Purchase Pattern Based Anti-Fraud Framework in Online E-Commerce Platform Using Graph Neural Network

Sanpeng Wang(✉), Yan Liu, Chu Zheng(✉), and Rui Lin(✉)

Retail Risk Management Group, JD, Beijing 100083, China
{wangsanpeng,liuyan961,zhengchu,linrui}@jd.com

Abstract. Click Farming is fraudulent behaviors sponsored by malicious merchants to increase exposure by hiring fraudulent teams to place fraudulent orders, posing a serious threat to the operation of platforms. Traditional anti-fraud strategies are no longer applicable as they analyzed fraudulent behaviors individually and only rely on static statistical characteristics. In this paper, we propose a novel graph-based fraud detection framework deployed on JD.com composed of *Dynamic Purchase Pattern learning* (DPP) and *Graph Neural Network with Similarities and Relations* (GSR). Specifically, the DPP module is a feature extractor based on user click location sequences collected from websites. And the GSR module is a neighborhood sampling and aggregation algorithm for locating more accurate fraud groups and aggregating various information encoded by different types of subgroups. We conduct graph node classification experiments on a large-scale real-world dataset to verify the effectiveness of our framework, and the experimental results show that the DPP is able to capture more discriminative user patterns. Furthermore, GSR achieves the best performance compared to several state-of-the-art methods. Our method can be easily extended to other domains with the same problems as our task.

Keywords: Fraud detection · User behavior · Graph neural networks

1 Introduction

For most online shopping platforms, sales and ratings are the two main factors that measure the quality of a product or a store [4]. Online shopping platforms are responsible for ensuring the authenticity of sales and ratings data. At the same time, there is an illegal industry called *Click Farming*. Click Farming is a fraudulent behavior that employs a group employees to place fraudulent orders and use fraudulent orders to increase the sales and ratings data. Therefore, fraudulent order detection has become an important issue for all online shopping platforms.

S. Wang and Y. Liu—Contribute equally to this work.

L. Fang et al. (Eds.): CICAI 2022, LNAI 13605, pp. 112–123, 2022.
https://doi.org/10.1007/978-3-031-20500-2_9

Currently, the are few studies on fraudulent order detection. First, since positive samples are scarce, most existing methods treat such tasks as anomaly detection problems [6,8,9,12,13,17]. Traditional methods rely on hand-designed features, which often have thousands of dimensions. They are hard to train and not robust enough. Another approach is use recurrent models for user behavior analysis, Wang et al. [19] took browsing item sequences as user behavior and achieved good performance with recurrent neural networks (RNN). The first disadvantage of this approach is that it only includes browsed products and ignores other representative behaviors, and another disadvantage is that millions of dynamically updated product embeddings are difficult to learn. The third approach are graph-based models [6,13,15]. Fraud gangs of fake orders have aggregation relationships in terms of IP address, devices ID etc. Therefore, some indistinguishable behaviors can be judged by the aggregating information. However, order networks typically behave as heterogeneous graphs in production environment, where orders are connected by various types of relationships. Therefore, there are two main challenges for further research.

Capture More Discriminative Features: Our approach innovatively adopts user behavior as features. We found that normal users and fraudulent users often have different behavioral features. Normal user always buys a suitable product by browsing product details or comparing similar products. While fraudulent users usually have another purchase pattern. Therefore, perhaps dynamic user behavior information is more effective.

Design More Effective Models: Device ID or IP are good edge-building features for building heterogeneous graph. But in heterogeneous graph, edges have different properties which can't be treated equally. Figure 1 depicts this situation.

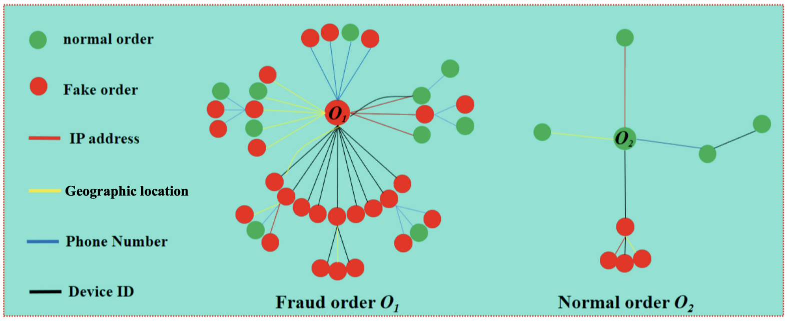

Fig. 1. Graph structure in orders network: O_1 is a fraudulent order but O_2 is a normal order, but O_1 has connections with normal orders by edge type of IP address, geographic location or phone number, O_2 has connection with fraud order by edge type of Device ID.

To address these issues, we propose a novel encoder-decoder fraud detection framework. The first part of the framework is a dynamic purchase pattern learning algorithm (DPP) which is responsible for capturing user behaviors based on click locations. We then model transactions as a heterogeneous graph and treat fraud detection as a graph node classification problem. The core of this heterogeneous graph is a sample and aggregation strategy based on neighborhood similarities and relationships. In summary, our contributions are fourfold:

- An encoder-decoder model is proposed to solve the fraudulent order detection problem. It consists of a recurrent model as an encoder and a heterogeneous graph model as a decoder (Fig. 2);
- We take the click location as user behavior and use a similarity-aware sampling strategy to avoid the negative effects of irrelevant neighbors;
- Our model outperforms other state-of-the-art models in our fraudulent order detection scenario;
- We deployed our framework on JD.com's fraudulent order detection and achieved accurate detection over 100 million online transactions per day.

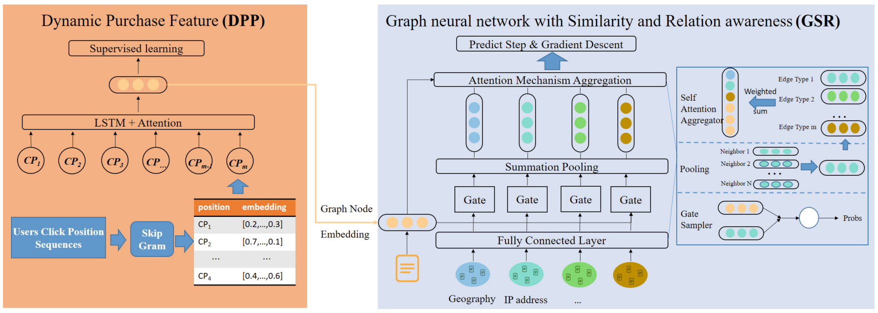

Fig. 2. Overall architecture of proposed framework.

2 Related Work

2.1 Fraud Detection

Various methods have been proposed for fraud detection, ranging from data mining algorithms [3] to deep models [5,7,8,20]. Bahnsen et al. [1] designed a feature generation method based on aggregated transactions. These hand-engineered features require domain expertise and can quickly become obsolete as fraud teams change their behavioral patterns. To bridge this gap, Wang et al. [19] elaborated user purchase behavior feature extraction by extending the item2vec algorithm [2]. However, this method only learns product embeddings but ignore the click behaviors.

2.2 Graph Representation Learning

Graph Neural Networks (GNNs) are a class of deep learning algorithms designed to perform inference on data described by graphs. Formally, GNNs follow a neighborhood aggregation and combination mechanism, the aggregator and the combinator are both trainable that optimized by a supervised, semi-supervised or unsupervised method [21]. The original Graph Convolution Networks (GCN) [11] is designed for semi-supervised learning in a transductive setting, and the algorithm requires the full graph during training. GraphSage [10] proposed a batched-training algorithm for GCN. It samples a tree rooted at each node by recursively expanding toe root node's neighbors by K steps with a fixed sample size. For each tree, it computes the root node's hidden representation by aggregating hidden representation from bottom to top hierarchically. Graph Attentional Models [18] learn to assign different edge weights at each layer based on node features and have achieved state-of-the-art results on several graph learning tasks. Pourhabibi et al. [14] made a comprehensive survey about anomaly detection in fraud detection applications based on graph, they suggested that it remains an open problem that handle graph data with nodes are not explicitly linked together.

3 Proposed Method

The fraudulent order detection problem is defined as using user behavior and the order information to determine whether the order is fraudulent. Specifically, let T denote the transaction set. It consists of user behavior B and order information S, denoted as $T = [B; S]$. A heterogeneous graph $\mathcal{G} = \langle \mathcal{V}, \mathcal{E} \rangle$ consists of node set $\mathcal{V}$ and edge set $\mathcal{E}$, where $\mathcal{E}$ is a subset of order information S.

Given any central node T_u with its neighbor nodes N_u, the neighbor nodes can be divided into z groups by their edge types, expressed as Eq. (1).

$$N_u = \bigcup_{l=1}^{z} N_u^l. \tag{1}$$

The goal of our method is to use the order information to predict the legitimacy of an order. It is modeled as a binary classification problem on graph nodes. As mentioned above, fraudulent order detection is an encoder-decoder model. The DPP module uses behavioral information to encode features F through a recurrent model. And the GSR uses the decoded features and a graph $\mathcal{G}$ to obtain predictions.

$$\begin{aligned} F &= \mathrm{DPP}(B) \\ \hat{y} &= \mathrm{GSR}(F, \mathcal{G}). \end{aligned} \tag{2}$$

3.1 Dynamic Purchase Pattern (DPP)

The browsed-products based approach [21] suffers from learning an embedding matrix from billions of products. Here we introduce a new purchase pattern called

Click Position (CP). CP is a set of clicked positions during the trading session. This model has several merits. First, click position types are less than 10,000, which are better for training. Second, CP are more meaningful and friendly to low-frequency products.

Different user behaviors play different roles in trading session, some behaviors are distinguishable and some are not. Figure (3) illustrates some CP instances and their relative occurrence in different types of orders. We take Eq. (3) to assign weights for all behaviors, and filter out all low rate behaviors.

$$B = \left\{ B_i \middle| \max\left(\frac{\mathrm{CP}_n \cdot \mathrm{All}_f}{\mathrm{CP}_f \cdot \mathrm{All}_n}, \frac{\mathrm{CP}_f \cdot \mathrm{All}_n}{\mathrm{CP}_n \cdot \mathrm{All}_f} \right) < \theta \right\}. \tag{3}$$

where CP_n is the number of click positions in normal orders, versa CP_f in fraud orders. All_n and All_f are the number of all click positions in normal and fraud order. And θ is the threshold.

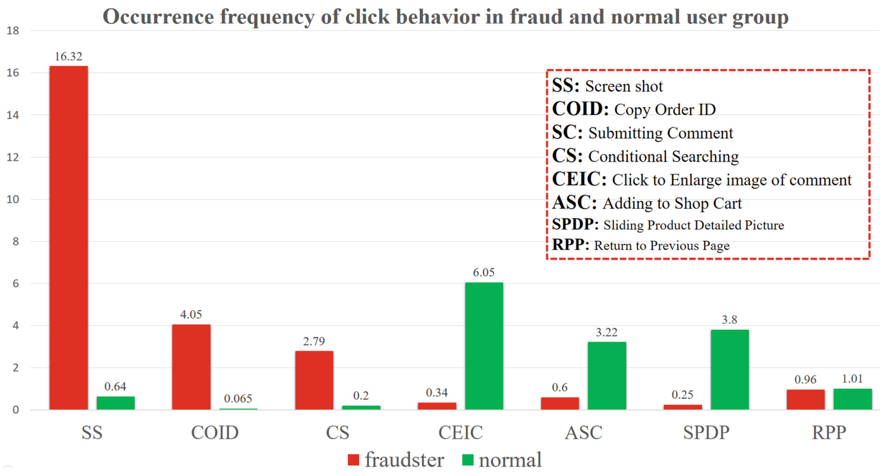

Fig. 3. Example of user click behavior difference.

Given a click positions sequence B. Our encoder consists of a combination of LSTM and self-attention, and is designed to encode the sequence of click positions. The encoding feature F is computed as Eq. (4).

$$\begin{aligned} H &= \mathrm{LSTM}(\mathrm{Embed}(B)) \\ F &= \text{Self-Attention}(H). \end{aligned} \tag{4}$$

At the end of the encoder, we employ a multi-layer perceptron (MLP) to obtain prediction results, aiming to learn robust encoding features.

$$\hat{y}^{\mathrm{DPP}} = \mathrm{MLP}(F). \tag{5}$$

DPP is trained by minimize the classification error of all click sequences, as show in Eq. (6).

$$\mathcal{L}_{\text{DPP}} = -\sum_{i=1}^{n} y_i^{\text{DPP}} \log \hat{y}_i^{\text{DPP}}. \tag{6}$$

3.2 GNN with Similarity and Relation (GSR)

We find that DPP cannot mine the internal dependencies between different orders. Therefore, we build a heterogeneous graph and use DPP as the feature extractor. The heterogeneous graph is built based on the relationship between different nodes. If a central node and its neighbor have the same label, we call them related neighbors, otherwise they are unrelated. Furthermore, we found that different edge types plays different roles in our task. Therefore, we proposed the following neighbors sampling and aggregation strategy.

Similarity Sampler: We introduce a gate-based structure to minimize the influence of unrelated neighbors. For a central node u and its' neighbor nodes v, a similarity measure function is used to measure the similarity between them. In our proposed method, cosine similarity is adopted:

$$\text{sim}(u, v) = \cos(H_u, H_v) = \frac{H_u \cdot H_v}{\|H_u\|\|H_v\|}, \tag{7}$$

where H is the sum of the central node embedding F, the embedding of in-degree nodes E_{in} and the embedding of out-degree nodes E_{out}, denoted as: $H = W \cdot F + W_{\text{in}} \cdot E_{\text{in}} + W_{\text{out}} \cdot E_{\text{out}}$. And W is the trainable weight matrix.

In our heterogeneous graph, different edge types have different properties. The difference property of edge types motivates us to use non-shareable weight matrix for them when calculating the similarity. Therefore, we apply z non-shareable and trainable weight matrices for z different edge types. Then we compute the similarity for all edge types, the neighbors' feature is updated by the product of similarities and features, as shown in Eq. (8).

$$\hat{F}_v = \text{sim}(u, v) \cdot F_v. \tag{8}$$

After analyzing our dataset, we found that the number of valid neighbors is usually less than 50. Therefore we sample the neighbors to a fix number of 50, and apply zero padding for central nodes of neighbors less than 50. For each subgroup of node u, adding the embeddings of all neighbors to get the embedding of this edge type, denoted as $\hat{F}_u^l$, as in the Eq. (9).

$$\hat{F}_u^l = \sum_{v \in N_u^l} \hat{F}_v. \tag{9}$$

Relation Aggregator: And we found the importance of edge types varies with the central node. Therefore, we employ a multi-head self-attention mechanism to learn a weighted for each type, as shown in Eq. (10).

$$\alpha_u^l = \frac{\exp\left(\text{LeakyReLU}(a^T[W_a \cdot F_u \| W_b \cdot \hat{F}_u^l])\right)}{\sum_{i=1}^{z} \exp\left(\text{LeakyReLU}(a^T[W_a \cdot F_u \| W_b \cdot \hat{F}_u^i])\right)}$$
$$\hat{F}_u = \frac{1}{z}\sum_{i=1}^{z} \alpha_u^l \hat{F}_u^l. \tag{10}$$

where W_a, W_b are linear matrices of the central node and neighbor sub-groups respectively, and $[\cdot\|\cdot]$ is the concatenation function. And a residual structure is used to compute the embedding of node u, denoted as Eq. (11):

$$E_u = F_u + \hat{F}_u. \tag{11}$$

Finally, we use MLP to get the prediction result of each central node u.

$$\hat{y}_u = \text{MLP}(E_u). \tag{12}$$

Model Training: For a graph with n nodes(orders), it is optimized by minimizing the classification loss of all nodes.

$$\mathcal{L}_{\text{class}} = -\sum_{u=1}^{n} y_u \log \hat{y}_u. \tag{13}$$

Furthermore, to guide the gate structure to find irrelevant nodes and speed up convergence, w penalize those sampled irrelevant neighbors with high similarity.

$$\mathcal{L}_{\text{sim}} = -\sum_{u=1}^{n}\sum_{l=1}^{z}\sum_{v\in N_u} \delta(y_u, \hat{y}_u) \cdot \text{sim}(u, v). \tag{14}$$

where δ is an *xor* function. The final optimization objective is balanced by a penalty coefficient λ, as shown in Eq. (15).

$$\mathcal{L}_{\text{FSO}} = \mathcal{L}_{\text{class}} + \lambda\mathcal{L}_{\text{sim}}. \tag{15}$$

4 Experiments

In this section, we first evaluate the performance of our method on a large real-world dataset. Then, we conduct a case study of user purchase patterns of this model. Finally, we compare our fraudulent order detection model with some other state-of-the-art classical graph algorithms.

4.1 Dataset

This fraudulent order detection dataset comes from JD.com, and all data has been desensitized to protect business secrets and user privacy. The dataset collects many details of orders, including order IDs, IP addresses and user click positions etc. To facilitate training and validating models, we group these labeled orders by date, and 3 days ($D_1 < D_2 < D_3$) of labeled data were selected to form the dataset. Table 1 lists the statistics for the daily datasets.

Table 1. Statistics of orders datasets.

Date	Normal	Frauludent	Total
D_1	39,916	6,376	46,292
D_2	41,878	7,403	49,281
D_3	43,135	6,669	49,804

4.2 Purchase Pattern Visualization

As mentioned earlier, group fraudsters employed by the same retailer tend to exhibit similar behavior patterns during the transaction session. On the other hand, normal users have their own personalized behavior patterns. We want to provide some intuition about which patterns of our model are captured. To do this, we randomly select some orders and generate their 100-dimensional vector via DPP. We then project them into a 2-D space using t-SNE. Figure 4 reveals some important insights: First, a clear boundary between normal and fraud can be easily observed. Compare to fraudsters, the behaviors of normal users are more fragmented. Finally, there are some samples fall into the opposite space, making identification difficult. Therefore, only using behavior features without analyzing sample relationships can lead to misclassification.

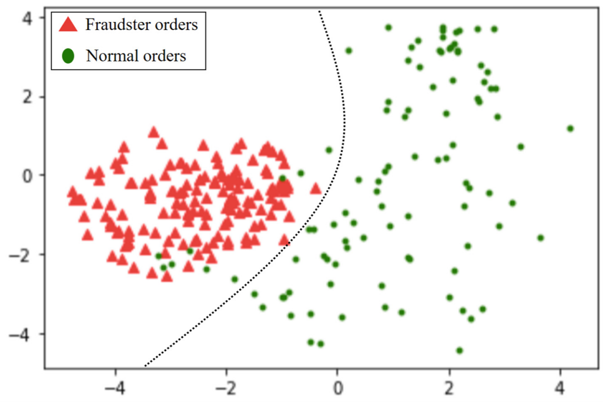

Fig. 4. Visualization of normal and fraudster group purchase behaviors using t-SNE.

Figure 5 gives an example of the classification performance of a simple linear layer with the behavior embedding as input. The x-axis represents the output risk probability. The y-axis represents the 1-dimension reduction result for each orders. As shown, it is difficult for the classifier to classify samples with probability between 0.2 and 0.8 due to ignoring the group context. Therefore, we review this part of data and use the graph model in Sect. 3.2 to identify fraudulent orders.

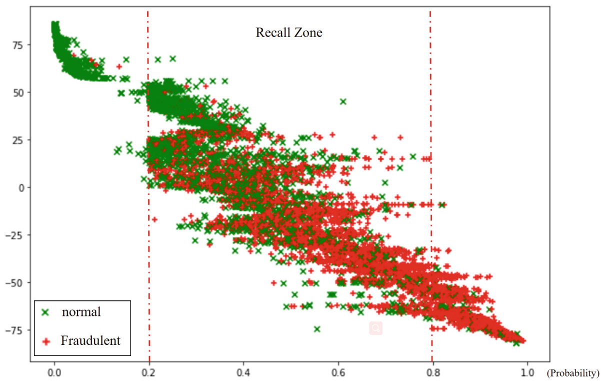

Fig. 5. Recall zone of toy classifier.

4.3 Fraudulent Order Detection Result of GSR

Orders Graph Construction: All the samples in recall zone will be used to build the graph to facilitate our graph-based model. Equivalence relationships in IP addresses, product IDs, phone numbers, device IDs and geography locations are used to link edge between orders. Therefore, there is a totally one type of node and five types of edges in the graph. Table 2 describes statistical graph data information in the three datasets.

Table 2. Statics information of three graph datasets.

Date	Nodes	IP Address	Product ID	Phone	Device ID	Geography	Edges
D_1	20,268	9,300	268,055	2,386,941	8,345	6,299	2,678,940
D_2	20,537	7,688	249,231	2,713,310	6,509	6,612	2,983,350
D_3	23,128	7,284	349,249	3,303,950	6,568	6,741	3,673,792

Comparison Baselines: To demonstrate the effectiveness of group fraud interconnection, we use several basic classifiers: Random Forest (RF), Support Vector Machine (SVM), Logistic Regression (LR), Deep Neural Network (DNN) as baselines, which ignore orders' internal dependencies, and name these models as isolated-models. To verify the superiority of neighbor similarity and the difference in relation types, we compare our model with three supervised graph neural networks, namely GCN [11], GraphSAGE [10], GAT [18], SCR [22] and SAGN [16]. To demonstrate the importance of similarity and relation, two additional baselines, GR and GS, are designed, which represent models without similarity and relation respectively. For comparison, all neighbors sampling parameters are set to match those used in our model. All models were trained from scratch until convergence. The initial node features are all the same dense user behavior vector. The dimension of node embedding is uniformly set to 100.

Metrics and Performance Evaluation: Precisely identifying fraud cases is our main focus, therefore, the performances of different models is evaluated using the F1 score. To comprehensively evaluate the effect of binary classification, the area of precision-recall curve (AUC) is also provided. All networks are trained on dataset D1 and D2 respectively, and dataset D3 is used as the test set. The average of the test set evaluation results of the models trained on the two training sets is taken as the final evaluation result.

Table 3 lists the experimental results. As can be seen from the table, all isolated-models underperform graph-based models. This result suggests that neighbors are helpful in identifying ambiguous cases. And we find that the three traditional GNN methods fail to consider filtering irrelevant neighbors when aggregating features, and in addition, ignore different relation types. Finally, our model achieves about 2% improvement compare to the best baseline model. We further validate the effect of node on similarity judgement and type attention mechanism through ablation study. The comparison results show that by dynamically controlling the similarity, we can flexible mask those unrelated neighbors. Furthermore, considering the type importance weights also enhances the fraud detection capability. Therefore, we can safely conclude that the proposed method is a robust and effective fraudulent order detection system.

Table 3. Comparison results.

Type	Methods	F1	AUC
isolated	RF	68.24	72.82
	LR	70.28	73.72
	SVM	70.39	–
	DNN	69.47	72.90
Graph-based	GCN	79.22	95.72
	GraphSAGE	83.49	96.07
	GAT	79.63	95.81
	SCR	80.61	95.96
	SAGN	82.86	96.03
Proposed	GS	84.43	95.21
	GR	83.21	96.01
	GSR	**85.27**	**96.26**

5 Conclusions and Future Work

In this paper, we exploit the sequence of click position to solve the task of fraudulent order detection. We propose a two stage encoder-decoder framework for this task. First, we model the click sequence through RNN and self-attention model to generate static features. Subsequently, a GNN model based on large-scale graph is used to identify the association between transactions through the similar neighbor sampling module and the edge-type based attention module. Experiments show that our model is credible on large scale datasets and greatly improves the recall of fraudulent order detection. For future work, since the construction of real-time trade charts is a challenge for us, we will give more consideration to combine real-time charts with our methods.

References

1. Bahnsen, A.C., Aouada, D., Stojanovic, A., Ottersten, B.: Feature engineering strategies for credit card fraud detection. Expert Syst. Appl. **51**, 134–142 (2016)
2. Barkan, O., Koenigstein, N.: ITEM2VEC: neural item embedding for collaborative filtering. In: 2016 IEEE 26th International Workshop on Machine Learning for Signal Processing (MLSP), pp. 1–6. IEEE (2016)
3. Bhattacharyya, S., Jha, S., Tharakunnel, K., Westland, J.C.: Data mining for credit card fraud: a comparative study. Decis. Support Syst. **50**(3), 602–613 (2011)
4. Bickart, B., Schindler, R.M.: Internet forums as influential sources of consumer information. J. Interact. Mark. **15**(3), 31–40 (2001)
5. Chandradeva, L.S., Amarasinghe, T.M., De Silva, M., Aponso, A.C., Krishnarajah, N.: Monetary transaction fraud detection system based on machine learning strategies. In: Yang, X.-S., Sherratt, S., Dey, N., Joshi, A. (eds.) Fourth International Congress on Information and Communication Technology. AISC, vol. 1041, pp. 385–396. Springer, Singapore (2020). https://doi.org/10.1007/978-981-15-0637-6_33
6. Eberle, W., Holder, L.: Discovering structural anomalies in graph-based data. In: Seventh IEEE International Conference on Data Mining Workshops (ICDMW 2007), pp. 393–398. IEEE (2007)
7. Fei, G., Mukherjee, A., Liu, B., Hsu, M., Castellanos, M., Ghosh, R.: Exploiting burstiness in reviews for review spammer detection. In: Seventh International AAAI Conference on Weblogs and Social Media (2013)
8. Georgieva, S., Markova, M., Pavlov, V.: Using neural network for credit card fraud detection. In: AIP Conference Proceedings, vol. 2159, p. 030013. AIP Publishing LLC (2019)
9. Glover, S., Benbasat, I.: A comprehensive model of perceived risk of e-commerce transactions. Int. J. Electron. Commer. **15**(2), 47–78 (2010)
10. Hamilton, W., Ying, Z., Leskovec, J.: Inductive representation learning on large graphs. In: Advances in Neural Information Processing Systems, pp. 1024–1034 (2017)
11. Kipf, T.N., Welling, M.: Semi-supervised classification with graph convolutional networks. In: International Coference on Learning Representations (ICLR) (2017)
12. Lim, E.P., Nguyen, V.A., Jindal, N., Liu, B., Lauw, H.W.: Detecting product review spammers using rating behaviors. In: Proceedings of the 19th ACM International Conference on Information And Knowledge Management, pp. 939–948 (2010)
13. Manzoor, E., Milajerdi, S.M., Akoglu, L.: Fast memory-efficient anomaly detection in streaming heterogeneous graphs. In: Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 1035–1044 (2016)
14. Pourhabibi, T., Ong, K.L., Kam, B.H., Boo, Y.L.: Fraud detection: a systematic literature review of graph-based anomaly detection approaches. Decis. Support Syst. **133**, 113303 (2020)

15. Shehnepoor, S., Salehi, M., Farahbakhsh, R., Crespi, N.: Netspam: a network-based spam detection framework for reviews in online social media. IEEE Trans. Inf. Forensics Secur. **12**(7), 1585–1595 (2017)
16. Sun, C., Gu, H., Hu, J.: Scalable and adaptive graph neural networks with self-label-enhanced training. arXiv preprint arXiv:2104.09376 (2021)
17. Tao, J., Wang, H., Xiong, T.: Selective graph attention networks for account takeover detection. In: 2018 IEEE International Conference on Data Mining Workshops (ICDMW), pp. 49–54. IEEE (2018)
18. Veličković, P., Cucurull, G., Casanova, A., Romero, A., Lio, P., Bengio, Y.: Graph attention networks. In: International Conference on Learning Representations (ICLR) (2018)
19. Wang, S., Liu, C., Gao, X., Qu, H., Xu, W.: Session-based fraud detection in online e-commerce transactions using recurrent neural networks. In: Altun, Y., et al. (eds.) ECML PKDD 2017. LNCS (LNAI), vol. 10536, pp. 241–252. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-71273-4_20
20. Wu, X., Dong, Y., Tao, J., Huang, C., Chawla, N.V.: Reliable fake review detection via modeling temporal and behavioral patterns. In: 2017 IEEE International Conference on Big Data (Big Data), pp. 494–499. IEEE (2017)
21. Xu, K., Hu, W., Leskovec, J., Jegelka, S.: How powerful are graph neural networks? In: International Conference on Learning Representations (ICLR) (2019)
22. Zhang, C., He, Y., Cen, Y., Hou, Z., Tang, J.: Improving the training of graph neural networks with consistency regularization. arXiv preprint arXiv:2112.04319 (2021)