SDSimPoint: Shallow-Deep Similarity Learning for Few-Shot Point Cloud Semantic Segmentation

Jiahui Wang, Haiyue Zhu† , Haoren Guo, Abdullah Al Mamun, Cheng Xiang,
Clarence W. de Silva *Life Fellow, IEEE* and Tong Heng Lee

***Abstract*—3D point cloud semantic segmentation is a fundamental task in computer vision. As the fully-supervised approaches suffer from the generalization issue with limited data, few-shot point cloud segmentation models have been proposed to address the flexible adaptation. Nevertheless, due to the class-agnostic nature of the few-shot pre-training, its pre-trained feature extractor is hard to capture the class-related intrinsic and abstract information. Therefore, we introduce the new concept of Shallow and Deep Similarities and propose a Shallow-Deep Similarity learning Network (SDSimPoint) that aims to learn both shallow (superficial geometry, color, etc.) and deep similarities (intrinsic context and semantics, etc.) between the support and query samples, thereby boosting the performance. Moreover, we design a Beyond Episode Attention Module (BEAM) to enlarge the region of attention mechanism from a single episode to the entire dataset by utilizing the memory units, which enhances the extraction ability to better capture the shallow and deep similarities. Furthermore, our distance metric function is learnable in the proposed framework, which can better adapt to complex data distributions. Our proposed SDSimPoint consistently demonstrates substantial improvements compared to baseline approaches across various datasets in diverse few-shot point cloud semantic segmentation settings.**

***Index Terms*—Few-Shot Learning, Point Cloud, Semantic Segmentation, Similarity Learning, Self-Supervised Pre-training.**

I. Introduction

THE learning and development of 3D scenes is becoming increasingly important in the modern world. This advancement in 3D scene learning is beneficial for a variety of applications, including robotics, autonomous vehicles, manufacturing, and other industries with various data formats such as point cloud and mesh [1]. Among these applications that benefit from 3D scene learning, the task of semantic segmentation of point cloud data is a critical and essential one. The goal of point cloud semantic segmentation is to recognize and classify the semantic properties and characteristics of each data point that makes up the 3D object or scene. With the recent developments in deep learning with neural networks, several fully-supervised models have been proposed for point cloud semantic segmentation [2]–[5]. For instance, PointNet [2] is the first work that processes point clouds with Multi-Layer Perceptron (MLP), which uses a symmetric function to extract the features from the point clouds. The LGGCM [5] adopted a local spatial attention convolution with a smooth module and a global spatial attention module with a gated unit for effective receptive fields with long-range contexts at a relatively low computational cost. Generally, those fully-supervised point cloud semantic segmentation methods are delicately designed to capture the discriminative features based on the trained classes. Therefore, the performance of a fully-supervised method suffers from poor generalization ability with serious class bias if the training dataset is insufficient and monotonous. As a result, for those point cloud semantic segmentation applications with data scarcity or task liability, the fully-supervised approaches are hard to meet the practical requirements.

To address the aforementioned issue, previous research adopts the few-shot learning to enable the model to adapt to novel classes effectively with a few samples. [6]–[9]. However, for the few-shot 3D point cloud semantic segmentation tasks, we hardly can use a well-trained backbone with abundant class numbers to conduct "open-vocabulary" inference due to the scale of the available data in the real world. Some researchers tried to achieve this task by pre-training a backbone in a relatively small dataset, learning the per-point feature, and then measuring the query-support distance for prediction [10]–[13]. This paradigm is effective but still suffers from a data scale for supervised pre-training.

Some researchers [14] used a self-supervised pre-training to enlarge the data scale, but the learned features are improvable in some perspective. The features extracted by such a backbone can be competent in many scenarios as they contain superficial information for effective similarity measures such as the geometry, color, and position properties between the objects, which we term as shallow similarity. However, due to the limitation on quality and scalability of the 3D datasets, the pre-trained 3D feature extractor might not be able to understand universal high-level features, especially when facing a complex scene. We argue that such shallow similarity using superficial information directly extracted by the backbone makes it difficult to capture the deep relationship between the objects, which relies on intrinsic information like semantics, context, and rationality, especially for a self-supervised pre-trained backbone. For instance, distinguishing sofas from different scenes with diverse colors and shapes may need such deep similarity for precise predictions, and the

J. Wang, H. Guo, A. Mamun, C. Xiang, and T. H. Lee are with the College of Design and Engineering, Electrical and Computer Engineering, National University of Singapore, 117582, Singapore.

†: Corresponding author. H. Zhu is with the Singapore Institute of Manufacturing Technology (SIMTech), Agency for Science, Technology and Research (A*STAR), 2 Fusionopolis Way, 138634, Singapore. zhu_haiyue@simtech.a-star.edu.sg

C. W. de Silva is with the Department of Mechanical Engineering, University of British Columbia, Vancouver, BC V6T 1Z4, Canada. desilva@mech.ubc.ca.

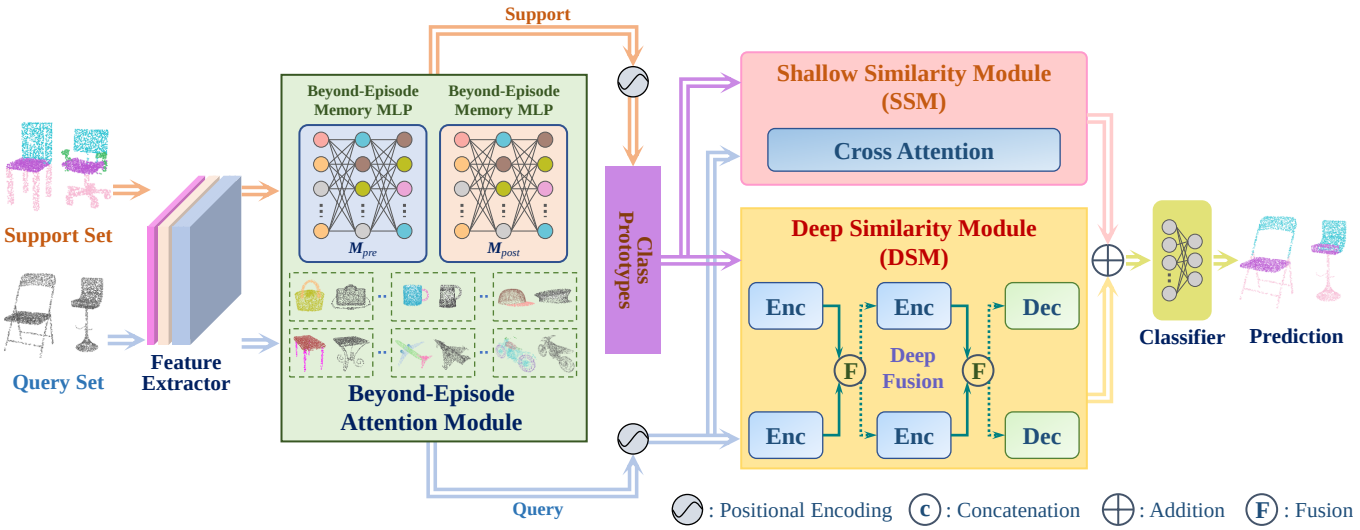


Fig. 1: The architecture of SDSimPoint. We introduce the new categorization of shallow and deep similarities for few-shot point cloud segmentation. The "shallow" refers to superficial information that can be well captured by the pre-trained extraction & attention module, while the "deep" refers to class-related intrinsic characteristics that are hard to capture due to class-agnostic pre-training. Therefore, we design the SSM and DSM to separately capture the shallow and deep similarities, where a query-prototype deep-fusion mechanism is introduced in DSM. Moreover, a BEAM module is introduced for effective attention to break the episode constraint and capture the long-range global information.

shallow similarity based on superficial information is incapable of capturing the intrinsic feature due to the diversities and inconsistency of appearances and contexts. Essentially, this is because the backbone extractor employed in few-shot models is pre-trained in a task-agnostic manner, which is different from the fully-supervised scenarios. As a result, such models only consider shallow similarity, which generally leads to the unsatisfied performance of complex objects and scenes. Inspired by this finding, we propose SDSimPoint in this work, which takes both the shallow and deep similarities into account simultaneously to enhance the prediction performance for complex scenarios.

Moreover, to better capture the effective similarity between query and support samples, the self-attention mechanism is commonly used in many few-shot learners to enhance the learning of context information [10], [15], [16]. In a traditional fully-supervised manner [10], the self-attention unit looks through the samples in a batch that contains varied samples with diverse information. As a result, a larger batch size encourages self-attention to capture the global dependencies better and learn more robust attention patterns, thus improving performance. However, when it comes to the episodic training in the few-shot learning, the horizon of self-attention is constrained within one episode in N-way K-shot. As N and K are small, they limit the exposure of self-attention to diverse examples and may negatively prioritize attention to local dependencies. Motivated by this finding, we propose a Beyond-Episode Attention Module (BEAM) for the few-shot learning in episode training, which facilitates self-attentions to capture the long-range global attention information across episodes. The main contributions of this work are summarized as follows.

- We develop a new categorization of shallow and deep similarities for few-shot learning and propose SDSimPoint, a shallow-deep similarity learning framework for point cloud semantic segmentation.
- A Shallow Similarity Module (SSM) is designed to learn the superficial similarity, and a Deep Similarity Module (DSM) is proposed to capture the intrinsic similarity through deep query-prototype interaction and fusion.
- A Beyond Episode Attention Module (BEAM) is introduced that enables the attention mechanism to break the episode constraint and capture the long-range global information.

II. Related Works

A. Few-Shot Learning

The mainstream few-shot learning methods can be broadly categorized into four categories, i.e., data-based [17], [18], model-based [7], [19]–[21], optimization-based [22]–[24], and metric-based [7]–[9], [25], [26] methods. Data-based methods, exemplified by FlipDA [27], aim to generate additional training data or features through augmentation or synthesis to improve the few-shot performance. Model-based methods leverage some specific model architectures, such as recurrent and memory-enhanced networks, to enable the memorization of samples even with minimal exposure. In optimization-based methods, MAML [24] focuses on finding optimal initial states or parameter settings during the training, which enables the model to perform well on new tasks by updating with a small number of samples. He et al. [28] proposed a registration module that enables the model to learn robust representation with few samples. For metric-based methods,

the central concept is to learn a good feature embedding space for acquiring similarity measure metrics, which can effectively capture the correspondence between support and query samples. ProtoNet [9] aims to generate the prototype for each class, which will be used as the anchor to calculate the similarity metric and predict the classes for queries. DFR [29] focuses on extracting the class-discriminative features from the variations distracting the metric learning, thus improving few-shot classification performance. TRSN [30] divides the few-shot learning task into three subparts, explicitly models the task-related saliency features, and makes use of them for prediction. BSSD [31] replaces global feature vectors with enhanced local feature maps that are more consistent between seen and unseen classes to mitigate the data-scarcity issue, then uses Sinkhorn distance to measure the similarity. Some works [8] use graph-based techniques to build accurate correspondence for better metric learning interpretability. Thanks to its simplicity and effectiveness, the metric-based methods have been widely used in recent few-shot learning research.

B. Point Cloud Semantic Segmentation

The pioneering work, PointNet [2], shows the potential of adopting deep learning in point cloud semantic segmentation. More models are proposed in different perspective [4], [5], [32]–[37]. They can be divided into the following categories: graph-based [5], [32], MLP-based [4], [33], [34] and transformer-based [35]–[37] methods. SPG [32] adopts a method to extract super points from a large point cloud scene and construct a correspondence graph for contextual information learning. MLP-based methods, taking PointNeXt [34] as an example, use a set of improved training and sampling strategies and design residual bottleneck layers as well as separable MLPs to facilitate efficient and impactful model up-scaling. PointTransformer [35] first import transformer into a 3D point cloud. It treats points of the down-sampled scene as the token of transformer blocks with a designed vector self-attention to effectively learn the information of 3D point clouds. Recently, PointTransformerV3 [37] represents a notable advancement over the original PointTransformer. This improvement primarily involves substituting the meticulous neighbor search with a more efficient serialized neighbor mapping technique applied to point clouds structured according to distinct patterns.

In the realm of few-shot point cloud semantic segmentation, several innovative approaches have emerged [10]–[13], [38]–[41]. AttMPTI [10] pioneered the use of label propagation to connect prototypes and query points using pre-trained feature extraction. PEFC [42] employs specialized components for prototype set expansion and adjustment based on query characteristics. PAPFZS3D [11] introduces a framework that simultaneously refines both prototypes and query features. SCAT [41] implements a class-specific, multi-layered attention transformer to establish detailed support-query feature associations. COSeg [13] utilizes a correlation memory-based network to uncover inherent query-prototype relationships. Seg-PN [40] offers a computationally efficient solution using a non-parametric encoder and correlation-driven interactions between support and query for point-wise classification.

III. Methodology

A. Problem Formulation

To facilitate the few-shot learning and inference pipeline, the few-shot point cloud semantic segmentation to address in this work is formulated as a N-way K-shot problem [10], [14], [43], [44]. Given a support set $\mathcal{S} = \{(\boldsymbol{P}_s^{n,k}, \boldsymbol{Y}_s^{n,k})_{k=1}^{K}\}_{n=1}^{N}$, where N is the number of classes, each class consists of K support point cloud samples $\boldsymbol{P}_s$ with the corresponding labels $\boldsymbol{Y}_s$. The query set is defined as $\mathcal{Q} = (\boldsymbol{P}_q, \boldsymbol{Y}_q)$ accordingly, where $\boldsymbol{Y}_q$ is the ground truth and not available during the training. The objective of this work is to design the network architecture $\mathcal{N}_\theta(\cdot)$ and obtain its optimal parameter θ^*,

$$\begin{aligned} \hat{\boldsymbol{Y}}_q &= \mathcal{N}_\theta(\boldsymbol{P}_q|\mathcal{S}), \\ \theta^* &= \underset{\theta}{\operatorname{argmin}} \sum_{\boldsymbol{P}_q \in \mathcal{Q}} \mathcal{L}\left(\boldsymbol{Y}_q, \hat{\boldsymbol{Y}}_q\right), \end{aligned} \tag{1}$$

where $\mathcal{L}$ is the loss function. By minimizing the loss, the information entropy between the prediction and the ground truth is optimized, which ensures a promising result. In this work, the episodic learning paradigm [10], [14], [43], [44] is employed for the few-shot training.

B. Overview

Fig. 1 illustrates the overall architecture of the proposed approach. Firstly, a pre-trained backbone is employed in the pipeline to perform the basic feature extraction, which is self-supervised from the public point cloud datasets [14]. To better capture the relationships across spatial locations in the feature embedding, the attention module is utilized to improve the similarity correspondence. Uniquely, the proposed BEAM in this work extends the attention horizon beyond the episode, which employs a memory unit across the episodes to enhance the long-range dependencies. After that, using the feature embedding from BEAM, the class prototypes are generated from the training samples in the support set, which will be used to measure the similarity with the query samples.

In this work, we propose shallow-deep similarity learning to effectively capture both the superficial and intrinsic information to build correspondence. Fig. 2 depicts the shallow similarity and deep similarity, respectively. The query feature embedding and the class prototypes from the support set are fed into both the Shallow Similarity Module (SSM) and the Deep Similarity Module (DSM) concurrently to grab the shallow and deep similarities. The SSM takes advantage of the direct feature embedding for similarity measure, which is responsible for capturing superficial similarities such as geometry, color, position, etc. The pre-trained feature extractor employed in few-shot models is trained in a task-agnostic manner, which is not as capable as those fully-supervised models to capture the class-specific high-level semantic information. Therefore, we propose the DSM with additional encoder-decoder structures, which are shown in Fig. 3 to make deep-fusion interactions between the query embedding and class prototypes. Finally, a learnable point classifier takes both the shallow and deep similarity feature maps as input and makes the final predictions for all points with the highest similarity classes.

C. Beyond-Episode Attention Module

Self-attention mechanism [45], [46] is widely used for capturing contextual information in many computer vision tasks. However, we argue that for the few-shot point cloud semantic segmentation, the attention horizon of self-attention is constrained within the current episode. This is very different from the fully-supervised scenarios, which can look through the samples from all randomly-formed batches across the whole abundant and diverse dataset. Such a constraint limits the understanding and utilization of the information in the scale of the whole training data. Furthermore, unlike the images, point clouds are hard to split into patches for self-attention due to their unordered nature. As a result, the self-attention for a point cloud with M points exhibits a time complexity of $\mathcal{O}(M^2)$, where M is normally a large number and thus requires tremendous computation resources.

Inspired by [47], the proposed BEAM adopts the concept of external attention and extends it into a two-stage hierarchical structure with beyond-episode awareness. Our BEAM consists of two memory units $\mathcal{M}_{pre}(\cdot)$ and $\mathcal{M}_{post}(\cdot)$, which is realized by the simple Multi-Layer Perceptron (MLP). Specifically, consider one point cloud $\boldsymbol{P} \in \{\mathcal{S}_i, \mathcal{Q}_i\}$ for i-th episode, the pre-memory unit $\mathcal{M}_{pre}(\cdot)$ processes the concatenation of backbone feature and position information,

$$\boldsymbol{F}_i^{pre} = \mathcal{M}_{pre}\Big(\big[\mathcal{B}(\boldsymbol{P}); pos(\boldsymbol{P})\big]\big|\boldsymbol{\theta}_i^{pre}\Big), \tag{2}$$

where $\mathcal{B}(\cdot)$ is the backbone extractor, $pos(\cdot)$ denotes the point xyz positions for $\boldsymbol{P}$, and $\boldsymbol{\theta}_i^{pre}$ refers to the weights for $\mathcal{M}_{pre}$ in the i-th episode. To enhance the memory unit with better beyond-episode awareness, we adopt a simple yet efficient Exponential Moving Average (EMA) rule for updating the weights, i.e.,

$$\widetilde{\boldsymbol{\theta}}_i^{pre} = \alpha\boldsymbol{\theta}_i^{pre} + (1-\alpha)\widetilde{\boldsymbol{\theta}}_{i-1}^{pre}, \tag{3}$$

where α is a decay hyper-parameter. During training, the $\boldsymbol{\theta}_i^{pre}$ is updated with the gradient descent. The feature from the pre-memory unit is then fed into a normalization layer, and the post-memory unit $\mathcal{M}_{post}(\cdot)$ takes the normalized feature as the input to generate the post-attention feature,

$$\boldsymbol{F}_i^{post} = \mathcal{M}_{post}\big(\text{Norm}(\boldsymbol{F}_i^{pre})|\boldsymbol{\theta}_i^{post}\big), \tag{4}$$

where $\text{Norm}(\cdot)$ denotes the normalization operation, and the same EMA rules apply for $\mathcal{M}_{post}(\cdot)$. Consequently, in one episode, for both support and query samples, the output feature is the concatenated feature from BEAM extraction and backbone extraction, together with the positional embedding, i.e.,

$$\boldsymbol{F}_t^{beam} = \text{Concat}\big(\mathcal{B}(\boldsymbol{P}_t); \boldsymbol{F}_t^{post}\big) + \boldsymbol{PE}, \tag{5}$$

where $t \in \{q, s\}$ denotes the respective query or support features, $\text{Concat}(\cdot)$ is the concatenation operation and $\boldsymbol{PE}$ represents the common sinusoidal positional embedding. $\boldsymbol{F}_t^{beam}$ will be used for both SSM and DSM to measure the similarity in the next stages.

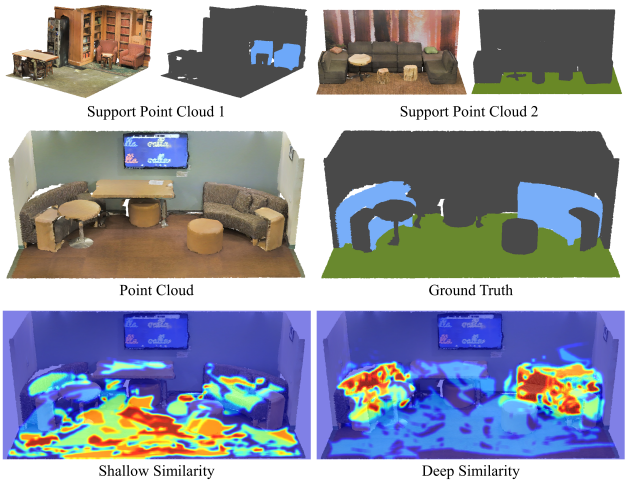
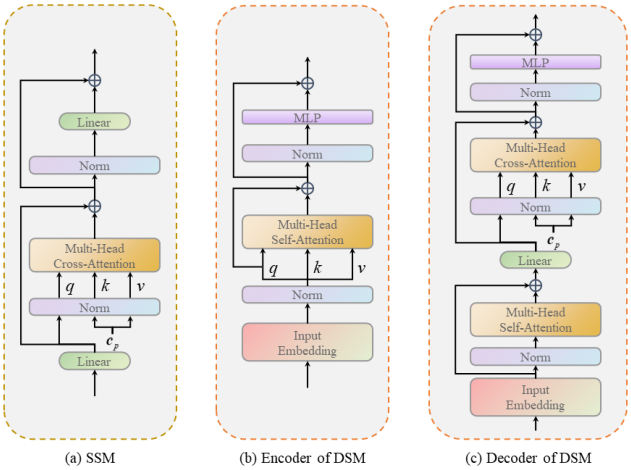


Fig. 2: Similarity visualization between query and support samples: The shallow similarity focuses more on the objects sharing similar geometry or color, while the deep similarity is more relative to the objects with different shapes and colors but sharing similar semantic information. The segmentation performance can be improved by combining both the shallow and deep similarities.

D. Shallow-Deep Similarity

We propose the shallow-deep similarity learning for few-shot point cloud segmentation. As common sense, objects within the same category are supposed to exhibit more commonalities in color appearance and geometric shape. Therefore, a native solution for few-shot point cloud segmentation is to leverage such characteristics to identify the analogous points/objects from the query point cloud to those support point clouds. Such superficial commonalities in color, geometry, etc., can be well captured by the pre-trained feature extractor, and many few-shot segmentors [10], [48], [49] have already adopted this approach as an intuitive-yet-efficient solution. In this work, we term this native superficial similarity as the "shallow similarity" to differentiate from the concept of "deep similarity", which conveys high-level intrinsic and abstract information such as semantics and context. It is worth noting that shallow similarity is an effective and complementary measure as it emphasizes the geometrical shape, color, and surface characteristics, which convey discriminative information at diverse frequency bands. Consequently, we propose an SSM to grasp such a shallow similarity.

However, we argue that due to the supervision constraints of the few-shot learning, a pre-trained few-shot feature extractor is good for capturing the shallow and superficial similarity between the query and support samples while being weak in capturing the deep and intrinsic similarity. It is worth noting that, for common fully-supervised learning, the concept of separation between shallow and deep similarities is not valid. This is because their feature extractors, either CNN or attentions, are supposed to handle both the shallow and deep similarities concurrently through the class-specific and sample-abundant fully supervised training. Essentially, the



(a) SSM (b) Encoder of DSM (c) Decoder of DSM

Fig. 3: The architectures of SSM and DSM. The SSM only has one cross-attention module to learn the superficial similarity, while each DSM block consists of an encoder and a decoder. The encoder in the DSM excavates deeper semantic features for queries and prototypes with self-attention and then sends such "high-level" information to the fusion process. The decoder in the DSM receives the fused feature and measures the deep similarity between query and prototypes with self- and cross-attention.

deep similarity is the intrinsic characteristics related to the specific class, which includes the high-level abstract semantics and context relationship with the background, etc. For a few-shot segmentation scene, both shallow and deep similarity will occur, but the attention paid to them should be adaptive due to different scenarios. For example, a scene with fewer objects and a straightforward layout can be segmented well based on shallow similarity. Segmentation for complex scenes, on the contrary, should involve deeper similarity for a more accurate result as they rely on high-level semantic-related information, which is more robust and representative.However, the pre-training of a few-shot feature extractor is not possible to have class-specific training as it aims to handle novel classes, which are unknown at the pretraining stage. Although we can fine-tune such a pre-trained network in a class-specific manner to enhance the learning for the deep similarity between samples, the few-shot tasks usually lack sufficient samples even for such fine-tuning. Therefore, a more effective way is to capture the deep similarity in the adaptation stage through some delicately designed mechanisms. To this end, we propose a DSM that uses a query-prototype deep-fusion mechanism and cascaded attention blocks to capture the deep similarity.

E. Shallow Similarity Module

In this work, our SSM is simply realized based on the Multi-Head Cross-Attention (MHCA) module, where its architecture is shown in Fig. 3(a). For both SSM and DSM, the similarity is measured between the query sample features $\boldsymbol{F}_q^{beam}$ and the class prototypes $\boldsymbol{c}_p$ generated from the support samples. The class prototype is the mean of the extracted support feature $\boldsymbol{F}_s^{beam}$ for points belonging to its respective class, which describes the distribution of class characteristics in the feature space. Next, both $\boldsymbol{F}_q^{beam}$ and $\boldsymbol{c}_p$ are projected into a latent space with simple linear layers and normalization, and the MHCA block is used to capture the cross attention for similarity measure, denoted as,

$$\boldsymbol{F}_q^{ca} = \text{MHCA}(\boldsymbol{F}_q^{beam}, \boldsymbol{c}_p, \boldsymbol{c}_p), \tag{6}$$

where $\text{MHCA}(\cdot)$ takes $\boldsymbol{F}_q^{beam}$ as query and $\boldsymbol{c}_p$ as key and value in the cross attention. Lastly, we employ the residual structure to get the final output for SSM,

$$\boldsymbol{F}_q^{ssm} = \mathcal{R}es(\boldsymbol{F}_q^{ca} + \boldsymbol{F}_q^{beam}), \tag{7}$$

where $\mathcal{R}es(\cdot)$ denotes the residual operation, i.e., $\mathcal{R}es(\boldsymbol{F}) = \boldsymbol{F} + \text{Linear}(\text{Norm}(\boldsymbol{F}))$ as indicated in Fig. 3(a).

F. Deep Similarity Module

Our intention for DSM is to facilitate more "interaction" between the query feature and support prototypes so that the intrinsic similarity can be measured through deep fusion. Such fusion technique has already been proved in many works such as GLIP [50], [51], FIBER [52], and etc. [25]. Different from SSM, which has no fusion process, this interaction enables the DSM gradually to learn different information rather than focus on dominant shallow similarity. The cascaded attention blocks help the DSM to utilize hierarchical features that facilitate the learning of deeper similarity. With these designs, the query feature not only holds basic representations but also incorporates details about the point-wise assignment, where the query point is classified into the group that shows the highest similarity to its corresponding prototype.

The DSM consists of M encoders and N decoders for both query and prototypes. The structure of the encoder and decoder are illustrated in Fig. 3(b) and (c), respectively. Let $Enc_{dsm}^m(\cdot)$ and $Dec_{dsm}^n(\cdot)$ denotes m-th encoder and n-th decoder, $m \in [0, M-1]$ and $n \in [0, N-1]$. The DSM takes the query sample features $\boldsymbol{F}_q^{beam}$ and the class prototypes $\boldsymbol{c}_p$ as the initial inputs, the query and prototype features are encoded for fusion as,

$$\boldsymbol{F}_t^{m+1} = Enc_{dsm}^m(\boldsymbol{F}_t^m), \tag{8}$$

where $t \in \{q, p\}$ denotes the query and class prototype, and $\boldsymbol{F}_q^0 = \boldsymbol{F}_q^{beam}$ and $\boldsymbol{F}_p^0 = \boldsymbol{c}_p$. After every encoder, the query and prototype features are firstly projected by a learnable matrix $\boldsymbol{W}_1 \in \mathbb{R}^{C \times C}$, and the fusion map $\boldsymbol{U}_m$ is calculated as,

$$\boldsymbol{U}_m = \text{SoftMax}\left(\frac{(\boldsymbol{F}_q^m \otimes \boldsymbol{W}_1) \otimes (\boldsymbol{F}_p^m \otimes \boldsymbol{W}_1)^\top}{\sqrt{C}}\right), \tag{9}$$

where $\otimes$ represents the matrix multiplication. Next, the fused features are obtained by multiplication with another learnable matrix $\boldsymbol{W}_2$,

$$\begin{aligned} \tilde{\boldsymbol{F}}_q^m &= \boldsymbol{U}_m \otimes (\boldsymbol{F}_p^m \otimes \boldsymbol{W}_2) \\ \tilde{\boldsymbol{F}}_p^m &= \boldsymbol{U}_m^\top \otimes (\boldsymbol{F}_q^m \otimes \boldsymbol{W}_2), \end{aligned} \tag{10}$$

where $\tilde{\boldsymbol{F}}_q^m$ and $\tilde{\boldsymbol{F}}_p^m$ denote the fused features respectively, where will be used as the inputs for next encoders.

In the decoding stage, the query and prototype features are decoded separately using $Dec_{dsm}^n(\cdot)$. The last decoded query feature will be used as the DSM output, which is denoted

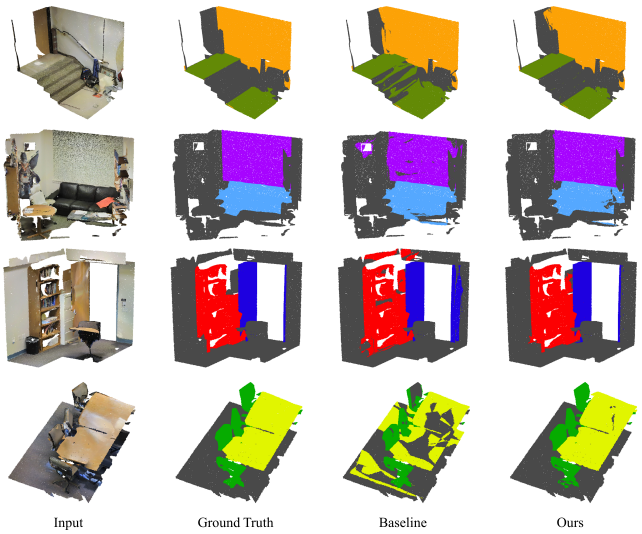


Fig. 4: Visualization of 2-way 1-shot segmentation result on **S3DIS** dataset. The involved classes are Wall and Floor (first row), Window and Sofa (second row), Bookcase and Door (third row), Table and Chair (last row), the background points are denoted by dark gray points.

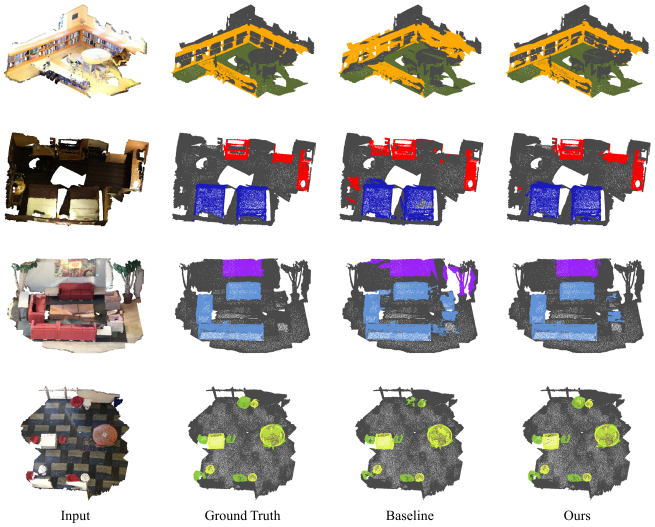


Fig. 5: Visualization of 2-way 1-shot segmentation result on **ScanNet** dataset. The involved classes are Bookshelf and Floor (first row), Cabinet and Bed (second row), Picture and Sofa (third row), Table and Chair (last row), the background points are denoted by dark gray points.

as $\boldsymbol{F}_q^{dsm}$. Finally, by fusing the SSM and DSM extracted similarity features, the prediction can be made by,

$$\boldsymbol{Y}_q = \text{CLS}(\boldsymbol{F}_q^{ssm} + \boldsymbol{F}_q^{dsm}), \tag{11}$$

where $\text{CLS}(\cdot)$ is a learnable MLP-based classifier. The loss functions for training and pre-training are discussed in Section IV-B.

IV. Experiment

A. Data Preparation

The S3DIS dataset [57] and the ScanNet dataset [58] are selected as benchmark datasets because they are widely recognized in 3D semantic segmentation tasks. The S3DIS dataset consists of five regions, comprising a total of 271 rooms, with each area containing multiple objects in 13 semantic categories, such as sofas, chairs, and tables, among others. However, processing entire rooms is challenging due to their substantial size. Therefore, we follow the approach outlined in [2] to divide rooms into blocks, with each block containing approximately 20,000 points. The ScanNet dataset [58] is more complex, with 1513 regions and 21 semantic classes. Similarly, we segment this dataset into blocks, with each block containing about 8000 points. Each point in the datasets mentioned above is represented by a 9-dimensional vector, which includes coordinate vectors: x, y, z; color vectors: r, g, b; and normed vectors n_x, n_y, n_z. During the training stage, we randomly sample 2048 points for each block. Following the approach in [10], [38], we divide each dataset into two non-overlapping subsets named S^0 and S^1 one of them is adopted only in testing, the other is used for the pre-training and training.

B. Implementation Details

For the backbone(feature extractor), we adopt DGCNN [59] as previous methods [10], [11] and pre-train it for 200 epochs to ensure the fairness in subsequent comparison. During the pre-training stage, we follow the paradigm in [14] that input two alternate views, denoted as $\boldsymbol{P}, \boldsymbol{P'} \in \mathbb{R}^{M\times 9}$, into the feature extractor $\mathcal{B}$ along with a classification head to obtain two segmentation outputs, $\boldsymbol{O}, \boldsymbol{O'} \in \mathbb{R}^{\mathbb{M}\times\mathbb{N}}$, where N_c represents the total number of classes in the pre-training dataset. The contrastive loss is then formulated to bring similar points in $\boldsymbol{O}$ and $\boldsymbol{O}^{'}$ closer together while pushing other points away in the feature space,

$$\mathcal{L}_{con} = \mathop{\mathbb{E}}_{\boldsymbol{o}_i\in\boldsymbol{O}}\left[-\log\left(\frac{e^{\boldsymbol{o}_i(\boldsymbol{o}_i')^\top}}{\sum_{j=1}^{M} e^{\boldsymbol{o}_i(\boldsymbol{o}_j')^\top}}\right)\middle|\boldsymbol{o}_j^{'}\in\boldsymbol{O}^{'}\right], \tag{12}$$

where o_i and o_i' represent the i-th point predictions in $\boldsymbol{O}$ and $\boldsymbol{O}'$, respectively. The learning rate is set to 0.001, and the value of k in the k-NN algorithm is set to 20. Our batch size is 16, designed for running the model on a single GPU, and a weight decay ratio of 0.01 is employed to prevent severe overfitting.

For the dynamic augmentor, the Chamfer Distance loss [60] is adopted to enhance the augmentation performance.

$$\mathcal{L}_{CD} = \lambda\{\frac{1}{|\boldsymbol{P}|}\sum_{x\in\boldsymbol{P}}\min_{y\in\boldsymbol{P'}}\|x-y\|_2^2 + \frac{1}{|\boldsymbol{P'}|}\sum_{y\in\boldsymbol{P'}}\min_{x\in\boldsymbol{P}}\|x-y\|_2^2\}, \tag{13}$$

where x and y denote points belonging to $\boldsymbol{P}$ and $\boldsymbol{P}'$, respectively. We update their parameters along with the feature extractor for 60 epochs, after which we fix the augmentor and solely pre-train the feature extractor. The coefficient λ for $\mathcal{L}_{CD}$ is set to -0.01. Given these settings, we utilize the Adam optimizer with a learning rate decay ratio of 0.5 and a decay step of 50 epochs.

In the training stage, the ground truth of the query set, $\boldsymbol{Y}_q$ and the prediction of our model, $\boldsymbol{Y}_q'$ are used for the cross-

TABLE I: S3DIS Semantic Segmentation Result Using m-IoU (Up,%) and Dice Coefficient (Down,%). S^i Means The Testing Dataset Is The i-th Split

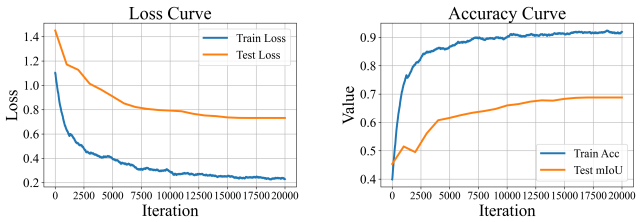
Method	2-way						3-way					
	1-shot			5-shot			1-shot			5-shot		
	S^0	S^1	mean	S^0	S^1	mean	S^0	S^1	mean	S^0	S^1	mean
FT [53]	36.34	38.79	37.57	56.49	56.99	56.74	30.05	32.19	31.12	46.88	47.57	47.23
ProtoNet [9]	48.39	49.98	49.19	57.34	63.22	60.28	40.81	45.07	42.94	49.05	53.42	51.24
AttProtoNet [9]	50.98	51.90	51.44	61.02	65.25	63.14	42.16	46.76	44.46	52.20	56.20	54.20
MPTI [10]	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 [10]	53.77	55.94	54.86	61.67	67.02	64.35	45.18	49.27	47.23	54.92	56.79	55.86
Aug-FT-P [38]	54.88	58.87	56.88	68.83	68.57	68.70	49.44	50.68	50.06	60.76	61.28	61.02
Aug-FT-A [38]	55.65	59.17	57.41	67.80	69.04	68.42	48.52	50.83	49.68	60.76	58.62	59.69
CWT [54]	52.14	57.86	55.00	61.64	66.48	64.06	-	-	-	-	-	-
BFG [55]	55.60	55.98	55.79	63.71	66.62	65.17	46.18	48.36	47.27	55.05	57.80	56.43
SCAT [56]	54.92	56.74	55.83	64.24	69.03	66.63	-	-	-	-	-	-
PEFC [42]	55.09	59.63	57.36	65.47	70.84	68.16	49.15	54.69	51.92	62.56	63.21	62.89
PAPFZS3D [11]	59.45	66.08	62.76	65.40	70.30	67.85	48.99	56.57	52.78	61.27	60.81	61.04
2CBR [12]	55.89	61.99	58.94	63.55	67.51	65.53	46.51	53.91	50.21	55.51	58.07	56.79
Seg-PN [40]	64.84	67.98	66.41	67.63	71.48	69.36	60.12	**63.22**	61.67	62.58	64.53	63.56
Ours	**68.73**	**70.61**	**69.67**	**72.12**	**72.72**	**72.42**	**62.28**	62.11	**62.19**	**65.17**	**66.10**	**65.64**
FT [53]	53.31	55.89	54.61	72.19	72.60	72.40	46.21	48.70	47.47	63.83	64.47	64.15
ProtoNet [9]	65.22	66.65	65.94	72.89	77.47	75.22	57.96	62.14	60.08	65.82	69.64	67.76
AttProtoNet [9]	67.53	68.33	67.93	75.79	78.97	77.41	59.31	63.72	61.55	68.59	71.96	70.30
MPTI [10]	68.65	67.97	68.32	74.16	75.44	74.80	61.37	63.87	62.64	68.20	65.38	66.81
AttMPTI [10]	69.94	71.75	70.85	76.29	80.25	78.31	62.24	66.01	64.16	70.90	72.44	71.68
Aug-FT-P [38]	70.87	74.11	72.51	81.54	81.35	81.45	66.17	67.27	66.72	75.59	75.99	75.79
Aug-FT-A [38]	71.51	74.35	72.94	80.81	81.68	81.25	65.34	67.40	66.38	75.59	73.91	74.76
CWT [54]	68.54	73.31	70.97	76.27	79.87	78.09	-	-	-	-	-	-
BFG [55]	71.47	71.78	71.62	77.83	79.97	78.91	63.18	65.19	64.20	71.01	73.26	72.15
SCAT [56]	70.90	72.40	71.66	78.23	81.68	79.97	-	-	-	-	-	-
PEFC [42]	71.04	74.71	72.90	79.13	82.93	81.07	65.91	70.71	68.35	76.97	77.46	77.22
PAPFZS3D [11]	74.57	79.58	77.12	79.08	82.56	80.85	65.76	72.26	69.09	75.98	75.63	75.81
2CBR [12]	71.70	76.54	74.17	77.71	80.60	79.18	63.49	70.05	66.85	71.39	73.47	72.44
Seg-PN [40]	78.67	80.94	79.81	80.69	83.37	81.91	75.09	**77.47**	76.29	76.98	78.44	77.72
Ours	**81.47**	**82.77**	**82.12**	**83.80**	**84.21**	**84.00**	**76.76**	76.63	**76.69**	**78.91**	**79.59**	**79.26**

entropy loss computation which can be expressed as,

$$\mathcal{L} = -\frac{1}{|\mathcal{Q}|}\frac{1}{M}\sum_{i=1}^{|\mathcal{Q}|}\sum_{j=1}^{M}\sum_{n=1}^{N}\mathbb{1}\{\boldsymbol{Y}_q[i,j]=n\}\log\left(\hat{\boldsymbol{Y}}_q[i,j,n]\right), \tag{14}$$

where the one-hot prediction $\hat{\boldsymbol{Y}}_q = \text{SoftMax}(\boldsymbol{Y}'_q)$ is the output of our model. We also update the parameters in the feature extractor with a small learning rate of 0.00005. The general learning rate for the SSM and DSM blocks is 0.001. For the BEAM module, we set the number of attention heads to 8 and the hyperparameter α to 0.99. To balance computation efficiency and performance, we configure the number of encoders and decoders in DSM to 3. The number of heads in all cross-attention units is set to 2, while for self-attention units, it is set to 4.

Each class has 100 prototype numbers. The dropout rate in DSM and SSM is set to 0, and LayerNorm [61] is applied to all normalization layers. The classifier is an MLP with the LeakyReLu activation function, with a threshold set to 0.2. To avoid overfitting, we additionally add a dropout layer with a 0.25 rate to the classifier. We train our model on an NVIDIA RTX 3090 GPU for 45,000 iterations, with a learning rate decay ratio of 0.5 and decay steps every 5,000 iterations. During training, we conduct the validation process every 2,000 iterations for efficiency. We evaluate the performance of our model using the mean Intersection over Union (m-IoU) [10] as the metric. The model with the best performance on the validation set and trained for more than 10,000 iterations is saved for subsequent evaluation. For testing, we select 100 unseen episodes and then add Gaussian noises to simulate practical disturbance.



(a) (b)

Fig. 6: Loss and accuracy curve for SDSimpoint during training and testing. The testing process is conducted every 1000 iterations.

C. Benchmark and Result Comparison

The experiments are conducted to benchmark the performance from various aspects. As suggested by [62], the pre-training can provide a better initialization for the feature extractor. This initialization is especially valuable for few-shot learning, as the limited availability of labeled data may not be sufficient to train the feature extractor effectively. [10] pre-trains the feature extractor with the same in-door scanning

TABLE II: ScanNet Semantic Segmentation Result Using m-IoU (Up,%) and Dice Coefficient (Down,%). S^i Means The Testing Dataset Is The i-th Split

Method	2-way						3-way					
	1-shot			5-shot			1-shot			5-shot		
	S^0	S^1	mean	S^0	S^1	mean	S^0	S^1	mean	S^0	S^1	mean
FT [53]	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 [9]	33.92	30.95	32.44	45.34	42.01	43.68	28.47	26.13	27.30	37.36	34.98	36.17
AttProtoNet [9]	37.99	34.67	36.33	52.18	46.89	49.54	32.08	28.96	30.52	44.49	39.45	41.97
MPTI [10]	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 [10]	42.55	40.83	41.69	54.00	50.32	52.16	35.23	30.72	32.98	46.74	40.80	43.77
Aug-FT-P [38]	48.49	43.06	45.78	58.97	54.35	56.66	38.49	35.07	36.78	50.95	45.69	48.32
Aug-FT-A [38]	42.89	40.78	41.84	54.35	51.39	52.87	35.47	30.55	33.01	47.11	41.80	44.46
CWT [54]	42.33	41.78	42.05	55.60	53.77	56.48	-	-	-	-	-	-
BFG [55]	42.15	40.52	41.34	51.23	49.39	50.31	34.12	31.98	33.05	46.25	41.38	43.82
SCAT [56]	54.92	56.74	55.83	64.24	69.03	66.63	-	-	-	-	-	-
PEFC [42]	45.31	44.86	45.09	56.26	54.06	55.16	38.78	36.13	37.46	51.72	46.05	48.89
PAPFZS3D [11]	57.08	55.94	56.51	64.55	59.64	62.10	55.27	55.60	55.44	59.02	53.16	56.09
2CBR [12]	50.73	47.66	49.20	52.35	47.14	49.75	47.00	46.36	46.68	45.06	39.47	42.27
Seg-PN [40]	63.15	64.32	63.74	67.08	**69.05**	68.07	61.80	**65.34**	63.57	62.94	**68.26**	65.60
Ours	**65.21**	**65.18**	**65.19**	**68.20**	68.49	**68.35**	**63.30**	63.86	**63.83**	**65.04**	66.27	**65.66**
FT [53]	47.97	44.89	46.45	59.86	54.27	57.12	38.70	32.07	35.46	51.77	43.87	47.93
ProtoNet [9]	50.66	47.27	48.99	62.39	59.16	60.80	44.32	41.43	42.89	54.40	51.83	53.12
AttProtoNet [9]	55.06	51.49	53.30	68.58	63.84	66.26	48.58	44.91	46.77	61.58	56.58	59.13
MPTI [10]	56.39	53.09	54.77	63.85	60.71	62.31	46.11	42.84	44.49	55.22	51.15	53.21
AttMPTI [10]	59.70	57.98	58.85	70.13	66.95	68.56	52.10	47.00	49.60	63.70	57.95	60.89
Aug-FT-P [38]	65.31	60.20	62.81	74.19	70.42	72.33	55.59	51.93	53.78	67.51	62.72	65.16
Aug-FT-A [38]	60.03	57.93	59.00	70.42	67.89	69.17	52.37	46.80	49.64	64.05	58.96	61.55
CWT [54]	59.48	58.94	59.20	71.47	69.94	72.19	-	-	-	-	-	-
BFG [55]	59.30	57.67	58.50	67.75	66.12	66.94	50.88	48.46	49.68	63.25	58.54	60.94
SCAT [56]	70.90	72.40	71.66	78.23	81.68	79.97	-	-	-	-	-	-
PEFC [42]	62.36	61.94	62.15	72.01	70.18	71.10	55.89	53.08	54.50	68.18	63.06	65.67
PAPFZS3D [11]	72.68	71.75	72.21	78.46	74.72	76.62	71.19	71.47	71.33	74.23	69.42	71.87
2CBR [12]	67.31	64.55	65.95	68.72	64.08	66.44	63.95	63.35	63.65	62.13	56.60	59.42
Seg-PN [40]	77.41	78.29	77.86	80.30	**81.69**	81.00	76.39	**79.04**	77.73	77.26	**81.14**	79.23
Ours	**78.94**	**78.92**	**78.93**	**81.09**	81.30	**81.20**	**77.53**	77.94	**77.92**	**78.82**	79.71	**79.27**

point cloud datasets as that in training, however, in practical application, the training dataset might not be large and complete enough for pre-training. Therefore, it is valuable to investigate the situation of using another dataset for pre-training while adopting our target dataset for training. According to [63], the two datasets used in [10] can be treated as in the same domain because they are both generated from real-world in-door scanning point clouds. Nonetheless, in some cases, it might be difficult to find an appropriate pre-training dataset within the same domain. Hence, we can explore datasets in other domains for pre-training, which is significant for real-world applications. Figure 6 tells the visualized loss and accuracy curve during training and testing. The loss value gradually leveled out in training and testing. Looking at it in combination with the accuracy curve, the testing accuracy does not show a significant decline as the training loss decreases, and the testing loss remains stable. We can conclude that our model has not experienced significant overfitting under the mentioned experimental setup.

In light of these considerations, we undertake experiments in three distinct settings: 1) Same-Domain Same Pre-training: This paradigm aligns with the approach outlined in [10]. 2) Same-Domain Cross Pre-training: In this scenario, the pre-training dataset differs from the training dataset, yet they both belong to the same domain. 3) Cross-Domain Cross Pre-training: Here, the pre-training dataset and the training dataset originate from distinct domains. This classification allows us to comprehensively explore the impact of different pre-training strategies on our experiments.

1) Same-Domain Same Pre-training: We conduct our SD-SimPoint pipeline on S3DIS and ScanNet, respectively, where the pre-training and training datasets are the same as baseline [10]. The results of our method compared to the baselines are summarized in Table I and Table II. With a larger shot number, which means more labeled samples, there are significant improvements in all methods. It has been noticed that the difficulty level of 3-way segmentation is higher than that of 2-way segmentation, which results in its generally lower performance. However, compared to other models, our method has a lower performance when changing the task from 2-way to 3-way. Such an improvement can be attributed to the "similarity-drive" learning of our method. Since the similarity between queries and prototypes has already been captured effectively during the forward process, we can directly predict the affiliation of each query point rather than constructing a graph with individual features [10] for prediction, which is easier to be influenced by noise.This not only proved the robustness of our method but also illustrated the similarity-based learning framework is effective for the few-shot semantic segmentation task. Fig. 4 and Fig. 5 illustrate our semantic segmentation result on S3DIS and ScanNet. The attMPTI [10] is adopted as the baseline, from the figures, it is evident that our method outperforms the baseline.

TABLE III: S3DIS Semantic Segmentation Result Using ScanNet For Pre-training.

Method	2-way						3-way					
	1-shot			5-shot			1-shot			5-shot		
	S^0	S^1	mean	S^0	S^1	mean	S^0	S^1	mean	S^0	S^1	mean
FT [53]	37.05	38.33	37.69	51.82	50.01	50.92	31.62	31.99	31.81	43.84	47.50	45.67
ProtoNet [9]	44.71	48.80	46.76	50.06	52.59	51.33	40.03	41.38	40.71	44.64	48.62	46.62
AttProtoNet [9]	49.96	50.01	49.99	53.35	55.61	54.48	37.28	44.69	40.99	42.33	52.35	47.34
MPTI [10]	48.27	47.56	47.92	55.21	52.56	53.89	36.80	37.71	37.23	39.09	41.73	40.41
AttMPTI [10]	51.14	48.90	50.02	57.80	62.61	60.20	39.80	42.06	40.93	43.25	51.68	47.47
PAPFZS3D [11]	58.65	64.91	61.78	64.72	69.33	67.03	48.52	56.05	52.29	61.11	59.54	60.33
Ours	**65.52**	**67.95**	**66.74**	**69.80**	**71.61**	**70.70**	**61.59**	**61.73**	**61.65**	**63.30**	**67.91**	**65.61**

TABLE IV: ScanNet Semantic Segmentation Result Using S3DIS For Pre-training.

Method	2-way						3-way					
	1-shot			5-shot			1-shot			5-shot		
	S^0	S^1	mean	S^0	S^1	mean	S^0	S^1	mean	S^0	S^1	mean
FT [53]	28.45	29.72	29.09	30.69	34.28	32.49	24.70	19.40	22.05	32.81	29.58	31.20
ProtoNet [9]	36.98	34.04	36.00	40.79	40.21	40.50	30.65	28.17	29.41	33.70	36.00	34.85
AttProtoNet [9]	37.04	33.85	35.45	47.30	41.82	44.56	32.00	28.45	30.23	39.71	35.40	37.56
MPTI [10]	35.81	33.69	34.75	42.67	38.11	40.39	30.53	27.49	29.01	33.80	31.94	32.87
AttMPTI [10]	35.36	34.28	34.82	45.19	38.80	42.00	28.96	25.33	27.15	35.14	30.96	33.05
PAPFZS3D [11]	56.37	55.05	55.71	62.80	59.33	61.07	52.41	53.72	53.06	58.10	52.57	55.34
Ours	**64.37**	**64.58**	**64.48**	**65.46**	**66.08**	**65.77**	**60.96**	**62.33**	**61.65**	**63.62**	**65.36**	**64.49**

TABLE V: Semantic Segmentation Result Using ShapeNetPart For Pre-training.

Dataset	Method	2-way						3-way					
		1-shot			5-shot			1-shot			5-shot		
		S^0	S^1	mean	S^0	S^1	mean	S^0	S^1	mean	S^0	S^1	mean
S3DIS	FT [53]	25.33	30.60	27.97	30.92	36.19	33.56	15.05	20.68	17.87	18.89	21.09	19.99
	ProtoNet [9]	39.16	39.00	39.08	45.72	47.39	46.56	30.84	32.11	31.48	40.23	41.70	40.97
	AttProtoNet [9]	42.25	39.70	40.98	51.48	51.71	51.60	35.62	36.10	35.86	42.50	43.54	43.02
	MPTI [10]	40.12	41.53	40.83	45.80	46.97	46.39	32.04	31.66	31.85	39.54	40.10	39.82
	AttMPTI [10]	43.45	44.91	44.18	49.77	53.81	51.79	33.90	32.11	33.01	46.94	43.90	45.42
	PAPFZS3D [11]	52.90	57.36	55.13	60.27	63.15	61.71	46.36	55.70	51.03	55.82	56.43	56.13
	Ours	**60.93**	**65.17**	**63.05**	**64.37**	**66.06**	**65.22**	**61.25**	**63.53**	**62.39**	**61.43**	**64.49**	**62.96**
ScanNet	FT [53]	12.39	12.04	12.22	13.57	13.02	13.30	8.56	9.43	9.00	10.17	11.48	10.83
	ProtoNet [9]	22.16	18.52	20.34	37.98	39.05	38.52	17.44	16.31	16.88	32.69	30.12	31.41
	AttProtoNet [9]	27.43	19.97	23.70	39.60	40.17	39.89	19.95	18.84	19.40	35.89	35.00	35.45
	MPTI [10]	20.78	19.53	20.16	39.85	39.02	39.44	16.26	19.48	17.87	32.72	29.91	31.32
	AttMPTI [10]	22.45	21.61	22.03	40.03	39.80	39.92	17.35	18.58	17.97	33.25	30.37	31.81
	PAPFZS3D [11]	49.28	50.14	49.71	52.06	52.50	52.28	45.83	44.79	45.31	49.46	48.08	48.77
	Ours	**54.94**	**53.20**	**54.07**	**55.39**	**54.25**	**55.30**	**50.47**	**50.75**	**50.61**	**51.19**	**52.38**	**51.79**

TABLE VI: Model Complexity and Inference Time

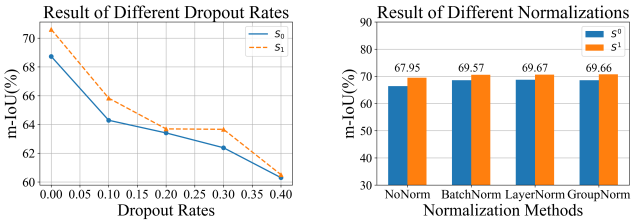
ID	Model	#Params.	Flops	Time (sec/block)
I	AttMPTI [10]	352.19K	7.12G	0.46
II	CWT [54]	685.64k	8.16G	-
III	PAPFZS3D [11]	2.48M	7.48G	0.53
IV	Ours	2.76M	8.37G	0.61

2) Same-Domain Cross Pre-training: Table III and Table IV illustrate the "cross pre-training" results, which means using one of the S3DIS and ScanNet datasets for pre-training and utilize the other for training. These settings are more practical because, in real-world semantic segmentation tasks, it is almost impossible for us to find the same annotated dataset to pre-train our model, while we can utilize a public dataset from the same domain to pre-train our model. We can tell from the tables that all the models have decreased performance compared to the same-domain same pre-training settings. This is because of the distribution shift between datasets. Although our model performs slightly worse in this scenario, it still outperforms other baselines, which indicates the superiority of our SDSimPoint in such a situation.

3) Cross-Domain Cross Pre-training: Additionally, we argue that since our method emphasizes similarity rather than specific class affiliation, it is possible to use datasets from other domains for pre-training as long as our feature extractor can learn the similarity. Therefore, we explore the performance of our method with ShapeNetPart [64], a widely adopted dataset for part segmentation tasks, which encompasses 16,846 point clouds across 50 semantic classes. Given that the ShapeNetPart dataset includes some objects that are also present in indoor scanning scenes, such as chairs, tables, and laptops, we propose the feasibility of pre-training our feature extractor

TABLE VII: Ablation Study On S3DIS S^0 With 2-way 1-shot

ID	BEAM	SA	DSM	SSM	Deep Fusion	Result
I						52.27
II		✓				53.77
III	✓					57.86
IV			✓			56.97
V			✓		✓	59.70
VI				✓		55.93
VII			✓	✓	✓	61.45
VIII		✓	✓			60.73
IX		✓	✓		✓	62.35
X		✓		✓		54.64
XI		✓	✓	✓	✓	62.65
XII	✓		✓			63.18
XIII	✓		✓		✓	65.35
XIV	✓			✓		64.49
XV	✓		✓	✓	✓	**68.73**



(a) (b)

Fig. 7: The effectiveness of (a) different dropout rates, (b) different normalization methods on the S3DIS dataset with 2-way 1-shot. The blue line and blue bars denote the split 0, and the orange dash line and orange bars denote the split 1. The numbers above the colored bar represent the mean value of the corresponding experiment group.

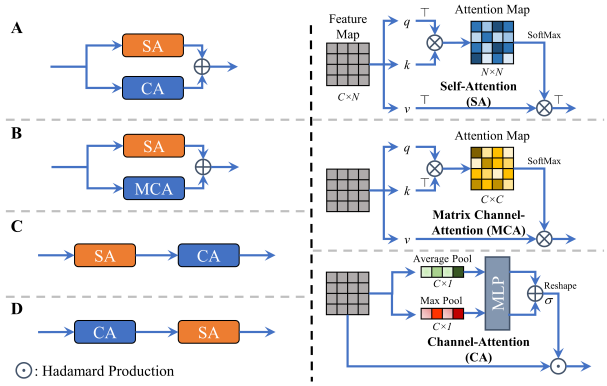


Fig. 8: Structure of different attention modules: CA represents the channel-attention, SA is the self-attention, and MCA indicates the matrix channel-attention. $\top$ in the figure is matrix transportation, σ is the Sigmoid function.

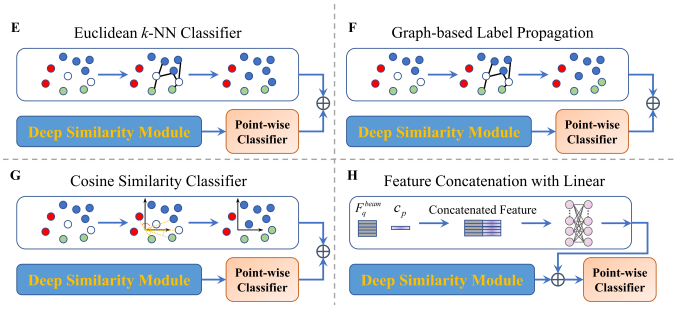


Fig. 9: Visualization of various forms of the SSM: The blue, red, and green circles represent prototypes of three different classes, while the unfilled circles represent query features awaiting labeling.

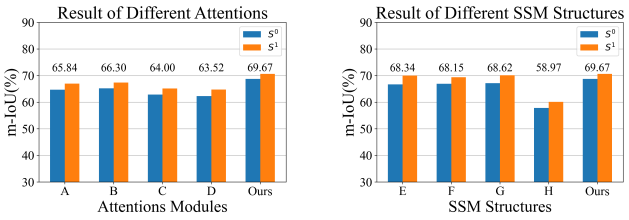
on ShapeNetPart and subsequently training our model on S3DIS and ScanNet datasets, respectively. Tables V presents the segmentation results. As shown in these tables, while the results are lower than those achieved by models pre-trained with datasets in the same domain (in-door scanning point clouds), our model outperforms the baseline models in these specific settings in terms of segmentation performance. Table VI illustrates the complexity and inference time of our model with the comparison of other methods. Due to our SSM and DSM being based on transformer structures, the parameters amount, the FLOPs, and the cost of time are larger than others, but these are acceptable scarification considering the improvement of performance.

D. Ablation Study

We initiate the ablation studies by conducting experiments on the S3DIS dataset to evaluate the effectiveness of different components within our SDSimPoint. Given the frequent use of attention-based blocks in our SSM and DSM, we delve deeper into the study of their dropout rates and normalization methods. To substantiate the advantage of our BEAM, we also compare its performance against other popular attention forms. Finally, we argue that the structure of the SSM can be flexible as long as it can capture sufficient shallow similarity. As such, we undertake experiments using SSMs with varying structures.

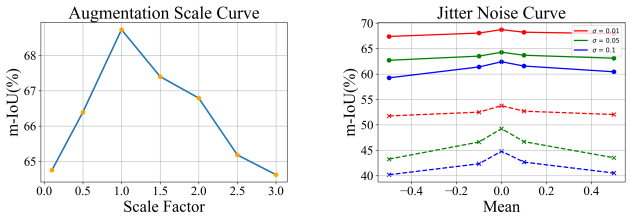
1) Model Components: To investigate the effectiveness of each component in our method, different settings of our model are selected. The BEAM block and the Self-Attention (SA) block will not be adopted at the same time since they share the same function. The deep fusion mechanism is not available when the DSM is not utilized. Based on these preconditions, an ablation study is conducted, and the results are shown in Table VII. Experiment I shows the result when disabling all modules of our network with the final prediction process is conducted as attMPTI [10]. Experiment II illustrates the performance when adding a self-attention for further feature refinement. As discussed in previous sections, the BEAM has better learning ability on the dataset scale, Experiment III tells that it can lead to a notable improvement if we adopt BEAM rather than Self-Attention. Experiment IV and Experiment VI illustrate the effectiveness of the DSM and SSM, they outperform the attMPTI without any help of attention mechanism. Experiment V indicates the importance of the deep fusion, it helps the DSM to better capture the similarity between the query samples and support samples.

2) Dropout Rates and Normalization Methods: Fig. 7 illustrates the results of different settings on dropout rates



(a) (b)

Fig. 10: Ablation studies for attention modules and SSM structures on S3DIS with 2-way 1-shot setting. The numbers above the colored bar represent the mean value of the corresponding experiment group.



(a) (b)

Fig. 11: Ablation study for different intensities of (a) scaling augmentation and (b) Gaussian jitter noise. The solid line in (b) denotes our method, while the dashed line in (b) represents the baseline.

TABLE VIII: Model Performance Under Different Hyper-parameters Settings on S3DIS S^0 in 2-way 1-shot.

Hyperparameter	Value	mIoU
Learning rate	5e-4, 1e-3, 5e-3	67.64, 68.73, 66.96
Decoder block number	2, 3, 4	68.50, 68.73, 68.76
Backbone k-NN number	10, 20, 30	67.41, 68.73, 68.82
Augmentation epochs	50, 60, 70	68.61, 68.73, 68.55
Pre-train batch size	8, 16, 32	67.22, 68.73, 68.81
EMA weight α	0.9, 0.99, 0.999	68.50, 68.73, 68.49
Prototype numbers	50, 100, 150	67.08, 68.73, 68.30

TABLE IX: Ablation Study of Different Distance Metric On S3DIS Dataset

Distance Metric	2-way 1-shot			3-way 1-shot		
	S^0	S^1	mean	S^0	S^1	mean
Euclidean	44.18	42.53	43.34	39.71	40.83	40.27
Cosine	56.42	55.07	55.75	51.61	51.49	51.55
Ours	**68.73**	**70.61**	**69.67**	**62.28**	**62.11**	**62.19**

and normalization methods. Fig. 7(a) indicates that with the increase in the dropout rate, the segmentation performance will be worse. This is because the data is relatively scarce in few-shot learning tasks, if the model has a large dropout rate, it is difficult for it to learn sufficient knowledge. From Fig. 7(b), we can tell that there is no huge difference between models with various normalization methods, and all of them are better than the model without any normalization method.

3) Attention Modules: To additionally verify the advantage of our BEAM, Fig. 8 illustrate some famous attention mechanisms including the dual attention [65], CBAM [66] and their combinations. We substitute our BEAM with an attention module whose form is A, B, C, D in Fig. 8 respectively, then conduct experiments on S3DIS with a 2-way 1-shot few-shot setting. Fig. 10(a) depicts the result comparison of different attention modules. The histogram shows that our BEAM excels other popular attention modules.

4) SSM Structures: Fig. 9 E, F, G, H illustrate possible forms of the SSM, the Euclidean k-NN classifier computes the k-nearest neighbors of the query sample and directly assigns the label of the class with most neighbors and the minimal average distance. The graph-base label propagation firstly constructs a graph with the query sample and its k-nearest neighbors, then applies the label propagation algorithm [67] to obtain the possibility for classification. The feature concatenation concatenates the query feature with the prototype feature of each class and then adopts a linear layer to obtain the similarity. Since some non-parametric algorithms in Fig. 9 directly output the one-hot prediction vector, we add the vector with the output from the point-wise classifier that deals with the features from DSM, then adopt an extra SoftMax operation to get the final prediction. Fig. 10(b) indicates the result of SSM different with structures, the numerical result illustrates that a non-parametric SSM also can perform well in our framework while the rough feature concatenation is not able to learn the shallow similarity well.

5) Noise intensity: To investigate the influence of augmentation intensity during testing, we categorize the noises into the following types: 1) scaling: scaling the point cloud with a certain number, and 2) jittering: adding Gaussian noise to each point. We vary the scaling factor from 0.1 to 3 and change the Gaussian noise's mean and derivation. Fig. 10(a) illustrates the result with different scaling factors. We can see that too small or too large scaling leads to decreased performance. This is because the scaled point cloud might confuse our model when grasping the global semantic information. Fig. 10(b) shows the performance under different intensities of noise. It can be concluded that with a larger derivation, the performance will decrease more. Compared to the baseline, our method is more robust, facing the change of mean.

6) Hyper-parameters analysis: Table VIII illustrates the influence of different hyper-parameter settings. A too-large or too-small learning rate will downgrade the model's performance due to an inappropriate gradient. Although more decoder blocks may improve the performance, it requires more training resources and suffers from the risk of overfitting. The k-NN number of the backbone controls the "receptive field", with a larger number, the model can grab more information, but the inference and training speed will be significantly slow down. The augmentation epochs of the dynamic augmentor during pre-training control the diversity of augmentation patterns, more epochs will bring more augmentation strategies but confuse the backbone, and fewer epochs might not be enough to learn a robust representation. The weight factor of the EMA process in BEAM and the prototype numbers influence the quality of features, which implicitly affect the captured similarity. With relative experiments, we find that 0.99 and 100 are suitable values considering both accuracy

and efficiency.

7) Distance metric: To investigate the influence of different distance metrics, we compare two distance functions with our learnable metric, the Euclidean distance and the cosine similarity. Table IX illustrates the comparison result. Since our features are fused with similarity, the non-parametric distance metric, like Euclidean distance, is not appropriate for prediction since it focuses on figuring out the distance between features rather than directly predicting affiliation from similarity. On the contrary, our learnable distance metric conducts point-wise classification from the similarity, which is more suitable for our framework.

V. Conclusion

In this work, we propose SDSimPoint, a noteworthy few-shot point cloud semantic segmentation network based on shallow-deep similarity learning. We design a BEAM that uses two memory units to replace the matrix multiplication in the attention mechanism, thus enabling the range of the attention map to expand from a single episode to the whole training dataset. We introduce an SSM and a DSM to capture both shallow and deep similarities. This helps our model to acquire sufficient information, even when pre-trained in a class-agnostic self-supervision manner. A query-prototype deep fusion mechanism has been devised for DSM with the objective of facilitating interactions between the query feature and class prototypes. Thus, allows for the fusion of query features with prototype information, ultimately enhancing prediction results. Comprehensive experiments are implemented to verify the effectiveness and robustness of the proposed method.

Acknowledgment

This research is supported by the National University of Singapore under the College of Design and Engineering Industry-focused Ring-Fenced Ph.D. Scholarship program. It is also supported by A*STAR under its "RIE2025 IAF-PP Advanced ROS2-native Platform Technologies for Cross sectorial Robotics Adoption (M21K1a0104)" program. Additionally, the authors would like to acknowledge valuable discussions with Dr. Hendrik Schafstall and Dr. Arno Zinke during their tenure previously at Hexagon, Manufacturing Intelligence Division, Simufact Engineering GmbH.

References

[1] Y. Li, L. Ma, Z. Zhong, F. Liu, M. A. Chapman, D. Cao, and J. Li, "Deep learning for lidar point clouds in autonomous driving: A review," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 32, no. 8, pp. 3412–3432, 2021.

[2] C. R. Qi, H. Su, K. Mo, and L. J. Guibas, "Pointnet: Deep learning on point sets for 3d classification and segmentation," in *Proceedings of the IEEE/CVF conference on Computer Vision and Pattern Recognition (CVPR)*, 2017, pp. 652–660.

[3] C. R. Qi, L. Yi, H. Su, and L. J. Guibas, "Pointnet++: Deep hierarchical feature learning on point sets in a metric space," *Advances in Neural Information Processing Systems (NeruIPS)*, vol. 30, 2017.

[4] S. Li, Y. Liu, and J. Gall, "Rethinking 3-d lidar point cloud segmentation," *IEEE Transactions on Neural Networks and Learning Systems*, pp. 1–12, 2021.

[5] Z. Du, H. Ye, and F. Cao, "A novel local-global graph convolutional method for point cloud semantic segmentation," *IEEE Transactions on Neural Networks and Learning Systems*, pp. 1–15, 2022.

[6] B. Liu, J. Jiao, and Q. Ye, "Harmonic feature activation for few-shot semantic segmentation," *IEEE Transactions on Image Processing*, vol. 30, pp. 3142–3153, 2021.

[7] B. Yang, F. Wan, C. Liu, B. Li, X. Ji, and Q. Ye, "Part-based semantic transform for few-shot semantic segmentation," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 33, no. 12, pp. 7141–7152, 2022.

[8] H. Gao, J. Xiao, Y. Yin, T. Liu, and J. Shi, "A mutually supervised graph attention network for few-shot segmentation: The perspective of fully utilizing limited samples," *IEEE Transactions on Neural Networks and Learning Systems*, pp. 1–13, 2022.

[9] J. Snell, K. Swersky, and R. Zemel, "Prototypical networks for few-shot learning," *Advances in Neural Information Processing Systems (NeurIPS)*, vol. 30, 2017.

[10] 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 (CVPR)*, 2021, pp. 8873–8882.

[11] S. He, X. Jiang, W. Jiang, and H. Ding, "Prototype adaption and projection for few- and zero-shot 3d point cloud semantic segmentation," *IEEE Transactions on Image Processing*, vol. 32, pp. 3199–3211, 2023.

[12] G. Zhu, Y. Zhou, R. Yao, and H. Zhu, "Cross-class bias rectification for point cloud few-shot segmentation," *IEEE Transactions on Multimedia*, vol. 25, pp. 9175–9188, 2023.

[13] Z. An, G. Sun, Y. Liu, F. Liu, Z. Wu, D. Wang, L. Van Gool, and S. Belongie, "Rethinking few-shot 3d point cloud semantic segmentation," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2024, pp. 3996–4006.

[14] J. Wang, H. Zhu, H. Guo, A. A. Mamun, C. Xiang, and T. H. Lee, "Few-shot point cloud semantic segmentation via contrastive self-supervision and multi-resolution attention," in *2023 IEEE International Conference on Robotics and Automation (ICRA)*, 2023, pp. 2811–2817.

[15] Z. Li, Z. Hu, W. Luo, and X. Hu, "Sabernet: Self-attention based effective relation network for few-shot learning," *Pattern Recognition*, vol. 133, p. 109024, 2023.

[16] O. K. Shirekar, A. Singh, and H. Jamali-Rad, "Self-attention message passing for contrastive few-shot learning," in *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*, January 2023, pp. 5426–5436.

[17] Z. Chen, Y. Fu, K. Chen, and Y.-G. Jiang, "Image block augmentation for one-shot learning," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 33, no. 01, 2019, pp. 3379–3386.

[18] L. Zhang, S. Zhang, G. Xie, J. Liu, H. Yan, J. Wang, F. Zheng, and Y. Jin, "What makes a good data augmentation for few-shot unsupervised image anomaly detection?" in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2023, pp. 4344–4353.

[19] A. Santoro, S. Bartunov, M. Botvinick, D. Wierstra, and T. Lillicrap, "Meta-learning with memory-augmented neural networks," in *Proceedings of the 33rd International Conference on International Conference on Machine Learning - Volume 48*, ser. ICML'16. JMLR.org, 2016, p. 1842–1850.

[20] N. Mishra, M. Rohaninejad, X. Chen, and P. Abbeel, "A simple neural attentive meta-learner," in *International Conference on Learning Representations (ICLR)*, 2018.

[21] S. Rahman, S. Khan, and F. Porikli, "A unified approach for conventional zero-shot, generalized zero-shot, and few-shot learning," *IEEE Transactions on Image Processing*, vol. 27, no. 11, pp. 5652–5667, 2018.

[22] A. A. Rusu, D. Rao, J. Sygnowski, O. Vinyals, R. Pascanu, S. Osindero, and R. Hadsell, "Meta-learning with latent embedding optimization," in *International Conference on Learning Representations (ICLR)*, 2019.

[23] A. Zhmoginov, M. Sandler, and M. Vladymyrov, "Hypertransformer: Model generation for supervised and semi-supervised few-shot learning," in *International Conference on Machine Learning (ICML)*. PMLR, 2022, pp. 27 075–27 098.

[24] C. Finn, P. Abbeel, and S. Levine, "Model-agnostic meta-learning for fast adaptation of deep networks," in *Proceedings of the 34th International Conference on Machine Learning - Volume 70*, ser. ICML'17. JMLR.org, 2017, p. 1126–1135.

[25] Z. Ji, Z. Hou, X. Liu, Y. Pang, and J. Han, "Information symmetry matters: A modal-alternating propagation network for few-shot learning," *IEEE Transactions on Image Processing*, vol. 31, pp. 1520–1531, 2022.

[26] B. Xi, J. Li, Y. Li, R. Song, D. Hong, and J. Chanussot, "Few-shot learning with class-covariance metric for hyperspectral image classification," *IEEE Transactions on Image Processing*, vol. 31, pp. 5079–5092, 2022.

[27] J. Zhou, Y. Zheng, J. Tang, L. Jian, and Z. Yang, "Flipda: Effective and robust data augmentation for few-shot learning," in *Proceedings of the*

60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), 2022, pp. 8646–8665.

[28] Y. He, R. Ge, X. Qi, Y. Chen, J. Wu, J.-L. Coatrieux, G. Yang, and S. Li, "Learning better registration to learn better few-shot medical image segmentation: Authenticity, diversity, and robustness," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 35, no. 2, pp. 2588–2601, 2024.

[29] H. Cheng, Y. Wang, H. Li, A. C. Kot, and B. Wen, "Disentangled feature representation for few-shot image classification," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 35, no. 8, pp. 10 422–10 435, 2024.

[30] Z. Zhou, L. Luo, S. Zhou, W. Li, X. Yang, X. Liu, and E. Zhu, "Task-related saliency for few-shot image classification," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 35, no. 8, pp. 10 751–10 763, 2024.

[31] Y. Liu, L. Zhu, X. Wang, M. Yamada, and Y. Yang, "Bilaterally normalized scale-consistent sinkhorn distance for few-shot image classification," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 35, no. 8, pp. 11 475–11 485, 2024.

[32] 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)*, 2018, pp. 4558–4567.

[33] X. Ma, C. Qin, H. You, H. Ran, and Y. Fu, "Rethinking network design and local geometry in point cloud: A simple residual MLP framework," in *International Conference on Learning Representations*, 2022.

[34] G. Qian, Y. Li, H. Peng, J. Mai, H. Hammoud, M. Elhoseiny, and B. Ghanem, "Pointnext: Revisiting pointnet++ with improved training and scaling strategies," *Advances in Neural Information Processing Systems*, vol. 35, pp. 23 192–23 204, 2022.

[35] 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 (ICCV)*, October 2021, pp. 16 259–16 268.

[36] X. Lai, J. Liu, L. Jiang, L. Wang, H. Zhao, S. Liu, X. Qi, and J. Jia, "Stratified transformer for 3d point cloud segmentation," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2022, pp. 8500–8509.

[37] X. Wu, L. Jiang, P.-S. Wang, Z. Liu, X. Liu, Y. Qiao, W. Ouyang, T. He, and H. Zhao, "Point transformer v3: Simpler, faster, stronger," *arXiv preprint arXiv:2312.10035*, 2023.

[38] L. Lai, J. Chen, C. Zhang, Z. Zhang, G. Lin, and Q. Wu, "Tackling background ambiguities in multi-class few-shot point cloud semantic segmentation," *Knowledge-Based Systems*, vol. 253, p. 109508, 2022.

[39] G. Zhang, G. Kang, Y. Yang, and Y. Wei, "Few-shot segmentation via cycle-consistent transformer," *Advances in Neural Information Processing Systems (NeurIPS)*, vol. 34, pp. 21 984–21 996, 2021.

[40] X. Zhu, R. Zhang, B. He, Z. Guo, J. Liu, H. Xiao, C. Fu, H. Dong, and P. Gao, "No time to train: Empowering non-parametric networks for few-shot 3d scene segmentation," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2024, pp. 3838–3847.

[41] C. Zhang, Z. Wu, X. Wu, Z. Zhao, and S. Wang, "Few-shot 3d point cloud semantic segmentation via stratified class-specific attention based transformer network," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 37, 2023, pp. 3410–3417.

[42] Q. Zhang, T. Wang, F. Hao, F. Wu, and J. Cheng, "Prototype expansion and feature calibration for few-shot point cloud semantic segmentation," *Neurocomputing*, vol. 558, p. 126732, 2023.

[43] J. Wang, H. Zhu, H. Guo, A. Al Mamun, P. Vadakkepat, and T. H. Lee, "Cam/cad point cloud part segmentation via few-shot learning," in *2022 IEEE 20th International Conference on Industrial Informatics (INDIN)*. IEEE, 2022, pp. 359–365.

[44] J. Wang, H. Zhu, H. Guo, A. Al Mamun, C. Silva, and T. H. Lee, "Few-shot point cloud semantic segmentation for cam/cad via feature enhancement and efficient dual attention," in *accepted to the 49th Annual Conference of the IEEE Industrial Electronics Society (IECON)*. IEEE, 2023.

[45] A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, S. Gelly, J. Uszkoreit, and N. Houlsby, "An image is worth 16x16 words: Transformers for image recognition at scale," in *International Conference on Learning Representations (ICLR)*, 2021.

[46] H. Zhao, J. Jia, and V. Koltun, "Exploring self-attention for image recognition," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020, pp. 10 076–10 085.

[47] M.-H. Guo, Z.-N. Liu, T.-J. Mu, and S.-M. Hu, "Beyond self-attention: External attention using two linear layers for visual tasks," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 45, no. 5, pp. 5436–5447, 2023.

[48] L. Wang, X. Li, and Y. Fang, "Few-shot learning of part-specific probability space for 3d shape segmentation," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020, pp. 4504–4513.

[49] V. G. Satorras and J. B. Estrach, "Few-shot learning with graph neural networks," in *International conference on learning representations (ICLR)*, 2018.

[50] L. H. Li, P. Zhang, H. Zhang, J. Yang, C. Li, Y. Zhong, L. Wang, L. Yuan, L. Zhang, J.-N. Hwang, K.-W. Chang, and J. Gao, "Grounded language-image pre-training," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2022, pp. 10 965–10 975.

[51] H. Zhang, P. Zhang, X. Hu, Y.-C. Chen, L. Li, X. Dai, L. Wang, L. Yuan, J.-N. Hwang, and J. Gao, "Glipv2: Unifying localization and vision-language understanding," *Advances in Neural Information Processing Systems (NeurIPS)*, vol. 35, pp. 36 067–36 080, 2022.

[52] Z.-Y. Dou, A. Kamath, Z. Gan, P. Zhang, J. Wang, L. Li, Z. Liu, C. Liu, Y. LeCun, N. Peng, J. Gao, and L. Wang, "Coarse-to-fine vision-language pre-training with fusion in the backbone," in *Advances in Neural Information Processing Systems (NeurIPS)*, 2022.

[53] S. Amirreza, B. Shray, L. Zhen, E. Irfan, and B. Byron, "One-shot learning for semantic segmentation," in *Proceedings of the British Machine Vision Conference (BMVC)*, G. B. Tae-Kyun Kim, Stefanos Zafeiriou and K. Mikolajczyk, Eds. BMVA Press, September 2017, pp. 167.1–167.13.

[54] Z. Lu, S. He, X. Zhu, L. Zhang, Y.-Z. Song, and T. Xiang, "Simpler is better: Few-shot semantic segmentation with classifier weight transformer," in *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, 2021, pp. 8741–8750.

[55] Y. Mao, Z. Guo, L. Xiaonan, Z. Yuan, and H. Guo, "Bidirectional feature globalization for few-shot semantic segmentation of 3d point cloud scenes," in *2022 International Conference on 3D Vision (3DV)*. IEEE, 2022, pp. 505–514.

[56] C. Zhang, Z. Wu, X. Wu, Z. Zhao, and S. Wang, "Few-shot 3d point cloud semantic segmentation via stratified class-specific attention based transformer network," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 37, no. 3, 2023, pp. 3410–3417.

[57] I. Armeni, A. Sax, A. R. Zamir, and S. Savarese, "Joint 2D-3D-Semantic Data for Indoor Scene Understanding," *ArXiv e-prints*, Feb. 2017.

[58] A. Dai, A. X. Chang, M. Savva, M. Halber, T. Funkhouser, and M. Nießner, "Scannet: Richly-annotated 3d reconstructions of indoor scenes," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.

[59] 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)*, vol. 38, no. 5, pp. 1–12, 2019.

[60] T. Wu, L. Pan, J. Zhang, T. Wang, Z. Liu, and D. Lin, "Density-aware chamfer distance as a comprehensive metric for point cloud completion," in *In Advances in Neural Information Processing Systems (NeurIPS), 2021*, 2021.

[61] J. L. Ba, J. R. Kiros, and G. E. Hinton, "Layer normalization," *stat*, vol. 1050, p. 21, 2016.

[62] A. Newell and J. Deng, "How useful is self-supervised pretraining for visual tasks?" in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020, pp. 7345–7354.

[63] M. Wang and W. Deng, "Deep visual domain adaptation: A survey," *Neurocomputing*, vol. 312, pp. 135–153, 2018.

[64] L. Yi, V. G. Kim, D. Ceylan, I.-C. Shen, M. Yan, H. Su, C. Lu, Q. Huang, A. Sheffer, and L. Guibas, "A scalable active framework for region annotation in 3d shape collections," *ACM Transactions on Graphics (ToG)*, vol. 35, no. 6, pp. 1–12, 2016.

[65] J. Fu, J. Liu, H. Tian, Y. Li, Y. Bao, Z. Fang, and H. Lu, "Dual attention network for scene segmentation," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019, pp. 3146–3154.

[66] S. Woo, J. Park, J.-Y. Lee, and I. S. Kweon, "Cbam: Convolutional block attention module," in *Proceedings of the European Conference on Computer Vision (ECCV)*, 2018, pp. 3–19.

[67] 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.