





Dynamic Network Embedding in Hyperbolic Space via Self-attention

Dingyang Duan^{1,2} , Daren Zha^{1,2}, Xiao Yang^{3,4}, Nan Mu^{1,2} ,
and Jiahui Shen^{1,2}

¹ Institute of Information Engineering, Chinese Academy of Sciences, Beijing, China
{duandingyang, zhadaren, munan, shenjiahui}@iie.ac.cn

² School of Cyber Security, University of Chinese Academy of Sciences,
Beijing, China

³ Aerospace Internet of Things Technology Co., Ltd., Beijing, China
yangxiao@casciot.com

⁴ China Aerospace Times Electronics Co., Ltd., Beijing, China

Abstract. Graph Neural Networks (GNNs) have recently become increasingly popular due to their ability to learn node representations in complex graphs. Existing graph representation learning methods primarily target static graphs in Euclidean space, while many graphs in practical applications are dynamic and evolve constantly over time. Besides, most of these methods underestimate the inherent complex and hierarchical properties in real-world graphs, leading to sub-optimal embeddings. In this work, we propose a **Dynamic Network in Hyperbolic** space via **Self-Attention**, referred to as DynHAT, a novel neural architecture that computes node representations through joint two dimensions of hyperbolic structural graph and temporal attention graph. More specifically, DynHAT maps the structural graph into hyperbolic space to capture the hierarchical information, then temporal graph captures time-varying dynamic evolution over multiple time steps by flexibly weighting historical representations. Experimental results on three real-world datasets demonstrate the superiority of DynHAT for dynamic graph embedding, as it consistently outperforms competing methods in link prediction tasks.

Keywords: Dynamic graphs · Hyperbolic space · Self-attention · Representation learning

1 Introduction

Graph Neural Networks (GNNs) are widely used to model the complex graphs due to their ability to learn node representations. Its basic idea is to map each node to a vector in a low-dimensional representation space. By learning graph representations, classical machine learning algorithms can be applied to solve various task, such as link prediction and node classification. However, many real-world graphs, such as social networks where graph structures constantly evolve

over time, often exhibit scale-free or hierarchical structure [4], and Euclidean embeddings, used by existing GCNs, have a high distortion when embedding such graphs [24]. 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 [26]. This requires the learned node representations to not only preserve structural proximity but also jointly capture their temporal evolution.

Most of these existing studies model to dynamic graphs can be divided into two different approaches: discrete-time approaches where the evolution of a dynamic graph can be described by a sequence of static graphs, with a fixed timestamp; and continuous-time approaches where the evolution is modeled at a finer temporal granularity to encompass different events in real time [21]. Essentially, these two approaches both are primarily designed for the graphs in Euclidean spaces. However, many real-world graphs, such as protein interaction networks and social networks, often exhibit scale-free or hierarchical structure [2]. In particular, the scale-free graphs have tree-like structure and in such graphs the graph volume, defined as the number of nodes within some radius to a center node, grows exponentially as a function of radius. In such cases, the polynomial expansion Euclidean space can neither capture the exponential complexity nor provide the most powerful or meaningful geometry for graph representation learning. So, the volume of balls in Euclidean space only grows polynomial with respect to the radius, which leads to high distortion embeddings [19], while in hyperbolic space, this volume grows exponentially. Therefore, Hyperbolic geometry offers an exciting alternative as it enables embeddings with much smaller distortion when embedding scale-free and hierarchical graphs.

Learning dynamic node representations is challenging due to the complex time-varying graph structures. This requires the learned node representations to not only preserve structural proximity but also jointly capture their temporal evolution. For instance, in email communication networks whose interaction structures may change dramatically due to sudden events, users will join or quit a network at any time, and also they may develop new relationships or break up with others over time. More information could be captured when we consider the dynamic features of a graph. In this case, it is common practice to build a recurrent neural networks (RNN) that summarize historical snapshots via hidden state, for example [7, 8] which mainly focus on mining the pattern of graph evolvement. However, the disadvantage of RNN is also obvious, which requires amounts of data and scale poorly with an increase in number of time steps. Attention mechanisms have recently achieved great success in many sequential learning tasks. The idea behind the attention mechanism was to permit the decoder to utilize the most relevant parts of the input sequence in a flexible manner, by a weighted combination of all of the encoded input vectors, with the most relevant vectors being attributed the highest weights. When a single sequence is used as both the input and context, it is called self-attention, which are initially designed to facilitate RNNs to capture long-range dependencies. This paper [22] demonstrates the efficiency of a pure self-attentional network in achieving state-of-the-art performance in machine translation. As the change

on graphs may be periodical and frequent, self-attention is able to draw context from all past graph snapshots to adaptively assign interpretable weights for previous time steps.

Inspired by the aforementioned insight, we present a novel neural architecture named Dynamic Network in Hyperbolic Space via Self-Attention, referred to as DynHAT, to learn latent node representations on dynamic graphs. DynHAT fully leverages the implicit hierarchical information to capture the spatial dependency and graph evolution over multiple time steps by flexibly weighting historical representations. In summary, the main contributions are stated as follows:

- We propose a novel hyperbolic temporal graph embedding model, named DynHAT, to learn temporal regularities, topological dependencies, and implicitly hierarchical organization.
- We devise a modular temporal self-attention layer, which captures the most relevant historical contexts through efficient self-attentions. To the best of our knowledge, this is the first study on dynamic graph embedding that utilizes joint hyperbolic structural and temporal self-attention.
- Experimental results on three real-world datasets demonstrate the superiority of DynHAT for dynamic graph embedding, as it consistently outperforms competing methods in link prediction tasks. 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 structure graph embeddings and temporal graph embeddings.

Structure Graph Embeddings. Static network embedding methods can be classified into two categories: one for plain networks, another one for complex information networks. The first type of approaches only utilizes the topological structure information for embedding. DeepWalk [18] transforms graph structure information into sequences by random walk. Node2vec [9] improves the random walk strategies of DeepWalk by a controllable deep or wide walking possibility. Instead of shallow embeddings, several graph neural network architectures have achieved tremendous success. GCN [13] performs graph convolutions for aggregation and update motivated by spectral convolution. GAT [23] incorporates the attention mechanism into the aggregation step. GraphSAGE [11] considers from the spatial perspective and introduces an inductive learning method. All these methods assume the representation space to be Euclidean.

Hyperbolic Graph Embeddings. Hyperbolic space provides an exciting alternative. An increasing number of studies generalize the graph convolution into hyperbolic space, recent works including HGNN [15], HGCN [3] and HGAT [27]. HGCN is a generalization of inductive GCNs in hyperbolic geometry that

benefits from the expressiveness of both graph neural networks and hyperbolic embeddings. HGAT employs the framework of gyrovector spaces to implement the graph processing in hyperbolic spaces and design an attention mechanism based on hyperbolic proximity. The superior performance brought by hyperbolic geometry on static graphs motivates us to explore it on temporal graphs.

Dynamic Graph Embeddings. Recently, several solutions for dynamic graph are proposed. As discussed earlier, Temporal graphs are mainly defined in two ways: discrete-time approaches, where its life span is a discrete set, hence the evolution of a dynamic graph can be described by a sequence of static graphs, with a fixed timestamp; and continuous-time approach, where its life span is a continuous set, therefore the evolution is modeled at a finer temporal granularity to encompass different events in real time [25]. We here mainly focus on representation learning over discrete temporal graphs. DynamicTriad [28] constraints the representation in each time step, by the formulation that triadic closure process is more frequent along graph evolving. Dyngraph2vec [7] and Dyngem [8] use Auto-Encoder to learn the graph and use the Recurrent Neural Network (RNN) to model the relations over time. DySAT [20] applies attention mechanism in it, learning structural and temporal attention to adaptively obtain useful information for embedding. Most of the prevalent methods are built-in Euclidean space which leads to high distortion embeddings.

3 Preliminary

In this section, we first present the problem formulation of temporal graph embedding. Then, we introduce some fundamentals of hyperbolic geometry.

3.1 Problem Formulation

In this work, we formally define the problem of dynamic graph representation learning. A dynamic graph is defined as a series of observed static graph snapshots, $\mathbb{G} = \{\mathcal{G}_1, \dots, \mathcal{G}_T\}$ where T is the number of time steps. Each snapshot $\mathcal{G}_t = (V_t, A_t) \in \mathcal{G}$ is a weighted and undirected network snapshot recorded at time t , where V_t is the set of vertices and A_t is the corresponding adjacency matrix at time step t . Unlike some previous methods that assume links can only be added in dynamic graphs, we also support removal of links over time. Dynamic graph representation learning aims to learn a mapping function that obtains a low-dimensional representation for each node at time steps $t = \{1, 2, \dots, T\}$. Each node embedding h_v^t preserves both local graph structures centered at v and its temporal evolutionary behaviors such as link connection and removal up to time step t .

3.2 Hyperbolic Geometry

A Riemannian manifold \mathcal{M} is a space that generalizes the notion of a 2D surface to higher dimensions [5]. For each point $\mathbf{x} \in \mathcal{M}$, it associates with a (Euclidean)

tangent space $\mathcal{T}_x\mathcal{M}$ of the same dimensionality as \mathcal{M} . Intuitively, $\mathcal{T}_x\mathcal{M}$ contains all possible directions in which one can pass through \mathbf{x} tangentially (see Fig. 1).

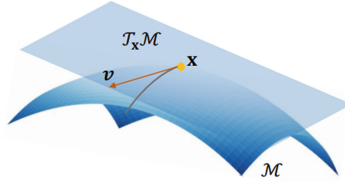


Fig. 1. The tangent space $\mathcal{T}_x\mathcal{M}$ and a tangent vector v , along the given point \mathbf{x} of a curve traveling through the manifold \mathcal{M} .

There are multiple equivalent models for hyperbolic space, with each representation conserving some geometric properties, but distorting others. In this paper, we adopt the Poincare ball model which is a compact representative providing visualizing and interpreting hyperbolic embeddings.

The Poincare ball model with negative curvature $-c(c \geq 0)$ corresponds to the Riemannian manifold $(\mathbb{H}^{n,c}, g_{\mathbb{H}})$, where $\mathbb{H}^{n,c} = \{\mathbf{x} \in \mathbb{R}^n : c\|\mathbf{x}\|^2 \leq 1\}$ 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, [5] shows how Euclidean and hyperbolic spaces can be continuously deformed into each other and provides 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 Proposed Model

The overall framework of the proposed model (DynHAT) is illustrated in Fig. 2. DynHAT has two primary modules: **Hyperbolic structure attention**; **Euclidean temporal attention**, which benefits from the expressiveness of both hyperbolic embeddings and temporal evolutionary embeddings. More specifically, our model can be summarized as two procedures: (1) Given the original input node feature, this procedure projects it into hyperbolic space, and aggregates the latent node embeddings via attention mechanism based on the hyperbolic proximity. This procedure concerns topological embedding in hyperbolic space, which is similar to the model HGAT that designs an attention-based graph convolution in hyperbolic space. (2) These sequences of node representations then feed as input to the temporal attention, which are performed in Euclidean space due to its computational efficiency. Owing to the superiorities of self-attention, this unit fuses the final embedding by figuring out the importance of each time step graph snapshots. The outputs of temporal module comprise the set of final dynamic node representations. Furthermore, we endow

our model with expressivity to capture dynamic graph evolution from different latent perspectives through multi-head attentions [22]. Finally, we feed the aggregated representations to a loss function for downstream task. We elaborate on the details of each respective module in the following paragraphs.

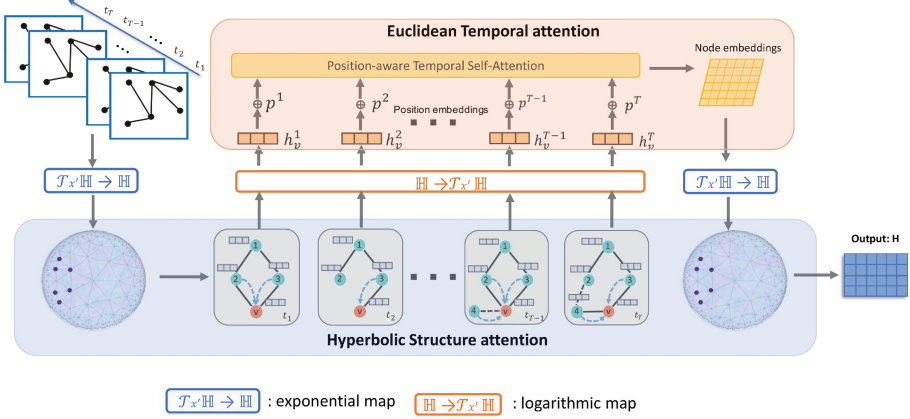


Fig. 2. Neural architecture of DynHAT: we employ structural attention layer in Hyperbolic space followed by temporal attention layers in Euclidean space.

4.1 Feature Map

Before going into the details of each module, we introduce two bijection operations, the exponential map and the logarithmic map, for mapping between hyperbolic space and tangent space with a local reference point [3, 15], as presented below.

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

$$\exp_{\mathbf{x}'}^c(a) = \mathbf{x}' \oplus^c \left(\tanh\left(\frac{\sqrt{c}\lambda_{\mathbf{x}'}^c \|a\|}{2}\right) \frac{a}{\sqrt{c}\|a\|} \right), \quad (1)$$

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

$$u \oplus v := \frac{(1 + 2c \langle u, v \rangle + c\|v\|^2)u + (1 - c\|u\|^2)v}{1 + 2c \langle u, v \rangle + c^2\|u\|^2\|v\|^2}. \quad (2)$$

The logarithmic map is given by:

$$\log_{\mathbf{x}'}^c(b) := \frac{2}{\sqrt{c}\lambda_{\mathbf{x}'}^c} \operatorname{artanh}\left(\sqrt{c}\|-\mathbf{x}' \oplus^c b\|\right) \frac{-\mathbf{x}' \oplus^c b}{\|-\mathbf{x}' \oplus^c b\|}. \quad (3)$$

Note that x' is a local reference point, we use the origin point 0 in our work.

4.2 Hyperbolic Structure Attention (HSA)

HSA is employed to extract features from higher-order local neighborhoods of each node through a self-attention aggregation and stacking, to compute intermediate node representations for each snapshot, which leveraging promising properties of hyperbolic geometry. The input of HSA is the node feature, whose norm could be out of the Poincare ball defined in hyperbolic space. To make the node feature available in hyperbolic space, we use the exponential map to project the feature into the hyperbolic space, shown in proposition 1. Specifically, let a Euclidean space vector $\mathbf{x}_i^E \in \mathbb{R}^d$ be the feature of node i , and then we regard it as the point in the tangent space $\mathcal{T}_{\mathbf{x}'}\mathbb{H}^{d,c}$ with reference point $\mathbf{x}' \in \mathbb{H}^{d,c}$, using the exponential map to project it into hyperbolic space, obtaining $\mathbf{x}^{\mathcal{H}} \in \mathbb{H}^{d,c}$, which is defined as:

$$\mathbf{x}_i^{\mathcal{H}} = \exp_{\mathbf{x}'}^c(\mathbf{x}_i^E). \quad (4)$$

We then transform $\mathbf{x}_i^{\mathcal{H}}$ into a higher-level latent representation $m_i^{\mathcal{H}}$ to obtain sufficient representation power, which is formulated as:

$$m_i^{\mathcal{H}} = W \otimes^c \mathbf{x}_i^{\mathcal{H}} \oplus^c b. \quad (5)$$

Considering vector multiplication and bias addition can not be directly applied since the operations in hyperbolic space fail to meet the permutation invariant requirement, for vector multiplication, we first project the hyperbolic vector to the tangent space, which is given by:

$$W \otimes^c \mathbf{x}_i^{\mathcal{H}} := \exp_{\mathbf{x}'}^c(W \log_{\mathbf{x}'}^c(\mathbf{x}_i^{\mathcal{H}})). \quad (6)$$

For bias addition, we transport the bias located at $\mathcal{T}_o\mathbb{H}$ 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 and Mobius addition to compute the bias addition: $\mathbf{x}^{\mathcal{H}} \oplus^c b := \exp_{\mathbf{x}}^c(P_{o \rightarrow x}(b))$.

The Hyperbolic Attention Mechanism. We perform a self-attention mechanism on the nodes. Aggregation is to calculate a weighted midpoint in Euclidean space [10], however, it is difficult to apply as it lacks a closed form to compute the derivative easily [1]. Similar to [3, 15, 27], we address this issue by applying the aggregation computation in the tangent space. The attention coefficient α_{ij} , which indicates the importance of node j to node i , can be computed as:

$$\alpha_{ij} = \text{softmax}_{(j \in \mathcal{N}(i))}(s_{ij}) = \frac{\exp(s_{ij})}{\sum_{j' \in \mathcal{N}_i} \exp(s_{ij'})}, \quad (7)$$

$$s_{ij} = \sigma(a^T [\log_0^c(m_i^l) \parallel \log_0^c(m_j^l)]), \forall (i, j) \in \mathcal{E}. \quad (8)$$

Thus, a hyperbolic structural attention layer applies on a snapshot \mathcal{G} outputs node embeddings, through a self-attentional aggregation of neighboring node embeddings, which can be viewed as a single message passing round among immediate neighbors.

4.3 Euclidean Temporal Attention (ETA)

The embeddings input to a layer can potentially vary across different snapshots. We denote the node representations output by the HSA block, as $h_v^1, h_v^2, \dots, h_v^T$, which feed as input to the temporal attention block. To further capture temporal evolutionary patterns in a dynamic graph, we design a temporal self-attention layer. Note that, this layer is performed in the tangent space due to its computational efficiency. The h_v^t is expressed as:

$$h_v^t = X_t^E = \log_{\mathbf{x}'}^c(\tilde{\mathbf{x}}_t^{\mathcal{H}}). \quad (9)$$

This unit receives the sequential input h_v^T for a particular node v at different time steps from hyperbolic structure attention layer. The input representations are assumed to sufficiently capture local structural information at each time step. Once obtained the node representations for each time step snapshot, the next is to aggregate these node embeddings across a series of time snapshots. First, we capture the ordering information in the temporal attention module by using *position* embedding [6], p^1, p^2, \dots, p^3 , which embed the absolute temporal position of each snapshot. The position embeddings are combined with the output of the HSA module to obtain a sequence of input representations: $h_v^1 + p^1, h_v^2 + p^2, \dots, h_v^T + p^T$ for node v across multiple time steps.

In contrast to HSA layer which operates on the representations of neighboring nodes, temporal attention layer takes all the temporal history of each node into account. To be specific, to compute the output representation of node v at time step t , we use scaled dot-product form of attention [22] where the queries, keys, and values are set as the input node representations. The (Q, K, V) are first transformed to a different space through linear projection matrices $W_q \in \mathbb{R}^{D' \times F'}$, $W_k \in \mathbb{R}^{D' \times F'}$ and $W_v \in \mathbb{R}^{D' \times F'}$ respectively. Then, we allow each time step t to attend over all time steps up to and including t , to preserve the auto-regressive property. The temporal self-attention function is defined as:

$$h_v^{ij} = \frac{((X_v W_q)(X_v W_k)^T)_{ij}}{\sqrt{F'}} + M_{ij}, \quad (10)$$

$$\beta_v^{ij} = \frac{\exp(h_v^{ij})}{\sum_{k=1}^T \exp(h_v^{ik})}, \quad (11)$$

$$Z_v = \beta_v (X_v W_v), \quad (12)$$

$$H^{\mathcal{H}} = \exp_{\mathbf{x}'}^c(Z_v) \quad (13)$$

where X_v denotes the $h_v^T + p^T$ which as the query to attend over its historical representations, $\beta_v \in \mathbb{R}^{T \times T}$ is the attention weight matrix obtained by multiplicative attention function and $M \in \mathbb{R}^{T \times T}$ is a mask matrix with each entry $M_{ij} \in \{-\infty, 0\}$ to enforce the auto-regressive property following the function [20]. To encode the temporal order, we define M as:

$$M_{ij} = \begin{cases} 0, & i \leq j \\ -\infty, & \text{otherwise} \end{cases} \quad (14)$$

When $M_{ij} = -\infty$, the softmax results in a zero attention weight, i.e., $\beta_v^{ij} = 0$, which switches off the attention from time-step i to j .

As the temporal attention is built in the Euclidean space, different with the structure attention unit, we need to feed the output embeddings back to the hyperbolic space (as given in Eqs. (13)).

4.4 Learning Objective

We formulate the learning objective to maximize the probability of linked nodes and minimize the probability of no interconnected nodes. In our model, we use the dynamic representation of a node v at time step t , h_v^t to preserve local proximity around v at time step t . The loss function \mathcal{L} is based on binary cross-entropy which is defined as:

$$\mathcal{L} = \sum_{t=1}^T \sum_{v \in V} \left(\sum_{u \in \mathcal{N}_{walk}^t(v)} -\log(p(h_u^t, h_v^t)) - w_n \sum_{u' \in P_n^t(v)} \log(1 - p(h_{u'}^t, h_v^t)) \right) \quad (15)$$

where p is the probability which could be inferred by the Fermi-Dirac function [3], and $\mathcal{N}_{walk}^t(v)$ is the set of nodes that co-occur with v on fixed-length random walks at snapshot t . P_n^t is a negative sampling distribution for time step t , and w_n is the negative sampling ratio, a hyper-parameter to balance the positive and negative samples. Note that learning the representation of each node in hyperbolic space, the loss function \mathcal{L}_t is only related to distance in the Poincare ball, and benefits to large-scale datasets.

5 Experiments and Analysis

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

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

5.1 Datasets

To evaluate the effectiveness of our model, we conduct experiments on three datasets from real-world platforms. The datasets are summarized in Table 1.

- **Enron** [14] Enron dataset was collected and prepared by the CALO Project (A Cognitive Assistant that Learns and Organizes). It contains emails between employees of the company between January 1991 and July 2002.
- **UCI** [16] UCI dataset draws on longitudinal network data from an online community to examine patterns of users' behavior and social interaction, and infer the processes underpinning dynamics of system use. In this network, connections between users are made through online information.

- **MovieLens** [12] MovieLens contains rating data of multiple users for multiple movies, including movie metadata information and user attribute information. GroupLens Research collected data on movie ratings provided by MovieLens users from the late 1990s to the early 2000s. In this paper, we use a subset of MovieLens, ML-10M, consists of a user-tag interactions where the links connect users with the tags they applied on certain movies.

Table 1. Summary of the datasets

Dataset	Enron	UCI	ML-10M
Nodes	143	1,809	20,537
Edges ^a	2347	16,822	43,760
Time steps ^b	16	13	13

^a Edge counts denotes the total edges across all time steps.

^b The duration of each snapshot will affect the total number of snapshots. The proper granularity is more beneficial to capture evolving patterns.

5.2 Baselines

We present comparisons against several static graph embedding methods to analyze the gains of using temporal information for link prediction. To ensure a fair comparison, we provide access to the entire history of snapshots by constructing an aggregated graph up to time t , with link weights proportional to the cumulative weight till t agnostic to link occurrence time. More importantly, we also conduct experiments on several temporal graph embedding models to further demonstrate the superiority of the proposed DynHAT. These models are all in Euclidean space. As for hyperbolic model, HG2Vec, a recent model for static graphs, is used for one of baselines.

- **Node2vec** [9] Node2vec is a static embedding method to generate vector representations of nodes on a graph. It learns low-dimensional representations for nodes in a graph through the use of random walks.
- **GraphSAGE** [11] GraphSAGE is a framework for inductive representation learning on large graphs. It can be used to generate node embeddings for previously unseen nodes or entirely new input graphs, as long as these graphs have the same attribute schema as the training data.
- **GAT** [23] GAT leverages masked self-attentional layers to address the shortcomings of prior methods based on graph convolutions or their approximations. The self-attention mechanism is an advanced method which our model also takes advantage of.

- **HGCN** [3] HGCN is a static embedding method which leverages both the expressiveness of GCNs and hyperbolic geometry to learn node representations for hierarchical and scale-free graphs.
- **EvolveGCN** [17] EvolveGCN is a temporal model that extends GCN, which computes a separate GCN model for each time step. The model is updated upon an input to the system every time, by using an RNN (e.g., GRU). There are two versions of the EvolveGCN: EvolveGCN-O and EvolveGCN-H. We test both and report the best result.
- **DynamicTriad** [28] DynamicTriad focuses on specific structure of triad to model how close triads are formed from open triads in dynamic networks.
- **DySAT** [20] Dynamic network employs GAT as a static layer and self-attention mechanism to capture the temporal graph evolution.

We use the same split as the previous works, i.e., 25% for the training set and 75% for the test set. The number of negative samples for each positive sample is 1. Besides, we fix the curvature as 1 in Hyperbolic structure attention layer.

5.3 Link Prediction Comparison (RQ 1)

We obtain node representations from DynHAT which can be applied to link prediction. In this paper, we conduct experiments on single-step and multi-step link prediction. Using the node representation trained on graph snapshots up to time step t , the single-step link prediction predicts the connections between nodes at time step $t + 1$, while the multi-step link prediction predicts at multiple time steps start from $t + 1$.

More specifically, given partially observed snapshots of a temporal graph $\mathcal{G} = \{\mathcal{G}_\infty, \dots, \mathcal{G}_T\}$, for single-step prediction, the latest embeddings h_v^t are used to predict the links at \mathcal{G}^{t+1} , classifying each node pair into links and non-links. For multi-step link prediction, the latest embeddings are used to predict the links at multiple future time steps $\{t + 1, \dots, t + \Delta\}$. In each dataset, we set $\Delta = 6$ for evaluation.

We use the Area Under the ROC Curve (AUC) [9] metric to evaluate link prediction performance. Note that we uniformly train both the baselines and DynHAT by using early stopping based on the performance of the training set.

Single-Step Link Prediction. The results of single-step link prediction are shown in Table 2. Our results indicate that the proposed model achieves gains of 3–4% AUC and AP, comparing to the best baseline across all datasets. On the one hand, the runners-up goes to the other temporal graph embedding model, which confirms the importance of temporal regularity in dynamic graph modeling. On the other hand, for hyperbolic model in static graph, our model consistently outperforms HGCN. In the following, we discuss the several insights from the comparative analysis of different methods.

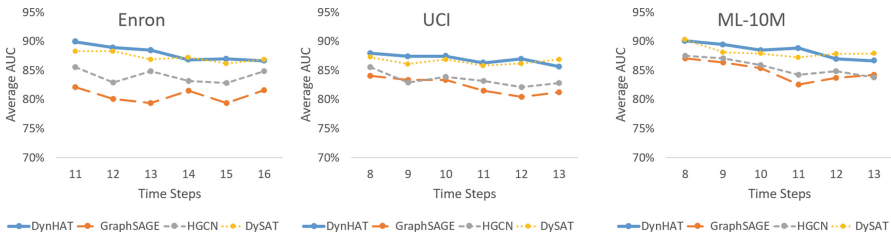
First of all, the DySAT achieves the competitive performance to dynamic embedding methods across different datasets, underperforming DynHAT only by 3.88% and 2.67% in AUC and AP scores, respectively, despite of in Euclidean

Table 2. AUC (left) and AP (right) scores of single-step link prediction result.

Dataset	AUC			AP		
	Enron	UCI	ML-10M	Enron	UCI	ML-10M
Node2vec	82.81 ± 0.8	79.45 ± 0.7	85.57 ± 0.2	81.44 ± 1.1	78.22 ± 0.8	87.45 ± 0.3
GraphSAGE	84.3 ± 0.9	82.11 ± 0.6	87.3 ± 0.1	85.57 ± 1.2	83.67 ± 0.9	86.2 ± 0.1
GAT	83.8 ± 1.2	80.07 ± 0.9	85.4 ± 0.1	83.17 ± 1.9	79.93 ± 0.6	86.4 ± 0.3
HGCN	85.66 ± 0.8	82.71 ± 0.7	88.32 ± 0.2	82.45 ± 0.9	85.63 ± 0.3	86.17 ± 0.4
EvolveGCN	84.85 ± 0.9	84.16 ± 0.4	87.9 ± 0.2	86.23 ± 0.7	84.73 ± 1.1	86.72 ± 0.2
DynamicTriad	81.02 ± 0.6	84.51 ± 0.4	86.42 ± 0.1	85.83 ± 0.9	82.35 ± 0.5	86.59 ± 0.1
DySAT	88.50 ± 0.3	86.65 ± 0.2	89.73 ± 0.2	88.20 ± 0.4	86.7 ± 0.2	89.96 ± 0.1
DynHAT	91.88 ± 1.2	88.78 ± 0.2	91.23 ± 0.3	90.87 ± 0.8	88.51 ± 1.2	93.16 ± 0.2

space. One possible explanation is that joint structural and temporal modeling with expressive aggregators like multi-head attentions plays an important role for superior performance on link prediction. The performance gap between DynHAT and DySAT suggests that the significantly benefit from hyperbolic geometry. Second, HGCN also has relatively good performance despite being agnostic to temporal information, which indicates further improvements to DynHAT on transforming the embeddings from Euclidean space to Hyperbolic space.

Multi-step Link Prediction. In this section, we select several relatively good performance models from difference aspect for comparison, then evaluate them on multi-step link prediction over $t + \Delta$ time steps, where Δ equals to 6. As is shown in Fig. 3, we observe a slight decay in performance overtime for all the models, which is expected. Specifically, we notice that the performance of each method except DySAT drops by different degrees, while our model and DySAT maintains a stable result overtime, that can attribute to the temporal attention module. For instance, the performance of the HGCN degrades dramatically on Enron from 85.54% to 82.88%, while DynHAT only declines about 1.3%. This demonstrates the capability of the temporal attention module to capture the most relevant historical context. Additionally, DynHAT maintains a consistent performance, comparing with DySAT which learns the node representation in Euclidean space. The superiority of hyperbolic space and the importance of modeling temporal context are supported in the experimental results.

**Fig. 3.** AUC performance of DynHAT with different models on multi-step link prediction.

5.4 Ablation Study (RQ 2)

To investigate the superiority of the main components of our model, we compare DynHAT with different variants on Enron, UCI and Movielens datasets. We show the variant models as following and their result in Table 3. To validate the performance of temporal self-attention block for long-term prediction task, we set the finer granularity of snapshot, which means the shorter of duration.

- **DynHA** DynHAT removes the temporal attention block. Note that the variant model is different from static models since the embeddings are jointly optimized in Eqs. (15) to predict snapshot-specific neighborhoods, however without any explicit temporal evolution modeling.
- **DynAT** DyHAT without hyperbolic geometry where aggregating processes are built in Euclidean space.

Table 3. Ablation study on structural and temporal attention layer.

Dataset	AUC			AP		
	Enron	UCI	ML-10M	Enron	UCI	ML-10M
DynHA	88.93 \pm 0.5	86.12 \pm 0.5	86.76 \pm 0.2	89.83 \pm 0.8	85.71 \pm 0.2	88.52 \pm 0.2
DynAT	89.82 \pm 0.4	87.35 \pm 0.3	90.68 \pm 0.5	89.62 \pm 0.3	87.67 \pm 0.3	91.73 \pm 0.3
Original	91.88 \pm 1.2	88.78 \pm 0.2	91.23 \pm 0.3	90.87 \pm 0.8	88.51 \pm 1.2	93.16 \pm 0.2

As is shown in Table 3, we first make the wrap-up observation that removing any of the components will cause performance degradation. The effect of temporal self-attention layer is significant since the performance is decayed by removing the temporal block. Besides, we find that the finer granularity of snapshots, which means the shorter of the duration, would more fully utilized the historical representations especially in MovieLens dataset. This observation conforms to the nature of graph evolution since the rating behaviors in MovieLens correlated with time-efficient, while the communications in Enron and UCI span longer time intervals. We confirm that the proposed model leverage the temporal evolution embedding to better perform the final results. Additionally, we design the Euclidean variant of DynHAT, namely DynAT, proves that hyperbolic geometry enables the preservation of the hierarchical layout in the graph data naturally. DynHAT consistently outperforms the variants with almost 3% average gain in AUC and AP scores, validating our choice of joint structural in hyperbolic and temporal self-attention. Finally, the addition of both DynHA and DynAT improves performance even further, suggesting that both components are important in DynHAT.

6 Conclusions

In this work, we have presented a novel model DynHAT for dynamic graph representation learning in hyperbolic space. In DynHAT, we stack temporal attention layers on top of hyperbolic structural attention layers, considering distortions when representing real-world in Euclidean space. More specifically, our model computes dynamic node representations through joint two modules: hyperbolic structure attention (HSA) and Euclidean temporal attention (ETA). HSA leverages the superiority of hyperbolic mechanism and ETA captures the most relevant historical contexts through efficient self-attentions. To the best of our knowledge, this is the first work to address the temporal graph embedding via self-attention mechanism built-in hyperbolic space. Experimental results show the superiority of DynHAT for link prediction on several real-world datasets. For future work, we hope that our work will inspire the future development of dynamic graph embeddings in hyperbolic space.

References

1. Bacák, M.: Computing medians and means in hadamard spaces. *SIAM J. Optim.* **24**(3), 1542–1566 (2014)
2. Bronstein, M.M., Bruna, J., LeCun, Y., Szlam, A., Vandergheynst, P.: Geometric deep learning: going beyond euclidean data. *IEEE Sig. Process. Mag.* **34**(4), 18–42 (2017)
3. Chami, I., Ying, Z., Ré, C., Leskovec, J.: Hyperbolic graph convolutional neural networks. *Adv. Neural. Inf. Process. Syst.* **32**, 4868–4879 (2019)
4. Clauset, A., Moore, C., Newman, M.E.: Hierarchical structure and the prediction of missing links in networks. *Nature* **453**(7191), 98–101 (2008)
5. Ganea, O.E., Bécigneul, G., Hofmann, T.: Hyperbolic neural networks. *arXiv preprint [arXiv:1805.09112](https://arxiv.org/abs/1805.09112)* (2018)
6. Gehring, J., Auli, M., Grangier, D., Yarats, D., Dauphin, Y.N.: Convolutional sequence to sequence learning. In: *International Conference on Machine Learning*, pp. 1243–1252. PMLR (2017)
7. Goyal, P., Chhetri, S.R., Canedo, A.: dyngraph2vec: capturing network dynamics using dynamic graph representation learning. *Knowl. Based Syst.* **187**, 104816 (2020)
8. Goyal, P., Kamra, N., He, X., Liu, Y.: DynGEM: deep embedding method for dynamic graphs. *arXiv preprint [arXiv:1805.11273](https://arxiv.org/abs/1805.11273)* (2018)
9. Grover, A., Leskovec, J.: node2vec: scalable feature learning for networks. In: *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 855–864 (2016)
10. Gulcehre, C., et al.: Hyperbolic attention networks. *arXiv preprint [arXiv:1805.09786](https://arxiv.org/abs/1805.09786)* (2018)
11. Hamilton, W.L., Ying, R., Leskovec, J.: Inductive representation learning on large graphs. In: *Proceedings of the 31st International Conference on Neural Information Processing Systems*, pp. 1025–1035 (2017)
12. Harper, F.M., Konstan, J.A.: The movielens datasets: history and context. *ACM Trans. Interact. Intell. Syst. (THIS)* **5**(4), 1–19 (2015)

13. Kipf, T.N., Welling, M.: Semi-supervised classification with graph convolutional networks. arXiv preprint [arXiv:1609.02907](https://arxiv.org/abs/1609.02907) (2016)
14. Klimt, B., Yang, Y.: The enron corpus: a new dataset for email classification research. In: Boulicaut, J.-F., Esposito, F., Giannotti, F., Pedreschi, D. (eds.) ECML 2004. LNCS (LNAI), vol. 3201, pp. 217–226. Springer, Heidelberg (2004). https://doi.org/10.1007/978-3-540-30115-8_22
15. Liu, Q., Nickel, M., Kiela, D.: Hyperbolic graph neural networks. arXiv preprint [arXiv:1910.12892](https://arxiv.org/abs/1910.12892) (2019)
16. Panzarasa, P., Opsahl, T., Carley, K.M.: Patterns and dynamics of users' behavior and interaction: network analysis of an online community. *J. Am. Soc. Inform. Sci. Technol.* **60**(5), 911–932 (2009)
17. Pareja, A., et al.: EvolveGCN: evolving graph convolutional networks for dynamic graphs. In: Proceedings of the AAAI Conference on Artificial Intelligence, vol. 34, pp. 5363–5370 (2020)
18. Perozzi, B., Al-Rfou, R., Skiena, S.: DeepWalk: online learning of social representations. In: Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 701–710 (2014)
19. Sala, F., De Sa, C., Gu, A., Ré, C.: Representation tradeoffs for hyperbolic embeddings. In: International Conference on Machine Learning, pp. 4460–4469. PMLR (2018)
20. Sankar, A., Wu, Y., Gou, L., Zhang, W., Yang, H.: DySAT: deep neural representation learning on dynamic graphs via self-attention networks. In: Proceedings of the 13th International Conference on Web Search and Data Mining, pp. 519–527 (2020)
21. Trivedi, R., Farajtabar, M., Biswal, P., Zha, H.: DyRep: learning representations over dynamic graphs. In: International Conference on Learning Representations (2019)
22. Vaswani, A., et al.: Attention is all you need. In: Advances in Neural Information Processing Systems, pp. 5998–6008 (2017)
23. Veličković, P., Cucurull, G., Casanova, A., Romero, A., Lio, P., Bengio, Y.: Graph attention networks. arXiv preprint [arXiv:1710.10903](https://arxiv.org/abs/1710.10903) (2017)
24. Wei, C., Fang, W., Hu, G., Mahoney, M.W.: On the hyperbolicity of small-world and tree-like random graphs. In: International Symposium on Algorithms and Computation (2012)
25. Yang, M., Meng, Z., King, I.: FeatureNorm: L2 feature normalization for dynamic graph embedding. In: 2020 IEEE International Conference on Data Mining (ICDM), pp. 731–740. IEEE (2020)
26. Yang, M., Zhou, M., Kalander, M., Huang, Z., King, I.: Discrete-time temporal network embedding via implicit hierarchical learning in hyperbolic space. In: Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining, pp. 1975–1985 (2021)
27. Zhang, Y., Wang, X., Shi, C., Jiang, X., Ye, Y.F.: Hyperbolic graph attention network. *IEEE Trans. Big Data* (2021)
28. Zhou, L., Yang, Y., Ren, X., Wu, F., Zhuang, Y.: Dynamic network embedding by modeling triadic closure process. In: Proceedings of the AAAI Conference on Artificial Intelligence, vol. 32 (2018)