOD w.r.t. the number of negative samples, and report the results in Figure 6. Observe that as increasing from 16 to 128, the overall performance increases and then becomes relatively stable, which proves the effectiveness of the augmentation techniques developed in SGOOD and also validates the superior performance of SGOOD when varying batch size and the number of negative samples.

**Varying different combinations of augmentations.** In Figure 7, we exhaust the pairwise combinations of all options in  $\mathcal{A} = \{I, SD, SG, SS\}$  and visualize the AUROC gain on graph-level OOD detection over SGOOD\A without graph augmentations. As shown in Figure 7, most combinations achieve positive gains for effective OOD detection.

**Varying  $L_1$  and  $L_2$ .** In the experiments above, we fix the layers of the two GINs in the two-level graph encoding in Section 3.1 to be  $L_1 = 3$  and  $L_2 = 2$  as default. If we search  $L_1$  and  $L_2$ , it is possible to get even better OOD detection results, as shown in Table 11 where  $L_1$  and  $L_2$  are varied with their sum fixed to be 5. For example, on BACE with  $L_1=2$  and  $L_2=3$ , AUROC is 62.26%, about 1% higher than the default setting.

## 5 RELATED WORK

**Graph-level Representation Learning.** Graph-level representation learning aims to learn representations of entire graphs [34]. GNNs [12, 17, 31, 36] are often adopted [11, 38] to first learn node representations by message passing on graphs, and then node representations are aggregated by flat pooling functions to get graph-level representations [36]. However, these traditional methods have limitations in capturing high-order structures with crucial semantics for graph-level tasks, e.g., functional groups in molecules [39]. Hence, there exist methods to leverage subgraphs, e.g., hierarchical pooling [9, 18, 39] and subgraph GNNs [42, 46]. Hierarchical pooling methods learn to assign nodes into different clusters and coarsen graphs hierarchically. Subgraph GNNs apply message passing on extracted rooted-subgraphs of nodes in a graph, and then aggregate subgraph representations [8]. These methods are not designed for graph OOD detection, and they assume that graphs are i.i.d in training and testing and learn task-specific substructures. In this paper, we explore leveraging task-agnostic substructures to learn expressive graph representations for OOD detection.

**Graph Out-of-distribution Detection.** Out-of-Distribution (OOD) detection has recently received considerable research attention on

graph data. [33] explore node-level OOD detection by using energy function to detect OOD nodes in a graph, which is a different problem from this paper. For graph-level OOD detection, [21] design a generative model that has the ability to identify outliers in training graph samples, as well as OOD samples during the testing stage. [24] develop a self-supervised learning approach to train their model to estimate OOD scores at test time. Recently, [41] proposes to learn anomalous substructures using deep random walk kernel, which depends on labeled anomalous graphs, while OOD graphs are unseen during the training stage and only available during the testing stage. Instead of training GNNs for OOD detection, AAGOD [10] develops an adaptive amplifier that modifies the graph structure to enlarge the gap between OOD and ID graphs. Observe that existing graph-level OOD detection methods mainly leverage node representations output by GNNs [17, 31, 36, 47] to get graph-level representations, while the rich substructure patterns hidden in graphs are under-investigated for graph-level OOD detection. On the other hand, our method SGOOD explicitly uses substructures in graphs to learn high-quality representations for effective graph-level OOD detection.

## 6 CONCLUSION

We study the problem of graph-level OOD detection, and present a novel SGOOD method with superior performance. The design of SGOOD is motivated by the exciting finding that substructure differences commonly exist between ID and OOD graphs, and SGOOD aims to preserve more distinguishable graph-level representations between ID and OOD graphs. Specifically, we build a super graph of substructures for every graph, and develop a two-level graph encoding pipeline to obtain high-quality structure-enhanced graph representations. We further develop a set of substructure-preserving graph augmentations. Extensive experiments on real-world graph datasets with various OOD types validate the superior performance of SGOOD over existing methods for graph-level OOD detection.

## REFERENCES

- [1] Anna O Basile, Alexandre Yahi, and Nicholas P Tatonetti. 2019. Artificial intelligence for drug toxicity and safety. *Trends in pharmacological sciences* (2019).
- [2] Ulrik Brandes, Daniel Dellinger, Marco Gaertler, Robert Gorke, Martin Hoefler, Zoran Nikoloski, and Dorothea Wagner. 2007. On modularity clustering. *TKDE* (2007).
- [3] Aaron Clauset, Mark EJ Newman, and Christopher Moore. 2004. Finding community structure in very large networks. *Physical review E* (2004).
- [4] Gennaro Cordasco and Luisa Gargano. 2010. Community detection via semi-synchronous label propagation algorithms. In *BASNA*.
- [5] Jörg Degen, Christof Wegscheid-Gerlach, Andrea Zaliani, and Matthias Rarey. 2008. On the Art of Compiling and Using ‘Drug-Like’ Chemical Fragment Spaces. *ChemMedChem: Chemistry Enabling Drug Discovery* (2008).
- [6] Inderjit S Dhillon, Yuqiang Guan, and Brian Kulis. 2007. Weighted graph cuts without eigenvectors a multilevel approach. *TPAMI* (2007).
- [7] Yingdong Dou, Kai Shu, Congying Xia, Philip S Yu, and Lichao Sun. 2021. User preference-aware fake news detection. In *SIGIR*.
- [8] Fabrizio Frasca, Beatrice Bevilacqua, Michael Bronstein, and Haggai Maron. 2022. Understanding and extending subgraph gnn by rethinking their symmetries. *NeurIPS* (2022).
- [9] Hongyang Gao and Shuiwang Ji. 2019. Graph u-nets. In *ICLR*.
- [10] 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 29th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*. 638–648.
- [11] Zhichun Guo, Bozhao Nan, Yijun Tian, Olaf Wiest, Chuxu Zhang, and Nitesh V Chawla. 2023. Graph-based molecular representation learning. In *IJCAI*.
- [12] William L Hamilton, Rex Ying, and Jure Leskovec. 2017. Inductive representation learning on large graphs. In *NeurIPS*.
- [13] Dan Hendrycks and Kevin Gimpel. 2017. A baseline for detecting misclassified and out-of-distribution examples in neural networks. In *ICLR*.
- [14] 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. *NeurIPS* (2020).
- [15] Yuanfeng Ji, Lu Zhang, Jiaxiang Wu, Bingzhe Wu, Long-Kai Huang, Tingyang Xu, Yu Rong, Lanqing Li, Jie Ren, Ding Xue, et al. 2023. DrugOOD: Out-of-Distribution (OOD) Dataset Curator and Benchmark for AI-aided Drug Discovery—A Focus on Affinity Prediction Problems with Noise Annotations. *AAAI* (2023).
- [16] Dejun Jiang, Zhenxing Wu, Chang-Yu Hsieh, Guangyong Chen, Ben Liao, Zhe Wang, Chao Shen, Dongsheng Cao, Jian Wu, and Tingjun Hou. 2021. Could graph neural networks learn better molecular representation for drug discovery? A comparison study of descriptor-based and graph-based models. *Journal of cheminformatics* (2021).
- [17] Thomas N. Kipf and Max Welling. 2017. Semi-Supervised Classification with Graph Convolutional Networks. In *International Conference on Learning Representations*.
- [18] Junhyun Lee, Inyeop Lee, and Jaewoo Kang. 2019. Self-attention graph pooling. In *ICLR*.
- [19] Kimin Lee, Kibok Lee, Honglak Lee, and Jinwoo Shin. 2018. A simple unified framework for detecting out-of-distribution samples and adversarial attacks. *NeurIPS* (2018).
- [20] Haoyang Li, Xin Wang, Ziwei Zhang, and Wenwu Zhu. 2022. Ood-gnn: Out-of-distribution generalized graph neural network. *TKDE* (2022).
- [21] Zenan Li, Qitian Wu, Fan Nie, and Junchi Yan. 2022. Graphde: A generative framework for debiased learning and out-of-distribution detection on graphs. *NeurIPS* (2022).
- [22] Shiyu Liang, Yixuan Li, and R Srikant. 2018. Enhancing The Reliability of Out-of-distribution Image Detection in Neural Networks. In *ICLR*.
- [23] Weitang Liu, Xiaoyun Wang, John Owens, and Yixuan Li. 2020. Energy-based out-of-distribution detection. *NeurIPS* (2020).
- [24] Yixin Liu, Kaize Ding, Huan Liu, and Shirui Pan. 2023. GOOD-D: On Unsupervised Graph Out-Of-Distribution Detection. In *WSDM*.
- [25] Rongrong Ma, Guansong Pang, Ling Chen, and Anton van den Hengel. 2022. Deep graph-level anomaly detection by glocal knowledge distillation. In *WSDM*.
- [26] Yifei Ming, Yiyu Sun, Ousmane Dia, and Yixuan Li. 2023. How to Exploit Hyperspherical Embeddings for Out-of-Distribution Detection?. In *ICLR*.
- [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. *CoRR* (2020).
- [28] Chen Qiu, Marius Kloft, Stephan Mandt, and Maja Rudolph. 2022. Raising the bar in graph-level anomaly detection. In *Proceedings of the Thirty-First International Joint Conference on Artificial Intelligence*.
- [29] Yu Rong, Yatao Bian, Tingyang Xu, Weiyang Xie, Ying Wei, Wenbing Huang, and Junzhou Huang. 2020. Self-Supervised Graph Transformer on Large-Scale Molecular Data. In *NeurIPS*.
- [30] Minglai Shao, Jianxin Li, Feng Chen, Hongyi Huang, Shuai Zhang, and Xunxun Chen. 2017. An efficient approach to event detection and forecasting in dynamic multivariate social media networks. In *WWW*.
- [31] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. 2018. Graph Attention Networks. In *ICLR*.
- [32] Jim Winkens, Rudy Bunel, Abhijit Guha Roy, Robert Stanforth, Vivek Natarajan, Joseph R Leddam, Patricia MacWilliams, Pushmeet Kohli, Alan Karthikesalingam, Simon Kohl, et al. 2020. Contrastive training for improved out-of-distribution detection. *CoRR* (2020).
- [33] Qitian Wu, Yiting Chen, Chenxiao Yang, and Junchi Yan. 2022. Energy-based Out-of-Distribution Detection for Graph Neural Networks. In *ICLR*.
- [34] Zonghan Wu, Shirui Pan, Fengwen Chen, Guodong Long, Chengqi Zhang, and S Yu Philip. 2020. A comprehensive survey on graph neural networks. *TNNLS* (2020).
- [35] Zhenqin Wu, Bharath Ramsundar, Evan N Feinberg, Joseph Gomes, Caleb Geniesse, Aneesh S Pappu, Karl Leswing, and Vijay Pande. 2018. MoleculeNet: a benchmark for molecular machine learning. *Chemical science* (2018).
- [36] Keyulu Xu, Weihua Hu, Jure Leskovec, and Stefanie Jegelka. 2018. How Powerful are Graph Neural Networks?. In *ICLR*.
- [37] Pinar Yanardag and SVN Vishwanathan. 2015. Deep graph kernels. In *KDD*.
- [38] Nianzu Yang, Kaipeng Zeng, Qitian Wu, Xiaosong Jia, and Junchi Yan. 2022. Learning substructure invariance for out-of-distribution molecular representations. In *NeurIPS*.
- [39] Zhitao Ying, Jiaxuan You, Christopher Morris, Xiang Ren, Will Hamilton, and Jure Leskovec. 2018. Hierarchical graph representation learning with differentiable pooling. *NeurIPS* (2018).
- [40] Yuning You, Tianlong Chen, Yongduo Sui, Ting Chen, Zhangyang Wang, and Yang Shen. 2020. Graph contrastive learning with augmentations. *NeurIPS* (2020).
- [41] 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. *NeurIPS* (2022).

- [42] Muhan Zhang and Pan Li. 2021. Nested graph neural networks. *NeurIPS* (2021).
- [43] Yan Zhang, Jonathon Hare, and Adam Prugel-Bennett. 2019. Deep set prediction networks. *NeurIPS* (2019).
- [44] Zaixi Zhang, Qi Liu, Hao Wang, Chengqiang Lu, and Chee-Kong Lee. 2021. Motif-based graph self-supervised learning for molecular property prediction. *NeurIPS* (2021).
- [45] Lingxiao Zhao and Leman Akoglu. 2021. On using classification datasets to evaluate graph outlier detection: Peculiar observations and new insights. *Big Data* (2021).
- [46] Lingxiao Zhao, Wei Jin, Leman Akoglu, and Neil Shah. 2021. From Stars to Subgraphs: Uplifting Any GNN with Local Structure Awareness. In *ICLR*.
- [47] Ziang Zhou, Jieming Shi, Shengzhong Zhang, Zengfeng Huang, and Qing Li. 2023. Effective stabilized self-training on few-labeled graph data. *Inf. Sci.* 631 (2023), 369–384.