Anonymous Edge Representation for Inductive Anomaly Detection in Dynamic Bipartite Graph

Lanting Fang
Southeast University and Purple Mountain Laboratories
China
lantingf@outlook.com

Kaiyu Feng
Beijing Institute of Technology
China
fengky@bit.edu.cn

Jie Gui
Southeast University and Purple Mountain Laboratories
China
guijie@seu.edu.cn

Shanshan Feng
Harbin Institute of Technology (Shenzhen)
China
victor_fengss@foxmail.com

Aiqun Hu
Southeast University and Purple Mountain Laboratories
China
aqhu@seu.edu.cn

ABSTRACT

The activities in many real-world applications, such as e-commerce and online education, are usually modeled as a dynamic bipartite graph that evolves over time. It is a critical task to detect anomalies inductively in a dynamic bipartite graph. Previous approaches either focus on detecting pre-defined types of anomalies or cannot handle nodes that are unseen during the training stage. To address this challenge, we propose an effective method to learn anonymous edge representation (AER) that captures the characteristics of an edge without using identity information. We further propose a model named AER-AD to utilize AER to detect anomalies in dynamic bipartite graphs in an inductive setting. Extensive experiments on both real-life and synthetic datasets are conducted to illustrate that AER-AD outperforms state-of-the-art baselines. In terms of AUC and F1, AER-AD is able to achieve 8.38% and 14.98% higher results than the best inductive representation baselines, and 6.99% and 19.59% than the best anomaly detection baselines.

PVLDB Reference Format:
Lanting Fang, Kaiyu Feng, Jie Gui, Shanshan Feng, and Aiqun Hu. Anonymous Edge Representation for Inductive Anomaly Detection in Dynamic Bipartite Graph. PVLDB, 16(5): 1154 - 1167, 2023.
doi:10.14778/3579075.3579088

PVLDB Artifact Availability:
The source code, data, and/or other artifacts have been made available at https://github.com/fanglanting/AER.

1 INTRODUCTION

With the rapid development of online services, user activities in many applications can be modeled as a dynamic bipartite graph, where the nodes can be divided into source nodes and destination nodes, and each edge connects a source node to a destination node. New nodes and edges are inserted into the dynamic bipartite graph over time. For instance, e-commerce can be modeled as a dynamic bipartite graph, where consumers and items are the two sets of nodes, and an edge represents a consumer buying an item. As another example, student engagements in Massive Open Online Courses (MOOCs) can also be modeled as a streaming bipartite graph, where students and courses are the two sets of nodes and an edge represents a student's access behavior to a course.

As the dynamic bipartite graph serves as the basis for many important tasks like query and recommendation in online services, it has become the focus of various attacks [9, 25, 34, 45]. It is a critical task to detect anomalies to strengthen protection against attacks especially on dynamic graphs. In this paper, we focus on the inductive anomaly detection problem in dynamic bipartite graphs. Specifically, given a dynamic bipartite graph, we aim to learn from partially observed data and apply the learned model to predict whether a newly observed edge is anomalous. The inductive anomaly detection problem is a daunting task due to the following challenges: **(1)** The inductive setting indicates that new nodes that have not been seen before would appear as the graph evolves. **(2)** Many graphs in real world are associated with limited or even none attribute. **(3)** There is no explicit definition of anomaly due to its flexible and dynamic nature.

At first glance, it may seem that existing studies including learning-based anomaly detection methods and graph neural networks (GNN) can be adopted to address the inductive anomaly detection problem. Unfortunately, this is not the case, as they cannot address the above three challenges simultaneously. Specifically, a host of machine learning based methods [12, 31, 43, 46, 55] assign a learnable vector for each node in the dynamic graph to capture the time-evolving patterns of the graph structure. These methods are not inductive, because the learned model cannot handle new nodes that have not been seen before. Therefore, they fail to address the first challenge. Several GNN-based approaches [8, 15, 47] aggregate node/edge features to learn edge representations so that they can handle unseen nodes and conduct inductive anomaly detection. However, in order to infer edge representations for anomaly detection, all of them are highly dependent on node/edge attributes, which could be absent in practice. Thus, they fail to tackle the second challenge. Many existing anomaly detection methods detect pre-defined anomalous

Kaiyu Feng is the Corresponding Author

Proceedings of the VLDB Endowment, Vol. 16, No. 5 ISSN 2150-8097.
doi:10.14778/3579075.3579088

changes to the graph, such as sudden appearance of subgraphs [11], burstiness of edges [6], hotspot nodes [50], prompt group change [5], and sudden arriving of similar edge groups [4]. Though these methods are capable of handling networks without attributes and detect anomalies in an inductive setting, it requires us to know which type of anomaly would occur a priori, which is impractical in real life. As above existing approaches fail to address all three challenges, it necessitates a new design of model that can detect anomalies based on structure information of dynamic graphs in an inductive setting.

In this paper, we propose a solution named ***A**nonymous **E**dge **R**epresentation-**A**nomaly **D**etection (AER-AD)*. Note that the term "anonymous" means that we do not directly utilize the node identity to encode an edge representation. Instead, we utilize the relation between the edge and its local graph structure, which makes it possible for us to infer the representation for a newly observed edge, even if the node has not been seen before.

AER-AD has two important properties: (1) It uses local graph structure to represent edges. This guarantees the inductive capability. (2) It adopts a mask-based method to select the appropriate local graph structure to preserve information for anomaly detection. This guarantees the prediction accuracy. The prediction module predicts whether the target edge $(\hat{u}, \hat{v}, \hat{t})$ is anomalous. It takes the AERs of the past edges from $\hat{u}$ and encodes them with gated recurrent unit. We conduct extensive experiments on three real-life datasets and two synthetic datasets. Our empirical evaluation illustrates that the proposed solution outperforms the best inductive representation baselines by 14.98% and 8.38% in terms of F1 and AUC, and outperforms the best anomaly detection baselines by 19.59% and 6.99%. We also show that the proposed solution is highly efficient. It achieves high throughput and is robust to hyper-parameters.

The main contributions of this paper are summarized as:

- To the best of our knowledge, we are the first to study the problem of inductive anomaly detection in both dynamic graphs with and without attributes, which addresses the limitation of existing anomaly detection methods.
- We design an effective method to learn anonymous edge representation that capture the characteristics of an edge without using its identity information while preserving sufficient information to distinguish anomalous edges from the normal ones.
- Based on the AER, we further design a model AER-AD to capture the temporal and structural changes in dynamic bipartite graphs. AER-AD is highly efficient, which conduces to online anomaly detection.
- The experiments demonstrate that AER-AD significantly outperforms state-of-the-art methods, and is robust in performance and computationally efficient.

2 RELATED WORK

Anomaly detection in dynamic graph. Many efforts have been devoted to anomaly detection in dynamic graphs. In the following, we briefly discuss their methodological foundations and related limitations.

Matrix factorization methods [37, 39, 51] leverage the "lowrank" property to capture the structural information of graphs. These methods always require a high complexity and cannot provide a timely detection. Distance-based methods [10, 26, 29, 49, 52, 53] propose time-evolving measures for nodes, edges or graph structures and define anomaly as a surge of the change rates of the measures. Probabilistic methods [1, 4, 23, 33, 42] rely on probabilistic models to characterize the normal patterns of the temporal graph and determine anomalies based on the pattern deviation from the models. [17] focuses on detecting spot fraudsters in the presence of camouflage or hijacked accounts based on a density metric. [36] identifies dense subtensors created within a short time. All these methods rely on measures for pre-defined type of anomalies. In contrast, our solution is capable of detecting *unknown anomalies*.

Recently, some machine learning methods [12, 31, 43, 46, 55] propose to detect undefined anomalies by learning a vector representation for each node in the graph. These methods are transductive and cannot handle unseen node. AEGIS [8] first studies the problem of inductive anomaly detection in graphs by proposing a graph differentiation network (GDN) to learn node representations from arbitrary-order neighborhoods. This method heavily depends on the node features and thus can only be used for graphs with rich attributes. IGRL [3] applies existing inductive representations to solve the anomaly detection problem. This method highly rely on the quality of representation. In contrast, our proposed AER represents an edge based on the graph structure , which makes it capable of handling *unseen nodes* in both graphs *with* and *without attributes*.

Table 1: AER-AD competitors: comparison of our proposed AER-AD and existing methods for anomaly detection in dynamic graphs. ✓/× indicate that the methods are able/not able to handle the property. * indicates some of this kind of methods are able to handle the property and some not. M/D/P represents matrix factorization methods [37, 39, 51], ML represents machine learning based transductive anomaly detection methods [12, 31, 43, 46, 55], DGR represents dynamic graph representation methods, and DGR-T represents transductive DGR [21, 40, 56].

Property	Anomaly Detection				DGR				
	M/D/P	ML	AEGIS [8]	IGRL [3]	DGR-T	TGAT [47]	CAW [44]	Graphsage [15]	AER-AD
Unknown anomalies	×	✓	✓	✓	✓	✓	✓	✓	✓
Unseen nodes	*	×	✓	✓	×	✓	✓	✓	✓
Without attributes	✓	*	×	✓	*	×	✓	✓	✓
With attributes	×	*	✓	✓	*	✓	✓	✓	✓
Dynamic graph	✓	*	×	×	✓	✓	✓	×	✓

Dynamic graph representation. A host of work [27, 28, 30, 54] has been proposed to learn representations for nodes in dynamic graphs for various tasks. Most of them aggregate the sequence of edges within consecutive time windows into network snapshots, and use machine learning models to encode the snapshots [14, 16, 19, 32, 35]. However, these approaches need to predetermine a time granularity for edge aggregation, which is difficult to learn

structural dynamics in different time scales. Therefore, approaches that directly work on edge streams have been proposed recently. DyRep [40] proposes a two-time scale deep temporal point process model to capture the information of two processes, namely dynamics of the network and dynamics on the network. JODIE [21] proposes a novel embedding technique to learn joint user and item embeddings from sequential data for dynamic graphs. DynamicTriad [56] studies on the triadic closure process to catch how open triads evolve into closed triads. However, all of the above models are not inductive as they assign a learnable vector to each node in the dynamic graph.

Recently, some inductive representation methods such as TGAT [47] propose to identify a node by aggregating features from its neighborhood. However, in these methods, the over smoothing between connected nodes may lead to a reduction of distinguishable information that can be preserved to support anomaly detection. CAW [44] inductively represents temporal networks based on temporal random walks. It adopts a novel anonymization strategy that replaces node identities with the hitting counts of the nodes based on a set of sampled walks to keep the method inductive. However, CAW is inexpressive for characterizing the edge abnormality. As each node on a random walk is encoded separately, CAW disregards the relationship between a node's interactions history and the normal pattern. Such information plays an important role in anomaly detection. Therefore, the performance of CAW is limited.

Table 1 compares AER-AD to existing methods.

3 PROBLEM FORMULATION

In this section, we formally introduce the inductive anomaly detection problem in dynamic bipartite graphs.

Definition 3.1 (Dynamic bipartite graph). A bipartite graph stream is an unbounded time-evolving sequence of edges $\mathcal{S} = \langle e_1, \ldots, e_n \rangle$, where each edge $e_i = (u, v, t)$ indicates an interaction from a source node u to a destination node v at time t. The stream $\mathcal{S}$ forms a **dynamic bipartite graph** $G(U, V, E)$, where U and V are the sets of *source nodes* and *destination nodes*, and E is the set of edges. We use $f(u) \in \mathbb{R}^{d_n}$ and $f(u, v, t) \in \mathbb{R}^{d_e}$ to denote the feature of a node u and an edge (u, v, t), respectively.

User activities in many real-life applications can be modeled as a dynamic bipartite graph. For instance, we can model the transactions in e-commerce as a dynamic bipartite graph, where the source and the destination nodes are consumers and items, and an edge (u, v, t) represents that a consumer u purchases an item v at time t. As another example, the learning activities in Massive Open Online Courses (MOOC) can also be modeled as a dynamic bipartite graph, where an edge (u, v, t) represents a student u's access behavior to a course at time t.

Anomaly detection in dynamic bipartite graphs is critical for many application scenarios. In e-commerce, we could identify and prevent fraudulent transactions to protect consumers and businesses. In MOOC, we could predict students' dropout to provide early intervention. In the aforementioned examples, the destination nodes are managed by enterprises and are unlikely to be anomalies. We focus on predicting whether a newly-arrived edge from a source node in the stream is anomalous. In addition, as the dynamic bipartite graph is evolving, new nodes and edges are inserted into the graph. For example, new customers continuously sign up and appear in the e-commence stream. It is necessary to handle unseen nodes. Therefore, we investigate the inductive anomaly detection problem in dynamic bipartite graphs.

Definition 3.2 (Inductive Anomaly Detection Problem). Consider a dynamic bipartite graph $G(U, V, E)$. Let $G_t = (U_t, V_t, E_t) \subseteq G$ be a subgraph of G which is observed for training and $G' = (U', V', E') \subseteq G$ is a subgraph of G which is newly observed for testing. The inductive anomaly detection problem aims to learn a model from G_t and use the model to predict whether a newly observed edge $\hat{e} = (\hat{u}, \hat{t}, \hat{t}) \in E'$ is anomalous.

4 INDUCTIVE ANOMALY DETECTION

In this section, we present the details of our proposed model named *AER-AD* for inductive anomaly detection.

4.1 Overview

Challenges. Due to the evolution of the dynamic bipartite graph, new nodes that are not seen before may appear. Therefore, it is a major challenge for inductive anomaly detection to represent the edges: *The edge representation has to be independent from node identity and preserve enough information so that we can distinguish an anomalous edge from the normal ones even if the nodes are not seen before.* To handle unseen nodes, previous work [8, 47] aggregates temporal neighborhood features to infer representation for new nodes and edges. However, these methods rely on node/edge features and cannot be generalized to graphs with no additional features, which are very common in real life.

In this paper, we design an effective solution named ***A**nonymous **E**dge **R**epresentation-**A**nomaly **D**etection (AER-AD)* for inductive anomaly detection. Specifically, we design *Anonymous Edge Representation* (AER), which captures the characteristics of an edge without using the node identities. AER is the key to guarantee the inductive capability of our solution.

Solution Overview. The architecture of the proposed solution *AER-AD* is illustrated in Figure 1. Given the bipartite graph stream, let $(\hat{u}, \hat{v}, \hat{t})$ be the recent edge that represents a source node $\hat{u}$ interacting with a destination node $\hat{v}$ at time $\hat{t}$. *AER-AD* first encodes the recent edges from $\hat{u}$ through the *representation module* (Figure 1 A). Specifically, since the node identity information cannot be utilized due to the inductive setting, the edges are first anonymized based on the relation between the edge and the local graph structure via *edge anonymization* (Figure 1 A.1). The anonymization is conducted from both the source and the destination nodes' aspects, respectively. The anonymized description of the edge is then passed to *representation learning* (Figure 1 A.2), which outputs the anonymous edge representation (AER) of the edge. The AERs of the retrieved edges from $\hat{u}$ are fed into the prediction module (Figure 1 B) to predict whether the target edge $(\hat{u}, \hat{v}, \hat{t})$ is anomalous.

4.2 Representation Module

The representation module is designed to generate anonymous edge representations (AERs). The AERs are expected to be independent from node identities and preserve sufficient information to distinguish the anomalous edges from the normal ones. To this

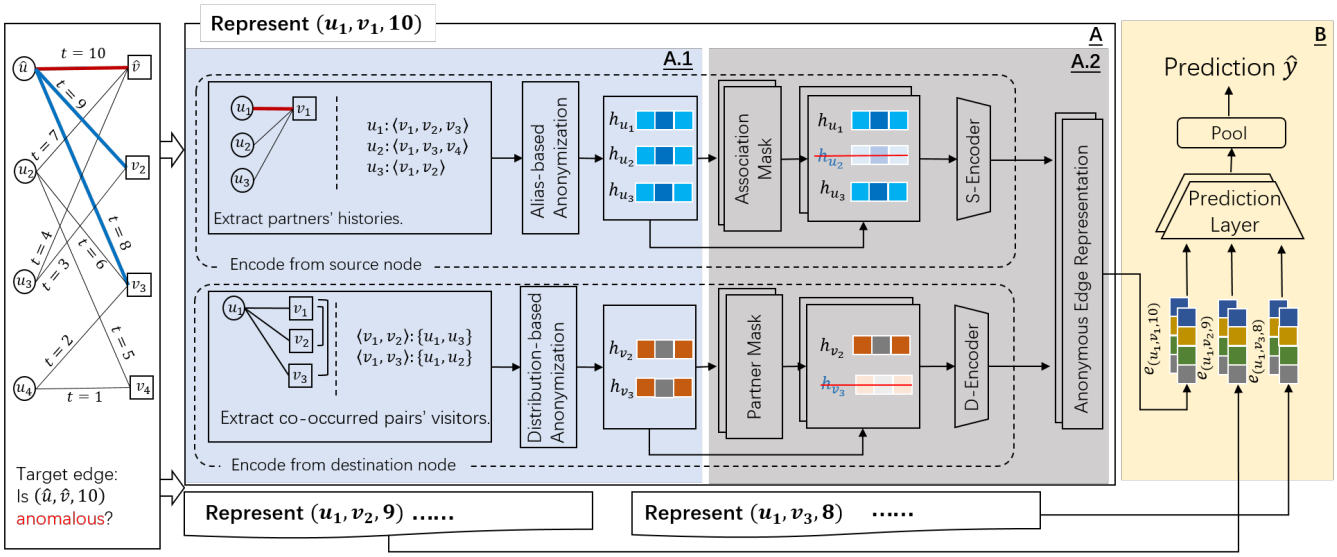


Figure 1: Illusion of the anomaly detection framework *AER-AD*. (A) Representation Module; (A.1) Edge Anonymization module; (A.2) Representation Learning Module; (B) Prediction Module.

end, the representation module consists of two components: (1) *Edge Anonymization* removes node identities and generates an anonymized description of the edge. (2) *Representation Learning* learns an anomaly-aware compact representation for each edge from the anonymized description.

For the ease of presentation and consistency, we always use u and v to denote the source and destination nodes of the edge whose AER we aim to learn. Before we present the design of the representation module, we first introduce two definitions that will be used later.

Definition 4.1 (Partners). Let (u, v, t) whose AER we aim to learn. The partners $P_{u,v,t}$ is the set of source nodes which have already visited v before time t: $P_{u,v,t} = \{u_i | (u_i, v, t_i) \in E \wedge t_i \leq t \wedge u_i \neq u\}$

In the remaining of this section, we use P to denote the partners for simplicity when the edge (u, v, t) can be inferred from the context.

Definition 4.2 (History). Let (u, v, t) be the edge whose AER we aim to learn, and P be the partners. For each $u_i \in P$, its history $H(u_i)$ is the sequence of destination nodes visited by u_i till u_i visited v, in the descending order of time. $H(u_i) = [v, v_1, \ldots, v_l]$, where $\forall j \in [1, l]$, $(u_i, v_j, t_j) \in E$, $t_j \leq t$, and $t_{i+1} \leq t_i$.

We use the following example to explain partners and their histories.

Example 4.3. Consider the example in Figure 2, where we aim to get the AER of (u, v, t). The partners are $P = \{u_1, u_2\}$, as both u_1 and u_2 visited v before time $t = 10$. History is defined for u and the partners. The histories of u, u_1 and u_2 are shown in the figure. Since u_3 is not a partner, we do not consider the destination nodes that u_3 visited.

Next, we elaborate on the two components, namely *edge anonymization* and *representation learning* in turn.

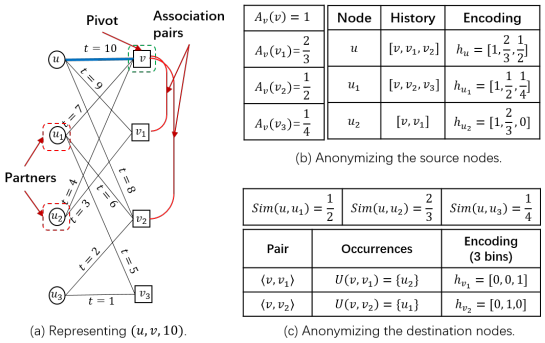


Figure 2: An example bipartite graph, where $(u, v, 10)$ is the edge whose AER we aim to get.

4.2.1 Edge Anonymization. As new nodes may appear due to the involvement of the dynamic bipartite graph, the inductive anomaly detection model cannot utilize the identity information of the nodes or edges. Edge anonymization module is thus designed to remove node identities and generate an anonymized description of the edge. The anonymized description keeps sufficient information from which we can learn the anomaly-aware AER for the edge.

The high level idea is as follows: Given an edge (u, v, t), we remove the node identities by utilizing the relation between the source/destination nodes and their local graph structures, respectively. Such a relation gives the node a relative identity, which preserves the characteristics of the edges and their correlations. A question to be asked is: *What local graph structure should we consider in removing node identities?* In fact, as source node and destination node play different roles in the bipartite graph, their local graph structures should be treated differently. We next elaborate on how to choose the local graph structure and utilize the relation

to anonymize node identities for source node and destination node in turn.

Removing source node's identity. As mentioned above, we would like to utilize the relation between u and its local graph structure to remove the source node u's identity. According to Definition 4.1, u and the partners P have interacted with v. It is observed that nodes sharing common neighbors in a bipartite graph are expected to exhibit similar behaviors [24, 38]. Motivated by this, we may remove the source node identity and generate an anonymized description of u based on the behaviors of u and the partners P. Intuitively, the behavior of a source node is reflected in its history, i.e., the sequence of destination nodes that the source node has visited before. Therefore, it is a natural idea to encode the histories of u and the partners P and group them as the anonymized description of u.

A naive idea for encoding a history is as follows: We first assign each destination node a unique id. The sequence of destination node in the history is converted to a vector directly, where the i-th dimension of the vector corresponds to the i-th node in the history. For instance, in Figure 2, history $H(u)$ is encoded as $[0, 1, 2]$ if we assign 0, 1, 2 as the id for v, v_1 and v_2, respectively.

However, there is an issue with this naive idea. Consider an example in e-commerce, where the source nodes and the destinations are customers and products, respectively. Let [Iphone, AirPods] and [Ipad, AirPods] be two histories. Though *Iphone* and *Ipad* correspond to different destination nodes in the bipartite graph, they are actually highly related. It is very likely that the two customers who record the above histories have similar preference. The naive idea disregards the similarity between destination nodes and fails to fully utilize the correlation across histories.

The above example tells us that good encodings of histories should capture the similarity between destination nodes from different histories. However, this is non-trivial. Given two histories H_1 and H_2, we can make $|H_1| \cdot |H_2|$ pairs of destination nodes from the two histories. It is expensive to enumerate all pairs and it is not easy to design an encoding that incorporates all the similarities. In order to address this challenge, we design "alias" for each destination node. Specifically, we select v, the destination node of the edge to be represented, as the pivot. For each destination node, the alias $A_v(v_i)$ is defined as

$$A_v(v_i) = \frac{|Nei(v_i) \cap Nei(v)|}{|Nei(v_i) \cup Nei(v)|} \tag{1}$$

, where $Nei(v)$ denotes the set of source nodes that are adjacent to v. Our design has the following consideration: (1) The alias of a destination node v_i evaluates the Jaccard similarity between the neighbors of v_i and v. It is solely based on graph structure and is independent from node identities. (2) The similarity between destination nodes are captured by their aliases. For instance, in the previous example, both the customers who bought Iphone and the customers who bought Ipad are likely to buy AirPods. Since the alias of Iphone (Ipad) evaluates the overlap between the buyers of Iphone (Ipad) and AirPods, it is very likely that the alias of Iphone is close to the alias of Ipad, which captures the similarity between Iphone and Ipad.

With the alias defined, we encode the history of u_i for each $u_i \in P \cup \{u\}$ as follows:

$$h_{u_i} = [A_v(v), A_v(v_1), \ldots, A_v(v_l)], \tag{2}$$

where $[v, v_1, \ldots, v_l]$ is the history of u_i.

We encode the histories of u and the partners and group the history encodings as the anonymized description of u:

$$\mathcal{D}_s(u) = [h_u, h_{u_1}, \ldots, h_{u_p}], \tag{3}$$

where h_u/a_{u_i} is the history encoding of u/u_i and $P = \{u_1, \ldots, u_p\}$ is the set of partners.

Example 4.4. Figure 2 shows an example of removing the source node's identity from the edge $(u, v, 10)$. In the example, the partners are $P = \{u_1, u_2\}$. Destination node v is selected as the pivot for alias computation. Take node v_1 as an example. The neighbors of v_1 is $Nei(v_1) = \{u, u_2\}$, while the neighbors of v is $Nei(v) = \{u, u_1, u_2\}$. Thus, the alias of v_1 is $A_v(v_1) = \frac{|Nei(v) \cap Nei(v_1)|}{|Nei(v) \cup Nei(v_1)|} = \frac{2}{3}$. With the aliases computed, we extract the history of u and the partners and encode the histories, as shown in Figure 2(b). Note that all history encodings should have the same length. Since the length of the history $H(u_2)$ is 2, we extend its history encoding with zero-padding.

Removing destination node's identity. Next, we present how to utilize v and its local graph structure to remove the destination node v's identity. We observe that the associations between v and other destination nodes that u has visited before implies the dynamic rule of how the graph evolves. For instance, it is very likely that a user purchases a phone case after buying a mobile phone. Such associations are useful for describing the destination node anonymously.

For each $v_i \in H(u)$, if the pair $\langle v, v_i \rangle$ is often co-visited by other source nodes, then it is less suspicious for u to visit v at time t. Motivated by this, we propose to enumerate the pairs $\langle v, v_i \rangle$ for any $v_i \in H(u)$ and encode the popularity of the pairs as the anonymized description of v.

A straightforward idea for encoding the popularity of the pairs is to use the frequency of the pairs directly. Specifically, let $v_i \in H(u)$ be a destination node that u visited before. For each pair $\langle v, v_i \rangle$, we compute the set $U(v, v_i)$ of source nodes which have interacted with both v and v_i:

$$U(v, v_i) = \{u_j | \exists (u_j, v, t_1) \in E \wedge \exists (u_j, v_i, t_2) \in E \wedge u_i \neq u\} \tag{4}$$

We can then encode the popularity of all pairs as $\mathcal{D}'_d = [|U(v, v_1)|, \ldots, |U(v, v_l)|]$, where $[v_1, \ldots, v_l]$ is the history $H(u)$, and the i-th dimension is the number of source nodes that visited both v and v_i.

Example 4.5. Reconsider the example in Figure 2. Source node u has visited v, v_1, and v_2. Thus, we can make two pairs $\langle v, v_1 \rangle$, and $\langle v, v_2 \rangle$. We observe that $U(v, v_1) = \{u_2\}$, and $U(v, v_2) = \{u_1, u_3\}$. Thus, we can encode the popularity of all pairs as $\mathcal{D}'_d = [1, 2]$.

Though the idea of using frequency to encode the popularity of all associations is easy to implement, it suffers from the following issue: It treats the source nodes equally and disregards the differences across the source nodes. Take e-commerce as an example. Assume that a computer scientist purchases a keyboard after buying a laptop and a football. There are two pairs of associations: ⟨keyboard, laptop⟩ and ⟨keyboard, football⟩. We assume that a software engineer also buys a keyboard and a laptop, and a student buys a keyboard and a football. The frequency-based method encodes the popularity as $[1, 1]$, since each pair is co-visited by only one

source node. However, as computer scientist has a lot in common with software engineer, the association between ⟨keyboard, laptop⟩ should be more strong.

To address the aforementioned problem, we propose a distribution-based method to encode the association pairs and anonymize the destination nodes. We incorporate the relation between u and other source nodes that have the association pairs into the encoding of the association popularity. Specifically, for each association pair $\langle v, v_i \rangle$, we compute the similarity between u and every $u_j \in U(v, v_i)$. We build a histogram from the similarities and use the distribution of the similarities as the encoding of the association popularity.

$$h_{v_i} = histogram(Sim(u, u_j) | u_j \in U(v, v_i)), \tag{5}$$

where the similarity between two source nodes is the Jaccard similarity between their neighbors, i.e., $Sim(u, u_j) = \frac{Nei(u) \cap Nei(u_j)}{Nei(u) \cup Nei(u_j)}$.

The anonymized description of the destination node is the encodings of the popularity all all association pairs.

$$\mathcal{D}_d(v) = [h_{v_1}, \ldots, h_{v_l}] \tag{6}$$

where $[v_1, \ldots, v_l]$ is the sequence of destination nodes that u has visited before (u, v, t).

Example 4.6. Reconsider the example in Figure 2. The anonymization of the destination node is shown in Figure 2. The pair $\langle v, v_1 \rangle$ is co-visited by u_2. Thus, we compute the similarity between u and u_2, $Sim(u, u_2) = \frac{2}{3}$. Assume that we use a histogram with three equal-sized bins $[0, 1/3)$, $[1/3, 2/3)$, $[2/3, 1]$ to group the similarities. The similarity $Sim(u, u_2)$ falls in the bin $[2/3, 1]$. Thus, the popularity of $\langle v, v_1 \rangle$ is encoded as $[0, 0, 1]$. In a similar way, the popularity of $\langle v, v_2 \rangle$ is encoded as $[0, 1, 0]$.

The anonymized descriptions of the source node and the destination node $[\mathcal{D}_s, \mathcal{D}_d]$ together forms the anonymized description of the edge. The anonymized description is solely dependent on the graph structure and is irrelevant to the node identity. Even if the nodes in the edge are not seen before, we can still generate such description.

Partners and history truncation. When the bipartite graph is dense, there could be many nodes that have visited v before, leading to a very large size of partners and histories. It would be inefficient to use all nodes in P and $H(v)$. Intuitively, the nodes that are visited recently have a higher impact than the nodes that are visited long time ago. Therefore, we only keep the most recent p and l nodes as the partners and history, respectively.

Sometimes graphs need a large partner size to achieve best performance, this will harm the efficiency of the propose model. For these scenarios, we propose an efficient partners sampling method to improve the performance of the proposed model w.r.t. small partner size. Given a node v, we assume nodes with similar degree or time span of all interactions to v are more important. Hence, we sample a partner with probability proportional to $e^{\beta|\Delta d|}$ or $e^{\beta|\Delta t|}$, where β is a hyper-parameter, Δd and Δt are degree and time span distance between node v and its partner, respectively.

4.2.2 Representation Learning. The representation learning module is designed to learn AERs for the edge (u, v, t) from its anonymized description $[\mathcal{D}_s, \mathcal{D}_d]$. The learned AERs are expected to highlight the distinguishable information for anomaly detection.

To learn good AERs, we need to answer the question: How does u visiting v at time t conform to normal pattern? This question can be answered from two perspectives. From the source node u's perspective, the normal pattern could be implied by the behaviors of the source nodes which share common interest. A simple idea is to directly infer the normal pattern from the partners. However, this idea is oversimplified. A source node may have multiple roles in the bipartite network. For instance, a computer scientist who just starts hiking may buy the same hiking stick as a hiking enthusiast. However, the behaviors of the computer scientist could be very different from the hiking enthusiast. This example indicates that we need to eliminate inappropriate partners when inferring the normal pattern for the source nodes.

From the destination node v's perspective, the normal pattern is implied in the popularity of the association rules, i.e., how normal it is for a source node to visit v after visiting v_i. However, similar to the source node, not all association pairs should be used. Sometimes the two destination nodes in a pair are irrelevant. For instance, the computer scientist may just bought a graphics card before he buys the hiking stick. The associations between graphics card and hiking stick are noises and should be eliminated.

The aforementioned issues motivate us to design *partner mask* and *association mask* to eliminate inappropriate partners and association pairs.

Partner Mask and Association Mask. Let $M_p \in \mathbb{R}^s$ and $M_a \in \mathbb{R}^l$ denote the partner mask and the association mask, respectively. Both masks are binary vectors, where $M_p[i] = 1$ ($M_a[i] = 1$) indicates the i-th partner $u_i \in P$ (the i-th association pair $\langle v, v_i \rangle$) is selected.

We assume that the two masks are subject to the Bernoulli distribution $M_*[i] \sim Bern(\theta_i)$ ($M_* \in \{M_p, M_a\}$), where the probability of $M_*[i] = 1$ is equal to θ_i. This stochastic binary mask captures more variability information compared with methods that uses deterministic functions to generate masks [48].

In order to learn the two masks, we adopt a re-parameterization trick [18] to approximate the sampling process and generate the mask matrix by:

$$\begin{aligned} &Uni \sim \text{Uniform}(0, 1), \\ &L = \log(\alpha) + \log(Uni) - \log(1 - Uni), \\ &\hat{M}_i^* = \frac{1}{1 + exp(-L/\lambda)} \end{aligned} \tag{7}$$

where $M_* \in \{M_p, M_a\}$ is one of the masks, $\log(\alpha)$ is the location parameter (to be explained later) and λ is the temperature parameter that controls the degree of approximation. As $\lambda \to 0$, the random variable $\hat{M}_i^*$ converges to Bernoulli distribution with parameter $\frac{\alpha}{1-\alpha}$.

The location parameter $\log(\alpha)$ is generated in a data-driven way. Specifically, for the partner mask, $\log(\alpha)$ is generated as follows.

$$\log \alpha = W_2^m \cdot \text{Relu}(W_1^m \cdot ([a_u || a_{u_i} || \Gamma_u]) + b_1^m) + b_2^m \tag{8}$$

For the association mask, $\log(\alpha)$ is generated as follows.

$$\log \alpha = W_4^m \cdot \text{Relu}(W_3^m \cdot ([a_{v_i} || \Gamma_v]) + b_3^m) + b_4^m, \tag{9}$$

Here, $\Gamma_u = \frac{1}{|P|}\sum_{u_i \in P} a_{u_i}$ is the center of the encodings in the anonymized description $\mathcal{D}_s$ of u, $\Gamma_u = \frac{1}{|P|}\sum_{u_i \in P} a_{u_i}$ is the center of the encodings in the anonymized description $\mathcal{D}_d$ of v $\cdot||\cdot$ is the concatenation operation, and W_*^m and b_*^m are learnable parameters.

As we mentioned before, the nodes in the bipartite graph may have multiple roles. Therefore, we generate multiple masks. A pair of a partner mask and a association mask selects a group of appropriate partners/association pairs. The selected group of partners and association pairs forms a scene of the anonymized description, from which we will learn an AER for the edge. Specifically, we generate n_s partner masks and n_d association masks, leading to $n_s \cdot n_d$ pairs of masks. We use M_p^i and M_a^j to denote the i-th partner mask and the j-th association mask, respectively.

With the two sets of masks, we can learn AERs from the anonymized description. We design two different networks to learn AER_s and AER_d from $\mathcal{D}_s$ and $\mathcal{D}_d$ separately and fuse them to get the final AER.

Learning AER_s from $\mathcal{D}_s$. Inspired by previous work [41, 52], we design an *S-Encoder* that learns a compact representation of u from $\mathcal{D}_s$ for each partner mask M_p^k. Specifically, the S-Encoder first encodes the partners that are selected by the mask into a latent space as follows.

$$x_P^k = \text{CNN}^s(\{M_p^k[i] \cdot a_{u_i} | \forall u_i \in P\}) \tag{10}$$

Here, a_{u_i} is the encoding of u_i's history in $\mathcal{D}_s$, P is the set of partners, M_p^k is the k-th partner mask, and $\text{CNN}^s(\cdot)$ is a 1D convolution that aggregates the selected partners.

Next, we extract anomaly-aware information based on the relation between u and the selected partners. We concatenate the history encoding a_u and x_P after a linear translation. The concatenation is fed through a fully-connected layer with a non-linear activation function to get the representation x_u of u as follows.

$$x_u^k = \sigma(W_s \cdot [W_1 \cdot a_u || W_2 \cdot x_P^k]), \tag{11}$$

where a_u is the encoding of u's history in $\mathcal{D}_s$, σ is the non-linear activation function, $\cdot||\cdot$ represents the concatenation operation, and W_1, W_2 and W_s are the parameters to be learned.

Since we have n_s partner masks, we get a set of representations $X_u = [x_u^1, \ldots, x_u^{n_s}]$ for u.

Learning AER_d from $\mathcal{D}_d$. With the association mask, we design a *D-Encoder* to learn a compact representation for v from $\mathcal{D}_d$ for each association mask M_a^k. The D-Encoder aggregates the association pairs selected by the association mask and feeds the aggregation through a fully-connected layer with a non-linear activation function to get the representation of v as follows.

$$x_v^k = \sigma(W_d \cdot \text{CNN}^d(\{M_a^k[v_i] \cdot a_{v_i}, \forall v_i \in H(u)\})), \tag{12}$$

where a_{v_i} is the encoding of the popularity of pair $\langle v, v_i \rangle$, $H(u)$ is the history of u, M_a^k is the k-th association mask, CNN^d is 1D convolution, σ is the activation function, and W_d is the parameter to be learned.

As we have n_d association masks, we get a set of representation $X_v = [x_v^1, \ldots, x_v^{n_d}]$ for node v.

Learning *AER*. We have presented X_u and X_v that captures the characteristics of an edge from the source and the destination nodes' perspectives, respectively. Given $x_u^i \in X_u$ and $x_v^j \in X_v$, we fuse them as follows.

$$x_e^{i,j} = f_3([f_1(x_u^i) || f_2(x_v^j)]) \tag{13}$$

where f_1, f_2 and f_3 are feed-forward networks.

We enumerate all $n_s \cdot n_d$ combinations from X_u and X_v and fuse every combination. The final AER of the edge is $X_e = [x_e^1, \ldots, x_e^{n_s \cdot n_d}]$

Remark. In real-world applications, sometimes graphs are associated with additional node/edge attribute features. Anomaly detection would benefit from such features if we integrate them into edge representation. Hence, when additional features are offered, we alternate Equation (13) by:

$$x_e^{i,j} = f_4([f_1(x_u^i) || f_2(x_v^j) || f_3(x_f)]) \tag{14}$$

where x_f represents additional node/edge features.

4.3 Prediction Module

Previously, we have introduced the representation module that learns AERs for each edge. In this subsection, we present how to predict whether the target edge $(\hat{u}, \hat{v}, \hat{t})$ is anomalous based on AER.

We first present how to prepare the input to the prediction module. Intuitively, the past edges from a source node reveal the change of the its preference. Therefore, in order to predict whether $(\hat{u}, \hat{v}, \hat{t})$ is anomalous, we retrieve the past edges from $\hat{u}$ to capture the structural dynamics. Specifically, let $\hat{S}$ denote the sequence of past edges from $\hat{u}$ in the descending order of time, i.e., $\hat{S} = \langle (\hat{u}, \hat{v}, \hat{t}), (\hat{u}, v_1, t_1), \ldots, (\hat{u}, v_{s-1}, t_{s-1}) \rangle$. We next generate $n_s \cdot n_d$ sequences of AERs for $\hat{S}$. The k-th AER sequence is generated as follows.

$$\hat{X}^k = [x_1^k, \ldots, x_s^k], \tag{15}$$

where x_i^k is the k-th AER of the i-th edge in $\hat{S}$. The prediction module takes as input an AER sequence and encode the sequence with a sequence encoder, e.g., Gated Recurrent Unit (GRU) [7]. One may use other RNNs or transformer networks instead of GRU to encode the sequences. The experimental results show the GRU have achieved good enough performance.

With the encoded vector $h_{\hat{X}^k}$, we pass it through a two-layer fully-connected neural network to predict the anomaly score of the target edge. Specifically, we calculate the anomaly score as follows.

$$score^k = \text{Sigmoid}(W_2^o \cdot \text{Relu}(W_1^o \cdot h_X + b_1^o) + b_2^o) \tag{16}$$

where W_*^o and b_*^o are learnable parameters.

Note that each combination of masks defines a scene of the anonymized description. The target edge $(\hat{u}, \hat{v}, \hat{t})$ should be normal as long as it is normal in at least one scene. Thus, the final prediction score for edge $(\hat{u}, \hat{v}, \hat{t})$ is defined as follows.

$$\hat{y} = \max(score^1, \ldots, score^{n_s \cdot n_d}) \tag{17}$$

4.4 The AER-AD Model

The whole procedure of AER-AD is summarized as follows.It first retrieves the past edges $\hat{S}$. Then, for each edge in $\hat{S}$, it generate n_s partition masks and n_d association masks. Based on the generated masks, it learns AER_s and AER_d and fuse them to get the AER. Then it builds $n_s \cdot n_d$ AER sequences for $\hat{S}$. For each AER sequence, it predicts the anomaly score. The final prediction score is computed by a pooling layer over all anomaly scores.

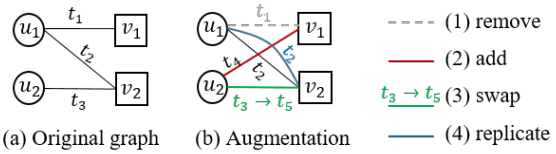


Figure 3: Data augmentation

The final score $\hat{y}$ is computed by a pooling layer over all anomaly scores (lines 15-16). To learn the parameters of AER-AD, we minimize the cross-entropy value:

$$L_\theta = -y \cdot \log(\hat{y}) - (1-y) \cdot \log(1-\hat{y}) \tag{18}$$

where θ denotes all learnable parameters in the entire neural network and y is the true label of the edge (u, v, t). We choose the Adam optimizer to learn θ as it can determine the learning rate adaptive.

Data augmentation. In many real life datasets for anomaly detection, the class labels are usually imbalanced, which may affect the performance of the learned model. In order to improve the performance, we adopt the down-sampling technique to randomly remove samples from the majority class. We also adopt up-sampling techniques to randomly re-sample from the minority class to balance the two classes.

The above method works well on most of the imbalanced datasets. However, when the dataset is extremely imbalanced, i.e., the absolute number of the minority class is very small, we may face the over-fitting problem. We propose to inject synthetic data into the original graph to balance the labels. Specifically, we first randomly sample an edge (u, v_k, t_k) with anomalous label (the minority label). Let E_u be the set of edges from u until time t_k, and t_{min}/t_{max} be the earliest/latest time of edges in E_u. We generate a new node u' and let $E_{u'}$ be the set of replicated edges from E_u, i.e., $\forall(u, v_i, t_i) \in E_u$, we add an edge (u', v_i, t_i) into $E_{u'}$. Next, we conduct six operations to modify $E_{u'}$ while keeping (u', v_k, t_k) fixed: (1) removing random edges from $E_{u'}$, (2) adding edges from u to random destination nodes outside of $E_{u'}$ with time between t_{min} and t_{max}, (3) swapping the time of random pair of edges in $E_{u'}$, (4) replicating random edges in $E_{u'}$, (5) removing random edges after operation (2) and (6) removing random edges after operation (4). The above six operations make the interaction pattern of u' differ from u. Finally, we inject u' and the edges in $E_{u'}$ into the original graph.

The proposed synthetic sample generation method can be combined with down-sampling to tackle the imbalance of the dataset.

4.5 Complexity

Complexity of Representation Module In edge anonymization step, it takes $O(pl')$ time to compute h_{u_i} for a source node u_i, where l' is the number of destination nodes that are visited by u and its partners. Note that $l \le l' \le pl$. It takes $O(pl)$ time to compute h_{v_i} for a destination node v_i. Hence, it takes $O(pl')$ time for the edge anonymization step.

In representation learning step, it takes $O(n_{p_1})$ time to generate masks and output the AER, where n_{p_1} is the number of learnable parameters in the representation module.

Table 2: Summary of dataset statistics.

Type	Real Anomalies			Syn. Anomalies	
Dataset	Mooc	Reddit	Amazon	Enron	Wikitemp
#edge	411,749	672,447	1,554,400	137,758	8,616,454
#anom.	4,066	366	155,300	12,523	783,314
#source	7,047	10,000	769,595	181	251,154
#dest.	98	984	529,279	365	1,120,716
$\#deg_s$	106	73	2	2,171	893
$\#deg_d$	7,229	4,905	4	1,163	91
attr. dim	4	172	0	0	0

Complexity of Prediction Module Assume it takes $O(t_{ru})$ time for a recurrent unit encoder to encode the AER sequence. It takes $O(k \cdot n_{p_2})$ time to compute the k scores through the two-layer fully connected network, where n_{p_2} is the number of parameters in the two-layer network. Hence, the time complexity of the prediction module is $O(t_{ru} + n_{p_2})$.

Putting above together, it takes $O(pl' + t_{ru} + n_p)$ time for AER-AD to make a prediction, where $n_p = n_{p_1} + n_{p_2}$ is the number of parameters in the representation module and in the two-layer network.

5 EXPERIMENTS

In this section, we conduct extensive experiments on five different anomaly detection datasets to demonstrate the effectiveness and efficiency of our model. Our experiments are designed to answer the following questions:

- **RQ1.** How well is AER-AD able to detect anomaly in dynamic bipartite graphs compared to state-of-the-art methods in the inductive setting?
- **RQ2.** How does each component of AER-AD contribute to the final detection performance?
- **RQ3.** Is AER-AD robust to the hyper-parameters and iteration numbers?
- **RQ4.** Is AER-AD efficient in detecting anomalies?

5.1 Datasets

We conduct experiments on five datasets that can be divided into two types: *datasets with real anomalies* and *datasets with synthetic anomalies*. The details of the datasets are reported in Table 2, where *#source*, *#dest*, *#edge* and *#anom.* represent the number of source nodes, destination nodes, edges, and edges with anomalous labels, respectively. $\#deg_s$ represents median of source node degree, $\#deg_d$ represents median of destination node degree, *attr. dim* represents the dimensionality of the additional attributes.

Datasets with real anomalies. We use three datasets with real anomalies. `MOOC`[1] [21] consists of a set of students and a set of courses. An edge pointing from a student to a course corresponds to an action, such as submitting an answer. An edge is labeled as anomaly if it is the last interaction before the student drops out of the course. Each edge is additionally associated with a 4-dimensional feature vector. `Reddit`[1] [21] collects posts on subreddits in one month. The source and destination nodes are users and subreddits,

[1] http://snap.stanford.edu/jodie

Table 3: Comparing the performance of AER-AD in terms of AUC with state of the art algorithms in inductive setting. Bold font† and bold font highlight the best performance among all methods and each type of methods, respectively.

Model	Real Anomalies						Syn. Anomalies			
	Mooc		Reddit		Amazon		Enron		Wikitemp	
	F1	AUC	F1	AUC	F1	AUC	F1	AUC	F1	AUC
Midas	50.09	48.50	49.23	51.62	50.04	50.21	**73.39**	**91.19**	52.16	52.60
F-Fade	50.17	54.24	**49.98**	49.90	50.62	50.97	45.10	13.53	50.23	51.48
AEGIS	62.29	67.42	50.78	**62.99**	n.a.	n.a.	n.a.	n.a.	n.a.	n.a.
IGRL-FI	**68.38**	**81.82**	44.77	59.67	48.61	64.12	31.92	76.01	out	out
IGRL-GS	67.01	74.02	49.88	62.66	**53.76**	**71.50**	40.97	78.68	60.88	85.69
TGAT	36.53	58.62	**38.43**	**62.94**	40.61	53.17	49.17	53.38	48.86	50.21
FI-GRL	64.52	73.26	35.58	62.49	36.68	63.23	56.74	74.68	out	out
Graphsage	72.19	79.32	36.39	61.45	36.18	62.09	57.03	**77.69**	31.29	62.41
CAW	**74.79**	**82.19**	37.92	62.86	**51.59**	**66.78**	**64.95**	77.17	**62.25**	**80.42**
AER-AD	**77.38**†	**83.01**†	**55.31**†	**80.15**†	**59.35**†	**75.38**†	**82.25**	**95.49**	80.51	86.13

respectively. An edge represents a user making a post in a subreddit. An anomalous edge represents the last interaction before the user is banned by Reddit. Each edge is associated with a 172-dimensional feature vector extracted from the context of the post. `Amazon` [1] is a user-product review graph. The label of each edge is determined based on the helpfulness votes. Following previous work [20],we define a review is normal if the fraction of helpful-to-total votes is ≤ 0.75, and anomaly if < 0.25. We extract the reviews with at least 20 votes and users whose reviews is more than three from the original dataset.

Datasets with synthetic anomalies. We use two datasets with synthetic anomalies. `Enron`[2] is an email graph that collects emails exchanged among 151 employees over 3 years in ENRON Corporation. `Wikitemp`[3] [22] is a temporal network representing wikipedia users editing each other's talk page. Since there is no anomaly label for the edges in `Enron` and `Wikitemp`, we follow previous work [2, 5, 49] to inject anomalies into these two datasets by randomly rewiring edges. Specifically, we inject 10% fake edges uniformly at random, the timestamp of an edge is generated within the range between the first and last edge uniformly at random.

5.2 Experimental setting

All the experiments are run on a computer with Intel(R) Core(TM) i7-9700K CPU @3.60 GHz processor, 64 GB RAM and two Geforce RTX 2080Ti graphics cards.

Evaluation and training protocols. Inductive anomaly detection aims to predict whether a newly-arrived edge in the bipartite graph stream is anomalous, where the source node is not observed before. We split the source node into 60%-10%-30% and use their associated edge as the training, validation and testing sets. This guarantees that the source nodes of the edges encountered in the test set are not observed during the training stage. We use binary cross entropy loss and Adam optimizer to train all the models. At training stage, we use down-sampling to balance the labels in `MOOC`, `Amazon`, `Enron` and `Wikitemp`. For `Reddit`, we adopt synthetic labels generation proposed in section 4.3 before down-sampling. Due to the skew of label distribution, we employ the Macro F1 and Area Under the ROC Curve (AUC) as the metrics.

Baselines: We compare our method with nine state-of-the-art methods, including two anomaly detection methods that proposed to detect pre-defined anomalies (Midas [4] and F-Fade[5]), two representation based anomaly detection methods (AEGIS [8] and IGRL [3]) and four inductive graph representation methods (TGAT [47], FI-GRL [13], GraphSage [15] and CAW [44]).

• **Midas** uses a hypothesis testing-based framework to detect arriving groups of suspicious edges (microcluster anomalies).
• **F-Fade** leverages a novel frequency-factorization technique for detection of anomalies in edge streams. This model is able to handle prompt group change and burst of interactions.
• **AEGIS** proposes a construct graph differentiation network (GDN) aggregate information in attributed graph. AEGIS is an unsupervised method that proposed for anomaly detection in static graphs. To fairly comparison, we adapt the AEGIS model by using GD to generate the edge representation and GRU network to make prediction.
• **IGRL** is a fraud detection model based on inductive graph representation. This model apply two promising inductive graph representation learning techniques: GraphSAGE [15] and Fast Inductive Graph Representation Learning [13], referred to as **IGRL-GS** and **IGRL-FI**, respectively.
• **TGAT** leverages GAT to extract node representations where the nodes' neighbors are sampled from the history and encodes temporal information. TGAT recognizes the node embeddings as functions of time and is able to inductively infer embeddings for unseen node.
• **FI-GRL** consists of two stages: decoupling and feature extraction. The first stage is designed for decoupling nodes' relations and the second stage extracts meaningful feature by low rank approximation. FI-GRL obtains representations for seen nodes with provable theoretical guarantees and can easily generalize to unseen nodes.
• **Graphsage** learns node representations by aggregating its neighboring attributes. Graphsage focuses on attribute-rich graphs, this approach can also applied to graphs without additional attribute by making use of node degrees.

[1] http://jmcauley.ucsd.edu/data/amazon/
[2] https://www.kaggle.com/wcukierski/enron-email-dataset
[3] https://snap.stanford.edu/data/wiki-talk-temporal.html

• **CAW** extracts temporal random walks and work as temporal network motifs to represent network dynamic. It encodes node identities with the counts of the nodes based on a set of sampled walks to keep the method inductive.

For all graph representation methods, we first adopt their models to generate the node representations. Then we predict their anomaly score for each edge with GRU network. Noted that some static graph baselines (IGRL, FI-GRL and GraphSage) are not easy to be adapted to support dynamic graphs. Hence, for those baselines, given an edge (u, v, t), we use the whole graph to predict its anomaly score instead the edges occurred before the time t.

5.3 Overall evaluation (RQ1)

Table 3 shows the anomaly detection results of AER-AD and state-of-the-art baselines, where "n.a." represents the method cannot be aplied on this dataset, and "out" represents out of memory.

Midas and F-Fade perform worst in the experiments. This is because Midas and F-Fade focus on specific anomalous types and fail to capture the unknown types of anomalies. We observe that Midas outperforms all baselines on `Enron` dataset. This may because Midas is specially designed to detect the arriving groups of suspicious edges, which is the similar to the manually injected anomalies in `Enron`. Recall that Midas cannot handle unknown anomaly patterns, which explains why Midas performs much worse on the datasets with real anomalies. AEGIS and TGAT heavily depend on attributes and the structural dynamics are not captured essentially. Therefore, they achieve poor performance on graphs without attributes (`Amazaon`, `Enron` and `Wikitemp`) or insufficient attributes (`Mooc`).

IGRL (IGRL-FI and IGRL-GS) achieves the best performance among all anomaly detection baselines. This method leverages inductive graph representation learning techniques to enhance the anomaly detection performance. However, this method relies on existing inductive edge representations, which is not specially designed for anomaly detection. Hence, its performance is still worse than AER-AD.

CAW achieves the best performance among all representation based baselines. Meanwhile, AER-AD is 14.98% and 8.38% higher than CAW on average w.r.t. F1 and AUC, respectively. CAW can successfully capture the structural changing information in the dynamic graph. However, it does not highlight the distinguishable information in the representation and further fails to preserve sufficient information for anomaly detection. Therefore, its performance is much worse than AER-AD.

We draw the following conclusions from the results. (1) Methods designed for detecting pre-defined anomalies fail to detect anomalies in datasets without prior knowledge; (2) Representation methods that rely on additional attributes cannot handle graphs associated with limited or even none attribute; (3) Highlighting distinguishable information in the representation can greatly improve the performance of anomaly detection.

5.4 Effect of each component (RQ2)

Ablation study with AER-AD. We conduct ablation studies by removing AER_s, AER_d, additional attributes (only `MOOC` and `Reddit` have attributes), and masks. Table 4 shows the ablation study results. We observe that the full solution achieves the best performance, which justifies the design of our solution. In addition, we observe that AER_s and AER_d learned from the anonymized description for the source and destination nodes contribute most to the performance, though the additional attributes also helps. Moreover, we observe that incorporating partition mask and association mask improves the performance w.r.t. F1 score.

Comparison with variants. In order to justify the design of the proposed representation module, we compare AER-AD with its variants in Table 5. We first compare the proposed method with *Straightforward* method (naive/straightforward representation method mentioned in Section 4.2.1). The straightforward solution generates two vectors $\hat{AER}_s \in \mathbb{R}^p$ and $\hat{AER}_d \in \mathbb{R}^l$, where the i-th dimension in $\hat{AER}_s$ is the Jaccard similarity between the interaction history of u and its i-th partner u_i, and the j-th dimension in $\hat{AER}_d$ is calculated by Equation (4). We concatenate $\hat{AER}_s$ and $\hat{AER}_d$ and pass it to the prediction module directly. We report the AUC and F1 score of this straightforward solution in the three real-life datasets in Table 5. By comparing its performance with our final solution, we observe that AER-AD outperforms this straightforward solution. This demonstrates the effectiveness of *Edge Anonymization* and *Representation Learning* components in our framework, which also justifies our design.

Next, we evaluate the effect of proposed stochastic mask generation method. In this set of experiments, we use deterministic functions [48] to replace the proposed mask generation module. Table 5 reports the performance with deterministic functions (*deterministic*). We observe that the proposed stochastic method for generating mask improves the F1 score by up to 7.12%.

Finally, we conduct experiments to investigate the effect of using different prediction layer in AER-AD. We consider three prediction operations: *LSTM*, *Transformer*, and *GRU*. As the shown in Table 5, we observe the proposed edge representation method is robust to predict methods, all of the test operations achieve good performances.

Effect of data augmentation. As mentioned in Section 4.4, we generate and inject synthetic samples to address the imbalance of dataset. In this experiment, we use the extremely imbalanced `Reddit` to evaluate the effect of the data augmentation method. After synthetic sample generation, the number of anomalous edges is 1547 (each of the six operations generates 221 anomalous edges). Table 6 reports the AUC w.r.t. different augmentation operations, where "none" represent using the training data without data augmentation. aug1 to aug6 represent using the training data with augmentation type 1 to 6, respectively. "All" represents using all types of data augmentation. We observe each augmentation operation contribute to the performance. With the all augmentation operations, the AUC of the learned model is significantly improved by 39.47% and 24.65%, respectively.

Effect of number of bins in histogram. In the distribution-based method for anonymizing the destination node, we build histogram to encode the popularity of association pairs. We next evluate how the number of bins in the histogram affect the performance. Table 7 reports the results. We observe that our solution is robust to the number of bins in the histogram.

Table 4: Comparing the performance of ablation study.

Model	Real Anomalies						Syn. Anomalies			
	Mooc		Reddit		Amazon		Enron		Wikitemp	
	F1	AUC	F1	AUC	F1	AUC	F1	AUC	F1	AUC
remove AER_s	74.85	78.72	36.66	61.25	48.64	61.36	61.85	80.66	69.62	75.83
remove AER_d	77.29	82.38	53.48	**81.76**	55.41	73.91	80.45	93.31	80.66	86.07
remove attr.	77.99	82.87	50.49	77.56	n.a.	n.a.	n.a.	n.a.	n.a.	n.a.
remove mask	76.95	82.16	53.24	81.63	50.94	73.56	82.14	95.17	80.21	85.47
original	**77.38**	**83.01**	**55.31**	80.15	**59.35**	**75.38**	**82.25**	**95.49**	**80.72**	**86.98**

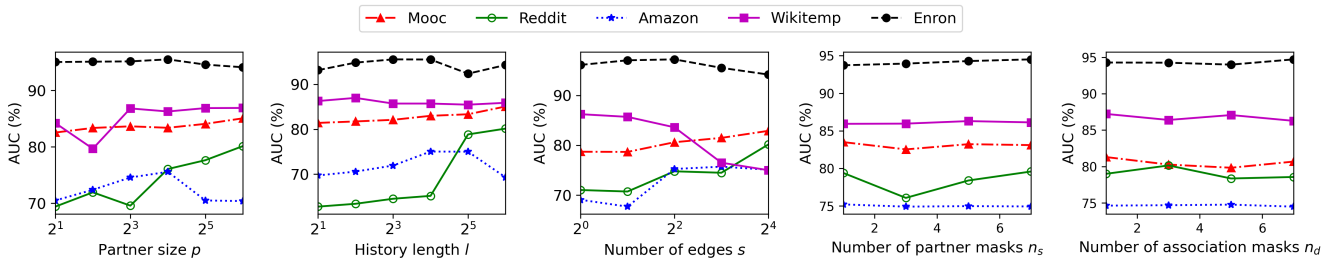


Figure 4: Varying hyper-parameters.

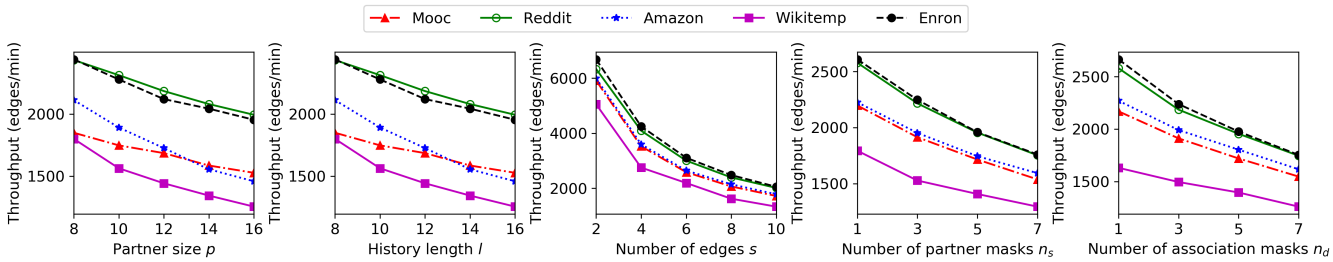


Figure 5: Throughput of whole model w.r.t hyper-parameters.

Table 5: Comparison with variants.

Model	Mooc		Reddit		Amazon	
	F1	AUC	F1	AUC	F1	AUC
Straightforward	78.95	84.52	51.17	60.68	36.11	63.72
Deterministic	75.52	82.39	53.37	**82.30**	53.95	73.19
AER-LSTM	76.97	83.88	55.82	77.44	57.14	74.88
AER-Transformer	78.60	84.59	44.82	73.78	48.11	74.52
AER-AD	**77.38**	**83.0**	**55.31**	80.15	**59.35**	**75.38**

Table 6: Effect of data augmentation.

	none	aug1	aug2	aug3	aug4	aug5	aug6	all
F1	42.41	50.41	49.19	45.21	48.52	46.82	44.65	55.31
AUC	75.47	80.00	79.28	76.42	78.86	80.70	78.85	80.15

Table 7: Effect of the number of bins in histogram.

Number	5	10	15	20
F1	55.32	59.35	59.34	59.34
AUC	65.30	75.38	74.76	75.24

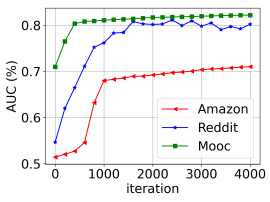


Figure 6: AUC w.r.t number of iterations

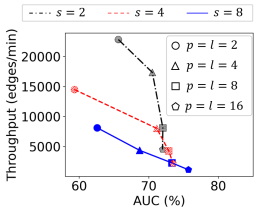
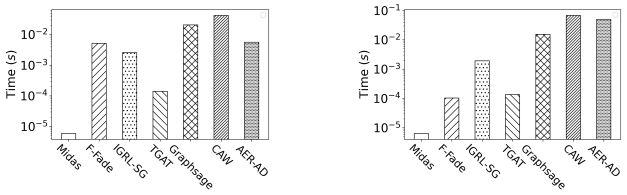


Figure 7: AUC w.r.t throughput.

5.5 Robustness (RQ3)

Hyper-parameter sensitivity. We analyze the effect of the hyper-parameters, including the partner size s, the history length d, the edge numbers l source masks numbers n_s and destination masks numbers n_d on the performance of the proposed model. When



(a) Wikitemp (b) Amazon

Figure 8: Average running time per edge.

Table 8: Effect of sampling methods in Amazon dataset. +(−)x% represent improve (decreae) x inference ratio compared with the model with the partner size ($p = 16$) that generate best performance.

		$p = 2$	$p = 3$	$p = 4$
Truncate	TP	**2258** (+95%)	**2109** (+82%)	**1942** (+68%)
	AUC	70.51 (-6.72%)	70.71 (-6.46%)	72.40 (-4.22%)
Degree	TP	2120 (+83%)	2033 (+76%)	1941 (+68%)
	AUC	**71.79** (-5.03%)	71.54 (-5.36%)	**73.47** (-2.80%)
TimeSpan	TP	2111 (+82%)	2027 (+75%)	1932 (+67%)
	AUC	71.64 (-5.23%)	**71.57** (-5.32%)	73.44 (-2.84%)

investigating each hyper-parameter, we set the rest ones to an optimal value found by grid search, and report the AUC performances on all test sets. As shown in Figure 4, AER-AD achieves the best performance when s, d and l are small. For most datasets, AUC achieves the best performance when $s < 16$, $d < 16$ and $l < 4$. For dense graphs (`Mooc`,`Reddit`,`Wikitemp` and `Enron`), AUC become saturate when $s > 16$ and $l > 16$. For sparse graph (`Mooc`), AUC first increases as s and l increases, and the AUC decreases when s and l greater than a threshold. We also analyze the effect of mask number in Figure 4, we observe the model is robust to the number of masks w.r.t AUC.

Number of iterations. Figure 6 shows the test AUC of AER-AD w.r.t the number of iterations. We observe the AUC become stable to iterations number after 1000 iterations.

5.6 Efficiency (RQ4)

Comparison with baselines. Figure 8 shows the efficiency of our method w.r.t all baselines (exclude IGRL-FI and FI-GRL). IGRL-FI and FI-GRL are matrix factorization based methods, they require a high complexity and cannot provide a timely detection. Both of IGRL-FI and FI-GRL take 37.2s to detect an edge in Amazon and out of memory when handle Wikitemp. We observe Midas and TGAT are more efficient than AER-AD. However, their AUC and F1 are much worse than AER-AD. AER-AD are more efficient than CAW and are comparable with F-Fade, IGRL-SG and Graphsage.

Throughput w.r.t hyper-parameters. Figure 5 shows the throughput of our proposed solution by varying hyper-parameters. We observe that the throughput of our solution is sublinear to all the hyperparameters. This is because the bottleneck of the solution is the edge anonymization, which involves frequent retrieval of partners, histories and set union/intersection operations. When generating the AERs for two consecutive edges, there are many redundant computations. In our implementation, we have optimized to avoid redundant computations and we anonymize the edges in a collective manner. Thus, when the hyperparameters increase, the throughput degrades sublinearly.

Tradeoff between AUC and throughput. In addition, we have conducted experiments to investigate the tradeoff between accuracy and throughput on Amazon dataset. Specifically, we set s (the length of AER sequence) as 2, 4, and 8, respectively. By varying the partner size and history length from 2 to 16, we report the tradeoff between throughput and AUC in Figure 7. We observe that the throughput decreases as AUC increases at first. When AUC reaches about 72%–75%, the throughput degrades drastically. It is difficult to improve AUC after AUC reaches 75%.

Effect of sampling methods. In section 4.2.1, we propose a sampling method to improve the performance of the proposed method with small partner size. As shown in Figure 4, Amazon achieve best performance when partner size $p = 16$ and its AUC is decrease 6.72% inference ratio when $p = 2$. Hence, we use Amazon dataset to test our sampling method. Table 8 show the throughput (TP) and AUC w.r.t different sampling method. We can draw the following conclusions from Table 8: (1) AER-AD can achieve good performance when the partner small; (2) Compared with the best performance hyper-parameters, smaller partner size will sacrifice accuracy and improve throughput; (3) Compared with deterministic method, the proposed sampling methods can improve the AUC at a small partner size.

Memory, CPU and GPU usage. We use a Python package **Psutil**[4] to keep track of the memory and CPU usage and **GPUtil**[5] to collect the GPU usage. The maximum memory usage of the proposed method is 9.04 GB. The total CPU utilization is 91.6% (Note that the CPU has 8 cores and full CPU utilization is 800%). The GPU utilization rate is 8%.

6 CONCLUSION

Inductive anomaly detection is a challenging but critical task nowadays. To address this problem, we propose an effective method to learn anonymous edge representation, which captures the characteristics of an edge without using its identity. AER enables us to represent edges while preserving sufficient information to discover anomalies. Based on AER, we design a model named AER-AD for inductive anomaly detection in dynamic bipartite graphs. The experiments demonstrate that AER-AD is computationally efficient, robust in performance and outperforms state-of-the-art methods significantly.

ACKNOWLEDGMENTS

This work was supported by the National Natural Science Foundation of China under Grant 61906039, Grant 62202124 and Grant 62172090, Youth Scholar Program of SEU, the Fundamental Research Funds for the Central Universities under Grant 2242022k30007, CAAI-Huawei MindSpore Open Fund and Alibaba Group through Alibaba Innovative Research Program.

[4]https: //pypi.org/project/psutil/
[5]https://pypi.org/project/GPUtil/

REFERENCES

[1] C. C. Aggarwal, Y. Zhao, and S. Y. Philip. Outlier detection in graph streams. In *Proceedings of the IEEE 27th international conference on data engineering*, pages 399–409, 2011.

[2] L. Akoglu, H. Tong, and D. Koutra. Graph based anomaly detection and description: a survey. *Data mining and knowledge discovery*, 29(3):626–688, 2015.

[3] R. V. Belle, C. V. Damme, H. Tytgat, and J. D. Weerdt. Inductive graph representation learning for fraud detection. *Expert Syst. Appl.*, 193:116463, 2022.

[4] S. Bhatia, B. Hooi, M. Yoon, K. Shin, and C. Faloutsos. Midas: Microcluster-based detector of anomalies in edge streams. In *Proceedings of the AAAI Conference on Artificial Intelligence*, pages 3242–3249, 2020.

[5] 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*, pages 589–597, 2021.

[6] Z. Chen and A. Sun. Anomaly detection on dynamic bipartite graph with burstiness. In *IEEE International Conference on Data Mining*, pages 966–971, 2020.

[7] K. Cho, B. Van Merriënboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio. Learning phrase representations using rnn encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078*, 2014.

[8] K. Ding, J. Li, N. Agarwal, and H. Liu. Inductive anomaly detection on attributed networks. In *Proceedings of the Twenty-Ninth International Conference on International Joint Conferences on Artificial Intelligence*, pages 1288–1294, 2021.

[9] M. Eslami, G. Zheng, H. Eramian, and G. Levchuk. Anomaly detection on bipartite graphs for cyber situational awareness and threat detection. In *IEEE International Conference on Big Data*, pages 4741–4743, 2017.

[10] D. Eswaran and C. Faloutsos. Sedanspot: Detecting anomalies in edge streams. In *IEEE International Conference on Data Mining*, pages 953–958, 2018.

[11] D. Eswaran, C. Faloutsos, S. Guha, and N. Mishra. Spotlight: Detecting anomalies in streaming graphs. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 1378–1386, 2018.

[12] Y. Fan, Y. Ye, Q. Peng, J. Zhang, Y. Zhang, X. Xiao, C. Shi, Q. Xiong, F. Shao, and L. Zhao. Metagraph aggregated heterogeneous graph neural network for illicit traded product identification in underground market. In *IEEE International Conference on Data Mining*, pages 132–141, 2020.

[13] J. Fei, Z. Lei, X. Jin, and Y. Philip. Fi-grl: Fast inductive graph representation learning via projection-cost preservation. In *IEEE International Conference on Data Mining*, pages 1067–1072. IEEE, 2018.

[14] P. Goyal, S. R. Chhetri, and A. Canedo. dyngraph2vec: Capturing network dynamics using dynamic graph representation learning. *Knowledge-Based Systems*, 187:104816, 2020.

[15] W. L. Hamilton, R. Ying, and J. Leskovec. Inductive representation learning on large graphs. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, pages 1025–1035, 2017.

[16] A. Hasanzadeh, E. Hajiramezanali, K. Narayanan, N. Duffield, M. Zhou, and X. Qian. Variational graph recurrent neural networks. *Advances in Neural Information Processing Systems*, 32, 2019.

[17] B. Hooi, H. A. Song, A. Beutel, N. Shah, K. Shin, and C. Faloutsos. Fraudar: Bounding graph fraud in the face of camouflage. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 895–904, 2016.

[18] E. Jang, S. Gu, and B. Poole. Categorical reparameterization with gumbel-softmax. *arXiv preprint arXiv:1611.01144*, 2016.

[19] T. N. Kipf and M. Welling. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907*, 2016.

[20] S. Kumar, B. Hooi, D. Makhija, M. Kumar, C. Faloutsos, and V. Subrahmanian. Rev2: Fraudulent user prediction in rating platforms. In *Proceedings of the Eleventh ACM International Conference on Web Search and Data Mining*, pages 333–341. ACM, 2018.

[21] S. Kumar, X. Zhang, and J. Leskovec. Predicting dynamic embedding trajectory in temporal interaction networks. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 1269–1278, 2019.

[22] J. Kunegis. Konect: the koblenz network collection. In *Proceedings of the 22nd international conference on World Wide Web*, pages 1343–1350, 2013.

[23] B. Le Bars and A. Kalogeratos. A probabilistic framework to node-level anomaly detection in communication networks. In *IEEE INFOCOM 2019-IEEE Conference on Computer Communications*, pages 2188–2196, 2019.

[24] W. Lei, X. He, M. de Rijke, and T.-S. Chua. Conversational recommendation: Formulation, methods, and evaluation. In *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 2425–2428, 2020.

[25] R. Li, P. Wang, P. Jia, X. Zhang, J. Zhao, J. Tao, Y. Yuan, and X. Guan. Approximately counting butterflies in large bipartite graph streams. *IEEE Transactions on Knowledge and Data Engineering*, 2021.

[26] C. Luo and A. Shrivastava. Arrays of (locality-sensitive) count estimators (ace) anomaly detection on the edge. In *The World Wide Web Conference*, pages 1439–1448, 2018.

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

[28] C. Meng, S. C. Mouli, B. Ribeiro, and J. Neville. Subgraph pattern neural networks for high-order graph evolution prediction. In *Proceedings of the AAAI Conference on Artificial Intelligence*, pages 3778–3787, 2018.

[29] V. Miz, B. Ricaud, K. Benzi, and P. Vandergheynst. Anomaly detection in the dynamics of web and social networks using associative memory. In *The World Wide Web Conference*, pages 1290–1299, 2019.

[30] N. Noorshams, S. Verma, and A. Hofleitner. Ties: Temporal interaction embeddings for enhancing social media integrity at facebook. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 3128–3135, 2020.

[31] G. Pang, L. Cao, and C. Aggarwal. Deep learning for anomaly detection: challenges, methods, and opportunities. In *Proceedings of the 14th ACM International Conference on Web Search and Data Mining*, pages 1127–1130, 2021.

[32] A. Pareja, G. Domeniconi, J. Chen, T. Ma, T. Suzumura, H. Kanezashi, T. Kaler, T. Schardl, and C. Leiserson. Evolvegcn: Evolving graph convolutional networks for dynamic graphs. In *Proceedings of the AAAI Conference on Artificial Intelligence*, pages 5363–5370, 2020.

[33] L. Peel and A. Clauset. Detecting change points in the large-scale structure of evolving networks. In *Proceedings of the 29th AAAI Conference on Artificial Intelligence*, pages 2914–2920, 2015.

[34] Y. Ren, H. Zhu, J. Zhang, P. Dai, and L. Bo. Ensemfdet: An ensemble approach to fraud detection based on bipartite graph. In *IEEE 37th International Conference on Data Engineering*, pages 2039–2044, 2021.

[35] A. Sankar, Y. Wu, L. Gou, W. Zhang, and H. Yang. Dysat: Deep neural representation learning on dynamic graphs via self-attention networks. In *Proceedings of the 13th International Conference on Web Search and Data Mining*, pages 519–527, 2020.

[36] K. Shin, B. Hooi, J. Kim, and C. Faloutsos. Densealert: Incremental dense-subtensor detection in tensor streams. In *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 1057–1066, 2017.

[37] J. Sun, D. Tao, and C. Faloutsos. Beyond streams and graphs: dynamic tensor analysis. In *Proceedings of the 12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 374–383, 2006.

[38] J. Sun, Y. Zhang, W. Guo, H. Guo, R. Tang, X. He, C. Ma, and M. Coates. Neighbor interaction aware graph convolution networks for recommendation. In *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 1289–1298, 2020.

[39] X. Teng, Y. R. Lin, and X. Wen. Anomaly detection in dynamic networks using multi-view time-series hypersphere learning. In *The Conference on Information and Knowledge Management*, pages 827–836, 2017.

[40] R. Trivedi, M. Farajtabar, P. Biswal, and H. Zha. Dyrep: Learning representations over dynamic graphs. In *International Conference on Learning Representations*, 2019.

[41] D. Wang, J. Lin, P. Cui, Q. Jia, Z. Wang, Y. Fang, Q. Yu, J. Zhou, S. Yang, and Y. Qi. A semi-supervised graph attentive network for financial fraud detection. In *IEEE International Conference on Data Mining*, pages 598–607, 2019.

[42] T. Wang, C. Fang, D. Lin, and S. F. Wu. Localizing temporal anomalies in large evolving graphs. In *Proceedings of the 2015 SIAM International Conference on Data Mining*, pages 927–935, 2015.

[43] X. Wang, D. Lyu, M. Li, Y. Xia, Q. Yang, X. Wang, X. Wang, P. Cui, Y. Yang, B. Sun, et al. Apan: Asynchronous propagation attention network for real-time temporal graph embedding. In *Proceedings of the 2021 International Conference on Management of Data*, pages 2628–2638, 2021.

[44] Y. Wang, Y. Y. Chang, Y. Liu, J. Leskovec, and P. Li. Inductive representation learning in temporal networks via causal anonymous walks. *International Conference on Learning Representations*, 2021.

[45] Y. Wang, J. Zhang, S. Guo, H. Yin, C. Li, and H. Chen. Decoupling representation learning and classification for gnn-based anomaly detection. In *Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 1239–1248, 2021.

[46] W. Xia, Y. Li, J. Wu, and S. Li. DeepIS: Susceptibility estimation on social networks. In *Proceedings of the 14th ACM International Conference on Web Search and Data Mining*, pages 761–769, 2021.

[47] D. Xu, C. Ruan, E. Korpeoglu, S. Kumar, and K. Achan. Inductive representation learning on temporal graphs. In *Proceedings of the International Conference on Learning Representations*, 2020.

[48] M. Yang, Y. Shen, H. Qi, and B. Yin. Soft-mask: Adaptive substructure extractions for graph neural networks. In *Proceedings of the Web Conference 2021*, pages 2058–2068, 2021.

[49] M. Yoon, B. Hooi, K. Shin, and C. Faloutsos. 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*, pages 647–657, 2019.

[50] W. Yu, C. C. Aggarwal, S. Ma, and H. Wang. On anomalous hotspot discovery in graph streams. In *IEEE 13th International Conference on Data Mining*, pages 1271–1276, 2013.

[51] W. Yu, C. C. Aggarwal, and W. Wang. Temporally factorized network modeling for evolutionary network analysis. In *Proceedings of the 10th ACM International Conference on Web Search and Data Mining*, pages 455–464, 2017.

[52] 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*, pages 2672–2681, 2018.

[53] D. Zambon, C. Alippi, and L. Livi. Concept drift and anomaly detection in graph streams. *IEEE transactions on neural networks and learning systems*, 29(11):5592–5605, 2018.

[54] Z. Zhang, J. Bu, M. Ester, J. Zhang, C. Yao, Z. Li, and C. Wang. Learning temporal interaction graph embedding via coupled memory networks. In *The Web Conference*, pages 3049–3055, 2020.

[55] L. Zheng, Z. Li, J. Li, Z. Li, and J. Gao. Addgraph: Anomaly detection in dynamic graph using attention-based temporal gcn. In *International Joint Conference on Artificial Intelligence*, pages 4419–4425, 2019.

[56] L. Zhou, Y. Yang, X. Ren, F. Wu, and Y. Zhuang. Dynamic network embedding by modeling triadic closure process. In *Proceedings of the AAAI Conference on Artificial Intelligence*, pages 571–578, 2018.