Nearest Neighbors Meet Deep Neural Networks for Point Cloud Analysis

Renrui Zhang[1,2], Liuhui Wang[1,2], Ziyu Guo[1], Jianbo Shi[2,3]
[1]Peking University, [2]Heisenberg Robotics, [3]University of Pennsylvania
{1700012927, 1900012932}@pku.edu.cn, jshi@seas.upenn.edu

Abstract

*Performances on standard 3D point cloud benchmarks have plateaued, resulting in oversized models and complex network design to make a fractional improvement. We present an alternative to enhance existing deep neural networks without any redesigning or extra parameters, termed as **S**patial-**N**eighbor Adapter (**SN-Adapter**). Building on any trained 3D network, we utilize its learned encoding capability to extract features of the training dataset and summarize them as prototypical spatial knowledge. For a test point cloud, the SN-Adapter retrieves k nearest neighbors (k-NN) from the pre-constructed spatial prototypes and linearly interpolates the k-NN prediction with that of the original 3D network. By providing complementary characteristics, the proposed SN-Adapter serves as a plug-and-play module to economically improve performance in a non-parametric manner. More importantly, our SN-Adapter can be effectively generalized to various 3D tasks, including shape classification, part segmentation, and 3D object detection, demonstrating its superiority and robustness. We hope our approach could show a new perspective for point cloud analysis and facilitate future research.*

1. Introduction

3D vision has wide usage in robotics and AI. Many methods have been proposed to tackle 3D tasks, including object recognition [25, 26, 13, 2, 42, 47] and scene-level understanding [1, 35, 4, 48, 21, 10]. Existing 3D methods are built upon the learnable deep neural networks and benefit from their abilities to process the irregular point clouds. Starting from the concise PointNet [25], later researches upgrade it with hierarchical architectures [26, 2], point-based convolutions [20, 31, 43], attention mechanisms [15], and so on [39, 42].

Recent works focused on inserting complicated modules or excessively increasing network parameters to boost benchmark scores. This trend has not only harmed the efficiency during training and inference, but gradually saturated the benchmarks. As examples for shape classification on ModelNet40 [40], CurveNet [42] delicately explores a set of spatial curves for aggregating local geometry, which leads to 10× slower training and 20× slower inference compared to PointNet++ [26]. PointMLP [2] brings +11.9M parameters for only +0.5% accuracy boost, which increases 19× more model scales than its elite version [2]. Therefore, we ask the question: *could we boost the performance of existing 3D networks at the least cost, even without additional parameters or re-training?*

We develop a non-parametric adapter module by retrieving 3D prototypical knowledge from the spatial neighbors, named **SN-Adapter**. It refers to the idea of k Nearest Neighbors algorithm (k-NN) and can directly enhance the existing trained 3D deep neural networks without extra re-training. As shown in Figure 1, our SN-Adapter is implemented in two steps: pre-construction of 3D prototypical knowledge and inference-time enhancement by interpolation. Specifically, we theoretically split a trained network into two parts. The first is the feature extractor that encodes an input raw point cloud into high-dimensional representations. The second, usually the last linear layer of the network, is named the 3D classifier, which categorizes the encoded vectors with classification logits. Using a trained extractor, we first obtain all the high-dimensional features of point clouds from the training dataset. For different 3D tasks, we summarize the features as various forms of prototypical spatial knowledge, e.g., sample-wise, part-wise, and object-wise prototypes in the top-right of Figure 1. During inference, the SN-Adapter is appended to the feature extractor and utilizes k-NN to retrieve 3D knowledge from the pre-constructed prototypes. Finally, we linearly interpolate the classification logits concurrently produced from the SN-Adapter and the trained 3D classifier, by which the original 3D network can be improved with marginal extra costs.

Through experimental analysis, we observe the enhancement is resulted from the complementary characteristics between the trained 3D classifier and our SN-Adapter: the former is learned to fit the training set but the latter reveals the feature-level similarities among 3D prototypes. By ex-

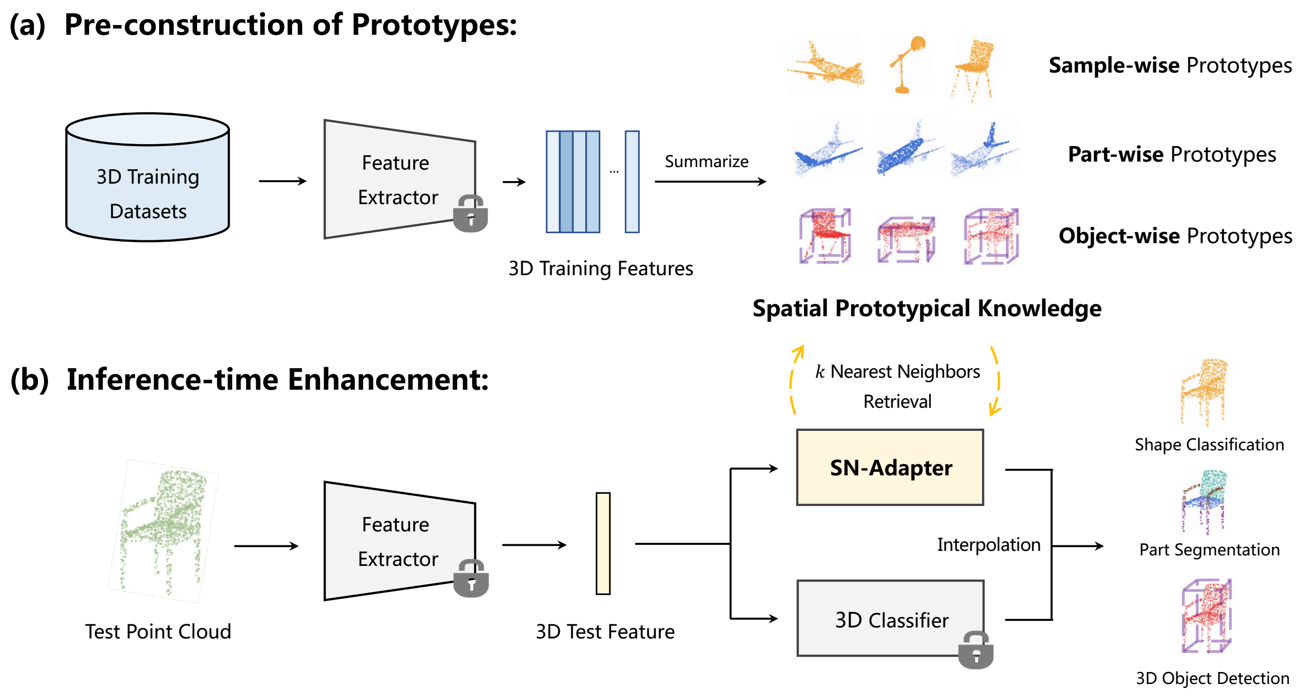

Figure 1. **The Pipeline of SN-Adapter by Two Steps: (a) and (b).** We divide the already trained deep neural network into the feature extractor and the 3D classifier, whose weights are frozen without fine-tuning. In **a**), we extract the 3D training features and construct the spatial prototypical knowledge. In **b**), we introduce SN-Adapter to conduct k-NN retrieval from the task-specific prototypes for non-parametric enhancement.

tensive experiments, our SN-Adapter is verified to widely improve the performance of existing methods on different 3D tasks, such as +1.34% classification accuracy on ModelNet40 [40], +0.17% segmentation mIoU on ShapeNet-Part [45], and +7.34% detection AR on ScanNetV2 [7].

Our main contributions are summarized as follows:

1. We propose SN-Adapter, a plug-and-play module to assist 3D deep neural networks via k-NN for better point cloud analysis.
2. By retrieving knowledge from the pre-constructed spatial prototypes, SN-Adapter efficiently improves the already trained models without any parameters or re-training.
3. We conduct complete experiments on various 3D benchmarks to demonstrate the effectiveness and robustness of our approach.

2. Related Work

Deep Learning for 3D Point Clouds. Point cloud based shape classification of synthetic data [40] and real-world data [33] have been widely studied by PointNet [25], PointNet++ [26] and so on [42, 39, 2, 20, 13]. Part segmentation [45] and scene segmentation [7, 29] ask for the per-point classification, the methods [21, 10, 35, 4, 48] of which normally extend feature decoders upon the classification networks to densely propagate the extracted features. 3D object detection has wide usages in *e.g.*, autonomous driving [5, 23, 19] and robotics [28, 6, 22]. Our SN-Adapter can be generalized to all 3D tasks, including shape classification, part segmentation, and 3D object detection, demonstrating our robustness for point cloud analysis.

Feature Adapters in Computer Vision. The feature adapter is a light-weight module to efficiently adapt large-scale pre-trained models for downstream tasks. Motivated by the adapter in NLP [16], CLIP-Adapter [12] and Tip-Adapter [46] introduce visual adapters using CLIP for few-shot image classification: freeze the pre-trained parameters of CLIP and only fine-tune the adapters of two-layer MLP. Follow-up works have successfully applied adapters to tasks such as image captioning [30], multi-view understanding [47], object detection [11], semantic segmentation [27], and video analysis [37]. Compared to previous works, our SN-Adapter is efficient, non-parametric, and aims at the tasks for 3D point clouds. We leverage the idea of k nearest neighbors to enhance the trained 3D networks without re-training.

Nearest-Neighbor Algorithm. Nearest-Neighbor Algorithm memorizes the training data and predicts labels based on the k nearest training samples (k-NN). Comparing with neural networks k-NN is still favored for its simplicity and efficiency. Models based on nearest-neighbor retrieval are able to provide strong baselines for many tasks, such as image captioning [8, 9], image restoration [24], few-shot learning [38], and representation learning [3, 36]. Besides computer vision, Nearest-Neighbor Algorithm also plays an important role for some language tasks, *e.g.*, language modeling [14, 18] and machine translation [17, 32]. Different from the above domains, for the first time, we explore how to augment existing deep neural networks with Nearest-Neighbor Algorithm for 3D point cloud analysis and propose an SN-Adapter with spatial prototypical knowledge retrieval.

3. Method

In this section, we respectively illustrate how our proposed Spatial-Neighbor Adapter (SN-Adapter) benefits the three 3D tasks: shape classification, part segmentation, and 3D object detection.

3.1. Shape Classification

Task Description. Given a trained 3D network for classification, we theoretically divide it into two parts: the feature extractor $\Phi(\cdot)$ and the 3D classifier $\Theta(\cdot)$. The feature extractor takes as input a raw point cloud $\{p_i\}_{i=1}^N$ of N points and outputs its C-dimensional global feature $f \in \mathbb{R}^C$. The 3D classifier then maps f into classification logits of K categories, $l^{cls} \in \mathbb{R}^K$, which denote the predicted probability for each category. We formulate them as

$$l^{cls} = \Theta(f); \quad f = \Phi(\{p_i\}_{i=1}^N). \tag{1}$$

Normally, $\Phi(\cdot)$ is invariant to the permutation of points with a pooling operation to capture the global characters, and $\Theta(\cdot)$ corresponds to the last linear projection layer of the network.

Sample-wise Spatial Prototypes. For shape classification, we construct the sample-wise spatial prototypes to retrieve 3D knowledge for each test point cloud. First, we utilize the trained feature extractor $\Phi(\cdot)$ to obtain the global features of all M samples from the training set, denoted as $F^{cls} \in \mathbb{R}^{M\times C}$. As each training sample is only represented by a single global vector, we are affordable to store all M features F^{cls} as the spatial prototypes for reserving complete prior 3D knowledge, denoted as $Proto^{cls} \in \mathbb{R}^{M\times C}$. To further explore the spatial distributions of different point clouds, we also obtain a global positional vector for each training sample by averaging the 3D positional encodings [34] of all input points, which are directly added to $Proto^{cls}$. Then during inference, we extract the global feature f of the test point cloud, and linearly interpolate the two classification logits predicted by the 3D classifier and our SN-Adapter, formulated as

$$l^{cls} = \Theta(f) + \gamma \text{SN-Adapter}(f,\ Proto^{cls}), \tag{2}$$

where γ denote the relative weights between two logits.

SN-Adapter. Analogous to all 3D tasks, SN-Adapter conducts k-NN algorithm to aggregate k-nearest spatial knowledge and adopts Euclidean distance as the distance metric between f and $Proto^{cls}$. We represent the retrieved k-nearest prototypes as $\mathcal{N}$ and the category set as $\mathcal{C}$. Then, the predicted probability of category $c \in \mathcal{C}$ in the logits is calculated as

$$Prob(c|f) = \frac{\sum_{pt\in\mathcal{N}_c} 1/d(f,pt)}{\sum_{c\in C}\sum_{pt\in\mathcal{N}_c} 1/d(f,pt)}, \tag{3}$$

where $\mathcal{N}_c$ denotes the retrieved prototypes of the c category, and $d(f, pt)$ denotes the distance between test point cloud's feature f and the prototype pt.

3.2. Part Segmentation

Task Description. Part segmentation task requires the network to classify each point in the input point cloud. The $\Phi(\cdot)$ is developed as an encoder-decoder architecture and outputs the extracted features $\{f_i\}_{i=1}^N$ for all the N points. We formulate this as

$$\{l_i^{seg}\}_{i=1}^N = \{\Theta(f_i)\}_{i=1}^N; \quad \{f_i\}_{i=1}^N = \Phi(\{p_i\}_{i=1}^N), \tag{4}$$

where $l_i^{seg} \in \mathbb{R}^K$ denotes the classification logits of the i-th point. Here, $\Theta(\cdot)$ is shared for every point and maps the point feature into logits of K part categories.

Part-wise Spatial Prototypes. We construct the part-wise spatial prototypes to retrieve 3D knowledge for every single point of the test point cloud. Considering the classification logits are to be made for each point, we need to extract and memorize the features F of all N points from M training samples as prototypical knowledge. However, it would be overloaded to store the $F^{seg} \in \mathbb{R}^{M\times N\times C}$, let alone the k-NN retrieval. Therefore, for each training sample, we propose to obtain its part-wise prototypical features by conducting average pooling on the points of the same part category, denoted as Part_Pooling$(\cdot)$. For example, a point cloud of a chair is annotated as three parts: leg, seat, and back. Then, we only need to store three prototypical features for this training sample, whose dimension is $\mathbb{R}^{3\times C}$. After the pre-construction, we acquire the spatial prototypical knowledge for part segmentation, $Proto^{seg}$, which is space-efficient and still in the same order as $Proto^{cls}$, formulated as

$$Proto^{seg} = \text{Part\_Pooling}(F^{seg}) \in \mathbb{R}^{M\times P\times C}, \quad (5)$$

where P is the maximum part category number of an object in the dataset, which is no more than six in ShapeNet-Part [45]. During inference, after extracting the features $\{f_i\}_{i=1}^N$, we combine the two classification logits for each N point of the test point cloud, formulated as

$$\begin{aligned}\{l_i^{seg}\}_{i=1}^N = \{&\Theta(f_i) \\ &+ \gamma \text{SN-Adapter}(f_i,\ Proto^{seg})\}_{i=1}^N.\end{aligned} \quad (6)$$

3.3. 3D Object Detection

Task Description. Taking a scene-level point cloud as input, the 3D object detector learns to localize and classify the objects in the 3D space. The detector would first utilize the $\Phi(\cdot)$ to extract the scene-level 3D features and group the features for each object proposal, denoted as $\{f_i\}_{i=1}^O$, where $f_i \in \mathbb{R}^C$ and O represents the proposed object number of the scene. Then, several parallel MLP-based heads are adopted to predict the category, 3D position and other attributes for each object proposal. We formulate the main process as

$$\begin{aligned}\{l_i^{det}\}_{i=1}^O &= \{\Theta_{cls}(f_i)\}_{i=1}^O; \\ \{p_i^{det}\}_{i=1}^O &= \{\Theta_{pos}(f_i)\}_{i=1}^O; \\ \{f_i\}_{i=1}^O &= \Phi(\{p_i\}_{i=1}^N),\end{aligned} \quad (7)$$

where $\Theta_{cls}(\cdot)$ and $\Theta_{pos}(\cdot)$ are responsible for predicting the classification logits $l_i^{det} \in \mathbb{R}^K$ and the 3D position $p_i^{det} \in \mathbb{R}^3$, which are shared for all object proposals. After this, the Non-Maximum Suppression (3D NMS) is applied to discard the duplicated predictions in the 3D space, which is significant to the final evaluation metric.

Object-wise Spatial Prototypes. We construct the object-wise spatial prototypes to retrieve 3D knowledge for each object proposal in the test point cloud. We first leverage the trained 3D detector to obtain the extracted object features and predicted 3D positions for all training samples, denoted as $F^{det}, P^{det} \in \mathbb{R}^{M\times O\times 3}$. On top of that, we adopt positional encodings [34] based on trigonometric functions to embed P^{det} and add them onto F^{det}. This provides F^{det} with sufficient 3D positional information of objects and facilitates the k-NN retrieval for SN-Adapter. We then calculate the spatial prototypical knowledge for 3D object detection as

$$Proto^{det} = F^{det} + \text{PE}(P^{det}) \in \mathbb{R}^{M\times O\times C}, \quad (8)$$

where PE denotes the positional encodings function.

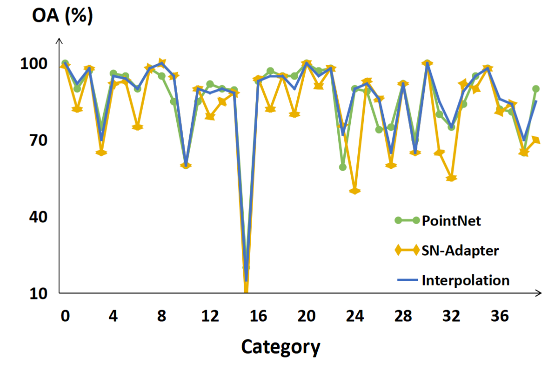

Figure 2. **Comparison of individual PointNet, SN-Adapter, and the interpolated model for different categories.** We show the the overall accuracy (OA) of 40 categories on ModelNet40 [40].

PointNet	SN-Adapter	Interpolation	Number
✓	✗	✓	72
✓	✗	✗	36
✗	✓	✓	68
✗	✓	✗	10
✗	✗	✓	1

Table 1. **Statistic of sample numbers where individual models produce different predictions.** ✓ and ✗ denote correct and wrong predictions, respectively.

During inference, for each object proposal we acquire its predicted $f_i, l_i^{det}, p_i^{det}$ and aggregate them likewise via positional encodings. The SN-Adapter retrieves spatial knowledge from nearest neighbors and enhances the classification logits predicted by Θ_{cls}, formulated as

$$\begin{aligned}\{l_i^{det}\}_{i=1}^O = \{&\Theta_{cls}(f_i) \\ &+ \gamma \text{SN-Adapter}(f_i + \text{PE}(p_i^{det}),\ Proto^{det})\}_{i=1}^O.\end{aligned} \quad (9)$$

Our SN-Adapter is inserted before the 3D NMS operation, which could rectify some 'false' classification made by Θ_{cls} and effectively avoid the removal of 'true' bounding boxes.

4. Analysis

4.1. Quantative Analysis

Here, we take PointNet [25] for shape classification on ModelNet40 [40] as the example. First, we show the performance of individual PointNet's 3D classifier and SN-Adapter compared to the interpolated one in Figure 2: the interpolated model achieves higher accuracy for most categories. Though SN-Adapter performs much worse than the learned 3D classifier on some categories, it could reversely enhance to achieve a better 3D classifier by interpolation. Specifically, we present the statistic for the interpolated prediction whose individual predictions of 3D classifier and SN-adapter are inconsistent. As shown in Table 1, when

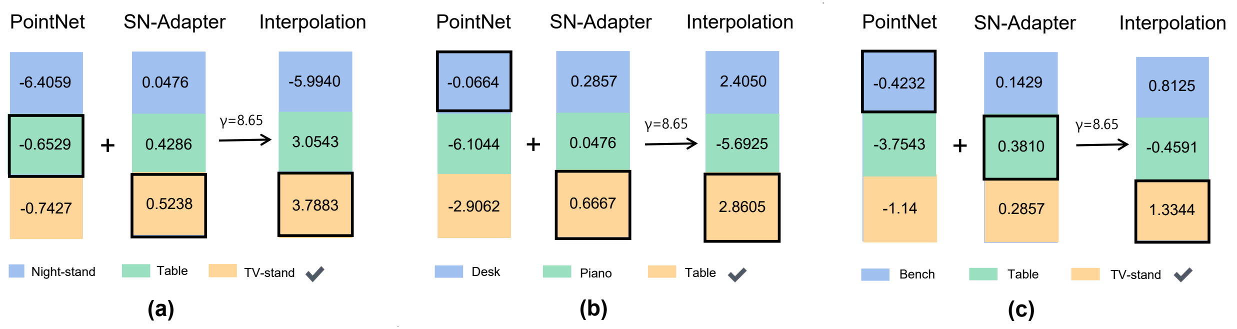

Figure 3. **Classification logits of PointNet, SN-Adapter and the interpolated model**. We report the numerical results of logits before the softmax function and denote different categories with different colors. We highlight the category of the highest value in the logits with a box and the ground-truth category with a check mark.

the original PointNet is wrong, but SN-Adapter is correct, our SN-Adapter can help rectify nearly 90%, 68/(68+10), of the predictions. More surprisingly, we observe that, even if both PointNet and SN-Adapter are wrong, the interpolated one still obtains the correct results, demonstrating the implicit complementary knowledge between the learned 3D classifier and the spatial prototypes.

To further illustrate the complementarity of SN-Adapter, we present the predicted classification logits before softmax function, for the cases where SN-Adapter corrects the false prediction of PointNet. As shown in Figure 3 (a), the PointNet's predicted values of 'night-stand' and 'table' are close, indicating it is difficult for PointNet to distinguish them. In contrast, the SN-Adapter could produce more discriminative values between the two categories and address the ambiguity of PointNet by an ensemble with a large weight γ. As for Figure 3 (b), when PointNet confidently predicts the wrong category, our SN-Adapter can put the final prediction back on track with the confidence score for the correct category. Figure 3 (c) shows that when both of their predictions are wrong, the interpolation of SN-Adapter can still contribute to the right answer.

4.2. Qualitative Analysis

Why does the k-NN retrieval work for point cloud analysis? For one, due to the difficulty of data acquisition, the 3D community lacks large-scale high-quality training datasets, and existing methods can only learn from limited samples. In this situation, the representative 3D prototypes become much more significant since the construction of prototypes are not overly dependent on data distribution and can well represent the typical features of a category. In contrast, the 3D classifiers of deep neural networks greatly suffer from long-tail distributions of training data. That is to say, when the 3D samples of some categories are insufficient during training, the learnable classifier would not form the prediction preference of those unusual categories and fail to recognize them for testing. The k-NN upon spatial prototypes inherently overcomes such category imbalance via similarity-based retrieval, which hardly depends on the amount of training data.

4.3. Theoretical Analysis

We start from the perspective of learned embedding space to illustrate how SN-Adapter boosts the learned deep neural networks. The k-NN algorithm of SN-Adapter is able to associate pre-constructed spatial prototypes in close proximity. These adjacent prototypes normally have the same ground-truth labels and share similar semantic knowledge. Spatially, the entire 3D space can be divided into many discrete spherical regions. We define a spherical region in the embedding space as $N_\epsilon(x) = \{x' | ||x' - x||_2 \leq \epsilon\}$, where x denotes the spherical center and ϵ denotes its radius. The goal of SN-Adapter is based on the extracted feature of a test point cloud to retrieve clusters and then obtain representative knowledge from them.

For better retrieval performance, each spherical center prefers to have sufficiently pure spherical regions. In other words, the information of the representative prototypes should be convincing enough, formulated as $\forall x' \in N_\epsilon(x), gt(x') = gt(x)$, where $gt(\cdot)$ denotes the ground-truth label. We then define $C(N_\epsilon)$ and $P(N_\epsilon)$ as the coverage and purity of all spherical regions. The optimal $C(N_\epsilon)$ desires ϵ to be large enough to cover the entire space, while the purity requires a smaller ϵ to contain as few deviating prototypes as possible. Therefore, we need to consider the trade-off between both coverage and purity. Formally, we expect to obtain the specific ϵ that satisfies $\epsilon^* = max\{\epsilon : P(N_\epsilon) \geq \alpha\}$, where α serves as a threshold for $P(N_\epsilon)$ and also the maximum function that helps increase $C(N_\epsilon)$. In our experiments, we do not explicitly set the value α, but leverage an appropriate number of k nearest neighbors to implicitly obtain the optimal trade-off for better retrieving prototypical knowledge.

Method	OA (%)	mAcc (%)	k
PointNet [25]	89.34	85.79	-
+ SN-Adapter	**90.68**	**86.47**	21
PointNet++ [26]	92.42	89.22	-
+ SN-Adapter	**93.48**	**90.00**	77
DGCNN [39]	92.18	89.10	-
+ SN-Adapter	**92.99**	**89.70**	24
PCT [15]	93.27	89.99	-
+ SN-Adapter	**93.56**	**90.17**	110
CurveNet [42]	93.84	91.14	-
+ SN-Adapter	**94.25**	**91.50**	2

Table 2. **Shape classification on ModelNet40 [40]** dataset.

Method	OA (%)	mAcc (%)	k
PointNet [25]	68.2	63.4	-
+ SN-Adapter	**70.1**	**64.2**	128
SpiderCNN [44]	73.7	69.8	-
+ SN-Adapter	**74.4**	**70.5**	68
PointNet++ [26]	77.9	75.4	-
+ SN-Adapter	**79.2**	**76.2**	16
DGCNN [39]	78.1	73.6	-
+ SN-Adapter	**78.9**	**74.0**	140
PointMLP [2]	85.7	84.0	-
+ SN-Adapter	**86.3**	**84.6**	5

Table 3. **Shape classification on ScanObjectNN [33]** dataset.

5. Experiments

5.1. Shape Classification

Settings We evaluate our SN-Adapter on two widely adopted datasets for shape classification: ModelNet40 [40] and ScanObjectNN [33]. We select several representative methods and append SN-Adapter upon them: PointNet [25], PointNet++ [26], SpiderCNN [44], DGCNN [39], PCT [15], CurveNet [42], and PointMLP [2]. We set the last linear layer as $\Theta(\cdot)$ and all the precedent layers as $\Phi(\cdot)$. The overall accuracy (OA) and class-average accuracy (mAcc) are adopted as evaluation metrics. Note that, as our SN-Adapter requires no training time, we utilize simple loops to search for the best k within minutes.

Performance In Table 2 and Table 3, we show the enhancement results of SN-Adapter on the two datasets, respectively. On ModelNet40 [40] of synthetic data, PointNet++ is boosted by +1.06% mAcc, which has surpassed the more complicated DGCNN by +1.30%. On ScanObjectNN [33] of real-world data, SN-Adapter shows stronger complementary characteristics to the trained networks, which boosts PointNet by 1.9% OA and PointNet++ by +1.3% OA. For the state-of-the-art PointMLP, our SN-Adapter improves it by +0.6% OA and +0.6% mAcc.

Method	mIoU$_I$ (%)	k
DGCNN [39]	85.17	-
+ SN-Adapter	**85.26**	22
PointNet++ [26]	85.40	-
+ SN-Adapter	**85.47**	1
PointMLP [2]	85.69	-
+ SN-Adapter	**85.86**	1
CurveNet [42]	86.58	-
+ SN-Adapter	**86.69**	64

Table 4. **Part segmentation on ShapeNetPart [45]**.

Method	AP$_{25}$ (%)	AR$_{25}$ (%)
VoteNet [10]	57.84	80.92
+ SN-Adapter	**58.46**	**83.74**
3DETR-m [21]	64.60	77.22
+ SN-Adapter	**65.16**	**84.56**

Table 5. **3D object detection on ScanNetV2 [7]**.

5.2. Part Segmentation

Settings For part segmentation, we test our SN-Adapter on ShapeNetPart [45] dataset and select the four baseline models: DGCNN [39], PointNet++ [26], PointMLP [2] and CurveNet [42]. We follow other settings the same as the shape classification experiments and report the mean IoU across all instances in the dataset, denoted as mIoU$_I$.

Performance As the part segmentation benchmark has long been saturated, a slight improvement for mIoU$_I$ is also worth mentioning. In Table 4, we observe the biggest improvement of +0.17% mIoU$_I$ is on PointMLP, compared to Curvenet's +0.11% and DGCNN's +0.09%. This indicates that the stronger feature encoder $\Phi(\cdot)$ contributes to better part-wise prototypes for the retrieval of SN-Adapter.

5.3. 3D Object Detection

Settings For 3D object detection on ScanNetV2 [7], we select VoteNet [10] and 3DETR-m [21] as the baseline models to test our SN-Adapter. We set the MLP-based classification head as $\Theta(\cdot)$ and the scene-level feature extractor as $\Phi(\cdot)$. The SN-Adapter is inserted after $\Theta(\cdot)$ and before the 3D NMS. We report the mean Average Precision (AP$_{25}$) and mean Average Precision (AR$_{25}$) at 0.25 IoU threshold. For time efficiency, the hyperparameter k is simply set as 32 for the two detectors.

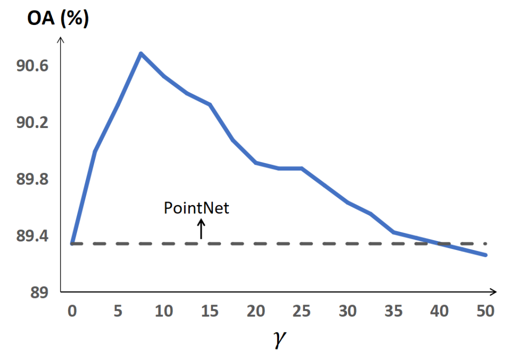

Figure 4. Ablation study of **interpolation ratio** γ.

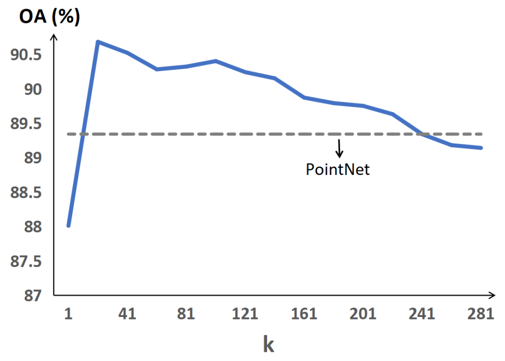

Figure 5. Ablation study of **the number of nearest neighbors,** k.

Metric	PointNet	DGCNN	CurveNet
Manhattan	90.36	92.63	94.00
Chebyshev	88.29	92.26	93.56
Hamming	88.37	92.22	93.72
Canberra	90.07	91.82	93.92
Braycurtis	90.24	92.46	94.04
Euclidean	**90.68**	**92.99**	**94.25**

Table 6. **Different distance metrics** for SN-Adapter on ModelNet40 [40] dataset with overall accuracy (OA) (%).

PE	Pooling	PointNet	PointNet++	PCT
-	-	90.11	93.19	93.52
Fourier	Avg.	89.99	93.11	93.52
Fourier	Max.	89.95	93.23	93.48
Sin/cos	Avg.	90.03	**93.48**	**93.56**
Sin/cos	Max.	**90.68**	93.15	93.48

Table 7. **Different positional encodings (PE) and pooling operations** of SN-Adapter for on ModelNet40 [40] dataset with overall accuracy (OA) (%). 'Fourier' and 'Sin/cos' denote Fourier and trigonometric encoding functions [21], respectively. The first row denotes SN-Adapter without any positional encodings.

Performance Table 5 presents the enhanced detection performance by SN-Adapter. For AR_{25}, we significantly improve by +2.82% on VoteNet and +7.34% on 3DETR-m. This indicates that the spatial prototypical knowledge can effectively avoid the removal of false duplicated bounding boxes in 3D space. More specifically, some spatially neighboring boxes, which have incorrectly similar scores and should have been removed by 3D NMS, can be rectified and reserved as outputs.

5.4. Ablation Study

Main hyperparameters We here conduct the ablation study concerning two hyperparameters: γ and k. We adopt PointNet [25] with SN-Adapter and experiment shape classification on ModelNet40 [41]. As the γ varies from 0 to 50 in Figure 4, the enhancement of SN-Adapter peaks around 8, but becomes harmful after 10. This indicates the SN-Adapter would adversely affect the baseline model under a too large proportion, and requires a proper interpolation ratio for best introducing the spatial prototypical knowledge. The results in Figure 5 show that our SN-Adapter is not very sensitive to k when it is large enough (over 80), which has already covered the most contributing prototypes to the final classification.

Distance metrics for retrieval. Different distance metrics for SN-Adapter affect the retrieval of nearest spatial prototypes, which further leads to different performance enhancement over the baseline models. We evaluate our SN-Adapter with different distance metrics for shape classification on ModelNet40 [41] and adopt three baseline models: PointNet [25], DGCNN [39], and CurveNet [42]. As reported in Table 6, for three baseline models, Euclidean distance performs better, which can be better reveal the point distribution in the 3D space.

Positional encodings. For shape classification, we equip the sample-wise $Proto^{cls}$ with global positional vectors to preserve the spatial distributions of points. In Table7, we explore the best way to obtain such vectors concerning the encoding functions and pooling operations. We evaluate three baseline models: PointNet [25], PointNet++ [26] and PCT [15] for shape classification on ModelNet40 [41]. As reported, 'Sin/cos' encoding function has more advantages, which can bring favorable performance boost to the 'SN-Adapter without PE'.

3D Object Detection We insert our SN-Adapter into the trained object detectors before 3D NMS, and summarize the object-wise prototypes with positional encodings. We here explore the effectiveness of both insert position and positional encodings. In Table 8, we select 3DETR-m [21] as

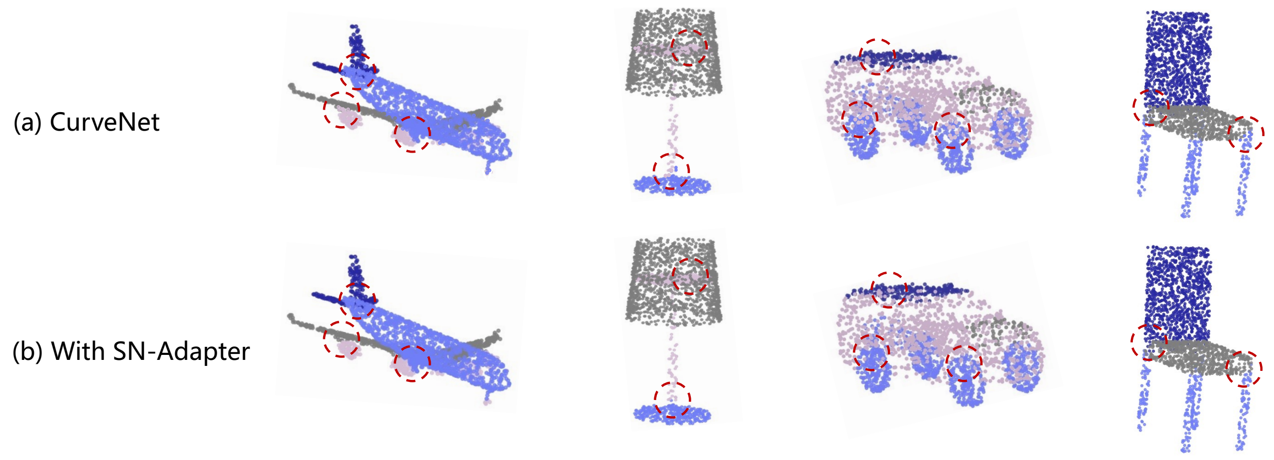

Figure 6. **Visualization of part segmentation without (a) and with (b) our SN-Adapter** on ShapeNetpart [45] dataset. We select CurveNet [42] as the baseline model and highlight the differences by red circles (Zoom in for a better view).

Method	AP_{25} (%)	AR_{25} (%)
3DETR-m	64.60	77.22
3DETR-m + SN-Adapter	**65.16**	**84.56**
After 3D NMS	64.62	78.19
Without PE	65.02	83.48

Table 8. Ablation study of **SN-Adapter for 3D object detection** on ScanNetV2 [7]. For last two rows, we respectively insert SN-Adapter after 3D NMS and discard the positional encodings.

Method	Score (%)	Latency	Memory
DGCNN	92.18	0.022s	9.74 GiB
+ SN-Adapter	**92.99**	0.046s	10.06 GiB
CurveNet	86.58	0.607s	10.93 GiB
+ SN-Adapter	**86.69**	0.834s	11.50 GiB

Table 9. **The extra costs of SN-Adapter for time and memory.** We test on a single RTX 3090 GPU with batch size 64 and report the OA/ $mIoU_I$ for DGCNN/ CurveNet.

our baseline on ScanNetV2 [7] dataset. As shown, if after 3D NMS, the SN-Adapter cannot bring noteworthy boosts, since the remaining 3D boxes filtered by NMS are already the most confident ones for the detector. Also, blending the positional encodings can improve the performance of SN-Adapter for introducing more positional knowledge into the prototypes.

Extra Costs of SN-Adapter. Besides the enhancement on scores, we explore if our SN-Adapter would cause too much extra time and memory costs over baseline models. We utilize a single RTX 3090 GPU with batch size 64 for testing and select two baseline models: DGCNN [39] for shape classification on ModelNet40 [40] and CurveNet [42] for part segmentation on ShapeNetPart [45]. As shown in Table 9, our non-parametric SN-Adapter can achieve superior performance-cost trade-off to enhance already trained networks without re-training.

5.5. Visualization

In Figure 6, we visualize the results of CurveNet [42] with and without our SN-Adapter for part segmentation on ShapeNetPart [45] dataset. As shown, our SN-Adapter mainly improves the segmentation of points located in the connection areas between different object parts. Such points normally contain the semantic knowledge of both object parts and would confuse the learned 3D classifier of deep neural networks. In contrast, our SN-Adapter could alleviate such issue by retrieving from the prototypes to obtain better part-wise discrimination capability.

6. Conclusion

We propose Spatial-Neighbor Adapter (SN-Adapter), a plug-and-play enhancement module for existing 3D networks without extra parameters and re-training. From the pre-constructed prototypes, SN-Adapter leverages k nearest neighbors to retrieve spatial knowledge and effectively boost the 3D networks by providing complementary characteristics. **Limitations.** Although our SN-Adapter can be generalized to various tasks, *e.g.*, shape classification, part segmentation, and 3D object detection, the performance enhancement for part segmentation is relatively lower than others. Our future work will focus on designing more advanced part-wise prototypes for better segmentation results.

References

[1] Aitor Aldoma, Zoltan-Csaba Marton, Federico Tombari, Walter Wohlkinger, Christian Potthast, Bernhard Zeisl, Radu Bogdan Rusu, Suat Gedikli, and Markus Vincze. Tutorial: Point cloud library: Three-dimensional object recognition and 6 dof pose estimation. *IEEE Robotics & Automation Magazine*, 19(3):80–91, 2012.

[2] Anonymous. Rethinking network design and local geometry in point cloud: A simple residual MLP framework. In *Submitted to The Tenth International Conference on Learning Representations*, 2022. under review.

[3] Mathilde Caron, Hugo Touvron, Ishan Misra, Hervé Jégou, Julien Mairal, Piotr Bojanowski, and Armand Joulin. Emerging properties in self-supervised vision transformers. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 9650–9660, 2021.

[4] Jingdao Chen, Zsolt Kira, and Yong K Cho. Deep learning approach to point cloud scene understanding for automated scan to 3d reconstruction. *Journal of Computing in Civil Engineering*, 33(4):04019027, 2019.

[5] Xiaozhi Chen, Huimin Ma, Ji Wan, Bo Li, and Tian Xia. Multi-view 3d object detection network for autonomous driving. In *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, pages 1907–1915, 2017.

[6] Nikolaus Correll, Kostas E Bekris, Dmitry Berenson, Oliver Brock, Albert Causo, Kris Hauser, Kei Okada, Alberto Rodriguez, Joseph M Romano, and Peter R Wurman. Analysis and observations from the first amazon picking challenge. *IEEE Transactions on Automation Science and Engineering*, 15(1):172–188, 2016.

[7] Angela Dai, Angel X Chang, Manolis Savva, Maciej Halber, Thomas Funkhouser, and Matthias Nießner. Scannet: Richly-annotated 3d reconstructions of indoor scenes. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 5828–5839, 2017.

[8] Jacob Devlin, Hao Cheng, Hao Fang, Saurabh Gupta, Li Deng, Xiaodong He, Geoffrey Zweig, and Margaret Mitchell. Language models for image captioning: The quirks and what works. *arXiv preprint arXiv:1505.01809*, 2015.

[9] Jacob Devlin, Saurabh Gupta, Ross Girshick, Margaret Mitchell, and C Lawrence Zitnick. Exploring nearest neighbor approaches for image captioning. *arXiv preprint arXiv:1505.04467*, 2015.

[10] Zhipeng Ding, Xu Han, and Marc Niethammer. Votenet: A deep learning label fusion method for multi-atlas segmentation. In *International Conference on Medical Image Computing and Computer-Assisted Intervention*, pages 202–210. Springer, 2019.

[11] Yu Du, Fangyun Wei, Zihe Zhang, Miaojing Shi, Yue Gao, and Guoqi Li. Learning to prompt for open-vocabulary object detection with vision-language model. *arXiv preprint arXiv:2203.14940*, 2022.

[12] Peng Gao, Shijie Geng, Renrui Zhang, Teli Ma, Rongyao Fang, Yongfeng Zhang, Hongsheng Li, and Yu Qiao. Clip-adapter: Better vision-language models with feature adapters. *arXiv preprint arXiv:2110.04544*, 2021.

[13] Ankit Goyal, Hei Law, Bowei Liu, Alejandro Newell, and Jia Deng. Revisiting point cloud shape classification with a simple and effective baseline. *arXiv preprint arXiv:2106.05304*, 2021.

[14] Edouard Grave, Moustapha M Cisse, and Armand Joulin. Unbounded cache model for online language modeling with open vocabulary. *Advances in neural information processing systems*, 30, 2017.

[15] Meng-Hao Guo, Jun-Xiong Cai, Zheng-Ning Liu, Tai-Jiang Mu, Ralph R Martin, and Shi-Min Hu. Pct: Point cloud transformer. *Computational Visual Media*, 7(2):187–199, 2021.

[16] Neil Houlsby, Andrei Giurgiu, Stanislaw Jastrzebski, Bruna Morrone, Quentin De Laroussilhe, Andrea Gesmundo, Mona Attariyan, and Sylvain Gelly. Parameter-efficient transfer learning for nlp. In *International Conference on Machine Learning*, pages 2790–2799. PMLR, 2019.

[17] Urvashi Khandelwal, Angela Fan, Dan Jurafsky, Luke Zettlemoyer, and Mike Lewis. Nearest neighbor machine translation. *arXiv preprint arXiv:2010.00710*, 2020.

[18] Urvashi Khandelwal, Omer Levy, Dan Jurafsky, Luke Zettlemoyer, and Mike Lewis. Generalization through memorization: Nearest neighbor language models. *arXiv preprint arXiv:1911.00172*, 2019.

[19] Kiyosumi Kidono, Takeo Miyasaka, Akihiro Watanabe, Takashi Naito, and Jun Miura. Pedestrian recognition using high-definition lidar. In *2011 IEEE Intelligent Vehicles Symposium (IV)*, pages 405–410. IEEE, 2011.

[20] 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*, 31:820–830, 2018.

[21] Ishan Misra, Rohit Girdhar, and Armand Joulin. An end-to-end transformer model for 3d object detection. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 2906–2917, October 2021.

[22] Arsalan Mousavian, Clemens Eppner, and Dieter Fox. 6-dof graspnet: Variational grasp generation for object manipulation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 2901–2910, 2019.

[23] Luis E Navarro-Serment, Christoph Mertz, and Martial Hebert. Pedestrian detection and tracking using three-dimensional ladar data. *The International Journal of Robotics Research*, 29(12):1516–1528, 2010.

[24] Tobias Plötz and Stefan Roth. Neural nearest neighbors networks. *Advances in Neural information processing systems*, 31, 2018.

[25] Charles R Qi, Hao Su, Kaichun Mo, and Leonidas J Guibas. Pointnet: Deep learning on point sets for 3d classification and segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 652–660, 2017.

[26] Charles R Qi, Li Yi, Hao Su, and Leonidas J Guibas. Pointnet++: Deep hierarchical feature learning on point sets in a metric space. *arXiv preprint arXiv:1706.02413*, 2017.

[27] Yongming Rao, Wenliang Zhao, Guangyi Chen, Yansong Tang, Zheng Zhu, Guan Huang, Jie Zhou, and Jiwen Lu.

Denseclip: Language-guided dense prediction with context-aware prompting. *arXiv preprint arXiv:2112.01518*, 2021.

[28] Radu Bogdan Rusu, Nico Blodow, Zoltan Csaba Marton, and Michael Beetz. Close-range scene segmentation and reconstruction of 3d point cloud maps for mobile manipulation in domestic environments. In *2009 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 1–6. IEEE, 2009.

[29] Shuran Song, Samuel P Lichtenberg, and Jianxiong Xiao. Sun rgb-d: A rgb-d scene understanding benchmark suite. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 567–576, 2015.

[30] Yi-Lin Sung, Jaemin Cho, and Mohit Bansal. Vl-adapter: Parameter-efficient transfer learning for vision-and-language tasks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5227–5237, 2022.

[31] Hugues Thomas, Charles R Qi, Jean-Emmanuel Deschaud, Beatriz Marcotegui, François Goulette, and Leonidas J Guibas. Kpconv: Flexible and deformable convolution for point clouds. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 6411–6420, 2019.

[32] Zhaopeng Tu, Yang Liu, Shuming Shi, and Tong Zhang. Learning to remember translation history with a continuous cache. *Transactions of the Association for Computational Linguistics*, 6:407–420, 2018.

[33] 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/CVF International Conference on Computer Vision*, pages 1588–1597, 2019.

[34] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in neural information processing systems*, pages 5998–6008, 2017.

[35] Francesco Verdoja, Diego Thomas, and Akihiro Sugimoto. Fast 3d point cloud segmentation using supervoxels with geometry and color for 3d scene understanding. In *2017 IEEE International Conference on Multimedia and Expo (ICME)*, pages 1285–1290. IEEE, 2017.

[36] Bram Wallace and Bharath Hariharan. Extending and analyzing self-supervised learning across domains. In *European Conference on Computer Vision*, pages 717–734. Springer, 2020.

[37] Mengmeng Wang, Jiazheng Xing, and Yong Liu. Actionclip: A new paradigm for video action recognition. *arXiv preprint arXiv:2109.08472*, 2021.

[38] Yan Wang, Wei-Lun Chao, Kilian Q Weinberger, and Laurens van der Maaten. Simpleshot: Revisiting nearest-neighbor classification for few-shot learning. *arXiv preprint arXiv:1911.04623*, 2019.

[39] Yue Wang, Yongbin Sun, Ziwei Liu, Sanjay E Sarma, Michael M Bronstein, and Justin M Solomon. Dynamic graph cnn for learning on point clouds. *Acm Transactions On Graphics (tog)*, 38(5):1–12, 2019.

[40] 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.

[41] 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.

[42] Tiange Xiang, Chaoyi Zhang, Yang Song, Jianhui Yu, and Weidong Cai. Walk in the cloud: Learning curves for point clouds shape analysis. *arXiv preprint arXiv:2105.01288*, 2021.

[43] Mutian Xu, Runyu Ding, Hengshuang Zhao, and Xiaojuan Qi. Paconv: Position adaptive convolution with dynamic kernel assembling on point clouds. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 3173–3182, 2021.

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

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

[46] 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.

[47] Renrui Zhang, Ziyu Guo, Wei Zhang, Kunchang Li, Xupeng Miao, Bin Cui, Yu Qiao, Peng Gao, and Hongsheng Li. Pointclip: Point cloud understanding by clip. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8552–8562, 2022.

[48] Bo Zheng, Yibiao Zhao, Joey C Yu, Katsushi Ikeuchi, and Song-Chun Zhu. Beyond point clouds: Scene understanding by reasoning geometry and physics. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3127–3134, 2013.