
Coneheads: Hierarchy Aware Attention

Albert Tseng
 Cornell University
 albert@cs.cornell.edu

Tao Yu
 Cornell University
 tyu@cs.cornell.edu

Toni J.B. Liu
 Cornell University
 j13499@cornell.edu

Christopher De Sa
 Cornell University
 cdesa@cs.cornell.edu

Abstract

Attention networks such as transformers have achieved state-of-the-art performance in many domains. These networks rely heavily on the dot product attention operator, which computes the similarity between two points by taking their inner product. However, the inner product does not explicitly model the complex structural properties of real world datasets, such as hierarchies between data points. To remedy this, we introduce cone attention, a drop-in replacement for dot product attention based on hyperbolic entailment cones. Cone attention associates two points by the depth of their lowest common ancestor in a hierarchy defined by hyperbolic cones, which intuitively measures the divergence of two points and gives a *hierarchy aware* similarity score. We test cone attention on a wide variety of models and tasks and show that it improves task-level performance over dot product attention and other baselines, and is able to match dot-product attention with significantly fewer parameters. Our results suggest that cone attention is an effective way to capture hierarchical relationships when calculating attention.

1 Introduction

In recent years, attention networks have achieved highly competitive performance in a variety of settings, often outperforming highly-engineered deep neural networks [7, 10, 34]. The majority of these networks use dot product attention, which defines the similarity between two points $u, v \in \mathbb{R}^d$ by their inner product $u^\top v$ [34]. Although dot product attention empirically performs well, it also suffers from drawbacks that limit its ability to scale to and capture complex relationships in large datasets [36, 30]. The most well known of these issues is the quadratic time and memory cost of computing pairwise attention. While many works on attention mechanisms have focused on reducing the computational cost of dot product attention, few have considered the properties of the dot product operator itself [36, 8].

Many real world datasets exhibit complex structural patterns and relationships which may not be well captured by an inner product [33, 19]. For example, NLP tasks often contain hierarchies over tokens, and images may contain clusters over pixels [37, 16]. Motivated by this, we propose a new framework based on hyperbolic entailment cones to compute attention between sets of points [12, 40]. Our attention mechanism, which we dub “cone attention”, utilizes partial orderings defined by hyperbolic cones to better model hierarchical relationships between data points. More specifically, we associate two points by the depth of their lowest common ancestor (LCA) in the cone partial ordering, which is analogous to finding their LCA in a latent tree and captures how divergent two points are.

Cone attention effectively relies on two components: hyperbolic embeddings and entailment cones. Hyperbolic embeddings, which use the underlying geometric properties of hyperbolic space, give low-

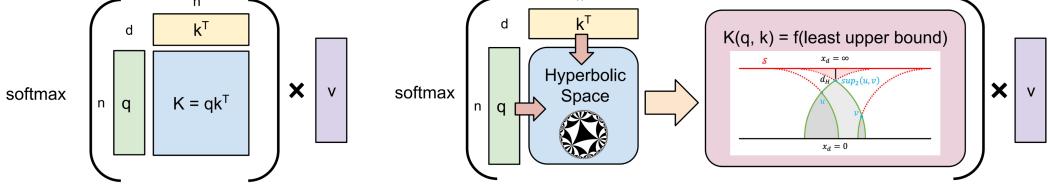


Figure 1: Overview of cone attention vs. dot product attention. In dot product attention (left), similarity scores are calculated with $K = qk^\top$. In cone attention (right), q and k are first projected onto hyperbolic space. Then, pairwise similarity is calculated from the lowest common ancestor of points in the partial ordering defined by entailment cones. Cone attention allows us to explicitly encode notions of hierarchy in attention, and empirically gives better performance than dot product attention.

distortion embeddings of hierarchies that are not possible with Euclidean embeddings [28]. Entailment cones, which rely on geometric cones to define partial orders between points, allow us to calculate explicit relationships between points, such as their LCA [12, 40]. To the best of our knowledge, we are the first to define a hierarchy-aware attention operator with hyperbolic entailment cones.

Functionally, cone attention is a drop-in replacement for dot product attention. We test cone attention in both “classical” attention networks and transformers, and empirically show that cone attention consistently improves end task performance across a variety of NLP, vision, and graph prediction tasks. Furthermore, we are able to match dot product attention with significantly fewer embedding dimensions, resulting in smaller models. To summarize, our contributions are:

- We propose cone attention, a hierarchy aware attention operator that uses the lowest common ancestor of points in the partial ordering defined by hyperbolic entailment cones.
- We evaluate cone attention on NLP, vision, and graph prediction tasks, and show that it consistently outperforms dot product attention and other baselines. For example, we achieve +1 BLEU and +1% ImageNet Top-1 Accuracy on the `transformer_iwslt_de_en` and DeiT-Ti models, respectively.
- We test cone attention at low embedding dimensions and show that we can significantly reduce model size while maintaining performance relative to dot product attention. With cone attention, we can use 21% fewer parameters for the IWSLT’14 De-En NMT task.

2 Background

In this section, we provide background on attention mechanisms, motivate the use of hyperbolic space to embed hierarchies, and describe our choice of entailment cones to encode partial orderings.

2.1 Attention

The attention operator has gained significant popularity as a way to model interactions between sets of tokens [34]. At its core, the attention operator A performs a “lookup” between a single query $q_i \in \mathbb{R}^d$ and a set of keys $k \in \mathbb{R}^{n \times d}$, and aggregates values $v \in \mathbb{R}^{n \times d}$ associated with the keys to “read out” a single value for q_i . Mathematically, this can be represented as

$$A(q_i, k) = C \sum_j (K(q_i, k_j)v_j) \quad (1)$$

where $C \sum_j K(q_i, k_j) = 1$. In “traditional” dot product attention, which has generally superseded “older” attention methods such as Additive Attention and Multiplicative Attention [6, 18],

$$K(q_i, k_j) = \exp\left(\frac{q_i k_j}{\sqrt{d}}\right) \quad C = \frac{1}{\sum_j K(q_i, k_j)} \quad A(q_i, k) = \text{softmax}\left(\frac{q_i k^\top}{\sqrt{d}}\right) v \quad (2)$$

similarity is scored with a combination of cosine similarity and embedding magnitudes.

Existing works have proposed replacing dot product attention to various degrees. A large body of works focus on efficiently computing dot product attention, such as with Random Fourier Features and low-rank methods [26, 36]. These methods generally perform worse than dot product attention, as they are approximations [41]. Some recent works, such as EVA attention, parameterize these approximations and effectively get a larger class of dot-product-esque attention methods [41]. Beyond this, Tsai et al. [32] replace K with compositions of classical kernels. Others extend dot product attention, such as by controlling the “width” of attention with a learned Gaussian distribution or by using an exponential moving average on inputs, although extensions do not usually depend on K [14, 19]. Closer to our work, Gulcehre et al. [13] introduced hyperbolic distance attention, which defines $K(q_i, k_i) = \exp(-\beta d_{\mathbb{H}}(q_i, k_i) - c)$ where $d_{\mathbb{H}}$ is the hyperbolic distance, $\beta \in \mathbb{R}^+$, and $c \in \mathbb{R}$. K can be interpreted as an analog of the distance from q_i to k_i on a latent tree. Finally, in an orthogonal direction, Tay et al. [29] ignore token-token interactions and synthesize attention maps directly from random alignment matrices. Whether token-token interactions are actually needed is outside the scope of this work, and we compare cone attention accordingly.

2.2 Hyperbolic Space

d -dimensional Hyperbolic space, denoted \mathbb{H}_d , is a simply connected Riemannian manifold with constant negative sectional curvature [4]. This negative curvature results in geometric properties that makes hyperbolic space well-suited for embedding tree-like structures [28, 38]. For example, the volume of a hyperbolic ball grows exponentially with respect to its radius; in a tree, the number of leaves grows exponentially with respect to depth. Furthermore, $d_{\mathbb{H}}(u, v) \approx d_{\mathbb{H}}(u, O) + d_{\mathbb{H}}(O, v)$, where O is the origin, which again mirrors a tree where $d_T(u, v) = d_T(u, \text{LCA}(u, v)) + d_T(\text{LCA}(u, v), v)$. Since hyperbolic space cannot be isometrically embedded into Euclidean space, it is usually represented on a subset of Euclidean space by a “model” of \mathbb{H}_d [13]. These models are isometric to each other, and the key differences between them lie in their different parameterizations, which allow for cleaner computations and visualizations for certain tasks.

In this work, we primarily use the Poincaré half-space model, which is the manifold $\mathbb{H}^d = (\mathcal{U}^d, g_u)$ where $\mathcal{U}^d = \{x \in \mathbb{R}^d : x_d > 0\}$, $g_u(x) = g_e/x_d^2$, and g_e is the Euclidean metric [4]. In the Poincaré half-space, “special” points and curves have particularly nice Euclidean forms. Ideal points, or points at infinity, are the points where $x_d = 0$ (the “ x -axis”) and the single point $x_d = \infty$ at which all lines orthogonal to the x -axis converge. Geodesics, the shortest path between two points, are Euclidean semicircles with the origin on the x -axis or vertical rays orthogonal to the x -axis. Horospheres, curves where all normal curves converge at the ideal point, are represented by either an Euclidean ball tangent to the x -axis or a horizontal hyperplane when the ideal point is $x_d = \infty$.

2.3 Entailment Cones

Entailment cones in hyperbolic space were first introduced by Ganea et al. [12] to embed partial orders. The general concept of Ganea’s entailment cones is to capture partial orders between points with membership relations between points and geodesically convex cones rooted at said points. That is, if $u \in$ the cone of v , then $v \prec u$. Ganea’s cones (figure 2 right) are defined on the Poincaré ball by a radial angle function $\psi(r)$, with an ϵ -ball around the origin where cones are undefined [4, 12]. This makes learning complicated models with Ganea’s cones difficult, as optimization on the Poincaré ball is nontrivial and the ϵ -ball negatively impacts embedding initializations [40, 12].

In this work, we instead use the shadow cone construction introduced in [40] and operate on the Poincaré half-space, which makes computing our desired attention function numerically simpler. Shadow cones are defined by shadows cast by points and a single light source \mathcal{S} , and consist of the penumbral and umbral settings (figure 2 left quadrant). In the penumbral setting, \mathcal{S} is a ball of fixed radius and points are points. The shadow and cone of u are both the region enclosed by geodesics through u tangent to \mathcal{S} . In the umbral setting, \mathcal{S} is instead a point, and points are centers of balls of fixed radius. Here, the shadow of u is the region enclosed by geodesics tangent to the ball around u that intersect at \mathcal{S} . However, to preserve transitivity, the cone of u is a subset of the shadow of u (see figure 2). The shadow cone formulation can also be achieved with subset relations between shadows instead of membership relations between points and cones, which may be conceptually clearer.

Infinite-setting Shadow Cones. For penumbral cones, when \mathcal{S} is a ball with center at $x_d = \infty$, \mathcal{S} ’s boundary is a horosphere of user-defined height h . Here, all shadows are defined by intersections of

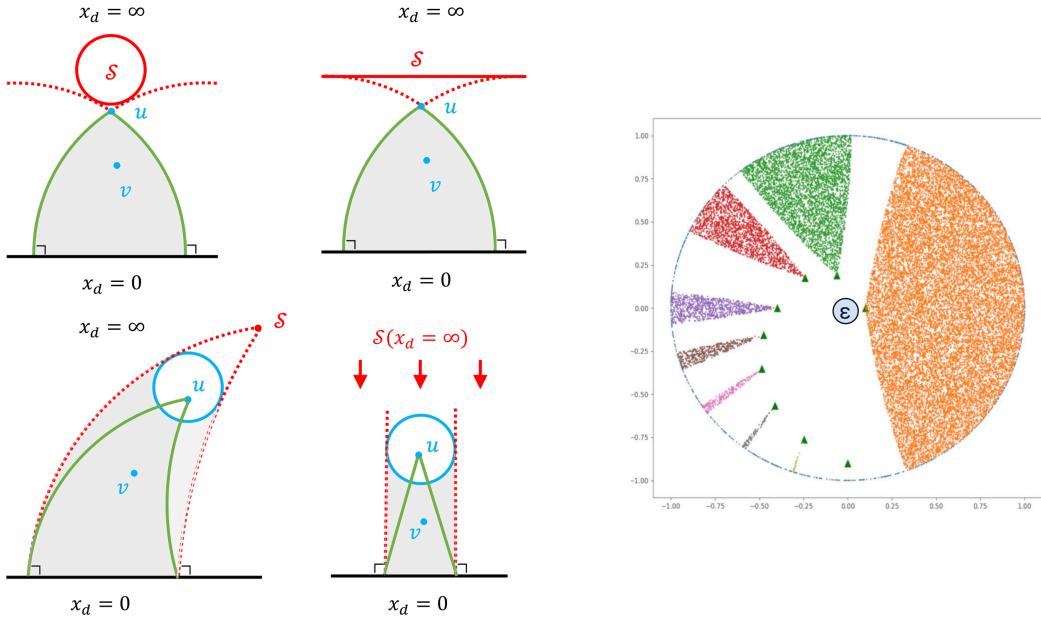


Figure 2: (Left Quadrant) Clockwise from top left: finite setting penumbral cone, infinite setting penumbral cone, infinite setting umbral cone, and finite setting umbral cone. All figures are in \mathbb{H}^2 . Shadows are represented by shaded regions, and cones are enclosed in green. In all figures, $u \prec v$ since v is in the cone of u . (Right) Ganea’s entailment cones, as defined on the Poincaré ball. Note the ϵ -ball where cones are not defined, which makes optimization nontrivial. Figure from [12].

Euclidean semicircles of Euclidean radius h . Notably, the infinite setting penumbral cone construction is similar to Ganea’s cones under an isometry from the Poincaré half-space to the Poincaré ball where S maps to the ϵ -ball [40]. For umbral cones, when S is $x_d = \infty$, shadows are regions bounded by Euclidean lines perpendicular to the x -axis.

Unlike Ganea’s cones and penumbral cones, umbral cones are not geodesically convex [40]. That is, the shortest path between two points in an umbral cone may not necessarily lie in the cone. In a tree, this corresponds to the shortest path between two nodes not being in the subtree of their LCA, which is not possible. Empirically, while still better than dot product attention, umbral attention usually performs worse than penumbral attention.

2.4 Lowest Common Ancestor (LCA) and Least Upper Bound

The LCA of two nodes u, v in a directed acyclic graph is the lowest (deepest in hierarchy) node that is an ancestor of both u and v . Although the two terms are similar, the LCA is *not* the same as the least upper bound of a partial ordering. The least upper bound of two points x, y in a partial ordering, denoted $\sup(x, y)$, is the point p such that $p \preceq x, y$ and $\forall q$ where $q \preceq x, y$, $q \preceq p$. The key difference is that all other upper bounds must precede the least upper bound, while not all ancestors of u and v must also be ancestors of $\text{LCA}(u, v)$. Furthermore, p may not actually exist.

3 Hierarchy Aware Attention

Here, we describe cone attention using the shadow cone construction, and discuss projection functions onto \mathbb{H}^d that allow us to use cone attention within attention networks. All definitions use the infinite-setting shadow cones, and proofs and derivations are in the appendix. For clarity, we refer to Ganea’s entailment cones as “Ganea’s cones” and the general set of cones that captures entailment relations (e.g. Ganea’s cones and shadow cones) as “entailment cones” [12, 40]. Our methods are agnostic entailment cone choice, and can also be used with Ganea’s cones.

3.1 Cone Attention

We wish to associate $u, v \in \mathbb{H}^d$ by their LCA in some latent tree T , which is analogous to finding their LCA, denoted $\sup_2(u, v)$, in the partial ordering defined by entailment cones. Formally,

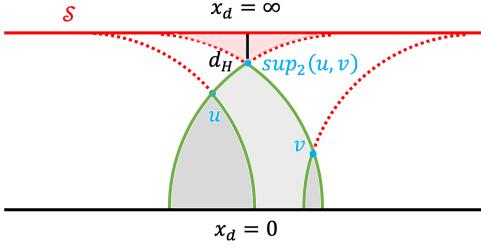


Figure 3: In this example using penumbral cones in H^d , $\sup_2(u, v)$ is the lowest common ancestor of u and v . The red region is the set of points P s.t. $p \in P \preceq u, v$. Of these, $\sup_2(u, v)$ is the lowest point whose cone (light gray) contains both u and v . Here, S is the root of all hierarchies, and points closer to $x_d = 0$ are closer to the ‘‘leaves’’ of hierarchies. If $d = 2$, then $\sup_2(u, v)$ is also the least upper bound of u and v in the partial ordering defined by entailment cones.

$$\sup_2(u, v) = r \left(\arg \max_{C: u, v \in C} d_H(S, r(C)) \right) \quad (3)$$

where $r(C)$ denotes the root of a cone C . This corresponds to finding the cone that is farthest away from S , which is the root of all hierarchies in the shadow cones construction. When $d = 2$, $\sup_2(u, v)$ also has the nice property of being the least upper bound $\sup(u, v)$ in the partial ordering defined by shadow cones. Then, we define the similarity between u and v as

$$K(u, v) = f(d_H(\sup_2(u, v), S)) \quad (4)$$

where f is a user-defined monotonically increasing function. If $K(u, v) > K(u, w)$, then $d_H(\sup_2(u, v), S) > d_H(\sup_2(u, w), S)$, which implies that u and v have a more recent ‘‘ancestor’’ than u and w . Thus, K gives a higher similarity score to points who have a more recent ‘‘ancestor’’ in T . In the infinite-setting shadow cone construction, $\sup_2(u, v)$ is root of the minimum height (literal lowest) cone that contains both u and v , or $\sup_2(u, v) = r(\arg \min_{C: u, v \in C} r(C)_d)$.

Using this, we provide definitions for $K(u, v)$ in the infinite-setting shadow cone construction. Both the umbral and penumbral definitions correspond to the Euclidean height of $\sup_2(u, v)$. In the penumbral setting, when $\sup_2(u, v)$ does not exist, we return the Euclidean height of lowest light source where $\sup_2(u, v)$ exists. x_d denotes the last dimension of x , and $x_{:-1}$ denotes the first $d - 1$ dimensions of x . $\gamma \in \mathbb{R}^+$ corresponds to the softmax ‘‘temperature’’ in attention. In the penumbral setting, $r \in \mathbb{R}^+$ is the user-defined height of the horosphere light source. In the umbral setting, $r \in \mathbb{R}^+$ is the user-defined radius of the ball centered at each point.

Definition 1. Penumbral Attention:

$$K(u, v) = \exp \left(-\gamma \max \left(u_d, v_d, \sqrt{r^2 - \left(\frac{\sqrt{r^2 - u_d^2} + \sqrt{r^2 - v_d^2} - \|u_{:-1} - v_{:-1}\|}{2} \right)^2} \right) \right) \quad (5)$$

when there exists a cone that contains u and v , and

$$K(u, v) = \exp \left(-\gamma \sqrt{\left(\frac{\|u_{:-1} - v_{:-1}\|^2 + u_d^2 - v_d^2}{2\|u_{:-1} - v_{:-1}\|} \right)^2 + v_d^2} \right) \quad (6)$$

otherwise. There exists a cone that contains u and v when

$$\left(\|u_{:-1} - v_{:-1}\| - \sqrt{r^2 - u_d^2} \right)^2 + v_d^2 < r^2 \quad (7)$$

Definition 2. Umbral Attention:

$$K(u, v) = \exp \left(-\gamma \max \left(u_d, v_d, \frac{\|u_{:-1} - v_{:-1}\|}{2 \sinh(r)} + \frac{u_d + v_d}{2} \right) \right) \quad (8)$$

These definitions possess a rather interesting property – when $u_d = v_d$ and K is normalized across a set of vs , cone attention reduces to the Laplacian kernel $K(u_{:-1}, v_{:-1}) = \exp(-\gamma \|u_{:-1} - v_{:-1}\|)$. Since Euclidean space is isomorphic to a horosphere, and u and v are on the same horosphere if $u_d = v_d$, cone attention can also be seen as an extension of the Laplacian kernel [22].

Cone attention and dot product attention both take $O(n^2d)$ time to compute pairwise attention between two sets of n d -dimensional tokens $q, k \in \mathbb{R}^{n \times d}$ [36]. However, cone attention takes more operations than dot product attention, which computes qk^\top . In transformers, our PyTorch cone attention implementations with `torch.compile` were empirically 10-20% slower than dot product attention with `torch.bmm` (cuBLAS)[24] (see section 7.6). `torch.compile` is not optimal, and a raw CUDA implementation of cone attention would likely be faster and narrow the speed gap between the two methods [24].

3.2 Mapping Functions

Here, we discuss mappings from Euclidean space to the Poincaré half-space. These mappings allow us to use cone attention within larger models. The canonical map in manifold learning is the exponential map $\text{Exp}_x(v)$, which maps a vector v from the tangent space of a manifold \mathcal{M} at x onto \mathcal{M} by following the geodesic corresponding to v [27, 4]. In the Poincaré half-space [39],

$$\text{Exp}_x(v)_{:-1} = x_{:-1} + \frac{x_d}{\|v\| / \tanh(\|v\|) - v_d} v_{:-1} \quad \text{Exp}_x(v)_d = \frac{x_d}{\cosh(\|v\|) - v_d \sinh(\|v\|) / \|v\|} \quad (9)$$

While $\text{Exp}_x(v)$ is geometrically well motivated, it suffers from numerical instabilities when $\|v\|$ is very large or small. These instabilities make using the exponential map in complicated models such as transformers rather difficult. Using `fp64` reduces the risk of numerical over/underflows, but `fp64` significantly reduces performance on GPUs, which is highly undesirable [1].

Instead, we use maps of the form $(x_{:-1}, x_d) \rightarrow (x_{:-1}f(x_d), f(x_d))$ that preserve the exponential volume of hyperbolic space while offering better numerics for large-scale optimization. Our use of alternatives to $\text{Exp}_x(v)$ follows Gulcehre et al. [13]’s psuedopolar map onto the Hyperboloid. Geometrically, since n -dimensional Euclidean space is isomorphic to a horosphere in \mathbb{H}^{n+1} , these maps correspond to selecting a horosphere with $f(x_d)$ and then projecting $x_{:-1}$ onto that horosphere [22]. To achieve exponential space as $x_d \rightarrow -\infty$, we use functions f of the form $\exp(\cdot)$.

For the infinite-setting umbral construction, since there is no restriction on where points can go in the Poincaré half-space, we map $x \in \mathbb{R}^d$ to \mathbb{H}^d with $\psi : \mathbb{R}^d \rightarrow \mathbb{H}^d$:

$$\psi(x)_{:-1} = x_{:-1} \exp(x_d) \quad \psi(x)_d = \exp(x_d) \quad (10)$$

For the infinite-setting penumbral construction, since the mapped point cannot enter the light source at height h , we map $x \in \mathbb{R}^d$ to area below the light source with $\xi : \mathbb{R}^d \rightarrow \mathbb{H}^d$

$$\xi(x)_{:-1} = x_{:-1} \frac{h}{1 + \exp(-x)} \quad \xi(x)_d = \frac{h}{1 + \exp(-x)} \quad (11)$$

While ξ is structurally the sigmoid operation, note that $\text{sigmoid}(x) = \exp(-\text{softplus}(-x))$. Since $\text{softplus}(x) \approx x$ for large values of x , ξ preserves the exponential volume properties we seek.

4 Experiments

Here, we present an empirical evaluation of cone attention in various attention networks. For each model we test, our experimental procedure consists of changing K in attention and training a new model from scratch. Unless otherwise noted in the appendix, we use the code and training scripts that the authors of each original model released. We assume released hyperparameters are tuned for dot product attention, as these models were state-of-the-art (SOTA) when new.

4.1 Baselines

Our main baseline is dot product attention, as it is the most commonly used form of attention in modern attention networks. Additionally, we compare cone attention against Gulcehre et al. [13]’s

hyperbolic distance attention and the Laplacian kernel. To the best of our knowledge, few other works have studied direct replacements of the dot product for attention.

Gulcehre et al. [13]’s original hyperbolic distance attention formulation not only computed the similarity matrix K in the Hyperboloid, but also aggregated values v in hyperbolic space by taking the Einstein midpoint with respect to weights α in the Klein model (see appendix for definitions) [27]. However, the Einstein midpoint, defined as

$$m(\alpha, v) = \sum_i \left(\frac{\alpha_i \gamma(v_i)}{\sum_j \alpha_j \gamma(v_j)} \right) \quad (12)$$

where $\gamma(v) = 1/\sqrt{1 - \|v\|^2}$, does not actually depend on how α is computed. That is, α could be computed with Euclidean dot product attention or cone attention and $m(\alpha, v)$ would still be valid. The focus of our work is on computing similarity scores, so we do not use hyperbolic aggregation in our experiments. We test hyperbolic distance attention using both Gulcehre et al. [13]’s original pseudopolar projection onto the Hyperboloid model and with our ψ map onto the Poincaré half-space.

4.2 Models

We use the following models to test cone attention and the various baselines. These models span graph prediction, NLP, and vision tasks, and range in size from a few thousand to almost 250 million parameters. While we would have liked to test larger models, our compute infrastructure limited us from feasibly training billion-parameter models from scratch.

Graph Attention Networks. Graph attention networks (GATs) were first introduced by Veličković et al. [35] for graph prediction tasks. GATs use self-attention layers to compute node-level attention maps over neighboring nodes. The original GAT used a concatenation-based attention mechanism and achieved SOTA performance on multiple transductive and inductive graph prediction tasks [35]. We test GATs on the transductive Cora and inductive multi-graph PPI datasets [20, 15].

Neural Machine Translation (NMT) Transformers. Transformers were first applied to NMT in Vaswani et al. [34]’s seminal transformer paper. We use the fairseq `transformer_iwslt_de_en` architecture to train a German to English translation model on the IWSLT’14 De-En dataset [23, 11]. This architecture contains 39.5 million parameters and achieves near-SOTA performance on the IWSLT’14 De-En task for vanilla transformers [23]. As this model is the fastest transformer to train out of the tested models, we use it for ablations.

Vision Transformers. Vision transformers (ViT) use transformer-like architectures to perform image classification [10]. In a ViT, image patches are used as tokens in a transformer encoder to classify the image. We use the *Data Efficient* Vision Transformer (DeiT) model proposed by FAIR, which uses a student-teacher setup to improve the data efficiency of ViTs [31]. DeiT and ViTs share the same architecture, and the only differences are how they are trained and the distillation token. We train DeiT-Ti models with 5 million parameters on the ImageNet-1K dataset for 300 epochs[31, 9]. We also train cone and dot product attention for 500 epochs, as we observed that training for more iterations improves performance.

Adaptive Input Representations for Transformers. Adaptive input representations were introduced by Baevski and Auli for transformers in language modeling tasks [5]. In adaptive inputs, rarer tokens use lower dimensional embeddings, which serves as a form of regularization. We use the fairseq `transformer_lm_wiki103` architecture (246.9M parameters) and train models on the WikiText-103 language modeling dataset with a block size of 512 tokens [23, 21]. We also test the same architecture without adaptive inputs, which has 520M parameters. This version converges in significantly fewer iterations, allowing us to train such a large model.

Diffusion Transformers. Diffusion transformers (DiT) are diffusion models that replace the U-Net backbone with a transformer [25]. DiTs operate on latent space patches, so we expect there to be less hierarchical information vs. taking image patches. We use DiTs to test cone attention when the data is less hierarchical. We train DiT-B/4 models with 130M parameters on ImageNet-1K [25, 9].

Table 1: Performance of various attention methods across models and tasks. * indicates the model failed to converge or ran into NaN errors. ↑ indicates higher is better, and ↓ indicates lower is better. Cone attention methods (†) generally outperform dot product attention and other baselines. Model default refers to a model’s default attention method, which is dot product attention except for GATs.

Method	NMT IWLST (BLEU ↑)	DeiT-Ti Imagenet Top-1 / 5 (Acc. ↑) 300 Epochs	DeiT-Ti Imagenet Top-1 / 5 (Acc. ↑) 500 Epochs	GAT Cora / PPI (Acc. ↑)
Model Default	34.56	72.05 / 91.17	73.65 / 91.99	0.831 / 0.977
Dot Product	34.56	72.05 / 91.17	73.65 / 91.99	0.834 / 0.985
Penumbral [†]	35.56	72.67 / 91.12	74.34 / 92.38	0.835 / 0.990
Umbral [†]	35.07	73.14 / 91.82	74.46 / 92.54	0.836 / 0.989
$d_{\mathbb{H}} H^n \xi$	32.54	49.19* / 74.68*	—	0.834 / 0.987
$d_{\mathbb{H}}$ Hyperboloid	33.80	0.10* / 0.45*	—	0.13* / 0.989
Laplacian Kernel	34.68	71.25 / 90.84	—	0.823 / 0.986
Method	Adaptive Inputs WikiText-103 0 / 480 Context Window (Ppl. ↓)		Without Adaptive Inputs	DiT-B/4 @ 400K Steps (FID-50K ↓)
Dot Product	20.86 / 19.22		26.62 / 24.73	68.9
Penumbral	20.72 / 19.01		26.44 / 24.31	67.7
Umbral	21.16 / 19.59		27.82 / 26.73	67.6

5 Results

Table 1 summarizes the performance of cone attention across various models and tasks. Both penumbral and umbral attention significantly outperform baselines on the NMT IWSLT and DeiT-Ti Imagenet tasks. For DeiT-Ti, umbral attention achieves 73.14% top-1 and 91.82% top-5 accuracy at 300 epochs and 74.46% top-1 and 92.54% top-5 accuracy at 500 epochs, which matches a distilled DeiT-Ti with more parameters (74.5% / 91.9%) [31]. On the GAT tasks, cone attention again outperforms baselines and achieves Graph Convolutional Network-level performance on the PPI dataset. Interestingly, almost all baselines, including dot product attention, outperform the concatenation-based attention in the original GAT paper. The two GAT tasks also reveal some interesting differences between penumbral and umbral attention. The Cora citation dataset is more tree-like with an average of 2 edges per node and chronological dependencies between nodes (a paper cannot cite a newer paper), while the PPI dataset is closer to a clustered graph with 14.3 edges per node [35, 15, 20]. As noted in [40], umbral cones appear to be better for strict hierarchies, while penumbral cones capture more complicated relationships such as those in the PPI dataset.

The adaptive inputs method regularizes models by reducing d for rare words. We expect rarer words to be closer to the leaves of a word hierarchy, so the adaptive inputs method also acts as a hierarchical prior on the data [37]. We use adaptive inputs to test cone attention when combined with other hierarchical priors. Penumbral attention outperforms dot product attention with or without adaptive inputs, but the gap between the two methods is larger without adaptive inputs. On the other hand, umbral attention performs worse than dot product attention and penumbral attention on the WikiText103 task, with or without adaptive inputs. Since umbral cones are not geodesically convex, which means that the shortest path between two points in a cone may not necessarily lie entirely in that cone, we suspect that convexity is important for the WikiText-103 language modeling task. In the DiT model, patches are taken at the latent space-level, which we suspect gives less hierarchical information than when patches are taken from an input image [25]. Here, cone attention still outperforms dot product attention, but to a lesser degree. This mirrors our expectation that the DiT model does not benefit as much from hierarchical attention, and verifies that in such cases, using cone attention does not hurt performance. Interestingly, umbral cones slightly outperform penumbral cones in the DiT-B/4 task, which may be a result of patches being over the latent space, and not the input image.

5.1 Effect of Mapping Functions

Table 2 compares how ψ and ξ compare to the exponential map and psuedopolar map (section 3.2) on the NMT IWSLT task. Here, we use the exponential map at the origin of H^d , $O = (0, 0, \dots, 1)$. To compare $\text{Exp}_O(v)$ to ψ , we first take $\text{Exp}_O(v)$ and then project the point onto the boundary of

Table 2: Comparison of various mappings from Euclidean space to hyperbolic models for the NMT IWSLT task (BLEU scores, higher is better). The exponential map generally performs worse than ψ and the pseudopolar map. While ψ also performs worse than the pseudopolar map, table 1 indicates that the pseudopolar map is less numerically stable than ψ .

Method	$\text{Exp}_O(v) \rightarrow \mathbb{H}^d$	$\xi \rightarrow \mathbb{H}^d$	$\psi \rightarrow \mathbb{H}^d$	Pseudopolar \rightarrow Hyperboloid
Penumbral	35.12	35.56	-	-
Umbral	34.63	-	35.07	-
$d_{\mathbb{H}}$	30.59	-	32.54	33.80

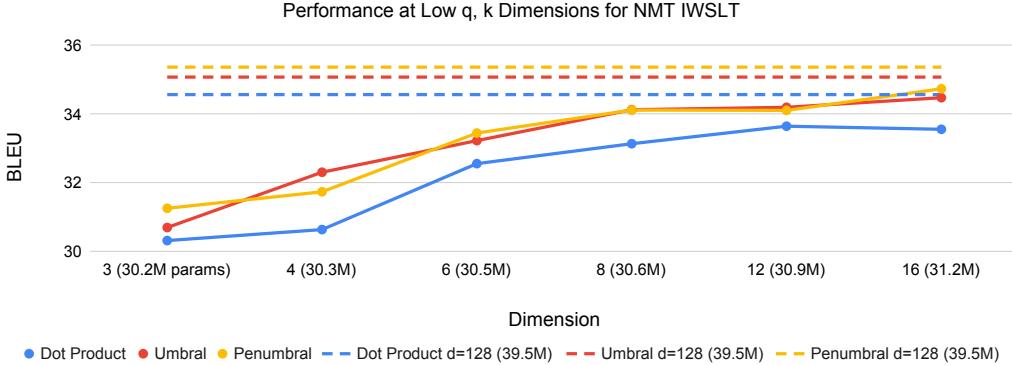


Figure 4: Performance of cone and dot product attention at low dimensions on NMT IWSLT. The base architecture uses 128 dimensions, and cone attention is able to match dot product attention with only 16 dimensions. For this model, this allows us to use 21% fewer parameters to reach performance parity, indicating that cone attention more efficiently captures hierarchical information.

L if $\text{Exp}_O(v)$ is in L . In the infinite penumbral setting, this corresponds to taking $\text{Exp}_O(v)_d = \min(\text{Exp}_O(v)_d, h)$. ψ and ξ generally perform much better than taking the exponential map at the origin O , which suggests that ψ and ξ have better optimization properties. For $d_{\mathbb{H}}$ attention, Gulcehre et al.’s pseudopolar map slightly outperforms ξ . However, table 1 indicates that outside of this specific task, using the pseudopolar map and Hyperboloid is less numerically stable.

5.2 Attention Efficiency and Model Size

Figure 4 shows the performance of cone attention vs. dot product attention at low *token* (q, k) embedding dimensions (d from before) on the NMT IWSLT task. Both umbral and penumbral attention are able to achieve significantly better performance than dot product attention in this regime, matching dot product attention at $d = 128$ with only $d = 16$. For this model, using 16 dimensions reduces the number of parameters from 39.5M to 31.2M. Table 3 shows the performance of DeiT-Ti at $d = 16$. Here, penumbral attention at $d = 16$ is able to outperform dot product attention at $d = 64$, again indicating that cone attention is able to more efficiently capture hierarchies.

5.3 Sensitivity to Initialization

Hyperbolic embeddings are known to be sensitive to initialization [12, 40]. To test cone attention’s sensitivity to initialization, we trained 5 seeds for the IWSLT De2En task for cone attention and dot

Table 3: Performance of DeiT-Ti at 64 (default) and 16 dimensions, 300 epochs.

Method	d = 64	d = 16
Dot Product	72.05 / 91.17	71.29 / 90.54
Penumbral	72.67 / 91.12	72.25 / 91.20
Umbral	73.14 / 91.82	71.67 / 90.71

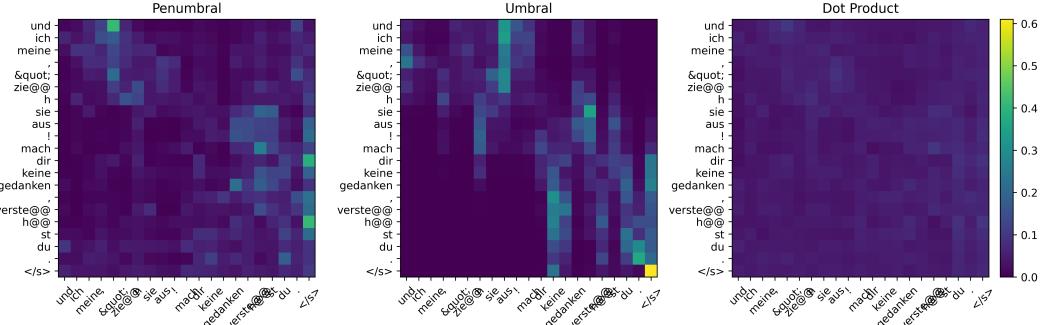


Figure 5: Attention heatmaps from an attention head in a trained IWSLT De2En translation model for the tokenized validation sequence “*Und ich meine, "Zieh sie aus! Mach dir keine gedanken, verstehst du.*”, which translates to “*I'm like, "Get it off! Don't worry about it, you know.*” The cone attention heatmaps (left and center) have a clear distinction between the two parts of the sequence separated by “!”, whereas the dot product heatmap (right) does not have a clear separation.

product attention. Dot product had an average BLEU score of 34.59 and standard deviation 0.12, penumbral achieved 35.41 ± 0.14 , and umbral achieved 35.03 ± 0.30 . There was one outlier in the umbral attention trials, and with the outlier removed umbral achieved 35.16 ± 0.09 . The cone attention methods appear to have slightly higher variance than dot product attention, but not significantly so.

5.4 Attention Heatmaps

A natural question arises about what cone attention methods actually learn. Figure 5 shows heatmaps from an attention head in a trained IWSLT De2En translation model. The heatmaps for penumbral and umbral attention show clearer separation than the dot product attention heatmap. Furthermore, the separation in the two cone attention methods happens at the exclamation mark, a natural segmentation of the sequence into two parts. Intuitively, cone attention can be seen as an attention method that satisfies certain “logical constraints,” such as “if $z \prec y$ and $y \prec x$, then $z \prec x$,” which leads to relations between attention scores. For example, if $K(x, y)$ and $K(y, z)$ are both high, then $K(x, z)$ should also be relatively high in cone attention. In dot product attention, this is not guaranteed. If $x = (1, 1, 0, 0)$, $y = (10, 10, 10, 10)$, and $z = (0, 0, 1, 1)$, then $\langle x, y \rangle = \langle y, z \rangle = 20$, but $\langle x, z \rangle = 0$. We suspect this is a reason why cone attention methods show better separation than dot product attention, which can aid performance.

6 Conclusion

We introduce cone attention, a *hierarchy-aware* method for calculating attention. Cone attention relies on entailment cones and the geometric properties of hyperbolic space to capture complex structural patterns that dot product attention does not explicitly model. We test cone attention in a variety of attention networks ranging from a few thousand to a few hundred million parameters, and achieve consistent performance improvements in NLP, vision, and graph prediction tasks over dot product attention and baselines. Cone attention also matches dot product attention with significantly fewer embedding dimensions, opening the potential for smaller models. These results suggest that cone attention is an effective way to encode hierarchical relationships in attention, and can potentially improve task-level performance in a wide variety of models and task domains.

Future Work. It remains to be seen how cone attention scales to very large models. Beyond this, [12] and [40] suggest that hyperbolic embeddings are sensitive to initialization, which implies that different transformer weight initializations may affect cone attention.

Acknowledgements

This work was supported by NSF Award IIS-2008102. Compute resources were provided by the Cornell G2 Cluster.

References

- [1] NVIDIA A100 Tensor Core GPU Architecture. Technical report, NVIDIA.
- [2] Noga Alon, Troy Lee, Adi Shraibman, and Santosh Vempala. The approximate rank of a matrix and its algorithmic applications. pages 675–684, 06 2013. doi: 10.1145/2488608.2488694.
- [3] Noga Alon, Shay Moran, and Amir Yehudayoff. Sign rank versus vc dimension. In *Conference on Learning Theory*, pages 47–80. PMLR, 2016.
- [4] James W. Anderson. *Hyperbolic geometry*. Springer, 2007.
- [5] Alexei Baevski and Michael Auli. Adaptive input representations for neural language modeling. *arXiv preprint arXiv:1809.10853*, 2018.
- [6] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*, 2014.
- [7] Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901, 2020.
- [8] Tri Dao, Dan Fu, Stefano Ermon, Atri Rudra, and Christopher Ré. Flashattention: Fast and memory-efficient exact attention with io-awareness. *Advances in Neural Information Processing Systems*, 35:16344–16359, 2022.
- [9] 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*, pages 248–255, 2009. doi: 10.1109/CVPR.2009.5206848.
- [10] 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.
- [11] Marcello Federico, Sebastian Stüker, and François Yvon, editors. *Proceedings of the 11th International Workshop on Spoken Language Translation: Evaluation Campaign*, Lake Tahoe, California, December 4-5 2014. URL <https://aclanthology.org/2014.iwslt-evaluation.0>.
- [12] Octavian Ganea, Gary Bécigneul, and Thomas Hofmann. Hyperbolic entailment cones for learning hierarchical embeddings. In *International Conference on Machine Learning*, pages 1646–1655. PMLR, 2018.
- [13] 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. *arXiv preprint arXiv:1805.09786*, 2018.
- [14] Maosheng Guo, Yu Zhang, and Ting Liu. Gaussian transformer: A lightweight approach for natural language inference. *Proceedings of the AAAI Conference on Artificial Intelligence*, 33(01):6489–6496, Jul. 2019. doi: 10.1609/aaai.v33i01.33016489. URL <https://ojs.aaai.org/index.php/AAAI/article/view/4614>.
- [15] Will Hamilton, Zhitao Ying, and Jure Leskovec. Inductive representation learning on large graphs. *Advances in neural information processing systems*, 30, 2017.
- [16] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. Mask r-cnn. In *Proceedings of the IEEE international conference on computer vision*, pages 2961–2969, 2017.
- [17] Troy Lee and Adi Shraibman. An approximation algorithm for approximation rank. In *2009 24th Annual IEEE Conference on Computational Complexity*, pages 351–357. IEEE, 2009.
- [18] Minh-Thang Luong, Hieu Pham, and Christopher D Manning. Effective approaches to attention-based neural machine translation. *arXiv preprint arXiv:1508.04025*, 2015.

- [19] Xuezhe Ma, Chunting Zhou, Xiang Kong, Junxian He, Liangke Gui, Graham Neubig, Jonathan May, and Luke Zettlemoyer. Mega: moving average equipped gated attention. *arXiv preprint arXiv:2209.10655*, 2022.
- [20] Andrew Kachites McCallum, Kamal Nigam, Jason Rennie, and Kristie Seymore. Automating the construction of internet portals with machine learning. *Information Retrieval*, 3(2):127–163, Jul 2000. ISSN 1573-7659. doi: 10.1023/A:1009953814988. URL <https://doi.org/10.1023/A:1009953814988>.
- [21] Stephen Merity, Caiming Xiong, James Bradbury, and Richard Socher. Pointer sentinel mixture models. *arXiv preprint arXiv:1609.07843*, 2016.
- [22] Hee Oh. Euclidean traveller in hyperbolic worlds. *arXiv preprint arXiv:2209.01306*, 2022.
- [23] Myle Ott, Sergey Edunov, Alexei Baevski, Angela Fan, Sam Gross, Nathan Ng, David Grangier, and Michael Auli. fairseq: A fast, extensible toolkit for sequence modeling. *arXiv preprint arXiv:1904.01038*, 2019.
- [24] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. Pytorch: An imperative style, high-performance deep learning library. *Advances in neural information processing systems*, 32, 2019.
- [25] William Peebles and Saining Xie. Scalable diffusion models with transformers. *arXiv preprint arXiv:2212.09748*, 2022.
- [26] Hao Peng, Nikolaos Pappas, Dani Yogatama, Roy Schwartz, Noah A Smith, and Lingpeng Kong. Random feature attention. *arXiv preprint arXiv:2103.02143*, 2021.
- [27] Wei Peng, Tuomas Varanka, Abdelrahman Mostafa, Henglin Shi, and Guoying Zhao. Hyperbolic deep neural networks: A survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 44(12):10023–10044, 2021.
- [28] Frederic Sala, Chris De Sa, Albert Gu, and Christopher Ré. Representation tradeoffs for hyperbolic embeddings. In *International conference on machine learning*, pages 4460–4469. PMLR, 2018.
- [29] Yi Tay, Dara Bahri, Donald Metzler, Da-Cheng Juan, Zhe Zhao, and Che Zheng. Synthesizer: Rethinking self-attention for transformer models. In *International conference on machine learning*, pages 10183–10192. PMLR, 2021.
- [30] Yi Tay, Mostafa Dehghani, Dara Bahri, and Donald Metzler. Efficient transformers: A survey. *ACM Computing Surveys*, 55(6):1–28, 2022.
- [31] Hugo Touvron, Matthieu Cord, Matthijs Douze, Francisco Massa, Alexandre Sablayrolles, and Hervé Jégou. Training data-efficient image transformers & distillation through attention. In *International conference on machine learning*, pages 10347–10357. PMLR, 2021.
- [32] Yao-Hung Hubert Tsai, Shaojie Bai, Makoto Yamada, Louis-Philippe Morency, and Ruslan Salakhutdinov. Transformer dissection: An unified understanding for transformer’s attention via the lens of kernel. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 4344–4353, Hong Kong, China, November 2019. Association for Computational Linguistics. doi: 10.18653/v1/D19-1443. URL <https://aclanthology.org/D19-1443>.
- [33] Albert Tseng, Jennifer J. Sun, and Yisong Yue. Automatic synthesis of diverse weak supervision sources for behavior analysis. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2211–2220, June 2022.
- [34] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

- [35] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Lio, and Yoshua Bengio. Graph attention networks. *arXiv preprint arXiv:1710.10903*, 2017.
- [36] Sinong Wang, Belinda Z Li, Madian Khabsa, Han Fang, and Hao Ma. Linformer: Self-attention with linear complexity. *arXiv preprint arXiv:2006.04768*, 2020.
- [37] Hanqi Yan, Lin Gui, and Yulan He. Hierarchical interpretation of neural text classification. *Computational Linguistics*, 48(4):987–1020, 2022.
- [38] Tao Yu and Christopher M De Sa. Numerically accurate hyperbolic embeddings using tiling-based models. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d’Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019. URL https://proceedings.neurips.cc/paper_files/paper/2019/file82c2559140b95ccda9c6ca4a8b981f1e-Paper.pdf.
- [39] Tao Yu and Christopher M De Sa. Representing hyperbolic space accurately using multi-component floats. *Advances in Neural Information Processing Systems*, 34:15570–15581, 2021.
- [40] Tao Yu, Toni J.B. Liu, Albert Tseng, and Christopher De Sa. Shadow cones: Unveiling partial orders in hyperbolic space. *arXiv preprint*, 2023.
- [41] Lin Zheng, Jianbo Yuan, Chong Wang, and Lingpeng Kong. Efficient attention via control variates. *arXiv preprint arXiv:2302.04542*, 2023.

7 Appendix

7.1 Additional Hyperbolic Manifolds and Maps

7.1.1 Pseudopolar Coordinates and the Hyperboloid Model

The pseudopolar map π used in [13] is given by

$$\pi(x)_{:-1} = x_{:-1} \sinh(x_d) \quad \pi(x)_d = \cosh(x_d) \quad (13)$$

using the notation from the main text.

The hyperboloid model \mathcal{H}^d is a model of \mathbb{H}^d in the $d + 1$ dimensional Minkowski space. Formally,

$$\mathcal{H}^d = \{x \in \mathbb{R}^{d+1} | \langle x, x \rangle_M = -1, x_{d+1} > 0\} \quad (14)$$

where

$$\langle q, k \rangle_M = \left(\sum_{i=1}^n q_i k_i \right) - q_{d+1} k_{d+1} \quad (15)$$

The distance metric on \mathcal{H}^d is given by $d_{\mathbb{H}}(q, k) = \text{arcosh}(-\langle q, k \rangle_M)$. We refer the reader to [13] and [4] for more information on the Hyperboloid model.

7.1.2 Klein Model

The Klein model used in [13] for Einstein aggregation is given by $\mathcal{K}^d = \{x \in \mathbb{R}^n | \|x\| < 1\}$. A point in \mathcal{K}^d can be obtained from a point in \mathcal{H}^d by taking the projection

$$\pi_{\mathcal{H}^d \rightarrow \mathcal{K}^d}(x)_{i \leq d} = \frac{x_i}{x_{d+1}} \quad (16)$$

with inverse

$$\pi_{\mathcal{K}^d \rightarrow \mathcal{H}^d}(x) = \frac{(x, 1)}{\sqrt{1 - \|x\|^2}} \quad (17)$$

The distance in \mathcal{K}^d can be computed with $d_{\mathbb{H}}(q, k) = \text{arcosh}(-\langle \pi_{\mathcal{K}^d \rightarrow \mathcal{H}^d}(q), \pi_{\mathcal{K}^d \rightarrow \mathcal{H}^d}(k) \rangle_M)$. We refer the reader to [13] and [4] for more information on the Klein model.

7.1.3 Poincaré Ball Model and Ganea's Cones

The Poincaré ball model is closely related to the Poincaré half-space, and is defined as the manifold (\mathcal{B}^d, g_p) , where $\mathcal{B}^d = \{x \in \mathbb{R}^d | \|x\| < 1\}$ and

$$g_p(x) = \left(\frac{2}{1 - \|x\|^2} \right)^2 g_e. \quad (18)$$

The distance between two points q and k on the Poincaré ball is given by

$$d_{\mathbb{H}}(q, k) = \text{arcosh} \left(1 + 2 \frac{\|q - k\|^2}{(1 - \|q\|^2)(1 - \|k\|^2)} \right) \quad (19)$$

When $\|x\| \rightarrow 1$, $d_{\mathbb{H}}$ changes very quickly, which makes optimization difficult near the boundary of the Poincaré ball.

Ganea's cones are defined by a radial angle function ψ . For a point x , the angle of the cone at x is given by

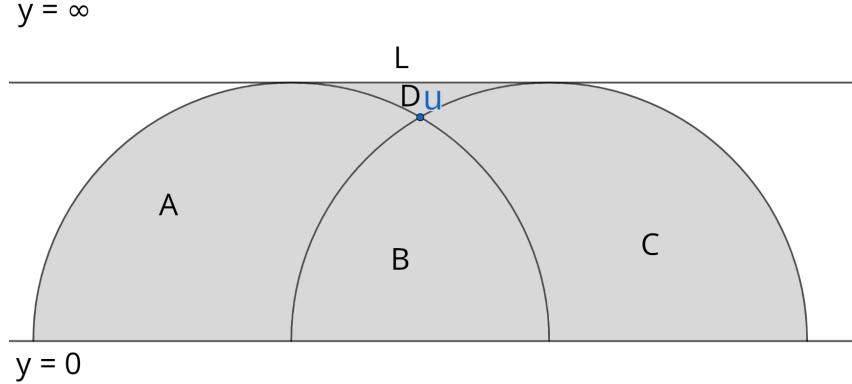


Figure 6: The gray area ($A \cup B \cup C \cup D$) is the region of all points that share a cone with u . B is the cone of u , and D is the region of points whose cones contain u (ancestors of u).

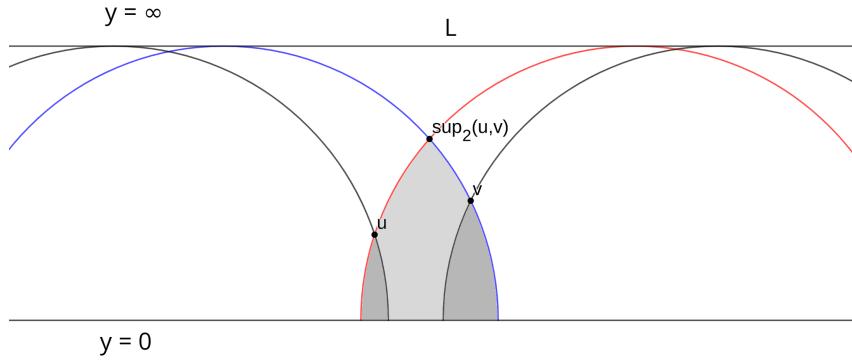


Figure 7: If $v_x \geq u_x$, then $\text{sup}_2(u, v)$ is the intersection of the “right” geodesic of u (red) and the “left” geodesic of v (blue).

$$\psi(x) = \arcsin \left(K \frac{1 - \|x\|^2}{\|x\|} \right) \quad (20)$$

where K is a user-chosen constant that defines the size of the ϵ -ball. Clearly, cones do not exist for small x , giving the ϵ -ball.

7.2 Penumbral Attention Derivation

Recall the definition of Penumbral Attention in the infinite setting shadow cone construction:

$$K(u, v) = \exp \left(-\gamma \max \left(u_d, v_d, \sqrt{r^2 - \left(\frac{\sqrt{r^2 - u_d^2} + \sqrt{r^2 - v_d^2} - \|u_{:-1} - v_{:-1}\|}{2} \right)^2} \right) \right) \quad (21)$$

when there exists a cone that contains u and v , and

$$K(u, v) = \exp \left(-\gamma \sqrt{\left(\frac{\|u_{:-1} - v_{:-1}\|^2 + u_d^2 - v_d^2}{2\|u_{:-1} - v_{:-1}\|} \right)^2 + v_d^2} \right) \quad (22)$$

otherwise. There exists a cone that contains u and v when

$$\left(\|u_{:-1} - v_{:-1}\| - \sqrt{r^2 - u_d^2} \right)^2 + v_d^2 < r^2 \quad \text{or} \quad \|u_{:-1} - v_{:-1}\| \leq \sqrt{r^2 - u_d^2} \quad (23)$$

Below, we give a derivation of this definition in the infinite penumbral cone construction. From theorem 2, $\sup_2(u, v)$ is the lowest cone that contains u and v in the plane containing u, v , and the ideal point of $\tilde{\mathcal{S}}$. This lets us derive the height of $\sup_2(u, v)$ in that plane, which is given by the intersection of Euclidean semicircle geodesics with Euclidean radius r .

First, we check if there exists a cone that contains u and v in this plane. Note that the region of points v s.t. \exists a cone containing u and v is the region contained by the two geodesics forming the cone of u (see figure 6). Assume $u_x = 0$. Then, in the infinite setting penumbral construction, these geodesics are the two semicircles centered at $(\pm \sqrt{r^2 - u_y^2}, 0)$ with radius r . We can check if a point is in this region by checking if it is in the quarter circle bounded by $(-\sqrt{r^2 - u_y^2}, 0)$ and the arc from $(-\sqrt{r^2 - u_y^2} - r, 0)$ to $(-\sqrt{r^2 - u_y^2}, r)$, the quarter circle bounded by $(\sqrt{r^2 - u_y^2}, 0)$ and the arc from $(\sqrt{r^2 - u_y^2} r, 0)$ to $(\sqrt{r^2 - u_y^2}, r)$, or the rectangular region between the two. This is given by

$$\left(\left(\left(v_x - \sqrt{r^2 - u_y^2} \right)^2 + (v_y - 0)^2 < r^2 \right) \wedge v_x > \sqrt{r^2 - u_y^2} \right) \vee v_x \leq \sqrt{r^2 - u_y^2} \quad (24)$$

which reduces to equation 23 since $u_x = 0 \implies v_x = \|u_{:-1} - v_{:-1}\|$.

Now, we derive the height of $\sup_2(u, v)$ when there exists a cone containing u and v . Assume WLOG that $v_x \geq u_x$, and normalize u and v s.t. $u_x = -\|u_{:-1} - v_{:-1}\|/2$ and $v_x = \|u_{:-1} - v_{:-1}\|/2$. If $u \not\prec v$ and $v \not\prec u$, $\sup_2(u, v)$ is intersection of the “right” geodesic of u and the “left” geodesic of v (see figure 7). The center of the right geodesic of u is given by $(u_x + \sqrt{r^2 - u_y^2}, 0)$, and the center of the left geodesic of v is given by $(v_x - \sqrt{r^2 - v_y^2})$. Since $\sup_2(u, v)$ is equidistant from these two centers, $\sup_2(u, v)_x = (\sqrt{r^2 - u_y^2} - \sqrt{r^2 - v_y^2})/2$. Then,

$$\sup_2(u, v)_y = \sqrt{r^2 - \left(\frac{\sqrt{r^2 - u_y^2} - \sqrt{r^2 - v_y^2}}{2} - (u_x + \sqrt{r^2 - u_y^2}) \right)^2} \quad (25)$$

$$= \sqrt{r^2 - \left(\frac{\|u_{:-1} - v_{:-1}\| - \sqrt{r^2 - u_y^2} - \sqrt{r^2 - v_y^2}}{2} \right)^2} \quad (26)$$

$$= \sqrt{r^2 - \left(\frac{\sqrt{r^2 - u_y^2} + \sqrt{r^2 - v_y^2} - \|u_{:-1} - v_{:-1}\|}{2} \right)^2} \quad (27)$$

If $u \prec v$ or $v \prec u$, then $\sup_2(u, v)$ is equal to u or v , respectively. In this situation, equation 27 is less than u_y or v_y , respectively. Thus, we take the max of u_y, v_y , and equation 27 to get equation 21.

When there does not exist a cone containing u and v , we return the height of the lowest light source s.t. there exists such a cone. This corresponds to the height of the highest point in the extended geodesic through u and v . Since geodesics are Euclidean semicircles, this height is the radius of the Euclidean semicircle through u and v . Let the center of this semicircle be $(x, 0)$, and normalize u and v s.t. $u_x = 0$ and $v_x = \|u_{:-1} - v_{:-1}\|$. Then, we have

$$x^2 + u_y^2 = (v_x - x)^2 + v_y^2 \quad (28)$$

$$2v_x x = v_x^2 + v_y^2 - u_y^2 \quad (29)$$

$$2\|u_{:-1} - v_{:-1}\| x = \|u_{:-1} - v_{:-1}\|^2 + v_y^2 - u_y^2 \quad (30)$$

$$x = \frac{\|u_{:-1} - v_{:-1}\|^2 + v_y^2 - u_y^2}{2\|u_{:-1} - v_{:-1}\|} \quad (31)$$

The radius of the semicircle through u and v is then

$$\sqrt{\left(\frac{\|u_{:-1} - v_{:-1}\|^2 + v_y^2 - u_y^2}{2\|u_{:-1} - v_{:-1}\|}\right)^2 + u_d^2} \quad (32)$$

It is easy to verify that this is symmetric in u and v and gives equation 22. Furthermore, it is also easy to verify that when

$$\left(\|u_{:-1} - v_{:-1}\| - \sqrt{r^2 - u_d^2}\right)^2 + v_d^2 = r^2 \quad (33)$$

the in-cone and out-of-cone heights are both r , and equations 21 and 22 both equal $\exp(-\gamma r)$, making K continuous with respect to the positions of u and v .

7.3 Umbral Attention Derivation

Recall the definition of Umbral Attention in the infinite setting shadow cone construction:

$$K(u, v) = \exp\left(-\gamma \max\left(u_d, v_d, \frac{\|u_{:-1} - v_{:-1}\|}{2 \sinh(r)} + \frac{u_d + v_d}{2}\right)\right) \quad (34)$$

As shown in [40], cone regions for infinite setting umbral cones are given by Euclidean triangles with angle apertures of $2 \arctan(\sinh(r))$. WLOG, assume that $v_x \geq u_x$ and normalize u and v s.t. $u_x = 0$ and $v_x = \|u_{:-1} - v_{:-1}\|$. When $u \prec v$ and $v \prec u$, $\sup_2(u, v)$ is given by the intersection of the line with slope $1/\sinh(r)$ through u and the line with slope $-1/\sinh(r)$ through v . This gives

$$y - u_y = \frac{x - u_x}{\sinh(r)} \quad (35)$$

$$y - v_y = \frac{x + v_x}{\sinh(r)} \quad (36)$$

$$2y = \frac{v_x - u_x}{\sinh(r)} + v_y + u_y \quad (37)$$

$$y = \frac{\|u_{:-1} - v_{:-1}\|}{2 \sinh(r)} + \frac{v_y + u_y}{2} \quad (38)$$

where $\sup_2(u, v) = (x, y)$. When $u \prec v$ or $v \prec u$, equation 38 is less than u_y or v_y , respectively. Thus, we take the max of u_y, v_y , and equation 38, giving us equation 34. When $v_x < u_x$, $\sup_2(u, v)$ becomes the intersection of the line with slope $1/\sinh(r)$ through v and the line with slope $-1/\sinh(r)$ through u , which gives us the same result.

7.4 Theorems

Theorem 1. *In the infinite shadow cone construction, the cone with root farthest away from \mathcal{S} that contains u and v is the minimum height cone that contains u and v .*

Proof. We prove the umbral and penumbral cases separately.

Penumbral Cones: The distance from boundary of \mathcal{S} to a point x is the length of geodesic orthogonal to S through x . In the infinite penumbral construction, S is a horosphere, so this geodesic is the vertical Euclidean line from x to \mathcal{S} . Clearly, the longer this line is, the lower x is.

Umbral Cones: Here \mathcal{S} is a point, and geodesics through \mathcal{S} are vertical Euclidean lines orthogonal to the “ x -axis” (using the definition of “ x -axis” from the main text). As with the penumbral case, since the geodesic from a point x to \mathcal{S} is vertical Euclidean line, the longer this line is the lower x is. \square

Theorem 2. *In the infinite shadow cone construction, the root of the minimum height cone that contains u and v lies on the plane containing u , v , and \mathcal{S} (or the ideal point of \mathcal{S}).*

Proof. We prove the umbral and penumbral cases separately.

Penumbral Cones: Consider the region of points x s.t. $x \prec u$. This is the region of geodesics through u that intersect \mathcal{S} and is axially symmetric around the vertical Euclidean line A_u through u [40]. Denote this region \mathcal{C}_u and its boundary \mathcal{B}_u . Note that the ideal point of \mathcal{S} , $x_d = \infty$, is the intersection of vertical line geodesics, so all planes through u and $x_d = \infty$ contain A_u . Since \mathcal{C}_u is axially symmetric w.r.t. A_u , it follows that it is reflection-symmetric across such planes.

Now, consider the intersection of \mathcal{C}_u and \mathcal{C}_v , which has boundary $\mathcal{B}_{u \cap v}$. $\mathcal{C}_u \cap \mathcal{C}_v$ is clearly reflection-symmetric along the plane P containing u , v , and $x_d = \infty$. As such, for any plane P' orthogonal to P , all points on $\mathcal{B}_{u \cap v} \cap P'$ are either on \mathcal{B}_u or \mathcal{B}_v but not both. Since the geodesics that form \mathcal{B}_u are monotonically increasing in height in the direction away from A_u (and likewise for v), the minimum height cone that contains u and v must have root in $\mathcal{B}_{u \cap v}$. Furthermore, the root of minimum height cone on $\mathcal{B}_{u \cap v} \cap P'$ is the closest point on $\mathcal{B}_{u \cap v} \cap P'$ to A_u . Since the set of closest points on a plane to a parallel line is the intersection of the orthogonal plane through the line, the minimum height cone on $\mathcal{B}_{u \cap v} \cap P'$ is on P . Thus, the minimum height cone that contains u and v , which is the minimum over P' of minimum height cones on $\mathcal{B}_{u \cap v} \cap P'$, is on P .

Umbral Cones: The proof for the umbral construction is identical to the penumbral construction, except that \mathcal{S} is a point at $x_d = \infty$. \square

7.5 Implementation Details

All experiments were run on a shared GPU cluster with various machine configurations. All GPUs in the cluster were NVIDIA Volta or newer cards, and all machines had Intel Skylake or newer CPUs or AMD Milan or newer CPUs. Most experiments used PyTorch 2.0 with `torch.compile` and TF32 turned on for matrix multiplications. We provide PyTorch implementations of the infinite-setting penumbral and umbral attention operators at <https://github.com/tsengalb99/coneheads>. Below, we give implementation details for each tested model. For penumbral attention, we set the height of \mathcal{S} to $h = 1$. For umbral attention, we set the radius of the ball around each point to $r = 0.1$.

Graph Attention Networks. We use the Pytorch-GAT repository (<https://github.com/gordicaleksa/pytorch-GAT>) for our experiments. In this repository, we modified the `//models/definitions/GAT.py` file to implement various attention mechanisms. We use the provided training commands and hyperparameters to train models. We experimented with different hyperparameters, which did not have a significant effect on the final results. We report the default attention results from the original GAT paper.

Neural Machine Translation (NMT) Transformers. For NMT experiments, we used the fairseq repository available at <https://github.com/facebookresearch/fairseq>. We primarily modified `//fairseq/modules/multihead_attention.py`. We used the commands provided at <https://github.com/facebookresearch/fairseq/blob/main/examples/translation/README.md> to download and preprocess the IWSLT’14 De-En dataset and to train models.

Vision Transformers. For DeiT-Ti experiments, we use the Facebook DeiT repository available at https://github.com/facebookresearch/deit/blob/main/README_deit.md. We observed that all methods performed better at 500 epochs than at 300 epochs, and we report our experimental results for 300 and 500 epochs. The DeiT repository relies on the Hugging Face timm repository, which is available at <https://huggingface.co/timm>. We primarily modify `//models/vision_transformer.py` in the timm.

Table 4: Training speed in iterations per second of various attention models across the 4 tested transformers. When used in transformers, cone attention is slightly slower than dot product attention. As mentioned in the main text, `torch.compile` is not optimal at fusing operations, and the performance gap can likely be narrowed with a custom CUDA implementation. `torch.compile` was not used for the NMT transformer since the training speed was sufficiently fast and, as of writing, `torch.cdist` has issues with dynamic-shaped tensors. We expect this to be fixed in future PyTorch releases and the performance to be similar to the Adaptive Inputs transformer. All numbers were obtained on a single NVIDIA Tesla V100 SXM2 GPU.

Method	NMT (4096 tokens/it)	Adaptive Inputs (4096 tokens/it)	DeiT-Ti (bs=256)	DiT-B/4 (bs=64)
Dot Product	9.09	0.526	62.9	1.09
Penumbral	7.10 (-21.9%)	0.490 (-6.86%)	51.3 (-18.6%)	1.08 (-0.92%)
Umbral	7.48 (-17.7%)	0.488 (-7.32%)	56.0 (-11.0%)	1.07 (-1.83%)
<code>torch.compile?</code>	No	Yes	Yes	Yes

Adaptive Input Representations for Transformers. For adaptive inputs experiments, we use the same fairseq repository as the NMT experiments. The `transformer_lm_wiki103` architecture in the fairseq repository uses 8 attention heads per module, which does not match 16 attention heads per module in [5]. Thus, our results and those from other papers that use this repository, such as [41], are not directly comparable to the results presented in [5].

Diffusion Transformers. For DiT-B/4 experiments, we use the code and training instructions available at <https://github.com/facebookresearch/DiT>. We use the Pytorch version of the codebase, and train with seed 42 to match the experiments presented in [25] and the repository. The DiT codebase also uses timm, so we make the same modifications as in DeiT. We report the DiT-B/4 seed 42 PyTorch result for dot product attention from the DiT Github repository.

7.6 Runtime Comparison

Table 4 shows the training speed of the 4 tested transformers in iterations per second. Like dot product attention, cone attention requires $O(n^2d)$ operations. However, cone attention requires more operations, since dot product attention can be computed with a batch matrix multiply. Inside a transformer, cone attention generally results in a 10-20% performance decrease when implemented with `torch.compile`. However, `torch.compile` is not perfect, and an optimized raw CUDA implementation would likely narrow the speed gap between dot product and cone attention. `torch.compile` was not used for the NMT transformer since the training speed was sufficiently fast and, as of writing, `torch.cdist` has issues with dynamic-shaped tensors. We expect this to be fixed in future PyTorch releases and performance to be similar to the Adaptive Inputs transformer.

7.7 α -approximate rank

Here, we discuss an interesting and potentially useful connection between the problem of encoding hierarchies in dot product attention and the α -approximate rank problem. This connection gives us some intuition as why larger models with large token embedding dimensions perform well, even with hierarchical data. We note that we have not extensively studied this connection and how it relates to training dynamics, such as in the context of learning attention with gradient based optimizers, and leave that for future work.

Consider the following formulation of encoding a partial ordering in attention:

$$K(x, y) = \exp(\gamma P(x, y)) \quad P(x, y) \in \begin{cases} [1, \alpha] & \text{if } x \preceq y \\ [-\alpha, -1] & \text{if } x \not\preceq y \end{cases} \quad (39)$$

where $1 \leq \alpha \leq \infty$. Essentially, for a set of keys and a single query, the attention matrix should give higher “weight” to keys who are descendants of the query, which is similar to cone attention. If α is close to 1, then this gives a tighter margin on the attention matrix, which gives a “higher quality” attention after softmax normalization. We wish to characterize the number of dimensions d needed in

dot product attention to encode a partial ordering with equation 39. Now, consider the α -approximate rank of a sign matrix A :

Definition 3. α -approximate rank: For a sign matrix $A \in \{-1, +1\}^{n \times n}$, the α -approximate rank is defined as the minimum rank of a matrix $A' \in \mathbb{R}^{n \times n}$ such that $J \leq A \circ A' \leq \alpha J$, where J is the all one's matrix, $\alpha \geq 1$, and \circ is the elementwise product.

In dot product attention, $P = qk^\top$, where $q, k \in \mathbb{R}^{n \times d}$. Note that P has rank d . Furthermore, if the partial ordering is encoded in a sign matrix S s.t. $S_{ij} = +1$ if $i \preceq j$ and -1 otherwise, finding the minimum d s.t. $\exists P$ that satisfies equation 39 reduces to finding the α -approximate rank of S . A number of works have focused on characterizing the α -approximate rank of arbitrary sign matrices. Below, we summarize some results from [2] and [17]. Lee and Shraibman [17] give the following bounds on the α -approximate rank of a $n \times m$ sign matrix A , denoted $\text{rk}_\alpha(A)$.

$$\frac{1}{\alpha^2} \gamma_2^\alpha(A)^2 \leq \text{rk}_\alpha(A) \leq \frac{8192\alpha^6}{(\alpha-1)^6} \ln^3(4mn) \gamma_2^\alpha(A)^6 \quad (40)$$

where $\gamma_2^\alpha(A)$ is defined as

$$\gamma_2^\alpha(A) = \min_{A': J \leq A \circ A' \leq \alpha J} \gamma_2(A') \quad (41)$$

and $\gamma_2(A)$ is defined as

$$\gamma_2(A) = \max_{u, v: \|u\| = \|v\| = 1} \|A \circ vu^\top\|_{tr}, \quad (42)$$

and

$$\text{rk}_{\frac{\alpha+t}{1-t}}(A) \leq \frac{4\gamma_2^\alpha(A)^2 \ln(4mn)}{t^2} \quad (43)$$

for $0 < t < 1$. These bounds depend on γ_2^α , which, to the best of our knowledge, has not been well characterized for partial ordering matrices. However, γ_2^α can be computed in polynomial time with a semidefinite program, which may be useful [17]. Equation 40 gives some indication of how α changes with $d = \text{rk}_\alpha$. From equation 40

$$\alpha \leq \frac{1}{1 - \sqrt[6]{\frac{8192 \ln^3(4n^2) \gamma_2^\alpha(S)^6}{d}}} \quad (44)$$

For fixed S , as d increases, this upper bound on the achievable α margin decreases with $O(1/(1 - \sqrt[6]{1/d}))$. Finally, if we go in the other direction and set $\alpha = \infty$, finding d reduces to finding the sign-rank of S . From [3], the sign-rank of S is bounded by $SC(S) + 1$, where SC is the minimum over all column permutations of S of the maximum number of sign changes in a row of S . If S encodes a hierarchy, and the nodes of that hierarchy are numbered depth first in S , then $SC(S) \leq 2$, and so the sign-rank is at most 3.