LAN: Learning Adaptive Neighbors for Real-Time Insider Threat Detection

Xiangrui Cai, Yang Wang, Sihan Xu, Hao Li, Ying Zhang, Zheli Liu, and Xiaojie Yuan

***Abstract*— Enterprises and organizations are faced with potential threats from insider employees that may lead to serious consequences. Previous studies on insider threat detection (ITD) mainly focus on detecting abnormal users or abnormal time periods (e.g., a week or a day). However, a user may have hundreds of thousands of activities in the log, and even within a day there may exist thousands of activities for a user, requiring a high investigation budget to verify abnormal users or activities given the detection results. On the other hand, existing works are mainly post-hoc methods rather than real-time detection, which can not report insider threats in time before they cause loss. In this paper, we conduct the first study towards real-time ITD at activity level, and present a fine-grained and efficient framework LAN. Specifically, LAN simultaneously learns the temporal dependencies within an activity sequence and the relationships between activities across sequences with graph structure learning. Moreover, to mitigate the data imbalance problem in ITD, we propose a novel hybrid prediction loss, which integrates self-supervision signals from normal activities and supervision signals from abnormal activities into a unified loss for anomaly detection. We evaluate the performance of LAN on two widely used datasets, i.e., CERT r4.2 and CERT r5.2. Extensive and comparative experiments demonstrate the superiority of LAN, outperforming 9 state-of-the-art baselines by at least 8.43% and 6.35% in AUC for real-time ITD on CERT r4.2 and r5.2, respectively. Moreover, LAN can be also applied to post-hoc ITD, surpassing 8 competitive baselines by at least 7.70% and 4.03% in AUC on two datasets. Finally, the ablation study, parameter analysis, and compatibility analysis evaluate the impact of each module and hyper-parameter in LAN. The source code can be obtained from https://github.com/Li1Neo/LAN.**

***Index Terms*— Insider threat detection, activity-level detection, real-time detection, graph structure learning, class imbalance.**

I. Introduction

MODERN information systems are vulnerable to attacks from insider employees who have authorized access to system, data, or internal network [1]. Such insider threats may corrupt the integrity, confidentiality, and availability of systems [2]. From 2020 to 2022, the number of insider threat incidents had increased by 44%, resulting in the average costs rising by 34%, from $11.45 million to $15.38 million [3]. Due to the destructive effects of insider threats, much effort has been devoted to Insider Threat Detection (ITD) to prevent from unpredictable impact brought by insider threats. Specifically, existing works on ITD can be grouped into three categories, i.e., the feature engineering-based methods [4], [5], the sequence-based methods [6], [7], [8], and the graph-based methods [9], [10], [11]. The first group extracts features such as the number of file accesses after work hours [4], and then trains a machine learning model to detect insider threats. The second group collects user activity sequence data for each time period (e.g., a day [6] or a session [8]), and then trains a sequence-based model to predict whether a specific time period is abnormal or not. Inspired by graph-based methods in the field of intrusion detection [12], [13], the third group proposes constructing a graph that incorporates relationships between users [9], [10] or activities [11] across different sequences, to detect insider threats.

Despite the progress, existing approaches are faced with two problems, which limit their practical application in real-world ITD. **First**, most studies on ITD focus on detecting either abnormal users [10] or abnormal time periods (e.g., a week or a day). However, a user may have hundreds of thousands of activities in the log, where an activity is the smallest unit of user behaviors (e.g., `open file`). Even within a day there may exist thousands of activities for a user. As a result, these approaches are too course-grained to be adopted, requiring a high investigation budget to verify the abnormal users or activities from the detection results. Actually, as shown in Figure 1a, an insider threat accident can be directly reflected by a set of abnormal activities of a user, which are more fine-grained and accurate. **Second**, existing works are mainly post-hoc methods rather than real-time detection. However, it is more desirable to report insider threats in time before they cause loss [14]. Figure 1b and Figure 1c illustrate *post-hoc* ITD and *real-time* ITD, respectively. It can be seen that *post-hoc* ITD identifies abnormal users or activities after an insider threat accident occurred and all the related data have been collected (e.g., logs during the past year). In contrast, *real-time* ITD monitors the entire system and detects ITD at runtime. Once an insider threat accident occurs, it is identified and reported promptly, which can effectively prevent the organization or enterprise from financial loss.

To address these issues, instead of detecting whether a user or a time period is abnormal, this paper focuses on more fine-grained and real-time ITD, i.e., detecting whether

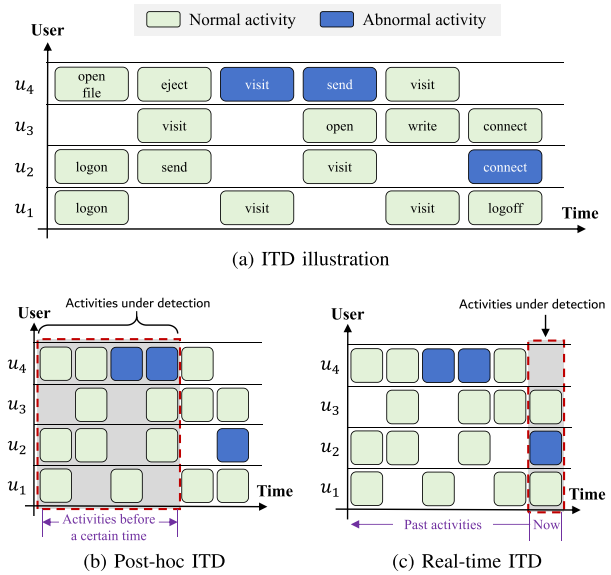
Received 17 March 2024; revised 14 September 2024; accepted 16 October 2024. Date of publication 31 October 2024; date of current version 13 November 2024. This work was supported in part by the National Key Research and Development Program of China under Grant 2022YFB3103202; in part by the National Natural Science Foundation of China under Grant 62202245 and Grant 72342017; and in part by the Natural Science Foundation of Tianjin, China, under Grant 23JCQNJC01960. The associate editor coordinating the review of this article and approving it for publication was Dr. Weizhi Meng. *(Corresponding author: Sihan Xu.)*

Xiangrui Cai, Ying Zhang, and Xiaojie Yuan are with the VCIP, DISSec, TMCC, TBI Center, College of Computer Science, Nankai University, Tianjin 300350, China (e-mail: caixr@nankai.edu.cn; yingzhang@nankai.edu.cn; yuanxj@nankai.edu.cn).

Yang Wang, Sihan Xu, and Zheli Liu are with the DISSec, TKLNDST, College of Cyber Science, Nankai University, Tianjin 300350, China (e-mail: wangyang@dbis.nankai.edu.cn; xusihan@nankai.edu.cn; liuzheli@nankai.edu.cn).

Hao Li is with the Science and Technology on Communication Networks Laboratory, Shijiazhuang 050081, China (e-mail: cuclihao@cuc.edu.cn).

Digital Object Identifier 10.1109/TIFS.2024.3488527



(a) ITD illustration

(b) Post-hoc ITD

(c) Real-time ITD

Fig. 1. (a) Illustration of Activity-level ITD. Activity-level ITD aims at discovering abnormal activities inside the system (shown in blue boxes). (b) Post-hoc ITD. It is usually deployed for retrospective detection, discovering abnormal activities in a past period. (c) Real-time ITD. It is usually applied to detect abnormality of a current activity.

a current activity is abnormal. Since activity is the smallest unit of user behaviors, activity-level ITD can provide more precise information to verify potential insider threats and attain efficient real-time detection. Specifically, this paper proposes a novel activity modeling framework named LAN for real-time ITD. It consists of three modules, i.e., Activity Sequence Modeling, Activity Graph Learning, and Anomaly Score Prediction. Since user activities occur sequentially and there are temporal dependencies between user activities, given a user activity sequence, the Activity Sequence Modeling module utilizes a sequence encoder with attentive aggregation operation to model the temporal dependencies and obtain the representation of each activity. Then, to further incorporate relationships between different activity sequences, we propose a graph structure learning model in the Activity Graph Learning module that automatically constructs user activity graphs, modeling the relationships between different activity sequences directly without any prior knowledge. After that, the Anomaly Score Prediction module employs a graph neural network to aggregate neighbors in the activity graph, so as to enhance the representation of the current activity. Finally, the anomaly score is obtained by calculating the probability of the next activity occurrence based on the activity representation.

Moreover, the data imbalance problem also poses a great challenge to activity-level ITD. The reason behind this is that an employee usually has many activity sequences, and an activity sequence usually contains many activities, while only a very small subset of employees may have occasional abnormal activities. As a result, the data imbalance between normal and abnormal samples for activity-level ITD is even worse than the problem for the user- or period-level ITD. This imbalance leads the model to lean towards predicting normal activities, thereby reducing the overall detection rate. To alleviate this issue, we introduce a novel hybrid loss, which integrates both supervision of abnormal activities and self-supervision of the normal activity sequence simultaneously.

We evaluated LAN on two widely-used public datasets (i.e., CERT r4.2 and CERT r5.2) and compared LAN against 9 baselines for real-time ITD. We also applied LAN for post-hoc ITD with slight modifications and compared it with 8 state-of-the-art approaches. The experimental results demonstrate that LAN outperforms all baselines with average improvements of 9.53% and 6.55% in AUC for real-time ITD and post-hoc ITD, respectively. We further conduct ablation studies and parameter analysis to evaluate the effectiveness of each module, hyper-parameter, and the proposed hybrid prediction loss for the data imbalance problem in real-time ITD.

In summary, we made the following novel contributions:

- To our best knowledge, we conduct the first study towards *activity-level real-time* insider threat detection. Specifically, we present a fine-grained and efficient framework named LAN, which employs graph structure learning to learn user activity graph adaptively, avoiding the bias introduced by manual graph construction.
- To alleviate the significant imbalance between normal and abnormal activities, we propose a novel hybrid prediction loss, which integrates self-supervision signals from normal activities and supervision signals from abnormal activities into a unified loss for anomaly detection.
- Extensive and comparative experiments demonstrate the superiority of LAN, outperforming 9 state-of-the-art baselines by at least 9.92% and 6.35% in AUC for real-time ITD on CERT r4.2 and r5.2, respectively. Moreover, LAN can be also applied to post-hoc ITD, surpassing 8 competitive baselines by at least 7.70% and 4.03% in AUC on two datasets.

II. Preliminaries

In this section, we first introduce the definition and attack vector of insider threat. Then, we describe the capabilities of attackers and formulate the problem of activity-level ITD.

A. Insider Threat Definition

An insider is defined as "a current or former employee, contractor, or business partner who has or had authorized access to an organization's network, system, or data" according to the technical report from the CERT Coordination Center [2]. An insider threat refers to an insider who has "intentionally exceeded or intentionally used the authorized access to negatively affect the confidentiality, integrity, or availability of the organization's information or information systems" [2].

B. Attack Vector

Insider threats are challenging to detect since they come from people who already have or had authorized access. Possible attack vectors associated with insider threats can be seen as follows.

- Data exfiltration. It means unauthorized copying or transferring the data of an organization [15].
- Data deletion. An insider may intentionally or unintentionally delete important data of an organization.
- Compromised passwords. It means an insider may steal the password of a colleague to disguise the colleague and commit fraud.

- Modification of critical data. It means illegally modifying critical data in the server or system [2].

C. The Capabilities of Attackers

We assume that the attackers have or had authorized access to the network, system, or data of an organization. Their capabilities depend on their positions and job responsibilities. For example, insiders with system administrator privileges have the capability to deliberately destroy or damage the system. We also assume that when the account is compromised, the attackers may disguise themselves as normal employees to commit fraud or steal confidential information.

Given an information system that can be accessed by N users, we denote the user set by $\mathcal{U} = \{u_i \mid i \in \mathbb{N}^+, i \leq N\}$. The system records the activity sequence of each user chronologically, e.g., `logon`, `visit website`, and `copy file`. We use $\mathcal{A} = \{a_i \mid i \in \mathbb{N}^+, i \leq M\}$ to denote the set of all activities inside the system, where M is the number of unique activities. An activity sequence of a user $u \in \mathcal{U}$ is denoted by $S^u = \{a_1^u, a_2^u, \ldots, a_{n_u}^u\}$, where $a_i^u \in \mathcal{A} (i = 1, 2, \ldots, n_u)$ and n_u is the activity sequence length for user u. The activities in S^u are arranged in chronological order. Note that the activity sequence length (i.e., n_u) could vary for different users. The activity sequences of all users form the whole sequence set $\mathcal{D} = \{S^u \mid u \in \mathcal{U}\}$ within the system.

D. Post-Hoc Insider Threat Detection

Post-hoc ITD aims at retrospectively determining whether insider threats once occurred. The input of post-hoc ITD is user activities during a past period, and the goal of post-hoc ITD is to determine whether the activities in this period are abnormal or not. Formally, given a time stamp t and the set of user activity sequences $\mathcal{D}_t$ where all activities were collected before the time stamp t, post-hoc ITD typically trains a model $\mathcal{F}_{\text{PH}}$ to find all the abnormal activities in $\mathcal{D}_t$.

E. Real-Time Insider Threat Detection

Real-time ITD aims at identifying and reporting an insider threat accident promptly at runtime. The input of real-time ITD is the previous activities and a current activity, which are usually monitored by the system. The output of real-time ITD is the prediction result about whether the current activity is abnormal or not. Formally, real-time ITD utilizes user activity sequence before the current time t, which is denoted by $\mathcal{D}_t$, to detect the abnormality of the current activity of each user. Let $\mathcal{A}_t$ be the set of activities occurred at time t of all users in the system, i.e., $\mathcal{A}_t = \{a^u \mid u \in \mathcal{U}, \text{time}(a^u) = t\}$, where a^u represents one activity of user u, and $\text{time}(a^u)$ refers to the occurrence time of a^u. Real-time ITD aims at developing an anomaly detection model $\mathcal{F}_{\text{RT}}$ based on $\mathcal{D}_t$ to discover abnormal activities in $\mathcal{A}_t$.

III. Approach

A. Overview of LAN

To ascertain the normalcy of the ongoing activity, we model the preceding sequence of activities and anticipate the current activity. A heightened probability in the prediction suggests that the current activity is less likely to be abnormal. Formally, assuming user u has generated an activity sequence of length n as $\{a_1^u, a_2^u, \ldots, a_n^u\}$, the goal of real-time ITD is to learn the activity patterns of normal users, model the probability of current activity a_{n+1}^u, and classify it as an insider threat if the probability falls below the detection threshold.

Figure 2 depicts the overall architecture of LAN, which consists of three modules, i.e., Activity Sequence Modeling, Activity Graph Learning, and Anomaly Score Prediction. Learning the representation of a user's preceding activity sequence lies at the essence of real-time ITD. In this paper, LAN initially employs a sequence encoder as the first step to obtain preliminary representations of the activities. However, utilizing only one sequence encoder can capture information for prediction solely from the user's historical activity sequence. To enhance activity representations and reduce false positives, LAN queries the activity vector pool and introduces related (similar) activity vectors. Then LAN constructs a graph among the activities and learns the graph structure adaptively. Finally, to resolve the imbalance problem between normal and abnormal activities, we propose a hybrid prediction loss to incorporate supervised information from abnormal samples in addition to self-supervision with only normal samples.

The LAN architecture can be applied to both real-time ITD and post-hoc ITD. However, these two scenarios involve distinct prediction paradigms while utilizing the same sequence encoder. Specifically, they employ next activity prediction for real-time ITD and activity cloze for post-hoc ITD. We introduce the details of each module of LAN for real-time ITD from Section III-B to III-D We explain the differences between post-hoc ITD and real-time ITD in Section III-E.

B. Activity Sequence Modeling

In this section, we model the historical activity sequence of a user to learn the temporal dependencies among activities and obtain a preliminary representation of the sequence. Specifically, we employ a sequential model for encoding activities and a multi-head attentive pooling layer to aggregate the information of the whole sequence.

1) Sequence Encoder: Sequence models have achieved tremendous success in the field of natural language processing due to their strong ability to learn contextual representations. We use a sequence encoder to achieve a representation of the user's previous activities $S = \{a_1, \ldots, a_n\}$. We omit the superscript u for brevity in the absence of ambiguity. First, we assign a numeric token to each user activity according to its type and timestamp. Recall that the activity type could be `open file`, `connect device`, `login`, etc., as illustrated in Figure 1a. Following [16], we divide a day into 24 time slots by hour, and map the timestamp to an integer between 0 and 23. Formally, the activity code c_i is obtained by:

$$c_i = \text{type}(a_i) \times 24 + \text{time}(a_i), \tag{1}$$

where $\text{type}(a_i)$ represents the activity type ID of a_i, and $\text{time}(a_i)$ the corresponding time slot of a_i's timestamp. By converting a_i to c_i in this manner, we integrate the tasks of predicting activity type and occurrence time slot.

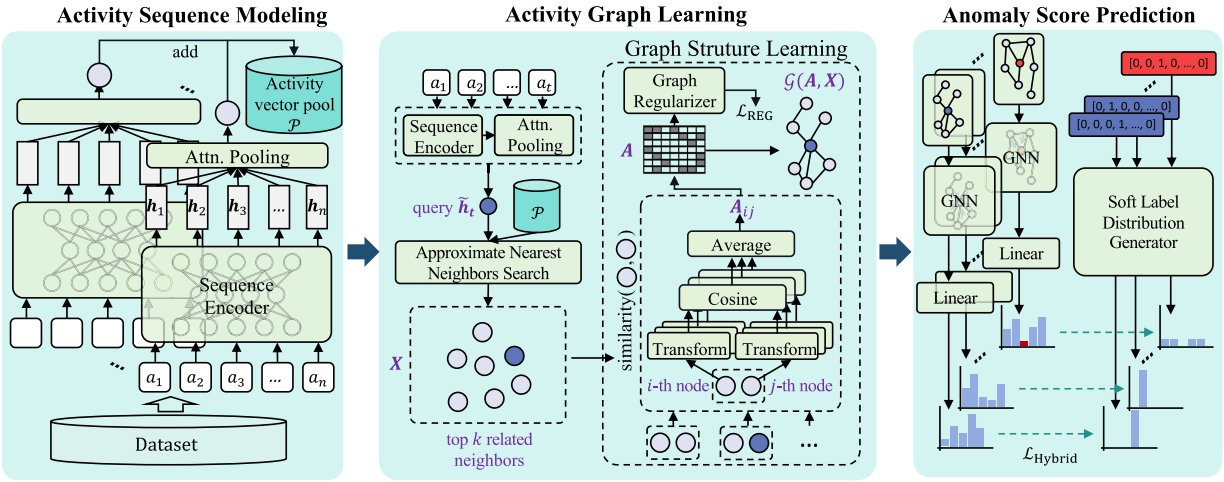


Fig. 2. Overall architecture of LAN.

Then we project the activity code to an embedding space with an embedding layer. We initialize an embedding matrix $\boldsymbol{W}_E = [\boldsymbol{w}_1, \boldsymbol{w}_2, \ldots, \boldsymbol{w}_M] \in \mathbb{R}^{d \times M}$, where d is the size of embedding vectors, and M the number of unique activities in the system. For each activity code c_i, its embedding $\boldsymbol{e}_i$ is obtained by looking up the embedding matrix, i.e.,

$$\boldsymbol{e}_i = \text{Embedding}(c_i) = \boldsymbol{w}_{c_i} \in \mathbb{R}^d , \tag{2}$$

where $\boldsymbol{e}_i$ refers to the embedding vector of activity a_i.

To achieve activity representations, several sequence modeling methods can be employed, e.g, Long-Short Term Memory (LSTM) [17], Gated Recurrent Unit (GRU) [18], Transformer [19], etc. Without loss of generality, we use LSTM for instance. The LSTM encodes the sequence of activity embeddings $\boldsymbol{E} = (\boldsymbol{e}_1, \boldsymbol{e}_2, \ldots, \boldsymbol{e}_n)$ and generates a sequence of hidden states correspondingly:

$$\boldsymbol{H} = (\boldsymbol{h}_1, \boldsymbol{h}_2, \ldots, \boldsymbol{h}_n) = \text{LSTM}(\boldsymbol{e}_1, \boldsymbol{e}_2, \ldots, \boldsymbol{e}_n) . \tag{3}$$

We investigate the performance of different sequence encoders on LAN in Section IV-F.

2) Multi-Head Attentive Pooling: To enable the model to focus on different aspects of the past hidden states while capturing various dependency relationships between historical user activities, we employ multi-head attentive pooling to aggregate contextual information of past hidden states.

Given a query $\boldsymbol{q}$ and a set of keys $\boldsymbol{K}$ and values $\boldsymbol{V}$, where $\boldsymbol{q} \in \mathbb{R}^d$, $\boldsymbol{K}, \boldsymbol{V} \in \mathbb{R}^{n \times d}$, the scaled dot-product attention operation [19] first computes attention scores as follows:

$$\alpha(\boldsymbol{q}, \boldsymbol{K}_i) = \frac{\boldsymbol{q}^\top \boldsymbol{K}_i}{\sqrt{d}} . \tag{4}$$

Then it employs the normalized scores as weights to aggregate representations of preceding activities:

$$\text{Attention}(\boldsymbol{q}, \boldsymbol{K}, \boldsymbol{V}) = \sum_i \frac{\alpha(\boldsymbol{q}, \boldsymbol{K}_i)}{\sum_j \alpha(\boldsymbol{q}, \boldsymbol{K}_j)} \cdot \boldsymbol{V}_i . \tag{5}$$

Multi-head attentive pooling enhances scaled dot-product attention by dividing it into multiple heads. Each head learns different feature representations and attention weights independently, focusing on different positions and semantics in the input sequence, boosting the model's expressive power:

$$\begin{aligned} \text{MHA}(S) &= \text{Concat}(\text{head}_1, \text{head}_2, \ldots, \text{head}_h) \cdot \boldsymbol{W}^O , \\ \text{head}_i &= \text{Attention}(\boldsymbol{q}_{\text{head}_i}, \boldsymbol{K}_{\text{head}_i}, \boldsymbol{V}_{\text{head}_i}) , \\ \boldsymbol{q}_{\text{head}_i} &= \boldsymbol{q} \cdot \boldsymbol{W}_i^q , \quad \boldsymbol{K}_{\text{head}_i} = \boldsymbol{K} \cdot \boldsymbol{W}_i^K , \quad \boldsymbol{V}_{\text{head}_i} = \boldsymbol{V} \cdot \boldsymbol{W}_i^V , \end{aligned} \tag{6}$$

where $\boldsymbol{q}_{\text{head}_i} \in \mathbb{R}^{d_k}$, $\boldsymbol{K}_{\text{head}_i}, \boldsymbol{V}_{\text{head}_i} \in \mathbb{R}^{n \times d_k}$, $d_k = \frac{d}{h}$ is the dimension after projection, h is the number of heads, $\boldsymbol{W}_i^q, \boldsymbol{W}_i^K, \boldsymbol{W}_i^V \in \mathbb{R}^{d \times d_k}$ are the weight matrices for the linear projection of the i-th head, and $\boldsymbol{W}^O \in \mathbb{R}^{hd_k \times d}$.

To consolidate the semantics of a user's activities, we consider the hidden state $\boldsymbol{h}_n$ at the last position of LSTM as the query $\boldsymbol{q}$ and all hidden states $\boldsymbol{H} = (\boldsymbol{h}_1, \boldsymbol{h}_2, \ldots, \boldsymbol{h}_n)$ as the keys $\boldsymbol{K}$ and values $\boldsymbol{V}$. Following pooling with the multi-head attention, we derive the aggregated representation $\tilde{\boldsymbol{h}}_n$ for the n-th activity.

C. Activity Graph Learning

Although the sequence-based methods can be applied to predict the next activity directly, they fail to consider the relationships among all activity sequences within the system. Such relationships could enrich the semantics of the activity representations, alleviating false positives as a result. To capture the relationships of activity sequences, we construct an activity graph specific to each detected activity. Unfortunately, constructing a static graph among activity sequences is sub-optimal. First, it is not suitable for real-time ITD, where user activities are constantly evolving. Second, static graph construction requires expert knowledge, which may result in poor scalability and high cost. Insight of this, we propose to automatically construct a dynamic graph that connects the detected activity with its closely associated activities, achieved through graph structure learning.

1) Activity Vector Pool: First, we apply the Activity Sequence Modeling module to all activity sequences within the system to achieve a pool of activity representation vectors. Specifically, we split an activity sequence of a user into several sessions, as previous studies did [16]. To achieve the representation of each activity, we further split each activity

sequence into several sub-sequences for real-time ITD. For instance, given a session of user u with a length of l, $S_{\text{session}} = \{a_1, a_2, \ldots, a_l\}$, we obtain l sub-sequences, where the i-th sub-sequence include the i-th activity and its preceding activities, i.e., $\{a_1, a_2, \ldots, a_i\}$. By applying the Activity Sequence Modeling module to these sub-sequences, we can obtain an activity vector pool within the system, which is denoted by $\mathcal{P}_{\text{RT}} \in \mathbb{R}^{M\times d}$. Recall that M is the number of unique activities in the system, and d is the size of the vectors. The vectors in the vector pool $\mathcal{P}_{\text{RT}}$ are initialized randomly and optimized together with the remaining model. We include all the vectors of normal activities in the training set into the activity vector pool and filter out duplicate user activities.

2) Graph Structure Learning: We retrieve the activity vector pool $\mathcal{P}_{\text{RT}}$ with $\tilde{\boldsymbol{h}}_n$ to obtain top k most related vectors of the detected activity a_n, i.e, $\{\boldsymbol{v}_1, \boldsymbol{v}_2, \ldots, \boldsymbol{v}_k\}, \boldsymbol{v}_i \in \mathbb{R}^d$. To be specific, we use the approximate nearest neighbor searching algorithm to reduce retrieval time. We then construct an activity graph $\mathcal{G} = (\boldsymbol{A}, \boldsymbol{X})$ based on $\tilde{\boldsymbol{h}}_n$ and the top k most related vectors, where $\boldsymbol{A} \in \mathbb{R}^{(k+1)\times(k+1)}$ is the adjacency matrix of the graph, and $\boldsymbol{X} = [\tilde{\boldsymbol{h}}_n, \boldsymbol{v}_1, \boldsymbol{v}_2, \ldots, \boldsymbol{v}_k] \in \mathbb{R}^{(k+1)\times d}$ is the node representations.

The matrix $\boldsymbol{A}$ is further optimized by the Graph Structure Learning component. It learns a function $f_{\mathcal{G}}(\cdot,\cdot)$ that maps the connectivity relationship between any two nodes to a real-valued measurement. For the i-th and j-th nodes with feature vectors $\boldsymbol{x}_i$ and $\boldsymbol{x}_j \in \boldsymbol{X}$, one simple measurement is to calculate the cosine similarity between the two vectors, i.e., $f_{\mathcal{G}}(i,j) = \cos(\boldsymbol{x}_i, \boldsymbol{x}_j)$. We harness the concept of multi-head attention by employing a multi-head variant of weighted cosine similarity, expressed as:

$$f_{\mathcal{G}}(i,j) = \frac{1}{Z}\sum_{z=1}^{Z} \cos(\boldsymbol{x}_i \odot \boldsymbol{w}_{\text{GSL}}^z, \boldsymbol{x}_j \odot \boldsymbol{w}_{\text{GSL}}^z)\,, \tag{7}$$

where $\odot$ represents the Hadamard product operation, Z is the number of attention heads, $\cos(\cdot,\cdot)$ is the cosine similarity function, and $\boldsymbol{W}_{\text{GSL}} = [\boldsymbol{w}_{\text{GSL}}^1, \boldsymbol{w}_{\text{GSL}}^2, \ldots, \boldsymbol{w}_{\text{GSL}}^Z] \in \mathbb{R}^{Z\times d}$ is the learnable weight matrix. The multi-head version of weighted cosine similarity allows the model to consider node relationships from different perspectives jointly. To ensure each element of $\boldsymbol{A}$ to be non-negative, we filter the values of $f_{\mathcal{G}}(\cdot,\cdot)$ that are negative and set a hard threshold ϵ to suppress the noise from neighbors, i.e.,

$$\boldsymbol{A}_{ij} = \begin{cases} f_{\mathcal{G}}(i,j), & f_{\mathcal{G}}(i,j) \geq \epsilon\,, \\ 0, & \text{otherwise}\,. \end{cases} \tag{8}$$

The graph structure is also learned together with other modules of LAN. It is capable of adapting to continuously occurring activities. Additionally, compared to constructing a global graph, we only construct a local graph, but incorporate the most related activity sequences, which is computationally efficient and practical.

3) Graph Regularization: Graph signals exhibit smooth variations between neighboring nodes [20]. Following [21], [22], we employ Dirichlet energy [23] to regularize the smoothness of the activity graph $\mathcal{G}$. A smaller Dirichlet energy indicates greater similarity and smoother graph signals, while a larger value indicates greater differences between adjacent nodes. The regularizer is defined as follows:

$$\mathcal{L}_D = \frac{1}{2}\sum_{i\in\mathcal{V}}\sum_{j\in\mathcal{N}_i} \boldsymbol{A}_{ij}\left\|\boldsymbol{x}_i - \boldsymbol{x}_j\right\|^2 = \text{tr}(\boldsymbol{X}^\top \cdot \boldsymbol{L} \cdot \boldsymbol{X})\,, \tag{9}$$

where $\mathcal{V}$ denotes the set of vertices in the graph $\mathcal{G}$, $\mathcal{N}_i$ represents the neighborhood of node i, $\boldsymbol{A}_{ij}$ represents the connectivity between nodes i and j in the graph, $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$ are the node representations, $\text{tr}(\cdot)$ represents the trace of a matrix, and $\boldsymbol{L} = \boldsymbol{D} - \boldsymbol{A}$ represents the Laplacian matrix of the graph, where $\boldsymbol{D}$ is the degree matrix. Besides, we can use $\hat{\boldsymbol{L}} = \boldsymbol{D}^{-\frac{1}{2}}\boldsymbol{L}\boldsymbol{D}^{-\frac{1}{2}}$ instead of $\boldsymbol{L}$ to make the smoothness invariant to node degrees [24].

Minimizing the Dirichlet energy penalizes the degree of connectivity between dissimilar nodes and encourages graphs with smooth signals to correspond to a sparse set of edges. In extreme cases, this can lead to a trivial solution, i.e., $\boldsymbol{A} = \boldsymbol{0}$. To ensure meaningful learned graphs, we impose a constraint on graph connectivity. Following [21], we add a logarithmic barrier term in the graph regularization loss:

$$\mathcal{L}_{log} = -\boldsymbol{1}^\top \cdot \log(\boldsymbol{A}\cdot\boldsymbol{1})\,. \tag{10}$$

Additionally, to directly control sparsity, We follow [22] and append the Frobenius norm:

$$\mathcal{L}_{sparsity} = \|\boldsymbol{A}\|_F^2\,. \tag{11}$$

Finally, the whole regularization loss is as follows:

$$\begin{aligned}\mathcal{L}_{\text{Reg}} &= \frac{\mu_1}{n^2}\mathcal{L}_D + \frac{\mu_2}{n}\mathcal{L}_{log} + \frac{\mu_3}{n^2}\mathcal{L}_{sparsity} \\ &= \frac{\mu_1}{n^2}\text{tr}(\boldsymbol{X}^\top\cdot\boldsymbol{L}\cdot\boldsymbol{X}) - \frac{\mu_2}{n}\boldsymbol{1}^\top\cdot\log(\boldsymbol{A}\cdot\boldsymbol{1}) + \frac{\mu_3}{n^2}\|\boldsymbol{A}\|_F^2\,.\end{aligned} \tag{12}$$

where μ_1, μ_2, μ_3 are the hyperparameters.

D. Anomaly Score Prediction

To detect abnormal activities, we leverage graph neural networks (GNN) to enrich the activity representation from the activity graph. Then we build a fully connected neural network to predict the activity code (c_i in Equation (1)) after the GNN. Additionally, we propose a novel hybrid prediction loss function to resolve the significantly imbalanced classes in real-time ITD.

1) Graph Neural Network: We utilize a GNN model to learn node embeddings in the activity graph. The GNN takes the activity graph $\mathcal{G}$ as input and applies a message passing mechanism to capture the dependencies between the nodes. The generalized GNN can be seen as a stack of layers composed of Aggregation steps and Update steps:

$$\begin{aligned}\boldsymbol{n}_i^{(p)} &= \underset{j\in\mathcal{N}_i}{\text{Aggregator}_p}\left(\boldsymbol{x}_j^{(p)}\right)\,, \\ \boldsymbol{x}_i^{(p+1)} &= \text{Updater}_p\left(\boldsymbol{x}_i^{(p)}, \boldsymbol{n}_i^{(p)}\right)\,,\end{aligned} \tag{13}$$

where Aggregator_p and Updater_p represent the Aggregation and Update operations at the p-th layer, $\boldsymbol{x}_i^{(p)}$ represents the representation of node i at the p-th layer, $\mathcal{N}_i$ is the

neighborhood of node i, and $\boldsymbol{n}_i^{(p)} \in \mathbb{R}^d$ refers to the aggregated information from the neighbors of node i at the p-th layer.

In this paper, we explore two widely-used GNN architectures to optimize node embeddings, i.e., Graph Convolutional Network (GCN) [25] and Graph Attention Network (GAT) [26]. For GCN, the Aggregation step and Update step are formulated as follows:

$$\begin{aligned} \boldsymbol{n}_i^{(p)} &= \sum_{j \in \mathcal{N}_i} \boldsymbol{D}_{ii}^{-\frac{1}{2}} \boldsymbol{A}_{ij} \boldsymbol{D}_{jj}^{-\frac{1}{2}} \boldsymbol{x}_j^{(p)}, \\ \boldsymbol{x}_i^{(p+1)} &= \delta\left(\boldsymbol{W}_{\mathrm{GCN}}^{(p)} \boldsymbol{n}_i^{(p)}\right), \end{aligned} \tag{14}$$

where $\boldsymbol{A}_{ij}$ represents the edge weight between nodes i and j in the graph, $\boldsymbol{D}_{ii} = \sum_{j=1}^{k+1} \boldsymbol{A}_{ij}$, $\boldsymbol{W}_{\mathrm{GCN}}^{(p)} \in \mathbb{R}^{d \times d}$ is a weight matrix at the p-th layer, and $\delta(\cdot)$ represents a non-linear activation function, e.g, Rectified Linear Unit (ReLU) [27].

The Aggregation step and Update step of GAT are formulated as follows:

$$\begin{aligned} \boldsymbol{n}_i^{(p)} &= \sum_{j \in \mathcal{N}_i} \gamma_{ij} \boldsymbol{x}_j^{(p)}, \\ \gamma_{ij} &= \frac{\exp\left(\beta\left(\boldsymbol{C}^{(p)\top}\left[\boldsymbol{W}_{\mathrm{GAT}}^{(p)} \boldsymbol{x}_i^{(p)}; \boldsymbol{W}_{\mathrm{GAT}}^{(p)} \boldsymbol{x}_j^{(p)}\right]\right)\right)}{\sum_{j' \in \mathcal{N}_i} \exp\left(\beta\left(\boldsymbol{C}^{(p)\top}\left[\boldsymbol{W}_{\mathrm{GAT}}^{(p)} \boldsymbol{x}_i^{(p)}; \boldsymbol{W}_{\mathrm{GAT}}^{(p)} \boldsymbol{x}_{j'}^{(p)}\right]\right)\right)}, \\ \boldsymbol{x}_i^{(p+1)} &= \delta\left(\boldsymbol{W}_{\mathrm{GAT}}^{(p)} \boldsymbol{n}_i^{(p)}\right), \end{aligned} \tag{15}$$

where $\gamma_{ij} \in \mathbb{R}$ represents the importance of node j to node i (i.e., the attention weight), $\boldsymbol{C}^{(p)} \in \mathbb{R}^{2d}$ is a weight vector of a linear layer, $\beta(\cdot)$ stands for the Leaky Rectified Linear Unit (Leaky ReLU) activation function [28], $\boldsymbol{W}_{\mathrm{GAT}}^{(l)} \in \mathbb{R}^{d \times d}$ is a shared learnable weight matrix at the p-th layer used to provide sufficient expressive power, and $\delta(\cdot)$ refers to the ReLU activation function.

After applying GCN or GAT to the activity graph $\mathcal{G} = (\boldsymbol{A}, \boldsymbol{X})$, the ongoing activity vector $\tilde{\boldsymbol{h}}_n$ further aggregates information from the top k most related neighbors. We obtain an enhanced vector after the graph neural network, which is denoted as $\tilde{\boldsymbol{h}}'_n \in \mathbb{R}^d$. Then, we use a fully connected layer to predict the next activity of the user, which is formulated as:

$$\hat{\boldsymbol{y}}_{n+1} = \mathrm{softmax}(\boldsymbol{W}_{\mathrm{FC}} \tilde{\boldsymbol{h}}'_n + \boldsymbol{b}_{\mathrm{FC}}), \tag{16}$$

where $\boldsymbol{W}_{\mathrm{FC}} \in \mathbb{R}^{M \times d}$ is the weight matrix, $\boldsymbol{b}_{\mathrm{FC}} \in \mathbb{R}^M$ is the bias, and $\hat{\boldsymbol{y}}_{n+1} \in \mathbb{R}^M$ represents the probability distribution of the predicted activity. We compare the probability corresponding to the current activity with the detection threshold to determine if it is anomalous.

2) Hybrid Prediction Loss: In a real enterprise system, abnormal activities are very limited and difficult to identify, leading to significantly imbalanced sample numbers between normal activities and abnormal ones [29]. Therefore, alleviating the impact of data imbalance is crucial for ITD.

In this paper, we alleviate the impact of data imbalance by leveraging the supervision of the limited abnormal activity labels. Given the substantial class imbalance, it is inappropriate to treat the ITD problem as a binary classification task due to the inadequacy of abnormal labels. The LAN paradigm leverages historical activities to predict the next one, deeming an activity abnormal if its occurrence probability is exceedingly low. This approach seamlessly integrates self-supervised signals derived from normal labels, fostering natural learning of normal activity patterns. However, training the model in this self-supervised manner introduces noise if the abnormal activities are treated as normal ones. Therefore, to improve the performance of ITD, we propose a novel hybrid prediction loss that combines both supervised and self-supervised learning, imparting awareness to the self-supervised model regarding labels associated with abnormal activities.

Specifically, given an activity sequence $S = \{a_1, a_2, \ldots, a_n\}$ and the corresponding anomaly labels of the activities $\boldsymbol{q} \in \mathbb{R}^n$, $\boldsymbol{q}_i \in \{0, 1\}$, the conventional approach for optimizing the next activity prediction is to use one-hot encoding of activities and the cross-entropy loss, which is defined by $\mathcal{L}_{\mathrm{CE}} = -\frac{1}{n-1} \sum_{i=2}^{n} \log(\hat{\boldsymbol{Y}}_{i,c_i})$, where $\hat{\boldsymbol{Y}} \in \mathbb{R}^{n \times M}$ represents the probability distribution of the model's predictions for the occurrence of behaviors at each position. $\hat{\boldsymbol{Y}}_i \in \mathbb{R}^M$ corresponds to the i-th position, and $\hat{\boldsymbol{Y}}_{i,c_i}$ denotes the predicted probability of the occurrence of the ground truth activity a_i (the c_i-th item in $\hat{\boldsymbol{Y}}_i$). Recall that c_i is the corresponding integer code of a_i.

However, this cross-entropy loss only optimizes the positive feedback of predicting the next activity. To explicitly utilize the negative feedback provided by anomaly labels, we take the labels of abnormal activities (i.e., $q_i = 1$) into account. Specifically, for abnormal user activities, we do not want the model to learn similar activity patterns. Thus, we prefer to suppress the occurrence probability of such anomalous activities compared to other ones. We achieve this goal indirectly by increasing the probabilities of other activities under the constraint of softmax, thus reducing the occurrence probability of anomalous activities.

We construct a new soft label distribution, which can be calculated by:

$$\boldsymbol{Y}' = \boldsymbol{Y} \odot (1 - \boldsymbol{\Omega}) + \frac{1}{M-1} \boldsymbol{\Omega} \odot (1 - \boldsymbol{Y} \odot \boldsymbol{\Omega}), \tag{17}$$

where $\boldsymbol{\Omega} = \boldsymbol{q} \otimes \mathbf{1}_M^\top \in \mathbb{R}^{n \times M}$ is a masking matrix, $\otimes$ represents the Kronecker product operation, $\mathbf{1}_M \in \mathbb{R}^M$ denotes a M-dimensional vector filled with ones. $\boldsymbol{Y} \in \mathbb{R}^{n \times M}$ represents the one-hot labels of activity sequence S. When $\boldsymbol{q}_i = 1$, the resulting label distribution from Equation (17) can be simplified as:

$$\boldsymbol{Y}'_{ij} = \begin{cases} 0, & j = c_i, \\ \dfrac{1}{M-1}, & j \neq c_i. \end{cases} \tag{18}$$

Furthermore, to ensure that the learning of normal activities is not affected, we keep the probability distribution of normal activities unchanged, i.e., $\boldsymbol{Y}'_{ij} = \boldsymbol{Y}_{ij}$, when $\boldsymbol{q}_i = 0$.

Furthermore, the importance of abnormal activity samples is different from that of normal samples. We assign a weight r to the abnormal samples and construct a sample weight vector $\boldsymbol{w}^s$, i.e.,

$$\boldsymbol{w}_i^s = 1 + (r - 1) \cdot \boldsymbol{q}_i, \boldsymbol{q}_i \in \{0, 1\}. \tag{19}$$

We refer to this operation as the weighting negative feedback operation for the next activity prediction task.

Finally, the hybrid prediction loss is defined as a weighted soft cross-entropy, i.e.,

$$\mathcal{L}_{\text{Hybrid}} = -\frac{1}{n-1}\sum_{i=2}^{n} \boldsymbol{w}_i^s \sum_{j=1}^{M} \boldsymbol{Y}'_{ij} \log(\hat{\boldsymbol{Y}}_{ij}) . \tag{20}$$

We optimize the model parameters by minimizing the overall loss function $\mathcal{L} = \mathcal{L}_{\text{Hybrid}} + \mathcal{L}_{\text{Reg}}$.

3) Model Update Strategy: The mode of running LAN is offline training and online inference, so that the training does not affect the inference. We suggest updating the model when collecting a set of misclassified samples. The threshold of the set size should be determined according to the system's significance and available resources. Note that LAN can not only learn from normal activities, but also learn from abnormal activities. In addition, the training efficiency can be improved by incremental training. Given new activity sequences, LAN can update the model weights incrementally without retraining from scratch. By incremental learning, the cost can be significantly reduced.

E. Post-Hoc ITD

In real-time ITD, we utilize the user's previous activity sequence to predict the next activity. However, in post-hoc ITD, we need to make some changes. Instead of predicting the next activity, we treat it as an "activity cloze" task. For a certain time step t, we mask the user's activity at that time step, resulting in a new activity sequence $S' = \{a_1, \ldots, a_{t-1}, \langle MASK \rangle, a_{t+1}, \ldots, a_n\}$. We feed this new S' into a sequence encoder with bidirectional contextual capabilities while keeping the rest of the LAN unchanged. Then, we predict the masked activity to complete the detection.

IV. Experiments

A. Experimental Setup

1) Dataset: As previous studies [4], [10], [11], [30], [31], [32], we evaluate the performance of LAN on two publicly available datasets, i.e., CERT r4.2 and CERT r5.2 [33]. They are synthetic ITD datasets that provide normal and malicious user activities. With different scales, CERT r4.2 and CERT 5.2 both contain user activity data in a company from January 2010 to June 2011. Table I summarizes the statistics of the datasets. Specifically, CERT r4.2 contains 1,000 employees with 32,770,222 user activities, among which 7,316 activities of 70 employees were manually injected as abnormal activities by domain experts. Similarly, CERT r5.2 contains 2,000 employees with 79,856,664 activities, and 10,306 activities of 99 employees were manually injected as abnormal activities. It can be seen that normal employees and normal activities occupy the vast majority of the whole dataset. To clearly show the data imbalance problem, we calculate the imbalance ratio (IR) by N_{maj}/N_{min}, where N_{maj} and N_{min} are the sample sizes of the majority class and the minority class, respectively [34]. The larger the IR, the greater the imbalance extent of the dataset. From Table I it can be seen that the IR is 13 for employees in CERT 4.2. When it comes to the activities, the IR becomes 4,478 in the same dataset, which indicates the difficulty in classifying normal and abnormal activities. For CERT r5.2, the data imbalance is even worse, which poses great challenges for activity-level ITD.

TABLE I
Statistics of the Datasets

Dataset	CERT r4.2	CERT r5.2
# Normal Employees	930	1,901
# Abnormal Employees	70	99
Imbalance Ratio	**13**	**19**
# Normal Activities	32,762,906	79,846,358
# Abnormal Activities	7,316	10,306
Imbalance Ratio	**4,478**	**7,748**
# Normal Activities after Preprocessing	7,664,484	27,254,280
# Abnormal Activities after Preprocessing	7,316	10,306
Imbalance Ratio	**1,048**	**2,645**

2) Data Preprocessing: Since there are multiple sources for user activity logs (e.g., logs of logon or website visit), the first step of data preprocessing is to aggregate the data from all sources so that the activities of the same user can be aggregated according to their time stamps. Then, as previous studies [7], [8], we split the activities of each user into sessions, where each session contains a set of activities in chronological sequence between a user's logon and logoff. Since the goal is to detect insider threats at the activity level, each activity is regarded as a sample for training the model. Specifically, for real-time ITD, each activity is represented by the activity itself along with its previous activities in the same session, so that an abnormal activity can be immediately detected once it occurs. As for post-hoc ITD, we put each activity into its corresponding session, and an activity can only be detected after the session ends to incorporate the bidirectional context of the entire session. Moreover, we filter out repetitive HTTP operations within the same hour following [16]. The statistics for data after preprocessing can also be seen in Table I. In order to convert user activities into machine-readable data, we employ the method described in Section III-B.1 to transform each activity into a unique numerical token. These numerical token sequences serve as input for all sequence models, with the goal of predicting anomalous tokens within the token sequence. Finally, we use the user activity data in 2010 for training and validation, and use the data from January 2011 to June 2011 for evaluation.

Note that the aforementioned data preprocessing was the same for all sequence models during inference. In the training phase, the preprocessing of LAN also included an additional step for constructing the activity vector pool as described in Section III-C.1. It took 39 additional seconds and 3.8 additional minutes for CERT r4.2 and CERT r5.2, respectively. LAN constructed the activity vector pool only once regardless of how many samples it infers unless the pool is expanded.

3) Baselines: To evaluate the performance of LAN on activity-level ITD, we compare LAN with competitive baselines that have the ability to detect insider threats at the activity level. In this paper, we configure LAN by employing LSTM as the sequence encoder and GCN as the graph neural network (GNN). For real-time ITD, we compare LAN with 9 state-of-the-art methods, i.e., RNN [35], GRU [36], DeepLog [37] (LSTM [17]), Transformer [19], RWKV [38], TIRESIAS [39], DIEN [40], BST [41], and FMLP [42]. All these baselines are sequence-based. Specifically, DeepLog

utilizes LSTM to predict whether each log entry is anomalous. RNN, GRU, LSTM, and Transformer are four widely used architectures for sequence modeling. RWKV is a recently proposed sequence modeling architecture that combines the advantages of efficient parallel training in Transformer and efficient sequential inference in RNN. TIRESIAS is an LSTM variant. FMLP, a sequential recommendation model, filters out noise from user historical activity data to reduce model overfitting. These sequence-based models are adapted for real-time ITD and trained in the auto-regressive manner, which predicts the next activity based on historical sequences. Additionally, we include DIEN and BST for real-time ITD. They are commonly used methods for user behavior modeling, which have been widely used for click-through rate prediction in recommendation systems.

For post-hoc ITD, we compare LAN with 8 state-of-the-art methods, i.e., RNN, GRU, DeepLog (LSTM), Transformer, FMLP, ITDBERT [16], OC4Seq [43], and log2vec [11]. Unlike real-time ITD, for the post-hoc ITD task we configure RNN, GRU, and DeepLog as bidirectional models to capture both preceding and succeeding contexts of each activity simultaneously. RNN, GRU, DeepLog, Transformer, and FMLP are trained using an auto-encoder approach, where a masked activity is predicted based on its neighboring activities, either preceding or following it. ITDBERT uses an attention-based LSTM for session-level prediction, using attention weights of each activity as the anomaly score. OC4Seq regards log anomaly detection as a one-class classification problem. It represents user activities using RNN and trains the model only on normal activities, searching for an optimal hypersphere in the latent space to enclose normal activities. During inference, it predicts whether an activity falls within the hypersphere to determine if it is anomalous. Log2vec employs graph-based methods to detect malicious activities. It designs heuristic rules to manually extract edges to represent relationships between activities. We reproduce log2vec following the instructions in [11]. Note that only log2vec is a graph-based method, the others are all sequence-based.

4) Evaluation Metrics: Like previous studies [4], [44], [45], [46], we utilize the Receiver Operating Characteristic (ROC) curves to visualize the detection rate (DR) and the false positive rate (FPR) of each model, where $DR = TP/(TP + FN)$ and $FPR = FP/(FP + FN)$. TP, FN, FP, and TN represent the number of true positives, false negatives, false positives, and true negatives, respectively. We compute the Area Under Curve (AUC) to evaluate the overall performance of all methods. Note that for practical use, it is necessary to fix a decision threshold for anomaly detection, i.e., activities with lower likelihood scores than the decision threshold are regarded as abnormalities. To achieve a trade-off between DR and FPR, a common practice is to set the decision threshold using the Youden index [47], which can be calculated by $DR - FPR$. We use the decision threshold with the maximum Youden index and report the DR and FPR according to the threshold. In addition, following previous studies [6], [45], [48], we also vary the investigation budget, which is the amount of suspicious activities that security analysts can inspect, and set the decision threshold accordingly. Specifically, we vary the investigation budget by 5%, 10%, and 15% of the number of activities to be tested, and report the corresponding detection rates DR@5%, DR@10%, and DR@15%, respectively.

5) Implementation Details: To conduct a fair comparison, we set the same hidden layer size (i.e., 128) for all methods, and all models were trained on the same training set with the early stopping strategy for 10 epochs. We select the best learning rate and use the AdamW optimizer [49] with a weight decay of 0.01 when training each model. For the hyper-parameters in LAN, we use 8 attention heads for multi-head attentive pooling, and the sizes of the activity vector pool are 1,025,920 and 5,359,987 for CERT r4.2 and CERT r5.2, respectively. To efficiently retrieve the top k vectors (k is set to 15 after parameter analysis) most relevant to the current activity vector, we exploit Faiss [50], a high-performance open-source library for searching approximate nearest neighbors in dense vector spaces, and utilize the Hierarchical Navigable Small Word (HNSW) algorithm [51] to build the index, which has a logarithmic retrieval time complexity. During graph structure learning, we utilize 4 attention heads in Equation (7) and set the hard threshold ϵ to suppress noise in Equation (8) to 0.5 based on experimental experience. As previous studies on graph structure learning [22], the hyper-parameters μ_1, μ_2, and μ_3 for the graph regularization loss are set to 0.2, 0.1 and 0.1, respectively. For the weight r in the hybrid prediction loss, we set it to the imbalance ratio of the dataset according to parameter analysis. Since the weight r is very large, to ensure numerical stability, we replicate the abnormal activity samples r times to achieve the same effect as the weighted loss. The experiments were conducted on a server with 2 Intel Xeon Gold 6226R CPUs running at 2.90GHz, 256GB of RAM, and one A6000 GPU with 48GB memory. The toolkit used for the experiments included Python 3.8, PyTorch 1.13, PyTorch Geometric 2.3, and Faiss 1.7.

B. Experimental Results

Figure 3 depicts the ROC curves of all methods for real-time and post-hoc ITD. Overall, it can be seen that the ROC curves of LAN are much closer to the top-left corner than the curves of all the baselines in all experiments. The results indicate that LAN achieves better overall performance than all the baselines for both real-time and post-hoc ITD.

Table II shows the detection results of LAN and 9 state-of-the-art baselines for real-time ITD. We use AUC, DR, FPR, DR@5%, DR@10%, and DR@15% to evaluate the performance of all methods. Among these metrics, the higher AUC, the better overall performance. For the detection metrics DR, DR@5%, DR@10%, and DR@15%, a higher detection rate indicates that the model can identify more anomalies. Finally, the lower FPR, the fewer false positives that waste the investigation budget.

It can be seen that LAN is superior to all the baselines with regard to all six metrics on two datasets, with the highest AUC and detection rates and the lowest FPRs among all methods. Specifically, LAN surpasses the baselines by at least 0.0843 and 0.0635 in AUC on CERT r4.2 and r5.2, respectively. For the FPR scores, it can be seen that the FPRs of LAN are 0.1411 and 0.0867 on CERT r4.2 and r5.2, respectively, lower than all the baselines by at least

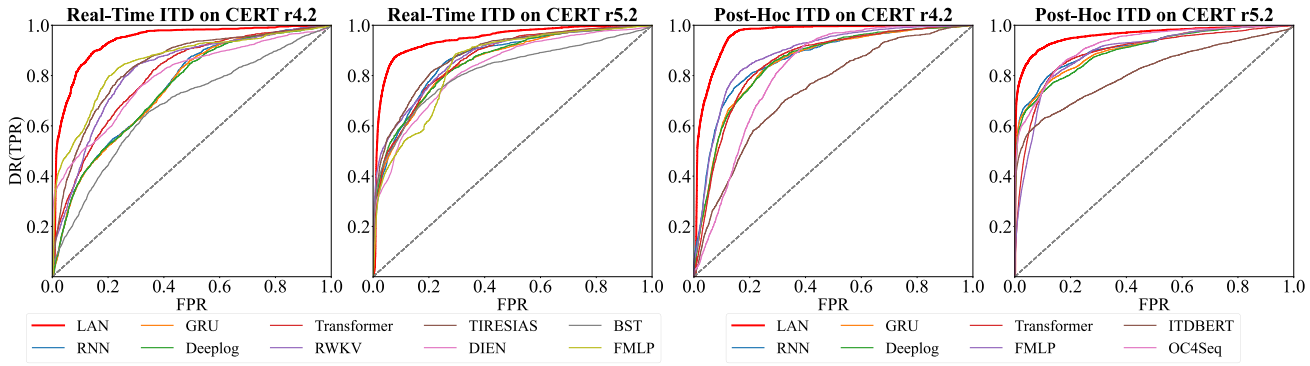


Fig. 3. ROC curves for real-time ITD and post-hoc ITD on two datasets.

TABLE II

PERFORMANCE COMPARISON OF LAN WITH NINE BASELINES FOR REAL-TIME ITD. THE BEST AND SECOND-BEST RESULTS ARE BOLDFACED AND UNDERLINED, RESPECTIVELY. "↑" INDICATES THE HIGHER THE BETTER, AND ""↓"" INDICATES THE LOWER THE BETTER

Model	CERT r4.2						CERT r5.2					
	AUC ↑	DR ↑	FPR ↓	DR@5%↑	DR@10%↑	DR@15%↑	AUC ↑	DR ↑	FPR ↓	DR@5%↑	DR@10%↑	DR@15%↑
RNN [35]	0.7521	0.6934	0.3622	0.2299	0.3821	0.4625	0.8641	0.8286	0.2361	0.4548	0.5928	0.6910
GRU [36]	0.7486	0.7119	0.3804	0.2391	0.3815	0.4614	0.8504	0.7911	0.2395	0.4637	0.5704	0.6499
DeepLog [37]	0.7469	0.7152	0.3767	0.2310	0.3842	0.4620	0.8549	0.7767	0.2336	0.4970	0.5954	0.6648
Transformer [19]	0.7981	0.7195	0.2799	0.2918	0.4201	0.5321	0.8628	0.7621	0.1985	0.4858	0.5745	0.6694
RWKV [38]	0.8165	0.7923	0.2576	0.2630	0.4348	0.5886	0.8727	0.8020	0.2345	0.5380	0.6224	0.6887
TIRESIAS [39]	0.8377	0.7820	0.2338	0.3761	0.5277	0.6484	0.8804	0.8129	0.2073	0.5463	0.6373	0.7297
FMLP [42]	0.8526	0.7983	0.2027	0.4783	0.5647	0.6837	0.8435	0.8757	0.2889	0.4278	0.5171	0.5659
DIEN [40]	0.7894	0.7461	0.3072	0.4147	0.4875	0.5342	0.8268	0.7724	0.2690	0.3811	0.5455	0.6175
BST [41]	0.6777	0.6554	0.3451	0.1625	0.2614	0.3647	0.8162	0.7417	0.2301	0.4772	0.5650	0.6548
LAN (Ours)	**0.9369**	**0.8875**	**0.1411**	**0.6832**	**0.8337**	**0.8951**	**0.9439**	**0.8814**	**0.0867**	**0.8089**	**0.8881**	**0.9076**
Abs. Improv.	0.0843	0.0892	0.0616	0.2049	0.2690	0.2114	0.0635	0.0057	0.1118	0.2626	0.2508	0.1779
Rel. Improv.(%)	9.89%	11.17%	30.39%	42.84%	47.64%	30.92%	7.21%	0.65%	56.32%	48.07%	39.35%	24.38%

0.0616 and 0.1118 on two datasets. The results indicate that LAN significantly reduces the number of false positives, which has great value when the investigation budget is finite. We can also observe that the recent auto-regressive models (e.g., Transformer, RWKV, TIRESIAS, and FMLP) perform better than the click-through prediction models (i.e., DIEN and BST). However, the auto-regressive models can only learn the normal activity patterns and can not learn from abnormal signals, leading to a high false positive rate. Moreover, comparing LAN with DeepLog, it can also be observed that although both of them utilize LSTM as the backbone, LAN achieves better performance by automatically learning the relationships between different activity sequences.

In addition to real-time ITD, we also apply LAN, with slight modifications, for post-hoc ITD. To evaluate the effectiveness of LAN for post-hoc ITD, we compare LAN with 8 state-of-the-art baselines (i.e., RNN [35], GRU [36], DeepLog [37], Transformer [19], FMLP [42], ITDBERT [16], OC4Seq [43], and log2vec [11]. Table III displays the results of all methods for post-hoc ITD on CERT r4.2 and r5.2. Note that log2vec constructs a graph for each abnormal user and performs clustering for the log entries of each user, assuming that smaller clusters tend to be suspicious. For this reason, the detection of abnormal activities relies on how to determine the threshold that represents the size of clusters. As a result, log2vec can not be applied when the investigation budget is fixed (e.g., 5% of all activities). Hence, we did not compute the DR@5%, DR@10%, and DR@15% for log2vec.

From Table III it can be seen that LAN achieves the best performance in terms of all six metrics on two datasets, with 0.9607 AUC, 0.9478 DR, 0.1222 FPR in CERT r4.2 and 0.9605 AUC, 0.9024 DR, 0.0865 FPR in CERT r5.2. The results demonstrate the effectiveness of LAN on post-hoc ITD, which can detect more anomalies with fewer false positives. Especially when the investigation budget is 15% of the activities under test, the detection rate (i.e., DR@15%) reaches 0.9739 and 0.9346 in CERT r4.2 and r5.2, respectively. Moreover, we can also observe that LAN surpasses all baselines by at least 0.0449 in FPR, which indicates a significant improvement in reducing the waste of the investigation budget. The auto-regressive models (i.e., RNN, GRU, DeepLog, Transformer, and FMLP) outperform other baselines, suggesting that the unsupervised auto-encoder training approach is also effective for post-hoc insider threat detection. The performance of the graph-based model (i.e., log2vec) is relatively poor compared with the other methods. A possible reason could be that the graph construction within log2vec is heuristic.

Combining the results in Table II and Table III, it can be concluded that LAN can be applied for both real-time ITD and post-hoc ITD, superior to all the state-of-the-art baselines with regards to all six metrics, especially in reducing false positives which might waste the manual investigation cost.

C. Efficiency Analysis

Table IV shows the training and inference costs of LAN on CERT r4.2 and CERT r5.2 for real-time and post-hoc ITD.

TABLE III

PERFORMANCE COMPARISON OF LAN WITH EIGHT BASELINES FOR POST-HOC ITD. THE BEST AND SECOND-BEST RESULTS ARE BOLDFACED AND UNDERLINED, RESPECTIVELY. "↑" INDICATES THE HIGHER THE BETTER, AND ""↓"" INDICATES THE LOWER THE BETTER

Model	CERT r4.2						CERT r5.2					
	AUC ↑	DR ↑	FPR ↓	DR@5%↑	DR@10%	DR@15%↑	AUC ↑	DR ↑	FPR ↓	DR@5%↑	DR@10%	DR@15%↑
RNN [35]	0.8652	0.8032	0.1996	0.4375	0.6707	0.7549	0.9108	0.8131	0.1359	0.6881	0.7699	0.8230
GRU [36]	0.8514	0.8103	0.2378	0.3364	0.5962	0.7001	0.9040	0.7879	0.1367	0.6780	0.7458	0.7957
DeepLog [37]	0.8531	0.7891	0.2259	0.3310	0.5908	0.6897	0.8985	0.7730	0.1385	0.6729	0.7329	0.7779
Transformer [19]	0.8533	0.8005	0.2112	0.3109	0.5451	0.6951	0.8929	0.8358	0.1586	0.5684	0.7357	0.8247
FMLP [42]	0.8837	0.8190	0.1671	0.4266	0.6772	0.7957	0.8920	0.8169	0.1614	0.4878	0.7412	0.8066
ITDBERT [16]	0.7413	0.6911	0.3153	0.2005	0.3272	0.4383	0.8139	0.6853	0.1996	0.5724	0.6243	0.6518
OC4Seq [43]	0.8113	0.8080	0.2940	0.1466	0.3019	0.4712	0.9202	0.8503	0.1727	0.6414	0.7383	0.8245
log2vec [11]	0.6563	0.6793	0.3022	-	-	-	0.6178	0.6441	0.3388	-	-	-
LAN(Ours)	**0.9607**	**0.9478**	**0.1222**	**0.7429**	**0.8929**	**0.9739**	**0.9605**	**0.9024**	**0.0865**	**0.8591**	**0.9088**	**0.9346**
Abs. Improv.	0.0770	0.1288	0.0449	0.3054	0.2157	0.1782	0.0403	0.0521	0.0494	0.1710	0.1389	0.1099
Rel. Improv.(%)	8.71%	15.73%	26.87%	69.81%	31.85%	22.40%	4.38%	6.13%	36.35%	24.85%	18.04%	13.33%

TABLE IV

INFERENCE AND TRAINING TIME OF LAN

Stage	Item	Real-Time ITD	Post-Hoc ITD
Inference	Single Activity	0.30ms	0.45ms
	CERT r4.2	11min	14min
	CERT r5.2	39min	54min
	#Activities/second	3,339	2,214
	#Activities/day	288,540,302	191,316,567
Training	Single Activity	0.52ms	1.30ms
	CERT r4.2	24min	38min
	CERT r5.2	2h9min	4h45min
	#Activities/second	1,911	770
	#Activities/day	165,113,825	66,556,538

TABLE V

THE NUMBERS OF CONCURRENT ACTIVITIES

	CERT r4.2			CERT r5.2		
	mean	min	max	mean	min	max
#Activities/sec	1.4	1	63	2.3	1	149
#Activities/min	17	1	184	51.7	1	415
#Activities/hour	683	1	2,599	2,241	1	9,261
#Activities/day	15,312	8	23,559	52,736	9	84,527

Note that the reported time does not include the time for preprocessing. It can be seen that it only took 0.30ms and 0.52ms to predict each activity in real-time and post-hoc ITD, respectively. Table V shows the number of concurrent activities in two datasets. We can infer that even when the number of concurrent activities within a second reached its maximum, it only took 18.9ms and 44.7ms to predict all activities within the second for CERT r4.2 and r5.2, respectively, which is acceptable for real-time ITD. Moreover, it can also be observed that LAN can infer 288 million and 191 million activities per day for real-time ITD and post-hoc ITD, respectively, which is much more than the maximum number of user activities within a day in two scenarios.

Figure 4 shows the histogram of inference time of real-time ITD across various context lengths, where the context length means the number of historical activities used to predict the next activity. It can be seen that as the context length increases, it requires more time to infer an activity. However, even when the context length is 80, the inference time does not exceed 1.6ms. Note that 80 is the largest context length within a session in CERT r5.2.

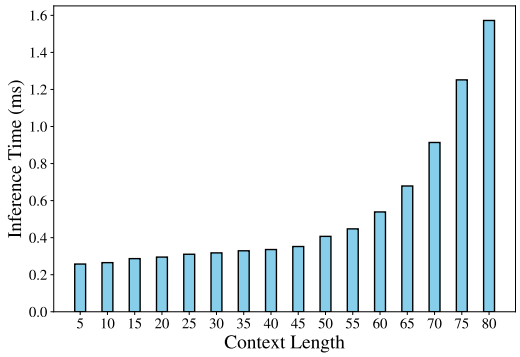


Fig. 4. Inference time for different context lengths.

D. Ablation Studies

In this section, we conduct ablation studies to evaluate the effectiveness of each module in LAN. As the comparative experiments, we configure LAN by using LSTM as the sequence encoder and GCN as the graph neural network in LAN. The ablation study was conducted on the CERT r4.2 dataset. We implement LAN with the different settings as follows.

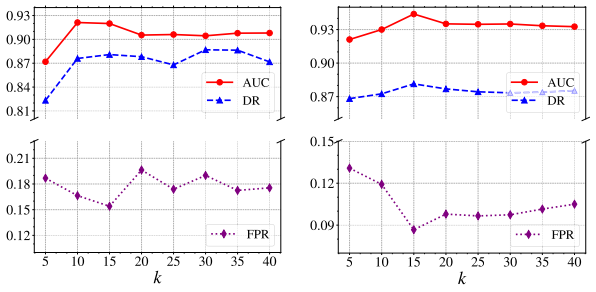
- **LAN** represents the complete version of the model proposed in this paper.
- **LAN-P**. We further remove the multi-head attentive pooling operation mentioned in Section III-B.2 from the complete LAN, replacing it with the average pooling operation.
- **LAN-P/W**. On the basis of LAN-P, we replace the weight of negative feedback mentioned in Equation (19) with 1, setting the same weights for normal and abnormal activities.
- **LAN-P/W/S**. On the basis of LAN-P/W, we further remove the weighted cosine similarity metric function in Equation (7), and replace it with the cosine similarity.
- **LAN-P/W/S/R**. On the basis of LAN-P/W/S, we further remove the graph regularization loss in Section III-C.3.
- **LAN-P/W/S/R/H**. On the basis of LAN-P/W/S/R, we replace the hybrid prediction loss mentioned in Section III-D.2 with the standard cross entropy loss.
- **LAN-P/W/S/R/H/G**. On the basis of LAN-P/W/S/R/H, we remove the entire graph structure. At this point, the model is actually a single sequence encoder such as LSTM.

Table VI shows the results of the ablation study. From the experimental results, it can be observed that the introduction

TABLE VI

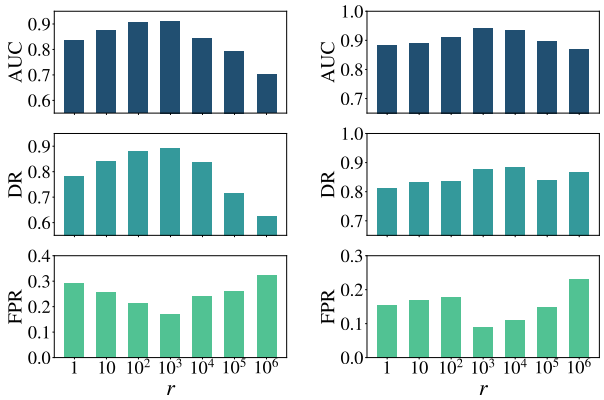
RESULTS OF THE ABLATION STUDY. G: INTRODUCING GRAPH STRUCTURE, H: HYBRID PREDICTION LOSS, R: GRAPH REGULARIZATION,S: WEIGHED COSINE SIMILARITY, W: WEIGHTING NEGATIVE FEEDBACK, P: MULTI-HEAD ATTENTIVE POOLING

Model	Settings						Real-Time Detection			Post-Hoc Detection		
	G	H	R	S	W	P	AUC↑	DR↑	FPR↓	AUC↑	DR↑	FPR↓
LAN	✓	✓	✓	✓	✓	✓	**0.9369**	**0.8869**	**0.1411**	**0.9607**	**0.9478**	0.1222
LAN-P	✓	✓	✓	✓	✓	✗	0.9253	0.8782	0.1535	0.9531	0.9059	**0.1151**
LAN-P/W	✓	✓	✓	✓	✗	✗	0.8340	0.7809	0.2945	0.8898	0.8375	0.1721
LAN-P/W/S	✓	✓	✓	✗	✗	✗	0.8325	0.8010	0.2979	0.8801	0.8298	0.1771
LAN-P/W/S/R	✓	✓	✗	✗	✗	✗	0.8314	0.7983	0.3001	0.8653	0.8282	0.1939
LAN-P/W/S/R/H	✓	✗	✗	✗	✗	✗	0.8144	0.7750	0.2901	0.8631	0.8119	0.2159
LAN-P/W/S/R/H/G	✗	✗	✗	✗	✗	✗	0.7469	0.7152	0.3767	0.8531	0.7891	0.2259



(a) CERT r4.2 (b) CERT r5.2

Fig. 5. Performance of LAN with different numbers of candidate neighbors k obtained through retrieval.



(a) CERT r4.2 (b) CERT r5.2

Fig. 6. Performance of LAN with different weights of negative feedback r in the hybrid prediction loss.

of each component leads to varying degrees of performance improvement in the model. From the results of LAN-P/W/S/R/H/G, LAN-P/W/S/R, and LAN-P/W/S, it can be observed that the removal of these modules leads to a varying degree of performance decrease. This indicates that using our framework is crucial because it allows the model to break free from the limitations of sequence models and automatically discover relationships between activities located in different sequences, resulting in a significant improvement in model performance. From the results of LAN-P/W/S/R/H and LAN-P/W, it can be observed that the removal of these modules also leads to a significant decrease in performance. This indicates the use of hybrid prediction loss and weighting negative feedback is also very important. This is because they enable our framework to make full use of limited label information and enhance the discrimination ability for abnormal activities. Ultimately, in the real-time ITD task, our model improves the initial model's AUC from 0.7469 to 0.9369, increases DR from 0.7152 to 0.8869, and reduces FPR from 0.3767 to 0.1411. In the post-hoc ITD task, our model improves the initial model's AUC from 0.8531 to 0.9607, increases DR from 0.7891 to 0.9478, and reduces FPR from 0.225 to 0.122. These improvements are crucial for activity-level ITD.

E. Parameter Analysis

In this section, we analyze the influences of two key hyper-parameters of LAN, i.e., the number of candidate neighbors k in the Activity Graph Learning module and the weight of negative feedback r in the hybrid prediction loss. Specifically, we vary the number of candidate neighbors k by 5, 10, 15, 20, 25, 30, and 40. Figure 5 illustrates the changes of AUC, DR, and FPR as k increases.

On both datasets, it can be observed that as k increases, AUC first increases rapidly and reaches the peak when $k = 10$, and then decreases slowly. Similarly, when we increase k, DR first increases and then fluctuates slightly. In contrast to AUC and DR, as k increases, FPR first drops rapidly, and then it increases with fluctuations.

In summary, in terms of AUC, LAN achieves the best performance when k is set to 10. In terms of FPR, LAN achieves the best performance when k is 15 and the FPR is the lowest. Moreover, it can also be observed that in many cases, when we increase k, both DR and FPR increase at the same time. A possible reason is that by considering more neighbors, the aggregation activities are more diverse, which makes the model detect more abnormal activities, but also introduces more noise. Overall, the changes in performance are not significant, which might be due to the noise-resistant capability provided by the design of LAN such as the graph regularization constraints and weighted cosine metric.

For the weight of negative feedback r in the hybrid prediction loss, we vary it by 1, 10, 10^2, 10^3, 10^4, 10^5, and 10^6. Figure 6 shows the results of AUC, DR, and FPR with different weights r. Overall, it can be seen that the influence of the weight r exhibits similar trends in the two datasets. As the negative sample weight r increases, AUC and DR first increase and reach the peak. After the peak, AUC and DR decrease when r increases. In contrast, when we increase r, FPR decreases at first, and achieves the lowest value when AUC and DR reach the peak. After that, FPR increases slowly. Specifically, on the CERT r4.2 dataset, LAN achieves the best

TABLE VII
PERFORMANCE OF DIFFERENT COMBINATIONS OF MODELS

	Dataset	Architecture	AUC ↑	DR ↑	FPR ↓
Real-Time	r4.2	GRU+GCN	0.9313	0.8983	0.1579
		GRU+GAT	0.9298	0.8826	0.1499
		LSTM+GCN	**0.9369**	0.8875	**0.1411**
		LSTM+GAT	0.9309	**0.9141**	0.1447
		Transformer+GCN	0.9086	0.8217	0.1765
		Transformer+GAT	0.9060	0.8385	0.1681
	r5.2	GRU+GCN	0.9420	0.8599	0.0923
		GRU+GAT	0.9434	0.8794	0.1006
		LSTM+GCN	**0.9439**	0.8814	0.0867
		LSTM+GAT	0.9426	**0.8829**	0.1000
		Transformer+GCN	0.9329	0.8700	0.1098
		Transformer+GAT	0.9349	0.8513	0.0831
Post-Hoc	r4.2	GRU+GCN	**0.9610**	0.9429	0.1262
		GRU+GAT	0.9590	0.9423	0.1206
		LSTM+GCN	0.9607	**0.9478**	0.1222
		LSTM+GAT	0.9578	0.9347	**0.1199**
		Transformer+GCN	0.9306	0.8934	0.1407
		Transformer+GAT	0.9391	0.9092	0.1405
	r5.2	GRU+GCN	0.9602	**0.9035**	0.0816
		GRU+GAT	0.9575	0.8952	0.0779
		LSTM+GCN	0.9605	0.9024	0.0865
		LSTM+GAT	**0.9607**	0.8984	**0.0767**
		Transformer+GCN	0.9410	0.8791	0.1013
		Transformer+GAT	0.9446	0.8794	0.0966

performance when r is set to 10^3, which is very close to the imbalance ratio of CERT r4.2 (i.e., 1,048). A possible explanation is that the model with r set by the imbalance ratio can pay appropriate attention to the negative feedback, which improves the detection ability for abnormal activities, and prevents the model from yielding to the majority of samples, which are normal activities. On the CERT r5.2 dataset, we obtain a similar observation. Specifically, when r is 10^3, AUC is the highest and FPR is the lowest among all values. When r is 10^4, DR achieves the highest value. The observation is consistent with the imbalance ratio of CERT r5.2 (2,645), which is between 10^3 and 10^4.

F. Compatibility Analysis

Since the compatibility of the proposed framework is significant for its practical use, we realize different implementations of LAN to verify the proposed framework for both real-time ITD and post-hoc ITD. Specifically, in the Activity Sequence Modeling module, we implement LAN with three representative sequence encoders, i.e., LSTM, GRU, and Transformer. For post-hoc detection, we configure LSTM and GRU as bidirectional to utilize context before and after a timestamp simultaneously. In the Anomaly Score Prediction module, we implement LAN with two representative graph neural networks, i.e., GCN and GAT. To conduct a fair comparison, the rest of the framework is the same for all models, and we test the implementations on two datasets for both real-time ITD and post-hoc ITD.

Table VII presents the experimental results of different implementations of LAN. First, it can be observed that all combinations perform well in both real-time ITD and post-hoc ITD on two datasets, surpassing the performance of all the baselines as shown in Table II and Table III. The results demonstrate the effectiveness and compatibility of the proposed framework, reflecting that LAN can incorporate the temporal dependencies and learn the relationships between user activities across different sequences simultaneously.

Among all the implementations of the proposed framework, it can be observed from Table VII that the use of LSTM and GCN leads to slightly better performance than other models. Regarding the GNN module, using GAT in LAN often leads to a slight decrease in AUC performance. However, in many cases, using GAT exhibits superior DR. The potential reason for this is that GAT may attempt to focus more on specific neighbors, breaking the limitations of graph structure learning. Nonetheless, breaking these limitations is not always beneficial, as it may introduce some noise. However, at times, it can also yield unexpected effects. Another interesting observation is that, while Table II and Table III show that the more complex and global sequence encoder, Transformer, generally outperforms the other three recurrent sequence encoders (RNN, GRU, and LSTM) when used independently, especially in terms of FPR. However, when used as a sequence encoder as part of our LAN framework, although its performance is also improved compared to using Transformer independently, and surpasses all other state-of-the-art methods, it is not as good as setting the sequence encoder of LAN directly to LSTM. A possible reason is that the representations generated by Transformer exhibit significant anisotropy in the vector space [52], [53], occupying a narrow cone-like structure. As a result, each output representation tends to be similar, which hinders the learning of graph structure.

G. Visualization

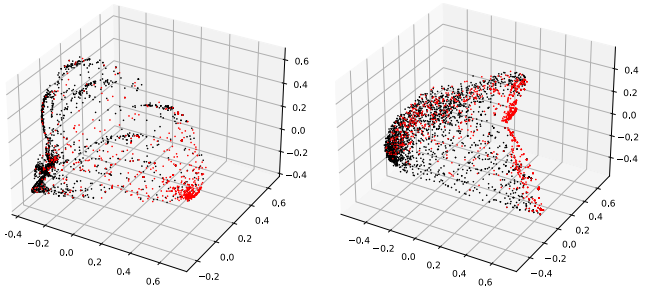
To further illustrate the capability of LAN to distinguish between normal and abnormal activities, we exploit Principal Component Analysis (PCA) to project the user activity vectors learned by LAN into a three-dimensional space. We compare LAN with DeepLog, since both of them use LSTM as a basic building block. Figure 7 shows the visualization of the activity vectors learned by two algorithms in the real-time ITD and post-hoc ITD tasks. The black and red points in Figure 7 represent the vectors of normal activities and abnormal activities, respectively. It can be observed that the points of the same color are more concentrated in the visualization of the activity vectors learned by LAN than those learned by Deeplog, which indicates that LAN distinguishes normal and abnormal activities better than Deeplog does. The visualization shows the superiority of LAN compared with Deeplog by the proposed graph-based networks and the hybrid prediction loss.

V. DISCUSSION

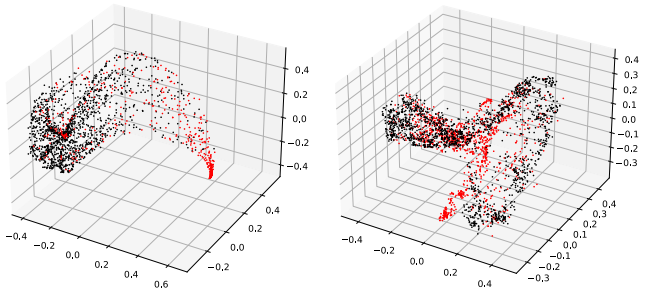
In this section, we discuss the potential challenges for ITD, including lack of labeled data, and adaptive attacks.

A. Lack of Labeled Data

Insider threat detection is faced with the problem of lacking labeled data. When deploying the proposed framework LAN in a real-world system, there may be a shortage of labeled data regarding the system. However, we provide three strategies to mitigate this problem. **First**, we can use the trained model that we already have to infer the samples in the



(a) LAN on Real-Time ITD (b) DeepLog on Real-Time ITD



(c) LAN on Post-Hoc ITD (d) DeepLog on Post-Hoc ITD

Fig. 7. Visualization of activity vectors. The black and red dots are the vectors of normal and abnormal activities, respectively.

new system, in order to obtain pseudo labels. Then, domain experts may verify the pseudo labels, especially the abnormal samples reported by the model, to obtain a ground-truth dataset for further training. **Second**, we can utilize weakly-supervised methods to train the activity-level detection models based on the course-grained labels (e.g., session- or user-level labels), which are relatively easier to obtain. Similar ideas have been successfully applied to fine-grained named entity recognition [54] and video anomaly detection [55]. **Third**, although the labeled data may not be sufficient, especially the abnormal samples, the proposed hybrid prediction loss aims at maximizing the use of all obtained signals, including both normal and abnormal signals, which may also mitigate the problem of lacking labeled data.

B. Adaptive Attacks

Attackers always try to evade detection by adaptive attacks. For example, adversarial samples are almost inevitable for a learning-based method [56], [57]. Moreover, abnormal users may intentionally disguise themselves as normal users, so as do evade insider threat detection. To address these adaptive attacks, we provide two strategies. The first strategy is to update the model frequently. Since the mode of running LAN is offline training and online inference, the training does not affect the inference. Therefore, we suggest updating the model when finding a new attack pattern or collecting a set of misclassified samples. We note that LAN can update the model weights incrementally without retraining from scratch. The second strategy is to ensemble multiple approaches (e.g., the feature engineering-based, the sequence-based, and the graph-based approaches) for insider threat detection, so as to enhance the overall robustness.

C. Detection Over Multiple Days or Sessions

In this paper, to conduct a fair comparison with the baselines, we follow previous studies [7], [8] to split activity sequences into sessions, where each session contains a sequence of activities between a user's logon and logoff. However, the proposed framework can be easily applied to different input lengths as required, and thus the attention span could be longer than a session. For instance, we can split user activity sequences by 7 days and apply LAN to detect insider threats that occur over multiple days without any modifications to the model. One interesting research direction is to learn attention spans adaptively for different datasets or organizations.

VI. Related Work

The research on ITD could date back to around 2000 [58], [59], with several rule-based and feature engineering-based approaches being proposed since then. Over the last decade, ITD has gained significant attention and progress with the rapid development of deep learning. In this section, we summarize the related work in the field of insider threat detection. Existing works can be grouped into three categories, i.e., the feature engineering-based methods, the sequence-based methods, and the graph-based methods. We also discuss the relationships and differences between ITD and user behavior analysis.

A. Feature Engineering-Based Methods

The first group of studies relies on feature engineering to detect insider threats [4], [5], [45], [60], [61]. Specifically, they extract the features for a user or a time period, such as the number of websites accessed on a shared PC, the number of sent emails, and the average size of email attachments. Based on the extracted features, a machine learning-based anomaly detection model is trained, such as logistic regression, random forest, neural network, XGBoost, autoencoder, and isolation forest. These studies detect insider threats at the user, weekly, daily, or session levels. Unlike these studies, in this paper, we conduct the first study on activity-level real-time ITD.

B. Sequence-Based Methods

Since feature engineering-based methods require extensive domain expert knowledge to select appropriate features for feature extraction, much effort has been made to automatically learn the representations of user behaviors. In recent years, deep learning techniques have gained much attention, and many works have introduced deep learning techniques for ITD. A natural practice is to aggregate user activities into an activity sequence and use sequence models in the natural language processing (NLP) field for anomaly detection [6], [7], [8], [16], [30], [62]. With sequence models, these studies can incorporate the temporal dependencies between user activities. Specifically, Yuan et al. [7] proposed a model that combines temporal point processes and recurrent neural networks for sequence-level ITD. After that, their follow-up work treated user behaviors as a sequence of activities and used few-shot learning to detect sequence-level insider threats [30]. Huang et al. [16] pre-trained a language model BERT [63] on the historical activity data and used a bidirectional LSTM for sequence-level detection. Tuor et al. [6] first extracted features for each user daily, and then fed the historical feature vectors

into an LSTM to predict the feature vector of the next day for daily-level ITD. Recently, Manoharan et al. [64] combined manually crafted features and sequential patterns extracted by LSTM to detect malicious users.

C. Graph-Based Methods

Recently, to incorporate more relationships between users and activities, graph neural networks have been widely used in the field of ITD [9], [10], [11], [65], [66]. Specifically, Jiang et al. [9] considered that user relationships can provide powerful information for detecting abnormal users. They modeled the relationships between users within an organization as a graph using email communication and user-based features, applying graph convolutional networks to detect insiders. Li et al. [10] converted user features and the user interaction structure into a heterogeneous graph and then used a dual-domain graph convolutional network to detect anomalous users. Liu et al. [11] represented user activities with nodes and designed several heuristic rules for graph construction. Finally, they constructed a heterogeneous graph, applied graph embedding algorithms on the graph, and utilized a clustering algorithm to detect anomalous activities. This is the only study that has focused on activity-level detection. However, it only considered post-hoc ITD and relied on heuristic rules designed by experts to construct graphs. Hong et al. [67] integrated manually selected features, sequential patterns, as well as organizational relationships together to detect daily anomalies. While they utilized a graph neural network to encode organizational relationships, their approach required a predefined graph and did not accommodate adaptive learning of relationships among users. In this paper, we employ graph structure learning to learn the user activity graph adaptively, avoiding the bias introduced by manual graph construction.

D. User Behaviour Analysis

User behavior analysis aims at analyzing the behaviors of users for a specific purpose. The concept has been applied to a wide range of applications, such as insider threat detection [11], [16], [64], [67], fraud detection [68], [69], [70], advanced persistent threat detection [71], [72], [73], human-computer interaction [74], [75], [76], user profiling prediction [77], [78], and recommendation system [42], [79], [80].

Insider threat detection is an important application of user behavior analysis. The task characteristics of insider threat detection include three aspects as follows. *(1) Stealthiness*. An insider threat is defined as "a current or former employee uses the authorized access to negatively affect the confidentiality, integrity, or availability of the organization's information or information systems" [2]. Therefore, an abnormal employee may disguise as a normal one. The stealthiness makes it difficult to detect potential insider threats. *(2) Irregularity*. The time intervals between adjacent activities are very irregular, ranging from less than a second to thousands of seconds. *(3) Multi-source heterogeneous data*. The system of an organization can monitor and record all types of user activities in the system chronologically (e.g., `send internal/external email`, `visit website`, `download file`, and `connect device`). The task of insider threat detection requires leverages the multi-source heterogeneous data to identify malicious user behaviors. Therefore, general user behavior analysis methods may not work in the research topic of insider threat detection.

VII. Conclusion

In this paper, we take the first step towards real-time ITD at the activity level. We present a fine-grained and efficient framework LAN, which can be applied for real-time ITD and post-hoc ITD with slight modifications. It leverages graph structure learning to autonomously learn the user activity graph without manual intervention, incorporating both the temporal dependencies and the relationships between user activities across different activity sequences. Furthermore, to mitigate the data imbalance issue in ITD, we also propose a novel hybrid prediction loss that integrates self-supervision signals from normal activities and supervision signals from abnormal activities into a unified loss. Extensive experiments demonstrate the effectiveness of LAN, superior to 9 state-of-the-art methods for real-time ITD and 8 competitive baselines for post-hoc ITD. We also conduct the ablation study and parameter analysis to measure the effectiveness of each component and hyper-parameters. One future plan is to integrate more effective sequence encoders and GNNs to improve performance and time efficiency. Moreover, since labeling abnormal samples requires much effort, we plan to design an interactive framework for anomaly detection.

References

[1] G. J. Silowash, D. M. Cappelli, A. P. Moore, R. F. Trzeciak, T. Shimeall, and L. Flynn, "Common sense guide to mitigating insider threats," CERT Insider Threat Center, Carnegie Mellon Univ., Pittsburgh, PA, USA, Tech. Rep. CMU/SEI-2015-TR-010, 2016.

[2] D. L. Costa, M. J. Albrethsen, and M. L. Collins, "Insider threat indicator ontology," Carnegie-Mellon Univ., Pittsburgh, PA, USA, Tech. Rep. CMU/SEI-2016-TR-007, 2016.

[3] Proofpoint, "2022 cost of insider threat global report," Ponemon, Traverse City, MI, USA, Tech. Rep., 2022. Accessed: Nov. 7, 2024. [Online]. Available: https://www.proofpoint.com/us/resources/threat-reports/cost-of-insider-threats

[4] D. C. Le, N. Zincir-Heywood, and M. I. Heywood, "Analyzing data granularity levels for insider threat detection using machine learning," *IEEE Trans. Netw. Service Manag.*, vol. 17, no. 1, pp. 30–44, Mar. 2020.

[5] L.-P. Yuan, E. Choo, T. Yu, I. Khalil, and S. Zhu, "Time-window based group-behavior supported method for accurate detection of anomalous users," in *Proc. 51st Annu. IEEE/IFIP Int. Conf. Dependable Syst. Netw. (DSN)*, Jun. 2021, pp. 250–262.

[6] A. Tuor, S. Kaplan, B. Hutchinson, N. Nichols, and S. Robinson, "Deep learning for unsupervised insider threat detection in structured cybersecurity data streams," 2017, *arXiv:1710.00811*.

[7] S. Yuan, P. Zheng, X. Wu, and Q. Li, "Insider threat detection via hierarchical neural temporal point processes," in *Proc. IEEE Int. Conf. Big Data (Big Data)*, Dec. 2019, pp. 1343–1350.

[8] M. Vinay, S. Yuan, and X. Wu, "Contrastive learning for insider threat detection," in *Proc. Int. Conf. Database Syst. Adv. Appl.* Cham, Switzerland: Springer, 2022, pp. 395–403.

[9] J. Jiang et al., "Anomaly detection with graph convolutional networks for insider threat and fraud detection," in *Proc. IEEE Mil. Commun. Conf. (MILCOM)*, Nov. 2019, pp. 109–114.

[10] X. Li et al., "A high accuracy and adaptive anomaly detection model with dual-domain graph convolutional network for insider threat detection," *IEEE Trans. Inf. Forensics Security*, vol. 18, pp. 1638–1652, 2023.

[11] F. Liu, Y. Wen, D. Zhang, X. Jiang, X. Xing, and D. Meng, "Log2vec: A heterogeneous graph embedding based approach for detecting cyber threats within enterprise," in *Proc. ACM SIGSAC Conf. Comput. Commun. Secur.*, Nov. 2019, pp. 1777–1794.

[12] S. Wang et al., "THREATRACE: Detecting and tracing host-based threats in node level through provenance graph learning," *IEEE Trans. Inf. Forensics Security*, vol. 17, pp. 3972–3987, 2022.

[13] C. Wang and H. Zhu, "Wrongdoing monitor: A graph-based behavioral anomaly detection in cyber security," *IEEE Trans. Inf. Forensics Security*, vol. 17, pp. 2703–2718, 2022.

[14] X. Hu, W. Gao, G. Cheng, R. Li, Y. Zhou, and H. Wu, "Toward early and accurate network intrusion detection using graph embedding," *IEEE Trans. Inf. Forensics Security*, vol. 18, pp. 5817–5831, 2023.

[15] F. Ullah, M. Edwards, R. Ramdhany, R. Chitchyan, M. A. Babar, and A. Rashid, "Data exfiltration: A review of external attack vectors and countermeasures," *J. Netw. Comput. Appl.*, vol. 101, pp. 18–54, Jan. 2018.

[16] W. Huang, H. Zhu, C. Li, Q. Lv, Y. Wang, and H. Yang, "ITDBERT: Temporal-semantic representation for insider threat detection," in *Proc. IEEE Symp. Comput. Commun. (ISCC)*, Sep. 2021, pp. 1–7.

[17] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Comput.*, vol. 9, no. 8, pp. 1735–1780, 1997.

[18] K. Cho et al., "Learning phrase representations using RNN encoder–decoder for statistical machine translation," 2014, *arXiv:1406.1078*.

[19] A. Vaswani et al., "Attention is all you need," in *Proc. Int. Conf. Neural Inf. Process. Syst.*, 2017, pp. 5998–6008.

[20] A. Ortega, P. Frossard, J. Kovačević, J. M. Moura, and P. Vandergheynst, "Graph signal processing: Overview, challenges, and applications," *Proc. IEEE*, vol. 106, no. 5, pp. 808–828, May 2018.

[21] V. Kalofolias, "How to learn a graph from smooth signals," in *Proc. 19th Int. Conf. Artif. Intell. Statist.*, 2016, pp. 920–929.

[22] Y. Chen, L. Wu, and M. Zaki, "Iterative deep graph learning for graph neural networks: Better and robust node embeddings," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 33, 2020, pp. 19314–19326.

[23] M. Belkin and P. Niyogi, "Laplacian eigenmaps and spectral techniques for embedding and clustering," in *Proc. Adv. Neural Inf. Process. Syst., Natural Synth.*, 2002, pp. 585–591.

[24] F. R. Chung, *Spectral Graph Theory*, vol. 92. Providence, RI, USA: AMS, 1997.

[25] T. N. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," 2016, *arXiv:1609.02907*.

[26] P. Veličković, G. Cucurull, A. Casanova, A. Romero, P. Liò, and Y. Bengio, "Graph attention networks," in *Proc. Int. Conf. Learn. Represent.*, 2018, pp. 1–12.

[27] V. Nair and G. E. Hinton, "Rectified linear units improve restricted Boltzmann machines," in *Proc. 27th Int. Conf. Mach. Learn.*, 2010, pp. 807–814.

[28] A. L. Maas et al., "Rectifier nonlinearities improve neural network acoustic models," in *Proc. 27th Int. Conf. Mach. Learn.*, Atlanta, GA, USA, 2013, vol. 30, no. 1, p. 3.

[29] H. Ding, Y. Sun, N. Huang, Z. Shen, and X. Cui, "TMG-GAN: Generative adversarial networks-based imbalanced learning for network intrusion detection," *IEEE Trans. Inf. Forensics Security*, vol. 19, pp. 1156–1167, 2024.

[30] S. Yuan, P. Zheng, X. Wu, and H. Tong, "Few-shot insider threat detection," in *Proc. 29th ACM Int. Conf. Inf. Knowl. Manag.*, Oct. 2020, pp. 2289–2292.

[31] M. AlSlaiman, M. I. Salman, M. M. Saleh, and B. Wang, "Enhancing false negative and positive rates for efficient insider threat detection," *Comput. Secur.*, vol. 126, Mar. 2023, Art. no. 103066.

[32] S. Yuan and X. Wu, "Deep learning for insider threat detection: Review, challenges and opportunities," *Comput. Secur.*, vol. 104, May 2021, Art. no. 102221.

[33] B. Lindauer. (Sep. 2020). *Insider Threat Test Dataset*. [Online]. Available: https://kilthub.cmu.edu/articles/dataset/Insider_Threat_Test_Dataset/12841247

[34] M. Buda, A. Maki, and M. A. Mazurowski, "A systematic study of the class imbalance problem in convolutional neural networks," *Neural Netw.*, vol. 106, pp. 249–259, Oct. 2018.

[35] J. L. Elman, "Finding structure in time," *Cogn. Sci.*, vol. 14, no. 2, pp. 179–211, Mar. 1990.

[36] J. Chung, C. Gulcehre, K. Cho, and Y. Bengio, "Empirical evaluation of gated recurrent neural networks on sequence modeling," 2014, *arXiv:1412.3555*.

[37] M. Du, F. Li, G. Zheng, and V. Srikumar, "DeepLog: Anomaly detection and diagnosis from system logs through deep learning," in *Proc. ACM SIGSAC Conf. Comput. Commun. Secur.*, Oct. 2017, pp. 1285–1298.

[38] B. Peng et al., "RWKV: Reinventing RNNs for the transformer era," 2023, *arXiv:2305.13048*.

[39] Y. Shen, E. Mariconti, P.-A. Vervier, and G. Stringhini, "Tiresias: Predicting security events through deep learning," in *Proc. ACM SIGSAC Conf. Comput. Commun. Security (CCS)*, 2018, pp. 592–605.

[40] G. Zhou et al., "Deep interest evolution network for click-through rate prediction," in *Proc. AAAI Conf. Artif. Intell.*, Jul. 2019, vol. 33, no. 1, pp. 5941–5948.

[41] Q. Chen, H. Zhao, W. Li, P. Huang, and W. Ou, "Behavior sequence transformer for e-commerce recommendation in Alibaba," in *Proc. 1st Int. Workshop Deep Learn. Pract. High-Dimensional Sparse Data*, Aug. 2019, pp. 1–4.

[42] K. Zhou, H. Yu, W. X. Zhao, and J.-R. Wen, "Filter-enhanced MLP is all you need for sequential recommendation," in *Proc. ACM Web Conf.*, 2022, pp. 2388–2399.

[43] Z. Wang, Z. Chen, J. Ni, H. Liu, H. Chen, and J. Tang, "Multi-scale one-class recurrent neural networks for discrete event sequence anomaly detection," in *Proc. 27th ACM SIGKDD Conf. Knowl. Discovery Data Mining*, Aug. 2021, pp. 3726–3734.

[44] A. L. Buczak and E. Guven, "A survey of data mining and machine learning methods for cyber security intrusion detection," *IEEE Commun. Surveys Tuts.*, vol. 18, no. 2, pp. 1153–1176, 2nd Quart., 2015.

[45] D. C. Le and N. Zincir-Heywood, "Anomaly detection for insider threats using unsupervised ensembles," *IEEE Trans. Netw. Service Manage.*, vol. 18, no. 2, pp. 1152–1164, Jun. 2021.

[46] I. J. King and H. H. Huang, "Euler: Detecting network lateral movement via scalable temporal link prediction," *ACM Trans. Privacy Secur.*, vol. 26, no. 3, pp. 1–36, 2023.

[47] W. J. Youden, "Index for rating diagnostic tests," *Cancer*, vol. 3, no. 1, pp. 32–35, 1950.

[48] D. C. Le and N. Zincir-Heywood, "Exploring anomalous behaviour detection and classification for insider threat identification," *Int. J. Netw. Manage.*, vol. 31, no. 4, p. e2109, Jul. 2021.

[49] I. Loshchilov and F. Hutter, "Decoupled weight decay regularization," 2017, *arXiv:1711.05101*.

[50] J. Johnson, M. Douze, and H. Jégou, "Billion-scale similarity search with GPUs," *IEEE Trans. Big Data*, vol. 7, no. 3, pp. 535–547, Jul. 2021.

[51] Y. A. Malkov and D. A. Yashunin, "Efficient and robust approximate nearest neighbor search using hierarchical navigable small world graphs," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 42, no. 4, pp. 824–836, Apr. 2020.

[52] K. Ethayarajh, "How contextual are contextualized word representations? Comparing the geometry of BERT, ELMo, and GPT-2 embeddings," 2019, *arXiv:1909.00512*.

[53] J. Gao, D. He, X. Tan, T. Qin, L. Wang, and T.-Y. Liu, "Representation degeneration problem in training natural language generation models," 2019, *arXiv:1907.12009*.

[54] S. Lee, S. Oh, and W. Jung, "Enhancing low-resource fine-grained named entity recognition by leveraging coarse-grained datasets," in *Proc. Conf. Empirical Methods Natural Lang. Process.*, 2023, pp. 3269–3279.

[55] A. Al-Lahham, N. Tastan, M. Z. Zaheer, and K. Nandakumar, "A coarse-to-fine pseudo-labeling (C2FPL) framework for unsupervised video anomaly detection," in *Proc. IEEE/CVF Winter Conf. Appl. Comput. Vis. (WACV)*, Jan. 2024, pp. 6779–6788.

[56] A. Shafahi, W. R. Huang, C. Studer, S. Feizi, and T. Goldstein, "Are adversarial examples inevitable?" in *Proc. 7th Int. Conf. Learn. Represent. (ICLR)*, 2019, pp. 1–17.

[57] A. Ilyas, S. Santurkar, D. Tsipras, L. Engstrom, B. Tran, and A. Madry, "Adversarial examples are not bugs, they are features," in *Proc. Adv. Neural Inf. Process. Syst.*, 2019, pp. 125–136.

[58] M. Maybury et al., "Analysis and detection of malicious insiders," in *Proc. Int. Conf. Intell. Anal.*, vol. 5, 2005, pp. 1–8.

[59] E. E. Schultz, "A framework for understanding and predicting insider attacks," *Comput. Secur.*, vol. 21, no. 6, pp. 526–531, Oct. 2002.

[60] P. A. Legg, O. Buckley, M. Goldsmith, and S. Creese, "Caught in the act of an insider attack: Detection and assessment of insider threat," in *Proc. IEEE Int. Symp. Technol. Homeland Secur. (HST)*, Apr. 2015, pp. 1–6.

[61] L. Liu, O. De Vel, C. Chen, J. Zhang, and Y. Xiang, "Anomaly-based insider threat detection using deep autoencoders," in *Proc. IEEE Int. Conf. Data Mining Workshops (ICDMW)*, Nov. 2018, pp. 39–48.

[62] J. Lu and R. K. Wong, "Insider threat detection with long short-term memory," in *Proc. Australas. Comput. Sci. Week Multiconf.*, Jan. 2019, pp. 1–10.

[63] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "BERT: Pre-training of deep bidirectional transformers for language understanding," 2018, *arXiv:1810.04805*.
[64] P. Manoharan, W. Hong, J. Yin, Y. Zhang, W. Ye, and J. Ma, "Bilateral insider threat detection: Harnessing standalone and sequential activities with recurrent neural networks," in *Proc. Int. Conf. Web Inf. Syst. Eng.* Cham, Switzerland: Springer, 2023, pp. 179–188.
[65] C. Zheng, W. Hu, and T. Li, "An insider threat detection method based on heterogeneous graph embedding," in *Proc. IEEE 8th Int. Conf. Big Data Security Cloud*, May 2022, pp. 11–16.
[66] W. Hong et al., "Graph intelligence enhanced bi-channel insider threat detection," in *Proc. Int. Conf. Netw. Syst. Secur.* Cham, Switzerland: Springer, 2022, pp. 86–102.
[67] W. Hong et al., "A graph empowered insider threat detection framework based on daily activities," *ISA Trans.*, vol. 141, pp. 84–92, Oct. 2023.
[68] A. Cherif, A. Badhib, H. Ammar, S. Alshehri, M. Kalkatawi, and A. Imine, "Credit card fraud detection in the era of disruptive technologies: A systematic review," *J. King Saud Univ. Comput. Inf. Sci.*, vol. 35, no. 1, pp. 145–174, Jan. 2023.
[69] J. Jiang, Y. Li, B. He, B. Hooi, J. Chen, and J. K. Z. Kang, "Spade: A real-time fraud detection framework on evolving graphs," *Proc. VLDB Endowment*, vol. 16, no. 3, pp. 461–469, Nov. 2022.
[70] F. Xiao, Y. Wu, M. Zhang, G. Chen, and B. C. Ooi, "MINT: Detecting fraudulent behaviors from time-series relational data," *Proc. VLDB Endowment*, vol. 16, no. 12, pp. 3610–3623, Aug. 2023.
[71] D. Han et al., "Anomaly detection in the open world: Normality shift detection, explanation, and adaptation," in *Proc. Netw. Distrib. Syst. Secur. Symp.*, 2023, pp. 1–18.
[72] F. Yang, J. Xu, C. Xiong, Z. Li, and K. Zhang, "PROGRAPHER: An anomaly detection system based on provenance graph embedding," in *Proc. 32nd USENIX Secur. Symp. (USENIX Secur.)*, 2023, pp. 4355–4372.
[73] H. Yue, T. Li, D. Wu, R. Zhang, and Z. Yang, "Detecting APT attacks using an attack intent-driven and sequence-based learning approach," *Comput. Secur.*, vol. 140, May 2024, Art. no. 103748.
[74] H. Mozannar, G. Bansal, A. Fourney, and E. Horvitz, "Reading between the lines: Modeling user behavior and costs in AI-assisted programming," in *Proc. CHI Conf. Human Factors Comput. Syst.*, May 2024, pp. 1–16.
[75] H. Mozannar, G. Bansal, A. Fourney, and E. Horvitz, "When to show a suggestion? Integrating human feedback in AI-assisted programming," in *Proc. AAAI Conf. Artif. Intell.*, 2024, vol. 38, no. 9, pp. 10137–10144.
[76] S. Barke, M. B. James, and N. Polikarpova, "Grounded copilot: How programmers interact with code-generating models," *Proc. ACM Program. Lang.*, vol. 7, no. OOPSLA1, pp. 85–111, Apr. 2023.
[77] B. Yang et al., "Empowering general-purpose user representation with full-life cycle behavior modeling," in *Proc. 29th ACM SIGKDD Conf. Knowl. Discovery Data Mining*, 2023, pp. 2908–2917.
[78] Y. Wu et al., "Personalized prompt for sequential recommendation," *IEEE Trans. Knowl. Data Eng.*, vol. 36, no. 7, pp. 3376–3389, Jul. 2024.
[79] X. Xie et al., "Contrastive learning for sequential recommendation," in *Proc. IEEE 38th Int. Conf. Data Eng. (ICDE)*, May 2022, pp. 1259–1273.
[80] X. Li, L. Sun, M. Ling, and Y. Peng, "A survey of graph neural network based recommendation in social networks," *Neurocomputing*, vol. 549, Sep. 2023, Art. no. 126441.

Xiangrui Cai received the Ph.D. degree in computer science from Nankai University, China. He is currently working as an Associate Professor with the College of Computer Science, Nankai University. His research interests include time series analysis, natural language processing, and AI safety.

Yang Wang received the B.S. degree from the School of Electrical and Information, Northeast Agricultural University, in 2022. He is currently pursuing the M.S. degree with the College of Cyber Science, Nankai University. His research interests include data mining, user behavior modeling, and anomaly detection.

Sihan Xu received the B.Sc. and Ph.D. degrees in computer science from Nankai University, in 2013 and 2018, respectively. For her research, she spent a year with the National University of Singapore. She is currently working as an Associate Professor with the College of Cyber Science, Nankai University. Her research interests include software engineering, big data analysis, and AI security.

Hao Li received the M.S. and Ph.D. degrees in electronics and communication engineering from the Communication University of China, in 2015 and 2018, respectively. His current research interests include artificial intelligence and its security, digital right management, cloud computing security, and attribute-based encryption.

Ying Zhang received the Ph.D. degree from Nankai University, China, in 2013. From 2011 to 2013, she studied at Purdue University, USA. She is currently working as a Professor with the College of Computer Science, Nankai University. Her research interests include sentiment analysis, multi-modal data analysis, and information retrieval.

Zheli Liu received the B.Sc. and M.Sc. degrees in computer science and the Ph.D. degree in computer application from Jilin University, China, in 2002, 2005, and 2009, respectively. After a post-doctoral fellowship with Nankai University, he joined the College of Cyber Science, Nankai University, in 2011, where he is currently working a Professor. His research interests include applied cryptography and data privacy protection.

Xiaojie Yuan received the B.S., M.S., and Ph.D. degrees in computer science from Nankai University. She is currently working as a Professor with the College of Cyber Science, Nankai University. She leads a research group working on topics of database, data mining, and information retrieval.