Removing Camouflage and Revealing Collusion: Leveraging Gang-crime Pattern in Fraudster Detection

Lewen Wang
Microsoft
Beijing, China
lewen.wang@microsoft.com

Haozhe Zhao
Peking University
Beijing, China
hanszhao@pku.edu.cn

Cunguang Feng
Peking University
Beijing, China
fcg@pku.edu.cn

Weiqing Liu
Microsoft
Beijing, China
weiqing.liu@microsoft.com

Congrui Huang
Microsoft
Beijing, China
conhua@microsoft.com

Marco Santoni
Flowe
Basiglio, Italy
marco.santoni@flowe.com

Manuel Cristofaro
Flowe
Basiglio, Italy
manuel.cristofaro@flowe.com

Paola Jafrancesco
Flowe
Basiglio, Italy
paola.jafrancesco@flowe.com

Jiang Bian
Microsoft
Beijing, China
jiang.bian@microsoft.com

ABSTRACT

As one of the major threats to the healthy development of various online platforms, fraud has become increasingly committed in the form of gangs since collusive fraudulent activities are much easier to obtain illicit benefits with lower exposure risk. To detect fraudsters in a gang, spatio-temporal graph neural network models have been widely applied to detect both temporal and spatial collusive patterns. However, a closer peek into real-world records of fraudsters can reveal that fraud gangs usually conduct community-level camouflage, specified by two types, i.e., temporal and spatial camouflage. Such camouflage can disguise gangs as benign communities by concealing collusive patterns and thus deceiving many existing graph neural network models. In the meantime, many existing graph neural network models suffer from the challenge of extreme sample imbalance caused by rare fraudsters hidden among massive users. To handle all these challenges, in this paper, we propose a generative adversarial network framework, named **A**dversarial **C**amouflage **D**etector[1], to detect fraudsters. Concretely, this ACD framework consists of four modules, in charge of **community division**, **camouflage identification**, **fraudster detection**, and **camouflage generation**, respectively. The first three modules form up a discriminator that uses spatio-temporal graph neural networks as the foundation model and enhance fraudster detection by amplifying the gangs' collusive patterns through automatically identifying and removing camouflage. Meanwhile, the camouflage generation module plays as the generator role that generates fraudsters samples by competing against the discriminator to alleviate the challenge of sample imbalance and increase the model robustness. The experimental result shows that our proposed method outperforms other methods on real-world datasets.

CCS CONCEPTS

• **Theory of computation** → **Graph algorithms analysis**; **Adversarial learning**; • **Computing methodologies**; • **Applied computing**;

KEYWORDS

Fraud; camouflage; graph neural networks; generative adversarial networks

ACM Reference Format:
Lewen Wang, Haozhe Zhao, Cunguang Feng, Weiqing Liu, Congrui Huang, Marco Santoni, Manuel Cristofaro, Paola Jafrancesco, and Jiang Bian. 2023. Removing Camouflage and Revealing Collusion: Leveraging Gang-crime Pattern in Fraudster Detection. In *Proceedings of the 29th ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD '23), August 6–10, 2023, Long Beach, CA, USA.* ACM, New York, NY, USA, 12 pages. https://doi.org/10.1145/3580305.3599895

1 INTRODUCTION

Fraud includes various harmful activities on different online platforms, such as spam reviews, credit card fraud, tax fraud, money laundering[2], etc. Fraud detection has become an important research topic in recent years due to significantly increasing losses caused by those harmful activities. Particularly, newly released Federal Trade Commission data demonstrates more than $5.8 billion in losses, reported by consumers, due to fraud in 2021, increased by over 70% year-on-year.

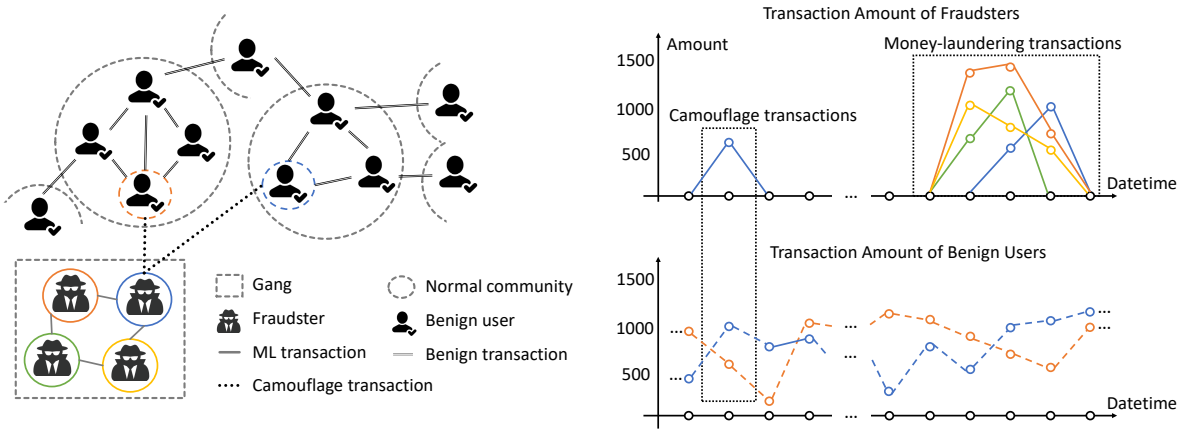
Some thorough investigations [30] have shown that the majority of fraud activities are conducted by gang crime, meaning that a

[1]The code is publicly available at https://github.com/lwwang1995/ACD.

[2]https://www.cimaglobal.com/Documents/ImportedDocuments/31_Fraud_and_Money_Laundering.pdf

KDD '23, August 6–10, 2023, Long Beach, CA, USA

ACM ISBN 979-8-4007-0103-0/23/08…$15.00
https://doi.org/10.1145/3580305.3599895



(a) Transaction Graph - Spatial Camouflage

(b) Transaction Amount Curve - Temporal Camouflage

Figure 1: A money-laundering gang case with the spatial and temporal camouflage: four launderers collaborated on money laundering, and one of them conducted a series of temporal and spatial camouflage activities with two benign users to decrease the suspicious of the gang. By deliberately making transactions with the users outside the community (Figure 1(a)), they could largely conceal the spatial collusive patterns. Meanwhile, these camouflage activities have happened in the salience stage (Figure 1(b)), which also to a great extent obscures the temporal collusive patterns of the gang. The color and line style of the circle surrounding the user on the left is aligned with the user transaction amount line on the right.

group of collusive users or accounts work together to facilitate fraud[36], since collusive fraudulent activities are much easier to obtain illicit benefits with lower exposure risks. Recent works[2, 20] have revealed that, compared to a benign community, fraudsters within a gang show obvious temporal aggregation and spatial aggregation patterns, in this paper, named **temporal collusion** and **spatial collusion**, respectively. Specifically, temporal collusion reflects that fraudsters in a gang tend to conduct illegal activities simultaneously within a very short time window in order to squeeze the risk of exposure to the regulator's detection. In addition, spatial collusion refers to the pattern that fraudsters in the same gang usually form dense inter-connections while, in contrast, yielding rare interactions with benign users.

Many existing works attempt to detect fraudsters by recognizing collusive patterns. In particular, graph neural network-based methods [15, 37, 38], network analysis methods [24, 29] and dense sub-graph detection methods [8, 28, 42] are used to discover the spatial collusion and further detect fraudsters. To handle temporal collusion, some works utilize the temporal consistency between users and neighbors to capture fraudsters [17]. Furthermore, some works focus on capturing both temporal and spatial collusion when detecting fraudsters [1, 7, 35].

However, few of these prior works have paid enough attention to the camouflage behaviors of fraudsters. After investigating many real-world records of fraud gangs, we find that fraudsters usually leverage **community-level camouflage** behaviors, specified by two types, i.e. **temporal camouflage** and **spatial camouflage**, to disguise themselves as benign communities by concealing collusive patterns. Specifically, to obscure the temporal collusive patterns, fraudsters in the gang introduce temporal camouflage by engaging in illicit activities not limited to a short time window. In addition, by leveraging spatial camouflage, fraudsters intentionally connect themselves to some benign entities outside the gang [21], which can effectively conceal the spatial collusive patterns. Figure 1 shows a typical real money-laundering[3] gang with temporal and spatial camouflage, captured by Flowe, a commercial bank from Europe. From this figure, we can clearly observe that the temporal and spatial collusive patterns are concealed by camouflage behaviors.

There have been some recent efforts to identify camouflage based on user-level information. For example, Liu et al. [18, 21] attempted to disclose camouflage by identifying the connections between fraudsters and benign users. However, such an approach yields limited capability for recognizing all camouflage since it ignores invaluable community-level information.

In this paper, we attempt to recognize and utilize collusive patterns of gangs to assist in identifying camouflage and further benefit fraud detection. Concretely, we design a generative adversarial network framework, *Adversarial Camouflage Detector*, referred to as ACD, for identifying fraudsters with camouflage behaviors. This ACD framework consists of delicately designed modules, in charge of **community division**, **camouflage identification**, **fraudster detection**, and **camouflage generation**, respectively. In the first place, the community division module aims to identify the communities among all users and extract community-level information. Based on community division, the camouflage identification module is used to detect the camouflage behaviors by amplifying collusive patterns of gang crimes. Given the identified camouflage, the fraudster detection module employs the spatio-temporal graph neural

[3]The process of money laundering is high-volume flows of funds through chains of accounts and finally moving dirty money to an untraceable destination.

network to capture both the temporal and spatial collusive patterns. However, due to the lack of fraud samples [3, 10, 19, 22] and supervision for camouflage, direct optimization over the above three modules may not be able to result in robust detection models. To handle this sample imbalance problem, the **camouflage generation module** is designed to conduct adversarial learning [13], by generating fraud gang samples with camouflage behaviors, which could also increase the robustness of the above three modules against undiscovered camouflage behaviors. Intuitively, the adversarial learning alternating between the camouflage generation module and the others is analogous to that of Generative Adversarial Networks [4], where the camouflage generation module refers to the generator and the other three together refer to the discriminator. Overall, the main contributions of our paper can be summarized as follows:

- We are the first to point out that community-level collusive patterns are key to identifying camouflage in fraud scenarios.
- We propose a novel framework *ACD* for identifying camouflage behaviors and detecting fraudsters, which also contains a fraud sample generation method based on the proposed collusive patterns, to handle the sample imbalance problem.
- We conduct extensive experiments on real-world banking transaction datasets and the public spam review dataset YelpZip to evaluate the performance of *ACD* and provide in-depth analysis on how it handles camouflage.

2 PROBLEM DEFINITION

To explore temporal and spatial collusive patterns for accurate fraud detection, we define the problem of fraud detection as a classification task over users who form a dynamic graph. In this graph, nodes represent users and edges represent connections between users, such as user transactions, using the same device, or rating the same product. To model the dynamic graph, we first define the concept of a snapshot graph to describe the information of users and connections within a certain time period, and then define a graph trajectory that consists of a continuous set of snapshot graphs to capture the dynamics of users and their connections.

Definition 1. Snapshot Graph. Snapshot graph $G_t = \{V, E_t, X_t\}$ is an undirected static graph that contains information within a certain time period t. $V = \{v_1, v_2, ..., v_n\}$ is the set of nodes and each node is associated with a user. $n = |V|$ represents the number of users. E_t is the set of edges and each edge is associated with a connection between users. $X_t = \{X_t(V), X_t(E_t)\}$, where $X_t(V) = \cup_{v \in V}\{x_t(v)\}$, $x_t(v)$ refers to features for node v during the time period t, $X_t(E_t) = \cup_{e \in E_t}\{x_t(e)\}$, $x_t(e)$ refers to features for edge e during the time period t.

Note that, in this paper, if not specified, lowercase letters such as $x_t(v)$ denote vectors, while uppercase letters such as $X_t(V)$ denote matrices by stacking the corresponding vectors $x_t(v)$. Then, we define the graph trajectory over a certain time span:

Definition 2. Graph Trajectory. $G_{\mathcal{T}} = \{G_{t_1}, G_{t_2}, ..., G_{t_k}\}$ is the Graph trajectory over a certain time span $\mathcal{T} = \{t_1, t_2, \ldots, t_k\}$. It contains a sequence of snapshot graphs, which is arranged in chronological order, where $k = |T|$ represents the number of time steps in $\mathcal{T}$. Each snapshot in $G_{\mathcal{T}}$ has the same node set V and a different edge set.

To obtain a community division to benefit camouflage identification and fraud detection, we define a universal graph as follows,

Definition 3. Universal Graph. A universal graph $\mathcal{G} = \{V, E, A\}$ is an undirected static graph. It shares the same node set V with each snapshot graph in the graph trajectory $G_{\mathcal{T}}$. E is the set of edges, which is the union of edges for all the graph snapshots in $G_{\mathcal{T}}$, represented as $E = \cup_{i \in 1,...,k} E_{t_i}$. The connection strength for edges can be donated by a symmetric adjacency matrix $A = \{a_{ij}\} \in \mathbb{R}^{n \times n}$, where a_{ij} denotes the weight for e_{ij}.

Then, community division could be obtained based on the universal graph. We formulate the community detection problem as follows,

Problem Definition 1. Community Detection. The goal of community detection is to assign users into m communities, written as $C = \{c_1, c_2, ..., c_m\}$, where $c_i = \{v_{i1}, v_{i2}, ..., v_{in_i}\}$, $\cup_{i=1}^{m} \cup_{j=1}^{n_i} \{v_{ij}\} = V$, $n_i = |c_i|$ refers to the number of users in community c_i. Note that each community c_i is a subgraph of $\mathcal{G}$. The communities are non-overlapping, where $c_i \cap c_j = \emptyset, \forall i, j (i \neq j)$.

The notion of a community seems rather intuitive, and there is no universally agreed-upon definition in the literature. In this paper, we propose that nodes sharing dense connections and similar behavior delineate communities. All communities can be divided into gangs and normal communities. Then, we formulate the fraud detection problem as a classification task as follows,

Problem Definition 2. Fraud Detection. Given the universal graph $\mathcal{G}$, the graph trajectory $G_{\mathcal{T}}$, and user-level label $Y(V) = \{y_{v_1}, ..., y_{v_1}\}, y_{v_i} \in \{0, 1\}$, we aim to get a model to make community division, and further classify whether a user v has ever committed fraud activities from t_1 to t_k based on the detected communities.

In the context of fraud detection, $y_v = 1$ represents fraudsters and $y_v = 0$ represents benign users. Given the ground-truth $Y(V)$ and the output of the model $\hat{Y(V)} = \{\hat{y_{v_1}}, ..., \hat{y_{v_m}}\}$, **the fraudster detection loss** for the classification task can be written as:

$$\ell^{fraud\_det} = -\frac{1}{n}\sum_{v \in V} y_v log(\hat{y_v}) + (1 - y_v) log(1 - \hat{y_v}). \quad (1)$$

Given the user-level label $Y(V)$ and community division C, the community-level label $Y(C)$ could be calculated according to the user-level label $Y(V)$. Similarly, $y_c = 1$ represents gangs and $y_c = 0$ represents benign communities.

3 COLLUSIVE PATTERNS AND CAMOUFLAGE BEHAVIOR FOR FRAUD GANGS

Fraudsters in the same gang are inclined to behave in collusive patterns compared to benign communities, including temporal collusion and spatial collusion. Specifically, temporal collusion reflects the phenomenon that fraudsters in a gang always make the fraud activities exposed in a limited time simultaneously since the regulator will freeze accounts as soon as suspicious activities have been detected. Spatial collusion refers to the pattern that fraudsters in the same gang usually tend to form dense interconnections. The reason may be the limitation of cost on the devices, accounts, and other resources, making fraudsters in the same gang have to frequently use only a small number of devices or accounts to reduce cost. To

Table 1: Comparison of the average collusion score for benign communities and fraud gangs in YelpZip and Flowe dataset.

dataset	YelpZip				Flowe
graph construction	U-P-U	U-S-U	U-T-U	U-PST-U	transaction
temporal collusion score (gang/benign community)	0.1728/0.0813	0.1881/0.0994	0.9645/0.9498	0.9510/0.9041	0.6316/0.3789
spatial collusion score	0.0237/0.0119	0.0826/0.0466	0.4692/0.3708	0.9997/0.9990	0.9173/0.7963

measure collusive patterns, we begin by quantifying temporal collusion and spatial collusion using the temporal collusion score and the spatial collusion score, respectively.

3.1 Definition of Collusion Score

Based on the community division, the temporal collusion score is calculated by temporal behavioral correlation among users within the same community. The temporal behavior of a user v can be represented as a vector $x_{v,tem} = (x_{v,t_1}, x_{v,t_2}, ..., x_{v,t_k})$, where x_{v,t_j} represents the total amount of activities/transactions/reviews during a fixed period of time. Then, the temporal correlation of v_{ip} and v_{iq} in c_i can be measured by *Pearson Correlation Coefficient*[4] calculated based on $x_{v_{ip},tem}$ and $x_{v_{iq},tem}$, written as $\rho_{v_{ip},v_{iq}}$. We use a **t**emporal **c**ollusion **m**atrix, abbreviated as TCM, $TCM \in \mathbb{R}^{n_i \times n_i}$, to represent the temporal correlation of any two users in the community c_i. The value of p-th row and the q-th column in the matrix is $\rho_{v_{ip},v_{iq}}$. Then the temporal collusion score for c_i can be measured by a simple average of the temporal correlation of any two users within the community as Equation 2.

$$\rho^t_{c_i} = \frac{\sum_{v_{ip} \in c_i} \sum_{v_{iq} \in \{c_i - v_{ip}\}} \rho_{v_{ip},v_{iq}}}{n_i(n_i - 1)}. \tag{2}$$

To quantify spatial collusion, we propose using the spatial collusion score to measure the relative closeness of intra-connections within a community compared with the connections with outside users. Inspired by the *Modularity Measure* [25] proposed by Newman, which is used to evaluate community detection algorithms, the spatial collusion score of community c_i can be calculated as follows,

$$\rho^s_{c_i} = \frac{\sum_{v_p \in c_i} \sum_{v_q \in c_i} a_{pq}}{\sum_{v_p \in c_i} \sum_{v_q \in V} a_{pq}}, \tag{3}$$

The weight a_{ij} can be either the total amount of transactions or the number of connections between the corresponding two users. If there is no connection between two users, the corresponding a is 0.

3.2 Statistical Analysis of Collusion Scores

To compare the collusion score for gangs and benign communities, we conduct statistical analysis on the money-laundering dataset Flowe and the spam review dataset YelpZip. Details about the two datasets will be introduced in Section 5.1. For both datasets, we take users as nodes. For the Flowe dataset, we take transactions as edges. For the YelpZip dataset, four kinds of edges are considered, where U-P-U connects two users who reviewed the same product, U-S-U connects two users who rated the same product with the same score, U-T-U connects two users who rated the same product in the same month, U-PST-U connects two users who rated the same product with the same score in the same month. Overlooking camouflage, we applied the state-of-the-art community detection method, Leiden algorithm [31], to generate a community division based on the universal graph. Table 1 shows that gangs have significantly higher collusion scores compared to benign communities, indicating that both temporal and spatial collusion scores are valuable indicators for detecting fraudsters.

[4]https://en.wikipedia.org/wiki/Pearson_correlation_coefficient

3.3 Camouflage Conceals Collusive Patterns

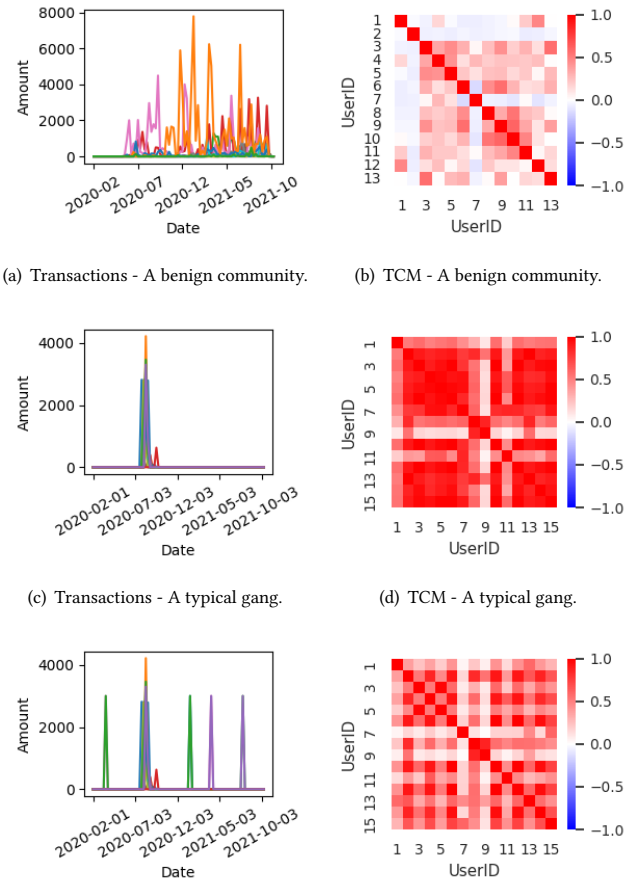
Taking temporal collusion and temporal camouflage as an example, the case demonstrated in Figure 2 shows a gang owns a higher temporal collusion score than a benign community, and how temporal camouflage conceals temporal collusion for the gang. Figure 2 is a visualization of historical transaction amounts and TCM(temporal collusion matrix) for a benign community(Figure 2(a) and Figure 2(b)), a typical money-laundering gang without temporal camouflage(Figure 2(c) and Figure 2(d)) and the gang with temporal camouflage(Figure 2(e) and Figure 2(f)). Figure 2(c) shows that launderers in the gang always transact actively during the money-laundering duration and have few transactions outside the duration, indicating temporal collusion. In contrast, Figure 2(a) shows that users in the benign community transact actively almost all the time and the transaction amounts of community members change randomly over time. As a result, the temporal collusion score of the money-laundering gang(Figure 2(d)) is higher than that of the benign community(Figure 2(b)). However, as shown in Figure 2(e) and Figure 2(f), although only a few camouflage transactions were made by the launderers, it resulted in a significant decrease in the temporal collusion score. On the contrary, if we could accurately identify and remove the camouflage transactions, the temporal collusion score would be significantly increased to benefit detection.

4 PROPOSED METHODS

We propose an *Adversarial Camouflage Detector* to identify fraudsters under disguise, namely ACD. In this section, we introduce the details of the proposed ACD.

4.1 Overall Framework

Similar to Generative Adversarial Networks, ACD contains a discriminator and a generator. The discriminator, consisting of a *Community Division Module*, a *Camouflage Identification Module*, and a *Fraudster Detection Module*, aims to detect fraudsters under camouflage. The community division module aims to assign users to communities to provide community-level information. Based on community division, the camouflage identification module is designed to identify and remove camouflage by amplifying gangs'



(a) Transactions - A benign community. (b) TCM - A benign community.

(c) Transactions - A typical gang. (d) TCM - A typical gang.

(e) Transactions - A gang with camouflage. (f) TCM - A gang with camouflage.

Figure 2: Case study for temporal collusion and temporal camouflage. The left sub-figures are historical transaction amounts, and lines with different colors represent different users. The right sub-figures are visualization of the temporal collusion matrix.

collusive patterns, serving as a prerequisite step for the fraudster detection module. After removing camouflage, the fraudster detection module detects fraudsters by capturing temporal and spatial collusive patterns. To solve the sample imbalance problem, the camouflage generation module serves as the generator to create fraudulent samples by generating camouflage in existing gangs. The discriminator and the generator are optimized alternatively. Figure 3 summarizes the optimization flows for the discriminator and the generator. We will elaborate on the details of them, together with the inference data flow, in the rest of this subsection.

4.1.1 Optimization Flow for the Discriminator. As shown in Figure 3, the optimization for the discriminator can be divided into two phases. In the first phase, the community division module is optimized to divide users into communities. At the very beginning, the community division is based on the initial universal graph $\mathcal{G}$ without considering camouflage. In the second phase, with community division C, the camouflage identification module identifies camouflage by maximizing collusive patterns of gangs. After removing the camouflage, the fraudster detection module is used to accurately detect fraudsters. The modules in the two phases are optimized alternatively. After removing camouflage in the second phase, the universal graph is updated and further used to optimize the community division module in the next round. After several rounds of optimization, when the discriminator converges, the optimization for the generator starts.

4.1.2 Optimization Flow for the Generator. Due to the lack of fraudster samples, the straightforward method to optimize the two modules may lead to the lack of robustness of the modules. In the optimization flow for the generator, the camouflage generation module is proposed to create fraud gang samples by adding camouflages to existing gangs. In this way, the generator learns to create fake fraudulent samples by incorporating feedback from the discriminator, specifically, to make the discriminator classify its output as benign users. After several rounds of optimization, when the generator converges, the generated fraudulent samples can be used to expand the fraudulent samples in the training set.

4.1.3 Inference Data Flow. Only the camouflage identification module and the fraudster detection module are used in the inference phase. To support newly registered users in actual business scenarios, the community division is not used in the inference phase. This enables ACD to handle newly added nodes in the universal graph and the graph trajectory.

4.2 Details of Architecture Design

All the modules share a similar spatio-temporal graph architecture [39]. In this section, we first introduce the spatio-temporal graph architecture. Then, we introduce the details of each module.

4.2.1 Spatio-temporal Graph Architecture. The input of modules is the graph trajectory $G_{\mathcal{T}}$. Before the spatio-temporal graph layer, two two-layer perceptrons are applied to each node to transform $X_t(V)$ to $H_t^0(V)$, and to each edge to transform $X_t(E_t)$ to $H_t(E_t)$, where $H_t(E_t) \in \mathbb{R}^{n^t \times d_e}$ and $H_t^0(V) \in \mathbb{R}^{n \times d_v}$. n^t is the number of edges at time t, d_v and d_e are the dimensions of node embedding and edge embedding. The spatio-temporal graph layer consists of a spatial embedding layer and a temporal embedding layer.

The spatial embedding layer is designed to mine the spatial patterns behind the interactions between users for each snapshot graph G_t. The input corresponding to graph snapshot G_t is $H_t^0(V)$ and $H_t(E_t)$. The spatial embedding layer consists of l graph attention layers[32]. In each layer, node embedding is updated with neighbors' information. Notably, the spatial embedding layers of the four modules share the same parameters to avoid over-fitting.

The input of the temporal embedding layer is the output of the spatial embedding layer at each time-step $\{H_{t_1}^l(V), ..., H_{t_k}^l(V)\}$. In the temporal embedding layer, an RNN model is used to capture temporal patterns based on the sequence $(h_{t_1}^l(v), ..., h_{t_k}^l(v))$. The output of the RNN model is written as $(h_{t_1}^{l+1}(v), ..., h_{t_k}^{l+1}(v))$. In this paper, if not specified, the RNN model is specified as an LSTM [9].

4.2.2 Community Division Module. The community division module divides users into communities based on the latest universal

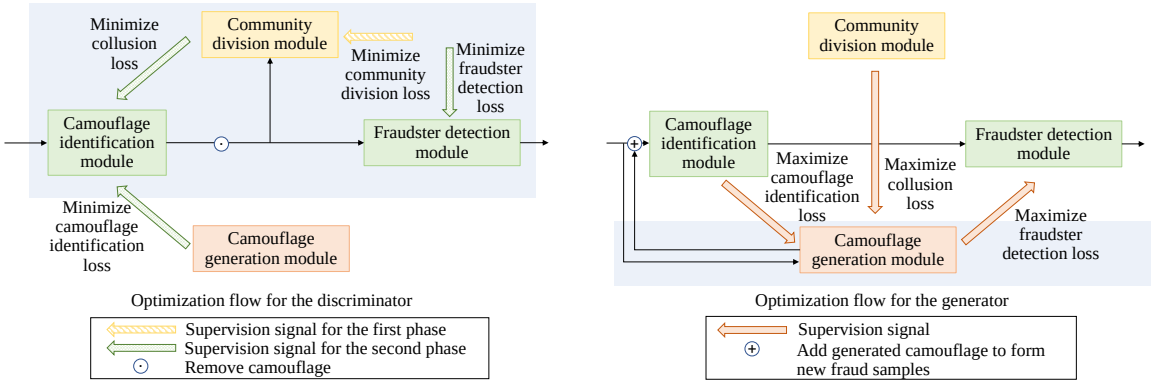


Figure 3: The framework of Adversarial Camouflage Detector.

graph. Contrastive learning with clustering is commonly used in community detection [26, 34]. In this paper, we follow this approach to divide the community. Based on the last time step output of the RNN, the k-means clustering algorithm is used to divide all the nodes into m communities, where m is a hyper-parameter. During the model training process, the obtained communities are labeled as gangs and benign communities according to the user-level label. Notably, the fraud gang samples generated by the camouflage generation module will not be included in community detection, they will maintain the original communities.

4.2.3 Camouflage Identification Module. Before introducing the camouflage identification module, we first introduce how camouflage is reflected in a graph snapshot and how to remove camouflage from the graph snapshot. Camouflage during period t is reflected in both node features and edge features at G_t. The operation of removing camouflage in this paper is from the edge as the entry point, and node features are adjusted according to the edge features. Intuitively, the node features could be the number of transactions/reviews/connections, and the edge features could be the transaction amount/number of connections between two users. The operation of increasing or decreasing the number of transactions/reviews on edges could be aggregated to related nodes.

To remove camouflage, the camouflage identification module outputs a mask for each edge related to fraudulent nodes in each snapshot graph. In the module, the temporal embedding layer is specified as a BiLSTM [11] to make node information from different time steps to enhance each other. Given the output of the temporal embedding layer and edge embedding $H_t(E)$, the module generates mask embedding $h'_t(m_{ij})$ for each edge related to fraudulent nodes. Details about how to obtain $h'_t(m_{ij})$ are introduced in the Section-A of the Appendix.

Finally, the module converts each mask embedding to a mask value, which is a continuous value between 0 and 1. To this end, we adopt a smooth approximation by tanh function[12] to generate a mask as the following equation,

$$y_t(\hat{m}_{ij}) = 0.5 * tanh(\beta(Wh'_t(m_{ij}) + b)) + 0.5, \tag{4}$$

where W and b are learnable parameters, β is a hyper-parameter used to control the strength of smooth approximation and the output $y_t(\hat{m}_{ij}) \in (0, 1)$. For convenience, we name the nodes and edges corresponding to the camouflage as the camouflage nodes and camouflage edges. For those normal edges, the mask is close to 1, while for camouflage edges, the mask should be less than 1. Then, the edge features will be updated by element-wise multiplication with the mask, which is represented as $\odot$ in Figure 3. Then, corresponding node features are adjusted to align with edge features. In this way, we can easily remove camouflage from the snapshot graph. Correspondingly, the edge weight a_{ij} of the camouflage edge in the universal graph $\mathcal{G}$ will also be reduced.

4.2.4 Fraudster Detection Module. The output of this model is the detection result. The spatial embedding layer and the temporal embedding layer could potentially mine spatial collusive patterns and temporal collusive patterns to assist in fraud detection. Based on the last step output of the RNN $h^{l+1}_{t_k}(v)$, a linear combination followed by a softmax is used to generate the output $Y(\hat{V})$.

4.2.5 Camouflage Generation Module. The camouflage generation module creates new fraudulent samples by generating and adding camouflage to existing fraud gangs. Firstly, the camouflage generation module generates camouflage by outputting camouflage edge feature $x_t(\hat{e}_{ij})$, which contains all the information related to the camouflage, for the connections between each fraudster and other users in each snapshot graph, regardless of whether there are edges between them in the original ones. To ensure the camouflage generation module can only change initial features by adding camouflage, ReLU is used to make each element in the generated camouflage edge features no less than 0. Then, by element-wise adding generated camouflage edge features $x_t(\hat{e}_{ij})$ with initial edge features $x_t(e_{ij})$, the edge features for the new fraudulent samples are generated. If there is no edge between v_i and v_j in the original snapshot graph, each bit in the $x_t(e_{ij})$ will be padded with 0. After that, similar to the remove operation, edge features and node features for the newly generated fraudulent samples should be aligned. The process is represented as $\bigoplus$ in Figure 3. Different from the remove operation, the universal graph $\mathcal{G}$ is not updated.

4.3 Optimization Objectives

This section will introduce the objective functions. For convenience, the trainable parameters in the community division module, the

camouflage identification module, the fraudster detection module, and the camouflage generation module are written as θ_1, θ_2, θ_3 and ω, respectively. The parameter to be optimized is referred to as the subscript of the objective function.

4.3.1 Optimization Objective for Community Division. The optimization objective for the community division module considers both graph structure and node attributes. We introduce contrastive learning to make the output of the community division module $h(v)$ contains more graph structure information by pulling the node v and its neighbors(positive sample) closer to each other in the embedding space while pushing the node v away from other nodes (negative sample). At the very beginning, camouflage may greatly hurt community division since camouflage edges related to spatial camouflage always connect benign and fraudulent nodes. To benefit community division at the early stage of the training process, we use temporal correlation introduced in Section 3.1 to weight positive samples. By measuring behavioral similarity, the temporal correlation could reduce the weights for camouflage edges. To this end, **the community division loss** can be formulated as follows,

$$\ell_{\theta_1}^{comm} = \frac{1}{n}\sum_{v_i\in V}\Big(\sum_{e_{ij}\in E}\rho_{v_i,v_j}^{t}\left\|h(v_i)-h(v_j)\right\|^2 - \sum_{v_j\in M}\left\|h(v_i)-h(v_j)\right\|^2\Big), \tag{5}$$

where M are negative samples, which are randomly selected from V except the node's neighbors. Notably, E and ρ_{v_i,v_j}^{t} are updated with the latest universal graph and graph trajectory, respectively.

4.3.2 Optimization Objectives for Camouflage Identification and Fraudster Detection. The optimization objectives of these two modules are from the view of the regulator, that is, accurately identifying camouflage and detecting fraudsters. In this paper, we use the collusion loss and the camouflage identification loss to optimize the camouflage identification module, and use the fraudster detection loss introduced in Section 2 to guide the two modules in detecting fraudsters. In this section, we introduce the collusion loss and the camouflage identification loss.

Collusion Loss is designed to guide the camouflage identification module to remove camouflage edges by maximizing the increment of the temporal and spatial collusion scores after the mask operation, compared to the initial collusion scores. The temporal collusion loss can be calculated as the following equation,

$$\ell_{\theta_2}^{temp} = -\sum_{c_i\in C} y_{c_i}(\hat{\rho}_{c_i}^{t}-\rho_{c_i}^{t}). \tag{6}$$

where $\hat{\rho}_{c_i}^{t}$ refers to the temporal collusion score of c_i after the mask operation. Similarly, the spatial collusion loss can be calculated as:

$$\ell_{\theta_2}^{spat} = -\sum_{c_i\in C} y_{c_i}(\hat{\rho}_{c_i}^{s}-\rho_{c_i}^{s}). \tag{7}$$

Camouflage Identification Loss could guide the model to remove camouflage with a supervision signal. At the very beginning, the supervision signal could only indicate spatial camouflage between the benign users and the fraudsters. After the first round of optimization, an extra supervision signal could be provided by the camouflage generation module. Based on that, the masks introduced in Section 4.2.3 can be learned in a supervised way with a binary classification task, the masks for camouflage edges generated by the camouflage generation module are optimized toward 0, and the masks for other edges are optimized toward 1. In this paper, cross-entropy is used to calculate the camouflage identification loss as the following equation,

$$\ell_{\theta_2}^{camo\_det} = -\frac{1}{|E|}\sum_{e_{ij}\in E} y_t(m_{ij})log(y_t(\hat{m}_{ij})) + (1-y_t(m_{ij}))log(1-y_t(\hat{m}_{ij})). \tag{8}$$

Similar to other multi-task learning works for fraud detection [33, 41], the objective function for the second phase optimization of the discriminator can be formulated as follows,

$$\ell_{\theta_2,\theta_3}^{detector} = \ell_{\theta_2,\theta_3}^{fraud\_det} + \gamma_1\ell_{\theta_2}^{spat} + \gamma_2\ell_{\theta_2}^{temp} + \gamma_3\ell_{\theta_2}^{camo\_det}, \tag{9}$$

where γ_1, γ_2, and γ_3 are hyper-parameters.

4.3.3 Optimization Objective of Camouflage Generation. The camouflage behaviors are generated from the view of the fraudsters, that is, to avoid being detected with minimal cost. Intuitively, camouflage behaviors bring a cost, the more camouflage, the higher the cost. Based on this, the generator follows a fundamental principle that adds as few camouflages as possible to conceal gangs' collusive patterns as much as possible. To this end, the objective function of the camouflage generation module consists of two parts. The first part is to compete against the discriminator, including maximizing the fraudster detection loss, the camouflage identification loss, and the collusion loss. Considering the camouflage cost, L1 regularization is used to make sure only a few camouflages are generated. The overall loss function for the generator can be formulated as the following equation,

$$\ell_{\omega}^{camo\_gen} = -\ell_{\omega}^{fraud\_det} - \gamma_4\ell_{\omega}^{spat} - \gamma_5\ell_{\omega}^{temp} - \gamma_6\ell_{\omega}^{camo\_det} + \gamma_7\ell_{\omega}^{L1}, \tag{10}$$

where γ_4, γ_5, γ_6 and γ_7 are hyperparameters. The generated camouflage further serves as the supervision signal to optimize the camouflage identification module. The number of generated camouflages is very small, resulting in an unbalanced supervision signal. This ensures that only a few edges are judged as camouflage edges by the camouflage identification module, which aligns the camouflage identification and camouflage generation.

4.3.4 Adversarial Optimization of the Discriminator and the Generator. *ACD* is worked based on the generative adversarial networks. Specifically, the generator competes against the discriminator from two aspects. Firstly, the camouflage generation module directly outputs camouflage. Its adversary, the camouflage identification module, attempts to identify camouflage. From the perspective of optimization, given the community division C, by fixing θ_2, the camouflage generate module optimizes ω by maximizing collusion loss ℓ_{ω}^{temp} and ℓ_{ω}^{spat} and maximizing camouflage identification loss $\ell_{\omega}^{camo\_det}$ to adverse the camouflage identification module. On the contrary, by fixing ω, the camouflage identification module optimizes θ_2 by minimizing collusion loss $\ell_{\theta_2}^{temp}$ and $\ell_{\theta_2}^{spat}$, and minimizing camouflage identification loss $\ell_{\theta_2}^{camo\_det}$. From the other

Table 2: Experimental results.

	Flowe						YelpZip					
	FRAUDRE	CARE	GC	STGNN	CD	ACD	FRAUDRE	CARE	GC	STGNN	CD	ACD
Precision	0.1488	0.1593	0.0911	0.2121	0.2112	**0.2495**	0.4369	0.4932	0.5013	0.5116	0.5005	**0.5427**
Recall	0.5249	0.5824	0.3295	0.7663	0.6811	**0.8652**	0.2539	0.2869	0.2914	0.2974	0.2926	**0.3283**
FPR	0.0083	0.0085	0.0091	0.0079	0.0078	**0.0075**	0.1718	0.1547	0.1521	0.1490	0.1519	**0.1430**
Marco-F1	0.6532	0.6672	0.5969	0.7173	0.6984	**0.7439**	0.5494	0.5793	0.5836	0.5890	0.5837	**0.6011**
AUC	0.8277	0.8608	0.9246	0.9439	0.9692	**0.9798**	0.5639	0.6341	0.6405	0.6412	0.6459	**0.6529**

aspect, the generator generates fraudulent samples, while the camouflage identification module and the fraudster detection module attempt to detect fraudsters. Similarly, by fixing θ_2 and θ_3, the generator optimizes ω by maximizing fraud detection error $\ell_{\omega}^{fraud\_det}$ to adverse the two modules. On the contrary, by fixing ω, the discriminator optimizes θ_2 and θ_3 by minimizing fraud detection error $\ell_{\theta_2,\theta_3}^{fraud\_det}$. In this way, the discriminator and the generator are optimized in an adversarial way.

5 EXPERIMENTS

5.1 Datasets

We conducted experiments with two datasets related to the money-laundering scenario and the spam review scenario, respectively.

5.1.1 Flowe money-laundering dataset. The Flowe money-laundering dataset includes transactional data provided by Flowe, a commercial bank in Europe. It contains 166,534 users and 2,367,000 transaction records for 21 months. According to annotations from experts, 349 users are labeled as launderers. Considering some launderers may be mislabeled, communities with more than 20% launderers are labeled as gangs. Each snapshot graph contains transaction information for one month, each node is associated with a user, and the edge connects two user nodes if they have conducted a transaction during the corresponding month. The weight a_{ij} is specified as the total amount of transactions between two users.

5.1.2 YelpZip review dataset. The YelpZip dataset was collected from Yelp.com and includes 608,598 restaurant reviews. The dataset contains reviews for 5,044 restaurants, written by 260,277 reviewers. Yelp has a filter algorithm that identifies fake or suspicious reviews, and labels legitimate and spam reviews. Different from most existing works [27], we treat users as nodes in the graph and connect them according to the U-PST-U relation mentioned in Section 3. Users with more than 50% spam reviews are labeled as fraudsters, and communities with more than 50% fraudsters are labeled as gangs. a_{ij} is specified as the number of connections between two users. Similar to the Flowe dataset, each snapshot graph G_t contains information for one month.

5.2 Experimental Setup

5.2.1 Evaluate Metrics. In this paper, we formulate the fraud detection task as a binary classification task with imbalanced samples. We use **precision**, **recall**, and **false positive rate (FPR)** for the fraud class to evaluate the performance for the fraud class. We use **Marco-F1** and **AUC** to evaluate the overall performance. F1-score is the trade-off between recall and precision, and Macro-F1 is an unweighted mean of the F1-score for normal users and fraudsters. Some other settings are attached in section D of the Appendix.

5.2.2 Compared Methods. The compared methods could be divided into three groups. The first baseline group utilizes spatio-temporal graph neural networks to capture collusive patterns for fraud detection, overlooking camouflage. The second baseline group contains three state-of-the-art works on handling user-level camouflage. And the third baseline group removes the camouflage generation module from *ACD* to show the effectiveness of the module.

- **Spatio-temporal Graph Neural Network(STGNN)** [2]: We use STGNN, which shares a similar structure as the fraudster detection module, to represent the general spatio-temporal graph neural network.
- **GraphConsis (GC)** [21]:This GNN-based fraud detector considers user-level inconsistency and tries to identify camouflage by a distance-based method with a fixed threshold.
- **CARE-GNN (CARE)** [5]: One of the state-of-the-art methods focuses on identifying user-level camouflage by similarity-based method with an adaptive threshold.
- **FRAUDRE** [40]: Another method focuses on solving the inconsistency problem by considering the distance of hidden representations between two users.
- **ACD without the generator (CD)**: In CD, only the discriminator is used. Without the generator, the sample imbalance problem will harm the robustness of the model.

5.3 Experimental Results and Analysis

As shown in the Table 2, ACD outperforms all the baseline methods introduced in Section 5.2.2. In addition, we test our model with newly registered users as a separate test set, and the results are shown in Table3 in the Appendix. Furthermore, we conduct a case study and a statistical analysis to show the effectiveness of the camouflage generation module and the camouflage identification module.

5.3.1 Case Study. The case demonstrated in Figure 4 is a money-laundering gang in the test set. We use the knowledge provided by experts to classify all transactions of the gang into money-laundering transactions(the top sub-figure) and camouflage transactions(the middle sub-figure). Corresponding to the graph snapshot, each point on the line represents the sum of the transaction amount within a month. We can infer that the gang facilitates two money-laundering activities. The first started in October 2020 and ended in December 2020, and the second started in May 2021 and ended in

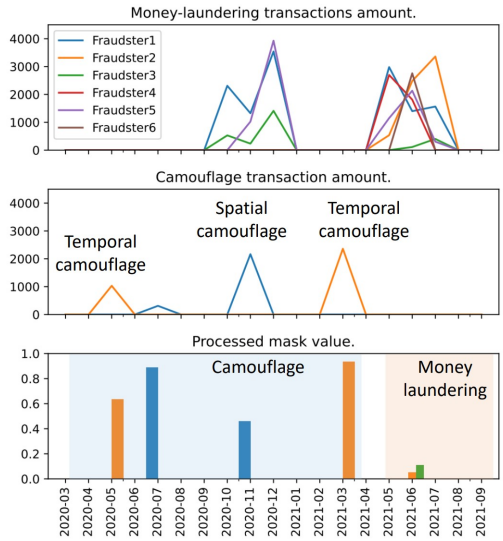


Figure 4: A case study shows the mask values for edges related to a money-laundering gang. To highlight the part which is judged as camouflage by the model, we show the value of 1 - $y_t(\hat{m}_{ij})$, called the processed mask value in the figure. Lines(the top and middle sub-figures) correspond to the transaction amount of users, and bars(the bottom sub-figure) refer to the processed mask value of corresponding edges.

July 2021. The bottom sub-figure shows the processed mask values of all the edges related to the gang. The processed mask values for camouflage edges are large, for other edges, most of the processed mask values are close to 0. This case verifies that the camouflage identification module could accurately identify the camouflage.

5.3.2 Statistical Analysis. To show the effectiveness of the camouflage identification module and camouflage generation module, we further analyze the collusion scores of fraud gangs in different steps. Taking the spatial collusion score as an example, as demonstrated in Figure 5, it is obvious that, for gangs, after adding camouflages, the overall spatial collusion scores drop significantly. Moreover, after the camouflage identification module, the overall spatial collusion scores improve significantly, even larger than that of the original gangs. This phenomenon illustrates the effectiveness of the camouflage generation module and camouflage identification module. According to previous experience with adversarial generative networks, the model tends to lack robustness with such a small number of fraudulent samples. In this paper, by fully considering the property of fraud gangs and designing the objective function, we successfully apply the adversarial generative networks in detecting fraudsters.

6 RELATED WORK

In this section, we summarize the previous works and discuss the connections between existing methods and our proposed *ACD*.

Fraud Detection based on Collusive Patterns. Most existing works focus on mining fraudulent patterns using temporal and spatial information. In particular, graph neural network-based methods [14, 16, 23], network analysis methods [6, 24, 29] and dense sub-graph detection methods [8, 28] could extract the fraudulent spatial patterns. Furthermore, some works have added consideration of temporal information, they introduce the dynamic graph [1, 7, 35] to benefit fraud detection. In this paper, based on the investigations that the majority of fraudulent activities are gang crimes, we summarize the fraudulent patterns as temporal collusive patterns and spatial collusive patterns. Potentially, the above works could mine collusive patterns to effectively detect fraudsters.

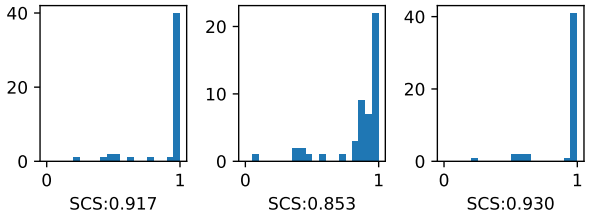


Figure 5: The distribution of spatial collusion scores for gangs in different periods. The left sub-figure is associated with the original fraudulent samples, the middle sub-figure is associated with the generated fraudulent samples. The right sub-figure is associated with all the fraudulent samples after removing camouflage. The SCS refers to the average spatial collusion scores of gangs.

Fraud Detectors against Camouflage. Most existing works identify camouflage by user-level information, such as calculating similarity or distance between two related users in a graph [5, 18, 21], which have already made great progress in the fraud detection scenarios. However, after analyzing many real-world records of fraud gangs, we discovered that fraudsters usually conduct community-level camouflages, which are ignored by most existing works and yield limited capability of recognizing all the camouflage. In this paper, we focus on identifying camouflage with the help of community-level information.

7 CONCLUSION

After investigating many real-world fraud scenarios, in this paper, we figure out that fraudsters always facilitate community-level camouflage, which could be summarized into temporal camouflage and spatial camouflage. Such community-level camouflage can significantly conceal collusive patterns, making gangs appear like benign communities. Toward a more effective work for fraud detection, we propose a generative adversarial network framework, named Adversarial Camouflage Detector, to detect fraudsters. The discriminator is designed based on spatio-temporal graph neural networks and focuses on identifying and removing camouflage by enhancing gangs' collusive patterns and then detecting fraudsters. The generator generates fraudster samples by competing against the discriminator to alleviate the sample imbalance problem. We conducted experiments on a real-world money-laundering dataset and a spam review dataset. The experimental results show that our proposed method outperforms other methods.

REFERENCES

[1] Lei Cai, Zhengzhang Chen, Chen Luo, Jiaping Gui, Jingchao Ni, Ding Li, and Haifeng Chen. 2021. Structural temporal graph neural networks for anomaly detection in dynamic graphs. In *Proceedings of the 30th ACM international conference on Information & Knowledge Management*. 3747–3756.

[2] Dawei Cheng, Sheng Xiang, Chencheng Shang, Yiyi Zhang, Fangzhou Yang, and Liqing Zhang. 2020. Spatio-temporal attention-based neural network for credit card fraud detection. In *Proceedings of the AAAI conference on artificial intelligence*, Vol. 34. 362–369.

[3] Jianfeng Chi, Guanxiong Zeng, Qiwei Zhong, Ting Liang, Jinghua Feng, Xiang Ao, and Jiayu Tang. 2020. Learning to undersampling for class imbalanced credit risk forecasting. In *2020 IEEE International Conference on Data Mining (ICDM)*. IEEE, 72–81.

[4] Antonia Creswell, Tom White, Vincent Dumoulin, Kai Arulkumaran, Biswa Sengupta, and Anil A Bharath. 2018. Generative adversarial networks: An overview. *IEEE Signal Processing Magazine* 35, 1 (2018), 53–65.

[5] Yingtong Dou, Zhiwei Liu, Li Sun, Yutong Deng, Hao Peng, and Philip S Yu. 2020. Enhancing graph neural network-based fraud detectors against camouflaged fraudsters. In *Proceedings of the 29th ACM International Conference on Information & Knowledge Management*. 315–324.

[6] Rafał Drężewski, Jan Sepielak, and Wojciech Filipkowski. 2015. The application of social network analysis algorithms in a system supporting money laundering detection. *Information Sciences* 295 (2015), 18–32.

[7] Yujie Fan, Mingxuan Ju, Shifu Hou, Yanfang Ye, Wenqiang Wan, Kui Wang, Yinming Mei, and Qi Xiong. 2021. Heterogeneous temporal graph transformer: An intelligent system for evolving android malware detection. In *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining*. 2831–2839.

[8] Wenjie Feng, Shenghua Liu, Danai Koutra, Huawei Shen, and Xueqi Cheng. 2020. Specgreedy: unified dense subgraph detection. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*. Springer, 181–197.

[9] Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation* 9, 8 (1997), 1735–1780.

[10] Mengda Huang, Yang Liu, Xiang Ao, Kuan Li, Jianfeng Chi, Jinghua Feng, Hao Yang, and Qing He. 2022. AUC-oriented Graph Neural Network for Fraud Detection. In *Proceedings of the ACM Web Conference 2022*. 1311–1321.

[11] Zhiheng Huang, Wei Xu, and Kai Yu. 2015. Bidirectional LSTM-CRF models for sequence tagging. *arXiv preprint arXiv:1508.01991* (2015).

[12] A Iliev, Nikolay Kyurkchiev, and S Markov. 2017. On the Approximation of the step function by some sigmoid functions. *Mathematics and Computers in Simulation* 133 (2017), 223–234.

[13] Athirai A Irissappane, Hanfei Yu, Yankun Shen, Anubha Agrawal, and Gray Stanton. 2020. Leveraging GPT-2 for classifying spam reviews with limited labeled data via adversarial training. *arXiv preprint arXiv:2012.13400* (2020).

[14] Zhao Li, Haishuai Wang, Peng Zhang, Pengrui Hui, Jiaming Huang, Jian Liao, Ji Zhang, and Jiajun Bu. 2021. Live-streaming fraud detection: a heterogeneous graph neural network approach. In *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining*. 3670–3678.

[15] Chen Liang, Ziqi Liu, Bin Liu, Jun Zhou, Xiaolong Li, Shuang Yang, and Yuan Qi. 2019. Uncovering insurance fraud conspiracy with network learning. In *Proceedings of the 42nd International ACM SIGIR Conference on Research and Development in Information Retrieval*. 1181–1184.

[16] Can Liu, Li Sun, Xiang Ao, Jinghua Feng, Qing He, and Hao Yang. 2021. Intention-aware heterogeneous graph attention networks for fraud transactions detection. In *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining*. 3280–3288.

[17] Xuan Liu, Pengzhu Zhang, and Dajun Zeng. 2008. Sequence matching for suspicious activity detection in anti-money laundering. In *International conference on intelligence and security informatics*. Springer, 50–61.

[18] Yang Liu, Xiang Ao, Zidi Qin, Jianfeng Chi, Jinghua Feng, Hao Yang, and Qing He. 2021. Pick and choose: a GNN-based imbalanced learning approach for fraud detection. In *Proceedings of the Web Conference 2021*. 3168–3177.

[19] Yang Liu, Xiang Ao, Qiwei Zhong, Jinghua Feng, Jiayu Tang, and Qing He. 2020. Alike and unlike: Resolving class imbalance problem in financial credit risk assessment. In *Proceedings of the 29th ACM International Conference on Information & Knowledge Management*. 2125–2128.

[20] 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.

[21] Zhiwei Liu, Yingtong Dou, Philip S Yu, Yutong Deng, and Hao Peng. 2020. Alleviating the inconsistency problem of applying graph neural network to fraud detection. In *Proceedings of the 43rd international ACM SIGIR conference on research and development in information retrieval*. 1569–1572.

[22] Joana Lorenz, Maria Inês Silva, David Aparício, João Tiago Ascensão, and Pedro Bizarro. 2020. Machine learning methods to detect money laundering in the Bitcoin blockchain in the presence of label scarcity. In *Proceedings of the First ACM International Conference on AI in Finance*. 1–8.

[23] Mingxuan Lu, Zhichao Han, Susie Xi Rao, Zitao Zhang, Yang Zhao, Yinan Shan, Ramesh Raghunathan, Ce Zhang, and Jiawei Jiang. 2022. BRIGHT-Graph Neural Networks in Real-time Fraud Detection. In *Proceedings of the 31st ACM International Conference on Information & Knowledge Management*. 3342–3351.

[24] Maryam Mahootiha, Alireza Hashemi Golpayegani, and Babak Sadeghian. 2021. Designing a New Method for Detecting Money Laundering based on Social Network Analysis. In *2021 26th International Computer Conference, Computer Society of Iran (CSICC)*. IEEE, 1–7.

[25] Mark EJ Newman and Michelle Girvan. 2004. Finding and evaluating community structure in networks. *Physical review E* 69, 2 (2004), 026113.

[26] Erlin Pan and Zhao Kang. 2021. Multi-view contrastive graph clustering. *Advances in neural information processing systems* 34 (2021), 2148–2159.

[27] Zidi Qin, Yang Liu, Qing He, and Xiang Ao. 2022. Explainable Graph-based Fraud Detection via Neural Meta-graph Search. (2022).

[28] Yuxiang Ren, Hao Zhu, Jiawei Zhang, Peng Dai, and Liefeng Bo. 2021. Ensemfdet: An ensemble approach to fraud detection based on bipartite graph. In *2021 IEEE 37th International Conference on Data Engineering (ICDE)*. IEEE, 2039–2044.

[29] David Savage, Qingmai Wang, Pauline Chou, Xiuzhen Zhang, and Xinghuo Yu. 2016. Detection of money laundering groups using supervised learning in networks. *arXiv preprint arXiv:1608.00708* (2016).

[30] Ansia Storm et al. 2013. Establishing the link between money laundering and tax evasion. *International Business & Economics Research Journal (IBER)* 12, 11 (2013), 1437–1450.

[31] Vincent A Traag, Ludo Waltman, and Nees Jan Van Eck. 2019. From Louvain to Leiden: guaranteeing well-connected communities. *Scientific reports* 9, 1 (2019), 1–12.

[32] Petar Velickovic, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Lio, and Yoshua Bengio. 2017. Graph attention networks. *stat* 1050 (2017), 20.

[33] Li Wang, Peipei Li, Kai Xiong, Jiashu Zhao, and Rui Lin. 2021. Modeling Heterogeneous Graph Network on Fraud Detection: A Community-based Framework with Attention Mechanism. In *Proceedings of the 30th ACM International Conference on Information & Knowledge Management*. 1959–1968.

[34] Xiao Wang, Nian Liu, Hui Han, and Chuan Shi. 2021. Self-supervised heterogeneous graph neural network with co-contrastive learning. In *Proceedings of the 27th ACM SIGKDD conference on knowledge discovery & data mining*. 1726–1736.

[35] Xuhong Wang, Ding Lyu, Mengjian Li, Yang Xia, Qi Yang, Xinwen Wang, Xinguang Wang, Ping Cui, Yupu Yang, Bowen Sun, et al. 2021. APAN: Asynchronous propagation attention network for real-time temporal graph embedding. In *Proceedings of the 2021 International Conference on Management of Data*. 2628–2638.

[36] Zhuo Wang, Songmin Gu, and Xiaowei Xu. 2018. GSLDA: LDA-based group spamming detection in product reviews. *Applied Intelligence* 48, 9 (2018), 3094–3107.

[37] Mark Weber, Jie Chen, Toyotaro Suzumura, Aldo Pareja, Tengfei Ma, Hiroki Kanezashi, Tim Kaler, Charles E Leiserson, and Tao B Schardl. 2018. Scalable graph learning for anti-money laundering: A first look. *arXiv preprint arXiv:1812.00076* (2018).

[38] Jiaxuan You, Tianyu Du, and Jure Leskovec. 2022. ROLAND: Graph Learning Framework for Dynamic Graphs. In *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*. 2358–2366.

[39] Bing Yu, Haoteng Yin, and Zhanxing Zhu. 2017. Spatio-temporal graph convolutional networks: A deep learning framework for traffic forecasting. *arXiv preprint arXiv:1709.04875* (2017).

[40] Ge Zhang, Jia Wu, Jian Yang, Amin Beheshti, Shan Xue, Chuan Zhou, and Quan Z Sheng. 2021. FRAUDRE: fraud detection dual-resistant to graph inconsistency and imbalance. In *2021 IEEE International Conference on Data Mining (ICDM)*. IEEE, 867–876.

[41] Shijie Zhang, Hongzhi Yin, Tong Chen, Quoc Viet Nguyen Hung, Zi Huang, and Lizhen Cui. 2020. Gcn-based user representation learning for unifying robust recommendation and fraudster detection. In *Proceedings of the 43rd international ACM SIGIR conference on research and development in information retrieval*. 689–698.

[42] Si Zhang, Dawei Zhou, Mehmet Yigit Yildirim, Scott Alcorn, Jingrui He, Hasan Davulcu, and Hanghang Tong. 2017. Hidden: hierarchical dense subgraph detection with application to financial fraud detection. In *Proceedings of the 2017 SIAM International Conference on Data Mining*. SIAM, 570–578.

A DETAILS OF MODULE STRUCTURE

A.1 Camouflage Identification Module

To remove camouflage, the camouflage identification module outputs a mask for each edge related to fraudulent nodes in each snapshot graph. After getting $H_t^l(V)$ from the spatial embedding layer, as shown in Equation 11, we learn a mask embedding $h_t(m_{ij})$ for each edge related to a fraudulent node. To fully utilize information from different time steps, we use a BiLSTM [11] to make mask embeddings of different times $\{h_{t_1}(m_{ij}), ..., h_{t_{p-1}}(m_{ij}), h_{t_{p+1}}(m_{ij}), ..., h_{t_k}(m_{ij})\}$ to enhance each other as Equation 12,

$$h_t(m_{ij}) = mlp_m(h_t^l(v_i) \parallel h_t^l(v_j) \parallel h_t(e_{ij})), \tag{11}$$

$$(h'_{t_1}(m_{ij}), ..., h'_{t_k}(m_{ij})) = BiLSTM((h_{t_1}(m_{ij}), ..., h_{t_k}(m_{ij}))), \tag{12}$$

where $\parallel$ refers to the concatenation operation.

B OPTIMIZATION OBJECTIVES

B.1 Collusion Loss

This section briefly introduces the calculation of collusion loss. Given a gang c_q and time t, the mask matrix can be written as Equation 13, where each element is associated with an output mask $y_t(\hat{m}_{ij})$. The transaction matrix can be written as Equation 14, where each element $x_{e_{ij},t}$ is associated with the total amount of transaction between v_i and v_j at t.

$$m_t(\hat{c_q}) = \begin{bmatrix} y_t(\hat{m}_{11}) & y_t(\hat{m}_{12}) & \cdots & y_t(\hat{m}_{1n_q}) \\ y_t(\hat{m}_{21}) & y_t(\hat{m}_{22}) & \cdots & y_t(\hat{m}_{2n_q}) \\ \vdots & \vdots & \ddots & \vdots \\ y_t(\hat{m}_{n_q1}) & y_t(\hat{m}_{n_q2}) & \cdots & y_t(\hat{m}_{n_qn_q}), \end{bmatrix} \tag{13}$$

$$t_t(c_q) = \begin{bmatrix} x_{e_{11},t} & x_{e_{12},t} & \cdots & x_{e_{1n_q},t} \\ x_{e_{21},t} & x_{e_{22},t} & \cdots & x_{e_{2n_q},t} \\ \vdots & \vdots & \ddots & \vdots \\ x_{e_{n_q1},t} & x_{e_{n_q2},t} & \cdots & x_{e_{n_qn_q},t}. \end{bmatrix} \tag{14}$$

As mentioned in Section 3, without considering camouflage, the temporal collusion score is calculated by Equation2 based on the vector $x_{u,tem} = (x_{u,t_1}, x_{u,t_2}, ..., x_{u,t_k})$, where $x_{u,t} = \sum_{j \in \mathcal{N}_u} x_{e_{uj},t}$. Considering the mask, the vector can be represented as $(\hat{x_{u,t_1}}, \hat{x_{u,t_2}}, ..., \hat{x_{u,t_k}})$, where $\hat{x_{u,t}} = \sum_{j \in \mathcal{N}_u} y_t(\hat{m}_{uj}) * x_{e_{uj},t}$, $\mathcal{N}_u$ refers to the neighbor of u. After the masking operation, the temporal collusion score of c_q can be calculated by Equation 2, written as $\hat{\rho}_{c_q}^t$.

C EXPERIMENTAL RESULTS AND ANALYSIS

Table 3 shows the test result with newly registered users in the test set.

D EXPERIMENTAL DETAILS

D.1 Flowe Money-laundering Dataset

Each node is associated with ten features, including some statistics of transaction information during the month, such as the total internal transaction amount, the total transaction amount with external accounts, the percentage of the amount transferred in, etc. Each edge is associated with ten features, including the number of transactions between two users, the total amount of transactions between two users, etc. We conducted some data filtering based on some rules to reduce the sample imbalance problem.

D.2 YelpZip Spam Review Dataset

Reviews include product and user information, timestamp, ratings, and a plaintext review. Since the graph is sparse, the spatial collusion scores for gangs and benign communities are close to 1. We choose to optimize the collusion score for the other three relations based on the graph constructed with the U-PST-U relation. The construction of the snapshot graph and graph trajectory is similar to the Flowe dataset, and we will not go into details.

D.3 Hyperparameters.

The important hyperparameters tuned in ACD include the number of communities, dimensions of node embedding d_n, edge embedding d_e, and mask embedding d_m. The parameter β is used in calculating the mask in Equation 4. The parameters are used to adjust different losses such as γ_1,γ_2, and so on. Besides, the number of layers in all the RNNs in the temporal embedding layer, batch size, and the learning rate are also tuned. All the hyperparameters are selected based on the performance of the validation set. For the Yelp review dataset, the threshold to label gangs from communities is also selected as a hyperparameter.

D.4 Learning Setting

We conducted a 4-fold method to evaluate the performance of ACD and all baseline methods. If a batch does not contain any fraudulent samples, we will reselect data to generate the batch. The model is optimized until the AUC on the validation set is not improved for 10 epochs. We used some pre-process methods to divide all the users into several independent blocks to speed up the clustering process.

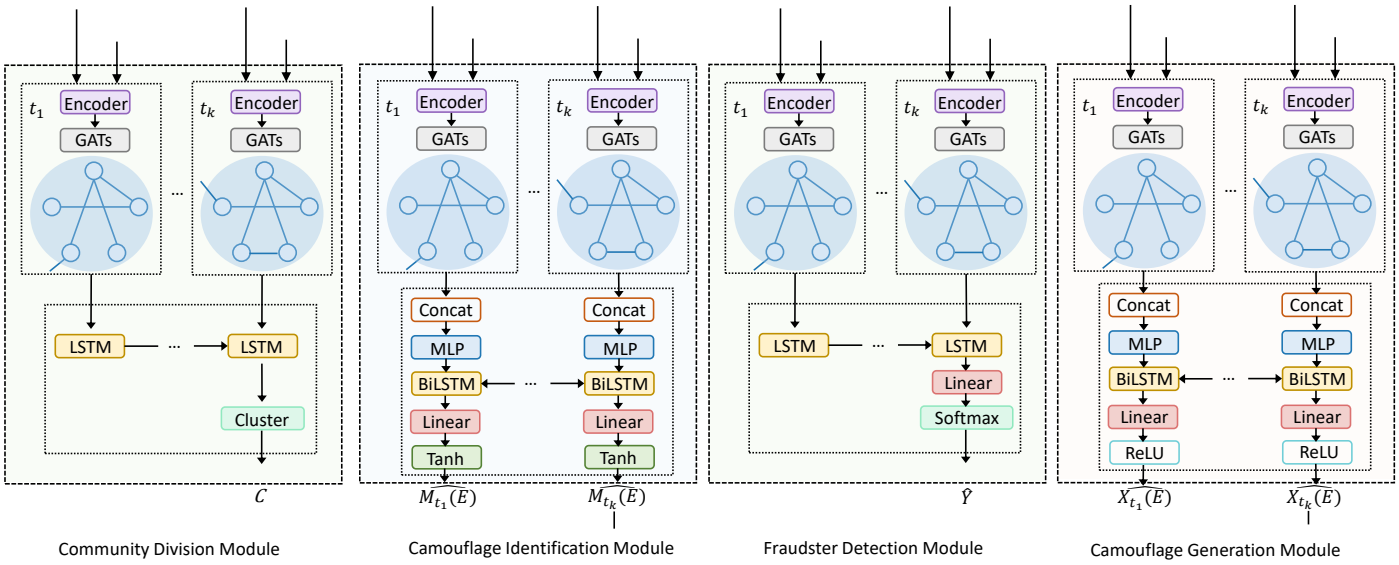


Figure 6: Structure of each module. The encoder includes a node encoder and an edge encoder.

Table 3: Model performance on the newly registered user set.

	Flowe						YelpZip					
	FRAUDRE	CARE	GC	STGNN	CD	ACD	FRAUDRE	CARE	GC	STGNN	CD	ACD
Precision	0.1034	0.1380	0.0690	0.1814	0.1681	**0.2003**	0.4889	0.4540	0.4889	0.4905	0.4952	**0.5201**
Recall	0.4085	0.5579	0.2774	0.7317	0.6494	**0.7988**	0.2707	0.2513	0.2707	0.2724	0.2742	**0.2953**
FPR	0.0088	0.0087	0.0093	0.0082	0.0080	**0.0080**	0.1604	0.1713	0.1604	0.1604	0.1584	**0.1544**
Marco-F1	0.6165	0.6557	0.5795	0.7007	0.6821	**0.7185**	0.5671	0.5487	0.5671	0.5680	0.5704	**0.5847**
AUC	0.8042	0.8438	0.9144	0.9249	**0.9696**	0.9692	0.5939	0.6438	0.6424	0.6376	0.6404	**0.6444**