Manifold Adversarial Learning for Cross-domain 3D Shape Representation

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

Abstract. On a variety of 3D vision tasks, deep neural networks (DNNs) for point clouds have outperformed the conventional non-learning-based methods. However, generalization to out-of-distribution 3D point clouds remains challenging for DNNs. As annotating large-scale point clouds is prohibitively expensive or even impossible, strategies for generalizing DNN models to unseen domains of point clouds without access to those domains during training are urgently needed but have yet to be substantially investigated. In this paper, we design an adversarial learning scheme to learn point cloud representation on a seen source domain and then generalize the learned knowledge to an unseen target domain. Specifically, we unify several geometric transformations into a manifold-based framework under which a distance between transformations is well-defined. Measured by the distance, adversarial samples are mined to form *intermediate domains* and retained in an adaptive replay-based memory. We further provide theoretical justification for the intermediate domains to reduce the generalization error of the DNN models. Experimental results on synthetic-to-real datasets illustrate that our method outperforms existing 3D deep learning models for domain generalization.

Keywords: 3D point cloud, domain generalization, adversarial learning, manifold and memory

1 Introduction

If a 3D point cloud classifier was trained on intact point clouds, would it work on partial point clouds? What if a neural network trained on point clouds uniformly sampled from clean CAD models is tested on real-scanned point clouds containing noise? Is it possible to deploy a classification model trained on point clouds with a certain prior under a wild condition where all point clouds are randomly collected from the Internet? Answers to these questions heavily depend on the capability of the classification models to deal with the *domain shift* problem, which refers to the distribution shift/discrepancy between the samples from training (source) domain and those from testing (target) domain [3, 24, 47].

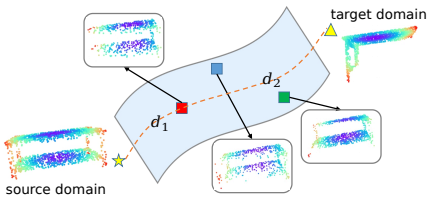

Fig. 1. Three intermediate domains consist of adversarial samples transformed from source point clouds. The distances between the transformed point cloud to the source and target are d_1 and d_2.

Although DNNs have been successful in various applications and separately obtained state-of-the-art classification results in both synthetic and real-scanned point clouds datasets [25, 26, 21, 5], studies in [27, 16] suggest that deep learning models' performance degrades significantly on out-of-distribution (OOD) datasets. Most statistical learning models, including DNNs, strongly rely on an over-simplified assumption, *i.e.*, the source and target samples are drawn from independent and identically (*i.i.d.*) distributions, while ignoring OOD scenarios commonly encountered in real world. A straight-forward solution to deal with domain shift is to collect samples from the target domain to fine-tune a source-domain-trained model. Such a schema is denoted as *domain adaptation* (DA) [28] which is infeasible when target samples are inaccessible or even unknown in advance. To resolve the domain shift and the absence of target data simultaneously, *domain generalization* (DG) [4] is introduced. The goal of DG is to train a DNN model on a single or multiple source domains such that the trained DNN can still perform well on a previously unseen OOD target domain. Despite that DG has received increasing attention in 2D vision tasks [9, 45, 30], few literature [16] focuses on DG (or DA [27]) in 3D vision. The recent work MetaSets [16] aims to train a classification model on source (synthetic) point clouds such that the trained model can also performance well on target (real-scanned) point clouds which are inaccessible during training. By applying a collection of specifically designed geometric transformations to source point clouds, the transformed point clouds imitate the real-world scenarios of occlusions, missing parts, and variations in scanning density, thus expanding the source domain towards the target domain.

In this work, we explore adversarial training [14, 31], whose main goal in previous literature is to increase the robustness of neural network models against fluctuations in the input. Distinct to imperceptible attacks utilized in conventional adversarial training, we instead aim to train DNN models which are robust to OOD samples which bridge the distribution gap between the source and target domains. In other word, we generate "fictitious" yet "challenging" point clouds in an adversarial way to mimic virtual *intermediate domains* as an expansion of the source domain, driving the model to learn domain-invariant features to improve its generalization performance, as illustrated in Fig. 1 conceptually. To utilize the adversarial samples in intermediate domains to the maximum extent, we design an adaptive replay-based memory mechanism to select and retain some more effective adversarial samples, *i.e.*, the ones pushing the intermediate domains *far* away from the source domain and moving *close* towards the target

domain. Despite that the target domain is inaccessible during training and thus we cannot directly measure the distance of an adversarial sample to the target domain, we assume that the adversarial samples which are far from the source domain could augment the diversity of the source domain, consequently, increasing the overlap shared among the source and the target domains, and potentially increasing the chance (likelihood) to close the gap between these two domains. A technical barrier is how to define the distance between domains (induced from the distance between samples in these domains) as measurement of *farness* and *closeness*. To circumvent this barrier, we unify the geometric transformations proposed in [16] in a manifold-based framework in which each transformation and the corresponding transformed point cloud are regard as points on the manifolds and the distance between points (on the manifolds) are well-defined.

Our contributions are: 1). We propose adversarial training to generate adversarial point clouds to form intermediate domains for tackling the DG problem in 3D vision; 2). We introduce a manifold-based framework to unify different geometric transformations and define distance between transformations to facilitate the construction of intermediate domains; 3). We design an adaptive memory to fully utilize the adversarial samples in intermediate domains for domain-invariant 3D point cloud feature learning. We validate our method on two Sim-to-Real benchmarks [16] and observe that our proposed method outperforms previous approaches for point clouds representation learning under DG settings.

2 Related Work

Cross-domain representation learning (for both DA and DG) has been extensively studied in 2D visionll tasks [15, 12, 44, 13], but has not yet been fully explored for 3D models. MetaSets [16] 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 [16] but instead we explore adversarial point clouds to construct intermediate domains for generalizable point cloud representation learning. **Adversarial learning** [14, 23] aims to increase the robustness of DNNs to adversarial examples with imperceptible perturbations added to the inputs. Previous works in 2D vision explore to adopt adversarial learning to train models that are robust to significant perturbations, *i.e.*, OOD samples [31, 17, 35, 34, 46]. These works show that adversarial domain augmentation (ADA) can effectively improve the generalization performance and robustness of models 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.

3 Method

We first give a brief introduction of some mathematical tools based which we formulate our manifold-based framework to measure point cloud transformations in

Sec. 3.1. Then, we unify different types of transformations under this framework in Sec. 3.2. Next, we describe our adversarial learning and adaptive memory for intermediate domains in 3.4. Lastly, we depict the learning scheme to train our DNN in Sec. 3.5. A schematic illustration of our method is shown in Fig. 2.

3.1 Manifolds of transformation and point cloud

Let $\mathcal{T}$ be a Lie group consisting of geometric transformations and $t \in \mathcal{T}$ be a specific transformation. The dimensionality of $\mathcal{T}$ equals to the number of free parameters of t. For instance, $\mathcal{T}$ can be 3D rotation group SO(3) and $t \in \text{SO}(3) : \mathbb{R}^3 \to \mathbb{R}^3$, parameterized by a vector $\theta_t \in \Theta \subset \mathbb{R}^3$ representing three rotation angles. We interpret t as a mapping from one point cloud to another. More specifically, we define a point cloud P with N points as a square integrable function $P = \sum_i^N p_i \delta_{p_i} : \mathbb{R}^{3\times N} \to \mathbb{R}^{3\times N}$ where δ_p denotes a Dirac delta function placed at a location $p = (x, y, z) \in \mathbb{R}^3$. The action of t on P is denoted as P_t, which can be regarded as a function that maps elements in the Lie group $\mathcal{T}$ to elements in a space of the transformed point clouds from P, denoted as $P_t : \mathcal{T} \to \mathbb{L}^2$, where $\mathbb{L}^2$ is the space of square integrable functions.

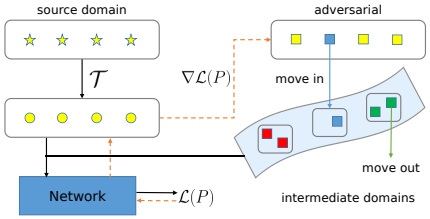

Fig. 2. Point clouds from a source domain are transformed by $\mathcal{T}$ and fed into to a network. The gradient of loss $\nabla\mathcal{L}(P)$ are used to generated adversarial samples on a manifold. The adversarial samples are retained in a memory as intermediate domains to close the source and target domain gap.

Given a Lie group $\mathcal{T}$, the function $d(t_1, t_2) : \mathcal{T} \times \mathcal{T} \to \mathbb{R}$ defines a metric to measure the distance between the two transformations t_1 and t_2. A native option is to instantiate $d(t_1, t_2) = \|t_1 - t_2\|_{L^2}$. However, this metric is inaccurate as it fails to consider the point cloud on which these transformations act. For instance, $d(\vec{0}_{\times 3}, [0, 0, 2\pi]) = 2\pi$, however, rotating any point cloud by 2π along any axis leaving the point cloud unchanged. Another instantiation $d(t_1, t_2) = \|P_{t_1} - P_{t_2}\|_{L^2}$ considers both the transformations and the point cloud, but it requires per-point correspondences to make the minus operation well-defined.

Geodesic distance, the length of the shortest path between $t_1, t_2 \in \mathcal{T}$, is a metric to measure the distance between the two transformations. This metric is valid only if a Riemannian metric is defined for $\mathcal{T}$. Inspired by [36, 18, 19], we formulate such a Riemannian metric by mapping $\mathcal{T}$ to the set of transformed point clouds of P: $\mathcal{M}(P) = \{P_t : t \in \mathcal{T}\}$. This set forms a manifold and we name it *Point Cloud Appearance Manifold* (PCAM). The Riemannian metric on $\mathcal{T}$ can be chosen such that the length of a path on $\mathcal{T}$, $\gamma(t) : [0, 1] \to \mathcal{T}$, equals to the length of the mapped path on $\mathcal{M}(P)$, $P_\gamma(t) : [0, 1] \to \mathcal{M}(P)$. The

geodesic distance between $t_1, t_2 \in \mathcal{T}$ thus equals to the geodesic distance between $P_{t_1}, P_{t_2} \in \mathcal{M}(P)$. Formally, the distance between $t_1, t_2 \in \mathcal{T}$ can be defined as:

$$d_P(t_1, t_2) = \min_{\gamma:[0,1]\to\mathcal{M}(P)} L(\gamma) \quad \text{s.t.} \quad \gamma(0) = P_{t_1}, \gamma(1) = P_{t_2} \ , \tag{1}$$

where $L(\gamma)$ is the length of the path γ. This metric depends on both the transformed point cloud P and the characteristics of the transformations t_1 and t_2.

By varying the parameter θ_t of a transformation t that controls the appearance of each point cloud, *e.g.*, location and orientation, we can get different point clouds transformed from the original one. In DG, we assume point clouds in the target domain $\mathbb{T}$ are transformed from the corresponding ones in the source domain $\mathbb{S}$ by a collection of unknown transformations $\mathcal{T}_{\mathbb{S}\to\mathbb{T}}$. (Notice that $\mathcal{T}_{\mathbb{S}\to\mathbb{T}}$ is not exactly the same as SO(3), but can be parameterized using SO(3) as formulated in Sec. 3.2.) Given two point clouds P in $\mathbb{S}$ and P' in $\mathbb{T}$, satisfying $P' = P_t$ where $t \in \mathcal{T}_{\mathbb{S}\to\mathbb{T}}$ is unknown, we define the distance of t relative to $\mathbb{S}$ as $d_P(e, t)$ where e is an identity transformation that keeps any point cloud unchanged, *i.e.*, $P_e = P$. The larger $d_P(e, t)$, the further distance of t relative to $\mathbb{S}$.

3.2 Geometric transformation unification

The distribution shift between source and target domains is usually caused by (unknown) geometric transformations. As indicated in [40, 16], occlusions, density changes, and scanning noises mainly contribute to the point cloud geometric variations. The work MetaSets [16] proposes three types of transformations to mimic these geometric variations in real-scanned point clouds. Similarly, we train a DNN model on a collection of transformed point clouds that covers a wide range of feasible variations across the source and the target domains to enhance the DNN model's generalization capability. Distinct from [16] where each type of transformation is parameterized by different transformation-specific hyper-parameters, we unify all three transformations by parameterizing them using a direct product of SO(3) and $\mathbb{R}$, *i.e.*, $\mathrm{SO}(3) \times \mathbb{R}$ which is still a Lie group. Utilizing the tools defined in Sec. 3.1, we can qualitatively measure the distance between transformations relative to the source domain, and thus mine effective adversarial samples based on the distance as described in Sec. 3.3 and 3.4. For the integrity of the paper, we cite and briefly review the three transformations formulated in [16] and refer the reader to [16] for detailed descriptions.

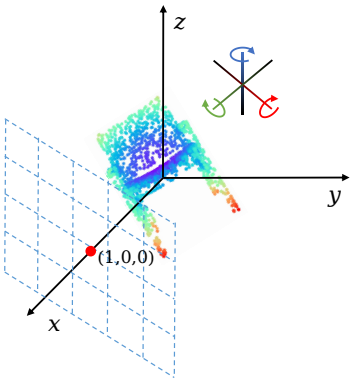

Fig. 3. Illustration of the unification of all transformations using a direct product of SO(3) and $\mathbb{R}$.

Non-uniform density (p_1, g)**.** First, randomly select a point p_1 from a unit sphere outside point cloud P and calculate its distances to each point in P. Then, the distances are normalized to $[0, 1]$ and multiplied by a multiplier $g > 1$. The resultant values are used as dropping rates of each point in P and discard points according to the dropping rates.
Dropping $(p_2, x\%)$**.** First, randomly select a point p_2 in point cloud P and then drop the nearest $x\%$ points in P. Broken or missing parts in real-world scanned objects can be simulated by this type of transformation.
Self-occlusion $(\vec{v}, W)$**.** First, a plane outside point cloud P is chosen and all points in P are projected onto the plane along its normal vector $\vec{v}$. The minimum distance between each candidate plane and P is the same for all. Then, equal-sized grids are drawn on the chosen plane with the size W. Only the nearest points to the chosen plane along the normal vector are retained in each grid.

For the three types of geometric transformations described above, the geometric shifts of self-occlusions, density changes, and missing assemblies are simulated and controlled by hyper-parameters $(p_1, g, p_2, x\%, \vec{v}, W)$. Distinct from [16] which samples discrete values of these hyper-parameters randomly and individually, we unify all the three transformations using a direct product of SO(3) and $\mathbb{R}$, and impose a manifold structure on the set of transformations. Specifically, as illustrated in Fig. 3, we centerize point cloud P and normalize it within a unit sphere. For ***non-uniform density*** (s, r), we fix p_1 as an anchor point $p_1 = (1, 0, 0)$ and rotate P by $s \in \mathrm{SO}(3)$. Then, we calculate the distances between the anchor point to each point in P. The distances are normalized to $[0, 1]$ and multiplied by a multiplier $r > 1$, resulting the dropping rates of each point in P. For ***dropping*** $(s, r\%)$, we fix an anchor vector $\vec{e} = (1, 0, 0)$ and rotate P by $s \in \mathrm{SO}(3)$. Then, we choose the point $p_{\vec{e}}$ in P with the minimal angle with the anchor vector and drop the nearest $r\%$ points around $p_{\vec{e}}$ in P. For ***self-occlusion*** (s, r), we first fix a plane parallel to yz-plane with the coordinate $x > 1$. We do not specify the concrete plane's x coordinate only if $x > 1$ as the projection is orthogonal. Then, we rotate P by $s \in \mathrm{SO}(3)$ and only keep the point with minimal distance to the plane within each grid with the grid size of r along its normal vector. In sum, we unify all the three transformations parameterized by a direct product of SO(3) and $\mathbb{R}$ as $\mathcal{T} = \{(s, r) \mid s \in \mathrm{SO}(3), r \in \mathbb{R}\}$, which achieves the same transformation effect as in [16]. We overload the notations in Sec. 3.1 and denote the set of unified transformations as $\mathcal{T}$ in which $\theta_t = [\theta_s : r] \in \Theta \subset \mathbb{R}^4$ for $t \in \mathcal{T}$ and the symbol : denotes concatenation.

3.3 Adversarial point cloud generation

We first review the generic additive adversarial point cloud generation under supervised settings. Let P be a training sample with N points and y be the corresponding label. A supervised learning model is denoted as $f_w : P \mapsto y$ where w parameterizes the model. An *adversarial sample* P^a is the worst-case example by adding perturbation to each point in P which maximizes the loss of the given model f_w [20, 42, 39]:

$$P^a = P + \alpha \texttt{sign}(\nabla_P \mathcal{L}(P, y; w)) \text{ s.t. } \|p_i - p_i'\|_{L^2} \leq \epsilon \ \forall i \in \{1, 2, \cdots, N\} \ , \quad (2)$$

where $\mathcal{L}$ is the supervised loss (*e.g.*, cross-entropy) and the condition term prevents the adversarial sample P^a from significantly deteriorating the original P.

However, the generic additive adversarial samples generated using Eq. 2 is purely driven by loss without considering the transformations towards shrinking the domain gap, thus not applicable to DG. We propose to search for the adversarial samples of a given transformed point cloud P_t maximizing the loss while still staying on PCAM. The searching process consists of two steps: 1) choosing the searching movement direction, and 2) mapping this movement back onto PCAM [19, 1, 2]. The idea is illustrated in Fig. 4 conceptually. Mathematically, let $\mathcal{L}(P_t)$ represent the loss evaluated at P_t of the model f_w, and we choose the direction maximizing the loss, *i.e.*, the direction of the gradient $\nabla_{P_t}\mathcal{L}(P_t)$. However, as we need to stay on the manifold $\mathcal{M}(P)$, we project the gradient $\nabla_{P_t}\mathcal{L}(P_t)$ onto the tangent space at P_t and such a tangent space is denoted as $T_{P_t}\mathcal{M}$ [1, 2]. The projection of $\nabla_{P_t}\mathcal{L}(P_t)$ onto $T_{P_t}\mathcal{M}$ is evaluated as [19, 6]:

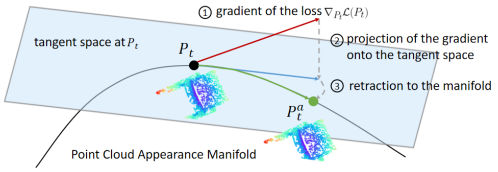

Fig. 4. Gradient computed at a given point P_t on PCAM is first projected to the tangent space. This projected gradient is then retracted to a point on the manifold, resulting in an adversarial sample P_t^a.

$$g = \mathcal{J}_{P_t}^{+}\nabla_{P_t}\mathcal{L}(P_t) \approx \arg\min_{x}\|\mathcal{J}_{P_t}x - \nabla_{P_t}\mathcal{L}(P_t)\| \in \mathbb{R}^4 \ , \tag{3}$$

where $\mathcal{J}_{P_t} \in \mathbb{R}^{(N\times 3)\times 4}$ is the Jacobian matrix of P_t (described below) whose columns form a basis of the tangent space at P_t, and $\mathcal{J}_{P_t}^{+}$ is the pseudo-inverse of $\mathcal{J}_{P_t}$. Due to the complexity of computing pseudo-inverse, the projection $g \in T_{P_t}\mathcal{M}$ can be approximated using the least squares method [6] as in Eq. 3.

To numerically define Jacobian matrix $\mathcal{J}_P$ of a point cloud P in Eq. 3, we first define the *difference field* of P relative to another point cloud P' as:

$$\Delta P_{\to P'} \triangleq \{\Delta p_{\to P'} \mid p \in P\} = \{\arg\min_{p'}\|p - p'\|_{L^2} - p \mid p \in P, p' \in P'\} \ . \tag{4}$$

That is, for each point $p \in P$, we find the nearest point (under the L^2 metric in Euclidean space) $p' \in P'$ and then use the *difference vector* $p' - p$ to measure the *change* of p from P to P'. Gathering all difference vectors of each point in P, we form the difference field of P denoting the changes from P to P'. Equipped with the difference field defined in Eq. 4, we can define the Jacobian matrix $\mathcal{J}_P$. For a point cloud P_t, we add a small $\Delta\theta$ to the parameters θ_t of the transformation t, and then the (i, j)-th entry of the Jacobian $\mathcal{J}_{P_t}$ is defined as:

$$\mathcal{J}_{P_t}[i, j] = \frac{\Delta p_{i\to P_t'}}{\Delta\theta[j]} \ , \tag{5}$$

where $p_i \in P_t$, P'_t is the point cloud transformed from P by $t \circ \Delta t$ (*i.e.*, changing the parameters from θ_t to $\theta_t + \Delta\theta$), and $\Delta\theta[j]$ is the j-th entry of $\Delta\theta$.

After generating the searching direction g, we update the parameter θ_t of the current transformation t to generate an adversarial transformation t^a parameterized by θ_t^a using gradient ascent:

$$\theta_t^a = \theta_t + \alpha \cdot g \ , \tag{6}$$

where $\alpha \in (0, +\infty) \subset \mathbb{R}$ is ascent rate. Once computing the adversarial transformation parameter θ_t^a, the next step is to map θ_t^a back onto $\mathcal{M}(P)$ using *retraction* [19, 1, 2], an operation of mapping a vector in a tangent space of a manifold back to the manifold. Here we focus on recover $s^a \in \mathrm{SO}(3)$ from $\theta_s^a = \{\theta_t^a[j] \mid j = 0, 1, 2\} \in \mathbb{R}^3$ and leave $r^a = \theta_t^a[3] \in \mathbb{R}$ unchanged.

Let G_i be the basis of the tangent space $T_e\mathrm{SO}(3)$ at the identity element e of the 3D rotation group $\mathrm{SO}(3)$ (see *supplementary materials*), the steps of retraction are summarized as: 1) mapping θ_s^a to a tangent vector in $T_e\mathrm{SO}(3)$ using G_i; 2) mapping the resultant tangent vector in $T_e\mathrm{SO}(3)$ to a rotation $s^a \in \mathrm{SO}(3)$, and 3) mapping the resultant transformation $t^a = (s^a, r^a) \in \mathcal{T}$ back to the manifold $\mathcal{M}(P)$. Formally, retraction $R_{P_t} : T_{P_t}\mathcal{M} \to \mathcal{M}(P)$ at $P_t \in \mathcal{M}(P)$ is defined as:

$$R_{P_t}(\alpha \cdot g) = P_{t^a}, \quad \text{s.t. } t^a = (s^a, r^a) \text{ and } s^a = \exp(\sum_{j=0}^{2} \theta_s^a[j] G_j) \ , \tag{7}$$

where $\theta_s^a[j]$ is the j-th entry of θ_s^a, also the coefficient w.r.t. the basis G_j. Thus, the adversarial sample from P_t can be defined as $P_t^a = R_P(t^a) = P_{t^a}$ as in Eq. 7.

3.4 Adaptive memory for intermediate domains

We describe an adaptive memory mechanism for determining which adversarial samples to be retained in an external memory [38] as intermediate domains. Adversarial samples from different types of transformations are grouped into different intermediate domains. This mechanism jointly considers the transformation distances relative to the source domain as shown in Fig. 1 and the intermediate domain sizes. We first estimate the probability of a current adversarial sample to move in memory and then choose a sample in memory to move it out simultaneously. Let $\boldsymbol{M}_i$ denotes the i-th intermediate domains formed by the adversarial samples transformed by the i-th type of transformation and retained in memory $\boldsymbol{M}$;

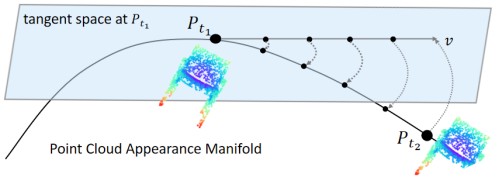

Fig. 5. Illustration of computing geodesic distance between $P_1, P_2 \in \mathcal{M}(P)$ using direct path with $N = 5$ segments.

$n_i = |\boldsymbol{M}_i|$ denotes the number of samples in domain $\boldsymbol{M}_i$; and $n = |\boldsymbol{M}|$ denotes the total number of samples in memory, satisfying $n = \sum_{i=1}^{3} n_i$.

Probability of new adversarial sample moving into memory. When a new adversarial sample $P_{t_i}^a$ is generated through transformation t_i, the chance of $P_{t_i}^a$ being moved into memory is estimated, with the central principle being the further distance from $P_{t_i}^a$ to the source domain, the higher probability of $P_{t_i}^a$ being retained. The probability is positively correlated to the geodesic distance $d_P(e, t_i)$. Furthermore, to avoid imbalance of samples among intermediate domains, we enforce the move-in probability to be inversely proportional to n_i.

Algorithm 1 A sketch of training scheme

Input: $\mathbb{S} = \{D_{\mathbb{S}}^{tr}, D_{\mathbb{S}}^{val}\}$, $\mathcal{T}$, $\boldsymbol{M}$, f_w, batch size B

while $\mathcal{L}^{val} < \epsilon$ **do**
  ▷ Training phase
  Sample $\{P_i\}_{i=1}^B$ from $D_{\mathbb{S}}^{tr}$, $\{t_i\}_{i=1}^B$ from $\mathcal{T}$
  Apply $\{t_i\}_{i=1}^B$ to $\{P_i\}_{i=1}^B$ to get $\{P_{t_i}\}_{i=1}^B$
  Sample $\{P_i^a\}_{i=1}^B$ from $\boldsymbol{M}$
  Compute $\mathcal{L}^{tr}(P) = f_w(\{P_i\}_{i=1}^B \cup \{P_i^a\}_{i=1}^B)$
  Compute projected gradient using Eq. 3
  Generate adversarial $\{P_i^a\}_{i=1}^M$ using Eq. 7
  Compute $d_P(e, t_i)$ using Eq. 13
  Compute move-in/out prob. as in Sec. 3.4
  Add some adversarial $\{P_i^a\}_{i=1}^m$ into $\boldsymbol{M}$
  ▷ Validation phase
  Sample $\{P_i\}_{i=1}^B$ from $D_{\mathbb{S}}^{val}$, $\{t_i\}_{i=1}^B$ from $\mathcal{T}$
  Compute $\mathcal{L}^{val}(P) = f_w(\{P_i\}_{i=1}^B)$
end while

Probability of existing adversarial sample moving out of memory. To move an existing adversarial sample out of memory, we perform a hierarchical sampling as proposed in [38] by first selecting an intermediate domain based on a score and then randomly move out a sample in the selected domain. The score for each domain is negatively correlated to the average geodesic distance of all samples in the domain. For the same reason as above, we enforce the score to be proportional to n_i. Formally, the score for the i-th intermediate domain is:

$$S_i = \frac{\exp(-\frac{n_i}{n}\bar{d}_i)}{\sum_j \exp(-\frac{n_j}{n}\bar{d}_j)}, \quad \text{s.t. } \bar{d}_i = \frac{1}{n_i}\sum_{k=1}^{n_i} d_P(e, t_k) \ , \tag{8}$$

and then we use S_i as the probability to sample intermediate domain $\boldsymbol{M}_i$ for moving out adversarial samples.

Theoretical justification. Given a source domain $\mathbb{S}$ and a target domain $\mathbb{T}$, the error of a classifier h on the target domain $\epsilon_{\mathbb{T}}(h)$ can be bounded by the sum of the source domain error $\epsilon_{\mathbb{S}}(h)$, the divergence between the distributions $\mathcal{D}_{\mathbb{S}}$ and $\mathcal{D}_{\mathbb{T}}$, and a constant C which is independent of h [3, 27]:

$$\epsilon_{\mathbb{T}}(h) < \epsilon_{\mathbb{S}}(h) + d_1(\mathcal{D}_{\mathbb{S}}, \mathcal{D}_{\mathbb{T}}) + C \ , \tag{9}$$

where $d_1(\mathcal{D}_{\mathbb{S}}, \mathcal{D}_{\mathbb{T}})$ is the total variation divergence (or L^1 distance when each set is countable) for distributions. In previous literature [3, 27, 8, 29], $\mathcal{H}\Delta\mathcal{H}$-distance $d_{\mathcal{H}\Delta\mathcal{H}}(\cdot, \cdot)$, an upper bound of $d_1(\cdot, \cdot)$, is adopted in place of the total variation

divergence in Eq. 9 for analysis. However, as indicated in [33], there exists a connection between the total variation divergence and transportation theory:

$$d_1(\mathcal{D}_{\mathbb{S}}, \mathcal{D}_{\mathbb{T}}) = \frac{1}{2}\|\mathcal{D}_{\mathbb{S}} - \mathcal{D}_{\mathbb{T}}\|_{L^1} = \inf_{\pi} \mathbb{E}_{\pi}[c(x, y)] \ , \tag{10}$$

where $x \sim \mathcal{D}_{\mathbb{S}}$, $y \sim \mathcal{D}_{\mathbb{T}}$, $c(x, y)$ is a cost function, and the expectation is taken w.r.t. the probability measure π on the space where (x, y) lives. In transportation theory, the cost function $c(x, y)$ is proportional to the distance between x and y. In our setting, we choose the cost function to be the geodesic distance between two point clouds, *i.e.*, $c(P_{t_1}, P_{t_2}) = d_P(t_1, t_2)$. Denoting intermediate domains as $\mathbb{M}$, to minimize the right-hand side of Eq. 9, we have:

$$\begin{aligned}
&\min d_1(\mathcal{D}_{\mathbb{S}}, \mathcal{D}_{\mathbb{T}}) \Rightarrow \min d_1(\mathcal{D}_{\mathbb{S}\cup\mathbb{M}}, \mathcal{D}_{\mathbb{T}}) && \triangleright\ \mathbb{M} \text{ is an expansion of } \mathbb{S} \\
&\Rightarrow \min \inf_{\pi} \mathbb{E}_{\pi}[c(P_{t_1} \in \mathbb{S} \cup \mathbb{M}, P_{t_2} \in \mathbb{T})] && \\
&\Rightarrow \min \inf_{\pi} \mathbb{E}_{\pi}[c(P_{t_1} \in \mathbb{M}, P_{t_2} \in \mathbb{T})] && \triangleright\ \mathbb{S} \text{ is given and fixed} \\
&\Rightarrow \min \inf_{\pi} \mathbb{E}_{\pi}[d_P(t_{\mathbb{S}\to\mathbb{M}}, t_{\mathbb{S}\to\mathbb{T}})] && \triangleright\ t_1 \coloneqq t_{\mathbb{S}\to\mathbb{M}} \text{ and } t_2 \coloneqq t_{\mathbb{S}\to\mathbb{T}} \\
&\Rightarrow \min \inf_{\pi} \mathbb{E}_{\pi}[d_P(e, t_{\mathbb{S}\to\mathbb{T}}) - d_P(e, t_{\mathbb{S}\to\mathbb{M}})] && \triangleright\ d_P(e, t_{\mathbb{S}\to\mathbb{T}}) = d_P(e, t_{\mathbb{S}\to\mathbb{M}}) + d_P(t_{\mathbb{S}\to\mathbb{M}}, t_{\mathbb{S}\to\mathbb{T}}) \\
&\Rightarrow \max \inf_{\pi} \mathbb{E}_{\pi}[d_P(e, t_{\mathbb{S}\to\mathbb{M}})] && \triangleright\ t_{\mathbb{S}\to\mathbb{T}} \text{ is unknown and assumed fixed}
\end{aligned} \tag{11}$$

The fourth comment assumes $d_P(e, t_{\mathbb{S}\to\mathbb{M}}) = d_1$ and $d_P(t_{\mathbb{S}\to\mathbb{M}}, t_{\mathbb{S}\to\mathbb{T}}) = d_2$, as shown in Fig. 1. We select the adversarial sample with the maximum geodesic distance of $d_P(e, t_{\mathbb{S}\to\mathbb{M}})$ and store it into the corresponding intermediate domain.

We now describe how to calculate the geodesic distance $d_P(t_1, t_2)$. Assuming $P_{t_1}, P_{t_2} \in \mathcal{M}(P)$, we adopt the *direct path* method [19] to approximately measure the geodesic distance between these two points as follows. First, map P_{t_2} to the tangent space at P_{t_1} as vector $v = \theta_{t_2} - \theta_{t_1}$, dividing v into smaller vector segments and re-map these segments back onto the manifold. Then, $d_P(t_1, t_2)$ is computed as the sum of distances between these interval points on $\mathcal{M}(P)$. Mathematically, the direct path from P_{t_1} to P_{t_2} can be estimated using retraction as:

$$\gamma(\tau) = R_{P_{t_1}}(\tau v) = P_{t_\tau} \in \mathcal{M}(P), \quad \text{s.t. } \theta_{t_\tau} = \theta_{t_1} + \tau v \text{ and } \tau \in [0, 1] \ . \tag{12}$$

For a pre-defined number of steps N, v can be divided into segments as $\hat{v} = \frac{v}{N}$, the geodesic distance can be measured as:

$$d_P(t_1, t_2) = d(P_{t_1}, P_{t_2}) = \sum_{i=1}^{N} \|R_{P_{t_1}}(i\hat{v}) - R_{P_{t_1}}((i-1)\hat{v}))\|_d = \sum_{i=1}^{N} \|P_{t_i} - P_{t_{i-1}}\|_d \ , \tag{13}$$

where $\theta_{t_i} = \theta_{t_1} + i\hat{v}$. In our case, we estimate $d_P(t_1, t_2)$ as $d(P_{t_1}, P_{t_2})$, since from Eq. 1, the distance between two transformations on $\mathcal{T}$ is defined as the distance between their transformed point clouds on PCAM $\mathcal{M}(P)$. Using the difference vectors defined in Eq. 4, we can induce a metric on $\mathcal{M}(P)$ defined as:

$$\|P - P'\|_d = \frac{1}{N} \sum_{p \in P} \|\Delta p_{\to P'}\|_{L^2} \ , \tag{14}$$

where N is number of points in P. $\|P - P'\|_d = 0$ if and only if P and P' are the same. Plugging Eq. 14 into the right-hand side of Eq. 13, we can approximate the geodesic distance between t_1 and t_2.

3.5 Training scheme

We sketch key steps of our training scheme in Algorithm 1. We split data in the source domain into training and validation sets, and apply transformations to a batch of sampled data in each iteration. The loss is applied to generate adversarial samples as in Sec. 3.3. We compute move-in/out probabilities of the adversarial samples based on geodesic distances as in Sec. 3.4 and move some samples into and/or out of the memory. We refer the reader to *supplementary materials* for a detailed training scheme.

4 Experiment

Dataset and implementation. We evaluate our model on the two Sim-to-Real benchmarks proposed in [16] in which two synthetic datasets, ModelNet [41] and ShapeNet [7], and a real-scanned dataset ScanObjectNN [32] are calibrated to serve as three domains. We use the common categories shared across each dataset. We refer the reader to *supplementary materials* and [16] for detailed description about the Sim-to-Real benchmarks. We split each source domain into training and validation sets with a ratio of $5:1$ as in [11, 16]. We implement f_w using a meta-learning algorithm [10, 17] with first-order approximation. The parameter $\theta_s \in \mathbb{R}^3$ for $s \in \mathrm{SO}(3)$ are optimized in adversarial learning without bounds, while the parameter $r \in \mathbb{R}$ are constrained using upper and lower bounds to avoid significantly deteriorating point clouds as suggested in [16].

4.1 Visualization of adversarial point clouds

Fig. 6 shows several adversarial samples that are generated for three types of transformations in experiments. We notice that compared to the clean and intact point clouds in synthetic ModelNet40 and ShapeNet datasets, real-scanned point clouds in ScanObjectNN are partial and noisy, raising the distribution shift issue that a model trained only using naive initial data from ModelNet40 or ShapeNet cannot perform well on ScanObjectNN as the target domain. By generating adversarial samples diverting away from synthetic clean data and towards partial and noisy data, the distributions of point clouds in the source and the target domains could gradually align. For instance, the sofa in ScanObjectNN contains a missing part (a hole) on its back, the transformed sofa also produces a hole on the back, making the training data geometrically closer to the testing one.

4.2 ModelNet40 to ScanObjectNN

We select 11 shared categories across ModelNet40 and ScanObjectNN datasets. We train the networks on ModelNet40 and evaluate the performance on ScanObjectNN. Our method is compared against the following classification models:

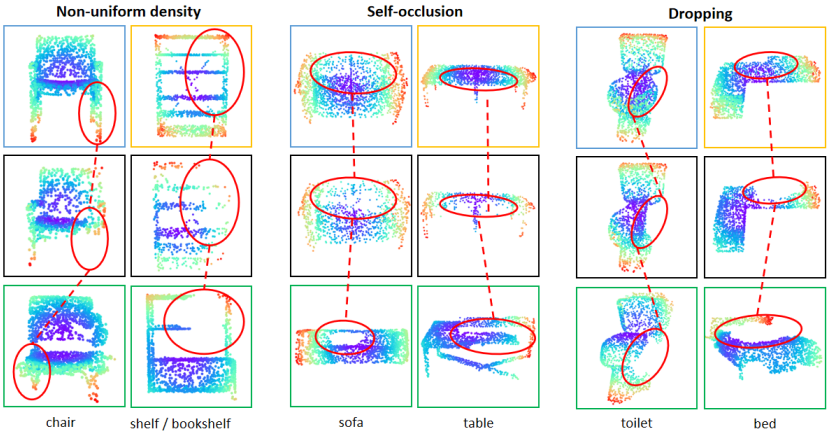

Fig. 6. Blue boxes: point clouds from ModelNet40 as the source domain. Orange boxes: point clouds from ShapeNet as the source domain. Black boxes: adversarial samples generated from the corresponding source domain point clouds (above). Green boxes: point clouds from ScanobjectNN as the target domain. The adversarially transformed point clouds form the intermediate domains closing the geometric gap between intact point clouds in the source domain and incomplete point clouds in the target domain.

PointNet [25], PointNet++ [26], DGCNN [37], ConvPoint [5], LDGCNN [43], PointCNN [22], PointDAN [27] and MetaSets [16]. Note that PointDAN is a DA method which utilizes unlabeled point clouds from the target domain during training. From Tab. 1, our approach outperforms all the compared methods for cross-domain point cloud classification in DG settings. Our superior performance over previous single-domain approaches (line $1 \sim 6$) is attributed to the intermediate domain expanding the source domain and aligning data distribution with the target domain. Compare with DA and DG approaches (line $7 \sim 9$), we still achieve optimal performance. Distinct from [16] which generates data for augmentation using random transformation parameters, our method produces adversarial transformations in a principle manner, *i.e.*, the transformed samples are more challenging w.r.t. the network and have further distances relative to the source domain. Moreover, the adaptive memory retains some adversarial samples during training to alleviate the potential *forgetting* problem in the network. We have to mention that superiority of adversarial learning could be partially countered, however, if massive amounts of data are transformed arbitrarily and encompass a broad spectrum of transformation possibilities.

4.3 ShapeNet to ScanObjectNN

Similar to Sec. 4.2, we select 9 categories shared across ShapeNet and ScanObjectNN, and train our model on ShapeNet and evaluate on ScanObjectNN. We compare our method with the same state-of-the-art approaches as forementioned and the results are listed in Tab. 2. Our method still outperforms all

Table 1. Accuracy (%) on the benchmark: ModelNet40 → ScanObjectNN.

Method	Object	Object & Background
PointNet [25]	55.90 ± 1.47	49.48 ± 2.28
PointNet++ [26]	47.30 ± 0.53	40.42 ± 1.17
ConvPoint [5]	57.40 ± 0.44	55.44 ± 0.32
DGCNN [21]	61.68 ± 1.26	57.61 ± 0.44
PointCNN [22]	50.32 ± 0.43	46.11 ± 0.43
LDGCNN [21]	62.29 ± 0.22	58.83 ± 0.43
PointDAN [27] with PointNet	63.32 ± 0.85	55.13 ± 0.97
MetaSets [16] with PointNet	68.28 ± 0.79	57.19 ± 1.23
MetaSets [16] with DGCNN	72.42 ± 0.21	65.66 ± 1.06
Ours with PointNet	69.84 ± 0.56	58.36 ± 0.94
Ours with DGCNN	**73.83 ± 0.48**	**66.71 ± 0.97**

the compared approaches, including the single-domain approaches and DA/DG approaches. More importantly, we provide a systematic way to quantitatively measure geometric transformations built on which we can explore adversarial samples and such a scheme is lacked in all previous approaches.

5 Ablation Study

We conduct ablative studies on our proposed method using PointNet as our backbone and evaluate on the benchmark from ModelNet40 to ScanObjectNN.
Effect of adaptive memory. To verify the effectiveness of the adaptive memory, we vary the total size of the memory and the results are listed in Tab. 3. With the increase of the memory size, the accuracy increases accordingly while the variance decreases.

Table 2. Accuracy (%) on the benchmark: ShapeNet → ScanObjectNN.

Method	Object	Object & Background
PointNet [25]	54.00 ± 0.32	45.50 ± 0.99
PointNet++ [26]	45.50 ± 0.64	43.25 ± 1.23
ConvPoint [5]	52.58 ± 0.58	50.67 ± 0.88
DGCNN [21]	57.42 ± 1.01	54.42 ± 0.80
PointCNN [22]	49.42 ± 0.29	43.92 ± 0.63
LDGCNN [21]	57.92 ± 0.63	52.50 ± 0.25
PointDAN [27] with PointNet	54.95 ± 0.87	43.00 ± 0.95
MetaSets [16] with PointNet	55.25 ± 0.35	49.50 ± 0.43
MetaSets [16] with DGCNN	60.92 ± 0.76	59.08 ± 1.01
Ours with PointNet	57.03 ± 0.47	51.35 ± 0.46
Ours with DGCNN	**62.21 ± 0.49**	**61.13 ± 0.86**

Table 3. Effect of the memory size on the benchmark: ModelNet40 → ScanObjectNN.

Method	Object	Object & Background
PointNet ($n = 100$)	69.22 ± 0.74	57.94 ± 1.15
PointNet ($n = 300$)	69.51 ± 0.69	58.17 ± 1.08
PointNet ($n = 500$)	69.84 ± 0.56	58.36 ± 0.94

Effect of manifold-based adversarial samples. We validate our manifold-based adversarial samples against the additive adversarial samples generated using Eq. 2 and the results are listed in Tab. 4. By varying ϵ, we control the *adversarialness*. Note that when the adversarialness increases, the performance even drops, as the per-point adversarial acts as noise and fails to bridge domain gaps as our proposed adversarial samples.

Table 4. Effect of additive adversarial on benchmark: ModelNet40 → ScanObjectNN.

Method	Ours	$\epsilon = 0.1$	$\epsilon = 0.25$	$\epsilon = 0.5$
PointNet on **Object**	69.84 ± 0.56	68.31 ± 0.46	64.29 ± 0.51	58.37 ± 0.74

6 Conclusions

This work presents a 3D point cloud representation learning method for domain generation. We construct a point cloud appearance manifold on which different types of geometric transformations can be unified and quantitatively measured using geodesic distance. The geodesic distance is then utilized to generate and select adversarial samples which are retained in an adaptive memory as intermediate domains to reduce domain shift. We evaluate our method on two Sim-to-Real benchmarks and achieve superior performance. The manifold-based framework can be extended to domain adaption by measuring the distance from the source to the target domains, which is left for future work.

Acknowledgments The authors appreciate the generous support provided by Inception Institute of Artificial Intelligence (IIAI) in the form of NYUAD Global Ph.D. Student Fellowship. This work was also partially supported by the NYUAD Center for Artificial Intelligence and Robotics (CAIR), funded by Tamkeen under the NYUAD Research Institute Award CG010.

References

1. Absil, P.A., Mahony, R., Sepulchre, R.: Optimization algorithms on matrix manifolds. Princeton University Press (2009)
2. Absil, P.A., Mahony, R., Sepulchre, R.: Optimization on manifolds: Methods and applications. In: Recent Advances in Optimization and Its Applications in Engineering, pp. 125–144. Springer (2010)
3. Ben-David, S., Blitzer, J., Crammer, K., Kulesza, A., Pereira, F., Vaughan, J.W.: A theory of learning from different domains. Machine Learning **79**(1), 151–175 (2010)
4. Blanchard, G., Lee, G., Scott, C.: Generalizing from several related classification tasks to a new unlabeled sample. Advances in Neural Information Processing Systems **24** (2011)
5. Boulch, A.: Convpoint: Continuous convolutions for point cloud processing. Computers & Graphics **88**, 24–34 (2020)
6. Buss, S.R.: Introduction to inverse kinematics with jacobian transpose, pseudoinverse and damped least squares methods. Journal of Robotics and Automation **17**(1-19), 16 (2004)
7. 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)
8. Chen, Y., Li, W., Sakaridis, C., Dai, D., Van Gool, L.: Domain adaptive faster r-cnn for object detection in the wild. In: Proceedings of the Conference on Computer Vision and Pattern Recognition. pp. 3339–3348 (2018)
9. Fan, X., Wang, Q., Ke, J., Yang, F., Gong, B., Zhou, M.: Adversarially adaptive normalization for single domain generalization. In: Proceedings of the Conference on Computer Vision and Pattern Recognition. pp. 8208–8217 (2021)
10. 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)
11. Ghifary, M., Kleijn, W.B., Zhang, M., Balduzzi, D.: Domain generalization for object recognition with multi-task autoencoders. In: Proceedings of the International Conference on Computer Vision. pp. 2551–2559 (2015)
12. 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)
13. 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)
14. Goodfellow, I.J., Shlens, J., Szegedy, C.: Explaining and harnessing adversarial examples. In: International Conference on Learning Representations (2015)
15. 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)
16. 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)
17. 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)

18. Jacques, L., De Vleeschouwer, C.: A geometrical study of matching pursuit parametrization. Transactions on Signal Processing **56**(7), 2835–2848 (2008)
19. 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)
20. Kim, M., Tack, J., Hwang, S.J.: Adversarial self-supervised contrastive learning. Advances in Neural Information Processing Systems **33**, 2983–2994 (2020)
21. 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)
22. 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)
23. 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)
24. Moreno-Torres, J.G., Raeder, T., Alaiz-Rodríguez, R., Chawla, N.V., Herrera, F.: A unifying view on dataset shift in classification. Pattern Recognition **45**(1), 521–530 (2012)
25. 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)
26. 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)
27. 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)
28. Saenko, K., Kulis, B., Fritz, M., Darrell, T.: Adapting visual category models to new domains. In: European Conference on Computer Vision. pp. 213–226. Springer (2010)
29. Saito, K., Watanabe, K., Ushiku, Y., Harada, T.: Maximum classifier discrepancy for unsupervised domain adaptation. In: Proceedings of the Conference on Computer Vision and Pattern Recognition. pp. 3723–3732 (2018)
30. Shu, Y., Cao, Z., Wang, C., Wang, J., Long, M.: Open domain generalization with domain-augmented meta-learning. In: Proceedings of the Conference on Computer Vision and Pattern Recognition. pp. 9624–9633 (2021)
31. Sinha, A., Namkoong, H., Duchi, J.: Certifiable distributional robustness with principled adversarial training. In: International Conference on Learning Representations (2018)
32. 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)
33. Villani, C.: Optimal transport: old and new, vol. 338. Springer (2009)
34. 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)
35. 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)

36. Wakin, M.B., Donoho, D.L., Choi, H., Baraniuk, R.G.: The multiscale structure of non-differentiable image manifolds. In: Optics & Photonics. vol. 5914, p. 59141B. International Society for Optics and Photonics (2005)
37. 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)
38. Wang, Z., Duan, T., Fang, L., Suo, Q., Gao, M.: Meta learning on a sequence of imbalanced domains with difficulty awareness. In: Proceedings of the International Conference on Computer Vision. pp. 8947–8957 (2021)
39. Wen, Y., Lin, J., Chen, K., Chen, C.P., Jia, K.: Geometry-aware generation of adversarial point clouds. Transactions on Pattern Analysis and Machine Intelligence (2020)
40. Wu, B., Zhou, X., Zhao, S., Yue, X., Keutzer, K.: Squeezesegv2: Improved model structure and unsupervised domain adaptation for road-object segmentation from a lidar point cloud. In: International Conference on Robotics and Automation. pp. 4376–4382. IEEE (2019)
41. 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)
42. 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)
43. Zhang, K., Hao, M., Wang, J., de Silva, C.W., Fu, C.: Linked dynamic graph cnn: Learning on point cloud via linking hierarchical features. arXiv preprint arXiv:1904.10014 (2019)
44. 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)
45. Zhang, X., Cui, P., Xu, R., Zhou, L., He, Y., Shen, Z.: Deep stable learning for out-of-distribution generalization. In: Proceedings of the Conference on Computer Vision and Pattern Recognition. pp. 5372–5382 (2021)
46. 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)
47. Zhou, K., Liu, Z., Qiao, Y., Xiang, T., Loy, C.C.: Domain generalization in vision: A survey. arXiv preprint arXiv:2103.02503 (2021)