Alleviating the Inconsistency Problem of Applying Graph Neural Network to Fraud Detection

Zhiwei Liu, Yingtong Dou, Philip S. Yu
Department of Computer Science, University of Illinois at Chicago
{zliu213,ydou5,psyu}@uic.edu

Yutong Deng
School of Software, Beijing University of Posts and Telecommunications
buptdyt@bupt.edu.cn

Hao Peng
Beijing Advanced Innovation Center for Big Data and Brain Computing, Beihang University
penghao@act.buaa.edu.cn

ABSTRACT

Graph-based models have been widely used to fraud detection tasks. Owing to the development of Graph Neural Networks (GNNs), recent works have proposed many GNN-based fraud detectors based on either homogeneous or heterogeneous graphs. These works leverage existing GNNs and aggregate the neighborhood information to learn the node embeddings, which relies on the assumption that the neighbors share similar context, features, and relations. However, the inconsistency problem incurred by fraudsters is hardly investigated, i.e., the context inconsistency, feature inconsistency, and relation inconsistency. In this paper, we introduce these inconsistencies and design a new GNN framework, GraphConsis, to tackle the inconsistency problem: (1) for the context inconsistency, we propose to combine the context embeddings with node features; (2) for the feature inconsistency, we design a consistency score to filter the inconsistent neighbors and generate corresponding sampling probability; (3) for the relation inconsistency, we learn the relation attention weights associated with the sampled nodes. Empirical analysis on four datasets demonstrates that the inconsistency problem is critical in fraud detection tasks. Extensive experiments show the effectiveness of GraphConsis. We also released a GNN-based fraud detection toolbox with implementations of SOTA models. The code is available at https://github.com/safe-graph/DGFraud.

CCS CONCEPTS

• **Security and privacy** → **Web application security**; • **Computing methodologies** → **Neural networks**.

KEYWORDS

Graph Neural Networks; Fraud Detection; Inconsistency Problem

ACM Reference Format:
Zhiwei Liu, Yingtong Dou, Philip S. Yu, Yutong Deng, and Hao Peng. 2020. Alleviating the Inconsistency Problem of Applying Graph Neural Network to Fraud Detection. In *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR '20), July 25–30, 2020, Virtual Event, China.* ACM, New York, NY, USA, 4 pages. https://doi.org/10.1145/3397271.3401253

SIGIR '20, July 25–30, 2020, Virtual Event, China

ACM ISBN 978-1-4503-8016-4/20/07…$15.00
https://doi.org/10.1145/3397271.3401253

1 INTRODUCTION

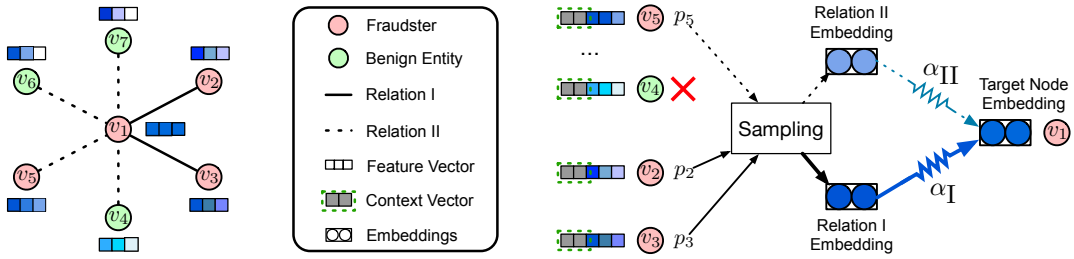
There are various kinds of fraudulent activities on the Internet [4], e.g., fraudsters disguise as regular users to post fake reviews [5] and commit download fraud [1]. By modeling the entities as nodes and the corresponding interactions between entities as edges [10], we can design a graph-based algorithm to detect the suspicious patterns and therefore can spot the fraudsters. Along with the development of Graph Neural Networks (GNNs) [2, 7, 13], which are powerful in learning the deep representation of nodes, recently, the previous endeavors also propose many GNN-based fraud detection frameworks [8, 9, 14–17].

Among those frameworks, [8, 15] detect opinion fraud in the online review system, [9, 14, 17] aim at financial fraud and [16] targets cyber-criminal in online forums. They model their problems upon either homogeneous [8, 15] or heterogeneous [8, 9, 14, 16, 17] graphs. Regarding the base model, FdGars [15] and GAS[8] adopt GCN [7], while SemiGNN and Player2Vec [16] adopt GAT [13]. Some works [8, 9, 17] devise new aggregators to aggregate the neighborhood information. Those GNN-based fraud detectors learn the node representation iteratively and predict the node suspiciousness in an end-to-end and semi-supervised fashion.

However, all existing methods ignore the inconsistency problem when designing a GNN model regarding the fraud detection task. The inconsistency problem is associated with the aggregation process of the GNN model. The mechanism of aggregation is based on the assumption that the neighbors share similar features and labels [3]. When the assumption holds, we can aggregate the neighborhood information to learn the node embedding. However, as Figure 1 (*Left*) shows, the inconsistency in fraud detection problem comes from three perspectives:

(1) **Context inconsistency**. Smart fraudsters can connect themselves to regular entities as camouflage [6, 12]. Meanwhile, the amount of fraudsters is much less than that of regular entities. Directly aggregating neighbors by the GNNs can only help the fraudulent entities aggregate the information from regular entities and thus prevent themselves from being spotted by fraud detectors. For example, in Figure 1 (*Left*), the fraudster v_1 connects to 3 benign entities under relation II.

(2) **Feature inconsistency**. Taking the opinion fraud (e.g., spam reviews) detection problem as an example [8], assuming there are two reviews posted from the same user but to products in distinct categories, those two reviews have an edge since they share the same user. However, their review content (features) are far from each other as they are associated with different products. Direct aggregation makes the GNN hard to distinguish the unique semantic characteristics of reviews and finally affects its ability to detect



Left: Inconsistency Problem ***Right***: Proposed **GraphConsis** Model

Figure 1: *Left*: A toy example of a graph with two relations constructed on a fraud dataset, $v_2 - v_7$ are neighbors of v_1. Context inconsistency: fraudster v_1 can connect to many benign neighbors (v_4, v_6, v_7 in Relation II) to disguise itself. Feature inconsistency: for v_2 and v_3 with the same relation to $v1$, their features may have great differences. Relation inconsistency: for v_1, Relation I connects more similar neighbors than Relation II. *Right*: To alleviate the inconsistency problem, we introduce three techniques. First, we propose to combine the context embeddings with feature vectors. Then, we calculate the consistency scores of neighbors to filter nodes and generate sampling probabilities. Finally, we aggregate the sampled neighbors with the attention mechanism over relation embeddings.

spam reviews. For example, in Figure 1 (*Left*), we can find that the feature of node v_1 is inconsistent with nodes v_4, v_6, and v_7.
(3) **Relation inconsistency**. Since the entities are connected with multiple types of relations, simply treating all the relations equally results in a relation inconsistency problem. For example, two reviews may either be connected by the same user or the same product, which are respectively *common-user* relation and *common-product* relation. Assuming that one review is suspicious, then the other one should have a greater suspiciousness if they are connected by *common-user* relation since fraudulent users tend to post many fraudulent reviews. For example, in Figure 1 (*Left*), we find that under relation II, the fraudster v_1 is connected to two other fraudsters. However, under relation I, the fraudster is connected to only one fraudster but three benign entities.

To tackle all above inconsistencies, we design a novel GNN framework, **GraphConsis**, to solve the fraud detection problem, as shown in Figure 1 (*Right*). GraphConsis is built upon a heterogeneous graph. GraphConsis differs existing GNNs from the aggregation process. Instead of directly aggregating neighboring embeddings, we design three techniques to tackle three inconsistency problems simultaneously. Firstly, to handle the context inconsistency of neighbors, GraphConsis assigns each node a trainable context embedding, which is illustrated as the gray block aside nodes in Figure 1 (*Right*). Secondly, to aggregate consistent neighbor embeddings, we design a new metric to measure the *embedding consistency* between nodes. By incorporating the embedding consistency score into the aggregation process, we ignore the neighbors with a low consistency score (e.g. the node v_4 is dropped in Figure 1 (*Right*)) and generate the sampling probability. Last but not least, we learn relation attention weights associated with neighbors in order to alleviate the relation inconsistency problem.

The contributions of this paper are:

- To the best of our knowledge, we are the first work addressing the inconsistency problem in GNN models.
- We empirically analyze three inconsistency problems regarding applying GNN models to fraud detection tasks.
- We propose GraphConsis to tackle three inconsistency problems, which combines context embedding, neighborhood information measure, and relational attention.

2 PRELIMINARIES

We detect the fraud entities in the graph by using the node representations. Hence, we need to introduce the node representation learning first. A heterogeneous graph $G = \{V, \mathbf{X}, \{E_r\}|_{r=1}^{R}\}$, where V denotes the nodes, $\mathbf{X}$ is the feature matrix of nodes, and E_r denotes the edges w.r.t. the relation r. We have R different types of relations. To represent the nodes as vectors, we need to learn a function $f : V \rightarrow \mathbb{R}^d$ that maps nodes to a d dimensional space, where $d \ll |V|$. The function f should preserve both the structural information of the graph and the original feature information of the nodes. With the learned node embeddings, we can train a classifier $C : \mathbb{R}^d \rightarrow \{0, 1\}$ to detect whether a given node is a fraudster, where 1 denotes fraudster, and 0 denotes benign entity. In this paper, we adopt the GNN framework to learn the node representation through neighbor aggregation. GNN framework can train the mapping function f and the classifier C simultaneously. We only need to input the graph and the labels of nodes to a GNN model. The general framework of a GNN model is:

$$\mathbf{h}_v^{(l)} = \mathbf{h}_v^{(l-1)} \oplus \text{AGG}^{(l)}\left(\left\{\mathbf{h}_{v'}^{(l-1)} : v' \in \mathcal{N}_v\right\}\right), \tag{1}$$

where $\mathbf{h}_v^{(l)}$ is the hidden embedding of v at l-th layer, $\mathcal{N}_v$ denotes the neighbors of node v, and the AGG represents the aggregation function that maps the neighborhood information into a vector. Here, we use $\oplus$ to denote the combination of neighbor information and the center node information, it can be direct addition or concatenation then passed to a neural network. For the AGG function, we first assign a sampling probability to the neighboring nodes. Then we sample Q nodes and average[1] them as a vector. The calculation of probability is introduced later in Eq. (4). Note that the framework of GNN is a L-layer structure, where $1 \leq l \leq L$. At l-th layer, it aggregates the information from $l-1$-th layer.

3 PROPOSED MODEL

3.1 Context Embedding

The aggregator combines the information of neighboring nodes according to Eq. (2). When $k = 1$, the hidden embedding $\mathbf{h}_v^{(0)}$ is

[1]Other pooling techniques can also be applied.

equivalent to the node feature. To tackle the context inconsistency problem, we introduce a trainable context embedding $\mathbf{c}_v$ for node v., instead of only using its feature vector $\mathbf{x}_v$. The first layer of the aggregator then becomes:

$$\mathbf{h}_v^{(1)} = \{\mathbf{x}_v \| \mathbf{c}_v\} \oplus \text{AGG}^{(1)}\left(\{\mathbf{x}_{v'} \| \mathbf{c}_{v'} : v' \in \mathcal{N}_v\}\right), \tag{2}$$

where $\|$ denotes the concatenation operation. The context embedding is trained to represent the local structure of the node, which can help to distinguish the fraud. If we use addition operation for $\oplus$, then $\mathbf{h}_v \in \mathbb{R}^{2d}$.

3.2 Neighbor Sampling

Since there exists a feature inconsistency problem, we should sample related neighbors rather than assign equal probabilities to them. Thus, we compute the **consistency score** between embeddings:

$$s^{(l)}(u, v) = \exp\left(-\|\mathbf{h}_u^{(l)} - \mathbf{h}_v^{(l)}\|_2^2\right), \tag{3}$$

where $s^{(l)}(\cdot, \cdot)$ denotes the consistency score for two nodes at l-th layer, and $\|\cdot\|_2$ is the l_2-norm[2] of vector. We first apply a threshold ϵ to filter neighbors far away from consistent. Then, we assign each node u to the filtered neighbors $\tilde{\mathcal{N}}_v$ of node v with a sampling probability by normalizing its consistency score:

$$p^{(l)}(u; v) = s^{(l)}(u, v) / \sum_{u \in \tilde{\mathcal{N}}_v} s^{(l)}(u, v). \tag{4}$$

Note that the probability is calculated at each layer for the $\text{AGG}^{(l)}$.

3.3 Relation Attention

We have R different relations in the graph. The relation information should also be included in the aggregation process to tackle the relation inconsistency problem. Hence, for each relation r, we train a relation vector $\mathbf{t}_r$, where $r = \{1, 2, \ldots, R\}$, to represent the relation information that should be incorporated. Since the relation information should be aggregated along with the neighbors to center node v, we adopt the self-attention mechanism [13] to assign weights for Q sampled neighbor nodes:

$$\alpha_q^{(l)} = \exp\left(\sigma\left(\{\mathbf{h}_q^{(l)} \| \mathbf{t}_{r_q}\}\mathbf{a}^\top\right)\right) / \sum_{q=1}^{Q} \exp\left(\sigma\left(\{\mathbf{h}_q^{(l)} \| \mathbf{t}_{r_q}\}\mathbf{a}^\top\right)\right), \tag{5}$$

where r_q denotes the relation of q-th sample with node v, σ is the activation function, and $a \in \mathbb{R}^{4d}$ represents the attention weights that is shared for all attention layer. The final $\text{AGG}^{(l)}$ is:

$$\text{AGG}^{(l)}\left(\left\{\mathbf{h}_q^{(l-1)}\right\}\Big|_{q=1}^{Q}\right) = \sum_{q=1}^{Q} \alpha_q^{(l)} \mathbf{h}_q^{(l)}, \tag{6}$$

where $\mathbf{h}_q^{(l)}$ is the embedding of q-th node sampled based on Eq. (4).

4 EXPERIMENTS

4.1 Experimental Setup

4.1.1 Dataset and Graph Construction. We utilize the YelpChi spam review dataset [11], along with three other benchmark datasets [2, 7] to study the graph inconsistency problem in the fraud detection task. The YelpChi spam review dataset includes hotel and restaurant reviews filtered (spam) and recommended (legitimate) by Yelp. In this paper, we conduct a spam review classification task on the YelpChi dataset which is a binary classification problem. We remove products with more than 800 reviews to restrict the size of the computation graph. The pre-processed dataset has 29431 users, 182 products, and 45954 reviews (%14.5 spams).

Based on previous studies [11] which show the spam reviews have connections in user, product, rating, and time, we take reviews as nodes in the graph and design three relations denoted by *R-U-R*, *R-S-R*, and *R-T-R*. *R-U-R* connects reviews posted by the same user; *R-S-R* connects reviews under the same product with the same rating; *R-T-R* connects two reviews under the same product posted in the same month. We take the 100-dimension Word2Vec embedding of each review as its feature like previous work [8].

4.1.2 Baselines. To show the ability of GraphConsis in alleviating inconsistency problems, we compare its performance with a non-GNN classifier, vanilla GNNs, and GNN-based fraud detectors.

- **Logistic Regression**. A non-GNN classifier that makes predictions only based on the reviews features.
- **FdGars (GCN)** [15]. A spam review detection algorithm using GCN [7].
- **GraphSAGE** [2]. A popular GNN framework which samples neighboring nodes before aggregation.
- **Player2Vec** [16]. A state-of-the-art fraud detection model which uses GCN to encode information in each relation, and uses GAT to aggregate neighbors from different relations.

4.1.3 Experimental Settings. We use Adam optimizer to train our model based on the cross-entropy loss. For the hyper-parameters, we choose 2-layer structure, and the number of samples is set as 10 and 5 for the first layer and second layer, respectively. The embedding dimension of the hidden layer is 200 and 100 for the first layer and second layer, respectively. We use F1-score to measure the overall classification performance and AUC to measure the performance of identifying spam reviews.

4.2 The Inconsistency Problem

We first take the Yelpchi dataset to demonstrate the inconsistency problem in applying GNN to fraud detection tasks. Table 1 shows the statistics of graphs built on YelpChi comparing to node classification benchmark datasets used by [2, 7]. *Yelp-ALL* is composed of three single-relation graphs.

Comparing to three widely-used benchmark node classification datasets, we find that a multi-relation graph constructed on YelpChi has a much higher density (the average node degree is greater than 100). It demonstrates that the real-world fraud graphs usually incorporate complex relations and neighbors, and thus render inconsistency problems. Before we compare the graph characteristics and analyze three inconsistency problems, similar to [3], we design two characteristic scores. One is the context characteristic score:

$$\gamma_r^{(c)} = \sum_{(u,v) \in E_r} \left(1 - \mathbb{I}\left(u \sim v\right)\right) / |E_r|, \tag{7}$$

where $\mathbb{I}(\cdot) \in \{0, 1\}$ is an indicator function to indicate whether node u and node v have the same label. We sum all the indication w.r.t. all the edges and normalized by the total number of edges $|E_r|$. The context characteristic measures the label similarity between

[2]Other metrics, such as l_1-norm, are also applicable.

neighboring nodes under a specific relation r. The other one is the feature characteristic score:

$$\gamma_r^{(f)} = \sum_{(u,v)\in E_r} \exp\left(-\|\mathbf{x}_u - \mathbf{x}_v\|_2^2\right) / |E_r| \cdot d, \tag{8}$$

where we employ the RBF kernel function[3] as the similarity measurement between two connected nodes. The overall feature characteristic score is normalized by the product of the total number of edges $|E_r|$ and the feature dimension d. Normalizing the similarity by feature dimension is to fairly compare the feature characteristics of different graphs, which may have different feature dimensions.

Context Inconsistency. We compute the context characteristic $\gamma_r^{(c)}$ based on Eq. (7), which measures the context consistency. For the graph *R-T-R*, *R-S-R* and *Yelp-ALL*, there are less than 10% of neighboring nodes have similar labels. It shows that fraudsters may hide themselves among regular entities under some relations.

Feature Inconsistency. We calculate the feature characteristic $\gamma_r^{(f)}$ using Eq. (8). The graph constructed by *R-U-R* relation (reviews posted by the same user) has higher feature characteristic than the other two relations. Thus, we need to sample the neighboring nodes not only based on their relations but also the feature similarities.

Relation Inconsistency. For graphs constructed by three different relations, the neighboring nodes also have different feature/label inconsistency score. Thus, we need to treat different relations with different attention weights during the aggregation.

Table 1: The statistics of different graphs.

	Graph	#Nodes	#Edges	$\gamma^{(f)}$	$\gamma^{(c)}$
Others	**Cora**	2,708	5,278	0.72	0.81
	PPI	14,755	225,270	0.48	0.98
	Reddit	232,965	11,606,919	0.70	0.63
Ours	***R-U-R***	45,954	98,630	0.83	0.90
	R-T-R	45,954	1,147,232	0.79	0.05
	R-S-R	45,954	6,805,486	0.77	0.05
	Yelp-ALL	45,954	7,693,958	0.77	0.07

4.3 Performance Evaluation

Table 2 shows the experiment results of the spam review detection task. We could see that GraphConsis outperforms other models under 80% and 60% of training data on both metrics, which suggests that we can alleviate the inconsistency problem. Compared with other GNN-based models, LR performs stably and better on AUC. It indicates that the node feature is useful, but the aggregator in GNN undermines the classifier in identifying fraudsters. This observation also proves that the inconsistency problem is critical and should be considered when applying GNNs to fraud detection tasks. Compared to Player2Vec which also learns relation attention, GraphConsis performs better. It suggests that solely using relation attention cannot alleviate the feature inconsistency. The neighbors should be filtered and then sampled based on our designed methods. FdGars directly aggregates neighbors' information and GraphSAGE samples neighbors with equal probability. Both of them perform worse than GraphConsis, which shows that our neighbor sampling techniques are useful.

[3]Other kernel functions can also be applied.

Table 2: Experiment results under different training %.

Method	40%		60%		80%	
	F1	AUC	F1	AUC	F1	AUC
LR	0.4647	**0.6140**	0.4640	0.6239	0.4644	0.6746
GraphSAGE	0.4956	0.5081	0.5127	0.5165	0.5158	0.5169
FdGars	0.4603	0.5505	0.4600	0.5468	0.4603	0.5470
Player2Vec	0.4608	0.5426	0.4608	0.5697	0.4608	0.5403
GraphConsis	**0.5656**	0.5911	**0.5888**	**0.6613**	**0.5776**	**0.7428**

5 CONCLUSION AND FUTURE WORKS

In this paper, we investigate three inconsistency problems in applying GNNs in fraud detection problem. To address those problems, we design three modules respectively and propose GraphConsis. Experiment results show the effectiveness of GraphConsis. Future work includes devising an adaptive sampling threshold for each relation to maximize the receptive field of GNNs. Investigating the inconsistency problems under other fraud datasets is another avenue of future research.

ACKNOWLEDGMENTS

This work is supported by the National Key R&D Program of China under grant 2018YFC0830804, and in part by NSF under grants III-1526499, III-1763325, III-1909323, and CNS-1930941. For any correspondence, please refer to Hao Peng.

REFERENCES

[1] Y. Dou, W. Li, Z. Liu, Z. Dong, J. Luo, and P. S. Yu. 2019. Uncovering download fraud activities in mobile app markets. In *ASONAM*.
[2] W. Hamilton, Z. Ying, and J. Leskovec. 2017. Inductive representation learning on large graphs. In *NeurIPS*.
[3] Y. Hou, J. Zhang, J. Cheng, K. Ma, R. T. B. Ma, H. Chen, and M. Yang. 2020. Measuring and Improving the Use of Graph Information in Graph Neural Networks. In *ICLR*.
[4] M. Jiang, P. Cui, and C. Faloutsos. 2016. Suspicious behavior detection: Current trends and future directions. *IEEE Intelligent Systems* (2016).
[5] P. Kaghazgaran, M. Alfifi, and J. Caverlee. 2019. Wide-Ranging Review Manipulation Attacks: Model, Empirical Study, and Countermeasures. In *CIKM*.
[6] P. Kaghazgaran, J. Caverlee, and A. Squicciarini. 2018. Combating crowdsourced review manipulators: A neighborhood-based approach. In *WSDM*.
[7] T.N. Kipf and M. Welling. 2017. Semi-supervised classification with graph convolutional networks. In *ICLR*.
[8] A. Li, Z. Qin, R. Liu, Y. Yang, and D. Li. 2019. Spam Review Detection with Graph Convolutional Networks. In *CIKM*.
[9] Z. Liu, C. Chen, X. Yang, J. Zhou, X. Li, and L. Song. 2018. Heterogeneous Graph Neural Networks for Malicious Account Detection. In *CIKM*.
[10] H. Peng, J. Li, Q. Gong, Y. Song, Y. Ning, K. Lai, and P. S. Yu. 2019. Fine-grained Event Categorization with Heterogeneous Graph Convolutional Networks. In *IJCAI*.
[11] S. Rayana and L. Akoglu. 2015. Collective Opinion Spam Detection: Bridging Review Networks and Metadata. In *KDD*.
[12] L. Sun, Y. Dou, C. Yang, J. Wang, P. S. Yu, and B. Li. 2018. Adversarial Attack and Defense on Graph Data: A Survey. *arXiv preprint arXiv:1812.10528* (2018).
[13] P. Veličković, G. Cucurull, A. Casanova, A. Romero, P. Lio, and Y. Bengio. 2017. Graph attention networks. In *ICLR*.
[14] D. Wang, J. Lin, P. Cui, Q. Jia, Z. Wang, Y. Fang, Q. Yu, J. Zhou, S. Yang, and Y. Qi. 2019. A Semi-supervised Graph Attentive Network for Fraud Detection. In *ICDM*.
[15] J. Wang, R. Wen, C. Wu, Y. Huang, and J. Xion. 2019. FdGars: Fraudster Detection via Graph Convolutional Networks in Online App Review System. In *WWW Workshops*.
[16] Y. Zhang, Y. Fan, Y. Ye, L. Zhao, and C. Shi. 2019. Key Player Identification in Underground Forums over Attributed Heterogeneous Information Network Embedding Framework. In *CIKM*.
[17] Q. Zhong, Y. Liu, X. Ao, B. Hu, J. Feng, J. Tang, and Q. He. 2020. Financial Defaulter Detection on Online Credit Payment via Multi-View Attributed Heterogeneous Information Network. In *WWW*.