Self-Supervised Learning for Domain Adaptation on Point-Clouds

Idan Achituve[1], Haggai Maron[2], and Gal Chechik[1,2]

[1] Bar-Ilan University, Ramat Gan, Israel
achitui@cs.biu.ac.il
[2] NVIDIA, Tel Aviv, Israel

Abstract. Self-supervised learning (SSL) allows to learn useful representations from unlabeled data and has been applied effectively for domain adaptation (DA) on images. It is still unknown if and how it can be leveraged for domain adaptation for 3D perception. Here we describe the first study of SSL for DA on point-clouds. We introduce a new pretext task, *Region Reconstruction*, motivated by the deformations encountered in sim-to-real transformation. We also demonstrate how it can be combined with a training procedure motivated by the MixUp method. Evaluations on six domain adaptations across synthetic and real furniture data, demonstrate large improvement over previous work.

Keywords: Point-clouds, Unsupervised domain adaptation, Self-supervised learning

1 Introduction

Self-supervised learning (SSL) was recently shown to be very effective for learning useful representations from unlabeled images [7, 8, 14, 26, 27] or videos [10, 25, 45, 47]. The key idea is to define an auxiliary, "pretext" task, train using supervised techniques, and then use the learned representation for the main task of interest. While SSL is often effective for images and videos, it is still not fully understood how to apply it to other types of data. Recently there have been some attempts at designing SSL pretext tasks for point-cloud data for representation learning [16, 36, 41, 57], yet this area of research is still largely unexplored. Since SSL operates on unlabeled data, it is natural to test its effectiveness for learning under unsupervised domain adaptation (DA).

DA has attracted a lot of attention in recent years, to name a few [11, 34, 43, 44]. In DA, one aims to classify data from a *Target* distribution, but the only labeled samples available are from another *Source* distribution. This problem has wide implications, including "sim-to-real" - training on simulated data where labels are abundant, and testing on real-world data. Recently, SSL was successfully utilized for learning across domains [2, 9, 31], and yielded promising results in domain adaptation for visual tasks such as object recognition and segmentation [39, 50]. While SSL has been used to adapt to new domains in images,

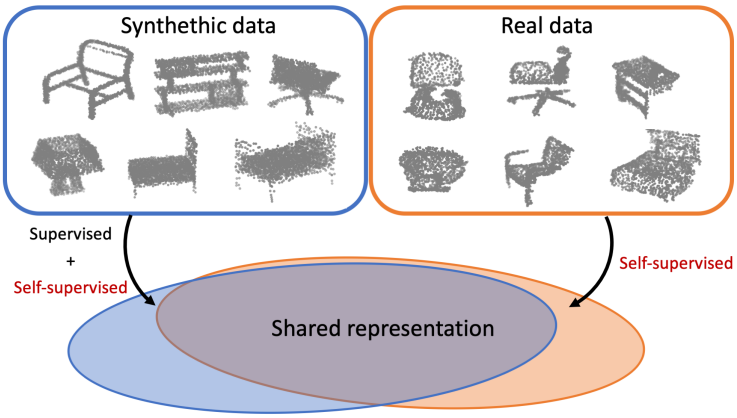


Fig. 1: We tackle the domain adaptation problem for 3D point-cloud data, with an emphasis on sim-to-real setup. Our method learns a shared point-cloud representation by leveraging the source labels as well as a novel self-supervised task on both target and source domains: reconstruction of deformed point-clouds

it is not known if and how SSL applies to DA on other data types, particularly on 3D data.

The current paper addresses the challenge of developing SSL for point-clouds in the context of DA. We describe, for the first time, an SSL approach for adapting to new point-cloud distributions. Our approach is based on a multi-task architecture with a multi-head network. One head is trained using a classification loss over the source domain while a second head is trained using a new SSL loss which can be applied on either source or target domains.

To learn a representation that captures the structure of the target domain, we develop a new pretext-task, *Region Reconstruction* (RegRec). We design it to address a common deformation that is encountered in sim-to-real point-clouds; Scanning objects in their natural environments often leads to missing object parts due to occlusion (see Figures 1, 3). The key idea behind the new pretext task is that it deforms a random region of the 3D shape in order for the network to map back those points to their place and reconstruct the missing region of the shape. Succeeding in doing so will indicate that the network learned a relationship between the missing region the rest of the shape. In this paper, we focused on the deformation of concentrating all points in a region into its center (we experimented with additional deformations as well). We find that training with RegRec improves classification accuracy in the target domain in several DA setups (mainly in those that involve the real domain).

We further present a new training procedure based on MixUp [56], called *Point-Cloud Mixup* (PCM), that is applied to source objects during training instead of the standard classification task. Standard Mixup generates new data-label pairs by applying convex combinations on images. Since point-clouds are unordered sets, applying Mixup naively probably will not generate anything

meaningful. Together with RegRec, PCM yields large improvements over the SoTA of domain adaptation in a benchmark dataset in this area [30].

This paper makes the following novel contributions. (1) This is the first paper that studies SSL for domain adaptation on point-clouds. (2) We describe RegRec, a new pretext task for point-clouds, motivated by the type of distortions encountered in sim-to-real scenario. (3) We develop a new variant of the Mixup method for point-cloud data and (4) We achieve a new SoTA performance for domain adaption on point-clouds including a large improvement over previous approaches in a sim-to-real task.

2 Related Work

Deep learning on point-clouds. Following the success of deep neural networks on images, powerful deep architectures for learning with 3D point-clouds were designed. Early methods, such as [24, 29, 49], applied volumetric convolutions to occupancy grids generated from point-clouds. These methods suffer from limited performance due to the low resolution of the discretization of 3D data. The seminal work of [28, 55] described the first models that work directly on a point-cloud representation. Following these studies, a plethora of architectures was suggested, aiming at generalizing convolutions to point-clouds [1,17,20,21,38,46]. We refer the readers to a recent survey [15] for more details.

Self-supervised learning for point-clouds. Recently, several studies suggested self-supervised tasks for learning meaningful representations of point-cloud data, mostly as a pre-training step. In [36] it is suggested to generate new point-clouds by splitting a shape to $3 \times 3 \times 3$ voxels and shuffle them. The task is to predict the voxel assignment of each point that reconstructs the original point-cloud. [41] proposed a network that predicts the next point in a space-filling sequence of points that covers a point-cloud. [57] generated pairs of half-shapes and proposed to learn a classifier to decide whether these two halves originate from the same point-cloud. [16] advocates combining three tasks: clustering, cluster classification, and point-cloud reconstruction from a perturbed input. [4] learns a point-cloud auto-encoder that also predicts pairwise relations between the points. [40] suggested learning local geometric properties by training a network to predict the point normal vector and curvature.

Domain adaptation for point-clouds. DA for point-cloud data received some attention lately. PointDAN [30] designed a dataset based on three widely-used point-cloud datasets: ShapeNet [3], ModelNet [49] and ScanNet [6]. They proposed a model that jointly aligns local and global point-cloud features. Several other studies considered domain adaptation for LiDAR data with methods that do not operate directly on the point-cloud representation [32, 35, 48]. [32] suggested a method for supervised DA from voxelized points using an object region proposal loss, point segmentation loss, and object regression loss. [35] addressed the task of vehicle detection from a bird's eye view (BEV) using a CycleGAN. [48] designed a training procedure for object segmentation of point-clouds projected onto a spherical surface.

Self-supervised learning for domain adaptation. SSL for domain adaptation is a relatively new research topic. Existing literature is mostly very recent, and is applied to images, which are fundamentally different from an unordered set of points. [13] offered to use a shared encoder for both source and target samples followed by a classification network for source samples and a reconstruction network for target samples. Recently [2,39,50] suggested using more recent SSL pretext tasks in a similar architecture. [50] suggested using SSL pretext tasks (like image rotation and patch location prediction) over a feature extractor. [39] extended the solution to a multi-task problem with several SSL pretext tasks. A central claim in their paper was that SSL tasks should be applied to both source and target data. We found that it is not mandatory to achieve good performance. [2] advocated the use of a Jigsaw puzzle [26] pretext task for domain generalization and adaptation. Our approach is similar to these approaches in the basic architectural design, yet it is different in the type of data, the pretext task and the use of a new training procedure based on Mixup. [33] addressed the problem of universal domain adaptation by learning to cluster in an unsupervised manner target data based on labeled source data. Several other studies have shown promising results in learning useful representations via SSL for cross-domain learning. [31] suggested to train a network with synthetic data using easy-to-obtain labels for synthetic images such as the surface normal, depth and instance contour. [9] offered to use SSL pre-text tasks, such as rotations, as part of their architecture for domain generalization.

Mixup for domain adaptation. Mixup is a type of training procedure suggested recently by [56]. The basic idea is to generate new training data-label pairs by convex combinations of training samples. Several studies demonstrated it's benefit for various tasks such as calibrating uncertainty [42] and domain adaptation for images [23,51,54].

3 Approach

In this section, we present the main building blocks of our approach. We first describe our general pipeline and training procedure, and then explain in detail our main contributions: the *Region Reconstruction* SSL task and the *Point-Cloud Mixup* training procedure.

3.1 Overview of Suggested Approach

We tackle unsupervised domain adaptation for point-cloud classification. Here, we observe labeled instances from a source distribution and unlabeled instances from a possibly different target distribution. Importantly, both distributions of point-clouds are of objects labeled by the same set of classes. Given instances from both distributions, the goal is to train a model that correctly classifies samples from the target domain.

We follow a common approach to tackle this learning setup, learning a shared feature encoder [53] which is trained on two tasks: (1) A supervised task on

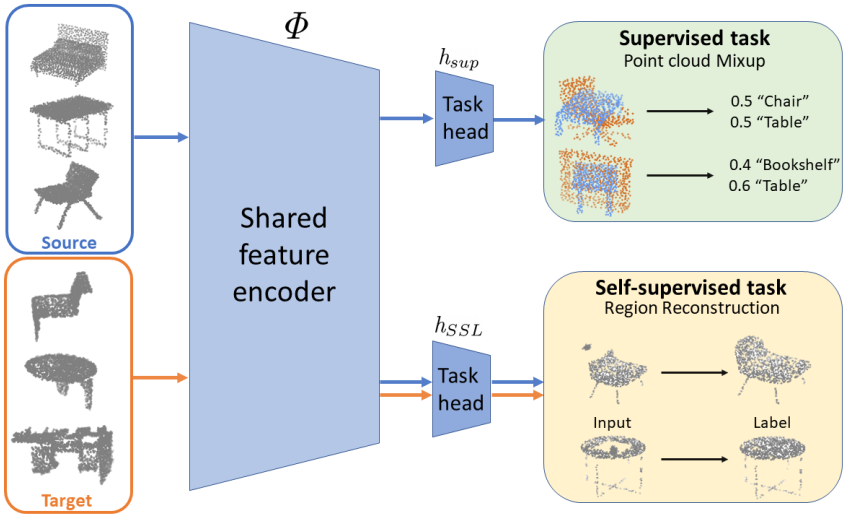


Fig. 2: Our architecture is composed of a shared feature encoder Φ, and two separate task-specific heads: One for the supervised classification task on the source domain h_{PCM}, and another for the self-supervised task on both domains h_{SSL}.

the source domain; and (2) A self-supervised task on both source and target domains (See Figure 2). To this end, we propose a new self-supervised task and a new supervised training procedure. In our self-supervised task, called *Region Reconstruction* (RegRec), we first deform a random region in an input point-cloud and then train our model to reconstruct it. Our supervised training procedure, *Point-Cloud Mixup* (PCM), is motivated by [56] and produces mixed instances of labeled input point-clouds. By jointly optimizing the encoder on both tasks, it learns a representation shared across domains that can be used for classification in the target domain. The basic pipeline of our approach is illustrated in Figure 2.

More formally, let $\mathcal{X}, \mathcal{Y}$ denote our input space and label space accordingly. Let $S \subset \mathcal{X} \times \mathcal{Y}$ represent labeled data from the source domain, and $T \subset \mathcal{X}$ represent unlabeled data from the target domain. Our training scheme has two separate data flows that are trained in an alternating fashion. Importantly, both data flows use the same feature encoder Φ (a neural network).

Supervised data flow. (Figure 2, top branch). The supervised data flow starts with sampling two labeled point-clouds $(x, y), (x', y') \in S$. These point-clouds are combined into a new labeled point-cloud $(\overline{x}, \overline{y}) \in \overline{S}$, where $\overline{S} \subset \mathcal{X} \times \mathcal{Y}$ is a set that contains all such combinations. $\overline{x}$ is then fed into the shared encoder Φ to produce a point-cloud representation $\Phi(\overline{x}) \in \mathbb{R}^d$. This representation is further processed by a fully connected sub-network (head) denoted by h_{PCM}.

The cross entropy loss L_{ce} is then applied to the output of h_{PCM} and the new label $\overline{y}$.

Self-supervised data flow. (Figure 2, bottom branch). The self-supervised data flow starts with generating a new input-label pair $(\widehat{x}, x) \in \widehat{T \cup S} \subset \mathcal{X} \times \mathcal{X}$. Here, the label x is a point-cloud in $S \cup T$, the input $\widehat{x}$ is a deformed version of x, and $\widehat{T \cup S}$ is a set that contains all such pairs. As in the supervised data flow, $\widehat{x}$ is first processed by Φ, producing a representation $\Phi(\widehat{x})$. This representation is then fed into another head, denoted h_{SSL} which is in charge of producing a reconstructed version of $\widehat{x}$. A reconstruction loss L_{SSL}, which penalizes deviations between the output $h_{\text{SSL}}(\Phi(\widehat{x}))$ and the original point-cloud x is then applied.

To summarize, the loss we use is a linear combination of a supervised loss and the SSL loss:

$$L(S, T; \Phi, h_{\text{sup}}, h_{\text{SSL}}) = L_{\text{ce}}(\overline{S}; \Phi, h_{\text{PCM}}) + \lambda L_{\text{SSL}}(\widehat{S \cup T}; \Phi, h_{\text{SSL}}) \tag{1}$$

Where λ is a parameter that controls the importance of the self-supervised term. We next explain both RegRec and PCM in detail.

3.2 The Region Reconstruction SSL Task

When designing a self-supervision task, several considerations should be taken into account. First, the task should encourage the model to capture the semantic properties of the inputs. The scale of these properties is important: a task that depends on local features may not capture the semantics, and a task that depends on full global features may be over permissive. It is in general useful to focus on meso-scale features, capturing information at the scale of "regions" or parts.

Second, for the specific case of designing SSL for DA, we want the SSL task to "bridge" the distribution gap from the *Source* to the *Target* distribution. Intuitively, it would be beneficial if the SSL deformation of target samples can imitate the same deformations that are observed from source to target because then the learned representation tends to be invariant to these gaps. We designed RegRec, our SSL task, with this intuition in mind.

The main idea of our SSL task is to reconstruct deformed input samples. (1) First, distort a mid-sized region of a point-cloud; (2) Use it as an input to the network; (3) Use the original point-cloud as the label and (4) train the encoder to produce features that can reconstruct the distorted point-cloud.

Here, we apply a simple variant of this approach and demonstrate that it works well in practice. To generate distorted point-clouds, we first split the input space (say, the box which bounds all points in the cloud) to $k \times k \times k$ equally-sized voxels (we use $k = 3$ throughout the paper, which gives us meso-scale voxels). Given a point-cloud $x \in \mathbb{R}^{n \times 3}$ with n points, we pick one voxel v uniformly at random and replace all the points in v with points sampled from an isotropic Gaussian distribution centered at the center of v with a small standard deviation. This process yields a distorted point-cloud $\hat{x} \in \mathbb{R}^{n \times 3}$. See Figure 2 (yellow box) for examples of input-output pairs.

As stated earlier, we would like the encoder to produce features that can reconstruct the distorted voxel. Therefore, We chose the loss function L_{SSL} to be the Chamfer distance between the set of points in x that falls into the chosen voxel $v \subset \mathbb{R}^3$ and their corresponding outputs. More explicitly, if $I \subset \{1, \ldots, n\}$ represents the indices of the points in $x \cap v$ the loss takes the following form:

$$L_{SSL}(\widehat{S \cup T}; \Phi, h_{\text{SSL}}) = \sum_{(\widehat{x}, x) \in \widehat{S \cup T}} d_{\text{Chamfer}}\left(\{x_i\}_{i \in I}, \{h_{\text{SSL}}(\Phi(\widehat{x}))_i\}_{i \in I}\right) \tag{2}$$

where x_i is the i-th point in the point-cloud x, and

$$d_{\text{Chamfer}}(A, B) = \sum_{a \in A} \min_{b \in B} \|a - b\|_2^2 + \sum_{b \in B} \min_{a \in A} \|b - a\|_2^2 \tag{3}$$

is the symmetric Chamfer distance between $A, B \subset \mathbb{R}^3$. Since the Chamfer distance is computed only on within-region points, it does not burden the computation.

3.3 Point-Cloud Mixup

In this subsection, we introduce a new training procedure that is motivated by the recent Mixup Method [56]. Mixup is designed based on the Vicinal Risk Minimization principle as opposed to Empirical Risk Minimization, and can also be viewed as an extension of data augmentation that involves both the input samples and their labels. Given two images and their "one-hot" labels $(x, y), (x', y')$, the Mixup method generates a new labeled sample as a convex combination of the inputs $(\gamma x + (1-\gamma)x', \gamma y + (1-\gamma)y')$, where γ is sampled from a $Beta$ distribution with fixed parameters.

Here, we generalize this method to point-clouds. We first note that a naive generalization of Mixup to point-clouds may not make sense since the points are arbitrarily ordered. In other words, such a combination would yield a meaningless point-cloud. Instead, we propose the following *Point-Cloud Mixup* (PCM) procedure. Given two point-clouds $x, x' \in \mathbb{R}^{n \times 3}$, we first sample a Mixup coefficient $\gamma \backsim Beta(\alpha, \beta)$ (We found that $\alpha, \beta = 1$ works well in our case). We then form a new shape by randomly sampling $\gamma \cdot n$ points from x and $(1-\gamma) \cdot n$ points from x'. The union of the sampled points yields a new point-cloud, $\overline{x} \in \mathbb{R}^{n \times 3}$. As in the original Mixup method, the label is a convex combination of the one-hot label vectors of the two point-clouds $\gamma y + (1-\gamma)y'$. See Figure 2 (green box) for examples of this procedure (colors are shown to help distinguish the shapes but are not a part of the input).

The main motivation for using Mixup for DA, especially for point-clouds, is that given a source distribution, the Mixup process generates a significantly wider variety of labeled point-clouds which is more likely to capture the target distribution.

Relation to Chen et al. [5]. In [5], the authors target object-detection on point-clouds. One of the data augmentation mechanisms used in the paper is a

method which the authors also call Mixup, in which they augment point-clouds with cropped objects from other point-clouds using their ground truth boxes. Our procedure is fundamentally different than theirs in two important aspects: (1) we construct a new point-cloud from two point-clouds by using a mixing coefficient like the original Mixup paper, and (2) we apply the Mixup to the labels; therefore, our use of mixup can be viewed as a new training procedure and not just data augmentation.

4 Experiments

We evaluated our method on a dataset designed by [30] for domain adaptation over point-clouds. The dataset consists of 3 subsets of three widely-used datasets: ShapeNet [3], ModelNet [49] and ScanNet [6]. All three subsets have the same ten distinct classes (like chair, table, bed).

ModelNet-10 (noted as ModelNet hereafter) contains 4183 train samples and 856 test samples sampled from clean 3D CAD models. *ShapeNet-10* (noted as ShapeNet hereafter), contains 17,378 train samples and 2492 test samples sampled from several online repositories of 3D CAD models. Due to this mix, classes in this set are more heterogeneous than ModelNet-10. *ScanNet-10* (noted as ScanNet hereafter) contains 6110 train and 1769 test samples. ScanNet is an RGB-D video dataset of scanned real-world indoor scenes. To generate a set suitable for the classification task, instances of 10 classes were cropped using annotated bounding boxes. Samples from this dataset are significantly harder to classify because: (i) Many objects are missing some parts, mostly due to"self-occlusion" since they were not scanned from all 360 degrees. (ii) Some objects are sampled sparsely. See Figure 3 for a comparison of typical shapes from all the datasets mentioned above.

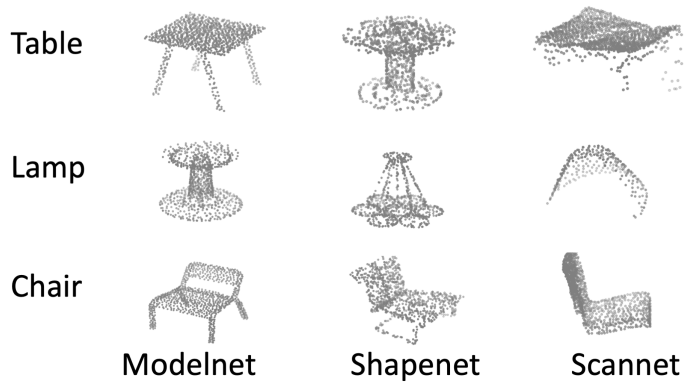


Fig. 3: A comparison of typical shapes from the datasets: ModelNet-10, ShapeNet-10 and ScanNet-10.

4.1 Data Processing & Experimental Setup

Following several studies [21, 28, 46] we assume that the upwards direction of all point-clouds in all datasets is known and aligned. Since point-clouds in ModelNet are aligned with the positive Z axis, we aligned samples from ShapeNet and ScanNet in the same direction by rotating them about the x-axis. We sampled 1024 points from shapes in ModelNet and ScanNet (which have 2048 points) using farthest point sampling (as in [28]). We split the training set to 80% for training and 20% for validation. We scaled shapes to the unit-cube and applied jittering as in [28] with standard deviation and clip parameters of 0.01 and 0.02 respectively. During training, we applied random rotations to shapes about the Z axis only.

We used a fixed batch size of 64, ADAM optimizer [19] and a cosine annealing learning rate scheduler as implemented by PyTorch. We rebalanced the domains by under-sampling the larger domain, source or target, in each epoch. We applied grid search over the learning rates $\{0.0005, 0.0001, 0.001\}$, weight decay $\{0.00005, 0.0001, 0.0005\}$ and SSL task weight $\lambda \in \{0.25, 1, 4\}$. We ran each configuration with 3 different seeds for 150 epochs and used source-validation based early stopping. In RegRec experiments, we defined a minimum threshold of 40 points for a region to be selected for reconstruction. This threshold guarantees that only meaningful regions are picked. The total training time of our proposed solution is 14 hours on a 16g Nvidia V100 GPU.

4.2 Architecture

The input to the network is a point-cloud that consists of 1024 points. For a feature extractor, we used DGCNN [46] with the same configurations as in the official PyTorch implementation: Four point-cloud convolution layers of sizes [64, 64, 128, 256] respectively and a 1D convolution layer with kernel size 1 (feature-wise fully connected) with a size of 1024 before extracting a global feature vector by max-pooling. The classification head h_{PCM} was implemented using three fully connected layers with sizes [512, 256, 10] respectively (where 10 is the number of classes). A dropout of 0.5 was applied to the two hidden layers. We implemented a spatial transformation network to align the input point set to a canonical space using two point-cloud convolution layers with sizes [64, 128] respectively, a 1D convolution layer of size 1024 and three fully connected layers of sizes [512, 256, 3] respectively.

The SSL head h_{SSL} takes as input the global feature vector (of size 1024) concatenated to the feature representations of each point from the initial four layers of the backbone network. The network was implemented using four 1D convolution layers of sizes [256, 256, 128, 3].

We applied batch normalization [18] after all convolution layers and used leaky relu activation with a slope of 0.2.

Table 1: Test set classification accuracy (%)

Method	ModelNet to ShapeNet	ModelNet to ScanNet	ShapeNet to ModelNet	ShapeNet to ScanNet	ScanNet to ModelNet	ScanNet to ShapeNet
Baselines						
Supervised-T	93.9 ± 0.4	78.4 ± 1.1	96.2 ± 0.2	78.4 ± 1.1	96.2 ± 0.2	93.9 ± 0.4
Supervised	89.2 ± 1.0	76.2 ± 1.0	93.4 ± 1.1	74.7 ± 1.2	93.2 ± 0.6	88.1 ± 1.2
Unsupervised	81.7 ± 0.2	42.9 ± 4.4	72.2 ± 1.4	44.2 ± 1.3	67.3 ± 3.8	65.1 ± 3.5
DANN [12]	75.3 ± 1.0	41.5 ± 0.4	62.5 ± 2.4	46.1 ± 4.9	53.3 ± 2.1	60.8 ± 1.9
RS-S/T [36]	79.4 ± 3.7	37.6 ± 8.8	65.1 ± 0.7	31.7 ± 3.8	62.8 ± 3.2	68.6 ± 2.4
PointDAN; PN [30]	80.2 ± 0.8	45.3 ± 2.0	71.2 ± 3.0	46.9 ± 3.3	59.8 ± 2.3	66.2 ± 4.8
PointDAN [30]	82.5 ± 1.3	44.5 ± 0.9	**77.0 ± 0.6**	48.5 ± 3.6	55.6 ± 1.0	67.2 ± 4.7
SSL, no PCM (ours)						
RegRec-T	82.1 ± 2.2	45.2 ± 0.7	73.9 ± 3.3	46.4 ± 1.1	69.7 ± 3.7	69.9 ± 1.9
RegRec-S/T; PN	80.0 ± 0.6	46.0 ± 5.7	68.5 ± 4.8	41.7 ± 1.9	63.0 ± 6.7	68.2 ± 1.1
RegRec-S/T	82.4 ± 1.5	**53.5 ± 1.9**	74.1 ± 2.2	44.5 ± 4.3	71.7 ± 1.3	69.1 ± 2.4
Baselines with PCM						
DANN	74.8 ± 4.9	42.1 ± 1.1	57.5 ± 0.7	50.9 ± 1.7	43.7 ± 5.0	63.6 ± 3.5
RS-T	82.0 ± 1.3	45.8 ± 4.6	71.4 ± 2.3	48.8 ± 3.4	72.6 ± 1.5	76.1 ± 3.7
RS-S/T	78.7 ± 3.7	47.0 ± 3.7	65.7 ± 1.0	50.3 ± 1.9	70.2 ± 4	73.9 ± 1.9
PointDAN; PN	82.7 ± 0.8	48.2 ± 0.8	66.3 ± 1.1	54.5 ± 1.1	47.3 ± 1.9	66.2 ± 4.3
PointDAN	**83.9 ± 0.6**	44.8 ± 2.5	63.3 ± 1.9	45.7 ± 1.2	43.6 ± 3.5	56.4 ± 2.6
SSL with PCM (ours)						
RegRec-T	81.9 ± 0.4	52.3 ± 1.2	71.7 ± 1.5	**55.3 ± 0.8**	**79.3 ± 1.9**	**76.7 ± 0.7**
RegRec-S/T; PN	81.1 ± 1.1	50.3 ± 2.0	54.3 ± 0.3	52.8 ± 2.0	54.0 ± 5.5	69.0 ± 0.9
RegRec-S/T	81.9 ± 1.1	52.5 ± 5.2	69.8 ± 1.4	51.1 ± 0.1	76.3 ± 4.7	76.0 ± 2.0

5 Results

Since our proposed solution is a combination of the RegRec task and the PCM training procedure, we compared our approach to all baselines twice: once as published (Table 1, first section) and then when adding PCM to each baseline (Table 1, third section). We use the acronyms S and T to denote methods applied to source and target samples respectively. The same pre-processing and model selection method described in section 4 was applied to all methods.

5.1 Classification Accuracy

We compared our approach with four baselines: (1) *Unsupervised*, using only labeled source samples without any modification to either source or target samples; (2) *DANN* [12], a baseline commonly used in the literature of DA for images; (3) *RS*, our architecture with an SSL task for point-clouds suggested in [36]; and (4) *PointDAN* [30] that suggested to align features both locally and globally. Since PointDAN used PointNet as a feature extractor with a batch size of 128, we also compared our approach to theirs in this setting (denoted *PN* in Table 1). Table 1 also presents two upper bounds: (1) *Supervised-T*, training with the target domain only and, (2) *Supervised*, training with source and target labels. The numbers reported in the tables are the mean classification accuracy and

standard deviation across three runs with different seeds. In addition, appendix A compares our method to the method presented in [16].

SSL vs Baselines. Table 1 (top part) shows that using RegRec on both source and target samples outperforms all competing standard baseline methods in 3 out of 6 adaptations and is comparable on the *ModelNet-to-ShapeNet* adaptation setup. Specifically, our method is better when simulated data is involved (note the large improvement of 8% on *ModelNet-to-ScanNet* adaptation setup compared to the best competitor). This observation validates our two intuitions that were discussed earlier: (i) Our SSL task promotes learning semantic properties of the shapes and (ii) The SSL task helps the model in generalizing to real data that has missing regions/parts.

SSL with PCM vs Baselines with PCM. Table 1 (bottom part) shows that our method outperforms all other competing methods on 5 out of 6 adaptation setups. That is, adding PCM to RegRec has a synergistic effect. We argue that PCM in a sense catches global dependencies by allowing to sample from a more diverse source distribution and RegRec captures local dependencies in a meso-scale resolution.

Global Comparison. When comparing all the methods in the table we note that our suggested methods (either *SSL without PCM* or *SSL with PCM*) improves the accuracy in 4 out of 6 adaptations compared to all baselines. We chose PCM with RegRec-T as our main method. This method outperforms all methods in 3 out of 6 adaptations. In addition, it outperforms all baseline methods (with and without PCM) on another setup (*ModelNet to ScanNet*).

It is interesting to see how PCM boosts almost all methods in sim-to-real adaptations but less so in sim-to-sim adaptations. Another interesting observation is that when PCM is used on source samples, adding RegRec does not always improve accuracy; possibly because of over regularization imposed by the two deformations.[1]

Finally, we note that applying pretext tasks inspired by image tasks are not guaranteed to help with 3D data. For example, the pretext task used in the RS baseline [36] which was inspired by the idea of shuffling image patches [26], does not work well and is sometimes even inferior to an unsupervised approach. This shows that it is crucial to design pre-text tasks that are specific to 3D data.

5.2 Accuracy per Class

Table 2 presents the accuracy for each class obtained with different methods. Here we show the variant that uses PCM on source samples and RegRec on target samples. We compare it with two baselines: *Unsupervised* and *PointDAN*. From the table, we notice that all methods are biased towards the common classes, yet our method is more resilient to the distribution shift compared to PointDAN. Specifically, our method performs better on some rare classes (bookshelf, plant) while PointDAN is more biased towards the more common classes (chair, sofa).

[1] We stress that each deformation was applied separately on source samples.

Table 2: Accuracy per class (%)

Method	Bathtab	Bed	Bookshelf	Cabinet	Chair	Lamp	Monitor	Plant	Sofa	Table	Avg.
ModelNet to ScanNet											
# Samples	26	85	146	149	801	41	61	25	134	301	–
Unsupervised	48.7	41.2	40.9	**3.8**	54.1	29.3	**57.9**	82.7	43.0	28.8	43.0
PointDAN [30]	56.4	**61.5**	29.9	2.4	**71.7**	30	42.6	26.6	**53**	14.8	38.9
PCM + RegRec-T (ours)	**57.7**	41.2	**49.8**	2	59.8	**35**	53.6	**88**	47.5	**62.8**	**49.7**
ModelNet to ShapeNet											
# Samples	85	23	50	126	662	232	112	30	330	842	–
Unsupervised	81.2	17.4	96.7	**1.6**	89.4	**66.5**	84.5	86.7	90.6	88.8	70.3
PointDAN [30]	82	36.2	**97.3**	0	**94.6**	54.90	**93.50**	95.6	**92.9**	**91.5**	**73.8**
PCM + RegRec-T (ours)	**87.5**	**43.5**	**97.3**	1.1	92.6	48.7	89.6	**96.7**	90.9	89.3	**73.7**

Table 3: Ablation study (%)

Method	ModelNet to ShapeNet	ModelNet to ScanNet	ShapeNet to ModelNet	ShapeNet to ScanNet	ScanNet to ModelNet	ScanNet to ShapeNet
RegRec-T	**82.1 ± 2.2**	45.2 ± 0.7	**73.9 ± 3.3**	46.4 ± 1.1	69.7 ± 3.7	69.9 ± 1.9
PCM	81.7 ± 1.0	49.7 ± 3.6	70.5 ± 2.2	50.7 ± 0.6	62.8 ± 1.0	74.2 ± 1.2
PCM + RegRec-T	81.9 ± 0.4	**52.3 ± 1.2**	71.7 ± 1.5	**55.3 ± 0.8**	**79.3 ± 1.9**	**76.7 ± 0.7**

We note that the low accuracies on the class *Cabinet* are probably because night-stand from the original ModelNet-40 data set [49] were regarded as cabinet when this dataset was designed. In summary, our proposed approach can be associated with learning at the tail of the distribution. This is an important topic that is outside the scope of this paper and we leave it to future study.

5.3 Ablation Experiments

To gain insight into the relative contribution of model components, we evaluate variants of our approach where we isolate the individual contribution of different components.

Table 3 presents three models: (1) *RegRec-T*, applying RegRec on target data only (no PCM), (2) *PCM*, applying PCM on source data only (without RegRec) and, (3) *PCM + RegRec-T*, applying PCM on source data and RegRec on target data. As can be seen from the table, when the RegRec and PCM are considered independently, no module consistently outperforms all other modules, yet when using all modules jointly (PCM + RegRec-T) there is a significant boost in the performance on almost all adaptation setups.

5.4 Deformation Scale

A key property of our proposed method is that it deforms a substantial region of the point cloud, large enough to contain meaningful semantic details, like the arm of a chair, or the leg of a table. An interesting question remains: How large should be the deformed region?

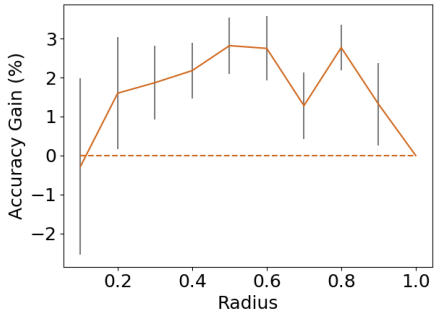


Fig. 4: Classification accuracy as a function of the deformation radius. Shown is the gain in accuracy compared with the accuracy for $r = 1.0$, averaged across six adaptation tasks. Error bars denote the standard deviation across six adaptations.

To answer this question we experimented with a similar deformation variant, which allows to smoothly control the size of the deformation region. In this variant, the deformation region is a sphere with a fixed radius r that is centered around one data point selected at random. As in RegRec, points in the deformation area are assigned new random locations that are sampled from a Gaussian distribution centered at the region center. Unlike RegRec, the deformation areas do not have to adhere to the 3-dimensional grid of $k \times k \times k$ voxels. A comparison between the methods is presented in appendix B.

Fig. 4 shows the mean difference across 6 adaptation tasks in the model performance between models with $r \in \{0.1, 0.2, \ldots, 1.0\}$. To make comparison across radii easier, we first subtracted the value at $r = 1.0$ from the curve of each adaptation, and then computed the average across adaptations, and the standard deviation.

Accuracy is highest mostly with middle-range radii, with an optimum at $r = 0.5$. This radius deforms objects at the scale of object parts. Here, a value of 1.0 is the length of half the side of the unit cube to which objects are scaled. Hence for large enough radii, all points of an object are being deformed, and the representation learned is less useful.

5.5 Qualitative Results

Fig. 5 demonstrate RegRec reconstruction from a deformed shape. Images of the same object are presented in the following order, the deformed shape (the input to the network), the original shape (the ground truth) and the reconstructed shape by the network. From the figure, it seems that the network manages to learn two important things: (1) It learns to recognize the deformed region and (2) it learns to reconstruct the region in a way that preserves the original shape. In appendix D we show examples of RegRec shape reconstruction for all classes

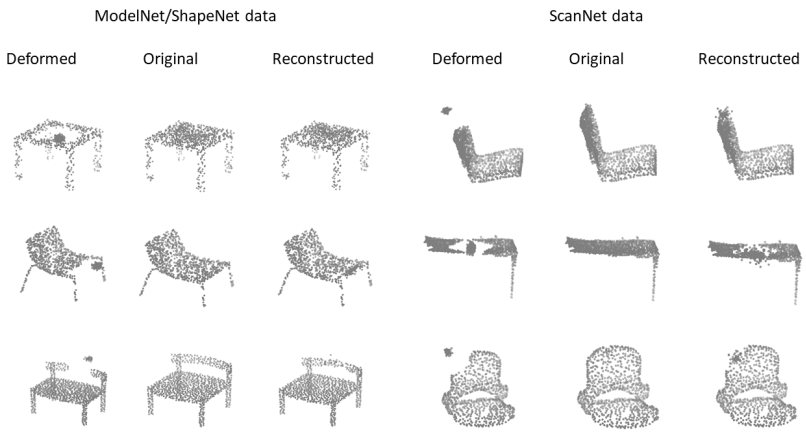


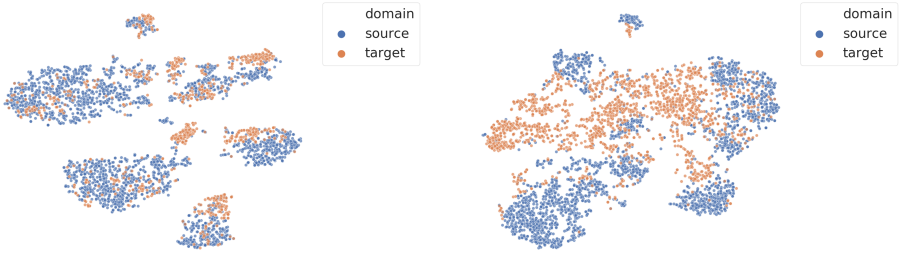
Fig. 5: Illustration of target reconstruction. Each triplet shows a sample deformed using RegRec, the ground truth original, and the resulting reconstruction. Left triplets: ShapeNet/ModelNet. Right triplets: ScanNet.

in the dataset. We also examine the reconstruction quality of different regions in the same shape.

Fig. 6 presents source and target distribution of test samples' activations of the last hidden layer in the classification network with t-SNE [22] visualization. Not surprisingly, it can be seen that data from the two simulated domains (*ShapeNet* and *ModelNet*) is more interleaved compared to data from a simulated domain (*ShapeNet*) and real domain (*ScanNet*). The distribution of target samples in the sim-to-real setup is denser. A related notion was presented in [52], showing that target samples tend to have lower norms compared to source samples. This shows that there is still room for improvement in this difficult adaptation task. We leave this to future work. Further analysis of the learned representation is shown in appendix C.

6 Conclusions

In this paper, we tackled the problem of domain adaptation on 3D point-clouds. We argue that using proper self-supervised pretext tasks helps in learning transferable representations that benefit the domain adaptation task. We designed *RegRec*; a novel self-supervised task inspired by the kind of deformations encountered in real 3D point-cloud data. In addition, we designed *PCM*; a new training procedure for 3D point-clouds based on the Mixup method. PCM is complementary to RegRec, and when combined they form a strong model with relatively simple architecture. We showed that our method is not sensitive to specific design choices made and we demonstrated the benefit of our method in a benchmark dataset on several adaptation setups, reaching new SoTA.



(a) ShapeNet to ModelNet (b) ShapeNet to ScanNet

Fig. 6: The distribution of samples from the Source (blue) domain and Target (orange) domains. Left: Shapnet to ModelNet (sim-to-sim). Right: ShapeNet to ScanNet (sim-to-real).

Acknowledgments

This study was funded by a grant to GC from the Israel Science Foundation (ISF 737/2018), and by an equipment grant to GC and Bar-ILan University from the Israel Science Foundation (ISF 2332/18).

References

1. Atzmon, M., Maron, H., Lipman, Y.: Point convolutional neural networks by extension operators. arXiv preprint arXiv:1803.10091 (2018)
2. Carlucci, F.M., D'Innocente, A., Bucci, S., Caputo, B., Tommasi, T.: Domain generalization by solving jigsaw puzzles. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 2229–2238 (2019)
3. 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)
4. Chen, S., Duan, C., Yang, Y., Li, D., Feng, C., Tian, D.: Deep unsupervised learning of 3D point clouds via graph topology inference and filtering. IEEE Transactions on Image Processing (2019)
5. Chen, Y., Liu, S., Shen, X., Jia, J.: Fast point r-CNN. In: Proceedings of the IEEE International Conference on Computer Vision. pp. 9775–9784 (2019)
6. Dai, A., Chang, A.X., Savva, M., Halber, M., Funkhouser, T., Nießner, M.: ScanNet: Richly-annotated 3D Reconstructions of Indoor Scenes. In: Proc. Computer Vision and Pattern Recognition (CVPR), IEEE (2017)
7. Doersch, C., Gupta, A., Efros, A.A.: Unsupervised visual representation learning by context prediction. In: Proceedings of the IEEE International Conference on Computer Vision. pp. 1422–1430 (2015)
8. Dosovitskiy, A., Fischer, P., Springenberg, J.T., Riedmiller, M., Brox, T.: Discriminative unsupervised feature learning with exemplar convolutional neural networks. IEEE transactions on pattern analysis and machine intelligence **38**(9), 1734–1747 (2015)
9. Feng, Z., Xu, C., Tao, D.: Self-supervised representation learning from multi-domain data. In: Proceedings of the IEEE International Conference on Computer Vision. pp. 3245–3255 (2019)
10. Fernando, B., Bilen, H., Gavves, E., Gould, S.: Self-supervised video representation learning with odd-one-out networks. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 3636–3645 (2017)
11. Ganin, Y., Lempitsky, V.: Unsupervised domain adaptation by backpropagation. In: Proceedings of the 32nd International Conference on International Conference on Machine Learning-Volume 37. pp. 1180–1189 (2015)
12. Ganin, Y., Ustinova, E., Ajakan, H., Germain, P., Larochelle, H., Laviolette, F., Marchand, M., Lempitsky, V.: Domain-adversarial training of neural networks. The Journal of Machine Learning Research **17**(1), 2096–2030 (2016)
13. Ghifary, M., Kleijn, W.B., Zhang, M., Balduzzi, D., Li, W.: Deep reconstruction-classification networks for unsupervised domain adaptation. In: European Conference on Computer Vision. pp. 597–613. Springer (2016)
14. Gidaris, S., Singh, P., Komodakis, N.: Unsupervised representation learning by predicting image rotations. In: International Conference on Learning Representations (2018), https://openreview.net/forum?id=S1v4N2l0-
15. Guo, Y., Wang, H., Hu, Q., Liu, H., Liu, L., Bennamoun, M.: Deep learning for 3D point clouds: A survey (2019)
16. Hassani, K., Haley, M.: Unsupervised multi-task feature learning on point clouds. In: Proceedings of the IEEE International Conference on Computer Vision. pp. 8160–8171 (2019)
17. Hua, B.S., Tran, M.K., Yeung, S.K.: Pointwise convolutional neural networks. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 984–993 (2018)

18. Ioffe, S., Szegedy, C.: Batch normalization: Accelerating deep network training by reducing internal covariate shift. In: International Conference on Machine Learning. pp. 448–456 (2015)
19. Kingma, D.P., Ba, J.: ADAM: A method for stochastic optimization. In: Proc. of the 3rd International Conference on Learning Representations (2014)
20. Klokov, R., Lempitsky, V.: Escape from cells: Deep kd-networks for the recognition of 3d point cloud models. In: Proceedings of the IEEE International Conference on Computer Vision. pp. 863–872 (2017)
21. Li, Y., Bu, R., Sun, M., Wu, W., Di, X., Chen, B.: Pointcnn: Convolution on x-transformed points. In: Advances in neural information processing systems. pp. 820–830 (2018)
22. Maaten, L.v.d., Hinton, G.: Visualizing data using t-SNE. Journal of machine learning research **9**(Nov), 2579–2605 (2008)
23. Mao, X., Ma, Y., Yang, Z., Chen, Y., Li, Q.: Virtual mixup training for unsupervised domain adaptation. arXiv preprint arXiv:1905.04215 (2019)
24. Maturana, D., Scherer, S.: Voxnet: A 3D convolutional neural network for real-time object recognition. In: 2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS). pp. 922–928. IEEE (2015)
25. Misra, I., Zitnick, C.L., Hebert, M.: Shuffle and learn: unsupervised learning using temporal order verification. In: European Conference on Computer Vision. pp. 527–544. Springer (2016)
26. Noroozi, M., Favaro, P.: Unsupervised learning of visual representations by solving jigsaw puzzles. In: European Conference on Computer Vision. pp. 69–84. Springer (2016)
27. Pathak, D., Krahenbuhl, P., Donahue, J., Darrell, T., Efros, A.A.: Context encoders: Feature learning by inpainting. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 2536–2544 (2016)
28. Qi, C.R., Su, H., Mo, K., Guibas, L.J.: Pointnet: Deep learning on point sets for 3d classification and segmentation. Proc. Computer Vision and Pattern Recognition (CVPR), IEEE **1**(2), 4 (2017)
29. Qi, C.R., Su, H., Nießner, M., Dai, A., Yan, M., Guibas, L.J.: Volumetric and multi-view CNNs for object classification on 3D data. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 5648–5656 (2016)
30. Qin, C., You, H., Wang, L., Kuo, C.C.J., Fu, Y.: PointDAN: A multi-scale 3D domain adaption network for point cloud representation. In: Advances in Neural Information Processing Systems. pp. 7190–7201 (2019)
31. Ren, Z., Jae Lee, Y.: Cross-domain self-supervised multi-task feature learning using synthetic imagery. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 762–771 (2018)
32. Rist, C.B., Enzweiler, M., Gavrila, D.M.: Cross-sensor deep domain adaptation for LiDAR detection and segmentation. In: 2019 IEEE Intelligent Vehicles Symposium (IV). pp. 1535–1542. IEEE (2019)
33. Saito, K., Kim, D., Sclaroff, S., Saenko, K.: Universal domain adaptation through self supervision. arXiv preprint arXiv:2002.07953 (2020)
34. Saito, K., Watanabe, K., Ushiku, Y., Harada, T.: Maximum classifier discrepancy for unsupervised domain adaptation. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 3723–3732 (2018)
35. Saleh, K., Abobakr, A., Attia, M., Iskander, J., Nahavandi, D., Hossny, M., Nahvandi, S.: Domain adaptation for vehicle detection from bird's eye view LiDAR point cloud data. In: Proceedings of the IEEE International Conference on Computer Vision Workshops. pp. 0–0 (2019)

36. Sauder, J., Sievers, B.: Self-supervised deep learning on point clouds by reconstructing space. In: Advances in Neural Information Processing Systems. pp. 12942–12952 (2019)
37. Snell, J., Swersky, K., Zemel, R.: Prototypical networks for few-shot learning. In: Advances in neural information processing systems. pp. 4077–4087 (2017)
38. Su, H., Jampani, V., Sun, D., Maji, S., Kalogerakis, E., Yang, M.H., Kautz, J.: Splatnet: Sparse lattice networks for point cloud processing. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 2530–2539 (2018)
39. Sun, Y., Tzeng, E., Darrell, T., Efros, A.A.: Unsupervised domain adaptation through self-supervision. arXiv preprint arXiv:1909.11825 (2019)
40. Tang, L., Chen, K., Wu, C., Hong, Y., Jia, K., Yang, Z.: Improving semantic analysis on point clouds via auxiliary supervision of local geometric priors. arXiv preprint arXiv:2001.04803 (2020)
41. Thabet, A., Alwassel, H., Ghanem, B.: Mortonnet: Self-supervised learning of local features in 3D point clouds. arXiv preprint arXiv:1904.00230 (2019)
42. Thulasidasan, S., Chennupati, G., Bilmes, J.A., Bhattacharya, T., Michalak, S.: On mixup training: Improved calibration and predictive uncertainty for deep neural networks. In: Advances in Neural Information Processing Systems. pp. 13888–13899 (2019)
43. Tzeng, E., Hoffman, J., Saenko, K., Darrell, T.: Adversarial discriminative domain adaptation. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 7167–7176 (2017)
44. Tzeng, E., Hoffman, J., Zhang, N., Saenko, K., Darrell, T.: Deep domain confusion: Maximizing for domain invariance. arXiv preprint arXiv:1412.3474 (2014)
45. Wang, X., Gupta, A.: Unsupervised learning of visual representations using videos. In: Proceedings of the IEEE International Conference on Computer Vision. pp. 2794–2802 (2015)
46. Wang, Y., Sun, Y., Liu, Z., Sarma, S.E., Bronstein, M.M., Solomon, J.M.: Dynamic graph CNN for learning on point clouds. ACM Transactions on Graphics (TOG) **38**(5), 1–12 (2019)
47. Wei, D., Lim, J.J., Zisserman, A., Freeman, W.T.: Learning and using the arrow of time. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 8052–8060 (2018)
48. 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: 2019 International Conference on Robotics and Automation (ICRA). pp. 4376–4382. IEEE (2019)
49. 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 IEEE conference on computer vision and pattern recognition. pp. 1912–1920 (2015)
50. Xu, J., Xiao, L., López, A.M.: Self-supervised domain adaptation for computer vision tasks. IEEE Access **7**, 156694–156706 (2019)
51. Xu, M., Zhang, J., Ni, B., Li, T., Wang, C., Tian, Q., Zhang, W.: Adversarial domain adaptation with domain mixup. arXiv preprint arXiv:1912.01805 (2019)
52. Xu, R., Li, G., Yang, J., Lin, L.: Larger norm more transferable: An adaptive feature norm approach for unsupervised domain adaptation. In: Proceedings of the IEEE International Conference on Computer Vision. pp. 1426–1435 (2019)
53. Xu, X., Zhou, X., Venkatesan, R., Swaminathan, G., Majumder, O.: d-SNE: Domain adaptation using stochastic neighborhood embedding. In: Proceedings of the

IEEE Conference on Computer Vision and Pattern Recognition. pp. 2497–2506 (2019)
54. Yan, S., Song, H., Li, N., Zou, L., Ren, L.: Improve unsupervised domain adaptation with mixup training. arXiv preprint arXiv:2001.00677 (2020)
55. Zaheer, M., Kottur, S., Ravanbakhsh, S., Poczos, B., Salakhutdinov, R.R., Smola, A.J.: Deep sets. In: Advances in neural information processing systems. pp. 3391–3401 (2017)
56. Zhang, H., Cisse, M., Dauphin, Y.N., Lopez-Paz, D.: mixup: Beyond empirical risk minimization. International Conference on Learning Representations (2018), https://openreview.net/forum?id=r1Ddp1-Rb
57. Zhang, L., Zhu, Z.: Unsupervised feature learning for point cloud by contrasting and clustering with graph convolutional neural network. arXiv preprint arXiv:1904.12359 (2019)

Appendix

A Comparison with Gaussian Perturbation

In this section, we compare our method to i.i.d Gaussian perturbation of the input point cloud. An SSL reconstruction task that uses this deformations process was suggested recently in [16]. We term this method *Reconstruct Gaussian Perturbed Shape (RGPS)*. Unlike our architecture, in which we concatenate the point features to the global shape features and apply 1D convolution, [16] reconstructs the shape from the global shape feature using an MLP. Overall, from Table 4 we conclude that our method is superior to reconstructing a Gaussian perturbed shape.

Table 4 compares between the following: (1) *RGPS-S/T*, the method proposed by [16] applied to both source and target samples. (2) *PCM + RGPS-T*, applying PCM on source samples and RGPS on target samples. (3) *PCM + RGPS-S/T*, applying PCM on source samples and RGPS on both source and target samples.[2] And (4) *PCM + RegRec-T* (our best model).

Table 4: Test-set classification accuracy - Gaussian perturbation (%)

Method	**ModelNet to ShapeNet**	**ModelNet to ScanNet**	**ShapeNet to ModelNet**	**ShapeNet to ScanNet**	**ScanNet to ModelNet**	**ScanNet to ShapeNet**
RGPS-S/T [16]	81.2 ± 0.9	49.6 ± 2.6	**75.9 ± 2.3**	46.6 ± 1.7	67.4 ± 0.3	64.7 ± 1.2
PCM + RGPS-T [16]	**83.1 ± 0.8**	47.2 ± 1.3	70.0 ± 1.8	52.8 ± 1.1	67.7 ± 3.6	73.7 ± 1.1
PCM + RGPS-S/T [16]	81.4 ± 2.0	51.7 ± 1.9	72.2 ± 3.0	48.3 ± 1.5	72.4 ± 1.7	71.1 ± 1.6
PCM + RegRec-T (ours)	81.9 ± 0.4	**52.3 ± 1.2**	71.7 ± 1.5	**55.3 ± 0.8**	**79.3 ± 1.9**	**76.7 ± 0.7**

Table 4 shows that in adaptations that involve ScanNet (real data) our methods obtain higher accuracies compared to RGPS based methods. Some by a large margin (such as *ShapeNet to ScanNet* and *ScanNet to ModelNet*). RGPS based methods are superior to ours in sim-to-sim adaptations. These results are another indication of the importance of deformation in a meso-scale resolution. We also note that PCM helps boost RGPS based methods in all adaptations except *ShapeNet to ModelNet*.

B Model Configurations

Recall that in our deformation process, we first split a bounding box to $3 \times 3 \times 3$ voxels, and then replace all the points in the chosen voxel with points sampled from a Gaussian distribution. Table 5 presents the model performance for alternative design choices in this process:

[2] We stress that RGPS and PCM weren't applied together on the same source shape, but rather separately.

Table 5: Model configurations (%)

Method	**ModelNet to ShapeNet**	**ModelNet to ScanNet**	**ShapeNet to ModelNet**	**ShapeNet to ScanNet**	**ScanNet to ModelNet**	**ScanNet to ShapeNet**
PCM + RegRec-T	81.9 ± 0.4	52.3 ± 1.2	71.7 ± 1.5	**55.3 ± 0.8**	**79.3 ± 1.9**	**76.7 ± 0.7**
PCM + RegRec-T ($2 \times 2 \times 2$)	81.8 ± 0.6	50.2 ± 5	72.5 ± 1.8	52.0 ± 3.5	73.8 ± 2.2	**76.8 ± 2.3**
PCM + RegRec-T (Uniform)	**83.0 ± 0.3**	52.9 ± 1.9	71.9 ± 1.6	52.9 ± 0.8	70.2 ± 2.8	72.4 ± 1.7
PCM + RegRec-T (Region Perturbation)	82.4 ± 0.8	**56.1 ± 0.8**	70.3 ± 4.2	51.7 ± 2.1	72.2 ± 5.1	74.1 ± 3.6
PCM + RegRec-T ($radius = 0.2$)	82.2 ± 1.1	54.9 ± 0.9	**72.8 ± 0.7**	52.8 ± 0.6	72.2 ± 2.6	73.4 ± 3.8
PCM + RegRec-T ($radius = 0.33$)	81.4 ± 0.3	52.2 ± 3.4	69.4 ± 2.7	54.2 ± 1.3	71.2 ± 3.7	75.6 ± 2.1
PCM + RegRec-T ($radius = 0.5$)	82.5 ± 0.5	53.9 ± 1.4	**72.8 ± 5.8**	54.4 ± 0.8	76.8 ± 2.4	75.2 ± 0.3

- *PCM + RegRec-T ($2 \times 2 \times 2$).* A lower splitting resolution of $2 \times 2 \times 2$ (a total of 8 voxels). As can be seen in Table 5 this method is comparable with our best method, yet there is some degradation in the results. We argue that it may stem from large deformed region that are not informative enough.
- *PCM + RegRec-T (uniform).* Replacing the Gaussian distribution with a uniform distribution in the voxel. From Table 5 we notice that this approach yields comparable results to our proposed approach. Furthermore, in the adaptation *ModelNet to ShapeNet* this approach achieves the highest accuracy among all of our methods.
- *PCM + RegRec-T (Region Perturbation).* Reconstructing a region deformed by a small Gaussian noise added to each point in it. From the table we see that this approach achieves comparable results on some adaptations and even a new best score on *ModelNet to ScanNet* adaptation, yet our proposed deformation is better in four out of six adaptations. This means that RegRec is able to learn some semantic properties that *PCM + RegRec-T (Region Perturbation)* cannot.
- *PCM + RegRec-T ($radius = x$).* Recall that in section 5.4 we suggested an extension to our proposed solution by replacing RegRec region selection with a ball encapsulating all points that are within a $radius > 0$ around a randomly chosen point. This method allows more flexibility in the choice of regions to reconstruct. Here, we present *PCM + RegRec-T* model performance for all adaptations with $radius \in \{0.2, 0.33, 0.5\}$. From Table 5 we conclude that this method is comparable to RegRec region selection method, with a slight advantage to the latter.

To summarize, we showed that our method is not sensitive to the specific design choices we made. Using SSL for domain adaptation according to the method described in this paper can be generalized in many different ways while preserving model performance.

Table 6: Log perplexity (lower is better)

Method	Standrad Perplexity	Class-Balanced Perplexity
ModelNet to ScanNet		
PointDAN	**25.3 ± 4.3**	36.4 ± 3.5
PCM + RegRec	**25.2 ± 1.7**	**33.4 ± 1.7**
ModelNet to ShapeNet		
PointDAN	6.8 ± 0.43	23.6 ± 3.5
PCM + RegRec	**5.3 ± 0.3**	**9.6 ± 0.61**

C Estimating Target Perplexity

A key property of a DA solution is the ability to find an alignment between source and target distributions that is also discriminative [34]. To test that we suggest to measure the log perplexity of target test data representation under a model fitted by source test data representation. Here we consider the representation of samples as the activations of the last hidden layer in the classification network. The log perplexity measures the average number of bits required to encode a test sample. A lower value indicates a better model with less uncertainty in it.

Let $(x_1^t, y_1^t), ..., (x_1^t, y_n^t) \in T$ be a set of target instances. We note by n_c the number of target instances belonging to class c. Using the chain rule, the likelihood of the joint distribution $p(x_j^t, y_j^t = c)$ can be estimated by the conditional distribution $P(x_j^t | y_j^t = c)$ and the marginal distribution $P(y_j^t = c)$. To model $P(x_j^t | y_j^t = c)$ per-class we propose to fit a Gaussian distribution $N(\mu_c, \Sigma_c)$ based on source samples from class c using maximum likelihood. To model $p(y_j^t = c)$ we take the proportion of source samples in class c.

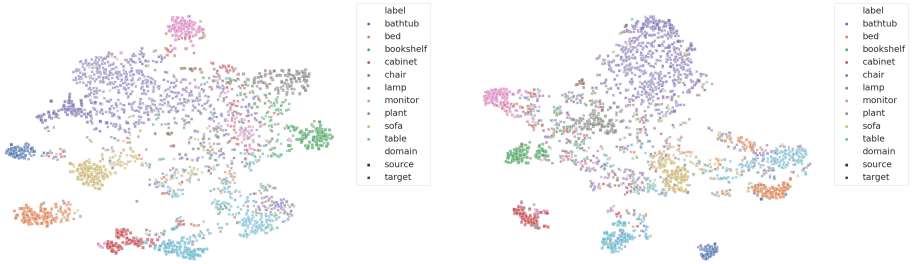
Modeling the marginal distribution with a Gaussian distribution relates to the notion proposed in [37]. [37] suggested to represent each class with a prototype (the mean embeddings of samples belonging to the class) and assign a new instance to the class associated with the closest prototype. The distance metric used is the squared Euclidean distance. This method is equivalent to fitting a Gaussian distribution for each class with a unit covariance matrix.

The log perplexity of the target is (noted as standard perplexity here after):

$$L(T) = \sum_{c=1}^{10} \sum_{j=1}^{n_c} \frac{1}{n} \log \left(p(x_j^t | y_j^t = c) p(y_j^t = c) \right) \tag{4}$$

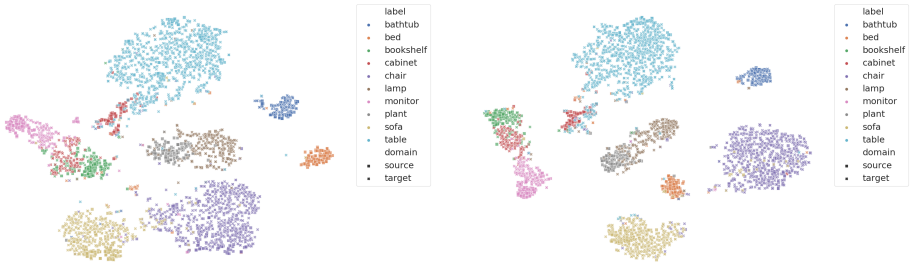
Alternatively we can measure the mean of a class-balanced log perplexity (noted as class-balanced perplexity here after):

$$L(T) = \frac{1}{10} \sum_{c=1}^{10} \sum_{j=1}^{n_c} \frac{1}{n_c} \log \left(p(x_j^t | y_j^t = c) p(y_j^t = c) \right) \tag{5}$$



(a) PCM + RegRec (b) PointDAN

Fig. 7: The distribution of samples for the adaptation ModelNet to ScanNet



(a) PCM + RegRec (b) PointDAN

Fig. 8: The distribution of samples for the adaptation ModelNet to ShapeNet

Table 6 shows the standard perplexity and class-balanced perplexity of *PCM + RegRec* and *PointDAN* [30] for the adaptations *ModelNet to ScanNet* and *ModelNet to ShapeNet*. Estimating the perplexity on the original space requires estimating a covariance matrix from relatively small number of samples which results in a degenerate matrix. Therefore, we estimated the perplexity after applying dimensionality reduction to a 2D space using t-SNE. We ran t-SNE with the same configurations with ten different seeds and reported the mean and standard error of the mean. In Figures 7 and 8 we plot the t-SNE representations of one of the seeds.

From the table and the figures we see that our method creates target and source representations that are more similar. When considering the class-balanced perplexity the gap is even larger. This is another indication that our model is doing a better job at learning under-represented classes. We note that *PointDAN* creates a denser representation of some classes (especially well-represented classes such as *Chair* and *Table*), however they are not mixed better between source and target.

D Additional Shape Reconstruction Results

Fig. 9 presents *PCM + RegRec-T* reconstruction of shapes from all classes in the data for the simulated domains (left column) and the real domain (right column). Note how in some cases, such as *Monitor* on the left column and *Lamp* on the right column, the reconstruction is not entirely consistent with the ground truth. The network reconstructs the object in a different (but still plausible) manner.

Figs 10 - 12 show *PCM + RegRec-T* reconstruction of *Chair*, *Table* and *Lamp* objects respectively from deformations of different regions in the objects. It can be seen that the network learns to reconstruct some regions nicely (such as the chair's top rail or table legs) while it fails to reconstruct well other regions (such as the chair's seat and the lamp's base). We speculate that it is harder for the network to reconstruct regions that have more diversity in the training set.

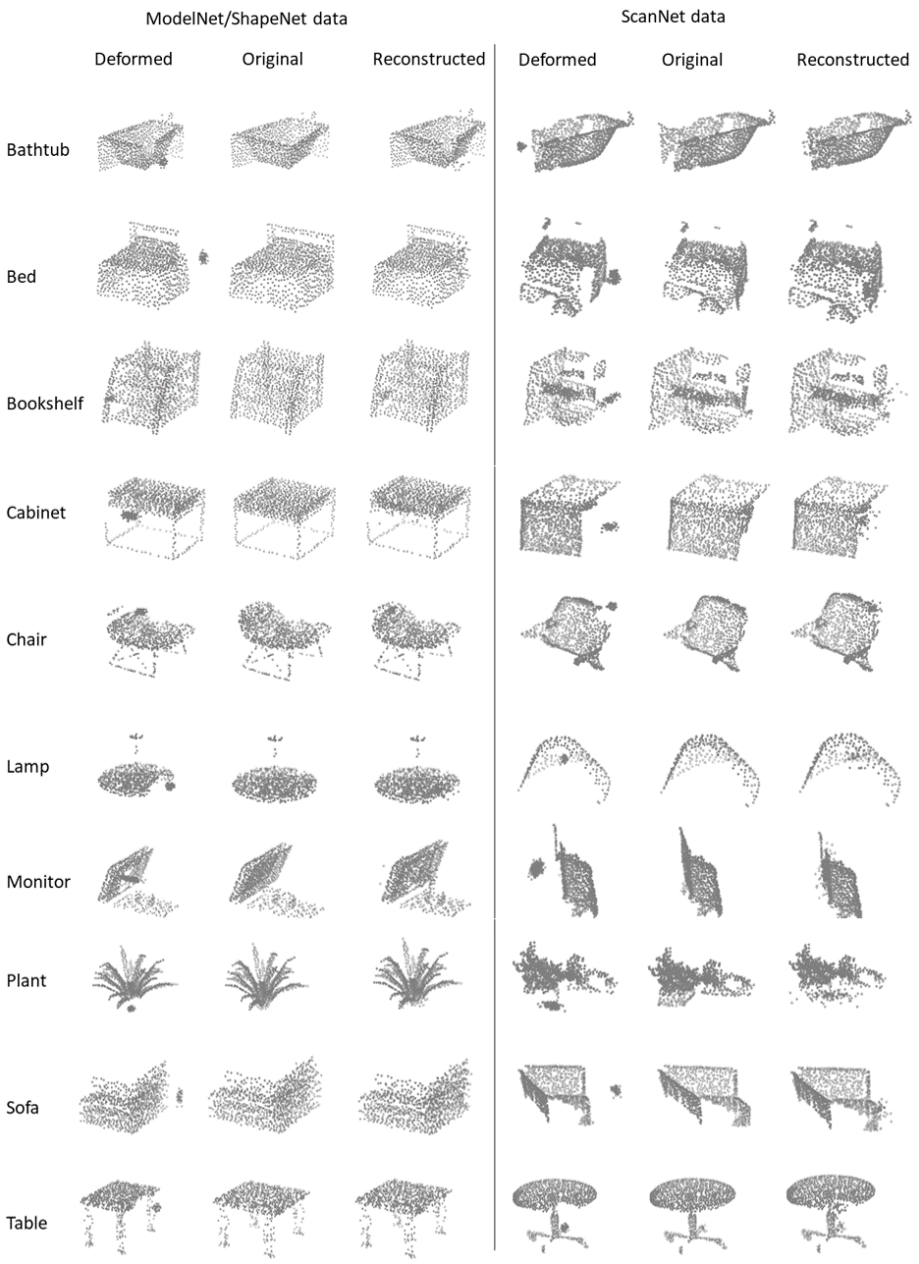


Fig. 9: Illustration of target reconstruction of all classes. Each triplet shows a sample deformed using PCM + RegRec-T, the ground truth original, and the resulting reconstruction. Left triplets: ShapeNet/ModelNet. Right triplets: ScanNet

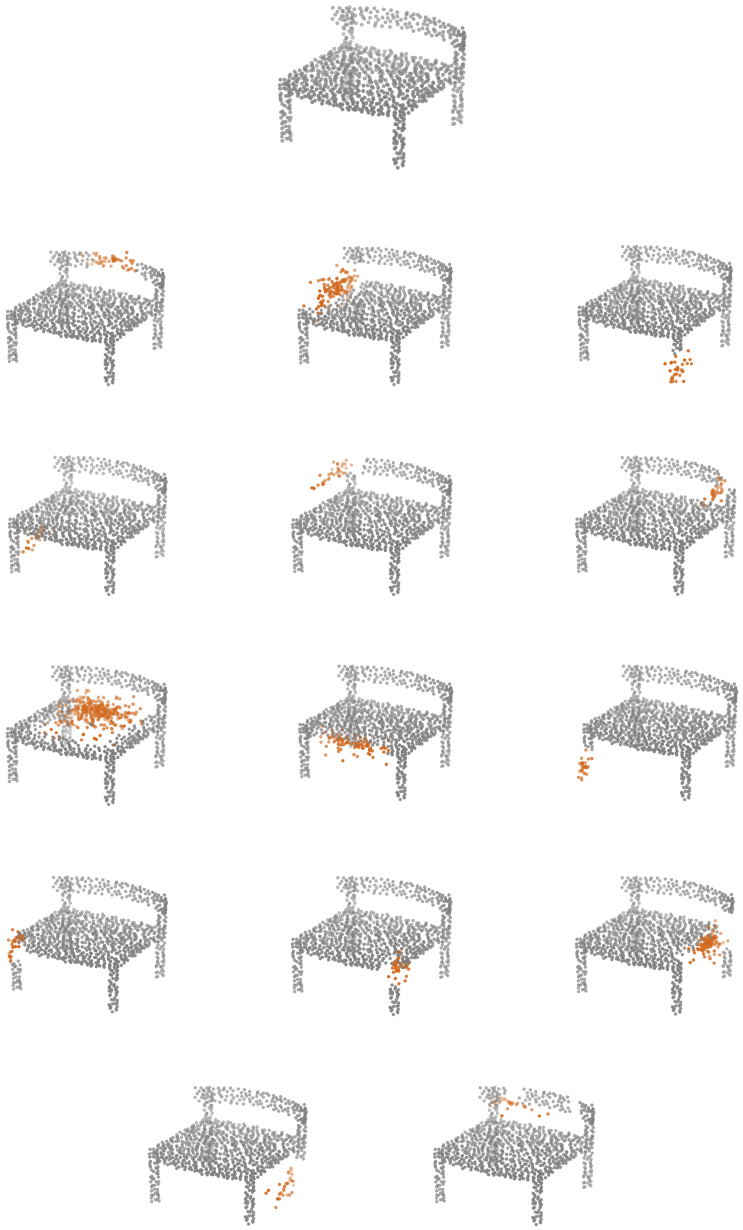


Fig. 10: Reconstruction using RegRec-T + PCM of a Chair object from ModelNet. The object in the first row is the ground truth. The other objects are reconstructed, each from deformation of different region in the object. The reconstructed region is marked by orange points

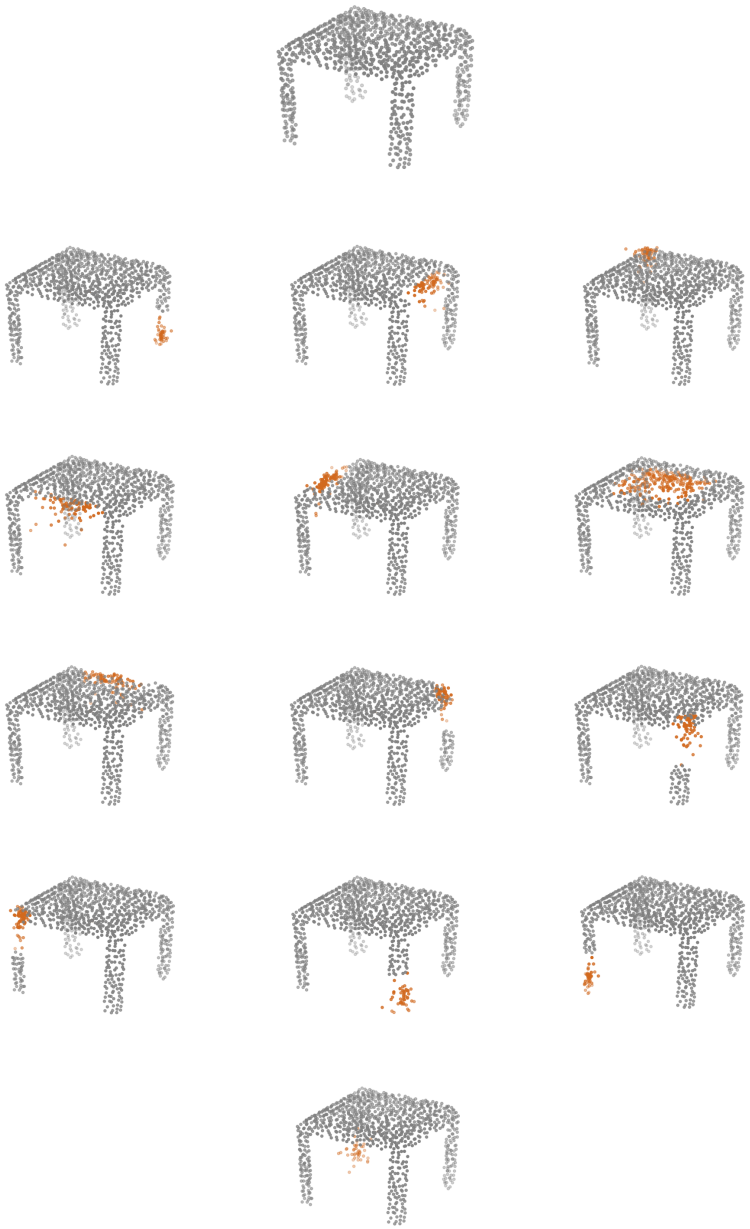


Fig. 11: Reconstruction using RegRec-T + PCM of a Table object from ModelNet. The object in the first row is the ground truth. The other objects are reconstructed, each from deformation of different region in the object. The reconstructed region is marked by orange points

Fig. 12: Reconstruction using RegRec-T + PCM of a Lamp object from ModelNet. The object in the first row is the ground truth. The other objects are reconstructed, each from deformation of different region in the object. The reconstructed region is marked by orange points