Rotation-invariant Hierarchical Segmentation on Poincaré Ball for 3D Point Cloud

Pierre Onghena* Leonardo Gigli* Santiago Velasco-Forero†
TheCrossProduct* MINES Paris, PSL Research University†
{pierre.onghena, leonardo.gigli}@thecrossproduct.com Santiago.Velasco@Mines-Paristech.fr

Abstract

Point clouds are a set of data points in space to represent the 3D geometry of objects. A fundamental step in the processing is to achieve segmentation of the point cloud at different levels of detail. Within this context, hierarchical clustering (HC) breaks the point cloud down into coherent subsets to recognize the parts that make up the object. Along with classic approaches that build a hierarchical tree bottom-up using linkage criteria, recent developments exploit the tree-likeness of hyperbolic metric space, embedding data into the Poincaré Ball and capturing a hierarchical structure with low distortion. The main advantage of this kind of solution is the possibility to explore the space of discrete binary trees using continuous optimization. However, in this framework, a similarity function between points is assumed to be known, while this cannot always be granted for point cloud applications. In our method, we propose to use metric learning to fit at the same time the good similarity function and the optimal embedding into the hyperbolic space. Furthermore, when arbitrary rotations are applied to a 3D object, the pose should not influence the segmentation quality. Therefore, to avoid extensive data augmentation, we impose rotation invariance to ensure the uniqueness of the hierarchical segmentation of point clouds. We show the performance of our method on two datasets, ShapeNet and PartNet, at different levels of granularity. The results obtained are promising when compared to state-of-the-art flat segmentation. [1]

1. Introduction

In today's Information Age, digital data has become a commodity that is easily accessible and widely disseminated. As a response, many sectors in the industry have embraced new information technologies, including LiDAR technology, to facilitate and improve decision-making processes. Nowadays, it's common to create a digital twin or point cloud by scanning a 3D object with LiDAR. A crucial step is the semantic segmentation of a point cloud, which aims to partition each point with a label. It allows achieving an understanding of 3D objects or scenes in many applications such as infrastructure maintenance or autonomous vehicles. To allow an extensive and accurate analysis, deep learning enables us to improve the performance in the quality of predictions. However, a flat segmentation that creates a partition of non-overlapping segments is not always sufficient to describe the input data.

Hierarchical Clustering (HC) provides an analysis of point clouds to improve the understanding of how parts are connected at different levels of detail. Hence, in contrast to flat segmentation, hierarchical clustering provides a hierarchy that structures parts across granularity levels, from coarse- to fine-grained concepts. Hierarchical clustering can be performed by a variety of techniques, including traditional algorithms based on similarities between points. These similarity-based HC methods leverage the pairwise similarities to decode a tree-like structure as the dendrogram with a linkage method. In particular, an approach has been proposed by Dasgupta [7] which, given a weighted-graph defined on a dataset, introduces a cost function to evaluate different hierarchical trees. However, the discrete optimization in this setting to find the optimal hierarchy is NP-Hard. Therefore, Chami et al. [5] exploit the ability of hyperbolic space to implicitly encode a hierarchical structure and propose a continuous relaxation of the Dasgupta's cost function in the Poincaré space.

Although the knowledge of similarities is a general hypothesis for weighted graphs, there is not a straightforward definition of a similarity relationship for point clouds. For example, the Euclidean distance is not informative enough, as two distinct points could be close to each other when separating two parts in space. Accordingly, hierarchical segmentation of point clouds is confronted with the barrier of finding a similarity function on the input space. To this end, we make use of state-of-the-art of deep learning models for point cloud segmentation to extract features of point clouds. The main contribution of this paper is threefold:

1. We integrate metric learning in hyperbolic space to find a similarity function. At the same time, the similarity

[1] Project code: https://github.com/TheCrossProduct/HPCS

function is evaluated to determine an optimal embedding successfully.

2. We propose a method to learn a hierarchical clustering using only the labels of a flat segmentation as ground truth by means of hyperbolic geometry.

3. Instead of most papers that assume pre-aligned 3D objects, we integrate a rotation invariant framework to allow the geometric processing of point clouds in arbitrary poses.

2. Related Work

Many research efforts in the field of hierarchical clustering have focused on the development of deep learning methods by offering a means of continuous representation. This optimization framework has enabled us to determine a hierarchy in hyperbolic space and support the functionality of point cloud segmentation that is invariant to its pose.

3D Point Cloud Processing The state-of-the-art application of deep learning to point cloud data is designed to process raw point cloud data [11]. PointNet [23] pioneered this manipulation, proposing to learn a stack of layers invariant by permutation, but is limited due to its inability to capture fine-grained concepts. Later, PointNet++ [24] introduced a hierarchical neural network to enhance the analysis of small neighborhoods by offering some notion of local similarity. Because of its inherent ambiguity in defining a neighborhood, over the years many ways of implementing convolution operations for point clouds have been proposed [17, 29, 34]. More recently, DGCNN [33] exploits the local neighborhood further by applying convolutions that compute edge features. VN-DGCNN [8] incorporates a framework in DGCNN to facilitate the construction of equivariance and invariance on arbitrary rotations.

Hierarchical Clustering The development of Dasgupta [7] formulates a discrete optimization function for HC that guides a good hierarchy with low cost. Later, Wang and Wang [31] proposed an extended formulation to induce a consistent hierarchy given a similarity graph. Chami et al. [5] derive a continuous analogue in hyperbolic space to define a relaxation of Dasgupta's cost function. More importantly, they propose to learn an optimal embedding in the Poincaré disk, where a tree-like structure is observed to decode a hierarchy. In addition, as studied by Sarkar [26], hyperbolic space has the property to realize low distortion. The original approaches assume that the similarity structure between points is known in advance. In the work of Gigli et al. [10], the pairwise similarities were learned with the application of a triplet loss and the hyperbolic clustering function of Chami et al. [5]. While most works focus on applying a hyperbolic embedding in text or graphs, only limited research efforts have assessed the quality of hierarchical clustering for 3D point clouds. Nonetheless, other works use the hierarchical nature of the hyperbolic space along with metric learning to learn representation of images [9, 12]. In regard to 3D point clouds, Montanaro et al. [20] propose to embed point clouds to account the part-whole hierarchy for classification. Our segmentation contribution differs from [20] due to the use of a rotation-invariant framework presented in Section 3 and a large margin cosine loss in Section 4.

3. Background

To sharpen the intuition before defining our method, this section firstly reviews the background to introduce the optimization of hierarchical clustering in hyperbolic space.

3.1. Similarity-based HC

The classical clustering procedure considers a dataset $\mathcal{D}$ of n datapoints where the pairwise similarities $(w_{ij})_{i,j\in[n]}$ are known in advance. The similarity between datapoints is employed by hierarchical clustering (HC) to construct a binary tree $\mathcal{T}$ where each leaf node corresponds to exactly one datapoint [13]. Clusters of different levels are formed by merging subtrees at internal nodes, in which the root node contains the entire dataset $\mathcal{D}$. When two leaves (i, j) are merged, their lowest common ancestor (LCA) is defined as $i \vee j$. Consequently, $\mathcal{T}[i \vee j]$ is the subtree that depicts the smallest cluster including both i and j. In the case of three leaf nodes (i, j, k), the relation $\{i, j|k\}$ holds if $i \vee j$ occurs deeper in $\mathcal{T}$ than $i \vee j \vee k$. The goal of similarity-based HC is to seek a hierarchical clustering by leveraging the similarity structure between datapoints [6]. Therefore, Dasgupta [7] proposed a cost function to assess the quality of a hierarchy:

$$C_{\text{Dasgupta}}(\mathcal{T}) = \sum_{ij} w_{ij} |\text{leaves}(\mathcal{T}[i \vee j])|. \tag{1}$$

In order to obtain a low cost, a high w_{ij} should be cut as far down as possible in tree $\mathcal{T}$ to minimize the number of leaves in $\mathcal{T}[i \vee j]$. Although the cost function (1) can initiate a good tree with a low cost, Wang and Wang [31] noticed that it resulted in an inconsistent hierarchy given a similarity structure. Therefore, they suggest a more meaningful cost function based on triplets of datapoints i, j, k and their lowest common ancestors (LCA):

$$C_{\text{Wang}}(\mathcal{T}; w) = \sum_{ijk} [w_{ij} + w_{ik} + w_{jk} - w_{ijk}(\mathcal{T}; w)] + 2 \sum_{ij} w_{ij}, \tag{2}$$

where

$$w_{ijk}(\mathcal{T}; w) = w_{ij} \mathbb{1}[\{i, j|k\}] + w_{ik} \mathbb{1}[\{i, k|j\}] + w_{jk} \mathbb{1}[\{j, k|i\}].$$

Consider a triplet (i, j, k) and assume that w_{ij} involves the largest similarity among the three pairwise similarities. The intuition implies that nodes i and j should be merged first before node k to pursue a relation $\{i, j|k\}$. The cost function is designed to reward this relation in $\mathcal{T}$ as it invokes the subtraction of the largest similarity w_{ij} in the first term of equation (2). Consequently, the lower similarities w_{ik} and w_{jk} remain to minimize the loss. For a bad tree $\mathcal{T}$ that merges k first with relation $\{i, k|j\}$ or $\{j, k|i\}$, w_{ij} is added only once and thus differs from Dasgupta's cost function. A HC is optimal when for all triplets, their relations in $\mathcal{T}$ are consistent with their similarities. Moreover, finding a tree that optimizes the cost function in (1) or (2) is NP-hard. Therefore, to introduce continuous optimization, Chami et al. [5] propose to relax (2) in the hyperbolic space.

3.2. Hyperbolic HC

Hyperbolic geometry has shown great advantages in encoding complex information such as point clouds [20], or images [2, 14]. In this section we review the elements necessary to understand our method. Keen readers can find more details at [1, 4, 25].

Hyperbolic Geometry A Riemannian manifold is defined as a pair consisting of a *manifold* $\mathcal{M}$ and a positive-definite inner product g_x in the tangent space $T_x\mathcal{M}$ at each point $x \in \mathcal{M}$. One can define an *exponential map* $\exp_x$ which projects any vector v of the tangent space $T_x\mathcal{M}$ onto $\mathcal{M}$, such that $\exp_x(v) \in \mathcal{M}$, and inversely a logarithmic map which projects any point in $\mathcal{M}$ back onto the tangent space at x. The N-dimensional hyperbolic space is an N-dimensional Riemannian manifold of constant negative curvature $-c$. It can be described using several isometric models [4]. We will use the Poincaré ball model $(\mathbb{B}_c^N, g_c^{\mathbb{B}})$ where $\mathbb{B}_c^N := \{p \in \mathbb{R}^N \mid c||p||^2 \leq 1\}$. We assume $c > 0$, such that $\mathbb{B}_c^N$ corresponds to a ball of radius $1/\sqrt{c}$ in Euclidean space. The Poincaré ball is coupled with a Riemannian metric $g_c^{\mathbb{B}}(p) = \left(\frac{2}{1-c||p||}\right)^2 g_{\mathbb{E}}$ where $p \in \mathbb{B}^N$ and $g_{\mathbb{E}}$ is the canonical metric of the Euclidean space. The geodesic distance between two points $p, q \in \mathbb{B}^N$ is specified as:

$$d_{\mathbb{B}^N}(p, q) = \cosh^{-1}\left(1 + 2\frac{\|p - q\|^2}{(1 - \|p\|^2)(1 - \|q\|^2)}\right). \tag{3}$$

With the origin $q = 0$, (3) could be simplified to:

$$d_o(p) = d(o, p) = 2\tanh^{-1}(\|p\|). \tag{4}$$

As opposed to Euclidean geometry, this particular space has the property to infer the underlying hierarchical structure of a dataset in its embedding [2, 15]. Moreover, as researched by Nickel et al. [22], (3) is able to reflect the similarity of objects with their hyperbolic distance. In addition, a Poincaré embedding provides low distortion to simultaneously learn the hierarchy and similarities for large trees.

Hyperbolic Optimization The perspective of hyperbolic space to represent the similarity information between points as their hyperbolic distance will guide the learning of an underlying hierarchical structure. A differentiable function that enables the similarities is sought to arrange the embedded vertices. To propose a continuous version of Dasgupta's cost function (1), Chami et al. [5] leverage the hyperbolic properties to determine a respective LCA. Firstly, given two leaf nodes i and j of tree $\mathcal{T}$, their LCA $i \vee j$ is the closest node to the root r that lies on the shortest path π_{ij} connecting both i and j. More formally, $i \vee j = \arg\min_{k \in \pi_{ij}} d_{\mathcal{T}}(r, k)$ measures the length of the path from the root r of $\mathcal{T}$ to node k. The same definition applies to the *hyperbolic* LCA, where the shortest path between two points z_i and z_j is represented by their geodesic and $z_i \vee z_j$ is thus the node on that geodesic $z_i \rightsquigarrow z_j$ closest to the origin (see Figure 1). Consequently, the hyperbolic LCA connects the two points in $z_i \vee z_j := \arg\min_{z \in z_i \rightsquigarrow z_j} d_o(z)$ and the hyperbolic distance between the origin and LCA is computed using (4).

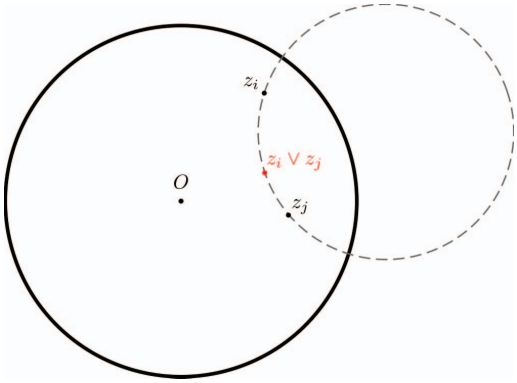

Figure 1. Given two points $z_i, z_j \in \mathbb{B}^N$, the *hyperbolic* LCA $z_i \vee z_j$ is defined as the point on the geodesic connecting z_i and z_j (gray dashed line) closest to the origin.

Equation (2) was non-differentiable due its term $w_{ijk}(\mathcal{T}; w)$ that sought the LCA among one of relations $\mathbb{1}[\{i, j|k\}], \mathbb{1}[\{i, k|j\}], \mathbb{1}[\{j, k|i\}]$. With the definition of hyperbolic LCA, Chami et al. [5] introduce a differentiable function to learn an optimal Poincaré embedding. Let $Z = \{z_1, ..., z_n\} \subset \mathbb{B}^N$ be an embedding of a tree $\mathcal{T}$ with n leaves, their cost function is then:

$$C_{\text{HypHC}}(Z; w, \tau) = \sum_{ijk}(w_{ij} + w_{ik} + w_{jk} - w_{\text{HypHC},ijk}(Z; w, \tau)) + 2\sum_{ij} w_{ij}, \tag{5}$$

where

$$w_{ijk}(Z; w, \tau) = (w_{ij}, w_{ik}, w_{jk}) \cdot \sigma_\tau(d_o(z_i \vee z_j), d_o(z_i \vee z_k), d_o(z_j \vee z_k))^\top.$$

The probability distribution outputted by the softmax function $\sigma_\tau(\cdot)$ is controlled by a temperature τ: $\sigma_\tau(w)_i = e^{w_i/\tau} / \sum_j e^{w_j/\tau}$. For instance, the model will be more confident about its predictions the closer the hyperparameter gets to zero. The embedding is accompanied by a learnable scale parameter that should increase over training to provide more information and certainty about the tree structure formed. Moreover, the scale normalizes the embedding in the Poincaré disk so that all points are on the same radius to calculate the LCA hyperbolic distance to the origin.

3.3. Rotation-invariant Framework

When arbitrary rotations are applied to a 3D object, the global object pose should not influence the segmentation quality. The most commonly used solution is to take advantage of *data augmentation*, where the inputs are randomly rotated during training to induce an insensible output to that transformation [16]. However, data augmentation does not imply neither equivariance nor invariance. There is a need to impose invariance while avoiding an extensive data augmentation of possible rotations during training time. A group-inspired solution [3], is to construct an invariant function, from a non-invariant function by summing over all the group SO(3), which is possible for image but intractable for 3D points. Another alternative is to use the fact that any feature computed on a Gram matrix is rotation invariant [27], however, this is computationally infeasible because an $n \times n$ matrix should be stored in memory during training. Therefore, Deng et al. [8] propose a lightweight framework that is integrable into existing architectures as DGCNN. The main component is a Vector Neuron (VN) representation that lifts classical neurons with 3D vectors. A neural network consists out of neurons where each one takes a scalar input $x \in \mathbb{R}$. The stacked scalars form a C dimensional latent feature that is fed to the neurons of a hidden layer. In specific, a classical feature $x = [x_1, x_2, ..., x_C]^\top \in \mathbb{R}^C$ with $x_i \in \mathbb{R}$ is extended to a vector feature $V = [v_1, v_2, ..., v_C]^\top \in \mathbb{R}^{C\times 3}$ with $v_i \in \mathbb{R}^3$. Thus a point cloud with n points, a collection $\mathcal{V} = \{V_1, V_2, ..., V_n\} \in \mathbb{R}^{n\times C\times 3}$ is obtained where the change of latent channels C follows a mapping between layers that only considers the second dimension of the tensor:

$$\mathcal{V}^{(d+1)} = f(\mathcal{V}^{(d)}; \theta) : \mathbb{R}^{n\times C^{(d)}\times 3} \to \mathbb{R}^{n\times C^{(d+1)}\times 3} \quad (6)$$

with d the layer depth and θ the learnable parameters of the network. The mapping (6) is required to induce rotation equivariance, i.e. for any rotation matrix $R \in$ SO(3), $f(\mathcal{V}R; \theta) = f(\mathcal{V}; \theta)R$. For more detailed information about the VN implementation of hidden layers - linear, non-linear, max pooling and normalization - we refer to the original paper of Deng et al. [8]. Nonetheless, the idea behind their invariant layer will be outlined as this is the property we seek within our method. Therefore, the invariant architecture leverages the output of equivariant VN layers to identify an object that is invariant to its pose. The basic principle is that the product of an equivariant signal $V \in \mathbb{R}^{C\times 3}$ by the transpose of another equivariant signal $T \in \mathbb{R}^{C'\times 3}$ is rotation invariant, i.e., $(VR)(TR)^\top = VRR^\top T^\top = VT^\top$ for all rotation matrix R. Note that as $VT^\top$ is a $C \times C'$ matrix, the computation is feasible for small values of C'. Accordingly, we consider $C' = 3$, and use the equivariant MLP proposed in [8], $T :=$ VN-MLP$([V, \overline{V}])$ where $\overline{V} := \frac{1}{n}\sum_n V \in \mathbb{R}^{C\times 3}$.

4. Method

The goal of Chami et al. [5] was to discover a hierarchy of datapoints by leveraging their given pairwise similarities. In this work, the similarities are unknown but learned end-to-end together with the optimization of a hyperbolic embedding. Therefore, a similarity function is learned such that it would enable us to determine a hierarchy in a Poincaré ball $\mathbb{B}^N$. The objective to seek a good similarity function is defined by the Large Margin Cosine Loss (LMCL) [32] with margin m and scale s:

$$\mathcal{L}_{\text{LMCL}}(Z, A; m, s) = \frac{1}{n}\sum_i -\log \frac{e^{s(A_{y_i}^T z_i^* - m)}}{e^{s(A_{y_i}^T z_i^* - m)} + \sum_{j\neq y_i} e^{sA_j^T z_i^*}}, \quad (7)$$

subject to $||A|| = 1$, where $A_{N\times J}$ is a learnable matrix, J the number of classes, and $z^* := z/||z||$. In (7) z_i is the i-th feature vector corresponding to the ground-truth class of y_i, the A_j is the weight vector of the j-th class. Note that due to norm-one constraints $A_j^T z_i^* = \cos(\theta_{ij})$ where θ_{ij} the angle between A_j and z_i. Geometrically, it means that we assign the sample i to class j if the angle between A_j and z_i is the smallest among all class centers A_j, and m plays the role of angular margin in the decision.

In this method, deep metric learning is integrated with the cost function in (5) to define similarities in hyperbolic space. As a consequence, the LMCL computed using cosine similarity is now evaluated in the Poincaré ball $\mathbb{B}^N$ to learn a similarity function $w_{i,j} = \frac{1}{2}(1 - \frac{\langle z_i, z_j\rangle}{\|z_i\|\|z_j\|})$. By intuition, with the operation of both hyperbolic and LMCL loss in $\mathbb{B}^N$, we think this would improve similarity learning and hyperbolic clustering. Hence, the loss function to find optimal parameters θ and optimal embedding $Z \in \mathbb{B}^N$ is the following:

$$\min_{\|A\|=1, Z\in\mathcal{Z}} \lambda C_{\text{HypHC}}(Z, w; \tau) + \mathcal{L}_{\text{LMCL}}(Z, A; m, s), \quad (8)$$

where λ is a trade-off parameter to control the importance of the hyperbolic loss.

With the hyperbolic space $\mathbb{B}^N$ in mind, the provided point cloud object will first be represented by a neural network in the feature space to extract geometric properties. The learned embedding, established by grouping points in Euclidean space $\mathbb{R}^N$, is projected to the Poincaré ball in $\mathbb{B}^N$ according to the Riemannian metric. At this point, the framework in figure 2 illustrates that the point cloud X is embedded in feature space by a neural network namely VN-DGCNN. This feature embedding U is in turn projected as a Poincaré embedding Z to optimize the loss function defined in (8) and encode a hierarchical structure. With the point clouds X and corresponding class labels Y, we recognize a supervised learning framework where the labels Y are leveraged for classification of LMCL and to define triplets in the computation of the hyperbolic loss. The challenge of estimating similarities and seeking an optimal hyperbolic embedding at the same time imposes the need for a more efficient triplet sampling technique. By defining a set of easy triplets $\mathcal{T}_{\text{Hyp}}$, we provide the hyperbolic loss with estimated similarities to first merge pairs of the same class. The approach is that LMCL triggers hard triplets to become easy. These easy triplets are in turn applied to the hyperbolic loss to perform hierarchical clustering.

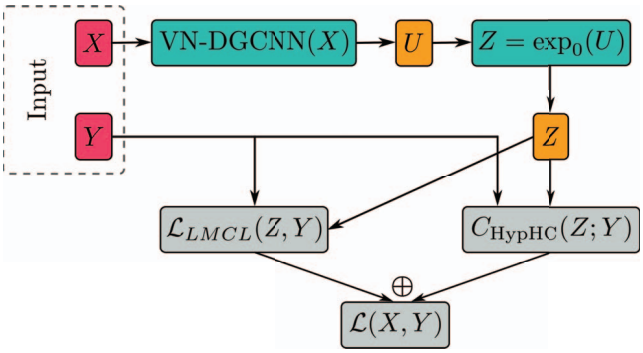

Figure 2. Overview of the method. VN-DGCNN is used to extract points features that are successively embedded into the hyperbolic space. $\mathcal{L}_{\text{LMCL}}$ is a loss for supervised classification in angular similarity and C_{HypHC} is included to promote better hierarchical structure in hyperbolic space.

The objective is to produce an embedding that structures different parts in the Poincaré space. With the leaf nodes representing points of an object, we seek a leaf embedding that organizes connected points more close to each together than unrelated ones. This process of differentiating between points is subject to similarity learning by including LMCL. It induces the convergence of similar points in the Poincaré space to form parts at segmentation levels. Moreover, we aim to arrange a Poincaré space such that it's able to decode a binary tree that captures the hierarchical structure of an object. This goal is illustrated in figure 3, where each part is ordered in a way such that a merge with neighboring parts induces a higher level in the hierarchy.

5. Experiments

5.1. Experimental Setup

Datasets The ShapeNet [35] and PartNet [19] datasets are employed to study the performance of our method. For ShapeNet, the object categories are in coarse-level segmentation. To focus and report the performance on different levels of granularity, PartNet allows us to work in coarse-grained, middle-grained and fine-grained segmentation. An example shape of the chair category in PartNet is given in figure 4 to illustrate the three segmentation levels. For the experiments, results are gathered to establish an evaluation about the hierarchical functionality of the method on different segmentation levels.

Figure 4. From left to right: coarse-, middle- and fine-grained in an example of *chair* class.

Architecture The DGCNN architecture for part segmentation is applied with the VN network to provide robustness over rotations. Consequently, vector neurons are plugged in the DGCNN architecture to create VN-DGCNN that will be used to study the performance of our method. The DGCNN architecture corresponds to VN-DGCNN with integrated VN layers. Three EdgeConv layers are used to capture local geometric features. The information from these layers is aggregated to encode a latent space that is in turn decoded by a MLP network into a feature embedding. This transformation is accompanied by a categorical vector to decode the latent space in an oriented way according to the object category. In specific, when training a model for all categories, the categorical vector is defined by one-hot encoding to adjust the feature embedding to the category of the object running through the network.

Metric Let k be the number of clusters that we want to determine in the generated dendrogram. For the eval-

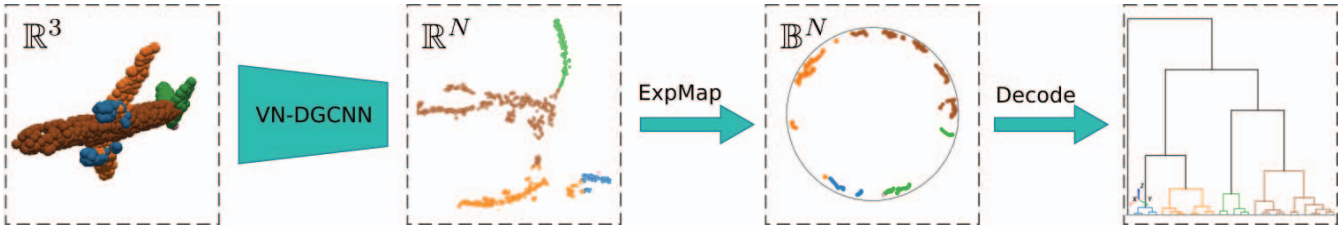

Figure 3. Example of prediction of an airplane object (left). Using VN-DGCNN model we extract features in the Euclidean space (middle-left) and we project them into the Poincaré space (middle-right). Finally, we decode a hierarchical tree (right) using a complete linkage algorithm computed on hyperbolic embeddings. We paint the points according to the prediction that is achieved by selecting the number $k = 4$ of clusters that maximize the IoU with the ground truth.

uation of an object, we recursively iterate over k to obtain a partition of k clusters from the hierarchy. For each k, the clustering quality of the prediction is measured by using the Intersection-over-Union (IoU) score. However, before computing the IoU score, the ground-truth labels $\{l_1, \ldots, l_n\}$ are associated with the cluster prediction labels $\{c_1, \ldots, c_k\}$. Therefore, we compute the matrix that contains the IoU scores of all couples $(l_i, c_j)_{i,j}$ between the classes and clusters. To each label in the ground truth, we assign the predicted label that maximizes the IoU score as follows, $l_i \leftarrow \arg\max_{j \in 1,\ldots,k} \text{IoU}(l_i, c_j)$. Moreover, we introduce *other* to classify all the remaining prediction labels and penalize the prediction when the number of clusters k becomes higher. The IoU score is then computed by using the remapped prediction labels. Finally, the k of the hierarchy that gives the highest IoU in comparison to the ground truth is selected. To compare our method with other benchmarks, the mean IoU (mIoU) is calculated for a category at a segmentation level by averaging the IoUs of all test objects. Thus, we observe the performance with the use of shape mIoU.

Hyperparameters For the experiments, 1024 points are randomly sampled from the point cloud to train, validate and test the model. The k is set to 20 for the k-nearest neighbor (k-NN) graph in the EdgeConv layers of VN-DGCNN. Although VN-DGCNN is invariant, we apply an arbitrary rotation of the SO(3) group to each object during training and test evaluation. The batch size and learning rate for ShapeNet is 16 and 0.001 respectively. For PartNet, the main batch size used is eight, with a learning rate of 0.05. A temperature value of 0.1 is used to control the softmax function $\sigma_\tau(\cdot)$ in (5). The trade-off λ in (8) is set to 0.1 in all our experiments. We fix the margin m and scale s of LMCL to 0.35 and 2 respectively.

We select complete-linkage to decode the Poincaré ball into a corresponding dendrogram. The cosine similarity is chosen as a metric to this agglomerative clustering method to exploit the similarity function learned by LMCL. In addition, it's noticed that the number of points differs for every class in a 3D object. Therefore, the number of triplets for an underrepresented class could be increased to an extent such that it would have more influence on the hyperbolic loss and be segmented.

5.2. Hierarchical Segmentation Results

In the following, an ablation study is performed to provide more insight in the effectiveness of similarity learning and hierarchical clustering. For an fair evaluation, the state-of-the-art performance is compared against our results that focus on hierarchical segmentation and rotation invariance. To this end, the comparison determines if there is any trade-off in the formation of a hierarchy.

5.2.1 ShapeNet

The ShapeNet dataset includes a category label to construct a category vector and train one model for all object categories. Therefore, the training set of ShapeNet contains all the 16 categories with a total of 50 classes to train one model. In addition, the model is tested per category to provide an evaluation in the segmentation quality of different shapes. The state-of-the-art results of ShapeNet dataset are shown at the top of table 1. The traditional point cloud architectures do not provide robustness over rotations with results corresponding to flat segmentation. With the state-of-the-art results of pre-aligned data for training and testing , I/I case, we want to assess the robustness of PointNet and DGCNN to rotations. The evaluation adopts an additional train/test setting, SO(3)/SO(3), with SO(3) for random rotations. The sensitivity of PointNet and DGCNN to data augmentation marks a degradation when the scores for the two rotation groups are compared in the first and second column of the table. As a result, the segmentation performance drops when training and evaluation adopt a random rotation from the SO(3) group. The VN-DGCNN architecture has a consistent segmentation quality over rotation settings. In the paper of Deng et al. [8], the invariant model shows a trade-off in score from 85.2 to 81.4% mIoU when the pre-aligned setting I/I is compared with DGCNN. However, for model (1) with 78.6%

	SO(3)	I	Airplane	Bag	Cap	Car	Chair	Ear	Guitar	Knife	Lamp	Laptop	Motor	Mug	Pistol	Rocket	Skate	Table
PointNet	62.3	83.7	83.4	78.7	82.5	74.9	89.6	73.0	91.5	85.9	80.8	95.3	65.2	93.0	81.2	57.9	72.8	80.6
PointNet++	76.7	85.1	82.4	79.0	**87.7**	77.3	90.8	71.8	91.0	85.9	83.7	95.3	71.6	94.1	81.3	58.7	76.4	82.6
PointCNN	71.4	**86.1**	84.1	86.5	86.0	**80.1**	90.1	**79.7**	**92.3**	**88.4**	**85.3**	**96.1**	**77.2**	95.3	**84.2**	**64.2**	**80.0**	83.0
DGCNN	78.6	85.2	**84.2**	83.7	84.8	77.1	**90.9**	78.5	91.5	87.3	82.9	96.0	67.0	93.3	82.6	59.7	75.5	82.0
(1) VN-DGCNN	78.6	78.6	77.0	71.3	73.0	61.2	85.7	55.2	88.0	78.3	78.8	79.9	25.1	83.2	65.6	44.4	61.6	79.9
(2) VN-DGCNN+**HC**	77.7	77.7	72.8	85.8	77.8	66.7	81.6	63.2	90.4	76.8	78.3	37.7	58.8	94.0	69.5	61.4	73.3	82.7
VN-DGCNN+**B-HC**	**82.1**	82.1	77.6	**87.7**	74.5	71.9	86.2	72.2	91.0	76.9	79.7	56.5	57.2	**96.1**	70.7	51.9	79.0	**86.2**

Table 1. The evaluation metric is shape mIoU (%). **Top:** state-of-the-art part segmentation results of traditional architectures in I/I setting. **Middle:** (1) part segmentation results of VN-DGCNN and (2) hierarchical segmentation results of our method with VN-DGCNN. Both models trained from scratch in SO(3)/SO(3) setting. **Bottom:** Hierarchical segmentation results of our method with VN-DGCNN. As opposed to model (2), model (1) is used as a backbone to train our hierarchical method. The model is thus not trained from scratch in SO(3)/SO(3) setting.

mIoU in the middle of table 1, we retrained from scratch with other hyperparameters than the original DGCNN setting of 2048 points, k set to 40 and batch size of 32.
The results of our HC approach using the VN-DGCNN architecture are shown by model (2) in the middle of table 1. The dimensionality of the Poincaré ball is set to $\mathbb{B}^{32}$ or $\mathbb{B}^{50}$ to provide enough space for the classes of all 16 categories. The analysis of the categorical results reveals that the hierarchical model (2) is able to compete with the flat segmentation of VN-DGCNN model (1). In the bottom of table 1, we use model (1) as a pre-trained feature extractor to train the HC approach. The VN-DGCNN model (1) in the middle of the table thus serves as a backbone to our hierarchical method. In particular, we assume that the pre-trained model gives an arranged feature embedding to map to the Poincaré ball for hierarchical segmentation. To this end, the results demonstrate that model (1) is able to adapt to our approach with scores higher than both in the middle of table 1 without backbone. From the SO(3)/SO(3) setting in the first column, it can be deducted that our method achieves categorical improvements and an impressive overall 82.1% mIoU score for both rotation groups. A prediction of the airplane category is illustrated in 3 to demonstrate the hierarchical ability of neighboring parts.

5.2.2 PartNet

In PartNet, we encounter a wide range of classes for middle- and fine-grained that need to be segmented. The state-of-the-art architecture with the highest shape mIoU scores is selected from the PartNet [19] paper. Namely, the results of PointNet++ are shown in the top of table 2. This point cloud architecture is not rotation-invariant with models corresponding to flat segmentation. Therefore, a network is trained for each category at each segmentation level to obtain the PointNet++ results. Instead of a model per segmentation level, we focus to train a network at fine-grained level for a category. This fine-grained model is in turn tested at coarser granularity levels to assess the hierarchical functionality of our method. Hence, in the bottom of table 2, our VN method is only trained at the finest level 3. The VN results of levels 1 and 2 are obtained according to the best cut of the dendrogram. We include the Pointnet++ results of levels 1 and 2 for comparison, but these models have been trained to predict the labels of each specific simplification level.

A 4-dimensional feature space is opted to map the Euclidean space to a Poincaré ball $\mathbb{B}^4$. A comparison of table 2 shows that the proposed method develops an hierarchical understanding on 3D shapes. We observe an improvement of categorical results, even for the coarser levels to prove the competence of hierarchical learning. An example of the *chair* category at fine-grained level is illustrated in figure

	Avg	Bed	Bott	Chair	Clock	Dish	Disp	Door	Ear	Fauc	Knife	Lamp	Micro	Frid	Stora	Table	Trash	Vase
PN++1	**75.9**	**54.7**	85.8	**84.5**	74.1	**81.9**	**90.7**	73.5	**77.8**	**73.6**	75.0	**65.5**	**80.3**	72.1	**61.2**	**86.7**	**71.5**	81.4
PN++2	**54.7**	**34.8**	-	**54.9**	-	**60.6**	-	57.0	-	-	-	56.8	**63.0**	**58.4**	**52.9**	53.6	-	-
PN++3	**53.4**	**25.1**	61.0	49.6	**46.1**	**52.5**	81.0	48.0	**56.1**	**60.4**	49.1	46.0	**54.3**	**50.7**	**50.6**	47.0	**54.7**	75.1
Avg	**63.7**	**38.2**	73.4	**63.0**	60.1	**65.0**	**85.8**	59.5	**67.0**	**67.0**	62.0	56.1	**65.9**	**60.4**	**54.9**	62.4	**63.1**	78.2
VN1+**HC**	70.2	48.1	**86.3**	70.5	**80.5**	56.7	87.4	**74.6**	67.9	61.3	**76.8**	63.5	76.7	**79.5**	40.6	79.3	59.6	**84.4**
VN2+**HC**	49.0	26.4	-	53.7	-	44.4	-	**73.8**	-	-	-	**62.2**	45.8	53.7	23.0	**58.4**	-	-
VN3+**HC**	51.9	24.2	**85.1**	**52.9**	42.3	42.7	**83.8**	**59.4**	43.3	46.5	**64.9**	**49.0**	41.0	42.4	22.9	**58.0**	47.0	**76.0**
Avg	60.1	32.9	**85.7**	59.0	**61.4**	47.9	85.6	**69.3**	55.6	53.9	**70.9**	**58.2**	54.5	58.5	28.8	**65.2**	53.3	**80.2**

Table 2. The evaluation metric is shape mIoU (%). The numbers 1, 2 and 3 refer to the three granularity levels: coarse-, middle- and fine-grained respectively. **Top:** state-of-the-art part segmentation results of PointNet++ in I/I setting. **Bottom:** hierarchical segmentation results of our method with VN-DGCNN in SO(3)/SO(3) setting.

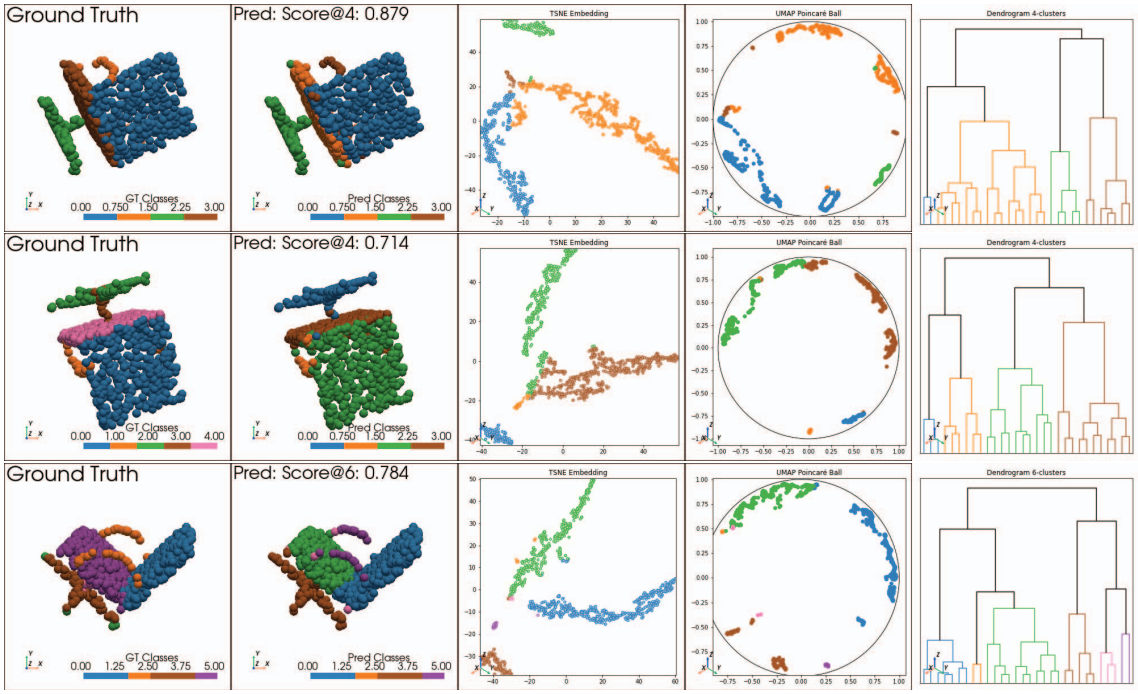

Figure 5. Hierarchical segmentation of *chair* example at each granularity level in PartNet dataset. The N-features in $\mathbb{R}^N$ and $\mathbb{B}^N$ are represented in two dimension using t-SNE [30] and UMAP [18] respectively. From top to bottom: coarse-, middle- and fine-grained segmentation.

5. It shows that the dendrogram is able to capture a similarity function that merges neighboring parts to induce a higher level in the hierarchy. However, the performance differs over categories and we note a limitation of our method over cuboid shapes as the *dishwasher* or *microwave* category. Due to the rotation-invariance, we assume that the model has difficulties to orientate the object among the rectangular surfaces. As opposed to pre-aligned 3D objects, the model is thus at a disadvantage to recognize the front of a microwave or dishwasher.

6. Conclusion

We applied metric learning to study the performance of hierarchical clustering in hyperbolic space. In particular for 3D point clouds, where a similarity function needs to be learned to arrange its parts over different levels of detail. We trained a rotation-invariant architecture on two datasets to examine our method of hierarchical segmentation and inspect any tradeoffs over flat segmentation. The proposed method achieves the best performance among the SO(3) group for ShapeNet and obtains very competitive hierarchical segmentation results trained on the finest level of PartNet. As future work, we can suggest the use of loss functions where the different levels of the hierarchy are specifically included as in [15, 21, 28].

References

[1] James W Anderson. *Hyperbolic geometry*. Springer Science & Business Media, 2006. 3

[2] Mina Ghadimi Atigh, Julian Schoep, Erman Acar, Nanne Van Noord, and Pascal Mettes. Hyperbolic image segmentation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4453–4462, 2022. 3

[3] Michael M Bronstein, Joan Bruna, Taco Cohen, and Petar Veličković. Geometric deep learning: Grids, groups, graphs, geodesics, and gauges. *arXiv preprint arXiv:2104.13478*, 2021. 4

[4] James W Cannon, William J Floyd, Richard Kenyon, Walter R Parry, et al. Hyperbolic geometry. *Flavors of geometry*, 31(59-115):2, 1997. 3

[5] Ines Chami, Albert Gu, Vaggos Chatziafratis, and Christopher Ré. From trees to continuous embeddings and back: Hyperbolic hierarchical clustering. In Hugo Larochelle, Marc'Aurelio Ranzato, Raia Hadsell, Maria-Florina Balcan, and Hsuan-Tien Lin, editors, *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*, 2020. 1, 2, 3, 4

[6] Giovanni Chierchia and Benjamin Perret. Ultrametric fitting by gradient descent. *Advances in neural information processing systems*, 32, 2019. 2

[7] Sanjoy Dasgupta. A cost function for similarity-based hierarchical clustering. In Daniel Wichs and Yishay Mansour,

editors, *Proceedings of the 48th Annual ACM SIGACT Symposium on Theory of Computing, STOC 2016, Cambridge, MA, USA, June 18-21, 2016*, pages 118–127. ACM, 2016. 1, 2

[8] Congyue Deng, Or Litany, Yueqi Duan, Adrien Poulenard, Andrea Tagliasacchi, and Leonidas J. Guibas. Vector neurons: A general framework for SO(3)-equivariant networks. In *2021 IEEE/CVF International Conference on Computer Vision, ICCV 2021, Montreal, QC, Canada, October 10-17, 2021*, pages 12180–12189. IEEE, 2021. 2, 4, 6

[9] Songwei Ge, Shlok Mishra, Simon Kornblith, Chun-Liang Li, and David Jacobs. Hyperbolic contrastive learning for visual representations beyond objects. *arXiv preprint arXiv:2212.00653*, 2022. 2

[10] Leonardo Gigli, Beatriz Marcotegui, and Santiago Velasco-Forero. End-to-End Similarity Learning and Hierarchical Clustering for unfixed size datasets. In *5th conference on Geometric Science of Information*, Paris, France, July 2021. 2

[11] Yulan Guo, Hanyun Wang, Qingyong Hu, Hao Liu, Li Liu, and Mohammed Bennamoun. Deep learning for 3D point clouds: A survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 43(12):4338–4364, 2020. 2

[12] Yunhui Guo, Xudong Wang, Yubei Chen, and Stella X Yu. Clipped hyperbolic classifiers are super-hyperbolic classifiers. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 11–20, 2022. 2

[13] Stephen C Johnson. Hierarchical clustering schemes. *Psychometrika*, 32(3):241–254, 1967. 2

[14] Valentin Khrulkov, Leyla Mirvakhabova, Evgeniya Ustinova, Ivan Oseledets, and Victor Lempitsky. Hyperbolic image embeddings. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 6418–6428, 2020. 3

[15] Liulei Li, Tianfei Zhou, Wenguan Wang, Jianwu Li, and Yi Yang. Deep hierarchical semantic segmentation. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2022, New Orleans, LA, USA, June 18-24, 2022*, pages 1236–1247. IEEE, 2022. 3, 8

[16] Ruihui Li, Xianzhi Li, Pheng-Ann Heng, and Chi-Wing Fu. Pointaugment: an auto-augmentation framework for point cloud classification. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 6378–6387, 2020. 4

[17] Yangyan Li, Rui Bu, Mingchao Sun, Wei Wu, Xinhan Di, and Baoquan Chen. Pointcnn: Convolution on x-transformed points. *Advances in neural information processing systems*, 31, 2018. 2

[18] Leland McInnes, John Healy, Nathaniel Saul, and Lukas Großberger. Umap: Uniform manifold approximation and projection. *Journal of Open Source Software*, 3(29):861, 2018. 8

[19] Kaichun Mo, Shilin Zhu, Angel X. Chang, Li Yi, Subarna Tripathi, Leonidas J. Guibas, and Hao Su. Partnet: A large-scale benchmark for fine-grained and hierarchical part-level 3d object understanding. In *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2019, Long Beach, CA, USA, June 16-20, 2019*, pages 909–918. Computer Vision Foundation / IEEE, 2019. 5, 7

[20] Antonio Montanaro, Diego Valsesia, and Enrico Magli. Rethinking the compositionality of point clouds through regularization in the hyperbolic space. In *Advances in Neural Information Processing Systems*, 2022. 2, 3

[21] Bruce R. Muller and W. Smith. A hierarchical loss for semantic segmentation. In *VISIGRAPP*, 2020. 8

[22] Maximilian Nickel and Douwe Kiela. Poincaré embeddings for learning hierarchical representations. In Isabelle Guyon, Ulrike von Luxburg, Samy Bengio, Hanna M. Wallach, Rob Fergus, S. V. N. Vishwanathan, and Roman Garnett, editors, *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*, pages 6338–6347, 2017. 3

[23] Charles Ruizhongtai Qi, Hao Su, Kaichun Mo, and Leonidas J. Guibas. Pointnet: Deep learning on point sets for 3d classification and segmentation. In *2017 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017, Honolulu, HI, USA, July 21-26, 2017*, pages 77–85. IEEE Computer Society, 2017. 2

[24] Charles Ruizhongtai Qi, Li Yi, Hao Su, and Leonidas J. Guibas. Pointnet++: Deep hierarchical feature learning on point sets in a metric space. In Isabelle Guyon, Ulrike von Luxburg, Samy Bengio, Hanna M. Wallach, Rob Fergus, S. V. N. Vishwanathan, and Roman Garnett, editors, *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*, pages 5099–5108, 2017. 2

[25] John G Ratcliffe, S Axler, and KA Ribet. *Foundations of hyperbolic manifolds*, volume 149. Springer, 1994. 3

[26] Rik Sarkar. Low distortion delaunay embedding of trees in hyperbolic plane. In *Proceedings of the 19th International Conference on Graph Drawing*, GD'11, page 355–366, Berlin, Heidelberg, 2011. Springer-Verlag. 2

[27] John Shawe-Taylor, Nello Cristianini, et al. *Kernel methods for pattern analysis*. Cambridge university press, 2004. 4

[28] Sindi Shkodrani, Yu Wang, Marco Manfredi, and Nóra Baka. United we learn better: Harvesting learning improvements from class hierarchies across tasks. *CoRR*, abs/2107.13627, 2021. 8

[29] Hugues Thomas, Charles R Qi, Jean-Emmanuel Deschaud, Beatriz Marcotegui, François Goulette, and Leonidas J Guibas. Kpconv: Flexible and deformable convolution for point clouds. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 6411–6420, 2019. 2

[30] Laurens van der Maaten and Geoffrey Hinton. Visualizing data using t-SNE. *Journal of Machine Learning Research*, 9:2579–2605, 2008. 8

[31] Dingkang Wang and Yusu Wang. An improved cost function for hierarchical cluster trees. *Journal of Computational Geometry*, 11(1):283–331, 2020. 2

[32] Hao Wang, Yitong Wang, Zheng Zhou, Xing Ji, Dihong Gong, Jingchao Zhou, Zhifeng Li, and Wei Liu. Cosface: Large margin cosine loss for deep face recognition. In *2018 IEEE*

Conference on Computer Vision and Pattern Recognition, CVPR 2018, Salt Lake City, UT, USA, June 18-22, 2018, pages 5265–5274. Computer Vision Foundation / IEEE Computer Society, 2018. 4

[33] Yue Wang, Yongbin Sun, Ziwei Liu, Sanjay E. Sarma, Michael M. Bronstein, and Justin M. Solomon. Dynamic graph CNN for learning on point clouds. *ACM Transactions on Graphics (ToG)*, 38(5), oct 2019. 2

[34] Yifan Xu, Tianqi Fan, Mingye Xu, Long Zeng, and Yu Qiao. Spidercnn: Deep learning on point sets with parameterized convolutional filters. In *Proceedings of the European conference on computer vision (ECCV)*, pages 87–102, 2018. 2

[35] Li Yi, Vladimir G Kim, Duygu Ceylan, I-Chao Shen, Mengyan Yan, Hao Su, Cewu Lu, Qixing Huang, Alla Sheffer, and Leonidas Guibas. A scalable active framework for region annotation in 3D shape collections. *ACM Transactions on Graphics (ToG)*, 35(6):1–12, 2016. 5