2-2022

Deep graph-level anomaly detection by glocal knowledge distillation

Rongrong MA

Guansong PANG
Singapore Management University, gspang@smu.edu.sg

Ling CHEN

Anton Van Den HENGEL

Deep Graph-level Anomaly Detection
by Glocal Knowledge Distillation

Rongrong Ma
Australian Artificial Intelligence Institute
University of Technology Sydney
Sydney, Australia
rongrong.ma-1@student.uts.edu.au

Guansong Pang*
School of Computing and Information Systems
Singapore Management University
Singapore
gspang@smu.edu.sg

Ling Chen*
Australian Artificial Intelligence Institute
University of Technology Sydney
Sydney, Australia
ling.chen@uts.edu.au

Anton van den Hengel
Australian Institute for Machine Learning
The University of Adelaide
Adelaide, Australia
anton.vandenhengel@adelaide.edu.au

ABSTRACT

Graph-level anomaly detection (GAD) describes the problem of detecting graphs that are abnormal in their structure and/or the features of their nodes, as compared to other graphs. One of the challenges in GAD is to devise graph representations that enable the detection of both *locally-* and *globally-*anomalous graphs, *i.e.*, graphs that are abnormal in their fine-grained (node-level) or holistic (graph-level) properties, respectively. To tackle this challenge we introduce a novel deep anomaly detection approach for GAD that learns rich global and local normal pattern information by *joint random distillation* of graph and node representations. The random distillation is achieved by training one GNN to predict another GNN with randomly initialized network weights. Extensive experiments on 16 real-world graph datasets from diverse domains show that our model significantly outperforms seven state-of-the-art models. Code and datasets are available at https://git.io/GLocalKD.

CCS CONCEPTS

• **Computing methodologies → Semi-supervised learning settings**; **Neural networks**; *Anomaly detection.*

KEYWORDS

Graph-level anomaly detection, Graph neural networks, Knowledge distillation, Deep learning

*Corresponding author: Guansong Pang, Ling Chen.

1 INTRODUCTION

Graph-level anomaly detection (GAD) aims to identify graphs that are significantly different from the majority of graphs in a collection. The ability to record complex relationships between diverse entities renders graphs an essential and widely used representation in real-world applications. As a result, anomaly detection in graphs has broad applications in, *e.g.*, recognizing drugs with severe side-effects [18], identifying toxic molecules from chemical compound graphs [1], and breaking drug-smuggling networks [39].

Despite the prevalence of graph data and the importance of anomaly detection therein, GAD has received little attention compared to anomaly detection in other types of data [2, 28]. One primary challenge in GAD is to learn expressive graph representations that capture local and global normal patterns in the graph structure and attributes (*e.g.*, descriptive features of nodes). This is essential for the detection of both **locally-anomalous graph** – relating to individual nodes and their local neighborhood (G_5 in Figure 1) – and **globally-anomalous graph** – relating to holistic graph characteristics (G_6 in Figure 1).

A related research line is to explore the identification of unusual changes in graph structure from a time-evolving sequence of a single graph, in which most nodes and structure at different time steps do not change [9, 21, 37, 46, 47, 52]. GAD, in contrast, requires identifying graph anomalies among a set of graphs that lack the cohesion of a time-ordered progression and have diverse structure and node features, and it is significantly less explored.

Deep learning has shown tremendous success in diverse representation learning tasks, including the recently emerged graph neural networks (GNN)-based methods [43]. Also, deep anomaly detection models, such as autoencoder (AE)-based methods [7, 11, 54], generative adversarial network (GAN)-based methods [24, 34] and one-class classifiers [31, 32, 53], have shown promising performance on different types of data (*e.g.*, tabular data, image data, and video data) [28]. There is limited work exploiting GNNs for the GAD task, however. A number of GNN-based models have been introduced for
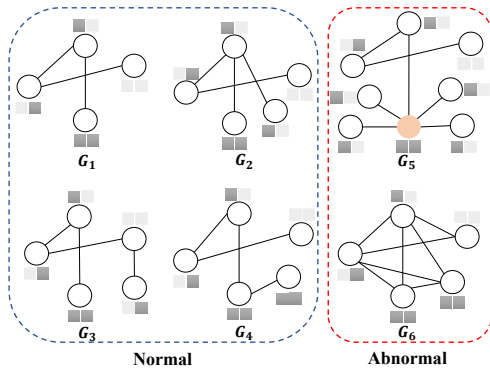
Figure 1: A set of graphs with two anomalous graphs indicated. The squares above/below the nodes represent node features. G_5 is a locally-anomalous graph due to the unusual local properties (*e.g.*, structure) of the orange node, while G_6 is a globally-anomalous graph because it does not conform to G_1 to G_4 in holistic graph properties.

(a) Random features   (b) Learned features   (c) Prediction errors

Figure 2: Demonstration of GLocalKD working on a popular dataset – AIDS. (a) Representations of training graphs output by the random target network. (b) Representations of training graphs learned by the predictor network. (c) Prediction errors (anomaly scores) of GLocalKD on test graphs. Visualization in (a) and (b) is based on t-SNE.

anomaly detection in graph data [8, 13, 16, 41, 51], but they focus on anomalous node/edge detection in a single large graph.

One challenge in adapting AE- and GAN-based detection methods to GAD is their dependence on reconstruction-error-based anomaly measures. This is because it is still challenging to faithfully reconstruct (or generate) graphs from a latent vector representation [43]. As shown by a comparative study in [50], the one-class model based on deep support vector data description (Deep SVDD) [32] may be adapted for GAD by directly optimizing the SVDD objective on top of GNN-based graph representations, but it focuses on detecting globally-anomalous graphs only. Further, its performance is largely restricted by the one-class hypersphere assumption of SVDD since there are often more complex distributions in the normal class in real-world datasets.

In this paper, we introduce a novel deep anomaly detection approach for GAD that learns both global and local normal patterns by **joint random distillation** of graph and node representations – global and local (*i.e.*, glocal) graph representation distillation. The random representation distillation is done by training one GNN to predict a **random GNN** that has its neural network weights fixed to random initialization, *i.e.*, the predictor network learns to produce the same representations as that in the random network, as shown in Figure 2(a) and (b). To accurately predict these fixed randomly-projected representations, the predictor network is enforced to learn all major patterns in the training data. By applying such a random distillation on both graph and node representations, our model learns glocal graph patterns across the given training graphs. When the training data consists of exclusively (or mostly) normal graphs, the learned patterns are a summarization of multi-scale graph regularity/normality information. As a result, given a graph that shows node/graph-level irregularity/abnormality w.r.t. these learned patterns, the model cannot accurately predict its representations, leading to a much larger prediction error than that of normal graphs, as shown in Figure 2(c). Thus, this prediction error can be defined as anomaly score to detect the aforementioned two types of graph anomalies.
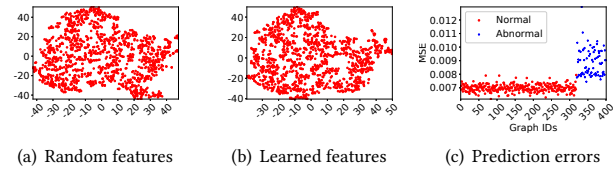
Accordingly, this paper makes the following major contributions:

- We formulate the GAD problem as the task of detecting locally- or globally-anomalous graphs (Sec. 3.1), and empirically verify the presence of these two types of graph anomalies in real-world datasets (Sec. 5.8).
- We introduce a novel deep anomaly detection framework that models *glocal graph regularity* and learns graph anomaly scores in an end-to-end fashion (Sec. 3.2). This results in the first approach specifically designed to effectively detect both types of anomalous graphs.
- A new GAD model, namely Global and Local Knowledge Distillation (GLocalKD), is further instantiated from the framework. GLocalKD implements the joint random distillation of graph and node representations by minimizing the graph- and node-level prediction errors of approximating a random graph convolutional neural network (Sec. 4). GLocalKD is easy-to-implement without requiring the challenging graph generation, and it can effectively learn diverse glocal normal patterns with small training data. It also shows remarkable robustness to anomaly contamination, indicating its applicability in both unsupervised (anomaly-contaminated unlabeled training data) and semi-supervised (exclusively normal training data) settings.

Extensive empirical results on 16 real-world datasets from chemistry, medicine, and social network domains show that (i) GLocalKD significantly outperforms seven state-of-the-art competing methods (Sec. 5.4); (ii) GLocalKD is substantially more sample-efficient than other deep detectors (Sec. 5.5), *e.g.*, it can use 95% less training samples to achieve the accuracy that still outperforms the competing methods by a large margin; and (iii) GLocalKD, using a single default GNN architecture, performs very stably w.r.t. different anomaly contamination rates (Sec. 5.6) and the dimensionality of the representations (Sec. 5.7).

2 RELATED WORK

2.1 Graph-level Anomaly Detection

Graph-based anomaly detection has drawn great attention in recent years [2], especially the recently emerged GNN-based approaches [8, 13, 16, 41, 51], but most of the studies focus on anomaly (*e.g.*,

anomalous nodes or edges) detection in a single large graph. Below we review related work on GAD.

Time-evolving Graphs. Most existing GAD studies are to identify anomalous graph changes in a sequence of time-evolving graphs [9, 17, 21, 37, 46, 47, 52]. However, these methods are designed to handle time-dependent graphs with very similar structure and difficult to generalize to graphs with large variations in the structure and/or descriptive features.

Static Graphs. Significantly less work has been done on anomalous graph detection in a set of static graphs. One research direction is to utilize powerful graph representation methods or graph kernels for GAD. A number of recent studies [25, 50] show promising GAD performance by applying off-the-shelf anomaly measures, such as isolation forest (iForest) [20], local outlier factor (LOF) [4], one-class support vector machine (OCSVM) [35], on top of vectorized graph representations learned by advanced graph kernels (such as Weisfeiler-Leman kernel (WL) [36] and propagation kernel (PK) [23]) or graph representation learning methods (such as Graph2Vec [22] and InfoGraph [38]). The key issue with these methods is that the graph representations are learned independently from the anomaly detectors, leading to suboptimal representations. Alternatively, there are studies on extracting graph-level patterns for GAD [25]. However, the performance of these methods is limited because graph-level patterns may differ significantly in graphs from different domains or application scenarios.

Deep Learning-based Methods. Despite great success of deep anomaly detection in different types of data [28], there is limited work done on GAD in this line. Deep graph learning techniques, such as graph convolutional network (GCN) [15] and graph isomorphism network (GIN) [45], have been powerful graph representation learning tools that empower diverse downstream tasks [43, 49]. Most of existing deep anomaly detection methods [7, 11, 27, 29, 31, 32, 34, 53, 54] depend heavily on data reconstruction/generative models. Consequently, the difficulty of reconstructing/generating graphs largely hinders the development of those deep methods for GAD. Zhao et al. [50] performs a large evaluation study on GAD, which shows that the Deep SVDD objective [32] can be applied on top of GNN-based graph representations for enabling GAD. Nevertheless, it is focused on high-level graph anomalies only and its performance is also restricted by the SVDD measure.

2.2 Knowledge Distillation

Knowledge Distillation (KD), where the initial goal is to train a simple model that distills the knowledge of a large model while maintaining similar accuracy as the large model, is first introduced in [12] and then extended to anomaly detection in a number of studies [3, 10, 19, 33, 44]. All of these methods train a simpler student network to distill the knowledge of **a pretrained teacher network on large-scale data**, such as ResNet/VGG networks pretrained on ImageNet [3, 33]. However, for learning tasks on graph-level data, no such general-purpose pretrained teacher networks are available; further, graph databases from different domains differ significantly from each other, which also prevents the application of this type of approach to the GAD task. Random knowledge distillation is originally introduced in [5] to address sparse reward problems in deep reinforcement learning (DRL). It uses the random distillation

errors to measure the novelty of states as some additional reward signals to encourage DRL agents' exploration in sparse-reward contexts. This idea is also used in [40] to regularize unsupervised representation learning, enabling better anomaly detection on tabular data. Inspired by this, we devise the GLocalKD model to jointly learn globally- and locally-sensitive graph normality. To the best of our knowledge, this is the first approach designed specifically for deep graph-level anomaly detection and for detecting both types of graph anomalies.

3 FRAMEWORK

3.1 Problem Statement

This work tackles the problem of end-to-end graph-level anomaly detection. Specifically, given a set of M normal graphs $\mathcal{G} = \{G_1, ..., G_M\}$, we aim at learning an anomaly scoring function $f : \mathcal{G} \rightarrow \mathbb{R}$, parameterized by Θ, such that $f(\hat{G}_i; \Theta) > f(\hat{G}_j; \Theta)$ if $\hat{G}_j$ conforms to $\mathcal{G}$ better than $\hat{G}_i$. In $\mathcal{G}$, each graph is denoted by $G = (\mathcal{V}_G, \mathcal{E}_G)$ with a vertex/node set $\mathcal{V}_G$ and an edge set $\mathcal{E}_G$. The graph structure for each G can be denoted by an adjacency matrix $\mathbf{A} \in \mathbb{R}^{N \times N}$ where N is the number of nodes in G, i.e., $\mathbf{A}(i, j) = 1$ if there exists an edge between nodes v_i and v_j ($\exists (v_i, v_j) \in \mathcal{E}_G$); and $\mathbf{A}(i, j) = 0$ otherwise. Each node of G, $v_i \in \mathcal{V}_G$, is further associated with a feature vector $\mathbf{x}_i \in \mathbb{R}^n$ if G is an *attributed graph*. G is otherwise a *plain graph*. As shown in our experiments, our approach is flexible to handle both types of graph data (see Table 1), and it also performs well in unsupervised settings where $\mathcal{G}$ is anomaly-contaminated and contains some unknown abnormal graphs (Sec. 5.6).

Anomalous graphs in a graph set can be classified into two categories, i.e., locally-anomalous graphs and globally-anomalous graphs, which are respectively defined as follows.

DEFINITION 1 (LOCALLY-ANOMALOUS GRAPH). *Given a graph data set $\mathcal{G} = \{G_i\}_i^M$, with each graph $G \in \mathcal{G}$ denoted by $G = (\mathcal{V}_G, \mathcal{E}_G)$, graph $\hat{G}$ is a locally-anomalous graph if $\hat{G}$ does not conform to the graphs in $\mathcal{G}$ due to the presence of some anomalous nodes $v, \forall v \in \mathcal{V}_{\hat{G}}$, that significantly deviate from similar nodes in the graphs in $\mathcal{G}$.*

DEFINITION 2 (GLOBALLY-ANOMALOUS GRAPH). *Given a graph data set $\mathcal{G} = \{G_i\}_i^M$, graph $\hat{G}$ is a globally-anomalous graph if the holistic graph properties of $\hat{G}$ do not conform to that of the graphs in $\mathcal{G}$.*

We aim to train a detection model that can detect these two types of abnormal graphs. Note that the detection of locally-anomalous graphs is different from anomalous node detection in [8, 13, 16, 41] because the former is to detect graphs by evaluating the nodes/edges across a set of independent and separate graphs while the latter is to detect nodes/edges given a set of dependent nodes and edges from a single graph.

3.2 The Proposed Framework

To solve the above problem, we propose an end-to-end scoring framework that synthesizes two graph neural networks and joint random knowledge distillation of graph and node representations to train a deep anomaly detector. The resulting model can effectively detect both types of anomalous graphs.

3.2.1 Overview of the Framework. Our framework jointly distills graph-level and node-level representations of each graph, to learn both global and local graph normality information. It consists of two graph neural networks – a fixed randomly initialized target network and a predictor network – with exactly the same architecture and two distillation losses. It learns the holistic (fine-grained) graph normality by training the predictor network to predict the graph (node) level representations produced by the random target network. Let $\mathbf{h}_G$ and $\hat{\mathbf{h}}_G$ respectively be the graph representation of G yielded by the predictor and target networks, and $\mathbf{h}_i$ and $\hat{\mathbf{h}}_i$ be the respective node representation for a node v_i in G produced by the two networks, the overall objective of our approach can be given as:

$$L = L_{graph} + \lambda L_{node}, \tag{1}$$

where λ is a hyperparameter that balances the importance of the two loss functions, L_{graph} and L_{node} are respective graph-level and node-level distillation loss functions:

$$L_{graph} = \frac{1}{|\mathcal{G}|} \sum_{G \in \mathcal{G}} KD\left(\mathbf{h}_G, \hat{\mathbf{h}}_G\right), \tag{2}$$

$$L_{node} = \frac{1}{|\mathcal{G}|} \sum_{G \in \mathcal{G}} \left(\frac{1}{|G|} \sum_{v_i \in \mathcal{V}_G} KD\left(\mathbf{h}_i, \hat{\mathbf{h}}_i\right) \right), \tag{3}$$

where $KD(\cdot, \cdot)$ is a distillation function that measures the difference between two feature representations and $|\mathcal{G}|$ is the number of graphs in $\mathcal{G}$.

The overall procedure of the training stage of our framework is shown in Figure 3, which works as follows:

(1) We first randomly initialize a graph network $\hat{\phi}$ as the target network and fix its weight parameters $\hat{\Theta}$. For every given graph G, it will yield a graph-level representation $\hat{\mathbf{h}}_G$ and node-level representation $\hat{\mathbf{h}}_i$ for each node v_i in G.

(2) A predictor network ϕ, with the same architecture as $\hat{\phi}$, is parameterized by Θ and trained to predict the representation outputs of the target network $\hat{\phi}$. That is, for every given graph G, it produces the graph-level representation $\mathbf{h}_G$ and the node-level representation $\mathbf{h}_i$, $\forall v_i \in \mathcal{V}_G$.

(3) Lastly, for graph G, $\hat{\mathbf{h}}_G$, $\mathbf{h}_G$, $\hat{\mathbf{h}}_i$, and $\mathbf{h}_i$ are integrated into a loss function L, which is minimized to train the predictor network ϕ.

At the evaluation stage, the anomaly score for a given graph G is defined as

$$f(G; \hat{\Theta}, \Theta^*) = KD\left(\mathbf{h}_G, \hat{\mathbf{h}}_G\right) + \lambda \frac{1}{|G|} \sum_{v_i \in \mathcal{V}_G} KD\left(\mathbf{h}_i, \hat{\mathbf{h}}_i\right), \tag{4}$$

where Θ^* are the learned parameters of the predictor network.

3.2.2 Key Intuition. The graph-level and node-level representations of graphs are learned by GNNs, whose powerful capabilities of capturing graph structure and semantic information have been proved in various learning tasks and applications. The joint random distillation in our framework forces both graph representations and node representations of the predictor network to be as close as possible to the corresponding outputs of the fixed random target network on normal graph data. This resembles the extraction of different patterns (either frequently or infrequently) presented in
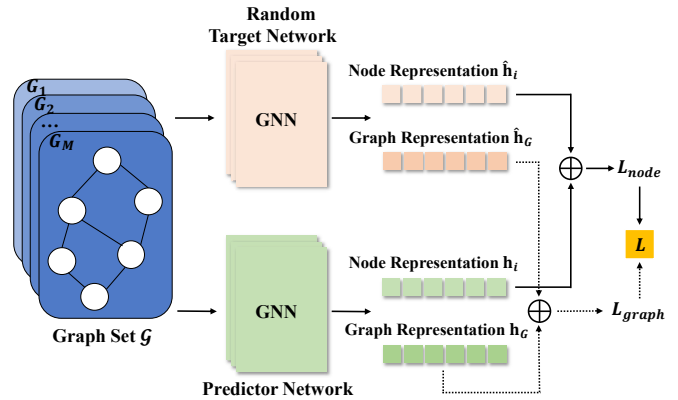

Figure 3: The proposed framework

the random representations of graphs and nodes, respectively. If a pattern frequently occurs in the random representation space, the pattern would be distilled better, *i.e.*, the prediction error in Eq. 2 or 3 is small due to a large sample size of the pattern; and the prediction error is large otherwise. As a result, our joint random distillation learns such regularity information from both graph and node representations. For a given test graph G, its anomaly score $f(G; \hat{\Theta}, \Theta^*)$ would be large if it does not conform to the regularity information embedded in the training graph set $\mathcal{G}$ at either the graph or the node level, *e.g.*, G_5 and G_6 in Figure 1; and $f(G; \hat{\Theta}, \Theta^*)$ would be small otherwise, *e.g.*, $G_1 - G_4$ in Figure 1.

4 JOINT RANDOM DISTILLATION OF GRAPH AND NODE REPRESENTATIONS

The proposed framework is instantiated into a method called Global and Local Knowledge Distillation (GLocalKD), in which we use widely-used graph convolutional network (GCN) to learn node and graph representations and the joint distillation is driven by two mean square error-based loss functions.

4.1 Graph Neural Network Architecture

4.1.1 Random Target Network. We first establish a target network with randomly initialized weights to obtain graph- and node-level representations in the random space. Different graph representation approaches may be used to generate the required representations as the prediction targets of the predictor network. Theoretically, various deep graph networks, such as GCN, GAT and GIN, can be employed as the graph representation learning module. In our work, a standard GCN is used, because GCN and its variants have proved their power to learn expressive features of graphs and good computational efficiency [43, 49].

Specifically, $\hat{\phi}(\cdot, \hat{\Theta}) : G = (\mathcal{V}_G, \mathcal{E}_G) \rightarrow \mathbb{R}^{N \times k}$ is a GCN with fixed randomly initialized weights $\hat{\Theta}$ (*i.e.*, the GCN is frozen after random weight initialization), where N is the number of nodes in G and k is a predefined dimensionality size of node representations. For each graph $G = (\mathcal{V}_G, \mathcal{E}_G)$ in $\mathcal{G}$, $\hat{\phi}(\cdot)$ takes adjacency matrix $\mathbf{A}$ and feature matrix $\mathbf{X}$ as input, and maps each node $v_i \in \mathcal{V}_G$ to the representation space using $\hat{\Theta}$. Let $\hat{\mathbf{h}}_i^l$ be the hidden representation

of node v_i in the lth layer, which is formally computed as follows:

$$\hat{\mathbf{h}}_i^l = \rho \left(\sum_{j \in \widetilde{\mathcal{N}}(i)} \frac{1}{\sqrt{\widetilde{\mathbf{D}}(i,i)\widetilde{\mathbf{D}}(j,j)}} \hat{\mathbf{h}}_j^{l-1} \hat{\Theta}^{l-1} \right) \tag{5}$$

where $\hat{\mathbf{h}}_j^{l-1}$ represents the hidden representation of node v_j in the $(l-1)^{th}$ layer, $\rho(a) = max(0,a)$ is the ReLU activation function, $\mathcal{N}(i)$ denotes the 1st-order neighbors of v_i and $\widetilde{\mathcal{N}}(i) = \mathcal{N}(i) \cup \{v_i\}$, $\mathbf{D}$ is a diagonal degree matrix with $\mathbf{D}(i,i) = \sum_j \mathbf{A}(i,j)$, $\widetilde{\mathbf{D}} = \mathbf{D} + \mathbf{I}$ ($\mathbf{I}$ is an identity matrix), and the input representation of v_i in the 0th layer, $\hat{\mathbf{h}}_i^0$, is initialized by its feature vector in $\mathbf{X}$, *i.e.*, $\hat{\mathbf{h}}_i^0 = \mathbf{X}(i,:)$. Thus, the output random node representation $\hat{\mathbf{h}}_i$ for node v_i can be written as:

$$\hat{\mathbf{h}}_i = \rho \left(\sum_{j \in \widetilde{\mathcal{N}}(i)} \frac{1}{\sqrt{\widetilde{\mathbf{D}}(i,i)\widetilde{\mathbf{D}}(j,j)}} \hat{\mathbf{h}}_j^{K-1} \hat{\Theta}^{K-1} \right) \tag{6}$$

where K is the number of layers of $\hat{\phi}(\cdot)$. The feature matrix $\mathbf{X}$ is composed of node attributes for attributed graphs. For plain graphs, following [48], we use the node degree as the node feature to construct a simple $\mathbf{X}$, since the degree of nodes is one of the key information for the discriminability of nodes and graphs.

Next, a READOUT operation is applied to the node representations to obtain the graph-level representation for G. There have been a number of READOUT operations introduced, *e.g.*, maxing, averaging, summation and concatenation [43, 49]. Considering that our goal is to detect anomalies, we need to aggregate extreme features across the node representations. Thus, the max-pooling is employed in the READOUT operation:

$$\hat{\mathbf{h}}_G = \max \{\hat{\mathbf{h}}_i, \forall v_i \in \mathcal{V}_G\}. \tag{7}$$

4.1.2 Predictor Network. The predictor network is a graph network used to predict the output representations of the target network, $\hat{\mathbf{h}}_i$ and $\hat{\mathbf{h}}_G$. We employ a GCN with the exactly same structure as the target network as the predictor network, which is denoted as $\phi(\cdot, \Theta) : G = (\mathcal{V}_G, \mathcal{E}_G) \rightarrow \mathbb{R}^{N \times k}$ with the weight parameters Θ to be learned. Then, similar to $\hat{\phi}$, $\phi(\cdot, \Theta)$ yields the node representation $\mathbf{h}_i$ for node v_i by the following formulation:

$$\mathbf{h}_i = \rho \left(\sum_{j \in \widetilde{\mathcal{N}}(i)} \frac{1}{\sqrt{\widetilde{\mathbf{D}}(i,i)\widetilde{\mathbf{D}}(j,j)}} \mathbf{h}_j^{K-1} \Theta^{K-1} \right) \tag{8}$$

After the same READOUT operation as in $\hat{\phi}$, the graph representation $\mathbf{h}_G$ is computed as follows:

$$\mathbf{h}_G = \max \{\mathbf{h}_i, \forall v_i \in \mathcal{V}_G\}. \tag{9}$$

Thus, the only difference between the random target network $\hat{\phi}(\cdot, \hat{\Theta})$ and the predictor network $\phi(\cdot, \Theta)$ is that $\hat{\Theta}$ is fixed after random initialization while Θ needs to be learned through the following glocal knowledge distillation.

4.2 Glocal Regularity Distillation

We further perform glocal regularity distillation by minimizing the distance between the (graph- and node-level) representations produced by the predictor network and the target network. Specifically,

the graph-level and node-level distillation loss are defined as:

$$L_{graph} = \frac{1}{|\mathcal{G}|} \sum_{G \in \mathcal{G}} \|\mathbf{h}_G - \hat{\mathbf{h}}_G\|^2, \tag{10}$$

$$L_{node} = \frac{1}{|\mathcal{G}|} \sum_{G \in \mathcal{G}} \left(\frac{1}{|G|} \sum_{v_i \in \mathcal{V}_G} \|\mathbf{h}_i - \hat{\mathbf{h}}_i\|^2 \right). \tag{11}$$

To learn the global and local graph regularity information simultaneously, our model is optimized by jointly minimizing the above two losses:

$$L = L_{graph} + L_{node}. \tag{12}$$

That is, λ in Eq. 1 is set to one in Eq. 12 since it is believed that it is equivalently important to detect both of locally- and globally-anomalous graphs. We will discuss in Sec. 4.4 in more details about why our model can learn the global and local graph regularity.

4.3 Anomaly Detection of Using GLocalKD

By joint global and local random distillation, the learned representations in our predictor network capture the regularity information at both the graph and node levels. Specifically, given a test graph sample G, its anomaly score is defined by the prediction errors in both graph and node-level representations:

$$f(G; \hat{\Theta}, \Theta^*) = \left\| \mathbf{h}_G - \hat{\mathbf{h}}_G \right\|^2 + \frac{1}{|G|} \sum_{v_i \in \mathcal{V}_G} \|\mathbf{h}_i - \hat{\mathbf{h}}_i\|^2. \tag{13}$$

This indicates that the locally- and globally-anomalous graph anomalies are treated equally important in our anomaly scoring, sharing the same spirit as the overall objective in Eq. 12.

4.4 Theoretical Analysis of GLocalKD

We show below that GLocalKD can normally produce a larger anomaly score for an abnormal graph than that for a normal one. Specifically, consider a regression problem with data distribution $\hat{\mathcal{G}} = \{G_i, y_i\}_i$ (y_i is the regression target) and a Bayesian setting in which a prior $p(\Theta^\star)$ over the parameters of a GCN $\phi(\cdot, \Theta^\star)$ is considered. The aim is to calculate the posterior after iteratively updating on the data. According to [5], our task can then be formulated as the optimization problem below:

$$\min_{\Theta} \mathrm{E}_{(G_i, y_i) \sim \hat{\mathcal{G}}} \|\phi(G_i, \Theta) + \phi(G_i, \Theta^\star) - y_i\|^2 + \mathcal{R}(\Theta), \tag{14}$$

where $\mathcal{R}(\Theta)$ is a regularization term from the prior [26]. Let $\mathcal{F}$ be the distribution over functions $f_\Theta = \phi(\cdot, \Theta) + \phi(\cdot, \Theta^\star)$, where Θ is the solution of Eq. 14 and $\Theta^\star$ is drawn from $p(\Theta^\star)$, then the ensemble $\mathcal{F}$ can bee seen as an approximation of the posterior [26].

When we select the graphs from the same distribution and set the label y_i to zero, the optimization problem

$$\arg\min_{\Theta} \mathrm{E}_{(G_i, y_i) \sim \hat{\mathcal{G}}} \|\phi(G_i, \Theta) + \phi(G_i, \Theta^\star)\|^2 \tag{15}$$

is equivalent to distilling a randomly drawn function from the prior. From this perspective, each entry of the representation outputs of the target and the predictor networks would correspond to a part of an ensemble and the prediction error would be an estimate of the predictive variance of the ensemble when the ensemble is assumed to be unbiased, as discussed in [5]. If we consider $\phi(\cdot, \Theta^\star)$ as the target network with randomly initialized $\Theta^\star$ and regard $\phi(\cdot, \Theta)$ as the predictor network, the prediction errors of the node

Table 1: AUC results (mean±std) on 16 real-world graph datasets. # Graphs: the number of graphs, # Nodes and # Edges: the average number of nodes and edges in each graph. The best performance is boldfaced.

Dataset	# Graphs	# Nodes	# Edges	InfoGraph		WL		PK		OCGCN	GLocalKD
				iForest	LESINN	iForest	LESINN	iForest	LESINN		
PROTEINS_full	1,113	39.06	72.82	0.464±0.019	0.336±0.047	0.639±0.018	0.712±0.053	0.627±0.009	0.572±0.031	0.718±0.036	**0.785**±0.034
ENZYMES	600	32.63	62.14	0.483±0.027	0.528±0.046	0.498±0.029	0.624±0.050	0.493±0.013	0.608±0.033	0.613±0.087	**0.636**±0.061
AIDS	2,000	15.69	16.2	0.703±0.036	0.955±0.023	0.632±0.050	0.584±0.016	0.476±0.014	0.421±0.010	0.664±0.080	**0.992**±0.004
DHFR	467	42.43	44.54	0.489±0.015	**0.625**±0.028	0.466±0.013	0.596±0.056	0.467±0.013	0.568±0.054	0.495±0.080	0.558±0.030
BZR	405	35.75	38.36	0.528±0.060	0.731±0.071	0.533±0.032	0.720±0.032	0.525±0.052	**0.775**±0.063	0.658±0.071	0.679±0.065
COX2	467	41.22	43.45	0.580±0.052	0.670±0.079	0.532±0.027	0.590±0.056	0.515±0.036	**0.671**±0.039	0.628±0.072	0.589±0.045
DD	1,178	284.32	715.66	0.475±0.012	0.310±0.034	0.699±0.006	0.638±0.045	0.706±0.010	**0.833**±0.023	0.605±0.086	0.805±0.017
NCI1	4,110	29.87	32.3	0.494±0.009	0.598±0.035	0.545±0.008	**0.743**±0.015	0.532±0.006	0.670±0.012	0.627±0.015	0.683±0.015
IMDB	1,000	19.77	96.53	0.520±0.028	0.565±0.017	0.442±0.032	**0.612**±0.046	0.442±0.035	0.585±0.047	0.536±0.148	0.514±0.039
REDDIT	2,000	429.63	497.75	0.457±0.003	0.262±0.027	0.450±0.013	0.239±0.028	0.450±0.012	0.487±0.013	0.759±0.056	**0.782**±0.016
HSE	8,417	16.89	17.23	0.484±0.026	**0.657**±0.051	0.477±0.000	0.528±0.000	0.489±0.003	0.469±0.016	0.388±0.041	0.591±0.001
MMP	7,558	17.62	17.98	0.539±0.022	0.571±0.037	0.475±0.000	0.307±0.000	0.488±0.002	0.322±0.008	0.457±0.038	**0.676**±0.001
p53	8,903	17.92	18.34	0.511±0.014	0.520±0.025	0.473±0.000	0.390±0.000	0.486±0.004	0.329±0.001	0.483±0.017	**0.639**±0.002
PPAR-gamma	8,451	17.38	17.72	0.521±0.023	0.541±0.036	0.510±0.000	0.461±0.000	0.499±0.017	0.388±0.015	0.431±0.043	**0.644**±0.001
COLLAB	5,000	74.49	2,457.78	0.453±0.003	0.319±0.033	0.506±0.020	0.536±0.014	0.529±0.023	**0.550**±0.043	0.401±0.183	0.525±0.014
hERG	655	26.48	28.79	0.607±0.033	0.701±0.048	0.665±0.042	**0.802**±0.047	0.679±0.034	0.798±0.052	0.569±0.049	0.704±0.049
			p-value	0.0005	0.0262	0.0004	0.1089	0.0005	0.1337	0.0018	–

representations as well as graph representations in the predictor network would be an estimate of the predictive variance of the results of two networks. In other words, our training process aims to train a predictor network so that the node representations and graph representations of the two networks on each training sample are as close as possible. Then, for the graph with patterns similar to many other training graphs, the prediction errors in Eqs. 10 and 11 are small, *i.e.*, small predictive variance in Eq. 14, because there are sufficient such samples to train the prediction model; the abnormal graphs, by contrast, are drawn from different distributions from the training graphs and dissimilar to most of the training data, leading to large predictive variance in Eq. 14. Thus, the prediction errors in our joint random distillation can distinguish both locally- and globally-anomalous graphs from normal graphs.

5 EXPERIMENTS AND RESULTS

5.1 Datasets

As shown in Table 1, we employ sixteen publicly available real-world datasets[1], which are collected from various critical domains [14]. The first six datasets in Table 1 are attributed graphs, *i.e.*, each node has some descriptive features; the others are plain graphs. HSE, MMP, p53 and PPAR-gamma are datasets with real anomalies. The other 12 datasets are taken from graph classification benchmarks and converted for anomaly detection tasks by treating the minority class as anomalies, following [6, 20, 29]. These datasets are selected mainly because the graph samples in the chosen anomaly class meet some key semantics of anomalies, *e.g.*, graphs are scatteredly or sparsely distributed in the representation space.

5.2 Competing Methods

Seven competing methods from two types of approach are used.

Two-step Methods. This approach first uses state-of-the-art graph representation-based methods to obtain vectorized graph representations, and then applies advanced off-the-shelf shallow anomaly detectors on top of the representations to calculate anomaly scores. InfoGraph [38], WL [36] and PK graph kernels [23] are used in our experiments. Anomaly detectors, including iForest [20] and kNN ensemble (LESINN) [30], are utilized. The combination of these embedding methods and detectors leads to six two-step methods.
End-to-end Methods. We also compare with the one-class GCN-based method, namely OCGCN [50], which can be trained in an end-to-end manner as GLocalKD. OCGCN is optimized using a SVDD objective on top of GCN-based representation learning.

5.3 Implementation and Evaluation

The target network and the predictor network in GLocalKD share the same network architecture – a network with three GCN layers. The dimension of the hidden layer is 512 and the output layer has 256 neural units. The learning rate is selected through the grid search, varying from 10^{-1} to 10^{-5}. The batch size is 300 for all data sets except the four largest datasets HSE, MMP, p53 and PPAR-gamma, for which the batch size is 2000. For the competing methods, the network architecture and the optimization of OCGCN is the same as our model. The other methods are taken from their authors. We probed a wide range of hyperparameter settings in both iForest and LESINN. We found that the performance of iForest does not change much with varying hyperparameter settings, while LESINN can obtain large improvement of using one subsampling size setting over the others (see Table 4 in Appendix E). Due to these observations, iForest with subsampling size and the number of trees respectively set to 256 and 100 is used by default, while LESINN with the subsampling size setting that performs best on most of the datasets is used. More implementation details can be found in Appendix B.

In terms of evaluation, we use the popular anomaly detection evaluation metric – Area Under Receiver Operating Characteristic Curve (AUC). Higher AUC indicates better performance. We report the mean AUC and standard deviation based on 5-fold cross-validation for all datasets, except HSE, MMP, p53 and PPAR-gamma that have widely-used training and test splits. For these four datasets, the results are based on five runs with different random seeds.

5.4 Comparison to State-of-the-art Methods

The AUC results of GLocalKD and its seven competing methods are reported in Table 1. Our GLocalKD model is the best performer on 7 datasets, achieving improvement ranging from 1% to 12% on many of these datasets compared to the best contenders per dataset, *e.g.*, AIDS (3.7%), PROTEINS_full (6.7%), PPAR-gamma (10.3%), MMP (10.5%), p53 (11.9%); and its performance is very close to the best contenders on some other datasets, such as DD and COLLAB. The consistent superiority of GLocalKD is mainly due to its capability in learning both global and local graph regularity. Its performance may drop significantly, *e.g.*, decrease to performance equivalent to a random detector, if only one of these patterns is captured (see Table 2). The seven competing methods fail to work in many datasets mainly because their graph representations capture only partial local/global pattern information.

We also perform a paired Wilcoxon signed rank test [42] to examine the significance of GLocalKD against each of the competing methods across the 16 datasets. As shown by the p-values in Table 1, GLocalKD significantly outperforms the iForest-based methods and OCGCN at the 99% confidence level. The confidence level of the superiority of GLocalKD over LESINN-based methods ranges from 85% and 95%. However, note that LESINN heavily relies on its subsample size (see Table 4 for the full results of InfoGraph-LESINN, WL-LESINN and PK-LESINN in Appendix E). GLocalKD works less effectively on COLLAB than some contenders, which may be due to the inseparability of anomalies from the normal graphs as the contenders also do not perform well on it.

In terms of computational efficiency, as shown by the results in Table 3 in Appendix C, GLocalKD and OCGCN have a similar time complexity and run much faster than the other methods in online detection, since iForest/LESINN methods require extra steps on top of the graph representations to compute the anomaly scores. On the other hand, GLocalKD and OCGCN are generally more computationally costly than the WL and PK based methods because GLocalKD and OCGCN typically require multiple iterations to perform well.

5.5 Sample Efficiency

5.5.1 Experiment Settings. This section examines the performance of our model w.r.t. the amount of training data, *i.e.*, sample efficiency, using the deep competing method OCGCN as baseline. We use respective 5%, 25%, 50%, 75% and 100% of original training samples to train the models, and evaluate the performance on the same test data set. We report the results on the attributed graph datasets only. Similar results can be found on the other datasets.

5.5.2 Findings. The AUC results are shown in Figure 4. It is very impressive that even when 95% less training data are used, GLocalKD can retain similarly good performance across nearly all the
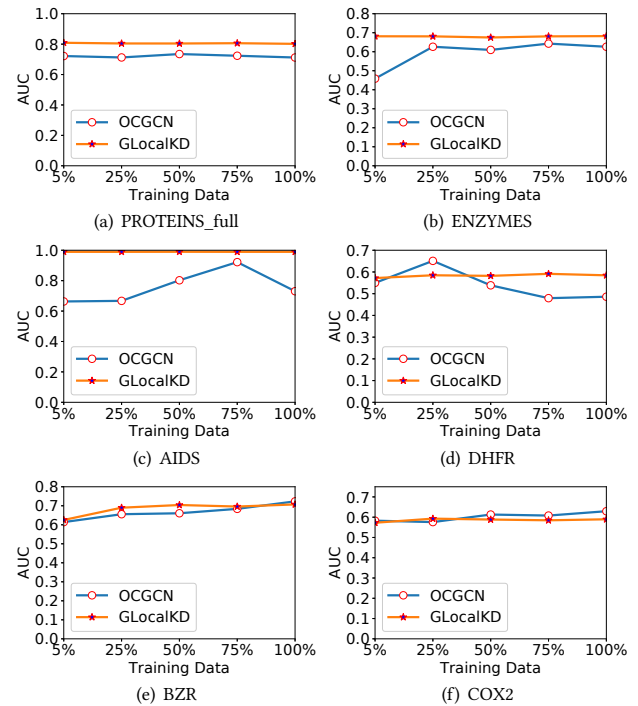
Figure 4: AUC performance of GLocalKD and OCGCN using different amount of training data.

six datasets. By contrast, the performance of OCGCN can drop significantly on some datasets, such as ENZYMES and AIDS, if the same amount of training data is reduced. As a result, GLocalKD can outperform OCGCN by large margins even it uses 95% less training data than OCGCN on such datasets.

5.6 Robustness w.r.t. Anomaly Contamination

5.6.1 Experiment Settings. Recall that we tackle the semi-supervised anomaly detection setting with exclusively normal training samples. However, the data collected in real applications may be contaminated by some anomalies or data noises. This section investigates the robustness of GLocalKD w.r.t. different anomaly contamination levels in the training data. We vary the contamination rates from 0% up to 16%. Again, we report the results on the six attributed graph datasets only due to page limitation; OCGCN is used as baseline.

5.6.2 Findings. AUC results of GLocalKD and OCGCN with different anomaly contamination rates are shown in Figure 5. GLocalKD is barely affected by the contamination and performs very stably on all the datasets, contrasting to OCGCN whose performance decreases largely with increasing contamination rate on ENZYMES and AIDS. This is mainly because GLocalKD essentially learns all types of patterns in the training data by the random distillation, by which it is able to detect the anomalies as long as those anomalous patterns are not as frequent as the normal patterns in the training data; whereas OCGCN is sensitive since its anomaly measure, SVDD, is sensitive to the anomaly contamination.
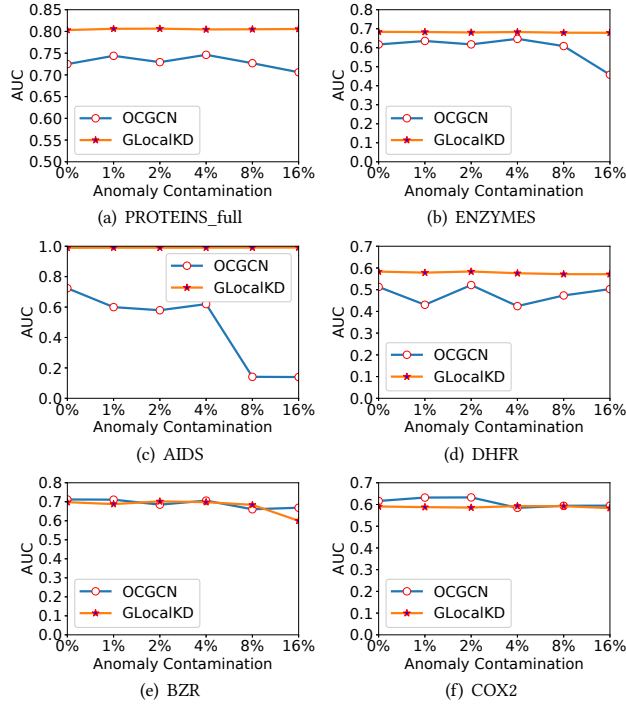
Figure 5: AUC performance of GLocalKD and OCGCN w.r.t. different anomaly contamination rates.

5.7 Sensitivity Test

5.7.1 Experiment Settings. This section tests the sensitivity of GLocalKD to the representation dimension and the GCN depth. For the first test, we vary the output dimension of GCN in {32, 64, 128, 256, 512}; for the GCN depth, we evaluate the performance of GLocalKD using k GCN layers, with $k \in \{1, 2, 3, 5\}$. The results are illustrated in Figures 6 and 7 in Appendix D.

5.7.2 Sensitivity. As can be seen from the results, GLocalKD performs stably using different representation dimensionality sizes on most datasets. The dimensionality size – 256 – is generally recommended as this setting enables GLocalKD to perform well on diverse datasets.

Besides, GLocalKD achieves better performance with increasing depth on nearly all the datasets, but the performance is flatten when increasing the depth from three to five. A network depth of three is generally recommended, since deeper GCN does not help achieve better performance but is more computationally costly.

5.8 Ablation Study

5.8.1 Experiment Settings. In this section, we examine the importance of the two components, L_{graph} and L_{node}, in our model. To do that, we derive two variants of GLocalKD, including GLocalKD w/o L_{node} that denotes the use of random distillation on the graph representations only, and GLocalKD w/o L_{graph} that represents the use of random distillation on the node representations only.

5.8.2 Findings. The results of GLocalKD and its two variants are shown in Table 2. It is clear that using L_{graph} (or L_{node}) only can obtain better performance on some datasets, while it may perform worse on the other datasets, compared with GLocalKD. Joint random distillation by using both of L_{graph} and L_{node} can achieve a good trade-off and perform generally good across all the datasets.

It is interesting that GLocalKD w/o L_{graph} significantly outperforms GLocalKD w/o L_{node} in a number of datasets, *e.g.*, AIDS, DHFR, DD, MMP, p53, PPAR-gamma and hERG, indicating the dominant presence of locally-anomalous graphs in those data; on the other hand, the inverse cases occur on ENZYMES, IMDB and HSE, indicating the dominance of globally-anomalous graphs in these three datasets. These results show that modeling fine-grained graph regularity is as important as, if not more important than, the holistic graph regularity for the GAD task, since both types of graph anomalies can present in the graph datasets.

Table 2: Detection of locally/globally-anomalous graphs.

Dataset	GLocalKD	w/o L_{node}	w/o L_{graph}
PROTEINS_full	**0.785**±0.034	0.686±0.045	0.757±0.040
ENZYMES	0.636±0.061	**0.642**±0.096	0.505±0.036
AIDS	0.992±0.004	0.963±0.014	**0.997**±0.006
DHFR	0.558±0.030	0.459±0.036	**0.596**±0.030
BZR	**0.679**±0.065	0.623±0.079	0.671±0.049
COX2	**0.589**±0.045	0.585±0.051	0.557±0.055
DD	0.805±0.017	0.528±0.093	**0.805**±0.017
NCI1	**0.683**±0.015	0.458±0.058	0.682±0.015
IMDB	0.514±0.039	**0.610**±0.103	0.490±0.044
REDDIT	**0.782**±0.016	0.574±0.085	0.781±0.016
HSE	0.591±0.001	**0.655**±0.007	0.589±0.000
MMP	0.676±0.001	0.543±0.016	**0.680**±0.000
p53	0.639±0.002	0.495±0.016	**0.641**±0.000
PPAR-gamma	0.644±0.001	0.600±0.044	**0.644**±0.000
COLLAB	0.525±0.014	0.501±0.055	**0.526**±0.012
hERG	**0.704**±0.049	0.566±0.043	0.703±0.057

6 CONCLUSION

This paper proposes a novel framework and its instantiation GLocalKD to detect abnormal graphs in a set of graphs. As shown in our experimental results, graph datasets can contain different types of anomalies – locally- and globally-anomalous graphs. To the best of our knowledge, GLocalKD is the first model designed to detect both types of graph anomalies. Extensive experiments demonstrate that GLocalKD performs significantly better in AUC and can be trained much more sample-efficiently when compared with its advanced counterparts. We also show that GLocalKD achieves promising AUC performance even when there is large anomaly contamination in the training data, indicating that GLocalKD can be applied in not only semi-supervised settings (exclusively normal training data) but also unsupervised settings (anomaly-contaminated unlabeled training data).

REFERENCES

[1] Charu C Aggarwal and Haixun Wang. 2010. Graph data management and mining: a survey of algorithms and applications. In *Managing and Mining Graph Data*. Springer, 13–68.

[2] Leman Akoglu, Hanghang Tong, and Danai Koutra. 2015. Graph based anomaly detection and description: a survey. *Data Mining and Knowledge Discovery* 29, 3 (2015), 626–688.

[3] Paul Bergmann, Michael Fauser, David Sattlegger, and Carsten Steger. 2020. Uninformed students: student-teacher anomaly detection with discriminative latent embeddings. In *CVPR*. 4183–4192.

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

[5] Yuri Burda, Harrison Edwards, Amos Storkey, and Oleg Klimov. 2018. Exploration by random network distillation. *arXiv preprint arXiv:1810.12894* (2018).

[6] Guilherme O Campos, Arthur Zimek, Jörg Sander, Ricardo JGB Campello, Barbora Micenková, Erich Schubert, Ira Assent, and Michael E Houle. 2016. On the evaluation of unsupervised outlier detection: measures, datasets, and an empirical study. *Data Mining and Knowledge Discovery* 30, 4 (2016), 891–927.

[7] Jinghui Chen, Saket Sathe, Charu Aggarwal, and Deepak Turaga. 2017. Outlier detection with autoencoder ensembles. In *SDM*. SIAM, 90–98.

[8] Kaize Ding, Jundong Li, Nitin Agarwal, and Huan Liu. 2020. Inductive anomaly detection on attributed networks. In *IJCAI*. 1288–1294.

[9] Dhivya Eswaran, Christos Faloutsos, Sudipto Guha, and Nina Mishra. 2018. Spotlight: detecting anomalies in streaming graphs. In *KDD*. 1378–1386.

[10] Mariana-Iuliana Georgescu, Antonio Barbalau, Radu Tudor Ionescu, Fahad Shahbaz Khan, Marius Popescu, and Mubarak Shah. 2021. Anomaly detection in video via self-supervised and multi-task learning. In *CVPR*. 12742–12752.

[11] Simon Hawkins, Hongxing He, Graham Williams, and Rohan Baxter. 2002. Outlier detection using replicator neural networks. In *DaWaK*. Springer, 170–180.

[12] Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. 2015. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531* (2015).

[13] Jianguo Jiang, Jiuming Chen, Tianbo Gu, Kim-Kwang Raymond Choo, Chao Liu, Min Yu, Weiqing Huang, and Prasant Mohapatra. 2019. Anomaly detection with graph convolutional networks for insider threat and fraud detection. In *MILCOM*. IEEE, 109–114.

[14] Kristian Kersting, Nils M. Kriege, Christopher Morris, Petra Mutzel, and Marion Neumann. 2016. Benchmark Data Sets for Graph Kernels. http://graphkernels.cs.tu-dortmund.de

[15] Thomas N Kipf and Max Welling. 2016. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907* (2016).

[16] Atsutoshi Kumagai, Tomoharu Iwata, and Yasuhiro Fujiwara. 2020. Semi-supervised anomaly detection on attributed graphs. *arXiv preprint arXiv:2002.12011* (2020).

[17] Sofiane Lagraa, Karima Amrouche, Hamida Seba, et al. 2021. A simple graph embedding for anomaly detection in a stream of heterogeneous labeled graphs. *Pattern Recognition* 112 (2021), 107746.

[18] Chun Yen Lee and Yi-Ping Phoebe Chen. 2021. Descriptive prediction of drug side-effects using a hybrid deep learning model. *International Journal of Intelligent Systems* 36, 6 (2021), 2491–2510.

[19] Haoliang Li, Shiqi Wang, Peisong He, and Anderson Rocha. 2020. Face anti-spoofing with deep neural network distillation. *IEEE Journal of Selected Topics in Signal Processing* 14, 5 (2020), 933–946.

[20] Fei Tony Liu, Kai Ming Ting, and Zhi-Hua Zhou. 2008. Isolation forest. In *ICDM*. IEEE, 413–422.

[21] Emaad Manzoor, Sadegh M Milajerdi, and Leman Akoglu. 2016. Fast memory-efficient anomaly detection in streaming heterogeneous graphs. In *KDD*. 1035–1044.

[22] Annamalai Narayanan, Mahinthan Chandramohan, Rajasekar Venkatesan, Lihui Chen, Yang Liu, and Shantanu Jaiswal. 2017. Graph2vec: learning distributed representations of graphs. *arXiv preprint arXiv:1707.05005* (2017).

[23] Marion Neumann, Roman Garnett, Christian Bauckhage, and Kristian Kersting. 2016. Propagation kernels: efficient graph kernels from propagated information. *Machine Learning* 102, 2 (2016), 209–245.

[24] Phuc Cuong Ngo, Amadeus Aristo Winarto, Connie Khor Li Kou, Sojeong Park, Farhan Akram, and Hwee Kuan Lee. 2019. Fence GAN: towards better anomaly detection. In *ICTAI*. IEEE, 141–148.

[25] Hung T Nguyen, Pierre J Liang, and Leman Akoglu. 2020. Anomaly detection in large labeled multi-graph databases. *arXiv preprint arXiv:2010.03600* (2020).

[26] Ian Osband, John Aslanides, and Albin Cassirer. 2018. Randomized prior functions for deep reinforcement learning. *arXiv preprint arXiv:1806.03335* (2018).

[27] Guansong Pang, Longbing Cao, Ling Chen, and Huan Liu. 2018. Learning representations of ultrahigh-dimensional data for random distance-based outlier

[28] Guansong Pang, Chunhua Shen, Longbing Cao, and Anton Van Den Hengel. 2021. Deep learning for anomaly detection: a review. *ACM Computing Surveys (CSUR)* 54, 2 (2021), 1–38.

[29] Guansong Pang, Chunhua Shen, and Anton van den Hengel. 2019. Deep anomaly detection with deviation networks. In *KDD*. 353–362.

[30] Guansong Pang, Kai Ming Ting, and David Albrecht. 2015. LeSiNN: detecting anomalies by identifying least similar nearest neighbours. In *2015 ICDMW*. IEEE, 623–630.

[31] Pramuditha Perera and Vishal M Patel. 2019. Learning deep features for one-class classification. *IEEE Transactions on Image Processing* 28, 11 (2019), 5450–5463.

[32] 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.

[33] Mohammadreza Salehi, Niousha Sadjadi, Soroosh Baselizadeh, Mohammad Hossein Rohban, and Hamid R Rabiee. 2020. Multiresolution knowledge distillation for anomaly detection. *arXiv preprint arXiv:2011.11108* (2020).

[34] Thomas Schlegl, Philipp Seeböck, Sebastian M Waldstein, Georg Langs, and Ursula Schmidt-Erfurth. 2019. f-AnoGAN: fast unsupervised anomaly detection with generative adversarial networks. *Medical Image Analysis* 54 (2019), 30–44.

[35] Bernhard Schölkopf, Robert Williamson, Alex Smola, John Shawe-Taylor, and John Platt. 1999. Support vector method for novelty detection. In *NIPS*, Vol. 12. 582–588.

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

[37] Kumar Sricharan and Kamalika Das. 2014. Localizing anomalous changes in time-evolving graphs. In *ACM SIGMOD*. 1347–1358.

[38] Fan-Yun Sun, Jordan Hoffmann, Vikas Verma, and Jian Tang. 2019. InfoGraph: unsupervised and semi-supervised graph-Level representation learning via mutual information maximization. In *ICLR*.

[39] A Varfis, E Kotsakis, A Tsois, AV Donati, M Sjachyn, E Camossi, P Villa, T Dimitrova, and M Pellissier. 2011. ConTraffic: maritime container traffic anomaly detection. In *MAD 2011 Workshop Proceedings*. Citeseer, 113.

[40] Hu Wang, Guansong Pang, Chunhua Shen, and Congbo Ma. 2020. Unsupervised Representation Learning by Predicting Random Distances. In *IJCAI*.

[41] Xuhong Wang, Baihong Jin, Ying Du, Ping Cui, Yingshui Tan, and Yupu Yang. 2021. One-class graph neural networks for anomaly detection in attributed networks. *Neural Computing and Applications* (2021), 1–13.

[42] RF Woolson. 2007. Wilcoxon signed-rank test. *Wiley Encyclopedia of Clinical Trials* (2007), 1–3.

[43] Zonghan Wu, Shirui Pan, Fengwen Chen, Guodong Long, Chengqi Zhang, and S Yu Philip. 2020. A comprehensive survey on graph neural networks. *IEEE Transactions on Neural Networks and Learning Systems* (2020).

[44] Qinfeng Xiao, Jing Wang, Youfang Lin, Wenbo Gongsa, Ganghui Hu, Menggang Li, and Fang Wang. 2021. Unsupervised anomaly detection with distillated teacher-student network ensemble. *Entropy* 23, 2 (2021), 201.

[45] Keyulu Xu, Weihua Hu, Jure Leskovec, and Stefanie Jegelka. 2018. How powerful are graph neural networks? *arXiv preprint arXiv:1810.00826* (2018).

[46] Minji Yoon, Bryan Hooi, Kijung Shin, and Christos Faloutsos. 2019. Fast and accurate anomaly detection in dynamic graphs with a two-pronged approach. In *KDD*. 647–657.

[47] Wenchao Yu, Wei Cheng, Charu C Aggarwal, Kai Zhang, Haifeng Chen, and Wei Wang. 2018. Netwalk: a flexible deep embedding approach for anomaly detection in dynamic networks. In *KDD*. 2672–2681.

[48] Muhan Zhang, Zhicheng Cui, Marion Neumann, and Yixin Chen. 2018. An end-to-end deep learning architecture for graph classification. In *AAAI*, Vol. 32.

[49] Ziwei Zhang, Peng Cui, and Wenwu Zhu. 2020. Deep learning on graphs: a survey. *IEEE Transactions on Knowledge and Data Engineering* (2020).

[50] Lingxiao Zhao and Leman Akoglu. 2020. On using classification datasets to evaluate graph outlier detection: peculiar observations and new insights. *arXiv preprint arXiv:2012.12931* (2020).

[51] Tong Zhao, Chuchen Deng, Kaifeng Yu, Tianwen Jiang, Daheng Wang, and Meng Jiang. 2020. Error-Bounded Graph Anomaly Loss for GNNs. In *CIKM*. 1873–1882.

[52] Li Zheng, Zhenpeng Li, Jian Li, Zhao Li, and Jun Gao. 2019. AddGraph: anomaly detection in dynamic graph using attention-based temporal GCN. In *IJCAI*. 4419–4425.

[53] Panpan Zheng, Shuhan Yuan, Xintao Wu, Jun Li, and Aidong Lu. 2019. One-class adversarial nets for fraud detection. In *AAAI*, Vol. 33. 1286–1293.

[54] Chong Zhou and Randy C Paffenroth. 2017. Anomaly detection with robust deep autoencoders. In *KDD*. 665–674.

APPENDIX

In the appendix, we provide more detailed information about the implementation details of our model GLocalKD and its competing methods, as well as some extra empirical results. The algorithmic procedure of GLocalKD is presented in Appendix A, while the GNN architecture, hyperparameter settings and optimization are presented in Appendix B. Appendix C presents the training and test time of GLocalKD and its competitors on three representative datasets. Appendix D shows the sensitivity test results. Lastly, Appendix E presents the influence of subsample size on the performance of LESINN.

A THE ALGORITHM OF GLOCALKD

Algorithm 1 presents the procedure of training GLocalKD. After random weight initialization of $\hat{\Theta}$ and Θ in Step 1, GLocalKD performs stochastic gradient descent-based optimization to learn Θ of the predictor network in Steps 2-11, while the parameters in $\hat{\Theta}$ are fixed. Particularly, Step 4 samples a mini-batch $\mathcal{B}$ with size $batch\_size$. We obtain node representations and graph representations from both of $\hat{\phi}(\cdot, \hat{\Theta})$ and $\phi(\cdot, \Theta)$ in Steps 6-7, respectively. Step 9 then performs gradient descent steps on our loss Eq. 12 w.r.t. the parameters in Θ. We finally obtain the predictor network $\phi(\cdot, \Theta^*)$ with the learned Θ^* and the random target network $\hat{\phi}(\cdot, \hat{\Theta})$.

Algorithm 1 Training GLocalKD

Input: Normal training graph set $\mathcal{G} = \{G_i\}_i$
Output: Target network – $\hat{\phi}(\cdot, \hat{\Theta})$, predictor network – $\phi(\cdot, \Theta^*)$
1: Randomly initialize $\hat{\Theta}$ and Θ, with $\hat{\Theta}$ fixed
2: **for** $i = 1$ to n_epochs **do**
3:   **for** $j = 1$ to $n\_batches$ **do**
4:     $\mathcal{B}$ ←Randomly sample $batch\_size$ graphs from $\mathcal{G}$
5:     **for** G in $\mathcal{B}$ **do**
6:       Compute node representations $\hat{\mathbf{h}}_i$ and $\mathbf{h}_i$, $\forall v_i \in \mathcal{V}_G$
7:       Compute graph representations $\hat{\mathbf{h}}_G$ and $\mathbf{h}_G$
8:     **end for**
9:     Perform a gradient descent step on Eq. 12 w.r.t. the parameters in Θ
10:   **end for**
11: **end for**
12: **return** $\hat{\phi}(\cdot, \hat{\Theta})$, $\phi(\cdot, \Theta^*)$

B IMPLEMENTATION DETAILS

All experiments are carried out on NVIDIA Quadro RTX 6000 GPU with Intel Xeon E-2288G 3.7GHz CPU, and all models are implemented with Python 3.6[2].

The target network and the predictor network in GLocalKD share the same network architecture – a network with three GCN layers. The dimension of the hidden layer is 512 and the output layer has 256 neural units. As indicated in Eq. 9, max pooling is used to obtain the graph representations. The GCN weight parameters are initialized using the Kaiming uniform method, with the bias parameters initialized to be zeros. For attributed graph datasets, the feature

---
[2]https://www.python.org/

matrix $\mathbf{X}$ is directly built upon their node features; for datasets with plain graphs, the degree of each node is used as the node features. The learning rate is set to 10^{-4} by default except on the ENZYMES, AIDS, DHFR, HSE, p53, MMP and PPAR-gamma datasets where the learning rate is set to 10^{-5}. Nevertheless, the performance of GLocalKD has small variations on these datasets either using 10^{-4} or 10^{-5} as the learning rate. On the dataset PROTEINS_full, the learning rate is set to 10^{-2} to obtain good performance. The batch size is 300 for all data sets except the four largest datasets HSE, MMP, p53 and PPAR-gamma, for which the batch size is 2000. The number of epochs is 150 for all data sets.

Table 3: Training and test time on three datasets: REDDIT, p53 and COLLAB.

	Dataset	InfoGraph		WL		PK		OCGCN	GLocalKD
		iForest	LESINN	iForest	LESINN	iForest	LESINN		
Training Time	REDDIT	3536.36	3536.36	8.01	8.01	127.09	127.09	1397.40	1395
	p53	60.61	60.61	9.42	9.42	821.29	821.29	297.37	337.80
	COLLAB	2059.66	2059.66	63.24	63.24	416.95	416.95	2421.83	2510.52
Test Time	REDDIT	5.79	15.19	3.72	29.37	84.46	112.20	4.65	4.97
	p53	19.67	24.54	225.44	207.95	301.89	250.79	0.66	0.97
	COLLAB	12.41	34.44	39.50	313.65	273.82	573.90	9.28	8.88

The architecture of GCN used in OCGCN is exactly the same as our model. The learning rate is also searched from 10^{-1} to 10^{-5}. We use the same method in Deep SVDD [32] to generate the one-class center. Both of GLocalKD and OCGCN are implemented using Pytorch 1.9[3]. Similarly, InfoGraph is taken from its official implementation[4], which uses a three-layer GIN architecture, with the learning rate set to 10^{-3}. Adam is the default optimizer used in the above three methods. Both WL and PK are directly taken from the GraKeL library 0.1.8[5]. For WL, we perform three iterations to obtain the graph representations, which utilize the same neighborhood information as a three-layer GCN as in our model and OCGCN. PK is used with the recommended setting in GraKeL. For iForest[6] in each method, we adjust its parameters, including the number of trees, subsample size and contamination rate. We finally use the recommended settings as in [20], *i.e.* 100 for the number of trees, 256 for subsampling size and 0.1 for contamination rate, since the results of iForest do not change much with varying hyperparameter settings. Two parameters in LESINN[7], *i.e.*, ensemble size and subsample size, are chosen from {2, 4, 8, 16, 32, 64, 128, 256}. In Table 1, we report the results of LESINN by setting both of these two parameters to 256, as this setting enables the most effective results across all datasets (see Table 4).

C TRAINING AND TEST TIME

Table 3 is the training and test time of each method on 3 datasets: REDDIT, p53 and COLLAB. REDDIT and COLLAB are the datasets with the largest average number of nodes and edges in all 16 datasets, respectively. The dataset p53 contains the largest number of graphs. The training time of the two-stage methods considers only the graph representations/embeddings learning time.

---
[3]https://pytorch.org/
[4]https://github.com/fanyun-sun/InfoGraph
[5]https://github.com/ysig/GraKeL
[6]https://github.com/scikit-learn/
[7]https://github.com/GuansongPang/deep-outlier-detection/

Table 4: Results of LESINN using different subsampling sizes.

| Dataset	Method	subsample size=2	subsample size=4	subsample size=8	subsample size=16	subsample size=32	subsample size=64	subsample size=128	subsample size=256
PROTEINS_full	Info-LESINN	0.399±0.048	0.398±0.049	0.390±0.049	0.380±0.049	0.367±0.047	0.357±0.047	0.345±0.048	0.336±0.047
	WL-LESINN	0.769±0.017	0.779±0.014	0.780±0.014	0.779±0.016	0.777±0.021	0.769±0.029	0.742±0.047	0.712±0.053
	PK-LESINN	0.759±0.023	0.766±0.024	0.769±0.020	0.769±0.018	0.765±0.019	0.702±0.068	0.633±0.057	0.572±0.031
ENZYMES	Info-LESINN	0.465±0.078	0.465±0.072	0.462±0.059	0.465±0.051	0.466±0.042	0.477±0.041	0.496±0.044	0.528±0.046
	WL-LESINN	0.519±0.066	0.488±0.069	0.485±0.065	0.498±0.050	0.518±0.040	0.538±0.037	0.577±0.044	0.624±0.050
	PK-LESINN	0.562±0.053	0.553±0.043	0.564±0.038	0.579±0.025	0.590±0.034	0.596±0.043	0.594±0.038	0.608±0.033
AIDS	Info-LESINN	0.883±0.041	0.889±0.039	0.900±0.038	0.912±0.035	0.924±0.033	0.935±0.030	0.944±0.027	0.955±0.023
	WL-LESINN	0.651±0.016	0.526±0.021	0.437±0.019	0.424±0.012	0.447±0.016	0.483±0.013	0.528±0.014	0.584±0.016
	PK-LESINN	0.578±0.026	0.468±0.037	0.385±0.021	0.352±0.017	0.358±0.010	0.374±0.009	0.393±0.010	0.421±0.010
DHFR	Info-LESINN	0.460±0.042	0.473±0.046	0.486±0.040	0.509±0.042	0.541±0.039	0.575±0.035	0.608±0.034	0.625±0.028
	WL-LESINN	0.365±0.038	0.401±0.047	0.457±0.049	0.480±0.048	0.509±0.052	0.538±0.055	0.573±0.059	0.596±0.056
	PK-LESINN	0.368±0.032	0.400±0.027	0.431±0.021	0.453±0.027	0.474±0.040	0.503±0.051	0.541±0.056	0.568±0.054
BZR	Info-LESINN	0.557±0.043	0.568±0.037	0.600±0.039	0.632±0.039	0.658±0.040	0.690±0.050	0.721±0.068	0.737±0.071
	WL-LESINN	0.540±0.054	0.549±0.050	0.576±0.061	0.620±0.055	0.679±0.053	0.700±0.050	0.717±0.043	0.720±0.032
	PK-LESINN	0.528±0.070	0.542±0.067	0.578±0.075	0.631±0.073	0.693±0.072	0.739±0.068	0.764±0.066	0.775±0.063
COX2	Info-LESINN	0.588±0.064	0.611±0.050	0.616±0.052	0.628±0.058	0.639±0.066	0.661±0.069	0.673±0.066	0.670±0.079
	WL-LESINN	0.444±0.101	0.487±0.089	0.512±0.074	0.557±0.075	0.583±0.073	0.599±0.074	0.605±0.067	0.590±0.056
	PK-LESINN	0.443±0.093	0.465±0.085	0.472±0.080	0.523±0.073	0.568±0.067	0.608±0.061	0.648±0.046	0.671±0.039
DD	Info-LESINN	0.320±0.038	0.318±0.033	0.315±0.032	0.313±0.031	0.310±0.032	0.308±0.032	0.307±0.032	0.310±0.034
	WL-LESINN	0.543±0.052	0.540±0.048	0.535±0.050	0.547±0.055	0.560±0.054	0.578±0.055	0.605±0.051	0.638±0.045
	PK-LESINN	0.800±0.023	0.811±0.028	0.816±0.028	0.819±0.027	0.822±0.026	0.827±0.027	0.831±0.025	0.833±0.023
NCI1	Info-LESINN	0.479±0.016	0.482±0.018	0.487±0.019	0.495±0.023	0.508±0.027	0.532±0.031	0.561±0.034	0.598±0.035
	WL-LESINN	0.533±0.029	0.566±0.029	0.590±0.024	0.621±0.019	0.650±0.015	0.676±0.014	0.710±0.014	0.743±0.015
	PK-LESINN	0.525±0.021	0.542±0.024	0.558±0.019	0.586±0.019	0.607±0.017	0.624±0.015	0.646±0.013	0.670±0.012
IMDB	Info-LESINN	0.431±0.033	0.438±0.033	0.441±0.029	0.467±0.045	0.482±0.043	0.505±0.037	0.541±0.023	0.565±0.017
	WL-LESINN	0.398±0.040	0.397±0.028	0.404±0.028	0.437±0.027	0.504±0.055	0.586±0.058	0.605±0.057	0.612±0.046
	PK-LESINN	0.392±0.045	0.384±0.037	0.385±0.033	0.406±0.023	0.462±0.045	0.552±0.057	0.582±0.050	0.585±0.047
REDDIT	Info-LESINN	0.449±0.023	0.461±0.030	0.418±0.038	0.346±0.048	0.290±0.032	0.276±0.028	0.268±0.027	0.262±0.027
	WL-LESINN	0.231±0.026	0.234±0.026	0.237±0.027	0.239±0.027	0.239±0.027	0.239±0.028	0.239±0.028	0.239±0.028
	PK-LESINN	0.224±0.024	0.295±0.035	0.422±0.017	0.440±0.013	0.448±0.013	0.457±0.010	0.471±0.010	0.487±0.013
HSE	Info-LESINN	0.586±0.116	0.589±0.107	0.596±0.100	0.606±0.092	0.617±0.083	0.629±0.071	0.644±0.060	0.657±0.051
	WL-LESINN	0.341±0.000	0.421±0.000	0.468±0.000	0.482±0.000	0.495±0.000	0.507±0.000	0.518±0.000	0.528±0.000
	PK-LESINN	0.361±0.005	0.393±0.011	0.407±0.011	0.419±0.006	0.435±0.004	0.446±0.008	0.462±0.013	0.469±0.016
MMP	Info-LESINN	0.626±0.051	0.612±0.051	0.600±0.048	0.587±0.043	0.579±0.039	0.574±0.038	0.571±0.038	0.571±0.037
	WL-LESINN	0.422±0.000	0.363±0.000	0.344±0.000	0.333±0.000	0.330±0.000	0.320±0.000	0.313±0.000	0.307±0.000
	PK-LESINN	0.400±0.010	0.362±0.002	0.354±0.004	0.348±0.004	0.341±0.006	0.332±0.005	0.326±0.007	0.322±0.008
p53	Info-LESINN	0.573±0.046	0.567±0.045	0.551±0.041	0.537±0.037	0.532±0.033	0.525±0.030	0.520±0.028	0.520±0.025
	WL-LESINN	0.435±0.000	0.429±0.000	0.413±0.000	0.413±0.000	0.409±0.000	0.403±0.000	0.396±0.000	0.390±0.000
	PK-LESINN	0.341±0.007	0.342±0.005	0.340±0.004	0.339±0.003	0.339±0.004	0.336±0.003	0.332±0.001	0.329±0.001
PPAR-gamma	Info-LESINN	0.629±0.026	0.625±0.038	0.616±0.042	0.605±0.044	0.594±0.049	0.574±0.049	0.553±0.043	0.541±0.036
	WL-LESINN	0.379±0.000	0.409±0.000	0.428±0.000	0.444±0.000	0.460±0.000	0.461±0.000	0.458±0.000	0.461±0.000
	PK-LESINN	0.408±0.005	0.405±0.006	0.404±0.007	0.400±0.012	0.400±0.014	0.397±0.016	0.388±0.015	0.388±0.015
COLLAB	Info-LESINN	0.286±0.048	0.272±0.038	0.264±0.032	0.260±0.026	0.255±0.023	0.255±0.024	0.275±0.029	0.319±0.033
	WL-LESINN	0.603±0.029	0.587±0.026	0.535±0.029	0.450±0.030	0.373±0.025	0.365±0.019	0.445±0.018	0.536±0.014
	PK-LESINN	0.621±0.037	0.606±0.031	0.558±0.046	0.474±0.057	0.394±0.054	0.382±0.051	0.472±0.051	0.550±0.043
hERG	Info-LESINN	0.574±0.044	0.601±0.046	0.610±0.050	0.628±0.053	0.641±0.052	0.659±0.051	0.685±0.049	0.701±0.048
	WL-LESINN	0.742±0.035	0.753±0.026	0.764±0.027	0.772±0.031	0.782±0.035	0.795±0.043	0.802±0.048	0.802±0.047
	PK-LESINN	0.762±0.036	0.769±0.037	0.775±0.039	0.779±0.042	0.791±0.043	0.798±0.049	0.800±0.054	0.798±0.052
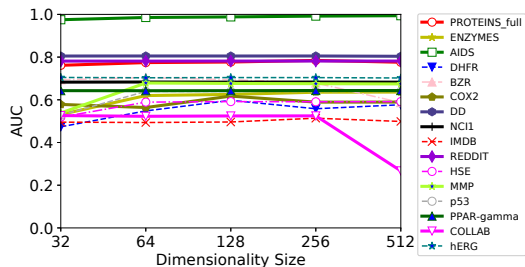
Figure 6: AUC results w.r.t. representation dimensionality.

D SENSITIVITY TEST

Figure 6 and Figure 7 show the performance of GLocalKD w.r.t. the GCN's output dimensionality and depth, respectively.

E DETAILED RESULTS OF LESINN

Table 4 shows the effect of subsample size on the performance of LESINN. We fix the ensemble size to 256 and vary the subsampling size in {2, 4, 8, 16, 32, 64, 128, 256} to obtain the results.
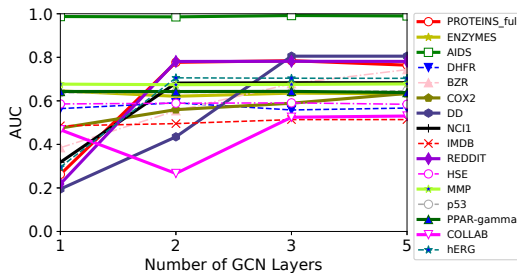
Figure 7: AUC of GLocalKD with different GCN depths.