HEXFORMER: HYPERBOLIC VISION TRANSFORMER WITH EXPONENTIAL MAP AGGREGATION

Anonymous authors
Paper under double-blind review

ABSTRACT

Data across modalities such as images, text, and graphs often contains hierarchical and relational structures, which are challenging to model within Euclidean geometry. Hyperbolic geometry provides a natural framework for representing such structures. Building on this property, this work introduces HexFormer, a hyperbolic vision transformer for image classification that incorporates exponential map aggregation within its attention mechanism. Two designs are explored: a hyperbolic ViT (HexFormer) and a hybrid variant (HexFormer-Hybrid) that combines a hyperbolic encoder with an Euclidean linear classification head. HexFormer incorporates a novel attention mechanism based on exponential map aggregation, which yields more accurate and stable aggregated representations than standard centroid based averaging, showing that simpler approaches retain competitive merit. Experiments across multiple datasets demonstrate consistent performance improvements over Euclidean baselines and prior hyperbolic ViTs, with the hybrid variant achieving the strongest overall results. Additionally, this study provides an analysis of gradient stability in hyperbolic transformers. The results reveal that hyperbolic models exhibit more stable gradients and reduced sensitivity to warmup strategies compared to Euclidean architectures, highlighting their robustness and efficiency in training. Overall, these findings indicate that hyperbolic geometry can enhance vision transformer architectures by improving gradient stability and accuracy. In addition, relatively simple mechanisms such as exponential map aggregation can provide strong practical benefits. Code is available at: [link hidden for blind review]

1 INTRODUCTION

Hyperbolic geometry has emerged as a powerful tool in deep learning due to its ability to model hierarchical and relational structures more effectively than Euclidean spaces (Nickel & Kiela, 2017; Bdeir et al., 2023). Vision Transformers (ViTs) form the backbone of many state-of-the-art vision models (Siméoni et al., 2025; Ravi et al., 2024). However, the reliance on Euclidean representations suggests they may struggle to capture complex, hierarchical structures in ways that hyperbolic spaces naturally can (Chamberlain et al., 2017).

Recent works have attempted to improve the performance by integrating hyperbolic geometry into transformer architectures. Early efforts mapped ViT outputs into hyperbolic space for metric learning (Ermolov et al., 2022), while more recent approaches such as HVT (Fein-Ashley et al., 2024) and LViT from the HyperCore framework (He et al., 2025b) incorporated hyperbolic operations into transformer modules. Although these methods reported improvements, they either restricted hyperbolic integration to limited components or did not investigate training dynamics in detail.

This work introduces a hyperbolic Vision Transformer based on the Lorentz model for image classification. The proposed approach consistently outperforms Euclidean baselines across multiple datasets, activation functions, and model scales, while also surpassing prior hyperbolic ViTs such as HVT (Fein-Ashley et al., 2024) and LViT (He et al., 2025b). Beyond accuracy gains, hyperbolic models also exhibit more stable training dynamics, reducing sensitivity to warmup schedules.

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

The main contributions are as follows:

- **Hyperbolic ViT with novel attention (HexFormer)**: A vision transformer architecture for image classification, entirely formulated in the Lorentz model of hyperbolic space, is developed. Its attention pairs scoring by the squared Lorentzian distance with aggregation through the exponential map: a deliberately simple mechanism that has not been used as the aggregation step of hyperbolic attention before, and that proves more stable than centroid/midpoint based averaging despite exponential and logarithmic maps having been argued against on precisely those grounds. The resulting model outperforms both Euclidean ViTs and prior hyperbolic ViTs on image classification. This accuracy gain comes from the combination of components rather than from the aggregation step alone, while exponential-map aggregation contributes numerical stability.
- **Hybrid encoder-classifier design (HexFormer-Hybrid)**: A model that combines a hyperbolic encoder with an Euclidean linear classification head, using each geometry where it is most effective: negative curvature for learning hierarchical features, and a standard Euclidean classifier for the final decision. This further improves performance over both fully hyperbolic and fully Euclidean models, whereas the converse arrangement, a Euclidean encoder with a hyperbolic head, is not competitive.
- **Analysis of gradient stability**: This work provides the first analysis of gradient stability in hyperbolic transformers. The study shows that hyperbolic ViTs maintain more concentrated gradient magnitudes and are more robust to warmup strategies, and that they degrade more gracefully than Euclidean models under input corruption.

Extensive experiments demonstrate that HexFormer and HexFormer-Hybrid achieve consistent improvements over Euclidean ViTs, with the hybrid variant delivering the strongest results overall. These findings indicate that hyperbolic geometry provides a principled approach to enhancing transformer architectures in terms of both representational capacity and training stability.

2 Related Work

2.1 Hyperbolic Geometry in Deep Learning

Hyperbolic embeddings have been employed extensively to encode hierarchical relationships in data (Nickel & Kiela, 2017; Bdeir et al., 2023; Chen et al., 2021). Early work projected taxonomies into the Poincaré ball (Nickel & Kiela, 2017). However, previous works have shown that the Poincaré ball suffers from training instabilities (Mishne et al., 2024). In computer vision, convolutional neural networks have been adapted to hyperbolic Lorentz space, yielding performance gains in classification and segmentation (Bdeir et al., 2023). Hyperbolic Lorentz representations have also benefited graph learning and fully connected architectures, particularly when modeling relational data (Yang et al., 2024).

2.2 Hyperbolic Vision Transformers

Vision transformers (Dosovitskiy et al., 2020) have shown state-of-the-art performance for image classification tasks. Nevertheless, they require a lot of data and tend to not perform so well with smaller datasets (Pandey et al., 2023). Due to this, a natural extension is to improve the ViT's architecture to reduce the amount of data needed by changing the representation space. Several approaches have explored integrating hyperbolic geometry into Vision Transformers, whose design differences are summarized in Table 4 (Appendix A.1): Hyp-ViT projects a vanilla ViT's output into the Poincaré ball and employs a pairwise cross-entropy loss in hyperbolic distance for metric learning, outperforming Euclidean baselines in retrieval tasks (Ermolov et al., 2022). HVT implements a Vision Transformer on ImageNet with self-attention adapted via Möbius transformations and hyperbolic distance within the Poincaré ball framework (Fein-Ashley et al., 2024). HyperCore introduces modules for constructing hyperbolic foundation models in both Lorentz and Poincaré geometries. It includes a hyperbolic ViT in Lorentz space (LViT), demonstrating improvements over Euclidean ViTs in image classification (He et al., 2025b). However, these lifts are only observed when finetuning from a bigger dataset, but not when training only on the downstream task. HypFormer (Yang et al., 2024) is a fully hyperbolic Lorentz transformer with linear attention, developed for graph-structured data rather than vision.

2.3 Aggregation and Training Dynamics in Hyperbolic Models

Most existing hyperbolic ViT variants use centroid based aggregation in attention mechanisms (Chen et al., 2021; Fein-Ashley et al., 2024; He et al., 2025b). These can introduce distortion in curved spaces when processing big values (Mishne et al., 2024). Exponential-map aggregation has not been explored in combination with hyperbolic attention scores. Logarithmic and exponential maps have been used in hyperbolic models, but where they appear in attention it is to move the whole computation into the tangent space rather than to serve as the aggregation step of an otherwise hyperbolic attention: HVT (Fein-Ashley et al., 2024) maps queries, keys, and values into tangent space and performs the entire attention computation there, which mitigates the benefits from the exponentially growing hyperbolic distance metric. Some work argues against these maps altogether. HypFormer (Yang et al., 2024) avoids exponential and logarithmic maps on the grounds that they introduce mapping error, destabilize training and add computational cost, and aggregates with centroids instead. The measurements reported here point the other way: exponential-map aggregation is the more numerically stable of the two (Appendix A.4), at a small additional runtime cost (Appendix A.5). HexFormer calculates the scores based on the hyperbolic distance and uses them to aggregate the points on the tangent plane, retaining the distance metric in the attention weights while avoiding the division by a Lorentzian norm that makes the centroid numerically fragile. Furthermore, the training behavior of hyperbolic transformers (specifically gradient stability and sensitivity to warmup) remains underexamined.

Prior work on hyperbolic models have demonstrated the potential of integrating both Euclidean and non-Euclidean geometry for improving performance (Ermolov et al., 2022; Bdeir et al., 2023). This motivates exploring hybrid approaches that leverage the strengths of both geometries.

3 Preliminaries

Hyperbolic geometry is a non-Euclidean geometry with constant negative curvature $K < 0$. Several equivalent models can represent hyperbolic space, such as the Poincaré ball model (Ganea et al., 2018), the Poincaré half-plane model (Tifrea et al., 2018), the Klein model (Gulcehre et al., 2018), and the Lorentz (hyperboloid) model (Nickel & Kiela, 2018). These models are isometric to one another, meaning transformations between them preserve hyperbolic distances (Ramsay & Richtmyer, 1995). The Lorentz model is frequently adopted in hyperbolic deep learning due to its numerical stability and the closed-form expressions of its distance function, exponential map, and logarithmic map (Chen et al., 2021; Yang et al., 2024).

3.1 The Lorentz Model

The n-dimensional Lorentz model is the Riemannian manifold $\mathbb{L}_K^n = (\mathcal{L}^n, \mathfrak{g}_x^K)$. It is defined in the $(n+1)$-dimensional Minkowski space based on the Riemannian metric tensor, using $\mathrm{diag}(-1, 1, \ldots, 1)$. The hyperboloid is then given by

$$\mathcal{L}^n \coloneqq \{x \in \mathbb{R}^{n+1} \mid \langle x, x\rangle_{\mathcal{L}} = 1/K,\ x_t > 0\},$$

where $K \in \mathbb{R}^-$ is the negative curvature and the Lorentzian inner product is given as

$$\langle x, y\rangle_{\mathcal{L}} \coloneqq -x_t y_t + x_s^\top y_s = x^\top \mathrm{diag}(-1, 1, \ldots, 1)\, y.$$

which corresponds to the upper sheet of the two-sheeted hyperboloid. The first coordinate x_t is often referred to as the time-like component, while the remaining n coordinates represent spatial dimensions x_s, reflecting the connection to the geometry of special relativity. Each point in $\mathbb{L}_K^n$ has the form $x = \left[\begin{smallmatrix} x_t \\ x_s \end{smallmatrix}\right], x \in \mathbb{R}^{n+1}, x_t \in \mathbb{R}, x_s \in \mathbb{R}^n$ and $x_t = \sqrt{||x_s||^2 - 1/K}$ (Chen et al., 2021; Bdeir et al., 2023).

3.2 Tangent Space

At each point $x \in \mathbb{L}_K^n$, the tangent space is the Euclidean subspace orthogonal to x under the Lorentzian inner product:

$$\mathcal{T}_x\mathbb{L}_K^n \coloneqq \{y \in \mathbb{R}^{n+1} \mid \langle y, x\rangle_{\mathcal{L}} = 0\}.$$

Although the ambient space is Minkowski, each tangent space is Euclidean. This property allows standard Euclidean operations to be performed in $\mathcal{T}_x\mathbb{L}_K^n$ before mapping results back to the manifold via the exponential map. The tangent space at the origin is denoted as $\mathcal{T}_0\mathbb{L}_K^n$.(Chen et al., 2021).

3.3 Exponential and Logarithmic Maps

The exponential and logarithmic maps connect the hyperbolic manifold $\mathbb{L}_K^n$ with its tangent spaces $\mathcal{T}_x\mathbb{L}_K^n$. For $x \in \mathbb{L}_K^n$ and $z \in \mathcal{T}_x\mathbb{L}^n$, the exponential map $\exp_x^K(z) : \mathcal{T}_x\mathbb{L}_K^n \to \mathbb{L}_K^n$, maps tangent vectors to hyperbolic spaces:

$$\exp_x^K(z) = \cosh(\alpha)\, x + \sinh(\alpha)\, \frac{z}{\alpha}, \quad \alpha = \sqrt{-K}\, \|z\|_{\mathcal{L}}, \quad \|z\|_{\mathcal{L}} = \sqrt{\langle z, z \rangle_{\mathcal{L}}}.$$

Conversely, for $y \in \mathbb{L}_K^n$, the logarithmic map $\log_x^K(y) : \mathbb{L}_K^n \to \mathcal{T}_x\mathbb{L}_K^n$, is the inverse of the exponential map that gives the tangent vector at x pointing toward y:

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

Where $\log_x^K(\exp_x^K(z)) = z$. These operators provide a consistent mechanism to switch between the curved geometry of $\mathbb{L}_K^n$ and its Euclidean tangent approximations, and are essential for optimization and representation learning in hyperbolic neural networks (Chen et al., 2021; Yang et al., 2024).

4 Methodology

This section describes the Hyperbolic Vision Transformer (HexFormer) variants and their building blocks, and then presents the hyperbolic attention mechanism with a comparison between centroid aggregation and exponential-map aggregation. Implementation choices that follow prior work are explicitly identified and modifications are highlighted. The Lorentz model notation and basic manifold maps follow the preliminaries in Section 3.

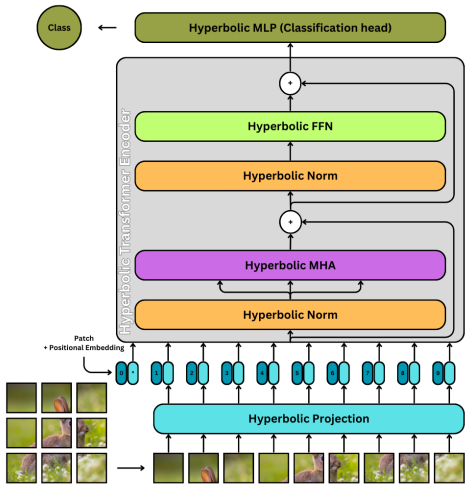


Figure 1: Overview of HexFormer architecture. Hyperbolic embedding, transformer encoder, and classification head operate on the Lorentz manifold. Residual connections are applied to space-like components only.

4.1 Model Variants and Component Overview

Three model variants are considered:

Euclidean ViT: a conventional Vision Transformer fully operating in Euclidean space. The main implementation change compared to the original formulation of ViTs (Dosovitskiy et al., 2020) is the adoption of pre-layer normalization instead of post-layer normalization (Radford et al., 2019).

HexFormer (hyperbolic ViT): a transformer whose patch embeddings, encoder layers (attention, MLP), normalization, and classification head are defined on the Lorentz, as visually presented in Figure 1.

HexFormer-Hybrid: a hybrid design that uses a Lorentz encoder for feature extraction and an Euclidean linear classifier as the final head.

Below, each major component is described, highlighting which parts are drawn from prior work, how they were combined, and which modifications were introduced to form the final HexFormer design.

Hyperbolic fully-connected layers (LorentzFC): The feedforward network (FFN) extends the formulation of HyboNet (Chen et al., 2021), who introduced Lorentz linear layers for hyperbolic neural networks. In their design, the time-like component is computed as

$$y_0 = \lambda\, \sigma(\mathbf{v}^\top \mathbf{x} + b) + \epsilon,$$

where σ is a sigmoid, λ controls the range, b is a learnable bias, and $\epsilon > \sqrt{1/K}$ ensures numerical validity. The space-like components are then rescaled such that the Lorentz norm constraint

$$\|\mathbf{y}_{1:n}\|^2 - y_0^2 = 1/K$$

is satisfied, placing outputs on the Lorentz manifold.

The LorentzFC implementation in HexFormer builds on the core formulation of Chen et al. (2021) and incorporates later adaptations, including multi-head support and optional normalization from HCNN (Bdeir et al., 2023) and curvature-aware time offsets from LResNet (He et al., 2025a). Additional refinements introduced in this work include a unified formulation that integrates multi-head processing, normalization, and curvature-aware offsets into a single flexible module, as well as broadcasting of the time component across heads to ensure Lorentz constraints are satisfied, maintaining Lorentzian geometry while improving stability and efficiency in transformer architectures.

Patch embeddings and Lorentz batch normalization: Following the approach of HCNN (Bdeir et al., 2023), image patches are first projected into the ambient Minkowski space and then transformed using a LorentzFC layer. Positional embeddings are added to the space-like components, and a classification token, initialized directly on the manifold, is prepended to the sequence. The time-like component is then computed according to the Lorentz norm constraint (Section 3), producing the complete hyperbolic embedding. Lorentz batch normalization, taken from HCNN (Bdeir et al., 2023), is applied in each Transformer block to ensure the normalized features remain consistent with the Lorentz geometry.

Classification Head: For classification, the Lorentz MLR formulation of HCNN (Bdeir et al., 2023) is employed, which generalizes multinomial logistic regression to the Lorentz manifold by measuring the hyperbolic distance of embeddings to class-defining hyperplanes. This approach naturally respects the geometry of the hyperbolic space and provides logits suitable for standard cross-entropy training. For HexFormer-Hybrid, an Euclidean linear head (standard classifier) is applied to the flattened Euclidean projection of the CLS token, allowing the model to leverage both hyperbolic and Euclidean representations for classification.

4.2 Hyperbolic attention and aggregation

The attention mechanism is the main point of departure. The attention score computation uses Lorentzian geometry and the squared Lorentzian distance between query and key embeddings. Aggregation of values is performed by exponential map aggregation rather than by computing centroids directly on the manifold. The architecture is shown in Figure 2.

Score computation: Rather than computing dot products in Euclidean space, the hyperbolic attention mechanism computes scores based on the squared Lorentzian distance (Law et al., 2019), which is defined as:

$$d^2(x, y) = -2K - 2\langle x, y\rangle_{\mathcal{L}}, \tag{1}$$

where K is a curvature-related constant and $\langle\cdot,\cdot\rangle_{\mathcal{L}}$ denotes the Lorentzian inner product. Queries and keys are generated by LorentzFC layers and live on $\mathbb{L}_K^n$. The pairwise compatibility is computed from the Lorentzian inner product $\langle\cdot,\cdot\rangle_{\mathcal{L}}$ and the geodesic distance. With curvature parameter $K < 0$ the (squared) geodesic distance may be expressed via the Lorentzian product. Scores are derived from negative squared distance and scaled. These distances, after appropriate scaling with a scale and a learnable temperature parameter, are passed through the softmax function to obtain the attention weights:

$$\text{score} = \text{softmax}\Big(\frac{-d^2(Q, K)}{\sqrt{d_k}} \cdot \tau\Big), \tag{2}$$

where τ is the temperature parameter. This temperature was also added to the Euclidean Transformer to ensure a fair comparison. Typically, the scale is fixed as $\sqrt{d_k}$, but by introducing an additional learnable scale parameter, referred to as temperature, the results improved.

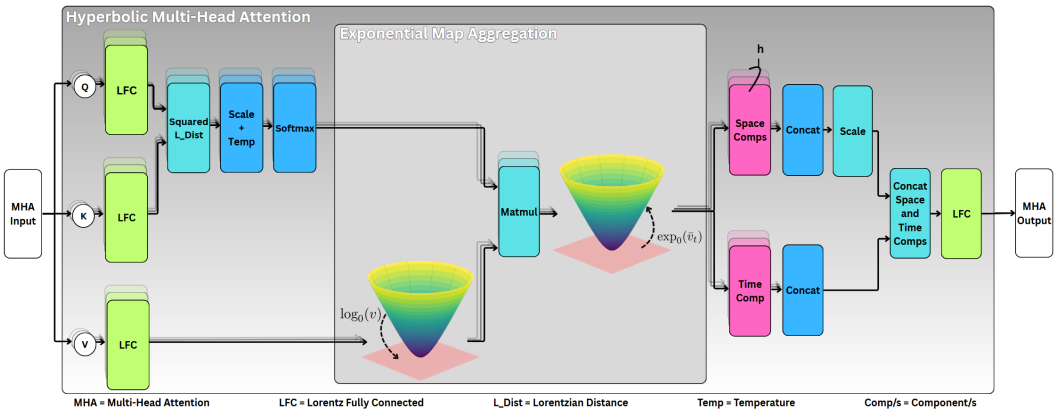


Figure 2: HexFormer multi-head attention. Queries, keys, and values are projected via Lorentz fully connected layers. Scores are computed using Lorentzian distances, aggregated in tangent space via exponential map, and recombined after scaling the space components (to normalize magnitudes and maintain stability in the Lorentz model), then passed as output through a Lorentz FC layer.

Exponential aggregation: In standard attention mechanisms, the softmax scores are Euclidean scalars that weight the value vectors before aggregation. Directly applying these weights in hyperbolic space is not geometrically consistent. To address this, an exponential aggregation scheme is adopted. First, each values vector V is mapped from the Lorentz manifold $\mathbb{L}_K^n$ to the tangent space at the origin using the logarithmic map,

$$\tilde{V} = \log_0^K(V) \tag{3}$$

The softmax score is then applied as a scalar weight in the tangent space by performing a dot product,

$$u = \sum_i \alpha_i \tilde{V}_i \tag{4}$$

where $\alpha \in \mathbb{R}$ and $u \in \mathcal{T}_0\mathbb{L}_K^n$. Finally, the result is mapped back to the manifold via the exponential map, ensuring that the aggregation remains Lorentz-consistent,

$$h_{agg} = \exp_0^K(u) \tag{5}$$

where $h_{agg} \in \mathbb{L}_K^n$. This procedure preserves the hyperbolic geometry while allowing Euclidean attention weights to modulate the contribution of value vectors in a geometrically valid manner. By operating in the tangent space for scalar multiplications and returning to the manifold through the exponential map, the method avoids distortion, leading to more stable training and better alignment of attention outputs with the manifold's curvature.

5 Experiments

5.1 Training Strategy

All models were trained under a unified training pipeline, differing only in optimizer choice, learning rate, and weight decay to respect the underlying geometry. Standard data augmentations, weight initialization, loss functions, and learning rate scheduling were applied consistently across experiments. The full training strategy, together with hyperparameter settings and implementation details, are provided in Appendix A.2. The Euclidean ViT serves as a matched control rather than a competitive recipe. The aim is to isolate the effect of the representation space, so all remaining factors are held fixed. Adopting a more recent vision transformer design or training recipe would raise absolute accuracy, but such refinements are largely independent of the representation space and apply equally to either geometry, so introducing them would vary several factors at once and confound the comparison. Because the two models differ only in geometry, the gap between them is attributable to that choice, and the same substitution is in principle available to any architecture built on a Euclidean ViT encoder.

Table 1: Accuracy comparison on CIFAR-10, CIFAR-100, and Tiny-ImageNet. The upper block gives the values reported by He et al. (2025b), the lower block gives models trained from scratch under the pipeline of this work. The hyperbolicity values (δ_{rel}) are taken from HCNN (Bdeir et al., 2023).

Model	**CIFAR-10** ($\delta_{rel} = 0.26$)	**CIFAR-100** ($\delta_{rel} = 0.23$)	**Tiny-ImageNet** ($\delta_{rel} = 0.20$)
As reported by He et al. (2025b)			
HVT† (Fein-Ashley et al., 2024)	61.44	42.77	40.12
LViT (He et al., 2025b)	85.02	69.11	53.01
Trained from scratch in this pipeline			
Euclidean ViT	$93.38_{\pm 0.118}$	$74.64_{\pm 0.408}$	$60.74_{\pm 0.356}$
LViT (He et al., 2025b)	$91.79_{\pm 0.299}$	$72.35_{\pm 0.792}$	$53.29_{\pm 1.046}$
HexFormer (ours)	$\underline{93.42}_{\pm 0.183}$	$\underline{75.65}_{\pm 0.434}$	$\underline{60.87}_{\pm 0.635}$
HexFormer-Hybrid (ours)	$\mathbf{93.53}_{\pm 0.287}$	$\mathbf{75.85}_{\pm 0.337}$	$\mathbf{62.93}_{\pm 0.447}$

†: HVT was not retrained in this pipeline. Its reported accuracies fall well below those of LViT in the same source, and below the Euclidean baseline, on all three datasets, so a matched-pipeline run would not affect the comparison, its released implementation also departs from the description in its paper (Appendix A.1).

5.2 Comparison of Hyperbolic Vision Transformers

Table 1 presents the accuracy comparison of various Vision Transformer (ViT) based models on CIFAR-10, CIFAR-100, and Tiny-ImageNet. HexFormer consistently outperforms the Euclidean ViT, as well as HVT (Fein-Ashley et al., 2024) and LViT (He et al., 2025b), across all datasets. The HexFormer-Hybrid variant achieves the highest accuracy overall, surpassing all other models.

All models in the lower block of Table 1 were trained from scratch under the same pipeline, differing only in the optimizer and learning-rate settings each geometry requires (Appendix A.2). LViT was retrained under this pipeline as well, where it exceeds its own published accuracy yet remains below both the Euclidean baseline and HexFormer on all three datasets. The consistent margin over these baselines highlights the effectiveness of HexFormer and its hybrid variant.

5.3 Warmup Dependency and Gradient Stability

During preliminary experiments, HexFormer and HexFormer-Hybrid consistently outperformed their Euclidean counterparts in the early epochs of training, before performance converged (Figure 5, Appendix A.3.4). They also reach substantially higher accuracy when training is stopped early without warmup (Appendix A.3.5, Table 14), suggesting that hyperbolic representations accelerate the initial learning phase. To further investigate this accelerated early training and its relation to warmup dependency, experiments were conducted with and without warmup across datasets and architectures. As reported in Table 2, HexFormer and HexFormer-Hybrid show minimal performance differences between warmup and no-warmup conditions, while Euclidean ViTs exhibit a more pronounced dependency on warmup, with larger performance gaps when tested on CIFAR-10, CIFAR-100, and Tiny-ImageNet. This trend holds across different ViT scales (Tiny, Small, and Base) as well, as detailed in Table 8.

To examine whether this reflects gradient behavior, absolute gradient histograms were plotted in the same manner as in RAdam (Liu et al., 2019), where gradient instability was used to motivate the necessity of warmup. As shown in Figure 3, gradients in HexFormer remain more stable than those in Euclidean ViTs when tested without warmup, where Euclidean ViT models exhibit higher distortion in their gradient distributions, especially during the early updates. HexFormer-Hybrid demonstrates similar stability, with additional comparisons provided in Appendix A.3.4. This stability may arise from the geometric properties of hyperbolic space: the negative curvature tends to spread representations and constrain the scale of activations, helping to prevent extreme values in the forward pass and, consequently, extreme gradients during backpropagation, independent of the choice of optimizer (Table 10 in Appendix A.3) or aggregation method (Table 3).

Table 2: Accuracy comparison between HexFormer, HexFormer-Hybrid and Euclidean ViT on CIFAR-10, CIFAR-100, and Tiny-ImageNet under warmup and no warmup.

Model	CIFAR-10		CIFAR-100		Tiny-ImageNet	
	Warmup = 0	Warmup = 10	Warmup = 0	Warmup = 10	Warmup = 0	Warmup = 10
Euclidean ViT	$91.75_{\pm 0.249}$	$93.38_{\pm 0.118}$	$71.86_{\pm 0.255}$	$74.64_{\pm 0.408}$	$59.46_{\pm 0.381}$	$60.74_{\pm 0.356}$
HexFormer	$\underline{93.55}_{\pm 0.424}$	$\underline{93.42}_{\pm 0.183}$	$\underline{75.06}_{\pm 0.515}$	$\underline{75.65}_{\pm 0.434}$	$\underline{60.12}_{\pm 0.275}$	$\underline{60.87}_{\pm 0.635}$
HexFormer-Hybrid	$\mathbf{93.91}_{\pm 0.211}$	$\mathbf{93.53}_{\pm 0.287}$	$\mathbf{75.44}_{\pm 0.623}$	$\mathbf{75.85}_{\pm 0.337}$	$\mathbf{62.17}_{\pm 0.295}$	$\mathbf{62.93}_{\pm 0.447}$

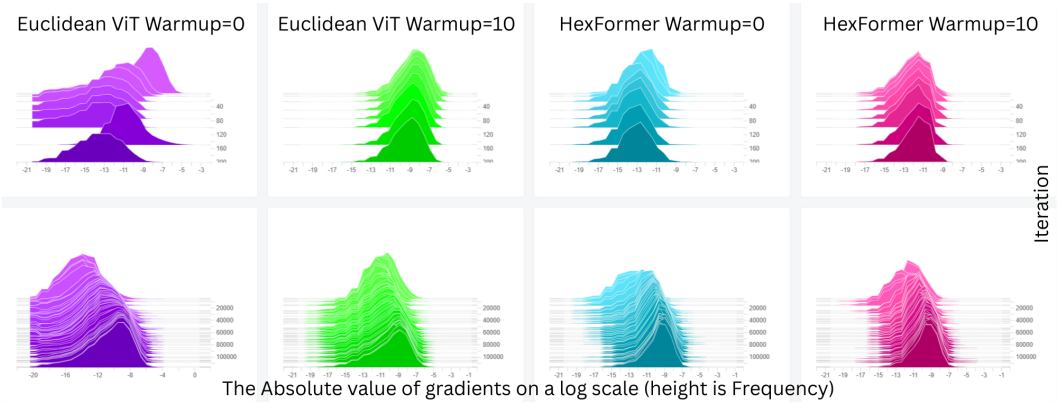


Figure 3: The histogram of the absolute value of gradients on a log scale during the training of ViT-Small on CIFAR-100 comparing HexFormer vs. Euclidean ViT with and without warmup (offset mode). Top row shows close-up for first 200 iterations.

This qualitative picture is confirmed quantitatively, and the full training run tells a stronger story than the early training window alone: measuring the standard deviation of $\log|g|$ across matched training snapshots for ViT-Small on CIFAR-100, both hyperbolic variants remain less dispersed than the Euclidean baseline over the entire run, in both the warmup and no-warmup regimes (the margin is positive at every matched snapshot). Without warmup, the full-run margin in this standard deviation over the Euclidean baseline is $+1.109$ for HexFormer and $+0.547$ for HexFormer-Hybrid. Removing warmup separately affects the validation accuracy of the Euclidean model considerably more than that of either hyperbolic variant. The same ordering in standard deviation holds with warmup ($+0.572$ and $+0.385$), and the gap widens rather than closes as training proceeds. The one exception is the first ten epochs under warmup, where the three models are briefly indistinguishable in gradient dispersion and the ordering even reverses momentarily. Read in isolation, that window would suggest the advantage disappears once warmup is applied, a conclusion the rest of the run does not support. Figure 11 plots this trend over training for both warmup conditions, Tables 12 and 13 give the per-window values, and the matched-pair methodology is described in Appendix A.3.4.

5.4 Aggregation Strategies

Table 3 compares the performance of HexFormer and HexFormer-Hybrid on CIFAR-100 using the centroid aggregation versus the ExpAgg (exponential map aggregation) method under warmup and no warmup (with both methods tuned equally to ensure a fair comparison). The accuracy differences between centroid and ExpAgg are relatively small, ExpAgg being slightly ahead for both HexFormer and HexFormer-Hybrid. The accuracy of HexFormer therefore rests on the combination of hyperbolic components described in Section 4 rather than on the aggregation step, what ExpAgg contributes is numerical stability. Centroid-based aggregation can occasionally produce NaN or infinite values during training, particularly when using the AdamW (Loshchilov, 2017) optimizer, causing the model to fail. For instance, in identical training runs across five seeds, four out of five runs with centroid aggregation encountered NaNs while training, whereas ExpAgg successfully completed training for all five seeds (Figure 14). Appendix A.4 provides a detailed explanation of why HexFormer with centroid aggregation is prone to such instabilities and produces those NaN/inf

Table 3: Accuracy comparison on CIFAR-100 under warmup and no warmup comparing centroid vs ExpAgg. Averaged over 10 seeds.

Model	Warmup = 0	Warmup = 10
HexFormer (Centroid)	$74.63_{\pm 0.589}$	$75.38_{\pm 0.351}$
HexFormer (ExpAgg)	$\mathbf{75.06}_{\pm 0.515}$	$\mathbf{75.65}_{\pm 0.434}$
HexFormer-Hybrid (Centroid)	$75.31_{\pm 0.329}$	$75.73_{\pm 0.357}$
HexFormer-Hybrid (ExpAgg)	$\mathbf{75.44}_{\pm 0.623}$	$\mathbf{75.85}_{\pm 0.337}$

values. Additional experiments comparing different optimizers are reported in Appendix A.3, with full results summarized in Table 10. Overall, ExpAgg ensures more reliable and robust training in practice, making it the preferred aggregation strategy for hyperbolic transformer architectures.

5.5 Other Experiments

Appendix A.3 reports further experiments, summarized here. Across model scales, HexFormer and HexFormer-Hybrid outperform the Euclidean baseline at Tiny, Small, and Base on CIFAR-100, and the margin widens with model size (Table 8). Larger models overfit more on this dataset, and the resulting gap between training and validation accuracy is substantially wider for the Euclidean ViT than for HexFormer, to the point where the Euclidean baseline loses accuracy as it is scaled up while the hyperbolic variants gain (Appendix A.3.1, Figure 4). The advantage is also robust to architectural and optimization choices (Tables 9 and 10, Figures 6–10). Riemannian AdamW performs best, but stable gradients are also observed with standard AdamW, indicating that the stability arises from the embedding geometry rather than from the optimizer.

The gains are, however, specific to hierarchical data. On MNIST, Fashion-MNIST, and SVHN, which lack meaningful class hierarchy, all three models perform comparably (Section A.3.3, Table 11). Under common image corruptions the hyperbolic variants degrade more gracefully than the Euclidean baseline, with the margin growing with corruption severity rather than being uniform (Appendix A.3.6). Both variants also converge faster, reaching competitive accuracy in fewer epochs without warmup (Appendix A.3.5, Table 14), at the cost of higher runtime and memory (Appendix A.5, Tables 19–22).

The hybrid design is the strongest variant throughout. The converse arrangement, a Euclidean encoder with a hyperbolic classification head, was also tested and is not competitive, indicating that the benefit comes from hyperbolic feature extraction rather than from hyperbolic classification.

6 Conclusion

HexFormer introduces a vision transformer for image classification with Lorentzian hyperbolic attention and exponential map aggregation, along with a hybrid design that combines a hyperbolic encoder and Euclidean classification. Across CIFAR-10, CIFAR-100, and Tiny-ImageNet, both variants consistently outperform Euclidean ViTs, with the hybrid model delivering the best overall performance. The accuracy gains come from the combination of hyperbolic components rather than from the aggregation step, while exponential aggregation proves to be more stable than centroid-based methods, avoiding divergence during training. The mechanism is deliberately simple, and prior work has argued against exponential and logarithmic maps on stability and cost grounds. The measurements reported here point the other way when the maps are confined to the aggregation step. More broadly, hyperbolic models exhibit smoother gradient dynamics than Euclidean transformers, reducing sensitivity to warmup schedules. HexFormer and HexFormer-Hybrid surpass both Euclidean baselines and prior hyperbolic vision transformers, establishing state-of-the-art results among hyperbolic models for image classification. These findings show that hyperbolic geometry can enhance both representation quality and optimization stability in vision transformers.

AI USE STATEMENT

In this work, we used generative AI tools for the retrieval and discovery of related work, in order to ensure that relevant literature was not overlooked. We have not used generative AI tools for research ideation, experimental design, the implementation of the proposed method, or the analysis and interpretation of results; all technical content, analyses, experiments, and conclusions presented in this paper were produced independently by the authors. Additionally, we used generative AI tools for writing assistance, specifically for drafting, rephrasing, and polishing text to improve clarity and readability, and for formatting assistance in structuring tables and templates in a consistent format. We have reviewed all AI-assisted work. All literature surfaced through AI-assisted search was independently located, read, and verified by the authors before being cited, and all AI-assisted text was checked by the authors against the underlying experimental results to confirm that it describes them accurately. We take responsibility for the final content of this work, including text, claims or artifacts produced with the aid of generative AI.

REPRODUCIBILITY STATEMENT

The source code for HexFormer paper is included as a zip in the supplementary material accompanying this submission. A link to a public repository will be provided once the paper is no longer under blind review. The full training strategy is given in Appendix A.2, which specifies the data augmentation pipeline, weight initialization, loss function, learning rate schedule, and optimizer, and reports every hyperparameter used: Table 5 lists the architectural specification of each ViT variant, Table 6 the training hyperparameters for the Tiny models, and Table 7 the learning rate and weight decay for the remaining variants and datasets, alongside the tuning procedure and the fixed curvature used throughout. The number of seeds averaged for each reported result is stated in the caption of the corresponding table. The protocols for the additional analyses are documented in the same way: Appendix A.3.6 describes the corruption benchmark and its evaluation procedure, and Appendix A.3.4 describes how the gradient-stability metric is computed from the logged histograms and how significance is assessed. All experiments use publicly available datasets (CIFAR-10, CIFAR-100, CIFAR-100-C, Tiny-ImageNet, MNIST, Fashion-MNIST, and SVHN) and all models are trained from scratch, so no pretrained checkpoints are needed to reproduce the results.

REFERENCES

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

Ahmad Bdeir, Johannes Burchert, Lars Schmidt-Thieme, and Niels Landwehr. Robust hyperbolic learning with curvature-aware optimization. *arXiv preprint arXiv:2405.13979*, 2024.

Benjamin Paul Chamberlain, James Clough, and Marc Peter Deisenroth. Neural embeddings of graphs in hyperbolic space. *arXiv preprint arXiv:1705.10359*, 2017.

Weize Chen, Xu Han, Yankai Lin, Hexu Zhao, Zhiyuan Liu, Peng Li, Maosong Sun, and Jie Zhou. Fully hyperbolic neural networks. *arXiv preprint arXiv:2105.14686*, 2021.

Ekin D Cubuk, Barret Zoph, Dandelion Mane, Vijay Vasudevan, and Quoc V Le. Autoaugment: Learning augmentation policies from data. *arXiv preprint arXiv:1805.09501*, 2018.

Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pp. 248–255. Ieee, 2009.

Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*, 2020.

Aleksandr Ermolov, Leyla Mirvakhabova, Valentin Khrulkov, Nicu Sebe, and Ivan Oseledets. Hyperbolic vision transformers: Combining improvements in metric learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 7409–7419, 2022.

Jacob Fein-Ashley, Ethan Feng, and Minh Pham. Hvt: a comprehensive vision framework for learning in non-euclidean space. *arXiv preprint arXiv:2409.16897*, 2024.

Octavian Ganea, Gary Bécigneul, and Thomas Hofmann. Hyperbolic neural networks. In *Proceedings of NeurIPS*, pp. 5345–5355, 2018.

Xavier Glorot and Yoshua Bengio. Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the thirteenth international conference on artificial intelligence and statistics*, pp. 249–256. JMLR Workshop and Conference Proceedings, 2010.

Caglar Gulcehre, Misha Denil, Mateusz Malinowski, Ali Razavi, Razvan Pascanu, Karl Moritz Hermann, Peter Battaglia, Victor Bapst, David Raposo, Adam Santoro, et al. Hyperbolic attention networks. In *Proceedings of ICLR*, 2018.

Neil He, Menglin Yang, and Rex Ying. Lorentzian residual neural networks. In *Proceedings of the 31st ACM SIGKDD Conference on Knowledge Discovery and Data Mining V. 1*, pp. 436–447, 2025a.

Neil He, Menglin Yang, and Rex Ying. Hypercore: The core framework for building hyperbolic foundation models with comprehensive modules, 2025b. URL https://arxiv.org/abs/2504.08912.

Dan Hendrycks and Thomas Dietterich. Benchmarking neural network robustness to common corruptions and perturbations. In *International Conference on Learning Representations*, 2019.

Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. 2009.

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

Ya Le and Xuan S. Yang. Tiny imagenet visual recognition challenge. 2015.

Liyuan Liu, Haoming Jiang, Pengcheng He, Weizhu Chen, Xiaodong Liu, Jianfeng Gao, and Jiawei Han. On the variance of the adaptive learning rate and beyond. *arXiv preprint arXiv:1908.03265*, 2019.

I Loshchilov. Decoupled weight decay regularization. *arXiv preprint arXiv:1711.05101*, 2017.

Ilya Loshchilov and Frank Hutter. Sgdr: Stochastic gradient descent with warm restarts. *arXiv preprint arXiv:1608.03983*, 2016.

Pascal Mettes, Mina Ghadimi Atigh, Martin Keller-Ressel, Jeffrey Gu, and Serena Yeung. Hyperbolic deep learning in computer vision: A survey. *International Journal of Computer Vision*, 132 (9):3484–3508, 2024.

Gal Mishne, Zhengchao Wan, Yusu Wang, and Sheng Yang. The numerical stability of hyperbolic representation learning, 2024. URL https://arxiv.org/abs/2211.00181.

Maximilian Nickel and Douwe Kiela. Poincaré embeddings for learning hierarchical representations. *Advances in Neural Information Processing Systems*, 30, 2017.

Maximilian Nickel and Douwe Kiela. Learning continuous hierarchies in the lorentz model of hyperbolic geometry. *arXiv preprint arXiv:1806.03417*, 2018.

Lalit Pandey, Samantha Wood, and Justin Wood. Are vision transformers more data hungry than newborn visual systems? *Advances in Neural Information Processing Systems*, 36:73104–73121, 2023.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, et al. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9, 2019.

Arlan Ramsay and Robert D Richtmyer. *Introduction to hyperbolic geometry*. Springer Science & Business Media, 1995.

Nikhila Ravi, Valentin Gabeur, Yuan-Ting Hu, Ronghang Hu, Chaitanya Ryali, Tengyu Ma, Haitham Khedr, Roman Rädle, Chloe Rolland, Laura Gustafson, et al. Sam 2: Segment anything in images and videos. *arXiv preprint arXiv:2408.00714*, 2024.

Oriane Siméoni, Huy V Vo, Maximilian Seitzer, Federico Baldassarre, Maxime Oquab, Cijo Jose, Vasil Khalidov, Marc Szafraniec, Seungeun Yi, Michaël Ramamonjisoa, et al. Dinov3. *arXiv preprint arXiv:2508.10104*, 2025.

Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jon Shlens, and Zbigniew Wojna. Rethinking the inception architecture for computer vision. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 2818–2826, 2016.

Alexandru Tifrea, Gary Becigneul, and Octavian-Eugen Ganea. Poincare glove: Hyperbolic word embeddings. In *Proceedings of ICLR*, 2018.

Menglin Yang, Harshit Verma, Delvin Ce Zhang, Jiahong Liu, Irwin King, and Rex Ying. Hypformer: Exploring efficient transformer fully in hyperbolic space. In *Proceedings of the 30th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pp. 3770–3781, 2024.

Sangdoo Yun, Dongyoon Han, Seong Joon Oh, Sanghyuk Chun, Junsuk Choe, and Youngjoon Yoo. Cutmix: Regularization strategy to train strong classifiers with localizable features. In *Proceedings of the IEEE/CVF international conference on computer vision*, pp. 6023–6032, 2019.

Hongyi Zhang. mixup: Beyond empirical risk minimization. *arXiv preprint arXiv:1710.09412*, 2017.

A Appendix

A.1 Design Comparison with Prior Hyperbolic Transformers

Table 4 summarizes how HexFormer differs from prior hyperbolic transformers in the three choices that matter for attention: the model of hyperbolic space, how attention scores are computed, and how values are aggregated. Using hyperbolic distance to compute attention scores is not itself new (Gulcehre et al., 2018; Law et al., 2019), and neither is the use of exponential and logarithmic maps inside a hyperbolic transformer (Fein-Ashley et al., 2024). What is new in HexFormer is the combination, scoring with the Lorentzian distance while aggregating through the exponential map.

Each entry in Table 4 follows the description given in the corresponding paper. One point is worth noting for completeness: in the implementation released with HVT (Fein-Ashley et al., 2024), queries, keys and values are mapped to the tangent space at the origin and the attention is computed there as a Euclidean dot product before the result is mapped back, rather than scoring with the Poincaré distance on the manifold as the paper describes. The table reports the published description in both cases.

Table 4: Design comparison of hyperbolic vision and graph transformers. "Hyperbolic scope" indicates which parts of the architecture operate on the manifold. Centroid aggregation is also referred to as midpoint aggregation.

Model	Geometry	Hyperbolic scope	Attention score	Value aggregation	Task
Hyp-ViT (Ermolov et al., 2022)	Poincaré	output embedding	Euclidean dot product	Euclidean	retrieval
HVT (Fein-Ashley et al., 2024)	Poincaré	full model	Poincaré distance	centroid	classification
LViT (He et al., 2025b)	Lorentz	full model	squared Lorentzian distance	centroid	classification
HypFormer (Yang et al., 2024)	Lorentz	full model	linear attention	centroid	graphs
HexFormer (ours)	Lorentz	full model	squared Lorentzian distance	exponential map	classification
HexFormer-Hybrid (ours)	Lorentz	encoder (Euclidean head)	squared Lorentzian distance	exponential map	classification

A.2 Training Strategy and Experiments Set-up

Data Augmentation: Random horizontal flips, random cropping with padding, CIFAR10Policy (Cubuk et al., 2018), normalization, and random erasing (probability 0.25) were applied. Dynamic augmentation using CutMix (Yun et al., 2019) or MixUp (Zhang, 2017) with probability 0.5 was applied on-the-fly during training.

Weight Initialization and Loss: Xavier Normal initialization (Glorot & Bengio, 2010) and Label Smoothing Cross-Entropy (Szegedy et al., 2016) were used.

Learning Rate Scheduler: Cosine Annealing Warmup Restarts (Loshchilov & Hutter, 2016) managed the learning rate, incorporating warmup periods followed by periodic annealing.

Optimizer: In the reported experiments, AdamW (Loshchilov, 2017) was used for Euclidean models, and Riemannian AdamW (Bdeir et al., 2024) was used for HexFormer and HexFormer-Hybrid models. In Appendix Subsection A.3 there is an additional experiment to compare both optimizers for both Euclidean ViT and HexFormer in Table 10.

Hyperparameters and Tuning: Table 6 lists the training hyperparameters for Tiny models, while Table 7 reports the learning rate and weight decay for the other models, with all remaining hyperparameters matching those in Table 6. Learning rate and weight decay were tuned for Euclidean ViT, HexFormer with ExpAgg and HexFormer with centroid, at every model scale. For the HexFormer-Hybrid the same hyperparameters that were tuned for HexFormer were used without further tuning. Tuning was carried out separately on CIFAR-100 and Tiny-ImageNet, and the CIFAR-100 values were reused for CIFAR-10. LViT was tuned in the same way when retrained under this pipeline. For the larger variants, the search was initialized from the scaled learning rate given by

$$\eta_{new} = \eta \times \sqrt{\frac{\text{hidden dimensions}}{\text{new hidden dimensions}}}.$$

and the learning rate and weight decay were then tuned at each scale. The resulting values for every variant are listed in Table 7.

Seeds: Unless stated otherwise in the corresponding caption, results on CIFAR-100 with ViT-Tiny are averaged over 10 seeds, results on CIFAR-10, Tiny-ImageNet and the corruption benchmark over 3 seeds, and results for ViT-Small and ViT-Base over 5 seeds. Single-run measurements are identified where they occur.

Table 5: Vision Transformer (ViT) model variants with architectural specifications.

Model	Patch Size	Hidden Dim	MLP Dim	#Layers	#Heads	Parameters
ViT-Tiny	4	192	384	9	12	$\sim$3M
ViT-Small	4	384	768	12	6	$\sim$14M
ViT-Base	4	768	3072	12	12	$\sim$85M

Table 6: Training hyperparameters for Euclidean ViT, HexFormer, HexFormer-Hybrid, and LViT using Tiny-ViT size on CIFAR-100 and CIFAR-10.

Hyperparameter	**Euclidean ViT**	**HexFormer**	**HexFormer-Hybrid**	**LViT**
Number of epochs	100	100	100	100
Warmup	0 or 10	0 or 10	0 or 10	10
Patch size	4	4	4	4
Batch size (train)	128	128	128	128
Batch size (test)	512	512	512	512
Learning rate	4.15×10^{-3}	4.35×10^{-3}	4.35×10^{-3}	3.5×10^{-3}
Weight decay	6×10^{-2}	5×10^{-2}	5×10^{-2}	3×10^{-2}
Optimizer	AdamW	Riemannian AdamW	Riemannian AdamW	Riemannian AdamW
Encoder	Euclidean	Lorentz	Lorentz	Lorentz
Classification head	Euclidean MLR	Lorentz MLR	Euclidean MLR	Lorentz MLR

Table 7: Learning rate and weight decay hyperparameters for Euclidean ViT, LViT and HexFormer (the Hyperparameters for HexFormer were also used for HexFormer-Hybrid).

Hyperparameter	**Learning rate**	**Weight decay**
Tiny-ViT HexFormer with Centroid	4.65×10^{-3}	5×10^{-2}
Small-ViT Euclidean	2.934×10^{-3}	6×10^{-2}
Small-ViT HexFormer	3.076×10^{-3}	5×10^{-2}
Base-ViT Euclidean	2.934×10^{-3}	1×10^{-1}
Base-ViT HexFormer	3.076×10^{-3}	1×10^{-1}
Tiny-ViT Euclidean Tiny-ImageNet dataset	3×10^{-3}	8×10^{-2}
Tiny-ViT Hexformer Tiny-ImageNet dataset	3×10^{-3}	5×10^{-2}
Tiny-ViT LViT Tiny-ImageNet dataset	3.5×10^{-3}	1×10^{-2}

Curvature Selection: All experiments in this work use a fixed curvature of -1. Additionally preliminary tests were conducted with alternative fixed curvatures of -0.5 and -2 for the Tiny-ViT configuration. While training and inference remained stable across these settings, the overall performance was consistently lower, and therefore these variants were not extended to larger models. We also experimented with making the curvature learnable and observed a slight improvement on CIFAR-100 for both warmup and no-warmup training regimes in the Tiny-ViT setting, without introducing any stability issues. These observations suggest that curvature learning is a promising direction for further improving hyperbolic Vision Transformers.

A.3 Additional Experiments

This section presents supplementary experiments evaluating HexFormer and HexFormer-Hybrid under various settings, including model scale, activation functions, optimizers, gradient behavior, and robustness to common image corruptions. Most of the datasets used in these experiments exhibit hierarchical class relations and high hyperbolicity (low δ_{rel}). Appendix A.3.3 deliberately departs from this by evaluating on datasets with little hierarchical structure. In addition to the main HexFormer and hybrid design, an alternative hybrid variant that uses a Euclidean encoder combined with a hyperbolic classification head (MLR) was experimented as well. However, this configuration did not yield competitive performance. The following subsections report results on ViT scale, activation functions, optimizer choices, performance on datasets with weak hierarchical structure, gradient stability (qualitative and quantitative), epoch-wise performance, and corruption robustness, highlighting both the settings in which hyperbolic representations help and those in which they do not.

The HexFormer-Hybrid variant achieves the best performance across all experiments, using each geometry where it is most effective: negative curvature for feature extraction, and a standard Euclidean classifier for the final decision.

Hyperbolicity: Hyperbolicity, often defined in the sense of Gromov, measures how close a metric space is to behaving like a tree or a negatively curved space. A space is said to be δ-hyperbolic if all of its geodesic triangles are "δ-slim," meaning each side lies within a small neighborhood of the other two. Equivalently, hyperbolicity can be characterized by a four-point condition based on pairwise distances. Low hyperbolicity values indicate strong tree-like structure, while higher values suggest the space is more Euclidean-like. This property is commonly used to assess whether data or graphs are well suited to hyperbolic representation.

Choice of Datasets: Hyperbolic geometry is especially effective for data with hierarchical or coarse-to-fine structure, as distances in negatively curved spaces grow exponentially with radius (Nickel & Kiela, 2017; Ganea et al., 2018). HCNN (Bdeir et al., 2023) has shown that datasets like CIFAR-10 and CIFAR-100 (Krizhevsky et al., 2009), and Tiny-ImageNet (Le & Yang, 2015), exhibit measurable hierarchical structure, reflected in their hyperbolicity scores (reported in Table 1). Concretely, CIFAR-10 contains latent semantic groupings (animals vs. vehicles), while CIFAR-100 has an explicit 20-superclass hierarchy defined in the dataset annotations. Tiny-ImageNet, being a subset of ImageNet, inherits the well-established WordNet taxonomy on which ImageNet is built. Categories follow paths such as mammal →carnivore →canine →dog →husky, providing deep hierarchical relationships (Deng et al., 2009). These properties make all three datasets suitable for evaluating hyperbolic representations.

In addition to these hierarchically structured datasets, Euclidean ViT, HexFormer, and HexFormer-Hybrid were also evaluated on datasets with minimal or weak hierarchical structure, namely MNIST, Fashion-MNIST, and SVHN. These datasets are semantically "flat," with labels corresponding to digits or simple clothing categories. As reported in Appendix A.3.3, all three models achieve very similar performance on these benchmarks, with no consistent advantage for hyperbolic methods. These results further support the conclusion that hyperbolic architectures provide limited benefits when the underlying data lack hierarchical organization.

A.3.1 Vision Transformer Scale Comparison

Table 8 shows CIFAR-100 performance on different ViT variants. HexFormer and HexFormer-Hybrid models outperform Euclidean models at every scale, with larger models amplifying the improvement. Table 5 in Appendix A.2 lists ViT variant architectures and parameter counts, and the hyperparameter choices for all models are provided in Appendix A.2. Although Small and Base ViTs would be expected to outperform Tiny-ViT by a larger margin, the larger configurations overfit on a dataset of this size, so the gains over Tiny-ViT remain modest for all three geometries. This overfitting is not geometry-neutral: as shown in Figure 4, the gap between training and validation accuracy grows with model size, and it is substantially wider for the Euclidean ViT than for HexFormer. The advantage of hyperbolic representations therefore increases rather than diminishes as the models grow, suggesting that hyperbolic feature representations provide robust gains that generalize across model sizes.

Table 8: Accuracy for different ViT models scales for HexFormer, HexFormer-Hybrid and Euclidean ViT on CIFAR-100 under warmup and no warmup. Averaged over 10 seeds for ViT-Tiny and 5 seeds for ViT-Small and ViT-Base.

ViT Variant	Model	CIFAR-100	
		Warmup = 0	**Warmup = 10**
ViT-Tiny	Euclidean ViT	$71.86_{\pm 0.255}$	$74.64_{\pm 0.408}$
	HexFormer	$\underline{75.06}_{\pm 0.515}$	$\underline{75.65}_{\pm 0.434}$
	HexFormer-Hybrid	$\mathbf{75.44}_{\pm 0.623}$	$\mathbf{75.85}_{\pm 0.337}$
ViT-Small	Euclidean ViT	$70.53_{\pm 0.451}$	$73.6_{\pm 1.046}$
	HexFormer	$\underline{75.59}_{\pm 0.47}$	$\underline{75.54}_{\pm 0.703}$
	HexFormer-Hybrid	$\mathbf{75.74}_{\pm 0.759}$	$\mathbf{76.09}_{\pm 0.68}$
ViT-Base	Euclidean ViT	$70.72_{\pm 0.59}$	$70.54_{\pm 0.375}$
	HexFormer	$\underline{75.73}_{\pm 1.074}$	$\underline{76.49}_{\pm 1.214}$
	HexFormer-Hybrid	$\mathbf{76.44}_{\pm 1.423}$	$\mathbf{76.81}_{\pm 0.51}$

Generalization gap across scales: Figure 4 tracks the difference between training and validation accuracy over the course of training for the Tiny and Base configurations. Because the training data is augmented and mixed (CutMix/MixUp, Appendix A.2), training accuracy normally sits *below* clean validation accuracy, so a negative gap is the healthy regime and a gap crossing zero indicates that the model is fitting the training distribution harder than it generalizes. All four configurations behave alike for the first 25 epochs, falling to a gap of roughly -20 points, and then separate by scale and by geometry. The Euclidean ViT-Base rises steadily from that point, crosses zero near epoch 70, and ends training at roughly $+8$ points. HexFormer ViT-Base rises far more slowly and remains negative throughout, ending near -1. Both Tiny configurations stay in the healthy regime for the entire run and end near -7. Overfitting at Base scale is therefore not a property of capacity alone: under the same pipeline and at the same capacity, only the Euclidean model crosses into the overfitting regime.

This accounts for the ordering in Table 8. The Euclidean baseline does not benefit from additional capacity on CIFAR-100: its accuracy falls from 74.64 at Tiny to 73.60 at Small and 70.54 at Base (warmup $= 10$), the only one of the three models to degrade monotonically with scale. HexFormer and HexFormer-Hybrid instead improve over the same range, reaching 76.49 and 76.81 at Base. The scale at which the two geometries diverge in Figure 4 is the scale at which their accuracies diverge in Table 8, indicating that the widening margin for the hyperbolic models at Small and Base reflects how well each geometry resists overfitting rather than a difference in tuning effort.

A.3.2 Additional Experiments on different activation function and on different optimizer

Additional experiments were conducted to evaluate the robustness of HexFormer and HexFormer-Hybrid under variations in activation functions and optimizer choices. Using Leaky ReLU instead of GELU in the encoder, both HexFormer and HexFormer-Hybrid outperform the Euclidean ViT under warmup and no warmup conditions (Table 9). The gradient stability observed during training is maintained across warmup settings, demonstrating the robustness of the hyperbolic representations to different activation functions.

Experiments with different optimizers further illustrate the behavior of the models. Euclidean ViTs perform better with AdamW (Loshchilov, 2017) (as expected, since Euclidean models do not have manifold parameters and thus do not require Riemannian AdamW (Bdeir et al., 2024)), while hyperbolic models benefit more from Riemannian AdamW (Bdeir et al., 2024) (Table 10). Importantly, the gradient stability is also evident when using standard AdamW (Loshchilov, 2017) for hyperbolic models, indicating that this stability arises from the hyperbolic geometry rather than the optimizer choice.

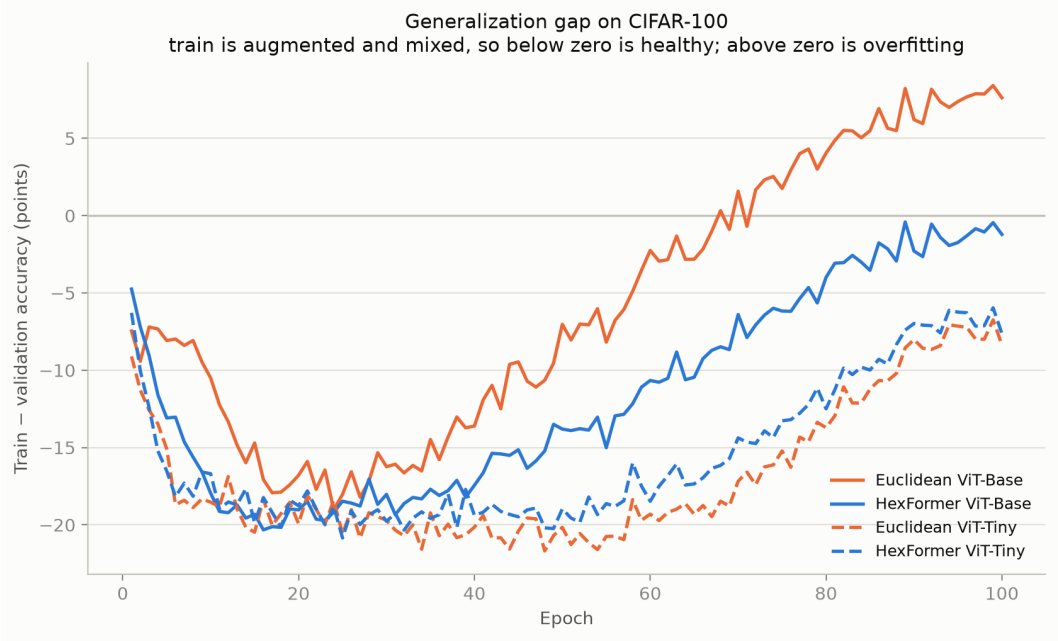


Figure 4: Generalization gap (training minus validation accuracy, in accuracy points) over training on CIFAR-100, for the Tiny and Base configurations of the Euclidean ViT and HexFormer. Training data is augmented and mixed, so a negative gap is the healthy regime and values above zero indicate overfitting. The Euclidean ViT-Base crosses into the overfitting regime near epoch 70 and ends around $+8$ points, whereas HexFormer ViT-Base remains below zero throughout. The two Tiny configurations track each other closely and stay in the healthy regime for the entire run.

Table 9: Accuracy comparison between HexFormer and Euclidean ViT on CIFAR-100 with Leaky ReLU activation under warmup and no warmup. Averaged over 5 seeds.

Model	**CIFAR-100**	
	Warmup = 0	**Warmup = 10**
Euclidean ViT	$72.17_{\pm 0.43}$	$74.2_{\pm 0.24}$
HexFormer	$\underline{74.65}_{\pm 0.407}$	$\underline{74.52}_{\pm 0.331}$
HexFormer-Hybrid	$\mathbf{74.85}_{\pm 0.459}$	$\mathbf{74.73}_{\pm 0.378}$

A.3.3 Experiments on weak/flat Hierarchical Datasets:

To further examine the behavior of HexFormer beyond hierarchically structured data, Euclidean ViT, HexFormer, and HexFormer-Hybrid were evaluated on datasets that exhibit minimal or weak hierarchical structure, namely MNIST, Fashion-MNIST, and SVHN. These datasets contain labels corresponding to digits or simple clothing categories and are considered ”flat” in terms of semantic hierarchy. Across all three datasets, the results were very similar among the three models. The experiments were ran 3 times on different seeds with warmup 10, the Euclidean ViT achieved slightly higher accuracy, HexFormer slightly outperforms the Hybrid variant on Fashion-MNIST, while the two are essentially tied on MNIST and SVHN. Overall, these findings indicate that hyperbolic architectures do not provide significant advantages when the underlying data lack hierarchical structure, and all three models behave comparably in such settings.

A.3.4 Gradients Stability

Figures 6 to 10 present histograms of the absolute gradient values in both offset and overlay view modes, comparing HexFormer, HexFormer-Hybrid, and Euclidean ViTs. HexFormer and

Table 10: Accuracy comparison between HexFormer and Euclidean ViT on CIFAR-100 under warmup and no warmup, using AdamW (Loshchilov, 2017) and Riemannian AdamW (Bdeir et al., 2024) optimizers.

Model \ Optimizer	AdamW		RiemannianAdamW	
	Warmup = 0	**Warmup = 10**	**Warmup = 0**	**Warmup = 10**
Euclidean ViT	$71.86_{\pm 0.255}$	$\mathbf{74.64}_{\pm 0.408}$	$71.49_{\pm 0.467}$	$73.7_{\pm 0.443}$
HexFormer	$\mathbf{74.68}_{\pm 0.52}$	$74.63_{\pm 0.427}$	$\mathbf{75.06}_{\pm 0.515}$	$\mathbf{75.65}_{\pm 0.434}$

Table 11: Results on weak/flat Hierarchical Datasets

Model	MNIST	Fashion-MNIST	SVHN
Euclidean ViT	$\mathbf{99.41}_{\pm 0.02}$	$\mathbf{94.56}_{\pm 0.10}$	$\mathbf{97.31}_{\pm 0.04}$
HexFormer	$99.35_{\pm 0.06}$	$\underline{94.02}_{\pm 0.20}$	$\underline{97.13}_{\pm 0.181}$
HexFormer-Hybrid	$\underline{99.39}_{\pm 0.01}$	$93.86_{\pm 0.13}$	$\underline{97.13}_{\pm 0.104}$

HexFormer-Hybrid display more stable gradients, maintaining smooth distributions even without warmup, whereas Euclidean models show clear distortions under the same conditions.

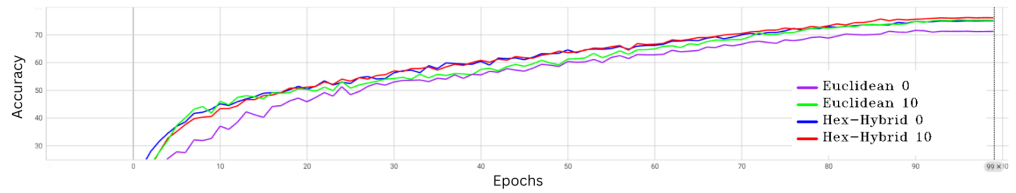
The histograms above show that the distributions differ, but not by how much, nor at which point in training. The remainder of this subsection reduces the same logged data to a single scalar per snapshot so that the claim can be tested statistically.

Metric: Every run logs a histogram of absolute gradient values at fixed iteration checkpoints. Rather than re-binning those histograms, the exact per-snapshot mean and standard deviation of $\log|g|$ were recovered analytically from the summary statistics stored with each record, namely the sum of the logged magnitudes, the sum of their squares and the number of entries, so no bucket approximation enters the measurement. The stability metric is the standard deviation of $\log|g|$ within a snapshot: a lower value means gradient magnitudes across parameters are concentrated within a narrower band, which is the property warmup is conventionally introduced to provide (Liu et al., 2019). Note that this measures the *spread* of gradient magnitudes, not their overall scale. A model is not called more stable here merely for having smaller gradients.

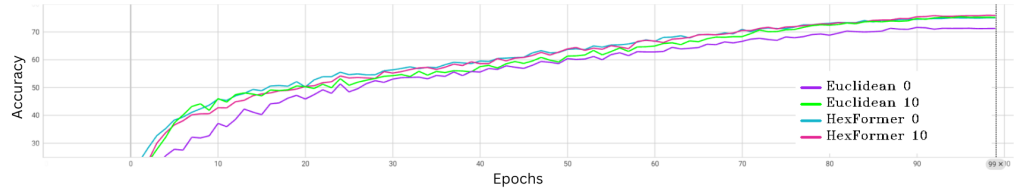
Because all runs log at identical iterations, any two of them can be compared as matched pairs. Snapshots are paired on the step value itself rather than by position, so a difference in logging cadence cannot silently misalign them. Each comparison is summarized by the mean margin over the matched snapshots, together with whether that margin keeps a consistent sign across them. Where the differences are uniformly signed, a paired significance test is at its numerical floor and depends only on the number of snapshots, so no p-values are reported. Experiments use ViT-Small on CIFAR-100, matching the setting of Figure 3, and each run covers 100 epochs, or roughly $117{,}000$ steps and $1{,}200$ logged snapshots.

Measurements are reported over four windows of training rather than one, because the picture changes with the window: an early window (the first 10 epochs, 126 snapshots up to step $11{,}800$), a middle window (epochs 20–50, 360 snapshots), a final window (epochs 80–100, 240 snapshots), and the full run ($1{,}200$ snapshots).

Warmup dependency is geometry-dependent: The Early column of Table 12 shows that warmup reduces gradient dispersion for all three models, but by very different amounts. For the Euclidean ViT the reduction is 0.815, roughly three times the reduction for HexFormer (0.259) and more than twice that for HexFormer-Hybrid (0.371). The accuracy consequence follows the same ordering: at this scale, removing warmup costs the Euclidean model 3.07 points of validation accuracy, whereas HexFormer and HexFormer-Hybrid change by $+0.05$ and -0.35 (Table 8).



(a) HexFormer-Hybrid vs. Euclidean on CIFAR-100



(b) HexFormer vs. Euclidean on CIFAR-100

Figure 5: Comparison of accuracy over epochs on CIFAR-100. The numbers 0 and 10 in the legend refer to the warmup.

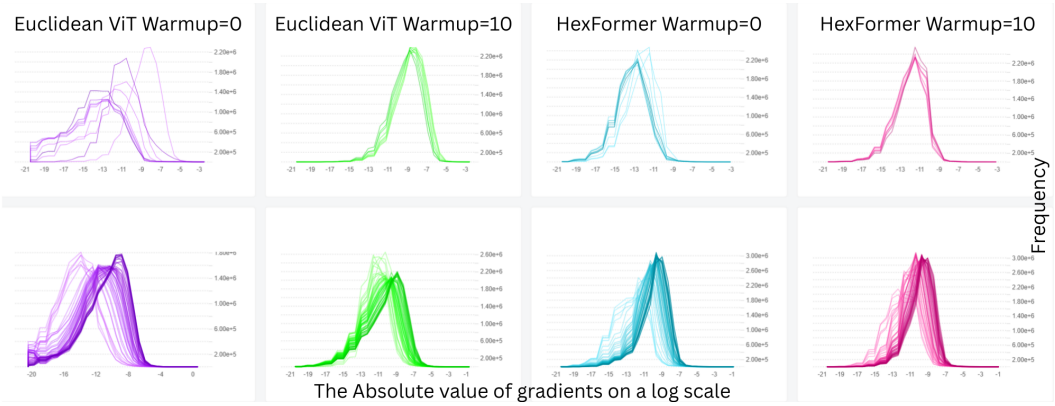


Figure 6: The histogram of the absolute value of gradients on a log scale during the training of ViT-Small on CIFAR-100 comparing HexFormer vs. Euclidean ViT with and without warmup (overlay mode). Top row shows close-up for first 200 iterations.

The stability advantage holds in both regimes, but is largest without warmup: Tables 12 and 13 isolate the effect of geometry at a fixed warmup setting. Without warmup, both hyperbolic models are more stable than the Euclidean baseline in every window, by $+1.109$ (HexFormer) and $+0.547$ (HexFormer-Hybrid) over the full run. With warmup the same holds, by $+0.572$ and $+0.385$ over the full run, with the gap widening as training proceeds: $+0.805$ and $+0.618$ over epochs 80–100.

The early window is the one exception, and it is the reason the window matters. Over the first 10 epochs with warmup, the three models sit within 0.035 of one another and the ordering briefly reverses, with the hyperbolic models marginally *more* dispersed than the baseline. This is the only setting in Table 13 in which the margin does not keep a consistent sign across snapshots. Reading that window alone would suggest the geometries are equivalent once warmup is applied, which the rest of the run contradicts.

Figure 11 shows why, by placing the whole run beside a close-up of that early window. In the close-up the hyperbolic curves rise through the window and end above the Euclidean one. The full-run panel reveals this to be a transient: the hyperbolic curves peak shortly after the window closes, around step $18{,}000$, and then fall back and flatten, while the Euclidean curve continues to climb for the rest of training. By the end, the Euclidean model is the most dispersed of the three under both

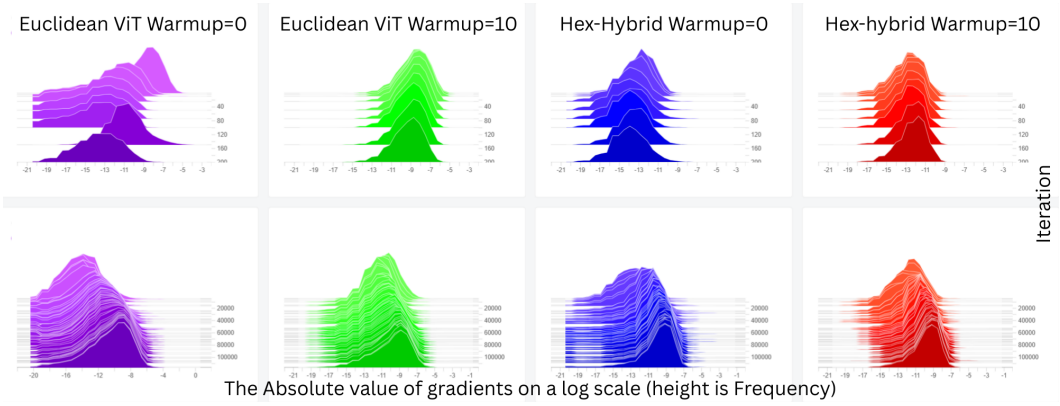


Figure 7: The histogram of the absolute value of gradients on a log scale during the training of ViT-Small on CIFAR-100 comparing HexFormer-Hybrid vs. Euclidean ViT with and without warmup (offset mode). Top row shows close-up for first 200 iterations.

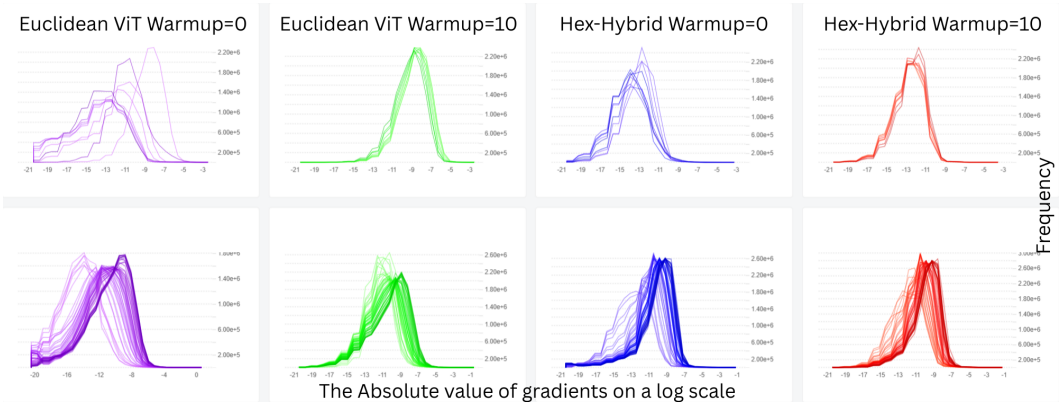


Figure 8: The histogram of the absolute value of gradients on a log scale during the training of ViT-Small on CIFAR-100 comparing HexFormer-Hybrid vs. Euclidean ViT with and without warmup (overlay mode). Top row shows close-up for first 200 iterations.

warmup conditions. The direction of travel differs as well: with warmup, Euclidean dispersion rises from 1.534 early to 2.215 over the final fifth of training, whereas HexFormer's falls from 1.558 to 1.411.

A rising spread of gradient magnitudes is not in itself a defect. It is expected as parameters converge and their gradients approach zero at different rates, and it matters chiefly early in training, when the adaptive optimizer's second-moment estimates are still poorly conditioned, which is exactly the regime warmup was introduced to protect (Liu et al., 2019). Two observations argue against reading the metric as a quality score. The mean of $\log|g|$ is slightly *higher* for both hyperbolic models at the end of training than for the Euclidean baseline, so nothing is vanishing. And dispersion does not track accuracy monotonically: under warmup, HexFormer-Hybrid is more dispersed than HexFormer (1.651 against 1.464 over the full run) and also more accurate (76.09 against 75.54, Table 8).

The reading these measurements support is that negative curvature supplies at initialization much of what warmup is otherwise needed to supply, which is why removing warmup costs the Euclidean model 3.07 points and the hyperbolic models considerably less, and that it additionally keeps gradient magnitudes more tightly concentrated for the remainder of training in both regimes. HexFormer and HexFormer-Hybrid do differ from each other, however. Without warmup, HexFormer-Hybrid's dispersion climbs over training ($1.940 \rightarrow 2.421$), so its margin over the Euclidean baseline narrows to $+0.296$ by the final window, whereas HexFormer's falls across the same span ($1.817 \rightarrow 1.597$)

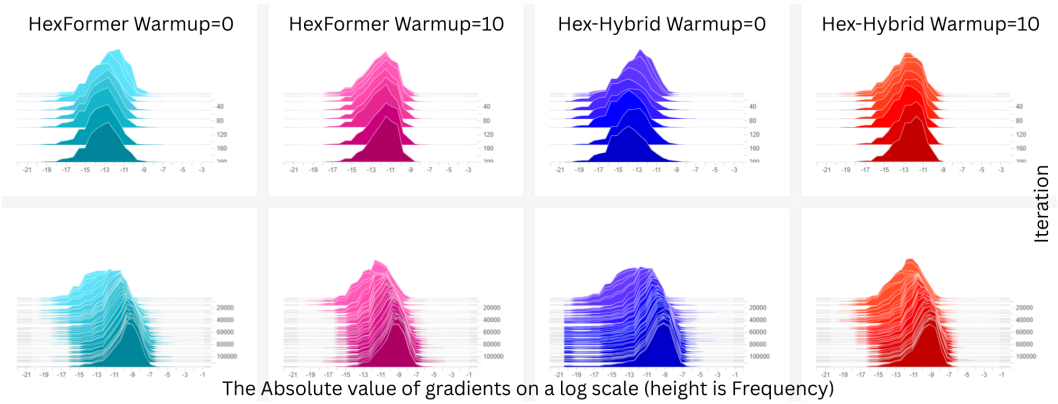


Figure 9: The histogram of the absolute value of gradients on a log scale during the training of ViT-Small on CIFAR-100 comparing HexFormer vs. HexFormer-Hybrid with and without warmup (offset mode). Top row shows close-up for first 200 iterations.

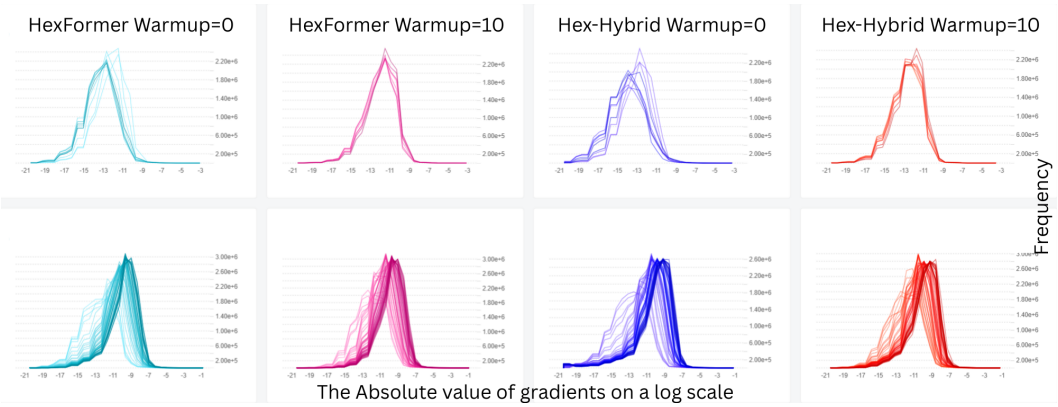


Figure 10: The histogram of the absolute value of gradients on a log scale during the training of ViT-Small on CIFAR-100 comparing HexFormer vs. HexFormer-Hybrid with and without warmup (overlay mode). Top row shows close-up for first 200 iterations.

and its margin widens to $+1.120$. The claim should therefore not be collapsed across the two variants.

A.3.5 Epoch-wise Performance Analysis

Table 14 shows accuracy across different training budgets on CIFAR-100 (no warmup). HexFormer and HexFormer-Hybrid reach higher accuracy than the Euclidean ViT at every budget, suggesting that the hyperbolic variants need fewer epochs to become competitive in this setting. Since each entry is a separate run with its own learning rate schedule, the table compares fixed training budgets rather than tracing a single convergence curve.

A.3.6 Robustness to Common Corruptions

The experiments reported so far evaluate all models on clean test data. Because negative curvature constrains the scale of activations and organizes representations radially, it is natural to ask whether hyperbolic representations also degrade more gracefully when the input distribution is perturbed. This subsection evaluates that question on the standard common-corruptions benchmark.

Setup: All models were evaluated on CIFAR-100-C (Hendrycks & Dietterich, 2019), which applies 19 corruption types at 5 severity levels each to the CIFAR-100 test set (95 corrupted test sets of 10,000 images, i.e. 950,000 forward evaluations per model). This is a pure evaluation protocol:

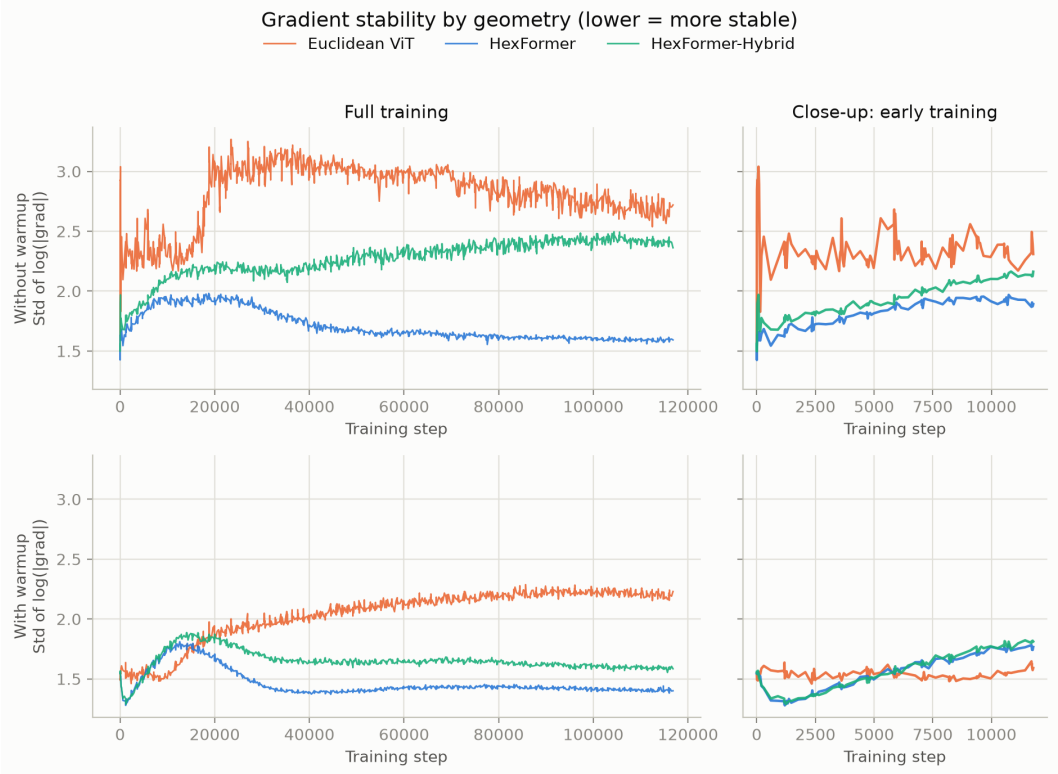


Figure 11: Standard deviation of $\log|g|$ against training step for ViT-Small on CIFAR-100, by encoder geometry (lower is more stable). Rows are the warmup conditions. The left column shows the whole 100-epoch run and the right column a close-up of the early window (the first 10 epochs), tabulated as "Early" in Tables 12 and 13. All four panels share a y-axis. Without warmup (top) the Euclidean model is markedly more dispersed throughout. With warmup (bottom) the close-up alone suggests the geometries are equivalent and that the hyperbolic models end the window slightly higher, but the full-run panel beside it shows this to be a transient that peaks near step 18,000 and reverses: the Euclidean curve climbs for the rest of training while both hyperbolic curves fall back and flatten.

Table 12: Gradient-magnitude dispersion (standard deviation of $\log|g|$, lower is more stable) by geometry, warmup condition and window of training. Lowest per column within each condition block in bold, second lowest underlined. The early window is plotted in the right-hand column of Figure 11. That window on its own is not representative of the full run.

Condition	Model	Window of training (epochs)			
		Early (0–10)	**Middle** (20–50)	**Final** (80–100)	**Full run** (0–100)
Warmup = 0	Euclidean ViT	2.349	3.035	2.717	2.816
	HexFormer	**1.817**	**1.751**	**1.597**	**1.707**
	HexFormer-Hybrid	<u>1.940</u>	<u>2.219</u>	<u>2.421</u>	<u>2.269</u>
Warmup = 10	Euclidean ViT	**1.534**	2.023	2.215	2.036
	HexFormer	<u>1.558</u>	**1.424**	**1.411**	**1.464**
	HexFormer-Hybrid	1.569	<u>1.665</u>	<u>1.598</u>	<u>1.651</u>

Table 13: Margin of each hyperbolic model over the Euclidean baseline, per window, with windows as defined in Table 12. Δ is the Euclidean dispersion minus the model's own, so positive means more tightly concentrated gradient magnitudes than the baseline. In every window except the two early-window comparisons under warmup, which are marked †, the margin is positive at every matched snapshot. Larger margin per column in bold.

Condition	Model	Δ vs. Euclidean ViT			
		Early	**Middle**	**Final**	**Full run**
Warmup = 0	HexFormer	$\mathbf{+0.532}$	$\mathbf{+1.284}$	$\mathbf{+1.120}$	$\mathbf{+1.109}$
	HexFormer-Hybrid	$+0.409$	$+0.816$	$+0.296$	$+0.547$
Warmup = 10	HexFormer	$\mathbf{-0.024}^{\dagger}$	$\mathbf{+0.599}$	$\mathbf{+0.805}$	$\mathbf{+0.572}$
	HexFormer-Hybrid	$-0.035^{\dagger}$	$+0.358$	$+0.618$	$+0.385$

†: the early window under warmup is the only setting in which the sign of the margin is not consistent across snapshots. The values for both models are negligible in magnitude.

no model was retrained, fine-tuned, or exposed to any of these corruptions during training, and the preprocessing applied to corrupted images is identical to the clean test transform (normalization only). Three independently trained ViT-Tiny checkpoints per geometry were used, each trained on CIFAR-100 under the settings of Table 1 and each evaluated once. Following Hendrycks & Dietterich (2019), the 19 corruptions are grouped into the four families *noise*, *blur*, *weather*, and *digital*. Alongside the mean corrupted accuracy (mCA, averaged over all 19×5 settings), *retention* is also reported: mCA divided by the model's own clean accuracy. Retention serves the same purpose as the Relative mCE of Hendrycks & Dietterich (2019), isolating how much a model degrades from how accurate it is to begin with, but normalizes each model by its own clean accuracy rather than against a separate reference model.

Unit of analysis: The evaluation itself is deterministic: the benchmark is a fixed set of pre-generated images, the loader is unshuffled and the only preprocessing is normalization, so re-evaluating a checkpoint reproduces its numbers exactly and every source of variance lies in training. One seed therefore contributes exactly one evaluation. Each evaluation is collapsed to a single scalar per seed before any test is applied, and significance is assessed across the three seeds with a paired t-test on the per-seed differences, with Holm correction applied over the four headline comparisons (mCA and retention, for each of the two hyperbolic models). Testing across the 95 corruption settings within a run instead would badly overstate significance, since those settings are one model scored on 95 overlapping views of the same 10,000 images and are far from independent. With three seeds the intervals below are correspondingly wide, and no rank test is reported because at $n = 3$ its smallest attainable two-sided p-value is 0.25.

Table 14: Model accuracy when trained for different epochs between HexFormer, HexFormer-Hybrid and Euclidean ViT on CIFAR-100. Notice that due to the learning rate scheduler, this is not analogous to training for 100 epochs and checking the performance at an intermediate epoch.

Model	Epochs	CIFAR-100
Euclidean ViT	20	55.47
HexFormer	20	**58.66**
HexFormer-Hybrid	20	<u>58.57</u>
Euclidean ViT	30	62.99
HexFormer	30	**65.34**
HexFormer-Hybrid	30	<u>64.12</u>
Euclidean ViT	40	66.99
HexFormer	40	<u>67.74</u>
HexFormer-Hybrid	40	**68.67**
Euclidean ViT	50	67.88
HexFormer	50	<u>70.54</u>
HexFormer-Hybrid	50	**70.92**
Euclidean ViT	60	69.54
HexFormer	60	<u>72.58</u>
HexFormer-Hybrid	60	**72.84**
Euclidean ViT	70	70.2
HexFormer	70	<u>72.51</u>
HexFormer-Hybrid	70	**73.98**

Table 15: Corruption robustness on CIFAR-100-C for ViT-Tiny models. mCA is the mean Top-1 accuracy over all 19 corruptions and 5 severities. Retention is mCA relative to each model's own clean accuracy, and therefore measures degradation independently of clean performance. Best value per column in bold, second best underlined. For the two drop columns lower is better. Mean $\pm$ standard deviation over three seeds.

Model	Clean	mCA	Abs. drop	Rel. drop (%)	Retention (%)
Euclidean ViT	74.41 ± 0.46	61.06 ± 0.39	<u>13.35</u>	<u>17.94</u>	<u>82.06</u> ± 0.07
HexFormer	<u>75.18</u> ± 0.34	<u>62.26</u> ± 0.12	**12.92**	**17.18**	**82.82** ± 0.38
HexFormer-Hybrid	**76.16** ± 0.41	**62.55** ± 0.19	13.60	17.86	82.14 ± 0.20

Overall accuracy under corruption: As shown in Table 15, both hyperbolic variants attain higher mean corrupted accuracy than the Euclidean baseline, with HexFormer-Hybrid highest at 62.55 against 61.06. Across seeds this margin is $+1.49$ points, 95% CI $[+1.01, +1.97]$, and it survives Holm correction ($p = 0.022$); HexFormer gains $+1.20$, CI $[+0.52, +1.87]$, which is significant before correction ($p = 0.017$) but not after ($p = 0.051$).

The retention column is the more informative one, and it does not follow. All three models retain almost exactly the same *fraction* of their accuracy under corruption (82.1–82.8%). For HexFormer-Hybrid the retention margin is $+0.08$ points with CI $[-0.56, +0.71]$, an interval centered on zero: its entire corrupted-accuracy advantage is inherited from its clean accuracy rather than reflecting any additional robustness. HexFormer is the one case where a residual effect may survive, at $+0.75$ retention points, CI $[-0.03, +1.54]$, suggestive but not established at three seeds. Averaged over the whole benchmark, therefore, these measurements do not support a general robustness gain from curvature beyond what curvature already buys in clean accuracy.

The advantage grows with severity: The effect that is *not* explained by clean accuracy appears when the benchmark is broken down by severity (Table 16 and Figure 13). At severity 1 HexFormer is 0.45 points ahead of the Euclidean baseline, an interval that includes zero. At severity 5 the margin

Table 16: Accuracy under corruption as a function of severity (mean over the 19 corruption types), and the corresponding margin over the Euclidean baseline in percentage points, averaged over three seeds. Best per severity in bold, second best underlined. * marks a margin whose 95% paired interval excludes zero. The margin grows with severity for both models, and a per-seed least-squares fit of margin against severity gives a slope of $+0.569$ points per level for HexFormer (CI $[+0.365, +0.774]$) and $+0.285$ for HexFormer-Hybrid (CI $[+0.195, +0.376]$).

Model	Severity 1	Severity 2	Severity 3	Severity 4	Severity 5
Euclidean ViT	69.51	65.80	62.42	56.64	50.94
HexFormer	<u>69.96</u>	<u>66.21</u>	<u>63.15</u>	**58.40**	**53.56**
HexFormer-Hybrid	**70.70**	**66.87**	**63.58**	**58.40**	<u>53.21</u>
Margin over Euclidean ViT (percentage points), mean over three seeds					
HexFormer	+0.45	+0.41	+0.73*	+1.76*	+2.63*
HexFormer-Hybrid	+1.19*	+1.07*	+1.16*	+1.75*	+2.28*

is 2.63 points with a paired interval of $[+1.79, +3.47]$. HexFormer-Hybrid rises from $+1.19$ to $+2.28$ over the same range. Fitting the margin against severity within each seed gives $+0.569$ points per severity level for HexFormer, 95% CI $[+0.365, +0.774]$, and $+0.285$ for HexFormer-Hybrid, $[+0.195, +0.376]$; both intervals are clear of zero, so the trend itself, and not merely its endpoint, is supported across seeds. The two geometries are near-indistinguishable on mildly perturbed inputs and diverge as the perturbation becomes severe, which is the signature of more graceful degradation rather than of a uniformly better model.

Table 17: Accuracy under corruption grouped into the four corruption families of Hendrycks & Dietterich (2019) (mean over the corruptions in each family and over all 5 severities). n is the number of corruption types per family, and accuracies are means over three seeds. Best per family in bold, second best underlined. Each margin carries its 95% paired interval. HexFormer and HexFormer-Hybrid both improve on blur, weather and digital corruptions. Both trail the baseline on the noise family, but by an amount whose interval includes zero.

Family	n	Top-1 accuracy			Margin over Euclidean	
		Euclidean	HexFormer	HexFormer-Hybrid	HexFormer	HexFormer-Hybrid
Noise	4	**50.35**	<u>49.91</u>	49.78	-0.44 $[-1.71, +0.83]$	-0.57 $[-1.99, +0.84]$
Blur	5	60.60	**63.12**	<u>62.96</u>	$+2.51$ $[+2.04, +2.98]$	$+2.35$ $[+2.25, +2.45]$
Weather	5	65.83	<u>67.21</u>	**67.89**	$+1.37$ $[+0.29, +2.45]$	$+2.06$ $[+1.60, +2.52]$
Digital	5	65.32	<u>66.33</u>	**67.03**	$+1.01$ $[-0.16, +2.18]$	$+1.71$ $[+1.18, +2.24]$

Which corruptions are helped and which are hurt: Table 17 and Figure 12 show that the aggregate picture hides a systematic split. Both hyperbolic variants improve substantially on blur ($+2.51$ and $+2.35$, both intervals clear of zero) and more modestly on weather and digital corruptions. Both trail the Euclidean baseline on the noise family (-0.44 and -0.57), but those intervals include zero, so the noise deficit is a consistent point estimate rather than an established effect at three seeds. The per-corruption breakdown in Table 18 localizes it: the losses sit in JPEG compression (-3.95 / -3.29) and in the three additive and signal-dependent Gaussian-family noises, Gaussian (-0.98 / -2.08), shot (-1.48 / -1.83) and speckle (-1.42 / -1.42). Impulse (salt-and-pepper) noise is the exception within the family and goes the other way ($+2.12$ / $+3.04$). The largest single gain is on contrast ($+4.27$ / $+5.22$), followed by fog, Gaussian blur, motion blur, pixelate and zoom blur, all above $+2$ points. Counted three ways, HexFormer-Hybrid is the strongest of the three models on 14 corruptions against 4 for the Euclidean baseline and only 1 for HexFormer, so HexFormer largely sits between the other two rather than leading.

The pattern is consistent across HexFormer and HexFormer-Hybrid and is interpretable: corruptions that act globally and at low spatial frequency (contrast, fog, blur, brightness) leave the coarse-to-fine structure that the hyperbolic embedding organizes largely intact, whereas high-frequency per-pixel noise perturbs feature norms directly. Since radial distance from the origin carries semantic weight in the Lorentz model, norm perturbations plausibly translate more directly into representation change there than in a Euclidean feature space. This remains a hypothesis suggested by the breakdown rather than an established mechanism. Isolating it would require controlled perturbations of feature norm at fixed direction, which were not part of these experiments.

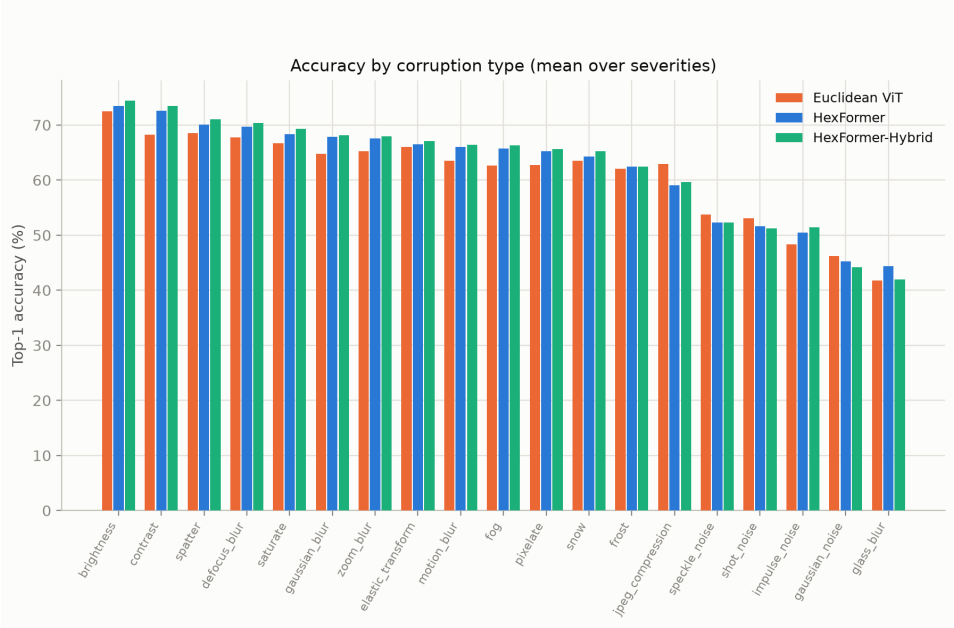
Table 18: Full per-corruption breakdown on CIFAR-100-C (Top-1 accuracy, mean over the 5 severities), grouped by corruption family. The last two columns give the margin over the Euclidean baseline in percentage points. Negative values indicate corruptions on which the Euclidean baseline is stronger. Accuracies are means over three seeds. Best per row in bold, second best underlined. The final row counts the corruptions on which each model is the strongest of the three, and therefore sums to 19.

Corruption	**Top-1 accuracy**			**Margin over Euclidean**	
	Euclidean	**HexFormer**	**HexFormer-Hybrid**	**HexFormer**	**HexFormer-Hybrid**
Noise					
Gaussian noise	**46.24**	<u>45.26</u>	44.16	−0.98	−2.08
Shot noise	**53.05**	<u>51.57</u>	51.22	−1.48	−1.83
Impulse noise	48.37	<u>50.49</u>	**51.41**	+2.12	+3.04
Speckle noise	**53.74**	<u>52.32</u>	52.31	−1.42	−1.42
Blur					
Defocus blur	67.74	<u>69.73</u>	**70.32**	+1.99	+2.58
Glass blur	41.76	**44.40**	<u>41.97</u>	+2.64	+0.21
Motion blur	63.50	<u>66.03</u>	**66.41**	+2.53	+2.91
Zoom blur	65.24	<u>67.57</u>	**67.93**	+2.33	+2.69
Gaussian blur	64.79	<u>67.86</u>	**68.15**	+3.06	+3.35
Weather					
Snow	63.54	<u>64.32</u>	**65.20**	+0.78	+1.66
Frost	62.01	<u>62.45</u>	**62.45**	+0.44	+0.44
Fog	62.64	<u>65.73</u>	**66.33**	+3.08	+3.68
Brightness	72.45	<u>73.42</u>	**74.41**	+0.97	+1.97
Spatter	68.53	<u>70.12</u>	**71.06**	+1.59	+2.53
Digital					
Contrast	68.27	<u>72.54</u>	**73.49**	+4.27	+5.22
Elastic transform	65.97	<u>66.49</u>	**67.07**	+0.52	+1.10
Pixelate	62.70	<u>65.23</u>	**65.60**	+2.53	+2.89
JPEG compression	**62.97**	59.02	<u>59.68</u>	−3.95	−3.29
Saturate	66.68	<u>68.38</u>	**69.32**	+1.70	+2.65
Mean (mCA)	61.06	<u>62.26</u>	**62.55**	+1.20	+1.49
Strongest on	<u>4 / 19</u>	1 / 19	**14 / 19**		

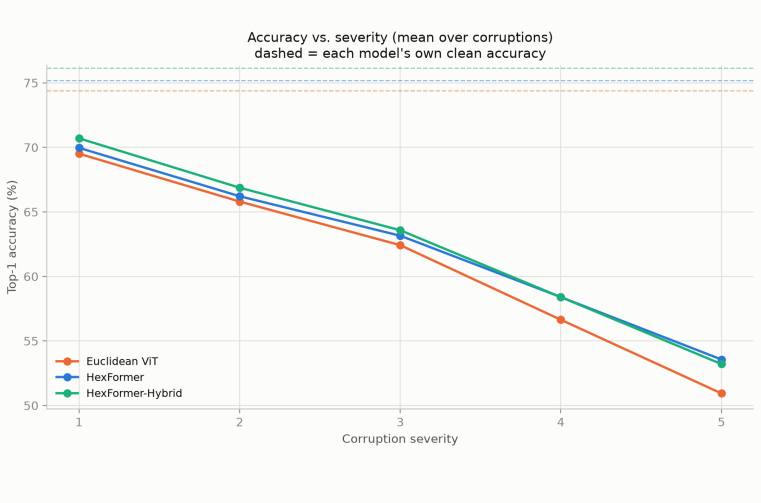
A.4 Centroid vs Exponential Map aggregation

As shown by Mishne et al. (2024), computations in Lorentz space using floating point representation may result in numerical instability. For example, for Float32 values that differ in more than 16 order of magnitude can not be added together since there is no space in the mantissa of the bigger value to store the smaller value. This could lead to invalid data points, i.e., there could be data points that do not satisfy the constraint $\langle x, x\rangle_{\mathcal{L}} = 1/K$. This becomes specially problematic whenever the addition is done with squared values as it happens with the Lorentz norm.

All of this results in a training process where models that use centroid might get NaN/inf values, which ends up collapsing the training run. However, when centroid is changed to the Exponential Map aggregation, this collapse is no longer observed as shown in Figure 14. In order to explain



(a) Top-1 accuracy per corruption type, averaged over the five severities and sorted by Euclidean accuracy.



(b) Accuracy against severity, averaged over the 19 corruption types. Dashed horizontal lines mark each model's own clean accuracy.

Figure 12: Corruption robustness on CIFAR-100-C for ViT-Tiny models, averaged over three seeds. The three models are ordered almost identically across corruption types, with the hyperbolic variants ahead on the blur, weather and digital corruptions on the left of panel (a) and behind on the additive-noise corruptions and JPEG compression on the right.

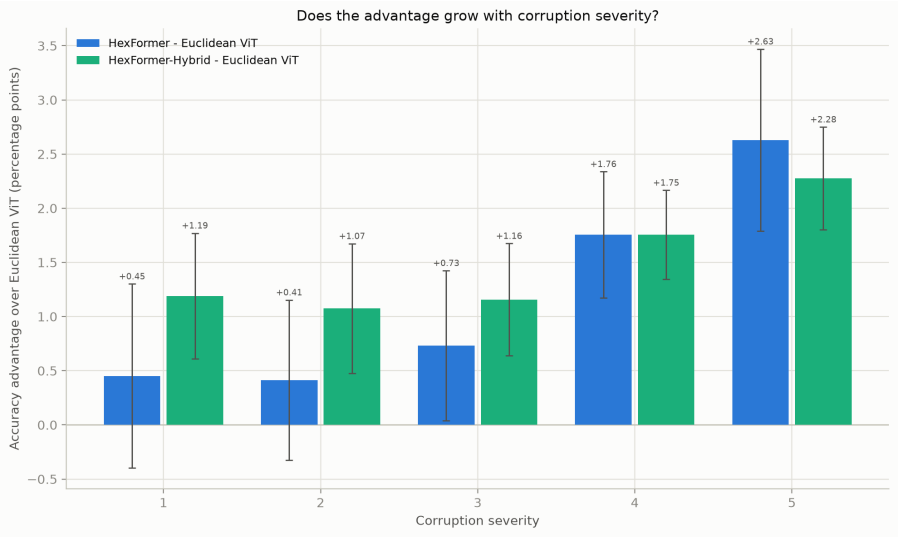


Figure 13: Margin of HexFormer and HexFormer-Hybrid over the Euclidean ViT baseline on CIFAR-100-C, as a function of corruption severity (percentage points, averaged over the 19 corruption types and over three seeds). Error bars are 95% confidence intervals on the per-seed paired margin. The margin grows with severity for both models, from well under one point at severity 1 to above two points at severity 5. At severities 1 and 2 the interval for HexFormer includes zero.

why this happens, a small example will be done for the Lorentz space $\mathcal{L}^1 := \{x \in \mathbb{R}^2 \mid \langle x, x\rangle_{\mathcal{L}} = 1/K,\ x_t > 0\}$ with $K = -1$.

Let's consider the space component from 2 tensors obtained from a *Lorentz linear layer* with values 10^8 for the tensor x and 0 for the tensor y. Then, in order to calculate the time components the Lorentz constraint has to be solved as follows:

$$\begin{aligned} x_t : &\langle x, x\rangle_{\mathcal{L}} = 1/K \Leftrightarrow -x_t^2 + x_s^2 = 1/K \Leftrightarrow x_t = \sqrt{x_s^2 - 1/K} = \sqrt{10^{16} + 1} \approx 10^8 \\ y_t : &\langle y, y\rangle_{\mathcal{L}} = 1/K \Leftrightarrow -y_t^2 + y_s^2 = 1/K \Leftrightarrow y_t = \sqrt{y_s^2 - 1/K} = \sqrt{0 + 1} = 1 \end{aligned} \tag{6}$$

Notice that due to the previously mentioned imprecision in float representation, x_t is approximated to 10^8 instead of taking the value $\sqrt{10^{16} + 1}$. Let's suppose that the values obtained for the attention scores (α_1 and α_2) are 0.5 for both x and y. Then the centroid calculation would be:

$$\mu = \frac{V}{\sqrt{-1/K}|||V||_{\mathcal{L}}|} = \frac{\alpha_1 x + \alpha_2 y}{\sqrt{-1/K}|||\alpha_1 x + \alpha_2 y||_{\mathcal{L}}|} \tag{7}$$

Let's calculate first V and $||V||_{\mathcal{L}}$:

$$\begin{aligned} V = \alpha_1 x + \alpha_2 y = 0.5 \begin{bmatrix} 10^8 \\ 10^8 \end{bmatrix} + 0.5 \begin{bmatrix} 1 \\ 0 \end{bmatrix} = \begin{bmatrix} 5 \cdot 10^7 + 0.5 \\ 5 \cdot 10^7 \end{bmatrix} \approx \begin{bmatrix} 5 \cdot 10^7 \\ 5 \cdot 10^7 \end{bmatrix} \\ ||V||_{\mathcal{L}} = -V_t^2 + V_s^2 = -(5 \cdot 10^7)^2 + (5 \cdot 10^7)^2 = 0 \end{aligned} \tag{8}$$

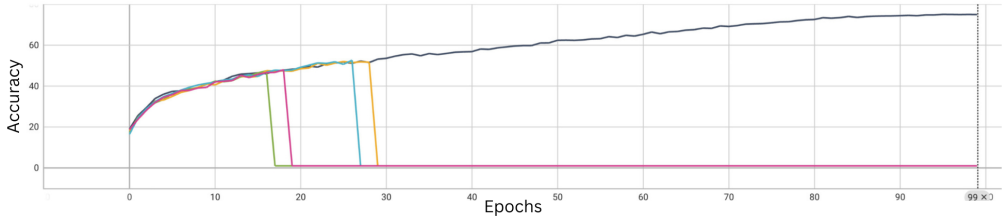
Notice that again due to the representation imprecision, V_t is wrongly calculated which leads to $||V||_{\mathcal{L}} = 0$. This results in a division with a denominator of 0 where NaN/inf values arise:

$$\mu = \frac{V}{\sqrt{-1/K}|||V||_{\mathcal{L}}|} = \frac{V}{0} = NaN/inf \tag{9}$$

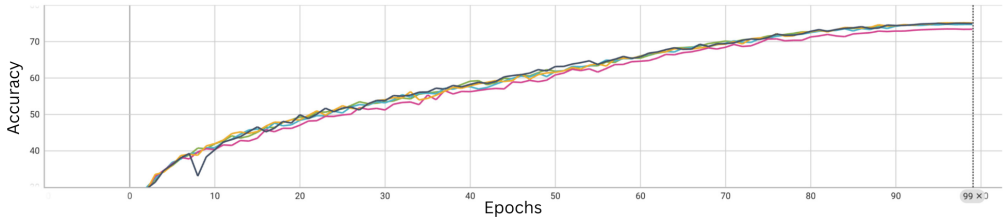
This issue can be side stepped using an ϵ threshold. That way the division can still be done. Since the ϵ threshold needs to be small to not perturb the cases where there is no imprecision/error in the calculation, a small value needs to be used. A standard value used is $\epsilon = 10^{-8}$. However, this leads to the backward pass receiving updates with the order of $10^7/10^{-8} = 10^{15}$ which can easily escalate and lead to model divergence as shown in Figure 14.

A.4.1 Failed Experiments with Centroid

During experiments using the AdamW (Loshchilov, 2017) optimizer, it was observed that HexFormer with centroid-based aggregation often produced NaN/inf values during training. The figure below compares the same training runs over 5 different seeds. For HexFormer with centroid (Figure 14), 4 out of 5 seeds encountered NaNs, causing training to fail. In contrast, HexFormer with exponential map aggregation (ExpAgg, Figure 14b) successfully completed training for all 5 seeds, demonstrating the improved numerical stability of the ExpAgg method.



(a) HexFormer-Tiny with centroid using AdamW on CIFAR-100 for 5 different seeds



(b) HexFormer-Tiny with ExpAgg using AdamW on CIFAR-100 for 5 different seeds

Figure 14: Comparison of the same runs on AdamW on 5 different seeds between HexFormer with centroid and HexFormer with ExpAgg showing failed training experiments with centroid.

A.5 Runtime, Memory, and Limitations

Hyperbolic networks are generally more computationally demanding and can exhibit numerical instability, which affects both training runtime and memory usage (Mettes et al., 2024). Table 19 reports the training runtime of each model. The Exponential Map Aggregation (ExpAgg) introduces around 5% or less additional computational overhead compared to centroid aggregation while improving numerical stability. All the results shown in this section were done on NVIDIA RTX A5000 GPU.

Table 19: Average runtime per iteration (ms) and iterations per second for different Tiny-ViT models on CIFAR-100, measured with `torch.compile` enabled.

Model	Avg. Iteration Time	Iterations per Second
Euclidean ViT	43.77 ms	22.85
HexFormer (Centroid)	111.23 ms	8.99
HexFormer (ExpAgg)	117.42 ms	8.52
HexFormer-Hybrid (Centroid)	118.06 ms	8.47
HexFormer-Hybrid (ExpAgg)	121.26 ms	8.25

Reserved GPU memory for HexFormer and HexFormer-Hybrid is higher than for Euclidean ViT, as reported in Table 21, while allocated memory is comparable across the three models. In both cases the requirement remains manageable on modern GPUs. Training runtime is longer for hyperbolic models, by a factor of roughly 2.7 relative to the Euclidean baseline (Table 19). Fewer epochs are needed to reach a given accuracy: all models were run for 100 epochs in the main experiments, but

HexFormer without warmup already surpasses the no-warmup Euclidean baseline trained for the full 100 epochs after roughly 60 epochs (Table 14, Table 8). This does not offset the higher per-iteration cost, and the total training time of the hyperbolic models remains larger. Inference runtime scales comparably, at roughly 2.3 times the Euclidean baseline (Table 20). Centroid variants are omitted from the inference and memory tables, since their runtime differs from ExpAgg by at most a few percent (Table 19), indicating negligible differences in practical use.

Table 20: Inference runtime for Euclidean and Hyperbolic models on CIFAR-100 and Tiny-ImageNet.

Dataset	Model	Total Inference Time (s)	Time per Image (ms)
CIFAR-100	Euclidean ViT	1.3296 s	0.1329 ms
	HexFormer	3.0991 s	0.3099 ms
	HexFormer-Hybrid	3.0614 s	0.3061 ms
Tiny-ImageNet	Euclidean ViT	4.0127 s	0.8025 ms
	HexFormer	8.2597 s	1.6519 ms
	HexFormer-Hybrid	8.1895 s	1.6379 ms

Table 21: Memory usage (in GB) for Tiny-ViT models on CIFAR-100, measured with `torch.compile` enabled. Values show allocated and reserved GPU memory.

Model	Allocated Memory	Reserved Memory
Euclidean ViT	0.06 GB	1.90 GB
HexFormer	0.06 GB	3.33 GB
HexFormer-Hybrid	0.06 GB	3.33 GB

Table 22: Comparison of forward FLOPs and number of parameters for different Tiny-ViT models on CIFAR-100.

Model	Trainable Params	Forward FLOPs
Euclidean ViT	2.70 M	0.351 GFLOPs
Euclidean HexFormer	2.71 M	0.352 GFLOPs
Hybrid HexFormer	2.69 M	0.352 GFLOPs

Forward FLOPs and the trainable parameter counts for the Tiny-ViT models are shown in Table 22. These were computed using the THOP library, which provides automated estimates of the number of floating-point operations for a given PyTorch model. THOP primarily accounts for standard linear, convolutional, and activation operations, but it does not automatically include more complex or custom operations, such as those used in hyperbolic layers (e.g., exponential maps, logarithmic maps, and Riemannian operations). As a result, the absolute FLOPs values reported here may underestimate the true computational cost for hyperbolic models.

Overall, hyperbolic transformers are more resource-intensive, and this computational overhead is a known challenge in the field. Ongoing research is actively exploring methods to improve the efficiency of hyperbolic architectures while retaining their advantages in numerical stability and hierarchical representation.