A Closer Look at Few-Shot 3D Point Cloud Classification

Chuangguan Ye[1] · Hongyuan Zhu[2] · Bo Zhang[3] · Tao Chen[1]

Abstract

In recent years, research on few-shot learning (FSL) has been fast-growing in the 2D image domain due to the less requirement for labeled training data and greater generalization for novel classes. However, its application in 3D point cloud data is relatively under-explored. Not only need to distinguish unseen classes as in the 2D domain, 3D FSL is more challenging in terms of irregular structures, subtle inter-class differences, and high intra-class variances when trained on a low number of data. Moreover, different architectures and learning algorithms make it difficult to study the effectiveness of existing 2D FSL algorithms when migrating to the 3D domain. In this work, for the first time, we perform systematic and extensive investigations of directly applying recent 2D FSL works to 3D point cloud related backbone networks and thus suggest a strong learning baseline for few-shot 3D point cloud classification. Furthermore, we propose a new network, Point-cloud Correlation Interaction (PCIA), with three novel plug-and-play components called Salient-Part Fusion (SPF) module, Self-Channel Interaction Plus (SCI+) module, and Cross-Instance Fusion Plus (CIF+) module to obtain more representative embeddings and improve the feature distinction. These modules can be inserted into most FSL algorithms with minor changes and significantly improve the performance. Experimental results on three benchmark datasets, ModelNet40-FS, ShapeNet70-FS, and ScanObjectNN-FS, demonstrate that our method achieves state-of-the-art performance for the 3D FSL task.

Keywords Machine learning · Few-shot learning · Meta learning · Point cloud classification

1 Introduction

Deep learning models have achieved promising performance on various computer vision (CV) and natural language pro-

✉ Tao Chen
eetchen@fudan.edu.cn

Chuangguan Ye
cgye19@fudan.edu.cn

Hongyuan Zhu
hongyuanzhu.cn@gmail.com

Bo Zhang
zhangbo@pjlab.org.cn

[1] School of Information Science and Technology, Fudan University, Shanghai, China

[2] Institute for Infocomm Research, A*STAR, Singapore, Singapore

[3] Shanghai AI Laboratory, Shanghai, China

cessing (NLP) tasks, with the advent of powerful computing resources and large-scale annotated datasets. The 3D point cloud understanding studies with deep learning techniques are also fast-growing (Qi et al., 2017a, b; Li et al., 2018; Liu et al., 2019b; Zhang et al., 2022; Cosmo et al., 2022). Unlike traditional point cloud recognition algorithms (Johnson and Hebert, 1999; Zhong, 2009; Rusu et al., 2009) that extract shape features based on the hand-crafted operators, deep-learning-based methods can get more informative descriptors for point cloud instances from shape projections (Yu et al., 2018; Qi et al., 2016; Feng et al., 2018) or raw points (Qi et al., 2017a, b; Liu et al., 2019b; Li et al., 2018; Wang et al., 2019b) with deep networks, which achieve better performance.

However, deep-learning-based methods have two crucial issues for point cloud classification. Firstly, the deep-learning technique usually requires large-scale labeled data to train, but it is cumbersome and costly to annotate extensive point cloud data. Secondly, the models trained on base classes often fail to generalize to novel or unseen classes. To overcome the data annotation problem, data-augmentation techniques (Tanner and Wong, 1987), semi-supervised learning (Chapelle et al., 2009), or other learning paradigms (Liao

et al., 2021) have been introduced to reduce laborious annotation by efficiently augmenting labeled examples or reasonably utilizing unlabeled examples. Point-augment (Li et al., 2020b) is proposed by leveraging an auto-augmentation framework to address the point cloud classification problem, which formulates a learnable point augmentation function using a shape-wise transformation and a point-wise displacement. PointMixup (Chen et al., 2020) is to generate new point cloud examples by assigning optimal paths function to mix up two objects. However, they may not obtain promising performance in the unseen or novel classes.

On the other hand, we humans can distinguish novel classes with extremely few labeled examples, which inspires the research in few-shot learning (FSL) (Snell et al., 2017; Sung et al., 2018; Garcia and Bruna, 2017; Vinyals et al., 2016; Ravi and Larochelle, 2017; Finn et al., 2017; Lee et al., 2019; Li et al., 2020a; Ye et al., 2021; Xu et al., 2022). The FSL generalizes deep networks to novel classes with limited annotated data, which has achieved great success in 2D image domain. Generally, most current 2D FSL algorithms adopt metric-based (Snell et al., 2017; Sung et al., 2018; Garcia and Bruna, 2017; Chen et al., 2021a) or optimization-based (Ravi and Larochelle, 2017; Finn et al., 2017; Lee et al., 2019) frameworks to learn transferable knowledge and propagate them to new tasks, e.g., Relation Net (Sung et al., 2018) introduces a learnable non-linear distance metric model to compare query and support images, and MAML (Finn et al., 2017) is to learn a model-agnostic parameter initialization for quickly fine-tuning with few gradient update steps. Moreover, the generalized few-shot learning framework (GFSL) (Ye et al., 2021) and any-shot learning (Xu et al., 2022) are recently introduced to improve the models' generalization ability to novel classes with few training images. However, compared to the success of FSL in the 2D image domain, its study in 3D data is still relatively underexplored with the following two challenges.

First, different from 2D images defined in structured grids with rich texture and appearance information, a point cloud instance is a group of irregular, unordered, and unstructured points in 3D Euclidean space (Guo et al., 2020). Such characteristics make it challenging to identify the critical local areas in a point cloud object, increasing the difficulty of extracting the representative and robust object features. Further, when the 3D point cloud meets with few-shot scenarios, to what extent existing point cloud architecture and learning algorithms can perform for 3D FSL is still unknown.

Second, most 2D FSL algorithms can generate more distinguishing embeddings by pre-training deeper networks on large-scale based class data (Chen et al., 2019), e.g., tiered-ImageNet (Ren et al., 2018) contains 608 classes and more than 800,000 examples in total. By contrast, most 3D datasets contain smaller-scale annotated data, e.g., ModelNet40 (Wu et al., 2015) only has a total number of 12,311 examples
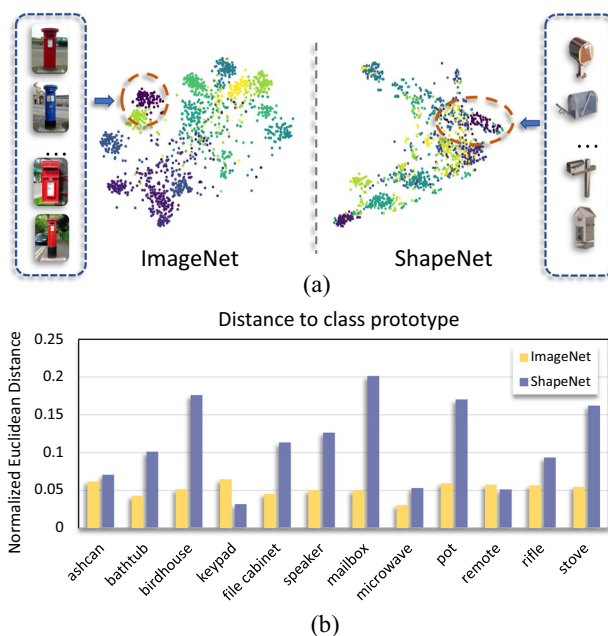

Fig. 1 The challenge of few-shot 3D point cloud classification. **a** Is the T-SNE on ImageNet (Russakovsky et al., 2015) and ShapeNet (Chang et al., 2015) using pre-trained 2D image and 3D point cloud features respectively with the "Mailbox" class highlighted. **b** Shows the mean normalized Euclidean distance between each example and its class prototype in the visual and point cloud embedding space, respectively. The embedding quality of 2D visual domain is much higher than 3D point cloud domains because image-based pre-trained models, such as ResNet (He et al., 2016), use deeper networks trained on millions of images, whereas point cloud-based models, such as PointNet (Qi et al., 2017a), use shallower networks trained on only a few hundred point clouds with subtle inter-class differences and high intra-class variations (Cheraghian et al., 2020)

and 40 classes. Hence the point cloud learning networks trained on a low number of data may generate poor-quality 3D feature clusters, with subtle inter-class differences and high intra-class variations, as shown in Fig. 1. Therefore, how to address these issues requires further exploration.

In this work, for the first time, we systematically study 3D FSL by performing extensive comparison experiments of certain typical 2D FSL algorithms on different kinds of point cloud learning architectures, and provide comprehensive benchmarks and a strong baseline for few-shot point cloud classification. Moreover, we propose a novel method, Point-cloud Correlation Interaction (PCIA), with three plug-and-play components called the Salient-Part Fusion (SPF) module, Self-Channel Interaction Plus (SCI+) module, and Cross-Instance Fusion Plus (CIF+) module to address the issues of subtle inter-class differences and high intra-class variance causing by the limited number of training data. Concretely, the SPF module is designed to generate more representative embeddings for unstructured point cloud instances by fusing local salient-part features. The SCI+ module and CIF+ module can further improve the feature distinction

by exploring the self-channel and cross-instances correlations. Extensive experiments on the three newly proposed 3D FSL benchmark datasets, including ModelNet40-FS, ShapeNet70-FS, and ScanObjectNN-FS, show that our proposed modules can be inserted into certain FSL algorithms flexibly with minor changes and significantly improve their performance.

Our key contributions in this work can be briefly summarized as follows:

– We provide systematic and comprehensive benchmarks and studies for 3D FSL, in terms of the different FSL learning frameworks and point cloud network architectures. We suggest a strong baseline for the few-shot 3D point cloud classification task.
– To address the challenges of 3D FSL, we propose three plug-and-play components, including the Salient-Part Fusion (SPF) module, Self Channel Interaction Plus (SCI+) module, and Cross Instance Fusion Plus (CIF+) module, which can be inserted into certain FSL algorithms with significant performance improvement.
– Our proposed network achieves the state-of-the-art performance on the three newly split benchmark datasets of 3D FSL, including two synthetic datasets, ModelNet40-FS and ShapeNet70-FS, and a real-world scanning dataset ScanobjectNN-FS, which improves the performance by about 10% for 1-shot and 4% for 5-shot.

Note that this paper is an extended version of our recent conference paper (Ye et al., 2022). The significant extensions include: (1) constructing an additional benchmark for few-shot 3D point cloud classification on a real-world scanning dataset ScanobjectNN-FS, and adding several recent relevant SOTA methods as comparison baselines [e.g., Point-BERT (Yu et al., 2021)]; (2) designing a new Salient-Part Fusion (SPF) module to capture the fine details of a point cloud object, by fusing local salient-part features (see Sect. 4.2); (3) proposing a new Self Channel Interaction Plus (SCI+) module to mine the task-related information, and use it to enhance the inter-class difference by channel interaction (see Sect. 4.3); (4) extending the Cross-Instance Fusion (CIF) module with a new branch further exploring the correlation among different instances, to become a new Cross-Instance Fusion Plus (CIF+) module (see Sect. 4.4); (5) providing more extensive experiments (see Sect. 5.2) and deep analyses (see Sects. 5.3–5.5) on model performance, complexity, robustness, etc.

We organize the rest of the paper as follows. Section 2 formulates the problem definition of 3D FSL and reviews the related works for 3D point cloud classification and 2D FSL. Section 3 performs systematic benchmark to investigate the performance of recent state-of-the-art 2D FSL algorithms on 3D data with different kinds of point cloud

learning backbones. Section 4 is to introduce our pipeline for 3D FSL and more details about the proposed method. Extensive experiments and discussions are presented in Sect. 5. Some concluding remarks are finally drawn in Sect. 6.

2 Problem Definition and Related Work

2.1 Problem Definition

We first define a point cloud instance x with its label y as (x, y), where $x \in \mathbb{R}^{n \times 3}$ is an irregular point set containing n points represented with xyz coordinates. In the customary N-way K-shot Q-query few-shot learning setting (Chen et al., 2019), the aim of FSL algorithms is to meta-train a predictor which can be generalized to new unlabeled query examples by few labeled support examples. We denote the labeled support examples as support set $\mathcal{S} = \{(x_i, y_i)\}_{i=1}^{N_s = N \times K}$, containing N classes with K examples for each class, and denote the unlabeled query examples as query set $\mathcal{Q} = \{(x_i, y_i)\}_{i=1}^{N_q = N \times Q}$, containing the same N classes with Q examples for each class.

For few-shot point cloud classification, we adopt the meta-learning paradigm with a set of meta-training episodes $\mathcal{T} = \{(\mathcal{S}_i, \mathcal{Q}_i)\}_{i=1}^{I}$ by optimizing following objectives:

$$\theta^*, \phi^* = \arg\min_{\theta, \phi} \mathcal{L}(\mathcal{T}; \theta, \phi), \tag{1}$$

where $\mathcal{T}$ are sampled from the training set and $\mathcal{L}$ denotes the cross-entropy loss function defined as:

$$\mathcal{L}(\mathcal{T}; \theta, \phi) = \mathbb{E}_{\mathcal{T}} \left[-\log p(\hat{y} = c|x) \right], \tag{2}$$

with the prediction $p(\hat{y} = c|x)$ can be given by:

$$p(\hat{y} = c|x) = softmax\left(\mathcal{C}_\theta\left(\mathcal{F}_\phi(x)\right)\right), \tag{3}$$

where x is the input point cloud instance, $\hat{y}$ is the predicted label and c is the ground truth label. The $\mathcal{F}$ is the backbone network parameterized by ϕ and $\mathcal{C}$ is the classifier parameterized by θ.

Once meta-training is finished, generalization of the predictor is evaluated on meta-testing episodes $\mathcal{V} = \{(\mathcal{S}_j, \mathcal{Q}_j)\}_{j=1}^{J}$, which are sampled from the testing set. Note that we denote the classes in $\mathcal{T}$ as base classes and the classes in $\mathcal{V}$ as novel classes. The base classes are disjoint with the novel classes.

Therefore, there are two crucial challenges in Point Cloud FSL: (1) how to properly represent the point cloud data for few-shot learning; (2) how to effectively transfer the knowledge gained in meta-training episodes to meta-testing episodes, in the case of subtle inter-class differences and high

intra-class variance. In the following section, we will review the recent efforts in 3D point cloud classification and 2D few-shot learning.

2.2 Related Work

2.2.1 3D Point Cloud Classification

Unlike the traditional point cloud recognition methods (Johnson and Hebert, 1999; Zhong, 2009), extracting features based on the hand-crafted operators, deep-learning-based algorithms can get more informative descriptors for point cloud instances with deep networks, which has attracted much attention. Many studies have made significant progress in adopting deep networks for point cloud recognition. According to the fashion of feature extracting from structured data or unordered points, existing deep-learning-based algorithms can be divided into two major categories, the projection-based methods and the point-based methods.

The projection-based methods first convert the irregular points into structured representations, such as multi-view images (Su et al., 2015; Yu et al., 2018; Chen et al., 2018), voxel (Maturana and Scherer, 2015; Riegler et al., 2017), hash tables (Shao et al., 2018) or lattices (Su et al., 2018; Rao et al., 2019), and then extract view-wise or structural features with the typical 2D or 3D CNN networks. VERAM (Chen et al., 2018) presents a view-enhanced recurrent attention model to select a sequence of views for 3D classification. OctNet (Riegler et al., 2017) designs a hybrid grid-octree to represent point cloud objects, and H-CNN (Shao et al., 2018) uses hierarchical hash tables and convolutional neural networks to facilitate shape analysis. However, these approaches may encounter the issue of explicit information loss during the conversion phase or higher memory consumption during the feature learning phase (Guo et al., 2020).

The point-based methods, in contrast, directly take the irregular points as input and embed the point cloud object by exploring the point-wise relationship with deep networks. PointNet (Qi et al., 2017a) is the first model to extract point cloud features using a deep network on raw points. Furthermore, the following works, such as Point-Net++ (Qi et al., 2017b), PointCNN (Li et al., 2018), RSCNN (Liu et al., 2019b), DensePoint (Liu et al., 2019a), and DGCNN (Wang et al., 2019b),further capture the local structural or geometric information using convolution-base networks or graph-based networks for learning more representative features. Moreover, RIConv++ (Zhang et al., 2022) introduces a Rotation-Invariant Convolution by modeling the internal relationship between the selected points and their local neighbors to improve feature descriptions. However, these deep-learning-based methods require large-scale labeled data to train. The recognition ability of deep networks depends on the contents of the training data, and hence they

may have a poor generalization of novel classes never seen before.

Furthermore, recent works (Sharma and Kaul, 2020; Stojanov et al., 2021; Yu et al., 2021) attempt to improve the learning ability of deep networks in 3D point cloud learning with few labeled training data (Sharma and Kaul, 2020) introduces a self-supervised pre-training method to obtain point-wise features for few-shot point cloud learning. LSSB (Stojanov et al., 2021) takes SimpleShot (Wang et al., 2019a) as the baseline, and improves the low-shot image classification generalization performance by learning a discriminative embedding space with 3D object shape bias. Point-BERT (Yu et al., 2021) proposes to generalize the Transformers (Vaswani et al., 2017) architecture to 3D point cloud, and introduces a BERT-style (Devlin et al., 2018) pre-training technique to pre-train a Transformers-based network with a Mask-Point-Modeling (MPM) task, achieving the SOTA performance on several downstream tasks (Feng et al., 2022) is to propose a network for few-shot point cloud classification by designing a feature supplement module to enrich the geometric information and using a channel-wise attention module to aggregate multi-scale features. However, they calculate the channel-wise attention with points' feature directly, which may encounter the issue of larger network parameters and higher computational consumption. Besides, the attention mechanism in Feng et al. (2022) only considers the point-wise relationship in a single sample and neglect the correlation between different samples.

Different from these works, we provide comprehensive benchmarks and studies for few-shot point cloud classification, by systematically reviewing certain typical FSL algorithms migrating to different 3D point cloud learning networks. We also propose three effective and light modules for 3D FSL, including the Salient-Part Fusion (SPF) module, the Self-Channel Interaction Plus (SCI+) module and the Cross-Instance Fusion Plus (CIF+) module, which can be easily inserted into the most FSL algorithms with significant performance improvement.

2.2.2 2D Few-Shot Learning

Recently, research on few-shot learning (FSL) has been fast-growing in the 2D image domain due to the less demand for labeled training data and better generalization for novel classes. Generally, most 2D FSL algorithms adopt a meta-learning strategy to learn the transferable meta-knowledge from base classes to novel classes, roughly divided into: metric-based approaches and optimization-based approaches.

The metric-based approaches try to learn a distinguishing feature space in which examples belonging to the same class are closer, or design efficient metric functions to measure the similarity of features. Matching Net (Vinyals et al., 2016)

designs a bidirectional Long-Shot-Term Memory (LSTM) module to obtain global contextual embeddings and predicts the query examples' label with the Cosine Similarity Distance. Prototypical Net (Snell et al., 2017) introduces the concept of class-prototype, which is the mean of the support features of each class, and uses the Squared Euclidean Distance to classify query examples, demonstrating better performance than Matching Net (Vinyals et al., 2016). Moreover, Relation Net (Sung et al., 2018) and FSLGNN (Garcia and Bruna, 2017) further propose learnable metric modules to obtain relation scores or model the instance-wise relationship of labeled support examples and query examples.

On the other hand, the optimization-based approaches attempt to fast adapt network parameters or optimization strategies to novel tasks with few gradient update steps. Meta-leaner (Ravi and Larochelle, 2017) replaces the stochastic gradient descent optimizer with an LSTM-based model. MAML (Finn et al., 2017) tries to learn a transferable parameter initialization that can be fast generalized to new tasks with few fine-tune steps. MetaOptNet (Lee et al., 2019) meta-trains a feature-relevant Support-Vector Machine (SVM) predictor for query examples by incorporating a differentiable quadratic programming solver.

Recently, many efforts have devoted to further promote the 2D FSL. Mangla et al. (2019) proposes the S2M2 to use self-supervision and regularization techniques to learn relevant feature manifold for FSL. Doersch et al. (2020) proposes the CrossTransformers to design a Transformer based network to find spatially-corresponding between query and support samples. Chen et al. (2021b) is to introduce Meta-baseline with a cosine metric classifier with learnable weight, showing better performance than Squared Euclidean Distance. Luo et al. (2022) proposes SimpleTrans to use a simple transform function to adjust the weight of different channel to alleviate the channel bias problem in FSL.

Differently, in this work, we empirically conduct a comprehensive study of different 2D FSL approaches migrating to few-shot point cloud classification tasks, and suggest a strong baseline for 3D FSL validated on three new splits of benchmark datasets.

3 Systematic Benchmark

This section introduces three new splits of benchmark datasets for 3D FSL and systematically studies certain typical 2D FSL methods on these benchmark datasets with different typical point-based network backbones. The details of experimental settings and implementation are described in Sect. 5.

3.1 Benchmark Datasets for 3D FSL

ModelNet40 (Wu et al., 2015) and ShapeNetCore (Chang et al., 2015) are two widely used 3D synthetic model benchmark datasets for general 3D point cloud classification, containing 40 and 55 categories with a total number of 12,311 and 51,192 objects, respectively. ScanObjectNN (Uy et al., 2019) is the first real-world point cloud classification dataset containing 15 categories with 2,902 unique objects in total, which are selected from the indoor datasets [SceneNN (Uy et al., 2019) and ScanNet (Dai et al., 2017)].

However, there are two issues in these datasets for 3D FSL: (1) existing splits of the training set and testing set contain overlapping classes, which should be disjunctive in the few-shot setting (Vinyals et al., 2016); (2) the number of examples in each class is imbalanced. To meet the FSL setting requirement and evaluate the performance objectively, we introduce new splits of these datasets and construct three benchmark datasets, ModelNet40-FS, ShapeNet70-FS and ScanObjectNN-FS, for 3D few-shot point cloud classification under different scenarios. Statistics of the proposed benchmark datasets are listed in Table 1.

ModelNet40-FS includes the same 40 classes proposed in the ModleNet40 (Wu et al., 2015), and we carefully select 30 classes as the base class set for training and the other 10 classes as the novel class set for testing, to ensure that all the base classes are sufficiently distinct from the novel classes.

ShapeNet70-FS is a larger dataset for 3D FSL. We first select 48 categories with a sufficient example size (at least 80 examples in a class) from ShapeNetCore (Chang et al., 2015), and then increase the class number to 70 by splitting some broad categories into several fine-grained subcategories according to the taxonomy offered by Chang et al. (2015). After that, these are split into 50 base classes for training and 20 novel classes for testing. Note that fine-grained classes separated from the same super category are all in the training

Table 1 Statistics of the new split benchmark datasets for few-shot point cloud classification

Bnechmark Datasets for 3D FSL		Train	Test	Total
ModelNet40-FS	Classes	30	10	40
	Instances	9204	3104	12,308
ShapeNet70-FS	Categories	34	14	48
	Classes	50	20	70
	Instances	21,722	8351	30,073
ScanObjectNN-FS	Classes	10	5	15
	Instances(S^0)	9509	4789	14,298
	Instances(S^1)	9129	5169	14,298
	Instances(S^2)	9958	4340	14,298

S^i denotes the i split of ScanObjectNN-FS

set or testing set, ensuring that the test classes are not similar to those seen during training.

ScanObjectNN-FS is a more challenging dataset for 3D FSL, because the objects scanned from real-world usually contain cluttered background noise and large perturbations. There are five variants in ScanObjectNN (Uy et al., 2019) with different degrees of perturbation. We choose the most challenging variant, PB_T50_RS, to construct the ScanObjectNN-FS, which consists of 15 categories with 14,298 perturbed objects in total. We evenly split the object classes into three non-overlapping subsets, and each split contains 5 classes.

3.2 Baselines

In terms of the few-shot point cloud classification, we first study six widely used SOTA 2D FSL methods, including three metric-based methods and three optimization-based methods. Here we introduce more adapting details about these FSL algorithms migrating to 3D point cloud data.

Metric-based methods: following the original setting proposed in *Prototypical Net* (Snell et al., 2017), we first average the support features as class prototype features and then take the Squared Euclidean Distance to classify the query examples. We train the network end-to-end by minimizing the cross-entropy loss. *Relation Net* (Sung et al., 2018) proposes a learnable relation module to predict the relation score of support-query pairs. Similar to Sung et al. (2018), we design a learnable relation module consisting of two convolutional blocks ((1×1 Conv with 128 filters, BN, ReLU), (1×1 Conv with one filter, BN, ReLU)) and two Fully-Connection Layers (128, N) (N is the number of classes at a meta-episode) and use the Mean Square Error loss to regress relation score to ground truth. *FSLGNN* (Garcia and Bruna, 2017) first introduces a graphical model into 2D FSL, which is easily extended to variants of frameworks. We construct four Edge-Conv layers with the setting introduced in Garcia and Bruna (2017) and adopt the Cross-Entropy loss as the optimal objective.

Optimization-based methods: *MAML* (Finn et al., 2017) aims to learn a potential parameter initialization that can be quickly generalized to novel classes with few gradient update steps. We adopt two FC layers (256, N) as the classifier and update the gradient twice during the meta-training stage. *Meta-learner* (Ravi and Larochelle, 2017) proposes to learn an optimizer for novel examples, which takes two LSTM layers as a meta-learner to replace the stochastic gradient descent optimizer. We follow the same settings introduced in Ravi and Larochelle (2017) and design two LSTM layers for gradient updating. *MetaOptNet* (Lee et al., 2019) tries to generalize to novel categories with a linear classifier, by using implicit differentiation and dual formulation. We use the multi-class

SVM presented in Crammer and Singer (2001) as the classification head.

3.3 Training and Testing Strategy

We train and test the network with the standard episode-based FSL setting (Chen et al., 2019). Firstly, we train the network from scratch with 80 epochs. Each epoch contains 400 meta-training episodes and 600 validating episodes, which are sampled from the training set data and validating set data randomly. Each episode contains N classes with K labeled support examples and Q query examples for each class, denoted as the N-way K-shot Q-query setting. Once the meta-training is ended, we test the network with 700 episodes randomly sampled from the testing set data with the same N-way K-shot Q-query setting, and report the mean classification results of the 700 episodes with 95% confidence intervals. During the testing stage, we also fix the random seed to ensure that all the methods can be compared fairly.

Specially, we employ the fivefold cross-validation training strategy for ModelNet40-FS and ShapeNet70-FS, where we randomly divided the training data into five even subsets. Each subset is used at a time as validation data to select the best-validation model. Then, we test the best-validation models and the last-training models on testing data, and take the better average performance of the five estimations as the final results. For ScanObjectNN-FS, we perform 3-fold cross-validation by selecting one split as the testing set and taking the remaining splits as the training set. Therefore, there are three different splits of the ScanObjectNN-FS, denoted as "S^0", "S^1", and "S^2", respectively.

3.4 State-of-the-Art 2D FSL on Point Cloud

Firstly, we investigate the performance of directly applying widely used SOTA 2D FSL algorithms to the few-shot point cloud classification task on our newly split benchmark datasets. Given its simplicity and efficiency, we adopt Point-Net (Qi et al., 2017a) as the embedding backbone network, and divide the state-of-the-arts into the following groups:

- Metric-based approaches M: prototypical Net (Snell et al., 2017), Relation Net (Sung et al., 2018), FSLGNN (Garcia and Bruna, 2017)
- Optimization-based approaches O: meta-learner (Ravi and Larochelle, 2017), MAML (Finn et al., 2017), MetaOptNet (Lee et al., 2019)

We conduct the experiments under a canonical 5-way setting presented in Sect. 5.1. The comparison results of mean accuracy and complexity, reported in Table 2, show that the metric-based approaches outperform the optimization-based

Table 2 Few-shot point cloud classification accuracy (%) on three newly proposed benchmark datasets with PointNet (Qi et al., 2017a) as the backbone network

Method		ModelNet40-FS		ShapeNet70-FS		ScanObjectNN-FS		5way-1shot-15query		
		5way-1shot	5way-5shot	5way-1shot	5way-5shot	5way-1shot	5way-5shot	PN	GFLOPs	TPS
M	Prototypical Net (Snell et al., 2017)	**65.31 ± 0.78**	79.04 ± 0.54	**65.96 ± 0.81**	**78.77 ± 0.67**	44.75 ± 0.31	**59.81 ± 0.26**	**0.15 M**	**6.16**	**118.23**
	Relation Net (Sung et al., 2018)	64.10 ± 0.72	75.75 ± 0.57	65.88 ± 0.85	76.25 ± 0.71	45.32 ± 0.32	55.43 ± 0.26	0.28M	6.50	96.03
	FSLGNN (Garcia and Bruna, 2017)	59.69 ± 0.73	76.06 ± 0.63	64.98 ± 0.84	76.14 ± 0.73	29.91 ± 0.39	32.77 ± 0.33	2.23M	10.44	69.02
O	Meta-learner (Ravi and Larochelle, 2017)	58.69 ± 0.81	76.60 ± 0.65	62.64 ± 0.91	73.10 ± 0.80	43.79 ± 0.34	58.89 ± 0.26	0.88M	6.28	9.14
	MAML (Finn et al., 2017)	57.58 ± 0.89	77.95 ± 0.62	59.20 ± 0.88	75.10 ± 0.75	**46.65 ± 0.37**	59.55 ± 0.23	0.68M	6.19	38.71
	MetaOptNet (Lee et al., 2019)	64.99 ± 0.87	**79.54 ± 0.61**	65.08 ± 0.89	77.81 ± 0.75	45.02 ± 0.31	59.39 ± 0.24	0.15 M	6.16	11.95

All the algorithms are tested on 700 meta-testing episodes with 95% confidence intervals. Bold means the best result and underline means the second-best. The *M* are the metric-based approaches and *O* are the optimization-based approaches. PN: parameter number. GFLOPs: the number of floating-point operations. TPS: inference tasks per second on one NVIDIA 2080Ti GPU

approaches in most few-shot classification settings on 3D point cloud data.

The reason may be that the optimization-based approaches are more sensitive to the architecture of neural networks and need laborious hyper-parameter adjusting to get good generalization, as discussed in recent research in 2D image domains (Antoniou et al., 2018). The optimization-based methods also have lower inference speed because they need to update the network's parameters in the inference stage.

Moreover, Prototypical Net (Snell et al., 2017) with Point-Net (Qi et al., 2017a) as backbone can achieve the top performance of 65.31, 65.96 and 59.81% respectively for all datasets, with less computational consumption and high inference speed. Noted that Prototypical Net (Snell et al., 2017) and MetaOptNet (Lee et al., 2019) have the same parameter size, as they share the same backbone network and use the parameter-free Square Euclidean Distance and Support Vector Machine (SVM) as the meta classifier, respectively. However, solving the quadratic programming of SVM in MetaOptNet (Lee et al., 2019) is very computationally expensive and time-consuming.

In summary, Prototypical Net (Snell et al., 2017) has a better trade-off between performance and complexity, but there is still a large room for further improvement.

3.5 Influence of Backbone Architecture on FSL

We further analyse the influence of different point cloud backbones on 3D FSL by comparing the vanilla few-shot classification to Prototypical Net (Snell et al., 2017). We select three types of state-of-the-art 3D networks including:

– Pointwise-based: PointNet (Qi et al., 2017a) and Point-Net++ (Qi et al., 2017b).
– Convolution-based: PointCNN (Li et al., 2018), RSCNN (Liu et al., 2019b) and DensePoint (Liu et al., 2019a).
– Graph-based: DGCNN (Wang et al., 2019b).

For vanilla few-shot classification, we first pre-train these classification networks (including their backbone and the classifier head) on base class data without any architecture change, and then evaluate the classification results on novel class data by fine-tuning the classifier head with labeled support samples in each meta-testing episode. Specifically, We pre-train the networks with their default parameter settings for 100 epochs on each dataset and fine-tune the classifier head for 40 epochs.

For Prototypical Net (Snell et al., 2017), we remove the last classifier layers of these networks and train the backbone from scratch with the meta-learning paradigm introduced in Sect. 3.3. Then, embeddings extracted by these backbones are fed into the Prototypical Net (Snell et al., 2017) for the few-shot classification.

Table 3 Comparison results (accuracy %) of vanilla few-shot classification (noted as ✻) and Prototypical Net (Snell et al., 2017) (noted as ‡) on three newly proposed benchmark datasets with different backbones

Method	ModelNet40-FS		ShapeNet70-FS		ScanObjectNN-FS	
	5w-1s	5w-5s	5w-1s	5w-5s	5w-1s	5w-5s
✻PointNet (Qi et al., 2017a)	60.03	78.25	60.53	77.91	40.00	56.83
‡PointNet (Qi et al., 2017a)	**65.31**	**79.04**	**65.96**	**78.77**	**44.75**	**59.81**
✻PointNet++ (Qi et al., 2017b)	61.18	82.31	50.85	79.10	34.30	57.74
‡PointNet++ (Qi et al., 2017b)	**64.96**	**83.66**	**66.33**	**80.95**	**46.83**	**62.27**
✻PointCNN (Li et al., 2018)	59.28	74.77	48.94	75.96	29.78	56.68
‡PointCNN (Li et al., 2018)	**60.38**	**76.95**	**64.02**	**76.34**	**46.09**	**59.11**
✻RSCNN (Liu et al., 2019b)	61.29	80.06	57.37	77.96	31.88	53.81
‡RSCNN (Liu et al., 2019b)	**69.72**	**84.79**	**68.66**	**82.55**	**48.58**	**63.89**
✻DensePoint (Liu et al., 2019a)	57.30	78.06	58.42	79.43	30.10	49.91
‡DensePoint (Liu et al., 2019a)	**66.99**	**82.85**	**65.81**	**80.74**	**43.68**	**57.36**
✻DGCNN (Wang et al., 2019b)	63.52	84.00	61.57	81.55	42.46	60.98
‡DGCNN (Wang et al., 2019b)	**69.95**	**85.51**	**69.03**	**82.08**	**46.28**	**63.94**

Improved results are given in bold

Table 3 reports the comparison results of vanilla few-shot classification (noted as ✻) and the prototypical Net (Snell et al., 2017) (noted as ‡). One can clearly obverse that the DGCNN (Wang et al., 2019b) outperforms other backbone networks in the vanilla classification setting, illustrating that the DGCNN (Wang et al., 2019b) has good potential in feature extraction and generalization. Moreover, the graph-based network DGCNN (Wang et al., 2019b) also achieves better performance under most settings in the Prototypical Net (Snell et al., 2017). The reason can be attributed to the dynamic update of the point-wise connection graph in the feature space and the efficient feature extracting in the Edge-Conv layers. Thus the DGCNN (Wang et al., 2019b) can obtain more informative features compared with other backbone networks.

Besides, the Prototypical Net (Snell et al., 2017) is found to significantly improve the performance in some settings and provide higher performance gains for weaker backbones, for example, 17% for RSCNN (Liu et al., 2019b) and 16% for PointCNN (Li et al., 2018) under the 5w-1s setting on ScanObjectNN-FS dataset. Therefore, we use Prototypical Net (Snell et al., 2017) with DGCNN (Wang et al., 2019b) as the strong baseline for 3D FSL.

4 Approach

4.1 Overview

Given the baselines in the previous section, 3D FSL still faces certain challenges: (1) most existing models are developed under a lab-controlled assumption, where point clouds are uniformly sampled from the surface of synthetic CAD models without any perturbation. Actually, point clouds collected from the real-world environment usually are incomplete and contain cluttered background points; (2) clusters of 3D fea-

tures tend to have subtle inter-class differences and strong intra-class variations when the amount of training data is low; (3) the aforementioned methods independently extract features from support set and query set, without considering the correlations between these two sets. Therefore they may suffer from a huge distribution shift, as demonstrated in Fig. 3a. To address these issues, we create a new and stronger network, Point-cloud Correlation Interaction (PCIA), for 3D FSL classification, as illustrated in Fig. 2.

For the first challenge, we propose a novel salient part fusion (SPF) module to obtain an informative global shape descriptor for unstructured point cloud instances. Specifically, the backbone network $\mathcal{F}_\phi$ takes support set $\mathcal{S}$ and query set $\mathcal{Q}$ as input, and embeds each instance $x \in \mathbb{R}^{n \times 3}$ into a d-dimension point-wise feature map $\boldsymbol{F}_{pw} = \mathcal{F}_\phi(x) = \{\boldsymbol{f}_{pw}^i\} \in \mathbb{R}^{n \times d}$, where $i \in [1, n]$. The feature map $\boldsymbol{F}_{pw}$ are then fed into the proposed SPF module in Sect. 4.2, to fuse local salient features to obtain an informative global shape descriptor $\boldsymbol{f}_g = SPF(\boldsymbol{F}_{pw}) \in \mathbb{R}^{1 \times d}$. After that, we define the prototype feature for class c_i in support set $\mathcal{S}$ as $\boldsymbol{f}_p^i = \frac{1}{|K|} \Sigma_{x_s^k \in \mathcal{S}^{c_i}} SPF(\mathcal{F}_\phi(x_s^k)) \in \mathbb{R}^{1 \times d}$ as the average of its K support examples $\mathcal{S}^{c_i}$, where $\mathcal{S}^{c_i} = \{x_s^1, x_s^2, .., x_s^K\}$, and the query feature for a query example x_q^j as $\boldsymbol{f}_q^j = SPF(\mathcal{F}_\phi(x_q^j)) \in \mathbb{R}^{1 \times d}$, where $i \in [1, N]$ and $j \in [1, N_q]$.

However, the first step only separately extracts features from the support set and query set, yet ignores mitigating the feature distribution gap by mining the correlation between these two sets. To address these challenges, our recent work (Ye et al., 2022) proposed two efficient components called Self-Channel Interaction (SCI) module and Cross-Instance Fusion (CIF) module to update the prototype feature $\boldsymbol{f}_p^i$ and query feature $\boldsymbol{f}_q^j$ by considering the channel-wise and instance-wise correlation, which can
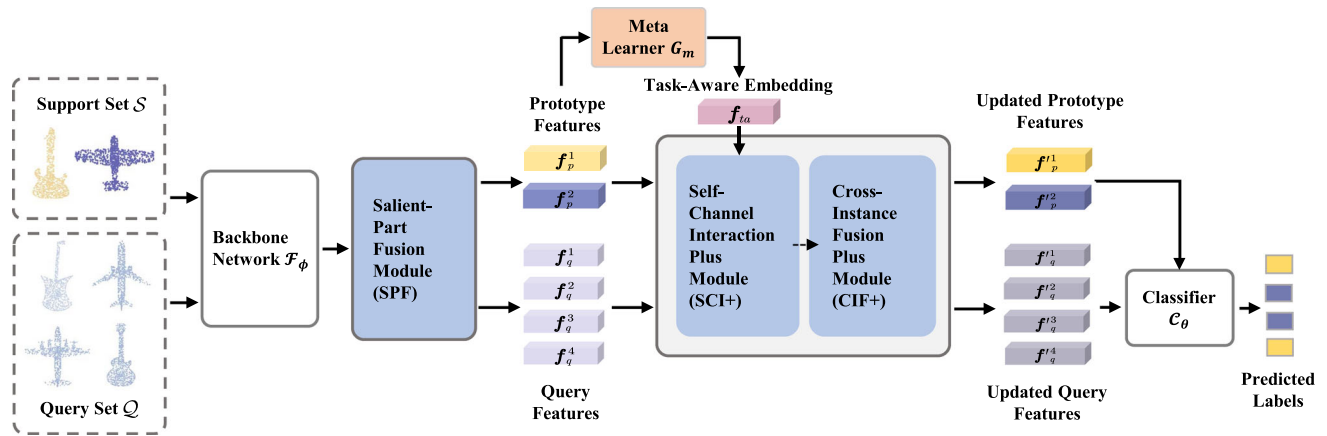
Fig. 2 The schematic of our proposed PCIA framework for 3D few-shot point cloud classification, which consists of three main components for addressing the challenges in 3D FSL: Salient-Part Fusion (SPF) mod-ule, Self-Channel Interaction Plus (SCI+) module and Cross-Instance Fusion Plus (CIF+) module. For clarity, only the 2-way 1-shot 2-query setting is presented here
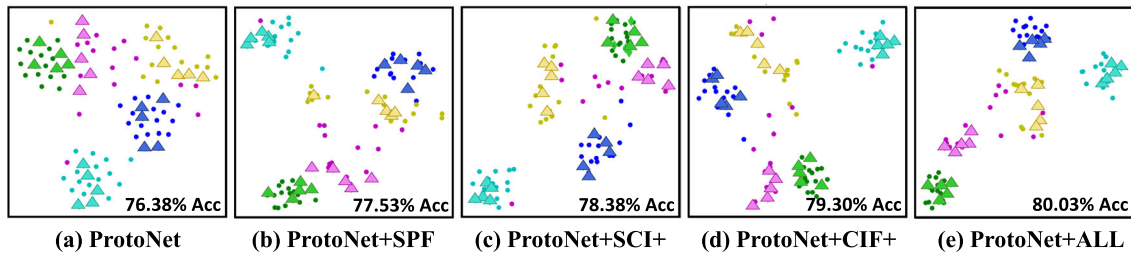

Fig. 3 The t-SNE comparisons of feature distribution on the same episode after using different modules. △ stands for the support features, ● represents the query features. We use the same color to represent the same class

be easily inserted into different baselines with significant performance improvement. However, they ignore the local task-related information offered by the labeled support examples and the implicit spatial correlation among the input instances.

Therefore, we newly design a Self-Channel Interaction Plus (SCI+) module in Sect. 4.3 and a Cross-Instance Fusion Plus (CIF+) module in Sect. 4.4 to further refine the prototype feature f_p^i and query feature by considering the shared and complementary information among prototypes, after which more diverse prototype features f'^i_p and query features f'^j_q are generated to improve the distribution of features for better classification, as shown in Fig. 3e.

After that, we take Square Euclidean Distance metric function as classifier $\mathcal{C}_\theta$, and the probability of predicting label $\hat{y}_j$ for f'^j_q as class c_i is denoted as:

$$p\left(\hat{y}_j = c_i | f'^j_q\right) = \frac{\exp\left(-d\left(f'^j_q, f'^i_p\right)\right)}{\sum_{i=1}^N \exp\left(-d\left(f'^j_q, f'^i_p\right)\right)}, \quad (4)$$

where $d\,(.,.)$ is the Square Euclidean Distance, f'^i_p and f'^j_q are the updated features generated by the SCI+ and CIF+module.

In the end, we can get the cross-entropy loss with Eq. (2) and optimize the network end-to-end by minimizing the following equation:

$$L_{CE} = -\frac{1}{N}\frac{1}{N_q}\sum_i^N \sum_j^{N_q} \mathbb{1}\left[y_j = c_i\right] \log\left(p\left(\hat{y}_j = c_i | f'^j_q\right)\right),$$

$$(5)$$

where N and N_q are the number of class prototypes and query examples, respectively, y_j is the ground truth of f'^j_q, and $\mathbb{1}$ denotes the Kronecker delta function.

4.2 Salient-Part Fusion (SPF) Module

Since point cloud instances scanned from real world usually contain cluttered background and large perturbations, we propose the Salient-Part Fusion (SPF) module to capture the fine details, and learn a more informative shape descriptor for each input point cloud instance by fusing the local salient part features with its coarse-grained global shape feature, as illustrated in Fig. 4.
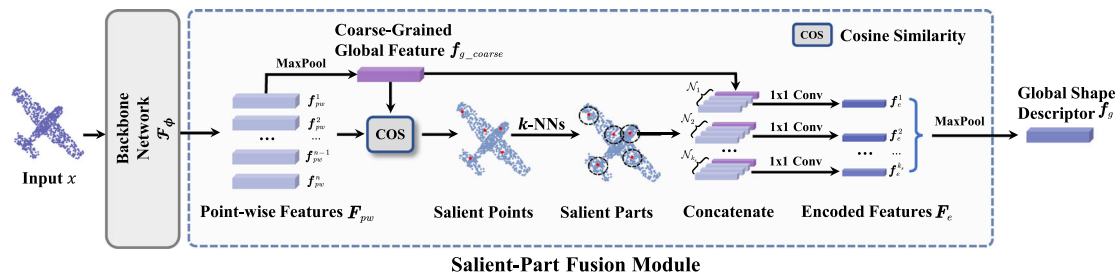
Fig. 4 An illustration of the Salient-Part Fusion (SPF) module. We first calculate a salient score for each point's feature f_{pw}^i, $i \in [1, n]$ between the coarse-grained global descriptor f_{g_coarse} by using cosine similarity. Then we select k_s points with the highest salient score and aggregate their k-nearest neighbors as the local salient part features $\mathcal{N}_j$, $j \in [1, k_s]$. After that, we concatenate the coarse-grained global feature to each local salient part feature, and adopt a 1×1 convolutional layer g to encode the concatenated features as F_e. Finally, we perform a channel-wise Max-pooling function on F_e and obtain the refined global shape descriptor f_g

Considering that points from different parts contribute differently to shape description, we tend to select relevant and salient parts to enhance the discrimination of shape descriptor embedding. To this end, we first define the coarse-grained global feature $f_{g-coarse} \in \mathbb{R}^{1 \times d}$ for the point cloud x, by a channel-wise Max-pooling function performed on F_{pw}:

$$f_{g\_coarse} = Maxpooling\left(F_{pw}\right) \in \mathbb{R}^{1 \times d}, \tag{6}$$

Then, we further define a salient score a_i to quantify the relevance between the ith point's feature f_{pw}^i and the coarse-grained global feature f_{g_coarse}, which can be calculated as:

$$a_i = \mathcal{M}\left(f_{pw}^i, f_{g\_coarse}\right), \tag{7}$$

where $\mathcal{M}$ is the cosine similarity metric function and $i \in [1, n]$. Note that the higher a_i indicates higher likelihood to be the salient points of the input point cloud x.

Given the salient scores, we select k_s highest salient points, and for each salient point p_j (where $j \in [1, k_s]$), we further use its k-nearest neighbors' (k-NNs) features to better describe its local part information as $\mathcal{N}_j \in \mathbb{R}^{k \times d}$. Next, we concatenate the coarse-grained global feature f_{g_coarse} to each local salient part feature $\mathcal{N}_j$, and perform a 1×1 convolution $g(\cdot)$ on the concatenated channels to encode the concatenated features as f_e^j, which is denoted as:

$$f_e^j = g\left([f_{g\_coarse}, \mathcal{N}_j]\right) \in \mathbb{R}^{1 \times d}, \tag{8}$$

Finally, we obtain the refined global shape descriptor f_g, by performing a channel-wise Max-pooling function on $F_e = \{f_e^j\}$ where $j \in [1, k_s]$, which can be described as:

$$f_g = Maxpooling\left(F_e\right) \in \mathbb{R}^{1 \times d}, \tag{9}$$

4.3 Self-Channel Interaction Plus (SCI+) Module

3D FSL faces the challenge of subtle inter-class differences when the network is trained on a small number of data that may contain fine-grained parts. For example, the samples from the "chair" category can come with distinct handles. On the other hand, different channels of a point cloud feature contain various shape structural information. Our previous work (Ye et al., 2022) attempted to enhance discriminative parts, by modeling the instance-wise internal structure relationships with the Self-Channel Interaction (SCI) module.

However, the SCI module only considers the channel attention within one single prototype and ignores the parts shared across other prototypes. For example, "chair" and "stool" can both have samples with similar handles and cushion structures. In order to increase the network's discriminability for structural similar objects from different classes, we hope that such inter-class similar structures could be weaken so that the inter-class differences become more obvious and easy to be distinguished.

So, to mine these shared information across different prototypes, we propose the Self Channel Interaction Plus (SCI+) module to elicit structures shared across prototypes within the same task, and then use the attention mechanism to adjust the weight of different channel.

In particular, we first design a Meta Learner G_m to generate the task-aware embedding f_{ta}, to model the shared context information among f_p^i. Specifically, the generation of f_{ta} can be described as:

$$f_{ta} = G_m([f_p^1, f_p^2, .., f_p^N]) \in \mathbb{R}^{1 \times d}, \tag{10}$$

where $[f_p^1, f_p^2, .., f_p^N] \in \mathbb{R}^{1 \times d \times N}$ is the concatenated feature of the N prototype features, and the G_m is a 1×1 Conv layer performing on the third dimension of the concatenated feature, to encode the N prototype features into a d-dimension task-aware embedding f_{ta}.
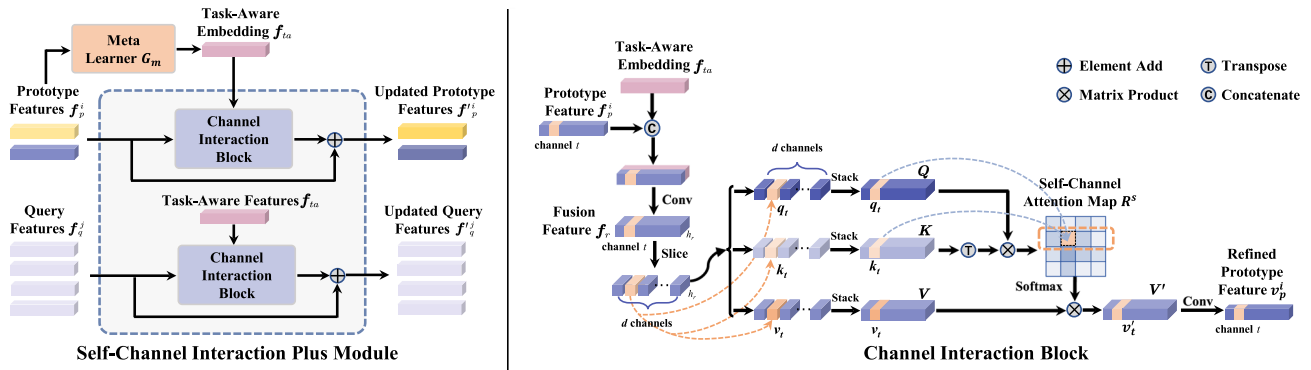
Fig. 5 An illustration of the Self-Channel Interaction Plus (SCI+) module and the Channel Interaction Block (CIB). (1) Firstly, the Meta Learner G_m takes in the prototype features and outputs the task-aware embedding f_{ta}. Then each prototype feature f_p^i (or query feature f_q^j), combined with the task-aware embedding, is fed into the CIB to generate the updated features. (2) In the CIB, we first concatenate the input feature (prototype feature f_p^i or query feature f_q^j) with the task-aware embedding f_{ta} and fuse them with a 1×1 Conv to increase the dimension of each channel. Then we adopt the self-attention mechanism (Vaswani et al., 2017) to obtain a Self-Channel Attention Map R^s and use it to refine the input feature

Then, each prototype feature f_p^i (or query features f_q^j), combined with the task-aware embedding f_{ta}, is fed to the Channel Interaction Block (CIB). We adopt the self-attention mechanism (Vaswani et al., 2017) in CIB to obtain a Self-Channel Attention Map R^s, and use it to refine the input prototype features (or query features) by increasing the weight of discriminative channels and deducing the weight of less discriminaive channels containing class-similar information.

The right part of Fig. 5 illustrates the process of refining prototype feature f_p^i in the CIB, where we first concatenate the f_p^i with the task-aware embedding f_{ta}, and then fuse them using a 1×1 Conv layer to encode each channel into h_r-dimension to get the fusion feature f_r:

$$f_r = Conv([f_p^i, f_{ta}]) \in \mathbb{R}^{1 \times d \times h_r}, \tag{11}$$

where each channel is represented by a h_r-dimension vector that integrates the task-related information.

After that, we slice the f_r along the channel dimension d. For each channel, we generate a query-vector $q_t \in \mathbb{R}^{1 \times h_r}$, a key-vector $k_t \in \mathbb{R}^{1 \times h_r}$ and a value-vector $v_t \in \mathbb{R}^{1 \times h_r}$ using three liner embedding layers, respectively, to extract salient features in each channel. For efficient computation, we pack these vectors of different channels together into matrices $Q \in \mathbb{R}^{1 \times d \times h_r}$, $K \in \mathbb{R}^{1 \times d \times h_r}$ and $V \in \mathbb{R}^{1 \times d \times h_r}$, and adopt the matrix-product to obtain the self-channel attention map R^s:

$$R^s = \frac{QK^T}{\sqrt{h_r}} \in \mathbb{R}^{1 \times d \times d}, \tag{12}$$

and the weighted value-matrix V' is:

$$V' = softmax(R^s)V \in \mathbb{R}^{1 \times d \times h_r}. \tag{13}$$

Finally, we use a 1×1 Conv layer to compress the third dimension of V' from h_r to 1, and reshape it into a $1 \times d$ vector v_p^i. After this we can compensate the discarded information by combining the refined features v_p^i with f_p^i, and get the updated prototype features f'^i_p:

$$f'^i_p = v_p^i + f_p^i \in \mathbb{R}^{1 \times d}, \tag{14}$$

where $v_p^i = Conv(V') \in \mathbb{R}^{1 \times d}$. Similarly, the Eqs. (11)–(14) are also applied to each query feature f_q^j to get the updated feature f'^j_q.

After being inserted with the SCI+ module, the network can learn more fine-grained discriminative features to separate different classes more obviously, as shown in Fig. 3c. More visualization analysis of the SCI+ module can be found in Sect. 5.5.

4.4 Cross-Instance Fusion Plus (CIF+) Module

Most existing methods (Vinyals et al., 2016; Snell et al., 2017; Sung et al., 2018; Garcia and Bruna, 2017) extract support features and query features independently. As a result, there often exist feature distribution shift (as shown in Fig. 3a) between support set and query set. Thus, we propose a simple-but-effective Cross-Instance Fusion (CIF) module in our previous conference paper (Ye et al., 2022). In this work we further explore the implicit spatial information and extend the CIF module by considering the global instance-wise correlation.

Specifically, as illustrated in Fig. 6 (here we only show the process of updating prototype features, and the update of query features is similar with the process), there are two main branches in the new CIF module where the upper branch
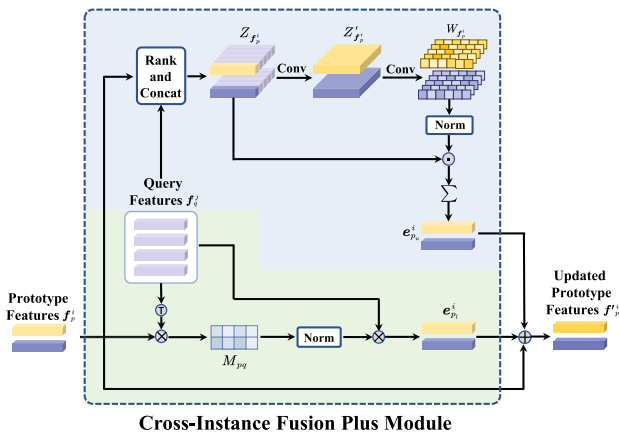
Fig. 6 An illustration of the Cross-Instance Fusion Plus (CIF+) module. The $\odot$ is element-wise product, $\oplus$ is element-wise add and $\otimes$ is matrix product. There are two main branches in this module. The upper branch (masked with blue color) fuses the features based on channel-wise weights, and the lower branch (masked with green color) fuses the features based on the instance-wise weights. Here we only show the process of updating prototype features and the update of query features is similar to this process (Color figure online)

(masked with blue color) fuses the features with channel-wise weights, and the lower branch (masked with green color) fuses the features with the instance-wise weights. Hence, we termed the new CIF module as CIF Plus (CIF+) module.

In the upper branch, we first concatenate each prototype feature f_p^i with its top K_1 similar query features based on the cosine distance, and get $Z_{f_p^i}$:

$$Z_{f_p^i} = \left[f_p^i, f_q^{\langle top1 \rangle}, .., f_q^{\langle topK_1 \rangle} \right] \in \mathbb{R}^{1 \times d \times (K_1+1)}, \qquad (15)$$

where $[\cdot]$ is the concatenation operation, d is the number of feature channels, $K_1 \leq N_q$, and $f_q^{\langle top1 \rangle}$ represents the query feature having the highest cosine similarity with prototype feature f_p^i.

We then employ two 1×1 Conv layers in the concatenation direction to encode the concatenated channels and generate a weight matrix W_{f_p}. After that, we update the prototype features by using the weighted sum of Z_{f_p} with W_{f_p} instead of averaging, which can fuse the channel-wise information flexibly.

Formally, the weight matrix $W_{f_p^i}$ for $Z_{f_p^i}$ is denoted as:

$$W_{f_p^i} = f_2\left(f_1\left(Z_{f_p^i} \right) \right) \in \mathbb{R}^{1 \times d \times (K_1+1)}, \qquad (16)$$

and the fusing feature $e_{p_u}^i$ of the upper branch can be obtained by adding up the K_1+1 concatenated features in $Z_{f_p^i}$ based on $W_{f_p^i}$:

$$e_{p_u}^i = \sum \left(softmax\left(W_{f_p^i} \right) \odot Z_{f_p^i} \right) \in \mathbb{R}^{1 \times d} \qquad (17)$$

where $f_1(\cdot)$ is the first 1×1 Conv layer encoding the $Z_{f_p^i}$ into a h-dim feature interaction $Z'_{f_p^i}$, and the second 1×1 Conv layer $f_2(\cdot)$ is designed to adjust the dimension of interaction $Z'_{f_p^i}$ to generate the weight matrix $W_{f_p^i}$, and $\odot$ is the element-wise product.

However, the 1×1 Convolution used in the upper branch only considers the relation within the concatenated channels, yet ignores the correlation between other channels. To further explore the latent spatial information represented by other channels, we extend the CIF module by adding a new branch (masked with green color in Fig. 6) that introduces instance-wise weights to finely adjust the features.

Specifically, we first stack the N prototype features and the N_q query features into prototype matrix $P_s \in \mathbb{R}^{N \times d}$ and query matrix respectively $Q_s \in \mathbb{R}^{N_q \times d}$, so that we can easily calculate the cross-instance relation map M_{pq} of prototype and query features with matrix-product:

$$M_{pq} = P_s Q_s^T \in \mathbb{R}^{N \times N_q}, \qquad (18)$$

where $N_q = N \times Q$ is the number of total query samples in a meta episode.

Then, we get the fusing features E_{pl} by combining the query features based on the cross-instance relation map M_{pq}:

$$E_{pl} = softmax(M_{pq}) Q_s \in \mathbb{R}^{N \times d}. \qquad (19)$$

Note that, we implement the Eq. (19) by matrix-product for efficiency, so the matrix E_{pl} needs to be reshaped into N d-dimension vectors $e_{pl}^i \in \mathbb{R}^{1 \times d}$, which is the weighted sum of query features based on the M_{pq}.

Finally, we update the prototype feature f_p^i by adding the fusing features $e_{p_u}^i$ and e_{pl}^i:

$$f'_p^i = f_p^i + e_{p_u}^i + e_{pl}^i \in \mathbb{R}^{1 \times d}, \qquad (20)$$

The Eqs. (16)–(20) are similarly applied to the update of query features with prototype features. As shown in Fig. 3d, the feature distribution gap of different classes is mitigated after using the CIF+ module. More visualization analyses of the CIF+ module are reported in Sect. 5.5.

5 Experiments

This section starts with the details about the newly split benchmark datasets and experimental settings, and then reports the comparison results of the SOTA FSL baseline and our proposed method under different scenarios. After that, extensive ablation studies, insightful analyses and qualitative

visualizations are conducted to evaluate the effectiveness of the proposed modules.

5.1 Implementation Details

We take DGCNN (Wang et al., 2019b) as the backbone network for feature embedding, consisting of four EdgeConv layers (64, 64, 128, 256) and an MLP encoded layer. The inputs are a set of meta-tasks containing $N = 5$ classes with $K = 1$ or $K = 5$ support instances and $Q = 15$ query instances for each class, termed the "5-way 1-shot 15-query" or "5-way 5-shot 15-query" setting. We randomly sample 512 points from the CAD model surface or the scanning object for each point cloud example. We use the Adam optimizer with an initial learning rate of 0.0008 and gamma of 0.5. We adopt the early stop strategy that stops the training when the validation accuracy does not increase in 30 epochs, to reduce overfitting. The random points jittering and rotating are used to augment data as in Qi et al. (2017a) during training. Other details of experimental setups are described in the following subsections.

5.2 Experimental Results

5.2.1 Results on Synthetic Point Clouds

We first compare our method with the aforementioned FSL baselines on two synthetic benchmark datasets, ModelNet40-FS and ShapeNet70-FS. Results reported in the first part of Table 4 prove that our proposed method outperforms other baselines on both datasets by a large margin, by about 10% for 1-shot and 4% for 5-shot. One possible explanation may be that the proposed modules can adjust the feature distribution of the support set and query set by considering channel-level and instance-level correlation, enhancing the distinction between prototypes and query instances in the feature space. Thus the metric-based classifier can better distinguish the boundaries of different classes.

The second part of Table 4 lists the comparison results of recent FSL algorithms. For *S2M2* (Mangla et al., 2019), we first pre-train the backbone and classifier head on the base class data with the auxiliary loss (including rotation and exemplar) and cross-entropy loss. After that, we use the support samples to fine-tune the network and predict the query samples' class with a cosine classifier. For Meta-Baseline (Chen et al., 2021b), we train the network from scratch using meta-leaning paradigm with a learnable cosine classifier. In SimpleTrans (Luo et al., 2022), we first use the proposed simple transform function to adjust the feature, and then feed them in the Prototypical Net (Snell et al., 2017) to get predictions.

We also compare our proposed method with four recent 3D learning works that explore low labeled training data

problems in the third part of Table 4. In particular, for the method proposed by Sharma and Kaul (2020), we first pre-train an embedding network using the self-supervised strategy introduced in their paper, and then fine-tune the whole network and classifier with the labeled support examples. *LSSB* (Stojanov et al., 2021) proposed to use 3D object shape bias to improve the generalization performance of low-shot image classification. To compare with this method, we follow the same experimental settings in Stojanov et al. (2021), which first train the *Simpleshot* (Wang et al., 2019a) on 3D models' multi-view RGB projections with ResNet18 (He et al., 2016) as the backbone, and then refine the latent embedding space by using 3D object shape bias, denoted as *Simpleshot+SB* (Stojanov et al., 2021). For *Point-BERT* (Stojanov et al., 2021), we first pre-train the dVAE model and Transformer encoder on ShapeNet55 (Chang et al., 2015), and then fine-tune the model to the few-shot classification task with the same parameter settings adopted in their paper. To compare with the channel-wise attention introduced in the Feng et al. (2022), we directly replace the attention mechanism in the SCI+ module by generating the Q-K-V vector for each point with three FC layers and using them to get the attention score. Then we refine the point-wise feature using this attention score as done in Feng et al. (2022).

The comparison results reported in Table 4 show that our proposed network achieves better performance on both benchmark datasets by about 12% for 1-shot setting and 4% for 5-shot setting on ModelNet40-FS, and by about 9% for 1-shot setting and 3% for 5-shot setting on ShapeNet70-FS, respectively.

5.2.2 Results on Real-World Point Clouds

We further compare the performance of different methods for 3D FSL under a more challenging real-world scenario. Table 5 summarizes the results of competing methods on the ScanObjectNN-FS. We can see that the classification performance drops rapidly in the real-world scenario because the data collected from the real world are noisy and cluttered. However, our method also can achieve the top mean accuracy of 56.24% for 1-shot and 70.93% for 5-shot, outperforming most methods on different data splits with different settings. So our proposed method is indicated to be promising and effective for 3D few-shot point cloud classification. Note that we can not compare with Simpleshot (Wang et al., 2019a) and LSSB (Simpleshot+SB) (Stojanov et al., 2021) on the ScanObjectNN-FS dataset, because they are image-based methods and the dataset does not provide multi-view 2D RGB projection images.

Table 4 Comparison results of few-shot point cloud classification (accuracy %) with 95% confidence intervals on ModelNet40-FS and ShapeNet70-FS with DGCNN (Wang et al., 2019b) as the backbone

Method	ModelNet40-FS		ShapeNet70-FS			5way-1shot-15query		
	5way-1shot	5way-5shot	5way-1shot	5way-5shot		PN	GFLOPs	TPS
Prototypical Net (Snell et al., 2017)	69.95 ± 0.67	85.51 ± 0.52	69.03 ± 0.84	82.08 ± 0.72		**0.61 M**	**96.88**	**12.03**
Relation Net (Sung et al., 2018)	68.57 ± 0.73	82.01 ± 0.53	67.87 ± 0.86	77.99 ± 0.70		0.75 M	97.33	11.08
FSLGNN (Garcia and Bruna, 2017)	61.96 ± 0.76	80.22 ± 0.55	66.25 ± 0.88	76.20 ± 0.77		2.69 M	101.27	10.47
Meta-learner (Ravi and Larochelle, 2017)	59.08 ± 0.86	76.99 ± 0.67	64.53 ± 0.83	74.61 ± 0.72		1.42 M	98.62	4.36
MAML (Finn et al., 2017)	62.57 ± 0.88	77.41 ± 0.73	64.39 ± 0.76	74.11 ± 0.68		1.03 M	97.08	9.38
MetaOptNet (Lee et al., 2019)	67.05 ± 0.78	85.05 ± 0.59	68.27 ± 0.93	81.06 ± 0.76		**0.61 M**	**96.88**	5.68
S2M2 (Mangla et al., 2019)	69.73 ± 0.64	83.25 ± 0.43	68.53 ± 0.73	79.71 ± 0.73		0.63 M	97.32	5.23
Meta-Baseline (Chen et al., 2021b)	71.33 ± 0.34	85.27 ± 0.23	70.16 ± 0.41	81.08 ± 0.33		**0.61 M**	97.03	6.78
SimpleTrans (Luo et al., 2022)	71.44 ± 0.33	86.78 ± 0.22	69.19 ± 0.40	83.37 ± 0.32		**0.61 M**	98.76	5.13
Sharma and Kaul (2020)	64.89 ± 0.82	79.59 ± 0.73	65.76 ± 0.72	79.19 ± 0.71		0.73 M	97.86	9.62
SimpleShot* (Wang et al., 2019a)	61.87 ± 0.79	78.17 ± 0.59	61.58 ± 0.80	74.63 ± 0.63		1.76 M	99.18	11.55
LSSB(SimpleShot+SB)[†] (Stojanov et al., 2021)	63.33 ± 0.75	76.41 ± 0.68	64.45 ± 0.83	73.77 ± 0.73		–	–	–
Point-BERT (Yu et al., 2021)	69.41 ± 3.16	86.83 ± 2.03	73.92 ± 3.60	82.86 ± 2.92		22.04 M	170.10	–
Feng et al. (2022)	61.36 ± 0.41	73.20 ± 0.31	65.09 ± 0.44	75.89 ± 0.29		1.935 M	201.89	6.03
Ours	**81.19 ± 0.64**	**89.30 ± 0.46**	**78.37 ± 0.79**	**85.15 ± 0.70**		0.63 M	97.37	7.49

Bold values denote the best results and underline denotes the second-best

We also report the comparison of parameter number (PN), floating-point operations (GFLOPs), and the number of inference tasks per second (TPS) on one NVIDIA 2080Ti GPU. Note that, *

means network takes 2D images as input, and † means taking 2D images with corresponding point cloud instances as input

Table 5 Comparison results of few-shot point cloud classification results (accuracy %) with 95% confidence intervals on the real-world point cloud classification dataset, ScanobjectNN-FS, with DGCNN (Wang et al., 2019b) as the backbone

Method	ScanObjectNN-FS							
	5way-1shot				5way-5shot			
	S^0	S^1	S^2	Mean	S^0	S^1	S^2	Mean
Prototypical Net (Snell et al., 2017)	45.10 ± 0.77	44.10 ± 0.61	49.64 ± 0.61	46.28	63.91 ± 0.50	64.43 ± 0.50	63.49 ± 0.45	63.94
Relation Net (Sung et al., 2018)	45.46 ± 0.76	45.20 ± 0.52	52.26 ± 0.52	47.64	60.45 ± 0.54	52.93 ± 0.50	63.02 ± 0.47	58.80
FSLGNN (Garcia and Bruna, 2017)	32.09 ± 0.78	29.37 ± 0.51	25.00 ± 0.38	28.82	38.78 ± 0.50	40.41 ± 0.50	26.77 ± 0.30	35.32
Meta-learner (Ravi and Larochelle, 2017)	38.11 ± 0.47	42.05 ± 0.45	41.39 ± 0.42	40.52	51.56 ± 0.42	50.72 ± 0.40	50.96 ± 0.36	51.08
MAML (Finn et al., 2017)	39.12 ± 0.74	45.77 ± 0.62	41.02 ± 0.62	41.95	50.22 ± 0.56	51.38 ± 0.49	49.58 ± 0.28	50.39
MetaOptNet (Lee et al., 2019)	45.98 ± 0.82	48.48 ± 0.64	58.33 ± 0.64	50.93	64.13 ± 0.51	68.27 ± 0.49	71.82 ± 0.47	68.07
S2M2 (Mangla et al., 2019)	43.15 ± 0.57	44.39 ± 0.47	**64.60 ± 0.56**	47.38	61.29 ± 0.64	65.37 ± 0.71	59.52 ± 0.83	62.06
Meta-Baseline (Chen et al., 2021b)	45.75 ± 0.73	47.41 ± 0.59	56.77 ± 0.63	49.97	59.35 ± 0.50	67.52 ± 0.46	67.62 ± 0.43	64.83
SimpleTrans (Luo et al., 2022)	45.63 ± 0.69	44.22 ± 0.57	54.89 ± 0.56	48.25	63.87 ± 0.49	66.07 ± 0.45	**74.74 ± 0.40**	68.23
Sharma et al. (Sharma and Kaul, 2020)	43.75 ± 0.73	40.31 ± 0.75	48.83 ± 0.59	44.29	58.38 ± 0.63	60.31 ± 0.52	62.42 ± 0.50	60.37
Point-BERT (Yu et al., 2021)	33.22 ± 2.99	33.92 ± 2.21	38.17 ± 3.60	35.10	50.32 ± 2.99	49.15 ± 2.53	53.55 ± 3.03	51.01
Feng et al. (Feng et al., 2022)	26.43 ± 0.48	46.37 ± 0.77	49.02 ± 0.57	40.60	56.10 ± 0.55	60.83 ± 0.47	52.68 ± 0.41	56.53
Ours	**48.61 ± 0.75**	**57.96 ± 0.61**	62.16 ± 0.70	**56.24**	**65.96 ± 0.50**	**73.82 ± 0.45**	73.11 ± 0.42	**70.93**

Bold values denote the best results and underline denotes the second-best

S^i denotes the results on split i

Table 6 Comparisons of the classification results (accuracy %) under larger N-way setting on ShapeNet70-FS with DGCNN (Wang et al., 2019b) as the backbone

Method	ShapeNet70-FS					
	10w-1s	10w-5s	15w-1s	15w-5s	20w-1s	20w-5s
Prototypical Net (Snell et al., 2017)	56.08	71.29	49.25	62.20	44.03	58.26
Relation Net (Sung et al., 2018)	53.49	66.14	48.68	58.31	41.55	53.78
MetaOpt (Lee et al., 2019)	55.68	68.33	47.99	60.72	43.30	55.67
Point-BERT (Yu et al., 2021)	_59.52_	_71.33_	_50.15_	**65.52**	_48.33_	**61.68**
Ours	**62.57**	**72.24**	**54.04**	_64.46_	**49.68**	_60.25_

Bold values denote the best results and underline denotes the second-best
N-w-K-s represent the N-way-K-shot setting

Table 7 Ablation study of the proposed SPF, SCI+ and CIF+ modules with DGCNN (Wang et al., 2019b) as the backbone

SPF	SCI+	CIF+	ModelNet40-FS		ShapeNet70-FS		ScanObjectNN-FS	
			5w-1s	5w-5s	5w-1s	5w-5s	5w-1s	5w-5s
			69.95	85.51	69.03	82.08	46.28	63.94
✓			71.89	87.39	69.86	83.28	47.71	67.56
	✓		72.48	87.86	70.02	83.33	47.13	67.93
		✓	78.54	87.89	75.65	84.44	53.11	70.20
✓	✓		73.43	88.31	70.53	83.64	48.74	68.28
✓		✓	79.74	88.11	77.93	84.59	55.87	70.53
	✓	✓	80.83	88.61	77.64	84.79	55.38	70.35
✓	✓	✓	**81.19**	**89.30**	**78.37**	**85.15**	**56.24**	**70.93**

Bold values denote the best results
We take Prototypical Net (Snell et al., 2017) as the baseline (the first row), ✓ means equipped with that block

5.2.3 Larger N-Way Classification

To examine the effect of the proposed network under a practical scenario, we conduct the experiments of the larger N-way setting (where N = 10, 15, 20) on ShapeNet70-FS with the backbone network DGCNN (Wang et al., 2019b). As in Table 6, we compare the metric-based methods Prototypical Net (Snell et al., 2017) and Relation Net (Sung et al., 2018), the optimization-based method MetaOptNet (Lee et al., 2019), and Point-Bert (Yu et al., 2021).

The results show that the performances of these methods drop significantly when increasing the number of N. But our approach compares favorably against other methods in most settings. We attribute the results to that larger N-way classification needs to further reduce the intra-class variation and enlarge the inter-class difference, and the proposed SCI+ and CIF+ modules can address these issues to get better performance. Note that, because the Point-BERT (Yu et al., 2021) is pre-trained on the ShapeNet55 (Chang et al., 2015), it has better performance when fine-tuned with more labeled examples (5-shot), but it may be overfitting when fine-tuning with one labeled example (1-shot).

Table 8 Complexity analysis of the proposed modules with DGCNN (Wang et al., 2019b) as the backbone

SPF	SCI+	CIF+	5way-1shot-15query		
			PN	GFLOPs	TPS
			0.618 M	96.88	12.03
✓			0.618 M	97.00	8.02
	✓		0.621 M	97.25	8.81
		✓	0.624 M	97.09	9.10
✓	✓	✓	0.627 M	97.37	7.49

We take Prototypical Net (Snell et al., 2017) as baseline (the first row), ✓ means equipped with that block

5.2.4 Complexity Comparisons

To study the complexity and cost of the competing FSL methods, we report the parameter number (PN), floating-point operations (GFLOPs) and the average inference tasks per second on one NVIDIA 2080Ti GPU for a 5-way 1-shot task with 15 query examples per class. As shown in the right part of Table 4, our proposed network introduces a small computational overhead (about 97.37 GFLOPs) and little parameters increase (about 0.63M) for the 5-way 1-shot task, corresponding to 0.5 GFLOPs and 0.02M relative increase over the

Table 9 Comparisons of the classification results (accuracy %) and improvement, after incorporating the SPF, SCI+ and CIF+ modules into different FSL algorithms with DGCNN (Wang et al., 2019b) as the backbone

Method	ModelNet40-FS		ShapeNet70-FS		ScanObjectNN-FS	
	5w-1s	5w-5s	5w-1s	5w-5s	5w-1s	5w-5s
Prototypical Net (Snell et al., 2017)	69.95	85.51	69.03	82.08	46.28	63.94
Prototypical Net (Snell et al., 2017)+ours	**81.19**	**89.30**	**78.37**	**85.15**	**56.24**	**70.93**
Improvement	+11.24	+3.79	+9.34	+3.07	+9.96	+6.99
Relation Net (Sung et al., 2018)	68.57	82.01	67.87	77.99	47.64	58.80
Relation Net (Sung et al., 2018)+ours	**75.97**	**85.45**	**73.91**	**83.20**	**51.40**	**65.32**
Improvement	+7.40	+3.44	+6.04	+5.21	+3.76	+6.52
FSLGNN (Garcia and Bruna, 2017)	61.96	80.22	66.25	76.20	28.82	35.32
FSLGNN (Garcia and Bruna, 2017) + ours	**65.67**	**83.57**	**71.20**	**80.84**	**30.32**	**39.56**
Improvement	+3.71	+3.35	+4.95	+4.64	+1.50	+4.24
Meta-learner (Ravi and Larochelle, 2017)	59.08	76.99	64.53	74.61	40.52	51.08
Meta-learner (Ravi and Larochelle, 2017) + ours	**65.43**	**79.45**	**70.14**	**80.19**	**43.89**	**52.23**
Improvement	+6.35	+2.46	+5.61	+5.58	+3.37	+1.15
MAML (Finn et al., 2017)	62.57	77.41	64.39	74.11	41.95	50.39
MAML (Finn et al., 2017) + ours	**66.48**	**79.38**	**71.24**	**77.03**	**44.34**	**56.02**
Improvement	+3.91	+1.97	+6.85	+2.92	+2.39	+5.63
MetaOptNet (Lee et al., 2019)	67.05	85.05	68.27	81.06	50.93	68.07
MetaOptNet (Lee et al., 2019) + ours	**76.84**	**88.13**	**75.44**	**84.07**	**52.62**	**69.05**
Improvement	+9.79	+3.08	+7.17	+3.01	+1.69	+0.98
S2M2 (Mangla et al., 2019)	69.73	83.25	68.53	79.71	47.38	62.06
S2M2 (Mangla et al., 2019) + ours	**75.35**	**86.23**	**73.86**	**82.37**	**51.26**	**68.26**
Improvement	+5.62	+2.98	+5.33	+2.66	+3.88	+6.20
Meta-Baseline (Chen et al., 2021b)	71.33	85.27	70.16	81.08	49.97	64.83
Meta-Baseline (Chen et al., 2021b) + ours	**81.37**	**88.58**	**78.88**	**84.89**	**6.98**	**71.12**
Improvement	+10.04	+3.31	+8.72	+3.81	+7.01	+6.29
SimpleTrans (Luo et al., 2022)	71.44	86.78	69.19	83.37	48.25	68.23
SimpleTrans (Luo et al., 2022)+ours	**80.36**	**87.90**	**77.92**	**84.20**	**55.83**	**70.30**
Improvement	+8.92	+1.12	+8.73	+0.83	+7.58	+2.07

Bold values denote the improved results

Prototypical Net (Snell et al., 2017) baseline, which are less than other baselines. Note that LSSB (SimpleShot+SB) (Stojanov et al., 2021) takes point cloud and image as input, and it does not provide images on ScanObjectNN-FS, hence we cannot report its complexity and cost on this dataset. During the testing stage, Point-BERT (Yu et al., 2021) needs to fine-tune the model, and the speed of inference depends on the number of retraining epochs, hence we also cannot report the TPS of Point-BERT precisely. However, the fine-tuning phase in Point-BERT is time consuming, so its TPS is much lower than other baselines. The attention used in the method (Feng et al., 2022) needs to generate the Q-K-V vector for each point's feature introducing larger number of parameters and higher computational cost (about 201.89 GFLOPs).

5.3 Ablative Analysis

In this section, ablative experiments are first conducted to study the contribution and complexity of each proposed module. Then we analyze the effects of applying the proposed modules to different FSL baselines.

5.3.1 Ablation Studies

We further perform ablation studies to analyze the contribution of each proposed module. We take Prototypical Net (Snell et al., 2017) with DGCNN (Wang et al., 2019b) embedding network as the baseline, and incorporate these modules into the baseline to form different combinations. Table 7 lists the comparison results, in which we can observe

that the three proposed components could provide positive impacts and improve the baseline's performance on all datasets. From the perspective of individual modules, the CIF+ module contributes more than SPF and SCI+ module, suggesting that exploiting cross-instance relationships can generate more information for classification than extracting a single instance feature independently. We also can conclude that the SPF module is more beneficial in the real-world scenario, which obtains about 4% improvement on ScanObjectNN-FS, and 1.8 and 1.2% improvement on ModelNet40-FS and ShapeNet70-FS under the 5-shot setting. By combining these modules, we obtain performance gains on all datasets, by 1.5–10%. Eventually, the integration of the three proposed modules yields the best performance, increasing the accuracy by about 10% for the 1-shot setting and by 4% for the 5-shot setting, respectively. Moreover, the comparison results of the SCI+ and CIF+ modules and the SCI and CIF modules can be referred to the Supplementary Material.

5.3.2 Component Complexity Analysis

We further analyze the complexity of each proposed module when inserted into the baseline network under the 5-way 1-shot 15-query setting. As shown in Table 8, the SPF introduces negligible parameters, and the SCI+ and the CIF+ only increase little parameters (about 0.003 M and 0.006 M) and floating-point operations (about 0.37 GFLOPs and 0.49 GFLOPs). However, they also lead to slight inference speed reduction, about four episodes on average.

5.3.3 Applying the proposed modules to Different FSL Baselines

Because the proposed models only impact the point cloud features, they can be easily inserted into the compared FSL baselines mentioned in the Tabel 4. Therefore, we embed the proposed modules, SPF, SCI+ and CIF+, into metric-based and optimization-based FSL baselines to validate their generalization ability to different networks. Fairly, we take the DGCNN (Wang et al., 2019b) as the backbone for feature embedding, and train the networks with the same strategy. Table 9 reports the improvement of inserting the proposed modules into different baselines. One can observe that there is an approximately 1 ∼ 11% consistent increase after integrating with the proposed module, which demonstrates that proposed modules can be inserted into both metric-based and optimization-based FSL baselines with significant performance improvement.
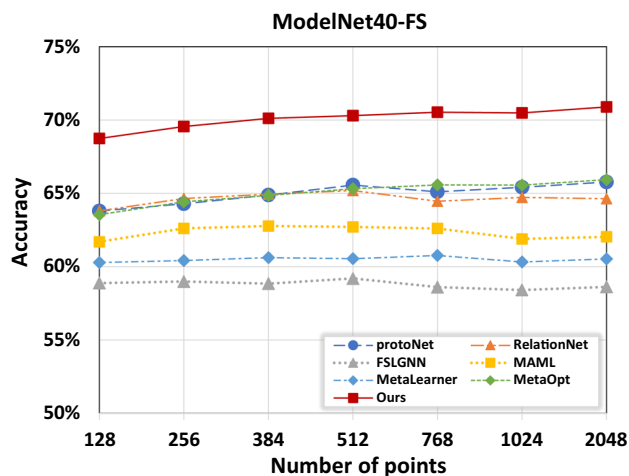

Fig. 7 Comparisons of the classification results with randomly picked points in a point cloud instance on ModelNet40-FS under the 5-way 1-shot setting
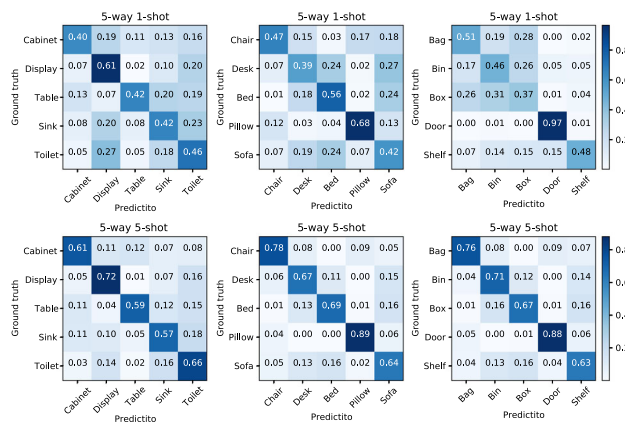

Fig. 8 Confusion matrices of classification results only using the coarse-grained global representation on the three splits of ScanObjectNN-FS

5.4 Other Analysis

5.4.1 Impact of Point Cloud Density

To study the robustness of our proposed network and the FSL baselines with different point cloud densities, we conduct the experiments starting with selecting 128 points and going up to 2048 points for each instance on ModelNet40-FS under the 5-way 1-shot setting. Due to the GPU memory constraint, we take PointNet (Qi et al., 2017a) as the backbone. Results in Fig. 7 show that the proposed network outperforms other baselines by a large margin under different point cloud density settings. Note that all the baselines have strong robustness for few-shot point cloud classification, even with fewer points. Moreover, for most FSL baselines, a larger number of points only slightly improve the classification results.

Table 10 Impact of the SPF module under different noise rates on the ScanObjectNN-FS with DGCNN (Wang et al., 2019b) as the backbone

Setting		Noise rate				
		0.0–0.2	0.2–0.4	0.4–0.6	0.6–0.8	0.8–1.0
5w1s	W/o SPF	56.44	55.41	51.44	48.29	**43.61**
	W/SPF	**58.61**	**56.83**	**52.92**	**48.69**	43.32
	Improvement	+2.17	+1.42	+1.48	+0.40	−0.29
5w5s	W/o SPF	73.76	72.89	67.40	63.11	58.05
	W/SPF	**76.10**	**74.48**	**69.23**	**64.50**	**58.66**
	Improvement	+2.34	+1.98	+1.83	+1.39	+0.61

Bold values denote the improved results

Table 11 The comparison results (accuracy %) of taking different metric functions as the meta classifier

Metric	ModelNet40-FS		ShapeNet70-FS		ScanObjectNN-FS	
	5w-1s	5w-5s	5w-1s	5w-5s	5w-1s	5w-5s
Cosine similarity	78.96	87.03	76.31	83.26	51.68	66.78
Euclidean distance	**81.19**	**89.30**	**78.37**	**85.15**	**56.24**	**70.93**

Bold values denote the best results

5.4.2 Impact of Noise Rate on the SPF Module

In the SPF module, salient parts selection relies highly on the coarse-grained global representations. To quantify the quality of the coarse-grained global representation, we first depict the confusion matrices of classification results only using the coarse-grained global representations on the three splits of ScanObjectNN-FS in Fig. 8. One can observe that the coarse-grained global representations also can provide distinguishing information for different classes, especially for the 5-shot setting.

Then, we further study the performance of the SPF module under different noise rate settings to explore the upper limit capability against cluttered background interference. Note that we define the noise rate as the ratio of noisy background points to total points in one instance. Results listed in Table 10 indicate that the SPF module can improve the performance by at least 1.4% for 1-shot and nearly 2% for 5-shot when the noise rate is less than 0.6, showing good resistance capability of the SPF module to marginal noise and background interference. However, when the noise rate is higher than 0.6, the SPF module only obtains marginal improvement or even impairs the accuracy for 1-shot. We attribute the results to that a large number of noisy points adversely impact the discrimination of global representations, and noisy backgrounds may also be considered as the potential salient parts, which limits the efficacy of the SPF module.

5.4.3 Impact of Global Feature on the SPF Module

This section investigates the impact of information loss when discarding the non-salient feature points, by removing the

global feature in Eq. (8) and simply generating the output feature using only salient parts. Table 12 reports the results on ModelNet40-FS and ScanObjectNN-FS datasets. We can observe that, without the global feature, the performance drops slightly on ModelNet40-FS dataset while more on ScanObjectNN-FS datasets.

5.4.4 Impact of Different Data Augmentations

This section studies the impact of different data augmentations when they are used to train the proposed method. Here we compare to other two data augmentations techniques, Point-Mixup (Chen et al., 2020) and Point-Augment (Li et al., 2020b). Note that we only augment the data during the training and validating stages using the methods proposed in their paper. Table 13 lists the few-shot classification results on ScanObjectNN-FS dataset. One can obverse that the proposed method can work well with different augmentation methods, which have a consistent performance improvement of about 2% compared with that without augmentation. Specially, the proposed model achieves 56.80% accuracy in the 1-shot setting if equipped with Point-Augment (Li et al., 2020b), exceeding the performance of jittering and rotating methods.

5.4.5 Impact of Different Classifiers

Here we conduct ablative experiments to study the impact of different classification metric functions for few-shot point cloud classification. Table 11 shows the comparison results of using squared Euclidean distance and cosine similarity as classifier $\mathcal{C}_\theta$. We improve the cosine similarity Classifier for a fair comparison by adding a learnable scaling parameter τ.

Table 12 Ablation results of removing the global features, where last row denotes the performance drop on different datasets under different settings after discarding global features

Method	ModelNet40-FS		ScanObjectNN-FS	
	5w-1s	5w-5s	5w-1s	5w-5s
W/global feature (ours)	**56.24**	**70.93**	**81.19**	**89.30**
Wo/global feature	55.91	70.91	80.42	88.98
Performance drop	− 0.33	− 0.02	− 0.77	− 0.32

Bold values denote the best results

Table 13 Comparison of different data augmentation methods when applied on the proposed few-shot classification framework on ScanObjectNN-FS

Method	ScanObjectNN-FS	
	5w-1s	5w-5s
No Data-Augment	54.99	68.08
Data-Augment (Ours)	56.24	**70.93**
Point-Mixup (Chen et al., 2020)	55.82	70.17
Point-Augment (Li et al., 2020b)	**56.80**	70.32

Bold values denote the best results

We can observe that the squared Euclidean distance outperforms the cosine similarity by about 2% on all the datasets, so we adopt squared Euclidean distance as the few-shot classi-fication metric for our network. A possible reason is that the squared Euclidean distance is a Bregman divergence, which is suitable for taking the average under point cloud scenario as analyzed by Snell et al. (2017).

5.5 Visualization Analysis

5.5.1 The t-SNE Visualization

We use t-SNE (Van der Maaten and Hinton, 2008) to visualize feature distribution under the 5way-5shot-15query setting on ShapeNet70-FS with DGCNN (Wang et al., 2019b) as the backbone network. The results are shown in Fig. 3, where (a) corresponds to the features of the Prototypical Net (Snell et al., 2017) baseline, achieving an accuracy of 76.38%, and (b)–(d) are the results of Prototypical Net (Snell et al., 2017)
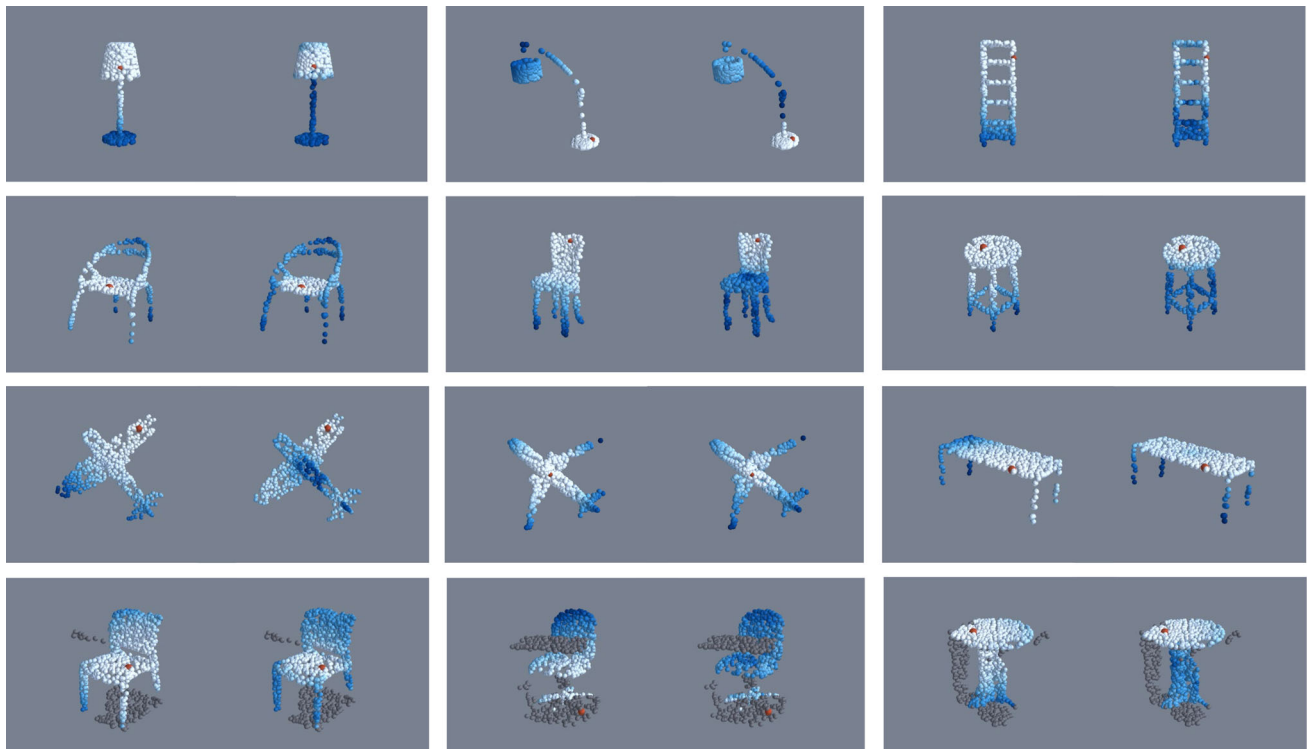

Fig. 9 The learned feature space is visualized as a distance from the *RED* point to the rest of points (White: near, Blue: far). For clear visualization, we color the the attached noisy background as Gray points. For each sub-figure, *Left*: Euclidean distance in the input $\mathbb{R}^3$ space, *Right*: Distance in the learned feature space of our method with DGCNN as the backbone. The semantically similar structures such as lampshade of a lamp, shelves of a bookshelf, back of a chair are brought close together in the feature space (Color figure online)
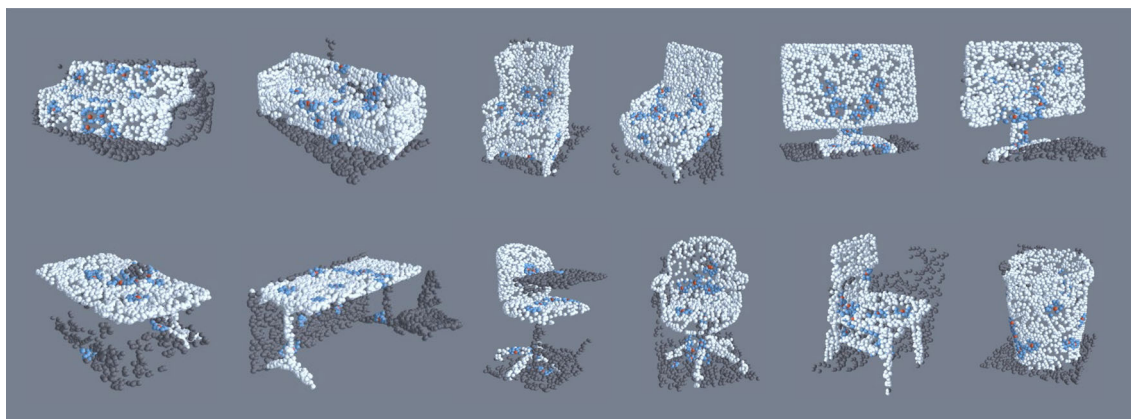
Fig. 10 Visualization of Salient Parts in the SPF module. *Red* (better zoom-in to view): the top k_s selected salient points, *Blue*: the k-NN points, *Gray*: the attached background points (Color figure online)
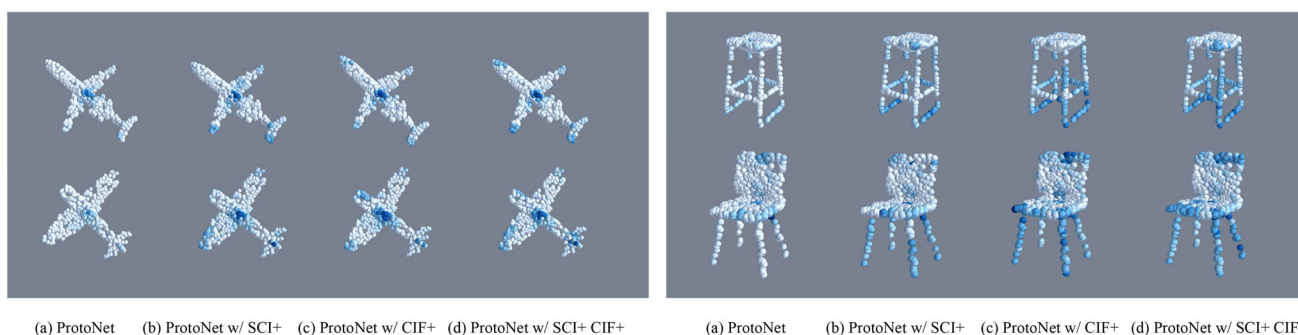

(a) ProtoNet   (b) ProtoNet w/ SCI+   (c) ProtoNet w/ CIF+   (d) ProtoNet w/ SCI+ CIF+   (a) ProtoNet   (b) ProtoNet w/ SCI+   (c) ProtoNet w/ CIF+   (d) ProtoNet w/ SCI+ CIF+

Fig. 11 The feature response heatmap of point cloud instances before and after using SCI+ and CIF+ modules. Darker color represents more intensive feature response in this region (Color figure online)

incorporating the SPF, SCI+, and CIF, achieving 77.53%, 78.38%, and 79.30% respectively, and (e) incorporates both SPF, SCI+ and CIF+ modules, obtaining better performance of 80.03%. Note that the learning support and query features of Prototypical Net (Snell et al., 2017) are dispersed with huge distribution shifts. After equipping the proposed modules, the features tend to move to the center to better differentiate from other classes, and the distribution shift between support and query sets is mitigated. More t-SNE visualization results can be referred to in Supplementary Material.

5.5.2 Visualization of Feature Space

To qualitatively analyze the feature space trained on a few labeled examples, in Fig. 9, we visualize the learned feature space output from the backbone network as a distance from the *RED* point to the rest of the points (White: near, Blue: far), as done in work (Sharma and Kaul, 2020). The gray points in the last row represent the attached noisy background. For each sub-figure, *Left*: Euclidean distance in the input $\mathbb{R}^3$ space, *Right*: distance in the learned feature space of our method with DGCNN as the backbone. One can clearly

observe that semantically similar structures such as the lampshade of a lamp, shelves of a bookshelf, and back of a chair, indicating that the proposed modules can help the backbone to learn a latent feature space in which more discriminative semantic and part information can be captured, although with few labeled training examples.

5.5.3 Visualization of Salient Parts in the SPF Module

We visualize the salient parts selected by SPF module in Fig. 10. *Red* points (better zoom-in to view on the computer) represent the salient points and *Blue* points are their k-NNs denoted as the local salient parts. *Gray* points are the noisy background. We show the results of ScanObjectNN-FS instances scanned from the real world. One can observe that the proposed model intends to select the points that are more likely to fall on the foreground object and far away from the edges or corners that usually contain background noisy points, indicating that the SPF module could fuse local salient parts more relevant to the shape description to enhance the discrimination of shape embedding, while reducing the influences from the noisy background.

5.5.4 Visualization of Feature Heatmap of SCI+ and CIF+ Module

We further visualize the feature response heatmap of the SCI+ and the CIF+ module for qualitative evaluation, by selecting the activated points after the MaxPooling layer and mapping their features into color intensity. The comparison results shown in Fig. 11 depict the change before and after incorporating the SCI+ and CIF+ module. Note that darker color represents a more intensive feature response in this region. We can observe that the SCI+ module pays closer attention to the critical parts of two fine-grained objects, such as the wing and the tail of these two kinds of airplanes, and the leg of different chairs, see Fig. 11b. While the CIF+ module focuses on more diverse regions, like the head of airplanes and the backrest of chairs, see Fig. 11c. We can conclude that incorporating the two proposed modules can help to generate more informative features by fusing refined structures and regions, see Fig. 11d.

6 Conclusion

In this work, we take a closer look at the 3D FSL problem with extensive experiments and analyses, and design a stronger algorithm for the few-shot point cloud classification task. Concretely, we first empirically study the performance of recent 2D FSL algorithms when migrating to the 3D domain with different kinds of point cloud backbone networks, and thus construct three comprehensive benchmarks and suggest a strong baseline for few-shot 3D point cloud classification. Furthermore, to address the subtle inter-class differences and high intra-class variance issues, we come up with a new network, *P*oint-cloud *C*orrelation *I*nter*a*ction (PCIA), with three plug-and-play modules, namely the Salient-Part Fusion (SPF) Module , the Self-Channel Interaction Plus (SCI+) Module, and the Cross-Instance Fusion Plus (CIF+) Module. These modules can be easily inserted into different FSL algorithms with significant performance improvement for 3D FSL. We validate the proposed network on three 3D FSL benchmark datasets, including ModelNet40-FS, ShapeNet70-FS and ScanObjectNN-FS, on which our network achieves the state-of-the-art performance.

References

Antoniou, A., Edwards, H., & Storkey, A. (2018). How to train your maml. arXiv preprint arXiv:1810.09502.

Chang, A. X., Funkhouser, T., Guibas, L., Hanrahan, P., Huang, Q., Li, Z., Savarese, S., Savva, M., Song, S., Su, H., & Xiao, J. (2015). Shapenet: An information-rich 3d model repository. arXiv preprint arXiv:1512.03012.

Chapelle, O., Scholkopf, B., & Zien, A. (2009). Semi-supervised learning (chapelle, o. et al., eds.; 2006)[book reviews]. *IEEE Transactions on Neural Networks, 20*(3), 542.

Chen, W. Y., Liu, Y. C., Kira, Z., Wang, Y. C. F., & Huang, J. B. (2019). A closer look at few-shot classification. In *International conference on learning representations*.

Chen, Y., Hu, V. T., Gavves, E., Mensink, T., Mettes, P., Yang, P., & Snoek, C. G. (2020). Pointmixup: Augmentation for point clouds. In *European conference on computer vision* (pp. 330–345). Springer.

Chen, Y., Liu, Z., Xu, H., Darrell, T., & Wang, X. (2021b). Meta-baseline: Exploring simple meta-learning for few-shot learning. In *International conference on computer vision* (pp. 9062–9071).

Chen, C., Li, K., Wei, W., Zhou, J. T., & Zeng, Z. (2021). Hierarchical graph neural networks for few-shot learning. *IEEE Transactions on Circuits and Systems for Video Technology, 32*(1), 240–252.

Chen, S., Zheng, L., Zhang, Y., Sun, Z., & Xu, K. (2018). Veram: View-enhanced recurrent attention model for 3d shape classification. *IEEE Transactions on Visualization and Computer Graphics, 25*(12), 3244–3257.

Cheraghian, A., Rahman, S., Campbell, D., & Petersson, L. (2020). Transductive zero-shot learning for 3d point cloud classification. In *Proceedings of the IEEE/CVF winter conference on applications of computer vision* (pp 923–933).

Cosmo, L., Minello, G., Bronstein, M., Rodolà, E., Rossi, L., & Torsello, A. (2022). 3d shape analysis through a quantum lens: The average mixing kernel signature. *International Journal of Computer Vision, 130*(6), 1474–1493.

Crammer, K., & Singer, Y. (2001). On the algorithmic implementation of multiclass kernel-based vector machines. *Journal of Machine Learning Research, 2*, 265–292.

Dai, A., Chang, A. X., Savva, M., Halber, M., Funkhouser, T., & Nießner, M. (2017). Scannet: Richly-annotated 3d reconstructions of indoor scenes. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 5828–5839).

Devlin, J., Chang, M. W., Lee, K., & Toutanova, K. (2018). Bert: Pre-training of deep bidirectional transformers for language understanding. arXiv preprint arXiv:1810.04805.

Doersch, C., Gupta, A., & Zisserman, A. (2020). Crosstransformers: Spatially-aware few-shot transfer. *Advances in Neural Information Processing Systems, 33*, 21981–21993.

Feng, H., Liu, W., Wang, Y., & Liu, B. (2022). Enrich features for few-shot point cloud classification. In *International Conference on Acoustics, Speech, and Signal Processing*.

Feng, Y., Zhang, Z., Zhao, X., Ji, R., & Gao, Y. (2018). Gvcnn: Group-view convolutional neural networks for 3d shape recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 264–272).

Finn, C., Abbeel, P., & Levine, S. (2017). Model-agnostic meta-learning for fast adaptation of deep networks. In *International conference on machine learning* (pp. 1126–1135). PMLR.

Garcia, V., & Bruna, J. (2017). Few-shot learning with graph neural networks. arXiv preprint arXiv:1711.04043.

Guo, Y., Wang, H., Hu, Q., Liu, H., Liu, L., & Bennamoun, M. (2020). Deep learning for 3d point clouds: A survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence, 43*(12), 4338–4364.

He, K., Zhang, X., Ren, S., & Sun, J. (2016). Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 770–778).

Johnson, A. E., & Hebert, M. (1999). Using spin images for efficient object recognition in cluttered 3d scenes. *IEEE Transactions on Pattern Analysis and Machine Intelligence, 21*(5), 433–449.

Lee, K., Maji, S., Ravichandran, A., & Soatto, S. (2019). Meta-learning with differentiable convex optimization. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition* (pp. 10657–10665).

Liao, Y., Zhu, H., Zhang, Y., Ye, C., Chen, T., & Fan, J. (2021). Point cloud instance segmentation with semi-supervised bounding-box mining. *IEEE Transactions on Pattern Analysis and Machine Intelligence, 44*(12), 10159–10170.

Li, Y., Bu, R., Sun, M., Wu, W., Di, X., & Chen, B. (2018). Pointcnn: Convolution on x-transformed points. *Advances in Neural Information Processing Systems, 31*, 828–838.

Li, R., Li, X., Heng, P. A., & Fu, C. W. (2020b). Pointaugment: An auto-augmentation framework for point cloud classification. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition* (pp. 6378–6387).

Li, A., Lu, Z., Guan, J., Xiang, T., Wang, L., & Wen, J. R. (2020). Transferrable feature and projection learning with class hierarchy for zero-shot learning. *International Journal of Computer Vision, 128*(12), 2810–2827.

Liu, Y., Fan, B., Meng, G., Lu, J., Xiang, S., & Pan, C. (2019a). Densepoint: Learning densely contextual representation for efficient point cloud processing. In *Proceedings of the IEEE/CVF international conference on computer vision* (pp. 5239–5248).

Liu, Y., Fan, B., Xiang, S., & Pan, C. (2019b). Relation-shape convolutional neural network for point cloud analysis. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition* (pp. 8895–8904).

Luo, X., Xu, J., & Xu, Z. (2022). Channel importance matters in few-shot image classification. In *International conference on machine learning* (pp. 14542–14559).

Mangla, P., Singh, M., Sinha, A., Kumari, N., Balasubramanian, V. N., & Krishnamurthy, B. (2019). Charting the right manifold: Manifold mixup for few-shot learning. In: *Workshop on Applications of Computer Vision* (pp. 2218–2227).

Maturana, D., & Scherer, S. (2015). Voxnet: A 3d convolutional neural network for real-time object recognition. In *2015 IEEE/RSJ international conference on intelligent robots and systems (IROS)* (pp. 922–928). IEEE.

Qi, C. R., Su, H., Mo, K., & Guibas, L. J. (2017a). Pointnet: Deep learning on point sets for 3d classification and segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 652–660).

Qi, C. R., Su, H., Nießner, M., Dai, A., Yan, M., & Guibas, L. J. (2016). Volumetric and multi-view cnns for object classification on 3d data. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 5648–5656).

Qi, C. R., Yi, L., Su, H., & Guibas, L. J. (2017). Pointnet++: Deep hierarchical feature learning on point sets in a metric space. *Advances in Neural Information Processing Systems, 30*, 5105–5114.

Rao, Y., Lu, J., Zhou, J. (2019). Spherical fractal convolutional neural networks for point cloud recognition. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition* (pp. 452–460).

Ravi, S., & Larochelle, H. (2017). Optimization as a model for few-shot learning. In *International conference on learning representations*.

Ren, M., Triantafillou, E., Ravi, S., Snell, J., Swersky, K., Tenenbaum, J. B., Larochelle, H., & Zemel, R. S. (2018). Meta-learning for semi-supervised few-shot classification. In *International conference on learning representations*.

Riegler, G., Osman Ulusoy, A., & Geiger, A. (2017). Octnet: Learning deep 3d representations at high resolutions. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 3577–3586).

Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., Huang, Z., Karpathy, A., Khosla, A., Bernstein, M., & Berg, A. C. (2015). Imagenet large scale visual recognition challenge. *International journal of computer vision, 115*(3), 211–252.

Rusu, R. B., Blodow, N., & Beetz, M. (2009). Fast point feature histograms (FPFH) for 3d registration. In *2009 IEEE international conference on robotics and automation* (pp. 3212–3217). IEEE.

Shao, T., Yang, Y., Weng, Y., Hou, Q., & Zhou, K. (2018). H-CNN: Spatial hashing based CNN for 3d shape analysis. *IEEE Transactions on Visualization and Computer Graphics, 26*(7), 2403–2416.

Sharma, C., & Kaul, M. (2020). Self-supervised few-shot learning on point clouds. *Advances in Neural Information Processing Systems, 33*, 7212–7221.

Snell, J., Swersky, K., & Zemel, R. (2017). Prototypical networks for few-shot learning. *Advances in Neural Information Processing Systems, 30*, 4080–4090.

Stojanov, S., Thai, A., & Rehg, J. M. (2021). Using shape to categorize: Low-shot learning with an explicit shape bias. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition* (pp. 1798–1808).

Su, H., Jampani, V., Sun, D., Maji, S., Kalogerakis, E., Yang, M. H., & Kautz, J. (2018). Splatnet: Sparse lattice networks for point cloud processing. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 2530–2539).

Su, H., Maji, S., Kalogerakis, E., & Learned-Miller, E. (2015). Multiview convolutional neural networks for 3d shape recognition. In *Proceedings of the IEEE international conference on computer vision* (pp. 945–953).

Sung, F., Yang, Y., Zhang, L., Xiang, T., Torr, P. H., & Hospedales, T. M. (2018). Learning to compare: Relation network for few-shot learning. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 1199–1208).

Tanner, M. A., & Wong, W. H. (1987). The calculation of posterior distributions by data augmentation. *Journal of the American statistical Association, 82*(398), 528–540.

Uy, M. A., Pham, Q. H., Hua, B. S., Nguyen, T., & Yeung, S. K. (2019). Revisiting point cloud classification: A new benchmark dataset and classification model on real-world data. In *Proceedings of the IEEE/CVF international conference on computer vision* (pp. 1588–1597).

Van der Maaten, L., & Hinton, G. (2008). Visualizing data using t-SNE. *Journal of Machine Learning Research, 9*(86), 2579–2605.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L., & Polosukhin, I. (2017). Attention is all you need. *Advances in Neural Information Processing Systems, 30*, 6000–6010.

Vinyals, O., Blundell, C., Lillicrap, T., & Wierstra, D. (2016). Matching networks for one shot learning. *Advances in Neural Information Processing Systems, 29*, 3637–3645.

Wang, Y., Chao, W. L., Weinberger, K. Q., & van der Maaten, L. (2019a). Simpleshot: Revisiting nearest-neighbor classification for few-shot learning. arXiv preprint arXiv:1911.04623.

Wang, Y., Sun, Y., Liu, Z., Sarma, S. E., Bronstein, M. M., & Solomon, J. M. (2019). Dynamic graph CNN for learning on point clouds. *ACM Transactions on Graphics (ToG), 38*(5), 1–12.

Wu, Z., Song, S., Khosla, A., Yu, F., Zhang, L., Tang, X., & Xiao, J. (2015). 3d shapenets: A deep representation for volumetric shapes. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 1912–1920).

Xu, W., Xian, Y., Wang, J., Schiele, B., & Akata, Z. (2022). Attribute prototype network for any-shot learning. *International Journal of Computer Vision, 130*(7), 1735–1753.

Ye, H. J., Hu, H., & Zhan, D. C. (2021). Learning adaptive classifiers synthesis for generalized few-shot learning. *International Journal of Computer Vision, 129*(6), 1930–1953.

Ye, C., Zhu, H., Liao, Y., Zhang, Y., Chen, T., & Fan, J. (2022). What makes for effective few-shot point cloud classification? In *Proceedings of the IEEE/CVF winter conference on applications of computer vision* (pp 1829–1838).

Yu, T., Meng, J., & Yuan, J. (2018). Multi-view harmonized bilinear network for 3d object recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp 186–194).

Yu, X., Tang, L., Rao, Y., Huang, T., Zhou, J., & Lu, J. (2021). Point-bert: Pre-training 3d point cloud transformers with masked point modeling. arXiv preprint arXiv:2111.14819.

Zhang, Z., Hua, B. S., & Yeung, S. K. (2022). Riconv++: Effective rotation invariant convolutions for 3d point clouds deep learning. *International Journal of Computer Vision, 130*(5), 1228–1243.

Zhong, Y. (2009). Intrinsic shape signatures: A shape descriptor for 3d object recognition. *2009 IEEE 12th international conference on computer vision workshops* (pp. 689–696). ICCV Workshops, IEEE.