GADY: Unsupervised Anomaly Detection on Dynamic Graphs

Anonymous submission

Abstract

Anomaly detection on dynamic graphs refers to detecting entities whose behaviors are obviously different from other normal entities on dynamic graphs. This field has drawn increasing attention due to its application in finance, network security, social network, and more. However, existing methods face two challenges: 1. difficulties in capturing graph structure with complex time information. 2. lack of genuine abnormal samples on dynamic graphs. To address these challenges, we propose Unsupervised **G**enerative **A**nomaly Detection on **Dy**namic Graphs (**GADY**). For the first challenge, we pioneer to use a **continuous dynamic graph** model to capture the fine-grained information, which breaks the limit of existing methods. Specially, we employ a message-passing framework combined with positional features to get edge embedding, which can be decoded as an anomaly or normal. For the second challenge, we pioneer Generative Adversarial Networks to generate negative interactions and design a loss function to alter the training goal of the generator while ensuring the diversity and quality of generated samples. Extensive experiments demonstrate that our proposed `GADY` significantly outperforms the previous 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 module.

Introduction

Anomaly detection on dynamic graphs, is an important task with 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). This task can help us better understand and capture 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. For example, in financial transactions, if two entities conduct a transaction and they are in two transaction collectives that had no transaction records before, this transaction is very suspicious.

Although many methods have made lots of progress, they are weak in two aspects: 1. difficulties in capturing graph structure with complex time information. 2. unable to construct excellent negative samples for unsupervised learning. For the first challenge, Anomaly detection usually needs entity embedding through the encoder part, which are then decoded as labels. Some people proposed to use discrete dynamic methods for modeling(Cai et al. 2021; Liu et al. 2021). Specially, they cut dynamic graphs into several slices, which can be captured by Graph Neural Networks (GNNs) or similar methods. Then they use Gated Recurrent Unit (GRU) and similar methods to construct the evolutionary process of slices (Zheng et al. 2019a; Cai et al. 2021; Liu et al. 2021). The strength of this kind of method is able to use well-developed GNNs methods to boost the model performance. However, using this method method inevitably results in two kinds of information loss: One is ignoring the repeated edges in the slice time interval, which will cause GNN to fail to capture complete interaction information, making the generated entity embedding defective, thereby reducing the model's performance. The other is ignoring the time difference within the same slice, making it difficult for the model to capture time dependencies, thereby also reducing the performance of the model.

In order to overcome the shortcomings of discrete dynamic graphs, we propose to use another method - the continuous dynamic graph method to model dynamic graphs. This method processes incoming edges individually in chronological order without missing edges or time information, thus overcoming the shortcomings of discrete dynamic methods.

Figure 1 gives an example to illustrate the superiority of continuous dynamic graph anomaly detection. Through discrete dynamic methods, original graph is modeled as many snapshots like sub-figure 2. During the modeling process, the edges at time 10.22 and 22.51 are modeled as one edge, resulting in information loss, which reduces the encoding quality and model accuracy. Also, all timestamps are labeled as 'T', causing all edges to be similar to other edges, and no abnormality can be found. However, through the continuous dynamic method, original graph is modeled as edges stream, and both edge and time information will be fully preserved. Here we divide them into two sub-figures for the convenience of analysis. Through the modeling results, it can be found that the behavior pattern of edge at 58:20 in the second half hour is different from its surrounding environment, which is easier to be judged as anomalous.

In addition to above problem, research on anomaly detection on dynamic graphs also faces the second challenge: unable to construct excellent negative samples for unsu-

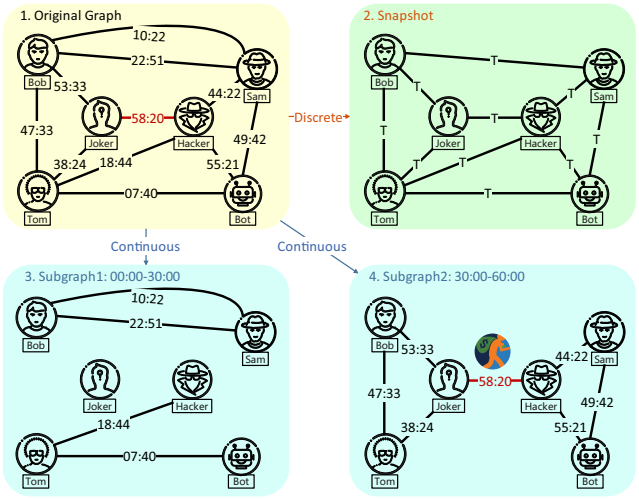

Figure 1: Example of the superiority of anomaly detection in a continuous dynamic graph. In this Figure, the dynamic graph is the original graph. The edge of 58:20 is the anomaly edge that should be detected. Discrete dynamic graph methods fall short of detecting this anomaly, while continuous dynamic methods can easily distinguish the anomaly behavior.

pervised learning, which is, on the existing dynamic graph datasets, there are no real anomalous samples to help us conduct supervised training. To solve this problem, some people propose a context-dependent negative sampling method to artificially construct negative samples by rules (Cai et al. 2021) for unsupervised learning. Specifically, for each edge, they randomly keep the head node or tail node, and replace the unreserved nodes with other irrelevant nodes, thus obtaining negative samples for training. However, the quality of negative samples in the training phase has a crucial impact on the model performance. Ideally, the generated negative samples will enable the model to efficiently learn diverse anomaly patterns during the training phase. This puts forward two requirements for negative samples: 1. Abnormal samples need to be of high quality; 2. Abnormal samples need to be diverse. The proposed method has limited performance in these two criteria.

To break above two limits, we propose a general model for continuous dynamic graph anomaly detection. Our main idea is to use anomaly generator to obtain diverse and high-quality negative samples to help training, and to use a continuous dynamic graph model to perform anomaly detection. Through the adversarial training of the two modules, an excellent discriminator can finally be obtained, thus achieving better performance on anomaly detection. Specifically, to break the limitation of the existing discrete dynamic graph detection methods, we use message-passing continuous dynamic models combined with positional features (Souza et al. 2022) to capture complex dynamic graph. Also, we use anomaly generator to generate diverse negative edges with high quality. To achieve this, we propose a novel loss function designed for dynamic graphs anomaly detection to help generator find its specific training goal. In experiments, we test our model by using the same evaluation method as (Liu et al. 2021) on three datasets, and outperforms the state-of-the-art method significantly, thus proving the superiority of our model. We also investigate the potential of current continuous dynamic graphs for anomaly detection. Detailed supplementary experiments about GAN module and generator design also demonstrate the necessity of anomaly generator and the superiority of our model.

In summary, our main contributions are as follows:

- We identify the drawbacks of existing discrete dynamic graph anomaly detection methods, and address it using continuous dynamic graph methods. We further explore the potential of continuous dynamic graph models to address existing anomaly detection problems.
- We pioneer GAN network for anomaly detection on dynamic graphs and demonstrate the effectiveness of the GAN model in improving anomaly detection capabilities.
- Extensive experiments on three datasets shows 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 anomaly generators and the superiority of model settings.

Related Work

Anomaly Detection on Dynamic Graph

In recent years, anomaly detection focus on using deep learning methods to model the entire process. (Jin et al. 2021; Zhao et al. 2021; Tariq, Le, and Woo 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 module to construct long and short term relationships between slices. *StrGNN* (Cai et al. 2021) extracted h-hop closed subgraphs centered 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 transformer to process diffusion-based spatial encoding, distance-based spatial encoding and relative time encoding, and then obtained edge representation through pooling layer, thus obtaining anomaly score. For continuous methods, it can be more helpful for anomaly detection with its strong modeling capabilities and sufficient time information. Recently, *SAD* (Tian et al. 2023) propose to use continuous dynamic methods to detect anomalies in semi-supervised method, which is different from our work.

Generative Adversarial Network in Anomaly Detection

Nowadays, using GAN to perform anomaly detection mainly falls into two branches. The first branch is to use

GAN to perform normal sample distribution and use reconstruction error as an anomaly score. For example, *AnoGAN* (Schlegl et al. 2017) can capture the latent distribution of normal data through training, and because the behavior patterns of abnormal data 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 image.

However, since the original goal of GAN is to generate high-quality data rather than anomaly detection, the anomaly score obtained by using reconstruct loss of GAN for anomaly detection may be suboptimal. Another branch is to use GAN to generate negative samples to assist model training. For example, *OCAN* (Zheng et al. 2019b) uses LSTM-Autoencoder to learn the representation of normal users, and then modifies the generator so that its training goal becomes generating abnormal data that is complementary to normal data, and obtains a better discriminator after training. *Fence-GAN* (Ngo et al. 2019) improves the model training effect by defining different loss functions so that the generated samples are located at the edge of normal samples distribution.

Although our model has similar ideas with *OCAN*, we use generator to get interactions directly rather than embeddings with different generator structure for its application in unattributed dynamic graphs. Also, *Fence-GAN* (Ngo et al. 2019) focus on anomaly detection in images. We design different loss function for its application in dynamic graphs.

Methodology

Given a dynamic graph $G = \{e_{uv}(t) = (u, v, t) : u, v \in \mathcal{V})\}$, our goal is to identify a fake interaction $e_{u'v'}(t') = (u', v', t')$ which is different from other edges. To achieve this goal, we design GADY, whose structure is shown in Figure 2. Our model has two parts: anomaly generator G and discriminator D. For the anomaly generator, it accepts the input noise $z_i \in (-1, 1)$ and output the fake interactions $G(z_i)$, where G is the generator function. This function is trained through a loss function $L_G(G, D, Z_i)$ where D is the discriminator function. For the discriminator, it contains two parts: encoder and decoder. For the part of encoder, it is aimed to train a function $E()$, where $E(e_{uv}(t))$ is the output edge embedding. For the part of decoder, it inputs the edge embedding and outputs the anomaly score of the interaction between these two nodes, and then judges whether the edge is an anomalous edge based on the anomaly score. This process can be formulated as $F(emb_u, emb_v)$ where $u, v \in \mathcal{V}$. For the whole discriminator, it can be formulated as $D(e_{uv}(t)) = F(E(e_{uv}(t)))$. Also, the loss function of the discriminator is $L_D(G(z_i), x_i, D)$ where $G(z_i)$ and x_i are the generated samples and real samples, respectively.

Generator

Framework Design The purpose of the 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 not exactly the same as real samples. Such negative samples can make the model work hard to distinguish true and false samples during the training phase. 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 random noise such as Gaussian noise or Uniform noise. For the type of input noise, through our experiment, we find that different types of noise are suitable for different datasets. In our model, we use gaussian noise for its superiority. For the generator structure, an immediate question is whether the generator should produce edges or edge embedding. Different from *OCAN* (Zheng et al. 2019b), Here we have proved through experiments that generating edges is more conducive to the effect of the model 5, which may be because ignoring the embedding process of encoder for interactions, resulting in bias and sub-optimal outcomes. Moreover, generating fake edges on dynamic graphs still faces some problems. The first is that the edges come in chronological order, and for each batch, the timestamps of the generated edges should be within this range, so as to make the negative samples more similar to the real samples. For the head node and tail node of the edge, we guarantee that the randomly generated nodes are within dataset. Based on the above considerations, we define the model as follows

$$G(z_i) = R(\tanh(f(z_i))) \tag{1}$$

where f is defined as a 3-layer multilayer perceptrons and z_i is the input 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 input form. Through the processing of multilayer perceptrons, noise is reconstructed into input embedding. Then, we reconstruct the output of the generator to the form of (u, v, t), and then output the anomaly score and judgment through the encoding-decoding process of the discriminator.

Loss Function design The design of the loss function usually affects the training goal of the model. In order to ensure that the model is trained according to the above two principles, we define the following loss function:

$$\begin{aligned} L_G(G, D, z_i) = \frac{1}{N}\sum_{i=1}^{N} \log(|\alpha - \mathrm{D}(\mathrm{G}(\mathrm{z_i}))|) \\ + \beta \times \frac{1}{\frac{1}{n}\sum_{k=1}^{n}(CL^k)} \end{aligned} \tag{2}$$

In this function, L_G is the loss function for the generator, and G and D refer to generator and discriminator respectively. The first part is to ensure the quality of generated samples. $\alpha \in [0, 1]$ is expected anomaly score of generated samples from discriminator. Setting α to 0.1 means that the generator hopes that the samples generated by itself will get scores of 0.1 from the discriminator, meaning that although they are judged as normal samples, they still has certain anomalous patterns. By tuning this parameter, we can control the quality of negative samples generated by the generator to adapt

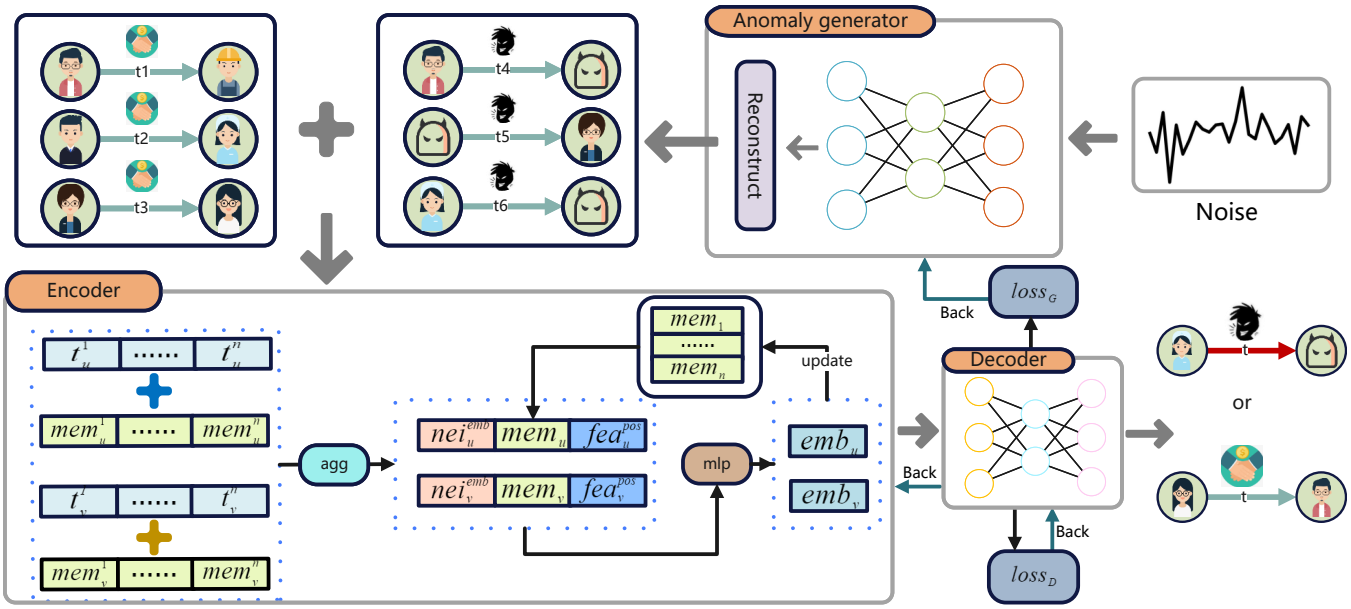

Figure 2: The architecture of GADY network includes two parts. The first part is the anomaly generator, which generates abnormal interactions through input noise, sorts the generated interactions in time, and inputs them into the encoder part together with the normal interactions. The second part is the discriminator, which contains encoder and decoder. After getting the interactions, encoder obtains the edge embedding by continuously aggregating the information of time neighbors (R-agg) for the head node and the tail node. Finally, the embedding of edge is fed into the decoder. For the decoder, it obtains the anomaly score of the edge through the representation of the head node and the tail node, finally judging whether the target edge is anomalous or normal.

to different datasets. For the second part, to judge the diversity of generated edges on dynamic graphs, we use the coefficient of variation that are different from *Fence-GAN* (Ngo et al. 2019). β shows how concerned we are about the diversity generated samples. Setting b to 10 means that we put more emphasis on the diversity of data in the loss function, which will prompt the generator to produce more diverse samples to help the discriminator to train. CL^k refers to the coefficient of variation on the K-th dimension. Finally we uses the reciprocal of the average value of the three dimensions of CL^k to measure the diversity of the data. And CL^k can be formulated as follows:

$$CL^k = \frac{\frac{1}{N}\sum_{i=1}^{N}(||G(z_i^k) - \mu^k||_2)}{\mu^k} \tag{3}$$

where CL^K is the $k-dim$ of the interaction (u, v, t), measuring the diversity of data in k-th dimension. And k is in $[1, 3]$ Through the design of this module, we can successfully generate high-quality and diverse negative samples, which lays a solid foundation for the superior performance of our model.

Discriminator

The purpose of the discriminator is to distinguish normal and abnormal edges from the input edges. This requires that it can capture complex continuous dynamic graph structures. Nowadays, the most powerful framework is the encoder-decoder structure. And our model design follows this architecture. where the two modules are defined as follows.

Encoder The performance of anomaly detection models usually depends heavily on the encoding ability of the discriminator. In existing continuous dynamic graph models, *TGNS* (Rossi et al. 2020) is a popular model. Based on this model, *PINT* (Souza et al. 2022) analyzes the inherent limitations of existing messaging models and proposes to use injective temporal aggregation and positional features to overcome the limitations of temporal Weisfeiler-Leman test (Maron et al. 2019; Morris et al. 2019). Although existing dynamic graph models are mainly designed for link prediction tasks, models that capture dynamic graph structures are still applicable to anomaly detection tasks because they can successfully capture complex dynamic graph structures. In our task, how to build a more excellent continuous dynamic graph model is not the focus of this paper. Based on above considerations, we use the encoder structure as *PINT* to capture complex continuous dynamic graphs.

Specifically, before the model starts training, we first calculate how many temporal paths there are between two nodes, and take this as a relative position feature into consideration in the message passing process. The detailed computing process can be found in *PINT* (Souza et al. 2022)

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

$$E(e_{uv}(t)) = concat(emb(u, t), emb(v, t)) \tag{4}$$

$$emb(u,t) = \sum_{j \in \eta_u^k([0,t])} h(s_j(t), e_{uj}, r_{j\to u}^{(t)}) \quad (5)$$

where h is a learnable function that can be designed with different forms. $s_v(t)$ are memory of v. e_{uj}, $v_u(t)$, and $v_v(t)$ is the attribute of edge and nodes, which can be set to zero if missing. By recursively aggregating the information of k-layers of neighbors, we can finally get the edge embedding.

To keep updating memory, we use the following process: For each interaction (u, v, t), we first find temporal neighbors for head and tail and then get messages from them. This process can be defined by the following formula:

$$s_u(t) = upd(s_u(t^-), s_v(t^-), \Delta t, e_{uv}(t)) \quad (6)$$

where upd is a learnable function that has different implementing ways. $s_u(t^-)$ is the memory of node u just before time t. Although function h and upd can be implemented in different forms, to ensure the performance of our detecting model, we follow the settings as *PINT* in practice.

Although link prediction and anomaly detection have limited generality in modeling continuous dynamic graphs, the different training objectives between link prediction and anomaly detection still lead to inability to do exactly the same in model settings. In fact, in order to complete the task of link prediction, existing dynamic graph models usually follow the framework of unsupervised training, which helps model training by constructing negative samples in the training phase, and performs model evaluation through negative sampling in the testing phase. However, different training objectives and evaluation methods usually require different construction methods of negative samples with different characteristics during model training for its application in anomaly detection. Directly applying training methods designed for link prediction in anomaly detection may lead to suboptimal results as shown in 1.

Decoder The purpose of the decoder part is to generate anomaly score from the edges embedding and make judgments. Although this part can be designed to be very complex, for simplicity, we define the model as follows:

$$F(e) = Linear(ReLU(Linear(z(e)))) \quad (7)$$

Where e is the input edge embedding from encoder. In our model, we use the following function as the overall error of the discriminator.

$$L_D(G(z_i), x_i, D) = \frac{1}{N}\sum_{i=1}^{N}[-\gamma \log(\mathrm{D}(\mathrm{G}(\mathrm{z_i}))) - log(1 - D(x_i)) \quad (8)$$

Where $\gamma > 0$ is a hyperparameter determining we put how much attention on abnormal edges. When γ is lower than 1, we put more attention on normal edges and less emphasis on anomalous edges. When γ is greater than 1, we put more attention on anomalous edges and less emphasis on normal edges. By adjusting this hyperparameter, GADY can adapt to various datasets.

Algorithm 1: Training process of GADY

```
Preprocess: calculate the positional features r^(t)_(i→j) of each node i relative to another node j.
repeat
    Input noise z_i to generator G and get fake interactions
    Input true and fake interactions to Discriminator D
    For each interaction (u, v, t)
    For each layer l :
        Aggregate message from temporal neighbors
    Get embedding of two nodes u and v
    Concatenate to get edge embedding E(e_u v(t))
    Decode to get judgement D(e_u v(t))
    Update memory
    Pass back L_G and L_D
until End
```

Training process

In the training process, firstly, the generator receives noise and outputs the generated pseudo edges, then the pseudo edges and the real edges are input into the encoder for encoding to obtain the edges embedding, and finally the edges embedding is decoded into the labels to determine as anomaly or not. Finally, the loss of the generator L_G and the loss of the discriminator L_D passed back to update parameters. This process can be found in Algorithm 1.

Experiments

Experiment Settings

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

AP (Average Precision) is a metric for evaluating the performance of object detection models on a single class. It is based on the area under the Precision-Recall (PR) curve, which shows the precision and recall of the model at different thresholds.

Different from 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. 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 our follow-up experiments, we also proved the superior performance of our model on this evaluating metric.

Datasets To test the performance of our model, we evaluated our framework on three datasets: **UCI Message**[1] (Opsahl and Panzarasa 2009), **Email-DNC**[2] (Rossi and Ahmed 2015), and **Bitcoin-OTC**[3] (Kumar et al. 2018).

[1] http://konect.cc/networks/opsahl-ucsocial

[2] http://networkrepository.com/email-dnc

[3] http://snap.stanford.edu/data/soc-sign-bitcoin-otc

Table 1: 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 second best performance is marked with underline. Model 1%, 5%, 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
PINT	0.9265	0.9232	0.9229	0.9227	0.9226	0.9225	0.9758	0.9754	0.9783
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 (No-GAN)	**0.9658**	**0.9655**	**0.9656**	**0.9826**	0.9836	0.9828	0.9636	0.9696	0.9648
GADY (GAN)	0.9600	0.9585	0.9597	0.9819	**0.9854**	**0.9839**	**0.9796**	**0.9835**	**0.9827**
TADDY(AP)	0.1760	0.3428	0.4743	0.1402	0.4287	0.5995	0.1046	0.2854	0.3986
GADY (No-GAN)(AP)	0.4811	0.4801	0.4784	0.6105	0.6093	0.6221	0.6149	0.6617	0.6173
GADY (GAN)(AP)	**0.5025**	**0.6771**	**0.7617**	**0.6842**	**0.8395**	**0.8797**	**0.7462**	**0.8583**	**0.8861**

These datasets are constructed from data from forum posts, emails, and trading systems respectively, details of which can be found in the appendix.

Baselines We selected four classes of models as our baselines: traditional non-deep learning methods *Node2vec* (Grover and Leskovec 2016) *Spectral Clustering* (Von Luxburg 2007) *DeepWalk* (Perozzi, Al-Rfou, and Skiena 2014), methods using a combination of deep and non-deep learning *NetWalk*,(Yu et al. 2018) continuous dynamic graph modeling methods directly used for anomaly detection *TGNs* (Rossi et al. 2020) *PINT* (Souza et al. 2022), and anomaly detection methods using deep learning *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.

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

In our experiments, the message function use the identity function used in *TGN* (Rossi et al. 2020) Number of layers is set to 2 and the training ratio is set to 0.5. Other detailed implement details can be found in appendix.

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.

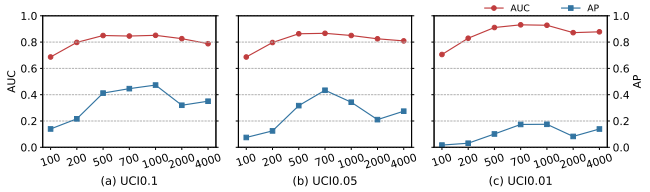

Figure 3: Effects of Slice Size on detection performance. Too small or too large slice size will limit model performance. So it is difficult for discrete dynamic methods to capture fine-grained time information, which is an fatal flaw of this kind of method and validate the superiority of continuous dynamic methods.

Main Results

We tested our model on three datasets and set the injected anomaly ratio to 1%, 5%, and 10% respectively. We also show the results without using the GAN model (by using the same training method as TADDY to train the discriminator), and the results of the complete model using GAN. The experimental results are shown in Table 1. From the experimental results, we summarize the following conclusions:

- Our model has excellent results, and the use of the GAN module can greatly improve the ability of the model to detect anomalies. Our model outperforms state-of-the-art method by 14.6% at most on the UCI dataset with anomaly ratio of 10%, achieving SOTA on three real world datasets, and the use of the GAN module can greatly improve the ability of the model to detect anoma-

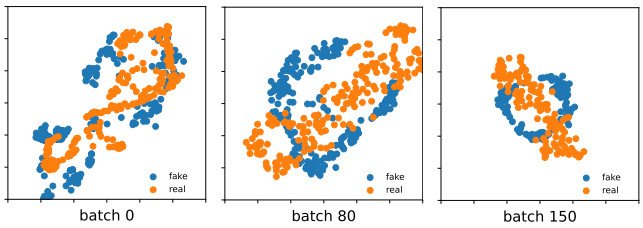

Figure 4: Comparison of negative sample distributions generated in different periods in the second epoch.It can be seen that with the training of the model, the generated negative samples gradually gather evenly near the real samples, which proves that our generator can indeed generate diverse and high-quality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

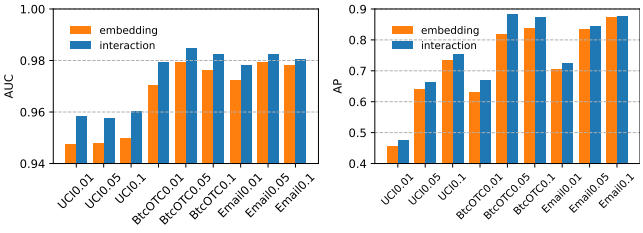

Figure 5: Comparison between generating interactions or embeddings. The results shows that generating interactions has better performance than generating embeddings directly.

lies which is shown in AP metric.

- Continuous dynamic graph models have remarkable advantages over discrete dynamic graphs. It can be seen from the effect of the GADY model that does not use GAN that it significantly surpasses the ordinary discrete dynamic graph, which verifies our idea.
- Our model performance is insensitive to anomaly ratios. From the table, it can be found that other models are more sensitive to abnormal injection ratio. However, our method remains a high performance regardless of what the anomaly ratio is.This will allow our model to have a wider range of applicability to different anomaly ratios.
- The idea of directly using continuous dynamic graph models for anomaly detection tasks is not enough. From the table, we can find that the detection effects of *TGN* (Rossi et al. 2020) and *PINT* (Souza et al. 2022) are similar to the traditional anomaly detection model TADDY, but still have a large distance compared with our model. This also verifies our conjecture: although continuous dynamic graph models for different tasks are all designed for capturing the dynamic graph structure, the model for link prediction tasks cannot be directly applied to anomaly detection. It needs to use different negative sampling methods in order to achieve better results.

Slice Size Study

In order to further validate the superiority of continuous dynamic graph anomaly detection compared to descrete graph processing, we adjust the slice size of to 100, 300, 500, 700, 1000, 1500, 2000, and 5000 respectively, and test *TADDY* (Liu et al. 2021) on the UCI dataset with anomaly rate 1%, 5%, 10%. The experimental results are shown in Figure 3.

Some people may say that a discrete form of dynamic graph can capture more fine-grained temporal information by increasing the number of slices, but this is unrealistically verified in this experiment.

From the experimental results, no matter whether the slice size is large or small, the model effect cannot reach the best. When the slice size is too small, it will be difficult for GNNs to capture the structural information on the slice; when the slice size is too large, too much time information will be ignored, thus limiting the anomaly detection effect of discrete dynamic graphs.

Performance of Generated Negative Edges

This experiment is to examine the performance of the generator on anomalous samples. Specifically, we select the anomalous samples generated in different batches in the second epoch, and visualized the data distribution results as shown in the Figure 4. Form 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, and the generated samples are evenly distributed around the real samples, which demonstrates the diversity of samples generated by the generator.

Generate edges or edge Embedding

In this experiment, we test the model performance of using the generator to directly generate edge embedding or edge interactions, which is shown in Figure 5. From the results, we can find that no matter which dataset and what the abnormality ratio is, the results of directly generating interactions are always better than those directly generating edge embedding. This may be because the method of directly generating edge encoding ignores the encoder's process of modeling the interaction, thus causing more bias. Therefore, we can conclude that using a generator to generate interactions is a better choice than directly generating edge representations.

Conclusion

In this paper, we discovered the shortcomings of existing discrete dynamic graphs for anomaly detection, and proposed that continuous dynamic graphs should be used for anomaly detection tasks. Also, our study discovers how existing continuous dynamic graphs for link prediction should perform on anomaly detection tasks. On this basis, we explored how to use the GAN model to obtain more high-quality and diverse samples. The final experimental results not only prove the excellent performance of the continuous dynamic graph in anomaly detection, but also prove the role of the GAN module in helping the model identify anomalies. Supplementary experiments on Slice Size, visualization of generated samples, generated edges or edge embedding further prove the shortcomings of discrete dynamic methods, the superiority of GAN to generate negative samples, and the superiority of model settings.

References

Cai, L.; Chen, Z.; Luo, C.; Gui, J.; Ni, J.; Li, D.; and Chen, H. 2021. Structural temporal graph neural networks for anomaly detection in dynamic graphs. In *Proceedings of the 30th ACM international conference on Information & Knowledge Management*, 3747–3756.

Dou, Y.; Liu, Z.; Sun, L.; Deng, Y.; Peng, H.; and Yu, P. S. 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, A.; and Leskovec, J. 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, Q.; Zhao, X.; Fang, Y.; Yang, S.; Lin, X.; and Ouyang, D. 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, X.; and Yuan, S. 2021. Unsupervised cross-system log anomaly detection via domain adaptation. In *Proceedings of the 30th ACM International Conference on Information & Knowledge Management*, 3068–3072.

Jin, M.; Liu, Y.; Zheng, Y.; Chi, L.; Li, Y.-F.; and Pan, S. 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, P.; Kader, M. F.; Islam, S. R.; Rahman, A. B.; Kamal, M. S.; Toha, M. U.; and Kwak, K.-S. 2021. Machine learning and deep learning approaches for brain disease diagnosis: principles and recent advances. *IEEE Access*, 9: 37622–37655.

Kumar, S.; Hooi, B.; Makhija, D.; Kumar, M.; Faloutsos, C.; and Subrahmanian, V. S. 2018. REV2: Fraudulent User Prediction in Rating Platforms. *Proceedings of the Eleventh ACM International Conference on Web Search and Data Mining*.

Liu, Y.; Pan, S.; Wang, Y. G.; Xiong, F.; Wang, L.; Chen, Q.; and Lee, V. C. 2021. Anomaly detection in dynamic graphs via transformer. *IEEE Transactions on Knowledge and Data Engineering*.

Maron, H.; Ben-Hamu, H.; Serviansky, H.; and Lipman, Y. 2019. Provably powerful graph networks. *Advances in neural information processing systems*, 32.

Morris, C.; Ritzert, M.; Fey, M.; Hamilton, W. L.; Lenssen, J. E.; Rattan, G.; and Grohe, M. 2019. Weisfeiler and leman go neural: Higher-order graph neural networks. In *Proceedings of the AAAI conference on artificial intelligence*, volume 33, 4602–4609.

Ngo, P. C.; Winarto, A. A.; Kou, C. K. L.; Park, S.; Akram, F.; and Lee, H. K. 2019. Fence GAN: Towards better anomaly detection. In *2019 IEEE 31St International Conference on tools with artificial intelligence (ICTAI)*, 141–148. IEEE.

Opsahl, T.; and Panzarasa, P. 2009. Clustering in weighted networks. *Social networks*, 31(2): 155–163.

Paszke, A.; Gross, S.; Massa, F.; Lerer, A.; Bradbury, J.; Chanan, G.; Killeen, T.; Lin, Z.; Gimelshein, N.; Antiga, L.; et al. 2019. Pytorch: An imperative style, high-performance deep learning library. *Advances in neural information processing systems*, 32.

Perozzi, B.; Al-Rfou, R.; and Skiena, S. 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, E.; Chamberlain, B.; Frasca, F.; Eynard, D.; Monti, F.; and Bronstein, M. 2020. Temporal graph networks for deep learning on dynamic graphs. *arXiv preprint arXiv:2006.10637*.

Rossi, R.; and Ahmed, N. 2015. The network data repository with interactive graph analytics and visualization. In *Proceedings of the AAAI conference on artificial intelligence*, volume 29.

Schlegl, T.; Seeböck, P.; Waldstein, S. M.; Schmidt-Erfurth, U.; and Langs, G. 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*, 146–157. Springer.

Shone, N.; Ngoc, T. N.; Phai, V. D.; and Shi, Q. 2018. A Deep Learning Approach to Network Intrusion Detection. *IEEE Transactions on Emerging Topics in Computational Intelligence*, 2: 41–50.

Souza, A.; Mesquita, D.; Kaski, S.; and Garg, V. 2022. Provably expressive temporal graph networks. *Advances in Neural Information Processing Systems*, 35: 32257–32269.

Tariq, S.; Le, B. M.; and Woo, S. S. 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, S.; Dong, J.; Li, J.; Zhao, W.; Xu, X.; Wang, B.; Song, B.; Meng, C.; Zhang, T.; and Chen, L. 2023. SAD: Semi-Supervised Anomaly Detection on Dynamic Graphs. *ArXiv*, abs/2305.13573.

Von Luxburg, U. 2007. A tutorial on spectral clustering. *Statistics and computing*, 17: 395–416.

Ye, J.; and Akoglu, L. 2015. Discovering Opinion Spammer Groups by Network Footprints. *Proceedings of the 2015 ACM on Conference on Online Social Networks*.

Yu, W.; Cheng, W.; Aggarwal, C. C.; Zhang, K.; Chen, H.; and Wang, W. 2018. Netwalk: A flexible deep embedding approach for anomaly detection in dynamic networks. In *Proceedings of the 24th ACM SIGKDD international conference on knowledge discovery & data mining*, 2672–2681.

Zhao, T.; Ni, B.; Yu, W.; Guo, Z.; Shah, N.; and Jiang, M. 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, L.; Li, Z.; Li, J.; Li, Z.; and Gao, J. 2019a. AddGraph: Anomaly Detection in Dynamic Graph Using Attention-based Temporal GCN. In *IJCAI*, volume 3, 7.

Zheng, P.; Yuan, S.; Wu, X.; Li, J.; and Lu, A. 2019b. One-class adversarial nets for fraud detection. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, 1286–1293.

Zhou, S.; Tan, Q.; Xu, Z.; Huang, X.; and Chung, F.-l. 2021. Subtractive aggregation for attributed network anomaly detection. In *Proceedings of the 30th ACM International Conference on Information & Knowledge Management*, 3672–3676.