Self-Ordering Point Clouds

Pengwan Yang, Cees G. M. Snoek*, Yuki M. Asano*

University of Amsterdam

`p.yang3@uva.nl`

Abstract

In this paper we address the task of finding representative subsets of points in a 3D point cloud by means of a point-wise ordering. Only a few works have tried to address this challenging vision problem, all with the help of hard to obtain point and cloud labels. Different from these works, we introduce the task of point-wise ordering in 3D point clouds through self-supervision, which we call self-ordering. We further contribute the first end-to-end trainable network that learns a point-wise ordering in a self-supervised fashion. It utilizes a novel differentiable point scoring-sorting strategy and it constructs an hierarchical contrastive scheme to obtain self-supervision signals. We extensively ablate the method and show its scalability and superior performance even compared to supervised ordering methods on multiple datasets and tasks including zero-shot ordering of point clouds from unseen categories.

1. Introduction

The goal of this paper is to order the points in a 3D point cloud, so as to enable compute-, memory- and communication-efficient deployment. This is relevant for autonomous driving [11, 44], scene understanding [4, 29], and virtual reality [42, 16], to name just three of the many applications. The key problem of this modality of data is the sheer dimensionality of the input: while each point is usually only 3-dimensional, the number of points per scene or object can easily range from thousands to even millions [13, 43]. As even the original data often contains highly redundant points, methods have been employed to reduce the number of points for particular tasks to a more manageable size. These include simple heuristics such as picking random points [31] or selecting the ones with large Euclidean distances [24, 30, 32]. More recently, methods have been proposed that order points based on their importance [46, 14, 22]. For example, Lang *et al.* [22] utilize a supervised network to optimize for smaller subsets of
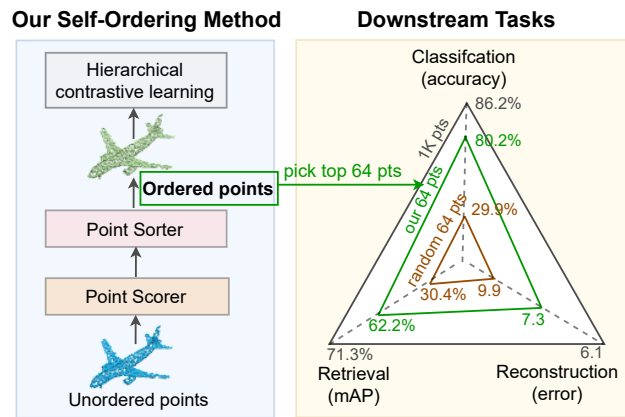

Figure 1: **Self-supervised learning to order point clouds.** We develop the first self-supervised method to order 3D points in terms of their importance. By utilizing a novel Point Scorer and differentiable Point Sorter module, we leverage a hierarchical contrastive loss for training to output a point-wise ordering without the need for any labels. This can be used to select subsets that not only outperform previous heuristic methods but even the current, supervised, state of the art on downstream tasks.

point clouds that can still yield good performance on various tasks. However, this method requires supervised annotations in the form of labels, which carry with it two main disadvantages. First, is the limited scalability – as obtaining human-annotations, as well as developing a systematic annotation scheme – is expensive. Second, labelling data captured in-the-wild (such as from a vehicle-mounted LiDAR sensor) is highly ambiguous if there is more than one object present: given a point cloud with 10K points, finding which one belongs to the street, a car, or another car is both difficult and tedious. To avoid the label-dependency of existing work, we propose in this paper the first self-supervised point cloud sorter (see Figure 1).

For self-supervised learning in the image domain, techniques such as clustering [6, 3] and contrastive learning [40, 19, 8] have recently shown great potential in learn-

ing generalizeable representations by designing suitable unsupervised loss functions. However, these methods are not directly applicable to the problem of point ordering due to two main differences. First, compared to the image domain, our goal is not representation learning *per-se*, but rather to leverage an unsupervised training signal for finding a reduced set of points that are representative for many downstream applications. Indeed, the difficulty lies in designing an effective module that can conduct the ordering – the key trainable component in our method – in a differentiable manner. Finally, in particular for contrastive learning, the generation of positive pairs is of key importance. While in the image domain this is trivially accomplished with augmentations, for point clouds this is generally difficult [9] and further complicated for the important case of having a very low number of points.

In this paper, we address these two challenges by ordering points in a 3D point cloud in a self-supervised manner – a setting we call *self-ordering*. The first challenge we address by front-loading a novel differentiable point scoring and sorting module that produces subsets, for which the self-supervised loss is then used for tuning. The second challenge we solve by naturally setting different strict subsets of particular point clouds as positive pairs – a choice that arises naturally and leads to a simple but novel hierarchical contrastive loss. As the first to tackle this setting in an unsupervised manner, we show that our method can not only produce good orderings of points, but even outperform supervised methods, when evaluated on a range of downstream tasks and datasets.

Overall, we make three contributions in this work.

- We propose the setting of self-ordering point clouds by self-supervised learning, in which each point in a 3D cloud is ranked according to its importance without relying on any annotations.

- We develop an end-to-end trainable approach to learn a point-wise ordering. For this, we develop a novel differentiable point scoring-sorting strategy and construct a hierarchical contrastive scheme to obtain self-supervision signals.

- We extensively ablate the method and show its scalability and superior performance even compared to supervised point ordering methods on multiple datasets and tasks including zero-shot ordering of point clouds.

Code: `https://github.com/PengWan-Yang/self-ordering`

2. Related work

Self-supervised point cloud representations. Most self-supervised learning works focus on learning single, global 3D object representation with applications to classic vision tasks, *e.g.*, classification [20, 18, 34], reconstruction [36, 18, 25], or part/semantic segmentation [20, 15, 17]. Recent works start to demonstrate potential for more high-level tasks or complex scenes, *e.g.*, Mersch *et al*. [27] predicted future 3D point clouds given a sequence of point clouds. Zhang *et al*. [45] proposed a self-supervised method to build representations for scene level point clouds without the need for multiple-views of the scene. Wang *et al*. [37] proposed a self-supervised schema to learn 4D spatio-temporal representations from dynamic point cloud clips by predicting their sequential orders without any ground-truth point-wise annotations. In this paper, our goal is not to learn a generic representation of point clouds but instead leverage self-supervision for the task of point cloud ordering.

Point cloud contrast construction. Unlike conventional data structures such as images or videos, raw point clouds are unordered sets of vectors. This difference naturally poses challenges for designing contrastive learning methods on point clouds. There are three main ways of creating positive and negative pairs (contrasts) for learning: 1) use contiguous chunks of a 3D point cloud as positives [15, 25]; 2) treat geometric transformation as a second view to generate positives [20]; 3) use natural or synthetic temporal sequences of point clouds [20, 23] as positives. Most works used one or two of them, *e.g.*, Sanghi *et al*. [34] use local chunks of 3D objects along with geometric transformation to build an effective contrastive learning. We do not use the common methods for contrast construction. Instead, we build a hierarchy of point clouds to generate a contrastive loss that operates on subsets inside the point cloud. These subsets in turn are not fixed or predefined, but are learned from our scorer network, and could thus be seen as a data-dependent view generator.

Differentiable top-k with optimal transport. Several works [41, 12, 5, 41] studied the introduction of a regularization term in the optimization to make a selection operator differentiable from the perspective of optimal transport. Specifically, Cuturi *et al*. [12] propose an optimal transport problem for finding the k-quantile in a set of n elements, while Xie *et al*. [41] parameterize the top-k operation as an optimal transport problem. Cordonnier *et al*. [10] implement a differentiable top-k operator in the image domain to select the most relevant parts of the input to efficiently process high resolution images. Inspired by the above technical algorithmic contributions, we treat point sorting as an optimal transport problem. To bring these techniques to the 3D domain, we develop novel point scorer and sorter modules which are tailored to point clouds as inputs and allow for differentiable ranking of points in a cloud. Our novelty is not to improve the top-k selection theory but to apply it to a novel realistic problem using self-supervision.

Point ordering with label supervision. Different from point cloud completion [38] and reconstruction [1], which
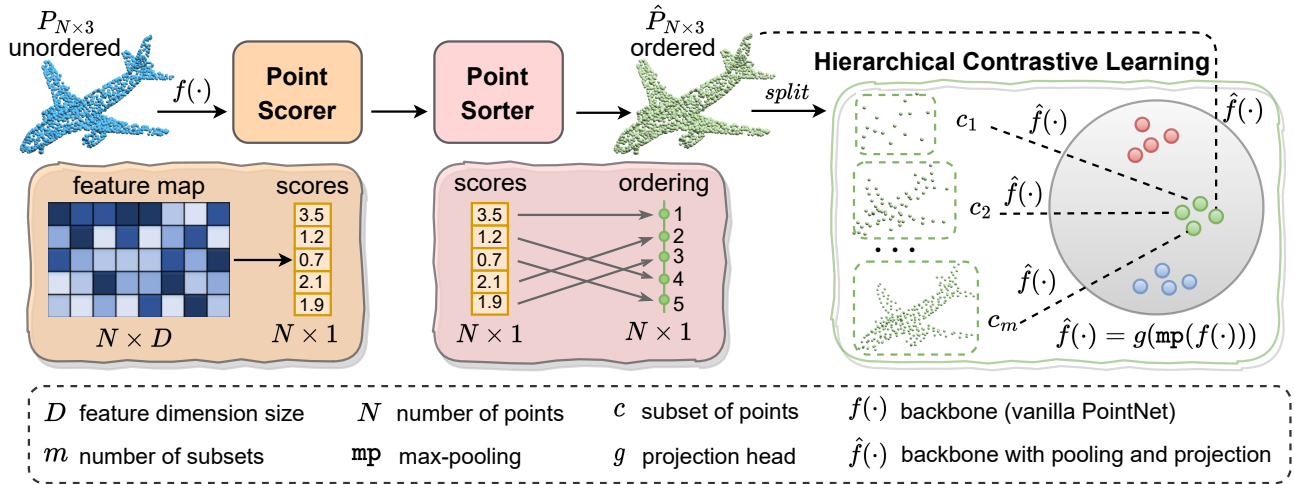
Figure 2: **Overview of the proposed model.** The Point Scorer network predicts a score for each point in the 3D point cloud. The Point Sorter module transfers the scores of points into a ranking of points, and the Hierarchical Contrastive Learning module forces the ranking to be correct. Note that we order the point cloud without the need for any label supervision.

change the point and cloud values, point ordering does not change the point and cloud values but reorders the points in a cloud. So far, ordering points in a cloud has only been addressed using supervised data [46, 14, 22]. Zheng *et al.* [46] assign each point a score by calculating the contribution of each point to the gradient of the loss function in a pre-trained model. Dovrat *et al.* [14] utilize a pre-trained task network as a teacher to optimize an ordered transformation of the original point cloud, then project the ordering information from the ordered transformation into the original point cloud. Lang *et al.* [22] follow Dovrat *et al.* in utilizing a pre-trained task network but introduce a differentiable projection tool to obtain an even more precise ordering. We are inspired by both Dovrat *et al.* and Lang *et al.*, as we tackle the same task, but rather than using labels we order point clouds without any supervision.

3. Method

Problem formulation. We seek to order points in 3D clouds using self-supervision. We denote a point cloud as $P=\{p_i\}_{i=1}^N$ with $p_i \in \mathbb{R}^3$. Our goal is to find an ordering of the points $\gamma^*=(i_1, i_2, \cdots, i_N)$ solely using an unlabeled dataset of point clouds that minimizes the downstream objective function ϕ:

$$\gamma^* = \arg\min_{\gamma} \phi(S_n), \qquad (1)$$

where various subsets of $S_n=\{p_{i_k}\}_{k=1}^n$, contain the top $n\ (\leq N)$ ranked points.

The challenge of this objective is two-fold. First, the objective is defined in terms of a downstream objective

function, yet the ordering γ^* is generated solely from unlabeled data P. Note that this means our task is two levels of abstraction away from the downstream task: we neither have labels per point cloud, nor do we have point-level annotations. Second, finding a right permutation is fundamentally a non-differentiable operation and directly learning this with gradient descent is difficult due to the discrete nature of permutations [46, 35].

In our method, we tackle both of these difficulties in turn by a) constructing a hierarchical self-supervised loss operating on subsets of 3D point clouds and b) constructing these via a differentiable scoring-sorting procedure.

Model overview. As illustrated in Figure 2, our model consists of a *Point Scorer* network, a *Point Sorter* module, and a *Hierarchical Contrastive Learning* module. The Point Scorer outputs a floating point value for each point in the 3D point cloud, the Point Sorter module orders the points according to these scores in a differentiable manner, and the Hierarchical Contrastive Learning module provides the necessary gradients to ensure the ordering is meaningful. The three modules are described in more detail below.

Point Scorer. The scorer is designed for scoring the points in the cloud. It thus provides a mapping f from a point cloud to a score vector, $f : P \to \mathbb{R}^{|P|}$. More specifically, the original point cloud P is firstly fed to a point cloud backbone $f(\cdot)$, yielding a feature map $\mathcal{F} \in \mathbb{R}^{|P| \times D}$, where D is the feature dimension size. Next, the goal is to compute a score for each point from these features. For this, we denote $\mathcal{F}_i=\{f_{i1}, f_{i2}, \cdots, f_{iD}\}$ as the D-dimensional feature of the i-th point, and f_{ij} as the ij-th element.

To arrive at a score for a point p, we compute the contribution of a point to the global feature of the point cloud

P. For a PointNet architecture [31], this global feature $\mathcal{G} = \{g_1, g_2, \cdots, g_D\}$ is computed by an order-invariant max pooling operation, *i.e.*,

$$g_j = \max_{i=1,2,\cdots,N} f_{ij}, \quad \text{for } j \in \{1, 2, \cdots, D\}, \quad (2)$$

or simply $\mathcal{G} = \text{mp}(\mathcal{F}, dim=0)$. Let idx_j denote the point which has the maximum value in the j-th dimension, *i.e.*,

$$\text{idx}_j = \arg\max_{i=1,2,\cdots,N} f_{ij}, \quad \text{for } j \in \{1, 2, \cdots, D\}, \quad (3)$$

The simplest metric to compute the importance of a point i to the final feature would be to count how often its features ended up in the global feature:

$$s_i = \frac{1}{D} \sum_{j=1}^{D} \delta(\text{idx}_j, i), \quad (4)$$

where s_i is the score of i-th point and $\delta(x, y)$ is a Kronecker-delta function, *i.e.*, $\delta(x, y)=0$ if $x \neq y$ and 1 if $x=y$. Intuitively, if one point's feature is descriptive of the whole global feature, it achieves a score of 1.0.

To enable backpropagation, we replace the operations of Equations 3-4 with a differentiable approximation which we formulate as:

$$s_i = \frac{1}{D} \sum_{j=1}^{D} 2 \cdot \sigma(f_{ij} - g_j), \quad (5)$$

where σ is a sigmoid operation with temperature τ, *i.e.*, $\sigma(x)=\frac{1}{1+e^{-x/\tau}}$. By scaling with 2, the sigmoid outputs arrive at the interval [0, 1]. Again, it holds that if one point's feature $\mathcal{F}_i$ is equal to the global descriptor $\mathcal{G}$, $s_i=1.0$. Note that from Equation 5, $s_i \propto \Sigma_j(\sigma(f_{ij} - g_j))$, leading to the simple calculation of the score vector for all points as $\mathcal{S}=\text{sum}(\sigma(\mathcal{F} - \mathcal{G}), \text{dim}=1)$.

Point Sorter. The task of this module is to sort the points according to their scores in a differentiable manner. For this, we apply a Top-K operator utilising optimal transport [41, 12]. We first parameterize point sorting in terms of an optimal transport problem. We try to find a transport plan from discrete distribution $\mathcal{S}=[s_1, s_2, \cdots, s_N]^T$ to discrete distribution $\mathcal{B}=[1, 2, \cdots, N]^T$. For this, we specify the marginals for both $\mathcal{S}$ and $\mathcal{B}$ as $\mu=\nu=\mathbf{1}_N/N$ and further denote $C \in \mathbb{R}^{N \times N}$ as the cost matrix with C_{ij} being the cost of transporting mass from s_i to b_j, which is the j-th element in $\mathcal{B}$. We take the cost to be the squared Euclidean distance, *i.e.*, $C_{ij} = (s_i - j)^2$. Given this formulation, the optimal transport problem can be formulated as:

$$\Gamma^* = \arg\min_{\Gamma \geq 0} \langle C, \Gamma \rangle + \epsilon h(\Gamma),$$
$$\text{s. t., } \Gamma\mathbf{1}_N = \mu, \Gamma^T\mathbf{1}_N = \nu, \quad (6)$$

where $\langle \cdot \rangle$ is the inner product, $h(\Gamma)=\sum_{ij} \Gamma_{ij} \log \Gamma_{ij}$ is an entropy regularizer which can rule out the discontinuity and obtain a smoothed and differentiable approximation to the Top-K operator. Then the approximate optimal Γ^* is referred to as the optimal transport plan. Finally, by scaling with N, we obtain $\gamma^*=N\Gamma^* \cdot \mathcal{B}$ as the ordering of the points, where the point with the highest score is set to 1, the point of second biggest score to 2, and so on. This yields the sorted/ordered point cloud $\hat{P} \in \mathbb{R}^{N \times 3}$ for the original unordered point cloud P.

Hierarchical Contrastive Learning. Now that we have a differentiable method for obtaining an ordered point cloud $\hat{P}$, the next step is to optimize this using self-supervision. We use a hierarchical scheme inside the ordered point cloud as a free supervision signal.

For this, we define a set of subsets with exponentially increasing cardinality, *i.e.*, $\mathcal{C}=\{c_k\}_{k=1}^{m}$, with $|c_k|=\theta^k$, $m=\log_\theta(N)$ and $\forall k : c_k \subset c_{k+1}$. Here, θ governs the growth rate of the subset. The first subset contains the top θ points in $\hat{P}$, the second subset the top θ^2 points and so on. We treat subsets $\mathcal{C}=\{c_k\}_{k=1}^{m}$ of $\hat{P}$ as positive pairs while negative pairs are constructed from subsets of other point clouds. Then we naturally arrive at the following multiple-instance NCE formulation [28]:

$$\mathcal{L}_{\text{NCE}}^{c_k}(\hat{P}) = -\log \frac{\sum_{c_k^+} \exp(\langle \hat{f}(\hat{P}), \hat{f}(c) \rangle / \varphi)}{\sum_{c_k^+ \cup c_k^-} \exp(\langle \hat{f}(\hat{P}), \hat{f}(c) \rangle / \varphi)}, \quad (7)$$

where c_k^+ is the positive set and c_k^- is the negative set for subset c_k. $\hat{f}(\cdot)=g(\text{mp}(f(\cdot)))$ is a procedure consisting of the same backbone f in the scorer network, a max-pooling operation mp and a projection head g. It can project the pooled features of point subsets into the shared latent space. The overall loss used for training is simply given by:

$$\mathcal{L}(\hat{P}) = \sum_{k=1}^{m} \mathcal{L}_{\text{NCE}}^{c_k}(\hat{P}). \quad (8)$$

Due to the increasing cardinality of the subsets, the top points will be used more often in the contrastive loss, effectively scaling their importance for the total loss and leading to scores that put the most contrastively informative points at the top.

4. Experimental setup

Datasets. We evaluate our approach on three datasets. ***ModelNet40*** [39] contains 40 categories from CAD models. We use the official split with 9,843 point clouds for training and 2,468 for testing. As the starting and maximum set, we uniformly sample 1,024 points from the mesh surface of each CAD model. ***3D MNIST*** [33] contains 6,000 raw point clouds, with each cloud containing 20,000 points. To enrich this dataset, we

	Base Tasks		
	Classification	Retrieval	Reconstruction
Training data	ModelNet40	ModelNet40	ModelNet40
Test data	ModelNet40	3D MNIST	ShapeNet Core55
Base network	PointNet [31]	PointNet++ [32]	autoencoder [1]
Metric	accuracy	mAP	reconstruction error

Table 1: **Evaluation scheme**. We evaluate across three downstream tasks, three datasets, and three base networks.

randomly select 1,024 points from each raw point cloud 10 times, resulting in a training set of size 50,000 and a test set of size 10,000, with each cloud consisting of 1,024 points. *ShapeNet Core55* [7] covers 55 object categories with 51,300 3D point clouds, with train/test sets following a 85%/15% split. We sample 1,024 points from the surface of each model to generate the initial set of data.

Tasks and evaluation. Our method generates an ordering of points in terms of their importance in a self-supervised manner. Given such an ordering, we consider three different downstream tasks for evaluation in an efficient manner: point cloud classification [31], retrieval [2], and reconstruction [1]. During training, we train our self-ordering approach on the ModelNet40 dataset without labels. During evaluation, we firstly train the task networks on the corresponding test dataset as shown in Table 1 and keep the trained task networks fixed. Then we evaluate on variously sized subsets of points generated by the self-ordering model by passing them to the frozen task networks. We report classification accuracy for classification, mean Average Precision (mAP) for retrieval, and the Chamfer distance [1] between the reconstructed points and original points for reconstruction, denoted as reconstruction error. Table 1 summarizes the evaluation tasks, test datasets, base networks for task and evaluation metrics.

Implementation details. For the Point Scorer network, we utilize a vanilla PointNet without any transformation layers (see Appendix for architectural details). We set the feature dimension size D to 2,048, instead of the default 1,024. We vary the exact shape of the sigmoid function $\sigma(\cdot)$ by scaling the input with a temperature τ (Equation 5). Unless otherwise noted, $\tau=0.5$. The Point Sorter is implemented using Sinkhorn-Knopp optimal transport, where entropy regularisation parameter $\epsilon=0.1$. The backbones in the Hierarchical Contrastive Learning share weights with the backbone in the Point Scorer network for passing information and efficiency. A 2-layer MLP projection head projects all subset representations to a 128-dimensional latent space for self-supervised contrastive learning. We set the subset control factor $\theta=2$ and the temperature φ in the NCE Loss function (Equation 7) as 0.7.

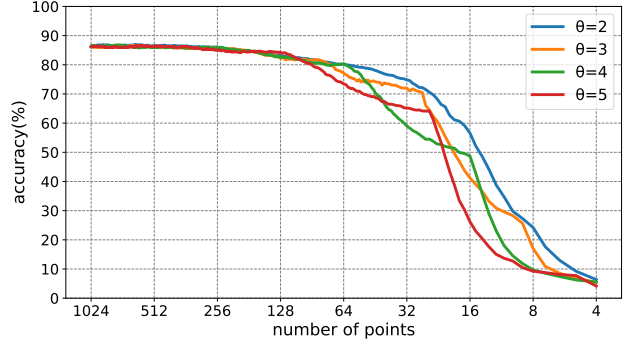**Training regime.** We train our self-ordering model, including the backbone, from scratch without any labels. We

Figure 3: **Subset size controller θ.** We evaluate for the classification task on ModelNet40. θ controls the size of the subsets for the Hierarchical Contrastive Learning. Having more subsets with a slower growth such as for $\theta=2$ yields superior performances without kinks at powers of θ.

use a batch size 128 for 250 epochs unless otherwise noted, and use the AdamW optimizer [21] with weight decay $1e-5$ and initialize the learning rate to $1e-4$ and decay the learning rate with the cosine decay schedule [26] to zero. Training on four Nvidia GTX 1080TI GPUs takes around 12 hours.

5. Results

5.1. Ablation study

Subset size controller θ. The primary hyperparameter in our approach is the growth rate of the subsets on which the self-supervised contrastive loss are applied. We evaluate its influence from two (smallest growth) to five (largest growth). We show the classification results on ModelNet40 using PointNet as backbone in Figure 3. We evaluate the classification effectiveness on point subsets across different sizes, ranging from all 1,024 points to the top 4 ranked points. We find that most of the performance is maintained with a cloud size from 1,024 to top 256 for all settings, indicating that only a subset of points is vital for classification. When retaining only the top 32 points, the accuracy is 73.5% for $\theta=2$, compared to 65.7%, 60.8%, 67.8% for respectively $\theta=3$, $\theta=4$, and $\theta=5$. The higher scores for the lowest θ value are a consequence of splitting the point cloud into more subsets in the Hierarchical Contrastive Learning. We conclude that using more subsets is beneficial for self-supervised ordering and use $\theta=2$ for all other experiments.

Sigmoid temperature τ. Next we ablate the sensitivity of the differential approximation in the Point Scorer to the sigmoid temperature τ. Table 2 shows results of various τ values with evaluation for the classification task on Model-Net40. We find that the performance of our method is quite stable with regards to this parameter even across orders of magnitude and find the best performance is achieved when

	number of points			
τ	16	32	64	128
5	51.2	72.2	79.2	81.9
2	52.4	73.1	79.6	82.4
1	52.7	73.3	79.8	82.6
0.5	52.8	73.5	80.2	82.7
0.1	52.9	73.4	79.8	82.3

Table 2: **Sigmoid temperature** τ. We evaluate for the classification task on the ModelNet40 dataset. Results are stable for $\tau \leq 2$, for all number of points. An optimal result is obtained when τ is set to 0.5.

	number of points					
D	16	32	64	128	CV	params(M)
512	51.2	72.1	79.0	81.8	2.1	1.2
1,024	52.3	72.9	79.9	82.4	1.7	1.7
2,048	52.8	73.5	80.2	82.7	1.3	2.6
4,096	52.9	73.8	80.4	82.8	1.2	4.3

Table 3: **Feature dimension size** D. We evaluate for the classification task on ModelNet40. Coefficient of Variance (CV) reflects the degree of dispersion among the point scores. Increasing dimension size D avoids point scores to cluster at a value of 0 and improves the performances over all number of points, but at the expense of increased parameter cost. Balancing performance and parameters, we use $D = 2,048$ for all experiments.

the sigmoid is sharpened by a small amount with $\tau = 0.5$, and use this value for all remaining experiments.

Feature dimension size D. The point scores come from the D-dimensional feature. If the dimension size D is small, scores may degenerate to only a small subset of points with most points approaching the score of 0. A large D helps avoid this degeneration, however, at the expense of an increased number of parameters. In Table 3, we ablate the influence of D on the trade-of between performance and the amount of parameters. We evaluate for the classification task on ModelNet40 and include the Coefficient of Variance (CV), which reflects the degree of dispersion among the point scores. A high CV value means the points may aggregate around some lower score values. When dimension size D grows from 512 to 4,096, the CV value shrinks and performance increases. However, the CV gap between dimension of 2,048 and that of 4,098 is quite small, *i.e.*, 1.3 *vs.* 1.2, and downstream performances are also close between the two dimension sizes, *i.e.*, 82.7 *vs.* 82.8 when using the top 128 points. However, the parameters increase by 65% from dimension size 2,048 to 4,096. Hence, we prefer to use 2,048 as the default value for the feature dimension size.

	number of points			
	16	32	64	128
scoring by max-pooling	36.3	51.6	66.2	73.7
scoring by sum	43.6	64.7	73.3	76.4
proposed Point Scorer	52.8	73.5	80.2	82.7

Table 4: **Different point scoring methods**. We evaluate on classification for ModelNet40. Among three different methods our proposed Point Scorer module produces the point scores leading to the best point cloud ordering.

		number of points			
	sharing weights	16	32	64	128
backbone	no	48.4	70.6	78.3	81.6
	yes	52.8	73.5	80.2	82.7
head	no	51.7	72.5	79.4	82.1
	yes	52.8	73.5	80.2	82.7

Table 5: **Sharing weights** in backbones and projection heads. We evaluate on classification and ModelNet40. We compare between sharing or not sharing weights for both backbones and projection heads. The result shows sharing weights both in backbones and in projection heads benefits our self-ordering module.

Different point scoring methods. We tried three different methods to score the points: (i) we use a max-pooling operation on the feature map along the feature dimension, and obtain the max value of each point as point score; (ii) we use the sum of the feature value of each point as score; (iii) our proposed scorer (Section 3). Results for classification on ModelNet40 in Table 4 show our proposed scorer produces point scores with the best point cloud ordering.

Sharing weights. In Table 5 we compare between sharing and not sharing weights for the backbones (*i.e.* for the Point Scorer and Hierarchical Contrastive Learning and within the Hierarchical Contrastive Learning between different projection heads for the different subsets. We find that in both cases, sharing weights benefits point order learning, with the largest increase coming from a shared backbone between the scorer and the self-supervised loss module. This is likely because similar features are required for both tasks and sharing weights encourages more general features to be favored and reinforced.

Scalability of ordering ability. We analyze the scalability of our approach in Table 6. We train self-ordering on point clouds of 256, 512 and 1,024 points and order point clouds containing 1,024 points in testing. The results remain stable, which demonstrates that self-ordering trained on small point clouds can generalize to order bigger point clouds. This enables ordering big point clouds with limited run time and GPU memory storage.
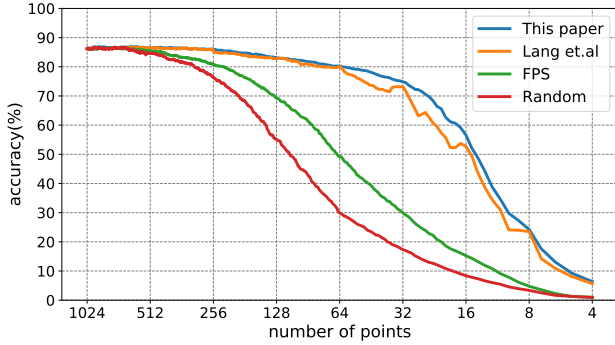
Figure 4: **Comparison on classification.** The instance classification accuracy is evaluated on consistent point numbers from all 1,024 points to the top 4 points. Our self-ordering method outperforms other ordering alternatives across all point numbers, including a fully-supervised alternative.
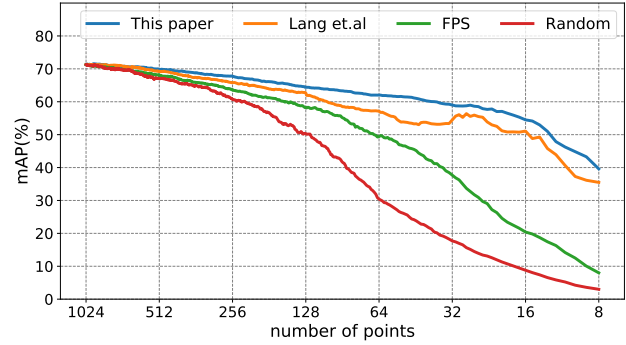

Figure 5: **Comparison on retrieval.** We measure retrieval mAP from all 1,024 points to the top 8 points based on the orderings. Our self-ordering performs better than the other three ordering methods independent of the number of points.

		Top #pts from 1,024				time	memory
	Training #pts	16	32	64	128	(hour)	(GB)
random selection	-	8.4	17.3	29.9	55.1	-	-
self-ordering	256	50.4	71.5	78.9	81.3	5.1	2.2
self-ordering	512	51.7	72.6	79.6	82.1	9.7	3.7
self-ordering	1,024	52.8	73.5	80.2	82.7	12.3	6.1

Table 6: **Scalability of ordering ability.** We evaluate the classification task on ModelNet40 by varying the number of input points for our self-supervised training and selecting top points for evaluation from the 1,024-sized point cloud.

	FLOPs/point cloud (M)	time/point cloud (millisecond)
PointNet++ (1024 pts)	1546	16.9
PointNet++ (128 pts)	198	2.3
Self-ordering module	137	1.7
Self-ordering + PointNet++ (128 pts)	335	3.9

Table 7: **Time and complexity reduction.** We evaluate for retrieval on 3D MNIST. Self-ordering makes for a great reduction in computation and time.

Time and computation reduction. Our time and compute advantage comes from using a small and cheap network to get the most important points and then give them to a larger model – as compared to feeding a large model all points – drastically reducing complexity and time. Table 7 quantifies this. When fed with a complete point cloud (1024 points), PointNet++ performs 1546M floating point multiplications (FLOPs). When feeding only top 128 points, FLOPs drop to 198M. Self-ordering requires 137M FLOPs to order the points. So the overall FLOPs is 335M, leading to a > 4x increase in throughput. Moreover, by replacing the entire cloud with the top 128 points, memory usage during inference can be reduced from 697MB to 122MB. This significant decrease enables the deployment of point cloud applications on mobile devices with limited resources.
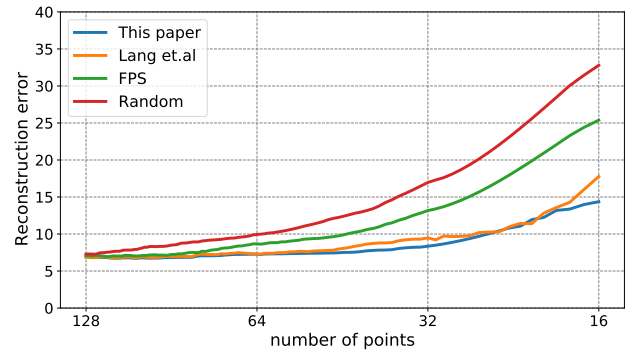

Figure 6: **Comparison on reconstruction.** We reconstruct from subsets of points selected by the orderings. From the top 128 points to the top 16 points, our self-ordering achieves a lower error compared to the alternatives.

5.2. Benchmarks

We compare to three other methods able to obtain subsets of point clouds. First, a baseline based on random point selection [31]; Second, a baseline based on Farthest Point Sampling (FPS) [24, 30, 32], which starts from a random point in the cloud, and iteratively selects the farthest point from the selected points; Third, a fully supervised point sorter proposed by Lang *et al*. [22]. Note that different from Lang *et al*., we rely on self-supervision without the need for any cloud class labels or point-wise annotations prior to generating the ordering.

Classification. In Figure 4, we show the comparison for classification on ModelNet40. We find our approach consistently outperforms the alternatives, independent of the number of points sampled for the downstream task. We even improve over the supervised point sorting by Lang *et al*.
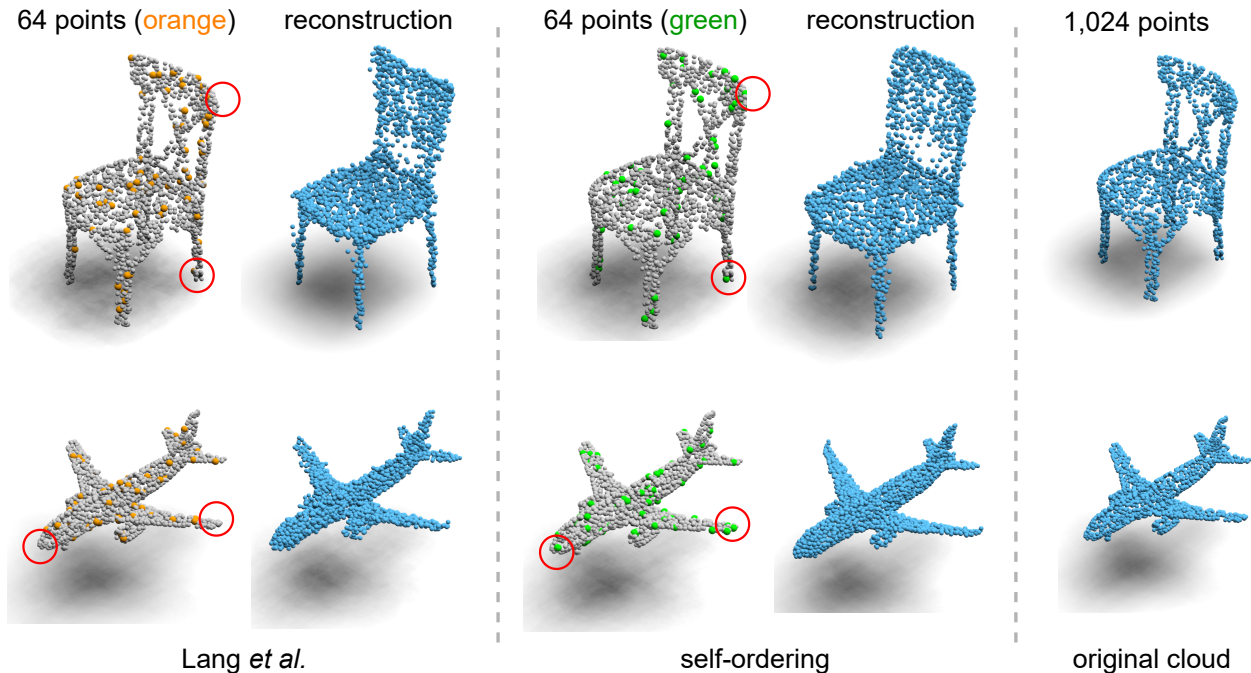
Figure 7: Qualitative reconstruction results. We show our ability to recover the full, 1,024 points from only 64 points and the reconstruction from the supervised subset of Lang *et al.* Our top 64 points (green points) seem to pay more attention to the salient parts of objects compared with the top 64 points (orange points) from the ordering of Lang *et al.* (highlight in red circle). This is reflected by the overall better reconstructions from our points when compared to the original. Further qualitative results are provided in the Appendix.

Retrieval. Figure 5 presents the comparison for retrieval on 3D MNIST. We experiment on retrieval from all points to the top 8 points based on various orderings. We observe considerably better performance when applying our self-ordering across all point numbers. For example, at top 8 points, we can achieve 39.6% mAP, above the performance of 35.5% obtained by the supervised method by Lang *et al.*

Reconstruction. In Figure 6 we compare self-ordering with the three alternatives on the reconstruction task. A complete point cloud is reconstructed from the top 128 points up to the top 16 points based on the orderings. Our self-ordering achieve the lowest error across almost all point numbers, except for the top 24 and top 21 points, where the supervised method by Lang *et al.* overtakes self-ordering (10.4 *vs.* 10.3 and 11.9 *vs.* 11.4). In Figure 7, we show two examples of reconstructed point clouds from only 64 selected points, as provided by our self-ordering and the supervised ordering of Lang *et al.* We observe our top 64 points seem to pay more attention to the salient parts of objects compared with the top 64 points from the ordering of Lang *et al.* And, our reconstructed shapes are closer to the original shapes. The reconstructed chair of Lang *et al.* becomes more square compared to the original round chair, while our reconstruction maintains curved

edges. The reconstructed airplane of Lang *et al.* has fuzzier edges compared with the original airplane cloud while our reconstructed airplane retains a sharp shape. The result further demonstrates the reconstruction ability of our method compared to the supervised alternative.

Zero-shot transfer. Finally, we evaluate the transfer performance of our self-ordering network. For this, we train on one dataset and evaluate the quality of the ordering of points it generates for a different dataset. We artificially create dataset splits that do not share common classes. To be precise, for ModelNet40 we generate a random 30/10-classes split. For the retrieval and reconstruction tasks, we remove the overlapping classes between the training and test datasets. All class splits are listed in the Appendix. Experiments across three tasks and three datasets in Table 8 show that the performances of the top points remain high even after removing the overlapping classes between the training and testing data. This shows that the self-ordering network has learned general features that can be used to transfer in a zero-shot fashion to unseen classes.

6. Conclusion

In this paper we tackled the problem of ordering points in a 3D point cloud such that their ranking can be used

	scenario	number of points			
		16	32	64	128
Classification (*accuracy*)					
random selection	-	8.4	17.3	29.9	55.1
self-ordering	base	52.8	73.5	80.2	82.7
	zero-shot	50.5	71.8	79.0	82.2
Retrieval (*mAP*)					
random selection	-	8.8	17.7	30.4	50.3
self-ordering	base	54.5	59.0	62.2	64.7
	zero-shot	52.9	58.3	61.4	64.0
Reconstruction (*error*)					
random selection	-	32.8	17.0	9.9	7.3
self-ordering	base	14.4	8.4	7.3	6.9
	zero-shot	15.1	8.9	7.7	7.0

Table 8: **Zero-shot transfer.** We remove overlapping classes between training data and test data for the zero-shot scenario. The performance remains competitive when evaluating in this zero-shot point ordering scenario across three tasks and three datasets, demonstrating the generalization ability of our self-ordering.

for selecting smaller subsets whilst retaining performance. For this, we proposed a novel method that utilises self-supervision to train a differentiable point orderer and evaluated its performance across three downstream tasks and datasets. The result is a principled, yet simple method that surpasses previous point selection heuristics by a large margin and even outperforms a supervised counterpart and enables ordering big point clouds in compute- and memory-constrained environments, showing the large potential that exists for self-supervised learning in point clouds.

References

[1] Panos Achlioptas, Olga Diamanti, Ioannis Mitliagkas, and Leonidas Guibas. Learning representations and generative models for 3d point clouds. *arXiv*, 2017.

[2] Mikaela Angelina Uy and Gim Hee Lee. Pointnetvlad: Deep point cloud based retrieval for large-scale place recognition. In *CVPR*, 2018.

[3] YM Asano, C Rupprecht, and A Vedaldi. Self-labelling via simultaneous clustering and representation learning. In *ICLR*, 2019.

[4] Jens Behley, Martin Garbade, Andres Milioto, Jan Quenzel, Sven Behnke, Cyrill Stachniss, and Jurgen Gall. Semantickitti: A dataset for semantic scene understanding of lidar sequences. In *ICCV*, 2019.

[5] Nicolas Bonneel, Gabriel Peyré, and Marco Cuturi. Wasserstein barycentric coordinates: histogram regression using optimal transport. *ACM Transactions on Graphics*, 2016.

[6] Mathilde Caron, Piotr Bojanowski, Armand Joulin, and Matthijs Douze. Deep clustering for unsupervised learning of visual features. In *ECCV*, 2018.

[7] Angel X Chang, Thomas Funkhouser, Leonidas Guibas, Pat Hanrahan, Qixing Huang, Zimo Li, Silvio Savarese, Mano-

lis Savva, Shuran Song, Hao Su, et al. Shapenet: An information-rich 3d model repository. *arXiv*, 2015.

[8] Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. A simple framework for contrastive learning of visual representations. In *ICML*, 2020.

[9] Yunlu Chen, Vincent Tao Hu, Efstratios Gavves, Thomas Mensink, Pascal Mettes, Pengwan Yang, and Cees GM Snoek. Pointmixup: Augmentation for point clouds. In *ECCV*, 2020.

[10] Jean-Baptiste Cordonnier, Aravindh Mahendran, Alexey Dosovitskiy, Dirk Weissenborn, Jakob Uszkoreit, and Thomas Unterthiner. Differentiable patch selection for image recognition. In *CVPR*, 2021.

[11] Yaodong Cui, Ren Chen, Wenbo Chu, Long Chen, Daxin Tian, Ying Li, and Dongpu Cao. Deep learning for image and point cloud fusion in autonomous driving: A review. *TIP*, 2021.

[12] Marco Cuturi, Olivier Teboul, and Jean-Philippe Vert. Differentiable ranking and sorting using optimal transport. *NeurIPS*, 2019.

[13] Angela Dai, Angel X Chang, Manolis Savva, Maciej Halber, Thomas Funkhouser, and Matthias Nießner. Scannet: Richly-annotated 3d reconstructions of indoor scenes. In *CVPR*, 2017.

[14] Oren Dovrat, Itai Lang, and Shai Avidan. Learning to sample. In *CVPR*, 2019.

[15] Benjamin Eckart, Wentao Yuan, Chao Liu, and Jan Kautz. Self-supervised learning on 3d point clouds by learning discrete generative models. In *CVPR*, 2021.

[16] Mohamed El Beheiry, Sébastien Doutreligne, Clément Caporal, Cécilia Ostertag, Maxime Dahan, and Jean-Baptiste Masson. Virtual reality: beyond visualization. *Journal of molecular biology*, 2019.

[17] Kexue Fu, Peng Gao, Renrui Zhang, Hongsheng Li, Yu Qiao, and Manning Wang. Distillation with contrast is all you need for self-supervised point cloud representation learning. *arXiv*, 2022.

[18] Kaveh Hassani and Mike Haley. Unsupervised multi-task feature learning on point clouds. In *ICCV*, 2019.

[19] Kaiming He, Haoqi Fan, Yuxin Wu, Saining Xie, and Ross Girshick. Momentum contrast for unsupervised visual representation learning. In *CVPR*, 2020.

[20] Siyuan Huang, Yichen Xie, Song-Chun Zhu, and Yixin Zhu. Spatio-temporal self-supervised representation learning for 3d point clouds. In *ICCV*, 2021.

[21] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *ICLR*, 2015.

[22] Itai Lang, Asaf Manor, and Shai Avidan. SampleNet: Differentiable point cloud sampling. In *CVPR*, 2020.

[23] Ruibo Li, Guosheng Lin, and Lihua Xie. Self-point-flow: Self-supervised scene flow estimation from point clouds with optimal transport and random walk. In *CVPR*, 2021.

[24] Yangyan Li, Rui Bu, Mingchao Sun, Wei Wu, Xinhan Di, and Baoquan Chen. Pointcnn: Convolution on x-transformed points. In *NeurIPS*, 2018.

[25] Xinhai Liu, Xinchen Liu, Yu-Shen Liu, and Zhizhong Han. Spu-net: Self-supervised point cloud upsampling by coarse-

to-fine reconstruction with self-projection optimization. *TIP*, 2022.

[26] Ilya Loshchilov and Frank Hutter. Sgdr: Stochastic gradient descent with warm restarts. *arXiv*, 2016.

[27] Benedikt Mersch, Xieyuanli Chen, Jens Behley, and Cyrill Stachniss. Self-supervised point cloud prediction using 3d spatio-temporal convolutional networks. In *Conference on Robot Learning*. PMLR, 2022.

[28] Antoine Miech, Jean-Baptiste Alayrac, Lucas Smaira, Ivan Laptev, Josef Sivic, and Andrew Zisserman. End-to-end learning of visual representations from uncurated instructional videos. In *CVPR*, 2020.

[29] Yinyu Nie, Ji Hou, Xiaoguang Han, and Matthias Nießner. Rfd-net: Point scene understanding by semantic instance reconstruction. In *CVPR*, 2021.

[30] Charles R Qi, Or Litany, Kaiming He, and Leonidas J Guibas. Deep hough voting for 3d object detection in point clouds. In *ICCV*, 2019.

[31] Charles R Qi, Hao Su, Kaichun Mo, and Leonidas J Guibas. Pointnet: Deep learning on point sets for 3d classification and segmentation. In *CVPR*, 2017.

[32] Charles Ruizhongtai Qi, Li Yi, Hao Su, and Leonidas J Guibas. Pointnet++: Deep hierarchical feature learning on point sets in a metric space. In *NeurIPS*, 2017.

[33] Danilo Jimenez Rezende, SM Ali Eslami, Shakir Mohamed, Peter Battaglia, Max Jaderberg, and Nicolas Heess. Unsupervised learning of 3d structure from images. In *NeurIPS*, 2016.

[34] Aditya Sanghi. Info3d: Representation learning on 3d objects using mutual information maximization and contrastive learning. In *ECCV*, 2020.

[35] Rodrigo Santa Cruz, Basura Fernando, Anoop Cherian, and Stephen Gould. Deeppermnet: Visual permutation learning. In *CVPR*, 2017.

[36] Jonathan Sauder and Bjarne Sievers. Self-supervised deep learning on point clouds by reconstructing space. *NeurIPS*, 2019.

[37] Haiyan Wang, Liang Yang, Xuejian Rong, Jinglun Feng, and Yingli Tian. Self-supervised 4d spatio-temporal feature learning via order prediction of sequential point cloud clips. In *WACV*, 2021.

[38] Yida Wang, David Joseph Tan, Nassir Navab, and Federico Tombari. Softpool++: An encoder–decoder network for point cloud completion. *IJCV*, 2022.

[39] Zhirong Wu, Shuran Song, Aditya Khosla, Fisher Yu, Linguang Zhang, Xiaoou Tang, and Jianxiong Xiao. 3d shapenets: A deep representation for volumetric shapes. In *CVPR*, 2015.

[40] Zhirong Wu, Yuanjun Xiong, Stella X Yu, and Dahua Lin. Unsupervised feature learning via non-parametric instance discrimination. In *CVPR*, 2018.

[41] Yujia Xie, Hanjun Dai, Minshuo Chen, Bo Dai, Tuo Zhao, Hongyuan Zha, Wei Wei, and Tomas Pfister. Differentiable top-k with optimal transport. *NeurIPS*, 2020.

[42] Siyang Yu, Si Sun, Wei Yan, Guangshuai Liu, and Xurui Li. A method based on curvature and hierarchical strategy for dynamic point cloud compression in augmented and virtual reality system. *Sensors*, 2022.

[43] Amir R Zamir, Tilman Wekel, Pulkit Agrawal, Colin Wei, Jitendra Malik, and Silvio Savarese. Generic 3d representation via pose estimation and matching. In *ECCV*, 2016.

[44] Yiming Zeng, Yu Hu, Shice Liu, Jing Ye, Yinhe Han, Xiaowei Li, and Ninghui Sun. Rt3d: Real-time 3-d vehicle detection in lidar point cloud for autonomous driving. *Robotics and Automation Letters*, 2018.

[45] Zaiwei Zhang, Rohit Girdhar, Armand Joulin, and Ishan Misra. Self-supervised pretraining of 3d features on any point-cloud. In *ICCV*, 2021.

[46] Tianhang Zheng, Changyou Chen, Junsong Yuan, Bo Li, and Kui Ren. Pointcloud saliency maps. In *ICCV*, 2019.