A Data-centric Framework to Endow Graph Neural Networks with Out-Of-Distribution Detection Ability

Yuxin Guo
Cheng Yang*
Beijing University of Posts and Telecommunications
Beijing, China

Yuluo Chen
Jixi Liu
Beijing University of Posts and Telecommunications
Beijing, China

Chuan Shi
Junping Du
Beijing University of Posts and Telecommunications
Beijing, China

ABSTRACT

Out-of-distribution (OOD) detection, which aims to identify OOD samples from in-distribution (ID) ones in test time, has become an essential problem in machine learning. However, existing works are mostly conducted on Euclidean data, and the problem in graph-structured data remains under-explored. Several recent works begin to study graph OOD detection, but they all need to train a graph neural network (GNN) from scratch with high computational cost. In this work, we make the first attempt to endow a well-trained GNN with the OOD detection ability without modifying its parameters. To this end, we design a post-hoc framework with Adaptive Amplifier for Graph OOD Detection, named AAGOD, concentrating on data-centric manipulation. The insight of AAGOD is to superimpose a parameterized amplifier matrix on the adjacency matrix of each original input graph. The amplifier can be seen as prompts and is expected to emphasize the key patterns helpful for graph OOD detection, thereby enlarging the gap between OOD and ID graphs. Then well-trained GNNs can be reused to encode the amplified graphs into vector representations, and pre-defined scoring functions can further convert the representations into detection scores. Specifically, we design a Learnable Amplifier Generator (LAG) to customize amplifiers for different graphs, and propose a Regularized Learning Strategy (RLS) to train parameters with no OOD data required. Experiment results show that AAGOD can be applied on various GNNs to enable the OOD detection ability. Compared with the state-of-the-art baseline in graph OOD detection, on average AAGOD has 6.21% relative enhancement in AUC and a 34 times faster training speed. Code and data are available at https://github.com/BUPT-GAMMA/AAGOD.

CCS CONCEPTS

• **Computing methodologies** → **Neural networks**.

KEYWORDS

Graph Neural Networks, Out-of-distribution Detection

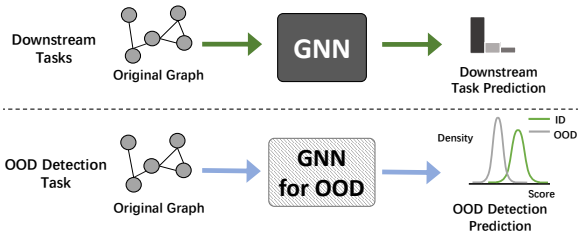
*Corresponding author, yangcheng@bupt.edu.cn

KDD '23, August 6–10, 2023, Long Beach, CA, USA.

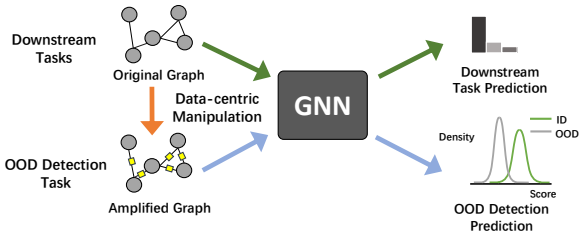
ACM ISBN 979-8-4007-0103-0/23/08…$15.00
https://doi.org/10.1145/3580305.3599244

ACM Reference Format:
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 (KDD '23), August 6–10, 2023, Long Beach, CA, USA.* ACM, New York, NY, USA, 11 pages. https://doi.org/10.1145/3580305.3599244

1 INTRODUCTION



(a) Typical retraining-based graph OOD detection methods [24, 39, 58].



(b) Our proposed data-centric framework for graph OOD detection.

Figure 1: Comparison of (a) existing methods that train a GNN specialized for the graph OOD detection task from scratch (*e.g.,* GOOD-D [24]) and (b) our proposed framework which enables an arbitrary well-trained GNN (trained for other tasks before) to excel at graph OOD detection without modifying its parameters via data-centric manipulation.

Modern machine learning models deployed in the open world often encounter out-of-distribution (OOD) input samples from a different distribution that the model has not yet seen during training. Ideally, a trustworthy model should not only perform well on ID samples but also recognize what they don't know [1, 21], especially when it comes to safety-critical applications [7, 18]. This highlights the significance of OOD detection [19, 32], which identifies ID and OOD data by their gaps in specified metrics or scores.

In view of this, a surge of recent works has explored various effective methods for OOD detection [6, 12, 19]. One line of work

proposes to train a neural network specialized for the OOD detection task [33, 35, 37], which we call the retraining-based OOD detection method. In practice, there are quantities of well-trained encoders (learned for other tasks) available to reuse. Hence, another line of work proposes to utilize the representations extracted from a well-trained encoder in a post-hoc manner instead of re-training a network [12, 15, 23], and we call it the reusing-based OOD detection method. In general, they first define a scoring function to map sample representations to scores, and then take those inputs with low scores as OOD samples. Without requiring any changes to the well-trained encoder, the reusing-based methods enjoy good efficiency and generalizability, and have attracted the attention of many researchers recently.

Recently, several pioneer studies [24, 39, 58] begin to study the OOD detection problem on graph data, but they all belong to the retraining-based method. For example, GOOD-D [24] proposes to train a new graph contrastive learning model for graph OOD detection from scratch, which utilizes hierarchical contrastive learning and detects OOD graphs based on the semantic inconsistency in different granularities. Nowadays, lots of cumbersome and deep graph neural network (GNN) encoders [5, 52] have been proposed for supervised or unsupervised scenarios. However, training such an encoder may suffer from complexity in computation [19] and difficulty in optimization [56]. Inspired by the success of reusing-based methods, we make the first attempt to enable a well-trained GNN to detect OOD graphs without altering its parameters.

As shown in Fig. 1, we propose a novel framework focusing on data-centric manipulation [55]. After a proper modification of the graph input, the well-trained GNN can be reused to encode the manipulated graph into vector representations, and pre-defined scoring functions (*e.g.,* those from previous reusing-based methods) can further convert the representations into detection scores. Intuitively, we expect the data-centric manipulation can help highlight the latent pattern of ID graphs, and thus enlarge the score gap between OOD and ID graphs. In this paper, we propose to implement the manipulation by superimposing a parameterized matrix (vividly named "**amplifier**") on the adjacency matrix of the original input graph as learnable instance-specific prompts [22, 26, 40]. However, finding the proper amplifier for each input graph remains non-trivial: (1) The number and order of nodes in graph data are uncertain, which makes it infeasible to design a global static amplifier suitable for all graphs; (2) OOD graphs are not available during training, which makes it hard to design a learning objective for distinguishing ID and OOD data.

To address the above difficulties, we propose a novel framework with **A**daptive **A**mplifier for **G**raph **O**OD **D**etection, named **AAGOD**. (1) To handle graphs with different sizes and unordered nodes, we innovatively design a Learnable Amplifier Generator (LAG) to adaptively generate graph-specific amplifiers. LAG will utilize the node representations encoded by the well-trained GNN to generate weights for the edges in the original graph. In this way, the knowledge in the well-trained graph encoder can be further reused, and LAG can customize amplifiers to determine which part of a graph should be highlighted for OOD detection. (2) To train the parameters in LAG without OOD data, we propose a Regularized Learning Strategy (RLS) that encourages high scores for amplified ID graphs and expects low scores when only seeing the amplifiers. In this way, we can ensure that the key factors leading to high scores (*i.e.,* ID patterns) only exist in ID graphs instead of amplifiers. In contrast, since LAG is not trained to perceive OOD graphs whose patterns are substantially different from those of ID graphs, the scores of amplified OOD graphs will keep low.

To validate the effectiveness of AAGOD, we conduct extensive experiments on real-world datasets from diverse domains and consider four models as well-trained GNNs, including two unsupervised methods (GCL [54], JOAO [53]) and two supervised methods (GIN [50], PPGN [29]). Experimental results show that AAGOD can successfully endow a well-trained GNN with the OOD detection ability, and has 25.21% average relative enhancement in AUC compared with a naive combination of typical reusing-based methods and well-trained GNN encoders. Moreover, AAGOD outperforms the SOTA methods in graph OOD detection, and gains 6.21% average relative enhancement in AUC compared to the best baseline. Besides, our AAGOD enjoys 34 times faster training speed than existing graph OOD detection methods, since we require no training of GNN encoders. We also investigate the interpretability of learned amplifiers by visualization.

Our contributions can be summarized threefold:

- To the best of our knowledge, we are the first work proposed for reusing-based graph OOD detection problem, which aims to endow a well-trained GNN with the OOD detection ability.
- We design an effective framework named AAGOD with the Learnable Amplifier Generator (LAG) and Regularize Learning Strategy (RLS), which can produce graph-specific amplifiers and enlarge the gap between amplified ID and OOD graphs.
- Extensive experiments on real-world datasets show that our proposed framework can be successfully applied on diverse GNNs and outperforms SOTA baselines in graph OOD detection.

2 RELATED WORK

2.1 Graph Neural Networks

Since the success of GCN [17] in modeling graph-structured data, lots of powerful GNNs have been proposed and shown promising results in both supervised and unsupervised scenarios [17, 43, 44, 54]. Now we will introduce several representative GNNs and apply our framework to them for graph OOD detection in the experiments.

In supervised and semi-supervised learning, GCN [17] is one of the most classical methods, and it performed layer-wise propagation based on node features to generate expressive representations. Further, GIN [50] leveraged an injective summation operation to make GNN as powerful as the WL test [47]. To break the limits of message passing mechanism [11], PPGN [29] proposed a simple and scalable model with guaranteed 3-WL expressiveness. For unsupervised settings, GCL [54] proposed several graph data augmentations and developed a general framework to learn graph representations based on different correlated views. Furthermore, JOAO[53] boosted GCL by selecting data augmentations automatically, adaptively, and dynamically instead of using ad hoc augmentation strategies.

2.2 Out-of-distribution Detection on Graphs

OOD detection [1, 2, 21] aims to identify OOD samples from ID ones, and has gained increasing traction due to its wide application [7, 18]. Extensive research of OOD detection for vision [12, 46] or language

data[38, 59] have been developed, while the analogous study on graph-structured data is greatly less investigated.

Recently, a few works began to focus on graph OOD detection. For instance, GKDE [58] designed a multi-source uncertainty framework for detecting OOD nodes. GPN [39] estimated the uncertainty of each node with the help of Bayesian posterior and density estimation. GOOD-D [24] introduced a carefully-crafted hierarchical graph contrastive learning model to detect OOD graphs based on their semantic inconsistency. However, these approaches need to train a GNN specialized for OOD detection from scratch, which may have high computation cost [19]. Differently, our proposed AAGOD has the advantages of previous reusing-based approaches [23, 46], enabling us to detect OOD graphs based on a well-trained GNN in a post-hoc manner. Without retraining a new graph encoder, AAGOD can benefit from high computational efficiency.

Another related research topic with graph OOD detection is graph anomaly detection [9, 10, 25]. Anomaly graphs are usually malicious samples [27] from a real system and available during training; while OOD graphs are samples from the distribution that the model has not seen during training. We also compare our AAGOD with SOTA graph anomaly detection methods [28, 57].

Besides, there are some concurrent works [16, 20, 48] relevant to our AAGOD. Specifically, GraphDE [20] and GNNSAFE [48] are model-centric methods for graph OOD detection. GTRANS [16] provides a data-centric method to improve the classification accuracy on OOD nodes, *i.e.*, OOD generalization task.

3 METHODOLOGY

In this section, we first introduce the notations and formalize the problem. Then we will present the proposed AAGOD framework for reusing-based graph OOD detection. Finally, we will discuss the benefits of our framework design.

3.1 Problem Statement

Notations. Let $\mathcal{P}_{\text{IN}}$ and $\mathcal{P}_{\text{OUT}}$ denote two distinct distributions defined in graph space. In the training phase, a graph dataset $\mathcal{D}_{\text{ID}} = \{G^1, ...G^n\}$ sampled from the in-distribution $\mathcal{P}_{\text{IN}}$ is available for model learning, where each graph $G^k = \{\mathcal{V}^k, \mathcal{E}^k, \mathbf{X}^k\}$ is with vertex set $\mathcal{V}^k$, edge set $\mathcal{E}^k$, and d_x-dimensional node features $\mathbf{X}^k \in \mathbb{R}^{|\mathcal{V}^k| \times d_x}$. We denote the adjacency matrix of graph G^k as $A^k \in \mathbb{R}^{|\mathcal{V}^k| \times |\mathcal{V}^k|}$, where each element $a^k_{ij} = 1$ for each edge $(v_i, v_j) \in \mathcal{E}^k$ and $a^k_{ij} = 0$ otherwise. Then the general purpose of graph OOD detection is to distinguish whether a graph belongs to in-distribution $\mathcal{P}_{\text{IN}}$ or not at the test phase.

Reusing-based Graph OOD Detection. In this paper, we make the first attempt to the reusing-based graph OOD detection, which performs detection by utilizing a well-trained GNN encoder in a post-hoc manner. Formally, with 0 indicating the OOD case and 1 the ID case, the reusing-based graph OOD detection aims to build a detection model $\text{detect}(\cdot)$ for distinguishing input graph G^k:

$$\text{detect}(G^k) = \begin{cases} 1, & g(G^k, f) \geq \gamma \\ 0, & g(G^k, f) < \gamma \end{cases} \tag{1}$$

where γ is a threshold, f is the well-trained GNN encoder with parameters Θ, and function g returns a score to estimate whether G^k is an ID graph or not with the help of f.

A Naive Solution. Since the GNN encoder f is sufficiently trained and can extract expressive graph representations, an intuitive idea for constructing the function g is to keep parameters Θ in encoder f untouched and directly apply a pre-defined non-parametric scoring function s on encoded representations. Formally, we can write $g(G^k, f)$ as follows:

$$g(G^k, f) = s\big(f(A^k, X^k; \Theta)\big), \tag{2}$$

where $f(A^k, X^k; \Theta)$ is the vector representation $\boldsymbol{h}^k$ of G^k encoded by GNN, and the scoring function $s(\cdot)$ measures the centrality of a sample in the representation space. Note that the function g includes no learnable parameters.

In this paper, we adopt two representative scoring functions: the local outlier factor detector (LOF) [4] and Mahalanobis distance-based detector (SSD) [36]. Specifically, LOF is one of the most classic methods and SSD is the state-of-the-art scoring function for OOD detection. To be self-contained, we will briefly introduce the two scoring functions.

LOF assumes the sample with low local density is more likely to be an outlier. For simplicity, it introduces a distance function $d_r(\boldsymbol{h}^k)$ to measure the distance between $\boldsymbol{h}^k$ and its r-th nearest neighbor, and calculates the local reachability density (lrd) of $\boldsymbol{h}^k$ via $d_r(\boldsymbol{h}^k)$. Then the final score of $\boldsymbol{h}^k$ is defined as:

$$s_{\text{LOF}}(\boldsymbol{h}^k) = \frac{1}{|N_r(\boldsymbol{h}^k)|} \sum_{\boldsymbol{h}^q \in N_r(\boldsymbol{h}^k)} \frac{lrd(\boldsymbol{h}^q)}{lrd(\boldsymbol{h}^k)}, \tag{3}$$

where $N_r(\boldsymbol{h}^k)$ is the collection of neighbors whose distance from $\boldsymbol{h}^k$ is less than $d_r(\boldsymbol{h}^k)$.

SSD leverages K-means clustering to separate the representations into T clusters, and then use the Mahalanobis distance between $\boldsymbol{h}^k$ and the corresponding cluster center to compute the score of $\boldsymbol{h}^k$. Formally, the scoring function of SSD is:

$$s_{\text{SSD}}(\boldsymbol{h}^k) = \frac{1}{\min_t (\boldsymbol{h}^k - \boldsymbol{\mu}_t)^\top \Sigma_t^{-1} (\boldsymbol{h}^k - \boldsymbol{\mu}_t)}, \tag{4}$$

where $\boldsymbol{h}^k$ belongs to the cluster t, $\boldsymbol{\mu}_t$ and Σ_t are the mean and covariance of representations in cluster t. Since all ID graphs are assumed to be drawn from the same distribution $\mathcal{P}_{\text{IN}}$, we simply set $T = 1$ in practice.

3.2 Adaptive Amplifier for OOD Detection

Instead of building a parameter-free function g as the above naive solution, we innovatively motivate our AAGOD to introduce additional learnable parameters in function g for better graph OOD detection performance. We will first present how we parameterize the function g, and then design a novel generator and an effective learning strategy for parameter training.

3.2.1 ***Overall Framework.*** The core of our AAGOD is to design a function $g(\cdot; \Omega)$ with learnable parameters Ω, which is trained to enlarge the score gap between ID graphs and OOD graphs:

$$\max_{\Omega} \mathbb{E}_{G^k \sim \mathcal{P}_{\text{IN}}} g(G^k; \Omega) - \mathbb{E}_{G^z \sim \mathcal{P}_{\text{OUT}}} g(G^z; \Omega). \tag{5}$$

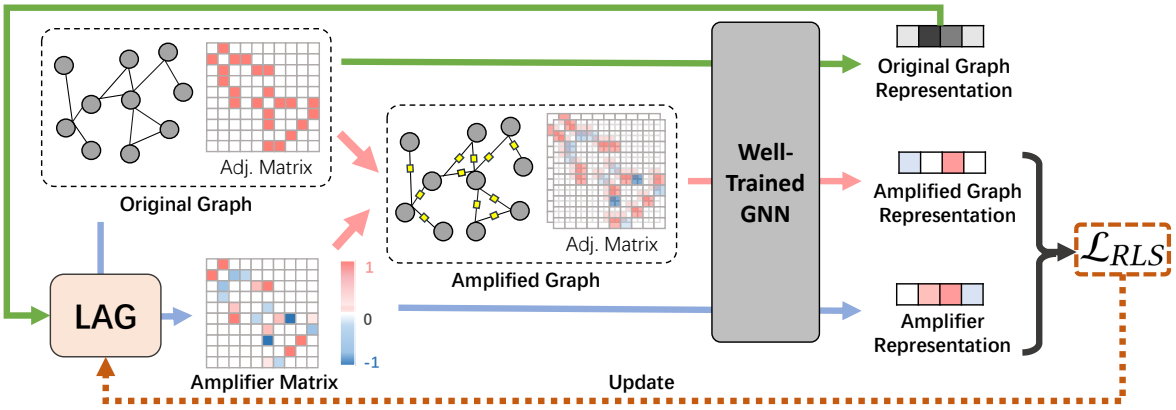


Figure 2: An illustration of the proposed AAGOD, which can enable a well-trained GNN to detect OOD graphs in a post-hoc manner. AAGOD adopts LAG to adaptively generate graph-specific amplifiers and RLS to train LAG without seeing OOD graphs.

As mentioned above, the well-trained GNN encoder $f(\cdot;\Theta)$ has fixed parameters Θ and the pre-defined scoring function $s(\cdot)$ is also non-parametric. Since it is hard to plug the parameters Ω into $f(\cdot;\Theta)$ or $s(\cdot)$, we propose to let the parameters Ω focus on data-level manipulation. Specifically, we superimpose a matrix (amplifier) on the adjacency matrix of the original input graph, which is expected to highlight the key sub-structures beneficial to OOD detection, thus making the ID graphs and OOD graphs easier to distinguish. The amplifier shares a similar principle with the prompt tuning technique, hoping to adapt to downstream tasks through data-centric operations. We can then rewrite $g(\cdot;\Omega)$ as:

$$g(G^k;\Omega) = s(f(A^k + \mathbf{M}_\Omega, X^k;\Theta)), \tag{6}$$

where $\mathbf{M}_\Omega$ is the amplifier matrix. However, finding a proper amplifier matrix is not trivial as graph-structured data have an arbitrary number of unordered nodes, and we also lack the information of unseen OOD graphs to optimize Eq. (5) directly. Therefore, we innovatively design a *Learnable Amplifier Generator* (LAG) to adaptively generate graph-specific amplifiers, and further propose a *Regularized Learning Strategy* (RLS) to approximate the optimization in Eq. (5) and learn the parameters Ω. The overall framework of our proposed AAGOD is shown in Fig. 2.

3.2.2 ***Learnable Amplifier Generator.*** To handle graphs with different sizes and unordered nodes, in this work we let the amplifier focus on generating weights for the edges in the original graph. Moreover, based on the assumption that graphs with similar topologies and node features should have similar ID patterns, LAG adaptively generates graph-specific amplifiers according to the encoded representations of original graphs from the well-trained GNN. In this way, the knowledge in the well-trained graph encoder can be reused, and LAG can customize amplifiers for graphs with different topological structures.

Specifically, since our proposed framework is model-agnostic and can be applied to any GNNs, we simply formalize the well-trained GNN by a two-step form:

$$\begin{aligned} \boldsymbol{p}_i^k &= \mathrm{MP}(v_i|\mathbf{A}^k, \mathbf{X}^k;\boldsymbol{\Theta}), \\ \boldsymbol{h}^k &= \mathrm{Pool}(\{\boldsymbol{p}_i^k|v_i \in \mathcal{V}^k\}), \end{aligned} \tag{7}$$

where $\boldsymbol{p}_i^k$ is the representation of node v_i in G^k, $\boldsymbol{h}^k$ is the final graph representation of G^k, MP is the message passing operation, and $\mathrm{Pool}(\cdot)$ is the pooling operation that transfers node representations to a holistic graph representation.

Note that the output node representations of the well-trained GNN encode not only node features but also local topology information. Therefore, we reuse the node representations of the well-trained GNN rather than raw features to discover the latent pattern of ID graphs and generate amplifiers. Formally, we use $\mathbf{M}^k \in \mathbb{R}^{|\mathcal{V}^k|\times|\mathcal{V}^k|}$ to describe the graph-specific amplifier of graph G^k, where the edge weight m_{ij}^k is designed to superimpose onto the edge between node v_i and v_j in G^k. Since the element of the graph-specific amplifier depends on the edges of the original graph, the elements $m_{ij}^k = 0$ when $a_{ij}^k = 0$. Otherwise, we calculate the elements m_{ij}^k as follows:

$$m_{ij;\Omega}^k = \mathrm{MLP}\left(\mathrm{Concat}(\boldsymbol{p}_i^k, \boldsymbol{p}_j^k);\Omega\right), \tag{8}$$

where $\mathrm{MLP}(\cdot;\Omega)$ represents a multi-layer perception (MLP) with parameters Ω, and $\mathrm{Concat}(\cdot)$ is the concatenation operator. For a graph $G^k \in \mathcal{D}_{\mathrm{ID}}$, we superimpose the the graph-specific amplifier $\mathbf{M}_\Omega^k$ on the adjacent matrix $\mathbf{A}^k$ to obtain an amplified graph $\mathbf{A}^k + \mathbf{M}_\Omega^k$. The amplified graph can be then encoded to $\boldsymbol{h}_\Omega^k$ by the well-trained GNN. We can rewrite Eq. (7) to formalize the above process as follows:

$$\boldsymbol{h}_\Omega^k = \mathrm{Pool}\left(\{\mathrm{MP}(v_i|\mathbf{A}^k + \mathbf{M}_\Omega^k, \mathbf{X}^k;\boldsymbol{\Theta})|v_i \in \mathcal{V}^k\}\right). \tag{9}$$

In this work, we use MLP to generate the elements of amplifiers without loss of generality. Although there may be alternatives to produce amplifiers, we reuse the well-trained GNN as much as possible and introduce few learnable parameters to achieve efficient graph OOD detection.

3.2.3 ***Regularized Learning Strategy.*** To train the parameters in LAG without OOD data, we propose an effective regularized learning strategy, which encourages amplified graphs with ID patterns to get higher scores, while regularizes LAG to avoid all amplified graphs getting high scores. In this way, we can ensure that the

Algorithm 1 Adaptive Amplifier for Graph OOD Detection

Require: Training set of ID graphs $\mathcal{D}_{\text{ID}}$, well-trained GNN model f with parameters Θ;
Ensure: LAG with learned parameters Ω;
1: Randomly initialize Ω;
2: **while** not converge **do**
3: **for** each graph $G^k \in \mathcal{D}_{\text{ID}}$ **do**
4: Compute $\boldsymbol{p}_i^k$ for each node $v_i \in \mathcal{V}^k$ by Eq. (7);
5: Compute the amplifier matrix $\mathbf{M}_\Omega^k$ by Eq. (8);
6: Compute $\boldsymbol{h}_{\text{ID};\Omega}^k$ and $\boldsymbol{h}_{\text{OOD};\Omega}^k$ by Eq. (9) and Eq. (11) with $\mathbf{M}_\Omega^k$, respectively;
7: **end for**
8: Update Ω by minimizing Eq. (13);
9: **end while**
10: **return** Learned parameters Ω for LAG.

ID patterns, which are the key factors leading to high scores, only exist in ID graphs instead of amplifiers. Thus, the scores will keep low for amplified OOD graphs.

Formally, the learning objective we designed is consistent with Eq. (5). For the first term in Eq. (5), we directly utilize the ID training data and denote the representation of the ID graph superimposing the graph-specific amplifier generated by LAG as $\boldsymbol{h}_{\text{ID};\Omega}^k$, which is calculated by Eq. (9). Then we design the learning objective for ID data by requiring a higher score for $\boldsymbol{h}_{\text{ID};\Omega}^k$ as follows:

$$\mathcal{L}_{\text{ID}} = \sum_{k=1}^{n} 1/s(\boldsymbol{h}_{\text{ID};\Omega}^k), \tag{10}$$

where n is the number of graphs in $\mathcal{D}_{\text{ID}}$. Note that to guarantee stability during training, we take the reciprocal of the score instead of the negative here.

By contrast, we expect the OOD graphs to remain low scores and propose to approximate the second term in Eq. (5) via regularizing the LAG to return low scores when only seeing the amplifiers. Formally, we encode the pure amplifier matrix $\mathbf{M}_\Omega^k$ generated for graph G^k, and then minimize the score of the amplifier. The learning strategy for OOD graphs can be given as follows:

$$\boldsymbol{h}_{\text{OOD};\Omega}^k = \text{Pool}\left(\{\text{MP}(v_i|\mathbf{M}_\Omega^k, \mathbf{X}^k; \Theta)|v_i \in \mathcal{V}^k\}\right), \tag{11}$$

$$\mathcal{L}_{\text{OOD}} = \sum_{k=1}^{n} s(\boldsymbol{h}_{\text{OOD};\Omega}^k), \tag{12}$$

Besides, we add another regularization term to prevent overfitting. Finally, the overall learning objective can be written as follows:

$$\mathcal{L}_{\text{RLS}} = \mathcal{L}_{\text{ID}} + \lambda\mathcal{L}_{\text{OOD}} + \tau \sum_{k=1}^{n} ||\mathbf{M}_\Omega^k||, \tag{13}$$

where λ and τ are the balance hyper-parameters, $||\cdot||$ is the L1-norm. The pseudo code of AAGOD is shown in Alg. 1.

3.3 Discussion

In this part, we will discuss the advantages of our proposed AAGOD framework in different aspects.

Easy to be Deployed. Our proposed AAGOD can exploit the inherent capability of well-trained GNNs, and endow them with the ability of OOD detection. The framework is compatible with diverse GNNs and scoring functions as it makes no assumptions about them. Furthermore, AAGOD conducts data-centric manipulation without modifying the original feed-forward computation of GNNs, and thus can be trained/used after the well-trained GNNs are deployed. Hence, our proposed AAGOD framework can be easily deployed in real-world systems.

Time and Space Complexity. The time complexity of the whole training process and the space complexity of our algorithm are both linear to the scale of datasets. Besides, the additional module we have introduced is only a two-layer MLP in Eq. (8), with a hidden dimension as 32 and output dimension as 1. Therefore, the extra parameters in our framework are few, and the overhead in AAGOD is more lightweight compared with the well-trained GNN encoders. In practice, the training speed of our AAGOD can be 34 times faster than retraining-based graph OOD detection methods [24].

Benefits of Amplifiers. The amplifier controls the edge weights in a graph, and can help emphasize the key positions helpful for OOD detection. For example, if the ID graphs in a molecule dataset share the similar backbone structure, then the amplifier should assign larger weights to the edges in the backbone. In fact, the edge weights in our learned amplifiers can provide some interpretable intuitions (detailed in Section 4.4), which is another advantage of our AAGOD compared with the naive solution. Besides, the amplifiers share some merit with the popular prompt tuning technique [22, 26, 40] in the pre-trained language modeling area, where the prompts will transform the original input texts into a form beneficial to downstream tasks. Hence, our amplifiers can also be interpreted as trainable instance-specific prompts for graph OOD detection.

4 EXPERIMENTS

In this section, we conduct experiments on five dataset pairs over four GNNs to answer the following research questions (RQs):

- RQ1: Can the proposed AAGOD improve the OOD detection performance of the well-trained GNN?
- RQ2: How effective is AAGOD compared with other graph OOD detection methods?
- RQ3: How about the efficiency of AAGOD compared with retraining a new GNN?
- RQ4: How about the hyper-parameter sensitivity of AAGOD?
- RQ5: What patterns can we observe from learned graph-specific amplifiers of AAGOD?

4.1 Experimental Setup

4.1.1 ***Datasets.*** Following previous works [24], we adopt five pairs of datasets as ID and OOD data, respectively. These datasets are selected from two mainstream graph data benchmarks (i.e., TU datasets [31] and OGB [14]), and each pair of datasets belongs to the same field while having mild domain shift. Specifically, three of these pairs are molecule datasets, one pair is a social network dataset and the last is a bioinformatics dataset. We also use the same dataset split strategy as [24], where 80% of ID graphs as the

Table 1: Graph OOD detection performance with unsupervised GNNs as GCL [54] and JOAO [53]. Here the subscript S/L indicates the SSD/LOF scoring function in Section 3.1. $GNN_{S/L}$ is the non-parametric naive solution, while $GNN_{S/L}$+ is our AAGOD.

ID	OOD	Metric	GCL_S	GCL_S+	Improv.	GCL_L	GCL_L+	Improv.	$JOAO_S$	$JOAO_S$+	Improv.	$JOAO_L$	$JOAO_L$+	Improv.
ENZYMES	PROTEIN	AUC ↑	62.97	**73.76**	+17.14%	62.56	**67.15**	+7.34%	61.20	**74.19**	+21.23%	59.68	**65.11**	+9.10%
		AUPR ↑	62.47	**75.27**	+20.49%	**65.45**	65.18	-0.41%	61.30	**77.10**	+25.77%	64.16	**64.49**	+0.51%
		FPR95 ↓	93.33	**88.33**	-5.36%	93.30	**85.00**	-8.90%	90.00	**81.67**	-9.26%	96.67	**85.00**	-12.07%
IMDBM	IMDBB	AUC ↑	80.52	**83.84**	+4.12%	61.08	**68.64**	+12.38%	80.40	**82.80**	+2.99%	48.25	**64.32**	+33.31%
		AUPR ↑	74.43	**80.16**	+7.70%	59.52	**68.03**	+14.30%	74.70	**77.77**	+4.11%	47.88	**61.62**	+28.70%
		FPR95 ↓	38.67	**38.33**	-0.88%	96.67	**91.33**	-5.52%	44.70	**42.00**	-6.04%	98.00	**94.00**	-4.08%
BZR	COX2	AUC ↑	75.00	**97.31**	+29.75%	34.69	**65.00**	+87.37%	80.00	**95.25**	+19.06%	41.80	**65.62**	+56.99%
		AUPR ↑	62.41	**97.17**	+55.70%	39.07	**62.89**	+60.97%	67.10	**94.34**	+40.60%	56.70	**67.22**	+18.55%
		FPR95 ↓	47.50	**15.00**	-68.42%	92.50	**80.00**	-13.51%	37.50	**12.50**	-66.67%	**97.50**	**97.50**	0.00%
TOX21	SIDER	AUC ↑	68.04	**71.27**	+4.75%	53.44	**58.25**	+9.00%	53.46	**69.39**	+29.80%	53.64	**55.67**	+3.78%
		AUPR ↑	69.28	**73.52**	+6.12%	56.81	**59.58**	+4.88%	56.02	**71.01**	+26.76%	**56.02**	**56.02**	0.00%
		FPR95 ↓	90.42	**89.53**	-0.98%	94.25	**92.72**	-1.62%	95.66	**90.55**	-5.34%	95.66	**89.66**	-6.27%
BBBP	BACE	AUC ↑	77.07	**80.64**	+4.63%	46.74	**50.53**	+8.11%	75.48	**78.54**	+4.05%	43.96	**51.28**	+16.65%
		AUPR ↑	68.41	**72.60**	+6.12%	45.35	**46.49**	+2.51%	69.32	**74.06**	+6.84%	44.77	**48.32**	+7.93%
		FPR95 ↓	71.92	**60.59**	-15.75%	92.12	**86.70**	-5.88%	76.85	**69.46**	-9.62%	94.09	**92.61**	-1.57%

training set, 10%/10% of ID graphs as well as the same number of OOD graphs are mixed together as the validation/test set.

4.1.2 ***Choices of Well-trained GNNs.*** Recall that our AAGOD can be applied to any well-trained GNNs. To fully evaluate the effectiveness of AAGOD, we select two categories of GNNs: unsupervised GNNs (*i.e.,* GCL [54] and JOAO [53]) and supervised GNNs (*i.e.,* GIN [50] and PPGN [29]). Specifically, GCL/GIN are classic methods, while JOAO/PPGN are more recent state-of-the-art unsupervised/supervised GNNs. The two supervised GNNs are trained by graph classification task. The details are as follows:

• GCL [54]: Following the same experimental settings as [54], we employ a 3-layer GIN encoder with hidden dimension as 32, weight decay of Adam optimizer as 0.01, and design heuristic search by exploring the learning rate from $\{0.001, 0.01, 0.1\}$. Moreover, we train GCL with 30 epochs and set the batch size as 128.

• JOAO [53]: As [53], we train a GIN encoder in JOAO with 3 layers and 32 hidden dimensions. We adopt the Adam optimizer with the same parameter settings as in GraphCL. Meanwhile, we tune the hyper-parameter that controls the trade-off during the optimization in the range of $\{0.01, 0.1, 1\}$.

• GIN [50]: We set GIN with 3 layers, the weight decay of Adam optimizer as 0.5. We explore the hidden dimension and learning rate from $\{128, 256\}$ and $\{0.001, 0.005\}$, respectively.

• PPGN [29]: We employ 3 concatenated PPGN blocks with hidden dimension selected from $\{16, 64, 128, 256\}$, learning rate in $\{0.001, 0.005\}$, learning rate decay in $\{0.5, 1\}$. We make each PPGN block generate a graph representation via the invariant max pooling [30], and then concatenate them to obtain the final graph representation.

4.1.3 ***Baseline Methods.*** As the first work for reusing-based graph OOD detection, we compare our proposed AAGOD with the naive solution introduced in Section 3.1 to demonstrate the effectiveness of our framework design. To further validate the superiority of our proposed AAGOD, we also compare with the SOTA graph OOD detection method (GOOD-D [24]) and two SOTA graph anomaly detection methods (OCGIN [57], GLocalKD [28]). It is worth noting that all three baselines need to train GNN models from scratch. A detailed introduction to these baselines is provided in Supplement A.2.

4.1.4 ***Implementation Details.*** For AAGOD, we leverage a 2-layer MLP in the learnable amplifier generator. Specifically, we adopt an Adam optimizer in MLP with weight decay as 0.1 and tune the learning rate in $[0.001, 0.1]$. The hidden dimension is 32 and the dropout probability is 0.8. We also use the sigmoid activation function to restrict the output edge weights of LAG. Meanwhile, we tune the parameter λ from $\{0.1, 1, 10, 50, 100\}$, τ from $\{0.01, 0.1, 1\}$. Moreover, we adopt the scoring functions mentioned in Section 3.1 for all GNNs, and annotate them in the model subscript (*e.g.,* GCL_S/GCL_L represent the naive solutions where GCL is combined with SSD/LOF scoring functions; GCL_S+/GCL_L+ are the corresponding versions with AAGOD). We run all experiments on a single GPU device of GeForce GTX 3090 with 22 GB memory.

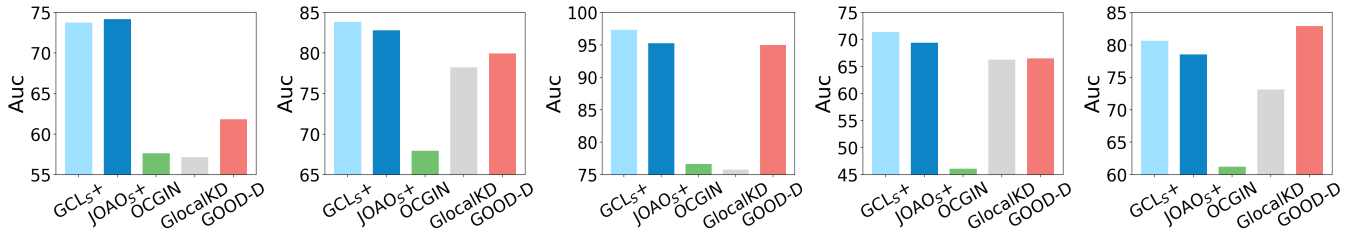
4.2 Analysis of Main Results

4.2.1 ***Comparison with the naive solution (RQ1).*** We present the results on five benchmark dataset pairs with two unsupervised and two supervised GNN models in Table 1 and 2, respectively. We bold the best results between the naive solution and AAGOD. The relative improvements of AAGOD over the naive solution are also reported. From the results, we have the following observations:

(1) The improvements via AAGOD are stable and substantial over all four GNN models on the five pairs of datasets. Compared with the naive solution based on the SSD scoring function, our corresponding AAGOD enhances the average AUC and AUPR by 30.28% and 25.76%, reduces the average FPR95 by 12.10% relatively. For the LOF scoring function, the relative improvements against the naive solution are 20.13% and 13.89% on AUC and AUPR, the

Table 2: Graph OOD detection performance with supervised GNNs as GIN [50] and PPGN [29]. Here the subscript S/L indicates the SSD/LOF scoring function in Section 3.1. $GNN_{S/L}$ is the non-parametric naive solution, while $GNN_{S/L}$+ is our AAGOD.

ID	OOD	Metric	GIN_S	GIN_S+	Improv.	GIN_L	GIN_L+	Improv.	$PPGN_S$	$PPGN_S$+	Improv.	$PPGN_L$	$PPGN_L$+	Improv.
ENZYMES	PROTEIN	AUC ↑	52.22	**66.22**	+26.81%	58.44	**65.89**	+12.75%	53.89	**66.67**	+23.71%	52.56	**63.22**	+20.28%
		AUPR ↑	50.41	**58.81**	+16.66%	53.82	**59.03**	+9.68%	54.06	**65.70**	+21.53%	51.21	**57.56**	+12.40%
		FPR95 ↓	93.33	**73.33**	-21.43%	90.00	**83.33**	-7.41%	**80.00**	**80.00**	0.00%	100.00	**83.33**	-16.67%
IMDBM	IMDBB	AUC ↑	42.05	**59.00**	+40.31%	57.24	**62.70**	+9.54%	40.62	**59.25**	+45.86%	47.90	**55.64**	+16.16%
		AUPR ↑	44.43	**57.82**	+30.14%	54.41	**62.21**	+14.34%	43.41	**55.04**	+26.79%	50.06	**52.76**	+5.39%
		FPR95 ↓	100.00	**90.67**	-9.33%	**87.17**	95.00	+8.98%	96.43	**85.17**	-11.68%	89.67	**89.33**	-0.38%
BZR	COX2	AUC ↑	35.25	**76.75**	+117.73%	60.75	**76.00**	+25.10%	62.75	**71.75**	+14.34%	65.00	**72.25**	+11.15%
		AUPR ↑	39.61	**66.32**	+67.43%	53.71	**63.18**	+17.63%	57.15	**78.94**	+38.13%	62.14	**77.59**	+24.86%
		FPR95 ↓	100.00	**70.00**	-30.00%	95.00	**45.00**	-52.63%	**65.00**	90.00	+38.46%	**80.00**	95.00	+18.75%
TOX21	SIDER	AUC ↑	63.73	**64.26**	+0.83%	51.47	**57.59**	+11.89%	36.98	**61.24**	+65.60%	54.61	**55.00**	+0.71%
		AUPR ↑	63.79	**67.35**	+5.58%	52.33	**56.55**	+8.06%	43.55	**58.16**	+33.55%	53.91	**57.91**	+7.42%
		FPR95 ↓	**83.78**	93.87	+12.04%	96.93	**92.34**	-4.74%	97.45	**82.63**	-15.21%	**94.38**	97.57	+3.38%
BBBP	BACE	AUC ↑	64.58	**67.80**	+4.99%	43.54	**57.13**	+31.21%	30.79	**70.20**	+128.00%	47.55	**56.96**	+19.79%
		AUPR ↑	58.39	**61.91**	+6.03%	43.80	**54.12**	+23.56%	44.06	**74.56**	+69.22%	49.71	**57.96**	+16.60%
		FPR95 ↓	**87.68**	90.64	+3.38%	**91.63**	92.12	+0.53%	97.56	**78.05**	-20.00%	100.00	**88.78**	-11.22%



(a) ENZYMES-PROTEIN (b) IMDBM-IMDBB (c) BZR-COX2 (d) TOX21-SIDER (e) BBBP-BACE

Figure 3: AUC of different methods on graph OOD detection. Higher AUC indicates better performance.

Table 3: Training time comparison between AAGOD and the corresponding well-trained GNNs. "Ratio" means the percentage of the running time of AAGOD to that of the original GNNs.

Dataset		Unsupervised						Supervised					
ID	OOD	GCL	GCL_S+	Ratio	JOAO	$JOAO_S$+	Ratio	GIN	GIN_S+	Ratio	PPGN	$PPGN_S$+	Ratio
ENZYMES	PROTEIN	30.65	1.99	6.48%	49.16	2.44	4.97%	47.48	4.99	10.52%	375.31	270.38	72.04%
IMDBM	IMDBB	44.57	6.39	14.33%	110.29	5.07	4.60%	43.45	12.68	29.19%	713.46	519.22	72.78%
BZR	COX2	17.78	1.84	10.34%	39.07	0.96	2.45%	39.87	6.35	15.92%	305.51	197.49	64.64%
TOX21	SIDER	460.76	16.66	3.62%	482.79	15.84	3.28%	509.40	12.76	2.50%	3474.90	2702.61	77.78%
BBBP	BACE	175.10	6.86	3.92%	134.74	5.84	4.34%	137.24	4.24	3.09%	1015.99	668.81	65.83%

reduction on FPR95 is 6.04%. This observation verifies our motivation of enabling well-trained GNNs to better detect OOD graphs, and demonstrates the effectiveness of our framework.

(2) Compared with the supervised GNNs, unsupervised GNNs are more suitable for graph OOD detection. In some challenging dataset pairs, supervised GNNs will unexpectedly encode OOD graphs into cluster centers in the representation space, and thus the naive solution is even worse than random guess. A possible reason is that representations encoded by supervised GNNs are more conducive to downstream tasks, whereas representations encoded by unsupervised GNNs are more universal and expressive. In spite of this, the relative improvements of AAGOD based on the unsupervised/supervised GNNs are 13.75/46.82% on AUC with the SSD scoring function, which means both unsupervised and supervised GNNs can benefit from our AAGOD. Additionally, the average relative improvements of GCL/JOAO/GIN/PPGN are 12.08/15.43/38.13/55.50% on AUC, which shows the potential generalizability on various GNN encoders of our proposed method.

(3) AAGOD has significant improvements for both SSD and LOF functions, which demonstrates that our AAGOD is compatible with different scoring functions. Also, the performance of SSD is generally better than that of LOF, aligning with the observations in previous work [24]. Thus we focus on the results based on SSD scoring function in the following experiments.

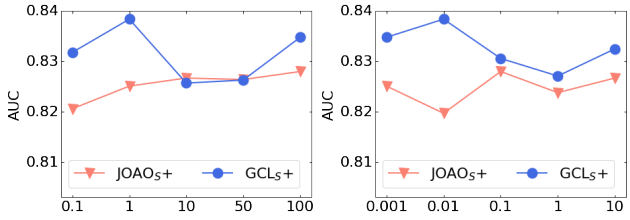


Figure 4: AUC sensitivity w.r.t. different hyper-parameters λ and τ on IMDBM-IMDBB over (a) GCL$_S$+ and (b) JOAO$_S$+.

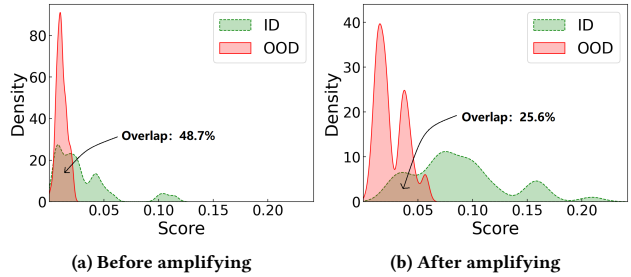


Figure 5: Scoring distributions before/after amplifying on BZR-COX2. Graphs with high (low) scores are regarded as ID (OOD) graphs, and AAGOD enlarges the distribution gap between ID and OOD graphs.

4.2.2 ***Comparison with the State-of-the-art Methods (RQ2).*** To validate our motivation that reusing a well-trained GNN to detect OOD graphs is feasible, we also compare our AAGOD based on GCL and JOAO with three competitive methods. Figure 3 shows the AUC of the five methods over the five pairs of datasets. We can see that GCL$_S$+ and JOAO$_S$+ significantly outperform other baselines on 4 out of 5 dataset pairs. In other words, our AAGOD can achieve better results without retraining any GNN encoders. Among the baseline methods, GOOD-D is the most competitive one while the other two graph anomaly detection methods perform poorly, which validates that the methods designed for graph anomaly detection task can not handle the OOD detection task well. This experiment shows the superiority of our framework in detecting OOD graphs.

4.3 Analysis of Extensive Studies

4.3.1 ***Analysis of Efficiency (RQ3).*** To demonstrate the efficiency of our proposed framework, we compare the training time of AAGOD with that of the corresponding well-trained GNN. We adopt AAGOD with the SSD scoring function and report the running time (seconds) in Table 3. From this table, we can see that the training time of AAGOD is only 23.68% of that of the corresponding GNN encoder on average, which shows that AAGOD is much more efficient than retraining a GNN encoder and validates our motivation to reuse well-trained GNNs. Moreover, we compare the training time with GOOD-D [24], the most competitive baseline. We find that on average our AAGOD (6.75s) runs 34 times faster than GOOD-D (229.31s), which indicates that AAGOD not only achieves better performance but also has higher efficiency. More details can be found in the Appendix.

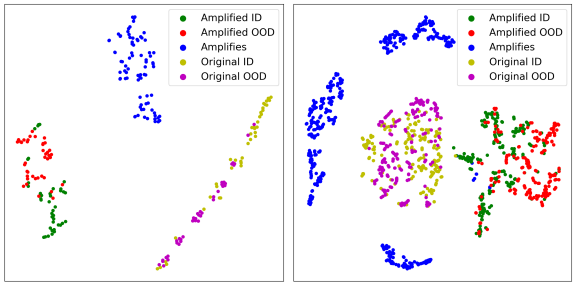


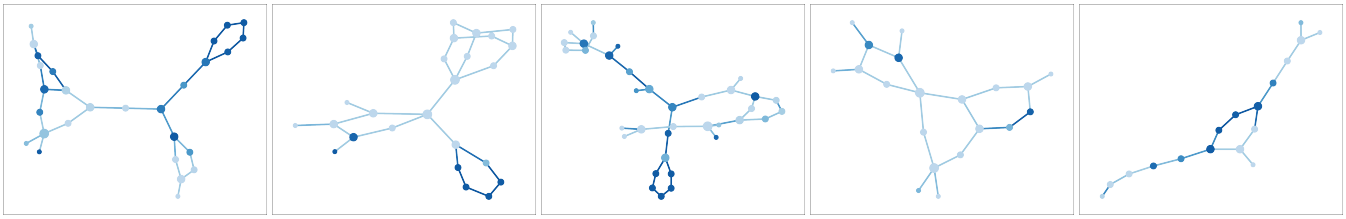
Figure 6: Visualization of graph representations on BZR-COX2 and BBBP-BACE. The representations of ID and OOD graphs (yellow v.s. purple) becomes more separable after amplification (green v.s. red).

4.3.2 ***Analysis of Hyper-parameters (RQ4).*** To analyze the impact of hyper-parameters λ and τ in Eq. (13), we conduct experiments on the IMDBM-IMDBB dataset pair for GCL$_S$+ and JOAO$_S$+. We vary λ in $\{0.1, 1, 10, 50, 100\}$ and τ in $\{0.001, 0.01, 0.1, 1, 10\}$, and present the AUC results in Fig. 4. The performance of AAGOD is stable when λ and τ change within a certain range. For example, if we fix $\lambda = 100$ and vary τ, the performance of GCL$_S$+ is in the range of 0.826 to 0.835. If we fix $\tau = 0.001$ and vary λ, the performance of GCL$_S$+ is in the range of 0.825 to 0.832. With different λ or τ, AAGOD always performs better than the naive solution around 0.805. In summary, our AAGOD is robust with respect to hyper-parameters λ and τ.

4.4 Analysis of Learned Amplifiers (RQ5)

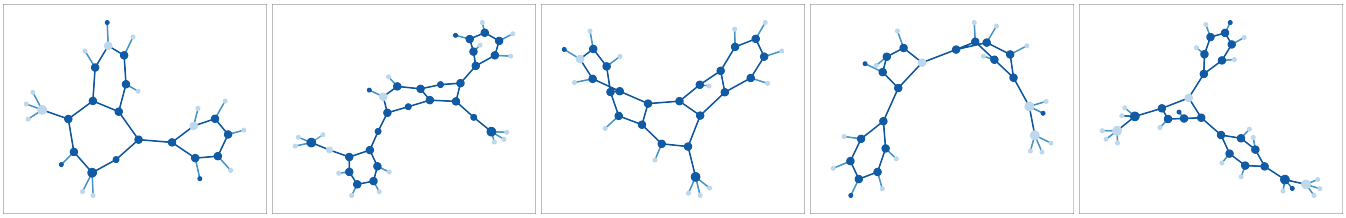
To interpret how AAGOD enhances the OOD detection ability of a well-trained GNN, we analyze the learned graph-specific amplifiers by presenting the following case studies based on GCL$_S$ and GCL$_S$+.

Visualization of Score Gap. To show that our data-centric manipulation can help enlarge the score gap between ID and OOD graphs, we present the scoring distributions before and after amplifying on BZR-COX2 dataset in Fig. 5. A larger score gap between ID and OOD graphs assures a better graph OOD detection performance. As we can see, the scores of both amplified ID and OOD graphs have been increased, while the growth of ID graphs is greater. Therefore, the learned amplifiers can enlarge the scoring distribution gap between ID and OOD graphs, thus improving the ability of the well-trained GNN to detect OOD graphs.



(a) ID (b) ID (c) ID (d) OOD (e) OOD

Figure 7: Visualization of different amplified graphs on BBBP-BACE. Here dark (light) edges/nodes have large (small) edge/self-loop weights, and the size of a node depends on its degree. The amplifiers emphasize the functional group with a hexagonal structure (*e.g.,* benzene ring) at the end of the backbone in ID graphs, which is the key structure making ID graphs different from OOD ones.



(a) ID (b) ID (c) ID (d) OOD (e) OOD

Figure 8: Visualization of different amplified graphs on BZR-COX2. The amplifiers highlight the backbone and ignore the side chains, since ID graphs usually have adjacent rings sharing common edges in the backbone, which is the critical characteristic making BZR dataset different from others.

Visualization of Representation Gap. Fig. 6 visualizes the graph representations via t-SNE [42] on BZR-COX2 and BBBP-BACE, and we use different colors to denote (amplified) ID/OOD graphs and amplifiers. We can see that the representations of original ID/OOD graphs mix with each other (yellow and purple dots), while those of amplified ID/OOD graphs (green and red dots) are more separable. Besides, the representations of amplifiers (blue dots) are far away from the ID graphs (green and yellow dots), which shows that there is no ID pattern in the amplifiers and demonstrates the effectiveness of our regularized learning strategy.

Visualization of Learned Amplifiers. To further understand learned amplifiers, Fig. 7 and 8 depict the amplified graphs on BBBP-BACE and BZR-COX2. Note that the edges in the amplified graphs are directed, and we average the weights of the same edge in different directions for convenience. As shown in Fig. 7, ID graphs all have a functional group with a hexagonal structure (*e.g.,* benzene ring) connected at the end of the backbone, which is a crucial pattern to distinguish ID graphs from others. Our learned amplifiers give edges in the functional group higher weights, indicating the ability to locate the key sub-structure of potential ID patterns. For Fig. 7, we can find that ID graphs have rings with common edges in the backbone, but rings in OOD graphs are independent. Therefore, the learned amplifiers on BZR-COX2 focus on underlining the graph backbone instead of side chains. Overall, the learned amplifiers can highlight the key positions helpful for graph OOD detection, and hence enlarge the difference between ID and OOD graphs.

5 CONCLUSION

In this paper, we innovatively propose to study the reusing-based graph OOD detection problem, which aims to endow an arbitrary well-trained GNN with the capability of identifying OOD graphs. To this end, we propose a data-centric framework named AAGOD, which can enlarge the score gap between ID and OOD graphs by superimposing amplifiers on the original input graphs. Specifically, we propose learnable amplifier generator to produce graph-specific amplifiers, and design an effective regularized learning strategy for parameter training. Extensive experiments on five pairs of real-world datasets show that AAGOD can successfully enable diverse well-trained GNNs to excel at graph OOD detection. AAGOD significantly outperforms the SOTA baselines in graph OOD detection, and enjoys a 34 times faster training speed. Moreover, we also present case studies to understand the learned graph-specific amplifiers more intuitively.

In future work, we will generalize our AAGOD for node-level OOD detection task as well, and explore the feasibility of applying the amplifiers to other graph learning scenarios to further improve their performance.

ACKNOWLEDGMENTS

We would like to thank the anonymous reviewers for their constructive feedback. This work is supported in part by the National Natural Science Foundation of China (No. 62192784, U1936104, U20B2045, 62172052, 62002029).

REFERENCES

[1] Dario Amodei, Chris Olah, Jacob Steinhardt, Paul Christiano, John Schulman, and Dan Mané. 2016. Concrete problems in AI safety. *arXiv preprint arXiv:1606.06565* (2016).

[2] Petra Bevandić, Ivan Krešo, Marin Oršić, and Siniša Šegvić. 2018. Discriminative out-of-distribution detection for semantic segmentation. *arXiv preprint arXiv:1808.07703* (2018).

[3] 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, suppl_1 (2005), i47–i56.

[4] Markus M Breunig, Hans-Peter Kriegel, Raymond T Ng, and Jörg Sander. 2000. LOF: identifying density-based local outliers. In *SIGMOD*. 93–104.

[5] Ming Chen, Zhewei Wei, Zengfeng Huang, Bolin Ding, and Yaliang Li. 2020. Simple and deep graph convolutional networks. In *ICML*. PMLR, 1725–1735.

[6] Xingyu Chen, Xuguang Lan, Fuchun Sun, and Nanning Zheng. 2020. A boundary based out-of-distribution classifier for generalized zero-shot learning. In *ECCV*. Springer, 572–588.

[7] Dengxin Dai and Luc Van Gool. 2018. Dark model adaptation: Semantic image segmentation from daytime to nighttime. In *ITSC*. IEEE, 3819–3824.

[8] Jesse Davis and Mark Goadrich. 2006. The relationship between Precision-Recall and ROC curves. In *ICML*. 233–240.

[9] Kaize Ding, Jundong Li, Nitin Agarwal, and Huan Liu. 2021. Inductive anomaly detection on attributed networks. In *IJCAI*. 1288–1294.

[10] Kaize Ding, Jundong Li, Rohit Bhanushali, and Huan Liu. 2019. Deep anomaly detection on attributed networks. In *ICDM*. SIAM, 594–602.

[11] Justin Gilmer, Samuel S Schoenholz, Patrick F Riley, Oriol Vinyals, and George E Dahl. 2017. Neural message passing for quantum chemistry. In *ICML*.

[12] Dan Hendrycks and Kevin Gimpel. 2017. A baseline for detecting misclassified and out-of-distribution examples in neural networks. *ICLR* (2017).

[13] Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. 2014. Distilling the knowledge in a neural network. *NeurIPS* (2014).

[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* 33 (2020), 22118–22133.

[15] Rui Huang, Andrew Geng, and Yixuan Li. 2021. On the importance of gradients for detecting distributional shifts in the wild. *NeurIPS* 34 (2021), 677–689.

[16] Wei Jin, Tong Zhao, Jiayuan Ding, Yozen Liu, Jiliang Tang, and Neil Shah. 2022. Empowering graph representation learning with test-time graph transformation. *arXiv preprint arXiv:2210.03561* (2022).

[17] Thomas N Kipf and Max Welling. 2017. Semi-supervised classification with graph convolutional networks. *ICLR* (2017).

[18] Matjaž Kukar. 2003. Transductive reliability estimation for medical diagnosis. *ARTIF INTELL MED* 29, 1-2 (2003), 81–106.

[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* 31 (2018).

[20] Zenan Li, Qitian Wu, Fan Nie, and Junchi Yan. 2022. Graphde: A generative framework for debiased learning and out-of-distribution detection on graphs. *Advances in Neural Information Processing Systems* 35 (2022), 30277–30290.

[21] Shiyu Liang, Yixuan Li, and Rayadurgam Srikant. 2018. Enhancing the reliability of out-of-distribution image detection in neural networks. *ICLR* (2018).

[22] Pengfei Liu, Weizhe Yuan, Jinlan Fu, Zhengbao Jiang, Hiroaki Hayashi, and Graham Neubig. 2023. Pre-train, prompt, and predict: A systematic survey of prompting methods in natural language processing. *Comput. Surveys* 55, 9 (2023), 1–35.

[23] Weitang Liu, Xiaoyun Wang, John Owens, and Yixuan Li. 2020. Energy-based out-of-distribution detection. *NeurIPS* 33 (2020), 21464–21475.

[24] Yixin Liu, Kaize Ding, Huan Liu, and Shirui Pan. 2023. GOOD-D: On Unsupervised Graph Out-Of-Distribution Detection. *WSDM* (2023).

[25] Yixin Liu, Zhao Li, Shirui Pan, Chen Gong, Chuan Zhou, and George Karypis. 2021. Anomaly detection on attributed networks via contrastive self-supervised learning. *TNNLS* 33, 6 (2021), 2378–2392.

[26] Zemin Liu, Xingtong Yu, Yuan Fang, and Xinming Zhang. 2023. GraphPrompt: Unifying Pre-Training and Downstream Tasks for Graph Neural Networks. In *Proceedings of the ACM Web Conference 2023*. 417–428.

[27] Xuexiong Luo, Jia Wu, Amin Beheshti, Jian Yang, Xiankun Zhang, Yuan Wang, and Shan Xue. 2022. Comga: Community-aware attributed graph anomaly detection. In *WSDM*. 657–665.

[28] Rongrong Ma, Guansong Pang, Ling Chen, and Anton van den Hengel. 2022. Deep Graph-level Anomaly Detection by Glocal Knowledge Distillation. In *WSDM*. 704–714.

[29] Haggai Maron, Heli Ben-Hamu, Hadar Serviansky, and Yaron Lipman. 2019. Provably powerful graph networks. *NeurIPS* 32 (2019).

[30] Haggai Maron, Heli Ben-Hamu, Nadav Shamir, and Yaron Lipman. 2019. Invariant and equivariant graph networks. *ICLR* (2019).

[31] 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).

[32] Eric Nalisnick, Akihiro Matsukawa, Yee Whye Teh, Dilan Gorur, and Balaji Lakshminarayanan. 2019. Do deep generative models know what they don't know? *ICLR* (2019).

[33] 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. *NeurIPS* 32 (2019).

[34] Lukas Ruff, Robert Vandermeulen, Nico Goernitz, Lucas Deecke, Shoaib Ahmed Siddiqui, Alexander Binder, Emmanuel Müller, and Marius Kloft. 2018. Deep one-class classification. In *ICML*. PMLR, 4393–4402.

[35] Thomas Schlegl, Philipp Seeböck, Sebastian M Waldstein, Ursula Schmidt-Erfurth, and Georg Langs. 2017. Unsupervised anomaly detection with generative adversarial networks to guide marker discovery. In *MICCAI*. Springer, 146–157.

[36] Vikash Sehwag, Mung Chiang, and Prateek Mittal. 2021. Ssd: A unified framework for self-supervised outlier detection. *ICLR* (2021).

[37] Joan Serrà, David Álvarez, Vicenç Gómez, Olga Slizovskaia, José F Núñez, and Jordi Luque. 2020. Input complexity and out-of-distribution detection with likelihood-based generative models. *ICLR* (2020).

[38] Zheyan Shen, Jiashuo Liu, Yue He, Xingxuan Zhang, Renzhe Xu, Han Yu, and Peng Cui. 2021. Towards out-of-distribution generalization: A survey. *arXiv preprint arXiv:2108.13624* (2021).

[39] Maximilian Stadler, Bertrand Charpentier, Simon Geisler, Daniel Zügner, and Stephan Günnemann. 2021. Graph posterior network: Bayesian predictive uncertainty for node classification. *NeurIPS* 34 (2021), 18033–18048.

[40] Mingchen Sun, Kaixiong Zhou, Xin He, Ying Wang, and Xin Wang. 2022. Gppt: Graph pre-training and prompt tuning to generalize graph neural networks. In *ACM SIGKDD*.

[41] Jeffrey J Sutherland, Lee A O'brien, and Donald F Weaver. 2003. Spline-fitting with a genetic algorithm: A method for developing classification structure- activity relationships. *Journal of Chemical Information and Computer Sciences* 43, 6 (2003), 1906–1915.

[42] Laurens Van der Maaten and Geoffrey Hinton. 2008. Visualizing data using t-SNE. *JMLR* 9, 11 (2008).

[43] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Lio, and Yoshua Bengio. 2018. Graph attention networks. *ICLR* (2018).

[44] Petar Velickovic, William Fedus, William L Hamilton, Pietro Liò, Yoshua Bengio, and R Devon Hjelm. 2019. Deep graph infomax. *ICLR* 2, 3 (2019), 4.

[45] Sachin Vernekar, Ashish Gaurav, Vahdat Abdelzad, Taylor Denouden, Rick Salay, and Krzysztof Czarnecki. 2019. Out-of-distribution detection in classifiers via generation. *NeurIPS* (2019).

[46] Qizhou Wang, Feng Liu, Yonggang Zhang, Jing Zhang, Chen Gong, Tongliang Liu, and Bo Han. 2022. Watermarking for Out-of-distribution Detection. *NeurIPS* (2022).

[47] Boris Weisfeiler and Andrei Leman. 1968. The reduction of a graph to canonical form and the algebra which appears therein. *nti, Series* 2, 9 (1968), 12–16.

[48] Qitian Wu, Yiting Chen, Chenxiao Yang, and Junchi Yan. 2023. Energy-based Out-of-Distribution Detection for Graph Neural Networks. *arXiv preprint arXiv:2302.02914* (2023).

[49] 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* 9, 2 (2018), 513–530.

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

[51] Pinar Yanardag and SVN Vishwanathan. 2015. Deep graph kernels. In *SIGKDD*. 1365–1374.

[52] Chengxuan Ying, Tianle Cai, Shengjie Luo, Shuxin Zheng, Guolin Ke, Di He, Yanming Shen, and Tie-Yan Liu. 2021. Do transformers really perform badly for graph representation? *NeurIPS* 34 (2021), 28877–28888.

[53] Yuning You, Tianlong Chen, Yang Shen, and Zhangyang Wang. 2021. Graph contrastive learning automated. In *ICML*. PMLR, 12121–12132.

[54] Yuning You, Tianlong Chen, Yongduo Sui, Ting Chen, Zhangyang Wang, and Yang Shen. 2020. Graph contrastive learning with augmentations. *NeurIPS* 33 (2020), 5812–5823.

[55] Daochen Zha, Zaid Pervaiz Bhat, Kwei-Herng Lai, Fan Yang, Zhimeng Jiang, Shaochen Zhong, and Xia Hu. 2023. Data-centric artificial intelligence: A survey. *arXiv preprint arXiv:2303.10158* (2023).

[56] Chiyuan Zhang, Samy Bengio, Moritz Hardt, Benjamin Recht, and Oriol Vinyals. 2021. Understanding deep learning (still) requires rethinking generalization. *Commun. ACM* 64, 3 (2021), 107–115.

[57] Lingxiao Zhao and Leman Akoglu. 2021. On using classification datasets to evaluate graph outlier detection: Peculiar observations and new insights. *Big Data* (2021).

[58] Xujiang Zhao, Feng Chen, Shu Hu, and Jin-Hee Cho. 2020. Uncertainty aware semi-supervised learning on graph data. *NeurIPS* 33 (2020), 12827–12836.

[59] Wenxuan Zhou, Fangyu Liu, and Muhao Chen. 2021. Contrastive out-of-distribution detection for pretrained transformers. *EMNLP* (2021).

[60] Ev Zisselman and Aviv Tamar. 2020. Deep residual flow for out of distribution detection. In *CVPR*. 13994–14003.

A APPENDIX

A.1 Descriptions of Datasets

Following the settings in GOOD-D [24], we employ five pairs of real-world benchmark datasets from diverse domains for experiments. The detailed descriptions of these datasets are as follows:

• ENZYMES and PROTEIN [3] are datasets consisting of macromolecules, where nodes represent secondary structure elements and the edge represents two nodes connected in an amino acid sequence or 3D space. The node feature denotes the type and chemical information of the element.

• IMDBM and IMDBB [51] are social networks extracted from actor collaborations, where nodes represent actors, and each edge represents whether two actors appear in the same movie.

• BZR and COX2 [41] are molecules datasets, composed of a set of ligands for the benzodiazepine receptor (BZR) and cyclooxygenase-2 (COX-2) inhibitors respectively.

• Tox21 and SIDER [14] are molecules datasets from MoleculeNet [49]. The nodes in molecule graphs represent atoms, and the edges connecting two nodes are chemical bonds.

• BBBP and BACE [14] are another molecules datasets, having similar configurations to Tox21 and SIDER.

A.2 Descriptions of Baseline Methods

In our experiments, we leverage the following three methods as our baselines:

• GOOD-D [24]: With the help of a carefully-crafted hierarchical graph contrastive learning framework, GOOD-D proposes to detect OOD graphs according to the semantic inconsistency in different granularities.

• OCGIN [57]: OCGIN is a one-class graph anomaly detection method, which proposes to optimize a GIN encoder by leveraging an SVDD [34] objective.

• GLocalKD [28]: Based on knowledge distillation [13], GLocalKD designs a novel deep graph anomaly detection method to detect both locally- and globally-anomalous graphs.

A.3 Descriptions of Evaluation Metrics

As used in previous works [45, 60], we measure the performance of graph OOD detection by three threshold-independent metrics [8]:

AUC is the area under the receiver operating characteristic curve and a higher AUC value represents a better detection performance.

Table 4: Training time comparison between AAGOD and GOOD-D.

ID	OOD	GOOD-D	GCL_S+
ENZYMES	PROTEIN	46.08	1.99
IMDBM	IMDBB	11.17	6.39
BZR	COX2	89.01	1.84
TOX21	SIDER	771.74	16.66
BBBP	BACE	228.54	6.86
Average		229.31	6.75

AUPR is the abbreviation of the area under the precision-recall curve (PR) similar to AUC while delivering a better performance evaluation for unbalanced data.

FPR95 is the false positive rate (FPR) while the true positive rate (TPR) reaching 95%, which is proposed to measure the probability that an OOD example is misclassified as ID when most ID samples are recalled. Specifically, a lower FPR95 value indicates a better detection performance.

A.4 Additional Experimental Results

A.4.1 Analysis of Efficiency. To further demonstrate the efficiency of our proposed AAGOD, we list the training time (seconds) of AAGOD based on GCL and GOOD-D in Table. 4. As we can see, our AAGOD consistently has less training time and enjoys 34 times faster training speed on average compared with GOOD-D. The experimental results validate our motivation of reusing a well-trained GNN to detect OOD graphs instead of retraining a new GNN from scratch.

A.4.2 Performance on More Datasets. To further validate the effectiveness of our proposed AGSKD, we test AAGOD on two additional dataset pairs used in GOOD-D [24]: ClinTox-LIPO and Esol- MUV. The AUCs of the naive solution and our AAGOD on ClinTox-LIPO and Esol- MUV are 50.83->**69.89** and 76.61->**91.18** with encoders trained by GraphCL. Hence our proposed framework can also achieve state-of-the-art level results with no encoder retraining required on other dataset pairs.

A.4.3 Additional Analysis of Figure 6. As the ID (BBBP) and OOD (BACE) graphs after amplifying still seem entangled in Figure 6, we sent the ID (BBBP) and OOD (BACE) representations before/after amplifying into an SVM classifier with ground truth labels as supervision, and their classification accuracies are 0.906/0.973. Therefore, the representations of amplified ID and OOD are more separable in fact. Due to the information loss caused by dimensionality reduction, they appear not well separated on the two-dimensional plane in Figure 6.

A.4.4 Analysis of Simply Adding Random Noise. Simply adding random noise to the original adjacency matrix to form OOD graphs is not feasible, since this way can not ensure that the ID patterns, which are the key factors leading to high scores, only exist in ID graphs instead of the OOD graphs we formed. To further verify the difference between our amplifier and random noise, we conduct experiments on dataset pair ENZYMES-PROTEINS to see the performance of simply adding random noise. Specifically, we add noise with different intensities to the original graph to form OOD data by randomly adding and removing edges. Then the ID and OOD graphs are fed into the well-trained GNN encoder to get their representations, and a classifier is trained to distinguish the ID and OOD representations. We present the results in Table and we can find that the OOD detection performance of adding random noise is very poor, even no different from random guessing.

Table 5: Graph OOD detection performance of simply adding random noise.

Noise intensity	0.1	0.5	0.8
AUC	41.50	58.72	47.50