Hyperbolic ViT on CIFAR-100: progressive ablation

October 13, 2025

Federico Forner

Abstract

I start from a Euclidean ViT-Tiny on CIFAR-100 and progressively "hyperbolicize" its modules (head, positional embeddings, residual, MLP, self-attention). I evaluated Top-1 / Top-5, computational cost (img / s, wall clock time), stability, and learned curvatures. The centered hyperbolic residual (residual around a learnable barycenter) provides the main gain (+4 Top-1 over the Euclidean baseline) at equal parameter budget, at the cost of significantly slower training. I discuss original components (residual and attention), limitations (memory, stability), and when hyperbolic modules are worthwhile.

1. Introduction and Method

Goal. I study how a standard Euclidean ViT-Tiny changes when I progressively introduce hyperbolic components. The motivation is that hyperbolic geometry can better represent hierarchical structure and is flexible: when a Euclidean solution is preferable, the learned curvature can approach zero.

Dataset and setup. I use CIFAR-100 (100 classes), intentionally "flat" so I do not expect a free boost from a hyperbolic head alone. The baseline is a standard ViT-Tiny (12 blocks, 3 heads, embed dim 192, patch size 4), entirely Euclidean. Primary metrics: Top-1/Top-5, the generalization *gap* (Top-1 − Train acc), numerical stability, throughput (img/s) and wall-clock time.

Pipeline. I run a *progressive ablation*: Head → Positional Embeddings → Residual → Linear/MLP → Self-Attention. The order isolates the effect of each block while keeping the training stable. I use the same pre/post clipping margin $t = 0.985$ across all hyperbolic modules. For attention, I use a short fine-tuning schedule (see below).

Compute constraints. Training runs on Kaggle (P100 16GB). For heavy stages (Residual/Linear/Attention), I use gradient accumulation to avoid OOM. Hyperbolic attention is optimized in two phases: 10 epochs *attention-only* starting from the Hyperbolic Linear variant, then 10 epochs with the whole model unfrozen (fine-tuning).

Note. There are many design choices in hyperbolic networks. Here I keep a consistent geometry and introduce original pieces (e.g., learnable centers in residual and attention). The only Euclidean component that I keep by design is LayerNorm (satisfactory hyperbolic variants are still lacking).

Hyperbolic head. Classes are prototypes on the Poincaré ball; logits are $-d_{\mathbb{H}}(x,c)^2/(2\sigma^2)+b$, with learnable σ and a dedicated curvature. $d_{\mathbb{H}}$ denotes the hyperbolic distance.

Hyperbolic positional embeddings. Positions are learned on the manifold and combined with tokens via Möbius addition ($\oplus$) after mapping to the Poincaré ball.

Hyperbolic residual. We introduce a learnable center

$$\begin{aligned} p &= s \otimes x \;\oplus\; (1-s) \otimes f(x), \\ s &= \sigma(\tilde{s}) \in (0,1). \end{aligned}$$

and compute the residual update *around* p:

$$\begin{aligned} x_p &= (-p) \oplus x, \\ y_p &= (-p) \oplus f(x), \\ h_p &= x_p \oplus \big(\gamma \otimes y_p\big), \\ h &= p \oplus h_p. \end{aligned}$$

Thus we do not compose at the origin but around an adaptive barycenter between x and $f(x)$, which can shift toward either operand. This structural change is novel in our setting and yields several extra Top-1 points (at higher compute cost).

Hyperbolic MLP. Each linear layer is replaced by a `HyperbolicLinear` that re-centers the operation on two learnable manifold points p_{in} (domain) and p_{out} (codomain), with curvature c_{hlin} learned via inverse-softplus and the usual pre/post clipping around $\exp_0$ / $\log_0$. Analytically, for input $x \in \mathbb{R}^D$,

$$\begin{aligned} h_x &= \exp_0\big(\operatorname{clip}_{\text{pre}}(x)\big), \\ x_p &= (-p_{\text{in}}) \oplus h_x, \\ z &= \text{mobius\_matvec}(W, x_p) \oplus \exp_0(b), \\ y_h &= p_{\text{out}} \oplus \operatorname{clip}_{\text{post}}(z), \\ y &= \log_0(y_h). \end{aligned}$$

so the affine map happens in the chart centered at $(p_{\text{in}}, p_{\text{out}})$, then I return to Euclidean. The full MLP is

$$x \xrightarrow{\texttt{HLinear}} y_1 \xrightarrow{\text{GELU}} y_2 \xrightarrow{\text{Dropout}} y_3$$
$$\xrightarrow{\texttt{HLinear}} y_4 \xrightarrow{\text{Dropout}} y$$

with GELU applied in Euclidean after each `HLinear`. Recentering $(p_{\text{in}}, p_{\text{out}})$ reduces distortion where curvature is active and provides an *affine* degree of freedom on the manifold (where the map is linearized). In our runs it reached **Top-1 53.92** (vs. 53.10 baseline) with much higher training time, and relatively high train loss, suggesting underfitting due to the short schedule; longer training and tuned hAMP/clipping typically help.

Hyperbolic self-attention. Queries/keys/values are produced by `HyperbolicLinear` layers and mapped to the Poincaré ball (curvature c_{attn} learned via inverse-softplus), with standard pre/post clipping around $\exp_0 / \log_0$ for stability. I augment keys/values with a bank of *shared global centroids* $C \in \mathbb{R}^{H \times M \times H_d}$ (one bank per head), trained in Euclidean and mapped to the ball at forward. The dot-product is replaced by a distance-based score

$$S \;=\; -\frac{d_{\mathbb{H}}(q_h, k_h)^2}{2\,\sigma_h^2}, \qquad W \;=\; \operatorname{softmax}(S),$$

with learnable per-head σ_h. The output is the Riemannian weighted midpoint of the value set

$$o_h \;=\; \operatorname{Midpoint}_{\mathbb{H}}(V_h; W),$$

computed along the token dimension, then returned to Euclidean with $\log_0$ and projected by a final `HyperbolicLinear`. For stability I use a short two-phase schedule (attention-only, then full fine-tuning; details in Results).

2. Results and Discussion

I report the best checkpoints (highest Top-1 or lowest Val loss) along the 100-epoch schedule (for attention I fine-tune for 20 epochs due to compute). For *Train acc* I consider the maximum among values logged approximately every $\sim$100 iterations. Abbreviations: **hball** = curvature used by the hyperbolic head, **pball** = curvature used by the positional embeddings, **Gap** = Top-1 $-$ Train acc.

Residual ablation (four modes). *Centered residual* is the best (**Top-1 57.39**; Top-5 82.99). *Only-residual* (same centered residual while all other blocks remain Euclidean) essentially ties it (**Top-1 57.37**). Among the baselines *without a center*, the *no center (at x)* variant (geodesic step with base point x) reaches **55.37** Top-1 (about -2.0 vs. centered), while *no center (at 0)* (Möbius add/mul at the origin) obtains **53.76** Top-1 (about -3.6 vs. centered and about -1.6 vs. *no center (at x)*). The two no-center variants are markedly faster (4h28m and 4h07m; 332.5 and 367.2 img/s) but less accurate, confirming that recentering on a learnable barycenter is the decisive source of the gain. *This near tie indicates that, for our setup, the gain is driven almost entirely by the residual geometry itself and is largely indifferent to whether the surrounding modules (head/pos/MLP/attn) are Euclidean or hyperbolic.*

What changes analytically. *No center (at x)*: geodesic update at base point x,

$$h_{\text{no-center@}x} \;=\; \exp_x\!\big(\gamma \, \log_x(f(x))\big).$$

No center (at 0): operate at the origin,

$$h_{\text{no-center@0}} \;=\; \log_0\!\Big(\exp_0(x) \;\oplus\; \big(\gamma \otimes \exp_0(f(x))\big)\Big).$$

With center: form the barycenter

$$\begin{aligned} p &= s \otimes x \;\oplus\; (1-s) \otimes f(x), \\ s &= \sigma(\tilde{s}) \in (0,1), \\ x_p &= (-p) \oplus x, \\ y_p &= (-p) \oplus f(x), \\ h_p &= x_p \oplus \big(\gamma \otimes y_p\big), \\ h &= p \oplus h_p. \end{aligned}$$

Here $\oplus$ denotes Möbius addition and $\otimes$ Möbius scalar multiplication. All operations are carried out on the Poincaré ball after mapping features with $\exp_0 / \log_0$, with pre/post clipping for stability. Geometrically, the learnable center p acts as a *moving chart*: we compose neither at the origin nor at x, but around an *adaptive* anchor between x and $f(x)$. This (i) reduces distortion where curvature is active (smaller norms, fewer saturations); (ii) *decouples* the residual direction from the specific base point x; and (iii) effectively adds an *affine* degree of freedom on the manifold (a learnable linearization offset). Empirically, this extra freedom explains the gains: both "centered" and "only-residual" converge to `pball` ≈ 1.9 and `hball` ≈ 0.99 and achieve almost identical Top-1, isolating the centered residual as the key contributor irrespective of whether the remaining modules are Euclidean or hyperbolic. **Center statistics.** On models trained *with* a center, the values of $s = \sigma(\tilde{s})$ across layers are $s_{\min} = 0.3402$, $s_{\max} = 0.7082$, $s_{\text{mean}} = 0.6046$, std $= 0.0757$; the barycenter therefore skews toward x on average ($s > 0.5$), consistent with the improvement.

Compute cost. Throughput drops and wall-clock time grows as more work moves to the manifold; the jump is especially visible from the residual stage onward, and becomes severe with attention.

Hyperbolic ViT on CIFAR-100: progressive ablation

Setting	Ep	Top-1	Top-5	Val Loss	Train Loss	Gap	Total time	imgs/s	pball	hball
Euclidean baseline (reference)	100	53.10	79.04	2.5203	0.1958	42.99	1h07m	1327.8	—	—
Hyperbolic head only	100	50.16	76.46	3.6272	0.1303	45.93	1h15m	1225.3	—	1.54
Hyperbolic positional only	100	45.11	72.35	3.5538	0.3197	37.70	1h19m	1133.3	1.72	1.47
Residual with center	100	**57.39**	82.99	2.6093	0.2028	37.14	8h05m	177.0	1.91	0.99
Only residual (rest Euclid, centered)	100	**57.37**	82.94	2.6113	0.1852	38.72	7h30m	180.0	1.91	0.99
Residual no center (at x)	100	55.37	81.55	2.7884	0.1921	41.51	4h28m	332.5	1.79	0.99
Residual no center (at 0)	100	53.76	80.43	3.0587	0.1496	39.21	4h07m	367.2	—	—
Hyperbolic MLP only	100	53.92	82.23	1.7604	1.2199	21.08	23h44m	60.1	2.48	0.82
Hyperbolic attention only	20	45.76	76.94	2.0386	1.9513	22.99	21h05m	12.4	2.48	0.81

Table 1. Comparison of Euclidean vs. hyperbolic variants. Residual rows include four modes and are ordered by validation Top-1 (desc). **hball** = head curvature; **pball** = positional curvature; **Gap** = $|\text{Top-1} - \text{Train acc}|$.

Memory and stability. Hyperbolic ops are memory-hungry; in early stages I disable autocast for stability. Gradient accumulation avoids OOM at the cost of slower training.

Accuracy and gap. Accuracy initially drops (53.1 $\to$ 48.6 $\to$ 45.9 with head/pos), then rises with the residual to 57.4 (+4.3 over baseline). Heavier modules tend to peak late; longer schedules/tuning would likely help. The gap shrinks once the residual is introduced, indicating improved generalization.

Underfitting signals. Under the short schedule I observe signs of underfitting (especially in MLP/Attention). Accuracy is expected to improve with longer training, tuned clipping margins t and hAMP, and per-block learning-rate scaling; some of these gains may extend to other modules as well.

Curvature. Learned curvatures do not collapse to zero, so the model actually exploits non-Euclidean geometry. As more modules become hyperbolic, I observe a trend where $pball$ tends to increase and $hball$ decreases, consistent with tokens being better arranged on the manifold and the head requiring less curvature.

Attention note. Only 20 epochs of fine-tuning (starting from the Linear variant) due to compute; results should be read with caution. I use shared global centroids across heads/layers to reduce parameters and focus gradients.

3. Conclusions

Structure matters: granting more geometric freedom (without increasing parameter count) can yield non-trivial gains. On a non-hierarchical dataset like CIFAR-100, hyperbolic modules surpassed the Euclidean baseline by over 4 Top-1 points, at the known costs of memory/stability and especially speed. They are most beneficial where hierarchies exist and on targeted parts of the model (in our study, the residual/skip proved particularly effective), rather than hyperbolicizing everything indiscriminately. To the best of our knowledge, no prior work on hyperbolic ViTs adopted a residual centered at an adaptive *barycenter* p (a learnable combination of x and $f(x)$) on which to compose the residual update. This *learnable center* in the residual—understood as a barycenter that *centers* composition on the manifold—was decisive: at fixed capacity, it shifts the optimum towards more expressive and stable combinations, translating into the largest observed Top-1 gain.

References

Dosovitskiy, A., Beyer, L., Kolesnikov, A., Weissenborn, D., Zhai, X., Unterthiner, T., Dehghani, M., Minderer, M., Heigold, G., Gelly, S., Uszkoreit, J., and Houlsby, N. An image is worth 16x16 words: Transformers for image recognition at scale. https://arxiv.org/abs/2010.11929, 2020.

Ermolov, A., Mirvakhabova, L., Khrulkov, V., Sebe, N., and Oseledets, I. Hyperbolic vision transformers: Combining improvements in metric learning. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2022.

Ganea, O.-E., Bécigneul, G., and Hofmann, T. Hyperbolic neural networks. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2018.

Google Research. Vision transformer: Official repository. https://github.com/google-research/vision_transformer, 2020.

Gülçehre, Ç., Denil, M., Malinowski, M., Razavi, A., Pascanu, R., Hadsell, R., and Lillicrap, T. Hyperbolic attention networks. In *International Conference on Learning Representations (ICLR)*, 2019.

Kochurov, M. and contributors. Geoopt documentation: Stereographic math (poincaré ball) and Möbius operations. https://geoopt.readthedocs.io/en/latest/_modules/geoopt/manifolds/stereographic/math.html, 2023.

Kochurov, M., Karimov, R., and Kozlukov, S. Geoopt: Riemannian optimization in pytorch. https://arxiv.org/abs/2005.02819, 2020.

Krizhevsky, A. Learning multiple layers of features from tiny images. Technical report, University of Toronto, 2009.

Nickel, M. and Kiela, D. Poincaré embeddings for learning hierarchical representations. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2017.

PyTorch Contributors. Pytorch documentation: torch.nn.softplus. https://docs.pytorch.org/docs/stable/generated/torch.nn.Softplus.html, 2025.

Snell, J., Swersky, K., and Zemel, R. Prototypical networks for few-shot learning. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2017.

Wightman, R. Pytorch image models. https://github.com/rwightman/pytorch-image-models, 2019. DOI: 10.5281/zenodo.4414861.