No Time to Train: Empowering Non-Parametric Networks for Few-shot 3D Scene Segmentation

Xiangyang Zhu*[1,3], Renrui Zhang*†‡[2,3], Bowei He[1], Ziyu Guo[2,3], Jiaming Liu[4]
Han Xiao[3], Chaoyou Fu[5], Hao Dong[4], Peng Gao[3]

* Equal contribution † Project leader ‡ Corresponding author

[1]City University of Hong Kong [2]The Chinese University of Hong Kong
[3]Shanghai AI Lab [4]Peking University [5]Tencent Youtu Lab
{xiangyzhu6-c, boweihe2-c}@my.cityu.edu.hk
{zhangrenrui, gaopeng}@pjlab.org.cn

Abstract

*To reduce the reliance on large-scale datasets, recent works in 3D segmentation resort to few-shot learning. Current 3D few-shot segmentation methods first pre-train models on 'seen' classes, and then evaluate their generalization performance on 'unseen' classes. However, the prior pre-training stage not only introduces excessive time overhead but also incurs a significant domain gap on 'unseen' classes. To tackle these issues, we propose a **N**on-parametric **N**etwork for few-shot 3D **Seg**mentation, **Seg-NN**, and its **P**arametric variant, **Seg-PN**. Without training, Seg-NN extracts dense representations by hand-crafted filters and achieves comparable performance to existing parametric models. Due to the elimination of pre-training, Seg-NN can alleviate the domain gap issue and save a substantial amount of time. Based on Seg-NN, Seg-PN only requires training a lightweight QUEry-Support Transferring (QUEST) module, which enhances the interaction between the support set and query set. Experiments suggest that Seg-PN outperforms previous state-of-the-art method by **+4.19%** and **+7.71%** mIoU on S3DIS and ScanNet datasets respectively, while reducing training time by **-90%**, indicating its effectiveness and efficiency. Code is available here.*

1. Introduction

Point cloud segmentation is an essential procedure in autonomous driving [8, 25], robotics [1, 11], and other computer vision applications [3, 30]. To achieve favorable segmentation, many learning-based methods have been proposed [9, 12, 13, 28]. However, such algorithms require the construction of large-scale and accurately annotated datasets, which is expensive and time-consuming.

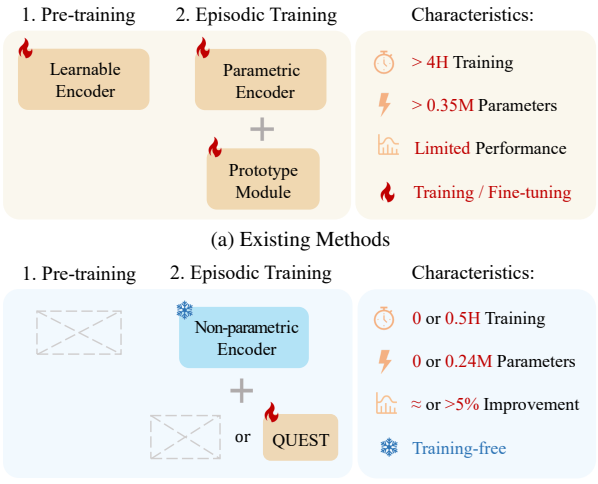
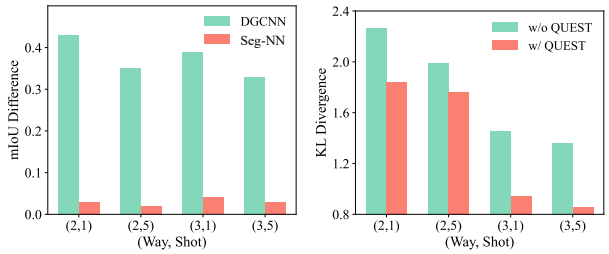


Figure 1. **Comparison of Existing Methods and Our Approaches.** Our non-parametric Seg-NN contains no learnable parameters and thus discards both pre-training and episodic training stages with superior efficiency, and the parametric Seg-PN further improves the performance with a lightweight QUEST module.

The data-hungry problem can be effectively mitigated by the few-shot learning strategy, which has garnered significant attention [6]. From Fig. 1 (a), existing 3D few-shot segmentation methods basically follow the meta-learning scheme to learn a 3D encoder and a prototype generation module. They mainly take three steps to solve the problem:

1. **Pre-training on 'seen' classes** by supervised learning. Considering the lack of pre-trained models in the 3D field, this step trains a learnable 3D encoder, e.g., DGCNN [24], to obtain the ability to extract general 3D point cloud representations.



(a) Performance Difference of Seen and Unseen Classes

(b) KL Divergence of Logits between Support and Query Sets

Figure 2. **Alleviating Domain Gap by Seg-NN (a) and Prototype Bias by Seg-PN (b)** on S3DIS [2]. The horizontal axis presents the number of ways and shots in the form of (Way, Shot).

2. **Episodic training on 'seen' classes** to fit the query-support few-shot segmentation tasks. In this step, the pre-trained encoder is appended with a learnable prototype module and fine-tuned to extract discriminative prototypes from the support set, which are utilized to guide the semantic segmentation of the query set.
3. **Testing on 'unseen' classes** to evaluate the model. After episodic training, the model is evaluated on test episodes that contain unseen classes. The model is expected to segment new classes by the same query-support paradigms as the episodic training.

However, this pipeline encompasses two noteworthy issues: 1) the learnable encoder being pre-trained and fine-tuned on 'seen' classes will inevitably introduce a domain gap when evaluated on 'unseen' classes; 2) the complexity of the training process, including pre-training and episodic training, incurs substantial time and resource overhead.

To address these issues, we extend the non-parametric network, Point-NN [35], to few-shot 3D scene segmentation tasks and propose a new model, **Seg-NN**, which is both efficient and effective. Seg-NN inherits the non-parametric encoder from Point-NN to encode 3D scenes but makes the following modifications: 1) We project the position and color information into a shared feature space and integrate them to obtain a comprehensive representation. 2) To reduce the noises and perturbations in natural 3D scenes, we sample the robust low frequencies and filter out the noisy high-frequency components. After encoding, we leverage the category prototypes to predict the segmentation masks for the query set by similarity matching. As shown in Fig. 1 (b), Seg-NN discards all two stages of pre-training and episodic training and performs comparably to some existing parametric methods. Such a training-free property simplifies the few-shot training pipeline with minimal resource consumption and mitigates the domain gap caused by different training-test categories. As visualized in Fig. 2 (a), we observe that Seg-NN shows marginal performance difference between seen and unseen categories, while the widely adopted DGCNN [24] encoder presents a much worse generalization ability due to cross-domain training and testing.

On top of this, we also propose a parametric variant, **Seg-PN**, to further boost the performance by efficient training. In detail, we adopt the non-parametric encoder of Seg-NN and append an additional parametric **QUE**ry-**S**upport **T**ransferring module, termed **QUEST**. QUEST enhances category prototypes in the support-set domain with the knowledge from the query-set domain, which suppresses prototype biases caused by the small few-shot support set. As shown in Fig. 2 (b), the reduced query-support distribution differences suggest that the prototypes are shifted to the query-set domain. Seg-PN only learns the QUEST module and does not require pre-training just as Seg-NN, as shown in Fig. 1 (b). Experiments show that Seg-PN achieves new state-of-the-art (SOTA) performance on both S3DIS [2] and ScanNet [4] datasets, surpassing the second-best by **+4.19%** and **+7.71%**, respectively, while reducing the training time by over **-90%**.

In summary, our contributions are as follows:

- We introduce a non-parametric few-shot learning framework, Seg-NN, for 3D point cloud semantic segmentation, which can also serve as a basis to construct the parametric better-performed variant, Seg-PN.
- We design a new query-support interaction module, QUEST, in Seg-PN to adapt category prototypes by learning the affinity between support and query sets.
- Comprehensive experiments are conducted to verify the efficiency and efficacy of our proposed method. We achieve SOTA results with the least parameters and a substantially simplified learning pipeline.

2. Problem Definition

We first illustrate the task definition of few-shot 3D semantic segmentation. We follow previous works [6, 37] to adopt the popular episodic training/test paradigm [23] after the pre-training stage. Each episode is instantiated as an N-way K-shot task, which contains a support set and a query set. The support set comprises N target classes, and each class corresponds to K point clouds with point-level labels. The query set contains a fixed number of point clouds that need to be segmented. Each episodic task aims to segment the query-set samples into N target classes along with a 'background' class based on the guidance of the support set.

To achieve this, we regard the semantic segmentation task as a point-level similarity-matching problem. We first utilize a feature encoder to extract the features of support-set point cloud samples and generate prototypes for all $N + 1$ classes. Then, we adopt the same feature encoder to obtain the feature of every point in query samples and conduct similarity matching with the prototypes for point-level classification. In this way, the query-set point clouds can be segmented into $N + 1$ semantic categories.

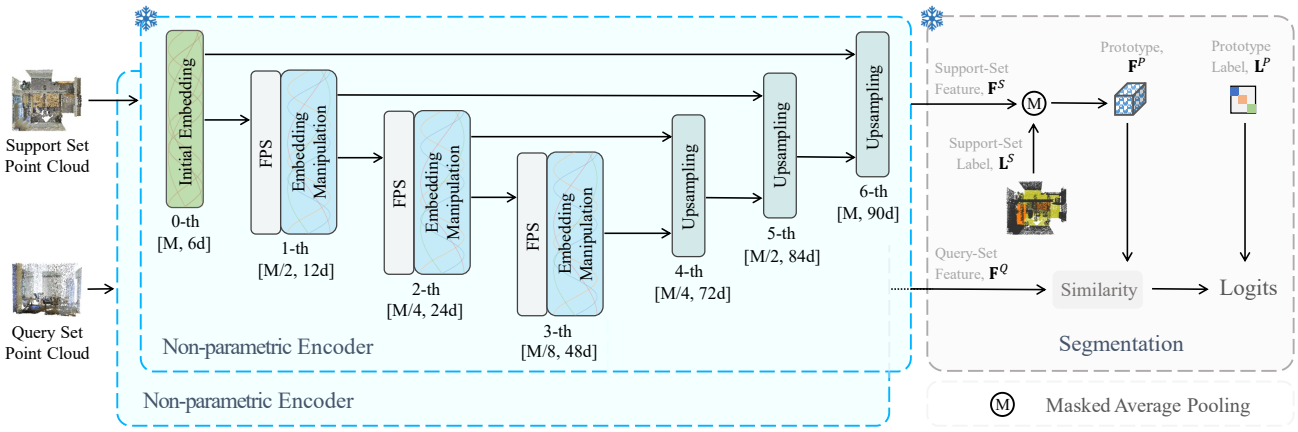


Figure 3. **The Framework of the Non-parametric Seg-NN**. The encoder extracts support- and query-set features and the segmentation head segments the query set based on similarity. To facilitate illustration, we assume the encoder consists of three manipulation layers.

In the following two sections, we respectively illustrate the details of our proposed efficient frameworks, the non-parametric Seg-NN and parametric Seg-PN.

3. Non-parametric Seg-NN

The detailed structure of Seg-NN is shown in Fig. 3, which contains a U-Net [19] style training-free encoder and a similarity-based segmentation head. The encoder embeds the point cloud into high-dimensional representations, and the segmentation head conducts similarity matching to produce the final prediction. Overall, Seg-NN adopts a similar framework to Point-NN [35], while we discard high-frequency noises to extract scene representations. The entire framework does not introduce any learnable parameters, thus deleting both pre-training and episodic training stages, differing from existing algorithms using DGCNN [24] as the learning-required encoder. Following, we describe the details of the encoder and the segmentation head.

3.1. Seg-NN Encoder

Given a point cloud $\{\mathbf{p}_i\}_{i=1}^{M}$ containing M points, our goal is to encode each point into embedding space in a training-free manner. Taking a random point $\mathbf{p} = (x, y, z)$ as an example, we denote its RGB color as $\mathbf{c} = (r, g, b)$. The encoder aims to extract shape knowledge based on both position and color information.

Firstly, we utilize trigonometric PEs to encode both positions and colors into high-dimension encodings as in Point-NN [35], which also map colors and positions into the same feature space. Specifically, we embed $\mathbf{p}$ and $\mathbf{c}$ with d frequencies $\mathbf{u} = [u_1, ..., u_d]$, denoted as $\mathrm{E}(\cdot)$:

$$\begin{aligned} \mathrm{E}(\mathbf{p};\mathbf{u}) &= [\sin(2\pi\mathbf{up}), \cos(2\pi\mathbf{up})] \quad \in \mathbb{R}^{6d}, \\ \mathrm{E}(\mathbf{c};\mathbf{u}) &= [\sin(2\pi\mathbf{uc}), \cos(2\pi\mathbf{uc})] \quad \in \mathbb{R}^{6d}, \end{aligned} \tag{1}$$

where $6d$ comes from the combination of 3 coordinates, (x, y, z), with 2 functions, $\sin(\cdot)$ and $\cos(\cdot)$. All frequency components in $\mathbf{u}$ adhere to a log-linear form, $u_i = \theta^{i/d}, i = 1, ..., d$, where θ and d are hyperparameters. Fig. 4 (a)(b) presents two encoding examples. The trigonometric PEs can not only encode the absolute point positions but also reveal the relative spatial relations between different points. For two points, $\mathbf{p_i}$ and $\mathbf{p_j}$, their embeddings $\mathrm{E}(\mathbf{p_i};\mathbf{u})$ and $\mathrm{E}(\mathbf{p_j};\mathbf{u})$ represent their absolute positional information. Then, the relative relation, $\mathbf{p_i} - \mathbf{p_j}$, can also be preserved by the dot production as $\mathrm{E}(\mathbf{p_i};\mathbf{u})\,\mathrm{E}(\mathbf{p_j};\mathbf{u})^T = \sum_{m=1}^{d} \cos\left((\mathbf{p_i}-\mathbf{p_j})u_m\right)$. Therefore, by such spatial-aware encoding, we effectively vectorize the point clouds.

We designate this as the 0-th layer and merge the position and color vector for a comprehensive embedding, $\mathbf{f}^0$,

$$\mathbf{f}^0 = \mathrm{E}(\mathbf{p};\mathbf{u}) + \mathrm{E}(\mathbf{c};\mathbf{u}) \quad \in \mathbb{R}^{6d}. \tag{2}$$

This trigonometric initial embedding can effectively represent position and color information. After that, we adopt two types of layers to further extract deep 3D embeddings as shown in Fig. 3: stacked ***Embedding Manipulation*** layers extract hierarchical embeddings by groups of hand-crafted filters; ***Upsampling*** layers integrate and upsample hierarchical embeddings to obtain the final point-level features.

3.1.1 Embedding Manipulation

As presented in Fig. 3, we stack embedding manipulation layers to get deep representations, which encode local shapes at different scales. Before each layer, we downsample the point cloud by half to increase the receptive field with Farthest Point Sampling (FPS). To obtain local embeddings, we denote point $\mathbf{p}$ as a local center and consider its neighborhood $\mathcal{N}_p$ searched by the k-Nearest Neighbor (k-NN) algorithm. We concatenate the neighbor point feature with the center feature along the channel dimension,

$$\hat{\mathbf{f}}_j^l = \mathrm{Concat}(\mathbf{f}^{l-1}, \mathbf{f}_j^{l-1}), \quad j \in \mathcal{N}_p, \tag{3}$$

where the subscript j represents the j-th point in $\mathbf{p}$'s neighborhood $\mathcal{N}_p$, and l denotes the l-th layer as in Fig. 3. After

concatenation, the expanded embedding $\hat{\mathbf{f}}_j^l \in \mathbb{R}^{2^l \times 6d}$ incorporate both center and neighbor information. The dimensionality is increased as $2^l \times 6d$ since we conduct Eq. 3 at every manipulation layer, each of which doubles the channel dimension. Then, to reveal local patterns, we design a group of filters based on previous efforts.

Existing works, *e.g.*, PointNet++ [18], DGCNN [24], and Point-MLP [16], efficiently extract features via multiple learning-required linear projections, which can be regarded as diverse filters from a frequency perspective. Inspired by this, we seek to manually design the filters. From Fig. 4 (a)(b), we can easily observe that the initial point encodings are band-limited signals. As shown in Fig. 4 (c), embedding frequencies are mainly distributed in the low- and high-frequency ranges. In this work, we aim to encode natural 3D scene points, which generally contain noises and perturbations. The large sharp noises contained in high frequencies may lead to significant differences in embeddings from clean points. Additionally, research has proved that neural networks tend to prioritize the learning of low-frequency information [15, 36], which indicates that low frequencies are discriminative and robust. Therefore, we rely on low-frequency bands to extract features and filter out high frequencies to further refine the scene representation.

We randomly sample low frequencies to construct a linear projection. As illustrated in Fig. 4 (d), we can follow Gaussian, uniform, Laplace, or other distributions to sample the frequencies. We denote them as $\mathbf{v} \in \mathbb{R}^{2^l \times 6d}$. Then, the projection weight is $\mathbf{W}^l = [\cos(2\pi \mathbf{v}\mathbf{k})] \in \mathbb{R}^{(2^l \times 6d) \times (2^l \times 6d)}$, where $\mathbf{k} = [1, ..., 2^l \times 6d]$. We integrate the relative position $\Delta \mathbf{p}_j$ from neighbor point j to the center point and color information for a comprehensive embedding. Thus, point j's embedding can be calculated via

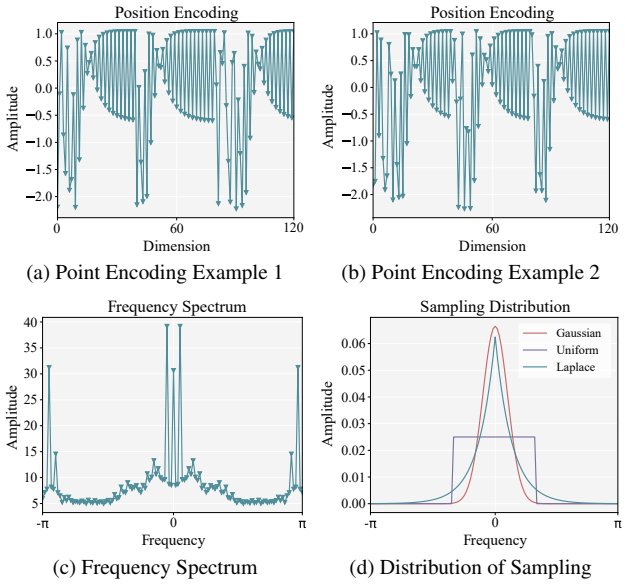
$$\mathbf{f}_j^l = \mathbf{W}^l \cdot \big(\hat{\mathbf{f}}_j^l + \mathrm{E}(\Delta \mathbf{p}_j; \mathbf{u}) + \mathrm{E}(\mathbf{c}_j; \mathbf{u})\big), \tag{4}$$

where $\mathbf{u}$ are log-linear frequencies similar to the initial encoding and $\mathbf{f}_j^l \in \mathbb{R}^{2^l \times 6d}$. Finally, we use maximum pooling to compress the neighborhood information into the central point and use $\mathbf{f}^l \in \mathbb{R}^{2^l \times 6d}$ to represent the embedding of point $\mathbf{p}$ in the l-th layer. We acquire hierarchical features $\{\mathbf{f}^l\}|_l$ from the manipulation layers and subsequently feed them into upsampling layers.

3.1.2 Upsampling

After all manipulation layers, we adopt the upsampling operation in Point-NN [35] to progressively upsample the hierarchical features to the input point number, as in Fig. 3. Specifically, we first interpolate the central point embedding via the weighted sum of neighbor point embeddings. Then, we concatenate the embedding of manipulation layers and upsampling layers in the channel dimension as the output.

After the upsampling layers, we obtain the final representation for each point, denoted as $\mathbf{f} \in \mathbb{R}^D, D = (2^0 + ... + 2^3) \times 6d$. In this way, the encoder produces point-level embeddings of the input point cloud, denoted as $\mathbf{F} = \{\mathbf{f}_m\}_{m=1}^M$, $\mathbf{F} \in \mathbb{R}^{M \times D}$, where M is point number.



(a) Point Encoding Example 1 (b) Point Encoding Example 2

(c) Frequency Spectrum (d) Distribution of Sampling

Figure 4. **Examples of Position Encodings and Frequencies,** where we set $d = 20$. (a) and (b) are examples of initial encodings. (c) is the average frequency spectrum over all points' encodings of a point cloud. (d) is the distribution of sampled frequencies.

By stacking these layers, the encoder can encode scene points without learnable parameters, so it eliminates tedious pre-training and episodic training. This advantage allows Seg-NN to save significant time and resources and alleviate domain gaps caused by disparate training and test classes.

3.2. Similarity-based Segmentation

We utilize the non-parametric encoder to respectively extract support sample features, denoted as $\mathbf{F}^S \in \mathbb{R}^{M \times D}$, and query sample features, $\mathbf{F}^Q \in \mathbb{R}^{M \times D}$. We also denote the support sample labels as $\mathbf{L}^S \in \mathbb{R}^M$. Then, we conduct a simple similarity-based segmentation [32, 33]. We use masked average pooling [6] to produce the prototypes of $N+1$ classes, denoted as $\mathbf{F}^P \in \mathbb{R}^{(N+1) \times D}$. Then, the cosine similarity between normalized $\mathbf{F}^Q$ and $\mathbf{F}^P$ is,

$$\mathbf{S}_{cos} = \mathbf{F}^Q \mathbf{F}^{P\top} \in \mathbb{R}^{M \times (N+1)}, \tag{5}$$

which represents the similarity between each point in the query set and $N+1$ prototypes. Finally, we weight and integrate the one-hot labels of the prototypes, $\mathbf{L}^P \in \mathbb{R}^{(N+1) \times (N+1)}$, to achieve the final prediction,

$$\text{logits} = \varphi(\mathbf{S}_{cos}\mathbf{L}^P) \in \mathbb{R}^{M \times (N+1)}, \tag{6}$$

where $\varphi(x) = \exp(-\gamma(1-x))$ acts as an activation function and γ is a scaling factor [31, 34]. For N-way-K-shot tasks, we produce K prototypes for each category. By this segmentation head, the entire Seg-NN framework can be purely training-free, thus achieving superior efficiency.

4. Parametric Seg-PN

To further achieve better performance, we propose a parametric version, Seg-PN. Seg-PN inherits the non-parametric 3D encoder of Seg-NN to encode point clouds and only introduces a learnable lightweight segmentation head, QUEST. In contrast, the parametric version of Point-NN, Point-PN [35], inserts learnable layers into Point-NN's encoder and requires time-consuming training.

An obvious problem in few-shot learning is that the small-scale support set might fail to represent the true distribution of each category, leading to biased prototypical learning. We propose a Query-Support Transferring module, QUEST, to mitigate this problem and transfer the prototypes from the support-set to the query-set domain. QUEST directly follows the encoder and adjusts the prototypes based on query-support interaction. The detailed structure of QUEST is shown in Fig. 5. We first conduct local maximum pooling along the point dimension to obtain statistics of each feature channel in the support-set and query-set features. Then, we leverage a shared projection operation, $\mathbf{W}$, to refine the statistics of each channel. This step can be denoted as

$$\begin{aligned} \mathbf{F}^S &= \mathbf{W} \cdot \mathrm{MaxPool}(\mathbf{F}^S) \quad \in \mathbb{R}^{M' \times D}, \\ \mathbf{F}^Q &= \mathbf{W} \cdot \mathrm{MaxPool}(\mathbf{F}^Q) \quad \in \mathbb{R}^{M' \times D}, \end{aligned} \tag{7}$$

where the kernel size and stride of the pooling operation are hyperparameters. Eq. 7 produces M' statistics from the M points to manifest the distribution of each feature channel. We use the same way to obtain $\mathbf{F}^P$ as in Seg-NN. On top of this, QUEST bridges the support and query data under self-correlation and cross-correlation schemes.

Cross-correlation. We investigate the cross-correlation of the geometric structures between the support set and query set, which is utilized to adjust the category prototypes $\mathbf{F}^P$. Specifically, the cross-correlation between $\mathbf{F}^S$ and $\mathbf{F}^Q$ is calculated as their inner product,

$$\mathbf{C}_{cross} = \mathbf{F}^{Q\top}\mathbf{F}^S \quad \in \mathbb{R}^{D \times D}, \tag{8}$$

where the diagonal elements of $\mathbf{C}_{cross}$ are the correlation between the geometric structures of the support sample and query sample, while the non-diagonal elements reflect the cross-correlation between different channels of the support-set and query-set features. Next, we modulate $\mathbf{C}_{cross}$ using the softmax function and use it to adjust the corresponding channels of $\mathbf{F}^P$ by matrix multiplication,

$$\mathbf{F}^P_{cross} = \mathrm{Softmax}(\mathbf{C}_{cross}) \cdot \mathbf{F}^{P\top}. \tag{9}$$

From Eq. 9, we obtain the adjusted category prototypes via query-support cross-correlation, which learns to transfer the prototypes $\mathbf{F}^P$ to the query-set domain. Compared to the cross-attention module in [6] and the bias rectification operation in [38], our cross-correlation scheme can better capture domain bias between the support set and query set.

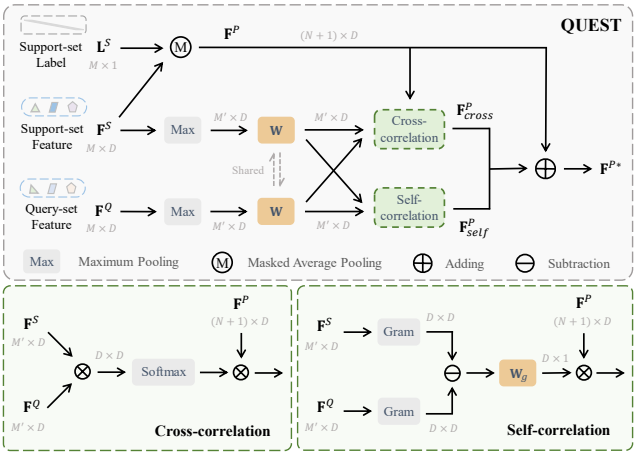


Figure 5. **Details of QUEST in Seg-PN.** QUEST finally outputs adjusted prototypes $\mathbf{F}^{P*}$.

Self-correlation. It is obvious that different feature channels extracted from our hand-crafted filters are not independent and certain correlations between channels exist. Further, this type of correlation exhibits differences between support-set and query-set domains. Therefore, we examine the difference in self-correlation between $\mathbf{F}^S$ and $\mathbf{F}^Q$, which measures the domain gap between the support set and query set. Generally, the Gram matrix of the feature vector is leveraged to represent the self-correlation [7, 10]. Denoted as $\mathbf{G}^S$ and $\mathbf{G}^Q$, the Gram matrices of the support-set and query-set features are calculated as

$$\mathbf{G}^S = \mathbf{F}^{S\top}\mathbf{F}^S, \quad \mathbf{G}^Q = \mathbf{F}^{Q\top}\mathbf{F}^Q \quad \in \mathbb{R}^{D \times D}, \tag{10}$$

both of which are symmetric matrices. The diagonal elements of the Gram matrix reflect the characteristics of each channel itself, while the non-diagonal elements reflect the dependence between different channels. In addition, the difference between $\mathbf{G}^S$ and $\mathbf{G}^Q$ measures the domain gap between support and query sets. We extract this domain gap with a linear projection, $\mathbf{W}_g \in \mathbb{R}^{D \times 1}$, and use it to rectify the channels of $\mathbf{F}^P$:

$$\mathbf{F}^P_{self} = \mathbf{F}^P \,\mathrm{diag}((\mathbf{G}^Q - \mathbf{G}^S)\mathbf{W}_g), \tag{11}$$

where $\mathrm{diag}(\cdot)$ is diagonalization and $\mathbf{F}^P_{self}$ is the rectified prototypes via self-correlation. In this way, we shift the prototype to the query-set domain.

We integrate the adjusted prototypes via self- and cross-correlation and original prototypes to obtain the final prototypes $\mathbf{F}^{P*} = \mathbf{F}^P + \mathbf{F}^P_{self} + \mathbf{F}^P_{cross}$. For N-way-K-shot problems, we average all K-shot adjusted prototypes.

After QUEST, we utilize the same similarity-matching scheme as Seg-NN to segment the query set. As the encoder is training-free, we do not need pre-training and only require episodic training to learn the QUEST module. During training, we adopt cross-entropy loss to optimize QUEST.

Method	Param.	Two-way						Three-way					
		One-shot			Five-shot			One-shot			Five-shot		
		S_0	S_1	Avg	S_0	S_1	Avg	S_0	S_1	Avg	S_0	S_1	Avg
Point-NN [35]	0.00 M	42.12	42.62	42.37	51.91	49.35	50.63	38.00	36.21	37.10	45.91	43.44	44.67
Seg-NN	0.00 M	49.45	49.60	49.53	59.40	61.48	60.44	39.06	40.10	39.58	50.14	51.33	50.74
Improvement	-	+7.33	+6.98	+7.16	+7.49	+12.13	+9.81	+1.06	+3.89	+2.48	+4.23	+7.89	+6.07
DGCNN [24]	0.62 M	36.34	38.79	37.57	56.49	56.99	56.74	30.05	32.19	31.12	46.88	47.57	47.23
ProtoNet [5]	0.27 M	48.39	49.98	49.19	57.34	63.22	60.28	40.81	45.07	42.94	49.05	53.42	51.24
MPTI [37]	0.29 M	52.27	51.48	51.88	58.93	60.56	59.75	44.27	46.92	45.60	51.74	48.57	50.16
AttMPTI [37]	0.37 M	53.77	55.94	54.86	61.67	67.02	64.35	45.18	49.27	47.23	54.92	56.79	55.86
BFG [17]	-	55.60	55.98	55.79	63.71	66.62	65.17	46.18	48.36	47.27	55.05	57.80	56.43
2CBR [38]	0.35 M	55.89	61.99	58.94	63.55	67.51	65.53	46.51	53.91	50.21	55.51	58.07	56.79
PAP3D [6]	2.45 M	59.45	66.08	62.76	65.40	70.30	67.85	48.99	56.57	52.78	61.27	60.81	61.04
Seg-PN	**0.24 M**	**64.84**	**67.98**	**66.41**	**67.63**	**71.48**	**69.36**	**60.12**	**63.22**	**61.67**	**62.58**	**64.53**	**63.56**
Improvement	-	+5.39	+1.90	+3.65	+2.23	+1.18	+1.71	+11.13	+6.65	+8.89	+1.31	+3.72	+2.52

Table 1. **Few-shot Results (%) on S3DIS.** S_i denotes the split i is used for testing, and Avg is their average mIoU. The shaded rows represent non-parametric methods. 'Param.' represents the total number of learnable parameters of each method.

5. Experiments

In this section, we first introduce the datasets and implementation details. Then we report the experimental results in comparison with existing approaches. At last, we present ablation studies to verify the effectiveness.

5.1. Experimental Details

Datasets. We validate our method using two public 3D datasets, S3DIS [2] and ScanNet [4]. Due to the large scale of original scenes, we adopt the data pre-processing in [6, 37] and partition them into small blocks. Then S3DIS and ScanNet contain 7,547 and 36,350 blocks, respectively. $M = 2048$ points are randomly sampled from each block. For each dataset, we generate a training class set C_{train} and a test class set C_{test} that have no overlap. We use C_{train} for episodic training and C_{test} for testing, performing cross-validation for each dataset. For N-way-K-shot test episodes, we iterate over all combinations of N classes from C_{test}, sampling 100 episodes for each combination.

Basic Settings. For few-shot settings, we experiment under 2/3-way-1/5-shot settings respectively, following [6, 17, 37]. For performance, we adopt the mean Intersection over Union (mIoU) as evaluation criteria. mIoU is computed by averaging the IoU scores across all unseen classes C_{test}. In Seg-NN, we set the frequency number in the initial embedding layer to $d = 20$ and sample log-linear spaced frequencies $\mathbf{u}$ with $\theta = 30$. In embedding manipulation layers, we sample frequencies $\mathbf{v}$ from a Gaussian distribution with variance 1. In the segmentation head, we set the scale factor γ to 1000. The non-parametric encoder contains totally three manipulation layers. We provide more detailed settings in supplementary materials.

5.2. Analysis

Baselines To evaluate our method, we compare it with two types of methods. First, we consider parametric 2D/3D few-shot segmentation methods, including DGCNN [20, 24], ProtoNet [5], MTPI [37], AttMPTI [37], BFG [17], 2CBR [38], and PAP3D [6]. Second, we re-implement the non-parametric model, Point-NN [35], under our settings. We report the results in Tab. 1 and 2.

Performance. For non-parametric Seg-NN, we compare it to Point-NN and observe significant improvement on the S3DIS dataset. In addition, Seg-NN even outperforms some parametric methods, such as DGCNN and ProtoNet without any training. For parametric Seg-PN, its results significantly outperform previous SOTA mIoU across all four few-shot tasks on 2 datasets. We achieve an average improvement of **+4.19%** and **+7.71%** across four tasks on the S3DIS and ScanNet datasets, respectively, demonstrating that our method can better alleviate domain gaps between seen and unseen classes. This also indicates the non-parametric encoder can extract discriminative and general knowledge for 3D shapes. From the perspective of parameter number, we only utilize 0.24M parameters, the least among existing methods and **-90%** less than PAP3D with better performance.

Efficiency. In Tab. 3, we compare the training time with existing works. Seg-NN achieves few-shot segmentation with minimal resource and time consumption as training is not required. For Seg-PN, we require only episodic training without pre-training, greatly reducing training time by more than **-90%** compared to existing methods. Both Seg-NN and Seg-PN efficiently simplify the few-shot pipeline by discarding the pre-training step.

Method	Param.	Two-way						Three-way					
		One-shot			Five-shot			One-shot			Five-shot		
		S_0	S_1	Avg	S_0	S_1	Avg	S_0	S_1	Avg	S_0	S_1	Avg
Point-NN [35]	0.00 M	28.85	31.56	30.21	34.82	32.87	33.85	21.24	17.91	19.58	26.42	23.98	25.20
Seg-NN	0.00 M	36.80	38.59	38.96	43.97	41.50	44.79	27.41	23.36	28.29	34.27	30.75	33.77
Improvement	-	+7.95	+7.03	+8.75	+9.15	+8.63	+10.94	+6.17	+5.45	+8.71	+7.85	+6.77	+8.57
DGCNN [24]	1.43 M	31.55	28.94	30.25	42.71	37.24	39.98	23.99	19.10	21.55	34.93	28.10	31.52
ProtoNet [5]	0.27 M	33.92	30.95	32.44	45.34	42.01	43.68	28.47	26.13	27.30	37.36	34.98	36.17
MPTI [37]	0.29 M	39.27	36.14	37.71	46.90	43.59	45.25	29.96	27.26	28.61	38.14	34.36	36.25
AttMPTI [37]	0.37 M	42.55	40.83	41.69	54.00	50.32	52.16	35.23	30.72	32.98	46.74	40.80	43.77
BFG [17]	-	42.15	40.52	41.34	51.23	49.39	50.31	34.12	31.98	33.05	46.25	41.38	43.82
2CBR [38]	0.35 M	50.73	47.66	49.20	52.35	47.14	49.75	47.00	46.36	46.68	45.06	39.47	42.27
PAP3D [6]	2.45 M	57.08	55.94	56.51	64.55	59.64	62.10	55.27	55.60	55.44	59.02	53.16	56.09
Seg-PN	**0.24 M**	**63.15**	**64.32**	**63.74**	**67.08**	**69.05**	**68.07**	**61.80**	**65.34**	**63.57**	**62.94**	**68.26**	**65.60**
Improvement	-	+6.07	+8.38	+7.23	+2.53	+9.41	+5.97	+6.53	+9.74	+8.13	+3.92	+15.10	+9.51

Table 2. **Few-shot Results (%) on ScanNet.** S_i denotes the split i is used for testing, and Avg is their average mIoU. The shaded rows represent non-parametric methods. 'Param.' represents the total number of learnable parameters of each method.

Method	mIoU	Param.	Pre-train Time	Episodic Train	Total Time
DGCNN [24]	36.34	0.62 M	4.0 h	0.8 h	4.8 h
AttMPTI [37]	53.77	0.37 M	4.0 h	5.5 h	9.5 h
2CBR [38]	55.89	0.35 M	6.0 h	0.2 h	6.2 h
PAP3D [6]	59.45	2.45 M	3.6 h	1.1 h	4.7 h
Seg-NN	49.45	**0.00 M**	**0.0 h**	**0.0 h**	**0.0 h**
Seg-PN	**64.84**	0.24 M	**0.0 h**	0.5 h	0.5 h

Table 3. **Performance (%) and Efficiency Comparison on S3DIS**. Train Time and Test Speed (episodes/second) are tested on one NVIDIA A6000 GPU. We calculate the total time of pre-training and episodic training. We report the accuracy under 2-way-1-shot settings on S_0 split.

Backbone	Projection	QUEST	mIoU	Parameters
Point-NN [35]	✓	✗	46.31	1.20 M
	✗	✓	55.28	1.28 M
Point-PN [35]	✗	✓	54.74	3.97 M
DGCNN [24]	✗	✓	55.86	0.42 M
Seg-NN	✓	✗	50.06	0.20 M
	✗	✓	**64.84**	0.24 M

Table 4. **Effect of Backbones and the QUEST Module on Seg-PN.** We combine different backbones with the QUEST module to demonstrate the efficacy of the proposed non-parametric encoder. We also substitute the QUEST module with a simple linear projection layer ('Projection') to verify its effect.

Layers	1	2	3	4	5
Seg-NN	36.43	42.38	**49.45**	46.19	43.18
Seg-PN	58.35	62.23	64.84	**65.92**	63.98

Table 5. **Ablation for Number of Layers in Seg-NN Encoder.**

5.3. Ablation Study

We conduct extensive ablation experiments to reveal the roles of different designs. By default, we perform 2-way-1-shot experiments on the S_0 split of S3DIS dataset.

Ablation for Seg-NN. We first investigate the effect of different numbers of manipulation layers in Tab. 5. We observe that Seg-NN and Seg-PN achieve their best performance with three and four layers respectively, indicating that Seg-PN prefers to learn deeper features. To demonstrate the efficacy of this hierarchical structure, we visualize the feature similarity map of each layer in Fig. 6. The similarity map suggests our hierarchical design can gradually capture discriminative point-level features and the integration of multi-layer features can effectively model local shape characteristics.

Ablation on Seg-PN. First, we investigate the role of the backbone and the QUEST module in Seg-PN in Tab. 4. By substituting our non-parametric encoder with other backbones, including Point-NN and pre-trained DGCNN and Point-PN (following [6]), we observe a performance drop, which highlights the effectiveness of our encoder in capturing 3D geometries. In addition, replacing the QUEST module with a projection layer also significantly impairs the performance. Tab. 4 proves that combining our non-parametric encoder with the QUEST module can remarkably improve prediction. Then, we examine different designs of the QUEST module in Tab. 6, including the roles of the self- and cross-correlation. We observe that using only self-correlation achieves lower performance than using cross-correlation, which indicates that the interaction between support and query sets is more important than only considering their self-correlation. Finally, utilizing both correlations allows for a higher performance.

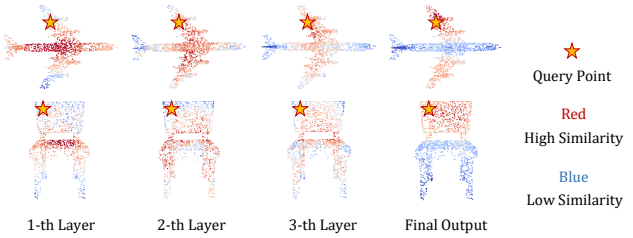


Figure 6. **Efficacy of the Hierarchical Structure.** We visualize the feature similarity between the "query point" and other points.

QUEST	Two Way			Three Way		
	S_0	S_1	Avg	S_0	S_1	Avg
w/o	47.32	50.05	48.68	38.46	40.66	39.56
Self	49.05	53.47	51.26	39.99	42.86	41.42
Cross	62.72	67.16	64.94	59.73	61.49	60.61
QUEST	**64.84**	**67.98**	**66.41**	**60.12**	**63.22**	**61.67**

Table 6. **Ablation for the QUEST Module** in Seg-PN. We report the results (%) under 2/3-way-1-shot settings. 'Self' and 'Cross' represent self- and cross-correlation, respectively.

6. Additional Tasks

To exhibit our generalization ability, we validate the Seg-NN encoder on other 3D tasks, including supervised classification, few-shot classification, and part segmentation.

6.1. Training-free Classification

We employ the Seg-NN encoder with minor modifications to encode each point cloud into a discriminative representation. Specifically, we discard the upsampling layers in the Seg-NN encoder and apply global maximum pooling to the embeddings obtained from the 3-th manipulation layer. This results in a global representation for each point cloud. To classify each point cloud, we refer to Point-NN [35] to adopt a similarity-matching scheme.

Performance We adopt ModelNet40 [27] and ScanObjectNN [22] datasets to evaluate the classification performance. We compare Seg-NN with Point-NN [35] in Tab. 7. From the table, our method sets the new state-of-the-art for training-free classification on ModelNet40 and achieves comparable results on ScanObjectNN. In terms of inference speed, our method outperforms Point-NN by around +50%.

6.2. Training-free Few-shot Classification

We follow [21] to use ModelNet40 dataset [27]. We conduct the N-way-K-shot classification tasks, $N \in \{5, 10\}$ and $K \in \{10, 20\}$. We randomly sample N classes and K point clouds for each class as the support set. For the query set, we pick 20 unseen objects from each of the N classes. We randomly repeat 10 times and report the average results.

Performance From Tab. 8, we observe that Seg-NN achieves the highest performance and significantly outperforms all existing parametric models, e.g., DGCNN [24] and PointCNN [14].

Method	MN40	SONN	Speed	SNPart	Speed
Point-NN [35]	81.8	**64.9**	209	70.4	52
Seg-NN	**84.2**	64.4	**306**	**73.2**	**88**

Table 7. **Training-free Performance (%) and Efficiency**. Inference speed (samples/second) is tested on one NVIDIA A6000 GPU. We report the classification accuracy (%) on ModelNet40 ('MN40') [27] and ScanObjectNN ('SONN') [22] datasets, and mIoU (%) of segmentation on ShapeNetPart ('SNPart') [29].

Method	5-way		10-way	
	10-shot	20-shot	10-shot	20-shot
DGCNN [24]	31.6	40.8	19.9	16.9
PointNet++ [18]	38.5	42.4	23.0	18.8
3D-GAN [26]	55.8	65.8	40.3	48.4
PointCNN [14]	65.4	68.6	46.6	50.0
Sharma et. al [21]	63.2	68.9	49.1	50.1
Point-NN [35]	88.8	90.9	79.9	84.9
Seg-NN	**89.7**	**91.0**	**81.7**	**86.1**

Table 8. **Few-shot Classification Results** on ModelNet40 [27].

6.3. Training-free Part Segmentation.

We extend our non-parametric 3D encoder to the part segmentation task. In detail, we utilize the Seg-NN encoder to extract prototypes for each part category and employ the same similarity-based method to segment 3D objects into pre-defined part classes.

Performance ShapeNetPart [29] is a commonly used dataset for object-level part segmentation. From Tab. 7, our method outperforms Point-NN by +2.8% in segmentation performance and +60% in inference speed.

7. Conclusion

We propose a non-parametric framework, Seg-NN, and a parametric variant, Seg-PN, for few-shot point cloud semantic segmentation. Based on trigonometric PEs, Seg-NN introduces no learnable parameters and discards any training. Seg-NN achieves comparable performance with existing parametric methods and only causes minimal resource consumption. Furthermore, Seg-PN introduces the QUEST module to overcome prototype bias and achieves remarkable improvement compared to existing approaches. Importantly, Seg-PN does not require pre-training, thus simplifying the pipeline of 3D few-shot segmentation and saving significant time. For future work, we will explore the application of non-parametric encoders to more 3D tasks and improve the performance and generalization ability.

References

[1] Syeda Mariam Ahmed, Yan Zhi Tan, Chee Meng Chew, Abdullah Al Mamun, and Fook Seng Wong. Edge and corner detection for unorganized 3d point clouds with application to robotic welding. In *IEEE International Conference on Intelligent Robots and Systems*, pages 7350–7355, 2018. 1

[2] 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 *IEEE Conference on Computer Vision and Pattern Recognition*, pages 1534–1543, 2016. 2, 6

[3] Saifullahi Aminu Bello, Shangshu Yu, Cheng Wang, Jibril Muhmmad Adam, and Jonathan Li. Deep learning on 3d point clouds. *Remote Sensing*, 12(11):1729, 2020. 1

[4] Angela Dai, Angel X Chang, Manolis Savva, Maciej Halber, Thomas Funkhouser, and Matthias Nießner. Scannet: Richly-annotated 3d reconstructions of indoor scenes. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 5828–5839, 2017. 2, 6

[5] Victor Garcia and Joan Bruna. Few-shot learning with graph neural networks. *arXiv preprint arXiv:1711.04043*, 2017. 6, 7

[6] Shuting He, Xudong Jiang, Wei Jiang, and Henghui Ding. Prototype adaption and projection for few-and zero-shot 3d point cloud semantic segmentation. *IEEE Transactions on Image Processing*, 2023. 1, 2, 4, 5, 6, 7

[7] Yongcheng Jing, Yezhou Yang, Zunlei Feng, Jingwen Ye, Yizhou Yu, and Mingli Song. Neural style transfer: A review. *IEEE Transactions on Visualization and Computer Graphics*, 26(11):3365–3385, 2019. 5

[8] Georg Krispel, Michael Opitz, Georg Waltner, Horst Possegger, and Horst Bischof. Fuseseg: Lidar point cloud segmentation fusing multi-modal data. In *IEEE Winter Conference on Applications of Computer Vision*, pages 1874–1883, 2020. 1

[9] Xin Lai, Jianhui Liu, Li Jiang, Liwei Wang, Hengshuang Zhao, Shu Liu, Xiaojuan Qi, and Jiaya Jia. Stratified transformer for 3d point cloud segmentation. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 8500–8509, 2022. 1

[10] Chunbo Lang, Gong Cheng, Binfei Tu, and Junwei Han. Learning what not to segment: A new perspective on few-shot segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 8057–8067, 2022. 5

[11] Benjamin Lewandowski, Jonathan Liebner, Tim Wengefeld, Steffen Müller, and Horst-Michael Gross. Fast and robust 3d person detector and posture estimator for mobile robotic applications. In *International Conference on Robotics and Automation*, pages 4869–4875, 2019. 1

[12] Xinke Li, Henghui Ding, Zekun Tong, Yuwei Wu, and Yeow Meng Chee. Primitive3d: 3d object dataset synthesis from randomly assembled primitives. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 15947–15957, 2022. 1

[13] Xiangtai Li, Henghui Ding, Wenwei Zhang, Haobo Yuan, Jiangmiao Pang, Guangliang Cheng, Kai Chen, Ziwei Liu, and Chen Change Loy. Transformer-based visual segmentation: A survey. *arXiv preprint arXiv:2304.09854*, 2023. 1

[14] Yangyan Li, Rui Bu, Mingchao Sun, Wei Wu, Xinhan Di, and Baoquan Chen. Pointcnn: Convolution on x-transformed points. *Advances in Neural Information Processing Systems*, 31, 2018. 8

[15] Tao Luo, Zheng Ma, Zhi-Qin John Xu, and Yaoyu Zhang. Theory of the frequency principle for general deep neural networks. *arXiv preprint arXiv:1906.09235*, 2019. 4

[16] Xu Ma, Can Qin, Haoxuan You, Haoxi Ran, and Yun Fu. Rethinking network design and local geometry in point cloud: A simple residual mlp framework. *arXiv preprint arXiv:2202.07123*, 2022. 4

[17] Yongqiang Mao, Zonghao Guo, LU Xiaonan, Zhiqiang Yuan, and Haowen Guo. Bidirectional feature globalization for few-shot semantic segmentation of 3d point cloud scenes. In *International Conference on 3D Vision*, pages 505–514, 2022. 6, 7

[18] Charles Ruizhongtai Qi, Li Yi, Hao Su, and Leonidas J Guibas. Pointnet++: Deep hierarchical feature learning on point sets in a metric space. *Advances in Neural Information Processing Systems*, 30, 2017. 4, 8

[19] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In *Medical Image Computing and Computer-Assisted Intervention*, pages 234–241, 2015. 3

[20] Amirreza Shaban, Shray Bansal, Zhen Liu, Irfan Essa, and Byron Boots. One-shot learning for semantic segmentation. *arXiv preprint arXiv:1709.03410*, 2017. 6

[21] Charu Sharma and Manohar Kaul. Self-supervised few-shot learning on point clouds. *Advances in Neural Information Processing Systems*, 33:7212–7221, 2020. 8

[22] Mikaela Angelina Uy, Quang-Hieu Pham, Binh-Son Hua, Thanh Nguyen, and Sai-Kit Yeung. Revisiting point cloud classification: A new benchmark dataset and classification model on real-world data. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1588–1597, 2019. 8

[23] Oriol Vinyals, Charles Blundell, Tim Lillicrap, Daan Wierstra, et al. Matching networks for one shot learning. In *Advances in Neural Information Processing Systems*, pages 3630–3638, 2016. 2

[24] Yue Wang, Yongbin Sun, Ziwei Liu, Sanjay E Sarma, Michael M Bronstein, and Justin M Solomon. Dynamic graph cnn for learning on point clouds. *ACM Transactions on Graphics*, 38(5):1–12, 2019. 1, 2, 3, 4, 6, 7, 8

[25] Yue Wang, Alireza Fathi, Abhijit Kundu, David A Ross, Caroline Pantofaru, Tom Funkhouser, and Justin Solomon. Pillar-based object detection for autonomous driving. In *European Conference on Computer Vision*, pages 18–34, 2020. 1

[26] Jiajun Wu, Chengkai Zhang, Tianfan Xue, Bill Freeman, and Josh Tenenbaum. Learning a probabilistic latent space of object shapes via 3d generative-adversarial modeling. *Advances in Neural Information Processing Systems*, 29, 2016. 8

[27] Zhirong Wu, Shuran Song, Aditya Khosla, Fisher Yu, Linguang Zhang, Xiaoou Tang, and Jianxiong Xiao. 3d shapenets: A deep representation for volumetric shapes. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1912–1920, 2015. 8

[28] Yu-Qi Yang, Yu-Xiao Guo, Jian-Yu Xiong, Yang Liu, Hao Pan, Peng-Shuai Wang, Xin Tong, and Baining Guo. Swin3d: A pretrained transformer backbone for 3d indoor scene understanding. *arXiv preprint arXiv:2304.06906*, 2023. 1

[29] Li Yi, Vladimir G Kim, Duygu Ceylan, I-Chao Shen, Mengyan Yan, Hao Su, Cewu Lu, Qixing Huang, Alla Sheffer, and Leonidas Guibas. A scalable active framework for region annotation in 3d shape collections. *ACM Transactions on Graphics*, 35(6):1–12, 2016. 8

[30] Xiangyu Yue, Bichen Wu, Sanjit A Seshia, Kurt Keutzer, and Alberto L Sangiovanni-Vincentelli. A lidar point cloud generator: from a virtual world to autonomous driving. In *ACM on International Conference on Multimedia Retrieval*, pages 458–464, 2018. 1

[31] Renrui Zhang, Rongyao Fang, Peng Gao, Wei Zhang, Kunchang Li, Jifeng Dai, Yu Qiao, and Hongsheng Li. Tip-adapter: Training-free clip-adapter for better vision-language modeling. *arXiv preprint arXiv:2111.03930*, 2021. 4

[32] Renrui Zhang, Ziyu Guo, Peng Gao, Rongyao Fang, Bin Zhao, Dong Wang, Yu Qiao, and Hongsheng Li. Point-m2ae: Multi-scale masked autoencoders for hierarchical point cloud pre-training. *NeurIPS 2022*, 2022. 4

[33] Renrui Zhang, Liuhui Wang, Ziyu Guo, and Jianbo Shi. Nearest neighbors meet deep neural networks for point cloud analysis. In *WACV 2023*, 2022. 4

[34] Renrui Zhang, Xiangfei Hu, Bohao Li, Siyuan Huang, Hanqiu Deng, Hongsheng Li, Yu Qiao, and Peng Gao. Prompt, generate, then cache: Cascade of foundation models makes strong few-shot learners. *CVPR 2023*, 2023. 4

[35] Renrui Zhang, Liuhui Wang, Yali Wang, Peng Gao, Hongsheng Li, and Jianbo Shi. Parameter is not all you need: Starting from non-parametric networks for 3d point cloud analysis. *arXiv preprint arXiv:2303.08134*, 2023. 2, 3, 4, 5, 6, 7, 8

[36] Yaoyu Zhang, Zhi-Qin John Xu, Tao Luo, and Zheng Ma. Explicitizing an implicit bias of the frequency principle in two-layer neural networks. *arXiv preprint arXiv:1905.10264*, 2019. 4

[37] Na Zhao, Tat-Seng Chua, and Gim Hee Lee. Few-shot 3d point cloud semantic segmentation. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 8873–8882, 2021. 2, 6, 7

[38] Guanyu Zhu, Yong Zhou, Rui Yao, and Hancheng Zhu. Cross-class bias rectification for point cloud few-shot segmentation. *IEEE Transactions on Multimedia*, 2023. 5, 6, 7