Simplifying Temporal Heterogeneous Network for Continuous-Time Link Prediction

Ce Li
Iowa State University
Ames, IA, USA
celi@iastate.edu

Rongpei Hong
University of Electronic Science and Technology of China
Chengdu, Sichuan, China
rongpei.hong@std.uestc.edu.cn

Xovee Xu*
University of Electronic Science and Technology of China
Chengdu, Sichuan, China
xovee.xu@gmail.com

Goce Trajcevski
Iowa State University
Ames, IA, USA
gocet25@iastate.edu

Fan Zhou
University of Electronic Science and Technology of China
Chengdu, Sichuan, China
Kash Institute of Electronics and Information Industry
Kashi, Xinjiang, China
fan.zhou@uestc.edu.cn

ABSTRACT

Temporal heterogeneous networks (THNs) investigate the structural interactions and their evolution over time in graphs with multiple types of nodes or edges. Existing THNs describe evolving networks as a sequence of graph snapshots and adopt mechanisms from static heterogeneous networks to capture the spatial-temporal correlation. However, these works are confined to the discrete-time setting and the implementation of stacked mechanisms often introduces a high level of complexity, both conceptually and computationally. Here, we conduct comprehensive examinations and propose STHN, a simplifying THN for continuous-time link prediction. Concretely, to integrate continuous dynamics, we maintain a historical interaction memory for each node. A link encoder that incorporates two components - type encoding and relative time encoding - is introduced to encapsulate implicit heterogeneous characteristics of interaction and extract the most informative temporal information. We further propose to use a patching technique that assists with Transformer feature extractor to support the interaction sequence with long histories. Extensive experiments on three real-world datasets empirically demonstrate that STHN outperforms state-of-the-art methods with competitive task accuracy and predictive efficiency on both transductive and inductive settings.

CCS CONCEPTS

• **Information systems** → **Information systems applications**;
• **Computing methodologies** → **Machine learning**.

*Corresponding author

CIKM '23, October 21–25, 2023, Birmingham, United Kingdom

ACM ISBN 979-8-4007-0124-5/23/10.
https://doi.org/10.1145/3583780.3615059

KEYWORDS

temporal heterogeneous network; graph representation learning; temporal link prediction

ACM Reference Format:
Ce Li, Rongpei Hong, Xovee Xu, Goce Trajcevski, and Fan Zhou. 2023. Simplifying Temporal Heterogeneous Network for Continuous-Time Link Prediction. In *Proceedings of the 32nd ACM International Conference on Information and Knowledge Management (CIKM '23), October 21–25, 2023, Birmingham, United Kingdom.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3583780.3615059

1 INTRODUCTION

In many real-world applications – such as social networks [43, 44], transportation [41], recommendation systems [42], citation networks [36], etc. – the data capturing the interplay of main entities can be represented as an evolving network of heterogeneous structures. Given the diversity of the types of nodes and relationships, the interaction of links with the nodes is extracted and heterogeneous network representation learning is often used to map nodes to low-dimensional space, preserving the heterogeneities of both node properties and structures, which is essential for enabling graph-related services [5, 17, 28, 34].

Traditional learning paradigms for heterogeneous networks have often focused on static settings – however, many heterogeneous networks are time-dependent, i.e., the graph structure and features are evolving over time [14, 30, 33]. Several recent studies have acknowledged the necessity for adaptations for dynamic scenarios and began to incorporate discrete dynamics into heterogeneous graph learning to capture the evolution over time for dynamic (GCN [15] and HGT [11, 29] necessitate specific adaptations for dynamic scenarios). The prevalent approach employed in existing dynamic heterogeneous graph learning methods is based on capturing snapshots of the graph at different timestamps [5, 13, 14, 34]. As shown in Figure 1(a), these snapshots refer to the state of the graph at a specific time instant and are learned through a static heterogeneous graph encoder to obtain their representations. Typically,

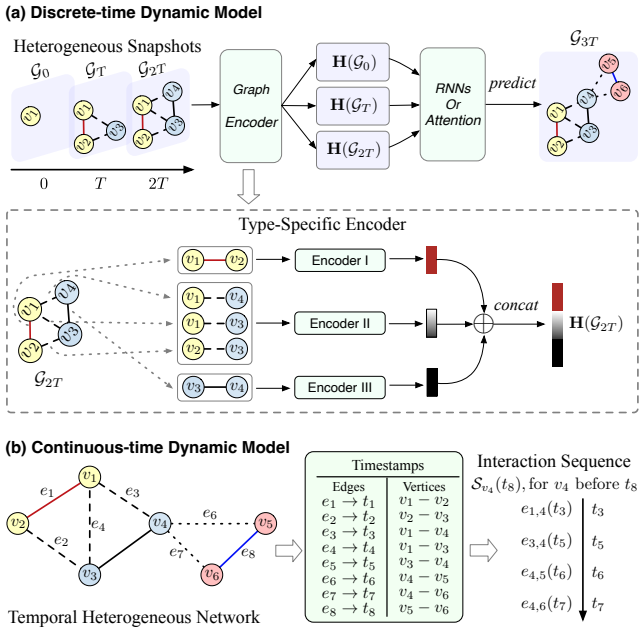

Figure 1: Visual illustration of dynamic heterogeneous graph models. (a) Discrete-time dynamic model with the type-specific graph encoder takes a two-step strategy (graph encoder followed by a sequence model) to capture spatial-temporal correlations. (b) Continuous-time dynamic model memorizes the historical interaction streams for each node.

the encoder splits the graph snapshot into multiple parts according to the node or edge types [6, 32, 39], simplifying the graph learning process from a heterogeneous network to multiple homogeneous subnetworks. These snapshots are subsequently encoded to produce a series of heterogeneity-preserving embeddings, and sequence-based models such as recurrent neural network (RNN) [10] and Transformers [26] are used to aggregate node embeddings at pre-specified timestamps as well as update their representations.

Challenges & Motivation: Despite the achieved improvements (e.g., in effectiveness), most works remain limited to discrete-time settings. However, representing dynamic graphs as a sequence of heterogeneous graph snapshots over time forces the model to digest discrete dynamics. This, in turn, may result in inefficient behavior that may not be acceptable for real-world applications, especially the ones where new edges can appear at *any* time [22]. Clearly, in many such cases, it is preferable for THNs models to make immediate link predictions. From a complementary perspective, learning graph dynamics with discrete snapshots is not scalable to large networks due to the high space usage and redundancy inherent in maintaining multiple snapshots. Another consideration is that current THNs adopt many building blocks from static heterogeneous networks, notably the explicit type-specific design and attention mechanism, as shown in Figure 1(a). The effects of type-specific designs have been investigated in static heterogeneous network studies [18, 40] and it has been shown that type-specific embeddings only bring minor improvements. However, no studies have examined the effects of these designs on dynamic heterogeneous networks. Part of our motivation was to close this gap by conducting comprehensive examinations of these building blocks and designs carried over from static graph learning models. Towards that, we got two significant findings: (1) splitting historical interactions into disparate subnetworks breaks the order that originally exists in the interaction sequences and hinders the model to capture the correlations between different but nearby relationships. (2) self-attention mechanisms are critical feature extractors, but their performance and effectiveness will be degraded when dealing with long interaction sequence.

Approach and Contributions: Motivated by the aforementioned observations and findings, we propose a novel temporal heterogeneous learning model STHN for link prediction, which studies heterogeneous node representations in a simpler and continuous-time manner. We maintain a *historical interaction memory* for each node (cf. Figure 1(b)) to capture node interaction dynamics, enabling immediate link prediction for both seen and unseen nodes. Moreover, we introduce a link encoder that incorporates two components – *type encoding* and *relative time* encoding – to encapsulate implicit heterogeneous characteristics of interaction and extract the most informative temporal information. Inspired by the patching technique [4], we decompose long interaction histories into patches as input tokens for a Transformer-based feature extractor. Extensive experiments on three real-world heterogeneous network datasets demonstrate that the proposed STHN model can effectively predict future links and outperform strong baselines. Our contributions can be summarized as follows:

- We identify the challenges in predicting links on temporal heterogeneous networks and present a continuous dynamic graph learning approach.
- We propose STHN, a novel method that does not use the common type-specific design and simplifies neighborhood aggregation by implicitly incorporating structural heterogeneity and temporal information.
- To enhance STHN's capability for learning long interaction sequences, we apply a patching technique that maintains local semantic proximities while reducing computational costs.
- We provide extensive experimental evaluations comparing STHN against seven baselines (from three different categories) over three datasets, and empirically demonstrate that STHN converges more rapidly and predicts links more accurately in both transductive and inductive settings.

2 PRELIMINARIES

We now introduce the necessary background and formally define the problem. Typically, a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ is a pair consisting of a set of vertices $\mathcal{V}$ and a set of edges $\mathcal{E} \subseteq \mathcal{V} \times \mathcal{V}$. This definition implicitly assumes a *homogeneity* among the (types of) nodes and edges. In a *heterogeneous* setting, $\mathcal{V}$ is a union of disjoint sets – i.e., $\mathcal{V} = V_1 \cup V_2 \cup \ldots \cup V_K$ and every node from a given subset is of the same type. More formally, we assume a collection of types $\mathcal{A} = \{A_1, A_2, \ldots, A_{T_A}\}$ to denote a set of node types. Each $v_i \in \mathcal{V}$ is associated with – equivalently, has properties that make it an instance of – a particular A_j. A type (respectively, property) can be either *primitive* (i.e., consisting of a single, atomic attribute) or *composite* (i.e., consisting of structures, possibly nested). Thus, we

assume that there exists a mapping $\phi : \mathcal{V} \rightarrow \mathcal{A}$. In the context of Figure 1(a) this is shown by the different colors of the nodes (we do not show the actual structures corresponding to the respective properties). Furthermore, we assume that each edge $e_i \in \mathcal{E}$ has a distinct type which essentially means: (1) it connects a pair of nodes of specific types; and (2) it has a particular structure describing its properties (e.g., weight). We use $\mathcal{R} = \{R_1, R_2, \ldots, R_{T_R}\}$ to denote the collection of edge-types – and we have a corresponding mapping $\psi : \mathcal{E} \rightarrow \mathcal{R}$. In the context of Figure 1(a), edges with the same color and shape belong to the same edge type (relation). For example, a solid red edge links yellow nodes; solid blue edge links red nodes; etc. With these in mind, we have the following definitions:

Definition 2.1. (**Heterogeneous Graph**) A heterogeneous graph is a six-tuple $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathcal{A}, \mathcal{R}, \phi, \psi)$, where $|\mathcal{A}| + |\mathcal{R}| > 2$ (in the case that $|\mathcal{A}| = |\mathcal{R}| = 1$, the graph is *homogeneous*).

Definition 2.1 formalized the concept of a graph with many types of nodes and edges. However, it did not capture the notion of evolution of respective entities over time. Following [32], we have:

Definition 2.2. **Temporal Heterogeneous Network**. A temporal heterogeneous network is a heterogeneous network in which certain links occur in different time instants, i.e.:

$$\mathcal{E} = \{(e_{i,j}(t), r) | v_i, v_j \in \mathcal{V}, r \in \mathcal{R}\}, \tag{1}$$

where $e_{i,j}(t)$ is a link between node v_i and v_j, $r (r \in \mathcal{R})$ is the link type, and t is the timestamp when $e_{i,j}(t)$ is built.

An example is provided in Figure 1 which shows the timestamps of occurrence for the respective edges in the graph.

Definition 2.3. **Interaction Sequence**. For a node v, its aggregated historical temporal information before timestamp t_n presented in chronological order, is defined as an *interaction sequence*:

$$\mathcal{S}_v(t_n) = \{(e_{i,j}(t), r) | v = v_i \text{ or } v = v_j, t < t_n\}. \tag{2}$$

Problem Statement. Given the definitions above, the problem of *Continuous-Time Link Prediction* is defined as: Given a node $u \in \mathcal{V}$ and a timestamp t_n, the goal of THNs is to learn a d-dimensional time-dependent node representation $\mathbf{h}_u(t_n) \in \mathbb{R}^d$ based on its (historical) interaction sequence $\mathcal{S}_u(t_n)$. With the learned node representation $\mathbf{h}_u(t_n)$ and $\mathbf{h}_v(t_n)$ for nodes (u, v), the *link prediction* is used to predict whether they are connected at timestamp t_n. Note that the link existence prediction between the nodes is not our only objective. In addition, we expect the models to make relationship type predictions for graphs with multiple types of links.

3 METHODOLOGY

In this section, we present the architecture of the proposed STHN model and discuss its main aspects. The overall framework is depicted in Figure 2 and, as shown, it consists of three main modules: (1) *Heterogeneous link encoder* is designed to encode different link features (e.g., link feature, link type, and link timestamps) into embeddings. (2) *Semantic patches fusion* is designed to summarize the information from temporal interaction sequence; (3) *Link predictor* makes a prediction for whether a link exists between two nodes based on the learned temporal node representations. In the next three subsections, we describe each of the modules in detail.

3.1 Heterogeneous Link Encoder

The historical interactions of the nodes are critical for predicting the respective future behaviors. Impacts of those interactions are not only related to the node features but also depend on the diverse relationships and interactions between the node and its neighborhood like, for example, type, frequency, and time instants (i.e., temporal information). Since there exist multiple kinds of interactions that may happen between two nodes, we maintain all the historical interactions $\mathcal{S}_u(t_n)$ that a node u has participated in before time t_n. Each link $(e_{i,j}(t), r) \in \mathcal{S}_u(t_n)$ is labeled with a timestamp t and the sequence of interactions is maintained in a chronological order. A particular link $(e_{i,j}(t), r)$, in addition to the timestampe t is also associated with another (second) basic feature – its type r. To capture the heterogeneities and produce temporal link embeddings, we develop a heterogeneous link encoder (cf. Figure 2) that includes two components - type encoding and time encoding – to enable the implicit involvement of heterogeneous interactions and extract the most important temporal information.

Type encoding. In a heterogeneous graph, any given node may have one or multiple types of links. To preserve the structural heterogeneity, dynamic heterogeneous GNNs differ in learning link type information. Existing works [6, 12, 39] divide associated links into different subnetworks according to their types, i.e., links of the same type will be grouped together into the same subnetwork. Each such subnetwork is equipped with a type-specific graph encoder to gather and aggregate the same-type neighbor information. This design is popular for static heterogeneous graph methods – however, in such settings there are no temporal dependencies between links. However, order and nearby cooperation relationships that exist in a dynamic/evolving interaction sequence cannot be actually captured in fixed/static setting. In fact, we will show quantitatively in Section 4 that type-specific designs are not effective for dynamic heterogeneous graph learning. In this work we introduce an implicit type signal as a part of link embedding. The link type set $\mathcal{R}$ can be considered as a collection of categorical variables with a finite set of values. Therefore, one-hot encoding $\mathbf{x}_{\mathcal{R}}(r) \in \mathbb{R}^{|\mathcal{R}|}$ is used here as an alternative way to produce and distinguish the type encoding, which is simple and easy to generalize.

Time encoding. The other key feature of temporal heterogeneous network is the link timestamp t, which indicates when the link was created and reveals critical temporal information [35]. Here, we follow the approaches in [3, 45] and involve the timestamp into training by providing a relative time encoding function $\Phi : t' \rightarrow \mathbb{R}^{d_t}$. The purpose is to map the relative timestamp value $t' = t_n - t$ from time domain to a d_t-dimensional vector space. The obtained time vectors (encoding) should show similarity or spatial proximity in vector space when the timestamps are close in time-axis. Specifically, the encoding function can be represented as

$$\Phi(t') = \cos(t' \times \omega), \omega = \{\alpha^{-(i-1)/\beta}\}_{i=1}^{d_t}, \tag{3}$$

where d_t is the dimension of the time encoding, and $\alpha = \beta = \sqrt{d_t}$. The mapping function Φ first maps relative time t' to monotonically exponentially decreasing vector $t' \times \omega \in (0, t]$, and then projects all the values of the vector $t' \times \omega$ to [-1, 1] by using cosine function. However, during our experimental evaluations we found that in practice this module may not work well for all the datasets, since

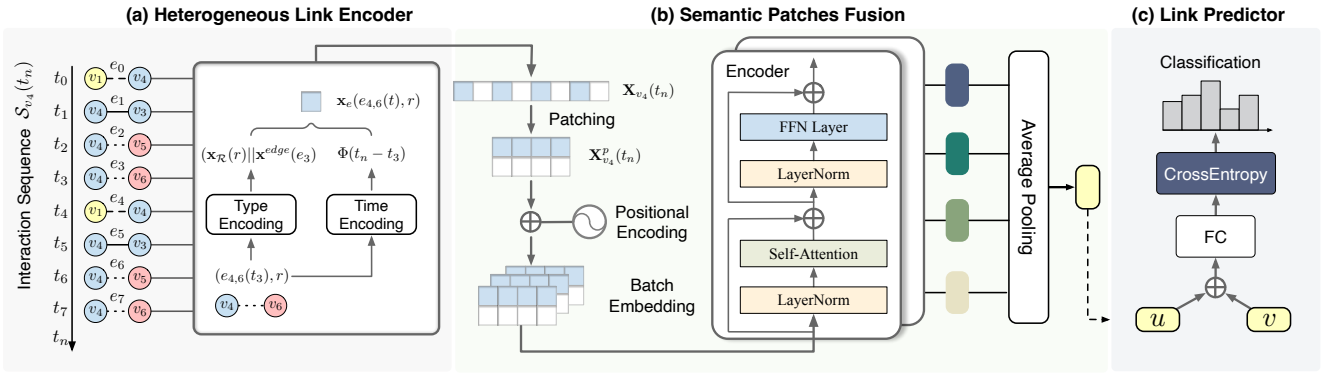

Figure 2: STHN architecture. (a) Heterogeneous Link Encoder with two components - type encoding and time encoding - embeds historical interaction sequence to produce temporal link representation. (b) In semantic patches fusion, sequential representations are divided into different patches, which are set as the token input of Encoder. Average mean pooling is used to compress the patch embeddings into a single vector. (c) Combining the representations of nodes u and v, the link predictor makes link prediction with FC layer and CrossEntropy loss.

their respective temporal scales are different. For example, in the MathOverflow dataset, timestamps are recorded at a granularity of seconds, whereas in the Netflix dataset, updates occur on a daily basis. This can produce numerical differences in timescales and contribute to instable training performance. To address this issue, we perform a MinMax Scaling for the relative time feature to first shift the data by minimum value and then scale it by the amount of $1/(max - min)$ represented as: $t' = \frac{t' - \textsc{Min}(t')}{\textsc{Max}(t') - \textsc{Min}(t')}$. After the normalization, we introduce a hyper-parameter μ – the maximum numerical difference of timestamp into encoding equation: $\Phi(t') = \cos(\mu t' \times \omega)$ to make the time encoding distinguished. Given the outputs of type embedding and time embedding, we can simply concatenate them with the original edge feature $\mathbf{x}^{\text{edge}}(e_{i,j}(t), r)$ into a vector and generate the mixing link embedding with one-layer Mlp as:

$$\mathbf{x}_e(e_{i,j}(t), r) = \textsc{Mlp}(\mathbf{x}_{\mathcal{R}}(r) || \Phi(t') || \mathbf{x}^{\text{edge}}(e_{i,j}(t), r)). \tag{4}$$

Adding to previous discussions, we emphasize that the encoding components allow the link encoder to receive and digest implicit heterogeneous characteristics and temporal information in a single building block, which is both conceptually simpler and enables a simplified network design, in turn, leading to better generalizability.

3.2 Semantic Patches Fusion

Sequence models are typically employed to aggregate neighboring link embeddings in traditional heterogeneous network learning methods, wherein point-wise input tokens are fed into the training network. However, the efficiency and effectiveness of this approach can be questionable when dealing with long sequences. Since nearby tokens usually exhibit stronger correlations compared to distant ones, we propose a shift from concentrating on individual point levels – and integrate local semantic information at a patch level.

Patching. Patching technique was introduced in ViT [4] to reshape a 2D image input into a 1D sequence of token embeddings for the purpose of adaptation to the Transformer architecture. Different from vision features, the historical interactions of a node in temporal network are already "natively" in 1D. In our case, each input interaction sequence $\mathcal{S}_u(t_n)$ of node u is first divided into patches that can be either overlapped or not. Let the link embeddings of $\mathcal{S}_u(t_n)$ be denoted as $\mathbf{X}_u(t_n) \in \mathbb{R}^{|\mathbf{X}_u(t_n)| \times d_S}$, where the patch length is p. The whole sequence will be divided into p-length patches and each patch is equipped with $N = \frac{|\mathcal{S}_u(t_n)|}{p}$ interactions. Finally, the sequence embeddings of patches $\mathbf{X}_u^p(t_n) \in \mathbb{R}^{p \times N \cdot d_S}$ is generated.

Transformer Encoder. As shown in the central part of Figure 2, we use a "vanilla" Transformer encoder (cf. [26]) to extract the reshaped patch features and aggregate them in the latent representation space. Specifically, the sequence embeddings of patches $\mathbf{X}_u^p(t_n)$ are first normalized by Layer Normalization [1] for Transformer input $\mathbf{S}_{\text{input}} = \text{LN}(\mathbf{X}_u^p(t_n))$ and then mapped to $\mathbf{Q}, \mathbf{K}, \mathbf{V}$ by three projection matrices $\mathbf{W}_Q, \mathbf{W}_K$ and $\mathbf{W}_V \in \mathbb{R}^{N \cdot d_S \times d_k}$ and an additive positional encoding $\mathbf{W}_{pe} \in \mathbb{R}^{P \times d_k}$. The matrices $\mathbf{Q}, \mathbf{K}$ and $\mathbf{V}$ are intermediate representations of the Transformer input and their rows can be treated as *queries, keys* and *values*, respectively. Then a scaled dot-product attention layer is employed:

$$\textsc{Attn}(\mathbf{Q}, \mathbf{K}, \mathbf{V}) = \textsc{Softmax}(\frac{\mathbf{Q}\mathbf{K}^\top}{\sqrt{d_k}})\mathbf{V} \tag{5}$$

to generate the weighted sum of the value vectors. The architecture of the Transformer encoder consists of two parts: self-attention layer and channel forward layer, which are linked by a skip connection which are performed as:

$$\mathbf{S}_{\text{token}} = \mathbf{S}_{\text{input}} + \textsc{Attn}(\mathbf{Q}, \mathbf{K}, \mathbf{V}), \tag{6}$$

$$\mathbf{S}_{\text{output}} = \mathbf{S}_{\text{token}} + \textsc{Mlp}_2(\textsc{Gelu}(\textsc{Mlp}_1(\text{LN}(\mathbf{S}_{\text{token}})))), \tag{7}$$

where $\mathbf{S}_{\text{output}} \in \mathbb{R}^{P \times d_k}$ is the output of Transformer encoder. Here, we use mean pooling to compress $\mathbf{S}_{\text{output}}$ into a single vector $\mathbf{s}_{\text{link}}^{u,t_n} \in \mathbb{R}^{d_k}$ as the link representation of a node u. Note that the application of patching operation actually reduces the number of tokens in Transformer by a factor of N which, in turn, makes it viable for longer sequence inputs. Our experiments in Section 4 demonstrate that this design significantly reduces the computational cost while retaining the effectiveness of the model.

3.3 Temporal Link Predictor

Before making predictions for the link, we start by aggregating the node features and capturing the node identity information from the node's neighborhood. Different from previous models that leverage multi-hop neighborhood information, we define $\mathcal{N}(u; t_n, t)$ as the 1-hop neighbors of node u with link times from t to t_n. Accordingly, we can gather the 1-hop neighborhood information using:

$$\mathbf{s}_{\text{node}}^{u,t_n} = \mathbf{X}_u + \textsc{Mean}\left\{\mathbf{X}_v | v \in \mathcal{N}(u; t_n, t)\right\}, \tag{8}$$

where t_n is the timestamp at which we will make the prediction for u. We note that, in this work, we do not extract the information from all neighbors for a given node. To achieve a reduced space cost (i.e., memory limitation), we set a bound on the maximum values for both the quantity of links and neighbors.

After obtaining the embeddings of a temporal link and a node, we concatenate these embeddings and predict the link existence and link type at time t_n, using the output of the heterogeneous link encoder $\mathbf{s}_{\text{link}}^{u,t_n}$ and node encoder $\mathbf{s}_{\text{node}}^{u,t_n}$. The representation of node u is obtained by concatenating the link and node embeddings:

$$\mathbf{h}_u(t_n) = \left[\mathbf{s}_{\text{link}}^{u,t_n} \| \mathbf{s}_{\text{node}}^{u,t_n}\right]. \tag{9}$$

Recall that our objective is to predict the existence (and type) of a link between two nodes at a specific timestamp based on all the available temporal graph information obtained prior to that timestamp. Such a prediction can be made by examining the two link temporal embeddings via a link predictor based on MLPs:

$$Pred = \textsc{Mlp}([\mathbf{h}_u(t_n) || \mathbf{h}_v(t_n)]). \tag{10}$$

4 EXPERIMENTS

We now present the details of our experimental evaluation. For reproducibility, the code is publicly available at https://github.com/celi52/STHN.

4.1 Setup

Datasets. We use three real-world datasets: MathOverflow [21, 39], Netflix [19], and Movielens [9]. The first dataset is collected from the stack exchange website - Math Overflow and open source at the SNAP platform. There are three different relationships between users on stack exchange: answer to question, comment to question, and comment to answer. Netflix and Movielens datasets are composed of historical film reviews from users on the Netflix platform and MovieLens website, respectively. These movie-related datasets include two types of nodes: users and movies, along with five distinctive link types representing the range of user ratings from 1 to 5. The statistics of each dataset are shown in Table 1. It is important to note that we have not incorporated any domain-specific mechanisms into the proposed STHN methodology – i.e., our results are not bound by any particular data domain(s). While MathOverflow is Q&A centric and Netflix and Movielens are oriented towards film reviews, our method retains the flexibility to be applicable to (data from) other fields like, for example, social networks.

Data preparation. In this study, we use chronological split for training, validation, and test sets (7/1.5/1.5), enabling our model and baselines to work for unseen nodes. Note that, for snapshot-based methods, we keep the default settings as described in the original papers [6, 39].

Table 1: Dataset Statistic

	MathOverflow	Netflix	Movielens
# Nodes	24,818	36,558	2,626
# Edges	506,550	1,500,000	100,000
# Node & Edge Types	1, 3	2, 5	2, 5
Timestamps Frequency	per second	daily	daily
Time Span	2350 days	3 years	7 months

Evaluation and Metrics. We evaluate STHN's efficiency via temporal link prediction task. At a future prediction time t_n, the model utilizes the information accumulated up to time t_n – i.e., we have all the historical information of the source node u and target node v, used to predict the existence or type of link between u and v in timestamp t_n. Following [39], we use the areas under the receiver operating characteristic (AUROC) and Precision-Recall Curve (AUPRC) to evaluate candidate models. For each node u with positive edge (u, v) at time t_n, we randomly selected nodes from the node set as the negative linkage samples for the source node u. Additionally, to test the generalization capability of the trained model on unseen data (those nodes that are not observed during the training phase), we consider two different evaluation rules: **(1)** *Transductive link prediction task* allows the trained model to use all the incoming links after training for validation and testing. **(2)** *Inductive link prediction task* focuses on the links associated with nodes that are not observed during the training phase [23]. Following the task design from [31], there are two types of links: (i) *new vs. new* links – i.e., the links between two unobserved nodes; (ii) *new vs. old* links – i.e., the links between an observed node and an unobserved node.

Baselines. STHN is compared with three groups of different GNNs. They are: **(1)** *Homogeneous static graph neural networks methods.* GraphSAGE [8] is the first inductive graph learning method, which leverages neighborhood sampling and message passing to support large-scale GNN learning. GAT [27] internalizes the self-attention mechanism into GNN neighbor aggregation. It studies the similarity and weights between neighbors by computing the attention. **(2)** *Homogeneous temporal networks.* JODIE [16] presents a continuous homogeneous temporal network and introduces coupled RNN that learns dynamic embeddings of nodes from a sequence of temporal interactions. TGAT [35] proposes to address inductive representation learning on time-dependent graphs and generalizes the functional time encoding in temporal network, which eliminates the need for an additional sequence model (like RNNs) to capture the time information. TGN [22] keeps the same temporal graph modeling as TGAT, but involves a Memory Update strategy in its training phase, which saves the historical states of the nodes and updates their states after each new interaction, and also makes it possible to memorize long term dependencies for the nodes in the graph. **(3)** *Snapshot-based dynamic heterogeneous networks.* DyHATR [39] and HTGNN [6] divide the dynamic heterogeneous graph into different snapshots and apply a two-step strategy (graph encoder followed by a sequence model) to extract the spatial-temporal correlation between these snapshots. Note that, the objective function used in DyHATR trains the model by increasing the neighbor's similarity, which can not obtain the representation for unseen nodes. Thus, we omit DyHATR in inductive experiments.

Table 2: Experimental Comparison for Transductive Temporal Link Prediction Tasks (Dash symbol means model DyHATR can not train with Netflix dataset due to excessive memory usage).

	MathOverflow		Netflix		Movielens		Math. Type		Netflix Type		Movielens Type	
	AUROC	AUPRC	AUROC	AUPRC	AUROC	AUPRC	AUROC	AUPRC	AUROC	AUPRC	AUROC	AUPRC
GraphSAGE [8]	83.49	85.94	92.28	91.82	84.70	81.42	74.92	47.57	68.03	27.64	68.01	27.49
GAT [27]	84.74	84.34	90.53	89.56	82.83	78.96	75.49	47.77	73.74	29.14	68.62	27.37
JODIE [16]	92.64	93.01	95.83	94.79	86.97	83.59	79.64	52.06	76.49	30.75	69.70	28.10
TGAT [35]	85.65	87.86	88.62	85.47	82.80	79.11	77.66	51.15	72.72	27.87	71.15	28.19
TGN [22]	90.21	91.62	96.56	95.59	87.58	83.54	80.02	52.91	75.77	31.43	75.36	31.88
DyHATR [39]	72.51	89.07	-	-	60.62	64.26	63.75	34.52	-	-	55.00	19.58
HTGNN [6]	84.85	81.27	86.39	84.18	76.74	72.79	78.21	49.31	73.03	29.00	67.08	25.65
STHN - Type	93.74	94.15	92.24	92.22	82.68	76.39	91.74	75.29	81.79	38.14	79.35	35.99
STHN - Trans	96.87	97.35	**96.75**	**96.27**	94.92	93.68	92.88	76.77	**85.26**	**42.11**	**84.94**	**42.50**
STHN - Patch	**96.99**	**97.44**	96.68	96.23	**95.13**	**94.09**	**93.43**	**77.94**	85.23	41.93	84.90	41.93

Model Variants. We propose three versions of STHN listed in the bottom part of Table 2. STHN-Type implies we integrate type-specific design in graph encoder, as discussed in Section 1. STHN-Trans means that the original interaction sequence is set as the token input of vanilla Transformer without patching. STHN-Patches indicates that we divide the interaction sequence into patches and obtain the node temporal embedding with semantic patches fusion.
STHN architecture. All of the models are following the same experimental setup. We use 100 hidden dimension for time encoding, transformer encoder, Layernorm and link predictor. We allow at most 20 epochs' validation loss increasing before early stopping. We report the test set AUROC and AUPRC when the best validation loss is achieved. We implemented all the homogeneous temporal networks as well as STHN using PyTorch 1.13.0 using NVIDIA RTX 3090 with 128GB RAM. We set the mini-batch size to 600 and the largest training epochs to 500. For semantic patches fusion module, the number of attention heads is 2, and the searching range of learning rate is [1e-2, 1e-4]. Considering the scales of the datasets, the default lengths K of the interaction sequence for MathOverflow, Netflix, and Movielens are set to 100, 50, and 100, respectively, and the default patch length N is 5.

4.2 Results in Transductive Evaluation Settings

We firstly compare our models with baselines under the transductive setting. Table 2 shows that STHN (and its variants) consistently outperforms the baseline models across all three datasets for both metrics, in both link prediction (first three columns) and type prediction (last three columns). Additionally, we notice that homogeneous temporal network tends to have a better performance in comparison with discrete-time heterogeneous models like DyHATR and HTGNN. The reason is that the homogeneous temporal networks - JODIE, TGAT, and TGN - are all continuous dynamic models and can capture more fine-grained temporal information. While TGN [22] tends to have a closer performance as our approach for link existence prediction on dataset Netflix, it fails to speculate in the link type prediction task. We also see the benefits from involving implicit type information to link encoding, where STHN significantly outperforms the baselines in all link type prediction tasks. On the one hand, the results suggest that type information is critical for link type prediction task. On the other hand, we note that the type-specific encoder design is not the only way to involve link type information. By integrating type encoding and time encoding in heterogeneous link encoder, STHN implicitly involves type and temporal information and also simplifies the architecture design. Moreover, the temporal models surpass the static methods, since a static homogeneous network does not incorporate time encoding of interaction into modeling. As for the link type prediction, we notice that STHN gets a much better performance when compared against the baseline models. This is reasonable since the historical link type information has been embedded in the node representation learning. Finally, we can see from Table 2 that the task of predicting the link type is harder than predicting the existence of a link.

4.3 Results in Inductive Evaluation Settings

In the inductive evaluation phase, we follow the experimental setting specified in [31] and remove the links that connected nodes which showed up during the training phase and split the remaining links into *new vs. new* and *new vs. old*. We describe the results of inductive evaluation in Table 3. It shows that, in comparison with transductive learning, it is more difficult for models to make link predictions for the unseen nodes, which always have limited historical information. On the one hand, the values for the link prediction accuracy of most models in *new vs. old* are lower than those in transductive setting (cf. Table 2). On the other hand, the performance of most models in *new vs. new* is worse than those in *new vs. old*. This, again, is reasonable since the newcoming nodes often have a short history, which makes it harder to forecast their behaviors. Overall, the results demonstrate the superiority of our approach in learning representations for inductive setting too.

4.4 Ablation Study

To verify the effectiveness of heterogeneous link encoding in STHN, we designed two variants (a) Without time encoding (STHN - w/o Time); or (b) Without type encoding (STHN - w/o Type). Figure 3 shows that our model outperforms the variants. The STHN -w/o Type variant brings the most significant performance decline, indicating the importance of the temporal information. Complementary to this, type information does not contribute to the link existence prediction a lot, but plays an important role in link type prediction

Table 3: Experimental Comparison for Inductive Temporal Link Prediction Tasks.

Task		Methods	MathOverflow		Netflix		Movielens		Math. Type		Netflix Type		Movielens Type	
			AUROC	AUPRC	AUROC	AUPRC	AUROC	AUPRC	AUROC	AUPRC	AUROC	AUPRC	AUROC	AUPRC
Inductive	*new vs. new*	GraphSAGE	84.68	87.43	45.55	45.88	28.36	37.85	69.68	41.67	48.94	16.46	49.22	18.89
		GAT	85.33	81.95	70.67	60.86	19.65	35.47	65.08	39.47	49.88	17.74	35.01	15.61
		JODIE	88.71	87.43	59.59	57.85	36.39	39.99	67.28	45.11	48.20	17.33	37.43	13.83
		TGAT	93.57	93.67	73.48	72.39	58.40	71.52	79.09	55.32	57.86	21.41	53.13	21.42
		TGN	94.82	**95.21**	66.46	57.10	47.01	44.39	77.49	52.81	59.61	20.54	38.97	13.88
		HTGNN	88.79	83.41	26.46	43.90	37.26	40.21	76.43	43.55	34.34	13.38	21.97	12.30
		STHN - Trans	**95.26**	95.18	78.08	81.43	**70.53**	73.60	89.87	71.28	63.00	25.89	70.05	31.02
		STHN -Patch	95.00	94.73	**79.12**	**82.65**	69.75	**76.29**	**90.84**	**71.95**	**63.87**	**26.31**	**78.65**	**37.93**
	new vs. old	GraphSAGE	66.60	68.60	94.08	93.64	86.63	83.73	62.84	38.96	70.05	28.88	69.91	28.33
		GAT	85.89	82.72	93.49	92.73	65.24	60.79	65.15	41.90	74.42	29.86	56.79	19.70
		JODIE	91.07	86.76	96.02	94.23	85.41	81.42	73.50	43.82	76.36	30.33	69.19	27.41
		TGAT	89.77	90.60	92.10	89.98	80.24	72.55	78.59	55.93	72.73	27.16	66.76	25.06
		TGN	93.22	93.93	97.05	96.41	89.53	87.47	81.62	60.48	76.54	31.04	71.84	28.18
		HTGNN	77.28	69.89	88.55	86.69	76.67	72.25	75.75	48.10	74.18	29.30	68.98	26.21
		STHN - Trans	92.52	93.47	**97.40**	**97.15**	94.06	92.62	87.83	69.12	84.95	**40.88**	84.02	40.37
		STHN -Patch	**94.70**	**94.48**	97.39	97.08	**95.65**	**94.03**	**91.39**	**74.56**	**85.00**	40.84	**84.57**	**40.84**

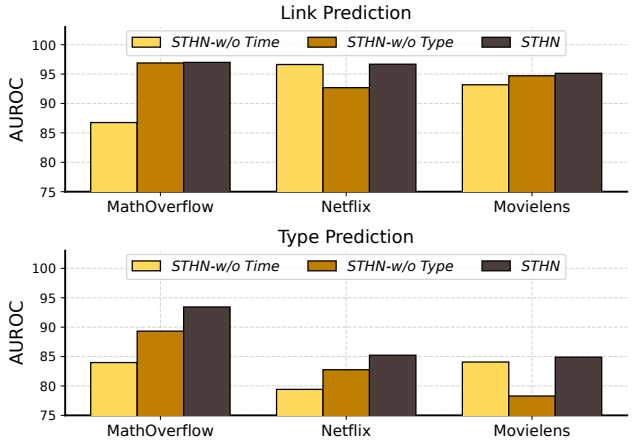

Figure 3: Ablation study: STHN variants w/o time or w/o type encoding. Top: link prediction. Bottom: type prediction.

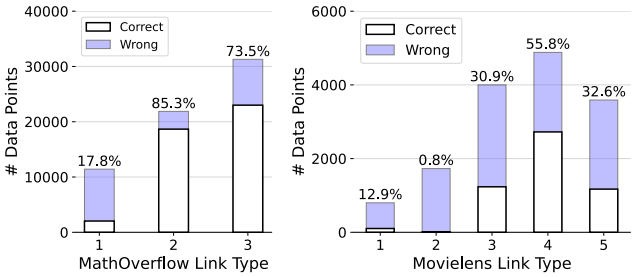

Figure 4: Class distribution and proportion of correct predictions on MathOverflow and Movielens.

task. The comparison demonstrates the effectiveness of temporal link encoder in collecting type and temporal information from historical interaction sequence.

4.5 Factors Affecting the Performance

We considered two sources that can affect the performance of STHN.

Table 4: Hyperparameter investigation of STHN architecture with varying interaction sequence ($K \in \{50, 100, 200\}$) and patch length ($N = \{5, 10\}$).

Models	Setting	MathOverflow	Netflix	Movielens
		AUROC	AUROC	AUROC
STHN-Trans	$K = 50$	96.91	97.24	94.92
	$K = 100$	96.81	96.80	92.27
	$K = 200$	96.73	92.06	88.43
STHN-Patch	$K = 50, N = 5$	96.97	96.88	95.14
	$K = 50, N = 10$	97.00	97.05	94.77
	$K = 100, N = 5$	96.97	96.03	92.57
	$K = 100, N = 10$	96.95	96.56	92.71
	$K = 200, N = 5$	96.95	96.27	88.73
	$K = 200, N = 10$	96.95	96.27	89.23

The effect of imbalanced class distribution. When comparing the results of metrics, it must be pointed out that the AUPRC metric always has lower values than AUROC for the link type prediction task, while they achieve close values for the traditional (i.e., existential) link prediction task. Generally, both metrics (AUROC and AUPRC) are useful in classification tasks. However, they differ in evaluation aspects of the performance for the classifier. AUROC is the area under the precision-recall curve and measures how well the model distinguishes between classes. AUPRC evaluates the trade-off between precision and recall at different thresholds. Under macro-average mode, the precision and recall are computed for each class independently and then averaged. It does not take the weight of class which makes AUPRC metric more sensitive to class imbalance. We provide the class distribution and proportion of correct predictions on Movielens dataset in Figure 4. It can be seen that the values of prediction accuracy for different classes are imbalanced. For example, the link type with a bigger proportion of the dataset, e.g., link type 4 in Movielens, gets 55.8% accuracy in link type prediction, while the link type 1 which only has around 1000 data points gets 12.9% accuracy for link type prediction. Although the model performs well on the majority classes, poor performance

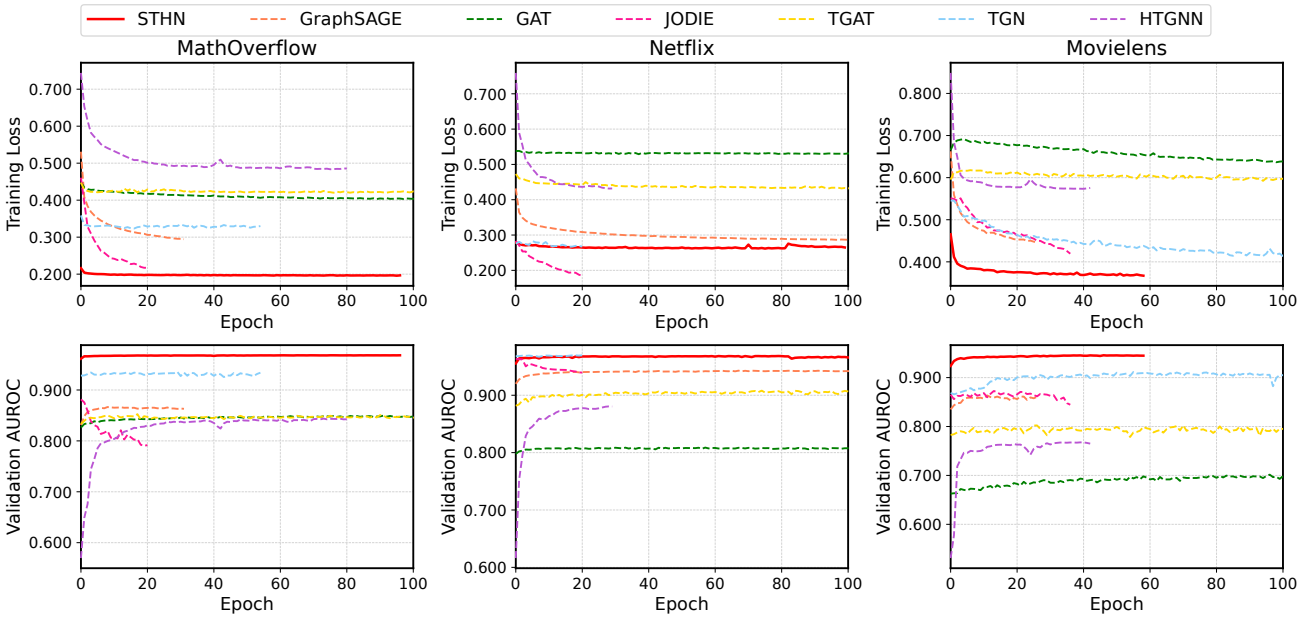

Figure 5: Training loss (top row) and validation AUROC (bottom row) on MathOverflow, Netflix, and Movielens datasets.

Table 5: Transductive link prediction computational costs on MathOverflow. "#Params" denotes the number of parameters. "Train(s)" and "Prediction(s)" denote the average epoch training time and prediction time, respectively.

Model	#Params	Train(s)	Prediction(s)	Memory(MB)
GraphSAGE	66101	5.8	5.3	23015
GAT	34801	5.9	0.7	7230
JODIE	61101	5.4	2.8	4159
TGAT	65101	5.5	0.8	10198
TGN	221901	8.1	7.5	12039
DyHATR	443584	594.9	6.8	2294
HTGNN	82437	1.0	0.1	2592
STHN-Type	446000	27.2	0.4	13289
STHN-Trans	121901	14.1	0.3	2885
STHN-Patch	172001	6.3	0.3	1763

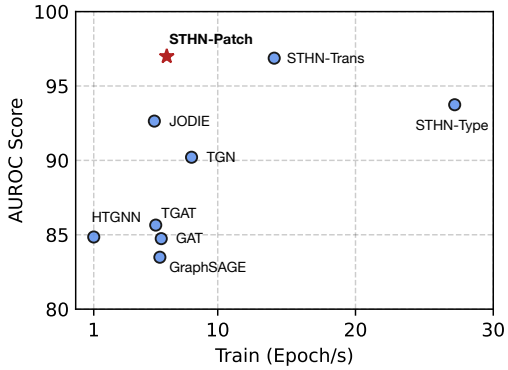

Figure 6: Model performance vs. training time consumption on MathOverflow for transductive link prediction task.

on minority classes could bring down the average precision and recall, hence lowering the AUPRC score. Comparing with Movielens, MathOverflow gains a better AUPRC score for link type prediction task, which can also be seen in Figure 4, showing that STHN gets a relatively higher accuracy for MathOverflow link type 2 and 3.

Hyperparameter investigation of patching. We systematically analyze the effect of hyperparameters used in STHN patching step, including the number of recent neighbors K and patch length N, as shown in Table 4. The experiments are conducted on three datasets with model STHN-Patch and STHN-Trans on AUROC metric, and results are summarized in Table 4. We search K within $\{50, 100, 200\}$ in all cases, and $\{5, 10\}$ for patch length N. Firstly, we observe that the performance of vanilla transformer tends to decline when the length of interaction sequence increasing, e.g., AUROC score of STHN-Trans on Netflix dataset will drop to 92.06 when K increases to 200. Transformer is designed to model global dependencies due to its fully connected attention mechanism. But, in practice, it struggles with long sequence and may get strong connection with one specific point, losing sight of the long-term dependencies. However, STHN-Patch divides the long sequence into patches, in which the information is gathered in patch-level. It helps Transformer encoder to capture comprehensive semantic information.

4.6 Efficiency Considerations

To compare the actual computational efficiency of model training, we collect model parameters, average epoch train/prediction time and GPU memory usage for the models involved in this study and show the statistics in Table 5. Because the model designs of homogeneous networks are similar, they get similar trainable parameters as well as training time. Since TGN is equipped with a memory updater to store the long-term information, the prediction time of TGN is obviously longer than other models. Comparing with

STHN-Type and STHN-Trans, STHN-Patch simplifies the architecture design by dropping the frequently used type-specific design and combining patching technique with Transformer information fusion. Due to simplicity, the GPU memory usage data shows that our STHN uses the least GPU memory and enjoys the fastest prediction speed among all temporal networks. One may observe that HTGNN exhibits fast Train and Prediction times, as it is a snapshot-based model. However, its Memory requirement is still higher than STHN-Patch – and, as mentioned earlier (cf. Tables 2 and 3) its effectiveness is lower than STHN-Patch.

To further compare the model performance in terms of training efficiency and effectiveness, in Figure 6 we show the average epoch training time and test AUROC score for temporal networks on MathOverflow dataset. As shown, STHN emerges as the best trade-off between model performance and training speed. We note that DyHATR fails to accomplish the link prediction task from the perspective of efficiency and effectiveness. Lastly, Figure 5 compares the convergence speed of STHN and the baseline models. Since we employ a 20-epoch early stopping regularization, the number of training epochs varies by model. As can be seen from the top row of Figure 5, STHN enjoys a fast convergence speed and achieves a low loss value at the very beginning of training process. The heterogeneous link encoder separates STHN from other temporal networks and enables it to extract meaningful information and converge faster on link prediction task.

5 RELATED WORK

Dynamic heterogeneous graph learning is a popular topic in the broader paradigm of graph learning and the results are useful for a plethora of downstream tasks such as evolving android malware detection [5], citation prediction [7, 13, 36], financial time series prediction [34], opioid overdose prediction [32], real-time event prediction [17] – to name a few. For a broader overview, we refer the readers to recent surveys on dynamic graph learning [25, 28, 38]. Depending on the updates manner, dynamic heterogeneous graph methods can be classified into *discrete* and *continuous* models (cf. [38]).

5.1 Discrete Dynamic Heterogeneous Network

Discrete-time dynamic heterogeneous graph learning was inspired by GNNs originally developed for static settings. In order to embed the temporal dependency between graph updates and capture evolutionary patterns of the graph structure, the most intuitive approach was to split the changing graphs into different snapshots and learn the node representation from a sequence of evolving heterogeneous graph snapshots. The methodologies used in discrete heterogeneous graph learning [2, 6, 7, 13, 24, 34] can be perceived as stemming from two perspectives: graph encoder and sequence model. Specifically, homogeneous GNNs (e.g., GCN [15] and GAT [27]) were first utilized as graph encoder to embed the graph structure information for each snapshot with a specific timestamp. Then the model obtains the node embedding by gathering node representations of specified timestamps with a sequence model, e.g., RNN or attention. However, discrete dynamic models suffer from two major shortcomings. First, the timestamps $\{1, \ldots, t\}$ of split snapshots are predefined and the prediction for whether a link will appear is only made at the specific $(t+1)$-th snapshot, which limits the model that it has to be retrained based on the data of timestamps $\{2, \ldots, t, t+1\}$ if we try to predict the $(t+2)$-th snapshot. Most of the discrete heterogeneous graph methods lack inductive learning capability and can only handle transductive tasks since they need re-training to infer embeddings for unseen nodes [35]. Second, type-specific aggregators are usually designed in discrete model to combine the information from links that belong to the same type. That means the number of aggregators is equal to the number of link types. The stacked aggregators bring high computational costs to heterogeneous graph learning [28] and these costs can quickly become prohibitively high for evolving graphs with many link types.

5.2 Continuous Dynamic Graph Learning.

Continuous-time graph learning (e.g., temporal networks [20]) approaches are usually confined to homogeneous settings, in which edges are labeled by time. Compared with discrete-time graph learning, in the continuous-time dynamic graph learning a temporal graph is constructed with newcoming nodes and edges which are added in a stream-like manner [37, 38]. Every interaction (edge) is annotated with a certain timestamp. CTDNE [20] is the first framework for learning time-dependent graph representations based on temporal random walk. JODIE [16] learns embedding trajectories of users and items with a coupled RNN. However, aforementioned models only generate embeddings for graphs at the final state, which is not scalable for inductive graph learning tasks. To addressed this, TGAT [35] and TGN [22] introduced temporal graph attention layer by mixing GraphSAGE [8] and GAT [27]. As mentioned, these kinds of temporal networks focused on modeling homogeneous graph, ignoring the cases of heterogeneity in nodes and/or links. We note that [32] incorporated continuous dynamics with heterogeneous graph learning targeting opioid overdose prediction and demonstrating strong performance on this task. Our STHN is more general in the sense of focusing on both existence and type of link prediction in transductive as well as inductive settings. The main distinction of STHN is how it simplifies the architecture design and improves the effectiveness and efficiency.

6 CONCLUSION

We presented STHN – a novel approach to incorporate both the heterogeneity of the nodes and links as well as the continuous granularity of the temporal dimension for the purpose of link prediction in evolving heterogeneous networks. We observed that assuming homogeneity of the types of nodes and links may be restrictive for many practical domains and discrete-time evolution models hinder the possibility of reasoning in continuous manner. The implicit incorporation of structural heterogeneity and temporal information makes STHN easy to generalize to different domains/settings. As part of our future work, we will investigate predicting not only the occurrence of a link but also its (minimal) duration.

ACKNOWLEDGMENTS

This work was supported in part by the NSF under Grant SWIFT 2030249 and Kingland Foundation; in part by the NSFC under Grant 62072077 and Grant 62176043; in part by the Natural Science Foundation of Sichuan Province, China, under Grant 2022NSFSC0505.

REFERENCES

[1] Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E Hinton. 2016. Layer normalization. *arXiv preprint arXiv:1607.06450* (2016).

[2] Ranran Bian, Yun Sing Koh, Gillian Dobbie, and Anna Divoli. 2019. Network Embedding and Change Modeling in Dynamic Heterogeneous Networks. In *SIGIR*. 861–864.

[3] Weilin Cong, Si Zhang, Jian Kang, Baichuan Yuan, Hao Wu, Xin Zhou, Hanghang Tong, and Mehrdad Mahdavi. 2023. Do We Really Need Complicated Model Architectures For Temporal Networks?. In *ICLR*.

[4] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. 2021. An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale. In *ICLR*.

[5] Yujie Fan, Mingxuan Ju, Shifu Hou, Yanfang Ye, Wenqiang Wan, Kui Wang, Yinming Mei, and Qi Xiong. 2021. Heterogeneous Temporal Graph Transformer: An Intelligent System for Evolving Android Malware Detection. In *KDD*. 2831–2839.

[6] Yujie Fan, Mingxuan Ju, Chuxu Zhang, and Yanfang Ye. 2022. Heterogeneous Temporal Graph Neural Network. In *SDM*. 657–665.

[7] Hao Geng, Deqing Wang, Fuzhen Zhuang, Xuehua Ming, Chenguang Du, Ting Jiang, Haolong Guo, and Rui Liu. 2022. Modeling Dynamic Heterogeneous Graph and Node Importance for Future Citation Prediction. In *CIKM*. 572–581.

[8] William L. Hamilton, Zhitao Ying, and Jure Leskovec. 2017. Inductive Representation Learning on Large Graphs. In *NeurIPS*. 1024–1034.

[9] F Maxwell Harper and Joseph A Konstan. 2015. The movielens datasets: History and context. *Acm transactions on interactive intelligent systems (tiis)* 5, 4 (2015), 1–19.

[10] John J Hopfield. 1982. Neural networks and physical systems with emergent collective computational abilities. *Proceedings of the national academy of sciences* 79, 8 (1982), 2554–2558.

[11] Ziniu Hu, Yuxiao Dong, Kuansan Wang, and Yizhou Sun. 2020. Heterogeneous Graph Transformer. In *WWW*. 2704–2710.

[12] Hong Huang, Ruize Shi, Wei Zhou, Xiao Wang, Hai Jin, and Xiaoming Fu. 2021. Temporal Heterogeneous Information Network Embedding. In *IJCAI*. 1470–1476.

[13] Song Jiang, Bernard Koch, and Yizhou Sun. 2021. HINTS: Citation Time Series Prediction for New Publications via Dynamic Heterogeneous Information Network Embedding. In *WWW*. 3158–3167.

[14] Mengyuan Jing, Yanmin Zhu, Yanan Xu, Haobing Liu, Tianzi Zang, Chunyang Wang, and Jiadi Yu. 2022. Learning Shared Representations for Recommendation with Dynamic Heterogeneous Graph Convolutional Networks. *ACM Transactions on Knowledge Discovery from Data* 17, 4 (2022), 1–23.

[15] Thomas N. Kipf and Max Welling. 2017. Semi-Supervised Classification with Graph Convolutional Networks. In *ICLR*.

[16] Srijan Kumar, Xikun Zhang, and Jure Leskovec. 2019. Predicting Dynamic Embedding Trajectory in Temporal Interaction Networks. In *KDD*. 1269–1278.

[17] Wenjuan Luo, Han Zhang, Xiaodi Yang, Lin Bo, Xiaoqing Yang, Zang Li, Xiaohu Qie, and Jieping Ye. 2020. Dynamic Heterogeneous Graph Neural Network for Real-time Event Prediction. In *KDD*. 3213–3223.

[18] Qingsong Lv, Ming Ding, Qiang Liu, Yuxiang Chen, Wenzheng Feng, Siming He, Chang Zhou, Jianguo Jiang, Yuxiao Dong, and Jie Tang. 2021. Are we really making much progress?: Revisiting, benchmarking and refining heterogeneous graph neural networks. In *KDD*. 1150–1160.

[19] Inc. Netflix. 2019. Netflix Prize data. https://www.kaggle.com/datasets/netflix-inc/netflix-prize-data

[20] Giang Hoang Nguyen, John Boaz Lee, Ryan A. Rossi, Nesreen K. Ahmed, Eunyee Koh, and Sungchul Kim. 2018. Continuous-Time Dynamic Network Embeddings. In *Companion of the The Web Conference*. 969–976.

[21] Ashwin Paranjape, Austin R Benson, and Jure Leskovec. 2017. Motifs in temporal networks. In *WWW*. 601–610.

[22] Emanuele Rossi, Ben Chamberlain, Fabrizio Frasca, Davide Eynard, Federico Monti, and Michael M. Bronstein. 2020. Temporal Graph Networks for Deep Learning on Dynamic Graphs. In *ICML 2020 Workshop on Graph Representation Learning*.

[23] Sina Sajadmanesh, Sogol Bazargani, Jiawei Zhang, and Hamid R. Rabiee. 2019. Continuous-Time Relationship Prediction in Dynamic Heterogeneous Information Networks. *ACM Trans. Knowl. Discov. Data* 13, 4 (2019), 44:1–44:31.

[24] Aravind Sankar, Yanhong Wu, Liang Gou, Wei Zhang, and Hao Yang. 2020. DySAT: Deep Neural Representation Learning on Dynamic Graphs via Self-Attention Networks. In *WSDM*. 519–527.

[25] Chuan Shi, Xiao Wang, and Philip S Yu. 2022. Dynamic Heterogeneous Graph Representation. In *Heterogeneous Graph Representation Learning and Applications*. 107–143.

[26] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is All you Need. In *NeurIPS*. 5998–6008.

[27] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. 2018. Graph Attention Networks. In *ICLR*.

[28] Xiao Wang, Deyu Bo, Chuan Shi, Shaohua Fan, Yanfang Ye, and S Yu Philip. 2022. A survey on heterogeneous graph embedding: methods, techniques, applications and sources. *IEEE Transactions on Big Data* 9, 2 (2022), 415–436.

[29] Xiao Wang, Houye Ji, Chuan Shi, Bai Wang, Yanfang Ye, Peng Cui, and Philip S Yu. 2019. Heterogeneous graph attention network. In *WWW*. 2022–2032.

[30] Xiao Wang, Yuanfu Lu, Chuan Shi, Ruijia Wang, Peng Cui, and Shuai Mou. 2020. Dynamic heterogeneous information network embedding with meta-path based proximity. *IEEE Transactions on Knowledge and Data Engineering* 34, 3 (2020), 1117–1132.

[31] Yanbang Wang, Yen-Yu Chang, Yunyu Liu, Jure Leskovec, and Pan Li. 2021. Inductive Representation Learning in Temporal Networks via Causal Anonymous Walks. In *ICLR*.

[32] Qianlong Wen, Zhongyu Ouyang, Jianfei Zhang, Yiyue Qian, Yanfang Ye, and Chuxu Zhang. 2022. Disentangled Dynamic Heterogeneous Graph Learning for Opioid Overdose Prediction. In *KDD*. 2009–2019.

[33] Zonghan Wu, Shirui Pan, Fengwen Chen, Guodong Long, Chengqi Zhang, and Philip S. Yu. 2019. A Comprehensive Survey on Graph Neural Networks. *CoRR* abs/1901.00596 (2019).

[34] Sheng Xiang, Dawei Cheng, Chencheng Shang, Ying Zhang, and Yuqi Liang. 2022. Temporal and Heterogeneous Graph Neural Network for Financial Time Series Prediction. In *CIKM*. 3584–3593.

[35] Da Xu, Chuanwei Ruan, Evren Körpeoglu, Sushant Kumar, and Kannan Achan. 2020. Inductive representation learning on temporal graphs. In *ICLR*.

[36] Xovee Xu, Ting Zhong, Ce Li, Goce Trajcevski, and Fan Zhou. 2022. Heterogeneous dynamical academic network for learning scientific impact propagation. *Knowledge-Based Systems* 238 (2022), 107839.

[37] Xovee Xu, Ting Zhong, Fan Zhou, Rongfan Li, Goce Trajcevski, and Qinggang Meng. 2023. Learning Spatiotemporal Manifold Representation for Probabilistic Land Deformation Prediction. *IEEE Transactions on Cybernetics* (2023).

[38] Guotong Xue, Ming Zhong, Jianxin Li, Jia Chen, Chengshuai Zhai, and Ruochen Kong. 2022. Dynamic network embedding survey. *Neurocomputing* 472 (2022), 212–223.

[39] Hansheng Xue, Luwei Yang, Wen Jiang, Yi Wei, Yi Hu, and Yu Lin. 2020. Modeling dynamic heterogeneous network for link prediction using hierarchical attention with temporal rnn. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*. 282–298.

[40] Xiaocheng Yang, Mingyu Yan, Shirui Pan, Xiaochun Ye, and Dongrui Fan. 2023. Simple and Efficient Heterogeneous Graph Neural Network. In *AAAI*, Vol. 37. 10816–10824.

[41] Mehmet Yildirimoglu, Isik Ilber Sirmatel, and Nikolas Geroliminis. 2018. Hierarchical control of heterogeneous large-scale urban road networks via path assignment and regional route guidance. *Transportation Research Part B: Methodological* 118 (2018), 106–123.

[42] Xiao Yu, Xiang Ren, Yizhou Sun, Quanquan Gu, Bradley Sturt, Urvashi Khandelwal, Brandon Norick, and Jiawei Han. 2014. Personalized entity recommendation: a heterogeneous information network approach. In *WSDM*. 283–292.

[43] Yutao Zhang, Jie Tang, Zhilin Yang, Jian Pei, and Philip S. Yu. 2015. COSNET: Connecting Heterogeneous Social Networks with Local and Global Consistency. In *KDD*. 1485–1494.

[44] Fan Zhou, Xovee Xu, Goce Trajcevski, and Kunpeng Zhang. 2021. A survey of information cascade analysis: Models, predictions, and recent advances. *Comput. Surveys* 54, 2 (2021), 1–36.

[45] Hongkuan Zhou, Da Zheng, Israt Nisa, Vasileios Ioannidis, Xiang Song, and George Karypis. 2022. TGL: A General Framework for Temporal GNN Training on Billion-Scale Graphs. *Proc. VLDB Endow.* 15, 8 (jun 2022), 1572–1580.