

Weakly Supervised 3D Point Cloud Segmentation via Multi-Prototype Learning

Yongyi Su[#], Xun Xu[#], *Senior Member, IEEE*, and Kui Jia^{*}

Abstract—Addressing the annotation challenge in 3D Point Cloud segmentation has inspired research into weakly supervised learning. Existing approaches mainly focus on exploiting manifold and pseudo-labeling to make use of large unlabeled data points. A fundamental challenge here lies in the large intra-class variations of local geometric structure, resulting in subclasses within a semantic class. In this work, we leverage this intuition and opt for maintaining an individual classifier for each subclass. Technically, we design a multi-prototype classifier, each prototype serves as the classifier weights for one subclass. To enable effective updating of multi-prototype classifier weights, we propose two constraints respectively for updating the prototypes w.r.t. all point features and for encouraging the learning of diverse prototypes. Experiments on weakly supervised 3D point cloud segmentation tasks validate the efficacy of proposed method in particular at low-label regime. Our hypothesis is also verified given the consistent discovery of semantic subclasses at no cost of additional annotations.

Index Terms—3D Point Cloud, Weakly Supervised Learning, Multi-Prototype Learning

I. INTRODUCTION

ANNOTATING 3D point cloud for segmentation is expensive as it requires extensively annotating large number of points in 3D space and the 3D characteristics, e.g. occlusion, etc., render annotation particularly harder. A recent approach towards tackling the annotation challenge is through learning from partially labelled data, a.k.a. weakly supervised learning [1]. This initial attempt provided insight into the mechanism of weakly supervised learning for 3D semantic segmentation using the central limit theorem. They further proposed to strengthen the task by learning geometric manifold and consistency based semi-supervised learning. Based on these insights, follow-up works are carried out by introducing propagation methods to produce better pseudo-labels as supervision [2], [3]. Despite the efforts on label propagation for more efficient use of limited labels, one inherent challenge in 3D point cloud segmentation, the large intra-class variation, remains unnoticed. For example, as illustrated in Fig. 1, there is a substantial diversity of visual appearance for “plane body”, “lampshade” and “table surface” even if they respectively represents a single semantic category. This intra-class variation results in subclasses clearly identified within each semantic category. When a linear classifier with cross-entropy loss is applied, data points with the same label are forced to group together and center around a *prototype* which is the

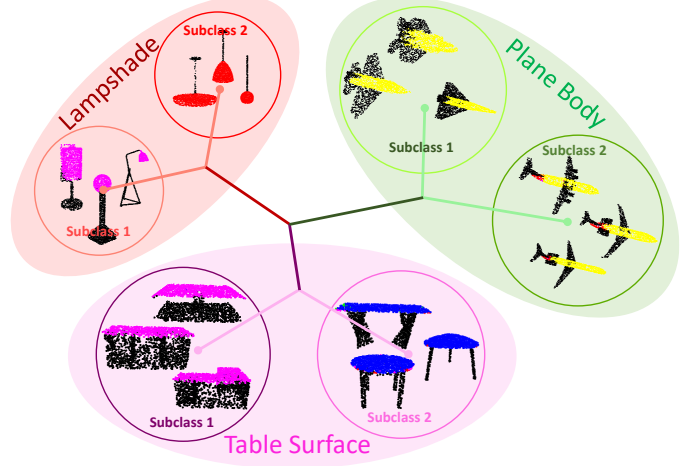


Fig. 1: Illustration of the subclass concept. Colored points indicate the activated prototype/classifier, a clear subclass structure is observed from shape part categories, e.g. “lampshade” of pendants and lamps standing on the ground, “plane body” of fighters and passenger jets and “table surface” of square and round tables.

weight of classifier [4]. As a consequence, this requires a very complicated representation function to map data points with varying appearance to a single point in the feature space. However, when labeled data is extremely low, e.g. only 1 point per category, training a single linear classifier (prototype) for each category is prone to underfitting [5].

Instead of having a single classifier/prototype for each semantic class, keeping multiple prototypes [5] has been adopted for few-shot learning to address multi-modal distribution within a semantic class. In this related problem, IMP [5] dynamically increases the number of prototypes following a Chinese restaurant process (CRP), when data points are far enough (above a threshold) to existing prototypes a new cluster center (prototype) is created. Despite the flexibility in determining the number of prototypes, CRP is essentially a sequential process and is only effective with a small support set in few-shot learning. Generalizing CRP to weakly supervised learning, where a large number of data (up to millions of points in a single mini-batch) determines the prototypes, is subject to prohibitive computation cost. In light of this challenge, we propose to introduce a multi-prototype classifier for weakly supervised segmentation, termed MulPro for simplicity. In specific, we first design a multi-prototype memory bank to store the prototypes for each semantic class and each prototype would represent one subclass. In contrast to the offline

Yongyi Su[#] and Xun Xu[#] contributed equally to this work. Corresponding author: Kui Jia^{*}. e-mail: kuijia@scut.edu.cn.

Xun Xu is with Institute for Infocomm Research, A*STAR. Yonyi Su and Kui Jia are with South China University of Technology.

prototype updating with K-means clustering [6] or moving average update adopted by [2], our design does not introduce any non-differentiable operations between prototypes and loss functions, thus enables end-to-end training.

With the introduction of multi-prototype memory bank, a new challenge arises as how to effectively train the prototypes. One naive way would be direct backpropagating from cross-entropy loss on labeled data. However, this will only provide very sparse supervision signal when label data is few. To tackle this issue, we propose to use both labeled and unlabeled data. Given the assignment of points to a prototype we enforce the prototype to be close to the feature of all assigned points, resembling taking the average, thus we name also this constraint as subclass averaging constraint. Subclass averaging is differentiable w.r.t. prototypes and can be used for gradient-base updating of prototypes. This whole process can be interpreted as nesting K-means clustering within the classifier, the forward pass computes the assignment and cluster centers are updated by subclass averaging in the backward update. Both steps can be efficiently conducted in a single forward-backward iteration.

We further notice that the multi-prototype update through subclass averaging does not prevent degenerate prototypes, e.g. some or all prototypes in one class become identical. Such a solution will cause random activation of prototypes, which is harmful for gradient-based updating. Therefore, we further propose to force the prototypes to be diverse within a category. This is achieved by penalizing the similarity between prototypes and introducing a constraint balancing prototype diversity and separability.

Finally, we carry out experiments on weakly supervised segmentation for 3D point cloud datasets and demonstrate the effectiveness of multi-prototype classifier. We reveal that it is particularly useful when intra-class variation is large and labeled data is few. We are also surprised to see subclasses being discovered simultaneously with weakly supervised training without requiring extra fine-grained annotation. The contributions of this work can be summarized as below.

- Observing the large intra-class variation and clear subclass structures in 3D point cloud segmentation data, we propose a multi-prototype classifier, termed as MulPro, to reduce the difficulty in representation learning when only sparse labeled data is available.
- We propose a subclass averaging constraint to exploit both labeled and unlabeled data to supervise prototypes learning. This can be interpreted as nesting K-means clustering within the classifier. We further propose additional constraints to encourage diversity between prototypes to avoid degenerate solutions.
- We improve weakly supervised 3D point cloud segmentation tasks and simultaneously discover subclasses within each semantic categories without any additional supervision.

II. RELATED WORK

A. 3D Point Cloud Segmentation

Segmentation is a fundamental task in understanding 3D environment. The recent surge of deep learning methods for

3D point cloud is attributed to PointNet [7] which adapted a MLP to learn keypoints for point cloud understanding. The follow-up work, PointNet++ [8], proposed to embed PointNet in a small neighborhood to capture local geometric information. Convolution based methods [9]–[12] and Graph convolution based methods [13]–[16] are subsequently proposed to further expand the receptive field and learn better local geometry. More recently, transformers [17], [18] are adapted to 3D point cloud, achieving unprecedented performance. A more detailed review of point cloud understanding can be found in [19]. Nevertheless, the success of 3D point cloud segmentation is mainly attributed to training on large amount of labeled data. While labeling on 3D data for segmentation is particularly expensive and it has inspired works addressing weakly-supervised learning for 3D point cloud segmentation.

B. 3D Point Cloud Weakly Supervised Learning.

Annotating 3D data for segmentation is expensive due to the high degree of freedom and articulated boundaries. Weakly supervised learning addresses this issue by exploiting sparsely labeled data [1]–[3], [20], [21]. [21] proposed a inexact annotation scheme by providing multiple binary labels to one region, class activation map (CAM) [22] is employed to exploit these labels to infer point-wise predictions. In another line of research, [1] assumes only a fraction of points are uniformly selected and labeled. They proved that under i.i.d. assumption, the weakly supervised learning gradient will approximate the fully supervised one. Inspired by the discovery made in [1], a pseudo-labeling approach [2] further improved label propagation to better exploit the unlabeled points. In [2], a per-region annotation assumption is adopted which exploits the unique information provided by ScanNet dataset. The strong assumption makes [2] restrictive to particular datasets where perfect super-point region is provided. More recently, motivated by saving annotation cost, an active learning approach towards point cloud data [23] is investigated to select most informative points or super-points to annotate. Alternative to mining the geometric properties, PointContrast [24] proposed an unsupervised contrastive pretraining approach to adjust model parameters on large unlabeled data. Finetuning on small label data demonstrates promising results on label-efficient learning. As opposed to propagating labels and pretraining, in this work, we are motivated by the large intra-class variation and concluded that having multiple prototypes/classifiers for each semantic class could alleviate the difficulty in representation learning.

C. Multi-Prototype Learning.

Originated in few-shot learning, multi-prototype learning aims to address the challenging of fitting prototypical network [25] for multi-modal data distribution [5], [26], [27] by learning prototypes for recognizing classes with few training examples. The first attempt, IMP [5], proposed to adaptively expand prototype pool following a Chinese restaurant process which sequentially processes data points. [27] proposes a k-means extension of Prototypical Networks. Despite the success in few-shot learning, it is impractical to trivially apply

the sequential IMP to weakly supervised learning due to computation cost while other offline methods prevents end-to-end training of prototypes. In this work, we develop a multi-prototype memory bank to capture the subclass structure and the proposed constraints allow effective multi-prototype training. In the context of 3D point cloud deep learning, exploiting multi-prototype was briefly mentioned in [26] which generates multiple prototypes by farthest point sampling on the embedding space for support set. Compared with [26], we provide the first in-depth analysis into multi-prototype in weakly supervised learning. The key novelty is how to effectively learn non-trivial multiple prototypes and demonstrating its existence. Without the introduced class averaging constraint and diversity constraints, as shown in the ablation, multi-prototype would not be effective. In contrast, multi-prototype is implemented as clustering support points and no concrete evidence of the existence of multiple prototypes is provided in [26].

III. METHODOLOGY

We introduce a multi-prototype classifier, MulPro, to exploit sparse annotations. In this section, we first formally define the weakly supervised segmentation task. Then, we describe how the multi-prototype classifier works in the weakly supervised model. Finally, in view of the difficulties in learning multiple prototypes we propose two constraints to further constrain the prototype representation learning.

A. Architecture Overview

To formally define the weakly-supervised 3D point segmentation task, we follow the settings proposed in [1]. In specific, a training dataset $\mathcal{D}_{lr} = \{\mathbf{X}_i, \mathbf{Y}_i, \mathbf{M}_i\}_{i=1 \dots N_{lr}}$ is provided, where $\mathbf{X}_i \in \mathbb{R}^{D_i \times N}$ is the input N point features, e.g. 3D coordinates with RGB color if available, $\mathbf{Y}_i \in \{0, 1\}^{K \times N}$ is the one-hot per-point segmentation label (K categories) and $\mathbf{M}_i \in \{0, 1\}^N$ is a binary mask indicating whether ground-truth label is available. An encoder network $\mathbf{Z} = f(\mathbf{X}; \Theta)$ maps input points into a feature space $\mathbf{Z} \in \mathbb{R}^{D_o \times N}$. A classifier $g(\mathbf{Z}; \Omega) \in \mathbb{R}^{K \times N}$ maps encoded features into logits in segmentation category space. The existing approaches [1], [2] often define cross-entropy loss on classifier outputs and additional regularization may derive from manifold [1], pseudo-labeling [2], etc. In contrast to the above approaches, we propose to improve the classifier layer by introducing multiple prototypes to exploit the underlying subclass structures. In specific, the linear classifier $g(\mathbf{Z}; \Omega)$ can be expressed as below if bias is removed,

$$g(\mathbf{Z}; \Omega) = \Omega^\top \mathbf{Z}, \quad s.t. \quad \Omega \in \mathbb{R}^{D_o \times K} \quad (1)$$

Training the linear classifier with cross-entropy loss can be seen as discovering K prototypes, each represented as $\omega_k \in \mathbb{R}^{D_o}$, in the encoded feature space such that points belonging to the same category group together and center around the prototype, while pushing different categories away [4]. With such a design, it is assumed that a single prototype ω_k is discovered for each semantic category. However, observing subclasses within each semantic category, the single prototype assumption could be too strong and potentially increases the

risk of underfitting when small labeled data is available. To tackle this issue, we introduce a multi-prototype memory bank which maintains multiple prototypes for each class and each prototype accounts for one subclass. An overview of MulPro is illustrated in Fig. 2.

B. Multi-Prototype Classifier

We define a multi-prototype memory bank as $\Omega \in \mathbb{R}^{D_o \times M \times K}$. In the forward pass, given encoded feature \mathbf{Z} , we first take the inner product with all prototypes and this results in an attention map $\mathbf{A} \in \mathbb{R}^{N \times M \times K}$ as below.

$$a_{nkm} = \sum_d z_{dn} \cdot \omega_{dmk} \quad (2)$$

We notice that when $M = 1$ the multi-prototype classifier simply degenerates to standard linear classifier. With $M > 1$, we apply a maxpooling operation on the attention map \mathbf{A} and this yields the classification logits $l_{kn} = \max_m a_{nkm}$. The logits are eventually used for calculating the cross-entropy loss as,

$$\mathcal{L}_{CE} = -\frac{1}{N} \sum_n \sum_k y_{kn} \log \frac{\exp(l_{kn})}{\sum_k \exp(l_{kn})} \quad (3)$$

Discussion. We provide a few insights for the multi-prototype design here. First, in the forward pass, multi-prototype classifier acts like a $K \times M$ way classifier, while the linear classifier is a K -way classifier. Each data point feature is evaluated against all prototypes through Eq. (2). Instead of directly classifying into one of the $K \times M$ classes, to respect the label ground-truth being K -way, the multi-prototype classifier first reduces the $K \times M$ -way prediction into K -way prediction through maxpooling over prototypes (the M dimension). As a result, $K \times M$ prototypes can be learned with only K -way labels provided. Because of the subclass structures, through this design we can identify the subclasses (up to M) as training goes on. Each prototype will naturally represent the subclass center.

C. Multi-Prototype Updating

We now elaborate how the multi-prototypes are updated. We first denote the activated prototype $\omega_{\hat{m}\hat{k}}$ as the following,

$$\omega_{\hat{m}\hat{k}_n} \in \mathbb{R}^{D_o} \quad s.t. \quad \hat{k}_n, \hat{m}_n = \arg \max_k \max_m a_{nkm} \quad (4)$$

Given an activated prototype $\omega_{\hat{m}\hat{k}}$, one could easily verify that the classification logit can be written as,

$$l_{kn} = \omega_{\hat{m}\hat{k}_n}^\top \mathbf{z}_n \quad (5)$$

Therefore, each activated prototype can be directly updated by backpropagating from cross-entropy loss defined in Eq. (3). Nevertheless, sparse labelled data provides weak supervision over prototypes and additional regularization is necessary for learning high quality prototypes. To this end, we further propose two constraints for training prototypes, namely subclass averaging constraint and prototype diversity constraint.

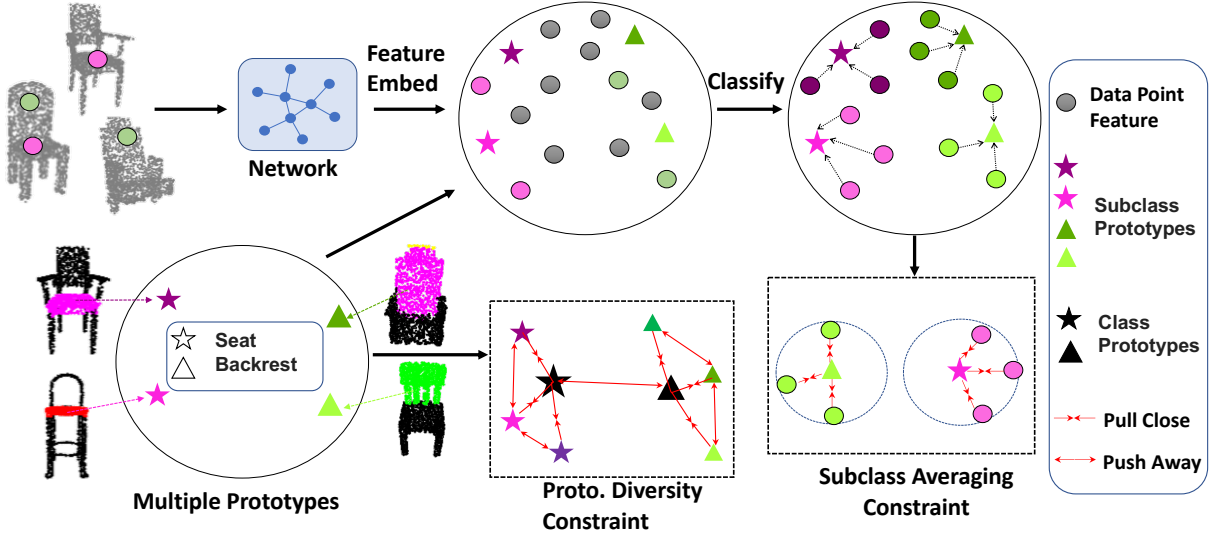


Fig. 2: An overview of MulPro for weakly supervised learning. With multiple prototypes per class (two stars and two triangles represent different types of seat and backrest respectively) data points are classified by the closest prototype. Subclass averaging and prototype diversity constraints are employed to learn multi-prototypes effectively.

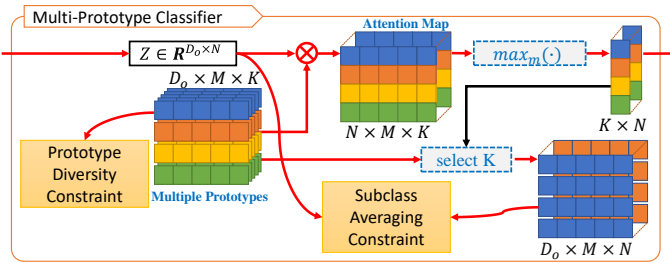


Fig. 3: Multi-prototype classifier schematic. Red lines indicate back propagation flow.

1) *Subclass Averaging Constraint*: The existing weakly supervised approaches update prototypes via exponential moving averaging over labeled data points [2]. As a result, it prohibits learning from unlabeled data points. In this section, we introduce a differentiable loss on all available data to provide supervision signal in addition to the cross-entropy loss.

Specifically, we first identify the per-class activated prototype for each point as $\Omega_{\hat{k}_n} = [\omega_{1\hat{k}_n}; \dots; \omega_{m\hat{k}_n}]$. These prototypes are further stacked as,

$$\tilde{\Omega} = [\Omega_{\hat{k}_1}; \dots; \Omega_{\hat{k}_N}] \in \mathbb{R}^{D_o \times M \times N} \quad (6)$$

The subclass averaging constraint \mathcal{L}_{avg} is then implemented as in Alg. 1. This algorithm accomplishes two tasks: activating the prototypes and updating the activated prototypes. We use a threshold γ to determine whether the data point feature belongs to one prototype. If one data point feature is similar to a prototype, above the threshold γ , \mathcal{L}_{avg} pulls them closer. Otherwise, we use it to update (activate) the farthest prototype. Therefore, the cosine distance $1 - \gamma$ represents the radius of a hypersphere covering the data points that are used to update the prototype in the center. More empirical analysis into the subclass averaging constraint design is presented in Sect. IV-E4.

Discussion. The proposed subclass averaging constraint can be interpreted as using data point features to update prototypes. Compared with moving average update adopted in [2], our design is superior in two ways. First, it is differentiable and can be combined with other learning objectives for an end-to-end training, while moving average update prevents combination with other losses to update the prototypes. Second, it allows using all data points, both labeled and unlabeled, to update prototypes. This enables discovering subclasses from all available data. Finally, since cosine similarity is agnostic to scale, minimizing \mathcal{L}_{avg} will not result in an explosion of scale.

2) *Prototype Diversity Constraint*: Empirical results from the experiment suggest that directly updating the multi-prototype with cross-entropy loss and subclass averaging constraint does not necessarily guarantee all subclasses being discovered. In another words, there is a risk of all prototypes within a semantic class collapsing into an identical one. To avoid the collapsing issue, we further propose prototype diversity constraints to encourage diverse prototypes being discovered.

Prototype Diversity within a Semantic Class. First of all, to encourage more diverse prototypes within a semantic class we penalize the accumulative similarity between prototypes as,

$$\mathcal{L}_{pd} = \sum_{k=1}^K \sum_{m_i=1}^M \sum_{m_j=1}^M sim(\omega_{m_i,k}, \omega_{m_j,k}) \quad (7)$$

For the selection of similarity metric sim , we take the following considerations into account. First, the similarity should avoid any trivial solution. Therefore any unconstrained similarity metrics should be excluded, e.g. inner product could result in an vanish of scale. Moreover, the diversity should not be overly emphasized, otherwise all prototypes could become equally distanced and they no longer characterize the subclasses within a single class. As a result, we propose to

adopt a piece-wise similarity function, specifically the follow thresholded cosine similarity is adopted,

$$sim = \max(0, \frac{\omega_{m_i,k}^\top \omega_{m_j,k}}{\|\omega_{m_i,k}\| \cdot \|\omega_{m_j,k}\|} - \sigma) \quad (8)$$

This indicates prototype diversity within a semantic class is encouraged only when the similarity is above a threshold σ and the normalized cosine similarity avoids scale vanishing.

Balancing Prototype Diversity and Class Separability. As discussed, overly emphasizing prototype diversity could result in equally distanced prototypes and harm the separability in semantic classes. To address this issue, we introduce a metric learning loss [28] to apply further constraints to the distribution of prototypes. We first denote the per-class mean prototype $\bar{\omega}_k$ as the average of all normalized prototypes within each semantic class, i.e. $\bar{\omega}_k = \frac{1}{M} \sum_m \omega_{mk} / \|\omega_{mk}\|_2$, and the scatterness of prototypes as the variance, $s_k = \frac{1}{M} \sum_m \|\omega_{mk} / \|\omega_{mk}\|_2 - \bar{\omega}_k / \|\bar{\omega}_k\|_2\|_2^2$. We define the constraint as minimizing the negative logarithm of the ratio between minimal inter-class mean prototype distance and maximal intra-class scatterness as in Eq. (9).

$$\begin{aligned} \mathcal{L}_{bds} &= -\log \frac{\min_{k_i, k_j} \|\frac{\bar{\omega}_{k_i}}{\|\bar{\omega}_{k_i}\|_2} - \frac{\bar{\omega}_{k_j}}{\|\bar{\omega}_{k_j}\|_2}\|_2^2}{\max_k s_k} \\ &= -\log \min_{k_i, k_j} \|\frac{\bar{\omega}_{k_i}}{\|\bar{\omega}_{k_i}\|_2} - \frac{\bar{\omega}_{k_j}}{\|\bar{\omega}_{k_j}\|_2}\|_2^2 + \log \max_k s_k \end{aligned} \quad (9)$$

D. Training Strategy

Finally, we combine cross-entropy loss, subclass averaging constraint and prototype diversity constraint to supervise the update of multi-prototype memory bank.

$$\mathcal{L} = \lambda_{CE} \mathcal{L}_{CE} + \lambda_{avg} \mathcal{L}_{avg} + \lambda_{pd} \mathcal{L}_{pd} + \lambda_{bds} \mathcal{L}_{bds} \quad (10)$$

We further notice that MulPro is compatible with the additional constraints and post-processing techniques introduced in [1].

Algorithm 1: Subclass averaging constraint algorithm

input : Point-wise encoded features $\mathbf{Z} \in \mathbb{R}^{N \times D_o}$ Prototypes within the category of $\omega_{\hat{m}_n \hat{k}_n}$ as $\tilde{\Omega} \in \mathbb{R}^{D_o \times M \times N}$
output: Subclass averaging loss \mathcal{L}_{avg}
 calculate cosine similarity between point-wise features and the prototypes as $\mathbf{S} \in \mathbb{R}^{N \times M}$, $s_{nm} = \frac{\omega_{\hat{m}_n \hat{k}_n}^\top \mathbf{z}_n}{\|\mathbf{z}_n\| \cdot \|\omega_{\hat{m}_n \hat{k}_n}\|}$
 initialize a $\mathbf{0}$ -matrix $\mathbf{W} \in \{0\}^{N \times M}$
for $n \leftarrow 1$ **to** N **do**
 if $\max_m s_{nm} > \gamma$ **then**
 $w_n = softmax(s_n / \tau)$
 else
 $w_n = softmax(-s_n / \tau)$
 $\mathcal{L}_{avg} = -\sum_n \sum_m w_{nm} \cdot s_{nm}$
return \mathcal{L}_{avg}

IV. EXPERIMENT

prototypes on ShapeNet dataset. In this section, we first introduce the benchmark datasets (Sect. IV-A). Next, we present details on our weakly supervised semantic segmentation experiments and compare with state-of-the-art methods (Sect. IV-B). We further provide analysis about multi-prototype (Sect. IV-C). Finally, the ablation study demonstrates the superiority of the multi-prototype classifier and the importance of the several losses that constrain the multi-prototype update (Sect. IV-E).

A. Datasets

We conduct experiments on two 3D point cloud segmentation datasets.

ShapeNet [29] is a richly-annotated, large-scale CAD model dataset including 16,881 shapes, divided into 16 categories, each annotated with 50 parts. We evaluate part segmentation task on this dataset following the weakly supervised setting proposed in [1]. For each training sample, a subset of points, 10 percent of all points or one point per part, are randomly selected to be labelled. For testing, the comparison is performed using the default protocol.

S3DIS [30] is an indoor real scene dataset, which is widely used as a benchmark dataset for 3D segmentation evaluation. It is composed of 6 areas each including several rooms, e.g. office areas, educational and exhibition spaces. For scene segmentation, it has 13 semantic categories of indoor scene objects. Each point is represented with xyz coordinate and RGB value. For weakly supervised supervised setting [1], a subset of points are uniformly labelled within each room. We choose Area 5 to be the test split.

PartNet [31] is a CAD model dataset with 24 shape categories and a total of 26,671 unique objects. For part segmentation task, we choose the most coarse level annotation. The experiment setting is kept the same with [1].

B. Weakly Supervised Semantic Segmentation

Encoder Network. We use the feature extraction encoder of DGCNN [13] with default parameters combined with our Multi-Prototype classifier as our network for fair comparison with previous work [1]. Here we set $\lambda_{CE} = \lambda_{avg} = \lambda_{pd} = \lambda_{bds} = 1$, the hyper-parameter threshold σ in prototype diversity constraint to 0.2 or 0.8 for different experiments, the hyper-parameter γ in subclass averaging constraint algorithm to $\cos \frac{\arccos \sigma}{2}$ and the τ to 0.1. M is set as 5 for ShapeNet, and 10 for S3DIS to obtain the best results.

Comparisons. We compare against DGCNN trained under fully supervision setting (Ful. Sup.), and previous weakly supervised approaches (Weak. Sup.). Among weakly supervised approaches, we compare with WeakSup [1] and One Thing One Click (OTOC) [2]. For a fair comparison with OTOC, we modify the encoder of One Thing One Click with DGCNN encoder network and retain the moving average update of memory bank/prototypes. The resulting method is thus termed OTOC*. Finally, we evaluate our multi-prototype classifier (MulPro) under the same settings.

TABLE I: mIoU (%) evaluation on ShapeNet dataset. The fully supervision (Ful. Sup.) methods are trained on 100% labelled points. Two levels of weak supervisions (1pt and 10%) are compared. The results presented here are with post-processing introduced in [1].

Setting	Model	SampAvg	CatAvg	Air.	Bag	Cap	Car	Chair	Ear.	Guitar	Knife	Lamp	Lap.	Motor.	Mug	Pistol	Rocket	Skate.	Table	
Ful.Sup.	DGCNN [13]	85.1	82.3	84.2	83.7	84.4	77.1	90.9	78.5	91.5	87.3	82.9	96.0	67.8	93.3	82.6	59.7	75.5	82.0	
Weak. Sup.	1pt	Π Model [32]	73.2	72.7	71.1	77.0	76.1	59.7	85.3	68.0	88.9	84.3	76.5	94.9	44.6	88.7	74.2	45.1	67.4	60.9
		MT [33]	72.2	68.6	71.6	60.6	79.3	57.1	86.6	48.4	87.9	80.0	73.7	94.0	43.3	79.8	74.0	45.9	56.9	59.8
		WeakSup [1]	75.5	74.4	75.6	74.4	79.2	66.3	87.3	63.3	89.4	84.4	78.7	94.5	49.7	90.3	76.7	47.8	71.0	62.6
		OTOC* [2]	75.5	74.3	75.0	74.0	85.6	63.2	88.2	58.0	89.0	85.1	80.6	94.6	48.4	91.6	72.2	49.8	70.7	62.2
		MulPro (Ours)	79.4	77.8	77.6	80.2	84.2	69.6	88.9	65.3	89.7	85.9	83.6	94.4	60.0	94.0	79.2	51.5	71.4	69.9
	10%	Π Model [32]	83.8	79.2	80.0	82.3	78.7	74.9	89.8	76.8	90.6	87.4	83.1	95.8	50.7	87.8	77.9	55.2	74.3	82.7
		MT [33]	81.7	76.8	78.0	76.3	78.1	64.4	87.6	67.2	88.7	85.5	79.0	94.3	63.3	90.8	78.2	50.7	67.5	78.5
		WeakSup [1]	85.0	81.7	83.1	82.6	80.8	77.7	90.4	77.3	90.9	87.6	82.9	95.8	64.7	93.9	79.8	61.9	74.9	82.9
		OTOC* [2]	85.0	82.2	83.5	84.1	85.0	77.2	90.7	76.7	91.2	88.0	84.9	95.8	68.8	93.2	81.7	57.4	74.9	81.5
		MulPro (Ours)	85.3	82.0	83.3	80.0	85.2	77.7	90.5	77.5	91.2	87.9	85.1	95.6	69.4	94.4	81.2	56.4	73.8	82.7

TABLE II: mIoU (%) evaluations on S3DIS (Area 5) dataset. We compared against fully supervised (Ful.Sup.) and alternative weakly supervised (Weak. Sup.) approaches.

Setting	Model	CatAvg	ceil.	floor	wall	beam	col.	win.	door	chair	table	book.	sofa	board	clutter	
Ful.Sup.	DGCNN [13]	47.0	92.4	97.6	74.5	0.5	13.3	48.0	23.7	65.4	67.0	10.7	44.0	34.2	40.0	
Weak. Sup.	1pt	Π Model [32]	44.3	89.1	97.0	71.5	0.0	3.6	43.2	27.4	62.1	63.1	14.7	43.7	24.0	36.7
		MT [33]	44.4	88.9	96.8	70.1	0.1	3.0	44.3	28.8	63.6	63.7	15.5	43.7	23.0	35.8
		WeakSup [1]	44.5	90.1	97.1	71.9	0.0	1.9	47.2	29.3	62.9	64.0	15.9	42.2	18.9	37.5
		OTOC* [2]	45.6	89.0	96.6	69.0	0.2	7.6	43.6	34.4	59.4	59.7	16.1	43.2	36.9	37.1
		MulPro (Ours)	47.5	90.1	96.3	71.8	0.0	6.7	46.7	39.2	67.2	67.4	21.8	39.2	33.0	38.0
	10%	Π Model [32]	46.3	91.8	97.1	73.8	0.0	5.1	42.0	19.6	66.7	67.2	19.1	47.9	30.6	41.3
		MT [33]	47.9	92.2	96.8	74.1	0.0	10.4	46.2	17.7	67.0	70.7	24.4	50.2	30.7	42.2
		WeakSup [1]	48.0	90.9	97.3	74.8	0.0	8.4	49.3	27.3	69.0	71.7	16.5	53.2	23.3	42.8
		OTOC* [2]	48.2	91.2	97.7	78.0	0.0	6.3	46.3	31.6	65.7	64.4	8.2	52.5	41.6	43.1
		MulPro (Ours)	49.0	89.7	96.9	75.5	0.0	14.0	45.7	40.7	68.5	66.8	13.9	49.4	34.4	41.2

TABLE III: Detailed results on PartNet dataset at 1pt annotation cost. We report mIoU (%) of segmentation for each object category.

Setting	Model	CatAvg	Bag	Bed	Bott.	Bowl	Chair	Clock	Dish.	Disp.	Door	Ear.	Fauc.	Hat	Key.	Knife	Lamp	Lap.	Micro.	Mug	Frid.	Scis.	Stora.	Table	Trash.	Vase
Ful.Sup.	DGCNN	65.6	53.3	58.6	48.9	66.9	69.1	35.8	75.2	91.2	68.5	59.3	62.6	63.7	69.5	71.8	38.5	95.7	57.6	83.3	53.7	89.7	62.6	65.3	67.8	66.8
Weak. Sup.	Baseline	50.2	24.4	30.1	20.5	38.0	65.9	35.3	64.9	84.3	52.6	36.7	47.1	47.9	52.2	55.2	34.1	92.4	49.3	59.5	49.6	80.1	44.6	49.8	40.4	49.5
	WeakSup [1]	54.6	28.4	30.8	26.0	54.3	66.4	37.7	66.3	81.0	51.7	44.4	51.2	55.2	56.2	63.1	37.6	93.5	49.7	73.5	50.6	83.6	46.8	61.1	44.1	56.8
	MulPro (Ours)	60.9	55.2	37.0	51.2	57.3	64.3	44.2	77.5	85.7	59.6	56.6	55.9	59.4	56.3	66.3	38.8	93.4	57.3	80.0	53.1	83.0	53.4	63.1	51.6	62.3

Evaluation Metric. We calculate the mean Intersect over Union (mIoU) for each test sample as its evaluation metric. For ShapeNet, we present the average mIoU over all samples (SampAvg) and the average mIoU over all categories (CatAvg) which we firstly calculate the average mIoU over samples in each category. For S3DIS, we present the average mIoU over all categories (CatAvg).

1) Quantitative Results for Semantic Segmentation:

ShapeNet Part Segmentation. We present the results on ShapeNet part segmentation in Tab. I. We make the following observations from the results. i) Our model outperforms other weakly supervised models significantly. For 1 point weak supervision, our model surpasses previous state-of-the-art model by nearly 4% and is only 5% lower than fully supervised approach. For 10% weak supervision, our model demonstrates comparable to OTOC*, surpasses WeakSup by 0.3% and is closer to the full supervision approach. ii) Improvement is more substantial at lower labeling regime, suggesting the

multi-prototype classifier is particularly effective when labels are sparse.

S3DIS Semantic Segmentation. We present the results on S3DIS semantic segmentation in Tab. II. We make following observations similarly. i) Our model outperforms other models, even comparable to fully supervised approach with only 1pt (less than 0.2%) labelled point. For 1pt weak supervision, our model outperform WeakSup [1] by 3% mIoU. For 10% weak supervision, our model surpasses the previous state-of-the-art model by 0.8% and is better than fully supervised approach. ii) Most of the categories have seen a big boost with few labeled data thanks to the multi-prototype classifier.

PartNet Semantic Segmentation. We present semantic segmentation results on PartNet segmentation task. We evaluate our MulPro final model with post-processing technique [1]. We also compare our results with WeakSup [1], the large margin suggests the efficacy of proposed multi-prototype classifier.

2) *Qualitative Results for Semantic Segmentation:* We show qualitative results of point cloud segmentation and compare the segmentation quality. For S3DIS dataset, we visualize selected segmentation samples in Fig. 5. From left to right, the RGB view, ground-truth, fully supervised segmentation, WeakSup [1] segmentation and our MulPro result are visualized. In these visualization results, both MulPro and WeakSup leverages 1pt labelled points in the training stage. We observe that our results better respect the ground-truth for classes with large intra-class variation. For example, a “clutter” category (in black color) exists in S3DIS which covers multiple types of objects that do not fall into the other 12 predefined classes. Because of the multi-prototype classifier, our model is able to identify subclasses within this “clutter” category. This is reflected by the more consistent predictions for “clutter” class. In contrast, WeakSup makes more erroneous predictions on the “clutter” class. For ShapeNet, we show the segmentation results in Fig. 6. These examples again demonstrate competitive performance by our model when facing categories with large intra-class variation, such as the examples of the car, lamp and plane.

C. Multi-Prototype Classifier Analysis

Discovering Subclasses. Multi-prototype classifier is motivated by the subclass structures within a semantic class. In this section, we provide qualitative results on the subclasses discovered by MulPro. In specific, for each point we define the corresponding activated prototype following Eq. (4). Given a maximal M prototypes for each category we can thus assign each point into one of the prototype by its activation. In Fig. 4, we selectively visualize points by the activated prototype, i.e. each individual color indicates one activated prototype. We are surprised to see many subclasses identified. For instance, despite a single *body* class is annotated for all planes, our multi-prototype classifier discovers an additional subclasses marked as red points in Fig. 4, corresponding to the tail part of *body* which is shared by all passenger jets but absent from fighter jets. Two types of wings are also discovered from the plane category, roughly differentiating passenger jets from fighters. Subclasses are also discovered from chairs, different *back supports* and *seats* are discovered, roughly distinguishing chairs with armrest from others. Subclasses are identified for *lampshade* as well, the ones in red are generally pendants while the pink ones are mostly floor lamps. The *table surface* again displays subclass structures with square-shaped desks being identified from round-shaped tables. The consistent and clean activation of prototypes among all semantic objects implies an obvious subclasses structures in the feature space.

D. Multi-Prototype Classifier at Higher Label Budget

Multi-prototype captures the intra-class variance and is motivated under the weakly supervised setting. As demonstrated by many research on large-scale datasets with large intra-class variation, e.g ImageNet, neural network is able to model the multi-modal distribution and a single prototype is enough for classification. However, when labeled data points are few the neural network would face underfitting, i.e. it fails to capture

the complex data distribution and can not group features in tight clusters. Therefore, multiple prototypes are necessary at weakly supervised learning.

To validate the advantage of MulPro, we carry out additional experiments at 100% labeled data. The results in Tab. IV clearly suggests multi-prototype is most effective at 1pt annotation and the gap between DGCNN and MulPro diminishes at higher labeling regime. This observation also motivates us to explore multiple prototypes in weakly supervised setting.

TABLE IV: Comparing MulPro at different labeling budgets with DGCNN as backbone.

Method	Annotation	SampAvg(%)	CatAvg(%)
DGCNN	1pt	72.6	72.2
DGCNN	10%	84.5	81.5
DGCNN	100%	85.1	82.3
DGCNN + MulPro	1pt	79.4	77.8
DGCNN + MulPro	10%	85.3	82.0
DGCNN + MulPro	100%	85.5	82.4

E. Ablation and Additional Study

1) *Importance of Individual Components.:* We analyze the importance of the proposed modules. Different combinations of the modules are evaluated on ShapeNet benchmark dataset with 1pt annotation scheme. The results are presented in Tab. VI. We notice that the prototype diversity must simultaneously incorporate Eq. (7) and Eq. (9) to avoid trivial solution and we combine them in the ablation study. From the ablation results, we first evaluate multi-prototype classifier alone, and it yields slightly worse result than baseline, suggesting multi-prototype cannot be effectively trained with cross-entropy alone. Then we combine multi-prototype classifier with subclass averaging loss, this improves upon multi-prototype alone, indicating multi-prototype requires more supervision than cross-entropy loss to update. Finally, we combine prototype diversity terms and demonstrate the best result. We also evaluate our results using the post-processing technique (w/PP) proposed in [1] which implements label propagation on network prediction and observe further improvement.

2) *Compatibility with Alternative Backbone:* We further evaluate the multi-prototype classifier with state-of-the-art 3D point cloud backbone network, namely the Point Cloud Transformer (PCT) [18] and present the results at 1pt annotation on S3DIS dataset in Tab. V. Significant improvement is observed by combining PCT with Multi-prototype classifier.

TABLE V: The results of Point Cloud Transformer encoder on S3DIS dataset.

Method	Annotation	mIoU (%)
PCT [18]	1pt	41.6
PCT + MulPro	1pt	43.0
PCT	100%	51.5

3) *Number of Multi-Prototype:* We investigate the impact of the number of multi-prototype on segmentation performance. In specific, we evaluate $M = 1 \cdots 10$ on ShapeNet

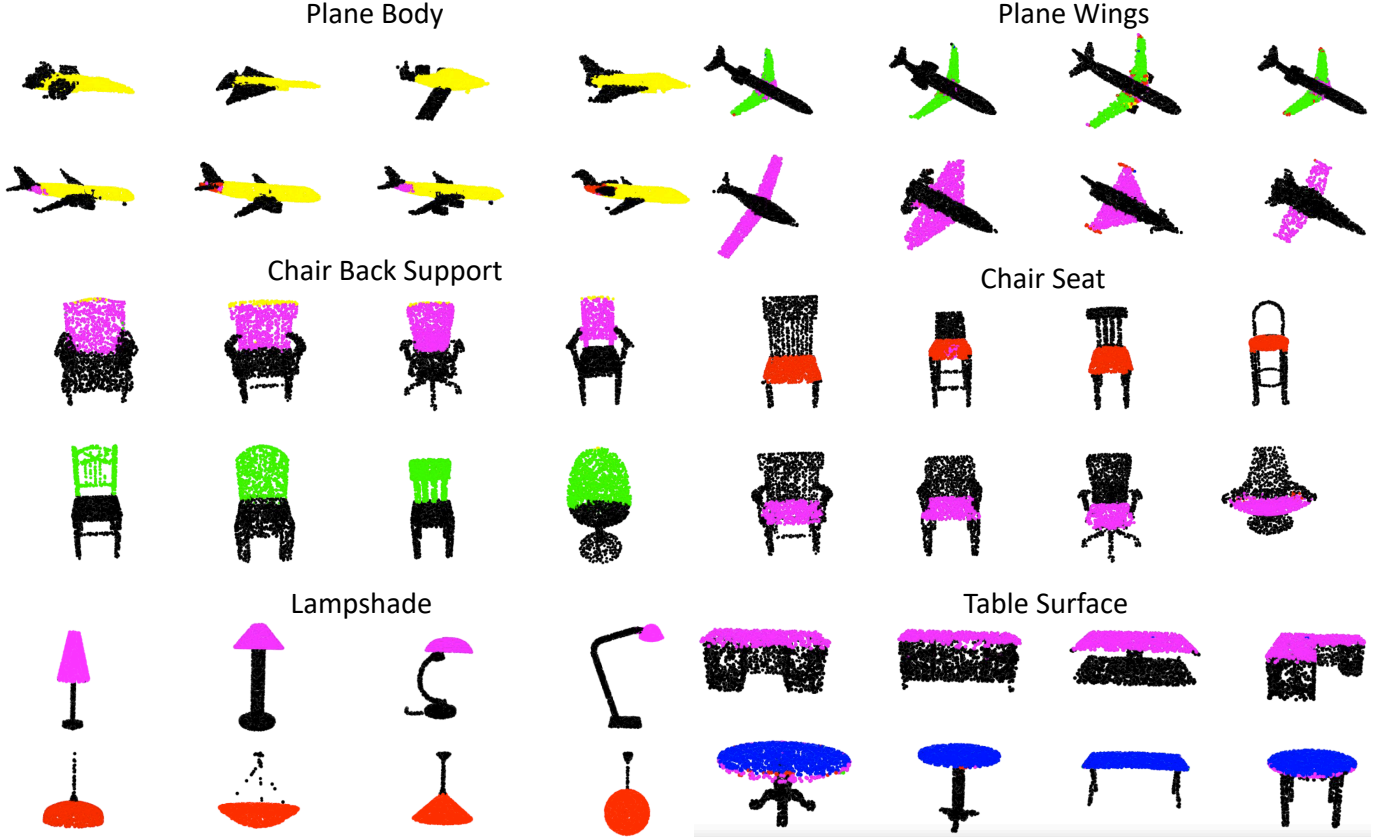


Fig. 4: Visualization of activated prototypes (indicated by different colors) on selected samples from ShapeNet dataset. 1pt labelled points are used to train the weak supervision model.

TABLE VI: Ablation study on the impact of individual modules. The results consist of without Post Processing (wo/PP) and with Post Processing (w/PP).

Components				mIoU (%)	
Multi-Prototype	Subclass	Averaging	Prototype Diversity	w/o PP	w/ PP [1]
-	-	-	-	73.8	74.4
✓	-	-	-	73.7	-
✓	✓	-	-	75.4	-
✓	✓	✓	✓	76.4	77.8

segmentation with 1 point per category label. We present the results in Fig. 7 from which we observe that the number of activated prototypes increases with the increase of the number of available prototypes (M), however, the mean category IoU reaches the maximum value when $M = 5$.

4) *Subclass Averaging Constraint Design.*: Due to the importance of subclass averaging loss, we investigate several alternative designs. First, a naive way to use both labeled and unlabeled data to update prototype is through pseudo-labeling [34]. We predict the pseudo-labels for all unlabeled data points and the pseudo-labels are in turn used to supervised cross-entropy loss on unlabeled data. Alternatively, we could use Frobenius norm to measure the distance between data points and corresponding activated prototypes. Specifically, we stack all activated prototypes over data points as, $\hat{\Omega} = [\omega_{\hat{m}_1 \hat{k}_1}; \dots; \omega_{\hat{m}_N \hat{k}_N}] \in \mathbb{R}^{D_o \times N}$. Then the subclass averaging

loss is calculated as, $\mathcal{L}_{avg1} = \|\hat{\Omega} - \mathbf{Z}\|_F^2$. Since the distance metric consists of Frobenius norm and inner product between $\hat{\Omega}$ and \mathbf{Z} , the Frobenius norm parts will affect the scale, it might lead to trivial solutions. To avoid these trivial solutions, we propose the second candidate, using cosine distance to perform subclass averaging loss as,

$$\mathcal{L}_{avg2} = - \sum_n \frac{\langle \omega_{\hat{m}_n \hat{k}_n}, \mathbf{z}_n \rangle}{\|\omega_{\hat{m}_n \hat{k}_n}\| \cdot \|\mathbf{z}_n\|} \quad (11)$$

Results for comparing all alternative designs are presented in Tab. VII. We make the following observations from the results. Firstly, pseudo-labeling gives the worst results, probably due to the confirmation bias [35] to blame. Furthermore, using the cosine similarity to update prototypes \mathcal{L}_{avg2} avoids the trivial solution and the performance is slightly better than that using L2 distance alone \mathcal{L}_{avg1} . Finally, our thresholded subclass averaging outperforms both candidates, suggesting it is necessary to selectively use most relevant point features to update prototypes.

We further compare the number of uniquely activated prototypes under different subclass averaging losses. We present the results on ShapeNet and S3DIS in Tab. VIII and Tab. IX respectively. We make the following observations from the results. First, the numbers of activated prototypes are very few with both L2 distance and cosine distance as subclass averaging loss on both ShapeNet and S3DIS. This is probably due to the challenge in prototype initialization. When one or a

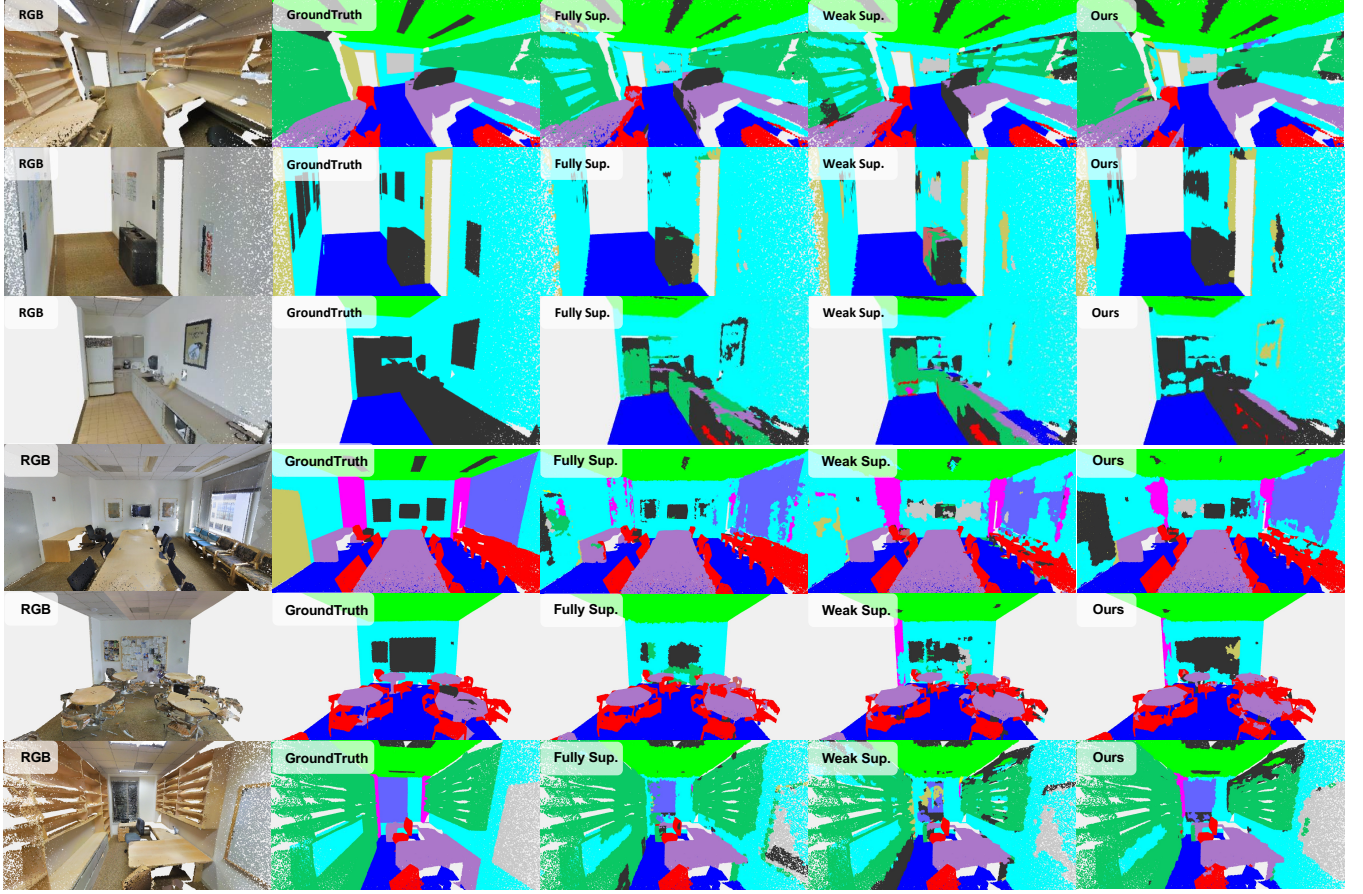


Fig. 5: Qualitative examples for S3DIS semantic segmentation. Weak Sup. refers to the results of [1].

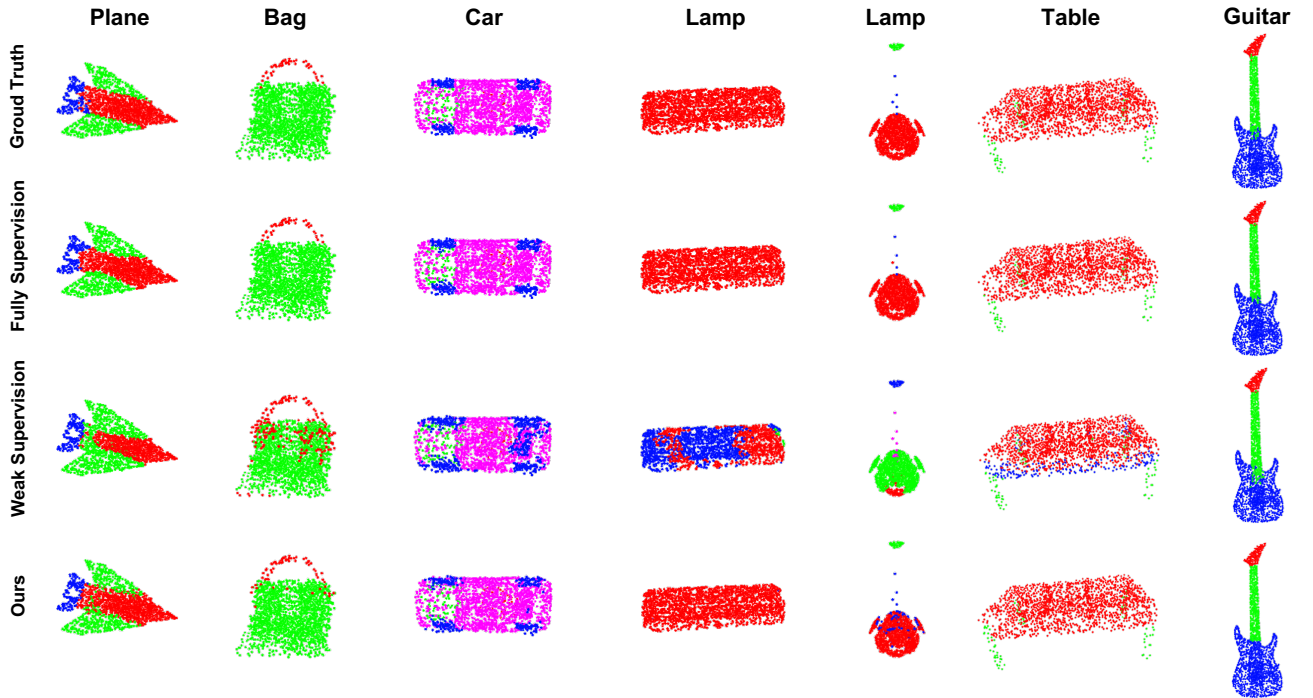


Fig. 6: Qualitative examples for ShapeNet part segmentation. Weak supervision refers to the method of [1].

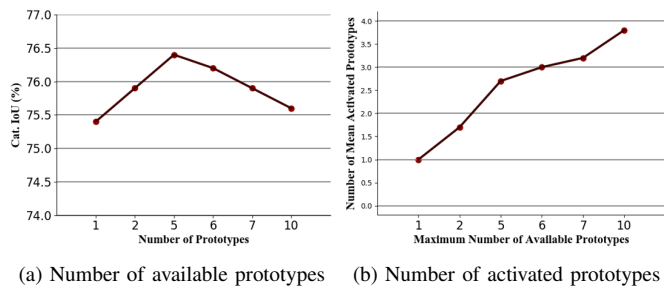


Fig. 7: The impact of #prototypes on ShapeNet dataset. (a) shows the change of mIoU with the number of prototypes; (b) shows the change of the mean number of activated prototypes with the number of prototypes.

TABLE VII: The results of different options of the subclass averaging loss.

Subclass Averaging Options	ShapeNet	S3DIS
Pseudo-Labeling	74.2	44.4
$\mathcal{L}_{CE} + \mathcal{L}_{avg1} + \mathcal{L}_{pd} + \mathcal{L}_{bds}$	74.8	44.9
$\mathcal{L}_{CE} + \mathcal{L}_{avg2} + \mathcal{L}_{pd} + \mathcal{L}_{bds}$	75.8	45.5
$\mathcal{L}_{CE} + \mathcal{L}_{avg} + \mathcal{L}_{pd} + \mathcal{L}_{bds}$ (Ours)	76.4	46.8

few prototypes are activated there is no force to encourage the activation of other ones. Moreover, with our proposed subclass averaging loss, we notice certain classes have more activated prototypes, e.g. “airplane”, “car”, “chair” on ShapeNet and “ceiling”, “door”, “chair”, “table”, “clutter” on S3DIS, suggesting large intra-class variation.

V. CONCLUSION

In this work, we first observe that existing approaches towards point cloud segmentation often employ a linear classifier to separate semantic classes. This is equivalent to allocating one prototype for each category. As we point out through experiment and intuition, clear subclass structures in each semantic class of 3D point cloud segmentation data widely exist and would result in large intra-class variation in feature representation. As a result, the single prototype may not capture the large variation and lead to inferior results. To tackle this issue, we proposed a multi-prototype memory bank where each prototype serves as the classifier for one subclass. To enable effective multi-prototype training, we further introduced two constraints. Extensive results on weakly supervised 3D point cloud segmentation benchmark suggest the advantage of maintaining multiple prototypes in particular at low-label regime. We hope the subclasses identified from ShapeNet could provide insights into future segmentation model design at low-label regime.

REFERENCES

- [1] X. Xu and G. H. Lee, “Weakly supervised semantic point cloud segmentation: Towards 10x fewer labels,” in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020.
- [2] Z. Liu, X. Qi, and C.-W. Fu, “One thing one click: A self-training approach for weakly supervised 3d semantic segmentation,” in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2021, pp. 1726–1736.

- [3] A. Tao, Y. Duan, Y. Wei, J. Lu, and J. Zhou, “Seggroup: Seg-level supervision for 3d instance and semantic segmentation,” *arXiv preprint arXiv:2012.10217*, 2020.
- [4] M. Boudiaf, J. Rony, I. M. Ziko, E. Granger, M. Pedersoli, P. Piantanida, and I. B. Ayed, “A unifying mutual information view of metric learning: cross-entropy vs. pairwise losses,” in *European Conference on Computer Vision*. Springer, 2020, pp. 548–564.
- [5] K. Allen, E. Shelhamer, H. Shin, and J. Tenenbaum, “Infinite mixture prototypes for few-shot learning,” in *International Conference on Machine Learning*. PMLR, 2019, pp. 232–241.
- [6] O. Rippel, M. Paluri, P. Dollar, and L. Bourdev, “Metric learning with adaptive density discrimination,” in *International Conference on Learning Representations (ICLR)*, 2016.
- [7] R. Q. Charles, H. Su, M. Kaichun, and L. 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.
- [8] C. R. Qi, L. Yi, H. Su, and L. J. Guibas, “Pointnet++: Deep hierarchical feature learning on point sets in a metric space,” in *Advances in Neural Information Processing Systems*, I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, Eds. Curran Associates, Inc., 2017.
- [9] Y. Li, R. Bu, M. Sun, W. Wu, X. Di, and B. Chen, in *Advances in Neural Information Processing Systems*, S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, Eds., 2018.
- [10] W. Wu, Z. Qi, and L. Fuxin, “Pointconv: Deep convolutional networks on 3d point clouds,” in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019.
- [11] S. Wang, S. Suo, W.-C. Ma, A. Pokrovsky, and R. Urtasun, “Deep parametric continuous convolutional neural networks,” in *CVPR*, 2018.
- [12] H. Thomas, C. R. Qi, J.-E. Deschaud, B. Marcotegui, F. Goulette, and L. Guibas, “Kpconv: Flexible and deformable convolution for point clouds,” in *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*, 2019, pp. 6410–6419.
- [13] Y. Wang, Y. Sun, Z. Liu, S. E. Sarma, M. M. Bronstein, and J. M. Solomon, “Dynamic graph cnn for learning on point clouds,” *ACM Transactions on Graphics (TOG)*, 2019.
- [14] L. Landrieu and M. Simonovsky, “Large-scale point cloud semantic segmentation with superpoint graphs,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2018.
- [15] G. Li, M. Müller, A. Thabet, and B. Ghanem, “Deepgcns: Can gcns go as deep as cnns?” in *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*, 2019, pp. 9266–9275.
- [16] H. Lei, N. Akhtar, and A. Mian, “Spherical kernel for efficient graph convolution on 3d point clouds,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 43, no. 10, pp. 3664–3680, 2021.
- [17] H. Zhao, L. Jiang, J. Jia, P. H. Torr, and V. Koltun, “Point transformer,” in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2021.
- [18] M.-H. Guo, J.-X. Cai, Z.-N. Liu, T.-J. Mu, R. R. Martin, and S.-M. Hu, “Pct: Point cloud transformer,” *Computational Visual Media*, 2021.
- [19] Y. Guo, H. Wang, Q. Hu, H. Liu, L. Liu, and M. Bennamoun, “Deep learning for 3d point clouds: A survey,” *IEEE transactions on pattern analysis and machine intelligence*, 2020.
- [20] Q. Meng, W. Wang, T. Zhou, J. Shen, Y. Jia, and L. Van Gool, “Towards a weakly supervised framework for 3d point cloud object detection and annotation,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2021.
- [21] J. Wei, G. Lin, K.-H. Yap, T.-Y. Hung, and L. Xie, “Multi-path region mining for weakly supervised 3d semantic segmentation on point clouds,” in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020, pp. 4384–4393.
- [22] B. Zhou, A. Khosla, A. Lapedriz, A. Oliva, and A. Torralba, “Learning deep features for discriminative localization,” in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 2921–2929.
- [23] X. Shi, X. Xu, K. Chen, L. Cai, C. S. Foo, and K. Jia, “Label-efficient point cloud semantic segmentation: An active learning approach,” *arXiv preprint arXiv:2101.06931*, 2021.
- [24] S. Xie, J. Gu, D. Guo, C. R. Qi, L. Guibas, and O. Litany, “Pointcontrast: Unsupervised pre-training for 3d point cloud understanding,” in *European Conference on Computer Vision*. Springer, 2020, pp. 574–591.
- [25] J. Snell, K. Swersky, and R. S. Zemel, “Prototypical networks for few-shot learning,” in *Advances in Neural Information Processing Systems*, 2017.

TABLE VIII: The numbers of activated prototypes in each shape on ShapeNet dataset.

Subclass Averaging Options	Air.	Bag	Cap	Car	Chair	Ear.	Guitar	Knife	Lamp	Lap.	Motor.	Mug	Pistol	Rocket	Skate.	Table
$\mathcal{L}_{CE} + \mathcal{L}_{avg1} + \mathcal{L}_{pd} + \mathcal{L}_{bds}$	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
$\mathcal{L}_{CE} + \mathcal{L}_{avg2} + \mathcal{L}_{pd} + \mathcal{L}_{bds}$	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
$\mathcal{L}_{CE} + \mathcal{L}_{avg} + \mathcal{L}_{pd} + \mathcal{L}_{bds}$ (Ours)	5	1	1	5	5	1	3	3	5	1	1	3	2	1	1	5

TABLE IX: The numbers of activated prototypes in each semantic part on S3DIS (Area 5) dataset.

Subclass Averaging Options	ceil.	floor	wall	beam	col.	win.	door	chair	table	book.	sofa	board	clutter
$\mathcal{L}_{CE} + \mathcal{L}_{avg1} + \mathcal{L}_{pd} + \mathcal{L}_{bds}$	1	1	5	1	1	1	2	1	2	1	1	1	6
$\mathcal{L}_{CE} + \mathcal{L}_{avg2} + \mathcal{L}_{pd} + \mathcal{L}_{bds}$	1	1	5	1	1	1	3	3	2	1	10	1	5
$\mathcal{L}_{CE} + \mathcal{L}_{avg} + \mathcal{L}_{pd} + \mathcal{L}_{bds}$ (Ours)	9	7	7	2	4	1	6	3	6	2	4	2	6

- [26] N. Zhao, T.-S. Chua, and G. H. Lee, “Few-shot 3d point cloud semantic segmentation,” in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2021.
- [27] J. Deuschel, D. Firmbach, C. I. Geppert, M. Eckstein, A. Hartmann, V. Bruns, P. Kuritsyn, J. Dextl, D. Hartmann, D. Perrin, T. Wittenberg, and M. Benz, “Multi-prototype few-shot learning in histopathology,” in *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV) Workshops*, October 2021, pp. 620–628.
- [28] X. Xu, L. Zhang, L.-F. Cheong, Z. Li, and C. Zhu, “Learning clustering for motion segmentation,” *IEEE Transactions on Circuits and Systems for Video Technology*, 2021.
- [29] L. Yi, V. G. Kim, D. Ceylan, I. Shen, M. Yan, H. Su, C. Lu, Q. Huang, A. Sheffer, L. Guibas *et al.*, “A scalable active framework for region annotation in 3d shape collections,” *ACM Transactions on Graphics*, 2016.
- [30] I. Armeni, O. Sener, A. R. Zamir, H. Jiang, I. Brilakis, M. Fischer, and S. Savarese, “3d semantic parsing of large-scale indoor spaces,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2016.
- [31] K. Mo, S. Zhu, A. X. Chang, L. Yi, S. Tripathi, L. J. Guibas, and H. Su, “Partnet: A large-scale benchmark for fine-grained and hierarchical part-level 3d object understanding,” in *CVPR*, 2019.
- [32] S. Laine and T. Aila, “Temporal ensembling for semi-supervised learning,” in *International Conference on Learning Representations (ICLR)*, 2017.
- [33] A. Tarvainen and H. Valpola, “Mean teachers are better role models: Weight-averaged consistency targets improve semi-supervised deep learning results,” in *Advances in Neural Information Processing Systems*, I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, Eds. Curran Associates, Inc., 2017.
- [34] A. Iscen, G. Tolias, Y. Avrithis, and O. Chum, “Label propagation for deep semi-supervised learning,” in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019.
- [35] E. Arazo, D. Ortego, P. Albert, N. E. O’Connor, and K. McGuinness, “Pseudo-labeling and confirmation bias in deep semi-supervised learning,” in *2020 International Joint Conference on Neural Networks (IJCNN)*. IEEE, 2020, pp. 1–8.