Crowdsourcing Fraud Detection over Heterogeneous Temporal MMMA Graph

Zequan Xu[1], Qihang Sun[2], Shaofeng Hu[2], Jieming Shi[3], and Hui Li[1⋆]

[1] School of Informatics, Xiamen University
xuzequan@stu.xmu.edu.cn, hui@xmu.edu.cn
[2] Tencent Inc.
aaronqhsun@tencent.com, hugohu@tencent.com
[3] The Hong Kong Polytechnic University
jieming.shi@polyu.edu.hk

Abstract. The rise of the click farm business using Multi-purpose Messaging Mobile Apps (MMMAs) tempts cybercriminals to perpetrate crowdsourcing frauds that cause financial losses to click farm workers. In this paper, we propose a novel contrastive multi-view learning method named CMT for crowdsourcing fraud detection over the heterogeneous temporal graph (HTG) of MMMA. CMT captures both heterogeneity and dynamics of HTG and generates high-quality representations for detection in a self-supervised manner. We deploy CMT on an industry-size HTG of a representative MMMA WeChat and it significantly outperforms other detection methods. CMT also shows promising results for fraud detection on a large-scale public financial HTG, indicating that it can be applied in other graph anomaly detection tasks. We provide our implementation at https://github.com/KDEGroup/CMT.

1 Introduction

Multi-purpose Messaging Mobile Apps (MMMAs) (e.g., WeChat[4] developed by Tencent) integrate several functionalities (e.g., chat, make transactions or book tickets) into one app, attracting billions of users. Due to their popularity, MMMAs have become primary platforms for click farms [3] where *click farm workers* are recruited to complete tasks (e.g., click on YouTube videos to create appearance of popularity). MMMAs are used to hire workers, assign tasks and deliver rewards. However, the rise of the click farm business tempts cybercriminals to perpetrate *crowdsourcing frauds* where victims are click farm workers.

This paper[5] studies crowdsourcing fraud detection (CFD) on WeChat with the permission from Tencent. Fig. 1(a) depicts a typical process of such frauds:

– **Step 1:** Fraudsters send "add friend" requests to potential victims (`ADD` in Fig. 1(a) - Step 1). Chat groups are created and cybercriminals join (`CREATE` and `ENTER` in Fig. 1(a) - Step 1). They disguise as normal users.

⋆ Corresponding Author: Hui Li

[4] https://www.wechat.com/en

[5] More details are in our technical report: https://arxiv.org/abs/2308.02793.

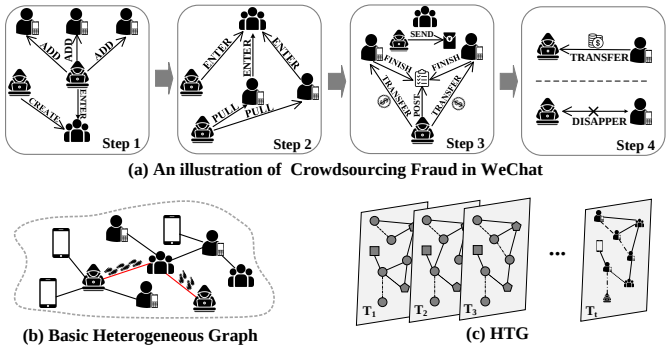

Fig. 1: Crowdsourcing fraud in WeChat.

- **Step 2:** Fraudsters invite victims (`PULL` in Fig. 1(a) - Step 2) to join fraud groups (`ENTER` in Fig. 1(a) - Step 2) by using high reward as bait.
- **Step 3:** Group members are encouraged to complete tasks (`FINISH` in Fig. 1(a) - Step 3) posted by the group owner (`POST` in Fig. 1(a) - Step 3). A typical task is transferring money to top up shopping cards. The cost of the first a few tasks is not high. Fraudsters will pay the commission (`TRANSFER` in Fig. 1(a) - Step 3) and may send bonus packages in the group (`SEND` in Fig. 1(a) - Step 3) as an incentive. Group members can receive a random portion of the bonus.
- **Step 4:** With victims' guard down, fraudsters post new tasks that request much more money. Victims can see that other group members (fraud conspirators) complete and get high rewards. Hence, they are deceived and transfer money to complete new tasks (i.e., `TRANSFER` in Fig. 1(a) - Step 4). After that, fraudsters disappear (i.e., `DISAPPEAR` in Fig. 1(a) - Step 4).

It is natural to model MMMA as a user-user interaction graph which is both *heterogeneous* and *dynamic*: (1) Users can perform diverse operations due to MMMAs' all-in-one functionality; (2) Fraud actions span multiple time points, making the MMMA graph a dynamic graph. Thus, the detection approach should model user behaviors from *multiple views* so that both diverse user interactions and temporal features can be fully leveraged. Moreover, CFD *lacks of supervision* since (1) it is hard to label the huge volume of MMMA users, and (2) user features are limited and accessing private information like chat content is forbidden due to privacy issues. Hence, the detection model should seek additional supervision.

To solve the above issues, we propose Contrastive Multi-view Learning over Heterogeneous Temporal Graph (CMT) for CFD. Our contributions are:

- We propose Heterogeneous Temporal Graphs (HTGs) to model MMMA data.
- We design a novel method CMT for CFD. CMT uses a heterogeneous graph encoder to capture the heterogeneity of HTG. To model dynamics of HTG, CMT constructs two types of user history sequences as two "views" of behavior patterns. CMT further augments each sequence and its contrastive learning encoder encodes sequences in a self-supervised manner.
- We deploy CMT for CFD on an industry-size HTG of WeChat and it significantly outperforms baselines. Additional promising experimental results on a large financial HTG show that CMT can be applied in other graph anomaly detection (GAD) tasks like financial fraud detection.

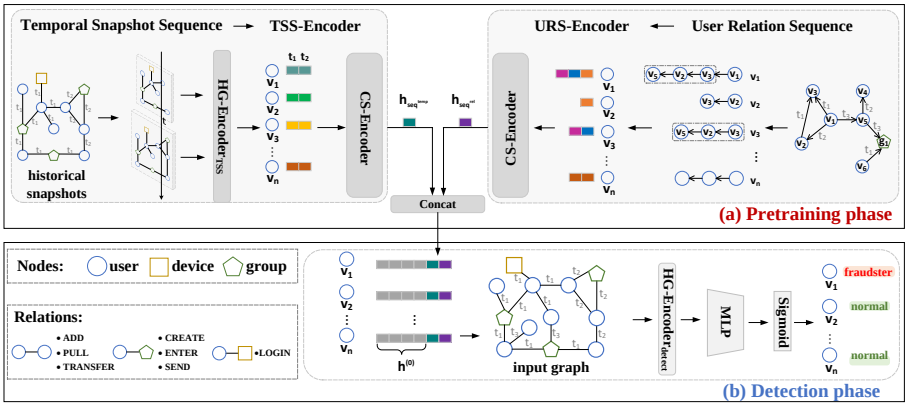

Fig. 2: Overview of CMT.

2 Related Work

Graph-based Anomaly Detection (GAD) detects anomalous graph objects (i.e., nodes, edges or sub-graphs) [7]. Recent GAD methods are mostly inspired by GNNs [13] and self-supervised learning [12]. Anomaly detection in dynamic graphs also attracts increasing interest, since many real-world networks can be generally represented in the form of dynamic graphs. Recent GAD methods on dynamic graphs mostly deploy temporal GNNs or combine GNNs and neural networks for sequential modeling (e.g., GRU) [13,1].

3 Our Framework CMT

3.1 Overview

Fig. 2 depicts CMT. It models MMMA as a Heterogeneous Temporal Graph (HTG, Sec. 3.2) and detects frauds over the HTG. It contains pretraining and detection phases. In pretraining, CMT captures both graph heterogeneity and dynamics: (1) CMT firstly uses a Heterogeneous GNN Encoder (HG-Encoder) to encode heterogeneity (Sec. 3.3); (2) For graph dynamics, CMT constructs two types of user history sequence as two "views" of user behavior patterns (Sec. 3.4.1); (3) Then, CMT generate two augmented sequences for each user history sequence (Sec. 3.4.2) to gain additional supervision; (4) After that, CMT adopts two contrastive learning enhanced encoders (CS-Encoder), namely TSS-Encoder and URS-Encoder, to encode sequences from different views (Sec. 3.4.3). During detection, CMT estimates the abnormality of each user.

3.2 Modeling MMMA Graph as HTG

Firstly, we pre-extract some features $\mathbf{p}_u \in \mathbb{R}^7$ for each MMMA user u in WeChat. *Note that they are chosen through a strict investigation process to protect users' privacy.* Tab. 1 describes 7 binary node attributes used as features:

Table 1: Description for binary node attributes used in CFD for WeChat

ID	Description
1	Was the account registered in the past 90 days?
2	Is the account currently active?
3	Has the account owner verified his/her real identity?
4	Has the account owner changed his/her real identity?
5	Is the account currently banned by any app or third-party services?
6	Was the account reported by other users in the past 14 days?
7	Is the account rented?

– Attributes 1 - 4 describe account "value": WeChat accounts that are registered early, active or having verified and consistent identity are more valuable.
– Attributes 5 - 7 are related to CFD, but they are common features across different anomaly detection tasks in WeChat: (1) normal businessmen may rent multiple accounts to advertise their products in chat groups, and (2) an account can be reported or banned in some services (e.g., games) due to reasons like unfriendly behavior in games or verbal abuse. Hence, only considering these attributes is not sufficient for accurate CFD.

Then, we construct a HTG to model MMMA. We first consider a basic heterogeneous graph (Fig. 1(b)) with three node types (i.e., user, group and device). Device nodes indicate devices that users uses to login, and are uniquely identified by IP address and device types. We include seven relation types: "create a group" (`CREATE`), "join a group" (`ENTER`), "login on a device" (`LOGIN`), "invite someone to join a group" (`PULL`), "send a bonus packet to a group" (`SEND`), "become WeChat friends" (`ADD`), "transfer money" (`TRANSFER`). Based on the basic graph, we further consider temporal dependencies and construct a HTG as a graph stream containing basic heterogeneous graphs from discrete snapshots (Fig. 1(c)).

3.3 Graph Heterogeneity Encoding

For simplicity, in the following, we use the basic heterogeneous graph to illustrate Heterogeneous GNN Encoder (HG-Encoder) for encoding graph heterogeneity.

Each user u's raw features $\mathbf{p}_u$ is projected as the initial embedding: $\mathbf{h}_u^{(0)} = \mathbf{W_h}\mathbf{p}_u$, where $\mathbf{W_h}$ is a learnable matrix. For each group/device node, we aggregate embeddings of its members (i.e., user neighbors) as its initial embeddings:

$$\mathbf{h}_g^{(0)} = \text{mean}(\{\mathbf{h}_v, \forall v \in N_g\}),\ \mathbf{h}_d^{(0)} = \text{mean}(\{\mathbf{h}_v, \forall v \in N_d\}), \tag{1}$$

where $\mathbf{h}_g^{(0)}$ and $\mathbf{h}_d^{(0)}$ are initial embeddings for group node g and device node d, respectively. N_g and N_d denotes neighboring user nodes of g and d in the HTG, respectively. $mean(\cdot)$ indicates average pooling.

The messaging passing mechanism in HG-Encoder is *relation-wise*. Representations of neighbors connected to a user u by the same relation r are aggregated:

$$\begin{aligned}
\mathbf{x}_{N_u^r}^{(k+1)} &= \text{mean}(\mathbf{h}_{r,j_1}^{(k)}, \ldots, \mathbf{h}_{r,j_*}^{(k)}) & \mathbf{h}_{N_u^r}^{(k+1)} &= \mathbf{x}_{N_u^r}^{(k+1)} \oplus \mathbf{y}_{N_u^r}^{(k+1)} \oplus \mathbf{z}_{N_u^r}^{(k+1)} \\
\mathbf{y}_{N_u^r}^{(k+1)} &= \max(\mathbf{h}_{r,j_1}^{(k)}, \ldots, \mathbf{h}_{r,j_*}^{(k)}) & \mathbf{m}_{N_u^r}^{(k+1)} &= \mathbf{W}_{r,m}\mathbf{h}_{N_u^r}^{(k+1)} + \mathbf{b}_{r,m} \\
\mathbf{z}_{N_u^r}^{(k+1)} &= \text{sum}(\mathbf{h}_{r,j_1}^{(k)}, \ldots, \mathbf{h}_{r,j_*}^{(k)})
\end{aligned} \tag{2}$$

where the superscript (k) indicates the k-th iteration, $\oplus$ is the concatenation operation, N_u^r denotes the relation-r-based neighbors of node u and $j_* \in N_u^r$. $\mathbf{h}_{r,j_*}^{(k)}$ is the representation of node j_* for relation r, and $\mathbf{h}_{r,j_*}^{(0)}$ is equivalent to $\mathbf{h}_{j_*}^{(0)}$. $\max(\cdot)$ and $\text{sum}(\cdot)$ are max pooling and sum pooling, respectively. $\mathbf{W}_{r,m}$ and $\mathbf{b}_{r,m}$ are parameters for the relation r.

HG-Encoder adds a self-connection to each user to retain its original features:

$$\mathbf{s}_{r,u}^{(k+1)} = \mathbf{W}_{r,s}\mathbf{h}_{r,u}^{(0)} + \mathbf{b}_{r,s}, \quad \mathbf{e}_{r,u}^{(k+1)} = \text{RELU}(\mathbf{m}_{N_u^r}^{(k+1)} \oplus \mathbf{s}_{r,u}^{(k+1)}) \tag{3}$$

where $\mathbf{W}_{r,s}$ and $\mathbf{b}_{r,s}$ are learnable parameters and $\text{RELU}(\cdot)$ is the Rectified Linear Unit. The acquired $\mathbf{e}_{r,u}$ is passed to a feedforward neural network:

$$\mathbf{q}_{r,u}^{(k+1)} = \text{RELU}(\mathbf{W}_{r,q}\,\mathbf{e}_{r,u}^{(k+1)} + \mathbf{b}_{r,q}), \quad \mathbf{h}_{r,u}^{(k+1)} = \mathbf{q}_{r,u}^{(k+1)} \Big/ \left\| \mathbf{q}_{r,u}^{(k+1)} \right\| \tag{4}$$

where $\mathbf{W}_{r,q}$ and $\mathbf{b}_{r,q}$ are learnable weights.

With R relations, we have R relation-specific embeddings $\{\mathbf{h}_{1,i}^{(k+1)}, \ldots, \mathbf{h}_{R,i}^{(k+1)}\}$ for a user node u. Since one relation-specific embedding only represents the node from one perspective, we further design an inter-relation aggregation module to learn more comprehensive user representations. It uses a relation-level attention mechanism to fuse different relation-wise representations of a user u into its overall representation $\mathbf{h}_u^{(k+1)}$:

$$\mathbf{h}_u^{(k+1)} = \sum_{r=1}^{R} \beta_r \cdot \mathbf{h}_{r,u}^{(k+1)}, \ \beta_r = \frac{\exp(w_r)}{\sum_{i=1}^{R} \exp(w_i)}, \ w_r = \frac{1}{|V|} \sum_{v \in V} \mathbf{a}^T \cdot \tanh(\mathbf{W}_{r,w} \cdot \mathbf{h}_{r,v}^{(k+1)} + \mathbf{b}_{r,w}) \tag{5}$$

where β_r weighs the importance of relation r, $\mathbf{W}_{r,w}$ and $\mathbf{b}_{r,w}$ are learnable weights, and $\mathbf{a}$ denotes the relation-level attention vector.

HG-Encoder stacks two GNN layers (Eqs. 2, 3, 4 and 5) to generate the final representation $\mathbf{h}_u^{(k+1)}$ for user u. Due to the large volume of MMMA data, we only apply messaging passing to user nodes to reduce the training cost. Group/device representations are the average of their neighboring users' representations.

HG-Encoder is connected to a score module containing a linear mapping layer followed by a sigmoid function to estimate user abnormality. HG-Encoder can be optimized with binary cross entropy loss over labeled user nodes. Note that HG-Encoder does not maintain node embeddings binding to specific nodes. Instead, learnable transformation weights $\mathbf{W}$, $\mathbf{b}$ and $\mathbf{a}$ are updated during optimization. Hence, it is *inductive* and can generate representations for new nodes.

3.4 Graph Dynamics Encoding

We design a dynamics encoding method to model HTG dynamics and seek additional supervision. It includes three parts: (1) user history sequence construction, (2) sequence augmentation and (3) contrastive multi-view sequence encoding.

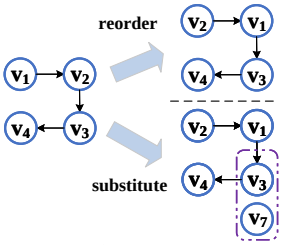

Fig. 3: Augmentation.

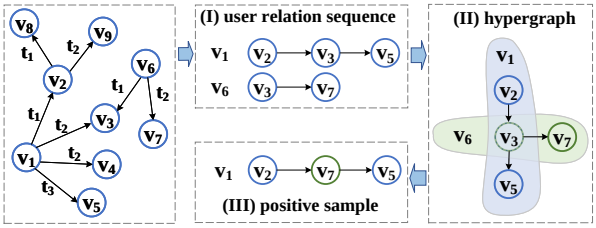

Fig. 4: An example for substitution.

3.4.1 User History Sequences CMT models two types of user history sequences from HTG, *temporal snapshot sequence* and *user relation sequence*, to capture graph dynamics. They provide two different *views* of user historical data.

Temporal Snapshot Sequence. In our observation, *a fraudster's fraud actions may spread across several time points within a short period.* Differently, a normal user typically does not have many actions within a relatively short period. Thus, we extract representations of a user in different snapshots as the temporal snapshot sequence to distinguish normal users and fraudsters. We apply HG-Encoder$_{\text{TSS}}$ (TSS is short for temporal snapshot sequence) over T basic heterogeneous graph in HTG to obtain the representations for all the nodes in each snapshot. HG-Encoder$_{\text{TSS}}$ adopts the encoder design illustrated in Sec. 3.3. For each user node u, its temporal snapshot sequence is $\text{seq}_u^{\text{temp}} = [\mathbf{h}_u^1, \mathbf{h}_u^2, \ldots, \mathbf{h}_u^T]$.

User Relation Sequence. A user's *direct* actions that manifest in its outedges reveal its characteristics. Fig. 1 shows that frauds typically involve several direct actions (e.g., `ADD`, `PULL` and `POST`) spanning different stages of frauds (i.e., search, gain trust and deceive). Thus, it is beneficial to model user relation sequence composed of edges in all the 1-hop subgraphs of a user node from different snapshots of HTG. Given a user node u and its 1-hop out-neighbors nodes $\{n_{u,1}, \ldots, n_{u,m}\}$, u' user relation sequence is $\{(u, n_{u,1}, t_1), \ldots, (u, n_{u,m}, t_m)\}$ where m is the number of out-neighbors of u and a tuple $(u, n_{u,i}, t_i)$ indicates that an edge from u to $n_{u,i}$ at time point t_i. $n_{u,i}$ can be a user node, a group node or a device node. If multiple possible nodes exist for $n_{u,i}$ at t_i, we sample one or more of them as $n_{u,i}$ as long as the number of the extracted user relation sequences is less than a threshold. For any two nodes $n_{u,i}$ and $n_{u,j}$ in the constructed sequence with $i < j$, $(u, n_{u,i}, t_i)$ and $(u, n_{u,j}, t_j)$ satisfy $t_i < t_j$. Some examples are provided in the upper right corner of Fig. 2. *User relation sequences describe user's behaviors over time, which remedies the limitation of temporal snapshot sequences that solely contain hidden states of the same user over time.*

3.4.2 Data Augmentation We design two augmentation approaches *reorder* and *substitute* to generate more meaningful sequences. As depicted in Fig. 3, they can augment the same sequence from different views. For each temporal snapshot sequence, we perform reordering twice to generate two augmented sequences. For each user relation sequence, we generate two augmented sequences of which one is from reordering and the other is from substitution.

Reorder. Fraudsters usually perpetrate crowdsourcing frauds as a gang. A fraudster may `PULL` victims into a group created by a conspirator, and then `CREATE` another group for future deceptions. The two behavior order "`CREATE-PULL`" and "`PULL-CREATE`" are both meaningful in CFD. Based on the above observation, we adopt the reordering operator to produce augmented sequences. Given a reordering ratio γ and a user history sequence s_u, we randomly shuffle a continuous subsequence in s_u with length $\lfloor \gamma * |s_u| \rfloor$ to generate another sequence s'_u. $|s_u|$ indicates the length of s_u and s'_u has the same length (i.e., $|s_u|$) as s_u.

Substitute. Substitution operator uses a similar element that roughly preserves the same information to replace the selected one in the input user relation sequence. We utilize *hypergraph* to pick such similar elements. As shown in Fig. 4, one sequence v_1: $v_2 \rightarrow v_3 \rightarrow v_5$ shares v_3 with another sequence v_6: $v_3 \rightarrow v_7$. They can naturally form a hypergraph with two hyperedges denoted by colored areas in Fig. 4(II). Hyperedges help reveal the hidden connection. For example, v_1 and v_6 are both fraudsters, v_3 is a normal group, and v_7 is a fraud group. In original HTG, v_1 has no direct connection with v_7, which may be the camouflage for escaping detection. One account joining too many groups may alert the system and fraudsters typically leverage multiple accounts for CFD. Using hypergraph, we can find that v_1 and v_7 has a hyperconnection via v_3, which reveals the hidden information. For this case, v_3 is an intermediate group that may be the camouflage, while v_7 is the true ending node (fraud group). Thus, we can replace v_3 with v_7 to generate another sequence $v_2 \rightarrow v_7 \rightarrow v_5$, being a positive view of the original sequence of v_1. More specially, given a substitution ratio δ and a user relation sequence $s_u^{\text{usr}} = [s_u^1, s_u^2, \ldots, s_u^{|s_u|}]$, we randomly select $\lfloor \alpha * |s_u| \rfloor$ nodes. For each element s_u^i in the selected element set and its associate hyperedge set E_i^{hyper}, we randomly select a hyperedge e from E_i^{hyper} and use the element s_u^j connected to s_u^i via e to substitute s_u^i.

3.4.3 Contrastive Sequence Encoder We design a Contrastive Sequence encoder (CS-Encoder) to encode user history sequences. We adopt a position encoding matrix $\mathbf{P} \in \mathbb{R}^{T \times d}$ to preserve the order of a sequence, where T is the time span of HTG. Given an input sequence representation $s = [\mathbf{h}_1, \mathbf{h}_2, \ldots, \mathbf{h}_T]$, the position embeddings are injected to construct the position-aware input: $\mathbf{h}_t^0 = \mathbf{h}_t + \mathbf{p}_t$, where $\mathbf{p}_t$ is the position encoding of the time point t and $1 \leq t \leq T$. $\mathbf{h}_t^0$ is fed into the multi-head self-attention module with eight heads of Transformer to generate the sequence representation. The position encoding layer and the multi-head self-attention module constitute an encoding block. We stack two encoding blocks in CS-Encoder.

Recall that there are two types of user history sequences and we have two encoders in CMT to encode each of them:

– As shown in top left corner of Fig. 2, CMT uses TSS-Encoder for encoding temporal snapshot sequences. TSS-Encoder includes an HG-Encoder$_{\text{TSS}}$ and a CS-Encoder. HG-Encoder$_{\text{TSS}}$ encodes the temporal snapshot sequence of a user u to $\text{seq}_u^{\text{temp}}$, which is later fed to the CS-Encoder. CS-Encoder generates the sequence representation of the temporal snapshot sequence.

– As shown in top right corner of Fig. 2, CMT uses URS-Encoder (URS is short for user relation sequence) for encoding user relation sequences. URS-Encoder adopts the same projection as Eq. 1 to generate node representations for constructing input sequence representations to CS-Encoder. Then, CS-Encoder in URS-Encoder generates the representation of user relation sequence.

Given a mini-batch of N users and one type of their corresponding user history sequence, we have augmented user history sequences of size $2N$. For any user u, we treat the representation $\mathbf{s}_u^a$ of its one augmented sequence as the anchor, and the representation $\mathbf{s}_u^p$ of the other augmented sequence naturally becomes the positive sample. The rest $2(N-1)$ augmented sequences within the same batch are treated as negative samples. The contrastive loss function for each positive pair $\langle \mathbf{s}_u^a, \mathbf{s}_u^p \rangle$ is shown as follows:

$$l(\mathbf{s}_u^a, \mathbf{s}_u^p) = \log \frac{e^{\theta(\mathbf{s}_u^a, \mathbf{s}_u^p)/\tau}}{e^{\theta(\mathbf{s}_u^a, \mathbf{s}_u^p)/\tau} + \sum_{v=1}^{N} \mathbb{1}_{[v \neq u]} (e^{\theta(\mathbf{s}_u^a, \mathbf{s}_v^a)/\tau} + e^{\theta(\mathbf{s}_u^a, \mathbf{s}_v^p)/\tau})} \tag{6}$$

where $\mathbb{1}_{[v \neq u]} \in \{0, 1\}$ is the indicator function that equals 1 if $v \neq u$ otherwise 0, and τ is a temperature parameter. We adopt the cosine similarity to calculate $\theta(u, v) = \mathrm{sim}\big(g(u), g(v)\big)$ where $g(\cdot)$ is a two-layer MLP.

Since we can exchange two views of a user history sequence (i.e., the two augmented sequences) in Eq. 6, we can define another loss by using $\mathbf{s}_u^p$ as anchor. The contrastive learning loss used to optimize TSS-Encoder/URS-Encoder is:

$$\mathcal{L}_{\mathrm{cl}} = \frac{1}{2N} \sum_{i=1}^{N} [l(\mathbf{s}_u^a, \mathbf{s}_u^p) + l(\mathbf{s}_u^p, \mathbf{s}_u^a)]. \tag{7}$$

We train both TSS-Encoder and URS-Encoder using a binary classification task (i.e., estimate whether the source of a sequence is a fraudster) together with the contrast task. For the binary classification task, TSS-Encoder/URS-Encoder is connected to a score module defined as a linear mapping layer followed by a sigmoid function. The suspicious score is used to calculate the binary cross-entropy loss $\mathcal{L}_{\mathrm{binary}}$ over limited labeled data. The overall loss for optimizing TSS-Encoder/URS-Encoder is defined as a multi-tasking loss: $\mathcal{L} = \mathcal{L}_{\mathrm{binary}} + \mathcal{L}_{\mathrm{cl}}$.

3.5 Putting All Together

In pretraining, TSS-Encoder, URS-Encoder and the detector HG-Encoder$_{\mathrm{detect}}$ are trained independently on training data with limited labels. HG-Encoder$_{\mathrm{detect}}$ uses the same design illustrated in Sec. 3.3 followed by the same score module as used in the binary classification task of TSS-Encoder/URS-Encoder.

During detection, for a user u, we generated temporal snapshot sequence representation $\mathbf{h}_{\mathrm{seq}_u^{\mathrm{temp}}}$ and user relation sequence representation $\mathbf{h}_{\mathrm{seq}_u^{\mathrm{rel}}}$ using pretrained TSS-Encoder and URS-Encoder, respectively. Then, as shown in Fig. 2(b), the two generated representations and the initial sequence representation $\mathbf{h}_u^{(0)}$ are concatenated and fed into HG-Encoder$_{\mathrm{detect}}$ followed by its scoring module to estimate the suspicious score of u. If the estimate score for u is larger than 0.5, u will be predicted as a fraudster.

4 Experiments

4.1 Experiment Setting

Data. We use two industry-size datasets for our experiments.

- **WeChat Dataset:** It contains 6.8 million user nodes, 151 thousand WeChat chat group nodes, 126 thousand device nodes and 29.7 million edges covering 7 relations introduced in Sec. 3.2. Only 53,660 user nodes are manually labeled by human experts: 10,749 of them are fraud users and 42,911 are normal users. We use one day as the time interval between two time point and we derive 14 separate graph snapshots for HTG. The maximum number of sampled user relation sequence for each user is 10. We randomly divide the labeled users by a ratio of 8:1:1 for training, validation and test.
- **FinGraph Dataset**[6]**:** It contains an anonymized dynamic user-user interaction graph in financial industry. It has approximately 4.1 million user nodes and 5 million edges. Every node has 17 features. Edges have 11 types. There are 82 thousand labeled user nodes: 1 thousand nodes are fraudsters and 81 thousand nodes are normal users. We randomly divide the labeled users by a ratio of 8:1:1 for training, validation and test.

Baselines. *We find that most GAD methods cannot handle an industry-size MMMA graph. We choose the following scalable baselines:*

- **Non-GNN classification methods**: XGBoost [2] and MLP. MLP is a feed-foward neural network with three hidden layers and it has 128, 64, 32 neurons in each layer, respectively.
- **Homogeneous graph based methods**: GCN [4] and GAT [11].
- **Heterogeneous graph based methods**: HG-Encoder illustrated in Sec. 3.3, RGCN [10] and Simple-HGN [6].
- **Graph based anomaly detection methods**: DCI [12] and GeniePath [5]. DCI adopts contrastive learning for GAD.
- **Temporal GAD methods**: AddGraph [13]. AddGraph-H replaces GCN in AddGraph with RGCN to model the heterogeneous information.

We adopt the same score module as CMT for baselines without a score module. All methods adopt Adam optimizer. Representation dimension and batch size are set to 64 and 256, respectively. Reordering and substitution ratios γ and α are 0.4. All methods are terminated when they converge.

Evaluation Metrics. We use AUC and KS as evaluation metrics [9].

4.2 Overall Detection Results.

Tab. 2 presents the overall results. We analyze the experimental results as follows:

1. GNN-based approaches GCN and GAT generally exceed non-GNN methods XGBoost and MLP, indicating the spatial dependencies depicted by graph structure contain rich information that can improve detection performance.

[6] `https://ai.ppdai.com/mirror/goToMirrorDetailSix?mirrorId=28`

Table 2: Overall detection performance. Results of CMT and best baselines are shown in bold.

	WeChat		FinGraph	
Method	**AUC**	**KS**	**AUC**	**KS**
XGBoost	0.7189	0.3281	0.7388	0.3911
MLP	0.7188	0.3404	0.7404	0.4072
GCN	0.8082	0.4790	0.7530	0.4205
GAT	0.7967	0.4591	0.7762	0.4496
RGCN	0.8400	0.5361	0.8002	0.5217
Simple-HGN	0.8484	0.5474	0.7798	0.4669
GeniePath	0.8234	0.5257	0.8115	0.5466
DCI	0.8328	0.5397	0.7737	0.4440
HG-Encoder	**0.8682**	**0.5905**	**0.8194**	**0.5485**
AddGraph	0.8221	0.4924	0.7488	0.3818
AddGraph-H	0.8452	0.5365	0.8013	0.5236
CMT	**0.9014**	**0.6624**	**0.8354**	**0.5720**

Table 3: Results of the ablation study. Best results are in bold.

	WeChat		FinGraph	
Method	**AUC**	**KS**	**AUC**	**KS**
HG-Encoder	0.8682	0.5905	0.8194	0.5485
CMT_{TSS}	0.8853	0.6169	0.8233	0.5554
$\text{CMT}_{\text{TSS}_{\text{cl}}}$	0.8957	0.6461	0.8331	0.5683
CMT_{URS}	0.8911	0.6337	0.8267	0.5554
$\text{CMT}_{\text{URS}_{\text{cl}}}$	0.8929	0.6347	0.8298	0.5625
$\text{CMT}_{\text{TSS}-\text{URS}}$	0.8889	0.6316	0.8312	0.5692
$\text{CMT}_{\text{TSS}_{\text{cl}}-\text{URS}}$	0.8999	0.6624	**0.8355**	0.5692
$\text{CMT}_{\text{TSS}-\text{URS}_{\text{cl}}}$	0.8946	0.6413	0.8324	0.5703
$\text{CMT}_{\text{TSS}_{\text{cl}}-\text{URS}_{\text{cl}}}$	**0.9014**	**0.6624**	0.8354	**0.5720**

2. HG-Encoder significantly exceeds other GNN-based methods. This observation supports our decision of using HG-Encoder as the backbone of CMT.
3. Both dynamic and heterogeneous graph based models achieve satisfactory results and heterogeneous graph based methods generally outperform homogeneous graph based approaches, showing that temporal dependencies and multi-relation information help model user behavior pattern better in CFD.
4. CMT significantly outperforms baselines including state-of-the-art dynamic GAD methods AddGraph and AddGraph-H. It consistently exceeds all baselines on two datasets w.r.t. two metrics. The results show that *CMT is superior to baselines on CFD and can also be applied in other GAD tasks.*

4.3 Ablation Study.

To verify the contribution of each component in CMT, we report results of different variations of CMT in Tab. 3:

– **HG-Encoder**: It only uses Heterogeneous GNN encoder.
– **Variations with the subscript TSS**: It removes URS-Encoder and uses TSS-Encoder only.
– **Variations with the subscript URS**: It removes TSS-Encoder and uses URS-Encoder only.
– **Variations with the subscript TSS-URS**: Both TSS-Encoder and URS-Encoder are used.
– **Smaller subscript "cl" for either TSS or URS**: The contrastive loss $\mathcal{L}_{\text{cl}}$ is used together with $\mathcal{L}_{\text{binary}}$ for optimizing TSS-Encoder or URS-Encoder. For the subscript TSS or URS without the smaller subscript "cl", only $\mathcal{L}_{\text{binary}}$ is used for optimizing TSS-Encoder or URS-Encoder.

From the results in Tab. 3, we can observe that:

1. Most modules bring promising performance gain. The incorporation of either TSS Encoder or USR Encoder brings performance gain, as both CMT-TSS

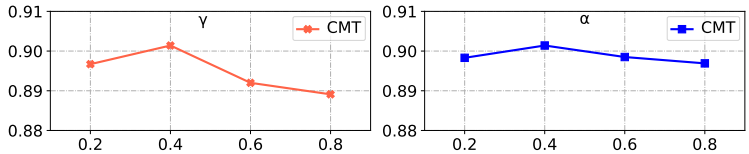

Fig. 5: Sensitivity analysis of γ and α on WeChat dataset.

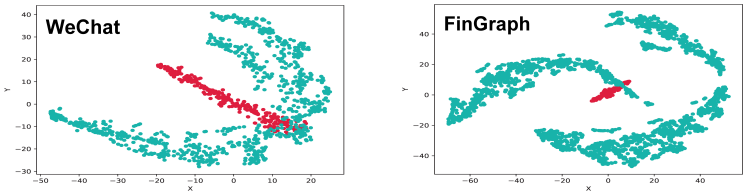

Fig. 6: t-SNE projection of user node representations generated by CMT: (1) Red: fraudsters. (2) Green: normal users.

and CMT-USR significantly outperform HG-Encoder. In a few cases (e.g., compare $\text{CMT}_{\text{TSS-URS}}$ and $\text{CMT}_{\text{TSS}_{\text{cl}}-\text{URS}}$), some modules do not significantly improve the results on one dataset, but they show obvious enhancement on the other dataset.

2. The complete CMT ($\text{CMT}_{\text{TSS}_{\text{cl}}-\text{URS}_{\text{cl}}}$) generally shows best results. For the case where it is not the best, the performance gap is subtle, showing the effectiveness of modeling two views of historical data together.

Overall, we conclude that each module in CMT contributes to its performance.

4.4 Sensitivity analysis of γ and α.

Fig. 5 reports AUC of CMT on WeChat when changing γ and α. We can see that changes of γ and α do not significantly affect the performance of CMT. And the resulting AUC performance is around 0.89-0.9. The observation shows that our data augmentation method is not sensitive to augmentation hyper-parameters.

4.5 Quality of Representation.

We adopt t-SNE [8] to project representations of user nodes in the test set into a 2-dimensional space. From Fig. 6, we can see that representations of normal users and fraudsters have a clear distinction, showing that CMT is able to produce high-quality representations for CFD.

5 Conclusion

In this paper, we present CMT for detecting crowdsourcing fraud. We adopt the idea of data augmentation, multi-view learning and contrastive learning when designing CMT. CMT can capture the heterogeneity and the dynamics of MMMA

data from different views and alleviate label reliance. Experiments demonstrates the effectiveness of CMT. In the future, we plan to further improve the design of CMT by leveraging the fraud patterns that CMT helps to discover.

Acknowledgments

Hui Li is supported by National Science and Technology Major Project (No. 2022ZD0118201), National Natural Science Foundation of China (No. 62002303, 42171456) and 2021 Tencent WeChat Rhino-Bird Focused Research Program. Jieming Shi is supported by Hong Kong RGC ECS (No. 25201221) and Natural Science Foundation of China (No. 62202404).

References

1. Cai, L., Chen, Z., Luo, C., Gui, J., Ni, J., Li, D., Chen, H.: Structural temporal graph neural networks for anomaly detection in dynamic graphs. In: CIKM. pp. 3747–3756 (2021)
2. Chen, T., Guestrin, C.: Xgboost: A scalable tree boosting system. In: KDD. pp. 785–794 (2016)
3. Guardian, T.: How low-paid workers at 'click farms' create appearance of online popularity. `https://www.theguardian.com/technology/2013/aug/02/click-farms-appearance-online-popularity` (2013)
4. Kipf, T.N., Welling, M.: Semi-supervised classification with graph convolutional networks. In: ICLR (2017), `https://openreview.net/forum?id=SJU4ayYgl`
5. Liu, Z., Chen, C., Li, L., Zhou, J., Li, X., Song, L., Qi, Y.: Geniepath: Graph neural networks with adaptive receptive paths. In: AAAI. pp. 4424–4431 (2019)
6. Lv, Q., Ding, M., Liu, Q., Chen, Y., Feng, W., He, S., Zhou, C., Jiang, J., Dong, Y., Tang, J.: Are we really making much progress?: Revisiting, benchmarking and refining heterogeneous graph neural networks. In: KDD. pp. 1150–1160 (2021)
7. Ma, X., Wu, J., Xue, S., Yang, J., Zhou, C., Sheng, Q.Z., Xiong, H., Akoglu, L.: A comprehensive survey on graph anomaly detection with deep learning. IEEE Trans. Knowl. Data Eng. **35**(12), 12012–12038 (2023)
8. van der Maaten, L., Hinton, G.: Visualizing data using t-sne. J. Mach. Learn. Res. **9**, 2579–2605 (2008)
9. Massey, F.J.: The kolmogorov-smirnov test for goodness of fit. Journal of the American Statistical Association **46**(253), 68–78 (1951)
10. Schlichtkrull, M.S., Kipf, T.N., Bloem, P., van den Berg, R., Titov, I., Welling, M.: Modeling relational data with graph convolutional networks. In: Gangemi, A., Navigli, R., Vidal, M., Hitzler, P., Troncy, R., Hollink, L., Tordai, A., Alam, M. (eds.) ESWC. pp. 593–607 (2018)
11. Velickovic, P., Cucurull, G., Casanova, A., Romero, A., Liò, P., Bengio, Y.: Graph attention networks. In: ICLR (2018), `https://openreview.net/forum?id=rJXMpikCZ`
12. Wang, Y., Zhang, J., Guo, S., Yin, H., Li, C., Chen, H.: Decoupling representation learning and classification for gnn-based anomaly detection. In: SIGIR. pp. 1239–1248 (2021)
13. Zheng, L., Li, Z., Li, J., Li, Z., Gao, J.: Addgraph: Anomaly detection in dynamic graph using attention-based temporal GCN. In: IJCAI. pp. 4419–4425 (2019)