A Generalizable Anomaly Detection Method in Dynamic Graphs

Xiao Yang[1,2,3], **Xuejiao Zhao**[1,2*], **Zhiqi Shen**[1*]

[1] College of Computing and Data Science, Nanyang Technological University (NTU), Singapore
[2] Joint NTU-UBC Research Centre of Excellence in Active Living for the Elderly (LILY), NTU, Singapore
[3] Joint NTU-WeBank Research Centre on Fintech, NTU, Singapore
yangxiao.cs@gmail.com, {xjzhao,zqshen}@ntu.edu.sg

Abstract

Anomaly detection aims to identify deviations from normal patterns within data. This task is particularly crucial in dynamic graphs, which are common in applications like social networks and cybersecurity, due to their evolving structures and complex relationships. Although recent deep learning-based methods have shown promising results in anomaly detection on dynamic graphs, they often lack of generalizability. In this study, we propose GeneralDyG, a method that samples temporal ego-graphs and sequentially extracts structural and temporal features to address the three key challenges in achieving generalizability: Data Diversity, Dynamic Feature Capture, and Computational Cost. Extensive experimental results demonstrate that our proposed GeneralDyG significantly outperforms state-of-the-art methods on four real-world datasets.

Code — https://github.com/YXNTU/GeneralDyG

Introduction

Graphs are extensively employed to model complex systems across various domains, such as social networks (Wang et al. 2019), human knowledge networks (Ji et al. 2021), e-commerce (Qu et al. 2020), and cybersecurity (Gao et al. 2020). Although the bulk of researches focus on static graphs, real-world graph data often evolves over time (Skarding, Gabrys, and Musial 2021). Taking knowledge networks as an example, there is new knowledge being added to the network every month, with connections between different concepts evolving over time. To model and analyze graphs where nodes and edges change over time, mining dynamic graphs gains increasing popularity in the graph analysis. Anomaly detection in dynamic graphs (Ma et al. 2021; Ho, Karami, and Armanfard 2024) is vital for identifying outliers that significantly deviate from normal patterns such as anomalous edges or anomalous nodes,including the detection of fraudulent transactions, social media spam, and network intrusions (Dou et al. 2020). By utilizing the temporal information and relational structures inherent in dynamic graphs, researchers can more effectively identify anomalies, thereby enhancing the security and integrity of various systems (Pourhabibi et al. 2020).

Recently, techniques based on deep learning have facilitated significant advancements in anomaly detection within dynamic graphs. For example, methods like GDN (Deng and Hooi 2021), StrGNN (Cai et al. 2021) focus on extracting structural information from graphs, while approaches such as LSTM-VAE (Park, Hoshi, and Kemp 2018) and TADDY (Liu et al. 2021) concentrate on capturing temporal information.In addition, self-supervised (Lee, Kim, and Shin 2024) and semi-supervised (Tian et al. 2023) methods have also been applied to dynamic graph anomaly detection.

Despite their improved performance, current deep learning-based methods lack the crucial generalizability (Brennan 1992) needed for dynamic graph tasks across different tasks or datasets. A model with strong generalization can adapt to different tasks without significant adjustments to its architecture or parameters, reducing the need for retraining or redesigning for new tasks(Bai, Ling, and Zhao 2022). Conversely, in anomaly detection, where identifying potential risks or issues is crucial, poor generalization may lead to missed critical anomalies in new scenarios, thereby diminishing the model's reliability in real-world applications. Specifically, the inadequate encoding of anomalous events[1] in existing methods results in poor generalization. Firstly, in the absence of raw event attributes, they fail to generate informative event encodings that accurately represent the properties of the events. For example, SimpleDyG (Wu, Fang, and Liao 2024) nearly discards all topological structure information, tokenizing only the nodes while ignoring the edges, which leads to the loss of critical structural information during node prediction tasks, making it unsuitable for node anomaly detection tasks and even less so for edge anomaly detection. The positional encoding method in TADDY (Liu et al. 2021) may not capture structural similarities and could fail to model the structural interactions between events, as demonstrated in SAT (Chen, O'Bray, and Borgwardt 2022). TADDY's node position-specific encoding may result in ambiguous structural information, leading to suboptimal results in node anomaly detection tasks. Furthermore, some methods, such as GDN (Deng and Hooi 2021), exhibit inadequate temporal information capture capabilities. For instance, GDN does not incorpo-

*Corresponding Author

[1] In this paper, both node anomalies and edge anomalies are collectively referred to as anomalous events.

rate the information provided by specific time values when modeling temporal data, resulting in poor performance on time-sensitive datasets such as Bitcoin-Alpha and Bitcoin-OTC (Liu et al. 2021).

Developing a highly generalizable dynamic graph anomaly detection method presents several challenges, primarily in: 1. **Data Diversity**: Differences across dynamic graph datasets, such as topological structures and node and edge attributes, can be substantial. The method must identify and adapt to a wide range of feature distributions. 2. **Dynamic Feature Capture**: Anomalies in dynamic graphs may occur locally (e.g., anomalous behavior of specific nodes or edges) or globally (e.g., abnormal changes in network topology). The method must capture both local and global dynamic features. 3. **Computational Cost**: Dynamic graph anomaly detection often involves large-scale graph data, making computational resources and time efficiency significant challenges.

Hence, in this work, we propose a novel approach for anomaly detection named GeneralDyG, which addresses the three key challenges mentioned above and ensures generalizability in dynamic graph anomaly detection tasks. It ensures simplicity by sampling ego-graphs around anomalous events, then uses a novel GNN extractor to capture structural information, and finally employs a Transformer module to capture temporal information. Specifically, the main contributions of our work are:

- We design a novel GNN extractor, which embeds nodes, edges, and topological structures into the feature space. By alternating the message-passing perspective between nodes and edges, it performs graph convolution on both simultaneously. This ensures that GeneralDyG adapts to diverse feature distributions.
- We introduce special tokens into the feature sequences to distinguish the hierarchical relationships between anomalous events, ensuring that the method captures global temporal information while maintaining focus on local dynamic features.
- We design a novel ego-graph[2] sampling method for training anomalous events instead of using the entire graph. This approach significantly reduces computational resources, enhancing the overall efficiency of the method.
- We demonstrate the effectiveness of GeneralDyG on four benchmark datasets for detecting anomalous events, showing that it achieves better performance than state-of-the-art anomaly detection methods.

Related Work

Anomaly Detection in Dynamic Graphs

Anomalies are infrequent observations that significantly deviate from the rest of the sample, such as data records or events. Dynamic graph anomaly detection primarily focuses on identifying unusual events within a dynamic graph (Ekle and Eberle 2024; Ho, Karami, and Armanfard 2024; Ma et al. 2021). Recently, deep learning methods have made significant advancements in anomaly detection for dynamic graphs. Modeling time series-related tasks as anomalous node detection in dynamic graphs is considered a viable approach (Su et al. 2019; Chen et al. 2022; Zhang, Zhang, and Tsung 2022; Dai and Chen 2022). Specifically, M-GAT employs a multi-head attention mechanism along with two relational attention modules—namely, intra-modal and inter-modal attention—to explicitly model correlations between different modalities (Ding, Sun, and Zhao 2023). MTAD-GAT incorporates two parallel graph attention layers to capture the complex dependencies in multivariate time series across both temporal and feature dimensions (Zhao et al. 2020). GDN integrates structural learning with graph neural networks and leverages attention weights to enhance the explainability of detected anomalies (Deng and Hooi 2021). FuSAGNet optimizes reconstruction and forecasting by combining a Sparse Autoencoder with a Graph Neural Network to model multivariate time series relationships and predict future behaviors (Han and Woo 2022).

Detection of edge anomalies in dynamic graphs has also garnered increasing attention. Classical methods include the randomized algorithm SEDANSPOT (Eswaran and Faloutsos 2018) and the hypothesis-based approach Midas (Bhatia et al. 2020). Many recent methods have employed discrete approaches to address this task. For instance, Addgraph utilizes a GCN to extract graph structural information from slices, followed by GRU-attention (Zheng et al. 2019). StrGNN extracts h-hop closed subgraphs centered on edges and employs GCN to model structural information on snapshots, with GRU capturing correlations between snapshots (Cai et al. 2021). Recently, SAD introduced a continuous dynamic approach for anomaly detection using a semi-supervised method (Tian et al. 2023).

Transformer on Dynamic Graphs

Transformers are a type of neural network that rely exclusively on attention mechanisms to learn representative embeddings for various types of data, as initially introduced in (Vaswani et al. 2017). Recent works have also applied Transformers to dynamic graph tasks. For instance, GraphERT pioneers the use of Transformers to seamlessly integrate graph structure learning with temporal analysis by employing a masked language model on sequences of graph random walks (Beladev et al. 2023). GraphLSTA captures the evolution patterns of dynamic graphs by effectively extracting and integrating both long-term and short-term temporal features through a recurrent attention mechanism (Gao et al. 2023). Taddy employs a Transformer to handle diffusion-based spatial encoding, distance-based spatial encoding, and relative time encoding, subsequently deriving edge representations through a pooling layer to calculate anomaly scores (Liu et al. 2021). SimpleDyG reinterprets dynamic graphs as a sequence modeling problem and presents an innovative temporal alignment technique. This approach not only captures the intrinsic temporal evolution patterns of dynamic graphs but also simplifies their modeling process (Wu, Fang, and Liao 2024).

[2]The subgraph that consists of the ego events and all events within the k-hop range from the ego event.

Preliminaries

Notations. A continuous-time dynamic graph (CTDG) is used to represent relational data in evolving systems. A CTDG is defined as $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, where $\mathcal{V}$ is the set of nodes that participate in temporal edges, and $\mathcal{E}$ is a chronologically ordered series of edges. Each edge $\delta(t) = (v_i, v_j, t, e_{ij})$ represents an interaction from node v_i to node v_j at time t with an associated feature e_{ij}. The node attributes for nodes $v_i, v_j \in \mathcal{V}$ are denoted by $x_{v_i}, x_{v_j} \in \mathbb{R}^d$, and the node attributes for all nodes are stored in $\mathcal{X} \in \mathbb{R}^{n \times d}$. Additionally, the edge attributes for edges $e_{ij} \in \mathcal{E}$ are denoted by $y_{e_{ij}} \in \mathbb{R}^d$, and the edge attributes for all edges are stored in $\mathcal{Y} \in \mathbb{R}^{m \times d}$, where n is the number of nodes and m is the number of edges in the CTDG. In this paper, we explore a method called GeneralDyG for handling node-level and edge-level anomalies. Therefore, in the following text, we treat nodes $\mathcal{V}$ and edges $\mathcal{E}$ collectively as anomaly events $\mathcal{A}$. Similarly, we consider node features $\mathcal{X}$ and edge features $\mathcal{Y}$ together as anomaly features $\mathcal{Z}$.

Transformer on CTDG. While Graph Neural Networks (GNNs) directly leverage the inherent structure of graphs, Transformers take a different approach by inferring relationships between nodes using their attributes rather than the explicit graph structure (Dwivedi and Bresson 2020). Transformer treats the dynamic graph as a collection of edges, utilizing the self-attention mechanism to identify similarities between them. The architecture of the Transformer(Fang et al. 2023a,b) consists of two fundamental components: a self-attention module and a feed-forward neural network.

In the self-attention module, the input anomaly features $\mathcal{Z}$ are initially projected onto the query (Q), key (K), and value (V) matrices through linear transformations, such that $Q = \mathcal{Z}W_Q$, $K = \mathcal{Z}W_K$, and $V = \mathcal{Z}W_V$, respectively. The self-attention can then be computed as follows:

$$\text{Attn}(\mathcal{Z}) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_{\text{out}}}}\right)V \in \mathbb{R}^{(m+n)\times d_{\text{out}}}. \quad (1)$$

To address dynamic graph tasks, multiple Transformer layers can be stacked to build a model that provides node-level representations of the graph (Wang et al. 2021). However, due to the permutation invariance of the self-attention mechanism, the Transformer generates identical representations for nodes with the same attributes, regardless of their positions or surrounding structures within the graph. This characteristic necessitates the incorporation of positional and contextual information into the Transformer, typically achieved through positional encoding (Cong et al. 2021; Sun et al. 2022).

Absolute encoding. Absolute encoding involves adding or concatenating positional or structural representations of the graph to the input node features before feeding them into the main Transformer model. Examples of such encoding methods include Laplacian positional encoding (Dwivedi and Bresson 2020), Random Walk Positional Encoding (Dwivedi et al. 2021), and Node Encoding (Liu et al. 2021). A key limitation of these methods is that they typically fail to capture the structural similarity between nodes and their neighborhoods, thereby not effectively leveraging the graph's structural information.

Problem Definition. The goal of this paper is to detect anomalous edges and nodes at each timestamp. Based on the previously mentioned notations, we model anomaly detection in dynamic graphs as a task of computing anomaly scores.

Definition 1. Given a dynamic graph $\mathcal{G}$, where each $\mathcal{G}_t = (\mathcal{V}_t, \mathcal{E}_t)$ represents the graph at timestamp t, the goal of anomaly detection is to identify unusual edges and nodes within this evolving structure. For each edge $e \in \mathcal{E}_t$ and each node $v \in \mathcal{V}_t$, the objective is to compute an anomaly score $f(e)$ and $f(v)$, respectively, where f is a learnable anomaly score function. The anomaly score quantifies the degree of abnormality for both edges and nodes, with a higher score $f(e)$ or $f(v)$ indicating a greater likelihood of anomaly for edge e or node v.

Building on previous research, we adopt an unsupervised approach for anomaly detection in dynamic graphs. During training, all edges and nodes are considered normal. Binary labels indicating anomalies are provided during the testing process to assess the performance of the algorithms. Specifically, a label $y_e = 1$ signifies that e is anomalous, whereas $y_e = 0$ denotes that e is normal. Similarly, a label $y_n = 1$ indicates that a node is anomalous. It is important to note that anomaly labels are often imbalanced, with the number of normal edges and nodes typically being much greater than the number of anomalous ones.

Methodology

In this section, we introduce the general framework of our approach, which consists of three main components: Temporal ego-graph sampling, Temporal ego-graph GNN extractor, and Temporal-Aware Transformer. An overview of the proposed framework is illustrated in Figure 1. Initially, we extract ego-graphs at the level of each anomaly event, capturing k-hop temporal dynamics. These temporal ego-graphs are then transformed into anomaly feature sequences, preserving their temporal and structural order, as demonstrated in Figure 1(a). To fully understand the structural information of these sequences, they are processed through a GNN model to extract the structural details of the temporal ego-graphs, as depicted in Figure 1(b). Finally, both the original sequence features and the structure-enriched sequence features are fed into the Transformer to evaluate the anomaly detection task, as shown in Figure 1(c).

Temporal ego-graph sampling

Unlike conventional methods that map dynamic graphs into a series of snapshots to obtain tokens, we use a more lightweight approach by employing anomalous events as tokens for the Transformer. Additionally, to acquire the contextual representation and hierarchical information of these anomalous events, we extract the temporal k-hop ego-graph of each event to capture historical interaction information across different structures.

Specifically, we denote $a_i \in \mathcal{A}$ as an event in $\mathcal{G}$. For each event a_i, we utilize a k-hop algorithm to extract the historically interacted events and construct

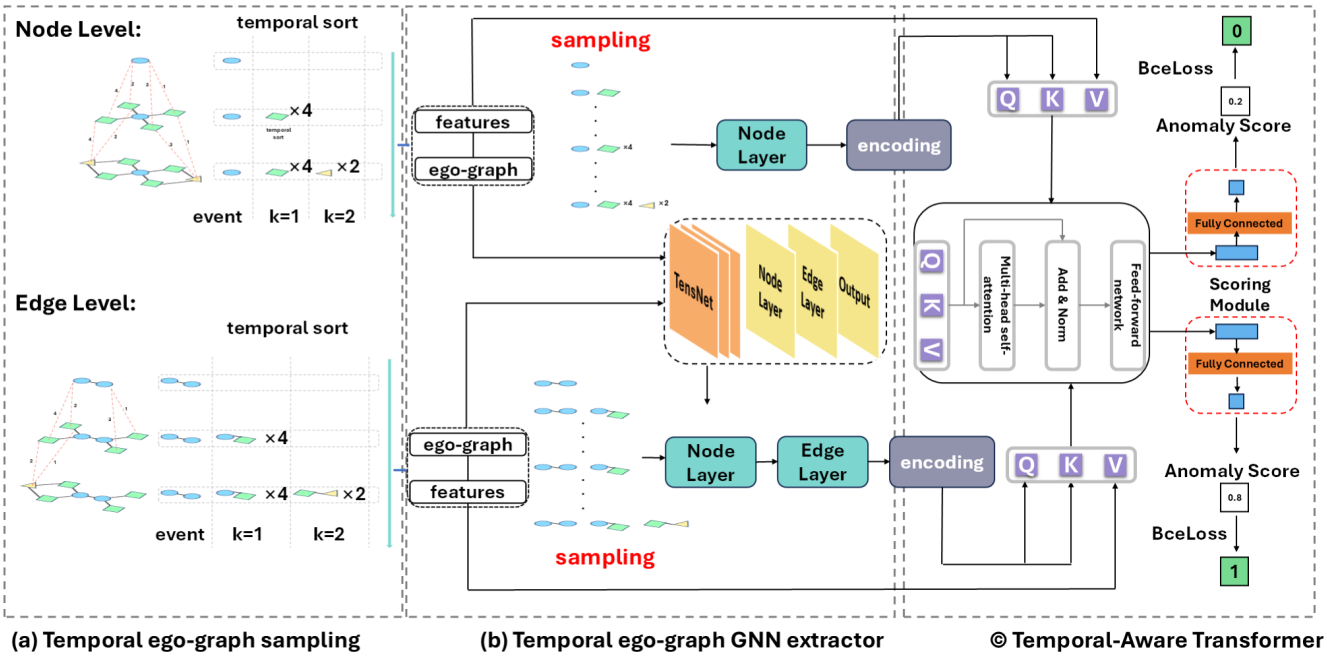

Figure 1: The proposed generalizable anomaly detection framework. GeneralDyG consists of three main components: (a)Temporal ego-graph sampling. (b)Temporal ego-graph GNN extractor. (c)Temporal-Aware Transformer

a series of k-hop ego-graphs centered around a_i, representing subsets of the largest k-hop ego-graph. Explicitly, we denote the temporal k-hop ego-graph for a_i as a chronologically ordered sequence $w_i = sampling(\langle a_i^1\rangle, \langle a_i^1, a_i^2, a_i^3\rangle, \ldots, \langle a_i^1, a_i^2, a_i^3, \ldots, a_i^{|w_i|}\rangle)$, where $|w_i|$ is the number of previously interacting events, and the maximum value of $|w_i|$ is the total number of events that have interacted with a_i. Note that $\forall 1 \leq j < j' \leq |w_i|$, a_i^j and $a^{j'}i$ represent historical interactions (a_i, a_i^j, ei, j) and $(a_i, a^{j'}i, ei, j')$, respectively, such that $e_{i,j} \leq e_{i,j'}$.

When implementing feature sequences sorted by time, it is crucial to simultaneously consider the hierarchical information introduced by the k-hop algorithm. Specifically, for the central event a_i, the set of events $a_{i;k}^j$ extracted by the k-hop algorithm exhibits greater similarity compared to the set of events $a_{i;k+1}^j$ extracted by the $(k+1)$-hop algorithm, as they share the same shortest path to the central point a_i. To better capture this hierarchical information, we draw inspiration from natural language processing methods and add special tokens to the feature sequence. These tokens ensure that the event sets between two special tokens maintain a chronological order. During training, the Transformer module and GNN extractor can receive the following input:

$$\text{input}_i = \langle |KHS| \rangle, a_i, \langle |KHS| \rangle, ai; 1^1, a_{i;1}^2, \ldots, a_{i;1}^{|a_{i;1}|}, \ldots \langle |KHS| \rangle, a_{i;k}^1, a_{i;k}^2, \ldots, a_{i;k}^{|a_{i;k}|}, \langle |KHS| \rangle, \quad (2)$$

where $\langle |KHS| \rangle$ is a special token signifying the beginning and end of the input hierarchical sequence. Specifically, adding such special tokens helps the model recognize and differentiate between the hierarchical layers of the ego-graph. It should be noted that we use sampled ego-graphs here to enhance the model's generalization capability. Therefore, the raw features obtained by the Transformer module and GNN extractor are denoted as z_i, where z_i is a subset of input$_i$.

Temporal ego-graph GNN extractor

A practical approach to extracting local structural information at an event a_i is to apply an existing GNN model to the input graph with event feature sequences z_i, and utilize the output representation at a_i as the ego-graph representation $\varphi(z_i)$. It is important to highlight that, to showcase the flexibility of our model, the GNN model employed here should be both straightforward and capable of simultaneously processing node features $\mathcal{X}$ and edge features $\mathcal{Y}$. Formally, we denote the selected GNN model with $\mathcal{K}$ layers applied to k-hop ego-graphs k-DG as $\text{GNN}_k^{\mathcal{K}}$. The output representation $\varphi(z_i)$ can be expressed as:

$$\varphi(z_i) = \text{GNN}_k^{\mathcal{K}}(z_i). \quad (3)$$

Next, we discuss the choice of the $\text{GNN}_k^{\mathcal{K}}$ model. When the dataset information is known prior to anomaly event prediction—such as in cases where the CTDG consists solely of node features—a conventional GNN model like GCN, GAT, or GIN can be effectively utilized to extract ego-graph structural information. However, for CTDGs with diverse attributes, including both node and edge features, we introduce the Temporal Edge-Node Based Structure Extractor GNN (TensGNN). TensGNN is specifically designed to

accommodate more complex scenarios by concurrently processing both types of features.

TensGNN encodes events by alternately applying node and edge layers, thereby embedding events into a shared feature space. Specifically, TensGNN employs operations analogous to spectral graph convolution for message passing on events. The node Laplacian-adjacency matrix with self-loops is defined as:

$$\bar{A_v} = D_v^{\frac{1}{2}} \left(A_v + I_v\right) D_v^{\frac{1}{2}}, \tag{4}$$

where D_v is the diagonal degree matrix of $A_v + I_v$, and I_v is the identity matrix. The node-level propagation rule for node features in the $(K+1)$-th layer is defined as:

$$H_v^{(K+1)} = \sigma\left(T^T H_e^{(K)} W_e' \odot \bar{A}_v H_v^{(K)} W_v\right), \tag{5}$$

where σ represents the activation function, the matrix $T \in \mathbb{R}^{N_v \times N_e}$ is a binary transformation matrix, with T_{ij} indicating whether edge j connects to node i. The symbol $\odot$ represents the Hadamard product. W_e' and W_v are learnable parameters for edges and nodes, respectively. Similarly, the Laplacianized edge adjacency matrix is defined as:

$$\bar{A_e} = D_e^{\frac{1}{2}} \left(A_e + I_e\right) D_e^{\frac{1}{2}}, \tag{6}$$

where D_e is the diagonal degree matrix of $A_e + I_e$, and I_e is the identity matrix. The propagation rule for edge features is then defined as:

$$H_e^{(K+1)} = \sigma\left(T^T H_v^{(K)} W_v' \odot \bar{A}_e H_e^{(K)} W_e\right). \tag{7}$$

Here, the matrix T is defined analogously to that in Equation 5, with W_v' and W_e representing the learnable weights for the nodes and edges, respectively. TensGNN alternates between stacking node layers and edge layers to iteratively refine the embeddings of both types of events. Specifically, to derive the final encoding of nodes, the last layer before the output is a node layer. Conversely, to obtain the final encoding of edges, the last layer before the output is an edge layer.

Temporal-Aware Transformer

To enhance the Transformer's understanding of the topological structure of the temporal ego-graph while preserving the original event features, we overlay the topological structure information onto the Query and Key, while retaining the original event features as the Value. This approach allows the model to leverage structural information for the attention mechanism while maintaining the integrity of the original feature values for effective representation. Formally, for the event feature to be predicted, $z_i \in \mathcal{Z}$, we adopt the method proposed in (Mialon et al. 2021) and rewrite the self-attention as kernel smoothing. The final embedding calculation is then given by:

$$\text{Attn}(z_i) = \sum_{z_j \in k-DG} \frac{\mathcal{F}_{\exp}(z_i, z_j)}{\sum_{z_w \in k-DG} \mathcal{F}_{\exp}(z_i, z_w)} \mathrm{w}_V z_j, \tag{8}$$

where w_V is the linear value function of the original event feature z_i, and $\mathcal{F}exp$ is an exponential kernel (non-symmetric), parameterized by w_Q and w_K:

$$\begin{aligned} \mathcal{F}exp(x, x') &:= \exp\left(\frac{\langle \mathrm{w}_Q x, \mathrm{w}_K x' \rangle}{\sqrt{d_{out}}}\right), \\ \mathrm{w}_V &= \mathrm{W} z_i + b, \\ \mathrm{w}_Q &= \mathrm{W}\varphi(z_i) + b, \\ \mathrm{w}_K &= \mathrm{W}\varphi(z_i) + b, \end{aligned} \tag{9}$$

where $\langle \cdot, \cdot \rangle$ denotes the dot product. By optimizing the objective function, we obtain the final embeddings for each anomaly feature z_i. These final embeddings are then fed into the scoring module to compute the anomaly scores. It is important to note that the scoring modules for node-level and edge-level anomalies differ in the datasets used in this paper. For edge-level anomalies, we directly use the final output embedding from the training process as the anomaly score. Conversely, for node-level anomalies, the final output consists of a set of binary labels indicating whether each time step is anomalous, which serves as the final anomaly score.

Experiments

Experimental Setup

Datasets. We use four real-world datasets, categorized into two types: Node-Level and Edge-Level anomaly detection tasks. For Node-Level, we utilize SWaT (Secure Water Treatment), a small-scale Cyber-Physical system managed by Singapore's Public Utility Board, and WADI (Water Distribution), an extension of SWaT that includes a more extensive water distribution network. Both datasets provide data from normal operations and controlled attack scenarios to simulate real-world anomalies. For Edge-Level, we employ Bitcoin-Alpha and Bitcoin-OTC, which are trust networks of Bitcoin users trading on platforms from www.btc-alpha.com and www.bitcoin-otc.com, respectively. In these datasets, nodes represent users, and edges indicate trust ratings between them, capturing interactions and trust dynamics within the Bitcoin trading community.

Experimental Design. In our experiments, The settings for Bitcoin-Alpha and Bitcoin-OTC are identical to those used in TADDY (Liu et al. 2021). We inject anomalies into the test set at proportions of 1%, 5%, and 10%. SWaT and WADI are identical to those used in GDN (Deng and Hooi 2021). AUC[3], AP[4] and F1[5] are used as the primary metrics to evaluate the performance of the proposed GeneralDyG and baselines.

Baselines. We evaluated GeneralDyG against 20 advanced baselines, which are classified into two categories: graph embedding methods and anomaly detection methods. A detailed description of the baselines can be found in the Appendix.

- **Graph Embedding Methods:** node2vec (Grover and Leskovec 2016), DeepWalk (Perozzi, Al-Rfou, and

[3]https://en.wikipedia.org/wiki/AUC

[4]https://builtin.com/articles/mean-average-precision

[5]https://en.wikipedia.org/wiki/F-score

Methods	Bitcoin-Alpha						Bitcoin-OTC					
	1%		5%		10%		1%		5%		10%	
	AUC	AP	AUC	AP	AUC	AP	AUC	AP	AUC	AP	AUC	AP
node2vec	69.10	9.17	68.02	7.31	67.85	9.95	69.51	8.31	68.83	6.45	67.45	4.77
DeepWalk	69.85	8.56	68.74	9.68	67.93	10.78	74.23	10.58	73.56	9.41	72.87	8.22
TGAT	85.32	11.36	84.16	11.08	83.98	12.05	88.87	16.87	87.59	15.24	87.55	15.37
TGN	86.92	13.00	86.78	16.85	86.21	17.00	84.33	11.33	83.49	11.25	83.47	10.79
ADDGRAPH	83.41	13.21	84.70	13.01	83.69	14.28	86.00	16.04	84.98	15.21	84.77	14.21
StrGNN	85.74	12.56	86.67	13.99	86.27	14.68	90.12	18.34	87.75	18.68	88.36	18.10
TADDY	**94.51**	16.51	93.41	18.32	94.23	19.67	94.55	16.10	93.40	18.47	94.25	18.92
SAD	90.69	19.99	90.55	21.08	90.33	22.99	91.88	26.32	90.99	27.33	90.04	26.79
SLADE	90.32	18.78	89.99	22.02	88.71	24.41	91.53	20.32	91.24	22.11	91.01	20.04
GeneralDyG	94.01	**24.00**	**95.41**	**24.02**	**96.28**	**26.73**	**94.66**	**27.89**	**94.86**	**29.97**	**95.59**	**27.13**

Table 1: Anomaly detection performance comparison on Edge-Level datasets. The best performing method in each experiment is in bold and the second-best method is indicated with underlining.

Skiena 2014), TGAT (Xu et al. 2020), TGN (Rossi et al. 2020).

- **Anomaly Detection Methods:** ADDGRAPH (Zheng et al. 2019), StrGNN (Cai et al. 2021), TADDY (Liu et al. 2021), SAD (Tian et al. 2023), SLADE (Lee, Kim, and Shin 2024), PCA (Shyu et al. 2003), KNN (Angiulli and Pizzuti 2002), GDN (Deng and Hooi 2021), BTAD (Ma, Han, and Zhou 2023), GRN-100 (Tang et al. 2023), DAGMM (Zong et al. 2018), MST-GAT (Ding, Sun, and Zhao 2023), FuSAGNet (Han and Woo 2022), LSTM-VAE (Park, Hoshi, and Kemp 2018), MTAD-GAT (Zhao et al. 2020).

Overall Performance

Edge Level. We compared our methods, GeneralDyG, with nine strong edge-level baseline methods, as shown in Table 1. Our methods consistently outperformed the baselines across both Bitcoin-Alpha and Bitcoin-OTC datasets. The baselines, lacking sufficient structural or temporal information, did not achieve state-of-the-art results. Specifically, GeneralDyG demonstrated an average AUC improvement of approximately 3.2% and 4.5%, respectively, compared to the best-performing baseline on the Bitcoin-Alpha dataset. In terms of Average Precision (AP), GeneralDyG achieved a significant improvement, with up to 24% in the 1% anomaly detection setting, representing a 19.8% increase over the best-performing baseline.

On the Bitcoin-OTC dataset, GeneralDyG also exhibited substantial gains, with an average AUC increase of about 3.6% and an AP improvement of up to 20.2% over the baselines. This demonstrates that our methods are more effective in generalizing and capturing the temporal dynamics necessary for robust anomaly detection in these datasets.
Node Level. We compared our methods, GeneralDyG, with ten strong node-level baseline methods, as shown in Table 2. Our methods generally outperformed the baselines. Specifically, GeneralDyG achieved the highest F1 score on the SWaT dataset with 85.19%, surpassing the second-best method, FuSAGNet, by 1.8%. On the WADI dataset, GeneralDyG reached an F1 score of 60.43%, which is slightly below FuSAGNet's 60.70%, but still demonstrates competitive performance.

Methods	SWaT	WADI
PCA	23.16	9.35
KNN	7.83	7.75
GDN	80.82	56.92
BTAD	81.43	53.77
GRN-100	74.96	48.28
DAGMM	39.37	36.09
MST-GAT	83.55	60.31
FuSAGNet	83.69	**60.70**
LSTM-VAE	73.85	24.82
MTAD-GAT	31.71	16.94
GeneralDyG	**85.19**	60.43

Table 2: Anomaly detection F1 scoring comparison on Node-Level datasets. The best performing method in each experiment is in bold and the second-best method is indicated with underlining.

The baselines, particularly those lacking robust temporal modeling capabilities like PCA and KNN, showed significantly lower F1 scores, with KNN performing the worst on both datasets. Compared to these methods, GeneralDyG shows a notable improvement of approximately 58% on SWaT and 53% on WADI in F1 score. Overall, these results highlight that our methods are better at generalizing and capturing the temporal dynamics necessary for effective anomaly detection in node-level datasets.

Ablation Study

We conducted an ablation study to assess the contribution of each component in the proposed GeneralDyG, as detailed below:

- **w/o ego-graph.** This variant omits the temporal ego-graph sampling process and directly uses the entire graph as input features.
- **w/o TensGNN.** This variant removes the GNN extractor, thereby omitting the extraction of structural information from the events.

Method	Bitcoin-Alpha		WADI
	AUC	AP	F1
GeneralDyG	**96.28**	**26.73**	**60.43**
w/o ego-graph	96.01	19.33	59.45
w/o TensGNN	92.02	22.63	55.13
w/o Transformer	93.71	20.20	58.46

Table 3: The performance of GeneralDyG and its variants on both Node-Level and Edge-Level datasets.

- **w/o Transformer.** This variant excludes the Transformer module, thus omitting the extraction of temporal information from the events.

The ablation study results in Table 3 highlight the significance of each component in GeneralDyG. Removing the temporal ego-graph sampling (w/o ego-graph) results in a decrease of AUC by 0.27% and AP by 27.9% on Bitcoin-Alpha, and a reduction in F1 score by 1.6% on WADI, indicating its critical role in capturing temporal dependencies. Excluding the GNN extractor (w/o TensGNN) leads to a significant decrease in AUC by 4.26%, AP by 15.8% on Bitcoin-Alpha, and F1 score by 8.6% on WADI, underscoring the importance of structural information. Removing the Transformer module (w/o Transformer) results in a decrease of AUC by 2.57%, AP by 24.6% on Bitcoin-Alpha, and F1 score by 3.3% on WADI, emphasizing the need for temporal information processing. These results confirm that each component is crucial for achieving optimal performance.

How to Set Up the Optimal GeneralDyG

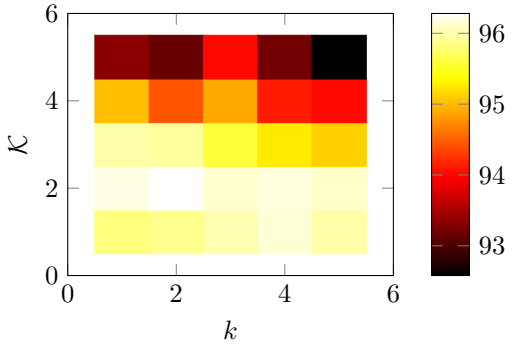

Figure 2: Effect of Parameters k and $\mathcal{K}$ on Bitcoin-Alpha

The heatmap in Figure 2 illustrates the impact of the parameters k and $\mathcal{K}$ on model performance for the Bitcoin-Alpha dataset. It indicates that higher values of $\mathcal{K}$ (the number of layers in the TensGNN) generally lead to decreased performance, suggesting that having too many layers can be detrimental to the model's effectiveness. This could be due to overfitting or increased model complexity without corresponding gains in performance.

On the other hand, the parameter k (which controls the temporal ego-graph sampling) has a less pronounced effect on performance. While increasing k does affect the results, it mainly impacts the training process due to the additional parameters it introduces.

Thus, the optimal setup should aim for a balance: choosing a modest number of layers ($\mathcal{K}$) to avoid overfitting while selecting an appropriate k that provides sufficient temporal information without excessively complicating the model. This balance will help in achieving both efficient training and robust performance.

Generalizable Analysis

Node-Level Method	Bitcoin-Alpha		Edge-Level Method	WADI
	AUC	AP		F1
GeneralDyG	**96.28**	**26.73**	GeneralDyG	**60.43**
GDN	83.84	13.28	TADDY	40.05
MST-GAT	86.66	18.97	SimpleDyG	33.24
FuSAGNet	87.76	20.01	SAD	36.75

Table 4: Generalizable analysis on Node-Level and Edge-Level tasks

In Table 4, GeneralDyG demonstrates its strong generalizability across different types of tasks. Specifically, GeneralDyG consistently outperforms the baseline methods that were evaluated in a mismatched dataset context. For instance, when the edge-level baselines are applied to the node-level dataset (WADI), their performance significantly drops, with metrics such as AUC and F1 score showing substantial declines compared to GeneralDyG. Similarly, node-level baselines tested on the edge-level dataset (Bitcoin-Alpha) exhibit poor performance, further emphasizing their lack of generalizability.

GeneralDyG, on the other hand, maintains high performance across both types of datasets, showcasing its robustness and adaptability. This indicates that GeneralDyG is capable of effectively handling both node-level and edge-level tasks, whereas the baseline methods exhibit considerable performance degradation when faced with different dataset types. These results underline the superior generalizability of GeneralDyG, as it maintains stable and effective performance across diverse scenarios where other methods fail to deliver consistent results.

Conclusion

In this work, we introduced a novel approach for anomaly detection in dynamic graphs called GeneralDyG, which effectively addresses the challenges of data diversity, dynamic feature capture, and computational cost, thereby demonstrating the generalizability of our method. GeneralDyG achieves this by mapping node, edge, and topological structure information into the feature space, incorporating hierarchical tokens, and sampling temporal ego-graphs to efficiently capture dynamic features. GeneralDyG excels across multiple benchmarks, demonstrating its effectiveness and high performance. For future work, we can build on this work to explore the interpretability of anomaly detection in dynamic graphs, providing more robust theoretical support.

Acknowledgments

This research is supported by the Joint NTU-WeBank Research Centre on Fintech, Nanyang Technological University, Singapore. It is also supported by the Joint NTU-UBC Research Centre of Excellence in Active Living for the Elderly (LILY) and the College of Computing and Data Science (CCDS) at NTU Singapore. This work is partially supported by the Wallenberg AI, Autonomous Systems and Software Program (WASP) funded by the Knut and Alice Wallenberg Foundation.

References

Angiulli, F.; and Pizzuti, C. 2002. Fast outlier detection in high dimensional spaces. In *European conference on principles of data mining and knowledge discovery*, 15–27. Springer.

Bai, G.; Ling, C.; and Zhao, L. 2022. Temporal domain generalization with drift-aware dynamic neural networks. *arXiv preprint arXiv:2205.10664*.

Beladev, M.; Katz, G.; Rokach, L.; Singer, U.; and Radinsky, K. 2023. GraphERT–Transformers-based Temporal Dynamic Graph Embedding. In *Proceedings of the 32nd ACM International Conference on Information and Knowledge Management*, 68–77.

Bhatia, S.; Hooi, B.; Yoon, M.; Shin, K.; and Faloutsos, C. 2020. Midas: Microcluster-based detector of anomalies in edge streams. In *Proceedings of the AAAI conference on artificial intelligence*, volume 34, 3242–3249.

Brennan, R. L. 1992. Generalizability theory. *Educational Measurement: Issues and Practice*, 11(4): 27–34.

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

Chen, D.; O'Bray, L.; and Borgwardt, K. 2022. Structure-aware transformer for graph representation learning. In *International Conference on Machine Learning*, 3469–3489. PMLR.

Chen, W.; Tian, L.; Chen, B.; Dai, L.; Duan, Z.; and Zhou, M. 2022. Deep variational graph convolutional recurrent network for multivariate time series anomaly detection. In *International conference on machine learning*, 3621–3633. PMLR.

Cong, Y.; Liao, W.; Ackermann, H.; Rosenhahn, B.; and Yang, M. Y. 2021. Spatial-temporal transformer for dynamic scene graph generation. In *Proceedings of the IEEE/CVF international conference on computer vision*, 16372–16382.

Dai, E.; and Chen, J. 2022. Graph-augmented normalizing flows for anomaly detection of multiple time series. *arXiv preprint arXiv:2202.07857*.

Deng, A.; and Hooi, B. 2021. Graph neural network-based anomaly detection in multivariate time series. In *Proceedings of the AAAI conference on artificial intelligence*, volume 35, 4027–4035.

Ding, C.; Sun, S.; and Zhao, J. 2023. MST-GAT: A multimodal spatial–temporal graph attention network for time series anomaly detection. *Information Fusion*, 89: 527–536.

Dou, Y.; Liu, Z.; Sun, L.; Deng, Y.; Peng, H.; and Yu, P. S. 2020. Enhancing graph neural network-based fraud detectors against camouflaged fraudsters. In *Proceedings of the 29th ACM international conference on information & knowledge management*, 315–324.

Dwivedi, V. P.; and Bresson, X. 2020. A generalization of transformer networks to graphs. *arXiv preprint arXiv:2012.09699*.

Dwivedi, V. P.; Luu, A. T.; Laurent, T.; Bengio, Y.; and Bresson, X. 2021. Graph neural networks with learnable structural and positional representations. *arXiv preprint arXiv:2110.07875*.

Ekle, O. A.; and Eberle, W. 2024. Anomaly Detection in Dynamic Graphs: A Comprehensive Survey. *ACM Transactions on Knowledge Discovery from Data*, 18(8): 1–44.

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

Fang, X.; Liu, D.; Fang, W.; Zhou, P.; Cheng, Y.; Tang, K.; and Zou, K. 2023a. Annotations Are Not All You Need: A Cross-modal Knowledge Transfer Network for Unsupervised Temporal Sentence Grounding. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, 8721–8733.

Fang, X.; Liu, D.; Zhou, P.; Xu, Z.; and Li, R. 2023b. Hierarchical local-global transformer for temporal sentence grounding. *IEEE Transactions on Multimedia*.

Gao, Y.; Li, X.; Peng, H.; Fang, B.; and Philip, S. Y. 2020. Hincti: A cyber threat intelligence modeling and identification system based on heterogeneous information network. *IEEE Transactions on Knowledge and Data Engineering*, 34(2): 708–722.

Gao, Z.-K.; Ma, L.-M.; Li, M.-T.; Zhao, G.; and Li, M. J.-J. 2023. Anomaly Detection in Dynamic Graphs Via Long Short-Term Temporal Attention Network. In *2023 International Conference on Machine Learning and Cybernetics (ICMLC)*, 153–158. IEEE.

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

Han, S.; and Woo, S. S. 2022. Learning sparse latent graph representations for anomaly detection in multivariate time series. In *Proceedings of the 28th ACM SIGKDD Conference on knowledge discovery and data mining*, 2977–2986.

Ho, T. K. K.; Karami, A.; and Armanfard, N. 2024. Graph Anomaly Detection in Time Series: A Survey. arXiv:2302.00058.

Ji, S.; Pan, S.; Cambria, E.; Marttinen, P.; and Philip, S. Y. 2021. A survey on knowledge graphs: Representation, acquisition, and applications. *IEEE transactions on neural networks and learning systems*, 33(2): 494–514.

Lee, J.; Kim, S.; and Shin, K. 2024. SLADE: Detecting Dynamic Anomalies in Edge Streams without Labels via Self-Supervised Learning. *arXiv preprint arXiv:2402.11933*.

Liu, Y.; Pan, S.; Wang, Y. G.; Xiong, F.; Wang, L.; Chen, Q.; and Lee, V. C. 2021. Anomaly detection in dynamic graphs via transformer. *IEEE Transactions on Knowledge and Data Engineering*, 35(12): 12081–12094.

Ma, M.; Han, L.; and Zhou, C. 2023. BTAD: A binary transformer deep neural network model for anomaly detection in multivariate time series data. *Advanced Engineering Informatics*, 56: 101949.

Ma, X.; Wu, J.; Xue, S.; Yang, J.; Zhou, C.; Sheng, Q. Z.; Xiong, H.; and Akoglu, L. 2021. A comprehensive survey on graph anomaly detection with deep learning. *IEEE Transactions on Knowledge and Data Engineering*, 35(12): 12012–12038.

Mialon, G.; Chen, D.; Selosse, M.; and Mairal, J. 2021. Graphit: Encoding graph structure in transformers. *arXiv preprint arXiv:2106.05667*.

Park, D.; Hoshi, Y.; and Kemp, C. C. 2018. A multimodal anomaly detector for robot-assisted feeding using an lstm-based variational autoencoder. *IEEE Robotics and Automation Letters*, 3(3): 1544–1551.

Perozzi, B.; Al-Rfou, R.; and Skiena, S. 2014. Deepwalk: Online learning of social representations. In *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*, 701–710.

Pourhabibi, T.; Ong, K.-L.; Kam, B. H.; and Boo, Y. L. 2020. Fraud detection: A systematic literature review of graph-based anomaly detection approaches. *Decision Support Systems*, 133: 113303.

Qu, X.; Li, Z.; Wang, J.; Zhang, Z.; Zou, P.; Jiang, J.; Huang, J.; Xiao, R.; Zhang, J.; and Gao, J. 2020. Category-aware graph neural networks for improving e-commerce review helpfulness prediction. In *Proceedings of the 29th ACM International Conference on Information & Knowledge Management*, 2693–2700.

Rossi, E.; Chamberlain, B.; Frasca, F.; Eynard, D.; Monti, F.; and Bronstein, M. 2020. Temporal graph networks for deep learning on dynamic graphs. *arXiv preprint arXiv:2006.10637*.

Shyu, M.-L.; Chen, S.-C.; Sarinnapakorn, K.; and Chang, L. 2003. A novel anomaly detection scheme based on principal component classifier. In *Proceedings of the IEEE foundations and new directions of data mining workshop*, 172–179. IEEE Press Piscataway, NJ, USA.

Skarding, J.; Gabrys, B.; and Musial, K. 2021. Foundations and modeling of dynamic networks using dynamic graph neural networks: A survey. *iEEE Access*, 9: 79143–79168.

Su, Y.; Zhao, Y.; Niu, C.; Liu, R.; Sun, W.; and Pei, D. 2019. Robust anomaly detection for multivariate time series through stochastic recurrent neural network. In *Proceedings of the 25th ACM SIGKDD international conference on knowledge discovery & data mining*, 2828–2837.

Sun, M.; Cui, W.; Yu, S.; Han, H.; Hu, B.; and Li, Y. 2022. A dual-branch dynamic graph convolution based adaptive transformer feature fusion network for EEG emotion recognition. *IEEE Transactions on Affective Computing*, 13(4): 2218–2228.

Tang, C.; Xu, L.; Yang, B.; Tang, Y.; and Zhao, D. 2023. GRU-based interpretable multivariate time series anomaly detection in industrial control system. *Computers & Security*, 127: 103094.

Tian, S.; Dong, J.; Li, J.; Zhao, W.; Xu, X.; Song, B.; Meng, C.; Zhang, T.; Chen, L.; et al. 2023. Sad: Semi-supervised anomaly detection on dynamic graphs. *arXiv preprint arXiv:2305.13573*.

Vaswani, A.; Shazeer, N.; Parmar, N.; Uszkoreit, J.; Jones, L.; Gomez, A. N.; Kaiser, Ł.; and Polosukhin, I. 2017. Attention is all you need. *Advances in neural information processing systems*, 30.

Wang, L.; Chang, X.; Li, S.; Chu, Y.; Li, H.; Zhang, W.; He, X.; Song, L.; Zhou, J.; and Yang, H. 2021. Tcl: Transformer-based dynamic graph modelling via contrastive learning. *arXiv preprint arXiv:2105.07944*.

Wang, L.; Yu, Z.; Xiong, F.; Yang, D.; Pan, S.; and Yan, Z. 2019. Influence spread in geo-social networks: a multi-objective optimization perspective. *IEEE Transactions on Cybernetics*, 51(5): 2663–2675.

Wu, Y.; Fang, Y.; and Liao, L. 2024. On the Feasibility of Simple Transformer for Dynamic Graph Modeling. In *Proceedings of the ACM on Web Conference 2024*, 870–880.

Xu, D.; Ruan, C.; Korpeoglu, E.; Kumar, S.; and Achan, K. 2020. Inductive representation learning on temporal graphs. *arXiv preprint arXiv:2002.07962*.

Zhang, W.; Zhang, C.; and Tsung, F. 2022. GRELEN: Multivariate Time Series Anomaly Detection from the Perspective of Graph Relational Learning. In *IJCAI*, 2390–2397.

Zhao, H.; Wang, Y.; Duan, J.; Huang, C.; Cao, D.; Tong, Y.; Xu, B.; Bai, J.; Tong, J.; and Zhang, Q. 2020. Multivariate time-series anomaly detection via graph attention network. In *2020 IEEE international conference on data mining (ICDM)*, 841–850. IEEE.

Zheng, L.; Li, Z.; Li, J.; Li, Z.; and Gao, J. 2019. AddGraph: Anomaly Detection in Dynamic Graph Using Attention-based Temporal GCN. In *IJCAI*, volume 3, 7.

Zong, B.; Song, Q.; Min, M. R.; Cheng, W.; Lumezanu, C.; Cho, D.; and Chen, H. 2018. Deep autoencoding gaussian mixture model for unsupervised anomaly detection. In *International conference on learning representations*.