A Synergistic Approach for Graph Anomaly Detection With Pattern Mining and Feature Learning

Tong Zhao, Tianwen Jiang, Neil Shah, and Meng Jiang

***Abstract*— Detecting anomalies on graph data has two types of methods. One is pattern mining that discovers strange structures globally such as quasi-cliques, bipartite cores, or dense blocks in the graph's adjacency matrix. The other is feature learning that mainly uses graph neural networks (GNNs) to aggregate information from local neighborhood into node representations. However, there is a lack of study that utilizes both the global and local information for graph anomaly detection. In this article, we propose a synergistic approach that leverages pattern mining to inform the GNN algorithms on how to aggregate local information through connections to capture the global patterns. Specifically, it uses a GNN encoder to perform feature aggregation, and the pattern mining algorithms supervise the GNN training process through a novel loss function. We provide theoretical analysis on the effectiveness of the loss function, as well as empirical analysis on the proposed approach across a variety of GNN algorithms and pattern mining methods. Experiments on real-world data show that the synergistic approach performs significantly better than existing graph anomaly detection methods.**

***Index Terms*— Graph anomaly detection, graph neural network (GNN), graph pattern mining, unsupervised learning.**

I. Introduction

ANOMALY detection on large-scale bipartite graphs is an important task in many real-world applications. Take social networks as an example: malicious users such as social bots, spammers, and fake reviewers are severely affecting customer experiences on the platforms, where we have "who-follows-whom" and "who-posts/reviews-what" bipartite graphs. To automatically learn node features for the downstream tasks of graph anomaly detection, graph neural networks (GNNs) have been recognized for their abilities of aggregating attributed information from local neighborhood [1]. Usually, the models aggregate feature information from nodes in local neighborhood through two to four layers (i.e., the number of hops of the neighborhood). If the graph data have a large scale, people are interested in learning a great number of node features in an unsupervised manner so that they can be used to train simple classifiers very quickly for any type of anomaly detection tasks when some *ad hoc* labels become available. So, to train GNN model parameters without node labels, random walk (RW) algorithms discover a specific global property, i.e., whether two nodes are connected within a random-walk distance that forms RW-based loss on the generated features from the last GNN layer [2]–[4].

However, we find that existing GNN-based models perform poorly on benchmarks in the task of graph anomaly detection. The reason is that graph anomalies do not have the aforementioned RW-based global property. In other words, nodes of the same class might not be closer in the graph than those of different classes. For example, the individual anomalies (e.g., fake reviewers) are not likely to be connected nor have common neighbors. They share global properties of being outliers (away from the majority) on the graph. Another example is that when a graph has multiple groups of anomalies (e.g., social botnet groups), the nodes in different groups do not have to be connected while having similar global properties of creating unexpected density. How to effectively train GNNs for graph anomaly detection on bipartite graphs by capturing proper global properties is important and nontrivial.

On the other hand, many graph pattern mining algorithms [5]–[8] have been designed for graph anomaly detection on bipartite graphs in the past few decades. These methods capture the global structural patterns to identify abnormal node groups from the graph. For example, Akoglu *et al.* [9] and Rayana and Akoglu [8] proposed efficient algorithms to detect graph anomalies by measuring the distance of their behavioral patterns from the patterns of the majority. Jiang *et al.* [10], [11] and Hooi *et al.* [5] identified botnets by measuring the unexpectedness of dense bipartite cores and greedily looking for them. However, these algorithms ignore node individual identity and feature information, assuming all the nodes in a group must have the same characteristics and the same label.

Designing a GNN-based representation learning method for graph anomaly detection is a nontrivial task, because the method, after being sufficiently trained, is expected to generate node representations, which can accurately predict minority groups (e.g., outliers) from class-imbalanced data [12]. Compared with the node population of the entire graph, graph

Manuscript received November 30, 2020; revised April 6, 2021; accepted July 31, 2021. This work was supported in part by Snap Research Fellowship and in part by the National Science Foundation (NSF) under Grant IIS-1849816 and Grant CCF-1901059. *(Corresponding author: Meng Jiang.)*

Tong Zhao and Meng Jiang are with the Department of Computer Science and Engineering, University of Notre Dame, Notre Dame, IN 46556 USA (e-mail: tzhao2@nd.edu; mjiang2@nd.edu).

Tianwen Jiang is with the Research Center for Social Computing and Information Retrieval, Harbin Institute of Technology, Harbin 150001, China (e-mail: twjiang@ir.hit.edu.cn).

Neil Shah is with Snap Research at Snap Inc., Seattle, WA 98121 USA (e-mail: nshah@snap.com).

Color versions of one or more figures in this article are available at https://doi.org/10.1109/TNNLS.2021.3102609.

Digital Object Identifier 10.1109/TNNLS.2021.3102609

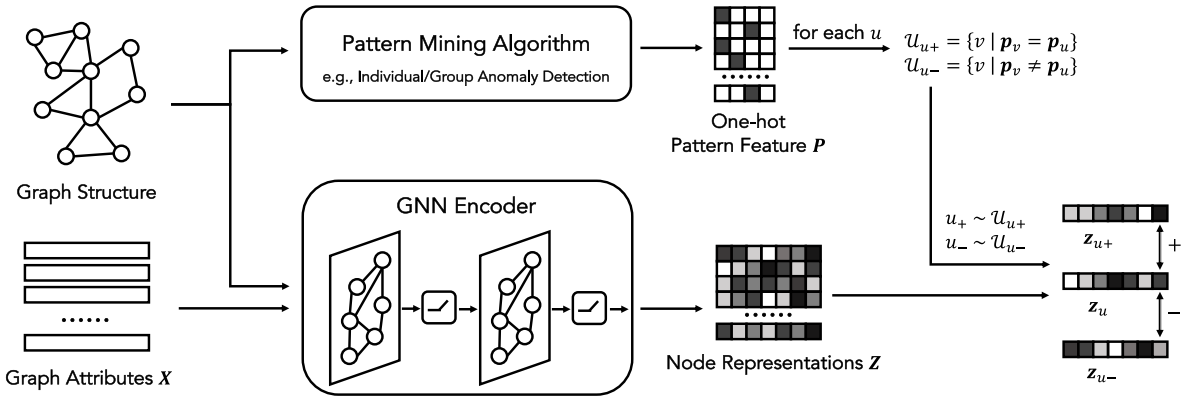

Fig. 1. Overview of the proposed PAMFUL framework: the pattern mining-based graph anomaly detection algorithm uses graph structural data to produce one-hot pattern features of each node, i.e., which global structures (e.g., individual/group anomalies in the graph) the node belongs to. The feature learning-based GNN encoder takes both the graph structure and raw attributes as input and generates low-dimensional representations for each node. During training, a margin loss based on the identified global structures is used to supervise the GNN encoder, with positive and negative pair-wise samples from the pattern features.

anomalies (e.g., fake reviewers, botnet accounts) are the minority. Such severe imbalance is detrimental to model performance: when representations were not properly trained, the models would over-fit on the minority classes and would perform poorly on test/unseen data [12]. Imbalanced machine learning has been studied from many perspectives [13], [14]; nevertheless, to the best of our knowledge, the research problem of reducing predictive error on imbalanced data for unsupervised graph representation learning is barely studied.

In this article, we evaluate node pair-wise similarity using the global structural patterns discovered by pattern mining algorithms, and we present a novel synergistic representation learning approach that combines Pattern Mining and Feature Learning (PAMFUL) for graph anomaly detection. Fig. 1 shows the overview of the proposed PAMFUL framework. PAMFUL takes the graph structure with raw node attributes as input and generates node representations. PAMFUL contains three components: 1) GNN encoder as a feature learning method that generates node representations from the attributed graph; 2) a pattern mining-based graph anomaly detection algorithm that learns the global patterns from the graph structure; and 3) an error-bounded distribution-aware margin loss function to train the GNN encoder based on the output of both the GNN encoder and pattern mining algorithms.

With the above-mentioned design, PAMFUL can effectively incorporate the global patterns learned by pattern mining algorithms into the feature learning process of the GNN encoder. By supervising the GNN encoder with global structural information, the learned node representations will contain not only local aggregated information but also global patterns from the supervisory signals. Therefore, the representations learned by PAMFUL will take advantage of both the local and global contexts for effective graph anomaly detection.

Our proposed PAMFUL does not have restrictions on the choice of GNN encoder or pattern mining algorithm. Any type of existing GNN architecture (e.g., graph convolutional network (GCN) [1], graph attentional network (GAT) [15], GRAPHSAGE [2]) can be used as the GNN encoder, and any unsupervised graph anomaly detection method can serve as the pattern mining algorithm (e.g., individual anomaly detection methods [9], [16] and group anomaly detection methods [5], [11]). Moreover, during the training process, PAMFUL encourages large margins for minority classes. It learns proper margins from the global imbalanced data distribution discovered by pattern mining algorithms. So, it generalizes well on predicting minority classes. Theoretically, we obtain and prove the bound on the prediction error.

The important features of PAMFUL, which are also the main contributions of this work, are summarized as follows.

1) *Effectiveness:* PAMFUL captures global structural patterns when the assumption of RW fails in measuring node similarities for anomaly detection. Experiments on two real-world datasets, aiming at detecting two different kinds of anomalies, demonstrate that any GNN algorithm trained in PAMFUL can perform significantly better than existing methods.
2) *Generalizability:* PAMFUL variants can be applied to train any arbitrary graph neural algorithm. They include loss functions for different global patterns, aiming at different tasks of graph anomaly detection such as individual anomaly detection and group anomaly detection.
3) *Theoretical Guaranteed Performance:* PAMFUL creates a better generalization on patterns of minority groups than existing methods. It maintains a bounded test prediction error on imbalanced data.

II. PROPOSED METHOD

In this section, we first formally define our research problem in Section II-A. Then we present our specific design of the main components of our proposed framework (as shown in Fig. 1): feature learning-based GNN encoders (Section II-B), pattern mining-based graph anomaly detection algorithms (Section II-C), and positive/negative sampling strategies with an error-bounded loss function that combines the two types of methods (Section II-D).

A. Problem Definition

The goal of our approach is to learn low-dimensional representations of user nodes on a bipartite graph for detecting anomalous users. Suppose that $\mathcal{U}$ is the set of users, $\mathcal{V}$ is the

set of items (e.g., products, hashtags), and $\mathcal{R} = \{r_{u,v} | u \in \mathcal{U}, v \in \mathcal{V}\}$, where $r_{u,v}$ denotes the weight of the edge between node u and node v. $r_{u,v} = 0$ indicates that u and v are not connected. If the graph is unweighted, $r_{u,v} = 1$ when the two nodes are connected. So, the problem is defined as follows:

Given a bipartite graph $G = (\mathcal{U}, \mathcal{V}, \mathcal{R})$ and a set of user's node feature vectors $\{\mathbf{x}_u \in \mathbb{R}^{d_x}, \forall u \in \mathcal{U}\}$ (where d_x is the dimension of raw features), **find** a mapping function of the representations of user nodes $f : u \in \mathcal{U} \rightarrow \mathbf{z}_u \in \mathbb{R}^d$, where d is the number of latent dimensions in user embeddings. We expect the user representations are optimized for the task of anomaly detection by preserving both user's node attribute information and proper global properties.

B. Bipartite GNN Encoder

For the feature learning part of the framework, we use GNN encoders to learn node representations from the local neighborhoods. For homogeneous graphs, the aggregations of GNNs are rather straightforward. GNNs generate embeddings of a node by aggregating the embeddings from nodes in its local neighborhood. The assumption is that neighboring nodes have related information and/or similar characteristics.

However, when working with bipartite graphs, this assumption does not hold as neighbors are different types of nodes. Without losing the generalizability of bipartite graphs, we take rating behaviors as an example. At each iteration of generating embeddings of a user, rather than aggregating item embeddings to the user, we aggregate information from other user nodes that have *corated* items (i.e., common neighboring items) to avoid mixing the representations of different types of nodes. Because of the large number of 2-hop neighbors (users with *corated* items), we use the (dis)similarity of behavioral patterns between users to prioritize the users during the aggregation process. Hence, the user–user similarity function $\text{simi}(u, u')$ would play an essential role in the embedding aggregation process on a bipartite graph. We use two user similarity measures that have been applied in user-based collaborative filtering (CF) techniques. Here, we denote $\mathcal{V}_{u,u'}$ for the set of *corated* items by u and u': $\mathcal{V}_{u,u'} = \{v \in \mathcal{V} | r_{u,v} > 0, r_{u',v} > 0\}$.

1) *Pearson Correlation Coefficient (PCC) [17]:* It measures the covariance of the *relative rating* distributions of two users u and u' divided by the product of their standard deviations. The relative rating is the actual rating $r_{u,v}$ minus the average rating of user u: $r_{u,v} - \bar{r}_u$, where u's average rating is

$$\bar{r}_u = \frac{\sum_{v \in \mathcal{V}} r_{u,v}}{\left|\{v \in \mathcal{V} | r_{u,v} > 0\}\right|}. \tag{1}$$

Then, we have the PCC-based user–user similarity

$$\text{simi}(u, u')^{\text{PCC}} = \frac{\sum_{v \in \mathcal{V}_{u,u'}} \left(r_{u,v} - \bar{r}_u\right)\left(r_{u',v} - \bar{r}_{u'}\right)}{\sqrt{\sum_{v \in \mathcal{V}_{u,u'}} \left(r_{u,v} - \bar{r}_u\right)^2 \sum_{v \in \mathcal{V}_{u,u'}} \left(r_{u',v} - \bar{r}_{u'}\right)^2}}. \tag{2}$$

2) *Cosine Similarity (COS):* It measures the similarity between the rating distributions of u and u' in an inner product space

$$\text{simi}(u, u')^{\text{COS}} = \frac{\sum_{v \in \mathcal{V}} r_{u,v} \cdot r_{u',v}}{\sqrt{\sum_{v \in \mathcal{V}} r_{u,v}^2 \sum_{v \in \mathcal{V}} r_{u',v}^2}}. \tag{3}$$

Note that the range of the similarity measures are different: $\text{simi}(u, u')^{\text{PCC}} \in [-1, 1]$ and $\text{simi}(u, u')^{\text{COS}} \in [0, 1]$.

Algorithm 1 Collaborative Aggregation on GSAGE

Input : Bipartite graph $G(\mathcal{U}, \mathcal{V}, \mathcal{R})$; raw features $\{\mathbf{x}_u, \forall u \in \mathcal{U}\}$; depth K; weight matrices $\mathbf{W}^k$ and aggregator functions AGG_k, $\forall k \in \{1, \ldots, K\}$; neighborhood functions $\mathcal{N}_{\mathcal{V}}$ and $\mathcal{N}_{\mathcal{U}}$; size of the set of "important" users S by importance sampling; size of the set of users S^{imp} for sampling; a user similarity function simi: $\mathcal{U} \times \mathcal{U} \rightarrow [-1, 1]$.

Output: Low-dimensional representations $\mathbf{z}_u$ for all $u \in \mathcal{U}$.

```
for u ∈ U do
    N_u^imp ← top S^imp users who has most corated items with u;
end
h_u^0 ← x_u, ∀u ∈ U;
for k = 1, ..., K do
    for u ∈ U do
        N_u^k ← uniformly sample S nodes from N_u^imp;
        h_{N_u}^k ← AGG_k({h_{u'}^{k-1} · simi(u, u') | u' ≠ u, u' ∈ N_u^k});
        h_u^k ← ELU(W^k · CONCAT(h_u^{k-1}, h_{N_u}^k));
    end
    h_u^k ← h_u^k / ||h_u^k||_2, ∀u ∈ U;
end
z_u ← h_u^K, ∀u ∈ U;
```

Algorithm 1 takes graph's sampling-and-aggregation embedding (GSAGE) [2] as an example to describe the collaborative aggregation process on bipartite graphs. Inputs are the bipartite graph $G(\mathcal{U}, \mathcal{V}, \mathcal{R})$ and the raw features for each node $\{\mathbf{x}_u, \forall u \in \mathcal{U}\}$. For each user node u, we first get a set of "important" users $\mathcal{N}_u^{\text{imp}}$ who have the most corated items with u. The rest of the aggregation process generally follows the standard message passing-based GNNs [1], [2], [15]. In each GNN layer, we use the user similarity $\text{simi}(u, u')$ as weight when aggregating the neighbor features and exponential linear unit (ELU) as the nonlinearity function to preserve the possible negative features because of negative similarity scores. The similarity scores are not used when GAT [15] is used as the GNN encoder, as the learned edge attention scores act as weights during aggregation. Note that our proposed framework PAMFUL can be easily generalized to conventional graphs by substituting the bipartite GNN encoder described in this section with traditional GNNs which are designed for homogeneous graphs.

Complexity Analysis: The per-batch space and time complexity is $O(S^K)$, where K is the number of depth in search for aggregation and S is the maximum size of sampled neighbor

users set $\mathcal{N}_u$ for each user u. As S is often a reasonable number (e.g., 100) and K is small (i.e., 2 or 3), the complexity is similar with existing GNNs with mini-batch [2] and reasonable.

C. Pattern Mining Algorithms

For graph pattern mining algorithms, we use unsupervised graph-based *individual anomaly detection* [7], [16] and *group anomaly detection* [5], [11] algorithms. These algorithms take the graph structure as input and output the global pattern role of each user nodes. Fig. 2 illustrates the two types of anomalies.

1) Individual Anomaly Detection Algorithms: assign binary labels to nodes unsupervisedly for the given graph. Here, we denote the output of these algorithms as a binary matrix $\mathbf{P} \in \{0, 1\}^{|\mathcal{U}|\times 2}$, where each line of $\mathbf{P}$ is a one-hot vector indicating the predicted label of a user. For example, if user u_i is predicted as an anomaly by the algorithm, then the ith row of $\mathbf{P}$ would be $\mathbf{P}_i = [1, 0]$. Following are the three categories of the algorithms.

1) *Feature-Based Graph Anomaly Detection:* These methods define the suspiciousness score of node u based on a pair of its particular features a_u and b_u [6], [11]

$$\text{suspiciousness}(u) = |b_u - \hat{b}_u| \quad \text{or} \quad \frac{\max(b_u, \hat{b}_u)}{\min(b_u, \hat{b}_u)} \cdot \log(|b_u - \hat{b}_u| + 1) \tag{4}$$

where $\hat{b}_u$ is the predicted feature value based on the observed a_u. Intuitively, the measure is the "distance to fitting line (a, b)." Akoglu *et al.* [6] adopted four basic features such as number of neighbors, number of edges, total weight, and principal eigenvalue of the weighted adjacency matrix. Power laws were observed between the features with a large population of nodes (i.e., $b_u \propto {a_u}^{\gamma}$, γ is a constant). Big distance to the power-law fitting line indicates the role of graph anomalies. Jiang *et al.* [11] proposed two high-order features: one is called synchronicity, which describes how similar a node's neighbors are with each other in the space of basic features (e.g., degree, PageRank); the other is called normality that describes how similar the neighbor nodes are with every node in the space. They found the synchronicity had a parabolic lower limit of the normality (i.e., $\text{sync}_u \propto \alpha \cdot {\text{norm}_u}^2 + \beta$, α and β are constants) and designed a suspiciousness scoring function to catch the suspicious nodes. Big synchronicity and small normality indicate suspiciousness.

2) *Structure-Based Graph Anomaly Detection:* These methods define the suspiciousness score using the graph structure. They assume that the majority of users have low suspiciousness score. Then users with high suspiciousness scores can be reported as anomalies [16]

$$\text{suspiciousness}(u) = 1 - \frac{\sum_{v\in\mathcal{N}_{\mathcal{V}}(u)} \hat{R}(u, v) + \alpha_1 \cdot \mu_f + \alpha_2 \cdot \prod_U(u)}{|\mathcal{N}_{\mathcal{V}}(u)| + \alpha_1 + \alpha_2} \tag{5}$$

where $\hat{R}(u, v)$ is the normality score of the rating from u to item v, μ_f is the prior belief of u's normality score given by BIRDNEST [18], $\prod_U(u)$ is u's behavior normality score, and α_1 and α_2 are constants.

3) *Model-Based Graph Anomaly Detection:* The idea behind these methods is that the majority of the graphs, or say, the structural dependence, can be learned by a specific graph model (e.g., compression model, generative model) and the anomalies deviate significantly from the model. Chakrabarti [19] and Shah *et al.* [20] used a compression scheme based on the minimum description length (MDL) philosophy. Therefore, the removal of anomalies led to the maximum reduction in compression cost. Gao *et al.* [21] used the hidden Markov random fields (HMRF) model to characterize normal communities and assumed that the anomalies follow a uniform distribution.

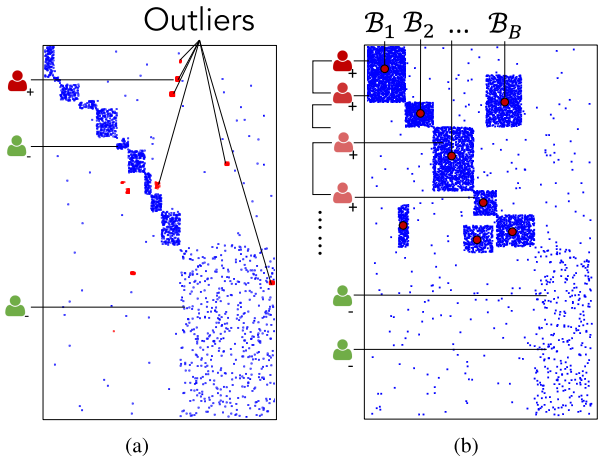

Fig. 2. Example of reordered adjacency matrices containing different graph anomalies. Individual anomaly detection algorithms detect suspicious user nodes that show distinct outlier patterns that differentiate them from the majority. Group anomaly detection algorithms locate the user node groups that form dense subgraphs, indicating they are all suspicious users. (a) Individual Anomalies. (b) Group Anomalies.

2) Group Anomaly Detection Algorithms: defined measurements on how suspicious a subgraph is with respect to the size and high density in a large graph, then employed an efficient algorithm scheme (e.g., greedy search) to detect the subgraphs of high suspiciousness. Suppose the detection algorithm finds B dense subgraphs (blocks) $\{\mathcal{B}_i = (\mathcal{U}_{b,i} \subseteq \mathcal{U}, \mathcal{V}_i \subseteq \mathcal{V})\}_{i=1}^{B}$, where $\mathcal{U}_{b,i}$ and $\mathcal{V}_i$ denote the set of user nodes and item nodes in block $\mathcal{B}_i$, respectively. Following the previous notations, we also denote the output of these algorithms as a binary matrix $\mathbf{P} \in \{0, 1\}^{|\mathcal{U}|\times(B+1)}$, where each line of $\mathbf{P}$ is a one-hot vector indicating which one of the dense blocks does the user belong to. In the case that user u_i does not belong to any of the B dense blocks, the ith row of $\mathbf{P}$ would be $\mathbf{P}_i = [0, \ldots, 0, 1]$ as the last column of $\mathbf{P}$ indicates that the node is not in any dense block.

For each block $\mathcal{B}_i$, we denote the size by $n_i = |\mathcal{U}_{b,i}|$ and $m_i = |\mathcal{V}_i|$; we denote the number of ratings in the block by $c_i = |\{r_{u,v} > 0 | u \in \mathcal{U}_{b,i}, v \in \mathcal{V}_i\}|$. The suspiciousness score is defined in different ways in different approaches.

1) *Average Degree (AD)* [22], [23]

$$\text{susp}^{\text{AD}}(\mathcal{B}_i) = \frac{c_i}{n_i}. \tag{6}$$

2) *Singular Value (SV)* [24], [25]

$$\text{susp}^{\text{SV}}(\mathcal{B}_i) = \frac{c_i}{\sqrt{n_i \cdot m_i}}. \tag{7}$$

3) *Kullback–Leibler divergence of Density (KL)* [26]

$$\text{susp}^{\text{KL}}(\mathcal{B}_i) = n_i \cdot m_i \cdot D_{\text{KL}}(\rho_i \parallel p) \tag{8}$$

where $\rho_i = c_i/(n_i \cdot m_i)$ is density of the block in the adjacency matrix, $p = (|\{r_{u,v} > 0 | u \in \mathcal{U}, v \in \mathcal{V}\}|)/(|\mathcal{U}| \cdot |\mathcal{V}|)$ is the density of the entire data matrix, and $D_{\text{KL}}(\rho_i \parallel p) = p - \rho + \rho \log(\rho/p)$ is the KL divergence between ρ_i and p.

D. Distribution-Aware Anomaly Margin Loss

Traditionally, to train the GNNs in an unsupervised manner, RW-based loss functions are often applied to learn the output representations, $\mathbf{z}_u, \forall u \in \mathcal{U}$, and to tune the weight matrices $\mathbf{W}^k, \forall k \in \{1, \ldots, K\}$ using stochastic gradient descent. The RW-based loss encourages nearby nodes to have similar representations as well as enforcing the representations of disparate nodes to be distinct, which can be formatted as [4]

$$\mathcal{L}_{\text{RW}}(u) = \mathbb{E}_{u_+ \sim \mathcal{U}_{u+}, u_- \sim \mathcal{U}_{u-}} \max\{0, \mathbf{z}_u^T \mathbf{z}_{u_-} - \mathbf{z}_u^T \mathbf{z}_{u_+} + \Delta\} \tag{9}$$

where Δ denotes a fixed margin hyperparameter, $\mathcal{U}_{u+}$ denotes the set of user nodes that are reachable with a fixed length random-walk starting from u, and $\mathcal{U}_{u-}$ denotes $\mathcal{U} \setminus \mathcal{U}_+$. However, as we mentioned before, it is neither proper nor effective when the task is to detect anomalies on graphs because of the imbalance problem of the data for anomaly detection. Fortunately, previous research have been done on learning with imbalanced data [12], [27], [28]. Here, we propose a class-distribution-aware margin loss function that is able to utilize the results of the pattern mining algorithms.

Let y_u denote the the label of user node u (note that user nodes in different anomalous groups should have different labels here). We assume that the class-conditional distribution $\mathcal{P}(u|y_u)$ is the same at training and testing. Then, let $\mathcal{P}_j$ denote the class-conditional distribution, that is, $\mathcal{P}_j = (u|y_u = j)$. For our GNN model, $f : \mathcal{U} \to \mathbb{R}^d$, we use function $g : \mathcal{U} \times \mathcal{U} \to \mathbb{R}$ to denote the similarity of the representations of any two user nodes u and u'

$$g(u, u') = f(u)^T \cdot f(u') \tag{10}$$

and $L_{\text{bal}}[g]$ to denote the standard 0–1 test error on the balanced data distribution

$$L_{\text{bal}}[g] = \Pr_{(u,j)\sim\mathcal{P}_{\text{bal}}}\left[\min_{y_{u_+}=j} g(u, u_+) < \max_{y_{u_-}\neq j} g(u, u_-)\right]. \tag{11}$$

The error L_j for class j is then defined similarly as

$$L_j[g] = \Pr_{u\sim\mathcal{P}_j}\left[\min_{y_{u_+}=j} g(u, u_+) < \max_{y_{u_-}\neq j} g(u, u_-)\right]. \tag{12}$$

Let n_j be the number of user nodes in class j and $S_j = u : y_u = j$ denote the set of user nodes with label j. Define the training margin for class j as

$$\gamma_j = \min_{u\in S_j}\left(\min_{y_{u_+}=j} g(u, u_+) - \max_{y_{u_-}\neq j} g(u, u_-)\right) \tag{13}$$

where $\gamma_{\min} = \min\{\gamma_1, \ldots, \gamma_j\}$ is the widely used training margin in previous studies [27]. Then, we let $L_{\gamma,j}$ denote the margin loss for class j when training

$$L_{\gamma,j}[g] = \Pr_{u\sim\mathcal{P}_j}\left[\min_{y_{u_+}=j} g(u, u_+) < \max_{y_{u_-}\neq j} g(u, v') + \gamma\right] \tag{14}$$

and let $\hat{L}_{\gamma,j}$ denote its empirical variant. For a hypothesis class $\mathcal{G}$, we use $\hat{\mathfrak{R}}(\mathcal{G})$ to denote the empirical Rademacher complexity of margin for class j

$$\hat{\mathfrak{R}}_j(\mathcal{G}) = \frac{1}{n_j}\mathbb{E}_\sigma\left[\sup_{g\in\mathcal{G}}\sum_{u\in S_j}\sigma_u\left[\min_{y_{u_+}=j} g(u, u_+) - \max_{y_{u_-}\neq j} g(u, u_-)\right]\right] \tag{15}$$

where σ is a vector of independent identically distributed (i.i.d.) uniform $\{-1, +1\}$ bits. Here, we consider the bound below for balanced test distribution by considering the margin of each class, which allows us to design distribution-aware margin loss function that is suitable for the imbalanced data.

Theorem 1: With probability $1 - \delta$ over the randomness of the training data, for all choices of class-dependent margins $\gamma_1, \gamma_2, \ldots, \gamma_k > 0$, all hypotheses $g \in \mathcal{G}$ will have balanced-class generalization bounded by

$$L_{\text{bal}}[g] \le \frac{1}{k}\left(\sum_{j=1}^{k}\hat{L}_{\gamma_j,j}[g] + \frac{4}{\gamma_j}\hat{\mathfrak{R}}_j(\mathcal{G}) + \varepsilon_j(\gamma_j)\right) \tag{16}$$

where $\varepsilon_j(\gamma) \triangleq ((\log\log_2(2\max_{u,v\in\mathcal{U},g\in\mathcal{G}}|g(u,v)|)/\gamma + \log(2c)/\delta)/n_j)^{1/2}$ is typically a low-order term in n_j. Concretely, the Rademacher complexity $\hat{\mathfrak{R}}_j(\mathcal{G})$ will typically scale as $((C(\mathcal{G}))/n_j)^{1/2}$ for some complexity measure $C(\mathcal{G})$, in which case

$$L_{\text{bal}}[g] \le \frac{1}{k}\left(\sum_{j=1}^{k}\hat{L}_{\gamma_j,j}[g] + \frac{4}{\gamma_j}\sqrt{\frac{C(\mathcal{G})}{n_j}} + \varepsilon_j(\gamma_j)\right). \tag{17}$$

Note that although the losses and empirical Rademacher complexity of margins are defined different from those in Theorem 2 in [12], the above inequality still holds. For the coherence of reading, the proof of Theorem 1 is provided in Section II-E.

The balanced generalization error bound [see (17)] suggests that in order to improve the generalization of minority classes, we should enforce larger margins for them. However, manually assigning larger margins for minority classes may lead to suboptimal margins for the frequent class and, hence, hurt the model's performance. Thus, here, we take the binary classification problem as an example of showing how to obtain the optimal tradeoff.

When $k = 2$, we aim to optimize the balanced generalization error bound in (17), which can be simplified to

(after removing constant factors, common factor $C(\mathcal{G})$ and low-order term $\varepsilon_j(\gamma_j)$) [12]

$$\frac{1}{\gamma_1\sqrt{n_1}}+\frac{1}{\gamma_2\sqrt{n_2}}. \tag{18}$$

Although the it is hard to get the optimal margins with the above equation as they are complicate functions of the parameters in $g(\cdot)$, we can figure out the relative scales between the two margins. Suppose we have $\gamma_1^*, \gamma_2^* > 0$ that minimize the equation above, we observe that any $\gamma_1' = \gamma_1^* - \delta$ and $\gamma_2' = \gamma_2^* + \delta$ (where $-2\gamma_2^* < \delta < \gamma_1^*$) can be realized by the same parameters with a shifted bias term. Therefore, for γ_1^*, γ_2^* to be optimal, the following inequality must be satisfied [12]:

$$\frac{1}{\gamma_1^*\sqrt{n_1}}+\frac{1}{\gamma_2^*\sqrt{n_2}} \le \frac{1}{\left(\gamma_1^*-\delta\right)\sqrt{n_1}}+\frac{1}{\left(\gamma_2^*+\delta\right)\sqrt{n_2}} \tag{19}$$

which implies that

$$\gamma_1^* \propto n_1^{-1/4}, \quad \text{and } \gamma_2^* \propto n_2^{-1/4}. \tag{20}$$

Given the tradeoff above, we can define our margin loss as

$$\mathcal{L}(u) = \max\left\{0, \max_{y_{v'}\neq y_u} g\left(u, v'\right) - \min_{y_v = y_u} g(u, v) + \Delta_{y_u}\right\}$$
$$\text{where } \Delta_{y_u} = \frac{C}{n_{y_u}^{1/4}}. \tag{21}$$

Here, C is a constant hyperparameter. When applying the above loss function on real-world graphs, it is impossible to enumerate all node pairs when calculating the minimum and maximum distances. Hence, we use positive and negative sampling to approximate the distances. That is, we propose the following margin loss function in our PAMFUL framework:

$$\mathcal{L}(u) = \mathbb{E}_{u_+\sim\mathcal{U}_{u+}, u_-\sim\mathcal{U}_{u-}} \max\left\{0, g(u, u_-) - g(u, u_+) + \Delta_{y_u}\right\}$$
$$\text{where } \Delta_{y_u} = \frac{C}{n_{y_u}^{1/4}}. \tag{22}$$

Here, $\mathcal{U}_{u+}$ denotes the set of user nodes that has the same label as u, $\mathcal{U}_{u-}$ denotes $\mathcal{U} \setminus \mathcal{U}_{u+}$, and $n_{y_u} = |\mathcal{U}_{u+}|$.

1) Positive and Negative Sampling: When applying the above distribution-aware anomaly margin loss [see (22)] for the training of GNNs, we are not aware of the ground-truth labels for the users as the task is unsupervised. Therefore, we utilize the results of graph pattern mining algorithms to estimate the user node sets $\mathcal{U}_{u+}$ and $\mathcal{U}_{u-}$ for each user node u, which is also the global pattern information learned by the algorithms.

With the pattern feature $\mathbf{P}$ from pattern mining algorithms, we now can sample positive and negative nodes for each user node based on their involvement in global patterns. That is, when sampling for each user node $u \in \mathcal{U}$ in the loss function [see (22)], we let

$$\mathcal{U}_{u+} = \{v | \forall v \in \mathcal{U}, \mathbf{P}_v = \mathbf{P}_u\}$$
$$\mathcal{U}_{u-} = \{v | \forall v \in \mathcal{U}, \mathbf{P}_v \neq \mathbf{P}_u\}. \tag{23}$$

With the positive and negative user node sets defined above for each user node, the model can effectively encourage the nodes with the same global patterns (e.g., both are individual anomalies or belong to the same anomalous group) to have similar representations. Moreover, it also enforces that the representations of pairs of nodes with different global patterns (e.g., anomalies and normal nodes; nodes belong to different anomalous groups) to be distinct in the latent space.

E. Proofs

To prove Theorem 1, we first need to get familiar with the following theorem for standard margin-based generalization bounded (Theorem 5 in [28]) in our notation.

Theorem 2 ([27], [28]): Consider and arbitrary function class $\mathcal{G}$ such that $\forall g \in \mathcal{G}$, we have $\sup_{x\in\mathcal{X}} \le C$. Then, with probability at least $1-\delta$ over the sample, for all margins $\gamma > 0$ and all $g \in \mathcal{G}$, we have

$$L[g] \le K_\gamma[g] + \frac{4}{\gamma_j}\mathfrak{R}_n(\mathcal{G}) + \sqrt{\frac{\log\log_2\frac{4C}{\gamma}}{n}} + \sqrt{\frac{\log\frac{1}{\delta}}{2n}} \tag{24}$$

where $K_\gamma[f]$ is the fraction of the data that have γ-margin mistakes.

Proof [28]: Let $l_\gamma(t)$ be defined as

$$l_\gamma(t) = \begin{cases} 1, & t \le 0 \\ 1 - \dfrac{t}{\gamma}, & 0 < t < \gamma \\ 0, & t \ge \gamma. \end{cases} \tag{25}$$

For $j = 0, 1, \ldots,$ set $\gamma_j = C/2^j$ and $\delta_j = \delta/(i+1)^2$. Applying Theorem 3 in [28] to the loss function l_{γ_j}, which has Lipschitz constant $1/\gamma_j$, we get the following inequality with probability at least $1-\delta_j$ for all $g \in \mathcal{G}$:

$$L_{\gamma_j}[g] \le \hat{L}_{\gamma_j}[g] + \frac{2}{\gamma_j}\mathfrak{R}_n(\mathcal{G}) + \sqrt{\frac{\log\frac{1}{\delta}}{2n}}. \tag{26}$$

Note that for any $\gamma > 0$ and any $g \in \mathcal{G}$, $L[g] \le J_\gamma[j]$ and $\hat{L}_\gamma[f] \le K_\gamma[f]$. Hence, with probability at least $1-\delta_i$, for all $g \in \mathcal{G}$, we have

$$L[g] \le K_{\gamma_j}[g] + \frac{2}{\gamma_j}\mathfrak{R}_n(\mathcal{G}) + \sqrt{\frac{\log\frac{1}{\delta}}{2n}}. \tag{27}$$

Taking union bound over all j, we get that with probability at least $1 - \pi^2\delta/6 \ge 1 - 2\delta$, for all j and all $g \in \mathcal{G}$, we still have

$$L[g] \le K_{\gamma_j}[g] + \frac{2}{\gamma_j}\mathfrak{R}_n(\mathcal{G}) + \sqrt{\frac{\log\frac{1}{\delta}}{2n}}. \tag{28}$$

Note that because $|g(x)| \le C$, $\hat{L}_C[g] = 1$ and so that bound claimed in the theorem holds trivially for $\gamma \ge C$. For $\gamma < C$ find the j such that $\gamma_j \le \gamma < \gamma_{j-1}$. Note that this means $j \le \log_2(C/\gamma) + 1$. Because $K_{\gamma_j}[g] \le K_\gamma[g]$, $1/\gamma_j \le 2/\gamma$, and $\log(1/\delta_j) \le \log(1/\delta) + 2\log\log_2(4C/\gamma)$, we have

$$L[g] \le K_\gamma[g] + \frac{4}{\gamma_j}\mathfrak{R}_n(\mathcal{G}) + \sqrt{\frac{2\log\log_2\frac{4C}{\gamma} + \log(1/\delta)}{2n}}. \tag{29}$$

□

Now we lead to the proof of Theorem 1:

Proof: We first prove the generalization seperately for each class j. Let $L_j[g]$ denote the 0–1 test error of classifier g on examples drawn from $\mathcal{P}_j$. Because all examples of class j are a set of n_j, i.i.d. draws from the conditional distribution $\mathcal{P}_j$; and we can apply the standard margin-based generalization bound (Theorem 2 [27], [28]) and obtain the following bound with probability $1-\delta/c$, for all $\gamma_j > 0$ and $g \in \mathcal{G}$:

$$L_j[g] \leq \hat{L}_{\gamma_j,j}[g] + \frac{4}{\gamma_k}\hat{\mathfrak{R}}_j(\mathcal{G}) + \sqrt{\frac{\log\log_2\left(\frac{2\max_{u,v\in\mathcal{U},g\in\mathcal{G}}|g(u,v)|}{\gamma}\right)}{n_j}} + \sqrt{\frac{\log\frac{2c}{\delta}}{n_j}}. \quad (30)$$

Because $L_{\text{bal}} = (1/k)\sum_{j=1}^{k} L_j$, we can union the above bound over all classes and average (30) to get the generalized bound. □

III. Experiments

In this section, we evaluate the proposed PamFul for anomalous user detection on two real-world datasets. Our code package and datasets can be found on GitHub.[1]

A. Experimental Settings

1) Datasets: **Bitcoin-Alpha** is a trust network of Bitcoin trading users on the Alpha platform [29], where each edge indicates a rating from one user to another with a rating score. The network has 3275 user nodes and 3742 items nodes. Kumar *et al.* [16] labeled 214 users: 83 and 131 are labeled as suspicious users and benign users, respectively. The raw feature vector is concatenated by three parts: 1) a one-hot vector indicating the degree of the user node; 2) the summation of one-hot vectors that each represents a rating score given by this user; and 3) the summation of one-hot vectors where each hot represents a time interval between two consecutive ratings.

Weibo is a user-posts-hashtag graph from a Twitter-like micro-blogging platform (Tencent Weibo). It has 8405 users and 61 964 hashtags [30]. The weight of user-hashtag edge is the number of the particular hashtag the user posted. Temporal information was used to label the users. The algorithm in [30] assumed that posting two messages within a specific number of seconds such as 10, 15, 30, 45, and 60 is a suspicious event. If a user made at least five suspicious events, he/she is labeled as a suspicious user; if a user made no suspicious event, he/she is a benign user. So, we have 868 suspicious users and 7537 benign users. Because the ground truth was generated using time information, we do not use timestamps to create raw user features. Therefore, the raw feature vector has two parts.

1) For each user, we first sum up one-hot vectors. Each hot represents the location where a micro-blog post was made. Then, we reduce #dimensions of the location features to 100 using singular value decomposition (SVD).
2) For each user, we reduce #dimensions of bag-of-words features to 300.

[1]https://github.com/zhao-tong/Graph-Anomaly-Loss

2) Baseline Methods and PamFul *Variants:*

a) Unsupervised group anomaly detection methods:

1) Fraudar *[5]:* It catches suspicious group anomalies with theoretical bounded densities for camouflage.
2) CatchSync *[11]:* It captures the synchronized behavior patterns in rating networks and social networks.
3) LockInfer *[10]:* It uses singular vectors of adjacency matrix to find anomalous groups of users in spectral subspaces.

b) Unsupervised individual anomaly detection methods:

1) Fraudar_r*:* It is the reverse of Fraudar. We use Fraudar to detect first several dense groups until the remaining density is very low and report the remaining users as anomalies.
2) LockInfer_r*:* It is the reverse of LockInfer. We use LockInfer to detect all user groups in spectral subspace and report users who are not contained in any group as anomalies.
3) FraudEagle *[9]:* It uses a belief propagation-based algorithm to give a fraud score to each user. Outlying users who behave more different than the majority are given higher fraud scores.
4) REV2 *[16]:* It uses an iterative algorithm to find unfair users whose behaviors can be considered anomalies compared with the majority.

c) Unsupervised node embedding methods:

1) Node2Vec *[31]:* It uses biased RWs to capture the homophily and local structure information of the network. Hyperparameters $p, q \in \{0.25, 0.5, 1, 2, 4\}$ control the search bias.
2) LINE *[32]:* It uses first- and second- order proximity to capture the local and 2-hop structure of the network via edge sampling.
3) BiNE *[33]:* It learns the representations of vertices in a bipartite network. Biased RWs are conducted to preserve the long-tail distribution of vertices.

d) Unsupervised GNN-based methods:

1) GCN *[1]:* It is a spectral-based GNN that learns node embeddings via a localized first-order approximation of spectral graph convolutions. It uses an unsupervised RW-based loss function.
2) GSAGE *[2]:* It is a GNN that enables specifying different weights to different nodes in a neighborhood. It uses an unsupervised RW-based loss function.
3) GAT *[15]:* It is a GNN that learns node embeddings inductively from its own feature and the aggregated features of its neighbors. It uses an unsupervised RW-based loss function.
4) Dominant *[34]:* It is a graph auto-encoder-based deep neural network model for graph anomaly detection. It reconstructs the graph structure and node attributes to find the anomaly nodes.

In summary, we compare among **14** baseline methods, including 3 group anomaly detection methods, 4 individual anomaly detection methods, 3 graph embedding methods, and 4 unsupervised GNN methods. We also compare PamFul itself among **10** variants: 8 with GSAGE and different

pattern mining algorithms (+FRAUDAR, +CATCHSYNC, +LOCKINFER, +FRAUDAR_R, +LOCKINFER_R, +FRAUDEAGLE, +BIRDNEST, +REV2), and 2 with other GNN encoders (GCN, GAT).

3) Implementation Details: For all graph embedding/representation learning methods including ours, the size of representation is 128 and the total sampling size is 100 times the number of user nodes. For all GNN methods, the number of layers are set as 2. For GSAGE [2], we use the Mean aggregator. For GAT [15], because of its high computational complexity, the number of heads for hidden layers' self-attention is set as 1 in order to have a hidden size of 128. For NODE2VEC, we use grid search to find the optimal hyperparameters p, q. For FRAUDAR [5], we use log-weighted AD as the suspiciousness metric. Parameters for other methods are set to typical values as used in previous studies. During the training of PAMFUL, a lightly weighted RW loss is used as a regularization term on the GNN encoder. For fair comparison, the collaborative aggregation described in Section II-B is used on all GNNs. We randomly split the users into training/validation/testing sets with the ratio of 3:1:2 for Weibo dataset and 2:1:1 for Bitcoin-Alpha dataset, as the number of labeled users in Bitcoin-Alpha is relatively small. Moreover, we report the median performance of five different training/validation/testing splits to further diminish the impact of limited evaluation samples.

4) Evaluation Settings: For all graph embedding/representation learning methods, we use a three-layer fully connected feed-forward neural network [multi-layer perceptron (MLP)] as the binary classification model to evaluate the learned representations. For DOMINANT [34], the output anomalous ranking score is used as the input of MLP classifier. For each dataset, the labeled users are divided into a training set, validation set, and testing set. The MLP model is trained on the training set, and the evaluation results on the testing set are reported when the model achieves the highest F1 score on validation set.

For all methods, we report evaluation metrics of Precision, Recall, F1 score, area under receiver operating characteristic (ROC) curve (AUC), and precision–recall curve for selective methods.

B. Results on Bitcoin-Alpha

Table I presents the performance of PAMFUL and all baselines for anomaly detection on Bitcoin-Alpha. Fig. 4(a), (c), and (e) presents the precision–recall curves of PAMFUL with three different GNN encoders (i.e., GCN, GAT, GSAGE) and two types of pattern mining-based anomaly detection algorithms.

1) *The Suspicious Behaviors on Bitcoin-Alpha Are More Likely to Form Individual Anomalies:* Most individual anomaly detection methods perform better than group anomaly detection methods. For example, FRAUDAR_R achieved an F1 of 0.6733, while CATCHSYNC had an F1 of 0.5616.
2) *Graph Representation Learning Models, Including GNNs, Combine Both Node Feature and Graph Structural Information, But the Improvement by Those Is Not Significant:* For example, NODE2VEC achieved an F1 of 0.6977 and FRAUDAR_R achieved an F1 of 0.6733. The reason is that the individual anomaly's local neighbors may not be anomalies.
3) PAMFUL *Variants Outperform All Baseline Methods:* PAMFUL with GSAGE and individual anomaly detection algorithm FRAUDAR_R achieved the best performance: An F1 of 0.7568, an average precision (AP) of 0.8221, and an AUC of 0.8556. It outperformed the best graph representation learning method GCN relatively by $+$**8.1%** and $+$**14.6%** on the two metrics. And it outperforms the best anomaly detection method FRAUDAR_R relatively by $+$**12.4%** and $+$**11.7%**.

TABLE I
PAMFUL WITH GSAGE AND INDIVIDUAL ANOMALY DETECTION ALGORITHMS PERFORM THE BEST ON BITCOIN-ALPHA DATASET

	Precision	Recall	F1	AUC
Unsupervised pattern mining algorithms				
FRAUDAR [35]	0.3879	1.0000	0.5589	0.2924
CATCHSYNC [11]	0.3923	0.9880	0.5616	0.5426
LOCKINFER [10]	0.3879	1.0000	0.5589	0.2342
FRAUDAR_R	0.5714	0.8193	0.6733	0.7659
LOCKINFER_R	0.5512	0.8434	0.6667	0.7593
FRAUDEAGLE [9]	0.4205	0.8916	0.5714	0.5149
REV2 [16]	0.6471	0.6627	0.6548	0.7094
Graph embedding methods				
NODE2VEC [31]	0.6522	0.7500	0.6977	0.7606
LINE [32]	0.4722	0.8500	0.6071	0.6455
BINE [33]	0.5143	0.9000	0.6545	0.7409
Graph neural network methods (GNNs)				
GCN [1]	0.5526	0.9545	0.7000	0.7463
GSAGE [2]	0.6471	0.6111	0.6286	0.7238
GAT [15]	0.6071	0.7727	0.6800	0.7199
DOMINANT [34]	0.4231	1.0000	0.5946	0.3966
PAMFUL with GSAGE + group anomaly detection methods				
+FRAUDAR	0.7000	0.7368	0.7179	0.8282
+CATCHSYNC	0.7895	0.6818	0.7317	0.7742
+LOCKINFER	0.7143	0.7500	0.7317	0.7970
PAMFUL with GSAGE + individual anomaly detection methods				
+FRAUDAR_R	0.7368	0.7778	**0.7568**	**0.8556**
+LOCKINFER_R	0.6800	0.7727	0.7234	0.7874
+FRAUDEAGLE	0.9231	0.6000	0.7273	0.8303
+BIRDNEST	0.7368	0.6364	0.6829	0.7683
+REV2	0.7500	0.7500	0.7500	0.8030
PAMFUL with Other GNNs + FRAUDAR_R				
with GCN	0.7143	0.7895	0.7500	0.8498
with GAT	0.6333	0.8636	0.7308	0.7273

Fig. 3 visualize the user embeddings. Fig. 4(a), (c), and (e) compare the precision–recall curves of NODE2VEC, GNN methods, and PAMFUL with two anomaly detection algorithms (REV2 and FRAUDAR_R) and different GNN encoders. We observe that GCN performs the best among the three GNNs, and PAMFUL variants perform the best.

C. Results on Weibo

Table II presents the performances on Weibo dataset. Note that because the labels are seriously biased, F1 is more

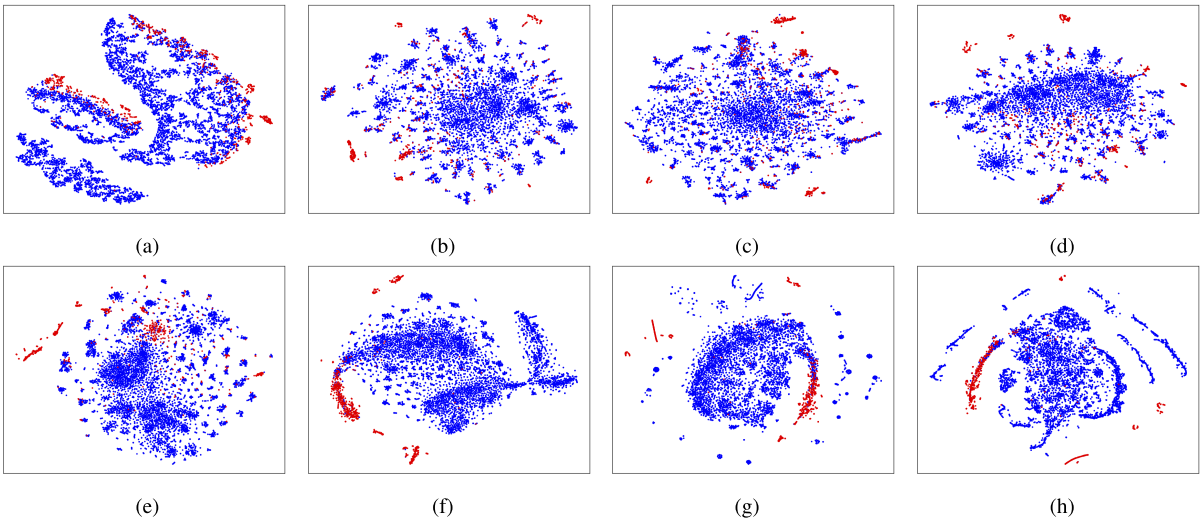

Fig. 3. Visualizing user embeddings in Weibo data. Blue dots represent benign users and red dots represent anomalous users. Representations learned by PAMFUL (e)–(h) are better than those by baselines (b)–(d). (a) NODE2VEC [31]. (b) GCN [1]. (c) GSAGE [2]. (d) GAT [15]. (e) PAMFUL with GSAGE + Fraudar. (f) PAMFUL with GSAGE + LOCKINFER. (g) PAMFUL with GCN + LOCKINFER. (h) PAMFUL with GAT + LOCKINFER.

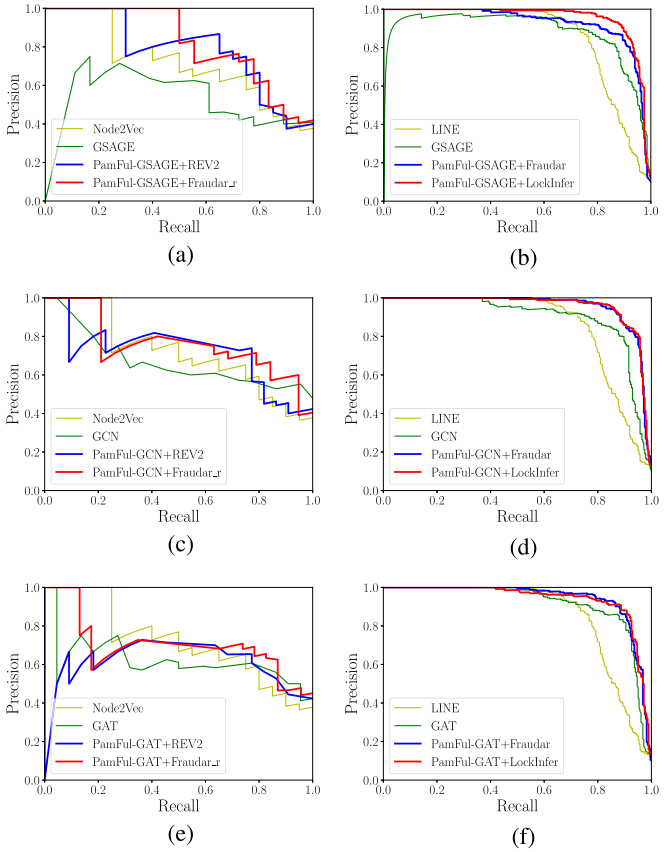

Fig. 4. Precision–recall curves are presented to compare the best graph embedding baseline, a graph neural algorithm, and two best PAMFUL variants. (a) PAMFUL–GSAGE on Bitcoin. (b) PAMFUL–GSAGE on Weibo. (c) PAMFUL–GCN on Bitcoin. (d) PAMFUL–GCN on Weibo. (e) PAMFUL–GAT on Bitcoin. (f) PAMFUL–GAT on Weibo.

representative than AUC on this dataset. Fig. 4(b), (d), and (f) shows the precision–recall curves of PAMFUL with three different GNN encoders and two pattern mining algorithms.

1) *The Anomalous Users on Weibo Are More Likely to Form Group Anomalies:* Group anomaly detection methods perform much better than individual anomaly detection methods. FRAUDAR achieved an F1 of 0.7540, while FRAUDEAGLE only made an F1 of 0.4102. The reason is that fraudsters had to post a large number of messages in a group to inflate the popularity of hashtag.
2) *Local Neighborhood Is More Informative Than Pure Global Structure:* Graph embedding models perform better than the pattern mining algorithms. For example, LINE achieved an F1 of 0.8105, while FRAUDAR achieved an F1 of 0.7540. The models that use local structures for embedding aggregation can preserve the user similarity of being in the same anomalous groups.
3) PAMFUL *Variants Outperform All Baseline Methods:* PAMFUL with GSAGE and group anomaly detection algorithm LOCKINFER performed the best: An F1 of 0.9042 and an AUC of 0.9843. It outperformed the best graph embedding method LINE relatively by +**11.6%** and +**4.4%** on the two metrics. And it outperformed GSAGE relatively by +**7.3%** and +**0.6%**.

Fig. 4(b), (d), and (f) compares the precision–recall curves of LINE, GNNs, and PAMFUL with two anomaly detection algorithms (FRAUDAR and LOCKINFER). PAMFUL variants with GSAGE perform the best.

Embedding Visualization: Fig. 3 shows the embeddings of users in the Weibo dataset given by four baselines of graph embedding methods Fig. 3(a)–(d) and our PAMFUL variants Fig. 3(e)–(h). The 128-dim embeddings are projected to 2-dim space via t-distributed stochastic neighbor embedding (t-SNE) [36]. From the plots, we observe that the representations learned by PAMFUL with group anomaly detection algorithms can better separate the *blue* benign users and *red* suspicious users. Specifically, the representations learned by the baseline methods Fig. 3(a)–(d) cannot effectively separate the *red* anomalous users and *blue* benign users: most anomalies are in the center area and are close to the benign users. For PAMFUL with GSAGE and FRAUDAR Fig. 3(e), we observe that most anomalies are grouped on the boundaries

TABLE II
PAMFUL WITH GSAGE AND GROUP ANOMALY DETECTION ALGORITHMS PERFORM THE BEST ON WEIBO DATASET

	Precision	Recall	F1	AUC
Unsupervised pattern mining algorithms				
FRAUDAR [5]	0.9624	0.6198	0.7540	0.9079
CATCHSYNC [11]	0.3200	0.8111	0.4589	0.7752
LOCKINFER [10]	0.9318	0.4562	0.6125	0.8674
FRAUDAR_R	0.1033	1.0000	0.1872	0.1915
LOCKINFER_R	0.1033	1.0000	0.1872	0.1326
FRAUDEAGLE [9]	0.3399	0.5173	0.4102	0.6910
REV2 [16]	0.1899	0.6175	0.2905	0.6626
Graph embedding methods				
NODE2VEC [31]	0.9111	0.7218	0.8055	0.9642
LINE [32]	0.8675	0.7606	0.8105	0.9432
BINE [33]	0.3678	0.7007	0.4824	0.8774
Graph neural network methods (GNNs)				
GCN [1]	0.8373	0.8697	0.8532	0.9690
GSAGE [2]	0.8200	0.8662	0.8425	0.9780
GAT [15]	0.8586	0.8768	0.8676	0.9706
DOMINANT [34]	0.3006	0.8627	0.4524	0.8167
PAMFUL with GSAGE + group anomaly detection methods				
+FRAUDAR	0.8669	0.8944	0.8804	0.9777
+CATCHSYNC	0.9067	0.8556	0.8804	0.9685
+LOCKINFER	0.9294	0.8803	**0.9042**	0.9843
PAMFUL with GSAGE + individual anomaly detection methods				
+FRAUDAR_R	0.8547	0.8697	0.8621	0.9697
+LOCKINFER_R	0.8591	0.8803	0.8696	0.9757
+FRAUDEAGLE	0.8581	0.8732	0.8656	0.9730
+BIRDNEST	n/a	n/a	n/a	n/a
+REV2	0.8759	0.8944	0.8850	0.9677
PAMFUL with Other GNNs + LOCKINFER				
with GCN	0.9459	0.8627	0.9024	**0.9853**
with GAT	0.9097	0.8873	0.8984	0.9838

while a small group of anomalies is still remaining in the middle area. We think the reason is that FRAUDAR could not detect that group of users as a dense block. In Fig. 3(f)–(h), our proposed PAMFUL with GNNs and LOCKINFER can better separate the anomalies and benign users. We also notice that in Fig. 3(g) and (h), several crowds of benign users are also grouped on the boundary of the latent space. We remark that these are the smaller benign communities that were detected by LOCKINFER and grouped together in the latent space because of our design of forcing the representations of all nodes in the same community/block to be similar [see (23)]. With the representations learned by PAMFUL, a simple classifier can be easily trained with few labels to detect the anomalies.

D. Ablation Study

To further analyze the effectiveness of PAMFUL, we conduct an ablation study to examine the contribution of different components by having different settings.

1) *GNN:* just the GNN encoder trained by the RW-based loss function [see (9)].
2) PAMFUL-SINGLEGROUP*:* When using group anomaly detection methods as pattern mining algorithm in PAMFUL, instead of separating all anomaly groups as described in (23), treat all anomalous groups as one single group (compressing the second dimension of $\mathbf{P}$ in to 2).
3) PAMFUL *With* FRAUDAR/LOCKINFER*:* PAMFUL with different GNN encoders and different group anomaly detection algorithms (FRAUDAR and LOCKINFER).

TABLE III
ABLATION STUDY RESULTS ON WEIBO DATA

GNN Encoder	Setting	F1	AUC
GSAGE	GNN	0.8425	0.9780
	PAMFUL-SINGLEGROUP	0.8754	0.9757
	PAMFUL-FRAUDAR	0.8804	0.9777
	PAMFUL-LOCKINFER	**0.9042**	**0.9843**
GCN	GNN	0.8532	0.9690
	PAMFUL-SINGLEGROUP	0.8909	0.9775
	PAMFUL-FRAUDAR	0.8982	0.9814
	PAMFUL-LOCKINFER	**0.9024**	**0.9853**
GAT	GNN	0.8676	0.9706
	PAMFUL-SINGLEGROUP	0.8722	0.9754
	PAMFUL-FRAUDAR	0.8822	**0.9921**
	PAMFUL-LOCKINFER	**0.8984**	0.9838

TABLE IV
PAMFUL WITH GCN + LOCKINFER OUTPERFORMS SELECTED BASELINES ON THE WEIBO-2012 DATASET

	Precision	Recall	F1	AUC
LOCKINFER [10]	0.9302	0.4516	0.6080	0.8408
GCN [1]	0.9385	0.8592	0.8971	0.9838
GSAGE [2]	0.9464	0.8185	0.8778	0.9812
PAMFUL	0.9091	0.9653	**0.9363**	**0.9863**

Table III summarizes the results of the ablative study, from which we have the following observations.

1) PAMFUL *Consistently Outperform GNNs:* PAMFUL framework is able to generate node representations that are more suitable for the task of anomaly detection.
2) *Separating Different Anomalous Groups Helps:* PAMFUL outperforms PAMFUL-SINGLEGROUP, which indicates that the design of separating the representations of anomaly users from different groups when using group anomaly detection algorithms (Section II-C) is able to learn representations that are more separable.

To show the stability and generalizability PAMFUL, we evaluate PAMFUL with selected baselines on another Weibo dataset that were collected from a different time span. This Weibo-2012 dataset has 8048 users and 56 573 hashtags, in which 826 (10.3%) users were labeled as anomalies. Table IV summarizes the performance of the best baseline methods and PAMFUL with GCN + LOCKINFER on the Weibo-2012 dataset. We observe that PAMFUL with GCN + LOCKINFER can effectively detect the anomalies in different datasets.

E. Sensitivity and Efficiency Analysis

When transforming the outputs of pattern mining algorithms to the binary pattern feature $\mathbf{P}$ as described in Section II-C,

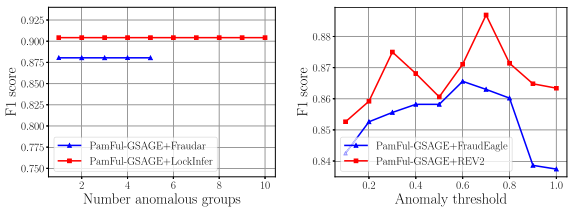

Fig. 5. Parameter sensitivity: On the effectiveness of pattern mining algorithms in PAMFUL. (a) Number of anomalous groups. (b) Anomaly threshold.

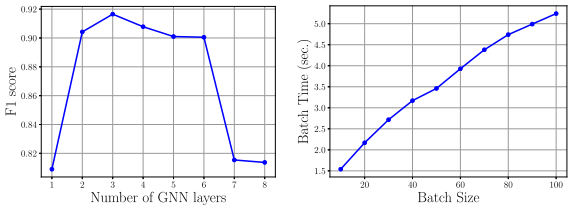

Fig. 6. Depth sensitivity and efficiency analysis. (a) Number of GNN layers. (b) Per-batch training time.

a classification threshold is usually needed to determine which users are labeled as potential anomalies based on the suspiciousness scores. For group anomaly detection algorithms, different groups have their unique labels instead of a too general positive label. With the pattern feature **P** indicating which anomalous group does each user belong to (when applicable), the node embeddings would not change when the threshold changed (and the number of groups changed), as shown in Fig. 5(a). On the other hand, for individual anomaly detection algorithms, the node suspiciousness scores are usually continuous and do not show obvious gap. We need a parameter search to determine the best threshold. Fig. 5(b) shows that the performance of PAMFUL is not sensitive to this parameter.

Fig. 6(a) shows the performance of PAMFUL with GSAGE + LOCKINFER w.r.t. the number of hidden layers in the GNN encoder. Similar to most existing GNN methods, the performance achieves the best performance when the number of hidden layers is not big or too small. Fig. 6(b) reports the average per-batch training time of PAMFUL with GSAGE + LOCKINFER on the Weibo dataset with regard to the batch size. The time cost is linear to batch size, which enables the training of PAMFUL with large graphs.

IV. RELATED WORK

In this section, we survey research work in the last ten years of as many as five related topics.

A. Imbalanced Learning

Learning with imbalanced data has always been a challenging problem for machine learning. Most existing work focused on sampling and generating techniques. These algorithms either under-sample/over-sample the data objects [37], [38] or generate new data objects for the minority classes [13]. Kakade *et al.* [28] proved that generalization error for both linear and nonlinear models with hinge losses is bounded. Recently, Cao *et al.* [12] showed that the error bound could be found on imbalanced datasets. Huang *et al.* [39] studied supervised image representation learning on imbalanced data with different intercluster and interclass margins.

B. Graph Embedding

The goal is to learn node embeddings in a low-dimensional space using random-walk paths or factorized features [31]–[33], [40], [41]. These algorithms are transductive as they directly train node embeddings for individual nodes and require retraining or additional training to generate embeddings for new nodes. DEEPWALK [40] and NODE2VEC [31] learned node embeddings by performing word embedding models WORD2VEC on "corpus" of nodes generated by RW. BINE [33] extended DEEPWALK and optimized for bipartite graphs. ADONE [42] was an unsupervised auto-encoder that learns outlier resistant embeddings.

C. Graph Neural Networks

GNNs are deep learning architectures for graph structured data. The core idea is to learn node representations through local neighborhoods. Many GNN variants have been developed in recent years, following the initial idea of convolution based on spectral graph theory [43]. Many spectral GNNs have since been developed [1], [44]. As spectral GNNs generally operate (expensively) on the full adjacency, spatial-based inductive GNNs that perform graph convolution with neighborhood aggregation became prominent [4], [15], [45]–[47]. More recently, several GNN-based methods [48]–[56] were also proposed for the task of semisupervised or supervised graph anomaly detection.

D. Graph Individual Anomaly Detection

The goal is to find outlier nodes in large graphs [6], [7]. Traditional density-based clustering methods [19], [57] regarded nodes in sparse regions as outliers. Similar approaches have been developed for bipartite graphs [58]. SPOTLIGHT [59] detected outliers in streaming graphs. REV2 [16] was an iterative algorithm that calculated reviewer fairness scores. DOMINANT [34] was an attributed graph auto-encoder that detects anomalous nodes. REPEN [60] learned representations for distance-based outlier detection.

E. Graph Group Anomaly Detection

The goal is to find suspicious nodes by locating dense subgraphs in the graph's adjacency matrix [61]. SPOKEN [24] found the "spokes" pattern on pairs of eigenvectors of graphs. LOCKINFER [10] identified pattern of communities based on singular vectors of graphs. FBOX [25] located mini-scale attacks missed by spectral techniques. CATCHSYNC [11] found the lockstep behaviors made by fraudulent users. Several methods [61]–[63] utilized dense subgraph detection algorithms to find the suspicious dense blocks. Hooi *et al.* [5] and Shin *et al.* [64] showed that the edge density-based suspiciousness of subgraph or subtensor can be maximized with approximation guarantee. MSTREAM [65] and MIDAS [66] focused on detection of group anomalies on edge streams.

V. Conclusion

In this work, we presented a novel synergistic approach PAMFUL that is able to unsupervisedly learn node representations that are tailored for graph anomaly detection. Unlike most existing graph anomaly detection algorithms or graph representation learning methods, PAMFUL combines pattern mining into the feature learning process. Specifically, PAMFUL leverages pattern mining algorithms to guide the training process of the GNN encoder such that the learned representations contain local neighborhood information as well as global structural properties. Experiments on two real-world datasets demonstrated that PAMFUL significantly outperformed 18 baselines, and the representations learned by PAMFUL are shown effective for graph anomaly detection empirically.

References

[1] T. N. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," 2016, *arXiv:1609.02907*. [Online]. Available: http://arxiv.org/abs/1609.02907

[2] W. Hamilton, Z. Ying, and J. Leskovec, "Inductive representation learning on large graphs," in *Proc. NIPS*, 2017, pp. 1–11.

[3] C. Zhang, D. Song, C. Huang, A. Swami, and N. V. Chawla, "Heterogeneous graph neural network," in *Proc. 25th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, Jul. 2019, pp. 793–803.

[4] R. Ying, R. He, K. Chen, P. Eksombatchai, W. L. Hamilton, and J. Leskovec, "Graph convolutional neural networks for web-scale recommender systems," in *Proc. 24th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, 2018, pp. 974–983.

[5] B. Hooi, H. A. Song, A. Beutel, N. Shah, K. Shin, and C. Faloutsos, "FRAUDAR: Bounding graph fraud in the face of camouflage," in *Proc. 22nd ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, Aug. 2016, pp. 895–904.

[6] L. Akoglu, M. McGlohon, and C. Faloutsos, "Oddball: Spotting anomalies in weighted graphs," in *Proc. Pacific-Asia Conf. Knowl. Discovery Data Mining*, 2010, pp. 410–421.

[7] L. Akoglu, H. Tong, and D. Koutra, "Graph based anomaly detection and description: A survey," *Data Mining Knowl. Discovery*, vol. 29, no. 3, pp. 626–688, May 2015.

[8] S. Rayana and L. Akoglu, "Collective opinion spam detection: Bridging review networks and metadata," in *Proc. 21th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, 2015, pp. 985–994.

[9] L. Akoglu, R. Chandy, and C. Faloutsos, "Opinion fraud detection in online reviews by network effects," in *Proc. ICWSM*, 2013, pp. 1–10.

[10] M. Jiang, P. Cui, A. Beutel, C. Faloutsos, and S. Yang, "Inferring lockstep behavior from connectivity pattern in large graphs," *Knowl. Inf. Syst.*, vol. 48, no. 2, pp. 399–428, 2016.

[11] M. Jiang, P. Cui, A. Beutel, C. Faloutsos, and S. Yang, "CatchSync: Catching synchronized behavior in large directed graphs," in *Proc. 20th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, Aug. 2014, pp. 941–950.

[12] K. Cao, C. Wei, A. Gaidon, N. Arechiga, and T. Ma, "Learning imbalanced datasets with label-distribution-aware margin loss," 2019, *arXiv:1906.07413*. [Online]. Available: https://arxiv.org/abs/1906.07413

[13] N. V. Chawla, K. W. Bowyer, L. O. Hall, and W. P. Kegelmeyer, "SMOTE: Synthetic minority over-sampling technique," *J. Artif. Intell. Res.*, vol. 16, pp. 321–357, Jun. 2002.

[14] H. He and E. A. Garcia, "Learning from imbalanced data," *IEEE Trans. Knowl. Data Eng.*, vol. 21, no. 9, pp. 1263–1284, Sep. 2009.

[15] P. Veličković, G. Cucurull, A. Casanova, A. Romero, P. Lio, and Y. Bengio, "Graph attention networks," 2017, *arXiv:1710.10903*. [Online]. Available: https://arxiv.org/abs/1710.10903

[16] S. Kumar, B. Hooi, D. Makhija, M. Kumar, C. Faloutsos, and V. S. Subrahmanian, "REV2: Fraudulent user prediction in rating platforms," in *Proc. 11th ACM Int. Conf. Web Search Data Mining*, Feb. 2018, pp. 333–341.

[17] J. Benesty, J. Chen, Y. Huang, and I. Cohen, "Pearson correlation coefficient," in *Noise Reduction in Speech Processing*. Berlin, Germany: Springer, 2009, pp. 1–4.

[18] B. Hooi *et al.*, "BIRDNEST: Bayesian inference for ratings-fraud detection," in *Proc. SIAM Int. Conf. Data Mining*, Jun. 2016, pp. 495–503.

[19] D. Chakrabarti, "Autopart: Parameter-free graph partitioning and outlier detection," in *Proc. PKDD*, 2004, pp. 112–124.

[20] N. Shah, D. Koutra, T. Zou, B. Gallagher, and C. Faloutsos, "TimeCrunch: Interpretable dynamic graph summarization," in *Proc. 21th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, Aug. 2015, pp. 1055–1064.

[21] J. Gao, F. Liang, W. Fan, C. Wang, Y. Sun, and J. Han, "On community outliers and their efficient detection in information networks," in *Proc. 16th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, 2010, pp. 813–822.

[22] Y. Asahiro, K. Iwama, H. Tamaki, and T. Tokuyama, "Greedily finding a dense subgraph," *J. Algorithms*, vol. 34, no. 2, pp. 203–221, Feb. 2000.

[23] R. Andersen, "A local algorithm for finding dense subgraphs," *ACM Trans. Algorithms*, vol. 6, no. 4, pp. 1–12, Aug. 2010.

[24] B. A. Prakash, A. Sridharan, M. Seshadri, S. Machiraju, and C. Faloutsos, "EigenSpokes: Surprising patterns and scalable community chipping in large graphs," in *Proc. Pacific–Asia Conf. Knowl. Discovery Data Mining*. Berlin, Germany: Springer, 2010, pp. 435–448.

[25] N. Shah, A. Beutel, B. Gallagher, and C. Faloutsos, "Spotting suspicious link behavior with fBox: An adversarial perspective," in *Proc. IEEE Int. Conf. Data Mining*, Dec. 2014, pp. 959–964.

[26] M. Jiang, A. Beutel, P. Cui, B. Hooi, S. Yang, and C. Faloutsos, "Spotting suspicious behaviors in multimodal data: A general metric and algorithms," *IEEE Trans. Knowl. Data Eng.*, vol. 28, no. 8, pp. 2187–2200, Aug. 2016.

[27] V. Koltchinskii and D. Panchenko, "Empirical margin distributions and bounding the generalization error of combined classifiers," *Ann. Statist.*, vol. 30, no. 1, pp. 1–50, Feb. 2002.

[28] S. M. Kakade, K. Sridharan, and A. Tewari, "On the complexity of linear prediction: Risk bounds, margin bounds, and regularization," in *Proc. NIPS*, 2009, pp. 1–10.

[29] S. Kumar, F. Spezzano, V. S. Subrahmanian, and C. Faloutsos, "Edge weight prediction in weighted signed networks," in *Proc. IEEE 16th Int. Conf. Data Mining (ICDM)*, Dec. 2016, pp. 221–230.

[30] M. Jiang, "Catching social media advertisers with strategy analysis," in *Proc. 1st Int. Workshop Comput. Methods CyberSafety*, Oct. 2016, pp. 5–10.

[31] A. Grover and J. Leskovec, "Node2vec: Scalable feature learning for networks," in *Proc. 22nd ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, 2016, pp. 855–864.

[32] J. Tang, M. Qu, M. Wang, M. Zhang, J. Yan, and Q. Mei, "Line: Large-scale information network embedding," in *Proc. 24th Int. Conf. World Wide Web*, 2015, pp. 1067–1077.

[33] M. Gao, L. Chen, X. He, and A. Zhou, "BiNE: Bipartite network embedding," in *Proc. 41st Int. ACM SIGIR Conf. Res. Develop. Inf. Retr.*, Jun. 2018, pp. 715–724.

[34] K. Ding, J. Li, R. Bhanushali, and H. Liu, "Deep anomaly detection on attributed networks," in *Proc. SIAM Int. Conf. Data Mining*, 2019, pp. 594–602.

[35] B. Hooi, K. Shin, H. A. Song, A. Beutel, N. Shah, and C. Faloutsos, "Graph-based fraud detection in the face of camouflage," *ACM Trans. Knowl. Discovery from Data*, vol. 11, no. 4, pp. 1–26, Aug. 2017.

[36] L. V. D. Maaten and G. Hinton, "Visualizing data using t-SNE," *J. Mach. Learn. Res.*, vol. 9, no. 11, pp. 1–27, 2008.

[37] N. V. Chawla, "C4. 5 and imbalanced data sets: Investigating the effect of sampling method, probabilistic estimate, and decision tree structure," in *Proc. ICML*, 2003, p. 66.

[38] D. Mease, A. J. Wyner, and A. Buja, "Boosted classification trees and class probability/quantile estimation," *J. Mach. Learn. Res.*, vol. 8, no. 3, pp. 1–31, 2007.

[39] C. Huang, Y. Li, C. C. Loy, and X. Tang, "Learning deep representation for imbalanced classification," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2016, pp. 5375–5384.

[40] B. Perozzi, R. Al-Rfou, and S. Skiena, "DeepWalk: Online learning of social representations," in *Proc. 20th ACM SIGKDD Int. Conf. Knowl. discovery data mining*, Aug. 2014, pp. 701–710.

[41] D. Wang, M. Jiang, Q. Zeng, Z. Eberhart, and N. V. Chawla, "Multi-type itemset embedding for learning behavior success," in *Proc. 24th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, Jul. 2018, pp. 2397–2406.

[42] S. Bandyopadhyay, S. V. Vivek, and M. N. Murty, "Outlier resistant unsupervised deep architectures for attributed network embedding," in *Proc. 13th Int. Conf. Web Search Data Mining*, Jan. 2020, pp. 25–33.

[43] J. Bruna, W. Zaremba, A. Szlam, and Y. LeCun, "Spectral networks and locally connected networks on graphs," 2013, *arXiv:1312.6203*. [Online]. Available: https://arxiv.org/abs/1312.6203

[44] R. Levie, F. Monti, X. Bresson, and M. M. Bronstein, "CayleyNets: Graph convolutional neural networks with complex rational spectral filters," *IEEE Trans. Signal Process.*, vol. 67, no. 1, pp. 97–109, Jan. 2019.

[45] K. Ding, Y. Li, J. Li, C. Liu, and H. Liu, "Feature interaction-aware graph neural networks," 2019, *arXiv:1908.07110*. [Online]. Available: https://arxiv.org/abs/1908.07110

[46] T. Zhao, Y. Liu, L. Neves, O. Woodford, M. Jiang, and N. Shah, "Data augmentation for graph neural networks," in *Proc. AAAI Conf. Artif. Intell.*, 2021, pp. 11015–11023.

[47] T. Zhao, G. Liu, D. Wang, W. Yu, and M. Jiang, "Counterfactual graph learning for link prediction," 2021, *arXiv:2106.02172*. [Online]. Available: https://arxiv.org/abs/2106.02172

[48] Z. Liu, C. Chen, X. Yang, J. Zhou, X. Li, and L. Song, "Heterogeneous graph neural networks for malicious account detection," in *Proc. 27th ACM Int. Conf. Inf. Knowl. Manage.*, 2018, pp. 2077–2085.

[49] S. Zhang, H. Yin, T. Chen, Q. V. N. Hung, Z. Huang, and L. Cui, "GCN-based user representation learning for unifying robust recommendation and fraudster detection," 2020, *arXiv:2005.10150*. [Online]. Available: https://arxiv.org/abs/2005.10150

[50] Y. Dou, Z. Liu, L. Sun, Y. Deng, H. Peng, and P. S. Yu, "Enhancing graph neural network-based fraud detectors against camouflaged fraudsters," in *Proc. 29th ACM Int. Conf. Inf. Knowl. Manage.*, Oct. 2020, pp. 315–324.

[51] K. Ding, J. Li, and H. Liu, "Interactive anomaly detection on attributed networks," in *Proc. 12th ACM Int. Conf. Web Search Data Mining*, 2019, pp. 357–365.

[52] K. Ding, J. Li, N. Agarwal, and H. Liu, "Inductive anomaly detection on attributed networks," in *Proc. 29th Int. Joint Conf. Artif. Intell.*, 2020, pp. 1288–1294.

[53] T. Zhao, B. Ni, W. Yu, and M. Jiang, "Early anomaly detection by learning and forecasting behavior," 2020, *arXiv:2010.10016*. [Online]. Available: https://arxiv.org/abs/2010.10016

[54] T. Zhao, C. Deng, K. Yu, T. Jiang, D. Wang, and M. Jiang, "Error-bounded graph anomaly loss for GNNs," in *Proc. 29th ACM Int. Conf. Inf. Knowl. Manage.*, Oct. 2020, pp. 1873–1882.

[55] B. Dong *et al.*, "Anomalous event sequence detection," *IEEE Intell. Syst.*, vol. 36, no. 3, pp. 5–13, May 2021.

[56] G. Pang, C. Shen, L. Cao, and A. Hengel, "Deep learning for anomaly detection: A review," *ACM Comput. Surv.*, vol. 54, no. 2, pp. 1–38, 2021.

[57] V. Hautamaki, I. Karkkainen, and P. Franti, "Outlier detection using K-nearest neighbour graph," in *Proc. 17th Int. Conf. Pattern Recognit. (ICPR)*, Aug. 2004, pp. 430–433.

[58] J. Sun, H. Qu, D. Chakrabarti, and C. Faloutsos, "Neighborhood formation and anomaly detection in bipartite graphs," in *Proc. IEEE Int. Conf. Data Mining (ICDM)*, Nov. 2005, p. 8.

[59] D. Eswaran, C. Faloutsos, S. Guha, and N. Mishra, "SpotLight: Detecting anomalies in streaming graphs," in *Proc. 24th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, Jul. 2018, pp. 1378–1386.

[60] G. Pang, L. Cao, L. Chen, and H. Liu, "Learning representations of ultrahigh-dimensional data for random distance-based outlier detection," in *Proc. 24th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, Jul. 2018, pp. 2041–2050.

[61] T. Zhao, M. Malir, and M. Jiang, "Actionable objective optimization for suspicious behavior detection on large bipartite graphs," in *Proc. IEEE Int. Conf. Big Data (Big Data)*, Dec. 2018, pp. 1248–1257.

[62] N. Shah, "FLOCK: Combating astroturfing on livestreaming platforms," in *Proc. 26th Int. Conf. World Wide Web*, Apr. 2017, pp. 1083–1091.

[63] H. Nilforoshan and N. Shah, "SliceNDice: Mining suspicious multi-attribute entity groups with multi-view graphs," in *Proc. IEEE Int. Conf. Data Sci. Adv. Analytics (DSAA)*, Oct. 2019, pp. 351–363.

[64] K. Shin, B. Hooi, and C. Faloutsos, "M-zoom: Fast dense-block detection in tensors with quality guarantees," in *Proc. Joint Eur. Conf. Mach. Learn. Knowl. Discovery Databases*, 2016, pp. 264–280.

[65] S. Bhatia, A. Jain, P. Li, R. Kumar, and B. Hooi, "MSTREAM: Fast anomaly detection in multi-aspect streams," 2020, *arXiv:2009.08451*. [Online]. Available: https://arxiv.org/abs/2009.08451

[66] S. Bhatia, B. Hooi, M. Yoon, K. Shin, and C. Faloutsos, "Midas: Microcluster-based detector of anomalies in edge streams," in *Proc. AAAI Conf. Artif. Intell.*, 2020, pp. 3242–3249.

Tong Zhao received the B.S. degree from Case Western Reserve University, Cleveland, OH, USA, in 2017. He is currently pursuing the Ph.D. degree with the University of Notre Dame, Notre Dame, IN, USA.

His research focuses on graph machine learning computational behavior modeling and anomaly detection.

Tianwen Jiang received the B.E. and Ph.D. degrees from the Department of Computer Science and Technology, Harbin Institute of Technology (HIT), Harbin, China, in 2016 and 2021, respectively.

The work of this article was done when he was a Ph.D. candidate with HIT. He is currently a Researcher with Tencent Inc., Shenzhen, China. He has authored or coauthored more than ten articles on natural language processing and knowledge graph in top conferences and journals of the relevant field such as IEEE/ACM TRANSACTIONS ON COMPUTATIONAL BIOLOGY AND BIOINFORMATICS (TCBB), IEEE TRANSACTIONS ON KNOWLEDGE AND DATA ENGINEERING (TKDE), ACM SIGKDD, EMNLP, IEEE International Conference on Bioinformatics and Biomedicine (BIBM), and ACM Conference on Information and Knowledge Management (CIKM). His research focuses on knowledge mining and structuring from massive text corpora.

Neil Shah received the Ph.D. degree in computer science from the Department of Computer Science, Carnegie Mellon University, Pittsburgh, PA, USA, in 2017.

He is a Research Scientist with Snap Inc., Seattle, WA, USA, with interests in data mining, machine learning, and computational social science on online platforms, with special focus on graph-based modeling for user behavior and misbehavior. His work has resulted in 35+ conference and journal publications, in top venues such as KDD, ICDM, WWW, SDM, DSAA, PAKDD, TKDD, and more, including several best-paper awards. He has also served as an organizer, chair, and on program committees at a number of these. He has had previous research experiences with Lawrence Livermore National Laboratory, Livermore, CA, USA; Microsoft Research, Redmond, WA; and Twitch.tv, San Francisco, CA, USA.

Meng Jiang received the bachelor's and Ph.D. degrees from Tsinghua University, Beijing, China, in 2010 and 2015, respectively.

He spent two years in University of Illinois at Urbana-Champaign (UIUC), Champaign, IL, USA, as a Post-Doctoral Researcher and joined University of Notre Dame (ND), Notre Dame, IN, USA, in 2017, where he is currently an Assistant Professor with the Department of Computer Science and Engineering. He has authored or coauthored more than 100 peer-reviewed articles of these topics. His research interests include data mining, machine learning, and natural language processing.

Dr. Jiang was a recipient of the Notre Dame International Faculty Research Award. The honors and awards he received include Best Paper Finalist in KDD 2014, Best Paper Award in KDD-DLG workshop 2020, and ACM SIGSOFT Distinguished Paper Award in ICSE 2021.