# SRBGCN: Tangent space-Free Lorentz Transformations for Graph Feature Learning

Abdelrahman Mostafa[1]
abdelrahman.mostafa@oulu.fi
Wei Peng[2]
wepeng@stanford.edu
Guoying Zhao[1,*]
guoying.zhao@oulu.fi

[1] Center for Machine Vision and Signal Analysis,
University of Oulu,
Oulu, Finland

[2] Department of Psychiatry and Behavioral Sciences,
Stanford University,
California, USA

**Abstract**

Hyperbolic graph convolutional networks have been successfully applied to represent complex graph data structures. However, optimization on Riemannian manifolds is non-trivial thus most of the existing hyperbolic networks build the network operations on the tangent space of the manifold, which is a Euclidean local approximation. This distorts the learnt features and limits the representation capacity of the network. In this work, we introduce a fully hyperbolic graph convolutional network (**GCN**), referred to as **SRBGCN**, which performs neural computations such as feature transformation and aggregation directly on the manifold, using manifold-preserving Lorentz transformations that include spatial rotation (**SR**) and boost (**B**) operations. Experiments conducted on static graph datasets for node classification and link prediction tasks validate the performance of the proposed method.

## 1 Introduction

Graph convolutional networks (GCNs) were proposed to make use of the graph topology and model the spatial relationship between graph nodes, hence generalizing the convolution operation to graph data [6, 11]. Initially, the proposed models were built in the Euclidean space [10, 14, 22, 26, 29] which is not the natural space for embedding graph data and produces distorted feature representations [2, 17]. Hyperbolic spaces are more suitable for representing graph data as the space volume is increasing exponentially which is perfect for embedding tree-like data structures that also grow exponentially with the depth of the tree whereas the space grows polynomially for the Euclidean space. Motivated by this, recent works built GCNs in the hyperbolic space to take advantage of the hyperbolic geometry properties [2, 13]. The hyperbolic graph convolutional networks (HGCNs) achieved better performance than the corresponding Euclidean ones which shows the effectiveness of using the hyperbolic space to model hierarchical data structures and graph data.

*Corresponding author.

However, these works performed the network operations in the tangent space of the manifold which is a Euclidean local approximation to the manifold at a point. The Euclidean network operations such as feature transformation and feature aggregation are not manifold-preserving and can not be directly applied on the manifold, that is why these methods resort to the tangent space. However, using a tangent space may limit the representation capabilities of the hyperbolic networks which is caused by distortion specially as most of these works used the tangent space at the origin.

In this work, we propose a full manifold-preserving Lorentz feature transformations using both boost and spatial rotation operations to build SRBGCN fully in the hyperbolic space without resorting to the tangent space. Experiments conducted on node classification and link prediction tasks on static graph datasets show the effectiveness of our proposed method. SRBGCN has a good physical interpretation and can be used to build hyperbolic networks to produce less distorted features.

## 2 Related Work

Chami et al. [2] proposed HGCNs where networks operations are performed in the tangent space of the manifold. They were able to achieve better performance than the Euclidean analogs on node classification and link prediction tasks. Concurrently to this work, Liu et al. [13] proposed the Hyperbolic Graph Neural Networks (HGNNs) which performed well on graph classification tasks. Several models have been proposed using different hyperbolic models especially the Lorentz and the Poincaré models for different other tasks such as image segmentation [9], word embeddings [23], human action recognition [16, 18], text classification [34], machine translation [9, 21], knowledge graph embeddings [3] and so on. Zhu et al. [33] built a two-stream network for Euclidean and hyperbolic features and used an interaction module to enhance the learnt feature representations in the two geometries. Similarly, Xu et al. [28] uses a two-stream network and adds joint layers to embed complex structures using the two geometries. Peng et al. [19] presented a comprehensive survey for hyperbolic networks.

Zhang et al. [31] rebuilt the network operations in HGCNs to guarantee that the learnt features follow the hyperbolic geometry and used the Lorentz centroid [12, 20] for aggregating the features. Zhang et al. [30] used attention modules in the hyperbolic space to build hyperbolic networks. Dai et al. [5] built a hyperbolic network by imposing the orthogonal constraint on a sub-matrix of the transformation matrix (subspace transformation). They used the same number of learnable parameters for the feature transformation step as the networks built on the tangent space, however the orthogonal constraint ensured that the transformation is manifold-preserving and they did not need to learn the parameters on the tangent space. They used the Einstein midpoint method defined in the Klein model [24] for the feature aggregation step. Chen et al. [4] used a normalization procedure to keep the points on the manifold. The idea is to learn a linear transformation then to normalize the vector to lie on the manifold. In this work, we introduce a full space manifold-preserving transformation matrix in SRBGCN without the need for normalization to keep the points on the manifold. Moreover, our method is more physically interpretable and also can be easily extended to spatio-temporal or dynamic graphs.

## 3 Background

### 3.1 Graph Convolutional Networks

A static graph $\mathcal{G} = \{\mathcal{V}, \mathcal{E}\}$ where $\mathcal{V} = \{v_1, v_2, \ldots, v_n\}$ is the set of $n$ graph nodes with $\mathcal{E}$ representing the set of graph edges. The edge set $\mathcal{E}$ can be encoded in an adjacency matrix $\mathbf{A} \in R^{n\times n}$ where $\mathbf{A}_{i,j} \in [0,1]$ if there is a link between $v_i$ and $v_j$ otherwise, $\mathbf{A}_{i,j} = 0$. Each node $v_i$ has a feature vector in the Euclidean space $x_i \in R^d$ of dimension $d$ and $\mathbf{X}$ is the set of features for all the n nodes in the graph. The feature transformation step in GCNs is:

$$\mathbf{Y}^l = \mathbf{X}^l \mathbf{W}^l + \mathbf{B}^l \tag{1}$$

where $\mathbf{W}^l$ is the weight matrix corresponding to the input $\mathbf{X}^l$ at layer $l$ and $\mathbf{B}^l$ is the bias translation matrix. The weight matrix acts as a linear transformer whereas the optional bias matrix makes the transformation affine. Then the feature aggregation from neighboring nodes step with nonlinear activation applied can be formulated as:

$$\mathbf{X}^{l+1} = \sigma(\mathbf{D}^{-1/2}(\mathbf{A}+\mathbf{I})\mathbf{D}^{-1/2}\mathbf{Y}^l) \tag{2}$$

where $\sigma$ is an activation function. $\mathbf{D}^{-1/2}(\mathbf{A}+\mathbf{I})\mathbf{D}^{-1/2}$ is the normalized adjacency matrix to normalize nodes weights in the neighboring set. $\mathbf{D}$ is a diagonal matrix where $\mathbf{D}_{i,i} = 1 + \sum_j \mathbf{A}_{i,j}$ and $\mathbf{I}$ is the identity matrix to keep identity features. $\mathbf{X}^{l+1}$ represents the output of layer $l$ which can be the input to the next layer $l+1$. A GCN is built by stacking a number of those layers. Clearly, the linear transformation matrix $\mathbf{W}$ can not be used in hyperbolic networks as this unconstrained transformation matrix will not keep points on the manifold i.e. not manifold preserving transformations. The same applies for the aggregation step as the Euclidean mean operation is not manifold-preserving.

### 3.2 Hyperbolic Rotation/ Squeeze Mapping

A hyperbolic geometry review is provided in Appendix A in the supplementary material to make the paper self-contained. A regular rotation is a linear map that preserves the Euclidean inner product $\langle .,. \rangle_{\mathcal{E}} : R^d \times R^d \to R$ where $\langle x, y \rangle_{\mathcal{E}} := \sum_{i=0}^{d-1} x_i y_i$ whereas a hyperbolic rotation or a squeeze mapping is a linear map that preserves the Lorentz-Minkowski inner product $\langle .,. \rangle_{\mathcal{L}} : R^{d+1} \times R^{d+1} \to R$. Regular rotations can be realized by trigonometric functions whereas hyperbolic rotations can be realized by hyperbolic functions which are related to their trigonometric counterparts through complex angles. Intuitively, regular rotations can be thought of as a regular rotation to the axes whereas hyperbolic rotations are rotations in the hyperbolic sense and can be thought of as squeezing the axes (see Figure 1 for visualization), hence the name. The following matrix is a squeeze mapping that keeps the points rotating on a hyperbola ($H^{1,K}$) which is a $1D$ hyperboloid with a $-1/\kappa$ constant negative curvature:

$$\mathbf{L}(\omega) = \begin{bmatrix} \cosh\omega & \sinh\omega \\ \sinh\omega & \cosh\omega \end{bmatrix} \tag{3}$$

where $\omega$ is the hyperbolic rotation parameter.

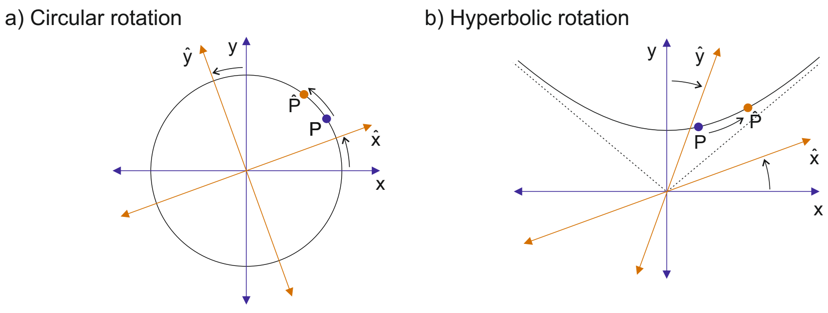

Figure 1: Regular/ circular rotation vs hyperbolic rotation/ squeeze mapping. The axes and the points are color coded for illustration purposes.

## 4 Spatial Rotation Boost Graph Convolutional Network

In this part, we show how to build Spatial Rotation Boost Graph Convolutional Network (SRBGCN) fully in the hyperbolic space. A manifold-preserving Lorentz transformation is used for feature transformation that includes the boost and spatial rotation operations.

### 4.1 Lorentz Transformations

The linear transformation matrix used in Equation 1 for Euclidean features can not be used directly for hyperbolic features as this linear transformation unconstrained matrix is not manifold-preserving for the hyperbolic space. Instead, a Lorentz transformation matrix $\Lambda$ should satisfy the following constraint:

$$\Lambda^T g_{\mathcal{L}} \Lambda = g_{\mathcal{L}} \tag{4}$$

where $\Lambda \in R^{(d+1)\times(d+1)}$,$T$ is the transpose operation of the matrix and $g_{\mathcal{L}} = diag(-1,1,\ldots,1)$ is a diagonal matrix that represents the Riemannian metric for the hyperbolic manifold. A Lorentz transformation matrix is a matrix that is orthogonal with respect to the Minkowski metric $g_{\mathcal{L}}$ and belongs to the Lorentz group. When $\Lambda_0^0$ is positive (the first element in the transformation matrix), the mapping remains on the upper half of the hyperboloid. Taking the determinant of this equation, we obtain $(\det\Lambda)^2 = 1$ i.e. $(\det\Lambda) = \pm 1$. The set of matrices $\Lambda$ with $(\det\Lambda) = 1$ and $\Lambda_0^0 > 0$ are referred to as the proper Lorentz group $SO(d,1)^+$. The Lorentz transformation can be decomposed into a boost and a spatial rotation operations [7, 15] by polar decomposition. The boost matrix is symmetric semi-positive and the spatial rotation matrix is unitary. The spatial rotation operation rotates the spatial coordinates and the boost operation moves a point along the time coordinate without rotating the spatial coordinates.

#### 4.1.1 Spatial Rotation

Intuitively, the subspace manifold at a given level $\hat{x}_0 \in x_0$, $\hat{x}_0 > K$ is a $(d-1)$-dimensional sphere for a $d$-dimensional hyperboloid since $\sum_{i=1}^{d} x_i^2 = \hat{x}_0^2 - K$ is a sphere for any value $\hat{x}_0 > K$. Hence, a regular rotation transformation matrix represents the spatial rotation operation in this subspace manifold.

The spatial rotation matrix is given by:

$$\mathbf{P} = \begin{bmatrix} 1 & 0 \\ 0 & \mathbf{Q} \end{bmatrix}_{(d+1)\times(d+1)} \tag{5}$$

where $\mathbf{Q}$ belongs to the special orthogonal group $SO(d)$ i.e. $\mathbf{Q^T Q} = \mathbf{I}$. It can easily verified that $\mathbf{P}$ is a Lorentz transformation matrix which satisfies Equation 4.

The rotation matrix $\mathbf{Q}$ in Equation 5 can be realized using different representations such as the basic rotations using Trigonometric functions ($d$ degrees of freedom in this case), axis and angle ($d+1$ degrees of freedom) or using the Gram-Schmidt orthonormalization process. We enforce the orthogonalization constraint on the spatial rotation feature transformation matrix as the angles in the other two methods form a discontinuous search space [32] and also there are singularities in this search space (gimbal lock problem).

### 4.1.2 Boost

The boost operation can be realized using hyperbolic rotation or squeeze mapping. Since the squeeze mapping matrix in Equation 3 satisfies the constraint 4, we can use such transformation matrix in the hyperbolic feature transformation step. A $d$ basic hyperbolic rotations as in Equation 3 for each spatial axis with the time axis can be used to realize the boost operation. A more compact hyperbolic rotation representation using a hyperbolic rotation axis $n_d \in R^d$ and a hyperbolic rotation parameter $\omega$ (regular circular rotation can be realized also using an axis and an angle for rotation) in a $d$-dimensional hyperboloid can be represented by the transformation matrix:

$$\mathbf{L} = \begin{bmatrix} \cosh\omega & (\sinh\omega)n_d^T \\ (\sinh\omega)n_d & \mathbf{I} - (1-\cosh\omega)n_d \otimes n_d \end{bmatrix}_{(d+1)\times(d+1)} \tag{6}$$

where $\otimes$ represents the outer product operation. The hyperbolic rotation plane is the plane parallel to the plane spanned by the normal vector $n_d$ (a normalized linear combination of the spatial axes) and the axis $x_0$ (referred to as the time axis in special relativity). Note that when $n$ is a canonical basis vector, the resulting matrix will be similar to the one in Equation 3 after padding with zeros for other canonical spatial basis vectors. The proofs for the derivation of $\mathbf{P}$ and $\mathbf{L}$ are provided in Appendix B in the supplementary material.

## 4.2 Feature Transformation and Aggregation in SRBGCN

Since $\mathbf{L}$ and $\mathbf{P}$ represent the boost and spatial rotation operations, respectively. We use the following Lorentz transformation matrix as the feature transformation matrix for the hyperbolic space:

$$\mathbf{M} = \mathbf{PL} \tag{7}$$

To show that $\mathbf{M}$ is a Lorentz transformation matrix: $(\mathbf{PL})^T g_{\mathcal{L}} (\mathbf{PL}) = \mathbf{L}^T(\mathbf{P}^T g_{\mathcal{L}} \mathbf{P})\mathbf{L} = \mathbf{L}^T g_{\mathcal{L}} \mathbf{L} = g_{\mathcal{L}}$ since both $\mathbf{P}$ and $\mathbf{L}$ are Lorentz transformation matrices as shown before. Figure 2 shows a visualization comparison between different methods. SRBGCN is fully hyperbolic as the boost and spatial rotation operations are manifold-preserving transformation.

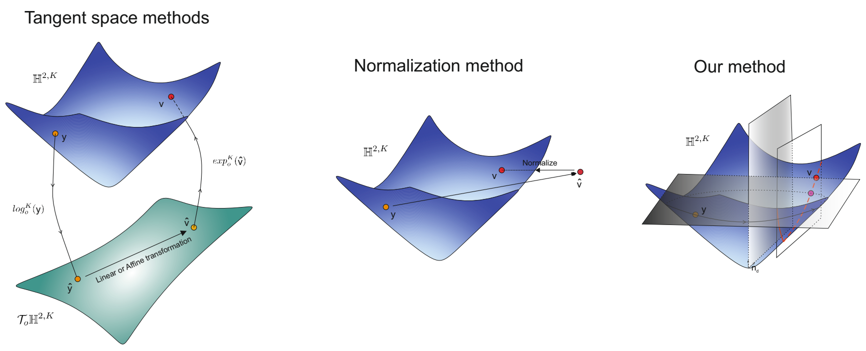

Figure 2: Left: the transformation step in the methods that rely on the tangent space of the manifold. Middle: the normalization method used for feature transformation in [4]. Right: our transformation method using a circular regular rotation and a hyperbolic rotation.

The feature transformation step in SRBGCN in contrast to the Euclidean one in Equation 1 can then be formulated as:

$$\mathbf{Y}_h^l = \mathbf{X}_h^l \mathbf{M}^l = \mathbf{X}_h^l \mathbf{P}^l \mathbf{L}^l \tag{8}$$

where the subscript $h$ represents features on the hyperboloid. To get the initial features representation on the hyperboloid, the exponential map at the origin is used to map the features from the tangent space at the origin to the hyperboloid.

For the feature aggregation step, we use the Lorentz centroid [12, 20] which minimizes the squared Lorentzian distance. It can be computed as:

$$x_i^{h,l+1} = \sqrt{K} \frac{\sum_{j \in NS(i)} w_{i,j} y_j^{h,l}}{|\| \sum_{j \in NS(i)} w_{i,j} y_j^{h,l} \|_{\mathcal{L}}|} \tag{9}$$

where $-1/K$ is the constant negative curvature ($K > 0$) ,$NS(i)$ is the neighbor set for node $i$ which includes the node itself and $w_{i,j}$ is the weight between nodes $i$ and $j$ in the normalized adjacency matrix. Figure 3 shows a visualization of different aggregation techniques used in hyperbolic networks.

## 4.3 Loss Functions in SRBGCN

The margin ranking loss is defined as:

$$Loss = max(d - \hat{d} + m, 0) \tag{10}$$

where $m$ is the non-negative margin hyperparameter. For node classification tasks where the goal is to predict the label of a given node in a graph, $d$ is the distance between a node and its correct class whereas $\hat{d}$ is distance between the node and wrong class. We calculate the hyperbolic distances between the node and a set of trainable centroids on the manifold and feed the distances vector to a softmax classifier for node classification. For link prediction tasks where the goal is to predict the existence of links between graph nodes, cross-entropy is used as the loss function. We use the Fermi-Dirac decoder [23] to calculate the probability for link existence between nodes.

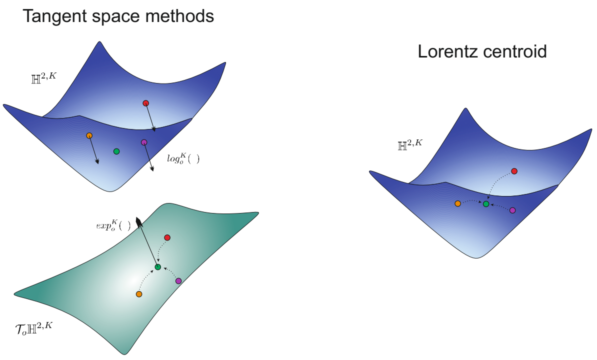

Figure 3: Left: aggregation step in the tangent space methods. Right: the Lorentz centroid.

## 5 Experiments

This section presents the experiments which are performed on four publicly available datasets. The datasets description, evaluation metrics and the evaluation results with comparison to other methods are presented in the following subsections.

### 5.1 Datasets and Evaluation Metrics

Experiments are conducted on four publicly available datasets: Disease, Airport, PubMed and Cora. The Disease dataset is constructed by using the SIR disease spreading model [1] and the label indicates whether the node is infected or not. In the Airport dataset, the nodes represent airports with the population of the city as the label and the edges represents the flight routes between different cities. PubMed and Cora are citation network datasets where the nodes representing the scientific papers with the academic area as the label and the edges representing the existence of citations between papers.

F1 score is used as the evaluation metric for node classification tasks and the Area Under Curve (AUC) as the evaluation metric for link prediction tasks. 10 independent runs are performed and the mean and the standard deviation for each experiment are reported with the same data splits as in [8] that were used in previous works.

### 5.2 Evaluation Results and Comparisons with other Graph Methods

Table 1 shows the performance of different methods including Euclidean and hyperbolic ones. The Euclidean methods include GCN [11], GAT [25], SAGE [10] and SGC [27]. HGCN [2], HAT [30], LGCN [31] and HYPONET [4] are the hyperbolic methods used in the comparison. The hyperbolic methods outperform the Euclidean ones specially for the tree-like Disease dataset which has a zero $\delta$-hyperbolicity (Gromovs hyperbolicity). This proves the effectiveness of using the hyperbolic space to model graph data specially graphs with small $\delta$-hyperbolicity. As shown in the table, SRBGCN outperforms all other methods for most of the benchmarks. Even for the other benchmarks that SRBGCN do not get the best performance using a latent representation of 16, our method can still get comparable results that are much better than other methods in an efficient way. This is a clear evidence of the advantage of learning the graph features directly in the hyperbolic space. It is worth

| | **Dataset** | Disease ($\delta$ = 0) | | Airport ($\delta$ = 1) | | PubMed ($\delta$ = 3.5) | | Cora ($\delta$ = 11) | |
|---|---|---|---|---|---|---|---|---|---|
| | **Method** | LP | NC | LP | NC | LP | NC | LP | NC |
| **Euclidean** | GCN | 64.7±0.5 | 69.7±0.4 | 89.3±0.4 | 81.4±0.6 | 91.1±0.5 | 78.1±0.2 | 90.4±0.2 | 81.3±0.3 |
| | GAT | 69.8±0.3 | 70.4±0.4 | 90.5±0.3 | 81.5±0.3 | 91.2±0.1 | 79.0±0.3 | 93.7±0.1 | 83.0±0.7 |
| | SAGE | 65.9±0.3 | 69.1±0.6 | 90.4±0.5 | 82.1±0.5 | 86.2±1.0 | 77.4±2.2 | 85.5±0.6 | 77.9±2.4 |
| | SGC | 65.1±0.2 | 69.5±0.2 | 89.8±0.3 | 80.6±0.1 | 94.1±0.0 | 78.9±0.0 | 91.5±0.1 | 81.0±0.1 |
| **Hyberpolic** | HGCN | 91.2±0.6 | 82.8±0.8 | 96.4±0.1 | 90.6±0.2 | 96.1±0.2 | 78.4±0.4 | 93.1±0.4 | 81.3±0.6 |
| | HAT | 91.8±0.5 | 83.6±0.9 | - | - | 96.0±0.3 | 78.6±0.5 | 93.0±0.3 | 83.1±0.6 |
| | LGCN | 96.6±0.6 | 84.4±0.8 | 96.0±0.6 | 90.9±1.7 | 96.8±0.1 | 78.6±0.7 | 93.6±0.3 | **83.3±0.7** |
| | HYPONET | 96.8±0.4 | **96.0±1.0** | 97.3±0.3 | 90.9±1.4 | 95.8±0.2 | 78.0±1.0 | 93.6±0.3 | 80.2±1.3 |
| | **SRBGCN** | **97.3±0.2** | 93.0±0.4 | **97.3±0.0** | **91.6±0.9** | **97.2±0.0** | **79.1±0.3** | **95.2±0.0** | 82.9±0.2 |

Table 1: Evaluation results and comparison with other methods. ROC AUC results are reported for Link Prediction (LP) tasks and F1 scores are reported for Node Classification (NC) tasks. The latent feature representation dimension is set to 16 for fair comparison.

| **Dataset** | dim | GAT | HGCN | HAT | LGCN | HYPONET | **SRBGCN** |
|---|---|---|---|---|---|---|---|
| Disease | 4 | 49.4±6.3 | 73.2±6.5 | - | 87.4±3.1 | 91.0±3.8 | **93.1±0.3** |
| | 8 | 76.7±0.7 | 81.5±1.3 | 82.3±1.2 | 82.9±1.2 | 92.9±1.0 | **93.3±0.4** |
| Cora | 64 | 83.1±0.6 | 82.1±0.7 | 83.1±0.5 | 83.5±0.5 | 81.5±0.9 | **83.8±0.3** |

Table 2: Comparison between different methods using different dimensions (dim) on the Disease and Cora datasets for the node classification task.

mentioning that HAT and LGCN methods use techniques such as attention modules to improve the performance. Our method is simple yet very effective. For the disease dataset that has a tree structure with depth of 4, our method achieved a very good performance using latent feature representation dimension of 4 or 8 compared to other methods which shows the effectiveness of using the intrinsic transformations on specially tree-like datasets. This in turn can help in building more compact models for such datasets. For larger and more complicated datasets with higher $\delta$-hyperbolicity such as Cora dataset, our method achieved better performance than other methods using higher dimensional latent space to embed the features for such more complicated datasets. Table 2 shows this comparison between the different methods.

## 5.3 Ablation study

We present an ablation study to show the effectiveness of using both the boost operation and the spatial rotation operation which are the decomposition of the Lorentz transformation matrix. Table 3 shows the performance comparison using the boost operation $\mathbf{Y}_h^l = \mathbf{X}_h^l \mathbf{L}^l$, the spatial rotation operation $\mathbf{Y}_h^l = \mathbf{X}_h^l \mathbf{P}^l$ and using the full Lorentz transformation $\mathbf{Y}_h^l = \mathbf{X}_h^l \mathbf{P}^l \mathbf{L}^l$. Using both the boost and the spatial rotation operations i.e. the full Lorentz transformation usually gives better performance and increases the model expressiveness of the hyperbolic network specially for the node classification tasks with deeper networks. For a model with a $d$-dimensional latent representation, the spatial rotation operation has $d \times d$ parameters (a quadratic operation) whereas the boost has $d+1$ parameters (a linear operation). So, the spatial rotation operation can be considered to contribute more to the performance of the model, which can be seen in the ablation study. It can be noticed that for the link prediction

| **Dataset** | Disease | | Airport | | PubMed | | Cora | |
|---|---|---|---|---|---|---|---|---|
| **Transformation** | LP | NC | LP | NC | LP | NC | LP | NC |
| B only $\mathbf{Y}_h^l = \mathbf{X}_h^l \mathbf{L}^l$ | 97.2±0.3 | 91.4±1.9 | 94.4±2.8 | 87.3±1.4 | 96.9±0.1 | 76.6±1.7 | 94.1±0.2 | 81.7±0.7 |
| SR only $\mathbf{Y}_h^l = \mathbf{X}_h^l \mathbf{P}^l$ | 97.2±0.3 | 92.1±0.6 | **97.3±0.0** | 89.7±1.4 | **97.2±0.0** | 78.8±0.4 | **95.2±0.0** | 81.4±0.4 |
| SR and B $\mathbf{Y}_h^l = \mathbf{X}_h^l \mathbf{P}^l \mathbf{L}^l$ | **97.3±0.2** | **93.3±0.4** | 96.8 ±0.0 | **91.6±0.9** | **97.2±0.0** | **79.1±0.3** | 94.3±0.0 | **83.8±0.3** |

Table 3: **Ablation study** on different datasets to show the effect of using only the spatial rotation (SR) transformation operation, the boost (B) operation and both the spatial rotation and the boost (SR and B) operations.

| **Dataset** | GCN | HGCN | **SRBGCN** |
|---|---|---|---|
| Disease | 67.92±54.91 | 1.04±0.55 | **0.35±0.03** |
| Airport | 175.02±216.90 | 1.39±0.64 | **0.27±0.00** |

Table 4: Distortion values for Disease and Airport datasets.

tasks, the boost operation can slightly lower the performance as the models for the link prediction tasks are usually shallow models, not as deep as the case for the node classification tasks, which seems to slightly benefit from such operation.

## 5.4 Distortion

The average distortion can be computed as: $\frac{1}{n^2}\sum_{i,j}^{n}\left(\left(\frac{ned_{i,j}}{ngd_{i,j}}\right)^2-1\right)^2$ where $ned_{i,j}$ is the normalized embedding distance between nodes $i$ and $j$ and $ngd_{i,j}$ is the normalized graph distance between them. The distortion reflects how close the structure of the graph is preserved in the graph embeddings and the closer the value to zero, the less-distorted the features. Table 4 shows the distortion using GCN, HGCN and SRBGCN methods on the Disease and Airport datasets. The Euclidean GCN method introduces a lot of distortion compared to the hyperbolic ones. At the same time, our method has less distortion than the HGCN method which resorts to the tangent space to perform network operations. This shows the effectiveness of using the intrinsic Lorentz transformations to build fully hyperbolic networks. Figure 4 shows the initial embeddings and the learnt embeddings in the last layer using these methods on the whole Disease dataset. For the link prediction task (top row), it is visually clear that the hyperbolic methods preserve the structure of the tree dataset better than the Euclidean method. Moreover, the hierarchies learnt from our method is much clearer than the one learnt by tangent-space methods. The visualizations and distortion values show that our method generates higher quality features with less distortion. The ability to build better features with less distortion leads to higher performance which shows the superiority of our method. Similarly, for the node classification task (bottom row), our method separates the 2 classes of the Disease dataset more efficiently.

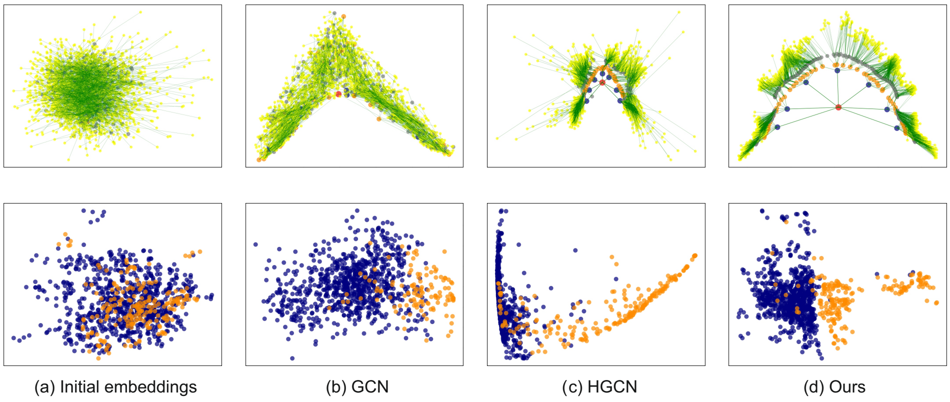

Figure 4: Initial and learnt embeddings using different methods on the whole Disease dataset for the link prediction (top row) and the node classification task (bottom row).

## 6 Conclusion

In this work, we presented a tangent-free full Lorentz transformation layer using the polar decomposition of the Lorentz transformation into both the boost operation and the spatial rotation operation. Our results show the effectiveness of using hyperbolic space to model graph data and to learn less distorted useful features which can be used to build more expressive and compact models for graph learning tasks. We hope this work can be extended to cover and build a unified framework that include other geometries such as Euclidean and Spherical spaces to match the geometry of more complicated structures.

## 7 Acknowledgement

This work was supported by the Research Council of Finland (former Academy of Finland) Academy Professor project EmotionAI (grants 336116, 345122), and the University of Oulu & Research Council of Finland Profi 7 (grant 352788).

## References

[1] Roy M Anderson and Robert M May. *Infectious diseases of humans: dynamics and control*. Oxford university press, 1992.

[2] Ines Chami, Zhitao Ying, Christopher Ré, and Jure Leskovec. Hyperbolic graph convolutional neural networks. *Advances in neural information processing systems*, 32: 4868–4879, 2019.

[3] Ines Chami, Adva Wolf, Da-Cheng Juan, Frederic Sala, Sujith Ravi, and Christopher Ré. Low-dimensional hyperbolic knowledge graph embeddings. In Dan Jurafsky, Joyce Chai, Natalie Schluter, and Joel R. Tetreault, editors, *Proceedings of the 58th Annual*

*Meeting of the Association for Computational Linguistics, ACL 2020, Online, July 5-10, 2020*, pages 6901–6914. Association for Computational Linguistics, 2020. doi: 10.18653/v1/2020.acl-main.617. URL https://doi.org/10.18653/v1/2020.acl-main.617.

[4] Weize Chen, Xu Han, Yankai Lin, Hexu Zhao, Zhiyuan Liu, Peng Li, Maosong Sun, and Jie Zhou. Fully hyperbolic neural networks. In Smaranda Muresan, Preslav Nakov, and Aline Villavicencio, editors, *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), ACL 2022, Dublin, Ireland, May 22-27, 2022*, pages 5672–5686. Association for Computational Linguistics, 2022. doi: 10.18653/v1/2022.acl-long.389. URL https://doi.org/10.18653/v1/2022.acl-long.389.

[5] Jindou Dai, Yuwei Wu, Zhi Gao, and Yunde Jia. A hyperbolic-to-hyperbolic graph convolutional network. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 154–163, 2021.

[6] Michaël Defferrard, Xavier Bresson, and Pierre Vandergheynst. Convolutional neural networks on graphs with fast localized spectral filtering. *Advances in neural information processing systems*, 29:3844–3852, 2016.

[7] Bernard R Durney. Lorentz transformations. *arXiv preprint arXiv:1103.0156*, 2011.

[8] Federico Errica, Marco Podda, Davide Bacciu, and Alessio Micheli. A fair comparison of graph neural networks for graph classification. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net, 2020. URL https://openreview.net/forum?id=HygDF6NFPB.

[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. Hyperbolic attention networks. In *International Conference on Learning Representations*, 2019. URL https://openreview.net/forum?id=rJxHsjRqFQ.

[10] Will Hamilton, Zhitao Ying, and Jure Leskovec. Inductive representation learning on large graphs. *Advances in neural information processing systems*, 30, 2017.

[11] Thomas N. Kipf and Max Welling. Semi-supervised classification with graph convolutional networks, 2017.

[12] Marc Law, Renjie Liao, Jake Snell, and Richard Zemel. Lorentzian distance learning for hyperbolic representations. In *International Conference on Machine Learning*, pages 3672–3681. PMLR, 2019.

[13] Qi Liu, Maximilian Nickel, and Douwe Kiela. Hyperbolic graph neural networks, 2019.

[14] Chien Lu, Jaakko Peltonen, Timo Nummenmaa, and Jyrki Nummenmaa. Nonparametric exponential family graph embeddings for multiple representation learning. In *Uncertainty in Artificial Intelligence*, pages 1275–1285. PMLR, 2022.

[15] Valter Moretti. The interplay of the polar decomposition theorem and the lorentz group. *arXiv preprint math-ph/0211047*, 2002.

[16] Abdelrahman Mostafa, Wei Peng, and Guoying Zhao. Hyperbolic spatial temporal graph convolutional networks. In *2022 IEEE International Conference on Image Processing (ICIP)*, pages 3301–3305, 2022. doi: 10.1109/ICIP46576.2022.9897522.

[17] Maximillian Nickel and Douwe Kiela. Learning continuous hierarchies in the lorentz model of hyperbolic geometry. In *International Conference on Machine Learning*, pages 3779–3788. PMLR, 2018.

[18] Wei Peng, Jingang Shi, Zhaoqiang Xia, and Guoying Zhao. Mix dimension in poincaré geometry for 3d skeleton-based action recognition. In *Proceedings of the 28th ACM International Conference on Multimedia*, pages 1432–1440, 2020.

[19] Wei Peng, Tuomas Varanka, Abdelrahman Mostafa, Henglin Shi, and Guoying Zhao. Hyperbolic deep neural networks: A survey. *IEEE Transactions on pattern analysis and machine intelligence*, 44(12):10023–10044, 2021.

[20] John G Ratcliffe, S Axler, and KA Ribet. *Foundations of hyperbolic manifolds*, volume 149. Springer, 1994.

[21] Ryohei Shimizu, YUSUKE Mukuta, and Tatsuya Harada. Hyperbolic neural networks++. In *International Conference on Learning Representations*, 2021. URL https://openreview.net/forum?id=Ec85b0tUwbA.

[22] Kaiyu Song, Kun Yue, Liang Duan, Mingze Yang, and Angsheng Li. Mutual information based bayesian graph neural network for few-shot learning. In *Uncertainty in Artificial Intelligence*, pages 1866–1875. PMLR, 2022.

[23] Alexandru Tifrea, Gary Bécigneul, and Octavian-Eugen Ganea. Poincare glove: Hyperbolic word embeddings. In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. OpenReview.net, 2019. URL https://openreview.net/forum?id=Ske5r3AqK7.

[24] Abraham A Ungar. *Analytic hyperbolic geometry: Mathematical foundations and applications*. World Scientific, 2005.

[25] Petar Velickovic, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. Graph attention networks. In *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*. OpenReview.net, 2018. URL https://openreview.net/forum?id=rJXMpikCZ.

[26] Petar Velickovic, William Fedus, William L Hamilton, Pietro Liò, Yoshua Bengio, and R Devon Hjelm. Deep graph infomax. *ICLR (Poster)*, 2(3):4, 2019.

[27] Felix Wu, Amauri H. Souza Jr., Tianyi Zhang, Christopher Fifty, Tao Yu, and Kilian Q. Weinberger. Simplifying graph convolutional networks. In Kamalika Chaudhuri and Ruslan Salakhutdinov, editors, *Proceedings of the 36th International Conference on Machine Learning, ICML 2019, 9-15 June 2019, Long Beach, California, USA*, volume 97 of *Proceedings of Machine Learning Research*, pages 6861–6871. PMLR, 2019. URL http://proceedings.mlr.press/v97/wu19e.html.

[28] Xiaoyu Xu, Guansong Pang, Di Wu, and Mingsheng Shang. Joint hyperbolic and euclidean geometry contrastive graph neural networks. *Information Sciences*, 609:799–815, 2022. ISSN 0020-0255. doi: https://doi.org/10.1016/j.ins.2022.07.060. URL https://www.sciencedirect.com/science/article/pii/S0020025522007496.

[29] Muhan Zhang, Zhicheng Cui, Marion Neumann, and Yixin Chen. An end-to-end deep learning architecture for graph classification. In *Thirty-second AAAI conference on artificial intelligence*, 2018.

[30] Yiding Zhang, Xiao Wang, Chuan Shi, Xunqiang Jiang, and Yanfang Fanny Ye. Hyperbolic graph attention network. *IEEE Transactions on Big Data*, pages 1–1, 2021. doi: 10.1109/TBDATA.2021.3081431.

[31] Yiding Zhang, Xiao Wang, Chuan Shi, Nian Liu, and Guojie Song. Lorentzian graph convolutional networks. In *Proceedings of the Web Conference 2021*, pages 1249–1261, 2021.

[32] Yi Zhou, Connelly Barnes, Jingwan Lu, Jimei Yang, and Hao Li. On the continuity of rotation representations in neural networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5745–5753, 2019.

[33] Shichao Zhu, Shirui Pan, Chuan Zhou, Jia Wu, Yanan Cao, and Bin Wang. Graph geometry interaction learning. *Advances in Neural Information Processing Systems*, 33:7548–7558, 2020.

[34] Yudong Zhu, Di Zhou, Jinghui Xiao, Xin Jiang, Xiao Chen, and Qun Liu. Hypertext: Endowing fasttext with hyperbolic geometry. In Trevor Cohn, Yulan He, and Yang Liu, editors, *Findings of the Association for Computational Linguistics: EMNLP 2020, Online Event, 16-20 November 2020*, volume EMNLP 2020 of *Findings of ACL*, pages 1166–1171. Association for Computational Linguistics, 2020. doi: 10.18653/v1/2020.findings-emnlp.104. URL https://doi.org/10.18653/v1/2020.findings-emnlp.104.