AD-TIN: Edge Anomaly Detection for Temporal Interaction Networks using Multi-representation Attention

Aming Wu, Young-Woo Kwon
School of Computer Science and Engineering
Kyungpook National University, Daegu, South Korea
wuaming@knu.ac.kr, ywkwon@knu.ac.kr

***Abstract*—Anomaly detection in temporal interaction networks (TINs) has become critical in network security, digital finance, and social networks. While recent studies based on Graph Neural Networks (GNNs) have yielded promising results, the existing methods are still limited by insufficient labels and noisy data, often ignoring the information filtering for unrelated user interactions. Therefore, this paper proposes a dynamic edge anomaly detection framework, AD-TIN, to address these challenges based on a multi-representation attention mechanism. It encodes graph structural information using a network information propagation module with neighbor sampling and graph diffusion. Furthermore, the network update module combines past node states with current structural features to capture the temporal information in potential user relationships, effectively mitigating the impact of noisy data. Extensive experiments on three real-world datasets demonstrate the robustness and efficacy of AD-TIN in addressing noise and unrelated interactions for edge anomaly detection.**

***Index Terms*—Anomaly detection, Temporal interaction network, Attention mechanism, Graph diffusion**

I. Introduction

Graph neural networks (GNNs) have presented an exceptional performance in various network-based tasks [1], rendering their widespread adoption in addressing issues like credit card fraud, malicious reviews of shopping websites, and network intrusions. However, a substantial portion of the existing literature concerning graph-based anomaly detection primarily revolves around static graphs [2]. While these static representations offer value, they often prove inadequate when confronting the dynamic and ever-changing landscapes in real-world scenarios, including those prevalent in social media and e-commerce systems. In contrast, dynamic graphs, harnessed for anomaly detection, offer the capability to simultaneously extract spatiotemporal information and structural features. Notably, anomaly detection based on dynamic graphs facilitates the simultaneous extraction of spatiotemporal information and structural features. Most studies depend on structural differences in continuous graph snapshots or local node changes to identify anomalies [3]. Nevertheless, these approaches often overlook the identification of suspicious interactions between nodes in temporal interaction networks (TINs), which presents a significant challenge in this context.

Recently, two primary approaches have been employed to address interaction anomalies within temporal networks: 1) Sketch-based methods: Randhaus et al. [4] employ a sketch-based approach to identify anomaly edges by leveraging global and local structural attributes. 2) Graph-based representation learning techniques: AddGraph [5] introduces the incorporation of a Gated Recurrent Unit (GRU) module into Graph Convolutional Networks (GCN) to capture dependencies between long-term and short-term patterns within dynamic graphs. Another extension of the GCN is DynAD [6], which employs control loop units with pooling layers for adaptive parameter learning, accommodating nodes with minor changes. In summary, the first category of method encounters limitations when dealing with complex temporal graph datasets and may grapple with challenges associated with biased sampling, which can subsequently impair sketch modeling. Conversely, the second category, exemplified by the latest approaches AddGraph and DynAD, combines GCN and recurrent neural networks to capture both the structural characteristics of extensive networks and sequence-based features.

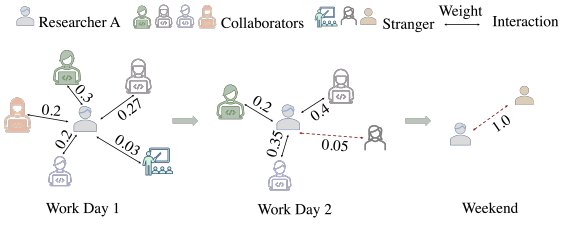


Fig. 1. An example of deformed edges results in negative smoothness. When researcher A's collaborators regard it as a reference, the normalized similarity weights between A and each neighbor present significant differences at different times.

Despite the improved performance achieved by the methods mentioned above, several pressing challenges warrant further scrutiny and resolution. ***Challenge1: Lack of reference neighbors leading to negative smooth effects.*** Common GNNs

ASONAM '23, November 6-9, 2023, Kusadasi, Turkey

ACM ISBN 979-8-4007-0409-3/23/11…$15.00
https://doi.org/10.1145/3625007.3627502

have the potential to inadvertently smooth the low-dimensional representations of unrelated neighboring nodes, particularly when there is a lack of other neighbor interactions for reference or user behavior undergoes temporal evolution, leading to distinct behavioral patterns in different time intervals. To illustrate this issue, Figure 1 illustrates a temporal network centered around researcher A in a social scenario, where the low-dimensional representations of unrelated nodes exhibit a high correlation, thereby significantly influencing the accuracy of edge anomaly detection in TINs. ***Challenge2: Modeling errors arising from noisy data.*** In practical scenarios, label data for TINs are often limited. As a result, many methods assume that the initially existing edges have interactions and use them as training sets to model parameters for detection tasks. Nevertheless, the dataset may contain noisy data due to human factors or measurement failures, which make the mode learn the features of the noisy data during the training phase, resulting in modeling errors and reduced detection performance.

To tackle the challenges above, we propose a multi-representation attention-based anomaly detection approach for interactive temporal networks. The main contributions of our work are presented as follows:

- We propose a graph attention network (GAT) with a multi-representation attention framework, AD-TIN, for detecting edge anomalies in TINs, the first exploration of negative smooth from unrelated interaction.
- In the context of information propagation from a neighboring node to a central node, we introduce a novel multi-representation attention strategy, which considers both the relative attention weight of each adjacent node compared to the sum of all neighboring node attention weights, as well as the correlation between the neighboring node and the central node.
- In dynamic information update, to capture the temporal information of nodes in dynamic TINs, we utilize a GRU for modeling temporal changes.
- A series of experimental studies on three datasets have verified the effectiveness and robustness of AD-TIN.

The rest of this paper is organized as follows. Section II describes the related work. Section III presents the preliminaries. Section IV focuses on the details of the model's structure and the optimization of algorithms. In Section V, we explain the details of the experience and dataset. Section VI verifies our work's effectiveness through various evaluation approaches. Finally, the whole work is concluded in Section VII.

II. Related Work

The emergence of GNNs has driven the development of detection techniques for dynamic graphs leveraging GNNs for network representation learning to analyze anomalies. This section reviews prevalent graph detection tasks at the node, subgraph, and edge levels.

A. Node-level and Graph-level Anomaly

In the context of anomaly detection, anomaly nodes are typically recognized as individual nodes that exhibit notable deviations from features of other nodes in the network [7], [8]. The GraphSAGE [7] employs an inductive learning strategy to replace the convolution operation in the network, effectively generating node embeddings. While DynAnom [8] locates anomalies for each node on large dynamic weighted graphs by quantifying the changes of individual nodes over time. On the other hand, graph-level anomaly involves detecting sudden changes in the graph structure as it evolves between consecutive graph snapshots [9], [10]. Shao et al. [9] introduce the dynamic evolution anomaly subgraph scan (dGraphScan) method to address anomaly subgraph issues as the network evolves. Meanwhile, SUGAR [10] proposes a hierarchical subgraph level selection approach combined with embedding-based methods to improve discriminative subgraph representations. However, these methods only rely on local attribute changes of individual nodes or subgraph structure variations, thus failing to capture abrupt fluctuations in edge weights between sparsely connected node pairs. Consequently, they are hard to identify edge anomalies.

B. Edge-level Anomaly Detection

Edge anomalies aim to identify sudden changes in edge weights between two nodes within a continuous graph snapshot [3]. Specifically, NetWalk [11] employs random walking and real-time update sampling to obtain context node sequences, generates low-dimensional node representations by minimizing the distance between node pairs, and applies clustering techniques to detect anomaly edges. Nonetheless, it lacks the consideration of temporal behavior in dynamic graphs. Therefore, some studies [12] further extend GNNs to recurrent neural networks, enabling the extraction of both temporal and structural features. Ouyang et al. [13] propose an end-to-end neural network architecture that models the probability distribution of edges based on local structures to identify anomaly edges. Most recently, Chang et al. [14] measure the interaction frequency between nodes to determine the reliability of the interaction. The above approaches obtain promising outcomes for edge anomaly detection in dynamic networks. However, they ignore the effect of initial refer neighbors and noisy data during modeling. As the introduction demonstrates, this oversight has constrained the GNN's effectiveness in TINs. Instead, our approach is inspired by robust representation learning capabilities of GATs [15] to identify weaknesses in existing designs and rejuvenate anomaly detection in large-scale network interactions.

III. Preliminaries

Definition 1: TINs. A TIN is a sequence of interaction networks arranged in chronological order, denoted as $G = g(t)_{\text{t=1}}^{T}$, where $g(t) = (V_t, E_t)$ represents the interaction network at timestamp t, V_t and E_t denote the sets of nodes and edges in the t-th timestamp, respectively. We use $V = \{V_1, V_2, ..., V_T\}$ as the node sets and $E = \{E_1, E_2, ..., E_T\}$ as the the edge

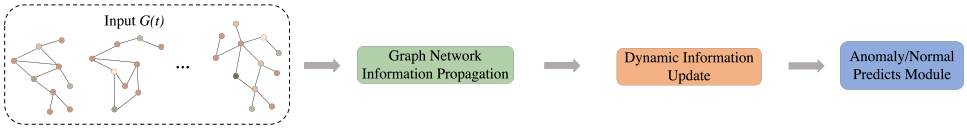


Fig. 2. The AD-TIN framework-based anomaly edge detection.

sets across time T, $|V| = n$ and $|E| = m$. $A_t \in \mathbf{R}^{m\times n}$ is the adjacency matrix representing at time t. If there is an edge connecting node v_i and node v_j at timestamp t, then $E_{t,ij}$ = 1, otherwise $E_{t,ij} = 0$.

Definition 2: Edge Anomaly Detection. Given a TIN $G = g(t)_{\text{t=1}}^{T}$, the edge anomaly detection based on TINs is to compute the anomaly probability $P(e_{i,j})_T$ for all target edges $e_{i,j}$ (the interaction between nodes v_i and v_j) that exist in the time period T. Still, anomaly probability is a value $f(e_{i,j})_t \in [0, 1]$ that is used to describe the anomaly possibility for edge $e_{i,j}$ during timestamp t. The anomaly probability $f(e_{i,j})_t \in [0, 1]$ is calculated based on the low-dimensional vector representations of the two users, and the specific method for calculating the anomaly probability is provided in Eq. 17.

IV. The Design of AD-TIN

In this section, we present a comprehensive overview of AD-TIN with two main modules: (i) Network information propagation module. It is responsible for capturing the structural features of the current network at each timestamp. (ii) Network update module. The goal is to capture the temporal dependency in user relationships. Figure 2 presents the overview framework of the AD-TIN.

A. GNN Information Propagation

In recent research, GNNs have been exploited to obtain low-dimensional vector representations of users in anomaly detection. Inspired by GAT [16], our approach focuses on a multi-layer information propagation structure. Specifically, in the timestamp t, the adjacency matrix M_t, global structure encoding Q_g, and the output ω_{t}^{l-1} of $l-1$ layer are taken as the input for the information propagation of the l-th layer. Figure 3(a) illustrates the information propagation process in a single-layer network.

Multi-representation Attention: In the GAT model, attention weights representing node interactions are typically computed by mapping connected nodes' vector representations to a single value and normalizing them using softmax. However, this approach can smooth unrelated nodes to a strong correlation, leading to misleading information propagation, especially when detecting target edges without neighboring node reference. To mitigate this issue, we propose a multi-representation attention mechanism that considers two factors affecting information dissemination: (i) The proportion of the attention weight of a neighboring node to the sum of all adjacent node attention weights; (ii) The correlation between each neighbor node and the central node. Additionally, we introduce two attention representation methods and map nodes to a higher-level feature space. Attention coefficients for the target node v_j to the central node v_i are as follows:

$$K_{(t,i)}^{l} = \omega_{(t,i)}^{l-1}\lambda_{k}^{l}, \tag{1}$$

$$S_{(t,j)}^{l} = \omega_{(t,j)}^{l-1}\lambda_{s}^{l}, \tag{2}$$

where $\omega_{(t,i)}^{l-1}$and $\omega_{(t,j)}^{l-1}$ represent the low-dimensional vector representations of nodes v_i and v_j, respectively; $\lambda_k^l, \lambda_s^l \in R^{d_v\times d_k}$ are two linear transformation matrices with different trainable parameters, and l is the parameter of network layers.

$$\tilde{e}_{(ij)_t}^{l} = sigmoid\left(K_{(t,i)}^{l}S_{(t,j)}^{l}\right), \tag{3}$$

$$\check{e}_{(ij)_t}^{l} = \frac{2\times e^{\left(K_{(t,i)}^{l}S_{(t,j)}^{l}\right)}}{\sum_I e^{\left(K_{(t,I)}^{l}S_{(t,j)}^{l}\right)} + \sum_J e^{\left(K_{(t,i)}^{l}S_{(t,J)}^{l}\right)}}. \tag{4}$$

(iii) Obtain the final attention coefficient by combining the two processed attention weights, which emphasizes the importance of nodes with related relationships while weakening the impact of nodes entirely unrelated to the central node. The standardized attention weight is calculated as follows:

$$\tilde{a}_{(ij)_t}^{l} = \frac{\tilde{e}_{(ij)_t}^{l} + \check{e}_{(ij)_t}^{l}}{2}, \tag{5}$$

where $\tilde{e}_{(ij)_t}^{l}$ represents the attention coefficient that considers the attention weight of the neighboring node itself and the center node, $\check{e}_{(ij)_t}^{l}$ represents the attention coefficient considering the proportion of the attention weight of the neighboring node to the sum of all adjacent node attention weights.

By incorporating these modifications into the attention mechanism, our proposed approach aims to improve the accuracy and robustness of information dissemination in the GAT model by accounting for the correlations and relevance between nodes and mitigating the impact of unrelated nodes during target edge detection.

Neighbor Sampling: As highlighted in previous studies [13], [17], unlabeled data often contain noisy edges, and anomalies tend to occur in local substructures, necessitating the need to expand the investigation scope. To mitigate noise, we introduce a random walk-based neighbor sampling algorithm for selective feature aggregation from neighbors during information propagation. Moreover, we integrate graph diffusion to obtain global patterns. Here, we describe the calculation of sampling probabilities for neighboring nodes and their utilization in addressing the issue of negative smoothing.

Random walk: To sample neighboring nodes, we employ a biased random walk [18] starting from the center node to capture the local structural information around the center node. This method combines the advantages of depth-first and breadth-first walking, where the transition probability from one node to another is influenced by both the edge weight and the transition state. Given an interactive network $G(t)$, a node v_i transitions to a neighboring node v_j with probability $\frac{\pi_{ij}}{B}$, where π_{ij} represents non-standard transfer probability and B is a normalization constant. The transfer probability is

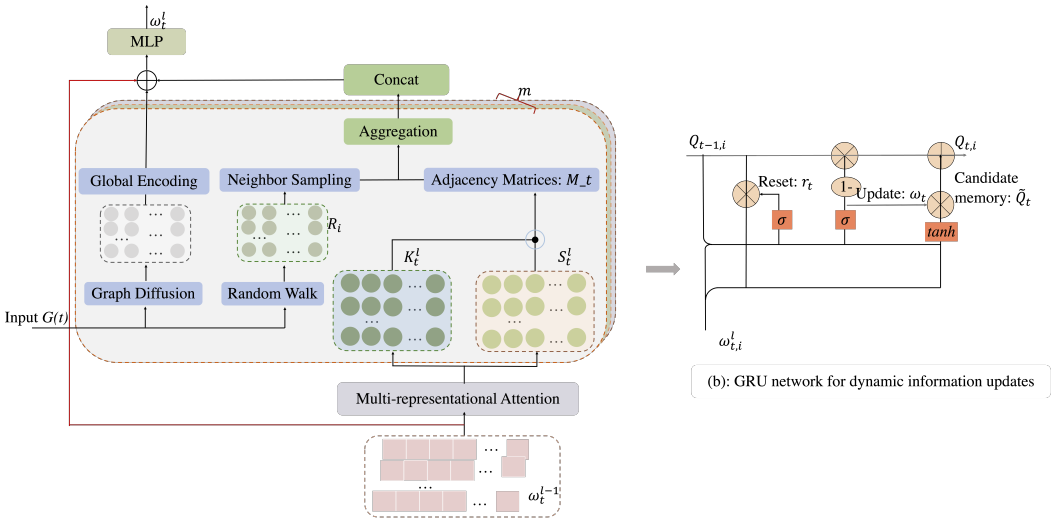


Fig. 3. Overall process of the proposed AD-TIN.

determined by edge weight ω_{ij} and transition state $\alpha_{pq}(r,j)$, denoted as $\pi_{ij} = \alpha_{pq}(r,j) \times \omega_{rj}$.

$$\alpha_{pq}(r,j) = \begin{cases} \frac{1}{p}, & if\ e_{rj} = \emptyset \\ 1, & if\ e_{rj} \text{ is first-order neighbor} \\ \frac{1}{q}, & if\ e_{rj} \text{ is second-order neighbor} \end{cases} \tag{6}$$

Where the node v_i and the node v_j before node v_r represent the node that was previously walked, the random walk sequences are generated using the aforementioned method. For node v_i, multiple random walk sequences starting from itself will be obtained, and the node set in all random walk sequences is represented as R_i. With nodes v_i, R_i, and the neighbor node set N_i, the sampling probability can be computed using the occurrence rate matrix $R \in R^{n \times m}$. The specific calculation method is as follows:

$$R_{ij} = \begin{cases} \frac{\xi}{\eta}, & if\ v_j \in N_i, v_j \in R_i \\ o, & others \end{cases} \tag{7}$$

Where ξ means the frequency of v_i appearing in the set R, $\eta = \sum_{j^* \in N_i} \xi^*_{ij}$ is used for standardization.

Based on the occurrence matrix R, for node v_i, the sample from the uniform distribution of [0,1] to obtain u. If $u < R_{ij}$, add node v_j as the sampling neighbor of node v_i. The sampled neighbors of v_i are represented as the set N^*_i.

Graph-diffusion: The h-hop neighbor sampling method based on a single snapshot may ignore neighbor nodes' importance. Meanwhile, the uneven distribution of node degrees in real-world datasets can lead to performance degradation and low efficiency. Therefore, we present a graph diffusion method that captures a fixed-size node set for each target edge, generating diffusion matrices. Taking target edge $e_{ij} = (v_i, v_j)$ as an example, we can obtain the connection vector $Y^t_{e_{ij}}$ (corresponding to the matrices) by summing the connection vector of the target edge's neighbor nodes in timestamp t:

$$Y^t_{e_{ij}} = Y^t_{v_i} + Y^t_{v_j}. \tag{8}$$

Subsequently, we sort the connection vector and build the neighbor node set $Y^t_{e_{ij}}$. For dynamic graphs, we merge the connection vector of multiple timestamps:

$$Y_e = \bigcup_{t=1}^{T} Y^t_{e_{ij}}. \tag{9}$$

The graph diffusion can generate a global feature encoding for each target edge. We arrange the sampled nodes based on their diffusion values within the sampled set of each edge, thereby obtaining the edge's global structure encoding F_t.

Aggregation: By calculating the normalized attention score between the target node and the sampled neighbor and performing linear aggregation of the sampled neighbor features of node v_i. The sampling neighbor node feature $\delta^l_{t,i} \in R^{d_k}$ of node v_i is calculated by the following method:

$$\delta^l_{t,i} = \sum_{v_j \in N^i_t} \tilde{a}^l_{(ij)_t} \cdot \omega^{l-1}_{(t,j)} \cdot \lambda^l_v, \tag{10}$$

where $\lambda^l_v \in R^{d_v \times d_k}$ represents a trainable parameter linear transformation matrix. At the same time, in order to extract information from different subspaces, the attention mechanism is extended to the multi-head attention mechanism. The multi-head attention mechanism model executes h independent attention mechanisms in parallel. After all the attention mechanisms are executed, the aggregation features are vector-connected. Finally, the connected vector dimensions are mapped to d_v, which can be expressed as:

$$\delta^l_t = concat\left(\left(\delta^l_t\right)_1, \left(\delta^l_t\right)_2, ..., \left(\delta^l_t\right)_h\right) \cdot \lambda^l_h, \tag{11}$$

where $concat(\cdot)$ denotes a concatenation operation, which performs vector splicing for all matrices $\left(\delta^l_t\right) \in R^{n \times d_k}$; $\lambda^l_h \in R^{(d_k * h) \times d_v}$ represents the trainable parameter linear transformation matrix; δ^l_t represents the obtained diverse sampled neighbor nodes features.

Finally, the low-dimension vector representation of the node is fused with the features of the multi-sampling neighbor

node. Using its powerful and flexible nonlinear modeling capability, the multi-layer perceptron model is used to model the multi-dimensional features between the target node and the multi-sampling neighbor node. Given the low-dimensional vector representation $\omega_{(t,i)}^{l-1}$ of node v_i and its diverse sampling neighbor node features $\delta_{(t,i)}^{l}$, the structural features $\omega_{(t,i)}^{l}$ obtained from information propagation in the timestamp t, l-th can be carried out as follows calculation:

$$\omega_{t,i}^{l} = MLP\left(\omega_{(t,i)}^{l-1} + \delta_{t,i}^{l} + F_t^l\right), \tag{12}$$

where MLP represents a multi-layer perceptron with hidden layers and an output layer. The structural features of output $\omega_{(t,i)}^{l}$ in the time period t of the l-th network layer.

B. Network Update

In this section, we discuss modeling dynamic TINs and calculating anomaly probability. We employ GRU, a variant of LSTM, to update node structural features (ω_t^l) at each period (t) and capture temporal information in the network (As shown in Figure 3(b)). The network update module obtains ω_t^l from the information propagation module. For each node (v_i), GRU updates the current hidden information ($Q_{t,i}$) based on the node structure features ($\omega_{t,i}^l$) and the previous hidden state (Q_{t-1}), expressed as:

$$u_t^i = \sigma\left(W_u\omega_{t,i}^l + U_u Q_{t-1,i} + b_u\right), \tag{13}$$

$$r_t^i = \sigma\left(W_r\omega_{t,i}^l + U_r Q_{t-1,i} + b_r\right), \tag{14}$$

$$\tilde{Q}_t^i = \tanh\left(W_n\omega_{t,i}^l + U_n\left(r_t^i \odot Q_{t-1,i}\right)\right), \tag{15}$$

$$Q_t^i = u_t^i \odot Q_{t-1,i} + (1 - u_{t,i}) \odot \tilde{Q}_t^i, \tag{16}$$

where $u_{t,i}$ indicates the update gate of the gating loop unit, filters invalid information, and adds new information; $r_{t,i}$ represents the reset gate of the control loop unit, determining how to fuse new information with previous information; $\sigma(\cdot)$ is the sigmoid function; W_u, U_u, W_r, U_r, W_n and U_n represent different trainable parameter matrices. By applying this modeling and iterating over time, we obtain low-dimensional vector representations Q_T for all nodes at the latest time.

Through these low-dimensional vector representations, we conduct anomaly detection by considering the interaction between node pairs. To minimize hyperparameters and interference, a fully connected layer is used as a predictor for computing the anomaly probability. For each target edge, given the low-dimensional vector representations $Q_{T,i}$ and $Q_{T,j}$, the anomaly probability of the target edge is defined as:

$$f(e_{i,j})^T = \sigma(FC\left(\beta \odot Q_{T,i} + \gamma \odot Q_{T,j}\right)), \tag{17}$$

where $\beta, \gamma \in R^{d_v}$ represent different trainable parameter vectors, respectively, which are used to help the model understand and model the interaction between node pairs on the basis of low dimensional vector representation.

TABLE I
COMPARISON OF THE NETWORK OF DIFFERENT MODELS.

Dataset	Node #	Edge #	Max.Degree	Avg.Degree
UCI Messages	1899	13838	255	14.57
Digg	30360	85155	283	5.61
Email	2029	39264	5813	38.70

C. Objective Optimization

An effective way to enhance the model performance is to utilize anomaly probability as an optimization factor. However, all existing edges are assumed normal in scenarios without edge labels. Thus, negative sampling is necessary before defining the loss function. Bernoulli distribution [19] is employed to generate negative samples by replacing the end node for each existing edge. The probabilities for selecting nodes v_i and v_j are $\frac{D_{t,i}}{D_{t,i}+D_{t,j}}$ and $\frac{D_{t,j}}{D_{t,i}+D_{t,j}}$, respectively. In addition, the AdaGrad optimization algorithm is employed to minimize the loss function and fine-tune the model. The loss function is formulated as follows:

$$L_t^m = -\sum_{i=1}^{m} \log\left(1 - f(e_{i,j})\right) + \log\left(f(e_{i^*,j^*})\right), \tag{18}$$

where $m \in E_t$ represents the number of edges in timestamp t, and e_{i^*,j^*} represents the negative edge sampled by the negative sampling strategy.

V. EXPERIMENTAL SETTING

In this section, we explain the basic experimental setup and present the analysis of the extensive experiment results for AD-TIN using three real-world datasets.

A. Experimental Datasets

To validate the proposed AD-TIN, we use three datasets as shown in Table I, which include:

• The UCI Message dataset [20] is from the network of the online community of students at the University of California, Irvine. Each user in the online community corresponds to a node in the network, and an edge appears with each message interaction between two users corresponds to an edge.

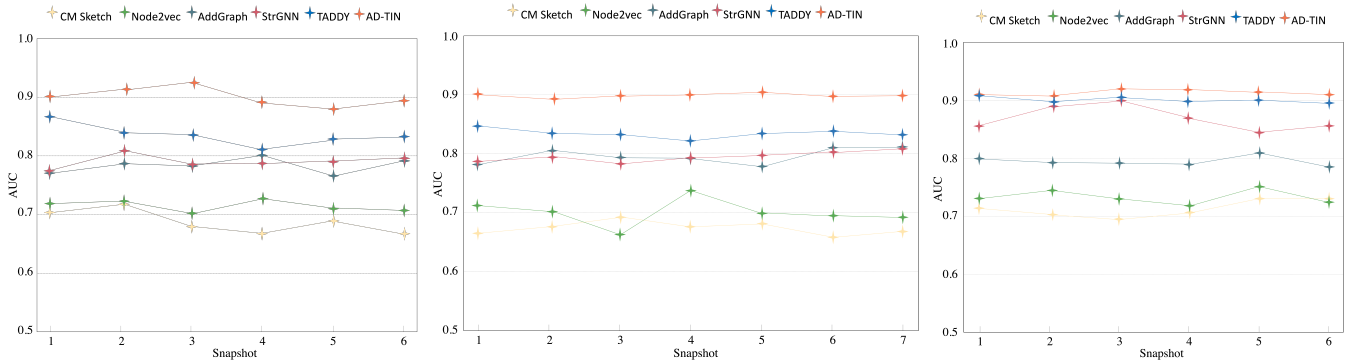
• The Digg dataset [21] is collected from the news aggregation website digg.com. Each user on a website is saved as a node, and replies between two users are considered edges.

• The Email dataset [22] contains a collection of emails obtained from the Democratic National Committee. In this dataset, each node represents an individual, and each edge represents an email communication between two individuals.

The edges in datasets contain timestamps. Both datasets are divided into a training set (50%) and a testing set (50%). Since they are not labeled, we generate and inject anomaly edges (exceptions) in the test set. Before evaluating the AD-TIN performance on interactive temporal networks, we conduct experiments on static interactive networks. In contrast, in the dynamic network, anomaly edges with 1%, 5%, and 10% are injected into three test sets to compare different models' performance. Table I summarizes the statistics of the dataset.

TABLE II
COMPARISON OF DIFFERENT MODELS ON STATIC NETWORKS UNDER DIFFERENT ANOMALY PROPORTIONS.

Datasets		Methods					
		CM Sketch	Node2Vec	AddGraph	StrGNN	TADDY	AD-TIN
UCI Messages	1%	0.7268	0.7375	0.8102	0.8011	0.8667	**0.9205**
	5%	0.7079	0.7512	0.8056	0.8152	0.8289	**0.9297**
	10%	0.6921	0.7013	0.7543	0.7883	0.8412	**0.9173**
	Mean (SD)	0.0142	0.0210	0.0253	0.0130	0.0157	**0.0052**
Digg	1%	0.6819	0.7331	0.8289	0.8005	0.8501	**0.8902**
	5%	0.6631	0.7124	0.8201	0.8121	0.8403	**0.8872**
	10%	0.6473	0.6648	0.8079	0.7827	0.8501	**0.8848**
	Mean (SD)	0.0141	0.0286	0.0086	0.0121	0.0046	**0.0021**
Email-DNC	1%	0.7219	0.7411	0.8303	0.8645	0.9037	**0.9335**
	5%	0.7006	0.7258	0.8120	0.9034	**0.9304**	0.9277
	10%	0.6917	0.7003	0.8057	0.8903	0.9134	**0.9228**
	Mean (SD)	0.0127	0.0168	0.0104	0.0163	0.0121	**0.0044**



(a) UCI Message (b) Digg (c) Email-DNC

Fig. 4. Overall performance on the dynamic network under different snapshots.

B. Benchmark Methods

To assess the effectiveness of AD-TIN, this study compares it with four benchmark methods for anomaly detection. The benchmark methods are divided into two groups: (i) Traditional anomaly detection methods - CM-Sketch [23], and (ii) Graph representation learning - Node2Vec [18], AddGraph [5], StrGNN [24], and TADDY [17].

C. Parameter Settings

The experimental parameter settings for all data sets are as follows: the training times set to 100, projection dimension d_v set to 3, control parameters p and q of the random walk set to 1 and 2, respectively. The run length and run times are set to 4 and 25, respectively. For the experiment on the UCI Message dataset, the learning rate for training, the number of information propagation layers, and the dimension of the hidden state are 0.001, 3, and 128, respectively. For the Digg and Email datasets, the learning rate for training, the number of information propagation layers, and the dimension of the hidden state are 0.0005, 3, and 256, respectively.

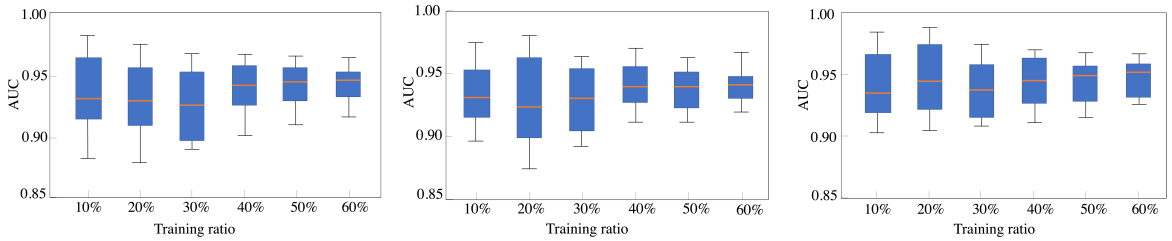
VI. RESULT ANALYSIS

A. Validation

In the effectiveness experiment, we compare the performance of the proposed AD-TIN with benchmark methods. The experiment consists of two stages: (i) Detecting anomalies in static networks to validate the effective extraction of structural features; (ii) Evaluating the model's performance in dynamic TINs by incorporating temporal features.

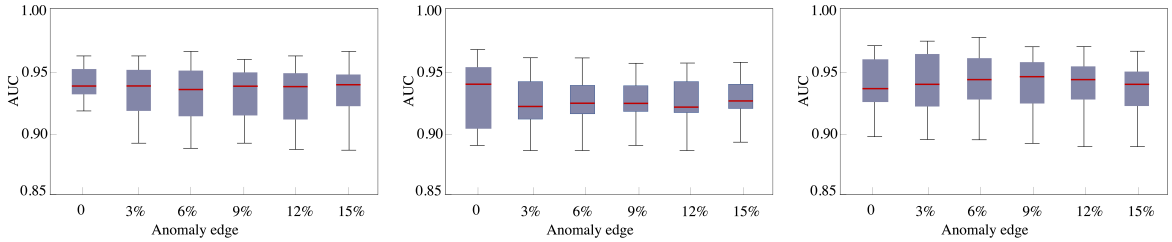
Effectiveness on static TINs. For the static network, we inject 1%, 5%, and 10% anomaly edges into the three test sets using the aforementioned anomaly injection method to compare the performance of AD-TIN and benchmark methods. Specific experimental results are presented in Table II, with the superior outcomes highlighted in bold. The findings reveal that AD-TIN consistently outperforms the benchmark methods across all datasets. Specifically, on the UCI Message dataset, AD-TIN achieves AD-TIN achieves approximately 0.92 AUC score, and its average AUC is about 7.0% higher than the most effective benchmark method (TADDY). Similarly, on the Digg dataset, AD-TIN achieves an average AUC score of around 0.88, outperforming TADDY by an average of 4.0%. On the Email-DNC, AD-TIN attains an average AUC score exceeding 0.93, surpassing TADDY by 1.5%. These results indicate that AD-TIN can effectively extract structural features from the dataset, enhancing its capability to identify anomaly edges.

Furthermore, we examine the standard deviations of AUC for three anomaly ratios, as presented in Table II. Compared with benchmark methods, our method exhibits a significantly lower standard deviation (SD), which suggests that AD-TIN achieves more stable performance across different proportions of anomaly data. A reasonable assumption is that AD-TIN considers the impact of anomaly edges and noise on the low-dimensional vector representation of nodes, making it particularly suitable for anomaly detection.



(a) UCI Message (b) Digg (c) Email-DNC

Fig. 5. Models' performance with different training ratios.



(a) UCI Message (b) Digg (c) Email-DNC

Fig. 6. Analysis of resilience to noise for proposed AD-TIN.

Effectiveness on dynamic TINs. Structural and temporal features are considered in the experiments on dynamic networks. Therefore, all datasets are divided into multiple snapshots. In this experiment, a 5% injection of anomaly edges is applied to all test sets, with an equitable division of the three training and testing sets into multiple snapshots. Figure 4 visualizes the test outcomes for the three datasets. Compared with other benchmarks, the proposed method outperforms other methods on three datasets. Specifically, on the UCI Message dataset, AD-TIN achieves an AUC score of approximately 0.940, marking a 10% improvement over the best-performing benchmark, TADDY. On the Digg dataset, AD-TIN's average AUC score (0.953) surpasses TADDY's by roughly 7.0%. Additionally, AD-TIN's average AUC score on the Email-DNC dataset is around 2.0% higher than other benchmarks. Therefore, the finding suggests that dynamic modeling user interactions lead to more accurate anomaly edge detection results than static networks. AD-TIN significantly improves average accuracy on TINs compared to other benchmark methods. Overall, it is meaningful to consider negative smoothing and noise issues in edge anomaly detection based on TINs. Meanwhile, these observations highlight the significance of employing dynamic modeling approaches for achieving enhanced accuracy in anomaly detection.

B. Parameter Sensitivity Analysis

This experiment aims to assess the influence of hyperparameters on the performance of AD-TIN. The model primarily relies on two hyperparameters: the number of information propagation layers l and the hidden state d_v dimension. We investigate the sensitivity of these two parameters by injecting 10% anomalies into each dataset. Table III displays an example of the AD-TIN parameters sensitivity experimental results on the UCI Message dataset.

TABLE III
AVERAGE OF AUC RESULTS WITH DIFFERENT PARAMETERS.

Network layer		Dimention layer					
		16	32	64	128	256	512
UCI Message	1	0.8327	0.8632	0.8702	0.9023	0.9067	0.9005
	2	0.8110	0.8559	0.8871	0.9122	0.9083	0.9053
	3	0.8212	0.8707	0.9034	0.9207	0.9027	0.9073
	4	0.8332	0.8625	0.8989	**0.9235**	0.9101	0.8883
	5	0.8601	0.8677	0.8798	0.9181	0.8989	0.8911
	6	0.8411	0.8513	0.8717	0.9124	0.9001	0.8765

AD-TIN achieves promising performance with the dimension parameter $d_v = 128$ and information propagation layer $l = 4$ on the UCI Message. The results are first to increase and then decrease, indicating that smaller dimensions are insufficient to express node features, while larger dimensions will increase the complexity of the framework and make it more challenging to converge to the optimal solution. Similarly, as the number of layers for network information dissemination increases, the algorithm's performance remains slightly decreased. This observation implies that an excessive number of layers may capture extraneous high-order neighbor information, potentially diminishing accuracy. Therefore, it is important to appropriately set hyperparameters to achieve competitive performance.

On the other hand, we investigate the impact of the training ratio on AD-TIN. Specifically, the training ratios of all datasets are set to 10%, 20%, 30%, 40%, 50%, and 60%, respectively. Figure 5 presents the AUC results using different training ratios on three datasets, with multiple network snapshot results presented in box graphs. The following findings are revealed: (i) Increasing the training ratio improves the median and average AUC, and the model's performance benefits from more data without overfitting specific datasets. (ii) Overall, AUC remains stable with varying training ratios. This finding implies that the proposed algorithm has strong robustness.

One possible reason is that AD-TIN considers the impact of unrelated interactions and noisy data, which improves system performance and stability.

C. Anti-interference Experiment

To further explore the algorithm's ability to resist noise interference, we conduct a noise resistance experiment using an equal number of random edges instead of a certain number of edges from the test set. Specifically, we replaced 3%, 6%, 9%, 12%, and 15% of the edges in the training set with random edges (i.e., noise), and 5% of anomaly edges are inserted into the test set. Figure 6 illustrates the experimental results of AD-TIN at different noise ratios in the form of box plots, which show that the experimental performance of AD-TIN is slightly lower than that of the experiment performed on noiseless data. The average AUC of AD-TIN on the three datasets with added noise differed from the average AUC score on noiseless data by no more than 0.02, suggesting that AD-TIN has a specific ability to resist noise interference.

VII. Conclusion

This study proposes a novel approach, AD-TIN, for detecting target edges in graphs while mitigating the challenges of noise and unrelated edges. The proposed method employs a multi-representation attention learning technique to mitigate the impact of unrelated interactions on feature extraction and reduce information propagation between nodes with unrelated features. In addition, the algorithm's robustness to noisy data is enhanced by combining two neighbor sampling methods: graph diffusion and dynamic random walk auxiliary neighbor sampling. The effectiveness and stability of AD-TIN are validated through detailed comparative experiments conducted on three real datasets. As a direction for future work, we will further explore different types of sampling methods to enhance the operational efficiency of the model.

Acknowledgment

This work was supported by the Basic Science Research Program through the National Research Foundation of Korea (NRF) funded by the Ministry of Education (NRF-2021R1I1A3043889) and the NRF grant funded by the Korea government (MSIT) (No.2021R1A5A1021944).

References

[1] S. Wang and P. S. Yu, "Graph neural networks in anomaly detection," *Graph Neural Networks: Foundations, Frontiers, and Applications*, pp. 557–578, 2022.

[2] L. Gutiérrez-Gómez, A. Bovet, and J.-C. Delvenne, "Multi-scale anomaly detection on attributed networks," in *Proceedings of the AAAI conference on artificial intelligence*, vol. 34, pp. 678–685, 2020.

[3] I. Souiden, M. N. Omri, and Z. Brahmi, "A survey of outlier detection in high dimensional data streams," *Computer Science Review*, vol. 44, p. 100463, 2022.

[4] S. Ranshous, M. Chaudhary, and N. F. Samatova, "Efficient outlier detection in hyperedge streams using minhash and locality-sensitive hashing," in *Complex Networks & Their Applications VI: Proceedings of Complex Networks 2017 (The Sixth International Conference on Complex Networks and Their Applications)*, pp. 105–116, Springer, 2018.

[5] L. Zheng, Z. Li, J. Li, Z. Li, and J. Gao, "Addgraph: Anomaly detection in dynamic graph using attention-based temporal gcn.," in *IJCAI*, vol. 3, p. 7, 2019.

[6] D. Zhu, Y. Ma, and Y. Liu, "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)*, pp. 870–875, IEEE, 2020.

[7] J. Liu, G. P. Ong, and X. Chen, "Graphsage-based traffic speed forecasting for segment network with sparse data," *IEEE Transactions on Intelligent Transportation Systems*, vol. 23, no. 3, pp. 1755–1766, 2020.

[8] X. Guo, B. Zhou, and S. Skiena, "Subset node anomaly tracking over large dynamic graphs," in *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pp. 475–485, 2022.

[9] M. Shao, J. Li, F. Chen, and X. Chen, "An efficient framework for detecting evolving anomalous subgraphs in dynamic networks," in *IEEE INFOCOM 2018-IEEE Conference on Computer Communications*, pp. 2258–2266, IEEE, 2018.

[10] Q. Sun, J. Li, H. Peng, J. Wu, Y. Ning, P. S. Yu, and L. He, "Sugar: Subgraph neural network with reinforcement pooling and self-supervised mutual information mechanism," in *Proceedings of the Web Conference 2021*, pp. 2081–2091, 2021.

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

[12] G. Xue, M. Zhong, J. Li, J. Chen, C. Zhai, and R. Kong, "Dynamic network embedding survey," *Neurocomputing*, vol. 472, pp. 212–223, 2022.

[13] L. Ouyang, Y. Zhang, and Y. Wang, "Unified graph embedding-based anomalous edge detection," in *2020 International Joint Conference on Neural Networks (IJCNN)*, pp. 1–8, IEEE, 2020.

[14] Y.-Y. Chang, P. Li, R. Sosic, M. Afifi, M. Schweighauser, and J. Leskovec, "F-fade: Frequency factorization for anomaly detection in edge streams," in *Proceedings of the 14th ACM International Conference on Web Search and Data Mining*, pp. 589–597, 2021.

[15] C. Liu, L. Sun, X. Ao, J. Feng, Q. He, and H. Yang, "Intention-aware heterogeneous graph attention networks for fraud transactions detection," in *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining*, pp. 3280–3288, 2021.

[16] Y. Zhang, X. Wang, C. Shi, X. Jiang, and Y. Ye, "Hyperbolic graph attention network," *IEEE Transactions on Big Data*, vol. 8, no. 6, pp. 1690–1701, 2021.

[17] Y. Liu, Z. Li, S. Pan, C. Gong, C. Zhou, and G. Karypis, "Anomaly detection on attributed networks via contrastive self-supervised learning," *IEEE transactions on neural networks and learning systems*, vol. 33, no. 6, pp. 2378–2392, 2021.

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

[19] S. Ji, S. Pan, E. Cambria, P. Marttinen, and S. Y. Philip, "A survey on knowledge graphs: Representation, acquisition, and applications," *IEEE transactions on neural networks and learning systems*, vol. 33, no. 2, pp. 494–514, 2021.

[20] T. Opsahl and P. Panzarasa, "Clustering in weighted networks," *Social networks*, vol. 31, no. 2, pp. 155–163, 2009.

[21] M. De Choudhury, H. Sundaram, A. John, and D. D. Seligmann, "Social synchrony: Predicting mimicry of user actions in online social media," in *2009 International conference on computational science and engineering*, vol. 4, pp. 151–158, IEEE, 2009.

[22] R. A. Rossi and N. K. Ahmed, "The network data repository with interactive graph analytics and visualization," in *AAAI*, 2015.

[23] S. Ranshous, S. Harenberg, K. Sharma, and N. F. Samatova, "A scalable approach for outlier detection in edge streams using sketch-based approximations," in *Proceedings of the 2016 SIAM international conference on data mining*, pp. 189–197, SIAM, 2016.

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