Xing et al. *EPJ Data Science* (2025) 14:66
https://doi.org/10.1140/epjds/s13688-025-00584-4

EPJ Data Science
a SpringerOpen Journal

EPJ.org

RESEARCH Open Access

A cross-social platform distributed anomaly behavior detection method

Ling Xing[1], Shiyu Li[1†], Honghai Wu[1*†], Qi Zhang[2†], Huahong Ma[1†] and Kaikai Deng[1†]

Handling Editor: Sabrina Gaito

*Correspondence:
honghai2018@haust.edu.cn
[1]College of Information Engineering, Henan University of Science and Technology, Kaiyuan Avenue, Luoyang, 471000 Henan, China
Full list of author information is available at the end of the article
[†]Equal contributors

Abstract

With the rapid development of social networks, anomaly behavior detection has become essential for ensuring platform security and enhancing user experience. Traditional anomaly detection methods often rely on centralized data processing, making it difficult to address the challenges of cross-platform collaborative detection and failing to fully leverage the temporal and graph structure information within social networks. To improve anomaly behavior detection's accuracy and generalization ability, this paper proposes a Cross-Platform-Based Distributed Anomaly Behavior Detection Method, FLAD. Specifically, the method employs the Federated Averaging (FedAvg) algorithm for model aggregation, constructing a decentralized anomaly behavior detection model. This approach avoids the exchange of raw user behavior data and enhances the effectiveness of cross-platform collaborative learning. Moreover, this paper introduces a detection model that combines a sliding window and Graph Convolutional Network (GCN), utilizing temporal data and the graph structure of social networks. By partitioning user behavior data into subgraphs via a sliding window, and employing GCN and Long Short-Term Memory (LSTM) models to learn the evolution patterns of temporal and behavioral data, the method improves the accuracy of anomaly behavior detection. Experimental results show that the proposed method achieves up to a 6.31% increase in F1-score compared to baseline models on the Epinions and Digg datasets. Ablation studies further demonstrate that the unified model, aggregated through federated learning, significantly improves accuracy over the initial global model. In the Epinions dataset, accuracy increased by 20.49%, and in the Digg dataset, accuracy improved by 23.13%.

Keywords: Anomaly Behavior Detection; Cross-Platform Social Networks; Federated Learning; Graph Convolutional Network (GCN)

1 Introduction

In the digital age, social media platforms have become the core of personal and professional communication, information dissemination, and community building [1]. With the explosive growth of social network users, the frequency of anomalous behaviors, such as cyberbullying [2], misinformation spread [3], and unauthorized data access, has also increased, posing significant threats to the integrity of social platforms and user safety.

Springer

Research on anomaly behavior in social networks helps enhance capabilities in areas such as user behavior analysis [4], community detection [5–7], anomaly warning [8, 9], and public opinion analysis [10, 11]. The diversity of social platforms enables them to attract users and support various forms of interaction. Traditional anomaly behavior detection methods typically rely on centralized data processing. While this approach can improve detection accuracy, it fails to reveal malicious behaviors that occur across multiple platforms fully. Since user behavior data often contains sensitive information, achieving cross-platform collaborative training while ensuring privacy protection has become a pressing challenge [12]. Additionally, existing detection models usually fail to fully leverage temporal data and graph structure information within social networks, making it difficult to capture the dynamic evolution of user behavior patterns accurately.

To cope with the evolving and complex anomalous behaviors in social networks, existing methods need to meet the following three requirements. First, cross-platform modeling capability, which can comprehensively analyze the user's behavioral patterns on multiple social platforms, and identify the anomalous behaviors that are hidden in the linkage between the platforms. Second, privacy protection mechanism, which avoids leaking the user's sensitive information in the process of data transmission or centralized processing; and third, dynamic behavioral modeling capability, which can combine graph structure and time series information to portray the evolution of user behaviors, thus achieving more accurate detection. The third is dynamic behavior modeling, which can combine graph structure and timing information to portray the evolution of user behaviors, thus achieving more accurate anomaly detection. These challenges motivate us to design a federated anomaly detection framework that jointly captures structural dependencies and temporal dynamics while preserving cross-platform privacy.

In this context, this paper proposes a cross-platform distributed abnormal behavior detection method called FLAD. This method supports collaborative training of global abnormal detection by multiple social network clients through decentralized data fusion. We use the FedAvg [13] method to initialize the global model on the server side and distribute it to each client. The client uses local data for training and uploads the optimization parameters. In addition, this paper also proposes an abnormal behavior detection model based on sliding windows and GCN. This model effectively improves the accuracy of abnormal behavior detection by combining time series data and graph structure information in social networks. By dividing user behavior data into subgraphs using a sliding window and embedding user behavior information using a GCN, combining an LSTM model to learn time series information, we can capture the evolution of behavior patterns and accurately assess abnormal behavior. The main contributions of this paper are as follows:

1. We propose a distributed collaborative training framework based on FedAvg, enabling multiple social network clients to train a global anomaly detection model collaboratively without sharing user behavior data through decentralized data fusion. Using the FedAvg method, the server initializes the global model and sends it to the client, which uses local data to train and upload optimization parameters. To protect user privacy, the client uses noise injection technology when uploading data to ensure data security. During the aggregation process, the server updates the global model using weighted averaging to improve the model's performance and convergence speed. This framework achieves a good balance between privacy

protection and abnormal behavior detection performance, and is suitable for large-scale social networks.

2. We propose a social network anomaly detection model based on sliding windows and GCN. This model improves anomaly detection accuracy by combining time series information with graph structure information. Specifically, user behavior data is divided into subgraphs using sliding windows, user behavior information and semantic information are embedded using GCN, and the final subgraph vector representation is generated using vector stitching and linear transformation. The LSTM is used to learn the time series information, capture behavior patterns' evolution, and evaluate abnormal behavior using the deep learning model Encoder-Decoder-Encoder. This model can effectively integrate time series and graph structure information to adapt to the dynamic user behavior patterns in social networks.
3. We conduct experiments on the real-world datasets Epinions and Digg. The results show the effectiveness of our proposed distributed collaborative training framework, where the accuracy of the unified model aggregated by federated learning is improved compared to the initial global model. In addition, our social network anomaly detection model based on sliding windows and GCN significantly improves the key performance indicator F1-score compared to the baseline model.

The rest of this paper is organized as follows: Sect. 2 reviews related research on anomaly detection; Sect. 3 introduces a cross-platform distributed anomaly detection method; Sect. 4 briefly describes the dataset and compares it with the baseline method to verify the effectiveness of the model; and Sect. 5 summarizes this study and looks ahead to future research directions.

2 Related work

2.1 Anomaly detection on social networks

Social network anomaly detection identifying behaviors that deviate from normal user activities in social networks. Traditional anomaly detection methods can be divided into text clustering-based anomaly detection and graph neural network-based anomaly detection. The text clustering-based anomaly detection method mainly analyzes the text content posted by users, extracts feature such as keywords and uses clustering algorithms to cluster the text content [14–17]. Outlier detection techniques are used to identify abnormal user groups. This method can effectively capture features in the text that deviate from regular patterns and help identify potentially abnormal behaviors. However, text feature extraction cannot entirely capture anomalies. Anomaly detection methods based on graph neural networks usually model social networks as graph structures, where nodes represent users and edges represent user interactions [18–20]. The graph neural network model can effectively learn the embedding representation of nodes to capture the complex relationships and interaction patterns between users [21–24]. The model can extract local and global network features on the graph structure, thereby identifying abnormal behaviors different from regular users in the social interaction structure, such as fake account clusters and online fraud [25–27].

Compared with text clustering methods, graph neural network-based detection methods are more suitable for processing user interaction data and can effectively identify abnormal behaviors that differ from regular user groups in the social network structure.

However, most existing graph neural network-based anomaly detection methods are limited to data analysis on a single platform and do not fully consider the challenges in cross-social network scenarios. In practice, user behaviors often span multiple platforms simultaneously, and malicious or fraudulent activities can propagate across different social networks, making cross-platform anomaly detection increasingly necessary. The data heterogeneity and complex interaction patterns between different platforms limit the effectiveness of these methods in a multi-platform environment.

2.2 Federated learning

Federated learning FL is a distributed machine learning technique that allows different data participants to collaboratively construct machine learning models without disclosing the original data. Existing federated learning primarily focuses on the core problem of aggregation, and related studies can be broadly categorized into three areas: the effectiveness of aggregation, the object of aggregation, and the constraints of aggregation. The first aspect is the effectiveness of aggregation. Some researchers have mitigated the model drift problem caused by heterogeneity through regularization constraints [28], multi-task modeling [29], and hierarchical parameter matching [30], with the common goal of improving the stability and convergence of federated learning in non-independent and identically distributed (Non-IID) environments. Second, in terms of the object of aggregation, the researcher proposes a personalized federal learning approach in response to the limitation that a single global model is difficult to adapt to the individual needs of each client [31–33]. These methods achieve personalized modeling by flexibly dividing parameters between sharing and localization, thus enhancing the adaptability and performance of federated learning in Non-IID scenarios. Finally, in terms of the constraints of aggregation, the researchers effectively guarantee the security, fairness, and robustness of federated learning training by introducing security protocols [34], weighting mechanisms [35], and regularization constraints [36] in the aggregation process.

In recent years, federated learning has extended to graph data scenarios [37–39], and researchers have proposed graph federated learning (FGL) to achieve distributed graph modeling using graph neural networks while preserving privacy [40–44]. He et al. [45] proposed the FedGraphNN framework to systematically combine GNN tasks with federated learning. Zhang et al. [46] proposed FedSage+ to improve performance in heterogeneous graph environments through graph sampling and neighbor generation optimization. Cai et al. [47], proposed LG-FGAD, a representative federated graph anomaly detection framework tailored for static graphs, incorporating hierarchical attention mechanisms and global aggregation. However, realistic networks change over time and have temporal dependencies, making traditional graph federated learning challenging in dealing with dynamic and temporal relationships. For this reason, researchers further proposed Dynamic Federated Graph Learning (D-FGL). Zhang et al. [48] propose Feddy, a federated learning framework that combines dynamic graph neural networks with secure aggregation mechanisms for efficiently modeling distributed dynamic graph data while preserving privacy. The DFDG framework proposed by Usman et al. [49] utilizes an adaptive federated learning mechanism to handle the task of traffic prediction in dynamic graph structures, aiming at efficiently modeling dynamic temporal features in a distributed environment. Overall, the existing dynamic graph federated learning, while making progress, is still limited to node classification and link prediction, and is under-explored for complex timing and efficient communication.

In this context, we propose FLAD, a cross-platform distributed framework for anomaly behavior detection in social networks. By leveraging GCNs on sliding window-based subgraphs and capturing sequential patterns through LSTM, our model effectively traces the dynamic evolution of user interactions. Furthermore, FLAD employs FedAvg to enable collaborative training across multiple platforms without exposing raw user data, thereby preserving privacy while improving model generalization in heterogeneous, distributed environments.

3 Methodology

3.1 Overview of the distributed collaborative system architecture

Due to the complexity of social networks and the heterogeneity of cross-platform data, traditional centralized anomaly detection methods often face challenges related to computational efficiency and data storage. To address this issue, this paper proposes a cross-social platform distributed anomaly behavior detection method, FLAD, whose core idea is to enhance the overall anomaly detection capability through collaboration between multiple social networks. Each social network first performs local anomaly behavior detection and then, through model parameter aggregation, achieves global anomaly detection across platforms. The main operational mechanism involves each social network detecting local user behavior data using an anomaly detection model based on sliding windows and GCN, providing data support for subsequent global detection. Then, each social network shares its local model parameters with the global model, using a decentralized information transmission and processing approach. After the global model is updated, the parameters are distributed back to the clients, enabling local model updates and improving the overall detection capability. Figure 1 illustrates the architecture of the Cross-Platform-Based Distributed Anomaly Behavior Detection Method, FLAD.

3.2 Subgraph partitioning method based on sliding window

We first apply a sliding window approach to segment the user behavior knowledge graph in the social network into subgraphs. For each time window, we construct a sequence of

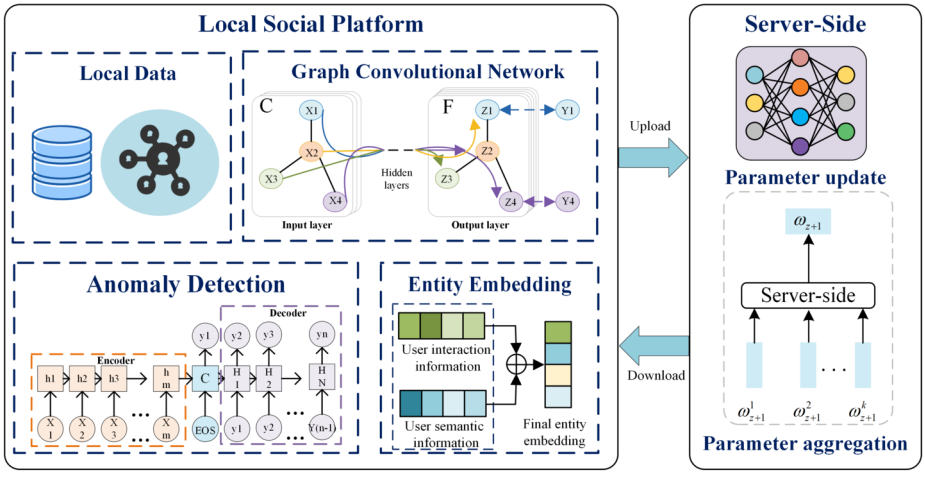

Figure 1 Architecture Diagram of FLAD

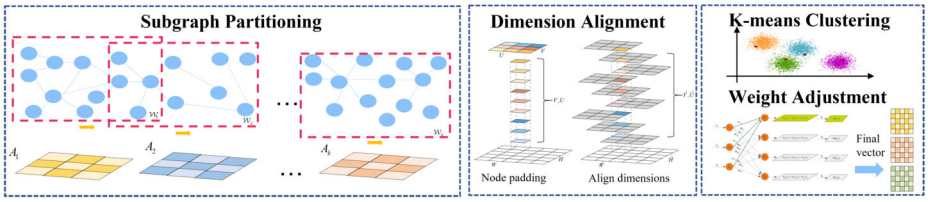

Figure 2 Flowchart of Subgraph Partitioning Based on Sliding Window

temporal adjacency matrices and perform zero-padding for missing nodes to achieve dimensional alignment. Meanwhile, spectral clustering is introduced to optimize the model parameters. Based on this, we initialize the hidden states of the nodes and dynamically adjust their weights to ultimately generate the representation vector for each subgraph. The detailed process is illustrated in Fig. 2.

First, the social network is formalized as a set of temporal graphs $\mathcal{G} = \{G^{(t)}\}_{t=1}^{\tau}$, where $G^{(t)} = \{V^{(t)}, E^{(t)}, X^{(t)}\}$ represents the graph structure at time t, including the node set $V^{(t)}$, the edge set $E^{(t)}$, and the node attribute matrix $X^{(t)} \in \mathbb{R}^{|V^{(T)}| \times d}$. Given a window length $T \in \mathbb{Z}^+$ and a sliding step size $S \in \mathbb{Z}^+$, the subgraphs constructed by the sliding window are defined as $\{\mathcal{W}\}_{k=1}^{K}$, where $K = \left\lfloor \frac{\tau - T}{S} \right\rfloor + 1$.

In this module, we first initialize the window size and construct the first subgraph window starting from the initial time $t = 1$. The content is given by Eq. (1):

$$\mathcal{W}_1 = \bigcup_{t=1}^{T} G^{(t)} \tag{1}$$

The nodes and edges satisfy Eq. (2) and Eq. (3), and then the window slides with a step size S. The k-th subwindow is defined as Eq. (4):

$$V^{\mathcal{W}_1} = \bigcup_{t=1}^{T} V^{(t)} \tag{2}$$

$$E^{\mathcal{W}_1} = \bigcup_{t=1}^{T} E^{(t)} \tag{3}$$

$$\mathcal{W}_k = \bigcup_{t=(k-1)S+1}^{(k-1)S+T} G^{(t)}, \quad \forall k \geq 2 \tag{4}$$

In our framework, we adopt a hybrid sliding window mechanism that combines data-driven parameter optimization and empirically informed configuration. Specifically, the window length T is automatically optimized using a spectral clustering-based strategy, which adaptively segments the behavioral graph sequence based on structural similarity and density patterns. This optimization ensures that each window captures relatively homogeneous interaction dynamics.

Meanwhile, the sliding step size S is fixed to 1 in our implementation, based on prior empirical observations. This setting ensures maximal overlap between adjacent windows, which helps preserve fine-grained temporal continuity and mitigate the loss of global behavioral context. We further discuss the impact of this choice in our ablation studies. Next,

for each window $\mathcal{W}_k$, we construct a temporal adjacency matrix sequence. The specific process is given by Eq. (5):

$$A_k = [A^{(1)}, A^{(2)}, \ldots, A^{(T)}] \in \mathbb{R}^{T \times N \times N} \tag{5}$$

where $N = \max_{1 \le t \le T} |V^{(t)}|$, Missing nodes are filled with zeros to maintain dimensional consistency. Then, we determine the window parameters through spectral clustering optimization. First, we compute the temporal similarity matrix by Eq. (6):

$$\sum_{ij} = \exp(-\gamma \|\phi(G^{(i)}) - \phi(G^{(j)})\|_2^2) \tag{6}$$

where $\phi(\cdot)$ is the graph topological feature extraction function. Next, we solve the eigenvalue equation $\sum v = \lambda v$ and take the top p largest eigenvectors to construct an embedded space. Finally, we use K-means clustering to determine the optimal window length, as given in Eq. (7):

$$T^* = \arg\min_{T} \sum_{i=1}^{\lfloor \tau / T \rfloor} \sum_{G^{(t)} \in C_i} \left\| \phi\left(G^{(t)} - \mu_i\right) \right\|^2 \tag{7}$$

To maintain continuity between subgraphs, we initialize hidden states for overlapping nodes, as described in Eq. (8):

$$x_v^{(k+1,0)} = \sigma(W_t x_v^{k,T} + c_t) \tag{8}$$

where $W_t \in \mathbb{R}^{d_h \times d_h}$ is the transition matrix, and $x_v^{(k+1,T)}$ represents the final state of window k. Then, we define the temporal decay function $\rho(t) = e^{-\beta \Delta t}$, which dynamically adjusts edge weights according to Eq. (9):

$$\tilde{A}_{uv}^{(t)} = \rho(t_{uv} - t) A_{uv}^{(t)} \tag{9}$$

where t_{uv} represents the most recent activation time of edge (u, v), and β is the decay coefficient. To address the potential limitation of global context loss, we incorporate several complementary mechanisms into our framework. Specifically, we fix the sliding step size based on prior empirical evidence, which maximizes the overlap between consecutive windows and helps maintain temporal continuity. Additionally, we introduce a state propagation module that initializes each window's node representation using the final state from the previous window, preserving semantic consistency across subgraph sequences. And we utilize an LSTM-based sequential encoder to model temporal dependencies across subgraphs, allowing the framework to capture long-range behavioral dynamics. Together, these mechanisms allow our model to mitigate the fragmentation introduced by sliding window segmentation and maintain awareness of the global temporal context.

3.3 Anomaly behavior detection based on GCN

In this section, we use GCN to perform graph embedding learning on the partitioned user behavior subgraphs, capturing relationships and pattern changes between nodes to

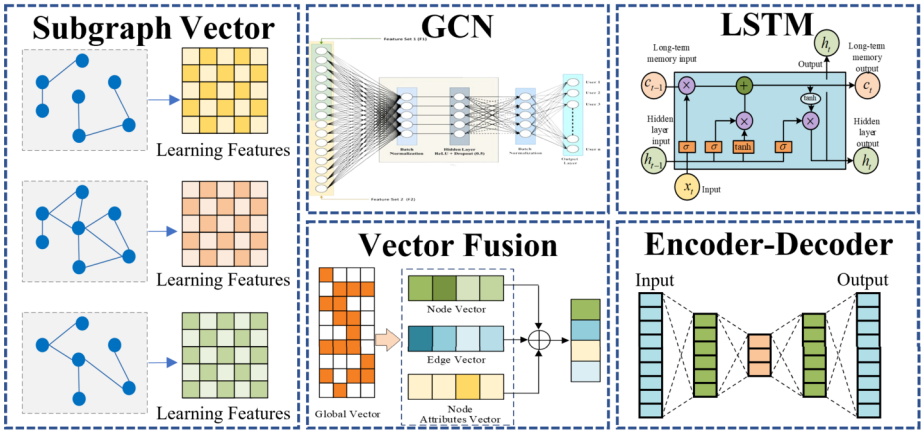

Figure 3 Flowchart of Anomaly Behavior Detection Based on GCN

improve anomaly behavior detection accuracy. Additionally, to enhance sensitivity to temporal variations, an LSTM network is incorporated to model time series data and identify the evolution of user behavior patterns. Considering both temporal information and graph structure, this method improves the model's generalization ability and accuracy in complex social network environments. The detailed process is shown in Fig. 3.

Next, we use a commonly adopted GCN to extract user interaction features from the constructed temporal subgraphs. In a single operation of GNN, node information propagation and feature aggregation are two key steps, as shown in Eq. (10) and Eq. (11). These steps allow the network to capture and integrate the local structural information of nodes within the graph. By employing this approach, GNN can effectively learn node representations in the graph, providing strong feature representations for subsequent tasks. In summary, this process refines node feature representations by considering both each node and its neighbors, enabling the model to better capture the complex structural properties of the graph.

$$h_{u,\text{nei}}^{l} = \text{aggregate}_{l+1}\left(\left\{h_{v}^{l} \mid v \in \text{Neighbor}(u)\right\}\right) \tag{10}$$

$$h_{u}^{l+1} = \text{combine}_{l+1}\left(h_{u}^{l} \parallel h_{u,\text{nei}}^{l}\right) \tag{11}$$

Among them, h_u^l represents the hidden representation of node u at layer l, and $h_{u,nei}^l$ represents the aggregation of neighbor information of u at layer l. The functions $aggregate_l(\cdot)$ and $combine_l(\cdot)$ correspond to the aggregation and update operations at layer l, respectively.

The graph neural network first maps the graph data to the frequency domain for convolution operations and then maps it back to the node space. The specific process is shown in Eq. (12):

$$Z^{(l+1)} = Act\left(\tilde{D}^{-1/2}\tilde{A}\tilde{D}^{-1/2}Z^{l}W^{l}\right) \tag{12}$$

Among them, Z^l represents the hidden representation of nodes at layer l in the GCN model, $\tilde{A} = A + I$ is the adjacency matrix with self-loops added, $\tilde{D} = \sum_{i=1}^{N}\tilde{A}_{ii}$ is the degree matrix of $\tilde{A}$, and $Act(\cdot)$ is the activation function. It is important to note that when

processing dynamic network data, not only do nodes have their own intrinsic meanings and attributes, but the edges or relationships connecting them also carry meaningful features and attributes. By performing sequential operations of graph embedding layers and semantic attention, hierarchical feature representations for each node can be extracted. Specifically, the GNN layer first obtains the initial representations of all nodes, then uses semantic attention to aggregate these representations to form the final node embeddings. This process considers not only the information of the nodes themselves but also the attributes and significance of the edges, as well as the interactions between nodes. This enables a more comprehensive understanding of the dynamics and complexity of social networks.

Both edge and node information should be fully utilized when constructing network structures and attribute feature extraction mechanisms. The method uses Eq. (13) to map the original graph to its line graph, where edges serve as the fundamental units of analysis in this transformed network.

$$R_{ij} = \begin{cases} 1, & s_{i,\text{from}} = s_{j,\text{from}} \text{ or } o_{i,\text{to}} = o_{j,\text{to}} \\ 0, & \text{otherwise} \end{cases} \tag{13}$$

Where $s_{i,from}$ is the source node of edge i, and $o_{i,to}$ is the target node of edge i. Then, Eq. (14) is used to extract the corresponding features on the line graph.

$$Z_E^{(l+1)} = Act\left(\tilde{D}_E^{-1/2}\tilde{E}\tilde{D}_E^{-1/2}Z_E^l W_E^l\right) \tag{14}$$

Two sets of GCNs are used to extract features from the original graph and its corresponding line graph, respectively, and then integrate them. After extracting these two types of information, the framework concatenates them and applies a linear transformation to integrate user behavioral and semantic information into the embedding process. This enhances the implicit representation and improves both the information content and the accuracy of the embedding.

By integrating LSTM and GCN, this method can comprehensively analyze users' behavioral and semantic features over time in social networks. This not only captures the time-varying characteristics of user behaviors but also provides a deeper understanding of network dynamics and richer information for detecting anomalous behaviors. These features are integrated into a global representation vector using a read function, which further enhances the model's ability to accurately reflect the network's evolution over time by incorporating temporal information. This approach is introduced to improve the model's performance on dynamic social network data, enabling it to identify and predict anomalous behavior more effectively.

This paper utilizes a readout function to extract representation vectors of all entities, relationships, and temporal information from the current subgraph representation space $U_v \in \mathbb{R}^{m\times d}$, ultimately obtaining the overall representation vector of the graph $s_t \in \mathbb{R}^d$. At time t, for the subgraph $G_t = (X_t, A_t)$, in the initialization step, a graph neural network is used to extract users' behavioral and semantic features. Then, a readout function is applied to obtain a global feature representation s_t. The global representation s_t is then used as input to a LSTM network at time t. During the model training phase, a variation loss function is introduced to impose constraints on the features extracted by the LSTM,

integrating dynamic feature variations in vector form. The specific process is shown in Eq. (15):

$$L_1 = \left\| y_t - \frac{1}{t-1} \sum_{i=1}^{t-1} y_i \right\|_2 \tag{15}$$

This loss function is designed to ensure that the model generates a representation vector that, when combined with the time series information, is as close as possible to the average of all previously generated vectors.

In this method, the subgraph information of the social network is first encoded into a distributed representation vector by an encoder (Encoder1). This vector is then used to train a decoder (Decoder) to efficiently extract key information from the global vector. The generated vector is subsequently encoded again by an encoder (Encoder2), which has a similar structure to Encoder1, to further optimize model performance. By comparing the difference between the representation vectors output by Encoder1 and Encoder2, this difference can be directly utilized as a basis for detecting anomalous behavior. This approach enables the model to efficiently identify anomalous data without requiring direct access to previous data distributions. The specific formulations are shown in Eq. (16), Eq. (17) and Eq. (18):

$$L_2 = \|Decoder(Encoder(X)) - X\|_2 \tag{16}$$

$$L_3 = \|Encoder(Decoder(Encoder(X))) - Encoder(X)\|_2 \tag{17}$$

$$Score_{anomaly} = \|Encoder(Decoder(Encoder(X))) - Encoder(X)\|_2 \tag{18}$$

3.4 Parameter update mechanism in federated averaging

After completing the initialization of the global model, the server distributes it as the base model. The global model is transmitted to each social network platform through a secure communication protocol, ensuring that every platform receives the same initial model. The detailed process is shown in Fig. 4.

Throughout the model training process, we adopt the FedAvg [13] method to handle large-scale data through a client-side distributed training strategy. Each client trains the model locally and uploads the trained model parameters to the server. The server aggregates the parameters from each client using a weighted averaging approach to update the global model. Through this method, the server and clients utilize a synchronous parameter server to update model parameters, accelerating model convergence and improving performance. After completing the initialization of the global model, the server distributes it as the base model, ensuring that each platform receives the same initial model. First, when detecting real user behavior data in local social networks, local differential privacy is applied to perform a noise injection operation on the local user behavior data. The specific process is given in Eq. (19).

$$\text{len}(b_i^{\text{noise}}) = \varepsilon \cdot \text{len}(b_i) \tag{19}$$

Where b_i represent the true behavior of user i, and b_i^{noise} represent the user behavior that did not occur, which is generated through noise injection. Together, they form the

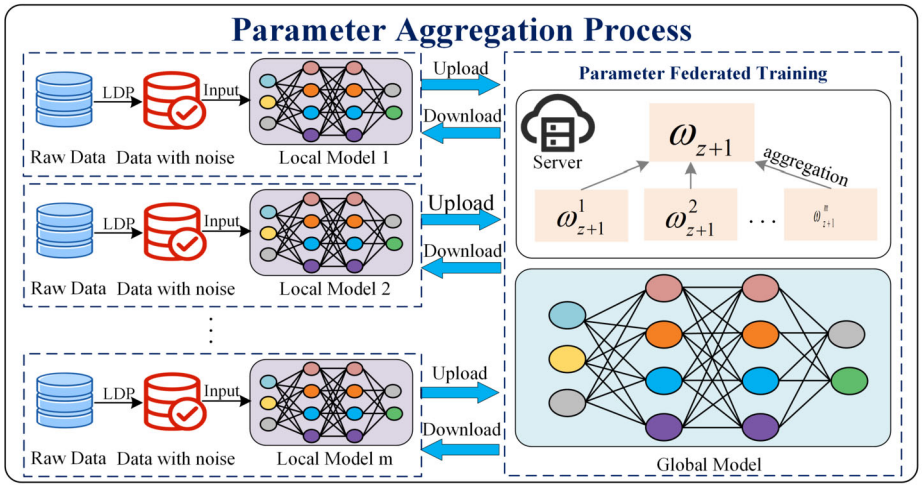

Figure 4 Flowchart of the Parameter Update Mechanism in FedAvg

behavior set Δ_i of user i. The parameter ε controls the proportion of noise data added. A higher proportion improves privacy protection but impacts model performance. Through experiments, it is found that when ε achieves a good balance between privacy protection and performance preservation, the anomaly detection performance of user behavior improves. When $\varepsilon \in (0.25, 0.35)$, it can simultaneously achieve privacy protection and enhance anomaly detection performance. First, a server and m participants are defined, where each participant owns its dataset D_i. Given the server's initialized model parameters ω, the global objective function $F(\omega)$ is obtained by aggregating the objectives $F_i(\omega)$ of each local participant. Specifically, in each training round with M social network platforms participating in federated learning, the local update for participant m is computed based on Eq. (20):

$$g_m = \Delta F_M(\omega_z) \tag{20}$$

Where ω_z is the global parameter downloaded from the server in round z, and each participant updates its local parameters according to Equation (21):

$$\forall m, \omega_{z+1}^m \leftarrow \omega_t - \eta g m \tag{21}$$

The server then aggregates the local parameters uploaded by all participants and performs a weighted average using the FedAvg method, as shown in Eq. (22):

$$\omega_{z+1} \leftarrow \sum_{m}^{M} \frac{n_m}{n} \omega_{z+1}^m \tag{22}$$

Among them, n_m is the data volume of the m-th participant, and ω_{z+1} is assigned to each participant in round $(z+1)$.

The above training process is repeated until the global model loss function reaches its minimum, as shown in Eq. (23), and is used as the global model.

$$F(\omega) = \sum_{i=1}^{N} \frac{|D_i|}{D} F_i(\omega) \tag{23}$$

Among them, D is the total size of all participants' datasets, given by $D = \sum_{i=1}^{N} |D_i|$. The local objective function $F_i(\omega)$ is the loss function defined on the dataset D_i. The aggregation process essentially follows the idea of FedAvg: in each round, the client performs several gradient descent updates on the local dataset, and then the server-side performs weighted averaging according to the amount of data to obtain new global model parameters. As participants continue to perform local optimization and global aggregation, the loss function $F(\omega)$ will gradually converge to the optimal solution. This occurs because, in each iteration, the global aggregation optimally combines local solutions, allowing the model to progressively approach the optimal global solution, ultimately resulting in the unified global model after aggregation.

4 Experiments

This paper selects several baseline models from the server-side model initialization perspective and evaluates their performance on two real-world datasets. Additionally, an ablation study is conducted on the proposed FLAD model to demonstrate the effectiveness of the federated learning framework for anomaly behavior detection.

4.1 Experiment setting

Data Set: We conduct experiments on real-world datasets, Epinions and Digg, to verify the effectiveness of the FLAD model.

Epinions Dataset: It is a signed social network where users can express trust or distrust toward others. The dataset contains 75,879 nodes and 508,837 edges, representing user-to-user relationships and review interactions. Similar to Digg, this dataset includes structural information, but it is not labeled.

Digg Dataset: This dataset collects social interaction data from the Digg news-sharing platform. It consists of approximately 7,708,000 nodes and 5,900,000 edges, capturing user activities such as sharing, commenting, and voting on news articles. The dataset includes structural and attribute information, and it is not labeled either.

Evaluation Metric: In this paper, we use F_1-score as the performance metric for our experiment. This metric represents the harmonic mean of precision and recall, with a maximum value of 1 and a minimum value of 0. It effectively reflects the accuracy of the proposed algorithm. The calculation formula is given in Eq. (24):

$$F_1\text{-}score = \frac{2 \times \text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}} \tag{24}$$

where Precision refers to the proportion of correctly classified positive samples among all predicted positive samples, and Recall refers to the proportion of correctly predicted positive samples among all actual positive samples. The formulas for Precision and Recall are given in Eq. (25) and Eq. (26):

$$\text{Precision} = \frac{TP}{TP + FP} \tag{25}$$

$$\text{Recall} = \frac{TP}{TP + FN} \tag{26}$$

where TP (True Positives) represents the number of correctly predicted positive samples, FP (False Positives) represents the number of misclassified samples predicted as positive, and FN (False Negatives) represents the number of actual positive samples misclassified as other classes.

Baseline Models:We select several classical graph embedding models as baseline models and evaluate the proposed FLAD model's performance on two real-world datasets.

- DeepWalk [50]: A classic graph representation learning algorithm that utilizes random walks to obtain node context and applies the skip-gram algorithm for learning node representation vectors.
- Node2Vec [51]: An improved version of the DeepWalk method that introduces biased search strategies.
- GraphSAGE [52]: This method employs a skip-gram-based loss function to achieve unsupervised learning.
- NetWalk [53]: This method uses reservoir sampling to preserve dynamic graph information and leverages deep autoencoders and clique embedding for node representation learning. In the original work, streaming means were used to score edge anomalies.

Parameter Settings: To verify the accuracy of the proposed federated learning-based cross-social network anomaly behavior detection model, we randomly select three-quarters of the data as the training set, while the remaining one-quarter is used as the test set. Since the original datasets do not contain anomaly labels, we apply anomaly injection on Epinions and Digg to conduct our evaluations. We injected 1%, 5%, and 10% anomalous elements into each dataset following commonly used strategies. To evaluate the experimental results, we compute the F1-score for each method. We set the dimension of the representation vector to 512, run each experiment 10 times, and report the average results as the final performance. In these anomalous graphs, 0 to 3 edges are randomly chosen and replicated 30 times, simulating anomaly edge injections. The model is trained on the anomaly-free training set and then tested on the test set to validate its structure and performance. This method effectively evaluates the model's capability and accuracy in detecting anomalies. For the selected baseline models and representation learning algorithms, we execute these algorithms on each graph to obtain edge vector representations. Next, a readout function is applied to extract information from these representation vectors, constructing a comprehensive representation of the entire graph. Additionally, we set an initial iteration count, where each iteration refines the model parameters based on the previous round's output to achieve optimal performance. After 50 iterations, the model better adapts to the dataset, extracting more precise and representative graph embeddings. Furthermore, noise is added to the parameter transmission process with values selected from 0, 0.1, 0.2, 0.3, 0.4, and 0.5, while the learning rate is set to 0.1, 0.05, and 0.01. The training process is terminated once the predefined number of training rounds is reached.

4.2 Experiment result

The experimental results show that the F1-score of the proposed FLAD with the baseline model is improved in both the Epinions dataset and the Digg dataset. The specific results are shown in Fig. 5, Fig. 6, Table 1 and Table 2. Figure 5 and Table 1 show the F1-score

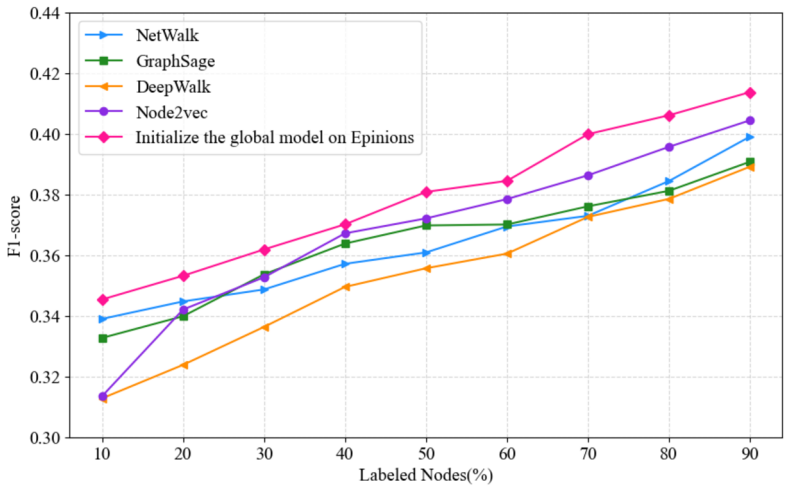

Figure 5 Performance Comparison Between FLAD and Baseline Models on the Epinions Dataset

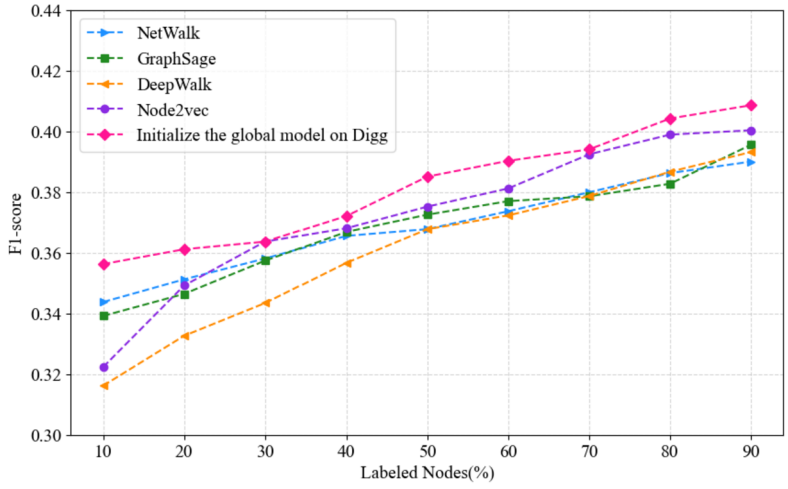

Figure 6 Performance Comparison Between FLAD and Baseline Models on the Digg Dataset

Table 1 Performance Comparison Between FLAD and Baseline Models on the Epinions Dataset

Model	10%	20%	30%	40%	50%	60%	70%	80%	90%
Netwalk	0.33892	0.34467	0.34862	0.35707	0.36082	0.36938	0.37289	0.38435	0.39898
Deepwalk	0.33265	0.3398	0.3536	0.36374	0.36972	0.37002	0.37602	0.38113	0.39070
Node2vec	0.31274	0.32377	0.33628	0.34946	0.3556	0.36043	0.37254	0.37847	0.38909
Graphsage	0.31344	0.342	0.35264	0.36711	0.37203	0.3784	0.38624	0.39562	0.40436
FLAD	0.34532	0.35313	0.36181	0.37008	0.38076	0.38441	0.3998	0.40602	0.41365

performance of FLAD with the baseline model in the Epinions dataset. Figure 6 and Table 2 show the F1-score performance of FLAD with the baseline model in the Digg dataset.

Experimental results on the Epinions dataset show that the FLAD model consistently outperforms the other four baseline models (Netwalk, Deepwalk, Node2vec, Graphsage) for different data scales. The F1-score of all models improves with the increase in the proportion of training data, suggesting that more data contributes to model per-

Table 2 Performance Comparison Between FLAD and Baseline Models on the Digg Dataset

Model	10%	20%	30%	40%	50%	60%	70%	80%	90%
Netwalk	0.34370	0.35117	0.35804	0.36550	0.36772	0.37363	0.37988	0.38619	0.38998
Deepwalk	0.33912	0.34637	0.35732	0.36682	0.37249	0.37695	0.37856	0.38268	0.39552
Node2vec	0.31612	0.33256	0.34335	0.35663	0.36770	0.37225	0.37863	0.38664	0.39313
Graphsage	0.32228	0.34912	0.36361	0.36802	0.37514	0.38113	0.39239	0.39887	0.40131
FLAD	0.35620	0.36109	0.36360	0.37205	0.38504	0.39024	0.39401	0.39919	0.40856

formance.FLAD is already higher than the other models initially, and the advantage of FLAD gradually widens with the increase in the proportion of data. At 90% data proportion, FLAD's F1-score improves by 6.31% over Netwalk, 5.87% over Deepwalk, 3.68% over Node2vec, and 2.3% over Graphsage. In particular, FLAD consistently leads at 50% and 70% data ratios.

Although FLAD performs well across all data proportions, the performance gap between FLAD and GraphSAGE slightly narrows as the data volume increases. For example, when using 70% of the data, the difference between FLAD and GraphSAGE is 0.01356, whereas at 80%, the gap reduces to 0.0104. This indicates that FLAD's relative advantage diminishes at higher data proportions. Among all baseline models, Node2Vec performs the weakest, especially at lower data proportions, likely due to its random walk-based strategy struggling with sparse data. In contrast, GraphSAGE, as a graph neural network-based model, demonstrates strong performance in handling complex graph structures. It approaches FLAD's performance, particularly at higher data proportions. Overall, FLAD achieves the best performance on the Epinions dataset, with its advantages becoming even more pronounced at higher data proportions.

With the Digg dataset at 90% data scale, FLAD's F1-score improves by 4.76% over Netwalk, 3.30% over Deepwalk, 3.92% over Node2vec, and 1.81% over Graphsage. FLAD performs consistently throughout the range of data scales. Although FLAD consistently outperforms, the gap between it and Graphsage narrows slightly as data increases. This suggests that at higher data ratios, FLAD's relative advantage diminishes but still leads to overall performance. Node2vec performs the weakest of all the models, especially at low data ratios, which may be related to the fact that its strategy based on randomized wandering does not work well when the data is sparse. In this experiment, the number of training iterations of the model is set to 50, and the number of training iterations of the separate models is set to 50. They are trained on Epinions and Digg datasets, respectively, and the results are shown in Fig. 7.

As the number of iterations increases, both datasets show significant improvement in accuracy, indicating that the model gradually achieves effective learning during the training process. The accuracy of the Epinions dataset (blue curve) is consistently higher than that of the Digg dataset (orange curve), and it converges faster, with a significant improvement in accuracy in the first 20 iterations. In contrast, the Digg dataset's accuracy improves more slowly in the early stages, but converges gradually faster in the later stages, eventually approaching the level of the Epinions dataset. This result suggests that despite the difference in initial performance between the two data sets, the model in the Epinions dataset demonstrates a more stable and efficient learning capability as training progresses. The possible reason for this is that the Epinions dataset differs from the Digg dataset in terms of the complexity of the features and data, resulting in easier learning of effective patterns

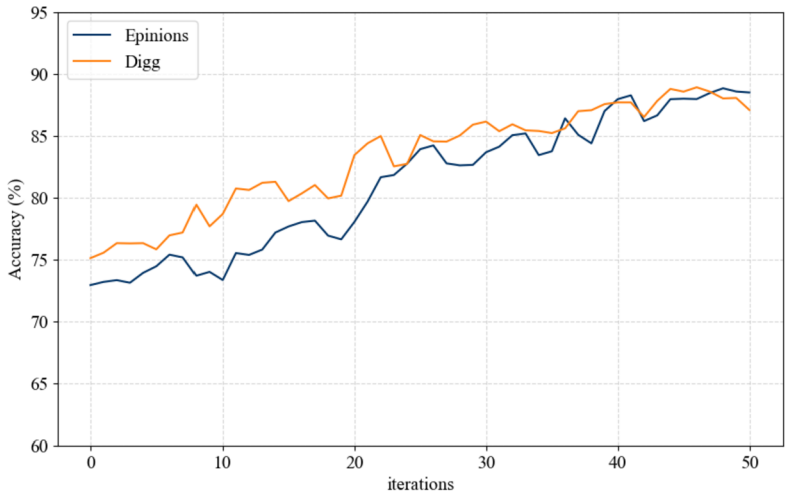

Figure 7 Relationship Between Accuracy and Number of Iterations in Model Training

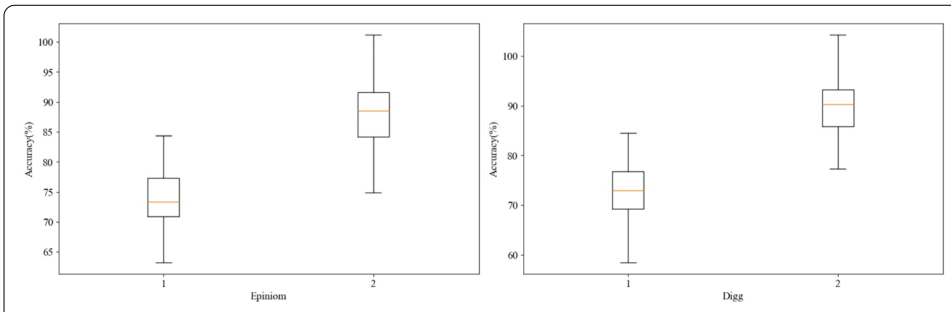

Figure 8 Comparison of Accuracy Between the Initialized Global Model and the Iterated Global Model

during training. Overall, the performance of the models during training highlights the impact of the features and complexity of the Epinions dataset on learning effectiveness.

Furthermore, our analysis suggests that the final effect of the model is correlated with the state of the dataset itself. The Epinions and Digg datasets, although both constructed based on real social networks, do not include the labeling of real anomalies. The anomalies labeled in the datasets were obtained using an anomaly injection algorithm, taking into account that a certain number of anomalous elements may also be present in the original network. When training is performed, the model is trained on a nominal training set that is considered to contain only normal elements. Therefore, the presence of anomalies in the dataset itself could potentially impact the model's effectiveness.

We conduct ablation experiments on FLAD to compare the performance of the initialized global model and the unified model after aggregation via federated learning on the Epinions and Digg datasets. The experimental results show that the accuracy of the unified model after federated learning aggregation is significantly improved on both datasets. Specifically, on the Epinions dataset, the accuracy is improved by 20.49%, while on the Digg dataset, the accuracy is improved by 23.13%. Figure 8 illustrates the accuracy of the initialized global model compared to the unified model after aggregation.

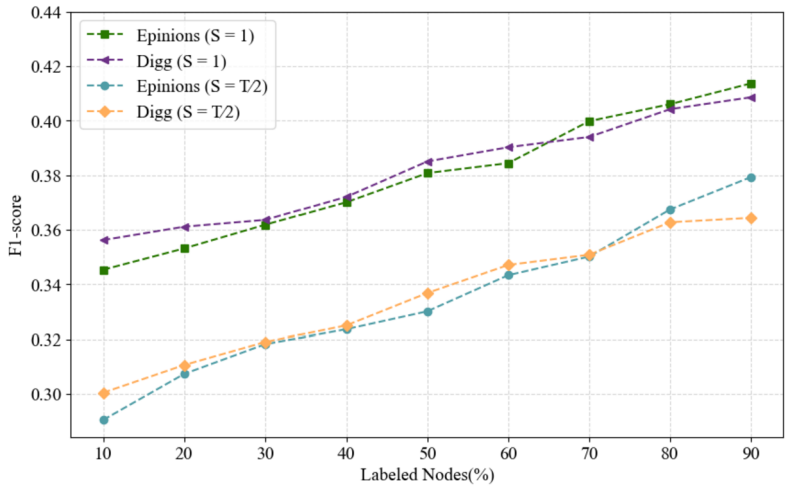

Figure 9 F1-score comparison under different step sizes S on datasets

To evaluate the impact of the sliding window step size S on detection performance, we conduct experiments on both the Epinions and Digg datasets under two settings: $S = 1$ and $S = T/2$. Figure 9 presents the F1-score variations with increasing proportions of labeled nodes. Across both datasets, models using $S = 1$ consistently outperform those using $S = T/2$, demonstrating the benefit of maximum temporal overlap in preserving behavioral continuity and capturing fine-grained anomaly signals. Specifically, on the Epinions dataset, the F1-score improves from 0.345 to 0.414 under $S = 1$, compared to 0.290 to 0.379 under $S = T/2$, indicating a relative gain of approximately 8.4% at the highest labeled ratio. Similarly, on the Digg dataset, the F1-score rises from 0.356 to 0.409 with $S = 1$, versus 0.300 to 0.364 with $S = T/2$. This consistent margin suggests that smaller step sizes are more effective at capturing long-range dependencies and mitigating global context fragmentation caused by window segmentation.

This increase in accuracy may be related to repeated iterations and parameter optimization during federated learning. In each iteration round of federated learning, the local model is continuously updated by training on the local data, which allows the global model to better adapt to the diversity and complexity of the data. After multiple rounds of optimization, the global model performs better in capturing data patterns and improving classification performance. Overall, the iterative global model shows more stable and efficient performance, further validating the key role of iterative optimization in improving model accuracy. In addition, we take a holistic perspective and divide the user behavior data in the dataset by user behavior time, and randomly select a subset of 1000 to verify the accuracy of different local social network training and compare them with the initialized abnormal behavior detection model (k=100), the detailed results are shown in Fig. 10.

As shown, the model's accuracy exhibits significant changes as the number of local platforms involved increases. The accuracy peaks at K=50 for the Epinions dataset and the Digg dataset. However, the accuracy leveled off as the number of participants exceeded 50 and decreased at K=1000. This phenomenon is closely related to the learning ability of the model. When the number of participants is small, the data samples are insufficient, resulting in limited learning ability of the model; while when the number of participants is too large, too much data may lead to overfitting of the model, which in turn affects

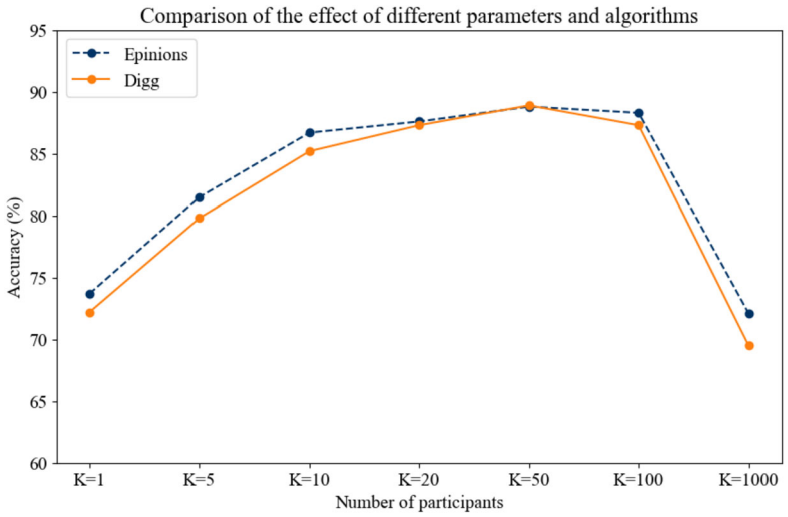

Figure 10 Flowchart of Subgraph Partitioning Based on Sliding Window

the performance. Overall, the accuracies of both datasets show a similar trend, i.e., the best performance is achieved at K=50. This result suggests that appropriately increasing the number of local platform participants can effectively improve model performance. However, when the number of participants is too large, the model performance may be affected by data redundancy, leading to overfitting performance. In conclusion, the effect of the number of local platform participants on model performance is significant, and the reasonable selection of the number of participants during training can optimize model performance to a certain extent, but too many participants may lead to adverse effects. Therefore, in practical applications, choosing the appropriate number of participants is important for improving the accuracy and avoiding model overfitting.

We analyze the computational complexity of our proposed framework. Our method involves several modules with relatively high computational overhead; among these, the following components are the primary contributors and warrant further optimization. First, the dual-path GCN encoders, designed to capture both structural and spectral representations of dynamic subgraphs, incur a complexity of $\mathcal{O}(K(|E|d + |V|d^2))$, where K is the number of subgraphs, and $|V|$, $|E|$, and d represent the number of nodes, edges, and feature dimensions, respectively. Second, the temporal modeling using LSTM introduces an additional cost of $\mathcal{O}(Ktd^2)$, especially pronounced as the time window size t increases. Moreover, federated training introduces non-negligible overhead, including client-side updates of $\mathcal{O}(RMEP)$ and server-side aggregation of $\mathcal{O}(RMP)$, where R is the number of communication rounds, M the number of clients, E the number of local epochs, and P the number of model parameters. While certain modules, such as subgraph similarity calculation and the encoder-decoder reconstruction, are relatively lightweight and executed only once during preprocessing, the overall computational cost of the system remains substantial. We acknowledge that the current version of our framework imposes significant computational demands, particularly in large-scale or resource-constrained environments. This motivates our future research direction toward improving computational efficiency through structural simplification, lightweight modeling, and adaptive computation strategies.

5 Conclusion

This paper conducts an in-depth study on the problem of anomaly behavior detection across social networks. This paper conducts an in-depth study on the problem of anomaly behavior detection across social networks. Aiming at the traditional centralized anomalous behavior detection method, which has low processing efficiency and fails to use the social network timing and graph structure information fully, we propose a cross-platform distributed anomalous behavior detection method based on cross-platform. We propose a cross-platform distributed anomaly behavior detection method. The method uses FedAvg for model aggregation, which avoids the exchange of raw data and effectively improves the effect of cross-platform collaborative learning. By combining sliding window and GCN, the model proposed in this paper can better mine temporal data and graph structure information of social networks. The user behavior data is divided into subgraphs by sliding window, and the GCN and LSTM models are used to learn the evolution laws of temporal and behavioral patterns, significantly improving the accuracy of abnormal behavior detection. We conducted experiments on our model on real datasets Epinions and Digg and the results show the effectiveness of the initialized dynamic graph anomalous behavior detection model. Our method improves up to 6.31% in the key performance metric of the F1-score compared to the baseline model. In addition, we conducted ablation experiments, and the results showed that the accuracy of the unified model completed by federated learning aggregation improved over the initialized global model in both cases, by 20.49% in the Epinions dataset and by 23.13% in the Digg dataset. Despite the positive results of this study, data heterogeneity and imbalance issues may affect the performance of the models, and future work could focus on optimizing the performance of the models in these complex environments. In addition, further improvement of computational efficiency and reduction of communication costs are important directions for future research.

Acknowledgements
This work is fully supported by the National Natural Science Foundation of China (62171180), in part by the Natural Science Foundation of Henan Province (252300421237), Key Research and Development Special projects of Henan Province (251111210900), in part by the Science and Technology Research Project of Henan Province (252102211014) and the Key Scientific Research Project of Colleges and Universities in Henan Province under Grant (25A510011).

Author contributions
Ling Xing, Shiyu Li, Honghai Wu, Qi Zhang, Huahong Ma, Kaikai Deng, all authors contributed to the final manuscript. The published data notes will be linked to the research article the data support.

Data availability
The datasets during the current study are available in the Digg and Epinions.

Materials availability
Not applicable

Code availability
The code reguired to reproduce these findings cannot be shared at this time as the data also forms part of an ongoing study.

Declarations

Ethics approval and consent to participate
Not applicable

Consent for publication
Not applicable

Competing interests
The authors declare no competing interests.

Author details
[1]College of Information Engineering, Henan University of Science and Technology, Kaiyuan Avenue, Luoyang, 471000 Henan, China. [2]College of Information Engineering, Southwest University of Science and Technology, Qinglong Avenue, Mianyang, 621010 Sichuan, China.

Received: 7 April 2025 Accepted: 21 August 2025 Published online: 29 August 2025

References

1. Xing L, Li S, Zhang Q, et al (2024) A survey on social network's anomalous behavior detection. Complex Intell Syst 10(4):5917–5932
2. Lin H, Liu GA, Wu JJ, et al (2020) Fraud detection in dynamic interaction network. IEEE Trans Knowl Data Eng 32(10):1936–1950
3. Ma X, Liu F, Wu J, Yang J, Xue S, Sheng QZ (2024) Rethinking unsupervised graph anomaly detection with deep learning: residuals and objectives. IEEE Trans Knowl Data Eng
4. Jin L, Chen Y, Wang T, Hui P, Vasilakos AV (2013) Understanding user behavior in online social networks: a survey. IEEE Commun Mag 51(9):144–150
5. Su X, Xue S, Liu F, Wu J, Yang J, Zhou C, Hu W, Paris C, Nepal S, Jin D, et al (2022) A comprehensive survey on community detection with deep learning. IEEE Trans Neural Netw Learn Syst
6. Cai B, Wang M, Chen Y, Hu Y, Liu M (2022) Mff-net: a multi-feature fusion network for community detection in complex network. Knowl-Based Syst 252:109408
7. Xing L, Huang Y, Zhang Q, Wu H, Ma H, Zhang X (2024) A counterfactual inference-based social network user-alignment algorithm. IEEE Trans Comput Soc Syst
8. Ruan ZY, Yu B, Shu XC, et al (2020) The impact of malicious nodes on the spreading of false information. Chaos 30(8):8
9. Vosoughi S, Roy D, Aral S (2018) The spread of true and false news online. Science 359(6380):1146–1151
10. Hu WB, Wang H, Qiu ZY, et al (2017) An event detection method for social networks based on hybrid link prediction and quantum swarm intelligent. World Wide Web 20(4):775–795
11. Persia F, Helmer S (2018) A framework for high-level event detection in a social network context via an extension of iseql. In: 2018 IEEE 12th International Conference on Semantic Computing (ICSC). IEEE, pp 140–147
12. Cheng S (2022) Research on data privacy protection technology of social network users based on differential disturbance. Ain Shams Eng J 13(5):101745
13. McMahan B, Moore E, Ramage D, Hampson S, Arcas BA (2017) Communication-efficient learning of deep networks from decentralized data. In: Proceedings of the 20th international conference on artificial intelligence and statistics (AISTATS). PMLR, pp 1273–1282
14. Mu J, Zhang XC, Li YG, et al (2021) Deep neural network for text anomaly detection in siot. Comput Commun 178:286–296
15. Qasim R, Bangyal WH, Alqarni MA, et al (2022) A fine-tuned bert-based transfer learning approach for text classification. J Healthcare Eng 2022:17
16. Al-Qurishi M, Hossain MS, Alrubaian M, et al (2018) Leveraging analysis of user behavior to identify malicious activities in large-scale social networks. IEEE Trans Ind Inform 14(2):799–813
17. Drif A, Hamida ZF, Giordano S (2019) Fake news detection method based on text-features, pp 27–32
18. Deepak S, Chitturi B (2020) Deep neural approach to fake-news identification. Proc Comput Sci 167:2236–2243
19. Keshavarzi A, Kannan N, Kochut K (2021) Regpattern2vec: link prediction in knowledge graphs. In: 2021 IEEE international IoT, electronics and mechatronics conference (IEMTRONICS). IEEE, pp 1–7
20. Lin H, Liu GA, Wu JJ, et al (2020) Fraud detection in dynamic interaction network. IEEE Trans Knowl Data Eng 32(10):1936–1950
21. Bahri L, Carminati B, Ferrari E (2018) Knowledge-based approaches for identity management in online social networks. Wiley Interdiscip Rev Data Min Knowl Discov 8(5):10
22. Fan HY, Zhang FB, Li ZY, et al (2020) Anomalydae: dual autoencoder for anomaly detection on attributed networks. In: IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP). IEEE, pp 5685–5689
23. Zhang L, Wu B, Dong P (2025) Attribute graph anomaly detection utilizing memory networks enhanced by multi-embedding comparison. Neurocomputing 129762
24. Guo J, Tang S, Li J, Pan K, Wu L (2023) Rustgraph: robust anomaly detection in dynamic graphs by jointly learning structural-temporal dependency. IEEE Trans Knowl Data Eng 36(7):3472–3485
25. Xue LG, Chen Y, Luo MN, et al (2020) An anomaly detection framework for time-evolving attributed networks. Neurocomputing 407:39–49
26. Yasami Y, Safaei F (2018) Detecting chaotic behaviors in dynamic complex social networks using a feature diffusion-aware model. Chaos 28(6):9
27. Lei T, Ou M, Gong C, Li J, Yang K (2024) An unsupervised deep global–local views model for anomaly detection in attributed networks. Knowl-Based Syst 300:112185
28. Li T, Sahu AK, Zaheer M, Sanjabi M, Talwalkar A, Smith V (2020) Federated optimization in heterogeneous networks. Proc Mach Learn Syst 2:429–450
29. Smith V, Chiang C-K, Sanjabi M, Talwalkar AS (2017) Federated multi-task learning. Adv Neural Inf Process Syst 30
30. Wang H, Yurochkin M, Sun Y, Papailiopoulos D, Khazaeni Y (2020) Federated learning with matched averaging. arXiv preprint. arXiv:2002.06440
31. Arivazhagan MG, Aggarwal V, Singh AK, Choudhary S (2019) Federated learning with personalization layers. arXiv preprint. arXiv:1912.00818
32. T. Dinh C, Tran N, Nguyen J (2020) Personalized federated learning with Moreau envelopes. Adv Neural Inf Process Syst 33:21394–21405
33. Li X, Jiang M, Zhang X, Kamp M, Dou Q (2021) Fedbn: federated learning on non-iid features via local batch normalization. arXiv preprint. arXiv:2102.07623

34. Bonawitz K, Ivanov V, Kreuter B, Marcedone A, McMahan HB, Patel S, Ramage D, Segal A, Seth K (2017) Practical secure aggregation for privacy-preserving machine learning. In: Proceedings of the 2017 ACM SIGSAC conference on computer and communications security, pp 1175–1191
35. Li T, Sanjabi M, Beirami A, Smith V (2019) Fair resource allocation in federated learning. arXiv preprint. arXiv:1905.10497
36. Acar DAE, Zhao Y, Navarro RM, Mattina M, Whatmough PN, Saligrama V (2021) Federated learning based on dynamic regularization. arXiv preprint. arXiv:2111.04263
37. Zhou X, Liang W, Kevin I, Wang K, Yada K, Yang LT, Ma J, Jin Q (2025) Decentralized federated graph learning with lightweight zero trust architecture for next-generation networking security. IEEE J Sel Areas Commun
38. Zhou J, Wu J, Ni J, Wang Y, Pan Y, Su Z (2025) Protecting your attention during distributed graph learning: efficient privacy-preserving federated graph attention network. IEEE Trans Inf Forensics Secur
39. Li Z, Li C, Li M, Yang L, Weng J (2025) Federated graph transformer with mixture attentions for secure graph knowledge fusions. Inf Fusion: 102954
40. Chen C, Hu W, Xu Z, Zheng Z (2021) Fedgl: federated graph learning framework with global self-supervision. arXiv preprint. arXiv:2105.03170
41. Chen F, Li P, Miyazaki T, Wu C (2021) Fedgraph: federated graph learning with intelligent sampling. IEEE Trans Parallel Distrib Syst
42. Ni X, Xu X, Lyu L, Meng C, Wang W (2021) A vertical federated learning framework for graph convolutional network. arXiv preprint. arXiv:2106.11593
43. Pei Y, Mao R, Liu Y, Chen C, Xu S, Qiang F, Tech BE (2021) Decentralized federated graph neural networks. In: IJCAI workshops
44. Zhang K, Yang C, Li X, Sun L, Yiu SM (2021) Subgraph federated learning with missing neighbor generation. In: NeurIPS (federated learning for graphs workshop)
45. He C, Balasubramanian K, Ceyani E, Yang C, Xie H, Sun L, He L, Yang L, Yu PS, Rong Y, et al (2021) Fedgraphnn: a federated learning system and benchmark for graph neural networks. arXiv preprint. arXiv:2104.07145
46. Zhang K, Yang C, Li X, Sun L, Yiu SM (2021) Subgraph federated learning with missing neighbor generation. Adv Neural Inf Process Syst 34:6671–6682
47. Cai J, Zhang Y, Fan J, Ng S-K (2024) Lg-fgad: an effective federated graph anomaly detection framework. In: Proceedings of the international joint conference on artificial intelligence
48. Jiang M, Jung T, Karl R, Zhao T (2022) Federated dynamic graph neural networks with secure aggregation for video-based distributed surveillance. ACM Trans Intell Syst Technol 13(4):1–23
49. Usman M, Lee Y (2025) Dfdg: adaptive federated learning for dynamic graph-based traffic forecasting. Knowl-Based Syst: 114019
50. Perozzi B, Al-Rfou R, Skiena S (2014) Deepwalk: online learning of social representations. In: Proceedings of the 20th ACM SIGKDD international conference on knowledge discovery and data mining, pp 701–710
51. Grover A, Leskovec J (2016) node2vec: scalable feature learning for networks. In: Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining, pp 855–864
52. Hamilton W, Ying Z, Leskovec J (2017) Inductive representation learning on large graphs. Adv Neural Inf Process Syst 30
53. Yu W, Cheng W, Aggarwal CC, Zhang K, Chen H, Wang W (2018) Netwalk: a flexible deep embedding approach for anomaly detection in dynamic networks. In: Proceedings of the 24th ACM SIGKDD international conference on knowledge discovery & data mining, pp 2672–2681

Publisher's note