HGN2T: A Simple but Plug-and-Play Framework Extending HGNNs on Heterogeneous Temporal Graphs

Huan Liu, Pengfei Jiao, *Member, IEEE*, Xuan Guo, Huaming Wu, *Senior Member, IEEE*, Mengzhou Gao, and Jilin Zhang, *Member, IEEE*

***Abstract*—Heterogeneous graphs (HGs) with multiple entity and relation types are common in real-world networks. Heterogeneous graph neural networks (HGNNs) have shown promise for learning HG representations. However, most HGNNs are designed for static HGs and are not compatible with heterogeneous temporal graphs (HTGs). A few existing works have focused on HTG representation learning but they care more about how to capture the dynamic evolutions and less about their compatibility with those well-designed static HGNNs. They also handle graph structure and temporal dependency learning separately, ignoring that HTG evolutions are influenced by both nodes and relationships. To address this, we propose HGN2T, a simple and general framework that makes static HGNNs compatible with HTGs. HGN2T is plug-and-play, enabling static HGNNs to leverage their graph structure learning strengths. To capture the relationship-influenced evolutions, we design a special mechanism coupling both the HGNN and sequential model. Finally, through joint optimization by both detection and prediction tasks, the learned representations can fully capture temporal dependencies from historical information. We conduct several empirical evaluation tasks, and the results show our HGN2T can adapt static HGNNs to HTGs and overperform existing methods for HTGs.**

***Index Terms*—Heterogeneous temporal graph, graph representation learning, heterogeneous graph neural network.**

I. Introduction

Graph Neural Networks (GNNs) represent and learn graph data combined with neural networks [1], [2], [3], [4], has driven advances in many domains such as social network analysis [5], [6], [7], bioinformatics [8], [9], [10], [11], and recommender systems [12], [13]. However, real-world networks tend to be Heterogeneous Graphs (HGs) that contain multiple types of entities and relations [14], [15], [16]. To better capture the complex structural and semantic information in HGs, Heterogeneous Graph Neural Networks (HGNNs) have been proposed and aroused extensive research enthusiasm [17], [18], [19], [20].

In recent years, a large number of HGNNs designed for different domains of HGs have been proposed and achieved remarkable achievements [20], [21], [22]. Generally, the process of HGNN representation learning can be generalized as 1) modeling different aspects and then 2) aggregating and updating node representations. For example, R-GCN [23] takes different types of neighbors as aspects and transforms them separately using regularized weights. Furthermore, HAN [18] treats different meta-paths as aspects and aggregates them by the semantic-level attention mechanism. More details about the HGNNs model are introduced in [16], [20]. Generally, different aspects reflect different structural and semantic information. Therefore, to improve the performance of HGs in different domains, the existing HGNN encoding architectures are carefully designed with prior knowledge [24], [25], [26].

Although HGNNs in many domains have achieved excellent performance in various downstream tasks, they focus only on HGs containing static nodes and edges. However, real-world networks are not only heterogeneous but also usually exhibit dynamic evolution over time, such as entities and relationships adding/disappearing [27], [28]. Therefore, such networks, which are both heterogeneous and dynamic, are more suitable to be modeled as Heterogeneous Temporal Graphs (HTGs) [29], [30], [31]. When faced with HTGs, static HGNNs cannot model temporal dependencies in dynamic evolution and thus cannot be applied to tasks involving temporal evolution, especially link prediction. To alleviate this issue, a few studies on Heterogeneous Temporal Graph Neural Networks (HTGNNs) are proposed [19], [27], [32], [33]. HTGNN [27] uses hierarchical intra- and inter-relation attention to deal with the complex topology of HTGs, and an across-time attention to model temporal evolution. Similarly, DyHATR [33] equips node- and edge-level attention mechanisms to model heterogeneity and a temporal attentive recurrent neural network to capture the temporal dependencies.

Manuscript received 15 July 2023; revised 19 December 2023; accepted 22 January 2024. Date of publication 14 February 2024; date of current version 4 September 2024. This work was supported in part by the National Natural Science Foundation of China under Grant 62372146 and Grant 62003120, in part by the Fundamental Research Funds for the Provincial Universities of Zhejiang under Grant GK229909299001-008, in part by Zhejiang Laboratory Open Research Project under Grant K2022QA0AB01, and in part by Hangzhou Artificial Intelligence Major Scientific and Technological Innovation Project under Grant 2022AIZD0114. Recommended for acceptance by Y. Tong. *(Corresponding authors: Pengfei Jiao; Jilin Zhang.)*

Huan Liu is with the School of Computer, Hangzhou Dianzi University, Hangzhou 310018, China (e-mail: huanliu@hdu.edu.cn).

Pengfei Jiao, Mengzhou Gao, and Jilin Zhang are with the School of Cyberspace, Hangzhou Dianzi University, Hangzhou 310018, China (e-mail: pjiao@hdu.edu.cn; mzgao@hdu.edu.cn; jilin.zhang@hdu.edu.cn).

Xuan Guo is with the College of Intelligence and Computing, Tianjin University, Tianjin 300350, China (e-mail: guoxuan@tju.edu.cn).

Huaming Wu is with the Center for Applied Mathematics, Tianjin University, Tianjin 300072, China (e-mail: whming@tju.edu.cn).

Digital Object Identifier 10.1109/TBDATA.2024.3366085

Despite the above methods having made some progress, their intrinsic design leads to two drawbacks: 1) they focus more on the dynamic evolution of HTGs and less on compatibility with those well-designed static HGNNs. As a consequence, they cannot adequately capture effective information, and a large number of research results of HGNNs in the community are wasted. 2) The processes of HG modeling and temporal evolution modeling are decoupled, which ignores the influence of relationships with other nodes and results in significant performance degradation.

In order to apply existing well-designed HGNNs to ubiquitous HTGs and alleviate the above two drawbacks, we consider: *How do we design a framework that not only extends existing well-designed HGNNs to be temporal but also improves the generalization of HTGNNs?* However, designing such a framework is not a trivial task. First, challenging dynamic tasks, such as link prediction, need to capture useful content from the large amount of historical dependencies information of HTGs. However, HGNNs cannot model temporal information in HTGs, and thus cannot mine useful content from historical information. Second, the historical information existing in HTGs changes constantly as the topology evolves over time, and static HGNNs need to update the historical information according to those topological changes. Although HGNNs have excellent heterogeneity modeling capabilities, they are difficult to capture and update historical information from the topology of multiple snapshots of HTGs.

In this article, we propose a simple but Plug-and-Play framework extending HGNNs on HTGs, named HGN2T. The HGN2T framework mainly consists of two modules: *Historical Feature Incorporation (HFI)* and *Topology Evolution Update (TEU)* modules. Specifically, HGN2T first integrates the current node attributes and the historical features derived from the previous timestamp snapshots through the HFI module. Then, HGN2T performs heterogeneous message-passing on the integrated node representation through HGNN on the current topology structure to mine valuable information. Please note that the HGNN can be flexibly replaced according to HGs with different properties. After being well trained, HGN2T will effectively combine historical features and capture the current HG topology to complete link detection, so the representations are called detective embeddings. On the other hand, to avoid the decoupling of the HG modeling and temporal evolution modeling processes, HGN2T introduces the neighbor structure into the update process of temporal evolution modeling through the TEU module. Topology-aware predictive node representations, i.e., predictive embeddings, significantly outperform existing decoupled methods in link prediction experiments. Finally, HGN2T jointly optimizes the detection and prediction tasks to learn representations from HTGs that fully capture rich structural information and temporal dependencies.

To evaluate the proposed HGN2T framework, we extend two classical HGNNs as examples and perform three types of link prediction tasks on three real-world datasets. The promising results and obvious performance improvements demonstrate the effectiveness of the proposed framework.

In summary, we make the following contributions:

- We study the problem that HGNNs cannot handle with HTGs and current HTGNNs are difficult to flexibly combine with existing HGNNs to generalize to different HTGs. There is an urgent need to extend the existing well-designed HGNN to be temporal.
- We design a unified temporal HGNN framework, HGN2T. Through the HFI and TEU modules, HGN2T can easily extend existing HGNN to handle HTGs by jointly optimizing the detection and prediction tasks.
- We exemplarily extend two classical HGNNs to be temporal and conduct thorough experiments on three real-world datasets to evaluate the effectiveness of our framework and outperform state-of-the-art baselines.

The remainder of this article is organized as follows. First, we introduce the related work of this article in Section II. Then, we provide the preliminary concepts for this article in Section III. Then we introduced the HGN2T framework in detail in Section IV, and conducted experimental evaluations on the performance of the framework in Section V. Finally, we summarize the article in Section VI.

II. Related Work

A. Heterogeneous Graph Neural Networks

HGNNs aim to capture structural information and semantic information in HGs through deep neural networks [34], [35]. According to the different properties of HGs, a large number of well-designed HGNNs have been proposed [18], [36], [37], [38].

To model a knowledge graph with multiple relations, R-GCN [23] is proposed to transform different types of neighbors separately using regularized weight matrices. Furthermore, HAN [18] is used to model HGs that contain rich semantic information in the meta-path graph. To be able to take into account the features of those intermediate nodes, MAGNN [39] generates node representations through intra- and inter-metapath attention. For HGs that are difficult to design meta-paths, HGT [21] incorporates nodes by learning attention weights for each meta relation through node- and edge-type dependent attention mechanisms.

The above well-designed HGNNs carefully explored the characteristics of HGs and achieved remarkable achievements. However, they do not consider temporal evolution and thus cannot effectively model dynamic networks, which are ubiquitous in the real world.

B. Dynamic Graph Neural Networks

Dynamic GNNs are designed to collaborate with GNN and sequence models so can adapt and capture the topology evolution of dynamic graphs [40], [41], [42].

For instance, VGRNN [41] combines GCN and RNN to capture the dynamic topology and improve expression power by modeling the uncertainty of hidden representations of nodes. To cope with the drastically changing node sets, EvolveGCN [43] is proposed to use RNN to model the evolution characteristics of GCN parameters instead of node features. Besides, to alleviate the long-term forgetting and poor scalability of large-scale graphs due to the RNN mechanism in the above methods, DySAT [42] jointly models structural neighbors and temporal dynamics by a self-attention mechanism to capture the graph structure evolution. In addition, ROLAND [44] and WinGNN [45] propose a dynamic GNN training framework based on meta-learning. HGWaveNet [46] proposes to model hierarchically structured dynamic graphs through hyperbolic dilated causal convolution and hyperbolic GRU and shows a relative improvement over SOTA methods.

The above model effectively models dynamic graphs with different characteristics and achieves excellent results. However, the above homogeneous GNN models have limitations and perform poorly when dealing with HTG containing complex evolutionary characteristics.

C. Heterogeneous Temporal Graph Neural Networks

Recently, a small number of HTGNNs have been proposed to simultaneously capture temporal dependencies and semantic heterogeneous structural [19], [27], [33], [47].

HTGNNs based on hierarchical attention mechanisms have been widely proposed, such as HTGNN [27] models HG through intra- and inter-relation hierarchical attention mechanisms and then aggregates the resulting node representations by temporal attention mechanisms. Besides, HDGAN [47] processes each snapshot through structural- and semantic-level attention mechanisms, and then obtains historical influence through time-level attention and applies it to the Hawkes process. On the other hand, DyHATR [33] considers that node embeddings at different times have different importance and uses a hierarchical attention mechanism to model each HG snapshot separately, and then the attention aggregates the hidden states of the RNN to obtain node representations.

By modeling temporal dependencies, the above model achieves significant performance improvements. However, these methods cannot effectively integrate with existing domain-oriented HGNNs, failing to generalize to diverse HTGs in different scenarios. Furthermore, these models ignore the influence of neighbor nodes in the network topology when modeling temporal dependencies, i.e., the temporal neighbors of the target node only have themselves in different snapshots. Therefore, the heterogeneous topology and attribute information of neighbors in historical snapshots will not be fully captured, resulting in performance degradation.

III. Preliminary

In this section, we define some of the concepts that will be used in this article and explain the formal representation, and we will use these notations throughout the article. The key representations used in this article and their explanations are summarized in Table I.

A. Definitions

Definition III.1 (Heterogeneous Graph): A *Heterogeneous Graph* is defined as a graph $G = (\mathcal{V}, \mathcal{E}, \mathcal{A}, \mathcal{R}, \phi, \psi)$, where $\mathcal{V}$ and $\mathcal{E}$ denote sets of nodes and edges, and it is associated with a node type mapping function $\phi : \mathcal{V} \rightarrow \mathcal{A}$ and an edge type mapping function $\psi : \mathcal{E} \rightarrow \mathcal{R}$, where $\mathcal{A}$ and $\mathcal{R}$ denote sets of object and link types, and $|\mathcal{A}| + |\mathcal{R}| > 2$.

TABLE I
Notations and Explanations

Notations	Explanations
G	A heterogeneous graph.
$\mathcal{G}$	A heterogeneous temporal graph.
$\mathcal{V}, \mathcal{E}$	The set of nodes and edges.
$\mathcal{A}, \mathcal{R}$	The set of node types and edge types.
$\mathcal{R}_i$	The relation set of node i.
$\mathcal{N}_i^r$	The relation-r-based neighbors of node i.
$\mathbf{X}$	The raw node attribute matrix.
$\mathbf{m}_{i,r}$	The massage vector of node i under relation r.
$\tilde{\mathbf{H}}, \mathbf{H}$	The projected and historical node feature.
$\mathbf{Z}, \hat{\mathbf{Z}}$	The detective and predictive node embedding.
$\mathbf{S}^t$	The hidden state matrix calculated from timestamp t.
$\mathbf{W}, \mathbf{b}$	The trainable transformation matrix and bias vector.
$\phi(\cdot), \psi(\cdot)$	The node and edge type mapping function.
$\sigma(\cdot)$	The nonlinear activation function.
$\Phi(\cdot)$	A heterogeneous graph neural network.

Given a node $i \in \mathcal{V}$, its *relation set* $\mathcal{R}_i$ is defined as the set of edge types connected to node i, denoted as $\mathcal{R}_i = \{\psi(i, j) \mid (i, j) \in \mathcal{E}\}$; its *relation-r-based neighbors* $\mathcal{N}_i^r$ is defined as the set of first-order neighbor nodes connected to it through edge type r, denoted as $\mathcal{N}_i^r = \{j \mid (i, j) \in \mathcal{E}, \psi(i, j) = r\}$.

Definition III.2 (Heterogeneous Temporal Graph): A *Heterogeneous Temporal Graph* is a list of observed heterogeneous snapshots $\mathcal{G} = \{G^1, G^2, \ldots, G^T\}$ ordered by timestamps, where T is the number of timestamps and $G^t = (\mathcal{V}^t, \mathcal{E}^t, \mathcal{A}, \mathcal{R}, \phi, \psi)$ represents the tth snapshot. The node set $\mathcal{V}^t$ and edge set $\mathcal{E}^t$ differs between snapshots, representing dynamic addition and removal of nodes and edges.

Definition III.3 (Heterogeneous Temporal Graph Representation Learning): Given a heterogeneous temporal graph $\mathcal{G}$, the *heterogeneous temporal graph representation learning* is to learn a non-linear mapping function that encodes node $i \in \mathcal{V}^t$ into a d-dimensional node representation $\mathbf{z}_i^t \in \mathbb{R}^d$ and $d \ll |\mathcal{V}^t|$. The node representations can capture both spatial heterogeneity and temporal dependencies involved in the heterogeneous temporal graph $\mathcal{G}$.

Definition III.4 (Link Detection and Prediction): Given a heterogeneous temporal graph $\mathcal{G} = \{G^t\}_{t=1}^T$ and the node representations $\mathbf{Z}^T$ learned from it, the *link detection* is the problem of predicting the probability $p = f(\mathbf{z}_i^t, \mathbf{z}_j^t)$ of an edge $(i, j) \in \mathcal{E}^t$ between a node pair i and j at timestamp $t \leq T$, which is also known as the *interpolation* or *completion* problem, which belongs to the *transductive learning* setting.

Link prediction is the problem of predicting the probability $p = g(\mathbf{z}_i^T, \mathbf{z}_j^T)$ that an edge $(i, j) \in \mathcal{E}^\tau$ between a node pair i and j at timestamp $\tau > T$, which is also known as the *extrapolation* problem, which belongs to the *inductive learning* setting.

In this article, the node embeddings used in the link detection problem and the link prediction problem are named *detective embedding* and *predictive embedding*, respectively.

B. Heterogeneous Graph Neural Networks

Unlike GNNs on homogeneous graphs, HGNNs need to model different types of nodes and edges. Based on the existing studies of message-passing GNNs [48], [49], the forward propagation of HGNNs can be summarized as two modules, the message function and the update function. Formally, the lth layer of an HGNN is defined as

$$\mathbf{m}_{i,r}^{(l)} = \mathrm{MSG}_r\left(\mathbf{h}_i^{(l)}, \left\{\mathbf{h}_j^{(l)}, \forall j \in \mathcal{N}_i^r\right\}\right), \tag{1}$$

$$\mathbf{h}_i^{(l+1)} = \mathrm{UPDATE}\left(\mathbf{h}_i^{(l)}, \left\{\mathbf{m}_{i,r}^{(l)}, \forall r \in \mathcal{R}_i\right\}\right), \tag{2}$$

where $\mathbf{m}_{i,r}^{(l)}$ is the message vector of target node i under relation type r. After obtaining message vectors of all relation $r \in \mathcal{R}_i$, HGNN aggregates and updates them to get the node representations at $l+1$th layer by the update function. The message function and the update function are defined and implemented differently by the particular HGNN algorithms.

Relational Graph Convolutional Networks (R-GCN [23]): The message function of R-GCN uses regularized weights to transform relation-r-based neighbor nodes. Then, the update function sums the transferred target node representations and message vectors and then inputs them to a nonlinear activation function. Specifically, the representation of node i is updated as follows:

$$\mathbf{m}_{i,r}^{(l)} = \sum_{j \in \mathcal{N}_i^r} \frac{1}{c_{i,r}} \cdot \mathbf{W}_r^{(l)} \cdot \mathbf{h}_j^{(l)}, \tag{3}$$

$$\mathbf{h}_i^{(l+1)} = \sigma\left(\mathbf{W}_0^{(l)} \cdot \mathbf{h}_i^{(l)} + \sum_{r \in \mathcal{R}_i} \mathbf{m}_{i,r}^{(l)}\right), \tag{4}$$

where $\mathbf{W}_r^{(l)}$ and $\mathbf{W}_0^{(l)}$ are the transformation matrices of relation-r-based neighbors and self-loops, respectively; $c_{i,r}$ is a normalization constant; $\sigma(\cdot)$ is an element-wise activation function like ReLU.

Hierarchical Attention Model (HA): HA is widely used in many studies [18], [22], [27], [29], [33], which compute node-level and edge-level attention weights and aggregate neighbor nodes by different attention weights. Specifically, the message and update function of HA is implemented as

$$\mathbf{m}_{i,r}^{(l)} = \sigma\left(\sum_{j \in \mathcal{N}_i^r} \alpha_{i,j}^{(l)} \cdot \mathbf{W}_r^{(l)} \cdot \mathbf{h}_j^{(l)}\right), \tag{5}$$

$$\mathbf{h}_i^{(l+1)} = \sum_{r \in \mathcal{R}_i} \beta_{i,r}^{(l)} \cdot \mathbf{m}_{i,r}^{(l)}, \tag{6}$$

where $\alpha_{i,j}^{(l)}$ and $\beta_{i,r}^{(l)}$ are node-level and edge-level attention weights, respectively; $\sigma(\cdot)$ is an element-wise activation function like ReLU. The implementations of them may be trivially different according to different attention mechanisms.

IV. HGN2T Framework

A. Overview

The proposed HGN2T framework mainly consists of two modules: *Historical Feature Incorporation (HFI)* and *Topology Evolution Update (TEU)* modules. To enable HGNNs to capture valuable parts from historical information, the HFI module incorporates historical information into the encoding process of HGNNs. The detective node embedding learned by the HFI module is used to perform the link detection task. Then, the TEU module updates the historical information based on the topology. The updated historical information contains the dynamic evolution dependencies up to the current snapshot, and we use its transformed vector as predictive node embedding to complete the link prediction task. Finally, by jointly optimizing the link detection and prediction tasks, the representations learned by the entire HGN2T framework can fully capture temporal dependencies from historical information. The overall structure of the proposed HGN2T framework is shown in Fig. 1.

B. Historical Feature Incorporation

There are rich historical dependencies contained in HG snapshots of HTGs. Therefore, we use HGNNs to incorporate topological structures in each timestamp to capture useful information from historical dependencies. The HFI module mainly consists of three steps: raw feature projection, historical feature combination, and detective embedding.

Raw Feature Projection: In a HG G^t, different types of raw features $\mathbf{x}^t$ may have different dimensions and are distributed in different feature spaces. Therefore, we first transform all types of node features into a common latent vector space through raw feature projection. Specifically, for the raw feature of node i of type $\phi(i)$ at timestamp $t \in \{1, 2, \ldots, T\}$, we perform the following transformations:

$$\tilde{\mathbf{h}}_i^t = \sigma\left(\mathbf{W}_{\phi(i)} \cdot \mathbf{x}_i^t + \mathbf{b}_{\phi(i)}\right), \tag{7}$$

where $\mathbf{x}_i^t \in \mathbb{R}^{d'}$ and $\tilde{\mathbf{h}}_i^t \in \mathbb{R}^d$ is the d'-dimensional raw feature and d-dimensional projected feature of node i, respectively; $\mathbf{W}_{\phi(i)} \in \mathbb{R}^{d \times d'}$ and $\mathbf{b}_{\phi(i)} \in \mathbb{R}^d$ are a trainable transformation and bias matrix for the type of node $\phi(i)$; $\sigma(\cdot)$ is a nonlinear activation function, such as ReLU.

Historical Feature Combination: We combine the projected node features $\tilde{\mathbf{h}}_i^t$ with the hidden state $\mathbf{s}_i^{t-1}$ containing history dependencies through a combiner to obtain the input features $\mathbf{h}_i^t$ of the HGNNs. Specifically, we aggregate the projected features $\tilde{\mathbf{h}}_i^t$ and hidden states $\mathbf{s}_i^{t-1}$ as follows:

$$\mathbf{h}_i^t = \mathrm{COMB}\left(\tilde{\mathbf{h}}_i^t, \mathbf{s}_i^{t-1}\right), \tag{8}$$

where COMB is a binary combiner of projected vector and hidden state, such as element-wise Hadamard product and concatenation; $\mathbf{s}_i^{t-1} \in \mathbb{R}^d$ is the d-dimensional hidden state generated by the HG-RNN described in Section IV-C. At the first snapshot or for the unobserved new nodes, we initialize the hidden state as $\mathbf{0}$.

Detective Embedding: We use an HGNN to learn the detective node embeddings $\mathbf{Z}^t$ from features containing historical

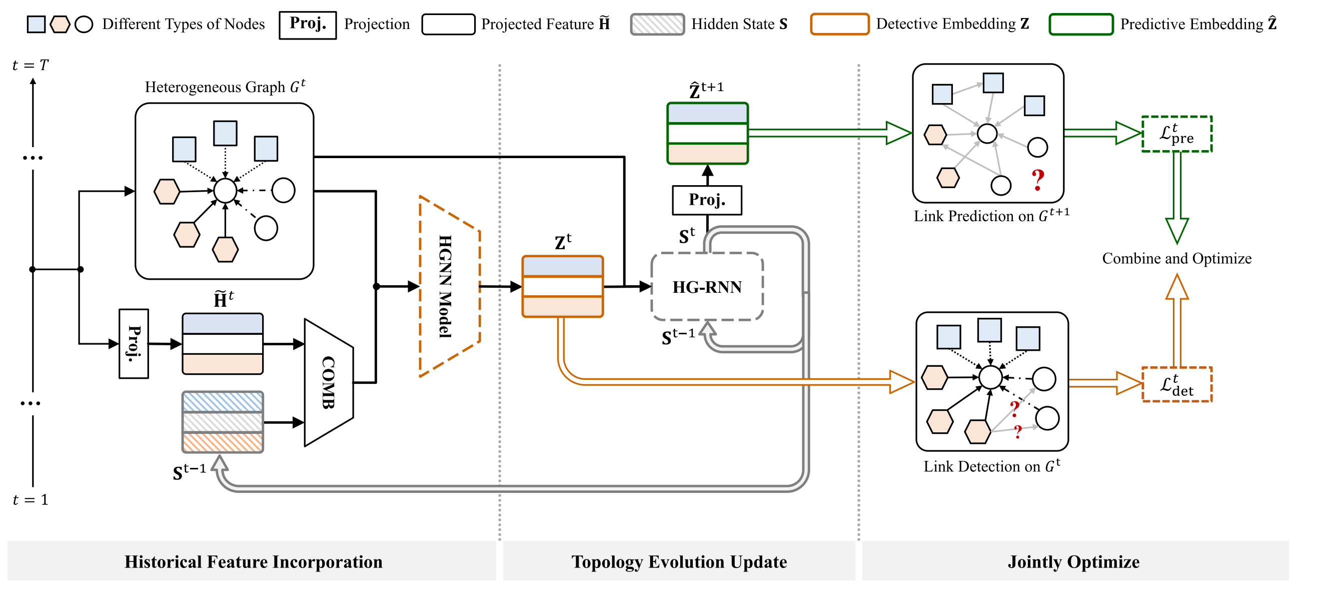

Different Types of Nodes
Proj. Projection
Projected Feature $\tilde{\mathbf{H}}$
Hidden State $\mathbf{S}$
Detective Embedding $\mathbf{Z}$
Predictive Embedding $\hat{\mathbf{Z}}$
$t = T$
$t = 1$
Heterogeneous Graph G^t
Proj.
$\tilde{\mathbf{H}}^t$
COMB
$\mathbf{S}^{t-1}$
HGNN Model
$\mathbf{Z}^t$
$\hat{\mathbf{Z}}^{t+1}$
Proj.
$\mathbf{S}^t$
HG-RNN
$\mathbf{S}^{t-1}$
Link Prediction on G^{t+1}
$\mathcal{L}^t_{\text{pre}}$
Combine and Optimize
Link Detection on G^t
$\mathcal{L}^t_{\text{det}}$
Historical Feature Incorporation
Topology Evolution Update
Jointly Optimize

Fig. 1. Overall architecture of the proposed HGN2T framework. The framework mainly includes two modules: *Historical Feature Incorporation (HFI)* and *Topology Evolution Update (TEU)* modules. The HFI module equips an existing well-designed HGNN to learn the relationship between historical features in $\mathbf{S}^{t-1}$ and current snapshot graph G^t. Then, the TEU module updates the topology-aware historical information based on the topology. The whole framework is jointly optimized by the link detection and the link prediction tasks.

information in each timestamp $t \in [1, T]$ to capture structural information, which is crucial in detection tasks. Note that we can flexibly replace suitable HGNNs for different HGs. Specifically, given a HG snapshot G^t and a historical feature representation $\mathbf{H}^t$, the detective node embedding is calculated as follows:

$$\mathbf{Z}^t = \Phi\left(G^t, \mathbf{H}^t\right), \tag{9}$$

where $\Phi(\cdot)$ can be any well-designed HGNN defined in Section III-B.

Through a single layer of HGNN, the information of the first-order heterogeneous neighborhood in the current snapshot can be aggregated. In order to obtain information on multi-order neighborhoods, the number of layers of HGNN here can be stacked, which we analyzed in Section V-E.

C. Topology Evolution Update

The temporal dependencies existing among snapshots of HTGs are crucial for link prediction tasks. However, existing research usually separates graph structure learning and temporal dependencies modeling, i.e., the temporal neighbors are only nodes themselves. Neglecting the relationship with other nodes in the historical snapshot will be unable to directly carry out message transmission and aggregation of heterogeneous neighborhood nodes according to the topology, resulting in insufficient capture of heterogeneous topology information and temporal dependencies information.

To address this issue, we introduce heterogeneous network topology into the process of modeling and updating the temporal evolution by the TEU module. Specifically, we first learn a hidden state $\mathbf{S}^t$ to preserve historical dependencies up to timestamp t. Then, we adopt HGNNs to capture the relationship effects on each HG snapshot to the recurrent hidden state. Specifically, we implement two TEU methods, HG-LSTM and HG-GRU.

HG Long-Short-Term Memory: Given a HG G^t with its detective node embeddings $\mathbf{Z}^t$ generated from Section IV-B at each time step $t \in \{1, 2, \ldots, T\}$, the update process of HG-LSTM is as follows:

$$\begin{aligned}
\mathbf{I}^t &= \sigma\left(\Phi_{ZI}\left(G^t, \mathbf{Z}^t\right) + \Phi_{SI}\left(G^t, \mathbf{S}^{t-1}\right)\right),\\
\mathbf{F}^t &= \sigma\left(\Phi_{ZF}\left(G^t, \mathbf{Z}^t\right) + \Phi_{SF}\left(G^t, \mathbf{S}^{t-1}\right)\right),\\
\mathbf{O}^t &= \sigma\left(\Phi_{ZO}\left(G^t, \mathbf{Z}^t\right) + \Phi_{SO}\left(G^t, \mathbf{S}^{t-1}\right)\right),\\
\tilde{\mathbf{C}}^t &= \tanh\left(\Phi_{ZC}\left(G^t, \mathbf{Z}^t\right) + \Phi_{SC}\left(G^t, \mathbf{S}^{t-1}\right)\right),\\
\mathbf{C}^t &= \mathbf{F}^t \odot \mathbf{C}^{t-1} + \mathbf{I}^t \odot \tilde{\mathbf{C}}^t,\\
\mathbf{S}^t &= \mathbf{O}^t \odot \tanh\left(\mathbf{C}^t\right),
\end{aligned} \tag{10}$$

where $\Phi_{\square}$ denotes not shared HGNNs in different inputs and gates; $\mathbf{I}^t$, $\mathbf{F}^t$ and $\mathbf{O}^t$ are the input gate, forget gate and output gate matrix; $\mathbf{C}^t$ and $\tilde{\mathbf{C}}^t$ are the memory cells and candidate memory cells matrix; $\mathbf{S}^t$ is the hidden state matrix.

HG Gated Recurrent Unit: Given G^t with its detective node embeddings $\mathbf{Z}^t$ at each time step $t \in \{1, 2, \ldots, T\}$, HG-GRU update the hidden state as follows:

$$\begin{aligned}
\mathbf{U}^t &= \sigma\left(\Phi_{ZU}\left(G^t, \mathbf{Z}^t\right) + \Phi_{SU}\left(G^t, \mathbf{S}^{t-1}\right)\right),\\
\mathbf{R}^t &= \sigma\left(\Phi_{ZR}\left(G^t, \mathbf{Z}^t\right) + \Phi_{SR}\left(G^t, \mathbf{S}^{t-1}\right)\right),\\
\tilde{\mathbf{S}}^t &= \tanh\left(\Phi_{US}\left(G^t, \mathbf{U}^t\right) + \Phi_{SS}\left(G^t, \mathbf{R}^t \odot \mathbf{S}^{t-1}\right)\right),\\
\mathbf{S}^t &= \mathbf{U}^t \odot \mathbf{S}^{t-1} + \left(1 - \mathbf{U}^t\right) \odot \tilde{\mathbf{S}}^t,
\end{aligned} \tag{11}$$

where $\Phi_{\square}$ denotes unshared HGNNs; where $\mathbf{U}^t$ and $\mathbf{R}^t$ are the update and reset gate matrix; $\sigma(\cdot)$ and $\odot$ are an activation function and an element-wise Hadamard product operator. All

HGNNs $\Phi_{\square}$ in both (10) and (11) can be replaced according to different HTGs.

Similarly, the number of HGNN layers here can also be stacked to capture the temporal dependencies and heterogeneous topology information of multi-order temporal neighborhoods. We analyze and report the performance of our framework with different stacked layers of HG-RNN in Section V-E.

Predective Embedding: After obtaining the updated hidden state $\mathbf{S}^t$ at each timestamp $t \in \{1, 2, \ldots, T\}$ via HG-GNN, we transform it as the predictive node embedding $\hat{\mathbf{z}}_i^{t+1}$ to perform prediction tasks in the next snapshot G^{t+1}

$$\hat{\mathbf{z}}_i^{t+1} = \sigma\left(\mathbf{W}_s \cdot \mathbf{s}_i^t + \mathbf{b}_s\right), \tag{12}$$

where $\mathbf{W}_s \in \mathbb{R}^{d\times d}$ and $\mathbf{b}_s \in \mathbb{R}^d$ are hidden state transformation and bias matrix; $\sigma(\cdot)$ is a nonlinear activation function such as ReLU.

D. Objective Function

The purpose of the HGN2T framework is to extend the HGNNs to be temporal so that detective and predictive node embeddings can be learned from HTGs. To achieve this, we optimize the overall model by reconstructing the network topologies in HTGs.

For both detective and predictive node embeddings, we use a shared discriminator to calculate the probability of edge presence. Specifically, for the embedding representation $(\mathbf{z}_i, \mathbf{z}_j)$ of a pair of nodes, we obtain the existence probability of edges as follows:

$$\hat{P}\left((i,j) \in \mathcal{E}^t \mid \mathbf{z}_i, \mathbf{z}_j\right) = \mathcal{D}\left(\mathbf{z}_i, \mathbf{z}_j\right) = \sigma\left(\text{MLP}\left(\mathbf{z}_i \| \mathbf{z}_j\right)\right), \tag{13}$$

where $\|$ denotes the concatenation; MLP represents a two-layer perceptron model; $\sigma(\cdot)$ is a nonlinear activation function Sigmoid to calculate the probabilities.

For the detection task, to effectively capture the structural information of the current snapshot, we reconstruct the current network topology using detective node embedding at each timestamp. Specifically, in the tth snapshot we compute the detective loss as follows:

$$\mathcal{L}_{\text{det}}^t = \sum_{i\in\mathcal{V}^t}\sum_{j\in\mathcal{N}_i^t}\left\{-\log\left(\mathcal{D}(\mathbf{z}_i^t, \mathbf{z}_j^t)\right)\right. \\ \left. +Q \cdot \mathbb{E}_{v_n\sim P_n^t(i)} \log\left(\mathcal{D}(\mathbf{z}_i^t, \mathbf{z}_{v_n}^t)\right)\right\}, \tag{14}$$

where $P_n^t(i)$ and Q are the negative sampling distribution and the number of negative samples, respectively.

To make the extended HGNN model better capture the temporal dependencies between snapshots, we also optimize the link prediction task. Similarly, predictive loss is computed here using predictive node embeddings as follows:

$$\mathcal{L}_{\text{pre}}^t = \sum_{i\in\mathcal{V}^t}\sum_{j\in\mathcal{N}_i^t}\left\{-\log\left(\mathcal{D}(\hat{\mathbf{z}}_i^t, \hat{\mathbf{z}}_j^t)\right)\right. \\ \left. +Q \cdot \mathbb{E}_{v_n\sim P_n^t(i)} \log\left(\mathcal{D}(\hat{\mathbf{z}}_i^t, \hat{\mathbf{z}}_{v_n}^t)\right)\right\}, \tag{15}$$

where $P_n^t(i)$ and Q are also the negative sampling distribution and the number of negative samples, same as (14).

TABLE II
STATISTICS OF THE DATASETS

Dataset	Avg. # Nodes	Avg. # Edges	# Snapshots
DBLP	Author (A): 8,470 Paper (P): 9,025 Venue (V): 1,074	A-P: 8,056 A-V: 8,056 P-V: 1,903	12
AMiner	Author (A): 8,882 Paper (P): 7,289 Venue (V): 1,970	A-P: 7,538 A-V: 7,538 P-V: 1,634	12
Yelp	Business (B): 771 User (U): 1,452 Star (S): 5	B-S: 1,080 B-U: 1,080 U-S: 1,080	9

By jointly optimizing the detection and prediction losses of each snapshot in HTG, the model can effectively capture the heterogeneous structure and temporal evolution information in two adjacent snapshots. The overall objective function is defined as follows:

$$\mathcal{L} = \sum_{t=1}^{T}\left(\mathcal{L}_{\text{det}}^t + \mathcal{L}_{\text{pre}}^t\right) + \lambda \cdot L_p, \tag{16}$$

where L_p is the penalty term to prevent over-fitting, i.e., L_2 regularization; λ is a hyper-parameter to the control penalty function.

E. Complexity Analysis

For the Feature Projection module in each snapshot, the computational complexity is $\mathcal{O}(|\mathcal{V}^t|dd')$, where $|\mathcal{V}^t|$ is the node number for snapshot G^t, d and d' are the dimensions of node representation and raw feature respectively. For the TEU module, the computational complexity is $\mathcal{O}(TP + |\mathcal{V}|Td)$, where T is the snapshot numbers, P depends on the HGNN module. The computational complexity of the detective and predictive loss is $\mathcal{O}(|\bar{\mathcal{E}}|TQd)$, where $|\bar{\mathcal{E}}|$ is the average edge sets size of snapshots, Q is the negative sample number. Overall, the complexity of HGN2T is $\mathcal{O}(|\bar{\mathcal{V}}|Tdd' + |\bar{\mathcal{E}}|TQd + TP)$, where $|\bar{\mathcal{V}}|$ the average node sets size of snapshots, which is computational highly efficient.

V. EXPERIMENTS

A. Datasets and Baselines

To evaluate the effectiveness of the proposed HGN2T framework, we extend the two HGNNs introduced in Section III-B and conduct experiments on three real-world datasets. The statistics of the dataset are summarized in Table II.

- *DBLP* is a computer science bibliography. In this article, we construct 12 network snapshots that contain 8,470 authors, 9,025 papers, and 1,074 venues.
- *AMiner* is an academic search engine that helps us to mine information from academic networks. We extract a subset of AMiner which contains 12 network snapshots with 8,882 authors, 7,289 papers, and 1,970 venues.
- *Yelp* records users rating on local business and social relations. Here we extract a subset of Yelp that is divided into 10 network snapshots by year and consists of 771 businesses, 1,452 users, and 5 stars.

TABLE III
AUC AND AP SCORES (MEAN ± SD %) OF THE LINK DETECTION TASK

Model	DBLP		AMiner		Yelp		Avg. Rank
	AUC	AP	AUC	AP	AUC	AP	
VGAE	95.32 ± 2.07	95.00 ± 2.03	98.75 ± 1.65	98.89 ± 1.32	51.60 ± 0.70	52.26 ± 1.58	10.17
GATv2	92.86 ± 0.42	90.79 ± 0.90	94.71 ± 0.28	95.46 ± 0.30	63.09 ± 1.58	63.30 ± 1.98	10.83
EvolveGCN	82.49 ± 5.02	83.13 ± 3.92	80.42 ± 9.37	85.93 ± 6.55	52.45 ± 3.40	55.80 ± 2.89	12.67
VGRNN	63.13 ± 0.47	76.82 ± 0.32	70.76 ± 0.81	82.57 ± 0.52	48.91 ± 3.07	65.18 ± 1.97	13.67
DySAT	97.12 ± 0.58	96.66 ± 0.79	99.47 ± 0.16	99.41 ± 0.17	78.68 ± 1.17	71.48 ± 1.68	4.83
HGWaveNet	69.37 ± 2.52	74.17 ± 2.69	79.94 ± 3.51	83.30 ± 3.07	56.39 ± 1.43	57.18 ± 1.36	13.67
Mp2vec	78.97 ± 1.69	74.79 ± 2.29	85.15 ± 2.20	83.87 ± 2.96	71.22 ± 1.63	65.50 ± 1.39	11.33
HGT	92.88 ± 2.13	91.21 ± 2.93	95.77 ± 0.63	95.84 ± 0.80	64.26 ± 2.15	61.43 ± 1.90	10.33
RGCN	93.53 ± 1.93	93.23 ± 2.03	96.56 ± 1.59	97.14 ± 1.56	72.59 ± 0.99	71.93 ± 0.89	7.83
HA	97.95 ± 0.26	**97.65 ± 0.40**	99.36 ± 0.20	99.34 ± 0.22	68.42 ± 1.40	62.97 ± 1.74	6.17
HTGNN	97.60 ± 0.52	96.98 ± 0.46	99.58 ± 0.23	99.56 ± 0.20	72.47 ± 2.44	70.76 ± 3.39	4.50
HGN2T$_{\text{RGCN-LSTM}}$	97.06 ± 0.25	95.91 ± 0.34	99.46 ± 0.10	99.44 ± 0.09	75.88 ± 1.75	77.15 ± 1.57	5.17
HGN2T$_{\text{RGCN-GRU}}$	97.20 ± 0.19	96.07 ± 0.38	99.56 ± 0.05	99.57 ± 0.04	75.81 ± 4.18	73.26 ± 7.66	4.50
HGN2T$_{\text{HA-LSTM}}$	97.22 ± 0.23	96.52 ± 0.42	99.66 ± 0.05	99.60 ± 0.08	76.05 ± 1.89	73.64 ± 3.02	3.17
HGN2T$_{\text{HA-GRU}}$	**98.03 ± 0.14**	97.48 ± 0.19	**99.67 ± 0.04**	**99.60 ± 0.05**	**81.41 ± 2.20**	**83.26 ± 2.96**	1.17

To comprehensively evaluate the performance of our proposed HTN2T framework, we compare static homogeneous GNNs: *VGAE* [50], *GATv2* [51]; dynamic homogeneous GNNs *EvolveGCN* [43], *VGRNN* [41], *DySAT* [42] and *HGWaveNet* [46]; static heterogeneous methods *metapath2vec* [17], *HGT* [21], *R-GCN* [23] and *Hierarchical Attention (HA)* mechanism [18], which is a hierarchical attention structure widely used in HGNN; dynamic heterogeneous GNN *HTGNN* [27]. To verify the effectiveness of the framework, we extend R-GCN and HA and implement HGN2T$_{\text{RGCN-LSTM}}$, HGN2T$_{\text{RGCN-GRU}}$, HGN2T$_{\text{HA-LSTM}}$ and HGN2T$_{\text{HA-GRU}}$ according to different gate architectures.

B. Implementation Details

For the homogeneous graph methods (i.e., GATv2, HGWaveNet, etc.), we simply ignore the heterogeneity of nodes and edges in the graph. For the static graph approaches (i.e., VGAE, GATv2, etc.), we analyze each snapshot separately, to demonstrate the importance of timing dependencies. For random walk-based methods metapath2vec we set the number of walks per node to 40, the walk length to 60, and the window size to 5. To better utilize the information of different types of nodes, for DBLP and AMiner datasets, we set the meta-path as *{BSUSB, SUBUS, UBSBU}*. For the Yelp dataset, we set the meta-paths to *{APVPA, PVAVP, VAPAV}*. For HTGNN, we set the time window to $\{3, 5, 7\}$ and report the best results. In terms of other parameters, we follow the settings in their original papers.

For the proposed HGN2T framework, we use Glorot initialization [52] and optimize the model with Adam [53] optimizer and ReduceLROnPlateau scheduler. We set the initial learning rate of the Adam optimizer to 1e-2 and the regularization parameter to 5e-4. The dropout of attention is set to 0.2. We train all the models with 1,000 epochs and use an early stopping strategy with a patience tune from 5 to 50. We adopt grid search in the embedding dimension from 4 to 64 and set it to 32 for each of the aforementioned baseline methods to allow a fair comparison. All models are randomly trained for 5 times, and the average results of test performance are reported.

We complete the experiment on the Ubuntu 20.04.3 LTS operating system with Intel(R) Xeon(R) Silver 4210R CPU @ 2.40 GHz processor and NVIDIA GeForce RTX 3090 GPUs, the CUDA version used 11.6, and the PyTorch version is 1.12.0. We implement the proposed HGN2T framework using Deep Graph Library (DGL) 0.8.2.

C. Dynamic Link Detection

We conduct link detection experiments using the learned detective node embeddings on the last 3 snapshots and report their average results. For each time, we randomly removed 10% and sampled an equal number of negative edges as validation and test sets. We conducted five repeated experiments and reported the mean and standard deviation in Table III.

From Table III, we can see that HGN2T$_{\text{HA-GRU}}$ generally outperforms all the other methods on all datasets. First, heterogeneous GNN models, such as R-GCN and HA, generally achieve higher prediction accuracy than homogeneous GNNs, such as GATv2 and HGWaveNet, which indicates the heterogeneity of the modeling HTG has a noticeable impact on predictive performance. The reason for the poor prediction results of HGWaveNet may be that the data set is mainly composed of heterogeneous triangles, which is inconsistent with the hyperbolic manifold of negative curvature. Second, compared with static models such as R-GCN and VGAE, dynamic models such as VGRNN and HTGNN have achieved better results, indicating that historical dependency information is beneficial to modeling the current snapshot. Third, compared with HTGNN, our framework achieves better performance on the three datasets, which demonstrates that the proposed TEU is more effective than modeling topology and timing information separately. Finally, compared with R-GCN and HA, our framework achieves significant performance improvement averages of 3.29% and 4.46%, which demonstrates that the proposed framework can effectively integrate historical information and improve the model's ability to model the heterogeneity of the current network.

In summary, the link detection task evaluates the model's ability to model heterogeneous network structures by completing the current snapshot topology. The results show that the proposed framework can effectively model network heterogeneity and improve modeling accuracy by incorporating historical information.

TABLE IV
AUC AND AP SCORES (MEAN ± SD %) OF THE LINK PREDICTION TASK

Model	DBLP		AMiner		Yelp		Avg. Rank
	AUC	AP	AUC	AP	AUC	AP	
VGAE	79.59 ± 3.09	84.14 ± 1.93	82.80 ± 3.43	87.75 ± 2.02	44.64 ± 3.20	47.25 ± 3.08	10.83
GATv2	81.15 ± 0.59	81.60 ± 0.82	86.11 ± 0.37	87.10 ± 0.28	55.89 ± 2.59	57.00 ± 1.46	10.17
EvolveGCN	70.50 ± 5.27	73.00 ± 4.90	75.86 ± 3.96	78.72 ± 3.96	58.58 ± 4.98	61.20 ± 5.21	12.17
VGRNN	68.65 ± 0.54	75.24 ± 0.50	73.23 ± 1.12	80.08 ± 0.83	85.72 ± 1.77	**89.78 ± 1.74**	9.17
DySAT	78.92 ± 1.32	81.06 ± 1.46	85.84 ± 1.27	87.62 ± 1.06	64.24 ± 2.02	60.91 ± 2.46	9.33
HGWaveNet	69.52 ± 2.96	73.46 ± 2.90	76.79 ± 3.33	80.31 ± 3.01	55.56 ± 1.53	56.03 ± 1.35	12.67
Mp2vec	74.23 ± 1.40	72.95 ± 1.95	80.12 ± 1.62	79.43 ± 1.95	70.05 ± 0.91	67.40 ± 0.89	10.17
HGT	69.24 ± 1.89	72.90 ± 1.89	67.64 ± 6.65	71.95 ± 5.93	61.95 ± 3.27	59.57 ± 3.28	13.67
RGCN	82.28 ± 3.79	84.18 ± 3.95	88.16 ± 1.55	89.66 ± 1.47	68.63 ± 4.53	67.30 ± 7.33	7.00
HA	86.77 ± 0.46	88.37 ± 0.32	89.88 ± 0.69	91.49 ± 0.58	62.21 ± 2.33	60.58 ± 2.20	7.00
HTGNN	84.73 ± 2.07	84.24 ± 2.40	90.61 ± 1.66	91.13 ± 1.79	63.91 ± 3.29	62.85 ± 2.11	6.67
HGN2T$_{\text{RGCN-LSTM}}$	92.80 ± 0.36	92.51 ± 0.30	96.58 ± 0.51	96.55 ± 0.46	73.70 ± 1.91	75.60 ± 1.93	4.33
HGN2T$_{\text{RGCN-GRU}}$	94.19 ± 0.19	94.17 ± 0.20	97.62 ± 0.11	97.65 ± 0.09	79.97 ± 3.46	79.65 ± 4.80	3.33
HGN2T$_{\text{HA-LSTM}}$	95.52 ± 0.25	94.86 ± 0.28	98.78 ± 0.07	98.50 ± 0.10	81.08 ± 0.80	80.88 ± 0.76	2.33
HGN2T$_{\text{HA-GRU}}$	**96.30 ± 0.22**	**95.91 ± 0.30**	**98.98 ± 0.12**	**98.79 ± 0.16**	**87.01 ± 0.83**	87.69 ± 0.89	1.17

TABLE V
AUC AND AP SCORES (MEAN ± SD %) OF THE **NEW** LINK PREDICTION TASK

Model	DBLP		AMiner		Yelp		Avg. Rank
	AUC	AP	AUC	AP	AUC	AP	
VGAE	54.22 ± 3.19	60.76 ± 1.88	61.83 ± 5.24	70.20 ± 3.57	43.81 ± 6.48	47.12 ± 4.61	11.67
GATv2	63.77 ± 0.59	61.39 ± 0.56	69.96 ± 0.44	69.98 ± 0.79	51.02 ± 2.94	52.23 ± 1.84	8.67
EvolveGCN	55.43 ± 1.83	55.86 ± 2.24	62.85 ± 3.00	64.22 ± 2.73	57.80 ± 5.16	62.15 ± 5.31	9.83
VGRNN	48.35 ± 0.69	52.38 ± 0.74	47.90 ± 1.63	54.52 ± 1.02	39.32 ± 0.49	52.06 ± 1.18	14.17
DySAT	57.87 ± 0.63	59.21 ± 0.93	63.52 ± 2.41	66.92 ± 1.58	49.92 ± 1.14	51.62 ± 1.59	11.17
HGWaveNet	50.90 ± 3.42	51.01 ± 2.59	58.48 ± 4.39	59.12 ± 3.34	37.90 ± 1.09	43.15 ± 0.61	14.50
Mp2vec	58.29 ± 0.50	57.70 ± 0.61	64.90 ± 2.05	64.80 ± 1.52	53.86 ± 0.62	55.81 ± 0.91	9.17
HGT	60.96 ± 2.69	62.18 ± 2.75	61.54 ± 2.80	64.41 ± 1.97	54.20 ± 0.75	53.88 ± 1.69	9.17
RGCN	59.73 ± 6.98	61.08 ± 7.46	69.03 ± 3.38	72.15 ± 3.48	51.16 ± 4.59	55.20 ± 5.91	8.33
HA	70.02 ± 0.45	69.03 ± 0.51	74.28 ± 0.84	75.64 ± 0.85	51.27 ± 1.44	52.14 ± 1.52	7.00
HTGNN	70.74 ± 2.22	68.31 ± 2.22	76.24 ± 1.93	74.71 ± 1.35	53.18 ± 3.99	54.23 ± 3.11	6.33
HGN2T$_{\text{RGCN-LSTM}}$	83.83 ± 0.68	80.65 ± 0.57	91.26 ± 0.51	90.28 ± 0.56	65.15 ± 2.58	67.48 ± 2.66	4.00
HGN2T$_{\text{RGCN-GRU}}$	86.07 ± 0.43	83.41 ± 0.52	92.95 ± 0.39	92.09 ± 0.41	66.27 ± 4.27	69.13 ± 3.83	3.00
HGN2T$_{\text{HA-LSTM}}$	89.56 ± 0.51	85.45 ± 0.69	96.77 ± 0.17	95.54 ± 0.24	71.12 ± 0.83	72.27 ± 0.58	2.00
HGN2T$_{\text{HA-GRU}}$	**90.77 ± 0.41**	**87.37 ± 0.48**	**97.06 ± 0.35**	**95.98 ± 0.47**	**72.55 ± 0.76**	**75.30 ± 0.62**	1.00

*D. Dynamic (*New*) Link Prediction*

We conduct link prediction experiments using predictive node embedding. To evaluate the framework's prediction results for new edges, i.e., edges exist in G^{T+1} but not in G^{T}, we also perform the *new* link prediction task. We conduct experiments on the last 3 snapshots, and the dataset splitting and performance metrics are the same as in Section V-C. We randomly run 5 times and report the mean and standard deviation in Tables IV and V, respectively.

As can be seen from Tables IV and V, HGN2T$_{\text{HA-GRU}}$ achieves the best results in almost all evaluation metrics. First, the results for homogeneous GNNs, such as VGAE and GATv2, are significantly worse than for heterogeneous GNNs like R-GCN and HA, demonstrating that heterogeneity in HTGs is very important for the embedding results. Second, the obvious performance degradation between the link detection and link prediction results of the static methods, such as VGAE and R-GCN, shows that the temporal dependencies are crucial for the link prediction tasks. Compared with the static HGNN baseline models, the HTGNN model achieves competitive results in the new link prediction task, which also proves the importance of modeling temporal dependencies for HTG representation learning. Third, comparing link prediction and *new* link prediction tasks, the obvious performance difference shows that modeling temporal history dependencies is important for predicting whether future nodes will generate edges or not. Finally, the obvious performance difference between the R-GCN and HA models in the link detection task and the link prediction task verifies that the HGN2T can effectively improve the ability of the static HGNNs to model temporal dependencies.

In summary, through the proposed framework HGN2T, extended models have achieved better experimental results than the baseline models in the link prediction task, and this advantage is more obvious in *new* link prediction tasks. This demonstrates the effectiveness of our framework in extending the static HGNN to simultaneously model temporal dependencies information and heterogeneous topological information.

E. Parameter Sensitivity

In this section, we performed a parameter sensitivity analysis of the HGN2T$_{\text{HA-GRU}}$ model. The results of three datasets are shown in Figs. 3, 4, and 5.

- *The number of predicted snapshots:* We conduct experiments with different prediction lengths from 2 to 6 to evaluate the performance. As shown in Fig. 2, although the performance of most models gradually degrades with

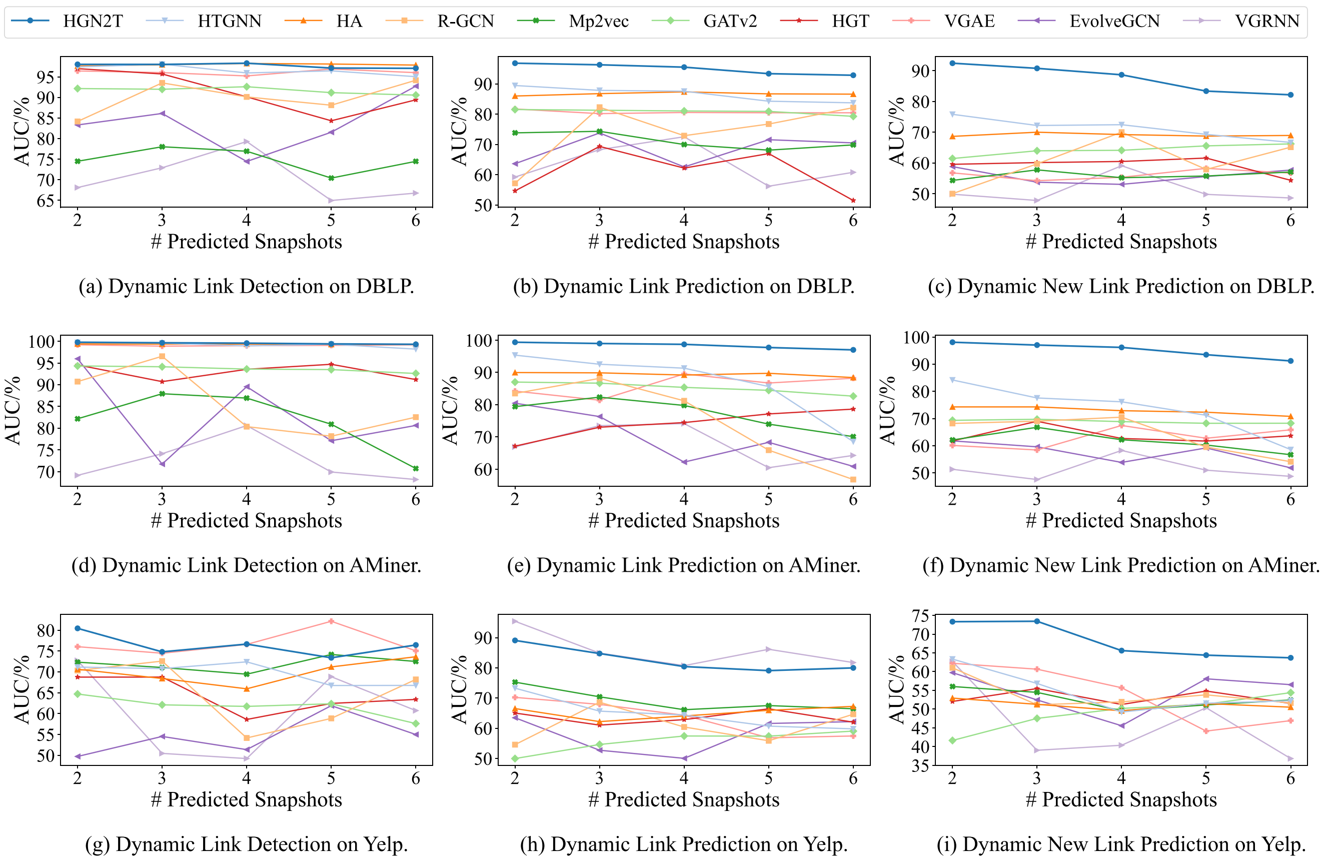


Fig. 2. Analysis of the number of predicted snapshots.

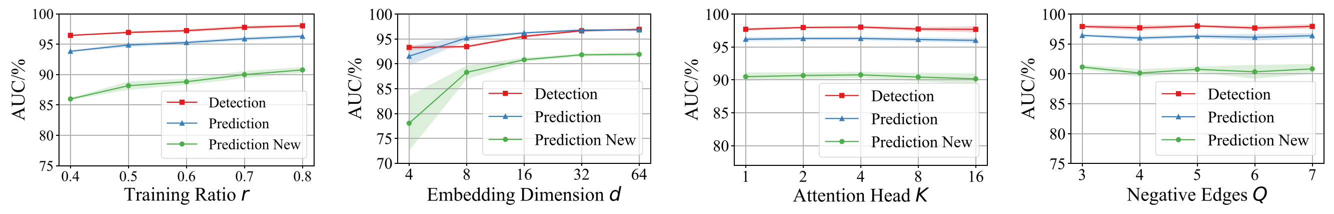


Fig. 3. Parameters sensitive analysis on DBLP dataset.

the increase of the number of predicted snapshots, HGN2T still achieves the best results. Compared with R-GCN and HA models, the HGN2T framework achieves more stable multi-snapshot prediction performance. This shows that the HGN2T framework has stable performance for both short-term and long-term prediction.

- *The ratio of the training data r:* We experiment by adjusting the number of training edges from 40% to 80% with a step size of 10% and report the three types of link prediction results. We can see that the AUC score gradually decreases with the reduction of the training edges. Overall, the performance of HGN2T is relatively stable across different training set sizes.
- *The number of the embedding dimension d:* We vary the number of embedding dimensions as $4, 8, 16, 32$ and 64. As shown in the results, the performance of the *new* link prediction task has a significant drop only when the node embedding dimension is extremely small ($d = 4$), which shows that the HGN2T framework can efficiently capture rich heterogeneity and temporal information.
- *The number of the network layers:* To verify the impact of the number of network layers of HGNN in HFI and TEU modules, we vary the number of layers from 1 to 5 and reported the results of HGN2T on the link detection and prediction task on the Yelp dataset. As shown in Fig. 6, we can see that HGN2T works best at layers 1-2, which shows that HGN2T can efficiently capture the temporal structural and semantic information.
- *The number of attention heads K:* We vary the number of attention heads as $1, 2, 4, 8$ and 16 to evaluate the

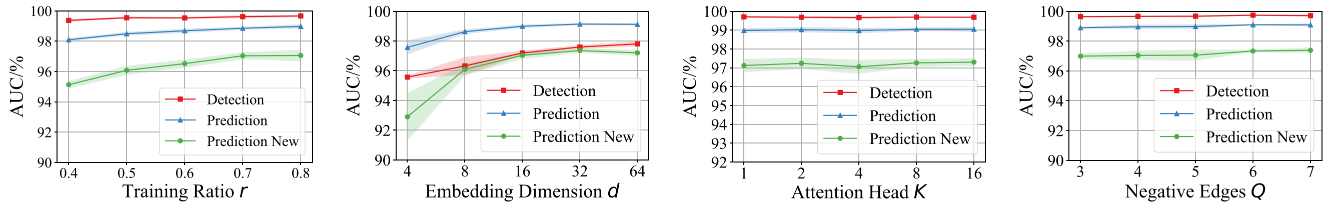


Fig. 4. Parameters sensitive analysis on AMiner dataset.

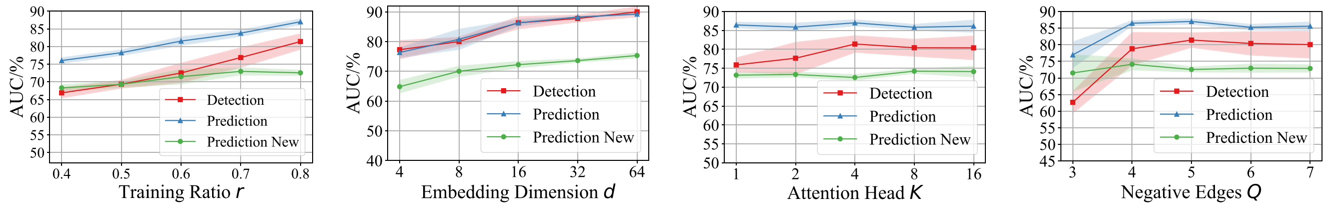


Fig. 5. Parameters sensitive analysis on yelp dataset.

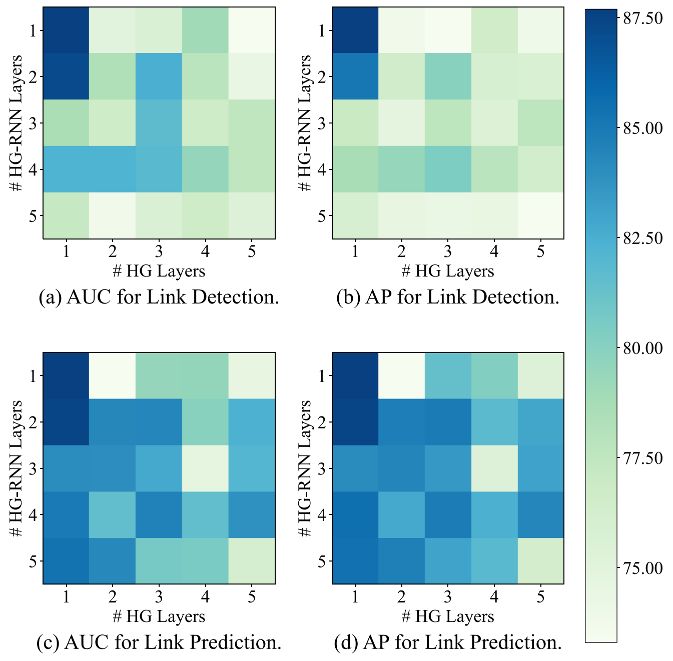


Fig. 6. Analysis of the number of network layers.

performance of the HGN2T$_{\text{HA-GRU}}$. It can be seen that the performance on all three tasks of HGN2T$_{\text{HA-GRU}}$ does not fluctuate significantly under different attention heads. This shows that the HGN2T framework can achieve stable performance with a small number of attention heads.

- *The number of negative samples edges Q:* We vary the negative sample multiples by $3, 4, 5, 6$ and 7. From the results, we can see that this parameter has no significant effect, which shows that the HGN2T can be well-optimized with a small number of negative samples.

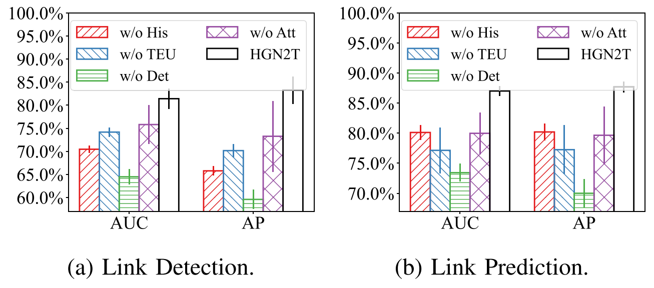


Fig. 7. Ablation study on yelp dataset.

F. Ablation Study

In the proposed framework HGN2T, we perform HTG representation learning through two main steps: HFI and TEU. To verify the effectiveness, we performed ablation experiments for each part of the HGN2T.

w/o His removes the combination of hidden state and only uses the projected features as input to HGNNs. *w/o TEU* replaces TEU module with a RNN model like GRU [54] to update the historical information. *w/o Det* removes the link detection loss to verify the effectiveness of Jointly Optimize. *w/o Att* removes the attention mechanisms denotes HGN2T$_{\text{RGCN-GRU}}$, which extends the R-GCN [23] to temporal and keeps the rest unchanged. **HGN2T** denotes the extended HGN2T$_{\text{HA-GRU}}$ model. Fig. 7 illustrates the results of the AUC and AP scores of the link detection and prediction experiments with ablated models on Yelp datasets.

As shown in Fig. 7, ablation of each part results in varying degrees of performance degradation. In the link detection task, HGN2T *w/o His* has the most obvious performance drop may be because the user's historical scoring record in the Yelp dataset has a great impact on the current moment. In link prediction

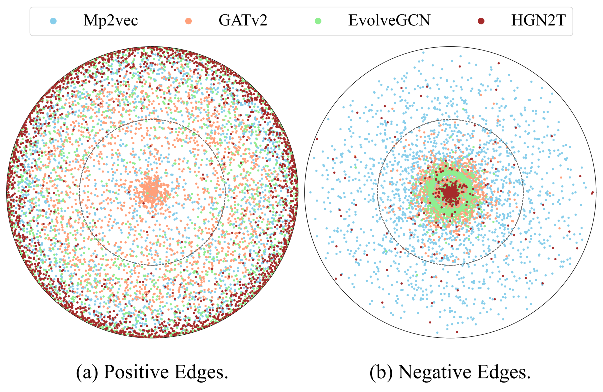


Fig. 8. Visualization of link detection on AMiner dataset.

tasks, the performance degradation of *w/o TEU* denotes that TEU can capture more structural and temporal information. *w/o Det* leads to the most significant performance degradation on both link detection and prediction tasks, which demonstrates the effectiveness of Joint Optimization for the framework. *w/o Att* yields performance degradation on all three tasks demonstrating the effectiveness of the attention mechanism. From the above analysis, it can be concluded that each part of HGN2T plays an indispensable role in HTG representation learning.

G. Visualization

To more intuitively observe the link detection and prediction performance of the proposed HGN2T framework, we conduct visualization experiments. We show the prediction results of positive and negative edges in Fig. 8, respectively. Each point represents the predicted probability of link detection for the tested edge. The greater the node distance from the center of the circle, the larger the predicted probability of the corresponding test edge, and vice versa. The circular dashed line in the figure indicates a predicted probability of 0.5. The colors of the points represent the prediction results of different methods. It can be clearly seen from Fig. 8 that, the points of positive edges of the HGN2T are generally scattered around the edge of the circle, while the points of negative edges are concentrated in the center of the circle. The visualization results show that the proposed HGN2T framework has obvious performance advantages for modeling HTGs.

In summary, the above empirical experimental results show that two existing commonly used HGNNs extended by the HGN2T framework have a stable and significant performance improvement in processing HTGs and outperform the baseline approach of spatio-temporal separation modeling, which exemplifies the effectiveness of the proposed HGN2T to model temporal and structural information in a coupled manner.

VI. Conclusion

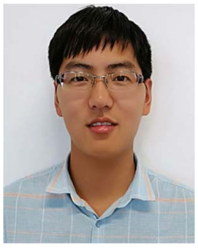
In this article, we propose a general framework that extends existing Heterogeneous Graph Neural Networks (HGNNs) for Heterogeneous Temporal Graph (HTG) representation learning, named HGN2T. Specifically, HGN2T includes two modules: historical feature incorporation and topology evolution update, which are used to capture historical information and update it respectively. We jointly optimize the link detection and link prediction tasks to capture full temporal dependencies from historical information. We conduct extensive link detection and prediction tasks on three real-world datasets, and our proposed framework outperforms the state-of-the-art baselines, which demonstrates its effectiveness.

References

[1] L. Wu, P. Cui, J. Pei, L. Zhao, and L. Song, "Graph neural networks," in *Graph Neural Networks: Foundations, Frontiers, and Applications*, Berlin, Germany: Springer, 2022, pp. 27–37.
[2] Z. Wu, S. Pan, F. Chen, G. Long, C. Zhang, and S. Y. Philip, "A comprehensive survey on graph neural networks," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 32, no. 1, pp. 4–24, Jan. 2021.
[3] P. Jiao, X. Guo, T. Pan, W. Zhang, Y. Pei, and L. Pan, "A survey on role-oriented network embedding," *IEEE Trans. Big Data*, vol. 8, no. 4, pp. 933–952, Aug. 2022.
[4] W. L. Hamilton, R. Ying, and J. Leskovec, "Representation learning on graphs: Methods and applications," *IEEE Data Eng. Bull.*, vol. 40, no. 3, pp. 52–74, Sep. 2017.
[5] Y. Zhang, S. Gao, J. Pei, and H. Huang, "Improving social network embedding via new second-order continuous graph neural networks," in *Proc. 28th ACM SIGKDD Conf. Knowl. Discov. Data Mining*, 2022, pp. 2515–2523.
[6] Y. Dong, N. Liu, B. Jalaian, and J. Li, "Edits: Modeling and mitigating data bias for graph neural networks," in *Proc. ACM Web Conf.*, 2022, pp. 1259–1269.
[7] D. Jin et al., "A survey of community detection approaches: From statistical modeling to deep learning," *IEEE Trans. Knowl. Data Eng.*, vol. 35, no. 2, pp. 1149–1170, Feb. 2023.
[8] D. Li, S. Zhang, and X. Ma, "Dynamic module detection in temporal attributed networks of cancers," *IEEE/ACM Trans. Comput. Biol. Bioinf.*, vol. 19, no. 4, pp. 2219–2230, Jul./Aug. 2022.
[9] W. Wu and X. Ma, "Network-based structural learning nonnegative matrix factorization algorithm for clustering of scRNA-seq data," *IEEE/ACM Trans. Comput. Biol. Bioinf.*, vol. 20, no. 1, pp. 566–575, Jan./Feb. 2023.
[10] X. Ma, W. Zhao, and W. Wu, "Layer-specific modules detection in cancer multi-layer networks," *IEEE/ACM Trans. Comput. Biol. Bioinf.*, vol. 20, no. 2, pp. 1170–1179, Mar./Apr. 2023.
[11] X. Gao et al., "Multi-view clustering with self-representation and structural constraint," *IEEE Trans. Big Data*, vol. 8, no. 4, pp. 882–893, Aug. 2022.
[12] S. Wu, F. Sun, W. Zhang, X. Xie, and B. Cui, "Graph neural networks in recommender systems: A survey," *ACM Comput. Surv.*, vol. 55, no. 5, pp. 1–37, 2022.
[13] 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.*, 2022, pp. 3217–3228.
[14] Y. Sun and J. Han, "Mining heterogeneous information networks: Principles and methodologies," *Synth. Lectures Data Mining Knowl. Discov.*, vol. 3, no. 2, pp. 1–159, 2012.
[15] C. Shi, Y. Li, J. Zhang, Y. Sun, and S. Y. Philip, "A survey of heterogeneous information network analysis," *IEEE Trans. Knowl. Data Eng.*, vol. 29, no. 1, pp. 17–37, Jan. 2017.
[16] X. Wang, D. Bo, C. Shi, S. Fan, Y. Ye, and S. Y. Philip, "A survey on heterogeneous graph embedding: Methods, techniques, applications and sources," *IEEE Trans. Big Data*, vol. 9, no. 2, pp. 415–436, Apr. 2023.
[17] Y. Dong, N. V. Chawla, and A. Swami, "metapath2vec: Scalable representation learning for heterogeneous networks," in *Proc. 23rd ACM SIGKDD Conf. Knowl. Discov. Data Mining*, 2017, pp. 135–144.
[18] X. Wang et al., "Heterogeneous graph attention network," in *Proc. World Wide Web Conf.*, 2019, pp. 2022–2032.
[19] H. Huang, R. Shi, W. Zhou, X. Wang, H. Jin, and X. Fu, "Temporal heterogeneous information network embedding," in *Proc. Int. Joint Conf. Artif. Intell.*, 2021, pp. 1470–1476.
[20] C. Yang, Y. Xiao, Y. Zhang, Y. Sun, and J. Han, "Heterogeneous network representation learning: A unified framework with survey and benchmark," *IEEE Trans. Knowl. Data Eng.*, vol. 34, no. 10, pp. 4854–4873, Oct. 2022.
[21] Z. Hu, Y. Dong, K. Wang, and Y. Sun, "Heterogeneous graph transformer," in *Proc. Web Conf.*, 2020, pp. 2704–2710.

[22] X. Wang, N. Liu, H. Han, and C. Shi, "Self-supervised heterogeneous graph neural network with co-contrastive learning," in *Proc. 27th ACM SIGKDD Conf. Knowl. Discov. Data Mining*, 2021, pp. 1726–1736.
[23] M. Schlichtkrull, T. N. Kipf, P. Bloem, R. v. d. Berg, I. Titov, and M. Welling, "Modeling relational data with graph convolutional networks," in *Proc. Eur. Semantic Web Conf.*, Springer, 2018, pp. 593–607.
[24] S. Fan et al., "Metapath-guided heterogeneous graph neural network for intent recommendation," in *Proc. 25th ACM SIGKDD Conf. Knowl. Discov. Data Mining*, 2019, pp. 2478–2486.
[25] J. Peng et al., "An end-to-end heterogeneous graph representation learning-based framework for drug–target interaction prediction," *Brief. Bioinf.*, vol. 22, no. 5, 2021, Art. no. bbaa430.
[26] Y. Pang et al., "Heterogeneous global graph neural networks for personalized session-based recommendation," in *Proc. 15th ACM Int. Conf. Web Search Data Mining*, 2022, pp. 775–783.
[27] Y. Fan, M. Ju, C. Zhang, and Y. Ye, "Heterogeneous temporal graph neural network," in *Proc. SIAM Int. Conf. Data Mining*, SIAM, 2022, pp. 657–665.
[28] P. Jiao et al., "Generative evolutionary anomaly detection in dynamic networks," *IEEE Trans. Knowl. Data Eng.*, vol. 35, no. 12, pp. 12234–12248, Dec. 2023.
[29] Q. Wen, Z. Ouyang, J. Zhang, Y. Qian, Y. Ye, and C. Zhang, "Disentangled dynamic heterogeneous graph learning for opioid overdose prediction," in *Proc. 28th ACM SIGKDD Conf. Knowl. Discov. Data Mining*, 2022, pp. 2009–2019.
[30] C. Shi, X. Wang, and P. S. Yu, "Dynamic heterogeneous graph representation," in *Heterogeneous Graph Representation Learning and Applications*, Berlin, Germany: Springer, 2022, pp. 107–143.
[31] P. Jiao, T. Li, H. Wu, C.-D. Wang, D. He, and W. Wang, "HB-DSBM: Modeling the dynamic complex networks from community level to node level," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 34, no. 11, pp. 8310–8323, Nov. 2023.
[32] Y. Ji, T. Jia, Y. Fang, and C. Shi, "Dynamic heterogeneous graph embedding via heterogeneous hawkes process," in *Proc. Joint Eur. Conf. Mach. Learn. Knowl. Discov. Databases*, Springer, 2021, pp. 388–403.
[33] H. Xue, L. Yang, W. Jiang, Y. Wei, Y. Hu, and Y. Lin, "Modeling dynamic heterogeneous network for link prediction using hierarchical attention with temporal RNN," in *Proc. Joint Eur. Conf. Mach. Learn. Knowl. Discov. Databases*, Springer, 2020, pp. 282–298.
[34] F. Wu, Y. Long, C. Zhang, and B. Li, "LINKTELLER: Recovering private edges from graph neural networks via influence analysis," in *Proc. Symp. Secur. Privacy*, 2022, pp. 2005–2024.
[35] J. M. Thomas, A. Moallemy-Oureh, S. Beddar-Wiesing, and C. Holzhüter, "Graph neural networks designed for different graph types: A survey," 2022, *arXiv:2204.03080*.
[36] C. Zhang, D. Song, C. Huang, A. Swami, and N. V. Chawla, "Heterogeneous graph neural network," in *Proc. 25th ACM SIGKDD Conf. Knowl. Discov. Data Mining*, 2019, pp. 793–803.
[37] V. W. Zheng et al., "Heterogeneous embedding propagation for large-scale e-commerce user alignment," in *Proc. IEEE Int. Conf. Data Mining*, 2018, pp. 1434–1439.
[38] Y. Liang, F. Meng, Y. Zhang, Y. Chen, J. Xu, and J. Zhou, "Emotional conversation generation with heterogeneous graph neural network," *Artif. Intell.*, vol. 308, 2022, Art. no. 103714.
[39] X. Fu, J. Zhang, Z. Meng, and I. King, "MAGNN: Metapath aggregated graph neural network for heterogeneous graph embedding," in *Proc. Web Conf.*, 2020, pp. 2331–2341.
[40] G. Xue, M. Zhong, J. Li, J. Chen, C. Zhai, and R. Kong, "Dynamic network embedding survey," *Neurocomputing*, vol. 472, pp. 212–223, 2022.
[41] E. Hajiramezanali, A. Hasanzadeh, K. Narayanan, N. Duffield, M. Zhou, and X. Qian, "Variational graph recurrent neural networks," in *Proc. 33rd Int. Conf. Neural Inf. Process. Syst.*, 2019, Art. no. 960.
[42] 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 ACM Int. Conf. Web Search Data Mining*, New York, NY, USA, 2020, pp. 519–527.
[43] A. Pareja et al., "EvolveGCN: Evolving graph convolutional networks for dynamic graphs," in *Proc. 34th AAAI Conf. Artif. Intell.*, 2020, pp. 5363–5370.
[44] J. You, T. Du, and J. Leskovec, "ROLAND: Graph learning framework for dynamic graphs," in *Proc. 28th ACM SIGKDD Conf. Knowl. Discov. Data Mining*, New York, NY, USA, 2022, pp. 2358–2366.
[45] Y. Zhu et al., "WinGNN: Dynamic graph neural networks with random gradient aggregation window," in *Proc. 29th ACM SIGKDD Conf. Knowl. Discov. Data Mining*, New York, NY, USA, 2023, pp. 3650–3662.
[46] Q. Bai, C. Nie, H. Zhang, D. Zhao, and X. Yuan, "HGWaveNet: A hyperbolic graph neural network for temporal link prediction," in *Proc. ACM Web Conf.*, New York, NY, USA, 2023, pp. 523–532.
[47] Q. Li, Y. Shang, X. Qiao, and W. Dai, "Heterogeneous dynamic graph attention network," in *Proc. IEEE Int. Conf. Knowl. Graph*, 2020, pp. 404–411.
[48] J. Gilmer, S. S. Schoenholz, P. F. Riley, O. Vinyals, and G. E. Dahl, "Neural message passing for quantum chemistry," in *Proc. 34th Int. Conf. Mach. Learn.*, PMLR, 2017, pp. 1263–1272.
[49] W. Hamilton, Z. Ying, and J. Leskovec, "Inductive representation learning on large graphs," in *Proc. 31st Int. Conf. Neural Inf. Process. Syst.*, 2017, pp. 1025–1035.
[50] T. N. Kipf and M. Welling, "Variational graph auto-encoders," in *NIPS Workshop Bayesian Deep Learn.*, 2016, pp. 1–3,
[51] S. Brody, U. Alon, and E. Yahav, "How attentive are graph attention networks?," in *Proc. Int. Conf. Learn. Representations*, 2021, pp. 1–14.
[52] X. Glorot and Y. Bengio, "Understanding the difficulty of training deep feedforward neural networks," in *Proc. 13th Int. Conf. Artif. Intell. Statist.*, 2010, pp. 249–256.
[53] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *Proc. Int. Conf. Learn. Representations*, 2015, pp. 1–11.
[54] F. A. Gers, J. Schmidhuber, and F. Cummins, "Learning to forget: Continual prediction with LSTM," *Neural Comput.*, vol. 12, no. 10, pp. 2451–2471, 2000.

Huan Liu received the bachelor's degree in computer science and technology from Shihezi University, in 2021. He is currently working toward the master's degree in computer science and technology with the School of Computer, Hangzhou Dianzi University. His current research interests include complex network analysis and network embedding.



Pengfei Jiao (Member, IEEE) received the PhD degree in computer science from Tianjin University, Tianjin, China, in 2018. From 2018 to 2021, he was a lecturer with the Center of Biosafety Research and Strategy of Tianjin University. He is currently a professor with the School of Cyberspace, Hangzhou Dianzi University, Hangzhou, China. His current research interests include complex network analysis and its applications.

Xuan Guo is currently working toward the doctoral degree with the College of Intelligence and Computing, Tianjin University, P. R. China. His current research interests include complex network analysis, role discovery, network representation learning and percolation model.

Huaming Wu (Senior Member, IEEE) received the BE and MS degrees from the Harbin Institute of Technology, China, in 2009 and 2011, respectively, both in electrical engineering, and the PhD degree with the highest honor in computer science from Freie Universität Berlin, Germany, in 2015. He is currently an Associate Professor with the Center for Applied Mathematics, Tianjin University, China. His research interests include wireless networks, mobile edge computing, Internet of Things and complex networks.

Mengzhou Gao received the BE degree in mechanical design, manufacturing and automation from the Harbin Institute of Technology, China, in 2012, and the PhD degree in control science and engineering from Zhejiang University, China, in 2017. Since 2017, she has been an Assistant Professor with the School of Cyberspace, Hangzhou Dianzi University, China. Her current research interests include cyber-physical systems security, secure control and complex networks.

Jilin Zhang (Member, IEEE) received the PhD degree in computer application technology from the University of Science Technology Beijing, Beijing, China, in 2009. He is currently a professor with the Key Laboratory of Complex Systems Modeling and Simulation, Ministry of Education, School of Computer Science and Technology, Hangzhou Dianzi University, Hangzhou, China. His research interests include high performance computing, Big Data technologies, and cloud computing.