GADY: Unsupervised Anomaly Detection on Dynamic Graphs

Yuyang Tian
Xi'an Jiaotong University
Xi'an
2191414578@stu.xjtu.edu.cn

***Abstract*—Anomaly detection on dynamic graphs aims to identify entities that exhibit abnormal behaviours compared to the standard patterns observed in the graphs and their temporal information. It has attracted increasing attention due to its applications in various domains, such as finance, network security, and social networks. However, existing methods face two significant challenges: (1) dynamic structure capture challenge: how to capture graph structure with complex temporal information effectively, and (2) negative sampling challenge: how to construct high-quality negative samples for unsupervised learning. To address these challenges, we propose a Generative Anomaly Detection on Dynamic Graphs (GADY). GADY is a continuous dynamic graph model that can capture fine-grained temporal information to tackle the dynamic structure capture challenge, overcoming the limitations of existing discrete methods. Specifically, we propose to use Prioritization Temporal Aggregation and Status Features to boost the dynamic graph encoder for anomaly detection. For the second challenge, we introduce a novel use of Generative Adversarial Networks to generate negative subgraphs. In addition, auxiliary loss functions were introduced in the generator training objective, ensuring the generated samples' diversity and quality simultaneously. Extensive experiments show that our proposed GADY significantly outperforms the state-of-the-art method on three real-world datasets. Supplementary experiments further validate the effectiveness of our model design and the necessity of each component.**

I. Introduction

Anomaly detection on dynamic graphs has become an essential task given its numerous applications, such as social media spammer detection [Ye and Akoglu(2015)], fraudulent transaction detection [Dou et al.(2020)], and network intrusion detection [Shone et al.(2018)]. Capturing anomalies on dynamic graphs can assist us in better understanding and capturing the evolution of social networks [Ye and Akoglu(2015)], financial transactions [Dou et al.(2020)], and disease diagnosis [Khan et al.(2021)] with fine-grained time information.

Nowadays, many studies in this area still capture dynamic graph patterns using discrete-time dynamic methods [Zheng et al.(2019a)], [Cai et al.(2021)], [Liu et al.(2021)]. They usually divide the entire temporal graph into several snapshots and conduct anomaly detection based on static Graph Neural Networks and Recurrent Neural Networks(RNNs). However, these methods fall short of time information loss, which is unavoidable when dividing the entire graph into several snapshots, and this is because they have to delete repeated edges in the same snapshot to apply static GNN methods. Also, the time differences between edges in the same snapshot are often ignored, resulting in time information loss; further discussion about this time information loss can be found in the appendix (https://anonymous.4open.science/r/GADY). Due to the limitation of discrete-time dynamic methods and sparsity labelled data, SAD [Tian et al.(2023)] tries to detect node anomalies using continuous-time dynamic methods in a semi-supervised task setting and get a boost in detection performance.

However, although some datasets contain a few labelled data, these data are only limited to node labels [Tian et al.(2023)], [Huang et al.(2023)], and real labelled edges are still tough to obtain. It should be noted that the label of node anomaly cannot be simply transformed into edge anomaly. For example, an anomalous node does not imply that all interactions involving this node are anomalies because even an anomalous node may still have many normal interactions to disguise itself, which further strengthens the difficulty of obtaining real labels. In brief, most dynamic graph datasets still have no labelled edge data. Therefore, edge anomaly detection on dynamic graphs is still mainly carried out in unsupervised mode.

Although significant research progress has been made in anomaly detection on dynamic graphs, there is potential for further advancements in two key aspects: dynamic structure capture methods, focusing on capturing dynamic graph structures with complex time information, and negative sampling methods, aiming to construct high-quality and diverse negative samples for unsupervised learning given a fact that most of the dynamic graph datasets still have no labelled edges data.

For the dynamic structure capture challenge, anomaly detection task usually requires that construction is powerful enough to capture potential anomaly patterns under complex dynamic graph structures. This is because the most effective models so far are usually encoder-decoder architectures [Liu et al.(2021)], [Tian et al.(2023)], and a high-quality embedding generated from the encoder is crucial for the decoder to get anomaly scores.

In this field, strGNN [Cai et al.(2021)] extracted h-hop closed subgraphs centred on edges and then used GCN and GRU to model the structural information on snapshots and the correlation between snapshots. Taddy [Liu et al.(2021)] uses a transformer to process diffusion-based spatial encoding, distance-based spatial encoding, and relative time encoding and then obtain edge representation through the pooling layer,

thus obtaining the anomaly score. These two models adopt different discrete-time dynamic methods to capture dynamic graph patterns but achieve limited detecting performance due to time information loss. SAD [Tian et al.(2023)] proposes using TGAT [Xu et al.(2020)] as its encoder that can capture fine-grained dynamic graph patterns. Combined with a semi-supervised learning setting, SAD obviously achieves better detection results on anomaly node detection. However, its encoder is still limited for two reasons. First, its encoder struggles to prioritize edges at different times effectively. Moreover, it does not consider the recent advancements in static Graph Neural Networks regarding graph expressiveness based on the WL-test, which highlights that injective aggregation functions can significantly improve model performance [Xu et al.(2019)]. Therefore, a more powerful model for edge anomaly detection on dynamic graphs is expected in this field.

As mentioned above, research on edge anomaly detection on dynamic graphs also faces the negative sampling challenge: the inability to construct excellent negative samples for unsupervised learning. Besides, for the moment, the tremendous difficulty of acquiring real labelled data blocks the possible supervised or semi-supervised training methods. Research about unsupervised anomaly detection is still meaningful in promoting the development of edge anomaly detection on dynamic graphs. To conduct unsupervised learning on anomaly detection on dynamic graphs, StrGNN [Cai et al.(2021)] propose a context-dependent negative sampling method to construct negative samples by rules for unsupervised learning artificially. Specifically, for each edge, they randomly keep the head node or tail node and replace the unselected node with another irrelevant node, thus generating a negative sample for training. This straightforward negative sampling method shows some effectiveness. Still, it cannot ensure the diversity and quality of negative samples and cannot perfectly satisfy the requirements of anomaly detection on dynamic graphs. Ideally, the generated negative samples will enable the model to learn challenging and diverse anomaly patterns efficiently during the training phase, and this can help the model cope with real-world anomalies.

Based on the above challenges, we propose **GADY**: Unsupervised **G**enerative **A**nomaly Detection on **Dy**namic Graphs. Specifically, we propose using a more powerful dynamic graph encoder for anomaly detection. We propose to use a prioritization temporal aggregation to prioritize different events, thus strengthening the capturing of complex time information, and use status features to boost the encoder's ability, thus enhancing the expressiveness of the model. Besides, we propose to utilize an anomaly generator to acquire high-quality and diverse synthetic anomalous edges for capturing potential anomaly patterns hidden in complex dynamic graphs. We design the loss function of the generator to guide the generation process of synthetic anomalous edges and the discriminator's loss function to help it comprehensively consider normal and abnormal edge detection results.

We follow the evaluation settings proposed by [Liu et al.(2021)] for a fair comparison. Experimental results show that our proposed GADY outperforms the current state of the art significantly on three benchmarks, demonstrating the superiority of our design choice. We also investigate the potential of current continuous dynamic graphs for anomaly detection. Detailed supplementary experiments on the GAN module and generator design also validate the necessity of the anomaly generator and the other components in GADY.

In summary, our main contributions are as follows:

- We propose GADY, an end-to-end unsupervised anomaly detection method, which has designed a loss function for the generator to generate high-quality and diverse anomalous dynamic edges and another loss function for the Discriminator to consider normal and abnormal edges comprehensively.
- We identify the drawbacks of the existing dynamic graph encoder and propose a more robust dynamic graph encoder for anomaly detection to capture potential anomaly patterns hidden in complex dynamic graph information.
- Extensive experiments on three datasets show that our model has achieved a maximum improvement of 14.6% in anomaly detection compared with existing methods and achieved extraordinary performance in all anomaly proportions of all datasets. Extensive supplementary experiments again demonstrate the necessity of the anomaly generator and the superiority of model settings.

II. Related Work

A. Anomaly Detection on Dynamic Graph

In recent years, anomaly detection has focused on using deep learning methods to model the entire process [Jin et al.(2021)], [Zhao et al.(2021)], [Tariq et al.(2022)], [Zhou et al.(2021)], [Han and Yuan(2021)], [Guo et al.(2022)]. The processing methods of dynamic graphs can be divided into discrete processing methods and continuous processing methods. Many methods have been proposed using discrete methods in recent years to solve this task, such as Addgraph [Zheng et al.(2019a)] using GCN to extract graph structure information on slices and then using GRU-attention modules to construct long and short-term relationships between slices. StrGNN [Cai et al.(2021)] extracted h-hop closed subgraphs centred on edges and then used GCN and GRU to model the structural information on snapshots and the correlation between snapshots, respectively. TADDY [Liu et al.(2021)] uses a transformer to process diffusion-based spatial encoding, distance-based spatial encoding, and relative time encoding and then obtain edge representation through the pooling layer, thus obtaining the anomaly score. Continuous methods can be more helpful for anomaly detection with their strong modelling capabilities and sufficient time information. Recently, SAD [Tian et al.(2023)] proposed using continuous dynamic methods to detect anomalies using the semi-supervised method, which is different from our work.

B. Generative Adversarial Network

Nowadays, much research about applications of GAN on anomaly detection focuses on Computer Vision. For exam-

ple, AnoGAN [Schlegl et al.(2017)] can capture the latent distribution of normal data through training. Because the behaviour patterns of abnormal and normal data are different, after processing by AnoGAN, the residual between original input and output is usually much larger than normal images, thus detecting anomalies in the image. Fence-GAN [Ngo et al.(2019)] improves the model training effect by defining different loss functions to locate the generated samples at the edge of the normal sample distribution.

There is also some research focused on the applications of GAN on anomaly detection on static graphs. For example, OCAN [Zheng et al.(2019b)] uses LSTM-Autoencoder to learn the representation of normal users and then modifies the generator to generate abnormal data complementary to normal data and obtain a better discriminator after training. However, to the best of our knowledge, we are the first pioneering GAN application on anomaly detection on dynamic graphs.

III. Preliminaries

A. Notations and Problem Definition

This section formally introduces the anomaly detection problem in dynamic graphs.

1) Notations: [Continuous-time dynamic graphs] Continuous-time dynamic graphs are important for modelling relational data in the real world. This kind of graph can be represented as $G = (\mathcal{V}, \mathcal{E})$, where $\mathcal{V} = \{v_1, v_2, \ldots, v_m\}$ is the set of nodes involved in all dynamic events, and $\mathcal{E}$ denotes a sequence of dynamic events. Typically, let $\mathcal{E} = \{e(u_i, v_i, t_i) \colon i \in [0, n], u_i \in \mathcal{V}, v_i \in \mathcal{V}\}$ be an event sequence that consists of the temporal network, and n is the number of events, with an event $e(u_i, v_i, t_i)$ indicating an edge happens from source node u_i to destination node v_i at time t_i. Obviously, omitting nodes that do not interact at all, we can use the $\mathcal{E}$ to gain all the information of G. Therefore, we will directly use $G = \{e(u_i, v_i, t_i) \colon i \in [0, n]\}$ to represent the whole G in the following text. Financial transactions can be modelled as a continuous-time dynamic graph. For instance, source node u_i and destination node v_i are different users, and an edge (u_i, v_i, t_i) represents that a user u_i transfers money to another user v_i at time t_i. A set of fund transfer records constitutes a continuous-time dynamic graph.

2) Problem Definition: In real applications, it is usually hard to get labelled data due to the unacceptable cost and the high variability of real edge labels. The goal of our work is to detect anomalies when actual labelled data is unavailable. To evaluate the performance of GADY, we usually inject synthetic anomalies into the test dataset and train the model with normal data and abnormal data generated by other negative sampling methods. Our task is defined as follows:

[Unsupervised anomaly detection on dynamic graphs] Consider a continuous-time dynamic graph $G = \{e(u_i, v_i, t_i) \colon i \in [0, n]\}$, where $e(u_i, v_i, t_i) = (u_i, v_i, t_i)$ and n is the number of events. Let $G_{train} \subseteq G$ be a subgraph of G used for training. $G_{test} = G - G_{train}$ is a subgraph of G used for testing. The unsupervised anomaly detection on dynamic graphs problem aims to learn a model from G_{train} and use the model to get anomaly scores for each edge.

IV. Methodology

A. Overall Framework

Given a dynamic graph $G = \{e(u_i, v_i, t_i) \colon i \in [0, n]\}$. Our goal is to get anomaly scores of different edges. To achieve this goal, we design GADY and show its structure in Figure 1. Our model has two parts: an anomaly generator and an anomaly discriminator. The anomaly discriminator is composed of two parts: an anomaly encoder and an anomaly detector. The anomaly generator inputs positive edges and their surrounding graphs. After concatenating with noise $z_i \in (-1, 1)$, the inputs are sent to neural networks to output fake edges with their fake surrounding graphs. These generated anomalous subgraphs will be used to train the anomaly discriminator. The anomaly encoder in the anomaly discriminator inputs the normal edges and anomalous edges and outputs source node embedding and destination node embedding for each edge. The anomaly detector in the anomaly discriminator inputs source and destination node embeddings of edges and outputs anomaly scores for these edges. These scores can be used for further judgment.

B. Anomaly Generator

1) Framework Design: The purpose of the anomaly generator is to generate high-quality and diverse negative samples. For the high quality of negative samples, our definition is to be as similar as possible to real samples but different from them. Such negative samples can make the discriminator work hard to distinguish true and false samples during training. For the diversity of samples, our definition is to have as many abnormal patterns as possible so that the model can learn to see more kinds of abnormalities during the training process, thereby improving the detection ability of the model. The whole generator part is designed following these two principles.

The standard input of our generator is the concatenation of source node embeddings, timestamps, neighbour embeddings, edge features and edge timestamps. After concatenating inputs with random noise, the generator generates final outputs through a Multilayer Perceptron. Besides, because graph neural networks usually have several layers, the generator also takes embedding from the last layer as input to monitor the aggregation process fully. Through our experiment, we found that different types of input noise are suitable for different datasets. In our model, we use Gaussian noise for its superiority. Moreover, generating anomalous samples on dynamic graphs still needs reconstruction due to the different data scales of different outputs.

Based on the above considerations, we define the model as follows:

$$Neg = R(\tanh(f(src_emb, ts, nei_emb, edg_fea, edg_ts, z_i))), \tag{1}$$

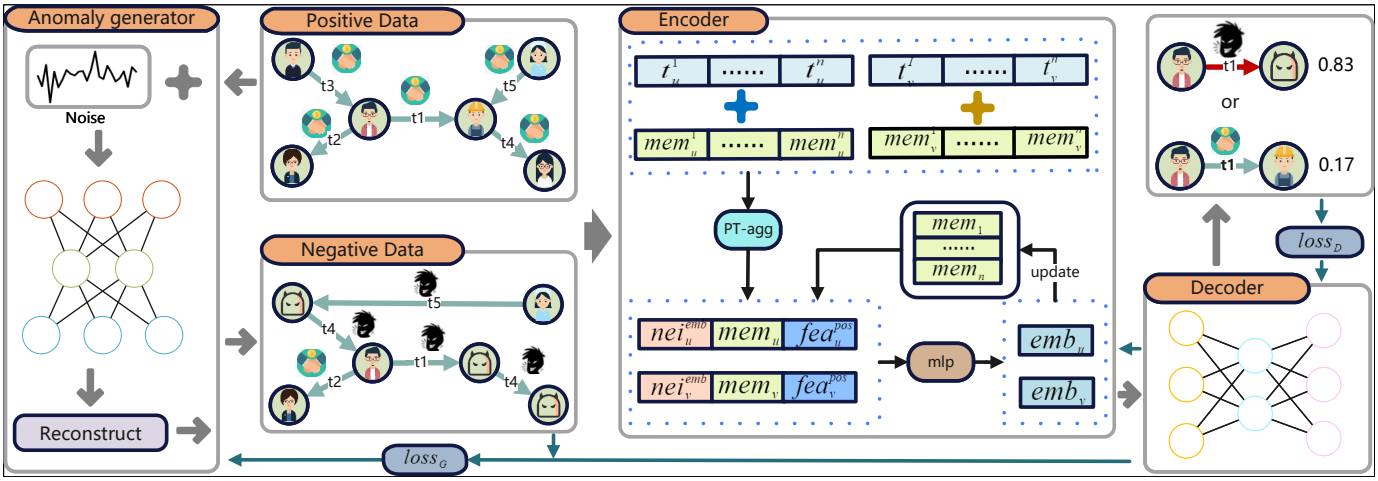

Fig. 1. The architecture of GADY network includes two parts. The first part is the anomaly generator, which generates negative samples through normal data and random noise, reconstructs the generated samples according to their scales and inputs them into the encoder part with the normal samples. The second part is the discriminator, which contains the encoder and decoder. After getting the samples, the encoder obtains the edge embedding by continuously aggregating the head and tail nodes' time neighbours' information according to Prioritization Temporal Aggregation (PT-agg). Finally, the embedding of the edge is fed into the decoder. The decoder obtains the anomaly score of the edge through the representation of the head node and the tail node, finally judging how possible the target edge is anomalous.

where f is defined as a five-layer multilayer perceptions, src_emb represents source node embedding, ts represents timestamps, nei_emb represents neighbor embeddings, and z_i is the noise. $\tanh()$ is an activation function reconstructing the output to $[0, 1]$, and R is a function reconstructing the output embedding of $\tanh()$ to the correct scale. Neg denotes the results of the anomaly generator, which consist of negative source node features, negative neighbour embeddings, negative edge features, and negative edge timestamps. After the process of multilayer perceptions, the inputs are reconstructed into outputs. After that, we reconstruct the generator's output to different original data scales and get the final anomaly scores through the discriminator's encoding-decoding process.

2) Generator Loss Function: The design of the loss function usually affects the training goal of the model. In order to ensure that the anomaly generator can generate anomalous edges with high quality and diversity, we propose a new loss function with two parts for anomaly detection. For the anomaly detection task, 1 denotes anomaly, and 0 denotes normal. The traditional generator loss function is shown as follows:

$$L_G = -E_{z\sim p_z(z)}[\log(1 - D(G(z)))]. \tag{2}$$

$D()$ represents the Discriminator, $G()$ represents the Generator, z is the input noise, $p_z(z)$ is the distribution of z, and E is the expected value. This loss function means that the generator aims to generate anomalous edges that are totally similar to normal edges, which usually misdirect the discriminator, thus harming detecting performance. Ideally, the anomalous edges should be similar to normal edges for boosting the discriminator's ability but not totally equal to normal edges in case of misleading discriminator [Xia et al.(2022)]. This is because generated anomalous edges are just normal edges if there are no differences between them. To get such high-quality anomalous edges, we propose a quality evaluation part to judge its quality:

$$L_Q = \frac{1}{N}\sum_{i=1}^{N}\log(1 - |0.1 - D(Neg)|), \tag{3}$$

where $D(Neg)$ denotes the anomaly scores of generated anomalous edges and N is the number of edges. 0.1 determines the expected anomaly scores from the discriminator. The value of 0.1 means that the anomaly generator hopes that the generated anomalous edges get scores of 0.1 through the discrimination of the discriminator, which assists the generator in generating anomalous edges similar to normal edges but not totally equal to them. Any deviation from this target will lead to punishment.

Another important criterion for the generated anomalous edges is diversity. A high diversity of generated samples can assist the discriminator in learning different anomaly patterns, subsequently boosting discrimination performance. We design the diversity evaluation part to achieve this purpose:

$$L_D = \frac{1}{\frac{1}{nm}\sum_{k=1}^{n}(\sum_{q=1}^{m}(CL_{kq}))}. \tag{4}$$

In this part, CL_{kq} refers to the coefficient of variation on the q-th dimension of K-th outputs of the generator dimension. These values evaluate the diversity of the different dimensions of the generator's outputs. In our model, these dimensions are source node features, neighbour embeddings, edge features, and edge timestamps dimensions. We measure the diversity of the fake samples through the mean values of all generator

outputs on different dimensions. And CL_{kq} can be formulated as follows:

$$CL_{kq} = \frac{\frac{1}{N}\sum_{i=1}^{N}(||Neg_{kq}^{i} - \mu^{kq}||_2)}{\mu^{kq}}, \tag{5}$$

where μ^{kq} is the mean value of q-th dimension of K-th outputs, Neg_{kq}^{i}) is the i-th value of q-th dimension of K-th Negative outputs. Finally, to comprehensively assess the quality and diversity of generated samples, the loss function is defined as follows:

$$L_G = L_Q + L_D, \tag{6}$$

The ultimate loss function can be represented as follows:

$$\begin{aligned} L_G(G, D, z_i) = \frac{1}{N}\sum_{i=1}^{N}\log(1 - |0.1 - D(Neg)|) \\ + \frac{1}{\frac{1}{nm}\sum_{k=1}^{n}(\sum_{q=1}^{m}(CL_{kq}))}. \end{aligned} \tag{7}$$

C. Anomaly Discriminator

The purpose of the discriminator is to generate anomaly scores of normal and abnormal edges. This requires it to capture complex continuous dynamic graph structures. Nowadays, the encoder-decoder structure is a powerful framework. Our model design follows this architecture and defines two modules: anomaly encoder and anomaly decoder. The anomaly encoder gets inputs from generated anomalous edges and normal edges and outputs node embedding. The anomaly decoder generates anomaly scores based on node embeddings. The two modules are defined as follows.

1) Anomaly Encoder: The performance of anomaly detection models usually heavily depends on the encoding ability of their encoder, which needs to capture potential anomaly patterns under complex dynamic graph structures. However, many existing methods of anomaly detection on dynamic graphs still adopt discrete-time dynamic graph methods, which inevitably result in time information loss. Therefore, we follow the continuous-time dynamic graphs settings for better performance.

Besides, existing methods of anomaly detection on dynamic graphs with continuous-time dynamic methods usually pass information from recent events with priority based on recency. However, they don't effectively prioritize these events, leading to sub-optimal results. Recently, some works [Maron et al.(2019)], [Morris et al.(2019)] began to explore the expressive power of graph neural networks. Weisfeiler-Leman test (WL-test) test has been used to evaluate the limitations of graph neural networks in recent years. However, existing anomaly detection methods on dynamic graphs do not consider the expressive power limitations of their encoder. To boost the power of the anomaly encoder, we propose to use the status features to break WL test limitations, which are seldom considered in previous methods. Therefore, we propose to use a new prioritization aggregation function to enhance model ability.

(a) Prioritization Temporal Aggregation. Existing works utilize poor prioritization aggregation function, thus weakening the ability to capture time information [Tian et al.(2023)]. We propose to use a decay coefficient to map the priorities of different events.

The aggregation process can be defined as follows:

$$\widetilde{h}_u^{(l)}(t) = \sum_{(v,e,t')\in Nei(u,t)} agg(concat(h_v^{(l-1)}(t), e))\alpha^{-\beta(\Delta t)}, \tag{8}$$

where h denotes the hidden embedding, $Nei(u,t)$ denotes the neighbor of u before time t. α and β are hyper-parameters, l is the aggregation layer and e is edge feature. Then we get embeddings $h_u^{(l)}(t)$ through the hidden embedding at layer l-1 $h_u^{(l-1)}$ and the aggregated embedding $\widetilde{h}_u^{(l)}(t)$.

$$h_u^{(l)}(t) = Upd^{l}(concat(h_u^{(l-1)}(t), \widetilde{h}_u^{(l)}(t))). \tag{9}$$

In our model, we instantiate functions $agg()$ and $Upd()$ as multilayer perceptions(MLP).

(b) Status Features. To boost the model construction ability, we first calculate how many temporal paths there are between two nodes and consider this as a status feature in the message-passing process. For an edge (u, v, t), the status features can be computed through the following process:

Formally, let $\nu_i^{(t^+)}$ record the nodes that needed to be updated after an interaction i involved tightly after t. It needs to be updated after each interaction through the following equation:

$$\nu_u^{(t^+)} = \nu_v^{(t^+)} = \nu_v^{(t)} \cup \nu_u^{(t)}, \tag{10}$$

where t^+ refers to the time tightly after time t. Then let $s_{(i\to j)}^{(t)} \in \mathbb{N}^d$ denote the status feature of node i relative to node j at time t. It is initialized through the following equation:

$$s_{i\to j}^{(0)} = \begin{cases} [1, 0, \cdots, 0]^T, & if\ i\ =\ j \\ [0, 0, \cdots, 0]^T, & if\ i\ \neq\ j \end{cases}, \tag{11}$$

where $s_{i\to j}^{(t)}[k]$ refers to how many temporal walks we can get from i to j in k steps. Then, we can recursively update it through the following formulas:

$$s_{i\to v}^{(t^+)} = Ps_{i\to u}^{(t)} + s_{i\to v}^{(t)} \quad \forall i \in \nu_u^{(t)} \tag{12}$$

$$s_{j\to u}^{(t^+)} = Ps_{j\to v}^{(t)} + s_{j\to u}^{(t)} \quad \forall j \in \nu_v^{(t)} \tag{13}$$

where P is a propagation $d \times d$ matrix initialized by padding a (d-1)-dim unit matrix with zeros on its top row and rightmost column. $Ps_{i\to u}^{(t^+)}$ compute number of temporal walks from neighbor u and $s_{i\to v}^{(t^+)}$ is the direct temporal walks from i to v. By looping through the above process, we can finally get the number of temporal walks of one node relative to other nodes.

(c) Overall Process. The overall process of the anomaly encoder is shown as follows:

Before the model starts training, we first calculate how many temporal paths between two nodes and consider this as a status feature in the message-passing process.

In the encoding process, we use the following process to get the edge embedding of the target edge.

$$E(e(u,v,t)) = concat(emb(u,t), emb(v,t)), \tag{14}$$

$$emb^k(u,t) = \sum_{j \in Nei_u^k([0,t])} h(S_j(t), v(j,t), E(u,j,t), s_{j\to u}^{(t)}), \tag{15}$$

where h is an aggregation function defined in injective temporal aggregations. $S_j(t)$ are memory of j. $E(u,j,t)$ and $v(j,t)$ are the attributes of edges and nodes, which can be set to zero if missing. $s_{j\to u}^{(t)}$ denotes the status feature from j to u at time t. By recursively aggregating the information of k-layers of neighbours, we can finally get the node embeddings.

To keep updating memory [Rossi et al.(2020)], we use the following process: For each interaction (u,v,t), we find temporal neighbours for head and tail and get their messages. Then, we use a $mem()$ function to construct recent messages and update the memory.

The following formula can define this process for node u:

$$meg_j(t) = concat(S_j(t^-), S_i(t^-), \Delta t, e_{vi}(t)), i \in Nei_v, \tag{16}$$

$$meg_u(t) = agg(meg_j(t_j), \cdots, meg_j(t_j)), j \in Nei_u, \tag{17}$$

where $agg()$ function is instantiated as mean message(average all messages) and t^- refers to the time tightly before time t. After getting the ultimate message, we update the memory with the $mem()$ function.

$$S_u(t) = mem(meg_u(t), S_u(t^-)), \tag{18}$$

$mem()$ function is instantialized with a Gated Recurrent Unit(GRU) and $S_u(t)$ is the memory of u at time t.

2) Anomaly Detector: The anomaly detector aims to get anomaly scores given two node embeddings of an edge. Although this part can be designed to be very complex, for simplicity, we define the model as follows: first, given two node embeddings $n\_embedding^u$ and $n\_embedding^v$, we concatenate them to get embedding of the edge between them $e\_embedding^{uv}$.

$$e\_embedding^{uv} = concat(n\_embedding^u, n\_embedding^v). \tag{19}$$

Then, we use a 2-layer Linear with Relu activation function to get ultimate anomaly scores.

$$Score = Linear(ReLU(Linear(e\_embedding^{uv})).sigmoid(), \tag{20}$$

where $.sigmoid()$ is a normalization function that normalizes the output to [0,1].

3) Discriminator loss function: The loss function of the discriminator usually needs to be evaluated comprehensively in terms of detecting performance. That means the loss design should fairly evaluate the normal and anomalous data. Thus, we determine the loss function as follows:

$$L_D(Neg_i, x_i, D) = \frac{1}{N}\sum_{i=1}^{N}[\log(D(Neg_i)) - \log(1 - D(x_i))], \tag{21}$$

where Neg_i is the i-th generated negative sample, x_i is the normal sample, and D is the Discriminator. In this equation, we fairly evaluate normal and abnormal edges to get a fair evaluation of the discriminator on the anomaly detection task.

Algorithm 1 Training process of GADY

```
Preprocess: calculate the positional features s^(t)_(i→j) of each node i relative to another node j.
repeat
    Input normal data to generator G and get fake samples
    Input true and fake samples to Discriminator D
    For each interaction (u, v, t)
        For each layer l :
            Aggregate messages from temporal neighbours
        Get embeddings of the two nodes u and v
    Concatenate to get the edge embedding E(e(u, v, t))
    Decode to get judgement D(e(u, v, t))
    Update the memory
    Pass back L_G and L_D
until End
```

D. Training process

In the training process, the generator receives noise and outputs the generated pseudo edges. Secondly, the pseudo and actual edges are inputted into the encoder for encoding to obtain the edges embedding. Thirdly, the edge embeddings are decoded into the anomaly scores to determine how possible this edge is anomalous. Finally, the loss function of the generator L_G and the loss of the discriminator L_D are passed back to update parameters. This process can be found in Algorithm 1.

V. Experiments

A. Experiment Settings

1) 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 to assess the performance of anomaly detection models. However, due to the sparsity of anomaly instances, the AUC results often primarily rely on normal data detection performance, limiting the difference among different methods. To further explore the effect of our Anomaly Generator module, we introduce another metric AP(Average Precision), which is shown in V-C.

AP (Average Precision) is a metric for evaluating the performance of object detection models on a single class. It was introduced to deal with imbalanced data [He and

TABLE I

COMPARISON RESULTS OF AUC AND AP METRIC BETWEEN DIFFERENT METHODS INJECTING DIFFERENT ABNORMAL RATIOS ON DIFFERENT DATASETS, WHERE THE BEST PERFORMANCE IS SHOWN IN **BOLD** AND THE SECOND BEST PERFORMANCE IS MARKED WITH UNDERLINE. 1%, 5%, AND 10% REFER TO ANOMALY RATIOS, RESPECTIVELY.

Method	UCI			Bitcoin-OTC			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	0.7785	0.7694	0.7534	0.8105	0.8371	0.8305
TGN	0.8771	8667	0.8539	0.9411	0.9284	0.9196	0.9677	0.9474	0.9326
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	0.9012	0.8775	0.8836	0.8775	0.9103	0.908
TADDY	0.8912	0.8398	0.837	0.9455	0.934	0.9425	0.9348	0.9257	0.921
GADY	**0.9505**	**0.9585**	**0.9481**	**0.9691**	**0.9743**	**0.9771**	**0.9779**	**0.9822**	**0.9815**

Garcia(2009)]. This metric is based on the area under the Precision-Recall (PR) curve, which shows the precision and recall of the model at different thresholds. Unlike AUC, AP comprehensively considers the accuracy and recall rate, which reflect whether the model can find all the anomalies and whether the anomalies found are correct. Thus, it can only focus on the anomaly detection performance without the interface of abundant normal data. Therefore, we believe that the AP metric can comprehensively measure the detection ability of the anomaly detection model better, which is seldom considered in the previous works [Yu et al.(2018)], [Zheng et al.(2019a)], [Cai et al.(2021)], [Liu et al.(2021)]. In the exploration of the GAN effects part, we use this metric to further explore the performance of our model.

2) Datasets: To test the performance of our model, we evaluated our framework on three datasets[1]: **UCI Message** [Opsahl and Panzarasa(2009)], **Email-DNC** [Rossi and Ahmed(2015)], and **Bitcoin-OTC** [Kumar et al.(2018)]. The detailed information of these datasets is as follows:

- **UCI Message** [Opsahl and Panzarasa(2009)] is a social network collected from a forum of the University of California, Irvine. It has 1,899 nodes and 13,838 edges, where each node represents a student and an edge represents a message sent.
- **Email-DNC** [Rossi and Ahmed(2015)] is an email network from the 2016 Democratic National Committee email leak. It has 1,866 nodes and 39264 edges, where each node represents a person in the Democratic Party, and an edge represents an email sent from one person to another.
- **Bitcoin-OTC** [Kumar et al.(2018)] is a network of who trusts whom among users who trade on the Bitcoin platform. It has 5,881 nodes, and 35,588 edges where nodes are users and edges are ratings between users.

3) Baselines: We selected four classes of models as our baselines: 1. Traditional non-deep learning methods Node2vec [Grover and Leskovec(2016)] Spectral Clustering [Von Luxburg(2007)] DeepWalk [Perozzi et al.(2014)]; 2. Methods using a combination of deep and traditional learning NetWalk [Yu et al.(2018)]; 3. Contrast dynamic graph methods Temporal Graph Networks(TGN) [Rossi et al.(2020)]; 4. Anomaly detection using deep learning methods AddGraph [Zheng et al.(2019a)] StrGNN [Cai et al.(2021)] TADDY [Liu et al.(2021)]. A detailed introduction to baselines can be found in the appendix (https://anonymous.4open.science/r/GADY).

4) Experiment Pipeline: For unsupervised anomaly detection on dynamic graphs, existing methods inject different ratios of anomalies into the test dataset. The ratios are usually set to 1%, 5%, and 10% for simulating different ratios of anomalies that might be encountered in the real world. We follow this setup for the convenience of comparing. In the training dataset, models are trained with other negative sampling methods (different from the method used in test set construction). For the discrete-time anomaly detection methods, they divide the original dynamic graph into different slices [Cai et al.(2021)], [Liu et al.(2021)] for further modelling. Although this process may have the information loss described in the introduction section, we attribute this to the flaws in these models themselves, not to the unfairness of the experiment setup.

5) Experiment Setup: In order to evaluate the performance of our model for anomaly detection and make a fair comparison between baselines, we follow the test set built method in TADDY [Liu et al.(2021)]. Specifically, we perform spectral clustering on the whole graph, randomly select nodes belonging to different categories, and remove node pairs duplicated with the original dataset. Then, we randomly generate timestamps for node pairs within the time range of the test set, and finally, we add the generated pseudo-edges to the test set and sort them by time.

In our experiments, number of layers is set to 2, and the training ratio is set to 0.5, which is used in TADDY [Liu et al.(2021)]. Hyper-parameter α is set to 2, and β is set to 0.00001. Three hyper-parameters are selected based on experiment results. Further detailed implementation details

[1]https://drive.google.com/drive/folders/15AmNpT2QTYjtDG7nD1WsIHQMgYAsUYxL?usp=sharing

and hyper-parameter selecting strategies can be found in the appendix (https://anonymous.4open.science/r/GADY).

Finally, we implement our method using PyTorch 1.12.1 [Paszke et al.(2019)]. All experiments are performed on a Linux server with a 2.30GHz Intel(R) Xeon(R) Silver 4316 CPU and NVIDIA Tesla-V100S GPU with 32GB memory.

B. Main Results

We tested our model on three datasets and set the injected anomaly ratio to 1%, 5%, and 10%, respectively. The experimental results are shown in Table I. From the experimental results, we summarize the following conclusions:

- Our model has excellent results, and the use of the GAN module can significantly improve the ability of the model to detect anomalies. Our model outperforms state-of-the-art methods by 13.7% at most on the UCI dataset with an anomaly ratio of 10%, achieving SOTA on three real-world datasets.
- The outstanding results show the effectiveness of applying a powerful anomaly encoder to capture potential anomaly patterns hidden in complex dynamic graph information.
- Our model performance is insensitive to anomaly ratios. From the table, it can be found that other models are more sensitive to abnormal injection ratios. However, our method remains highly effective regardless of the anomaly ratio. This will allow our model to have a broader range of applicability to different anomaly ratios.

C. GAN Ablation Study

We conduct experiments to explore the effects of the GAN module. Specifically, we compare GADY with GADY w/o GAN in all datasets with three different anomaly ratios. We implement GADY w/o GAN using the negative sampling methods from StrGNN [Cai et al.(2021)] that is the same as TADDY [Liu et al.(2021)] and the anomaly discriminator from GADY. The results are shown in Figure 2

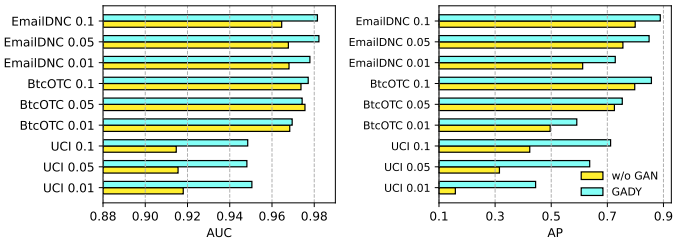

Fig. 2. Comparison between GADY and GADY w/o GAN.

The results show that GADY wins in most datasets except in the BitOTC dataset with a 5% anomaly rate and demonstrates the superior performance of GADY and its broad applicability. Results on the AP metric further widen the performance gap, which not only illustrates the effectiveness of the AP metric in measuring anomaly detection performance but also proves the excellent effect of GADY on anomaly detection. Therefore, we conclude that the GAN module is necessary for GADY model and can assist the model in improving detecting anomalies.

D. Encoder Ablation Study

We conduct experiments to explore the effects of the GAN module. Specifically, we compare GADY with GADY without Prioritization Temporal Aggregation (w/o agg) and GADY without Status Features (w/o stat) in all datasets with three different anomaly ratios. We implement GADY w/o agg and GADY w/o stat by abandoning the decay coefficient and status features. The results are shown in Figure 3.

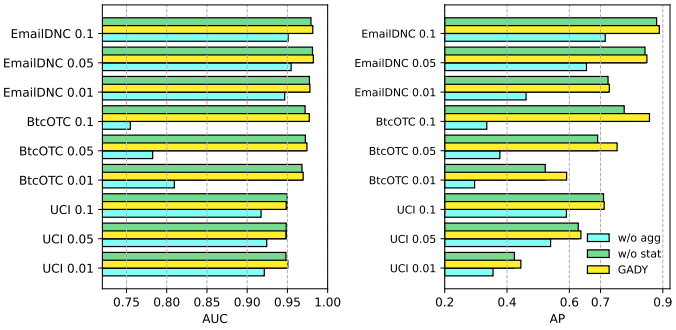

Fig. 3. Comparison between GADY and GADY w/o agg and GADY w/o stat.

The results show that GADY wins in almost all datasets on both the AUC and the AP metric. This experiment shows the importance of Prioritization Temporal Aggregation and Status Features parts, further validating our model design.

E. Performance of Generated Negative Edges

This experiment examines the generator's performance in generating anomalous samples. Specifically, we select the anomalous edges embedding obtained from the anomaly encoder in different batches in the second epoch and visualize the data distribution results as shown in Figure 4.

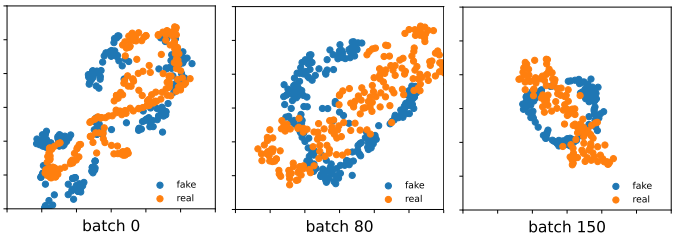

Fig. 4. Comparison of negative sample distributions generated in different periods in the second epoch. It can be seen that with the model's training, the generated negative samples gradually gather evenly near the real samples, proving that our generator can generate diverse and high-quality negative samples.

From the results, we find that with the continuous training of the model, the generated samples are gathered at the boundary of the real samples, which proves the high quality of the samples generated by the generator. The generated samples are evenly distributed around the real samples, demonstrating the diversity of samples generated by the generator. These results further demonstrate the effectiveness of our generator loss function design. This loss function successfully helps the generator output high-quality and diverse anomalous edges.

The high quality can be observed through the fact that fake samples are close to the real samples. The diversity can be observed through the fact that fake samples are around the real samples.

F. Time complexity analysis

In the comparison of time complexity, we conducted a comparative experiment, and the results are shown in Figure 5. The result shows that GADY is slower than TADDY overall, and the running time of each epoch is getting shorter due to the preprocessing time. However, the overall time cost is still within an acceptable range. It is worth mentioning that if the model of the encoder part is replaced with a lighter structure, the time overhead will be much lower while maintaining superior performance. The specific time complexity will be highly dependent on the model selected by the encoder part. Further time complexity analysis and performance analysis of the lighter structure model can be found in the appendix (https://anonymous.4open.science/r/GADY).

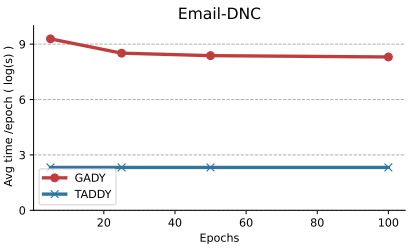

Fig. 5. Comparison of running time per epoch between GADY and TADDY.

VI. Conclusion and Discussion

In this paper, we discovered the shortcomings of existing anomaly detection on dynamic graph methods and proposed a novel continuous dynamic graph encoder for anomaly detection task. On this basis, we explored using the GAN model to generate high-quality and diverse negative samples. The final experimental results prove the excellent performance of the GADY in anomaly detection and its broad applicability. Supplementary experiments on the GAN ablation study, Encoder ablation study, visualization of generated samples, and time complexity analysis further prove the superiority of GADY and the validity of GADY settings. Besides, the current popular diffusion model shows its ability to surpass GANs, and we save the application of diffusion models to generate exceptions for future work.

References

[Cai et al.(2021)] Lei Cai, Zhengzhang Chen, Chen Luo, Jiaping Gui, Jingchao Ni, Ding Li, and Haifeng Chen. 2021. Structural temporal graph neural networks for anomaly detection in dynamic graphs. In *Proceedings of the 30th ACM international conference on Information & Knowledge Management*. 3747–3756.

[Dou et al.(2020)] Yingtong Dou, Zhiwei Liu, Li Sun, Yutong Deng, Hao Peng, and Philip S Yu. 2020. Enhancing graph neural network-based fraud detectors against camouflaged fraudsters. In *Proceedings of the 29th ACM International Conference on Information & Knowledge Management*. 315–324.

[Grover and Leskovec(2016)] Aditya Grover and Jure Leskovec. 2016. node2vec: Scalable feature learning for networks. In *Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining*. 855–864.

[Guo et al.(2022)] Qiuyu Guo, Xiang Zhao, Yang Fang, Shiyu Yang, Xuemin Lin, and Dian Ouyang. 2022. Learning Hypersphere for Few-shot Anomaly Detection on Attributed Networks. In *Proceedings of the 31st ACM International Conference on Information & Knowledge Management*. 635–645.

[Han and Yuan(2021)] Xiao Han and Shuhan Yuan. 2021. Unsupervised cross-system log anomaly detection via domain adaptation. In *Proceedings of the 30th ACM International Conference on Information & Knowledge Management*. 3068–3072.

[He and Garcia(2009)] Haibo He and Edwardo A. Garcia. 2009. Learning from Imbalanced Data. *IEEE Transactions on Knowledge and Data Engineering* (2009), 1263–1284.

[Huang et al.(2023)] Xuanwen Huang, Yang Yang, Yang Wang, Chunping Wang, Zhisheng Zhang, Jiarong Xu, Lei Chen, and Michalis Vazirgiannis. 2023. DGraph: A Large-Scale Financial Dataset for Graph Anomaly Detection. arXiv:2207.03579 [cs.SI] https://arxiv.org/abs/2207.03579

[Jin et al.(2021)] Ming Jin, Yixin Liu, Yu Zheng, Lianhua Chi, Yuan-Fang Li, and Shirui Pan. 2021. Anemone: Graph anomaly detection with multi-scale contrastive learning. In *Proceedings of the 30th ACM International Conference on Information & Knowledge Management*. 3122–3126.

[Khan et al.(2021)] Protima Khan, Md Fazlul Kader, SM Riazul Islam, Aisha B Rahman, Md Shahriar Kamal, Masbah Uddin Toha, and Kyung-Sup Kwak. 2021. Machine learning and deep learning approaches for brain disease diagnosis: principles and recent advances. *IEEE Access* 9 (2021), 37622–37655.

[Kumar et al.(2018)] Srijan Kumar, Bryan Hooi, Disha Makhija, Mohit Kumar, Christos Faloutsos, and V. S. Subrahmanian. 2018. REV2: Fraudulent User Prediction in Rating Platforms. *Proceedings of the Eleventh ACM International Conference on Web Search and Data Mining* (2018).

[Liu et al.(2021)] Yixin Liu, Shirui Pan, Yu Guang Wang, Fei Xiong, Liang Wang, Qingfeng Chen, and Vincent CS Lee. 2021. Anomaly detection in dynamic graphs via transformer. *IEEE Transactions on Knowledge and Data Engineering* (2021).

[Maron et al.(2019)] Haggai Maron, Heli Ben-Hamu, Hadar Serviansky, and Yaron Lipman. 2019. Provably powerful graph networks. *Advances in neural information processing systems* 32 (2019).

[Morris et al.(2019)] Christopher Morris, Martin Ritzert, Matthias Fey, William L Hamilton, Jan Eric Lenssen, Gaurav Rattan, and Martin Grohe. 2019. Weisfeiler and leman go neural: Higher-order graph neural networks. In *Proceedings of the AAAI conference on artificial intelligence*, Vol. 33. 4602–4609.

[Ngo et al.(2019)] Phuc Cuong Ngo, Amadeus Aristo Winarto, Connie Khor Li Kou, Sojeong Park, Farhan Akram, and Hwee Kuan Lee. 2019. Fence GAN: Towards better anomaly detection. In *2019 IEEE 31St International Conference on tools with artificial intelligence (ICTAI)*. IEEE, 141–148.

[Opsahl and Panzarasa(2009)] Tore Opsahl and Pietro Panzarasa. 2009. Clustering in weighted networks. *Social networks* 31, 2 (2009), 155–163.

[Paszke et al.(2019)] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. 2019. Pytorch: An imperative style, high-performance deep learning library. *Advances in neural information processing systems* 32 (2019).

[Perozzi et al.(2014)] Bryan Perozzi, Rami Al-Rfou, and Steven Skiena. 2014. Deepwalk: Online learning of social representations. In *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*. 701–710.

[Rossi et al.(2020)] Emanuele Rossi, Ben Chamberlain, Fabrizio Frasca, Davide Eynard, Federico Monti, and Michael Bronstein. 2020. Temporal graph networks for deep learning on dynamic graphs. *arXiv preprint arXiv:2006.10637* (2020).

[Rossi and Ahmed(2015)] Ryan Rossi and Nesreen Ahmed. 2015. The network data repository with interactive graph analytics and visualization. In *Proceedings of the AAAI conference on artificial intelligence*, Vol. 29.

[Schlegl et al.(2017)] Thomas Schlegl, Philipp Seeböck, Sebastian M Waldstein, Ursula Schmidt-Erfurth, and Georg Langs. 2017. Unsupervised anomaly detection with generative adversarial networks to guide marker discovery. In *Information Processing in Medical Imaging: 25th International Conference, IPMI 2017, Boone, NC, USA, June 25-30, 2017, Proceedings*. Springer, 146–157.

[Shone et al.(2018)] Nathan Shone, Tran Nguyên Ngoc, Vu Dinh Phai, and Qi Shi. 2018. A Deep Learning Approach to Network Intrusion Detection. *IEEE Transactions on Emerging Topics in Computational Intelligence* 2 (2018), 41–50.

[Tariq et al.(2022)] Shahroz Tariq, Binh M Le, and Simon S Woo. 2022. Towards an Awareness of Time Series Anomaly Detection Models' Adversarial Vulnerability. In *Proceedings of the 31st ACM International Conference on Information & Knowledge Management*. 3534–3544.

[Tian et al.(2023)] Sheng Tian, Jihai Dong, Jintang Li, Wenlong Zhao, Xiaolong Xu, Baokun Wang, Bowen Song, Changhua Meng, Tianyi Zhang, and Liang Chen. 2023. SAD: Semi-Supervised Anomaly Detection on Dynamic Graphs. *ArXiv* abs/2305.13573 (2023). https://api.semanticscholar.org/CorpusID:258841516

[Von Luxburg(2007)] Ulrike Von Luxburg. 2007. A tutorial on spectral clustering. *Statistics and computing* 17 (2007), 395–416.

[Xia et al.(2022)] Xuan Xia, Xizhou Pan, Nan Li, Xing He, Lin Ma, Xiaoguang Zhang, and Ning Ding. 2022. GAN-based anomaly detection: A review. *Neurocomputing* 493 (2022), 497–535. https://doi.org/10.1016/j.neucom.2021.12.093

[Xu et al.(2020)] Da Xu, Chuanwei Ruan, Evren Korpeoglu, Sushant Kumar, and Kannan Achan. 2020. Inductive representation learning on temporal graphs. *arXiv preprint arXiv:2002.07962* (2020).

[Xu et al.(2019)] Keyulu Xu, Weihua Hu, Jure Leskovec, and Stefanie Jegelka. 2019. How Powerful are Graph Neural Networks? arXiv:1810.00826 [cs.LG] https://arxiv.org/abs/1810.00826

[Ye and Akoglu(2015)] Junting Ye and Leman Akoglu. 2015. Discovering Opinion Spammer Groups by Network Footprints. *Proceedings of the 2015 ACM on Conference on Online Social Networks* (2015).

[Yu et al.(2018)] Wenchao Yu, Wei Cheng, Charu C Aggarwal, Kai Zhang, Haifeng Chen, and Wei Wang. 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*. 2672–2681.

[Zhao et al.(2021)] Tong Zhao, Bo Ni, Wenhao Yu, Zhichun Guo, Neil Shah, and Meng Jiang. 2021. Action sequence augmentation for early graph-based anomaly detection. In *Proceedings of the 30th ACM International Conference on Information & Knowledge Management*. 2668–2678.

[Zheng et al.(2019a)] Li Zheng, Zhenpeng Li, Jian Li, Zhao Li, and Jun Gao. 2019a. AddGraph: Anomaly Detection in Dynamic Graph Using Attention-based Temporal GCN.. In *IJCAI*, Vol. 3. 7.

[Zheng et al.(2019b)] Panpan Zheng, Shuhan Yuan, Xintao Wu, Jun Li, and Aidong Lu. 2019b. One-class adversarial nets for fraud detection. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 33. 1286–1293.

[Zhou et al.(2021)] Shuang Zhou, Qiaoyu Tan, Zhiming Xu, Xiao Huang, and Fu-lai Chung. 2021. Subtractive aggregation for attributed network anomaly detection. In *Proceedings of the 30th ACM International Conference on Information & Knowledge Management*. 3672–3676.