Relation Time-aware Heterogeneous Dynamic Graph Neural Networks

Yili Wang[a,1], **Jiamin Chen**[a,1], **Qiutong Li**[a], **Changlong He**[a] **and Jianliang Gao**[a,*]

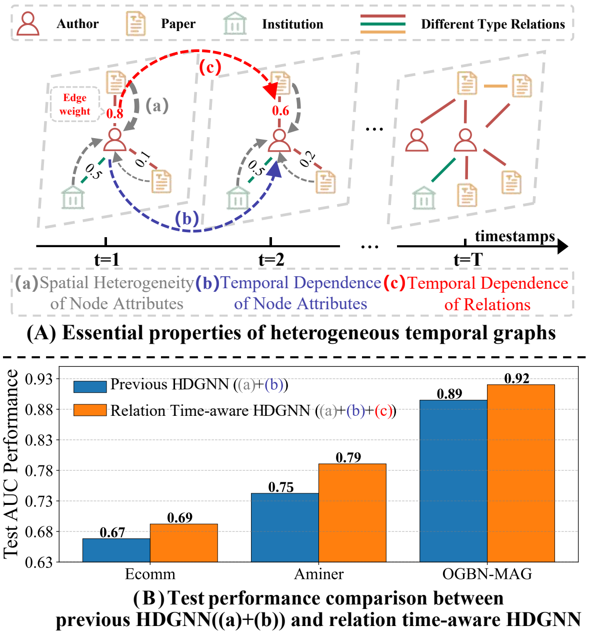
[a]Central South University

Abstract. Heterogeneous dynamic graph neural networks (HDGNNs) are effective methods for processing heterogeneous temporal graphs (HTGs), which serve as ubiquitous data structures in real-world scenarios. The previous HDGNN paradigm obtains representations of future target nodes by mining the spatial heterogeneity and temporal dependence of node attributes, ignoring the learning of relation temporal dependence. However, through experience, we find that the learning of relation temporal dependence, which describes the evolving trends in the importance of neighbors under a certain relation, is beneficial for representation learning of HTGs. To bridge this gap, we propose a novel end-to-end heterogeneous temporal graph learning paradigm called **Re**lation **T**ime-**a**ware Heterogeneous Dynamic **G**raph Neural Networks (ReTag). Compared to previous HDGNNs, ReTag extracts the temporal dependence of relations from historical relation information and the evolving node attributes to drive subsequent spatio-temporal representation learning. As far as we know, ReTag is the first attempt to perform learning of the temporal dependence of relation, which can generate a more effective representation for different downstream tasks of HTGs. The experimental results of different downstream tasks of HTGs based on multiple benchmark datasets show that ReTag can obtain obvious performance advantages compared with the sota baseline method.

1 Introduction

Heterogeneous temporal graphs (HTGs) are ubiquitous data structures that exists widely in the real world, such as e-commerce networks [21, 18, 7], academic networks [30, 14], and epidemic networks [5]. These HTGs consist of multiple evolving snapshots, with each snapshot representing a heterogeneous graph characterized by diverse node types and relations among connected nodes. Compared to static heterogeneous graphs, modeling with HTGs necessitates not only addressing spatial heterogeneity in node attributes, but also capturing intricate temporal dependencies across evolving snapshots.

Recently, heterogeneous dynamic graph neural networks (HDGNNs) have emerged as a powerful method for modeling HTGs and achieved remarkable success [13, 33, 8, 39]. To refine HTG modeling and further enhance performance, we conducted an in-depth analysis of the essential properties of heterogeneous temporal graphs, as shown in Figure 1 (A). Taking the academic graph OGBN-MAG [8] as an example, HTGs exhibit three essential properties: (a) spatial heterogeneity of node attributes, denoting the diverse feature distributions and semantic information across different node types, presenting challenges when aggregating neighbors; (b) temporal dependence of node attributes, denoting the regularity of node attribute evolves over time; and (c) temporal dependence of relations, denoting the regularity of relations evolves over time. The previous HDGNN paradigm fully exploits the (a) spatial heterogeneity and (b) temporal dependence of node attributes of HTGs through tailored spatio-temporal aggregation methods. Specifically, these methods [8, 33, 39, 13] typically employ attention mechanisms to address spatial heterogeneity and use sequential models such as recurrent neural network (RNN) or Transformer [26] to learn the temporal dependencies of node attributes. However, through experience, we find that the learning of (c) temporal dependence of relations, which describes the evolving trends in the importance of



(A) Essential properties of heterogeneous temporal graphs

(B) Test performance comparison between previous HDGNN((a)+(b)) and relation time-aware HDGNN

Figure 1. (A) shows three important properties of heterogeneous temporal graphs: (a) spatial heterogeneity of node attribute, (b) temporal dependence of node attribute, and (c) temporal dependence of relations. The numbers on relations represent edge weights, reflecting the importance level of neighbors. Edge weight from past snapshots inspires subsequent snapshots through temporal dependence of relations. (B) shows the test AUC performance comparison between the previous HDGNN and the relation time-aware HDGNN paradigm on multiple datasets Ecomm, Aminer, and OGBN-MAG. This results prove the effectiveness of relation time-aware learning in enhancing the performance.

[*] Corresponding Author Email: Gaojianliang@csu.edu.cn

[1] Equal contribution.

neighbors under a certain relation, is beneficial for representation learning of HTGs. For example, important neighbors identified in the preceding snapshot of HTGs typically maintain their importance in subsequent snapshots if their attributes remain relatively stable and vice versa. Obviously, previous works have overlooked the temporal dependence of relations since they treat each snapshot separately and use separate weights for learning (detailed in section 2.2). In this manner, historical aggregation experiences are unable to inspire subsequent aggregation processes due to this separate learning process, resulting in the loss of temporal information and sub-optimal performance. To visualize our findings, we incorporate the learning process of temporal dependence of relations (referred as relation time-aware learning in this work) into the typical method DyHATR [33], which is based on the HDGNN paradigm. Subsequently, we conduct performance comparison with the original DyHATR. As shown in Figure 1 (B), the performance comparison results show that since temporal dependence of relations are captured, the relation time-aware HDGNN paradigm ((a)+(b)+(c)) can generate better effective node representation to improve performance compared to previous HDGNN paradigm ((a)+(b)).

To this end, we propose a novel end-to-end heterogeneous temporal graph learning paradigm called **Re**lation **T**ime-**a**ware Heterogeneous Dynamic **G**raph Neural Network (ReTag) driven by relation time-aware learning. Specifically, ReTag based on the three essential properties of heterogeneous temporal graphs, containing three clear learning processes, namely *relation time-aware learning*, *spatial heterogeneity learning*, and *temporal dependence learning*. First, ReTag uses the gate recurrent unit (GRU) to learn the temporal dependence of relations through past edge weight and evolving node attributes to generate the time-aware relation adjacency matrices for subsequent spatio-temporal learning. Subsequently, based on the time-aware relation adjacency matrix, ReTag utilizes efficient graph convolution and attention mechanisms to extract heterogeneous spatial information across different snapshots. Finally, ReTag employs GRU to learn the temporal dependence of spatial representation to acquire the spatio-temporal representation, and thus predicting node representations for downstream tasks. We conducted extensive experiments on various downstream tasks of HTGs. Extensive experiments on various downstream tasks of HTGs compared to the latest and sota baseline method show that our method has significant performance advantages. The contributions of this work are summarized below.

- We rethink the property of HTGs and find the limitations of the previous HDGNN paradigm. In view of this limitation, we are the first to propose learning the temporal dependence of relations within HTGs as far as we know.
- We propose a novel end-to-end heterogeneous temporal graph learning paradigm, namely ReTag, which can generate more effective target node representation for different downstream HTG tasks driven by relation time-aware learning.
- In different downstream HTG task tests based on multiple benchmark datasets, ReTag can obtain significant performance advantages compared with previous optimal HDGNN methods, which fully demonstrates the effectiveness of our proposed novel paradigm.

2 Related Work

2.1 *Heterogeneous Graph Neural Networks.*

Recently, various heterogeneous graph neural networks [25, 29, 38, 6] have been proposed with successful applications [1, 2, 11]. In order to utilize task-relevant information in static heterogeneous graphs for downstream tasks, these models have made attempts in various aspects. RGCN [25] employs one-hop neighbors as the message receptive field and learns multiple convolution kernels to ensure unambiguous semantic information. HAN [29], MAGNN [10], and NIRec [19] capture various semantic information through manually designed meta-graphs [15] and a semantic-level attention mechanism. HGT [13] learns the semantic information from each relation based on the Transformer architecture [26]. GTN [37], MHGCN [36], DiffMG [6], and NDS [32] attempt to employ automated design methods to discover the optimal message-passing paths that can generate more comprehensive representation. However, models designed for static heterogeneous graphs face challenges in capturing complex spatio-temporal information and evolving semantic information in HTGs.

2.2 *Dynamic Graph Neural Networks.*

Dynamic graphs structures [24, 40] have been extensively explored in the literature, with numerous successful applications [12, 9, 28, 16]. To generalize the success of dynamic homogeneous GNNs [4, 23, 35, 3], there's been considerable research on dynamic heterogeneous GNNs [22, 31]. Recently, attention-based HDGNNs have demonstrated promising results on HTGs [34, 33, 17, 8, 39]. These efforts attempt to extract spatial information using attention mechanisms with non-shared parameters on each snapshot, overlooking the underlying connections between individual snapshots. As a result, they may overlook crucial temporal information. Specially, HGT+ [13] and DHGAS [39], as Transformer [26] variants, compress dynamic graph snapshots into a single static graph attempting to address this issue. However, their essence still lies in treating each snapshot separately since they use independent trainable weight for the same neighbors at different time, rather than learning the trend of importance changes over time. Additionally, they encounter out-of-memory (OOM) issues when dealing with large-scale graphs. In comparison, Retag can capture temporal dependence of relations in HTGs, utilizing past aggregation experiences to inspire the current aggregation process. This approach leads to a more effective and efficient representation learning for HTGs.

3 Preliminary and Notations

3.1 *Heterogeneous Temporal Graph (HTG)*

The heterogeneous temporal graph consists of multiple snapshots that evolve over time. Each snapshot is a directed heterogeneous graph $G = (V, E)$, in which V is the node set associated with a node type mapping function $f_n : V \to \mathcal{T}_n$ and E represents the edge set associated with a relation[2] type mapping function $f_r : E \to \mathcal{T}_r$. The $\mathcal{T}_n$ and $\mathcal{T}_r$ represent type set of nodes and relations, where $|\mathcal{T}_n| + |\mathcal{T}_r| \geq 2$. The heterogeneous temporal graph is defined as a set of directed heterogeneous graphs $\mathcal{G} = \left(\left\{G^t\right\}_{t=1}^{T}, \mathcal{V}, \mathcal{E}\right)$, where T is the number of timestamps, $G^t = \left(V^t, E^t\right)$ is the snapshot of directed heterogeneous graph at timestamp t, $\mathcal{V} = \bigcup_{t=1}^{T} V^t$, $\mathcal{E} = \bigcup_{t=1}^{T} E^t$, denote the node set and edge set of $\mathcal{G}$. Heterogeneous temporal graphs are generalized graph data representations ubiquitous in the real world. Homogeneous graph and static graph are special cases of HTGs when $|\mathcal{T}_n| + |\mathcal{T}_r| = 2$ and $T = 1$ respectively.

[2] In this paper, we use 'edge' to refer to a specific edge and 'relation' to represent a collection of edges of a certain type.

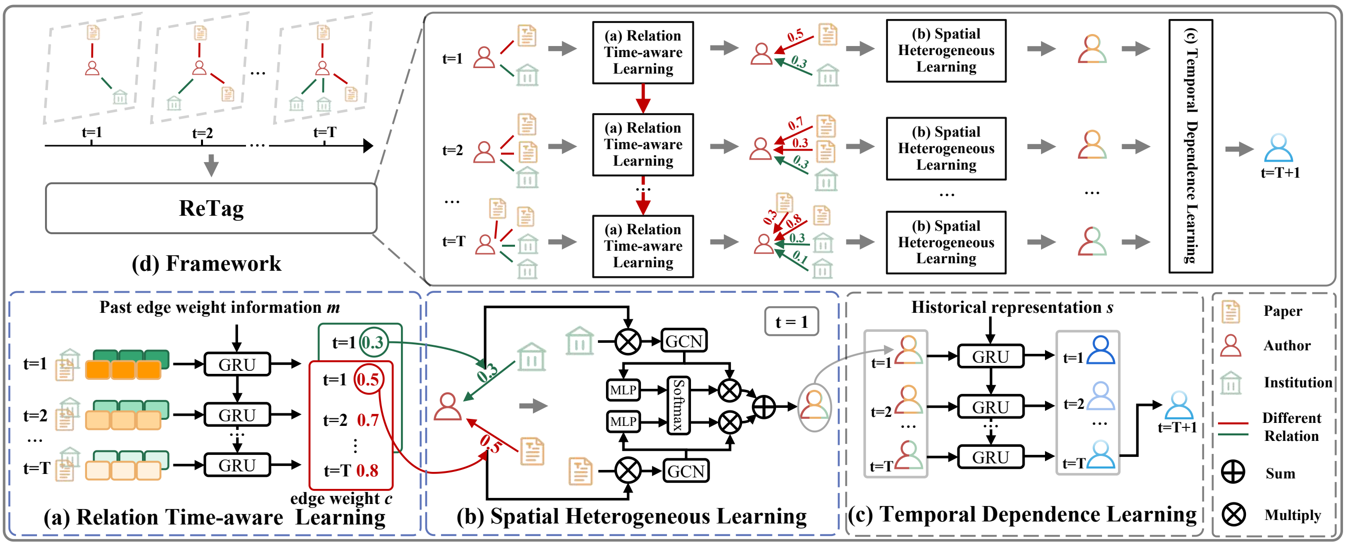

t=1
t=2
...
t=T
ReTag
(d) Framework
(a) Relation Time-aware Learning
(b) Spatial Heterogeneous Learning
(c) Temporal Dependence Learning
t=T+1
0.5
0.3
0.7
0.8
0.1
Past edge weight information m
GRU
edge weight c
(a) Relation Time-aware Learning
t = 1
GCN
MLP
Softmax
(b) Spatial Heterogeneous Learning
Historical representation s
(c) Temporal Dependence Learning
Paper
Author
Institution
Different Relation
Sum
Multiply

Figure 2. The overall framework of Relation Time-aware Heterogeneous Dynamic Graph Neural Network. (a) It is relation time-aware learning, which can effectively calculate edge weight from the attribute of the current node and past edge weight information based on gate recurrent unit (GRU). (b) It uses an efficient graph convolution network (GCN) and an attention mechanism to learn the effective spatial heterogeneity of node attributes with the assistance of edge weights. (c) Effective target node representations can be generated by learning the temporal dependence of node representations. We take the last state of the output of GRU model as the final spatio-temporal embedding for downstream task. (d) shows the overall framework and workflow of Retag.

3.2 *Heterogeneous Dynamic Graph Neural Networks*

Graph neural networks (GNNs) [20] employ the message passing mechanism to learn local topological structure features of graphs, thereby obtaining node representations. This mechanism consists of two processes: message aggregation and representation updating. The representation $\mathbf{h}_v^{l+1}$ of node v at GNN layer $l+1$ can be updated as follows:

$$\mathbf{h}_v^{l+1} = \mathbf{Update}(\mathbf{h}_v^{l}, \mathbf{Agg}(\{\mathbf{h}_u^{l} : u \in \mathcal{N}(v)\})), \qquad (1)$$

where $\mathcal{N}(v)$ denotes the neighbor nodes of v, $\mathbf{Agg}(\cdot)$ is the message aggregation operation to aggregate the neighbor node representations of v. $\mathbf{Update}(\cdot)$ stands for the representation update function to obtain node representation at the next GNN layer. Based on the formal definition of GNN, the previous heterogeneous dynamic graph neural network (HDGNN) paradigm is formalized as follows:

$$\begin{aligned} \mathbf{h}_v^{t,l+1} &= \mathbf{Update}(\mathbf{h}_v^{t,l}, \mathbf{Agg}(\{\mathbf{h}_u^{t,l} : u \in \mathcal{N}_r(v), r \in \mathcal{T}_r\})) \\ \mathbf{h}_v^{T+1} &= \mathbf{Sequence}(\{\mathbf{h}_v^{t}\}_{t=1}^{T}), \end{aligned} \qquad (2)$$

where $\mathbf{h}_v^{t,l}$ is the representation of l^{th} layer of node v at timestamp t, $\mathbf{h}_v^{t}$ represents the updated representation of final layer. $\mathbf{Agg}(\cdot)$ is message aggregation operation, $\mathcal{N}_r(v)$ is the neighbor of node v with the relation type r. The $\mathbf{Sequence}(\{\cdot\}_{t=1}^{T})$ stands for the sequence-based method that can aggregate the representation $\mathbf{h}_v^{t}$ in the time window T to generate the representation $\mathbf{h}_v^{T+1}$ of target node v.

4 The Proposed Method

In this section, we will detail the three clear learning processes of the ReTag method: relation time-aware learning, spatial heterogeneity learning, and temporal dependence learning. The framework of ReTag is shown in Figure 2.

4.1 *Relation Time-aware Learning*

The temporal dependence of relations is an important property of HTG, which describes the evolving trends in the importance of neighbors under a certain relation. For example, important neighbors identified in the preceding snapshot typically maintain their importance in subsequent snapshots if their attributes remain relatively stable and vice versa. However, previous work typically aggregates neighbors of each snapshot separately, overlooking the evolving trends between them.

In view of this limitation, we design the Relation Time-aware Learning module to capture the temporal dependence of the relations, as shown in Figure 2(a). To achieve this goal, this module utilizes historical edge weight information to predict current edge weights by considering the current source node attributes of that edge. Then it outputs time-aware relation adjacency matrix, which generated by predicted edge weights and original adjacency matrix, to guide the subsequent aggregation processes. Concretely, given an edge between a source node v and target node u corresponding to relation r, Retag learns the edge weight information $\mathbf{m}_{v,r}^{t}$ at time t based on past edge weight information $\mathbf{m}_{v,r}^{t-1}$ and current node attribute x_v^t, and then generates the edge weight vector $c_{v,r}^{t}$ of source node v at time t corresponding to relation r through a softmax operation. Note that for a trade-off between efficiency and performance, only the attribute of the source node v, which is defined as the starting point for message passing, is involved during the calculation process. In this manner, it is sufficient to compute once for all edges that originate from a certain source node. This process is formalized as follows[3]:

$$\underbrace{\overbrace{\mathbf{m}_{v,r}^{t}}^{\text{edge weight information}}}_{\text{hidden state}} = \mathrm{GRU}(\underbrace{\overbrace{\mathbf{x}_v^{t}}^{\text{node attribute}}}_{\text{input}}, \underbrace{\overbrace{\mathbf{m}_{v,r}^{t-1}}^{\text{past edge weight information}}}_{\text{hidden state}}), \qquad (3)$$

[3] To simplify notations, we omit the layer superscript

where $\mathbf{m}_{v,r}^{t-1} \in \mathbb{R}^{1\times 2}$ represents the hidden state which contains historical edge weight information. $t \in [1, T]$, where T represents the number of snapshots. Furthermore, $\mathbf{m}_{v,r}^{0}$ is initialized as the zero vector. GRU denotes the gated recurrent units, which are utilized as the temporal dependence learner. To ensure effective model convergence, we constrain edge weight values to be between 0 and 1. Within this range, larger values indicate greater importance of neighbors, while "0" signifies that neighbors are completely unimportant and do not need to be aggregated. Therefore, we use a softmax operation to transform edge weight information $\mathbf{m}_{v,r}^{t-1}$ into edge weight vector $c_{v,r}^{t}$ as follow:

$$c_{v,r}^{t}[i] = \frac{\exp\left(\mathbf{m}_{v,r}^{t}[i]/\tau\right)}{\sum_{i'\in\{0,1\}} \exp\left(\mathbf{m}_{v,r}^{t}\left[i'\right]/\tau\right)}, \tag{4}$$

where $c_{v,r}^{t} \in \mathbb{R}^{1\times 2}$ denotes the edge vector, τ denotes the temperature parameter to control the “sharpness” of the vector $c_{v,r}^{t}$. When $\tau \rightarrow 0$, the vector $c_{v,r}^{t}$ becomes one-hot, which enables the model to make more decisive judgments about the importance of neighbors. Due to the end-to-end optimization design, the specific definition of which element in weight vector $c_{v,r}^{t}$ to serves as the edge weight does not affect the optimization result. Therefore, we choose the second element $c_{v,r}^{t}[1]$ as the edge weight. Then, the relation mask matrix $\mathbf{M}_{r}^{t}$ with the same dimensions as the adjacency matrix can be developed using the following formula:

$$\mathbf{M}_{r}^{t}[:, v] = c_{v,r}^{t}[1] \cdot \mathbf{v}, \tag{5}$$

where $\mathbf{M}_{r}^{t}[:, v]$ denotes the v^{th} column vector of $\mathbf{M}_{r}^{t}$, $\mathbf{v} \in \mathbb{R}^{N\times 1}$ denotes a N-dimensional vector filled with value 1. Then ReTag can output a time-aware relation adjacency matrix $\mathbb{A}_{r}^{t}$ as follows:

$$\mathbb{A}_{r}^{t} = \mathbf{M}_{r}^{t} \odot \mathbf{A}_{r}^{t} \tag{6}$$

where $\odot$ denotes an element-wise multiplication operator, $\mathbf{A}_{r}^{t}$ denotes original adjacency matrix corresponding to relation r at timestamp t. Next, the time-aware relation adjacency matrix $\mathbb{A}_{r}^{t}$ will replace the original adjacency matrix $\mathbf{A}_{r}^{t}$ in the subsequent spatial aggregation process. Compared to original adjacency matrices, time-aware relation adjacency matrices contain edge weights inspired by past aggregation experiences, which can guide subsequent aggregation operations to obtain better node representations.

4.2 *Spatial Heterogeneity Learning*

The objective of spatial heterogeneity learning is to update node representations at each snapshot through heterogeneous neighbor information. In a heterogeneous graph, each node type may have its own attribute space. To map node's distinct attribute vector into a same attribute space, we first project them into the common latent space using a type-specific projection. Mathematically, given a node v with node type $f_n(v)$ at timestamp t, we have the following:

$$\mathbf{h}_{v}^{t,0} = \mathbf{W}_{f_n(v)} \cdot \mathbf{x}_{v}^{t} + \mathbf{b}_{f_n(v)} \tag{7}$$

where $\mathbf{x}_{v}^{t}$ denotes raw attribute, $\mathbf{W}_{f_n(v)}$ is the trainable type-specific transformation matrix, $\mathbf{b}_{f_n(v)}$ is the trainable type-specific bias. Next, for higher efficiency considerations, we implement the GCN [20] in a non-parametric way as the aggregation function to aggregate neighbors information. Notably, the adjacency matrix $\mathbf{A}$ in the GCN formula will be replaced by the time-aware relation adjacency matrix $\mathbb{A}$, which contains edge weights inspired by past aggregation experiences. In this manner, neighbor attributes are multiplied by edge weights before being aggregated into the target node, thus reflecting differences in importance. Additionally, neighbors under the same type of relations exhibit differences at the level of importance, while those of different types also show distinctions. Therefore, apart from using $\mathbb{A}$ to distinguish differences at a finer granularity, we will also learn a coefficients α to fusion different type of neighbors information at a coarser granularity. As shown in Figure 2 (b), this process can be mathematically expressed as the following:

$$\mathbf{H}_{f_n(v)}^{t,l} = \sum_{r\in\mathcal{R}(v)} \overline{\alpha}_{r}^{t,l}\left(\mathbb{A}_{r}^{t,l}\mathbf{H}_{f_n(v)}^{t,l-1}\right), \tag{8}$$

$$\alpha_{r}^{t,l} = \frac{\exp\left(\sigma\left(\mathbb{A}_{r}^{t,l}\mathbf{H}_{f_n(v)}^{t,l-1}\mathbf{W}^{l} + \mathbf{b}^{l}\right)\right)}{\sum_{r'\in\mathcal{R}(v)} \exp\left(\sigma\left(\mathbb{A}_{r'}^{t,l}\mathbf{H}_{f_n(v)}^{t,l-1}\mathbf{W}^{l} + \mathbf{b}^{l}\right)\right)} \tag{9}$$

where $\alpha_{r}^{t,l} \in \mathbb{R}^{n\times 1}$ denotes attention coefficient, n denotes the number of node, $\overline{\alpha}_{r}^{t,l} \in \mathbb{R}^{1}$ denotes the average of $\alpha_{r}^{t,l}$, $\mathbf{H}_{f_n(v)}^{t,l}$ denotes intermediate representations of l^{th} layer of node type $f_n(v)$ at time t, $\mathbb{A}_{r}^{t,l}$ denotes learned time-aware relation adjacency matrix of l^{th} layer corresponding to relation r at time t, $\mathcal{R}(v)$ denotes the set of relations with $f_n(v)$ as their target node type, σ is the activation function (*i.e.*, tanh function). The trainable transformation matrix $\mathbf{W}^{l}$ and bias $\mathbf{b}^{l}$ at the l^{th} layer are shared across different timestamps and relations to improve efficiency. Note that here we use the perspective of node types $f_n(v)$ in the mathematical expression to align with the GCN formula. From an individual node perspective, given a node v, its heterogeneous spatial representation $\mathbf{h}_{v}^{t,l}$ corresponds to a certain row in $\mathbf{H}_{f_n(v)}^{t,l}$. After updating all types of nodes, we can obtain the heterogeneous spatial representation $\mathbf{h}_{v}^{t}$ for different snapshots.

4.3 *Temporal Dependence Learning*

Temporal dependence learning is designed to capture interactions among the temporal neighbors of the target node, enabling the prediction of representation at the next time step for downstream tasks. Temporal neighbors are defined as the same nodes that exist in other graph snapshots, including that node itself. This module takes the heterogeneous spatial representation of each snapshot as input and outputs a spatio-temporal embedding for target node. As shown in Figure 2 (c), given a node v, Retag utilizes gate recurrent unit to capture the temporal dependence of spatial representation, and predicts next time representation for downstream task. This process can be mathematically expressed as the following:

$$\overbrace{\underbrace{\mathbf{s}_{v}^{t}}_{\text{hidden state}}}^{\text{future representation}} = \mathrm{GRU}(\overbrace{\underbrace{\mathbf{h}_{v}^{t}}_{\text{input}}}^{\text{spatial representation}}, \overbrace{\underbrace{\mathbf{s}_{v}^{t-1}}_{\text{hidden state}}}^{\text{historical representation}}), \tag{10}$$

where $\mathbf{h}_{v}^{t}$ denotes spatial representation of node v at time t obtained by above module, $\mathbf{s}_{v}^{t}$ denotes hidden state generated by GRU. Furthermore, $\mathbf{s}_{v}^{0}$ is also initialized as the zero vector. Finally, we take the last state $\mathbf{s}_{v}^{T}$ of the output of GRU model as the final spatio-temporal embedding h_{v}^{T+1} for downstream task.

4.4 *Optimization*

We then pass the yielded representation h_{v}^{T+1} of node v to a two layer multilayer perceptron (MLP) to capture non-linear interactions between the representations.

$$\mathbf{h}_{v} = \mathrm{MLP}(\mathbf{h}_{v}^{T+1}), \tag{11}$$

Table 1. Statistics of datasets.

Dataset	Graph	Time Span	Node	Relation	Data Split
Aminer	# Graph: 16 Granularity: year	1990-2005	# Paper : 18,464 # Author : 23,035 # Venue : 22	# Paper-publish-Venue : 18,464 # Author-write-Paper : 52,545	Training: 14 Validation: 1 Testing: 1
OGBN-MAG	# Graph: 10 Granularity: year	2010-2019	# Author: 17,764 # Paper: 282,039 # Field: 34,601 # Institution: 2,276	# Author-Paper: 2,061,677 # Paper-Paper:2,377,564 # Paper-Field: 289,376 # Author-Institution: 40,307	Training: 8 Validation: 1 Testing: 1
Ecomm	# Graph: 10 Granularity: day	10/6/2019- 19/6/2019	# User : 1,476 # Item : 34,505	# User-click-Item : 57,917 # User-buy-Item : 5,529 # User-cart-Item : 15,066 # User-favorite-Item: 6,701	Training: 8 Validation: 1 Testing: 1
YELP	# Graph: 12 Granularity: month	01/2012- 12/2021	# User : 55,702 # Business : 12,524	# User-review-Business : 87,846 # User-tip-Business : 35,508	Training: 10 Validation: 1 Testing: 1
COVID-19	# Graph: 304 Granularity: day	05/01/2020- 02/28/2021	# State: 54 # County: 3223	# State-State: 269 # State-County: 3,141 # County-County: 22,176	Training: 244 Validation: 30 Testing: 30

Next, we introduce distinct loss functions for three tasks over HTGs, that is, node classification, recommendation, and node regression.

For node classification, the node representations will be projected by MLP to the hidden dimension corresponding to the number of classes. Then the MLP is trained to minimize the cross-entropy loss:

$$\mathcal{L} = -\sum_{v\in\mathcal{V}}\sum_{c=1}^{|\mathcal{T}_n|} y_v[c]log(\hat{y}_v[c]), \tag{12}$$

where $\mathcal{V}$ denotes the set of labeled nodes, $\mathcal{T}_n$ denotes the node type set, y_v is a one-hot vector indicating the label of node v, $\hat{y}_v = \mathrm{softmax}(h_v)$ is the predicted label for the corresponding node.

For link prediction, we use the following loss:

$$\mathcal{L} = -\sum_{(u,v)\in\Omega+} \log\sigma\left(\mathbf{h}_u^\top\mathbf{h}_v\right) - \sum_{(u',v')\in\Omega-} \log\sigma\left(-\mathbf{h}_{u'}^\top\mathbf{h}_{v'}\right), \tag{13}$$

where Ω^+ and Ω^- denote the set of observed positive pairs and the set of negative pairs respectively, and σ denotes the sigmoid function. $h_u, h_v, h_{u'}, h_{v'}$ are output representations.

For node regression, we use the MAE loss as follow:

$$\mathcal{L} = \frac{1}{|\mathcal{V}_L|}\sum_{v\in\mathcal{V}_L}|y_v - \hat{y}_v|, \tag{14}$$

where $\mathcal{V}_L$ is the set of target node, y_v is the ground-truth label (an integer) of node v and $\hat{y}_v = h_v$ is the regression value node v output by the MLP.

5 Experiments

In this section, we present the experimental evaluations of Retag. We evaluated the performance of our model on three tasks: node classification, link prediction, and node regression. In addition, we conducted ablation studies to analyze the individual components of our model. We also evaluate the efficiency and impact of hyperparameters of our model.

5.1 Dataset

We utilized three link prediction datasets: OGBN-MAG, Aminer, and Ecomm, one node classification dataset YELP, and one node regression dataset COVID-19. We follow the splits provided by previous works [8, 39]. The statistics of the datasets are summarized in Table 1 and their descriptions are as follows.

- **Aminer**[4]**:** Aminer [39] is an academic citation dataset for papers that were published during 1990-2006. The dataset has three types of nodes (paper, author and venue), and two types of relations (paper-*publish*-venue and author-*writer*-paper).
- **OGBN-MAG**[5]**:** The original OGBN-MAG dataset [8] is a static heterogeneous network composed of a subset of the Microsoft Academic Graph (MAG). Previous work selects authors that consecutively publish at least one paper every year. Then it collects these authors' affiliated institutions, published papers, and the papers' field of studies in each year to construct this HTG. Each snapshots is a heterogeneous graph that contains four types of nodes (paper, author, institutions, and fields of study), and four types of relations among them (author-*affiliated with*-institution, author-*writes*-paper, paper-*cites*-paper, and paper-*has a topic of* -field of study).
- **Ecomm**[6]**:** Ecomm [39] is a recommendation dataset, recording shopping behaviors of users. It has two types of nodes (user and item) and four types of relations (user-*click*-item, user-*buy*-item, user-*cart*-item and user-*favorite*-item).
- **Yelp**[7]**:** Yelp [39] is a business review dataset, containing user reviews and tips on business. Following, we consider interactions of three categories of business including "American (New) Food", "Fast Food" and "Sushi" from January 2012 to December 2012.
- **COVID-19**[8]**:** This dataset [8] contains both state and county level daily case reports (*e.g.*, confirmed cases, new cases, deaths, and recovered cases). We use the daily new COVID-19 cases as the time-series data for each state and county. We then build a HTG including 304 graph slices. Each graph slice is also a heterogeneous graph consisting of two types of nodes (state and county) and three types of relations between them, *i.e.*, one administrative affiliation relation (state-*includes*-county) and two geospatial relations (state-*near*-state, county-*near*-county).

[4] https://www.aminer.cn/collaboration
[5] https://github.com/snap-stanford/ogb
[6] https://tianchi.aliyun.com/competition/entrance/231719
[7] https://www.yelp.com/dataset
[8] https://coronavirus.1point3acres.com/en

Table 2. The AUC results of recommendation, Macro-F1 results of node classification($\% \pm \sigma$) and MAE result for node regression. We further list the average rank of each method on all datasets, denoted by Avg. Rank. The best and second best results are shown in **bold** and underline. OOM stands for out of memory.

Type	Method	**Link Prediction** (AUC%) ↑			**Node Classification** (Macro F1%) ↑	**Node Regression** (MAE) ↓	Avg. Rank↓
		Aminer	OGBN-MAG	Ecomm	YELP	COVID-19	
Homogeneous GNNs	GCN *(ICLR'2017)*	73.12 ± 0.46	78.16 ± 1.32	56.62 ± 0.74	37.21 ± 0.52	841 ± 98	8.6
	GAT *(ICLR'2018)*	81.56 ± 1.25	79.97 ± 1.98	56.61 ± 1.45	35.39 ± 1.20	814 ± 95	8.2
Heterogeneous GNNs	RGCN (ESWC'2018)	82.12 ± 0.12	81.25 ± 1.99	68.65 ± 0.48	37.86 ± 0.89	830 ± 85	5.8
	HGT *(WWW'2020)*	78.81 ± 1.29	84.38 ± 1.22	62.21 ± 2.14	34.28 ± 1.08	799 ± 82	7
	DiffMG *(KDD'2021)*	85.14 ± 0.32	87.98 ± 1.26	60.49 ± 0.89	39.32 ± 0.89	621 ± 58	4.8
Heterogeneous Dynamic GNNs	DyHATR (ECML'2020)	86.46 ± 1.28	89.49 ± 0.65	66.82 ± 0.63	38.89 ± 0.64	668 ± 56	3.6
	HGT+ *(WWW'2020)*	85.88 ± 0.38	OOM	61.46 ± 2.56	38.33 ± 0.60	OOM	7
	HTGNN *(SDM'2022)*	85.92 ± 0.93	91.21 ± 0.91	58.58 ± 0.63	36.65 ± 1.13	555 ± 34	5
	DHGAS *(AAAI'2023)*	86.97 ± 0.90	OOM	62.56 ± 0.58	40.57 ± 1.8	540 ± 54	3.8
	ReTag *(Ours)*	**89.88 ± 0.81**	**92.98 ± 1.23**	**70.73 ± 1.12**	**44.3 ± 0.92**	**527 ± 58**	**1.0**

5.2 Baseline

We compare our method with state-of-the-art baselines. Specifically, we compare Retag with the following competitive baselines.

- **Homogeneous GNNs:** We take GCN [20] and GAT [27]. On heterogeneous temporal graph, these models ignore the heterogeneity and evolution of the graph.
- **Static Heterogeneous GNNs:** We take RGCN [25], HGT [13], and DiffMG[6]. These models also ignore the evolution of HTGs.
- **Dynamic Heterogeneous GNNs:** We take DyHATR [33], HGT+ [13], HTGNN [8] and DHGAS [39].

Notably, among these methods, DiffMG and DHGAS are automated models, while the rest are manually designed models.

5.3 Implementation Details

We use PyTorch Geometric implementations for GCN, GAT, RGCN, HGT and HGT+. Others are implemented using the source codes released by the authors. Experiments are conducted on Ubuntu 20.04.4 LTS with NVIDIA RTX 3090 GPU and 128GB RAM. For all baselines and datasets, we choose the number of message-passing layers in $\{1, 2, 3\}$ and the number of attention heads in $\{1, 2, 4, 8\}$. The hidden representation dimensionality set as d = 64 except d = 8 for COVID-19. Other hyperparameters for baselines are kept the same as in the original paper. The max number of epochs is set as 500, and we set the early stopping round on the validation set as 25 to alleviate over-fitting. We report the test performance based on the best epoch of the validation set. For our method, we adopt the Adam optimizer with a learning rate searched in $\{1,3,5\} \times \{10^{-2}, 10^{-3}\}$, and the weight decay rate is searched in $\{1,2,5\} \times \{10^{-4}, 10^{-5}\}$. The layer number of our methods is set as 2, except 3 for Aminer, and the hidden dimension is 64, except 8 for COVID-19. We repeat all experiments 5 times and report the average results and standard deviations. Additionally, we return DHGAS [39] because their code leaks temporal information.

5.4 Main results

Experimental results as shown in table 2 demonstrate that we achieve optimal performance across multiple tasks. Our model consistently ranks first on average across all datasets. Next, we introduce evaluation metrics and analyze the experimental results from three tasks.

5.4.1 Link Prediction

For link prediction task, we use the are under the ROC curve (AUC) as evaluation metrics. AUC is the probability that the model predicts the score of a random positive example greater than that of a random negative one. The AUC results of different methods on the recommendation task are reported in Table 2. We have the following findings. (1) HDGNNs such as DyHATR, HGT+, etc., generally outperform GNNs, demonstrating the importance of incorporating temporal information when dealing with HTGs. (2) DHGAS and DiffMG, as automated sota baseline, reports reasonably good results and outperforms most manually designed heterogeneous methods, demonstrating the potentials of automatically designing neural architectures. However, DiffMG struggles to capture the dynamic information, while DHGAS, despite taking time information into account, lacks in-depth exploration of the temporal dependence of relations in HTGs, which leads to sub-optimal performance in both methods. This proves the significance of mining temporal dependence of relations for downstream tasks. (3) Both HGT+ and DHGAS, which are variants of the Transformer [26] model, are unable to handle large-scale dynamic graphs like OGBN-MAG due to their significant GPU memory consumption. (4) **ReTag** achieves the best result on three dataset with a large margin, i.e, improving the AUC by approximately 3%, 1.8% and 2.1% over the most competent baseline. The results demonstrate that **ReTag** can effectively capture diverse semantic information to handle the link prediction task on HTGs datasets driven by temporal dependence of relations learning.

5.4.2 Node Classification

Next, we compare different methods for the node classification task. We use Macro-F1 as evaluation metrics for node classification, which represents the unweighted mean of the F1-score for each label. The F1-score can be interpreted as a harmonic mean of the precision and recall, where precision is the proportion of true positive in the predicted positive samples. From the results also shown in Table 2, we have the following observations: (1) DiffMG reports the competitive result on the node classification task, demonstrating the effectiveness of exploring structural information, even without considering temporal. (2) HGT+ and DHGAS have demonstrated excellent performance. However, there is still a gap between them and our method because they compress the HTGs into a single graph. In this manner, historical aggregation experiences are discarded, resulting in the loss of temporal information and sub-optimal performance. (3) Our proposed method once again achieves the best results, improving the Macro-F1 score by approximately 4%. These results demonstrate that **ReTag** effectively handles the node classification task by exploring the trend in the importance of neighbors to generate better spatio-temporal representation.

5.4.3 Node Regression

For the node regression task, we use the mean absolute error (MAE) as evaluation metrics, which measure the average magnitude of errors between predicted and actual values. We report the results in Table 2 and observe the following findings: (1) For this task, HDGNN baselines (*i.e.*, DyHATR and HTGNN) significantly outperforms static methods, underscoring the critical role of modeling temporal information in predicting COVID-19 cases, a finding consistent with existing literature. (2) Similar to the other two tasks, **ReTag** again achieves the best performance. The results demonstrate that **ReTag** can adaptively handle diverse applications of HTGs.

5.5 Ablation Studies

In this section, we compare ReTag with its three variants on dataset Aminer, OGBN-MAG and Ecomm to validate the effectiveness of each component in ReTag. The results are given in Table 3 and the description of the used variants are given as below.

- ReTag$_{w/oR}$ denotes ReTag without the relation time-aware learning, *i.e.*, all nodes update their representations according the original adjacency matrix.
- ReTag$_{w/oA}$ denotes ReTag without the attention in spatial heterogeneity learning., *i.e.*, the attributes of all types of nodes are simply added.
- ReTag$_{w/oT}$ denotes ReTag without the temporal dependence learning., *i.e.*, all nodes only update their spatial representation.

The results demonstrate that ReTag consistently outperforms all its variants across the three link prediction datasets. Additionally, we observed that ReTag without relation time-aware learning exhibited the most pronounced decline in performance. This also validates our motivation, emphasizing the importance of learning temporal dependence of relations.

Table 3. Ablation study results on various ReTag variants.

Dataset	Aminer	OGBN-MAG	Ecomm
ReTag$_{w/oR}$	82.84 ± 0.61	88.21 ± 0.98	66.73 ± 1.98
ReTag$_{w/oA}$	89.17 ± 0.85	92.17 ± 1.22	70.12 ± 1.28
ReTag$_{w/oT}$	87.98 ± 0.70	90.84 ± 0.98	69.66 ± 1.89
ReTag	**89.88 ± 0.81**	**92.98 ± 1.23**	**70.73 ± 1.12**

5.6 Additional Analyses

5.6.1 Efficiency Analysis

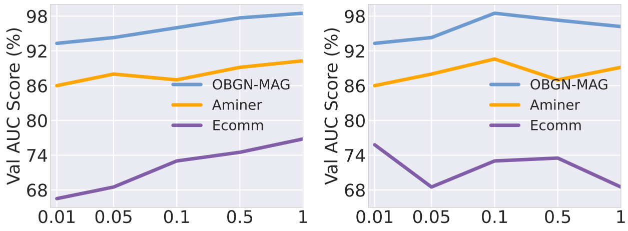
Table 4 compares the training cost of our method against existing works. Here, HTGNN is an attention-based dynamic heterogeneous graph neural network, and DHGAS is an architecture search method for dynamic heterogeneous attention graph neural network. Note that we include the time for both the training of the supernet and the retraining process of DHGAS for a fair comparison. We have the following findings. (1) We achieve superior training efficiency across all datasets compared to previous work, especially notable in the Aminer, OGBN-MAG, and COVID-19 datasets. On the COVID-19 dataset, we are nearly 3 times faster than HTGNN and 20 times faster than the DHGAS. This could be attributed to the time-consuming training of the extensive and non-shared parameters associated with attention mechanisms when handling large datasets or datasets with extended temporal spans. (3) Experimental results demonstrate the competitiveness of our approach in terms of training efficiency.

Table 4. Training cost compared in GPU second.

Dataset	Aminer	OGBN-MAG	Ecomm	YELP	COVID-19
HTGNN	472	1132	135	165	1403
DHGAS	351	-	206	164	11,120
ReTag	240	450	130	101	547

5.6.2 Impact of Hyper-parameters

The temperature τ_{max} and τ_{min} controls the sharpness of the output distribution of softmax. In this work, τ is steadily annealed from an initial value τ_{max} to a small value τ_{min}. We vary τ_{max} and τ_{min} in $\{0.01, 0.05, 0.1, 0.5, 1\}$ to explore the impact of parameters on the validation performance for getting the optimal parameter for testing. As shown in Figure 3(a), we observe that a larger initial temperature τ_{max} is suitable for these three datasets. This is likely because a small initial temperature value may lead to gradient oscillation and training instability. For the parameter τ_{min}, it determines the final sharpness. As shown in Figure 3(b), we observe varying preferences for sharpness across different datasets. Therefore, we choose different τ_{min} parameters based on the validation set for better test results.



(a) τ_{max}

(b) τ_{min}

Figure 3. Comparison of the different temperature τ_{max} and τ_{min} on the link prediction dataset (AUC%).

6 Conclusion

In this work, we rethink the essential properties of HTGs, and find the limitations of the previous HDGNN paradigm for handling HTGs. Specifically, it failed to consider learning the temporal dependence of relations. To bridge the gap left by previous work, we propose a novel heterogeneous temporal graph learning paradigm called ReTag. ReTag learns the temporal dependence of relations through past edge weight and evolving node attributes to generate the time-aware relation adjacency matrices for subsequent spatio-temporal learning. This process utilizes historical aggregation experiences to inspire subsequent aggregations, resulting in better node representations for downstream tasks. As far as we know, ReTag is the first attempt to perform learning of temporal dependence of relation. Based on extensive experiments, ReTag outperforms all baseline methods in performance, which fully proves the effectiveness of our method.

Acknowledgements

The work is supported by the National Natural Science Foundation of China (No. 62272487).

References

[1] Y. Bi, L. Song, M. Yao, Z. Wu, J. Wang, and J. Xiao. A heterogeneous information network based cross domain insurance recommendation system for cold start users. In *Proceedings of the international ACM SIGIR conference on research and development in information retrieval*, pages 2211–2220, 2020.

[2] Y. Cao, H. Peng, J. Wu, Y. Dou, J. Li, and P. S. Yu. Knowledge-preserving incremental social event detection via heterogeneous gnns. In *Proceedings of the Web Conference*, pages 3383–3395, 2021.

[3] W. Cong, S. Zhang, J. Kang, B. Yuan, H. Wu, X. Zhou, H. Tong, and M. Mahdavi. Do we really need complicated model architectures for temporal networks? In *International Conference on Learning Representations*, pages 1–27, 2023.

[4] da Xu, chuanwei ruan, evren korpeoglu, sushant kumar, and kannan achan. Inductive representation learning on temporal graphs. In *International Conference on Learning Representations*, pages 1–19, 2020.

[5] S. Deng, S. Wang, H. Rangwala, L. Wang, and Y. Ning. Cola-gnn: Cross-location attention based graph neural networks for long-term ili prediction. In *Proceedings of the ACM international conference on information & knowledge management*, pages 245–254, 2020.

[6] Y. Ding, Q. Yao, H. Zhao, and T. Zhang. Diffmg: Differentiable meta graph search for heterogeneous graph neural networks. In *Proceedings of the ACM SIGKDD Conference on Knowledge Discovery & Data Mining*, pages 279–288, 2021.

[7] Y. Fan, M. Ju, S. Hou, Y. Ye, W. Wan, K. Wang, Y. Mei, and Q. Xiong. Heterogeneous temporal graph transformer: An intelligent system for evolving android malware detection. In *Proceedings of the ACM SIGKDD Conference on Knowledge Discovery & Data Mining*, pages 2831–2839, 2021.

[8] Y. Fan, M. Ju, C. Zhang, and Y. Ye. Heterogeneous temporal graph neural network. In *Proceedings of the 2022 SIAM International Conference on Data Mining*, pages 657–665, 2022.

[9] A. Feng and L. Tassiulas. Adaptive graph spatial-temporal transformer network for traffic forecasting. In *Proceedings of the ACM International Conference on Information & Knowledge Management*, pages 3933–3937, 2022.

[10] X. Fu, J. Zhang, Z. Meng, and I. King. Magnn: Metapath aggregated graph neural network for heterogeneous graph embedding. In *Proceedings of The Web Conference*, pages 2331–2341, 2020.

[11] W. Guan, F. Jiao, X. Song, H. Wen, C.-H. Yeh, and X. Chang. Personalized fashion compatibility modeling via metapath-guided heterogeneous graph learning. In *Proceedings of the international ACM SIGIR conference on research and development in information retrieval*, pages 482–491, 2022.

[12] W. Hu, M. Fey, M. Zitnik, Y. Dong, H. Ren, B. Liu, M. Catasta, and J. Leskovec. Open graph benchmark: Datasets for machine learning on graphs. *Advances in neural information processing systems*, 33:22118–22133, 2020.

[13] Z. Hu, Y. Dong, K. Wang, and Y. Sun. Heterogeneous graph transformer. In *Proceedings of The Web Conference*, pages 2704–2710, 2020.

[14] H. Huang, R. Shi, W. Zhou, X. Wang, H. Jin, and X. Fu. Temporal heterogeneous information network embedding. In *Proceedings of the International Joint Conferences on Artificial Intelligence*, pages 1470–1476, 2021.

[15] Z. Huang, Y. Zheng, R. Cheng, Y. Sun, N. Mamoulis, and X. Li. Meta structure: Computing relevance in large heterogeneous information networks. In *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 1595–1604, 2016.

[16] S. Ji, X. Lu, M. Liu, L. Sun, C. Liu, B. Du, and H. Xiong. Community-based dynamic graph learning for popularity prediction. In *Proceedings of the ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pages 930–940, 2023.

[17] Y. Ji, T. Jia, Y. Fang, and C. Shi. Dynamic heterogeneous graph embedding via heterogeneous hawkes process. In *The European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases*, pages 388–403, 2021.

[18] Y. Ji, M. Yin, Y. Fang, H. Yang, X. Wang, T. Jia, and C. Shi. Temporal heterogeneous interaction graph embedding for next-item recommendation. In *European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases*, pages 314–329, 2021.

[19] J. Jin, J. Qin, Y. Fang, K. Du, W. Zhang, Y. Yu, Z. Zhang, and A. J. Smola. An efficient neighborhood-based interaction model for recommendation on heterogeneous graph. In *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, page 75–84, 2020.

[20] T. N. Kipf and M. Welling. Semi-supervised classification with graph convolutional networks. In *International Conference on Learning Representations*, pages 1–14, 2017.

[21] W. Luo, H. Zhang, X. Yang, L. Bo, X. Yang, Z. Li, X. Qie, and J. Ye. Dynamic heterogeneous graph neural network for real-time event prediction. In *Proceedings of the ACM SIGKDD international conference on knowledge discovery & data mining*, pages 3213–3223, 2020.

[22] A. Milani Fard, E. Bagheri, and K. Wang. Relationship prediction in dynamic heterogeneous information networks. In *European Conference on Information Retrieval*, pages 19–34, 2019.

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

[24] A. Sankar, Y. Wu, L. Gou, W. Zhang, and H. Yang. Dysat: Deep neural representation learning on dynamic graphs via self-attention networks. In *Proceedings of the international conference on web search and data mining*, pages 519–527, 2020.

[25] M. Schlichtkrull, T. N. Kipf, and P. Bloem. Modeling relational data with graph convolutional networks. In *The Semantic Web*, pages 593–607, 2018.

[26] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

[27] P. Velickovic, G. Cucurull, A. Casanova, A. Romero, P. Liò, and Y. Bengio. Graph attention networks. In *International Conference on Learning Representations*, pages 1–14, 2018.

[28] J. Wang, Q. Chen, D. Zeng, Z. Song, C. Chen, and M. Guo. Stag: Enabling low latency and low staleness of gnn-based services with dynamic graphs. *arXiv preprint arXiv:2309.15875*, 2023.

[29] X. Wang, H. Ji, C. Shi, B. Wang, Y. Ye, P. Cui, and P. S. Yu. Heterogeneous graph attention network. In *Proceedings of the Web Conference*, pages 2022–2032, 2019.

[30] X. Wang, H. Ji, C. Shi, B. Wang, Y. Ye, P. Cui, and P. S. Yu. Heterogeneous graph attention network. In *The world wide web conference*, pages 2022–2032, 2019.

[31] X. Wang, Y. Lu, C. Shi, R. Wang, P. Cui, and S. Mou. Dynamic heterogeneous information network embedding with meta-path based proximity. *IEEE Transactions on Knowledge and Data Engineering*, 34: 1117–1132, 2020.

[32] Z. Wang, H. Zhao, F. Liang, and C. Shi. Node-dependent semantic search over heterogeneous graph neural networks. In *Proceedings of the ACM International Conference on Information and Knowledge Management*, pages 2646–2655, 2023.

[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 *The European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases*, pages 282–298, 2021.

[34] L. Yang, Z. Xiao, W. Jiang, Y. Wei, Y. Hu, and H. Wang. Dynamic heterogeneous graph embedding using hierarchical attentions. In *European Conference on Information Retrieval*, pages 425–432, 2020.

[35] J. You, T. Du, and J. Leskovec. Roland: graph learning framework for dynamic graphs. In *Proceedings of the ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pages 2358–2366, 2022.

[36] P. Yu, C. Fu, Y. Yu, C. Huang, Z. Zhao, and J. Dong. Multiplex heterogeneous graph convolutional network. In *Proceedings of the ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pages 2377–2387, 2022.

[37] S. Yun, M. Jeong, R. Kim, J. Kang, and H. J. Kim. Graph transformer networks. *Advances in Neural Information Processing Systems*, 32:1–11, 2019.

[38] C. Zhang, D. Song, C. Huang, A. Swami, and N. V. Chawla. Heterogeneous graph neural network. In *Proceedings of the ACM SIGKDD international conference on knowledge discovery & data mining*, pages 793–803, 2019.

[39] Z. Zhang, Z. Zhang, X. Wang, Y. Qin, Z. Qin, and W. Zhu. Dynamic heterogeneous graph attention neural architecture search. pages 1–9, 2023.

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