Discrete-time Temporal Network Embedding via Implicit Hierarchical Learning in Hyperbolic Space

Menglin Yang*
The Chinese University of Hong Kong
mlyang@cse.cuhk.edu.hk

Min Zhou
Noah's Ark Lab, Huawei Technologies
zhoumin27@huawei.com

Marcus Kalander
Noah's Ark Lab, Huawei Technologies
marcus.kalander@huawei.com

Zengfeng Huang
Fudan University
huangzf@fudan.edu.cn

Irwin King
The Chinese University of Hong Kong
king@cse.cuhk.edu.hk

ABSTRACT

Representation learning over temporal networks has drawn considerable attention in recent years. Efforts are mainly focused on modeling structural dependencies and temporal evolving regularities in Euclidean space which, however, underestimates the inherent complex and hierarchical properties in many real-world temporal networks, leading to sub-optimal embeddings. To explore these properties of a complex temporal network, we propose a hyperbolic temporal graph network (HTGN) that fully takes advantage of the exponential capacity and hierarchical awareness of hyperbolic geometry. More specially, HTGN maps the temporal graph into hyperbolic space, and incorporates hyperbolic graph neural network and hyperbolic gated recurrent neural network, to capture the evolving behaviors and implicitly preserve hierarchical information simultaneously. Furthermore, in the hyperbolic space, we propose two important modules that enable HTGN to successfully model temporal networks: (1) hyperbolic temporal contextual self-attention (HTA) module to attend to historical states and (2) hyperbolic temporal consistency (HTC) module to ensure stability and generalization. Experimental results on multiple real-world datasets demonstrate the superiority of HTGN for temporal graph embedding, as it consistently outperforms competing methods by significant margins in various temporal link prediction tasks. Specifically, HTGN achieves AUC improvement up to 9.98% for link prediction and 11.4% for new link prediction. Moreover, the ablation study further validates the representational ability of hyperbolic geometry and the effectiveness of the proposed HTA and HTC modules.

CCS CONCEPTS

• **Theory of computation** → **Dynamic graph algorithms**; • **Computing methodologies** → **Dimensionality reduction and manifold learning**.

KEYWORDS

Hyperbolic space; Temporal network; Graph neural network; Representation learning

ACM Reference Format:
Menglin Yang, Min Zhou, Marcus Kalander, Zengfeng Huang, and Irwin King. 2021. Discrete-time Temporal Network Embedding via Implicit Hierarchical Learning in Hyperbolic Space. In *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD '21), August 14–18, 2021, Virtual Event, Singapore.* ACM, New York, NY, USA, 11 pages. https://doi.org/10.1145/3447548.3467422

1 INTRODUCTION

Data describing the relationships between nodes of a graph are abundant in real-world applications, ranging from social networks analysis [25], traffic prediction [41], e-commerce recommendation [9], and protein structure prediction [11] to disease propagation analysis [6]. In many situations, networks are intrinsically changing or time-evolving with vertices (including their attributes) and edges appearing or disappearing over time. Learning representations of dynamic structures is challenging but of high importance since it describes how the network interacts and evolves, which will help to understand and predict the behavior of the system.

A number of temporal graph embedding methods have been proposed, which can be divided into two main categories: discrete-time network embeddings and continuous network embeddings [38]. Discrete-time network embeddings are represented in discrete time intervals denoted as multiple snapshots [31, 33]. As for continuous-time network embeddings, these can be described as time-dependent event-based models where the events, denoted by edges, occur over a time span [7, 28, 36]. Essentially, these two schemes both focus on capturing the underlying characteristics of a temporal graph: temporal dependency and topology evolution in Euclidean space. Euclidean space is the natural generalization of intuition-friendly and visualizable three-dimensional space with appealing vectorial structure, and closed-form formulas of distance and inner-product [13]. However, the quality of the representations is determined by how well the geometry of the embedding space matches the structure of the data [15], which triggers one basic question: whether the widely used Euclidean space is the best option for network embedding of an arbitrary temporal graph. Several works [5, 39] show that most of the graph data, e.g., social networks, communication networks, and disease-spreading networks exhibit non-Euclidean latent anatomies that show hierarchical structures and scale-free distributions as illustrated in Figure 1. This motivates us to rethink

*Work mainly done during an internship at Huawei Noah's Ark Lab.

KDD '21, August 14–18, 2021, Virtual Event, Singapore
© 2021 Association for Computing Machinery.
ACM ISBN 978-1-4503-8332-5/21/08…$15.00
https://doi.org/10.1145/3447548.3467422

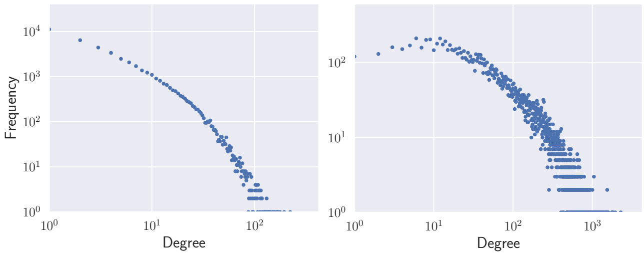

Figure 1: Degree distributions of a social network (FB, left) and a citation network (HepPh, right), which are asymptotically power-law distributed.

(1) whether the Euclidean manifold is the most suitable geometry for graph embedding of this kind of data and (2) is there any more powerful or proper alternative manifold to intrinsically preserve the graph properties, and if it exists, what benefits can it bring to temporal graph embedding?

Recently, hyperbolic geometry has received increasing attention and achieves state-of-the-art performance in several static graph embedding tasks [6, 24, 29, 30, 40]. One fundamental property of hyperbolic space is that it expands exponentially and can be regarded as a smooth version of trees, abstracting the hierarchical organization [22]. Therefore, the (approximate) hierarchical structure and tree-like data can naturally be represented by hyperbolic geometry, which instead will lead to severe structural inductive biases and high distortion if directly embedded into Euclidean space.

Despite the recent achievements in hyperbolic graph embedding, attempts on temporal networks are still scant. To fill this gap, in this work, we propose a novel hyperbolic temporal graph network (HTGN), which fully leverages the implicit hierarchical information to capture the spatial dependency and temporal regularities of evolving networks via a recurrent learning paradigm.

Following the concise and effective discrete-time temporal network embedding paradigm, a temporal network is first converted into a series of snapshots over time. In HTGN, we project the nodes into the hyperbolic space, leverage hyperbolic graph neural network (HGNN) to learn the topological dependencies of the nodes at each snapshot, and then employ hyperbolic gated recurrent unit (HGRU) to further capture the temporal dependencies. A temporal network is complex and may have cyclical patterns, and a distant snapshot may be more significant than the closest one [23, 33]. Recurrent neural networks (RNNs) [8, 18] usually restrict the model to emphasize the most nearby time-steps due to its time monotonic assumption. We, therefore, design a wrapped hyperbolic temporal contextual attention (HTA) module that incorporates context from the latest w historical states in hyperbolic space and assigns different weights for both distant and nearby snapshots. On the other hand, temporal coherence serves as a critical signal for sequential learning since a regular temporal graph is usually continuous and smoothly-varying. Inspired by the cycle-consistency in video tracking [10, 37], we propose a novel hyperbolic temporal consistency (HTC) component that imposes a constraint on the latent representations of consecutive snapshots, ensuring the stability and generalization for tracking the evolving behaviors. In summary, the main contributions are stated as follows:

- We propose a novel hyperbolic temporal graph embedding model, named HTGN, to learn temporal regularities, topological dependencies, and implicitly hierarchical organization. To the best of our knowledge, this is the first study on temporal graph embedding built upon a hyperbolic geometry powered by the recurrent learning paradigm.
- HTGN applies a flexible wrapped hyperbolic temporal contextual attention (HTA) module to effectively extract the diverse scope of historical information. A hyperbolic temporal consistency (HTC) constraint is further put forward to ensure the stability and generalization of the embeddings.
- Extensive experimental results on diverse real-world temporal graphs demonstrate the superiority of HTGN as it consistently outperforms the state-of-the-art methods on all the datasets by large margins. The ablation study further gives insights into how each proposed component contributes to the success of the model.

2 RELATED WORKS

Our work mainly relates to representation learning on temporal graph embedding and hyperbolic graph embedding.

Temporal graph embedding. Temporal graphs are mainly defined in two ways: (1) discrete temporal graphs, which are a collection of evolving graph snapshots at multiple discrete time steps; and (2) continuous temporal graphs, which update too frequently to be represented well by snapshots and are instead denoted as graph streams [35]. Snapshot-based methods can be applied to a timestamped graph by creating suitable snapshots, but the converse is infeasible in most situations due to a lack of fine-grained timestamps. Hence, we here mainly focus on representation learning over discrete temporal graphs. For systematic and comprehensive reviews, readers may refer to [35], and [1].

The set of approaches most relevant to our work is the recurrent learning scheme that integrates graph neural networks with the recurrent architecture, whereby the former captures graph information and the latter handles temporal dynamism by maintaining a hidden state to summarize historical snapshots. For instance, GCRN [34] offers two different architectures to capture the temporal and spatial correlations of a dynamic network. The first one is more straightforward and uses a GCN to obtain node embeddings, which are then fed into an LSTM to learn the temporal dynamism. The second is a modified LSTM that takes node features as input but replaces the fully-connected layers therein by graph convolutions. A similar idea is explored in DySAT [33], which instead computes node representations through joint self-attention along the two dimensions of the structural neighborhood and temporal dynamics. VRGNN [17] integrates GCRN with VGAE [20] and each node at each time-step is represented with a distribution; hence, the uncertainty of the latent node representations are also modeled. On the other hand, EvolveGCN [31] captures the dynamism of the graph sequence by using an RNN to evolve the GCN parameters rather than the temporal dynamics of the node embeddings. Most of the prevalent methods are built-in Euclidean space which, however, may underemphasize the intrinsic power-law distribution and hierarchical structure.

Hyperbolic graph embedding. Hyperbolic geometry has received increasing attention in machine learning and network science communities due to its attractive properties for modeling data with latent hierarchies. It has been applied to neural networks for problems of computer vision, natural language processing [16, 29, 30, 32], and graph embedding tasks [6, 16, 24, 40]. In the graph embedding field, recent works including HGNN [24], HGCN [6], and HGAT [40] generalize the graph convolution into hyperbolic space (the name of these methods are from corresponding literature) by moving the aggregation operation to the tangent space, where the vector operations can be performed. HGNN [24] focuses more on graph classification tasks and provides an extension to dynamic graph embeddings. HGAT [40] introduces a hyperbolic attention-based graph convolution using algebraic formalism in gyrovector and focuses on node classification and clustering tasks. HGCN [6] introduces a local aggregation scheme in local tangent space and develops a learnable curvature method for hyperbolic graph embedding. Besides, works in [15, 42] propose to learn representations over multiple geometries. The superior performance brought by hyperbolic geometry on static graphs motivates us to explore it on temporal graphs.

3 PRELIMINARY AND BACKGROUND

In this section, we first present the problem formulation of temporal graph embedding and introduce the widely used graph recurrent neural networks framework. Then, we introduce some fundamentals of hyperbolic geometry.

3.1 Problem Formulation

In this work, we focus on discrete-time temporal graph embedding. A discrete-time temporal graph [1, 35] is composed of a series of snapshots $\{G_1, ..., G_t, ..., G_T\}$ sampled from a temporal graph $\mathcal{G}$, where T is the number of snapshots. Each snapshot $G_t = (V_t, A_t)$ contains the current node set V_t and the corresponding adjacency matrix A_t. As time evolves, nodes may appear or disappear, and edges can be added or deleted. The graph embedding aims to learn a mapping function that obtains a low-dimensional representation H_t by giving the snapshots till timestamp t. A general learning framework can be written as:

$$H_t = f_2(f_1(A_t, X_t), H_{t-1}), \tag{1}$$

where X_t is the initial node features or attributes and H_{t-1} is the latest historical state. This learning paradigm is widely used in discrete-time temporal graph embedding [17, 34, 41] where f_1 is graph neural network, e.g., GCN [21] aiming at modeling structural dependencies and f_2 is a recurrent network, e.g., GRU [8] to capture the evolving regularities.

3.2 Hyperbolic Geometry

A Riemannian manifold $(\mathcal{M}, g)$ is a branch of differential geometry that involves a smooth manifold $\mathcal{M}$ with a Riemannian metric g. For each point $\mathbf{x}$ in $\mathcal{M}$, there is a tangent space $\mathcal{T}_\mathbf{x}\mathcal{M}$ as the first-order approximation of $\mathcal{M}$ around $\mathbf{x}$, which is a n-dimensional vector space (see Figure 2).

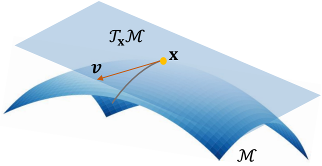

Figure 2: The tangent space $\mathcal{T}_\mathbf{x}\mathcal{M}$ and a tangent vector v, along the given point x of a curve traveling through the manifold $\mathcal{M}$.

There are multiple equivalent models for hyperbolic space. We here adopt the Poincaré ball model which is a compact representative providing visualizing and interpreting hyperbolic embeddings. The Poincaré ball model with negative curvature $-c$ $(c > 0)$ corresponds to the Riemannian manifold $(\mathbb{H}^{n,c}, g_\mathbb{H})$, where $\mathbb{H}^{n,c} = \left\{\mathbf{x} \in \mathbb{R}^n : c\|\mathbf{x}\|^2 < 1\right\}$ is an open n-dimensional ball. If $c = 0$, it degrades to Euclidean space, i.e., $\mathbb{H}^{n,c} = \mathbb{R}^n$. In addition, [13] shows how Euclidean and hyperbolic spaces can be continuously deformed into each other and provide a principled manner for basic operations (e.g., addition and multiplication) as well as essential functions (e.g., linear maps and softmax layer) in the context of neural networks and deep learning.

4 METHODOLOGY

The overall framework of the proposed Hyperbolic Temporal Graph Network (HTGN) is illustrated in Figure 3. As sketched in Figure 3(a), HTGN is a recurrent learning paradigm and falls into the prevalent discrete-time temporal graph architecture formulated by equation (1). An HTGN unit, shown in Figure 3(b), mainly consists of three components: (1) HGNN, the graph neural network to extract topological dependencies in hyperbolic space; (2) HTA module, an attention mechanism based on the hyperbolic proximity to obtain the attentive hidden state; (3) HGRU, the hyperbolic temporal recurrent module to capture the sequential patterns. Furthermore, we propose a hyperbolic temporal consistency constraint denoted as HTC to ensure stability and smoothness. We elaborate on the details of each respective module in the following paragraphs. For the sake of brevity, the timestamp t is omitted in Section 4.1 and Section 4.2.

4.1 Feature Map

Before going into the details of each module, we first introduce two bijection operations, the exponential map and the logarithmic map, for mapping between hyperbolic space and tangent space with a local reference point [6, 24], as presented below.

Proposition 1. *For* $\mathbf{x}' \in \mathbb{H}^{d,c}, \mathbf{a} \in \mathcal{T}_{x'}\mathbb{H}^{d,c}, \mathbf{b} \in \mathbb{H}^{d,c}$, *and* $\mathbf{a} \neq \mathbf{0}$, $\mathbf{b} \neq \mathbf{x}'$, *then the exponential map is formulated as:*

$$\exp_{\mathbf{x}'}^c(\mathbf{a}) = \mathbf{x}' \oplus^c \left(\tanh(\frac{\sqrt{c}\lambda_{\mathbf{x}'}^c \|\mathbf{a}\|}{2})\frac{\mathbf{a}}{\sqrt{c}\|\mathbf{a}\|}\right), \tag{2}$$

where $\lambda_{\mathbf{x}'}^c := \frac{2}{1-c\|\mathbf{x}'\|^2}$ *is conformal factor, and* $\oplus$ *is the Möbius addition, for any* $\mathbf{u}, \mathbf{v} \in \mathbb{H}^{d,c}$*:*

$$\mathbf{u} \oplus \mathbf{v} := \frac{\left(1 + 2c\langle\mathbf{u}, \mathbf{v}\rangle + c\|\mathbf{v}\|^2\right)\mathbf{u} + \left(1 - c\|\mathbf{u}\|^2\right)\mathbf{v}}{1 + 2c\langle\mathbf{u}, \mathbf{v}\rangle + c^2\|\mathbf{u}\|^2\|\mathbf{v}\|^2}. \tag{3}$$

The logarithmic map is given by:

$$\log_{\mathbf{x}'}^c(\mathbf{b}) := \frac{2}{\sqrt{c}\lambda_{\mathbf{x}'}^c}\operatorname{arctanh}(\sqrt{c}\|-\mathbf{x}' \oplus^c \mathbf{b}\|)\frac{-\mathbf{x}' \oplus^c \mathbf{b}}{\|-\mathbf{x}' \oplus^c \mathbf{b}\|}. \tag{4}$$

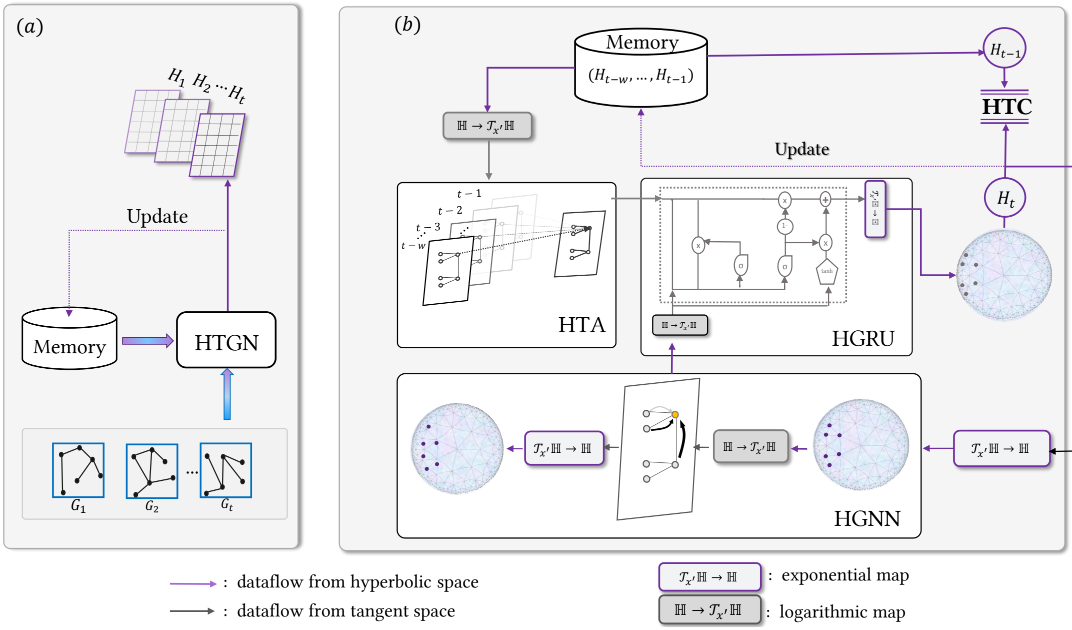
(a)
H_1 H_2 ⋯ H_t
Update
Memory
HTGN
G_1
G_2
⋯
G_t
(b)
Memory
$(H_{t-w}, \dots, H_{t-1})$
$\mathbb{H} \to \mathcal{T}_{x'}\mathbb{H}$
H_{t-1}
HTC
Output: H_t
Update
$t-1$
$t-2$
$t-3$
$t-w$
HTA
HGRU
H_t
Input:
$G_t(X_t, A_t)$
$\mathcal{T}_{x'}\mathbb{H} \to \mathbb{H}$
HGNN
: dataflow from hyperbolic space
: dataflow from tangent space
$\mathcal{T}_{x'}\mathbb{H} \to \mathbb{H}$: exponential map
$\mathbb{H} \to \mathcal{T}_{x'}\mathbb{H}$: logarithmic map

Figure 3: Schematic of HTGN. Figure (a) is a sketch of HTGN illustrating the recurrent paradigm and Figure (b) shows the data flow of an HTGN unit.

Note that $\mathbf{x}'$ is a local reference point, we use the origin point $\mathbf{0}$ in our work.

4.2 Hyperbolic Graph Neural Network (HGNN)

HGNN is employed to learn topological dependencies in the temporal graph leveraging promising properties of hyperbolic geometry. Analogous to GNN, an HGNN layer also includes three key operations: hyperbolic transformation, hyperbolic aggregation, and hyperbolic activation. Given a Euclidean space vector $\mathbf{x}_i^E \in \mathbb{R}^d$, we regard it as the point in the tangent space $\mathcal{T}_{\mathbf{x}'}\mathbb{H}^{d,c}$ with the reference point $\mathbf{x}' \in \mathbb{H}^{d,c}$ and use the exponential map to project it into hyperbolic space, obtaining $\mathbf{x}_i^{\mathcal{H}} \in \mathbb{H}^{d,c}$,

$$\mathbf{x}_i^{\mathcal{H}} = \exp_{\mathbf{x}'}^c(\mathbf{x}_i^E). \tag{5}$$

Then the update rule for one HGNN layer is expressed as:

$$\mathbf{m}_i^{\mathcal{H}} = W \otimes^c \mathbf{x}_i^{\mathcal{H}} \oplus^c \mathbf{b}, \tag{6a}$$

$$\tilde{\mathbf{m}}_i^{\mathcal{H}} = \exp_{\mathbf{x}'}^c\Big(\sum_{j \in \mathcal{N}(i)} \alpha_{ij} \log_{\mathbf{x}'}^c(\mathbf{m}_i^{\mathcal{H}})\Big), \tag{6b}$$

$$\tilde{\mathbf{x}}_i^{\mathcal{H}} = \exp_{\mathbf{x}'}^c(\sigma(\log_{\mathbf{x}'}^c(\tilde{\mathbf{m}}_i^{\mathcal{H}})). \tag{6c}$$

Hyperbolic Linear Transformation (6a). The hyperbolic linear transformation contains both vector multiplication and bias addition which can not be directly applied since the operations in hyperbolic space fail to meet the permutation invariant requirements. Therefore, for vector multiplication, we first project the hyperbolic vector to the tangent space and apply the operation, which is given by:

$$W \otimes^c \mathbf{x}_i^{\mathcal{H}} := \exp_{\mathbf{x}'}^c(W \log_{\mathbf{x}'}^c(\mathbf{x}_i^{\mathcal{H}})). \tag{7}$$

For bias addition, we transport a bias $\mathbf{b}$ located at $\mathcal{T}_{\mathbf{o}}\mathbb{H}$ to the position $\mathcal{T}_{\mathbf{x}}\mathbb{H}$ in parallel. Then, we use $\exp_{\mathbf{x}}^c$ to map it back to hyperbolic space:

$$\mathbf{x}^{\mathcal{H}} \oplus^c \mathbf{b} := \exp_{\mathbf{x}}^c(P_{\mathbf{o} \to \mathbf{x}}(\mathbf{b})). \tag{8}$$

Hyperbolic Aggregation (6b). The (weighted) mean operation is necessary to perform aggregation in an Euclidean graph neural network. An analog of mean aggregation in hyperbolic space is the Fréchet mean [12], however, it is difficult to apply as it lacks a closed-form to compute the derivative easily [2]. Similar to [6, 24, 40], we perform the aggregation computation in the tangent space. We adopt the attention-based aggregation, which is formulated as:

$$\begin{aligned} \alpha_{ij} &= softmax_{(j \in \mathcal{N}(i))}(s_{ij}) = \frac{\exp(s_{ij})}{\sum_{j' \in \mathcal{N}_i} \exp(s_{ij'})}, \\ s_{ij} &= \text{LeakReLU}(a^T [\log_0^c(m_i^l) \,\|\, \log_0^c(m_j^l)]), \end{aligned} \tag{9}$$

where a is a trainable vector, $\|$ denotes concatenation operation and s_{ij} indicates the correlation between the neighbors $j \in \mathcal{N}(i)$ and the center node i. Besides, we also consider the Laplacian based method [6], that is $\alpha_{ij} = \frac{1}{\sqrt{(d_i+1)(d_j+1)}}$ where d_i and d_j are the degree of node i and node j.

Hyperbolic Activation (6c). As given in equation (6c), the hyperbolic activation is achieved by applying logarithmic and exponential mapping. Noted that the two curvatures can be different.

4.3 Hyperbolic Temporal Attention (HTA)

Historical information plays an indispensable role in temporal graph modeling since it facilitates the model to learn the evolving patterns and regularities. Although the latest hidden state H_{t-1} obtained by the recurrent neural network already carries historical information before time t, some discriminative contents may still be under-emphasized due to the monotonic mechanism of RNNs that temporal dependencies are decreased along the time span [23].

Algorithm 1 HTA learning procedure

```
Input: $\{H_\tau^{\mathcal{H}}\}_{\tau=t-w}^{\tau=t-1}$
Output: $\tilde{H}_{t-1}$
for $\tau = t - w$ to $t - 1$ do
    $H_\tau^E = \log_{\mathbf{x}'}^c(H_\tau^{\mathcal{H}})$
end for
$M = [H_{t-w}^E \|, ..., \|H_{t-1}^E]$
$A_{tt} = r^T \tanh(QM)$
$\tilde{A}_{tt[i,:]} = softmax(A_{tt[i,:]})$
$H_{t-1}^E = \tilde{A}_{tt}M$
return $\tilde{H}_{t-1}^{\mathcal{H}} = \exp_{\mathbf{x}'}^c(H_{t-1}^E)$
```

Inspired by [9], our proposed HTA generalizes H_{t-1} to the latest w snapshots $\{H_{t-w}, \cdots, H_{t-1}\}$, attending on multiple historical latent states to get a more informative hidden state. The procedure is illustrated in Algorithm 1. Specifically, we first project w historical states in the state memory bank into the tangent space and concatenate them together. Then, the learnable weight matrix Q and vector $\mathbf{r}$ are utilized to extract contextual information, where Q weights the node importance in each historical state and $\mathbf{r}$ determines the weights across time windows.

4.4 Hyperbolic Gated Recurrent Unit (HGRU)

GRU [8], a variant of LSTM [18], is used in this work to incorporate the current and historical node states. Similar to LSTM, the GRU adopts gating units to modulate the flow of information but gets rid of the separate memory cell. Note that, our HGRU[1] is performed in the tangent space due to its computational efficiency.

HGRU receives the sequential input $\tilde{X}_t^{\mathcal{H}}$ from HGNN and the attentive hidden state $\tilde{H}_{t-1}^{\mathcal{H}}$ obtained from HTA as the input, and we denote $H_t^{\mathcal{H}}$ as the output. The dataflow in the HGRU unit is characterized by the following equations:

$$X_t^E = \log_{\mathbf{x}'}^c(\tilde{X}_t^{\mathcal{H}}), \tag{10a}$$

$$H_{t-1}^E = \log_{\mathbf{x}'}^c(\tilde{H}_{t-1}^{\mathcal{H}}), \tag{10b}$$

$$P_t^E = \sigma(W_z X_t^E + U_z H_{t-1}^E) \tag{10c}$$

$$R_t^E = \sigma(W_r X_t^E + U_r H_{t-1}^E), \tag{10d}$$

$$\tilde{H}_t^E = \tanh(W_h X_t^E + U_h(R_t \odot H_{t-1}^E)), \tag{10e}$$

$$H_t^E = (1 - P_t^E) \odot \tilde{H}_t^E + P_t^E \odot H_{t-1}^E, \tag{10f}$$

$$H_t^{\mathcal{H}} = \exp_{\mathbf{x}'}^c(H_t^E). \tag{10g}$$

where $W_z, W_r, W_h, U_z, U_r, U_h$ are the trainable weight matrices, P_t^E is the update gate to control the output and R_t^E is the reset gate to balance the input and memory. As the GRU is built in the tangent space, logarithmic maps are needed (equations (10a), (10b)). Then, we feed the states into the GRU (equations (10c) to (10f)) and map the hidden state back to hyperbolic space (equation (10g)). As we can see, the final $H_t^{\mathcal{H}}$ fuses structural, content, and temporal information.

Algorithm 2 The learning procedure of HTGN

```
Input: Node interaction stream $\{A_t\}_{t=1}^{t=T}$ and attributes $\{X_t^E\}_{t=1}^{t=T}$.
Output: $H_T^{\mathcal{H}}, c$.
 1: Initialize $w \times \{H_0^{\mathcal{H}}\}$ and curvature $c$
 2: repeat
 3:     for $t = 1$ to $T$ do
 4:         $X_t^{\mathcal{H}} = \exp_{\mathbf{x}'}^c(X_t^E)$
 5:         $\tilde{X}_t^{\mathcal{H}} = \mathbf{HGNN}(X_t^{\mathcal{H}})$
 6:         $\tilde{H}_{t-1}^{\mathcal{H}} = \mathbf{HTA}(H_{t-w}; ...; H_{t-1})$
 7:         $H_t^{\mathcal{H}} = \mathbf{HGRU}(\tilde{X}_t^{\mathcal{H}}, \tilde{H}_{t-1}^{\mathcal{H}})$
 8:         $\mathcal{L}_t = \mathcal{L}_{t,c} + \lambda \mathcal{L}_{t,r}$
 9:         Minimize $\mathcal{L}_t$
10:         Update state memory bank
11:     end for
12: until Convergence
13: return $H_T^{\mathcal{H}}, c$
```

4.5 Proposed Learning Algorithm

Uniting the above modules, we have the overall learning procedure as summarized in Algorithm 2. In line 5, we also consider include the historical state as the input as [17] and for brevity, we ignore it here. Note that we design the objective function $\mathcal{L}_t$ from two aspects: temporal evolution and topological learning, corresponding to the following hyperbolic temporal consistency loss and hyperbolic homophily loss.

4.5.1 Hyperbolic Temporal Consistency Loss. In terms of the time perspective, intuitively, the embedding position in the latent space changes gradually over time, which ensures stability and generalization. We thus pose a hyperbolic temporal consistency constraint $\mathcal{L}_{t,c}$ on two consecutive snapshots (G_t, G_{t-1}), to ensure the representation a certain temporal smoothness and long-term prediction ability, which is defined as:

$$\mathcal{L}_{t,c} = \frac{1}{N} \sum_{i=1}^{N} d^{\mathcal{H}}(\mathbf{x}_{t,i}^{\mathcal{H}}, \mathbf{x}_{(t-1),i}^{\mathcal{H}}), \tag{11}$$

where the subscript t denotes the loss is with respect to time snapshot t, and $d^{\mathcal{H}}$ is the distance of two points $\mathbf{u}, \mathbf{v} \in \mathbb{H}^{d'}$:

$$d^{\mathcal{H}}(\mathbf{u}, \mathbf{v}) = \frac{2\text{arctanh}\left(\sqrt{c}\|-\mathbf{u} \oplus^c \mathbf{v}\|\right)}{\sqrt{c}}. \tag{12}$$

4.5.2 Hyperbolic Homophily Loss. Graph homophily that linked nodes often belong to the same class or have similar attributes is a property shared by many real-world networks. The hyperbolic homophily loss $\mathcal{L}_{t,r}$ aims to maximize the probability of linked nodes through the hyperbolic feature and minimize the probability of no interconnected nodes. $\mathcal{L}_{t,r}$ is based on cross-entropy where the probability is inferred by the Fermi-Dirac function [6, 22] which is formulated as:

$$p_f(\mathbf{x}_i^{\mathcal{H}}, \mathbf{x}_j^{\mathcal{H}}) := [\exp((d^{\mathcal{H}}(\mathbf{x}_i^{\mathcal{H}}, \mathbf{x}_j^{\mathcal{H}}) - r)/s)]^{-1}, \tag{13}$$

where r is the radius of $\mathbf{x}_i^{\mathcal{H}}$, so points within that radius may have an edge with $\mathbf{x}_i^{\mathcal{H}}$, and s specifies the steepness of the logical function.

[1] A HyperGRU defined by Ganea et al. [13] is also applicable in our framework. However, we experimentally found that the proposed method built in the tangent space $\mathcal{T}_{\mathbf{x}'}\mathcal{H}$ obtains similar performance but is more efficient for large-scale data.

Table 1: Complexity analysis.

Components	Time Complexity
HTA	$O(Nwd')$
HTC	$O(Nd)$
HGNN	$O(Ndd' + d'\|\mathcal{E}_t\|)$

Then, $\mathcal{L}_{t,r}$ is given by:

$$\mathcal{L}_{t,r} = \frac{1}{E_1}\sum_{e_{ij}\in\mathcal{E}_t} -\log(p_f(\mathbf{x}^{\mathcal{H}}_{t,i}, \mathbf{x}^{\mathcal{H}}_{t,j})) - \frac{1}{E_2}\sum_{e_{i'j'}\notin\mathcal{E}_t}(1-\log(p_f(\mathbf{x}^{\mathcal{H}}_{t,i'}, \mathbf{x}^{\mathcal{H}}_{t,j'})). \tag{14}$$

To efficiently compute the loss and gradient, we sample the same number of negative edges as there are positive edges for each timestamp, i.e., $E_1 = E_2 = |\mathcal{E}_t|$.

4.5.3 The Unified Model. As temporal consistency and homophily regularity mutually drive the evolution of the temporal graphs, we set the final loss function as:

$$\mathcal{L}_t = \mathcal{L}_{t,r} + \lambda\mathcal{L}_{t,c}, \tag{15}$$

where $\lambda \in [0, 1]$ is the hyper-parameter to balance the temporal smoothness and homophily regularity.

Proposition 2. *Let N be the number of nodes, T be the number of training timestamps, $\mathcal{N}(i)$ be the neighbors of node i, and $|\mathcal{E}_t|$ be the number of links in timestamp t. Then, minimizing the loss $\mathcal{L} = \sum_{t=1}^{T}\mathcal{L}_t$ is equivalent to (1) minimizing the hyperbolic distance of a node with its current and historically connected nodes, and maximizing with the sampled negative neighbors, which are weighted by $\frac{1}{|\mathcal{E}_t|}$; (2) minimizing the distance between the same node over two consecutive timestamps, that is:*

$$\mathcal{L} = \sum_{i}^{N}\sum_{t=1}^{T}\Bigg(\sum_{j\in\mathcal{N}(i)}\frac{1}{|\mathcal{E}_t|}d^{\mathcal{H}}(\mathbf{x}^{\mathcal{H}}_{t,i}, \mathbf{x}^{\mathcal{H}}_{t,j}) - \sum_{j\notin\mathcal{N}(i)}\frac{1}{|\mathcal{E}_t|}d^{\mathcal{H}}(\mathbf{x}^{\mathcal{H}}_{t,i}, \mathbf{x}^{\mathcal{H}}_{t,j'}) + \frac{\lambda}{N}d^{\mathcal{H}}(\mathbf{x}^{\mathcal{H}}_{t,i}, \mathbf{x}^{\mathcal{H}}_{t-1,i})\Bigg). \tag{16}$$

Proof. See Appendix A.2.

Proposition 2 shows that our loss builds a message-passing connection within its neighbors from different times and local structures. In other words, if two nodes are not connected directly, they may still have implicit interactions through their common neighbors even at different times. This enables the representation to encode more patterns directly and indirectly, which is essential for high-quality representation and further prediction.

As we can see, the loss function $\mathcal{L}_t$ is only related to distance in the Poincaré ball, and thus scales well to large-scale datasets. Given two points $u, v \in \mathbb{H}^{d'}$, the gradient [29] of their distance in the Poincaré Ball is formulated as:

$$\Delta_u(d(u,v)) = \frac{4}{\beta\sqrt{\gamma^2 - 1}}\left(\frac{\|u\|^2 - 2 < u, v > +1}{\alpha^2}u - \frac{v}{\alpha}\right), \tag{17}$$

where $\alpha = 1 - \|u\|^2, \beta = 1 - \|v\|^2, \gamma = 1 + \frac{2}{\alpha\beta}\|u - v\|^2$. The computational and memory complexity of one backpropagation step depends linearly on the embedding dimension.

Table 2: Dataset statistics.

Datasets	DISEASE	HepPH	FB	AS733	Enron	COLAB
#Snapshots	7	36	36	30	11	10
#Test k	3	6	3	10	3	3
#Nodes	2665	15,330	45,435	6,628	184	315
#Total Edges	2664	976,097	180,011	13,512	790	943
Density ρ (0.01)*	0.41	1.37	0.04	0.2	3.37	0.94
Hyperbolicity $\delta^\star$	0.0	1.0	2.0	1.5	1.5	2.0

* 0.01 denotes the value is in units of 0.01.
⋆ The smaller δ indicates the dataset has a more evident hierarchical structure.

4.5.4 Complexity Analysis. We analyze the time complexity of the main components of the proposed HTGN model in each timestamp and present a summary in Table 1, where N and $|\mathcal{E}_t|$ are the number of nodes and edges, d and d' are respectively the dimensions of input feature and output feature, and w denotes state memory length. Note that the above modules can be paralleled across all nodes and are computationally efficient. Furthermore, as we use a constant memory state bank, the extra storage cost is negligible. Numerical analysis on the scalability is presented in Section 6.2.

5 EXPERIMENTS AND ANALYSIS

In this section, we conduct extensive experiments with the aim of answering the following research questions (**RQs**):

- **RQ1**. How does HTGN perform?
- **RQ2**. What does each component of HTGN bring?

5.1 Experimental Setup

5.1.1 Datasets. To verify the generality of our proposed method, we choose a diverse set of networks for evaluation, including disease-spreading networks, DISEASE; academic co-author networks, HepPh and COLAB; social networks, FB; email communication networks, Enron; and Internet router network, AS733, as recorded in Table 2. Notable, DISEASE is a synthetic dataset based on the SIR spreading model [3], which is also feasible for COVID-19 path prediction. At the same time, we list the Gromov's hyperbolicity δ [19, 27] and the average density ρ. Gromov's hyperbolicity is a notion from graph theory and measures the "tree-likeness" of metric spaces. The lower hyperbolicity, the more tree-like, with $\delta = 0$ denoting a pure tree. The average density is defined as the ratio of the number of edges and all possible edges, describing how dense a graph is. In these datasets, HepPh and COLAB are relatively dense and FB is highly sparse. More details about the datasets are presented in Appendix A.3.1.

5.1.2 Baselines. We compare the performance of our proposed model against a diverse set of competing graph embedding methods. The first two are advanced static network embedding models: GAE and VGAE[2]. **GAE** is simply composed of two-layer graph convolutions; **VGAE** additionally introduces variantial variables. Compared with the graph models tailored for temporal graphs, the static models ignore the temporal regularity. We use all the edges in the training shots for training and the remaining as the test set. We moreover compare with **GRUGCN**, conceptually the same version as in [34] and also a basic architecture of temporal graph embedding model, to show the effectiveness of Hyperbolic geometry and our proposed HTA module. More importantly, we also

[2] https://github.com/tkipf/gae

Table 3: AUC (left) and AP (right) scores of temporal link prediction on temporal network datasets.

	AUC						AP					
Dataset	**DISEASE**	**HepPh**	**FB**	**AS733**	**Enron**	**COLAB**	**DISEASE**	**HepPh**	**FB**	**AS733**	**Enron**	**COLAB**
GAE	72.55 ± 1.20	69.44 ± 0.56	63.07 ± 0.93	93.21 ± 1.53	92.50 ± 0.68	84.57 ± 0.64	60.55 ± 1.01	73.61 ± 0.58	65.35 ± 0.90	94.75 ± 0.90	*93.48 ± 0.64*	87.69 ± 0.44
VGAE	83.08 ± 1.27	72.39 ± 0.11	67.16 ± 0.53	*95.76 ± 0.91*	91.93 ± 0.34	85.16 ± 0.74	78.34 ± 1.25	75.78 ± 0.06	69.73 ± 0.17	96.42 ± 0.55	93.45 ± 0.49	88.70 ± 0.35
EvolveGCN	73.55 ± 4.23	76.82 ± 1.46	76.85 ± 0.85	92.47 ± 0.04	90.12 ± 0.69	83.88 ± 0.53	73.25 ± 3.44	81.18 ± 0.89	80.87 ± 0.64	95.28 ± 0.01	92.71 ± 0.34	87.53 ± 0.22
GRUGCN	79.25 ± 1.69	*82.86 ± 0.53*	*79.38 ± 1.02*	94.96 ± 0.35	92.47 ± 0.36	84.60 ± 0.92	65.26 ± 1.94	*85.87 ± 0.23*	*82.77 ± 0.75*	96.64 ± 0.22	93.38 ± 0.24	87.87 ± 0.58
DySAT	73.74 ± 2.28	81.02 ± 0.25	76.88 ± 0.08	95.06 ± 0.21	93.06 ± 0.97	*87.25 ± 1.70*	63.81 ± 1.86	84.47 ± 0.23	80.39 ± 0.14	*96.72 ± 0.12*	93.06 ± 1.05	*90.40 ± 1.47*
VGRNN	*86.44 ± 3.12*	77.65 ± 0.99	78.11 ± 1.11	95.17 ± 0.62	*93.10 ± 0.57*	85.95 ± 0.49	*82.00 ± 3.83*	80.95 ± 0.94	80.40 ± 0.74	96.69 ± 0.31	93.29 ± 0.69	87.77 ± 0.79
HTGN (Ours)	**89.65 ± 0.70**	**91.13 ± 0.14**	**83.70 ± 0.33**	**98.75 ± 0.03**	**94.17 ± 0.17**	**89.26 ± 0.17**	**84.63 ± 0.65**	**89.52 ± 0.28**	**83.80 ± 0.43**	**98.41 ± 0.03**	**94.31 ± 0.26**	**91.91 ± 0.07**
Gain (%)	+3.71	+9.98	+5.44	+3.12	+1.15	+2.30	+3.21	+4.25	+1.24	+1.75	+0.89	+1.67

Table 4: AUC (left) and AP (right) scores of temporal new link on temporal network datasets.

	AUC						AP					
Dataset	**DISEASE**⋆	**HepPh**	**FB**	**AS733**	**Enron**	**COLAB**	**DISEASE**	**HepPh**	**FB**	**AS733**	**Enron**	**COLAB**
EvolveGCN	73.55 ± 4.23	74.79 ± 1.61	74.49 ± 0.89	75.82 ± 0.67	82.85 ± 0.97	73.49 ± 0.86	73.25 ± 3.44	79.04 ± 1.02	78.33 ± 0.66	83.57 ± 0.46	85.01 ± 0.22	77.11 ± 0.44
GRUGCN	79.25 ± 1.69	*81.97 ± 0.49*	*77.69 ± 1.03*	*83.14 ± 1.21*	87.59 ± 0.57	75.60 ± 1.60	65.26 ± 1.94	*84.78 ± 0.22*	*81.07 ± 0.77*	88.14 ± 0.76	*88.41 ± 0.45*	78.55 ± 1.05
DySAT	73.74 ± 2.28	79.01 ± 0.26	74.97 ± 0.12	82.84 ± 0.72	87.94 ± 3.78	*79.74 ± 4.35*	63.81 ± 1.86	82.53 ± 0.25	78.34 ± 0.07	*89.07 ± 0.57*	86.83 ± 5.01	*83.47 ± 3.01*
VGRNN	*86.44 ± 3.12*	74.52 ± 1.05	75.31 ± 1.10	78.86 ± 3.39	*88.43 ± 0.75*	77.09 ± 0.23	*82.00 ± 3.83*	77.83 ± 0.93	77.61 ± 0.64	84.59 ± 1.90	87.57 ± 0.57	79.63 ± 0.94
HTGN (Ours)	**89.65 ± 0.70**	**90.11 ± 0.14**	**82.21 ± 0.41**	**96.62 ± 0.22**	**91.26 ± 0.27**	**81.74 ± 0.56**	**84.63 ± 0.65**	**88.18 ± 0.31**	**81.70 ± 0.46**	**95.52 ± 0.25**	**90.62 ± 0.34**	**84.06 ± 0.41**
Gain (%)	+3.71	+9.93	+5.82	+11.40	+3.20	+2.51	+3.21	+4.01	+0.78	+7.24	+2.5	+0.71

⋆ In the DISEASE dataset, all edges in the test set are new, so the results of new link prediction are equivalent to the results of link prediction.

conduct experiments on several state-of-art temporal graph embedding models: **EvolveGCN**[3] [31], **DySAT** [33], and **VGRNN** [17] to further demonstrate the superiority of the proposed HTGN.

5.1.3 Evaluation Tasks and Metrics. We obtain node representations from HTGN which can be applied to various downstream tasks. In temporal graph embedding, link prediction is widely used for evaluation, as the addition or removal of edges over time leads to the network evolution. Here, we use the Fermi-Dirac function defined in equation (13) to predict links between two nodes. Similar to VGRNN [17], we evaluate our proposed models on two different dynamic link prediction tasks: *temporal link prediction* and *temporal new link prediction*. More specifically, given partially observed snapshots of a temporal graph $\mathcal{G} = \{G_1, ..., G_t\}$, *dynamic link prediction* task is defined to predict the link in the next snapshots G_{t+1} or next multi-step snapshots and *dynamic new link prediction* task is to predict new links in G_{t+1} that are not in G_t.

Following the same setting as in VGRNN [17], we choose the last k snapshots as the test set and the rest of the snapshots as the training set. To thoroughly verify the effectiveness of the model, we select different lengths for testing and the corresponding k values are listed in Table 2. We test the models regarding their ability of correctly classifying true and false edges by computing average precision (AP) and area under the ROC curve (AUC) scores. We assume all known edges in the test snapshots as true and sample the same number of non-links as false. Note that we uniformly train both the baselines and HTGN by using early stopping based on the performance of the training set.

5.2 Experimental Results (RQ1)

The code of HTGN is publicly available here.[4] We repeat each experiment five times and report the average value with the standard deviation on the test sets in Table 3 and Table 4, where the best results are in bold and the second-best results are in italics for each dataset. It is observed that HTGN consistently and significantly outperforms the competing methods of both tasks across all six datasets, demonstrating the effectiveness of the proposed method. On the other hand, the runners-up go to the other temporal graph embedding methods, which confirms the importance of temporal regularity in temporal graph modeling. In the following, we discuss the results on link prediction and new link prediction, respectively.

5.2.1 Link Prediction. Table 3 shows the experiments on the link prediction task. In summary, HTGN outperforms the competing methods significantly considering both AUC and AP scores, which shows that our proposed models have better generalization ability to capture temporal trends. For instance, HTGN achieves an average gain of 4.28% in AUC compared to the best baseline. Predicting the evolution of very sparse graphs (e.g., FB) or long-term sequences (e.g., HepPh) indeed are hard tasks. Notably, our proposed HTGN obtains remarkable gains for these datasets and successfully pushes the performance to a new level.

It is worthwhile mentioning that all edges in the test set are new in the DISEASE dataset, which then requires the model's stronger inductive learning ability. Despite the difficulty, the proposed HTGN still outperforms the baselines by large margins and achieves notable results in both AUC and AP metrics.

5.2.2 New Link Prediction. New link prediction aims to predict the appearance of new links, which is more challenging. Note that static methods are not applicable to this task as the sequential order is omitted in the learning procedure of a static method, and GAE and VGAE are thus not evaluated. From Table 4, we are able to find similar observations to the link prediction task, demonstrating the superiority of the proposed HTGN. Specifically, we notice that the performance of each method drops by different degrees compared to the corresponding link prediction task, while our HTGN model produces more consistent results. For instance, the performance of the baselines degrades dramatically on AS733 (e.g., the second-best, GRUGCN drops from 94.64% to 83.14%), but our HTGN only declines about 2%, which shows our proposed HTGN strong inductive ability.

[3] There are two versions of the EvolveGCN: EvolveGCN-O and EvolveGCN-H. We test both and report the best result.

[4] https://github.com/marlin-codes/HTGN

Table 5: Ablation study.

Dataset	DISEASE	HepPh	FB	AS733	Enron	DBLP
δ	0.0	1.0	2.0	1.5	1.5	2.0
$\rho(0.01)$	0.41	1.37	0.04	0.2	3.37	0.94
w/o HTC	78.22 ± 0.26	76.30 ± 0.95	79.74 ± 0.34	95.07 ± 0.63	92.88 ± 0.24	85.16 ± 0.28
Gain (%)	-14.6	-19.44	-4.97	-3.87	-1.39	-4.81
w/o HTA	83.51 ± 1.71	90.87 ± 0.10	82.97 ± 0.44	98.00 ± 0.05	93.51 ± 0.19	88.60 ± 0.20
Gain (%)	-7.35	-0.29	-0.88	-0.77	-0.71	-0.74
w/o $\mathbb{H}$	73.88 ± 2.66	83.33 ± 0.47	80.53 ± 0.24	95.12 ± 0.36	92.28 ± 0.32	84.20 ± 0.79
Gain (%)	-21.35	-9.36	-3.94	-3.82	-2.05	-6.01

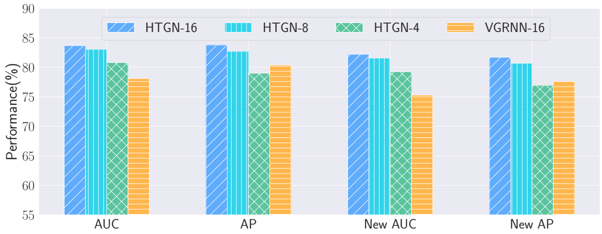

Figure 4: AUC scores of different embedding dimensions on FB.

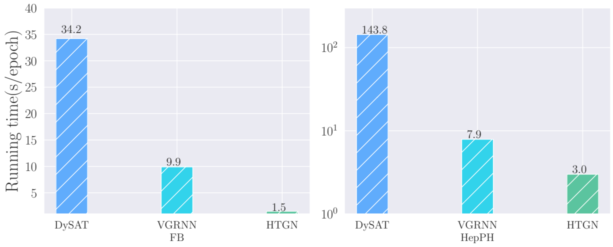

Figure 5: Running time on FB and HepPh.

5.3 Ablation Study (RQ2)

We further conduct an ablation study to validate the effectiveness of the main components of our proposed model. We name the HTGN variants as follows:

- **w/o HTC**: HTGN without the temporal attention module, i.e., the HGRU unit directly takes the hidden state of the last timestamp as the input.
- **w/o HTA**: HTGN without the hyperbolic temporal consistency, i.e., the model is trained by minimizing the hyperbolic homophily loss only.
- **w/o** $\mathbb{H}$: HTGN without hyperbolic geometry where all modules and learning processes are built-in Euclidean space. Correspondingly, the HTA and HTC modules are converted to Euclidean versions.

We repeat each experiment five times and report the average AUC on the test set for the link prediction task, as shown in Table 5. We first make the wrap-up observation that removing any of the components will cause performance degradation, which highlights the importance of each component. In the following, we take a closer look into the details about **w/o HTC** and **w/o HTA** at first. Discussion of **w/o** $\mathbb{H}$ is present in section 6.1.

Benefit from the HTC module. The effect of the temporal consistency constraint is significant as the performance drops vastly if the HTC module is removed. The model degradation is significant even for long-term prediction tasks (i.e., HepPh and AS733), which confirms that the HTC module facilitates the proposed model to capture the high-level temporal smoothness of an evolving network and ensures more stable and generalized prediction performance.

Benefit from the HTA module. As observed from Table 5, the performance decays by different extents if the HTA module is removed. In particular, the degradation is the most severe on the DISEASE dataset, which assembles node attributes. It confirms our assumption that HTA is able to collect the contextual information carried in the previous snapshots to further impel the learning of HTGN.

6 DISCUSSION

In this section, we further analyze HTGN with the aim of answering the following research questions:

- **RQ3**. What does hyperbolic geometry bring?
- **RQ4**. How is the learning efficiency in large networks?

6.1 Merits of Hyperbolic Geometry (RQ3)

Hierarchical awareness. We remove the hyperbolic geometry and build the learning process in Euclidean space. The HTA and HTC modules are converted to the corresponding Euclidean versions. As shown in Table 5 in the row for **w/o** $\mathbb{H}$, we know that the introduction of hyperbolic geometry significantly improves the performance. Particularly, for the pure tree-like DISEASE dataset, removing the hyperbolic projection will cause the AUC to degrade about 21.35%, and for another low-hyperbolicity dataset, HepPh,the deterioration is also significant with an AUC drop of about 9%. It verifies that hyperbolic geometry enables the preservation of the hierarchical layout in the graph data naturally and assists in producing high-quality node representations with smaller distortion.

Low-dimensional embedding. Benefiting from the exponential capacity of the hyperbolic space, we are permitted to use embeddings with lower dimensions to achieve notable performance. Taking the large dataset FB as an example, as shown in Figure 4, HTGN equipped with an 8-dimension embedding space still outperforms the runner-up, VGRNN. With 4-dimension embedding, HTGN is comparable with the 16-dimension VGRNN. This is another benefit that hyperbolic space can bring to the temporal graph network, i.e., reducing the embedding space and the corresponding learning parameters, which is valuable for embedding large-scale temporal graphs or deploying on low-memory/storage devices, e.g., mobile phones and UAV embedded units.

6.2 Running Time Comparison (RQ4)

In terms of efficiency, a theoretical analysis has been presented in Section 4.5.4. Here, we further numerically verify the scalability of HTGN by comparing the running time with the two second-best methods: VGRNN and DySAT. VGRNN is a GRNN-based method similar to HTGN, while DySAT uses attention to capture both the spatial and temporal dynamics. Figure 5 depicts the runtime per epoch of the three models on FB and HepPH, using a machine with GPU NVIDIA GeForce GTX TITAN X and 8 CPU cores. FB is a large-scale network with 45,435 nodes, and HepPH has fewer nodes 15,330 but more connection links.

As observed, HTGN achieves substantially lower training times, e.g., the running time per epoch of HTGN is 1.5 seconds compared to 9.9s for VGRNN and 34.2s for DySAT. The main reason is that HTGN deploys a shared HGNN before feeding into HGRU while VGRNN utilizes different GNNs[5]. On the other hand, DySAT requires computing both temporal attention and structural attention, which is computationally heavy. For the more dense HepPH network, the computing cost of DySAT which utilizes a pure self-attentional architecture increases dramatically, demonstrating the efficient recurrent learning paradigm in dynamic graph embedding.

7 CONCLUSION

In this work, we introduce a novel hyperbolic geometry-based node representation learning framework, denoted as hyperbolic temporal graph network, HTGN, for temporal network modeling. In general, HTGN follows the concise and effective GRNN framework but leverages the power of hyperbolic graph neural network and facilitates hierarchical arrangement to capture the topological dependency. More specifically, two novel modules: hyperbolic temporal contextual self-attention (HTA) and hyperbolic temporal consistency (HTC), respectively extract attentive historical states and ensuring stability and generalization, are proposed to impel the success of HTGN. When evaluated on multiple real-world temporal graphs, our approach outperforms the state-of-the-art temporal graph embedding baselines by a large margin. For future work, we will generalize our method to more challenging tasks and explore continuous-time learning to incorporate the fine-grained temporal variations.

ACKNOWLEDGEMENTS

This work is partially supported by the National Key Research and Development Program of China (No. 2018AAA0100204) and CUHK 3133238, Research Sustainability of Major RGC Funding Schemes (RSFS). We would like to thank the anonymous reviewers for their constructive comments.

REFERENCES

[1] Charu Aggarwal and Karthik Subbian. 2014. Evolutionary network analysis: A survey. *ACM Computing Surveys (CSUR)* 47, 1 (2014), 1–36.

[2] Miroslav Bacák. 2014. Computing medians and means in Hadamard spaces. *SIAM Journal on Optimization* 24, 3 (2014), 1542–1566.

[3] Ottar N Bjørnstad, Bärbel F Finkenstädt, and Bryan T Grenfell. 2002. Dynamics of measles epidemics: estimating scaling of transmission rates using a time series SIR model. *Ecological monographs* 72, 2 (2002), 169–184.

[4] Martin R Bridson and André Haefliger. 2013. *Metric spaces of non-positive curvature*. Vol. 319. Springer Science & Business Media.

[5] Michael M Bronstein, Joan Bruna, Yann LeCun, Arthur Szlam, and Pierre Vandergheynst. 2017. Geometric deep learning: going beyond euclidean data. *IEEE Signal Processing Magazine* 34, 4 (2017), 18–42.

[6] Ines Chami, Zhitao Ying, Christopher Ré, and Jure Leskovec. 2019. Hyperbolic graph convolutional neural networks. In *NeurIPS*. 4868–4879.

[7] Xiaofu Chang, Xuqin Liu, Jianfeng Wen, Shuang Li, Yanming Fang, Le Song, and Yuan Qi. 2020. Continuous-Time Dynamic Graph Learning via Neural Interaction Processes. In *CIKM*. 145–154.

[8] Kyunghyun Cho, Bart Van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using RNN encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078* (2014).

[9] Qiang Cui, Shu Wu, Yan Huang, and Liang Wang. 2019. A hierarchical contextual attention-based network for sequential recommendation. *Neurocomputing* 358 (2019), 141–149.

[10] Debidatta Dwibedi, Yusuf Aytar, Jonathan Tompson, Pierre Sermanet, and Andrew Zisserman. 2019. Temporal cycle-consistency learning. In *CVPR*. 1801–1810.

[11] Alex Fout, Jonathon Byrd, Basir Shariat, and Asa Ben-Hur. 2017. Protein interface prediction using graph convolutional networks. In *NeurIPS*. 6530–6539.

[12] Maurice Fréchet. 1948. Les éléments aléatoires de nature quelconque dans un espace distancié. In *Annales de l'institut Henri Poincaré*, Vol. 10. 215–310.

[13] Octavian Ganea, Gary Bécigneul, and Thomas Hofmann. 2018. Hyperbolic neural networks. In *NeurIPS*. 5345–5355.

[14] Xavier Glorot and Yoshua Bengio. 2010. Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the thirteenth international conference on artificial intelligence and statistics*. JMLR Workshop and Conference Proceedings, 249–256.

[15] Albert Gu, Frederic Sala, Beliz Gunel, and Christopher Ré. 2019. Learning mixed-curvature representations in product spaces. In *ICLR*.

[16] Caglar Gulcehre, Misha Denil, Mateusz Malinowski, Ali Razavi, Razvan Pascanu, Karl Moritz Hermann, Peter Battaglia, Victor Bapst, David Raposo, Adam Santoro, et al. 2019. Hyperbolic attention networks. In *ICLR*.

[17] Ehsan Hajiramezanali, Arman Hasanzadeh, Krishna Narayanan, Nick Duffield, Mingyuan Zhou, and Xiaoning Qian. 2019. Variational graph recurrent neural networks. In *NeurIPS*. 10701–10711.

[18] Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation* 9, 8 (1997), 1735–1780.

[19] Edmond Jonckheere, Poonsuk Lohsoonthorn, and Francis Bonahon. 2008. Scaled Gromov hyperbolic graphs. *Journal of Graph Theory* 57, 2 (2008), 157–180.

[20] Thomas N Kipf and Max Welling. 2016. Variational graph auto-encoders. *Bayesian Deep Learning Workshop (NIPS 2016)* (2016).

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

[22] Dmitri Krioukov, Fragkiskos Papadopoulos, Maksim Kitsak, Amin Vahdat, and Marián Boguná. 2010. Hyperbolic geometry of complex networks. *Physical Review E* 82, 3 (2010), 036106.

[23] Jun Liu, Gang Wang, Ping Hu, Ling-Yu Duan, and Alex C Kot. 2017. Global context-aware attention lstm networks for 3d action recognition. In *CVPR*. 1647–1656.

[24] Qi Liu, Maximilian Nickel, and Douwe Kiela. 2019. Hyperbolic graph neural networks. In *NeurIPS*. 8230–8241.

[25] Yozen Liu, Xiaolin Shi, Lucas Pierce, and Xiang Ren. 2019. Characterizing and forecasting user engagement with in-app action graph: A case study of snapchat. In *KDD*. 2023–2031.

[26] Daniel W Lozier. 2003. NIST digital library of mathematical functions. *Annals of Mathematics and Artificial Intelligence* 38, 1 (2003), 105–119.

[27] Onuttom Narayan and Iraj Saniee. 2011. Large-scale curvature of networks. *Physical Review E* 84, 6 (2011), 066108.

[28] Giang Hoang Nguyen, John Boaz Lee, Ryan A Rossi, Nesreen K Ahmed, Eunyee Koh, and Sungchul Kim. 2018. Continuous-time dynamic network embeddings. In *WWW*. 969–976.

[29] Maximillian Nickel and Douwe Kiela. 2017. Poincaré embeddings for learning hierarchical representations. In *NeurIPS*. 6338–6347.

[30] Maximillian Nickel and Douwe Kiela. 2018. Learning Continuous Hierarchies in the Lorentz Model of Hyperbolic Geometry. In *ICML*. 3779–3788.

[31] Aldo Pareja, Giacomo Domeniconi, Jie Chen, Tengfei Ma, Toyotaro Suzumura, Hiroki Kanezashi, Tim Kaler, Tao B Schardl, and Charles E Leiserson. 2020. EvolveGCN: Evolving Graph Convolutional Networks for Dynamic Graphs. In *AAAI*. 5363–5370.

[32] Frederic Sala, Chris De Sa, Albert Gu, and Christopher Re. 2018. Representation Tradeoffs for Hyperbolic Embeddings. In *ICML*. 4460–4469.

[33] 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.

[34] Youngjoo Seo, Michaël Defferrard, Pierre Vandergheynst, and Xavier Bresson. 2018. Structured sequence modeling with graph convolutional recurrent networks. In *ICONIP*. Springer, 362–373.

[35] Joakim Skarding, Bogdan Gabrys, and Katarzyna Musial. 2020. Foundations and modelling of dynamic networks using Dynamic Graph Neural Networks: A survey. *arXiv preprint arXiv:2005.07496* (2020).

[36] Rakshit Trivedi, Mehrdad Farajtabar, Prasenjeet Biswal, and Hongyuan Zha. 2019. Dyrep: Learning representations over dynamic graphs. In *ICLR*.

[37] Xiaolong Wang, Allan Jabri, and Alexei A Efros. 2019. Learning correspondence from the cycle-consistency of time. In *CVPR*. 2566–2576.

[38] Menglin Yang, Ziqiao Meng, and Irwin King. 2020. FeatureNorm: L2 Feature Normalization for Dynamic Graph Embedding. In *2020 IEEE International Conference on Data Mining (ICDM)*. 731–740. https://doi.org/10.1109/ICDM50108.2020.00082

[39] Zhitao Ying, Jiaxuan You, Christopher Morris, Xiang Ren, Will Hamilton, and Jure Leskovec. 2018. Hierarchical graph representation learning with differentiable pooling. In *NeurIPS*. 4800–4810.

[40] Yiding Zhang, Xiao Wang, Xunqiang Jiang, Chuan Shi, and Yanfang Ye. 2019. Hyperbolic graph attention network. In *AAAI*.

[41] Ling Zhao, Yujiao Song, Chao Zhang, Yu Liu, Pu Wang, Tao Lin, Min Deng, and Haifeng Li. 2019. T-gcn: A temporal graph convolutional network for traffic prediction. *IEEE Transactions on Intelligent Transportation Systems* (2019).

[42] Shichao Zhu, Shirui Pan, Chuan Zhou, Jia Wu, Yanan Cao, and Bin Wang. 2020. Graph Geometry Interaction Learning. In *NeurIPS*, Vol. 33.

[5] https://github.com/VGraphRNN/VGRNN/blob/master/VGRNN_prediction.py

A APPENDIX

A.1 Geometry Initiations of Hyperbolic Space

Riemannian manifolds with different curvatures define different geometries, where curvature measures how much a geometric object deviates from a flat plane, or in the case of a curve, deviates from a straight line. Different from the well-known Euclidean geometry which has zero curvature, hyperbolic space is a type of manifold with constant negative curvature and thus shows distinguishing properties.

There exist multiple equivalent models for hyperbolic space. The most commonly used in the machine learning community are the Poincaré (disk) model and the Lorentz (hyperboloid) model. The Lorentz model is well-suited for Riemannian optimization and the Poincaré disk provides a very intuitive way for visualizing and interpreting hyperbolic embeddings. We here take Poincaré disk to illustrate some intuitions of hyperbolic space.

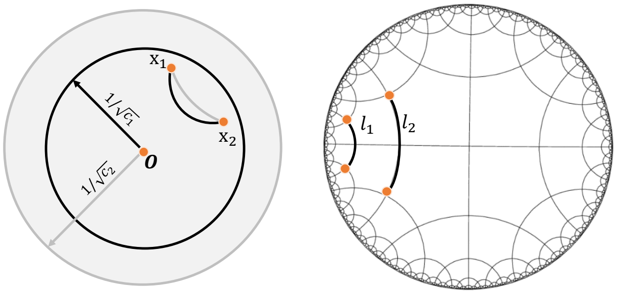

Figure 6: Left: Visualizations of Poincaré disks with different curvatures $-c_1(c_1 > 0)$ (area within the black circle and the corresponding radius is $1/\sqrt{c_1}$), $-c_2$ (area within the grey circle and the corresponding radius is $1/\sqrt{c_2}(c_2 > 0)$), where $\mathbf{x}_1$ and $\mathbf{x}_2$ are two points on Poincaré disks. Right: Two lines l_1 and l_2 with the same length in Poincaré disk.

Figure 6 gives some visualizations of the geometry properties on the Poincaré disk. As observed on the left, the geodesic length of two points in different Poincaré disks is different and related to its curvature. When the radius $1/\sqrt{c}$ decreases (i.e., space bends more or the absolute value of curvature increases), the distance between two given points will increase, and the line is closer to the origin. From the right, l_1 and l_2 are two lines with the same length though, l_1 is shorter from our Euclidean view. Then when two lines are with the same "Euclidean" length, the one closer to the border is actually longer in hyperbolic space and can accommodate more. We further give some mathematical expressions to illustrate the exponentially increased capacity of hyperbolic space.

According to [26], the n-dimensional volume of a Euclidean ball of radius r is:

$$V_n^{\mathbb{E}}(r) = \frac{\pi^{\frac{n}{2}}}{\Gamma\left(\frac{n}{2}+1\right)} r^n. \tag{18}$$

While the n-dimensional volume of a hyperbolic space of radius r in n-dimensional hyperbolic space, referring to [4], is given as:

$$V_n^{\mathbb{H}}(r) = V\left(\mathbb{S}^{n-1}\right)\int_0^r \sinh^{n-1} t dt, \tag{19}$$

where $V\left(\mathbb{S}^{n-1}\right)$ is the volume of the tangent space centered on the origin of the Poincaré model of radius r. In the 2-dimensional case, we then have the explicit expression $V_n^{\mathbb{H}}(r) = 4\pi\sinh^2(\frac{r}{2})$. For all $x \in \mathbb{H}^n$, we have

$$V_n^{\mathbb{H}}(r) \sim \frac{\mathrm{Vol}\left(\mathbb{S}^n\right)}{2^{n-1}} e^{(n-1)r}, as \quad r \to \infty. \tag{20}$$

As concluded from equations (18) and (20), the volume of a ball in hyperbolic space grows exponentially with the radius, while the counterpart in Euclidean spaces expands polynomially. Meanwhile, the nodes of a tree also grows exponentially with the depth (e.g., a perfect binary tree with depth n has $2^{n+1}-1$ nodes). A hyperbolic space can thus be regarded as a continuous analogous of trees and can be applied to naturally model data with hierarchical structures or tree-like layout.

A.2 Proof of Proposition 2

Proof. First, recall the equation of loss function $\mathcal{L}_t$:

$$\begin{aligned}\mathcal{L}_t &= \mathcal{L}_{t,r} + \lambda\mathcal{L}_{t,c}\\ &= \frac{1}{E_1}\sum_{e_{ij}\in\mathcal{E}} -\log(p_f(\mathbf{x}_i^{\mathcal{H}},\mathbf{x}_j^{\mathcal{H}})) - \frac{1}{E_2}\sum_{e_{i'j'}\notin\mathcal{E}}(1-\log(p_f(\mathbf{x}_{i'}^{\mathcal{H}},\mathbf{x}_{j'}^{\mathcal{H}})))\\ &\qquad + \lambda\cdot\frac{1}{N}\sum_{i=1}^{N} d^{\mathcal{H}}(\mathbf{x}_{t,i}^{\mathcal{H}},\mathbf{x}_{(t-1),i}^{\mathcal{H}}).\end{aligned} \tag{21}$$

The first term can be arranged as:

$$\begin{aligned}&\frac{1}{E_1}\sum_{e_{ij}\in\mathcal{E}} -\log(p_f(\mathbf{x}_i^{\mathcal{H}},\mathbf{x}_j^{\mathcal{H}}))\\ &= \frac{1}{E_1}\sum_{e_{ij}\in\mathcal{E}} -\log([\exp((d^{\mathcal{H}}(\mathbf{x}_i^{\mathcal{H}},\mathbf{x}_j^{\mathcal{H}})-r)/s)]^{-1})\\ &= \frac{1}{E_1}\sum_{e_{ij}\in\mathcal{E}} d^{\mathcal{H}}(\mathbf{x}_i^{\mathcal{H}},\mathbf{x}_j^{\mathcal{H}})-r)/s.\end{aligned} \tag{22}$$

Similarly, the second term can be arranged as:

$$\begin{aligned}&\frac{1}{E_2}\sum_{e_{i'j'}\notin\mathcal{E}} 1-\log(p_f(\mathbf{x}_{i'}^{\mathcal{H}},\mathbf{x}_{j'}^{\mathcal{H}}))\\ &= \frac{1}{E_2}\sum_{e_{i'j'}\notin\mathcal{E}} 1-\log([\exp((d^{\mathcal{H}}(\mathbf{x}_{i'}^{\mathcal{H}},\mathbf{x}_{j'}^{\mathcal{H}})-r)/s)]^{-1})\\ &= \frac{1}{E_2}\sum_{e_{i'j'}\notin\mathcal{E}} 1+(d^{\mathcal{H}}(\mathbf{x}_{i'}^{\mathcal{H}},\mathbf{x}_{j'}^{\mathcal{H}})-r)/s).\end{aligned} \tag{23}$$

Then, we have:

$$\begin{aligned}\mathcal{L}_t = \epsilon_0 + \epsilon_1\sum_{i}^{n} d^{\mathcal{H}}(\mathbf{x}_{(t,i)}^{\mathcal{H}},\mathbf{x}_{(t-1,i)}^{\mathcal{H}}) + \epsilon_2\sum_{e_{ij}\in\mathcal{E}} d^{\mathcal{H}}(\mathbf{x}_i^{\mathcal{H}},\mathbf{x}_j^{\mathcal{H}})\\ - \epsilon_3\sum_{e_{i'j'}\notin\mathcal{E}} d^{\mathcal{H}}(\mathbf{x}_{i'}^{\mathcal{H}},\mathbf{x}_{j'}^{\mathcal{H}}),\end{aligned} \tag{24}$$

where $\epsilon_0 = -1, \epsilon_1 = \frac{\lambda}{|V|}, \epsilon_2 = \epsilon_3 = \frac{1}{s|\mathcal{E}_t|}$. Note that we sample the same number of negative edges as there are positive ones, hence, $E_1 = E_2 = |\mathcal{E}_t|$. In our experiments, s is set to 1.0. Adding up the loss of all timestamps, we have:

$$
\begin{aligned}
\mathcal{L} &= \sum_{t=1}^{T} \mathcal{L}_t \\
&= -T + \sum_{t=1}^{T}\sum_{i}^{N} \frac{\lambda}{|V|} d^{\mathcal{H}}(\mathbf{x}^{\mathcal{H}}_{(t,i)}, \mathbf{x}^{\mathcal{H}}_{(t-1,i)}) + \sum_{t=1}^{T}\sum_{e_{ij}\in\mathcal{E}} \frac{1}{|\mathcal{E}_t|} d^{\mathcal{H}}(\mathbf{x}^{\mathcal{H}}_{i}, \mathbf{x}^{\mathcal{H}}_{j}) \\
&\qquad - \sum_{t=1}^{T}\sum_{e_{i'j'}\notin\mathcal{E}} \frac{1}{|\mathcal{E}_t|} d^{\mathcal{H}}(\mathbf{x}^{\mathcal{H}}_{i}, \mathbf{x}^{\mathcal{H}}_{j})
\end{aligned}
\tag{25}
$$

Next, we center the above loss to each node. For each node, the constraint comes from two aspects: (1) temporal homophily loss, which minimizes the hyperbolic distance between the node and its positive neighbors in all timestamps, and maximizes the distance between the node and the sampled negative neighbors, where the weights are determined by $\frac{1}{|\mathcal{E}_t|}$; (2) consistency constraint between the same node over two consecutive timestamps, that is:

$$
\begin{aligned}
\mathcal{L} = \sum_{i}^{N}\sum_{t=1}^{T} \Bigg(&\sum_{j\in\mathcal{N}(i)} \frac{1}{|\mathcal{E}_t|} d^{\mathcal{H}}(\mathbf{x}^{\mathcal{H}}_{i}, \mathbf{x}^{\mathcal{H}}_{j}) - \sum_{j\notin\mathcal{N}(i)} \frac{1}{|\mathcal{E}_t|} (\mathbf{x}^{\mathcal{H}}_{i}, \mathbf{x}^{\mathcal{H}}_{j'}) \\
&+ \frac{\lambda}{|V|} d^{\mathcal{H}}(\mathbf{x}^{\mathcal{H}}_{(t,i)}, \mathbf{x}^{\mathcal{H}}_{(t-1,i)}) \Bigg).
\end{aligned}
\tag{26}
$$

□

A.3 Experiment details

A.3.1 Data processing. Most of the data is in a timestamp format and we process it according to the physical meaning of the real world. The details are as follows:

Enron[6] is constructed from emails exchanged by 184 Ernon employees. The nodes represent the employees and the edges indicate the email interactions between them. We follow the same processing procedure as [17] to obtain 10 snapshots. The network does not contain any node and edge information.

COLAB[7] is an academic cooperation network, including the academic cooperation of 315 researchers from 2000 to 2009. Each node on the graph represents an author, and an edge denotes a co-authorship relation. We split the dataset by year following [17] and obtain 10 snapshots.

FB[8] is a social network graph of Facebook Wall posts where each node is a user and each edge is the interaction related to their wall posts. We take the activates over the last three years in the dataset as 36 snapshots. The FB dataset is associated with a large number of users but very sparse connections.

HepPh[9] is a citation network related to high energy physics phenomenology, which is collected from the e-print arXiv website. Each node represents a paper, and an edge represents one paper citing another. The data covers papers in the period between January 1993 to April 2003 (124 months in total). It is a directed graph network, but we learn and predict as if it was an undirected graph. According to the real physical meaning, we use three months of data per snapshot and use the last 36 months as the full dataset in our work.

AS733[10] is an Internet router network, which is collected from the University of Oregon Route Views Project. This dataset contains 733 instances and spans the time from November 8, 1997, to January 2, 2000, with an interval of 785 days. We split the snapshots per day and select the last 30 snapshots to use in this work. It is worth noting that this network is different from the citation networks where the nodes and edges only increase over time (i.e., no deleted edges or nodes), the AS733 data set also contains the removal of nodes and edges over time.

DISEASE[11] is a synthetic dataset based on the SIR disease spreading model [3], which also feasible for the COVID-19 path, where each node represents a person, the node feature describes the symptom of a person, and the edges indicate the spreading relationship. We split the dataset by the time they appear, and there are a large number of unobserved nodes in the test set.

A.3.2 Parameter settings. Note that most of the benchmark datasets for dynamic graph embedding are only associated with topology. Enron and COLAB are associated with a small number of nodes, we use identity matrix as the node feature which is identical with the processing in [17]. For the other datasets, i.e., HepPh, FB, and AS733, their node features are initialized by a dense vector with 128 dimensions using glorot's method [14] and are set as trainable matrices. The DISEASE is associate with node feature and we directly use it in our work. We set the final embeddings dimension of all models as 16, and it needs to be clear that different embedding sizes can lead to different results, but our method can always achieve impressive results.

We set the number of GRU layers as 1 for all models if there is a recurrent unit (e.g., RNN, GRU, LSTM, HGRU) for a fair comparison. In HTGN, we set the number of historical windows in HTA as 4 for DISEASE and 5 for the other datasets. We did not do any heavy parameter tuning since our main work is to verify whether HTA is an effective using module in HTGN. The hyper-parameters of r and t in the Fermi-Dirac function are set as 2.0 and 1.0 which is a common choice as [6].

[6] https://www.cs.cornell.edu/~arb/data/email-Enron/

[7] https://github.com/VGraphRNN/VGRNN/tree/master/data

[8] http://networkrepository.com/ia-facebook-wall-wosn-dir.php

[9] https://snap.stanford.edu/data/cit-HepPh.html

[10] https://snap.stanford.edu/data/as-733.html

[11] https://github.com/HazyResearch/hgcn/tree/master/data/disease_lp