International Journal of Machine Learning and Cybernetics
https://doi.org/10.1007/s13042-024-02521-9

ORIGINAL ARTICLE

Attribute encoding transformer on unattributed dynamic graphs for anomaly detection

Shang Wang[1] · Haihong Hao[1] · Yuan Gao[1] · Xiang Wang[1] · Xiangnan He[1]

Received: 29 June 2024 / Accepted: 24 December 2024

Abstract
Dynamic graphs represent connections in complex systems changing over time, posing unique challenges for anomaly detection. Traditional static graph models and shallow dynamic graph methods often fail to capture the temporal dynamics and interactions effectively, limiting their ability to detect anomalies accurately. In this work, we introduce the Attribute Encoding Transformer (AET), a novel framework specifically designed for anomaly detection in unattributed dynamic graphs. The AET integrates advanced encoding strategies that leverage both spatial and historical interaction data, enhancing the model's ability to identify anomalous patterns. Our approach includes a Link Prediction Pre-training methodology that optimizes the transformer architecture for dynamic contexts by pre-training on link prediction tasks, followed by fine-tuning for anomaly detection. Comprehensive experiments on four real-world datasets demonstrate that our framework outperforms the state-of-the-art methods in detecting anomalies, thereby addressing key challenges in dynamic graph analysis. This study not only advances the field of graph anomaly detection but also sets a new benchmark for future research on dynamic graph data analysis.

Keywords Unattributed dynamic graphs · Anomaly detection · Attribute encoding · Transformer

1 Introduction

Recently, research on dynamic graph structure data has gained increasing attention. Unlike traditional static graphs, dynamic graphs, which represent data evolving over time, more accurately reflect the true dynamics and variability inherent in datasets. This characteristic is notably absent in static graph data. For instance, in real-world datasets such as social networks [8, 17], financial transaction networks [7], and citation graphs [11, 30, 31], nodes represent individual users or entities, and edges represent interactions between them. The positions of nodes and edges, as well as their interactions, are in constant flux, making static graph analysis techniques inadequate for analyzing such data. For example, in real financial transaction networks like Bitcoin-Alpha [12], it is crucial to detect anomalies as new edges form and old edges vanish continuously. Static graph modeling only captures network information at a specific moment, failing to ascertain ongoing anomalous behaviors. Therefore, robust dynamic graph analysis techniques are essential for modeling dynamic graphs to capture temporal variations and unearth hidden structural information.

In the context of anomaly detection on dynamic graphs, shallow methods such as CM-Sketch [18] and Goulier [1], which rely on model structural analysis and historical behavior analysis, have demonstrated limited efficacy in handling large graph datasets. Recently, deep learning-based approaches have shown remarkable success in dynamic graph learning due to their efficiency in processing large datasets. For instance, NetWalk [28] employs deep graph embedding techniques coupled with cluster-based anomaly detection; AddGraph [33] and StrGNN [3] utilize end-to-end deep neural network models to address anomaly detection; EvolveGCN [4] segments the dynamic graph into discrete time-step snapshots, integrating Recurrent Neural Networks (RNNs) and various Graph Neural Network (GNN) modules to capture both temporal information and spatial

✉ Xiangnan He
xiangnanhe@gmail.com

Shang Wang
cwws@ustc.mail.edu.cn

Haihong Hao
haohaihong@ustc.mail.edu.cn

[1] University of Science and Technology of China, Hefei, China

Published online: 24 January 2025

topological details of graphs. Although these advancements have improved the performance of dynamic graph modeling, several key limitations still persist.

Although some progress has been made in dynamic graph anomaly detection [24], existing methods still face multiple challenges. Firstly, effectively capturing the dynamism of graphs and the complex dependencies over time is a critical issue [20, 29]. Many current dynamic graph learning methods rely on simplified temporal models, such as decomposing the graph into a series of snapshots at discrete time points [21], which may overlook critical information within the temporal gaps. Additionally, most methods fail to effectively integrate spatial and temporal information of graphs, leading to inaccurate or inefficient detection of anomalous patterns in dynamic graphs. For instance, some methods based on Graph Convolutional Networks (GCNs) [33] can handle the spatial structure of graphs but are limited in capturing the evolution of graphs over time. Furthermore, the data volume to be processed is often enormous, posing higher demands on the computational efficiency and scalability of algorithms. Overall, the challenges associated with anomaly detection in unattributed dynamic graphs can be summarized as follows:

- Encoding nodes and edges in unattributed graphs is difficult, and currently, there is no established paradigm for encoding nodes or edges in such graphs. Each node or edge lacks initial features, and due to changes over time or privacy issues that prevent access, it is challenging to directly construct node attribute information from dynamic graph data.
- Previous methods have not sufficiently encoded spatial and historical interaction information [25]. Particularly with historical interactions, earlier methods did not account for the dynamic nature of dynamic graphs. In dynamic graphs, the same edge often appears multiple times at different timestamps, and existing methods typically use static graph encoding strategies that eliminate duplicate edges, which is not reasonable. The repetition of an edge at multiple different timestamps often implies that the edge is more significant. The time modeling in dynamic graphs is either short-term or coarse-grained. On one hand, the method of dividing dynamic graphs into discrete snapshots discards fine-grained temporal information, making it difficult to capture short-term interactions within the graph. On the other hand, using static graph modeling methods on individual discrete snapshots makes capturing long-term dependencies in historical graph data challenging.
- Previous attempts to apply the transformer architecture to anomaly detection in unattributed graphs have not fully leveraged the performance capabilities of the transformer structure. Although TADDY utilized a transformer to capture the spatial-temporal features coupled with graph structure, it lacked an effective training strategy. Due to its large number of parameters [22], the transformer structure often requires appropriate pre-training tasks and fine-tuning on the target task to perform optimally. Most previous works were trained directly on anomaly detection tasks, resulting in suboptimal outcomes.

To address the aforementioned challenges, the main contributions of this paper are as follows:

- We propose a novel encoding paradigm for edges in unattributed graphs. This encoding process considers both global and local spatial information of the target edge, as well as the edge's historical interaction information including the relative temporal sequence and frequency of repetitions. This approach ensures that the generated encodings contain richer information, which better supports downstream tasks.
- We introduce a link prediction pre-training methodology, which is both simple and efficient. This method enables the transformer architecture to fully leverage its capabilities when applied to anomaly detection in unattributed dynamic graphs. The model, after pre-training and fine-tuning, produces high-quality embeddings that enhance support for anomaly detection tasks.
- Extensive experiments have been conducted using our Attribute Encoding Transformer (AET) on existing public unattributed dynamic graphs datasets. The performance of our method surpasses that of other existing approaches, demonstrating the effectiveness of our encoding paradigm and pre-training and fine-tuning strategy.

2 Related work

2.1 Encoding for nodes or edges

In anomaly detection methodologies, particularly within graphs with node attributes like text graphs, encoding processes typically involve encoding the textual attributes of nodes [2, 14]. For instance, in Graphformers [26] the textual features of nodes are independently encoded by language models. However, these approaches face a significant challenge: they are unable to generate new node attributes when original node attributes are absent. In real-world scenarios, due to concerns about privacy and security, nodes and edges often lack attribute information, and commonly used dynamic graph datasets also do not include attributes for nodes or edges. Therefore, methods that explicitly encode node or edge attributes cannot be directly applied to unattributed dynamic graphs [27]. Nevertheless, we can center

on these nodes or edges, sampling the surrounding nodes or edges, and using these surroundings as attributes of the center. For example, by focusing on an edge, we can sample neighboring nodes and use the structural properties of these neighbors as attributes of the central node as Taddy [13]. Since dynamic graphs contain temporal information, often represented as timestamps in datasets, for nodes without attributes, the dynamic graph can include historical interaction data between two nodes, such as the number of interactions and the timing of these interactions, which can also serve as attributes for encoding the nodes.

2.2 Transformers in graph

Transformer [22], a potent neural network architecture based on self-attention mechanisms, was first introduced for tasks in natural language processing. Subsequently, Bert [6] built upon the transformer by incorporating pre-training techniques, thereby broadening its applications and extending its application to areas such as multimodality [10]. Due to issues such as over-smoothing and over-squashing encountered in Graph Neural Networks (GNNs), some researchers have adapted the transformer for graph data learning, effectively mitigating these problems. For instance, Graphformer [27] diversifies node encoding by integrating node-specific feature sets into the transformer architecture, while [15] employs a graph masking attention mechanism that incorporates graph-related prior knowledge before the transformer. GraphBERT [32] constructs a Bert-like network model and uses a pre-trained self-supervised model aimed at embedding learning in static graphs. Recently, As the state-of-the-art model, Teddy [13] has introduced the transformer structure into anomaly detection tasks specifically designed for dynamic graphs. In contrast, our work focuses more on the historical interaction information within dynamic graphs and employs a novel pre-training method to optimize transformer performance in dynamic graph anomaly detection tasks.

3 Question definition

In this section, we define the dynamic graph and the problem of dynamic graph anomaly detection. 1 Let T be the maximum timestamp. A graph steam is represented by $\mathbb{G} = \{\mathcal{G}\}_{t=1}^{T}$, where each $\mathcal{G}^t = \left(\mathcal{V}^t, \mathcal{E}^t\right)$ represents the snapshot at timestamp t, and $\mathcal{V}^t$ and $\mathcal{E}^t$ are the set of nodes and edges respectively. An edge $e_{i,j}^t = \left(v_i^t, v_j^t\right) \in \mathcal{E}^t$, means that the i-th node and the j-th node have a connection in the graph at the timestamp t, where $v_i^t, v_j^t \in \mathcal{V}^t$. We use $n^t = \left|\mathcal{V}^t\right|$ and $m^t = \left|\mathcal{E}^t\right|$ to denote the number of nodes and edges at timestamp t respectively. A adjacency matrix $\mathbf{A}^t \in \mathbb{R}^{n^t \times n^t}$ is used to represented $\mathcal{G}^t$, if there is a link between two nodes, $\mathbf{A}_{i,j}^t = 1$, otherwise, $\mathbf{A}_{i,j}^t = 0$.

The goal of this work is to detect the anomalous edges in each timestamp. Concretely, for each $e_{i,j}^t \in \mathcal{E}^t$, the model produces $f\left(e_{i,j}^t\right)$, the anomalous probability of $e_{i,j}^t$, where $e_{i,j}^t$ is a learnable anomaly score function. In the training phase, we do not use the labelled anomalous data, but in the testing phase we leverge the abnormal binary labels. Specifically, if the edge $e_{i,j}^t$ is an anomalous edges, the label $y_{e_{i,j}^t} = 1$, otherwise, the label $y_{e_{i,j}^t} = 0$.

All important notations in this paper are summarized in Table 1.

4 Method

Given a dynamic graph $\mathbb{G} = \{\mathcal{G}\}_{t=1}^{T}$, our goal is to identify a fake interaction edge $\epsilon^t \notin \mathcal{E}^t$, which should not exist in the graph. To achieve this, we have developed the Attribute Encoding Transformer (AET) on Unattributed Dynamic Graphs for Anomaly Detection. Specifically, our AET model consists of four main components: the Negative

Table 1 Glossary of notations

Notation	Definition
$\mathbb{G} = \{\mathcal{G}\}_{t=1}^{T}$	A graph steam with a maximum timestamp of T
$\mathcal{G}^t = \left(\mathcal{V}^t, \mathcal{E}^t\right)$	The snapshot graph at timestamp t
$\mathcal{V}^t$	The node set at timestamp t
$\mathcal{E}^t$	The edge set at timestamp t
$e_{i,j}^t = \left(v_i^t, v_j^t\right) \in \mathcal{E}^t$	An edge between v_i^t and v_j^t at the timestamp t
$\epsilon^t \notin \mathcal{E}^t$	An edge not in $\mathcal{E}^t$
n^t	The number of nodes at timestamp t
m^t	The number of edges at timestamp t
$\mathbf{A}^t$	The binary adjacency matrix at timestamp t
$f(\cdot)$	Anomaly score function
$\mathcal{S}(e_{i,j}^t)$	The substructure node set of target edge $e_{i,j}^t$
$\mathbf{X}_{glb}(V)$	The global spatial encoding of the node set
$\mathbf{X}_{loc}(V)$	The local spatial encoding of the node set
$\mathbf{X}_{temp}(V)$	The relative temporal encoding of the node set
$\mathbf{X}(e_{i,j}^t)$	The encoding matrix of target edge $e_{i,j}^t$
$\mathbf{Q}^{(l)}$	The query matrix of the l-th layers of transformer
$\mathbf{K}^{(l)}$	The key matrix of the l-th layers of transformer
$\mathbf{V}^{(l)}$	The value matrix of the l-th layers of transformer
$\mathbf{H}^{(l)}$	The output embedding of the l-th layers of transformer
k	The number of contextual nodes
λ	The size of time window
L	The number of layers of transformer

Sample Generator, Substructure Sampling, Spatial Historical Encoder, and the transformer Module. To maximize the potential of the transformer module, our model employs a training strategy that begins with pre-training followed by fine-tuning on the target task, which we refer to as link prediction pre-training. In Sect. 4.1, we introduce the relevant preliminary knowledge. Subsequently, in Sect. 4.2, we present the overall pipeline of our model. Sections 4.3 and 4.4 detail some key structures within our model. In Sect. 4.5, we provide a comprehensive description of our link prediction pre-training process, including the specific loss functions and other details.

4.1 Our attribute encoding transformer (AET)

The overview of our proposed framework is depicted in Fig. 1. Here, we sequentially describe the functionalities of the four components mentioned above. Since unattributed dynamic graphs typically contain only positive samples, the negative sample generator is essential both during pre-training and fine-tuning to artificially construct negative samples needed for training. We unify edge anomalies in dynamic graphs with Link Prediction, where Link Prediction serves as a pre-training task and anomaly detection as a fine-tuning task. At timestamp t, we randomly sample a set number of nodes as candidates for anomalous samples. We then verify that these node pairs do not belong to the existing set of normal edges across all training timestamps, as done in previous methods like TADDY. Once positive and negative samples are obtained, the substructure sampling module operates by centering on an edge to sample a substructure, which serves as a surrogate for edge attributes. This substructure, along with historical interaction data extracted from the original graph, is then processed using the spatial historical encoder to generate an edge encoding that can represent edge attributes. Subsequently, these edge encodings are treated as individual tokens input into the transformer Module, where they interact with one another through the module's multi-head attention mechanism, enhancing the encoding of the dynamic attributes of the graph. Finally, the output edge encoding is utilized for the link prediction task during pre-training or the Anomaly Detection task during fine-tuning.

4.2 Substructure sampling

Before conducting substructure sampling on the samples, it is essential to have both positive and negative samples, necessitating the use of a negative sample generator to produce the negative samples.

For the link prediction pre-training task, we need to determine whether two nodes are connected; hence, an edge formed by connecting two randomly unconnected nodes serves as a negative sample for link prediction. However, during the fine-tuning phase for the Anomaly Detection task, the challenge arises because our training dataset lacks real negative samples. How can we generate pseudo-negative samples from positive samples? Inspired by Taddy [13], we use edges that do not exist in the original dynamic graph as negative samples. Specifically, just like in the negative sample generator for link prediction pre-training, we select two random nodes and verify that these nodes are not connected at any timestamp. An edge connecting these two random nodes, therefore, represents an edge that does not exist in the original dynamic graph, thus providing a negative sample.

Formally, we define graph diffusion $\mathbf{S} \in \mathbb{R}^{n\times n}$ by

$$\mathbf{S} = \sum_{k=0}^{\infty} \Theta_k \mathbf{T}^k \in \mathbb{R}^{n\times n} \tag{1}$$

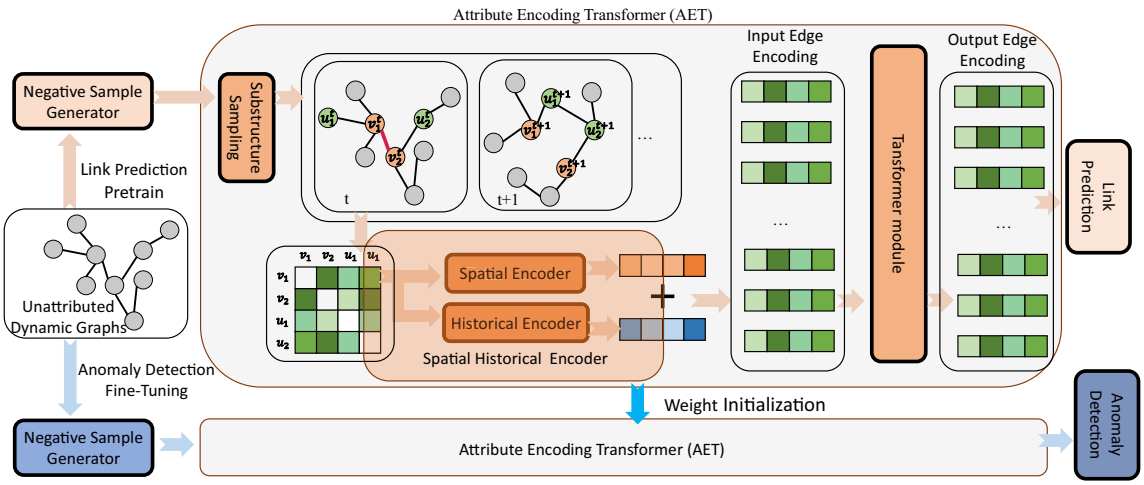


Fig. 1 Pipeline of attribute encoding transformer on unattributed dynamic graphs for anomaly detection

where $\mathbf{T} \in \mathbb{R}^{n\times n}$ is the generalized transition matrix and Θ is the weighting coefficient which determines the ratio of global–local information. The adjacency matrix $\mathbf{A} \in \mathbb{R}^{n\times n}$ and the diagonal degree matrix $\mathbf{D} \in \mathbb{R}^{n\times n}$ are used by generalized graph diffusion to define the Personalized PageRank(PPR) instantiations. PPR chooses $\mathbf{T} = \mathbf{AD}^{-1}$ and $\theta_k = \alpha(1-\alpha)^k$ where $\alpha \in (0, 1)$ is the teleport probability. The solutions to PPR is formulated as:

$$\mathbf{S}^{PPR} = \alpha\left(\mathbf{I_n} - (1-\alpha)\mathbf{D}^{-1/2}\mathbf{A}\mathbf{D}^{-1/2}\right)^{-1} \tag{2}$$

From a global perspective, a row s_i of the diffusion matrix $\mathbf{S}$ represents the connectivity between the i-th node and the other nodes. After that, to gain the representation of the target edge $e_{i,j} = (v_1, v_2)$, we can choose a fix number of most important nodes, and compute the connectivity of $e_{i,j}$ by adding the connectivity vectors.

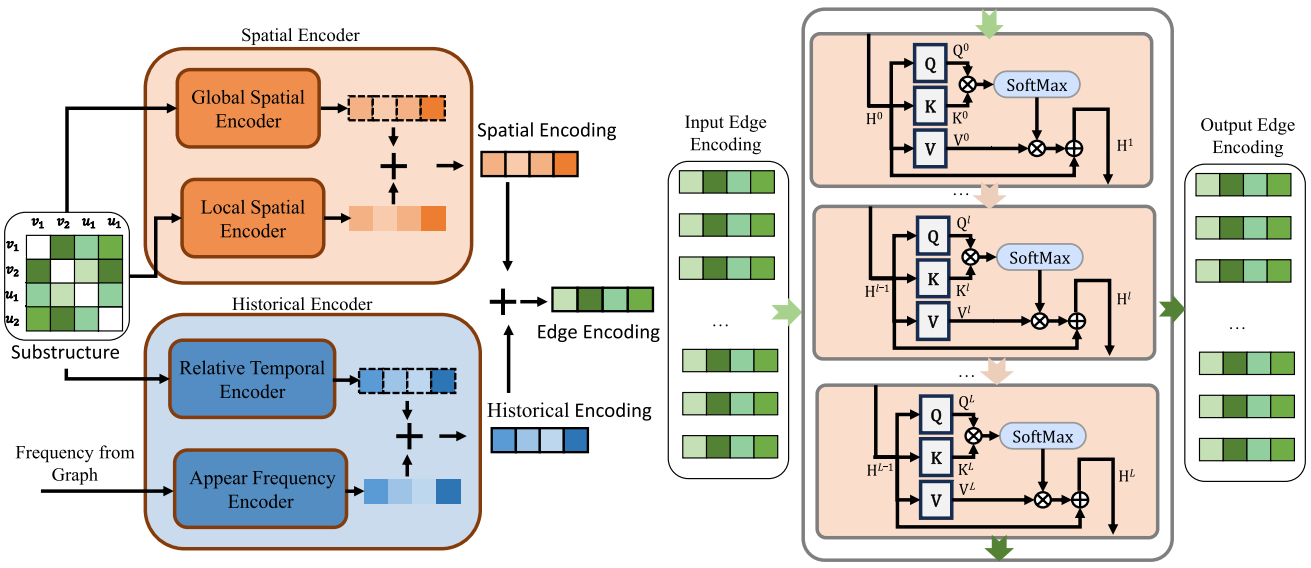
$$\mathbf{s}_{e_{i,j}} = \mathbf{s}_{v_1} + \mathbf{s}_{v_2} \tag{3}$$

We then turn our attention to dynamic graphs, initially capturing their temporal evolution through multiple timestamps. For a given target edge $e^t_{i,j} = (v^t_1, v^t_2)$ at timestamp t, the sequence of graphs is defined as $\mathbb{G}^t_\lambda = \{\mathcal{G}^{t-\lambda+1}, \cdots, \mathcal{G}^t\}$, where λ represents the size of the time window. A sliding window mechanism is subsequently employed, enabling the capture of dynamic evolving between timestamps $(t-\lambda+1)$ and t. For each substructure within the time frame, we compute the diffusion matrix and obtain a representation vector for the target edge. By sorting the elements within this connection vector, we can select k nodes. Integrating the source and destination nodes, the substructure node set of the i-th timestamp can be sampled as $\mathcal{S}^i(e^t_{i,j})$.

4.3 Spatial historical encoder

Our spatial historical encoder architecture is illustrated in Fig. 2a. It consists of four key components: the global spatial encoder, local spatial encoder, relative temporal encoder, and appear frequency encoder. These elements are designed to synergistically process and integrate different types of spatial and temporal data, thereby enhancing the overall encoding performance for our tasks. To capture the spatial information of the target edge, including the neighboring conditions and the degree of connectivity, we employ a spatial encoder to encode the obtained substructure, thereby extracting the spatial attributes of the target edge. To acquire the historical interaction information of the target edge, which includes the sequence of timestamps when the target edge appears and the frequency of its occurrence, we utilize a historical encoder. This encoder processes both the ubstructure and the occurrence data gathered from the original dataset to encode the historical interaction attributes of the target edge. Finally, we integrate the spatial and historical interaction attributes of the target edge to form an intermediate encoding for the target edge.

Specifically, for the substructure obtained from substructure sampling, the target edge's substructure at timestamp t is $\mathcal{S}(e^t_{i,j})$. In the spatial encoder, it is essential to capture both global and local spatial information. Inspired by Graphformers [26], the graph diffusion in substructure sampling provides a



(a) Spatial Historical Encoder (b) Transformer module

Fig. 2 Key structures within our AET

global view of each node's structural role. We rank nodes based on their diffusion values and use these rankings as a data source, embedding the target edge's spatial information within these rankings. The formula for this process is:

$$\mathbf{X}_{glb}(V) = g(\mathcal{R}(V \subset \mathbf{S}(e^t_{i,j}))) \tag{4}$$

where V is the node set based on the diffusion values, $g(\cdot), \mathcal{R}(\cdot)$ are the linear mapping function and ranking function.

Additionally, for the target edge, it is crucial to capture local spatial information. We calculate the shortest distance from the nodes in the substructure to the target edge for its encoding. If a node in the substructure is a source or target node of the target edge, the distance is set to zero. A single-layer learnable linear function is also employed to ensure dimensional alignment with the component that captures global spatial information:

$$\mathbf{X}_{loc}(V) = g(\bigcup_{i=t-\lambda+1}^{t} min(dist(v_i, v_1), dist(v_i, v_2))) \tag{5}$$

where $dist(\cdot), min(\cdot)$ are the relative distance computing function and minimum value function.

Then, by integrating both global and local spatial information, we derive the spatial attributes of the target edge. In the sistorical encoder, it is crucial to capture the relative order of timestamps for the target edge. However, merely encoding the relative sequence of timestamps is insufficient for capturing the edge's historical interactions effectively. Many edges reappear at different timestamps, and the practice of removing duplicate edges during the preprocessing stage significantly reduces the richness of information. At this stage, we can tally the frequency of occurrences of the target edge and encode this information simultaneously to prevent loss of data due to the removal of duplicates. For the relative order of the target edge's timestamps, we encode using the current timestamp in relation to the timestamps of the target edge. The resulting data is then dimensionally aligned using a single-layer learnable linear function. The frequency of occurrences of the target edge also requires dimensional alignment through a single-layer learnable linear function, which, when combined with the previous outputs, yields the target edge's historical interaction attributes, denoted as $\gamma(V)$. After that, we merge the target edge's spatial attributes and historical interaction attributes to form an intermediate encoding for the target edge, represented as:

$$\mathbf{X}_{temp}(V) = g(\gamma(V) + \bigcup_{i=t-\lambda+1}^{t} (\|\mathrm{t} - i\|)) \tag{6}$$

where $\|\cdot\|$ is the relative time computing function.

Finally, given a target edge $e^t_{i,j}$, we compute the encoding of the substructure node set for the given edge by summing the three encoding terms together:

$$\mathbf{X}(e^t_{i,j}) = \mathbf{X}_{glb}(V) + \mathbf{X}_{loc}(V) + \mathbf{X}_{temp}(V) \tag{7}$$

4.4 Transformer module

Our tansformer module is illustrated in Fig. 2b. To further enhance the quality of encoding, inspired by Graphformer, different tokens within the transformer can interact and learn from each other, thereby enriching the content of the tokens. In our approach, we input the edge encodings into the transformer in chronological order based on their timestamps. At this stage, each edge encoding not only contains spatial information but also historical interaction data. The transformer architecture is thus able to capture both spatial and temporal features simultaneously, facilitating the exchange of information between different edge encodings. Specifically:

$$\mathbf{Q}^{(l)} = \mathbf{H}^{(l-1)}\mathbf{W}^{(l)}_{Q}, \mathbf{K}^{(l)} = \mathbf{H}^{(l-1)}\mathbf{W}^{(l)}_{K}, \mathbf{V}^{(l)} = \mathbf{H}^{(l-1)}\mathbf{W}^{(l)}_{V} \tag{8}$$

where $\mathbf{W}^{(l)}_{Q}, \mathbf{K}^{(l)}_{K}, \mathbf{V}^{(l)}_{V} \in \mathbb{R}^{\lambda(k+2)\times d}$, are learnable parameter matrices of the l-th attention layer. Then a single attention layer can be written as:

$$\mathbf{H}^{(l)} = softmax\left(\frac{\mathbf{Q}^{(l)}\mathbf{K}^{(l)\top}}{\sqrt{d}}\right)\mathbf{V}^{(l)} = attention(\mathbf{H}^{l-1}) \tag{9}$$

where $\mathbf{H}^{(l)}$ is the output embedding of the l-th layer, d is the dimension for node embedding, and $\mathbf{Q}^{(l)}, \mathbf{K}^{(l)}, \mathbf{V}^{(l)} \in \mathbb{R}^{\lambda(k+2)\times d}$ are the query matrix, key matrix and value matrix. The output of the attention layer $\mathbf{H}^{(l)}$ is represented as the node embedding matrix from the transformer module, where each row corresponds to the embedding vector of a respective node, $\mathbf{H}^{(0)}$ is defined as the encoding matrix of the target edge $\mathbf{X}(e^t_{i,j})$, which is the output of spatial historical encoder.

4.5 Link prediction pre-training

As previously mentioned, due to the large number of parameters in the transformer, an appropriate pre-training strategy is essential to fully harness its capabilities. The link prediction pre-training task, which assesses whether two nodes are connected, is a common measure of the performance of GNNs. Given a pair of nodes, q and k, the model is trained to predict their connectivity based on the embedding of the edge. When fine-tuning on the target dataset, we use all other datasets as the training data for the link prediction pre-training. Samples generated by the negative sample generator serve as negative samples e_{neg}, while edges that are connected in the original graph are used as positive samples e_{pos}. The loss function used in this context is denoted as:

$$\mathcal{L} = -\sum_{i=1}^{m'} \log(1 - f(e_{pos,i}) + \log(f(e_{neg,i})) \tag{10}$$

After completing the link prediction pre-training, we save its weights and use these to initialize our Attribute Encoding Transformer (AET). We then reduce the learning rate to proceed with the anomaly detection tasks. This method not only leverages the powerful learning capabilities of the Transformer but also ensures that our model is finely tuned for the specific nuances of anomaly detection in dynamic graphs.

5 Experimental studies

This section provides a comprehensive overview of various experiments conducted on multiple real-world benchmark datasets.

5.1 Datasets

We evaluate our work on four real-world benchmark datasets of dynamics graphs. The details of the datasets are shown in Table 2. The *UCI Messages* dataset [16] is collected from an online community platform of students at the University of California, Irvine. Each node represents a user in the platform, and each edge is for a message between two users. The *Email-DNC* dataset [19] is network of emails in the 2016 Democratic National Committee email leak. Each node corresponds to a person. And each edge denotes an email communication between two persons. The *Digg* dataset [5] is a response notwork of digg, a social news site. Each node represents a website user, and each edge indicates that one user replies to another. The *Bitcoin-Alpha* [12] is a network connection dataset collected from a Bitcoin transaction platform www.btc-alpha.com. Nodes are users from the platform, and there is an edge appear when one user rates another user.

Table 2 The information of datasets

Dataset	Nodes	Edges	Avg. degree
UCI	1899	13,838	14.57
Email-DNC	1866	39,264	42.08
Digg	30,360	85,155	5.61
Bitcoin-Alpha	3777	24,173	12.80

5.2 Baselines

We compare our work with seven state-of-the-art baselines, the details of which were compared as follows:

Node2vec [9] use the Skip-gram technology to learn node embedding. It combines breadth-first traversal and depth-first traversal when generating random walks.

Spectral Clustering [23] treats all data nodes as vertices in a space, forming a graph wherein the objective is to partition this graph. The process involves optimizing the cut of the graph such that the sum of the edge weights between different subgraphs is minimized, while the sum of the edge weights within each subgraph is maximized. This method aims to achieve effective clustering by ensuring that the subgraphs are internally cohesive yet distinctly separated from each other.

DeepWalk [17] is a random walk-based method for graph embedding. It generates the random walks when given length starting from nodes and uses Skip-gram to learn the node embeddings.

NetWalk [28] first builds node embeddings based on random walks. Specifically, it utilizes a random alk-based approach to learn a unified embedding for each node using auto-encoder technology. The embedding representation will be update over time via an attention-based GRU structure, and then detects anomaly using the clustering on the node embeddings.

AddGraph [33] is an end-to-end dynamic graph anomaly detection model. It leverages GCN and GRU-attention to capturte strucral and temporal information respectively.

StrGNN [3] is an end-to-end dynamic graph anomaly detection approach. It leverges h-hop enclosing subgraphs as the network's input, and then used GCN and GRU to model the structural and temporal information for each edge.

TADDY [13] constructs an informative node encoding method, and uses a transformer model to successfully captures the coupled spatial-temporal information in dynamic graphs.

5.3 Experimental design

Evaluation Protocol AUC (Area Under Curve) is a metric for evaluating the performance of binary classification models. It takes account of the evaluation ability of both positive and negative classes, and is widely used in anomaly detection models.

Our proposed method is implemented using PyTorch 1.13.0 and CUDA 12.0. All experiments are conducted on a Linux server with 8*NVIDIA A40(40GB memory each) GPUs, with Intel Xeon Gold 6248R @ 3.00GHz * 2 and 1 T of RAM. The number of transformer layers is 2 with the embedding dim of 448 for all the datasets, and the number of attention heads is 6. *Setting of Link Prediction Pre-training*

The framework is pre-trained by Adam optimizer with a learning rate of 0.0005 with weight decay of 5e−5. In Pre-training stage, we train all datasets with 400 epochs for Link Prediction task then save the model's weight for initialization in anomaly detection task. *Setting of Anomaly Detection fine-tuning* The framework is trained by Adam optimizer with a learning rate of 0.0001 with weight decay of 2e−5. We train all datasets with 80 epochs. As baseline used, each dataset is divided into two subsets: the first 50% of timestamps is denoted as training set, while the latter 50% as test set. we consider three different anomaly proportions: 1%, 5%, and 10%, when injecting the anomalous data into the test set in the preprocessing stage.

5.4 Results analysis

In this section, we first report and analyze the performance achieved on various attribute-less dynamic graph datasets, with all results summarized in Table 3.

- Our model consistently outperforms all baseline models across four different real-world datasets. Notably, on the UCI dataset, our model significantly surpasses other methods, achieving an average performance improvement of approximately 3% across three different settings. Particularly when the anomaly proportion is 1%, our model exhibits nearly a 6% increase in performance.
- Compared to other baselines, our model is less sensitive to variations in anomaly proportions. As indicated in the table, the performance of other models is greatly influenced by the anomaly ratio, whereas our model maintains robust performance regardless of the anomaly proportion. This robustness is likely due to our model's superior capability in extracting high-quality edge embeddings, which significantly broadens its applicability.
- When compared with graph embedding methods such as NetWalk, AddGraph, and StrGNN, our results consistently hold an advantage. We attribute this to our effective utilization of the transformer structure's capability for long-distance modeling, allowing for the exchange of information across different timestamps. Moreover, compared to methods like Taddy that also use the transformer architecture, our approach remains superior. We speculate this is partly due to our effective exploitation of the transformer's potential through an appropriate pre-training strategy and partly because our historical encoder adeptly captures temporal and historical interaction information. This enriches the node encoding, thereby enhancing anomaly detection performance.

Secondly, we evaluated the impact of varying training data proportions on the performance of AET. The training ratios tested ranged from 10% to 60%, with other parameters set to default values. We utilized the Bitcoin-Alpha dataset, which contains 10% anomalies, as depicted in Fig. 3. As

Table 3 Performance comparison of various methods

Method	UCI			Digg		
	1%	5%	10%	1%	5%	10%
NODE2VEC	0.7371	0.7433	0.6960	0.7364	0.7081	0.6508
SPECTRAL	0.6324	0.6104	0.5794	0.5949	0.5823	0.5591
DEEPWALK	0.7514	0.7391	0.6979	0.7080	0.6881	0.6396
NETWALK	0.7758	0.7647	0.7226	0.7563	0.7176	0.6837
ADDGRAPH	0.8083	0.8090	0.7688	0.8341	0.8470	0.8369
STRGNN	0.8179	0.8252	0.7959	0.8162	0.8254	0.8272
TADDY	0.8912	0.8398	0.8370	0.8617	0.8545	0.8440
AET (ours)	0.8966	0.8679	0.8893	0.8755	0.8640	0.8725
Method	Email-DNC			Bitcoin-Alpha		
	1%	5%	10%	1%	5%	10%
NODE2VEC	0.7391	0.7284	0.7103	0.6910	0.6802	0.6785
SPECTRAL	0.8096	0.7857	0.7759	0.7401	0.7275	0.7167
DEEPWALK	0.7481	0.7303	0.7197	0.6985	0.6874	0.6793
NETWALK	0.8105	0.8371	0.8305	0.8385	0.8357	0.8350
ADDGRAPH	0.8393	0.8627	0.8773	0.8665	0.8403	0.8498
STRGNN	0.8775	0.9103	0.9080	0.8574	0.8667	0.8627
TADDY	0.9348	0.9257	0.9210	0.9451	0.9341	0.9423
AET (ours)	0.9537	0.9582	0.9498	0.9485	0.9472	0.9624

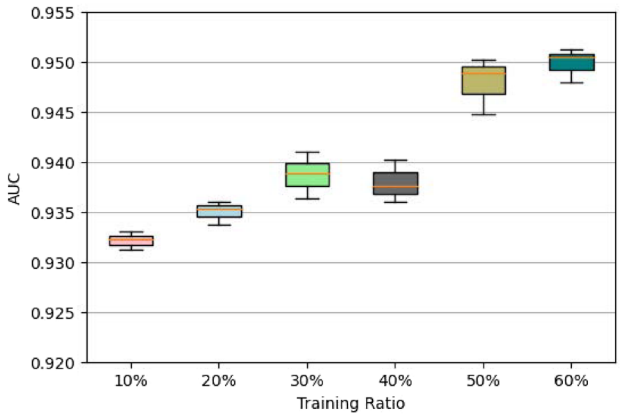


Fig. 3 AUC results on Bitcoin-Alpha with different training ratios

the training ratio increases, the AUC value steadily rises, indicating that more training data can provide the model with more effective information. Even in cases of insufficient training data, our model is capable of learning information-rich representations. These results demonstrate that our framework can effectively detect anomalies in dynamic graphs, regardless of the sufficiency of training data.

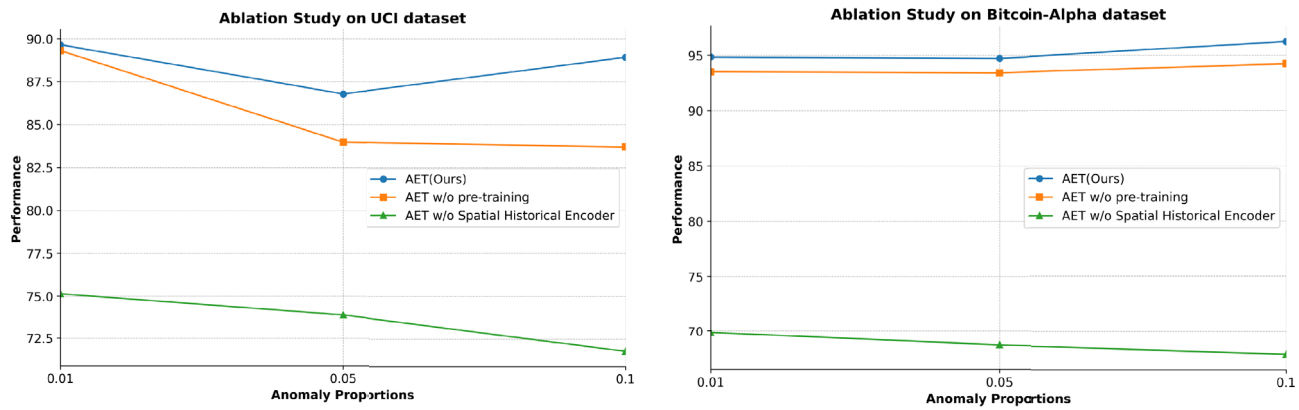
5.5 Ablation study

To analyze the impact of spatial historical encoder and link prediction pre-training methods on the overall performance, we carry out an ablation study of the proposed AET framework. Specifically, we evaluate several variations of the node encoding: without the pre-training model (w/o pre-training.), and without the Spatial Historical Encoder (w/o Spatial Historical Encoder.). In each case, the corresponding component is removed, while the remaining method is retained.

As illustrated in Fig. 4, the results clearly demonstrate that, on one hand, after pre-training, our model becomes less sensitive to variations in anomaly ratios. This improvement is associated with the enriched graph structure exposure during the pre-training phase, which broadens the initial knowledge base of the model. On the other hand, there is a significant difference in performance between models with and without the spatial historical encoder. This disparity can be attributed to the challenges of obtaining attributes for edges in attribute-less graphs. Without appropriate encoding strategies and substructure sampling, models struggle to acquire useful attributes, hindering their ability to learn effective encodings.

6 Conclusion

In this paper, we introduce a novel encoding paradigm for edges in attribute-less graphs, termed the Attribute Encoding Transformer, which comprehensively incorporates both spatial and historical interaction information of target edges. Additionally, we propose a pre-training methodology, link prediction pretrain, designed to optimize the performance of the transformer architecture. Experimental evaluations on multiple real-world datasets demonstrate that our Attribute Encoding Transformer framework efficiently detects anomalies in dynamic graphs and outperforms existing methods. Overall, our AET successfully addresses the challenges of encoding attribute-less graphs, thereby enhancing support for downstream tasks such as anomaly detection.



(a) Ablation Study on UCI dataset

(b) Ablation Study on Bitcoin-Alpha

Fig. 4 Ablation study

Author Contributions SW had the idea for the article. All authors contributed to the study conception and design. Material preparation, literature search, data collection and analysis were performed by SW and HH. The first draft of the manuscript was written by SW and all authors commented on previous versions of the manuscript. All authors read and approved the final manuscript.

Data availability No datasets were generated or analysed during the current study.

Declarations

Conflict of interest The authors declare no conflict of interest.

References

1. Aggarwal CC, Zhao Y, Philip SY (2011) Outlier detection in graph streams. In: 2011 IEEE 27th international conference on data engineering. IEEE, pp 399–409
2. Akoglu L, Tong H, Koutra D (2015) Graph based anomaly detection and description: a survey. Data Min Knowl Disc 29:626–688
3. Cai L, Chen Z, Luo C, Gui J, Ni J, Li D, Chen H (2021) Structural temporal graph neural networks for anomaly detection in dynamic graphs. In: Proceedings of the 30th ACM international conference on information & knowledge management, pp 3747–3756
4. Chen J, Pareja A, Domeniconi G, Ma T, Suzumura T, Kaler T, Schardl TB, Leiserson CE (2022) Evolving graph convolutional networks for dynamic graphs Dec 27 , uS Patent 11,537,852
5. De Choudhury M, Sundaram H, John A, Seligmann DD (2009) Social synchrony: predicting mimicry of user actions in online social media. In: 2009 International conference on computational science and engineering, vol 4. IEEE, pp 151–158
6. Devlin J, Chang MW, Lee K, Toutanova K (2018) Bert: pre-training of deep bidirectional transformers for language understanding. arXiv preprint arXiv:1810.04805
7. Dou Y, Liu Z, Sun L, Deng Y, Peng H, Yu PS (2020) Enhancing graph neural network-based fraud detectors against camouflaged fraudsters. In: Proceedings of the 29th ACM international conference on information & knowledge management, pp 315–324
8. Fan W, Ma Y, Li Q, He Y, Zhao E, Tang J, Yin D (2019) Graph neural networks for social recommendation. In: The world wide web conference, pp 417–426
9. Grover A, Leskovec J (2016) node2vec: Scalable feature learning for networks. In: Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining, pp 855–864
10. Hao H, Wang S, Ben H, Hao Y, Wang Y, Wang W (2024) Hierarchical space-time attention for micro-expression recognition. arXiv preprint arXiv:2405.03202
11. Kanakia A, Shen Z, Eide D, Wang K (2019) A scalable hybrid research paper recommender system for Microsoft academic. In: The world wide web conference, pp 2893–2899
12. Kumar S, Spezzano F, Subrahmanian V, Faloutsos C (2016) Edge weight prediction in weighted signed networks. In: 2016 IEEE 16th international conference on data mining (ICDM). IEEE, pp 221–230
13. Liu Y, Pan S, Wang YG, Xiong F, Wang L, Chen Q, Lee VC (2021) Anomaly detection in dynamic graphs via transformer. IEEE Trans Knowl Data Eng 35(12):12081–12094
14. Ma X, Wu J, Xue S, Yang J, Zhou C, Sheng QZ, Xiong H, Akoglu L (2021) A comprehensive survey on graph anomaly detection with deep learning. IEEE Trans Knowl Data Eng 35(12):12012–12038
15. Min E, Rong Y, Xu T, Bian Y, Luo D, Lin K, Huang J, Ananiadou S, Zhao P (2022) Neighbour interaction based click-through rate prediction via graph-masked transformer. In: Proceedings of the 45th international ACM SIGIR conference on research and development in information retrieval, pp 353–362
16. Opsahl T, Panzarasa P (2009) Clustering in weighted networks. Soc Netw 31(2):155–163
17. Perozzi B, Al-Rfou R, Skiena S (2014) Deepwalk: online learning of social representations. In: Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining, pp 701–710
18. Ranshous S, Harenberg S, Sharma K, Samatova NF (2016) A scalable approach for outlier detection in edge streams using sketch-based approximations. In: Proceedings of the 2016 SIAM international conference on data mining. SIAM, pp 189–197
19. Rossi R, Ahmed N (2015) The network data repository with interactive graph analytics and visualization. In: Proceedings of the AAAI conference on artificial intelligence, vol 29
20. Rossi E, Chamberlain B, Frasca F, Eynard D, Monti F, Bronstein M (2020) Temporal graph networks for deep learning on dynamic graphs. arXiv preprint arXiv:2006.10637
21. Sankar A, Wu Y, Gou L, Zhang W, Yang H (2020) Dysat: Deep neural representation learning on dynamic graphs via self-attention networks. In: Proceedings of the 13th international conference on web search and data mining, pp 519–527
22. Vaswani A, Shazeer N, Parmar N, Uszkoreit J, Jones L, Gomez AN, Kaiser Ł, Polosukhin I (2017) Attention is all you need. Adv Neural Inf Process Syst 30
23. von Luxburg U A tutorial on spectral clustering. statistics and computing. Data Structures and Algorithms (cs. DS); Machine Learning, pp 395–416
24. Wu Y, Fang Y, Liao L (2024) On the feasibility of simple transformer for dynamic graph modeling. arXiv preprint arXiv:2401.14009
25. Yang C, Zhou L, Wen H, Zhou Z, Wu Y (2020) H-vgrae: a hierarchical stochastic spatial-temporal embedding method for robust anomaly detection in dynamic networks. arXiv preprint arXiv:2007.06903
26. Yang J, Liu Z, Xiao S, Li C, Lian D, Agrawal S, Singh A, Sun G, Xie X (2021) Graphformers: Gnn-nested transformers for representation learning on textual graph. Adv Neural Inf Process Syst 34:28798–28810
27. Ying C, Cai T, Luo S, Zheng S, Ke G, He D, Shen Y, Liu TY (2021) Do transformers really perform badly for graph representation? Adv Neural Inf Process Syst 34:28877–28888
28. Yu W, Cheng W, Aggarwal CC, Zhang K, Chen H, Wang W (2018) Netwalk: a flexible deep embedding approach for anomaly detection in dynamic networks. In: Proceedings of the 24th ACM SIGKDD international conference on knowledge discovery & data mining, pp 2672–2681
29. Yu L, Sun L, Du B, Lv W (2023) Towards better dynamic graph learning: new architecture and unified library. Adv Neural Inf Process Syst 36:67686–67700
30. Yu X, Zhou C, Fang Y, Zhang X (2023) Multigprompt for multi-task pre-training and prompting on graphs. arXiv preprint arXiv:2312.03731
31. Yu X, Fang Y, Liu Z, Zhang X (2024) Hgprompt: bridging homogeneous and heterogeneous graphs for few-shot prompt learning. In: Proceedings of the AAAI conference on artificial intelligence, vol 38, pp 16578–16586

32. Zhang J, Zhang H, Xia C, Sun L (2020) Graph-bert: only attention is needed for learning graph representations. arXiv preprint arXiv:2001.05140
33. Zheng L, Li Z, Li J, Li Z, Gao J (2019) Addgraph: anomaly detection in dynamic graph using attention-based temporal gcn. In: IJCAI. 3, 7