Supplementary Materials
MetaSets: Meta Learning on Point Sets for Generalizable Representations

Chao Huang*, Zhangjie Cao*, Yunbo Wang*, Jianmin Wang, Mingsheng Long (✉)†
School of Software, BNRist, Tsinghua University, China
`{microhhh9,caozhangjie14,yunbo.thu}@gmail.com, {jimwang,mingsheng}@tsinghua.edu.cn`

1. Experimental Details

We introduce additional experiment details including dataset details and hyper-parameters details in this section.

1.1. Dataset Construction

1.1.1 Sim-to-Real dataset

Our Sim-to-Real dataset consists of three domains: ModelNet, ShapeNet and ScanObjectNN.
ModelNet[1] is a comprehensive clean collection of 3D CAD models of 40 common object categories in the world. The CAD models are collected from online search engines by querying for each object category term. The dataset has two forms: ModelNet40 and ModelNet10. ModelNet40 contains all the CAD models of all the categories while ModelNet10 contains 10 popular object categories from the 40 categories, where models that did not belong to these categories are manually deleted. We use ModelNet40 in the experiments, and use the official training set as the meta-training set.
ShapeNet[2] consists of several subsets: ShapeNetCore for classification and ShapeNetSem for segmentation. We use ShapeNetCore in the experiments. ShapeNetCore is a subset of the full ShapeNet dataset with single clean 3D models and manually verified category and alignment annotations. It covers 55 common object categories with about 51,300 unique 3D models. We use the official split of training and validation as our training and validation set.
ScanObjectNN[3] is a new real-world point cloud object dataset based on scanned indoor scene data, which is built on two popular scene meshes datasets: SceneNN [4] and ScanNet [2]. It contains about 15,000 objects of 15 categories with 2,902 unique object instances. We use the official split with about 80% training shapes and about 20% testing shapes.

*Equal contribution
†Corresponding author: Mingsheng Long
[1]https://modelnet.cs.princeton.edu/
[2]https://www.shapenet.org/
[3]https://hkust-vgd.github.io/scanobjectnn/

Table 1. Selected categories for each domain generalization benchmark.

ModelNet→ScanObjNN	ShapeNet→ScanObjNN
Bed	Bag
Cabinet (Dresser, Wardrobe)	Bed
Chair (Bench, Chair, Stool)	Cabinet
Desk	Chair
Display (Monitor)	Display
Door	Pillow
Shelf (Bookshelf)	Shelf (Bookshelf)
Sink	Sofa
Sofa	Table
Table	-
Toilet	-

For both ModelNet40 and ShapeNet, we adopt the method from Qi *et al.* [6] to generate the point clouds. The shared categories in each dataset are shown in Table 1. As for data pre-processing, we follow the work from Qin *et al.* [7] to normalize the point clouds into a unit ball.

1.2. Hyperparameters for Transformed Point Sets

As stated in the main text, we first confirm a valid hyperparameter range that changes the geometry of the shape but still make it recognizable, and then we randomly sample in the range. We try different random sampling method. We find that total random sampling sometimes samples similar hyperparameters, which decreases the diversity of the geometry priors contained in each transformed dataset. Therefore, we use a stratified sampling strategy that first evenly divides the valid hyperparameter range into 3 sub-ranges, where 3 is the number of specific transformations for each type of transformation. Then we randomly sample a hyperparameter in each sub-range. We use the same transformation hyperparameters for ModelNet→ScanObjNN and ShapeNet→ScanObjNN, which are shown in Table 3.

Method	bed	cabinet	chair	desk	display	door	shelf	sink	sofa	table	toilet	Avg
PointNet [6]	40.91	1.33	**97.44**	50.00	61.90	**100.00**	61.22	37.50	52.38	51.85	29.41	53.09
MetaSets on PointNet	**63.64**	**18.67**	94.87	**53.33**	**69.05**	95.24	**75.51**	37.50	**71.43**	**74.07**	**64.71**	**65.27**

Table 2. Accuracy per class (%) and the average on ModelNet→ScanObjNN.

Table 3. Hyperparameters for different transformed point sets.

Transformation	Param 1	Param 2	Param 3
Self-occlusion (grid size W)	0.035	0.022	0.017
Non-uniform density (gate g)	1.3	1.4	1.6
Dropping (drop ratio $x\%$)	24%	36%	45%

1.3. Training Hyperparameters

In all experiments, the validation convergence bound ϵ is set to 0.1%, and the batch size is set to 128. We use Adam optimizer [5]. For all benchmarks, η is 0.0003, β is 0.001. We conduct experiments on a machine with 64 CPUs and 4 GeForce 2080ti GPUs. The total training of 18,000 iterations uses about 16 hours.

2. More Experimental Results

In this section, we show more experiment results including class-wise results in the ModelNet→ScanObjNN dataset, variants of soft-sampling, variants of meta-training and meta-objective, static or dynamic transformation and hyper-parameter sensitivity.

2.1. Class-wise results.

Table 2 shows the class-wise classification accuracy. We can observe that MetaSets outperforms the backbone network PointNet remarkably on most classes. In particular, on difficult classes such as *cabinet*, PointNet yields extremely low accuracy due to the large domain shift, *e.g.*, the cabinets from the synthetic dataset has complex and detailed inner structures that are probably invisible in the real dataset. However, the expanded tasks of MetaSets can induce a larger set of geometry priors, which have a higher chance to include geometry priors that are similar to those from the target domain. Such priors enable MetaSets to mitigate the domain shift and perform still strongly for difficult cases of imperfect point clouds. Even though on some classes like chair and door, MetaSets does not perform as well as PointNet. We manually check the objects of these classes in ModelNet and ScanObjectNN datasets and find that the objects are in different shapes like different kinds of chairs. So the classes suffer from huge domain shift but not geometric shift, which can only be addressed by access to the target data and is not the focus of the paper.

Table 4. Variants of Meta-Validation and Meta-Objective. 'Per-batch $\mathcal{D}_s^{\text{train}}$' means meta-validation on a batch of meta-training set. '$\mathcal{D}_s^{\text{train}}$' means meta-validation on the meta-training set $\mathcal{D}_s^{\text{train}}$. 'Per-batch $\mathcal{D}_s^{\text{val}}$' means meta-validation on a batch of meta-validation set. '$\mathcal{D}_s^{\text{train}}$' means meta-validation on the meta-validation set $\mathcal{D}_s^{\text{val}}$, which is the proposed MetaSets.

Per-batch $\mathcal{D}_s^{\text{train}}$	$\mathcal{D}_s^{\text{train}}$	Per-batch $\mathcal{D}_s^{\text{val}}$	$\mathcal{D}_s^{\text{val}}$ (Proposed)
63.16	64.42	66.53	68.28

2.2. Variants of Soft-sampling

We further explore the soft-sampling algorithm. We compare performance of soft-sampling probabilities computed by meta-validation based on different sets: a mini-batch of $\mathcal{D}_s^{\text{train}}$, the whole $\mathcal{D}_s^{\text{train}}$, a mini-batch $\mathcal{D}_s^{\text{val}}$, and the whole $\mathcal{D}_s^{\text{val}}$ (the proposed MetaSets). As shown in Table 4, computing the soft-sampling probabilities based on the whole $\mathcal{D}_s^{\text{val}}$ outperforms all the other variants. This can be explained by that computing the probability of a batch of data is unstable while meta-validation on $\mathcal{D}_s^{\text{train}}$ causes the overfitting problem.

Table 5. Variants of Meta-Training and Meta-Objective. 'Mixture' means mixing the data of all three transformations into a single task and conduct meta-learning on the single task. 'Maximum Loss' means minimizing the maximum meta-training loss among all tasks.

Mixture	Maximum Loss	MetaSets
63.47	61.89	68.28

2.3. Variants of Meta-Training and Meta-Objective

One close variant of the proposed MetaSets is conduct meta-learning on the mixture of the data from three transformations, which uses no soft-sampling but combine the sets of data from three transformations into one set. The variant influences the gradient computation on Line 9-10 in the algorithm. We show the result as 'Mixture' in Table 5, where we observe that MetaSets outperforms 'Mixture'. The result indicates that separating the different transformations into different tasks is important to learning the knowledge from different tasks while mixing the data from different transformations may disentangle the knowledge.

According to [1], when minimizing the losses for different tasks, a alternative is to maximizing the maximum of all the losses, which may increase the convergence speed. We compare MetaSets by minimizing all the losses with

MetaSets by minimizing the maximum loss of all the tasks ('Maximum Loss') in Table 5, we can observe that MetaSets achieves higher accuracy than 'Maximum Loss', which indicates the necessity of minimizing all the task losses. This observation matches the claim in [8] that directly using *distributionally robust optimization* (DRO) [3] still achieves low worst-group test accuracy.

Table 6. Time for each epoch (s) and accuracy (%) for PointNet/MetaSets based on the dynamic transformation and the static transformation.

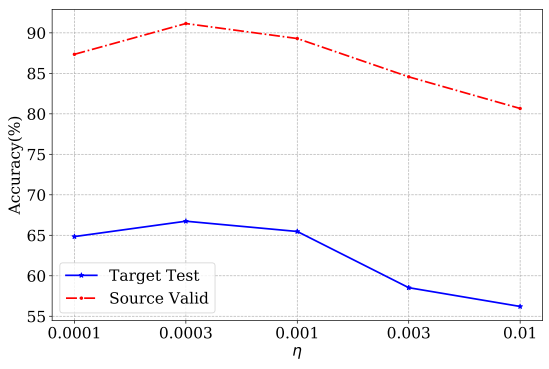
Transformation	PointNet		MetaSets	
	Time	Acc	Time	Acc
Dynamic	358	63.16	1423	68.28
Static	334	45.89	1350	49.21

2.4. Static or Dynamic Transformation

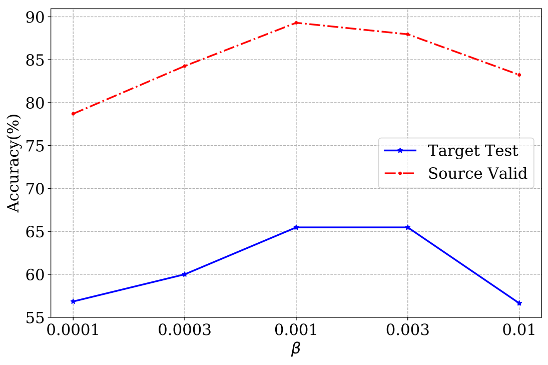
In MetaSets, we need to randomize the parameters $\vec{v}$, P_1 and P_2 in every training iteration (dynamic transformation), which needs an extra transformation cost in every iteration. A more efficient way is using static transformation, where we first transform each point cloud with one parameter into a transformed point cloud and form a set of transformed point clouds. We demonstrate that dynamic transformation is necessary and does not introduce two much cost than static transformation. We compare PointNet/MetaSets based on dynamic transformation with PointNet/MetaSets based on static transformation. As shown in Table 6, the dynamic transformation approach achieves much higher performance than static transformation in accuracy but only little more cost time for each epoch for both PointNet and MetaSets. Also, the number of epochs to converge is about 30 for all experiments. Therefore, the dynamic transformation significantly improves the performance with only little more time cost.

2.5. Parameter Sensitivity

We test the classification accuracy for different learning rates η and β on the ModelNet→ScanObjNN benchmark. To find the best-performing value of η, we fix the β, and vice versa. The results are shown in Figure 1, we can observe that η is not sensitive in the range of $[0.0001, 0.001]$ and β is not sensitive in the range of $[0.001, 0.003]$. However, even the accuracy drops out of these ranges. The trend of the validation accuracy curve and the test accuracy curve are similar, which indicates that the best learning rates can be obtained by cross-validation on the source validation set.



(a) η



(b) β

Figure 1. Sensitivity analyses about η and β on the ModelNet→ScanObjNN benchmark.

References

[1] Martin Arjovsky, Léon Bottou, Ishaan Gulrajani, and David Lopez-Paz. Invariant risk minimization. *arXiv preprint arXiv:1907.02893*, 2019.

[2] Angela Dai, Angel X Chang, Manolis Savva, Maciej Halber, Thomas Funkhouser, and Matthias Nießner. Scannet: Richly-annotated 3d reconstructions of indoor scenes. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5828–5839, 2017.

[3] John Duchi, Peter Glynn, and Hongseok Namkoong. Statistics of robust optimization: A generalized empirical likelihood approach. *arXiv preprint arXiv:1610.03425*, 2016.

[4] Binh-Son Hua, Quang-Hieu Pham, Duc Thanh Nguyen, Minh-Khoi Tran, Lap-Fai Yu, and Sai-Kit Yeung. Scenenn: A scene meshes dataset with annotations. In *2016 Fourth International Conference on 3D Vision (3DV)*, pages 92–101. IEEE, 2016.

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

[6] Charles R Qi, Hao Su, Kaichun Mo, and Leonidas J Guibas. PointNet: Deep learning on point sets for 3D classification and segmentation. In *CVPR*, pages 652–660, 2017.

[7] Can Qin, Haoxuan You, Lichen Wang, C-C Jay Kuo, and Yun Fu. PointDAN: A multi-scale 3D domain adaption network for point cloud representation. In *NeurIPS*, pages 7190–7201, 2019.

[8] Shiori Sagawa, Pang Wei Koh, Tatsunori B Hashimoto, and Percy Liang. Distributionally robust neural networks. In *International Conference on Learning Representations*, 2019.