Unseen Anomaly Detection on Networks via Multi-Hpersphere Learning

Shuang Zhou [*] Xiao Huang [*] Ninghao Liu [†] Qiaoyu Tan [‡] Fu-Lai Chung [*]

Abstract

Network anomaly detection is a crucial task since a few anomalies can cause huge losses. Semi-supervised anomaly detection methods can effectively leverage a small number of labels as prior knowledge to enhance detection accuracy. But in real-world scenarios, novel types of anomalies (i.e., unseen anomalies) usually exist on networks which may present different characteristics with the seen anomalies and are hard to be identified by prior semi-supervised anomaly detection methods. In this paper, we propose the novel problem of unseen network anomaly detection that aims to identify both seen and unseen anomalies to eliminate potential dangers. Accordingly, we propose a method called Multi-hypersphere Graph Learning (MHGL) to effectively leverage existing labels by learning fine-grained normal patterns to discriminate anomalies. Experiments demonstrate that MHGL outperforms state-of-the-art methods significantly.

Keywords: Anomaly Detection; Attributed Networks; Unseen Anomaly

1 Introduction

Network anomaly detection is a crucial task that aims to identify instances that present strange behaviors and deviate from corresponding normal background significantly [1]. In real-world networked systems, such as social networks [2, 3] and payment transaction networks [4], a few anomalies, e.g., financial frauds, may cause tremendous loss. Therefore, detecting such network anomalies is of great significance.

Based on feature compactness, anomalies roughly fall into two groups [5, 6]: scattered anomalies (a.k.a., point anomalies) and rare categories (a.k.a., clustered anomalies or group anomalies). Scattered anomalies are individual instances that randomly appear in feature space, and deviate with the majority of other individual samples [7]. Whereas, rare categories denote some minority groups of data objects that exhibit compact properties in feature space (i.e., share similar behavior patterns) and deviate significantly from the vast majority as a whole [8, 9]. Although many methods [10, 11, 12, 13, 14] have been proposed for network anomaly detection, they are mainly unsupervised methods and cannot leverage existing labels as prior knowledge and may find data noise [15]. In practice, a few number of labeled anomalies are usually available which can be exploited for effective network anomaly detection. Recently, semi-supervised methods [4, 15] have been proposed to effectively leverage the limited labels to identify more anomalies with similar characteristics. However, in real-world scenarios, networks usually exist novel types of anomalies (i.e., unseen anomalies), because: (1) The limited amount of labeled seen anomalies can hardly cover all types of anomalies in practice; (2) New types of anomalies will emerge on networks as time goes. For instance, frauds may misuse new biometric technologies or devices for illegal gains. Existing semi-supervised network anomaly detection methods may fail to identify such unseen anomalies, since corresponding labels are not available during training and unseen anomalies may present different characteristics with the known anomalies. Therefore, it is necessary to propose a tailored approach that can effectively detect both seen and unseen network anomalies to eliminate potential dangers.

However, there are several difficulties to achieve this goal. First, networks contain heterogeneous information (i.e., node attributes and network structure), and existing solutions [31, 32] from other domains (e.g., images or multi-dimensional data) cannot be effectively applied to graph structural data. But the structure information can benefit network anomaly detection and should not be ignored. For example, a group of financial frauds are usually linked with each other, and the network structure helps to effectively identify more frauds. Hence, how to leverage the heterogeneous information to identify network anomaly is really a problem. Second, (unseen) anomalies usually present unknown patterns [6] while normal patterns are complicated, which may cause anomalies mix with the normal patterns (e.g., anoma-

[*]The Hong Kong Polytechnic University. {csszhou,xiaohuang,cskchung}@comp.polyu.edu.hk.

[†]University of Georgia. ninghao.liu@uga.edu.

[‡]Texas A&M University. qytan@tamu.edu.

lies might camouflage themselves as normal users [33]) and are hard to be detected. Existing efforts [16] that learn an advanced classifier by automatically selecting a threshold to reject nodes not belonging to seen classes as unseen classes may not properly handle the issue. Because the normal nodes with the same label often have diverse patterns, simply taking it as a binary classification problem may lead to sub-optimal performance.

Inspired by the fact that learning normal patterns may discriminate anomalies, we propose to address the above difficulties by mapping the heterogeneous network information into a unified latent space and learning fine-grained feature patterns to identify anomalies. Specifically, we firstly adopt established graph neural networks (GNNs) to map nodes in the network into a latent space. Then, we conduct multi-hypersphere learning on the latent space to generate diverse normal patterns via multiple hyperspheres. After that, both seen and unseen anomalies could be directly identified based on these hyperspheres.

The challenges of developing such an intuitive solution are two-fold. The first one is how to accurately identify fine-grained feature patterns with the purpose of conducting multi-hypersphere learning. Although conventional clustering methods, such as K-means [17], can be directly applied to get multiple feature clusters, these methods may lead to sub-optimal performance, as they ignore that normal patterns are complex and may mistake unseen anomalies as normal ones. Therefore, accurately modeling fine-grained patterns are necessary. Second, since anomalies are rare while the majority of samples are normal in practice, how to define an effective learning objective to leverage existing labels and handle the imbalanced data distribution to detect both seen and unseen anomalies is another challenge. Note that traditional hypersphere learning objective [18] is not suitable for our purpose as it only learns one hypersphere in latent space, whereas we want to conduct multi-hypersphere learning to effectively detect both seen and unseen anomalies.

To address the first challenge, we propose a component called Pattern Distribution Estimator. It can map the heterogeneous network information into a latent space and then accurately model the complex feature patterns into fine-grained patterns by estimating their pattern distributions. Second, we devise a novel multi-hypersphere learning objective to learn fine-grained normal patterns by enclosing each of them within a corresponding hypersphere in the latent space, while pushing labeled seen anomalies far away. After training, seen anomalies are mapped to far-away regions and are identified, while normal data are further enclosed within the learned multiple hyperspheres to discriminate unseen anomalies. We finally define anomaly score as the Euclidean distance between a testing instance and its closest hypersphere center in the latent space to discriminate all the anomalies with normal data. We summarize our contributions below.

- **Problem:** To the best of our knowledge, we are the first to investigate and formally define the novel problem of unseen anomaly detection on networks. In particular, we simulate real-world scenarios and emphasize on detecting both seen and unseen anomalies.

- **Algorithm:** We propose a principled method MHGL to identify seen and unseen anomalies. It combines the advantages of graph neural networks and our proposed multi-hypersphere learning to learn an effective anomaly detection model.

- **Experimental Findings:** We perform extensive experiments to verify the effectiveness of our method, and the results comprehensively demonstrate the superior performance of MHGL for unseen anomaly detection on networks. Besides, we also discuss how many labels are sufficient to achieve fair performance and how each component contributes to the overall performance.

2 Problem Statement

In this paper, we use bold uppercase letters, bold lowercase letters, lowercase letters, and calligraphic fonts to denote matrices (e.g., $\mathbf{X}$), vectors (e.g., $\mathbf{h}$), scalars (e.g., s), and sets (e.g., $\mathcal{V}$), respectively. Notably, the input is an attributed network $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathbf{X})$, where $\mathcal{V}$ is the set of nodes, i.e., $\{v_1, v_2, ..., v_n\}$ and $\mathcal{E}$ is the set of edges. The node attributes are represented by $\mathbf{X} = [\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_n] \in \mathbb{R}^{n\times f}$, $\mathbf{x}_i \in \mathbb{R}^f$ is the attribute vector associated with node v_i and f is the attribute dimension.

Problem: Unseen Anomaly Detection on Networks. Given an attributed network $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathbf{X})$, $\mathcal{V} = \mathcal{V}_{train} \cup \mathcal{V}_{test}$, where $\mathcal{V}_{train}$ denotes training set and $\mathcal{V}_{test}$ denotes testing set. Specifically, $\mathcal{V}_{train} = \mathcal{N}_{train} \cup \mathcal{S}_{train}$, where $\mathcal{N}_{train}$ is a set of labeled normal nodes and $\mathcal{S}_{train}$ denotes a set of labeled seen anomalies; $\mathcal{V}_{test} = \mathcal{N}_{test} \cup \mathcal{S}_{test} \cup \mathcal{U}$, where $\mathcal{N}_{test}$ and $\mathcal{S}_{test}$ are respectively the rest of normal nodes and the rest of seen anomalies, while $\mathcal{U}$ is a set of unseen anomalies that have never appeared in $\mathcal{V}_{train}$. Our goal is to learn an anomaly detection model, which can effectively leverage existing labels, and output a ranking list where (both seen and unseen) anomalies[1] own higher anomaly score than normal data and are detected from the network.

[1]In this paper, we mainly focus on detecting rare categories.

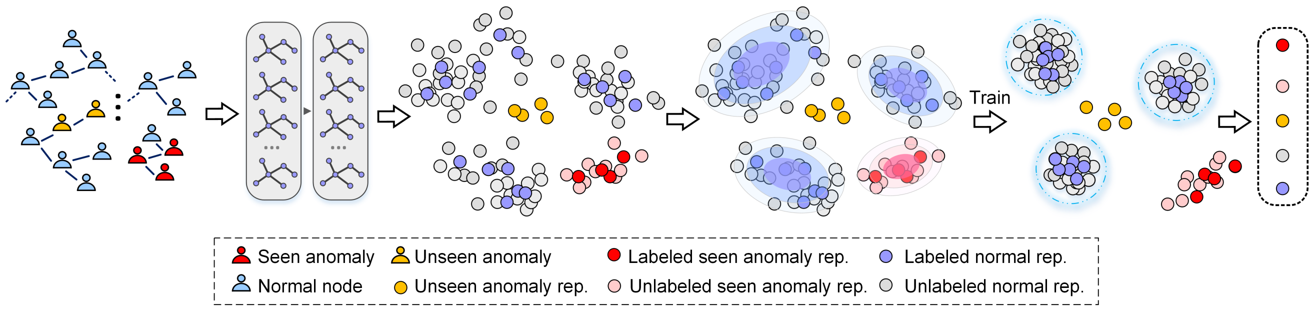

Figure 1: Overview of the Multi-hypersphere Graph Learning method. It first leverages GNNs to obtain node representations in latent space, then estimates diverse fine-grained feature patterns, and finally samples high-confidence data to conduct multi-hypersphere learning. After training, seen anomalies are mapped to far-away regions and are identified, while normal data are further enclosed within the learned multiple hyperspheres to discriminate unseen anomalies.

3 Proposed Method

In this section, we present the details of the proposed Multi-hypersphere Graph Learning (MHGL) method for unseen anomaly detection on networks. Specifically, our method MHGL addresses the discussed challenges with the following key components: (1) Pattern Distribution Estimator, a tailored component that enables to accurately model fine-grained patterns by estimating pattern distributions to identify unseen abnormal patterns in the constructed latent space; and (2) a multi-hypersphere graph learning algorithm, which is devised to learn multiple hyperspheres to enclose fine-grained normal patterns to further discriminate both seen and unseen anomalies on the network. An overview of the proposed method MHGL is depicted in Figure 1.

3.1 Pattern Distribution Estimator

We hypothesize that the feature patterns of normal data and anomalies are complex, which may mix with unseen abnormal patterns thus hampering effective detection. To address this issue, we propose to divide the feature patterns into various fine-grained patterns to identify unseen abnormal patterns. Accordingly, we propose a component called Pattern Distribution Estimator (PDE), which is composed of two key building blocks: (1) a network encoder for learning node representations; (2) a distribution modeling block to estimate the fine-grained normal patterns as well as abnormal patterns on networks.

Network Encoder. In order to effectively leverage the heterogeneous network information for anomaly detection, we build the network encoder module. Specifically, it consists of multiple GNN layers that encode each node to a low-dimensional representation in latent space. Generally, most of existing GNNs adopt a neighborhood message-passing mechanism to compute node representations by aggregating information from nodes' neighborhood. Formally, a vanilla GNN layer can be defined as two key functions:

$$\begin{aligned} \mathbf{h}^k_{\mathcal{N}_i} &= \textsc{Aggregate}^k\left(\mathbf{h}^{k-1}_j, \forall v_j \in \mathcal{N}_i\right), \\ \mathbf{h}^k_i &= \textsc{Combine}^k\left(\mathbf{h}^{k-1}_i, \mathbf{h}^k_{\mathcal{N}_i}\right), \end{aligned} \tag{3.1}$$

where $\mathbf{h}^k_i$ is the node representation of v_i at the k-th layer and $\mathcal{N}_i$ is a set of nodes adjacent to node v_i. Note that, the Aggregate is to receive messages from the neighborhood and the Combine is to compute nodes' new representation based on the representation from the previous GNN layer and the information from neighbors.

To further capture the long-range dependencies on network to learn informative representations, we stack multiple GNN layers in the network encoder. Thus, it can be described as:

$$\mathbf{H} = \mathrm{GNN}^k(\mathbf{H}^{k-1}, \mathbf{A}), \tag{3.2}$$

where $\mathbf{H}$ is a matrix of output representations generated by the network encoder, which has incorporated nodes' attribute and structure information in a unified latent space. Note that the network encoder is compatible with established GNNs, and we employ GCN [19] in our implementation.

Pattern Distribution Estimation. After adopting the Network Encoder, we can represent all the nodes in the latent space, thus obtaining complex normal and abnormal feature patterns. The next is to identify fine-grained feature patterns which can discriminate unseen abnormal patterns. Although conventional clustering methods, such as K-means [17], can be directly applied to get multiple feature clusters, they may lead to sub-optimal performance, as they do not consider the diversity of normal patterns and may treat unseen anomalies as normal data by mistake. Accordingly, we adopt a Gaussian Mixture Model (GMM) [20] to model the complex feature patterns into fine-grained patterns by estimating their pattern distributions. Specifically, given the representations of normal data and anomalies respectively, the GMM represents each fine-grained pattern as a mixture of Gaussian distributions by

$$p(\mathbf{h}_j \mid \lambda) = \sum_{i=1}^{k} \omega_i \boldsymbol{g}\left(\mathbf{h}_j \mid \mu_i, \Sigma_i\right), \tag{3.3}$$

where $\mathbf{h}_j \in \mathbb{R}^d$ is the representation of node v_j, $\lambda = \{w_i, \mu_i, \Sigma_i\}$ are the GMM parameters, k is the number of Gaussians (e.g., pattern distributions), $w_{i,i=1,\ldots,k}$ are the mixture weights with a constraint that $\sum_{i=1}^{k} \omega_i = 1$, $g\left(\mathbf{h}_j \mid \mu_i, \Sigma_i\right)$ are the component Gaussian densities. Each component density is a Gaussian function with mean vector μ_i and covariance matrix Σ_i. Here, it summarizes the probability density of $\mathbf{h}_j$ belonging to the i-th pattern distribution, and we use γ_{ji} to denote it for simplicity.

To ensure obtaining fine-grained feature patterns, PDE estimates the patterns in a hierarchical manner. The main idea is to further split coarse-grained patterns into smaller ones. As aforementioned, γ_{ji} indicates how likely a node v_j lying in the i-th pattern distribution P_i, and we simply assign node v_j to the distribution that it owns the largest probability ξ_j via

$$\xi_j = \underset{i \in \{1,2,\ldots,k\}}{\arg\max}\, \gamma_{ji}, \tag{3.4}$$

and assign it to the corresponding node set $\mathcal{S}^i$. Hence, the granularity of each pattern can be reflected by the size of the node set $\mathcal{S}^i$. The detailed algorithm is summarized in Algorithm 1. For the sake of optimal parameters that best match the distribution of the training data, we adopt Expectation-Maximization (EM) algorithm [21] for optimization. After optimization, PDE can model the normal and abnormal patterns into fine-grained patterns, respectively.

Algorithm 1 Pattern Distribution Estimation (PDE).

```
Input: Node set $\mathcal{N}$, node representations $\mathbf{H}$, distribution number
    $k$, node set size $u$.
Output: Fine-grained feature patterns $P$, a node set $\mathcal{S}^i$ for each
    pattern.
 1: Based on Eq. 3.3, model input representations by $k$ distributions;
 2: Get pattern distributions $P_{i,i\in\{1,2,\ldots,k\}}$;
 3: Apply Eq. 3.4 to take node $v_j, \forall v_j \in \mathcal{N}$ into $\mathcal{S}^i$;
 4: for $i = 1$ to $k$ do
 5:     if $|\mathcal{S}^i| > u$ then
 6:         Compute split number $k' = \frac{|\mathcal{S}^i|}{u}$;
 7:         Take $k', \mathcal{S}^i$ as input, do PDE to further split $P_i$;
 8:         Add the obtained $P^i_{new}, \mathcal{S}^i_{new}$ into $P, \mathcal{S}^i$;
 9:     end if
10: end for
11: Return fine-grained patterns $P$, node sets $\mathcal{S}$.
```

3.2 Multi-hypersphere Graph Learning In previous sub-section, we introduce how to use the PDE to learn fine-grained feature patterns. The follow-up question is how to conduct multi-hypersphere learning, which can enclose the normal patterns to further discriminate both seen and unseen anomalies in the latent space. In this sub-section, we first present how to obtain hypersphere centers from the learned normal pattern distributions, then introduce a tailored multi-hypersphere learning objective, and finally describe how to perform model training to facilitate the unseen anomaly detection on networks.

Hypersphere Center Computation. In order to perform multi-hypersphere learning in the latent space, it is necessary to find a suitable hypersphere center for each fine-grained normal pattern. We exploit the PDE to find nodes that most likely belong to corresponding feature pattern to compute hypersphere centers. Specifically, we adopt Eq. 3.4 to infer a set of nodes $\mathcal{S}^i$ that most likely lie in the fine-grained pattern P_i . Therefore, the hypersphere centers $\mathbf{c}_i$ for all the fine-grained patterns can be obtained by averaging the corresponding node representations, namely

$$\mathbf{c}_i = \frac{1}{|\mathcal{S}^i|} \sum_{v_j \in \mathcal{S}^i} \mathbf{h}_j. \tag{3.5}$$

Multi-hypersphere Learning Objective. To facilitate multi-hypersphere learning in the latent space for effective anomaly detection, we propose a tailored multi-hypersphere learning objective. The intuitive idea is to enclose normal instances within corresponding hypersphere while pushing anomalies far away from all the normal hyperspheres by jointly learning the parameters of the network encoder and minimizing the volume of the data description hyperspheres. Formally, the multi-hypersphere learning objective is defined as:

$$\begin{aligned} &\min_{\Theta} \frac{1}{p \cdot n} \sum_{i=1}^{p} \sum_{j=1}^{n} \|\mathbf{h}_j - \mathbf{c}_i\|_2^2 + \\ &\frac{\sigma}{q \cdot p} \sum_{r=1}^{q} \sum_{i=1}^{p} \left(\|\mathbf{h}_r - \mathbf{c}_i\|_2^2\right)^{-1} + \frac{\lambda}{2}\|\Theta\|_F^2, \end{aligned} \tag{3.6}$$

where $\mathbf{c}_i \in \mathbb{R}^d$ is the center for the i-th normal hypersphere, Θ denotes all the learnable network parameters, p is the number of normal patterns P, n denotes the number of nodes in pattern P_i, σ serves as a weight to control the balance between the labeled normal and the abnormal term, and q is the number of labeled anomalies. The first term is a quadratic loss for penalizing the distance of each normal node representation to the corresponding hypersphere center $\mathbf{c}_i$. The second term is to penalize the inverse of distances such that anomalies will be mapped far away from all the normal hyperspheres. The last term is a network weight decay regularizer with hyperparameter $\lambda > 0$.

When minimizing the objective function, normal hyperspheres are contracted in the latent space. By training the model with fine-grained normal data and a few labeled seen anomalies, it will (i) map seen anomalies to far-away regions, (ii) learn a description boundary for each fine-grained normal pattern, such that unseen anomalies, which may originally reside in the complex normal patterns, can also be identified. Therefore, the trained model can effectively detect both seen and unseen anomalies. Specifically, for a given test node v_j, we can naturally define the anomaly score as the Euclidean distance between its own representation and the closest normal hypersphere center $\mathbf{c}_i$ in the latent space, i.e.,

$$score(v_j) = min\|\mathbf{h}_j - \mathbf{c}_i\|_2^2, \forall i \in \{1, 2, \ldots, p\}. \tag{3.7}$$

Model Learning. Ideally, the proposed learning objective requires both labeled normal and abnormal instances from corresponding fine-grained patterns respectively for model training. However, such a requirement is demanding since existing labels lack such fine-grained information. To overcome this issue, we propose to use pseudo-labels for model training.

The pseudo-labels are generated by leveraging the learned fine-grained pattern distributions from PDE. Note that, during the initialization stage of our method, we have adopted PDE to model the complex feature patterns into various fine-grained pattern distributions. Here, for each fine-grained pattern, we sample some high-confidence data to build initial hyperspheres and then generate virtual representations as pseudo-labels for multi-hypersphere training. Specifically, there are three steps: (1) For each pattern, we take the labeled nodes with probability value ξ_j (see Eq. 3.4) larger than a suitable threshold t as its high-confidence data; (2) For each hypersphere center, we calculate its distance with the corresponding high-confidence data and take the largest distance as its initial radius r, thus forming initial hyperspheres; (3) Finally, we repeatedly sample two representations from the corresponding hypersphere, combining them to generate virtual representations as pseudo-labels. Specifically, the virtual representations are generated by

$$\mathbf{h}_{\text{new}} = (1 - \beta) \cdot \mathbf{h}_a + \beta \cdot \mathbf{h}_b, \tag{3.8}$$

where β is a random variable in the range of (0, 1). The detailed algorithm is summarized in Algorithm 2.

Algorithm 2 Multi-hypersphere Graph Learning.

```
Input: Attributed network $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathbf{X})$, training epochs $T$,
    augmentation factor $\alpha$.
Output: A node list with anomaly score.
 1: Randomly initialize MHGL, base on Eq. 3.2 to get $\mathbf{H}$;
 2: Adopt the PDE to obtain $x$ normal patterns and $y$ abnormal
    patterns, and a node set $\mathcal{S}^i$ for each pattern;
 3: Calculate hypersphere centers $\mathbf{c}_{i,i\in[1,x+y]}$;
    // Identify high-confidence data
 4: Select a suitable probability value as threshold $t$;
 5: Add node $v_j$ into high-confidence set $\mathcal{F}^i$, if $\xi_j > t, \forall v_j \in \mathcal{S}^i$;
    // Compute initial radius for hyperspheres
 6: For $i$-th hypersphere, take $\arg\max_{j\in\mathcal{F}^i} \left(d_{euc}\left(\mathcal{F}^i_j, \mathbf{c}_i\right)\right)$ as $r_i$;
 7: For each hypersphere, take node $v_j \in \mathcal{V}$ with $d_{euc}(\mathbf{c}_i, \mathbf{h}_j) \leqslant r_i$
    as final high-confidence set $\mathcal{H}^i$;
 8: while $t < T$ do
 9:     for $i = 1$ to $\alpha \cdot |\mathcal{H}^i|$ do
10:         // Generate pseudo-labels
11:         Use Eq. 3.8 to generate $\mathbf{h}_{\text{new}}$;
12:         Add $\mathbf{h}_{\text{new}}$ into $\mathcal{D}_i$ as pseudo-labels for $\mathbf{c}_i$;
13:     end for
14:     Based on Eq. 3.6, minimize loss with $\mathcal{D}_i$;
15:     Update $\Theta$ via stochastic gradient descent;
16: end while
17: Compute anomaly scores based on Eq. 3.7 for $\mathcal{V}_{test}$.
```

4 Experiment

In this section, we perform empirical evaluations on real-world attributed networks to answer the following research questions: **RQ1.** How effectively does our method for anomaly detection, especially the unseen anomaly detection? **RQ2.** How is the label leveraging efficiency of our method? As there exists a trade-off between high performance and high cost, we intend to explore how many labels are sufficient for achieving fair performance. **RQ3.** What is the contribution of each module of our method to its overall performance?

4.1 Datasets

We employ three real-world attributed networks that have been widely used in prior researches [22, 23, 24] to evaluate the performance of different methods. Details of the datasets are as follows:

- **Computer** is from the Amazon co-purchase graph [25] in which nodes represent products and edges denote co-purchase relationships. Nodes fall into 10 classes based on product category, and node attributes are a bag-of-words representation of users' comments.

- **Photo** is also from the Amazon co-purchase graph [25] in which nodes denote products and the edges reflect co-purchase relationships. Node attributes are a bag-of-words representation of a product's reviews.

- **CS** is a co-author network from the Microsoft Academic Graph [26], in which nodes represent authors and the edges indicate co-author relationships. Node attributes are a bag-of-words representation of the keywords from an author's papers.

Following the standard anomaly detection settings [9, 29, 30], we take the nodes from the smallest classes

Table 1: Data statistics of the three real-world attributed networks with anomalies.

Datasets	CS	Computer	Photo
nodes	18,333	13,381	7,487
edges	81,894	245,778	119,043
attributes	6,805	767	745
classes	15	10	8
rare categories	3	2	2
total anomalies	909	580	696
anomalies prop.	4.96%	4.33%	9.30%
seen anomalies	420	297	331
unseen anomalies	489	283	365

(i.e., rare categories) as anomalies, while the remaining nodes are considered as normal data. As aforementioned, in real-world scenarios, a limited number of labels on normal instances and anomalies are usually available and there exist novel types of anomalies (i.e., unseen anomalies). To further replicate the scenarios, we follow related works [27] by taking one rare category as seen anomalies while considering the remained rare categories as unseen anomalies, and then randomly sample a few (e.g., $q = 20$) nodes from the seen anomalies as labels. On top of that, p% of the normal nodes are randomly sampled as labeled normal data. Both of the labeled seen anomalies and the labeled normal data are used as training set, while the remaining data serves as testing set. The details of the datasets are shown in Table 1.

4.2 Experimental Settings

Comparative Methods. We compare our proposed method with state-of-the-art semi-supervised methods for network anomaly detection. Details of these compared baseline methods are as follows:

- **SPARC** [9] is a cost-sensitive based method for rare category detection on attributed networks.
- **DeepSAD** [18] is a popular deep learning method that utilizes both labeled normal and abnormal instances for anomaly detection. In our experiments, it leverages node attributes as the input features.
- **OpenWGL** [16] is an advanced GNNs that can identify unseen classes. Here, we adopt it for unseen anomaly detection.
- **GDN** [15] is a state-of-the-art GCN-based method that leverages a few labeled anomalies for effective network anomaly detection.
- **OCGNN** [29] is an advanced method which combines GNNs with the classical one-class objective for network anomaly detection. It can only exploit normal instances to discriminate anomalies.

Evaluation Metrics. We follow previous works [12, 15, 30] and adopt three standard evaluation metrics (AUC, AUPR, and Precision@K) to measure the effectiveness of all the methods.

Implementation Details. We implement the proposed framework in PyTorch. Specifically, the network encoder consists of four GCN layers (dimension size is 256, 128, 64, and 32, respectively), with ReLU as activation function. In all the experiments, we optimize the objective function via Adam optimizer with suitable learning rate lr and a weight decay of 0.0005. Besides, we set the initial distribution number $k = 10$, training epochs $T = 300$, and augmentation factor $\alpha = 2$. We leverage the generated virtual representations (see Eq. 3.8) for model training, and tune the rest of hyper-parameters, e.g., node set size u (see ALG. 1), lr, and σ, by selecting ones with the best evaluation performance on the available labeled data. For all the comparative methods, codes are publicly accessible. For fair comparison, we follow a previous work [15] to split the existing labels into train and validation set with a suitable proportion (3:2), select the hyper-parameters with the best performance on the validation set, and report the results on the same testing data.

4.3 Effectiveness Results (RQ1)

Overall Performance. To replicate the real-world scenarios that people want to detect as many anomalies as possible to eliminate potential risks, we evaluate the overall performance (i.e., detect both seen and unseen anomalies) of all the methods. We set the number of labeled anomalies as 20 and the labeled normal ratio as 10%. We present the evaluation results w.r.t. AUC, AUPR, and Precision@K in Table 2. According to the results, we have the following observations: **(1)** For overall anomaly detection, our method MHGL outperforms all the comparative methods by a large margin in two datasets, while achieving comparable performance in the CS dataset. **(2)** For comparative anomaly detection methods, SPARC, GDN and DeepSAD, which leverage both normal and abnormal data for training, have achieved great performances and significantly outperform OCGNN, which only learns normal patterns to discriminate anomalies. It demonstrates that effectively leveraging existing labels and learning both normal and abnormal patterns is a key to achieve great performance for anomaly detection. **(3)** Network anomaly detection methods (e.g., SPARC, GDN, and our MHGL) can outperform DeepSAD, a conventional anomaly detection method which cannot leverage network structure information. It indicates that it is necessary to propose tailored methods to leverage the heterogeneous information on networks for anomaly detection.

Table 2: Overall performance comparison in AUC, AUPR, and Precision@K on the three datasets.

Methods	**Computer**			**Photo**			**CS**		
	AUC	**AUPR**	**P@600**	**AUC**	**AUPR**	**P@600**	**AUC**	**AUPR**	**P@600**
SPARC	0.8291	0.5217	0.4833	0.6977	0.2940	0.3383	0.8432	**0.5493**	0.6433
DeepSAD	0.8246	0.5064	0.4747	0.7002	0.3022	0.3489	0.7956	0.5129	0.6438
OpenWGL	0.8399	0.5357	0.4650	0.7416	0.3202	0.2296	0.8032	0.3275	0.3779
GDN	0.9486	0.7529	0.6537	0.7424	0.4969	0.4781	0.7264	0.5276	**0.7074**
OCGNN	0.9104	0.2752	0.3031	0.6765	0.2178	0.3241	0.7257	0.1403	0.2116
MHGL (Ours)	**0.9925**	**0.8768**	**0.8300**	**0.8604**	**0.5183**	**0.4950**	**0.8503**	0.5217	0.6498

Table 3: Performance comparison w.r.t. Precision@K for unseen anomaly detection on the three datasets.

Methods	**Computer**			**Photo**			**CS**		
	P@400	**P@500**	**P@600**	**P@400**	**P@500**	**P@600**	**P@400**	**P@500**	**P@600**
SPARC	0.0425	0.0500	0.0550	0.0625	0.0640	0.0683	0.0000	0.0080	0.0150
DeepSAD	0.1105	0.1030	0.0971	0.0791	0.0787	0.0786	0.0849	0.0871	0.0858
OpenWGL	0.1012	0.0840	0.0733	0.2518	0.2201	0.1842	0.1077	0.1007	0.0867
GDN	0.1026	0.1638	0.1937	0.0440	0.0518	0.0562	0.0119	0.0615	0.0827
OCGNN	0.1645	0.1483	0.1325	0.2945	0.2532	0.2034	0.1420	0.1349	0.1187
MHGL (Ours)	**0.1835**	**0.3190**	**0.3700**	**0.3543**	**0.3184**	**0.2925**	**0.1928**	**0.1830**	**0.1683**

Performance on Unseen Anomaly Detection. In the experiments, we also evaluate the performance of all the methods in identifying unseen anomalies (i.e., novel types of anomalies) on networks. We still set the number of labeled anomalies as 20 and the labeled normal ratio as 10% for all the datasets. We present the evaluation results w.r.t. Precision@K in Table 3. Note that, we temporarily mask all the seen anomalies in the output ranking list as normal, such that we can fairly evaluate these methods' effectiveness on unseen anomaly detection. Accordingly, we can see that the proposed method MHGL significantly outperforms all the comparison methods w.r.t. Precision@K on the three datasets. It verifies the effectiveness of our method for unseen anomaly detection on networks.

4.4 Label Leveraging Efficiency (RQ2) This subsection examines the label leveraging efficiency of our method. Efficiently utilizing labels is necessary in real-world scenarios, as it is difficult to obtain large amount of labeled data in practice. However, if not providing a sufficient amount of labels, it is difficult to achieve fair performance in identifying anomalies (especially unseen anomalies), as the task is really challenging. Therefore, we examine the label leveraging efficiency of the proposed model as well as another network anomaly detection methods and intend to explore how many labels are sufficient for fair performance. Accordingly, we set two sets of experiments to examine the efficiency of utilizing labeled anomalies and labeled normal data, respectively. For the former one, the number of labeled anomalies in training varies from 5 to 40, with labeled normal data fixed as 10%. For the latter one, we fix the number of labeled anomalies as 20 and vary the amount of labeled normal data from 3% to 10% in training. For all these experiments, the testing data is fixed unchanged. The results are reported in Figure 2 and Figure 3. As shown in Figure 2, the performance of GDN and our proposed MHGL increases when exploiting more labeled anomalies, whereas for OCGNN, which cannot leverage existing labeled anomalies, its performance is not improved. We further find that, when only a few (e.g., 5) labeled anomalies are available, adding labels usually get more obvious gains on performance than the scenarios where a large amount of labels (e.g., 30) are available. The results demonstrate that adopting labeled anomalies is conducive for better detection performance, which is intuitive and also consistent with the conclusions from related works [18]. As for leveraging normal data, the results in Figure 3 show that all the anomaly detection methods can achieve greater performance by providing more labeled normal data. However, as aforementioned, there exists a trade-off between high detection performance and high cost. So considering effectiveness, we recommend to adopt a small value (e.g., 20 labeled anomalies and 7% normal data) to achieve fair performance.

4.5 Ablation Study (RQ3) To better examine the contribution of each component in our method, we also include several variants of MHGL for ablation study. We respectively exclude the PDE, the multi-hypersphere learning, and both of them, thus resulting in three model variants: MHGL-, HGL, and HGL-. Note that excluding the PDE means that we use alternative method, such as K-means, for hypersphere center computation, and excluding the multi-hypersphere learning denotes merely building one hypersphere in

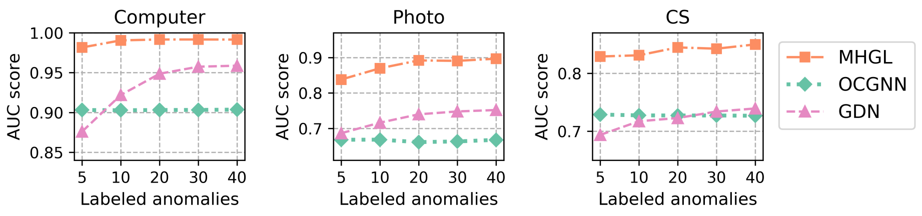

Figure 2: AUC Performance w.r.t. using different number of labeled anomalies.

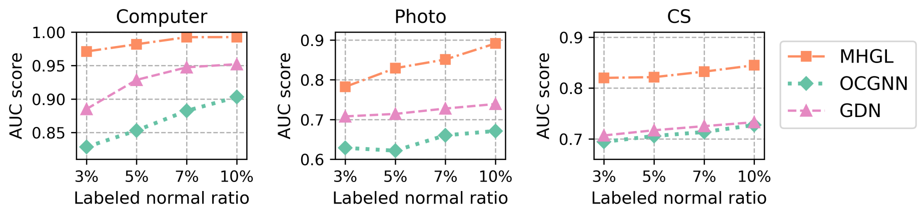

Figure 3: AUC Performance w.r.t. using different ratio of labeled normal data.

the latent space for model training. We present the overall anomaly detection results w.r.t. AUC and Precision@600 in Figure 4. Our observations are two-fold: (1) By incorporating the PDE, MHGL outperforms MHGL- for anomaly detection. Specifically, the averaged performance improvement w.r.t. AUC score and Precision@600 is 0.04 and 0.13, respectively. It is because PDE can accurately estimate the distribution of fine-grained normal and abnormal patterns. (2) By conducting multi-hypersphere learning, MHGL outperforms HGL significantly. For instance, the averaged performance boost w.r.t. AUC score and Precision@600 exceed 0.04 and 0.15, respectively. Since unseen anomalies usually present unknown patterns and may mix with normal patterns in latent space, learning the fine-grained normal patterns within multiple hyperspheres enables to identify unseen anomalies (as shown in Figure 1), thus enhancing the accuracy for network anomaly detection.

5 Conclusion

In this paper, we formally investigate the problem of unseen anomaly detection on networks. To handle this problem, we propose a novel multi-hypersphere graph learning framework MHGL, which consists of two main components: pattern distribution estimator (PDE) and multi-hypersphere learning. The PDE can encode nodes' attributes and network structure information into a unified latent space and estimate the complicated pattern distribution of normal and abnormal data, thus identifying unseen abnormal patterns in the latent space. The subsequent multi-hypersphere learning enables to enclose fine-grained normal patterns within multiple hyperspheres to further discriminate anomalies. Through empirical evaluations, we demonstrate that our proposed method MHGL can outperform state-of-the-art anomaly detection methods significantly.

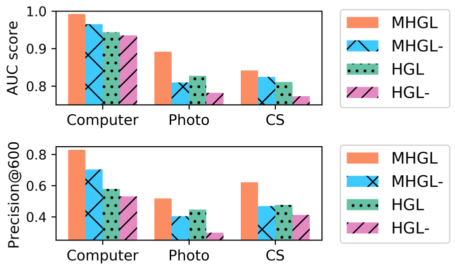

Figure 4: Ablation study to analyze the effect of each component in MHGL.

References

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

[2] X. Hu, J. Tang, and H. Liu, "Online social spammer detection," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 28, no. 1, 2014.

[3] Q. Tan, N. Liu, and X. Hu. Deep representation learning for social network analysis. *Frontiers in big Data*, 2:2, 2019.

[4] D. Wang, J. Lin, P. Cui, Q. Jia, Z. Wang, Y. Fang, Q. Yu, J. Zhou, S. Yang, and Y. Qi, "A semi-supervised graph attentive network for financial fraud detection," in *2019 IEEE International Conference on Data Mining (ICDM)*. IEEE, 2019, pp. 598–607.

[5] F. T. Liu, K. M. Ting, and Z.-H. Zhou, "On detecting clustered anomalies using sciforest," in *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*. Springer, 2010, pp. 274–290.

[6] G. Pang, C. Shen, L. Cao, and A. V. D. Hengel, “Deep learning for anomaly detection: A review,” *ACM Computing Surveys (CSUR)*, vol. 54, no. 2, pp. 1–38, 2021.

[7] H. Xu, Y. Wang, Y. Wang, and Z. Wu, “Mix: a joint learning framework for detecting both clustered and scattered outliers in mixed-type data,” in *2019 IEEE International Conference on Data Mining (ICDM)*. IEEE, 2019, pp. 1408–1413.

[8] J. Han, J. Pei, and M. Kamber, *Data mining: concepts and techniques*. Elsevier, 2011.

[9] D. Zhou, J. He, H. Yang, and W. Fan, “Sparc: Self-paced network representation for few-shot rare category characterization,” in *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, 2018, pp. 2807–2816.

[10] J. Gao, F. Liang, W. Fan, C. Wang, Y. Sun, and J. Han, “On community outliers and their efficient detection in information networks,” in *Proceedings of the 16th ACM SIGKDD international conference on Knowledge discovery and data mining*, 2010, pp. 813–822.

[11] N. Liu, X. Huang, and X. Hu, “Accelerated local anomaly detection via resolving attributed networks.” in *IJCAI*, 2017, pp. 2337–2343.

[12] Z. Peng, M. Luo, J. Li, L. Xue, and Q. Zheng, “A deep multi-view framework for anomaly detection on attributed networks,” *IEEE Transactions on Knowledge and Data Engineering*, 2020.

[13] L. Gutiérrez-Gómez, A. Bovet, and J.-C. Delvenne, “Multi-scale anomaly detection on attributed networks,” in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 34, no. 01, 2020, pp. 678–685.

[14] S. Zhou, Q. Tan, Z. Xu, X. Huang, and F. Chung. Subtractive aggregation for attributed network anomaly detection. In *Proceedings of the 30th ACM International Conference on Information & Knowledge Management*, pages 3672–3676, 2021.

[15] K. Ding, Q. Zhou, H. Tong, and H. Liu, “Few-shot network anomaly detection via cross-network meta-learning,” in *Proceedings of the Web Conference 2021*, 2021, pp. 2448–2456.

[16] M. Wu, S. Pan, and X. Zhu, “Openwgl: Open-world graph learning,” in *2020 IEEE International Conference on Data Mining (ICDM)*. IEEE, 2020, pp. 681–690.

[17] J. A. Hartigan and M. A. Wong, “Algorithm as 136: A k-means clustering algorithm,” *Journal of the royal statistical society. series c (applied statistics)*, vol. 28, no. 1, pp. 100–108, 1979.

[18] L. Ruff, R. Vandermeulen, N. Goernitz, L. Deecke, S. A. Siddiqui, A. Binder, E. Müller, and M. Kloft, “Deep one-class classification,” in *International conference on machine learning*. PMLR, 2018, pp. 4393–4402.

[19] T. N. Kipf and M. Welling, “Semi-supervised classification with graph convolutional networks,” in *International Conference on Learning Representations (ICLR)*, 2017.

[20] D. A. Reynolds, “Gaussian mixture models.” *Encyclopedia of biometrics*, vol. 741, pp. 659–663, 2009.

[21] A. P. Dempster, N. M. Laird, and D. B. Rubin, “Maximum likelihood from incomplete data via the em algorithm,” *Journal of the Royal Statistical Society: Series B (Methodological)*, vol. 39, no. 1, pp. 1–22, 1977.

[22] O. Shchur, M. Mumme, A. Bojchevski, and S. Günnemann, “Pitfalls of graph neural network evaluation,” *arXiv preprint arXiv:1811.05868*, 2018.

[23] Q. Zhao, Z. Ye, C. Chen, and Y. Wang, “Persistence enhanced graph neural network,” in *International Conference on Artificial Intelligence and Statistics*. PMLR, 2020, pp. 2896–2906.

[24] K. Zhou, X. Huang, Y. Li, D. Zha, R. Chen, and X. Hu, “Towards deeper graph neural networks with differentiable group normalization,” *arXiv preprint arXiv:2006.06972*, 2020.

[25] J. McAuley, C. Targett, Q. Shi, and A. Van Den Hengel, “Image-based recommendations on styles and substitutes,” in *Proceedings of the 38th international ACM SIGIR conference on research and development in information retrieval*, 2015, pp. 43–52.

[26] A. Sinha, Z. Shen, Y. Song, H. Ma, D. Eide, B.-J. Hsu, and K. Wang, “An overview of microsoft academic service (mas) and applications,” in *Proceedings of the 24th international conference on world wide web*, 2015, pp. 243–246.

[27] G. Pang, C. Ding, C. Shen, and A. v. d. Hengel, “Explainable deep few-shot anomaly detection with deviation networks,” *arXiv preprint arXiv:2108.00462*, 2021.

[28] L. Ruff, R. A. Vandermeulen, N. Görnitz, A. Binder, E. Müller, K.-R. Müller, and M. Kloft, “Deep semi-supervised anomaly detection,” *arXiv preprint arXiv:1906.02694*, 2019.

[29] X. Wang, B. Jin, Y. Du, P. Cui, Y. Tan, and Y. Yang, “One-class graph neural networks for anomaly detection in attributed networks,” *Neural Computing and Applications*, pp. 1–13, 2021.

[30] F. Zhang, H. Fan, R. Wang, Z. Li, and T. Liang, “Deep dual support vector data description for anomaly detection on attributed networks,” *International Journal of Intelligent Systems*, 2021.

[31] D. Hendrycks, M. Mazeika, and T. Dietterich, “Deep anomaly detection with outlier exposure,” *Proceedings of the International Conference on Learning Representations*, 2019.

[32] G. Pang, C. Shen, H. Jin, and A. v. d. Hengel, “Deep weakly-supervised anomaly detection,” *arXiv preprint arXiv:1910.13601*, 2019.

[33] Y. Dou, Z. Liu, L. Sun, Y. Deng, H. Peng, and P. S. Yu, “Enhancing graph neural network-based fraud detectors against camouflaged fraudsters,” in *Proceedings of the 29th ACM International Conference on Information & Knowledge Management*, 2020, pp. 315–324.