# Hyperbolic Variational Graph Neural Network for Modeling Dynamic Graphs

**Li Sun[1], Zhongbao Zhang[1*], Jiawei Zhang[2], Feiyang Wang[1], Hao Peng[3], Sen Su[1*] and Philip S. Yu[4]**

[1]State Key Laboratory of Networking and Switching Technology, Beijing University of Posts and Telecommunications, China
[2]IFM Lab, Department of Computer Science, Florida State University, FL, USA
[3]Beijing Advanced Innovation Center for Big Data and Brain Computing, Beihang University, China
[4]Department of Computer Science, University of Illinois at Chicago, IL, USA
{l.sun, zhongbaozb, fywang}@bupt.edu.cn; jiawei@ifmlab.org; penghao@act.buaa.edu.cn; psyu@uic.edu

## Abstract

Learning representations for graphs plays a critical role in a wide spectrum of downstream applications. In this paper, we summarize the limitations of the prior works in three folds: representation space, modeling dynamics and modeling uncertainty. To bridge this gap, we propose to learn dynamic graph representation in hyperbolic space, for the first time, which aims to infer stochastic node representations. Working with hyperbolic space, we present a novel Hyperbolic Variational Graph Neural Network, referred to as HVGNN. In particular, to model the dynamics, we introduce a Temporal GNN (TGNN) based on a theoretically grounded time encoding approach. To model the uncertainty, we devise a hyperbolic graph variational autoencoder built upon the proposed TGNN to generate stochastic node representations of hyperbolic normal distributions. Furthermore, we introduce a reparameterisable sampling algorithm for the hyperbolic normal distribution to enable the gradient-based learning of HVGNN. Extensive experiments show that HVGNN outperforms state-of-the-art baselines on real-world datasets.

## Introduction

Recent years have witnessed a surge of representation learning on graphs (Perozzi, Al-Rfou, and Skiena 2014; Tang et al. 2015; Xhonneux, Qu, and Tang 2020). 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 graph analysis tasks, such as link prediction and node classification (Kipf and Welling 2016a).

In this paper, we summarize the limitations of the prior graph representation learning works in three folds:

**1) *Representation Space.*** Most of existing studies (Kipf and Welling 2016b; Xu et al. 2020) model graphs in the Euclidean space. Euclidean models tend to have distortions when representing real-world graphs with latent hierarchies (Chami et al. 2019; Chen et al. 2013). In particular, for such graphs, the number of nodes surrounding to a center node grows *exponentially w.r.t.* radius. However, the size of the Euclidean space only grows polynomially *w.r.t.* radius, while this size grows *exponentially* in hyperbolic space (Krioukov et al. 2010). *Hyperbolic* space provides a more promising alternative. Actually, recent results (Papadopoulos et al. 2012; Liu, Nickel, and Kiela 2019) show that hyperbolic space is well-suited for modeling graphs.

**2) *Modeling Dynamics.*** Most of existing models (Li et al. 2020; Tu et al. 2018; Wang, Cui, and Zhu 2016) consider the graphs to be static. Actually, graphs are usually *dynamic* and constantly evolving over time. Dynamic graphs have been typically observed in social networks, transportation networks and financial transaction networks (Zhou et al. 2018b). Ignoring the inherent dynamics of graphs usually leads to questionable inference. Such models may mistakenly utilize future information for predicting past interactions as the evolving constraints are disregarded.

**3) *Modeling Uncertainty.*** Most of existing models (Liu et al. 2020; Grover and Leskovec 2016) map nodes to deterministic vectors. However, the formation and evolution of graphs are full of uncertainties, especially for low-degree nodes which deliver less information and bear more uncertainties (Zhu et al. 2018). Actually, uncertainty is an inherent characteristic of graphs. The deterministic representation cannot model uncertainty. Alternatively, *stochastic* representation provides a promising approach to model such characteristic. It naturally captures the uncertainty to represent nodes as normal distributions, *i.e.*, the mean and variance.

To address the aforementioned limitations, we propose to learn dynamic graph representation in hyperbolic space, for the first time, which aims to learn stochastic node representations modeling graph dynamics and its uncertainty.

To this end, we present a novel Hyperbolic Variational Graph Neural Network, referred to as HVGNN. In HVGNN, to address the first limitation, instead of the Euclidean space, we utilize hyperbolic space as the representation space. To address the second limitation, we introduce a novel Temporal GNN (TGNN) to model the dynamics. In particular, TGNN performs a time-aware attention based on a theoretically grounded time encoding approach, which distinguishes nodes in time domain. To address the third limitation, we devise a hyperbolic graph variational autoencoder built upon TGNN to jointly model the uncertainty and dy-

*Corresponding Authors

namics. In particular, we generate stochastic representations of wrapped normal distributions, the generalized normal distributions in hyperbolic space, whose parameters are defined by TGNN. Furthermore, we introduce a reparameterisable sampling algorithm for wrapped normal distributions to enable the gradient-based learning of HVGNN.

We evaluate HVGNN by two typical downstream tasks on graph data: link prediction and node classification. Extensive experiments on real-world datasets show that HVGNN outperforms several state-of-the-art methods.

Overall, main contributions of our work are listed below:

- To the best of our knowledge, this is the first attempt to learn the node representations for dynamic graphs in hyperbolic space.
- We propose a novel Hyperbolic Variational Graph Neural Network, HVGNN, which generates stochastic representations to model graph dynamics and its uncertainty in hyperbolic space.
- Experimental results show the superiority of HVGNN on several real-world datasets.

## Preliminaries: Hyperbolic Geometry

For the in-depth expositions, refer mathematical materials (Loomis 2013; Hopper and Andrews 2010). Throughout the paper, we denote the Euclidean norm and inner product by $\|\cdot\|$ and $\langle\cdot,\cdot\rangle$, respectively.

### Riemannian Manifold

A *manifold* $\mathcal{M}$ is a space that generalizes the notion of a 2D surface to higher dimensions (Ganea, Bécigneul, and Hofmann 2018). For each point $\mathbf{x} \in \mathcal{M}$, it associates with a *tangent space* $\mathcal{T}_{\mathbf{x}}\mathcal{M}$ of the same dimensionality as $\mathcal{M}$. Intuitively, $\mathcal{T}_{\mathbf{x}}\mathcal{M}$ contains all possible directions in which one can pass through $\mathbf{x}$ tangentially. On the associated tangent space $\mathcal{T}_{\mathbf{x}}\mathcal{M}$, the *Riemannian metric*, $g_{\mathbf{x}}(\cdot,\cdot) : \mathcal{T}_{\mathbf{x}}\mathcal{M} \times \mathcal{T}_{\mathbf{x}}\mathcal{M} \to \mathbb{R}$, defines an inner product specifying the geometry of $\mathcal{M}$. A *Riemannian manifold* is then defined as the tuple of $(\mathcal{M}, g)$.

Mapping between the tangent space and the manifold is done via exponential and logarithmic maps. For a point $\mathbf{x} \in \mathcal{M}$ in the manifold, the *exponential map* at $\mathbf{x}$, $\exp_{\mathbf{x}}(\mathbf{v}) : \mathcal{T}_{\mathbf{x}}\mathcal{M} \to \mathcal{M}$, projects the vector $\mathbf{v} \in \mathcal{T}_{\mathbf{x}}\mathcal{M}$ onto the manifold $\mathcal{M}$. The *logarithmic map* at $\mathbf{x}$, $\log_{\mathbf{x}}(\mathbf{y}) : \mathcal{M} \to \mathcal{T}_{\mathbf{x}}\mathcal{M}$, projects the vector $\mathbf{y} \in \mathcal{M}$ back to the tangent space $\mathcal{T}_{\mathbf{x}}\mathcal{M}$. Mapping between tangent spaces is done via parallel transport. For two points $\mathbf{x}, \mathbf{y} \in \mathcal{M}$ in the manifold, the *parallel transport* from $\mathbf{x}$ to $\mathbf{y}$, $P_{\mathbf{x}\to\mathbf{y}}(\mathbf{v}) : \mathcal{T}_{\mathbf{x}}\mathcal{M} \to \mathcal{T}_{\mathbf{y}}\mathcal{M}$, carries the vector $\mathbf{v} \in \mathcal{T}_{\mathbf{x}}\mathcal{M}$ to $\mathcal{T}_{\mathbf{y}}\mathcal{M}$ along the *geodesic*, a smooth path on the manifold of minimal length between $\mathbf{x}$ and $\mathbf{y}$.

As one of the fundamental objects, the hyperbolic space is a Riemannian manifold with constant negative curvature. There are four common equivalent models of hyperbolic space: the Poincaré disk model, Poincaré half-plane model, Lorentz (*a.k.a.* hyperboloid/Minkowski) model and Klein model (Liu, Nickel, and Kiela 2019). Next, we give more details of the latter two, which will both be utilized in this paper, for further discussions.

### The Lorentz Model

We denote $\mathbb{L}^{d,K}$ as the Lorentz model in $d$ dimensions with constant negative *curvature* $-1/K$ where $K > 0$. The Lorentz model $\mathbb{L}^{d,K}$ is defined on a subset of $\mathbb{R}^{d+1}$, $\mathbb{L}^{d,K} = \{\mathbf{x} \in \mathbb{R}^{d+1} | \langle \mathbf{x}, \mathbf{x}\rangle_{\mathcal{L}} = -K\}$, where $\langle\cdot,\cdot\rangle_{\mathcal{L}} : \mathbb{R}^{d+1} \times \mathbb{R}^{d+1} \to \mathbb{R}$ is Lorentzian inner product defined as,

$$\langle \mathbf{x}, \mathbf{y}\rangle_{\mathcal{L}} = -x_0 y_0 + x_1 y_1 + x_2 y_2 + \cdots + x_d y_d. \quad (1)$$

We denote $\mathcal{T}_{\mathbf{x}}\mathbb{L}^{d,K}$ as the *tangent space* of $\mathbf{x} \in \mathbb{L}^{d,K}$. We have $\mathcal{T}_{\mathbf{x}}\mathbb{L}^{d,K} = \{\mathbf{v} \in \mathbb{R}^{d+1} | \langle \mathbf{v}, \mathbf{x}\rangle_{\mathcal{L}} = 0\}$ and $\|\mathbf{v}\|_{\mathcal{L}} = \sqrt{\langle \mathbf{v}, \mathbf{v}\rangle_{\mathcal{L}}}$ is the norm of $\mathbf{v}$. For $\mathbf{u}, \mathbf{v} \in \mathcal{T}_{\mathbf{x}}\mathbb{L}^{d,K}$, we can give the Riemannian metric tensor $g^K_{\mathbf{x}}(\mathbf{u}, \mathbf{v}) = \langle \mathbf{u}, \mathbf{v}\rangle_{\mathcal{L}}$.

Next, we give the closed form equations of exponential map, logarithmic map and parallel transport. For any $\mathbf{x}, \mathbf{y} \in \mathbb{L}^{d,K}$ and $\mathbf{v} \in \mathcal{T}_{\mathbf{x}}\mathbb{L}^{d,K}$, we have the following equations:

$$\exp^K_{\mathbf{x}}(\mathbf{v}) = \cosh\left(\frac{\|\mathbf{v}\|_{\mathcal{L}}}{\sqrt{K}}\right)\mathbf{x} + \sqrt{K}\sinh\left(\frac{\|\mathbf{v}\|_{\mathcal{L}}}{\sqrt{K}}\right)\frac{\mathbf{v}}{\|\mathbf{v}\|_{\mathcal{L}}}, \quad (2)$$

$$\log^K_{\mathbf{x}}(\mathbf{y}) = \frac{d^K_{\mathcal{L}}(\mathbf{x}, \mathbf{y})}{\left\|\mathbf{y} + \frac{1}{K}\langle \mathbf{x}, \mathbf{y}\rangle_{\mathcal{L}}\mathbf{x}\right\|_{\mathcal{L}}}\left(\mathbf{y} + \frac{1}{K}\langle \mathbf{x}, \mathbf{y}\rangle_{\mathcal{L}}\mathbf{x}\right), \quad (3)$$

$$P^K_{\mathbf{x}\to\mathbf{y}}(\mathbf{v}) = \mathbf{v} - \frac{\left\langle \log^K_{\mathbf{x}}(\mathbf{y}), \mathbf{v}\right\rangle_{\mathcal{L}}}{d^K_{\mathcal{L}}(\mathbf{x}, \mathbf{y})^2}\left(\log^K_{\mathbf{x}}(\mathbf{y}) + \log^K_{\mathbf{y}}(\mathbf{x})\right), \quad (4)$$

where $d^K_{\mathcal{L}}(\mathbf{x}, \mathbf{y}) = \sqrt{K}\cosh^{-1}(-\langle \mathbf{x}, \mathbf{y}\rangle_{\mathcal{L}}/K)$.

### The Klein Model

We denote $\mathbb{K}^{d,K}$ as the Klein model in $d$ dimensions with constant negative curvature $-1/K$ where $K > 0$. The Klein model $\mathbb{K}^{d,K}$ is defined on a subset of $\mathbb{R}^d$, $\mathbb{K}^{d,K} = \{\mathbf{x} \in \mathbb{R}^d \mid ||\mathbf{x}||^2 < K\}$. A point $\mathbf{x} \in \mathbb{K}^{d,K}$ is projected from the corresponding $\mathbf{y} \in \mathbb{L}^{d,K}$. We derive the projection with the $i^{th}$ entry:

$$\pi^K_{\mathbb{L}\to\mathbb{K}}(y_i) = \sqrt{K}\frac{y_i}{y_0}, \quad (5)$$

whose inverse is given as

$$\pi^K_{\mathbb{K}\to\mathbb{L}}(\mathbf{x}) = \eta^K(\mathbf{x})(\sqrt{K}, \mathbf{x}), \eta^K(\mathbf{x}) = \sqrt{\frac{K}{K - \|\mathbf{x}\|^2}}, \quad (6)$$

where $\eta^K(\mathbf{x})$ is the function defining the Lorentz factor, and $(\cdot,\cdot)$ denotes concatenation.

## Problem Definition

In this paper, we consider a dynamic graph where edges evolve over time. The emergence/disappearance of nodes will lead to the addition/deletion of a set of incident edges concurrently, which can be easily modeled with our proposed approach in a similar way. Formally, we give the definition of a dynamic graph as follows:

**Definition 1 (Dynamic Graph).** *A dynamic graph is defined as a triple $G = (V, E, T)$, where $V = \{v_1, v_2, \cdots, v_n\}$ is the node set and each node $v_i$ is associated with a feature vector $\mathbf{x}_i \in \mathbb{R}^f$. $E = \{(v_i, v_j)\}$ is the edge set, and $T = \{t_k\}$ is the timestamp set. Each edge $(v_i, v_j)$ is associated with a timestamp $t_k \in T$ representing that $v_i$ and $v_j$ interact with each other at time $t_k$.*

Without loss of generality, we consider the dynamic graph to be attributed. The timestamps on edges record every interaction among nodes, and thereby fine-grained graph dy-

namics is captured. We are interested in inferring stochastic representations in account of graph dynamics and its uncertainty in hyperbolic space. We prefer to work with the Lorentz model $\mathbb{L}^{d,K}$ of hyperbolic space owing to its numerical stability and clean closed form expressions with Lorentzian inner product (Law et al. 2019; Nickel and Kiela 2018; Chami et al. 2019). In the Lorentz model, we utilize generalized normal distributions to generate stochastic representations, where the mean and variance are innate to model the uncertainty. Formally, we define the problem of representation learning on dynamic graphs as follows:

**Definition 2 (Representation Learning on Dynamic Graphs).** *For a dynamic graph $G = (V, E, T)$, the representation learning problem in this paper is to find a map $\Phi : V \to \mathbb{L}^{d,K}$ so that, for each node $v_i$, we can infer stochastic representation $\mathbf{z}_i(t)$ at any time $t$ in hyperbolic space $\mathbb{L}^{d,K}$. The stochastic representation is drawn from a generalized normal distribution in hyperbolic space, modeling graph dynamics and its uncertainty.*

## HVGNN: Hyperbolic Variational GNN

In a nutshell, HVGNN generates stochastic representations jointly modeling graph dynamics and uncertainty. Specifically, to model the dynamics, we propose a novel Temporal Graph Neural Network (TGNN). To model the uncertainty, we introduce a hyperbolic Variational Graph AutoEncoder (VGAE), where we utilize TGNN as encoder and give task-oriented decoder for specific task, *e.g.*, link prediction and node classification. Next, we elaborate on the building blocks of HVGNN, *i.e.*, temporal GNN and hyperbolic VGAE, respectively.

### Temporal GNN

We propose the novel TGNN to model graph dynamics in two types of representation spaces: hyperbolic space $\mathbb{L}^{d,K}$ and Euclidean space $\mathbb{R}^d$. Similar to GAT (Veličković et al. 2018), the basic idea of TGNN follows the graph attention network. However, the attention itself cannot handle graph dynamics. To bridge this gap, the key is to design the following theoretically grounded time encoding approach, distinguishing nodes in time domain.

**Hyperbolic Time Encoding** We propose a novel time encoding approach to equip graph attention with the ability of modeling dynamics. Formally, we aim to learn a map $\phi_{\mathbb{L}} : T \to \mathbb{L}^{d,K}$ from a point in time domain $T$ to a vector in hyperbolic space $\mathbb{L}^{d,K}$, encoding the temporal information. The time domain $T$ is $[0, t_{max}]$ and $t_{max}$ denotes the maximum time point in the observed data.

Usually, it is the relative timespan rather than the absolute value of time that reveals critical temporal information. Thus, in learning the map $\phi_{\mathbb{L}} : T \to \mathbb{L}^{d,K}$, we are interested in the patterns between the relative timespan $|t_i - t_j|$ and the Riemannian metric of hyperbolic space, *i.e.*, Lorentzian inner product. We thereby define a Lorentzian kernel $\mathcal{K}_{\mathbb{L}} : T \times T \to \mathbb{R}$ with $\mathcal{K}_{\mathbb{L}}(t_i, t_j) = \langle \phi_{\mathbb{L}}(t_i), \phi_{\mathbb{L}}(t_j) \rangle_{\mathcal{L}}$. It is ideal that the kernel can be expressed as a function of relative timespan, namely, *translation invariance*. Formally, $\mathcal{K}_{\mathbb{L}}(t_i, t_j) = \psi_{\mathbb{L}}(t_i - t_j)$ for some function $\psi : [-t_{max}, t_{max}] \to \mathbb{R}$.

Instead of investigating $\mathcal{K}_{\mathbb{L}}(t_i, t_j)$ explicitly, we propose a two-step approach for hyperbolic time encoding as follows:
**Step** 1: *Construct a translation invariant Euclidean time encoding map.* We study its Euclidean counterpart $\mathcal{K}_{\mathbb{R}}(t_i, t_j) = \langle \phi_{\mathbb{R}}(t_i), \phi_{\mathbb{R}}(t_j) \rangle$ where $\phi_{\mathbb{R}} : T \to \mathbb{R}^d$. Note that, $\mathcal{K}_{\mathbb{R}}$ is positive semidefinite as it is defined by Euclidean inner product. Accordingly, we have $\mathcal{K}_{\mathbb{R}}$ translation invariant, $\mathcal{K}_{\mathbb{R}}(t_i, t_j) = \psi_{\mathbb{R}}(t_i - t_j)$. Thus, $\mathcal{K}_{\mathbb{R}}$ satisfies the assumption of the *Bochner's theorem* (Loomis 2013): *A translation-invariant kernel $\mathcal{K}_{\mathbb{R}}(t_i, t_j) = \psi_{\mathbb{R}}(t_i - t_j)$ is positive definite iff there exists a nonnegative measure $p(\omega)$ on $\mathbb{R}$ such that $\psi_{\mathbb{R}}(\cdot)$ is the Fourier transform of the measure.*

According to the Bochner's theorem, we have

$$\mathcal{K}_{\mathbb{R}}(t_i, t_j) = \int_{\mathbb{R}} e^{\mathbf{i}\omega(t_i - t_j)} p(\omega) d\omega = \mathbb{E}_{\omega}\left[\xi_{\omega}(t_i)\, \xi_{\omega}(t_j)^*\right], \quad (7)$$

where $\xi_{\omega}(t) = e^{\mathbf{i}\omega t}$, $\mathbf{i}$ is the imaginary unit, and $^*$ denotes the conjugate complex. As both kernel $\mathcal{K}_{\mathbb{R}}$ and its associated map $\phi_{\mathbb{R}}$ are real, we extract the real part of the expectation in Eq. (7), which can be approximated by Monte Carlo integral (Rahimi and Recht 2007), *i.e.*,

$$\mathcal{K}_{\mathbb{R}}(t_i, t_j) \approx \frac{1}{d} \sum_{i=1}^{d} [\cos(\omega_i t_i) \cos(\omega_i t_j) + \sin(\omega_i t_i) \sin(\omega_i t_j)]. \quad (8)$$

According to the formulation above, we define $\phi_{\mathbb{R}}$ below,

$$\phi_{\mathbb{R}}(t) = \sqrt{\tfrac{1}{d}} \left(\cos(\omega_1 t + \theta_1), \cos(\omega_2 t + \theta_2), \ldots, \cos(\omega_d t + \theta_d)\right), \quad (9)$$

where $(\cdot, \cdot)$ denotes the concatenation, and we will learn the $\omega$s and $\theta$s with different subscripts as model parameters.
**Step** 2: *Project Euclidean time encoding to the hyperbolic space.* We refer to $(\sqrt{K}, 0, \cdots, 0)$ as the *origin* of Lorentz model, denoted as $\mathcal{O}$. For $\mathbf{x} \in \mathbb{R}^d$, we have $(0, \mathbf{x})$ live in the tangent space of the origin $\mathcal{T}_{\mathcal{O}}\mathbb{L}^{d,K}$, since $\langle \mathcal{O}, (0, \mathbf{x}) \rangle_{\mathcal{L}} = 0$. Then, $(0, \mathbf{x})$ can be projected onto the $\mathbb{L}^{d,K}$ via the exponential map. Finally, we define the projection as

$$\pi^K_{\mathbb{R} \to \mathbb{L}}(\mathbf{x}) = \exp^K_{\mathcal{O}}((0, \mathbf{x})), \quad (10)$$

and obtain $\phi_{\mathbb{L}}(t) = \pi^K_{\mathbb{R} \to \mathbb{L}}(\phi_{\mathbb{R}}(t))$, *i.e.*,

$$\phi_{\mathbb{L}}(t) = \left(\sqrt{K} \cosh\left(\sqrt{\tfrac{1}{K}} \|\phi_{\mathbb{R}}(t)\|\right), \tfrac{\sqrt{K}}{\|\phi_{\mathbb{R}}(t)\|} \sinh\left(\sqrt{\tfrac{1}{K}} \|\phi_{\mathbb{R}}(t)\|\right) \phi_{\mathbb{R}}(t)\right). \quad (11)$$

Now, we prove the translation invariance of Lorentzian kernel $\mathcal{K}_{\mathbb{L}}(t_i, t_j)$ with the hyperbolic time encoding map $\phi_{\mathbb{L}}$.

**Theorem 1 (Translation Invariance).** *The Lorentzian kernel $\mathcal{K}_{\mathbb{L}}(t_i, t_j) = \langle \phi_{\mathbb{L}}(t_i), \phi_{\mathbb{L}}(t_j) \rangle_{\mathcal{L}}$ with the proposed $\phi_{\mathbb{L}}(\cdot)$ is translation invariant, i.e., $\mathcal{K}_{\mathbb{L}}(t_i, t_j) = \psi_{\mathbb{L}}(t_i - t_j)$.*

*Proof.* We prove the translation invariance of the Lorentzian kernel $\mathcal{K}_{\mathbb{L}}(t_i, t_j)$ by proving the existence of the function $\psi_{\mathbb{L}}$. Expanding the Lorentzian product with the definition given in Eq. (1), we have the following equation hold:

$$\langle \phi_{\mathbb{L}}(t_i), \phi_{\mathbb{L}}(t_j) \rangle_{\mathcal{L}} = A \langle \phi_{\mathbb{R}}(t_i), \phi_{\mathbb{R}}(t_j) \rangle + B, \quad (12)$$

where

$$A = -K \sinh\left(\frac{\phi_{\mathbb{R}}(t_i)}{\sqrt{K}}\right) \sinh\left(\frac{\phi_{\mathbb{R}}(t_j)}{\sqrt{K}}\right) \frac{1}{\|\phi_{\mathbb{R}}(t_i)\| \|\phi_{\mathbb{R}}(t_i)\|},$$
$$B = -K \cosh\left(\frac{\phi_{\mathbb{R}}(t_i)}{\sqrt{K}}\right) \cosh\left(\frac{\phi_{\mathbb{R}}(t_j)}{\sqrt{K}}\right), \quad (13)$$

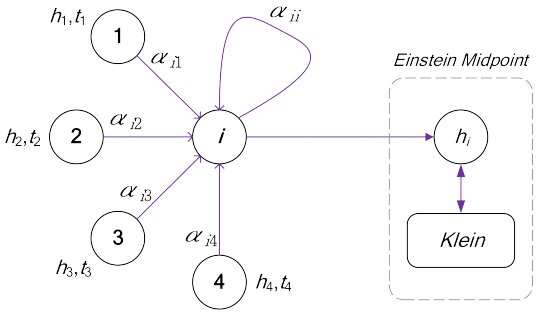



Figure 1: HYPTGA Layer. In the time-aware neighborhood, the timestamps $t_1, \cdots, t_4$ are prior to the given time $t$. Einstein midpoint is calculated with the aid of Klein model.

According to the Bochner's theorem, we have $\mathcal{K}_{\mathbb{R}}(t_i, t_j) = \langle \phi_{\mathbb{R}}(t_i), \phi_{\mathbb{R}}(t_j) \rangle = \psi_{\mathbb{R}}(t_i - t_j)$. Thus, for given $t_i$ and $t_j$,

$$\mathcal{K}_{\mathbb{L}}(t_i, t_j) = \langle \phi_{\mathbb{L}}(t_i), \phi_{\mathbb{L}}(t_j) \rangle_{\mathcal{L}} = \psi_{\mathbb{L}}(t_i - t_j),$$

where $\phi^{\mathbb{L}} = f \circ \psi^{\mathbb{R}}$ and $f(x) = Ax + B$. □

**Hyperbolic Operators** In TGNN, we generalize aggregation, addition and linear transformation from Euclidean space $\mathbb{R}^d$ to hyperbolic space $\mathbb{L}^{d,K}$. Aggregation is to calculate a weighted midpoint in Euclidean space (Gulcehre et al. 2019). The Euclidean midpoint is generalized to Einstein midpoint in hyperbolic space. Specifically, we have a vector set $\{\mathbf{x}_i | \mathbf{x}_i \in \mathbb{L}^{d,K}, i \in \Omega\}$, where $\Omega$ is the index set and each $\mathbf{x}_i$ is associated with a weight $\alpha_i$. Their Einstein midpoint $\text{AGG}^K\left(\{\alpha_i, \mathbf{x}_i\}_{i \in \Omega}\right)$ is calculated as

$$\text{AGG}^K\left(\{\alpha_i, \mathbf{x}_i\}_{i \in \Omega}\right) = \sum\nolimits_i \left[ \frac{\alpha_i \eta^K(\mathbf{x}_i)}{\sum_\ell \alpha_\ell \eta^K(\mathbf{x}_\ell)} \right] \mathbf{x}_i, \quad (14)$$

where $\eta^K(\cdot)$ is defined in Eq. (6), and $\mathbf{x}$'s are the coordinates in Klein model. Fortunately, different models of hyperbolic space are essentially the same. We can transform coordinates between Klein and Lorentz model via Eqs. (5) and (6). Additionally, we regard the weighted addition $\oplus^K$ as its special, and have $\mathbf{x}_1 \oplus^K \mathbf{x}_2 = \text{AGG}^K(\{\alpha_i, \mathbf{x}_i\}_{i \in \{1,2\}})$.

Linear transformation in hyperbolic space is realized by matrix vector multiplication, denoted as $\otimes^K$. We define $\otimes^K$ via exponential and logarithmic maps as follows:

$$\mathbf{W} \otimes^K \mathbf{x} = \exp_{\mathcal{O}}^K\left(\mathbf{W} \log_{\mathcal{O}}^K(\mathbf{x})\right), \quad (15)$$

where $\mathbf{W}$ is the weight matrix. The intuition is that we perform linear transformation in the Euclidean tangent space and then map the result back onto hyperbolic space.

**Hyperbolic Temporal Graph Attention** We denote hyperbolic TGNN in hyperbolic space $\mathbb{L}^{d,K}$ as $\text{TGNN}_{\mathbb{L}}$. We build $\text{TGNN}_{\mathbb{L}}$ by stacking its sole building block layer, *i.e.*, the *hyperbolic temporal graph attention* (HYPTGA) layer. The HYPTGA layer aims to renew node representations at time point $t$, modeling graph dynamics.

As opposed to static graph attention (*e.g.*, GAT) receiving all the neighbors' feature, we conduct aggregation in account of the interaction time between the neighbors. Specifically, for a target node $v_i$ at time $t$, we define its time-aware neighborhood $N_{i,t} = \{v_1, \cdots, v_N\}$ such that the timestamp $t_j$ of the interaction between $v_i$ and $v_j \in N_{i,t}$ is prior to $t$. HYPTGA layer takes node representations $\mathbf{h}(t)$ and timestamps of node union $\{v_i \cup N_{i,t}\}$ as the input.

In addition to graph attention, the HYPTGA layer further integrates the time encoding $\phi_{\mathbb{L}}(\cdot)$ so that we can distinguish the neighbors in time domain to model graph dynamics. We focus on relative temporal pattern across the neighbors, which remains the same between a shift. Thanks to *translation invariance* of $\phi_{\mathbb{L}}(\cdot)$, we can encode $\hat{t}_j = |t - t_j|$ for each neighbor $v_j$ since $|t_i - t_j| = |(t - t_i) - (t - t_j)|$. Then, we obtain the time-aware representation $\tilde{\mathbf{h}}_j(t)$ as follows:

$$\tilde{\mathbf{h}}_j(t) = \phi_{\mathbb{L}}(\hat{t}_j) \oplus^K \mathbf{W} \otimes^K \mathbf{h}_j(t), \quad (16)$$

where $\oplus^K$ and $\otimes^K$ are defined before. $\otimes^K$ always has a higher priority than $\oplus^K$, similar to their Euclidean counterparts. As shown in Fig. 1, target node representation $\mathbf{h}_i(t)$ is updated by aggregating time-aware representations in $N_{i,t}$,

$$\mathbf{h}_i(t) = \text{AGG}^K(\{\alpha_{ij}, \tilde{\mathbf{h}}_j(t)\}_{j \in \Omega}),\ \Omega = i \cup N_{i,t}, \quad (17)$$

where we add a self-loop for the target node with $\hat{t}_j = 0$. The attention weight $\alpha_{ij}$ is calculated by the attention function $\text{ATTN}(\cdot, \cdot)$. Naturally, we define $\text{ATTN}(\cdot, \cdot)$ by Lorentzian inner product with a nonlinear activation as follows,

$$\text{ATTN}(\tilde{\mathbf{h}}_i(t), \tilde{\mathbf{h}}_j(t)) = h(\gamma \langle \tilde{\mathbf{h}}_i(t), \tilde{\mathbf{h}}_j(t) \rangle_{\mathcal{L}} + c), \quad (18)$$

where $\gamma$ and $c$ are weight and bias, respectively. The bias is placed as Lorentzian inner product is restricted in $\mathbb{L}^{d,K}$, *i.e.*, $\langle \mathbf{x}, \mathbf{y} \rangle_{\mathcal{L}} < -K$. We define $h(\cdot)$ as $sigmoid(\cdot)$. Recalling Eq. (15), $\text{TGNN}_{\mathbb{L}}$ parameters live in the Euclidean tangent space. Such design will facilitate the model learning.

In $\text{TGNN}_{\mathbb{L}}$, for Euclidean input features, we use $\pi^K_{\mathbb{R} \to \mathbb{L}}(\cdot)$ in Eq. (10) to project them onto hyperbolic space $\mathbb{L}^{d,K}$. Meanwhile, we give its Euclidean counterpart, $\text{TGNN}_{\mathbb{R}}$. In particular, we utilize Euclidean time encoding $\phi_{\mathbb{R}}(\cdot)$, and replace hyperbolic operators of $\text{TGNN}_{\mathbb{L}}$ with Euclidean ones.

## Hyperbolic VGAE

To model the uncertainty, we introduce a Hyperbolic Variational Graph AutoEncoder (HVGAE) built upon TGNN. At the time point $t$, HVGAE infers stochastic representations $\mathbf{z}(t)$ of generalized normal distributions with a variational approach in hyperbolic space $\mathbb{L}^{d,K}$. We bridge the gaps of variational approach in hyperbolic space, which are 1) generalizing the (usual) normal distribution, and 2) defining its variational family in hyperbolic space.

**Wrapped Normal Distribution** A canonical approach for generalization is to map a usual normal distribution onto hyperbolic space. Such a probability measure is referred to as wrapped normal distribution (Mathieu et al. 2019). We derive the Probability Density Function (PDF) in $\mathbb{L}^{d,K}$:

$$\begin{aligned} &\mathcal{N}^K_{\mathbb{L}}(\mathbf{z} \mid \boldsymbol{\mu}, \text{diag}(\boldsymbol{\sigma}^2)) \\ =&F \mathcal{N}([P^K_{\boldsymbol{\mu} \to \mathcal{O}}(\mathbf{u})]_- \mid \mathbf{0}, \text{diag}(\boldsymbol{\sigma}^2)) \sinh(\sqrt{\frac{1}{K}} \|\mathbf{u}\|_{\mathcal{L}})^{1-d}, \end{aligned} \quad (19)$$

where $F = \left(\frac{\|\mathbf{u}\|_{\mathcal{L}}}{\sqrt{K}}\right)^{d-1}$, $\mathbf{u} = \log^K_{\boldsymbol{\mu}}(\mathbf{z})$ and $[\mathbf{x}]_- = \mathbf{x}_{2:d+1}$ for $\mathbf{x} \in \mathbb{R}^{d+1}$. The wrapped normal distribution owns two parameters, *i.e.*, a mean $\boldsymbol{\mu} \in \mathbb{L}^{d,K}$ in Lorentz model, and a

variance $\boldsymbol{\sigma} \in \mathbb{R}^d$ of a normal distribution $\mathcal{N}(\mathbf{0}, \mathrm{diag}(\boldsymbol{\sigma}^2))$ in Euclidean space. The advantage of wrapped normal distribution is that the PDF is *differentiable w.r.t. the parameters*, so that we can introduce a reparameterisable sampling algorithm, Algorithm 1, to enable the gradient-based learning. Specifically, for a wrapped normal sample, a) we sample a vector from usual normal distribution in Euclidean space (Line 1), b) move the vector to the mean $\boldsymbol{\mu}$ (Line 2 and 3), and c) map it onto hyperbolic space $\mathbb{L}^{d,K}$ (Line 4).

**Algorithm 1:** Reparametrisable Sampling

**Input:** parameter $\boldsymbol{\mu} \in \mathbb{L}^{d,k}$, $\boldsymbol{\sigma} \in \mathbb{R}^d$
**Output:** a sample $\mathbf{z} \sim \mathcal{N}_{\mathbb{L}}^K(\mathbf{z} \mid \boldsymbol{\mu}, \mathrm{diag}(\boldsymbol{\sigma}^2))$
1 Sample $\tilde{\mathbf{v}} \sim \mathcal{N}(\mathbf{0}, \mathrm{diag}(\boldsymbol{\sigma}^2)) \in \mathbb{R}^d$;
2 Construct $\mathbf{v} = (0, \tilde{\mathbf{v}}) \in \mathcal{T}_{\mathcal{O}}\mathbb{L}^{d,K}$;
3 Transport $\mathbf{v}$ to $\mathbf{u} = P_{\mathcal{O} \to \boldsymbol{\mu}}(\mathbf{v}) \in \mathcal{T}_{\boldsymbol{\mu}}\mathbb{L}^{d,K}$ via Eq. (4);
4 Map $\mathbf{u}$ to $\mathbf{z} = \exp_{\boldsymbol{\mu}}(\mathbf{u}) \in \mathbb{L}^{d,K}$ via Eq. (2).

**Hyperbolic Variational Family** In HVGAE, we need to define a prior distribution and the family of corresponding variational posterior in hyperbolic space. In particular, the prior is the standard distribution $\mathcal{N}_{\mathbb{L}}^K(\cdot|\mathbf{0}, \mathbf{I})$, The variational family is $\{\mathcal{N}_{\mathbb{L}}^K(\cdot|\boldsymbol{\mu}, \mathrm{diag}(\boldsymbol{\sigma}^2)) \,|\boldsymbol{\mu} \in \mathbb{L}^{d,K}, \boldsymbol{\sigma} \in \mathbb{R}^d\}$, where we propose to define the parameters as $\boldsymbol{\mu} = \mathrm{TGNN}_{\mathbb{L}}(G)$ and $\log \boldsymbol{\sigma} = \mathrm{TGNN}_{\mathbb{R}}(G)$. *The novelty is two-fold*: 1) We define the distribution parameters by the time-aware TGNN, so that the uncertainty is jointly modeled with the dynamics. 2) We introduce a reparametrisable sampling algorithm to enable the gradient-based learning.

## Overall Architecture

Thanks to reparameterisable sampling, HVGNN can be learned in an end-to-end approach with a specific learning task. HVGNN is built with an encoder-decoder framework, illustrated in Fig. 2, and the aforementioned representation learning map $\Phi$ can be defined by the encoder. In particular, we utilize TGNN encoder and a task-oriented decoder. In this paper, we provide the decoder for link prediction and node classification.

**Encoder** For node $v_i$ at time $t$, we denote the stochastic representation as $\mathbf{z}_i(t) \in \mathbb{L}^{d,K}$, summarized in $\mathbf{Z}_t$, and denote the class label as $y_i = k \in [1, C]$, summarized in $\mathcal{Y}$. With the TGNN encoder, we give the posterior as follows:

$$q(\mathbf{Z}_t \mid G) = \prod_{i=1}^{n} q\left(\mathbf{z}_i(t) \mid G\right), \tag{20}$$

where $n$ is the number of nodes in the graph. $q\left(\mathbf{z}_i(t)|G\right)$ is defined by the hyperbolic variational family above.

**Decoder** For the task of *link prediction*, we utilize the Fermi-Dirac decoder. Formally, we have

$$p(G|\mathbf{Z}_t) = \prod_{(v_i,v_j)\in E} p\left((v_i, v_j) \in E|\mathbf{z}_i(t), \mathbf{z}_j(t)\right), \tag{21}$$

whose likelihood is defined by the Fermi-Dirac function (Chami et al. 2019). For *node classification*, we utilize the hyperbolic multinomial logistic decoder, formally,

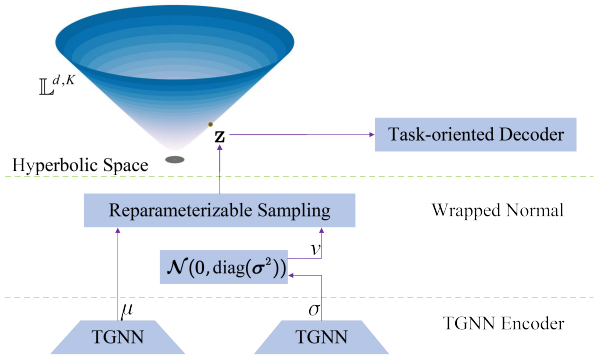



Figure 2: The overall architecture of HVGNN.

$$p(\mathcal{Y}|\mathbf{Z}_t) = \prod_{i=1}^{n} p\left(y_i = k \mid \mathbf{z}_i(t)\right), \tag{22}$$

whose likelihood is defined by the hyperbolic multinomial logistic function (Ganea, Bécigneul, and Hofmann 2018).

**Learning objective** We formulate the learning objective following the vanilla Variational AutoEncoder (Kingma and Welling 2014). The decoder is trained together with the encoder by evidence lower bound (ELBO) defined as follows:

$$\mathcal{J} = \mathbb{E}_{q(\mathbf{Z}_t|G)}[\log p(\cdot \mid \mathbf{Z}_t)] - \mathrm{KL}[q(\mathbf{Z}_t \mid G) \| p(\mathbf{Z}_t)], \tag{23}$$

where $p(\cdot \mid \mathbf{Z}_t)$ is the likelihood of corresponding decoder. $\mathrm{KL}[\cdot\|\cdot]$ is the Kullback-Leibler (KL) divergence between posterior $q(\mathbf{Z}_t \mid G)$ and prior $p(\mathbf{Z}_t) = \prod_i \mathcal{N}_{\mathbb{L}}^K(\mathbf{z}_i(t)|\mathbf{0}, \mathbf{I})$. The reparametrisable sampling enables the evaluation of the gradient of the ELBO *w.r.t.* of TGNN parameters living in Euclidean spaces. Consequently, we can make use of usual optimizer to learn the model. With the learned model, stochastic representations can be inferred inductively modeling graph dynamics and uncertainty in hyperbolic space.

# Experiments

We evaluate HVGNN by link prediction and node classification on several datasets. We repeat each experiment 10 times and report the mean with the standard deviations.

## Experimental Setups

**Datasets** We choose three real-world datasets, *i.e.*, **Reddit** (Xu et al. 2020), **Wikipedia** (Kumar, Zhang, and Leskovec 2019) and **DBLP** (Zhou et al. 2018a). In Reddit, we collect active users and their posts under subreddits, yielding a dynamic graph with $12,000$ nodes and $763,055$ timestamped edges. In Wikipedia, we collect top edited pages and active users, yielding a graph of $9,300$ nodes and $160,572$ timestamped edges. DBLP is a citation network from several CV conferences, yielding a graph of $1,909$ nodes and $8,237$ edges with publication time as timestamps.

**Comparison Method** We compare the proposed model, HVGNN, against the models that can be evaluated in both transductive and inductive settings. In particular, Euclidean models include two baselines for static graphs, *i.e.*, **GAT** (Veličković et al. 2018) and **GraphSAGE** (Hamilton, Ying, and Leskovec 2017), and a recent one for dynamic graphs,

| Model | Reddit | | Wikipedia | | DBLP | |
|---|---|---|---|---|---|---|
| | Accuracy | AP | Accuracy | AP | Accuracy | AP |
| GAT | $90.54 \pm 0.18$ | $95.05 \pm 0.21$ | $87.05 \pm 0.33$ | $94.52 \pm 0.20$ | $88.24 \pm 0.54$ | $94.71 \pm 0.28$ |
| GraphSAGE | $91.21 \pm 0.22$ | $95.12 \pm 0.16$ | $85.21 \pm 0.28$ | $93.16 \pm 0.25$ | $87.36 \pm 0.12$ | $94.08 \pm 0.35$ |
| TGAT | $92.52 \pm 0.25$ | $96.11 \pm 0.20$ | $87.65 \pm 0.24$ | $95.02 \pm 0.08$ | $89.11 \pm 0.46$ | $95.46 \pm 0.17$ |
| HGCN | $92.03 \pm 0.41$ | $95.86 \pm 0.08$ | $87.12 \pm 0.33$ | $94.67 \pm 0.38$ | $88.83 \pm 0.45$ | $95.18 \pm 0.14$ |
| $\text{TGNN}_{\mathbb{R}}$ | $92.18 \pm 0.06$ | $96.03 \pm 0.27$ | $88.11 \pm 0.47$ | $95.36 \pm 0.32$ | $89.25 \pm 0.32$ | $95.67 \pm 0.29$ |
| $\text{TGNN}_{\mathbb{L}}$ | $93.36 \pm 0.31$ | $97.25 \pm 0.07$ | $89.74 \pm 0.08$ | $96.62 \pm 0.22$ | $91.12 \pm 0.51$ | $97.12 \pm 0.36$ |
| EVGNN | $93.20 \pm 0.27$ | $97.37 \pm 0.35$ | $90.05 \pm 0.62$ | $96.85 \pm 0.39$ | $90.89 \pm 0.06$ | $97.18 \pm 0.14$ |
| **HVGNN** | $\mathbf{94.72 \pm 0.32}$ | $\mathbf{98.79 \pm 0.48}$ | $\mathbf{91.67 \pm 0.40}$ | $\mathbf{98.02 \pm 0.15}$ | $\mathbf{92.17 \pm 0.21}$ | $\mathbf{98.25 \pm 0.09}$ |

Table 1: The performance of *transductive* link prediction in terms of Accuracy and AP (%).

| Model | Reddit | | Wikipedia | | DBLP | |
|---|---|---|---|---|---|---|
| | Accuracy | AP | Accuracy | AP | Accuracy | AP |
| GAT | $88.12 \pm 0.22$ | $93.37 \pm 0.28$ | $82.14 \pm 0.31$ | $91.12 \pm 0.39$ | $84.26 \pm 0.12$ | $91.88 \pm 0.61$ |
| GraphSAGE | $87.63 \pm 0.15$ | $94.64 \pm 0.24$ | $82.26 \pm 0.42$ | $90.86 \pm 0.33$ | $83.75 \pm 0.26$ | $92.35 \pm 0.39$ |
| TGAT | $89.25 \pm 0.21$ | $95.12 \pm 0.33$ | $85.05 \pm 0.18$ | $93.51 \pm 0.27$ | $86.38 \pm 0.51$ | $94.47 \pm 0.36$ |
| HGCN | $89.07 \pm 0.29$ | $95.18 \pm 0.26$ | $84.93 \pm 0.49$ | $93.18 \pm 0.35$ | $86.41 \pm 0.29$ | $94.02 \pm 0.26$ |
| $\text{TGNN}_{\mathbb{R}}$ | $89.08 \pm 0.40$ | $94.87 \pm 0.57$ | $85.63 \pm 0.26$ | $93.95 \pm 0.23$ | $86.24 \pm 0.02$ | $94.13 \pm 0.41$ |
| $\text{TGNN}_{\mathbb{L}}$ | $90.34 \pm 0.20$ | $96.11 \pm 0.39$ | $87.05 \pm 0.38$ | $95.14 \pm 0.19$ | $88.60 \pm 0.45$ | $95.35 \pm 0.17$ |
| EVGNN | $90.27 \pm 0.03$ | $95.93 \pm 0.57$ | $87.52 \pm 0.48$ | $95.08 \pm 0.42$ | $88.48 \pm 0.10$ | $95.22 \pm 0.35$ |
| **HVGNN** | $\mathbf{91.89 \pm 0.06}$ | $\mathbf{97.67 \pm 0.20}$ | $\mathbf{89.13 \pm 0.11}$ | $\mathbf{97.15 \pm 0.45}$ | $\mathbf{90.33 \pm 0.07}$ | $\mathbf{97.36 \pm 0.13}$ |

Table 2: The performance of *inductive* link prediction in terms of Accuracy and AP (%).

*i.e.*, **TGAT** (Xu et al. 2020). Hyperbolic model includes **HGCN** (Chami et al. 2019), a recent model for static graphs. These methods generate deterministic representations and thereby cannot capture uncertainty.

**Ablation Study** We include the proposed dynamic graph models, *i.e.*, $\text{TGNN}_{\mathbb{R}}$ and $\text{TGNN}_{\mathbb{L}}$, without modeling the uncertainty as baselines. Additionally, we design the Euclidean variant of HVGNN, namely EVGNN, which models dynamic graphs with the uncertainty in Euclidean space. Specifically, we utilize two $\text{TGNN}_{\mathbb{R}}$ to parameterize usual normal distributions. EVGNN is then optimized with reparameterisable trick of the standard VAE. On the one hand, we can study the importance of modeling uncertainty by comparing HVGNN (EVGNN) against $\text{TGNN}_{\mathbb{L}}$ ($\text{TGNN}_{\mathbb{R}}$). On the other hand, we can study the effect of the representation space by comparing hyperbolic models against their Euclidean counterparts. In the experiment, we stack the corresponding attention layer twice in the models above. Refer Supplementary Material for further experimental details.

**Transductive and Inductive Settings** The transductive setting examines output representations of the nodes that have been observed in training. The inductive setting examines output representations of *unseen* nodes while training. Node representations are initialized as its raw feature. We do chronological train-validation-test split with $80\% - 5\% - 15\%$ according to the timestamps.

## Link Prediction

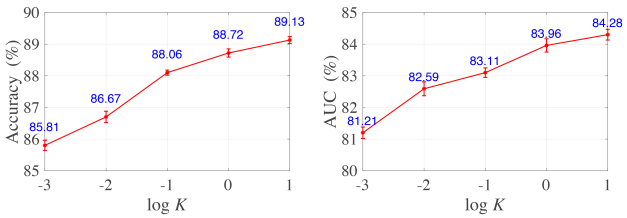
The task of link prediction is to predict the probability of two nodes being connected. For hyperbolic models, we utilize the Fermi-Dirac decoder with Lorentz inner product to compute the probability based on the learned representations. For Euclidean models, we replace the Lorentz inner product with normal inner product. The graph models are trained by minimizing the cross-entropy loss using negative sampling. We randomly sample an equal amount of negative node pairs to the positive links, and employ the and classification *Accuracy* and *Average Precision* (*AP*) as evaluation metrics. The performances under transductive and inductive settings are reported in Tables 1 and 2, respectively. The proposed model, HVGNN, consistently outperforms its competitors. For example, on Wikipedia dataset, HVGNN obtains $3\%$ and $3.64\%$ performance gains against its best competitor in transductive and inductive settings, respectively. The reason is that HVGNN models the inherent characteristics of graphs, *i.e.*, the dynamics and uncertainty, in the promising hyperbolic space. Additionally, we provide further insights though the ablation study: 1) Hyperbolic models (HVGNN and $\text{TGNN}_{\mathbb{L}}$) outperform the Euclidean counterparts (EVGNN and $\text{TGNN}_{\mathbb{R}}$). This suggests hyperbolic space is a more promising representation space well-suited for modeling real-world graphs. 2) HVGNN and EVGNN achieve better performance than $\text{TGNN}_{\mathbb{L}}$ and $\text{TGNN}_{\mathbb{R}}$, respectively. This shows that the uncertainty cannot be ignored for modeling graphs.

## Node Classification

The task of node classification is to predict the label of the node based on node representations. We utilize usual multinomial logistic loss for Euclidean models, while the hyperbolic multinomial logistic loss for hyperbolic models. We train the comparison models together with the link prediction loss above. Owing to the label imbalance on the

| Model | Transductive Setting | | | Inductive Setting | | |
|---|---|---|---|---|---|---|
| | Reddit | Wikipedia | DBLP | Reddit | Wikipedia | DBLP |
| GAT | $64.12 \pm 0.48$ | $81.54 \pm 0.80$ | $78.48 \pm 1.29$ | $62.55 \pm 0.61$ | $79.18 \pm 0.12$ | $77.03 \pm 0.55$ |
| GraphSAGE | $60.86 \pm 0.59$ | $82.05 \pm 0.72$ | $77.67 \pm 0.05$ | $59.63 \pm 0.47$ | $79.67 \pm 0.31$ | $76.12 \pm 0.19$ |
| TGAT | $64.88 \pm 0.67$ | $83.28 \pm 0.56$ | $78.73 \pm 0.12$ | $63.24 \pm 0.33$ | $80.05 \pm 0.80$ | $77.86 \pm 0.36$ |
| HGCN | $64.67 \pm 0.42$ | $82.96 \pm 0.13$ | $79.42 \pm 0.45$ | $63.98 \pm 0.16$ | $80.12 \pm 0.92$ | $77.51 \pm 0.23$ |
| $\text{TGNN}_{\mathbb{R}}$ | $64.53 \pm 0.14$ | $83.19 \pm 0.49$ | $78.79 \pm 0.72$ | $63.17 \pm 0.39$ | $79.89 \pm 0.61$ | $77.95 \pm 0.47$ |
| $\text{TGNN}_{\mathbb{L}}$ | $65.92 \pm 0.54$ | $84.55 \pm 0.96$ | $80.05 \pm 0.33$ | $64.03 \pm 0.25$ | $81.27 \pm 0.35$ | $79.16 \pm 0.29$ |
| EVGNN | $66.35 \pm 0.69$ | $84.71 \pm 0.15$ | $81.13 \pm 0.24$ | $65.18 \pm 0.61$ | $82.02 \pm 0.54$ | $80.24 \pm 0.56$ |
| **HVGNN** | $\mathbf{68.13 \pm 0.51}$ | $\mathbf{86.22 \pm 0.42}$ | $\mathbf{82.67 \pm 0.14}$ | $\mathbf{67.26 \pm 0.75}$ | $\mathbf{83.96 \pm 0.24}$ | $\mathbf{81.72 \pm 0.58}$ |

Table 3: The performance of *transductive* and *inductive* node classification in terms of AUC (%).




(a) Link Prediction (b) Node Classification

Figure 3: The effects of curvature $-1/K$ for inductive learning on Wikipedia dataset.

datasets, we employ the *area under the ROC curve* (*AUC*) as the evaluation metric. We summarize the performances under transductive and inductive settings in Table 3. As reported in Table 3, HVGNN achieves the best performance consistently. For example, on Wikipedia dataset, HVGNN outperforms its best competitor by 2.94% and 3.84% in transductive and inductive settings, respectively. Additionally, the superiority of hyperbolic space and the importance of modeling dynamics and uncertainty are supported in the experimental results.

### The Curvature of Hyperbolic Space

In HVGNN, we consider the curvature of hyperbolic space, controlled by $K$, as a hyperparameter. In fact, the hyperbolic spaces with different curvatures are essentially the same (Loomis 2013). However, owing to the limited machine precision and normalization, hyperbolic spaces with different curvatures lead to different performances (Chami et al. 2019). For instance, as shown in Figure 3, we obtain performance gain by adjusting the curvature. Motivated by this observation, instead of a fixed curvature, in HVGNN, we give the generalized formulation of a trainable curvature to improve its learning capacity.

## Related Work

### Graph Representation Learning

Graph representation learning generates vector representations for graphs, and thereby activates advances in machine learning with vector inputs, e.g., a recent strong classifier (Xia et al. 2019) whose inputs are coarse granular features. Thus, it is attracting increasing attentions, and finds itself from network alignment (Sun et al. 2019; Su et al. 2018) to text classification (Peng et al. 2019). Among graph modeling methods, GNNs (Wang, Cui, and Zhu 2016; Ma et al. 2019; Kipf and Welling 2016a) play an important role. In general, graph modeling is widely studied in static settings while models for dynamic graphs are still scant. Recently, several solutions (Zhou et al. 2018b; Xhonneux, Qu, and Tang 2020) for dynamic graphs are proposed. For example, EvolveGCN (Pareja et al. 2020) models dynamics with a sequence of snapshots. JODIE (Kumar, Zhang, and Leskovec 2019) models the node trajectories. VGRNN (Hajiramezanali et al. 2019) further models the uncertainty. To our knowledge, all prior models for dynamic graphs consider node representations in Euclidean space,

### Hyperbolic Representation Learning

Most of representation learning methods assume the representation space to be Euclidean. Actually, hyperbolic space provides an exciting alternative. It is well-suited to model hierarchical data (Krioukov et al. 2010; Papadopoulos et al. 2012). An increasing number of studies report hyperbolic model compares favorably to its Euclidean counterpart in a wide spectrum of applications, such as word embedding (Nickel and Kiela 2017; Tifrea, Becigneul, and Ganea 2019), question answering (Tay, Tuan, and Hui 2018), clustering (Monath et al. 2019), network alignment (Sun et al. 2020) and reasoning in knowledge graph (Balazevic, Allen, and Hospedales 2019). Most of existing methods generate deterministic vectors living hyperbolic space, while some recent works (Mathieu et al. 2019; Nagano et al. 2019) study the generalized normal distributions for stochastic representations. Recently, hyperbolic graph models (Ganea, Bécigneul, and Hofmann 2018; Liu, Nickel, and Kiela 2019) have been proposed where the graph is considered to be static. Distinguishing from these studies, we propose the first hyperbolic model for dynamic graphs.

## Conclusion

We have presented a novel HVGNN for dynamic graph representation learning in hyperbolic space. HVGNN captures graph dynamics and uncertainty in the stochastic representations of wrapped normal distributions in hyperbolic space. HVGNN further incorporates a reparameterisable sampling algorithm to enable its gradient-based learning. Experimental results show the superiority of HVGNN for link prediction and node classification on several real-world datasets.

## Acknowledgments

This work was supported by the National Key Research and Development Program of China under Grant 2018YFB1003804, National Natural Science Foundation of China under Grant U1936103, 61921003 and 62002007, and Fundamental Research Funds for the Central Universities 2019XD11. Philip S. Yu was supported by National Science Foundation under grants III-1763325, III-1909323 and SaTC-1930941. Jiawei Zhang was supported by National Science Foundation under grants IIS-1763365.

## Broader Impact

This paper introduced a novel research problem: dynamic graph representation learning in hyperbolic space. This paper broadens the current graph representation learning research in multiple dimensions, *e.g.*, hyperbolic representation space, dynamic graphs, and stochastic representation learning, to new stages and greatly enriches the current graph neural network studies. This paper possibly motivates further studies on representation learning in the Riemannian manifolds in a more general and elegant way. Furthermore, the model (HVGNN) proposed in this paper has transformative impacts in various real-world applications and a wide spectrum of interdisciplinary studies on graph data, such as online social computing, recommender systems, biomedical studies, and neural science. Both results and source code will be released to the public, and others who otherwise have limited access to the models can use our open-source materials in their researches or applications. We would encourage researchers to explore further applications of our approach, and also welcome the discussion on any theoretical and empirical details, and all kinds of improvements and enhancements from any research field.

## References

Balazevic, I.; Allen, C.; and Hospedales, T. 2019. Multi-relational Poincaré graph embeddings. In *Advances in NeurIPS*, 4465–4475.

Chami, I.; Ying, Z.; Ré, C.; and Leskovec, J. 2019. Hyperbolic graph convolutional neural networks. In *Advances in NeurIPS*, 4869–4880.

Chen, W.; Fang, W.; Hu, G.; and Mahoney, M. W. 2013. On the hyperbolicity of small-world and treelike random graphs. *Internet Mathematics* 9(4): 434–491.

Ganea, O.; Bécigneul, G.; and Hofmann, T. 2018. Hyperbolic neural networks. In *Advances in NeurIPS*, 5345–5355.

Grover, A.; and Leskovec, J. 2016. node2vec: Scalable feature learning for networks. In *Proceedings of KDD*, 855–864.

Gulcehre, C.; Denil, M.; Malinowski, M.; Razavi, A.; Pascanu, R.; Hermann, K. M.; Battaglia, P.; Bapst, V.; Raposo, D.; Santoro, A.; and de Freitas, N. 2019. Hyperbolic Attention Networks. In *Proceedings of ICLR*.

Hajiramezanali, E.; Hasanzadeh, A.; Duffield, N.; Narayanan, K. R.; Zhou, M.; and Qian, X. 2019. Variational Graph Recurrent Neural Networks. In *Advances in NeurIPS*, 10700–10710.

Hamilton, W.; Ying, Z.; and Leskovec, J. 2017. Inductive representation learning on large graphs. In *Advances in NeurIPS*, 1024–1034.

Hopper, C.; and Andrews, B. 2010. *The Ricci flow in Riemannian geometry*. Springer.

Kingma, D. P.; and Welling, M. 2014. Auto-Encoding Variational Bayes. In *Proceedings of ICLR*.

Kipf, T. N.; and Welling, M. 2016a. Semi-supervised classification with graph convolutional networks. In *Proceedings of ICLR*.

Kipf, T. N.; and Welling, M. 2016b. Variational graph auto-encoders. *NeurIPS Bayesian Deep Learning Workshop* .

Krioukov, D.; Papadopoulos, F.; Kitsak, M.; Vahdat, A.; and Boguná, M. 2010. Hyperbolic geometry of complex networks. *Physical Review E* 82(3): 036106.

Kumar, S.; Zhang, X.; and Leskovec, J. 2019. Predicting Dynamic Embedding Trajectory in Temporal Interaction Networks. In *Proceedings of KDD*, 1269–1278.

Law, M.; Liao, R.; Snell, J.; and Zemel, R. 2019. Lorentzian Distance Learning for Hyperbolic Representations. In *Proceedings of ICML*, 3672–3681.

Li, Y.; Tian, Y.; Zhang, J.; and Chang, Y. 2020. Learning Signed Network Embedding via Graph Attention. In *Proceedings of the AAAI*, 4772–4779.

Liu, Q.; Nickel, M.; and Kiela, D. 2019. Hyperbolic graph neural networks. In *Advances in NeurIPS*, 8228–8239.

Liu, Z.; Zhou, D.; Zhu, Y.; Gu, J.; and He, J. 2020. Towards Fine-grained Temporal Network Representation via Time-Reinforced Random Walk. In *Proceedings of AAAI*, 4973–4980.

Loomis, L. H. 2013. *Introduction to abstract harmonic analysis*. Courier Corporation.

Ma, J.; Cui, P.; Kuang, K.; Wang, X.; and Zhu, W. 2019. Disentangled Graph Convolutional Networks. In *Proceedings of ICML*, 4212–4221.

Mathieu, E.; Le Lan, C.; Maddison, C. J.; Tomioka, R.; and Teh, Y. W. 2019. Continuous Hierarchical Representations with Poincaré Variational Auto-Encoders. In *Advances in NeurIPS*, 12544–12555.

Monath, N.; Zaheer, M.; Silva, D.; McCallum, A.; and Ahmed, A. 2019. Gradient-based Hierarchical Clustering using Continuous Representations of Trees in Hyperbolic Space. In *Proceedings of KDD*, 714–722.

Nagano, Y.; Yamaguchi, S.; Fujita, Y.; and Koyama, M. 2019. A wrapped normal distribution on hyperbolic space for gradient-based learning. In *Proceedings of ICML*, 4693–4702.

Nickel, M.; and Kiela, D. 2017. Poincaré embeddings for learning hierarchical representations. In *Advances in NeurIPS*, 6338–6347.

Nickel, M.; and Kiela, D. 2018. Learning Continuous Hierarchies in the Lorentz Model of Hyperbolic Geometry. In *Proceedings of ICML*, 3779–3788.

Papadopoulos, F.; Kitsak, M.; Serrano, M. Á.; Boguná, M.; and Krioukov, D. 2012. Popularity versus similarity in growing networks. *Nature* 489(7417): 537.

Pareja, A.; Domeniconi, G.; Chen, J.; Ma, T.; Suzumura, T.; Kanezashi, H.; Kaler, T.; and Leisersen, C. E. 2020. Evolvegcn: Evolving graph convolutional networks for dynamic graphs. In *Proceedings of AAAI*.

Peng, H.; Li, J.; Gong, Q.; Wang, S.; He, L.; Li, B.; Wang, L.; and Yu, P. S. 2019. Hierarchical Taxonomy-Aware and Attentional Graph Capsule RCNNs for Large-Scale Multi-Label Text Classification. *IEEE Trans. on Knowledge and Data Engineering* .

Perozzi, B.; Al-Rfou, R.; and Skiena, S. 2014. Deepwalk: Online learning of social representations. In *Proceedings of KDD*, 701–710.

Rahimi, A.; and Recht, B. 2007. Random Features for Large-Scale Kernel Machines. In *Advances in NeurIPS*, 1177–1184.

Su, S.; Sun, L.; Zhang, Z.; Li, G.; and Qu, J. 2018. MASTER: across Multiple social networks, integrate Attribute and STructure Embedding for Reconciliation. In *Proceedings of IJCAI*, 3863–3869.

Sun, L.; Zhang, Z.; Ji, P.; Wen, J.; Su, S.; and Yu, P. S. 2019. DNA: Dynamic Social Network Alignment. In *Proceedings of IEEE BigData*, 1124–1231.

Sun, L.; Zhang, Z.; Zhang, J.; Wang, F.; Du, Y.; Su, S.; and Yu, P. S. 2020. PERFECT: A Hyperbolic Embedding for Joint Social Network Alignment. In *Proceedings of ICDM*.

Tang, J.; Qu, M.; Wang, M.; Zhang, M.; Yan, J.; and Mei, Q. 2015. Line: Large-scale information network embedding. In *Proceedings of WWW*, 1067–1077.

Tay, Y.; Tuan, L. A.; and Hui, S. C. 2018. Hyperbolic representation learning for fast and efficient neural question answering. In *Proceedings of WSDM*, 583–591.

Tifrea, A.; Becigneul, G.; and Ganea, O.-E. 2019. Poincare Glove: Hyperbolic Word Embeddings. In *Proceedings of ICLR*.

Tu, K.; Cui, P.; Wang, X.; Yu, P. S.; and Zhu, W. 2018. Deep recursive network embedding with regular equivalence. In *Proceedings of KDD*, 2357–2366.

Veličković, P.; Cucurull, G.; Casanova, A.; Romero, A.; Liò, P.; and Bengio, Y. 2018. Graph Attention Networks. In *Proceedings of ICLR*.

Wang, D.; Cui, P.; and Zhu, W. 2016. Structural deep network embedding. In *Proceedings of KDD*, 1225–1234.

Xhonneux, L.-P.; Qu, M.; and Tang, J. 2020. Continuous Graph Neural Networks. In *Proceedings of ICML*.

Xia, S.; Wang, G.; Chen, Z.; Duan, Y.; and Liu, Q. 2019. Complete Random Forest based Class Noise Filtering Learning for Improving the Generalizability of Classifiers. *IEEE Trans. on Knowledge and Data Engineering* 31(11): 2063–2078.

Xu, D.; Ruan, C.; Korpeoglu, E.; Kumar, S.; and Achan, K. 2020. Inductive representation learning on temporal graphs. In *Proceedings of ICLR*.

Zhou, D.; He, J.; Yang, H.; and Fan, W. 2018a. SPARC: Self-Paced Network Representation for Few-Shot Rare Category Characterization. In *Proceedings of KDD*, 2807–2816.

Zhou, L.; Yang, Y.; Ren, X.; Wu, F.; and Zhuang, Y. 2018b. Dynamic network embedding by modeling triadic closure process. In *Proceedings of AAAI*, 571–578.

Zhu, D.; Cui, P.; Wang, D.; and Zhu, W. 2018. Deep variational network embedding in Wasserstein space. In *Proceedings of KDD*, 2827–2836.