Dual-prompting based Event Anomaly Detection in Dynamic Graphs

Haodan Ran[1,4], Yang Fang[2(✉)], Jiuyang Tang[3], Weiming Zhang[1], Jinzhi Liao[3], and Xiang Zhao[3]

[1] National Key Laboratory of Information Systems Engineering, National University of Defense Technology, Changsha, China
[2] National University of Defense Technology, Changsha, China
[3] Laboratory for Big Data and Decision, National University of Defense Technology, Changsha, China
[4] College of Information and Communication, National University of Defense Technology, Wuhan, China
{ranhaodan,fangyang12,jiuyang_tang,wmzhang,liaojinzhi12,xiangzhao}@nudt.edu.cn

Abstract. Event anomaly detection in dynamic graphs, such as the identification of abnormal transaction behavior on e-commerce platforms, focuses on detecting anomalous activities as the graphs evolve over time. While many approaches have demonstrated significant advantages in managing complex graph structures, relatively few have addressed the core challenges in dynamic graph anomaly detection, namely the heavy reliance on labeled data and the insufficient mining and integration of spatial and temporal information. In this paper, we present DPEAD, a method for detecting event anomalies in dynamic graphs, which uses a pre-training and prompt fine-tuning framework. First, a self-supervised pre-training approach is employed to alleviate the scarcity of labeled data. Second, a dual-prompt mechanism for anomaly detection is developed to improve the integration and utilization of spatiotemporal information, effectively bridging the gap between pre-training and downstream tasks. Specifically, DPEAD utilizes both snapshot-view and event-view approaches to model dynamic graphs at different stages, enabling the effective extraction of dynamic information. Our experimental results, based on four real-world datasets, demonstrate that DPEAD achieves superior performance compared to existing state-of-the-art approaches.

Keywords: Anomaly detection · Dynamic graph · Prompt fine-tuning.

1 Introduction

Anomaly detection is the task of identifying data points in a dataset that deviate significantly from expected patterns or behaviors. This task is crucial in various applications, such as fraud detection, system health monitoring, and network security, where recognizing unusual or potentially harmful activities is essential. Dynamic graphs represent non-linear data structures that evolve over time and are commonly found in a variety of real-world applications, including social networks, financial transactions, and communication systems. In these graphs, the formation of edges indicates the occurrence of specific behaviors. For example, in e-commerce networks, purchasing behaviors represent interactions between users and items. These behaviors, or edges, are referred to as events and provide crucial insights into the structure and dynamics of the graph. Therefore, event anomaly detection in dynamic graphs is a critical task that helps identify anomalous behaviors, security threats, and system malfunctions.

Given its importance, considerable effort has been devoted to event anomaly detection in dynamic graphs. Statistical and probabilistic methods [1,9,22,23] detect anomalies by comparing observed data with model predictions, highlighting deviations from the normal. However, these methods often struggle to handle large datasets efficiently, having impractically long computation times and suboptimal anomaly detection in large-scale and frequently evolving dynamic graphs. Recently, deep learning-based methods have shown greater effectiveness in addressing anomaly detection, especially in scenarios involving large and complex datasets. Among these, Graph Neural Networks (GNNs) have emerged as a mainstream approach. Notable examples include AddGraph [9], TADDY [18], and SAD [25]. These methods have demonstrated superior performance by leveraging the powerful representation capabilities of GNNs. Despite the progress in dynamic graph anomaly detection, existing methods still struggle with the unique complexities and temporal dynamics of such graphs, posing significant challenges to traditional techniques.

Challenge 1. Insufficient labeled data. Anomaly detection often relies on well-labeled datasets to learn the characteristics of normal behavior and to identify deviations that may indicate anomalies. However, in many real-world applications, obtaining sufficient labeled data is challenging due to several factors, including the high cost of manual labeling, the rarity of anomalies, and the complexity of the data. Although semi-supervised methods have been explored in existing studies [18,25], these approaches may still struggle to capture the complex and dynamic features necessary for understanding network behavior and identifying anomalies.

Challenge 2. Insufficient mining of dynamic information. Currently, the common approach to modeling dynamic graphs involve treating them either as a collection of snapshots at discrete timestamps or as a sequence of temporal events. The snapshot approach simplifies the management and analysis of dynamic graphs but overlooks the inherent continuity of their temporal evolution. Conversely, the temporal event sequence approach captures the continuity of the graph but may fail to provide a comprehensive understanding of its evolutionary processes. Consequently, effectively leveraging the strengths of both approaches for the mining of spatial and temporal information remains a key challenge.

Challenge 3. Insufficient integration of spatiotemporal information. Existing methods often focus either on the structural features of graphs at specific timestamps or on the temporal evolution of graphs, but they fail to effectively integrate both dimensions. This limitation can result in a biased understanding of graph behaviors, potentially missing critical patterns that emerge from the interplay between structure and time. Therefore, effectively integrating spatiotemporal information is a key challenge.

To address the above challenges, we propose a novel **D**ual-**P**rompting based **E**vent **A**nomaly **D**etection in Dynamic Graphs (DPEAD). DPEAD is based on a pre-training and prompt fine-tuning framework. To the best of our knowledge, this is the first dynamic graph anomaly detection method to adopt this framework. In the pre-training stage, we employ a self-supervised contrastive learning method based on a snapshot-view graph, which captures the normal patterns of graphs without relying on labeled data. In anomaly detection, we adopt an event-view graph approach, which is more suitable for event detection and can further extract features of the graph in continuous time. This two-view mining approach, working in tandem, comprehensively captures the evolving nature of dynamic graphs. To bridge the gap with the pre-training task, we design a dual-prompt fine-turning mechanism that integrates both structural and temporal prompts. Anomaly detection results are generated through prompt-based answers, guiding the model to perform the downstream task. The main contributions of this work are summarized as follows:

- We propose a pre-training and prompt fine-tuning framework for event anomaly detection in dynamic graphs. A unique perspective of the framework is the adoption of a two-view mining approach, which combines snapshot-view and event-view to effectively mine spatiotemporal information.
- We introduce a novel dual-prompt fine-tuning mechanism to bridge the gap between pre-training and downstream tasks, thereby enhancing the integration of both structural and temporal information.
- Through comprehensive experiments on four dynamic graph datasets, we demonstrate that DPEAD outperforms other state-of-the-art baselines, highlighting its effectiveness and robustness.

2 Related Work

2.1 Graph Prompt Fine-tuning.

Prompt fine-tuning has emerged as a powerful technique for narrowing the gap between pre-training and downstream tasks. Motivated by the success of prompt fine-tuning in Natural Language Processing (NLP) [17], researchers have begun to adapt this concept to the graph domain[6].

Graph prompt fine-tuning is initially applied to static graph tasks, where it demonstrates notable success and significant performance improvements across a variety of graph-related tasks [4,10]. Building on the progress in static graphs, researchers have shifted their focus to dynamic graphs. Specifically, [5] proposed a prompt generator based on temporal difference coding to provide temporally aware prompts for various dynamic graph tasks.

2.2 Anomaly Detection in Dynamic Graphs.

Anomaly detection has garnered significant attention due to its broad range of applications, including cyber-security [16], fraud detection [26], and biological research [29]. From a structural perspective, anomalies in graphs can be categorized into three types: node anomaly [8], edge(or event) anomaly[2], and subgraph anomaly [12]. Due to the evolving nature of graph structures over time, event anomaly detection is a particularly important and popular task in dynamic graphs.

Initially, event anomaly detection in dynamic graphs primarily relied on statistical and probabilistic methods. Notable examples include GOutlier [1], CAD [23], CM-Sketch [22], StreamSpot [19] and SpotLight [9]. With the continuous advancement of deep learning techniques, anomaly detection methods based on deep learning have found widespread application in dynamic graphs. NetWalk [28] proposes a method that obtains node embeddings in dynamic graphs using a clique embedding approach and employed clustering-based techniques to calculate anomaly scores for edges. AddGraph [9] presents an end-to-end framework based on an attention model with extended temporal Graph Convolutional Network (GCN) for anomaly detection. StrGNN [3] adopts GCN to extract node features from h-hop subgraphs around the target edge in each snapshot. A Gated Recurrent Unit (GRU) is then employed to capture the temporal information, which is crucial for anomaly detection. TADDY [18] proposes a multi-encoding strategy for nodes, using a transformer-based framework to achieve spatiotemporal coupling in dynamic graphs. SAD [25] employs a semi-supervised method that combines a memory bank and pseudo-labeling with contrastive learning to detect anomalous events in graph streams.

3 Preliminaries

3.1 Dynamic Graph Modeling

Definition 1 (Snapshot-view Graph). A snapshot-view graph (SVG) is a series of discrete graph snapshots, each associated with a specific timestamp, up to a maximum timestamp $\mathcal{T}$. Formally, an SVG is denoted as $\mathcal{G} = \{\mathcal{G}_{t_i}\}_{t_i=1}^{\mathcal{T}}$, where each $\mathcal{G}_{t_i} = \{\mathcal{V}_{t_i}, \mathcal{E}_{t_i}\}$ represents the graph snapshot at timestamp t_i. Here, $\mathcal{V}_{t_i}$ and $\mathcal{E}_{t_i}$ are the sets of nodes and edges, respectively, in $\mathcal{G}_{t_i}$.

Definition 2 (Event-view Graph). An event-view graph (EVG) is a dynamic graph represented by a sequence of events with continuous timestamps. An EVG is denoted as $\mathcal{G} = \{e(t_i)\}_{t_i=1}^{\mathcal{T}}$, where each event is defined as $e(t_i) = (u_{t_i}, v_{t_i}, t_i)$, and $e(t_i)$ represents that node u_{t_i} and v_{t_i} have an interaction at timestamp t_i.

Definition 3 (Event Anomaly Detection in Dynamic Graphs). Event anomaly detection in dynamic graphs (EADDG) involves identifying unusual or unexpected events that deviate from normal behaviors. Formally, an anomalous event in $\mathcal{G} = \{e(t_i)\}_{t_i=1}^{\mathcal{T}}$ is an interaction that deviates from the expected behavior. In this paper, we define event anomaly detection in dynamic graphs as an event classification problem. Specifically, the goal is to find a classification model to predict the label $y_i \in \{0, 1\}$ for each event $e(t_i)$, where $y_i = 0$ indicates a normal event and $y_i = 1$ indicates an anomalous event.

4 Proposed Approach

4.1 Overall Framework

The overall framework of our proposed model DPEAD is illustrated in Fig. 1. DPEAD employs a self-supervised dynamic graph contrastive learning pre-

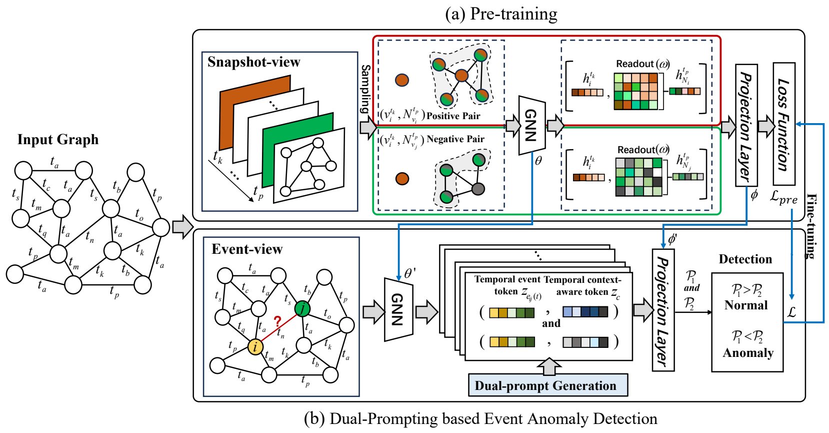


Fig. 1. Overall framework of DPEAD. The framework consists of two components: pre-training and dual-prompting based event anomaly detection. In the pre-training phase, we samples two snapshots t_k, t_p of target node v_i and extracts positive pairs $(v_i^{t_k}, N_i^{t_p})$ and negative pairs $(v_i^{t_k}, N_j^{t_p})$ for training. In the event anomaly detection section, we mark the target edge as red (timestamp is t_n). The detailed procedure of the dual-prompt mechanism is illustrated in Fig 2.

training method to capture the normal patterns of dynamic graphs under a snapshot-view. Subsequently, we introduce a dual-prompt fine-tuning mechanism to guide the anomaly detection process under an event-view.

4.2 Pre-training

In most real-world scenarios, only a small fraction of events are anomalous, while the majority are normal. Therefore, we adopt a self-supervised method for pre-training without labeled data to capture the normal patterns of the graph. Leveraging the assumption of temporal translation invariance proposed in [27], which posits that the characteristics of the same node should remain consistent across different timestamps, we use node pairs from different snapshots as comparative pairs in a snapshot-view graph. This phrase aims to maximize the global information at the node level and capture the normal patterns of the graph.

Snapshot Sampling. Given a SVG $\mathcal{G} = \{\mathcal{G}_{t_i}\}_{t_i=t_{\min}}^{t_{\max}}$ with time span $\Delta t = t_{\max} - t_{\min}$, the set of timestamps obtained by the sampling strategy $\mathbb{R}_{\text{sample}}$ is defined as follows:

$$I = \mathbb{R}_{\text{sample}}\left(n, s, \Delta t\right), \tag{1}$$

where $\mathbb{R}_{\text{sample}}$ returns a random set of sample timestamps $I = (t_{s_1}, t_{s_2}, ..., t_{s_n})$, the hyperparameter n is the number of snapshots sampled, the hyperparameter s controls the timespan size, and the timespan between each snapshot is $\Delta t/s$. And $t_{s_i} \in I = \left[(t_{\min} + \Delta t/2s), (t_{\max} - \Delta t/2s)\right], \forall i \in [1, n]$.

Then, we get a set of snapshots as $\mathcal{G}_t = \left\{\mathcal{G}_{t_{s_1}}, \mathcal{G}_{t_{s_2}}, ..., \mathcal{G}_{t_{s_n}}\right\}$ corresponding to set I.

Node Pair Selection. For the node $v_i^{t_k}$ (i.e., v_i in snapshot $\mathcal{G}_{t_k}$), positive pairs are formed with its neighbors $N_i^{t_p}$ in snapshot $\mathcal{G}_{t_p}$ ($t_p \neq t_k$), denoted as $(v_i^{t_k}, N_i^{t_p})$, $N_i^{t_p} = \left\{v_j | (v_i, v_j, t_p) \in \mathcal{G}_{t_p} \textit{ and } t_p \in I\right\}$. Negative pairs are formed with the neighbors of other nodes $v_j^{t_p}$ in snapshot $\mathcal{G}_{t_p}$ ($i \neq j, t_p \neq t_k$), denoted as $(v_i^{t_k}, N_j^{t_p})$, $N_j^{t_p} = \left\{v_m | (v_j, v_m, t_p) \in \mathcal{G}_{t_p} \textit{ and } t_p \in I\right\}$.

GNN Encoder. For each snapshot $\mathcal{G}_{t_i} = \{X_{t_i}, A_{t_i}\}$, where X_{t_i} is the feature matrix and A_{t_i} is the adjacency matrix. Here, we employ a GCN [13] with parameters θ as the encoder. Node embeddings produced by the snapshot $\mathcal{G}_{t_i}$ are denoted as $H_{t_i} = f_{\mathrm{GCN}}\left(X_{t_i}, A_{t_i}; \theta\right)$.

Readout Function. After passing through the encoder to obtain the embeddings of all nodes across all snapshots, and taking $h_i^{t_p}$ as an example, we compute the aggregated embeddings of each node's neighbors using the readout function:

$$h_{N_i}^{t_p} = \mathrm{Readout}(H_{N_i^{t_p}}, \omega) = \mathrm{POOL}\left(\left\{h_s | v_s \in N_i^{t_p}\right\}\right), \tag{2}$$

where Readout$(\cdot)$ denotes the readout function with parameters ω, and $N_i^{t_p}$ represents the neighbor set of node v_i in $\mathcal{G}_{t_p}$. The matrix $H_{N_i^{t_p}}$ contains the embeddings of these neighbors, and each row corresponds to a neighbor's embedding (i.e., h_s, and $v_s \in N_i^{t_p}$).

Loss Function. The loss function is designed to maximize the similarity scores of positive sample pairs while minimizing the similarity scores of negative sample pairs. In this paper, we adopt the Deep InfoMax [11] and define the loss function as:

$$\mathcal{L}_{\mathrm{pre}} = -\sum_{i=1}^{N}\sum_{p=1}^{M}\sum_{k\neq p}^{M} \log \frac{\exp\left(\mathcal{P}_\phi\left(h_i^{t_p}, h_{N_i}^{t_k}\right)/\tau\right)}{\sum_{j=1}^{V}\exp\left(\mathcal{P}_\phi\left(h_i^{t_p}, h_{N_j}^{t_k}\right)/\tau\right)}, \tag{3}$$

where M is the number of snapshots obtained from sampling. For each training epoch, N nodes are sampled from multiple snapshots as a minibatch. $\mathcal{P}$ is the projection head with parameters ϕ used to evaluate the similarity score of node pairs, such as a multiple-layer perceptron (MLP). And τ is a temperature hyperparameter.

Furthermore, the objective of the loss function $\mathcal{L}_{\mathrm{pre}}$ is to minimize the discrepancy between these similarity scores and the true matching status, as indicated by the indicator function Match$(\cdot)$. The goal is to adjust the parameters θ, w, and ϕ so that the model can more effectively distinguish between matching and non-matching pairs.

$$\arg\min_{\theta,w,\phi} \mathcal{L}_{\mathrm{pre}}\left(\mathcal{P}_\phi\left(h_i^{t_p}, h_{N_n}^{t_k}\right), \mathrm{Match}(h_i^{t_p}, h_{N_n}^{t_k})\right), \tag{4}$$

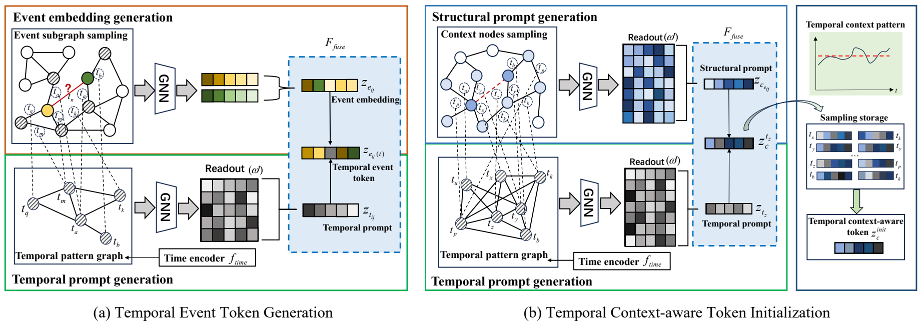


Fig. 2. The generation of the dual-prompt. In (a), the red solid line indicates the detected target event at timestamp t_n, illustrating the temporal event token generation process. In (b), the red dashed line marks the labeled event at timestamp t_z, serving as an instance of event category c, which illustrates the temporal context-aware token generation process.

where the supervisory signal is derived from the matching function $\mathrm{Match}(h_i^{t_p}, h_{N_n}^{t_k})$.

4.3 Prompting for Event Anomaly Detection

After snapshot-view pre-training, in this phase, we adopt the event-view method for anomaly detection. This two-view approach can mining more dynamic information. In EVG, it represents the dynamic graph as $\mathcal{G} = \{e_{ij}(t)\}_{t=t_{\min}}^{t_{\max}}$. To facilitate more efficient transfer of pre-trained knowledge to event anomaly detection, we propose a dual-prompt method.

Prompt Design. Inspired by [24], we propose a prompt function f_{prompt}, which transforms an input event $e_{ij}(t) = (v_i, v_j, t)$ into a prompt z_c using a predefined template for a token pair. This transformation is expressed as:

$$\mathbf{p}_{ij} = f_{\mathrm{prompt}}(e_{ij}(t)) = \left[z_{e_{ij}(t)}, z_{\mathrm{c}}\right], \tag{5}$$

where $z_{e_{ij}}(t)$ is a temporal event token, and z_{c} is a learnable embedding of the event type in anomaly detection, called a temporal context-aware token.

We design a dual-prompt method to generate the two tokens mentioned above. The generation process is illustrated in Fig. 2. The dual-prompt consists of two parts: (1) Temporal prompt, which captures temporal information in graphs, such as the temporal feature of the event and context; (2) Structural prompt, which captures connectivity patterns and context relationships of the events.

(1) Temporal Prompt. Temporal information is a characteristic property that cannot be ignored in the dynamic evolution of graphs, as it plays a crucial role in capturing the temporal patterns and changes within the graph structure. However, the temporal information within the graph structure is complex. If it is only treated as a simple weighted aggregation of the event attributes along with the structural information [5], it will result in information loss. This approach fails to fully capture the nuanced dynamics and temporal dependencies that are essential for a comprehensive understanding of the graph's evolution. Therefore, we propose a temporal pattern graph construction approach to generate temporal prompts. The detailed process is as follows.

Temporal Pattern Graph. In a k-hop subgraph centered on $e_{ij}(t)$, the set of timestamps of all events in the subgraph $\mathcal{S}_{e_{ij}(t)}$ is $T_{\text{sub}} = \{t_1, t_2, ..., t_n\}, n = |T_{\text{sub}}|$. Afterward, we construct the temporal pattern graph $\mathcal{G}^T_{e_{ij}}$. In this graph, the nodes represent the timestamps from T_{sub}, referred to as time nodes. Subsequently, we employ a time encoder f_{time} to encode these timestamps and map them to the embedding space of the nodes in the event subgraph $\mathcal{S}_{e_{ij}(t)}$. Features of time nodes are obtained as follows:

$$h_{t_i} = f_{\text{time}}(t_i), t_i \in T_{sub}, \tag{6}$$

where t_i represents the timestamps associated with the events in the subgraph, and h_{t_i} is the resulting temporal feature vector. The time encoder f_{time} consists of two linear layers followed by a ReLU activation layer.

If events from $\mathcal{S}_{e_{ij}(t)}$ share a common node, then there is an edge between the corresponding time nodes in $\mathcal{G}^T_{e_{ij}}$. It should be noted that although the same timestamps may appear on different events in $\mathcal{S}_{e_{ij}(t)}$, there will be no duplicate time nodes or parallel edges when constructing $\mathcal{G}^T_{e_{ij}}$. Each unique timestamp corresponds to a single time node, and the presence of an edge between time nodes indicates that the corresponding events in $\mathcal{S}_{e_{ij}(t)}$ share at least one common node. This construction method ensures that the temporal relationships and interactions within $\mathcal{S}_{e_{ij}(t)}$ are accurately represented.

Finally, we use the pre-trained GNN to process the nodes in $G^T_{e_{ij}}$ and apply readout operations to node embeddings to summarize the information. The temporal prompt denoted as $z_{T_{\text{sub}}}$:

$$z_{T_{\text{sub}}} = \text{Readout}(H_{\text{T}}, \omega) = \text{POOL}\left(\{h_{t_i} | t_i \in T_{sub}\}\right), \tag{7}$$

where the readout function with parameter ω is taken from the pre-training phase, and H_{T} is the node embedding matrix of $\mathcal{G}_{e^T_{ij}}$.

(2) Structural Prompt. In graphs, the category of an event can be represented by a k-hop subgraph centered on the event, referred to as the context subgraph. The set of nodes in the subgraph, called context nodes set, is obtained by sampling as $\{v_1, v_2, ..., v_N\}$. Their embeddings can be represented by a matrix H_{sub}, where each row represents a node's embedding h_{v_i}. The structural prompt can be generated by:

$$z_{c_{e_{ij}}} = \text{Readout}(H_{\text{sub}}, \omega) = \text{POOL}\left(\left\{h_{v_i} | v_i \in C_{e_{ij}(t)}\right\}\right), \tag{8}$$

where the readout function with parameter ω is taken from the pre-training phase, and $C_{e_{ij}(t)}$ is the context subgraph of the event $e_{ij}(t)$.

Prompt Generation. We designed a dual-prompt fusion method to generate the temporal event token and the temporal context-aware token. To better capture the nonlinear relationships between them, we use a multi-layer perceptron (MLP) as the fusion function f_{fuse}.

(1)Temporal Event Token Generation. First, we sample the event subgraph to generate temporal pattern graph of the target event $e_{ij}(t)$. As illustrated in Fig. 2(a), we obtain the temporal event token as follows:

$$z_{e_{ij}(t)} = f_{\text{fuse}}(z_{e_{ij}}, z_{t_{ij}}), \tag{9}$$

where $z_{t_{ij}}$ is the temporal prompt generated by temporal pattern graph, capturing the temporal features associated with $e_{ij}(t)$. f_{fuse} is the fusion function, which combines the event embedding and temporal prompt into a time-aware representation.

Typically, the embedding of an event is computed as the average of the embeddings of its two endpoint nodes, which can be expressed as:

$$z_{e_{ij}} = 1/2(h_i + h_j), \tag{10}$$

where h_i and h_j are the embeddings of nodes v_i and v_j, and obtained from the pre-trained GNN.

The detailed steps are as follows. First, we concatenate the event embedding $z_{e_{ij}}$ and temporal prompt $z_{t_{ij}}$ to obtain a 2d-dimensional vector $z_{\text{concat}} = [z_{e_{ij}}; z_{t_{ij}}]$. Then, the concatenated vector z_{concat} is processed through a multilayer perceptron (MLP). The MLP is designed to learn a nonlinear mapping from the concatenated vector z_{concat} to the final temporal event token $z_{e_{ij}(t)}$.

(2) Temporal Context-Aware Token Initialization. Temporal context-aware token is a learnable vector, and thus, its initialization problem needs to be properly addressed. For efficiency, we only sample M context nodes from the context subgraph during the token initialization, ensuring that the computational cost remains manageable while still capturing the essential structural information. As presented in Fig. 2 (b), we follow these steps.

First, considering that normal patterns of dynamic graphs may be perturbed over time, and to minimize the impact of this perturbation on the temporal context-aware token, we sample 'anomalous' and 'normal' labeled events separately. Similar to the temporal event token generation, the temporal context-aware token of each sampled event is computed as follows:

$$z_c^{t_k} = f_{\text{fuse}}(z_{c_{e_{ij}}}, z_{t_k}), \tag{11}$$

where $z_{c_{e_{ij}}}$ is the structural prompt that captures the context information surrounding the event. z_{t_k} is the temporal prompt generated by the temporal pattern graph, representing the temporal information of the context subgraph.

Then, we store sampled temporal context-aware tokens in a sampling memory. This ensures that we have a diverse set of contexts that can be used to initialize the temporal context-aware token. Finally, we compute the mean values of the stored context information. This helps smooth out the noise and provides a more stable and representative initial value for the temporal context-aware token. The initialization is as follows:

$$z_c^{init} = \frac{1}{N}\sum_{k=1}^{N} z_c^{t_k}, \tag{12}$$

where N is the number of sampled events, and $z_c^{t_k}$ represents the context token of the k-th sampled event.

Prompt-based answer. In anomaly detection, events are usually categorized into two classes: anomalous and normal, represented as $c \in \{c_\mathrm{a}, c_\mathrm{n}\}$. Specially, the temporal context-aware token can be represented as a learnable prompt matrix $\mathrm{Z_p} = [z_{c_\mathrm{a}}, z_{c_\mathrm{n}}]^\mathsf{T} \in \mathbb{R}^{2\times d}$.

Each event $e_{ij}(t)$ is input into the prompt function f_prompt, resulting in two token pairs: $\left[z_{e_{ij}(t)}, z_{c_\mathrm{a}}\right]$ and $\left[z_{e_{ij}(t)}, z_{c_\mathrm{n}}\right]$. These token pairs are concatenated and fed into a pre-trained projection head $\mathcal{P}_{\phi'}$ to obtain an evaluation probability. If $\mathcal{P}_{\phi'}(z_{e_{ij}(t)}, z_{c_\mathrm{n}}) > \mathcal{P}_{\phi'}(z_{e_{ij}(t)}, z_{c_\mathrm{a}})$, the $e_{ij}(t)$ is classified as normal. Otherwise, it is classified as anomalous.

Prompt-based fine-tuning. By following the pre-training objective through our designed prompts, we can optimize the following loss function for fine-tuning:

$$\arg\min_{\theta',\phi'} \mathcal{L}_\mathrm{pre}\left(\mathcal{P}_{\phi'}\left(z_{e_{ij}(t)}, z_c\right); \mathrm{Match}(e_{ij}(t), c)\right), \tag{13}$$

where $\mathrm{Match}(e_{ij}(t), c)$ is an indicator function:

$$\mathrm{Match}(e_{ij}(t), c) = \begin{cases} 1 & \text{if } e_{ij}(t) \text{ is labeled } c, \\ 0 & \text{otherwise.} \end{cases} \tag{14}$$

In this way, the optimization process after prompt-based fine-tuning aims to tune the parameters θ' and ϕ' so that the model can accurately determine the event type, thereby enabling effective event anomaly detection. To prevent overfitting and enhance class separability, we introduce an orthogonal constraint on the prompt matrix, $\mathcal{L}_C = \frac{1}{2}\left\|\mathrm{Z_p}\mathrm{Z_p}^\mathsf{T} - \mathrm{I}\right\|_2^2$, ensuring that the learned representations are more distinct and generalize better. Finally, the loss function is defined as:

$$\mathcal{L} = \mathcal{L}_\mathrm{pre}\left(\mathcal{P}_{\phi'}\left(z_{e_{ij}(t)}, z_c\right); \mathrm{Match}(e_{ij}(t), c)\right) + \lambda\mathcal{L}_C, \tag{15}$$

where λ is a hyperparameter that adjusts the importance of this constraint.

As a result, we narrow the optimization objective gap, thereby unifying the pre-training and anomaly detection tasks. This alignment ensures that the model not only benefits from the rich representations learned during pre-training but also effectively adapts to the specific requirements of anomaly detection, leading to improved performance and generalization.

5 Experiment

5.1 Experimental Setup

Datasets. We select four real-world dynamic graph benchmark datasets to evaluate the proposed model, i.e., UCI Messages dataset [20], Digg dataset [7], Bitcoin-OTC [15] and Bitcoin-Alpha [14]. The statistics of these datasets are summarized in Table 1.

For the pre-processing of the datasets, we follow the methodology outlined in previous work [28]. In each dataset, edges are annotated with timestamps, and node features are initialized using randomly generated vectors. We use the method described in [18] to inject anomalous edges.

Table 1. Statistics of datasets.

Dataset	# Nodes	# Edges	# Time span	# Avg. Degree
UCI Messages	1,899	59,835	4 months	14.57
Digg	30,360	87,627	48 months	5.61
Bitcoin-Alpha	5,881	35,588	60 months	12.80
Bitcoin-OTC	3,783	24,186	61 months	12.10

Baselines. We compare the proposed framework with state-of-the-art baselines, i.e., GOutlier [1], CM-Sketch [22], DeepWalk [21], NetWalk [28], Addgraph [9], StrGNN [3], TADDY [18] and SAD [25]. Detailed descriptions of these methods are presented in Section 2.

Experimental Design. In the pre-training phase, we adopt an unsupervised setup with the linear evaluation scheme as described in [11]. For the anomaly detection task, the pre-trained model is loaded, and we inject the dataset with anomalies at three different percentages: 1%, 5%, and 10%. The dataset is divided into a training set and a test set in chronological order, with the first 50% of timestamps designated as the training set. Given the scarcity of labels, we sample all available anomaly samples within the training set. To ensure that the anomaly rate in the training set matches the global anomaly rate, we additionally sample negative samples (i.e., normal nodes without anomalies) for training. We use the Area Under the Curve (AUC) value as a performance metric for all methods.

Parameter Setting. We use a two-layer GCN as the base encoder with a batch size of 256. For UCI Message and Digg dataset, the size of the dimension of embedding is 64. For Bitcoin-Alpha and Bitcoin-OTC, the size of the dimension of embedding is 32. The number of snapshots is set as follows: 1,000 for UCI Messages and Digg, and 2,000 for Bitcoin-Alpha and Bitcoin-OTC. In the pre-training phase, 1,000 epochs are used for training, with a learning rate of 4×10^{-3}. In the anomaly detection phase, varying anomaly ratios ($\eta = \{1\%, 5\%, 10\%\}$), the k-hop subgraph centered on the target event are selected to generate the temporal pattern graph, where $k = 1$. For context token initialization, the number of sampled nodes $n = 30$ in the UCI Message, Digg, and Bitcoin-Alpha, and $n = 35$ in the Bitcoin-OTC. 500 epochs are used for training, with a learning rate of 1×10^{-3}, and $\lambda = 0.01$.

5.2 Anomaly Detection Results on Benchmark Datasets

We perform anomaly event detection experiments on four benchmark datasets, comparing DPEAD to eight baseline methods. Table 2 demonstrates the average AUC performance of all methods for anomaly detection on the four datasets.

Across four datasets and a range of anomaly proportions (1%,5% and 10%), DPEAD consistently outperformed all baseline methods in terms of AUC. Even when compared to state-of-the-art baselines, our method achieves an average improvement of 2.3% in AUC across the four datasets.

Table 2. Overall performance of all methods in terms of AUC on the dynamic graph event anomaly detection task.

Methods	**UCI Messages**			**Digg**			**Bitcoin-Alpha**			**Bitcoin-OTC**		
	1%	**5%**	**10%**	**1%**	**5%**	**10%**	**1%**	**5%**	**10%**	**1%**	**5%**	**10%**
GOutlier	0.7181	0.7053	0.6707	0.6963	0.6763	0.6353	0.6685	0.6621	0.6239	0.7150	0.7017	0.6632
CM-Sketch	0.7270	0.7086	0.6861	0.6871	0.6581	0.6179	0.6903	0.6815	0.6784	0.7352	0.7232	0.6935
DeepWalk	0.7514	0.7391	0.6979	0.7808	0.6881	0.6396	0.6985	0.6874	0.6793	0.7423	0.7356	0.7287
NetWalk	0.7758	0.7674	0.7226	0.7563	0.7176	0.6837	0.8385	0.8357	0.8350	0.7785	0.7694	0.7534
AddGraph	0.8083	0.8090	0.7688	0.8341	0.8470	0.8369	0.8632	0.8514	0.8665	0.8885	0.8621	0.8739
StrGNN	0.8179	0.8252	0.7959	0.8162	0.8254	0.8272	0.8574	0.8667	0.8627	0.9012	0.8775	0.8836
TADDY	0.8912	0.8398	0.8370	0.8617	0.8545	0.8440	0.9451	0.9341	0.9423	0.9455	0.9340	0.9425
SAD	0.9025	0.8812	0.8670	0.8716	0.8652	0.8340	0.9488	0.9405	0.9301	0.9520	0.9432	0.9483
DPEAD	**0.9323**	**0.9257**	**0.9306**	**0.8974**	**0.8815**	**0.8634**	**0.9535**	**0.9537**	**0.9467**	**0.9588**	**0.9552**	**0.9506**

Across varying anomaly proportions, the performance of DPEAD exhibited greater stability compared to other baseline methods. This is especially noticeable on the UCI Messages, Bitcoin-Alpha and Bitcoin-OTC, where the fluctuation in metrics is consistently within approximately 1%. The observed stability is largely due to the flexibility and robustness of the pre-training and prompt fine-tuning framework, which allows the model to adapt effectively to different data distributions and anomaly scenarios.

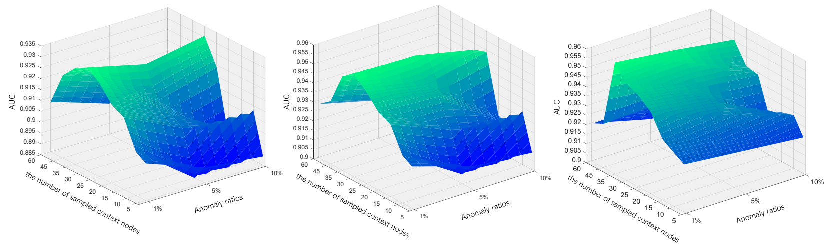
Compared to statistical methods (GOutlier, CM-Sketch) and graph embedding methods (DeepWalk, NetWalk), models utilizing GNNs exhibit superior AUC performance in the context of anomaly detection within dynamic graphs. This enhancement is attributed to the improved structural awareness provided by the inductive mechanism in GNNs, which consequently improves the precision of outlier detection.

In the UCI Messages dataset, DPEAD shows a significant improvement in performance metrics, with an average increase of 4.63% over other baseline approaches. This notable advantage can be attributed to DPEAD's enhanced ability to capture complex spatiotemporal dependencies, which are particularly critical for detecting anomalies in social networks. Specifically, the superior performance of DPEAD is due to its design that integrates temporal information and structural information through the dual-prompt mechanism. This mechanism enables DPEAD to better grasp the dynamic interactions and positional relationships within the network, thus enhancing the accuracy of anomaly detection.

5.3 Parameter Sensitivity Analysis

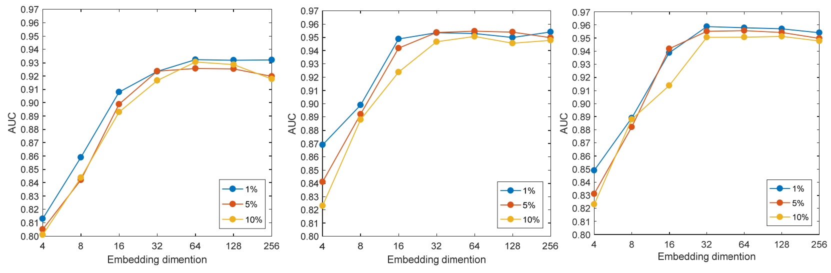
Considering the efficiency issue, we choose three lightweight datasets — UCI Messages, Bitcoin-Alpha, and Bitcoin-OTC — to start the experiments.

Analysis of Temporal Context-aware Token Initialization. To evaluate the impact of the number of sampled context nodes on model performance, we further investigate the variation in AUC across different numbers of sampled context nodes ($n = \{5, 10, 15, 20, 25, 30, 35, 45, 60\}$) and varying anomaly ratios ($\eta = \{1\%, 5\%, 10\%\}$). The results are illustrated in Fig. 3. As the number of



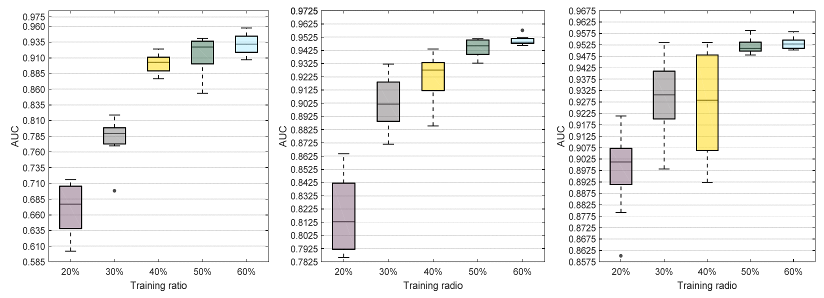
(a) UCI Messages (b) Bitcoin-Alpha (c) Bitcoin-OTC

Fig. 3. Impact of the context nodes number for temporal context-aware token initialization on three datasets.



(a) UCI Messages (b) Bitcoin-Alpha (c) Bitcoin-OTC

Fig. 4. The sensitivity of embedding dimension on three datasets.



(a) UCI Messages (b) Bitcoin-Alpha (c) Bitcoin-OTC

Fig. 5. AUC results for different training ratios on three datasets.

context nodes increases, the model performance initially exhibits a gradual improvement. After the performance reaches its peak, it begins to decline. We attribute this trend to the introduction of noise when an excessive number of sampled context nodes are included. Notably, the number of context nodes at which performance peaks varies across different datasets.

Analysis of the GNN Embedding Dimensionality. To analyze the impact of the size of dimension of hidden state on the performance of our GNN model, we conduct a series of experiments by varying the dimensions of the node embeddings ($d = \{4, 8, 16, 32, 64, 128, 256\}$). We consider the variation of the model's performance over varying anomaly ratios. To further illustrate the findings, the results are presented in Fig. 4. For UCI Messages, $d = 64$ appears to be the optimal choice, as it provides the best balance between model performance and computational efficiency. For the Bitcoin-Alpha and Bitcoin-OTC datasets, the AUC values tend to plateau when $d = 32$. This suggests that an embedding dimension of 32 provides a sufficient representational capacity for these datasets, beyond which further increases in dimensionality do not significantly improve performance.

Analysis of the Training Radio. To evaluate the impact of different training ratios on model performance, we conduct experiments with varying training ratios under a 10% anomaly condition. The results are presented in Fig. 5. From the experimental results, it is evident that as the training ratio increases, the model's performance shows an upward trend. This can be attributed to the fact that increasing the training data effectively increases the number of both anomalous and normal samples.

5.4 Ablation Study

To evaluate the effectiveness of each constituent element within the DPEAD, a comparative analysis is conducted against the following variants under a 10% anomaly condition.

- **W/O structural prompt**. In this variant, the structural prompt is entirely omitted. The structural prompt is directly replaced with the embedding of the labeled event.
- **W/O temporal prompt**. Here, we remove the temporal prompt from the model, eliminating the incorporation of temporal information into both the event and context tokens.
- **W/O prompt initialization**. This variant skips the initialization setup phase for prompts and opts instead for randomly initialized temporal context-aware tokens.
- **W/O constraint**. This variant removes the constraint term $\lambda \mathcal{L}_C$ as defined in Equation (15) from the model.

The results in Table 3 reveal that: 1) Compared to DPEAD, the W/O structural prompt variant shows a decrease in AUC. This highlights the importance of the structural prompt in enhancing the model's ability to detect anomalies. 2) The W/O temporal prompt variant also exhibits a decrease in AUC. It suggests that the inclusion of the temporal prompt significantly improves the model's performance. 3) W/O prompt initialization variant shows an average reduction in AUC of approximately 6.85% compared to DPEAD. During the initialization of context tokens, the model can capture a richer set of information, providing more supervisory signals for training. 4) The W/O constraint variant also shows a decrease in AUC. It underscores the critical importance of the orthogonal constraint in enhancing the model's overall performance.

Table 3. Results of the ablation study.

Model	UCI Messages	Bitcoin-Alpha	Bitcoin-OTC
DPEAD	0.9306	0.9467	0.9506
W/O structural prompt	0.8913	0.9045	0.9053
W/O temporal prompt	0.9104	0.9125	0.9257
W/O prompt initialization	0.8781	0.8682	0.8879
W/O constraint	0.9291	0.9458	0.9501

6 Conclusion

In this paper, we propose an dual-prompting based event anomaly detection framework in dynamic graph, aimed at addressing the following challenges in the field: the dependency on labeled data and the inadequate mining and integration of spatiotemporal information. To further enhance the extraction of dynamic information, we design a two-view modeling approach. By employing a pre-training and prompt fine-tuning framework, we develop a self-supervised pre-training method that effectively mitigates the scarcity of labeled data. Moreover, we design a dual-prompt fine-tuning mechanism, which enhances the model's capability to capture and utilize complex patterns and dynamics within the data. Experimental results show that DPEAD significantly outperforms existing state-of-the-art methods across four real-world datasets. In the future, we intend to extend the model to accommodate anomaly detection tasks for various types of dynamic graph data, such as dynamic heterogeneous graphs and attributed graphs.

Acknowledgments. This work was partially supported by National Key R&D Program of China (No. 2022YFB3102600), NSFC (Nos. 62272469, 62306322, 72301284), and The Science and Technology Innovation Program of Hunan Province (No. 2023RC1007).

References

1. Aggarwal, C.C., et al.: Outlier detection in graph streams. In: ICDE. pp. 399–409 (2011)
2. Bhatia, S., et al.: Real-time anomaly detection in edge streams. ACM Trans. Knowl. Discov. Data **16**(4), 75:1–75:22 (2022)
3. Cai, L., et al.: Structural temporal graph neural networks for anomaly detection in dynamic graphs. In: CIKM. pp. 3747–3756 (2021)
4. Chen, C., et al.: Dipping plms sauce: Bridging structure and text for effective knowledge graph completion via conditional soft prompting. In: ACL. pp. 11489–11503 (2023)
5. Chen, X., et al.: Prompt learning on temporal interaction graphs. CoRR **abs/2402.06326** (2024)
6. Chen, Z., et al.: GPL-GNN: graph prompt learning for graph neural network. Knowl. Based Syst. **286**, 111391 (2024)
7. Choudhury, M.D., et al.: Social synchrony: Predicting mimicry of user actions in online social media. In: CSE. pp. 151–158 (2009)
8. Ekle, O.A., Eberle, W.: Dynamic pagerank with decay: A modified approach for node anomaly detection in evolving graph streams. In: FLAIRS (2024)

9. Eswaran, D., et al.: Spotlight: Detecting anomalies in streaming graphs. In: KDD. pp. 1378–1386 (2018)
10. Ge, Q., et al.: PSP: pre-training and structure prompt tuning for graph neural networks. In: ECML PKDD. vol. 14945, pp. 423–439 (2024)
11. Hjelm, R.D., et al.: Learning deep representations by mutual information estimation and maximization. In: ICLR (2019)
12. Huang, L., et al.: Hybrid-order anomaly detection on attributed networks. IEEE Trans. Knowl. Data Eng. **35**(12), 12249–12263 (2023)
13. Kipf, T.N., Welling, M.: Semi-supervised classification with graph convolutional networks. In: ICLR (2017)
14. Kumar, S., et al.: Edge weight prediction in weighted signed networks. In: ICDM. pp. 221–230 (2016)
15. Kumar, S., et al.: REV2: fraudulent user prediction in rating platforms. In: WSDM. pp. 333–341 (2018)
16. Liao, J., et al.: DPDGAD: A dual-process dynamic graph-based anomaly detection for multivariate time series analysis in cyber-physical systems. Adv. Eng. Informatics **61**, 102547 (2024)
17. Liu, P., et al.: Pre-train, prompt, and predict: A systematic survey of prompting methods in natural language processing. ACM Comput. Surv. **55**(9), 195:1–195:35 (2023)
18. Liu, Y., et al.: Anomaly detection in dynamic graphs via transformer. IEEE Trans. Knowl. Data Eng. **35**(12), 12081–12094 (2023)
19. Manzoor, E.A., et al.: Fast memory-efficient anomaly detection in streaming heterogeneous graphs. CoRR **abs/1602.04844** (2016)
20. Panzarasa, P., et al.: Patterns and dynamics of users' behavior and interaction: Network analysis of an online community. J. Assoc. Inf. Sci. Technol. **60**(5), 911–932 (2009)
21. Perozzi, B., et al.: Deepwalk: online learning of social representations. In: KDD. pp. 701–710 (2014)
22. Ranshous, S., et al.: A scalable approach for outlier detection in edge streams using sketch-based approximations. In: SIAM. pp. 189–197. SIAM (2016)
23. Sricharan, K., Das, K.: Localizing anomalous changes in time-evolving graphs. In: SIGMOD. pp. 1347–1358 (2014)
24. Sun, M., et al.: GPPT: graph pre-training and prompt tuning to generalize graph neural networks. In: KDD. pp. 1717–1727 (2022)
25. Tian, S., et al.: SAD: semi-supervised anomaly detection on dynamic graphs. In: IJCAI. pp. 2306–2314 (2023)
26. Wen, Z., et al.: Voucher abuse detection with prompt-based fine-tuning on graph neural networks. In: CIKM. pp. 4864–4870 (2023)
27. Xu, Y., et al.: CLDG: contrastive learning on dynamic graphs. In: ICDE. pp. 696–707 (2023)
28. Yu, W., et al.: Netwalk: A flexible deep embedding approach for anomaly detection in dynamic networks. In: Guo, Y., Farooq, F. (eds.) KDD. pp. 2672–2681 (2018)
29. Yuan, Z., et al.: Motif-level anomaly detection in dynamic graphs. IEEE Trans. Inf. Forensics Secur. **18**, 2870–2882 (2023)