SimpliMix: A Simplified Manifold Mixup for Few-shot Point Cloud Classification

Minmin Yang, Weiheng Chai, Jiyang Wang, Senem Velipasalar
Syracuse University
{myang47, wchai01, jwang127, svelipas}@syr.edu *

Abstract

Few-shot learning often assumes that base classes are abundant and diverse with plentiful well-labeled samples for each class. This ensures that models can generalize effectively from a small amount of data by leveraging prior knowledge learned from base classes. This assumption holds for 2D few-shot learning since the benchmark datasets are large and diverse. However, 3D point cloud few-shot benchmarks are low in magnitude and diversity. We conduct experiments and show that many existing methods overlook this issue and suffer from overfitting on base classes, which hinders generalization ability and test performance. To alleviate the overfitting issue, we propose a simplified manifold mixup, referred to as the SimpliMix, which mixes hidden representations and forces the models to learn more generalized features. We incorporate SimpliMix into existing prototype-based models, perform experiments on ModelNet40-FS, ModelNet40-C-FS and ScanObjectNN-FS datasets, and improve the models by a significant margin. We further conduct cross-domain few-shot classification experiments and show that networks with SimpliMix learn more generalized and transferable features and achieve better performance. The code is available at https://github.com/LexieYang/SimpliMix

1. Introduction

Relying on the well-labeled large datasets, deep learning-based methods have made significant strides in various tasks, *e.g.* image classification, object detection, and semantic segmentation. To alleviate the burden of labor-intensive data labeling, few-shot learning (FSL), which aims to quickly generalize to new tasks given only a few labeled data, has become increasingly popular [6, 13, 15, 22, 25, 26, 35, 37]. In the area of few-shot 2D image classification, many excellent methods have been proposed [7, 16, 22, 26, 29]. Among them, the framework of meta-learning has been widely adopted [7, 22, 25] to solve the few-shot classification problem. One category of meta-learning is metric-based methods [22, 25, 29], which aim to learn a transferable distance function over samples. In general, metric-based methods consist of a backbone, such as ResNet-12 [10] and WRN-28-10 [39], and a distance metric. The backbone extracts features, and the distance metric learns how to compare and classify examples in the feature space. ProtoNet [22] is one of the widely used methods considering its simplicity and effectiveness. ProtoNet extracts image features with a convolutional neural network (CNN), then computes the prototypes for each class by taking the mean of features of all support samples from the same class, and finally computes the Euclidean distance among query samples and prototypes in the feature space to classify the query samples.

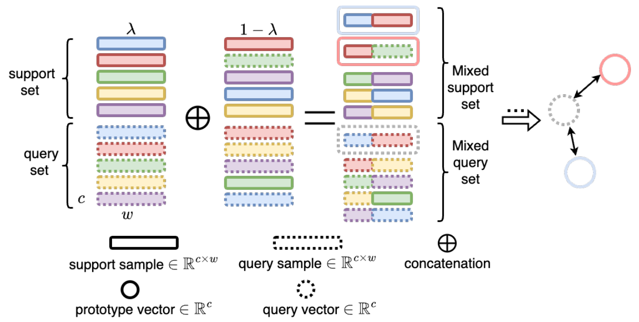

support set
query set
λ
$1-\lambda$
c
w
Mixed support set
Mixed query set
support sample $\in \mathbb{R}^{c\times w}$
query sample $\in \mathbb{R}^{c\times w}$
concatenation
prototype vector $\in \mathbb{R}^{c}$
query vector $\in \mathbb{R}^{c}$

Figure 1. The general workflow of SimpliMix. A 5-way-1-shot-1-query task is presented with classes represented by different colors. The c and w represent the feature dimension and the number of points, respectively. λ controls how many points are mixed from the other sample. After mixing, the mixed samples continue to be processed by the remaining layers in the network. Finally, the mixed query vectors are forced to be close to the classes of the samples they get mixed from. We only draw one query vector and two prototype vectors for simplicity.

Compared to few-shot image classification, however, few-shot 3D point cloud classification is still under-explored, and most of the existing works [1, 5, 35, 37, 38] adopted the idea of ProtoNet [22]. More specifically, 3D point cloud classification networks, such as DGCNN [31] and PointNet [19], can be utilized to extract features, and

*This work was supported in part by the National Science Foundation (NSF) under Grant 2221875 and New York State Department of Economic Development under Contract Number C080116.

prototype representations of each class can be computed and used for classifying query samples. In [37], it is demonstrated that ProtoNet also works in the field of few-shot point cloud learning by replacing the CNN with DGCNN. Based on such a framework, existing methods [5, 37] achieve promising results. However, these methods overlook the significant size difference between 2D few-shot datasets [2, 20, 29] and 3D point cloud datasets [24, 27, 33], which leaves room for improvement. To put it more precisely, the success of few-shot image classification can be largely attributed to the substantial size and remarkable diversity of 2D image datasets, such that networks trained on the base classes can generalize well to the never-before-seen classes. On the contrary, 3D point cloud datasets used for few-shot learning are relatively small in size and lack diversity. To verify our hypothesis, we conduct experiments and show that existing few-shot point cloud classification methods [5, 37] suffer from overfitting to base classes in the training stage (more details are provided in Sec. 3.1), which hinders their generalization ability and test accuracy. Therefore, we propose SimpliMix, which works as a regularizer to alleviate the overfitting issue.

Inspired by the manifold mixup [28], we adapt the idea of linearly interpolating samples in the feature space to enhance model generalization. The key structure of SimpliMix is shown in Fig. 1. Overall, we adopt meta-learning for few-shot classification with a feature extractor $\mathcal{F}_\theta(\cdot)$, where $\mathcal{F}_\theta(\cdot)$ denotes the feature extractor parameterized by θ consisting of L layers, and a prototype-based few-shot head. In each episode, we have a support set, which contains $N \times K$ samples, where N is the number of classes and K is the number of support samples per class, and a query set, which includes $N \times M$ samples, where M is the number of query samples per class. Therefore, each episode has $N \times (K + M)$ input samples in total. Then, for the feature extractor $\mathcal{F}_\theta$, we randomly select a layer l for mixup. Each sample in the episode, regardless of whether it is a support or a query sample, is mixed with another random sample in the same episode. Different from the manifold mixup, SimpliMix controls the level of mixup by a parameter λ sampled from a uniform distribution, i.e. $\lambda \sim U(0, 1)$. Manifold mixup [28], on the other hand, adopts a Beta distribution with a hyperparameter α adjusted for different datasets to achieve the best performance. The mixed query samples are classified based on their distance to the mixed prototype embeddings. Finally, we compute cross-entropy loss twice for each mixed query sample, since it is a combination of two samples. This is the second difference of SimpliMix from manifold mixup, which constructs soft labels as targets so that the network learns a smoother decision boundary. We use sample mixup as a way of adding noise and forcing the network to capture the underlying class-relevant features for classification. For example, if the two target labels for the mixed query sample are "airplane" and "bed", after mixup, this query sample is pushed towards both "airplane" and "bed" classes. Our insight is that two contradictory objectives make the learning process more difficult, which consequently leads to better generalization. Even if the two target labels for the mixed query sample are the same, the network is forced to learn better representations under the supervision of doubled cross-entropy loss in Eq. (6). The SimpliMix is only performed during training, so we do not increase the testing complexity. We apply SimpliMix to multiple few-shot point cloud learning methods and improve their performance by a significant margin on the ModelNet40-FS, ModelNet40-C-FS and ScanObjectNN-FS datasets. Furthermore, we conduct cross-domain few-shot learning and show that networks employing SimpliMix achieve higher accuracy than networks without SimpliMix. This shows that networks with SimpliMix have better generalization ability and transferability. The main contributions of this work include the following:

- We demonstrate and draw attention to the effect of size disparity between 2D image and 3D point cloud few-shot datasets on FSL performance, along with the overfitting issue that arises from overlooking this crucial distinction.
- We propose a simplified manifold mixup, called SimpliMix, which mixes all samples in each training episode to alleviate the overfitting issue and improve the networks' generalization ability, while forcing a mixed sample to be close to the class(es) it is mixed from.
- We conduct extensive experiments, including intra-domain and cross-domain few-shot classification, and improve the performance of many prototype-based methods by incorporating SimpliMix.

2. Related Work

2.1. Few-shot Point Cloud Classification

With the increasing availability of powerful computing resources and large-scale annotated datasets, deep learning methods have found widespread use in various real-world applications. Few-shot learning (FSL) techniques have been devised to handle never-before-seen classes without the need for retraining. Yet, it is noteworthy that the majority of existing FSL approaches primarily focus on 2D perception, while the domain of FSL for 3D point cloud classification remains conspicuously and relatively under-explored.

Ye *et al.* [37] firstly applied ProtoNet [22] to few-shot point cloud classification as a strong baseline, and proposed a cross-instance adaption (CIA) module for updating features, which includes two attention modules, namely the self-channel interaction (SCI) module and the cross-instance fusion (CIF) module. With the assistance of CIA, the features are more discriminative. Following [37], Yang *et al.* [35] proposed a network that extracts features of point cloud data using DGCNN [31] and features of the depth images using ResNet-18 [10]. Prototypes from two sets of features are obtained individually, and Euclidean distances between query features and prototype features are computed for point cloud features as well as for depth image features. Predictions are made by taking the average of the distances. Chen *et al.* [5] point out that point-based backbones are sensitive to issues of real-world captures of 3D point clouds, such as point cloud data being affected by occlusions or having missing points. In comparison, depth images, which are obtained by projecting point cloud data onto planes from different view angles, are more robust to these issues. Therefore, Chen *et al.* [5] proposed ViewNet, a 2D projection-based backbone, which extracts features from depth images. ViewNet can be employed together with different few-shot classification heads, such as RelationNet [25] and ProtoNet [22]. Chen *et al.* [5] show that ViewNet with SCI and CIF modules and prototype-based classification method achieves the best performance.

2.2. Mixup

Zhang *et al.* [40] introduced the concept of mixup as an innovative data augmentation technique, demonstrating its ability to enhance accuracy of image classification. They showed effectiveness of mixup in increasing the robustness of neural networks, particularly in scenarios involving learning from corrupted labels or facing adversarial examples. They also proved that mixup serves as a valuable data augmentation for stabilizing the training process of Generative Adversarial Networks (GANs), contributing to improved convergence and generation quality. The formulation of mixup is illustrated in Eq. (1), where x_i and x_j represent the unprocessed input vectors, y_i and y_j correspond to the one-hot label encodings, and λ is sampled from a Beta distribution.

$$\begin{aligned}\tilde{x} &= \lambda x_i + (1-\lambda)x_j \\ \tilde{y} &= \lambda y_i + (1-\lambda)y_j\end{aligned} \tag{1}$$

Drawing upon the mixup framework, diverse enhancements focusing on specific tasks have been introduced. Notably, for tasks such as image classification [8, 34], model robustness [14,18], domain generalization [4,30,36] and GAN applications [9, 32], distinct refinements have been put forth.

2.3. Manifold Mixup in Few-shot Learning

Manifold mixup [28], which is an extension of mixup [40], is a feature-space regularizer that linearly interpolates hidden representations of data and their one-hot labels. With mixed hidden representations of data and their soft labels as training signals, it improves the generalization of the neural networks by smoothing the decision boundary and flattening the class-representations. It is worth mentioning that manifold mixup randomly selects an eligible layer in the neural network to perform mixup, and reduces to input mixup [40] when mixing at the input layer. Manifold mixup has been proven to be effective in improving the generalization performance in 2D FSL [17, 21]. Mangla *et al.* [17] firstly trained a backbone model with self-supervised loss and classification loss, and then adopted manifold mixup to fine-tune the backbone model to learn a more general-purpose representation. Experiments showed that manifold mixup can boost the few-shot accuracy, but the two-step training process is tedious. Roy *et al.* [21] adopted manifold mixup to mix base samples and novel samples to address the data scarcity issue in FSL. However, the training process is even more complex. It includes six stages to train the model, pseudo-label and filter the whole base dataset, generate and select new samples by mixing novel data and base data, and finally retrain the model with the generated data. In contrast, our SimpliMix can be directly incorporated into the feature extractor and trained end-to-end without extra training/pretraining stages.

3. Proposed Method

In this section, we first provide our motivation, and then introduce the preliminaries of few-shot learning and our proposed method, SimpliMix.

3.1. Motivation

Better performance of 2D FSL methods [6,15,22,25,26] can be attributed to the assumption that number of base classes are large and classes are diverse, with sufficiently well-labeled examples, so that networks can better generalize to novel classes with a few labeled data during testing. This assumption is reasonable in few-shot image classification, since image datasets, such as miniImageNet [29], tieredImageNet [20] and CIFAR-FS [2], are larger and more diverse than 3D point cloud datasets, such as ModelNet40 [33] and ScanObjectNN [27], which are often re-splitted and used for few-shot point cloud classification [5,37]. More specifically, following the standard evaluation protocols [11,26], there are 351 base classes in tieredImageNet, 64 base classes in miniImageNet and CIFAR-FS. As in [5, 35], ModelNet40 [33] is divided into only 30 base classes while ScanObjectNN [27] is re-splitted and only contains 10 base classes. We present Fig. 2 to provide a comprehensive comparison of 2D image and 3D point cloud few-shot datasets in terms of the number of classes and the number of samples per class. We argue that few-shot point cloud classification networks trained with such low numbers of base classes are more prone to memorize and overfit on the base classes, which leads to low testing accuracy. To prove our hypothesis, we adopt DGCNN [31] as the feature extractor and ProtoNet as the few-shot head, and conduct experiments to demonstrate that low number of base classes

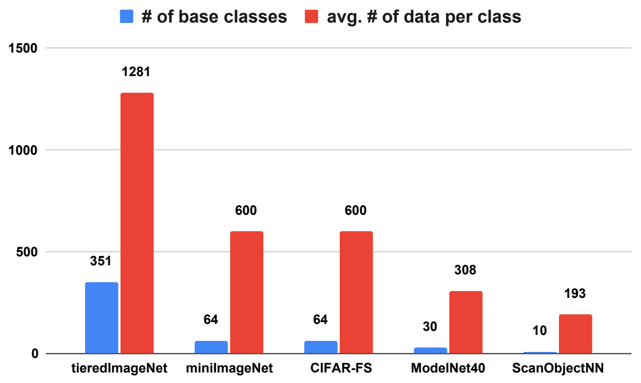


Figure 2. The comparison of the number of base classes and the number of samples per class for 2D image and 3D point cloud datasets.

	# of base classes	test acc. (%)
Exp. 1	first 10 classes	68.93
Exp. 2	first 20 classes	70.83
Exp. 3	30 classes	71.65

Table 1. The experimental results on ModelNet40 [33]. We adopt DGCNN [31] as the feature extractor and ProtoNet [22] for few-shot classification. We sorted the class IDs in ascending order, and take the first 10 classes, the first 20 classes and 30 classes for experiments. The results show that the greater the number of base classes, the higher the testing accuracy.

does affect the generalization ability of the network. As illustrated in Tab. 1, the few-shot testing accuracy increases as the number of base classes increases. Moreover, we visualize the embeddings of the base samples with IDs from 0 to 4 and novel samples with IDs from 30 to 34 as shown in Fig. 3. The base class embeddings from Exp. 1 (Fig. 3 (a)) cluster more tightly than the embeddings from Exp. 2 (Fig. 3 (b)) and Exp. 3 (Fig. 3 (c)). Nonetheless, the novel class embeddings from Exp. 1 are not well-clustered (Fig. 3 (d)) and clusters in Fig. 3 (f) (corresponding to Exp. 3) are better separated as anticipated. Taking both results in Tab. 1 and Fig. 3 into consideration, we can conclude that models trained with lower numbers of base classes are more prone to overfit on the base classes and have worse model generalization. So, how can we alleviate the overfitting issue without collecting and labeling new data or generating synthetic data using generative models? To address this issue, in this work, we propose SimpliMix, which is a simplified and modified manifold mixup and works as a regularizer for improving generalization ability of FSL networks.

3.2. Preliminaries

Before introducing our method, we first define the N-way-K-shot-M-query few-shot point cloud classification problem. Each input data is a set of 3D points, $\mathcal{P} = \{p_1, p_2, \cdots, p_w\}$ (w is the total number of points) and each point p_i is represented by (x_i, y_i, z_i). For each meta-training task, input data is constructed by a support set, $\mathcal{S} = \{(\mathcal{P}_i^S, \mathcal{Y}_i^S)\}_{i=1}^{N\times K}$ and a query set $\mathcal{Q}_i = (\{\mathcal{P}_i^Q, \mathcal{Y}_i^Q)\}_{i=1}^{N\times M}$ with all data sampled from base classes, $\mathcal{C}_b$. There are $N \times K$ samples in the support set and $N \times M$ samples in the query set, where N is the number of classes, and K and M are the number of support samples and query samples, respectively, for each class. The models are trained on a bunch of meta-training tasks with different support and query sets, and tested on support and query sets constructed from novel classes, $\mathcal{C}_n$. The overlap between $\mathcal{C}_b$ and $\mathcal{C}_n$ is empty, i.e. $\mathcal{C}_b \cap \mathcal{C}_n = \emptyset$.

ProtoNet. ProtoNet [22] is a popular meta learning-based few-shot image classification method. Suppose that the input image is denoted by $\mathbf{x}$ and its label is $\mathbf{y}$. It includes a convolutional feature extractor, $\mathcal{F}_\phi$, with learnable parameters ϕ. The feature extractor computes a D-dimensional feature vector for each sample, $h_i \in \mathbb{R}^D$. For samples in the support set, their feature vectors belonging to the same class are averaged to get the class prototype, c_k:

$$c_k = \frac{1}{K} \sum_{(\mathbf{x}_i, \mathbf{y}_i)\in\mathcal{S}_k} \mathcal{F}_\phi(\mathbf{x}_i), \tag{2}$$

where k represents class k. Then, it computes softmax over distances between each query sample $\mathbf{x}$ and the prototypes. The distribution over classes for query $\mathbf{x}$ is

$$p(\mathbf{y} = k|\mathbf{x}; \phi) = \frac{exp(-d(\mathcal{F}_\phi(\mathbf{x}, c_k))}{\sum_{k'} exp(-d(\mathcal{F}_\phi(\mathbf{x}, c'_k))}, \tag{3}$$

where d is a distance function, such as Euclidean distance.

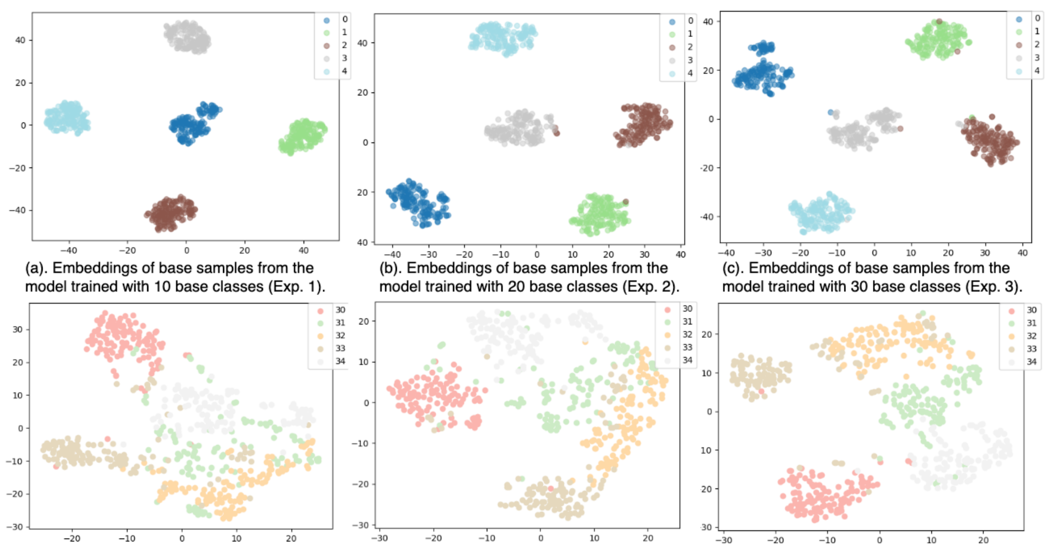
ProtoNet can be quickly modified for few-shot point cloud learning by replacing the convolutional feature extractors with point cloud feature extractors.

3.3. SimpliMix

Given a meta-training task, the batch of all inputs, $s = \{(\mathcal{P}_i, \mathcal{Y}_i)\}_{i=1}^{N\times(K+M)}$ includes $N \times K$ support samples and $N \times M$ query samples. Suppose that the feature extractor $\mathcal{F}_\theta$ (θ will be ignored in the following expressions for simplicity) consists of L layers, and we select a layer l with equal probability. This may include the input layer $\mathcal{F}^0(\mathcal{P}_i)$ as in manifold mixup [28]. We then feed the input samples to the feature extractor and process them until reaching layer l. Then, we randomly shuffle representations of all samples and get a new batch of representations, $\{\mathcal{F}^l(\mathcal{P}'_i)\}_{i=1}^{N\times(K+M)}$. Each mixed data $\hat{\mathcal{H}}$ is defined as

$$\hat{\mathcal{H}}_i = \text{Mix}_\lambda(\mathcal{F}^l(\mathcal{P}_i), \mathcal{F}^l(\mathcal{P}'_i)), \tag{4}$$

where λ is sampled from a uniform distribution, i.e. $\lambda \sim U(0, 1)$. Suppose that $\mathcal{F}^l(\mathcal{P}_i) \in \mathbb{R}^{c\times w}$, where c and w represent the feature dimensionality and the number of points, respectively. For the operation $\text{Mix}_\lambda(\cdot, \cdot)$, we randomly select a subset of points from feature representations $\mathcal{F}^l(\mathcal{P}_i)$ with the size of $(\lfloor \lambda \times w \rfloor)$, and a subset of points from



(a). Embeddings of base samples from the model trained with 10 base classes (Exp. 1). (b). Embeddings of base samples from the model trained with 20 base classes (Exp. 2). (c). Embeddings of base samples from the model trained with 30 base classes (Exp. 3).

(d). Embeddings of novel samples from the model trained with 10 base classes (Exp. 1). (e). Embeddings of novel samples from the model trained with 20 base classes (Exp. 2). (f). Embeddings of novel samples from the model trained with 30 base classes (Exp. 3).

Figure 3. The t-SNE visualization of the base and novel class embeddings from models trained with different number of base classes.

$\mathcal{F}^l(\mathcal{P}_i')$ with the size of $(w - \lfloor \lambda \times w \rfloor)$. The mixed representation $\hat{\mathcal{H}}_i$ is constructed by taking the union of the two selected subsets. We repeat the mixing process for each data to get a mixed batch. Later, the mixed batch will be fed to the remaining layers in the network and get a prediction $\hat{\mathcal{Y}}_i$ for each mixed query data. Then, we compute the loss. Each mixed query data is a combination of two samples with labels $\mathcal{Y}_i$ and $\mathcal{Y}_i'$. Therefore, we compute the cross-entropy loss as follows:

$$L = -\sum_{i=1}^{NM} \mathcal{Y}_i \cdot log(\hat{\mathcal{Y}}_i) - \sum_{i=1}^{NM} \mathcal{Y}_i' \cdot log(\hat{\mathcal{Y}}_i). \tag{5}$$

When the labels $\mathcal{Y}_i$ and $\mathcal{Y}_i'$ are different, enforcing the mixed query to be close to both class prototypes regularizes the network. However, when the labels $\mathcal{Y}_i$ and $\mathcal{Y}_i'$ are the same, the mixed query contains partial points from each of two point clouds. The network is forced to learn representations that are more robust, since only partial points are presented. Moreover, the loss reduces to Eq. (6) when $\mathcal{Y}_i$ and $\mathcal{Y}_i'$ are the same, and provides a stronger supervision, since the difference between the predicted label and the ground truth is magnified.

$$L = -2 \times \sum_{i=1}^{NM} \mathcal{Y}_i \cdot log(\hat{\mathcal{Y}}_i) \tag{6}$$

Algorithm 1 and Algorithm 2 show the pseudocode of SimpliMix based on PyTorch.

We now explain other design choices and why we choose to mix in the way used in SimpliMix. First, we employ uniform distribution while sampling λ instead of the Beta distribution ($\lambda \sim Beta(\alpha, \alpha)$ where α is a hyper-parameter) used in the manifold mixup. Considering that our goal is to

Algorithm 1 SimpliMix(x, y, λ) in PyTorch style.

```
function SimpliMix($x, y, lam$)
        ▷ Compute the mixed data. Return mixed inputs, pairs of targets, and lambda
    $batch\_size = x.shape[0]$                                ▷ Get batch size
    $indices = randperm(batch\_size)$                          ▷ Random permutation of indices
    $mixed\_x = Mix\_X(x, x[indices], lam)$                    ▷ Compute mixed inputs
    $y\_a, y\_b = y, y[indices]$                               ▷ Create pairs of targets
    return $mixed\_x, y\_a, y\_b, lam$
end function
```

Algorithm 2 Mix_X(x_1, x_2, λ) in PyTorch style.

```
function Mix_X($x1, x2, lam$)
    $w = x1.shape[2]$                          ▷ Get the number of points
    $npoints\_x1 = floor(lam \times w)$
    $npoints\_x2 = w - npoints\_x1$
    $new\_x2 = x2[:, :, : npoints\_x2]$
    $new\_x1 = x1[:, :, : npoints\_x1]$
    $mixed\_x = concat(new\_x1, new\_x2, dim = -1)$
    return $mixed\_x$                          ▷ Return mixed x
end function
```

introduce more randomness to prevent the network remembering base classes, the uniform distribution with a fixed range is sufficient and eliminates the trouble of additional hyperparameter tuning. Second, we do not create soft labels as done in manifold mixup, and instead, assume that the mixed data belongs to both classes. Our goal is to prevent the model from memorizing base samples. The combination of point features from two point clouds and the combination of two losses force the network to learn class-relevant representations for both classes. In the extreme cases of λ being close to 0 or 1, the difficulty of the task increases, since the network needs to extract more robust features that are not greatly affected when only a few points are provided from that class. Third, there are two other possible ways of performing the mixing: (1) only mixing the query samples and leaving the support samples unchanged (named as QMix), (2) mixing support samples first and then mixing the query samples based on the way supports are mixed (named as SQMix), since we classify query samples based on their distance to the prototypes in the embedding space. These two options are also capable of improving accuracy, but perform slightly worse than SimpliMix. Moreover, the second option (SQMix) is more complex in terms of implementation. Since all mixing approaches involve some degree of randomness and increase the data variability, they can prevent a model from memorizing and overfitting to base classes. Yet, SimpliMix introduces more randomness and has higher entropy between inputs and labels. It randomly mixes samples with any other samples in the same batch without considering if the other samples are from support set or query set, or considering if samples from the same class have consistent mixed labels. Therefore, it is more difficult for the model to remember the base classes and the model does not get over-confident. Supporting experimental results are provided in Sec. 4.5.1 and the pseudocode for QMix and SQMix is provided in the supplementary material.

4. Experiments

4.1. Datasets and Setup

Datasets ShapeNetCore [3] is a richly-annotated and large-scale point cloud benchmark. It covers 55 object categories with about 52,000 CAD models. ModelNet40 [33] is also a 3D dataset containing 12,311 CAD models from 40 categories (*e.g*. airplane, bed, plant, sink). All CAD models in ModelNet40 are well-segmented, and noise-free. In real-world applications, however, 3D point clouds are often corrupted. To analyze the corruption robustness of existing point cloud classification models, Sun *et al*. [24] constructed ModelNet40-C [24] dataset with 15 types of meticulously designed corruptions added to the objects in ModelNet40 to mimic the real-world point cloud distortion. Therefore, ModelNet40-C contains the same number of object categories and 3D CAD models as ModelNet40. ScanObjectNN [27] is a more challenging real-world dataset, consisting of objects from 15 categories, which suffers from partial occlusion.

Intra-domain Few-shot Learning Setup We run experiments on three datasets, namely ModelNet40-FS, ModelNet40-C-FS and ScanObjectNN-FS, which are obtained from ModelNet40 [33], ModelNet40-C [24] and ScanObjectNN [27] datasets, respectively, by re-splitting the training and testing sets.

Following the data preparation process in [5], we split ModelNet40 [33] into 4 folds, with 10 classes in each fold, based on class IDs to build ModelNet40-FS. The same split is applied to ModelNet40-C to construct ModelNet40-C-FS benchmark. We split ScanObjectNN [27] into 3 folds with 5 classes in each fold to build ScanObjectNN-FS benchmark as in [5]. We perform 3-fold cross-validation on ScanObjectNN-FS, and 4-fold cross-validation on ModelNet40-FS and ModelNet40-C-FS.

Cross-domain Few-shot Learning Setup For the cross-domain experiments, we choose base classes from a synthetic dataset and novel classes from the real-world ScanObjectNN dataset. For the target domain dataset, ScanObjectNN, we use all 15 classes as novel classes. For source domain datasets, we construct ModelNet40-XFS, ModelNet40-C-XFS and ShapeNetCore-XFS from ModelNet40, ModelNet40-C and ShapeNetCore, respectively. The classes included in ModelNet40-XFS, ModelNet40-C-XFS and ShapeNetCore-XFS do not overlap with classes in ScanObjectNN. More specifically, both ModelNet40-XFS and ModelNet40-C-XFS contain 26 base classes and ShapeNetCore-XFS includes 44 base classes. More details about the setups are provided in the supplementary material. Overall, we perform three cross-domain few-shot learning experiments, ModelNet40-XFS $\rightarrow$ ScanObjectNN, ModelNet40-C-XFS $\rightarrow$ ScanObjectNN and ShapeNetCore-XFS $\rightarrow$ ScanObjectNN.

4.2. Implementation Details

Training: We train the networks with SimpliMix on base classes via Adam [12] optimizer with a learning rate of 0.0001, and for 110 epochs for all datasets. Each epoch contains 400 randomly sampled episodes. In each episode, 15 query point clouds per class are sampled for all experiments.

Evaluation: We randomly sample 600 episodes from the set of novel classes and report the mean classification accuracy with the 95% confidence interval. The results of networks without SimpliMix are from [5].

4.3. Baselines

To show the effectiveness of our method, we compare the performance of three models, CIA [23], ViewNet [5] and ProtoNet [22] with and without SimpliMix. In CIA, point cloud data is fed to DGCNN [31] to extract features. Then, these features are updated by a self-channel interaction (SCI) module and cross-instance fusion (CIF) module in sequence, and used for calculating squared Euclidean distance for matching and classification. In contrast to point-based feature extractors, ViewNet is a 2D projection-based feature extractor. Chen *et al*. [5] show that using ViewNet

Method	5-way 1-shot					5-way 5-shot				
	fold0	fold1	fold2	fold3	Mean	fold0	fold1	fold2	fold3	Mean
ProtoNet	85.42±0.64	79.46±0.76	70.06±0.39	70.73±0.42	76.42±0.55	93.99±0.29	88.65±0.54	84.76±0.51	85.56±0.48	88.24±0.45
CIA	89.97±0.63	83.46±0.83	74.08±0.95	76.13±0.86	80.91±0.82	94.61±0.30	89.15±0.50	85.00±0.51	86.71±0.50	88.87±0.47
ViewNet	92.57±0.52	82.68±0.80	75.28±0.90	80.95±0.75	82.87±0.74	96.23±0.26	89.64±0.55	85.74±0.51	90.18±0.45	90.45±0.44
ProtoNet_SimpliMix	88.38±0.52	80.06±0.74	69.18±0.81	76.16±0.76	78.45±0.71 (↑**2.03**)	96.09±0.22	89.46±0.53	86.45±0.47	89.59±0.40	90.40±0.41 (↑**2.16**)
CIA_SimpliMix	92.88±0.45	85.36±0.75	74.96±0.85	81.65±0.78	83.71±0.71 (↑**2.8**)	96.87±0.20	90.47±0.49	87.09±0.46	90.31±0.39	91.19±0.39 (↑**2.32**)
ViewNet_SimpliMix	93.26±0.43	84.43±0.71	76.94±0.83	83.07±0.72	84.43±0.67 (↑**1.56**)	97.04±0.20	90.70±0.47	88.14±0.45	91.30±0.35	91.80±0.37 (↑**1.35**)

Table 2. Few-shot classification results on ModelNet40-FS.

Method	5-way 1-shot					5-way 5-shot				
	fold0	fold1	fold2	fold3	Mean	fold0	fold1	fold2	fold3	Mean
ProtoNet	81.29±0.71	75.83±0.79	61.76±0.84	69.83±0.84	72.18±0.80	90.97±0.39	86.21±0.50	76.99±0.65	83.19±0.51	84.34±0.51
CIA	85.70±0.75	79.67±0.90	65.68±1.0	74.32±0.94	76.34±0.89	92.07±0.36	86.81±0.56	76.11±0.71	83.71±0.51	84.68±0.54
ViewNet	89.47±0.58	81.05±0.78	69.56±0.89	76.29±0.85	79.09±0.78	94.95±0.31	88.75±0.49	81.53±0.60	86.78±0.46	88.00±0.47
ProtoNet_SimpliMix	85.51±0.58	77.41±0.77	65.70±0.87	72.77±0.78	75.34±0.75 (↑**2.88**)	93.55±0.31	87.60±0.47	80.61±0.62	86.12±0.43	86.97±0.46 (↑**2.63**)
CIA_SimpliMix	90.41±0.58	82.02±0.74	70.61±0.97	75.57±0.85	79.65±0.79 (↑**3.31**)	95.00±0.25	88.52±0.47	81.23±0.65	87.44±0.42	88.05±0.45 (↑**3.37**)
ViewNet_SimpliMix	91.32±0.53	81.27±0.73	71.35±0.86	79.10±0.78	80.76±0.73 (↑**1.7**)	96.19±0.23	88.71±0.44	83.20±0.61	89.03±0.37	89.28±0.41 (↑**1.28**)

Table 3. Few-shot classification results on ModelNet40-C-FS.

Method	5-way 1-shot				5-way 5-shot			
	fold0	fold1	fold2	Mean	fold0	fold1	fold2	Mean
ProtoNet	50.81±0.73	60.46±0.67	58.72±0.78	56.66±0.73	68.42±0.54	70.20±0.52	68.76±0.49	69.13±0.52
CIA	50.58±0.82	62.17±0.68	62.59±0.74	58.45±0.75	62.94±0.51	71.31±0.45	70.21±0.48	68.15±0.48
ViewNet	60.90±0.76	66.48±0.60	64.10±0.77	63.83±0.71	73.66±0.48	74.77±0.45	77.46±0.46	75.30±0.46
ProtoNet_SimpliMix	52.64±0.70	62.41±0.51	68.09±0.72	61.05±0.64 (↑**4.39**)	68.76±0.44	72.54±0.38	83.43±0.33	74.91±0.38 (↑**5.78**)
CIA_SimpliMix	56.54±0.73	63.58±0.67	72.11±0.77	64.08±0.72 (↑**5.63**)	69.43±0.43	73.26±0.37	85.06±0.32	75.92±0.37 (↑**7.77**)
ViewNet_SimpliMix	65.98±0.69	68.94±0.65	69.98±0.66	68.30±0.67 (↑**4.47**)	78.38±0.40	78.63±0.38	81.13±0.38	79.38±0.39 (↑**4.08**)

Table 4. Few-shot classification results on ScanObjectNN-FS.

as the backbone, then applying SCI and CIF modules to update features and finally computing prototypes for matching achieves better performance. Thus, we also adopt such a framework for ViewNet in our experiments.

4.4. Discussion of Results

4.4.1 Intra-domain Few-shot Learning Results

We conduct 5-way 1-shot 15-query and 5-way 5-shot 15-query experiments. The results on ModelNet40-FS, ModelNet40-C-FS and ScanObjectNN-FS are shown in Tables 2, 3 and 4, respectively. We incorporate our SimpliMix into ProtoNet, CIA and ViewNet, and name them as ProtoNet_SimpliMix, CIA_SimpliMix and ViewNet_SimpliMix. The results show that SimpliMix consistently improves the performance of all the networks. Among the three methods, SimpliMix boosts the accuracy of CIA the most. This can be attributed to the fact that features extracted by the DGCNN become more distinguishing by incorporating SimpliMix. Hence, when these features are further processed by SCI and CIF modules, which aim to increase the inter-class difference and decrease intra-class variance, the performance can be taken to a higher level. Among the three datasets, adopting SimpliMix yields more performance gains on ScanObjectNN-FS. It is because ScanObjectNN-FS has the lowest number of base classes and the overfitting issue is more severe. On ModelNet40-C-FS, we also perform Singular Value Decomposition (SVD) of the class representations from well-trained ProtoNet and ProtoNet_SimpliMix. For the first three classes in ModelNet40-C-FS, ProtoNet_SimpliMix greatly reduces the largest singular value. For class ID=0, we obtain 639.18 with ProtoNet versus 100.54 with ProtoNet_SimpliMix. For class ID=1, we obtain 505.45 with ProtoNet versus 61.55 with ProtoNet_SimpliMix. For class ID=2, we obtain 660.07 with ProtoNet versus 81.98 with ProtoNet_SimpliMix. We plot the second to ninth largest singular values in Fig. 4. ProtoNet_SimpliMix results in smaller singular values. This means reduced variance in some directions and flattened representation, which leads to better generalization [28].

4.4.2 Cross-domain Few-shot Learning Results

For cross-domain experiments, we also report 5-way 1-shot 15-query and 5-way 5-shot 15-query results as illustrated in the Tab. 5. We train on the source domain and test on the target domain without fine-tuning. All methods achieve higher accuracy when SimpliMix is incorporated, which proves that networks with SimpliMix lead to more generalized representations. We further visualize some class representations of ScanObjectNN in Fig. 5. The CIA and CIA_SimpliMix are trained on ModelNet40-XFS and we use the well-trained models to extract features. For both models, the embeddings of classes “display” and “door” are clustered better than the classes of “shelf”, “bed” and “sofa”. We believe it is because the classes of “shelf”, “bed” and “sofa” have higher intra-class variations. Having said that, the embeddings for classes “shelf”, “bed” and “sofa” obtained with CIA_SimpliMix are better clustered than CIA, and the clusters for classes “display” and “door” from CIA_SimpliMix are more compact.

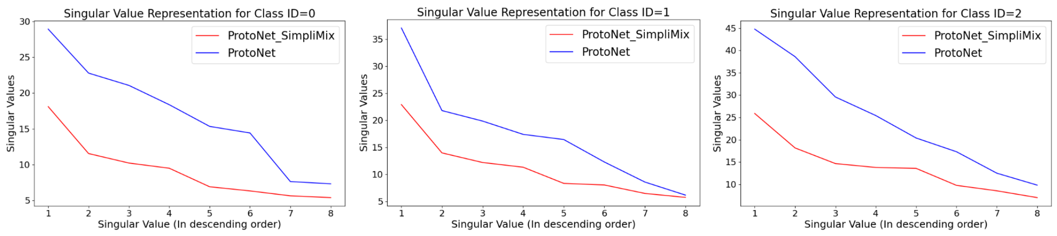


Figure 4. The Singular Value Decomposition (SVD) on the class-specific representations from ProtoNet and ProtoNet_SimpliMix. We only plot the second largest value to the ninth largest value for classes with ID from 0 to 2.

Method	ShapeNet-XFS → ScanObjectNN		ModelNet40-XFS → ScanObjectNN		ModelNet40-C-XFS → ScanObjectNN	
	1-shot	5-shot	1-shot	5-shot	1-shot	5-shot
ProtoNet	44.39±0.80	62.12±0.72	53.04±0.91	65.53±0.75	53.70±0.89	65.81±0.76
ProtoNet_SimpliMix	51.91±0.81	68.94±0.68	56.83±0.90	72.71±0.69	56.80±0.89	71.82±0.69
CIA	49.29±0.90	64.37±0.77	55.59±1.01	65.61±0.74	57.07±0.94	66.29±0.76
CIA_SimpliMix	55.53±0.91	70.88±0.67	60.95±0.98	73.84±0.67	58.56±0.96	73.42±0.69
ViewNet	48.00±0.75	65.90±0.70	58.19±0.89	73.59±0.71	57.48±0.88	72.06±0.69
ViewNet_SimpliMix	55.45±0.84	70.95±0.71	64.54±0.93	76.68±0.65	64.20±0.95	75.71±0.66

Table 5. The cross-domain few-shot classification results.

Method	5-way 1-shot					5-way 5-shot				
	fold 0	fold 1	fold 2	fold 3	Mean	fold 0	fold 1	fold 2	fold 3	Mean
ProtoNet	81.29±0.71	75.83±0.79	61.76±0.84	69.83±0.84	72.18±0.80	90.97±0.39	86.21±0.50	76.99±0.65	83.19±0.51	84.34±0.51
SimpliMix	85.51±0.58	77.41±0.77	65.70±0.87	72.77±0.78	**75.34±0.75**	93.55±0.31	87.60±0.47	80.61±0.62	86.12±0.43	**86.97±0.46**
SQMix	83.53±0.64	77.03±0.74	63.91±0.88	71.77±0.78	74.06±0.76	94.21±0.28	87.52±0.44	79.49±0.61	85.77±0.44	86.75±0.44
QMix	84.05±0.66	75.81±0.74	63.04±0.84	70.2±0.79	73.28±0.76	93.38±0.36	87.5±0.45	78.67±0.60	84.16±0.44	85.93±0.46

Table 6. Experimental results on ModelNet40-C-FS with different mixing methods.

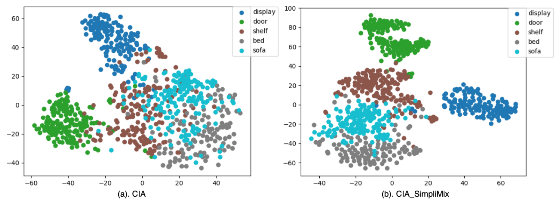


Figure 5. The t-SNE visualization of the class representations from CIA and CIA_SimpliMix.

4.5. Ablation Studies

4.5.1 Comparison of Different Mixing Methods

As introduced in Sec. 3, there are various ways of mixing samples while SimpliMix is the easiest one to implement and performs the best. We choose ProtoNet as the baseline and incorporate different mixing methods into ProtoNet to prove the effectiveness of SimpliMix. We conduct experiments on ModelNet40-C-FS dataset and report the results in Tab. 6. As expected, all mixing methods outperform ProtoNet and SimpliMix performs best in the 1-shot setting. For the 5-shot setting, SimpliMix and SQMix perform comparably. Overall, SimpliMix is favorable, since it is easier to implement.

4.5.2 Analysis of Mixing Coefficient (λ)

In SimpliMix, we sample $\lambda \sim U(0, 1)$, which is equivalent to sampling $\lambda \sim \text{Beta}(\alpha, \alpha)$ when $\alpha = 1$. We also experiment with different Beta distributions by taking various values of α on ModelNet40-C-FS. Tab. 7 shows that

	5-way 1-shot				
	fold 0	fold 1	fold 2	fold 3	Mean
α=0.1	83.64±0.65	76.11±0.75	64.87±0.85	71.43±0.80	74.01±0.76
α=0.5	84.05±0.60	76.87±0.73	65.32±0.81	71.29±0.79	74.38±0.73
α=1.0	**85.51±0.58**	**77.41±0.77**	65.70±0.87	**72.77±0.78**	**75.34±0.75**
α=2.0	84.40±0.62	76.76±0.78	65.39±0.82	72.01±0.79	74.64±0.75
α=4.0	83.8±0.64	76.73±0.73	**66.33±0.80**	72.61±0.81	74.86±0.75

Table 7. Experiments on the value of α.

sampling $\lambda \sim U(0, 1)$ achieves better average performance than sampling from smaller values of α and slightly better results than sampling from larger values of α. Therefore, we choose to sample $\lambda \sim U(0, 1)$ for simplicity.

5. Conclusion

In contrast to large-scale image datasets, few-shot 3D point cloud datasets are comparatively small in size and lack diversity. We have first shown that few-shot point cloud testing accuracy increases with increasing number of base classes, indicating that few-shot point cloud classification networks trained with low numbers of classes are more prone to overfit to the base classes. To alleviate overfitting and improve model generalization, we have proposed an adapted manifold mixup method, SimpliMix. Our proposed approach mixes representations of two samples to get a mixed sample, regularizes the network and forces the network to learn more robust representations. We have conducted intra-domain and cross-domain few-shot point cloud classification experiments and shown that incorporating SimpliMix into different models consistently improves their performance by significant margins.

References

[1] Tejas Anvekar and Dena Bazazian. Gpr-net: Geometric prototypical network for point cloud few-shot learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4178–4187, 2023. 1

[2] Luca Bertinetto, Joao F Henriques, Philip HS Torr, and Andrea Vedaldi. Meta-learning with differentiable closed-form solvers. *arXiv preprint arXiv:1805.08136*, 2018. 2, 3

[3] Angel X Chang, Thomas Funkhouser, Leonidas Guibas, Pat Hanrahan, Qixing Huang, Zimo Li, Silvio Savarese, Manolis Savva, Shuran Song, Hao Su, et al. Shapenet: An information-rich 3d model repository. *arXiv preprint arXiv:1512.03012*, 2015. 6

[4] Jia-Ren Chang, Min-Sheng Wu, Wei-Hsiang Yu, Chi-Chung Chen, Cheng-Kung Yang, Yen-Yu Lin, and Chao-Yuan Yeh. Stain mix-up: Unsupervised domain generalization for histopathology images. In *Medical Image Computing and Computer Assisted Intervention–MICCAI 2021: 24th International Conference, Strasbourg, France, September 27–October 1, 2021, Proceedings, Part III 24*, pages 117–126. Springer, 2021. 3

[5] Jiajing Chen, Minmin Yang, and Senem Velipasalar. Viewnet: A novel projection-based backbone with view pooling for few-shot point cloud classification. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 17652–17660, 2023. 1, 2, 3, 6

[6] Zitian Chen, Subhransu Maji, and Erik Learned-Miller. Shot in the dark: Few-shot learning with no base-class labels. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 2668–2677, 2021. 1, 3

[7] Chelsea Finn, Pieter Abbeel, and Sergey Levine. Model-agnostic meta-learning for fast adaptation of deep networks. In *International conference on machine learning*, pages 1126–1135. PMLR, 2017. 1

[8] Adrian Galdran, Gustavo Carneiro, and Miguel A González Ballester. Balanced-mixup for highly imbalanced medical image classification. In *Medical Image Computing and Computer Assisted Intervention–MICCAI 2021: 24th International Conference, Strasbourg, France, September 27–October 1, 2021, Proceedings, Part V 24*, pages 323–333. Springer, 2021. 3

[9] Farzin Ghorban, Nesreen Hasan, Jörg Velten, and Anton Kummert. Improving fm-gan through mixup manifold regularization. In *2021 IEEE International Symposium on Circuits and Systems (ISCAS)*, pages 1–5. IEEE, 2021. 3

[10] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016. 1, 3

[11] Yiren Jian and Lorenzo Torresani. Label hallucination for few-shot classification. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 36, pages 7005–7014, 2022. 3

[12] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014. 6

[13] Kwonjoon Lee, Subhransu Maji, Avinash Ravichandran, and Stefano Soatto. Meta-learning with differentiable convex optimization. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 10657–10665, 2019. 1

[14] Saehyung Lee, Hyungyu Lee, and Sungroh Yoon. Adversarial vertex mixup: Toward better adversarially robust generalization. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 272–281, 2020. 3

[15] Bin Liu, Yue Cao, Yutong Lin, Qi Li, Zheng Zhang, Mingsheng Long, and Han Hu. Negative margin matters: Understanding margin in few-shot classification. In *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part IV 16*, pages 438–455. Springer, 2020. 1, 3

[16] Chen Liu, Yanwei Fu, Chengming Xu, Siqian Yang, Jilin Li, Chengjie Wang, and Li Zhang. Learning a few-shot embedding model with contrastive learning. In *Proceedings of the AAAI conference on artificial intelligence*, volume 35, pages 8635–8643, 2021. 1

[17] Puneet Mangla, Nupur Kumari, Abhishek Sinha, Mayank Singh, Balaji Krishnamurthy, and Vineeth N Balasubramanian. Charting the right manifold: Manifold mixup for few-shot learning. In *Proceedings of the IEEE/CVF winter conference on applications of computer vision*, pages 2218–2227, 2020. 3

[18] Tianyu Pang, Kun Xu, and Jun Zhu. Mixup inference: Better exploiting mixup to defend adversarial attacks. *arXiv preprint arXiv:1909.11515*, 2019. 3

[19] Charles R Qi, Hao Su, Kaichun Mo, and Leonidas J Guibas. Pointnet: Deep learning on point sets for 3d classification and segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 652–660, 2017. 1

[20] Mengye Ren, Eleni Triantafillou, Sachin Ravi, Jake Snell, Kevin Swersky, Joshua B Tenenbaum, Hugo Larochelle, and Richard S Zemel. Meta-learning for semi-supervised few-shot classification. *arXiv preprint arXiv:1803.00676*, 2018. 2, 3

[21] Aniket Roy, Anshul Shah, Ketul Shah, Prithviraj Dhar, Anoop Cherian, and Rama Chellappa. Felmi: few shot learning with hard mixup. *Advances in Neural Information Processing Systems*, 35:24474–24486, 2022. 3

[22] Jake Snell, Kevin Swersky, and Richard Zemel. Prototypical networks for few-shot learning. *Advances in neural information processing systems*, 30, 2017. 1, 2, 3, 4, 6

[23] Stefan Stojanov, Anh Thai, and James M Rehg. 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*, pages 1798–1808, 2021. 6

[24] Jiachen Sun, Qingzhao Zhang, Bhavya Kailkhura, Zhiding Yu, Chaowei Xiao, and Z. Morley Mao. Benchmarking robustness of 3d point cloud recognition against common corruptions. *arXiv preprint arXiv:2201.12296*, 2022. 2, 6

[25] Flood Sung, Yongxin Yang, Li Zhang, Tao Xiang, Philip HS Torr, and Timothy 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. 1, 3

[26] Yonglong Tian, Yue Wang, Dilip Krishnan, Joshua B Tenenbaum, and Phillip Isola. Rethinking few-shot image classification: a good embedding is all you need? In *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part XIV 16*, pages 266–282. Springer, 2020. 1, 3

[27] Mikaela Angelina Uy, Quang-Hieu Pham, Binh-Son Hua, Duc Thanh Nguyen, and Sai-Kit Yeung. Revisiting point cloud classification: A new benchmark dataset and classification model on real-world data. In *International Conference on Computer Vision (ICCV)*, 2019. 2, 3, 6

[28] Vikas Verma, Alex Lamb, Christopher Beckham, Amir Najafi, Ioannis Mitliagkas, David Lopez-Paz, and Yoshua Bengio. Manifold mixup: Better representations by interpolating hidden states. In *International conference on machine learning*, pages 6438–6447. PMLR, 2019. 2, 3, 4, 7

[29] Oriol Vinyals, Charles Blundell, Timothy Lillicrap, Daan Wierstra, et al. Matching networks for one shot learning. *Advances in neural information processing systems*, 29, 2016. 1, 2, 3

[30] Yufei Wang, Haoliang Li, and Alex C Kot. Heterogeneous domain generalization via domain mixup. In *ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 3622–3626. IEEE, 2020. 3

[31] Yue Wang, Yongbin Sun, Ziwei Liu, Sanjay E Sarma, Michael M Bronstein, and Justin M Solomon. Dynamic graph cnn for learning on point clouds. *Acm Transactions On Graphics (tog)*, 38(5):1–12, 2019. 1, 3, 4, 6

[32] Lin Wu, Yang Wang, Feng Zheng, Qi Tian, and Meng Wang. T-person-gan: Text-to-person image generation with identity-consistency and manifold mix-up. *arXiv preprint arXiv:2208.12752*, 2022. 3

[33] Zhirong Wu, Shuran Song, Aditya Khosla, Fisher Yu, Linguang Zhang, Xiaoou Tang, and Jianxiong Xiao. 3d shapenets: A deep representation for volumetric shapes. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1912–1920, 2015. 2, 3, 4, 6

[34] Chuanguang Yang, Zhulin An, Helong Zhou, Linhang Cai, Xiang Zhi, Jiwen Wu, Yongjun Xu, and Qian Zhang. Mixskd: Self-knowledge distillation from mixup for image recognition. In *European Conference on Computer Vision*, pages 534–551. Springer, 2022. 3

[35] Minmin Yang, Jiajing Chen, and Senem Velipasalar. Cross-modality feature fusion network for few-shot 3d point cloud classification. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pages 653–662, 2023. 1, 3

[36] Huaxiu Yao, Yiping Wang, Linjun Zhang, James Y Zou, and Chelsea Finn. C-mixup: Improving generalization in regression. *Advances in Neural Information Processing Systems*, 35:3361–3376, 2022. 3

[37] Chuangguan Ye, Hongyuan Zhu, Yongbin Liao, Yanggang Zhang, Tao Chen, and Jiayuan 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

[38] Chuangguan Ye, Hongyuan Zhu, Bo Zhang, and Tao Chen. A closer look at few-shot 3d point cloud classification. *International Journal of Computer Vision*, 131(3):772–795, 2023. 1

[39] Sergey Zagoruyko and Nikos Komodakis. Wide residual networks. *arXiv preprint arXiv:1605.07146*, 2016. 1

[40] Hongyi Zhang, Moustapha Cisse, Yann N Dauphin, and David Lopez-Paz. mixup: Beyond empirical risk minimization. *arXiv preprint arXiv:1710.09412*, 2017. 3