


Hyperbolic Segmentation on Point Clouds

WS23/24 Practical Course Deep Learning Models

Julian Kalitzke 

 julian.kalitzke@tum.de

April 1, 2024

Abstract

Recently, logistic regression on hyperbolic space rather than Euclidean space has been shown to improve accuracy in pixel-wise semantic image segmentation. In this project, I extend this idea to three dimensions and segment point clouds using hyperbolic multinomial logistic regression (MLR). While I cannot find a significant increase in accuracy when comparing hyperbolic and Euclidean variants, I do observe the importance of proper parameter initialization in hyperbolic MLR. I furthermore find that the inclusion of hierarchical information improves accuracy much more than in the Euclidean case. And lastly, average inter-class results tended to be better in the hyperbolic setting, indicating a better generalization ability.

1 Introduction

Deep learning algorithms operating on hyperbolic manifolds instead of Euclidean space have recently gathered increasing research interest as hyperbolic space seems to lend itself well for problems exhibiting a hierarchical structure [10]. In the field of computer vision, hyperbolic neural networks [5] have been shown to be effective in prototype learning [7], image segmentation [1] and in modeling uncertainty [8].

While computer vision is often associated with two-dimensional image data, increased effort is being put in research on processing three-dimensional (3D) objects, as they represent reality more accurately. One problem which naturally extends to this setting from the two-dimensional case is semantic segmentation. Just as image segmentation is typically described as a pixel-wise classification problem, 3D part segmentation is the classification of each point in a point-cloud, i.e. a set of points in 3D space representing an object in question.

In this practical course, I aim to extend efforts in hyperbolic image segmentation to the 3D setting by applying hyperbolic multinomial logistic regression (MLR) to a 3D part segmentation problem. In the following section, I will provide mathematical back-

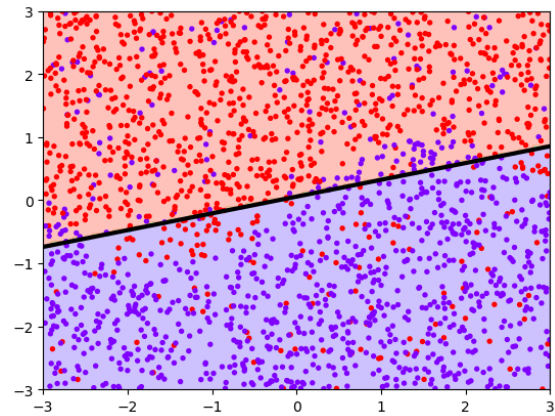


Figure 1 Simple logistic regression

ground information on hyperbolic geometry and logistic regression. This is followed by a section detailing my experimentation setup to compare Euclidean and hyperbolic variants and various hyper-parameters. In the fourth section, I present the results along with some specifically interesting findings. Finally, I discuss those results, providing some hypotheses as well as an outlook on possible further research.

I would like to thank my supervisors Lu Sang and Simon Weber who advised me on this project.

2 Mathematical Principles

Euclidean logistic regression Logistic regression is a common way to model a binary outcome depending on data points comprised of multiple input features. Typically, each input data point (x_1, \dots, x_n) is viewed as an n -dimensional vector in \mathbb{R}^n which shall be mapped to either 1 or 0, usually corresponding to "is in class" or "is not in class". In linear logistic regression, a hyperplane is fit such that it bests separates the two possibilities. The fitted hyperplane is also known as decision boundary. An example using synthetic data in \mathbb{R}^2 can be seen in Figure 1 with the decision boundary shown in black. It is possible to also model the probability of a data point to belong to the class, with the formula $p = e^l$ with l being the perpendicular (i.e. geodesic) distance between the data

point and the hyperplane. Note that depending on the side of the boundary, l is either positive or negative. l is also called the data point's logit.

So far, we have only modeled the probability of a data point belonging to a class or not. In the multinomial case, we are interested in the choice among K possible classes where K hyperplanes are fit by minimizing cross entropy and each input produces K logits l_k . The most likely class is the one with the largest probability (i.e. logit).

In their seminal paper on hyperbolic neural networks, Ganea et al. formulate this exact concept over hyperbolic geometry, i.e. a Riemannian manifold with constant negative curvature [5]. Earlier, hyperbolic space had been shown to be effective when embedding highly hierarchical, tree-like structures, as the "additional" space can be exploited to better disentangle child nodes while keeping short distances to their parents [10].

Hyperbolic geometry Hyperbolic space is generally considered as a Riemannian manifold with constant negative curvature c . In literature, c is often simply considered to be 1. n -dimensional hyperbolic space cannot be isometrically embedded into n -dimensional Euclidean space. Instead, there are multiple (isometrically equivalent) models of hyperbolic geometry. In my opinion, the most intuitive one is the Lorentz (also known as hyperboloid) model which is defined over the top sheet of a two-sheeted hyperbola. It is intuitive because the n -dimensional manifold can be isometrically embedded in a $(n + 1)$ -dimensional Minkowski space [11], i.e. the hyperbolic distance between two n -dimensional points is equal to the geodesic distance along their $(n + 1)$ -dimensional embedding. An example is given in Figure 2. The blue hyperbola represents an environment around the origin in 1-dimensional space with constant curvature -1 , embedded in a 2-dimensional Minkowski space. Plotted are the two points (-1.5) and (-0.5) . In 1-dimensional Euclidean space, their distance would be $|-1.5 - (-0.5)| = 1$. However, their hyperbolic distance is roughly 1.867 (the length of the purple line). This means that there is almost double "as much space" between these two points in hyperbolic space than in Euclidean space.

One of the most prominent models to represent hyperbolic geometry is the Poincaré ball for which closed-form formulas to most relevant geometric notions have been derived. Unlike the Lorentz model, the n -dimensional Poincaré ball is fully embedded in

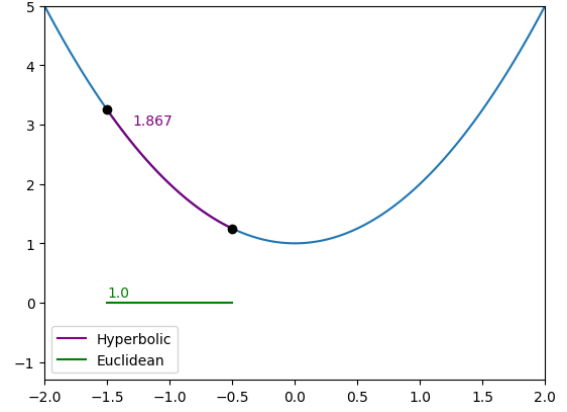


Figure 2 Two points and their distance on the 1-dimensional Lorentz model embedded in a 2-dimensional Minkowski space

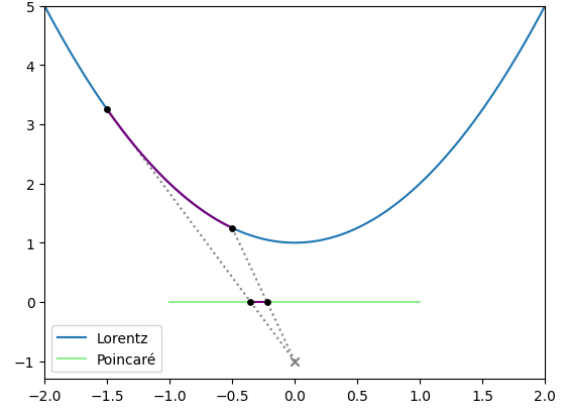


Figure 3 Two points and their distance projected from the 1-dimensional Lorentz model to the 1-dimensional Poincaré ball

n -dimensional Minkowski space. The n -dimensional Poincaré ball with constant negative curvature c is defined as the set $\mathbb{D}_c^n = \{x \in \mathbb{R}^n : c|x| < 1\}$ with Riemannian metric $g_x^{\mathbb{D}_c^n} = \frac{2}{1-c|x|^2} \mathbb{I}_n$ where \mathbb{I}_n is the n -dimensional identity matrix (i.e. the Euclidean metric) [1]. Given a point x , the Riemannian metric determines the norm in the point's tangent space $\mathcal{T}_x \mathbb{D}_c^n$: $|v|_x = \sqrt{v g_x^{\mathbb{D}_c^n} v}$. Note that this norm depends on an anchor point x which is necessary for all operations. To ensure that values be comparable, this anchor point is typically set to the origin, which also simplifies many expressions. In the 2-dimensional space with curvature -1 , the Poincaré ball is the set of points on the open unit disk. Because the embedding is not isometric, the exponentially growing distance to the origin is not as easily graspable.

As all hyperbolic models are isometrically equivalent, there exists a stereographic projection from the

Lorentz model to the Poincaré ball. This projection is useful to develop an intuition for the distribution of space in the Poincaré model. Figure 3 visualizes this projection for the 1-dimensional case. In hyperbolic space, the two purple lines have the same length, indicating that the Poincaré model significantly "compresses" space.

Hyperbolic logistic regression In order to extend logistic regression to hyperbolic space, Ganea et al. define the notion of *gyroplanes* as hyperplanes in hyperbolic space. A gyroplane can be uniquely defined by a point p and a vector a in the tangent space at p that is orthogonal to the plane. The hyperbolic multinomial logistic regression (MLR) model is then defined by replacing the hyperplanes in Euclidean MLR by hyperbolic gyroplanes.

Specifically, an MLR layer classifying K classes receives a set of n -dimensional points x_i . Each x_i is first mapped from n -dimensional Euclidean space to n -dimensional hyperbolic space, by interpreting it as a vector in the (Euclidean) tangent space at the origin $\mathcal{T}_0\mathbb{D}_c^n$ and then applying the exponential map at the origin, $z_i = \exp_0(x_i)$ which projects tangent space onto the manifold. Then, K logits are calculated as the gyrodesic distances to the respective K gyroplanes, each defined by points p_k and orthogonal vectors a_k . The model is fit to minimize cross entropy loss by optimizing the points p_k using Riemannian Stochastic Gradient Descent [3] or Riemannian Adaptive Optimization Methods [2]. As the orthogonal vectors a_k reside in the Euclidean tangent space, they may be optimized by rescaling and applying regular gradient descent methods [5].

Figure 4 and Figure 5 show examples of hyperbolic simple logistic regression in 2D and 3D space in the Poincaré ball ($n = 2, 3$). In 3D space, gyroplanes embedded in the Poincaré ball resemble a parabolic satellite dish (by nature). Note that the gyroplanes are perfectly straight. The visible curvature is due to the before-mentioned projection from an $(n + 1)$ -dimensional isometric embedding to an n -dimensional non-isometric embedding. The greater the Euclidean distance of a point on the disk to the origin, the exponentially larger is their true (hyperbolic) distance.

Figure 6 displays hyperbolic multinomial logistic regression with four classes and corresponding data. The true class is indicated by a data point's color while predicted classes are shown by partitioning the embedding space.

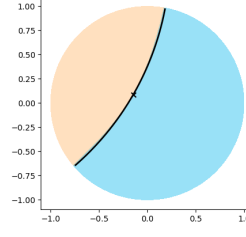


Figure 4 Simple hyperbolic logistic regression in 2D space

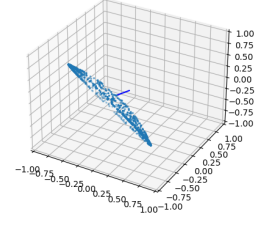


Figure 5 Gyroplane in 3D space

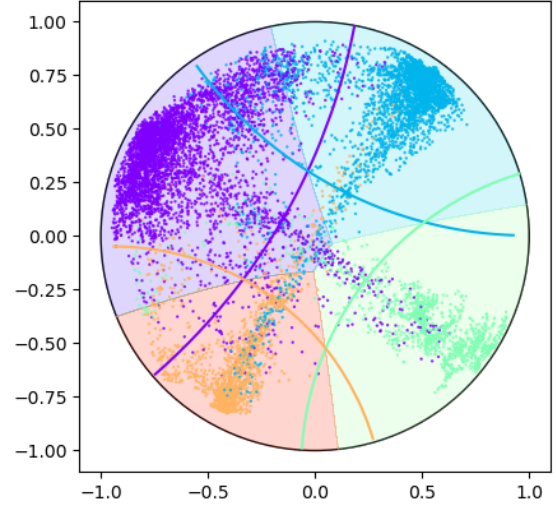


Figure 6 Multinomial hyperbolic logistic regression

3 Experimentation

Atigh et al. use such hyperbolic MLR classifiers on a pixel-wise image segmentation task to show how especially on small dimensions, embedding pixel features into hyperbolic space significantly increases segmentation accuracy [1].

In their evaluation, they use a pre-trained ResNet101 network and DeepLabV3+ as the model backbone followed by a hyperbolic MLR layer to perform the actual classification.

Setup and Architecture In my setup, I aim to extend this concept to the three-dimensional space and perform point-wise segmentation on the ShapeNet-Part dataset [4]. ShapeNet-Part consists of 16 classes of objects each made up of up to 6 parts. In total, there are 50 different parts and each part can only occur in instances of a single class. Example shapes are shown in Figure 9 with their different parts coded in color. Shape classes and parts can be seen as a hierarchy and the idea is that this may lead to easier classification

when embedded in hyperbolic space versus Euclidean space.

A point cloud is represented as a set of N points, randomly sampled from the shape mesh. Each point p_i is characterized by its 3D (Euclidean) coordinates $(p_{i,1}, p_{i,2}, p_{i,3})$. A point cloud is thus a matrix of shape $N \times 3$.

I use the popular PointNet [12] architecture as the backbone of the network. Originally, I experimented with an extension, PointNet++ [13], but decided on the simpler original variant, as the results were comparable and the former is considerably more complex. Given a point cloud input matrix of shape $N \times 3$, PointNet produces an n -dimensional (Euclidean) feature vector x_i per point p_i , i.e. a matrix of shape $N \times n$. In my experiments, I compare $n = 50$, $n = 3$ and $n = 2$. I chose these values as Atingh et al. note that hyperbolic space performs the better compared to Euclidean space the lower the dimension.

Each feature vector is then fed into an MLR layer which classifies it as one of K parts. We denote the k -th logit as l_k and the corresponding probability p_k . Note that in my evaluation, the network’s accuracy over the 16 shape classes is not (directly) measured and no class-related probabilities are returned. Instead, only the 50 parts are output by the logistic regression.

Loss function Analogously to Atigh et al., I compare two different variants of calculating the probabilities p_k from the logits l_k . First, I try regressing and classifying just the 50 parts using cross entropy, thus $p_k = e^{l_k}$. In this variant, the network is unaware of the additional information that some parts are semantically connected by belonging to the same shape class. In the second variant, which Atigh et al. call hierarchical softmax, 16 + 50 gyroplanes are fit. The first 16 logits $l_1 \dots l_{16}$ correspond to the 16 shape classes and the remaining 50 logits $l_{17} \dots l_{66}$ to the 50 parts. For a given shape class C with index $k_C \in [1; 16]$ and parts \mathcal{P}_C with indices $k_{\mathcal{P}_C} = \{k_P : P \in \mathcal{P}_C\} \subseteq [17; 66]$ the probability of an input point belonging to a specific part P with index k_P is then calculated as

$$\begin{aligned} Pr[part = P] = \\ Pr[class = C] \cdot Pr[part = P | class = C] = \\ \frac{e^{l_{k_C}}}{\sum_{k=1}^{16} e^{l_k}} \cdot \frac{e^{l_{k_P}}}{\sum_{k \in k_{\mathcal{P}_C}} e^{l_k}} \end{aligned}$$

This calculates the conditional probability that the point is classified as the part’s shape class and among this class’ parts, it is classified as the part in question.

Evaluation metrics To measure segmentation quality, I calculate the mean union over intersection (mIoU) of the true segmentation and the predicted segmentation in a validation batch. In order to assess the network’s ability to generalize well over structurally heterogeneous classes, I also calculate the mean union over intersection after grouping samples by class. Lastly, I determine the class posterior based on part probabilities as suggested by Weber et al. [14]. Given a class C and its parts \mathcal{P}_C , the class posterior is defined as

$$\begin{aligned} Pr[class = C] = \\ \sum_{P \in \mathcal{P}_C} Pr[part = P] = \\ \sum_{P \in \mathcal{P}_C} p_{k_P} \end{aligned}$$

Note that in the case of the simple loss, p_{k_P} are learned without hierarchical knowledge, whereas for hierarchical softmax, class-part relations are implicitly learnt by the network. We would thus expect better per-class accuracy using hierarchical softmax than simple softmax.

The MLR layer is characterized by three hyperparameters: the dimension of the Poincaré ball n , the number of parts K to decide on, and the negative Riemannian curvature c . In all experiments $c = 1$, as Atigh et al. reported this value to perform best on small dimensions and most literature chooses it implicitly.

To investigate the difference in performance based on point cloud density, I compare training on a mesh resolution of 2048 points and a batch size of 32 to a resolution of 1024 with a batch size of 16. Network Euclidean and hyperbolic parameters are optimized over 100 epochs using Euclidean and Riemannian Adaptive Optimization Methods respectively, both with a learning rate of 10^{-3} .

4 Results

Over all experiments, it can be said that in general, hyperbolic logistic regression did not significantly improve classification accuracy. Leaving all other hyperparameters unchanged, both configurations usually



Figure 7 In each of the three accuracy metrics, the top three configurations converge to the same maximum

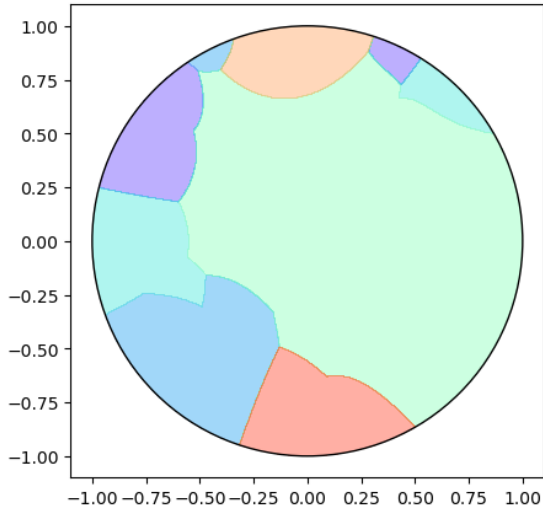


Figure 8 Segmented embedding space for $n = 2$

converged to the same maximal accuracy, with the Euclidean variant often performing slightly better. Figure 9 shows true versus predicted segmentation using hyperbolic MLR, dimension $n = 50$, hierarchical softmax and a point cloud resolution of 1024 points.

The overall best per-instance mIoU accuracy (0.8356) was achieved using Euclidean space, with dimension $n = 50$, hierarchical softmax and the smaller resolution of 1024 points. This is followed by the equivalent configuration instead using hyperbolic space (0.834) and then by the variant using Euclidean space, $n = 50$, simple softmax and a resolution of 2048 points (0.8327).

In terms of per-class mIoU accuracy, using Euclidean space, $n = 50$, simple softmax and 2048 points yielded the best result (0.7791). The next-best per-class accuracy were both using hyperbolic space, $n = 50$. The second-best accuracy was achieved using hierarchical softmax and the lower resolution

(0.7751), followed by the simple softmax and 2048 points (0.7746).

The best class posterior accuracy (0.9897) was found using hyperbolic space, $n = 50$, hierarchical softmax and the smaller resolution of 1024 points. The next-best accuracy (0.989) was achieved by three configurations: both variants (Euclidean and hyperbolic) using dimension $n = 50$, simple softmax and 2048 points, and, remarkably, the hyperbolic variant using $n = 3$, hierarchical softmax and small dimension of 1024 points.

Dimension The embedding dimension n of the embedding space played the largest role in the quality of the classification. Variants using $n = 50$ reached on average a per-instance mIoU of 0.833, an average per-class mIoU of 0.775, and a mean class posterior accuracy of 0.989. For $n = 3$, these numbers drop to 0.812, 0.713 and 0.986, and in the two-dimensional case $n = 2$ they fall to merely 0.736, 0.558 and 0.949.

For $n = 50$, the average per-instance mIoU, per-class mIoU and class posterior accuracy among Euclidean versus hyperbolic variants differed only marginally by 0.2%, 0.5% and 0.1%, respectively.

In the case of $n = 3$, average hyperbolic per-class mIoU was 2.5% higher than among Euclidean variants. Per-instance mIoU and class posterior accuracy did not change significantly.

Compressing embedding space to $n = 2$ lead to a higher average per-instance mIoU among Euclidean variants (1.9%), while per-class mIoU and class posterior accuracy were higher among hyperbolic variants (2.5% and 1.6%, respectively).

Softmax The use of the hierarchical softmax as described by Atigh et al. clearly benefits classification quality. Averaged over all experiments, us-

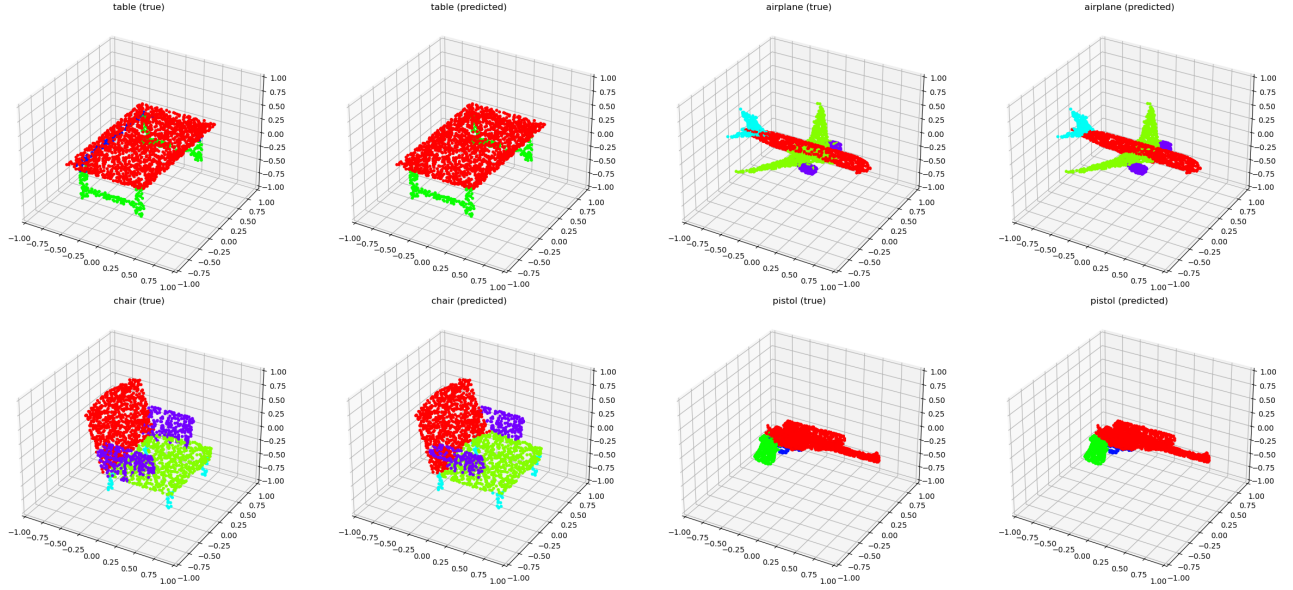


Figure 9 True segmentation versus predicted using hyperbolic MLR

ing hierarchical softmax increased per-instance mIoU from 0.7705 to 0.7899 (+2.5%), per-class mIoU from 0.6455 to 0.6653 (+3.1%) and class posterior accuracy from 0.954 to 0.9837 (+3.1%).

When comparing the hyperbolic configuration with $n = 2$ and the smaller point resolution, per-instance mIoU reaches 0.5763 for the simple softmax versus 0.7531 with hierarchical softmax, which is a gain of roughly 30.7%. An even starker improvement can be seen when examining the class-based metrics. Per-class mIoU climbs from 0.3728 to 0.609 (+63.4%), indicating that inter-class accuracy varies significantly less when training on the hierarchical loss. Class posterior accuracy sees an increase from 0.8148 to 0.9867 (+21.1%).

In the equivalent Euclidean configuration, using hierarchical softmax impacts the accuracy much less. Per-instance mIoU insignificantly decreases from 0.7744 to 0.7739 (-0.1%), per-class mIoU increases from 0.5395 to 0.5933 (+10.0%) and class posterior accuracy improves from 0.9633 to 0.9794 (+1.7%).

Over all experiments, hyperbolic models improved in per-instance mIoU by 6.9%, in per-class mIoU by 9.3% and in class posterior accuracy by 6.4% when using hierarchical softmax. Euclidean models worsened on average, with a decrease by 1.2% in per-instance mIoU and 2.4% in per-class mIoU. Class posterior accuracy did not change significantly.

Resolution Comparing hyperbolic and Euclidean variants in terms of input resolution and batch size, I found that Euclidean performance changed insignif-

icantly. Hyperbolic models responded much more to the increased detail with average per-instance mIoU increasing from 0.7683 to 0.7951 (+3.5%), average per-class mIoU from 0.6566 to 0.6886 (+4.9%) and average class posterior accuracy from 0.9583 to 0.988 (+3.1%). While hyperbolic models performed below Euclidean configurations in the lower resolution, they were on par when training on more detailed point clouds.

I did observe an interesting effect when comparing the hyperbolic variant on $n = 3$ dimensions and hierarchical loss, first with a point cloud resolution of 2048 and then 1024 points. I found that the per-instance mIoU did not change significantly, aside from a slower convergence within the first half of training. However, per-class mIoU started out worse than in the high-resolution setting, but increased quickly and ended up at 0.6834 for the 1024 point configuration versus 0.6378 for the 2048 point variant (improvement of roughly 7.15%). An evolution of these two metrics over the epoch is shown in Figure 10 and Figure 11, respectively. Reducing the dimension to $n = 2$, this effect vanished and both variants converged to the same accuracy.

Initialization As Ganea et al. note in [6], points p in the hyperbolic MLR layer must be initialized close to the origin. Originally, I sampled both points p and directions a from a (multivariate) standard normal distribution ($\mu = 0$, $\sigma^2 = 1$). After changing the per-component distribution of p to the uniform distribution with bounds $[-10^{-3}; 10^{-3}]$, per-instance

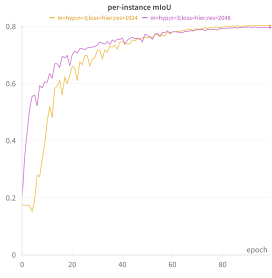


Figure 10 Per-instance mIoU for 1024 mesh points vs. 2048 mesh points

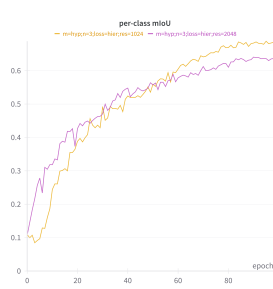


Figure 11 Per-class mIoU for 1024 mesh points vs. 2048 mesh points

mIoU increased from 0.7155 to 0.7489 (+4.7%) and per-class mIoU improved significantly from 0.489 to 0.6049 (+23.7%) when examining the case $n = 2$ using hierarchical loss.

5 Discussion

Unfortunately, I ultimately could not produce the same result as Atigh et al., where the hyperbolic model performed significantly better in terms of accuracy than the Euclidean equivalent.

In most experiments, both variants seem to converge to the same maximum accuracy, but the hyperbolic model performs worse, sometimes significantly, but in general so little that performance can be considered as equal.

The results suggest that for higher embedding dimensions, Euclidean and hyperbolic models are nearly identical. This is aligned with evaluations performed by Atigh et al. and makes sense because in higher dimensions, the network has ample space to distribute points, thus the ability of using space efficiently is not critical. Forcing the network to densely embed points in lower dimensions hinted at an advantage of hyperbolic variants in class-related metrics. This seems plausible as these measures benefit from an embedding that reflects the hierarchical relationship, which hyperbolic space is suspected to be better at. The fact that Euclidean networks perform higher in terms of per-instance accuracy but reach a lower per-class accuracy indicates that they performed better on some classes but much worse on others than their hyperbolic counterparts (higher inter-class variance in classification accuracy).

The implicitly learnt topological knowledge when using hierarchical softmax clearly enables hyperbolic networks to better disentangle classes, as is indicated by the immensely improved accuracy, especially in

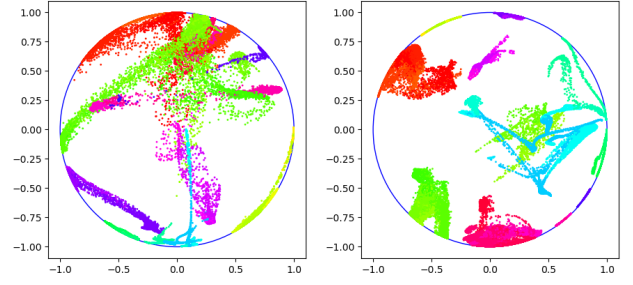


Figure 12 Hyperbolic embeddings with simple softmax versus hyperbolic softmax

per-class mIoU. Figure 12 depicts point embeddings after training using simple versus hierarchical softmax. It is evident that parts of the same class (blobs of similar color) are much better geometrically isolated in the latter case and much more entangled in the former.

Examining the embeddings more closely, some class are embedded almost exclusively on the edge of the disk, i.e. they collapse at the boundary. This effect has been observed by various authors (e.g. [6] or [9]) and has been linked to degraded performance due to numerical instability of the Poincaré model stemming from limited precision in floating-point arithmetic.

In the experiments published by Atigh et al., classifiers appear to converge to a state where gyroplanes representing children were geometrically enclosed by the gyroplanes of their parent, mirroring the hierarchical relations. In my trials, I could not observe the same effect, even when using hierarchical softmax. Rather, class gyroplanes seemed to not at all geometrically relate to gyroplanes of their children. I tried to add a loss that would penalize if the gyroplane point p of a class was farther from the origin (larger $|p|$) than those of its parts' gyroplane. This resulted in significantly decreased performance, so I did not pursue the idea further.

Resolution seemed to not have a significant effect on overall accuracy in general. Curiously, the larger dimension sometimes produced worse accuracy than the smaller variant. However, because training was always limited to 100 epochs, this could simply indicate that the PointNet backbone converges slower on the larger resolution, which seems plausible. The curious case I noted in the results section, where per-class mIoU seemed to converge to a better accuracy in the lower resolution variant could indicate that higher resolution leads to overfitting on some class geometries and thus more inter-class accuracy variance.

Upon investigation, the bottleneck seems to be mostly the inability of PointNet to disentangle the

shape geometries any further. This is supported by the fact that in most experiments, hyperbolic and Euclidean variants converge to the same classification accuracy. Because the Euclidean configuration is a simple linear layer on PointNet output, its accuracy on $n = 50$ is equivalent to plain PointNet. The fact that this configuration scored best in terms of per-instance mIoU and the top hyperbolic models were almost equivalent in performance, suggests that the upper bound on accuracy is mostly determined by PointNet and whether using hyperbolic or Euclidean embedding space only plays a marginal role.

Outlook As the overall network output seems to be limited by PointNet, an obvious first step would be to consider better backbone models to improve disentanglement of geometries, especially in low dimensions such as $n = 2$ and $n = 3$.

Another option is pre-training the backbone, for example in a Euclidean setting and then optimizing just the hyperbolic embedding and classification in a second step. In fact, Atigh et al. use a pre-trained ResNet101 as their backend.

In order to further assess the effect of larger resolutions, experiments could be rerun for longer periods to examine the convergence behavior depending on shape resolution. Furthermore, modifying the batch size in my experiments could have introduced bias to the evaluation on the effect of resolution.

Another hyper-parameter to examine is the Riemannian curvature c . While I do not believe that much greater accuracy can be achieved, the optimum might not be at $c = 1$, but at some slightly different level.

Lastly, improvements have been suggested to enhance embedding stability on the Pointcaré ball. Ganea et al. present hyperbolic entailment cones [6] to force the spatial distribution of embeddings to reflect hierarchical relations. Another approach makes use of tiling methods, such as in [15], to confine precision loss to fixed bounds across the whole embedding space.

6 Conclusion

In this project, I explored the performance of hyperbolic neural networks on semantic point-wise 3D shape segmentation. I compare Euclidean and hyperbolic configurations, as well as other hyper-parameters such as point-cloud resolution, embedding dimension and the harnessing of hierarchical relations in the data. I find that especially the latter had a significant effect

on hyperbolic models which is aligned with prior research. In terms of achieved accuracy, however, hyperbolic classification metrics would converge to perform either below or on par with the Euclidean equivalents.

References

- [1] Mina Ghadimi Atigh et al. “Hyperbolic Image Segmentation”. In: *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2022, pp. 4443–4452. DOI: 10.1109/CVPR52688.2022.00441.
- [2] Gary Bécigneul and Octavian-Eugen Ganea. “Riemannian Adaptive Optimization Methods”. In: *CoRR* abs/1810.00760 (2018). arXiv: 1810.00760. URL: <http://arxiv.org/abs/1810.00760>.
- [3] Silvere Bonnabel. “Stochastic Gradient Descent on Riemannian Manifolds”. In: *IEEE Transactions on Automatic Control* 58.9 (Sept. 2013), pp. 2217–2229. ISSN: 1558-2523. DOI: 10.1109/tac.2013.2254619. URL: <http://dx.doi.org/10.1109/TAC.2013.2254619>.
- [4] Angel X. Chang et al. “ShapeNet: An Information-Rich 3D Model Repository”. In: *CoRR* abs/1512.03012 (2015). arXiv: 1512.03012. URL: <http://arxiv.org/abs/1512.03012>.
- [5] Octavian Ganea, Gary Becigneul, and Thomas Hofmann. “Hyperbolic Neural Networks”. In: *Advances in Neural Information Processing Systems*. Ed. by S. Bengio et al. Vol. 31. Curran Associates, Inc., 2018. URL: https://proceedings.neurips.cc/paper_files/paper/2018/file/dbab2adc8f9d078009ee3fa810bea142-Paper.pdf.
- [6] Octavian-Eugen Ganea, Gary Bécigneul, and Thomas Hofmann. “Hyperbolic Entailment Cones for Learning Hierarchical Embeddings”. In: *CoRR* abs/1804.01882 (2018). arXiv: 1804.01882. URL: <http://arxiv.org/abs/1804.01882>.
- [7] Valentin Khrulkov et al. “Hyperbolic Image Embeddings”. In: *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2020, pp. 6417–6427. DOI: 10.1109/CVPR42600.2020.00645.
- [8] Pascal Mettes et al. *Hyperbolic Deep Learning in Computer Vision: A Survey*. 2023. arXiv: 2305.06611 [cs.CV].

- [9] Gabriel Moreira et al. *Hyperbolic vs Euclidean Embeddings in Few-Shot Learning: Two Sides of the Same Coin*. 2023. arXiv: 2309.10013 [cs.CV].
- [10] Maximilian Nickel and Douwe Kiela. “Poincaré Embeddings for Learning Hierarchical Representations”. In: *CoRR* abs/1705.08039 (2017). arXiv: 1705.08039. URL: <http://arxiv.org/abs/1705.08039>.
- [11] Wei Peng et al. “Hyperbolic Deep Neural Networks: A Survey”. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 44.12 (2022), pp. 10023–10044. DOI: 10.1109/TPAMI.2021.3136921.
- [12] Charles Ruizhongtai Qi et al. “PointNet: Deep Learning on Point Sets for 3D Classification and Segmentation”. In: *CoRR* abs/1612.00593 (2016). arXiv: 1612.00593. URL: <http://arxiv.org/abs/1612.00593>.
- [13] Charles Ruizhongtai Qi et al. “PointNet++: Deep Hierarchical Feature Learning on Point Sets in a Metric Space”. In: *CoRR* abs/1706.02413 (2017). arXiv: 1706.02413. URL: <http://arxiv.org/abs/1706.02413>.
- [14] Simon Weber et al. *Flattening the Parent Bias: Hierarchical Semantic Segmentation in the Poincaré Ball*. 2024. arXiv: 2404.03778 [cs.CV].
- [15] Tao Yu and Christopher M De Sa. “Numerically Accurate Hyperbolic Embeddings Using Tiling-Based Models”. In: *Advances in Neural Information Processing Systems*. Ed. by H. Wallach et al. Vol. 32. Curran Associates, Inc., 2019. URL: https://proceedings.neurips.cc/paper_files/paper/2019/file/82c2559140b95ccda9c6ca4a8b981f1e-Paper.pdf.