ViewNet: A Novel Projection-Based Backbone with View Pooling for Few-shot Point Cloud Classification

Jiajing Chen, Minmin Yang, Senem Velipasalar
Electrical Engineering and Computer Science Dept., Syracuse University, Syracuse, NY, USA
{jchen152, myang47, svelipas}@syr.edu [*]

Abstract

Although different approaches have been proposed for 3D point cloud-related tasks, few-shot learning (FSL) of 3D point clouds still remains under-explored. In FSL, unlike traditional supervised learning, the classes of training and test data do not overlap, and a model needs to recognize unseen classes from only a few samples. Existing FSL methods for 3D point clouds employ point-based models as their backbone. Yet, based on our extensive experiments and analysis, we first show that using a point-based backbone is not the most suitable FSL approach, since (i) a large number of points' features are discarded by the max pooling operation used in 3D point-based backbones, decreasing the ability of representing shape information; (ii) point-based backbones are sensitive to occlusion. To address these issues, we propose employing a projection- and 2D Convolutional Neural Network-based backbone, referred to as the ViewNet, for FSL from 3D point clouds. Our approach first projects a 3D point cloud onto six different views to alleviate the issue of missing points. Also, to generate more descriptive and distinguishing features, we propose View Pooling, which combines different projected plane combinations into five groups and performs max-pooling on each of them. The experiments performed on the ModelNet40, ScanObjectNN and ModelNet40-C datasets, with cross validation, show that our method consistently outperforms the state-of-the-art baselines. Moreover, compared to traditional image classification backbones, such as ResNet, the proposed ViewNet can extract more distinguishing features from multiple views of a point cloud. We also show that ViewNet can be used as a backbone with different FSL heads and provides improved performance compared to traditionally used backbones.

[*]The information, data, or work presented herein was funded in part by National Science Foundation under Grant 1816732 and Federal Highway Administration Exploratory Advanced Research Program under Agreement No. 693JJ31950022. The views and opinions of authors expressed herein do not necessarily state or reflect those of the United States Government or any agency thereof.

1. Introduction

3D point cloud data has a wide range of applications including robotics, self driving cars and simultaneous localization and mapping (SLAM). In recent years, different approaches have been proposed for traditional point cloud-related tasks, such as point cloud classification, segmentation and object detection. Yet, few-shot learning of 3D point clouds remains relatively under-explored. In contrast to structured 2D images, a 3D point cloud is a set of unordered points. Thus, traditional Convolution Neural Networks (CNNs) cannot be directly used with 3D point clouds. To address this, PointNet [14] was proposed, which employs a max pooling operation to obtain permutation invariant features. This has been shown to be effective in capturing 3D objects' shape, and could be used for downstream tasks, such as point cloud classification and segmentation. However, in PointNet, each point's features are learned independently, and features from neighboring points are not aggregated. Thus, later works presented different approaches, wherein a better representation can be learned by incorporating features from neighboring points [15, 22, 24, 25]. Despite having different network structures, these point-based methods all employ a max pooling module to obtain permutation invariant features for the downstream tasks.

Traditional supervised learning needs a large number of labeled samples for training, and performs testing on the same classes used in training. In contrast, with few-shot learning (FSL), a model performs prediction on classes, which have not been seen during training, with only a few labeled samples provided in a support set. Let $(x,\ y)$ denote a point cloud sample and its label. In N-way-K-shot FSL, a support set $S = \{(x_i, y_i)\}_{i=1}^{N\times K}$ contains N classes with K samples for each class. A query set $Q = \{x_j\}_{j=1}^{N\times q}$ contains the same classes, with q samples for each class. The model matches each sample in Q with a sample in S to predict the labels of query samples. Support and query sets are used both in training and testing. The model gains the ability to learn the similarities between samples from the same class, and dissimilarities between different classes.

Existing approaches for FSL from point clouds [26, 27] use DGCNN [22], a well-known point-based method, as their backbone due to its simplicity and effectiveness in representing 3D object shapes. In DGCNN, non-local features are learned for each point by aggregating features from different neighbors in each Edge Convolution Layer. At the end of the network, max pooling is performed to obtain permutation invariant features, which are then used for the FSL tasks. In this paper, we first show that point-based methods are not the most suitable backbones for FSL for the following reasons: (i) The representation ability of a point-based method is correlated with the number of points kept after max-pooling [3]. Our extensive experiments show that, in FSL, a point-based backbone utilizes only a small portion of points after max pooling. Considering that classification with FSL is already more challenging than traditional supervised classification, it is even more important to make effective use of the available data points. Discarding 3D points during max-pooling decreases the shape representation ability of a point-based approach; (ii) Real-world point cloud data is affected by occlusions and has missing points, and point-based methods are very sensitive to these issues. For instance, almost all point-based methods [14, 15, 22] perform well on the ModelNet40 [23] dataset, which was generated from CAD models, and thus is not affected by missing point issues. On the other hand, the performances of these methods drop on the ScanObjectNN dataset [21], which was collected by scanning real-world objects.

To address the aforementioned issues, instead of a point-based backbone, we propose a 2D projection-based backbone, referred to as the ViewNet, for FSL of point clouds. The proposed ViewNet is inspired by GaitSet [2], which was proposed for gait recognition from videos. ViewNet is designed by incorporating our proposed novel View Pooling, which extracts more descriptive and distinguishing features from 2D projection images of point clouds, which are then fed into a few-shot head for downstream FSL tasks. More specifically, we project a point cloud into six orthogonal planes (front, back, left, right, top and bottom) to generate six depth images by using the SimpleView [7] projection method. Some example depth images are shown in Fig. 2. In addition, we propose View Pooling, which combines different projected plane combinations into five groups and performs max-pooling on each of them to generate more descriptive features. The experiments performed on the ModelNet40 [23], ScanObjectNN [21] and ModelNet40-C [18] datasets, with cross validation, show that our proposed method consistently outperforms the state-of-the-art (SOTA) on the few-shot point cloud classification task. The main contributions of this work include the following:

- We first provide an analysis of the commonly used point-based backbones in terms of point utilization, and argue that they are not well-suited for the FSL task especially with real-word point clouds obtained via scanning.
- By visualizing projected depth images of point clouds, we have observed that some projections are robust to missing points and deformations. Motivated by this, we propose the ViewNet, a 2D projection-based backbone, for few-shot point cloud classification.
- We propose View Pooling to generate more descriptive and distinguishing features.
- Our approach achieves SOTA performance on ScanObjectNN, ModelNet40-C and ModelNet40 datasets, and outperforms four different baselines [10, 17, 19, 26] on few-shot point cloud classification task.
- Ablation studies show that the proposed ViewNet backbone can generalize and be employed together with different few-shot prediction heads, providing better performance than a point-based backbone.

2. Related Work

Point Cloud Classification: PointNet [14] employs max-pooling to obtain permutation invariant features, which can be used for downstream tasks, such as classification and segmentation. Following works [4,5,15,16,22,24,25] introduce different network structures to aggregate information from neighboring points, yet most of them still employ the same max-pooling operation to obtain permutation invariant features. These methods are referred to as the point-based methods. Other methods convert 3D point clouds into 2D images, and use image processing methods to perform prediction. SimpleView [7] projects points onto six orthogonal planes to create depth images, and then uses ResNet [8] for classification. Lawin et al. [9] project point clouds onto 120 synthetic 2D images, and feed these images into a CNN.
Few-shot Learning: Prototypical Network [17] is a milestone FSL work, which learns a metric space, wherein the prediction could be performed by calculating the Euclidean distance between the features of samples in query and support sets. Chen and Wang [6] use discrete cosine transformation to generate a frequency representation. Features of frequency and spatial domain are used together for final prediction. Sung et al. [19] propose a module to obtain the relation scores between the support and query sets. Currently, most FSL models focus on 2D images, while FSL from 3D point clouds remains under-explored. Zhao et al. [27] presented one of the first works for few-shot semantic point cloud segmentation, which uses an attention-aware, multi-prototype transductive method. A recent point cloud FSL work [26] uses DGCNN [22] as the backbone, and presents a Cross-Instance Adaption module, which achieves good FSL performance on CAD-based point cloud datasets.

3. Motivation

Current point cloud FSL models [26, 27] employ point-based DGCNN as their backbone, to extract features, since it was shown in [26] that DGCNN outperformed other backbones. Different from these methods, we propose a projection-based backbone for few-shot point cloud classification. For motivation, we first show that DGCNN only keeps a small portion of point features, which can then be used in the FSL task, while completely discarding other points. We then show the sensitivity of point-based DGCNN, as backbone, to occlusions and missing points in point clouds, which are very common for real-world point cloud data.

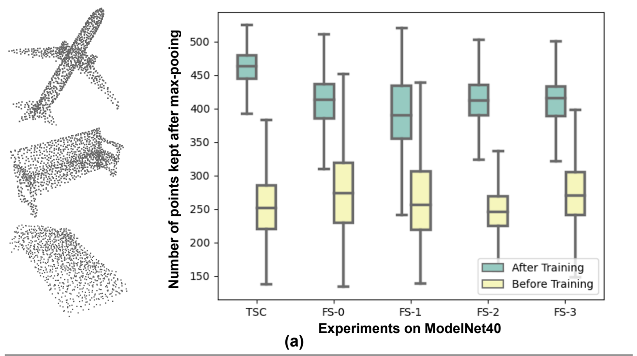
3.1. Point Utilization Analysis

Chen et al. [3] showed that point-based methods, such as PointNet [14], PointNet++ [15] and DGCNN [22], employ a max-pooling module, and use only a portion of points' features while discarding the other points. If a point has no features participating/used in the set of permutation invariant features, this point is referred to as 'discarded by max pooling'. Chen et al. [3] also showed that these discarded points are actually useful for a task at hand.

We first investigate the number of points utilized after max-pooling in DGCNN, for both traditional supervised and few-shot point cloud classification, on ModdelNet40 and ScanObjectNN datasets. For supervised point cloud classification, DGCNN is trained on the training set, and we evaluate the number of points retained by max pooling on testing set directly. For few-shot point cloud classification experiments, we employ the recent work by Ye et al. [26], which provided the SOTA performance with DGCNN as its backbone. The classes in the datasets are split into n folds, to perform n-fold cross validation. For all the experiments, the number of input points is 1024.

3.1.1 Experiments on the ModelNet40 Dataset

ModelNet40 contains objects from 40 classes. For traditional supervised classification (TSC), DGCNN is trained on the training set, which contains 9840 objects, and evaluated on the testing set containing 2468 objects. For few-shot classification, we sort 40 classes by their class ID in ascending order, and evenly split them into 4 folds, with objects from 10 classes in each fold. Some example objects from the dataset and the experiment results for point utilization are shown in Fig. 1 (a) and top half of Table 1. As can be seen, in all the experiments, the number of points kept after max-pooling in DGCNN, increases at the end of training compared to before training. This indicates that the network is learning to pick up a set of points that can better describe an object's shape for final prediction. For TSC, 464 points are utilized in DGCNN. For few-shot point cloud classification, on the other hand, a maximum of 416 points are utilized. In FSL, the model performs prediction on classes that were not seen during training, which makes few-shot classification more challenging than TSC, and also making it difficult to pick up useful points for prediction after max-pooling. Thus, with the number of points utilized for few-shot classification being less than that for TSC, it is hard to expect DGCNN to extract the best set of features as a backbone to describe 3D objects for FSL.



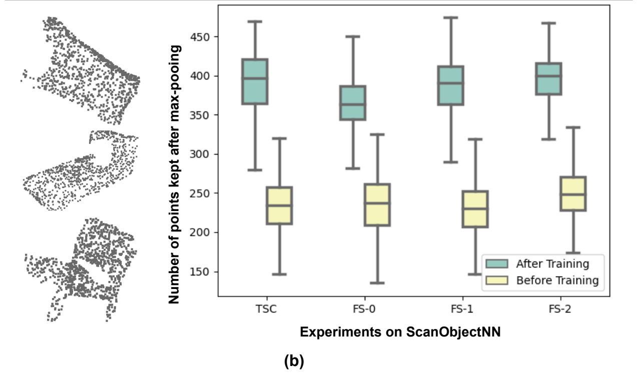


Figure 1. (a) and (b) show example 3D objects and box plots of the number of points used by DGCNN for ModelNet40 and ScanObjectNN datasets, respectively. TSC refers to traditional supervised classification, and FS-n represents few-shot point cloud classification experiment at fold n. The number of input points is 1024 for all experiments. Only about 250 points are utilized by DGCNN before the training. The number of points kept by max-pooling increases after the model is well trained.

3.1.2 Experiments on the ScanObjectNN Dataset

Different from the ModelNet40 dataset, wherein point clouds are complete and regular, points in ScanObjectNN come from scanning of real-world objects. Thus, missing points are commonly observed as seen in Fig. 1(b). Even with supervised classification, although ScanObjectNN has only 15 classes, all point-based methods [14,15,22] provide worse performance compared to the ModelNet40 dataset.

For few-shot point cloud classification, 15 classes are sorted by the class ID in ascending order, and evenly split into 3 folds for cross validation. As shown in Fig. 1(b) and lower half of Tab. 1, for TSC, a well-trained DGCNN only makes use of 397 points and provides an accuracy of 83.10% on ScanObjectNN, which is lower than the 92.51% accuracy obtained with 460 points on ModelNet40. For few-shot point cloud classification, while a well-trained DGCNN can make use of more than 400 points on ModelNet40, it uses less points on ScanObjectNN, and provides lower accuracy. From this, it can be inferred that missing points and deformed shapes can negatively affect the max-pooling, causing it to pick up inadequate points to represent a 3D object's features and shape.

Two strategies can be used to address this problem: (i) the discarded points can be recycled [3] to increase the point utilization, and make the backbone output a better set of features to describe an object's shape; (ii) the point-based backbone, such as DGCNN, can be replaced with another backbone to output more representative features. In this paper, we present an approach based on the second strategy. The reason is that if there are already missing points in the cloud to begin with, they cannot be recycled. Projections onto different view planes provide robustness against this issue, and a backbone analysis using these projections is provided in detail in Sec. 3.2.

Dataset	Experiment Name	MED of no. of kept pnts	Accuracy
ModelNet40	TSC	252→464	92.51%
ModelNet40	FS-0	274→414	89.97%
ModelNet40	FS-1	257→390	83.46%
ModelNet40	FS-2	246→413	74.08%
ModelNet40	FS-3	271→416	76.13%
ScanObjectNN	TSC	234→397	83.10%
ScanObjectNN	FS-0	237→363	50.58%
ScanObjectNN	FS-1	230→391	62.17%
ScanObjectNN	FS-2	248→400	62.59%

Table 1. $b \rightarrow a$ shows the median value of the number of utilized points before and after training, respectively. TSC is the traditional supervised point cloud classification, and FS-n is few-shot point cloud classification at fold n.

3.2. Point Projection Analysis

Occlusion and missing points are common problems with point clouds captured from LiDAR and other scanning devices. Point-based backbones [14, 15, 22] use 3D points as input directly. Thus, missing points and deformation in object shapes negatively affect their performance. However, if 3D points are projected into depth images from different angles, some depth images can be more robust against missing points, as illustrated in Fig. 2, which shows an example from the ModelNet40-C dataset [18]. This dataset contains the same 40 classes as ModelNet40 [23], but in addition to the point clouds formed by sampling a CAD model, the dataset contains point clouds obtained by introducing different types of common and realistic corruptions. The first row of Fig. 2 shows a point cloud sampled from a CAD model (referred to as Original), and clouds with simulated missing points seen from five different angles. Rows 2 through 5 show the projection images on different planes. For Angles 1 and 2, the left part of the car is missing. For Angles 3 and 4, the right part of the car is missing. In Angle 5,

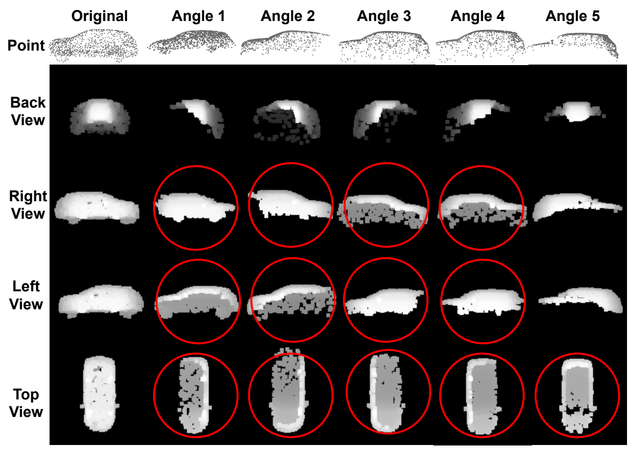


Figure 2. Projections of the original point cloud and of clouds with simulated missing points seen from five different angles. The contours of the projections of occluded clouds, shown in red circles, are similar to the ones obtained from the original point cloud.

the lower part of the car is occluded. Although the missing portion of the point cloud can be different due to scanning device's position, some projection images can provide robustness to varying occlusions. For instance, for Angles 1 and 2 in Fig. 2, the object's contour in left and right projection views and the top view are similar to those obtained from the CAD-based point cloud. Points missing on some part of the object do not affect all the projected views in the same way. For example, when the points from Angle 1 and Angle 2 are in the support set and query set, respectively, during the few-shot classification, if the backbone is given the projected depth images, and can focus on the features from a side view and top view, rather than the back view, there is a better chance of predicting the correct label for the points from Angle 2, compared to using the point clouds themselves directly.

Projection-based approach is commonly used for supervised point cloud classification. A SOTA approach is presented in [7], which projects points onto six orthogonal planes to create sparse depth images, and then uses ResNet [8] to perform the prediction task on depth images. However, for few-shot classification, wherein the model needs to perform prediction on classes that were not seen during training, we argue that a traditional image classification backbone, such as ResNet, may not be able to extract distinguishing features from depth images. Traditional CNN-based backbones are composed of convolution layers and process all depth images separately, without a module for extracting distinguishing features among all views' feature maps. To show this, we chose to use ProtoNet [17] in our analysis, since ProtoNet is a well known milestone work on FSL, with many following works developed based on it. We used ProtoNet with DGCNN and ResNet as the backbones, separately, for few-shot classification on the ModelNet40 dataset. Data splitting is done the same way as de-

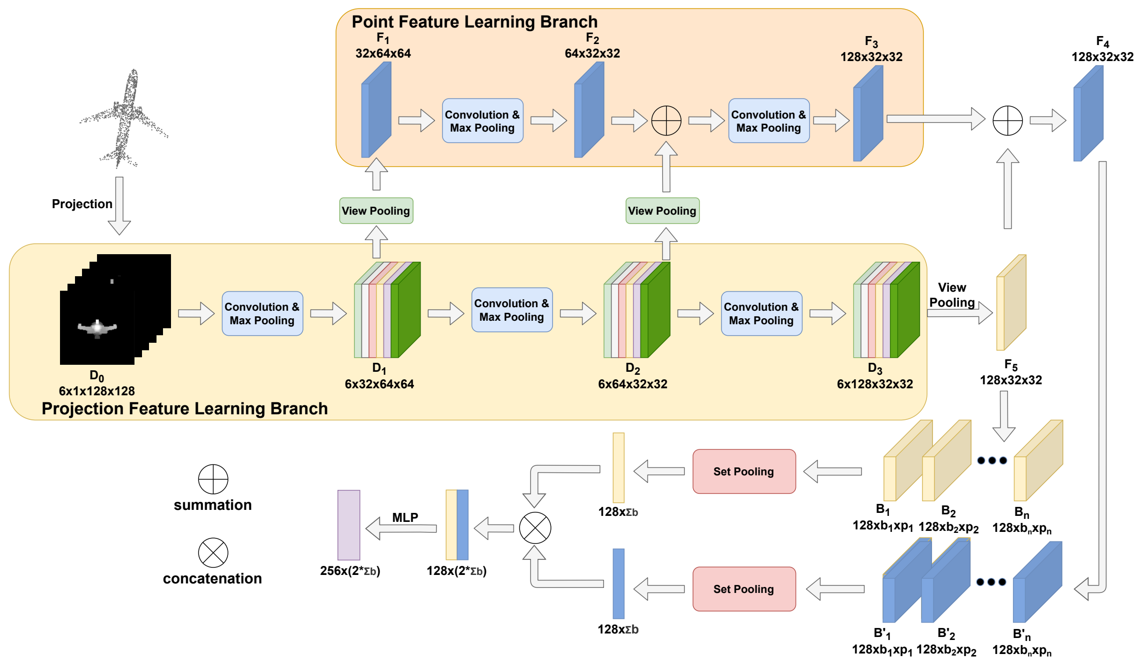


Figure 3. **Pipeline of ViewNet**. The Projection Feature Learning Branch processes feature maps of each depth image individually through convolution and max pooling. The Point Feature Learning Branch learns the features describing the point cloud's shape from the feature maps of *all six* projections.

scribed in Sec. 3.1. The results are summarized in Table 2. When ResNet is used as the backbone, points are first projected onto six orthogonal planes to obtain depth images, as described in [7]. Then, these images are fed into the ResNet, which outputs features for the downstream few-shot classification. As seen in Table 2, the average performance of DGCNN, as backbone, is better than ResNet, for both 1- and 5-shot classification. With 4-fold cross validation, DGCNN outperforms ResNet in 3 of the 4 folds. Although ResNet has a good ability to extract features for the supervised image classification task, these results show that it is not able to learn distinguishing features among projection depth images for few-shot point cloud classification.

	Model	fold 0	fold 1	fold 2	fold 3	Mean
5-way 1-shot	DGCNN+ProtoNet	**85.42%**	**79.46%**	**70.06%**	70.73%	**76.42%**
	ResNet+ProtoNet	83.29%	79.35%	64.44%	**74.42%**	75.38%%
5-way 5-shot	DGCNN+ProtoNet	**93.99%**	**88.65%**	**84.76%**	85.56%	**88.24%**
	ResNet+ProtoNet	92.61%	87.39%	80.91%	**86.96%**	86.97%%

Table 2. Comparison of Protonet's performance on ModelNet40, with DGCNN and ResNet as backbones, for 5-way 1-shot and 5-way 5-shot classification.

4. Proposed ViewNet

We propose a projection-based backbone, referred to as the ViewNet, for few-shot 3D point cloud classification. ViewNet is inspired by GaitSet [2], which focuses on gait recognition from videos. ViewNet incorporates our proposed novel View Pooling to extract more descriptive and distinguishing features. As shown in Fig. 3, ViewNet is composed of two main branches: Projection Feature Learning Branch and Point Feature Learning Branch.

Projection Feature Learning Branch takes $D_0 \in \mathbb{R}^{6\times 1\times H\times W}$ as input, where 6 is the number of projection depth images (front, back, left, right, top and bottom views), and H and W are the height and width of a depth image, respectively. In this branch, convolution and max pooling are used to process each depth image independently, and obtain intermediate feature maps $\{D_i | i \in \{1, 2, 3\}\}$ for View Pooling. View Pooling extracts features from different combinations of views, which are then fed into the Point Feature Learning Branch for further processing. The details of the View Pooling are described in Sec. 4.1.

Point Feature Learning Branch learns a set of feature maps F_i to describe point cloud features based on the output of View Pooling. This branch is also composed of convolution layers and max pooling, and interacts with the Projection Feature Learning Branch via feature summation to output F_4 (Fig. 3). To learn discriminative features in different receptive fields, the pixels in feature maps F_4 and F_5 are divided into n-many bins, where $n \in \{1, 2, 4, 8, 16\}$, resulting in bin feature matrices $\{B'_i | i = 1, 2., 5\}$ and $\{B_i | i = 1, 2.., 5\}$, respectively. In each B_i and B'_i, since bin sizes are different, bin features can cover different receptive fields. By set pooling, illustrated in Fig. 4, discriminative features from each bin can be extracted. Set pooling is composed of max pooling and average pooling, whose outputs are summed to produce the output. In the end, bin features obtained from F_4 and F_5

are concatenated, and Multiple Layer Perceptron (MLP) is applied to obtain the final output of the whole backbone.

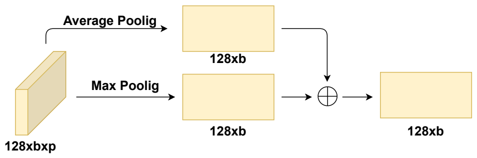


Figure 4. **Set Pooling**. b is the number of bins, p is the number of pixels in each bin. The pooling is performed along the pixel dimension.

4.1. View Pooling

As shown in Fig. 2, missing parts of a point cloud do not greatly affect some projection depth images. To leverage this, we propose a novel module, referred to as View Pooling. As shown in Fig. 5, View Pooling takes depth image feature map $D_i \in \mathbb{R}^{6\times C\times H\times W}$ as input, where 6 is the number of projection images, C is the number of feature channels for each projection feature map, $i \in \{1, 2, 3\}$ and H and W represent the feature map's size. We obtain 5 new feature maps (C_i) from 6 view features in D by using them in different combinations such that $\{C_i \in \mathbb{R}^{N\times C\times H\times W}|i = 1, .., 5\}$, where N is the number of projection feature maps combined for C_i. Three of these five feature maps are obtained by taking all pairs of opposite projection feature maps into account, i.e. $\{(left, right), (front, back), (top, bottom)\}$, since if a small part of points is missing due to occlusion, it is likely that at least one projection image in this combination of opposite projections is not affected greatly, as seen in Fig. 2. Remaining two of the five feature maps come from triplet combinations $\{(left, front, top), (right, back, bottom)\}$, since these three-view drawings are able to depict an object's 3D shape. Max pooling is performed on these combinations together with the original D_i, along the dimension of the number of projection feature maps, and distinguishing feature $\{G_i \in \mathbb{R}^{C\times H\times W}|(i = 1, ..., 6)\}$ for each projection combination is obtained. Finally, these features are concatenated, and then fed into a convolutional layer to obtain the output feature $F \in \mathbb{R}^{C\times H\times W}$.

5. Few-shot Head

In this paper, we employ the Cross Instance Adaption module (CIA) [26] as the few-shot head. CIA is composed of a Self-Channel Interaction Module and a Cross-Instance Fusion Module. In Self-Channel Interaction Module, each support and query feature $f \in \mathbb{R}^{1\times d}$ is updated by a self-attention mechanism. Firstly, a query-vector $q \in \mathbb{R}^{1\times d}$ and a key-vector $k \in \mathbb{R}^{1\times d}$ are obtained by two separate embedding linear functions operating on f. Then, a channel-wise relation score map is calculated as follows:

$$R = q^T k, R \in \mathbb{R}^{d\times d}. \tag{1}$$

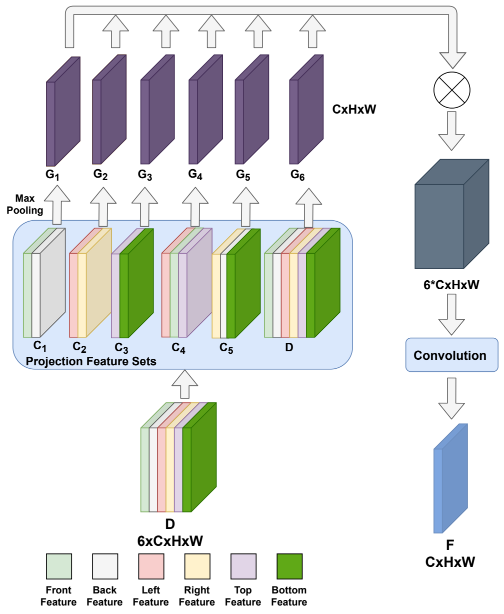


Figure 5. Pipeline of View Pooling. The input D is used together with the different projection feature combinations C_i obtained from D. Max pooling is applied to extract distinguishing features G_i for each combination.

Then, the updated feature is obtained as $f' = f + fR'$, $f' \in \mathbb{R}^{1\times d}$, where $R'_{ij} = \frac{exp(R_{ij})}{\sum_{k=1}^{d} exp(R_{kj})}$, $R' \in \mathbb{R}^{d\times d}$.

In Cross-Instance Fusion Module, each query feature f_q^i aggregates k_1 nearest neighbor support feature $\{f_s^j|j = 0..k_1\}$. Each support feature f_s^i also aggregates k_2 nearest neighbor query feature $\{f_q^j|j = 0..k_2\}$. Details can be found in the original CIA paper [26].

Different from other backbones outputting a single vector $f \in \mathbb{R}^d$, our proposed ViewNet divides all pixels in the feature map into a set of bins, and concatenates features from all bins to give the output $O \in \mathbb{R}^{B\times D}$, where B is number of bins, and D is the feature dimension. For each query and support feature $O_i \in \mathbb{R}^{B\times D}$, bin-wise CIA is applied. Then, j^{th} bin's feature of i^{th} sample, $O_i'^j$, is obtained as $CIA(O_1^j, O_2^j, ..., O_N^j)$, where N is the total number of query and support samples. Triplet loss is applied to the features of each bin. The j^{th} bin's triplet loss is

$$L_{tp}^j = Max(D(A^j, P^j) - D(A^j, N^j) + M, 0), \tag{2}$$

where $D(A^j, P^j)$ and $D(A^j, N^j)$ are the distances of the anchor to a positive and negative sample in the j^{th} bin, respectively. M is a margin value, which is set to 0.2. The final loss $L = \frac{1}{B}\sum_{k=1}^{B} L_{tp}^j$, where B is the number of bins.

6. Experiments

It was shown in [26] that DGCNN performs better as a backbone compared to other methods [11–15]. Thus, the baselines we use for comparison employ DGCNN as the backbone, together with different modules as few-shot heads. All methods are run with the same training and evaluation settings, including the learning rate, optimizer, training epochs, and 10 queries. Experiments are performed with n-fold cross validation on ScanObjectNN [20], ModelNet40-C [18] and ModelNet40 [23] datasets. Testing episodes' mean accuracy and 95% confidence intervals are computed for models' performance evaluation. The results show that our proposed method consistently outperforms the SOTA work [26] as well as other baselines, for both 1-shot and 5-shot classification.

6.1. Few-shot Classification on ScanObjectNN

ScanObjectNN [20] dataset contains 15k objects from 15 categories. Since the point clouds are scanned from real-world objects, missing points due to occlusion frequently occur, posing challenges for point cloud analysis models. Sample point clouds from ScanObjectNN are shown in Fig 1(b). The classes are sorted in ascending order based on their ID, and then evenly divided into 3 folds for cross validation. We compare our method with several SOTA baselines, namely ProtoNet [17], MetaOptNet [10], RelationNet [19] and CIA [26], which all use DGCNN as the backbone. With our approach, we use the proposed ViewNet as the backbone together with CIA as the few-shot head. The results are summarized in Table 3. Our method outperforms all the baselines with significant margins for all folds, and for both 1-shot and 5-shot classification. In 1-shot classification, our approach outperforms the second best performer (CIA) by 5.38%. In 5-shot classification, our method outperforms the second best performer (ProtoNet) by 6.17%.

		fold 0	fold 1	fold 2	Mean
5-way 1-shot	MetaOpt	41.92±0.72	61.12±0.66	53.87±0.78	52.30±0.72
	RelationNet	50.29±0.76	54.23±0.63	51.45±0.64	51.99±0.68
	ProtoNet	50.81±0.73	60.46±0.67	58.72±0.78	56.66±0.73
	CIA	50.58±0.82	62.17±0.68	62.59±0.74	58.45±0.75
	Ours	**60.90±0.76**	**66.48±0.60**	**64.10±0.77**	**63.83±0.71**
5-way 5-shot	MetaOpt	63.86±0.56	67.73±0.45	70.19±0.49	67.26±0.50
	RelationNet	58.65±0.53	66.72±0.50	65.94±0.52	63.77±0.52
	ProtoNet	68.42±0.54	70.20±0.52	68.76±0.49	69.13±0.52
	CIA	62.94±0.51	71.31±0.45	70.21±0.48	68.15±0.48
	Ours	**73.66±0.48**	**74.77±0.45**	**77.46±0.46**	**75.3±0.46**

Table 3. Few-shot classification results on ScanObjectNN. Our proposed method outperforms all baselines with a significant margin for each fold, and for both 1-shot and 5-shot classification.

6.2. Few-shot Classification on ModelNet40-C

ModelNet40-C [18] is a recently published dataset containing the point clouds of the same 40 classes as the commonly used ModelNet40 dataset [23]. Different from ModelNet40, ModelNet40-C contains point clouds formed with different types of corruption to simulate real-world scenarios. In this experiment, we use point clouds, which are corrupted to simulate clouds collected by LiDAR from five different angles. Example set of point clouds are shown in Fig. 2. 40 classes are first sorted by their class ID in ascending order, and then divided into 4 folds for cross-validation. The experimental results are summarized in Table 4. Our method outperforms all the baselines with significant margins for each fold and for both 1-shot and 5-shot classification. Our method outperforms the second best performer (CIA) by 2.75% for 1-shot classification, and by 3.32% for 5-shot classification, on average of all 4 folds.

		fold 0	fold 1	fold 2	fold 3	Mean
5-way 1-shot	Metaopt	78.28±0.79	75.34±0.84	58.07±0.86	66.29±0.91	69.50±0.85
	RelationNet	79.59±0.74	74.63±0.84	59.03±0.81	68.38±0.86	70.41±0.81
	ProtoNet	81.29±0.71	75.83±0.79	61.76±0.84	69.83±0.84%	72.18±0.80
	CIA	85.70±0.75	79.67±0.90	65.68±1.0	74.32±0.94	76.34±0.89
	Ours	**89.47±0.58**	**81.05±0.78**	**69.56±0.89**	**76.29±0.85**	**79.09±0.78**
5-way 5-shot	Metaopt	91.09±0.40	84.19±0.57	75.10±0.73	81.34±0.53	82.93±0.56
	RelationNet	87.12±0.46	83.55±0.54	70.18±0.78	79.01±0.58	79.97±0.59
	ProtoNet	90.97±0.39	86.21±0.50	76.99±0.65	83.19±0.51	84.34±0.51
	CIA	92.07±0.36	86.81±0.56	76.11±0.71	83.71±0.51	84.68±0.54
	Ours	**94.95±0.31**	**88.75±0.49**	**81.53±0.60**	**86.78±0.46**	**88±0.47**

Table 4. Few-shot classification result on ModelNet40-C dataset.

6.3. Few-shot classification on ModelNet40

ModelNet40 [23] is a commonly used point cloud dataset, which contains 12,311 CAD models of 40 man-made object categories. Point clouds are randomly sampled from CAD models' surfaces. Compared to ScanObjectNN and ModelNet40-C, points in ModelNet40 are more regular, and data does not suffer from missing point issues. Thus, point cloud analysis models usually have a better performance on this dataset. We split the data in the same way as in Sec. 3.1.1, and perform 4-fold cross validation. The experiment results are shown in Table 5. Our method outperforms all baselines in terms of mean accuracy for both 1-shot and 5-shot classification. Our approach improves the performance of the original CIA by 1.96% on 1-shot classification and by 1.58% on 5-shot classification. Compared to ScanObjectNN and ModelNet40-C, wherein points are more irregular and objects have missing points, our method provides less improvement on ModelNet40 as expected.

The significant improvements provided on ScanObjectNN and ModelNet40-C datasets show that our proposed method is also effective on point clouds scanned from the real-world, and verify the analysis provided in Sec. 3.

		fold 0	fold 1	fold 2	fold 3	Mean
5-way 1-shot	MetaOpt	82.87±0.72	75.77±0.83	65.31±0.92	66.97±0.93	72.73±0.85
	RelationNet	82.14±0.69	77.46±0.80	66.09±0.91	69.47±0.84	75.23±0.81
	ProtoNet	85.42±0.64	79.46±0.76	70.06±0.39	70.73±0.42	76.42±0.55
	CIA	89.97±0.63	**83.46±0.83**	74.08±0.95	76.13±0.86	80.91±0.82
	Ours	**92.57±0.52**	82.68±0.80	**75.28±0.90**	**80.95±0.75**	**82.87±0.74**
5-way 5-shot	MetaOpt	92.37±0.38	86.44±0.62	82.10±0.58	83.15±0.55	86.02±0.53
	RelationNet	91.53±0.38	85.11±0.61	79.36±0.63	83.01±0.52	84.75±0.53
	ProtoNet	93.99±0.29	88.65±0.54	84.76±0.51	85.56±0.48	88.24±0.45
	CIA	94.61±0.30	89.15±0.55	85.00±0.51	86.71±0.50	88.87±0.47
	Ours	**96.23±0.26**	**89.64±0.55**	**85.74±0.51**	**90.18±0.45**	**90.45±0.44**

Table 5. Few-shot classification results on the ModelNet40 dataset.

7. Ablation Studies

For the ablation studies, we use the ScanObjectNN dataset, since it is composed of point clouds scanned from real-world objects.

7.1. Analysis of the Bin-wise Loss

The backbones used in other works, such as DGCNN and ResNet, output a vector $f \in \mathbb{R}^D$ to describe an object's shape, where D is the vector's dimension. Then, f is fed into a few-shot head for prediction and loss calculation. However, our proposed ViewNet outputs multiple feature matrices (one for each bin) $O \in \mathbb{R}^{B \times D}$, where B is the number of bins. Different bins contain features from different receptive fields. Few-shot head processes each bin's feature matrix separately, and calculates loss $L_{b_i} \in \mathbb{R}^B$ based on prediction on each bin. The final loss is $L = \frac{1}{B} \sum L_{b_i}$.

In this ablation study, the bin feature matrix $O \in \mathbb{R}^{B \times D}$ is compressed into a vector $O' \in \mathbb{R}^D$ by a linear layer so that O' has the same dimension as other backbones' output f. The training and evaluation are performed on O and O' with our method, and the results are shown in Table 6. As can be seen, if O is compressed into O', the performance drops for all folds for both 1-shot and 5-shot classification. Since different bins contain features from different receptive fields, if they are compressed into one bin by force, information might be lost, causing a performance drop.

	Backbone Feature	fold 0	fold 1	fold 2	Mean
5-way 1-shot	O'	57.29%	64.47%	62.52%	61.43%
	O	**60.90%**	**66.48%**	**64.10%**	**63.83%**
5-way 5-shot	O'	73.28%	74.41%	75.42%	74.37%
	O	**73.66%**	**74.77%**	**77.46%**	**75.30%**

Table 6. Ablation study for the bin-wise loss: prediction results from O and O'.

7.2. ViewNet's Generalizability as a Backbone

It was shown in [26] that DGCNN provides the best performance as backbone compared to other networks. In this section, we show that ViewNet can be used as a backbone with different few-shot heads and provides better performance than DGCNN. For fair comparison, the ViewNet's output $O \in \mathbb{R}^{B \times D}$ is compressed into $O' \in \mathbb{R}^d$, which has the same dimension as DGCNN's output. Although this can cause information lost as shown in Sec. 7.1, it is guaranteed that the output features of ViewNet and DGCNN are fed into the same few-shot head and same loss function is used during training and evaluation. The results are shown in Table 7. With all of the four few-shot heads and for both 5-shot and 1-shot classification, ViewNet achieves higher mean accuracy than DGCNN. ViewNet outperforms DGCCN as a backbone in 23 of the 24 different fold experiments for 5-shot and 1-shot classification.

		fold 0	fold 1	fold 2	Mean
5-way 1-shot	DGCNN+MetaOpt	41.92%	61.12%	53.87%	52.3%
	ViewNet+MetaOpt	**48.74% ($\uparrow$6.82%)**	**61.62% ($\uparrow$0.5%)**	**58.95% ($\uparrow$5.08%)**	**56.44% ($\uparrow$4.14%)**
	DGCNN+RelationNet	50.29%	54.23%	51.45%	51.99%
	ViewNet+MetaOpt	**55.73% ($\uparrow$5.44%)**	**60.32% ($\uparrow$6.09%)**	**59.10% ($\uparrow$7.65%)**	**58.38% ($\uparrow$6.39%)**
	DGCNN+ProtoNet	50.81%	60.46%	58.72%	56.66%
	ViewNet+ProtoNet	**56.02% ($\uparrow$5.21%)**	**64.06% ($\uparrow$3.6%)**	**64.05% ($\uparrow$5.33%)**	**61.37% ($\uparrow$4.71%)**
	DGCNN+CIA	50.58%	62.17%	62.59%	58.45%
	ViewNet+CIA	**60.81% ($\uparrow$10.23%)**	**65.84% ($\uparrow$3.67%)**	**64.19% ($\uparrow$1.6%)**	**63.61% ($\uparrow$5.16%)**
5-way 5-shot	DGCNN+MetaOpt	63.86%	67.73%	70.19%	67.26%
	ViewNet+MetaOpt	**67.97% ($\uparrow$4.11%)**	**73.04% ($\uparrow$5.31%)**	**75.12% ($\uparrow$4.93%)**	**72.04% ($\uparrow$4.78%)**
	DGCNN+RelationNet	58.65%	**66.72%**	65.94%	63.77%
	ViewNet+RelationNet	**67.49% ($\uparrow$8.84%)**	66.51% ($\downarrow$0.21%)	**72.01% ($\uparrow$6.08%)**	**68.67%($\uparrow$4.9%)**
	DGCNN+ProtoNet	68.42%	70.2%	68.76%	69.13%
	ViewNet+ProtoNet	**75.13% ($\uparrow$6.71%)**	**74.41% ($\uparrow$4.21%)**	**77.07% ($\uparrow$8.31%)**	**75.54% ($\uparrow$6.41%)**
	DGCNN+CIA	62.94%	71.31%	70.21%	68.15%
	ViewNet+CIA	**72.69% ($\uparrow$9.75%)**	**73.56% ($\uparrow$2.25%)**	**75.33% ($\uparrow$5.12%)**	**73.86% ($\uparrow$5.71%)**

Table 7. Comparison of DGCNN and ViewNet as backbones.

7.3. Analysis of View Pooling

In our proposed View Pooling module shown in Fig. 5, max pooling is performed on D as well as C_i. To show the contribution of C_i, we perform an experiment, wherein F is only obtained from D, without any use of C_i. The results in Table 8 show that, if pairs of feature maps from opposite views as well as their triplet combinations are included in the view pooling (with C_i), more distinguishing features can be learned, and final performance is improved.

	View Pooling Type	fold 0	fold 1	fold 2	Mean
5-way 1-shot	Without C_i	**60.98%**	63.41%	63.81%	62.73%
	With C_i	60.90%	**66.48%**	**64.10%**	**63.83%**
5-way 5-shot	Without C_i	73.14%	72.76%	75.85%	73.92%
	With C_i	**73.66%**	**74.77%**	**77.46%**	**75.30%**

Table 8. Comparison of View Pooling with and without C_i.

7.4. Analysis of Computational Complexity

The computational costs of DGCNN and ViewNet, obtained by Pytorch-summary [1], are shown in Tab. 9. The total size of ViewNet is less than one third of DGCNN's.

	Forw./Backw. Pass Size	Param Size	Total Size
DGCNN	352.01 MB	4.87MB	356.90MB
ViewNet	106.50MB	6.70MB	113.22MB

Table 9. Network Complexity of DGCNN and ViewNet.

8. Conclusion

We have proposed a robust and effective backbone, ViewNet, for 3D few-shot point cloud classification. In order to address issues of missing points and occlusion, ViewNet uses six different depth images that are projections of a point cloud. Furthermore, to generate more descriptive and discriminative features from the projected six planes, we have proposed View Pooling, which uses different combinations of view features, and performs max-pooling on each of them. We have performed extensive experiments on the ScanObjuctNN, ModelNet40-C and ModelNet40 datasets for 5-way 1-shot and 5-way 5-shot 3D point cloud classification. Results have shown that ViewNet achieves SOTA performance, and can be used as a backbone together with various few-shot heads. As future work, we will apply ViewNet to other related tasks, such as zero-shot learning of point clouds, or traditional supervised point cloud classification.

References

[1] S. Chandel. Model summary in pytorch. https://github.com/sksq96/pytorch-summary. 8

[2] H. Chao, Y. He, J. Zhang, and J. Feng. Gaitset: Regarding gait as a set for cross-view gait recognition. In Proceedings of the AAAI conference on artificial intelligence, volume 33, pages 8126–8133, 2019. 2, 5

[3] J. Chen, B. Kakillioglu, H. Ren, and S. Velipasalar. Why discard if you can recycle?: A recycling max pooling module for 3d point cloud analysis. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pages 559–567, 2022. 2, 3, 4

[4] J. Chen, B. Kakillioglu, and S. Velipasalar. Hierarchical grow network for point cloud segmentation. In 2020 54th Asilomar Conference on Signals, Systems, and Computers, pages 1558–1562. IEEE, 2020. 2

[5] J. Chen, B. Kakillioglu, and S. Velipasalar. Background-aware 3-d point cloud segmentation with dynamic point feature aggregation. IEEE Transactions on Geoscience and Remote Sensing, 60:1–12, 2022. 2

[6] X. Chen and G. Wang. Few-shot learning by integrating spatial and frequency representation. In 2021 18th Conference on Robots and Vision (CRV), pages 49–56. IEEE, 2021. 2

[7] A. Goyal, H. Law, B. Liu, A. Newell, and J. Deng. Revisiting point cloud shape classification with a simple and effective baseline. In International Conference on Machine Learning, pages 3809–3820. PMLR, 2021. 2, 4, 5

[8] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In Proceedings of the IEEE conference on computer vision and pattern recognition, pages 770–778, 2016. 2, 4

[9] F. J. Lawin, M. Danelljan, P. Tosteberg, G. Bhat, F. S. Khan, and M. Felsberg. Deep projective 3d semantic segmentation. In International Conference on Computer Analysis of Images and Patterns, pages 95–107. Springer, 2017. 2

[10] K. Lee, S. Maji, A. Ravichandran, and S. Soatto. Meta-learning with differentiable convex optimization. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pages 10657–10665, 2019. 2, 7

[11] Y. Li, R. Bu, M. Sun, W. Wu, X. Di, and B. Chen. Pointcnn: Convolution on x-transformed points. Advances in neural information processing systems, 31, 2018. 6

[12] Y. Liu, B. Fan, G. Meng, J. Lu, S. Xiang, and C. Pan. Densepoint: Learning densely contextual representation for efficient point cloud processing. In Proceedings of the IEEE/CVF International Conference on Computer Vision, pages 5239–5248, 2019. 6

[13] Y. Liu, B. Fan, S. Xiang, and C. Pan. Relation-shape convolutional neural network for point cloud analysis. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pages 8895–8904, 2019. 6

[14] C. R. Qi, H. Su, K. Mo, and L. J. Guibas. Pointnet: Deep learning on point sets for 3d classification and segmentation. In Proceedings of the IEEE conference on computer vision and pattern recognition, pages 652–660, 2017. 1, 2, 3, 4, 6

[15] C. R. Qi, L. Yi, H. Su, and L. J. Guibas. Pointnet++: Deep hierarchical feature learning on point sets in a metric space. Advances in neural information processing systems, 30, 2017. 1, 2, 3, 4, 6

[16] S. Qiu, S. Anwar, and N. Barnes. Geometric back-projection network for point cloud classification. IEEE Transactions on Multimedia, 2021. 2

[17] J. Snell, K. Swersky, and R. Zemel. Prototypical networks for few-shot learning. Advances in neural information processing systems, 30, 2017. 2, 4, 7

[18] J. Sun, Q. Zhang, B. Kailkhura, Z. Yu, C. Xiao, and Z. M. Mao. Benchmarking robustness of 3d point cloud recognition against common corruptions. arXiv preprint arXiv:2201.12296, 2022. 2, 4, 7

[19] F. Sung, Y. Yang, L. Zhang, T. Xiang, P. H. Torr, and T. M. Hospedales. Learning to compare: Relation network for few-shot learning. In Proceedings of the IEEE conference on computer vision and pattern recognition, pages 1199–1208, 2018. 2, 7

[20] M. A. Uy, Q.-H. Pham, B.-S. Hua, D. T. Nguyen, and S.-K. Yeung. Revisiting point cloud classification: A new benchmark dataset and classification model on real-world data. In International Conference on Computer Vision (ICCV), 2019. 7

[21] M. A. Uy, Q.-H. Pham, B.-S. Hua, T. Nguyen, and S.-K. Yeung. Revisiting point cloud classification: A new benchmark dataset and classification model on real-world data. In Proceedings of the IEEE/CVF international conference on computer vision, pages 1588–1597, 2019. 2

[22] Y. Wang, Y. Sun, Z. Liu, S. E. Sarma, M. M. Bronstein, and J. M. Solomon. Dynamic graph cnn for learning on point clouds. Acm Transactions On Graphics (tog), 38(5):1–12, 2019. 1, 2, 3, 4

[23] Z. Wu, S. Song, A. Khosla, F. Yu, L. Zhang, X. Tang, and J. Xiao. 3d shapenets: A deep representation for volumetric shapes. In Proceedings of the IEEE conference on computer vision and pattern recognition, pages 1912–1920, 2015. 2, 4, 7

[24] T. Xiang, C. Zhang, Y. Song, J. Yu, and W. Cai. Walk in the cloud: Learning curves for point clouds shape analysis. In Proceedings of the IEEE/CVF International Conference on Computer Vision, pages 915–924, 2021. 1, 2

[25] M. Xu, J. Zhang, Z. Zhou, M. Xu, X. Qi, and Y. Qiao. Learning geometry-disentangled representation for complementary understanding of 3d object point cloud. arXiv preprint arXiv:2012.10921, 2, 2021. 1, 2

[26] C. Ye, H. Zhu, Y. Liao, Y. Zhang, T. Chen, and J. Fan. What makes for effective few-shot point cloud classification? In Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision, pages 1829–1838, 2022. 1, 2, 3, 6, 7, 8

[27] N. Zhao, T.-S. Chua, and G. H. Lee. Few-shot 3d point cloud semantic segmentation. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pages 8873–8882, 2021. 1, 2