BAG: Benchmarking Anomaly Detection on Dynamic Graphs

Fengrui Hua[1,2*], Yiyan Qi[1*], Zikai Wei[1], Yuxing Tian[3], Chengjin Xu[1,4], Xiaojun Wu[1,2], Jia Li[2†], Jian Guo[1†]

[1]IDEA Research, International Digital Economy Academy
[2]Data Science and Analytics Thrust, Hong Kong University of Science and Technology (Guangzhou)
[3]Université de Montréal
[4]DataArc Tech Ltd.
{fhua430,xwu647,jialee}@connect.hkust-gz.edu.cn, yuxing.tian@umontreal.ca,
{huafengrui,qiyiyan,weizikai,xuchengjin,wuxiaojun,guojian}@idea.edu.cn

Abstract

Anomaly detection in dynamic graphs is a critical area of research that focuses on identifying abnormal components within evolving graph structures that deviate significantly from typical patterns. Despite advancements in traditional temporal pattern mining and deep learning techniques, a comprehensive benchmarking framework for Dynamic Graph Anomaly Detection (DyGAD) has been lacking. To address this gap, we introduce **BAG**, the first comprehensive benchmark specifically designed for anomaly detection on dynamic graphs. BAG enables extensive evaluation of 25 leading DyGAD models, covering both classical approaches and advanced Dynamic Graph Neural Networks (DGNNs), across 10 diverse real-world datasets that include both synthetic and naturally occurring anomalies. The framework supports evaluations at both the edge and node levels, offering a robust tool to advance DyGAD research. Our main finding is that Continuous-time Dynamic Graph (CTDG) models demonstrate superior performance and potential in detecting anomalies in dynamic graph edges, compared to Discrete-time Dynamic Graph (DTDG) models. Furthermore, the results reveal that existing methods are less effective at detecting organic anomalies, primarily due to the presence of temporal anomalies and highly imbalanced samples. The proposed BAG benchmark significantly enhances the evaluation of DyGAD methods by improving dataset selection, metric application, and model training. Moreover, BAG supports reproducibility and further exploration in this field by integrating all models, datasets, and evaluation protocols into an open-source repository.

Code — https://github.com/frhuaaa/BAG

1 Introduction

Graph Anomaly Detection (GAD) focuses on identifying unusual graph components, such as nodes, edges, or substructures, that significantly deviate from the norm within a graph. Over the past two decades, GAD has gained considerable attention due to its applicability in various real-world scenarios such as detecting financial fraud (Huang, Yang et al. 2022; Chen and Tsourakakis 2022), identifying money laundering (Dumitrescu, Bălтoiu, and Budulan 2022; Li, Liu et al. 2020), predicting network intrusions (Caville et al. 2022), monitoring device failures (Wu, Dai, and Tang 2021; Chen et al. 2021), identifying spam reviews (Noekhah, binti Salim et al. 2020; Ding, Shu et al. 2021), and uncovering fake news (Gangireddy et al. 2020). Given the dynamic nature of graph data, dynamic graph anomaly detection (DyGAD) extends this concept by incorporating the temporal evolution of the graph, thus providing additional insights into evolving fraudulent patterns. However, this dynamic approach introduces unique challenges, such as modeling temporal dependencies, handling dynamic feature changes, and addressing issues like concept drift, temporal sparsity, and evolving relationships.

The definitions and objectives of DyGAD can vary widely based on specific applications and goals. In this paper, we focus on the prevalent task of identifying anomalous edges and nodes in dynamic graphs. A range of methods have been developed for this purpose, encompassing traditional temporal pattern mining techniques and advanced deep learning approaches. However, several key limitations persist in current practices of model development and evaluation:

- **Lack of comprehensive DyGAD benchmarks:** Existing benchmarks and tools, as summarized in Table 1, mainly focus on static graphs and single task. There is a notable deficiency in benchmarks designed specifically for DyGAD, which impedes effective evaluation and comparison of DyGAD models. This gap can lead to potential biases and incomplete assessments of algorithmic capabilities.
- **Insufficient comparative studies between different methods:** Handling time information in dynamic graphs can vary significantly across different methodologies. Yet, comparative studies that encompass static graph models, Discrete-Time Dynamic Graph (DTDG) methods, and Continuous-Time Dynamic Graph (CTDG) methods are notably scarce. This shortage of comprehensive comparisons hinders our ability to discern the relative strengths and weaknesses of each approach. Consequently, it becomes challenging to determine the most appropriate methods for handling various types of dynamic graph data and specific applications. This limitation restricts the development and refinement of more effective dynamic graph algorithms.

*Equal contribution.
†Corresponding authors.

Benchmark	Datasets	Models	Supervision Scenario	Task
BOND (Liu et al. 2022)	9	Static	Unsupervised	Node-level
GADBench (Tang et al. 2024)	10	Static	Fully- and Semi- Supervised	Node-level
TGB-LAD (Poštuvan, Grohnfeldt et al. 2024)	8	CTDG	Fully- and Self- Supervised	Edge-level
BAG	10	Static, DTDG, CTDG	Fully- and Self- Supervised	Node- and Edge-level

Table 1: Comparison of existing Graph Anomaly Detection benchmarks across datasets, models, and supervision scenarios.

- **Limited exploration on synthetic and organic anomalies within dynamic graphs:** Many studies (Zheng et al. 2019; Liu et al. 2021; Yu, Cheng et al. 2018) rely on synthetic labels created by randomly inserting anomalies due to the limited availability of graph datasets with real-world anomalies. However, these synthetic labels can vastly differ from real-world anomalies, which may compromise the validity of experimental results. Although recent releases of datasets with real-world labels offer new opportunities, it is essential to rigorously test different methods against both synthetic and real-world labels to ensure a thorough evaluation of their performance.

To address these issues, we propose **BAG**, a comprehensive and unified benchmark for dynamic graph anomaly detection. This benchmark encompasses a diverse array of 25 models, including static GNNs, DTDG models, CTDG models, as well as models specifically designed for DyGAD. Our evaluations are conducted on 10 real-world datasets, which vary in complexity and size, ranging from thousands to nearly a million edges, and include both synthetic and organic anomalies. Our findings reveal that, in detecting organic anomalies, CTDG models generally outperform DTDG models. However, the majority of methods are less effective, primarily due to the presence of temporal anomalies and highly imbalanced samples. This highlights a critical area for further improvement. In summary, our contributions are threefold:

- We introduce BAG, which systematically evaluates the dynamic graph anomaly detection task by comparing 25 well-known models across 10 diverse real-world datasets under both self-supervised and fully supervised settings.
- We categorize the properties of anomalies in dynamic graphs, identify limitations in existing DyGAD evaluation schemes, and enhance them through improved dataset selection, metric utilization, and model training.
- We consolidate all models, datasets, and protocols into an open-source repository. This repository enables users to reproduce our results easily and facilitates the evaluation of their own datasets and models with minimal effort.

2 Problem Definition

Dynamic graphs extend static graphs by incorporating temporal information. At any given time t, the graph is represented as $G_t = (\mathcal{V}_t, \mathcal{E}_t, \mathcal{X}_t)$, with $\mathcal{V}_t$, $\mathcal{E}_t$, and $\mathcal{X}_t$ representing nodes, edges, and features at that time. There are two prevalent approaches to representing dynamic graphs: DTDG and CTDG. In DTDG, a time period $T_n = [t_1 : t_n]$ is divided into n time intervals, representing the dynamic graph as a series of snapshots $G_T = \{G_{t_1}, \ldots, G_{t_n}\}$. Each G_{t_i} captures the latest graph structure up to time t_i. For computational efficiency, DTDG models often sample $S << n$ snapshots from G_T during the learning process. In the static setting, this can be seen as a special case where the number of snapshots is just one. Conversely, in the CTDG approach, graph information is processed as a continuous stream of events $G_T = \{e_{t_1}, e_{t_2}, \ldots, e_{t_n}\}$, where $e_{t_i} = \{u, v, t_i\}$ is an edge from node u to node v at time t_i. CTDG maintains a single graph structure at any given time t by incorporating all event stream data to form G_t.

Unlike static graphs where node and edge patterns remain stable, dynamic graphs exhibit temporal variation in structure and behavior. For instance, in transaction networks, fraudsters often imitate legitimate users in the early stages (homophilic behavior) but later engage in anomalous activities that diverge from normal patterns (increased heterophily). To better characterize anomalies in dynamic graphs, we classify them into the following three types:

- **Structural Anomalies** — Nodes or edges that have never appeared in the historical graph but emerge in the current timestamp with anomalous behaviors. These typically correspond to previously unseen structural patterns.
- **Temporal Anomalies** — Nodes or edges that have consistently behaved normally in the past but become anomalous at the current timestamp. They reflect abrupt temporal deviations from previously regular patterns.
- **Persistent Anomalies** — Nodes or edges that were anomalous in previous snapshots and continue to exhibit abnormal behavior. While not temporally novel, their persistence is crucial for long-term anomaly tracking.

This fine-grained typology captures both temporal and structural dimensions of abnormality, offering a comprehensive foundation for evaluating diverse DyGAD methods. In this paper, we focus on the detection of both anomalous edges and nodes. Specifically, given a DTDG or CTDG G_T and a subset of edges $\mathcal{E}'$ or nodes $\mathcal{V}'$, the objective is to develop scoring functions $f(e_t)$ and $f(v_t)$. These functions assign scores to edge $e_t \in \mathcal{E}'$ or node $v_t \in \mathcal{V}'$, identifying it as anomalous if $f(e_t)$ or $f(v_t) > \theta$, as shown in Figure 1.

3 Related Work

Before introducing our benchmark, we offer a concise overview of anomaly detection methods used in dynamic graphs, as well as a review of other advanced models for dynamic graph learning.

3.1 Anomaly Detection in Dynamic Graphs

Edge-level Anomaly Detection Early studies on dynamic edge anomaly detection primarily utilized probabilistic and

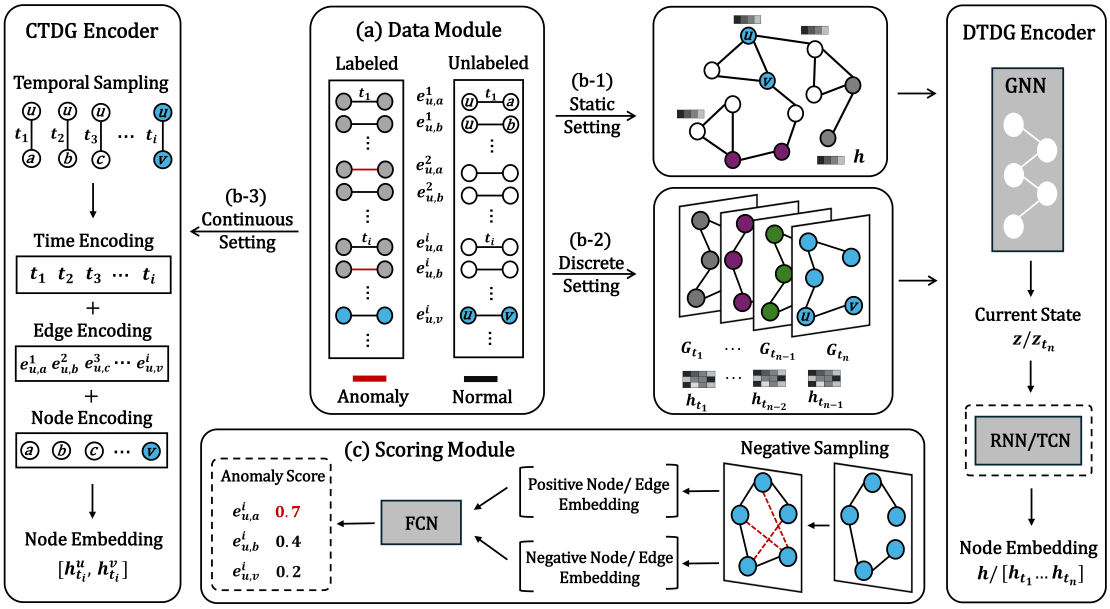

Figure 1: The pipeline of BAG

machine learning models to analyze and represent neighborhood relationships and patterns in dynamic graphs (Aggarwal, Zhao, and Philip 2011; Ranshous, Harenberg et al. 2016; Shah, Beutel et al. 2016; Eswaran and Faloutsos 2018; Yoon, Hooi et al. 2019; Bhatia, Liu et al. 2022; Bhatia et al. 2023). Graph embeddings, which effectively capture network representations, are now integrated with anomaly detection in advanced learning-based methods (Zhu, Ma, and Liu 2020; Li, Zhu et al. 2023; Lou, Zhang et al. 2023). For instance, NetWalk (Yu, Cheng et al. 2018) uses a random walk-based encoder to generate node embeddings and models network evolution through dynamically updating reservoirs. AddGraph (Zheng et al. 2019) constructs an end-to-end neural network to capture the spatial and temporal patterns of dynamic graphs, using GCN as a structural feature extractor and GRU-attention module for combining short-term and long-term dynamic evolutions. More recently, GeneralDyG (Yang, Zhao, and Shen 2025) proposes a generalizable framework by sampling temporal ego-graphs around anomalous events and jointly modeling structural and temporal dependencies, enabling effective detection of both edge-level and node-level anomalies. Comprehensive surveys on these methods can be found in (Ma, Wu et al. 2021; Barros, Mendonça et al. 2021; Ekle and Eberle 2024).

Node-level Anomaly Detection Node-level tasks aim to identify nodes whose features, behaviors, or connectivity deviate significantly from expected patterns in a graph. OCAN (Zheng, Yuan et al. 2019) integrates an LSTM-Autoencoder with a complementary GAN, enabling the generator to produce complementary samples rather than replicating the original data distribution. To address the scarcity of anomaly ground-truth and temporal information, DGraph (Huang, Yang et al. 2022) provides a large-scale, real-world dynamic graph dataset from the financial industry, featuring over 3 million diverse nodes to advance dynamic node anomaly detection research. Meanwhile, SAD (Tian, Dong et al. 2023) leverages temporal graph encoding, a time-equipped memory bank, and pseudo-label contrastive learning to fully utilize unlabeled data and overcome label scarcity. SLADE (Lee et al. 2024) introduces a self-supervised framework that detects dynamic anomalies by minimizing temporal drift and reconstructing long-term patterns, enabling efficient label-free detection in edge streams.

3.2 Dynamic Graph Neural Networks

Existing dynamic graph neural networks can be broadly classified into DTDG models and CTDG models (Longa, Lachi et al. 2023; Yang, Chatelain, and Adam 2024; Zheng, Yi, and Wei 2024; Feng, Wang et al. 2024). DTDG models generate discrete snapshots of graphs and integrate information from these snapshots. Most DTDG models (Zhao et al. 2019; Taheri et al. 2019; Manessi, Rozza, and Manzo 2020; Pareja et al. 2020; Panagopoulos et al. 2021) are combinations of GNNs and recurrent architectures, whereby the former capture graph structure information and the latter handle the dynamic character. However, these methods often lead to information loss due to the time discretization missing crucial interactions. To address this issue, there is a growing interest in CTDG models, which treat dynamic graph data as link streams and continuously learn node representations from ongoing interactions.

Specifically, current CTDG models (Xu et al. 2020; Wang, Lyu et al. 2021; Wang, Chang et al. 2021; Yu et al. 2023) often use recurrent neural networks (RNNs) or self-attention mechanisms as foundational components. Further enhancing their capabilities, some approaches integrate additional tech-

Datasets	#Nodes	#Edges	Unique Edges	Unique Timestep	Feature/Num	Anomaly	Domains	Tasks
Digg	30,398	87,627	86,404	83,943	✗	✗	Social	Edge
UCI	1,899	59,835	20,296	58,911	✗	✗	Email	Edge
Bitcoin-Alpha	3,783	24,186	24,186	1,647	✗	✗	Finance	Edge
Bitcoin-OTC	5,881	35,592	35,592	35,592	✗	✗	Finance	Edge
AS-Topology	34,761	171,403	114,496	32,824	✗	✗	Network	Edge
Email-DNC	1,891	39,264	5,598	19,383	✗	✗	Email	Edge
EU-Core	986	332,334	24,929	207,880	✗	✗	Email	Edge
Wikipedia	9,227	157,474	18,257	152,757	Edge/172	0.14%	Social	Edge&Node
Reddit	10,984	672,447	78,516	669,065	Edge/172	0.05%	Social	Edge&Node
MOOC	7,144	411,749	178,443	345,600	Edge/4	0.99%	Social	Edge&Node

Table 2: Statistics of all datasets in BAG, detailing the number of nodes and edges, unique timestamps, node or edge features, the ratio of anomalous labels, and their respective domains.

niques such as memory networks (Kumar et al. 2019; Trivedi et al. 2019; Rossi et al. 2020), ordinary differential equations (ODE), random walk (RW) strategies (Wang et al. 2021), and temporal point processes (TPP) (Chang, Liu et al. 2020; Huang, Sun, and Wang 2020) to more effectively capture continuous temporal dynamics. Recently, some models (Cong et al. 2023; Tian, Qi, and Guo 2024) employ the MLP-Mixer as their primary architecture, complemented by uniquely designed modules.

4 The Setup of BAG

In this section, we provide an overview of BAG. First, we introduce the benchmark datasets and models employed in our evaluation. Next, Section 4.1 presents the experimental settings in detail. Finally, Section 4.2 outlines the data processing procedures and evaluation metrics adopted in BAG.

Benchmark Datasets In BAG, we compile 10 real-world datasets from various domains, as outlined in Table 2. These datasets cover areas such as social networks, e-commerce, and e-finance, ensuring a comprehensive evaluation across varied application scenarios. To enable a fair comparison of different methods, BAG incorporates datasets of varying scales, with the number of edges ranging from tens of thousands to nearly a million. Our benchmark features two types of anomalies: (1) synthetic anomalies in datasets without ground-truth labels at varying anomaly ratios; and (2) organic anomalies that naturally occur in real-world datasets. We leverage both types to evaluate model performance across diverse anomaly detection scenarios.

Benchmark Models In BAG, we collect 25 diverse and representative models, as shown in Table 3. For parameter settings, we adopt the default configurations used in previous related benchmarks (Rozemberczki et al. 2021; Yu et al. 2023; Tang et al. 2024) or as specified in the original papers.

4.1 Experimental Setting

In the BAG framework, an extensive spectrum of graph neural networks is employed to generate node embeddings Z^t at each timestamp t. To obtain the embedding of an edge $e = \{u, v, t\}$, connecting nodes u and v at time t, we use the Hadamard product $\circ$, which is defined as the element-wise multiplication of two vectors. Thus, the embedding of the edge, $Z_e^t = Z_u^t \circ Z_v^t$, where Z_u^t and Z_v^t are the embeddings of nodes u and v at time t.

Negative Sampling While the labeled dataset in anomaly detection can be formulated as a supervised imbalanced binary classification task, learning from unlabeled samples presents a more formidable challenge. In BAG, we address this by adopting a self-supervised strategy based on negative sampling. Specifically, for each edge $e = \{u, v, t\}$, we randomly select node pairs that do not exist in $\mathcal{E}^t$, thereby creating a negative sample $e' = \{u', v', t\}$.

Anomaly Detector To distinguish abnormal samples from normal samples, we use a two-layer fully connected neural network as the scoring mechanism. The anomaly score for each edge e or node v is computed as:

$$f(x) = W_2 \cdot \text{ReLU}(W_1 \cdot Z_x + b_1) + b_2$$

where Z_x denotes the feature representation of the edge or node x; W_1, W_2 and b_1, b_2 are the weight matrices and bias vectors of the two layers. A higher score indicates a greater likelihood that x is anomalous.

Objective Function For datasets with organic anomalies, we employ the binary cross-entropy loss with logits:

$$\mathcal{L}^t = \sum_{(u,v)\in\mathcal{E}^t} \Big[\log(1 + e^{f(u,v)}) - y_{uv} f(u,v)\Big] + \lambda \mathcal{L}_{\text{reg}}$$

where $f(u, v)$ denotes the scoring function, $y \in \{0, 1\}$ is the binary label (1 for anomalous, 0 for normal). $\mathcal{L}_{\text{reg}}$ is an L2 regularization term weighted by the hyperparameter λ.

For unlabeled datasets, we cannot assume that the generated samples are truly anomalous. Therefore, directly assigning pseudo-labels to these samples would not be methodologically sound. Instead, we adopt the margin-based pairwise loss function from (Zheng et al. 2019), which encourages higher scores for the sampled edges while penalizing normal edges to enhance classification. The overall loss function for unlabeled data is defined as follows:

$$\mathcal{L}^t = \sum_{(u,v)\in\mathcal{E}^t} \sum_{(u',v')\notin\mathcal{E}^t} \max\{0, \gamma + f(u,v) - f(u',v')\}$$

Here, γ is a margin that enforces score separation between normal and sampled edges.

Static GNNs	ChebNet (Defferrard et al. 2016), SGC (Wu et al. 2019) GCN (Kipf and Welling 2017), GIN (Xu et al. 2019) GAT (Veličković et al. 2017), GT (Shi, Huang et al. 2021)
DTDGs	DyGrAE (Taheri et al. 2019), TGCN (Zhao et al. 2019) EvolveGCN-O, EvolveGCN-H (Pareja et al. 2020) MPNNLSTM (Panagopoulos et al. 2021)
CTDGs	JODIE (Kumar et al. 2019), DyRep (Trivedi et al. 2019) TGN (Rossi et al. 2020), TGAT (Xu et al. 2020) CAWN (Wang et al. 2021), TCL (Wang, Chang et al. 2021) GraphMixer (Cong et al. 2023), DyGFormer (Yu et al. 2023) FreeDyG (Tian, Qi, and Guo 2024)
Specialized GNNs	AddGraph (Zheng et al. 2019), SAD (Tian, Dong et al. 2023) RFGraph (Tang et al. 2024), XGBGraph (Tang et al. 2024) SLADE (Lee et al. 2024)

Table 3: Overview of all models used by BAG and their corresponding groupings.

4.2 Other Details

Data Preparation Initially, all edges are sorted by their timestamps. All nodes are reindexed to a contiguous range from 0 to $N-1$, where N is the total number of nodes. In the static setting, the temporal information of each edge is ignored. In the discrete setting, snapshots are created based on varying edge counts across datasets, assuming that each snapshot includes all nodes with dynamic edges. In the continuous setting, the sorted dataset is used directly without further modification.

Data Split For both edge-level and node-level tasks, we chronologically split all datasets into training (50%), validation (20%), and test (30%) sets. For the discrete-time setting, each set is divided into 10, 4, and 6 snapshots, respectively.

Anomaly Injection For datasets lacking ground-truth labels, we refer to the procedure introduced by (Liu et al. 2021). Specifically, nodes are first partitioned into distinct communities via spectral clustering. Anomalous edges are then generated by randomly sampling node pairs from different communities and injecting them into random timestamps within the validation and test sets at three anomaly ratios: 1%, 5%, and 10%.

Evaluation Metrics We select four commonly used metrics for anomaly detection task: Area Under the Precision-Recall Curve (AUPRC), Area Under the Receiver Operating Characteristic Curve (AUROC), Recall at K (Rec@K) and F1 Score. These metrics enable fair comparisons across models and scenarios. The checkpoint with the highest validation AUPRC is used for test evaluation.

5 Experiment Results

In this section, we present and analyze the experimental results of all benchmarked models to address the following research questions:

- **RQ1 (Section 5.1):** How do algorithms perform across organic versus synthetic outliers?
- **RQ2 (Section 5.2):** How do algorithms perform under different properties of anomalies?
- **RQ3 (Section 5.4):** How do algorithms perform in terms of computational efficiency and memory usage?

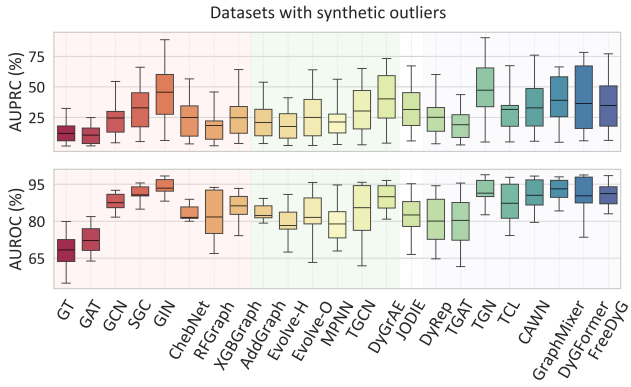

Figure 2: Comparison of AUPRC and AUROC on synthetic edge outliers across all baseline models. Each boxplot represents the distribution of 21 experimental results (7 unlabeled datasets × 3 anomaly ratios) for each model. The central line within each box indicates the mean score.

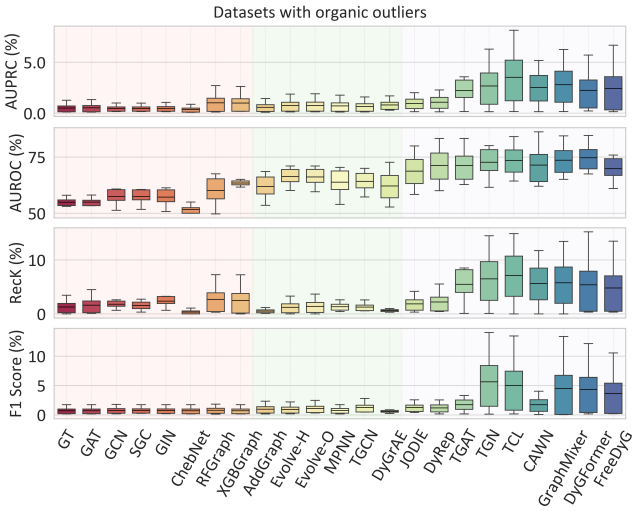

Figure 3: Comparison of anomaly detection performance on organic edge outliers across all baseline models. Each boxplot encompasses the results across three labeled datasets.

5.1 Performance on Synthetic and Organic Edge Anomalies

Synthetic anomalies do not fully capture organic ones. By comparing Figure 2 and Figure 3, it is evident that, regardless of the chosen metric, synthetic anomalies do not accurately reflect model performance on organic anomalies. On synthetic outliers, different types of GNN show relatively small performance gaps, and some static GNNs (e.g., SGC and GIN) even achieve performance comparable to specialized anomaly detection models. However, this apparent competitiveness does not transfer to organic anomalies. When evaluated on organically occurring outliers, nearly all models experience a substantial performance degradation, with the drop being particularly pronounced in AUPRC. This discrepancy indicates that synthetic anomalies fail to capture key

structural and temporal characteristics of real-world anomalies, leading to overly optimistic conclusions about model effectiveness.

Within the class of dynamic GNNs, CTDG approaches such as TGN, GraphMixer, and FreeDyG exhibit superior performance in both scenarios, indicating strong robustness. Beyond that, the average performance of DTDG and CTDG models on synthetic anomalies is relatively similar. DTDG requires adjustment of the number of snapshots for each dataset to achieve optimal performance. An excessive or insufficient number of snapshots typically leads to suboptimal performance of DTDG models. Notably, when detecting organic anomalies, CTDG models consistently outperform DTDG models across the board.

Model performance varies with anomaly ratio. As shown in Figure 4, most evaluation metrics decline sharply as the anomaly ratio decreases, revealing that static GNN, DTDG, and CTDG models are all sensitive to label imbalance in the dataset. Notably, AUROC scores remain relatively stable and do not exhibit a clear downward trend. This highlights a critical limitation: under the extreme class imbalance, AUROC may provide an overly optimistic assessment of model performance, as it may remain high even when all anomalies are not ranked near the top and thus missed during early detection. Therefore, relying solely on AUROC can be misleading. To ensure a more robust and informative evaluation, it is essential to consider multiple metrics that more accurately reflect the model's ability to detect rare anomalies under imbalanced conditions.

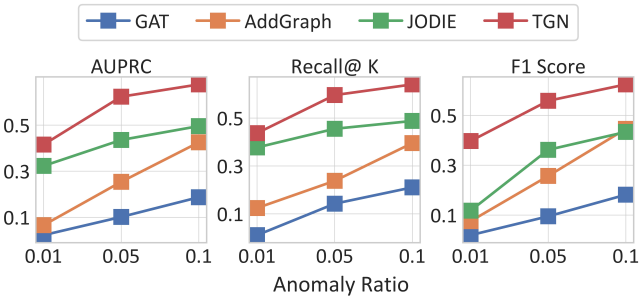

Figure 4: Model performance under varying anomaly injection ratios on the bit-otc dataset.

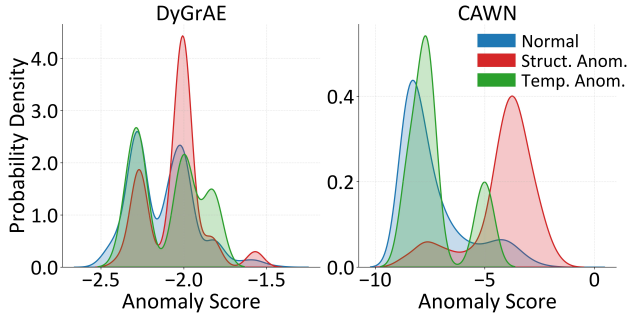

Figure 5: Comparison of DyGrAE and CAWN anomaly score density distributions across different anomaly types on the Wikipedia dataset.

5.2 Performance on Different Properties of Anomalies

Synthetic and organic anomalies exhibit distinct properties. We examine the properties of anomalous edges in each dataset following the definitions provided in Section 2. Synthetic anomalous edges contain only structural anomalies, as they are formed by connecting node pairs from different clusters that do not exist in the historical edge set. In contrast, organic datasets exhibit multiple properties of anomalies, with temporal anomalies being predominant: 82.8% in Reddit, 62.7% in Wiki, and 48.0% in MOOC, whereas the remaining anomalies are structural. Furthermore, the proportion of anomalies in labeled datasets is notably low, typically around 0.1%. Such extreme class imbalance may inflate evaluation metrics in artificially balanced settings and hinder a model's ability to generalize to real-world scenarios.

Most GNNs are highly sensitive to structural anomalies. As synthetic datasets contain only structural anomalies by design, we observe in Section 5.1 that most GNNs, whether static GNNs, DTDG, or CTDG, achieve strong performance in this setting. This suggests that message-passing architectures are naturally sensitive to abrupt changes in graph topology. Structural anomalies directly alter local neighborhood structures, and these changes are amplified through aggregation operations. As a result, structural deviations alone can induce clear embedding shifts, allowing anomalies to be detected even without informative semantic node features. However, attention-based models often underperform their non-attention counterparts. Without informative node features, attention mechanisms tend to favor frequent and structurally consistent patterns, causing anomalous edges fail to induce noticeable perturbations in node embeddings and become relatively hard to detect.

Temporal anomalies make organic anomalies hard to detect. To further investigate why models perform poorly on organic anomalies, we visualize the score distributions assigned by the models to normal edges and different types of anomalous edges. From Figure 5, we observe that the DTDG model DyGrAE struggles to distinguish between normal edges and temporal anomalies, as their score distributions are highly overlapping. Moreover, the presence of temporal anomalies appears to confuse the model, leading to the misclassification of some normal edges as anomalous, which in turn degrades overall detection performance.

However, the CTDG model remains effective in distinguishing normal edges from structural anomalies. Although most of temporal anomalies are still misclassified as normal, a portion of them can be detected, with anomaly scores typically falling between normal edges and structural anomalies.

5.3 Performance on Organic Node Anomalies

For the node-level task, the goal is to identify whether a node becomes anomalous at different time points. Since the dataset is partitioned sequentially by time, using a discrete approach may result in nodes having multiple labels within a

Models	MOOC		Wikipedia		Reddit	
	PRC	AUC	PRC	AUC	PRC	AUC
JODIE	1.79	66.4	0.88	82.3	0.12	56.2
DyRep	1.83	68.4	0.92	84.4	0.13	58.9
TGN	2.44	71.9	1.17	84.7	0.16	61.6
TGAT	1.42	60.3	2.79	79.8	0.14	61.7
TCL	**7.91**	**73.7**	3.19	83.2	0.13	61.1
CAWN	2.74	57.0	2.35	**87.2**	0.17	65.8
GraphMixer	3.05	66.2	1.24	85.2	0.16	64.6
SAD	1.44	60.1	**3.30**	81.0	0.15	63.0
SLADE	1.82	65.3	1.09	86.3	0.16	63.7
DyGFormer	6.40	72.5	1.62	84.3	0.20	67.5
FreeDyG	3.38	69.0	1.63	83.2	**0.25**	**68.2**

Table 4: Comparison of AUPRC and AUROC scores for each model on the node-level task. Best results are highlighted in **bold**, while the second-best are underlined.

single snapshot, preventing the model from effectively learning useful information. However, removing duplicate nodes would lead to an unfair benchmark evaluation. To address this, we exclusively perform node anomaly detection using CTDG. Table 4 shows that TCL attains leading AUPRC results on the MOOC and Wiki datasets, further underscoring the strength of its co-attentional Transformer architecture. Additionally, DyGFormer achieves top-2 performance across most datasets and metrics, demonstrating the effectiveness of its co-occurrence encoding and patching mechanisms.

5.4 Computational Efficiency

We also analyzed the runtime, CPU usage, and GPU memory consumption for the bit-otc dataset with a 5% anomaly ratio across all baseline models. As shown in Figure 6, CTDG models generally require more time and GPU resources than DTDG and static methods.

This overhead primarily stems from the event-wise computation pattern of CTDG: for every interaction, the model must retrieve temporally valid neighbors, compute continuous-time encoding, and update attention/memory states based on event timestamps (Chen, Liao et al. 2023). In contrast, DTDG employs a fixed computation process, sampling only from snapshots at discrete time steps, resulting in a more predictable and limited computational load. Additionally, although RFGraph and XGBGraph do not rely on GPU computation, their runtime is among the lowest, highlighting the efficiency of tree-based algorithms in GAD tasks.

6 Conclusion and Further Directions

In this paper, we introduce a comprehensive and unified benchmark for anomaly detection in dynamic graphs. Through a rigorous comparison across 10 datasets and 25 algorithms in static, discrete, and continuous settings, we demonstrate that continuous-time dynamic graph neural networks exhibit superior performance in detecting organic outliers. However, our experimental results also reveal a significant gap: due to **temporal anomalies** and **highly imbalanced samples**, findings on synthetic data may not transfer effectively to organic anomalies. To address these challenges and advance the field, we identify several research directions:

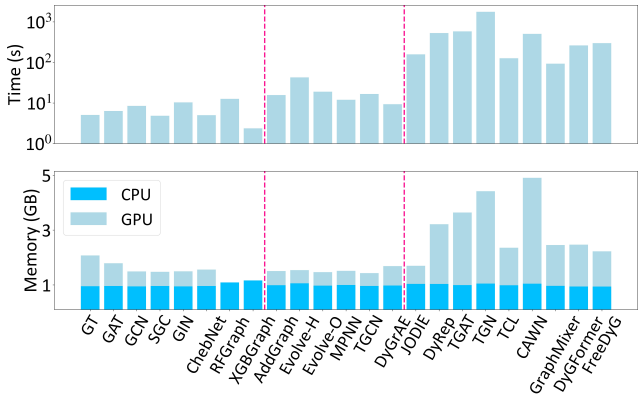

Figure 6: Comparison of all baseline models on the bit-otc dataset (5% anomaly ratio) in terms of runtime and CPU/GPU memory usage. Runtime values have been transformed using $\log_{10}$ for better visualization.

- **Improving synthetic anomaly construction and expanding organic datasets.** Our experiments show that existing synthetic datasets mainly target structural anomalies, often neglecting the joint temporal and structural irregularities characteristic of real-world scenarios. To bridge this gap, it is essential to improve synthetic anomaly generation methods by incorporating both structural and temporal aspects, leading to more realistic datasets. Moreover, collecting and curating additional real-world datasets with genuine anomaly labels will be crucial for rigorous evaluation and further advancement of anomaly detection methods (Liu et al. 2022; Han, Hu et al. 2022).
- **Investigating the potential of CTDG for detecting temporal anomalies.** As discussed in Section 5.2, both CTDG and DTDG methods struggle with organic anomalies, as real-world dynamic graphs are often temporally heterophilic, with connected nodes differing in attributes and labels over time (Xu et al. 2024; Gao, Wang et al. 2023; Tang et al. 2022). Future work could focus on developing models that explicitly capture and leverage these heterophilic relationships to better detect temporal anomalies.
- **Designing more efficient dynamic graph algorithms for streaming data.** While CTDG models generally outperform DTDG models in detecting organic anomalies, they incur substantially higher computational and memory overhead. As real-world applications increasingly involve large-scale, high-velocity graph streams, developing dynamic anomaly detection algorithms that achieve a better balance between detection effectiveness and efficiency remains a critical challenge (Chen et al. 2023).

Our work lays the foundation for systematic research in dynamic GAD and establishes BAG as a valuable community resource. By making BAG open-source, we aim to stimulate further experimentation and drive advances in both academic research and industry applications. We anticipate this initiative will accelerate the development of more effective and robust algorithms for dynamic graphs.

References

Aggarwal, C. C.; Zhao, Y.; and Philip, S. Y. 2011. Outlier detection in graph streams. In *2011 IEEE 27th international conference on data engineering*, 399–409. IEEE.

Barros, C. D.; Mendonça, M. R.; et al. 2021. A survey on embedding dynamic graphs. *ACM Computing Surveys (CSUR)*, 55(1): 1–37.

Bhatia, S.; Liu, R.; et al. 2022. Real-time anomaly detection in edge streams. *ACM Transactions on Knowledge Discovery from Data (TKDD)*, 16(4): 1–22.

Bhatia, S.; et al. 2023. Sketch-based anomaly detection in streaming graphs. In *Proceedings of the 29th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, 93–104.

Caville, E.; et al. 2022. Anomal-E: A self-supervised network intrusion detection system based on graph neural networks. *Knowledge-based systems*, 258: 110030.

Chang, X.; Liu, X.; et al. 2020. Continuous-time dynamic graph learning via neural interaction processes. In *CIKM*.

Chen, C.; et al. 2023. NeutronStream: A Dynamic GNN Training Framework with Sliding Window for Graph Streams. *Proceedings of the VLDB Endowment*, 17(3): 455–468.

Chen, T.; and Tsourakakis, C. 2022. Antibenford subgraphs: Unsupervised anomaly detection in financial networks. In *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, 2762–2770.

Chen, X.; Liao, Y.; et al. 2023. SPEED: Streaming Partition and Parallel Acceleration for Temporal Interaction Graph Embedding. *arXiv preprint arXiv:2308.14129*.

Chen, Z.; et al. 2021. Learning graph structures with transformer for multivariate time-series anomaly detection in IoT. *IEEE Internet of Things Journal*, 9(12): 9179–9189.

Cong, W.; et al. 2023. Do We Really Need Complicated Model Architectures For Temporal Networks? In *ICLR*.

Defferrard, M.; et al. 2016. Convolutional neural networks on graphs with fast localized spectral filtering. *Advances in neural information processing systems*, 29.

Ding, K.; Shu, K.; et al. 2021. Cross-domain graph anomaly detection. *IEEE Transactions on Neural Networks and Learning Systems*, 33(6): 2406–2415.

Dumitrescu, B.; Bălţoiu, A.; and Budulan, Ş. 2022. Anomaly detection in graphs of bank transactions for anti money laundering applications. *IEEE Access*, 10: 47699–47714.

Ekle, O. A.; and Eberle, W. 2024. Anomaly Detection in Dynamic Graphs: A Comprehensive Survey. *ACM Transactions on Knowledge Discovery from Data*.

Eswaran, D.; and Faloutsos, C. 2018. Sedanspot: Detecting anomalies in edge streams. In *2018 IEEE International conference on data mining (ICDM)*, 953–958. IEEE.

Feng, Z.; Wang, R.; et al. 2024. A Comprehensive Survey of Dynamic Graph Neural Networks: Models, Frameworks, Benchmarks, Experiments and Challenges. *arXiv preprint arXiv:2405.00476*.

Gangireddy, S. C. R.; et al. 2020. Unsupervised fake news detection: A graph-based approach. In *Proceedings of the 31st ACM conference on hypertext and social media*, 75–83.

Gao, Y.; Wang, X.; et al. 2023. Addressing heterophily in graph anomaly detection: A perspective of graph spectrum. In *Proceedings of the ACM Web Conference 2023*, 1528–1538.

Han, S.; Hu, X.; et al. 2022. Adbench: Anomaly detection benchmark. *Advances in neural information processing systems*, 35: 32142–32159.

Huang, X.; Yang, Y.; et al. 2022. Dgraph: A large-scale financial dataset for graph anomaly detection. *Advances in Neural Information Processing Systems*, 35: 22765–22777.

Huang, Z.; Sun, Y.; and Wang, W. 2020. Learning Continuous System Dynamics from Irregularly-Sampled Partial Observations. In *NeurIPS*.

Kipf, T. N.; and Welling, M. 2017. Semi-Supervised Classification with Graph Convolutional Networks. In *ICLR*.

Kumar, S.; et al. 2019. Predicting dynamic embedding trajectory in temporal interaction networks. In *Proceedings of the 25th ACM SIGKDD international conference on knowledge discovery & data mining*, 1269–1278.

Lee, J.; et al. 2024. 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*, 1506–1517.

Li, X.; Liu, S.; et al. 2020. Flowscope: Spotting money laundering based on graphs. In *Proceedings of the AAAI conference on artificial intelligence*, volume 34, 4731–4738.

Li, Y.; Zhu, J.; et al. 2023. THGNN: An Embedding-based Model for Anomaly Detection in Dynamic Heterogeneous Social Networks. In *Proceedings of the 32nd ACM International Conference on Information and Knowledge Management*, 1368–1378.

Liu, K.; Dou, Y.; Zhao, Y.; et al. 2022. Bond: Benchmarking unsupervised outlier node detection on static attributed graphs. *Advances in Neural Information Processing Systems*, 35: 27021–27035.

Liu, Y.; et al. 2021. Anomaly detection in dynamic graphs via transformer. *IEEE Transactions on Knowledge and Data Engineering*, 35(12): 12081–12094.

Longa, A.; Lachi, V.; et al. 2023. Graph neural networks for temporal graphs: State of the art, open challenges, and opportunities. *arXiv preprint arXiv:2302.01018*.

Lou, S.; Zhang, Q.; et al. 2023. GADY: Unsupervised Anomaly Detection on Dynamic Graphs. *arXiv preprint arXiv:2310.16376*.

Ma, X.; Wu, J.; et al. 2021. A comprehensive survey on graph anomaly detection with deep learning. *IEEE Transactions on Knowledge and Data Engineering*, 35(12): 12012–12038.

Manessi, F.; Rozza, A.; and Manzo, M. 2020. Dynamic graph convolutional networks. *Pattern Recognition*, 97: 107000.

Noekhah, S.; binti Salim, N.; et al. 2020. Opinion spam detection: Using multi-iterative graph-based model. *Information processing & management*, 57(1): 102140.

Panagopoulos, G.; et al. 2021. Transfer graph neural networks for pandemic forecasting. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, 4838–4845.

Pareja, A.; Domeniconi, G.; Chen, J.; et al. 2020. Evolvegcn: Evolving graph convolutional networks for dynamic graphs. In *Proceedings of the AAAI conference on artificial intelligence*, volume 34, 5363–5370.

Poštuvan, T.; Grohnfeldt, C.; et al. 2024. Learning-Based Link Anomaly Detection in Continuous-Time Dynamic Graphs. *arXiv preprint arXiv:2405.18050*.

Ranshous, S.; Harenberg, S.; et al. 2016. A scalable approach for outlier detection in edge streams using sketch-based approximations. In *Proceedings of the 2016 SIAM international conference on data mining*, 189–197. SIAM.

Rossi, E.; et al. 2020. Temporal Graph Networks for Deep Learning on Dynamic Graphs. *CoRR*, abs/2006.10637.

Rozemberczki; et al. 2021. Pytorch geometric temporal: Spatiotemporal signal processing with neural machine learning models. In *Proceedings of the 30th ACM international conference on information & knowledge management*, 4564–4573.

Shah, N.; Beutel, A.; et al. 2016. Edgecentric: Anomaly detection in edge-attributed networks. In *2016 IEEE 16Th international conference on data mining workshops (ICDMW)*, 327–334. IEEE.

Shi, Y.; Huang, Z.; et al. 2021. Masked Label Prediction: Unified Message Passing Model for Semi-Supervised Classification. In *IJCAI*.

Taheri, A.; et al. 2019. Predictive temporal embedding of dynamic graphs. In *Proceedings of the 2019 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining*, 57–64.

Tang, J.; Li, J.; Gao, Z.; and Li, J. 2022. Rethinking graph neural networks for anomaly detection. In *International Conference on Machine Learning*, 21076–21089. PMLR.

Tang, J.; et al. 2024. Gadbench: Revisiting and benchmarking supervised graph anomaly detection. *Advances in Neural Information Processing Systems*, 36.

Tian, S.; Dong, J.; et al. 2023. Sad: Semi-supervised anomaly detection on dynamic graphs. *arXiv preprint arXiv:2305.13573*.

Tian, Y.; Qi, Y.; and Guo, F. 2024. FreeDyG: Frequency Enhanced Continuous-Time Dynamic Graph Model for Link Prediction. In *The Twelfth International Conference on Learning Representations*.

Trivedi, R. S.; et al. 2019. DyRep: Learning Representations over Dynamic Graphs. In *ICLR*.

Veličković, P.; et al. 2017. Graph attention networks. In *ICLR*.

Wang, L.; Chang, X.; et al. 2021. TCL: Transformer-based Dynamic Graph Modelling via Contrastive Learning. *CoRR*, abs/2105.07944.

Wang, X.; Lyu, D.; et al. 2021. APAN: Asynchronous Propagation Attention Network for Real-time Temporal Graph Embedding. In *SIGMOD*.

Wang, Y.; et al. 2021. Inductive Representation Learning in Temporal Networks via Causal Anonymous Walks. In *ICLR*.

Wu, F.; et al. 2019. Simplifying graph convolutional networks. In *ICML*, 6861–6871.

Wu, Y.; Dai, H.-N.; and Tang, H. 2021. Graph neural networks for anomaly detection in industrial internet of things. *IEEE Internet of Things Journal*, 9(12): 9214–9231.

Xu, D.; et al. 2020. Inductive representation learning on temporal graphs. In *ICLR*.

Xu, F.; Wang, N.; Wu, H.; et al. 2024. Revisiting graph-based fraud detection in sight of heterophily and spectrum. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, 9214–9222.

Xu, K.; Hu, W.; Leskovec, J.; and Jegelka, S. 2019. How powerful are graph neural networks? *ICLR*.

Yang, L.; Chatelain, C.; and Adam, S. 2024. Dynamic graph representation learning with neural networks: A survey. *IEEE Access*, 12: 43460–43484.

Yang, X.; Zhao, X.; and Shen, Z. 2025. A generalizable anomaly detection method in dynamic graphs. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 39, 22001–22009.

Yoon, M.; Hooi, B.; et al. 2019. Fast and accurate anomaly detection in dynamic graphs with a two-pronged approach. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, 647–657.

Yu, L.; et al. 2023. Towards Better Dynamic Graph Learning: New Architecture and Unified Library. In *Thirty-seventh Conference on Neural Information Processing Systems*.

Yu, W.; Cheng, W.; et al. 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, L.; et al. 2019. T-gcn: A temporal graph convolutional network for traffic prediction. *IEEE transactions on intelligent transportation systems*, 21(9): 3848–3858.

Zheng, L.; et al. 2019. AddGraph: Anomaly Detection in Dynamic Graph Using Attention-based Temporal GCN. In *IJCAI*, volume 3, 7.

Zheng, P.; Yuan, S.; et al. 2019. One-class adversarial nets for fraud detection. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, 1286–1293.

Zheng, Y.; Yi, L.; and Wei, Z. 2024. A survey of dynamic graph neural networks. *arXiv preprint arXiv:2404.18211*.

Zhu, D.; Ma, Y.; and Liu, Y. 2020. A flexible attentive temporal graph networks for anomaly detection in dynamic networks. In *2020 IEEE 19th International Conference on Trust, Security and Privacy in Computing and Communications (TrustCom)*, 870–875. IEEE.