Anomaly Detection in Dynamic Graphs via Transformer

Yixin Liu, Shirui Pan, Yu Guang Wang, Fei Xiong, Liang Wang, Qingfeng Chen, and Vincent CS Lee, *Senior Member, IEEE*

Abstract—Detecting anomalies for dynamic graphs has drawn increasing attention due to their wide applications in social networks, e-commerce, and cybersecurity. Recent deep learning-based approaches have shown promising results over shallow methods. However, they fail to address two core challenges of anomaly detection in dynamic graphs: the lack of informative encoding for unattributed nodes and the difficulty of learning discriminate knowledge from coupled spatial-temporal dynamic graphs. To overcome these challenges, in this paper, we present a novel Transformer-based Anomaly Detection framework for DYnamic graphs (**TADDY**). Our framework constructs a comprehensive node encoding strategy to better represent each node's structural and temporal roles in an evolving graphs stream. Meanwhile, TADDY captures informative representation from dynamic graphs with coupled spatial-temporal patterns via a dynamic graph transformer model. The extensive experimental results demonstrate that our proposed TADDY framework outperforms the state-of-the-art methods by a large margin on six real-world datasets.

Index Terms—Anomaly detection, dynamic graphs, transformer

1 Introduction

In recent years, graphs have attracted a surge of research attention with the development of networked applications in social networks [1], human knowledge networks [2], business networks [3] and cybersecurity [4]. However, the bulk of the existing researches focus on static graphs [5], [6], yet the real-world graph data often evolves over time [7], [8]. Taking social networks as an example, there are always fresh persons who enroll in the community every month, and the relation between individuals is changing over time. To model and analyze graphs where nodes and edges change over time, mining dynamic graphs gains increasing popularity in the community of graph analysis.

- *Yixin Liu, Shirui Pan, and Vincent CS Lee are with the Department of Data Science and AI, Faculty of IT, Monash University, Clayton, VIC 3800, Australia. E-mail: {yixin.liu, shirui.pan, Vincent.CS.Lee}@monash.edu.*
- *Yu Guang Wang is with the Institute of Natural Sciences, School of Mathematical Sciences, Shanghai Jiao Tong University, Shanghai 200240, China, and also with the Max Planck Institute for Mathematics in Sciences, Mathematics Machine Learning Group, 04103 Leipzig, Germany. E-mail: yuguang.wang@sjtu.edu.cn.*
- *Fei Xiong is with the Laboratory of Communication and Information Systems, Beijing Municipal Commission of Education, Beijing Jiaotong University, Beijing 100044, China. E-mail: xiongf@bjtu.edu.cn.*
- *Liang Wang is with the School of Computer Science, Northwestern Polytechnical University, Xi'an, Shaanxi 10072, China. E-mail: liangwang@nwpu.edu.cn.*
- *Qingfeng Chen is with the School of Computer, Electronic and Information, Guangxi University, Nanning 530004, China. E-mail: qingfeng@gxu.edu.cn.*

Manuscript received 30 April 2021; revised 21 August 2021; accepted 27 October 2021. Date of publication 2 November 2021; date of current version 8 November 2023.
This work was supported in part by the Australian Research Council (ARC) under a Future Fellowship under Grant FT210100097 and in part by the National Natural Science Foundation of China Project under Grant 61963004. (Corresponding author: Shirui Pan.)
Recommended for acceptance by A. SI.
Digital Object Identifier no. 10.1109/TKDE.2021.3124061

Among various analysis problems for dynamic graphs, detecting the anomalous edges in an evolving graph stream is a critical task [9], [10]. Considering a user-item network in the e-commerce scenario, the attackers tend to make fake purchase orders to increase the influence of certain goods illegally. It is of great significance to detect such fake orders to maintain a fair trading environment.

Detecting anomalies in dynamic graphs, however, is not a trivial task since there are two challenges in dynamic graph learning. *Challenge 1* is the lack of raw attribute information in most dynamic graphs. Due to the explosive demand for data volume of time-evolving attributes or the inaccessible attributes caused by privacy issues, it is hard to construct attribute information to represent each node from the mainstream raw dynamic graph datasets. To fill the gap, an effective encoding method that constructs artificial features to represent evolving nodes is required. *Challenge 2* is the difficulty of learning discriminative knowledge from dynamic graphs where spatial (structural) information and temporal information are coupled. Fig. 1 provides a toy example to illustrate how coupled information affects the detection of edge abnormality. The green edge tends to be normal since there are close structural communications between their neighborhoods in the previous timestamps. The red edge, on the contrary, is an anomalous edge with a high probability because the two red nodes always keep a distance from each other in the former snapshots. The point is that both structural (i.e., shared neighborhoods) and temporal (i.e., previous interaction) factors should be considered simultaneously when making decisions, raising the challenge in understanding such coupled information.

Aiming to detect anomalies in dynamic graphs, various types of approaches are proposed in the recent decade. The shallow methods like GOutlier [11] and CM-Sketch [12] utilize shallow learning mechanisms (e.g., structural

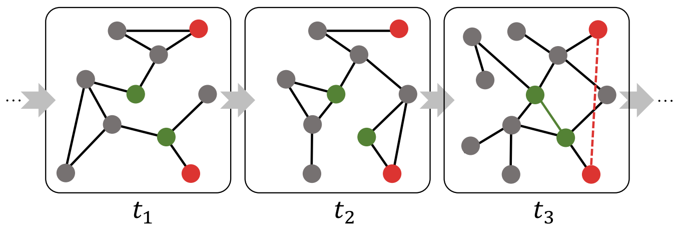


Fig. 1. A toy example to illustrate how the coupled information affects the detection of edges' legality. The three graphs are a fragment from a dynamic graph stream at a sequential timeline $\{t_1, t_2, t_3\}$. The solid green line represents a normal edge at t_3, while the red dash line indicates an anomalous edge. We highlight the corresponding nodes in the previous timestamps with colors.

connectivity model or historical behavior analysis) to detect anomalies. However, empirical experiments show that these methods suffer from limited performance when detecting anomalous edges in large and complex dynamic graphs [9]. Very recently, as a novel branch, deep learning-based methods, have shown to be a powerful solution for dynamic graph learning. For example, NetWalk [9] leverages dynamic deep graph embedding technique with a clustering-based detector to detect anomalies; AddGraph [10], StrGNN [13] and H-VGRAE [14] further exploit end-to-end deep neural network models to solve the problem.

Despite their improved performance, the existing deep learning-based methods fail to address the aforementioned challenges very well. Specifically, when facing the lack of raw node attributes, they do not create informative node encodings to represent the nodes' properties. The one-hot identity features in [9], [14] and the random initialized features in [10] cannot express any structural or temporal property of each node. The distance-based node labeling strategy in [13] only considers local structural information, limiting its expressive power. Furthermore, most of them use two individual network modules to extract spatial and temporal features, resulting in their insufficient capability to capture the coupled information. For instance, in AddGraph [10] and StrGNN [13], Graph Convolutional Networks (GCNs) are employed to acquire spatial knowledge, with following Gated Recurrent Units (GRUs) capturing temporal information. The isolated processing of two types of information results in missing the coupled spatial-temporal features and further leads to a sub-optimal solution.

Aiming to resolve these challenges, in this paper, we propose a novel **T**ransformer-based **A**nomaly **D**etection framework for **DY**namic graph (**TADDY** for abbreviation). Our theme is to construct a node encoding to cover sufficient spatial and temporal knowledge and leverage a sole transformer model to capture the coupled spatial-temporal information. More specifically, to overcome *Challenge 1*, we carefully design a comprehensive node encoding composed of three functional terms to distill global spatial, local spatial, and temporal information. Learnable mapping functions are integrated into the node encoding, which helps the framework automatically extract informative encoding in an end-to-end manner. To address *Challenge 2*, we develop a dynamic graph transformer model to simultaneously learn spatial and temporal knowledge. An edge-based substructure sampling is performed to capture contextual information crossing time as the input of the transformer model. Then, the coupled spatial-temporal information is extracted by the attention mechanism crossing structure and time. To sum up, the main contributions of this paper are:

- We propose an end-to-end transformer-based learning framework, TADDY, for anomaly detection on dynamic graphs. This is the first transformer-based method for dynamic graph learning and graph anomaly detection to the best of our knowledge.
- We design a comprehensive encoding method for nodes in dynamic graphs. The proposed node encoding integrates various knowledge, including global spatial, local spatial and temporal information.
- We present a dynamic graph transformer model which aggregates spatial and temporal knowledge simultaneously. A novel edge-based substructure sampling strategy is leveraged to provide sufficient receipt fields for the learning model.
- We evaluate the effectiveness of TADDY on six benchmark datasets. The extensive experiments demonstrate that our method delivers state-of-the-art performance.

We organize the rest of this paper as follows. The related works are reviewed in Section 2. We describe the problem definition in Section 3. In Section 4 the overall pipeline and each component of our framework are introduced. The experimental results are demonstrated in Section 5. Finally, we conclude the contributions and future works of this work in Section 6.

2 Related Work

This section briefly reviews existing anomaly detection methods for dynamic graphs and transformers.

2.1 Anomaly Detection in Dynamic Graphs

Anomaly detection in dynamic graphs attracts considerable interest by the research community [15], for which, many methods have been proposed in recent years. For example, GOutlier [11] employs a structural connectivity model to detect outliers in graph streams and builds dynamic network partition to maintain the connectivity behavior model. CAD [16] detects node relationships by tracking a measure that combines information regarding changes in graph structure and in edge weights. CM-Sketch [12] considers both the local structural information and historical behavior to discriminate the edge's anomalous property. StreamSpot [17] is a clustering-based approach that utilizes a novel similarity function for heterogeneous graphs property comparison and leverages a centroid-based clustering method to model the behaviors of graph stream. SpotLight [18] uses a randomized sketching technique to guarantee a large mapped distance between anomalous and normal instances in the sketch space. Since these approaches leverage the shallow mechanisms to detect the anomalous edges, we categorize them into shallow learning-based methods.

Another branch of approach employs deep learning technique to capture anomalous data in dynamic graphs, which is denoted as the category of deep learning-based method. NetWalk [9] leverages a random walk-based encoder to generate node embeddings with clique embedding objective and then models the network evolving via dynamic updating reservoirs. Finally, a dynamic clustering-based anomaly detector is employed to score the abnormality of each edge. AddGraph [10] further constructs an end-to-end neural network model to capture dynamic graphs' spatial and temporal patterns. A GCN [19] is served as a structural features extractor, and a GRU-attention module is designed to combine short-term and long-term dynamic evolving. StrGNN [13] extracts the h-hop enclosing subgraph of edges and leverages stacked GCN [19] and GRU to capture the spatial and temporal information. The learning model is trained in an end-to-end way with negative sampling from "context-dependent" noise distribution. H-VGRAE [14] builds a hierarchical model by combining variational graph autoencoder and recurrent neural network. To detect anomalous edges, the edge reconstruction probability is used to measure the abnormality.

Our proposed TADDY framework can be categorized into the deep learning-based methods but has two main differences compared to the existing approaches mentioned above. Most of the above approaches employ different network modules to separately extract spatial and temporal features, while TADDY uses a transformer network to model spatial and temporal information simultaneously.

Second, these methods consider naive node encoding from unattributed dynamic graphs as the network input, which may fail to provide sufficient information for the downstream neural network. In contrast to them, TADDY constructs a comprehensive node encoding that includes both spatial and temporal information.

2.2 Transformers

Transformers are a family of neural networks solely based on attention mechanisms to learn representative embedding for various data. The Transformer model is first proposed in [20], which focuses on the machine translation tasks in natural language processing (NLP). BERT [21] further applies transformers to multiple deep language understanding tasks by introducing the pre-training technique. Following Transformer and BERT, a large number of variant works are presented and reach state-of-the-art results on various NLP tasks [22], [23], [24]. Very recently, the transformer model is extended to the field of computer vision (CV) [25]. For instance, DETR [26] first leverages transformers on the object detection task. ViT [27] splits an image into multiple patches and uses a pure transformer model to learn the representation for image classification directly. SETR [28] utilizes a ViT-like encoder for feature extraction and adopts a multi-level feature aggregation module for pixel-wise image segmentation. For further details about transformers on NLP and CV please see related surveys [29], [30].

Some recent works also introduce transformers to the field of graph machine learning. GTN [31] is performed on heterogeneous graphs with transformers by meta-path-based relation learning. HGT [32] is a transformer model for the representation learning on web-scale heterogeneous graphs, which reaches state-of-the-art results on various downstream tasks. GROVER [33] integrates the message passing mechanism into the transformer architecture to learn representation for molecule graph data. Graph-BERT [34] constructs a BERT-like network model for static graph learning and introduces various well-designed tasks for self-supervised model pre-training [35].

Our proposed framework introduces transformers as our backbone neural network model due to its powerful expressive capability. Differently, we extend transformers to dynamic graphs, which is a more complex learning scenario where both structural and temporal features should be considered. By comparison, most of the existing methods focus on static graphs.

3 Problem Definition

In this paper, we model a dynamic graph as a graph stream represented by a series of discrete snapshots. The definition of dynamic graphs is given as follows:

Definition 1. *Considering a dynamic graph with a maximum timestamp of T, a graph steam is represented by $\mathbb{G} = \{\mathcal{G}^t\}_{t=1}^T$, where each $\mathcal{G}^t = (\mathcal{V}^t, \mathcal{E}^t)$ is the snapshot at timestamp t, $\mathcal{V}^t$ is the node set at timestamp t, and $\mathcal{E}^t$ is the edge set at timestamp t. An edge $e_{i,j}^t = (v_i^t, v_j^t) \in \mathcal{E}^t$ indicates that there is a connection between node v_i^t and v_j^t at the timestamp t, where $v_i^t, v_j^t \in \mathcal{V}^t$. We use $n^t = |\mathcal{V}^t|$ and $m^t = |\mathcal{E}^t|$ to denote the number of nodes and edges at timestamp t respectively. A binary adjacency matrix $\mathbf{A}^t \in \mathbb{R}^{n^t \times n^t}$ is employed to denote $\mathcal{G}^t$, where $\mathbf{A}_{i,j}^t = 1$ if there is a link between nodes v_i and v_j at timestamp t, otherwise $\mathbf{A}_{i,j}^t = 0$.*

The goal of this paper is to detect the anomalous edges in each timestamp. According to the aforementioned notation, we formalize the anomaly detection in dynamic graphs as a scoring problem:

Definition 2. Anomaly Detection in Dynamic Graphs. *Given a dynamic graph $\mathbb{G} = \{\mathcal{G}^t\}_{t=1}^T$ where each $\mathcal{G}^t = (\mathcal{V}^t, \mathcal{E}^t)$, for each $e_{i,j}^t \in \mathcal{E}^t$, the goal of anomaly detection is to produce the anomaly score $f(e_{i,j}^t)$, where $f(\cdot)$ is a learnable anomaly score function. The anomaly score indicates the abnormality degree of the edge, where a larger score $f(e_{i,j}^t)$ shows a higher anomalous probability of $e_{i,j}^t$.*

Following the previous works [9], [10], [13], we consider an unsupervised setting for anomaly detection in dynamic graphs. Specifically, in the training phase, no labeled data for anomalies is given, but we assume that all edges in the training set are normal. The binary labels of abnormality are given in the testing phase to evaluate the performance of algorithms. Concretely, a label $y_{e_{i,j}^t} = 1$ indicates that edge $e_{i,j}^t$ is an anomalous edge, and $y_{e_{i,j}^t} = 0$ indicates that $e_{i,j}^t$ is normal. Note that the distribution of normal and anomalous edges is often imbalanced, which means the number of normal edges is much larger than anomalous edges.

For the convenience of readers, the notation used in this paper is summarized in Table 1.

4 Methodology

In this section, we introduce the general framework of TADDY. The overview of our proposed framework is illustrated in Fig. 2. On the highest level, TADDY consists of four components, namely *edge-based substructure sampling*, *spatial-temporal node encoding*, *dynamic graph transformer*, and *discriminative anomaly detector*. The framework is trained in an end-to-end manner, indicating that the anomaly scores are output

TABLE 1
Commonly Used Notation With Explanations

Notation	Explanation
$\mathbb{G} = \{\mathcal{G}^t\}_{t=1}^T$	A graph steam with a maximum timestamp of T.
$\mathcal{G}^t = (\mathcal{V}^t, \mathcal{E}^t)$	The snapshot graph at timestamp t.
$\mathcal{V}^t$	The node set at timestamp t.
$\mathcal{E}^t$	The edge set at timestamp t.
$v_i^t \in \mathcal{V}^t$	A node with index i at the timestamp t.
$e_{i,j}^t = (v_i^t, v_j^t) \in \mathcal{E}^t$	An edge between v_i^t and v_j^t at the timestamp 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.
$\mathbb{G}_\tau^t = \{\mathcal{G}^{t-\tau+1}, \cdots, \mathcal{G}^t\}$	The sequence of graphs with timestamp t as the end and τ as the window size (length).
$\mathcal{S}(e_{\text{tgt}}^t)$	The substructure node set of target edge e_{tgt}^t.
$\mathbf{x}_{diff}(v_j^i)$	The diffusion-based spatial encoding of node v_j^i.
$\mathbf{x}_{dist}(v_j^i)$	The distance-based spatial encoding of node v_j^i.
$\mathbf{x}_{temp}(v_j^i)$	The relative temporal encoding of node v_j^i.
$\mathbf{x}(v_j^i)$	The fused encoding of node v_j^i.
$\mathbf{X}(e_{\text{tgt}}^t)$	The encoding matrix of target edge e_{tgt}^t.
$\mathbf{H}^{(l)}$	The output embedding of the l-th layers of Transformer.
$\mathbf{Q}^{(l)}$	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.
$\mathbf{z}(e_{\text{tgt}}^t)$	The embedding of target edge e_{tgt}^t.
$\mathbf{W}_Q^{(l)}, \mathbf{W}_K^{(l)}, \mathbf{W}_V^{(l)}$	The learnable parameters of Transformer.
$e_{\text{pos},i} \in \mathcal{E}^t$	The i-th positive edge from $\mathcal{E}^t$.
$e_{\text{neg},i} \in \mathcal{E}_n^t \sim P_n(\mathcal{E}^t)$	The i-th negative edge by negative sampling $P_n(\mathcal{E}^t)$.
$\mathbf{w}_S, b_S$	The learnable parameters of Anomaly Detector.
k	The number of contextual nodes.
τ	The size of time window.
d_{enc}	The dimension of encoding.
d_{emb}	The dimension of embedding.
L	The number of layers of Transformer.

and learned directly. At first, to capture the spatial-temporal contexts of each target edge, we perform *edge-based substructure sampling* to acquire the target nodes and the contextual nodes in multiple timestamps. Then, the *spatial-temporal node encoding* generates the node encodings as the input of the transformer model. Both spatial and temporal information for each node are integrated into a fixed-length encoding. After that, the *dynamic graph transformer* extracts the spatial-temporal knowledge of edges via a sole transformer model composed of a transformer module and a pooling module. Finally, in the *discriminative anomaly detector*, we perform a negative sampling to generate pseudo negative edges, and an edge scoring module trained by binary cross-entropy loss is employed to calculate the output anomaly scores. In Sections 4.1, 4.2, 4.3, and 4.4, we concretely introduce the four main components of TADDY framework. In Section 4.5, we analyze the time complexity of our proposed framework.

4.1 Edge-Based Substructure Sampling

As is noticed in previous works [13], [36], anomalies often occur in local substructures of graphs, indicating that we should zoom our receptive field to a suitable local scale. Therefore, instead of working on a complete dynamic graph, we first sample substructures as the data elements of our anomaly detection framework. Since we focus on detecting anomalous edges, we perform an edge-based sampling: each edge in dynamic graphs is viewed as the center of the sampled substructure and is denoted as a *target edge*. For a given target edge, we denote the source node and destination node as *target nodes*.

In addition to the target nodes, it is necessary to include other neighboring nodes in the sampled substructure. In this paper, we denote these neighboring nodes as *contextual nodes*. To acquire contextual information, a natural question arises here is: *how to efficiently sample contextual nodes from a given target edge in a dynamic graph?* We first consider the sampling problem in a single snapshot which can be regarded as a static graph. A naive solution is to extract the h-hop neighbors of target nodes as contexts. However, this strategy has several drawbacks. First, with h-hop neighbors sampling, the imbalanced distribution of node degrees in real-world datasets would lead to a performance decline and low efficiency. For example, the average degree of UCI Messages dataset is 14.47, while its maximum degree is 255. For those popular nodes with high degrees, the numbers of h-hop neighbors would be explosive, resulting in the noisy information in sampled contexts and damaging the running efficiency. Second, sampling h-hop neighbors ignores the different roles and importance of nodes in the substructure. Considering two target nodes with both shared and exclusive neighbors, it is obvious that the shared neighbors contribute more to detecting the target edges. However, this simple strategy just views the shared and exclusive neighbors equally when sampling contextual nodes.

To address the aforementioned limitations, in this work, we borrow the graph diffusion technique [37], [38] to sample a fixed-size and importance-aware contextual node set for each target edge. With graph diffusion, a global view of graph structure is acquired, and then we can quantify the importance of each node for a given target node/edge. Formally, given an adjacency matrix of a static graph $\mathbf{A} \in \mathbb{R}^{n\times n}$, we define graph diffusion $\mathbf{S} \in \mathbb{R}^{n\times n}$ by

$$\mathbf{S} = \sum_{k=0}^{\infty} \theta_k \mathbf{T}^k, \tag{1}$$

where $\mathbf{T} \in \mathbb{R}^{n\times n}$ is the generalized transition matrix and θ_k is the weighting coefficient which determines the ratio of global-local information. To guarantee convergence, some stricter conditions are considered which requires that $\sum_{k=0}^{\infty} \theta_k = 1, \theta_k \in [0,1]$ and the eigenvalues λ_i of $\mathbf{T}$ are bounded by $\lambda_i \in [0,1]$. By applying specific definitions of $\mathbf{T}$ and θ, different instantiations of graph diffusion can be computed. For instance, Personalized PageRank (PPR) [39] and the heat kernel [40] are two popular examples of diffusion. Concretely, PPR chooses $\mathbf{T} = \mathbf{A}\mathbf{D}^{-1}$ and $\theta_k = \alpha(1-\alpha)^k$, where $\mathbf{D} \in \mathbb{R}^{n\times n}$ is the diagonal degree matrix and $\alpha \in (0,1)$ is the teleport probability. The heat kernel considers $\mathbf{T} = \mathbf{A}\mathbf{D}^{-1}$ and $\theta_k = e^{-\beta}\beta^k/k!$, where β is the diffusion time. To avoid multiple steps of iteration, the solutions to PPR and heat kernel can be formulated as:

$$\mathbf{S}^{\text{PPR}} = \alpha\Big(\mathbf{I}_n - (1-\alpha)\mathbf{D}^{-1/2}\mathbf{A}\mathbf{D}^{-1/2}\Big)^{-1}, \tag{2}$$

$$\mathbf{S}^{\text{heat}} = \exp\big(\beta\mathbf{A}\mathbf{D}^{-1} - \beta\big). \tag{3}$$

Given a diffusion matrix $\mathbf{S}$, a row $\mathbf{s_i}$ indicates the connectivity between the ith node and each node from a global perspective. For example, $s_{i,j}$ represents the degree of connectivity

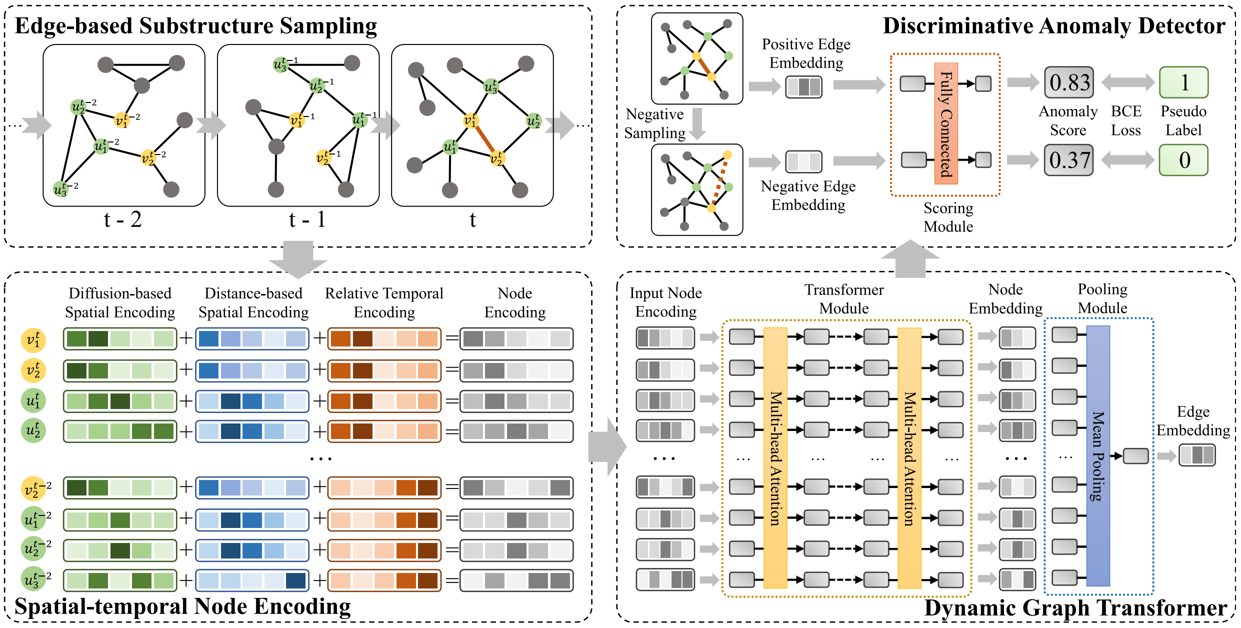


Fig. 2. The overall framework of TADDY. The framework is composed of four components: edge-based substructure sampling, spatial-temporal node encoding, dynamic graph transformer, and discriminative anomaly detector. Here we regard the edge in red $\{v_1^t, v_2^t\}$ as a target edge and exhibit a running example. In edge-based substructure sampling, the target nodes (in yellow) and contextual nodes (in green) from multiple timestamps are sampled to construct the substructure node set, where neighboring node number k and time window size τ are both set to be 3. Then, three types of encoding are computed for each node and further fused into the node encoding. Taking the node encoding as input data, the dynamic graph transformer learns latent node embedding with attention layers and leverages mean pooling to calculate the edge embedding. Finally, in the discriminative anomaly detector, the negative edges are acquired by negative sampling. The scoring module computes the anomaly scores for positive and negative edges. The whole framework is trained with a binary cross-entropy loss in an end-to-end manner.

between the i- and j- th nodes with a continuous value. By leveraging this property, we can pick a fixed number of the most important enclosing nodes for a given target edge. Taking edge $e_{\text{tgt}} = (v_1, v_2)$ as an example, we can compute the connectivity vector of e_{tgt} by adding the connectivity vectors of two target nodes:

$$\mathbf{s}_{e_{\text{tgt}}} = \mathbf{s}_{v_1} + \mathbf{s}_{v_2}. \tag{4}$$

Then, we sort the connectivity vector $\mathbf{s}_{e_{\text{tgt}}}$ and select the top-k nodes with the larger values to form the contextual node set $\mathcal{U}(e_{\text{tgt}})$. Note that the target nodes themselves should be excluded when selecting the top-k connectivity nodes. Finally, the sampled node set for substructure can be denoted as the union of contextual node set and target nodes, which can be formalized as $\mathcal{S}(e_{\text{tgt}}) = \{v_1, v_2, \mathcal{U}(e_{\text{tgt}})\}$.

According to the diffusion-based sampling, we can generate the contextual nodes from a single static graph. However, for dynamic graphs, multiple timestamps should be considered to capture the dynamic evolving. Here, we simply extend the static sampling method to dynamic graphs. Given a target edge $e_{\text{tgt}}^t = (v_1^t, v_2^t)$ at timestamp t, we consider a sequence of graphs $\mathbb{G}_\tau^t = \{\mathcal{G}^{t-\tau+1}, \ldots, \mathcal{G}^t\}$ with length τ, where the time window size τ is a hyper-parameter and determines the receipt fields on time axis. With a sliding window mechanism, TADDY is able to capture dynamic evolving between timestamps $(t-\tau+1)$ and t. Then, for each $\mathcal{G}^i \in \mathbb{G}_\tau^t$, we calculate the diffusion matrix S^i and acquire the corresponding connectivity vector $\mathbf{s}^i_{e^t_{\text{tgt}}}$. By picking the top-$k$ nodes and adding the target nodes, the substructure node set of the ith timestamp can be sampled as $\mathcal{S}^i(e_{\text{tgt}}^t)$. By integrating the node set of multiple timestamps together, we can obtain the final substructure node set $\mathcal{S}(e_{\text{tgt}}^t) = \bigcup_{i=t-\tau+1}^{t} \mathcal{S}^i(e_{\text{tgt}}^t)$.

4.2 Spatial-Temporal Node Encoding

Unlike image and attributed graph data where each data instance (e.g., image patch or node) has its raw features, the dynamic graphs we study in this paper are often unattributed, which indicates that it is hard to find naturally appropriate data as the input of neural network models. This raises the question of *how to construct an informative encoding as network input from unattributed dynamic graphs*. Similar to the concept of one-hot word encoding in NLP, an available solution is to use identity node encoding as the raw node feature, where a unique one-hot vector represents each node. However, identity node encoding has two limitations. First, the one-hot encoding is unable to contain enough structural and temporal information. The one-hot encoding only indicates the nodes' identity, but hard to express its structural roles and temporal status. Second, the identity node encoding is not friendly to large-scale and node-changing dynamic graphs. Third, the fixed dimension cannot adapt to the dynamic changing set of nodes which is a common situation in dynamic graphs.

Inspired by the positional encoding in Transformer [20], we introduce a novel spatial-temporal node encoding for dynamic graph transformers. The proposed node encoding consists of three components, namely diffusion-based spatial encoding, distance-based spatial encoding, and relative temporal encoding. The two terms of spatial encoding represent the structural role of each node from a global and local perspective respectively. The temporal encoding term, differently, provides the temporal information of each element in the substructure node set. To the end, the three encoding terms are fused as the input node encoding which contains comprehensive spatial-temporal information. Note that we generate the encoding by learnable linear projections instead of frequency-aware sin/cos functions used in [20]. The reason is that the learnable functions are more flexible to model the correlations between different timestamps or positions.

In the rest of this subsection, we discuss the three encoding terms sequentially and then introduce the encoding fusion operation.

4.2.1 *Diffusion-Based Spatial Encoding*

As is introduced in Section 4.1, graph diffusion provides a global view of the structural role of each node. With the edge connectivity vector $\mathbf{s}_{e_{\mathrm{tgt}}}$ computed by Eq. (4), it is easy to acquired the strength of connectivity between the target edge and contextual node. Such a property inspires us to design a spatial encoding that bases on graph diffusion. To prevent the indistinguishable encoding caused by similar diffusion values, we do not adopt the raw diffusion values directly but use a rank-based encoding. Specifically, for each node in a single-timestamp substructure node set $v_j^i \in \mathcal{S}^i(e_{\mathrm{tgt}}^t)$, we sort nodes according to their diffusion values and adopt the ranking as the data source. According to the ranks, we compute the diffusion-based spatial encoding with a learnable encoding function (a single-layer linear mapping), which is similar to the learned positional encoding in [20], [41]. The definition of the diffusion-based spatial encoding is given as:

$$\mathbf{x}_{\mathrm{diff}}(v_j^i) = linear(rank(\mathbf{s}^i_{e_{\mathrm{tgt}}}[idx(v_j^i)])) \in \mathbb{R}^{d_{enc}}, \tag{5}$$

where $idx(\cdot)$, $rank(\cdot)$ and $linear(\cdot)$ are the index enquiring function, ranking function and learnable linear mapping respectively, and d_{enc} is the dimension of node encoding.

4.2.2 *Distance-Based Spatial Encoding*

While the diffusion-based spatial encoding capture the global structural information, the local roles of each node should also be considered. Since the transformer model does not take the graph structure (e.g., adjacency matrix) as input like GNNs, we design a distance-based spatial encoding to represent the local connection around the target edge. Concretely, for each node in a single-timestamp substructure node set $v_j^i \in \mathcal{S}^i(e_{\mathrm{tgt}}^t)$, we denote its distance to the target edge as the data source for encoding. The distance to the target edge can be decomposed into the minimum value of the relative distances to the two target nodes. For the target nodes themselves, the distances are denoted as 0. A single-layer linear mapping is served as the learnable encoding function here, which is the same as the diffusion-based encoding. Formally, the distance-based spatial encoding can be expressed as:

$$\mathbf{x}_{\mathrm{dist}}(v_j^i) = linear(min(dist(v_j^i, v_1^i), dist(v_j^i, v_2^i)) \in \mathbb{R}^{d_{enc}}, \tag{6}$$

where $dist(\cdot)$, $min(\cdot)$ and $linear(\cdot)$ are the relative distance computing function, minimum value function and learnable linear mapping respectively, and d_{enc} is the dimension of node encoding.

4.2.3 *Relative Temporal Encoding*

The temporal encoding is to represent the temporal information of each node in the substructure node set. Instead of the absolute encoding in [20], we consider a relative encoding for dynamic graphs. Concretely, for each node in the substructure node set $v_j^i \in \mathcal{S}^i(e_{\mathrm{tgt}}^t)$, the data source for relative time encoding is defined as the difference between the occurring time t of target edge and the current time of timestamp i. The motivation behind this is that our main task is to predict the legality of the target edge, so the relative time to the target edge is a more significant factor for anomaly detection. Similar linear mapping is also applied as the encoding function, and the formal expression of relative temporal encoding is given as:

$$\mathbf{x}_{\mathrm{temp}}(v_j^i) = linear(\|t - i\|) \in \mathbb{R}^{d_{enc}}, \tag{7}$$

where $\|\cdot\|$ and $linear(\cdot)$ are the relative time computing function and the learnable linear mapping respectively, and d_{enc} is the dimension of node encoding.

4.2.4 *Encoding Fusion*

After computing the three terms of encoding, we fuse them as the input node encoding of the downstream transformer model. For the sake of running efficiency, we define the fused node encoding as the summation of three encoding terms rather than concatenating them into a vector with a higher dimension. The encoding fusion is formalized as follows:

$$\mathbf{x}(v_j^i) = \mathbf{x}_{\mathrm{diff}}(v_j^i) + \mathbf{x}_{\mathrm{dist}}(v_j^i) + \mathbf{x}_{\mathrm{temp}}(v_j^i) \in \mathbb{R}^{d_{enc}}. \tag{8}$$

Finally, given a target edge e_{tgt}^t, we calculate the encoding of each node in its substructure node set, and stack them into a encoding matrix which represents the property of e_{tgt}^t. The encoding matrix is represented by:

$$\mathbf{X}(e_{\mathrm{tgt}}^t) = \bigoplus_{v_j^i \in \mathcal{S}(e_{\mathrm{tgt}}^t)} [\mathbf{x}(v_j^i)]^\intercal \in \mathbb{R}^{(\tau(k+2)) \times d_{enc}}, \tag{9}$$

where $\bigoplus$ is the concatenation operation and $[\cdot]^\intercal$ is the transpose operation.

4.3 Dynamic Graph Transformer

To learn knowledge from dynamic graphs, the neural network model should consider both of the spatial structure information and temporal dynamic information. In most situations, the spatial information and temporal information are coupled and should be captured simultaneously for efficient anomaly detection. Taking the dynamic graph in Fig. 2 as an example, the node v_1^t and v_2^t have an connection at time t, a precursor is that their communities have several connections in the previous timestamps, e.g., $u_1^{t-1} - u_2^{t-1}$ and $u_1^{t-2} - u_2^{t-2}$. For the design of dynamic graph encoder, a question that arises is: *how can a neural network-based encoder consider the spatial and temporal information simultaneously?* A general solution in the existing works is to use hybrid networks stacked by spatial module and temporal module. The spatial/temporal modules are employed in such hybrid models to capture spatial/temporal information respectively and separately. For instance, in StrGNN, the GCN serves as a spatial module, and GRU processes the output of GCN from different timestamps to capture temporal information. A limitation of such hybrid models is that they may miss some information that crosses spatial and temporal domains, which further leads to a sub-optimal solution.

To learn the spatial and temporal knowledge in the dynamic graphs, we propose to adopt a transformer model solely as the encoder. With the multiple timestamps of node encoding as input, the *dynamic graph transformer* can simultaneously capture both spatial and temporal features with a single encoder. The dynamic graph transformer is composed of two modules: the transformer module and the pooling module. With the transformer module, the abundant cross-domain knowledge is captured by the attention mechanism, and the final attention layer generates the informative node embeddings. After that, the pooling module aggregates the embedding of all nodes in the substructure node set into an embedding vector for the target edge.

4.3.1 *Transformer Module*

The target of the transformer module is to aggregate the encodings of nodes within a substructure node set into node embeddings. To this end, a number of attention layers are utilized to exchange the information of different nodes. To be concrete, a single attention layer can be written as:

$$\mathbf{H}^{(l)} = attention\Big(\mathbf{H}^{(l-1)}\Big) = softmax\left(\frac{\mathbf{Q}^{(l)}\mathbf{K}^{(l)\top}}{\sqrt{d_{emb}}}\right)\mathbf{V}^{(l)}, \tag{10}$$

where $\mathbf{H}^{(l)}$ and $\mathbf{H}^{(l-1)}$ is the output embedding of the l- and $(l-1)$-th layer, d_{emb} is the dimension of node embedding, and $\mathbf{Q}^{(l)}$, $\mathbf{K}^{(l)}$, $\mathbf{V}^{(l)} \in \mathbb{R}^{(\tau(k+2))\times d_{emb}}$ are the query matrix, key matrix and value matrix for feature transformation and information exchange. Concretely, the $\mathbf{Q}^{(l)}$, $\mathbf{K}^{(l)}$ and $\mathbf{V}^{(l)}$ are computed by:

$$\begin{cases} \mathbf{Q}^{(l)} = \mathbf{H}^{(l-1)}\mathbf{W}_Q^{(l)}, \\ \mathbf{K}^{(l)} = \mathbf{H}^{(l-1)}\mathbf{W}_K^{(l)}, \\ \mathbf{V}^{(l)} = \mathbf{H}^{(l-1)}\mathbf{W}_V^{(l)}, \end{cases} \tag{11}$$

where $\mathbf{W}_Q^{(l)}, \mathbf{W}_K^{(l)}, \mathbf{W}_V^{(l)} \in \mathbb{R}^{d_{emb}\times d_{emb}}$ are the learnable parameter matrices of the l-th attention layer. In an attention layer, $\mathbf{Q}^{(l)}$ and $\mathbf{K}^{(l)}$ calculate the contributions of different nodes' embeddings, while $\mathbf{V}^{(l)}$ projects the input into a new feature space. Eq. (10) combines them and acquires the output embedding of each node by aggregating the information of all nodes adaptively.

In our transformer module, the input of the transformer module $\mathbf{H}^{(0)}$ is defined as the encoding matrix of the target edge $\mathbf{X}(e^t_{\text{tgt}})$, and here we simply set $d = d_{emb} = d_{enc}$ to align the dimension. The output of the final attention layer $\mathbf{H}^{(L)}$ is denoted as the output node embedding matrix $\mathbf{Z}$ of the transformer module, where each row represents an embedding vector of the corresponding node.

4.3.2 *Pooling Module*

The target of pooling module is to transfer the embeddings of nodes in substructure $\mathbf{Z}$ into a target edge embedding vector $\mathbf{z}(e^t_{\text{tgt}})$. Here we utilize the average pooling operation as our pooling function, which has been applied in previous works [36]. The pooling function is formalized as:

$$\mathbf{z}(e^t_{\text{tgt}}) = pooling(\mathbf{Z}) = \sum_{k=1}^{n_s} \frac{(\mathbf{Z})_k}{n_s}, \tag{12}$$

where $(\mathbf{Z})_k$ is the kth row of $\mathbf{Z}$, and $n_s = \tau(k+2)$ is the number of nodes of the substructure node set $\mathcal{S}(e^t_{\text{tgt}})$.

4.4 Discriminative Anomaly Detector

After the edge embedding is acquired, the target of anomaly detection is to learn an anomaly score for each edge in the dynamic graph. Here, we consider an end-to-end framework where a neural network-based anomaly detector computes the anomaly score. However, in our learning setting, there is not any ground-truth anomaly sample in the training set. Such a situation brings a new challenge: *How to learn an anomaly detector without any given anomalous sample?* Our solution is to generate pseudo anomalous edges via a negative sampling strategy and train the anomaly detector with the existing edges in the training set (positive edges) as well as the pseudo anomalous edges (negative edges) together.

A simple negative sampling strategy is performed in our framework. For each timestamp of graph whose number of edges is m^t, we randomly sample the equal number of node pairs as the candidates of negative pairs. Then, we check all these node pairs to ensure that they do not belong to the existing normal edge set in all the training timestamps. We resample a new pair and perform validation for each illegal node pair until the node pair is valid. After negative sampling, we use context sampling to acquire the substructure node set of each negative edge and perform spatial-temporal node encoding. Then, the encoding is fed into the dynamic graph transformer model to obtain the embedding of the negative edge.

The anomaly detector is constructed to discriminate the positive and negative edge embeddings. A fully connected neural network layer with Sigmoid activation is served as the scoring module which computes the anomaly scores of edge embeddings, which is formalized by

$$f(e) = Sigmoid\big(\mathbf{z}(e)\mathbf{w}_S + b_S\big), \tag{13}$$

where $f(e)$ and $\mathbf{z}(e)$ is the anomaly score and edge embedding of edge e respectively, $Sigmoid(\cdot)$ is the Sigmoid activation function, $\mathbf{w}_S \in \mathbb{R}^{d_{emb}}$ and $b_S \in \mathbb{R}$ are the weights and bias parameters of fully connected neural network layer respectively.

We employ a binary cross-entropy loss function with pseudo labels to train the framework in an end-to-end manner. For the positive edges, we expect them to have a small anomaly score, hence their pseudo label is 0; for the negative edges, conversely, the pseudo label is 1. For a training timestamp $\mathcal{G}^t = (\mathcal{V}^t, \mathcal{E}^t)$ whose edge number is m^t, the loss function is given as

$$\mathcal{L} = -\sum_{i=1}^{m^t} \log\big(1 - f(e_{\text{pos},i})\big) + \log\big(f(e_{\text{neg},i})\big), \tag{14}$$

where $e_{\text{pos},i} \in \mathcal{E}^t$ is the ith positive edge and $e_{\text{neg},i} \in \mathcal{E}_n^t \sim P_n(\mathcal{E}^t)$ is the ith negative edge sampled by the negative sampling strategy.

To the end, the overall training procedure of our TADDY framework is depicted in Algorithm 1. The framework is trained in an iterative and end-to-end manner. In each iteration, different negative edges are sampled to prevent training bias and over-fitting. For all positive and negative edges, we perform substructure sampling, node encoding, transformer embedding and anomaly score computing sequentially. Finally, the parameters are updated by back propagation under the supervision of binary cross-entropy loss function. When the framework is well trained, the anomaly scores for test edges can be obtained by executing the line 6-9 in Algorithm 1.

Algorithm 1. The Overall Training Procedure of TADDY

```
Input: Training set of dynamic graph: $\mathbb{G} = \{\mathcal{G}^t\}_{t=1}^T$, Number of
    training epochs: $I$, Number of sampled contextual nodes: $k$,
    Size of time window: $\tau$.
 1: Randomly initialize the parameters of encoding linear
    mappings, transformer model and scoring function
 2: for $i \in 1, 2, \cdots, I$ do
 3:   for timestamp $\mathcal{G}^t = (\mathcal{V}^t, \mathcal{E}^t) \in \{\mathcal{G}^t\}_{t=\tau}^T$ do
 4:     Sample negative edge set $\mathcal{E}_n^t \sim P_n(\mathcal{E}^t)$ by negative
        sampling strategy
 5:     for $e \in \mathcal{E}^t \cup \mathcal{E}_n^t$ do
 6:       Set $e$ as the target edge and sample its substructure
          node set $\mathcal{S}(e)$ with $\tau(k+2)$ nodes
 7:       Calculate node encoding matrix $\mathbf{X}(e)$ via Eqs. (5), (6),
          (7), and (8)
 8:       Calculate edge embedding vector $\mathbf{z}(e)$ via Eqs. (10),
          (11), and (12)
 9:       Calculate anomaly score $f(e)$ via Eq. (13)
10:     end for
11:     Calculate loss function $\mathcal{L}$ via Eq. (14)
12:     Back propagation and update the parameters
13:   end for
14: end for
```

4.5 Complexity Analysis

In this subsection, we analyze the time complexity of each component in TADDY framework. For edge-based substructure sampling, the complexity is mainly caused by the computation of graph diffusion, which is $\mathcal{O}(T\widetilde{n}^2)$ where $\widetilde{n}$ is the average number of nodes for graph timestamps and T is the number of timestamps. In spatial-temporal node encoding and dynamic graph transformer, we process all nodes in the substructure node set for each target edge, which brings a complexity of $\mathcal{O}(\tau k)$ for one edge. Therefore, for I iterations, the total time complexity is $\mathcal{O}(\tau kmI)$ where m is the number of edges. For discriminative anomaly detector, the time complexity is $\mathcal{O}(mI)$, which is far less than the other components and can be ignored. To sum up, the overall time complexity is $\mathcal{O}(\tau kmI + T\widetilde{n}^2)$.

5 Experiments

In this section, we evaluate the performance of the proposed TADDY via extensive experimental studies. We first introduce the setup for our experiments. We demonstrate the experimental results in the rest three subsections, including performance comparison, parameter study, and ablation study.

TABLE 2
The Statistics of the Datasets

Dataset	♯ nodes	♯ edges	Avg. Degree
UCI Messages	1,899	13,838	14.57
Digg	30,360	85,155	5.61
Email-DNC	1,866	39,264	42.08
Bitcoin-Alpha	3,777	24,173	12.80
Bitcoin-OTC	5,881	35,588	12.10
AS-Topology	34,761	171,420	9.86

For each dataset, the number of nodes, the number of edges, and the average degree are reported.

5.1 Experimental Setup

5.1.1 Datasets

We evaluate our proposed TADDY framework on six real-world benchmark datasets of dynamic graphs. The statistics of the datasets are given in Table 2, and the detailed descriptions are demonstrated as follows.

UCI Messages[1] [42] is a social network dataset collected from an online community of students at University of California, Irvine. In the constructed dynamic graph, each node indicates a user, and each edge represents a message between two users.

Digg[2] [43] is a network dataset collected from a news website digg.com. In Digg dataset, each node is a website user, and each edge indicates that one user replies to another user.

Email-DNC[3] [44] is network of emails in the 2016 Democratic National Committee email leak. Each node corresponds to a person in the United States Democratic Party, and each edge denotes that a person has sent an email to another person.

Bitcoin-Alpha[4] and *Bitcoin-OTC*[5] [45], [46] are two who-trusts-whom networks of bitcoin users trading on the platforms from www.btc-alpha.com and www.bitcoin-otc.com respectively. In these two datasets, the nodes are the users from the platform, and an edge appears when one user rates another on the platform.

AS-Topology[6] [47] is a network connection dataset collected from autonomous systems of the Internet. Each node in the graph corresponds to an autonomous system, and each edge indicates a connection between two autonomous systems.

We pre-process the datasets following previous works [9], [10]. The edges in each dataset are annotated with timestamps. The repeated edges in the edge stream are removed in the pre-processing phase. Since there is no ground-truth anomalous edge in the original datasets, we follow the approach used in [9] to inject anomalous edges in each dataset. To be concrete, the training data is totally clean. For each

1. http://konect.cc/networks/opsahl-ucsocial
2. http://konect.cc/networks/munmun_digg_reply
3. http://networkrepository.com/email-dnc
4. http://snap.stanford.edu/data/soc-sign-bitcoin-alpha
5. http://snap.stanford.edu/data/soc-sign-bitcoin-otc
6. http://networkrepository.com/tech-as-topology

snapshot $\mathcal{G}^t$ in the test set, we randomly link $p_A \times m^t$ pairs of disconnected nodes as anomalous edges, where p_A is the anomaly proportion indicating the percentage of anomalous edges in each snapshot, and m^t is the (original) number of edges in $\mathcal{G}^t$.

5.1.2 Baselines

We compared TADDY framework against six state-of-the-art baselines that can be categorized into two groups: graph embedding methods and deep dynamic graph anomaly detection methods.

DeepWalk [48] is a random walk-based method for graph embedding. It generates random walks with a given length starting from a target node and uses a Skip-gram-like manner to learn embedding for unattributed graphs.

node2vec [49] considers breadth-first traversal and depth-first traversal when generating random walks. The Skip-gram technology is also employed to learn node embedding in node2vec.

Spectral Clustering [50] learns node embedding by maximizing the similarity between bodes in neighborhood. The intuition behind this method is to preserve the local connection relationship in graphs.

NetWalk [9] is a representative anomaly detection method for dynamic graph. It utilizes a random walk-based approach to generate contextual information and learns node embedding with an auto-encoder model. The node embeddings are updated incrementally over time via a reservoir-based algorithm. The anomaly is detected using a dynamic-updated clustering on the learned embedding.

AddGraph [10] is an end-to-end dynamic graph anomaly detection approach. It leverages a GCN module to capture spatial information, and employs a GRU-attention module to extract short- and long- term dynamic evolving.

StrGNN [13] is an end-to-end graph neural network model for detecting anomalous edges in dynamic graphs. It leverages an h-hop enclosing subgraph as the network's input and combines GCN and GRU to learn structural-temporal information for each edge.

For the graph embedding methods, the K-means clustering-based anomaly detector, which is presented in NetWalk [9] is utilized to detect anomalies based on the learned node embeddings.

5.1.3 Experimental Design

In our experiments, 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 p_A (i.e., 1%, 5%, and 10%) when injecting the anomalous data into the test set. To measure the performance of the proposed framework as well as the baselines, ROC-AUC (AUC for short) is employed as our primary metric. The ROC curve indicates a plot of true positive rate against false positive rate where anomalous labels are viewed as "positive". AUC is defined as the area under the ROC curve, which indicates the probability that a randomly selected anomalous edge is ranked higher than a normal edge. The value range of AUC is 0 to 1, and a larger value represents a better anomaly detection performance.

5.1.4 Parameter Settings

All the parameters can be tuned by 5-fold cross-validation on a rolling basis. For edge-based substructure sampling, we set the number k of contextual nodes to 5 and τ is selected from 1 to 4. We use PPR diffusion in our experiments, which is computed by Eq. (2). For spatial-temporal node encoding, the dimension of encoding d_{enc} is 32, which is the same as d_{emb} in Dynamic Graph Transformer. The number of attention layers is 2 for all the datasets, and the number of attention heads is 2. The framework is trained by Adam optimizer with a learning rate of 0.001. We train UCI Messages, Bitcoin-Alpha, and Bitcoin-OTC datasets with 100 epochs and the rest three datasets for 200 epochs. The snapshot size is set to 1,000 for UCI Messages and Bitcoin-OTC, 2,000 for Email-DNC and Bitcoin-Alpha, and 6,000 for Digg and AS-Topology, respectively.

5.1.5 Computing Infrastructures

The proposed method is implemented using PyTorch 1.7.1 [51]. All experiments are conducted on a computer server with four Quadro RTX 6000 (24GB memory each) GPUs, an Intel Xeon Silver 4214R (2.40 GHz) CPU and 64 GB of RAM.

5.2 Anomaly Detection Results

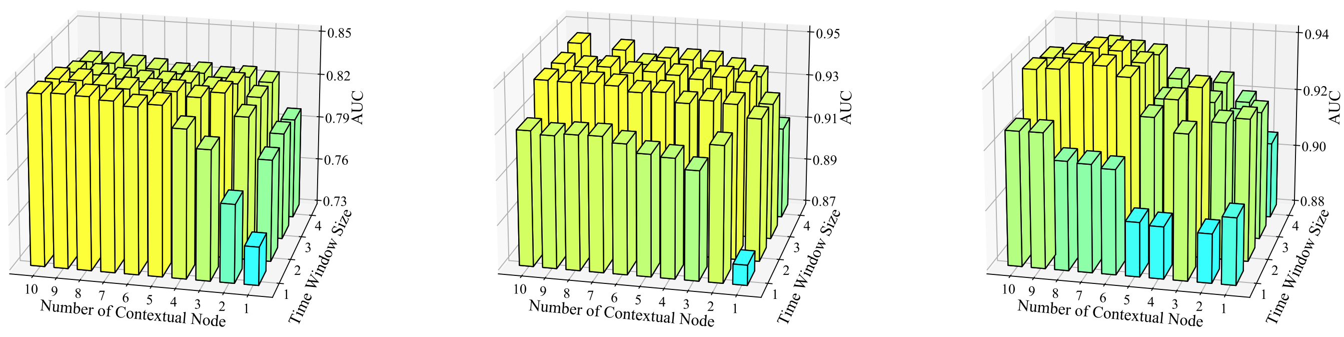
In this subsection, we report anomaly detection performance and compare our proposed TADDY framework with the baseline methods. The anomaly detection performance comparison of average AUC on all test timestamps is demonstrated in Table 3. We summarize the following observations for the results:

- The proposed TADDY framework consistently outperforms all baselines on the six dynamic graph datasets with different anomaly proportions. Compared to the baseline method with the best results, TADDY reaches a performance gain of 4.49% on AUC averagely. The main reason is that TADDY extracts the spatial-temporal information by constructing informative node encoding and captures structural dynamic and temporal dynamic simultaneously with a transformer encoder.
- Compared to three graph embedding-based methods, the deep dynamic graph anomaly detection methods, NetWalk, AddGraph, StrGNN and TADDY, always have a more competitive performance. We attribute this performance advantage to the leverage of temporal information. By considering the interaction in previous timestamps, these methods learn the dynamic evolving in graphs.
- TADDY has a larger performance gain when the anomalies are scarce. Concretely, the average performance gap on AUC between TADDY and the best baseline under 1% anomaly proportion is 5.35%, while the gaps under 5% and 10% anomaly proportions are 3.69% and 4.43%, respectively. A possible reason is that we train the framework with an efficient negative sampling strategy, which ensures the robustness under different anomaly proportion in the test set.
- On the two Bitcoin datasets, TADDY achieves more remarkable results. Compared to the best baseline,

TABLE 3
Anomaly Detection Performance Comparison Reported in AUC Measure

Methods	UCI Messages			Digg			Email-DNC		
	1%	5%	10%	1%	5%	10%	1%	5%	10%
node2vec	0.7371	0.7433	0.6960	0.7364	0.7081	0.6508	0.7391	0.7284	0.7103
Spectral Clustering	0.6324	0.6104	0.5794	0.5949	0.5823	0.5591	0.8096	0.7857	0.7759
DeepWalk	0.7514	0.7391	0.6979	0.7080	0.6881	0.6396	0.7481	0.7303	0.7197
NetWalk	0.7758	0.7647	0.7226	0.7563	0.7176	0.6837	0.8105	0.8371	0.8305
AddGraph	0.8083	0.8090	0.7688	0.8341	0.8470	0.8369	0.8393	0.8627	0.8773
StrGNN	0.8179	0.8252	0.7959	0.8162	0.8254	0.8272	0.8775	0.9103	0.9080
TADDY	**0.8912**	**0.8398**	**0.8370**	**0.8617**	**0.8545**	**0.8440**	**0.9348**	**0.9257**	**0.9210**
Methods	Bitcoin-Alpha			Bitcoin-OTC			AS-Topology		
	1%	5%	10%	1%	5%	10%	1%	5%	10%
node2vec	0.6910	0.6802	0.6785	0.6951	0.6883	0.6745	0.6821	0.6752	0.6668
Spectral Clustering	0.7401	0.7275	0.7167	0.7624	0.7376	0.7047	0.6685	0.6563	0.6498
DeepWalk	0.6985	0.6874	0.6793	0.7423	0.7356	0.7287	0.6844	0.6793	0.6682
NetWalk	0.8385	0.8357	0.8350	0.7785	0.7694	0.7534	0.8018	0.8066	0.8058
AddGraph	0.8665	0.8403	0.8498	0.8352	0.8455	0.8592	0.8080	0.8004	0.7926
StrGNN	0.8574	0.8667	0.8627	0.9012	0.8775	0.8836	0.8553	0.8352	0.8271
TADDY	**0.9451**	**0.9341**	**0.9423**	**0.9455**	**0.9340**	**0.9425**	**0.8953**	**0.8952**	**0.8934**

The upper three baselines belong to graph embedding methods, and the middle three baselines belong to deep dynamic graph anomaly detection methods. The best performing method in each experiment is in bold.



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

Fig. 3. The sensitivity of contextual node number k and time window size τ on three datasets. The vertical axis represents the AUC values of TADDY with different k and τ. A warmer color indicates a higher AUC value.

the average AUC gain on Bitcoin-Alpha and Bitcoin-OTC is 6.42%, which is significantly higher than the AUC gain on the rest datasets (3.53%). The reason for such remarkable advantages is that the abnormality of Bitcoin transaction is more closely related to spatial-temporal dynamic, and TADDY can successfully capture such dynamic by comprehensive node embedding and attention mechanism.

5.3 Parameter Sensitivity

In this subsection, we investigate the influence of hyperparameters on TADDY, including the number of contextual nodes and time window size in edge-based substructure sampling, the dimension of encoding/embedding and the number of attention layers in dynamic graph transformer, and the ratio of training data. Here we carry out the experiments on three datasets (UCI Messages, Bitcoin-Alpha and Bitcoin-OTC). In these experiments, we keep the other parameter as default, and the performance is examined under a 10% anomaly proportion setting.

5.3.1 Parameters of Edge-Based Substructure Sampling

To evaluate the effect of number of contextual nodes k and time window size τ in edge-based substructure sampling, we set the range of k to $\{1,2,3,4,5,6,7,8,9,10\}$ and the range of τ to $\{1,2,3,4\}$. The sensitivity of k and τ is exhibited in Fig. 3. According to the results, we have the following observations.

When the contextual node number k is extremely small, the anomaly detection performance is relatively pool. With the growth of k, there is a significant boost in AUC. When $k > 5$, the detection performance tends to be stable, with k getting larger. The performance trend indicates that contextual node sets with a sufficient size are significant to anomaly detection since the anomalous property of edges highly relies on their neighboring local structure. However, when considering an excessive number of contextual nodes, the performance gain is minor. However, a large k is harmful to the running efficiency due to the linear relationship between

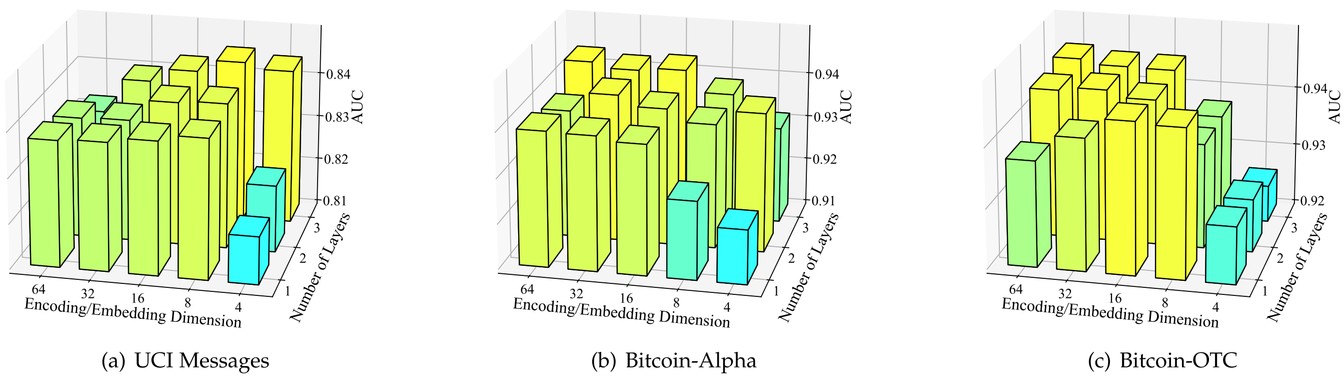


Fig. 4. The sensitivity of encoding/embedding dimension d and the number of layers L on three datasets. The vertical axis represents the AUC values of TADDY with different d and L. A warmer color indicates a higher AUC value.

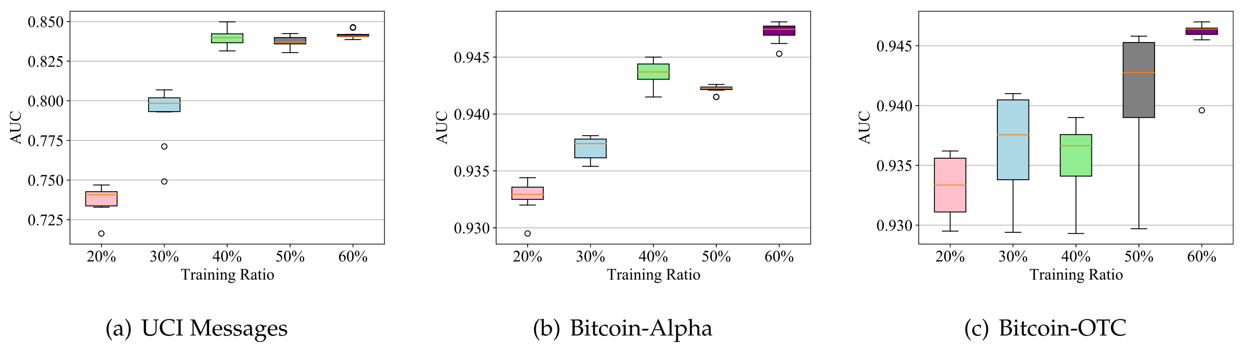


Fig. 5. AUC values of TADDY on three datasets with different training ratios. The circular markers indicate the results which is viewed as outliers.

time complexity and k. Consequently, we fix the value of k to 5 for the consideration of the trade-off between performance and efficiency.

For different datasets, the appropriate size τ of the time window is different. For instance, a smaller time window is beneficial to UCI Messages, while the two Bitcoin datasets need a more extended time horizon. The main reason is that the temporal reliance of edges in dynamic graphs highly depends on the datasets. When the edge appearance has a long-term dependency on the previous graph evolving, a larger time window is needed to capture the dependency. For other datasets like UCI Messages where anomalies are related to the latest snapshots, a wide time window may result in noisy and redundant input for TADDY framework. Therefore, we select the best τ value for each dataset in our experiments.

5.3.2 *Parameters of Dynamic Graph Transformer*

We further investigate the effort of encoding/embedding dimension d and the number of layers L in Transformer model. The value of d is selected from $\{4, 8, 16, 32, 64\}$ and which of L is selected from $\{1, 2, 3\}$. The results are summarized in Fig. 4.

In Bitcoin-Alpha and Bitcoin-OTC, the AUC values increase gradually from $d = 4$ to $d = 16$, and then go steady after $d = 32$. Such observation demonstrates that when d is small, the model may miss useful information. For UCI Messages, $d = 8$ seems to be the best choice. When d getting larger, there is no significant performance degradation. Our explanation is that when d is too large, the noisy information would be captured by the transformer model.

Compared to d, the number of layers has a limited impact on performance. An exception is Bitcoin-Alpha, whose AUC drops when $L = 1$, which indicates that a sufficient number of layers can bring adequate interaction among structuring nodes. As such, we fix $L = 2$ for each dataset to balance the running speed and detection performance.

5.3.3 *Training Ratios*

In this experiment, we discuss the performance of TADDY framework using training data with different ratios. The range of training ratio is $\{20\%, 30\%, 40\%, 50\%, 60\%\}$ and other parameters are set to default. Fig. 5 displays the results on three datasets.

We observe from Fig. 5 that the AUC values increase smoothly when the training ratio goes larger, demonstrating that more training data provides a better supervision signal for training. We can also find that even if the training data is rate (20%), our framework still has a competitive performance, especially on two Bitcoin datasets. This observation shows that TADDY can learn an informative representation even trained with scarce data. Moreover, the variance of AUC decreases with the increase of the training ratio, which illustrates that our proposed framework tends to have a stable performance when training data is adequate.

5.4 Ablation Study

To study the contribution of each component in the spatial-temporal node encoding towards the overall performance,

TABLE 4
Ablation Study for TADDY and its Variants on Three Datasets

	UCI Messages	Bitcoin-Alpha	Bitcoin-OTC
TADDY	0.8370	**0.9423**	**0.9262**
`w/o diff. enc.`	0.8304	0.9329	0.9153
`w/o dist. enc.`	0.5362	0.5043	0.5187
`w/o temp. enc.`	**0.8399**	0.9326	0.9021

we conduct the ablation study of the proposed TADDY framework. In particular, We evaluate the following variants of the node encoding: `w/o diff. enc.`, `w/o dist. enc.` and `w/o temp. enc.`, where the diffusion-based spatial encoding, distance-based spatial encoding and relative temporal encoding are discarded respectively when excusing the encoding fusion. We perform the ablation study with 10% anomaly proportion for each dataset, and all the parameters are set as default. Table 4 reports the proposed framework and its variants on three datasets. We have the following observations according to the results:

- The distance-based spatial encoding is the most critical term in node encoding. Without this term, the AUC values decrease sharply to about 50%, which indicates that the anomalies become indistinguishable. This observation proves that the local structural information is significant in detecting anomalies, which is also pointed out in previous works [13], [36].
- The diffusion-based spatial encoding and relative temporal encoding both have a minor contribution in detecting anomalies. In the vast majority of cases, removing one of them would lead to a slight performance drop. We infer from the results that the diffusion-based spatial encoding provides a global view for graph structure which has a minor relation to anomaly detection. Moreover, the temporal encoding points out the occurrence time of neighborhoods which is relatively unimportant since the nodes have been selected to the substructure set.
- In most of the cases, combining all of the three types of encoding has the highest AUC values, excepted on UCI Messages dataset. This shows that using a comprehensive spatial-temporal encoding is meaningful to anomaly detection. As for the exception, we guess that emphasizing the relative time distance may lead to an over-fitting on such a property in some cases, which further results in a slight side effect on performance.

6 Conclusion

In this paper, we make the first attempt to utilize transformer models for the graph anomaly detection problem in dynamic graph scenarios. We propose an end-to-end anomaly detection framework, TADDY, which is composed of four components: edge-based substructure sampling, spatial-temporal node encoding, dynamic graph transformer, and discriminative anomaly detector. Our framework constructs an informative and comprehensive node encoding to better represent the roles of nodes in an evolving graph space and successfully captures the coupled spatial-temporal information within dynamic graphs with a sole transformer model. Experiments on several real-world datasets show that the proposed framework detects anomalies with high effectiveness in dynamic graphs and outperforms the existing methods significantly.

We believe that the proposed framework provides a novel perspective to learn informative representation from dynamic graphs and present a new solution for the graph anomaly detection problem. In the future, we will improve TADDY by considering a more meaningful encoding method and anomaly-aware negative sampling strategy to better learn the anomaly patterns in dynamic graphs. We will also extend the transformer model to more types of graph data, e.g., heterogeneous graph and spatial-temporal graph.

References

[1] L. Wang, Z. Yu, F. Xiong, D. Yang, S. Pan, and Z. Yan, "Influence spread in GEO-social networks: A multiobjective optimization perspective," *IEEE Trans. Cybern.*, vol. 51, no. 5, pp. 2663–2675, May 2021.

[2] S. Ji, S. Pan, E. Cambria, P. Marttinen, and P. S. Yu, "A survey on knowledge graphs: Representation, acquisition, and applications," *IEEE Trans. Neural Netw. Learn. Syst.*, early access, Apr. 26, 2021, doi: 10.1109/TNNLS.2021.3070843.

[3] Y. Zheng *et al.*, "Clustering social audiences in business information networks," *Pattern Recognit.*, vol. 100, 2020, Art. no. 107126.

[4] Y. Gao, X. Li, H. Peng, B. Fang, and P. Yu, "HinCTI: A cyber threat intelligence modeling and identification system based on heterogeneous information network," *IEEE Trans. Knowl. Data Eng.*, early access, Apr. 20, 2020, doi: 10.1109/TKDE.2020.2987019.

[5] D. Jin *et al.*, "A survey of community detection approaches: From statistical modeling to deep learning," *IEEE Trans. Knowl. Data Eng.*, early access, Aug. 11, 2021, doi: 10.1109/TKDE.2021.3104155.

[6] F. Xia *et al.*, "Graph learning: A survey," *IEEE Trans. Artif. Intell.*, vol. 2, no. 2, pp. 109–127, Apr. 2021.

[7] H. Peng *et al.*, "LIME: Low-cost incremental learning for dynamic heterogeneous information networks," *IEEE Trans. Comput.*, early access, Feb. 11, 2021, doi: 10.1109/TC.2021.3057082.

[8] P. Jiao *et al.*, "Temporal network embedding for link prediction via VAE joint attention mechanism," *IEEE Trans. Neural Netw. Learn. Syst.*, early access, Jun. 9, 2021, doi: 10.1109/TNNLS.2021.3084957.

[9] W. Yu, W. Cheng, C. C. Aggarwal, K. Zhang, H. Chen, and W. Wang, "NetWalk: A flexible deep embedding approach for anomaly detection in dynamic networks," in *Proc. 24th ACM SIGKDD Int. Conf. Knowl. Discov. Data Mining*, 2018, pp. 2672–2681.

[10] L. Zheng, Z. Li, J. Li, Z. Li, and J. Gao, "AddGraph: Anomaly detection in dynamic graph using attention-based temporal GCN," in *Proc. 28th Int. Joint Conf. Artif. Intell.*, 2019, pp. 4419–4425.

[11] C. C. Aggarwal, Y. Zhao, and S. Y. Philip, "Outlier detection in graph streams," in *Proc. IEEE 27th Int. Conf. Data Eng.*, 2011, pp. 399–409.

[12] S. Ranshous, S. Harenberg, K. Sharma, and N. F. Samatova, "A scalable approach for outlier detection in edge streams using sketch-based approximations," in *Proc. SIAM Int. Conf. Data Mining*, 2016, pp. 189–197.

[13] L. Cai *et al.*, "Structural temporal graph neural networks for anomaly detection in dynamic graphs," 2020, *arXiv:2005.07427*.

[14] C. Yang, L. Zhou, H. Wen, Z. Zhou, and Y. Wu, "H-VGRAE: A hierarchical stochastic spatial-temporal embedding method for robust anomaly detection in dynamic networks," 2020, *arXiv:2007.06903*.

[15] H. Peng *et al.*, "Streaming social event detection and evolution discovery in heterogeneous information networks," *ACM Trans. Knowl. Discov. Data*, vol. 15, no. 5, pp 1–33, 2021.

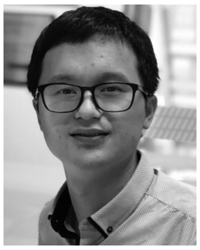
[16] K. Sricharan and K. Das, "Localizing anomalous changes in time-evolving graphs," in *Proc. ACM SIGMOD Int. Conf. Manage. Data*, 2014, pp. 1347–1358.

[17] E. Manzoor, S. M. Milajerdi, and L. Akoglu, "Fast memory-efficient anomaly detection in streaming heterogeneous graphs," in *Proc. 22nd ACM SIGKDD Int. Conf. Knowl. Discov. Data Mining*, 2016, pp. 1035–1044.

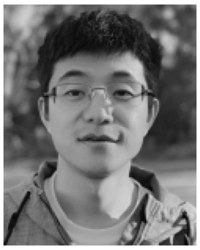
[18] D. Eswaran, C. Faloutsos, S. Guha, and N. Mishra, "SpotLight: Detecting anomalies in streaming graphs," in *Proc. 24th ACM SIGKDD Int. Conf. Knowl. Discov. Data Mining*, 2018, pp. 1378–1386.

[19] T. N. Kipf and M. Welling, “Semi-supervised classification with graph convolutional networks,” in *Proc. Int. Conf. Learn. Representations*, 2017, pp. 1–14.
[20] A. Vaswani *et al.*, “Attention is all you need,” in *Proc. 31st Int. Conf. Neural Inf. Process. Syst.*, 2017, pp. 6000–6010.
[21] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, “BERT: Pre-training of deep bidirectional transformers for language understanding,” in *Proc. Conf. North Amer. Chapter Assoc. Comput. Linguistics, Hum. Lang. Technol.*, 2019, pp. 4171–4186.
[22] Z. Lan, M. Chen, S. Goodman, K. Gimpel, P. Sharma, and R. Soricut, “ALBERT: A lite bert for self-supervised learning of language representations,” in *Proc. Int. Conf. Learn. Representations*, 2019, pp. 1–17.
[23] Y. Liu *et al.*, “RoBERTa: A robustly optimized bert pretraining approach,” 2019, *arXiv:1907.11692*.
[24] Z. Yang, Z. Dai, Y. Yang, J. Carbonell, R. R. Salakhutdinov, and Q. V. Le, “XLNet: Generalized autoregressive pretraining for language understanding,” in *Proc. 33rd Int. Conf. Neural Inf. Process. Syst.*, 2019, pp. 5753–5763.
[25] L. Liu, W. L. Hamilton, G. Long, J. Jiang, and H. Larochelle, “A universal representation transformer layer for few-shot image classification,” in *Proc. Int. Conf. Learn. Representations*, 2021, pp. 1–12.
[26] N. Carion, F. Massa, G. Synnaeve, N. Usunier, A. Kirillov, and S. Zagoruyko, “End-to-end object detection with transformers,” in *Proc. Eur. Conf. Comput. Vis.*, 2020, pp. 213–229.
[27] A. Dosovitskiy *et al.*, “An image is worth 16x16 words: Transformers for image recognition at scale,” in *Proc. Int. Conf. Learn. Representations*, 2021, pp. 1–21.
[28] S. Zheng *et al.*, “Rethinking semantic segmentation from a sequence-to-sequence perspective with transformers,” in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2021, pp. 6881–6890.
[29] K. Han *et al.*, “A survey on visual transformer,” 2020, *arXiv:2012.12556*.
[30] Y. Tay, M. Dehghani, D. Bahri, and D. Metzler, “Efficient transformers: A survey,” 2020, *arXiv:2009.06732*.
[31] S. Yun, M. Jeong, R. Kim, J. Kang, and H. J. Kim, “Graph transformer networks,” in *Proc. 33rd Int. Conf. Neural Inf. Process. Syst.*, 2019, pp. 11 983–11 993.
[32] Z. Hu, Y. Dong, K. Wang, and Y. Sun, “Heterogeneous graph transformer,” in *Proc. Web Conf.*, 2020, pp. 2704–2710.
[33] Y. Rong *et al.*, “Self-supervised graph transformer on large-scale molecular data,” in *Proc. Int. Conf. Neural Inf. Process. Syst.*, 2020, pp. 12559–12571.
[34] J. Zhang, H. Zhang, C. Xia, and L. Sun, “Graph-Bert: Only attention is needed for learning graph representations,” 2020, *arXiv:2001.05140*.
[35] Y. Liu, S. Pan, M. Jin, C. Zhou, F. Xia, and P. S. Yu, “Graph self-supervised learning: A survey,” 2020, *arXiv:2103.00111*.
[36] Y. Liu, Z. Li, S. Pan, C. Gong, C. Zhou, and G. Karypis, “Anomaly detection on attributed networks via contrastive self-supervised learning,” *IEEE Trans. Neural Netw. Learn. Syst.*, early access, Apr. 05, 2021, doi: 10.1109/TNNLS.2021.3068344.
[37] J. Klicpera, S. Weißenberger, and S. Günnemann, “Diffusion improves graph learning,” in *Proc. Int. Conf. Neural Inf. Process. Syst.*, 2019, pp. 13 333–13 345.
[38] K. Hassani and A. H. Khasahmadi, “Contrastive multi-view representation learning on graphs,” in *Proc. Int. Conf. Mach. Learn.*, 2020, pp. 3451–3461.
[39] L. Page, S. Brin, R. Motwani, and T. Winograd, “The pagerank citation ranking: Bringing order to the web,” Stanford InfoLab, Stanford, CA, USA: Tech. Rep.1999–66, 1999.
[40] R. I. Kondor and J. Lafferty, “Diffusion kernels on graphs and other discrete structures,” in *Proc. Int. Conf. Mach. Learn.*, 2002, pp. 315–322.
[41] J. Gehring, M. Auli, D. Grangier, D. Yarats, and Y. N. Dauphin, “Convolutional sequence to sequence learning,” in *Proc. 34th Int. Conf. Mach. Learn.*, 2017, pp. 1243–1252.
[42] T. Opsahl and P. Panzarasa, “Clustering in weighted networks,” *Social Netw.*, vol. 31, no. 2, pp. 155–163, 2009.
[43] M. De Choudhury , H. Sundaram, A. John, and D. D. Seligmann, “Social synchrony: Predicting mimicry of user actions in online social media,” in *Proc. Int. Conf. Comput. Sci. Eng.*, 2009, pp. 151–158.
[44] R. A. Rossi and N. K. Ahmed, “The network data repository with interactive graph analytics and visualization,” in *Proc. 29th AAAI Conf. Artif. Intell.*, 2015, pp. 4292–4293. [Online]. Available: http://networkrepository.com
[45] S. Kumar, F. Spezzano, V. Subrahmanian, and C. Faloutsos, “Edge weight prediction in weighted signed networks,” in *Proc. IEEE 16th Int. Conf. Data Mining.*, 2016, pp. 221–230.
[46] S. Kumar, B. Hooi, D. Makhija, M. Kumar, C. Faloutsos, and V. Subrahmanian, “REV2: Fraudulent user prediction in rating platforms,” in *Proc. 11th ACM Int. Conf. Web Search Data Mining.*, 2018, pp. 333–341.
[47] B. Zhang, R. Liu, D. Massey, and L. Zhang, “Collecting the internet AS-level topology,” *ACM SIGCOMM Comput. Commun. Rev.*, vol. 35, no. 1, pp. 53–61, 2005.
[48] B. Perozzi, R. Al-Rfou , and S. Skiena, “DeepWalk: Online learning of social representations,” in *Proc. 20th ACM SIGKDD Int. Conf. Knowl. Discov. Data Mining*, 2014, pp. 701–710.
[49] A. Grover and J. Leskovec, “node2vec: Scalable feature learning for networks,” in *Proc. 22nd ACM SIGKDD Int. Conf. Knowl. Discov. Data Mining*, 2016, pp. 855–864.
[50] U. Von Luxburg , “A tutorial on spectral clustering,” *Statist. Comput.*, vol. 17, no. 4, pp. 395–416, 2007.
[51] A. Paszke *et al.*, “PyTorch: An imperative style, high-performance deep learning library,” in *Proc. Int. Conf. Neural Inf. Process. Syst.*, 2019, pp. 8026–8037.

Yixin Liu received the BS and MS degrees from Beihang University, Beijing, China, in 2017 and 2020, respectively. He is currently working toward the PhD degree in computer science with Monash University, Melbourne, VIC, Australia. His research interests include data mining, machine learning, and deep learning on graphs.

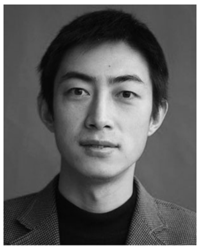


Shirui Pan received the PhD degree in computer science from the University of Technology Sydney (UTS), Ultimo, NSW, Australia. He is currently a senior lecturer with the Faculty of Information Technology, Monash University, Australia. His research interests include data mining and machine learning. He has authored or coauthored more than 100 research papers in top-tier journals and conferences, including the *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *IEEE Transactions on Neural Networks and Learning Systems*, and the *IEEE Transactions on Knowledge and Data Engineering*. He was the recipient of the ARC Future Fellowship Award in 2021.



Yu Guang Wang received the PhD degree in applied mathematics from the University of New South Wales, Australia. He is currently an adjunct associate lecturer with UNSW Sydney. He is also a scientist with Max Planck Institute for Mathematics in Sciences, Mathematics Machine Learning Group. His research interests include computational mathematics, statistics, machine learning, and data science.

Fei Xiong received the PhD degree from Beijing Jiaotong University, Beijing, China, in 2013. From 2011 to 2012, he was a visiting scholar with Carnegie Mellon University, Pittsburgh, PA, USA. He is currently an associate professor with the School of Electronic and Information Engineering, Beijing Jiaotong University. His research interests include Web mining, complex networks, and complex systems.



Liang Wang recevied the PhD degree in computer science from the Shenyang Institute of Automation, Chinese Academy of Sciences, Shenyang, China, in 2014. In 2017, he was a postdoctoral researcher with Northwestern Polytechnical University, Xi'an, China, where he is currently an associate professor. His research interests include ubiquitous computing, mobile crowd sensing, and data mining.

Qingfeng Chen received the BSc and MSc degrees in mathematics from Guangxi Normal University, China, in 1995 and 1998, respectively, and the PhD degree in computer science from the University of Technology Sydney, in September 2004. He is currently a professor with Guangxi University, China, and the Hundred Talent Program of Guangxi. His research interests include bioinformatics, data mining, and artificial intelligence.

Vincent CS Lee (Senior Member, IEEE) received the PhD degree in adaptive systems from The University of NewCastle, Australia, in 1992. He is currently an associate professor with the Department of Data Science and Artificial Intelligence, Faculty of IT, Monash University, Australia. He is a multi-interdisciplinary researcher spanning adaptive signal processing and control system, computational intelligence, AI in economic and finance, deep machine learning and computer vision, digital health process mining, and information security and network cryptography disciplines.