Graph Anomaly Detection with Unsupervised GNNs

Lingxiao Zhao
Carnegie Mellon University
Pittsburgh, PA, USA
lingxiao@cmu.edu

Saurabh Sawlani
SoundHound
Berlin, Germany
saurabh.sawlani@gmail.com

Arvind Srinivasan
Carnegie Mellon University
Pittsburgh, PA, USA
arvindsr@andrew.cmu.edu

Leman Akoglu
Carnegie Mellon University
Pittsburgh, PA, USA
lakoglu@andrew.cmu.edu

***Abstract*—Graph-based anomaly detection finds numerous applications in the real-world. Thus, there exists extensive literature on the topic that has recently shifted toward deep detection models due to advances in deep learning and graph neural networks (GNNs). A vast majority of prior work focuses on detecting node/edge/subgraph anomalies within a single graph, with much less work on graph-level anomaly detection in a graph database. This work aims to fill two gaps in the literature: We (1) design GLAM, an end-to-end *graph-level* anomaly detection model based on GNNs, and (2) focus on *unsupervised model selection*, which is notoriously hard due to lack of any labels, yet especially critical for deep NN based models with a long list of hyperparameters. Further, we propose a new pooling strategy for graph-level embedding, called MMD-pooling, that is geared toward detecting *distribution* anomalies which has not been considered before. Through extensive experiments on 15 real-world datasets, we show that (i) GLAM outperforms node-level and two-stage (i.e. not end-to-end) baselines, and (ii) model selection picks a significantly more effective model than expectation (i.e. average) –without using any labels– among candidates with otherwise large variation in performance.**

***Index Terms*—graph-level anomaly detection, unsupervised model selection, graph neural networks**

I. Introduction

Given a *collection* of graphs, possibly with weighted edges, and labeled or multi-attributed nodes, how can we identify the anomalous graphs that stand out from the majority? Graph-level anomaly detection, different from detecting anomalies in a *single* graph, aims to discover unusual graphs among *multiple* graphs in a (graph) database. The problem applies to many real-world domains, where each graph may capture a chemical compound [1], human pose [2], cargo shipment [3], system-call [4], command flow [5], information cascade [6], etc.

Graph-based anomaly detection has been studied in the literature [7]. However, majority of the work focuses on (node, edge, subgraph) anomalies within a *single* graph, most often in plain graphs [8], [9], and less often in labeled [10] or attributed graphs [11]. Prior work on graph-level anomaly detection is much sparser, majority of which are traditional substructure mining based techniques [10], [4], [12]. These do not easily generalize to graphs with complex properties; e.g. SpotLight [12] cannot accommodate node labels or attributes, Subdue [10] and StreamSpot [4] cannot handle weighted edges or multi-attributed nodes.

With recent advances in deep learning, focus has shifted toward deep neural network based anomaly detection models (see surveys, [27], [14], [28]). However, we are not aware of any existing work on end-to-end graph-level anomaly detection based on GNNs. A conceptual comparison of related work is given in Table I. In addition, deep methods rely on many hyperparameters (HPs) that influence their performance, such as number of hidden layers/units and epochs, drop-out/weight decay/learning rates, to name a few [29]. Hence model selection is challenging for *unsupervised* anomaly detection, in the absence of any labeled data.

This work fills two gaps in the literature: We (1) design a GNN-based end-to-end model called GLAM to address the graph-level anomaly detection problem, and (2) address the unsupervised model selection task, that is, effectively select the hyperparameters of GLAM *without using any labels*. Further, we specify two different types of ($i.$ point and $ii.$ distribution) graph anomalies, design a novel pooling strategy for the latter.

- **Deep Graph-level Anomaly Detection:** We propose GLAM, a novel Graph-Level Anomaly detection Model based on GNNs. It embeds graphs in two ways: by *mean-pooling* and our newly proposed *MMD-pooling* to detect point and distribution anomalies, respectively. The latter treats each graph as a *set* of its node embeddings, and helps identify complementary anomalies.
- **Unsupervised Anomaly Model Selection:** Besides various advantages, GLAM also inherits a list of hyperparameters that require tuning for effective performance. We systematically address the unsupervised model selection (UMS) problem.
- **Effectiveness:** Through experiments on 15 real-world graph databases, we show that (i) the effectiveness of GLAM against 8 GNN-based (two-stage and node-level) baselines, (ii) the ability of our UMS component to pick a model with superior performance as compared to a model with fixed configuration, and (iii) the contributing factors behind GLAM through various ablation studies.

Our code is available at https://github.com/sawlani/GLAM. For a full version of the paper we refer to arxiv.

II. Problem Definition

We consider anomaly detection in a graph database $\mathcal{G} = \{G_1{=}(V_1, E_1), \ldots, G_N{=}(V_N, E_N)\}$, containing graphs with labeled or attributed nodes, which we define as follows.

Problem 1 (Graph-level anomaly detection (GLAD))**.** *Given an unlabeled graph database $\mathcal{G} = \{G_i = (V_i, E_i)\}_{i=1}^{N}$ containing N unordered, node-labeled or node-attributed graphs; Identify the unusual graphs that differ significantly from the majority of graphs in $\mathcal{G}$.*

Table I
COMPARISON OF RELATED WORK IN TERMS OF DESIRED PROPERTIES FOR GRAPH-LEVEL ANOMALY DETECTION.

Desired Properties	[13], [14] point-cloud	[15], [16], [17], [18] node-level	[19], [20] node-level	[21] node-level	[10] traditional	[4] traditional	[12] traditional	[22], [23] graph embedding	[24] graph embedding	[25], [26] graph embedding	this paper
end-to-end anomaly detection	✓		✓	✓							✓
unsupervised (vs. (semi-)supervised)	✓	✓	✓		✓	✓	✓		✓	✓	✓
graph-structured data (vs. point-cloud)		✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
graph-level detection (vs. node/edge-level)					✓	✓	✓	✓	✓	✓	✓
graph embedding						✓	✓	✓	✓	✓	✓
handle labeled nodes		✓	✓	✓	✓	✓		✓	✓	✓	✓
handle multi-attributed nodes		✓	✓	✓				✓	✓		✓
handle weighted edges		✓	✓	✓			✓	✓	✓		✓
unsupervised model selection											✓

Importantly, GLAD comes bundled with an associated problem: unsupervised model selection. As many models (especially deep NNs) exhibit a list of hyperparameters and the performance is sensitive to the choice of their values [30], it is critical to address GLAD with a built-in UMS solution.

Problem 2 (Unsupervised Model Selection (UMS))**.** *Consider an anomaly detection model $M(\mathbf{\Theta})$ for GLAD, where $\mathbf{\Theta}$ denotes the set of hyperparameters. Given a new unlabeled dataset $\mathcal{G}$, Select an effective model – without using any labels – among candidates $\{M(\Theta_1), M(\Theta_2), \dots\}$ induced by different $\mathbf{\Theta}$ configurations/values.*

III. PROPOSED METHOD: GLAM

We introduce GLAM for graph-level anomaly detection (Problem 1), *built-in* with UMS (Problem 2). GLAM first generates graph-level representations with GNNs using both mean and MMD-pooling (Sec. III-A), on top of which it employs an anomaly detection objective (Sec. III-B), and finally performs UMS (Sec. III-C).

A. Graph-level Representation

The first step of GLAM is "flattening" each graph to a set of node embeddings (i.e. vectors). To this end, we employ the GIN model that was shown to be one of the most expressive among a large class of MPNNs [23]. To model the COMBINE and AGGREGATE functions, GIN employs multi-layer perceptrons (MLPs), with learnable parameters, for their injectiveness property. As MLPs can represent the composition of functions, the update equation of node embeddings at layer l is written as

$$\mathbf{h}_v^{(l)} = \mathsf{MLP}^{(l)}\big((1+\epsilon^{(l)})\cdot \mathbf{h}_v^{(l-1)} + \sum_{u\in\mathcal{N}(v)} \mathbf{h}_u^{(l-1)}\big)\ . \tag{1}$$

Upon node embedding (by L-layer GIN), each graph $G_i \in \mathcal{G}$ can be regarded as a set $\mathcal{S}_i = \{\mathbf{h}_{1,i}\ ,\ \mathbf{h}_{2,i}\ , \dots,\ \mathbf{h}_{n_i,i}\}$ (superscript $^{(L)}$'s dropped) with cardinality $n_i = |V_i|$ where $\mathbf{h}_{v,i} \in \mathsf{R}^{d'}$ is the (vector) embedding of node $v \in V_i$. Then, the graph database can be seen as a set of sets.

In this work, we aim to detect graph-level anomalies of two different types: (See Figure 1 for an illustration.)

- **Point graph anomaly**: defined as a graph that is a *set containing anomalous nodes*, and
- **Distribution graph anomaly**: defined as a graph that is an *anomalous set* of not-necessarily-anomalous nodes.

Correspondingly, we derive two different graph representations: the typical Mean-pooling and the newly-proposed MMD-pooling, described as follows.

1) **Mean-pooling for Point Anomalies:** To detect graphs containing anomalous nodes, we simply use MEAN as the READOUT aggregation function, that is,

$$\mathbf{h}_{G_i} = \frac{1}{n_i}\sum_{l=1}^{n_i} \mathbf{h}_{l,i}\ ,\ \text{ for all } G_i \in \mathcal{G}\ . \tag{2}$$

The intuition is that anomalous nodes would not only have significantly different embeddings, but also affect the embeddings of other nodes in their local neighborhood due to message-passing. Mean-pooling would be effective in capturing sets with such anomalies as average is sensitive to outliers.

2) **MMD-pooling for Distribution Anomalies:** Differently, a distribution anomaly can arise from non-anomalous nodes, rendering mean-pooling ineffective. Solely node-level detection approaches [20], [19] would also fall short for the same reason. Distinctly, GLAM takes into account the distribution information provided by each $\mathcal{S}_i$ to identify such anomalies.

Suppose the samples (i.e. node embeddings) in each $\mathcal{S}_i$ are distributed according to a (unknown) probability distribution $\mathsf{P}_i \in \mathcal{P}$, where $\mathcal{P}$ is the set of all probability distributions on the node embedding space. Given two graphs G_i and G_j, we define their similarity by a distribution kernel $\kappa(\cdot,\cdot)$ on $\mathcal{P}$, i.e. $\kappa : \mathcal{P}\times\mathcal{P} \mapsto \mathsf{R}$, applied to their probability distributions as

$$\kappa(\mathsf{P}_i, \mathsf{P}_j) = \langle \mu_{\mathbb{P}_i}, \mu_{\mathbb{P}_j}\rangle_{\mathcal{H}}\ , \tag{3}$$

which is equal to the inner product between the kernel embeddings of P_i and P_j in RKHS $\mathcal{H}$.

Based on MMD's properties,

$$MMD^2(\mathsf{P},\mathsf{Q}) = \langle \mu_{\mathbb{P}_i}, \mu_{\mathbb{P}_i}\rangle_{\mathcal{H}} + \langle \mu_{\mathbb{P}_j}, \mu_{\mathbb{P}_j}\rangle_{\mathcal{H}} - 2\langle \mu_{\mathbb{P}_i}, \mu_{\mathbb{P}_j}\rangle_{\mathcal{H}} \tag{4}$$

and we can derive $\kappa(\mathsf{P}_i, \mathsf{P}_j)$ as equal to

$$\langle \mu_{\mathbb{P}_i}, \mu_{\mathbb{P}_j}\rangle_{\mathcal{H}} = \mathsf{E}_{\mathbf{h},\mathbf{h}'}[k(\mathbf{h},\mathbf{h}')] = \int\int k(\mathbf{h},\mathbf{h}')d\mathsf{P}_i(\mathbf{h})d\mathsf{P}_j(\mathbf{h}'). \tag{5}$$

Given the (finite) sample sets $\mathcal{S}_i$ and $\mathcal{S}_j$, we can estimate the distribution kernel similarity in Eq. (5) empirically, as

$$\kappa(\widehat{\mathsf{P}}_i, \widehat{\mathsf{P}}_j) = \frac{1}{n_i\cdot n_j}\sum_{u=1}^{n_i}\sum_{v=1}^{n_j} k(\mathbf{h}_{u,i}, \mathbf{h}_{v,j}) \tag{6}$$

where we use the (characteristic) Gaussian kernel for $k(\cdot,\cdot)$.

We refer to Figure 1 for an intuitive comparison of MMD- vs. Mean-pooling, and an understanding of their complementary strengths.

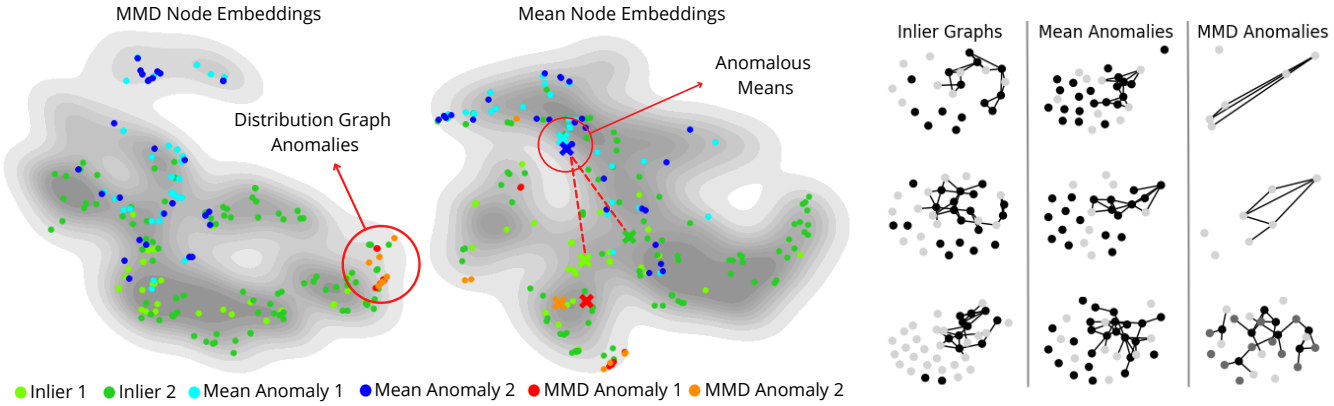

Figure 1. Node embedding space by (left) MMD- and (middle) Mean-pooling in PROTEINS. Heatmap (gray) reflects the background distribution; i.e. density of all node embeddings across $\mathcal{G}$. Symbols w/ same color are nodes of the same graph (see legend). Notice that MMD is complementary to Mean: (*i*) Distribution of node-embeddings (red and orange dots) for MMD-anomalies (on left) differs significantly from background, yet (in middle) Mean-pooling misses them as their means (red&orange crosses) are close to the means of inlier graphs (light/dark green crosses). (*ii*) On the other hand, node-embeddings of Mean-anomalies blend well into background distribution (on left) while (in middle) their means (light/dark blue crosses) are far away from those of inliers (light/dark green crosses). (right) E.g. inliers, and top Mean- and MMD-anomalous graphs. MMD anomalies have distinct distributions, w.r.t. graph structure and/or node labels.

Efficient and Explicit Graph Embedding: The kernel $\kappa(\cdot,\cdot)$ is a positive definite kernel on $\mathcal{P}$. Given a graph database $\mathcal{G} = \{\mathcal{S}_i = \{\mathbf{h}_{1,i}, \ldots, \mathbf{h}_{n_i,i}\}\}_{i=1}^N$, we can use Eq. (6) to obtain the $(N \times N)$ empirical kernel (a.k.a. Gram) matrix $\mathbf{K}$ that can be directly input to the quadratic program (QP) solver for the dual OCSVM. However, this would be expensive for graph databases with very large N, and even infeasible if N is too large for $\mathbf{K}$ to fit in memory. Apart from computational reasons, the kernel embedding of a distribution $\mathsf{P} \in \mathcal{P}$ is a *function* –rather than a *vector* embedding– in the RKHS of functions, just like a distribution in essence is a probability density function in the input space. Having an explicit vector representation for each graph would provide flexibility, enabling the use of other detectors. Therefore, we aim to obtain a decomposition of the Gram matrix $\mathbf{K} = \mathbf{H}\mathbf{H}^T$ such that $\mathbf{H} \in \mathsf{R}^{N \times r}$ can act as an empirical kernel map, where r is the rank of $\mathbf{K}$. Then,

$$\mathbf{K}_{ij} = \kappa(\widehat{\mathsf{P}}_i, \widehat{\mathsf{P}}_j) = \langle \mathbf{H}_i, \mathbf{H}_j \rangle \tag{7}$$

which can be seen akin to $\langle \phi(G_i), \phi(G_j) \rangle$, that is the empirical inner product of the implicit embeddings in the RKHS as induced by the kernel, corresponding to Eq. (3). Notably, $\mathbf{H}$ would consist of a vector embedding for each graph, i.e. $\mathbf{H}_i \equiv h_{G_i}$, this time capturing characteristics w.r.t. the distribution of node embeddings. Following earlier terminology, we refer to this procedure as *MMD-pooling*.

The question is how to obtain such a decomposition. Eigendecomposition of $\mathbf{K} = \mathbf{U}\mathbf{\Sigma}\mathbf{U}^T$ is an option, where the empirical kernel map can be written as $\mathbf{H} = \mathbf{U}\mathbf{\Sigma}^{1/2}$, i.e. as the eigenvectors of $\mathbf{K}$ scaled by the square-root of their corresponding eigenvalues. However, it takes $O(N^2 r)$ time and $O(N^2)$ space as $\mathbf{K}$ needs to be explicitly constructed in memory. Therefore, it does not address the aforementioned efficiency challenges.

Instead, we use the Nyström method, widely used for scaling kernelized algorithms [31]. Given $\mathcal{G}$, it selects a subsample $\mathcal{B} \subset \mathcal{G}$ of size $k < N$,[1] and provides a rank-k approximation:

$$\mathbf{K} \approx \mathbf{K}_{\mathcal{G},\mathcal{B}}\, \mathbf{K}_{\mathcal{B},\mathcal{B}}^{-1}\, \mathbf{K}_{\mathcal{G},\mathcal{B}}^T \tag{8}$$

where $\mathbf{K}_{\mathcal{G},\mathcal{B}} \in \mathsf{R}^{N \times k}$ is the Gram matrix between all examples in $\mathcal{G}$ and those in the subsample $\mathcal{B}$. Similarly, $\mathbf{K}_{\mathcal{B},\mathcal{B}} \in \mathsf{R}^{k \times k}$ is the Gram matrix between pairs of examples in $\mathcal{B}$. Note that the Nyström method does not require the $(N \times N)$ $\mathbf{K}$ matrix explicitly in memory. It only necessitates the kernel computation between $Nk + k^2$ graph pairs, effectively down-scaling complexity to linear in the database size. The matrix inverse $\mathbf{K}_{\mathcal{B},\mathcal{B}}^{-1}$ can also be carried out efficiently, considering k is a small constant. Upon eigendecomposition of the $(k \times k)$ matrix $\mathbf{K}_{\mathcal{B},\mathcal{B}} = \mathbf{V}\mathbf{\Lambda}\mathbf{V}^T$, we can derive the approximation for the empirical kernel map $\mathbf{H}$ as

$$\mathbf{H} \approx \mathbf{K}_{\mathcal{G},\mathcal{B}}\, \mathbf{V}\, \mathbf{\Lambda}^{-1/2}\,. \tag{9}$$

This way MMD-pooling produces explicit graph-level embeddings efficiently.

B. Anomaly Detection

Having obtained explicit vector embeddings of graphs, we train a one-class classifier, optimizing the Deep-SVDD objective [32]:

$$\min_{\mathcal{W}} \quad \frac{1}{N}\sum_{i=1}^{N} \|\text{GIN}(G_i;\mathcal{W}) - \mathbf{c}\|_2^2 + \frac{\lambda}{2}\sum_{l=1}^{L} \|\mathbf{W}^{(l)}\|_F^2 \tag{10}$$

where $\text{GIN}(G_i) = \mathbf{h}_{G_i}$ denotes the vector embedding for G_i (based on Mean- or MMD-pooling), $\mathbf{W}^l$ denotes the (MLP) parameters of GIN at the l-th layer, $\mathcal{W} = \{\mathbf{W}^{(1)}, \ldots, \mathbf{W}^{(L)}\}$, $\mathbf{c}$ is the center of the hypersphere in the representation space (set to the average of all graph representations upon initializing the GIN), and finally λ is the weight-decay hyperparameter.

As discussed in [32], deep SVDD classification suffers from "hypersphere collapse", where the trained model maps all input instances directly to the fixed center $\mathbf{c}$. We employ

[1] Nyström method's performance depends on the sampling scheme; we use random sampling for efficiency, with *multiple* sample sizes as hyperparameter.

the regularizations proposed therein (no bias terms, etc.) to prevent this problem.

After training the model on all graphs, the distance to center is used as the anomaly score for each graph, that is

$$score(G_i) = \|\text{GIN}(G_i; \mathcal{W}) - \mathbf{c}\|_2 \ . \tag{11}$$

Overall, GLAM consists of an L-layer GIN architecture (with 2-layer MLPs, see Eq. (1)) for node embedding, followed by readout (i.e. Mean- or MMD-pooling), trained end-to-end via stochastic gradient descent, optimizing the deep SVDD objective at the output layer.

C. Unsupervised Model Selection

Recently, two model selection techniques have been proposed for deep unsupervised disentangled representation learning, namely UDR [33] and ModelCentrality (MC) [34].[2] Both are simple consensus-based approaches, leveraging the agreement between models in the candidate pool to assign a "reliability" score to each model. We extend on these ideas for GLAM by fine-tuning the reliability scores recursively. We compare to UDR and MC in the experiments.

Specifically, we extend the idea of MC [34] by computing centrality *recursively* based on a weighted bipartite network between candidate models and input graphs—wherein a model gains higher centrality (i.e. reliability) the more they point with high anomaly score (edge weight) to graphs that are pointed by other high-centrality (reliable) models.

One of the earliest methods for computing centrality, namely hubness h and authority a, of pages on the Web is the HITS algorithm [37]. Here, we employ this idea to estimate model "reliability" by constructing a complete bipartite network between M candidate models and N graphs, where

$h_i \ \propto$ sum of a_j's of all graphs j that model i points to ,

$a_j \ \propto$ sum of h_i's of all models i that point to graph j ,

which are estimated alternatingly over iterations. Upon convergence, the model with the largest hubness can be selected. However, we recognize that this approach also provides an ensemble ranking of the graphs based on the final authority scores. We employ this strategy (called HITS-ENS) for GLAM.

IV. Experiments

A. Setup

1) **Datasets:** We evaluate GLAM on 15 public benchmark graph databases, 11 with node-labeled graphs and 4 containing node-attributed graphs.[3] A summary of the datasets is given in Table II. Detailed descriptions can be found in full version.

Our datasets are repurposed from binary graph classification datasets, where we designate one class as the inlier class, and down-sample the other class(es) at ≈5% to constitute the anomalous class.[3] For training and evaluation, we split each dataset into two; training data consists exclusively of inliers and test data contains both inliers and anomalies. Note that there exists **no** validation set containing labeled anomalies, since we consider unsupervised detection.

[2]We found a couple of existing work on UMS for anomaly detection [35], [36] to be computationally too expensive and relatively much less effective.

[3] All datasets are from TU Datasets: https://chrsmrrs.github.io/datasets/docs/datasets/.

Table II
DATASETS IN EXPERIMENTS. NUMBER OF ATTRIBUTES IN PARENTHESES.

Name	Avg #nodes	Type	|Train|	|Test|
MIXHOP	100	Node-Labeled	532	493
PROTEINS	39.06	Node-Labeled	358	333
TOX21	18.09	Node-Labeled	472	503
COLLAB	74.49	Node-Labeled	395	397
IMDB	19.77	Node-Labeled	270	240
NCI1	29.87	Node-Labeled	1014	1096
MUTAGEN	30.32	Node-Labeled	1189	1274
REDDIT	23.93	Node-Labeled	2418	2594
DD	284.32	Node-Labeled	375	336
AIDS-L	15.69	Node-Labeled	202	206
DHFR-L	42.43	Node-Labeled	154	147
BZR	35.75	Attributed (3)	170	154
COX2	41.22	Attributed (3)	195	176
AIDS-A	15.69	Attributed (4)	202	206
DHFR-A	42.43	Attributed (3)	154	147

2) **Baselines:** We compare to two types of baselines:

i) Two-stage baselines first use unsupervised graph-level embedding/kernel techniques to obtain vector/kernel representations of the graphs (stage 1: • Weisfeiler-Lehman (WL) kernel [26], • Propagation kernel (PK) [24], and graph2vec (G2VEC) [25]), and then employ point outlier detectors in the embedding/kernel space (stage 2: • density-based LOF [38] and • one-class based OCSVM [39]). Note that WL and G2VEC apply to labeled graphs only.

ii) Node-level baselines: These are recent GNN based node anomaly detection methods in a *single* graph. We repurpose them to graph-level detection by scoring each graph with the average anomaly scores of its nodes.

- OCGNN [19] builds on GNN-based node embeddings learned through the OCSVM objective;
- DOMINANT [20] employs a reconstruction-based loss for both graph structure and node attribute vectors.

B. Results

1) **Detection Performance:** Table III gives the ROC-AUC performances for GLAM and all the baselines on each dataset.[4]

On average across HPs, GLAM outperforms all the baselines. Owing to different characteristics of the datasets, we do not expect GLAM to be the best model on every dataset. However, we observe that this is still true on a majority (8 out of 15) of the datasets. Moreover, no baseline stands out as a clear runner-up – some baselines which exhibit the best performance for a dataset can be seen to be arbitrarily bad on others.

The comparison can be further demonstrated in Figure 2, which displays the results of pairwise one-sided Wilcoxon signed rank tests between all competing methods. The entry in cell (i, j) is the p-value, given the (alternative) hypothesis that method j performs better than method i against the null (that they are not different). Each paired test is done on the ROC-AUC values from 15 datasets.

[4]DOMINANT threw out-of-memory error on REDDIT and DD. OCGNN threw that error on REDDIT, and is incompatible with MIXHOP's dense format.

Table III

ANOMALY DETECTION PERFORMANCE OF ALL METHODS. FOR BASELINES, WE RUN THE EXPERIMENT OVER A GRID OF HYPERPARAMETERS[5] AND REPORT THE AVERAGE AND STAND. DEV. OF ROC-AUC SCORES. AS GLAM EMPLOYS (UNSUPERVISED) MODEL SELECTION, IT ONLY OUTPUTS **one** RANKING, AND WE SIMPLY REPORT ITS ROC-AUC. PER DATASET RANK PROVIDED IN PARENTHESES (THE LOWER THE BETTER). AVERAGE PERFORMANCE AND RANK ACROSS DATASETS GIVEN IN THE LAST ROWS. SYMBOLS ▲ AND △ DENOTE THE CASES WHERE GLAM IS SIGNIFICANTLY BETTER THAN BASELINE W.R.T. THE WILCOXON SIGNED RANK TEST, $p<0.01$ AND $p<0.1$ RESPECTIVELY. (O.O.M.: OUT-OF-MEMORY)

Dataset	PK-LOF	PK-OCSVM	WL-OCSVM	WL-LOF	G2VEC-LOF	G2VEC-OCSVM	DOMINANT	OCGNN	GLAM
MIXHOP	0.67 ± 0.17(5)	0.69 ± 0.14(4)	0.72 ± 0.09(3)	0.57 ± 0.01(7)	0.67 ± 0.08(5)	0.51 ± 0.03(8)	**1.00 ± 0.00(1)**	N/A	0.99(2)
PROTEINS	0.68 ± 0.12(5)	0.67 ± 0.09(6)	0.72 ± 0.05(4)	0.78 ± 0.03(2)	0.47 ± 0.05(8)	0.46 ± 0.09(9)	0.48 ± 0.23(7)	0.75 ± 0.34(3)	**0.82(1)**
TOX21	0.53 ± 0.05(5)	0.57 ± 0.04(4)	0.65 ± 0.05(3)	0.66 ± 0.02(2)	0.47 ± 0.06(8)	0.47 ± 0.06(8)	0.53 ± 0.02(5)	0.52 ± 0.05(7)	**0.70(1)**
COLLAB	0.73 ± 0.05(7)	0.78 ± 0.02(5)	0.81 ± 0.06(4)	0.84 ± 0.00(3)	0.54 ± 0.18(9)	0.75 ± 0.03(6)	0.85 ± 0.02(2)	0.58 ± 0.22(8)	**0.87(1)**
IMDB	0.61 ± 0.09(6)	0.68 ± 0.06(2)	0.65 ± 0.04(3)	0.63 ± 0.03(4)	0.45 ± 0.09(8)	0.35 ± 0.04(9)	0.62 ± 0.01(5)	0.51 ± 0.05(7)	**0.70(1)**
NCI1	0.64 ± 0.06(2)	0.45 ± 0.07(7)	**0.73 ± 0.04(1)**	0.60 ± 0.08(3)	0.48 ± 0.10(6)	0.33 ± 0.11(9)	0.49 ± 0.12(5)	0.45 ± 0.16(7)	0.59(4)
MUTAGEN	0.57 ± 0.05(4)	0.53 ± 0.02(7)	0.56 ± 0.03(6)	0.52 ± 0.03(8)	**0.64 ± 0.04(1)**	0.57 ± 0.05(4)	0.59 ± 0.07(2)	0.49 ± 0.06(9)	0.59(2)
REDDIT	0.45 ± 0.06(6)	0.75 ± 0.02(3)	0.54 ± 0.14(5)	**0.83 ± 0.01(1)**	0.43 ± 0.07(7)	0.58 ± 0.04(4)	O.O.M.	O.O.M.	0.76(2)
DD	0.83 ± 0.06(4)	0.76 ± 0.04(5)	**0.92 ± 0.01(1)**	0.91 ± 0.01(2)	0.31 ± 0.03(8)	0.46 ± 0.03(7)	O.O.M.	0.60 ± 0.06(6)	0.84(3)
AIDS-L	0.77 ± 0.11(4)	0.73 ± 0.10(5)	**0.95 ± 0.04(1)**	0.94 ± 0.07(2)	0.30 ± 0.16(8)	0.12 ± 0.12(9)	0.68 ± 0.04(6)	0.42 ± 0.14(7)	0.91(3)
DHFR-L	0.64 ± 0.07(5)	0.74 ± 0.05(4)	0.77 ± 0.07(3)	0.81 ± 0.05(2)	0.62 ± 0.12(6)	0.40 ± 0.08(7)	0.36 ± 0.07(9)	0.37 ± 0.14(8)	**0.83(1)**
BZR	**0.69 ± 0.11(1)**	0.58 ± 0.06(2)	-	-	-	-	0.37 ± 0.01(5)	0.55 ± 0.07(3)	0.55(3)
COX2	0.70 ± 0.05(2)	0.55 ± 0.10(4)	-	-	-	-	0.42 ± 0.02(5)	0.56 ± 0.13(3)	**0.71(1)**
AIDS-A	0.53 ± 0.12(3)	0.84 ± 0.04(2)	-	-	-	-	0.05 ± 0.00(5)	0.43 ± 0.11(4)	**0.94(1)**
DHFR-A	0.70 ± 0.06(2)	0.62 ± 0.03(3)	-	-	-	-	0.43 ± 0.00(5)	0.58 ± 0.06(4)	**0.82(1)**
avg (labeled)	*0.65 ± 0.11*	*0.67 ± 0.11*	*0.73 ± 0.13*	*0.74 ± 0.14*	*0.49 ± 0.13*	*0.47 ± 0.15*	*0.62 ± 0.20*	*0.52 ± 0.11*	***0.78 ± 0.13***
avg (all)	*0.65 ± 0.10*	*0.66 ± 0.11*	-	-	-	-	*0.53 ± 0.24*	*0.52 ± 0.10*	***0.77 ± 0.13***
avg rank	**4.07▲**	**4.20▲**	**3.09**	**3.27△**	**6.73▲**	**7.27▲**	**4.77▲**	**5.85▲**	**1.8**

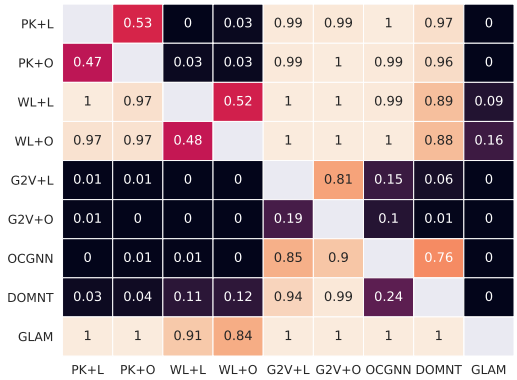

Figure 2. Comparison of detection methods by one-sided paired Wilcoxon signed-rank test. p-values smaller than 0.05 correspond to the cases where col-method is significantly better than the row-method.

Table IV

(LEFT) MEAN±STD. PERFORMANCE OF CANDIDATE MODELS (AVG'ED OVER ALL HP CONFIGS) BASED ON MMD- VS. MEAN-POOLING. (RIGHT) PERFORMANCE CHANGE (%) AFTER MODEL SELECTION FROM MEAN-POOL *only* VS. MMD-POOL *only* (RATHEN THAN BOTH). MMD- AND MEAN-POOLING ARE COMPLEMENTARY, BOOSTING OVERALL PERFORMANCE.

Dataset	MMD-pool	Mean-pool
MIXHOP	0.92 ± 0.07	**0.98 ± 0.02**
PROTEINS	**0.79 ± 0.04**	0.71 ± 0.11
TOX21	0.60 ± 0.04	**0.70 ± 0.04**
COLLAB	0.84 ± 0.04	**0.87 ± 0.07**
IMDB	**0.62 ± 0.06**	0.59 ± 0.05
NCI1	0.56 ± 0.04	**0.62 ± 0.04**
MUTAGEN	0.54 ± 0.06	**0.61 ± 0.06**
REDDIT	**0.70 ± 0.06**	0.55 ± 0.07
DD	0.70 ± 0.07	**0.85 ± 0.04**
AIDS-L	0.79 ± 0.07	**0.86 ± 0.07**
DHFR-L	0.77 ± 0.08	**0.79 ± 0.06**
BZR	0.50 ± 0.12	**0.69 ± 0.05**
COX2	0.65 ± 0.10	**0.70 ± 0.07**
AIDS-A	**0.83 ± 0.10**	0.64 ± 0.12
DHFR-A	0.72 ± 0.09	**0.84 ± 0.05**

GLAM:	w/o MMD	w/o Mean
	1.00	0
	-6.49	1.20
	0	-1.11
	9.09	-11.11
	3.33	-1.16
	-9.375	-4.47
	6.35	-1.72
	0	-1.22
	10.60	-7.27
	-40.74	5.00
	5.61	-3.70
	-42.42	0
	21.43	16.67
	1.39	4.05
	5.75	1.20
avg :	**-2.29**	**-0.24**

Not only does GLAM outperform all the baselines on average (as shown in Table III), as these test results show, the difference is also significant at $p \leq 0.01$ against all baselines but WL. WL appears more competitive than other baselines, however p-values are still fairly low (0.09 w/ LOF and 0.16 w/ OCSVM), and it comes with the caveat that it only applies to node-labeled graphs. Based on these results, we conclude that end-to-end graph-level detection with GLAM is more effective than these existing baselines.

2) **Mean-pooling vs. MMD-pooling:** In Table IV (left), we compare the performance of the models in the MMD vs. Mean pools on average (avg.'ed across HPs) for all 15 datasets. The better pooling technique is not consistent across datasets. Moreover, the differences can be quite large in both directions, showcasing their complementary strengths. To leverage both, GLAM combines these two pools, over which model selection is performed. As shown in Table IV (right), using *both* Mean- *and* MMD-pooling achieves improved results as compared to vanilla Mean-only or MMD-only pooling. We see that there is a drop in performance when model selection excludes Mean-pooling and an even bigger drop in performance when it excludes MMD-pooling.

Case Study: Next we take a closer look at the anomalous graphs detected by Mean- vs. MMD-pooling. Figure 1 shows the node embedding spaces by MMD- vs. Mean-pooling, respectively, on PROTEINS. Inlier nodes (green) span the whole background, as expected, while those of MMD anomalous graphs (red, orange) cluster in small zones, exhibiting distinct distributions. MMD-anomalies also look quite distinct visually. Notice that Mean-pooling alone, without the distributional "lens", falls short in spotting them.

3) **Model Selection:** To analyze the benefits of employing unsupervised model selection techniques, in Figure 3 we compare (using Wilcoxon signed rank test, as before) the performance of the model as selected by UDR, MC, HITS, or HITS-ENS across datasets.We also include the average model in this comparison. We find that all strategies are competitive, significantly outperforming Average at $p \leq 0.1$. HITS-ENS, the consensus ranking by HITS that GLAM employs, outperforms others and is the most competitive.

In Figure 4, we show the performance for all candidate models in the GLAM pool (Mean+MMD) per dataset (gray dots), where Average (circle) and the GLAM-selected model's performance (triangle) are marked. Notice that the pools consist

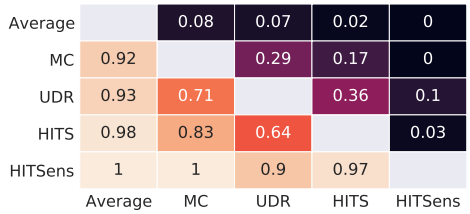

Figure 3. Comparison of selection methods. Improvement over average performance is significant at $p \leq 0.1$.

of models with a large variation of performance, demonstrating the potential value for selection. Notably, GLAM (with selection) is consistently similar to or better than Average, providing up to 16% improvement.

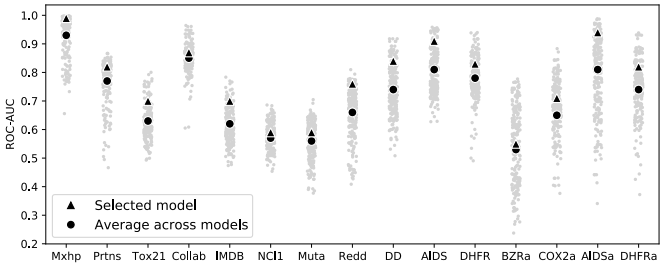

Figure 4. Performance of GLAM models w/ different HP configs in the pool. GLAM (w/ selection) consistently improves over Avg. (i.e. random choice).

V. Conclusion

In this work we presented GLAM, a novel Graph-Level Anomaly detection Model based on GNNs. GLAM employs an end-to-end anomaly detection objective and targets distribution graph anomalies as graphs with anomalous sets of nodes with designed MMD-pooling. We also systematically address the unsupervised model selection (UMS) problem. Extensive experiments showed that GLAM significantly outperforms key baselines in expectation across varying hyperparameter values. We also showed that both MMD-pooling and UMS are key players in GLAM's effectiveness.

References

[1] Q. Rong Jiang and J. Ma, "A novel graph kernel on chemical compound classification." *J. Bioinform. Comput. Biol.*, vol. 16, no. 6, 2018.

[2] A. Markovitz, G. Sharir, I. Friedman, L. Zelnik-M, and S. Avidan, "Graph embedded pose clustering for anomaly detection." in *CVPR*, 2020.

[3] W. Eberle, L. Holder, and D. Cook, "Identifying threats using graph-based anomaly detection," in *Mach. Learn. in Cyber Trust*, 2009.

[4] E. A. Manzoor, S. M. Milajerdi, and L. Akoglu, "Fast memory-efficient anomaly detection in streaming heterogeneous graphs." in *KDD*, 2016.

[5] C. Liu, X. Yan, H. Yu, J. Han, and P. S. Yu, "Mining behavior graphs for backtrace of noncrashing bugs." in *SDM*. SIAM, 2005, pp. 286–297.

[6] F. Monti, F. Frasca, D. Eynard, D. Mannion, and M. M. Bronstein, "Fake news detection on social media using geometric deep learning," *arXiv:1902.06673*, 2019.

[7] L. Akoglu, H. Tong, and D. Koutra, "Graph based anomaly detection and description: a survey." *DAMI*, vol. 29, no. 3, pp. 626–688, 2015.

[8] T. Idé and H. Kashima, "Eigenspace-based anomaly detection in computer systems." in *KDD*. ACM, 2004, pp. 440–449.

[9] B. Hooi, H. A. Song, A. Beutel, N. Shah, K. Shin, and C. Faloutsos, "FRAUDAR: Bounding graph fraud in the face of camouflage," in *KDD*. ACM, 2016, pp. 895–904.

[10] C. C. Noble and D. J. Cook, "Graph-based anomaly detection." in *KDD*. ACM, 2003, pp. 631–636.

[11] B. Perozzi and L. Akoglu, "Scalable anomaly ranking of attributed neighborhoods." in *SDM*. SIAM, 2016, pp. 207–215.

[12] D. Eswaran, C. Faloutsos, S. Guha, and N. Mishra, "SpotLight: Detecting anomalies in streaming graphs." in *KDD*. ACM, 2018.

[13] R. Chalapathy and S. Chawla, "Deep learning for anomaly detection: A survey," 2019, cite arxiv:1901.03407.

[14] L. Ruff, J. R. Kauffmann, R. A. Vandermeulen, G. Montavon, W. Samek, M. Kloft, T. G. Dietterich, and K.-R. Müller, "A unifying review of deep and shallow anomaly detection." *arXiv:2009.11732*, 2020.

[15] B. Perozzi, R. Al-Rfou, and S. Skiena, "DeepWalk: Online learning of social representations," in *KDD*. ACM, 2014, pp. 701–710.

[16] W. L. Hamilton, Z. Ying, and J. Leskovec, "Inductive representation learning on large graphs." in *NIPS*, 2017, pp. 1024–1034.

[17] T. Zhao, C. Deng, K. Yu, T. Jiang, D. Wang, and M. Jiang, "Error-bounded graph anomaly loss for gnns," in *CIKM*, 2020, pp. 1873–1882.

[18] W. Yu, W. Cheng, C. C. Aggarwal, K. Zhang, H. Chen, and W. Wang, "Netwalk: A flexible deep embedding approach for anomaly detection in dynamic networks," in *KDD*. ACM, 2018, pp. 2672–2681.

[19] X. Wang, Y. Du, P. Cui, and Y. Yang, "OCGNN: one-class classification with graph neural networks," *CoRR*, vol. abs/2002.09594, 2020.

[20] K. Ding, J. Li, R. Bhanushali, and H. Liu, "Deep anomaly detection on attributed networks," in *SDM*. SIAM, 2019, pp. 594–602.

[21] A. Kumagai, T. Iwata, and Y. Fujiwara, "Semi-supervised anomaly detection on attributed graphs," *arXiv:2002.12011*, 2020.

[22] M. Niepert, M. Ahmed, and K. Kutzkov, "Learning convolutional neural networks for graphs," in *ICML*, vol. 48, 2016, pp. 2014–2023.

[23] K. Xu, W. Hu, J. Leskovec, and S. Jegelka, "How powerful are graph neural networks?" in *ICLR*. OpenReview.net, 2019.

[24] M. Neumann, R. Garnett, C. Bauckhage, and K. Kersting, "Propagation kernels: efficient graph kernels from propagated information," *Machine Learning*, vol. 102, no. 2, pp. 209–245, 2016.

[25] A. Narayanan, M. Chandramohan, R. Venkatesan, L. Chen, Y. Liu, and S. Jaiswal, "graph2vec: Learning distributed representations of graphs," *arXiv:1606.08928*, 2017.

[26] N. Shervashidze, P. Schweitzer, E. J. Van Leeuwen, K. Mehlhorn, and K. M. Borgwardt, "Weisfeiler-Lehman graph kernels." *J. Mach. Learn. Res.*, vol. 12, no. 9, 2011.

[27] G. Pang, C. Shen, L. Cao, and A. V. D. Hengel, "Deep learning for anomaly detection: A review," *ACM Comp. Surv.*, vol. 54, no. 2, 2021.

[28] X. Ma, J. Wu, S. Xue, J. Yang, C. Zhou, Q. Z. Sheng, H. Xiong, and L. Akoglu, "A comprehensive survey on graph anomaly detection with deep learning," *TKDE*, 2021.

[29] O. Shchur, M. Mumme, A. Bojchevski, and S. Günnemann, "Pitfalls of graph neural network evaluation." *CoRR*, vol. abs/1811.05868, 2018.

[30] G. O. Campos, A. Zimek, J. Sander, R. J. G. B. Campello, B. Micenková, E. Schubert, I. Assent, and M. E. Houle, "On the evaluation of unsupervised outlier detection: measures, datasets, and an empirical study," *DMKD*, vol. 30, no. 4, pp. 891–927, 2016.

[31] C. K. I. Williams and M. W. Seeger, "Using the nyström method to speed up kernel machines." in *NIPS*, 2000, pp. 682–688.

[32] L. Ruff, N. Görnitz, L. Deecke, S. A. Siddiqui, R. A. Vandermeulen, A. Binder, E. Müller, and M. Kloft, "Deep one-class classification," in *ICML*, vol. 80, 2018, pp. 4390–4399.

[33] S. Duan, L. Matthey, A. Saraiva, N. Watters, C. Burgess, A. Lerchner, and I. Higgins, "Unsupervised model selection for variational disentangled representation learning." in *ICLR*. OpenReview.net, 2020.

[34] Z. Lin, K. Thekumparampil, G. Fanti, and S. Oh, "InfoGAN-CR and ModelCentrality: Self-supervised model training and selection for disentangling GANs," in *ICML*, 2020, pp. 6127–6139.

[35] H. O. Marques, R. J. G. B. Campello, A. Zimek, and J. Sander, "On the internal evaluation of unsupervised outlier detection." in *SSDBM*. ACM, 2015, pp. 7:1–7:12.

[36] N. Goix, "How to evaluate the quality of unsupervised anomaly detection algorithms?" *CoRR*, vol. abs/1607.01152, 2016.

[37] J. M. Kleinberg, "Authoritative sources in a hyperlinked environment," *J. ACM*, vol. 46, pp. 604–632, 1999.

[38] M. M. Breunig, H.-P. Kriegel, R. T. Ng, and J. Sander, "Lof: identifying density-based local outliers," in *SIGMOD*, 2000, pp. 93–104.

[39] B. Schölkopf, J. C. Platt, J. Shawe-Taylor, A. J. Smola, and R. C. Williamson, "Estimating the support of a high-dimensional distribution," *Neural Computation*, vol. 13, no. 7, pp. 1443–1471, 2001.