STM: A Spatio-Temporal Model for Dynamic Graph Fraud Detection

Yuxiang Wang[1], Runhuai Chen[1⋆], Bin Chen[1⋆⋆✉], Zhiyuan Yu[1⋆⋆⋆], Yuxiang Wang[1], and Tianxing Wu[2†]

[1] Hangzhou Dianzi University, Hangzhou, China
[2] Southeast University, Nanjing, China

Abstract. Graph Neural Network (GNN) models are increasingly being applied in fraud detection, which involves identifying fraudsters in graphs. GNN-based methods on static graphs ignore the important temporal information, resulting in low accuracy in time-sensitive applications. GNN-based methods on dynamic graphs take into account the temporal information but overlook the influence differences between long-term and short-term temporal information, limiting detection accuracy. To address these issues, we propose STM: a Spatio-Temporal Model for fraud detection on dynamic graphs. We split original graphs into a set of snapshot graphs based on time frames. From intra-perspective, in each snapshot graph, we present a differential aggregation method based on spatio-temporal distance to effectively aggregate both the spatio and short-term temporal information. From the inter-perspective, we capture the long-short term temporal information among all snapshot graphs based on self-attention mechanism. Extensive experiments on real-world datasets demonstrate that our method effectively and efficiently detects fraudsters in dynamic graphs. Our code can be found at https://anonymous.4open.science/r/STM/.

Keywords: Fraud detection · Spatio-temporal model · Graph neural network.

1 Introduction

Recently, with the rapid growth of the Internet, various forms of fraudulent activities, such as review fraud [26], financial transaction scams [8], and malicious websites [27], have become increasingly prevalent. Detecting and preventing these activities is crucial for reducing social fraud and avoiding financial losses. In real-world scenarios, entities are often interconnected through interactive relationships that can be represented as graphs. In a financial transaction network, users and merchants can be modeled as nodes, while transactions

⋆ crh@hdu.edu.cn
⋆⋆ hducb@hdu.edu.cn
⋆⋆⋆ zy_yu@hdu.edu.cn
† tianxingwu@seu.edu.cn

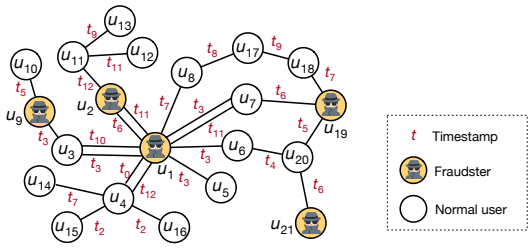


Fig. 1: A dynamic transaction network

are represented by edges. This rich graph-based information is highly valuable for identifying fraudulent entities, thereby Graph Neural Network (GNN)-based fraud detection has emerged as a significant research direction [15, 1, 5, 19, 17].

Existing solutions. According to the graph types, static or dynamic, GNN-based fraud detection has two categories.

GNN-based fraud detection on static graphs. This line of researches, such as PC-GNN [12], CARE-GNN [5], and H2-FDetector [17], leverage original graph structures to pass spatio information from neighbors to the target nodes, enriching the representations of target nodes. However, they only consider the spatio information but ignore the important temporal information (i.e., dynamic changes of the graph topology), resulting in low accuracy in time-sensitive applications [25], such as online financial fraud alert on transaction networks. Figure 1 illustrates a transaction network modelled as a dynamic graph. Each node represents a user (normal or fraudster) of a financial platform, such as Alipay, and each edge represents a financial transaction between two users that is happened at a certain timestamp t. It's evident that not only the users involved in fraud activities but also the timing of these activities are crucial for fraud detection. This motivates the research on GNN-based fraud detection on dynamic graphs.

GNN-based fraud detection on dynamic graphs. Recent studies have proposed several models for this line of researches. TGAT [25] introduced a functional time encoding technique to embed the temporal information into learned node representations. It overlooks the influence differences between long-term and short-term temporal information and treat them equally in fraud detection. In fact, both long-term and short-term temporal information are critical and play different role in identifying fraudsters [1], inspiring that we should consider them separately. FTM [1] proposed a frame-level timeline modelling method that can capture the short-term temporal information of a graph within each time frame, thereby achieving better short-term temporal feature representation. It still has two drawbacks. First, given a target node u and its neighbors denoted by $N(u)$, only few nodes in $N(u)$ are important for identifying if u is a fraudster, thereby aggregating features of all nodes in $N(u)$ to u would introduce a lot of noise. Second, prior studies claim that fraudsters tend to be surrounded by more normal users [9, 6, 26], aggregating features from more normal users would make fraudsters more similar to normal users, thereby resulting in incorrect predictions.

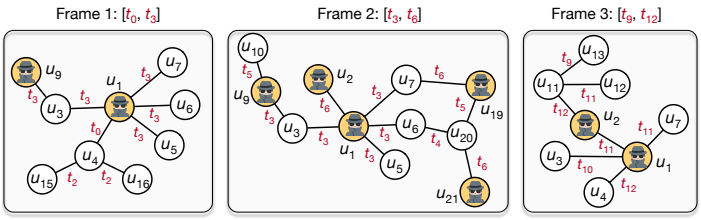


Fig. 2: Three snapshot graphs of the transaction network in Figure 1, within range of time frame $[t_0, t_3]$ (left), $[t_3, t_6]$ (middle), and $[t_9, t_{12}]$ (right)

Challenges and our solutions. We conclude challenges faced by GNN-based fraud detection methods on dynamic graphs and outline our solutions.

Challenge 1: *How to incorporate both long-term and short-term temporal information to facilitate fraud detection?* Recent studies [3, 28, 13] claim that fraudsters typically aim to maximize their gains within the shortest time possible. Consequently, they often conduct a significant number of fraudulent activities in a short period. Therefore, capturing the features within a short range of timestamps (i.e., short-term temporal features) is important. On the other hand, [24, 2] claim that the consistency of user behavior can be assessed by analyzing long-term features, and if a user's behavior pattern undergoes a sudden and significant change, it could be an indicator of fraud activities. In a nutshell, both the long-term and short-term temporal information are critical for fraud detection. To incorporate them together in one model, in this paper, we propose STM: a Spatio-Temporal Model for fraud detection on dynamic graphs. Specifically, we split original graphs into a set of snapshot graphs based on time frames and aim to capture long-term and short-term temporal information from two perspectives. For example, in Figure 2, we illustrate three snapshot graphs within a range of time frame $[t_0, t_3]$, $[t_3, t_6]$, and $[t_9, t_{12}]$, respectively. From intra-frame perspective, in each snapshot graph, we propose the intra-frame aggregation to enhance the fraud features within a short period by aggregating temporal information from a target entity's localized neighborhood. From the inter-frame perspective, we present the inter-frame aggregation to enhance the historical fraud features across the three snapshots based on self-attention mechanism.

Challenge 2: *How to handle the issues of noise features and homogeneity features when aggregating features from neighborhood?* The key for handling the issues is to increase the contributions of valuable neighbors to fraud detection and maintain the feature differences of fraudsters and normal users. Prior study [22] claims that not only the temporal information but also the multi-hop spatial information are benefit for distinguishing which neighbors are important. To reduce the impact of noise features, we present a spatio-temporal distance to measure the importance of each neighbor to fraud detection, enabling aggregation from selected top-k neighbors rather than all neighbors. For example, in Figure 2 (middle part), although node u_3 is closer to the target node u_1 than the 2-hop node u_{20}, u_{20} is more important to u_1 as its active period $[t_4, t_6]$ is close to the active period $[t_3, t_6]$ of u_1, while u_3 only active at time t_3. To mitigate the impact of homogeneity features, we present a differential aggregation to

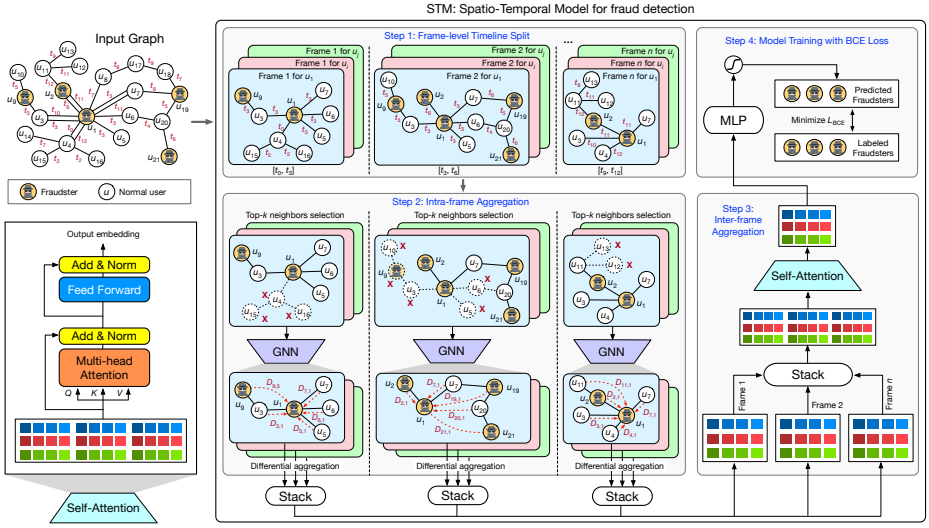


Fig. 3: The framework of the proposed Spatio-Temporal Model (STM) for fraud detection on dynamic graphs, including four steps discussed in §3.3-§3.5

aggregate the feature differences between neighbors and the target node rather than the whole features of neighbors, thereby maintaining the distinct features of fraudster. For example, in Figure 2 (left part), the fraudster u_1 is surrounded by five normal users, the more iterations of differential aggregation from them to u_1, the larger the feature differences between u_1 and normal users are retained.

Contributions. Our main contributions are as follows:

- We propose STM: a Spatio-Temporal Model for fraud detection on dynamic graphs (§3.1), which considers short-term and long-term temporal information in one model via intra-frame and inter-frame aggregation, respectively.
- For intra-frame aggregation, we present a differential aggregation based on the spatio-temporal distance to effectively capture the short-term temporal features from neighborhood (§3.3). For inter-frame aggregation, we leverage self-attention to enhance the historical fraud features across all frames (§3.4).
- Experiments on real-world datasets demonstrate the superiority of STM on both effectiveness and efficiency of fraud detection on dynamic graphs (§4).

2 Preliminary

In this section, we first define dynamic graph and then introduce the fraud detection problem on dynamic graphs.

Dynamic graph. A dynamic graph is defined as $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathcal{X}, \mathcal{Y})$. (1) $\mathcal{V} = \{v_1, \cdots, v_{|\mathcal{V}|}\}$ is the node set consisting of $|\mathcal{V}|$ nodes. (2) Each node $v_i \in \mathcal{V}$ has a feature vector $\boldsymbol{x}_i \in \mathcal{X} \in \mathbb{R}^{|\mathcal{V}| \times d}$ and a node label $y_i \in \mathcal{Y}$ that indicates a node's type. (3) $\mathcal{E} = \{e_1, \cdots e_{|\mathcal{E}|}\}$ is the edge set consisting of $|\mathcal{E}|$ edges. (4) Each edge $e \in \mathcal{E}$ has a feature vector $\boldsymbol{x}_e$ and a timestamp t that indicates when e exists.

In the scenario of fraud detection on dynamic graphs, each node is either a *benign* or *fraud* node that is often represented by a binary label $y \in \{0, 1\}$, where

$y = 0$ indicates a benign node while $y = 1$ indicates a fraud node. Besides, each edge represents a normal or fraudulent activity that happens at time t.

Fraud detection on dynamic graphs. Given a dynamic graph $\mathcal{G}$, it involves spatio information reflected by the original topology, temporal information provided as timestamps of edges, semantic information represented by node feature vectors, and label information associated on each node. The objective of fraud detection on dynamic graphs is to first use spatio and temporal information to update node feature vectors, and then use the node features vectors to determine whether a target node v in $\mathcal{G}$ is a fraudster. Since each node is labeled as either a fraud node or a benign node, fraud detection can be viewed as a binary node classification problem. Specifically, we expect to predict v's label as $\hat{y}$ and aim to ensure it is the exactly same as the original label y, i.e., $\hat{y} = y$.

Remark. It is worth mentioning that the node feature vectors $\mathcal{X}$ may be missing due to the privacy protection or other reasons. In this case, we still can use the position encoding [21] or neighborhood feature vectors [23] as instead.

3 The Proposed Approach

In this section, we present STM, a *Spatio-Temporal Model* for fraud detection on dynamic graphs, including four steps discussed below.

3.1 STM Overview

Figure 3 illustrates the framework of the proposed STM, which consists of four steps. In step ① (*Frame-level Timeline Split*, **§3.2**), we split the original dynamic graph into a set of frame-level snapshot graphs, thereby explicitly distinguishing between the short-term temporal information within a short period that is preserved in each frame and the long-term temporal information that is preserved in the evolution process across all frames. Next, in step ② (*Intra-frame Aggregation*, **§3.3**), we present a differential aggregation method based on spatio-temporal distance to effectively aggregate both the spatio and short-term temporal information via GNN within each frame. In step ③ (*Inter-frame Aggregation*, **§3.4**), we capture the historical fraud features across all frames via self-attention mechanism. Recalling Figure 2, we note that the fraudulent activities initiated by u_1, targeting u_3 and u_7, persisted across three frames, making it necessary to emphasize the influence of u_3 and u_7 throughout the entire time period when determining whether u_1 is a fraudster. Finally, in step ④ (*Model Training*, **§3.5**), we adjust the leaned parameters in steps ② and ③ (i.e., GNN and self-attention component) by minimizing the binary cross-entropy (BCE) loss, ensuring that the fraud classification results are close to the labeled ground-truth.

3.2 Frame-level Timeline Split

Considering a target node u, intuitively, those nodes that are spatial-close to u and have similar activity periods to u are helpful in determining whether u is fraudster. With this in mind, we present a frame-level timeline split method to extract different frames of a certain target node u from u's neighborhood.

Specifically, we first obtain an h-hop neighborhood subgraph of u as the source graph for frame-level split, denoted by $\mathcal{G}_h[u] \subseteq \mathcal{G}$, where every node in $\mathcal{G}_h[u]$ is at most h-hop away from u. Next, given a specific time period $T = [t_{\text{start}}, t_{\text{end}}]$, we extract a snapshot graph $\mathcal{G}_h^T[u] \subseteq \mathcal{G}_h[u]$ as a frame of u within $[t_{\text{start}}, t_{\text{end}}]$, where every edge in $\mathcal{G}_h^T[u]$ has a timestamp $t \in [t_{\text{start}}, t_{\text{end}}]$. By varying the time period given the target node u unchanged, we can obtain different frames of u that reflects its behavioral features over various short-term periods. Conversely, by varying the target node given the time period T unchanged, we can capture the behavioral differences between different nodes at the same short-term period.

Example 1 *In step 1 of Figure 3, given the target node as u_1 (the fraudster in the middle), $h = 3$, and various time periods $T_1 = [t_0, t_3]$, $T_2 = [t_3, t_6]$, and $T_3 = [t_9, t_{12}]$, we extract three frames for u_1 that are illustrated in the blue rectangles, reflecting the different behavioral features of u_1 in three periods. By varying the target node as any other u_i and u_j in $\mathcal{G}$, we can obtain the corresponding frames for u_i and u_j, shown in the red and green rectangles, respectively.*

3.3 Intra-frame Aggregation

Given a frame $\mathcal{G}_h^T[u]$ of a target node u within the period of T, the objective of intra-frame aggregation is to refine the feature vector $\boldsymbol{x}_u$ of u, by utilizing the spatial and short-term temporal information inherited in $\mathcal{G}_h^T[u]$. A straightforward way is to leverage original graph structures to pass node features from neighbors to u. However, this is problematic as it suffers from the issues of noisy features and feature homogeneity (as claimed in Challenge 2 of §1). To overcome the issue of noisy features, we present a top-k neighbors selection based on spatio-temporal distance (S-T distance). While for the issue of feature homogeneity, we propose a differential aggregation over the selected top-k neighbors.

Top-k neighbors selection based on S-T distance. We propose the concept of spatio-temporal distance (S-T distance) and then calculate the S-T distance between neighbors of each target node in a certain time frame $\mathcal{G}_h^T[u]$ as follows.

- Spatial distance $d_s(u, v)$ is the shortest distance between u, v in a graph $\mathcal{G}$.
- Temporal distance $d_t(u, v) = |t_u - t_v|$ is the difference on the temporal scale, where t_u (t_v) represents the largest timestamp among all timestamps associated on u's (v's) adjacency edges.
- S-T distance $d_{st} = d_s + M \cdot d_t$ is the combination of d_s and d_t, where $M > 0$ could be any real number to balance the two distances.
- S-T k-closest neighbor set $N(v, k)$ refers the set of the k closest neighbors to node v in terms of S-T distance $d_{st}(u, v)$.

We next enumerate each node in $\mathcal{G}_h^T[u]$ (excluding the target node u itself) to obtain their S-T distances to u and maintain the S-T k-closest neighbor set $N(u, k)$ for the next step of spatio-temporal differential aggregation on $N(u, k)$.

Example 2 *In Step 2 of the first frame in Figure 3, with $M = 2$, $k = 5$, and $h = 2$, the S-T distance between u_1 and u_4 is $d_{st}(u_1, u_4) = 1 + 2 \cdot (3 - 0) = 7$, while the S-T distances for $\{u_3, u_5, u_6, u_7, u_9, u_{15}, u_{16}\}$ are 1, 1, 1, 1, 2, 4,*

and 4, respectively. Although u_4 has a $d_s = 1$ from u_1, it is discarded due to a large $d_t = 6$. On the other hand, u_9 has a $d_s = 2$ from u_1, but with the $d_t = 0$ to u_1, it is retained. Thus, $N(u_1, 5) = \{u_3, u_5, u_6, u_7, u_9\}$.

Differential aggregation over top-k neighbors $N(u, k)$. Considering a target node u from a frame $\mathcal{G}_h^T[u]$ with the time range $T = [t_{\text{start}}, t_{\text{end}}]$. We leverage the S-T distance to obtain the top-k closest neighbors $N(u, k)$ for u and define a subframe comprised of all nodes from $N(u, k)$ as $f_{u,k}^T \subseteq \mathcal{G}_h^T[u]$. Next, we show how to conduct the differential aggregation in the scope of $f_{u,k}^T$, which includes two steps: (1) S-T feature aggregation of selected neighbors and (2) differential aggregation between selected neighbors and target node. Specifically, we first adopt the regular manner to aggregate S-T features of all the top-k neighbors and then we consider the feature differences between these neighbors and the target node to perform the differential aggregation.

(1) S-T feature aggregation of neighbors. Given a top-k closest neighbors $N(u, k)$, we aggregate the S-T features of all neighbors in $N(u, k)$ as follows. For each neighbor $v \in N(u, k)$, we prepare three types of features for aggregation. First, we require the temporal feature of v w.r.t. the time range $T = [t_{\text{start}}, t_{\text{end}}]$, which is computed as $\Phi(\Delta t)$. Here, $\Delta t_v = t_{\text{end}} - t_v$ denotes the relative time span between t_v and t_{end}, and $\Phi(\cdot)$ is the temporal encoding function mentioned in TGAT. Second, we require the edge feature $\boldsymbol{x}_{uv}$ of e_{uv}. Third, we require the node feature $\boldsymbol{x}_v$ of v itself. Then, we conduct the aggregation as Eq. 1 to obtain the representation of all top-k neighbors in each hidden layer, denoted by $h_{N(u,k)}(f_{u,k}^T)$. We perform the AGG as a concatenation of all involved features.

$$h_{N(u,k)}(f_{u,k}^T) = \text{AGG}(\boldsymbol{x}_v, \boldsymbol{x}_{uv}, \Phi(\Delta t_v)) \tag{1}$$

(2) Differential aggregation between neighbors and target node. To perform the differential aggregation, we first compute the hidden layer representation of the target node u as Eq. 2, where $\Delta t_u = t_{\text{end}} - t_u$, $\boldsymbol{x}_u$ is the node feature of u, and MLP represents the multilayer perceptron. Next, we compute the feature differences between all the top-k neighbors and the target node u as Eq. 3, where $\text{LN}(\cdot)$ represents the normalization function. Finally, we can update the hidden layer representation of u by Eq. 4 based on the top-k neighbors representation $h_{N(u,k)}(f_{u,k}^T)$ (Eq. 1), feature differences $h_u^{\text{dif}}(f_{u,k}^T)$ between top-k neighbors and u (Eq. 3), and the current hidden layer representation $h_u(f_{u,k}^T)$ of u (Eq. 2).

$$h_u(f_{u,k}^T) = \text{MLP}(\text{AGG}(\boldsymbol{x}_u, \Phi(\Delta t_u))) \tag{2}$$

$$h_u^{\text{dif}}(f_{u,k}^T) = \text{LN}(h_{N(v,k)}(f_{u,k}^T) - h_u(f_{u,k}^T)) \tag{3}$$

$$h_u(f_{u,k}^T) = \text{MLP}(\text{AGG}(h_u^{\text{dif}}(f_{u,k}^T), h_{N(u,k)}(f_{u,k}^T, h_u(f_{u,k}^T)))) \tag{4}$$

3.4 Inter-frame Aggregation

In this section, we employ a transformer-based inter-frame aggregation to obtain the historical features for a certain target node u across all time frames

$\{T_1, \ldots, T_n\}$. Specifically, given the target node u that appears in each frame T_i, we first concatenate its representation in various frames as a complete representation h_u by Eq. 5. Next, we compute the query, key, and value vectors Q, K, and V by Eq. 6 based on three weight matrices W^Q, W^K, and W^V. Then, we can compute the self-attention by Eq. 7.

$$h_u = [h_u(f_{u,k}^{T_1}) \parallel \ldots \parallel h_u(f_{u,k}^{T_n})] \tag{5}$$

$$Q = h_u W^Q, K = h_u W^K, V = h_u W^V \tag{6}$$

$$\text{head} = \text{softmax}(\frac{QK^T}{\sqrt{d_K}})V \tag{7}$$

To enhance the capture of historical temporal features across all frames, we use the multi-head attention mechanism to compute m various attentions $\{\text{head}_i, \ldots, \text{head}_m\}$ following Eq. 6 and Eq. 7 using various weight matrices. Then, we can compute the final representation of the target node u by Eq. 8 using a weigh matrix W_{head}.

$$h_u^{\text{final}} = [\text{head}_1 \parallel, \ldots, \parallel \text{head}_m] W_{\text{head}} \tag{8}$$

3.5 Model Training

In fraud detection, the downstream task is a binary prediction for fraudulent nodes using a pre-trained GNN. So, we adopt the binary cross-entropy (BCE) loss for training our model. Our optimization task is as follows, where N denotes the total number of target nodes, $p_{u_i} \in (0, 1)$ denotes the probability of target node u_i being fraudulent, and $y_{u_i} \in \{0, 1\}$ denotes the ground-truth label of u_i.

$$\mathcal{L} = -\frac{1}{N} \sum_{i=1}^{N} [y_{u_i} \log(p_{u_i}) + (1 - y_{u_i}) \log(1 - p_{u_i})] \tag{9}$$

$$p_{u_i} = \text{sigmoid}(\text{MLP}(h_{u_i}^{\text{final}})) \tag{10}$$

4 Experiments

We conducted an experimental study to answer the following questions. Our code was available at [4]. The experiments were run on a 2 GHz Linux server with an RTX 4090D (24GB).

Q1. How does STM perform in terms of effectiveness and efficiency? (**§4.2-4.3**)
Q2. What is the offline training overhead of STM? (**§4.4**)
Q3. What are the effect of each component on STM's effectiveness? (**§4.5**)
Q4. How does STM help to find fraudster? (case study in **§4.6**)
Q5. How is the parameter sensitivity of STM? (**§4.7**)

Datasets →	Wikipedia	Mooc	Reddit	DGraph-Fin
Nodes	9,227	7,074	10,984	3,700,550
Edges	157,474	411,749	672,447	4,300,999
Anomalies	217	4,066	366	15,509
Timespan	30 days	30 days	30 days	2 years

Table 1: Statistics of datasets

4.1 Experimental Setup

Datasets. Table 1 provides some statistics of four real-world datasets, including the number of nodes and edges, anomalies, and timespan. Wikipedia[11] is a dynamic network created by taking the most edited pages and active users as nodes, and the corresponding editing actions as interactions. The 0/1 labels of the inter-actions indicate whether the users are banned from posting. The dynamic network MOOC[14] tracks student behavior on the MOOC online course platform, where interaction events represent user actions in course activities. The 0/1 labels indicate whether the student is actively participating in the course or has dropped out. Reddit[10] is also a dynamic network that tracks active users' posts on Reddit, where interaction events represent users' posts in subreddits. The 0/1 labels of interactions indicate whether users are banned from posting in the subreddits. DGraph-Fin[7] is a directed, unweighted dynamic graph representing the financial social network among Finvolution Group users. In this graph, a node represents a Finvolution user, and an edge from one user to another indicates that the user considers the other as an emergency contact. In Dgraph-Fin, nodes are divided into target nodes and background nodes. Target nodes are labeled as normal (class 0) or fraudulent (class 1). Background nodes are not involved in the prediction task but may play a crucial role in maintaining graph connectivity. For all datasets, we used the same chronological split, with 70% for training, and 15% each for validation and testing.

Parameter settings. We set the default number of frames to 3, the k value of top-k closest neighbors to 20, and the h value of h-hop neighborhood to 2. The parameter M in the S-T distance calculation is set to 1. We use the Adam optimizer with a learning rate of $1e-4$. The batch size for training is set to 256.

Comparing methods. We compare the proposed STM with several baselines, including FTM [1], which proposes graph framing technique, and SAD [18], a current state-of-the-art semi-supervised anomaly detection method, and four other methods Jodie [11], TGAT [25], DyRep [20], and TGN [16].

Metrics. For effectiveness, considering the likelihood of imbalanced datasets, where the number of legitimate entities far exceeds that of fraudulent ones, and in order to account for the varying performance of the model across different thresholds, we adopt AUC as the metric to assess the performance of the fraud detection tasks. AUC is a standard measure in anomaly detection, involving the plotting of the relationship between the true positive rate and the false positive rate. A higher AUC value indicates better performance. For efficiency, we report the online inference time (second), offline training time (second), and

Methods ↓	DGraph-Fin	Wikipedia	Reddit	Mooc	Total Rank
Jodie [11]	0.7011 (5)	0.8534 (6)	0.5717 (7)	0.6155 (7)	25
TGAT [25]	0.6985 (6)	0.6974 (7)	0.6520 (4)	0.6311 (5)	22
DyRep [20]	0.7508 (3)	0.8730 (4)	0.6026 (6)	0.6222 (6)	19
TGN [16]	0.7346 (4)	<u>0.8796</u> (2)	0.6185 (5)	0.6401 (4)	15
FTM [1]	<u>0.7813</u> (2)	0.8693 (5)	0.6751 (3)	**0.7503** (1)	<u>11</u>
SAD [18]	0.6452 (7)	0.8771 (3)	**0.6843** (1)	0.6457 (3)	14
Our STM	**0.7845** (1)	**0.9058** (1)	<u>0.6814</u> (2)	<u>0.7438</u> (2)	**6**

Table 2: Effectiveness in terms of AUC for all models on dynamic graphs

Methods ↓	DGraph-Fin	Wikipedia	Reddit	Mooc	Total Rank
Jodie [11]	1832 (6)	438 (7)	<u>35</u> (2)	147 (6)	21
TGAT [25]	**3** (1)	4 (3)	**17** (1)	**10** (1)	**6**
DyRep [20]	1880 (7)	8 (5)	40 (4)	18 (3)	19
TGN [16]	1648 (5)	7 (4)	38 (3)	<u>17</u> (2)	14
FTM [1]	85 (3)	**2** (1)	147 (5)	41 (5)	14
SAD [18]	745 (4)	41 (6)	320 (7)	647 (7)	24
Our STM	<u>84</u> (2)	**2** (1)	153 (6)	36 (4)	<u>13</u>

Table 3: Online inference time (second) of different methods

model's storage space (KB). We run 100 tests for each experiment and provide the average results. To better present the results, we highlight the top-2 best results in bold, underline, respectively.

4.2 Effectiveness Evaluation

To fairly evaluate the effectiveness of our STM and other six baselines, we configure the neural network structures with similar depth, the same number of hops for GNN aggregation, and the same optimizer settings. The classification results were learned from the respective node embeddings output by each method on the same datasets. The AUC values of the dynamic graph classification results are shown in Table 2 (with the rankings in parentheses). The proposed STM model performs the best on DGraph-Fin and Wikipedia, and achieves the comparable effectiveness to FTM and SAD on other datasets. From the macro perspective (see total rank), STM is the best across all datasets. In a nutshell, STM is the best-performing overall, demonstrating its advantages on dynamic graphs. We attribute these improvements to STM's comprehensive consideration of both short-term and long-term temporal information as well as spatial information, enabling effective aggregation and representation of domain information.

4.3 Efficiency Evaluation

Table 3 shows the online inference time (second) of all methods across four datasets (with the rankings in parentheses). We also present the total rank of

all methods in the last column, showing the overall performance from the macro perspective. We note that TGAT exhibit the best online inference performance, however, it suffers from the effectiveness issue. In contrast, our STM achieves considerable online efficiency compared to TGN and FTM, and outperforms DyRep, Jodie, and SAD, demonstrating a modest inference time.

Methods ↓	**DGraph-Fin**	**Wikipedia**	**Reddit**	**Mooc**	**Total Rank**
Jodie [11]	19776 (5)	439 (6)	**159** (1)	1580 (5)	17
TGAT [25]	2702 (3)	135 (5)	475 (3)	573 (2)	**13**
DyRep [20]	65899 (7)	35 (2)	673 (5)	**84** (1)	15
TGN [16]	19863 (6)	**25** (1)	520 (4)	706 (3)	14
FTM [1]	**1313** (1)	42 (3)	2099 (6)	961 (4)	14
SAD [18]	3125 (4)	815 (7)	3149 (7)	5703 (7)	25
Our STM	1400 (2)	103 (4)	353 (2)	1847 (6)	14

Table 4: Offine training time (second) of different methods

Methods ↓	**DGraph-Fin**	**Wikipedia**	**Reddit**	**Mooc**	**Total Rank**
Jodie [11]	2.5×10^6 (5)	1.1×10^5 (4)	4.5×10^5 (4)	2.8×10^5 (5)	18
TGAT [25]	1.1×10^6 (4)	2.3×10^6 (7)	9.3×10^5 (7)	3.3×10^4 (4)	22
DyRep [20]	2.5×10^6 (5)	1.2×10^5 (5)	4.6×10^5 (5)	2.8×10^5 (5)	20
TGN [16]	2.5×10^6 (5)	1.2×10^5 (5)	4.6×10^5 (5)	2.9×10^5 (7)	22
FTM [1]	**41** (1)	**2304** (1)	**2304** (1)	**1971** (1)	**4**
SAD [18]	7530 (3)	7685 (3)	7685 (3)	7601 (3)	12
Our STM	51 (2)	2804 (2)	2803 (2)	2720 (2)	8

Table 5: Model storage overhead (KB) of different methods

4.4 Offline Training and Model Storage Overhead

Table 4 reports the offline training time (second) of all methods across all datasets (with the rankings in parentheses). We found that STM achieves a comparable offline training efficiency for the largest million-scale graph DGraph-Fin and the densest graph Reddit. While for small graphs, DyRep and TGN have simpler structure, which results in better training efficiency. From the macro perspective, STM is marked as the second efficient in terms of offline training. Moreover, Table 5 reports the model storage (KB) of all methods across all datasets (with the rankings in parentheses). For all datasets, FTM and STM are the top-2 storage efficient methods, demonstrating their superiority from the macro perspective of total rank in the last column.

Methods ↓	DGraph-Fin	Wikipedia	Reddit	Mooc
STM w/o IntraFA	0.7816	0.8679	<u>0.6879</u>	<u>0.7557</u>
STM w/o DiffA	0.7804	0.8760	0.6750	0.7546
STM w/o InterFA	<u>0.7830</u>	<u>0.8967</u>	0.6781	0.7523
Our STM	**0.7845**	**0.9122**	**0.7000**	**0.7575**

Table 6: Ablation study on STM's different components

4.5 Ablation Study

To evaluate the contributions of different components in the proposed STM, we conducted ablation studies to fully explore the impact of each component on the effectiveness. We removed each component one by one to observe the changes in the model's effectiveness results. The specific settings are as follows.

- Our STM: The complete STM model involves all components.
- STM w/o IntraFA: This variant involves discarding the entire intra-frame aggregation component from the complete STM model.
- STM w/o DiffA: This variant involves remaining the intra-frame aggregation via original GNN's aggregation but without the differential aggregation.
- STM w/o InterFA: This variant involves removing the Inter-frame Aggregation based on self-attention from the complete STM model.

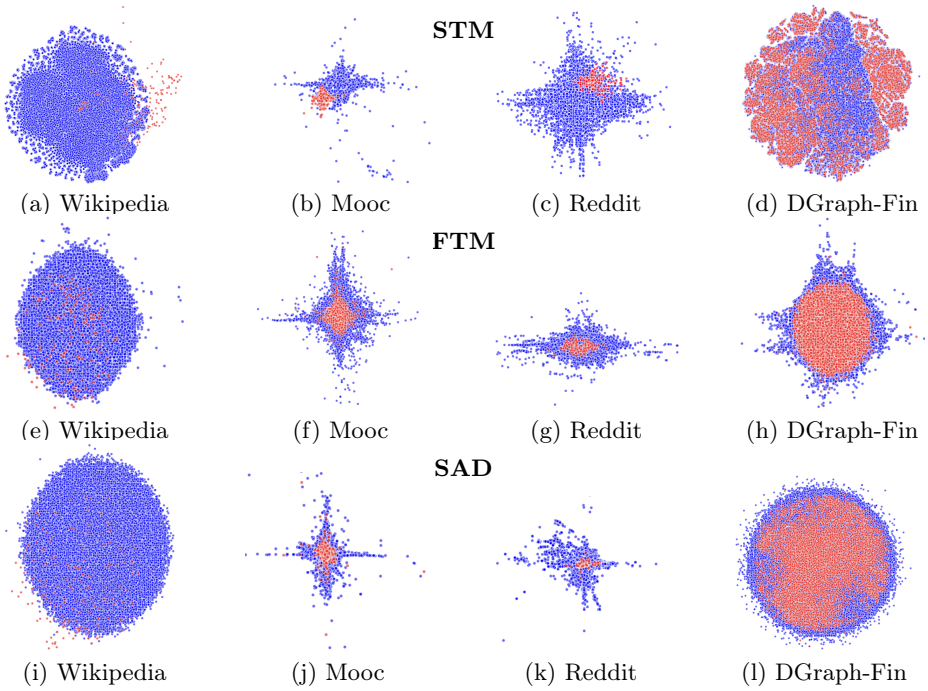
The results shown in Table 6 demonstrate that each component we proposed contributes to improving the model's effectiveness in terms of AUC. The absence of the differential aggregation on the top-k closest neighbors in terms of S-T distances has a significant impact on the model's effectiveness. This is because both short-term temporal and spatial information within a frame are essential for downstream fraud classification tasks. Using the S-T distance can effectively capture the temporal and spatial information and distinguish which neighbors contribute more to the fraud classification.

4.6 Case Study

We visualized the learned node embeddings to evaluate if the proposed STM can well distinguish the abnormal nodes from the normal nodes. The hidden layer's output high-dimensional node embeddings were reduced to 2D using a visualization tool t-SNE, where the blue and red points represent the normal and abnormal nodes, respectively. As shown in Figure 4, comparing to the visualization results of the other latest baselines FTM and SAD, our STM can better distinguish abnormal nodes from normal nodes in the vector space.

4.7 Parameter Sensitivity

In this section, we conduct a series of experiment to study the effect of various parameters of STM on the effectiveness, including the number of frames when performing frame-level timeline split on the original graphs and the k values of the top-k closest neighbors that are used in the differential aggregation in



(a) Wikipedia (b) Mooc (c) Reddit (d) DGraph-Fin

(e) Wikipedia (f) Mooc (g) Reddit (h) DGraph-Fin

(i) Wikipedia (j) Mooc (k) Reddit (l) DGraph-Fin

Fig. 4: Visualization of the learned node embeddings w.r.t. different datasets. The blue and red points represent the normal and abnormal nodes, respectively.

the intra-frame aggregation component. We tested the STM's effectiveness with different frame numbers varied from 1 to 6. The experiments showed that as the frame number increases, the model's classification AUC generally increases, indicating that a higher number of frames helps enhance the model's ability to capture long-term and short-term temporal information. However, an excessive number of frames may lead to a reduction in short-term temporal information, as illustrated by the curves for the DGraph-Fin and Reddit datasets, where prediction accuracy declines after reaching a peak. Moreover, the parameter analysis of the top-k closest neighbors reveals that STM's classification effectiveness tends to improve as the k increases. However, an excessive k also generates side-effect to STM because some nodes with noise features would be included for a large k. Besides, due to the low average degree of nodes in DGraph-Fin (only 1.6), the effect of k is not pronounced.

5 Conclusions

Graph Neural Network (GNN)-based fraud detection models on dynamic graphs have been widely studied. Considering the fact that existing methods overlook the influence differences between long-term and short-term temporal information, we propose STM: a Spatio-Temporal Model for fraud detection on dynamic graphs, which split a dynamic graph into a set of frames and capture short-term

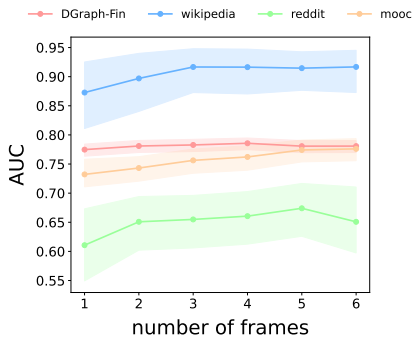


Fig. 5: Effect of the number of frames on STM's effectiveness

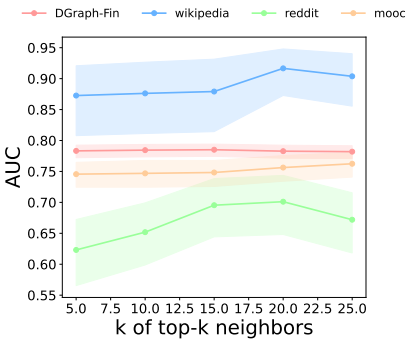


Fig. 6: Effect of top-k closest neighbors on STM's effectiveness

and long-term temporal information via intra-frame and inter-frame aggregation, respectively. For intra-perspective, in each frame, we present a differential aggregation method based on spatio-temporal distance to effectively aggregate both the spatio and short-term temporal information. From the inter-perspective, we capture the long-short term temporal information across all frames based on self-attention mechanism. Extensive experiments on four real-world dynamic graphs demonstrate that STM outperforms existing state-of-the-art methods.

References

1. Cao, B., Ye, Q., Xu, W., Zou, Y.: Ftm: A frame-level timeline modeling method for temporal graph representation learning. In: Proceedings of the AAAI Conference on Artificial Intelligence. vol. 37, pp. 6888–6896 (2023)
2. Cao, B., Ye, Q., Xu, W., Zou, Y.: FTM: A frame-level timeline modeling method for temporal graph representation learning. In: AAAI. pp. 6888–6896. AAAI Press (2023)
3. Cao, Q., Yang, X., Yu, J., Palow, C.: Uncovering large groups of active malicious accounts in online social networks. In: Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security. p. 477–488. CCS '14, Association for Computing Machinery, New York, NY, USA (2014). https://doi.org/10.1145/2660267.2660269, https://doi.org/10.1145/2660267.2660269
4. Code, datasets: Git repository. https://anonymous.4open.science/r/STM/ (2024)
5. Dou, Y., Liu, Z., Sun, L., Deng, Y., Peng, H., Yu, P.S.: Enhancing graph neural network-based fraud detectors against camouflaged fraudsters. In: Proceedings of the 29th ACM international conference on information & knowledge management. pp. 315–324 (2020)
6. Dou, Y., Liu, Z., Sun, L., Deng, Y., Peng, H., Yu, P.S.: Enhancing graph neural network-based fraud detectors against camouflaged fraudsters. In: CIKM. pp. 315–324. ACM (2020)
7. Huang, X., Yang, Y., Wang, Y., Wang, C., Zhang, Z., Xu, J., Chen, L., Vazirgiannis, M.: Dgraph: A large-scale financial dataset for graph anomaly detection. In: NeurIPS (2022)

8. Jemai, J., Zarrad, A., Daud, A.: Identifying fraudulent credit card transactions using ensemble learning. IEEE Access **12**, 54893–54900 (2024). https://doi.org/10.1109/ACCESS.2024.3380823, https://doi.org/10.1109/ACCESS.2024.3380823
9. Kaghazgaran, P., Caverlee, J., Squicciarini, A.C.: Combating crowdsourced review manipulators: A neighborhood-based approach. In: WSDM. pp. 306–314. ACM (2018)
10. Kumar, S., Hamilton, W.L., Leskovec, J., Jurafsky, D.: Community interaction and conflict on the web. In: WWW. p. 933–943 (2018)
11. Kumar, S., Zhang, X., Leskovec, J.: Predicting dynamic embedding trajectory in temporal interaction networks. In: SIGKDD. pp. 1269–1278 (2019)
12. Liu, Y., Ao, X., Qin, Z., Chi, J., Feng, J., Yang, H., He, Q.: Pick and choose: a gnn-based imbalanced learning approach for fraud detection. In: Proceedings of the web conference 2021. pp. 3168–3177 (2021)
13. Liu, Z., Chen, C., Yang, X., Zhou, J., Li, X., Song, L.: Heterogeneous graph neural networks for malicious account detection. CoRR **abs/2002.12307** (2020)
14. Liyanagunawardena, T.R., Adams, A.A., Williams, S.A.: Moocs: A systematic study of the published literature 2008-2012. International review of research in open and distributed learning **14**(3), 202–227 (2013)
15. 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 Transactions on Knowledge and Data Engineering **35**(12), 12012–12038 (2021)
16. Rossi, E., Chamberlain, B., Frasca, F., Eynard, D., Monti, F., Bronstein, M.M.: Temporal graph networks for deep learning on dynamic graphs. CoRR **abs/2006.10637** (2020)
17. Shi, F., Cao, Y., Shang, Y., Zhou, Y., Zhou, C., Wu, J.: H2-fdetector: A gnn-based fraud detector with homophilic and heterophilic connections. In: Proceedings of the ACM web conference 2022. pp. 1486–1494 (2022)
18. Tian, S., Dong, J., Li, J., Zhao, W., Xu, X., Song, B., Meng, C., Zhang, T., Chen, L., et al.: Sad: Semi-supervised anomaly detection on dynamic graphs. arXiv preprint arXiv:2305.13573 (2023)
19. Tian, S., Dong, J., Li, J., Zhao, W., Xu, X., Wang, B., Song, B., Meng, C., Zhang, T., Chen, L.: SAD: semi-supervised anomaly detection on dynamic graphs. In: IJCAI. pp. 2306–2314. ijcai.org (2023)
20. Trivedi, R., Farajtabar, M., Biswal, P., Zha, H.: Dyrep: Learning representations over dynamic graphs. In: ICLR (2019)
21. Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A.N., Kaiser, L.u., Polosukhin, I.: Attention is all you need. In: NeurIPS. vol. 30 (2017)
22. Wang, Y., Zhang, J., Huang, Z., Li, W., Feng, S., Ma, Z., Sun, Y., Yu, D., Dong, F., Jin, J., Wang, B., Luo, J.: Label information enhanced fraud detection against low homophily in graphs. In: Ding, Y., Tang, J., Sequeda, J.F., Aroyo, L., Castillo, C., Houben, G. (eds.) Proceedings of the ACM Web Conference 2023, WWW 2023, Austin, TX, USA, 30 April 2023 - 4 May 2023. pp. 406–416. ACM (2023). https://doi.org/10.1145/3543507.3583373, https://doi.org/10.1145/3543507.3583373
23. Wang, Y., Gou, X., Xu, X., Geng, Y., Ke, X., Wu, T., Yu, Z., Chen, R., Wu, X.: Scalable Community Search over Large-scale Graphs based on Graph Transformer. In: SIGIR (2024)
24. Xie, Y., Liu, G., Yan, C., Jiang, C., Zhou, M.: Time-aware attention-based gated network for credit card fraud detection by extracting transactional behaviors. IEEE Trans. Comput. Soc. Syst. **10**(3), 1004–1016 (2023)

25. Xu, D., Ruan, C., Körpeoglu, E., Kumar, S., Achan, K.: Inductive representation learning on temporal graphs. In: 8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020. OpenReview.net (2020), https://openreview.net/forum?id=rJeW1yHYwH
26. Yang, X., Lyu, Y., Tian, T., Liu, Y., Liu, Y., Zhang, X.: Rumor detection on social media with graph structured adversarial learning. In: IJCAI. pp. 1417–1423. ijcai.org (2020)
27. Zhang, Y., Fan, Y., Ye, Y., Zhao, L., Shi, C.: Key player identification in underground forums over attributed heterogeneous information network embedding framework. In: CIKM. pp. 549–558. ACM (2019)
28. Zhao, Y., Xie, Y., Yu, F., Ke, Q., Yu, Y., Chen, Y., Gillum, E.: Botgraph: Large scale spamming botnet detection. In: NSDI. pp. 321–334. USENIX Association (2009)