eFraudCom: An E-commerce Fraud Detection System via Competitive Graph Neural Networks

Article *in* ACM Transactions on Information Systems · July 2021

DOI: 10.1145/3474379

6 authors, including:

Jia Wu
Macquarie University
173 PUBLICATIONS **2,986** CITATIONS
SEE PROFILE

Chuan Zhou
Chinese Academy of Sciences
122 PUBLICATIONS **1,917** CITATIONS
SEE PROFILE

Jian Yang
Macquarie University
240 PUBLICATIONS **3,994** CITATIONS
SEE PROFILE

eFraudCom: An E-commerce Fraud Detection System via Competitive Graph Neural Networks

GE ZHANG, Macquarie University, Australia
ZHAO LI*, Alibaba Group, China
JIAMING HUANG, Alibaba Group, China
JIA WU, Macquarie University, Australia
CHUAN ZHOU, Academy of Mathematics and Systems Science, Chinese Academy of Sciences, China
JIAN YANG, Macquarie University, Australia

With the development of e-commerce, fraud behaviors have been becoming one of the major threats faced by online e-commerce platforms. Fraud behaviors seriously damage the ranking system of e-commerce platforms and adversely influence the shopping experience of users. Therefore, it is of great practical significance to study how to quickly and effectively identify those fraud behaviors, which is non-trivial since the adversarial actions taken by fraudsters and the upgrade of fraud patterns. Existing fraud detection systems used in the e-commerce industry easily suffer from performance decay and can not adapt to the upgrade of fraud patterns as they take already known fraud behaviors as supervision information to detect other suspicious behaviors.

In this paper, we propose a competitive graph neural networks (CGNN) based fraud detection system (eFraudCom) to detect fraud behaviors at one of the largest e-commerce platforms, "Taobao"[1], in China. In the eFraudCom system, (1) the competitive graph neural networks (CGNN) as the core part of eFraudCom can classify behaviors of users directly by modeling the distributions of normal and fraud behaviors separately; (2) some normal behaviors will be utilized as weak supervision information to guide the CGNN to build the profile for normal behaviors which are more stable than fraud behaviors. The algorithm dependency on fraud behaviors will be eliminated, which enables eFraudCom to detect fraud behaviors in presence of the new fraud patterns; (3) the mutual information regularization term can maximize the separability between normal and fraud behaviors to further improve CGNN. We implement eFraudCom into a prototype system and evaluate it on the e-commerce platform "Taobao". The experiments on two Taobao and two public datasets demonstrate that the proposed deep framework CGNN is superior to other baselines in detecting fraud behaviors. A case study on Taobao datasets verifies that CGNN is still robust when the fraud patterns have been upgraded.

CCS Concepts: • **Applied computing** → **Online shopping**; • **Computing methodologies** → **Neural networks**.

Additional Key Words and Phrases: Online E-commerce Platforms, Fraud Detection System, Graph Neural Networks

ACM Reference Format:
Ge Zhang, Zhao Li, Jiaming Huang, Jia Wu, Chuan Zhou, and Jian Yang. 2018. eFraudCom: An E-commerce Fraud Detection System via Competitive Graph Neural Networks. In *Woodstock '18: ACM Symposium on Neural Gaze Detection, June 03–05, 2018, Woodstock, NY*. ACM, New York, NY, USA, 29 pages. https://doi.org/10.1145/1122445.1122456

*Corresponding Author
[1]http://www.alibaba.com

Manuscript submitted to ACM

1 INTRODUCTION

Undoubtedly, online e-commerce has been attracting a significant number of users and creating huge economic benefits. According to statistics, the global e-commerce market crossed two trillion dollars in 2020[2]. Taobao, the largest e-commerce platform in China, shows an increase of over one trillion dollars in annual Gross Merchandise Volume (GMV) from April 2019 to March 2020. Taobao platform calculates ranking indices and reputation factors for online merchants using the behaviors of users, e.g. clicks, add-to-favorite, transactions, and reviews. In general, items with a better ranking index will be listed in the front of the search results and users prefer those items with a good reputation. Regular ways to boost these measurements include providing high-quality items, good services, and advertising, which usually takes much effort. It motivates some dishonest merchants to promote their items by fraud behaviors [13, 29, 32, 41, 52, 58]. As shown in Fig. 1, a heterogeneous graph can be constructed by the behaviors taken by users on items, where the edges between users and items can be constructed by one of the behaviors, such as clicks, add-to-favorite, and transactions.

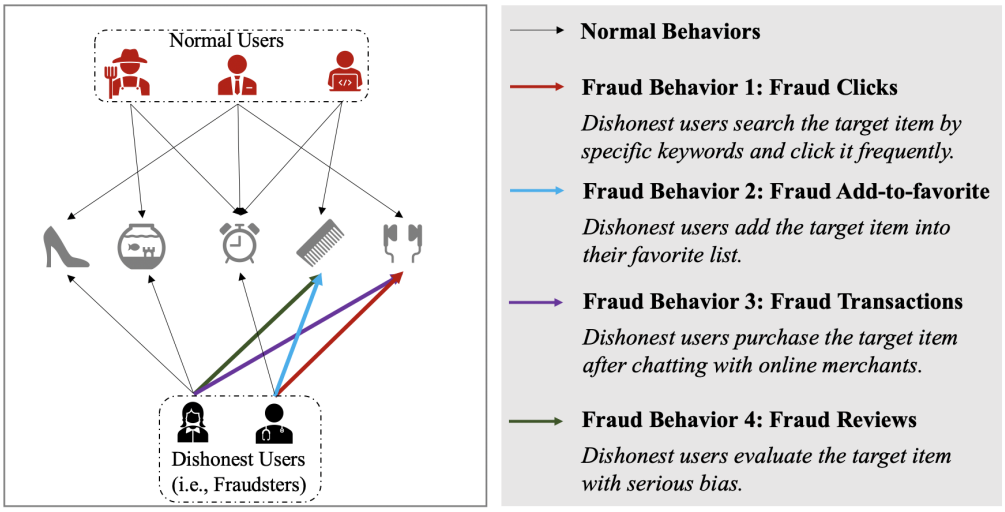

Fig. 1. Different fraud behaviors on the Taobao platform.

Such fraud behaviors lead to serious consequences: 1) For users, they are misled to purchase items seemingly only good on sales volume over quality; 2) For regular merchants, their incomes are directly affected, which causes unfair competition; 3) For e-commerce platforms, it increases the difficulty of recognizing good sellers and establishing the fair seller reputation system.

E-commerce platforms usually treat the fraud detection task as a classification problem and propose different systems to detect fraud behaviors. For example, some algorithms [27, 28, 50–52] treat confirmed fraud behaviors as "fraudster seeds", which are predefined by domain experts and are utilized as supervision information to detect other fraud items or dishonest users. Due to the adversarial actions taken by fraudsters and the rapid mutation of fraud patterns, these "fraudster seeds" based systems suffer from performance decay [19], as shown in Fig. 2. The reason is that these "fraudster seeds" based systems essentially try to learn the distributions of fraud information. However, it is difficult to

[2]http://www.statista.com/study/42335/ecommerce-report/

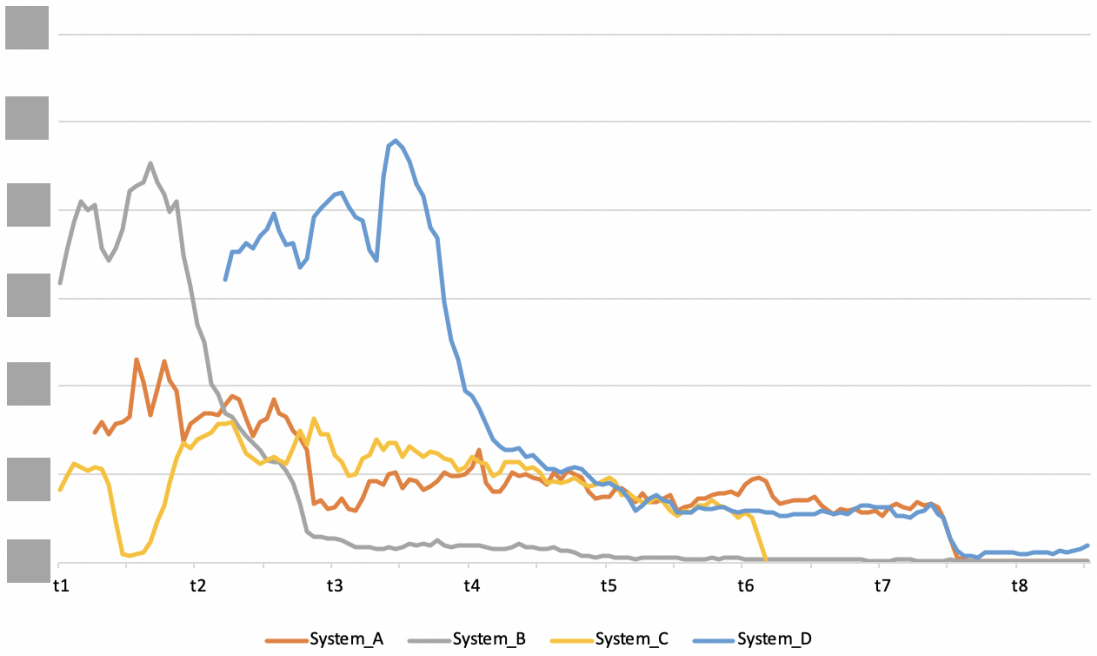

Fig. 2. The figure shows the change of recall ability of four different fraud detection systems based on the "fraudster seeds" with the working time. Both the y-axis and x-axis values are deliberately hidden for confidentiality reasons. Specifically, the x-axis represents the date from t1 to t8 (in recent years), and the y-axis represents the number of fraud behaviors have been recalled. As shown in this figure, the recall ability of system_B and system_D dropped deeply within a short time. System_A and System_C failed to recall any fraudsters after t6 and t7, respectively.

model the fraud distributions accurately which are highly unpredictable and unstable. Due to the upgrade of fraud patterns, the distributions of fraud behaviors will be changed frequently. However, limited by the fraud pattern modeled by the "fraudster seeds", most fraud behaviors detected by "fraudster seeds" based systems follow the already known fraud patterns[13]. Besides, e-commerce platforms generate tremendous data every day, most of which are normal, labeling a part of fraud behaviors as "fraudster seeds" has prohibitive costs. One possible solution to alleviate the above problems is to propose a fraud detection algorithm that can abandon the dependence on "fraudster seeds".

Recently, some anomaly detection algorithms [9, 10, 22] employing Graph Auto-Encoder (GAE) have achieved some empirical success. These algorithms combine Graph Neural Networks (GNNs) [7, 14, 16, 54, 57] and Auto-Encoder framework [12] to detect anomalous nodes which are significantly different from other majority nodes in graph-structured data in an unsupervised manner. The GAE framework-based anomaly detection algorithms follow the idea that the Auto-Encoder framework can project the data from different data spaces to a broader range distance, the normal data as the majority can be well learned and the Auto-Encoder framework can provide smaller reconstruction errors for it; on the contrary, samples with high reconstruction errors are usually anomalous. Although this kind of unsupervised method has the potential to help fraud detection systems remove the dependence on "fraudster seeds", two challenges are faced:

- **The threshold selection issue**: Existing reconstruction-based algorithms can only classify normal samples and anomalous ones by setting a threshold for reconstruction errors. However, it is hard to predefine a threshold that works for all cases.
- **The robustness issue**: A significant number of anomalous samples could also lurk with a normal level of reconstruction errors [59]. On the e-commerce platforms, the issue would be more serious since the dishonest users hired by third-party fraud service platforms[3] are likely to camouflage their fraud behaviors by imitating normal users' behaviors. In this case, the classical reconstruction-based method is also hard to adapt to the pattern mutation taken by third-party fraud service platforms although it has the potential to eliminate algorithm dependence on "fraudster seeds".

Faced with the above challenges, we propose eFraudCom, a competitive graph neural networks (CGNN) based deep e-commerce fraud detection system, works on the heterogeneous graph built by users' behaviors on items. Different from the classic Auto-Encoder, the proposed architecture CGNN is composed of one graph encoder and two graph decoders. These two graph decoders, after a few rounds of training, one graph decoder will be specialized in reconstructing normal behaviors referred to as the *inlier decoder*, while the other graph decoder will be specialized in reconstructing fraud behaviors referred to as the *outlier decoder*. These two graph decoders work competitively due to they provide different reconstruction errors for inputs, but for each input behavior, CGNN only adopts the result of the graph decoder which can provide a smaller reconstruction error for it.

To make the *inlier decoder* and the *outlier decoder* clear their responsibilities and compete orderly, CGNN utilizes a part of normal behaviors, which can be obtained easily, to build the profile for normal behaviors due to the distribution of normal behaviors is much more stable compared with the fraud behaviors'. These normal behaviors, as weak supervision, are only reconstructed by the *inlier decoder*. With the guidance of weak supervision information, after a period of training: 1) the graph encoder can model the stable distribution of normal behaviors accurately and map normal and fraud behaviors to different hidden subspaces; 2) the *inlier decoder* becomes specialized in reconstructing normal behaviors and; 3) the fraud behaviors will be assigned to the *outlier decoder*. Specifically, in the well-trained CGNN, for an unlabeled behavior, although both graph decoders can provide a reconstruction error for it, if its a normal behavior, it will be reconstructed by the *inlier decoder* due to a well-trained *inlier decoder* can provide the smaller reconstruction error for it; one the contrary, it will be reconstructed by the *outlier decoder*. By designing two competitive graph decoders and regard them as a discriminator, predefining a threshold for reconstruction errors to classify normal samples and anomalous is no longer needed. Besides, the new fraud patterns-based fraud behaviors, which camouflage themselves by approaching normal behaviors in the data space are difficult to be learned by the classic Auto-Encoder due to they are likely to hide in a normal level of reconstruction errors. However, these fraud behaviors can also be detected by CGNN. The encoder and the *inlier decoder* in CGNN can accurately model and reconstruct the normal behaviors under the guidance of weak supervision information, and these fraud behaviors will be assigned to the *outlier decoder* due to they do not completely obey the stable distribution of normal behaviors.

The graph encoder in CGNN shared by *inlier decoder* and *outlier decoder* can maps normal and fraud behaviors from data space into different subspaces. To further improve CGNN, it is reasonable to add a regularization term on the subspace to maximize the separability of the normal and fraud behaviors. Thus, we introduce a regularizer

[3]Dishonest online merchants on the Taobao platform usually turn to third-party fraud service platforms and post tasks on these platforms. To earn the commission, some people will register on these platforms and help dishonest merchants improve the ranking of target items by fraud behaviors. The third-party fraud platforms upgrade fraud patterns frequently to make dishonest users employed by them difficult to be detected by fraud detection systems.

(named mutual information regularization) based on the graph mutual information computation on neural networks [31, 33, 43, 56] to preserve as much information related to normal and fraud behaviors in the embedding subspace. Overall, this paper makes the following contributions:

- Firstly, we propose eFraudCom, a deep e-commerce fraud detection system, which eliminates the algorithm dependence on fraud behaviors by discarding the "fraudster seeds" which would restrict the ability of detection methods and regrading the normal behaviors as weak supervision information.
- Secondly, we treat both the *inlier decoder* and *outlier decoder* in the competitive graph neural networks (CGNN) as a discriminator to model the normal and fraud behaviors separately and get rid of hyper-parameter selection issue faced by classical GAE based detection methods.
- Thirdly, mutual information regularization would further separate the normal and fraud behaviors by reserving fraud information on embedding space to the extent to cope with unpredictable fraud patterns.
- Fourthly, the system is evaluated on one of the largest real-world e-commerce platforms. Through the evaluation, we demonstrate that eFraudCom is effective in practical scenarios: 1) it has a strong ability to detect fraud behaviors on e-commerce platforms and; 2) it can alleviate the negative impact brought by fraud pattern mutation and adversarial actions on the fraud detection system.

2 RELATED WORK

Firstly, we revisit the related work about graph anomaly detection algorithms based on reconstruction and other techniques. Secondly, fraud detection systems on e-commerce platforms and other industries are surveyed.

2.1 Anomaly Detection on Graph-structured Data

Most anomaly detection algorithms on graph-structured data having network topology and nodal attributes are dedicated to detecting the anomalous nodes which are significantly different from the other majority normal nodes. These anomaly detection algorithms usually utilize the reconstruction errors between the original and reconstructed networks to measure the anomalous degree of each node in an unsupervised manner. Compared with normal samples, anomalies are harder to be reconstructed [2, 59] and have high reconstruction errors. For instance, Radar [21] characterizes the residuals of attribute information and its coherence with network topology by matrix decomposition to do anomaly detection. Peng et al. [34] propose a joint framework named ANOMALOUS to conduct attribute selection and anomaly detection as a whole based on CUR decomposition and residual analysis. Bandyopadhyay et al. [3] propose an outlier resistant unsupervised deep architecture for both node embedding and anomaly detection tasks on attributed networks, which utilizes the conventional autoencoder framework composed of fully-connect neural layers to model and reconstruct network topology and nodal attributes separately to measure the suspiciousness of nodes.

In addition to the above reconstruction errors-based methods, there are also some other techniques-based anomaly detection methods. For example, Perozzi and Akoglu [35] consider ego-networks to characterize anomalous neighbors and detecting communities in the attributed network by measuring community normality. Liu et al. [25] propose an anomaly detection algorithm named ALAD to detect local anomalous nodes which well align with the global condition but inconsistent with other nodes in community structures [23]. ALAD utilizes the Nonnegative Matrix Factorization (NMF) based context extraction and local suspiciousness evaluation to differentiate normal and anomalous nodes in graphs and introduce a distributed algorithm for accelerated optimization. Wang et al. [49] propose a Bayesian generative model-based multi-view anomaly detection algorithm, which utilized plenty of labeled normal data as

supervision information to detect anomalous nodes by enabling the detector to capture normal behaviors. Bojchevski and Günnemann [5] propose a probabilistic generative model named PAICAN that explicitly models partial anomalies and clusters nodes in attributed networks by generalizing the idea of Degree Corrected Stochastic Block Models and Bernoulli Mixture Models. Despite their empirical success, none of the above techniques have the capability like GCN in solving the challenges faced by graph learning tasks such as network sparsity, data nonlinearity, and complex modality interactions [10].

With the rocketing growth of Graph Convolution Neural Networks (GCN) [7, 14, 16, 54, 57], some anomaly detection methods based on GCN and autoencoder framework [12] have been proposed. For instance, anomaly detection algorithms [9–11, 22] explicitly model the topological structure and nodal attributes seamlessly for node embedding learning with the prevalent GCN technique, and the anomaly detection problem is solved by virtue of deep autoencoder that leverage the learned embeddings to reconstruct the original data, nodes with high reconstruction errors regarded as anomalies. Although anomaly detection algorithms on attribute networks which make use of reconstruction errors to be anomalous scores of nodes have achieved some empirical success, Zong et al. [59] propose that anomalous samples could be different from the normal samples and some of them do demonstrate unusually high reconstruction errors, but a significant amount of anomalous samples could also lurk with the normal level of reconstruction errors.

2.2 Fraud Detection on Industries

Fraud detection is a practical issue faced by various industries including large-scale e-commerce platforms and other industries such as online social networks and online review platforms. In many fields, fraud detection work would help improve user experience and create more economic value.

Fraud detection on e-commerce platforms: On e-commerce platforms, many fraud detection algorithms have focused on finance crimes such as credit card frauds and risky account [4, 6, 8, 26, 27, 30, 36, 45, 48]. For example, Santiago et al. [39] propose a comprehensive approach to address the problem of credit card fraud detection in the emerging market of online payment services. Wang et al. [27] propose an algorithm based on deep learning to detect risky accounts which share dense connections within a fraud community [42]. There are also some works focusing on detecting fraud items with false high sales on e-commerce platforms. For instance, Weng et al. [52] propose a novel online e-commerce fraud detection system in 2018, which can detect fraud items on the Taobao platform by fraudster seed identification (i.e., identifying some fraud items beforehand as supervision information) and fraud behaviors propagation strategy. Weng et al. [51] also propose a fraud detection algorithm in 2019, which consists of four components: a data collector, a semantic analyzer, a feature extractor. The "fraudster seed" predefined by domain experts beforehand plays an important role in these two algorithms. However, it is not conducive to the detection of new fraud behaviors.

Fraud detection on other industries: Fraud is also a tricky problem faced by online social networks and online review platforms which can provide people with public comment services [20, 40, 47, 53, 55]. For example, on online social networks, malicious users try to harvest private user data, spread rumors, and even interfere in political elections. Wang et al.[44] propose an algorithm named GANG based on a novel pairwise Markov Random Field (pMRF), which detects malicious users who are keen to achieve illegal purposes by fraud behaviors. GANG detects these users who always camouflage themselves by making lots of one-way connections with benign users by modeling the social network as a directed graph. Additionally, on review systems, opinion spammers seek to increase sales and improve the reputation of products with poor quality by fraud reviews. Akoglu et al. [1] propose a fast and effective framework based on pMRF for spotting these users who have fraud behaviors on online review systems. The algorithm pays much

attention to the emotion of reviewers instead of the review content as those fraud users always try to camouflage themselves by imitating the writing style of high-quality users. Li et al. [19] propose a spam review detection system by utilizing the local and global context of comments, which works on the largest second-hand goods app "Xianyu" of Alibaba. The system can detect fraudsters who camouflage themselves by using different expressions with similar meaning and deforming the comments. Overall, fraud organizers in various industries try to use different fraud patterns to camouflage themselves and evade detection.

3 PROBLEM DEFINITION

Following the commonly used notations, we use the italics uppercase letter to denote set (e.g., U), the bold uppercase to denote matrix (e.g., $\mathbf{X}$) and the bold lowercase letter to denote vector (e.g., $\mathbf{x}$) in this paper.

We define a heterogeneous network $G = \{U, V, E^{no}, E^{un}, \mathbf{X}_U, \mathbf{X}_V, \mathbf{X}_{E^{no}}, \mathbf{X}_{E^{un}}\}$ consisting of: (1) the set U of users, where $U = \{u_i | i = 1, 2, ...\}$; (2) the set V of items, where $V = \{v_j | j = 1, 2, ...\}$; (3) the set of edges, $E = \{E^{no}, E^{un}\}$. The edges in G constructed by one type of users' behaviors (e.g., add-to-favorite). As aforementioned, in eFraudCom, we use some normal behaviors as weak supervision information, so the edge set E is divided into two parts, i.e., normal and unlabeled. Specifically, we use E^{no} to denote the set of normal edges unrelated to any fraud behaviors, the edge e_l^{no} in E^{no} is normal. These normal edges come from the representative samples detected by domain experts beforehand. Similarly, we use E^{un} to denote the set of unlabeled edges, the edge e_r^{un} in E^{un} is unlabeled. It should be noted that for the convenience of mathematical representation, we may use e_k in E to represent the edge e_l^{no} in E_{no} or e_r^{un} in E_{un} in the following when a mathematical operation need to be carried out on normal and unlabeled edges both and; (4) the attribute matrix $\mathbf{X} = \{\mathbf{X}_U, \mathbf{X}_V, \mathbf{X}_{E^{no}}, \mathbf{X}_{E^{un}}\}$. We use $\mathbf{x}_{u_i}$, $\mathbf{x}_{v_j}$, $\mathbf{x}_{e_l^{no}}$ and $\mathbf{x}_{e_r^{un}}$ to denote the attribute vectors of users $u_i, \forall u_i \in U$, items $v_j, \forall v_j \in V$, normal edges $e_l^{no}, \forall e_l^{no} \in E^{no}$ and unlabeled edges $e_r^{un}, \forall e_r^{un} \in E^{un}$, respectively. Similarly, we use $\mathbf{x}_{e_k}$ to denote the attribute vector of edge e_k when we need to use e_k to substitute e_l^{no} or e_r^{un}.

Given the network G, our task is to learn the binary labels of all unlabeled edges in E^{un}. Y is the label assignment set for all unlabeled edges, $y_r = 1$ if the unlabeled edge e_r^{un} related to fraud behaviors, and it can be reconstructed better by the *outlier decoder*, while $y_r = 0$ if unlabeled edge e_r^{un} unrelated to any fraud behaviors and it can be reconstructed better by the *inlier decoder*.

4 THE DEEP FRAUD DETECTION SYSTEM: EFRAUDCOM

The architecture of the deep system eFraudCom composed of a data processor and a fraud detector is illustrated in Fig. 3. The MaxCompute platform[4] and the GraphScope platform[5] shown in Fig. 3 both provided by Alibaba.

Next, we will describe the data processor in eFraudCom in section 4.1. In the fraud detector, the CGNN as the core part will be explained in section 4.2 and section 4.3. The mutual information regularization and loss function will be introduced in section 4.4. Finally, we will discuss the time complexity of CGNN and the implementation details of eFraudCom in section 4.5 and section 4.6, respectively.

4.1 Data Processor

As shown in Fig. 3, the function of the data processor is processing data and constructing graphs. When there is an edge formed by user behaviors on items on the Taobao platform, the logs will be stored on the MaxCompute platform. As aforementioned, CGNN needs some confirmed normal behaviors as weak supervision information, so domain experts

[4]http://cn.aliyun.com/product/odps
[5]https://github.com/alibaba/GraphScope

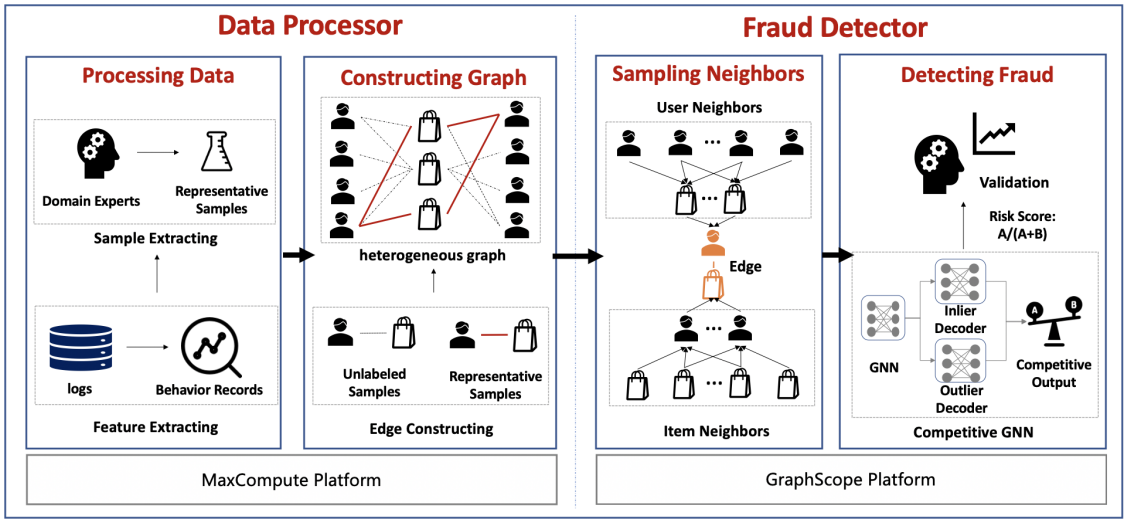

Fig. 3. eFraudCom: System Architecture. As observed, eFraudCom mainly includes two parts: a data processor and a fraud detector. The data processor contains two components. One is used to process data with domain experts for sampling few representative normal samples after collecting data in a real-time fashion from the MaxCompute platform. The other is designed for constructing a heterogeneous graph with representative samples and unlabeled ones. The fraud detector also contains two components. One is used for sampling neighbors on the graph, and the other is for detecting edges related to fraud behaviors by the competitive graph neural networks (CGNN) on the GraphScope platform and outputting a risk score for each suspicious edge (the domain experts are required to make validation on sampled outputs for evaluating the performance periodically).

detect some representative samples from MaxCompute beforehand and abstract some normal samples from these representative instances. All the remaining stored by MaxCompute will be regarded as unlabeled samples. Then, based on the representative and unlabeled samples, a large-scale heterogeneous graph is constructed by the data processor.

4.2 Fraud Detector: Graph Encoding

As shown in Fig. 3, the main function of the fraud detector is to measure the suspiciousness of all unlabeled input instances. The core part of the fraud detector is the competitive graph neural networks (CGNN) illustrated in Fig. 4. As shown in Fig. 4, CGNN mainly includes two parts: (1) the graph encoding, in which the latent representations of edges are learned by subpart-1 composed of fully-connected dense layers and the latent representations of the corresponding user and item pairs are learned by subpart-2 composed of two convolution layers; (2) dual graph decoders (i.e., the *inlier decoder* D_{in} and the *outlier decoder* D_{out}), which can be regarded as a discriminator and learn the labels of edges in a competitive mechanism.

As mentioned in section 3 PROBLEM DEFINITION, for simplicity of expression, symbol e_k can be utilized to represent normal and unlabeled edges when an operation need to be both taken on normal and unlabeled edges. In the next paper, normal edge e_l^{no} and its embedding $\mathbf{h}_{e_l^{no}}$ may be represented by e_k and $\mathbf{h}_{e_k}$, respectively. Similarity, unlabeled edge e_r^{un} and its embedding $\mathbf{h}_{e_r^{un}}$ may also be represented by e_k and $\mathbf{h}_{e_k}$, respectively. The reason is that CGNN need to generate latent representations for all edges. For example, as shown in Fig. 4, we use e_k to represent e_l^{no} and e_r^{un} and point out whether e_k is normal or unlabeled.

The graph encoding part composed of subpart-1 and subpart-2 will be discussed in detail next.

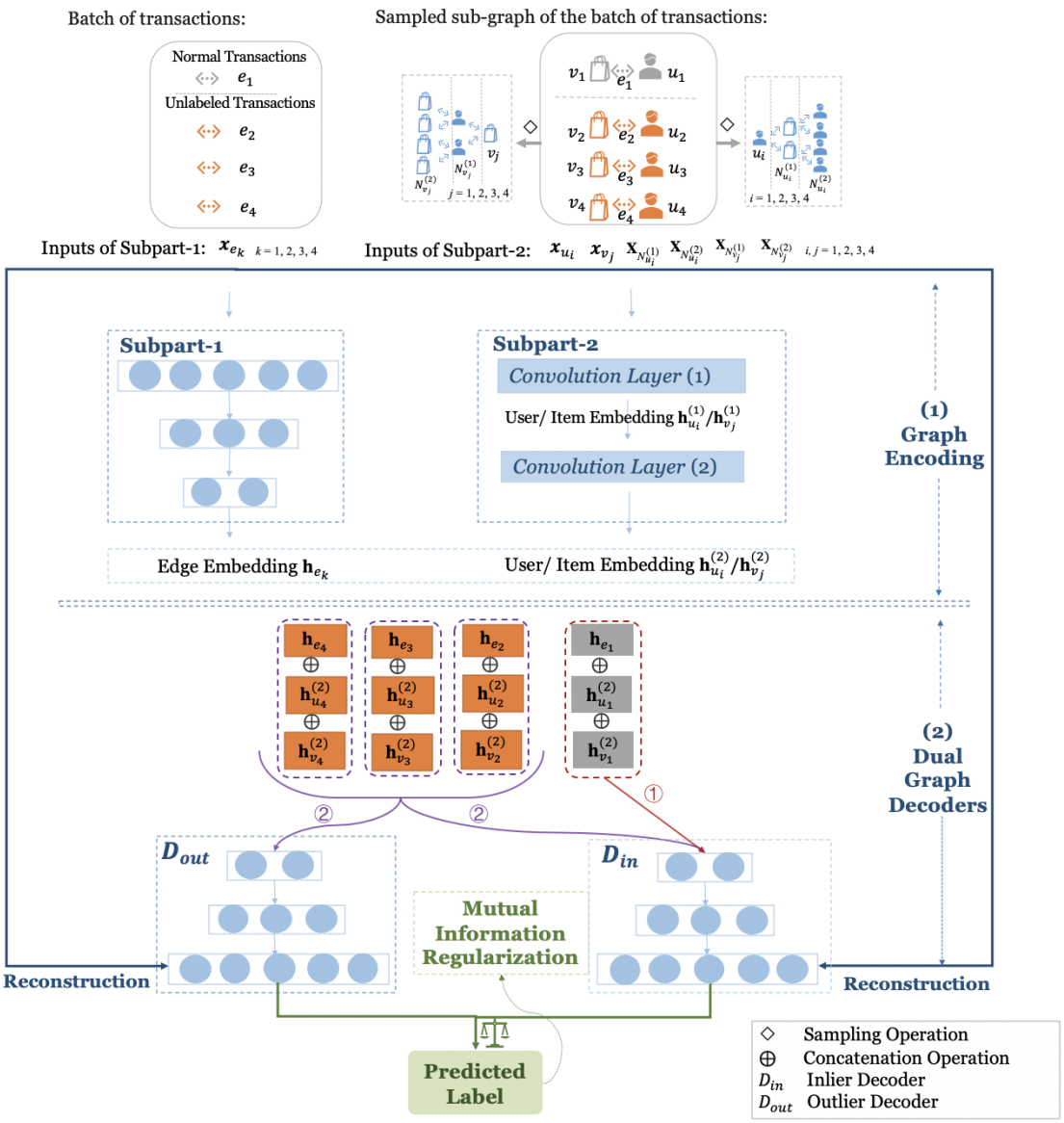

Fig. 4. The overall deep architecture of CGNN. Supposing that the heterogenous graph between users and items built by users' transaction behaviors. As shown in the figure, the formation of e_1, e_2, e_3 and e_4 due to u_1's, u_2's, u_3's and u_4's transaction behavior on v_1, v_2, v_3, and v_4, respectively. The inputs of subpart-1 include the attributed vector $\mathbf{x}_{e_k}$ of edge e_k, $k = 1, 2, 3, 4$, where e_1 is normal, e_2, e_3 and e_4 are unlabeled. We use $N_{u_i}^{(1)}$, $N_{u_i}^{(2)}$ to denote the sets of 1-depth and 2-depth neighbors of user u_i, respectively, $N_{v_j}^{(1)}$, $N_{v_j}^{(2)}$ to denote the sets of 1-depth and 2-depth neighbors of item v_j, respectively. The inputs of subpart-2 include the attribute vector $\mathbf{x}_{u_i}$ and $\mathbf{x}_{v_j}$, and the attribute matrices of 1-depth and 2-depth neighbors, $\mathbf{X}_{N_{u_i}^{(1)}}$, $\mathbf{X}_{N_{v_j}^{(1)}}$, $\mathbf{X}_{N_{u_i}^{(2)}}$, $\mathbf{X}_{N_{v_j}^{(2)}}$, $i, j = 1, 2, 3, 4$. In CGNN, if an edge is normal, it will be processed according to the arrow labeled with ①, that is, only be reconstructed by the *inlier decoder* D_{in}; If an edge is unlabeled, it will be processed according to the arrow labeled with ②, that is, reconstructed by the *inlier decoder* D_{in} and the *outlier decoder* D_{out} in a competitive mechanism at the same time.

Subpart-1 in CGNN: After constructing the heterogeneous graph by the data processor, the fraud detector separates graph data into different batches by sampling at first, the process is illustrated in Fig. 5 (a). As shown in Fig. 5 (a), edges, users, and items marked by red are sampled into a batch, the remaining edges will be sampled into other batches.

Subpart-1 is responsible for generating the embedding $\mathbf{h}_{e_k}$ for edges e_k by fully-connected dense layers. As shown in Fig. 4, the inputs of subpart-1 are the attribute vectors $\mathbf{x}_{e_k}$ of edges in each batch.

Subpart-2 in CGNN: Subpart-2 is responsible for learning the final embedding representation $\mathbf{h}_{u_i}^{(2)}$ and $\mathbf{h}_{v_j}^{(2)}$ for user u_i and item v_j related to edges e_k learned by subpart-1. This part consists of two convolution layers specially designed for the e-commerce scenario, which learns the embedding of users and items by aggregating information of their 1-depth and 2-depth neighborhoods. It is worth noting that we uniformly sample a fixed-size set of 1-hop and 2-hop neighbors for each user and item based on the link relationship among nodes, instead of using all neighbors.

Taking learn the embedding for a user as an example, we explain why CGNN needs to aggregate information from 1-depth and 2-depth neighbors of the target user instead of only utilize the attributes of the target user node itself to learn its embedding. In general, a fraudulent user is unlikely to promote only one malicious item. That is to say, likely, most of the items purchased or added to the favourite list by the fraudulent user related to fraud promotion, and the fraud information will also be reflected in edges generated by target user's behaviors on items. It is very important to aggregate the 1-depth neighbor information to the target user, which can make up for the sparse attribute information of a single user node. Similarly, a malicious item will be promoted by more than one fraudulent user, so the 2-depth neighbors of a fraud user are likely to relate to fraud behaviors. This is the reason that we also integrate the information of 2-depth neighbors to the target user. Based on a similar principle, CGNN aggregates the information of 1-depth and 2-depth neighbors to the target item when learning the embeddings of items.

The processes of sampling neighborhoods based on link relationship among nodes and forward propagation process of convolution layers are discussed in detail next:

- **Sampling neighbors:** In order to get the final embedding representation $\mathbf{h}_{u_i}^{(2)}$ and $\mathbf{h}_{v_j}^{(2)}$ for user u_i and item v_j by convolution operations, we firstly need to sample the 1-depth neighbors $N_{u_i}^{(1)}$, $N_{v_j}^{(1)}$ and the 2-depth neighbors $N_{u_i}^{(2)}$ and $N_{v_j}^{(2)}$ for user u_i and item v_j, respectively. The detailed sampling process is shown in Fig. 5 (b) and Fig. 5 (c). Different from other graph convolution based algorithms, we take both node and edge neighbors into consideration when sampling neighbors for users and items. As shown in Fig. 5 (b) and Fig. 5 (c), taking how to get the 1-depth neighbor set $N_{u_i}^{(1)}$ of user u_i as an example, in this work, we uniformly sample a fixed-size set of 1-depth item neighbors from all 1-depth-item neighbors of user u_i into $N_{u_i}^{(1)}$, the edges between user u_i and its sampled 1-depth item neighbors also assigned into $N_{u_i}^{(1)}$. Then, we need to sampled the 2-depth neighbors $N_{u_i}^{(2)}$ consisted of 2-depth user neighbors and 2-depth edge neighbors for user u_i. For item v_j, the 1-depth neighbors $N_{v_j}^{(1)}$ and 2-depth neighbors $N_{v_j}^{(2)}$ can be sampled with the same sampling operation.
- **Forward propagation strategy:** Once we get the 1-depth and 2-depth neighbors of user u_i and item v_j, the next process is the forward propagation process completed by two convolution layers. As shown in Fig. 4, edge e_k is formed by user u_i's behavior on item v_j, the inputs of subpart-2 are the attribute vectors $\mathbf{x}_{u_i}$ of user u_i, the attribute vectors $\mathbf{x}_{v_j}$ of item v_j, and the attribute matrix $\mathbf{X}_{N_{u_i}^{(1)}}$, $\mathbf{X}_{N_{u_i}^{(2)}}$, $\mathbf{X}_{N_{v_j}^{(1)}}$ and $\mathbf{X}_{N_{v_j}^{(2)}}$ of u_i's and v_j's 1-depth neighbors and 2-depth neighbors, respectively. The core process of subpart-2 is the localized convolution operation, which can learn embeddings for users and items by aggregating information from neighborhoods, the detailed convolution process is shown in **Algorithm 1**.

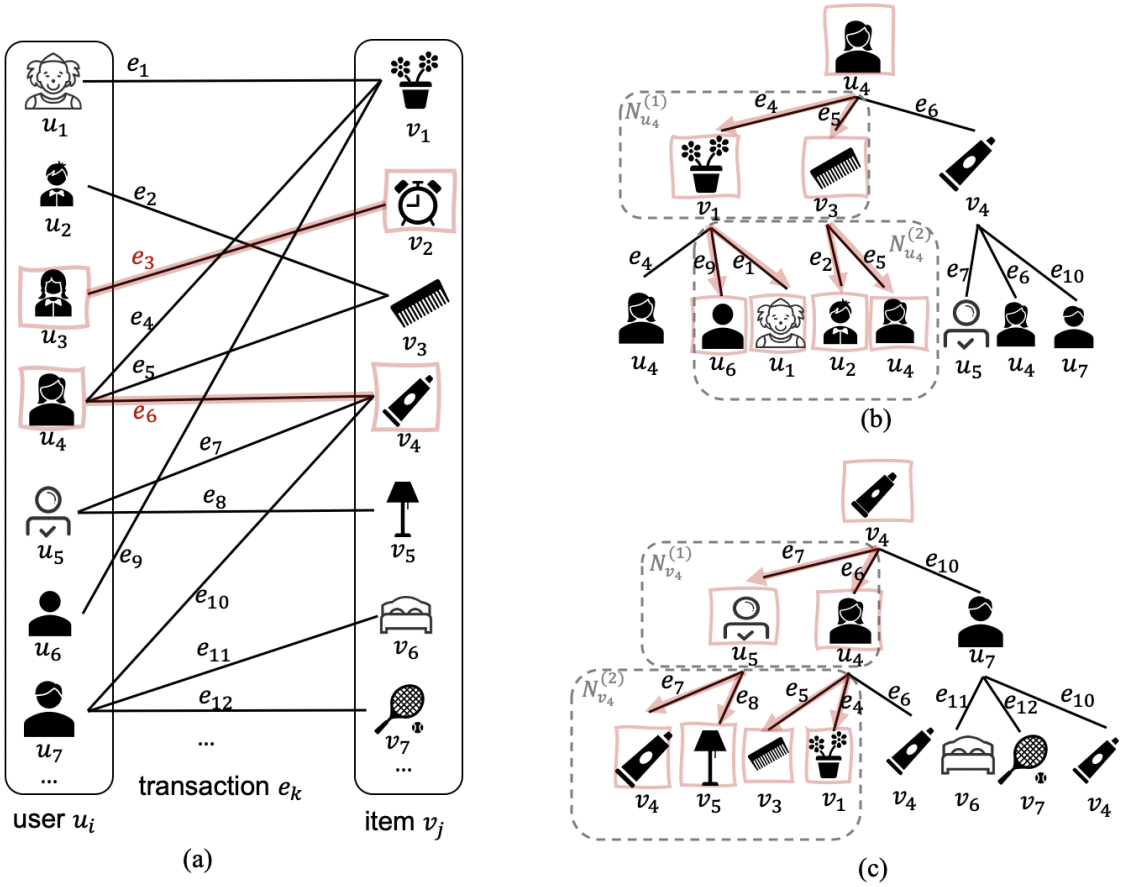

Fig. 5. A simple illustration of sampling process. Supposing the heterogeneous graph between users and items built by users' transaction behaviors. (a) is an illustration of sampling edges from a heterogeneous network in the e-commerce scenario. As shown in (a), users from u_1 to u_7 connect to items from v_1 to v_7 by edges from e_1 to e_{12}. Edges e_3 and e_6 and the related user-item pairs u_3-v_2 and u_4-v_4 marked by red sampled in to a batch, the remaining edges marked by black will be sampled into other batches. Then, taking edge e_6 as an example, (b) and (c) show that how to sample the 1-depth and 2-depth neighbors of user u_4 and item v_4, which are the ends of edge e_6. We utilize $N_{u_4}^{(1)}$, $N_{v_4}^{(1)}$, $N_{u_4}^{(2)}$ and $N_{v_4}^{(2)}$ to denote sets of the 1-depth and 2-depth neighbors of user u_4 and item v_4, respectively, neighborhoods in these four sets are uniformly sampled based on the link relationship among nodes. Different from other algorithms, we take both node and edge neighbors into consideration when sampling neighborhoods for users and items. Taking user u_4 as a example, as shown in (b), $N_{u_4}^{(1)}$ includes 1-depth item neighbors v_1 and v_3 and 1-depth edge neighbors e_4 and e_5, and $N_{u_4}^{(2)}$ includes 2-depth user neighbors u_6, u_1, u_2, u_4 and 2-depth edge neighbors e_9, e_1, e_2 and e_5. The 1-depth and 2-depth neighbors of item v_4 can be sampled by the same way. As shown in (c), $N_{v_4}^{(1)}$ includes users u_4 and u_5 and edges e_6 and e_7, $N_{v_4}^{(2)}$ includes items v_4, v_5, v_3, v_1 and edges e_7, e_8, e_5, e_4.

To facilitate readers to understand the convolution operations in this work easily, we also draw an illustration to explain the process of learning the embedding for user u_i, as shown in Fig. 6. Simply speaking, the final embedding results $\mathbf{h}_{u_i}^{(2)}$ of user u_i depends on the concatenation between feature transformation of the first

layer embedding $\mathbf{h}_{u_i}^{(1)}$ and the embedding representation $\mathbf{H}_{N_{u_i}^{(1)}}$, which is the embedding matrix of the 1-depth neighbors of user u_i and it can be learned by aggregating the 2-depth neighbors of user u_i.

After getting the embeddings of edges and corresponding user and item pairs, these three parts will be concatenated as the input of the next part of CGNN.

Algorithm 1: Convolution process of user u_i

Input: The attribute vector $\mathbf{x}_{u_i}$ of user u_i;
The attribute vectors $\mathbf{x}_{v_{j'}}$ and $\mathbf{x}_{e_{k'}}$, $\forall v_{j'}, e_{k'} \in N_{u_i}^{(1)}$;
The attribute vectors $\mathbf{x}_{u_{i''}}$ and $\mathbf{x}_{e_{k''}}$, $\forall u_{i''}, e_{k''} \in N_{u_i}^{(2)}$;
The weight matrics and bias.

Output: The final embedding $\mathbf{h}_{u_i}^{(2)}$ of user u_i.

Initial: The weight matrices $\mathbf{W}$, $\mathbf{Q}$, $\mathbf{S}$, $\mathbf{P}$, $\mathbf{Z}$ and $\mathbf{B}$ and bias b_1, b_2, b_3, b_4, b_5 and b_6.

// Calculating the first-order embedding $\mathbf{h}_{u_i}^{(1)}$ of user u_i.

$$\mathbf{\Psi} \leftarrow \sigma\left(\mathbf{W} * \mathbf{x}_{u_i} + b_1\right)$$

$$\mathbf{\Omega} \leftarrow \sigma\left(\mathbf{Q} * Aggregate\left(\left\{Concat\left(\mathbf{x}_{v_{j'}}, \mathbf{x}_{e_{k'}}\right), \forall v_{j'}, e_{k'} \in N_{u_i}^{(1)}\right\}\right) + b_2\right)$$

$$\mathbf{h}_{u_i}^{(1)} \leftarrow \sigma\left[Concatenate\left(\mathbf{\Psi}, \mathbf{\Omega}\right)\right]$$

// Calculating the embedding $\mathbf{H}_{N_{u_i}^{(1)}}$.

$$\mathbf{\Psi} \leftarrow \sigma\left(\mathbf{S} * \mathbf{x}_{v_{j'}} + b_3\right), \forall v_{j'} \in N_{u_i}^{(1)}$$

$$\mathbf{\Omega} \leftarrow \sigma\left(\mathbf{P} * Aggregate\left(\left\{Concat\left(\mathbf{x}_{u_{i''}}, \mathbf{x}_{e_{k''}}\right), \forall u_{i''}, e_{k''} \in N_{u_i}^{(2)}\right\}\right) + b_4\right)$$

$$\mathbf{H}_{N_{u_i}^{(1)}} \leftarrow \sigma\left[Concatenate\left(\mathbf{\Psi}, \mathbf{\Omega}\right)\right]$$

// Calculating the second-order embedding $\mathbf{h}_{u_i}^{(2)}$ of user u_i.

$$\mathbf{\Psi} \leftarrow \sigma\left(\mathbf{Z} * \mathbf{h}_{u_i}^{(1)} + b_5\right)$$

$$\mathbf{\Omega} \leftarrow \sigma\left(\mathbf{B} * \mathbf{H}_{N_{u_i}^{(1)}} + b_6\right)$$

$$\mathbf{h}_{u_i}^{(2)} \leftarrow \sigma\left[Concatenate\left(\mathbf{\Psi}, \mathbf{\Omega}\right)\right]$$

// "*Concatenate*" and "*Aggregate*" denote the conatenation operation and aggregation operation, respectively. We use mean aggregator in this paper. σ is activation function "*ReLu*".

//The final embedding $\mathbf{h}_{v_j}^{(2)}$ of item v_j can be learned by the similar procedure.

4.3 Fraud Detector: Dual Graph Decoders

To model the distributions of edges related or unrelated fraud behaviors separately and acquire their binary labels directly, we design dual graph decoders in CGNN, which is composed of (1) the *inlier decoder* D_{in} and; (2) the *outlier decoder* D_{out}. D_{in} and D_{out} work in a competitive way as a discriminator since they both try to give a low reconstruction error for the inputs. However, each unlabeled edge can only be interpreted by one graph decoder due to the normal and fraud behaviors follow different distributions.

We define the reconstruction errors $R_{in}(\mathbf{x}_{e_r^{un}})$ and $R_{out}(\mathbf{x}_{e_r^{un}})$ of an unlabeled edge e_r^{un} calculated by *inlier decoder* and *outlier decoder* as:

$$R_{in}(\mathbf{x}_{e_r^{un}}) = D_{in}\left(Concatenate\left(\mathbf{h}_{u_i}^{(2)}, \mathbf{h}_{v_j}^{(2)}, \mathbf{h}_{e_r^{un}}\right)\right), \tag{1}$$

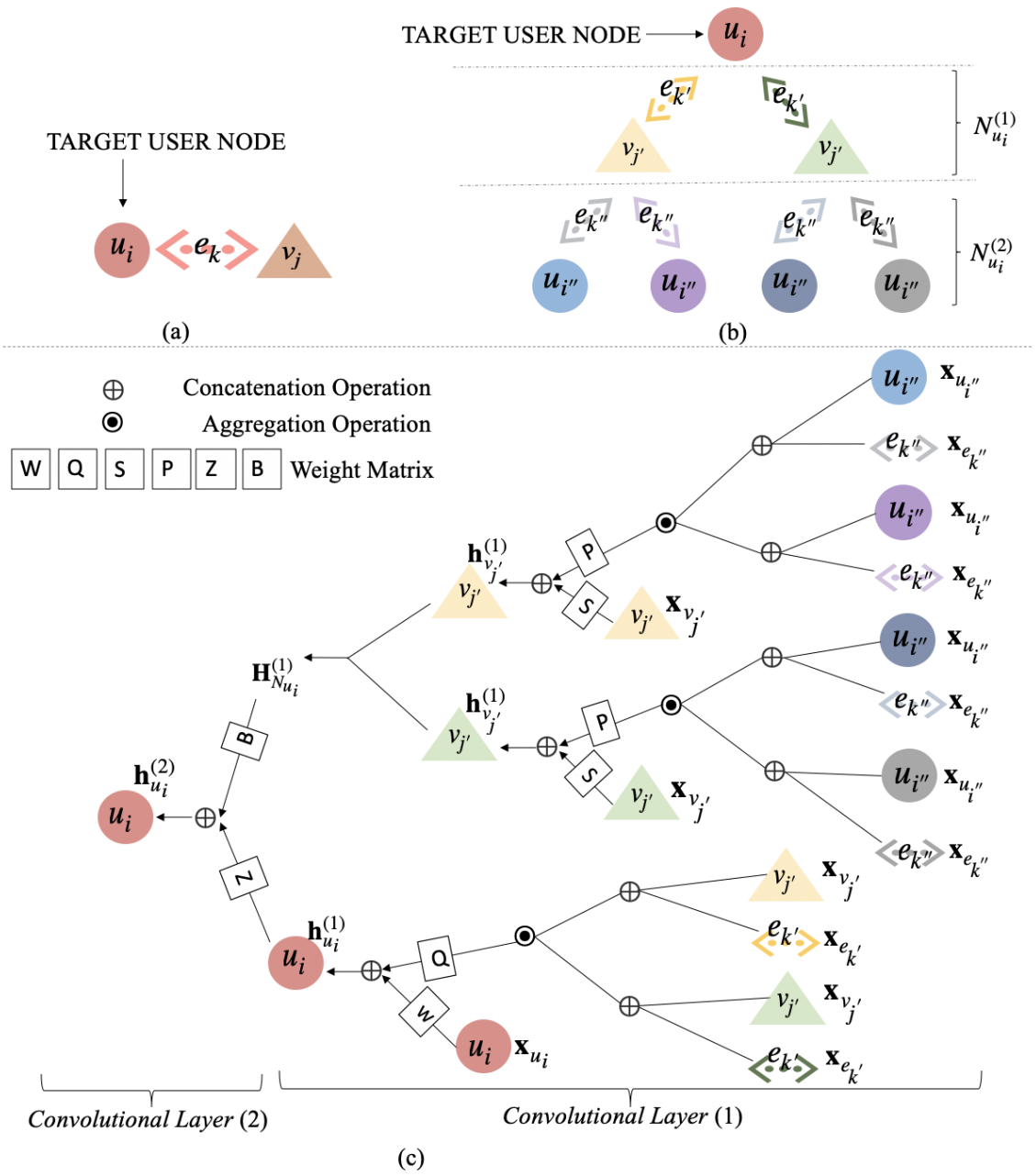

Fig. 6. The convolution process of user u_i in subpart-2 of CGNN. (a) the diagram about sampling an edge e_k formed by user u_i and item v_j; (b) the diagram about sampling 1-depth neighbors $N_{u_i}^{(1)}$ and 2-depth neighbors $N_{u_i}^{(2)}$ for user u_i; (c) the diagram about how to learn the final embedding $\mathbf{h}_{u_i}^{(2)}$ of user u_i by two convolution layers in subpart-2.

and

$$R_{out}(\mathbf{x}_{e_r^{un}}) = D_{out}\left(Concatenate\left(\mathbf{h}_{u_i}^{(2)}, \mathbf{h}_{v_j}^{(2)}, \mathbf{h}_{e_r^{un}}\right)\right), \tag{2}$$

respectively, where "*Concatenate*" denotes the concatenation operation. As aforementioned, we take some normal edges as weakly supervised information, which only can be reconstructed by the *inlier decoder* D_{in}, the reconstruction error $R_{in}(\mathbf{x}_{e_l^{no}})$ of the normal edge e_l^{no} is:

$$R_{in}(\mathbf{x}_{e_l^{no}}) = D_{in}\left(Concatenate\left(\mathbf{h}_{u_i}^{(2)}, \mathbf{h}_{v_j}^{(2)}, \mathbf{h}_{e_l^{no}}\right)\right). \tag{3}$$

Finally, the reconstruction loss $\mathcal{L}_1$ for normal edges and the reconstruction loss $\mathcal{L}_2$ for unlabeled edges are defined as:

$$\mathcal{L}_1 = \sum_{l=1}^{b}\left(\left\|\mathbf{x}_{e_l^{no}} - R_{in}\left(\mathbf{x}_{e_l^{no}}\right)\right\|_2^2\right), \tag{4}$$

and

$$\mathcal{L}_2 = \sum_{r=1}^{d}\left((1-y_r)\left\|\boldsymbol{x}_{e_r^{un}} - R_{in}\left(\boldsymbol{x}_{e_r^{un}}\right)\right\|_2^2 + y_r\left\|\boldsymbol{x}_{e_r^{un}} - R_{out}\left(\boldsymbol{x}_{e_r^{un}}\right)\right\|_2^2\right), \tag{5}$$

respectively, where b and d are the numbers of normal and unlabeled edges that have been sampled in a batch, respectively. b is 0 if normal edges are not sampled by the data processor into the batch. For an unlabeled edge e_r^{un}, y_r=1 if,

$$\left\|\mathbf{x}_{e_r^{un}} - R_{in}\left(\mathbf{x}_{e_r^{un}}\right)\right\|_2^2 > \left\|\mathbf{x}_{e_r^{un}} - R_{out}\left(\mathbf{x}_{e_r^{un}}\right)\right\|_2^2, \tag{6}$$

otherwise, 0. The loss function $\mathcal{L}_{MAIN}$ is different for each batch as we do not restrict the type of sampled edge e_k (normal or unlabeled) in the process of edge sampling. In each batch, if normal edges are not sampled the loss $\mathcal{L}_{MAIN}$ is $\mathcal{L}_2$, while if normal edges are sampled, the loss $\mathcal{L}_{MAIN}$ is:

$$\mathcal{L}_{MAIN} = \mathcal{L}_1 + \mathcal{L}_2. \tag{7}$$

4.4 Fraud Detector: Mutual Information Regularization

Graphical Mutual Information (GMI) has achieved some empirical success in learning node representations within graph-structured data in an unsupervised manner [33, 37, 43, 56]. These methods follow the intuitions from Deep InfoMax [15] which supports the idea that preserving as much information as possible from the input space into the embedding space can facilitate learning high-quality representation of inputs. To preserve enough information from inputs, Deep InfoMax trains an encoder to maximize the mutual information between inputs (i.e., images) and outputs (i.e., the hidden vectors), which can be implemented by maximizing the distance between the positive samples (i.e., original inputs) and negative samples (i.e., modified inputs) [43, 56] in the embedding space. Inspired by this idea, we can separate the normal and fraud behaviors in the embedding spaces to an extreme by maximizing the mutual information between fraud behaviors and their hidden vectors (i.e., embedding) under the assumption that normal behaviors are negative samples and fraud behaviors are positive samples. It will facilitate CGNN to further separate the normal and fraud behaviors on the embedding space. We will discuss how to apply the mutual information regularization on CGNN next in detail.

After each iteration, eFraudCom assigns an unlabeled edge to the *inlier decoder* D_{in} or the *outlier decoder* D_{out}. As aforementioned, the computation of mutual information in the neural network is based on the comparison between positive and negative samples. We denote the edges assigned to the *outlier decoder* as positive samples, whereas other

Algorithm 2: CGNN

Input: The heterogeneous network $G = \{U, V, E^{no}, E^{un}, \mathbf{X}_U, \mathbf{X}_V, \mathbf{X}_{E_{no}}, \mathbf{X}_{E_{un}}\}$;
The weight matrices and bias;
The hyper-parameters λ and N (i.e., the number of 1-depth neighbors sampled for each node).
Output: The label y_r of unlabeled edge $e_r^{un}, \forall e_r^{un} \in E^{un}$.

Initial: The weight matrices and bias, hyper-parameter λ, and hyper-parameter N.
for *epoch* $\leftarrow$ *1, 2, ...* **do**
  for *batch* $\leftarrow$ *1, 2, ...* **do**
    Sampling the 1-depth and 2-depth neighbors for users and items related to edges in the batch.
    Learning the embeddings $\mathbf{h}_{e_k}$ for edges in the batch by dense layers.
    Learning the embeddings $\mathbf{h}_{u_i}^{(2)}$ and $\mathbf{h}_{v_j}^{(2)}$ for all user and item pairs related to edges in the batch (**Algorithm 1**).
    for *edge* e_k *in the batch* **do**
      if *edge* e_k *is normal* **then**
        $R_{in}(\mathbf{x}_{e_l^{no}}) = D_{in}\left(Concatenate\left(\mathbf{h}_{u_i}^{(2)}, \mathbf{h}_{v_j}^{(2)}, \mathbf{h}_{e_l^{no}}\right)\right)$ (Eq. (3))
      else
        $R_{in}(\mathbf{x}_{e_r^{un}}) = D_{in}\left(Concatenate\left(\mathbf{h}_{u_i}^{(2)}, \mathbf{h}_{v_j}^{(2)}, \mathbf{h}_{e_r^{un}}\right)\right)$ (Eq. (1))
        $R_{out}(\mathbf{x}_{e_r^{un}}) = D_{out}\left(Concatenate\left(\mathbf{h}_{u_i}^{(2)}, \mathbf{h}_{v_j}^{(2)}, \mathbf{h}_{e_r^{un}}\right)\right)$ (Eq. (2))
        if $\left\|\mathbf{x}_{e_r^{un}} - R_{in}\left(\mathbf{x}_{e_r^{un}}\right)\right\|_2^2 > \left\|\mathbf{x}_{e_r^{un}} - R_{out}\left(\mathbf{x}_{e_r^{un}}\right)\right\|_2^2$, **then**
          setting the binary label y_r of the unlabeled edge e_r^{un} to 1.
        else
          setting the binary label y_r of the unlabeled edge e_r^{un} to 0.
    Computing loss $\mathcal{L}_{MAIN}$ by Eq. (7) and loss $\mathcal{L}_{MI}$ by Eq. (8).
    Computing loss $\mathcal{L}_{CGNN}$ by Eq. (12).
    Backpropagation and update model parameters.

unlabeled edges assigned to the *inlier decoder* and the normal edges in E^{no} are regarded as negative samples. The maximum mutual information between the fraud behaviors and their embedding can be acquired by minimizing the distance between the inputted fraud behaviors and the embedding of fraud behaviors and maximizing the distance between the inputted fraud behaviors and the embedding of normal behaviors [15], the formulation representations are:

$$\mathcal{L}_{MI} = \frac{1}{b+d}\left(\mathcal{L}_3 + \mathcal{L}_4\right), \tag{8}$$

where

$$\mathcal{L}_3 = \sum_{r=1}^{d} y_r E_{\mathbb{P}}\left[\log\left(\mathcal{D}\left(\mathbf{x}_{e_r^{un}}, \mathbf{z}\right)\right)\right], \tag{9}$$

and

$$\mathcal{L}_4 = \sum_{r=1}^{d}(1-y_r)E_{\mathbb{P}}\left[\log\left(1-\mathcal{D}\left(\mathbf{x}_{e_r^{un}}, \mathbf{z}\right)\right)\right] + \sum_{l=1}^{b} E_{\mathbb{P}'}\left[\log\left(1-\mathcal{D}\left(\mathbf{x}_{e_l^{no}}, \mathbf{z}\right)\right)\right], \tag{10}$$

where $\mathbf{z} = \sigma\left(\frac{1}{d_1}\sum_{r=1}^{d} y_r \cdot Concat\left(\mathbf{h}_{u_i}^{(2)}, \mathbf{h}_{v_j}^{(2)}, \mathbf{h}_{e_r^{un}}\right)\right)$, σ is the sigmoid nonlinearity and $\mathcal{D} : D \times D' \rightarrow \mathbb{R}$ denotes a simple bilinear function [31]:

$$\mathcal{D}\left(\mathbf{x}_{e_r^{un}}, \mathbf{z}\right) = \sigma\left(\mathbf{x}_{e_r^{un}} \mathbf{\Theta} \mathbf{z}\right), \tag{11}$$

where $\mathbf{\Theta}$ represents a trainable matrix and σ is the activation function sigmoid. Finally, CGNN will be optimized by minimizing the following loss function:

$$\mathcal{L}_{CGNN} = \mathcal{L}_{MAIN} - \lambda \mathcal{L}_{MI}, \tag{12}$$

where λ>0 is a parameter that controls the relative importance of the mutual information regularization term. We observe that λ=0.1 produces promising results. The detailed training process is shown in **Algorithm 2**.

Furthermore, we use the following formula to get the risk score of an unlabeled edge e_r^{un} assigned to the *outlier decoder*:

$$risk\ score(e_r^{un}) = \frac{\left\| \boldsymbol{x}_{e_r^{un}} - R_{in}\left(\boldsymbol{x}_{e_r^{un}}\right) \right\|_2^2}{\left\| \boldsymbol{x}_{e_r^{un}} - R_{in}\left(\boldsymbol{x}_{e_r^{un}}\right) \right\|_2^2 + \left\| \boldsymbol{x}_{e_r^{un}} - R_{out}\left(\boldsymbol{x}_{e_r^{un}}\right) \right\|_2^2}. \tag{13}$$

4.5 Time Complexity

We uniformly sample a fixed-size set of neighbors before running CGNN, with this sampling, the time complexity of CGNN is composed of two parts. In the first stage, graph encoding, the main time complexity comes from the embedding learning of users and items. The time complexity of this stage is fixed at $O\left(\prod_{i=1}^{K} L_i\right)$, where $i \in 1, ..., K$, K depth nodes as neighborhoods and aggregate their features to obtain user or item node embedded representation, and L_i is the fixed-number of neighbors sampled in the i-th layer. In the second stage, graph decoding, the time complexity is $O\left(\prod_{h=1}^{H} C_{h-1} C_h\right)$, where H is the depth of decoder network, C_{h-1} and C_h are the number of input and output channels, the time complexity of the second stage can be regarded as constant complexity $O(n)$. To sum up, the overall time complexity of our model is $O\left(\prod_{i=1}^{K} L_i\right) + O(n)$, after ignoring constant, is $O\left(\prod_{i=1}^{K} L_i\right)$.

4.6 Implementation

According to the design, we implement eFraudCom as a prototype on MaxCompute provided by Alibaba. Specifically, we implement the data processor in Python using MaxCompute SQL. The data is stored in a distributed file system and MapReduce is supported for computing and processing. Fraud detector, the core part of eFraudCom, based on distributed PyTorch and GraphScope, will be implemented in a parallelized style for fast processing.

5 EXPERIMENTS

We evaluate the performance of our proposed system eFraudCom on four datasets, two Taobao and two public datasets [6]. In the experiment section, we mainly present:

- Introducing the experimental setup, such as datasets, evaluation metrics, baselines (Sections 5.1, 5.2, and 5.3).
- Evaluating the proposed system eFraudCom on two Taobao datasets by comparing the performance with baselines (Section 5.4).
- Evaluating the proposed system eFraudCom on two public datasets by comparing the performance with baselines (Section 5.5).
- Utilizing ablation experiments to prove the importance of individual components in CGNN, followed by the parameter analysis (Section 5.6).

[6]The code and two public datasets are available at https://github.com/GeZhangMQ/eFraudCom

- Utilizing a case study on two Taobao datasets to prove the capacity of eFraudCom in adapting to the fraud pattern upgraded by third party fraud platforms (Section 5.7).

To run baselines on Taobao datasets, we download the data collected by the data processor as offline data.

Table 1. Detailed information of two taobao datasets D_1 and D_2, and two public datasets MOOC student drop-out and Bitcoin-Alpha. The statistical information of D_1, D_2, and MOOC student drop-out are shown in (a), and the statistical information of Bitcoin-Alpha is shown in (b).

(a) The statistical information of D_1, D_2, and MOOC student drop-out. Edges are the connections between users and items. We do the fraud behaviors detection task on these three datasets. As shown in the table, 21.86% and 21.79% of edges in D_1 and D_2 are labeled by domain experts, and all edges are labeled in MOOC student drop-out. We use "Normal Edges" and "Fraud Edges" to denote edges unrelated or related to fraud behaviors, respectively. In datasets D_1 and D_2, the number of attributes of edges, items, and users is 19, 16, and 20, respectively. In MOOC student drop-out, the number of attributes of edges, items, and users is 4, 4, and 27, respectively.

Data	#Edges	#Labeled Edges(%)	Labeled Edges		#Users	#Items
			#Normal Edges	#Fraud Edges		
D_1	1,087,828	21.86%	193,296	44,455	896,462	249,578
D_2	1,084,622	21.79%	192,466	43,855	896,462	249,578
MOOC student drop-out	10,022	100%	8,011	2,011	5,034	97

(b) The statistical information of Bitcoin-Alpha. The network is made bipartite by splitting each user into the user set with all its outgoing edges and the item set with all incoming edges, each edge indicates a rating from one user to another with a rating score. We do the fraud user detection task on it due to only users have attributes and labels. Then, we transform the graph from a heterogeneous graph between users and items to a homogeneous graph among users. Edges in the table are built by connecting users who rate the same user (i.e., item) according to the original Bitcoin-Alpha bipartite graph. As shown in the table, only 6% of users in Bitcoin-Alpha are labeled. We use "Normal Users" and "Fraud Users" to denote users in Bitcoin-Alpha unrelated or related fraud behaviors, respectively. In Bitcoin-Alpha, the number of attributes of users is 148.

Data	#Users	#Labeled Users(%)	Labeled Users		#Edges	#Items
			#Normal Users	#Fraud Users		
Bitcoin-Alpha	3,275	6%	131	83	484,543	3,742

5.1 Datasets

The four datasets are introduced in detail next:

- **Taobao datasets:** The statistical information of these two Taobao datasets named D_1 and D_2 is shown in Table 1 (a). Specifically, edges in dataset D_1 are generated by users' transaction behavior on items, and edges in dataset D_2 are generated by users' add-to-favorite behavior on items. We solemnly declare that these two datasets are used for evaluating the performance of eFraudCom, and any derived information (e.g. the fraud-normal ratio) does not represent the true scenario of Alibaba.

  In dataset D_1, the attributes of an edge generated by the user's transaction behavior on an item composed of the number of total transactions among the user and the item, the average transaction prices and average purchase time interval, and the variance of the purchase time interval, and the browsing time before placing the order, etc. In dataset D_2, the attributes of an edge generated by the user's add-to-favorite behavior on an item come from the number of totals "like", browsing, and average browsing time, and the variance of browsing time, etc. In datasets D_1 and D_2, the attributes of a user from the number of transactions completed by the user, the average and variance of transaction prices and browsing time before each transaction behavior, etc. The attributes of an item generated by the total sales, the number of transaction prices, the average and variance of selling prices, and the conversion rate (i.e., $purchases/clicks \times 100\%$), etc.

- **MOOC student drop-out:** The statistical information of MOOC student drop-out is shown in Table 1 (a). It is a public dataset consist of users, items, and actions taken by users on items, e.g., viewing a video, submitting an answer, etc., done by Students on a MOOC online course[18]. This dataset consists of 5,034 users interacting with 97 items (videos, answers, etc.) resulting in 10,022 interactions. There are 2,011 drop-out events. In this work, we regard the actions related to drop-out behavior as fraud actions and others as normal actions.

  The original dataset only contains the attributes of actions. We set the attribute vector for each user and item according to their properties. The attribute vector of a user node concatenated by there parts: (i) a one-hot vector indicating the degree of the user node; (ii) the average of attribute vectors of actions completed by the user; (iii) the summation of one-hot vectors where each one-hot vector represents a time interval between two consecutive actions taken by the user. The attribute vector of an item is the average of attribute vectors of actions related to it.
- **Bitcoin-Alpha:** The statistical information of Bitcoin-Alpha is shown in Table 1 (b). It is a Bitcoin trust network in the Alpha platform [17]. The network is made bipartite by splitting each user into the user set with all its outgoing edges and the item set with all incoming edges, each edge indicates a rating from one user to another with a rating score. The network has 3,275 user nodes and 3,742 item nodes, 214 users have labels, of which 83 and 131 are labeled as fraudulent users and normal users, respectively. The ground truth is defined: normal users are the platform's founder and users who rated highly positively (≥ 0.5). Fraudulent users are the ones that these trusted users uniformly rate negatively (≤ -0.5).

  The attribute vector of each user is concatenated by three parts: (i) a one-hot vector indicating the degree of the user node; (ii) the summation of one-hot vectors that each represents rating score given by this user; (iii) the summation of one-hot vectors where each hot represents a time interval between two consecutive ratings. The dimension of the attribute vector of each user is 148.

  As the Bitcoin-Alpha dataset only has the labels and attributes vectors of users, we can only do anomalous user nodes detection task on it, while CGNN proposed for detecting edges related to fraud behaviors. To make CGNN can detect anomalous user nodes on the Bitcoin-Alpha dataset, we modify both the Bitcoin-Alpha dataset and CGNN. Specifically, 1) for the Bitcoin-Alpha dataset, we build a homogeneous graph among users by connecting users who rate the same user (i.e., item) according to the original Bitcoin-Alpha bipartite graph; 2) for CGNN, the two subparts (i.e., subpart-1 and subpart-2) in CGNN are replaced by the graph convolution layers proposed in GraphSAGE [14].

  We use the Bitcoin-Alpha dataset to verify that the competitive decoder structure proposed by this work can also perform well when detecting anomaly nodes on homogeneous graphs.

Table 2. Evaluation Metrics. tp: true positives; tn:true negatives; fp: false positives; fn: false negatives.

Evaluation Metric	**Formula**
AUC value	The area under the ROC curve.
Precision@K	$\frac{tp}{tp+fp}$, tp and fp calculated from the edges with top K risk scores.
Recall@K	$\frac{tp}{tp+fn}$, tp calculated from the edges with top K risk scores.
#Recall@90%	the value of K, when Recall reaches 90% (i.e., $\frac{tp}{tp+fn} = 0.9$).
Accuracy	$\frac{tpr+tnr}{2}$, where $tpr = \frac{tp}{tp+fn}$, $tnr = \frac{tn}{fp+tn}$

5.2 Evaluation Metrics

In this section, we will discuss how to evaluate the performance of eFraudCom and baselines on two Taobao datasets in detail. As shown in Table 1 (a), only about 22% edges in two Taobao datasets are labeled. For two Taobao datasets, the performance of all baselines and eFraudCom on the labeled part can be evaluated by public metrics automatically, and the performance on the unlabeled part needs to be evaluated by domain experts in Alibaba.

For edges with labels in Taobao datasets: Four evaluation metrics, ROC-AUC, Precision@K, Recall@K, and #Recall@90% are used to measure the performance on the labeled part of two Taobao datasets. Their formulation is presented in Table 2, the fraud class and the normal classes are regarded as the positive class and the negative class, respectively.

- **ROC-AUC:** The **ROC curve** is a plot of true positive rate against false positive rate according to the ground truth and the detection results. **AUC value** is the area under the ROC curve, representing the probability that a randomly chosen fraud edge is ranked higher than a normal edge. If the AUC value approaches 1, the method is of high quality.
- **Precision@K:** As each detection algorithm outputs a ranking list of risk scores, we regard the proportion of true positive edges that a specific detection method discovered in the edges with top K risk scores as Precision@K. The larger the value of Precision@K, the better the algorithm is.
- **Recall@K:** We regard the proportion of the number of true positive edges with top K risk scores that a specific detection method discovered in the number of ground truth edges related to fraud behaviors as Recall@K. The larger the value of Recall@K, the better the algorithm is.
- **#Recall@90%:** We use #Recall@90% to denote when the recall reaches 90%, what is the value of K need to be set. The smaller the value of # Recall@90%, the better the system is.

For edges without labels in Taobao datasets: For the unlabeled parts in two Taobao datasets, we could not evaluate the performance of detection methods with the above four public metrics due to the lack of ground truth. The results on unlabeled parts will be submitted to Alibaba and evaluated by domain experts. Domain experts will select the top 30,000 edges from the unlabeled data according to the risk score sorted in descending order, and then labeled these edges. Compared with labels predicted by eFraudCom, the evaluation is made from the perspective of **Accuracy**. The formulation of "Accuracy" is also shown in Table 2.

5.3 Baselines

The baselines include:

- Isolation Forest (IF) [24]: It is an anomaly detection algorithm based on the assumption that the anomaly data is always rare and far from the center of normal clusters. The idea of the algorithm based on the binary tree structures is building an ensemble of series of iTrees for a given data set through random sampling. The main idea of isolation tree is to take advantage of anomalous instances being "few and different".
- AE [12]: It includes a conventional autoencoder framework composed of fully-connect neural networks. AE models the attribute vectors for embedding learning, and the anomaly detection problem is solved by a deep decoder that leveraging the learned embeddings to reconstruct the original data, instances with high reconstruction errors regarded as anomalies.
- AE(2): In AE(2), we use three different fully-connected neural networks to encode edges, corresponding users, and items, respectively. Then, the concatenation results of the above three embedding parts will be used to

reconstruct the attributes of edges by the decoder part composed of fully connected layers, instances with high reconstruction errors regarded as anomalies.

- DAGMM [59]: It utilizes a deep autoencoder to generate a low-dimensional representation and reconstruction errors for inputs. Then, the low-dimensional representation and reconstruction errors of samples will be fed into a Gaussian Mixture Model (GMM). The samples located in low-density areas will be regarded as anomalies.
- Deep SVDD [38]: It is inspired by kernel-based one-classification and minimum volume estimation. Deep SVDD trains a neural network while minimizing the volume of a hypersphere that encloses the network representation of data. Data that deviates from the hypersphere considered anomalies.
- DeepFD [46]: It is an encoder-decoder structured deep neural network model for fraud detection. It learns embeddings by minimizing the difference of pairwise distance between embeddings and a modified Jaccard similarity. In this method, only is the structure information of the heterogeneous graph to be used.
- DOMINANT [10]: It compresses the input attributed network to low-dimensional embedding representations using the graph convolution structure proposed in [16], then the network topology and nodal attributes reconstructed by corresponding decoders, respectively. The reconstruction errors of nodes following the encoder and decoder phases are then leveraged for spotting anomalous nodes on attributed networks.

  DOMINANT is designed to detect anomalous nodes in the homogeneous graph originally, so we cannot run it on Taobao and MOOC datasets directly. To run DOMINANT on two Taobao and MOOC datasets, we change the heterogeneous graph among users and items into the homogeneous graph among users by connecting users related to the same items.
- eFraudCom: the proposed system.

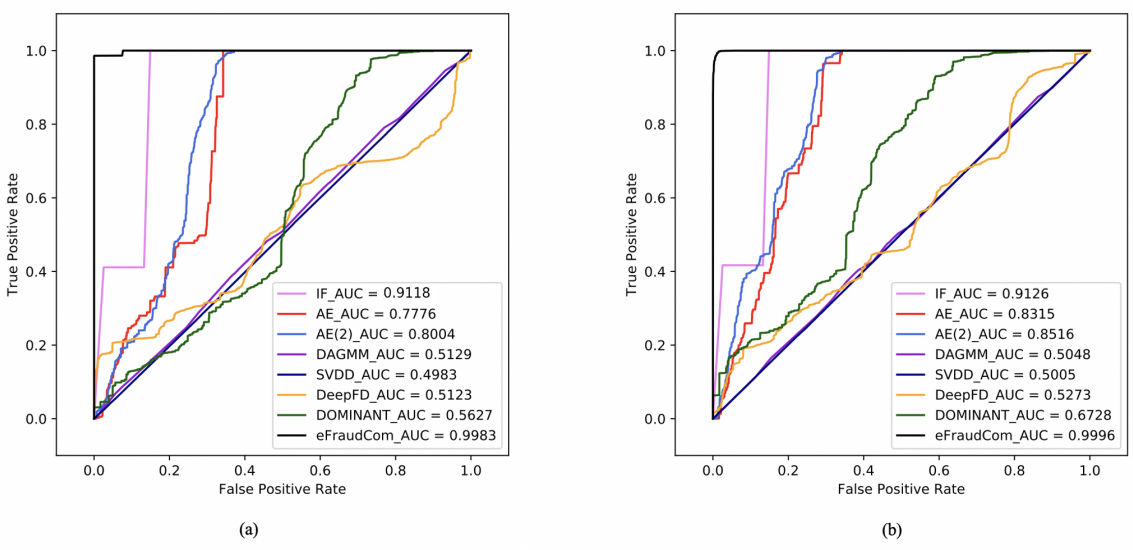

Fig. 7. (a) and (b) show the ROC curves and AUC scores of all baseline methods and eFraudCom on two taobao datasets D_1 and D_2, respectively.

Table 3. Performance comparison on the labeled part of the Taobao dataset D_1 w.r.t. #Recall@90%, Precision@K and Recall@K, which are shortened as #R@90% , P@K and R@K, respectively.

Method	#R@90%	P@20,000	P@25,000	P@30,000	R@20,000	R@25,000	R@30,000
IF	68,211	0.7876	0.7313	0.6094	0.3542	0.4112	0.4112
AE	106,212	0.3524	0.3211	0.3028	0.1585	0.1806	0.2043
AE(2)	100,332	0.3660	0.3437	0.3140	0.1646	0.1932	0.2119
DAGMM	211,950	0.2076	0.2001	0.1980	0.0934	0.1128	0.1336
SVDD	214,123	0.1816	0.1826	0.1830	0.0817	0.1027	0.1235
DeepFD	225,212	0.4598	0.3729	0.3210	0.2068	0.2097	0.2167
DOMINANT	172,371	0.2300	0.2187	0.1891	0.1035	0.1230	0.1276
eFraudCom	**41,961**	**0.9911**	**0.9913**	**0.9878**	**0.4459**	**0.5069**	**0.6666**

Table 4. Performance comparison on the labeled part of the Taobao dataset D_2 w.r.t. #Recall@90%, Precision@K and Recall@K, which are shortened as #R@90% , P@K and R@K, respectively.

Method	#R@90%	P@20,000	P@25,000	P@30,000	R@20,000	R@25,000	R@30,000
IF	67,687	0.7865	0.7313	0.6094	0.3587	0.4196	0.4196
AE	95,455	0.4003	0.3878	0.3787	0.1826	0.2211	0.2591
AE(2)	92,394	0.4774	0.4950	0.5091	0.2177	0.2882	0.3483
DAGMM	213,876	0.1851	0.1858	0.1894	0.0844	0.1059	0.1295
SVDD	213,461	0.1897	0.1881	0.1871	0.0864	0.1072	0.1280
DeepFD	197,860	0.3337	0.3376	0.2878	0.1522	0.1925	0.1969
DOMINANT	152,324	0.3879	0.3423	0.3145	0.1769	0.1952	0.2152
eFraudCom	**39,862**	**0.9930**	**0.9956**	**0.9903**	**0.4548**	**0.5676**	**0.6775**

5.4 Performance on Taobao Datasets

Due to the different evaluation methods used in the labeled and unlabeled parts of Taobao datasets, we discuss the performance of eFraudCom on the labeled and unlabeled parts separately.

The performance on labeled parts: We verify the effectiveness of our proposed system eFraudCom on the labeled part of Taobao datasets by comparing it with the seven baselines mentioned above. The experimental results in terms of ROC curve and AUC value on two Taobao datasets D_1 and D_2 are shown in Fig. 7 (a) and (b), respectively. The performance concerning Precision@K, Recall@K, and #Recall@90% on D_1 and D_2 are shown in Table 3 and Table 4, respectively.

As shown in Table 3 and Table 4, we set three different values for K: 20,000, 25,000, and 30,000. On the fraud detection task under e-commerce scenarios, domain experts usually pay special attention to the edges with high-risk scores, and the precision among these edges is expected to reach a very high level (i.e., $\geq 98\%$). The reason is that: if an edge is classified by the fraud detection system as fraud, the user and item related to the edge are faced to be punished by e-commerce platforms. However, in the real business scenario, only those edges classified as fraud class with very high-risk scores, the corresponding user and item will be punished due to the cost of mis-punishing normal users and items is prohibitive. Based on it, we set relatively high values for K to evaluate whether fraud detection algorithms can reach high precision among edges with high-risk scores and ensure the minimum mis-punishment rate. As shown in Table 3 and Table 4, only the proposed fraud detection system eFraudCom achieves the requirement that precision among edges with relatively high-risk scores is not less than 98%.

From the evaluation results shown in Fig. 7, Table 3 and Table 4, it is noted that the superiority of our proposed system eFraudCom in detecting fraud behaviors:

- As shown in Fig. 7 (a), eFraudCom prevails over the baseline methods IF, AE, AE(2), DAGMM, SVDD, DeepFD, DOMINANT by 8.65%, 22.07%, 19.39%, 48.54%, 50.03%, 48.6%, 43.56% w.r.t. AUC value on Taobao dateset D_1, respectively. As shown in Fig. 7 (b), on dataset D_2, the prpoposed system eFraudCom still shows the similar superiority.
- As shown in Table 3 and Table 4, eFraudCom also shows the superiority on Precision@K and Recall@K for all K values.
- As shown in Table 3 and Table 4, when the recall rate reaches 90%, our proposed system eFraudCom only needs to set K as 41,961 and 39,862 on two Taobao datasets D_1 and D_2, respectively. Compared with other baseline methods, eFraudCom has superiority in recall capacity. Even though baseline method IF and AE(2) have achieved competitive results w.r.t. ROC-AUC, IF needs to set K as 68,211 and 67,687 when the recall rate reaches 90% on two Taobao datasets D_1 and D_2, respectively, and AE(2) needs to set K as 100,332 and 92,394 when the recall rate reaches 90% on D_1 and D_2, respectively.

There are also some other experimental results worthy of further analysis:

- As a GCN-based algorithm, DOMINANT failed to outperform the conventional methods such as IF and AE, the reason may be that: firstly, DOMINANT utilizes GCN directly without any modification. In each convolution layer, it aggregates information from all the immediate neighbors when learn embedding for a target node, but such a smoothing operation is not in line with anomaly detection. The principle is that aggregating all neighborhood attributes to an anomalous node will probably smooth out the suspiciousness of it due to it may have normal neighbors; Secondly, as we mentioned in section 5.3, DOMINANT is designed to detect anomalous nodes on homogeneous graph originally. To run DOMINANT on Taobao datasets, we have to transform the heterogeneous graph into a homogeneous graph among users. DOMINANT detects the edges related to fraud behaviors in an indirect way, which may be the other reason.
- As a conventional method, one of the baselines IF outperforms most other baselines on two Taobao datasets. The reason is that as a tree-based algorithm, IF has strong anti-noise capacity due to two randomness: the first is that each tree randomly selects different samples for processing, the second is the random feature selection of each tree. On two Taobao datasets, except for the small number of representative samples detected by domain experts, all of the other data stored by MaxCompute Platform automatically, noise is inevitable.

The performance on unlabeled parts: The performance of IF, AE(2), and eFraudCom on unlabeled parts of dataset D_1 and D_2 w.r.t.Accuracy evaluated by domain experts in Alibaba are shown in Table 5. The reason for only evaluating the performance of IF and AE(2) on the unlabeled part of datasets is that these two algorithms have relatively better performance on labeled parts of D_1 and D_2 compared with other baseline methods.

Table 5. Performance comparison on unlabeled parts of D_1 and D_2 w.r.t Accuracy.

	IF	AE(2)	eFraudCom
D_1	0.6779	0.6684	0.7023
D_2	0.6869	0.6583	0.7353

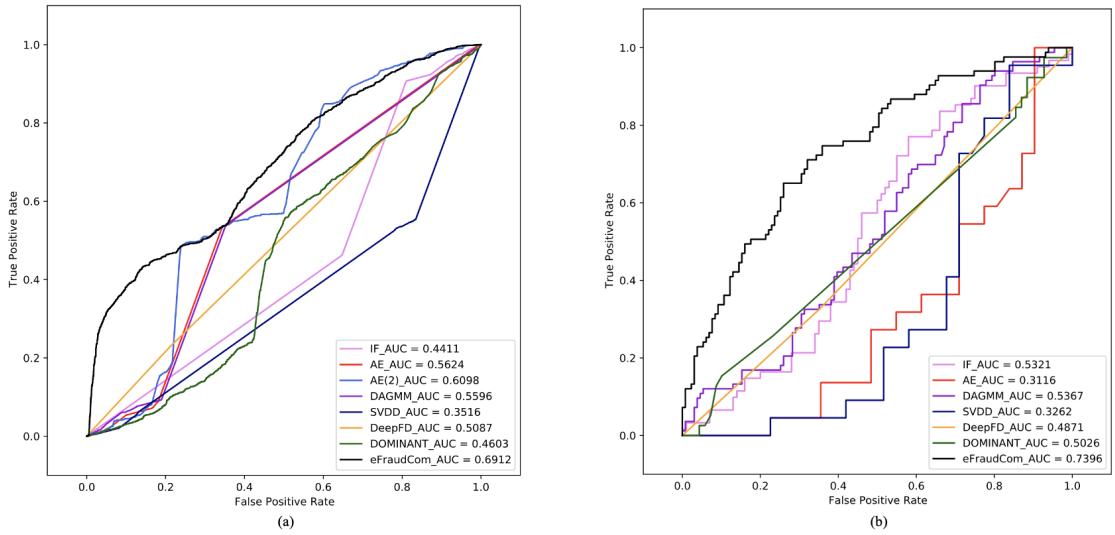

Fig. 8. (a) and (b) show the ROC curves and AUC scores of all baseline methods and eFraudCom on two public datasets MOOC student drop-out and Bitcoin-Alpha, respectively.

5.5 Performance on Public Datasets

We also evaluate the performance of the proposed system eFraudCom on these two public datasets, MOOC student drop-out, and Bitcoin-Alpha. The performance of all methods on MOOC student drop-out and Bitcoin-Alpha in terms of the ROC Curve and AUC value is shown in Fig. 8 (a) and (b), respectively.

As shown in Fig. 8 (a) and Fig. 8 (b), eFraudCom still achieves the best performance on these two public datasets. The performance of GNN-based anomaly detection algorithm DOMINANT still failed to outperform most of the baselines, even on dataset Bitcoin-Alpha that allows DOMINANT to detect anomalous nodes directly. The experiment result confirms our assumption in section 5.4 that using the GCN structure which aggregates all neighbors to the target node will lead to over-smooth and reduce the suspiciousness of fraud behaviors. Additionally, as shown in Fig. 8 (b), algorithm AE cannot perform as well as it did on the other three datasets, the reason is that AE does not rely on any information from the neighborhood when learning the embedding for target users on Bitcoin-Alpha, extremely sparse information makes it unable to fully learn the characteristics of users. We do not show the performance of AE(2) on Bitcoin-Alpha in Fig. 8 (b) due to the dataset only has the attributes of users, while the input of AE(2) must contain attributes of edges.

Through the experimental results on two Taobao and two public datasets, we can conclude that our system eFraudCom has superiority in detecting fraud behaviors.

5.6 Ablation Experiments and Parameter Analysis

5.6.1 Ablation Experiments. To prove the importance of individual components (i.e., the two convolution layers in subpart-2, the competitive graph decoder structure, and the mutual information regularizer) in CGNN of the eFraudCom system, we utilize the baseline methods AE and AE(2) and two different variants of eFraudCom to do ablation experiments on two Taobao datasets D_1 and D_2.

As mentioned in section 5.3, algorithm AE only utilizes the attribute vectors of edges to detect edges related to fraud behaviors. In AE(2), three different fully connected neural networks are responsible for encoding edges, users, and items, respectively. Then the concatenation results about latent representations of edges, users, and items are utilized to reconstruct the attributes of edges.

The other two variants of eFraudCom are:

- eFraudCom-a: It is the first variant of CGNN in the eFraudCom system, in which we retain the subpart-1 and subpart-2 in graph encoding, while the dual graph decoders substituted by one graph decoder and the mutual information regularization is not to be utilized. All edges need to be reconstructed by a single graph decoder, and reconstruction errors of edges will be regarded as a risk score.
- eFraudCom-b: It is the second variant of CGNN in the eFraudCom system. Compared with the complete CGNN introduced in section 4, only the mutual information regularization is not to be utilized.

Table 6. Ablation analysis on the labeled part of dataset D_1 w.r.t. AUC, #Recall@90%, Precision@K and Recall@K, #Recall@90%, Precision@K and Recall@K are shortened as #R@90% , P@K and R@K, respectively.

Method	AUC	#R@90%	P@20,000	P@25,000	P@30,000	R@20,000	R@25,000	R@30,000
AE	0.7776	106,212	0.3524	0.3211	0.3028	0.1585	0.1806	0.2043
AE(2)	0.8004	100,332	0.3660	0.3437	0.3140	0.1646	0.1932	0.2119
eFraudCom-a	0.8398	96,612	0.4774	0.4754	0.4497	0.2148	0.2674	0.3035
eFraudCom-b	0.9202	66,786	0.9011	0.7323	0.6309	0.4054	0.4118	0.4258
eFraudCom	**0.9983**	**41,961**	**0.9911**	**0.9913**	**0.9878**	**0.4459**	**0.5069**	**0.6666**

Table 7. Ablation analysis on the labeled part of dataset D_2 w.r.t. AUC, #Recall@90%, Precision@K and Recall@K, #Recall@90%, Precision@K and Recall@K are shortened as #R@90% , P@K and R@K, respectively.

Method	AUC	#R@90%	P@20,000	P@25,000	P@30,000	R@20,000	R@25,000	R@30,000
AE	0.8315	95,455	0.4003	0.3878	0.3787	0.1826	0.2211	0.2591
AE(2)	0.8516	92,394	0.4774	0.4950	0.5091	0.2177	0.2882	0.3483
eFraudCom-a	0.9267	65,989	0.6986	0.6482	0.6134	0.3186	0.3696	0.4196
eFraudCom-b	0.9464	60,154	0.9365	0.7521	0.6796	0.4271	0.4287	0.4649
eFraudCom	**0.9996**	**39,862**	**0.9930**	**0.9956**	**0.9903**	**0.4548**	**0.5676**	**0.6775**

The result of ablation analysis on dataset D_1 and D_2 are shown in Table 6 and Table 7, respectively. From the evaluation results, we can make the following conclusion:

- Firstly, comparing the performance of AE and AE(2) on two Taobao datasets, it can be observed that integrating the information of corresponding user and item pairs of edges can improve the performance of fraud detection. The reason is that on e-commerce data, some information especially the fraud information on edges would be reflected in the corresponding user and item pairs.
- Secondly, comparing the performance of AE(2) and eFraudCom-a on two Taobao datasets, it can be observed that utilizing the two convolution layers in subpart-2 to learn the latent representations of users and items can model the users and items better and the performance of fraud detection can be further improved.
- Thirdly, eFraudCom-b performs better than eFraudCom-a, which proves that CGNN with dual graph decoders is effective in modeling the distributions of normal and fraud edges separately.

- Fourthly, the performance of eFraudCom is better than eFraudCom-b, which indicates that mutual information regularization could further improve the effectiveness of CGNN by maximizing the separability of normal and fraud edges in the embedding space.

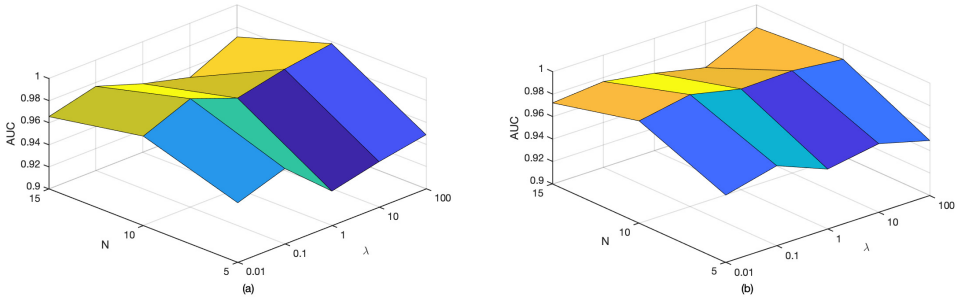

Fig. 9. (a) and (b) show the AUC value achieved by eFraudCom with different λ and N on two Taobao datasets D_1 and D_2, respectively.

5.6.2 Parameter Analysis. On the one hand, as shown in Fig. 9, we conduct the parameter analysis on two Taobao datasets from two perspectives, one is the weight λ of mutual information regularization term, the other one is the sampling number N of 1-depth neighbors (for a single node, no matter is user node or item node, we sample the same and fixed number of 1-depth neighbors for it). As shown in Fig. 9, eFraudCom achieves acceptable results on different parameter settings, when $\lambda = 0.1$ and N=10, eFraudCom obtains the best performance.

On the other hand, we also run CGNN with different convolution layers on the public dataset MOOC student drop-out. As shown in Table 8, we observe that CGNN can achieve the best performance on the model with two convolution layers.

Table 8. AUC value achieved by eFraudCom with different convolution layers on the public dataset MOOC student drop-out.

Layers	1	2	3
AUC	0.6524	0.6912	0.6386

5.7 A Case Study

We did a case study on dataset D_1 to prove the ability of the eFraudCom system in detecting fraud behaviors when the fraud patterns of the third-party fraud service platforms are constantly upgraded. As mentioned above, edges in dataset D_1 are generated by the user's transaction behaviors. The result of the case study is shown in Fig. 10.

In Fig. 10, "base" represents 30,000 transactions sampled from dataset D_1 randomly, we use "base" as the benchmark in this case study. "eFraudCom-a", "eFraudCom-b" and "eFraudCom" represent the transactions with top 30,000 risk scores detected by eFraudCom-a, eFraudCom-b and eFraudCom, respectively. We will discuss the case study from two perspectives:

From the perspective of detecting dishonest users: The analysis result is shown in Fig. 10 (a). On the e-commerce platform, users who had completed fraud transactions in the past are likely to complete fraud transactions again. Even

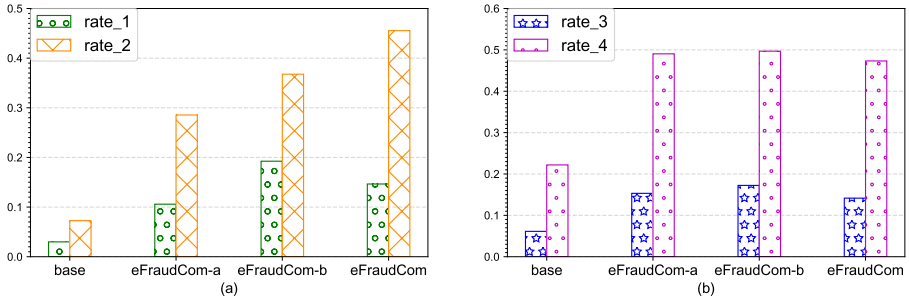

Fig. 10. In (a), "rate_1" represents the percentage of users who have fraud behaviors in the past with the top 30,000 risk scores, and "rate_2" represents the percentage of the users associated with a user who has fraud behaviors in the past with the top 30,000 risk scores. In (b), "rate_3" represents the percentage of merchants who have fraud behaviors in the past with the top 30,000 risk scores, and "rate_4" represents the percentage of the merchants associated with a merchant who has fraud behaviors in the past with the top 30,000 risk scores.

if the fraud patterns have been upgraded, dishonest users employed by the third-party fraud service platforms may be pretty much the same. This is the reason why we introduced "rate_1" in the case study as one of the metrics to evaluate the system's ability to cope with the escalation of fraud patterns. As shown in Fig. 10 (a), our system eFraudCom and the variant eFraudCom-b perform the best on the metric "rate_1" and they outperform the result on "base" with a big margin. It proves that the fraud detection system with dual graph decoders has a strong ability to adapt to the third-party platform's fraud pattern mutation.

"rate_ 2" can be used as a measure of the detection ability of fraud detection systems in a concrete type of fraud pattern mutation. The fraud upgrading strategy investigated by this case study is that third-party fraud service platforms will limit the number of fraud transactions completed by a single buyer account on the Taobao platform. To earn enough commission, dishonest users have to register different accounts to continue to complete other fraud transactions. Different accounts that use the same ID card number, name of the receiver, mobile number of receiver, or address will be regarded as associated accounts. In "rate_ 2", eFraudCom and eFraudCom-b have gained great advantages, especially eFraudCom. It again proves that our system design has a great advantage in detecting the fraud pattern mutation manipulated by changing buyer accounts frequently.

From the perspective of detecting dishonest merchants: The analysis result is demonstrated in Fig. 10 (b). Merchants who used to turn to the third-party fraud service platform to complete fraud transactions in the past are likely to do it again. As shown in Fig. 10 (b), we use "rate_3" as the metric to measure the adaptability of our fraud detection system to the fraud patterns mentioned above. In "rate_3", eFraudCom-b has the best result among all methods. Although eFraudCom does not get the first place, it also greatly exceeds the performance of the "base".

Similarly, some dishonest merchants may evade detection by changing the main body of their online store, and continue to use new stores to improve the ranking of target items by fraud transactions. We use "rate_4" to measure the ability of a fraud detection system in detecting such merchants who build a new online store on the Taobao platform to continue their fraud behaviors. As shown in Fig. 10 (b), the CGNN based eFraudCom-b also has the best result w.r.t. "rate_4".

6 CONCLUSION

Existing fraud detection systems applied to the e-commerce industry face the challenge: it is difficult for them to adapt to the constant fraud pattern mutation since they seriously depend on the "fraudster seeds" predefined by domain experts. To address this issue, in this paper, we propose a competitive graph neural networks (CGNN) based fraud detection system eFraudCom, which is composed of a data processor and a fraud detector. The system eliminates the algorithm dependence on fraud behaviors by taking some normal behaviors as weak supervision information and modeling the normal and fraud behaviors separately. In the data processor, we built a heterogeneous graph among users and items according to user behaviors on the Taobao platform. In the fraud detector, we use CGNN which includes dual graph decoders to model the distributions of normal and fraud behaviors in a competitive mechanism. Furthermore, the mutual information regularization term could maximize the separability of the normal and fraud behaviors in the embedding subspace to further improve CGNN. Experiments on datasets from the Taobao platform and public datasets demonstrate the superiority of eFraudCom in detecting fraud behaviors. A case study proves that eFraudCom has a strong ability to cope with the fraud patterns mutation of third-party fraud service platforms.

ACKNOWLEDGMENT

This work was supported by the ARC DECRA Project (No.DE200100964).

REFERENCES

[1] Leman Akoglu, Rishi Chandy, and Christos Faloutsos. 2013. Opinion fraud detection in online reviews by network effects. *ICWSM* 13, 2-11 (2013), 29.

[2] baran Bandyopadhyay, Harsh Kara, Aswin Kannan, and M Narasimha Murty. 2018. Fscnmf: Fusing structure and content via non-negative matrix factorization for embedding information networks. *arXiv preprint arXiv:1804.05313* (2018).

[3] Sambaran Bandyopadhyay, Saley Vishal Vivek, and MN Murty. 2020. Outlier Resistant Unsupervised Deep Architectures for Attributed Network Embedding. In *Proceedings of the 13th International Conference on Web Search and Data Mining*. 25–33.

[4] Tej Paul Bhatla, Vikram Prabhu, and Amit Dua. 2003. Understanding credit card frauds. *Cards business review* 1, 6 (2003), 1–15.

[5] Aleksandar Bojchevski and Stephan Günnemann. 2018. Bayesian robust attributed graph clustering: Joint learning of partial anomalies and group structure. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 32.

[6] Nuno Carneiro, Goncalo Figueira, and Miguel Costa. 2017. A data mining based system for credit-card fraud detection in e-tail. *Decision Support Systems* 95 (2017), 91–101.

[7] Yukuo Cen, Xu Zou, Jianwei Zhang, Hongxia Yang, Jingren Zhou, and Jie Tang. 2019. Representation learning for attributed multiplex heterogeneous network. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. 1358–1368.

[8] Andrea Dal Pozzolo, Olivier Caelen, Yann-Ael Le Borgne, Serge Waterschoot, and Gianluca Bontempi. 2014. Learned lessons in credit card fraud detection from a practitioner perspective. *Expert systems with applications* 41, 10 (2014), 4915–4928.

[9] Kaize Ding, Jundong Li, Nitin Agarwal, and Huan Liu. 2020. Inductive anomaly detection on attributed networks. In *Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence*.

[10] Kaize Ding, Jundong Li, Rohit Bhanushali, and Huan Liu. 2019. Deep anomaly detection on attributed networks. In *Proceedings of the 2019 SIAM International Conference on Data Mining*. SIAM, 594–602.

[11] Haoyi Fan, Fengbin Zhang, and Zuoyong Li. 2020. AnomalyDAE: Dual autoencoder for anomaly detection on attributed networks. In *ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 5685–5689.

[12] Ian Goodfellow, Yoshua Bengio, Aaron Courville, and Yoshua Bengio. 2016. *Deep learning*. Vol. 1. MIT press Cambridge.

[13] Qingyu Guo, Zhao Li, Bo An, Pengrui Hui, Jiaming Huang, Long Zhang, and Mengchen Zhao. 2019. Securing the deep fraud detector in large-scale e-commerce platform via adversarial machine learning approach. In *The World Wide Web Conference*. 616–626.

[14] Will Hamilton, Zhitao Ying, and Jure Leskovec. 2017. Inductive representation learning on large graphs. In *Advances in neural information processing systems*. 1024–1034.

[15] R Devon Hjelm, Alex Fedorov, Samuel Lavoie-Marchildon, Karan Grewal, Phil Bachman, Adam Trischler, and Yoshua Bengio. 2018. Learning deep representations by mutual information estimation and maximization. *arXiv preprint arXiv:1808.06670* (2018).

[16] Thomas N Kipf and Max Welling. 2016. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907* (2016).

[17] Srijan Kumar, Francesca Spezzano, VS Subrahmanian, and Christos Faloutsos. 2016. Edge weight prediction in weighted signed networks. In *2016 IEEE 16th International Conference on Data Mining (ICDM)*. IEEE, 221–230.

[18] Srijan Kumar, Xikun Zhang, and Jure Leskovec. 2019. Predicting dynamic embedding trajectory in temporal interaction networks. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. ACM, 1269–1278.

[19] Ao Li, Zhou Qin, Runshi Liu, Yiqun Yang, and Dong Li. 2019. Spam review detection with graph convolutional networks. In *Proceedings of the 28th ACM International Conference on Information and Knowledge Management*. 2703–2711.

[20] Huayi Li, Geli Fei, Shuai Wang, Bing Liu, Weixiang Shao, Arjun Mukherjee, and Jidong Shao. 2017. Bimodal distribution and co-bursting in review spam detection. In *Proceedings of the 26th International Conference on World Wide Web*. 1063–1072.

[21] Jundong Li, Harsh Dani, Xia Hu, and Huan Liu. 2017. Radar: Residual Analysis for Anomaly Detection in Attributed Networks.. In *IJCAI*. 2152–2158.

[22] Yuening Li, Xiao Huang, Jundong Li, Mengnan Du, and Na Zou. 2019. SpecAE: Spectral AutoEncoder for Anomaly Detection in Attributed Networks. In *Proceedings of the 28th ACM International Conference on Information and Knowledge Management*. 2233–2236.

[23] Fanzhen Liu, Shan Xue, Jia Wu, Chuan Zhou, Wenbin Hu, Cecile Paris, Surya Nepal, Jian Yang, and Philip S Yu. 2020. Deep Learning for Community Detection: Progress, Challenges and Opportunities. In *Proceedings of the 29th International Joint Conference on Artificial Intelligence*. 4981–4987.

[24] Fei Tony Liu, Kai Ming Ting, and Zhi-Hua Zhou. 2008. Isolation forest. In *2008 Eighth IEEE International Conference on Data Mining*. IEEE, 413–422.

[25] Ninghao Liu, Xiao Huang, and Xia Hu. 2017. Accelerated Local Anomaly Detection via Resolving Attributed Networks.. In *IJCAI*. 2337–2343.

[26] Ziqi Liu, Chaochao Chen, Xinxing Yang, Jun Zhou, Xiaolong Li, and Le Song. 2018. Heterogeneous graph neural networks for malicious account detection. In *Proceedings of the 27th ACM International Conference on Information and Knowledge Management*. 2077–2085.

[27] Jun Ma, Danqing Zhang, Yun Wang, Yan Zhang, and Alexey Pozdnoukhov. 2018. GraphRAD: A Graph-based Risky Account Detection System. (2018).

[28] Xiaoxiao Ma, Jia Wu, Shan Xue, Jian Yang, Quan Z Sheng, and Hui Xiong. 2021. A Comprehensive Survey on Graph Anomaly Detection with Deep Learning. *arXiv preprint arXiv:2106.07178* (2021).

[29] Renxin Mao, Zhao Li, and Jinhua Fu. 2015. Fraud transaction recognition: A money flow network approach. In *Proceedings of the 24th ACM International on Conference on Information and Knowledge Management*. 1871–1874.

[30] Eric WT Ngai, Yong Hu, Yiu Hing Wong, Yijun Chen, and Xin Sun. 2011. The application of data mining techniques in financial fraud detection: A classification framework and an academic review of literature. *Decision support systems* 50, 3 (2011), 559–569.

[31] Aaron van den Oord, Yazhe Li, and Oriol Vinyals. 2018. Representation learning with contrastive predictive coding. *arXiv preprint arXiv:1807.03748* (2018).

[32] Chung-Hoon Park and Young-Gul Kim. 2003. Identifying key factors affecting consumer purchase behavior in an online shopping context. *International journal of retail & distribution management* (2003).

[33] Zhen Peng, Wenbing Huang, Minnan Luo, Qinghua Zheng, Yu Rong, Tingyang Xu, and Junzhou Huang. 2020. Graph Representation Learning via Graphical Mutual Information Maximization. In *Proceedings of The Web Conference 2020*. 259–270.

[34] Zhen Peng, Minnan Luo, Jundong Li, Huan Liu, and Qinghua Zheng. 2018. ANOMALOUS: A Joint Modeling Approach for Anomaly Detection on Attributed Networks. In *Twenty-Seventh International Joint Conference on Artificial Intelligence IJCAI-18*.

[35] Bryan Perozzi and Leman Akoglu. 2018. Discovering communities and anomalies in attributed graphs: Interactive visual exploration and summarization. *ACM Transactions on Knowledge Discovery from Data (TKDD)* 12, 2 (2018), 1–40.

[36] S Benson Edwin Raj and A Annie Portia. 2011. Analysis on credit card fraud detection methods. In *2011 International Conference on Computer, Communication and Electrical Technology (ICCCET)*. IEEE, 152–156.

[37] Yuxiang Ren, Bo Liu, Chao Huang, Peng Dai, Liefeng Bo, and Jiawei Zhang. 2019. Heterogeneous deep graph infomax. *arXiv preprint arXiv:1911.08538* (2019).

[38] Lukas Ruff, Robert Vandermeulen, Nico Goernitz, Lucas Deecke, Shoaib Ahmed Siddiqui, Alexander Binder, Emmanuel Müller, and Marius Kloft. 2018. Deep one-class classification. In *International conference on machine learning*. 4393–4402.

[39] Gabriel Preti Santiago, Adriano CM Pereira, and Roberto Hirata Jr. 2015. A modeling approach for credit card fraud detection in electronic payment services. In *Proceedings of the 30th Annual ACM Symposium on Applied Computing*. 2328–2331.

[40] Saeedreza Shehnepoor, Mostafa Salehi, Reza Farahbakhsh, and Noel Crespi. 2017. Netspam: A network-based spam detection framework for reviews in online social media. *IEEE Transactions on Information Forensics and Security* 12, 7 (2017), 1585–1595.

[41] Ning Su, Yiqun Liu, Zhao Li, Yuli Liu, Min Zhang, and Shaoping Ma. 2018. Detecting Crowdturfing" Add to Favorites" Activities in Online Shopping. In *Proceedings of the 2018 World Wide Web Conference*. 1673–1682.

[42] Xing Su, Shan Xue, Fanzhen Liu, Jia Wu, Jian Yang, Chuan Zhou, Wenbin Hu, Cecile Paris, Surya Nepal, Di Jin, et al. 2021. A Comprehensive Survey on Community Detection with Deep Learning. *arXiv preprint arXiv:2105.12584* (2021).

[43] Petar Velickovic, William Fedus, William L Hamilton, Pietro Lio, Yoshua Bengio, and R Devon Hjelm. 2019. Deep Graph Infomax.. In *ICLR (Poster)*.

[44] Binghui Wang, Neil Zhenqiang Gong, and Hao Fu. 2017. GANG: Detecting fraudulent users in online social networks via guilt-by-association on directed graphs. In *2017 IEEE International Conference on Data Mining (ICDM)*. IEEE, 465–474.

[45] Daixin Wang, Jianbin Lin, Peng Cui, Quanhui Jia, Zhen Wang, Yanming Fang, Quan Yu, Jun Zhou, Shuang Yang, and Yuan Qi. 2019. A Semi-supervised Graph Attentive Network for Financial Fraud Detection. In *2019 IEEE International Conference on Data Mining (ICDM)*. IEEE, 598–607.

[46] Haibo Wang, Chuan Zhou, Jia Wu, Weizhen Dang, Xingquan Zhu, and Jilong Wang. 2018. Deep structure learning for fraud detection. In *2018 IEEE International Conference on Data Mining (ICDM)*. IEEE, 567–576.

[47] Jianyu Wang, Rui Wen, Chunming Wu, Yu Huang, and Jian Xion. 2019. Fdgars: Fraudster detection via graph convolutional networks in online app review system. In *Companion Proceedings of The 2019 World Wide Web Conference*. 310–316.

[48] Shuhao Wang, Cancheng Liu, Xiang Gao, Hongtao Qu, and Wei Xu. 2017. Session-based fraud detection in online e-commerce transactions using recurrent neural networks. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*. Springer, 241–252.

[49] Zhen Wang and Chao Lan. 2020. Towards a Hierarchical Bayesian Model of Multi-View Anomaly Detection. In *Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence*.

[50] Rui Wen, Jianyu Wang, Chunming Wu, and Jian Xiong. 2020. Asa: Adversary situation awareness via heterogeneous graph convolutional networks. In *Companion Proceedings of the Web Conference 2020*. 674–678.

[51] Haiqin Weng, Shouling Ji, Fuzheng Duan, Zhao Li, Jianhai Chen, Qinming He, and Ting Wang. 2019. CATS: Cross-Platform E-commerce Fraud Detection. In *2019 IEEE 35th International Conference on Data Engineering (ICDE)*. IEEE, 1874–1885.

[52] Haiqin Weng, Zhao Li, Shouling Ji, Chen Chu, Haifeng Lu, yu Du, and Qinming He. 2018. Online e-commerce fraud: A large-scale detection and analysis. In *2018 IEEE 34th International Conference on Data Engineering (ICDE)*. IEEE, 1435–1440.

[53] Yongji Wu, Defu Lian, Yiheng Xu, Le Wu, and Enhong Chen. 2020. Graph Convolutional Networks with Markov Random Field Reasoning for Social Spammer Detection. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 34. 1054–1061.

[54] Zonghan Wu, Shirui Pan, Fengwen Chen, Guodong Long, Chengqi Zhang, and S Yu Philip. 2020. A comprehensive survey on graph neural networks. *IEEE Transactions on Neural Networks and Learning Systems* (2020).

[55] Zenghua Xia, Chang Liu, Neil Zhenqiang Gong, Qi Li, Yong Cui, and Dawn Song. 2019. Characterizing and detecting malicious accounts in privacy-centric mobile social networks: A case study. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. 2012–2022.

[56] Xu Yang, Cheng Deng, Feng Zheng, Junchi Yan, and Wei Liu. 2019. Deep spectral clustering using dual autoencoder network. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 4066–4075.

[57] Rex Ying, Ruining He, Kaifeng Chen, Pong Eksombatchai, William L Hamilton, and Jure Leskovec. 2018. Graph convolutional neural networks for web-scale recommender systems. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. 974–983.

[58] Mengchen Zhao, Zhao Li, Bo An, Haifeng Lu, Yifan Yang, and Chen Chu. 2018. Impression Allocation for Combating Fraud in E-commerce Via Deep Reinforcement Learning with Action Norm Penalty.. In *IJCAI*. 3940–3946.

[59] Bo Zong, Qi Song, Martin Renqiang Min, Wei Cheng, Cristian Lumezanu, Daeki Cho, and Haifeng Chen. 2018. Deep autoencoding gaussian mixture model for unsupervised anomaly detection. In *International Conference on Learning Representations*.