Manifold Adversarial Learning for Cross-domain 3D Shape Representation - Supplementary Material

Hao Huang[1,2,3,4], Cheng Chen[1,3,4], and Yi Fang[1,2,3,4†]

[1] NYU Multimedia and Visual Computing Lab
[2] NYUAD Center for Artificial Intelligence and Robotics
[3] NYU Tandon School of Engineering, New York University, USA
[4] New York University, Abu Dhabi, UAE
{hh1811,cc6858,yfang}@nyu.edu

1 Lie Group and Transformation

This section describes properties of transformation groups used in the main paper as a reference. Lengthy derivations are omitted and more details can be found in a technical report [3].

1.1 Lie group and Lie algebra

A Lie group $\mathcal{T}$ is a smooth differentiable manifold as well as a group. The collection of matrices $s \in \mathrm{SO}(3) \subset \mathbb{R}^{n\times n}$ forms a Lie group. The multiplication and inversion operations defined for the group SO(3) are implemented as matrix multiplication and inversion. As this group forms a specific subset of non-singular $n \times n$ matrices, its degrees of freedom are less than n^2.

Consider the Lie group $\mathcal{T}$ represented in $\mathbb{R}^{n\times n}$, with k degrees of freedom. The Lie algebra $\mathfrak{g}$ is the tangent space around the identity of $\mathcal{T}$. This tangent space is a k-dimensional vector space with basis elements $\{G_1, \cdots, G_k\}$ which is named *generators*. Elements of $\mathfrak{g}$ are represented as matrices in $\mathbb{R}^{n\times n}$, but under addition and scalar multiplication, rather than matrix multiplication. For such a Lie algebra $\mathfrak{g}$, we denote the linear combination of generators $\{G_i\}$ specified by a vector of coefficients g as $\mathrm{alg}(g)$:

$$\mathrm{alg}(g) : \mathbb{R}^k \to \mathfrak{g} \subset \mathbb{R}^{n\times n}, \;\; g \mapsto \sum_{i=1}^{k} g_i G_i \;\; . \tag{1}$$

Note that the tangent vector is in fact an $n \times n$ matrix and can be represented as the vector of coefficients of the generators.

1.2 Exponential map

The exponential map takes elements $\mathfrak{g}$ in the algebra to elements in the group $\mathcal{T}$. It moves along the group (transformation) manifold in the differential direction specified by the tangent vector in the algebra. For matrix groups, the exponential map is matrix exponentiation defined as:

$$\exp(X) : \mathfrak{g} \to \mathcal{T}, \;\; X \mapsto \sum_{k=0}^{\infty} \frac{1}{k!} X^k, \;\; \forall X \in \mathfrak{g} \;\; . \tag{2}$$

1.3 SO(3) transformation

SO(3) is the group of rotations in 3D space, represented by 3×3 orthogonal matrices with unit determinant. It has *three* degrees of freedom: one for each rotation axis. The identity element in SO(3) is $e = \mathbf{I}_{3\times3}$.

The Lie algebra $\mathfrak{so}(3)$ is the set of antisymmetric 3×3 matrices, generated by the differential rotations along each axis:

$$G_1 = \begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & -1 \\ 0 & 1 & 0 \end{pmatrix}, G_2 = \begin{pmatrix} 0 & 0 & 1 \\ 0 & 0 & 0 \\ -1 & 0 & 0 \end{pmatrix}, G_3 = \begin{pmatrix} 0 & -1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix} \;\; . \tag{3}$$

The mapping $\mathrm{alg}(g) : \mathbb{R}^3 \to \mathfrak{so}(3)$ sends a 3-dimensional vector to a skew matrix:

$$\mathrm{alg}(g) = \begin{pmatrix} 0 & -c & b \\ c & 0 & -a \\ -b & a & 0 \end{pmatrix}, \;\; \forall g = [a, b, c]^\top \in \mathbb{R}^3 \;\; . \tag{4}$$

1.4 Retraction from $\boldsymbol{\theta}$ to $\boldsymbol{\mathcal{M}(P)}$

Given the adversarial transformation parameter θ_s^a computed along the search direction g, and let G_i (defined in Eq. 3) be the basis of the tangent space $T_e\mathcal{T}$ where $\mathcal{T} = \mathrm{SO}(3)$, also denoted as $\mathfrak{g}$ in above, at the identity element $e = \mathbf{I}_{3\times3}$ of the transformation group $\mathcal{T}$, the retraction [12] at P_t can be summarized as:

1. Mapping θ_s^a to a tangent vector in $T_e\mathcal{T}$ using G_i defined in Eq. 4 as $\mathrm{alg}(\theta_s^a) = \sum_{j=0}^{2} \theta_s^a[i] G_i$;
2. Mapping the resultant tangent vector $\mathrm{alg}(\theta_s^a)$ in $T_e\mathcal{T}$ to a transformation $t^a \in \mathcal{T}$ using exponential map defined in Eq. 2 as $t^a = \exp(\mathrm{alg}(\theta_s^a)) = \exp\left(\sum_{j=0}^{2} \theta_s^a[i] G_i\right)$;
3. Mapping the resultant transformation $t^a \in \mathcal{T}$ back to PCAM $\mathcal{M}(P)$ using $\mathcal{R}_{P_t}(g) = P_{t^a}$.

1.5 Jacobian matrix of point cloud

Given a transformation t parameterized by $\theta_t \in \mathbb{R}^M$ and we denote the m-th parameter of t as θ_t^m where $m \in \{1, 2, \cdots, M\}$. The jacobian matrix $\mathcal{J}_{P_t} \in$

$\mathbb{R}^{(N\times 3)\times M}$ of the action of t on a point cloud P with N points $\{p_i\}_{i=1}^N$ is defined as follows:

$$\mathcal{J}_{P_t} = \begin{bmatrix} \frac{\partial P_t}{\partial \theta_t^1} & \frac{\partial P_t}{\partial \theta_t^2} & \cdots & \frac{\partial P_t}{\partial \theta_t^M} \end{bmatrix} = \begin{bmatrix} \frac{\partial p_1}{\partial \theta_t^1} & \frac{\partial p_1}{\partial \theta_t^2} & \cdots & \frac{\partial p_1}{\partial \theta_t^M} \\ \frac{\partial p_2}{\partial \theta_t^1} & \frac{\partial p_2}{\partial \theta_t^2} & \cdots & \frac{\partial p_2}{\partial \theta_t^M} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial p_N}{\partial \theta_t^1} & \frac{\partial p_N}{\partial \theta_t^2} & \cdots & \frac{\partial p_N}{\partial \theta_t^M} \end{bmatrix} . \quad (5)$$

We define $\partial\theta_t^m$ as the m-th entry of a small perturbation $\Delta\theta$ added to θ_t, and ∂p_i as the difference between p_i and the nearest point (under the L^2 metric in Euclidean space) $p \in P_t'$, where P_t' is the point cloud transformed from P by $t \circ \Delta t$ (*i.e.*, changing the parameters from θ_t to $\theta_t + \Delta\theta$).

2 Dataset Description

The Sim-to-Real benchmarks proposed in [10] consist of two synthetic datasets, ModelNet [30] and ShapeNet [2], and a real-scanned dataset ScanObjectNN [24]. The three datasets are calibrated to serve as three domains. ModelNet40 is a collection of clean 3D CAD models which consists of 9,843 training shapes and 2,468 test shapes across 40 categories. ShapeNetCore is a subset of a larger ShapeNet dataset and it contains around 51,300 models covering 55 categories. ScanobjectNN includes about 15,000 models across 15 categories which are captured from real-scanned indoor scene. We use the shared categories in each dataset as shown in Tab. 1 as proposed in [10]. The categories with similar object appearance but different names are merged together.

ModelNet40 → ScanObjectNN	**ShapeNet → ScanObjectNN**
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	-

Table 1. Shared categories in two domains consisting of three datasets.

Algorithm 1 Training scheme

```
Input: source domain $\mathbb{S} = \{D^{tr}_{\mathbb{S}}, D^{val}_{\mathbb{S}}\}$, transformation set $\mathcal{T}$, neural network $f_w$,
    memory $\boldsymbol{M}$, number of the generated adversarial samples $M$, number of top-$N$
    adversarial samples to move into memory, batch size $B$, validation error bound $\epsilon$,
    number of sampled transformation (per point cloud) $K$, learning rates $\eta$ (inner-
    loop) and $\beta$ (outer-loop)
 1: Initialize $w \leftarrow w_0$
 2: while $\mathcal{L}^{val} < \epsilon$ do
 3:     repeat                                                        ▷ Training phase
 4:         Sample $\{(P_i, y_i)\}_{i=1}^{B}$ from $D^{tr}_{\mathbb{S}}$
 5:         Sample $\{t_k\}_{k=1}^{K}$ from $\mathcal{T}$ with uniform distribution          ▷ Sample $K$ Tfm.
 6:         for $k = 1, \ldots, K$ do
 7:             Apply $t_k$ to $\{P_i\}_{i=1}^{B}$ to get $\{P_{i,t_k}\}_{i=1}^{B}$          ▷ Obtain transformed PCs.
 8:             if $\boldsymbol{M}$ contains at least $B$ samples then
 9:                 Sample $\{P_i^a\}_{i=1}^{B}$ from $\boldsymbol{M}$
10:             end if
11:             Merge $\{P_{i,t_k}\}_{i=1}^{B}$ and $\{P_i^a\}_{i=1}^{B}$ to $\{P_i\}_{i=1}^{2B}$
12:             Compute $\mathcal{L}_k^{tr}(P) \leftarrow f_w(\{P_i\}_{i=1}^{2B})$
13:             $w'_k \leftarrow w - \eta \nabla_w \mathcal{L}_k^{tr}(P)$
14:         end for
15:         Compute $\mathcal{L}^{tr}(P) \leftarrow \sum_{k=1}^{K} \mathcal{L}^{tr}(f_{w'_k}(\{P_i\}_{i=1}^{2B}))$
16:         $w \leftarrow w - \beta \nabla_w \mathcal{L}^{tr}(P)$
17:         Compute projected gradient $g = \mathcal{J}^{+}_{P_t} \nabla_{P_t} \mathcal{L}^{tr}(P_t)$   ▷ Eq. 3 in the main text
18:         Generate adversarial $\{P_i^a\}_{i=1}^{M}$ using retraction       ▷ Eq. 7 in the main text
19:         Compute geodesic distance $d_P(e, t_i)$                          ▷ Eq. 13 in the main text
20:         Compute move-in/out Prob. of $\{P_i^a\}_{i=1}^{M}$              ▷ Sec. 3.4 in the main text
21:         Select $N$ adversarial from $\{P_i^a\}_{i=1}^{M}$ and add into $\boldsymbol{M}$ according to Prob.
22:     until end of $D^{tr}_{\mathbb{S}}$
23:     repeat                                                        ▷ Validation phase
24:         Sample $\{(P_i, y_i)\}_{i=1}^{B}$ from $D^{val}_{\mathbb{S}}$
25:         Sample $\{t_i\}_{i=1}^{B}$ from $\mathcal{T}$ with uniform distribution          ▷ Sample $B$ Tfm.
26:         Compute $\mathcal{L}^{val}(P) \leftarrow \mathcal{L}^{val}(f_w(\{P_i\}_{i=1}^{B}))$
27:     until end of $\mathcal{D}^{val}_{\mathbb{S}}$
28: end while
```

3 Training Scheme

In Algorithm 1, we detail the training scheme which is based on MAML [4], *i.e.*, gradient-based model agnostic meta-learning method. We follow the training scheme as in [10] but incorporate the adversarial samples and adaptive memory. The referred equations are provided in the main paper.

4 Related Work

Deep learning for point clouds. Various DNN architectures have been proposed to learn point cloud representations for 3D vision tasks. PointNet [19] and PointNet++ [20] are pioneering works for point cloud classification and segmentation by applying 1D convolution to unordered points. PointCNN [15] proposes special operators to reorganize unordered point clouds and connects multiple point clouds into a graph. PointConv [29] constructs convolution kernels as nonlinear functions of point coordinates. DGCNN [28] performs graph convolutions on local neighborhood regions in graphs to capture local geometric features of point clouds. DeepGCN [14] investigates the benefits of increasing the depth of graph convolutional networks [31]. ConvPoint [1] proposes to generalize discrete convolutional neural networks by replacing discrete kernels with continuous ones. Inspired by Transformer [25] for machine translation, Point Transformer [35] designs vector self-attention to aggregate geometric features for each local neighborhood for point cloud. However, all the mentioned works only focus on a single domain where both training and testing data are sampled from *i.i.d.* distributions.

Cross-domain representation learning. Uy *et al.* [24] shows that DNNs used for 3D vision have a tendency to overfit to the geometry of synthetic point clouds, and the performance suffers degradation when applied to real-world scanned point clouds due to shifts in geometry or changes in density. Cross-domain representation learning (for both DA and DG) has been extensively studied in 2D vision tasks [8, 5, 34, 6], but has not yet been fully explored for 3D models. PointNet [19] adopts jittering as a native way for point cloud augmentation, but no potential data discrepancy across domains is explicitly considered. PointDAN [21] minimizes Maximum Mean Discrepancy [9] across domains to reduce domain gaps for domain adaption. MetaSets [10] designs several geometric transformations for data augmentation based on the priors of real-scanned point clouds, aiming to bridge domain gaps. Our work is built upon [10] but instead we explore adversarial point clouds to construct intermediate domains for generalizable point cloud representation learning.

Adversarial learning for cross-domain. Szegedy *et al.* [23] first shows that DNNs for image classification is fragile to imperceptible image distortion. Adversarial learning [7, 18] aims to increase the robustness of DNNs to adversarial examples with imperceptible perturbations added to the inputs. Gradient-based adversarial perturbations are widely adopted to generate adversarial samples to reduce vulnerability of DNNs [7, 13]. In 3D vision, given benign point clouds, the works [32, 16, 33, 17] generate adversarial counterparts by perturbing individual points. Previous works in 2D vision explore to adopt adversarial learning to train models that are robust to significant perturbations, *i.e.*, OOD samples [22, 11, 27, 26, 36]. These works show that adversarial domain augmentation (ADA) can effectively improve the generalization performance and robustness of models by synthesizing virtual images according to distance metrics on domain distributions. However, few work has explored ADA for cross-domain generalizable point cloud representation learning. To our best knowledge, we are among the first to extend ADA for point cloud representation learning for DG. We refer the reader to *supplementary material* for detailed related work.

References

1. Boulch, A.: Convpoint: Continuous convolutions for point cloud processing. Computers & Graphics **88**, 24–34 (2020)
2. Chang, A.X., Funkhouser, T., Guibas, L., Hanrahan, P., Huang, Q., Li, Z., Savarese, S., Savva, M., Song, S., Su, H., et al.: Shapenet: An information-rich 3d model repository. arXiv preprint arXiv:1512.03012 (2015)
3. Eade, E.: Lie groups for computer vision. Cambridge University, Cambridge, UK, Tech. Rep **2** (2014)
4. Finn, C., Abbeel, P., Levine, S.: Model-agnostic meta-learning for fast adaptation of deep networks. In: International Conference on Machine Learning. pp. 1126–1135. PMLR (2017)
5. Gong, B., Shi, Y., Sha, F., Grauman, K.: Geodesic flow kernel for unsupervised domain adaptation. In: Proceedings of the Conference on Computer Vision and Pattern Recognition. pp. 2066–2073. IEEE (2012)
6. Gong, R., Li, W., Chen, Y., Gool, L.V.: Dlow: Domain flow for adaptation and generalization. In: Proceedings of the Conference on Computer Vision and Pattern Recognition. pp. 2477–2486 (2019)
7. Goodfellow, I.J., Shlens, J., Szegedy, C.: Explaining and harnessing adversarial examples. In: International Conference on Learning Representations (2015)
8. Gopalan, R., Li, R., Chellappa, R.: Domain adaptation for object recognition: An unsupervised approach. In: Proceedings of the International Conference on Computer Vision. pp. 999–1006. IEEE (2011)
9. Gretton, A., Borgwardt, K.M., Rasch, M.J., Schölkopf, B., Smola, A.: A kernel two-sample test. The Journal of Machine Learning Research **13**(1), 723–773 (2012)
10. Huang, C., Cao, Z., Wang, Y., Wang, J., Long, M.: Metasets: Meta-learning on point sets for generalizable representations. In: Proceedings of the Conference on Computer Vision and Pattern Recognition. pp. 8863–8872 (2021)
11. Huang, Z., Wang, H., Xing, E.P., Huang, D.: Self-challenging improves cross-domain generalization. In: European Conference on Computer Vision. pp. 124–140. Springer (2020)
12. Kanbak, C., Moosavi-Dezfooli, S.M., Frossard, P.: Geometric robustness of deep networks: analysis and improvement. In: Proceedings of the Conference on Computer Vision and Pattern Recognition. pp. 4441–4449 (2018)
13. Kim, M., Tack, J., Hwang, S.J.: Adversarial self-supervised contrastive learning. Advances in Neural Information Processing Systems **33**, 2983–2994 (2020)
14. Li, G., Muller, M., Thabet, A., Ghanem, B.: Deepgcns: Can gcns go as deep as cnns? In: Proceedings of the International Conference on Computer Vision. pp. 9267–9276 (2019)
15. Li, Y., Bu, R., Sun, M., Wu, W., Di, X., Chen, B.: Pointcnn: Convolution on x-transformed points. Advances in Neural Information Processing Systems **31** (2018)
16. Liu, D., Yu, R., Su, H.: Extending adversarial attacks and defenses to deep 3d point cloud classifiers. In: International Conference on Image Processing. pp. 2279–2283. IEEE (2019)
17. Liu, D., Yu, R., Su, H.: Adversarial shape perturbations on 3d point clouds. In: European Conference on Computer Vision. pp. 88–104. Springer (2020)
18. Madry, A., Makelov, A., Schmidt, L., Tsipras, D., Vladu, A.: Towards deep learning models resistant to adversarial attacks. In: International Conference on Learning Representations (2018)

19. Qi, C.R., Su, H., Mo, K., Guibas, L.J.: Pointnet: Deep learning on point sets for 3d classification and segmentation. In: Proceedings of the Conference on Computer Vision and Pattern Recognition (2017)
20. Qi, C.R., Yi, L., Su, H., Guibas, L.J.: Pointnet++: Deep hierarchical feature learning on point sets in a metric space. Advances in Neural Information Processing Systems (2017)
21. Qin, C., You, H., Wang, L., Kuo, C.C.J., Fu, Y.: Pointdan: A multi-scale 3d domain adaption network for point cloud representation. Advances in Neural Information Processing Systems **32** (2019)
22. Sinha, A., Namkoong, H., Duchi, J.: Certifiable distributional robustness with principled adversarial training. In: International Conference on Learning Representations (2018)
23. Szegedy, C., Zaremba, W., Sutskever, I., Bruna, J., Erhan, D., Goodfellow, I., Fergus, R.: Intriguing properties of neural networks. arXiv preprint arXiv:1312.6199 (2013)
24. Uy, M.A., Pham, Q.H., Hua, B.S., Nguyen, T., Yeung, S.K.: Revisiting point cloud classification: A new benchmark dataset and classification model on real-world data. In: Proceedings of the International Conference on Computer Vision. pp. 1588–1597 (2019)
25. Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A.N., Kaiser, Ł., Polosukhin, I.: Attention is all you need. Advances in Neural Information Processing Systems **30** (2017)
26. Volpi, R., Murino, V.: Addressing model vulnerability to distributional shifts over image transformation sets. In: Proceedings of the International Conference on Computer Vision. pp. 7980–7989 (2019)
27. Volpi, R., Namkoong, H., Sener, O., Duchi, J.C., Murino, V., Savarese, S.: Generalizing to unseen domains via adversarial data augmentation. Advances in Neural Information Processing Systems **31** (2018)
28. Wang, Y., Sun, Y., Liu, Z., Sarma, S.E., Bronstein, M.M., Solomon, J.M.: Dynamic graph cnn for learning on point clouds. Transactions On Graphics **38**(5), 1–12 (2019)
29. Wu, W., Qi, Z., Fuxin, L.: Pointconv: Deep convolutional networks on 3d point clouds. In: Proceedings of the Conference on Computer Vision and Pattern Recognition. pp. 9621–9630 (2019)
30. Wu, Z., Song, S., Khosla, A., Yu, F., Zhang, L., Tang, X., Xiao, J.: 3d shapenets: A deep representation for volumetric shapes. In: Proceedings of the Conference on Computer Vision and Pattern Recognition. pp. 1912–1920 (2015)
31. Wu, Z., Pan, S., Chen, F., Long, G., Zhang, C., Philip, S.Y.: A comprehensive survey on graph neural networks. Transactions on Neural Networks and Learning Systems **32**(1), 4–24 (2020)
32. Xiang, C., Qi, C.R., Li, B.: Generating 3d adversarial point clouds. In: Proceedings of the Conference on Computer Vision and Pattern Recognition. pp. 9136–9144 (2019)
33. Yang, J., Zhang, Q., Fang, R., Ni, B., Liu, J., Tian, Q.: Adversarial attack and defense on point sets. arXiv preprint arXiv:1902.10899 (2019)
34. Zhang, L., Wang, S., Huang, G.B., Zuo, W., Yang, J., Zhang, D.: Manifold criterion guided transfer learning via intermediate domain generation. Transactions on Neural Networks and Learning Systems **30**(12), 3759–3773 (2019)
35. Zhao, H., Jiang, L., Jia, J., Torr, P.H., Koltun, V.: Point transformer. In: Proceedings of the International Conference on Computer Vision. pp. 16259–16268 (2021)

36. Zhao, L., Liu, T., Peng, X., Metaxas, D.: Maximum-entropy adversarial data augmentation for improved generalization and robustness. Advances in Neural Information Processing Systems **33**, 14435–14447 (2020)