Data Augmentation-free Unsupervised Learning for 3D Point Cloud Understanding

Guofeng Mei[1]
guofeng.mei@student.uts.edu.au
Cristiano Saltori[2]
cristiano.saltori@unitn.it
Fabio Poiesi[3]
poiesi@fbk.eu
Jian Zhang[1]
jian.zhang@uts.edu.au
Elisa Ricci[2]
e.ricci@unitn.it
Nicu Sebe[2]
niculae.sebe@unitn.it
Qiang Wu[1]
qiang.wu@uts.edu.au

[1] Global Big Data Technologies Centre
University of Technology Sydney
Sydney, Australia

[2] Multimedia and Human Understanding Group
University of Trento
Trento, Italy

[3] Technologies of Vision Lab
Fondazione Bruno Kessler
Trento, Italy

Abstract

Unsupervised learning on 3D point clouds has undergone a rapid evolution, especially thanks to data augmentation-based contrastive methods. However, data augmentation is not ideal as it requires a careful selection of the type of augmentations to perform, which in turn can affect the geometric and semantic information learned by the network during self-training. To overcome this issue, we propose an augmentation-free unsupervised approach for point clouds to learn transferable point-level features via soft clustering, named SoftClu. SoftClu assumes that the points belonging to a cluster should be close to each other in both geometric and feature spaces. This differs from typical contrastive learning, which builds similar representations for a whole point cloud and its augmented versions. We exploit the affiliation of points to their clusters as a proxy to enable self-training through a pseudo-label prediction task. Under the constraint that these pseudo-labels induce the equipartition of the point cloud, we cast SoftClu as an optimal transport problem. We formulate an unsupervised loss to minimize the standard cross-entropy between pseudo-labels and predicted labels. Experiments on downstream applications, such as 3D object classification, part segmentation, and semantic segmentation, show the effectiveness of our framework in outperforming state-of-the-art techniques [code].

1 Introduction

The rapid progress of 3D capturing devices, such as 3D laser scanners and depth sensors, led to convenient and effective ways to process 3D data, which, if combined with RGB images, can further improve the understanding of environments. Applications like robotic navigation [5, 56], autonomous driving [23] and exploration [47], and augmented and virtual reality [31] are among the major reasons for a higher attention toward 3D data understanding.

Learning discriminative and transferable point cloud features is a crucial problem in the area of 3D shape understanding [24, 33], as it allows efficient training of downstream tasks, such as object detection [42] and tracking [53], segmentation [50], reconstruction [55], classification [35], and registration [22, 28, 29]. Therefore, learning from unlabeled or partially labeled data to alleviate human labeling efforts is an emerging research topic in point cloud understanding. Along this line, unsupervised representation learning is an attractive, yet potent, alternative approach to learning features without human intervention [15].

Unsupervised learning approaches can be broadly categorized as generative or discriminative [17]. The former includes self-reconstruction or auto-encoding [51], generative adversarial network [38], and auto-regressive [44] methods. These methods can map an input point cloud into a global latent representation [36, 43], or a latent distribution in the variational case [18, 19] through an encoder and then attempt to reconstruct the input by a decoder. Generative methods can be effective to model high-level and structural properties of the input point clouds. However, because they are sensitive to Euclidean transformations, they typically assume that all 3D objects have the same pose in a given category [37].

Unlike generative methods, discriminative methods learn to predict or discriminate augmented versions of the input. These methods can yield rich latent representations for downstream tasks [46]. Examples of these include contrastive methods [14, 36, 37], which have shown remarkable results for unsupervised representation learning. Contrastive methods also promote learning of rotation-invariant representations via data augmentation [34]. Typically, these algorithms require several negative samples and heavily depend on the selection criteria to mine negatives [11, 17]. Often, they require large batch sizes, memory banks, or customized strategies to retrieve informative pairs [17]. Moreover, it is somewhat unclear what constitutes an effective semantics-preserving data augmentation strategy given that a point cloud is typically defined as a set of 3D coordinates. Any disturbance of the original geometry, such as cropping or view-based occlusions, could potentially degrade its semantics [15]; for example, random crops of a point cloud may correspond to different objects and introduce inconsistent learning signals. For this reason, contrastive approaches need humans to carefully design combinations of data augmentations to learn informative representations. On the other hand, training on whole object instances can lead to the learning of global representations, which in turn can produce fewer discriminant representations as local geometric differences may be disregarded [18, 36, 49]. Therefore, our first motivation is to design a data augmentation-free unsupervised learning approach to avoid the inconvenience of building chains of ad-hoc combinations of data augmentations. Second, we develop an unsupervised method that is not based on global features but instead can optimize local features, which facilitates the network to learn 3D spatial geometric information of point clouds.

In this paper, we propose an unsupervised method to learn informative point-level representations of 3D point clouds without using data augmentation. Our framework learns cluster affiliation scores to softly group the 3D points of each point cloud into a given number of geometric partitions, i.e. through soft clustering. We learn point-level feature representations by minimizing the standard cross-entropy of a single equation, which is the result of an EM-like algorithm [30]. The Expectation step employs an optimal transport [32] based clustering algorithm to generate point-level pseudo-labels, i.e. focusing on local geometric information. In particular, we softly label points based on their distance from the centroids in both feature and geometric spaces, with the constraint that labels partition data in equally-sized subsets (uniform distribution). Optimal transport serves as a potent means for comparing probability distributions with each other, as well as for producing optimal mappings to minimize distances [32]. The Maximization step adapts the E-step for a point-to-cluster loss to optimize

the metric learning network. Our approach learns the partitioning network itself and softly assigns points into geometrically coherent overlapping clusters, overcoming the weakness of conventional GMM and K-means that involve expensive iterative procedures. In doing so, we avoid data augmentation that might degrade its geometric coherency and thus its semantic information. Our approach is inspired by DeepCluster [6], SeLa [4] and SwAV [7] but it differs from them, as they implement clustering in the feature space at the instance level, they use data-augmentation, and they may degrade the geometric information when used with 3D data. We show that pre-training on datasets using SoftClu can improve the performance of a range of downstream tasks and outperforms the recent state-of-the-art methods without any data augmentation and also across different domains.
To summarize, our contributions are:

- We propose a data augmentation-free unsupervised method, which does not rely on data augmentations, negative pair sampling, and large batches, to learn transferable point-level features on a 3D point cloud;
- We extend the pseudo-label prediction to an optimal transport problem, which can be efficiently solved by using an efficient variant of the Sinkhorn-Knopp [12] algorithm;
- We conduct thorough experiments, and SoftClu achieves state-of-the-art performance without having to use data augmentation.

2 Related Work

In this section, we briefly review existing works related to unsupervised learning on point clouds, which can be classified into two categories: generative and discriminative methods.
Generative methods. Generative methods learn features via self-reconstruction [18]. For instance, FoldingNet [51] leverages a graph-based encoder and a folding-based decoder to deform a canonical 2D grid onto the surface of a point cloud. L2g [26] uses a local-to-global auto-encoder to simultaneously learn the local and global structure of point clouds. In [9], a graph-based decoder with a learnable graph topology is used to push the codeword to preserve representative features. In [1] a combination of hierarchical Bayesian and generative models is trained to generate plausible point clouds. GraphTER [16] self-trains a feature encoder by reconstructing node-wise transformations from the representations of the original and transformed graphs. However, generative models are sensitive to transformations, weakening the learning of robust point cloud representations for different downstream tasks. Moreover, it is not always feasible to reconstruct back the shape from pose-invariant features [37].
Discriminative methods. Discriminative methods are based on auxiliary handcrafted prediction tasks to learn point cloud representations. Jigsaw3D [39] uses a 3D jigsaw puzzle approach as the self-supervised learning task. Recently, contrastive approaches [11, 36, 37, 49], which are robust to transformation, achieved state-of-the-art performance. Info3D [37] maximizes the mutual information between the 3D shape and a geometric transformed version of the 3D shape. PointContrast [49] is the first to research a unified framework of the contrastive paradigm for 3D representation learning. We argue that the success of contrastive methods relies on the correct design of negative mining strategies and on the correct choice of data augmentations that should not affect the semantics of the input. To mitigate these issues, we propose an unsupervised learning method, SoftClu, which is formulated by implementing an EM-like algorithm and provides point-level supervision to extract discriminative point-wise features. SoftClu needs no data augmentation procedures to learn informative representations.

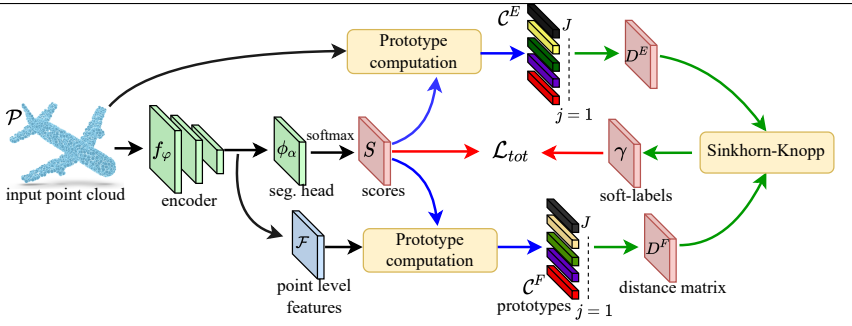

Figure 1: The architecture of our SoftClu. It consists of three steps: prototype computation (blue line), soft-label $\boldsymbol{\gamma}$ assignment (green line), and optimization (red line).

3 Proposed method

We formulate the problem of representation learning as a soft-clustering problem. Figure 1 illustrates our framework that consists of three steps: prototype computation, soft-label assignment, and optimization. Given a 3D point cloud as an unordered set $\mathcal{P} = \{\boldsymbol{p}_i\}_{i=1}^N$ of N points where each point $\boldsymbol{p}_i \in \mathbb{R}^3$ is represented by a 3D coordinate $\boldsymbol{p}_i = \{x, y, z\}$, our goal is to train, in an unsupervised way, a feature encoder f_φ with parameters φ (*e.g.*, PointNet) that extracts informative point-wise features $\mathcal{F} = \{f_\varphi(\boldsymbol{p}_i)\}_{i=1}^N$ from $\mathcal{P}$. To this end, we apply a segmentation head ϕ_α that takes as input $\mathcal{F}$ and outputs joint log probabilities, and a softmax operator that acts on log probabilities to generate a classification score matrix $\boldsymbol{S}$. The prototype computation block estimates the J cluster centroids (prototypes) $\boldsymbol{C}^E$ and $\boldsymbol{C}^F$ to represent each partitions. Next, soft-labels $\boldsymbol{\gamma}_{ij} \in \boldsymbol{\gamma}$ of each input point $\boldsymbol{p}_i$ are based on these prototypes and we use the Sinkhorn-Knopp [12] algorithm to perform the soft-label assignment, i.e., $\boldsymbol{\gamma}$ softly groups $\mathcal{P}$ into partitions. $\boldsymbol{\gamma}_{ij} \in [0,1]$ is a soft-label score that point $\boldsymbol{p}_i$ belongs to cluster j. The final optimization step is to minimize the average cross-entropy loss $\mathcal{L}_{tot}$ between the soft-label $\boldsymbol{\gamma}$ and the predicted class probability $\boldsymbol{S}$.

Prototype computation. We begin by computing a prototype for each cluster (partition) as the most representative feature for a set of points. Specifically, we use the point-wise features $\mathcal{F} = \{\boldsymbol{f}_i\}_{i=1}^N$, where $\boldsymbol{f}_i = f_\varphi(\boldsymbol{p}_i)$, to compute a set of classification scores $\boldsymbol{S}$ as $\boldsymbol{S} = \{\sigma(\phi_\alpha(\boldsymbol{f}_i)\}_{i=1}^N$. σ is the softmax operation, and ϕ_α is the segmentation layer with parameters α. For each feature $\boldsymbol{f}_i$, ϕ_α produces a probability score s_{ij} indicating the likelihood that $\boldsymbol{p}_i$ belongs to partition j. We use two types of prototypes as the representatives for each category, one in the feature space and another in the geometric space. Specifically, according to the type, we compute J prototypes as the weighted average of the features $\mathcal{F}$ or 3D coordinates $\mathcal{P}$ based on their scores $\boldsymbol{S}$. Let $\boldsymbol{C}^E = \{\boldsymbol{c}_j^E\}_{j=1}^J$ and $\boldsymbol{C}^F = \{\boldsymbol{c}_j^F\}_{j=1}^J$ be the set of prototypes in the geometric and the features space, respectively, which are defined as

$$\boldsymbol{c}_j^E = \frac{1}{\sum_{i=1}^N s_{ij}} \sum_{i=1}^N s_{ij} \boldsymbol{p}_i, \quad \boldsymbol{c}_j^F = \frac{1}{\sum_{i=1}^N s_{ij}} \sum_{i=1}^N s_{ij} \boldsymbol{f}_{p_i}. \tag{1}$$

Soft-label assignment. We introduce the soft-label assignment step that labels points based on their distance to the prototypes estimated by Eq. (1). If we only use features for soft-label assignment, this would highly likely produce disconnected and scattered clusters. Hence, we concatenate point coordinates with the features so that the label is more localized. Specifically, we encode the pseudo-labels $\boldsymbol{\gamma} = \{\gamma_{ij} \in [0,1]\}_{i,j}^{N,J}$ as posterior distributions, i.e. soft-labels, satisfying $\sum_{j=1}^J \gamma_{ij} = 1$. γ_{ij} is the posterior probability that $\boldsymbol{p}_i$ belongs to partition j. We base

the assignment of soft-labels to the respective points on the prototypes $\boldsymbol{C}^E$ and $\boldsymbol{C}^F$, and by following two assumptions:

i) Cluster cohesion: If a point $\boldsymbol{p}_i$ belongs to partition j, point $\boldsymbol{p}_i$ and prototype $\boldsymbol{c}_j^E$ should have the shortest distance among the distances of $\boldsymbol{p}_i$ with other prototypes in $\boldsymbol{C}^E$. The same holds true in the feature space.

ii) Uniform distribution: Each point cloud is assumed to be segmented into equally-sized partitions of $\lfloor \frac{N}{J} \rfloor$ elements, where $\lfloor \cdot \rfloor$ indicates the greatest integer less than or equal to its argument.

Assumption i) inspires us to label points based on their distance from the centroids. It can be formalized as an expression, i.e., if $\boldsymbol{p}_i$ belongs to cluster j, then $\|\boldsymbol{p}_i - \boldsymbol{c}_j^E\|_2 \leq \|\boldsymbol{p}_i - \boldsymbol{c}_k^E\|_2, \|\boldsymbol{f}_i - \boldsymbol{c}_j^F\|_2 \leq \|\boldsymbol{f}_i - \boldsymbol{c}_k^F\|_2$ and $\gamma_{ij} \geq \gamma_{ik}, k \neq j, k = 1, \cdots, J$. $\|\cdot\|_2$ is the L2 norm. This can be ensured by minimizing the following objective,

$$\min_{\boldsymbol{\gamma}} \frac{1}{N} \sum_{i=1}^{N} \sum_{j=1}^{J} \left(\lambda \|\boldsymbol{p}_i - \boldsymbol{c}_j^E\|_2^2 + (1-\lambda) \|\boldsymbol{f}_i - \boldsymbol{c}_j^F\|_2^2 \right) \gamma_{ij}, \tag{2}$$

where $\lambda \in [0,1]$ is a learned parameter. For convenience, we define the following matrix form $\boldsymbol{D} = \lambda \boldsymbol{D}^E + (1-\lambda)\boldsymbol{D}^F$, where $\boldsymbol{D}^F = \{\|\boldsymbol{f}_i - \boldsymbol{c}_j^F\|_2^2\}_{i,j}^{N,J}$ and $\boldsymbol{D}^E = \{\|\boldsymbol{p}_i - \boldsymbol{c}_j^E\|_2^2\}_{i,j}^{N,J}$ are matrices of size equal to $N \times J$. Then, Eq. (2) can be rewritten as:

$$\min_{\boldsymbol{\gamma}} \left\langle \frac{\boldsymbol{\gamma}}{N}, \boldsymbol{D} \right\rangle, \tag{3}$$

where $\langle \cdot, \cdot \rangle$ is the Frobenius matrix dot product.

Assumption ii) is formulated in a constraint condition as $\sum_{i=1}^{N} \gamma_{ij} = \frac{N}{J}$, which can mitigate the problem that all data points are assigned to a single (arbitrary) label. Therefore, based on $\sum_{i=1}^{N} \gamma_{ij} = \frac{N}{J}$ and the property of the posterior probability $\sum_{j=1}^{J} \gamma_{ij} = 1$, $\boldsymbol{\gamma}$ satisfies the following constraints

$$\frac{1}{N} \boldsymbol{\gamma}^\top \mathbf{1}_N = \frac{1}{J} \mathbf{1}_J, \frac{1}{N} \boldsymbol{\gamma} \mathbf{1}_J = \frac{1}{N} \mathbf{1}_N, \tag{4}$$

where $\mathbf{1}_k (k = N, J)$ denotes the vector of ones in dimension k.

Let $\boldsymbol{\Gamma} = \frac{\boldsymbol{\gamma}}{N}$ with elements defined as $\Gamma_{ij} = \frac{\gamma_{ij}}{N}$. $\boldsymbol{\Gamma}$ satisfies $\sum_{ij}^{NJ} \boldsymbol{\Gamma}_{ij} = 1$. By replacing the variable $\boldsymbol{\gamma}$ with $\boldsymbol{\Gamma}$ in Eq. (3) and Eq. (4), the joint objective of assumptions i) and ii) can be formulated as an optimal transport (OT) problem [32] as

$$\min_{\boldsymbol{\Gamma}} \langle \boldsymbol{\Gamma}, \boldsymbol{D} \rangle, \text{ s.t. } \boldsymbol{\Gamma}^\top \mathbf{1}_N = \frac{1}{J} \mathbf{1}_J, \boldsymbol{\Gamma} \mathbf{1}_J = \frac{1}{N} \mathbf{1}_N. \tag{5}$$

The minimization of Eq. (5) can be solved in polynomial time as a linear program. However, the linear program involves millions of data points and thousands of classes and traditional algorithms hardly scale to large problems [12]. We address this issue by adopting an efficient version of the Sinkhorn-Knopp algorithm [12]. Our implementation of the Sinkhorn-Knopp algorithm is described in Alg. 2 (Appendix). This requires the following regularization term

$$\min_{\boldsymbol{\Gamma}} \langle \boldsymbol{\Gamma}, \boldsymbol{D} \rangle - \varepsilon H(\boldsymbol{\Gamma}), \quad \text{s.t. } \boldsymbol{\Gamma}^\top \mathbf{1}_N = \frac{1}{J} \mathbf{1}_J, \ \boldsymbol{\Gamma} \mathbf{1}_J = \frac{1}{N} \mathbf{1}_N, \tag{6}$$

where $H(\boldsymbol{\Gamma}) = \langle \boldsymbol{\Gamma}, \log \boldsymbol{\Gamma} - 1 \rangle$ denotes the entropy of $\boldsymbol{\Gamma}$ and $\varepsilon > 0$ is a regularization parameter. For very small ε, optimizing Eq. (6) is equivalent to optimizing Eq. (5), but even for moderate values of ε, the objective tends to have approximately the same optimizer [12]. The larger the ε, the faster the convergence, please refer to [12] for details. In our case, using a fixed $\varepsilon = 1e-3$ is appropriate as we are interested in the final clustering and representation learning results, rather than in solving the transport problem exactly. The solution to Eq. (6) takes the form of the following normalized exponential matrix [12],

$$\boldsymbol{\Gamma} = \operatorname{diag}(\boldsymbol{\mu}) \exp\left(\boldsymbol{D}/\varepsilon\right) \operatorname{diag}(\boldsymbol{\nu}), \tag{7}$$

where $\boldsymbol{\mu} = (\mu_1, \mu_2, \cdots, \mu_N)$ and $\boldsymbol{\nu} = (\nu_1, \nu_2, \cdots, \nu_J)$ are renormalization vectors in $\mathbb{R}^N$ and $\mathbb{R}^J$. The vectors $\boldsymbol{\mu}$ and $\boldsymbol{\nu}$ can be obtained by iterating the updates via $\boldsymbol{\mu}_i = \left[\exp\left(\boldsymbol{D}/\varepsilon\right)\boldsymbol{\nu}\right]_i^{-1}$ and $\boldsymbol{\nu}_j = \left[\exp\left(\boldsymbol{D}/\varepsilon\right)^\top \boldsymbol{\mu}\right]_j^{-1}$ with initial values $\boldsymbol{\mu} = \frac{1}{N}\mathbf{1}_N$ and $\boldsymbol{\nu} = \frac{1}{J}\mathbf{1}_J$, respectively. The initialization of $\boldsymbol{\mu}$ and $\boldsymbol{\nu}$ can be any distribution, and choosing the constraints as initial values allow a faster convergence [12]. $[\cdot]_j^{-1}$ defines as the inverse value of the j^{th} element of its argument. In all our experiments, we use 20 iterations as we found it works well in practice. After solving Eq. (7), we can infer the soft-label matrix as $\boldsymbol{\gamma} = N \cdot \boldsymbol{\Gamma}$.

Optimization. The optimization step follows an EM-like scheme where the Expectation step E optimizes prototypes and soft labels, while the Maximization step M optimizes the trained parameters for representation learning. Each step can be detailed as follows:

- E: Given the current encoder and segmentation layer, we compute prototypes $\boldsymbol{C}^E$ and $\boldsymbol{C}^F$ following Eq. (1), and obtain soft-labels $\boldsymbol{\gamma}$ through $\boldsymbol{\gamma} = N \cdot \boldsymbol{\Gamma}$.
- M: Given the current soft-labels $\boldsymbol{\gamma}$ from step E, we optimize the encoder f_φ and segmentation layer ϕ_α parameters.

During E, we solve the OT problem with the Sinkhorn-Knopp algorithm. During M, we minimize the segmentation loss based on the resulting soft labels, such as

$$\mathcal{L}_{soft}(\boldsymbol{\gamma}, \boldsymbol{S}) = -\frac{1}{N}\langle \boldsymbol{\gamma}, \log \boldsymbol{S} \rangle = -\frac{1}{N}\sum_{i=1}^{N}\sum_{j=1}^{J} \boldsymbol{\gamma}_{ij} \log s_{ij}, \tag{8}$$

which corresponds to the minimization of the standard cross-entropy loss between soft-labels $\boldsymbol{\gamma}$ and predictions $\boldsymbol{S}$. However, the optimization of Eq. (8) does not ensure encoder f_φ from predicting the same features for all the points, i.e. all centroids collapse into the same vector. We further promote centroids separation by minimizing the orthogonal regularization loss $\mathcal{L}_{orth}(\boldsymbol{C}) = \|{\boldsymbol{C}^E_*}^\top \boldsymbol{C}^E_* - \boldsymbol{I}\|_{Fr} + \|{\boldsymbol{C}^F_*}^\top \boldsymbol{C}^F_* - \boldsymbol{I}\|_{Fr}$, where $\boldsymbol{C}^k_* = [\frac{\boldsymbol{c}^k_1}{\|\boldsymbol{c}^k_1\|_2}, \frac{\boldsymbol{c}^k_2}{\|\boldsymbol{c}^k_2\|_2}, \cdots, \frac{\boldsymbol{c}^k_J}{\|\boldsymbol{c}^k_J\|_2}]$ with $k = E, F$, and $\|\cdot\|_{Fr}$ is Frobenius norm. We define the final objective loss for the M step as

$$\mathcal{L}_{tot} = \mathcal{L}_{soft} + \eta\,\mathcal{L}_{orth}, \tag{9}$$

where $\eta = 0.01$ is a weighting parameter. We set the value of η empirically and found that $\eta \leq 0.01$ can slightly improve the performance. The minimization of this loss leads to the maximization of the expected number of points correctly classified, associating the correct neighbor prototypes. This facilitates the encoder to learn more local geometric information. Our implementation of SoftClu is described in Alg. 1 in the supplementary material.

4 Experiments

In this section, we present the implementation details, the setup of pre-training and the downstream fine-tuning. We invite the reader to check in the supplementary material for additional results on few-shot learning, ablation studies, Transformer, and additional visualizations.

4.1 Pre-training setup

We explore pre-training strategies on single objects (ShapeNet [8]) and complex scenes with multiple objects (ScanNet [13]) to evaluate the effectiveness of SoftClu. We implemented SoftClu in PyTorch and executed our experiments on two Tesla V100-PCI-E-32G GPUs. During pre-training, we set $J = 64$ and $\varepsilon = 1e-3$.

ShapeNet [8] is a collection of single-object CAD models and contains 57448 synthetic objects from 55 object categories. We follow the experimental setup presented in [21, 39], we use PointNet [35] and DGCNN [48] as encoder networks. The latent dimension of both

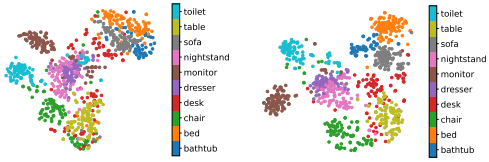

Figure 2: T-SNE-processed network features obtained with (left) OcCo [46] and (right) SoftClu on ModelNet10.

Figure 3: Color-coded points based on PCA projections of the learned features: (left) ModelNet40, (right) ShapeNet.

encoders is 1024. Following [21], each point cloud is randomly downsampled to 2048 points. We use the official training split of ShapeNet for pre-training. Our pre-training involves 250 epochs by using the AdamW [27] optimizer, the batch size is equal to 32, and initial learning is equal to 0.001 that decays by 0.7 every 20 epochs.

ScanNet [13] is a dataset of indoor scenes with multiple objects and consists of 1513 reconstructed meshes for 707 unique scenes. Following [49], we choose SR-UNet provided in PointContrast [49] as backbone. For pre-training, we use an SGD optimizer with a learning rate of 0.1 and a batch size of 32. The learning rate is decreased by a factor of 0.99 every 1K iteration. The model is trained for 30K iterations.

4.2 Downstream fine-tuning

We evaluate SoftClu on three downstream tasks: classification, part segmentation, and semantic segmentation. We compare SoftClu to state-of-the-art discriminative approaches (Jigsaw3D [39], STRL [21], CrossPoint [2], SimCLR [10], STRL [21], PointContrast [49], and ContrastiveScene [20]) and a generative approach (OcCo [46] and ParAE [15]). The details of setups for downstream tasks can be found in supplementary for three encoders.

Object classification. We use linear SVM classification on ModelNet40 [40] and ModelNet10 [40] datasets to evaluate the quality of their pre-trained versions on ShapeNet. Table 1 reports the classification accuracy of SoftClu, compared to the other approaches. Results show that the SoftClu is more effective than the alternative pre-training methods on both datasets. Specifically, on ModelNet40, SoftClu with PointNet backbone achieves the same classification accuracy (90.3%) as ParAE [15] while outperforming the contrastive approach STRL [21] (88.3%). The linear SVM classification performance of our method even surpasses the performance of the fully supervised PointNet, which achieves an 89.2% test accuracy. With the DGCNN encoder, our method achieves a 91.9% test accuracy, outperforming ParAE (91.6%) by 0.3%, and generating model OcCo [46] by 2.7%. On ModelNet10, SoftClu outperforms OcCo [46] and Jigsaw3D [39] with both the encoding networks. Compared to jigsaw tasks that coarsely segment a point cloud into disjoint partitions, SoftClu learns the partitioning function itself to softly assign point clouds into coherent clusters. Moreover, we also report results (SoftClu*) of SoftClu pre-trained on ModelNet40, since some methods are pretrained on ModelNet40. SoftClu also achieves competitive results. We further use visualization to explore pre-trained features before fine-tuning with the DGCNN encoder. Figure 2 shows the features visualized with T-SNE [45] of OcCo and SoftClu on ModelNet10. Our method yields a better separation of the features than OcCo, which indicates a better ability of SoftClu in clustering objects in the feature space. In Figure 3, we color the points based on the PCA projections of the network features and shows how the pre-trained encoder can effectively embed geometric information.

Part segmentation. We follow [46] and use the ShapeNetPart [52] benchmark dataset. Table 2 reports the part segmentation results of SoftClu in comparison with the alternative

Table 1: Linear classification comparisons on ModelNet40 and ModelNet10. ⋆ indicates that models are pre-trained on ModelNet40, otherwise, models are pre-trained on ShapeNet.

Method	Year	PointNet		DGCNN	
		ModelNet40	ModelNet10	ModelNet40	ModelNet10
DeepCluster [3]	2018	86.3	91.6	90.4	94.1
Jigsaw3D⋆ [39]	2019	87.5	91.3	87.8	92.6
Jigsaw3D [39]	2019	87.3	91.6	90.6	94.5
Rotation3D [34]	2020	88.6	-	90.8	-
SwAV [3]	2020	85.4	92.1	90.3	93.5
OcCo [46]	2021	88.7	91.4	89.2	92.7
SimCLR [10]	2021	88.4	91.4	90.1	92.1
STRL [21]	2021	88.3	-	90.9	-
ParAE [15]	2021	**90.3**	-	91.6	-
CrossPoint [3]	2022	89.1	-	91.2	-
SoftClu⋆ (Ours)	-	88.4	93.0	91.4	94.5
SoftClu (Ours)	-	**90.3**	**93.5**	**91.9**	**94.8**

Table 2: Part segmentation results.

Encoder	Method	Metrics	
		OA (%)	mIoU (%)
PointNet	Random	92.8	82.2
	Jigsaw3D [39]	93.1	82.2
	OcCo [46]	93.4	83.4
	CrossPoint [3]	93.2	82.7
	SoftClu (Ours)	**93.9**	**83.8**
DGCNN	Random	92.2	84.4
	Jigsaw3D [39]	92.7	84.3
	OcCo [46]	94.4	85.0
	CrossPoint [3]	94.4	85.3
	SoftClu (Ours)	**94.6**	**85.7**

Table 3: Semantic segmentation results.

Encoder	Method	Metrics	
		OA (%)	mIoU (%)
PointNet	Random	78.9	47.0
	Jigsaw3D [39]	80.1	52.6
	OcCo [46]	82.0	54.9
	CrossPoint [3]	81.8	54.5
	SoftClu (Ours)	**82.9**	**55.3**
DGCNN	Random	83.7	54.9
	Jigsaw3D [39]	84.1	55.6
	OcCo [46]	84.6	58.0
	CrossPoint [3]	84.7	58.4
	SoftClu (Ours)	**85.4**	**59.2**

approaches on ShapeNetPart [52]. SoftClu outperforms all the other approaches with both PointNet and DGCNN encoders in terms of both OA and mIoU. With the PointNet encoder, SoftClu achieves 93.9% OA and 83.8% mIoU, improving over state-of-the-art CrossPoint (93.2% OA, 82.7% mIoU) by 0.7% OA and 1.1% mIoU. With the DGCNN encoder, we achieve 94.6% OA and 85.7% mIoU, outperforming CrossPoint (94.4 OA, 85.3% mIoU) of about 0.2% OA and 0.4% mIoU. Figure 4 in the supplementary material shows examples of qualitative part segmentation results.

Semantic segmentation. We first evaluate SoftClu features on semantic segmentation by using the S3DIS [3] benchmark dataset, where features are pre-trained on ShapeNet. Table 3 reports the segmentation results of SoftClu and that of the other baselines on S3DIS [3]. SoftClu outperforms all the other approaches with both PointNet and DGCNN encoders. With the PointNet encoder, SoftClu achieves 82.9% OA and 55.3% mIoU, outperforming both the state-of-the-art OcCo (82.0% OA, 55.3% mIoU) and Jigsaw3D (80.1% OA, 52.6% mIoU). With the DGCNN encoder, SoftClu achieves 85.4% OA and 59.2% mIoU, outperforming CrossPoint [2] (84.7% OA and 58.4% mIoU), OcCo (84.6% OA, 58.0% mIoU) and Jigsaw3D (84.1% OA, 55.6% mIoU). We also compare SoftClu with point-level methods such as PointContrast [49] and ContrastiveScene [20] when features are pre-trained on ScanNet with SR-UNet backbone. Table 4 shows that our pre-training method achieves 73.4% mIoU and 79.1% mAcc, outperforming the pre-training results of PointContrast [49] and ContrastiveScene [20].

Table 4: Results of semantic segmentation with SR-UNet backbone [6].

Method	Scratch	PointContrast [49]	ContrastiveScene [20]	SoftClu
mIoU	68.2	70.3	72.2	**73.4**
mAcc	75.5	76.9	-	**79.1**

4.3 Ablation study and analysis

Feature and geometric prototypes. We study the contribution of feature and geometric prototypes of SoftClu (Eq. 1). We perform pre-training by using (i) only feature prototypes, (ii) only geometric prototypes, and (iii) both prototypes. We perform this study on ModelNet40 and ModelNet10. In Table 5 we can observe that when only feature prototypes are used, SoftClu achieves the lowest performance. This is due to wrong cluster assignments of those points sharing similar features but belonging to different geometric regions (e.g. wings of an airplane). Using both feature and geometric prototypes, SoftClu can achieve the best performance on ModelNet40 and ModelNet10, with both PointNet and DGCNN.

Table 5: Ablation study of SoftClu by using different prototypes.

Encoder	Geometry	Feature	Accuracy	
			ModelNet40	ModelNet10
PointNet	✓		88.7	92.9
		✓	86.5	92.7
	✓	✓	**90.3**	**93.5**
DGCNN	✓		91.4	94.5
		✓	90.5	93.3
	✓	✓	**91.9**	**94.8**

Number of clusters. We assess the effect of selecting different numbers of cluster partitions J by using ModelNet40. We pre-train SoftClu with different values of J, i.e. from 16 to 128, and report the results in Table 6. SoftClu achieves the best results with $J = 64$ for both PointNet and DGCNN. We observed stability with the results throughout different values of J. Additional results of the batch size and soft-labels please refer to the supplementary material.

Table 6: Ablation study results of SoftClu with different number of clusters J.

Method	16	32	48	64	72	96	112	128
PointNet	92.4	93.0	93.1	93.5	93.4	93.3	93.2	93.1
DGCNN	94.2	94.8	94.6	94.8	94.7	94.6	94.6	94.5

Running times. Our method is used only for pre-training, where each iteration consists of two parts: a backbone forward pass and SoftClu optimization. We run our method on one Tesla V100 GPU (32G) and two Intel(R) 6226 CPUs and measured the iteration time over several iterations. SoftClu adds an average overhead of 0.014ms for each iteration on ShapeNet with a DGCNN backbone. Notice that, the inference time of each backbone used remains the original one.

5 Conclusions

We presented SoftClu, a data augmentation-free unsupervised representation learning scheme for 3D point cloud understanding. SoftClu implicitly alternates between clustering the point-level features to produce point-wise pseudo-labels and utilizing these soft-labels to train the representations. Our approach showed promising results in transferring the pre-trained representations to different downstream 3D understanding tasks, such as classification, part segmentation, and semantic segmentation. Our SoftClu is independent from specific deep network architectures, enabling us to use our method as a generic method for feature extraction from raw point cloud data to improve other 3D models performances.

Acknowledgments. This work was supported by the EU H2020 AI4Media No. 951911 project and the EUREGIO project OLIVER.

References

[1] Panos Achlioptas, Olga Diamanti, Ioannis Mitliagkas, and Leonidas Guibas. Learning representations and generative models for 3d point clouds. In ICML, 2018.

[2] Mohamed Afham, Isuru Dissanayake, Dinithi Dissanayake, Amaya Dharmasiri, Kanchana Thilakarathna, and Ranga Rodrigo. Crosspoint: Self-supervised cross-modal contrastive learning for 3d point cloud understanding. In CVPR, June 2022.

[3] Iro Armeni, Ozan Sener, Amir R Zamir, Helen Jiang, Ioannis Brilakis, Martin Fischer, and Silvio Savarese. 3d semantic parsing of large-scale indoor spaces. In CVPR, 2016.

[4] Yuki M. Asano, Christian Rupprecht, and Andrea Vedaldi. Self-labelling via simultaneous clustering and representation learning. In ICLR, 2020.

[5] Joydeep Biswas and Manuela Veloso. Depth camera based indoor mobile robot localization and navigation. In ICRA, 2012.

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

[7] Mathilde Caron, Ishan Misra, Julien Mairal, Priya Goyal, Piotr Bojanowski, and Armand Joulin. Unsupervised learning of visual features by contrasting cluster assignments. NeurIPS, 33:9912–9924, 2020.

[8] Angel X. Chang, Thomas Funkhouser, Leonidas Guibas, Pat Hanrahan, Qixing Huang, Zimo Li, Silvio Savarese, Manolis Savva, Shuran Song, Hao Su, Jianxiong Xiao, Li Yi, and Fisher Yu. Shapenet: An information-rich 3d model repository. In arXiv, 2015.

[9] Siheng Chen, Chaojing Duan, Yaoqing Yang, Duanshun Li, Chen Feng, and Dong Tian. Deep unsupervised learning of 3d point clouds via graph topology inference and filtering. TIP, 29:3183–3198, 2019.

[10] Ting Chen, Simon Kornblith, et al. A simple framework for contrastive learning of visual representations. ICML, 2020.

[11] Xinlei Chen and Kaiming He. Exploring simple siamese representation learning. In CVPR, 2021.

[12] Marco Cuturi. Sinkhorn distances: Lightspeed computation of optimal transport. NeurIPS, 2013.

[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, pages 5828–5839, 2017.

[14] Bi'an Du, Xiang Gao, Wei Hu, and Xin Li. Self-contrastive learning with hard negative sampling for self-supervised point cloud learning. In ACM MM, pages 3133–3142, 2021.

[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] Xiang Gao, Wei Hu, and Guo-Jun Qi. Graphter: Unsupervised learning of graph transformation equivariant representations via auto-encoding node-wise transformations. In CVPR, 2020.

[17] Jean-Bastien Grill, Florian Strub, Florent Altché, Corentin Tallec, Pierre H Richemond, Elena Buchatskaya, Carl Doersch, Bernardo Avila Pires, Zhaohan Daniel Guo, Mohammad Gheshlaghi Azar, et al. Bootstrap your own latent: A new approach to self-supervised learning. In NeurIPS, 2020.

[18] Zhizhong Han, Xiyang Wang, et al. Multi-angle point cloud-vae: unsupervised feature learning for 3d point clouds from multiple angles by joint self-reconstruction and half-to-half prediction. In ICCV, pages 10441–10450, 2019.

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

[20] Ji Hou, Benjamin Graham, Matthias Nießner, and Saining Xie. Exploring data-efficient 3d scene understanding with contrastive scene contexts. In CVPR, pages 15587–15597, 2021.

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

[22] Xiaoshui Huang, Guofeng Mei, and Jian Zhang. Feature-metric registration: A fast semi-supervised approach for robust point cloud registration without correspondences. In CVPR, pages 11366–11374, 2020.

[23] Ying Li, Lingfei Ma, Zilong Zhong, Fei Liu, Michael A Chapman, Dongpu Cao, and Jonathan Li. Deep learning for lidar point clouds in autonomous driving: a review. TNNLS, 2020.

[24] Xuanxiang Lin, Ke Chen, and Kui Jia. Object point cloud classification via poly-convolutional architecture search. In ACM MM, pages 807–815, 2021.

[25] Haotian Liu, Mu Cai, and Yong Jae Lee. Masked discrimination for self-supervised learning on point clouds. arXiv preprint arXiv:2203.11183, 2022.

[26] Xinhai Liu, Zhizhong Han, et al. L2g auto-encoder: Understanding point clouds by local-to-global reconstruction with hierarchical self-attention. In ACM MM, pages 989–997, 2019.

[27] Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. In ICLR, 2018.

[28] Guofeng Mei. Point cloud registration with self-supervised feature learning and beam search. In DICTA, pages 01–08, 2021.

[29] Guofeng Mei, Xiaoshui Huang, Jian Zhang, and Qiang Wu. Overlap-guided coarse-to-fine correspondence prediction for point cloud registration. In ICME, pages 1–6. IEEE, 2022.

[30] Todd K Moon. The expectation-maximization algorithm. IEEE Signal processing magazine, 1996.

[31] Youngmin Park, Vincent Lepetit, and Woontack Woo. Multiple 3d object tracking for augmented reality. In ISMAR, 2008.

[32] Gabriel Peyré, Marco Cuturi, et al. Computational optimal transport: With applications to data science. Foundations and Trends® in Machine Learning, 2019.

[33] F. Poiesi and D. Boscaini. Learning general and distinctive 3D local deep descriptors for point cloud registration. TPAMI, 2022.

[34] Omid Poursaeed, Tianxing Jiang, Han Qiao, Nayun Xu, and Vladimir G Kim. Self-supervised learning of point clouds via orientation estimation. In 3DV, 2020.

[35] Charles R Qi, Hao Su, et al. Pointnet: Deep learning on point sets for 3d classification and segmentation. In CVPR, pages 652–660, 2017.

[36] Yongming Rao, Jiwen Lu, and Jie Zhou. Global-local bidirectional reasoning for unsupervised representation learning of 3d point clouds. In CVPR, 2020.

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

[38] Muhammad Sarmad, Hyunjoo Jenny Lee, et al. Rl-gan-net: A reinforcement learning agent controlled gan network for real-time point cloud shape completion. In CVPR, pages 5898–5907, 2019.

[39] Jonathan Sauder and Bjarne Sievers. Self-supervised deep learning on point clouds by reconstructing space. In NeurIPS, pages 12942–12952, 2019.

[40] Abhishek Sharma, Oliver Grau, and Mario Fritz. Vconv-dae: Deep volumetric shape learning without object labels. In ECCV, pages 236–250, 2016.

[41] Charu Sharma and Manohar Kaul. Self-supervised few-shot learning on point clouds. NeurIPS, 33:7212–7221, 2020.

[42] Shaoshuai Shi, Chaoxu Guo, Li Jiang, Zhe Wang, Jianping Shi, Xiaogang Wang, and Hongsheng Li. Pv-rcnn: Point-voxel feature set abstraction for 3d object detection. In CVPR, 2020.

[43] Yi Shi, Mengchen Xu, Shuaihang Yuan, and Yi Fang. Unsupervised deep shape descriptor with point distribution learning. In CVPR, pages 9353–9362, 2020.

[44] Yongbin Sun, Yue Wang, Ziwei Liu, Joshua Siegel, and Sanjay Sarma. Pointgrow: Autoregressively learned point cloud generation with self-attention. In WACV, pages 61–70, 2020.

[45] Laurens Van der Maaten and Geoffrey Hinton. Visualizing data using t-sne. JMLR, 2008.

[46] Hanchen Wang, Qi Liu, Xiangyu Yue, Joan Lasenby, and Matthew J Kusner. Unsupervised point cloud pre-training via view-point occlusion, completion. In ICCV, 2020.

[47] Y. Wang and A. Del Bue. Where to Explore Next? ExHistCNN for History-aware Autonomous 3D Exploration. In ECCV, 2020.

[48] 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 TOG, 38(5):1–12, 2019.

[49] Saining Xie, Jiatao Gu, Demi Guo, Charles R Qi, Leonidas Guibas, and Or Litany. Pointcontrast: Unsupervised pre-training for 3d point cloud understanding. In ECCV, 2020.

[50] Mingye Xu, Zhipeng Zhou, and Yu Qiao. Geometry sharing network for 3d point cloud classification and segmentation. In AAAI, pages 12500–12507, 2020.

[51] Yaoqing Yang, Chen Feng, Yiru Shen, and Dong Tian. Foldingnet: Point cloud auto-encoder via deep grid deformation. In CVPR, pages 206–215, 2018.

[52] 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 TOG, 2016.

[53] Tianwei Yin, Xingyi Zhou, and Philipp Krahenbuhl. Center-based 3d object detection and tracking. In CVPR, 2021.

[54] Xumin Yu, Lulu Tang, Yongming Rao, Tiejun Huang, Jie Zhou, and Jiwen Lu. Point-bert: Pre-training 3d point cloud transformers with masked point modeling. In CVPR, pages 19313–19322, 2022.

[55] Wentao Yuan, Tejas Khot, David Held, Christoph Mertz, and Martial Hebert. Pcn: Point completion network. In 3DV, 2018.

[56] Y. Zhou, Y. Wang, F. Poiesi, Q. Qin, and Y. Wan. Loop closure detection using local 3D deep descriptors. IEEE RAL, 2022.