Distance-Aware Learning for Inductive Link Prediction on Temporal Networks

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

***Abstract*— Inductive link prediction on temporal networks aims to predict the future links associated with node(s) unseen in the historical timestamps. Existing methods generate the predictions mainly by learning node representation from the node/edge attributes as well as the network dynamics or by measuring the distance between nodes on the temporal network structure. However, the attribute information is unavailable in many realistic applications and the structure-aware methods highly rely on nodes' common neighbors, which are difficult to accurately detect, especially in sparse temporal networks. Thus, we propose a distance-aware learning (DEAL) approach for inductive link prediction on temporal networks. Specifically, we first design an adaptive sampling method to extract temporal adaptive walks for nodes, increasing the probability of including the common neighbors between nodes. Then, we design a dual-channel distance measuring component, which simultaneously measures the distance between nodes in the embedding space and on the dynamic graph structure for predicting future inductive edges. Extensive experiments are conducted on three public temporal network datasets, i.e., MathOverflow, AskUbuntu, and StackOverflow. The experimental results validate the superiority of DEAL over the state-of-the-art baselines in terms of accuracy, area under the ROC curve (AUC), and average precision (AP), where the improvements are especially obvious in scenarios with only limited data.**

***Index Terms*— Adaptive sampling, distance-aware learning (DEAL), inductive link prediction, temporal network.**

I. Introduction

GRAPH networks offer an effective way to study complex systems by viewing the elements as nodes and regarding their interactions as edges [1], [2], [3], [4]. Moreover, in realistic scenarios such as social media [5], [6], web search [7], [8], etc., the network data are usually time-varying, i.e., nodes and edges continuously evolve, forming the temporal networks [9], [10], [11]. For example, as shown in Fig. 1, new nodes and edges keep appearing in the temporal network, where edges can be separated into *transductive* edges connecting a pair of seen nodes and *inductive* edges associated with unseen node(s). As an important method to predict the future evolution of temporal networks, inductive link prediction is proposed aiming to forecast the appearance of future inductive edges [12], [13], [14].

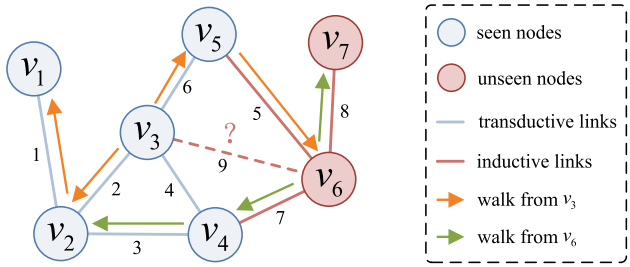

Fig. 1. Temporal network example, where the number below each edge denotes the corresponding timestamp.

Existing methods for inductive link prediction on dynamic graphs mainly focus on learning node representation from the node/edge attributes and the time dynamics [13], [15], [16], [17]. For instance, the temporal attention network is proposed to simultaneously consider both the edge attribute information and the timespan between interactions [13]. However, the application scope of such methods is limited, since the node/edge attributes are missing or incomplete in many realistic scenarios due to the issues of privacy protection, data access restriction, etc., [18], [19], [20], [21]. This results in the node representation unable to be learned from the attribute information. Recently, temporal anonymous walk [22], [23] is proposed to measure the distance between nodes on the dynamic graph structure for inductive link prediction [14]. However, it highly relies on the common neighbors between nodes, suffering from two main disadvantages, especially in the sparse temporal networks: On the one hand, it can merely explicitly model the connectivity between nodes on extracted walks, failing to consider the closely connected but not sampled node pairs, such as v_6 and v_3 on walk $v_6 \rightarrow v_4 \rightarrow v_3$ in Fig. 1. On the other hand, previous methods sample neighbors either randomly [13] or by simply selecting recent neighbors [14], [24], failing to accurately locate the common neighbors between nodes.

To address the above issues, we propose a distance-aware learning (DEAL) approach for inductive link prediction on temporal networks. Specifically, we first design an adaptive

Manuscript received 30 August 2022; revised 21 August 2023; accepted 18 October 2023. This work was supported in part by the National Natural Science Foundation of China under Grant 62372458 and in part by the Postgraduate Scientific Research Innovation Project of Hunan Province under Grant CX20230083. *(Corresponding author: Fei Cai.)*

Zhiqiang Pan, Fei Cai, and Honghui Chen 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; caifei08@nudt.edu.cn; chenhonghui@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).

Color versions of one or more figures in this article are available at https://doi.org/10.1109/TNNLS.2023.3328924.

Digital Object Identifier 10.1109/TNNLS.2023.3328924

sampling method to dynamically select historical interacted nodes as neighbors for extracting the temporal adaptive walks for nodes. Then, we design a dual-channel distance measuring component, in which we respectively generate the embedding- and structure-based prediction scores between source node and target node as follows.

1) Implicitly measuring the distance in the embedding space using the similarity of their representations generated by multi-hop information propagation.
2) Explicitly measuring the distance on the dynamic graph structure by modeling their connectivity using the anonymous distance encoding.

Finally, the embedding- and structure-based scores are inputted into a multilayer perception to output the final prediction score, i.e., the probability of generating a temporal link between source node and target node.

To verify the effectiveness of our proposal, we conduct experiments on three public temporal network datasets, i.e., MathOverflow, AskUbuntu, and StackOverflow. The experimental results demonstrate that the proposed DEAL model can achieve the state-of-the-art performance in terms of accuracy, area under the ROC curve (AUC), and average precision (AP).

Our contributions in this article can be summarized as follows.

1) We propose DEAL for inductive link prediction on temporal networks. To the best of our knowledge, we are the first to simultaneously measure the distance between source node and target node in the embedding space implicitly and on the dynamic graph structure explicitly.
2) We propose an adaptive sampling method to dynamically select neighbors for extracting temporal adaptive walks, improving the probability of capturing the common neighbors of source node and target node, which thus enhances the representation learning in the embedding space and on the graph structure.
3) We compare the performance of DEAL against the state-of-the-art baselines on three public temporal network datasets, and the experimental results demonstrate the superiority of DEAL over the baselines in terms of accuracy, AUC, and AP.

II. Related Work

We review the previous works from two angles, i.e., static graph representation learning and dynamic graph representation learning.

A. Static Graph Representation Learning

Static graph representation learning models can be divided into non-graph neural network (GNN)-based methods and GNN-based methods [25].

1) Non-GNN-Based Methods: Factorization-based methods are typical early works which utilize the matrix factorization to learn the embeddings of nodes from the adjacent matrix of graphs [26], [27], [28]. Then, borrowing the spirits from [29], random walk-based methods such as DeepWalk [30] and Node2vec [31] are proposed, which first extract node's random walks and then train the node embeddings based on the assumption that the nodes simultaneously occurring on one walk should be similar. Moreover, Tang et al. [32] propose the large-scale information network embedding (LINE), which takes the first- and second-order proximities of nodes into consideration by minimizing two separate corresponding loss functions. In addition, Wang et al. [33] design an autoencoder-based method, i.e., structural deep neural embedding (SDNE), to preserve the proximities in the graph by reconstructing the adjacent matrix. Furthermore, Kipf and Welling [34] adapt the variational autoencoder [35] to graphs and design variational graph autoencoder (VGAE) to learn from the adjacent matrix and node attributes for generating the node vectors.

2) GNN Based Methods: Recently, deep learning methods have been widely applied to graphs, leading to various GNNs on static graphs [36], [37], [38], [39]. For example, borrowing the mathematical concepts from graph signal processing, Kipf and Welling [40] design the graph convolutional network (GCN), which utilizes a localized first-order approximation of spectral graph convolutions for message passing. Then, the spatial based method GraphSAGE [41] is proposed to generate node representation by sampling and aggregating features from node's multi-hop neighbors. Moreover, Velickovic et al. [37] design graph attention network (GAT), which propagates information in an attentive way by assigning different weights to neighbors. In addition, Li et al. [42] propose a gated graph neural network (GGNN), which adopts the gated recurrent unit (GRU) to combine the information at various hops during propagation. Furthermore, Wang et al. [43] propose multi-hop attention GNN, which increases the receptive field on the basis of GAT [37], and You et al. [44] design identity-aware GNN (ID-GNN) to improve previous GNNs by inductively considering nodes' identities in information propagation.

However, both the above-mentioned non-GNN- and GNN based methods are designed for static graphs, which cannot adapt to the dynamic scenarios where nodes and edges are time-varying, failing to simultaneously take both the graph structure information and the time dynamics into consideration on temporal networks.

B. Dynamic Graph Representation Learning

According to the involved scenarios, dynamic graph representation learning models can be separated into transductive dynamic methods and inductive dynamic methods.

1) Transductive Dynamic Methods: Transductive dynamic methods focus on scenarios where nodes are all seen during the training stage [16], [45], [46], [47]. For example, Goyal et al. [48] adopt two fully connected layers for encoding and decoding to generate the temporal network embeddings, and apply LSTM to capture the dynamic structure evolution, so as to predict its future structure. Moreover, Kumar et al. [46] propose a coupled recurrent neural network (RNN) model to learn the node embeddings and predict their trajectory for future interaction prediction. In addition, Hajiramezanali et al. [49] design a variational graph RNN (VGRNN), which adopts high-level latent random variables to increase the expressive power and the ability of modeling the representation uncertainty. Pareja et al. [50] adopt RNNs

to evolve the GCN parameters for capturing the time dynamics between different graph snapshots and further estimate the future GCN parameters for prediction.

2) Inductive Dynamic Methods: Researches on dynamic graph representation learning in inductive scenarios are widely investigated in recent years [13], [14], [24], [51]. For example, Trivedi et al. [51] consider both the association and communication processes in the network dynamics by a two-time scale deep temporal point process approach. Moreover, Xu et al. [13] design a temporal graph attention (TGAT) layer to consider the graph structure as well as the temporal information modeled by a time encoding function based on the Bochner's theorem [52]. Furthermore, Rossi et al. [24] propose the temporal graph network (TGN), which adopts memory networks [53] and graph-based operators to generate dynamic network embeddings. In addition, the most closely related work to our proposal is CAW-N [14], which proposes to utilize the temporal anonymous walk to generate the relative identity embedding between nodes for measuring their distance on the temporal network structure, so as to predict the future links in an inductive way.

However, the transductive dynamic methods can merely be applied to the cases where nodes are all seen during training, which cannot generalize to the inductive scenarios [13], [14]. Moreover, the existing inductive dynamic models generally rely on the attribute information [13], [24], [51] or abundant interactions between nodes [14] to generate predictions, where the generalization ability to various scenarios is unsatisfying, i.e., failing to make correct predictions when the temporal network is non-attributed or sparse.

III. Preliminaries

A. Problem Statement

Inductive link prediction aims to predict the appearance of future edges which are associated with new node(s) not appearing in the training process. Let $\mathcal{G} = \{\mathcal{V}, \mathcal{E}\}$ denote the temporal network, where $\mathcal{V}$ and $\mathcal{E}$ are the contained nodes and edges, respectively, which are both time-varying. Each edge $(v_i, v_j, t_{ij}) \in \mathcal{E}$ indicates that a temporal edge connects nodes v_i and v_j at timestamp t_{ij}. In addition, given a temporal network, following the setting in previous works [13], [14], we first separate the temporal graph data chronologically into the training set, the validation set and the test set, respectively. Then, let V_s denote the seen nodes appearing in the training stage and V_u indicate the unseen nodes which only appear in the future timestamps. Moreover, we denote the ending timestamp of the training edges as the current timestamp t_{cur}, which is earlier than the timestamp of each edge in the validation and test sets. Finally, the inductive link prediction task can be formulated as predicting the probability of each edge $\{(v_i, v_j, t_{ij}) | v_i \in V_u || v_j \in V_u, t_{ij} > t_{cur}\}$ in the validation and test sets.

B. Existing Pipeline

The pipeline of existing methods mainly includes three components, i.e., temporal walk extraction, distance measuring as well as prediction and optimization.

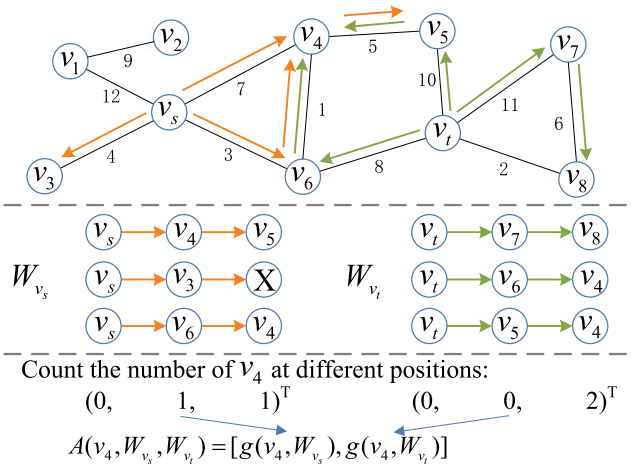

Fig. 2. Example of anonymous distance encoding, where X denotes the padding neighbor.

1) Temporal Walk Extraction: Specifically, give the future edge denoted as (v_s, v_t, t_{st}) to predict, the temporal walks of the source node v_s and the target node v_t are first extracted from the temporal network structure.

2) Distance Measuring: Then, the extracted temporal walks of v_s and v_t are utilized for the node encoding to measure the distance between v_s and v_t, where the methods include modeling the temporal signals [13], further considering the explicit distance on the graph structure [14], etc.

3) Prediction and Optimization: Finally, the representation of v_s and v_t are combined to generate the probability of forming a future link between source node v_s and target node v_t, and then the model is optimized to learn the contained learnable parameters through the back-propagation through time (BPTT) algorithm [54].

C. Anonymous Distance Encoding

Anonymous distance encoding proposed in [14] aims to generate a vector for measuring the distance between source node v_s and target node v_t associated with the inductive edge (v_s, v_t, t_{st}) on the graph structure. Specifically, let the neighbor walks of v_s and v_t be $W_{v_s} = \{w_1^s, \ldots, w_M^s\}$ and $W_{v_t} = \{w_1^t, \ldots, w_M^t\}$, respectively, where each walk w consists of a sequence of nodes backtracking time, such $v_s \rightarrow v_1 \rightarrow v_2$ in Fig. 2, and M is the number of walks. Then, we can generate the anonymous distance encoding vector for each neighbor v_i appearing in $W_{v_s} \cup W_{v_t}$ as follows:

$$\mathbf{a}_i = A(v_i, W_{v_s}, W_{v_t}) = \left[g(v_i, W_{v_s}), g(v_i, W_{v_t})\right] \quad (1)$$

where $\mathbf{a}_i \in \mathbb{R}^{2m}$ is the distance vector for v_i, m is the walk length, and $g(v_i, W_{v_x}) \in \mathbb{R}^m$ indicates the times neighbor v_i appears at certain positions in W_{v_x}, which can be formulated as follows:

$$g(v_i, W_{v_x})[j] = |\{w_{kj} | w_k \in W_{v_x}, v_i = w_{kj}\}| \quad \forall j = 1, \ldots, m \quad (2)$$

where w_{kj} denotes the jth node on walk w_k.

Specifically, as the example in Fig. 2 shows, $g(v_i, W_{v_s})$ measures the distance of v_i at different positions to the source

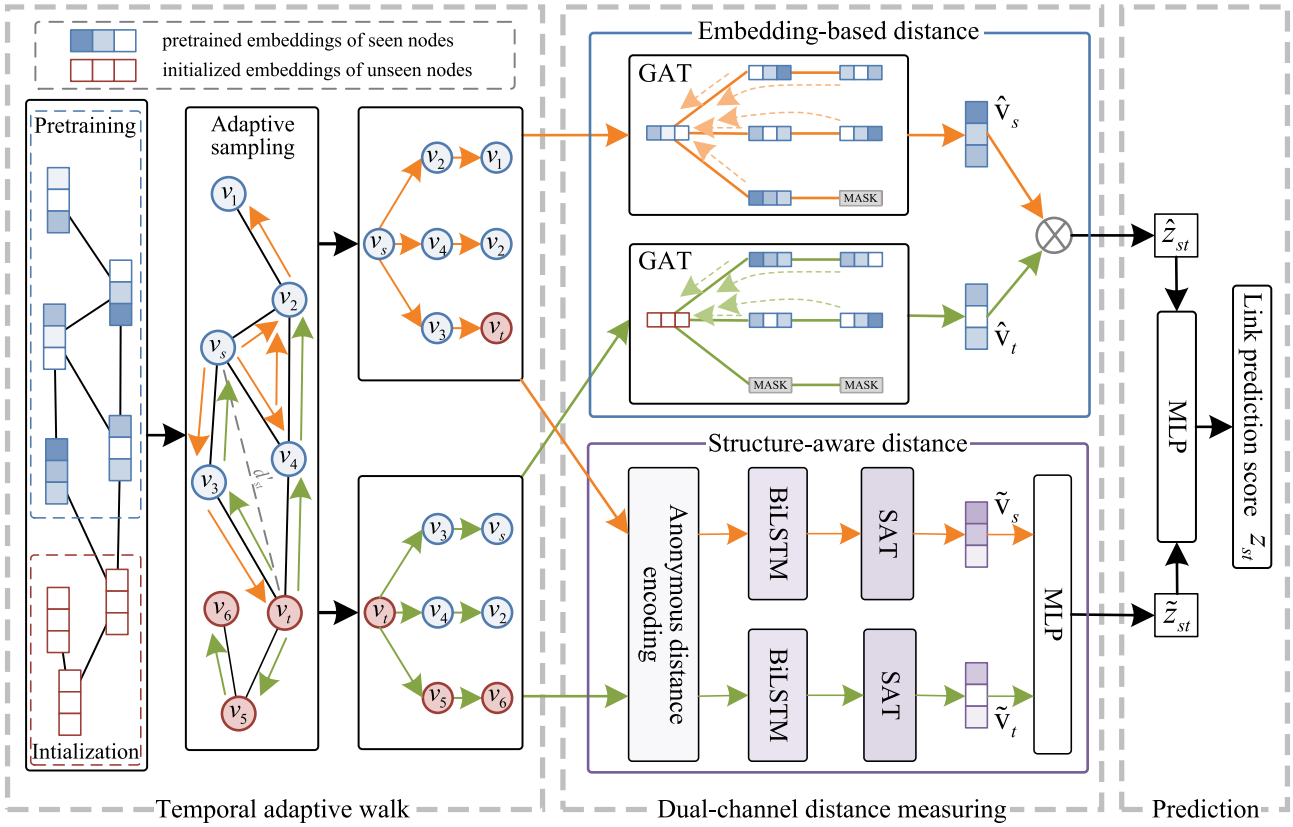

Fig. 3. Framework of DEAL.

node v_s while $g(v_i, W_{v_t})$ reflects the distance between v_i and the target node v_t. By concatenating $g(v_i, W_{v_s})$ and $g(v_i, W_{v_t})$, the anonymous distance encoding generates a vector $\mathbf{a}_i \in \mathbb{R}^{2m}$ for each neighbor $v_i \in W_{v_s} \cup W_{v_t}$ to measure the distance between the source node v_s and the target node v_t by viewing neighbor v_i as an intermediate node. Since the above distance encoding process is conducted in an anonymous way, i.e., without the node IDs, this is applicable to the inductive scenario on temporal networks [14], [22].

IV. Approach

Our proposed DEAL method consists of three main components, i.e., temporal adaptive walk (see Section IV-A), dual-channel distance measuring (see Section IV-B), and prediction (see Section IV-C). The workflow of DEAL is plotted in Fig. 3. Give the source node v_s and target node v_t, we first design an adaptive sampling method to extract their temporal adaptive walks using the pretrained embeddings of seen nodes. Then, we measure the distance between v_s and v_t in the embedding space and on the dynamic graph structure to generate the corresponding prediction scores, respectively. Finally, the embedding- and structure-based scores are inputted to a fully connected layer for predicting the probability of forming a future link between v_s and v_t.

A. Temporal Adaptive Walk

Different from existing works which either randomly extract the neighbor walks for nodes on temporal networks [13] or simply select the most recent neighbors [14], [24], we propose to extract the temporal adaptive walks by dynamically combining the former two methods, so as to improve the probability of including the common neighbors of source node and target node. The temporal adaptive walk extraction contains two main stages, i.e., seen node pretraining and adaptive sampling.

In addition, it is worth noting that though an intuitive way that performing multiple neighbor samplings can indeed increase the probability of selecting the common neighbors of the node pair, this may suffer from two main disadvantages.

1) More unrelated neighbors will also be introduced as bias, disturbing the accurate estimation of the distance between source node and target node, making it unable to guarantee the performance increasing [14].
2) It requires much more time to conduct the multiple samplings as well as the following distance encoding, information propagation, etc. This largely decreases the computational efficiency of the link predictor, making it hard to be applied in realistic scenarios.

1) Seen Node Pretraining: We first pretrain the embeddings of the seen nodes appearing in the training set. Specifically, given the current temporal edge denoted as (v_s, v_t, t_{st}) where both v_s and v_t are seen nodes, we first extract the random walks for source node v_s and target node v_t to obtain their corresponding multi-hop neighbors. Specifically, we collect the linked nodes by backtracking the edge timestamps, thus forming each random walk denoted as follows:

$$w = \{(v_1, v_2, \ldots, v_m) | (v_{i-1}, v_i, t_i) \in \mathcal{E} \quad \forall i = 2, 3, \ldots, m\} \tag{3}$$

where m is the walk length, and $t_2 > t_3 > \cdots > t_m$, which indicates that nodes on the same walk should be in the chronological order. Moreover, v_1 is the starting point of the walk, i.e., v_s or v_t in the current edge (v_s, v_t, t_{st}), and $(v_{i-1}, v_i, t_i) \in \mathcal{E}$ is a temporal edge indicating that node v_{i-1} interacts with v_i at timestamp t_i. Moreover, we collect M many m-step temporal random walks for source node v_s and target node v_t, where the number of walks and the length of each walk are M and m, respectively. The collected walks of v_s and v_t can be denoted as $\bar{W}_{v_s} = \{\bar{w}_1^s, \bar{w}_2^s, \ldots, \bar{w}_M^s\}$ and $\bar{W}_{v_t} = \{\bar{w}_1^t, \bar{w}_2^t, \ldots, \bar{w}_M^t\}$, respectively.

After that, taking the source node v_s as an example, we first group the neighbors in $\bar{W}_{v_s}$ according to the hop numbers, where the lth hop neighbors can be denoted as $\bar{N}_{v_s}^l$. Then, we aggregate information from multi-hop neighbors to update the vector of v_s

$$\begin{aligned} \bar{\mathbf{v}}_s^l &= \mathcal{F}_{\text{mean-pool}}\left(\left[\mathbf{v}_1^l, \mathbf{v}_2^l, \ldots, \mathbf{v}_M^l\right]\right) \\ \bar{\mathbf{v}}_s &= \mathcal{F}_{\text{mean-pool}}\left(\left[\mathbf{v}_s, \bar{\mathbf{v}}_s^1, \ldots, \bar{\mathbf{v}}_s^{m-1}\right]\right) \end{aligned} \tag{4}$$

where $\mathbf{v}_\tau^l \in \mathbb{R}^d$ is the embedding of neighbor v_τ in the lth hop neighbors $\bar{N}_{v_s}^l = \{v_1, v_2, \ldots, v_M\}$ and $\mathbf{v}_s$ is the embedding of source node v_s, which are initialized by an embedding layer, and d is the embedding dimension. Moreover, $\bar{\mathbf{v}}_s^l$ is the aggregated representation of the l-th hop neighbors, and $\bar{\mathbf{v}}_s \in \mathbb{R}^d$ is the final updated representation of v_s. Similarly, the target node representation can be generated as $\bar{\mathbf{v}}_t \in \mathbb{R}^d$. Here we adopt the mean pooling to avoid introducing additional trainable parameters, thus focusing on measuring the distance between the node embeddings.

Then, we measure the distance between the representation of the source node v_s and target node v_t generated by (4), i.e., $\bar{\mathbf{v}}_s$ and $\bar{\mathbf{v}}_t$, respectively, using the L2 normalization

$$d_{st} = ||\bar{\mathbf{v}}_s - \bar{\mathbf{v}}_t||_2. \tag{5}$$

Finally, we adopt the cross-entropy to train the node embeddings

$$L_{\text{pre}} = \sum_{(v_s, v_t, t_{st}) \in \mathcal{E}} -\log(\sigma(d_{st})) - \log(\sigma(1 - d_{sn})) \tag{6}$$

where d_{st} is the positive predicted score between the source node v_s and the target node v_t, d_{sn} is the negative predicted distance score between the source node v_s and a sampled negative target node v_n, and σ is the sigmoid function which transforms the predicted scores into a value between 0 and 1. By optimizing the pretraining loss L_{pre}, we can learn close embeddings for positive node pairs while separating the embeddings of negative node pairs in the embedding space.

2) Adaptive Sampling: After pretraining, we design an adaptive neighbor sampling method in the inductive scenarios on temporal networks. First, we follow a similar procedure to the pretraining stage for generating the representations of the source node v_s and target node v_t associated with the inductive link (v_s, v_t, t_{st}). Specifically, we aggregate information from their corresponding multi-hop neighbors on the randomly sampled walks. The only distinction is that merely information from the seen neighbors (i.e., the neighbors appearing in the training set) are propagated, while the unseen neighbors are omitted since the embeddings of them are initialized without well learning in pretraining. Finally, the representations of v_s and v_t can be respectively generated as $\bar{\mathbf{v}}_s'$ and $\bar{\mathbf{v}}_t'$, which are then inputted into (5) to generate d_{st}' for measuring the distance between v_s and v_t in the inductive scenarios.

Then, for the node pair v_s and v_t associated with the inductive link which has a large distance, the historical neighbors of v_s or v_t may be recently appearing nodes which are unseen during training, such as node v_5 in Fig. 3 whose neighbors v_t and v_6 are both unseen nodes. In this case, we would like to utilize the recent sampling [14] to sample recent neighbors for improving the probability of capturing the common neighbors between v_s and v_t. Specifically, the recenter the neighbor is interacted with v_s or v_t, the larger the probability of it to be sampled is, where we will formulate this in detail in (7). In contrast, for the node pair with a small distance, their historical neighbors tend to be similar seen nodes, thus we would like to adopt the random sampling to sample from the full temporal scale of the neighbors of v_s and v_t to avoid omitting their early common neighbors.

Inspired by the above intuition, we design an adaptive sampling method to select neighbors for extracting adaptive walks by relying on the distance d_{st}' between v_s and v_t associated with the inductive edge (v_s, v_t, t_{st}). Specifically, the probability of neighbor v_i to be sampled during the temporal adaptive walk extraction process can be formulated as follows:

$$\mathcal{T}(v_i) = \begin{cases} \dfrac{e^{\lambda(t_i - t_{i-1})}}{\sum_{v_j \in N_{v_{i-1}}^o} e^{\lambda(t_j - t_{i-1})}}, & d_{st}' > d_{\text{threshold}} \\ \dfrac{1}{|N_{v_{i-1}}^o|}, & d_{st}' \le d_{\text{threshold}} \end{cases} \tag{7}$$

where $N_{v_{i-1}}^o$ is the historical interacted nodes of the last step on the adaptive walk, i.e., v_{i-1}, and λ is a hyperparameter for determining the intensity of recent sampling, and the recent sampling is degenerated to uniform sampling when λ equals to 0. Moreover, $d_{\text{threshold}}$ is a trade-off parameter controlling the threshold of choosing random sampling or recent sampling.

B. Dual-Channel Distance Measuring

1) Embedding-Based Distance Measuring: After obtaining the temporal adaptive walks, we propagate information from the seen neighbors of source node and target node to learn their corresponding representations for distance measuring in the embedding space. Taking the source node v_s as an example, its multi-hop seen neighbors on the adaptive walks can be denoted as $\hat{N}_{v_s} = \{\hat{N}_{v_s}^1, \hat{N}_{v_s}^2, \ldots, \hat{N}_{v_s}^{m-1}\}$, where $\hat{N}_{v_s}^l$ is the l-th hop seen neighbors.

First, we generate a d-dimensional embedding vector $\mathbf{v}_i \in \mathbb{R}^d$ for each node v_i using an embedding layer. Then, for the source node v_s with the l-th hop seen neighbors $\hat{N}_{v_s}^l$, we adopt the GAT [37] to aggregate the embeddings of neighbors in $\hat{N}_{v_s}^l$ for updating the representation of v_s. Specifically, we first learn the weight of each seen neighbor

$$e_\tau^l = \text{LeakyRelu}\left(\hat{\mathbf{W}}_{\text{att}}^{\text{T}}\left[\hat{\mathbf{W}}_q \mathbf{v}_s, \hat{\mathbf{W}}_k \mathbf{v}_\tau^l\right]\right) \tag{8}$$

where $\mathbf{v}_\tau^l$ is the embeddings of a seen neighbor $v_\tau^l \in \hat{N}_{v_s}^l$ of node v_s at the l-hop, $\hat{\mathbf{W}}_{\text{att}} \in \mathbb{R}^{1 \times d}$ and $\hat{\mathbf{W}}_q, \hat{\mathbf{W}}_k \in \mathbb{R}^{d \times d}$ are trainable parameters, LeakyRelu is the activation function, and

[,] denotes the concatenation operation. Then, we normalize the attention weights using a softmax function

$$\alpha_\tau^l = \text{Softmax}\left(e_\tau^l\right) = \frac{\exp\left(e_\tau^l\right)}{\sum_{v_i^l \in N_{v_s}^l} \exp\left(e_i^l\right)}. \tag{9}$$

After that, the embeddings of node v_s's seen neighbors at the l-hop (i.e., $\hat{N}_{v_s}^l$) are combined according to the attention scores

$$\hat{\mathbf{v}}_s^l = \sigma\left(\sum_{v_\tau^l \in N_{v_s}^l} \alpha_\tau^l \hat{\mathbf{W}}_v \mathbf{v}_\tau^l\right) \tag{10}$$

where $\hat{\mathbf{W}}_v \in \mathbb{R}^{d\times d}$ indicates the learnable parameters.

Finally, we adopt the mean pooling to combine the embedding of v_s and the aggregated representations of seen neighbors at different hops for generating the final representation of v_s as follows:

$$\hat{\mathbf{v}}_s = \mathcal{F}_{\text{mean-pool}}\left(\mathbf{v}_s, \hat{\mathbf{v}}_s^1, \ldots, \hat{\mathbf{v}}_s^{m-1}\right) \tag{11}$$

where $\mathbf{v}_s \in \mathbb{R}^d$ is the initialized embedding of v_s, $\mathcal{F}_{\text{mean-pool}}$ indicates the mean-pooling aggregation function. At last, we can obtain the final representation of v_s as $\hat{\mathbf{v}}_s \in \mathbb{R}^d$. Similarly, the representation $\hat{\mathbf{v}}_t \in \mathbb{R}^d$ can also be generated for the target node v_t following the same procedure.

It is worth noting that in the inductive link prediction task, the source node v_s and target node v_t are both seen nodes during training, where the embeddings can be learned by back propagation; however, v_s or v_t associated with the inductive edges may be unseen during test, whose embedding cannot be well trained. In our proposal, benefiting from the multi-hop information propagation, the unseen nodes v_s or v_t in the edge (v_s, v_t, t_{st}) can be linked to the seen neighbors, thus obtaining valuable representations for unseen nodes in (v_s, v_t, t_{st}) to generate accurate predictions.

Finally, we multiply the updated representations of the source node v_s and target node v_t as follows:

$$\hat{z}_{st} = \hat{\mathbf{v}}_s^\top \hat{\mathbf{v}}_t \tag{12}$$

where the generated prediction score $\hat{z}_{st}$ measures the distance between v_s and v_t in the embedding space.

2) Structure-Aware Distance Measuring: Besides measuring the embedding-based distance, we propose to generate the structure-aware representations of v_s and v_t for measuring the distance on the dynamic graph structure.

Specifically, given the adaptive walks of v_s and v_t as $\hat{W}_{v_s} = \{\hat{w}_1^s, \hat{w}_2^s, \ldots, \hat{w}_M^s\}$ and $\hat{W}_{v_t} = \{\hat{w}_1^t, \hat{w}_2^t, \ldots, \hat{w}_M^t\}$, we first generate a distance encoding vector for each neighbor $v_i \in \hat{W}_{v_s} \cup \hat{W}_{v_s}$ as $\mathbf{a}_i \in \mathbb{R}^{2m}$ by the anonymous distance encoding using (1). Then, taking the source node v_s as an example, we input the distance vector of each neighbor v_i into a multi-layer perceptron (MLP) to obtaining its distance-aware representation $\mathbf{s}_i \in \mathbb{R}^d$

$$\mathbf{s}_i = \text{MLP}(\mathbf{a}_i) = \mathbf{W}_2(\sigma(\mathbf{W}_1 \mathbf{a}_i)) \tag{13}$$

where $\mathbf{W}_1 \in \mathbb{R}^{d\times 2m}$ and $\mathbf{W}_2 \in \mathbb{R}^{d\times d}$ are trainable parameters in MLP, and σ denotes the ReLu function.

In addition, to take the time dynamics of graph structure into consideration, following [13], [14], we adopt the existing method random Fourier features proposed in [52] for the time encoding, which may approach any positive definite kernels according to the Bochner's theorem [55], thus is suitable for processing the continuous time interval between edges on the temporal network. The time encoding can be formulated as follows:

$$\mathcal{F}_{\text{time}}(t) = [\cos(\omega_1 t), \sin(\omega_1 t), \ldots, \cos(\omega_d t), \sin(\omega_d t)] \tag{14}$$

where $[\omega_1, \omega_2, \ldots, \omega_d]$ are learnable parameters. Thus, given the current timestamp t_{st} in edge (v_s, v_t, t_{st}) and the prior timestamp t_i in edge (v_{i-1}, v_i, t_i), we can obtain the time encoding vector $\mathbf{t}_i \in \mathbb{R}^{2d}$ as follows:

$$\mathbf{t}_i = \mathcal{F}_{\text{time}}(\Delta t_i) = \mathcal{F}_{\text{time}}(t_{st} - t_i). \tag{15}$$

Then, we generate the temporal distance-aware representation of neighbor v_i as $\mathbf{h}_i \in \mathbb{R}^{3d}$ by concatenating its distance-aware representation and time encoding vector as follows:

$$\mathbf{h}_i = [\mathbf{s}_i, \mathbf{t}_i]. \tag{16}$$

After representing the neighbors on each walk, we adopt the bi-directional long short-term memory (BiLSTM) to model each individual walk as follows:

$$\begin{aligned}
\overrightarrow{\mathbf{w}_{i,j}} &= \overrightarrow{\text{BiLSTM}}\left(\overrightarrow{\mathbf{w}_{i,j-1}}, \mathbf{h}_{i,j}\right) \\
\overleftarrow{\mathbf{w}_{i,j}} &= \overleftarrow{\text{BiLSTM}}\left(\overleftarrow{\mathbf{w}_{i,j-1}}, \mathbf{h}_{i,j}\right) \\
\tilde{\mathbf{w}}_{i,j} &= \overrightarrow{\mathbf{w}_{i,j}} + \overleftarrow{\mathbf{w}_{i,j}}
\end{aligned} \tag{17}$$

where $\overrightarrow{\mathbf{w}_{i,j}} \in \mathbb{R}^d$ and $\overleftarrow{\mathbf{w}_{i,j}} \in \mathbb{R}^d$ are the hidden state of walk $\hat{w}_i$ in the forward and backward passes, respectively, $\hat{w}_i$ is the i-th adaptive walk of node v_s, and $\mathbf{h}_{i,j}$ is the temporal distance-aware representation of the j-th neighbor on walk $\hat{w}_i$. Finally, we adopt the last state of BiLSTM as the representation of $\hat{w}_i$, i.e., $\tilde{\mathbf{w}}_i = \tilde{\mathbf{w}}_{i,m}$.

It is worth noting that our proposal is agnostic to the walk encoder. That is, other alternatives such as the Transformer [56] can be also adopted to replace BiLSTM in our framework. Here we adopt BiLSTM as following the state-of-the-art baseline CAW-N [14], which can also guarantee a high-computational efficiency considering that the walk length, i.e., m, is not long (generally set to 3) in our proposal.

Next, to combine the temporal distance information on different walks, we apply the self-attention network (SAT) [56] to dynamically determine the importance of various walks for generating the final representation of the source node

$$\begin{cases}
\text{SAT}\left(\tilde{\mathbf{W}}_{v_s}\right) = \text{Softmax}\left(\dfrac{\mathbf{Q}\mathbf{K}^\top}{\sqrt{d}}\right)\mathbf{V} \\
\mathbf{Q} = \tilde{\mathbf{W}}_q \tilde{\mathbf{W}}_{v_s}, \ \mathbf{K} = \tilde{\mathbf{W}}_k \tilde{\mathbf{W}}_{v_s}, \ \mathbf{V} = \tilde{\mathbf{W}}_v \tilde{\mathbf{W}}_{v_s}
\end{cases} \tag{18}$$

where $\tilde{\mathbf{W}}_{v_s} = \{\tilde{\mathbf{w}}_1, \tilde{\mathbf{w}}_2, \ldots, \tilde{\mathbf{w}}_M\}$ are the representations of different adaptive walks of v_s, M is the number of walks, and $\tilde{\mathbf{W}}_q, \tilde{\mathbf{W}}_k, \tilde{\mathbf{W}}_v \in \mathbb{R}^{d\times d}$ are learnable parameters in the SAT.

After that, we apply the mean pooling to combine the representations of various walks as follows:

$$\tilde{\mathbf{v}}_s = \mathcal{F}_{\text{mean-pool}}\left(\text{SAT}\left(\tilde{\mathbf{W}}_{v_s}\right)\right) \tag{19}$$

where $\tilde{\mathbf{v}}_s \in \mathbb{R}^d$ is the final structure-aware representation of source node v_s. Similarly, the target node v_t can be represented as $\tilde{\mathbf{v}}_t \in \mathbb{R}^d$ by following the same modeling process.

Finally, we generate a prediction score to measure the distance between source node and target node on the dynamic graph structure by inputting the concatenation of $\tilde{\mathbf{v}}_s$ and $\tilde{\mathbf{v}}_t$ into a MLP

$$\tilde{z}_{st} = \text{MLP}\left(\left[\tilde{\mathbf{v}}_s, \tilde{\mathbf{v}}_t\right]\right) = \mathbf{W}_4\left(\sigma\left(\mathbf{W}_3\left[\tilde{\mathbf{v}}_s, \tilde{\mathbf{v}}_t\right]\right)\right) \tag{20}$$

where $\mathbf{W}_3 \in \mathbb{R}^{d\times 2d}$ and $\mathbf{W}_4 \in \mathbb{R}^{d\times d}$ are learnable parameters in MLP, and σ denotes the ReLu function.

C. Prediction and Optimization

After obtaining the embedding-based score as $\hat{z}_{st}$ and the structure-aware score as $\tilde{z}_{st}$, we input them into a MLP to generate the final prediction score for measuring the probability of forming a link between the source node and the target node in the future timestamp t_{st}

$$z_{st} = \text{MLP}\left(\left[\hat{z}_{st}, \tilde{z}_{st}\right]\right) = \mathbf{W}_6\left(\sigma\left(\mathbf{W}_5\left[\hat{z}_{st}, \tilde{z}_{st}\right]\right)\right) \tag{21}$$

where $\mathbf{W}_5 \in \mathbb{R}^{d\times 2}$ and $\mathbf{W}_6 \in \mathbb{R}^{1\times d}$ are trainable parameters, and σ is the ReLu function.

Finally, to train our proposed DEAL model, we employ the cross-entropy function as the optimization objective to learn the trainable parameters as follows:

$$L = \sum_{(v_s, v_t, t_{st})\in\mathcal{E}} -\log(\sigma(z_{st})) - Q\mathbb{E}_{v_n\sim P_n(v)}\log(\sigma(1 - z_{sn})) \tag{22}$$

where $(v_s, v_t, t_{st}) \in \mathcal{E}$ is the observed temporal edge in the training set, and σ is the sigmoid function. Moreover, v_n indicates the negative sample, and z_{sn} is the corresponding prediction score of negative edge (v_s, v_n, t_{st}), which replaces the target node v_t in (v_s, v_t, t_{st}) by v_n. In addition, Q is the negative sampling number and $P_n(v)$ indicates the negative sampling distribution in the node space. Following [13], [14], we set the negative sampling number to 1. Finally, we apply the BPTT algorithm to train the proposed DEAL model.

V. Experiments

A. Research Questions

We investigate the effectiveness of our proposed DEAL model by addressing the following five research questions.

(RQ1) Can DEAL achieve the state-of-the-art performance for inductive link prediction on temporal networks and how is the time complexity of DEAL?

(RQ2) How is the contribution of each component in DEAL to the inductive link prediction performance?

(RQ3) How does DEAL perform in scenarios of various data sparsity compared with the competitive baselines?

(RQ4) Can the designed adaptive sampling method help improve the performance and what is the impact of parameter λ?

(RQ5) What is the impact of different hyper parameters on the model performance of DEAL?

TABLE I
Statistics of the Datasets Used in Our Experiments

Statistics	MathOverflow	AskUbuntu	StackOverflow
# Nodes	3,135	8,160	51,246
# Edges	16,455	20,389	124,828
# Seen nodes	2,435	6,002	39,645
# Unseen nodes	396	1,271	6,615
# Training edges	11,518	14,272	87,379
# Valid edges	1,000	2,075	11,878
# Test edges	1,201	2,362	13,578
Timespan	90 days	30 days	3 days

B. Datasets

To examine the effectiveness of our proposed DEAL method, we evaluate the performance of DEAL and the baselines on three sparse non-attributed temporal network datasets, i.e., MathOverflow,[1] AskUbuntu[2] and StackOverflow,[3] which are collected from three stack exchange websites, i.e., Math Overflow, Ask Ubuntu, and Stack Overflow, to respectively constitute a dynamic graph of interactions. Specifically, the temporal edge $(v_s, v_t, t_{st}) \in \mathcal{E}$ in each dynamic network contains three types of interactions, i.e., user v_s answers user v_t's question, comments on user v_t's question, and comments on user v_t's answer at timestamp t_{st}.

In addition, the most recent 90 days of MathOverflow, 30 days of AskUbuntu and three days of StackOverflow are taken for experiments. Moreover, we separate the temporal edges in three datasets for training, validation and test according the chronological order with a proportion of 70%, 15%, and 15%. In addition, the inductive edges in the validation and test sets which are associated with unseen node(s) not appearing in the training set are adopted for model evaluation. Table I summarizes the statistics of three datasets after preprocessing.

C. Model Summary

The baselines selected for comparison with DEAL are as follows.

1) **GraphSAGE** [41] inductively learns the node representation by propagating information from the sampled seen nodes, whose embeddings can be learned during training.
2) **GAT** [37] adopts the same setting as GraphSAGE in our experiments except utilizing an attentive way to determine the weights of different neighbors in message passing.
3) **GraphSAGE+T** considers the time dynamics on the basis of GraphSAGE by concatenating the node embedding with the time encoding vector generated by (14).
4) **GAT+T** combines the node embedding obtained by GAT and the time vector encoded by (14) using concatenation.
5) **TGAT** [13] proposes a TGAT layer to aggregate the temporal and topological information from the neighbor nodes, and further designs a time encoding function based on the Bochner's theorem [52] to achieve continuous time encoding.

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

TABLE II
MODEL PERFORMANCE OF DIFFERENT LINK PREDICTION METHODS. THE RESULTS OF THE BEST PERFORMING BASELINE AND THE BEST PERFORMER IN EACH COLUMN ARE UNDERLINED AND BOLDFACED, RESPECTIVELY

Method	MathOverflow			AskUbuntu			StackOverflow		
	ACC(%)	AUC(%)	AP(%)	ACC(%)	AUC(%)	AP(%)	ACC(%)	AUC(%)	AP(%)
GraphSAGE	59.12	66.21	64.88	54.36	60.58	61.19	67.74	76.40	76.01
GAT	62.70	71.84	70.23	70.65	77.65	77.08	64.99	79.93	79.60
GraphSAGE+T	53.08	57.68	60.44	55.63	64.10	64.33	71.73	78.45	79.77
GAT+T	61.58	69.60	70.42	73.48	79.70	79.11	70.36	77.32	79.05
TGAT	55.16	58.13	60.84	64.79	73.11	74.93	68.81	77.46	79.91
TGN	58.56	61.43	63.46	66.57	74.98	75.08	70.23	78.86	81.13
CAW-N	67.87	75.48	80.69	72.50	81.32	83.34	80.07	88.93	90.56
DEAL	**73.50**	**83.43**	**86.37**	**75.31**	**85.33**	**87.52**	**80.16**	**90.09**	**91.94**

6) **TGN** [24] develops a TGN, which learns the dynamic node embeddings via memory networks [53] and graph-based operators, and further introduces an advanced training strategy for efficiently learning from the data sequence.
7) **CAW-N** [14] proposes a casual anonymous walk method to generate the relative identity embedding between two nodes for conducting inductive link prediction by measuring their distance on the dynamic graph structure.

It is worth noting that some representative attribute-aware methods such as [57] and [58] can also be transferred to our investigated task by removing the node/edge attributes and adding the time features. However, they will be degenerated to the methods similar to the existing baselines such as TGAT and GraphSAGE+T, thus here we omit them in the compared baselines for avoiding duplication.

D. Experimental Setup

The hyperparameters of DEAL are tuned on the validation set for determining the best selection on different datasets. Specifically, the vector dimension d is tuned in $\{60, 80, 100, 120, 140\}$, the scaling parameter λ is searched in $\{1e^{-7}, 1e^{-6}, 1e^{-5}, 1e^{-4}\}$, the trade-off threshold $d_{\text{threshold}}$ is ranged in $\{0.1, 0.15, 0.2, 0.25, 0.3\}$, the walk number M is tuned in $\{2, 4, 8, 16, 32\}$, and the walk length m is set to 3 following [13], [14]. Moreover, we adopt the Adam optimizer with a learning rate of $1e^{-4}$ for training the learnable parameters in DEAL, and set the batch size to 64. Furthermore, we adopt accuracy (ACC), AUC, and AP as the metrics for evaluating the inductive link prediction performance of our proposal as well as the baselines.

To facilitate the reproducibility of our results in this article, we share the code and data used to run the experiments at https://github.com/nudtzpan/DEAL.

VI. RESULTS AND DISCUSSION

A. Overall Comparison

1) Performance Comparison: The results of our proposed DEAL approach and the competitive baselines are presented in Table II.

With regards to the baselines, let us begin by comparing the embedding-based methods GraphSAGE and GAT with their respective temporal version, namely GraphSAGE+T and GAT+T. It is evident that GraphSAGE+T consistently outperforms GraphSAGE on both AskUbuntu and StackOverflow while losing the competition in the case of MathOverflow. And GAT+T outperforms GAT on AskUbuntu, yet exhibits inferior performance in most cases on MathOverflow and StackOverflow. This observation suggests that the simple combination of the time encoding with the node embedding fails to consistently enhance the prediction performance across different scenarios. Furthermore, we can discern that the performance of TGAT and TGN is unsatisfactory, owing to their heavy reliance on the edge attributes, which results in the poor performance within our experimental setting where the attribute information is unavailable. Differently, by learning the relative identity vectors for nodes using anonymous distance encoding, CAW-N achieves the most remarkable performance among the baselines in terms of almost all metrics across three datasets.

Moving on to our proposed DEAL approach, we can first obtain the observation from Table II that DEAL consistently outperforms the competitive baselines in terms of accuracy, AUC, and AP across all three datasets, thereby validating the effectiveness of our proposal. Moreover, it is worth noting that the performance improvements are more conspicuous on small datasets compared to the large ones. Specifically, DEAL improves the performance above the best baseline CAW-N in terms of accuracy, AUC, and AP by 8.30%, 10.53%, and 7.04%, respectively, on MathOverflow, while the improvement rates are 3.88%, 4.93%, and 5.02% on AskUbuntu and 0.11%, 1.30%, and 1.52% on StackOverflow, respectively. These findings indicate that DEAL not only outperforms the competitive baselines in scenarios with sparse data, but also exhibits particularly evident effectiveness in realistic applications constrained by limited data availability.

2) Time Complexity Comparison: In addition, we also compare the time complexity of our proposed DEAL and the state-of-the-art baseline CAW-N, which is the best-performing baseline as shown in Table II. The results are presented in Table III, where s' and s denote the number of epochs in the pretraining and training phases, respectively, $|\mathcal{E}|$ means the number of edges in the temporal network, M and m indicate the number of walks and the length of each walk, respectively, and d represents the embedding dimension. Then, the time complexity of DEAL can be denoted as $s'|\mathcal{E}|(\text{Mmd} + md + d + 1)$ in the pretraining stage, and

TABLE III
COMPARISON OF THE TIME COMPLEXITY BETWEEN CAW-N AND DEAL

Methods	CAW-N	DEAL
Pretraining	-	$s'\|\mathcal{E}\|(Mmd + md + d + 1)$
Node encoding	$s\|\mathcal{E}\|(Mmd^2 + Md^2 + d^2 + Md + md + d)$	$s\|\mathcal{E}\|(Mmd^2 + Md^2 + d^2 + Mmd + Md + md + d)$
Prediction	$s\|\mathcal{E}\|(d^2 + d + 1)$	$s\|\mathcal{E}\|(d^2 + d + 1)$

TABLE IV
ABLATION STUDY RESULTS OF DEAL

Method	MathOverflow			AskUbuntu			StackOverflow		
	ACC(%)	AUC(%)	AP(%)	ACC(%)	AUC(%)	AP(%)	ACC(%)	AUC(%)	AP(%)
DEAL	73.50	83.43	86.37	75.31	85.33	87.52	80.16	90.09	91.94
w/o AdaSampler	69.50	80.88	83.90	73.85	84.52	86.38	78.47	88.69	90.39
w/o Embedding	73.00	79.68	84.35	75.82	84.26	86.28	79.72	88.48	90.23
w/o Structure	63.58	73.33	72.29	68.62	76.76	74.79	65.16	79.73	79.04

by combining the node encoding and prediction parts, the complexity of DEAL and CAW-N in the training stage are $s|\mathcal{E}|(\text{Mmd}^2 + Md^2 + d^2 + \text{Mmd} + Md + md + d + 1)$ and $s|\mathcal{E}|(\text{Mmd}^2 + Md^2 + d^2 + Md + md + d + 1)$, respectively. From the results, we can observe that for the training phase, the time complexity of DEAL and CAW-N are highly similar, except that DEAL introduces an additional time complexity of $s|\mathcal{E}|\text{Mmd}$ from the embedding-based distance measuring. In addition, as for the pretraining part, our proposal can be applied in realistic scenarios by fine-tuning the existing pretrained embeddings of the seen nodes termly with new edges coming on the temporal network. This can largely decrease the added time consumption incorporated by the pretraining in DEAL compared to the competitive baseline CAW-N, making DEAL practical in real-world applications.

B. Ablation Study

To answer RQ2, we compare our proposed DEAL method with its three variants: 1) w/o AdaSampler, which replaces the adaptive neighbor sampling in DEAL with random sampling; 2) w/o embedding; and 3) w/o structure, which removes the embedding-based prediction score and the structure-aware prediction score from the dual-channel distance measuring component, respectively. The results of DEAL and the variants in terms of accuracy, AUC, and AP are presented in Table IV. First, we can observe that DEAL outperforms the variant w/o AdaSampler in all cases on three datasets, indicating that through the pretrained seen node embeddings, our designed adaptive sampling method can help dynamically select the proper sampling strategy from recent sampling and random sampling for capturing the common neighbors of source node and target node, improving the prediction performance.

In addition, comparing w/o embedding and w/o structure to DEAL, we can see that both the embedding-based distance and the structure-aware distance contribute to the accurate inductive link prediction on temporal networks. However, there exists an exceptional phenomenon that DEAL obviously outperforms w/o embedding in terms of AUC and AP while not in terms of accuracy. We analyze that this may be attributed to the threshold selection [59], [60] and the presence of data noise [61], [62], which may impact the accuracy metric. In such scenarios, AUC and AP serve as the more comprehensive and robust metrics for evaluating the model performance. Thus, the obviously better performance of DEAL than w/o embedding in terms of AUC and AP can adequately prove the effectiveness of the embedding-based distance.

Moreover, the structure-aware distance has a larger impact than the embedding-based one since w/o embedding achieves a consistently better performance than w/o structure on three datasets as shown in Table IV. This may be due to that in the scenarios with sparse data, the seen node embeddings are hard to learn, thus the effectiveness of the explicit distance encoding on the dynamic graph structure is relatively more obvious. Interestingly, we can find that the performance gap between w/o embedding and w/o structure is especially obvious on large datasets. For instance, w/o embedding improves the performance above w/o Structure by 14.82%, 8.66%, and 12.06% in terms of accuracy, AUC, and AP on MathOverflow, respectively, while the corresponding improvement rates are 22.34%, 10.97%, and 14.16% on StackOverflow. This indicates that the superiority of the structure-aware distance measuring over the embedding-based one is more obvious on large datasets than on small ones.

C. Impact of Data Sparsity

For RQ3, to test the sensitivity of DEAL to the data sparsity, we randomly drop the temporal links in the training set and preserve the original edges of a proportion γ, where γ is tuned in {20%, 40%, 60%, 80%, 100%}. In addition, we also include the best baseline CAW-N and the variant w/o AdaSampler into consideration, so as to see the effectiveness of the adaptive sampling by comparing DEAL to w/o AdaSampler and the utility of dual-channel distance measuring by comparing w/o AdaSampler to CAW-N in scenarios of various data sparsity. We adopt AskUbuntu and StackOverflow for experiments without the Mathoverflow dataset considering its tiny scale, which makes the training unstable when removing a large proportion of temporal links in the training set. The results in terms of AP are plotted in Fig. 4, where the results in terms

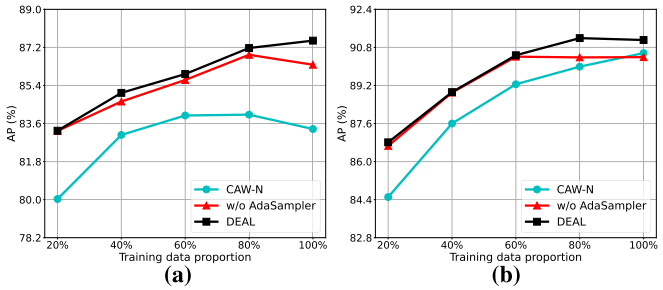

Fig. 4. Model performance in terms of AP on scenarios of various data sparsity on (a) AskUbuntu and (b) StackOverflow.

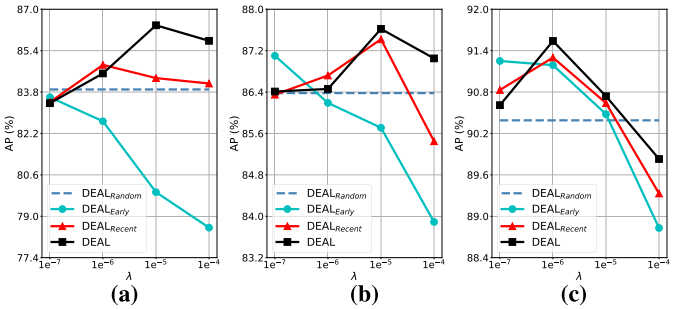

Fig. 5. Model performance of DEAL in terms of AP with different neighbor sampling methods under various λ. (a) MathOverflow. (b) AskUbuntu. (c) StackOverflow.

of accuracy and AUC show similar phenomena and thus are omitted due to the space limitation.

From Fig. 4, first we can observe that with more training data involving, the performance of all models on StackOverflow generally increases except that w/o AdaSampler shows a stable trend on large proportions. However, differently on AskUbuntu, when the training data proportion increases, the performance of DEAL consistently increases, while the performance of CAW-N and w/o AdaSampler first increases, achieving the best performance around a proportion of 80%, and then begins to decrease. The performance drop of CAW-N and w/o AdaSampler may be due to that on small datasets, without our designed adaptive sampling method, the common neighbors of source node and target node cannot be accurately detected, leading to an unsatisfying performance.

In addition, by comparing DEAL to w/o AdaSampler, we can see that on both datasets, DEAL slightly performs better than w/o AdaSampler on small training data proportions. However, when the proportion reaches 80%, the performance gap between DEAL and w/o AdaSampler becomes obvious, indicating that the adaptive sampling plays a relatively more important role in the scenarios with abundant training data. Moreover, comparing w/o AdaSampler to CAW-N on StackOverflow, we can see that the improvements are especially obvious on small training data proportions, indicating that the embedding-based distance measuring contributes relatively more to the performance on sparse large datasets.

D. Impact of Neighbor Sampling

For RQ4, to validate the superiority of our designed adaptive sampling over other neighbor sampling methods, we compare DEAL with the following variants of DEAL.

1) $\text{DEAL}_{\text{Random}}$, which replaces the adaptive sampling in DEAL by random sampling.
2) $\text{DEAL}_{\text{Recent}}$, which utilizes the recent sampling to select recent neighbors by adopting a probability proportional to $\exp(\lambda(t_i - t_{i-1}))$, where t_i and t_{i-1} are the timestamps of the candidate neighbor and the previous node on the walk during sampling, respectively.
3) $\text{DEAL}_{\text{Early}}$ is opposite to $\text{DEAL}_{\text{Recent}}$, which tends to select the early nodes in historical interactions as neighbors with a probability proportional to $\exp(\lambda(t_{i-1} - t_i))$.

In addition, we tune the scaling parameter λ in $\{1e^{-7}, 1e^{-6}, 1e^{-5}, 1e^{-4}\}$ in DEAL as well as the variants $\text{DEAL}_{\text{Recent}}$ and $\text{DEAL}_{\text{Early}}$ to see the impact of λ on the model performance. The results of DEAL and three variants in terms of AP on three datasets are presented in Fig. 5.

From Fig. 5, first we can observe that with the scaling parameter λ increasing, the performance of $\text{DEAL}_{\text{Early}}$ consistently decreases on all three datasets, which is due to that the selected early neighbors fail to take the time dynamics on temporal networks into consideration. As for $\text{DEAL}_{\text{Recent}}$, we can see that when λ increases, the performance of $\text{DEAL}_{\text{Recent}}$ first increases, and then generally decreases. This indicates that when λ is small, $\text{DEAL}_{\text{Recent}}$ can help capture the recent neighbors of source node and target node, which considers the network dynamics to some extent, obtaining a better performance than $\text{DEAL}_{\text{Random}}$ and $\text{DEAL}_{\text{Early}}$. However, after achieving the peak prediction performance at a λ value of $1e^{-5}$ on MathOverflow as well as AskUbuntu and $1e^{-6}$ on StackOverflow, respectively, the limitation of the temporal scale for sampling neighbors hurts the model performance.

In addition, we can see that DEAL can generally outperform the variants on three datasets, validating the effectiveness of our designed adaptive sampling for selecting the informative neighbors for distance measuring between nodes. Specifically, by relying on the distance between source node and target node measured in the pretrained embedding space using the L2 normalization, the adaptive sampling strategy can dynamically determine the selection of random sampling or recent sampling for sampling neighbors to extract temporal adaptive walks. That is, the adaptive sampling can effectively consider the network dynamics while guaranteeing the temporal scale at the same time for sampling neighbors.

E. Impact of Hyperparameters

To answer RQ5, we tune the hyperparameters of DEAL including the pretraining epochs in $\{6, 8, 10, 12, 14\}$, the sampling threshold $d_{\text{threshold}}$ in $\{0.1, 0.15, 0.2, 0.25, 0.3\}$, the walk number M in $\{2, 4, 8, 16, 32\}$, and the vector dimension d in $\{60, 80, 100, 120, 140\}$, to test the sensitivity of DEAL to different hyperparameters. The results in terms of AP on three datasets are presented in Fig. 6.

1) Pretraining Epochs: From Fig. 6(a), we can observe that when the pretraining epoch number increases from 6 to 14, the performance of DEAL on StackOverflow generally increases, while the performance on MathOverflow and AskUbuntu slightly decreases. This difference could be due to the reason that the scale of three datasets are different, which indicates

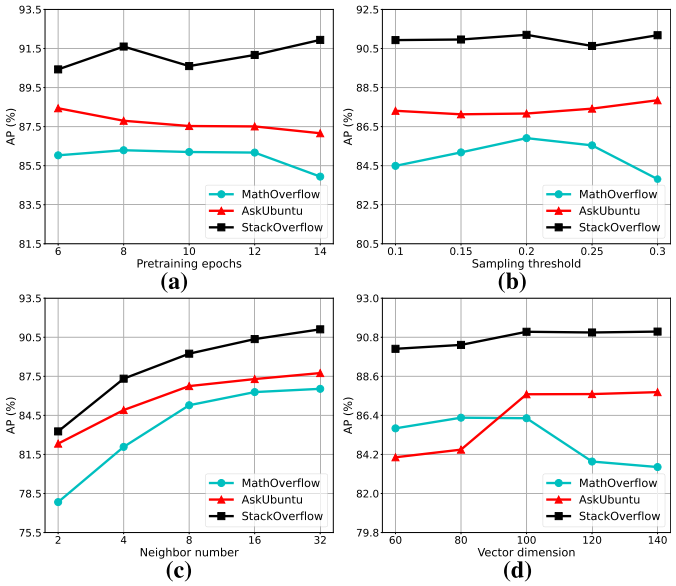

Fig. 6. Model performance of DEAL in terms of AP under different hyperparameters. (a) Impact of pretraining epochs. (b) Impact of sampling threshold. (c) Impact of neighbor number. (d) Impact of vector dimension.

that it requires relatively more epochs for pretraining on a large dataset.

2) Sampling Threshold: From Fig. 6(b), we can observe that with the threshold $d_{\text{threshold}}$ in adaptive sampling increasing from 0.1 to 0.3, the performance of DEAL on AskUbuntu and StackOverflow fluctuates and generally shows a slightly increasing trend, while the performance on MathOverflow first increases, and then decreases after achieving the peak value at a threshold of 0.2. This may be due to that for the small dataset MathOverflow, the node embeddings cannot be well learned as in the large datasets AskUbuntu and StackOverflow, which makes the node embeddings relatively more uniformly distributed in the embedding space. Thus, a small threshold is proper on the MathOverflow dataset.

3) Walk Number: From Fig. 6(c), we can see that when the walk number M increases, the performance of DEAL on all three datasets generally increases. Moreover, the increasing rate consistently decreases with the walk number increasing on MathOverflow and AskUbuntu, especially after the walk number reaching 8 where the performance remains stable. Differently on StackOverflow, the performance still increases rapidly on large walk numbers. This may be due to that on large-scale datasets, relatively more temporal adaptive walks are required to help measure the distance between source node and target node in the embedding space and on the dynamic graph structure, simultaneously.

4) Vector Dimension: From Fig. 6(d), we can see that with the vector dimension d increasing from 60 to 140, the performance of DEAL on two large datasets, i.e., AskUbuntu and StackOverflow, obviously increases, and then shows a stable trend after $d = 100$. This is due to that large vector dimensions increase the representation ability of DEAL for distance measuring. Differently on MathOverflow, when the vector dimension increases, the performance of DEAL first increases from $d = 60$ to 100, and then shows a consistently decreasing trend. This may be due to that on small datasets, large vector dimensions lead to an over-fitting problem, decreasing the generalization ability of DEAL to the test set.

VII. Conclusion

In this article, we propose a DEAL approach for inductive link prediction on temporal networks. DEAL designs an adaptive sampling method relying on the pretrained embeddings of seen nodes to dynamically sample neighbors, which improves the probability of capturing the common neighbors of source node and target node. In addition, we propose a dual-channel distance measuring module which simultaneously measures the distance between source node and target node in the embedding space and on the dynamic graph structure for predicting the future links. Extensive experiments conducted on three temporal network datasets validate that DEAL can obviously improve the inductive link prediction performance in terms of accuracy, AUC, and AP.

As to future work, on the one hand, we are interested in enhancing the node representation learning by incorporating the self-supervisions [63], [64]. On the other hand, we would like to design a generic link prediction framework for predicting the transductive and inductive edges simultaneously on temporal networks.

References

[1] Y. Lu, X. Wang, C. Shi, P. S. Yu, and Y. Ye, "Temporal network embedding with micro- and macro-dynamics," in *Proc. 28th ACM Int. Conf. Inf. Knowl. Manage. (CIKM)*, Nov. 2019, pp. 469–478.

[2] Y. Ma, Z. Guo, Z. Ren, J. Tang, and D. Yin, "Streaming graph neural networks," in *Proc. 43rd Int. ACM SIGIR Conf. Res. Develop. Inf. Retr. (SIGIR)*, Jul. 2020, pp. 719–728.

[3] G. Du et al., "Graph-based class-imbalance learning with label enhancement," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 34, no. 9, pp. 6081–6095, Sep. 2021.

[4] 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.

[5] J. Wang, G. Song, Y. Wu, and L. Wang, "Streaming graph neural networks via continual learning," in *Proc. 29th ACM Int. Conf. Inf. Knowl. Manage. (CIKM)*, Oct. 2020, pp. 1515–1524.

[6] 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.

[7] Z. Liu, C. Huang, Y. Yu, and J. Dong, "Motif-preserving dynamic attributed network embedding," in *Proc. Web Conf. (WWW)*, Apr. 2021, pp. 1629–1638.

[8] L. Cai et al., "Structural temporal graph neural networks for anomaly detection in dynamic graphs," in *Proc. 30th ACM Int. Conf. Inf. Knowl. Manage. (CIKM)*, Oct. 2021, pp. 3747–3756.

[9] J. Li, H. Dani, X. Hu, J. Tang, Y. Chang, and H. Liu, "Attributed network embedding for learning in a dynamic environment," in *Proc. ACM Conf. Inf. Knowl. Manage. (CIKM)*, Nov. 2017, pp. 387–396.

[10] S. M. Kazemi et al., "Representation learning for dynamic graphs: A survey," *J. Mach. Learn. Res.*, vol. 21, pp. 70:1–70:73, Apr. 2020.

[11] S. Tang, Z. Meng, and S. Liang, "Dynamic co-embedding model for temporal attributed networks," *IEEE Trans. Neural Netw. Learn. Syst.*, early access, Jul. 28, 2022, doi: 10.1109/TNNLS.2022.3193564.

[12] Y. Hao, X. Cao, Y. Fang, X. Xie, and S. Wang, "Inductive link prediction for nodes having only attribute information," in *Proc. 29th Int. Joint Conf. Artif. Intell. (IJCAI)*, Jul. 2020, pp. 1209–1215.

[13] D. Xu, C. Ruan, E. Körpeoglu, S. Kumar, and K. Achan, "Inductive representation learning on temporal graphs," in *Proc. Int. Conf. Learn. Represent. (ICLR)*, 2020.

[14] Y. Wang, Y. Chang, Y. Liu, J. Leskovec, and P. Li, "Inductive representation learning in temporal networks via causal anonymous walks," in *Proc. Int. Conf. Learn. Represent. (ICLR)*, 2021.

[15] 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. (AAAI)*, 2018, pp. 571–578.

[16] 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 (WSDM)*, Jan. 2020, pp. 519–527.

[17] S. Tian, R. Wu, L. Shi, L. Zhu, and T. Xiong, "Distance—Aware learning for inductive link prediction on temporal networks," in *Proc. 30th ACM Int. Conf. Inf. Knowl. Manage. (CIKM)*, Oct. 2021, pp. 1814–1823.

[18] M. Kosinski, D. Stillwell, and T. Graepel, "Private traits and attributes are predictable from digital records of human behavior," *Proc. Nat. Acad. Sci. USA*, vol. 110, no. 15, pp. 5802–5805, Apr. 2013.

[19] J. A. Kroll, "Accountable algorithms," Ph.D. dissertation, Dept. Comput. Sci., Princeton Univ., Princeton, NJ, USA, 2015.

[20] Y.-A. de Montjoye, C. A. Hidalgo, M. Verleysen, and V. D. Blondel, "Unique in the crowd: The privacy bounds of human mobility," *Sci. Rep.*, vol. 3, no. 1, pp. 1–5, Mar. 2013.

[21] H. V. Jagadish et al., "Big data and its technical challenges," *Commun. ACM*, vol. 57, no. 7, pp. 86–94, Jul. 2014.

[22] 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.

[23] H. Yin, M. Zhang, Y. Wang, J. Wang, and P. Li, "Algorithm and system co-design for efficient subgraph-based graph representation learning," 2022, *arXiv:2202.13538*.

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

[26] A. Ahmed, N. Shervashidze, S. Narayanamurthy, V. Josifovski, and A. J. Smola, "Distributed large-scale natural graph factorization," in *Proc. 22nd Int. Conf. World Wide Web*, May 2013, pp. 37–48.

[27] S. Cao, W. Lu, and Q. Xu, "Grarep: Learning graph representations with global structural information," in *Proc. Int. Conf. Inf. Knowl. Manag. (CIKM)*, 2015, pp. 891–900.

[28] M. Ou, P. Cui, J. Pei, Z. Zhang, and W. Zhu, "Asymmetric transitivity preserving graph embedding," in *Proc. 22nd ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining (KDD)*, Aug. 2016, pp. 1105–1114.

[29] T. Mikolov, K. Chen, G. Corrado, and J. Dean, "Efficient estimation of word representations in vector space," in *Proc. Int. Conf. Learn. Represent. (ICLR)*, 2013.

[30] 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 (KDD)*, Aug. 2014, pp. 701–710.

[31] A. Grover and J. Leskovec, "Node2vec: Scalable feature learning for networks," in *Proc. 22nd ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining (KDD)*, Aug. 2016, pp. 855–864.

[32] J. Tang, M. Qu, M. Wang, M. Zhang, J. Yan, and Q. Mei, "Line: Large-scale information network embedding," in *Proc. 24th Int. Conf. World Wide Web (WWW)*, May 2015, pp. 1067–1077.

[33] D. Wang, P. Cui, and W. Zhu, "Structural deep network embedding," in *Proc. 22nd ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining (KDD)*, Aug. 2016, pp. 1225–1234.

[34] T. N. Kipf and M. Welling, "Variational graph auto-encoders," 2016, *arXiv:1611.07308*.

[35] D. P. Kingma and M. Welling, "Auto-encoding variational Bayes," in *Proc. Int. Conf. Learn. Represent. (ICLR)*, 2014.

[36] R. Ying, R. He, K. Chen, P. Eksombatchai, W. L. Hamilton, and J. Leskovec, "Graph convolutional neural networks for web-scale recommender systems," in *Proc. 24th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining (KDD)*, Jul. 2018, pp. 974–983.

[37] P. Velickovic, G. Cucurull, A. Casanova, A. Romero, P. Liò, and Y. Bengio, "Graph attention networks," in *Proc. Int. Conf. Learn. Represent. (ICLR)*, 2018.

[38] M. Fey, J. E. Lenssen, F. Weichert, and J. Leskovec, "GNNAutoScale: Scalable and expressive graph neural networks via historical embeddings," in *Proc. Int. Conf. Mach. Learn. (ICML)*, 2021, pp. 3294–3304.

[39] S. Suresh, V. Budde, J. Neville, P. Li, and J. Ma, "Breaking the limit of graph neural networks by improving the assortativity of graphs with local mixing patterns," in *Proc. 27th ACM SIGKDD Conf. Knowl. Discovery Data Mining (KDD)*, Aug. 2021, pp. 1541–1551.

[40] T. N. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," in *Proc. Int. Conf. Learn. Represent. (ICLR)*, 2017.

[41] W. L. Hamilton, Z. Ying, and J. Leskovec, "Inductive representation learning on large graphs," in *Proc. Conf. Neural Inf. Process. Syst. (NIPS)*, 2017, pp. 1024–1034.

[42] Y. Li, D. Tarlow, M. Brockschmidt, and R. Zemel, "Gated graph sequence neural networks," in *Proc. Int. Conf. Learn. Represent. (ICLR)*, 2016.

[43] G. Wang, R. Ying, J. Huang, and J. Leskovec, "Multi-hop attention graph neural networks," in *Proc. 30th Int. Joint Conf. Artif. Intell. (IJCAI)*, Aug. 2021, pp. 3089–3096.

[44] J. You, J. M. G. Selman, R. Ying, and J. Leskovec, "Identity-aware graph neural networks," in *Proc. AAAI Conf. Artif. Intell.*, 2021, pp. 10737–10745.

[45] 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.

[46] 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 (KDD)*, Jul. 2019, pp. 1269–1278.

[47] 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.

[48] P. Goyal, S. R. Chhetri, and A. Canedo, "Dyngraph2vec: Capturing network dynamics using dynamic graph representation learning," *Knowl.-Based Syst.*, vol. 187, Jan. 2020, Art. no. 104816.

[49] 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.

[50] A. Pareja et al., "EvolveGCN: Evolving graph convolutional networks for dynamic graphs," in *Proc. AAAI Conf. Artif. Intell. (AAAI)*, 2020, pp. 5363–5370.

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

[52] 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.

[53] S. Sukhbaatar, A. Szlam, J. Weston, and R. Fergus, "End-to-end memory networks," in *Proc. Conf. Neural Inf. Process. Syst. (NIPS)*, 2015, pp. 2440–2448.

[54] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning representations by back-propagating errors," *Nature*, vol. 323, no. 6088, pp. 533–536, Oct. 1986.

[55] A. Rahimi and B. Recht, "Random features for large-scale kernel machines," in *Proc. Conf. Neural Inf. Process. Syst. (NIPS)*, 2007, pp. 1177–1184.

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

[57] A. Bojchevski and S. Günnemann, "Deep Gaussian embedding of graphs: Unsupervised inductive learning via ranking," in *Proc. Int. Conf. Learn. Represent. (ICLR)*, 2018.

[58] Y. Hao, X. Cao, Y. Fang, X. Xie, and S. Wang, "Inductive link prediction for nodes having only attribute information," in *Proc. 29th Int. Joint Conf. Artif. Intell. (IJCAI)*, Jul. 2020, pp. 1209–1215.

[59] J. Davis and M. Goadrich, "The relationship between precision-recall and ROC curves," in *Proc. 23rd Int. Conf. Mach. Learn. (ICML)*, 2006, pp. 233–240.

[60] Y. Yang, R. N. Lichtenwalter, and N. V. Chawla, "Evaluating link prediction methods," *Knowl. Inf. Syst.*, vol. 45, no. 3, pp. 751–782, Dec. 2015.

[61] Y. Liu, Y. Zheng, D. Zhang, H. Chen, H. Peng, and S. Pan, "Towards unsupervised deep graph structure learning," in *Proc. ACM Web Conf. (WWW)*, Apr. 2022, pp. 1392–1403.

[62] M. A. Hasan and M. J. Zaki, "A survey of link prediction in social networks," in *Social Network Data Analytics*. New York, NY, USA: Springer, 2011, pp. 243–275.

[63] S. Gidaris, P. Singh, and N. Komodakis, "Unsupervised representation learning by predicting image rotations," in *Proc. Int. Conf. Learn. Represent. (ICLR)*, 2018.

[64] T. Chen, S. Kornblith, M. Norouzi, and G. Hinton, "A simple framework for contrastive learning of visual representations," in *Proc. Int. Conf. Mach. Learn. (ICML)*, 2020, pp. 1597–1607.

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, *ACM Transactions on Information Systems* (TOIS), *International Railway Journal* (IRJ), and *Neural Computing and Applications* (NCAA). His research interests include graph analytics and recommender systems.

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

He is an Associate Professor with the National University of Defense Technology, Changsha, China. He has published several papers in WWW, SIGIR, CIKM, FnTIR, *ACM Transactions on Information Systems* (TOIS), IEEE TRANSACTIONS ON KNOWLEDGE AND DATA ENGINEERING (TKDE), and IEEE TRANSACTIONS ON NEURAL NETWORKS AND LEARNING SYSTEMS (TNNLS). 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 (TKDE), *ACM Transactions on Information Systems* (TOIS), *ACM Transactions on Knowledge Discovery from Data* (TKDD), IEEE TRANSACTIONS ON NEURAL NETWORKS AND LEARNING SYSTEMS (TNNLS), *The VLDB Journal* (VLDBJ), *Information Processing and Management* (IPM), and *Journal of the Association for Information Science and Technology* (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 currently a Professor with the School of Computer, National University of Defense Technology. He has authored or coauthored more than 80 peer-reviewed papers, including those in highly regarded journals and conferences, such as IEEE TRANSACTIONS ON PATTERN ANALYSIS AND MACHINE INTELLIGENCE (TPAMI), IEEE TRANSACTIONS ON KNOWLEDGE AND DATA ENGINEERING (TKDE), IEEE TRANSACTIONS ON IMAGE PROCESSING (TIP), IEEE TRANSACTIONS ON NEURAL NETWORKS AND LEARNING SYSTEMS (TNNLS), IEEE TRANSACTIONS ON MULTIMEDIA (TMM), IEEE TRANSACTIONS ON INFORMATION FORENSICS AND SECURITY (TIFS), 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.

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 with the National University of Defense Technology. He has published many papers in SIGIR, WWW, *ACM Transactions on Information Systems* (TOIS), and *Information Processing and Management* (IPM) as well as other top conference proceedings and journals. His research interests include information systems and information retrieval.