Hyperattentive Residuals

Grant Murphy
Massachusetts Institute of Technology

Neil He
University of Illinois Urbana-Champaign

Aryan Sharma
Yale University

Menglin Yang
Hong Kong University of Science and Technology (Guangzhou)

Rex Ying
Yale University

Abstract

Hyperbolic neural networks have been shown to be highly effective due to their ability to effectively model data with latent hierarchical structure in domains like graph learning, image classification, and language modeling. However, these models have not been scaled to a size commensurate with modern foundation models, partially due to issues with training stability when training larger models. In this work, we show that traditional residual connection methods which run in sequence may be the culprit for this instability and we propose a new method to combat this. Our residual connection, LAttnRes, performs an attention-weighted aggregation across all layers up to the point of the residual connection, allowing embeddings throughout the forward pass to embed in a more stable region and become more expressive. We test this method on a variety of domains to show its competitive performance and stability at scale. Building LAttnRes and further modules for effective scaled architectures are important steps in building fully hyperbolic foundation models which can compete at the frontier.

Keywords

Hyperbolic Geometry; Residual Connections; Lorentz Model; Training Stability

1 Introduction

Recent work has increasingly explored neural network architectures that operate beyond traditional Euclidean geometry, opening new directions for representation learning [18, 19, 29]. In particular, hyperbolic neural networks have attracted substantial attention as they provide a natural geometry for data with hierarchical or tree-like structure [3, 7, 17]. Hyperbolic geometry's constant negative curvature induces exponential volume growth, allowing hierarchical relationships to be embedded with lower distortion than in Euclidean space [16, 21]. This geometric advantage has motivated the use of hyperbolic representations across a broad range of domains, including graph learning [3, 17, 30] and visual representation learning [4, 6].

At the same time, recent years have seen the rise of large pre-trained foundation models achieving incredible performance and diverse applications [9, 20, 22]. However, while some preliminary efforts were made [11], hyperbolic models have not enjoyed the same level of scaling as their Euclidean counterparts. This is in part due to the more unstable nature of forming embeddings in hyperbolic space in comparison to unconstrained Euclidean space, especially as the dimensionality of both parameter and embedding spaces grow, coupled with the need to scale these spaces in order to train high-performing pre-trained models [15]. Efforts to solve issues of training instability are crucial to building hyperbolic models which can be competitive with the frontier of foundation models and have strong returns to performance on compute-intensive training runs.

In this work, we demonstrate through systematic experiments that existing hyperbolic models still face significant training instabilities due to their residual connection methods. In particular, nearly all methods in the literature [1, 5] rely on sequential methods which are called with an input accumulated over all previous residual connections along with a new residual vector. Many [14, 28] also use centroid-based methods which take a midpoint of the two embeddings fed into the connection. These cause performance issues such as diluting the impact of early layers, creating excessive mapping errors over the course of the forward pass, forcing residuals to partially encode the input within their embedding, and bringing embeddings further away from the numerically stable subspace near the origin.

To alleviate these issues, we propose a residual connection which performs a weighted midpoint across all prior feedforward and attention outputs. It distributes weights to each embedding with a layer-invariant attention mechanism that provides flexible encodings of all prior layers independently in a single operation, allowing attention and feedforward operations to transform inputs into expressive embeddings and maintain stability throughout training.

2 Preliminaries

This section summarizes the primary elements of hyperbolic geometry used in this work, with particular attention to the Lorentz model and the standard constructions to support addition under it.

Lorentz model. The n-dimensional Lorentz model is the Riemannian manifold $\mathbb{L}^{K,n}$ associated with constant negative curvature $K < 0$. A point $\mathbf{x} \in \mathbb{L}^{K,n}$ is written as $[x_t, \mathbf{x}_s]^T$, where $x_t \in \mathbb{R}$ denotes the time-like coordinate and $\mathbf{x}_s \in \mathbb{R}^n$ denotes the space-like coordinates. The canonical inner product for the model, the *Lorentzian inner product*, is defined for $\mathbf{x}, \mathbf{y} \in \mathbb{L}^{K,n}$ as $\langle \mathbf{x}, \mathbf{y} \rangle_{\mathcal{L}} = -x_t y_t + \mathbf{x}_s^T \mathbf{y}_s$. The corresponding Lorentzian norm is $\|\mathbf{x}\|_{\mathcal{L}} \coloneqq \sqrt{|\langle \mathbf{x}, \mathbf{x} \rangle_{\mathcal{L}}|}$. Equivalently, the Lorentz manifold is the upper sheet of the hyperboloid

$$\mathbb{L}^{K,n} = \left\{ \mathbf{x} \in \mathbb{R}^{n+1} : \langle \mathbf{x}, \mathbf{x} \rangle_{\mathcal{L}} = \frac{1}{K},\ x_t > 0 \right\}.$$

Its origin is $\mathbf{o} = [\sqrt{-1/K}, 0, \ldots, 0]^T$.

Tangent space. For any point $\mathbf{x} \in \mathbb{L}^{K,n}$, the tangent space can be defined as a local approximation of the manifold at that point to a separate space which is isometric to ordinary Euclidean space.

It consists of all vectors in the ambient space that are Lorentz-orthogonal to $\mathbf{x}$: $\mathcal{T}_{\mathbf{x}}\mathbb{L}^{K,n} = \left\{\mathbf{y} \in \mathbb{R}^{n+1} : \langle \mathbf{x}, \mathbf{y}\rangle_{\mathcal{L}} = 0\right\}$.

Exponential and logarithmic maps. At each $\mathbf{x} \in \mathbb{L}^{K,n}$, the exponential map $\exp_{\mathbf{x}}^{K} : \mathcal{T}_{\mathbf{x}}\mathbb{L}^{K,n} \to \mathbb{L}^{K,n}$ sends a tangent vector to the manifold, while the logarithmic map $\log_{\mathbf{x}}^{K} : \mathbb{L}^{K,n} \to \mathcal{T}_{\mathbf{x}}\mathbb{L}^{K,n}$ performs the inverse operation. These maps are given by

$$\exp_{\mathbf{x}}^{K}(\mathbf{v}) = \cosh(\alpha)\mathbf{x} + \frac{\sinh(\alpha)}{\alpha}\mathbf{v}, \qquad \alpha = \sqrt{-K\langle \mathbf{v}, \mathbf{v}\rangle_{\mathcal{L}}}, \tag{1}$$

and

$$\log_{\mathbf{x}}^{K}(\mathbf{y}) = \frac{\cosh^{-1}(\beta)}{\sqrt{\beta^2 - 1}}\left(\mathbf{y} - \beta\mathbf{x}\right), \qquad \beta = K\langle \mathbf{x}, \mathbf{y}\rangle_{\mathcal{L}}. \tag{2}$$

Parallel transport. Parallel transport provides the hyperbolic analogue of translation by mapping vectors between tangent spaces. Given $\mathbf{z} \in \mathcal{T}_{\mathbf{x}}\mathbb{L}^{K,n}$, its transport from $\mathcal{T}_{\mathbf{x}}\mathbb{L}^{K,n}$ to $\mathcal{T}_{\mathbf{y}}\mathbb{L}^{K,n}$ is

$$\mathbf{P}_{\mathbf{x}\to\mathbf{y}}(\mathbf{z}) = \mathbf{z} + \frac{\langle \mathbf{y}, \mathbf{z}\rangle_{\mathcal{L}}}{-1/K - \langle \mathbf{x}, \mathbf{y}\rangle_{\mathcal{L}}}(\mathbf{x} + \mathbf{y}). \tag{3}$$

Lorentzian centroid. Lorentzian centroid is a method for computing a weighted midpoint in the Lorentz model. For points $\mathbf{x} = x_1, \ldots, x_n \in \mathbb{L}^{K,n}$ and weights $\mathbf{w} = w_1, \ldots, w_n \in \mathbb{R}$,

$$\text{LorentzianCentroid}(\mathbf{w}, \mathbf{x}) = \frac{\sum_{i=1}^{n} w_i x_i}{\sqrt{-K}|\| \sum_{i=1}^{n} w_i x_i\|_{\mathcal{L}}|} \tag{4}$$

Hyperbolic addition methods. Several methods have been proposed for performing addition in hyperbolic space. The parallel transport method of Möbius addition from [3, 7] and the tangent space addition method used for aggregation in [3] rely on tangent spaces. A third approach is the space-like dimension addition method from [1], given by

$$\mathbf{x} \oplus_S \mathbf{y} = \left[\sqrt{\|\mathbf{x}_s + \mathbf{y}_s\|^2 - \frac{1}{K}}, \mathbf{x}_s + \mathbf{y}_s\right]^T, \tag{5}$$

where $\mathbf{x}_s$ and $\mathbf{y}_s$ denote the space-like components of $\mathbf{x}$ and $\mathbf{y}$.

An unweighted Lorentzian centroid operation between two points may also be used as a form of addition. Building upon this for trainable hyperbolic models, LResNet [14] forms a notion of addition by performing a Lorentzian centroid with learnable weights and includes an optional scaling γ to account for mapping error in the centroid operation.

3 Motivation

Prior methods suffer from a key issue of embeddings from early layers to become diluted as they lie deep within the nested aggregations. Take for example an evenly weighted Lorentz centroid:

$$d_1 = \sqrt{-K}\left|\|\mathbf{y} + \mathbf{z}\|_{\mathcal{L}}\right| \tag{6}$$

$$\text{LorentzianCentroid}((1,1),(\mathbf{y},\mathbf{z})) = \frac{\mathbf{y} + \mathbf{z}}{d_1} \tag{7}$$

$$d_2 = \sqrt{-K}\left|\left\|\mathbf{x} + \frac{\mathbf{y} + \mathbf{z}}{d_1}\right\|_{\mathcal{L}}\right| \tag{8}$$

$$\begin{aligned}&\text{LorentzianCentroid}\left((1,1),(\mathbf{x}, \text{LorentzianCentroid}((1,1),(\mathbf{y},\mathbf{z})))\right)\\ &= \frac{\mathbf{x} + \frac{\mathbf{y}+\mathbf{z}}{d_1}}{d_2} = \frac{\mathbf{x}}{d_2} + \frac{\mathbf{y}}{d_1 d_2} + \frac{\mathbf{z}}{d_1 d_2}\end{aligned} \tag{9}$$

Here, x and y are diluted by additional scaling factors while z remains closer to its original embedding. Although this can be solved somewhat by weighting inputs more heavily than residuals in outer centroid operations, simple weighting is not specific to each embedding and is an incomplete solution. Additionally, each successive projection back to the manifold creates additional mapping errors. Thus, in a deep model where these early layers are heavily nested, they will be strongly diluted and there will be large mapping errors implicit in the construction of the final embeddings.

Training Instability of Prior Methods Here we demonstrate the instabilities of prior methods when the model is scaled to larger sizes, where we doubled the size of the HELM-D model [11] in embedding dimensions, feedforward layer width, and attention heads. As shown in Figure 1, while they improve stability over tangent space methods [2, 14], both LResNet and the Space Addition method suffer from early training loss plateaus. Additionally, the vector norms in the early layers are extremely large for both methods, suggesting that the model is attempting to compensate for the diluted signal from earlier layers. This effect is further amplified in hyperbolic space, as geodesics diverge at an exponential rate. In comparison, LAttnRes, which is our method that we introduce below, alleviates this issue and stabilizes the norm and training dynamics, as also shown in Figure 1.

4 Method

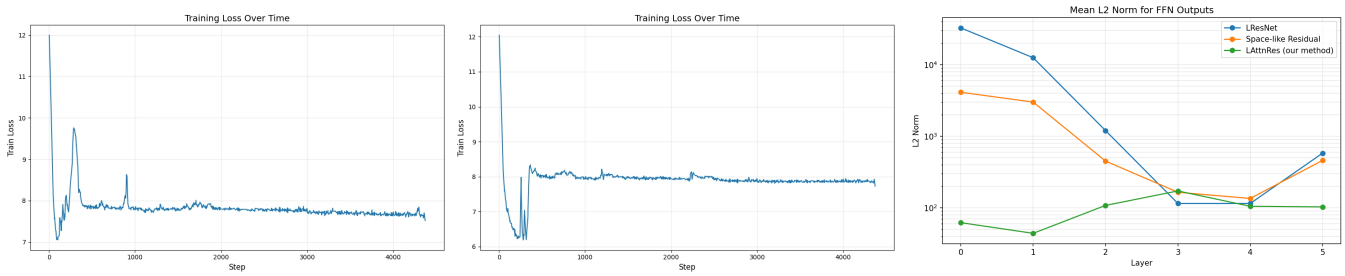
We propose Hyperattentive Residuals (**LAttnRes**) to remedy the limitations found with existing hyperbolic residual connection methods. We take inspiration from Attention Residuals [24], a method used to overcome losing information from early layers and instability from output growth in Euclidean Transformer models. Instead of performing repeated connections on an accumulated residual over the course of the forward pass, we take a weighted Lorentzian centroid of outputs from all prior attention and feedforward layers, along with the newly computed embedding, so that all layers have a strong ability to influence the representation. We define LAttnRes as

$$\text{LAttnRes}\big(h_l, h_{l-1}\big) = \text{LorentzianCentroid}\big(\alpha_{0\to l:l\to l}, h_{0:l}\big) \tag{10}$$

where $\alpha_{i\to l}$ is an attention weight for each of the layers h_i for $i = 0, \ldots, l$ on the residual connection following layer l. For simplicity, we denote each attention and feedforward operation as its own layer and the initial embedding as h_0. These $\alpha_{i\to l}$ are computed by taking softmax over layers' dot product similarity with a weight vector w_l corresponding to that specific layer:

$$\alpha_{i\to l} = \frac{\exp\left(w_l^T h_i\right)}{\sum_{j=0}^{l}\exp\left(w_l^T h_j\right)} \tag{11}$$

Weight vectors are initialized to $\vec{0}$ and learned throughout training to learn an unbiased representation of the importance of all past layers to the new embedding.



(a) LResNet Training PPL **(b) Space Addition Train PPL** **(c) FFN vector norm across layers**

Figure 1: (a), (b) Training dynamics of LResNet and space-like addition, (c) the Euclidean vector norms across layers following FFN of final training checkpoints under inference on the test set of HellaSwag [31]

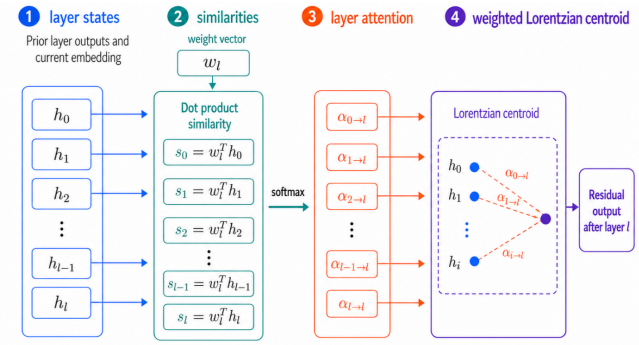


Figure 2: Overview of the Hyperattentive Residuals method. The module first computes similarity scores with the current layer's weight vector, then applies softmax over these scores to weight all layers in the next residual output.

By learning embedding-specific weights, LAttnRes leverages a strong learnability to prevent dilution of early layers. Aggregating over all individual layers at each residual connection minimizes mapping error from projections back to the manifold accumulating over many connections and allows outputs from each layer to express only their contribution to the representation.

5 Experiments

We evaluate LAttnRes against traditional centroid-based residual connections on graph node classification with the Hypformer [28] architecture, image classification with a hyperbolic ViT, and language modeling with HELM-D [11]. In each setting, we sweep width, depth, or overall model scale.

5.1 Graph Node Classification

Setup. We replace Hypformer's Lorentzian centroid residual connection with LAttnRes and evaluate performance on Amazon2M, ogbn-arxiv, and ogbn-proteins. Specifically, we sweep the number of hidden channels (HC) $\in \{256, 512, 1024\}$ and layer depth $\in \{1, 2, 4, 8\}$ for both methods while holding all other hyperparameters fixed. Test accuracy is the best over five seeds.

Results. Table 1 reports results across our sweep. LAttnRes shows a slight edge over the centroid method on two of the three datasets and matches closely in the third, with all differences within seed variance. This parity holds across every (HC, depth) combination, demonstrating that LAttnRes is a drop-in replacement for LResNet on standard graph benchmarks without a drop in accuracy. Furthermore, both methods reach the same memory bound at the upper end of the sweep on Amazon2M and ogbn-arxiv.

5.2 Image Classification

Setup. We use a hyperbolic ViT (LViT) [13] with LResNet for the residual connection and the DeiT-III [25] training recipe. The base configuration uses depth 12 for 400 epochs over 3 seeds. We additionally run a depth sweep on CIFAR-100 under simple training, varying LResNet's residual scaling factor $\gamma \in \{10, 20, 30\}$ across depths $\{12, 18, 24, 32\}$ and comparing to LAttnRes under a single fixed configuration.

Results. Table 2 reports test accuracy. At the base configuration, LResNet obtains 94.85, 71.77, and 54.27 on CIFAR-10, CIFAR-100, and Tiny-ImageNet respectively, against LAttnRes's 91.14, 65.81, and 43.11 — a small-scale accuracy cost we revisit in Section 6.

Table 3 reports the depth sweep on CIFAR-100. Under simple training, LResNet consistently collapses due to instability — each value of γ only supports a narrow range of depths while LAttnRes trains stably across all depths under one fixed configuration. Stability extends beyond the depths shown, as we found LAttnRes to remain stable at depths 48 and 64. Whether stronger recipes such as DeiT-III extend LResNet's working depth is left to future work.

5.3 Language Modeling

Setup. We train HELM-D on English Wikipedia [26] tokenized with the Llama-3.1 [8] tokenizer (128k vocabulary). The base configuration uses LResNet for residual connection, 6 layers, 6 attention heads, and a hidden dimension and feedforward width 390, while the scaled (2×) configuration keeps the layer count constant and doubles the embedding dimension, FFN width, and attention heads. All other hyperparameters are held fixed across the four cells.

Results. We find that switching the residual connection to LAttnRes causes a slight improvement in training loss, but we see a much starker change once we scale the model further. Table 4 reports final training loss on the scaled models. LAttnRes *improves* over the scaling while the performance for LResNet drops dramatically. This indicates that training larger models becomes untenable

Table 1: Test accuracy on graph node classification with Hypformer. OOM denotes out of memory during training or testing. All results are taken over five seeds.

		Amazon2M		ogbn-arxiv		ogbn-proteins	
HC	L	centroid	LAttnRes	centroid	LAttnRes	centroid	LAttnRes
256	1	89.39 ± 0.03	89.44 ± 0.03	73.35 ± 0.11	73.18 ± 0.13	80.78 ± 0.16	80.87 ± 0.13
256	2	89.38 ± 0.05	89.45 ± 0.05	73.34 ± 0.06	73.22 ± 0.25	80.75 ± 0.21	80.85 ± 0.26
256	4	OOM	OOM	73.40 ± 0.09	73.25 ± 0.13	80.75 ± 0.07	80.86 ± 0.16
512	1	OOM	OOM	73.10 ± 0.07	73.22 ± 0.13	80.81 ± 0.14	80.79 ± 0.14
512	2	OOM	OOM	73.34 ± 0.15	73.27 ± 0.13	80.58 ± 0.06	80.82 ± 0.25
1024	1	OOM	OOM	72.82 ± 0.18	72.93 ± 0.11	80.84 ± 0.16	80.86 ± 0.07
1024	2	OOM	OOM	OOM	OOM	80.77 ± 0.23	80.91 ± 0.08

Table 2: Test accuracy of LViT under DeiT-III at the base configuration. All numbers are means over three seeds.

Dataset	LResNet	LAttnRes
CIFAR-10	94.85	91.14
CIFAR-100	71.77	65.81
Tiny-ImageNet	54.27	43.11

Table 3: LViT depth sweep on CIFAR-100 under simple training, comparing LResNet performance over depth and residual scaling while keeping LAttnRes unscaled. COLLAPSE denotes failure to train above the ∼10% random baseline.

	LResNet			
Depth	γ=10	γ=20	γ=30	LAttnRes
12	COLLAPSE	74.08 ± 1.09	74.04 ± 0.14	71.32 ± 1.34
18	COLLAPSE	COLLAPSE	74.87 ± 0.30	72.12 ± 1.32
24	COLLAPSE	COLLAPSE	75.99 ± 0.23	71.01 ± 1.79
32	COLLAPSE	COLLAPSE	COLLAPSE	72.13 ± 1.31

Table 4: HELM-D final training loss on English Wikipedia for models with feedforward layer width, embedding dimension, and attention heads scaled (2×).

FFN width	dim	heads	LResNet	LAttnRes
390	390	6	4.12	3.63
780	780	12	7.90	**3.22**

with LResNet due to a failure to remain stable under scaled parameter counts and embedding spaces while performance continues to improve with LAttnRes under this shift.

6 Future Work

Several directions follow naturally to strengthen this work. We believe that sweeping LAttnRes hyperparameters, such as curvature schedules, learning rate, training duration, and more model-specific parameters should narrow the accuracy gap to match or exceed the performance of original methods on standard benchmarks. Beyond this, we expect LAttnRes's stability advantages to scale cleanly to larger configurations of HELM, where LResNet's failure to use additional capacity should further amplify.

Investigating further how the weight vector is learned throughout training and how attention weights form across layers will assist in finding these parameters. More importantly, it will help us to reach a deeper theoretical understanding of what makes the method effective over others in the cases where it achieves superior performance.

Further modifications to the method itself may produce better performance as well. A post-residual scaling may boost performance in some cases, as it does for LResNet, though LAttnRes's full aggregation of past layers at every residual should combat the repeated accumulation of mapping error over each residual connection which this scaling helped LResNet correct for. The original Attention Residuals work normalized layers before computing similarity with the weight vector which may provide even stronger stability to the method, but it may cause us to lose the hierarchical expressiveness which the layers express through their norms. They additionally interpret the role of the weight vector as more of a query vector and the layers as a series of keys; we can experiment with a fully hyperbolic weight vector, allowing us to compute similarity with past layers via negative hyperbolic distance or some other metric more attuned to the manifold.

More broadly, a head-to-head comparison with Attention Residuals at matched scale would directly tell us whether the stability advantage we report here is geometric or simply a property of softmax-routed residuals. Such constructions can also suffer from softmax pathologies analogous to attention entropy collapse [32], attention sinks [10, 27], and attention logit explosion at trillion-parameter scale [23]; we hope to explore whether these issues share a common cause.

We also aim to explore the efficacy of the method on a wider variety of models and datasets and against a more consistent and diverse set of hyperbolic residual connection methods from the literature for these setups. Our initial experiments comparing with Möbius addition and space-like dimension addition on HELM models continue to show the dominance of LAttnRes. We hope to see whether this method generalizes to a broad number of domains and how sensitive it is to growing parameter counts and datasets.

Finally, foundation models on hierarchical data [11, 12] are a natural setting for this work, since hyperbolic geometry helps most where models are deepest and widest, meaning stability is especially important. We view LAttnRes as a starting point for non-Euclidean residual designs that hold up at scale.

Acknowledgments

References

[1] Ahmad Bdeir, Kristian Schwethelm, and Niels Landwehr. 2023. Fully hyperbolic convolutional neural networks for computer vision. *arXiv preprint arXiv:2303.15919* (2023).

[2] Ahmad Bdeir, Kristian Schwethelm, and Niels Landwehr. 2024. Fully Hyperbolic Convolutional Neural Networks for Computer Vision. In *ICLR*.

[3] Ines Chami, Zhitao Ying, Christopher Ré, and Jure Leskovec. 2019. Hyperbolic graph convolutional neural networks. In *NeurIPS*. 4868–4879.

[4] Bike Chen, Wei Peng, Xiaofeng Cao, and Juha Röning. 2023. Hyperbolic uncertainty aware semantic segmentation. *TITS* (2023).

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

[6] Aleksandr Ermolov, Leyla Mirvakhabova, Valentin Khrulkov, Nicu Sebe, and Ivan Oseledets. 2022. Hyperbolic vision transformers: Combining improvements in metric learning. In *CVPR*. 7409–7419.

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

[8] Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, et al. 2024. The Llama 3 Herd of Models. arXiv:2407.21783 [cs.AI] https://arxiv.org/abs/2407.21783

[9] Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Amy Yang, Angela Fan, et al. 2024. The Llama 3 Herd of Models. *arXiv:2407.21783* (2024).

[10] Xiangming Gu, Tianyu Pang, Chao Du, Qian Liu, Fengzhuo Zhang, Cunxiao Du, Ye Wang, and Min Lin. 2024. When Attention Sink Emerges in Language Models: An Empirical View. *arXiv preprint arXiv:2410.10781* (2024).

[11] 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).

[12] Neil He, Jiahong Liu, Buze Zhang, Ngoc Bui, Ali Maatouk, Menglin Yang, Irwin King, Melanie Weber, and Rex Ying. 2025. Position: Beyond Euclidean – Foundation Models Should Embrace Non-Euclidean Geometries. *arXiv preprint arXiv:2504.08896* (2025).

[13] Neil He, Menglin Yang, and Rex Ying. 2025. HyperCore: The Core Framework for Building Hyperbolic Foundation Models with Comprehensive Modules. *arXiv preprint arXiv:2504.08912* (2025).

[14] Neil He, Menglin Yang, and Rex Ying. 2025. Lorentzian Residual Neural Networks. In *KDD*.

[15] Jordan Hoffmann, Sebastian Borgeaud, Arthur Mensch, Elena Buchatskaya, Trevor Cai, Eliza Rutherford, Diego de Las Casas, Lisa Anne Hendricks, Johannes Welbl, Aidan Clark, Tom Hennigan, Eric Noland, Katie Millican, George van den Driessche, Bogdan Damoc, Aurelia Guy, Simon Osindero, Karen Simonyan, Erich Elsen, Jack W. Rae, Oriol Vinyals, and Laurent Sifre. 2022. Training Compute-Optimal Large Language Models. arXiv:2203.15556 [cs.CL] https://arxiv.org/abs/2203.15556

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

[17] Qi Liu, Maximilian Nickel, and Douwe Kiela. 2019. Hyperbolic graph neural networks. In *NeurIPS*. 8230–8241.

[18] Pascal Mettes, Mina Ghadimi Atigh, Martin Keller-Ressel, Jeffrey Gu, and Serena Yeung. 2023. Hyperbolic Deep Learning in Computer Vision: A Survey. *arXiv preprint arXiv:2305.06611* (2023).

[19] Wei Peng, Tuomas Varanka, Abdelrahman Mostafa, Henglin Shi, and Guoying Zhao. 2021. Hyperbolic deep neural networks: A survey. *TPAMI* (2021).

[20] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, Gretchen Krueger, and Ilya Sutskever. 2021. Learning Transferable Visual Models From Natural Language Supervision. *arXiv* (2021). https://arxiv.org/abs/2103.00020

[21] Rik Sarkar. 2011. Low distortion delaunay embedding of trees in hyperbolic plane. In *International Symposium on Graph Drawing*. Springer, 355–366.

[22] Gemma Team. 2024. Gemma: Open Models Based on Gemini Research and Technology. *arXiv preprint arXiv:2403.08295* (2024). https://arxiv.org/abs/2403.08295

[23] Kimi Team. 2025. Kimi K2: Open Agentic Intelligence. *arXiv preprint arXiv:2507.20534* (2025).

[24] Kimi Team, Guangyu Chen, Yu Zhang, Jianlin Su, Weixin Xu, Siyuan Pan, Yaoyu Wang, Yucheng Wang, Guanduo Chen, Bohong Yin, et al. 2026. Attention residuals. *arXiv preprint arXiv:2603.15031* (2026).

[25] Hugo Touvron, Matthieu Cord, and Hervé Jégou. 2022. DeiT III: Revenge of the ViT. In *European Conference on Computer Vision (ECCV)*.

[26] Wikimedia Foundation. [n. d.]. Wikimedia Downloads. https://dumps.wikimedia.org. Accessed: 2026-04-29.

[27] Guangxuan Xiao, Yuandong Tian, Beidi Chen, Song Han, and Mike Lewis. 2024. Efficient Streaming Language Models with Attention Sinks. *arXiv preprint arXiv:2309.17453* (2024).

[28] 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.

[29] Menglin Yang, Min Zhou, Zhihao Li, Jiahong Liu, Lujia Pan, Hui Xiong, and Irwin King. 2022. Hyperbolic graph neural networks: a review of methods and applications. *arXiv preprint arXiv:2202.13852* (2022).

[30] Menglin Yang, Min Zhou, Hui Xiong, and Irwin King. 2022. Hyperbolic Temporal Network Embedding. *TKDE* (2022).

[31] Rowan Zellers, Ari Holtzman, Yonatan Bisk, Ali Farhadi, and Yejin Choi. 2019. HellaSwag: Can a Machine Really Finish Your Sentence? arXiv:1905.07830 http://arxiv.org/abs/1905.07830

[32] Shuangfei Zhai, Tatiana Likhomanenko, Etai Littwin, Dan Busbridge, Jason Ramapuram, Yizhe Zhang, Jiatao Gu, and Joshua Susskind. 2023. Stabilizing Transformer Training by Preventing Attention Entropy Collapse. *arXiv preprint arXiv:2303.06296* (2023).