Unsupervised Anomaly Detection in Dynamic Graphs via Compatibility Modeling and Boundary Learning

Jiachi Luo[1] , **Shameng Wen**[2] , **Ziyang Qiu**[1] , **Chen Jiang**[1] , **Qi Huang**[1] ,
Yuxing Tian[3] and **Aiwen Jiang**[1*]
[1]Jiangxi Normal University
[2]Hangzhou Dianzi University
[3]Université de Montréal
{202126202006,202329001047,202429001047,huangqi,jiangaiwen}@jxnu.edu.cn,
wenshameng@126.com, tianyxx@gmail.com

Abstract

Anomaly detection in dynamic graphs is essential for monitoring evolving systems such as transaction networks and online platforms. Yet existing methods remain limited in realistic edge-stream settings: snapshot-based approaches discretize continuous interactions and miss fine-grained temporal signals, while many continuous-time models rely on scarce node/edge attributes and often fail to explicitly assess whether a destination is compatible with a source's recent context. Moreover, under extreme class imbalance and the lack of anomaly labels, unsupervised detectors frequently yield ill-defined normality criteria and ambiguous decision boundaries. We propose **BAD**, an unsupervised framework for anomaly detection in continuous-time dynamic graphs. BAD adopts a minimalistic design that represents nodes with learnable identity embeddings and performs pairwise compatibility modeling via cross-attention between each destination node and the source's recent neighbors, enabling direct characterization of context-dependent deviations without requiring attributes. To obtain a principled separating boundary, BAD further integrates normalizing flows to model the distribution of normal interactions and derive likelihood-based anomaly scores. Extensive experiments on four real-world datasets demonstrate that BAD consistently performs well, highlighting its robustness under feature-scarce and label-scarce conditions.

1 Introduction

Anomaly detection in graphs is a fundamental task that aims to identify instances that deviate substantially from expected patterns, with critical applications in fraud detection [Zhang *et al.*, 2021], cybersecurity [Zhang *et al.*, 2022], and infrastructure monitoring [Berger-Wolf and Saia, 2006; Greene *et al.*, 2010]. While extensive prior work has focused on static graphs [Peng *et al.*, 2018; Ding *et al.*, 2019; Fan *et al.*, 2020; Yuan *et al.*, 2021; Xu *et al.*, 2022; Tang *et al.*, 2023], the continuous evolution of real-world networks has spurred growing interest in dynamic settings. Existing studies on dynamic graph anomaly detection [Yu *et al.*, 2018; Liu *et al.*, 2023; Zheng *et al.*, 2019; Li *et al.*, 2023; Cai *et al.*, 2021] predominantly adopt the discrete-time dynamic graph (DTDG) formulation, modeling graph evolution as a sequence of snapshots sampled at fixed intervals. However, many practical systems generate continuous interaction events, where timely detection is crucial to mitigate downstream impact. The snapshot-based paradigm inevitably introduces temporal discretization error and detection latency, and it may obscure fine-grained temporal dependencies within each interval. These limitations motivate edge-stream modeling with incremental computation, enabling constant-time anomaly evaluation per event.

Although many studies [Eswaran and Faloutsos, 2018; Bhatia *et al.*, 2020; Chang *et al.*, 2021; Bhatia *et al.*, 2023] have proposed edge-stream anomaly detectors, these methods are typically rule-based and tailored to specific anomaly types (e.g., burstiness), which limits their ability to capture more complex anomalies. More recently, learning-based continuous-time dynamic graph models have been introduced for dynamic graph anomaly detection, but they still face several challenges: **(1) Feature scarcity:** Real-world edge streams often contain little information beyond timestamps, as node and edge attributes are frequently missing or extremely sparse. This is particularly detrimental in anomaly detection, where anomalies are rare and supervision is limited: models must distinguish a small number of abnormal nodes from a large normal population using only subtle interaction patterns. Under such feature-scarce conditions, timestamp-only signals tend to produce weakly discriminative representations, making abnormal and normal nodes difficult to separate. **(2) Inadequate interaction modeling:** Many anomalies are inherently context-dependent: an interaction may be anomalous not because it is temporally infrequent, but because the destination node is incompatible with the source node's recent context. However, most existing methods encode source and destination nodes independently and then estimate edge likelihood via an MLP, which only implicitly captures source–destination compatibility and

*Corresponding author: Aiwen Jiang.

is therefore less effective at identifying such mismatched interactions. **(3) Ambiguous decision boundary:** In practice, anomalous events are orders of magnitude rarer than normal interactions, leading to severe class imbalance. Since reliable anomaly labels are difficult to acquire, most existing methods [Yu *et al.*, 2018; Cai *et al.*, 2021; Liu *et al.*, 2023; Postuvan *et al.*, 2024; Yang *et al.*, 2024] adopt an unsupervised setting, learning normal patterns from normal or unlabeled data and flagging deviations as anomalies. Yet in the absence of explicit supervision, the notion of a meaningful deviation is often ambiguous, resulting in unstable decision boundaries and limited discriminability.

To address these challenges, we propose **BAD**, an unsupervised dynamic graph anomaly detection framework guided by boundary learning. BAD is designed to address **feature scarcity** and **context-dependent anomalies** with a minimalistic continuous-time architecture that relies on *learnable node identity embeddings* and *pairwise compatibility modeling*, without requiring node/edge attributes or heavy memory-based components. Specifically, BAD assigns each node a learnable embedding and applies cross-attention between the destination node and the source node's recent neighbors to explicitly capture source–destination compatibility, enabling the detector to identify interactions that deviate from the source's historical context. To further resolve the **ambiguous decision boundary** induced by the lack of anomaly labels, BAD leverages normalizing flows to learn the feature distribution of normal interactions and derive an explicit separating boundary near its support, allowing anomalies to be identified as samples falling outside the learned normal region. This combination yields a fully unsupervised detector with strong discriminability and robust generalization under realistic feature-scarce and label-scarce settings. Overall, our contributions are threefold:

1. We introduce **BAD**, an unsupervised anomaly detection framework for continuous-time dynamic graphs that combines temporal modeling with cross-attention to capture context-dependent spatial-temporal deviations.

2. We incorporate normalizing flows to model the distribution of normal interactions and derive an explicit separating boundary, enabling robust detection.

3. Experiments on four real-world datasets against 12 strong baselines show that BAD consistently delivers state-of-the-art performance, demonstrating strong robustness under feature-scarce and label-scarce conditions.

2 Related Work

2.1 Anomaly Detection in Graphs

Anomaly detection in graphs has made substantial progress, especially in static settings, where methods such as node2vec [Grover and Leskovec, 2016], DeepWalk [Perozzi *et al.*, 2014], and contrastive learning [Xu *et al.*, 2022] have been used to identify anomalous nodes, edges, or subgraphs. Benchmark studies [Tang *et al.*, 2023] have further evaluated these approaches, highlighting both their strengths and limitations. However, these methods are tailored to static graphs and do not account for the temporal dynamics inherent to real-world networks.

In dynamic settings, many approaches model graphs as sequences of static snapshots and apply static detectors to each snapshot independently [Yu *et al.*, 2018; Zheng *et al.*, 2019; Li *et al.*, 2023]. While this strategy captures coarse temporal variation, it introduces information loss due to fixed sampling intervals and cannot faithfully represent continuous evolution. Some works such as StrGNN [Cai *et al.*, 2021] and TADDY [Liu *et al.*, 2023] incorporate both spatial and temporal signals by coupling separate modules. However, this loosely coupled design often struggles to capture the tightly intertwined spatial-temporal dependencies required for accurate anomaly detection. A major challenge in dynamic graph anomaly detection is the scarcity of labeled anomalous data [Ma *et al.*, 2023; Qiao *et al.*, 2025; Tian *et al.*, 2026], leading to the widespread use of unsupervised learning. Many studies rely on synthetic anomalies, such as randomly connecting unlinked nodes [Liu *et al.*, 2023], to evaluate models. While practical, these methods generate simplistic patterns that fail to reflect real-world complexities, such as coordinated behaviors, sudden structural changes, or temporally correlated events. This highlights the need for more sophisticated anomaly generation techniques and evaluation frameworks that better mimic the dynamic nature of real-world graphs [Hua *et al.*, 2026].

2.2 Continuous-Time Dynamic Graph Representation Learning

Given the importance of dynamic graphs, representation learning on such graphs has attracted significant attention in recent years. One common approach treats dynamic graphs as sequences of snapshots and applies static graph learning methods to each snapshot independently, leading to discrete-time methods. These methods sample graph snapshots at fixed time intervals and are easy to combine with existing static graph models. However, they require manually setting the time interval, and different choices of interval may lead to different graph structures. Moreover, because they rely on static techniques within each snapshot, they often ignore the temporal order of interactions among nodes inside the same snapshot. As a result, they may struggle to capture interactions across different time granularities, which can reduce prediction accuracy. To address these limitations, there has been increasing interest in continuous-time methods, which treat dynamic graphs as continuous streams of timestamped links. These methods extend Graph Neural Networks (GNNs) to encode temporal networks and often incorporate memory mechanisms [Rossi *et al.*, 2020b; Ma *et al.*, 2020; Trivedi *et al.*, 2019; Chang *et al.*, 2020; Kumar *et al.*, 2019]. In these approaches, each node is associated with a vector representation that is updated when new interactions occur. Some methods update representations using the node currently being interacted with [Kumar *et al.*, 2019; Trivedi *et al.*, 2019], while others use historical interactions or temporal neighborhoods [Xu *et al.*, 2020; Tian *et al.*, 2024b; Wang *et al.*, 2021a; Tian *et al.*, 2024a]. However, these methods primarily focus on temporal patterns and often pay less attention to the structural patterns inherent in graph data. Inspired by static networks, some studies have incorporated random walks [Wang *et al.*, 2021b;

Jin *et al.*, 2022] to extract structural features, thereby enhancing representation learning on dynamic graphs.

3 Preliminaries

Notions. A Dynamic Graph G can be represented as a Continuous-Time Dynamic Graph (CTDG), which captures the evolution of interactions over time as a chronological sequence of node pairs: $G = \{(v_i^0, v_j^0, t_0), \ldots, (v_i^n, v_j^n, t_n)\}$. Here, t_n denotes the timestamp and the timestamps are ordered as $(0 \leq t_0 \leq t_1 \leq ... \leq t_n)$. $v_i^n, v_j^n \in V$ denote the node id of the $n-th$ interaction at timestamp t_n, V is the entire node set. Each node $v \in V$ is associated with node feature $F_v \in R^{d_v}$, and each edge (v_i, v_j, t) represents an interaction from node v_i to node v_j at time t accompanied by an edge feature $e_{i,j}^t \in R^{d_e}$. The dimension d_v and d_e correspond to the feature sizes of the nodes and edges, respectively. This representation allows for a fine-grained modeling of temporal dynamics and structural changes in the graph, making it particularly suitable for tasks such as anomaly detection in dynamic networks.

Problem Definition. In this work, our objective is to detect anomalous edges in dynamic graphs. Specifically, given an interaction (v_i, v_j, t) and historical events before t, we aim to develop a model that learns the edge representation for (v_i, v_j, t) and assigns it a continuous anomaly score in $[0, 1]$, thereby quantifying its degree of abnormality and determining whether the event is anomalous. Following the common practice in dynamic graph anomaly detection, we adopt an unsupervised learning scheme. During training, all edges are treated as normal samples, and no anomaly labels are used.

4 Methodology

4.1 Encoding Layer

To illustrate the computation process of our model, we use the example of computing the representation for an edge. Given the edge (v_i, v_j, t), we begin by sampling L first-hop historical neighbors for node v_i, v_j constructing the neighbor node sequence $S_* = [s_*^1, \ldots, s_*^l, \ldots, s_*^L]$, where $*$ represents either i or j. And the set of neighbor $\{s_*^l\}_{l=1,\ldots,L}$ are ordered based on temporal proximity, with l denoting the $l-th$ neighbor in the sequence. If a node has fewer than L historical neighbors, zero-padding is applied to ensure consistent sequence lengths. Next, we will introduce the encoding layer, which captures both the structural and temporal characteristics of the interactions.

Node/Edge Raw Feature Embedding Encoding. To derive embeddings for interactions, we extract the intrinsic node and edge features of the neighbors in the sequence S_*^t. Following prior work, we construct the node and edge embedding sequences $\mathbf{S}_*$ as follows:

$$\mathbf{E}_*^{node} = [\mathbf{F}_{s_*^1}, \mathbf{F}_{s_*^2} \cdots \mathbf{F}_{s_*^L}] \tag{1}$$

$$\mathbf{E}_*^{edge} = [\mathbf{e}_{*,s_*^1}, \mathbf{e}_{*,s_*^2} \cdots \mathbf{e}_{*,s_*^L}] \tag{2}$$

where $\mathbf{E}_*^{node} \in \mathbb{R}^{L \times d_v}$ and $\mathbf{E}_*^{edge} \in \mathbb{R}^{L \times d_v}$ is the concatenation of the corresponding embeddings of nodes and edges in $\mathbf{S}_*$, respectively.

Learnable Node Embedding (Position encoding): Notably, for non-attributed dynamic graphs, setting node and edge raw feature to zero vectors makes the model rely entirely on structural and temporal signals. To represent nodes with unique identifiers in this feature-scarce setting, we introduce learnable node embeddings that are trained jointly with the model. These embeddings serve as latent identifiers and allow the model to incorporate contextual information from each node's interaction history. Specifically, we learn a positional embedding for the neighbors in

$$\mathbf{E}_*^{pos} = [\mathbf{P}_{s_*^1}, \mathbf{P}_{s_*^2} \cdots \mathbf{P}_{s_*^L}] \tag{3}$$

where $\mathbf{E}_*^{pos} \in \mathbb{R}^{L \times d_p}$ is the concatenation of the corresponding depth embeddings of nodes in $\mathbf{S}_*^t$.This positional encoding enables the model to capture both short-term and long-term interaction tendencies of the source node, which are critical for characterizing future behaviors. Here, the position of each node reflects its temporal order in the observed interaction sequence. Nevertheless, positional information alone is insufficient, since different nodes may share the same position in the history and thus cannot be uniquely identified by depth alone.

Time Embedding: To enhance nodes' positional information, we also encoding time intervals that usually convey important behaviour information. Specifically, we utilize the widely adopted time encoding function $cos(t_n\omega)$, where $\boldsymbol{\omega} = \{\alpha^{-(i-1)/\beta}\}_{i=1}^{d_t}$ is employed to encode timestamps. α and β are hyperparameters to make $t_{max} \times \alpha^{-(i-1)/\beta}$ close to 0 when i close to d_t. A cosine function is then applied to project these values into the range of $[-1, +1]$. Notably, we use relative timestamps instead of absolute timestamps for encoding. In other words, if the timestamp of the sampled interaction is t_0, and the specific timestamp for link prediction is t, we utilize $cos((t_n - t)\omega)$ as the effective relative time encoding function. So the time embedding can be represent as follow:

$$\mathbf{E}_*^{time} = [cos((t_1 - t)\omega), cos((t_2 - t)\omega) \cdots cos((t_L - t)\omega)] \tag{4}$$

where $\mathbf{E}_*^{\mathbf{time}} \in \mathbb{R}^{L \times d_t}$ is the concatenation of the corresponding time-interval embeddings of nodes in $\mathbf{S}_*$.

Finally, we get the input node embedding $\hat{\mathbf{E}}_* \in \mathbb{R}^{L \times (d_v + d_e + d_p + d_t)}$ for next cross-attention transformer module :

$$\tilde{\mathbf{E}}_* = \mathbf{E}_*^{node} \| \mathbf{E}_*^{edge} \| \mathbf{E}_*^{pos} \| \mathbf{E}_*^{time} \tag{5}$$

4.2 Pairwise Compatibility Modeling

Considering that edge anomaly detection fundamentally requires judging whether a destination node is *compatible* with the source node's recent interaction context, we introduce pairwise compatibility modeling to explicitly capture this source–destination relationship. Unlike common designs that first compress each node's history into an independent representation and then apply an MLP to score an interaction, our goal is to let the destination directly *query* the source's historical neighborhood, so that incompatibility can be detected as a contextual mismatch rather than a purely temporal rarity.

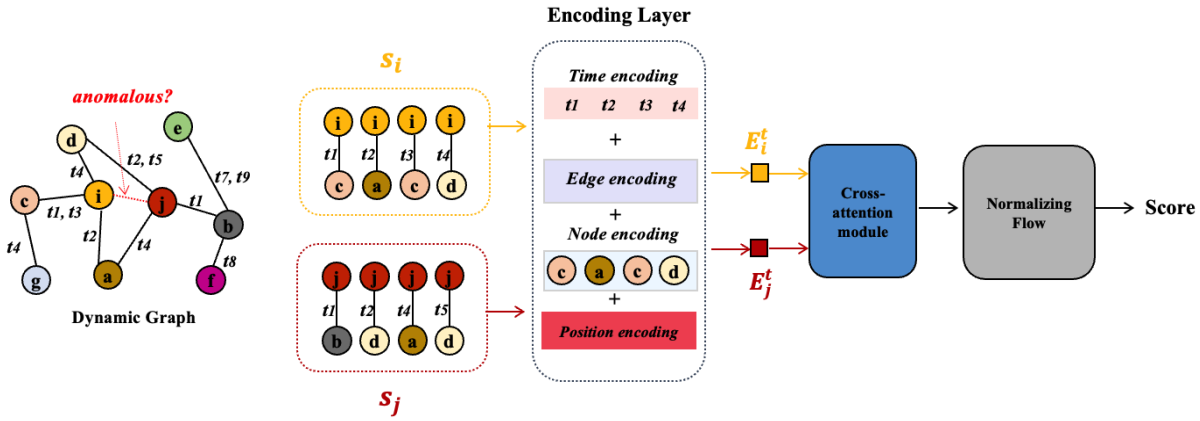

Figure 1: The overall architecture of our proposed method BAD. Given a source node v_i and a set of candidate destination nodes at time t, we first retrieve the L most recent neighbors of v_i prior to t, and look up the corresponding node embeddings for both these neighbors and the candidate destinations. For the l-th neighbor of v_i, we add a positional encoding to its embedding to preserve the temporal order of the neighborhood sequence (Section 4.1). We then apply a cross-attention module between the candidate destinations and the source's recent neighbors to construct edge representations (Section 4.2). Finally, we employ a boundary-based anomaly detector built upon normalizing flows to learn the normal feature distribution and identify anomalous interactions (Section 4.3).

To this end, we implement compatibility modeling via cross-attention between each candidate destination and the source's recent neighbors. Intuitively, the source's neighbor sequence summarizes its evolving interaction preference, and cross-attention allows each destination to selectively attend to the most relevant neighbors when forming its interaction-aware representation. This mechanism enables direct assessment of whether the destination aligns with the source's temporal context, which is crucial for identifying context-dependent anomalies. Concretely, the module aims to produce dynamic representations for the interacting nodes v_i (source) and v_j (destination) at timestamp t by leveraging the constructed embedding sequences. We adopt a two-stream encoder with a cross-attention architecture. In the first stage, a graph Transformer block independently encodes the source-side neighborhood sequence and the destination-side candidate sequence, producing intermediate embeddings $\tilde{\boldsymbol{H}}_i$ and $\tilde{\boldsymbol{H}}_j$. In the second stage, a cross-attentional Transformer block explicitly models their dependency by exchanging information between the two streams, yielding the final embedding matrices $\boldsymbol{H}_i$ and $\boldsymbol{H}_j$. We then extract the dynamic embeddings $\mathbf{h}_{i(t)}$ and $\mathbf{h}_{j(t)}$ for the interacting nodes v_i and v_j at time t. The core of the cross-attentional block is a multi-head cross-attention operation, where the destination stream queries the source stream and aggregates key–value information to form compatibility-aware representations. Formally, the operations are defined as follows:

$$\boldsymbol{Q}_i, \boldsymbol{K}_i, \boldsymbol{V}_i = \tilde{\boldsymbol{H}}_i \boldsymbol{W}_Q, \tilde{\boldsymbol{H}}_i \boldsymbol{W}_K, \tilde{\boldsymbol{H}}_i \boldsymbol{W}_V \tag{6}$$

$$\boldsymbol{O}_i = \mathrm{MultiHead}(\boldsymbol{Q}_i, \boldsymbol{K}_j, \boldsymbol{V}_j) \tag{7}$$

$$\boldsymbol{H}_i = \mathrm{LN}(\mathrm{FFN}(\mathrm{LN}(\boldsymbol{O}_i + \boldsymbol{Q}_i)) + \mathrm{LN}(\boldsymbol{O}_i + \boldsymbol{Q}_i)) \tag{8}$$

where LN denotes the layer normalization and FFN represents a two-layer fully-connected feed-forward network module defined as:

$$\mathrm{FFN}(x) = \sigma\left(xW_1 + b_1\right) W_2 + b_2 \tag{9}$$

where σ is an activation function. We sequentially apply these operations to get $\boldsymbol{H}_*$, which is the output of our graph Transformer block. The cross-attention mechanism enables the model to focus on shared semantic information between the two sequences while filtering out irrelevant details. By exchanging key-value pairs, the encoder emphasizes meaningful patterns and suppresses noise. Additionally, the attentive information from one sequence is integrated into the final representation of the other, ensuring that both representations are enriched with relevant contextual insights. This results in semantically meaningful and highly informative embeddings. Finally, the dynamic node embeddings for the interacting nodes are retrieved from the enriched embedding matrices, allowing the model to effectively capture the temporal and structural nuances of the interactions.

4.3 Boundary-based anomaly detector

In this scetion, we first briefly review normalizing flows and then describe how we integrate them into our framework.

Normalizing Flow. We follow the common definition and denote $\varphi_\theta : \mathcal{X} \in \mathbb{R}^d \rightarrow \mathcal{Z} \in \mathbb{R}^d$ as the normalizing flow. It is built as a composition of coupling layers such that $\varphi_\theta = \varphi_L \circ \cdots \circ \varphi_2 \circ \varphi_1$, where θ is the trainable parameters and L is the total number of layers. Defining d-dimensional input and output features of normalizing flow as $y_0 = x \in \mathcal{X}$ and $y_L = z \in \mathcal{Z}$, the latents can be computed as $y_l = \varphi_l(y_{l-1})$, where $\{y_l\}_{l=1}^{L-1}$ are the intermediate outputs. The input distribution estimated by model $p_\theta(x)$ can be calculated according to the change of variables formula as follows:

$$\log p_\theta(x) = \log p_{\mathcal{Z}}(\varphi_\theta(x)) + \sum_{l=1}^{L} \log \left| \det J_{\varphi_l}(y_{l-1}) \right| \tag{10}$$

where $J_{\varphi_l}(y_{l-1}) = \frac{\partial \varphi_l(y_{l-1})}{\partial y_{l-1}}$ is the Jacobian matrix of the

transformation φ_l at y_{l-1}, and $\det$ means determinant. Normalizing flow can approximate the feature distribution $p_{\mathcal{X}}$ with $p_\theta(x)$. The set of parameters θ is obtained by optimizing the log-likelihoods across the training distribution $p_{\mathcal{X}}$:

$$\theta^* = \underset{\theta\in\Theta}{\operatorname{argmin}}\, \mathbb{E}_{x\sim p_{\mathcal{X}}}[-\log p_\theta(x)] \tag{11}$$

We then use normalizing flow to learn the embedding distribution through maximum likelihood optimization. The latent variable distribution $p_{\mathcal{Z}}(z), z \in \mathbb{R}^d$, is typically assumed to follow a multivariate Gaussian distribution:

$$p_{\mathcal{Z}}(z) = (2\pi)^{-\frac{d}{2}} \det(\Sigma^{-\frac{1}{2}}) \mathrm{e}^{-\frac{1}{2}(z-\mu)^T\Sigma^{-1}(z-\mu)} \tag{12}$$

where μ is the mean and Σ is the covariance. For simplicity, when training on normal features, the latent variables are assumed to follow $\mathcal{N}(0, \mathbf{I})$. Substituting $p_{\mathcal{Z}}(z) = (2\pi)^{-\frac{d}{2}} \mathrm{e}^{-\frac{1}{2}z^T z}$ in formula (10), the optimization objective becomes::

$$\theta^* = \underset{\theta\in\Theta}{\operatorname{argmin}}\, \mathbb{E}_{x\sim p_{\mathcal{X}}}\Big[\frac{d}{2}\log(2\pi) + \frac{1}{2}\varphi_\theta(x)^T\varphi_\theta(x) - \sum\nolimits_{l=1}^{L} \log\big|\det J_{\varphi_l}(y_{l-1})\big|\Big] \tag{13}$$

The maximum likelihood loss function for learning the normal distribution is defined as:

$$\mathcal{L}_{ml} = \mathbb{E}_{x\in\mathcal{X}^n}\Big[\frac{d}{2}\log(2\pi) + \frac{1}{2}\varphi_\theta(x)^T\varphi_\theta(x) - \sum\nolimits_{l=1}^{L} \log\big|\det J_{\varphi_l}(y_{l-1})\big|\Big] \tag{14}$$

Identify Boundary. With the learned normal feature distribution, we derive an explicit and compact separating boundary. Since the feature space is high-dimensional, we construct decision boundary directly in the log-likelihood space. Specifically, using Eq. (15), we compute the log-likelihoods of all normal features and obtain $\mathcal{P}_n = \{\log p_(x_i)_{i=1}^N\}$, which approximates the normal log-likelihood distribution. We then introduce a percentile hyperparameter β to control how far the boundary lies from the distribution center, and set the normal boundary b_n as the β-th percentile of the sorted values in $\mathcal{P}_n$ (e.g., $\beta = 1$). This choice ensures that the expected false positive rate on normal samples is upper bounded by $\beta\%$. To further improve robustness, we add a margin τ (e.g., $\tau = 0.1$) and define an abnormal boundary as $b_a = b_n - \tau$. This margin creates a buffer region that facilitates more reliable separation between normal and anomalous features.

Anomaly Scoring. The key advantage of normalizing flows is that they enable exact evaluation of the log-likelihood $\log p(x)$ for each input sample x:

$$\log p(x) = -\frac{d}{2}\log(2\pi) - \frac{1}{2}\varphi_\theta(x)^T\varphi_\theta(x) + \sum\nolimits_{l=1}^{L} \log\big|\det J_{\varphi_l}(y_{l-1})\big| \tag{15}$$

Given $\log p(x)$, the corresponding likelihood is $\exp(\log p(x))$. Because the exponential function is monotonic, $\log p(x)$ and $\exp(\log p(x))$ induce the same ordering over samples. Moreover, since we maximize the log-likelihood of normal samples in Eq. (14), this likelihood provides a direct measure of normality. We therefore define the anomaly score of a sample x as:

$$score(x) = \max_{x'\in\mathcal{X}}(\exp(\log p(x'))) - \exp(\log p(x)) \tag{16}$$

5 Experiments

	Wikipedia	Reddit	Bitcoin-alpha	Bitcoin-OTC
# Nodes	9,227	10,984	3,783	5,881
# Edges	157,474	672,447	24,186	35,592
# Features	172	172	0	0
# anomalies	217	366	874	2,568
anomalies ratio	0.14%	0.05%	3.61%	7.22%

Table 1: Statistics of datasets.

5.1 Datasets

We evaluate our method on four real-world dynamic graph datasets: Wikipedia, Reddit, Bitcoin-Alpha, and Bitcoin-OTC. The Wikipedia dataset records user edits on Wikipedia pages, where a user is labeled as anomalous if they are banned after a specific edit. The Reddit dataset consists of user posts on subreddits, and the label indicates whether the user is banned after the corresponding post. Bitcoin-Alpha and Bitcoin-OTC are who-trusts-whom networks of bitcoin traders from [www.btc-alpha.com] and [www.bitcoin-otc.com], where users assign ratings from -10 (total distrust) to +10 (complete trust). We use these time-stamped ratings to identify anomalous nodes and assign dynamic labels. For all datasets, we use the final 15% of interactions in chronological order as the test set for both our method and all baselines. Dataset statistics are provided in Table 1.

5.2 Baselines

We extensively compare BAD against 12 baselines, which can be classified into four categories: rule-based methods, learning-based discrete-time and continuous-time dynamic graph methods.

- **Rule-based Stream Methods**: SedanSpot [Eswaran and Faloutsos, 2018], MIDAS [Bhatia *et al.*, 2020], F-FADE [Chang *et al.*, 2021], Anoedge [Bhatia *et al.*, 2023]
- **Learning-based DTDG Methods**: StrGNN [Cai *et al.*, 2021], TADDY [Liu *et al.*, 2023]
- **Learning-based CTDG Methods**: JODIE [Kumar *et al.*, 2019], DyRep [Trivedi *et al.*, 2019], TGAT [Xu *et al.*, 2020], TGN [Rossi *et al.*, 2020a], SAD [Tian *et al.*, 2023], SLADE [Lee *et al.*, 2024]

For the four rule-based methods, which do not involve representation learning, we directly evaluate their performance on the test set without training. For the six learning-based methods, we train each model on the training set and use the checkpoint with the best validation performance for testing. Except for SAD, these learning-based baselines follow the standard two-stage pipeline in [Lee *et al.*, 2024]: first training the encoder via link prediction–based self-supervised learning, and then training a classifier with labels in a supervised manner.

Method	Wikipedia		Reddit		Bitcoin-alpha		Bitcoin-OTC	
	AUC	AP	AUC	AP	AUC	AP	AUC	AP
SedanSpot	82.88 ± 1.54	0.66 ± 0.06	58.97 ± 1.85	0.09 ± 0.00	69.09 ± 0.76	12.99 ± 0.06	71.71 ± 0.73	16.14 ± 0.01
MIDAS	62.92 ± 3.53	0.41 ± 0.02	59.94 ± 0.95	0.12 ± 0.00	64.57 ± 0.11	12.99 ± 0.06	62.16 ± 1.99	16.14 ± 0.01
F-FADE	44.88 ± 0.00	0.18 ± 0.00	49.79 ± 0.05	0.09 ± 0.00	53.57 ± 0.00	6.34 ± 0.00	51.12 ± 0.00	8.80 ± 0.00
Anoedge	47.38 ± 0.32	0.18 ± 0.01	48.47 ± 0.46	0.09 ± 0.00	62.52 ± 0.24	12.52 ± 0.86	65.99 ± 0.19	17.03 ± 0.11
STrGNN	50.95 ± 0.46	0.21 ± 0.01	53.30 ± 0.22	0.10 ± 0.01	51.26 ± 0.01	5.14 ± 0.05	47.85 ± 0.15	6.06 ± 0.03
TADDY	56.48 ± 1.41	0.23 ± 0.02	54.54 ± 0.43	0.11 ± 0.02	57.52 ± 1.41	6.92 ± 0.18	64.06 ± 0.38	14.52 ± 0.09
JODIE	85.75 ± 0.17	1.40 ± 0.02	61.47 ± 0.58	0.25 ± 0.05	73.53 ± 1.26	14.31 ± 0.40	69.13 ± 0.96	15.79 ± 0.11
DyRep	85.68 ± 0.34	1.53 ± 0.02	63.42 ± 0.62	0.14 ± 0.01	73.30 ± 1.51	8.41 ± 0.58	70.76 ± 0.70	16.14 ± 1.74
TGAT	83.24 ± 1.11	1.48 ± 0.14	65.12 ± 1.65	0.27 ± 0.12	71.67 ± 0.90	12.19 ± 0.13	68.33 ± 1.23	18.81 ± 0.63
TGN	87.47 ± 0.22	1.30 ± 0.04	67.16 ± 1.03	0.19 ± 0.01	69.90 ± 0.99	9.65 ± 0.40	76.23 ± 0.23	19.87 ± 1.75
SAD	86.15 ± 0.63	**2.77 ± 0.91**	68.45 ± 1.27	0.28 ± 0.06	68.56 ± 3.17	12.55 ± 0.58	64.67 ± 4.97	14.33 ± 0.88
SLADE	<u>88.68 ± 0.39</u>	1.56 ± 0.14	<u>75.08 ± 0.50</u>	<u>0.30 ± 0.05</u>	<u>76.92 ± 0.36</u>	<u>14.86 ± 0.08</u>	<u>77.18 ± 0.27</u>	<u>20.46 ± 0.06</u>
BAD	**89.17 ± 0.60**	<u>1.82 ± 0.11</u>	**75.77 ± 0.54**	**0.32 ± 0.06**	**77.60 ± 0.41**	**15.03 ± 0.05**	**77.95 ± 0.44**	**20.68 ± 0.06**

Table 2: Performance comparison of AUC and AP (avg ± std in %). The first four methods are rule-based models, and the others are learning-based. The best and the second-best results are highlighted in **bold** and <u>underlined</u>, respectively.

5.3 Experimental setting

Most prior work only use the Area Under the Receiver Operating Characteristic Curve (AUC) as the evaluation metric. However, AUROC can be optimistic under highly skewed label distributions. We therefore also report the Area Under the Precision–Recall Curve (AP) for a more comprehensive assessment. All models are trained for up to 100 epochs, and the checkpoint with the best validation performance is used for testing. We fix the batch size to 100 across all methods and datasets. To mitigate randomness, we report the mean and standard deviation over five independent runs. We optimize model parameters using Adam [Kingma and Ba, 2015] and perform grid search over its main hyperparameters.

5.4 Experimental Results

We compare our proposed method, BAD, with 12 strong baselines across four datasets. As shown in Table 2, BAD achieves the best overall performance across all four datasets, consistently outperforming strong rule-based and learning-based baselines. The results also reveal clear differences among baseline categories. **Rule-based methods** generally perform poorly, especially on Wikipedia and Reddit, where AP is close to zero. This is expected because these methods are designed for specific anomaly patterns such as burstiness and rely on hand-crafted statistics, which limits their ability to detect more diverse and complex anomalies. **Learning-based DTDG methods** improve slightly over rule-based approaches but remain substantially worse than CTDG methods. This gap highlights the limitations of the snapshot-based paradigm, where temporal discretization and information loss prevent accurate modeling of fine-grained dynamics in edge streams. **Learning-based CTDG methods** achieve markedly stronger results, confirming the importance of continuous-time modeling for edge-stream anomaly detection. Among them, SLADE is the strongest baseline across datasets, while SAD occasionally yields competitive AP (e.g., on Wikipedia), indicating that explicitly leveraging anomaly-oriented training objectives can benefit precision–recall performance. Nevertheless, BAD consistently surpasses these methods on all datasets in AUC and achieves the best AP on three out of four datasets, demonstrating that our minimalistic identity-based representation and explicit source–destination compatibility modeling provide additional discriminative power beyond existing CTDG encoders.

In addition, we evaluate the considered methods under different training-test splits. Specifically, we use the first $T\%$ of edges as the train set and evaluate each model on the remaining edges. We refer to T as a test start ratio. As shown in Figure 2, BAD outperforms all baseline methods in most settings across different test start ratios. Moreover, on the Wikipedia dataset, BAD shows only a marginal performance drop when the test start ratio changes from 80% to 40%, indicating that using only about half of the dataset for training is sufficient to retain strong detection performance.

We also provide a qualitative analysis of the anomaly scores produced by BAD on the Wikipedia and Reddit datasets in Figure 3. In the top row (Figures 3(a) and 3(b)), the estimated score distributions exhibit a clear separation between normal and anomalous samples. Normal interactions concentrate sharply near low anomaly scores, forming a dominant peak close to zero, whereas anomalies shift toward substantially higher scores with a broader spread. This consistent rightward shift indicates that BAD assigns systematically higher abnormality to truly anomalous events, rather than relying on a few extreme outliers. Although a small overlap remains in the intermediate score range, the two distributions are largely distinguishable, suggesting that BAD learns a discriminative boundary in the score space. The bottom row (Figures 3(c) and 3(d)) further visualizes anomaly scores across the full test stream. Normal samples remain stably

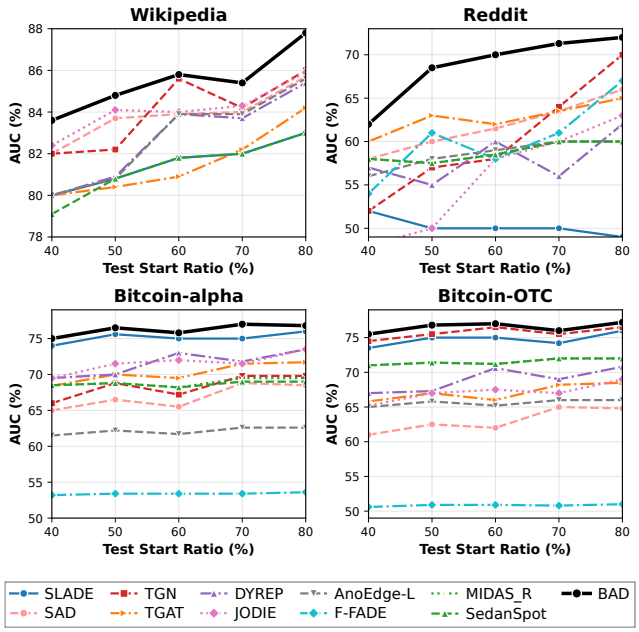

Figure 2: AUC (in %) when varying the test start ratio. For learning-based methods, temporal edges preceding the test start ratio in the dataset are employed for training. If validation is needed, the last 10% of the training set is used for validation.

clustered at low scores throughout the timeline, demonstrating that BAD produces consistent confidence estimates under continuous temporal evolution. In contrast, anomalous samples appear as sparse points with noticeably higher scores, and this pattern persists across the entire stream rather than being confined to specific periods. This behavior is important for edge-stream anomaly detection, where anomalies are rare and may occur at arbitrary times. Overall, these observations confirm that BAD yields well-separated score distributions and maintains reliable detection signals over long temporal ranges, supporting its strong performance in Table 2.

5.5 Ablation Study

We conduct an ablation study to assess the contribution of each component in BAD by systematically removing key modules. We consider the following variants: **w/o pos**, which removes the learnable node identity embeddings; **w/o time**, which excludes time encoding and thus discards temporal information; **w/o CA**, which replaces cross-attention with self-attention; and **w/o NF**, which replaces the normalizing flow with a simple MLP classifier. We compare BAD against these variants under the same hyperparameter settings as the full model. As summarized in Table 3, BAD consistently achieves the best performance across all datasets, indicating that each component contributes to the overall effectiveness, albeit to different extents. Among the variants, removing time encoding (**w/o time**) causes the largest degradation across all datasets. Replacing cross-attention with self-attention (**w/o CA**) also leads to a clear performance drop, indicating that explicit source–destination compatibility modeling is critical for identifying context-dependent anomalies. In contrast,

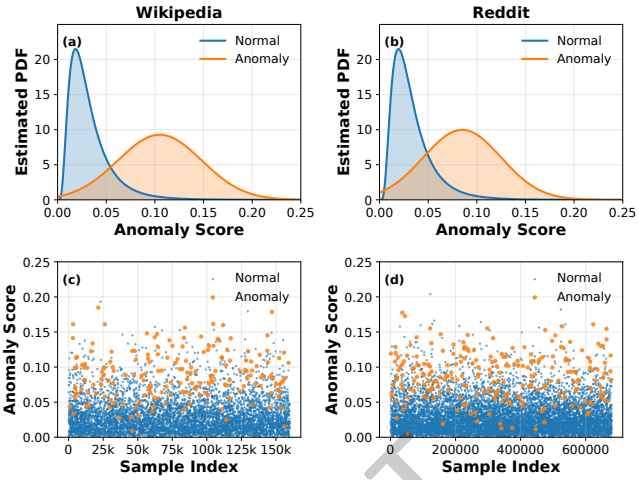

Figure 3: Top two figures show the distribution of anomaly scores assigned by BAD in Wikipedia and Reddit datasets. Bottom two figures show the anomaly scores.

	Wikipedia	Reddit	Bitcoin-alpha	Bitcoin-OTC
w/o CA	85.72 ± 1.36	71.51 ± 1.56	76.09 ± 0.72	74.10 ± 1.10
w/o pos	86.64 ± 1.17	72.49 ± 0.95	75.07 ± 1.38	74.57 ± 0.43
w/o NF	86.69 ± 0.84	73.30 ± 0.96	76.84 ± 0.70	76.62 ± 0.62
w/o time	84.57 ± 0.35	70.77 ± 0.32	74.84 ± 1.40	74.61 ± 0.75
BAD	**89.17 ± 0.60**	**75.77 ± 0.54**	77.60 ± 0.41	77.95 ± 0.44

Table 3: Comparison of AUC (in %) of BAD and its four variants. The best results are highlighted in **bold**. BAD with all components performs the best overall, showing the effectiveness of each.

removing node identity embeddings (**w/o pos**) results in a smaller yet consistent decline, suggesting that learnable identifiers provide complementary discriminative information under feature-scarce settings. Finally, replacing the normalizing flow with an MLP classifier (**w/o NF**) yields the most competitive variant but still underperforms the full model, demonstrating that explicit boundary learning further improves separability in the unsupervised detection setting.

6 Conclusion

In this paper, we proposed **BAD**, an unsupervised framework for anomaly detection in continuous-time dynamic graphs. BAD addresses realistic challenges under extreme class imbalance. It adopts a minimalistic design with learnable node identity embeddings and cross-attention to explicitly model source–destination compatibility, and further leverages normalizing flows to learn the normal interaction distribution and derive a robust separating boundary without anomaly labels. Experiments on four real-world datasets against 12 strong baselines show that BAD consistently achieves superior AUROC and AUPRC, demonstrating its effectiveness in feature-scarce and label-scarce settings.

Acknowledgements

This work is supported by National Natural Science Foundation of China (No. 62366021).

References

[Berger-Wolf and Saia, 2006] Tanya Y. Berger-Wolf and Jared Saia. A framework for analysis of dynamic social networks. In *Proceedings of the 12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '06, page 523–528, New York, NY, USA, 2006. Association for Computing Machinery.

[Bhatia *et al.*, 2020] Siddharth Bhatia, Bryan Hooi, Minji Yoon, Kijung Shin, and Christos Faloutsos. Midas: Microcluster-based detector of anomalies in edge streams. In *Proceedings of the AAAI conference on artificial intelligence*, volume 34, pages 3242–3249, 2020.

[Bhatia *et al.*, 2023] Siddharth Bhatia, Mohit Wadhwa, Kenji Kawaguchi, Neil Shah, Philip S. Yu, and Bryan Hooi. Sketch-based anomaly detection in streaming graphs. KDD '23, page 93–104, New York, NY, USA, 2023. Association for Computing Machinery.

[Cai *et al.*, 2021] Lei Cai, Zhengzhang Chen, Chen Luo, Jiaping Gui, Jingchao Ni, Ding Li, and Haifeng Chen. Structural temporal graph neural networks for anomaly detection in dynamic graphs. In *Proceedings of the 30th ACM International Conference on Information & Knowledge Management*, CIKM '21, page 3747–3756, New York, NY, USA, 2021. Association for Computing Machinery.

[Chang *et al.*, 2020] Xiaofu Chang, Xuqin Liu, Jianfeng Wen, Shuang Li, Yanming Fang, Le Song, and Yuan Qi. Continuous-time dynamic graph learning via neural interaction processes. In *CIKM*, 2020.

[Chang *et al.*, 2021] Yen-Yu Chang, Pan Li, Rok Sosic, M. H. Afifi, Marco Schweighauser, and Jure Leskovec. F-fade: Frequency factorization for anomaly detection in edge streams. In *Proceedings of the 14th ACM International Conference on Web Search and Data Mining*, WSDM '21, page 589–597, New York, NY, USA, 2021. Association for Computing Machinery.

[Ding *et al.*, 2019] Kaize Ding, Jundong Li, Rohit Bhanushali, and Huan Liu. Deep anomaly detection on attributed networks. In *Proceedings of the 2019 SIAM international conference on data mining*, pages 594–602. SIAM, 2019.

[Eswaran and Faloutsos, 2018] Dhivya Eswaran and Christos Faloutsos. Sedanspot: Detecting anomalies in edge streams. In *ICDM*, 2018.

[Fan *et al.*, 2020] Haoyi Fan, Fengbin Zhang, and Zuoyong Li. Anomalydae: Dual autoencoder for anomaly detection on attributed networks. In *ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 5685–5689. IEEE, 2020.

[Greene *et al.*, 2010] Derek Greene, Dónal Doyle, and Pádraig Cunningham. Tracking the evolution of communities in dynamic social networks. In *2010 International Conference on Advances in Social Networks Analysis and Mining*, pages 176–183, 2010.

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

[Hua *et al.*, 2026] Fengrui Hua, Yiyan Qi, Zikai Wei, Yuxing Tian, Chengjin Xu, Xiaojun Wu, Jia Li, and Jian Guo. Bag: Benchmarking anomaly detection on dynamic graphs. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 40, pages 14892–14900, 2026.

[Jin *et al.*, 2022] Ming Jin, Yuan-Fang Li, and Shirui Pan. Neural temporal walks: Motif-aware representation learning on continuous-time dynamic graphs. In *NeurIPS*, 2022.

[Kingma and Ba, 2015] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *International Conference on Learning Representations (ICLR)*, 2015.

[Kumar *et al.*, 2019] Srijan Kumar, Xikun Zhang, and Jure Leskovec. Predicting dynamic embedding trajectory in temporal interaction networks. In *KDD*, 2019.

[Lee *et al.*, 2024] Jongha Lee, Sunwoo Kim, and Kijung Shin. Slade: Detecting dynamic anomalies in edge streams without labels via self-supervised learning. In *Proceedings of the 30th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, KDD '24, page 1506–1517, New York, NY, USA, 2024. Association for Computing Machinery.

[Li *et al.*, 2023] Song Li, Jiandong Zhou, Chong MO, Jin LI, Geoffrey K. F. Tso, and Yuxing Tian. Motif-aware temporal gcn for fraud detection in signed cryptocurrency trust networks, 2023.

[Liu *et al.*, 2023] Yixin Liu, Shirui Pan, Yu Guang Wang, Fei Xiong, Liang Wang, Qingfeng Chen, and Vincent CS Lee. Anomaly detection in dynamic graphs via transformer. *IEEE Transactions on Knowledge and Data Engineering*, 35(12):12081–12094, 2023.

[Ma *et al.*, 2020] Yao Ma, Ziyi Guo, Zhaochun Ren, Jiliang Tang, and Dawei Yin. Streaming graph neural networks. In *SIGIR*, 2020.

[Ma *et al.*, 2023] Xiaoxiao Ma, Jia Wu, Shan Xue, Jian Yang, Chuan Zhou, Quan Z. Sheng, Hui Xiong, and Leman Akoglu. A comprehensive survey on graph anomaly detection with deep learning. *IEEE Transactions on Knowledge and Data Engineering*, 35(12):12012–12038, 2023.

[Peng *et al.*, 2018] Zhen Peng, Minnan Luo, Jundong Li, Huan Liu, Qinghua Zheng, et al. Anomalous: A joint modeling approach for anomaly detection on attributed networks. In *IJCAI*, volume 18, pages 3513–3519, 2018.

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

[Postuvan *et al.*, 2024] Tim Postuvan, Claas Grohnfeldt, Michele Russo, and Giulio Lovisotto. Learning-based link

anomaly detection in continuous-time dynamic graphs. *Transactions on Machine Learning Research*, 2024.

[Qiao *et al.*, 2025] Hezhe Qiao, Hanghang Tong, Bo An, Irwin King, Charu Aggarwal, and Guansong Pang. Deep graph anomaly detection: A survey and new perspectives. *IEEE Transactions on Knowledge and Data Engineering*, 37(9):5106–5126, 2025.

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

[Rossi *et al.*, 2020b] Emanuele Rossi, Ben Chamberlain, Fabrizio Frasca, Davide Eynard, Federico Monti, and Michael M. Bronstein. Temporal graph networks for deep learning on dynamic graphs. *CoRR*, abs/2006.10637, 2020.

[Tang *et al.*, 2023] Jianheng Tang, Fengrui Hua, Ziqi Gao, Peilin Zhao, and Jia Li. Gadbench: Revisiting and benchmarking supervised graph anomaly detection. *Advances in Neural Information Processing Systems*, 36:29628–29653, 2023.

[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. Sad: semi-supervised anomaly detection on dynamic graphs. In *Proceedings of the Thirty-Second International Joint Conference on Artificial Intelligence*, IJCAI '23, 2023.

[Tian *et al.*, 2024a] Yuxing Tian, Aiwen Jiang, Qi Huang, Jian Guo, and Yiyan Qi. Latent diffusion-based data augmentation for continuous-time dynamic graph model. In *Proceedings of the 30th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, KDD '24, page 2900–2911, New York, NY, USA, 2024. Association for Computing Machinery.

[Tian *et al.*, 2024b] Yuxing Tian, Yiyan Qi, and Fan Guo. Freedyg: Frequency enhanced continuous-time dynamic graph model for link prediction. In *The Twelfth International Conference on Learning Representations*, 2024.

[Tian *et al.*, 2026] Yuxing Tian, Yiyan Qi, Fengran Mo, Weixu Zhang, Jian Guo, and Jian-Yun Nie. Learning discriminative and generalizable anomaly detector for dynamic graph with limited supervision, 2026.

[Trivedi *et al.*, 2019] Rakshit S. Trivedi, Mehrdad Farajtabar, Prasenjeet Biswal, and Hongyuan Zha. Dyrep: Learning representations over dynamic graphs. In *ICLR*, 2019.

[Wang *et al.*, 2021a] Xuhong Wang, Ding Lyu, Mengjian Li, Yang Xia, Qi Yang, Xinwen Wang, Xinguang Wang, Ping Cui, Yupu Yang, Bowen Sun, and Zhenyu Guo. APAN: asynchronous propagation attention network for real-time temporal graph embedding. In *SIGMOD*, 2021.

[Wang *et al.*, 2021b] Yanbang Wang, Yen-Yu Chang, Yunyu Liu, Jure Leskovec, and Pan Li. Inductive representation learning in temporal networks via causal anonymous walks. In *ICLR*, 2021.

[Xu *et al.*, 2020] Da Xu, chuanwei ruan, evren korpeoglu, sushant kumar, and kannan achan. Inductive representation learning on temporal graphs. In *ICLR*, 2020.

[Xu *et al.*, 2022] Zhiming Xu, Xiao Huang, Yue Zhao, Yushun Dong, and Jundong Li. Contrastive attributed network anomaly detection with data augmentation. In *Pacific-Asia conference on knowledge discovery and data mining*, pages 444–457. Springer, 2022.

[Yang *et al.*, 2024] Xiao Yang, Xuejiao Zhao, and Zhiqi Shen. A generalizable anomaly detection method in dynamic graphs, 2024.

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

[Yuan *et al.*, 2021] Xu Yuan, Na Zhou, Shuo Yu, Huafei Huang, Zhikui Chen, and Feng Xia. Higher-order structure based anomaly detection on attributed networks. In *2021 IEEE International Conference on Big Data (Big Data)*, pages 2691–2700. IEEE, 2021.

[Zhang *et al.*, 2021] Shilei Zhang, Toyotaro Suzumura, and Li Zhang. Dyngraphtrans: Dynamic graph embedding via modified universal transformer networks for financial transaction data. In *2021 IEEE International Conference on Smart Data Services (SMDS)*, pages 184–191, 2021.

[Zhang *et al.*, 2022] Chenxi Zhang, Xin Peng, Chaofeng Sha, Ke Zhang, Zhenqing Fu, Xiya Wu, Qingwei Lin, and Dongmei Zhang. Deeptralog: trace-log combined microservice anomaly detection through graph-based deep learning. In *Proceedings of the 44th International Conference on Software Engineering*, ICSE '22, page 623–634, New York, NY, USA, 2022. Association for Computing Machinery.

[Zheng *et al.*, 2019] Li Zheng, Zhenpeng Li, Jian Li, Zhao Li, and Jun Gao. Addgraph: Anomaly detection in dynamic graph using attention-based temporal gcn. In *IJCAI*, volume 3, page 7, 2019.