Hyperbolic Convolution via Kernel Point Aggregation

Eric Zhonghang Qu [1] Dongmian Zou [1]

[1]Division of Natural and Applied Sciences, Duke Kunshan University, Jiangsu 215316, China

昆山杜克大学 DUKE KUNSHAN UNIVERSITY

Introduction

- Hyperbolic space could effectively embed hierarchical or tree-like data.
- There has been increasing interest in representation learning in hyperbolic space.
- However, many Euclidean neural operations, such as convolution, do not extend to hyperbolic space.
- We fill this gap by proposing the HKConv, a novel trainable hyperbolic convolution operation.
- HKConv not only expressively learns local hyperbolic features, but also equivariant to permutation and invariant to parallel transport of a local neighborhood.
- Using HKConv layers, the Hyperbolic Kernel Network (HKN) advances state-of-the-art in various tasks.

Hyperbolic Geometry

- Hyperbolic geometry is a special kind of Riemannian geometry with a constant negative curvature.
- We use the Lorentz model as the coordinate system, represented by $\mathbb{L}^n_\kappa$. Every point $\boldsymbol{x} \in \mathbb{L}^n_\kappa = \mathbb{L}^n$ satisfies $\langle \boldsymbol{x}, \boldsymbol{x} \rangle_\mathcal{L} = 1/\kappa$, where

$$\langle \boldsymbol{x}, \boldsymbol{y} \rangle_\mathcal{L} := \boldsymbol{x}^\top \mathfrak{g} \boldsymbol{y} = -x_t y_t + \boldsymbol{x}_s^\top \boldsymbol{y}_s, \ \boldsymbol{x}, \boldsymbol{y} \in \mathbb{L}^n. \quad (1)$$

- For $\boldsymbol{x}, \boldsymbol{y} \in \mathbb{L}^n$, $d_\mathcal{L}(\boldsymbol{x}, \boldsymbol{y})$ is the length of the geodesic between them, also called hyperbolic distance.
- For $\boldsymbol{x} \in \mathbb{L}^n$, the tangent space at $\boldsymbol{x}$ is $\mathcal{T}_{\boldsymbol{x}}\mathbb{L}^n$.
- For $\boldsymbol{x}, \boldsymbol{y} \in \mathbb{L}^n$ and $\boldsymbol{v} \in \mathcal{T}_{\boldsymbol{x}}\mathbb{L}^n$, $\exp_{\boldsymbol{x}}(\boldsymbol{v})$ is the exponential map of $\boldsymbol{v}$ at $\boldsymbol{x}$.
- We use $\log_{\boldsymbol{x}} : \mathbb{L}^n \to \mathcal{T}_{\boldsymbol{x}}\mathbb{L}^n$ to denote the logarithmic map such that $\log_{\boldsymbol{x}}(\exp_{\boldsymbol{x}}(\boldsymbol{v})) = \boldsymbol{v}$.
- For two points $\boldsymbol{x}, \boldsymbol{y} \in \mathbb{L}^n$, we use $\mathrm{PT}_{\boldsymbol{x}\to\boldsymbol{y}}$ to denote the parallel transport map which "transports" a vector from $\mathcal{T}_{\boldsymbol{x}}\mathbb{L}^n$ to $\mathcal{T}_{\boldsymbol{y}}\mathbb{L}^n$ along the geodesic from $\boldsymbol{x}$ to $\boldsymbol{y}$.
- For the full description of the hyperbolic operations, please see the full paper.

Hyperbolic Kernel Points

- To construct an HKConv layer, the first step is to fix K kernel points $\{\tilde{\boldsymbol{x}}_k\}_{k=1}^K$ in the input space $\mathbb{L}^m$.
- The locations of kernel points are predetermined to ensure stable training.
- The kernel points can be regarded as located in near the hyperbolic origin $\boldsymbol{o} = [1, \mathbf{0}_m^\top]^\top \in \mathbb{L}^m$.
- We make the kernel points far away from each other, but also not too far from origin (Thomas et al., 2019).
- Specifically, we minimize the following loss function with Riemannian gradient descent:

$$\mathfrak{L}(\{\tilde{\boldsymbol{x}}_k\}_{k=1}^K) = \sum_{k=1}^K \sum_{l \neq k} \frac{1}{d_\mathcal{L}(\tilde{\boldsymbol{x}}_l, \tilde{\boldsymbol{x}}_k)} + \sum_{k=1}^K d_\mathcal{L}(\boldsymbol{o}, \tilde{\boldsymbol{x}}_k). \quad (2)$$

- Figure 1 shows 5 and 10 optimal kernel points in $\mathbb{L}^2$.

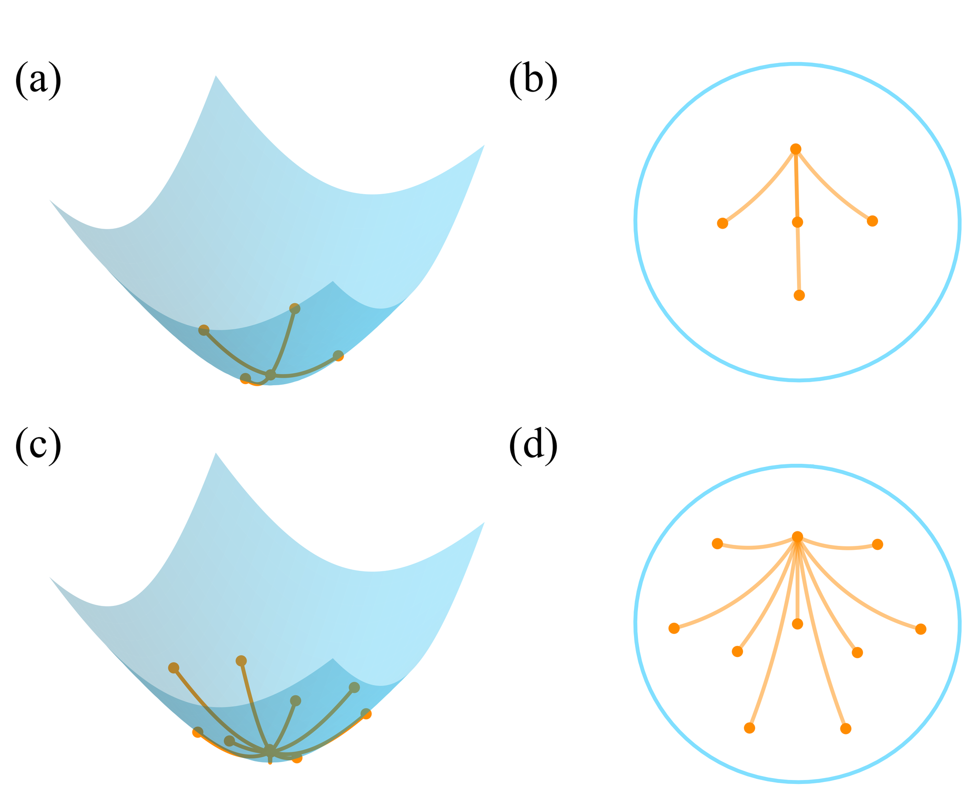

Figure 1. Illustration of kernel points in $\mathbb{L}^2$. Left: Lorentz model. Right: Poincaré model.

Acknowledgement

We thank the Office of Undergraduate Studies and the DKU Summer Research Scholars Program for supporting this work.

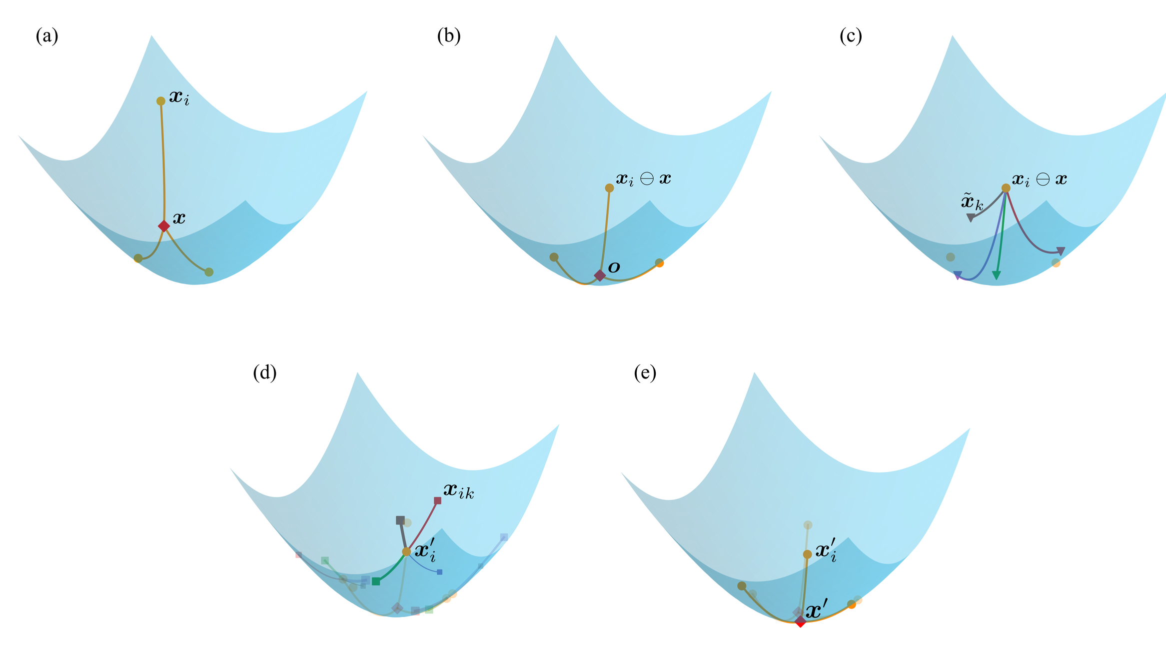

Figure 2. Illustration of HKConv.

Hyperbolic Kernel Convolution

- HKConv applies the kernel points to local neighborhoods of input hyperbolic features.
- Let $\mathbb{X} \subset \mathbb{L}^m$ denote the collection of input hyperbolic features. For each $\boldsymbol{x} \in \mathbb{X}$, we use $\mathcal{N}(\boldsymbol{x}) \subset \mathbb{X} - \{\boldsymbol{x}\}$ to denote the neighbors of $\boldsymbol{x}$.

Step 1. Transformation of relative input features.

- To get the relative input features, we need to perform the hyperbolic "translation".
- Specifically, given $\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{u} \in \mathbb{L}^m$, we define

$$\mathrm{T}_{\boldsymbol{x}\to\boldsymbol{y}}(\boldsymbol{u}) := \exp_{\boldsymbol{y}}\left(\mathrm{PT}_{\boldsymbol{x}\to\boldsymbol{y}}(\log_{\boldsymbol{x}}(\boldsymbol{u}))\right). \quad (3)$$

- Also, to analogize to "translation by $\boldsymbol{x}$", we denote

$$\boldsymbol{u} \ominus \boldsymbol{x} := \mathrm{T}_{\boldsymbol{x}\to\boldsymbol{o}}(\boldsymbol{u}). \quad (4)$$

- Then, for each $\boldsymbol{x}_i \in \mathcal{N}(\boldsymbol{x})$, the translation $\boldsymbol{x}_i \ominus \boldsymbol{x}$ gives the relative input feature.
- The first step of HKConv is to extract features from translated input features using a learnable hyperbolic linear layer (Chen et al., 2022) for each kernel point $\tilde{\boldsymbol{x}}_k$:

$$\boldsymbol{x}_{ik} = \mathrm{HLinear}_k(\boldsymbol{x}_i \ominus \boldsymbol{x}), \quad k = 1, \cdots, K. \quad (5)$$

Step 2. Aggregation based on correlation with kernels.

- We first calculate the correlation between the input features and the kernel points.
- Here, each neighbor $\boldsymbol{x}_i \in \mathcal{N}(\boldsymbol{x})$ is translated by $\boldsymbol{x}_i \ominus \boldsymbol{x}$, thus the correlation is $d_\mathcal{L}(\boldsymbol{x}_i \ominus \boldsymbol{x}, \tilde{\boldsymbol{x}}_k)$.
- To aggregate the input features, we use hyperbolic centroid denoted by HCent, which could be seen as a weighted mean in hyperbolic space (Law et al., 2019).
- In the second step, we use the correlations as the weight to aggregate the transformed input features:

$$\boldsymbol{x}'_i = \mathrm{HCent}(\{\boldsymbol{x}_{ik}\}_{k=1}^K, \{d_\mathcal{L}(\boldsymbol{x}_i \ominus \boldsymbol{x}, \tilde{\boldsymbol{x}}_k)\}_{k=1}^K). \quad (6)$$

Step 3. Final aggregation with optional attention.

- The last step in HKconv is to aggregate all the $\boldsymbol{x}'_i$'s from the neighborhood:

$$\boldsymbol{x}' = \mathrm{HCent}(\{\boldsymbol{x}'_i\}_{i=1}^N, \{w_i\}_{i=1}^N), \quad (7)$$

where $\{w_i\}_{i=1}^N$ could be $\mathbf{1}_N$ or learnt attention weights.

The whole process of HKConv is illustrated in Figure 2.

Properties of HKConv

Since (7) is permutation equivariant, HKConv is equivariant to permutation of the order of data points:

Proposition 1. (Equivariance to permutation of the order of data points) *Let $\boldsymbol{X} \in \mathbb{R}^{N\times(m+1)}$ be the data matrix whose j-th row $\boldsymbol{x}_j \in \mathbb{R}^{m+1}$ is identified as a point in $\mathbb{L}^m$. For any matrix $\boldsymbol{P} \in \mathbb{R}^{N\times N}$ that permutes the rows of $\boldsymbol{X}$,*

$$\mathrm{HKConv}(\boldsymbol{PX}) = \boldsymbol{P}\mathrm{HKConv}(\boldsymbol{X}),$$

where HKConv is applied to each row of $\boldsymbol{X}$.

HKConv also enjoys invariance under local translations:

Theorem 2. (Local translation invariance) *Fix $\boldsymbol{x} \in \mathbb{X}$ and its neighborhood $\mathcal{N}(\boldsymbol{x}) \subset \mathbb{X}$. For any $\boldsymbol{y} \in \mathbb{L}^n$ in the geodesic from $\boldsymbol{o}$ to $\boldsymbol{x}$,*

$$\mathrm{HKConv}(\mathrm{T}_{\boldsymbol{x}\to\boldsymbol{y}}(\boldsymbol{x}); \mathrm{T}_{\boldsymbol{x}\to\boldsymbol{y}}(\mathcal{N}(\boldsymbol{x}))) = \mathrm{HKConv}(\boldsymbol{x}; \mathcal{N}(\boldsymbol{x})).$$

Please see the full paper for the complete proof.

Experiments

Graph Classification

- In graph classification, one assigns labels to full graphs.
- When a graph has a small Gromov hyperbolicity δ, it can be well embedded in the hyperbolic space (Sonthalia and Gilbert, 2020).
- We test our HKN on graph datasets with small δ.
- We also compare HKN with three alternative ablations:
  - **HKN-direct**: we apply hyperbolic linear layers directly to each $\boldsymbol{x}_i$ in (5) instead of $\boldsymbol{x}_i \ominus \boldsymbol{x}$, and use $d_\mathcal{L}(\boldsymbol{x}_i, \mathrm{PT}_{\boldsymbol{o}\to\boldsymbol{x}}(\tilde{\boldsymbol{x}}_k))$ to translate the kernels to the neighborhood of $\boldsymbol{x}$ in (6).
  - **HKN-random**: we randomly sample the kernel points from a wrapped normal distribution, instead of solving (2).
  - **HKN-train**: we regard the positions of the kernel points as trainable parameters.

Table 1. Graph classification results. Mean accuracy (%) and standard deviation are reported. NaN indicates a numerically unstable result.

	PTC	ENZYMES	PROTEIN	IMDB-B	IMDB-M
# of graphs	344	600	1113	1000	1500
# of classes	2	6	2	2	3
Avg # of nodes	25.56	32.63	39.06	19.77	13
Avg # of edges	25.96	62.14	72.82	96.53	65.94
Avg Hyperbolicity	$\delta = 0.725$	$\delta = 1.15$	$\delta = 1.095$	$\delta = 0.2385$	$\delta = 0.1157$
GCN	63.87±2.65	66.39±6.91	74.54±0.45	73.32±0.39	50.27±0.38
GIN	66.58±6.78	59.79±4.31	70.67±1.08	72.78±0.86	47.91±1.03
GMT	65.89±2.16	67.52±4.28	75.09±0.59	73.48±0.76	50.66±0.82
HGCN	55.17±3.21	53.63±5.12	68.41±2.15	61.71±0.97	50.12±0.71
H2H-GCN	66.14±3.91	60.84±4.72	72.12±3.63	68.19±0.82	48.31±0.63
HyboNet	65.56±4.19	56.82±5.39	65.36±2.81	71.26±1.28	54.35±0.98
HKN	**73.69**±4.81	**82.46**±5.62	**83.22**±5.19	**79.92**±1.31	**56.53**±1.02
HKN-direct	66.42±3.13	58.31±4.81	64.78±3.51	67.65±0.89	46.31±0.61
HKN-random	61.24±8.32	67.41±8.62	68.85±7.81	68.97±4.38	52.14±3.83
HKN-train	NaN	NaN	NaN	NaN	NaN

Node Classification

- In node classification, one classifies nodes within a graph according to some nodes with known labels.
- We also test on graphs with small δ.

Table 2. Node classification results. Mean F1 score (%) and standard deviation are reported. NaN indicates a numerically unstable result.

	Cornell	Texas	Wisconsin	Chameleon	Squirrel	Actor
# of Nodes	183	183	251	2,277	5,201	7,600
# of Edges	280	295	466	31,421	198,493	26,752
# of Features	1,703	1,703	1,703	2,325	2,089	931
# of Classes	5	5	5	5	5	5
Hyperbolicity	$\delta = 1$	$\delta = 1$	$\delta = 1$	$\delta = 1.5$	$\delta = 1.5$	$\delta = 1.5$
MLP	81.89±6.40	80.81±4.75	85.29±3.31	46.21±2.99	28.77±1.56	36.53±0.70
GCN	60.54±5.30	55.14±5.16	51.76±3.06	64.82±2.24	53.43±2.01	27.32±1.10
GAT	61.89±5.05	52.16±6.63	49.41±4.09	60.26±2.50	40.72±1.55	27.44±0.89
GraphSAGE	75.95±5.01	82.43±6.14	81.18±5.56	58.73±1.68	41.61±0.74	34.23±0.99
GCNII	77.86±3.79	77.57±3.83	80.39±3.40	63.86±3.04	38.47±1.58	37.44±1.30
Geom-GCN	60.54±3.67	66.76±2.72	64.51±3.66	60.00±2.81	38.15±0.92	31.59±1.15
WRGAT	81.62±3.90	83.62±5.50	86.98±3.78	65.24±0.87	48.85±0.78	36.53±0.77
LINKX	77.84±5.81	74.60±8.37	75.49±5.72	68.42±1.38	61.81±1.80	36.10±1.55
GloGNN	83.51±4.26	84.32±4.15	87.06±3.53	69.78±2.42	57.54±1.39	37.35±1.30
GloGNN++	**85.95**±5.10	84.05±4.90	88.04±3.22	71.21±1.84	57.88±1.76	37.70±1.40
HGCN	79.43±0.47	70.13±0.32	83.26±0.51	NaN	62.31±0.57	36.58±0.79
HyboNet	77.27±0.71	72.23±0.94	86.52±0.51	74.91±0.58	69.07±0.64	45.74±0.82
HKN (Ours)	84.14±0.53	**90.94**±0.66	**91.31**±0.36	**85.27**±0.57	**75.26**±0.38	**67.20**±0.68

Machine Translation and Dependency Tree Probing

Table 3. Machine translation and dependency tree probing results. The BLEU (BiLingual Evaluation Understudy) scores on the test set are reported for machine translation. UUAS, Dspr., Root%, Nspr. are reported for dependency tree probing.

	Machine Translation				Dependency Tree Probing			
	IWSLT' 14	WMT' 17			Distance		Depth	
	d=64	d=64	d=128	d=256	UUAS	Dspr.	Root%	Nspr.
ConvSeq2Seq	23.6	14.9	20.0	21.8	-	-	-	-
Transformer	23.0	17.0	21.7	25.1	0.36	0.3	12	0.88
HNN++	22.0	17.0	19.4	21.8	-	-	-	-
HATT	23.7	18.8	22.5	25.5	0.5	0.64	49	0.88
HyboNet	25.9	19.7	23.3	26.2	0.59	0.7	64	0.92
HKN (Ours)	**27.3**	**20.1**	**25.6**	**29.1**	**0.62**	**0.74**	**72**	**0.94**

More experimental results are presented in the full paper.

References

Chen, W., Han, X., Lin, Y., Zhao, H., Liu, Z., Li, P., Sun, M., and Zhou, J. (2022). Fully hyperbolic neural networks. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 5672–5686.

Law, M., Liao, R., Snell, J., and Zemel, R. (2019). Lorentzian distance learning for hyperbolic representations. In *International Conference on Machine Learning*, pages 3672–3681. PMLR.

Sonthalia, R. and Gilbert, A. (2020). Tree! i am no tree! i am a low dimensional hyperbolic embedding. *Advances in Neural Information Processing Systems*, 33:845–856.

Thomas, H., Qi, C. R., Deschaud, J.-E., Marcotegui, B., Goulette, F., and Guibas, L. J. (2019). Kpconv: Flexible and deformable convolution for point clouds. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 6411–6420.