Computers, Materials & Continua

Tech Science Press

DOI: 10.32604/cmc.2023.045853

ARTICLE

Part-Whole Relational Few-Shot 3D Point Cloud Semantic Segmentation

Shoukun Xu[1], Lujun Zhang[1], Guangqi Jiang[1], Yining Hua[2] and Yi Liu[1,*]

[1]School of Computer Science and Artificial Intelligence, Aliyun School of Big Data and School of Software, Changzhou University, Changzhou, 213164, China

[2]Department of Computer Science, University of Aberdeen, King's College, Aberdeen, AB24 3FX, UK

*Corresponding Author: Yi Liu. Email: liuyi0089@gmail.com

Received: 09 September 2023 Accepted: 27 November 2023 Published: 26 March 2024

ABSTRACT

This paper focuses on the task of few-shot 3D point cloud semantic segmentation. Despite some progress, this task still encounters many issues due to the insufficient samples given, e.g., incomplete object segmentation and inaccurate semantic discrimination. To tackle these issues, we first leverage part-whole relationships into the task of 3D point cloud semantic segmentation to capture semantic integrity, which is empowered by the dynamic capsule routing with the module of 3D Capsule Networks (CapsNets) in the embedding network. Concretely, the dynamic routing amalgamates geometric information of the 3D point cloud data to construct higher-level feature representations, which capture the relationships between object parts and their wholes. Secondly, we designed a multi-prototype enhancement module to enhance the prototype discriminability. Specifically, the single-prototype enhancement mechanism is expanded to the multi-prototype enhancement version for capturing rich semantics. Besides, the shot-correlation within the category is calculated via the interaction of different samples to enhance the intra-category similarity. Ablation studies prove that the involved part-whole relations and proposed multi-prototype enhancement module help to achieve complete object segmentation and improve semantic discrimination. Moreover, under the integration of these two modules, quantitative and qualitative experiments on two public benchmarks, including S3DIS and ScanNet, indicate the superior performance of the proposed framework on the task of 3D point cloud semantic segmentation, compared to some state-of-the-art methods.

KEYWORDS

Few-shot; point cloud; semantic segmentation; CapsNets

1 Introduction

Semantic segmentation of point clouds involves the assignment of semantic categories to individual points, enabling the identification of elements like walls, floors, and windows within a building. This task holds significant importance in 3D scene understanding and finds widespread applications in fields such as robotics, autonomous driving, and augmented reality [1–3]. Current approaches for fully supervised point cloud semantic segmentation heavily rely on deep learning networks. However,

progress in this area is hindered by the laborious process of annotating large-scale point cloud datasets, impeding the development of deep point cloud semantic segmentation methods [4,5]. It is an important topic for 3D scene understanding with a wide range of applications, such as robotics, autonomous driving, and augmented reality. The current fully supervised point cloud semantic segmentation methods [6–8] are predominantly based on deep learning networks. However, the extensive labor involved in annotating large-scale point cloud data acts as a bottleneck in the development of current deep point cloud semantic segmentation techniques [9–11].

In contrast to the fully supervised framework, the few-shot learning paradigm [12] aims to address novel category recognition tasks with minimal labeled data. This approach involves splitting the training data into two segments: a labeled support set and an unlabeled query set. Few-shot learning seeks to generalize categorical features learned from a limited number of samples in the support set to classify novel categories in the query set. Inspired by this, few-shot learning has been successfully applied to the domain of point cloud semantic segmentation few-shot [13–15]. The task of few-shot 3D point cloud semantic segmentation has been explored in several instances. For instance, Zhao et al. [13] initiated this research and introduced a dual-branch attention-aware multi-prototype transduction inference method to address the challenge of limited data in existing point cloud segmentation through few-shot learning. Moreover, they enhanced feature discriminability by employing multiple diverse feature learning networks. In point cloud semantic segmentation, challenges often arise in the background area. There can be ambiguity between the foreground of one class in the support set and the background of another class. Subsequently, Lai et al. [14] introduced a regularization technique targeting the prediction entropy of the support set within the loss function to mitigate background ambiguities. Additionally, Mao et al. [15] introduced a Bidirectional Feature Globalization (BFG) module to facilitate global feature learning and promote interaction between features at various levels, thereby enhancing the model's ability to perceive complex structures. While these methods make preliminary steps towards the task of few-shot 3D point cloud semantic segmentation, they still encounter challenging problems: i) These methods pay no attention to semantic wholeness. Usually, an object is composed of several structures, i.e., parts of the object, which possess different shapes at different locations in the image. The previous methods usually miss some part regions in the segmentation, leading to incomplete object shapes of different semantic objects, as shown in the third column of Fig. 1. Contents circled in red show AttMPTI [13] pays no attention to the semantic wholeness; ii) The previous methods usually focus on the discriminability of the features within separate categories. However, there are many semantic objects belonging to many categories. The previous methods, ignoring the discriminability across different categories, will be confused by different categories, leading to the problem of semantic ambiguity in subsequent matching labels.

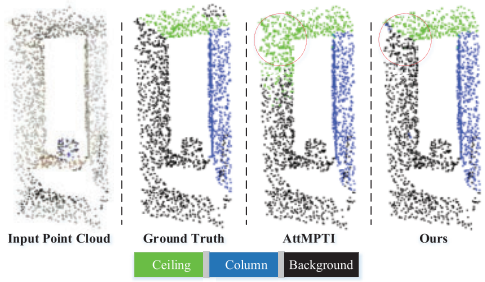

Figure 1: Comparative analysis of semantic segmentation on the S3DIS [16] dataset

This paper aims to address these two critical challenges. To tackle the first issue, we leverage the concept of part-whole relationships, empowered by Capsule Networks (CapsNets) [17] within the few-shot point cloud semantic segmentation task. This is realized by implementing dynamic routing in 3D CapsNets [18] in the encoder to generate latent capsules encoding the semantic wholeness required for accurate semantic segmentation. Specifically, CapsNets model point cloud data to generate latent capsules, each revealing the geometric or local structure of the point cloud object and its associated existence probability. The dynamic routing mechanism in CapsNets amalgamates geometric information to construct higher-level feature representations, capturing relationships between object parts and their wholes through capsule routing.

In this paper, we design a multi-prototype enhancement module for the second challenge to handle category prototypes. Specifically, i) expanding the single-prototype enhancement mechanism [19] to a multi-prototype enhancement is more advantageous in capturing rich semantics; ii) the shot-correlation is calculated via the interaction of different samples within the category, which is further utilized to enhance the intra-category similarity.

All in all, our contributions can be summarized as follows:

- We involve CapsNets to explore part-whole relations for the problem of few-shot 3D point cloud semantic segmentation, which is the first attempt to the best of our knowledge.
- A generalized multi-prototype enhancement module is proposed to improve the category discrepancy for few-shot point cloud semantic segmentation.
- We conduct comprehensive experiments and achieve competitive results on benchmark datasets S3DIS [16] and ScanNet [20].

The rest of the paper is organized as follows. Section 2 reviews the development of 3D point cloud segmentation, few-shot learning, few-shot 3D point cloud segmentation, and 3D capsule networks. Section 3 mainly introduces our proposed methods, 3D Capsule Network, and Multi-prototype Enhancement. Section 4 shows semantic segmentation results and demonstrates our method's superiority through ablation experiments. Section 5 summarizes our article.

2 Related Work

This section provides an overview of the relevant research, including the current state-of-the-art techniques in 3D point cloud semantic segmentation, few-shot learning methods, few-shot learning in 3D point cloud semantic segmentation, and the applications of 3D capsule networks.

2.1 3D Point Cloud Semantic Segmentation

Semantic segmentation of point clouds requires accurate classification of each point in a 3D point cloud scene, which has important applications in the real world, such as autonomous driving and robotics. Point clouds were mostly converted into other forms such as voxel grids for processing. PointNet [11] was the earliest work for point cloud semantic segmentation by an end-to-end deep network and did not need to change the point cloud into other forms. PointNet used shared MLP to learn point-wise features and constructed symmetric functions to solve the disorder problem of point clouds. Its efficient and simple network structure promoted the rapid development of point cloud networks.

Although PointNet [11] used a symmetric function to process point clouds as a whole, it overlooked the interconnection between each point and its neighboring points. The absence of local

features in PointNet poses challenges in analyzing complex scenes. PointNet++ [21] introduced two main improvements to better extract local features. PointNet++ used a hierarchical approach to iteratively extract features from local regions of point sets, allowing it to learn features with larger local scales. Additionally, since the point set distribution is often uneven, using a default uniform approach would result in worse network performance. Therefore, the author proposed a feature extraction method with adaptive density. Through these two methods, PointNet++ can learn more features to become more efficient and robust.

Dynamic graph CNN (DGCNN) [22] designed a graph-based point cloud network to convert point cloud data into graph data and proposed an EdgeConv module to capture local aggregate information. Specifically, DGCNN first constructed each point as a node into an undirected graph and connected the relationship between neighbor points and the central point as edges. It then formed a fixed-size data graph through local neighborhoods at each point. Next, DGCNN ensured the model's perception of local information by introducing a distance-based neighbor sampling technique and used the K-NN aggregation function on each point to calculate the feature representation of its neighbor points. Finally, in the process of dynamic graph convolutional neural network, the characteristics calculated by each layer were different. This means that the connection relationship of the graph was learned by the network itself, so each layer's graph had different vertices.

Linked Dynamic Graph CNN (LDGCNN) [23] was an improved work based on DGCNN and PointNet, where the transformation network of DGCNN was removed to reduce the model size. Additionally, because the deep features and their neighborhoods could be too similar to provide valuable edge vectors, LDGCNN connected the hierarchical features of different dynamic graphs. This approach was called Linked Dynamic Graph CNN (LDGCNN). In LDGCNN, the first layer was removed, and each EdgeConv layer could fully utilize the features extracted by the previous layer. We adopt LDGCNN as the backbone to extract the local and global information of point cloud data.

2.2 Few-Shot Learning

Traditional machine learning relies on a large number of labeled data sets. In practical applications, due to the high cost of data collection and labeling, it is common to have a limited amount of labeled data, and most categories do not have sufficient data accumulation. Therefore, we hope that the model can quickly generalize to new categories after learning a large amount of prior knowledge. Unlike traditional neural networks, the main purpose of few-shot learning is to extract general features and regularities from a small amount of labeled data and quickly transfer this knowledge to new tasks or domains.

The metric-based few-shot learning scheme is a common method that mainly calculates the similarity between different samples based on the distance metric. Commonly used metric-based few-shot learning schemes include Siamese Network [24], Matching Network [12], and Prototypical Network [25].

Specifically, the Siamese Network is mainly used to calculate the similarity between each sample in the support set and each sample in the query set. It is mostly used for image matching and target tracking. The Matching Network uses embedding functions to learn the general features of categories on the support set and then uses a classifier to calculate the similarity between support set samples and query set samples. It is mostly used for classification tasks. Unlike the Matching Network, the Prototype Network does not need to perform sample-level matching but abstracts the sample set of each category into a prototype vector to realize the rapid classification of new samples.

ProtoNet [26] was modeled based on the Prototypical Networks and used global average pooling to generate a single prototype vector from the features extracted from each type of sample to centrally express the features of each semantic class in the support point set. A similarity measure function was then applied to establish the relationship between the prototypes and the features of the query point set. We use the Prototypical Networks for the learning scheme inspired by [13]. We also use a multi-prototype generation approach.

2.3 Few-Shot 3D Point Cloud Semantic Segmentation

Compared with 2D images, 3D point cloud data is sparser. Due to the sparsity and noise of point cloud data, higher complexity models are often required for processing. However, complex models require more data for training, which increases the demand for labeled data. Few-shot learning methods enable the model to achieve good performance with only a small amount of data by quickly adapting to new tasks.

AttMPTI [13] introduced a prototype-based transformation learning method to predict the semantic class of a set of query points. For the first time, prototype meta-learning was applied to the task of semantic segmentation of 3D point clouds. Additionally, to better express the characteristics of a category, it also extracted multiple prototypes of support set features to better represent rich foreground semantics.

Aug-FT-A [14] considered that in the task of multi-target categories, the method of labeling samples involved using a binary mask to mark the points of the target category as foreground and mark other points as background. However, in tasks involving multiple object categories, since the background can have different meanings for different object categories, background ambiguity could arise. Some points labeled as background in one sample could belong to the foreground category of other samples. This led to incorrect annotations and had a significant impact on segmentation performance. Furthermore, for the few-shot point cloud semantic segmentation task, there were feature differences between samples. Support and query samples of the same category could have different characteristics due to the diversity among instances. Therefore, it designed a parameter-less feature transformation operation to propagate the information in the query sample to the support set, aiming to reduce feature variance and improve the generalization performance of the model.

To improve the generalization ability of point features, BFG [15] proposed a two-way feature globalization (BFG) method, which included two methods: Point-to-Prototype Globalization and Prototype-to-Point Globalization. These methods aimed to achieve a sparse prototype with global representation and embed global awareness into local point features.

CIF [19] considered that the generated category prototypes had subtle inter-class differences due to the small sample size, which posed a challenge for small sample classification. Inspired by [19], we applied it to point cloud semantic segmentation and multiple prototype generation.

2.4 3D Capsule Networks

Capsule Networks [17,27–29], also known as CapsNets, were a relatively new type of neural network architecture that was introduced by Geoffrey Hinton and his team. The idea behind Capsule Networks was to create a hierarchical representation of objects in the form of capsules, which were groups of neurons that represented specific properties of an object, such as its orientation, scale, and deformation.

The development of Capsule Networks was motivated by the limitations of traditional neural networks in handling spatial relationships between different features. In traditional neural networks, features were treated as independent, and their spatial relationships were not explicitly considered. This could lead to poor performance in tasks such as object recognition, where the relative positions of different parts of an object are important.

Capsule Networks aimed to address this limitation by explicitly modeling the spatial relationships between different features using capsules. Each capsule contained a set of neurons that represented a specific feature and its properties. The activation of these neurons was weighted by the probability of the feature being present in the input. Capsules were then organized hierarchically based on the relationships between different features, allowing the network to capture more complex relationships between features. Since the introduction of CapsNets, they have been applied to various tasks such as image recognition, semantic segmentation, object detection [30–34], etc., and have achieved promising results in improving the accuracy and efficiency of these tasks.

Recently, CapsNets have been applied to the research of 3D point clouds. 3D Point Capsule Networks [18] was the first attempt to apply CapsNets to solve 3D point cloud tasks. The latent capsule layer was designed to tackle 3D reconstruction, local feature extraction, part interpolation, and replacement tasks. Geometric Capsule Autoencoders [35] proposed the concept of geometric capsules and regarded each target as two components: a pose and a feature. Quaternion Equivariant Capsule Networks [36] proposed a quaternion equivariant capsule module to solve the rigid transformation problem of point clouds. We use the part-object relational property of CapsNets to solve the problem of few-shot 3D point cloud semantic segmentation, which helps improve object wholeness.

3 Methodology

In this section, we present the methodology employed in our proposed approach, which includes the utilization of a 3D capsule network and a multi-prototype enhancement model. The point cloud data undergoes feature generation using an embedding network, specifically a 3D capsule network. These generated features are then refined through multi-prototype enhancement to serve as prototypes. Finally, the labels for the query set are determined using label propagation based on these refined prototypes.

3.1 Overview

We present a novel approach for point cloud semantic segmentation by leveraging a strategy originally proposed for few-shot learning [12]. To guarantee that the model can generalize well to unseen point cloud data, the dataset is split into two non-overlapping sets, C_{train} and C_{test} where ($\mathrm{C}_{\text{train}} \cap \mathrm{C}_{\text{test}} = \varnothing$), each containing unique types of point cloud data. A set of few-shot tasks from C_{train} to train the model and evaluate the model's performance using another set of few-shot tasks sampled from C_{test}. Each few-shot task is a N-way K-shot episode point cloud segmentation task. We denote the point cloud as $\boldsymbol{P}$ contains M points. Each point belongs to the Euclidean space $\mathfrak{R}^D$, where D is typically set to 3 to represent the 3D coordinates of a point. However, D can exceed 3 to include additional information such as color, surface, etc. For each episode, the support set S with a few labeled examples can be expressed as $S = \left\{\left(\boldsymbol{P}_s^{1,k}, \boldsymbol{M}^{1,k}\right)_{k=1}^{K}, \cdots, \left(\boldsymbol{P}_s^{N,k}, \boldsymbol{M}^{N,k}\right)_{k=1}^{K}\right\}$, which contains K pairs of support point cloud $\boldsymbol{P}_s^{n,k}$ sampled from the training set $\mathrm{C}_{\text{train}}$ and its corresponding binary mask $\boldsymbol{M}^{n,k}$ for each of the N classes. The query set Q with unlabeled examples is expressed as $Q = \left\{\boldsymbol{P}_q^i, \boldsymbol{L}^i\right\}_{i=1}^{T}$, which can also be obtained from the training set $\mathrm{C}_{\text{train}}$. There are T pairs of point clouds in total,

$S \cap Q = \varnothing$ and the label $\boldsymbol{L}$ of the query set is only used during training. An N-way K-shot task is to sample N classes from the support set with K samples from each class, as input to the model. Our goal is to learn a model $f_\Phi\left(\boldsymbol{P}_q^{\,i}, S\right)$ to predict the distribution $\boldsymbol{H} \in \Re^{M \times (N+1)}$ of the query point cloud $\boldsymbol{P}_q$ based on S:

$$\Phi^* = \arg\min_{\Phi} E_{(S,Q)} \sim T_{\text{train}} \left[\sum_{\left(P_q^i, L^i\right) \in Q} J\left(\boldsymbol{L}^i f_\Phi\left(\boldsymbol{P}_q^i, S\right)\right) \right], \tag{1}$$

where T_{train} is obtained from the combination of S and Q to represent all episodes, and $J(.)$ is the loss function that will be defined in Section 3.5.

Our network framework is based on the AttMPTI [13] baseline, consisting of two branches: the support branch and the query branch. Both branches use weight-sharing feature embedding networks to extract point features, which we denote as F_S and F_Q for the support and query branches, respectively.

In the support branch, the prototype generation module generates prototypes of the features and masks. The labels of the prototypes are propagated to the query set features through the label propagation algorithm, and the cross-entropy loss function is used to measure the overall loss of the network.

Fig. 2 provides an overview of our network framework, which comprises four main parts, including i) the embedding network for mapping the input point cloud data to the multi-level feature space; ii) the multi-prototype enhancement module for enhancing the prototype representation of point cloud features; iii) the label propagation algorithm for propagating labels to the unlabeled query set; iv) the cross-entropy loss function for training the network.

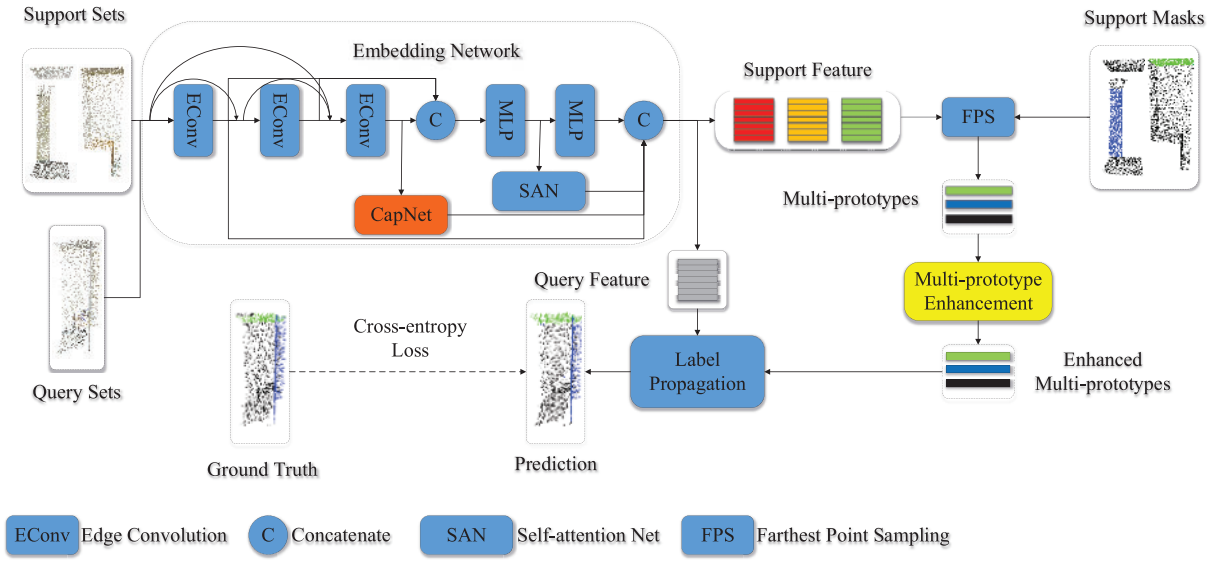

Figure 2: Overview of our proposed network framework

3.2 Embedding Network

The embedding network is responsible for extracting unique features from all points in both the support and the query sets. Different network frameworks are capable of learning features with distinct

characteristics. Ideally, each feature should maximize the use of learned spatial features, encode local and global features, recognize the relationship between object parts, and remain flexible to adapt to various few-shot tasks.

3.2.1 Multi-Level Feature Learning Network

We design a novel embedding network that facilitates multi-level feature learning to achieve these objectives. The network comprises four distinct modules: a feature extractor, capsule learner, attention learner, and metric learner. For the feature extraction of point cloud data, we employ LDGCNN [23] as the backbone network, and the local features are generated as a result of the first layer, EdgeConv.

Specifically, we fully leverage the extracted features by utilizing dense connections as the feature extractor and the first EdgeConv output as local features. The capsule learner in our proposed approach facilitates learning of object parts and their relationships at the part-to-part and part-to-whole levels. Since the self-attention mechanism does not affect the order of the input [37,38], the self-attention network (SAN) is also incorporated into our proposed method to explore the semantic information of the global context. A multi-layer perceptron (MLP) is the metric learner that adapts to different down-sampling tasks. The four-level features are finally connected to produce the output of the embedding network.

3.2.2 3D Capsule Network

Taking 2-way 1-shot as an example, our embedding network takes three $n \times D$ point clouds as input, where two are support sets and one is the query set. Here n represents the number of samples in each point cloud and D is the dimension of the point cloud. Fig. 2 illustrates the network architecture, which includes a support branch (top) and a query branch (bottom) employing weight-sharing feature embedding networks for point feature extraction. In the support branch, prototypes are generated on point features and masks using the multi-prototypes generation module. The labels of these prototypes are then propagated to the query set features through the label propagation algorithm. The network is trained using the cross-entropy loss function. The backbone network LDGCNN [23] extracts the point cloud data, and each layer of EdgeConv generates a feature vector of size $n \times 16$. The output F from the last layer of EdgeConv is used as the input for CapsNet. The CapsNet module is shown in Fig. 3. F serves as the input of the capsule with the size of $n \times 64$, where n and 64 are the number of points and feature dimension, respectively. Based on F, capsules can be obtained by the following equation:

$$\mathrm{U_{j|n}} \in \mathfrak{R}^{1\times 16} = \mathrm{W_{n|j}F_n}, \mathrm{F_n} \in \mathfrak{R}^{1\times 64}, \tag{2}$$

where j is one of 64 types of capsules. Subsequently, the output jth capsule S is computed by

$$\mathrm{S_j} \in \mathfrak{R}^{1\times 16} = \left(\mathrm{C_{j|1}} \times \mathrm{U_{j|1}} + \ldots + \mathrm{C_{j|n}} \times \mathrm{U_{j|n}}\right), \tag{3}$$

where $\mathrm{C_{j|n}} \in \mathfrak{R}^{1\times 16} = \mathrm{Softmax}\left(\mathrm{B_{j|n}}\right)$, and the computation of $\mathrm{B_{j|n}}$ can be found in [17]. The final output capsules are

$$\mathrm{S} \in \mathfrak{R}^{64\times 16} = \{\mathrm{S_1}, \mathrm{S_2}, \ldots, \mathrm{S_{64}}\}. \tag{4}$$

The nonlinear function Squash is applied to compress S and obtain V, which compresses the probability of capsule existence represented by the modulus length of the vector to a range between 0 and 1. The routing process iterates T times, where each time the product of V and U is added to the initial value of B to update its value. Finally, linear interpolation is used to decode V into a feature vector F′ of size $\mathrm{n} \times 16$.

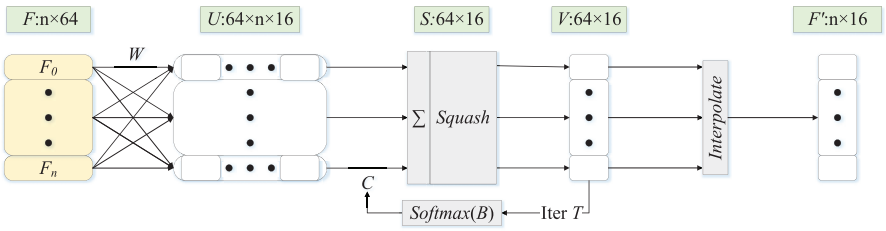

Figure 3: The 3D CapsNet module for 3D few-shot point cloud semantic segmentation

3.3 Multi-Prototype Enhancement

Multi-prototype enhancement aims to increase the diversity of categories and make it easier for the network to distinguish between different categories. Multi-prototype enhancement consists of two parts, the first is the generation of multi-prototypes and the second is to enhance the generated prototypes.

3.3.1 Multi-Prototype Generation

To represent the prototype of a class, we adopt a clustering approach that utilizes the mean of features extracted from a different prototype network [25]. Specifically, for N classes in the support set, we generate $(N+1) \times c$ prototypes (including one background class), where c is the number of prototypes of each class. For a 2-way 1-shot task, the size of the support set prototypes is $3c \times d$ where d is the dimension of the prototype. We use the simple farthest point sampling (FPS) [21] clustering method to generate prototypes. Once we obtain the support set feature F_s, we use FPS to obtain the corresponding prototypes P_s, denoted as

$$P_s \in \Re^{(N+1)c \times d} = FPS\left(F_s\right), F_s \in \Re^{n \times d}. \tag{5}$$

3.3.2 Multi-Prototype Generation

By utilizing the Multi-Prototype Enhancement block, we effectively address the issue of prototype class differentiation reduction. We obtained the prototype P_s of the initial $N+1$ classes, where each class can be expressed as $P_s = \{P_s^i\}_{i=1}^{N+1}$. The Multi-prototype of the first class can be denoted as P_s^1.

We input the multi-prototype of each class into the Multi-Prototype Enhancement block. The query-vector q, key-vector k and value-vector v are generated through three convolutional layers: $Conv1$, $Conv2$ and $Conv3$.

$$\begin{aligned} q \in \Re^{c \times d} &= Conv1\left(P_s^1\right), P_s^1 \in \Re^{c \times d}, \\ v \in \Re^{c \times d} &= Conv2\left(P_s^1\right), P_s^1 \in \Re^{c \times d}, \\ k \in \Re^{c \times d} &= Conv3\left(P_s^1\right), P_s^1 \in \Re^{c \times d}. \end{aligned} \tag{6}$$

Similar objects have subtle differences, but different channels convey different semantic information [39,40]. R_1 obtained by modeling channel relations, learns more diverse features to address subtle interclass differences. In addition, we obtain R_2 by modeling the object prototype relationship, which serves to supplement the prototype information lost during channel modeling.

$$\begin{aligned} R_1 \in \Re^{d \times d} &= q^T k, \\ R_2 \in \Re^{c \times c} &= k q^T. \end{aligned} \tag{7}$$

Then v is weighted with R_1 and R_2 to obtain v_1 and v_2,

$$v_1 \in \Re^{c \times d} = v \times Softmax(R_1),$$
$$v_2 \in \Re^{c \times d} = Softmax(R_2) \times v. \tag{8}$$

Finally, v_1, v_2, and the initial prototype P_s^1, are enhanced prototype $P_s^{1'}$.

$$P_s^{1'} \in \Re^{c \times d} = P_s^1 + v_1 + v_2. \tag{9}$$

Architecture of the Multi-Prototype Enhancement (MPE) module is shown in Fig. 4. The initial prototype P_s^1 obtains the query-vector q, key-vector k and value-vector v through three convolutional layers of shared parameters. Then the relationship map R_1 is obtained through the channel relationship interaction, and the prototype relationship interaction is obtained relationship map R_2. R_1 and R_2 are reweighted with v after *Softmax* to obtain new features v_1 and v_2, and finally combined with the initial features to obtain enhanced features $P_s^{1\prime}$.

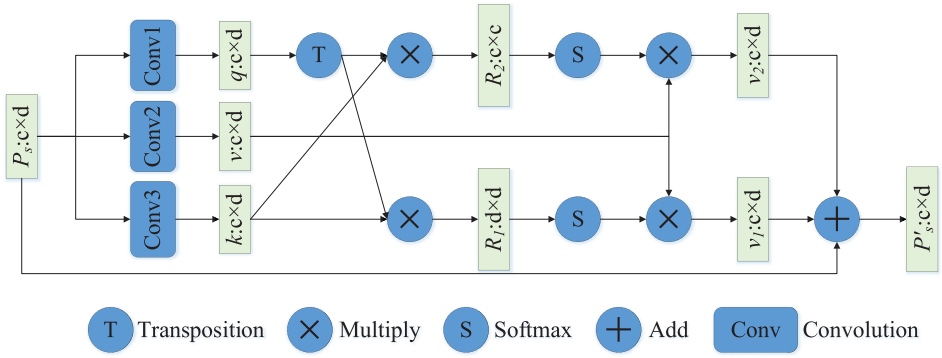

Figure 4: The Multi-Prototype Enhancement module

Fig. 5 illustrates the distinct categories through varied shapes and colors. After the Multi-Prototype Enhancement module, the differences of different categories become more apparent. Compared with single prototype augmentation, there are two specific differences [19]. (1) We extended the single prototype to multiple prototypes. (2) We expanded the original channel correlation to shot correlation.

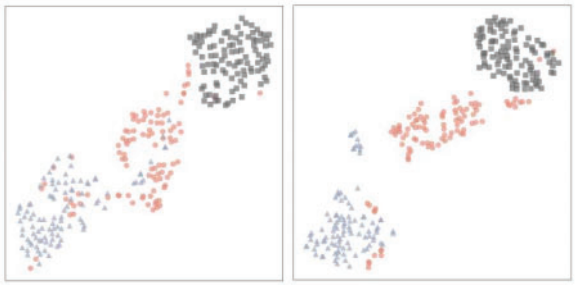

Figure 5: The support set prototype distribution before (a) and after (b) using the Multi-Prototype Enhancement module on the S3DIS dataset is compared by T-SNE

3.4 Label Propagation

The label propagation algorithm [41] is employed to transfer the labels from the support set to the query set. This algorithm is based on a graph model, where each vertex represents a sample, including the support set prototypes and the query set features. The graph is composed of vertices and edges, and its size $V = c \times (N+1) + n \times 1$ is determined by the $c \times (N+1)$ support samples and $c \times 1$ query samples. To reduce the number of query points and improve computational efficiency, we construct a k-Nearest Neighbor ($K - NN$) graph structure. The edge between two vertices represents their similarity or the probability of vertex i to vertex j. Gaussian similarity is used to calculate the distance between a node and the sparse affinity matrix $\boldsymbol{A} \in \mathfrak{R}^{V \times V}$ is defined as

$$\boldsymbol{A_{ij}} = \exp\left(-\frac{||v_i - v_j||_2^2}{2\sigma^2}\right), for\ v_j \in N_k(v_i), \tag{10}$$

where v_j represents the vertex feature and σ^2 is the variance of the distance between two vertices.

We adopt the approach of [42] by letting $\boldsymbol{W} = \boldsymbol{A} + \boldsymbol{A^T}$, which ensures that the matrix is non-negative and symmetric. The normalized matrix is calculated by $\boldsymbol{S} = \boldsymbol{D^{-1/2}WD^{-1/2}}$, where $\mathbf{D}$ is the diagonal matrix of $\mathbf{W}$, and the diagonal value is the sum of the rows of $\mathbf{W}$. We define a label matrix $\boldsymbol{Y} \in \mathfrak{R}^{V \times (N+1)}$, where the rows corresponding to labeled examples are one-hot encoded labels and the rest are zero. The label propagation algorithm is then applied to propagate the labels from the support set to the query set, where the affinity matrix $\mathbf{S}$ and the label $\mathbf{Y}$ are used to diffuse the label according to the following formula:

$$\boldsymbol{Z} = (\boldsymbol{I} - \alpha \boldsymbol{S})^{-1} \mathbf{Y}, \tag{11}$$

where $\alpha \in [0, 1)$ is the parameter to represent the relative amount of the initial label of its neighboring nodes.

3.5 Cross-Entropy Loss Function

Once $\mathbf{Z}$ has been acquired, the labels $\{z^i\}_{i=1}^T$ for the corresponding T query sets $\boldsymbol{P}_q{}^i$ can be obtained using the *Softmax*.

$$\boldsymbol{H}_{m,n}^i = \frac{\exp\left(\boldsymbol{z}_{m,n}^i\right)}{\sum_{j=1}^{N+1} \exp\left(\boldsymbol{z}_{m,j}^i\right)}. \tag{12}$$

Then, we compute the cross-entropy loss between predictions of the point cloud $\{\boldsymbol{H}^i\}_{i=1}^T$ and the ground truth labels $\{\boldsymbol{L}^i\}_{i=1}^T$ to train the entire network.

$$J_\Phi = -\frac{1}{T}\frac{1}{M}\sum_{i=1}^{T}\sum_{m=1}^{M}\sum_{n=1}^{N+1} 1\left[\mathbf{L}_{m^i} = n\right] \log\left(\mathbf{H}_{m,n}^i\right), \tag{13}$$

where Φ is the set of parameters of our model $f_\Phi\left(\boldsymbol{P}_q^i, S\right)$ and M is the number of points in each point cloud scene.

4 Experiments

4.1 Datasets and Metrics

To validate the effectiveness of our proposed experimental method, we evaluate it on two benchmark datasets, Standford 290 Large-Scale 3D Indoor Spaces Dataset(S3DIS) [16] and ScanNet [20].

The datasets are partitioned into $\boldsymbol{N}$-way $\boldsymbol{K}$-shot groups with consistent cross-validation parameters, and we compare our results with the most advanced baseline proposed by [16].

The S3DIS dataset comprises 271 point cloud rooms across six regions, where each point is assigned semantic information from a total of 12 categories plus clutter. Each point is represented by nine dimensions, including coordinate dimension information (x, y, z), color information (r, g, b), and normalized coordinate information (X, Y, Z). The ScanNet dataset contains 1513 point cloud clusters, 707 scenes, and 21 categories, where each point is represented by six dimensions comprising coordinate and color information.

To set up the dataset in N-way K-shot format, we divide it into two disjoint subsets, C_{train} and C_{test}, based on category. The specific split results are presented in Table 1. For training, we select one of the split results and use the other for testing. Since the original dataset has many points per scene, we follow the method outlined in [11] to divide each room into multiple $1m \times 1m$ blocks along the xy plane. During the training process, we sample $M = 2048$ points for each block. The training set T_{train} is sampled from C_{train} as follows: we randomly select N categories from the training set based on the N-way K-shot task to be processed, and then randomly sample a support set S and a query set Q. We repeat this process for 100 episodes for both support and query sets, as well as for the test set. For a 2-way 1-shot task on the S3DIS dataset, we randomly select two categories from the six categories in split $= 0$ resulting in $C_2^6 = 15$ combinations. We sample blocks with these two categories, each having no fewer than 100 points. The sampled block is then divided into a support set S and a query set Q, which also include the binary mask $\boldsymbol{M}$ of the support set and the label $\boldsymbol{L}$ of the query set. We create 100 combinations of $N \times K$ support point clouds and one query set point cloud to form one episode. We repeat this process to create 100 combinations for our training set T_{train}. The test set T_{test} is generated using the blocks with split $= 1$.

Table 1: The details of classes splitting of S3DIS and ScanNet

	Split = 0	Split = 1
S3DIS [16]	beam, board, bookcase, ceiling, chair, column	door, floor, sofa, table, wall, window
ScanNet [20]	bathtub, bed, bookshelf, cabinet, chair, counter, curtain, desk, door, floor	other furniture, picture, refrigerator, show curtain, sink, sofa, table, toilet, wall, window

The metric employed in this context is the mean Intersection over Union (mean-IoU) metric, which represents the standard metric for evaluating semantic segmentation. It calculates using the following formula:

$$I\text{oU} = TP/\left(TP + FP + FN\right), \tag{14}$$

where TP represents a positive class that has been accurately classified, FP stands for a negative class that has been misclassified as a positive class, TN stands for a negative class that has been accurately classified, and FN represents a positive class that has been misclassified as a negative class.

4.2 Implementation Details

To perform a 2-way 1-shot task, we sample the support set of 2-point cloud sets, each containing $M = 2048$ points, and input the data of 9 dimensions of each point, $2 \times 2048 \times 9$, into the network. The

Embedding network uses different feature extraction modules to map the data to a high-dimensional space of $2 \times 2048 \times 208$. The prototype generation module generates a data size of 300×208, which includes 200 category prototypes and 100 background prototypes. Then, the features extracted from the labeled support set prototype and the unlabeled query set are generated, resulting in a total of $2048 + 300$ sparse nodes that are represented by matrix structures and connected, where each node connects the most similar 100 points with Gaussian similarity. Next, we predict the label distribution of the query set using the label propagation algorithm and measure the loss with the cross-entropy loss function.

To pre-train the Embedding Network, we use the C_{train} dataset. We train the optimizer using Adam with a learning rate of 0.001, a batch size of 32, and a total of 100 epochs. We then use the trained weights as initialization to train the entire network using Adam as the optimizer with a batch size of 32, a learning rate of 0.001, and a halving of the learning rate every 5000 iterations. Additionally, we augment the data with Gaussian jittering and z-axis rotation of the dataset.

4.3 Quantitative Results

Tables 2 and 3 display the results of our experiments on two benchmark datasets, with a total of 4 task settings: 2/3-way 1/5-shot. S^i denotes the split i is used for testing. The baselines we use are ProtoNet, AttProtoNet, MPTI, AttMPTI from [13], BFG from [14], and CrossProto from [43]. Our experimental findings reveal that the classification accuracy decreases as the category increases, indicating that for small samples, less N-way is better, which is consistent with daily life cognition. Moreover, our results show that the segmentation performance significantly improves as the K-shot increases, suggesting that the more samples of the same category, the better the model can category features of objects. On both datasets, our method outperforms other baselines, especially about 5% higher than AttMPTI on the 2-way 1-shot task, and 2% to 5% higher on other tasks by mean-IoU improvements. This shows that our method can improve the accuracy of segmentation even when the sample size is very small.

Table 2: Quantitative results on the S3DIS dataset using mean-IoU metric (%)

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
ProtoNet [13]	48.3	49.98	49.19	57.34	63.22	60.28	40.81	45.07	42.94	49.05	53.42	51.24
AttProtoNet [13]	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 [13]	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 [13]	53.77	55.94	54.86	61.67	67.02	64.35	45.18	49.27	47.23	54.92	56.79	55.86
BFG [14]	55.60	55.98	55.79	63.71	66.62	65.17	46.18	48.36	47.27	55.05	57.80	56.43
CrossProto [43]	43.20	41.49	42.35	49.20	50.32	49.76	–	–	–	–	–	–
Ours	**60.75**	**60 .81**	**60 .78**	**65 .27**	**69 .12**	**67 .20**	**47 .37**	**50 .68**	**49 .03**	**55 .19**	**58 .49**	**56 .84**

4.4 Quantitative Results

Fig. 6 shows the qualitative results of the S3DIS dataset in the 2-way 1-shot setting. The target classes from top to bottom of the three tasks are “chair & bookcase”, “board & beam”, and “ceiling & column”, respectively. Our proposed method is compared with the AttMPTI baseline on 6 categories

of indoor point clouds. Referring to Ground Truth in our method, it can be found that the ceiling, bookshelf, etc., have high accuracy. Significant improvements are achieved compared to the case of confounding the 2 segmentation classes and the background in AttMPTI (third column).

Table 3: Quantitative results on ScanNet dataset using mean-IoU metric (%)

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
ProtoNet [13]	33.92	30.95	32.44	45.34	42.01	43.68	28.47	26.13	27.3	37.36	34.98	36.17
AttProtoNet [13]	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 [13]	39.27	36.14	37.71	46.90	43.59	45.35	29.96	27.26	28.61	38.14	34.36	36.25
AttMPTI [13]	42.55	40.83	41.69	54.00	50.32	52.16	35.23	30.72	32.98	46.74	40.80	43.77
BFG [14]	42.15	40.52	41.34	51.23	49.39	50.31	34.12	31.98	33.05	46.25	41.38	43.82
CrossProto [43]	37.20	34.38	35.79	40.27	41.01	40.14	–	–	–	–	–	–
Ours	**47.40**	**45.54**	**46.47**	**59.16**	**55.73**	**57 .45**	**37 .51**	**34 .08**	**35 .80**	**49 .48**	**45 .03**	**47 .26**

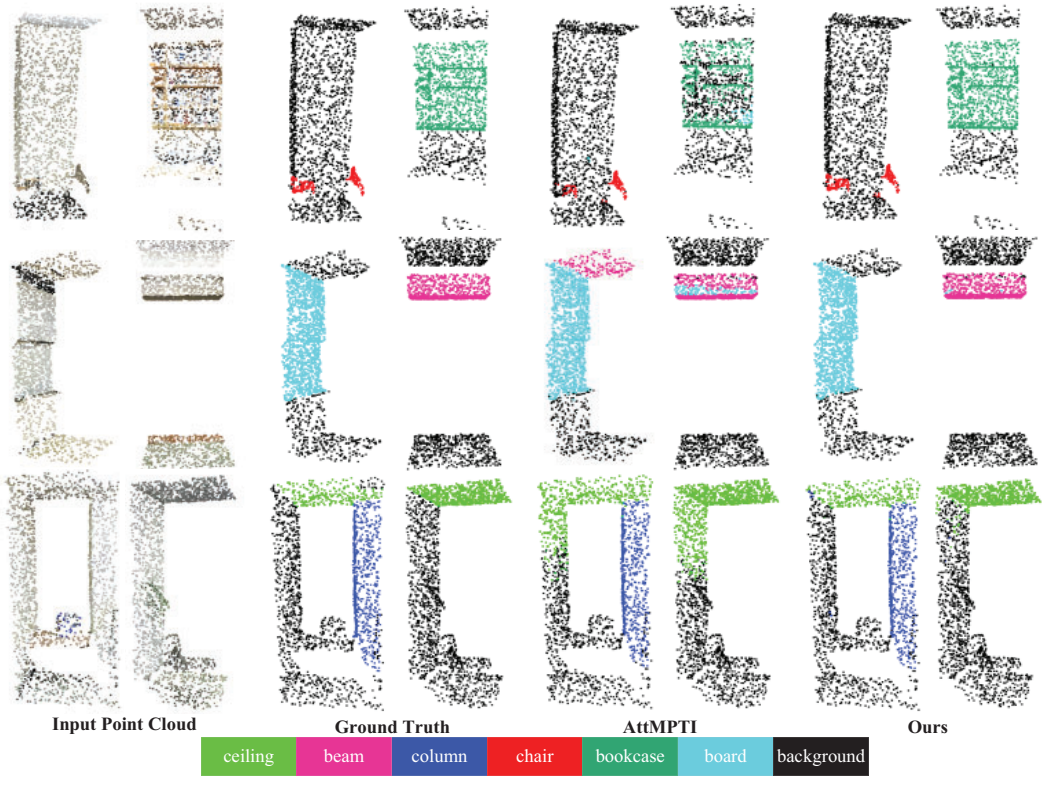

Figure 6: Comparison of qualitative results for 2-way 1-shot tasks on the S3DIS dataset against AttMPTI

4.5 Ablation Study

4.5.1 Multi-Prototype Generation

Table 4 shows the effect of the prototype enhancement block on single-prototype and multi-prototype in the 2-way 1-shot task on the S3DIS dataset. In the single prototype, we use the mean value instead of FPS to obtain the prototype representation of the category. As shown in Table 4, multi-prototypes are more representative of categories than single-prototypes, and the multi-prototype augmentation module has about a 3% improvement in mean-IoU.

Table 4: The effects of single-prototype enhancement and multi-prototype enhancement on the mean-IoU on the S3DIS dataset without prototype enhancement

Column	Column heading
Single-prototype	41.98
Single-prototype enhancement	43.05
Multi-prototype	53.77
Multi-prototype enhancement	56.78

Under the 2-way 1-shot task on the two datasets, we also separately compare the effect of using only channel interaction and MPE. The result is shown in the Table 5.

Table 5: The effect of using only channel interaction and MPE

Method	S3DIS		ScanNet	
	S^0	S^1	S^0	S^1
MPE-channel	59.22	59.77	47.00	44.74
MPE	60.75	60.81	47.40	45.54

4.5.2 Multi-Prototype Generation

Ablation study on the 3D Capsule Network: To verify the simplicity of using interpolation, we test Time and FLOPs. As shown in Table 6, DR indicates that the part of dynamic routing takes the longest time. Decoder1 comes from 3D Point-Capsule Networks [18], Decoder2 is calculated by using MLP to calculate the random grid in encoder1, and decoder3 uses linear interpolation to reduce the time spent.

4.5.3 Ablation Study for Each Module

Table 7 demonstrates the effectiveness of the proposed prototype enhancement module and capsule network module individually. A dense connection means that the backbone network uses LDGCNN. The top-down ablation analysis reveals progress in both datasets, with the improvement gradually diminishing.

Table 6: The time, floating-point operations per second, and number of parameters used by the 3D CapsNet

Method	Time	FlOPs	Params
DR	342.772 ms	2.621 M	20.5 K
Decoder1	24.511 ms	2.560 M	19.5 K
Decoder2	25.404 ms	2.687 M	19.5 K
Decoder3	6.233 ms	0	0

Table 7: Different modules' impact on ScanNet and S3DIS

MPE	CapsNet	Dense	S3DIS		ScanNet	
			S^0	S^1	S^0	S^1
✗	✗	✗	53.77	55.94	42.55	40.83
✓	✗	✗	56.78	57.78	45.11	42.89
✗	✓	✗	57.54	57.81	44.82	42.31
✗	✗	✓	55.08	56.26	44.90	42.21
✓	✓	✗	58.79	58.44	46.20	44.27
✗	✗	✓	58.50	58.41	46.66	44.52
✗	✓	✓	58.27	59.09	45.87	43.09
✓	✓	✓	60.75	60.81	47.40	45.54

The most favorable results are obtained on mean-IoU when all modules are implemented together.

4.5.4 Ablation Study on the Hyper-Parameters

We study the influence of the number of prototypes. The results are shown in Table 8. The number of prototypes was evaluated on the 2-way 1-shot task under the S3DIS dataset. From the table, it is roughly concluded that when the number of prototypes is close to 100, the mean-IoU can achieve the best.

Table 8: The effect of the number of prototypes for different categories on the mean-IoU on the S3DIS

Prototypes per class	S3DIS
50	57.59
100	60.75
150	57.46
200	56.85

5 Limitations and Future Work

Although our method performs well on indoor data, it is still not satisfactory when encountering outdoor environments, where there are some challenges such as incomplete point cloud collection with sparsity and more complex backgrounds. In the future, we will incorporate cutting-edge research to improve our model, such as prompt tuning [44] to improve semantic identification and Transformer [37] for feature discrimination.

6 Conclusion

In this paper, the few-shot semantic segmentation of the 3D point cloud is combined with the capsule network, and a new type of capsule decoder is designed, which pays attention to the relationship between the whole object and its parts to make the segmented object more complete. In addition, the multi-prototype features generated by few-shot learning are enhanced to make the features more distinguishable between different categories. Extensive experiments on benchmark datasets S3DIS and ScanNet demonstrate the effectiveness of the method.

Acknowledgement: The author wishes to extend sincere appreciation to Professor Lin Shi for the generous provision of equipment support, which significantly aided in the successful completion of this research. Furthermore, the author expresses gratitude to Associate Professor Ning Li and Teacher Wei Guan for their invaluable academic guidance and unwavering support. Their expertise and advice played a crucial role in shaping the direction and quality of this research.

Funding Statement: This work is supported by the National Natural Science Foundation of China under Grant No. 62001341, the National Natural Science Foundation of Jiangsu Province under Grant No. BK20221379, and the Jiangsu Engineering Research Center of Digital Twinning Technology for Key Equipment in Petrochemical Process under Grant No. DTEC202104.

Author Contributions: Study conception and design: Lujun Zhang, Yi Liu; data collection: Shoukun Xu, LuJun Zhang; analysis and interpretation of results: Shoukun Xu, Lujun Zhang; draft manuscript preparation: Lujun Zhang, Guangqi Jiang, Yining Hua. All authors reviewed the results and approved the final version of the manuscript.

Availability of Data and Materials: All publicly available datasets are used in the study.

Conflicts of Interest: The authors declare that they have no conflicts of interest to report regarding the present study.

References

[1] L. Zhao and W. Tao, "JSNet++: Dynamic filters and pointwise correlation for 3D point cloud instance and semantic segmentation," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 33, no. 4, pp. 1854–1867, 2022.

[2] Z. Song, L. Zhao and J. Zhou, "Learning hybrid semantic affinity for point cloud segmentation," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 32, no. 7, pp. 4599–4612, 2022.

[3] F. Yin, Z. Huang, T. Chen, G. Luo, G. Yu *et al.,* "DCNet: Large-scale point cloud semantic segmentation with discriminative and efficient feature aggregation," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 33, no. 8, pp. 4083–4095, 2023.

[4] J. Liu, Y. Chen, B. Ni and Z. Yu, "Joint global and dynamic pseudo labeling for semi-supervised point cloud sequence segmentation," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 33, no. 10, pp. 5679–5691, 2023.
[5] X. Chang, H. Pan, W. Sun and H. Gao, "A multi-phase camera-lidar fusion network for 3D semantic segmentation with weak supervision," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 33, pp. 3737–3746, 2023.
[6] S. Wu, C. Duan, B. Ren, L. Ren, T. Jiang *et al.,* "Point cloud based semantic segmentation method for unmanned shuttle bus," *Intelligent Automation & Soft Computing*, vol. 37, no. 3, pp. 2707–2726, 2023.
[7] A. Boulch, "ConvPoint: Continuous convolutions for point cloud processing," *Computers & Graphics*, vol. 88, pp. 24–34, 2020.
[8] X. Ye, J. Li, H. Huang, L. Du and X. Zhang, "3D recurrent neural networks with context fusion for point cloud semantic segmentation," in *Proc. of the European Conf. on Computer Vision*, Munich, Germany, pp. 403–417, 2018.
[9] Q. Hu, B. Yang, L. Xie, S. Rosa, Y. Guo *et al.,* "RandLA-Net: Efficient semantic segmentation of large-scale point clouds," in *IEEE/CVF Conf. on Computer Vision and Pattern Recognition*, Seattle, USA, pp. 11108–11117, 2020.
[10] Q. Huang, W. Wang and U. Neumann, "Recurrent slice networks for 3D segmentation of point clouds," in *IEEE Conf. on Computer Vision and Pattern Recognition*, Salt Lake City, USA, pp. 2626–2635, 2018.
[11] C. R. Qi, H. Su, K. Mo and L. J. Guibas, "PointNet: Deep learning on point sets for 3D classification and segmentation," in *IEEE Conf. on Computer Vision and Pattern Recognition*, Honolulu, USA, pp. 652–660, 2017.
[12] O. Vinyals, C. Blundell, T. Lillicrap, D. Wierstra *et al.,* "Matching networks for one shot learning," in *Advances in Neural Information Processing Systems*, vol. 29. Cambridge, Massachusetts, USA: The MIT Press, 2016.
[13] N. Zhao, T. S. Chua and G. H. Lee, "Few-shot 3D point cloud semantic segmentation," in *Proc. of the IEEE/CVF Conf. on Computer Vision and Pattern Recognition*, Nashville, TN, USA, pp. 8873–8882, 2021.
[14] L. Lai, J. Chen, C. Zhang, Z. Zhang, G. Lin *et al.,* "Tackling background ambiguities in multi-class few-shot point cloud semantic segmentation," *Knowledge-Based Systems*, vol. 253, pp. 109508, 2022.
[15] Y. Mao, Z. Guo, X. Lu, Z. Yuan and H. Guo, "Bidirectional feature globalization for few-shot semantic segmentation of 3D point cloud scenes," arXiv preprint arXiv:2208.06671, 2022.
[16] I. Armeni, O. Sener, A. R. Zamir, H. Jiang, I. Brilakis *et al.,* "3D semantic parsing of large-scale indoor spaces," in *Proc. of the IEEE Conf. on Computer Vision and Pattern Recognition*, Las Vegas, NV, USA, pp. 1534–1543, 2016.
[17] S. Sabour, N. Frosst and G. E. Hinton, "Dynamic routing between capsules," in *Advances in Neural Information Processing Systems*, vol. 30. Cambridge, Massachusetts, USA: The MIT Press, 2017.
[18] Y. Zhao, T. Birdal, H. Deng and F. Tombari, "3D point capsule networks," in *IEEE/CVF Conf. on Computer Vision and Pattern Recognition*, Long Beach, USA, pp. 1009–1018, 2019.
[19] C. Ye, H. Zhu, Y. Liao, Y. Zhang, T. Chen *et al.,* "What makes for effective few-shot point cloud classification?," in *IEEE/CVF Conf. on Computer Vision and Pattern Recognition*, Waikoloa, USA, pp. 1829–1838, 2022.
[20] A. Dai, A. X. Chang, M. Savva, M. Halber, T. Funkhouser *et al.,* "ScanNet: Richly-annotated 3D reconstructions of indoor scenes," in *IEEE Conf. on Computer Vision and Pattern recognition*, Honolulu, USA, pp. 5828–5839, 2017.
[21] 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*, vol. 30, pp. 5099–5180, 2017.
[22] Y. Wang, Y. Sun, Z. Liu, S. E. Sarma, M. M. Bronstein *et al.,* "Dynamic graph CNN for learning on point clouds," *ACM Transactions on Graphics*, vol. 38, no. 5, pp. 1–12, 2019.
[23] K. Zhang, M. Hao, J. Wang, C. W. de Silva and C. Fu, "Linked dynamic graph CNN: Learning on point cloud via linking hierarchical features," arXiv preprint arXiv:1904.10014, 2019.

[24] G. Koch, R. Zemel and R. Salakhutdinov, "Siamese neural networks for one-shot image recognition," in *ICML Deep Learning Workshop*, Lille, France, vol. 2, no. 1, 2015.
[25] J. Snell, K. Swersky and R. Zemel, "Prototypical networks for few-shot learning," in *Advances in Neural Information Processing Systems*, vol. 30. Cambridge, Massachusetts, USA: The MIT Press, 2017.
[26] N. Dong and E. P. Xing, "Few-shot semantic segmentation with prototype learning," in *BMVC*, Northumbria, UK, 2018.
[27] G. E. Hinton, A. Krizhevsky and S. D. Wang, "Transforming auto-encoders," in *Int. Conf. on Artificial Neural Networks*, Espoo, Finland, pp. 44–51, 2011.
[28] G. E. Hinton, S. Sabour and N. Frosst, "Matrix capsules with EM routing," in *Advances in Neural Information Processing Systems*. Cambridge, Massachusetts, USA: The MIT Press, 2018.
[29] A. Kosiorek, S. Sabour, Y. W. Teh and G. E. Hinton, "Stacked capsule autoencoders," in *Advances in Neural Information Processing Systems*, vol. 32. Cambridge, Massachusetts, USA: The MIT Press, 2019.
[30] Y. Liu, D. Zhang, Q. Zhang and J. Han, "Part-object relational visual saliency," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 44, no. 7, pp. 3688–3704, 2021.
[31] Y. Liu, D. Zhang, N. Liu, S. Xu and J. Han, "Disentangled capsule routing for fast part-object relational saliency," *IEEE Transactions on Image Processing*, vol. 31, pp. 6719–6732, 2022.
[32] Y. Liu, D. Zhang, Q. Zhang and J. Han, "Integrating part-object relationship and contrast for camouflaged object detection," *IEEE Transactions on Information Forensics and Security*, vol. 16, pp. 5154–5166, 2021.
[33] Y. Liu, X. Dong, D. Zhang and S. Xu, "Deep unsupervised part-whole relational visual saliency," *Neurocomputing*, vol. 563, pp. 126916, 2023.
[34] Y. Liu, Q. Zhang, D. Zhang and J. Han, "Employing deep part-object relationships for salient object detection," in *IEEE/CVF Int. Conf. on Computer Vision*, Seoul, South Korea, pp. 1232–1241, 2019.
[35] N. Srivastava, H. Goh and R. Salakhutdinov, "Geometric capsule autoencoders for 3D point clouds," arXiv preprint arXiv:1912.03310, 2019.
[36] Y. Zhao, T. Birdal, J. E. Lenssen, E. Menegatti, L. Guibas *et al.,* "Quaternion equivariant capsule networks for 3D point clouds," in *European Conf. on Computer Vision*, Glasgow, UK, pp. 1–19, 2020.
[37] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones *et al.,* "Attention is all you need," *Advances in Neural Information Processing Systems*, vol. 30, pp. 5998–6008, 2017.
[38] Y. Wu, H. Ding, M. Gong, A. K. Qin, W. Ma *et al.,* "Evolutionary multiform optimization with two-stage bidirectional knowledge transfer strategy for point cloud registration," *IEEE Transactions on Evolutionary Computation*, pp. 1, 2022.
[39] Y. Gao, X. Han, X. Wang, W. Huang and M. Scott, "Channel interaction networks for fine-grained image categorization," in *AAAI Conf. on Artificial Intelligence*, New York, USA, pp. 10818–10825, 2020.
[40] J. Yosinski, J. Clune, A. Nguyen, T. Fuchs and H. Lipson, "Understanding neural networks through deep visualization," arXiv preprint arXiv:1506.06579, 2015.
[41] X. J. Zhu and Z. Ghahramani, "Learning from labeled and unlabeled data with label propagation," *Technical Report CMU-CALD-02–107*, Carnegie Mellon University, 2002.
[42] A. Iscen, G. Tolias, Y. Avrithis and O. Chum, "Label propagation for deep semi-supervised learning," in *IEEE/CVF Conf. on Computer Vision and Pattern Recognition*, Long Beach, USA, pp. 5070–5079, 2019.
[43] Z. Y. Zhao, Z. Y. Wu, X. Y. Wu, C. Y. Zhang and S. Wang, "Crossmodal few-shot 3D point cloud semantic segmentation," in *30th ACM Int. Conf. on Multimedia*, Lisbon, Portugal, pp. 4760–4768, 2022.
[44] X. L. Li and P. Liang, "Prefix-tuning: Optimizing continuous prompts for generation," arXiv preprint arXiv:2101.00190, 2021.