Supplementary Material for
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

1. Details about 3D Point Cloud Backbones

In this section, we present more structural details about how we incorporate SimpliMix into the backbones deployed in Section 4 of our main paper.

ViewNet [1] ViewNet is a projection-based backbone. It consists of two branches: the projection feature learning branch and the point feature learning branch. The projection feature learning branch accepts the input depth images and processes them independently using a sequence of convolutional layers. The point feature learning branch learns depth image features based on the output of View Pooling module, which combines different projected plane combinations into five groups and performs max-pooling on each of them. We only incorporate the SimpliMix into the projection feature learning branch for simplicity as shown in Fig. 1 below.

DGCNN [2] DGCNN is a point-based network, which consists of four EdgeConv layers. The EdgeConv acts on the k-nearest neighbor graphs of point features and generates edge features. Fig. 2 below shows possible places where SimpliMix may be incorporated into DGCNN.

2. Details about Datasets

In this section, we provide more details about the datasets used in our experiments. First, we present the correspondence between class IDs and class names for ModelNet40-FS and ModelNet40-C-FS in Tab. 1 and ScanObjectNN-FS in Tab. 2. Then, we provide the class details of cross-domain experimental setups in Tab. 3 and Tab. 4.

3. Pseudocode for QMix and SQMix

Two alternatives for SimpliMix are QMix and SQMix. The QMix only mixes query samples and leaves support samples as they are. The SQMix randomly shuffles the classes, and makes sure that samples from one set are mixed with samples from the same set. For example, if class A is mixed with class B, then all the support (query) samples belonging to class A are mixed with support (query) samples belonging to class B. We provide the pseudocode for QMix and SQMix in Algorithm 1 and Algorithm 2 below.

ID	Class Name	ID	Class Name	ID	Class Name	ID	Class Name
0	airplane	10	cup	20	laptop	30	sofa
1	bathtub	11	curtain	21	mantel	31	stairs
2	bed	12	desk	22	monitor	32	stool
3	bench	13	door	23	night stand	33	table
4	bookshelf	14	dresser	24	person	34	tent
5	bottle	15	flower pot	25	piano	35	toilet
6	bowl	16	glass box	26	plant	36	tv stand
7	car	17	guitar	27	radio	37	vase
8	chair	18	keyboard	28	range hood	38	wardrobe
9	cone	19	lamp	29	sink	39	xbox

Table 1. Class IDs and class names for ModelNet40-FS and ModelNet40-C-FS.

ID	Class Name	ID	Class Name	ID	Class Name
0	bag	5	desk	10	bed
1	bin	6	display	11	pillow
2	box	7	door	12	sink
3	cabinet	8	shelf	13	sofa
4	chair	9	table	14	toilet

Table 2. Class IDs and class names for ScanObjectNN-FS dataset

ShapeNetCore-XFS → ScanObjectNN		
Dataset	Domain	Name of class
ShapeNetCore-XFS	Source	airplane, basket, bathtub, bench, bicycle, birdhouse, bottle, bowl, bus, camera, can, cap, car, clock, keyboard, dishwasher, earphone, faucet, file cabinet, guitar, helmet, jar, knife, lamp, laptop, loudspeaker, microphone, microwaves, motorbike, mug, piano, pistol, flowerpot, printer, remote, rifle, rocket, skateboard, stove, telephone, tower, train, watercraft, washer
ScanObjectNN	Target	bag, bin, box, cabinet, chair, desk, display, door, shelf, table, bed, pillow, sink, sofa, toilet

Table 3. Details of classes included in the cross-domain experiments with ShapeNetCore-XFS as the source domain and ScanObjectNN as the target domain.

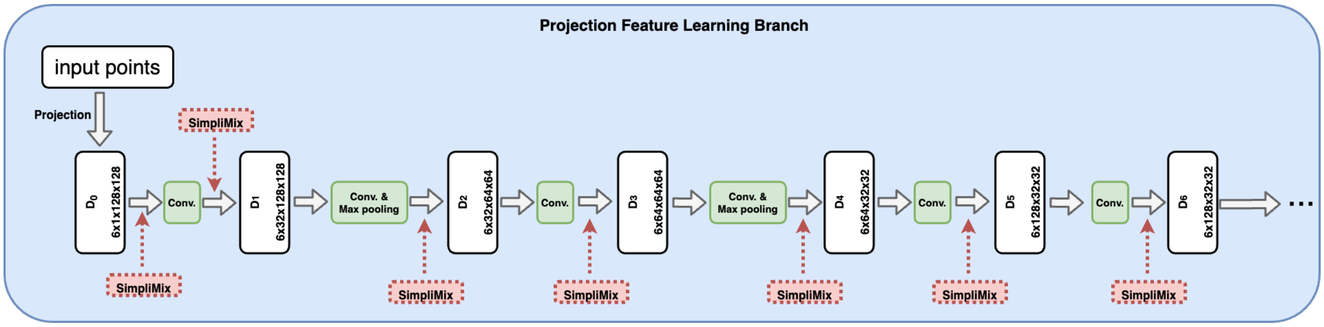

Figure 1. The structure of the point feature learning branch in ViewNet. We point out the possible places where the SimpliMix may be applied. The feature map D_6 is further processed by additional layers, which are ignored in the figure.

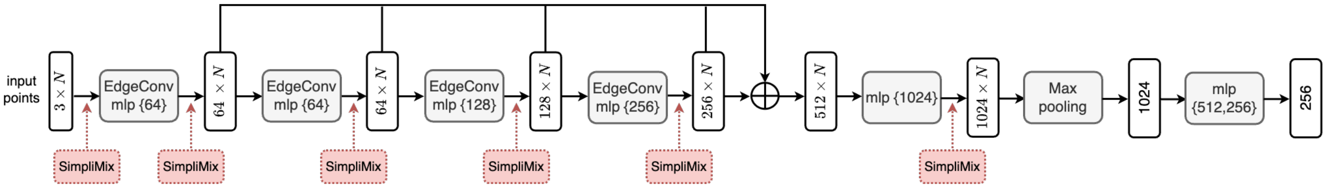

Figure 2. The architecture of DGCNN. We point out the possible places where SimpliMix may be applied.

ModelNet40-XFS → ScanObjectNN		
Dataset	Domain	Name of class
ModelNet40-XFS	Source	airplane, bathtub, bottle, bowl, car, cone, cup, curtain, flower pot, glass box, guitar, keyboard, lamp, laptop, mantel, nightstand, person, piano, plant, radio, range hood, stairs, tent, tv stand, vase, xbox
ScanObjectNN	Target	bag, box, desk, pillow, sofa, bed, cabinets, display, shelves, table, bin, chair, door, sink, toilet

Table 4. Details of classes included in the ModelNet40-XFS, which is the source domain dataset, and ScanObjectNN, which is the target domain dataset for cross-domain experiments. The ModelNet40-C-XFS has the same classes as the ModelNet40-XFS.

Algorithm 1 QMix($x, y, \lambda, n\_way, k\_shot, m\_query$) in PyTorch style.

```
function QMix($x, y, lam, n\_way, k\_shot, m\_query,$)
    $support = x[: n\_way \times k\_shot]$    ▷ Get support samples
    $query = x[n\_way \times k\_shot :]$    ▷ Get qeury samples
    $query\_size = query.shape[0]$
    $indices = randperm(query\_size)$
    $mixed\_query = Mix\_X(query, query[indices], lam)$    ▷ Get mixed query samples
    $sppt\_y, qry\_y = y[: n\_way \times k\_shot], y[n\_way \times k\_shot :]$
    $qry\_y\_a, qry\_y\_b = qry\_y, qry\_y[indices]$
    $mixed\_x = concatenate(support, mixed\_query)$
    $y\_a = concatenate(sppt\_y, qry\_y\_a)$
    $y\_b = concatenate(sppt\_y, qry\_y\_b)$
    return $mixed\_x, y\_a, y\_b, lam$
end function
```

References

[1] 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] 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

Algorithm 2 SQMix($x, y, \lambda, n\_way, k\_shot, m\_query$) in PyTorch style.

```
function SQMix($x, y, lam, n\_way, k\_shot, m\_query$)
    $cls\_indices = randperm(n\_way)$                                        ▷ Random permutation of classes
    $sppt\_x, query\_x = x[: n\_way \times k\_shot], x[n\_way \times k\_shot :]$
    $sppt\_y, query\_y = y[: n\_way \times k\_shot], y[n\_way \times k\_shot :]$
    $sppt\_x = sppt\_x.reshape(n\_way, k\_shot, *sppt\_x.shape[1 :])$
    $query\_x = query\_x.reshape(n\_way, m\_query, *query\_x.shape[1 :])$
    $sppt\_y = sppt\_y.reshape(n\_way, k\_shot)$
    $query\_y = query\_y.reshape(n\_way, m\_query)$
    $sppt\_x1, sppt\_x2 = sppt\_x, sppt\_x[cls\_indices]$
    $query\_x1, query\_x2 = query\_x, query\_x[cls\_indices]$
    $sppt\_x1 = sppt\_x1.reshape(n\_way \times k\_shot, *sppt\_x1.shape[2 :])$
    $sppt\_x2 = sppt\_x2.reshape(n\_way \times k\_shot, *sppt\_x2.shape[2 :])$
    $query\_x1 = query\_x1.reshape(n\_way \times m\_query, *query\_x1.shape[2 :])$
    $query\_x2 = query\_x2.reshape(n\_way \times m\_query, *query\_x2.shape[2 :])$
    $mixed\_sppt\_x = Mix\_X(sppt\_x1, sppt\_x2, lam)$
    $mixed\_query\_x = Mix\_X(query\_x1, query\_x2, lam)$
    $sppt\_y\_a, sppt\_y\_b = sppt\_y, sppt\_y[cls\_indices]$
    $qry\_y\_a, qry\_y\_b = query\_y, query\_y[cls\_indices]$
    $sppt\_y\_a, sppt\_y\_b = sppt\_y\_a.reshape(-1), sppt\_y\_b.reshape(-1)$
    $qry\_y\_a, qry\_y\_b = qry\_y\_a.reshape(-1), qry\_y\_b.reshape(-1)$
    $mixed\_x = concatenate(mixed\_sppt\_x, mixed\_query\_x)$                  ▷ Concatenate mixed data
    $y\_a = concatenate(sppt\_y\_a, qry\_y\_a)$                                ▷ Concatenate mixed labels
    $y\_b = concatenate(sppt\_y\_b, qry\_y\_b)$
    return $mixed\_x, y\_a, y\_b, lam$
end function
```