2018 IEEE International Conference on Data Mining

Deep Structure Learning for Fraud Detection

Haibo Wang*[‡§], Chuan Zhou[†], Jia Wu[¶], Weizhen Dang*[‡§], Xingquan Zhu[‖], Jilong Wang[‡§]
*Department of Computer Science and Technology, Tsinghua University
[†]Institute of Information Engineering, Chinese Academy of Sciences, Beijing, China
[‡]Institute for Network Sciences and Cyberspace, Tsinghua University
[§]Tsinghua National Laboratory for Information Science and Technology
[¶]Department of Computing Faculty of Science and Engineering, Macquarie University, Sydney, Australia
[‖] Dept. of Computer & Electrical Engineering and Computer Science, Florida Atlantic University, USA
Email: {wang-hb15, dangwz16}@mails.tsinghua.edu.cn, zhouchuan@iie.ac.cn, jia.wu@mq.edu.au,
xzhu3@fau.edu, wjl@cernet.edu.cn

***Abstract*—Fraud detection is of great importance because fraudulent behaviors may mislead consumers or bring huge losses to enterprises. Due to the lockstep feature of fraudulent behaviors, fraud detection problem can be viewed as finding suspicious dense blocks in the attributed bipartite graph. In reality, existing attribute-based methods are not adversarially robust, because fraudsters can take some camouflage actions to cover their behavior attributes as normal. More importantly, existing structural information based methods only consider shallow topology structure, making their effectiveness sensitive to the density of suspicious blocks. In this paper, we propose a novel deep structure learning model named DeepFD to differentiate normal users and suspicious users. DeepFD can preserve the non-linear graph structure and user behavior information simultaneously. Experimental results on different types of datasets demonstrate that DeepFD outperforms the state-of-the-art baselines.**

***Keywords*-Fraud Detection, Density Block, Graph Structure Learning, Behavior Similarity**

I. Introduction

Fraudulent behaviors, such as shilling attacks and network attacks, have become widespread because they can bring huge profit to fraudsters [1]. A typical fraudulent behavior pattern is that fraudsters manipulate a large number of accounts or IP addresses to rate or click some given items (*e.g.*, products, websites). For example, fake reviews on ecommerce websites could mislead consumers to buy defective products, and network attacks could reduce the processing capacity of the servers in enterprises. Therefore, it is of great significance for shopping websites and enterprises to detect the fraudulent behaviors effectively for improving their service.

The interaction between users and items can be modeled as a bipartite graph with users being source nodes and items being sink nodes. Due to the lockstep feature of fraudulent behaviors, the fraud detection problem can be treated as finding suspicious dense blocks in the attributed bipartite graph. Most previous works [2]–[4] aim at finding these dense blocks by exploiting the attribute information like timestamp and review. They assume that fraudulent activities tend to be concentrated in time and reviews for items given by fraudsters are similar.

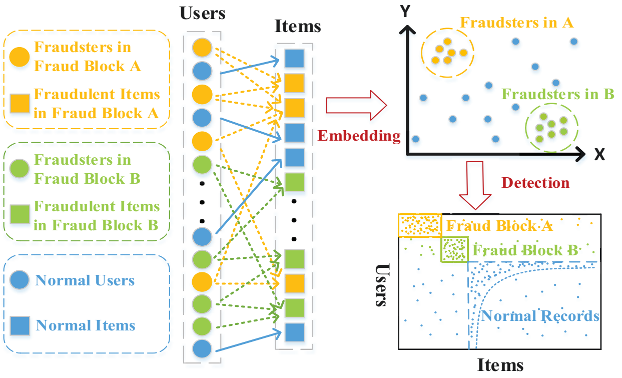


Figure 1. Deep Structure Learning for Fraud Detection. Blue circles and squares represent normal users and items. Yellow/Green ones represent fraudsters and corresponding fraudulent items in one/another fraud block.

In reality, these attribute-based methods for fraud detection are not adversarially robust, because deliberate fraudsters can take some camouflage actions to evade the detection. For instance, suspicious users may intentionally make differential rating scores or reviews on the targeted items, and network attackers could conduct long-term and low-rate distributed denial-of-service (DDos) attacks to make themselves look normal [5]. What's more, the attribute information in bipartite graphs is partially missing or even unavailable in many application scenarios, which makes it difficult for these attribute-based methods to be widely used.

In contrast, the structure information of bipartite graphs can be easily accessible with low statistical noise, and cannot be easily camouflaged. Each fraudulent behavior inevitably generates an edge in the corresponding bipartite graph, which is unable to be cleared away and hidden from view. In other words, the topology of the bipartite graph contains a complete set of structure information of suspicious behaviors. Hence, an important open problem is: whether and to what extent we can address the fraud detection problem if only structure information of the bipartite graph is available.

Existing approaches that directly work on structure information, such as maximizing the arithmetic or geometric

978-1-5386-9159-5/18/$31.00 ©2018 IEEE
DOI 10.1109/ICDM.2018.00072

degree [6]–[9], only consider shallow topology structure, making them sensitive to the density of suspicious blocks. Despite of existing research efforts in the field, we are still missing effective and robust approaches that can make deep structure learning of the bipartite graph for fraud detection problem.

In this paper, we propose a novel fraud detection method, DeepFD, which embeds all user nodes into a latent space by a deep structure learning method. Our goal is to make the representations of the suspicious users in the same fraud block as close as possible, while the representations of the normal users distribute uniformly in the remaining latent space, so fraud blocks can be detected accurately by density-based detection methods. The intuitive idea is shown in Fig. 1. DeepFD can simultaneously preserve the global non-linear graph structure and behavior differences between diverse users. After obtaining embedding results, fraud blocks can be easily detected by the position distribution of user nodes in the latent space. Our method only makes use of graph topology information without attribute information. Additionally, DeepFD is featured to automatically find multiple fraud blocks without predefining the block number. We conduct the experiments on three semi-real synthetic datasets and one real-world DDos attack dataset. The results show that our method outperforms the state-of-the-art competitors.

The main contributions can be summarized as follows:

(1) To the best of our knowledge, we are among the first to detect fraudulent behaviors based on deep structure learning with the intuitive comprehensibility.

(2) We propose a novel deep network embedded fraud detection model, DeepFD, where the graph topology information is well exploited. The model can simultaneously preserve global non-linear graph structure and user behavior information. As a result, the embedding results are applicable to differentiate normal users and suspicious users.

(3) Experimental results on three semi-real synthetic datasets and one real-world network attack dataset demonstrate the effectiveness of the proposed DeepFD model. Moreover, the model is robust in automatically detecting multiple fraud blocks without predefining the block number, which cannot be achieved by the state-of-the-art baselines.

II. Problem Definition

In this section, we formally define deep network embedded fraud detection in an interaction information graph.

Definition 1: (Interaction Information Graph) An interaction information graph is a special form of a bipartite graph, which is defined as $G = (X, Y, E)$, where $X = \{x_1, ..., x_m\}$ represents m user nodes, $Y = \{y_1, ..., y_n\}$ represents n item nodes and $E = \{e_{ij}\}_{i=1,...,m}^{j=1,...,n}$ represents directed edges from X to Y. If there exits an edge from x_i to y_j, $e_{ij} = 1$. Otherwise, $e_{ij} = 0$. Note that X and Y are two disjoint sets.

For example, when users purchase products or visit websites, the user-item relationships can form an interaction information graph. In order to detect the fraudulent behaviors in the interaction information graph accurately, we expect that the deep network embedded fraud detection method can embed all user nodes into a low-dimensional latent space, and further differentiate normal and suspicious users by their position relationship in the latent space. Because fraudsters inevitably generate more suspicious edges in the interaction information graph, the topology structure is the most straightforward signal that reflects the fraudulent behaviors. However, due to the camouflage actions of fraudsters ($e.g.$, fraudsters buy some popular normal products), shallow and local topology structure may be not sufficient. We need to further capture the different behaviors of user nodes. As we have explained above, the embedding of user nodes in the interaction information graph should preserve graph structure information and users' behavioral differences simultaneously.

The graph structure information can be defined as the connections between user nodes and item nodes, and the differences of behavior across user nodes can be described by the following similarity metric.

Definition 2: (Similarity Metric) Given two different user nodes x_i and x_j in the interaction information graph, the similarity metric between them can be defined as $sim_{ij} = \frac{|N_i \cap N_j|}{|N_i \cup N_j|}$, where $N_i := \{y_j \in Y : e_{ij} = 1\}$ represents the item node set associated with the user node x_i.

Intuitively, if two user nodes share a lot of item nodes, they tend to have a large similarity metric. In practice, for the purpose of fraud, suspicious user nodes inevitably associate with more of the same item nodes so that the similarity between them is relatively higher, while the behaviors of normal user nodes are independent, which leads to low similarity in general. Therefore, the similarity metric is effective to capture the differences of behavior across diverse user nodes.

For two user nodes x_i and x_j, there may exist two extreme cases: (1) the sets N_i and N_j do not share any common element ($i.e.$, $|N_i \cap N_j| = 0$); (2) the set N_i is identical to N_j, which implies that $|N_i \cap N_j| = |N_i \cup N_j|$. In order to make the similarity metric more robust, we add a smoothing term into it. The improved similarity metric is shown as follows:

$$sim_{ij} = \begin{cases} \frac{|N_i \cap N_j|+1}{|N_i \cup N_j|+n} & |N_i \cap N_j| = \varnothing, \\ \frac{|N_i \cap N_j|+(n-1)}{|N_i \cup N_j|+n} & |N_i \cap N_j| = |N_i \cup N_j|), \\ \frac{|N_i \cap N_j|}{|N_i \cup N_j|} & otherwise. \end{cases}$$

Here n is the total number of the item nodes.

With the above definitions, we further formally define the problem of deep network embedded fraud detection.

Definition 3: (Deep Network Embedded Fraud Detection) Given an interaction information graph $G = (X, Y, E)$, deep

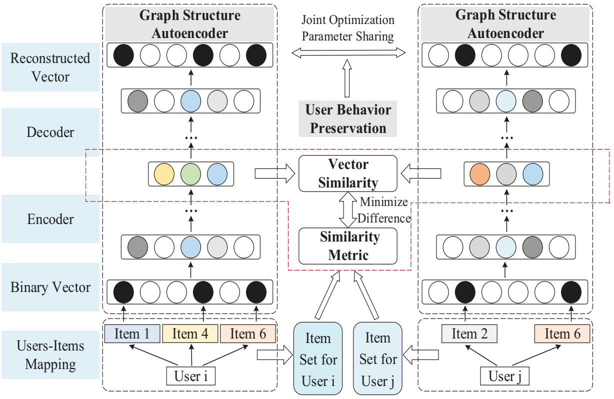


Figure 2. The deep embedding framework of DeepFD model

network embedded fraud detection first learns a mapping function $f : X \rightarrow \mathbb{R}^d$, where $d \ll |Y|$. The mapping function should simultaneously preserve the graph structure information and differences of behavior across diverse user nodes. After that, all fraud blocks in the graph can be detected automatically based on the position distribution of user nodes in a low-dimensional latent space.

III. Our Solution: DeepFD Model

In order to detect fraudulent behaviors, DeepFD proposes to model user behaviors in a latent space, and classifies users based on their latent space feature distributions. The deep structure learning framework of DeepFD is shown in Fig. 2, which mainly consists of two components. The first component aims to reconstruct the original graph structure by vector representations of user nodes. The second component preserves different behaviors among diverse users. By jointly optimizing both two components, the embedding results can preserve global graph structure information and user behavior characteristics simultaneously, which will be used for the fraud detection by density-based detection methods. In the following parts, we will introduce how to realize and optimize the model in details.

A. Graph Structure Reconstruction

The structure of the interaction information graph refers to the connections between user nodes and item nodes. As shown in Fig. 2, to preserve the global non-linear graph structure information, we introduce a deep auto-encoder to model the relationship between user nodes and item nodes. The encoder part consists of multiple non-linear transformation layers to map the connections for each user node into a low-dimensional space. The decoder part also composes of several non-linear functions to reconstruct original graph structure by vector representations of user nodes.

Given an interaction information graph $G = (X, Y, E)$, the graph structure can be expressed as $S = \{s_1, s_2, ..., s_m\}$, where $s_i = \{s_{ij}\}_{j=1}^{n}$, $s_{ij} = 1$ if there exists a directed edge from x_i to y_j, m is the number of user nodes and n is the number of item nodes. Taking s_i as input of the auto-encoder, we can get the hidden vector representations for the corresponding non-linear transformation layers in the encoder part, which are shown as follows:

$$\begin{aligned} g_i^{(1)} &= \sigma(W^{(1)} s_i + b^{(1)}) \\ g_i^{(l)} &= \sigma(W^{(l)} g_i^{(l-1)} + b^{(l)}), l = 2, ..., K \end{aligned} \tag{1}$$

where K is the number of layers in the encoder, and σ is the activation function. Similarly, feeding user node representation $g_i^{(K)}$ into the decoder will produce reconstructed vector output $\hat{s_i}$ by another K non-linear mappings. In this work, we use the *ReLu* activation function in both encoder and decoder parts. Our goal is to minimize the reconstruction error, $i.e.$, minimize the distance between the original input s_i and the reconstructed output $\hat{s_i}$. Therefore, the reconstructed loss function is designed as follows:

$$\mathcal{L}_{tmp} = \sum_{i=1}^{m} ||\hat{s_i} - s_i||_2^2 \tag{2}$$

However, there may exist some issues with the reconstruction based loss function. In practice, both suspicious and normal user nodes tend to be associated with a small portion of item nodes, leading to the fact that the interaction information graph is extremely sparse. Because the reconstructed loss function in Eq. (2) treats all elements of the input vector s_i equally and the number of zero elements in s_i is far more than that of non-zero elements, the auto-encoder is more likely to reconstruct zero elements. For fraud detection, the item nodes that are associated with a given user node are more effective to reflect the suspicious behaviors, but this information is not considered in Eq. (2). To solve the problem, we revise the loss function by setting larger weights to non-zero elements, which is shown as follows:

$$\begin{aligned} \mathcal{L}_{recon} &= \sum_{i=1}^{m} ||(\hat{s_i} - s_i) \odot h_i||_2^2 \\ &= ||(\hat{S} - S) \odot H||_2^2 \end{aligned} \tag{3}$$

where $\odot$ is the Hadamard product, $\hat{S} = \{\hat{s_1}, \hat{s_2}, ..., \hat{s_m}\}$, $H = \{h_1, h_2, ..., h_m\}$ and h_i is the weight vector for the input vector s_i. For $h_i = \{h_{ij}\}_{j=1}^{n}$, if $s_{ij} = 0$, $h_{ij} = 1$; otherwise, $h_{ij} = \beta > 1$.

By minimizing the loss function of the revised model, the vector representations generated by the K_{th} layer of the encoder for all user nodes preserve the global non-linear graph structure information. The deep auto-encoder mainly describes the relationship between user nodes and item nodes. In order to detect fraudulent behaviors accurately, we need to further capture the different behaviors across diverse user nodes, which will be introduced in the following part.

B. User Behavior Preservation

As we have explained, the behaviors of suspicious user nodes in the same fraud block tend to be similar while the behaviors of normal user nodes tend to be independent. We expect that the vector representations of user nodes can preserve the traits, which will be useful to detect fraud blocks. In the low-dimensional latent space, the behavioral differences between two user nodes can be described by the distance of their vector representations.

For user node x_i and user node x_j, the distance measure of their vector representations is defined as follows:

$$dis_{ij} = ||(g_i^{(K)} - g_j^{(K)})||_2^2 \tag{4}$$

where $g_i^{(K)}$ is the vector representations of user node x_i, which is defined in Section III-A. In order to convert the distance measure into the similarity measure of vector representations, we further employ a mapping function, which is shown as follows:

$$\widehat{sim}_{ij} = \exp\left(-\lambda \cdot dis_{ij}\right) \tag{5}$$

where $\lambda \geq 0$. When the distance of the two user nodes is close to 0, the value of $\widehat{sim}_{ij}$ is close to 1, which means that their vector representations are very similar. While when the distance is large enough, the value of $\widehat{sim}_{ij}$ is close to 0, which means that their vector representations are quite different. In Section II, we have defined an empirical similarity metric sim_{ij} to characterize the differences of behavior. To preserve the different behaviors across diverse user nodes, we treat the empirical similarity metric as prior information, and approximate it by the similarity of the vector representations. The objective function is shown as follows:

$$\mathcal{L}'_{sim} = \sum_{i,j=1}^{m} ||\widehat{sim}_{ij} - sim_{ij}||_2^2 \tag{6}$$

We aim to detect the fraud blocks by the distribution of user nodes in the latent space, which requires the model to aggregate suspicious user nodes in the same fraud block and spread out normal user nodes. Moreover, similar behaviors for a group of users are likely to be fraudulent signals in general. Therefore, capturing similar behaviors is more crucial. To make the objective function more robust, we impose larger weights to user node pairs that are more similar. The revised objective function is shown as follows:

$$\mathcal{L}_{sim} = \sum_{i,j=1}^{m} sim_{ij} \cdot ||\widehat{sim}_{ij} - sim_{ij}||_2^2 \tag{7}$$

C. Model Summary and Optimization

As shown in Fig. 2, in order to simultaneously preserve the global graph structure information and behavior characteristics of different users, we jointly optimize both two components. The overall objective function is the combination of Eq. (3) and Eq. (7), which is shown as follows:

$$\mathcal{L} = \mathcal{L}_{recon} + \alpha\mathcal{L}_{sim} + \gamma\mathcal{L}_{reg} \tag{8}$$

where α and γ are two hyper-parameters, and L_{reg} is the L2-norm regularizer term to prevent overfitting, which is defined as follows:

$$\mathcal{L}_{reg} = \frac{1}{2}\sum_{l=1}^{K}(||W^{(l)}||_2^2 + ||\hat{W}^{(l)}||_2^2 + ||b^{(l)}||_2^2 + ||\hat{b}^{(l)}||_2^2) \tag{9}$$

In order to minimize the overall objective function, we need to calculate the partial derivatives of $\partial\mathcal{L}/\partial\hat{W}^{(l)}$, $\partial\mathcal{L}/\partial\hat{b}^{(l)}$, $\partial\mathcal{L}/\partial W^{(l)}$ and $\partial\mathcal{L}/\partial b^{(l)}$. The detail forms of them are shown as follows:

$$\frac{\partial\mathcal{L}}{\partial\hat{W}^{(l)}} = \frac{\partial\mathcal{L}_{recon}}{\partial\hat{W}^{(l)}} + \gamma\frac{\partial\mathcal{L}_{reg}}{\partial\hat{W}^{(l)}} \tag{10}$$

$$\frac{\partial\mathcal{L}}{\partial\hat{b}^{(l)}} = \frac{\partial\mathcal{L}_{recon}}{\partial\hat{b}^{(l)}} + \gamma\frac{\partial\mathcal{L}_{reg}}{\partial\hat{b}^{(l)}} \tag{11}$$

$$\frac{\partial\mathcal{L}}{\partial W^{(l)}} = \frac{\partial\mathcal{L}_{recon}}{\partial W^{(l)}} + \alpha\frac{\partial\mathcal{L}_{sim}}{\partial W^{(l)}} + \gamma\frac{\partial\mathcal{L}_{reg}}{\partial W^{(l)}} \tag{12}$$

$$\frac{\partial\mathcal{L}}{\partial b^{(l)}} = \frac{\partial\mathcal{L}_{recon}}{\partial b^{(l)}} + \alpha\frac{\partial\mathcal{L}_{sim}}{\partial b^{(l)}} + \gamma\frac{\partial\mathcal{L}_{reg}}{\partial b^{(l)}} \tag{13}$$

where $l = 1, 2, ..., K$.

We first focus on the calculation of Eq. (10) and Eq. (11). The first terms of Eq. (10) and Eq. (11) can be represented as follows:

$$\frac{\partial\mathcal{L}_{recon}}{\partial\hat{W}^{(l)}} = \frac{\partial\mathcal{L}_{recon}}{\partial\hat{S}} \cdot \frac{\partial\hat{S}}{\partial\hat{W}^{(l)}} \tag{14}$$

$$\frac{\partial\mathcal{L}_{recon}}{\partial\hat{b}^{(l)}} = \frac{\partial\mathcal{L}_{recon}}{\partial\hat{S}} \cdot \frac{\partial\hat{S}}{\partial\hat{b}^{(l)}} \tag{15}$$

The partial derivative of $\partial\mathcal{L}_{recon}/\partial\hat{S}$ can be easily calculated based on Eq. (3), which is shown as follows:

$$\frac{\partial\mathcal{L}_{recon}}{\partial\hat{S}} = 2(\hat{S} - S) \odot H \tag{16}$$

Because $\hat{S} = \sigma(\hat{W}^{(K)}G^{(K-1)} + b^{(K)}), G^{(K-1)} = \{g_1^{K-1}, g_2^{K-1}, ...g_m^{K-1}\}$, the partial derivatives of $\partial\hat{S}/\partial\hat{W}^{(l)}$ and $\partial\hat{S}/\partial\hat{b}^{(l)}(l = 1, 2, ..., K)$ can be obtained iteratively by back-propagation. Similarly, the partial derivatives of $\partial\mathcal{L}_{recon}/\partial W^{(l)}$ and $\partial\mathcal{L}_{recon}/\partial b^{(l)}(l = 1, 2, ..., K)$ can also be obtained during this process, which are the first terms of Eq. (12) and Eq. (13).

The second terms of Eq. (10) and Eq. (11) can be represented as $\gamma \cdot \hat{W}^{(l)}$ and $\gamma \cdot \hat{b}^{(l)}$ respectively according to Eq. (9). Based on the above analysis, we can calculate Eq. (10) and Eq. (11) successfully.

Then we focus on the calculation of Eq. (12) and Eq. (13). Since the first terms ($\partial\mathcal{L}_{recon}/\partial W^{(l)}$ and $\partial\mathcal{L}_{recon}/\partial b^{(l)}$) have been calculated before, we only need to consider the second and the third terms.

The partial derivatives of $\partial\mathcal{L}_{sim}/\partial W^{(l)}$ and $\partial\mathcal{L}_{sim}/\partial b^{(l)}$ can be rephrased as follows:

$$\frac{\partial\mathcal{L}_{sim}}{\partial W^{(l)}} = \sum_{i,j=1}^{m} \frac{\partial\mathcal{L}_{sim}}{\partial\widehat{sim}_{ij}} \cdot \frac{\partial\widehat{sim}_{ij}}{\partial dis_{ij}} \cdot \frac{\partial dis_{ij}}{\partial W^{(l)}} \tag{17}$$

$$\frac{\partial\mathcal{L}_{sim}}{\partial b^{(l)}} = \sum_{i,j=1}^{m} \frac{\partial\mathcal{L}_{sim}}{\partial\widehat{sim}_{ij}} \cdot \frac{\partial\widehat{sim}_{ij}}{\partial dis_{ij}} \cdot \frac{\partial dis_{ij}}{\partial b^{(l)}} \tag{18}$$

where $\partial\mathcal{L}_{sim}/\partial\widehat{sim}_{ij} = 2sim_{ij}(\widehat{sim}_{ij} - sim_{ij})$ and $\partial\widehat{sim}_{ij}/\partial dis_{ij} = -\lambda\exp\left(-\lambda \cdot dis_{ij}\right)$. $\partial dis_{ij}/\partial W^{(l)}$ and $\partial dis_{ij}/\partial b^{(l)}(l = 1, 2, ..., K)$ can be easily obtained by back-propagation.

The third terms of Eq. (12) and Eq. (13) are $\gamma \cdot W^{(l)}$ and $\gamma \cdot b^{(l)}$ respectively.

However, optimizing the overall objective function (Eq. (8)) is computationally expensive because all pairwise user node similarities are used when preserving the different user behavior information in the second component of the deep embedding framework. To address this problem, we adopt the idea of negative sampling [10] to optimize the computational cost of pairwise user similarities. The basic idea is that for one user node, we do not need to calculate its similarities with all other user nodes, but only a subset of representative user nodes. In this problem, an assumption is that two user nodes are likely similar if they share many item nodes. Therefore, in order to preserve the similarities in the low-dimensional latent space, for a given user node x_i, we first need to choose the user nodes that share at least one of the same item nodes with x_i to calculate the similarities. Besides, we hope to distinguish user nodes with different behavior characteristics. Therefore, we also choose L negative user nodes that share no common items with x_i randomly based on the idea of negative sampling. By this way, we can greatly reduce the computational overhead.

D. Deep Structure Learning for Fraud Detection

The deep structure learning introduced in Section III-C makes a deep mining of the topological information of corresponding bipartite graph, and embeds the excavated structure knowledge into a low-dimensional space. After obtaining embedding results, we expect that the vector representations of suspicious users in the same fraud block will distribute as closely as possible, while the vector representations of normal users distribute uniformly in the remaining latent space. Therefore, the fraud detection problem in the bipartite graph is ingeniously transformed to a dense region detection problem in the latent vector space. The dense region detection problem can be achieved easily by many existing density-based detection methods. In this paper, we adopt DBSCAN algorithm [16], which is one of the most common density-based clustering algorithms. User nodes that are close in distance and satisfy a certain number will be clustered together, and the algorithm will further label them as fraudsters in a fraud block. Specifically, the trait that normal users distribute uniformly by deep structure learning will make DBSCAN algorithm work more effectively and efficiently.

Because DeepFD is highly nonlinear, before optimizing the model, we employ Deep Belief Network to initialize the parameters, which has been widely used in deep learning to find the optimal solution [11], [12]. Finally, we adopt stochastic gradient descent algorithm to minimize the overall loss function. The pseudocode of the DeepFD model is presented in Algorithm 1.

Table I
TIME COMPLEXITY COMPARISON FOR FRAUD DETECTION METHODS

Methods	Time Complexity	Parameters
M-Zoom [2]	$\mathcal{O}(\lvert E\rvert \log W)$	$\lvert E\rvert$: Number of edges
D-Cube [3]	$\mathcal{O}(\lvert E\rvert W)$	m: Number of user nodes
HoloScope [4]	Subquadratic $\mathcal{O}(m^2)$	n: Number of item nodes
FRAUDAR [6]	$\mathcal{O}(\lvert E\rvert \log{(m+n)})$	W: $\max{(m,n)}$
DeepFD	$\mathcal{O}(I(d_1 d_2 + L)mh)$	Other Parameters are shown in Section III-C

Based on the above analysis, we can conclude that the time complexity of the DeepFD model is $\mathcal{O}(I(d_1 d_2 + L)mh)$, where m is the number of user nodes, h is the maximum dimension of the hidden layers in the first component of the deep embedding framework, d_1 is the average degree of the user nodes, d_2 is the average degree of the item nodes, L is the number of the negative samples and I is the number of the iterations. In the interaction information graph, the values of d_1 and d_2 are very small in general. Therefore, the time complexity of the DeepFD model is linear to the number of user nodes approximately. Table I shows the comparison of time complexity between our method and four state-of-the-art methods that will be described in Section IV-B. The results show that the time complexity of our method is comparable with the state-of-the-art baselines.

IV. EXPERIMENTS

In this section, we empirically evaluate the effectiveness and robustness of DeepFD on fraud detection. We compare our method with four state-of-the-art fraud detection methods on three semi-real synthetic datasets and one real-world DDos attack dataset. To show the superiority of our deep structure learning, we also design three new network embedded fraud detection methods for comparisons.

Algorithm 1 DeepFD model

```
Input: The interaction information graph $G = (X, Y, E)$,
    The hyper-parameters $\alpha$ and $\gamma$, Batch size $p$, Embedding
    size $d$, Number of iterations $I$
Output: Fraudsters for all fraud blocks $fraudblock$
Initialize the parameter set $\{\hat{W}^{(l)}, W^{(l)}, \hat{b}^{(l)}, b^{(l)}\}_{l=1}^{K}$ by
    Deep Belief Network;
$sim =$ GetSimliariyByNegativeSampling($G$);
for $i = 0$ to $I$ do
    $sim' =$ Shuffle($sim$);
    while True do
        $minibatch =$ Sample($sim'$, $p$);
        $nodeset =$ ConstructUserNodes($minibatch$);
        $loss =$ Optimize($\alpha$, $\gamma$, $nodeset$, $minibatch$);
        if Batch sampling is end then
            break;
        end if
    end while
end for
$f =$ GetEmbeddingResults($X$);
$fraudblock =$ FindFraudblockByDBSCAN($f$);
return $fraudblock$;
```

A. Datasets

To validate the effectiveness and robustness of DeepFD, we collect different types of datasets to evaluate the proposed method, which include business reviews, product rating and real-world flow records of DDos attack in a university of China.

In these datasets, Yelp [3], Amazon Instrument [4] and Amazon Movie [4] are semi-real datasets where synthetic fraud blocks are injected. The descriptions of them are listed as follows:

- **Yelp** [3]: It is a subset of the business reviews of Yelp, including restaurant reviews, hotel reviews, *etc.* In this dataset, user nodes are consumers, item nodes are businesses and edges represent the user reviews on businesses.
- **Amazon Instrument** and **Amazon Movie** [4]: They are purchase records of products in Amazon. In the two datasets, user nodes are consumers, item nodes are products and edges represent rating scores for products.

The DDos attack dataset are real-world flow records in a network, which consist of the labeled anomaly flow records to be detected. The dataset are described as follows:

- **DDos attack**: It is a subset of real-world DDos attack on the hosts in T University. This dataset records all network flow records during the attack. The format of a flow record is (source IP address, source port, destination IP address, destination port, protocol). All anomaly flow records have been labeled by DPI (Deep Packet Inspection) device and further checked by network administrators. In this dataset, user nodes refer to source IP addresses and item nodes refer to IP addresses of hosts in T University.

The detail statistics of datasets are shown in Table II.

B. Baseline Algorithms

In order to evaluate the performance of DeepFD model, we first employ the following four state-of-the-art fraud detection methods as baselines.

- **M-Zoom** [2]: It employs a greedy search algorithm to detect suspicious dense blocks by a given density measure in the attributed bipartite graph. It can support various density measures.
- **D-Cube** [3]: It extends M-Zoom by searching attributes sequentially in the attributed bipartite graph to improve accuracy.

Table II
DATASET STATISTICS

Dataset	#(User nodes)	#(Item nodes)	#(Edges)
Yelp	686K	85.3K	2.68M
Amazon Instrument	339K	83K	0.5M
Amazon Movie	2090K	201K	4.61M
DDos attack	130K	70K	32.1M

- **HoloScope** [4]: It designs a systematic metric to detect the fraud blocks. The metric combines several suspicious signals, namely graph topology information, temporal bursts and drops and rating deviation.
- **FRAUDAR** [6]: It is a graph-based fraud detection algorithm. The proposed metric can detect fraudsters under camouflage by weighting edges' suspiciousness.

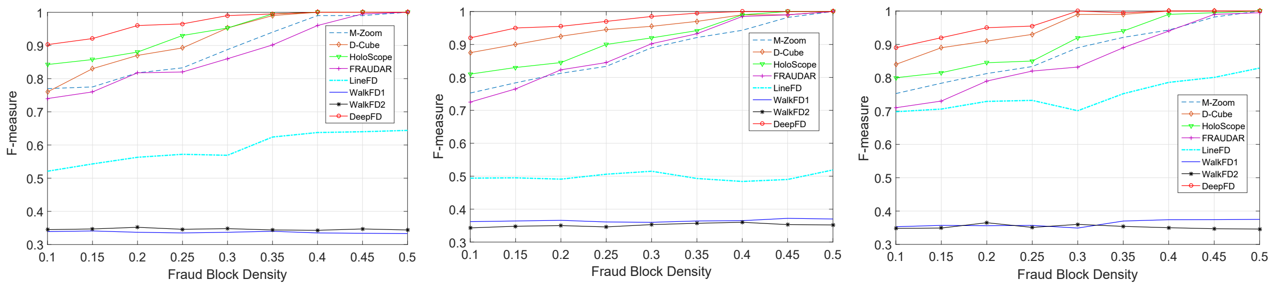
DeepFD is a network embedded fraud detection method. In order to demonstrate the superiority of our deep structure learning, in addition to the above four fraud detection methods, we also replace the embedding algorithm of DeepFD with three state-of-the-art embedding methods, including DeepWalk [13], node2vec [14] and LINE [15], to compare with DeepFD.

Note that these three network embedding methods are not designed for bipartite graphs. In order to employ them, we need to convert bipartite graph into homogenous graph. Since our goals are to learn the vector representations of user nodes and further detect fraudsters, we here construct user topological graph from the user behavior data. Considering that if two users share some common items, they are likely similar. We define that if two user nodes share at least one common item, there exists an edge between them. Otherwise, there is no edge between them. In this way, we can construct a user topological graph. For DeepWalk, the edges in the graph are binary, while for node2vec and LINE, each edge is associated with a weight, which indicates the similarity between the two user nodes. The similarity metric can be defined like that in Section II.

Based on the constructed graph, the three network embedding methods can map user nodes into low-dimensional vector space. After getting the vector representations, we can detect the fraud blocks by the position distribution of user nodes in the latent space. Without loss of generality, in this paper, we uniformly adopt DBSCAN algorithm [16] to detect fraud blocks, which is similar to DeepFD. Note that the embedding results obtained by different methods are not within the same value range. For fair comparisons, we map the embedding results into the same value range between 0 and 1 before applying DBSCAN algorithm.

The three network embedded comparison methods can be summarized as follows:

- **WalkFD1**: It is a combination of DeepWalk and DBSCAN. We first use DeepWalk to learn the low-dimensional latent vector representations for user nodes in constructed user topological graph by local information obtained from uniform random walks, and then employ DBSCAN to detect fraud blocks.
- **WalkFD2**: It is a combination of node2vec and DBSCAN. node2vec employs a flexible biased random walk to learn the vector representations of user nodes in the weighted user topological graph constructed as above. After that, DBSCAN detects fraud blocks by the distribution of user nodes.



(a) Yelp (b) Amazon Instrument (c) Amazon Movie

Figure 3. Experimental results for the performance (F-measure) comparison between DeepFD and different baseline methods (four state-of-the-art fraud detection methods and three network embedded fraud detection methods) on three semi-real synthetic datasets

- **LineFD**: It is a combination of LINE and DBSCAN. The user topological graph constructed here is also weighted and LINE can preserve the first-order proximity and the second-order proximity between user nodes.

C. Parameter settings

In this part, we will introduce the parameter settings for the DeepFD model and baseline algorithms respectively.

The parameter settings for the four state-of-the-art fraud detection methods are described as follows: For M-Zoom, we set the density measure to arithmetic average mass, and dimension of attributions to 2, which represents that we only consider the user nodes and item nodes without other extra attributions. For D-cube, a specific parameter is the policy for choosing an attribute from which attribute values are removed. We set this parameter to maximum cardinality policy, which means the attribute with the largest cardinality is chosen. The other parameters are the same with M-Zoom. For FRAUDAR, the most important parameter is edges' suspiciousness, the form of which is set to $1/log(d+c)$, where d is the degree of item node in a given edge and c is a small constant (set to 5 in the experiments). For HoloScope, we only consider the graph topology information as suspicious signal, and adopt the default settings for other parameters.

The parameter settings for the three network embedded fraud detection baseline methods are described as follows: For WalkFD1, we set window size to 10, walk length to 40, the number of walks per vertex to 40. For WalkFD2, parameter p is set to 1, parameter q is set to 2, and the other parameters are the same with WalkFD1. For LineFD, we consider both first-order and second-order proximity. The number of negative samples is set to 5, the learning rate is set to 0.025, and the number of iterations is set to 5.

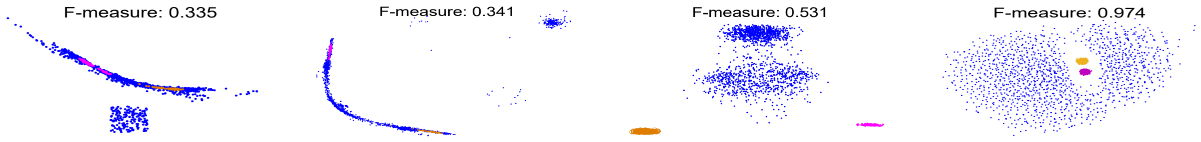
For DeepFD model, the hyper-parameters α and β are selected by grid search, which get the best performance on the validation set. In the experiments, we set α to 10, β to 20, the regularizer term weight γ to 0.001 and the learning rate to 0.025. We apply a three layers auto-encoder for all datasets. The dimension of the first layer depends on the number of item nodes in different datasets, the dimension of the second layer is set to 128 and the dimension of the third layer represents the dimension of the embedding results. In Section IV-F, we will show that the performance of the algorithm is insensitive to the dimension of the learnt vector representations. Therefore, for better understanding and visualization, we set the dimension of vector representations to 2.

D. Evaluation on Synthetic Datasets

In order to demonstrate the effectiveness of our method, we mimic fraudsters to generate fraudulent behaviors, and inject synthetic fraud blocks with different desity settings into Yelp, Amazon Instrument and Amazon Movie respectively. We adopt the fraud blocks injection method used typically in previous works [4], [6]. First, we choose some unpopular item nodes that are rarely associated with user nodes as fraudulent items. We then add a certain percentage of fraudsters and generate edges from fraudsters to fraudulent items based on various block density settings varying from 0.1 to 0.5. The generated fraud block is in a uniform manner. Finally, we add some camouflage actions for fraudsters, which means that each fraudster will select some normal items to review or rate for evading the detection.

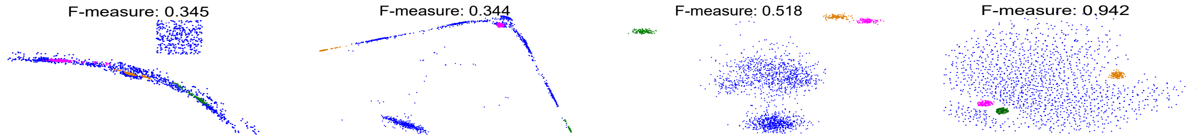
We first compare DeepFD with the four state-of-the-art fraud detection methods ($i.e.$, M-Zoom, D-Cube, HoloScope and FRAUDAR) on the three synthetic datasets. The experimental results are shown in Fig. 3. We can observe that the overall trends of F-measure for different datasets are consistent. When the fraud block density is larger than 0.4, almost all algorithms can achieve a high performance. However, when the density is reduced, the performance of these baselines has an obvious drop, while DeepFD can keep a high F-measure and is consistently better than other baseline algorithms. The reason behind is that these baseline methods are based on different kinds of density measures, which are prone to be sensitive to the density of fraud block. However, DeepFD makes the best use of deep graph structure and takes full consideration of user behavior differences, which make it more robust to distinguish normal users and fraudsters. The experiments demonstrate the effectiveness of DeepFD.

In order to further demonstrate the superiority of our deep structure learning (Algorithm 1) in fraud detection, we also compare DeepFD with three other network embedded



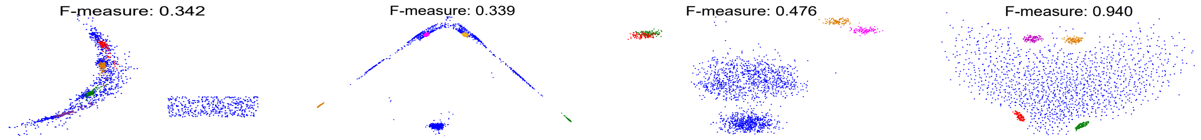
(a) WalkFD1 (b) WalkFD2 (c) LineFD (d) DeepFD

Figure 4. F-measure comparison for different fraud detection methods and visualization for embedding results (**2 fraud blocks injected**). Blue color represents normal users, and other colors represent different fraud blocks.



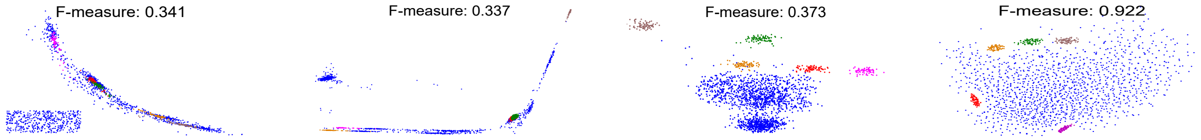
(a) WalkFD1 (b) WalkFD2 (c) LineFD (d) DeepFD

Figure 5. F-measure comparison for different fraud detection methods and visualization for embedding results (**3 fraud blocks injected**). Blue color represents normal users, and other colors represent different fraud blocks.



(a) WalkFD1 (b) WalkFD2 (c) LineFD (d) DeepFD

Figure 6. F-measure comparison for different fraud detection methods and visualization for embedding results (**4 fraud blocks injected**). Blue color represents normal users, and other colors represent different fraud blocks.



(a) WalkFD1 (b) WalkFD2 (c) LineFD (d) DeepFD

Figure 7. F-measure comparison for different fraud detection methods and visualization for embedding results (**5 fraud blocks injected**). Blue color represents normal users, and other colors represent different fraud blocks.

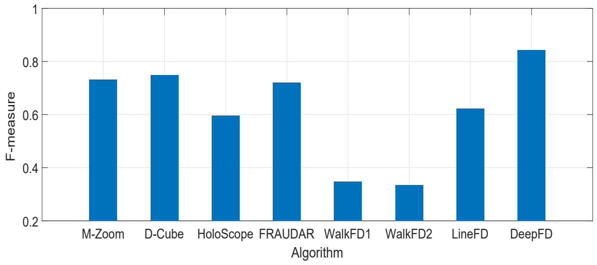
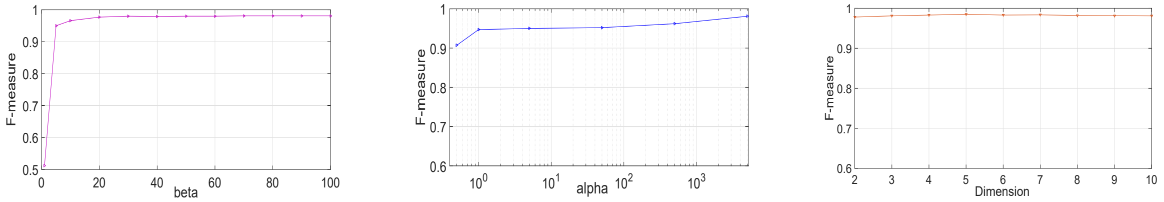


Figure 8. Experimental results on real-world DDos attack dataset

fraud detection methods introduced in Section IV-B. For fair comparison, all these baseline methods apply DBSCAN algorithm with the same settings to detect fraud block after getting the vector representations of user nodes. The results can also be found in Fig. 3. We can observe that the F-measure of DeepFD is much higher than that of the three baseline methods for different fraud block density settings. It indicates that the embedding results learnt from DeepFD are more effective to capture the suspicious information in the graph, which is because that these baseline methods cannot preserve the independent relationship among normal user nodes and lose the partial topological relationship between user nodes and item nodes. Besides, we can find that LineFD outperforms WalkFD1 and WalkFD2. It is because that LineFD captures the similarity of user nodes that share common item nodes, while WalkFD1 and WalkFD2 extract the graph structure information by random walk, which makes them difficult to distinguish normal users and fraudsters.

In practice, there may exist multiple uncorrelated fraud blocks in an interaction information graph. It is very difficult for previous fraud detection algorithms to accurately detect all fraud blocks without predefining the number of the blocks. However, DeepFD is robust to solve the problem well. The vector representations of user nodes learnt by the embedding model of DeepFD preserve the different user behavior characteristics, which brings two favorable traits to the position distribution of user nodes. On the one hand, all normal users are distributed uniformly in the latent space. On the other hand, suspicious users in the same fraud block tend to form a cluster. To demonstrate the robustness of DeepFD, we inject a different number of fraud blocks into Amazon Instrument dataset, and evaluate the effectiveness of DeepFD and the three netwrok embedded fraud detection methods (*i.e.*, WalkFD1, WalkFD2 and LineFD). Fig. 4, Fig. 5, Fig. 6 and Fig. 7 respectively show the F-measure of fraud detection and distribution of embedding results for DeepFD and the three baselines when different number of fraud blocks are injected. We can see that DeepFD is robust in automatically detecting multiple fraud blocks in the interaction information graph. The F-measure is always high



(a) β (b) α (c) Dimension

Figure 9. Parameter sensitivity

and stable with the change of the number of fraud blocks. While the other three netwrok embedded methods perform poorly when the number of fraud blocks increases. Besides, the distribution of embedding results for DeepFD show that fraudsters belonging to the same fraud block are clustered together. We can clearly distinguish between normal users and different fraud blocks. For WalkFD1 and WalkFD2, the characteristics of random walk mix fraud blocks with normal users, which leads to a high false-positive rate. For LineFD, in addition to fraudsters, normal users are also clustered easily, which makes it difficult to accurately detect fraud blocks. Therefore, the visualization results demonstrate that DeepFD preserves favorable traits in embedding results and has intuitive comprehensibility.

E. *Effectiveness on Real-world Dataset*

In this section, we evaluate the effectiveness of DeepFD on real-world DDos attack dataset described in Section IV-A. A typical form of DDos attack is that a large number of zombie hosts initiate requests to target hosts, resulting in a decrease in service capability of target hosts, which can also be seen as a type of fraudulent behavior.

Here we employ above 7 fraud detection baseline algorithms for comparison to detect zombie hosts. It should be pointed that M-Zoom and D-Cube have shown their ability to detect suspicious hosts in previous works [2], [3]. Our experimental results are shown in Fig. 8. We can see that our method achieves significant improvement on F-measure compared with other baseline methods, including M-Zoom and D-Cube. DeepFD outperforms the best baseline D-Cube by around 10%, which demonstrates that our method is also effective on real-world dataset.

F. *Parameter Sensitivity*

In this section, we investigate the effect of the two key hyper-parameters (β and α) and the dimension of network embedding on the performance of DeepFD. We respectively study the trend of F-measure with the change of the three parameters on the Amazon Movie dataset.

The hyper-parameter β is used to control the reconstruction weight of non-zero elements in auto-encoder. Increasing the value of β means that auto-encoder will be more prone to reconstruct non-zero elements in training. Fig. 9(a) shows the trend of F-measure under different values of β. We can observe that the performance of DeepFD is stable and F-measure is consistently higher than 0.95 when $\beta \geq 5$. However, when β is small ($e.g.$, $\beta = 1$), the performance becomes poor, which is because auto-encoder tends to reconstruct zero elements in s_i. As we have explained before, non-zero elements are more crucial than zero elements in fraud detection.

The hyper-parameter α balances the weight between the two components of embedding framework for DeepFD in (8). Fig. 9(b) shows the trend of F-measure when α takes values under different orders of magnitude. We can see that the performance of DeepFD is insensitive to α. The insensitivity to hyper-parameters implies that our model can be easily trained.

Fig. 9(c) shows the trend of F-measure with the change of embedding dimension. We can see that the F-measure is always high under different dimension settings, which means embedding dimension has no significant effect on the performance of DeepFD. In this paper, we set the dimension to 2 for better visualization.

V. Related Work

Fraud detection: Most of the popular methods identify fraudulent behavior with the help of attributes in the bipartite graph. M-Zoom [2], D-Cude [3] and CROSSPOT [17] model attribute information as a tensor, and further search for dense blocks by different density measures. Shah $et\ al.$ detect the anomaly by the distribution of rating scores [18], while Jindal $et\ al.$ find fraud activities by review text [19]. Besides, some works assume that the temporal burst is a signal of the suspicious behaviors, and use the bursty patterns to detect review spam [20], [21]. In order to make the detection method more effective, Liu $et\ al.$ systematically use multiple signals to form a suspiciousness metric, which include graph topology information, temporal bursts and drops and rating deviation [4]. However, fraudsters tend to take some camouflage actions, which make the attributes information hard to be distinguished. Therefore, it is difficult for the attribute-based methods to be widely used. In contrast, since the fraudsters inevitably generate fraudulent edges in the bipartite graph, the graph topology structure cannot be easily camouflaged. It provides us an idea that whether we can detect the fraud blocks only by graph structure information. However, most of the existing topology based detection methods [6]–[9] only consider shallow structure information. They are sensitive to the density of fraud blocks. Moreover, almost all of them cannot automatically detect all fraud blocks without predefining the number of the blocks. Our deep network embedded detection method fills the gap. After getting the embedding results for user nodes in the interaction information graph, all fraud blocks can be detected automatically by density-based detection methods.

Network embedding: Network embedding aims to learn low-dimensional vector representations for nodes of the network. DeepWalk [13], node2vec [14] and LINE [15] try to learn representations from local network structure, while GraRep [22] tends to preserve global network topology structure information. In order to capture highly non-linear network structure, SDNE [11] designs an embedding model that contains multiple layers of non-linear functions. Besides, network embedding can also be applied to many specific tasks, such as finding structurally inconsistent nodes in the graph [23], [24]. Our work is inspired by these methods. In this paper, we propose a novel deep network embedded fraud detection model named DeepFD to detect the fraudulent behaviors in the interaction information graph.

VI. Conclusion

In this paper, we propose a novel deep structure learning method DeepFD for fraud detection. DeepFD can simultaneously capture global graph structure information and local user behavior characteristics. For this goal, we design a deep auto-encoder to reconstruct original bipartite graph topology and approximate the empirical similarity of user behavior by the similarity of vector representations. The deep embedding results could then be used for effective fraud block detection by the position distribution of user vector representations. Experimental results on three semi-real synthetic datasets and one real-world DDos attack dataset demonstrate that DeepFD not only significantly outperforms other baseline methods, but is also more robust for automatic detection of multiple fraud blocks.

Acknowledgment

This work was supported by the National Key Research and Development Program of China (No. 2016YFB0801301 and 2016QY12Z2103), the NSFC (No. 61502479 and 61872360), the MQNS (No. 9201701203), the MQ EPS (No. 9201701455), the US National Science Foundation (NSF) through Grant IIS-1763452, the Youth Innovation Promotion Association CAS (No. 2017210), and the Collaborative Research Project (CRP) between Macquarie University and Data61 on dynamic graph mining. Chuan Zhou is the corresponding author.

References

[1] X. Zhu, H. Tao, Z. Wu, J. Cao, K. Kalish, and J. Kayne, “Fraud Prevention in Online Digital Advertising,” Springer, 2017.

[2] K. Shin, B. Hooi, and C. Faloutsos, “M-zoom: Fast dense-block detection in tensors with quality guarantees,” in *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*. 2016.

[3] K. Shin, B. Hooi, J. Kim, and C. Faloutsos, “D-cube: Dense-block detection in terabyte-scale tensors,” in WSDM 2017.

[4] S. Liu, B. Hooi, and C. Faloutsos, “Holoscope: Topology-and-spike aware fraud detection,” *arXiv preprint arXiv:1705.02505*, 2017.

[5] Y. Xiang, K. Li, and W. Zhou, “Low-rate ddos attacks detection and traceback by using new information metrics,” *TIFS*, vol. 6, no. 2, pp. 426–437, 2011.

[6] 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 2016.

[7] M. Jiang, P. Cui, A. Beutel, C. Faloutsos, and S. Yang, “Inferring strange behavior from connectivity pattern in social networks,” in PAKDD 2014.

[8] A. Beutel, W. Xu, V. Guruswami, C. Palow, and C. Faloutsos, “Copycatch: stopping group attacks by spotting lockstep behavior in social networks,” in WWW 2013.

[9] B. Perozzi, L. Akoglu, P. Iglesias Sánchez, and E. Müller, “Focused clustering and outlier detection in large attributed graphs,” in KDD 2014.

[10] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean, “Distributed representations of words and phrases and their compositionality,” in NIPS 2013.

[11] D. Wang, P. Cui, and W. Zhu, “Structural deep network embedding,” in KDD 2016.

[12] D. Erhan, Y. Bengio, A. Courville, P.-A. Manzagol, P. Vincent, and S. Bengio, “Why does unsupervised pre-training help deep learning?” *Journal of Machine Learning Research*, vol. 11, no. Feb, pp. 625–660, 2010.

[13] B. Perozzi, R. Al-Rfou, and S. Skiena, “Deepwalk: Online learning of social representations,” in KDD 2014.

[14] A. Grover and J. Leskovec, “node2vec: Scalable feature learning for networks,” in KDD 2016.

[15] J. Tang, M. Qu, M. Wang, M. Zhang, J. Yan, and Q. Mei, “Line: Large-scale information network embedding,” in WWW 2015.

[16] M. Ester, H.-P. Kriegel, J. Sander, X. Xu *et al.*, “A density-based algorithm for discovering clusters in large spatial databases with noise.” in KDD 1996.

[17] M. Jiang, A. Beutel, P. Cui, B. Hooi, S. Yang, and C. Faloutsos, “A general suspiciousness metric for dense blocks in multimodal data,” in ICDM 2015.

[18] N. Shah, A. Beutel, B. Hooi, L. Akoglu, S. Gunnemann, D. Makhija, M. Kumar, and C. Faloutsos, “Edgecentric: Anomaly detection in edge-attributed networks,” in ICDMW 2016.

[19] N. Jindal and B. Liu, “Opinion spam and analysis,” in WSDM 2008.

[20] S. Xie, G. Wang, S. Lin, and P. S. Yu, “Review spam detection via temporal pattern discovery,” in KDD 2012.

[21] H. Li, G. Fei, S. Wang, B. Liu, W. Shao, A. Mukherjee, and J. Shao, “Modeling review spam using temporal patterns and co-bursting behaviors,” *arXiv preprint arXiv:1611.06625*, 2016.

[22] S. Cao, W. Lu, and Q. Xu, “Grarep: Learning graph representations with global structural information,” in CIKM 2015.

[23] R. Hu, C. C. Aggarwal, S. Ma, and J. Huai, “An embedding approach to anomaly detection,” in ICDE 2016.

[24] K. Sricharan and K. Das, “Localizing anomalous changes in time-evolving graphs,” in SIGMOD 2014.