Unified Graph Embedding-Based Anomalous Edge Detection

Linshu Ouyang, Yongzheng Zhang* and Yipeng Wang
Institute of Information Engineering, Chinese Academy of Sciences, Beijing, China
School of Cyber Security, University of Chinese Academy of Sciences, Beijing, China
Email: {ouyanglinshu, zhangyongzheng, wangyipeng}@iie.ac.cn

***Abstract*—Detecting anomalous edges in graph-structured data plays an important role in many fields such as finance, social network, and network security. Recently, graph embedding based anomaly detection methods show promising results. These methods typically encode graph structure information into vector representation and apply general anomaly detection methods. However, since the parameters in these two parts are learned separately with different objectives, the learned representation may contain some information irrelevant to the task. It would be ideal if we can combine representation learning and anomaly detection into one objective function to force the model to focus on learning task relevant patterns. In this paper, we propose a novel end-to-end neural network architecture that can accurately estimate the probability distribution of edges in the graph based on its local structure. An edge has a high chance to be considered an anomaly if the probability of its existence is low. Extensive experiments on several public datasets at different scales show that the accuracy and scalability of our method outperform other methods by a large margin.**

***Index Terms*—Anomaly Detection, Anomalous Edge, Network Embedding**

I. Introduction

Anomaly detection in graph-structured data has attracted more and more attention in recent years as it has many important applications in different fields. For example, in the network intrusion detection problem, the network communication behavior can be represented as a directed graph, where the hosts are vertexes and the network connections are edges. The connections incurred by network attacks can be detected as anomalous edges [1]. In the social network scenario, users are regarded as vertexes, and the interactions between users are edges. Graph anomaly detection algorithms can be used to detect fraud or spam activities in this graph [2], [3].

In this work, we focus on the problem of detecting anomaly edges in static plain graphs, where the structure is the only available information. This is the basis of several possible extensions such as detecting anomaly edges in dynamic or attribute graphs. A lot of techniques have been proposed in the past decades for anomaly detection in graph structure data. Traditional approaches usually rely on the hand-craft features [4], [5] or the idea of community [6], [7] to manually extract the information of graph structure. Recently, graph embedding based anomaly detection [8] shows superior detection accuracy as graph embedding technique is proved to be effective to encode graph structure information into low-dimensional vector space in an unsupervised fashion. These methods typically consist of two phases. The first phase focus on learning the embedded representation of vertexes and the second phase conducts anomaly detection utilizing general anomaly detection algorithms designed for multi-dimensional data. However, the embedding learned in this way may not be well suited for anomaly detection task since these two phases are two separate models learned with different objective functions. The objective of embedding learning is preserving the local structure of the graph, and the objective of anomaly detection is building the model characterizing normal behavior.

Intuitively, it would be ideal if the embedding learning procedure is integrated into the process of modeling normal behavior to get anomaly oriented embedding and better representation of normal behavior. In this paper, we propose a novel unified and end-to-end framework to jointly learn the embedding of graph structure and the model of normal behavior. For end-to-end we mean that our model takes the test edges along with the original graph as input and directly outputs the anomaly scores for each test edge. Larger anomaly score indicates that the corresponding edge tends to be anomalous. There are three main components of our proposed framework. The first component is a conditional probability distribution designed to unify embedding learning and normality modeling into one same objective function. Inspired by CBOW [9], for a specific vertex, we consider the neighbors of this vertex as context, and we want to know the possibility that there exists an edge between this vertex and each other vertex in the graph. We express this idea with a probability distribution over all the vertexes in the graph conditioned on a specific vertex and its neighbors. This probability distribution can be used as an objective to simultaneously guide the learning of embedding and the modeling of the normality. The learned probability distribution itself can be directly used to detect anomalous edges. The second component of our framework is an end-to-end neural network architecture parameterizing above conditional probability distribution. Generally, this neural network is a mapping from one specific vertex along with its neighbors to the conditional probability distribution. It first maps the vertex and its neighbors into corresponding embedding vectors. Then these embedding vectors are aggregated to get the representation of the context of the vertex. We choose mean as the aggregation function since the order of the

* Corresponding author

978-1-7281-6926-2/20/$31.00 ©2020 IEEE

neighbors is not important. Finally, this representation of the context of the vertex is mapped to the probability distribution over all the vertexes with a fully connected layer followed by a softmax function. The third component is an unsupervised training algorithm designed to learn the parameters from the raw graph data. The learning is achieved by treating one vertex from the neighbors of a specific vertex as the classification target, and this specific vertex along with its rest neighbors as input. To improve training efficiency, we devise a sampling strategy to train the model in a mini-batch fashion.

To summarize, our main contributions are as follows:

- We propose a novel framework unifying the embedding learning and anomalous edge detecting in static plain graph. All the parameters in our framework are optimized jointly and directly for the anomaly detection task.
- We propose an end-to-end neural network architecture to accurately encode the graph structure information for anomalous edge detection.
- We conduct extensive experiments on several commonly used public datasets to compare our method with other state-of-the-art graph anomaly detection methods in static plain graph setting. Experiment results show the superiority of our method in terms of anomaly detection accuracy and scalability.

II. Related Work

The framework we proposed is conceptually related to previous anomaly detection methods in static plain graph, general graph embedding approaches, and anomaly detection in multi-dimensional data based on neural network.

A. Graph anomaly detection

For anomaly detection in static plain graph, the only available information is the structure of the graph. There are plenty of works designed hand-craft features [4], [5] or utilized the idea of community [6], [7]. Recently, with the advancement of graph embedding, several graph anomaly detection methods based on graph embedding have been proposed.

NetWalk [8] is a typical two-stage graph anomaly detection algorithm. It proposes a clique embedding method that minimizes the reconstruction error of sparse autoencoder and pairwise distance to learn graph embedding from random walks. Then the learned embedding is used in clustering algorithms for anomaly detection. It also has an extension to dynamically update the learned representations for dynamic graph detection.

DeepSphere [10] proposes an anomaly detection method for dynamic graphs based on autoencoder. It builds an RNN autoencoder to reconstruct the time series of adjacency matrices and treats reconstruction error as the anomaly score. Although this is a one-stage anomaly detection method, it mainly extracts information in the time dimension, and the structural information of the graph is not well utilized. Thus it is only applicable to dynamic graphs. Besides, taking the adjacency matrix as input means that its input dimension is the square of the number of vertexes. This limits the scalability of the method.

Ding et al. [11] combines graph convolutional network and autoencoder to detect anomalies in attribute networks in a unified fashion. However, their work focus on detecting anomalous nodes in attribute networks while we aim at detecting anomalous edges in plain networks.

B. Graph embedding

Recently, word embedding leads to great progress in the NLP field [12]–[14]. Many recent works employ this idea to learn the embedding of graph [15]–[17]. These methods typically generate a set of paths from the graph with different random walk algorithms. These paths are treated as sentences, and some methods similar to word2vec are utilized to learn the vector representation of the vertex. Node2vec [16] proposes a special random walk algorithm to generate the path, then uses the skip-gram method to learn the embedding of the vertex. It assumes that the interaction between the two vertexes of an edge is symmetrical, but in our model, we assume this is asymmetrical. LINE [18] proposes a novel objective function based on first-order and second-order proximity to learn the representation that preserves both the local and global network structures.

C. Neural network based anomaly detection

There have been many works using neural networks for anomaly detection in multi-dimensional data. These works can be grouped into two categories: The first group is based on the idea of autoencoder [19], [20]. These works use autoencoders to learn the representation of normal data, and the reconstruction errors are used as anomaly scores. The learned autoencoder will have higher reconstruction errors for abnormal samples. The second group uses a neural network to fit a probability distribution over all possible events to predict the normal events [21]. If the actual event is far from the predicted normal events, it is likely to be an abnormality.

III. Methods

In this section, we describe our proposed framework in detail. Our framework focuses on detecting anomalous edges in static plain graphs and assumes there exists a clean training set. We first introduce the conditional probability distribution we proposed to be used as the objective to unify embedding learning and anomaly detection. Then we describe the neural network architecture designed to parameterize this conditional probability distribution. Next, we generate training samples to learn the parameters of the neural network in an unsupervised fashion. Finally, we describe how to use the learned model to detect anomalous edges. We use $G = (V, E)$ to denote a static plain graph where V represents the vertex set and E is the edge set. $|V|$ is the number of vertexes in the graph. (u, v) denotes an edge between two vertexes u and v.

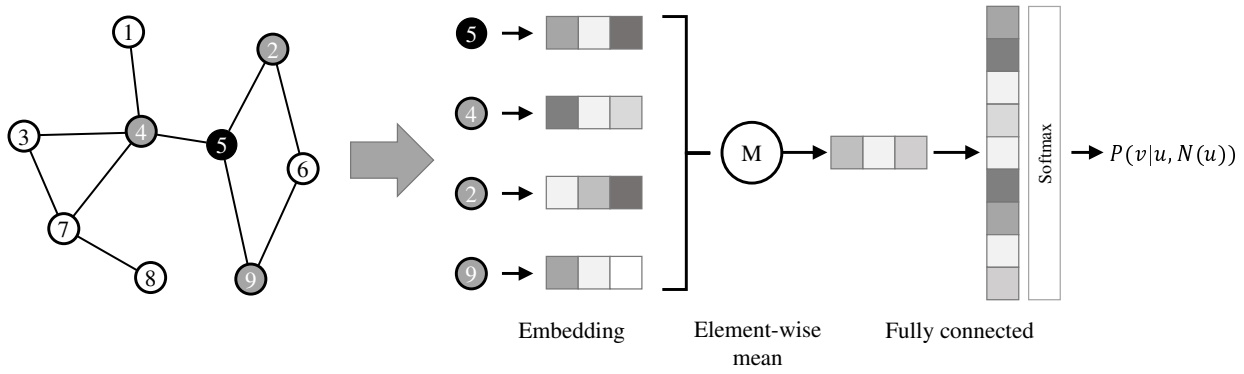

Fig. 1. The end-to-end architecture mapping a specific vertex along with its neighbors to the conditional probability distribution of edges. u and v are vertexes in the graph, and $N(u)$ denotes the neighbors of vertex u. If an edge (u, v) has a low probability $P(v|u, N(u))$, then it can be considered as an anomaly

A. Conditional probability distribution of the edges

According to the definition from [22], an anomaly is a rare instance that deviates significantly from normal behavior. Therefore, modeling the normal behavior accurately is the key to anomaly detection. To detect the anomalous edges in a unified and end-to-end manner, we establish a conditional probability distribution of edges, in which the normal edges have a higher probability of occurrence. In this way, an edge with a lower probability is likely to be an anomaly. Formally, the conditional probability distribution is defined as follows:

Def 1: Let u, v denote two different vertexes in the graph, $N(u)$ denote the neighborhoods of u. We define a conditional probability distribution $P(v|u, N(u))$ over all pairs of u, v in the graph to represent the probability that there exists an edge between u and v.

From the above definition, we can see that the probability distribution conditioned on a specific vertex u and its neighbors. This implies that we define a probability distribution for each vertex, and this distribution will be influenced by neighbors of each vertex. This design choice comes from the prior knowledge about graph structure data: nodes in a densely connected group tend to have similar behaviors. Note that the probability distribution is asymmetrical: each edge (u, v) will appear in two probability distribution $P(v|u, N(u))$, $P(u|v, N(v))$, and $P(v|u, N(u)) \neq P(u|v, N(v))$ since $N(u)$ and $N(v)$ are highly likely to be different. Once we have obtained this distribution representation, we can conduct anomaly detection on the graph structure based on this distribution directly: for an edge (u, v), the smaller the $P(v|u, N(u))$ and $P(u|v, N(v))$, the more likely this edge is anomalous.

B. Representation

We parameterize the conditional probability distribution in Def 1 with a neural network, which learns a function $f(v, u, N(u)) = \hat{P}(v|u, N(u))$. Fig. 1 shows the overall architecture of our proposed model. The input of the model is a specific vertex u along with its neighbors $N(u)$, and the output is a $|V|$ dimensional vector estimating the probability distribution. We decompose this function into three parts:

Part 1: The first part is an embedding layer which is a mapping relationship C that maps each vertex $u \in V$ to a k-dimensional vector $C(u) \in \mathbf{R}^k$, where k is a hyperparameter representing the embedding size. This mapping relationship is represented by a parameter matrix of $|V| \times k$, where the i-th row represents the k-dimensional vector corresponding to vertex i. For each input u and its neighbors $N(u)$, we can get the embedded representation $C(u)$ and $\{C(n)|n \in N(u)\}$.

Part 2: In the second part, we use an aggregate function h to aggregate the embedded representations obtained in the first part. In the NLP problem, there is a sequential relationship between words, but in the graph structure data, the set of neighbors is not ordered. So we prefer the aggregate function to be symmetric, that is, the change in the order of the input element does not change the output. Here we use mean as aggregate function to average the embedded representations:

$$\begin{aligned} H &= h(C(u), \{C(n)|n \in N(u)\}) \\ &= \frac{1}{|N(u)| + 1} \left(C(u) + \sum_{n \in N(u)} C(n) \right) \end{aligned} \tag{1}$$

Notice that the output of the aggregate function is still a k-dimensional vector.

Part 3: In the third phase, we map H to the probability distribution $\hat{P}(v|u, N(u))$ by mapping relation g:

$$\begin{aligned} f(v, u, N(u)) &= \hat{P}(v|u, N(u)) \\ &= g(H)|_v = Softmax(W \cdot H)|_v \end{aligned} \tag{2}$$

This is a fully-connected layer parameterized by matrix W. The input dimension is k, the embedding size. The output dimension is $|V|$, the number of vertexes in the graph. Thus this layer has $k \times |V|$ free parameters. The outputs of the fully connected layer are normalized to be a valid probability distribution by the softmax function. The output of softmax is the estimate of $P(v|u, N(u))$.

Notice that this neural network architecture is end-to-end differentiable, thus all the parameters including the embedding can be jointly optimized for the anomaly detection task. The

learned embedding is more suitable for anomaly detection tasks. More importantly, we get accurate mappings from the local structure of each vertex to the probability distribution of edges. This leads to better anomaly detection performance.

One might have the concern that when the output dimension $|V|$ is large, the neural network will tend to be unstable. However, subsequent experimental results show that the neural network can accurately fit this distribution and conduct anomaly detection at high accuracy even when the number of vertexes is large.

C. Learning

Here we introduce how to learn the parameters of the neural network from the data in an unsupervised fashion. We first introduce the optimization objective, then propose the method to generate training samples from data.

Recall that we expect the learned probability distribution is a good representation of the distribution of normal edges, thus $\hat{P}(v|u, N(u))$ should be large for a normal edge (u, v) from the training set. We use $\theta = \{C, W\}$ to represent all the parameters trainable in our model. The training objective is to find θ that maximizes the following log-likelihood function:

$$L = \frac{1}{|E|} \sum_{(u,v)\in E} \ln \hat{P}(v|u, N(u); \theta) \tag{3}$$

Note that this is essentially a $|V|$-class classification problem. With u and $N(u)$ as inputs, the objective is to predict which vertex may have an edge connected with u. Thus cross-entropy can be used directly as loss function:

$$J = \text{cross-entropy}(f(u, N(u)), v) \tag{4}$$

With this setting, each edge (u, v) along with $N(u) - v$ can form a training sample $(u, N(u) - v) \rightarrow v$ where $(u, N(u) - v)$ is the input and v is the target. However, it is computationally prohibitive to use all the neighbors as context. Concatenate the training samples into batches is crucial for efficient training, but it is not realistic to pad all the samples to the same length as the longest one since the degree distributions typically follow the power law and vary greatly. Therefore, for each edge (u, v) in the graph, we sample s neighbors uniformly with replacement from the set $N(u) - v$ as an approximation of $N(u)$, denoted as $sub(N(u) - v)$. We refer to this subset as **support set**, and s is another hyperparameter of our model. Here, we choose to sample with replacement because, for the vertexes that $|N(u) - v| < s$, in this way we can make sure that $|sub(N(u) - v)| = s$. Since we use mean to aggregate the embeddings of these sampled neighbors, the expectation of the aggregated embeddings of $sub(N(u) - v)$ is equal to the aggregated embeddings of $N(u) - v$ when $|N(u) - v| < s$.

The pseudo-code of the training procedure is shown in Algorithm 1. Note that in practice we usually implement this procedure in mini-batch fashion to improve the training speed.

Algorithm 1 Unsupervised Training Procedure

Input: graph G, embedding size k, size of support set s
Output: $f(u, N(u))$ with learned $\theta = \{C, W\}$
Randomly initialize $\theta = \{C, W\}$
for each u in graph G **do**
  $N \leftarrow$ the set of vertexes adjacent to u
  for each v in N **do**
    $S \leftarrow$ sample s vertexes from $N - v$ with replacement
    Perform a forward pass to compute $f(u, S)$
    $J(\theta) \leftarrow$ cross-entropy$(f(u, S), v)$
    Perform back propagation to compute $\nabla_\theta J(\theta)$
    Update θ with $\nabla_\theta J(\theta)$
  end for
end for

D. Inference

With the learned estimation of the probability distribution, we can determine whether an edge is normal or not by investigating its probability of existence. Consider an edge (u, v), we can estimate the probability of existence from the perspective of u and v respectively:

$$\begin{aligned} p_v &= \hat{P}(v|u, N(u)) \\ p_u &= \hat{P}(u|v, N(v)) \end{aligned} \tag{5}$$

Note that in the inference phase, we use all the neighbors as context, unlike the training phase where we sample a subset of neighbors as context. According to the definition of our probability distribution, the smaller the probability, the more likely this edge is anomalous. We can combine these two probabilities with an aggregate function A to get the anomaly score:

$$score = A(1 - P(v|u, N(u)), 1 - P(u|v, N(v))) \tag{6}$$

There are many options for the aggregate function A here, such as mean, max, min. Intuitively, mean will be effective for most scenario since it is more stable.

E. Algorithm complexity analysis

There are two groups of free parameters in our model: $\theta = \{C, W\}$. Recall that C is a $|V| \times k$ matrix and W is a $k \times |V|$ matrix, therefor our model has a total number of $2 \times k \times |V|$ free parameters. Our model only has two hyperparameters, which are the embedding size and the size of the support set. This makes our model easy to fine-tune for specific datasets.

IV. Experiments

In this section, we conduct extensive experiments to evaluate the proposed method. First, we introduce the settings of the experiments, including the datasets used for evaluation, the baseline methods to compare, the implementation details of our method, the metrics, and the generation of ground truth. Then we compare the anomaly detection performance of our proposed method with other state-of-the-art anomaly detection

methods. Finally, we test the stability of our method and explore the sensitivity of the hyperparameters.

A. Experiment Settings

1) Datasets: We conduct evaluations on six publicly available graph datasets.

- **UCI messages.** This dataset contains online messages between students from University of California, Irvine [23]. The vertexes represent students and edges denote messages between students.
- **arXiv hep-th.** This dataset reflex the collaboration relationship between authors of research papers from the arXiv's High Energy Physics-Theory (hep-th) section [24]. The vertexes represent authors, and edges indicate their collaboration for a common paper.
- **Digg.** This dataset collects from the social news website Digg [25]. Each vertex is a user on the website, and each edge is a reply from one user to another.
- **DBLP co-authorship.** This is a research collaboration network extracted from DBLP computer science bibliography [26]. The vertexes are authors and edges show the collaboration between authors.
- **email-Eu-core.** This is a email communication network from a large European research institution [24]. The vertexes are members of the institution and edges represent the emails sent between them. This dataset has ground truth label of the groups.
- **Enron email.** This is a email communication network in a large corporation Enron. The Federal Energy Regulatory Commission released this dataset during its investigation [27]. The vertexes are employees and edges show the email communication between them.

The statistics of these six datasets are summarized in Table I. The first four datasets are utilized to evaluate the accuracy and scalability of our method, and the last two datasets are used to evaluate the effectiveness of our method on graphs that have special properties.

These datasets have no ground truth anomalies which is well-known challenging to obtain. To quantitatively compare the anomaly detection methods, we choose to manually inject anomaly edges into these datasets with anomaly injection technique [8], [22]. For the first four datasets, we divide each datasets into two parts where the first half of edges are used for training, and the rest edges are reserved as the test set. Then 1% or 5% anomaly edges are injected into the test sets to build two groups of test sets. In order to evaluate our method in areas with different densities, we build four datasets where anomalous edges are injected in areas with different densities. For the email-Eu-core dataset with ground truth groups, we randomly insert 1% anomaly edges between nodes in different groups to simulate the malicious emails sent by attackers such as spear phishing emails. In this scenario, attackers tend to use the identities of the insiders as the sender of emails to trick the victims. We refer to this dataset as "Groups". For most graphs, there are dense area where nodes have high degrees and sparse area where nodes have low degrees. We'd like to know whether our method can learn good representations and have decent anomaly detection performance in both dense areas and sparse areas. Enron email dataset is utilized to build three different test sets. We sort the nodes by the degrees, then we select the top 25% and bottom 25% nodes as "High-Deg" and "Low-Deg" respectively. We randomly inject 1% anomalous edges between "High-Deg" and "High-Deg" nodes to test the performance of our method in dense areas, and 1% anomalous edges between "Low-Deg" and "Low-Deg" nodes for sparse areas. We also inject anomalous edges between "High-Deg" and "Low-Deg" which is the intersection of dense areas and sparse areas. We refer to these three datasets as: "Low-Deg, Low-Deg", "Low-Deg, High-Deg" and "High-Deg, High-Deg".

TABLE I
THE STATISTICS OF THE SIX PUBLIC AVAILABLE DATASETS USED IN THE EXPERIMENTS.

Dataset	#Vertex	#Edge	Max Degree	Avg Degree
UCI messages	1899	13838	255	14.57
arXiv hep-th	9877	25998	65	5.26
Digg	30398	86312	285	5.67
DBLP	317080	1049866	343	6.62
email-Eu-core	1005	16706	347	33.24
Enron email	36692	183831	1383	10.02

2) Performance Metric: We choose AUC(area under curve) as the primary metric to compare the predictive power of each method. AUC is a classification-threshold-invariant metric that aggregates the performance of the model across all possible thresholds. For a specific application, a threshold can be chosen to trade between precision and recall.

3) Compared Methods: We compare our method with several state-of-the-art methods. The first method is NMF(non-negative matrix factorization). NMF is a low-rank approximation algorithm that can be applied on graph adjacency matrix to find the inherent communities. Edges associated with large residual values are more likely to be anomalies. The second group of methods based on graph embedding methods. We follow the NetWalk [8] to utilize unsupervised clustering methods to conduct anomaly detection upon the representation learned by these graph embedding methods. NetWalk [8] is the state-of-the-art graph embedding method for anomaly detection on dynamic plain graphs. It proposes a novel clique embedding technique to accurately learn the embedding of the graph. The learned embedding is clustered to find the anomalies. It also has an extension to dynamically update the embedding and conduct anomaly detection in dynamic graphs. In this work, we compare with its performance on static graphs only for completeness as this method is mainly designed for dynamic graph which is different to our scenario. We also compare our method with other four methods. The learned representation from these methods are used for anomaly detection in the way similar to NetWalk. The purpose of including these methods is to demonstrate the magnitude of the improvements

TABLE II
ANOMALY DETECTION PERFORMANCE COMPARISON

Methods	UCI Messages		arXiv hep-th		Digg		DBLP	
	1%	5%	1%	5%	1%	5%	1%	5%
Spectral Clustering	0.6367	0.6212	0.6241	0.6183	0.6069	0.5915	0.6178	0.6191
DeepWalk	0.7455	0.7432	0.7414	0.7152	0.7070	0.6997	0.7373	0.7289
node2vec	0.7471	0.7490	0.7391	0.7201	0.7236	0.7111	0.7423	0.7245
SDNE	0.7356	0.7236	0.7302	0.7086	0.7133	0.7009	0.7289	0.7120
NetWalk	0.7813	0.7761	0.7436	0.7226	0.7591	0.7213	0.7598	0.7351
NMF	0.8454	0.8660	0.7588	0.7637	0.7261	0.7298	0.7394	0.7374
Our Method	**0.8551** $\pm$0.0073	**0.8730** $\pm$0.0033	**0.7739** $\pm$0.0029	**0.7735** $\pm$0.0075	**0.8030** $\pm$0.0043	**0.8082** $\pm$0.0018	**0.8315** $\pm$0.0013	**0.8302** $\pm$0.0008

TABLE III
ANOMALY DETECTION PERFORMANCE IN AREAS WITH DIFFERENT DENSITIES

Dataset	NMF	Our method
Groups	0.9166	$\mathbf{0.9491} \pm 0.0093$
Low-Deg, Low-Deg	0.9131	$\mathbf{0.9567} \pm 0.0011$
Low-Deg, High-Deg	0.8890	$\mathbf{0.9095} \pm 0.0018$
High-Deg, High-Deg	0.7740	$\mathbf{0.8281} \pm 0.0020$

of our method over NetWalk. Spectral Clustering [28]: Spectral Clustering learns the representations of vertexes by calculating the eigenvectors of the Graph Laplacian.

DeepWalk [15]: DeepWalk is one of the first work on utilizing the idea of word embedding to generate the representation of graph structure.

node2vec [16]: node2vec learn the representation of graph structure similarly with DeepWalk. The main advantage of node2vec is the novel random walk algorithm.

Structural Deep Network Embedding (SDNE) [29]: SDNE proposes a representation learning method based on deep autoencoder to preserve the network structure.

4) Implementation Details: We implement the proposed method with PyTorch and run all the experiments on a GPU with 11GB memory. The hyperparameters of our methods can be divided into two groups. The first group contains hyperparameters of the model, including embedding size and support set size. The second group is hyperparameters of the optimization procedure, which contains the choice of the optimization algorithm, batch size, learning rate, and the number of epochs. For all the experiments, we use the same hyperparameters of the optimization procedure. We choose Adam as the default optimization algorithm, and the batch size is set to 64. Each model is trained for 3 epochs and the learning rate is 0.01.

B. Performance Comparison

In this section, we compare the performance of our proposed method with NMF and other baseline methods.

There are several kinds of randomness in our experiments: 1) Datasets are randomly divided into train and test set; 2) The randomly generated anomaly edges; 3) The randomly initialized parameters of our neural network model. Due to these inherent variabilities, we have to make sure that the performance improvement of our method is consistent. Thus we run our method 10 times with different random seed for each dataset and report the mean, max, min and std to reliably evaluate the performance of our method.

The size of the support set and the sample rate are fixed at 3 and 0 in all the experiments for performance comparison. Note that a zero sample rate means for each edge we only sample once from the neighbors of the vertex to construct one training sample. The choice of embedding size is specific to each dataset. We set the embedding size to 4, 32, 256, 32 for UCI Messages, arXiv hep-th, Digg and DBLP respectively.

The AUC scores of all the experiments are listed in Table II. The bottom four lines report the statistic value of the results of the ten runs of our method. From the table we can observe the **main result** that our proposed method consistently outperforms NMF by a large margin on all four datasets both with 1% and 5% anomalies. Note that the improvements in Digg and DBLP are more significant. This implies that our method has good scalability. Also, note that our method is more robust as the number of anomalous edges increased. The AUC scores of our method in the test set with 5% anomalies are similar to the scores in 1% setting, while most of the baseline methods including NetWalk perform slightly worse in 5% anomalies setting than in 1% anomalies setting. The last row of the table shows that the stand variances of the ten runs of our method on each dataset are low, which is a proof that our method is stable and the performance improvement is consistent. As is said in the previous section, one might have the concern that a large number of classes for the classification model will lead to instability and inferior accuracy. However, from the table, we can see that our proposed method achieves a significantly better AUC score on Digg and DBLP dataset which has over three hundred thousand vertexes. The improvement of our method over NMF in DBLP is even larger than in arXiv hep-th and UCI Messages. This result demonstrates that our proposed model is capable of learning the accurate representation of the conditional probability distribution of edges when the number of vertexes is large.

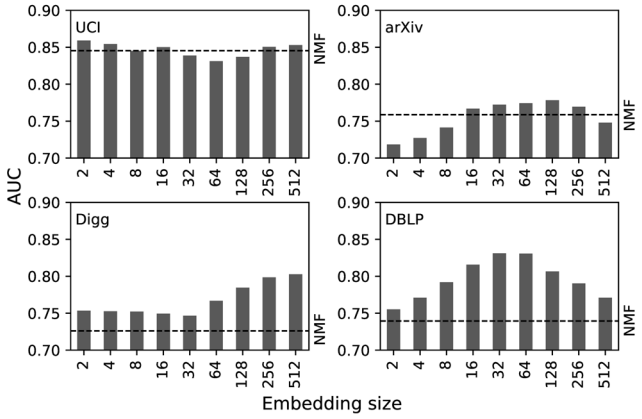

Fig. 2. Accuracy of anomaly detection with different embedding size k.

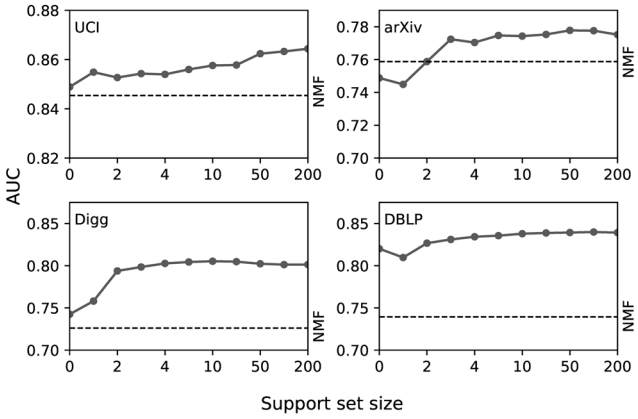

Fig. 3. Accuracy of anomaly detection with different size of support set s, where $s = 0$ represents that we don't aggregate the information from neighbors.

C. Performance in Areas With Different Densities

In this section we evaluate the performance of our method in areas with different densities. One might have the concern that our method will need lots of training samples, thus it may perform well in dense area and badly in sparse area. However, the experiment results listed in Table III demonstrate that our method outperform NMF by a large margin both in sparse areas and dense areas.

D. Hyperparameter Sensitivity

In this section, we explore the hyperparameter space of our proposed model to show that our model is stable for different hyperparameters. Besides, there are only two hyperparameters in our model: embedding size and support set size. Thus, our model is easy to adapt for specific dataset and will has less risks of overfitting.

1) Embedding Size: We first examine the different choices of embedding size. We vary the embedding size from 2^1 to 2^9. The sample rate is set to 0, which means for each edge we only sample once to generate one training sample. This is the most training efficient setting. The size of the support set is fixed to 3, which is a rather arbitrary choice. Intuitively, the suitable choice of embedding size depends on the number of vertexes as more vertexes will require larger embedding space to encode the structure information. Experiment results summarized in Fig. 2 confirm this intuition. For small dataset such as UCI Messages that only has no more than two thousand vertexes, embedding size as small as 2 is enough to achieve decent AUC score. In datasets that have more vertexes, we can observe the trend that larger embedding size typically leads to a better AUC score. However, notice that for arXiv hep-th and DBLP, keep increasing the embedding size after a certain threshold will damage the model performance, and there is no similar phenomenon for UCI messages and Digg. We argue that this is caused by the different degree distributions of these datasets. Recall that the number of free parameters for each vertex in our model is twice the dim of embedding, and the number of effective training samples for each vertex is proportional to the degree. Specifically, the degree distribution of Digg is more skewed than the distributions in other datasets. Since embedding size has important impacts on the accuracy of the model and is closely related to the properties of the specific dataset, it is the most important hyperparameter and requires careful adjustment for each dataset.

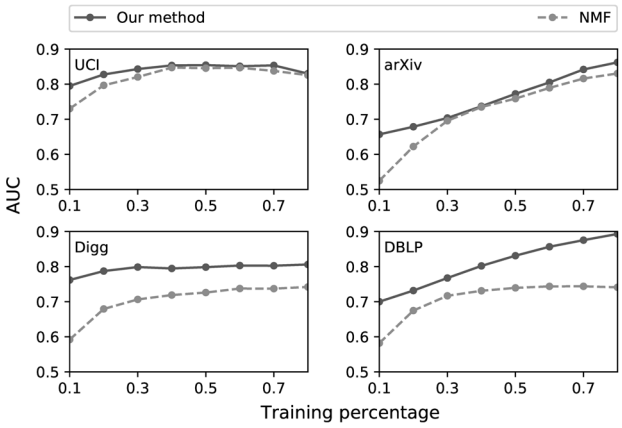

Fig. 4. Accuracy of anomaly detection using different percentage of dataset as training set.

2) Size of Support Set: We vary the size of support set from 0 to 200 to evaluate their impacts on the performance. Note that setting the size of support set to 0 means that we no longer consider the neighbors, and the learning objective essentially degenerates into $\hat{P}(v|u)$. Experiment results are depicted in Fig. 3. It is evident that: 1) aggregating the information from neighbors can significantly improve the accuracy of the model; 2) larger support set consistently leads to better model performance, but the improvement is relatively small. Besides, a larger support set requires more memory and computation. Thus, in most scenarios, a small support set size $(2 < s < 5)$ is enough to achieve satisfying accuracy.

3) Training Percentage: By varying the training percentage, we can gain more insight into the performance of the proposed model. Fig. 4 shows the AUC scores of the proposed model and NMF on four datasets with different training percentages

from 0.1 to 0.8. It can be seen from the figure that our model can learn a decent representation of the graph structure even with a small number of training samples. In UCI Messages and Digg, our model achieve decent accuracy even trained with only 30% of the training set. But in arXiv hep-th and DBLP we can observe a steady improvement of the AUC as the training percentage increased. We speculate that this difference also comes from the different degree distributions of these datasets.

V. Conclusions

We have presented our framework unifying the embedding learning and anomaly detecting. We first propose a conditional probability distribution used as an objective function to simultaneously guide the learning of embedding and the modeling of normal behavior for anomaly detection. Then we parameterize this probability distribution with an end-to-end neural network that leveraged the prior knowledge of the graph structure. Finally, we design a training algorithm to efficiently learn the parameters of the neural network in an unsupervised fashion. Experiments show that our model outperforms the state-of-the-art NetWalk by a large margin in terms of accuracy.

Acknowledgment

This work is supported by the National Key Research and Development Program of China with Grant No. Y950121201. The corresponding author is Yongzheng Zhang.

References

[1] Q. Ding, N. Katenka, P. Barford, E. Kolaczyk, and M. Crovella, "Intrusion as (anti) social communication: characterization and detection," in *Proceedings of the 18th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 2012, pp. 886–894.

[2] M. Ott, C. Cardie, and J. Hancock, "Estimating the prevalence of deception in online review communities," in *Proceedings of the 21st international conference on World Wide Web*. ACM, 2012, pp. 201–210.

[3] X. Hu, B. Li, Y. Zhang, C. Zhou, and H. Ma, "Detecting compromised email accounts from the perspective of graph topology," in *Proceedings of the 11th International Conference on Future Internet Technologies*. ACM, 2016, pp. 76–82.

[4] L. Akoglu, M. McGlohon, and C. Faloutsos, "Oddball: Spotting anomalies in weighted graphs," in *Pacific-Asia Conference on Knowledge Discovery and Data Mining*. Springer, 2010, pp. 410–421.

[5] C. C. Aggarwal, "Outlier ensembles: position paper," *ACM SIGKDD Explorations Newsletter*, vol. 14, no. 2, pp. 49–58, 2013.

[6] J. Sun, H. Qu, D. Chakrabarti, and C. Faloutsos, "Neighborhood formation and anomaly detection in bipartite graphs," in *Fifth IEEE International Conference on Data Mining (ICDM'05)*. IEEE, 2005, pp. 8–pp.

[7] H. Tong and C.-Y. Lin, "Non-negative residual matrix factorization with application to graph anomaly detection," in *Proceedings of the 2011 SIAM International Conference on Data Mining*. SIAM, 2011, pp. 143–153.

[8] 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 *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. ACM, 2018, pp. 2672–2681.

[9] T. Mikolov, K. Chen, G. Corrado, and J. Dean, "Efficient estimation of word representations in vector space," *arXiv preprint arXiv:1301.3781*, 2013.

[10] X. Teng, M. Yan, A. M. Ertugrul, and Y.-R. Lin, "Deep into hypersphere: Robust and unsupervised anomaly discovery in dynamic networks." in *IJCAI*, 2018, pp. 2724–2730.

[11] K. Ding, J. Li, R. Bhanushali, and H. Liu, "Deep anomaly detection on attributed networks," in *Proceedings of the 2019 SIAM International Conference on Data Mining*. SIAM, 2019, pp. 594–602.

[12] Y. Bengio, R. Ducharme, P. Vincent, and C. Jauvin, "A neural probabilistic language model," *Journal of machine learning research*, vol. 3, no. Feb, pp. 1137–1155, 2003.

[13] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean, "Distributed representations of words and phrases and their compositionality," in *Advances in neural information processing systems*, 2013, pp. 3111–3119.

[14] J. Pennington, R. Socher, and C. Manning, "Glove: Global vectors for word representation," in *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, 2014, pp. 1532–1543.

[15] B. Perozzi, R. Al-Rfou, and S. Skiena, "Deepwalk: Online learning of social representations," in *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 2014, pp. 701–710.

[16] A. Grover and J. Leskovec, "node2vec: Scalable feature learning for networks," in *Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 2016, pp. 855–864.

[17] R. Trivedi, M. Farajtabar, P. Biswal, and H. Zha, "Dyrep: Learning representations over dynamic graphs," in *International Conference on Learning Representations*, 2019. [Online]. Available: https://openreview.net/forum?id=HyePrhR5KX

[18] J. Tang, M. Qu, M. Wang, M. Zhang, J. Yan, and Q. Mei, "Line: Large-scale information network embedding," in *Proceedings of the 24th international conference on world wide web*. International World Wide Web Conferences Steering Committee, 2015, pp. 1067–1077.

[19] C. Zhou and R. C. Paffenroth, "Anomaly detection with robust deep autoencoders," in *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, 2017, pp. 665–674.

[20] B. Zong, Q. Song, M. R. Min, W. Cheng, C. Lumezanu, D. Cho, and H. Chen, "Deep autoencoding gaussian mixture model for unsupervised anomaly detection," in *International Conference on Learning Representations*, 2018. [Online]. Available: https://openreview.net/forum?id=BJJLHbb0-

[21] Y.-L. Kong, Q. Huang, C. Wang, J. Chen, J. Chen, and D. He, "Long short-term memory neural networks for online disturbance detection in satellite image time series," *Remote Sensing*, vol. 10, no. 3, p. 452, 2018.

[22] 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.

[23] T. Opsahl and P. Panzarasa, "Clustering in weighted networks," *Social networks*, vol. 31, no. 2, pp. 155–163, 2009.

[24] J. Leskovec, J. Kleinberg, and C. Faloutsos, "Graph evolution: Densification and shrinking diameters," *ACM Transactions on Knowledge Discovery from Data (TKDD)*, vol. 1, no. 1, p. 2, 2007.

[25] M. De Choudhury, H. Sundaram, A. John, and D. D. Seligmann, "Social synchrony: Predicting mimicry of user actions in online social media," in *2009 international conference on computational science and engineering*, vol. 4. IEEE, 2009, pp. 151–158.

[26] J. Yang and J. Leskovec, "Defining and evaluating network communities based on ground-truth," *Knowledge and Information Systems*, vol. 42, no. 1, pp. 181–213, 2015.

[27] J. Leskovec, K. J. Lang, A. Dasgupta, and M. W. Mahoney, "Community structure in large networks: Natural cluster sizes and the absence of large well-defined clusters," *Internet Mathematics*, vol. 6, no. 1, pp. 29–123, 2009.

[28] U. Von Luxburg, "A tutorial on spectral clustering," *Statistics and computing*, vol. 17, no. 4, pp. 395–416, 2007.

[29] D. Wang, P. Cui, and W. Zhu, "Structural deep network embedding," in *Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 2016, pp. 1225–1234.