Hyperbolic Deep Learning for Foundation Models: A Tutorial

Tutor: Neil He, Menglin Yang, Rex Ying
Contributor: Ngoc Bui, Hiren Madhu

neil.he@yale.edu, menglin.yang@outlook.edu, rex.ying@yale.edu

Yale University

Outline

- Preliminary
  - Motivation
  - Hyperbolic Geometry
- Building Blocks
  - Basic Hyperbolic NN Operations
  - Hyperbolic NN Architectures
- Hyperbolic Foundation Models
  - Hyperbolic LLMs & Transformers
  - Hyperbolic Vision Foundation Models
  - Hyperbolic Multi-Modal Foundation Models

Our goals is to introduce:

1. Motivations for Hyperbolic Foundation Models
2. Hyperbolic Geometry Basics
3. Hyperbolic Basic Neural Operations
4. Current Methods in Hyperbolic Foundation Models
5. Future Directions

Part 1: Preliminary

Motivation	Geometry of Inputs to Foundation Models	
	Limitations of Euclidean Embeddings	
	Alternative Geometric Spaces	
Hyperbolic Geometry	Riemannian Manifold & Hyperbolic Space	Poincare Ball
		Lorentz Hyperboloid
	Tangent Spaces & Geodesics	Exponential Maps
		Logarithmic Maps
		Parallel Transport

Part 1: Preliminary – Goals (45 Min):

1. Motivate Hyperbolic Geometry for Foundation Models
2. Introduce Basics of Hyperbolic Geometry

Part 2: Building Blocks

Hyperbolic Basic NN Operations	Linear Transformations
	Residual connection
	Normalization
	Activation
	Attention Mechanisms
Hyperbolic NN Model Architecture	MLP
	ResNet & CNN
	GNN

Part 2: Building Blocks – Goals (55 Min):

1. Introduce Basics Hyperbolic Neural Network Operations (e.g. Linear Transformations, Attention Mechanisms)
2. Introduce Basic Hyperbolic Neural Networks Models

Part 3: Hyperbolic Foundation Models

Hyperbolic LLMs & Transformers	FNN, HNN++, HAN
	HypFormer
	HypLoRA
	HELM
Hyperbolic Vision Foundation Models	Hyp-ViT, HVT, LViT
	HCL, RHCL
Hyperbolic Multi-Modal Foundation Models	MERU, HypCoCLIP, L-CLIP
	H-BLIP-2

Part 3: Hyperbolic Foundation Models – Goals (70 Min):

1. Introduce Current Methods in Hyperbolic Foundation Models
2. Discuss Potential Feature Directions

Part 1: Background: Motivation & Theory (40 Minutes)

Token Relationship

- The sun rises above the river.
- The river flows through the forest.
- The forest is dense with tall trees.
- Trees sway gently in the wind.
- The wind carries the scent of flowers.
- Flowers bloom brightly under the sun.
- The sun sets over the mountains.
- The mountains echo with the sound of birds.
- Birds fly freely across the sky.
- The sky turns dark as stars appear.

How do we analyze token relationship?

- Word Transition: which words lead to each other in a piece of writing?
- Co-occurrence: which words tend to appear together in a Transformer input/output context?
- Pointwise Mutual Information: how many times more often two words co-occur than if they were independent?

Token Relationship Example: Word Transition

- "co-occurrence" of window size 1

	above	dense	flows	forest	is	rises	river	sun	tall	the	through	trees	with
above	0	0	0	0	0	0	0	0	0	1	0	0	0
dense	0	0	0	0	0	0	0	0	0	0	0	0	1
flows	0	0	0	0	0	0	0	0	0	0	1	0	0
forest	0	0	0	0	1	0	0	0	0	0	0	0	0
is	0	1	0	0	0	0	0	0	0	0	0	0	0
rises	1	0	0	0	0	0	0	0	0	0	0	0	0
river	0	0	1	0	0	0	0	0	0	0	0	0	0
sun	0	0	0	0	0	1	0	0	0	0	0	0	0
tall	0	0	0	0	0	0	0	0	0	0	0	1	0
the	0	0	0	2	0	0	2	1	0	0	0	0	0
through	0	0	0	0	0	0	0	0	0	1	0	0	0
trees	0	0	0	0	0	0	0	0	0	0	0	0	0
with	0	0	0	0	0	0	0	0	1	0	0	0	0

Token Relationship Example: Word Transition

	above	dense	flows	forest	is	rises	river	sun	tall	the	through	trees	with
above	0	0	0	0	0	0	0	0	0	1	0	0	0
dense	0	0	0	0	0	0	0	0	0	0	0	0	1
flows	0	0	0	0	0	0	0	0	0	0	1	0	0
forest	0	0	0	0	1	0	0	0	0	0	0	0	0
is	0	1	0	0	0	0	0	0	0	0	0	0	0
rises	1	0	0	0	0	0	0	0	0	0	0	0	0
river	0	0	1	0	0	0	0	0	0	0	0	0	0
sun	0	0	0	0	0	1	0	0	0	0	0	0	0
tall	0	0	0	0	0	0	0	0	0	0	0	1	0
the	0	0	0	2	0	0	2	1	0	0	0	0	0
through	0	0	0	0	0	0	0	0	0	1	0	0	0
trees	0	0	0	0	0	0	0	0	0	0	0	0	0
with	0	0	0	0	0	0	0	0	1	0	0	0	0

Word “the”: Token frequency is 5, out-degree is 5, in-degree is 2

Observations

- There is significant patterns in token relationships
- Tokens are not equal (in terms of frequencies)

Token Relationship Example: Word Transition

	above	dense	flows	forest	is	rises	river	sun	tall	the	through	trees	with
above	0	0	0	0	0	0	0	0	0	1	0	0	0
dense	0	0	0	0	0	0	0	0	0	0	0	0	1
flows	0	0	0	0	0	0	0	0	0	0	1	0	0
forest	0	0	0	0	1	0	0	0	0	0	0	0	0
is	0	1	0	0	0	0	0	0	0	0	0	0	0
rises	1	0	0	0	0	0	0	0	0	0	0	0	0
river	0	0	1	0	0	0	0	0	0	0	0	0	0
sun	0	0	0	0	0	1	0	0	0	0	0	0	0
tall	0	0	0	0	0	0	0	0	0	0	0	1	0
the	0	0	0	2	0	0	2	1	0	0	0	0	0
through	0	0	0	0	0	0	0	0	0	1	0	0	0
trees	0	0	0	0	0	0	0	0	0	0	0	0	0
with	0	0	0	0	0	0	0	0	1	0	0	0	0

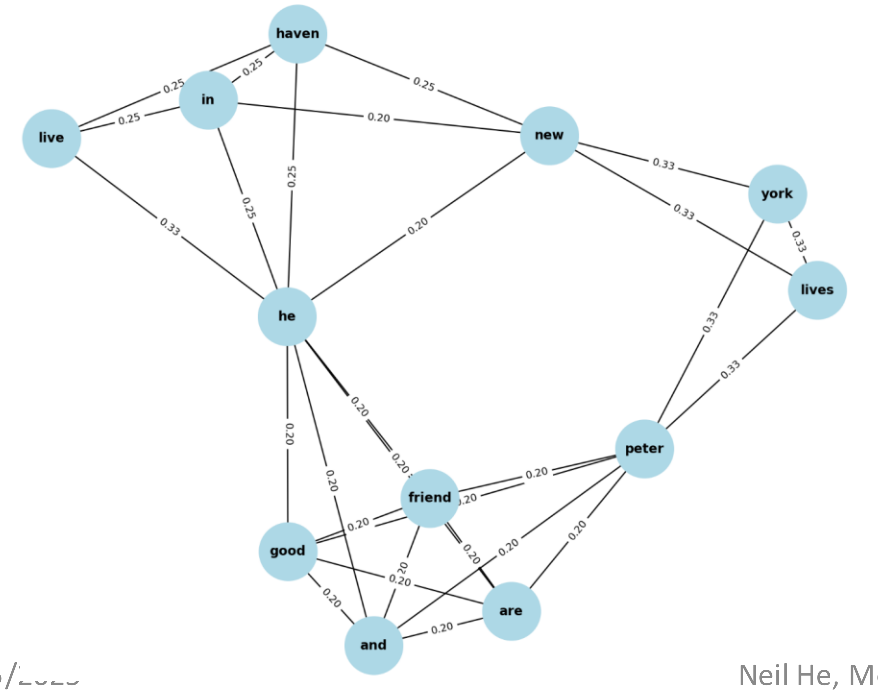
Most other token frequency, out/in degree are 1 or 0

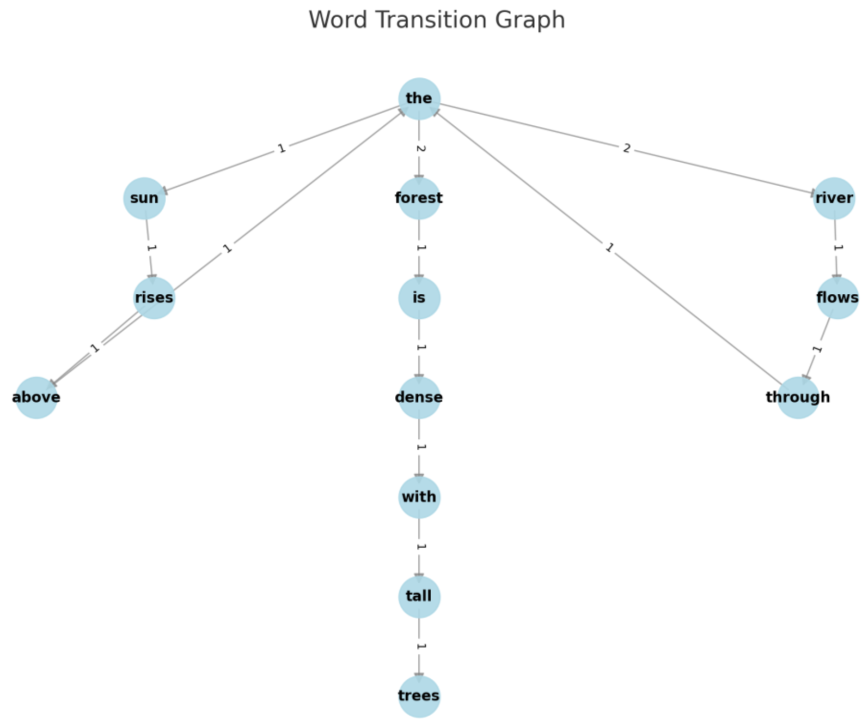
Observations

- There is significant patterns in token relationships
- Tokens are not equal (in terms of frequencies)

Token Relationship Example: Word Transition

- Observations
- There is significant patterns in token relationships
- Tokens are not equal (in terms of frequencies)
- Tokens have underlying structure

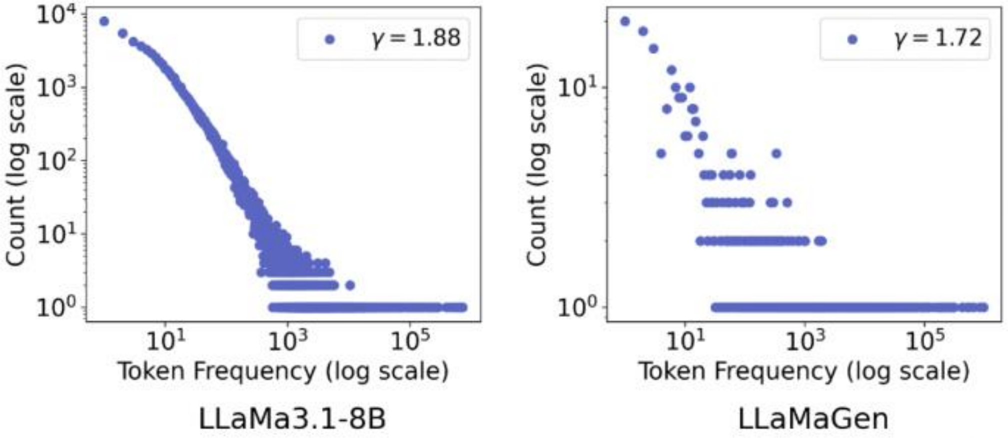




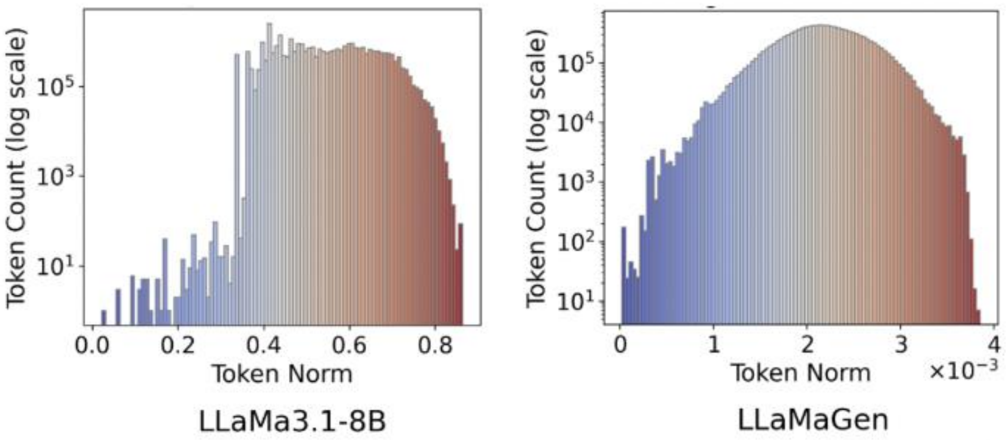
Scale-Free Property in Token Relationships

- Scale-free property across foundation models and modalities
  - Very few (exponentially) tokens appear very frequently/have large norm

Token Frequency (x-axis) v.s. Token count (y-axis)
“How many tokens appears *x* number of times”

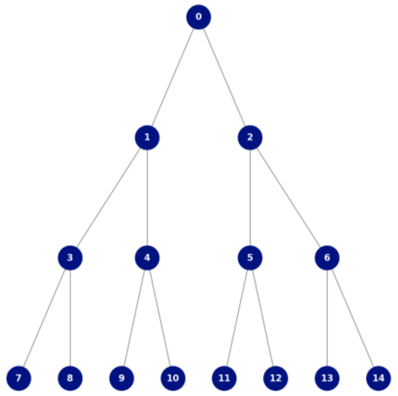


Token norm (x-axis) v.s. Token count (y-axis)
“How many tokens have a norm of value *x*”

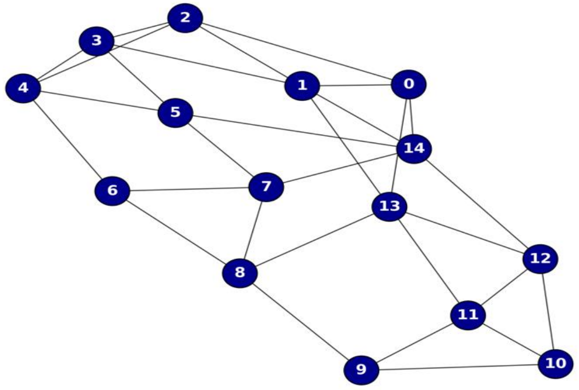


Corpus: RedPajama (subset) (arXiv, C4, Common Crawl, GitHub, Wikipedia, and StackExchange); Mathematical Reasoning (GSM8K, MATH50K, MAWPS, SVAMP); Common Sense Reasoning (BoolQ, WinoGrande, OpenBookQA)

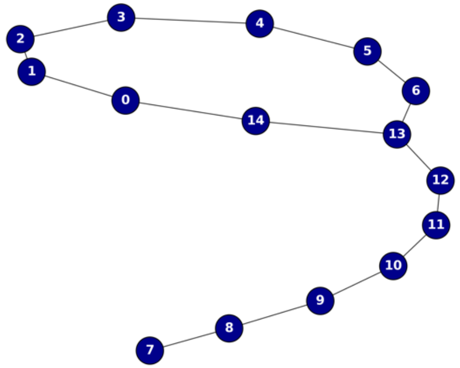
Quantitate Analysis: Hyperbolicity



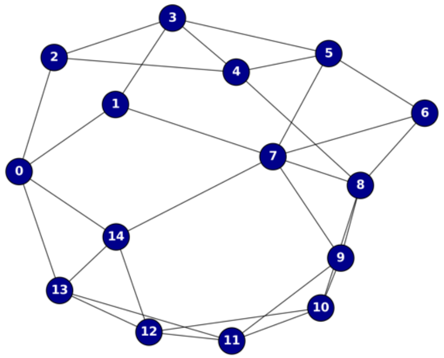
Hyperbolicity(∂)=0



Hyperbolicity(∂)=0.5



Hyperbolicity(∂)=0.25



Hyperbolicity(∂)=0.75

Hyperbolicity quantifies the distance of a graph from a tree-like structure

∂ = 0, tree-like structure, no cycles.

∂ = 0.25, one cycle, slight deviation from tree metric.

∂ = 0.5, moderate interconnectedness, more loops.

∂ = 0.75, dense structure, multiple loops, far from a tree.

Smaller hyperbolicity indicates fewer cycles,

with certain nodes playing crucial roles.

Hierarchies in LLM Token Distribution

- Hyperbolicity (0-1): measures how much data points are tree-like (hierarchical)
  - Lower values indicate more hierarchical distribution

Table 2. δ-Hyperbolicity of the token embedding in various LLMs across several datasets.

Model	arXiv	C4	Common Crawl	GitHub	StackExchange	Wikipedia
RoBERTa-Base (Liu et al., 2019b)	0.15 ± 0.06	0.18 ± 0.04	0.17 ± 0.04	0.12 ± 0.04	0.17 ± 0.07	0.07 ± 0.05
LLaMA3.1-8B (Grattafiori et al., 2024)	0.15 ± 0.05	0.16 ± 0.07	0.15 ± 0.06	0.12 ± 0.05	0.18 ± 0.06	0.10 ± 0.04
GPT-NeoX-20B (Black et al., 2022)	0.14 ± 0.03	0.17 ± 0.06	0.15 ± 0.05	0.11 ± 0.04	0.14 ± 0.04	0.09 ± 0.03
Gemma2-9B (Team et al., 2024)	0.17 ± 0.06	0.19 ± 0.04	0.20 ± 0.05	0.15 ± 0.05	0.18 ± 0.04	0.15 ± 0.03

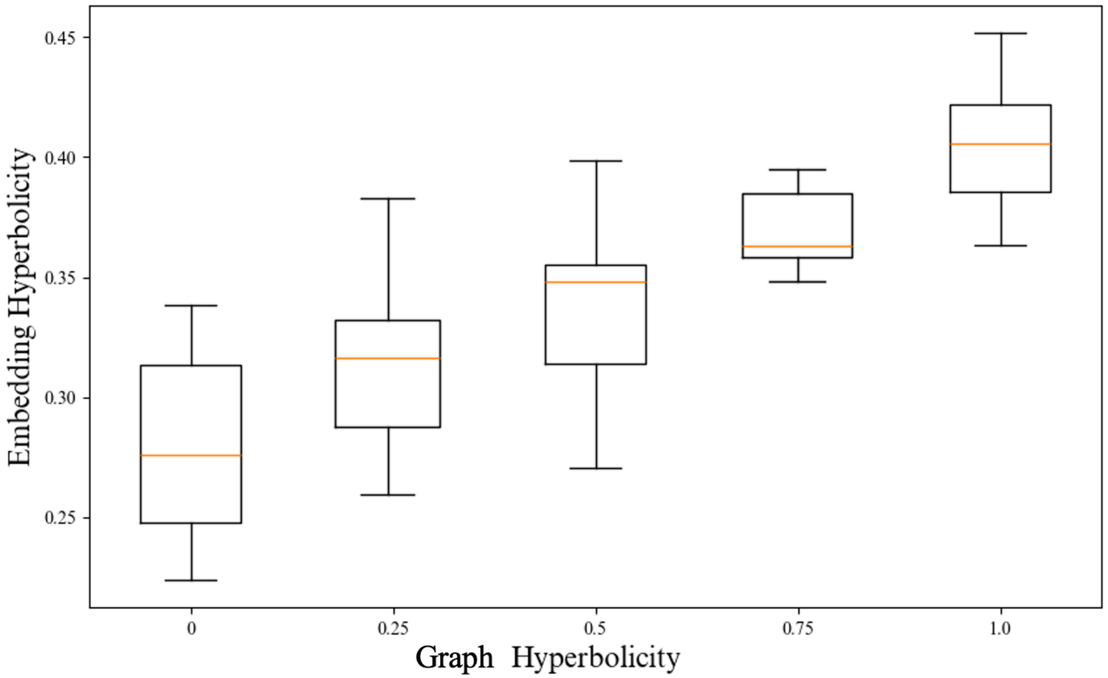
Indicates hierarchical structure in token distribution

Reference values

Table 3. Hyperbolicity values δ for different metric spaces.

	Sphere Space	Dense Graph	PubMed Graph	Poincare Space	Tree Graph
δ	0.99 ± 0.01	0.62 ± 0.01	0.40 ± 0.04	0.14 ± 0.01	0.0

Embedding Hyperbolicity vs Graph Hyperbolicity



Positive correlation between graph hyperbolicity and embedding hyperbolicity

Compute token embedding hyperbolicity as a proxy for structure; lower values indicate a more tree-like shape.

Embedding Norm vs Token Frequency

Table 7: Mean, Minimum, and Maximum Norm Values for Different Models and Groups

Model	Group	Norm (Mean (Min∼Max))
LLaMA-7B	Group 1: *to, have, in, that, and, is, for*	0.95 (0.79∼1.06)
	Group 2: *how, much, many, time, cost*	1.22 (1.12∼1.30)
	Group 3: *animals, fruit, numbers, items, colors*	1.36 (1.32∼1.43)
	Group 4: *dog, cow, apple, hours, dollars, minute, second, shoes, purple, bananas, puppies*	1.37 (1.31∼1.44)
LLaMA-13B	Group 1: *to, have, in, that, and, is, for*	1.03 (0.83∼1.26)
	Group 2: *how, much, many, time, cost*	1.43 (1.35∼1.49)
	Group 3: *animals, fruit, numbers, items, colors*	1.50 (1.46∼1.54)
	Group 4: *dog, cow, apple, hours, dollars, minute, second, shoes, purple, bananas, puppies*	1.50 (1.47∼1.57)
Gemma-7B	Group 1: *to, have, in, that, and, is, for*	3.16 (3.06∼3.30)
	Group 2: *how, much, many, time, cost*	3.56 (3.49∼3.63)
	Group 3: *animals, fruit, numbers, items, colors*	3.84 (3.71∼3.92)
	Group 4: *dog, cow, apple, hours, dollars, minute, second, shoes, purple, bananas, puppies*	4.03 (3.43∼4.82)
LLaMA3-8B	Group 1: *to, have, in, that, and, is, for*	0.35 (0.33∼0.40)
	Group 2: *how, much, many, time, cost*	0.46 (0.39∼0.50)
	Group 3: *animals, fruit, numbers, items, colors*	0.53 (0.51∼0.55)
	Group 4: *dog, cow, apple, hours, dollars, minute, second, shoes, purple, bananas, puppies*	0.59 (0.50∼0.70)

Embeddings Space Choices

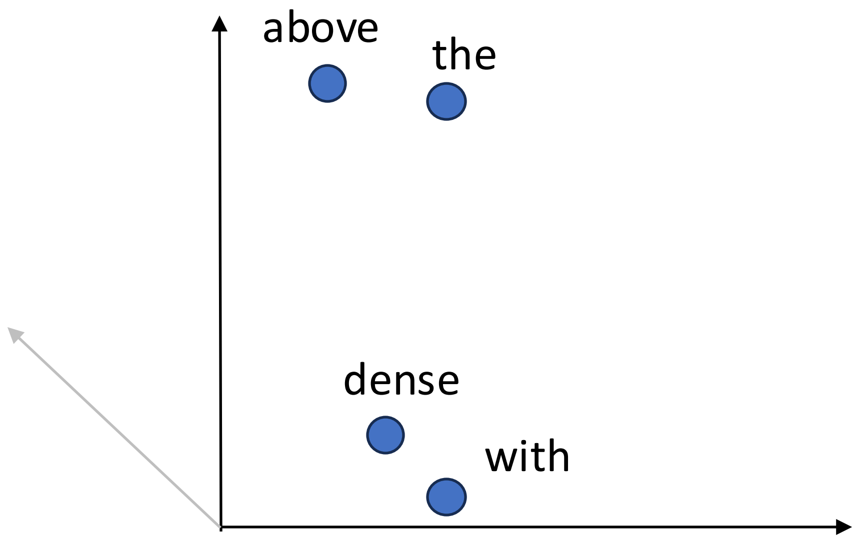
- The **embedding space** is crucial for a model to faithfully represent such relationships between data points
  - Should Euclidean geometry remain the de facto choice for foundation models?

Embeddings Space Intuition

	above	dense	flows	forest	is	rises	river	sun	tall	the	through	trees	with
above	0	0	0	0	0	0	0	0	0	1	0	0	0
dense	0	0	0	0	0	0	0	0	0	0	0	0	1

Intuition: Co-occurring words should be embedded closer together!

Embedding space $\mathbb{R}^n$



Issues with Euclidean Embeddings: Distortion

- Euclidean space leads to ***significant distortion*** regardless of the embedding dimensions

Theorem

(Informal; Lee et al., (2007)) There is a lower bound in the minimal distortion of embedding hierarchical structures (e.g. token relationships) into Euclidean space ($\mathbb{R}^n$).

"There is a ***performance bottleneck*** on how well Euclidean foundation models can represent complex token relationships"

Issues with Euclidean Embeddings: Dimension Dilemma

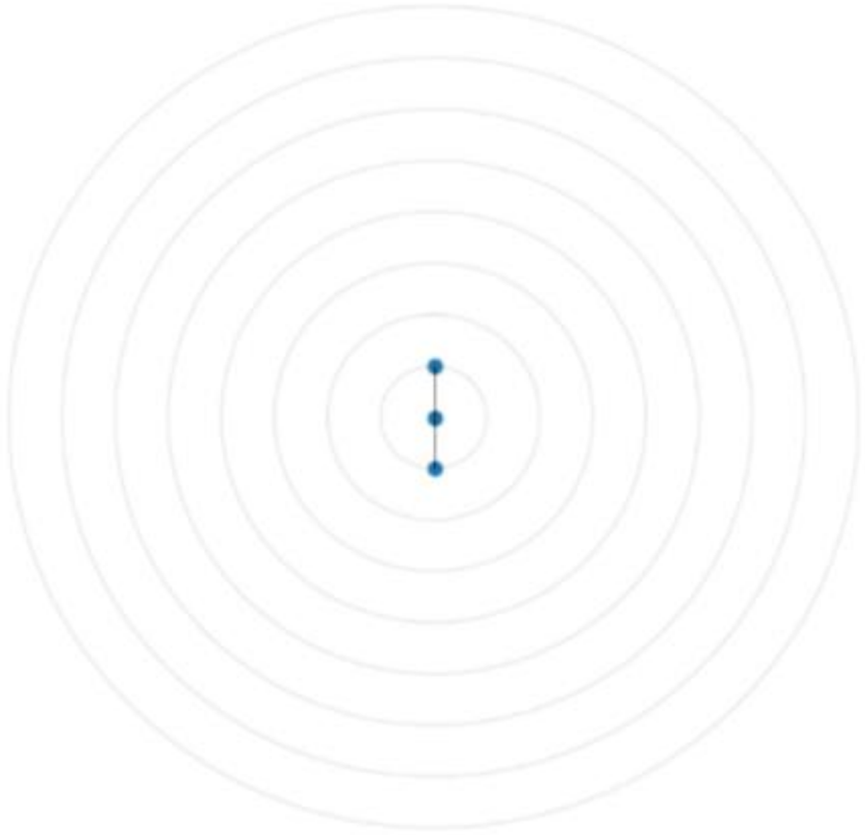
- Euclidean space face the dilemma of **dimension-distortion tradeoffs**
  - High dimensionality is often required to embed complex token relations in Euclidean space with (relatively) low distortion

Theorem

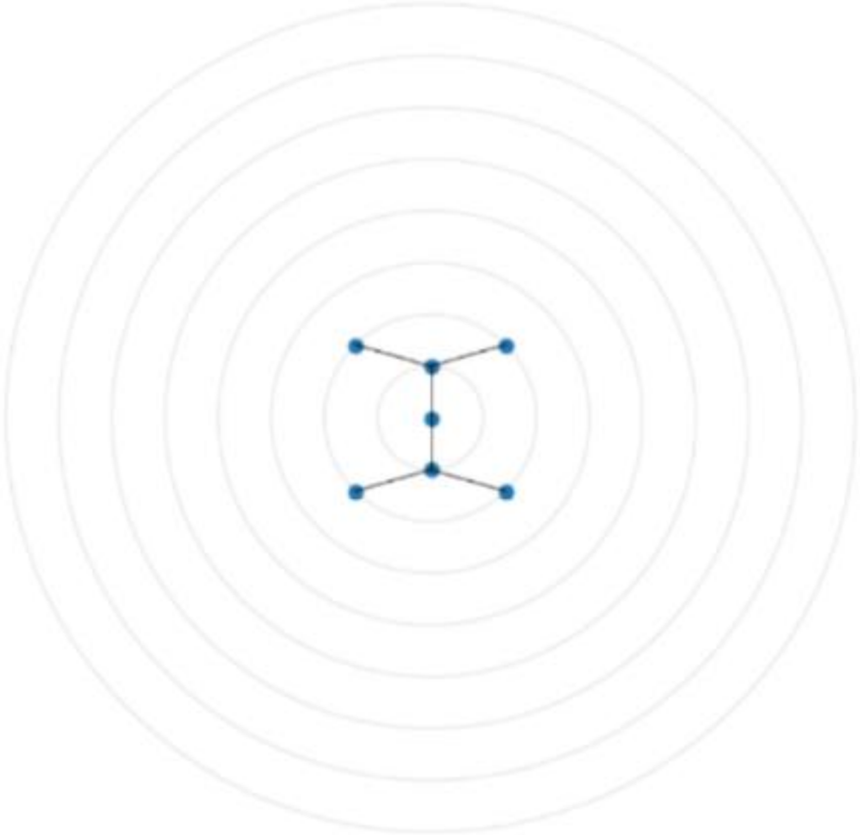
(Informal; Matoušek (2002)) The dimension required when embedding unweighted graphs (in the form of token relationships/self-attention) grows **near-quadratically** *w.r.t to distortion.*

"Euclidean foundation models have ***limited scalability***"

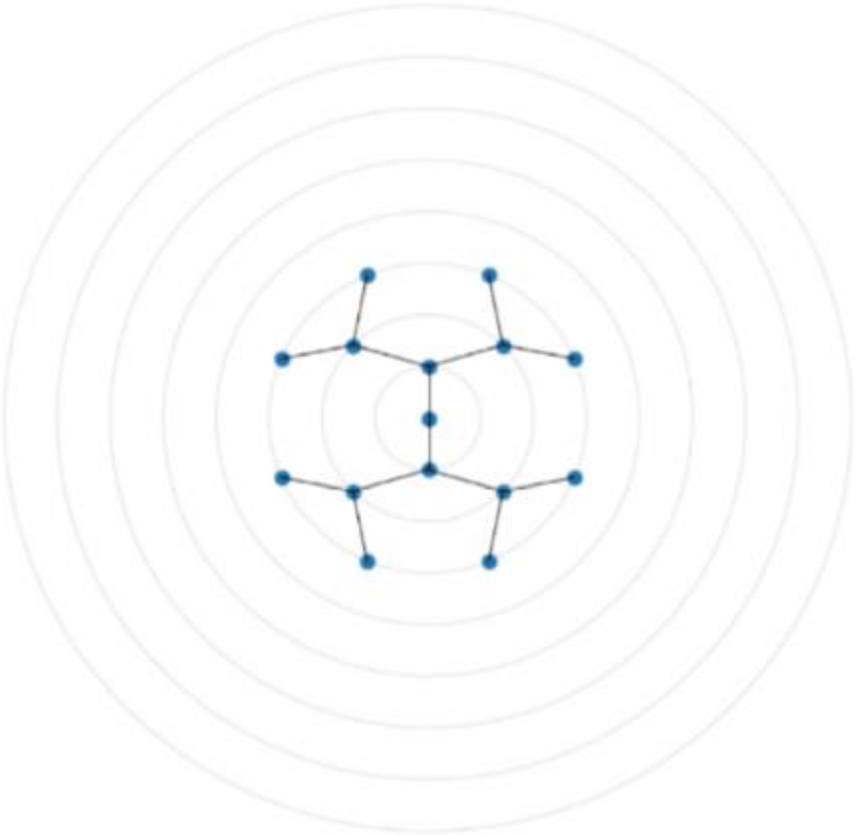
Example: Embedding Tree-structured Data



Example: Embedding Tree-structured Data

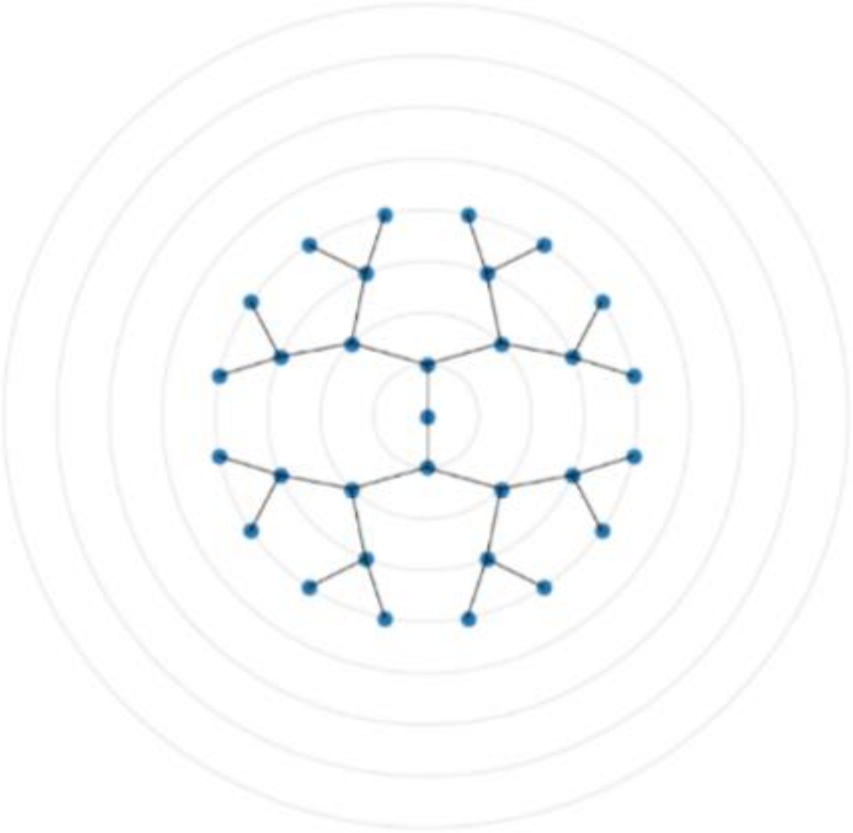


Example: Embedding Tree-structured Data

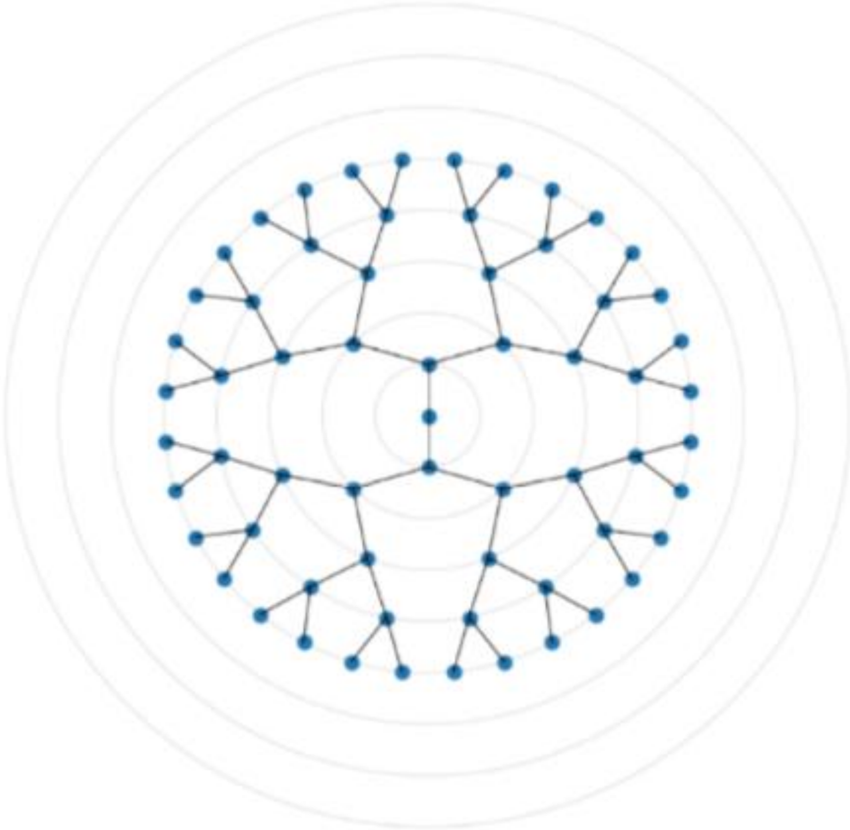


So far, so good
Nodes are close i.f.f. they are connected by an edge

Example: Embedding Tree-structured Data



Example: Embedding Tree-structured Data

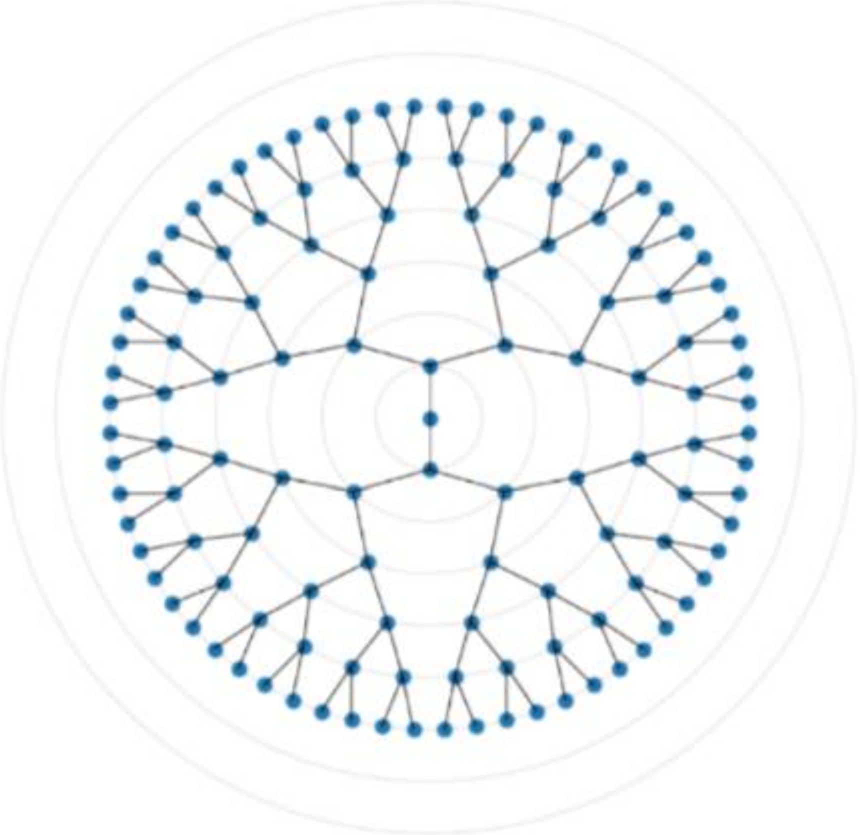


But the outermost nodes are becoming increasingly close to one another.

….

Even though they are not connected by an edge in the graph.

Example: Embedding Tree-structured Data

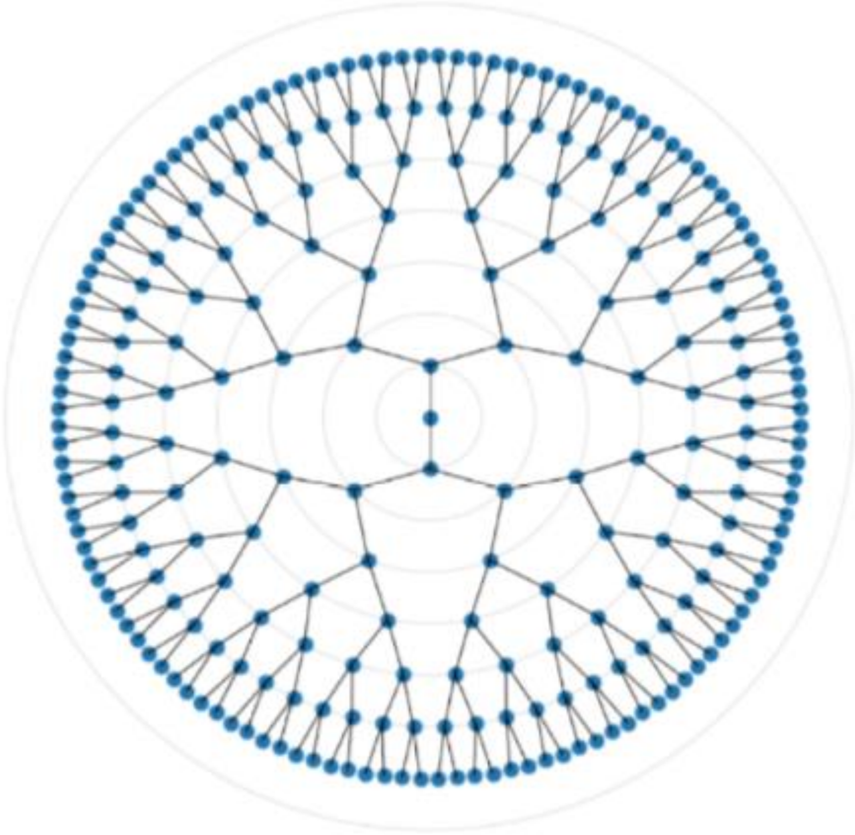


But the outermost nodes are becoming increasingly close to one another.

….

Even though they are not connected by an edge in the graph.

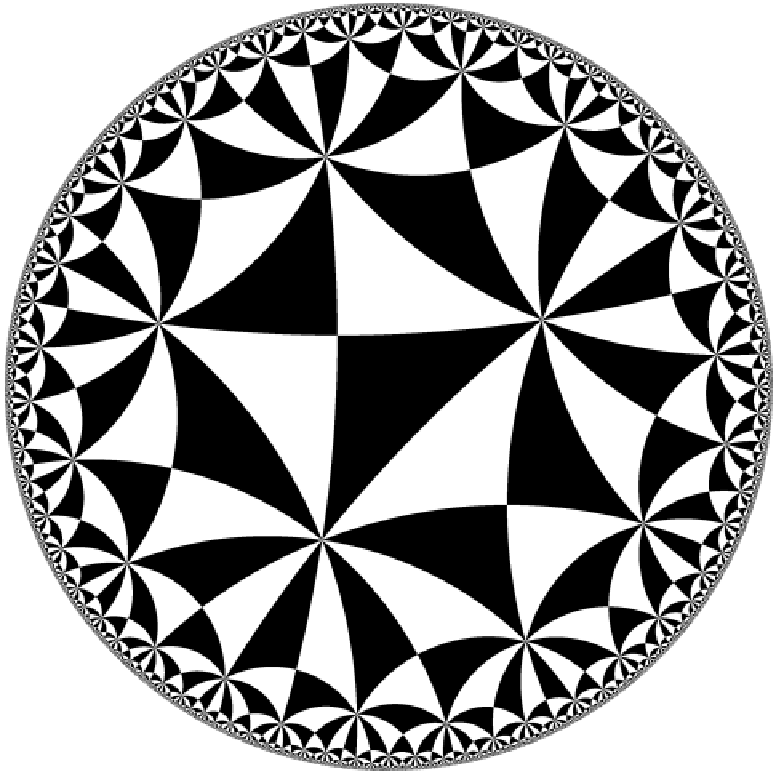
Example: Embedding Tree-structured Data



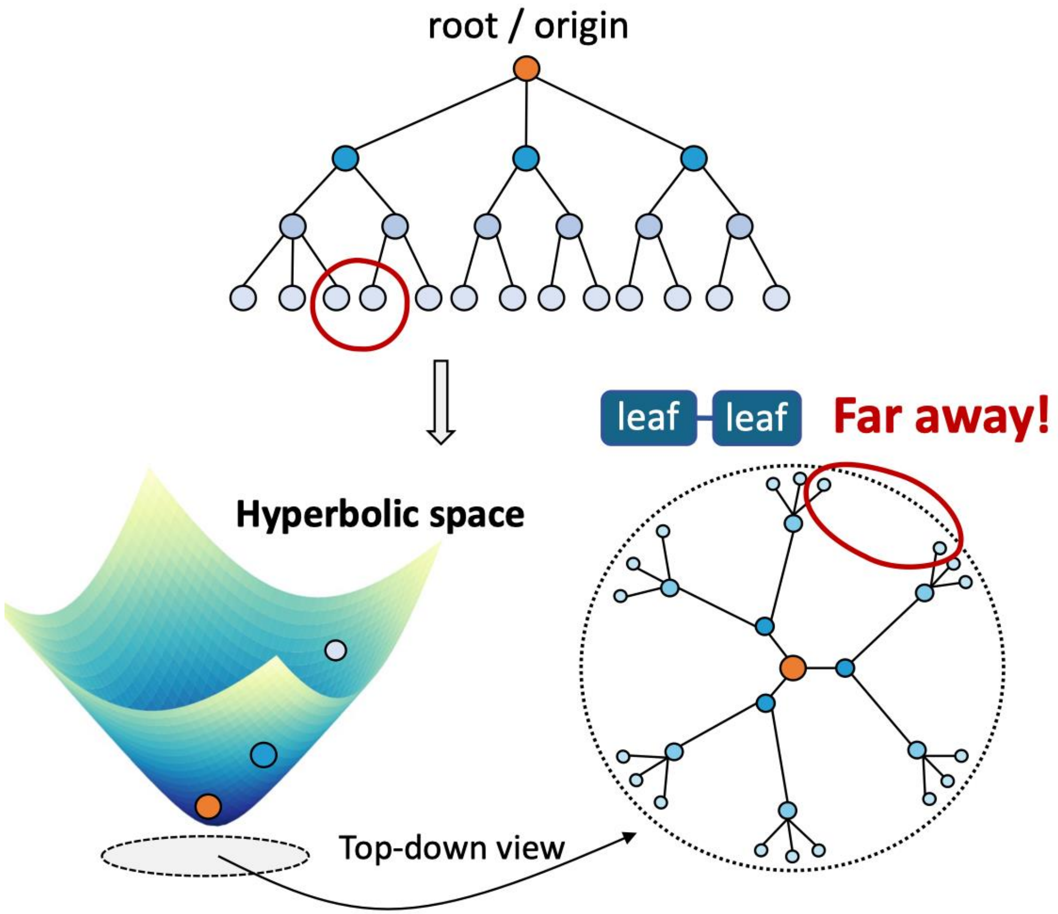
Things only get worse!
We have lost our property:

“close i.f.f share edge”

Potential Solution: Hyperbolic Embedding Space



The volume of a ball in the hyperbolic space grows **exponentially** with its radius

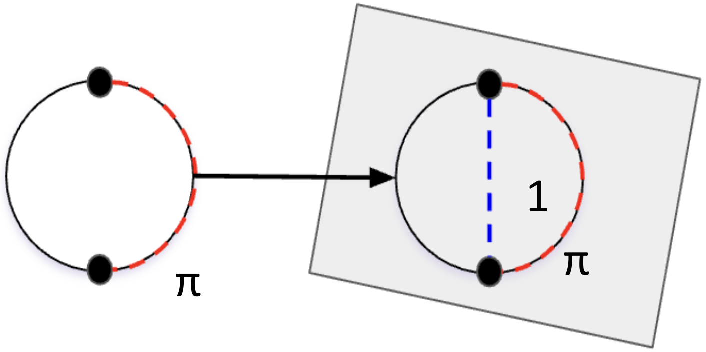


Euclidean Embedding: Common Misunderstanding

- Nash Embedding Theorem (and similar): roughly, any n-dimensional Riemannian manifold can be embedded in R^{2n}
  - This is an embedding of ***manifolds*** instead of ***metric spaces***, i.e. distance is still globally distorted

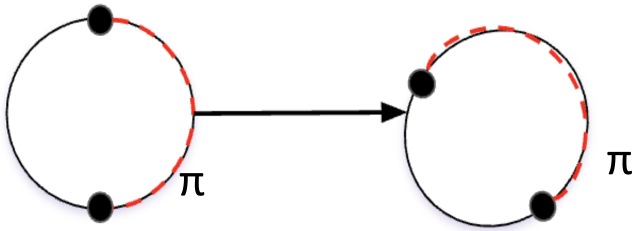
Isometric Embedding of Manifolds

- Shortest path between points are not necessarily the same globally
- e.g. Embedding sphere in Euclidean space



Isometric Embedding of Metric Spaces

- Distance between any two points (global behavior) is preserved in the new space
- e.g. Rotation

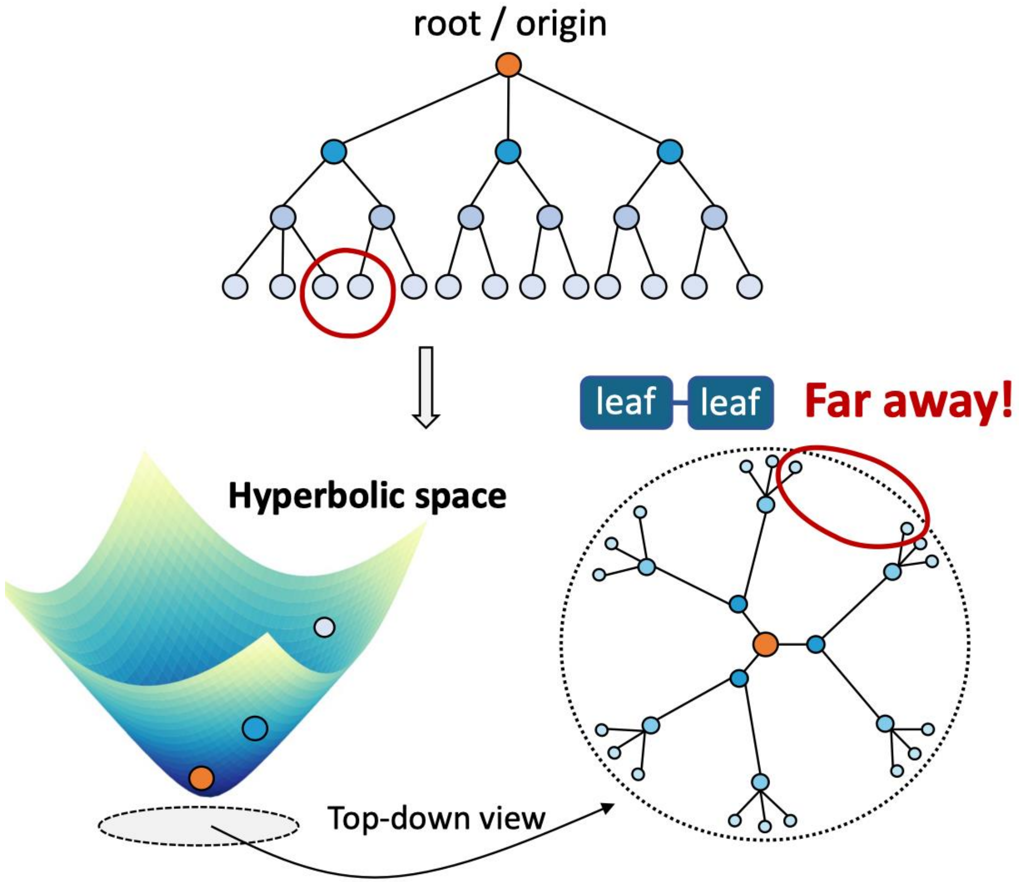


Hyperbolic Geometry for Foundation Models

We need an embedding space that can ***better represent token relationship***!

- The distance between low-level tokens on different branches should be maximized and far away
- The distance between a high-level token and a low-level token should be minimized and close

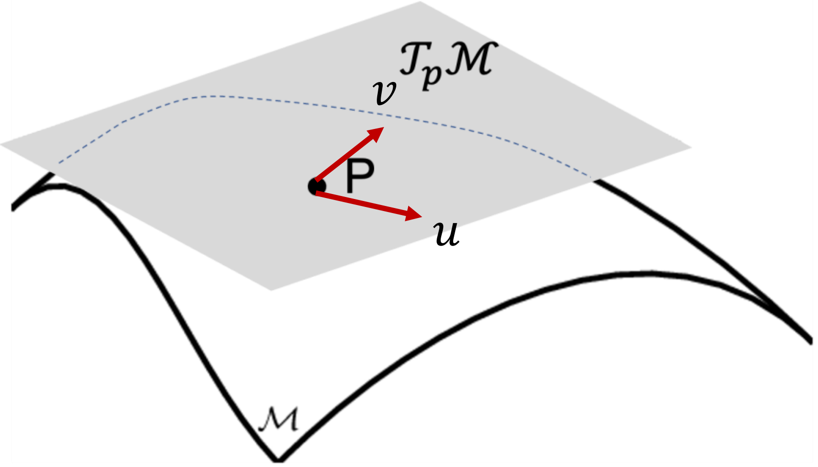
- Solution: any tree (i.e. hierarchical distribution) can be embedded into ***hyperbolic space*** with ***arbitrarily low distortion***!!



Riemannian Manifold

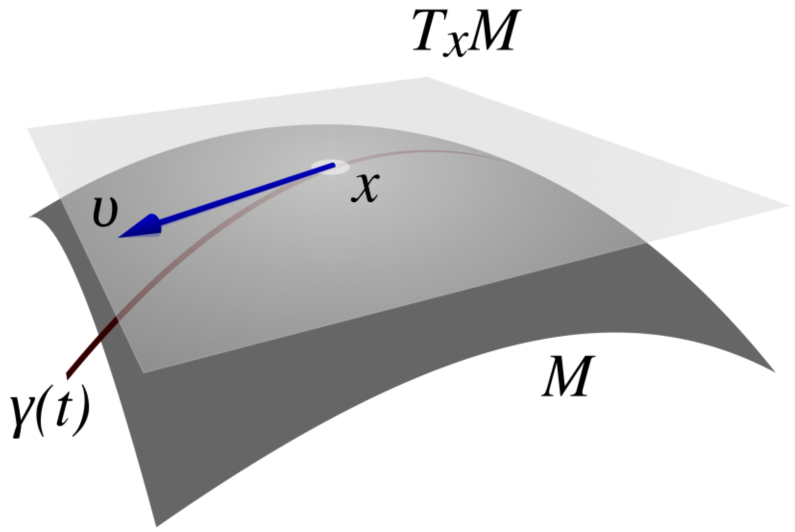
- **Manifold**: high-dimensional surface
- **Riemannian Manifold $\mathcal{M}$**
  - Equipped with
    - *Tangent space $\mathcal{T}_p\mathcal{M}$*: an $\mathbb{R}^d$ that approximates the manifold at any point $p \in \mathcal{M}$
    - *Inner product g_p*: $\mathcal{T}_p\mathcal{M} \times \mathcal{T}_p\mathcal{M} \to \mathbb{R}$
  - Both functions vary smoothly (differentiable) on the manifold

$u, v \in \mathcal{T}_p\mathcal{M},\ g_p(u, v) \in \mathbb{R}$



Tangent Space

- **Curve:** smooth path along manifold $\gamma: [0,1] \to \mathcal{M}$
- **Speed:** direction of change along the curve $\dot{\gamma}: [0,1] \to \mathcal{T}_x\mathcal{M}$
- **Tangent space $\mathcal{T}_x\mathcal{M}$:** space of **speed vectors $\boldsymbol{v}$** of all curves γ that **go through point $\boldsymbol{x}$** on the manifold $\mathcal{M}$

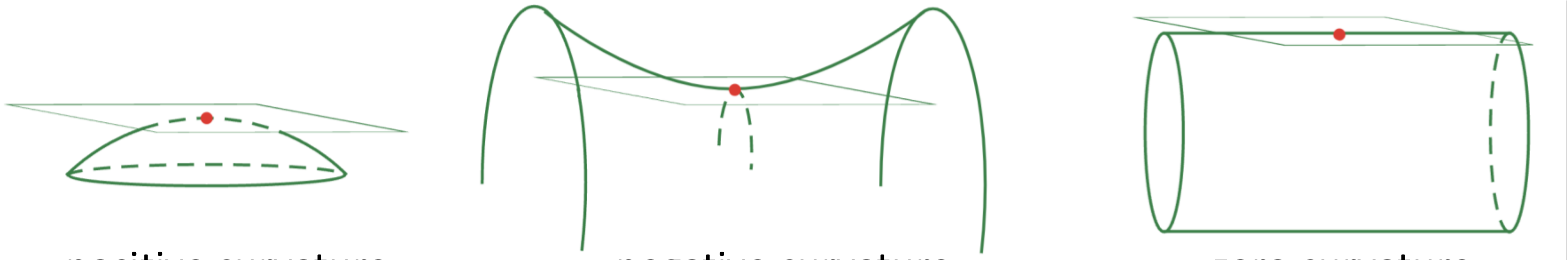


Curvature

- The **curvature** (sectional curvature) at a point measures how drastically a surface **bends away** from its tangent plane at this point

High-level Intuition:

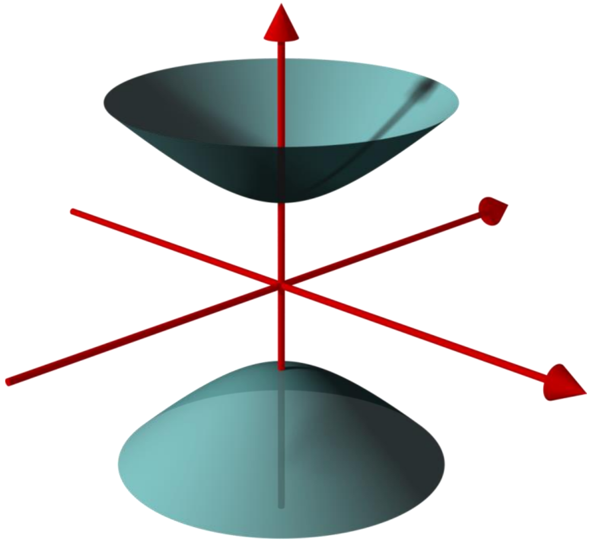
- If the surface locally lives **entirely on one side** of the tangent space $\mathcal{T}_p\mathcal{M}$ ⇒ **Positive** curvature at point p
- If the tangent space $\mathcal{T}_p\mathcal{M}$ **cuts through** the surface ⇒ **Negative** curvature at point p
- If the surface has a line along which the **surface agrees with the tangent space** $\mathcal{T}_p\mathcal{M}$ ⇒ **Zero** curvature at point p



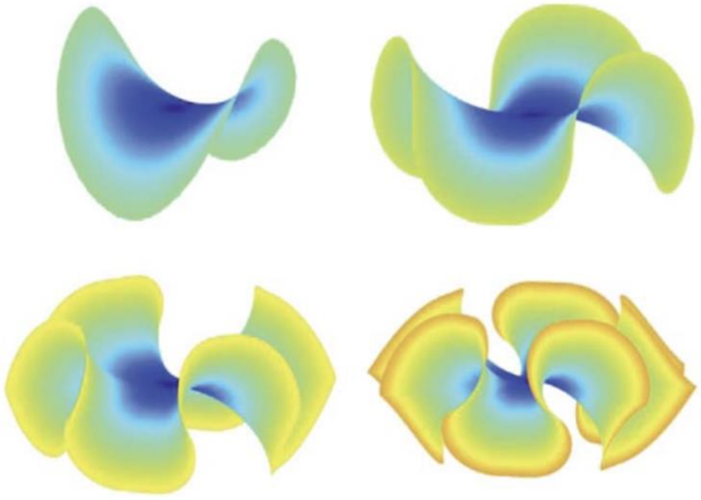
positive curvature negative curvature zero curvature

Hyperbolic Space

- **Hyperbolic space** is a Riemannian manifold with constant negative curvature $-1/K$, where $(K > 0)$
  - Becomes Euclidean when $K \to \infty$
- In **Euclidean space**, we can also find manifolds with constant negative curvature:



two sheet hyperboloid (source: Wikipedia)

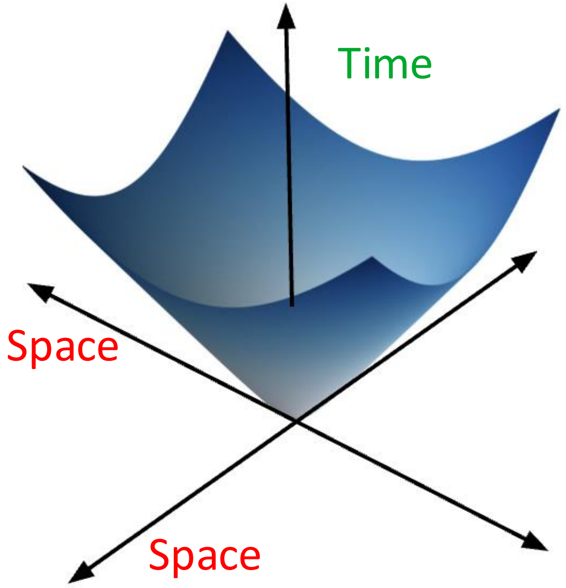


Periodic Amsler Surfaces

Hyperbolic Space and Minkowski Space

- Hyperbolic space can be naturally embedded into a **Minkowski Space**

- The **Minkowski metric** in the Minkowski space is different from the Euclidean metric.
  - **Euclidean Metric:** $g_E(\boldsymbol{u}, \boldsymbol{v}) = u_0 v_0 + u_1 v_1 + \cdots + u_d v_d$
  - **Minkowski Metric:** $g_M(\boldsymbol{u}, \boldsymbol{v}) = \pm(u_0 v_0 - u_1 v_1 - \cdots - u_d v_d)$
    - Without loss of generality we can take the $+$ sign
  - Note: dimension 1 is treated differently in Minkowski Space.



Inner Product

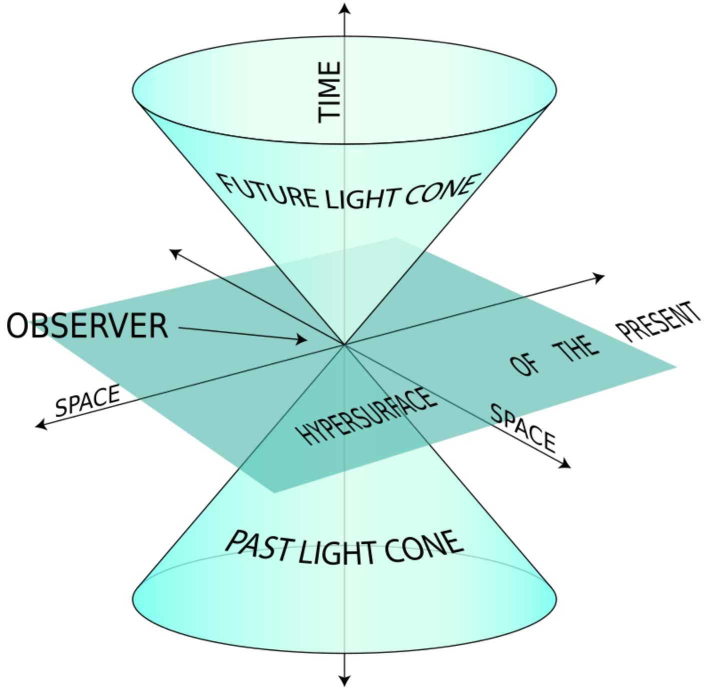
- **Hyperboloid model** as a Riemannian manifold:
  - With Constant **Minkowski metric**:

$$\langle .,. \rangle_{\mathcal{L}} : \mathbb{R}^{d+1} \times \mathbb{R}^{d+1} \to \mathbb{R}$$

$$\langle \boldsymbol{x}, \boldsymbol{y} \rangle_{\mathcal{L}} = -x_0 y_0 + x_1 y_1 + \ldots + x_d y_d$$

Time-like ($-x_0 y_0$)

Space-like ($x_1 y_1 + \ldots + x_d y_d$)

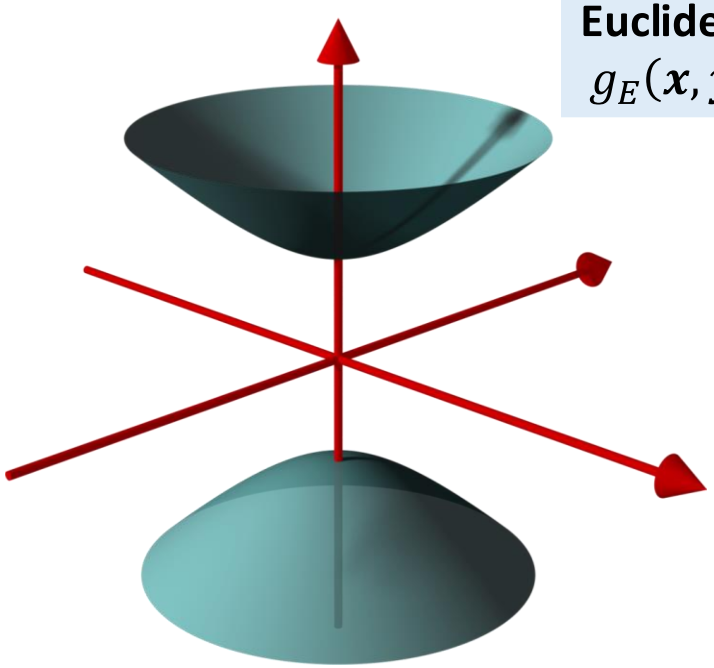


- **Hyperboloid model** $\mathbb{H}^{d,K} = \{\boldsymbol{x} \in \mathbb{R}^{d+1}: \langle \boldsymbol{x}, \boldsymbol{x} \rangle_{\mathcal{L}} = -K\}$, $-\frac{1}{K}$ is the curvature
- Note: the points in hyperboloid model $\mathbb{H}^{d,K}$ are represented in $(d+1)$-dimensional Minkowski space.
- The metric of hyperboloid model is different from the Euclidean metric!

Hyperboloid in Different Spaces

Euclidean Metric

$g_E(\boldsymbol{x}, \boldsymbol{y}) = x_1 y_1 + x_2 y_2 + x_3 y_3$



Two sheet hyperboloid in **3D Euclidean space**

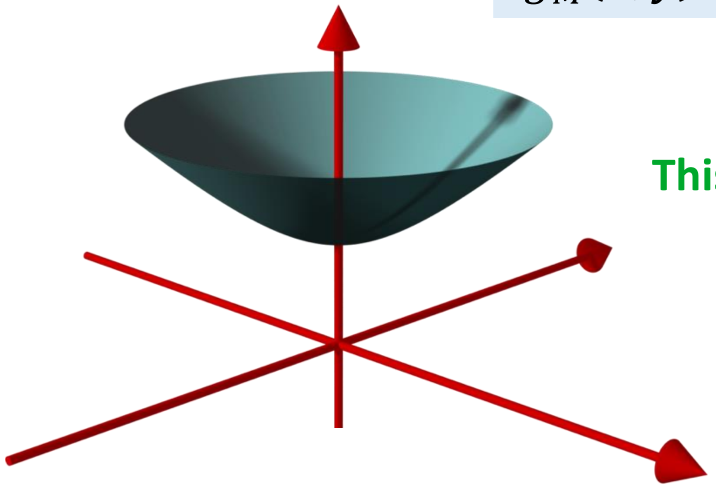
Geodesic distance in Euclidean hyperboloid:

$d_E(\boldsymbol{x}, \boldsymbol{y}) = \sqrt{2(1 - g_E(\boldsymbol{x}, \boldsymbol{y}))}$

(with normalized $\boldsymbol{x}$ and $\boldsymbol{y}$)

Minkowski Metric

$g_M(\boldsymbol{x}, \boldsymbol{y}) = -x_1 y_1 + x_2 y_2 + x_3 y_3$



This is hyperbolic

2D Hyperboloid model in **3D Minkowski space**

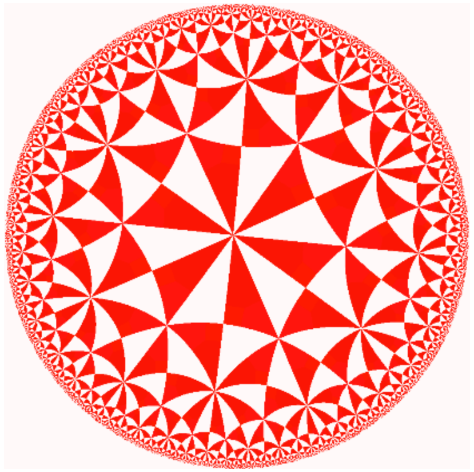
Geodesic distance in Minkowski hyperboloid:

$D_M^K(\boldsymbol{x}, \boldsymbol{y}) = \sqrt{K} \operatorname{arcosh}(-\frac{g_M(\boldsymbol{x}, \boldsymbol{y})}{K})$

Performing deep learning operations in hyperbolic space is non-trivial

Poincaré Model

- **Poincaré Model**
  - Radius proportional to $\sqrt{K}$ ($-\frac{1}{K}$ is the curvature)
  - Open ball (exclude boundary)
  - Each triangle in the figure has the **same** area
  - **Exponentially many triangles** with the same area towards the boundary of Poincaré Ball

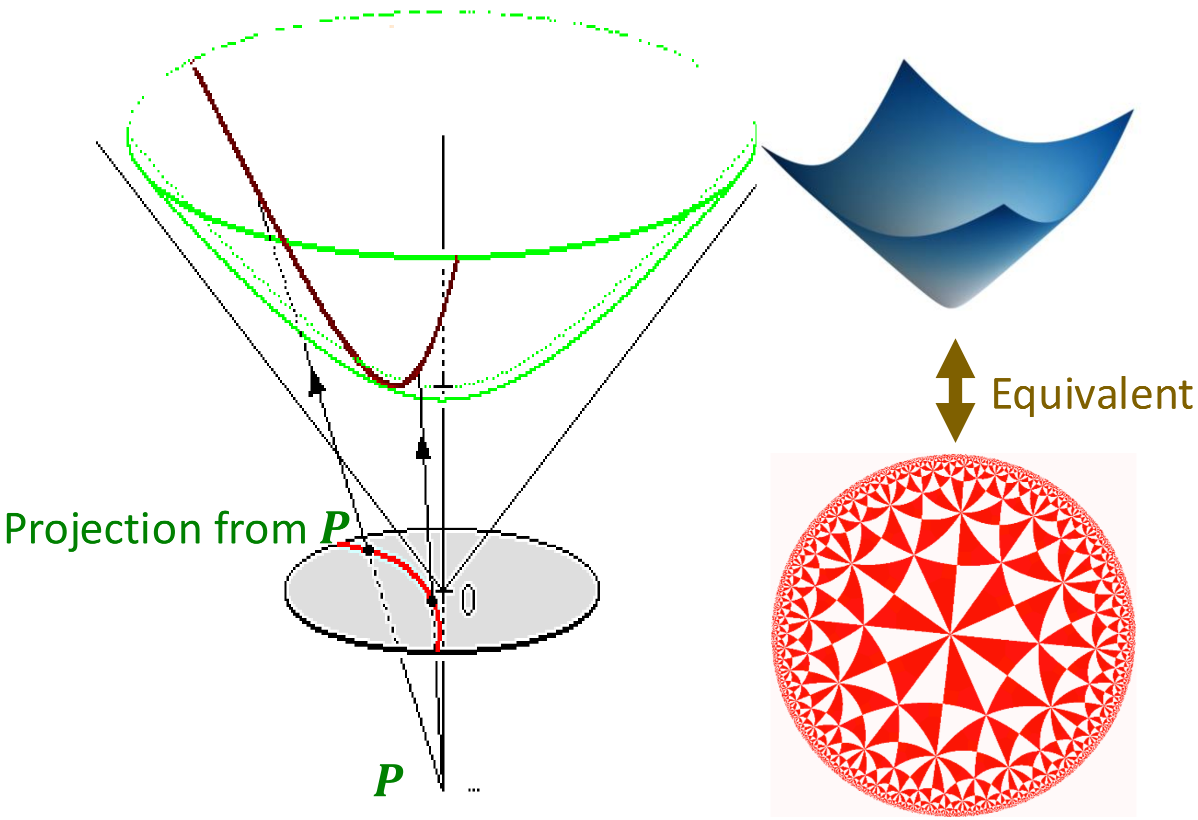


Poincaré: intuitive visualization

Other models exist as well, e.g. Klein model

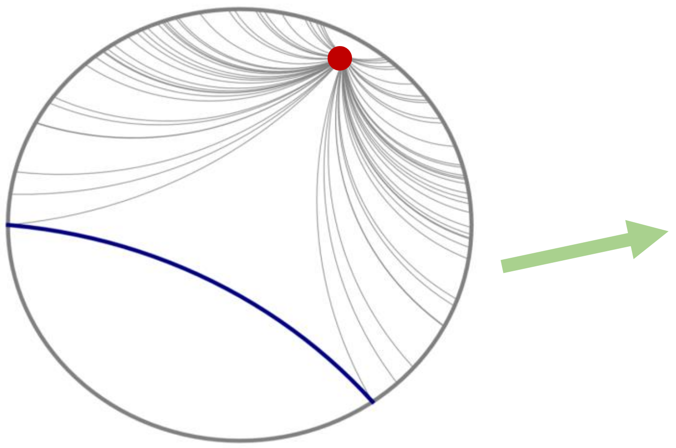
Equivalence

- d-dimensional Poincaré model and $(d+1)$-dimensional hyperboloid model are **equivalent**!
- 2d Poincaré model can be derived using a **projection** of 3d hyperboloid model through a specific point onto the unit circle of the $z = 0$ plane.



Geodesic

- **Geodesic**: shortest path in manifold
  - Analogous to straight lines in $\mathbb{R}^n$
  - Curved in hyperbolic space
- Geodesics visualization in Poincaré model: curved!



Set of geodesic lines from the red point to boundary of the Poincare ball that are parallel to the blue line

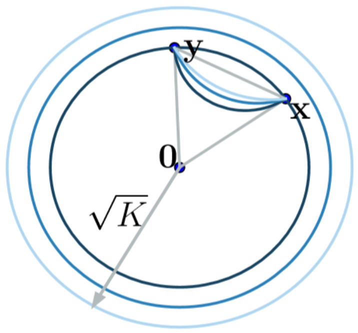
Geodesic Distance

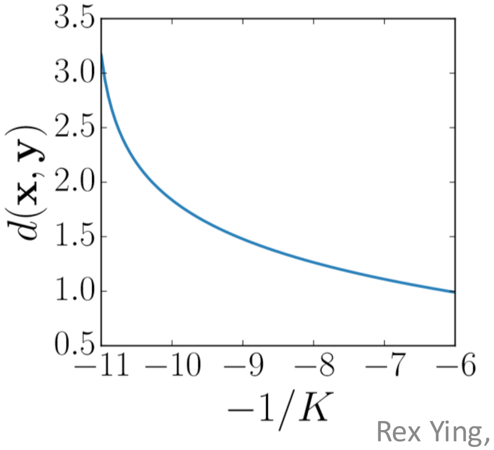
- **Geodesic distance** between $\boldsymbol{x}$ and $\boldsymbol{y}$ for $\mathbb{H}^{d,\mathrm{K}}$:

$$D_{\mathcal{L}}^{K}(\boldsymbol{x},\boldsymbol{y}) = \sqrt{K}\operatorname{arcosh}(-\frac{\langle \boldsymbol{x},\boldsymbol{y}\rangle_{\mathcal{L}}}{K})$$

- Negative Lorentz Distance: $D_{\mathcal{L}}^{K}(\boldsymbol{x},\boldsymbol{y}) = \frac{1}{K} - 2\langle \boldsymbol{x},\boldsymbol{y}\rangle_{\mathcal{L}}$
- The more negative the curvature:
  - the more geodesics bends inward
  - geodesic distance increases

$$\operatorname{arcosh}(x) = \ln(x + \sqrt{x^2+1})$$

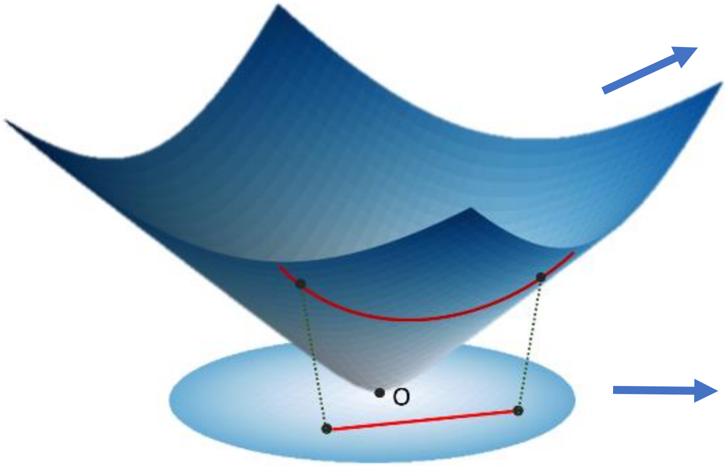




Dark blue: high curvature boundary and geodesics
Light blue: low curvature boundary and geodesics

Tangent Space

- Tangent space expression under **hyperboloid model** $\mathbb{H}^{d,K}$ at point $\boldsymbol{x}$:
  - $\mathcal{T}_{\boldsymbol{x}}\mathbb{H}^{d,K} = \{\boldsymbol{v} \in \mathbb{R}^{d+1} \colon \langle \boldsymbol{v}, \boldsymbol{x} \rangle_{\mathcal{L}} = 0\}$
- A vector space (linear structure) with the same dimension as the hyperboloid model: it is Euclidean!
- The best linear approximation to the manifold $\mathbb{H}^{d,K}$ at point $\boldsymbol{x}$



Hyperboloid model

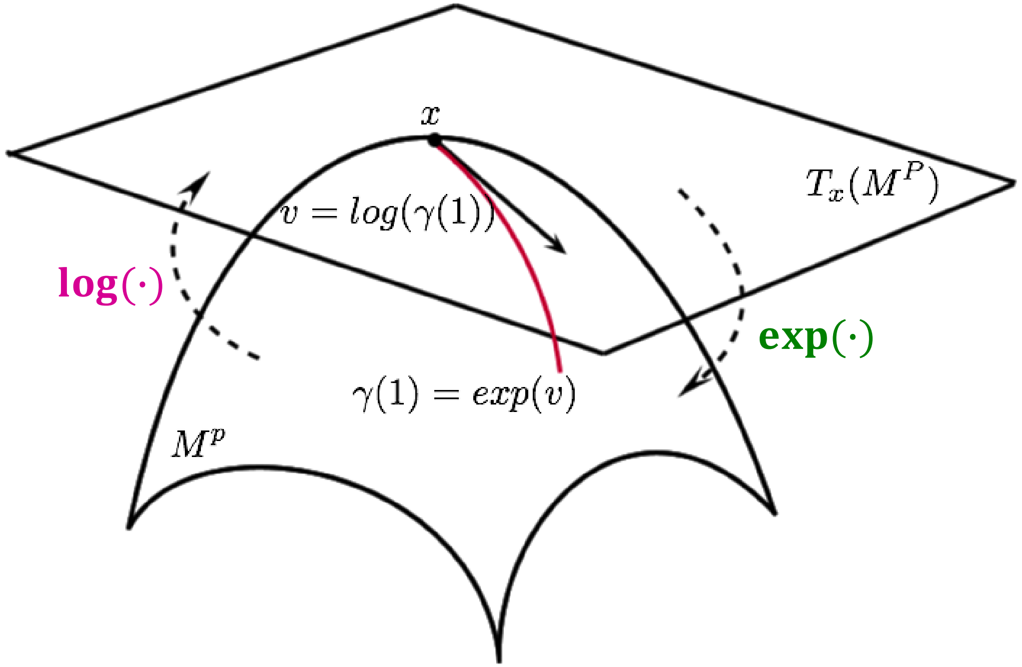
$\boldsymbol{d}$: hyperbolic space dimension

$\boldsymbol{K}$: negative inverse of curvature

Tangent space at north pole o

Mapping to and from Tangent Space

- **Exponential map**: $\mathcal{T}_{\boldsymbol{x}}\mathbb{H}^{\mathrm{d,K}} \rightarrow \mathbb{H}^{\mathrm{d,K}}$
  - from tangent space (Euclidean) to manifold
- **Logarithmic map**: $\mathbb{H}^{\mathrm{d,K}} \rightarrow \mathcal{T}_{\boldsymbol{x}}\mathbb{H}^{\mathrm{d,K}}$
  - from manifold to tangent space
  - inverse operation of exponential map

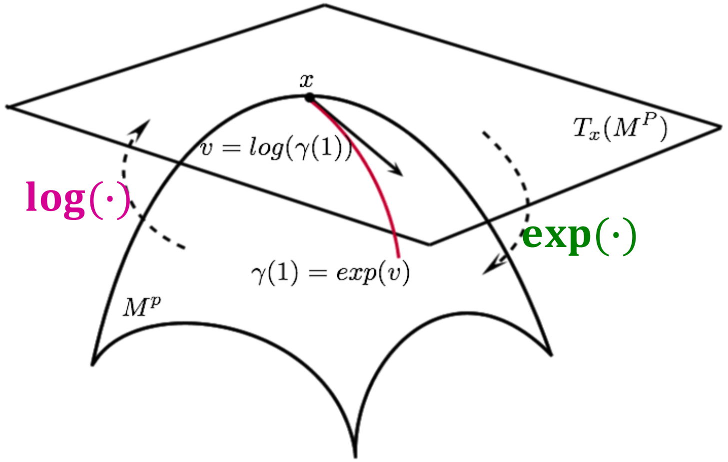


Exponential Map:

- For **hyperboloid model** $\mathbb{H}^{\mathrm{d,K}} = \{\boldsymbol{x} \in \mathbb{R}^{d+1}: \langle \boldsymbol{x}, \boldsymbol{x} \rangle_{\mathcal{L}} = -K\}$ at point $\boldsymbol{x}$
- **Exponential Map:**

$$\exp_{\boldsymbol{x}}^{K}(\boldsymbol{v}) = \cosh\left(\frac{\|\boldsymbol{v}\|_{\mathcal{L}}}{\sqrt{K}}\right)\boldsymbol{x} + \sqrt{K}\sinh\left(\frac{\|\boldsymbol{v}\|_{\mathcal{L}}}{\sqrt{K}}\right)\frac{\boldsymbol{v}}{\|\boldsymbol{v}\|_{\mathcal{L}}}$$

  - $\boldsymbol{v} \in \mathcal{T}_{\boldsymbol{x}}\mathbb{H}^{\mathrm{d,K}}$
  - $\cosh(x) = \frac{e^{x}+e^{-x}}{2}$, $\sinh(x) = \frac{e^{x}-e^{-x}}{2}$
  - $\|\boldsymbol{v}\|_{\mathcal{L}} = \langle \boldsymbol{v}, \boldsymbol{v} \rangle_{\mathcal{L}}$

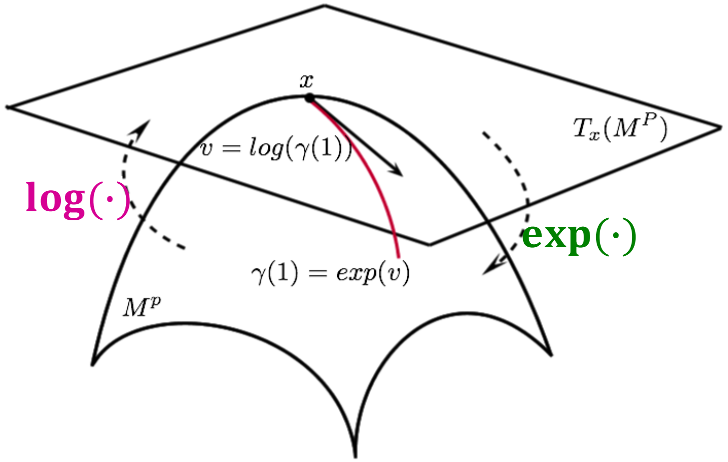


Logarithmic Map

- For **hyperboloid model** $\mathbb{H}^{\mathrm{d,K}} = \{\boldsymbol{x} \in \mathbb{R}^{d+1}: \langle \boldsymbol{x}, \boldsymbol{x} \rangle_{\mathcal{L}} = -K\}$ at point $\boldsymbol{x}$
- **Logarithmic map**:

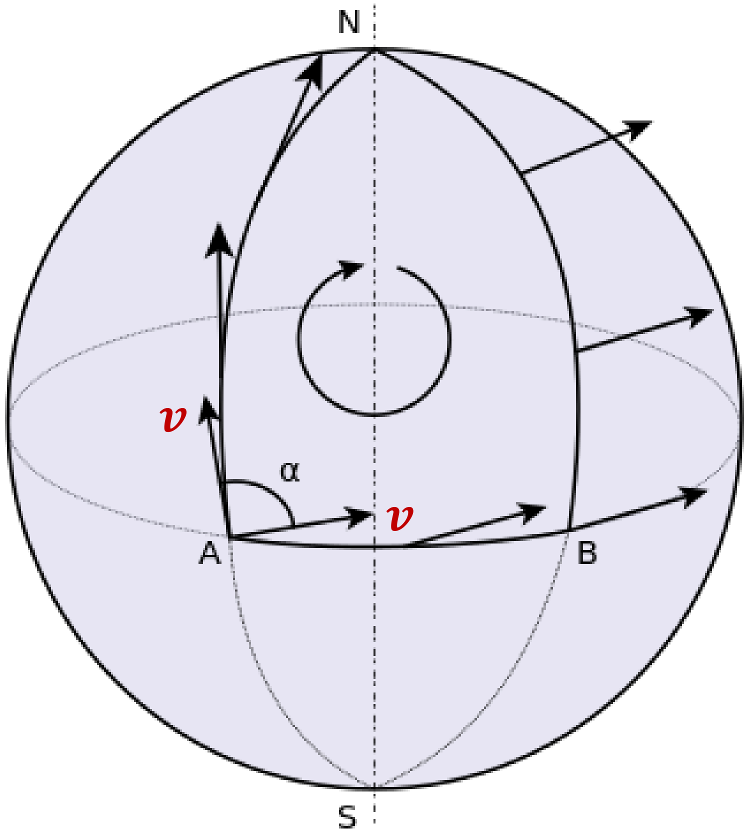
$$\log_{\boldsymbol{x}}^{K} \boldsymbol{y} = D_{\mathcal{L}}^{K}(\boldsymbol{x}, \boldsymbol{y}) \frac{\boldsymbol{y} + \frac{1}{K}\langle \boldsymbol{x}, \boldsymbol{y} \rangle_{\mathcal{L}} \boldsymbol{x}}{\left\| \boldsymbol{y} + \frac{1}{K}\langle \boldsymbol{x}, \boldsymbol{y} \rangle_{\mathcal{L}} \boldsymbol{x} \right\|_{\mathcal{L}}}$$

  - $\boldsymbol{y} \in \mathbb{H}^{\mathrm{d,K}}$
  - $D_{\mathcal{L}}^{K}(\boldsymbol{x}, \boldsymbol{y}) = \sqrt{K} \mathrm{arcosh}(-\frac{\langle \boldsymbol{x}, \boldsymbol{y} \rangle_{\mathcal{L}}}{K})$ is geodesic distance



Parallel Transport (1)

- **Parallel Transport:** transport a vector along a smooth curve on the surface and keep parallel to itself locally.



Transport a tangent vector $\boldsymbol{v}$ along the surface with non-zero curvature. When travelling from A to N to B back to A, the direction of the vector $\boldsymbol{v}$ changes!

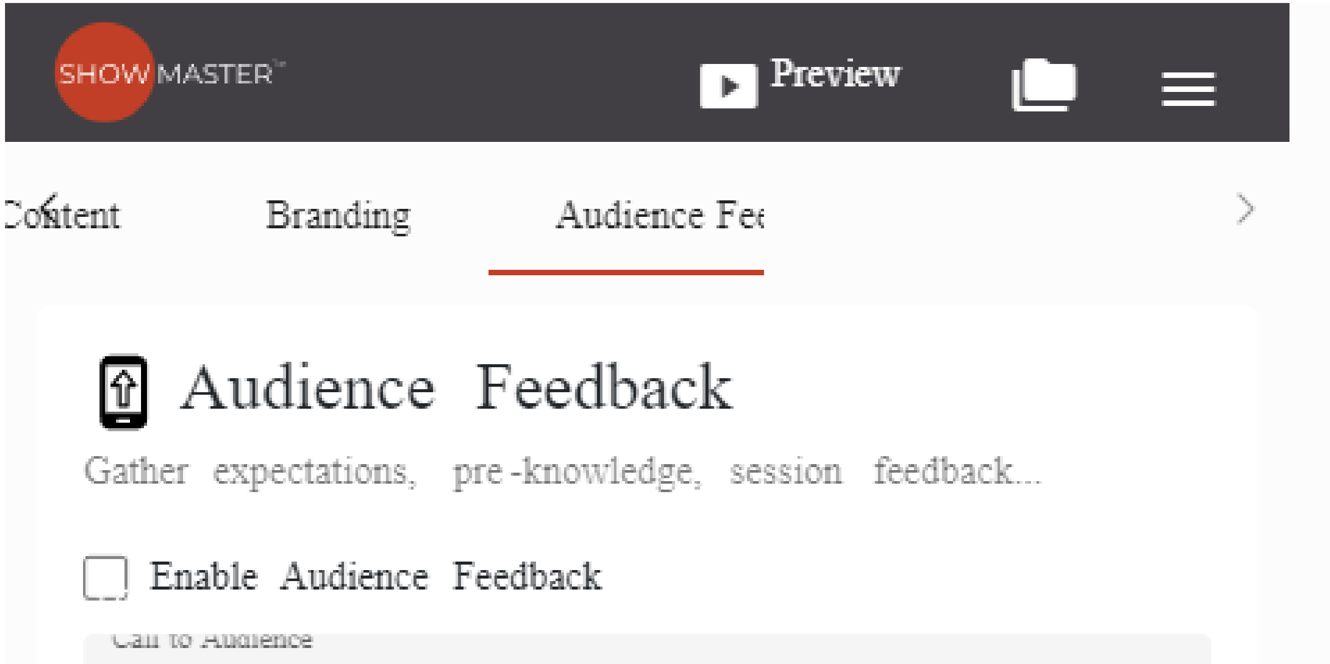
Parallel Transport (2)

- Parallel Transport $P_{\boldsymbol{x}\to\boldsymbol{y}}(\cdot)$ maps a vector $\boldsymbol{v} \in \mathcal{T}_{\boldsymbol{x}}\mathcal{M}$ to $P_{\boldsymbol{x}\to\boldsymbol{y}}(\boldsymbol{v}) \in \mathcal{T}_{\boldsymbol{y}}\mathcal{M}$
- If two points $\boldsymbol{x}$ and $\boldsymbol{y}$ on the hyperboloid $\mathbb{H}^{\mathrm{d,K}}$ are connected by a geodesic, then the parallel transport of tangent vector $\boldsymbol{v} \in \mathcal{T}_{\boldsymbol{x}}\mathbb{H}^{\mathrm{d,K}}$ to $\mathcal{T}_{\boldsymbol{y}}\mathbb{H}^{\mathrm{d,K}}$:

$$P_{\boldsymbol{x}\to\boldsymbol{y}}(\boldsymbol{v}) = \boldsymbol{v} - \frac{\langle \log_{\boldsymbol{x}}^{K}(\boldsymbol{y}), \boldsymbol{v}\rangle_{\mathcal{L}}}{D_{\mathcal{L}}^{K}(\boldsymbol{x},\boldsymbol{y})^2}\left(\log_{\boldsymbol{x}}^{\mathrm{K}} \boldsymbol{y} + \log_{\boldsymbol{y}}^{\mathrm{K}} \boldsymbol{x}\right)$$

  - $\log_{\boldsymbol{x}}^{\mathrm{K}}$ is the **Logarithmic map** at point x.
  - $D_{\mathcal{L}}^{K}(\boldsymbol{x},\boldsymbol{y}) = \sqrt{K}\operatorname{arcosh}\left(-\frac{\langle \boldsymbol{x},\boldsymbol{y}\rangle_{\mathcal{L}}}{K}\right)$ is geodesic distance

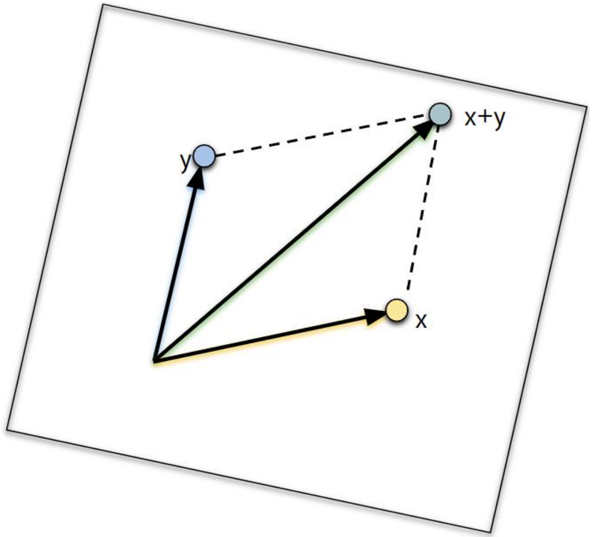
End of Part 1 (10 Minutes Break)



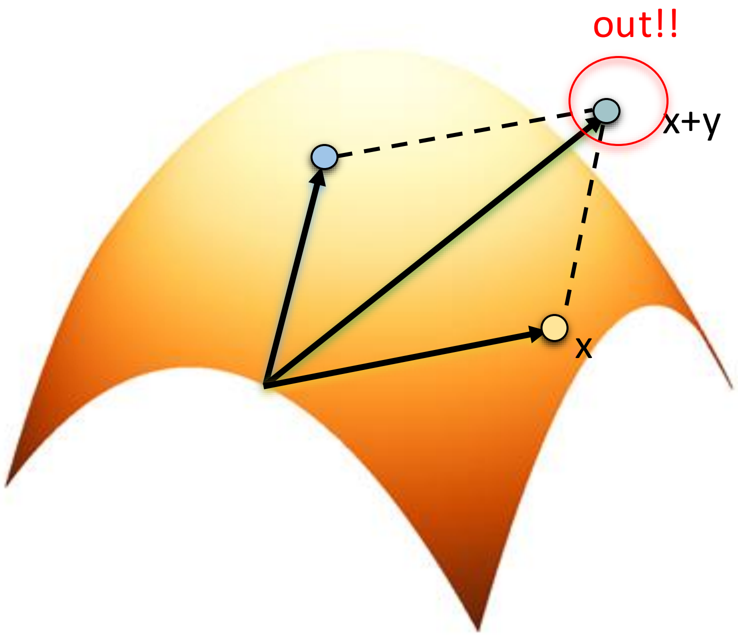
Part 2: Building Blocks for Hyperbolic Operations: Hyperbolic Neural Operations (50 Minutes)

Hyperbolic Operations: Difficulties

Addition in Euclidean Space



Addition in Hyperbolic Space?



Considerations:

1. Satisfy manifold constraints
2. Satisfy neural operation properties

Strategy 1: Tangent-Space Based Operations (1)

Recall: The tangent space is an Euclidean space

- Intuition: we know how to perform Euclidean operations!

General Recipe: Use a Euclidean function $f: \mathbb{R}^{d+1} \to \mathbb{R}^{d+1}$ on the tangent space

- e.g. Linear transformer: $f(x) = Wx + b$, non-linear activation: $f(x) = ReLU(x)$

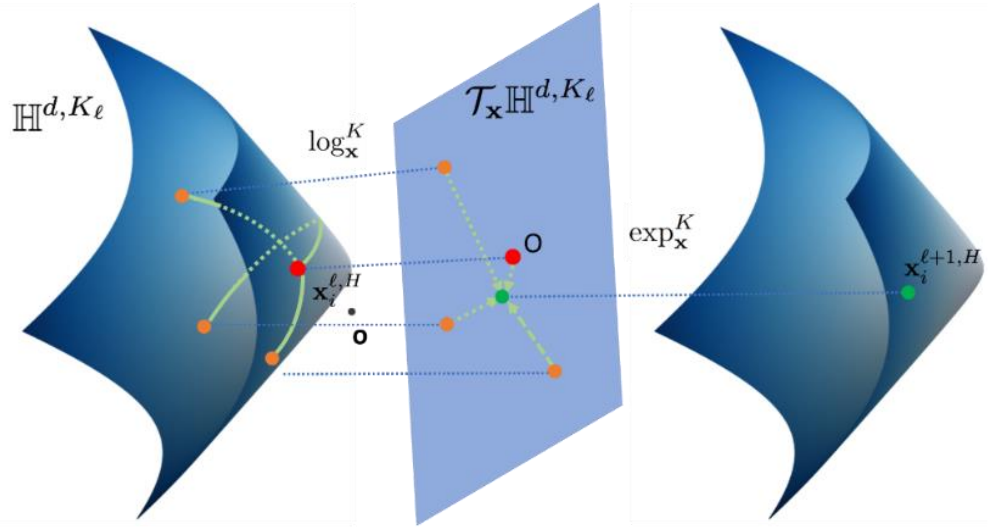


Image Source: Chami, Ines, et al. "Hyperbolic graph convolutional neural networks." Advances in neural information processing systems 32 (2019).

Strategy 1: Tangent-Space Based Operations (2)

Map input to tangent space of the origin, so f is a valid operation

↓

Perform Euclidean operation

↓

Lift the output back to $\mathbb{H}^{d,K}$

↓

$$f^{T,K}(x) = \exp_{\boldsymbol{o}}^{K}(f(\log_{\boldsymbol{o}}^{K}(x)))$$

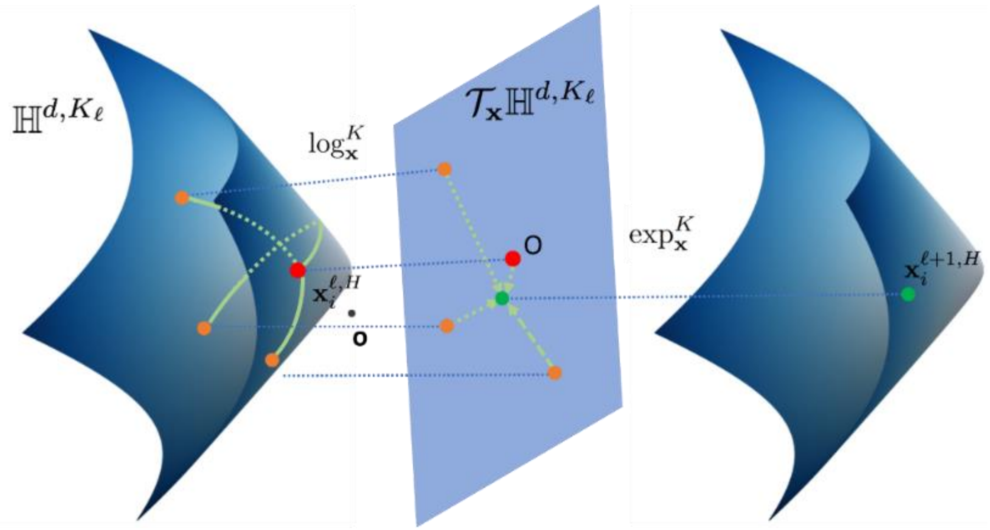


Image Source: Chami, Ines, et al. "Hyperbolic graph convolutional neural networks." Advances in neural information processing systems 32 (2019).

Strategy 1: Cons

Computational Inefficiency: the repeated mappings to and from the tangent space cause significant computational overhead

Numerical Instability: the mappings could cause numerical stability issues; e.g. in logarithmic map:

$$\log_{\boldsymbol{x}}^{K} \boldsymbol{y} = D_{\mathcal{L}}^{K}(\boldsymbol{x}, \boldsymbol{y}) \frac{\boldsymbol{y} + \frac{1}{K}\langle \boldsymbol{x}, \boldsymbol{y} \rangle_{\mathcal{L}} \boldsymbol{x}}{\left\| \boldsymbol{y} + \frac{1}{K}\langle \boldsymbol{x}, \boldsymbol{y} \rangle_{\mathcal{L}} \boldsymbol{x} \right\|_{\mathcal{L}}}$$

If the points are close together, we risk dividing by or calling $arccosin$ on 0.

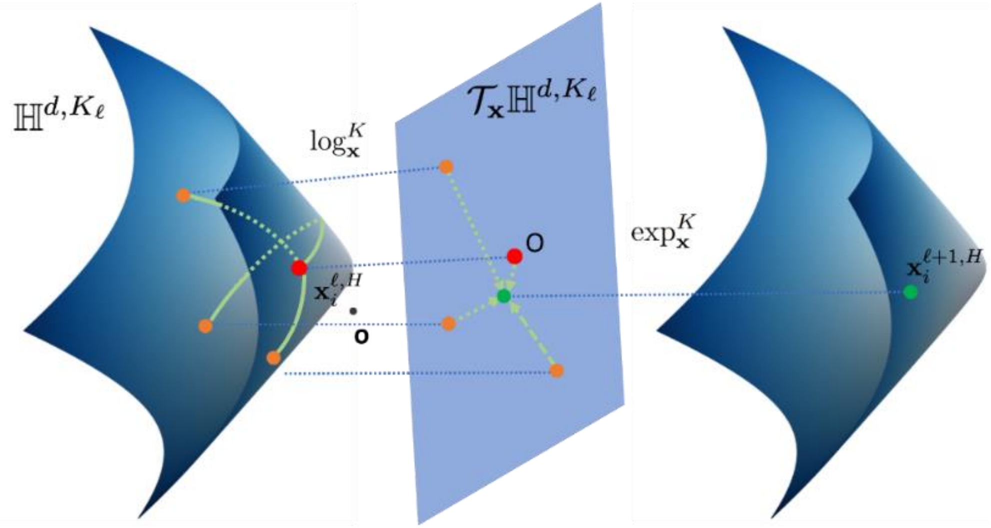


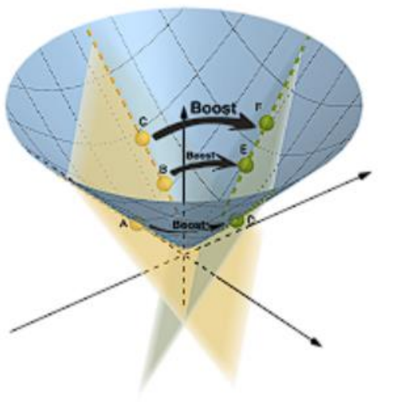
Image Source: Chami, Ines, et al. "Hyperbolic graph convolutional neural networks." Advances in neural information processing systems 32 (2019).

Strategy 1: Cons: Lorentz Rotation & Lorentz Boost

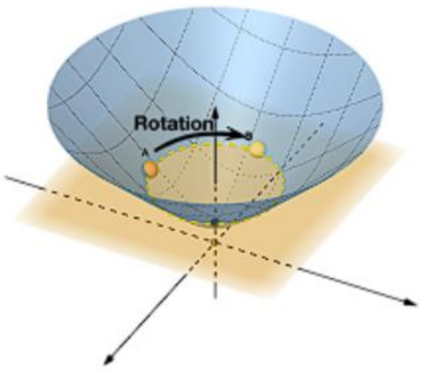
Expressiveness Issues: transformations implemented through $f^{T,K}$ might not cover all types of operations

- Lorentz linear transformation consists of a *Lorentz Boost* and a *Lorentz Rotation*, but tangent-space-based operations do not cover all cases

Constant velocity transformation without rotating the spatial axis



Lorentz Boost



Lorentz Rotation

Rotating the spatial axis by applying a rotation matrix on the space-like dimension

Image Source: Weize Chen, Xu Han, Yankai Lin, Hexu Zhao, Zhiyuan Liu, Peng Li, Maosong Sun, and Jie Zhou. 2021. Fully Hyperbolic Neural Networks. arXiv:2105.14686 (2021).

Strategy 2: Fully Hyperbolic Operations

Solution: operate directly on the manifold *"Fully Hyperbolic"*

Two strategies: *Pseudo Lorentz Rotation* v.s. *Pseudo Lorentz Boost*

Pseudo Lorentz Boost : Use a Euclidean function $f: \mathbb{R}^{d+1} \rightarrow \mathbb{R}^{d+1}$

- e.g. Linear transformer: $f(x) = Wx + b$

Perform f on $x \in \mathbb{H}^{d,K}$

Transformation on ***both*** time and space dimensions

Compute the associating time-like dimension

Impose Lorentzian constraints

$$f^{F,K}(x) = \left(\underbrace{\sqrt{||Wx_{time,space}||^2 - 1/K}}_{time-like\ dim}, \underbrace{Wx_{time,space}}_{space-like\ dim} \right)$$

Reference: Chami, Ines, et al. "Hyperbolic graph convolutional neural networks." Advances in neural information processing systems 32 (2019).
Menglin Yang, Harshit Verma, Delvin Ce Zhang, Jiahong Liu, Irwin King, and Rex Ying. 2024. Hypformer: Exploring efficient transformer fully in hyperbolic space. In KDD. 3770–3781.

Strategy 2: Fully Hyperbolic Operations Cont’d

Example: Tangent-space-based Linear Transformation $f^{T,K}$ is a Pseudo Lorentz Rotation!

- $f^{T,K}(x) = \exp_{\boldsymbol{o}}^{K}(f(\log_{\boldsymbol{o}}^{K}(x)))$
- $f(x) = Wx + b$

$$\begin{pmatrix} * & 0 \\ 0 & f(\cdot) \end{pmatrix} \log_{\boldsymbol{o}}^{K} \begin{pmatrix} x_{time} \\ x_{space} \end{pmatrix}$$

First coordinate of tangent vectors(of the origin) is ***0***, so the upper left entry does not affect the output

$$f^{T,K}(x) = \begin{pmatrix} \frac{\cosh(\beta)}{-Kx_{time}} & 0 \\ 0 & \frac{\sinh(\beta)W}{\sqrt{-K}||Wx_{space}||} \end{pmatrix} \begin{pmatrix} x_{time} \\ x_{space} \end{pmatrix} ; \beta = \frac{\sqrt{-K}\operatorname{arccosh}(\sqrt{-Kx_{time}})W}{\sqrt{-Kx^{2}_{time}}} ||Wx_{space}||$$

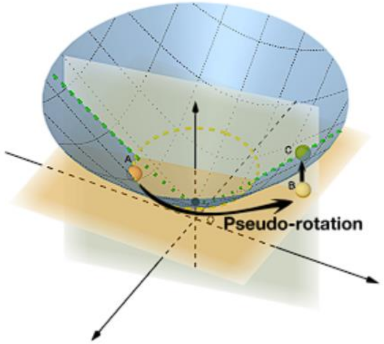


Image Source and Reference: Chami, Ines, et al. "Hyperbolic graph convolutional neural networks." Advances in neural information processing systems 32 (2019).

Strategy 2: Fully Hyperbolic Operations Cont’d

Solution: operate directly on the manifold *“Fully Hyperbolic”*

Two strategies: *Pseudo Lorentz Rotation* v.s. *Pseudo Lorentz Boost*

Pseudo Lorentz Rotation: Use a Euclidean function $f: \mathbb{R}^{d+1} \to \mathbb{R}^{d+1}$

- e.g. Linear transformer: $f(x) = ReLU(x)$

Perform f on the *space-like dimension* of $x \in \mathbb{H}^{d,K}$ — Transformation on ***only*** the space dimension

Compute the associating time-like dimension — Impose Lorentzian constraints

$$f^{F,K}(x) = \left(\underbrace{\sqrt{||f(x_{space})||^2 - 1/K}}_{time-like\ dim}, \underbrace{f(x_{space})}_{space-like\ dim} \right)$$

Reference: Chami, Ines, et al. "Hyperbolic graph convolutional neural networks." Advances in neural information processing systems 32 (2019).
Menglin Yang, Harshit Verma, Delvin Ce Zhang, Jiahong Liu, Irwin King, and Rex Ying. 2024. Hypformer: Exploring efficient transformer fully in hyperbolic space. In KDD. 3770–3781.

Strategy 2: Fully Hyperbolic Operations Cont'd

Pseudo Lorentz Rotation v.s. *Pseudo Lorentz Boost: Comparison*

Pseudo Lorentz Rotation: transformation on without time and space interaction

$$\begin{pmatrix} \frac{\sqrt{||f(x_{space})||^2 - 1/K}}{x_{time}} & 0 \\ 0 & f(\cdot) \end{pmatrix} \begin{pmatrix} x_{time} \\ x_{space} \end{pmatrix}$$

Off-diagonal values are zero

Pseudo Lorentz Boost: transformation on both time and space-like dimension

$$\begin{pmatrix} \sqrt{||Wx||^2 - 1/K)}\boldsymbol{e_0}, & W_{0,:} \\ \sqrt{||Wx||^2 - 1/K)}\boldsymbol{e_{1:d'}} & W_{1:,:} \end{pmatrix} \begin{pmatrix} x_{time} \\ x_{space} \end{pmatrix}$$

Non-zero off-diagonal terms

Refining Hyperbolic Operations

Intuition: take advantages of the *freedom in curvature – vary the curvature* through hyperbolic operations/layers

For *tangent-space-based* operations: $f^{T}_{K,K'}(x) = \sqrt{\frac{K}{K'}} f^{T,K}(x)$

For *fully hyperbolic* operations: $f^{F}_{K,K'}(x) = \sqrt{\frac{K}{K'}} f^{F,K}(x)$

Recalibrate coefficient for curvature changes:

$$\sqrt{\frac{K}{K'}} x = \exp_{\boldsymbol{o}}^{K'}\left(\log_{\boldsymbol{o}}^{K}(x)\right)$$

- Tangent space is the same across different curvature spaces!

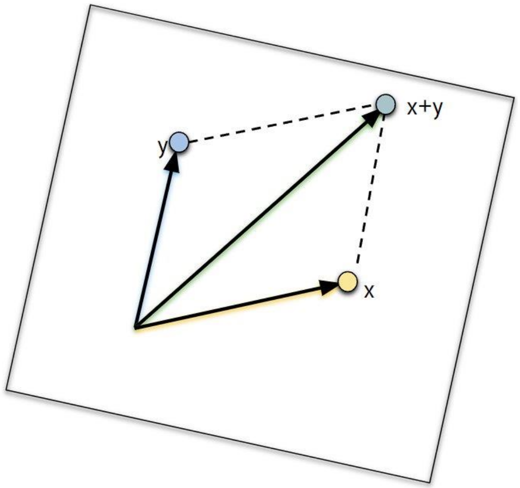
Hyperbolic Residual Connection & Addition

Recall: Addition is difficult in hyperbolic space!

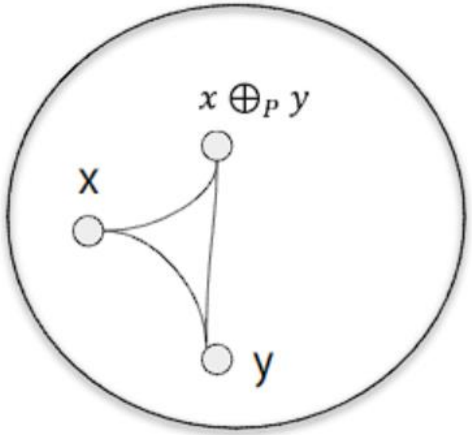
Tangent-space based method: *Möbius Addition* based on *parallel transport*:

$$x \oplus_P y = \exp_{\boldsymbol{x}}^K(P_{\boldsymbol{o}\to\boldsymbol{x}}(\log_{\boldsymbol{o}}^K(y)))$$

Vector Space formulation



Gyrovector Space formulation



Hyperbolic Residual Connection & Addition

Recall: Addition is difficult in hyperbolic space!

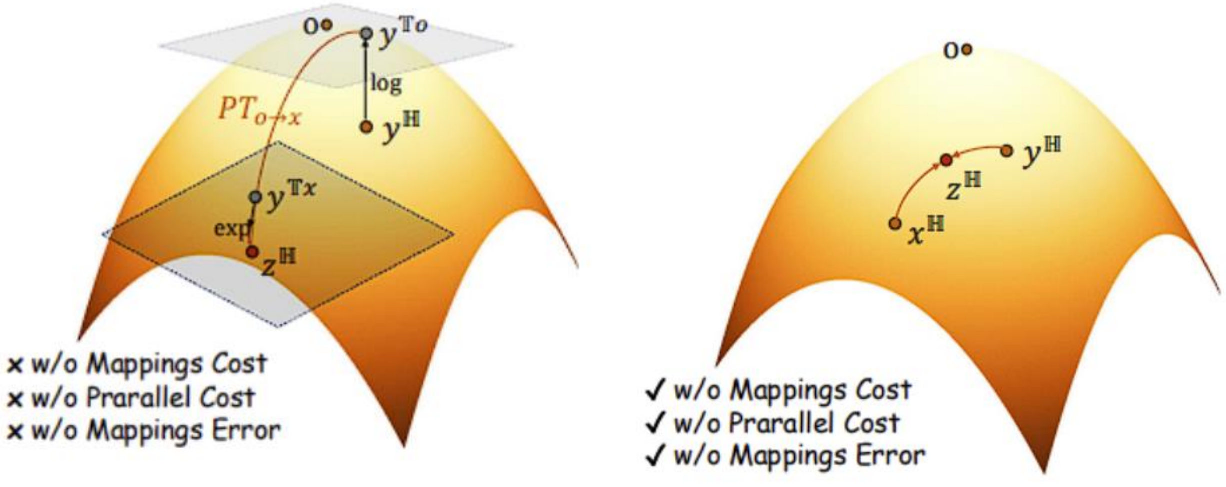
Fully hyperbolic method: generalized Lorent weighted sum

$$x \oplus_L y = \alpha \boldsymbol{x} + \beta \boldsymbol{y}$$

$$\alpha = \frac{w_x}{\sqrt{-K}\left\| |w_x \boldsymbol{x} + w_y \boldsymbol{y}| \right\|_{\mathcal{L}}}$$

$$\beta = \frac{w_y}{\sqrt{-K}\left\| |w_x \boldsymbol{x} + w_y \boldsymbol{y}| \right\|_{\mathcal{L}}}$$

$$w_x, w_y > 0$$



$x \oplus_P y$ $\quad$ $x \oplus_L y$

More *efficient*, *stable*, and *expressive*

Image Source: Neil He, Menglin Yang, and Rex Ying. 2025. Lorentzian Residual Neural Networks. In KDD.

Euclidean Self-Attention

Self-attention is a vital component in Euclidean Transformer-based foundation models, e.g.

- LLMs – text data
- ViTs – visual data
- CLIP models – multi-modal data

The key is to compute a ***weighted sum*** of value vector $\{V_j\}$ using weights based on similarity scores of keys $\{K_j\}$ and queries $\{Q_i\}$

$$Z_i = \sum_{j=1} \frac{\exp(Q_i K_j^T / \sqrt{d'})}{\sum_{j=1} \exp(Q_i K_j^T / \sqrt{d'})} V_j$$

How to generalize midpoint operations to hyperbolic space?

Hyperbolic Midpoint Operations

Hyperbolic midpoint has close forms in Lorentz model $LMid_K$, Poincare mode $PMid_K$, and Klein model $KMid_K$ (Einstein Midpoint)

- All of these operations are ***equivalent*** under ***isometric mappings***

Lorentzian Midpoint

$$LMid_K(x_1, \dots, x_N; \{v_i\}) = \frac{\sum_j v_j x_j}{\sqrt{-K} \left\| | \sum_j v_j x_j | \right\|_{\mathcal{L}}}$$

Poincaré Midpoint

$$PMid_K(x_1, \dots, x_N; \{v_i\}) = \frac{1}{2} \otimes_K \frac{\sum_j v_j \lambda_{x_i}^K x_j}{\sum_j |v_j| (\lambda_{x_i}^K - 1)}$$

$$\lambda_x^K = \frac{2}{1 + K||x||^2}$$

Gyrovector space scalar multiplication: implemented through $f^{T,K}$

Image Source: Neil He, Menglin Yang, and Rex Ying. 2025. Lorentzian Residual Neural Networks. In KDD.

Hyperbolic Midpoint Operations

Hyperbolic midpoint has close forms in Lorentz model $LMid_K$, Poincare mode $PMid_K$, and Klein model $KMid_K$ (Einstein Midpoint)

- All of these operations are ***equivalent*** under ***isometric mappings***

Lorentzian Midpoint

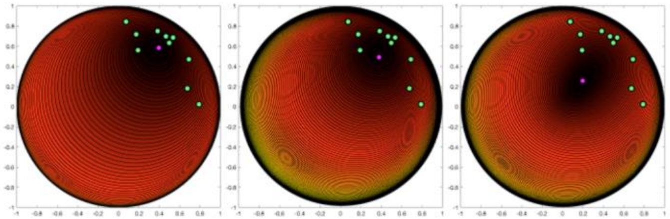
$$LMid_K(x_1, \dots, x_N; \{v_i\}) = \frac{\sum_j v_j x_j}{\sqrt{-K}\left\| |\sum_j v_j x_j| \right\|_{\mathcal{L}}}$$

Poincaré Midpoint

$$PMid_K(x_1, \dots, x_N; \{v_i\}) = \frac{1}{2} \otimes_K \frac{\sum_j v_j \lambda_{x_i}^K x_j}{\sum_j |v_j| (\lambda_{x_i}^K - 1)}$$

$$\lambda_x^K = \frac{2}{1 + K||x||^2}$$

Gyrovector space scalar multiplication: implemented through $f^{T,K}$



Marc Law, Renjie Liao, Jake Snell, and Richard Zemel. 2019. Lorentzian distance learning for hyperbolic representations. In ICML. PMLR, 3672–3681.

Hyperbolic Self-Attention

Hyperbolic self-attention can be formulated with hyperbolic midpoint operations and similarity score computed using negative hyperbolic distance

Hyperbolic Self-Attention

$$LAtten(Q, K, V) = LMid\left(v_1, \ldots, v_N, \{\alpha_{i,j}\}_{j=1}\right)$$

$$PAtten(Q, K, V) = PMid\left(v_1, \ldots, v_N, \{\alpha_{i,j}\}_{j=1}\right)$$

Attention Score

$$\alpha_{i,j} = \frac{\exp(-d_H^2(q_i, v_j))}{\sum_\ell \exp(-d_H^2(q_i, v_\ell))}$$

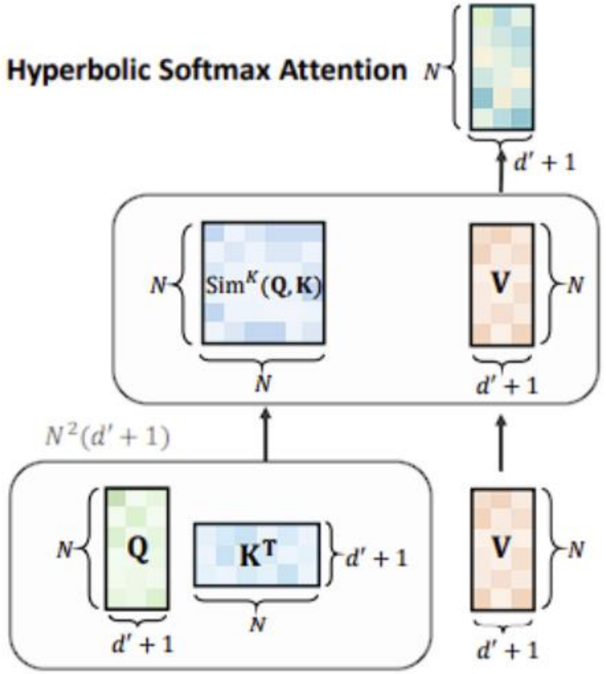
Image Source: Neil He, Menglin Yang, and Rex Ying. 2025. Lorentzian Residual Neural Networks. In KDD.

Hyperbolic Linear-Attention

Hyperbolic self-attention requires *quadratic time complexity* w.r.t. input tokens:

- Many applications such as graph Transformers requires the model to handle long token sequences

Solution: linear time approximation for attention mechanism



Menglin Yang, Harshit Verma, Delvin Ce Zhang, Jiahong Liu, Irwin King, and Rex Ying. 2024. Hypformer: Exploring efficient transformer fully in hyperbolic space. In KDD. 3770–3781.

Hyperbolic Linear-Attention Cont'd

Hyperbolic Linear Attention

$$LiAttn_{K_1,K_2}(Q,K,V) = \left[\sqrt{||Z||^2 - \frac{1}{K_2}}, Z\right]^T + f^F_{K_1,K_2}(V_s)$$

$$Z = \frac{Q'(K'^T V')}{Q'(K'^T \mathbf{1})}$$

Notations

$Q' = \phi(Q_s), K' = \phi(K_s), V' = \phi(V_s)$

$$\phi(x) = \frac{||\tilde{x}||}{||\tilde{x}^p||}\tilde{x}^p$$

$\tilde{x} = ReLU(x)/t$

t, p parameters

X_s denotes the space-like dimension

Image Source: Neil He, Menglin Yang, and Rex Ying. 2025. Lorentzian Residual Neural Networks. In KDD.

Hyperbolic Linear-Attention Cont'd

Hyperbolic Linear Attention

$$Q' = \phi(Q_s), K' = \phi(K_s), V' = \phi(V_s)$$

$$LiAttn_{K_1,K_2}(Q,K,V) = \left[\sqrt{||Z||^2 - \frac{1}{K_2}}, Z\right]^T + f^F_{K_1,K_2}(V_s)$$

$$Z = \frac{Q'(K'^T V')}{Q'(K'^T \mathbf{1})}$$

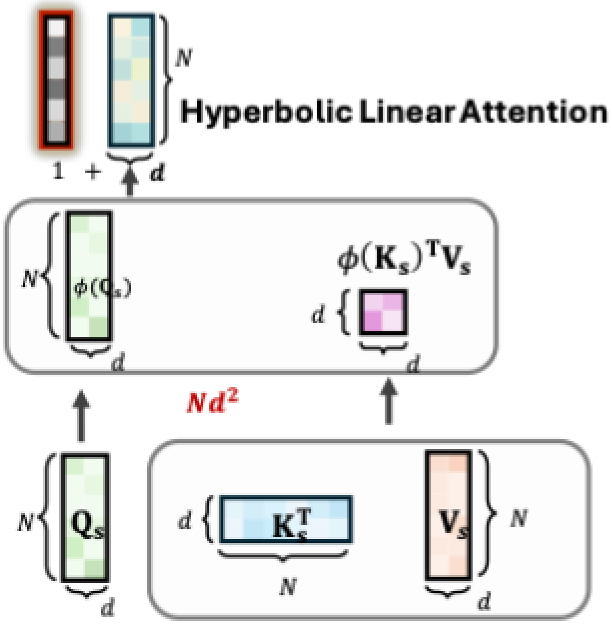


Image Source: Neil He, Menglin Yang, and Rex Ying. 2025. Lorentzian Residual Neural Networks. In KDD.

Hyperbolic Normalization Methods

Normalization methods are critical for neural network and foundation models, e.g.

- Layer normalization in Transformers
- Batch normalization in Convolutional Neural Networks

Considerations:

- *Meaningful normalizing operations*
- *Computational efficiency*

Hyperbolic Normalization Methods Cont'd

Consideration 1: Meaningful normalization – similar to the Euclidean case, the goal is to *center the feature vectors across batches/layers* and scale the *keep the variance of their norms within a manageable range*

- Initial work proposed using the ***Fréchet Mean***
- However, this is ***computational expensive***
  - Up to 77% of all compute in the forward pass in hyperbolic CNNs!

Consideration 2: Finding computationally efficient methods while maintaining consideration 1

References: Max van Spengler, Erwin Berkhout, and Pascal Mettes. 2023. Poincaré ResNet. CVPR (2023)

Hyperbolic Batch Normalization

Method 1: use *hyperbolic midpoint operations* instead of Fréchet mean

- Approximately centering the vectors at the origin

Compute mean $\mu = PMid_K(x_1, \dots, x_N, \{1\})$ (or $\mu = LMid_K(x_1, \dots, x_N, \{1\})$)

↓

Compute variance $\sigma^2 = \frac{1}{N}\sum_i d_H^2(x_i, \mu)$

↓

Return normalization term $\tilde{x}_i = \exp_\beta^K(\frac{\sqrt{\gamma}}{\sigma} P_{\mu\to\beta}(\log_\mu^K(x_i)))$

Learnable parameters

Set new mean as learnable β

Optional: re-centering at the origin first: simple geodesics at the origin

$$P_{o\to\beta}(\frac{\sqrt{\gamma}}{\sigma} P_{\mu\to o}(\log_\mu^K(x_i)))$$

References: Max van Spengler, Erwin Berkhout, and Pascal Mettes. 2023. Poincaré ResNet. CVPR (2023)
Ahmad Bdeir, Kristian Schwethelm, and Niels Landwehr. 2024. Fully Hyperbolic Convolutional Neural Networks for Computer Vision. In ICLR.

Hyperbolic Layer Normalization

Method 2: use *fully hyperbolic* formulation in *Lorentz space*

- Computationally efficient
- Retain normalizing capabilities

Normalizing the space-like dimension: $y_s = LayerNorm(x_s)$(or $y_s = RSMNorm(x_s)$, etc)

Compute the time-like dimension and return normalized vectors:

$$\left[\sqrt{||y_s||^2 - \frac{1}{K}}, y_s\right]^T$$

Normalizing space dimension approximates normalization locally and centers around the origin: $o = \left[\sqrt{-\frac{1}{K}}, 0, \dots, 0\right]$

References: Menglin Yang, Harshit Verma, Delvin Ce Zhang, Jiahong Liu, Irwin King, and Rex Ying. 2024. Hypformer: Exploring efficient transformer fully in hyperbolic space. In KDD. 3770–3781
Neil He, Rishabh Anand, Hiren Madhu, Ali Maatouk, Smita Krishnaswamy, Leandros Tassiulas, Menglin Yang, and Rex Ying. 2025. HELM: Hyperbolic Large Language Models via Mixture-of-Curvature Experts. arXiv preprint arXiv:2505.24722 (2025).

Hyperbolic Positional Encoding

Positional encodings (PE) enables the model to *learn ordering information of tokens* in the input sequence

Learn ***relative positional information***:

- Though hyperbolic addition: $PE_K(x) = x \oplus_L \epsilon f^{F,K}(x)$; ϵ learnable parameters
- Adding positional encoding as bias term in $f^{F,K}$

Assumes PE also follows a linear layer

Pros of relative positional encoding:

- Improves generalizability to different sequence length
- Improves context understanding

Cons of relative positional encoding:

- Introduces additional parameters and computational/memory costs
- Potential overfitting & requires further tuning

References: Menglin Yang, Harshit Verma, Delvin Ce Zhang, Jiahong Liu, Irwin King, and Rex Ying. 2024. Hypformer: Exploring efficient transformer fully in hyperbolic space. In KDD. 3770–3781
Neil He, Rishabh Anand, Hiren Madhu, Ali Maatouk, Smita Krishnaswamy, Leandros Tassiulas, Menglin Yang, and Rex Ying. 2025. HELM: Hyperbolic Large Language Models via Mixture-of-Curvature Experts. arXiv preprint arXiv:2505.24722 (2025).
Weize Chen, Xu Han, Yankai Lin, Hexu Zhao, Zhiyuan Liu, Peng Li, Maosong Sun, and Jie Zhou. 2021. Fully Hyperbolic Neural Networks. arXiv:2105.14686 (2021).

Hyperbolic Rotary Positional Encoding

Alternative: Rotary incorporates aspects from both *absolute and relative* encoding method

- Euclidean RoPE: apply ***rotational matrix*** to feature vectors

Apply ***Lorentzian rotation*** to hyperbolic vectors:

$$HoPE(z_i) = \left[\sqrt{R_{i,\Theta}(z_i)_s - \frac{1}{K}}, R_{i,\Theta}(z_i)_s \right]^T$$

$$\Theta = \{\theta_1, \dots, \theta_{\frac{d}{2}}\}$$

$R_{i,\Theta} \in \mathbb{R}^{d \times d}$ where the *diagonal are 2×2 block matrices*

R_{i,θ_j}, which are 2×2 rotation matrices of angle $i\theta_j$

z_i can either be query q_i or key k_i

- ***Long-term decay***: the attention score between a *key-query pair decays* when the *relative position increases*

- ***Robustness***: *robust* attention across *arbitrary relative distances*

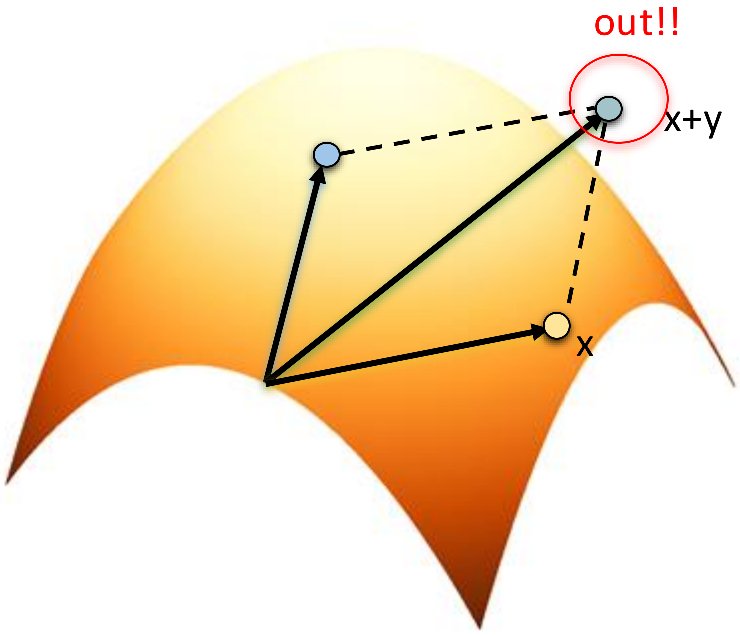
- ***Learning Complex Relations***: attention heads with HoPE can learn *diagonal (attends to only itself)* and *off-diagonal (attends to only predecessor)* attention patterns

Neil He, Rishabh Anand, Hiren Madhu, Ali Maatouk, Smita Krishnaswamy, Leandros Tassiulas, Menglin Yang, and Rex Ying. 2025. HELM: Hyperbolic Large Language Models via Mixture-of-Curvature Experts. arXiv preprint arXiv:2505.24722 (2025).

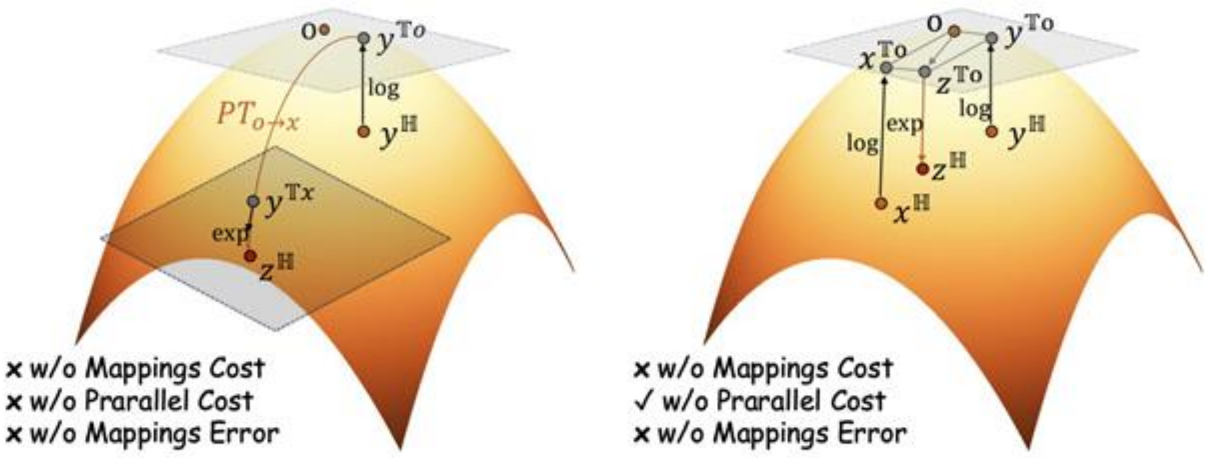
Part 3: Hyperbolic Foundation Models (70 Minutes)

Our Approaches: Hyperbolic Residual Connection

- Euclidean residual connection relies on *vector addition*
  - This is not a valid hyperbolic operation!



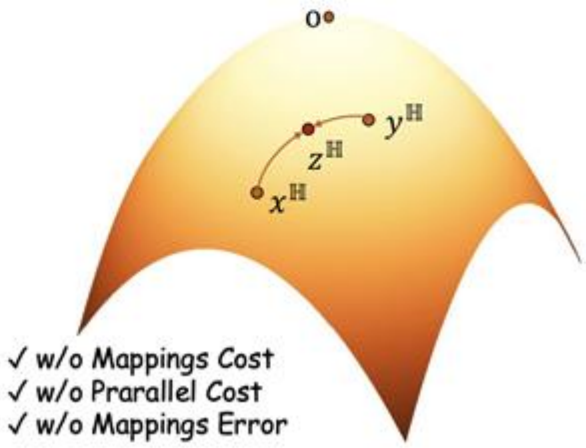
- Previous methods: Parallel Transport, Tangent Space, Space Addition (not shown)
  - ***Problems:*** Numerical Instability, Mapping Errors, Non-commutative, Computational Inefficiency, Lack of Geometric Meaning



(a) Parallel Transport Based Method (b) Tangent Space Based Method

Our Approach: LResNet

- We propose a general addition method that computes a (normalized) weighted sum of x and f(x)



(c) LResNet (Ours)

$$\mathbf{x} \oplus_{\mathcal{L}} f(\mathbf{x}) := \alpha_{w_x,w_y}\mathbf{x} + \beta_{w_x,w_y} f(\mathbf{x})$$

Normalizing Weight

$$\alpha_{w_x,w_y} = w_x / \sqrt{-K} | \|w_x \mathbf{x} + w_y f(\mathbf{x})\|_{\mathcal{L}} |$$

$$\beta_{w_x,w_y} = w_y / \sqrt{-K} | \|w_x \mathbf{x} + w_y f(\mathbf{x})\|_{\mathcal{L}} |$$

Ensures the sum is valid in hyperbolic space

$$w_x, w_y \in \{\mathbb{R}^2 - (0,0)\}$$

- Provably numerical stable
  - The denominator is ***never*** zero!
- No mapping errors
- Time efficiency
  - Over ***2000X speedup*** on random vectors with dimension 4092 and size 100,000
- Previous methods are special cases
  - In the geodesic sense

Our Approaches: Hyperbolic Transformer - HypFormer

Challenge 1: Problematic Transformation

- Tangent space Based Transformation: high computation cost, mapping errors
- Fully Lorentz Transformation: cannot preserve relative distance, complex computations

🔥 **Proposed Method: Direct Lorentz Transformation with Distance Preservation**

Challenge 2: Incomplete Modules

- Not all necessary basic modules for Transformer are well-defined, e.g. layer normalization and positional encoding

🔥 **Proposed Method: Definition all necessary basic modules for hyperbolic Transformer**

Challenge 3: Quadratic Time Complexity

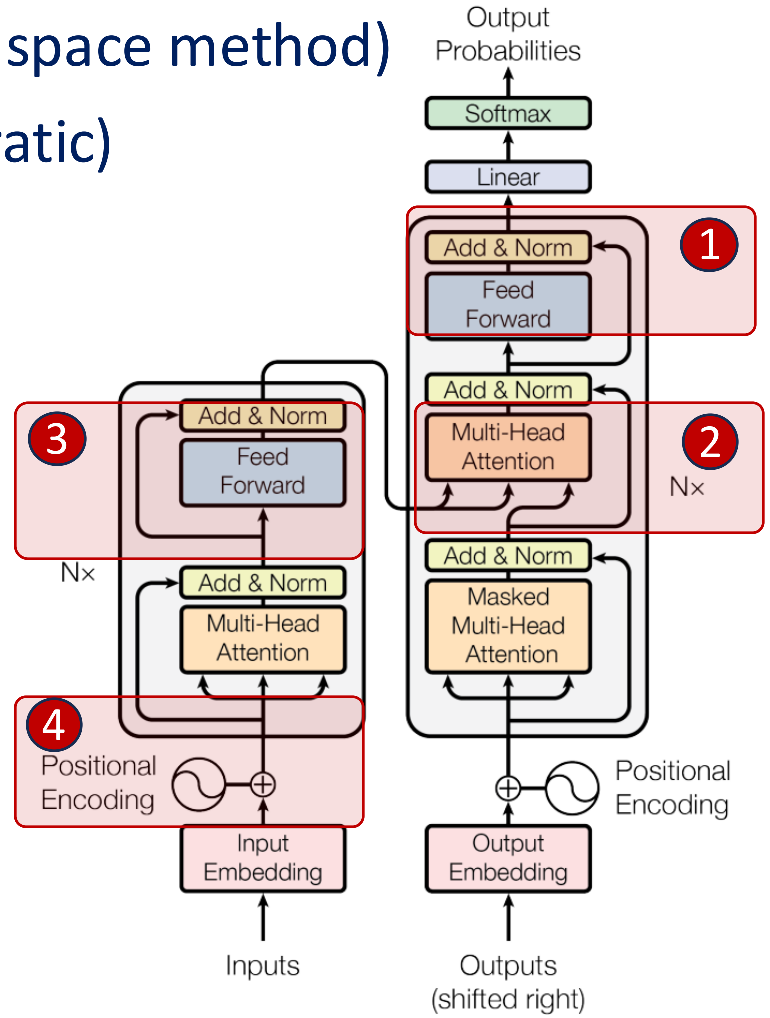
- Cannot process long-sequence input tokens and large-scale graphs

🔥 **Proposed Method: Linear time complexity, w.r.t number of token and nodes**

Challenges: Faithful Hyperbolic Modules

- ✅ FeedForward Layer (tangent space method)
- ✅ Multi-Head Attention (quadratic)

- 🚫 Positional Encoding
- 🚫 Add & LayerNorm
- 🚫 Multihead concatation
- 🚫 Dropout & ReLU operations



Core modules in Transformer

1. FeedForward Layer
2. Multi-Head Attention
3. Addition and LayerNorm
4. Positional Encoding

Solution to Challenge 1 & 2: Two Basic Transformation Block

- **CAR: Curvature Adaptative Rotation Transformation**, defining ReLU, LayerNorm, BatchNorm, Concatnation
- **CAB: Curvature Adaptative Boost Transformation**, defining Linear Transformation
- Guarantee that the results are valid hyperbolic embeddings

$$y^H = \left(\underbrace{\sqrt{\frac{K'}{K} ||f(x_{space})||^2 - K'}}_{time-like\ dim}, \underbrace{\sqrt{\frac{K'}{K}} f(x_{space})}_{space-like\ dim} \right)$$

CAR

$$y^H = \left(\underbrace{\sqrt{\frac{K'}{K} ||W x_{time,space}||^2 - K'}}_{time-like\ dim}, \underbrace{\sqrt{\frac{K'}{K}} W x_{time,space}}_{space-like\ dim} \right)$$

CAB

(1) Computationally efficient

(2) Adaptive curvature; and preserves the relative distance after altering the curvatures

(3) Comprehensive set of operations needed in Transformers

Solution to Challenge 1 & 2: Two Basic Transformation Block

- **CAR: Curvature Adaptative Rotation Transformation**, for ReLU, LayerNorm, BatchNorm, Concatnation
- **CAB: Curvature Adaptative Boost Transformation**, for Linear Transformation
- **Guarantee that the results are valid hyperbolic embeddings**

$$y^H = \left(\underbrace{\sqrt{\frac{K'}{K}||f(x_{space})||^2 - K'}}_{time-like\ dim}, \underbrace{\sqrt{\frac{K'}{K}} f(x_{space})}_{space-like\ dim} \right)$$

CAR

$$y^H = \left(\underbrace{\sqrt{\frac{K'}{K}||Wx_{time,space}||^2 - K'}}_{time-like\ dim}, \underbrace{\sqrt{\frac{K'}{K}} Wx_{time,space}}_{space-like\ dim} \right)$$

CAB

$$\begin{pmatrix} \frac{\sqrt{\frac{K'}{K}||f(z_{space})||^2 - K'}}{z_{time}} & 0 \\ 0 & f(\cdot) \end{pmatrix} \begin{pmatrix} z_{time} \\ z_{space} \end{pmatrix}$$

Off-diagonal values are zero

Pseudo Lorentz Rotation: transformation on without time and space interaction

$$\begin{pmatrix} \sqrt{\frac{K'}{K}||Wx||^2 - K')\boldsymbol{e_0}}, & \sqrt{\frac{K'}{K}} W_{0,:} \\ \sqrt{\frac{K'}{K}||Wx||^2 - K')\boldsymbol{e_{1:d'}}} & \sqrt{\frac{K'}{K}} W_{1:,:} \end{pmatrix} \begin{pmatrix} 1 \\ x \end{pmatrix}$$

Pseudo Lorentz Boost: transformation on both time and space-like dimension

Solution to Challenge 3: Hyperbolic Linear Attention

The linear attention mechanism is designed through the following steps: (1) linear transformation via **CAB** (denoted as HTC), **(2) computation of the linear attention score in the space-like dimension of the hyperboloid model**, and (3) recalibration.

$$\mathcal{Q} = \text{HTC}(\mathbf{X}; f_t, \mathbf{W}^Q, \kappa_1, \kappa_2),$$

$$\mathcal{K} = \text{HTC}(\mathbf{X}; f_t, \mathbf{W}^K, \kappa_1, \kappa_2),$$

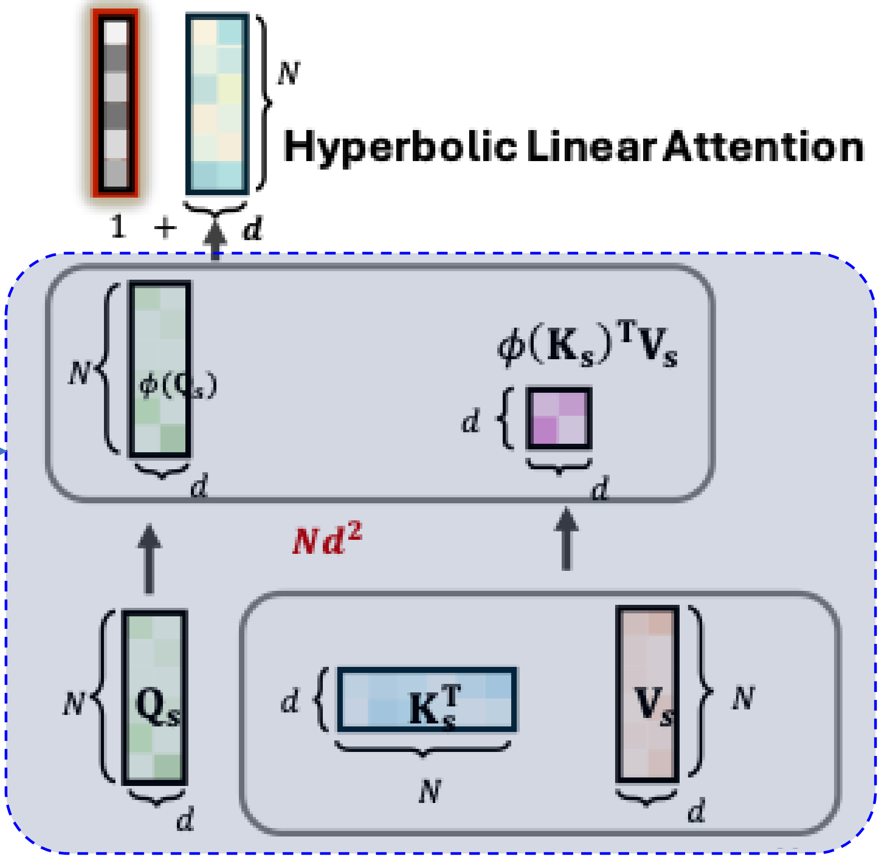
$$\mathcal{V} = \text{HTC}(\mathbf{X}; f_t, \mathbf{W}^V, \kappa_1, \kappa_2),$$

$$\mathcal{Q}_s, \mathcal{K}_s, \mathcal{V}_s = \phi(\mathcal{Q}_{[1:]}), \phi(\mathcal{K}_{[1:]}), \phi(\mathcal{V}_{[1:]}).$$

$$\mathcal{Z}_s = \frac{\mathcal{Q}_s(\mathcal{K}_s^T \mathcal{V}_s)}{\mathcal{Q}_s(\mathcal{K}_s^T \mathbf{1})}.$$

$$\mathcal{Z}_t = \sqrt{\frac{\kappa_2}{\kappa_3}\|\mathcal{Z}_s\|^2 - 1/\kappa_3},$$

$$\mathcal{Z} = \left(\mathcal{Z}_t, \sqrt{\frac{\kappa_2}{\kappa_3}}\mathcal{Z}_s\right).$$



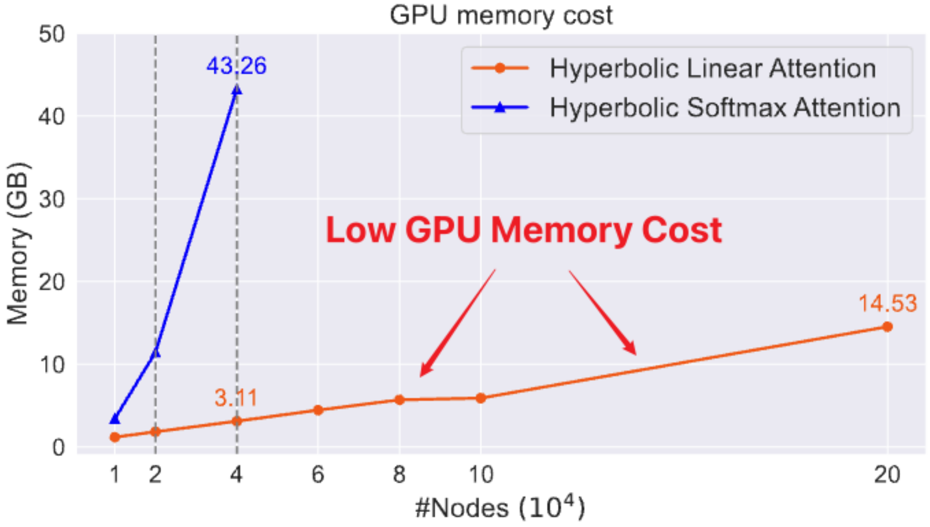
Experiment Snapshot: Scalability Evaluation

Method	ogbn-proteins	Amazon2m	ogbn-arxiv	Papers100M
#Nodes	132, 534	2, 449, 029	169, 343	111, 059, 956
#Edges	39, 561, 252	61, 859, 140	1, 166, 243	1, 615, 685, 872
MLP	72.0 ± 0.5	63.5 ± 0.1	55.5 ± 0.2	47.2 ± 0.3
GCN [33]	72.5 ± 0.4	83.9 ± 0.1	71.7 ± 0.3	OOM
SGC [70]	70.3 ± 0.2	81.2 ± 0.1	67.8 ± 0.3	63.3 ± 0.2
GCN-NSampler	73.5 ± 1.3	83.8 ± 0.4	68.5 ± 0.2	62.0 ± 0.3
GAT-NSampler	74.6 ± 1.2	85.2 ± 0.3	67.6 ± 0.2	63.5 ± 0.4
SIGN [21]	71.2 ± 0.5	81.0 ± 0.3	70.3 ± 0.3	65.1 ± 0.1
GraphFormer [83]	OOM	OOM	OOM	OOM
GraphTrans [73]	OOM	OOM	OOM	OOM
GraphGPS [54]	OOM	OOM	OOM	OOM
HAN [25]	OOM	OOM	OOM	OOM
HNN++ [60]	OOM	OOM	OOM	OOM
F-HNN [9]	OOM	OOM	OOM	OOM
NodeFormer [71]	77.5 ± 1.2	87.9 ± 0.2	59.9 ± 0.4	OOT
SGFormer [72]	79.5 ± 0.3	89.1 ± 0.1	72.4 ± 0.3	65.8 ± 0.5
Hypformer	**80.4 ± 0.5**	**89.4 ± 0.3**	**73.2 ± 0.2**	**66.1 ± 0.4**

GraphFormer Model

Hyperbolic (Graph)Transformer (*failed!!*)

Successfully working on billion-level graph data and process 10K~200K input tokens



More efficiency and save half of running time

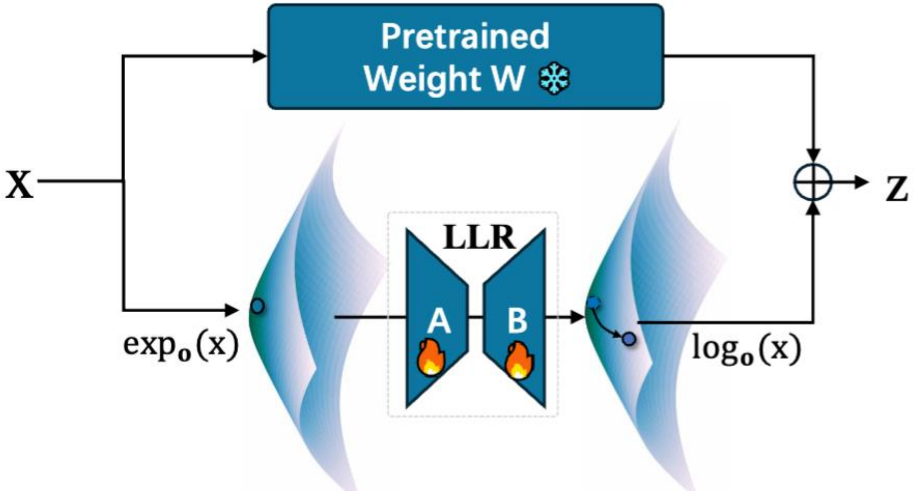
Method	ogbn-proteins Train	ogbn-proteins Test	Amazon2M Train	Amazon2M Test	ogbn-arxiv Train	ogbn-arxiv Test
Hypformer (Softmax)	11.9	-	37.38	-	7.8	-
Hypformer (Linear)	5.3	2.4	16.32	2.5	3	2.5

LLM Integration: Hyperbolic Fine-Tuning (HypLoRA)

Building on existing Euclidean LLMs:

- Maintains flexibility while producing hyperbolic representations
- Leverages pre-trained knowledge

Our proposed method



$$\mathbf{z}^E = W_{\text{LoRA}}(\mathbf{x}^E) = W\mathbf{x}^E + \Delta W\mathbf{x}^E$$

$$= W\mathbf{x}^E + \log_{\mathbf{o}}^K\Big(\underbrace{\mathbf{LLR}(BA, \overbrace{\exp_{\mathbf{o}}^K(\mathbf{x}^E)}^{\mathbf{x}^H})}_{\text{Transformation on } \mathbf{x}^H}\Big),$$

LLR(BA, X) is based on our CAB Transformation

$$\mathbf{LLR}(BA, \mathbf{x}^H) = \left(\sqrt{\|B\mathbf{y}^H\|_2^2 + K}, B\mathbf{y}^H\right), \quad (4)$$

$$\text{where } \mathbf{y}^H = \left(\sqrt{\|A\mathbf{x}^H\|_2^2 + K}, A\mathbf{x}^H\right), \quad (5)$$

Experiment Snapshot: Mathematical Reasoning

MAWPS: Paul had 95 pens and 153 books. After selling some books and pens in a garage sale he had 13 books and 23 pens left. How many books did he sell in the garage sale?

Dataset	Domain	# Train	# Test	Answer
MAWPS	Math	-	239	Number
GSM8K	Math	8.8K	1,319	Number
AQuA	Math	100K	254	Option
SVAMP	Math	-	1,000	Number

GSM8K: James decides to run 3 sprints 3 times a week. He runs 60 meters each sprint. How many total meters does he run a week?

AQuA: Find out which of the following values is the multiple of X, if it is divisible by 9 and 12?
"options": ["A)36", "B)12", "C)3", "D)9", "E)6"]

Experiment Snapshot: Mathematical Reasoning

Model	PEFT Method	MAWPS(8.5%)	SVAMP(35.6%)	GSM8K(46.9%)	AQuA(9.0%)	M.AVG
GPT-3.5	None	**87.4**	**69.9**	**56.4**	**38.9**	**62.3**
LLaMA-7B	None	51.7	32.4	15.7	16.9	24.8
	Prefix*	63.4	38.1	24.4	14.2	31.7
	Series*	77.7	52.3	33.3	15.0	42.2
	Parallel*	82.4	49.6	35.3	18.1	42.8
	LoRA*	79.0	52.1	37.5	18.9	44.6
	LoRA†	81.9	48.2	38.3	18.5	43.7
	DoRA	80.0	48.8	39.0	16.4	43.9
	HypLoRA (Ours)	79.0	49.1	39.1	20.5	44.4
LLaMA-13B	None	65.5	37.5	32.4	15.0	35.5
	Prefix*	66.8	41.4	31.1	15.7	36.4
	Series*	78.6	50.8	44.0	22.0	47.4
	Parallel*	81.1	55.7	43.3	20.5	48.9
	LoRA*	83.6	54.6	47.5	18.5	50.5
	LoRA†	83.5	54.7	48.5	18.5	51.0
	DoRA	83.0	54.6	OOT	18.9	NA
	HypLoRA (Ours)	83.2	54.8	49.0	21.5	51.5
Gemma-7B	None	76.5	60.4	38.4	25.2	48.3
	LoRA	91.6	76.2	66.3	28.9	68.6
	DoRA	91.7	75.9	65.4	27.7	68.0
	HypLoRA (Ours)	91.5	78.7	69.5	32.7	71.3
LLaMA3-8B	None	79.8	50.0	54.7	21.0	52.1
	LoRA	**92.7**	78.9	70.8	30.4	71.9
	DoRA	92.4	79.3	71.3	33.1	72.5
	HypLoRA (Ours)	91.6	**80.5**	**74.0**	**34.2**	**74.2**

+11% (LLaMA-7B)

+16% (LLaMA-13B)

+13% (Gemma-7B)

+13.4% (LLaMA3-8B)

Improvements over LoRA

HypLoRA performs better on harder questions.

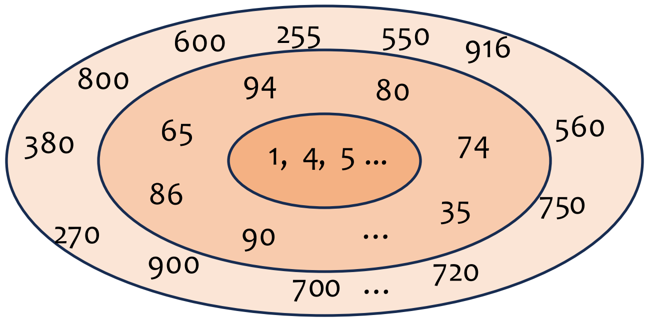
HypLoRA introduce higher-order interaction and hierarchies-related terms compared with LoRA.

The update of query Q is related to high-order Information and token's norm

$$\Delta Q^{\text{Hyp}} \approx (BA)\mathbf{x} + \frac{\|\mathbf{x}\|^2}{6R^2}(BA)\mathbf{x}.$$

$$\Delta Q^{\text{LoRA}} = (BA)\mathbf{x}.$$

Case Study

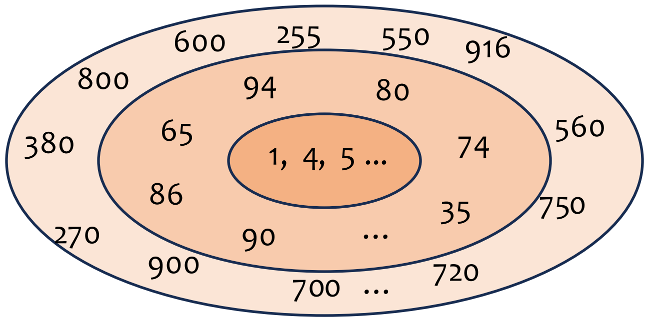


Numbers (token) are arranaged based on their norms in LLaMA 3

Question	To have the car for the weekend, Wilson's report card needs to show that he received 80 or higher in his math class. His math grade is calculated by the average of 5 test scores. On those tests, he received: 65, 94, 81, 86, and 74. What is his current math grade? **Answer:** 80.0
LoRA	To find Wilson's current math grade, we need to take the average of his 5 test scores. First, we add up all 5 test scores: 65 + 94 + 81 + 86 + 74 = 380. Then, we divide the sum by the number of tests: $380 \div 5 = 76$. Therefore, Wilson's current math grade is 76. **Answer: 76**.
HypLoRA	To find Wilson's current math grade, we need to take the average of his 5 test scores. To do this, we add up the scores and divide by 5: 65 + 94 + 81 + 86 + 74 = 400. Then, we divide by 5: $400 \div 5 = 80$. Therefore, Wilson's current math grade is 80. **Answer: 80**.

HypLoRA provides better understanding of number's hierarchies (especially for these leaf tokens) for prediction and accurate computation

Case Study

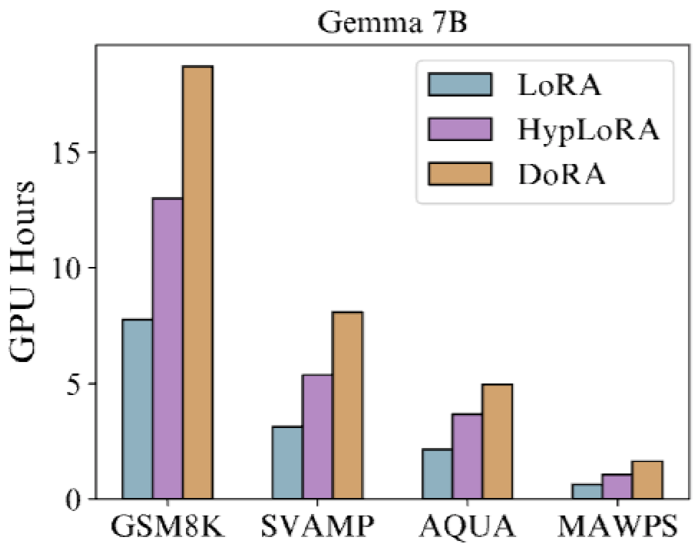


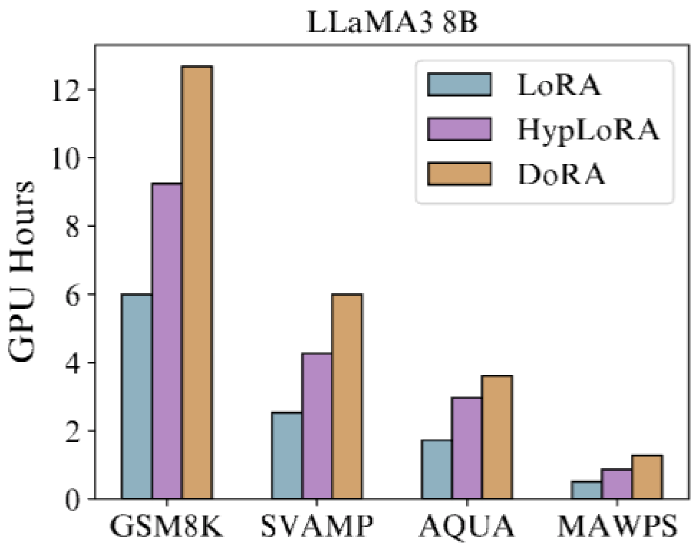
Numbers (token) are arranaged based on their norms in LLaMA 3

Question	Gissela, Gordy, and Gary are truck drivers. Gissela has a truck large enough to haul 4,000 pounds of gravel. Gordy's truck can haul 800 pounds more than Gissela's truck. When Gary brings his truck and joins Gissela and Gordy, the three trucks combined can haul a total of 11,600 pounds of gravel. How many pounds of gravel can Gary's truck carry? **Answer: 2,800 pounds**
LoRA	Let's start by assigning variables to represent the amount of gravel each truck can haul: Gissela's truck hauls 4,000 pounds. Gordy's truck hauls $4,000 + 800 = 4,800$ pounds. Now, we set up an equation for the total amount of gravel the three trucks can haul: $4,000 + 4,800 + x = 11,600$, where x is the amount Gary's truck hauls. Simplifying the equation: $9,800 + x = 11,600$, we subtract $9,800$ from both sides to get $x = 11,600 - 9,800 = 1,800$. Therefore, Gary's truck hauls 1,800 pounds. **Answer: 1,800**.
HypLoRA	Gissela's truck can haul 4,000 pounds of gravel. Gordy's truck can haul 800 more pounds than Gissela's truck, so it can haul 4,800 pounds of gravel. Together, Gissela and Gordy's trucks can haul 8,800 pounds of gravel. If the three trucks combined can haul 11,600 pounds, then Gary's truck can haul $11,600 - 8,800 = 2,800$ pounds of gravel. **Answer: 2,800**.

HypLoRA provides better understanding of number's hierarchies (especially for these leaf tokens) for prediction and accurate computation

Efficiency





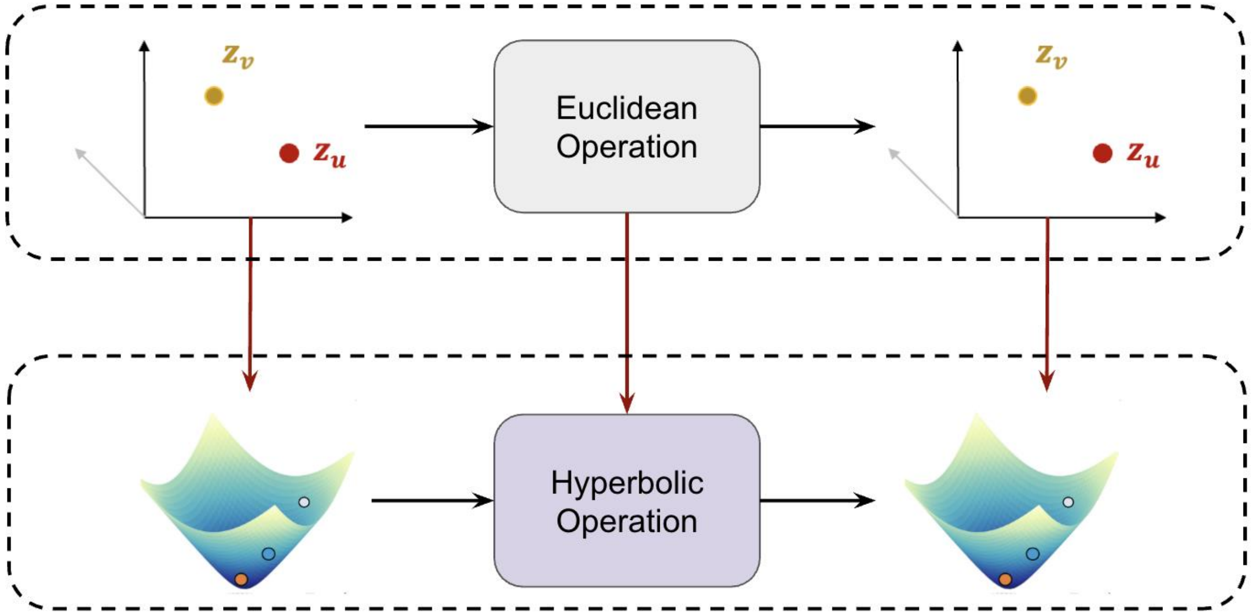
Although the proposed method increases the computational burden compared to the original LoRA, it remains significantly more efficient than DoRA, one of the state-of-the-art adapters.

Towards Non-Euclidean Foundation Models

"Hyperbolic-fy Operations/Modules in foundation models", e.g.,

- Residual Connection -> LResNet
- Attention Mechanism -> Hyperbolic Attention
- Linear Layer -> CARB
- Activation -> CAR
- LoRA -> HypLoRA

Euclidean Foundation Model



Hyperbolic Foundation Model

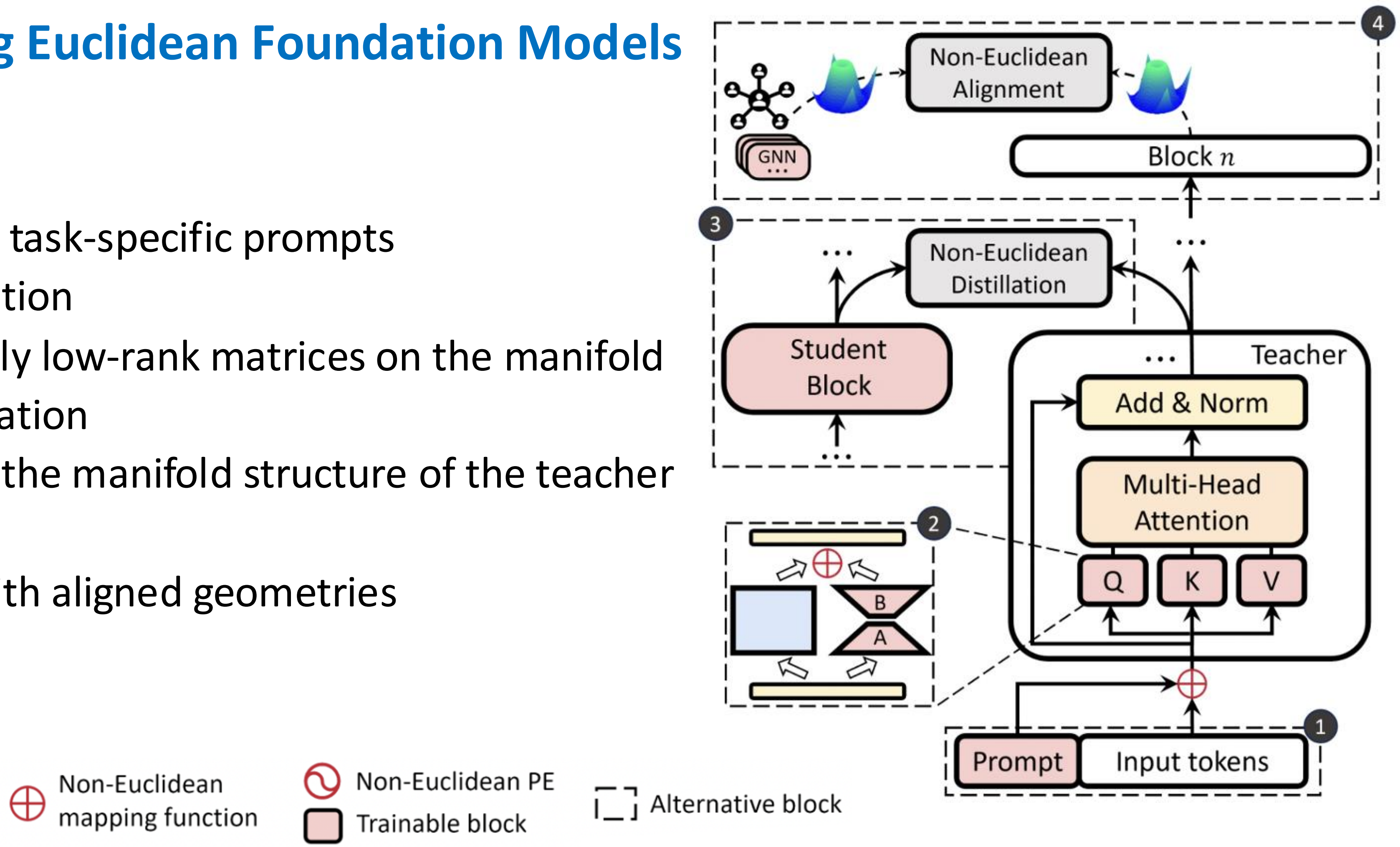
But what else??

*Goal: Encode geometric structure into the model that the model **cannot** do a good job learning otherwise*

Towards Non-Euclidean Foundation Models

Option 1: Fine-tuning Existing Euclidean Foundation Models

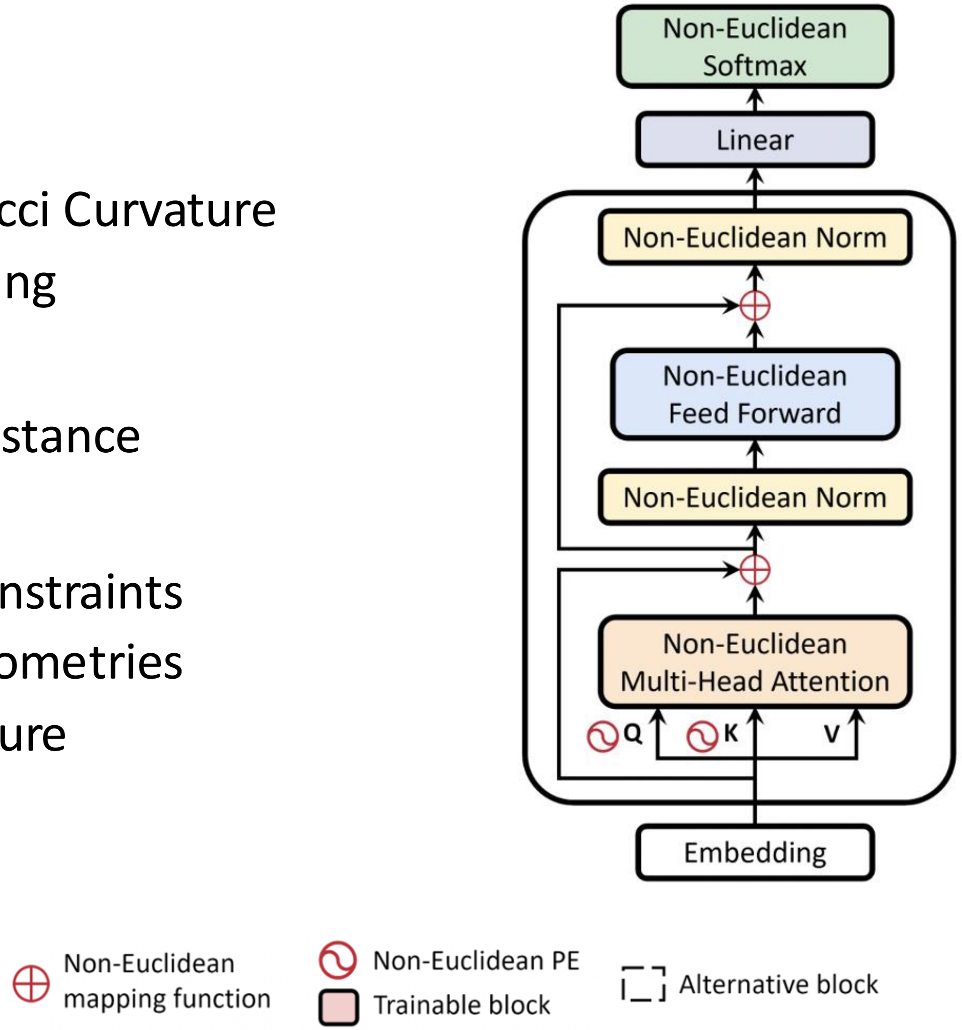
- Four Strategies
  1. Geometric Prompt Tuning
     - Add trainable geometric task-specific prompts
  2. Geometric Low-Rank Adaptation
     - Project input and multiply low-rank matrices on the manifold
  3. Geometric Knowledge Distillation
     - Teach student to inherit the manifold structure of the teacher
  4. Geometric Transfer Learning
     - Learn across domains with aligned geometries



Towards Non-Euclidean Foundation Models

Option 2: Pretraining from Scratch

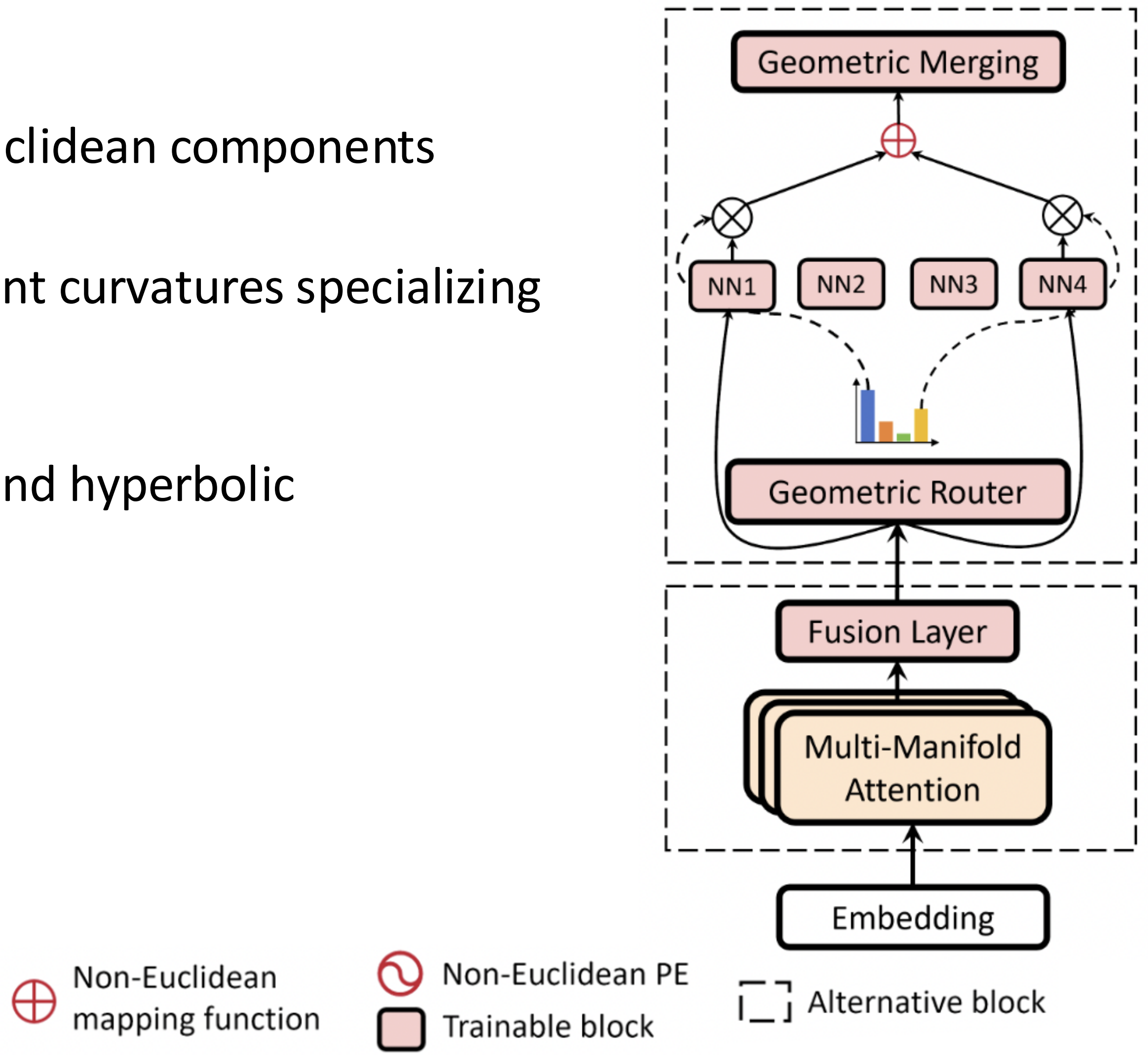
- **Curvature** Estimation/Trainable Curvature
  - Graph data: directly from structure topology, e.g. Ricci Curvature
  - Non-graph data: estimate through learned embedding
- Non-Euclidean **Attention** Mechanism
  - Define attention score through negative manifold distance
- Other Important Modules
  - Positional encoding taking into account manifold constraints
  - Residual connections need to be formulated with isometries
  - Layer/batch norm need to consider manifold curvature



Towards Non-Euclidean Foundation Models

Option 3: Hybrid Architectures

- Combine both Euclidean and non-Euclidean components
- Potentially build experts with different curvatures specializing in different types of corpus
- Consider non-Euclidean spaces beyond hyperbolic embedding spaces
- Build on top of the first 2 options



Challenges

- Building hyperbolic foundation models *would not be simple*
  - Require developing methods with abundance of knowledge in differential geometry
  - Special geometric functions and difficulty in implementing even basic operations, e.g. addition
  - Scattered prior research and incompatibilities
- **Issues with Existing Tools**
  - Limited Modules
  - Inflexibility and Unintuitive-Usage
    - Require extensive geometry knowledge
  - Limited Model Support: difficult to build advanced foundation models
  - Limited to one formulation of hyperbolic space (Poincare or Lorentz)

Introducing HyperCore!

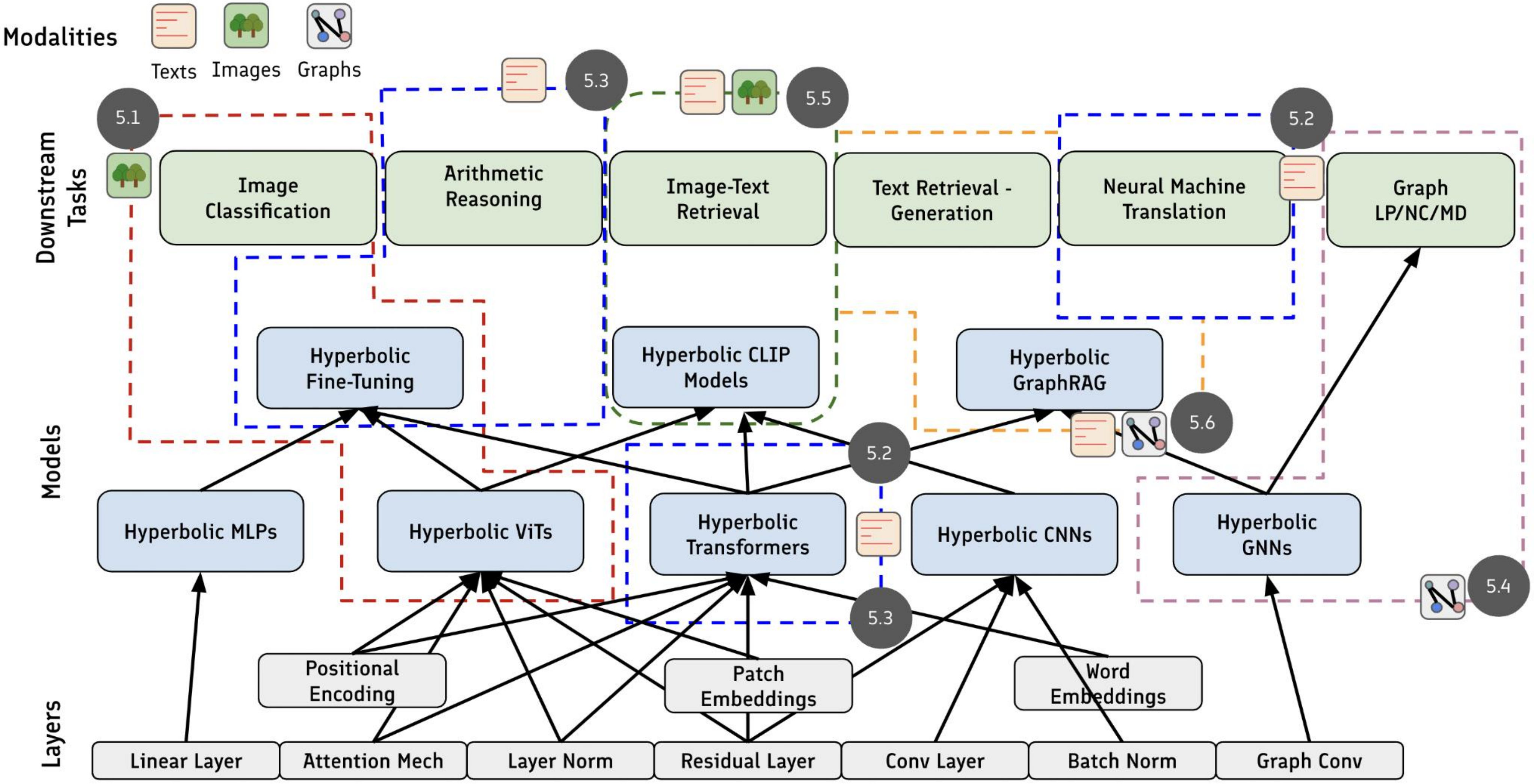
- **Flexible** to Create various SoTA models
  - Spotlight Examples: LViT, L-CLIP, Hyperbolic GraphRAG
- **Comprehensive** Modules and Model Support
- **Intuitive** Foundation Model Support
  - Focus on making it easier to build foundation model pipelines
- User **Accessibility**
  - Use the library without being an expert in hyperbolic geometry

Framework	MLPs	GNNs	CNNs	Transformers	ViTs	Fine Tuning	CLIP	Graph RAG	$\mathbb{L}^{n,K}$	$\mathbb{P}^{n,K}$
HypLL [55]	✓	✗	✓	✗	✗	✗	✗	✗	✗	✓
Hyperlib [1]	✓	✓	✗	✗	✗	✗	✗	✗	✓	✓
HyperCore	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓

Library Overview

- **Modules**
  - Neural network layers (e.g. linear, convolutional, MLR)
  - Transformer layers (e.g. softmax self-attention, linear attention, latent attention, positional encoding, word embedding, patch embedding)
  - Graph related (e.g. graph convolutional layers and neighborhood aggregation)
  - Fine-tuning
  - Essential modules (e.g. batch and layer normalization, residual connection, pooling layers)
- **Optimizers**
  - Support for different training schemes on Euclidean v.s. manifold parameters
- **Manifold**
  - Basic manifold operations and additional operations (e.g. concatenation and splitting vectors, hyperbolic entailment cones)

Snapshot of Library Taxonomy



Example: Transformer Block

Euclidean Transformer Block

```
import torch
from torch import nn
from collections import OrderedDict

class TransformerBlock(nn.Module):
    def __init__(self, d_model: int, n_head: int):
        super().__init__()

        self.attn = nn.MultiheadAttention(d_model, n_head,
    batch_first=True)
        self.ln_1 = nn.LayerNorm(d_model)
        self.mlp = nn.Sequential(
            OrderedDict(
                [
                    ("c_fc", nn.Linear(d_model, d_model * 4)),
                    ("gelu", nn.GELU()),
                    ("c_proj", nn.Linear(d_model * 4, d_model)),
                ]
            )
        )
        self.ln_2 = nn.LayerNorm(d_model)

    def forward(self, x: torch.Tensor, attn_mask: torch.Tensor |
    None = None):
        lx = self.ln_1(x)
        ax = self.attn(lx, lx, lx, need_weights=False, attn_mask=
    attn_mask)[0]
        x = x + ax
        x = x + self.mlp(self.ln_2(x))
        return x
```

Lorentz Transformer Block w/ HyperCore

```
import torch
import torch.nn as nn
import hypercore.nn as hnn
from collections import OrderedDict

class LTransformerBlock(nn.Module):
    def __init__(self, manifold, d_model: int, n_head: int):
        super().__init__()
        dim_per_head = d_model // n_head
        self.manifold = manifold
        self.attn = hnn.LorentzMultiheadAttention(manifold,
    dim_per_head, dim_per_head, n_head, attention_type='full',
    trans_heads_concat=True)
        self.ln_1 = hnn.LorentzLayerNorm(manifold, d_model -1)
        self.mlp = nn.Sequential(
            OrderedDict(
                [
                    ("c_fc", hnn.LorentzLinear(manifold, d_model,
    d_model*4-1)),
                    ("gelu", hnn.LorentzActivation(manifold,
    activation=nn.GELU())),
                    ("c_proj", hnn.LorentzLinear(manifold, d_model
    *4, d_model-1)),
                ]
            )
        )
        self.ln_2 = hnn.LorentzLayerNorm(manifold, d_model-1)
        self.res1 = hnn.LResNet(manifold, use_scale=True)
        self.res2 = hnn.LResNet(manifold, use_scale=True)

    def forward(self, x, attn_mask=None):
        lx = self.ln_1(x)
        ax = self.attn(lx, lx, output_attentions=False, mask=
    attn_mask)
        x = self.res1(x, ax)
        x = self.res2(x, self.mlp(self.ln_2(x)))
        return x
```

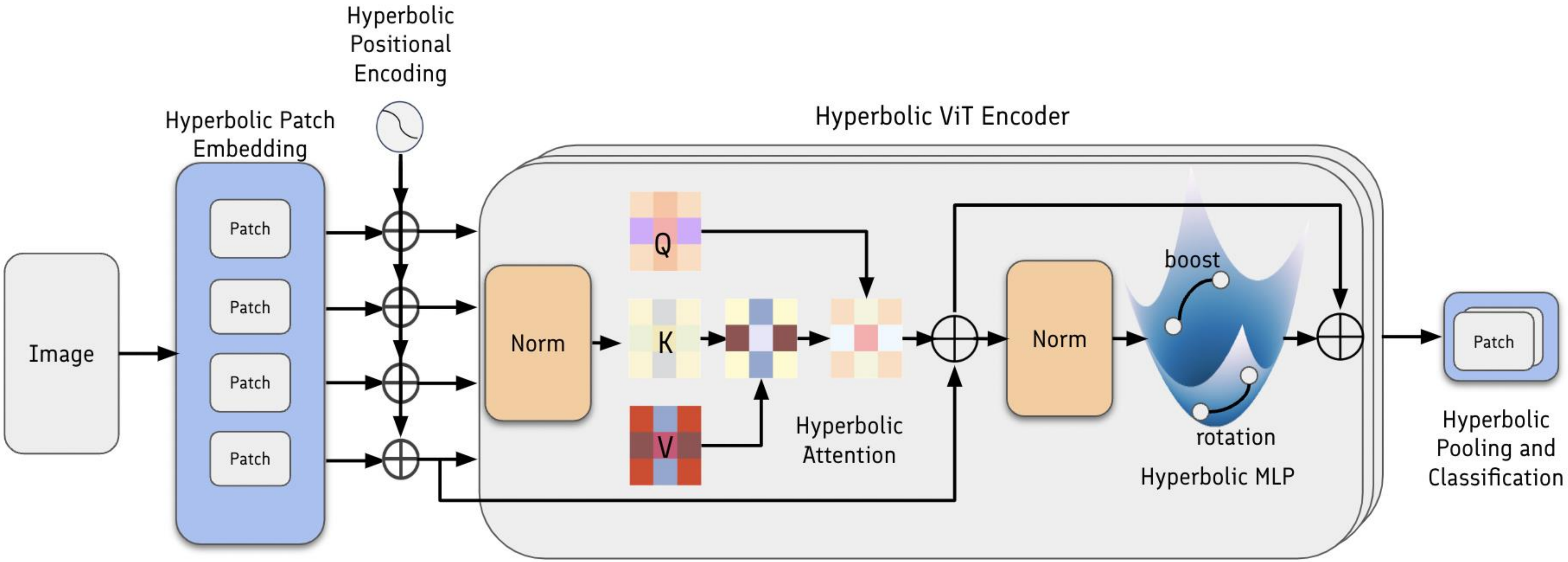
Hyperbolic Fine-tuning of LLMs – Option 1

- Recreate experiments from HypLoRA: hyperbolic fine-tuning of Gemma-7B and LLaMA3-8B
- Training set GSM8K, MAWPS, MAWPS-single, AQuA, and the math-10K dataset consisting of step-by-step rationales generated by ChatGPT
- Testing set: part of GSM8K + MAWPS + AQuA
    - No overlap with training set

Model	MAWPS(8.5%)	GSM8K(46.9%)	AQuA(9.0%)
Gemma-7B [53]	91.2	68.7	32.9
LLaMA3-8B [27]	**91.5**	**73.3**	**34.3**

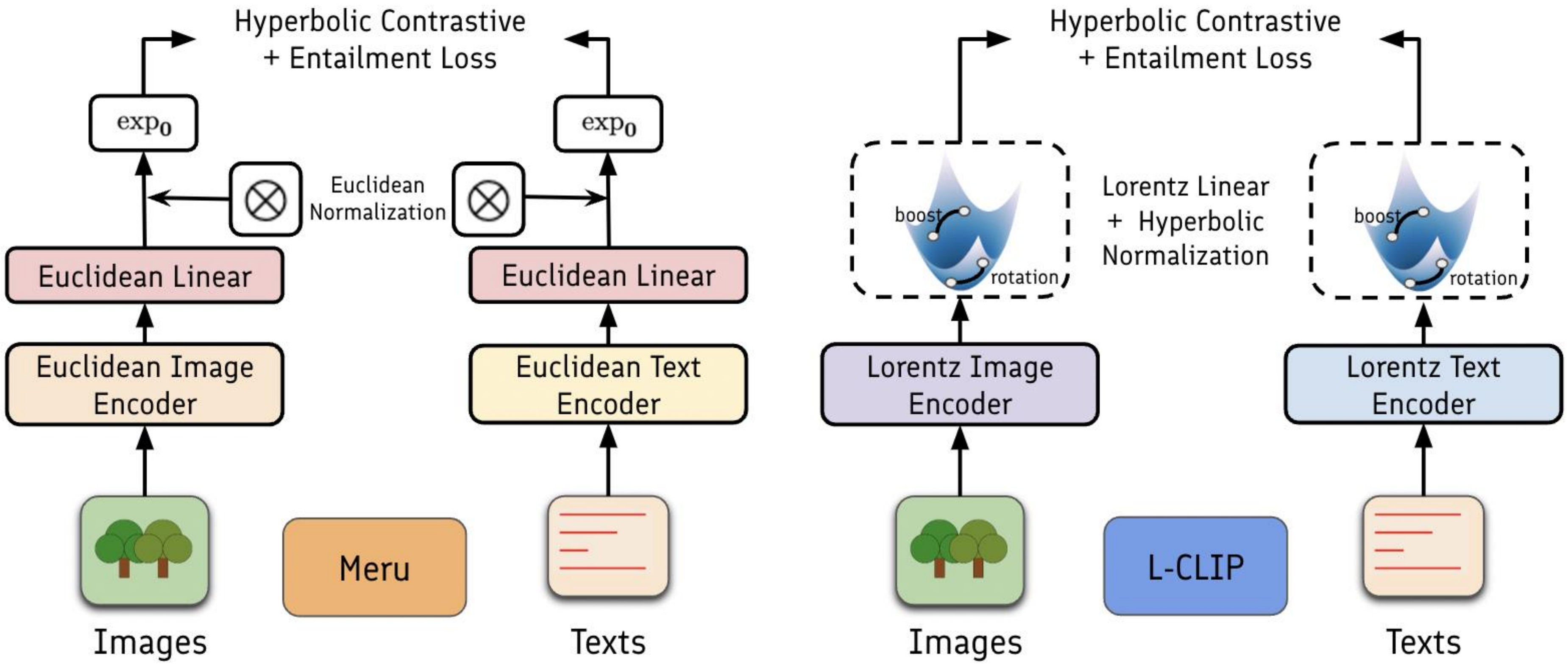
New Hyperbolic Foundation Models w/ HyperCore: LViT – Option 2

- First fully hyperbolic vision transformer with a fine-tuning pipeline, built with HyperCore



New Hyperbolic Foundation Models w/ HyperCore: L-CLIP – Option 2

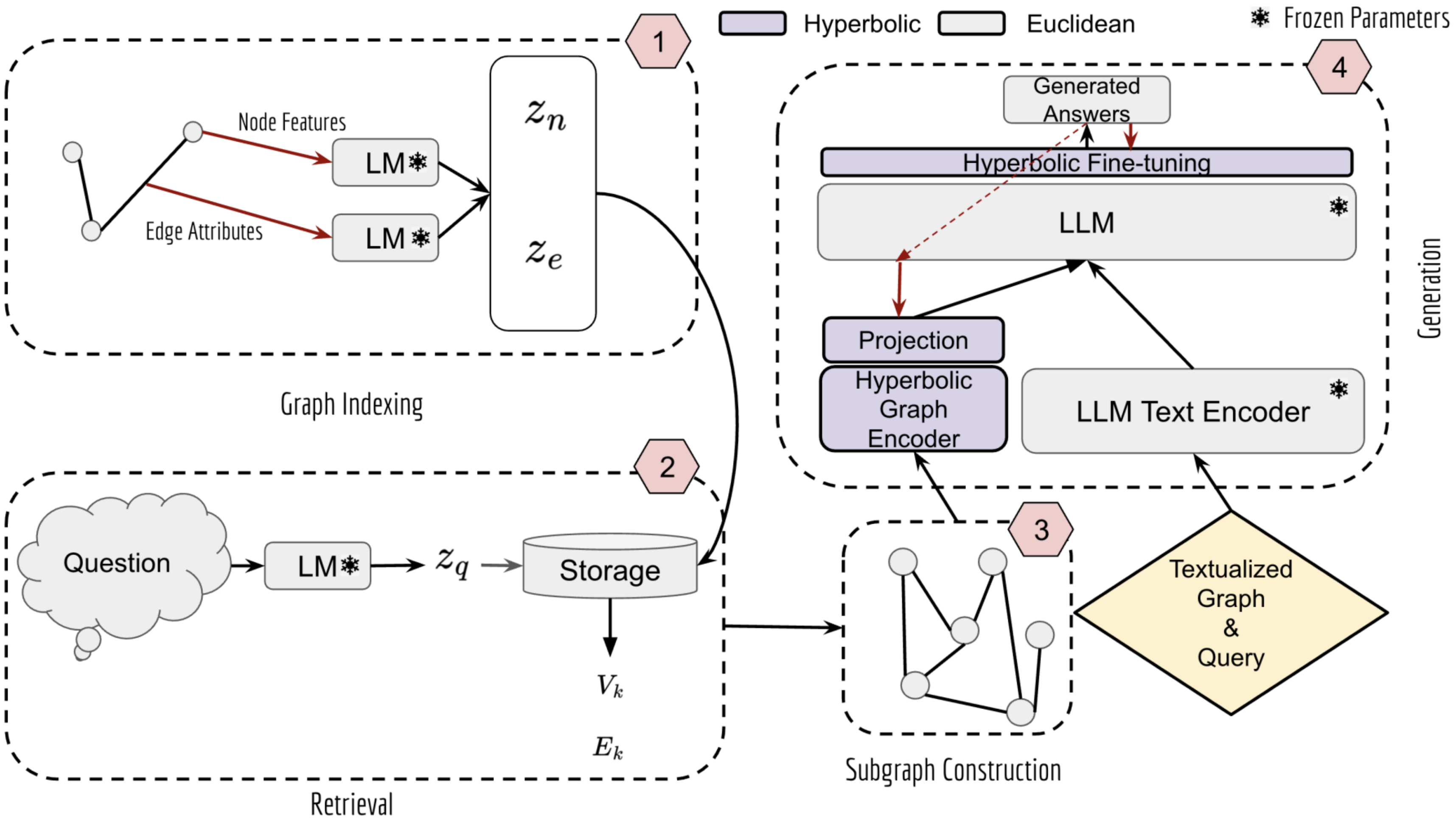
- First fully hyperbolic multi-modal CLIP model
  - Compared to MERU, which is a hybrid prior work



New Hyperbolic Foundation Models w/ HyperCore: HypGraphRAG – Option 3

First Hyperbolic GraphRAG model:
- Uses a hyperbolic graph encoder
- Uses hyperbolic fine-tuning

Better represent the knowledge graph structure



Testing New Hyperbolic Models – LViT

- Image Classification with LViT
  - Fine-tuning with HypLoRA on smaller datasets

- Datasets
  - ImageNet-1K: 1.2M images of 1,000 classes
  - CIFAR10 and CIFAR100: 60K images of 10 (100) classes
  - TinyImageNet: 100K images of 200 classes

Every hyperbolic model here is implemented with HyperCore

Dataset	CIFAR-10	CIFAR-100	TINY-IMAGENET	IMAGENET
Hyperbolicity	$\delta = 0.26$	$\delta = 0.23$	$\delta = 0.20$	-
HCNN [54]	95.02 ± 0.19	77.31 ± 0.21	65.01 ± 0.29	-
Poincaré ResNet [6]	94.71 ± 0.13	76.91 ± 0.34	63.11 ± 0.59	-
ViT [21]	98.13	87.13	-	77.91
HVT [24]	61.44	42.77	40.12	78.2
LViT (built by us)	85.02	69.11	53.01	79.4
LViT (fine-tuned w/ HypLoRA)	**98.18**	**87.36**	**74.11**	**79.4**

Hyperbolic ResNets

Euclidean ViT

Tangent Space ViT

Testing New Hyperbolic Models – L-CLIP & Hyperbolic GraphRAG

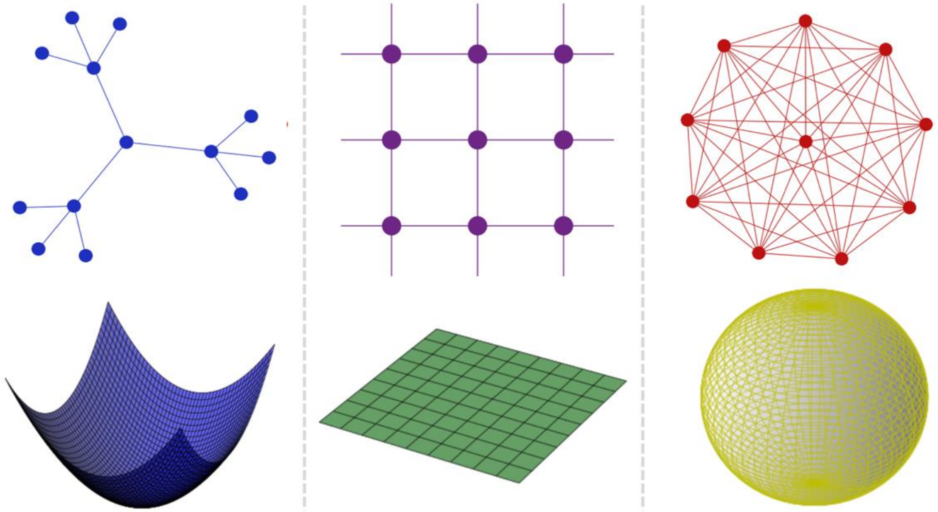
- Image-Text Retrieval on COCO benchmark with L-CLIP
  - Image encoder: LViT; Text encoder: hyperbolic Transformer
- HypGraphRAG: Question-answering tasks in a graph QA dataset (WebQSP)
  - Skip-connected hyperbolic GNN; LLaMA3.1-8B fine-tuned with HypLoRA

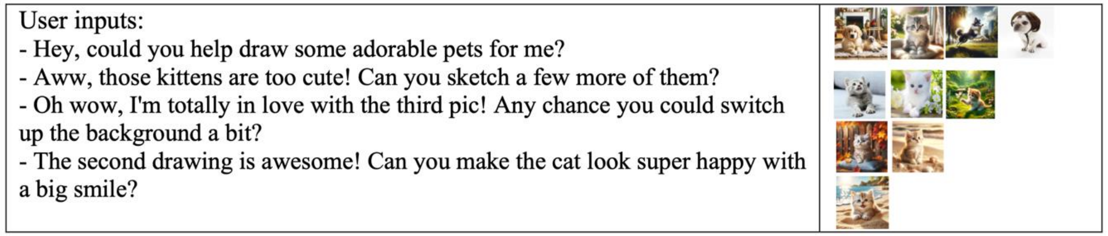
Experimental Goal: To demonstrate what's possible

Model	L-CLIP		HypGraphRAG
Dataset	COCO		WebQSP
Task	Image-Text Retrieval		Question-answering
Metric	Recall@5	Recall@10	Hi@1
Restults	28.0	38.1	73.89 ± 1.09

Future works

Ultimate goal: Combine non-Euclidean foundation model with large model for Geometric-aware AI





Examples of generating images from corse-grained to fine-grained, aligning human cognition process

From hyperbolic space to adaptive curvature space

From language model to multimodal models

Non-Euclidean Foundation Model

Multimodal LM

Geometric AI

References

- Hypformer: Exploring Efficient Hyperbolic Transformer Fully in Hyperbolic Space (KDD 2024)

Menglin Yang, Harshit Verma, Delvin Ce Zhang, Jiahong Liu, Irwin King, Rex Ying

- Hyperbolic Fine-tuning of Large Language Models (NeurIPS 2024 Workshop)

Menglin Yang, Aosong Feng, Bo Xiong, Jiahong Liu, Irwin King, Rex Ying

- Lorentzian Residual Network (KDD 2025)

Neil He, Menglin Yang, Rex Ying

- Position: Beyond Eculidean – Foundation Models Should Embrace Non-Euclidean Geometries (ICML 2025 in sub)

Neil He, Jiahong Liu, Buze Zhang, Ngoc Bui, Ali Maatouk, Menglin Yang, Irwin King, Melanie Weber, Rex Ying

- HyperCore: The Core Framework for Building Hyperbolic Foundation Models with Comprehensive Modules

TheWebConf 2025 NEGEL Workshop (Oral)

Neil He, Menglin Yang, Rex Ying

Thank You

YALE

HARVARD

CUHK

NSF

GORDON AND BETTY MOORE FOUNDATION

amazon

Goldman Sachs

NetApp

Snapchat

SAMSUNG

Google