A Self-supervised Riemannian GNN with Time Varying Curvature for Temporal Graph Learning

Li Sun
ccesunli@ncepu.edu.cn
North China Electric Power University
Beijing, China

Junda Ye
jundaye@bupt.edu.cn
Beijing University of Posts and Telecommunications
Beijing, China

Hao Peng
penghao@act.buaa.edu.cn
Beihang University
Beijing, China

Philip S. Yu
psyu@uic.edu
University of Illinois at Chicago
Chicago, IL, USA

ABSTRACT

Representation learning on temporal graphs has drawn considerable research attention owing to its fundamental importance in a wide spectrum of real-world applications. Though a number of studies succeed in obtaining time-dependent representations, it still faces significant challenges. On the one hand, most of the existing methods restrict the embedding space with a certain curvature. However, the underlying geometry in fact shifts among the positive curvature hyperspherical, zero curvature Euclidean and negative curvature hyperbolic spaces in the evolvement over time. On the other hand, these methods usually require abundant labels to learn temporal representations, and thereby notably limit their wide use in the unlabeled graphs of the real applications. To bridge this gap, we make the first attempt to study the problem of *self-supervised temporal graph representation learning in the general Riemannian space*, supporting the time-varying curvature to shift among hyperspherical, Euclidean and hyperbolic spaces. In this paper, we present a novel self-supervised Riemannian graph neural network (SelfRGNN). Specifically, we design a curvature-varying Riemannian GNN with a theoretically grounded time encoding, and formulate a functional curvature over time to model the evolvement shifting among the positive, zero and negative curvature spaces. To enable the self-supervised learning, we propose a novel reweighting self-contrastive approach, exploring the Riemannian space itself without augmentation, and propose an edge-based self-supervised curvature learning with the Ricci curvature. Extensive experiments show the superiority of SelfRGNN, and moreover, the case study shows the time-varying curvature of temporal graph in reality.

CCS CONCEPTS

• **Computing methodologies** → **Unsupervised learning**; **Neural networks**; • **Information systems** → *Data mining*.

CIKM '22, October 17–21, 2022, Atlanta, GA, USA

ACM ISBN 978-1-4503-9236-5/22/10…$15.00
https://doi.org/10.1145/3511808.3557222

KEYWORDS

Temporal Graphs, Riemannian Geometry, Contrastive Learning

ACM Reference Format:
Li Sun, Junda Ye, Hao Peng, and Philip S. Yu. 2022. A Self-supervised Riemannian GNN with Time Varying Curvature for Temporal Graph Learning. In *Proceedings of the 31st ACM International Conference on Information and Knowledge Management (CIKM '22), October 17–21, 2022, Atlanta, GA, USA.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3511808.3557222

1 INTRODUCTION

Graph representation learning is now becoming the de facto standard when dealing with the ubiquitous graph data [11, 30, 40]. In the literature, the vast majority of graph representation learning methods [17, 43, 54] consider the static setting with the structure of graphs frozen in time. In reality, an abundance of graphs are temporal in nature and constantly evolving over time, referred to as *temporal graphs*. For instance, new interactions (edges) constantly arrive and the structure of graph changes in social networks, citation networks, e-commerce networks and the World Wide Web [53, 55]. Naive application of the static methods on temporal graphs fails to capture the temporal information, and ignoring the temporal information usually leads to questionable inference [10, 48]. Thus, the representation learning on temporal graphs has drawn increasing attention in recent years [3, 6, 22, 35].

To date, a series of temporal graph representation learning methods have been designed to output time-dependent representations [1, 16], which can be roughly divided into two main categories: discrete-time methods and continuous-time methods. The discrete-time methods frame the temporal graphs into a sequence of snapshots, and recurrent architectures are frequently employed [10, 29]. A major drawback of discrete-time methods is that the appropriate granularity for temporal discretization is often subtle. In contrast, the continuous-time methods [42, 46, 48], which directly integrate temporal information into the representation learning, are able to model the temporal graphs with a finer granularity. Despite the success of prior works, representation learning on temporal graphs still faces significant challenges.

Challenge 1: *Embedding Space Supporting Time-varying Curvature.* To the best of our knowledge, existing studies restrict the embedding space in a Riemannian space of certain curvature. In the literature, the vast majority of temporal graph models [10, 29, 42, 46, 48] work with traditional zero-curvature Euclidean space, and it

is not until very recently a few graph neural networks [39, 50] for temporal graphs are proposed in the negative-curvature hyperbolic space. In fact, rather than restricted in a certain curvature, the curvature of the underlying space varies as the temporal graph evolves over time [18, 33]. Even more challenging, the time-varying curvature actually shifts among the positive-curvature hyperspherical, zero-curvature Euclidean and negative-curvature hyperbolic spaces in the evolvement over time [28]. Therefore, it calls for a promising approach in the general Riemannian space with time-varying curvature to model the evolvement shifting among the Riemannian spaces of various curvatures.

Challenge 2: *Self-supervised Learning for Temporal graph in Riemannian space.* Most of the learning methods [19, 29, 39, 46, 50] require abundant labels to learn the time-dependent representations. Labels are usually scarce in real applications, and undoubtedly, labeling graphs is expensive, either manual annotation or paying for permission, and is even impossible to acquire because of the privacy policy. Hence, representation learning on temporal graphs without labels is more preferable, and fortunately, self-supervised learning [5, 21] has recently emerged as a principled way by exploring the similarity of data themselves without external annotations. In the literature, though the self-supervised learning for static graphs is being extensively studied [12, 32, 44], the effort for temporal graph is still limited. Recently, Tian et al. [41] propose a self-supervised learning method for temporal graph in the traditional Euclidean space. However, it cannot be applied to the general Riemannian space owing to essential distinction in geometry. That is, to the best of our knowledge, self-supervised learning for temporal graphs still remains open, especially for the general Riemannian space with time-varying curvature.

To address the aforementioned challenges, we propose to study the problem of *self-supervised temporal graph representation learning in the general Riemannian space* for the first time so as to learn temporal representations modeling the graph evolvement over Riemannian spaces of various curvatures without external guidance.

In this paper, we propose a novel Self-supervised Riemannian Graph Neural Network, referred to SelfRGNN, for the representation learning on temporal graphs. The evolvement of temporal graph is naturally described as the time-varying curvature of the embedding space in the language of Riemannian geometry. Consequently, we first propose a curvature-varying Riemannian graph neural network, in which we formulate a time encoding of arbitrary curvature to capture the temporal information, and further prove that the encoding function is translation invariant in time. Then, we formulate a functional curvature over time to model the temporal evolvement over Riemannian spaces of various curvatures from hyperspherical to Euclidean and hyperbolic spaces. To enable its self-supervised learning, we propose a Riemannian reweighted self-contrastive approach to learn temporal representations in the absence of labels. Specifically, we introduce a novel self-augmentation underpinned by the functional curvature to get rid of introducing new graphs, and formulate a reweighted contrastive objective that reweights the negative samples without sampling bias. In addition, we propose an edge-based self-supervised curvature learning with the well-defined Ricci curvature, completing the self-supervised learning of SelfRGNN.

Contributions. Noteworthy contributions are summarized below:

- *Problem.* To the best of our knowledge, we make the first attempt on formulating the representation learning problem for temporal graphs in the general Riemannian space, supporting time-varying curvature to shift among hyperspherical, Euclidean and hyperbolic spaces in the evolvement over time.
- *Methodology.* We propose the novel SelfRGNN, in which the curvature-varying Riemannian GNN and its functional curvature over time are designed to model the evolvement in the general Riemannian space of various curvatures, and the Riemannian reweighting self-contrastive approach is proposed to enable its self-supervised learning.
- *Experiments.* Extensive experiments on real-world datasets show that SelfRGNN even outperforms the state-of-the-arts supervised methods, and the case study shows the time-varying curvature of temporal graph in reality.

2 PRELIMINARIES AND PROBLEM

In this section, we first introduce the fundamentals of Riemannian geometry and the notation of curvature, and then formulate the problem of temporal graph learning in general Riemannian Space.

2.1 Preliminaries

Riemannian Geometry. It provides an elegant mathematical framework to study the geometry beyond Euclid. The fundamental object in Riemannian geometry is a smooth *manifold* $\mathcal{M}$, which generalizes the notion of the surface to higher dimensions. Each point $\boldsymbol{x} \in \mathcal{M}$ associates with a *tangent space* $\mathcal{T}_{\boldsymbol{x}}\mathcal{M}$, the first order approximation of $\mathcal{M}$ around $\boldsymbol{x}$. On the tangent space of $\boldsymbol{x}$, the *Riemannian metric*, $g_{\boldsymbol{x}}(\cdot,\cdot) : \mathcal{T}_{\boldsymbol{x}}\mathcal{M} \times \mathcal{T}_{\boldsymbol{x}}\mathcal{M} \to \mathbb{R}$, defines an inner product so that geometric notions can be induced. A *Riemannian manifold* is then defined on the smooth manifold paired with a Riemannian metric, denoted as the tuple $(\mathcal{M}, g)$. The length of the shortest walk connecting two points $\boldsymbol{x}, \boldsymbol{y}$ on the Riemannian manifold is called (geodesic) distance $d_{\mathcal{M}}(\boldsymbol{x}, \boldsymbol{y})$. Refer to mathematical materials [31, 36] for in-depth expositions.

Curvature on the Manifold. In Riemannian geometry, the *constant curvature* κ is the notion to measure how a smooth manifold deviates from being flat. There are three canonical types of constant curvature space that we can define with respect to its sign: the positively curved hyperspherical space $\mathbb{S}$ ($\kappa > 0$), the negatively curved hyperbolic space $\mathbb{H}$ ($\kappa < 0$), and the flat Euclidean space $\mathbb{E}$ ($\kappa = 0$), which is regarded as a special case.

Ricci Curvature. More specifically, for a point $\boldsymbol{x}$ on the manifold, and for each pair of linearly independent vectors $\boldsymbol{v}$ and $\boldsymbol{u}$ in $\mathcal{T}_{\boldsymbol{x}}\mathcal{M}$, the sectional curvature at $\boldsymbol{x}$ is defined on the surface spanned by the exponential map of $\boldsymbol{v}$ and $\boldsymbol{u}$, encoding the local geometry around $\boldsymbol{x}$. Given a tangent vector $\boldsymbol{v}$ at $\boldsymbol{x}$, if we average the sectional curvatures at $\boldsymbol{x}$ over a set of orthonormal vectors, we obtain the *Ricci curvature*, from which the *constant curvature* κ of the Riemannian space can be induced [31]. In this paper, we leverage the coarse Ricci curvature [27] to provide supervision signal for the proposed model.

2.2 Problem Formulation

In this paper, we consider the temporal graph that evolves over time, and utilize the interactions with temporal information (timestamped edges) to model the temporal graphs with fine granularity.

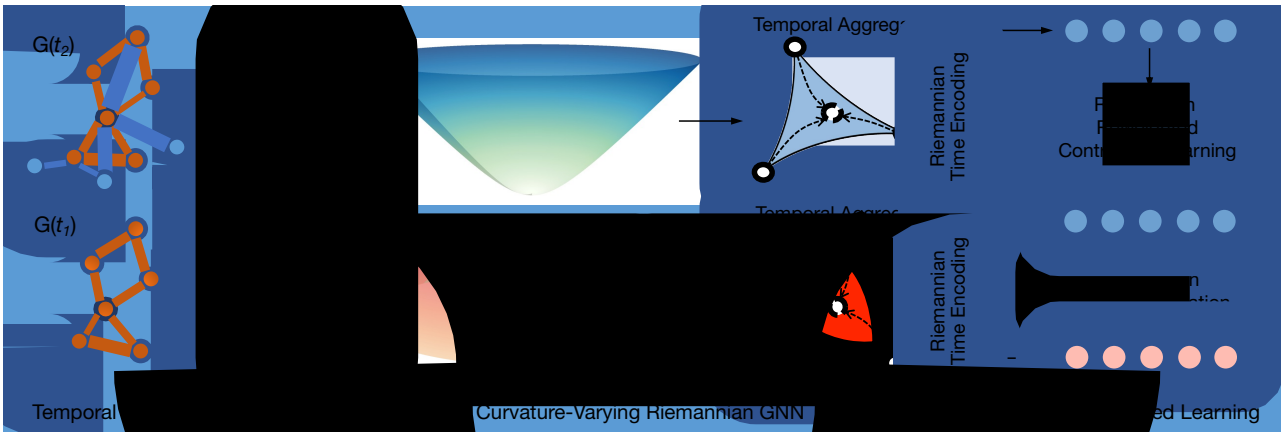


Figure 1: Overall architecture of SelfRGNN. We first design a curvature-varying RGNN shifting among the Riemannian spaces of various curvatures, e.g., positive hyperspherical (orange) and negative hyperbolic (blue) spaces, and then enable its self-supervised learning with the self-augmentation and reweighted contrastive objective without introducing new graphs.

Definition 1 (Temporal Graph). *A temporal graph, denoted as the tuple $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \boldsymbol{X}, T)$, is defined on a set of nodes $\mathcal{V} = \{v_1, v_2, \cdots, v_N\}$, a set of timestamped edges $\mathcal{E} = \{(v_i, v_j, t_l)\}$, a time domain T and a feature matrix $\boldsymbol{X} \in \mathbb{R}^{N \times F}$. The number of nodes is $|\mathcal{V}|$, and each node v_i is associated with a feature vector recorded in the corresponding row in $\boldsymbol{X}$, denoted as $\boldsymbol{x}_i \in \mathbb{R}^F$. A timestamped edge (v_i, v_j, t_l) describes the temporal interaction between nodes $v_i \in \mathcal{V}$ and $v_j \in \mathcal{V}$ at time $t_l \in T$.*

In temporal graphs, for a given node, the members of its neighborhood is time-dependent with the new arrival of edges, which is different from that of the static setting in essence. Hence, we define the neighborhood in temporal graphs as temporal neighborhood, and give the formal definition as follows:

Definition 2 (Temporal Neighborhood). *Given a time point t, a temporal neighborhood of v_i, denoted as $\mathcal{N}_t(v_i)$, is defined as a collection of the nodes linking to v_i with the elder timestamps, i.e., $\mathcal{N}_t(v_i) = \{v_j | \, (v_i, v_j, t_l) \in \mathcal{E} \wedge t_{ij} \leq t\}$.*

With the preliminaries on Riemannian geometry and definitions above, we formulate the studied problem as follows:

Problem Definition (Self-supervised Temporal Graph Representation Learning in the General Riemannian Space). *Given a temporal graph $\mathcal{G} = (\mathcal{V}, \mathcal{E}, X, T)$, the problem is to find an encoding function $\Phi : \mathcal{V} \to \mathcal{M}^{d,\kappa}$ so that, for each node v_i, we can infer the representation $\boldsymbol{h}_i(t)$ at time t in the general Riemannian space with time-varying curvature without any external guidance, encoding the evolvement of the temporal graph over time.*

In other words, we are interested in designing a novel encoding function for temporal graphs that i) can model the graph evolvement shifting among the positive-curvature hyperspherical, zero-curvature Euclidean and negative-curvature hyperbolic spaces over time, and ii) is endowed with the self-supervised learning ability.

3 METHODOLOGY

In this section, we propose a novel Self-supervised Riemannian Graph Neural Network (SelfRGNN) for the temporal graph learning in the general Riemannian space with time-varying curvature. We illustrate the overall architecture of SelfRGNN in Figure 2. As sketched in Figure 2, we first design a curvature-varying Riemannian graph neural network, modeling the evolvement shifting among the Riemannian spaces of various curvatures over time, and then propose a novel Riemannian self-supervised learning approach, obtaining temporal representations in the absence of external guidance. Next, we will elaborate on each component in the following subsections, respectively.

First of all, we introduce the Riemannian manifolds we use in this paper before we construct SelfRGNN on them.

Riemannian manifolds: We opt for the hyperboloid (Lorentz) model for hyperbolic space and the corresponding hypersphere model for hyperspherical space with the unified formalism, owing to the numerical stability and closed form expressions [4, 52]. Specifically, we have a d-dimensional manifold of curvature κ,

$$\mathcal{M}^{d,\kappa} = \begin{cases} \mathbb{S}^{d,\kappa} & = \{\boldsymbol{x} \in \mathbb{R}^{d+1} : \langle \boldsymbol{x}, \boldsymbol{x} \rangle_2 = \frac{1}{\kappa}\} & \text{for } \kappa > 0, \\ \mathbb{E}^d & = \{\boldsymbol{x} \in \mathbb{R}^d\} & \text{for } \kappa = 0, \\ \mathbb{H}^{d,\kappa} & = \{\boldsymbol{x} \in \mathbb{R}^{d+1} : \langle \boldsymbol{x}, \boldsymbol{x} \rangle_{\mathcal{L}} = \frac{1}{\kappa}\} & \text{for } \kappa < 0, \end{cases} \tag{1}$$

where $\langle \cdot, \cdot \rangle_2$ and $\langle \cdot, \cdot \rangle_{\mathcal{L}}$ denote the standard inner product and Minkowski inner product on $\mathbb{R}^{d+1}$, respectively, and the Minkowski inner product is defined as

$$\langle \boldsymbol{x}, \boldsymbol{y} \rangle_{\mathcal{L}} = \boldsymbol{x}^\top \operatorname{diag}(-1, 1, \cdots, 1) \boldsymbol{y}. \tag{2}$$

The *origin* of the manifold $(\sqrt{\frac{1}{|\kappa|}}, 0, \cdots, 0)$ is denoted as $\boldsymbol{O} \in \mathcal{M}^{d,\kappa}$.

3.1 Curvature-varying Riemannian GNN

We propose a novel curvature-aware Riemannian GNN (RGNN) with a theoretically grounded time encoding to define the encoding function Φ above, in which we formulate a functional curvature over time to model temporal evolvement shifting among Riemannian spaces of various curvatures (positive, zero and negative). This novel idea distinguishes us with the vast majority of existing studies that restrict the embedding space in a Riemannian space of certain curvature with an inductive bias given in prior.

3.1.1 Time Encoding of Arbitrary Curvature. Interaction time between nodes (timestamps) records the graph evolvement specifically. To tackle with timestamps, we propose a novel time encoding function of arbitrary curvature $\varphi^\kappa : t \to \boldsymbol{t}^\kappa \in \mathcal{M}^{d,\kappa}$ which maps a

Table 1: Summary of the operations with unified formalism.

Operation	Unified formalism in $\mathcal{M}^{d,\kappa}$
Distance Metric	$d_{\mathcal{M}}(\boldsymbol{x},\boldsymbol{y}) = \frac{1}{\sqrt{\|\kappa\|}} \cos_\kappa^{-1}(\|\kappa\|\langle \boldsymbol{x},\boldsymbol{y}\rangle_\kappa)$
Scalar Multiplication	$r \otimes_\kappa \boldsymbol{x} = exp_O^\kappa\left(r\ log_O^\kappa(\boldsymbol{x})\right)$
Matrix Multiplication	$\boldsymbol{M} \otimes_\kappa \boldsymbol{x} = exp_O^\kappa\left(\boldsymbol{M}\ log_O^\kappa(\boldsymbol{x})\right)$
Applying Functions	$f^{\otimes_\kappa}(\boldsymbol{x}) = exp_O^\kappa\left(f\left(log_O^\kappa(\boldsymbol{x})\right)\right)$

time point $t \in T$ to a vector $\boldsymbol{t}^\kappa$ in Riemannian space of curvature κ, so that the temporal information is transformed as a Riemannian feature compatible with the graph convolution. We begin the elaboration with the simple and special case, Euclidean time encoding.

Euclidean Time Encoding. In a nutshell, the Euclidean time encoding defines a generic function $\varphi^0 : t \to \boldsymbol{t}^0$, where $t \in T$ and $\boldsymbol{t}^0 \in \mathbb{R}^d$. Stemming from the design of random Fourier features, a Bochner-type (trigonometric) encoding function can be derived as

$$\varphi^0(t) = \sqrt{\frac{1}{d}}\left[\cos(\omega_1 t), \sin(\omega_1 t), \ldots, \cos(\omega_d t), \sin(\omega_d t)\right], \quad (3)$$

parameterized by the ω's [48]. According to the Bochner's Theorem, the encoding function is translation-invariant in time. Specifically, there exists a function $\psi^E(\cdot)$ on $\mathbb{R}$ so that the induced kernel $\mathcal{K}^E(t_i,t_j) = \langle \varphi^0(t_i), \varphi^0(t_j)\rangle_2$ can be expressed as $\mathcal{K}^E(t_i,t_j) = \psi^E(t_i - t_j)$, i.e., we have the following equation holds for any t_0,

$$\mathcal{K}^E(t_i,t_j) = \mathcal{K}^E(t_0 - t_i, t_0 - t_j) = \psi^E(t_i - t_j). \quad (4)$$

Riemannian Time Encoding. Based on the Euclidean encoding above, we formulate a Riemannian time encoding of arbitrary curvature with the aid of exponential map, and further *prove that the translation invariant property also holds for the proposed formulation.*

Proposition 1 (Exponential and Logarithmic Maps). *For the points on the manifold of curvature κ, $\boldsymbol{x},\boldsymbol{y} \in \mathcal{M}^{d,\kappa}$, and $\boldsymbol{v}$ in the tangent space of $\boldsymbol{x}$, $\boldsymbol{v} \in \mathcal{T}_{\boldsymbol{x}}\mathcal{M}^{d,\kappa}$, such that $\boldsymbol{x} \neq \boldsymbol{y}$ and $\boldsymbol{v} \neq \boldsymbol{0}$, the exponential map $exp_{\boldsymbol{x}}(\boldsymbol{v}) : \mathcal{T}_{\boldsymbol{x}}\mathcal{M}^{d,\kappa} \to \mathcal{M}^{d,\kappa}$ at $\boldsymbol{x}$ projects the vector $\boldsymbol{v}$ of its tangent space onto the manifold $\mathcal{M}^{d,\kappa}$,*

$$exp_{\boldsymbol{x}}^\kappa(\boldsymbol{v}) = \cos_\kappa\left(\sqrt{|\kappa|}\|\boldsymbol{v}\|_\kappa\right)\boldsymbol{x} + \sin_\kappa\left(\sqrt{|\kappa|}\|\boldsymbol{v}\|_\kappa\right)\frac{\boldsymbol{v}}{\sqrt{|\kappa|}\|\boldsymbol{v}\|_\kappa}.$$

The logarithmic map $log_{\boldsymbol{x}}^\kappa(\boldsymbol{y}) : \mathcal{M}^{d,\kappa} \to \mathcal{T}_{\boldsymbol{x}}\mathcal{M}^{d,\kappa}$ at $\boldsymbol{x}$ projects the vector $\mathbf{y} \in \mathcal{M}$ back to the tangent space $\mathcal{T}_{\boldsymbol{x}}\mathcal{M}^{d,\kappa}$,

$$log_{\boldsymbol{x}}^\kappa(\boldsymbol{y}) = \frac{\cos_\kappa^{-1}(\kappa\langle \boldsymbol{x},\boldsymbol{y}\rangle_\kappa)}{\sin_\kappa\left(\cos_\kappa^{-1}(\kappa\langle \boldsymbol{x},\boldsymbol{y}\rangle_\kappa)\right)}\left(\boldsymbol{y} - \kappa\langle \boldsymbol{x},\boldsymbol{y}\rangle_\kappa \boldsymbol{x}\right)$$

Remarks: We summarize all the necessary operations for this paper in Table 1 with the curvature-aware definition of trigonometrics, e.g., $\cos_\kappa(\cdot) = \cosh(\cdot)$ if $\kappa < 0$ and $\cos_\kappa(\cdot) = \cos(\cdot)$ if $\kappa > 0$.

We derive the Riemannian time encoding of arbitrary curvature as follows. Specifically, we first augment $\boldsymbol{t}^0$ as $\bar{\boldsymbol{t}^0} = [0\|\boldsymbol{t}^0]$, where $[\cdot\|\cdot]$ denotes the concatenation of vectors, and it is easy to check that the augmented encoding resides in the tangent space of the origin O of the Riemannian manifold, i.e., $\langle O, \bar{\boldsymbol{t}^0}\rangle_\kappa = 0$ holds for any $\boldsymbol{t}^0$. Then, we project the tangent vector $\bar{\boldsymbol{t}^0}$ via the exponential map at the origin $exp_O^\kappa(\cdot)$ to obtain $\boldsymbol{t}^\kappa$ at Riemannian manifold of curvature κ, yielding the Riemannian time encoding as follows:

$$\boldsymbol{t}^\kappa = \left[\frac{\cos_\kappa\left(\sqrt{|\kappa|}\|\bar{\boldsymbol{t}^0}\|_\kappa\right)}{\sqrt{|\kappa|}}, \frac{\sin_\kappa\left(\sqrt{|\kappa|}\|\bar{\boldsymbol{t}^0}\|_\kappa\right)}{\sqrt{|\kappa|}\|\bar{\boldsymbol{t}^0}\|_\kappa}\boldsymbol{t}^0\right], \quad (5)$$

where $\boldsymbol{t}^0$ is the Euclidean encoding. $\|\bar{\boldsymbol{t}^0}\|_\kappa = \sqrt{\langle [0,\boldsymbol{t}^0],[0,\boldsymbol{t}^0]\rangle_{\mathcal{L}}} = \|\boldsymbol{t}^0\|_2$ when $\kappa < 0$, while $\|\bar{\boldsymbol{t}^0}\|_\kappa = \sqrt{\langle [0,\boldsymbol{t}^0],[0,\boldsymbol{t}^0]\rangle_2} = \|\boldsymbol{t}^0\|_2$ when $\kappa > 0$. Note that, $\|\bar{\boldsymbol{t}^0}\|_\kappa = \|\boldsymbol{t}^0\|_2 = 1$. Therefore, we give the unified formulation of the time encoding function as follows

$$\varphi^\kappa(t) = \left[\frac{\cos_\kappa\left(\sqrt{|\kappa|}\right)}{\sqrt{|\kappa|}}, \frac{\sin_\kappa\left(\sqrt{|\kappa|}\right)}{\sqrt{|\kappa|}}\varphi^0(t)\right], \quad (6)$$

for any $t \in T$, where $\varphi^0(\cdot)$ is defined in Eq. (3).

Moreover, we prove that the Riemannian time encoding is translation invariant in time as its Euclidean counterpart.

Proposition 2 (Translation Invariant Property of Riemannian Time Encoding). *Given the Riemannian time encoding of arbitrary curvature $\varphi^\kappa(\cdot)$, the induced Riemannian kernel $\mathcal{K}^R(t_i,t_j) = \langle \varphi^\kappa(t_i), \varphi^\kappa(t_j)\rangle_\kappa$ is translation invariant over time in domain T, i.e, there exists a function $\psi^R(\cdot)$ so that*

$$\mathcal{K}^R(t_i,t_j) = \psi^R(t_i - t_j).$$

Proof. We provide the idea and key equations only due to the limit of space. We prove the translation invariant property of the induced Riemannian kernel by proving the existence of the function $\psi^R(\cdot)$ regardless of the sign of κ. Specifically, we have

$$\begin{aligned}\mathcal{K}^R(t_i,t_j) &= \langle \boldsymbol{t}_i^\kappa, \boldsymbol{t}_j^\kappa\rangle_\kappa \\ &= -\frac{\left(\sin_\kappa\left(\sqrt{|\kappa|}\right)\right)^2}{|\kappa|}\langle \boldsymbol{t}_i^0, \boldsymbol{t}_j^0\rangle_\kappa + \frac{\left(\cos_\kappa\left(\sqrt{|\kappa|}\right)\right)^2}{|\kappa|} \\ &= -\frac{\left(\sin_\kappa\left(\sqrt{|\kappa|}\right)\right)^2}{|\kappa|}\mathcal{K}^E(t_i,t_j) + \frac{\left(\cos_\kappa\left(\sqrt{|\kappa|}\right)\right)^2}{|\kappa|} \\ &= \psi^R(t_i - t_j),\end{aligned} \quad (7)$$

and $\psi^R = g \circ \psi^E$. That is, for arbitrary κ, we have $\mathcal{K}^R(t_i,t_j) = \psi^R(t_i - t_j)$ and $\psi^R = g \circ \psi^E$, where $g(x) = Ax + B$,

$$A = \frac{\left(\sin_\kappa\left(\sqrt{|\kappa|}\right)\right)^2}{|\kappa|}, \quad B = \frac{\left(\cos_\kappa\left(\sqrt{|\kappa|}\right)\right)^2}{|\kappa|}. \quad (8)$$

□

3.1.2 Riemannian Temporal Attention Layer. We propose a Riemannian temporal attention layer, which is the building block layer of $\mathcal{R}$GNN to update temporal representations $\boldsymbol{h}_i(t)$ in the general Riemannian space. As opposed to static graph convolution receiving messages of all members in the neighborhood, we conduct temporal aggregation on the *temporal neighbors* (Definition 2) $\mathcal{N}_t(v_i)$ at time t in the following two steps.

Step One. *We build the temporal message with the stereographic projection in the Riemannian manifold.* For a neighbor node v_j linked at time t_l, its temporal message is built by the representation of nodes v_j and time encoding $\varphi^\kappa(t_l)$ in the Riemannian manifold. However, operating vectors in the Riemannian manifold is nontrivial, and concatenation is generally illegal in the Riemannian manifold. Fortunately, addition is well-defined in the gyrovector spaces $\oplus_G$ with the elegant non-associative algebraic formalism:

$$\boldsymbol{x} \oplus_G \boldsymbol{y} = \frac{\left(1 - 2\kappa \boldsymbol{x}^T \boldsymbol{y} - \kappa \|\boldsymbol{y}\|^2\right) \boldsymbol{x} + \left(1 + \kappa \|\boldsymbol{x}\|^2\right) \boldsymbol{y}}{1 - 2\kappa \boldsymbol{x}^T \boldsymbol{y} + \kappa^2 \|\boldsymbol{x}\|^2 \|\boldsymbol{y}\|^2}. \quad (9)$$

The mappings between the gyrovector space and the Riemannian manifold of Eq. (1) is done via the stereographic projection $SP(\cdot)$ and its inverse as follows:

$$SP(\boldsymbol{x}) = \frac{1}{1 + \sqrt{\kappa} \boldsymbol{x}_{[1]}} \boldsymbol{x}_{[2:d+1]},\ SP^{-1}(\boldsymbol{x}') = \left[\frac{1}{\sqrt{\kappa}} \left(\lambda^{\kappa}_{\boldsymbol{x}'} - 1\right), \lambda^{\kappa}_{\boldsymbol{x}} \boldsymbol{x}' \right], \quad (10)$$

where $\lambda^{\kappa}_{\boldsymbol{x}} = \frac{2}{1+\kappa\|\boldsymbol{x}\|_2^2}$, and $\boldsymbol{x}'$ is the corresponding point of $\boldsymbol{x}$ in the gyrovector space. Thus, we have the addition in the general Riemannian manifold as follows:

$$\boldsymbol{x} \oplus_{\kappa} \boldsymbol{y} = SP^{-1}\left(SP\left(\boldsymbol{x}\right) \oplus_G SP\left(\boldsymbol{y}\right)\right), \quad (11)$$

and formulate the temporal message from v_j as follows:

$$\boldsymbol{m}_j(t) = \left(\boldsymbol{W}_1 \otimes_{\kappa} \boldsymbol{h}_j(t)\right) \oplus_{\kappa} \left(\boldsymbol{W}_2 \otimes_{\kappa} \varphi^{\kappa}\left(t_l\right)\right), \quad (12)$$

where $\boldsymbol{W}_1$ and $\boldsymbol{W}_2$ are parameter matrices. Owing to *the translation invariant with respect to time*, we alternatively set $\bar{t}_l = t - t_l$ in practice as $\mathcal{K}^R(\bar{t}_i, \bar{t}_2) = \mathcal{K}^R(t_1, t_2)$ for any t_1, t_2 in the time domain.

Step Two. *We perform the aggregation with the attention mechanism in the Riemannian space.* As the importance of neighbor nodes is usually different, we introduce the attentional aggregation in account of the importance of different neighbors. Specifically, we first lift the Riemannian temporal message to the tangent space via logarithmic map log^{κ}_{O} and model the importance parameterized by $\boldsymbol{\theta}$ and $\mathbf{W}_3$ as follows:

$$ATT^{\kappa}(\boldsymbol{m}_i(t), \boldsymbol{m}_j(t)) = \sigma\left(\boldsymbol{\theta}^{\top} \left[\boldsymbol{W}_3 log^{\kappa}_{O}(\boldsymbol{m}_i(t)), \boldsymbol{W}_3 log^{\kappa}_{O}(\boldsymbol{m}_j(t))\right]\right), \quad (13)$$

where $\sigma(\cdot)$ denotes the sigmoid activation. Then, we compute the attention weight over the temporal neighborhood via softmax:

$$\alpha_{ij} = e^{ATT^{\kappa}(\boldsymbol{m}_i(t), \boldsymbol{m}_j(t))} / \sum_{v_n \in \mathcal{N}_i(t)} e^{ATT^{\kappa}(\boldsymbol{m}_i(t), \boldsymbol{m}_n(t))}. \quad (14)$$

Finally, we update $\boldsymbol{h}_i(t)$ by performing the attentional aggregation with the aid of the tangent space, i.e.,

$$AGG^{\kappa}\left(\{\boldsymbol{m}_j(t)\}\right) = \delta^{\otimes_{\kappa}}\left(exp^{\kappa}_{\boldsymbol{h}_i(t)}\left(\sum_{j \in \mathcal{N}_t(v_i)} \alpha_{ij}\ log^{\kappa}_{\boldsymbol{h}_i(t)}\left(\boldsymbol{m}_j(t)\right)\right)\right), \quad (15)$$

where δ is the applied nonlinearity in the Riemannian space.

3.1.3 Functional Curvature in General Riemannian Space. We propose a novel functional curvature over time, a key ingredient of our $\mathcal{R}$GNN. In this way, we can model the evolvement shifting among the Riemannian space of various curvatures over time, and distinguishes us against the studies restricting in a certain curvature with the inductive bias given in prior.

In the Riemannian geometry, the evolvement of temporal graph is naturally described as the time-varying curvature of the embedding space. Specifically, we need to figure out a curvature function over time $f : t \to \kappa$. In practice, we first perform $\varphi^0(t)$ to obtain a time encoding $\boldsymbol{t}^0$, and then feed the encoding vector into a neural network, CurNN. The CurNN is built with an MLP followed by a bilinear output layer to obtain the constant curvature of any sign,

$$CurNN(t) = MLP\left(\varphi^0(t)\right)^{\top} \boldsymbol{W}_4 MLP\left(\varphi^0(t)\right). \quad (16)$$

With the functional curvature in Eq. (16), $\mathcal{R}$GNN is able model the graph evolvement shifting among hyperspherical, Euclidean and hyperbolic spaces over time.

3.2 Riemannian Self-supervised Learning

To enable the self-supervised learning, we propose a novel Riemannian self-supervised learning approach. The novelty lies in that we explore the rich information in the Riemannian space of the temporal graph itself, getting rid of the effort for data augmentation. The self-supervised learning tasks are dual, i.e., the temporal representation and the curvature of the Riemannian Space. To this end, we propose a Riemannian reweighted self-contrastive learning and an edge-based self-supervised curvature learning for the temporal graph, respectively.

3.2.1 Reweighted Self-Contrastive Learning in the Riemannian Space. Contrastive learning explores the semantic similarity of data themselves, and learn the representations by contrasting positive and negative samples [34, 41]. Consider that a latent semantic class $c \in C$ is assigned to each observation x over $\mathcal{X}$ via $h : \mathcal{X} \to C$. Given an observation x, if x' and x share the same semantic class, x' is said to be a positive sample whose conditional distribution is given as $p^+(x') = p(x' \mid h(x') = h(x))$, while a negative sample is drawn from $p^-(x') = p(x' \mid h(x') \neq h(x))$. We cannot access to the sampling distributions p^+ and p^- in practice. The main ingredients of a contrastive learning framework are: i) proxies of p^+ of a given node v_i, and ii) a loss function discriminating positive and negative samples. Unfortunately, both of them are challenging in the context of temporal graphs.

Riemannian Self-augmentation. For the first challenge (obtaining p^+), data augmentation is usually performed and thereby different views are constructed for contrast. In computer vision, the augmentation can be easily given by semantic preserving transformations, e.g., cropping and rotating [5]. However, the analog is not obvious for graphs, and the study [12] shows that different augmentations (i.e., node dropping, edge perturbation) behave differently according to the distributions of the underlying graphs.

To address this challenge, we propose a novel Riemannian self-augmentation, which leverages the functional curvature to augment auxiliary views. In this way, we obtain p^+ without introducing a new graph, instead of struggling in defining augmented graphs as prior works [12, 32, 44]. Specifically, given the α view at time t_1, the proposed self-augmentation aims to generate its β view for the contrastive learning. The temporal representations of α view are obtained via $\mathcal{R}$GNN. Alternatively, we can employ another time point t_2 as a reference and infer the temporal representations of β view at t_1 based on corresponding curvatures. Thanks to *the proposed functional curvature over time*, we can obtain the curvatures in the evolvement of temporal graph via Eq. (16), and generate the β view with a Riemannian projection as follows:

$$RiemannianProj_{t_2 \to t_1}(\cdot) = exp_O^{CurNN(t_1)}\left(log_O^{CurNN(t_2)}(\cdot)\right). \quad (17)$$

Riemannian Reweighting. For the second challenge (constrastive loss), different formulations of constrastive loss are proposed. However, in practice, the ideal negative sampling distribution p^- is replaced by the data distribution $p(x)$ over $\mathcal{X}$ (*sampling bias*), since labels cannot be accessed in the self-supervised learning. Additionally, the "negative" samples behave uniformly in the constrastive objective (*hardness unawareness*). The phenomena are formalized in Robinson et al. [34], however, its solution cannot be applied in the Riemannian space owing to the essential distinction in geometry.

To bridge this gap, we propose a novel Riemannian reweighting contrastive loss to i) get rid of the sampling bias as well as ii) select negative samples in account of the hardness. Specifically, we first confront the bias incurred by the absence of ideal p^- with tractable distributions. We decompose the data distribution as $p(x') = \tau^+ p^+(x') + \tau^- p^-(x')$, where $\tau^+ = p(h(x') = h(x))$ is the class prior of x's semantic class and can be estimated from data in practice [14], and thus p^- is yielded as

$$p^-(x') = \left(p(x') - \tau^+ p^+(x')\right)/\tau^-, \tag{18}$$

with two tractable distributions. Note that we have samples from p and p^+ is given via the self-augmentation above. Second, we introduce a probability q_ξ^- in Riemannian space to select negative samples. A hard negative sample is an x^- whose semantic class is different from the x but the representation is similar to x, and thus $q_\xi^-(x^-) = q_\xi(x^- \mid h(x) \neq h(x^-))$ is defined as

$$q_\xi^-(x^-) \propto e^{\xi s_{\mathcal{M}}(x,x^-)} \cdot p(x^-), \tag{19}$$

where $s_{\mathcal{M}}$ is a score function to output the similarity, and $\xi > 0$ is the parameter to upweight the hardness. The intuition is that a hard negative with similar representation in Riemannian space has a larger probability of getting sampled. With the prior of p, the negative sampling is *essentially reweighted by the likelihood of an exponential term*, which is the Bayesian interpretation of Eq. (19).

Obviously, it further requires a score function to contrast between positive and negative samples. Defining the score function is nontrivial as the existing Euclidean functions cannot be used in the Riemannian space, and temporal information needs to be considered for representation learning on temporal graphs. Thanks to the *translation invariant property* of the proposed time encoding of arbitrary curvature, we propose a novel score function as follows:

$$s_{\mathcal{M}}(\boldsymbol{h}_i(t), \boldsymbol{h}_i(t)) = \mathcal{K}(t_i, t_j) d_{\mathcal{M}}(\boldsymbol{h}_i(t), \boldsymbol{h}_i(t)), \tag{20}$$

which means that the samples are discriminated by the distance in the manifold penalized by the relative relationship in time domain.

3.2.2 Edge-based Self-supervised Curvature Learning. We propose to utilize the Ricci curvature on the edges to supervise the functional curvature of the graph. The (coarse) Ricci curvature κ_{ij} on edge (v_i, v_j) is defined by comparing the Wasserstein distance $W(m_i^\lambda, m_j^\lambda)$ to the geodesic distance $d_{\mathcal{M}}(\boldsymbol{h}_i(t), \boldsymbol{h}_j(t))$ on the manifold [27], where m_i^α is a probability measure around node v_i, i.e.,

$$\kappa_{ij} = 1 - W(m_i^\lambda, m_j^\lambda)/d_{\mathcal{M}}(\boldsymbol{h}_i(t), \boldsymbol{h}_j(t)). \tag{21}$$

Given a node v_i with the temporal neighborhood $\mathcal{N}_t(v_i)$, the probability measure m_i^λ is defined as

$$m_i^\lambda(v) = \begin{cases} \lambda & \text{if } v = v_i \\ (1-\lambda)/K & \text{if } v \in \mathcal{N}_t(v_i), \end{cases} \tag{22}$$

and otherwise, $m_i^\lambda(v) = 0$, where K is the number of the nodes in the neighborhood, and α is the parameter to keep probability mass of α at node v_i itself, which is set to 0.5 in practice according to the study of Ye et al. [51]. For a given time point t, the curvature of the graph is induced from the Ricci curvatures [31], and we employ a GRU to mimic the mapping. Concretely, we first pair the Ricci curvature of an edge κ_{ij} with its time encoding $\boldsymbol{t}_j^0$ to incorporate the temporal information. Then, we feed the augmented $[\kappa_{ij} || \boldsymbol{t}_j^0]$ into the GRU unit in chronological order, whose output layer is replaced by a bilinear one to obtain the graph curvature $\hat{\kappa}$ of any sign. $\hat{\kappa}$ is presented as the supervision signal to the graph curvature κ given by function of $CurNN(t)$, and thus we have the objective of self-supervised curvature learning, i.e., $\mathcal{L}_{curvature} = \sum_t |\kappa - \hat{\kappa}|$.

Algorithm 1: The Self-supervised Learning of SelfRGNN

Input: A temporal graph $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \boldsymbol{X}, T)$, weighting coefficient w.
Output: The parameters of SelfRGNN.
while *not converging* **do**
  // The α view:
  Estimate curvature at the time t_1 via Eqs. (3) and (16);
  $\boldsymbol{h}(t_1) \leftarrow CurvatureVaryingRGNN$ at t_1;
  $\boldsymbol{h}(t)_\alpha \leftarrow \boldsymbol{h}(t_1)$;
  // The β view from Riemannian self-augmentation:
  Estimate curvature at the time t_2 via Eqs. (3) and (16);
  $\boldsymbol{h}(t_2) \leftarrow CurvatureVaryingRGNN$ at t_2;
  $\boldsymbol{h}(t)_\beta \leftarrow RiemannianProj_{t_2 \to t_1}(\boldsymbol{h}(t_2))$;
  // Riemannian self-supervised learning objective:
  for *temporal representations in α and β views* **do**
    Contrast via the score function in Eq. (20);
    Calculate Riemannian reweighting contrastive loss $\mathcal{L}_{(\alpha,\beta)}$ and $\mathcal{L}_{(\beta,\alpha)}$ via Eqs. (23)-(25);
  $\hat{\kappa} \leftarrow GRU([\kappa_{ij} || \boldsymbol{t}_j^0])$;
  // Update neural network parameters:
  Calculate the gradients of the overall objective:
  $$\nabla\left(\mathcal{L}_{contrast} + w\mathcal{L}_{curvature}\right).$$

3.2.3 Self-supervised Learning Objective. First, we instantiate the formulation of Riemannian reweighted self-contrastive learning. We start with defining $\mathcal{L}_{(\alpha,\beta)}$ that contrasts with $\boldsymbol{h}(t)_\beta^+ \sim p^+$ and $\{\boldsymbol{h}_i(t)_\beta^-\}_{i=1}^N \sim p^-$ from the self-augmented β view as follows:

$$\mathbb{E}_{\boldsymbol{h}(t)_\alpha, \boldsymbol{h}(t)_\beta^+}\left[s_{\mathcal{M}}(\boldsymbol{h}(t)_\alpha, \boldsymbol{h}(t)_\beta^+) - \mathbb{E}_{\boldsymbol{h}_i(t)_\beta^-}\left[log \sum_{i=1}^N e^{\left(s_{\mathcal{M}}(\boldsymbol{h}(t)_\alpha, \boldsymbol{h}_i(t)_\beta^-)\right)}\right]\right], \tag{23}$$

and the expectation term of $\mathbb{E}_{\boldsymbol{h}(t)_\beta^-}$ is replaced by the Riemannian reweighting accordingly,

$$\frac{1}{\tau^- Z_\xi}\left(\mathbb{E}_{\boldsymbol{h}_i(t)_\beta^-}\left[e^{s_{\mathcal{M}}(\boldsymbol{h}(t)_\alpha, \boldsymbol{h}_i(t)_\beta^-)}\right] - \frac{\tau^+}{Z_\xi^-}\mathbb{E}_{\boldsymbol{h}(t)_\beta^+}\left[e^{s_{\mathcal{M}}(\boldsymbol{h}(t)_\alpha, \boldsymbol{h}(t)_\beta^+)}\right]\right), \tag{24}$$

where $q_\xi^+(\boldsymbol{h}(t)_\beta^-) \propto e^{\xi s_{\mathcal{M}}(\boldsymbol{h}(t)_\alpha, \boldsymbol{h}(t)_\beta^+)} \cdot p^+(\boldsymbol{h}(t)_\beta^-)$, and the factors

$$Z_\xi = \mathbb{E}_{\boldsymbol{h}_i(t)_\beta^-}\left[e^{\xi s_{\mathcal{M}}(\boldsymbol{h}(t)_\alpha, \boldsymbol{h}_i(t)_\beta^-)}\right],\ Z_\xi^+ = \mathbb{E}_{\boldsymbol{h}(t)_\beta^+}\left[e^{\xi s_{\mathcal{M}}(\boldsymbol{h}(t)_\alpha, \boldsymbol{h}(t)_\beta^+)}\right], \tag{25}$$

are given to normalize the probability mass. Note that, if we set hardness $\xi = 0$, the formulation in Eqs. (23)-(25) degenerates into a typical InfoNCE loss treating the samples uniformly, and in fact it is easy to check the following proposition holds.

Proposition 3 (Riemannian Reweighting Contrastive Loss). *The proposed formulation in Eqs. (23)-(25) is equivalent to InfoNCE objective formulated in [13] if class prior is omitted, $\tau^+ = 0$.*

That is, *we generalize the formulation of InfoNCE in the Riemannian space with reweighting*. The α view is contrasted with the β view, and vice versa. We have $\mathcal{L}_{contrast} = -\mathcal{L}_{(\alpha,\beta)} - \mathcal{L}_{(\beta,\alpha)}$.

Table 2: The summary of AUC (%) for node classification on Wikipedia, MOOC and Cora datasets. The highest scores are in bold, and the second *underlined*.

		Model	Wiki	MOOC	Cora
Supervised	$\mathbb{E}$	EvolveGCN	72.33(0.6)	65.35(0.1)	76.10(0.3)
		VGRNN	80.15(0.1)	71.02(0.2)	82.05(0.2)
		DyRep	79.24(0.2)	72.67(0.0)	82.89(0.1)
		TGAT	83.69(0.7)	69.46(0.4)	85.27(0.2)
		CAWNet	86.77(0.3)	68.77(0.4)	88.95(0.7)
	$\mathbb{H}$	HVGNN	86.22(0.2)	73.90(0.3)	*89.48* (0.1)
		HTGN	85.08(0.5)	*75.12* (0.1)	87.22(1.0)
Self	$\mathbb{E}$	DDGCL	*89.32* (0.5)	74.54(0.2)	87.67(0.3)
		SelfRGNN	**93.64**(0.)	**81.28**(0.1)	**94.06**(0.2)
		(Gain)	+4.32%	+6.16%	+5.58%

Finally, incorporating the objective of curvature learning, we have the overall objective for Riemannian self-supervised learning,

$$\mathcal{L}_{self} = \mathcal{L}_{contrast} + w\mathcal{L}_{curvature}, \tag{26}$$

where w is a weighting coefficient. We summarize the self-supervised learning of the proposed SelfRGNN in Algorithm 1, whose computational complexity is $O(N_e(|\mathcal{E}| + N_w|\mathcal{V}|))$, where N_e and N_w are the numbers of epochs and reweighted samples, respectively. Note that, the complexity order of SelfRGNN is same as that of self-supervised graph methods in Euclidean space [12, 41], but *we support time-varying curvature to model the evolvement shifting among hyperspherical, Euclidean and hyperbolic spaces over time.*

4 EXPERIMENT

In this section, we conduct extensive experiments on a variety of datasets, aiming to answer the following research questions (*RQs*):

- ***RQ1***: How does the proposed SelfRGNN perform?
- ***RQ2***: How does each component contributes to the success of the proposed SelfRGNN?
- ***RQ3***: How does the curvature evolve over time?

4.1 Experimental Setups

4.1.1 Datasets. We conduct extensive experiments on a diverse set of benchmark temporal graphs including **Wikipedia**, **MOOC** and **Social** of Xu et al. [48], and **Cora** and **Physics** of Hajiramezanali et al. [10]. We adopt the same chronological data split with 70% for training, and 15% for validation and testing over all the datasets.

4.1.2 Baselines. To evaluate the performance of SelfRGNN, we choose several state-of-the-arts baselines. We only consider the models for temporal graphs as we are interested in the representation learning on temporal graphs in this study.

Euclidean Model: *For the supervised models*, we compare with the strong baselines including **VGRNN** [10], **EvolveGCN** [29], **DyRep** [42], **TGAT** [48] and the recent **CAWNet** [46]. *For the self-supervised model*, we include **DDGCL** [41], a recent contrastive learning method for temporal graphs.

Riemannian Model: *For the supervised models*, we compare with the recent **HTGN** [50] and **HVGNN** [39], and both of them are in the Riemannian space of negative curvature (hyperbolic space). The proposed SelfRGNN is the first Riemannian model with time-varying curvature, to the best of our knowledge. *For the self-supervised model*, there is few work in the literature, and we also fill this gap in SelfRGNN.

4.1.3 Evaluation Tasks. Both node classification and link prediction are utilized as evaluation tasks.

Node Classification: We evaluate the performance on Wikipedia, MOOC, and Cora. The node label of Wikipedia and MOOC is given following the study [41], and the nodes of Cora is given in the original citation network. The labels are utilized by the supervised models in both training and testing. In contrast, similar to Veličković et al. [44], self-supervised models first learn representations without labels, and then were evaluated by specific learning task, which is performed by directly using these representations to train and test.

Link Prediction: We evaluate the performance on all the datasets. In this work, we not only care about the link prediction between the trained nodes, but also expect the models to predict links between the new nodes. Hence, we introduce two types of link prediction tasks: i) *Transductive link prediction* task allows temporal links between all nodes to be observed up to a time point during the training phase, and uses all the remaining links after that time point for testing. ii) *Inductive link prediction* task predicts links associated with at least one node not observed in the training set. We first conduct the chronological data split as the transductive setting and then randomly select 10% nodes to determine the edges to remove following the study [46].

4.1.4 Implementation Details. To enhance the reproducibility, we provide the implementation details in the subsection.

Euclidean Input: The input feature is Euclidean by default. In this case, we map input features to the Riemannian space before feeding into the non-Euclidean models. Specifically, we utilize the exponential map $exp_O^\kappa(\cdot)$ to perform the mapping from $\mathbb{R}^d$ to $\mathcal{M}_\kappa^d$. The curvature κ can be either set as the parameter of a negative constant for the hyperbolic model, HVGNN and HGTN, or adopted the parametric formulation for the proposed SelfRGNN.

SelfRGNN: We stack the proposed temporal aggregation layers twice in SelfRGNN, and we utilize a two-layer MLP to build CurNN for curvature learning. The dimensionality of our temporal representations is set to 32, while the hyperparameters of the baselines are set for the best performance according to the original papers.

4.2 Overall Performance (*RQ1*)

We utilize AUC, area under RoC curve, as the evaluation metric for the tasks of node classification and link prediction, and report its mean value with 95% confidence interval of 10 independent runs for each model to achieve fair comparisons. The confidence interval are given in the brackets in the tables.

4.2.1 Node Classification. Traditional classifiers work with the Euclidean space, and cannot be directly applied to the Riemannian space due to the essential distinction in geometry. Thus, we first discuss the node classification in the Riemannian space. In this work, following Liu et al. [20], we introduce an output transformation which transforms output representations to Euclidean encodings. Specifically, given an output $\boldsymbol{h}_i(t)$, we first introduce a set of centroids $\{\boldsymbol{\mu}_1, \cdots, \boldsymbol{\mu}_C\}_{(t)}$, where $\boldsymbol{\mu}_c(t)$ is the centroid in Riemannian space learned jointly with the learning model. Then,

Table 3: The summary of AUC(%) for *inductive link prediction* and *transductive link prediction* on Wiki, MOOC, Cora, Social and Physics. The highest scores are in bold, and the second *underlined*. Note, (0.) means that the interval is less than ±0.05%.

		Method	*Inductive Link Prediction* Wiki	MOOC	Social	Cora	Physics	*Transductive Link Prediction* Wiki	MOOC	Social	Cora	Physics
Supervised		EvolveGCN	57.26(1.2)	51.52(2.4)	48.85(1.0)	69.18(0.3)	74.25(1.6)	60.48(0.5)	50.36(0.8)	60.36(0.6)	68.02(0.1)	77.50(1.3)
		VGRNN	62.40(0.7)	61.33(0.5)	65.48(0.8)	75.67(0.1)	73.88(1.0)	71.20(0.7)	90.02(0.3)	78.28(0.7)	79.55(0.2)	78.02(1.0)
	$\mathbb{E}$	DyRep	73.39(1.0)	84.23(1.8)	86.44(0.2)	81.30(0.1)	76.05(1.3)	77.40(0.1)	90.49(0.)	90.85(0.)	82.13(0.2)	78.67(0.7)
		TGAT	95.20(0.6)	69.33(0.1)	53.79(1.1)	76.92(0.2)	81.46(0.2)	96.36(0.1)	72.09(0.3)	56.63(0.5)	77.36(0.1)	95.12(1.1)
		CAWNet	*98.24*(0.5)	90.67(0.6)	95.15(0.7)	*95.20*(0.5)	95.12(0.1)	**99.89**(0.)	92.38(0.6)	94.79(0.2)	95.89(0.3)	*97.86*(0.7)
	$\mathbb{H}$	HVGNN	96.55(0.4)	95.20(0.)	89.12(0.1)	93.04(1.0)	*96.02*(0.3)	98.62(0.1)	89.33(0.2)	84.67(0.1)	93.67(0.5)	97.33(0.1)
		HTGN	87.17(0.5)	91.48(0.6)	*97.33*(0.2)	95.15(0.7)	89.24(0.2)	91.75(0.7)	95.10(1.1)	*98.05*(1.0)	*96.27*(0.2)	91.67(0.1)
Self	$\mathbb{E}$	DDGCL	98.05(0.1)	*96.16*(0.7)	97.08(0.4)	94.89(0.3)	95.70(0.2)	97.92(0.1)	*96.92*(0.1)	95.18(0.1)	95.05(0.6)	96.10(0.3)
		SelfRGNN	**99.16**(0.2)	**98.12**(0.1)	**97.80**(0.1)	**96.36**(0.3)	**97.99**(0.1)	*99.67*(0.1)	**98.85**(0.1)	**99.24**(0.)	**97.68**(1.0)	**98.05**(0.2)

Table 4: Ablation study with node classification in terms of AUC (%) on Wiki, MOOC and Cora datasets.

	Variant	Wiki	MOOC	Cora
$\mathbb{S}$	SelfRGNN$^+$w/oRR	85.18(0.3)	69.24(0.2)	83.60(0.2)
	SelfRGNN$^+$	88.46(0.1)	73.78(0.6)	87.45(1.0)
$\mathbb{E}$	SelfRGNN^{0}w/oRR	83.95(0.1)	70.33(0.2)	84.17(0.5)
	SelfRGNN^0	88.02(0.5)	75.82(0.2)	87.05(0.3)
$\mathbb{H}$	SelfRGNN$^-$w/oRR	85.60(0.6)	72.50(0.1)	88.33(0.1)
	SelfRGNN$^-$	90.82(1.2)	78.16(0.5)	*93.89*(0.3)
Varying	SelfRGNNw/oRR	89.27(0.3)	77.08(0.3)	91.48(0.1)
	SelfRGNN	**93.64**(0.)	**81.28**(0.1)	**94.06**(0.2)
	RGNN(Supervised)	*92.03*(0.1)	*80.89*(0.2)	92.50(1.1)

encoding of $\boldsymbol{h}_i(t)$ is defined as $\boldsymbol{\xi}_i(t) = (\xi_{i1}, \ldots, \xi_{iC})^\top$, where $\xi_{ij} = d_{\mathcal{M}}(\boldsymbol{h}_i(t), \boldsymbol{\mu}_j(t))$, summarizing the position of temporal representations relative to the centroids. Now, we are ready to use logistic regression for node classification, and the likelihood is given as

$$p(y|\boldsymbol{h}(t)) = Sigmoid(\boldsymbol{w}^\top \boldsymbol{h}(t)), \tag{27}$$

where $\boldsymbol{w} \in \mathbb{R}^{|C|}$ is the parameter, and y is the label. Let $\boldsymbol{h}(t) = \boldsymbol{\xi}(t)$ for non-Euclidean models (i.e., the hyperbolic HVGNN and HGTN, the proposed SelfRGNN) and $\boldsymbol{h}(t)$ is the output of Euclidean ones. Note that, the hyperbolic logistic regression proposed in the study [8] cannot be generalized to the Riemannian space of arbitrary curvature. We summarize the experimental results in Table 2, and it is obvious that our SelfRGNN consistently outperforms the state-of-the-arts supervised methods. The superiority lies in that SelfRGNN model the temporal evolvement among hyperspherical, Euclidean, and hyperbolic spaces in reality, and the proposed self-supervised approach learns temporal representations effectively.

4.2.2 Link Prediction. For link perdition tasks, we utilize the Fermi-Dirac decoder, a generalization of sigmoid, to compute probability scores for edges. Formally, given output representations $\boldsymbol{h}(t)$, we have the probability formulated as follows:

$$p((v_i, v_j) \in \mathcal{E} |\ \boldsymbol{h}_i(t), \boldsymbol{h}_j(t)) = \left(\exp\left(\frac{d_{\mathcal{M}}(\boldsymbol{h}_i(t), \boldsymbol{h}_j(t))^2 - r}{t}\right) + 1\right)^{-1}, \tag{28}$$

where r and t are hyperparameters. For each method, $d_{\mathcal{M}}$ is the distance metric of corresponding representation space, e.g., $||\boldsymbol{h}_i(t) - \boldsymbol{h}_j(t)||_2$ for Euclidean models, and we utilize $d_{\mathcal{M}}$ in Table 1 for the Riemannian models. We report the experimental results of both inductive link prediction and transductive link prediction in Table 3. The proposed SelfRGNN achieves the best performance on all the datasets except for one case, the transductive setting on Wiki against CAWNet, which is a supervised method with a calibrated inductive bias for interaction prediction.

4.3 Ablation Study (*RQ2*)

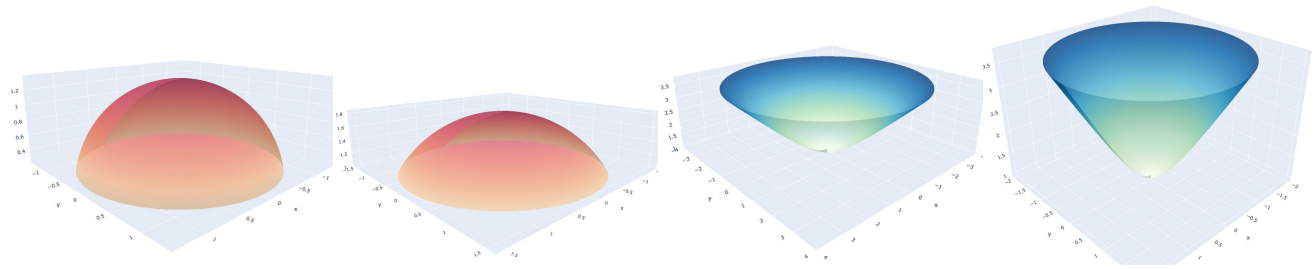
We conduct ablation study to show how each proposed component contributes to the success of SelfRGNN. To this end, we design two types of variants as follows:

- The first type of variants is to verify the effectiveness of *the time-varying curvature*. We use the superscript to distinguish the shape of Riemannian space, e.g., SelfRGNN$^-$ of the negative sign denotes the corresponding model work with the negative curvature hyperbolic space.
- The second type of variants is to verify the effectiveness of the *Riemannian Reweighed contrastive loss*. We use the suffix of **w/oRR** to denotes the corresponding model trained with the original InfoNCE loss.

Additionally, we train the curvature-varying RGNN with labels, referred to as RGNN(Supervised), to show the effectiveness of the proposed self-supervised approach. That is, we have eight variants in total, and note that SelfRGNN^0 means that the proposed SelfRGNN degenerates into a special case of Euclidean space. We utilize node classification as the evaluation task for the ablation study, and the performance of variants are summarized in Table 4. We find that: i) The proposed SelfRGNN with time-varying curvature outperforms its variants with a certain curvature, especially for the zero curvature (Euclidean space). We will further discuss it in our case study. ii) The performance of the proposed RR loss beats that of the original InfoNCE loss, and SelfRGNN even obtain better results than the supervised RGNN, showing the effectiveness of Riemannian reweighted contrastive loss.

4.4 Case Study and Discussion (*RQ3*)

In the case study, we show the curvature of the underlying geometry evolves over time on Physics [10], a citation network related to



(a) The embedding space in May, 1996 (b) The embedding space in May, 1998 (c) The embedding space in May, 2000 (d) The embedding space in May, 2002

Figure 2: Visualization of the embedding spaces over time on Physics datasets.

Table 5: Curvature evolvement on Physics datasets.

Time point	May, 1996	May, 1998	May, 2000	May, 2002
Curvature	*+0.552*	*+0.303*	*-0.259*	*-1.022*

high energy physics from Jan, 1993 to April, 2003. We embed the network at different time points in the 2-dimensional Riemannian spaces, and learn the corresponding curvatures from the data, illustrated in Fig. 3, where the curvature is reported in Table 5. As shown in our case study, *rather than remained in a certain curvature, the underlying space evolves from positive curvature hyperspherical to negative curvature hyperbolic space.* At the beginning, a number of paper join in the network forming triangles or other cyclical structures. As the time progresses, high-impact papers of high citations acquire a better visibility to receive more citations, thus making the underlying geometry evolve to be hyperbolic. Similar phenomenon is also observed in the study [33]. That is, the embedding space shifts among the Riemannian space of various curvatures (positive, zero and negative) in the graph evolvement over time, explaining the inferior of methods with certain curvature and the superior of the proposed SelfRGNN with time-varying curvatures.

5 RELATED WORK

Representation Learning on Temporal Graphs. Representation learning on temporal graphs [1, 16] consider the representations to be time-dependent as the graph evolves over time. For the discrete-time methods, recurrent architectures are frequently employed capture the time-dependence over snapshots, e.g., VGRNN [10] and EvolveGCN [29]. For the continuous-time methods, temporal random walks [23, 25] have shown to be effective, and the recent CAWNet [46] is based on the causual anonymous walks. Temporal point process [15, 47, 55] is another important tool, e.g., DyRep [42] considers an additional hop of interactions for further expressiveness. Recently, GNN-based models have also emerged to deal with continuous time, e.g., TGAT [48] extends GAT [43] to the temporal graphs. JODIE [19] models the message exchanges with the mutual RNNs for bipartite graphs specifically. These studies usually rely on the labels to learn the representations. Recently, DDGCL [41] enables the self-supervised learning in the traditional Euclidean space as the prior works do. To the best of our knowledge, none of the existing studies consider *the self-supervised learning on temporal graphs in the general Riemannian space.*

Riemannian Representation Learning. Recently, it emerges as an exciting alternative to the traditional Euclidean representation learning [8, 24, 26, 38]. We mainly focus on the studies on *graphs.* Most of the existing studies in the literature investigate on the static graphs. A number of hyperbolic GNNs are proposed, e.g., HAN [9] generalize the attention mechanism. HGCN [4], HGNN [20] and LGNN [52] design the hyperbolic graph convolution with different formulations. Fu et al. [7] studies curvature learning for hyperbolic GNN in a joint optimization objective with a reinforcement learning method. Beyond the hyperbolic space, κ-GCN [2] generalizes GCN to arbitrary constant-curvature spaces. Yang et al. [49] propose to represent graphs in the dual space of Euclidean and hyperbolic ones. Recently, Sun et al. [37] study graph learning in the mixed-curvature space, and enable the self-supervised learning with a novel Riemannian contrastive learning. A concurrent work introduces a mixed-curvature model for knowledge graphs specifically with the supervised fashion [45]. It is not until very recently the Riemannian representation learning for temporal graphs are explored. Concretely, HVGNN [39] directly models the temporal interaction with a solid encoding approach in an attentional architecture. In the meanwhile, HTGN [50] designs a novel recurrent architecture on the sequence of snapshots. However, both of them restrict themselves in the negative curvature hyperbolic space. Distinguishing with the recent advances, we propose *the first time-varying curvature model shifting among hyperspherical, Euclidean, and hyperbolic spaces in the evolvement.*

6 CONCLUSION

We for the first time study the representation learning problem on temporal graph in the general Riemannian space, shifting among hyperspherical, Euclidean, and hyperbolic spaces in the evolvement. We present the novel SelfRGNN. Specifically, we propose the curvature-varying GNN with the theoretically grounded time encoding, where the functional curvature is designed to model the evolvement over time. To explore the information in Riemannian space itself, we propose the reweighting self-contrastive approach for representation learning and the edge-based self-supervised curvature learning with Ricci curvature. Extensive experiments show the superiority of SelfRGNN on temporal graph learning.

ACKNOWLEDGMENTS

This paper was supported in part by National Key R&D Program of China through grant 2021YFB1714800, the Fundamental Research Funds for the Central Universities 2022MS018 and S&T Program of Hebei through grant 21340301D. Philip S. Yu is supported in part by NSF under grants III-1763325, III-1909323, III-2106758, and SaTC-1930941. For any correspondence, please refer to Li Sun and Hao Peng.

REFERENCES

[1] Charu C. Aggarwal and Karthik Subbian. 2014. Evolutionary Network Analysis: A Survey. *ACM Comput. Surv.* 47, 1 (2014), 10:1–10:36.

[2] Gregor Bachmann, Gary Bécigneul, and Octavian Ganea. 2020. Constant Curvature Graph Convolutional Networks. In *Proceedings of ICML*, Vol. 119. 486–496.

[3] Ranran Bian, Yun Sing Koh, Gillian Dobbie, and Anna Divoli. 2019. Network Embedding and Change Modeling in Dynamic Heterogeneous Networks. In *Proceedings of the 42nd ACM SIGIR*. ACM, 861–864.

[4] Ines Chami, Zhitao Ying, Christopher Ré, and Jure Leskovec. 2019. Hyperbolic graph convolutional neural networks. In *Advances in NeurIPS*. 4869–4880.

[5] Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey E. Hinton. 2020. A Simple Framework for Contrastive Learning of Visual Representations. In *Proceedings of ICML*, Vol. 119. 1597–1607.

[6] Lun Du, Yun Wang, Guojie Song, Zhicong Lu, and Junshan Wang. 2018. Dynamic Network Embedding: An Extended Approach for Skip-gram based Network Embedding. In *Proceedings of IJCAI*. ijcai.org, 2086–2092.

[7] Xingcheng Fu, Jianxin Li, Jia Wu, Qingyun Sun, Cheng Ji, Senzhang Wang, Jiajun Tan, Hao Peng, and Philip S. Yu. 2021. ACE-HGNN: Adaptive Curvature Exploration Hyperbolic Graph Neural Network. In *Proceedings of the 21st ICDM*. IEEE, 111–120.

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

[9] Caglar Gulcehre, Misha Denil, Mateusz Malinowski, Ali Razavi, Razvan Pascanu, Karl Moritz Hermann, Peter Battaglia, Victor Bapst, David Raposo, Adam Santoro, and Nando de Freitas. 2019. Hyperbolic Attention Networks. In *Proceedings of ICLR*.

[10] Ehsan Hajiramezanali, Arman Hasanzadeh, Nick Duffield, Krishna R Narayanan, Mingyuan Zhou, and Xiaoning Qian. 2019. Variational Graph Recurrent Neural Networks. In *Advances in NeurIPS*. 10700–10710.

[11] Will Hamilton, Zhitao Ying, and Jure Leskovec. 2017. Inductive representation learning on large graphs. In *Advances in NeurIPS*. 1024–1034.

[12] Kaveh Hassani and Amir Hosein Khas Ahmadi. 2020. Contrastive Multi-View Representation Learning on Graphs. In *Proceedings of ICML*, Vol. 119. 4116–4126.

[13] R. Devon Hjelm, Alex Fedorov, Samuel Lavoie-Marchildon, Karan Grewal, Philip Bachman, Adam Trischler, and Yoshua Bengio. 2019. Learning deep representations by mutual information estimation and maximization. In *Proceedings of the 7th ICLR*. OpenReview.net, 1–24.

[14] Shantanu Jain, Martha White, and Predrag Radivojac. 2016. Estimating the class prior and posterior from noisy positives and unlabeled data. In *Advances in NeurIPS*. 2685–2693.

[15] Yugang Ji, Tianrui Jia, Yuan Fang, and Chuan Shi. 2021. Dynamic Heterogeneous Graph Embedding via Heterogeneous Hawkes Process. In *Proceedings of ECML-PKDD (Lecture Notes in Computer Science)*, Vol. 12975. Springer, 388–403.

[16] Seyed Mehran Kazemi, Rishab Goel, Kshitij Jain, Ivan Kobyzev, Akshay Sethi, Peter Forsyth, and Pascal Poupart. 2020. Representation Learning for Dynamic Graphs: A Survey. *Journal Machine Learning Research (JMLR)* 21 (2020), 70:1–70:73.

[17] Thomas N Kipf and Max Welling. 2017. Semi-supervised classification with graph convolutional networks. In *Proceedings of ICLR*.

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

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

[20] Qi Liu, Maximilian Nickel, and Douwe Kiela. 2019. Hyperbolic graph neural networks. In *Advances in NeurIPS*. 8228–8239.

[21] Xiao Liu, Fanjin Zhang, Zhenyu Hou, Zhaoyu Wang, Li Mian, Jing Zhang, and Jie Tang. 2021. Self-supervised Learning: Generative or Contrastive. *IEEE Trans. on Knowledge and Data Engineering* (2021), 1–24.

[22] Zhijun Liu, Chao Huang, Yanwei Yu, and Junyu Dong. 2021. Motif-Preserving Dynamic Attributed Network Embedding. In *Proceedings of the ACM Web Conference (WWW)*. ACM / IW3C2, 1629–1638.

[23] Zhining Liu, Dawei Zhou, Yada Zhu, Jinjie Gu, and Jingrui He. 2020. Towards Fine-grained Temporal Network Representation via Time-Reinforced Random Walk. In *Proceedings of AAAI*. 4973–4980.

[24] Emile Mathieu, Charline Le Lan, Chris J Maddison, Ryota Tomioka, and Yee Whye Teh. 2019. Continuous Hierarchical Representations with Poincaré Variational Auto-Encoders. In *Advances in NeurIPS*. 12544–12555.

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

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

[27] Yann Ollivier. 2009. Ricci curvature of Markov chains on metric spaces. *Journal of Functional Analysis* 256, 3 (2009), 810–864.

[28] Fragkiskos Papadopoulos, Maksim Kitsak, M Ángeles Serrano, Marián Boguná, and Dmitri Krioukov. 2012. Popularity versus similarity in growing networks. *Nature* 489, 7417 (2012), 537.

[29] Aldo Pareja, Giacomo Domeniconi, Jie Chen, Tengfei Ma, Toyotaro Suzumura, Hiroki Kanezashi, Tim Kaler, and Charles E Leisersen. 2020. Evolvegcn: Evolving graph convolutional networks for dynamic graphs. In *Proceedings of AAAI*.

[30] Bryan Perozzi, Rami Al-Rfou, and Steven Skiena. 2014. Deepwalk: Online learning of social representations. In *Proceedings of KDD*. 701–710.

[31] Peter Petersen. 2006. *Riemannian Geometry*. Vol. 171. Springer.

[32] Jiezhong Qiu, Qibin Chen, Yuxiao Dong, Jing Zhang, Hongxia Yang, Ming Ding, Kuansan Wang, and Jie Tang. 2020. GCC: Graph Contrastive Coding for Graph Neural Network Pre-Training. In *Proceedings of KDD*. 1150–1160.

[33] Erzsébet Ravasz and Albert-László Barabási. 2003. Hierarchical organization in complex networks. *Physical review E* 67, 2 (2003), 026112.

[34] Joshua David Robinson, Ching-Yao Chuang, Suvrit Sra, and Stefanie Jegelka. 2021. Contrastive Learning with Hard Negative Samples. In *Proceedings of the 9th ICLR*. OpenReview.net, 1–29.

[35] Emanuele Rossi, Ben Chamberlain, Fabrizio Frasca, Davide Eynard, Federico Monti, and Michael M. Bronstein. 2020. Temporal Graph Networks for Deep Learning on Dynamic Graphs. *CoRR* arxiv.org/abs/2006.10637 (2020).

[36] Michael Spivak. 1979. *A comprehensive introduction to differential geometry*.

[37] Li Sun, Zhongbao Zhang, Junda Ye, Hao Peng, Jiawei Zhang, Sen Su, and Philip S. Yu. 2022. A Self-Supervised Mixed-Curvature Graph Neural Network. In *Proceedings of AAAI*. AAAI Press, 4146–4155.

[38] Li Sun, Zhongbao Zhang, Jiawei Zhang, Feiyang Wang, Yang Du, Sen Su, and Philip S. Yu. 2020. Perfect: A Hyperbolic Embedding for Joint User and Community Alignment. In *Proceedings of the 20th ICDM*. IEEE, 501–510.

[39] Li Sun, Zhongbao Zhang, Jiawei Zhang, Feiyang Wang, Hao Peng, Sen Su, and Philip S. Yu. 2021. Hyperbolic Variational Graph Neural Network for Modeling Dynamic Graphs. In *Proceedings of AAAI*. AAAI Press, 4375–4383.

[40] Jian Tang, Meng Qu, Mingzhe Wang, Ming Zhang, Jun Yan, and Qiaozhu Mei. 2015. Line: Large-scale information network embedding. In *Proceedings of WWW*. 1067–1077.

[41] Sheng Tian, Ruofan Wu, Leilei Shi, Liang Zhu, and Tao Xiong. 2021. Self-supervised Representation Learning on Dynamic Graphs. In *Proceedings of CIKM*. ACM, 1814–1823.

[42] Rakshit Trivedi, Mehrdad Farajtabar, Prasenjeet Biswal, and Hongyuan Zha. 2019. DyRep: Learning Representations over Dynamic Graphs. In *Proceedings of the 7th ICLR*. OpenReview.net, 1–25.

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

[44] Petar Veličković, William Fedus, William L. Hamilton, Pietro Liò, Yoshua Bengio, and R. Devon Hjelm. 2019. Deep Graph Infomax. In *Proceedings of ICLR*. 1–24.

[45] Shen Wang, Xiaokai Wei, Cícero Nogueira dos Santos, Zhiguo Wang, Ramesh Nallapati, Andrew O. Arnold, Bing Xiang, Philip S. Yu, and Isabel F. Cruz. 2021. Mixed-Curvature Multi-Relational Graph Neural Network for Knowledge Graph Completion. In *Proceedings of The ACM Web Conference (WWW)*. ACM / IW3C2, 1761–1771.

[46] Yanbang Wang, Yen-Yu Chang, Yunyu Liu, Jure Leskovec, and Pan Li. 2021. Inductive Representation Learning in Temporal Networks via Causal Anonymous Walks. In *Proceedings of the 9th ICLR*. OpenReview.net, 1–21.

[47] Zhihao Wen and Yuan Fang. 2022. TREND: TempoRal Event and Node Dynamics for Graph Representation Learning. In *Proceedings of the WWW (The ACM Web Conference 2022)*. ACM, 1159–1169.

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

[49] Haoran Yang, Hongxu Chen, Shirui Pan, Lin Li, Philip S. Yu, and Guandong Xu. 2022. Dual Space Graph Contrastive Learning. In *Proceedings of The ACM Web Conference (WWW)*. ACM, 1238–1247.

[50] Menglin Yang, Min Zhou, Marcus Kalander, Zengfeng Huang, and Irwin King. 2021. Discrete-time Temporal Network Embedding via Implicit Hierarchical Learning in Hyperbolic Space. In *Proceedings of KDD*. ACM, 1975–1985.

[51] Ze Ye, Kin Sum Liu, Tengfei Ma, Jie Gao, and Chao Chen. 2020. Curvature Graph Network. In *Proceedings of the 8th ICLR*. OpenReview.net, 1–15.

[52] Yiding Zhang, Xiao Wang, Chuan Shi, Nian Liu, and Guojie Song. 2021. Lorentzian Graph Convolutional Networks. In *Proceedings of WWW*. 1249–1261.

[53] Lekui Zhou, Yang Yang, Xiang Ren, Fei Wu, and Yueting Zhuang. 2018. Dynamic network embedding by modeling triadic closure process. In *Proceedings of AAAI*. 571–578.

[54] Dingyuan Zhu, Peng Cui, Daixin Wang, and Wenwu Zhu. 2018. Deep variational network embedding in Wasserstein space. In *Proceedings of KDD*. 2827–2836.

[55] Yuan Zuo, Guannan Liu, Hao Lin, Jia Guo, Xiaoqian Hu, and Junjie Wu. 2018. Embedding temporal network via neighborhood formation. In *Proceedings of KDD*. 2857–2866.