Time-Aware Graph Learning for Link Prediction on Temporal Networks

Zhiqiang Pan, Honghui Chen, Wanyu Chen, Fei Cai, and Xinwang Liu, *Senior Member, IEEE*

***Abstract*—Link prediction on temporal networks aims to predict the future edges by modeling the dynamic evolution involved in the graph data. Previous methods relying on the node/edge attributes or the distance on the graph structure are not practical due to the deficiency of the attributes and the limitation of the explicit distance estimation, respectively. Moreover, the existing graph representation learning methods mostly rely on graph neural networks (GNNs), which cannot adequately take the dynamic correlations between nodes into consideration, leading to the generating of inferior node embeddings. Thus, we propose a time-aware graph (TAG) learning method for link prediction on temporal networks. We first conduct a theoretical causal analysis proving that the correlations between nodes are required to be unchanged for the temporal graph representation learning using GNNs. Then, we model the recent dynamic node correlations by designing an edge-dropping (ED) module and adopting a recent neighbor sampling (RNS) strategy so as to approximate the above condition. Besides, we also preserve the long-term stable node correlations by introducing additional self-supervisions using the contrastive learning. Comprehensive experiments were conducted on four public temporal network datasets, i.e., MathOverflow, StackOverflow, AskUbuntu, and SuperUser, demonstrate that TAG can achieve state-of-the-art performance in terms of average precision (AP) and area under the ROC curve (AUC). In addition, TAG can ensure high computational efficiency by making the temporal graph lightweight, letting it be practical in real-world applications.**

***Index Terms*—Causal colliding, contrastive learning, edge dropping (ED), link prediction, recent sampling, temporal networks.**

I. Introduction

GRAPH networks are an effective way to model complex systems by representing the elements as nodes and their interactions as edges [1], [2], [3], [4], [5], [6], which are widely applied in current researches regarding recommender systems [7], [8], [9], [10] and social networks [11], [12]. Moreover, the network data usually keeps evolving over time in the realistic scenarios [13], [14], [15], [16], [17]. As an example shown in Fig. 1, the number of nodes and their interactions generally increase over time. The link prediction task on temporal networks aims to learn the temporal evolution from the existing dynamic graph data and then predict the links that may appear in the future [18], [19]. Compared to that on the static graphs, the link prediction on temporal networks is more challenging due to the constantly appearing new nodes as well as edges and the evolving temporal node correlations. In addition, as explained in Fig. 1, temporal link prediction includes predicting transductive links connecting two seen nodes (i.e., nodes already observed) and the more challenging inductive links associated with unseen nodes (i.e., nodes to appear in the future), respectively.

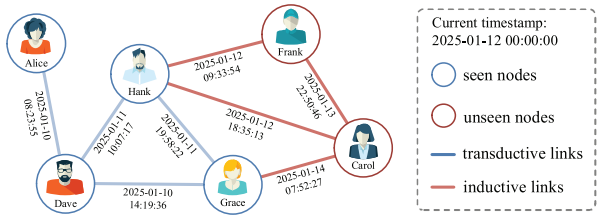

Fig. 1. Example of the temporal network in the social network scenario, where each node denotes a user and each temporal edge indicates that two users become friends at the corresponding timestamp marked using the number below each edge. Moreover, users appearing before and after the current timestamp can be divided into seen nodes and unseen nodes, respectively. In addition, transductive links and inductive links indicate links connecting two seen nodes and links associated with unseen nodes, respectively.

Existing methods mainly propagate the time dynamics together with the node/edge features to learn the node representations for making predictions [20], [21], [22], [23], [24]. Moreover, the explicit distance between nodes on the temporal topological structure is also taken into consideration, which is measured by regarding the common neighbors of the node pair as their intermediate nodes [18], [25], [26]. In addition, the embedding-based methods adopt the temporal topological structures to map the nodes into the embedding space where the embedding vectors of nodes can be learned, which can measure the general distance between nodes without limitation of the explicit structures [20], [27], [28].

Although the aforementioned methods have achieved satisfactory performance, there still remain several problems. First,

Received 10 January 2024; revised 14 November 2024 and 23 January 2025; accepted 18 February 2025. This work was supported in part by the National Natural Science Foundation of China under Grant 62372458 and Grant 62302511, in part by the Independent Innovation Research Fund of National University of Defense Technology under Grant 23-ZZCX-JDZ-43, in part by the Basic Strengthening Program (Young Elite Scientists Sponsorship Program) under Grant 2023-JCJQ-QT-048, and in part by the Postgraduate Scientific Research Innovation Project of Hunan Province under Grant CX20230083. *(Corresponding authors: Wanyu Chen; Fei Cai.)*

Zhiqiang Pan, Honghui Chen, and Fei Cai are with the Science and Technology on Information Systems Engineering Laboratory, National University of Defense Technology, Changsha 410073, China (e-mail: panzhiqiang@nudt.edu.cn; chenhonghui@nudt.edu.cn; caifei08@nudt.edu.cn).

Wanyu Chen is with the College of Electronic Countermeasures, National University of Defense Technology, Hefei 230037, China (e-mail: wanyuchen@nudt.edu.cn).

Xinwang Liu is with the School of Computer, National University of Defense Technology, Changsha 410073, China (e-mail: xinwangliu@nudt.edu.cn).

Digital Object Identifier 10.1109/TNNLS.2025.3545021

the required features in the dynamic feature-aware methods are usually unavailable in many real-life scenarios, which limits the application scope of such methods. Second, the approaches using distance encoding fail to consider the potential distance between the loosely connected nodes, and the complex encoding process leads to inefficient training as well as inference. Moreover, the graph neural networks (GNNs) in the existing embedding-based methods cannot fully take the time dynamics in the temporal networks into consideration, easily introducing bias from the outdated temporal graph data.

In order to solve the above problems, we propose a time-aware graph (TAG) learning approach by improving the embedding-based methods. Specifically, we first conduct a theoretical analysis of the temporal graph representation learning using the causality theory, proving that the graph learning on temporal networks requires the condition of unchanged correlations between nodes. Then, for approximating this condition on temporal networks by modeling the recent dynamic correlations between nodes, we design an edge-dropping (ED) module to abandon the outdated training data in the model parameters learning, where a recent neighbor sampling (RNS) strategy is also adopted to select the neighbors for information aggregation in GNNs. Besides, to preserve the long-term stable node correlations, we incorporate additional self-supervisions to enhance the representation learning by contrasting the node representations generated by propagating information from the long-term and recent neighbors, respectively.

Extensive experiments are conducted on four public temporal network datasets, i.e., MathOverflow, StackOverflow, AskUbuntu as well as SuperUser, and the experimental results validate the effectiveness of TAG in terms of average precision (AP) and area under the ROC curve (AUC) on the link prediction task on temporal networks.

The main contributions of this article can be summarized as follows.

1) We conduct a theoretical analysis of the temporal graph learning with the causality theory, which concludes the condition needed for using GNNs to effectively model the dynamic evolution of temporal networks, i.e., the recent dynamic correlations between nodes should be emphasized.
2) For approximating the above condition, we design an ED module and adopt an RNS strategy for the graph representation learning so as to take the recent dynamic node correlations into consideration.
3) We propose to contrast the node representations aggregating the long-term and recent neighbor information, respectively, which can preserve the long-term stable node correlations and thus enhance the node representation learning.
4) Comprehensive experiments are conducted to validate the effectiveness of TAG by proving the superiority of TAG above the state-of-the-art baselines in terms of AP and AUC on four publicly available temporal network datasets.

II. Related Work

In this section, we first review the related works for link prediction on temporal networks mainly from two categories following [25], [29], i.e., the discrete-time and continuous-time dynamic graph learning. In addition, we summarize the previous methods of contrastive learning for link prediction which are also highly related to our work.

A. Dynamic Graph Learning

1) Discrete-Time Dynamic Graph Learning: Discrete-time dynamic graph learning focuses on the node representation learning on the discrete-time temporal networks, where the dynamic graph data is represented using a sequence of graph snapshots [20], [22], [30], [31], [32], [33], [34]. Specifically, a discrete-time dynamic graph can be represented as $\mathcal{G} = \{\mathcal{G}^1, \ldots, \mathcal{G}^T\}$, which consists of T static graph snapshots, where T is the limited time step of the dynamic graph. Each graph snapshot $\mathcal{G}^i = \{\mathcal{V}^i, \mathcal{E}^i\}$ is a static graph consisting with the corresponding nodes $\mathcal{V}^i$ and edges $\mathcal{E}^i$ in the ith snapshot. Discrete-time temporal link prediction aims to predict links in the future graph snapshots from historical snapshots.

For example, Zhou et al. [30] propose to simultaneously take the structure as well as the evolution of the network into consideration by modeling the development of the triads in temporal networks. Then, Singer et al. [31] incorporate the temporal structure evolution into the node representation learning by an initialization using the static node embeddings with further alignment to various timestages. Moreover, Hajiramezanali et al. [22] propose to incorporate the latent random variables into the graph recurrent neural networks for modeling the dynamics of the topology and the node attributes. Furthermore, Pareja et al. [20] adopt the graph convolution networks (GCNs) to learn the node embeddings, where the parameters of GCNs in different time steps are regulated using a recurrent neural network. Yang et al. [32] propose to further take the implicitly hierarchical properties of nodes into consideration by mapping the temporal network into a hyperbolic space. Moreover, Zheng et al. [33] design an incremental computation of the node embeddings with edges arriving in the temporal networks. In addition, Gong et al. [34] design an unsupervised approach which maintains dynamic topology by capturing the neighborhood features and models the network evolution using the long short-term memory (LSTM) network layers.

However, the discrete-time dynamic graph learning methods can merely model the temporal networks consisting of graph snapshots at different timestages, failing to deal with the continuously streaming edges and make instant predictions, which are more common in real-world applications [25], [35].

2) Continuous-Time Dynamic Graph Learning: Continuous-time dynamic graph learning aims to learn accurate node representations on the continuous-time temporal networks, where the time-evolving edges are stored in a continuous way [18], [23], [25], [27], [35], [36], [37], [38], [39]. Specifically, a continuous-time dynamic graph can be denoted as $\mathcal{G} = \{\mathcal{V}, \mathcal{E}\}$, where the contained nodes $\mathcal{V}$ and edges $\mathcal{E}$ continually increase. Moreover, the edge set $\mathcal{E}$ is constituted of continuous edges $\{(v_s^1, v_t^1, t_{st}^1), (v_s^2, v_t^2, t_{st}^2), \ldots\}$, where each

TABLE I
COMPARISON OF THE REPRESENTATIVE APPROACHES FOR LINK PREDICTION ON TEMPORAL NETWORKS

Model	Continuous	Time	Structure	Embedding
[30], [31], [22], [20] [32], [33], [34]	✗	✗	✗	✗
[36]	✓	✗	✗	✗
[23], [37], [38]	✓	✓	✗	✗
[18], [25], [39]	✓	✓	✓	✗
[35], [27]	✓	✗	✗	✓
TAG	✓	✓	✓	✓

edge (v_s, v_t, t_{st}) connects two nodes v_s and v_t at the timestamp t_{st}. The aim of continuous-time temporal link prediction is to forecast future links by learning from the historical continuous edges.

For example, Nguyen et al. [35] generalize the random walk [40] to the temporal scenarios for considering the time dependencies between nodes in the node embedding learning. Then, Kumar et al. [27] design a coupled recurrent neural network to learn the dynamic embeddings of nodes in the temporal interaction networks and predict the future trajectory of the node embeddings afterward. Trivedi et al. [36] propose a two-time scale deep temporal point process model which performs representation learning on temporal networks as a latent mediation process expressed by the association and communication processes. Moreover, Xu et al. [23] propose to simultaneously take the time-varying topological structures as well as the node/edge features into consideration using the self-attention networks [41]. Furthermore, the explicit distance between nodes on the temporal graph structure is incorporated using the temporal anonymous walk [18], [25]. Yang et al. [37] introduce a meta-learner with hierarchical adaptions of the time interval level and node level to extract the general knowledge for representation learning in the few-shot scenarios. Moreover, Cong et al. [38] propose a simple architecture GraphMixer which yields a fast convergence speed and well generalization ability on the temporal link prediction task. In addition, Yu et al. [39] propose to model the node correlations with a neighbor co-occurrence encoding mechanism and consider long neighbor sequences using a patching strategy.

However, most existing continuous-time dynamic graph learning methods either rely on the node/edge features [23], [37], [38] or the distance on the temporal structure [18], [25], which are not practical in many realistic scenarios due to the deficiency of features and the limitation of the explicit distance computation, respectively. Moreover, the embedding-based methods aforementioned [27], [35] fail to adequately take the complex time dependencies involved in the node correlations on the temporal networks into consideration, which limits the model performance.

Different from the above methods, our work focuses on the node representation learning for link prediction on temporal networks, which emphasizes the recent dynamic correlations between nodes using an ED module as well as a recent sampling strategy, and further preserves the long-term stable node correlations through the contrastive learning.

In addition, to provide an intuitive comparison of the representative approaches for temporal link prediction reviewed in Sections II-A1 and II-A2, we compare the advantages and disadvantages of them as well as our proposed TAG model in Table I. In Table I, "Continuous" denotes whether each model can process the continuous-time temporal networks, and "Time," "Structure," and "Embedding" indicate that whether the continuous time encoding, the explicit structure encoding, and the temporal node embedding are implemented on the continuous-time temporal networks, respectively. The compared models are first categorized to the discrete-time methods [20], [22], [30], [31], [32], [33], [34] and the continuous-time methods [18], [23], [25], [27], [35], [36], [37], [38], [39], and the continuous-time methods are further divided into the temporal point process-based method [36], the temporal feature aware methods [23], [37], [38], the explicit structure enhanced methods [18], [25], [39], the embedding-based methods [27], [35], and our proposed TAG model.

B. Contrastive Learning for Link Prediction

Contrastive learning, which aims to learn similar representations for positive sample pairs while distinguishing those of negative sample pairs, has been widely applied in various applications such as computer vision [42] and natural language processing [43]. Recently, researchers have also applied contrastive learning on graphs for making accurate link predictions [44], [45]. For example, Wang et al. [44] utilize self-supervised learning to capture both global and local information in the context subgraph, so as to learn high-order semantic associations between nodes for conducting link predictions on heterogeneous networks. Moreover, Zhang et al. [45] incorporate an auxiliary contextualized self-supervised learning task to exploit the structural contexts brought by context nodes and context subgraphs for boosting accurate link predictions.

In addition, contrastive learning has also been introduced in recent research on dynamic graph learning for making temporal link predictions [46], [47], [48]. For instance, Tian et al. [46] propose to apply self-supervised learning for comparing nodes of two different temporal views and design a debiased contrastive loss for eliminating bias introduced in the negative sampling process. Moreover, Chen et al. [47] design a temporal subgraph sampling method which simultaneously models both structural and temporal information of neighbors, where the sampled subgraph is further summarized by considering the influence of neighbors on the central node. In addition, Liu et al. [48] propose to emphasize the sharp shifts in the node evolution pattern, which disentangles the temporal shift embeddings from temporal consistency embeddings by considering both perspectives of local and global connectivity using self-supervised learning.

Different from the above-mentioned methods, in our work under the condition of accurately learning the dynamic network evolution guided by causality theory, we utilize contrastive learning to contrast the node representations, respectively, aggregating long-term and recent neighbor

TABLE II
MAIN NOTATIONS USED IN THIS PAPER

Notation	Description
$\mathcal{G} = \{\mathcal{V}, \mathcal{E}\}$	the temporal network with the nodes $\mathcal{V}$ and edges $\mathcal{E}$
$\mathcal{V}_S$, $\mathcal{V}_U$	the seen and unseen nodes in $\mathcal{G}$
$\mathcal{E}_T$, $\mathcal{E}_I$	the transductive and inductive edges in $\mathcal{G}$
N_{v_x}	the neighbors of v_x in $\mathcal{G}$
$\mathbf{a}_{v_x}^k$	the information from v_x's neighbors at the k-layer GNN
$\mathbf{h}_{v_x}^k$	the final representations of v_x at the k-layer GNN
$\mathbf{e}_{v_i}$	the embeddings of v_i generated by the embedding layer
$\mathbf{E}_{Ac,t}$	the embeddings of the active nodes at timestamp t
$\mathbf{E}_{In,t}$	the embeddings of the inactive nodes at timestamp t
$\mathbf{I}_t$	the new links generated between timestamps $t-1$ and t
$\mathbf{C}_t$	the correlations between $\mathbf{E}_{Ac,t}$ and $\mathbf{E}_{In,t}$ at timestamp t
$\hat{\mathbf{h}}_{v_x}$	the dynamic representations of v_x generated by GNNs
$\tilde{\mathbf{h}}_{v_x}$	the stable representations of v_x generated by GNNs
$\mathbf{y}_{st}$, $\mathbf{y}_{sn}$	the prediction score of the positive and negative samples
L, L_{main}, L_{cl}	the final, main and contrastive training loss
α	the parameter controlling the edge dropping intensity
β	the parameter controlling the neighbor sampling intensity
ρ	the parameter of the training data selection percentage
τ	the temperature parameter in the contrastive learning
λ	the parameter balancing the main and contrastive loss

information, thus ensuring the preservation of long-term stable correlations between nodes.

III. Approach

A. Task Formulation

Given a temporal network denoted as $\mathcal{G} = \{\mathcal{V}, \mathcal{E}\}$, where $\mathcal{V}$ and $\mathcal{E}$ are the node-set as well as the edge set, respectively, and both $\mathcal{V}$ and $\mathcal{E}$ are time-evolving. Each edge $(v_s, v_t, t_{st}) \in \mathcal{E}$ indicates that a temporal edge connects the source node v_s and the target node v_t at the timestamp t_{st}. Given the current timestamp, we denote the nodes appearing before it as the seen nodes $\mathcal{V}_S$, and the nodes not appearing until the current timestamp as the unseen nodes $\mathcal{V}_U$, respectively. Correspondingly, the edges connecting two seen nodes are indicated as the transductive edges $\mathcal{E}_T = \{(v_s, v_t, t_{st}) | v_s, v_t \in \mathcal{V}_S\}$, and the edges associated with the unseen nodes are denoted as the inductive edges $\mathcal{E}_I = \{(v_s, v_t, t_{st}) | v_s \in \mathcal{V}_U \| v_t \in \mathcal{V}_U\}$. The link prediction task on temporal networks aims to predict the links including the transductive and inductive edges that may appear in the future.

The main notations used in this article are listed in Table II.

B. Causal Analysis on GNNs

Existing embedding-based methods for link prediction on temporal networks generally adopt GNNs to learn accurate representations of nodes. However, learning with GNNs needs to keep the correlations between nodes unchanged, which cannot be well maintained on the temporal networks. Thus, here we first conduct a causal analysis on GNNs by following [49], [50] to identify the key problems which need to be addressed for adopting GNNs to learn the node representations on temporal networks.

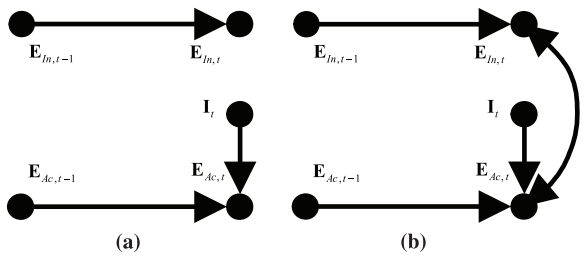

Fig. 2. Causal graphs for the graph representation learning on temporal networks. (a) Causal graph of the temporal graph learning. (b) Causal graph with unchanged node correlations.

1) Information Propagation in GNNs: Given the anchor node v_x and its neighbors N_{v_x}, GNNs propagate information from the neighbors to the anchor node to generate an accurate representation for v_x. Specifically, the representation learning in GNNs mainly consists of two stages, i.e., the information aggregation and the representation combination, which can be formulated as follows:

$$\begin{aligned} \mathbf{a}_{v_x}^k &= \text{AGGREGATE}\left(\mathbf{e}_{v_i}^k : v_i \in N_{v_x}\right) \\ \mathbf{h}_{v_x}^k &= \text{COMBINE}\left(\mathbf{h}_{v_x}^{k-1}, \mathbf{a}_{v_x}^k\right) \end{aligned} \tag{1}$$

where $\mathbf{e}_{v_i}^k$ indicates the embeddings of the neighbor node $v_i \in N_{v_x}$, and the function AGGREGATE first aggregates the embeddings of the neighbor nodes in N_{v_x} to obtain the information propogated to v_x at the k-layer, i.e., $\mathbf{a}_{v_x}^k$. Then, combining $\mathbf{a}_{v_x}^k$ with the representation of v_x at the $k-1$-layer $\mathbf{h}_{v_s}^{k-1}$ using the function COMBINE, we can generate the updated representation of v_x at the k-layer $\mathbf{h}_{v_x}^k$.

2) Causal Analysis on Temporal Graph Learning: Given the temporal networks at the timestamps $t-1$ and t as $\mathcal{G}_{t-1} = \{\mathcal{V}_{t-1}, \mathcal{E}_{t-1}\}$, and $\mathcal{G}_t = \{\mathcal{V}_t, \mathcal{E}_t\}$, respectively, we denote the new links appearing in $\mathcal{G}_t$ compared with $\mathcal{G}_{t-1}$ as $\mathbf{I}_t$. The nodes in $\mathcal{V}_{t-1}$ not connected by the new links $\mathbf{I}_t$ are represented as the inactive nodes, while the nodes in $\mathcal{V}_{t-1}$ associated with $\mathbf{I}_t$ and the newly appearing nodes in $\mathcal{G}_t$ compared with $\mathcal{G}_{t-1}$ are denoted as the active nodes. Then we can construct a causal graph detailing the causal correlations between those variables as shown in Fig. 2(a).

In Fig. 2(a), $\mathbf{E}_{\text{Ac},t}$ and $\mathbf{E}_{\text{In},t}$ are the embeddings of the active and inactive nodes at the timestamp t, respectively, while $\mathbf{E}_{\text{Ac},t-1}$ and $\mathbf{E}_{\text{In},t-1}$ are the corresponding embeddings at the timestamp $t-1$. Then, we introduce the meanings of the edges in the causal graph as follows.

1) $(\mathbf{E}_{\text{Ac},t-1}, \mathbf{I}_t) \rightarrow \mathbf{E}_{\text{Ac},t}$ indicates that the active node embeddings at the timestamp t are updated using the new links $\mathbf{I}_t$ on the basis of their embeddings at the timestamp $t-1$.
2) $\mathbf{E}_{\text{In},t-1} \rightarrow \mathbf{E}_{\text{In},t}$ denotes that the inactive node embeddings remain unchanged from the timestamp $t-1$ to t.

However, merely adopting $\mathbf{E}_{\text{Ac},t-1}$ and $\mathbf{I}_t$ to learn $\mathbf{E}_{\text{Ac},t}$ is insufficient for making temporal link predictions, because the correlations between active nodes and inactive nodes are not fully considered. To solve this problem, $\mathbf{E}_{\text{In},t-1}$ can also be adopted to enhance the representation learning of $\mathbf{E}_{\text{Ac},t}$. In order to establish the correlation

between $\mathbf{E}_{\mathrm{In},t-1}$ and $\mathbf{E}_{\mathrm{Ac},t}$, we first make an assumption as follows.

Assumption 1: The correlations between the active and inactive nodes remain unchanged with the temporal network evolving from timestamp $t-1$ to t, which can be formulated as follows:

$$\mathbf{C}_t = c_{t-1} = \mathrm{corr}(\mathbf{E}_{\mathrm{In},t-1}, \mathbf{E}_{\mathrm{Ac},t-1}) \tag{2}$$

where $\mathbf{C}_t$ is the correlations between $\mathbf{E}_{\mathrm{In},t}$ and $\mathbf{E}_{\mathrm{Ac},t}$ at the timestamp t, c_{t-1} is the corresponding correlations at the timestamp $t-1$, which is already known at the timestamp t under the assumption, and corr is the correlation measuring function.

Based on the assumption wherein $\mathbf{C}_t$ is known, the correlation between $\mathbf{E}_{\mathrm{In},t}$ and $\mathbf{E}_{\mathrm{Ac},t}$ can be established as follows.

1) $\mathbf{E}_{\mathrm{In},t} \leftrightarrow \mathbf{E}_{\mathrm{Ac},t}$, which indicates that based on the condition $\mathbf{C}_t = c_{t-1}$, $\mathbf{E}_{\mathrm{In},t}$ and $\mathbf{E}_{\mathrm{Ac},t}$ are dependent.

Then, the causal graph conditioned on Assumption 1 can be represented as Fig. 2(b). Based on the correlations of $\mathbf{E}_{\mathrm{In},t-1} \rightarrow \mathbf{E}_{\mathrm{In},t}$ and $\mathbf{E}_{\mathrm{In},t} \leftrightarrow \mathbf{E}_{\mathrm{Ac},t}$, we can deduce $\mathbf{E}_{\mathrm{In},t-1} \rightarrow \mathbf{E}_{\mathrm{Ac},t}$. Thus, for learning $\mathbf{E}_{\mathrm{Ac},t}$, two main sources of effects are considered as follows.

a) Direct Effect: Based on $(\mathbf{E}_{\mathrm{Ac},t-1}, \mathbf{I}_t) \rightarrow \mathbf{E}_{\mathrm{Ac},t}$, $\mathbf{E}_{\mathrm{Ac},t}$ will be influenced by the direct effect (DE) from the old representations of the active nodes $\mathbf{E}_{\mathrm{Ac},t-1}$ and the new interactions $\mathbf{I}_t$, which is denoted as $\mathrm{DE}_{\mathbf{E}_{\mathrm{Ac},t-1},\mathbf{I}_t}$. During the model optimization of the link prediction task, the DE can be considered by updating the embeddings of the active nodes from $\mathbf{E}_{\mathrm{Ac},t-1}$ to $\mathbf{E}_{\mathrm{Ac},t}$ with using $\mathbf{I}_t$ as the ground truth.

b) Colliding Effect: Based on $\mathbf{E}_{\mathrm{In},t-1} \rightarrow \mathbf{E}_{\mathrm{In},t} \leftrightarrow \mathbf{E}_{\mathrm{Ac},t}$, $\mathbf{E}_{\mathrm{Ac},t}$ will also receive the colliding effect (CE) [50] from the old embeddings of the inactive nodes $\mathbf{E}_{\mathrm{In},t-1}$. Specifically, given the known $\mathbf{C}_t = c_{t-1}$, the colliding effect from $\mathbf{E}_{\mathrm{In},t-1}$ to $\mathbf{E}_{\mathrm{Ac},t}$ can be calculated as

$$\begin{aligned}
&P(\mathbf{E}_{\mathrm{Ac},t}|\mathbf{C}_t = c_{t-1}, \mathbf{E}_{\mathrm{In},t-1}, \mathbf{E}_{\mathrm{Ac},t-1}, \mathbf{I}_t) \\
&\quad - P(\mathbf{E}_{\mathrm{Ac},t}|\mathbf{C}_t = c_{t-1}, \mathbf{E}_{\mathrm{In},t-1} = \mathbf{0}, \mathbf{E}_{\mathrm{Ac},t-1}, \mathbf{I}_t) \\
&\quad \overset{(1)}{=} P(\mathbf{E}_{\mathrm{Ac},t}|\mathbf{C}_t = c_{t-1}, \mathbf{E}_{\mathrm{In},t-1}, \mathbf{E}_{\mathrm{Ac},t-1}, \mathbf{I}_t) \\
&\quad \overset{(2)}{=} \sum_{\mathbf{E}_{\mathrm{In},t}} P(\mathbf{E}_{\mathrm{Ac},t}|\mathbf{C}_t = c_{t-1}, \mathbf{E}_{\mathrm{In},t-1}, \mathbf{E}_{\mathrm{Ac},t-1}, \mathbf{I}_t, \mathbf{E}_{\mathrm{In},t}) \\
&P(\mathbf{E}_{\mathrm{In},t}|\mathbf{E}_{\mathrm{In},t-1}) \\
&\quad \overset{(3)}{=} \sum_{\mathbf{E}_{\mathrm{In},t}} P(\mathbf{E}_{\mathrm{Ac},t}|\mathbf{C}_t = c_{t-1}, \mathbf{I}_t, \mathbf{E}_{\mathrm{In},t}) P(\mathbf{E}_{\mathrm{In},t}|\mathbf{E}_{\mathrm{In},t-1})
\end{aligned} \tag{3}$$

where the colliding effect can be first defined as the changing of $\mathbf{E}_{\mathrm{Ac},t}$ when $\mathbf{E}_{\mathrm{In},t-1}$ is transformed from the reference state (i.e., $\mathbf{E}_{\mathrm{In},t-1} = \mathbf{0}$) to the reality state. Next, we explain the derivation step by step: the first equation works because the latter part can be treated as zero considering that $\mathbf{E}_{\mathrm{Ac},t}$ will not be updated by $\mathbf{E}_{\mathrm{In},t-1}$ if $\mathbf{E}_{\mathrm{In},t-1}$ is empty; the second equation is based on the definition of the total probability formula; the final one is due to that in the left part, $\mathbf{E}_{\mathrm{Ac},t-1}$ which can be calculated by c_{t-1} and $\mathbf{E}_{\mathrm{In},t-1}$ can be removed, and $\mathbf{E}_{\mathrm{In},t-1}$ is already the condition of the right part which can be further reduced in the left part. Finally, the colliding effect can be represented as a sum of the conditional probability of $\mathbf{E}_{\mathrm{Ac},t}$, weighted by the probability distribution of $\mathbf{E}_{\mathrm{In},t}$ toward $\mathbf{E}_{\mathrm{In},t-1}$.

3) Discussion: By comparing the modeling process of GNNs and the causal analysis of the temporal graph representation learning, we can observe that the embeddings from the anchor node itself (i.e., $\mathbf{h}^0_{v_x}$) to be optimized by the new links can be regarded as the direct effect $\mathrm{DE}_{\mathbf{E}_{\mathrm{Ac},t-1},\mathbf{I}_t}$. Moreover, the AGGREGATE function which propagates information from the historical neighbors can correspond to the colliding effect $\mathrm{CE}_{\mathbf{E}_{\mathrm{In},t-1}}$. And the COMBINE function can be mapped to considering the direct effect $\mathrm{DE}_{\mathbf{E}_{\mathrm{Ac},t-1},\mathbf{I}_t}$ as well as the colliding effect $\mathrm{CE}_{\mathbf{E}_{\mathrm{In},t-1}}$ simultaneously.

Based on those analysis, we can conclude that under the assumption that the correlations between the active and the inactive nodes remain unchanged, i.e., Assumption 1, we can adopt the modeling process of GNNs to incorporate the historical neighbors of the anchor node $v_x \in \{v_s, v_t\}$ into learning accurate representations for v_x in the current timestamp.

However, the graph data keeps evolving in the temporal networks, indicating that the correlations between most nodes are dynamic. For example, in social networks, the relationship between two friends becomes estranged due to a conflict. Thus, an important problem occurs:

> How can we deal with the recent dynamic node correlations to approximate Assumption 1 so that GNNs can be well adopted to learn the node representations on temporal networks?

In addition, besides the recent dynamic correlations between nodes, there also exists long-term stable correlations between nodes, such as the kinship in the social networks. Thus, another problem shows up:How to preserve the long-term stable correlations between nodes in the dynamic evolution of the temporal networks?

To tackle the two problems and learn accurate node representations, we propose a TAG learning approach for link prediction on temporal networks based on the conclusion with the aforementioned causal analysis.

C. Tag Learning

The workflow of TAG is presented in Fig. 3, which contains three main components, i.e., the dynamic correlation modeling (DCM) module, the stable correlation preserving (SCP) module and the multitask learning component. Given the existing temporal graph, on the one hand, in the DCM module, we first abandon the outdated training samples which may bring noise for accurate node representation learning, and then sample the recent neighbors of $v_x \in \{v_s, v_t\}$ to generate the dynamic node representations of v_x for the main model optimization; on the other hand, in the SCP module, we contrast the dynamic and stable representations of nodes to preserve the long-term stable node correlations. Finally, the TAG model is optimized in a multitask learning way by combining the former two components.

1) Dynamic Correlation Modeling: Through the causal analysis in Section III-B, we obtain the conclusion that the recent dynamic node correlations should be emphasized to

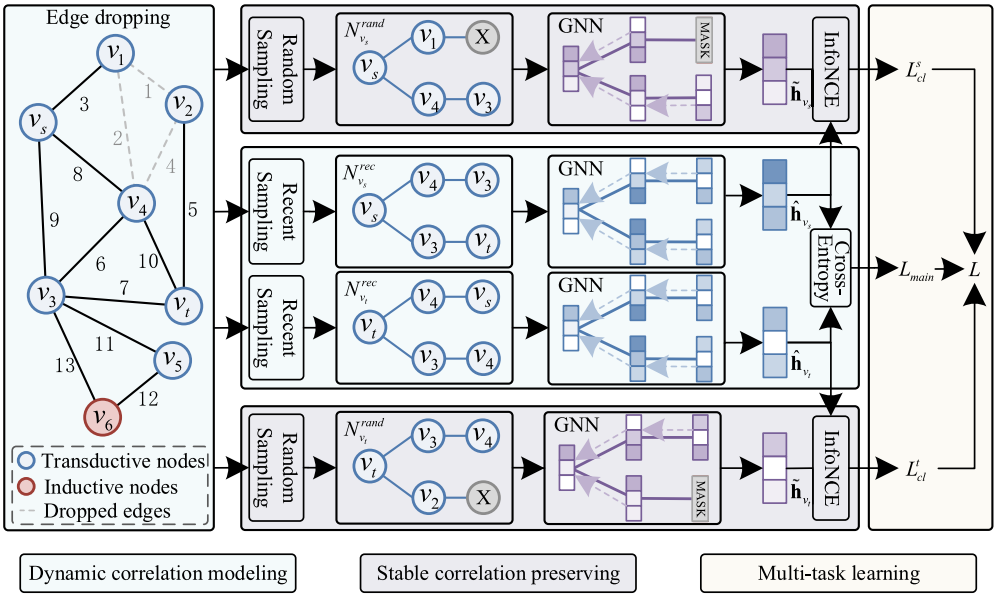

Fig. 3. Framework of TAG, where X denotes the padding neighbor.

approximate Assumption 1 so that we can adopt GNNs to learn accurate node representations. However, the condition that the previous correlations between the neighbors N_{v_x} and the anchor node v_x remain unchanged in the current timestamp (i.e., $\mathbf{C}_t = c_{t-1}$) cannot be well approximated with existing methods. First, the current training strategies generally take all the previous data in the training set for the model optimization. Besides, existing GNNs either adopt all the previous neighbors or randomly sample the neighbors for information aggregation. Those strategies inevitably introduce the outdated node correlations far away from the current timestamp on the timeline, which lead to $\mathbf{C}_t \neq c_{t-1}$.

To solve the above issues, we propose to conduct the DCM by avoiding the unrelated bias that leads to $\mathbf{C}_t \neq c_{t-1}$ through two strategies, i.e., the ED and the RNS, respectively.

a) Edge Dropping: First, in the training data selection stage, we propose to abandon the outdated training samples where the contained node correlations obviously differ from the current timestamp. In detail, training on such samples may introduce bias from those outdated correlations and hurt the model performance in terms of future link predictions.

Specifically, given the current timestamp t_{st} and each training sample in the training set, i.e., $(v_{s_i}, v_{t_i}, t_{s_i t_i}) \in \mathcal{E}$, the probability of each sample to be selected for training can be generated by the recent sampling strategy as

$$F_{\text{edge}}((v_{s_i}, v_{t_i}, t_{s_i t_i})) = \frac{e^{\alpha(t_{s_i t_i} - t_{st})}}{\sum_{(v_{s_i}, v_{t_i}, t_{s_i t_i}) \in \mathcal{E}_{\text{train}}} e^{\alpha(t_{s_i t_i} - t_{st})}} \tag{4}$$

where α is the hyper-parameter for controlling the intensity of the recent training sample selection, and the recent sampling is degenerated to the random sampling when $\alpha = 0$.

Finally, the training samples are selected by the nonrepeated sampling strategy with the above-calculated probabilities F_{edge}. In addition, a tradeoff parameter ρ ($\rho < 1$) is set to control the proportion of the training samples to be selected. The new training set is denoted as $\mathcal{E}' \subset \mathcal{E}$, and correspondingly the other samples are abandoned. This can be formulated as

$$\mathcal{E}' = \text{Sample}(F_{\text{edge}}, \rho). \tag{5}$$

b) Neighbor Sampling: In addition to the ED, in the neighbor sampling stage, we adopt the recent sampling strategy to select the historical neighbors for information aggregation. Specifically, the closer to the current timestamp t_{st} the neighbor $v_i \in N_{v_x}$ interacted with v_x, the larger the probability of it being sampled is, which can be formulated as

$$F_{\text{ngh}}(v_i) = \frac{e^{\beta(t_i - t_{st})}}{\sum_{v_i \in N_{v_x}} e^{\beta(t_i - t_{st})}} \tag{6}$$

where t_i is the timestamp of the interaction between the anchor node $v_x \in \{v_s, v_t\}$ and the neighbor $v_i \in N_{v_x}$, and β is a hyper-parameter for deciding the intensity of the RNS. That is, through the recent sampling, the timespan between the timestamps that v_x interacts with the sampled neighbors to the current timestamp can be shortened, thus approximating the condition that the correlations between the neighbors and the anchor node remain unchanged, i.e., $\mathbf{C}_t = c_{t-1}$.

Finally, we can extract the recent neighbors of v_x from N_{v_x} as $N^{\text{rec}}_{v_x}$, where $M = |N^{\text{rec}}_{v_x}|$ is the number of the sampled neighbors. It is worth noting that multihop neighbors can also be extracted, however here we set the hop length to 1 by following [51], [52]. This is due to: first, it has been validated that one-hop neighbors can already well represent the historical characteristics of nodes [51]; moreover, though multihop neighbors may bring in relatively more abundant interactions for describing nodes, bias is also easily introduced [25]; in addition, considering multihop neighbors will obviously decrease the computational efficiency, limiting its applications in real-world scenarios.

Here, in order to conform to the theoretical conclusion obtained in Section III-B that the recent dynamic node correlations should be emphasized, we adopt the simple recent sampling strategy to select the corresponding training samples and neighbors for making temporal link predictions. More effective and complicated strategies can also be adopted, which we leave to our future work for investigation.

c) Link Prediction and Optimization: After sampling the recent neighbors of the anchor node $v_x \in \{v_s, v_t\}$ as $N_{v_x}^{\text{rec}}$, we implement the AGGREGATE function in (1) using the simple meaning pooling to aggregate the information from $N_{v_x}^{\text{rec}}$ to v_x. Note that our proposal is a model-agnostic approach which can be applied to different aggregation methods such as graph convolution layer [53] and graph attention layer [54]. Here, we adopt the mean pooling as an example where no additional parameters are introduced, which can simultaneously avoid the overfitting problem and reduce the computation complexity. This can be formulated as follows:

$$\hat{\mathbf{h}}_{v_x}^{k} = \mathcal{F}_{\text{mean-pool}}\left(\mathbf{e}_{v_i} | v_i \in N_{v_x}^{\text{rec}}\right) \tag{7}$$

where $\hat{\mathbf{h}}_{v_x}^{k}$ is the dynamic representation of v_x generated at the k-layer.

Next, for the COMBINE function in (1), we also adopt the mean pooling to combine the embeddings of the anchor node v_x and the information propagated from different layers of neighbors for generating its final representation $\hat{\mathbf{h}}_{v_x}$

$$\hat{\mathbf{h}}_{v_x} = \mathcal{F}_{\text{mean-pool}}\left(\hat{\mathbf{h}}_{v_x}^{0}, \hat{\mathbf{h}}_{v_x}^{1}, \ldots, \hat{\mathbf{h}}_{v_x}^{K}\right) \tag{8}$$

where it is worth noting that the representation at the 0-layer $\hat{\mathbf{h}}_{v_x}^{0}$ is the node embeddings of v_x, i.e., $\mathbf{e}_{v_x} \in \mathbb{R}^d$, which is generated by the embedding layer as $\mathbf{e}_{v_x} = \text{Embedding}(v_x)$, and d is the embedding dimension.

After that, we can obtain the respective dynamic representations of the source node v_s and the target node v_t as $\hat{\mathbf{h}}_{v_s}$ and $\hat{\mathbf{h}}_{v_t}$ by aggregating the corresponding neighbor information. Next, we first concatenate $\hat{\mathbf{h}}_{v_s}$ and $\hat{\mathbf{h}}_{v_t}$, and then input them into a multiperception (MLP) layer to calculate the probability of generating a link between them at the timestamp t_{st}, which can be formulated as

$$\mathbf{y}_{st} = \text{MLP}([\hat{\mathbf{h}}_{v_s}, \hat{\mathbf{h}}_{v_t}]) = \mathbf{W}_2(\sigma(\mathbf{W}_1[\hat{\mathbf{h}}_{v_s}, \hat{\mathbf{h}}_{v_t}])) \tag{9}$$

where $[\cdot]$ indicates the concatenation operation, $\mathbf{W}_1 \in \mathbb{R}^{d \times 2d}$ and $\mathbf{W}_2 \in \mathbb{R}^{1 \times d}$ are the trainable parameters in MLP, and σ denotes the rectified linear unit (ReLU) function.

At last, we adopt the cross-entropy function as the optimization objective to learn the trainable parameters

$$L_{\text{main}} = \sum_{(v_s, v_t, t_{st}) \in \mathcal{E}'} -\log(\sigma(\mathbf{y}_{st})) - \log(\sigma(1 - \mathbf{y}_{sn})) \tag{10}$$

where $\mathbf{y}_{st}$ and $\mathbf{y}_{sn}$ are the prediction scores of the observed temporal edge in the training set $(v_s, v_t, t_{st}) \in \mathcal{E}'$ and the corresponding negative edge (v_s, v_n, t_{st}), respectively, v_n is the randomly selected negative sample, and σ indicates the Sigmoid function.

2) Stable Correlation Preserving: Though the DCM module can take the recent interactions of the anchor node $v_x \in \{v_s, v_t\}$ into consideration, however, this may limit the temporal scale of the receptive neighborhood. Specifically, the long-term stable correlations between nodes, such as the kinship in the social networks, are easily ignored in the DCM. To prevent this problem, we adopt the contrastive learning to maximize the mutual information between the node representations generated with the long-term stable and recent dynamic correlations, respectively. Thus, the representation learning on the temporal networks can be enhanced to preserve the long-term stable node correlations.

Specifically, given the node pairs in the current mini-batch as $[(v_{s_1}, v_{t_1}, t_{s_1 t_1}), (v_{s_2}, v_{t_2}, t_{s_2 t_2}), \ldots, (v_{s_N}, v_{t_N}, t_{s_N t_N})]$, the dynamic representations of the nodes can be generated by GNNs through (6)–(8) as $[(\hat{\mathbf{h}}_{s_1}, \hat{\mathbf{h}}_{t_1}), (\hat{\mathbf{h}}_{s_2}, \hat{\mathbf{h}}_{t_2}), \ldots, (\hat{\mathbf{h}}_{s_N}, \hat{\mathbf{h}}_{t_N})]$. In addition to the dynamic representations, we further randomly select the neighbors of each source and target node $v_x \in \{v_{s_i}, v_{t_i}\}$ with a long-term temporal scale from the full historical neighbors N_{v_x} as $N_{v_x}^{\text{rand}}$. Then, we generate the corresponding stable representations of the source and target nodes by fusing the long-term stable neighbor information as $[(\tilde{\mathbf{h}}_{s_1}, \tilde{\mathbf{h}}_{t_1}), (\tilde{\mathbf{h}}_{s_2}, \tilde{\mathbf{h}}_{t_2}), \ldots, (\tilde{\mathbf{h}}_{s_N}, \tilde{\mathbf{h}}_{t_N})]$ by (7) and (8).

After that, take the source nodes as an example, we aim to push their dynamic representations $[\hat{\mathbf{h}}_{s_1}, \hat{\mathbf{h}}_{s_2}, \ldots, \hat{\mathbf{h}}_{s_N}]$ and stable representations $[\tilde{\mathbf{h}}_{s_1}, \tilde{\mathbf{h}}_{s_2}, \ldots, \tilde{\mathbf{h}}_{s_N}]$ together in the embedding space. Thus, the long-term stable neighbor information can be injected into the dynamic representation generation to preserve the stable correlations between nodes. Specifically, we adopt the InfoNCE [55] as the contrastive loss, which takes the dynamic representations of v_{s_i} (i.e., $\hat{\mathbf{h}}_{s_i}$) with its stable representations $\tilde{\mathbf{h}}_{s_i}$ as the positive pair, and regards $\hat{\mathbf{h}}_{s_i}$ with the stable representations of the other source nodes in the current mini-batch (i.e., $[\tilde{\mathbf{h}}_{s_1}, \ldots, \tilde{\mathbf{h}}_{s_{i-1}}, \tilde{\mathbf{h}}_{s_{i+1}}, \ldots, \tilde{\mathbf{h}}_{s_N}]$) as the negative pairs. Thus the contrastive loss among the source nodes $[v_{s_1}, v_{s_2}, \ldots, v_{s_N}]$ in the current mini-batch, which we denote as L_{cl}^{s}, can be formulated as

$$L_{\text{cl}}^{s} = \sum_{i=1}^{N} \frac{\exp(\tau \text{sim}(\hat{\mathbf{h}}_{s_i}, \tilde{\mathbf{h}}_{s_i}))}{\exp(\tau \text{sim}(\hat{\mathbf{h}}_{s_i}, \tilde{\mathbf{h}}_{s_i})) + \sum_{j=1, j \neq i}^{N} \exp(\tau \text{sim}(\hat{\mathbf{h}}_{s_i}, \tilde{\mathbf{h}}_{s_j}))} \tag{11}$$

where $\text{sim}(\mathbf{u}, \mathbf{v}) = \cos(\mathbf{u}, \mathbf{v}) = \mathbf{u}^{\top}\mathbf{v}/\|\mathbf{u}\|\|\mathbf{v}\|$ is the function measuring the similarity between the vectors $\mathbf{u} \in \mathbb{R}^d$ and $\mathbf{v} \in \mathbb{R}^d$, $\|\cdot\|$ denotes the L2 normalization, and τ is the temperature parameter for scaling the cosine similarity.

Similar to that among the source nodes, the contrastive loss among the target nodes $[v_{t_1}, v_{t_2}, \ldots, v_{t_N}]$ in the current mini-batch can also be calculated as follows:

$$L_{\text{cl}}^{t} = \sum_{i=1}^{N} \frac{\exp(\tau \text{sim}(\hat{\mathbf{h}}_{t_i}, \tilde{\mathbf{h}}_{t_i}))}{\exp(\tau \text{sim}(\hat{\mathbf{h}}_{t_i}, \tilde{\mathbf{h}}_{t_i})) + \sum_{j=1, j \neq i}^{N} \exp(\tau \text{sim}(\hat{\mathbf{h}}_{t_i}, \tilde{\mathbf{h}}_{t_j}))} \tag{12}$$

where $\hat{\mathbf{h}}_{t_i}$ is the dynamic representations of the target node v_{t_i}, and $\tilde{\mathbf{h}}_{v_{t_i}}$ and $\tilde{\mathbf{h}}_{v_{t_j}}$ are the stable representations of the nodes v_{t_i} and v_{t_j}, respectively.

Finally, we combine the contrastive loss among the source nodes (i.e., L_{cl}^{s}) and the target nodes (i.e., L_{cl}^{t}) in the current

Algorithm 1 Training Procedure of Our Proposed TAG

Input: $\mathcal{G} = \{\mathcal{V}, \mathcal{E}\}$: The temporal network;
Output: The links appearing in $\mathcal{G}$ in the future;
1: $\mathcal{E}'$ = EdgeDropping($\mathcal{E}$) based on Equations (4) and (5);
2: **for** epoch in range(Epochs) **do**
3: **for** b in range(Batches) **do**
4: **for** (v_s, v_t, t_{st}) in $\mathcal{E}'_b \subseteq \mathcal{E}'$ **do**
5: $v_n \leftarrow$ RandomNegativeSampling($\mathcal{V}$);
6: $N^{rec}_{v_s}, N^{rec}_{v_t}, N^{rec}_{v_n} \leftarrow$ RecentNeighborSampling(v_s, v_t, t_{st}) based on Equation (6);
7: $\hat{\mathbf{h}}_{v_s}, \hat{\mathbf{h}}_{v_t}, \hat{\mathbf{h}}_{v_n} \leftarrow$ GNN($N^{rec}_{v_s}, N^{rec}_{v_t}, N^{rec}_{v_n}$) based on Equations (7) and (8);
8: $\mathbf{y}_{st}, \mathbf{y}_{sn} \leftarrow$ Prediction($\hat{\mathbf{h}}_{v_s}, \hat{\mathbf{h}}_{v_t}, \hat{\mathbf{h}}_{v_n}$) based on Equation (9);
9: $L_{\text{main}} \leftarrow$ Cross-Entropy($\mathbf{y}_{st}, \mathbf{y}_{sn}$) based on Equation (10);
10: $N^{rand}_{v_s}, N^{rand}_{v_t} \leftarrow$ RandomNeighborSampling(v_s, v_t, t_{st});
11: $\tilde{\mathbf{h}}_{v_s}, \tilde{\mathbf{h}}_{v_t} \leftarrow$ GNN($N^{rand}_{v_s}, N^{rand}_{v_t}, N^{rand}_{v_n}$) based on Equations (7) and (8);
12: $L^s_{\text{cl}} \leftarrow$ InfoNCE($\hat{\mathbf{h}}_{v_s}, \tilde{\mathbf{h}}_{s_j} | j = 1, \ldots, N$) based on Equation (11);
13: $L^t_{\text{cl}} \leftarrow$ InfoNCE($\hat{\mathbf{h}}_{v_t}, \tilde{\mathbf{h}}_{s_j} | j = 1, \ldots, N$) based on Equation (12);
14: $L_{\text{cl}} \leftarrow L^s_{\text{cl}} + L^t_{\text{cl}}$ based on Equation (13);
15: **end for**
16: Optimize the final loss $L_{\text{main}} + \lambda L_{\text{cl}}$ with joint learning;
17: **end for**
18: **end for**
19: **return** The learnable parameters of TAG and the node embeddings.

mini-batch as the final contrastive loss L_{cl} of the SCP component as follows:

$$L_{\text{cl}} = L^s_{\text{cl}} + L^t_{\text{cl}}. \tag{13}$$

3) Multitask Learning: After obtaining the main supervised loss by (10) and the contrastive loss by (13), we combine them together as the final training loss

$$L = L_{\text{main}} + \lambda L_{\text{cl}} \tag{14}$$

where λ is the tradeoff parameter. Finally, we adopt the back-propagation through time (BPTT) algorithm to train our proposed TAG model.

The training process of TAG is shown in Algorithm 1. Specifically, given the temporal networks until the current timestamp as $\mathcal{G} = \{\mathcal{V}, \mathcal{E}\}$, we first conduct the ED in line 1. Then, for each remained training sample, we first randomly select the negative samples in line 5. Next, we extract the recent dynamic neighbors of the source node v_s and the target node v_t as well as the negative sample v_n by the recent sampling in line 6; and then generate the dynamic representations of them using GNNs in line 7. After that, the prediction scores of the positive and negative node pairs are generated in line 8, which are then adopted to obtain the main supervised loss in line 9. In addition, the long-term stable neighbors of v_s and v_t are randomly sampled in line 10, and the corresponding stable representations are generated by GNNs in line 11. Then, we obtain the contrastive loss in line 14, from the aspects of the source node side in line 12 and the target node side in line 13, respectively. Finally, the main supervised loss and the contrastive loss are combined for the model optimization in line 16.

In addition, the time complexity of TAG can be analyzed from Algorithm 1. Specifically, let M be the number of the sampled neighbors, K be the number of GNN layers, and N be the number of instances in each mini-batch, TAG's time complexity is mainly constituted of the following two parts: 1) lines 7–9, the time complexity of the DCM module is $Md + Kd + d^2$, which mainly consists of Md from (7), Kd from (8), d^2 from (9), respectively; and 2) lines 11–13, the time complexity of the SCP module is $Md + Kd + Nd^2$, which mainly consists of Md from (7), Kd from (8), Nd^2 from (11) and (12), respectively. In addition, the time complexity of lines 1, 5, 6, 10, 14, and 16 is negligible compared to the above two parts and thus can be neglected. Thus, the time complexity of TAG is $Md + Kd + Nd^2$ by summarizing the time complexity of lines 7–9 and lines 11–13.

IV. Experiments

A. Research Questions

We investigate the effectiveness of the proposed TAG model by answering the following five research questions.

(RQ1) Can TAG achieve state-of-the-art performance on the link prediction task on temporal networks, and how is the efficiency of TAG compared to the baselines?
(RQ2) How is the contribution of different designed components of TAG to its prediction performance?
(RQ3) How is the impact of the ED ratio ρ on the prediction performance as well as the efficiency of TAG?
(RQ4) How is the performance of TAG on the networks containing different time stages of temporal graph data?
(RQ5) How is the sensitivity of TAG to its contained hyperparameters?

B. Datasets

To validate the effectiveness of TAG on the link prediction task on temporal networks, we adopt four publicly available

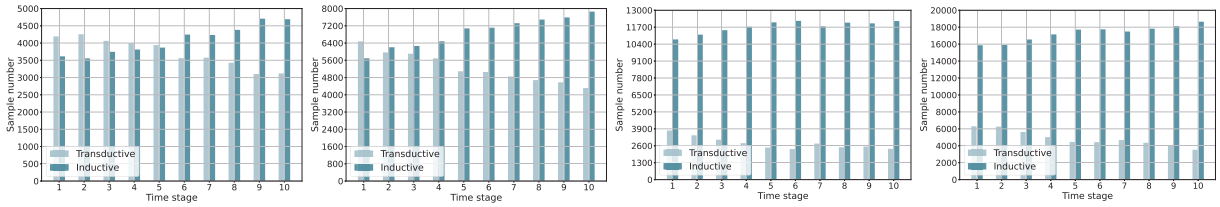

Fig. 4. Data distribution on different time stages. (a) MathOverflow. (b) StackOverflow. (c) AskUbuntu. (d) SuperUser.

TABLE III
STATISTICS OF THE DATASETS USED IN OUR EXPERIMENTS

Statistics	MathOverflow	StackOverflow	AskUbuntu	SuperUser
# Nodes	24,759	47,194	157,222	192,409
# Training nodes	16,305	34,749	112,144	123,107
# Unseen valid nodes	2,913	4,416	15,323	23,408
# Unseen test nodes	5,541	8,299	29,755	45,894
# Edges	390,441	608,421	726,661	1,108,739
# Training edges	273,309	425,895	508,662	776,117
# Valid edges	39,044	60,842	72,667	110,874
# Test edges	78,088	121,684	145,332	221,748
# Transductive test edges	37,238	52,603	28,003	48,718
# Inductive test edges	40,850	69,081	117,329	173,030
Training node percentage	65.85%	73.63%	71.32%	63.98%
Density	0.00063	0.00027	0.00003	0.00003

datasets for experiments, i.e., MathOverflow,[1] StackOverflow,[2] AskUbuntu,[3] and SuperUser,[4] which are collected from the stack exchange web sites Math Overflow, Stack Overflow, Ask Ubuntu, and Super User, respectively. Four datasets are all the temporal networks of interactions, where MathOverflow, AskUbuntu, and SuperUser are directly adopted without further processing, and the first 1% of the interactions in StackOverflow are selected for experiments considering its extremely large size. In addition, we divide the full data in each dataset according to the percents of 70%, 10%, and 20% for training, validation, and test, respectively, and further divide the test edges into the transductive and inductive test edges. The statistics of the processed datasets are presented in Table III, where the inductive valid and test nodes denote the nodes newly appearing in the validation set and test set, respectively. Moreover, the training node percentage is represented as the ratio of the number of the training nodes to that of all the nodes in the dataset, and the density is calculated by $\mathcal{E}|/|\mathcal{V}|^2$. In addition, we chronologically divided the test data into 10 time stages with the same test edge number, and plot the distribution of the number of the transductive and inductive test edges on various time stages in Fig. 4. It is obvious that with time going on, the number of transductive edges gradually decreases while the number of inductive edges generally increases, respectively.

C. Model Summary

To validate the effectiveness of our proposed TAG, three groups of 11 competitive baselines are compared against TAG, including: 1) one method without using GNNs, i.e., MF [56]; 2) four embedding-based methods with GNNs, including three static approaches, i.e., SAGE [57], GAT [54] as well as GIN [58], and one dynamic approach, i.e., DySAT [28]; and 3) six temporal structure aware methods, i.e., TGAT [23], GraphMixer [38], RepeatMixer [59], CAW [18], NTW [25], and DyGFormer [51].

1) MF maps the nodes into the embedding space and learns the embeddings by reconstructing the evolving adjacent matrix.
2) SAGE generates the node representations by aggregating information from the neighbors in a sampled local neighborhood.
3) GAT improves SAGE by adopting an attention mechanism to dynamically assign weights to the neighbors in the substructure.
4) GIN develops a GNN as provably maximally powerful as the Weisfeiler-Lehman test [60] under the information aggregation framework by summing the neighbor features together.
5) DySAT computes the node representations by simultaneously considering the structural neighborhood and temporal dynamics using the self-attention mechanism.
6) TGAT designs a temporal attention layer to aggregate the information of temporal dynamics as well as topological structure, and achieves the continuous time encoding by the Bochner theorem [61].
7) GraphMixer proposes a simple neural architecture which yields fast convergence speed and well generalization ability using the MLP-Mixer [62].
8) RepeatMixer improves the neighbor sampling of existing methods by designing a repeat-aware sampling strategy for exploiting the pair-wise temporal pattern between nodes.
9) CAW proposes a temporal anonymous walk to generate the relative identity embeddings between nodes by measuring their distance on the temporal graph structure for link prediction.
10) NTW improves CAW by characterizing the temporal evolution of nodes by introducing the spatiotemporal-biased random walks to retrieve the representative motifs.
11) DyGFormer designs a neighbor co-occurrence encoding mechanism to learn the correlations between nodes and adopt a patching strategy to benefit from the node's long historical neighbor sequence.

D. Experimental Setup

The hyper-parameters of TAG and the baselines are tuned using the full validation data, which are then applied to the

[1]https://snap.stanford.edu/data/sx-mathoverflow.html
[2]https://snap.stanford.edu/data/sx-stackoverflow.html
[3]https://snap.stanford.edu/data/sx-askubuntu.html
[4]https://snap.stanford.edu/data/sx-superuser.html

TABLE IV

OVERALL PERFORMANCE OF THE BASELINES AND TAG IN TERMS OF AP AND AUC. THE RESULTS OF THE BEST PERFORMING BASELINE AND THE BEST PERFORMER IN EACH COLUMN ARE UNDERLINED AND BOLDFACED, RESPECTIVELY

Task	Method	MathOverflow		StackOverflow		AskUbuntu		SuperUser	
		AP(%)	AUC(%)	AP(%)	AUC(%)	AP(%)	AUC(%)	AP(%)	AUC(%)
Transductive	MF	$79.96_{\pm 0.24}$	$77.41_{\pm 0.43}$	$69.78_{\pm 0.07}$	$66.70_{\pm 0.10}$	$65.81_{\pm 0.38}$	$61.68_{\pm 0.40}$	$65.08_{\pm 0.80}$	$61.38_{\pm 0.99}$
	SAGE	$80.86_{\pm 0.09}$	$80.16_{\pm 0.09}$	$71.23_{\pm 0.14}$	$69.47_{\pm 0.15}$	$65.94_{\pm 0.21}$	$63.55_{\pm 0.22}$	$67.41_{\pm 0.35}$	$65.27_{\pm 0.42}$
	GAT	$80.82_{\pm 0.11}$	$80.16_{\pm 0.15}$	$70.84_{\pm 0.20}$	$69.26_{\pm 0.20}$	$66.20_{\pm 0.55}$	$64.02_{\pm 0.59}$	$68.44_{\pm 0.16}$	$66.37_{\pm 0.16}$
	GIN	$74.86_{\pm 0.23}$	$74.71_{\pm 0.16}$	$66.92_{\pm 0.20}$	$66.40_{\pm 0.12}$	$65.83_{\pm 0.30}$	$64.23_{\pm 0.21}$	$65.30_{\pm 0.26}$	$63.85_{\pm 0.32}$
	DySAT	$80.76_{\pm 0.03}$	$80.03_{\pm 0.10}$	$70.87_{\pm 0.15}$	$69.14_{\pm 0.14}$	$66.12_{\pm 0.22}$	$63.77_{\pm 0.26}$	$67.33_{\pm 0.52}$	$65.31_{\pm 0.53}$
	TGAT	$52.30_{\pm 0.35}$	$50.61_{\pm 0.43}$	$55.56_{\pm 0.98}$	$53.24_{\pm 0.92}$	$53.39_{\pm 0.24}$	$50.48_{\pm 0.42}$	$53.66_{\pm 0.63}$	$50.73_{\pm 0.72}$
	GraphMixer	$52.78_{\pm 0.48}$	$51.44_{\pm 0.70}$	$55.13_{\pm 0.76}$	$52.98_{\pm 0.83}$	$53.25_{\pm 0.80}$	$50.68_{\pm 1.09}$	$53.34_{\pm 0.41}$	$50.48_{\pm 0.43}$
	RepeatMixer	$57.99_{\pm 0.25}$	$61.34_{\pm 0.20}$	$58.84_{\pm 0.77}$	$59.60_{\pm 0.58}$	$61.56_{\pm 0.38}$	$62.75_{\pm 0.26}$	$60.28_{\pm 0.25}$	$61.12_{\pm 0.23}$
	CAW	$71.81_{\pm 0.80}$	$74.11_{\pm 0.73}$	$67.62_{\pm 0.40}$	$67.31_{\pm 0.36}$	$69.39_{\pm 1.20}$	$68.69_{\pm 0.46}$	$67.06_{\pm 0.86}$	$66.41_{\pm 0.46}$
	NTW	$74.84_{\pm 0.33}$	$75.19_{\pm 0.27}$	$65.57_{\pm 1.09}$	$65.86_{\pm 0.18}$	$71.57_{\pm 0.46}$	**$69.16_{\pm 0.17}$**	$67.80_{\pm 0.87}$	$66.07_{\pm 0.66}$
	DyGFormer	$74.03_{\pm 1.86}$	$75.95_{\pm 0.85}$	$67.92_{\pm 0.14}$	$67.39_{\pm 0.10}$	$68.48_{\pm 1.29}$	$68.04_{\pm 1.37}$	$68.96_{\pm 0.71}$	$67.36_{\pm 0.84}$
	TAG	**$86.42_{\pm 0.12}$**	**$84.16_{\pm 0.24}$**	**$72.45_{\pm 0.08}$**	**$70.61_{\pm 0.12}$**	**$73.64_{\pm 0.45}$**	$68.17_{\pm 0.49}$	**$74.94_{\pm 0.10}$**	**$69.70_{\pm 0.17}$**
Inductive	MF	$51.17_{\pm 0.21}$	$49.54_{\pm 0.32}$	$49.15_{\pm 0.15}$	$49.28_{\pm 0.22}$	$52.04_{\pm 0.33}$	$49.40_{\pm 0.38}$	$52.13_{\pm 0.16}$	$50.05_{\pm 0.23}$
	SAGE	$75.21_{\pm 0.14}$	$73.70_{\pm 0.08}$	$62.01_{\pm 0.15}$	$63.19_{\pm 0.15}$	$66.08_{\pm 0.35}$	$62.66_{\pm 0.33}$	$65.51_{\pm 0.21}$	$61.89_{\pm 0.07}$
	GAT	$74.45_{\pm 0.36}$	$73.13_{\pm 0.21}$	$60.68_{\pm 0.09}$	$61.95_{\pm 0.07}$	$64.95_{\pm 0.57}$	$61.67_{\pm 0.51}$	$65.21_{\pm 0.07}$	$60.79_{\pm 0.18}$
	GIN	$72.29_{\pm 0.15}$	$71.07_{\pm 0.20}$	$63.39_{\pm 0.07}$	$62.13_{\pm 0.11}$	$66.07_{\pm 0.44}$	$62.49_{\pm 0.45}$	$63.37_{\pm 0.07}$	$60.14_{\pm 0.07}$
	DySAT	$75.00_{\pm 0.13}$	$73.64_{\pm 0.15}$	$61.98_{\pm 0.18}$	$62.83_{\pm 0.17}$	$66.48_{\pm 0.16}$	$62.83_{\pm 0.12}$	$65.44_{\pm 0.28}$	$61.71_{\pm 0.11}$
	TGAT	$53.64_{\pm 0.22}$	$50.31_{\pm 0.24}$	$55.10_{\pm 0.67}$	$52.55_{\pm 0.65}$	$53.95_{\pm 0.20}$	$50.31_{\pm 0.17}$	$53.96_{\pm 0.50}$	$50.43_{\pm 0.50}$
	GraphMixer	$53.76_{\pm 0.43}$	$50.75_{\pm 0.54}$	$54.82_{\pm 0.68}$	$52.53_{\pm 0.77}$	$53.91_{\pm 0.33}$	$50.42_{\pm 0.44}$	$53.66_{\pm 0.13}$	$50.26_{\pm 0.18}$
	RepeatMixer	$62.07_{\pm 0.08}$	$64.60_{\pm 0.04}$	$55.62_{\pm 0.27}$	$55.90_{\pm 0.12}$	$61.32_{\pm 0.18}$	$61.40_{\pm 0.10}$	$58.63_{\pm 0.11}$	$59.05_{\pm 0.08}$
	CAW	$67.25_{\pm 0.47}$	$66.35_{\pm 0.45}$	$63.26_{\pm 0.33}$	$61.25_{\pm 0.14}$	$66.17_{\pm 0.43}$	$63.31_{\pm 1.58}$	$63.60_{\pm 0.40}$	$61.51_{\pm 0.36}$
	NTW	$67.29_{\pm 0.36}$	$66.47_{\pm 0.27}$	$59.35_{\pm 0.79}$	$58.55_{\pm 0.31}$	$67.52_{\pm 0.37}$	$63.66_{\pm 1.47}$	$63.65_{\pm 1.10}$	$61.67_{\pm 0.40}$
	DyGFormer	$68.26_{\pm 0.70}$	$66.82_{\pm 0.60}$	$63.50_{\pm 0.21}$	$61.20_{\pm 0.18}$	$65.41_{\pm 0.76}$	$61.67_{\pm 1.98}$	$63.67_{\pm 1.47}$	$61.07_{\pm 1.42}$
	TAG	**$79.44_{\pm 0.10}$**	**$75.69_{\pm 0.23}$**	**$66.13_{\pm 0.12}$**	**$65.67_{\pm 0.11}$**	**$69.02_{\pm 0.73}$**	$63.56_{\pm 0.71}$	**$67.47_{\pm 0.04}$**	**$62.14_{\pm 0.11}$**

test set for evaluation, including testing on the transductive and inductive test edges. Specifically, the node embedding dimension and the batch size are set to 100 and 64, respectively, and the Adam optimizer is adopted for training the model parameters, where the learning rate is set to $1e^{-3}$. Moreover, for our proposed TAG, the parameters α in the ED module and β in the RNS strategy are both searched in $\{1e^{-8}, 1e^{-7}, 1e^{-6}, 1e^{-5}, 1e^{-4}\}$, the proportion of the data selection ρ is tuned in $\{10\%, 20\%, \ldots, 90\%\}$, the parameter τ in the contrastive learning component is ranged in $\{6, 8, 10, 12, 14\}$, and the parameter λ is searched in $\{1e^{-3}, 1e^{-2}, 1e^{-1}, 0.5, 1\}$. In addition, in the neighbor sampling module, the number of the sampled neighbors M is searched in $\{2, 4, 8, 16, 32\}$.

Following [18], [23], we adopt AP and AUC as the metrics for evaluating the link prediction performance of our proposed TAG as well as the baselines.

V. RESULTS

A. Overall Comparison

1) Performance Comparison: We present the experimental results of the proposed TAG and the compared baselines in terms of AP and AUC in Table IV.

First, by comparing MF and the embedding-based GNN methods (i.e., SAGE, GAT, GIN, and DySAT), we can see that the GNN-based methods except GIN can generally outperform MF. We attribute this to the fact that the neighbor embeddings of the source and target nodes can be incorporated by GNNs and updated during training, which can then help to generate accurate node representations in the inference stage. Besides, it is worth noting that in the inductive scenarios, the performance of MF is close to the random prediction strategy, where the GNN-based methods can still achieve satisfactory performance. This is because MF completely loses the ability to represent the unseen nodes while GNNs can well represent the unseen nodes to some extent by propagating information from the seen nodes.

Moreover, it is observed that among the embedding-based GNN methods, SAGE achieves the best performance in most cases, which is also the state-of-the-art performance among all baselines in most cases. The unsatisfying performance of GAT and DySAT may be caused by the abundant trainable parameters in those models which can easily lead to overfitting on the training set and failing to well generalize to the test set. The worst performance of GIN among the embedding-based GNN methods could be explained by the fact that GIN is specially designed for classification tasks, which is not optimized for the link prediction task on temporal networks.

Furthermore, for the temporal structure-aware methods, we can see that the performance of TGAT and GraphMixer, is not satisfying in all cases on four datasets, indicating that on the temporal networks without the node/edge attributes, merely considering the temporal information cannot ensure accurate

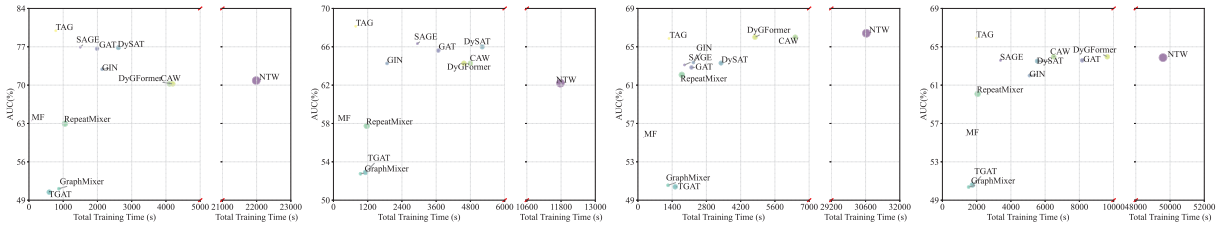

Fig. 5. Training efficiency comparison of TAG and the baselines. (a) MathOverflow. (b) StackOverflow. (c) AskUbuntu. (d) SuperUser.

link predictions. Moreover, RepeatMixer obviously improves the performance of TGAT and GraphMixer since it explicitly models the repeat behavior pattern on temporal networks. Furthermore, by taking the topological correlations between nodes on the temporal structure into consideration, CAW, NTW, and DyGFormer obviously improve the performance of TGAT, GraphMixer, and RepeatMixer. By considering both spatial and temporal information during neighbor sampling for extracting the node substructure, NTW outperforms CAW in most cases on four datasets. In addition, by designing a neighbor co-occurrence encoding scheme and adopting a patching technique for dealing with long neighbor sequences, DyGFormer outperforms CAW and NTW in most cases, especially in the transductive scenarios.

As for our proposed TAG model, we can see that TAG achieves the best performance in almost all cases in terms of AP and AUC on both transductive and inductive scenarios on four datasets, validating the effectiveness of our proposal on the link prediction task on temporal networks. Moreover, on MathOverflow, AskUbuntu,[3] and SuperUser, the performance improvement ratios of TAG over the best baselines on the transductive scenarios (for example, 6.87%, 2.89%, and 8.67% in terms of AP) are generally larger than those on the inductive scenarios (i.e., 5.62%, 2.22%, and 2.99%). This indicates that our proposed TAG is better at predicting the transductive edges between seen nodes on these three datasets than predicting the inductive ones. This is reasonable since the node embeddings in TAG are learned on the training set, thus bringing relatively more improvements on the transductive scenarios where the contained nodes are all seen during training. However, the phenomenon is opposite on the StackOverflow dataset, where the improvement rate on the transductive scenario is 1.71% in terms of AP is smaller than that on the inductive scenario 4.14%. The reason may be that compared to the other three datasets, the training node percentage is larger on StackOverflow as shown in Table III, which may lead TAG to overfit on the training set, thus limiting the performance on the transductive scenarios.

2) Efficiency Comparison: In addition, we further compare the efficiency of TAG with the baselines, the results are presented in Fig. 5. Specifically, in Fig. 5, the x-axis denotes the total time required for the model convergence during training, which is measured using PyTorch 1.8.1 on a Ubuntu 18.04 server with NVIDIA GeForce RTX 3090 GPU with 24-GB memories, the y-axis is the average AUC performance of the transductive and inductive scenarios, and the bubble size indicates the number of model parameters. From Fig. 5, we can observe that TAG has an obviously faster convergence than most baselines except MF, TGAT, and GraphMixer, which all shows unsatisfactory performance as shown in Table IV. We attribute the high efficiency of TAG to the designed ED module for training data selection in Section III-C1.a and the adopted light GNNs for representation learning in Section III-C1.c. In general, TAG is superior above the baselines in terms of both effectiveness and efficiency, indicating its practicability in real-world applications.

B. Ablation Study

To individually verify the effectiveness of different designed components of TAG and compare their contributions to the performance of TAG, we compare TAG with its four variants: 1) w/o ED, which abandons the ED module implemented by (4); 2) w/o RNS, which replaces the RNS strategy of selecting neighbors for information aggregation with the random sampling strategy; 3) w/o DCM, which removes the DCM by deleting the combination of the ED and the RNS; and 4) w/o SCP, which discards the SCP component implemented by the contrastive learning. The experimental results of TAG and the variants in terms of AP and AUC are presented in Table V.

From Table V, we can observe that TAG can outperform four variants in almost all cases, validating the utility of our designed various components. First, comparing the variants w/o DCM and w/o SCP, we can observe that in the transductive scenarios the DCM contributes more, while in the inductive scenarios, the SCP plays a more important role. The difference may be caused by the fact that the neighbor number of the seen nodes is generally large. Thus concentrating on the recent training samples and neighbors can simultaneously well consider the temporal correlations between nodes and avoid the bias introduced by the outdated training sampled as well as neighbors. Differently, in the inductive scenarios where the neighbor number is smaller than that in the transductive scenarios, selecting the recent training samples and neighbors will make the long-term stable node correlations not fully considered, thus the SCP component is relatively more necessary.

In addition, by comparing the variants w/o ED and w/o RNS, we can observe that the RNS plays a more important role in the performance improvement on both transductive and inductive scenarios on all four datasets. We analyze the reason may be that the ED takes the temporal correlations between nodes into consideration by simply removing the useless training samples. However, the RNS can not only focus on the recent correlations between nodes, but also abandon the outdated correlations, thus achieving a relatively

TABLE V
ABLATION STUDY

Task	Method	MathOverflow		StackOverflow		AskUbuntu		SuperUser	
		AP(%)	AUC(%)	AP(%)	AUC(%)	AP(%)	AUC(%)	AP(%)	AUC(%)
Transductive	TAG	**86.42**$_{\pm 0.12}$	**84.16**$_{\pm 0.24}$	**72.45**$_{\pm 0.08}$	**70.61**$_{\pm 0.12}$	**73.64**$_{\pm 0.45}$	**68.17**$_{\pm 0.49}$	**74.94**$_{\pm 0.10}$	**69.70**$_{\pm 0.17}$
	w/o ED	86.19$_{\pm 0.03}$	83.83$_{\pm 0.08}$	71.27$_{\pm 0.18}$	69.40$_{\pm 0.25}$	73.33$_{\pm 0.09}$	68.10$_{\pm 0.06}$	74.86$_{\pm 0.07}$	69.63$_{\pm 0.11}$
	w/o RNS	80.12$_{\pm 0.17}$	78.82$_{\pm 0.20}$	70.47$_{\pm 0.08}$	68.43$_{\pm 0.17}$	67.39$_{\pm 0.15}$	63.49$_{\pm 0.12}$	67.65$_{\pm 0.12}$	64.41$_{\pm 0.14}$
	w/o DCM	80.11$_{\pm 0.09}$	78.82$_{\pm 0.11}$	69.51$_{\pm 0.16}$	67.50$_{\pm 0.16}$	66.25$_{\pm 0.13}$	62.94$_{\pm 0.08}$	67.43$_{\pm 0.25}$	64.16$_{\pm 0.35}$
	w/o SCP	85.31$_{\pm 0.12}$	83.55$_{\pm 0.16}$	71.45$_{\pm 0.11}$	70.04$_{\pm 0.11}$	71.86$_{\pm 0.53}$	68.01$_{\pm 0.53}$	73.55$_{\pm 0.15}$	69.60$_{\pm 0.15}$
Inductive	TAG	**79.44**$_{\pm 0.10}$	**75.69**$_{\pm 0.23}$	**66.13**$_{\pm 0.12}$	**65.67**$_{\pm 0.11}$	69.02$_{\pm 0.73}$	63.56$_{\pm 0.71}$	**67.47**$_{\pm 0.04}$	**62.14**$_{\pm 0.11}$
	w/o ED	79.24$_{\pm 0.08}$	75.44$_{\pm 0.13}$	64.60$_{\pm 0.25}$	64.27$_{\pm 0.24}$	**69.35**$_{\pm 0.09}$	**63.83**$_{\pm 0.16}$	67.39$_{\pm 0.04}$	62.06$_{\pm 0.08}$
	w/o RNS	78.20$_{\pm 0.12}$	75.21$_{\pm 0.20}$	64.92$_{\pm 0.15}$	64.62$_{\pm 0.10}$	68.28$_{\pm 0.69}$	63.07$_{\pm 0.69}$	66.66$_{\pm 0.07}$	61.67$_{\pm 0.07}$
	w/o DCM	77.91$_{\pm 0.11}$	74.90$_{\pm 0.17}$	63.68$_{\pm 0.15}$	63.63$_{\pm 0.12}$	68.65$_{\pm 0.11}$	63.48$_{\pm 0.12}$	66.63$_{\pm 0.01}$	61.65$_{\pm 0.06}$
	w/o SCP	74.62$_{\pm 0.08}$	72.07$_{\pm 0.16}$	60.80$_{\pm 0.07}$	62.23$_{\pm 0.06}$	67.14$_{\pm 0.55}$	62.84$_{\pm 0.54}$	65.56$_{\pm 0.03}$	61.15$_{\pm 0.04}$

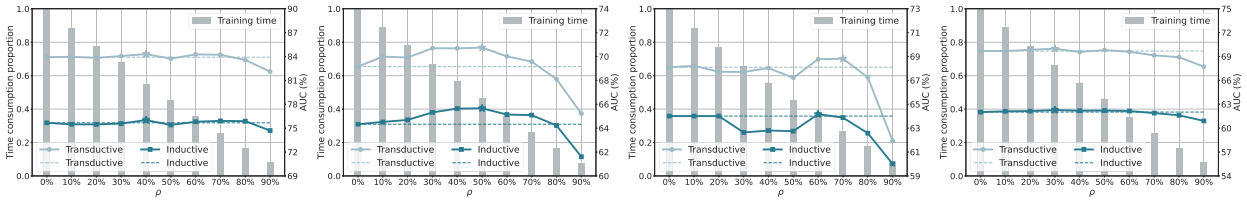

Fig. 6. Impact of parameter ρ on the performance and efficiency of TAG, where the prediction performance and the time consumption are presented by the lines and bars, respectively. (a) Time consumption and AUC performance on MathOverflow. (b) Time consumption and AUC performance on StackOverflow. (c) Time consumption and AUC performance on AskUbuntu. (d) Time consumption and AUC performance on SuperUser.

larger promotion to the prediction performance. In addition, by comparing the performance gap between w/o ED and w/o RNS in the transductive and inductive scenarios, we can find that the gap is more obvious in the transductive scenarios. For example, on MathOverflow, w/o ED exceeds w/o RNS by 7.58% and 6.36% in terms of AP and AUC on the transductive scenario, respectively, while the corresponding rates are 1.33% and 0.31% on the inductive scenario. We analyze the reason that the historical neighbor sequence length is different in two scenarios, where the length is shorter on the inductive scenario with new nodes appearing, thus the improvements of w/o ED by w/o RNS are larger on the transductive scenario.

C. Impact of the ED Ratio

In order to analyze the impact of the ED ratio ρ in the ED module on the performance as well as the computational efficiency of TAG, we search the hyper-parameter ρ in $\{10\%, 20\%, \ldots, 90\%\}$. The performance in terms of AUC and the training time consumption of TAG with various ρ are presented in Fig. 6, where the training time consumption is reported as a relative value to the training time consumption with no ED (i.e., $\rho = 0\%$). In addition, the best performance of TAG in each case is marked with a star symbol.

First, let us focus on the performance changing with various ρ. From Fig. 6, we can observe that on MathOverflow and SuperUser, with ρ increasing, the performance of TAG on both transductive and inductive scenarios first changes gently, achieving the best performance at a ρ of 40% and 30% on MathOverflow and SuperUser, respectively, and then shows the performance decline after $\rho = 80\%$ and 60% on the respective MathOverflow and SuperUser datasets. While on StackOverflow and AskUbuntu, with ρ increasing, the performance of TAG on both transductive and inductive scenarios changes obviously and reaches the peak performance at around a ρ of 50% and 70% on StackOverflow and AskUbuntu, respectively, and then the performance begins to drop. The difference between the gentle and obvious changes in the above two cases may be caused by the training node percentage of those datasets. That is, the large training node percentages on StackOverflow and AskUbuntu make the learned embeddings of the seen nodes have an obvious influence on the future link predictions. Specifically, the datasets with smaller percentages of training node (i.e., 65.85% on MatheOverflow and 63.98% on SuperUser) receive less influence from the parameter ρ, while the impact of ρ on the datasets with larger percentages of training node (i.e., 73.63% on StackOverflow and 71.32% on AskUbuntu) is more obvious.

Moreover, zooming in on the computational efficiency, we can see that with ρ increasing, the time consumption of TAG obviously decreases on all datasets. In addition, the time consumption decrease is close to a linear way, and the slope is less than -1. This indicates that in addition to the time consumption decreased by the reduced training samples, the reduction of the training samples also makes the temporal graph lightweight, further saving the consumed training time. Besides, as shown by the dotted horizontal lines, under the condition without performance decreasing, the time consumption can be saved with a percentage up to 74.45%, 74.01%, 64.55%, and 54.32% on MathOverflow, StackOverflow,[2] AskUbuntu,[3] and SuperUser, respectively. This indicates

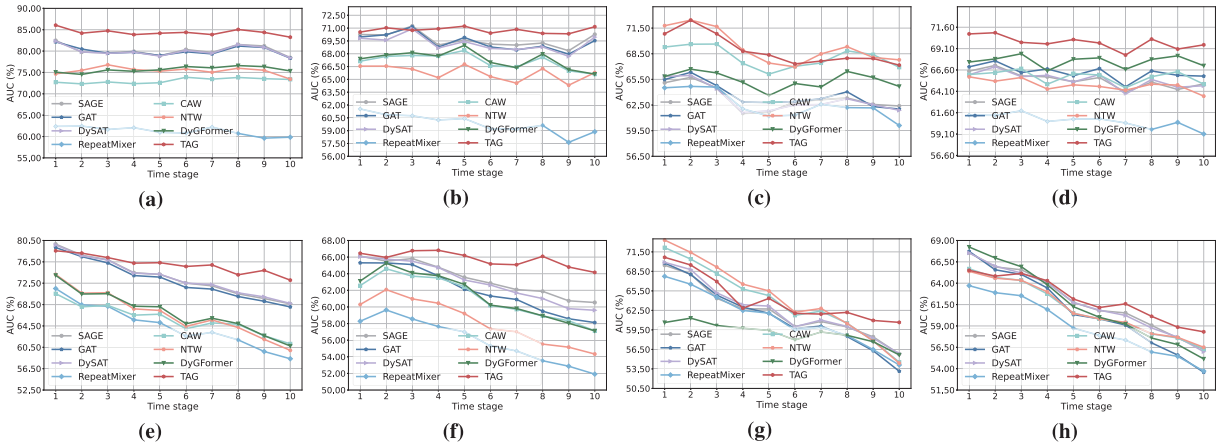

Fig. 7. Performance of TAG on different future time stages. (a) Transductive AUC performance on MathOverflow. (b) Transductive AUC performance on StackOverflow. (c) Transductive AUC performance on AskUbuntu. (d) Transductive AUC performance on SuperUser. (e) Inductive AUC performance on MathOverflow. (f) Inductive AUC performance on StackOverflow. (g) Inductive AUC performance on AskUbuntu. (h) Inductive AUC performance on SuperUser.

that in addition to improving the model performance, the ED module can largely improve the computational efficiency, making it applicable in real-world applications.

D. Performance on Different Future Time Stages

In order to test the ability of TAG as well as the baselines on predicting future links in different time stages, we divide the test data among MathOverflow, StackOverflow,[2] AskUbuntu, and SuperUser into ten-time stages chronologically with the same size of test samples. Here we select the competitive baselines including three embedding-based methods with GNNs, i.e., SAGE, GAT, and DySAT, and four temporal structure-aware methods, i.e., RepeatMixer, CAW, NTW, and DyGFormer for comparison. We present the performance of those models in terms of AUC on different time stages, where the results are presented in Fig. 7.

First, from Fig. 7, we can observe that TAG can achieve the best performance in terms of AUC in most cases on both transductive and inductive scenarios on four datasets. Moreover, we can see that the performance of all models either fluctuates or decreases gently on the transductive scenarios on four datasets, while on the inductive scenarios, the corresponding performance decreases rapidly. The difference is caused by the fact that the representations of the unseen nodes are generated by propagating information from the seen nodes using GNNs. However, in the inductive scenarios, the percentage of the seen nodes contained in the neighbors of the unseen nodes gradually decreases over time, thus the unseen nodes cannot be well represented as time goes on.

Moreover, focusing on the performance changing of TAG on the transductive scenarios presented in Fig. 7(a)–(d), it is observed that the performance of TAG in terms of AUC fluctuates and generally shows a stable trend on MathOverflow and StackOverflow, while the performance of TAG generally decreases on SuperUser and AskUbuntu. The phenomenon may be because the density of the datasets are different, where the densities of AskUbuntu and SuperUser (i.e., both 0.00003) are obviously smaller than those of MathOverflow and StackOverflow (i.e., 0.00063 and 0.00027). The small density in AskUbuntu and SuperUser makes the embeddings of the seen nodes unable to be well learned, thus leading to an unsatisfying performance in the latter time stages.

In addition, from Fig. 7(e)–(h), we can observe that on the inductive scenarios, the performance gap between TAG and the baselines becomes more obvious over time. This indicates that TAG can learn the node embeddings which conform to the dynamic evolution of the temporal networks, making TAG able to generalize to the predictions of future inductive links with large time margins to the current timestamp. Furthermore, there exists a phenomenon that the performance improvements of TAG by the baselines are different on the recent time stages for four datasets. Specifically, the performance gap between TAG and the baselines is obviously larger on MathOverflow and StackOverflow than that on AskUbuntu and SuperUser. We analyze the possible reason that as shown in Fig. 4, the number of inductive edges is much larger than the transductive edges on AskUbuntu and SuperUser, limiting the generalization ability of TAG on predicting future inductive links.

E. Hyper-Parameter Sensitivity

In order to test the sensitivity of TAG to the hyper-parameters, we investigate the impact of the parameters by ranging the parameters α contained in the ED module as well as β introduced in the RNS strategy both in $\{1e^{-8}, 1e^{-7}, 1e^{-6}, 1e^{-5}, 1e^{-4}\}$, and tuning the parameter λ incorporated in the contrastive learning component in $\{1e^{-3}, 1e^{-2}, 1e^{-1}, 0.5, 1\}$. In addition, when the parameters in $\{\alpha, \beta, \lambda\}$ are not tuned, the defaulted values of them are set as $\alpha, \beta = 1e^{-6}$, and $\lambda = 0.5$, respectively. It should be noted that the combination of choosing the defaulted values $\alpha, \beta = 1e^{-6}$, and $\lambda = 0.5$ can achieve the performance which approaches the optimal performance generated by conducting grid search on all four datasets. The results of TAG in terms of AUC with various hyper-parameters are presented in Fig. 8, where Fig. 8(a)–(l) shows the impact of the parameters α, β, and λ, respectively.

1) Hyper Parameter α of the ED Module: From Fig. 8(a)–(d), we can see that with α increasing, the

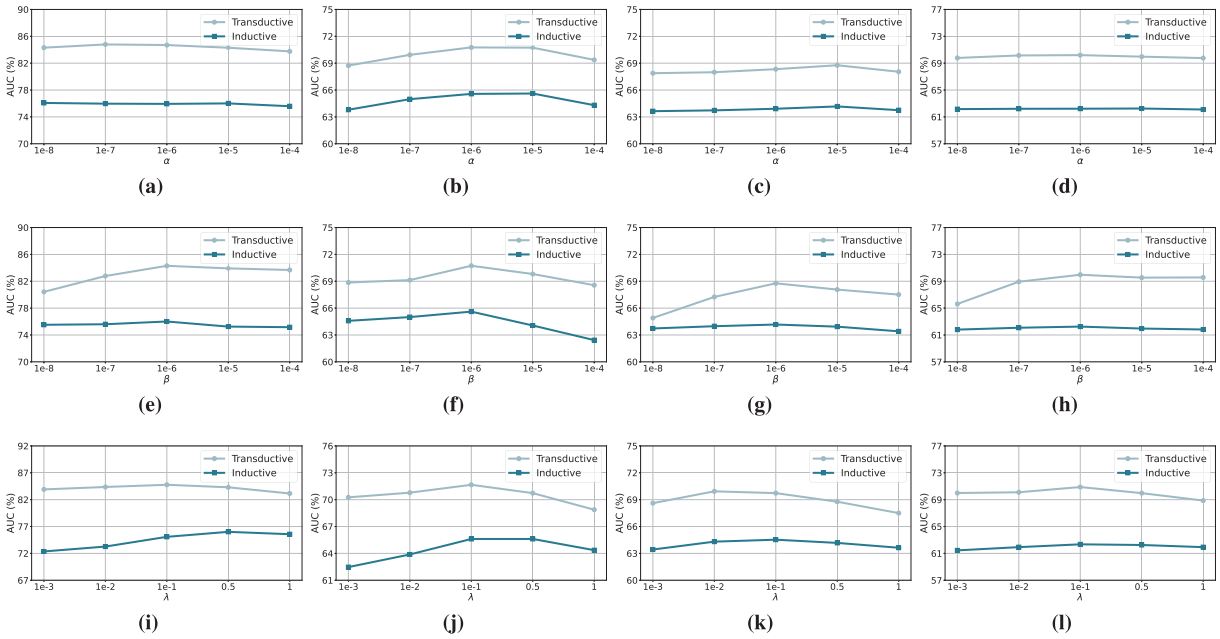

Fig. 8. Hyper parameter sensitivity. (a) AUC performance with various α on MathOverflow. (b) AUC performance with various α on StackOverflow. (c) AUC performance with various α on AskUbuntu. (d) AUC performance with various α on SuperUser. (e) AUC performance with various β on MathOverflow. (f) AUC performance with various β on StackOverflow. (g) AUC performance with various β on AskUbuntu. (h) AUC performance with various β on SuperUser. (i) AUC performance with various λ on MathOverflow. (j) AUC performance with various λ on StackOverflow. (k) AUC performance with various λ on AskUbuntu. (l) AUC performance with various λ on SuperUser.

performance of TAG on both transductive and inductive scenarios on four datasets generally first increases and then shows a decreasing trend. This is due to that a small α makes the outdated training samples unable to be filtered while a large α abandons the long-term stable node correlations. In addition, the influence of α is more obvious on StackOverflow and AskUbuntu than that on MathOverflow and SuperUser. This may be due to that the training node percentages are larger on StackOverflow and AskUbuntu than MathOverflow and SuperUser as shown in Table III, resulting in a more obvious impact of the outdated training data filtering on StackOverflow and AskUbuntu.

2) Hyper Parameter β of the RNS: From Fig. 8(e)–(h), we can observe that with β increasing, the performance of TAG on the transductive scenarios on all four datasets generally first shows an increasing trend and then decreases. This is because β controls the intensity of selecting the recent neighbors, where the larger β is, the more recent the neighbors are selected. The changing trends of the performance on the inductive scenarios are similar to those on the transductive ones, where the difference is that the performance changing on the inductive scenarios is relatively gentler. This is due to the fact that the neighbor number of the unseen nodes is smaller than that of the seen nodes, making the impact of the RNS strategy not obvious on the inductive link prediction.

3) Hyper Parameter λ of the Contrastive Learning: From Fig. 8(i)–(l), we can observe that with λ increasing from $1e^{-3}$ to 1, the performance of TAG on both transductive and inductive scenarios on all four datasets generally shows obvious trends that first increases and then decreases. This is due to the fact that the parameter λ impacts the incorporation of the long-term stable node correlations in the historical interactions on the temporal networks. Specifically, a small λ like $1e^{-3}$ makes TAG lose the ability to consider the long-term stable correlations between nodes while a large λ such as 1 leads TAG unable to focus on the recent dynamic node correlations.

VI. Conclusion

In this article, we propose a TAG learning method for link prediction on temporal networks. Specifically, a theoretical analysis is conducted on the temporal graph learning, guiding our modeling of the temporal dynamics of the graph data. In detail, TAG designs an ED module and adopts an RNS strategy for considering the recent dynamic correlations between nodes; and adopt the contrastive learning for preserving the long-term stable correlations between nodes by introducing additional self-supervisions. Comprehensive experiments conducted on four public temporal networks validate the superiority of TAG above the competitive baselines in terms of AP and AUC.

As to our future work, we would like to adopt the machine unlearning mechanism [63], [64], [65] for achieving effective and efficient data forgetting during the continuous representation learning on temporal networks. In addition, we are also interested in improving the computational efficiency in both training and inference by pruning the temporal networks using the graph structure learning [66], [67].

References

[1] S. Khoshraftar and A. An, "A survey on graph representation learning methods," 2022, *arXiv:2204.01855*.

[2] Z. Wu, S. Pan, F. Chen, G. Long, C. Zhang, and P. S. Yu, "A comprehensive survey on graph neural networks," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 32, no. 1, pp. 4–24, Jan. 2021.

[3] D. Zhu, P. Cui, Z. Zhang, J. Pei, and W. Zhu, "High-order proximity preserved embedding for dynamic networks," *IEEE Trans. Knowl. Data Eng.*, vol. 30, no. 11, pp. 2134–2144, Nov. 2018.

[4] P. Jiao et al., "Temporal network embedding for link prediction via VAE joint attention mechanism," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 33, no. 12, pp. 7400–7413, Dec. 2022.

[5] S. Tang, Z. Meng, and S. Liang, "Dynamic co-embedding model for temporal attributed networks," *IEEE Trans. Neural Netw. Learn. Syst.*, vol.35, no.3, pp.1–15,Mar.2024.

[6] J. Zheng, F. Cai, J. Liu, Y. Ling, and H. Chen, "Multistructure contrastive learning for pretraining event representation," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 35, no. 1, pp. 842–854, Jan. 2024.

[7] J. Wang, K. Ding, L. Hong, H. Liu, and J. Caverlee, "Next-item recommendation with sequential hypergraphs," in *Proc. 43rd Int. ACM SIGIR Conf. Res. Develop. Inf. Retr.*, Jul. 2020, pp. 1101–1110.

[8] H. Zhou, Q. Tan, X. Huang, K. Zhou, and X. Wang, "Temporal augmented graph neural networks for session-based recommendations," in *Proc. 44th Int. ACM SIGIR Conf. Res. Develop. Inf. Retr.*, Jul. 2021, pp. 1798–1802.

[9] C. Wei, J. Liang, B. Bai, and D. Liu, "Dynamic hypergraph learning for collaborative filtering," in *Proc. 31st ACM Int. Conf. Inf. Knowl. Manage.*, Oct. 2022, pp. 2108–2117.

[10] Z. Yang, M. Ding, B. Xu, H. Yang, and J. Tang, "STAM: A spatiotemporal aggregation method for graph neural network-based recommendation," in *Proc. ACM Web Conf.*, Apr. 2022, pp. 3217–3228.

[11] S. Tu and S. Neumann, "A viral marketing-based model for opinion dynamics in online social networks," in *Proc. ACM Web Conf.*, 2022, pp. 1570–1578.

[12] S. Kumar, A. Mallik, A. Khetarpal, and B. S. Panda, "Influence maximization in social networks using graph embedding and graph neural network," *Inf. Sci.*, vol. 607, pp. 1617–1636, Aug. 2022.

[13] S. M. Kazemi et al., "Representation learning for dynamic graphs: A survey," *J. Mach. Learn. Res.*, vol. 21, pp. 70–73, Apr. 2020.

[14] C. D. T. Barros, M. R. F. Mendonça, A. B. Vieira, and A. Ziviani, "A survey on embedding dynamic graphs," *ACM Comput. Surveys*, vol. 55, no. 1, pp. 1–37, Jan. 2023.

[15] Y. Wang, P. Li, C. Bai, and J. Leskovec, "TEDIC: Neural modeling of behavioral patterns in dynamic social interaction networks," in *Proc. Web Conf. (WWW)*, Apr. 2021, pp. 693–705.

[16] Y. Wang et al., "DisenCTR: Dynamic graph-based disentangled representation for click-through rate prediction," in *Proc. 45th Int. ACM SIGIR Conf. Res. Develop. Inf. Retr.*, Jul. 2022, pp. 2314–2318.

[17] M. Qin, C. Zhang, B. Bai, G. Zhang, and D.-Y. Yeung, "High-quality temporal link prediction for weighted dynamic graphs via inductive embedding aggregation," *IEEE Trans. Knowl. Data Eng.*, vol. 35, no. 9, pp. 9378–9393, Sep. 2023.

[18] Y. Wang, Y. Chang, Y. Liu, J. Leskovec, and L. Pan, "Inductive representation learning in temporal networks via causal anonymous walks," in *Proc. Int. Conf. Learn. Represent. (ICLR)*, Jan. 2021, pp. 1–12.

[19] C. Gao, J. Zhu, F. Zhang, Z. Wang, and X. Li, "A novel representation learning for dynamic graphs based on graph convolutional networks," *IEEE Trans. Cybern.*, vol. 53, no. 6, pp. 3599–3612, Jun. 2023.

[20] A. Pareja et al., "EvolveGCN: Evolving graph convolutional networks for dynamic graphs," in *Proc. AAAI Conf. Artif. Intell.*, Apr. 2020, vol. 34, no. 4, pp. 5363–5370.

[21] F. Manessi, A. Rozza, and M. Manzo, "Dynamic graph convolutional networks," *Pattern Recognit.*, vol. 97, Aug. 2019, Art. no. 107000.

[22] E. Hajiramezanali, A. Hasanzadeh, K. R. Narayanan, N. Duffield, M. Zhou, and X. Qian, "Variational graph recurrent neural networks," in *Proc. Conf. Neural Inf. Process. Syst. (NIPS)*, 2019, pp. 10700–10710.

[23] D. Xu, C. Ruan, E. Körpeoğlu, S. Kumar, and K. Achan, "Inductive representation learning on temporal graphs," in *Proc. Int. Conf. Learn. Represent. (ICLR)*, Apr. 2020, pp. 1–21.

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

[25] M. Jin, Y. Li, and S. Pan, "Neural temporal walks: Motif-aware representation learning on continuous-time dynamic graphs," in *Proc. Conf. Neural Inf. Process. Syst. (NeurIPS)*, 2022, pp. 1–12.

[26] Y. Liu, J. Ma, and P. Li, "Neural predicting higher-order patterns in temporal networks," in *Proc. ACM Web Conf. (WWW)*, Apr. 2022, pp. 1340–1351.

[27] S. Kumar, X. Zhang, and J. Leskovec, "Predicting dynamic embedding trajectory in temporal interaction networks," in *Proc. 25th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, Jul. 2019, pp. 1269–1278.

[28] A. Sankar, Y. Wu, L. Gou, W. Zhang, and H. Yang, "DySAT: Deep neural representation learning on dynamic graphs via self-attention networks," in *Proc. 13th Int. Conf. Web Search Data Mining*, Jan. 2020, pp. 519–527.

[29] Y. Zhu, F. Lyu, C. Hu, X. Chen, and X. Liu, "Encoder–decoder architecture for supervised dynamic graph learning: A survey," 2022, *arXiv:2203.10480*.

[30] L. Zhou, Y. Yang, X. Ren, F. Wu, and Y. Zhuang, "Dynamic network embedding by modeling triadic closure process," in *Proc. AAAI Conf. Artif. Intell.*, 2018, vol. 32, no. 1, pp. 571–578.

[31] U. Singer, I. Guy, and K. Radinsky, "Node embedding over temporal graphs," in *Proc. 28th Int. Joint Conf. Artif. Intell. (IJCAI)*, Aug. 2019, pp. 4605–4612.

[32] M. Yang, M. Zhou, M. Kalander, Z. Huang, and I. King, "Discrete-time temporal network embedding via implicit hierarchical learning in hyperbolic space," in *Proc. 27th ACM SIGKDD Conf. Knowl. Discovery Data Mining*, Aug. 2021, pp. 1975–1985.

[33] Y. Zheng, H. Wang, Z. Wei, J. Liu, and S. Wang, "Instant graph neural networks for dynamic graphs," in *Proc. 28th ACM SIGKDD Int. Conf. Knowl. Discov. Data Min.*, 2022, pp. 2605–2615.

[34] M. Gong, S. Ji, Y. Xie, Y. Gao, and A. K. Qin, "Exploring temporal information for dynamic network embedding," *IEEE Trans. Knowl. Data Eng.*, vol. 34, no. 8, pp. 3754–3764, Aug. 2022.

[35] G. H. Nguyen, J. B. Lee, R. A. Rossi, N. K. Ahmed, E. Koh, and S. Kim, "Continuous-time dynamic network embeddings," in *Proc. Companion Web Conf. Web Conf.*, 2018, pp. 969–976.

[36] R. Trivedi, M. Farajtabar, P. Biswal, and H. Zha, "DyRep: Learning representations over dynamic graphs," in *Proc. Int. Conf. Learn. Represent. (ICLR)*, May 2019, pp. 1–17.

[37] C. Yang et al., "Few-shot link prediction in dynamic networks," in *Proc. ACM Int. Conf. Web Search Data Min.*, 2022, pp. 1245–1255.

[38] W. Cong et al., "Do we really need complicated model architectures for temporal networks?," in *Proc. Int. Conf. Learn. Represent. (ICLR)*, Jan. 2023, pp. 1–16.

[39] L. Yu, L. Sun, B. Du, and W. Lv, "Towards better dynamic graph learning: New architecture and unified library," 2023, *arXiv:2303.13047*.

[40] B. Perozzi, R. Al-Rfou, and S. Skiena, "DeepWalk: Online learning of social representations," in *Proc. 20th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, Aug. 2014, pp. 701–710.

[41] A. Vaswani et al., "Attention is all you need," in *Proc. Adv. Neural Inform. Process. Syst. (NIPS)*, 2017, pp. 5998–6008.

[42] L. Jing and Y. Tian, "Self-supervised visual feature learning with deep neural networks: A survey," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 43, no. 11, pp. 4037–4058, Nov. 2021.

[43] X. Liu et al., "Self-supervised learning: Generative or contrastive," *IEEE Trans. Knowl. Data Eng.*, vol. 35, no. 1, pp. 857–876, Jan. 2023.

[44] P. Wang, K. Agarwal, C. Ham, S. Choudhury, and C. K. Reddy, "Self-supervised learning of contextual embeddings for link prediction in heterogeneous networks," in *Proc. Web Conf.*, Ljubljana, Slovenia, Apr. 2021, pp. 2946–2957.

[45] D. Zhang, J. Yin, and P. S. Yu, "Link prediction with contextualized self-supervision," *IEEE Trans. Knowl. Data Eng.*, vol. 35, no. 7, pp. 7138–7151, Jul. 2023.

[46] S. Tian, R. Wu, L. Shi, L. Zhu, and T. Xiong, "Self-supervised representation learning on dynamic graphs," in *Proc. Int. Conf. Inf. Knowl. Manag.*, 2021, pp. 1814–1823.

[47] K.-J. Chen, L. Liu, L. Jiang, and J. Chen, "Self-supervised dynamic graph representation learning via temporal subgraph contrast," *ACM Trans. Knowl. Discovery Data*, vol. 18, no. 1, pp. 1–20, Jan. 2024.

[48] L. Liu, G. Wen, P. Cao, J. Yang, W. Li, and O. R. Zaiane, "Capturing temporal node evolution via self-supervised learning: A new perspective on dynamic graph learning," in *Proc. 17th ACM Int. Conf. Web Search Data Mining*, Mar. 2024, pp. 443–451.

[49] S. Ding, F. Feng, X. He, Y. Liao, J. Shi, and Y. Zhang, "Causal incremental graph convolution for recommender system retraining," 2021, *arXiv:2108.06889*.

[50] X. Hu, K. Tang, C. Miao, X.-S. Hua, and H. Zhang, "Distilling causal effect of data in class-incremental learning," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2021, pp. 3956–3965.

[51] L. Yu, L. Sun, B. Du, and W. Lv, "Towards better dynamic graph learning: New architecture and unified library," in *Proc. Conf. Neural Inf. Process. Syst. (NeurIPS)*, 2023, pp. 1–8.

[52] Y. Tian, Y. Qi, and F. Guo, "FreeDyG: Frequency enhanced continuous-time dynamic graph model for link prediction," in *Proc. Int. Conf. Learn. Represent. (ICLR)*, 2024, pp. 1–18.

[53] T. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," in *Proc. Int. Conf. Learn. Represent. (ICLR)*, Jan. 2016, pp. 1–16.
[54] P. Veličković, G. Cucurull, A. Casanova, A. Romero, P. Lió, and Y. Bengio, "Graph attention networks," in *Proc. Int. Conf. Learn. Represent. (ICLR)*, Jan. 2017, pp. 1–8.
[55] A. Van Den Oord, Y. Li, and O. Vinyals, "Representation learning with contrastive predictive coding," 2018, *arXiv:1807.03748*.
[56] S. Rendle, C. Freudenthaler, Z. Gantner, and L. Schmidt-Thieme, "BPR: Bayesian personalized ranking from implicit feedback," in *Proc. 25th Conf. Uncertainty Artif. Intell. (AUAI)*, pp. 452–461.
[57] W. L. Hamilton, Z. Ying, and J. Leskovec, "Inductive representation learning on large graphs," in *Proc. NIPS*, 2017, pp. 1024–1034.
[58] K. Xu, W. Hu, J. Leskovec, and S. Jegelka, "How powerful are graph neural networks," in *Proc. Int. Conf. Learn. Represent. (ICLR)*, Oct. 2018, pp. 1–7.
[59] T. Zou, Y. Mao, J. Ye, and B. Du, "Repeat-aware neighbor sampling for dynamic graph learning," in *Proc. 30th ACM SIGKDD Conf. Knowl. Discovery Data Mining*, Aug. 2024, pp. 4722–4733.
[60] B. Weisfeiler and A. Lehman, "A reduction of a graph to a canonical form and an algebra arising during this reduction," *Nauchno-Technicheskaya Informatsia*, vol. 2, no. 9, pp. 12–16, 1968.
[61] D. Xu, C. Ruan, E. Körpeoglu, S. Kumar, and K. Achan, "Self-attention with functional time representation learning," in *Proc. Conf. Neural Inf. Process. Syst. (NIPS)*, 2019, pp. 15889–15899.
[62] I. Tolstikhin et al., "MLP-mixer: An all-MLP architecture for vision," in *Proc. 35th Conf. Neural Inf. Process. Syst.*, vol. 34, 2021, pp. 24261–24272.
[63] L. Bourtoule et al., "Machine unlearning," in *Proc. IEEE Symp. Secur. Privacy (SP)*, May 2021, pp. 141–159.
[64] M. Chen, Z. Zhang, T. Wang, M. Backes, M. Humbert, and Y. Zhang, "Graph unlearning," in *Proc. ACM SIGSAC Conf. Comput. Commun. Secur.*, Nov. 2022, pp. 499–513.
[65] J. Wu, Y. Yang, Y. Qian, Y. Sui, X. Wang, and X. He, "GIF: A general graph unlearning strategy via influence function," in *Proc. ACM Web Conf.*, Apr. 2023, pp. 651–661.
[66] Y. Liu, Y. Zheng, D. Zhang, H. Chen, H. Peng, and S. Pan, "Towards unsupervised deep graph structure learning," in *Proc. ACM Web Conf.*, 2022, pp. 1392–1403.
[67] N. Liu, X. Wang, L. Wu, Y. Chen, X. Guo, and C. Shi, "Compact graph structure learning via mutual information compression," in *Proc. ACM Web Conf.*, Apr. 2022, pp. 1601–1610.

Zhiqiang Pan received the B.S. and M.S. degrees from the National University of Defense Technology, Changsha, China, in 2019 and 2021, respectively, where he is currently pursuing the Ph.D. degree with the College of Systems Engineering.

He has published several papers in conference proceedings and journals, such as SIGIR, CIKM, TOIS, IEEE TRANSACTIONS ON NEURAL NETWORKS AND LEARNING SYSTEMS, IRJ, and NCAA. His research interests include graph mining and recommender systems.

Honghui Chen received the Ph.D. degree in operational research from the National University of Defense Technology, Changsha, China, in 2007.

He is a Professor at the National University of Defense Technology. He has published several papers in SIGIR, IPM, and other top journals. His research interests include information systems and information retrieval.

Wanyu Chen received the Ph.D. degree in information systems from the National University of Defense Technology, Changsha, China, in 2021.

She is a Lecturer at the National University of Defense Technology. She has published articles in SIGIR, CIKM, TOIS, and IPM. Her research interests include query suggestion and recommender systems.

Dr. Chen serves as a PC Member for CIKM as well as a reviewer for SIGIR, CIKM, and TOIS.

Fei Cai received the Ph.D. degree in computer science from the University of Amsterdam, Amsterdam, Netherlands, in 2016, under the supervision of Prof. Maarten de Rijke.

He is an Associate Professor at the National University of Defense Technology, Changsha, China. He has published several articles in WWW, SIGIR, CIKM, FnTIR, TOIS, IEEE TRANSACTIONS ON KNOWLEDGE AND DATA ENGINEERING, and IEEE TRANSACTIONS ON NEURAL NETWORKS AND LEARNING SYSTEMS. His research interests include information retrieval and query formulation.

Dr. Cai serves as a PC Member for KDD, WWW, SIGIR, WSDM, CIKM, and RecSys as well as a reviewer for top journals such as IEEE TRANSACTIONS ON KNOWLEDGE AND DATA ENGINEERING, TOIS, TKDD, IEEE TRANSACTIONS ON NEURAL NETWORKS AND LEARNING SYSTEMS, VLDBJ, IPM, and JASIST.

Xinwang Liu (Senior Member, IEEE) received the Ph.D. degree from the National University of Defense Technology (NUDT), Changsha, China, in 2013.

He is a Professor at the School of Computer, National University of Defense Technology. He has authored or co-authored more than 60 peer-reviewed articles, including those in highly regarded journals and conferences, such as IEEE TRANSACTIONS ON PATTERN ANALYSIS AND MACHINE INTELLIGENCE, IEEE TRANSACTIONS ON KNOWLEDGE AND DATA ENGINEERING, IEEE TRANSACTIONS ON IMAGE PROCESSING, IEEE TRANSACTIONS ON NEURAL NETWORKS AND LEARNING SYSTEMS, IEEE TRANSACTIONS ON MULTIMEDIA, IEEE TRANSACTIONS ON INFORMATION FORENSICS AND SECURITY, ICML, NeurIPS, ICCV, CVPR, AAAI, and IJCAI. His current research interests include kernel learning and unsupervised feature learning.

Dr. Liu serves as an Associate Editor for *Information Fusion* journal, IEEE TRANSACTIONS ON CYBERNETICS, and IEEE TRANSACTIONS ON NEURAL NETWORKS AND LEARNING SYSTEMS.