Joint supervised and self-supervised learning for 3D real world challenges

Joint Supervised and Self-Supervised Learning for 3D Real World Challenges

Antonio Alliegro[1], Davide Boscaini[2], Tatiana Tommasi[1,3]
[1]Politecnico di Torino, Turin, Italy email: {name.surname}@polito.it
[2]Fondazione Bruno Kessler, Trento, Italy email: dboscaini@fbk.eu
[3] Istituto Italiano di Tecnologia, Torino, Italy

***Abstract*—Point cloud processing and 3D shape understanding are challenging tasks for which deep learning techniques have demonstrated great potentials. Still further progresses are essential to allow artificial intelligent agents to interact with the real world. In many practical conditions the amount of annotated data may be limited and integrating new sources of knowledge becomes crucial to support autonomous learning. Here we consider several scenarios involving synthetic and real world point clouds where supervised learning fails due to data scarcity and large domain gaps. We propose to enrich standard feature representations by leveraging self-supervision through a multi-task model that can solve a 3D puzzle while learning the main task of shape classification or part segmentation. An extensive analysis investigating few-shot, transfer learning and cross-domain settings shows the effectiveness of our approach with state-of-the-art results.**

I. Introduction

Artificial intelligence made great strides in recent years with terrific results in the area of computer vision. This naturally reflects in our every-day life with apps able to collect images and automatically classify objects, detect faces, recognize places, and much more. Still the task of fully understanding our real world remains far-fetched. The reasons range from the inherent difficulty of dealing with a three-dimensional space, as well as time and domain variations which makes it hard to learn robust models able to generalize to any new scenario. Currently, many of those issues are the same that maintain a large gap between having a smartphone in our hands and a robot at home. Embodied intelligent systems need more than 2D visual perception: 3D shapes have to be reliably recognized regardless of the plethora of environmental constraints, and possibly segmented in their functional parts to allow a robot completing a simple tasks as can be opening a honey jar.

Thanks to the rise of powerful computational resources, 3D research is also progressively flourishing together with new ways to collect and describe 3D data. LiDAR scanners and stereo cameras gave rise to massive point cloud datasets possibly spanning even an entire city. However, they come with three main drawbacks: point clouds are un-structured, un-ordered and eager for precise manual annotation due to many possible sources of noise. The first two properties make typical convolutional neural networks (CNN) unsuitable for point clouds. Possible solutions consist in rendering point clouds to multiple 2D views or pre-processing them with a voxelization procedure to make data suitable for 3D CNN, but these techniques are either computationally expensive or come with inevitable loss of information and with negative effects on the overall recognition or segmentation performance. The third property has initially guided research towards very well lab-controlled and synthetic CAD object datasets where labeling is simpler. However, the most recent results on those kind of testbed are witnessing a trend of performance saturation raising the question of how to move forward. All these challenges describe a research area in need of new deep learning models able to deal with large amount of unsupervised real-world point cloud data.

We can summarize the contributions of our work as following:

- **we design a new research landscape by tackling at once several aspects of cross-domain learning for 3D vision in real world scenarios.** Recent 2D image analysis literature has shown how the lack of data annotation and the possible domain shift issues can be alleviated by integrating different source of knowledge in the learning process through Transfer Learning (TL), Domain Adaptation (DA) and Generalization (DG). Self-supervised learning has also shown to be a helpful support (see Section II for an overview). We investigate how to follow this trend for 3D tasks.
- **we propose a new multi-task end-to-end deep learning model for point clouds that combines supervised and self-supervised learning** (see Figure 1). Specifically we define a deep architecture that solves 3D puzzles while jointly training a main supervised task. We show how these two tasks complement each other making the obtained model (a) more precise in case of large amount of labeled data, (b) more robust in case of scarce labeled data, (c) easier to transfer for adaptation and (d) more reliable for out of domain generalization.
- **we present extensive experiments across four different point clouds datasets**: our multi-task method outperforms the standard supervised learning baseline and defines the new state-of-the-art for both shape classification and part segmentation in the most challenging real world settings.

II. Related Work

How to use powerful deep learning methods for *supervised learning* of classification and segmentation models on point clouds is an extremely active area of research. Early approaches root back to PointNet [1], a MLP architecture combining symmetry and spatial transform functions to learn point features which are then aggregated into global shape represen-

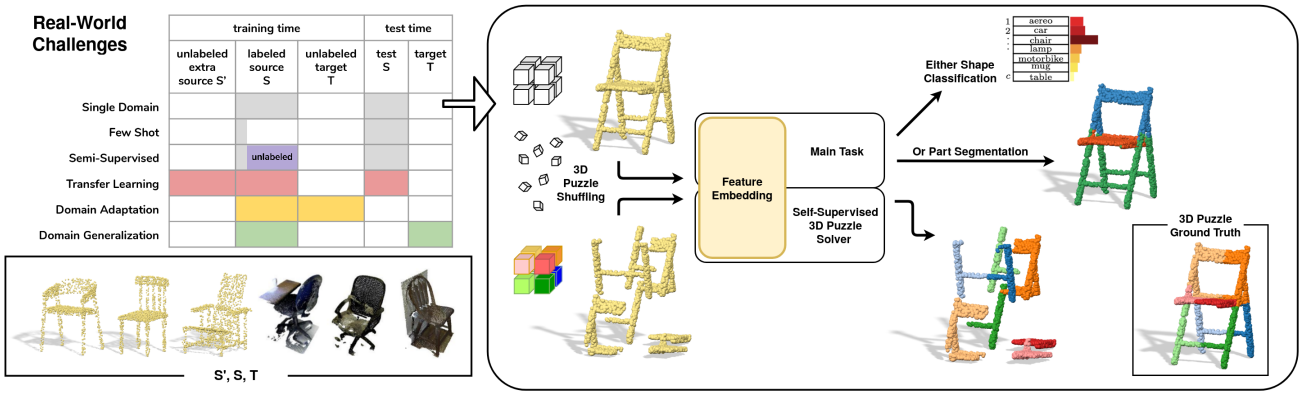


Fig. 1. Overview of the considered real world challenges and of the proposed multi-task approach. We deal with different domains covering real and synthetic 3D point clouds as well as several learning settings across domains and scarce annotations.

tations. PointNet++ [2] extended the previous model by hierarchically combining multiple PointNet modules. PointCNN [3] maps shape vertices to a canonical space where their order is preserved and therefore allows the application of traditional convolutional operators on them. Many other solutions have also been proposed with the aim of extending convolutional filters on point clouds either in the spatial [4], [5], [6], [7] or in the spectral domain [8], [9], [10].

Self-supervised learning has recently achieved large attention in the 2D computer vision community. It deals with originally unlabeled data for which a supervised signal is obtained by first hiding part of the available information and then trying to recover it. This procedure is generally indicated as *pretext* task and possible examples are image completion [11], colorization [12], [13], patch reordering [14], [15] and rotation recognition [16]. Solving the pretext task allows to capture high-level semantic knowledge from the data so that the learned representation can be transferred to other *downstream* tasks as a powerful warm-up initialization. Self-supervision has shown to be relevant also to describe 3D structures. Recent works proposed autoencoder-based approaches to reconstruct 3D point clouds [17], [18] and methods to deform a 2D grid onto the underlying 3D object surface [19]. In [20] point clouds are split into a front and a back half from several angles and a model is trained to predict one from the other. In [21] a network verifies whether two randomly sampled parts from the dataset belong or not to the same object, while [22] proposes to reconstruct point clouds whose parts have been randomly rearranged. Finally, reconstruction, clustering, and self-supervised classification are combined in [23], defining a fully unsupervised multi-task approach for feature learning.

The pretext and downstream stages define a particular case of *Transfer Learning* where unsupervised data is exploited to support supervised learning on a task of interest. In many real world applications, despite unlabeled data may be freely available, their distribution can significantly differ from that of the supervised data at hand, rising the extra issue of how to deal with a *domain shift* for which the transfer procedure may backfire. A similar problem holds also when the unsupervised collection is not an extra source of knowledge, but corresponds instead to the test on which we need to evaluate a supervised model. For instance, when train and test data are respectively drawn from photos and painting or from virtual reality simulators and real world pictures. *Domain Adaptation* literature focuses on this scenario supposing that the unsupervised test data is transductively available at training time. Many adaptive solutions have been proposed in the last years for 2D vision problems involving either feature alignment strategies with dedicated losses [24], [25], [26], ad-hoc network layers [27], [28] or adversarial learning [29], [30]. Several recent works also involve generative style transfer methods [31], [32], reconstruction penalties [33], [34], [35] or feature norms constraints [36]. An even wider and more challenging problem is the one tackled by *Domain Generalization*. In this case, the specific target data are not provided at training time and the goal is that of learning a model robust to any kind of new domain shift that can appear during deployment. Only few works have shown good results in this setting, mainly considering feature alignment among multiple data sources when available [33], [34], [37], data augmentation [38], [39], and meta-learning [40], [41]. Most recently, self-supervised learning has also shown promising results in the DA and DG scenarios [42], [43].

Our approach connects all the described frameworks in a novel way. Instead of using one [20], [22] or multiple [21], [23] 3D self-supervised tasks as pretext, we consider self-supervision as an auxiliary objective to be optimized jointly with the supervised one in a multi-task model. Specifically, we choose to define and solve 3D puzzles while learning to classify or segment 3D shapes. We focus on real word point clouds which may be severely cluttered with noise and background points [44]. Our analysis largely extends that recently presented in [45] which has just started to investigate DA for 3D point clouds and does not tackle DG nor TL and other challenging settings like semi-supervised learning.

III. Method

A. The Intuition

At the basis of our work there is the intuition that 3D point cloud understanding can still be extremely challenging even

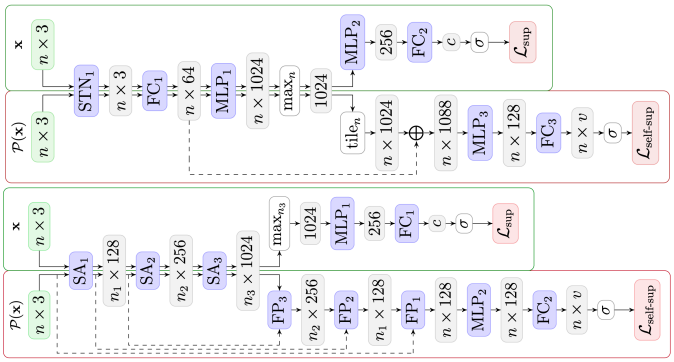


Fig. 2. Our multi-task architecture with PN (top) and PN++ (bottom) backbone used for shape classification. We refer the interested reader respectively to [1] and [2] for the details of each component. Color scheme: green = input data, blue = parametric layer, white = non-parametric layer, grey = output features, red = loss function.

when supervised knowledge is provided. This becomes particularly true when moving from synthetic to real-world data and tasks. Due to their un-ordered nature, how to properly exploit local neighbours and at the same time taking into account the global structure of the 3D shape is quite challenging. Self-supervised learning is helpful in this respect: a simple task like solving a 3D puzzle leverages on the spatial co-location of shape parts and exploits reliable knowledge on relative point positions both at global and local level. Thus, while learning to solve a 3D puzzle we gain useful knowledge that can support recognition at different scales (whole object and parts). We design our learning model as a multi-task deep network where a main supervised task and the self-supervised puzzle task jointly learn a shared data representation.

B. A More Formal Definition

Let us assume to observe data $S = \{(\mathbf{x}_i, \mathbf{y}_i)\}_{i=1}^N$ where each sample $\mathbf{x}_i = \{\mathbf{p}_1^i, \mathbf{p}_2^i, \ldots, \mathbf{p}_K^i\}$ is a un-ordered set of K 3D points $\mathbf{p}_k^i \in \mathbb{R}^3$ described by their Euclidean coordinates. The corresponding label $\mathbf{y}_i$ depends on the specific task at hand. In case of *shape classification*, $\mathbf{y}_i = y_i \in \{1, \ldots, C\}$ is a scalar denoting one out of C object categories. For *part segmentation*, $\mathbf{y}_i$ is a K-dimensional vector with each component $y_{ik} \in \{1, \ldots, Q\}$ where Q is the number of object parts. In the following we will refer either to classification or part segmentation as our *main task* and we will describe how each of them can be jointly learned with the auxiliary self-supervised task of 3D puzzle solving. Our multi-task model combines two parametric non-linear functions: $\mathbf{\Phi}_{\theta_f,\theta_m}$ and $\mathbf{\Psi}_{\theta_f,\theta_p}$, where the subscripts of the parameters θ refer respectively to the feature extraction (f), main task (m), and puzzle solution (p) modules of our deep network. The feature encoder is shared between the two functions and is in charge of summarizing the local and global geometric information from the input point cloud to a richer latent space. For each sample $\mathbf{x}$ that enters the network, $\mathbf{\Phi}_{\theta_f,\theta_m}(\mathbf{x})$ is its final output. The loss function $\mathcal{L}_m(\mathbf{\Phi}_{\theta_f,\theta_m}(\mathbf{x}), \mathbf{y})$ measures the prediction error on the main task by comparing the network output with the corresponding ground truth label.

The auxiliary function $\mathbf{\Psi}$ deals with a *puzzled* variant $\tilde{\mathbf{x}} = \mathcal{P}(\mathbf{x})$ of the original input point cloud. To get it, we start from $\mathbf{x}$, scale it to unit cube and split each axis into l equal lengths intervals forming l^3 voxels which are labeled according to their original position. Each vertex contained inside a voxel inherits its label. Finally, all the voxels are randomly swapped, producing a new shuffled point cloud. We indicate with $\tilde{S} = \{(\tilde{\mathbf{x}}_i, \tilde{\mathbf{y}}_i)\}_{i=1}^N$ the obtained puzzled samples where the voxel position label for each point is $y_{ik} \in \{1, \ldots, l^3\}$. Once these new displaced data are encoded in the feature latent space, a second network head focuses on solving the 3D puzzle problem. It minimizes the auxiliary loss that measures the reordering error $\mathcal{L}_p(\mathbf{\Psi}_{\theta_f,\theta_p}(\tilde{\mathbf{x}}), \tilde{\mathbf{y}})$ in terms of the difference between the assigned voxel label and the correct one per point. Overall we train the network to obtain the optimal model through

$$\arg\min_{\theta_f,\theta_m,\theta_p} \sum_{i=1}^{N} \mathcal{L}_m(\mathbf{\Phi}_{\theta_f,\theta_m}(\mathbf{x}_i), \mathbf{y}_i) + \alpha\, \mathcal{L}_p(\mathbf{\Psi}_{\theta_f,\theta_p}(\tilde{\mathbf{x}}_i), \tilde{\mathbf{y}}_i), \quad (1)$$

where both $\mathcal{L}_m$ and $\mathcal{L}_p$ are cross-entropy losses. Note that, while the first loss deals only with original samples, the second involve both original and puzzled samples, given the random nature of the voxel shuffling procedure.

C. Hyper-parameters and Implementation Choices

The described learning problem has one main hyper-parameter α, which weights the self-supervised loss and balances its importance with respect to the main supervised task. Since we exploit self-supervision as an auxiliary objective we reasonably assign less importance to it with respect to the main task and set $\alpha = 0.6$ for all our analysis. A further parameter of the problem is the axis quantization step used to define the puzzle parts: we set $l = 3$. An ablation analysis on both α and l is provided in sec. IV-E. To realize our model we built over two well known and reliable architectures: PointNet (PN) and PointNet++ (PN++) by extending their structure with the inclusion of a new ending branch dedicated to 3D puzzle resolution. Figure 2 illustrates the corresponding architectures for our multi-task approach.

D. Relation to other Puzzle Solvers

Our learning architecture is trained by jointly optimizing both over the 3D puzzle and the main supervised task. This makes our approach different from the recently published work [22] that discusses a 3D puzzle task whose self-supervised model is learned in isolation and only in a second phase transferred to a down-stream task. On the other hand, a related work exploiting supervised and self-supervised multi-task learning is [42] where, however, the puzzle task is defined in 2D with a completely different logic with respect to that proposed in our work. Specifically in [42] the whole puzzled sample is described by a single index which identifies the permutation applied on the image patches. The puzzle task is formalized as a classification problem to predict that index. The straightforward implementation of this strategy in 3D fails in capturing local information and, when integrated in the

multi-task model, does not show any advantage with respect to the single-task supervised baseline. Indeed it does not deal properly with the non-Euclidean nature of point clouds and in particular with the case of empty voxels which makes the permutation index classification particularly difficult or even unsolvable. Our model is instead tailored for 3D problems: we assign a label to each object voxel, which is inherited by the vertices of the point cloud contained in that voxel. After shuffling, the puzzle solver performs a per-point voxel label prediction. Since the focus is only on the points and not globally on the voxels, the issue of empty voxels does not affect our approach.

IV. Experiments

A. Settings

We consider several experimental settings involving a source dataset $\mathcal{S}$ divided into two disjoint parts $\mathcal{S}_{\text{train}}$, $\mathcal{S}_{\text{test}}$, a possible extra set of unlabeled data from a different source domain $\mathcal{S}'$ and an unlabeled target domain $\mathcal{T}$, different from both $\mathcal{S}$, $\mathcal{S}'$.
Single Domain (SD): the whole set of annotated samples from $\mathcal{S}_{\text{train}}$ are available for supervised learning. We test on the portion $\mathcal{S}_{\text{test}}$ of the same original dataset.
Few-Shot (FS): it considers the case of limited training samples. We reduce the cardinality of $\mathcal{S}_{\text{train}}$ at different percentage scales and we evaluate on $\mathcal{S}_{\text{test}}$.
Semi-Supervised (SS): this setting is analogous to the previous one but the percentage of samples which is not included in $\mathcal{S}_{\text{train}}$ can still be used as unlabeled data during training.
Transfer Learning (TL): besides the annotated data from $\mathcal{S}$, a further set of unlabeled samples from a different domain $\mathcal{S}'$ is available at training time. In this case, when running the multi-task approach, we feed the self-supervised task with the extra unlabeled samples while the supervised data is only used for the main task. The final evaluation is performed on $\mathcal{S}_{\text{test}}$.
Domain Generalization (DG): this is analogous to SD but in the evaluation phase the performance is computed on the target collection $\mathcal{T}$ (belonging to a different domain).
Domain Adaptation (DA): both the supervised data $\mathcal{S}$ and the unsupervised target data $\mathcal{T}$ are available at training time and enter the self-supervised part of our multi-task method. Instead the main task is learned only on the supervised $\mathcal{S}$. As standard practice, the final model is evaluated on $\mathcal{T}$ data.

B. Datasets

Synthetic data from ModelNet ModelNet40 [46] contains 12311 3D CAD models from 40 man-made object categories. We use the official dataset split, consisting of 9843 train and 2468 test shapes. By following [1], from each CAD model we extract a point cloud by uniformly sampling 2048 vertices from the faces of the synthetic mesh. Each point cloud is then centered in the origin and scaled to fit in the unit sphere.

Synthetic data from ShapeNet ShapeNet is one of the largest repositories of annotated 3D models. We use two of its variants depending on the annotations required by the main task. ShapeNetCore [47] contains 51300 clean 3D CAD models from 55 different classes, each annotated with the object category. ShapeNetPart [48] contains 16881 3D shapes from 16 different categories. We use the official dataset split containing 12137 train, 1870 validation, and 2874 test shapes. Each shape is annotated with 2 to 6 parts, for a total of 50 distinct parts among all categories. To reduce the high variability of vertex density across different categories, 2048 vertices are randomly sampled from each shape. Also in this case each point cloud is normalized to fit in the unit sphere.

Real world data from ScanObjectNN ScanObjectNN [44] is a recent dataset containing 2902 3D scans of real-world objects from 15 categories (mostly furniture) originated from ScanNet [49]. Real-world 3D scans are much more challenging than CAD models due to the presence of acquisition artifacts such as vertex noise, non-uniform vertex density, missing parts, and occlusions. Moreover, real-world data contains background vertices, which are absent in the synthetic models of ModelNet and ShapeNet. Several variants of the ScanObjectNN dataset are provided. The vanilla version `OBJ_ONLY` is the closest to synthetic datasets since it contains only foreground vertices. `OBJ_BG` contains the same shapes but with the addition of background vertices. Finally, there are the most challenging cases where 50% Translation, Rotation around the gravity axis and Scaling along each axis are applied to 3D scans: `PB_T50_RS_BG` and `PB_T50_RS`, respectively with and without background. Interestingly, 11 among the 15 categories of ScanObjectNN overlap with the those in ModelNet40: we focus on them to investigate the domain shift between the two datasets in terms of DG and DA experiments[1].
Real world data from ScanNet ScanNet [49] dataset contains 3D reconstructions of real-world indoor scenes. The annotated object instances extracted from the scenes miss parts and are occluded by the surroundings. For our experiments we followed [45] considering a subset of 10 object classes shared with ModelNet and ShapeNet.

C. Classification Results

We dedicate the first part of our experimental analysis to the main task of shape classification. The goal is to predict the object category of the observed point cloud from a set of C classes. We evaluate the performance in terms of overall accuracy. All reported results are averaged over three runs.

Training details and Baseline Throughout all the classification experiments we use the following parameters. For the PN backbone: batch size of 64, Adam optimizer, and an initial learning rate of 0.001 decreased by a factor of 4 every 20 epochs. For PN++: batch size of 64, SGD optimizer with momentum 0.9, and an initial learning rate of 0.01, decreased by a factor of 2 every 20 epochs. Data augmentation is performed following verbatim the procedure proposed by [1], *i.e.* random vertex jittering drawn from $\mathcal{N}(0, 0.01)$, and

[1]Our cross-domain setting is different from that in [44] where the source model is trained on ModelNet40 (M) with all its 40 classes and predicts a 40-dim score vector. Only 11 of the classes in ScanObjectNN (SONN) overlap with those of M, thus a SONN sample whose prediction is one of the 29 classes missing in the target is considered as misclassified. Since the class mapping is known a priori, this evaluation procedure sounds odd, moreover we empirically verified that it decreases the baseline performance.

TABLE I

SHAPE CLASSIFICATION ACCURACY (%) OF OUR MULTI-TASK APPROACH WITH RESPECT TO THE MAIN CLASSIFICATION BASELINE ($\alpha = 0$) IMPLEMENTED ON TWO DIFFERENT BACKBONES (PN [1], PN++ [2]). IN THE TRANSFER LEARNING (TL) SETTING, EXTRA UNSUPERVISED DATA SOURCES ARE INTEGRATED IN THE LEARNING PROCESS AS INPUT TO THE SELF-SUPERVISED TASK (SHAPENET FOR MODELNET40 AND MODELNET40 FOR SCANOBJECTNN). SUFFIX BG INDICATES THAT THE POINT CLOUDS CONTAINS BACKGROUND VERTICES.

Backbone	Method	ModelNet40	ScanObjectNN				
			OBJ_ONLY	OBJ_BG	PB_T50_RS	PB_T50_RS_BG	AVG
PN	Baseline	88.65	75.22	70.22	71.37	62.56	69.84
	Our SD	89.71	75.04	71.26	73.39	65.20	71.22
	Our TL	**90.72**	77.45	71.26	73.49	65.61	**71.95**
PN++	Baseline	91.93	84.17	83.99	78.66	77.90	81.18
	Our SD	**92.10**	85.89	83.13	79.22	78.00	81.56
	Our TL	91.58	84.68	84.33	80.46	79.08	**82.14**

TABLE II

SHAPE CLASSIFICATION ACCURACY (%) WHEN THE TRAINING SET CONTAINS A LIMITED AMOUNT OF ANNOTATED DATA.

Backbone	Method	ModelNet40			
		20%	40%	60%	AVG
PN	Baseline	82.94	85.49	87.11	85.18
	Our FS	82.09	87.03	88.13	85.75
	Our SS	83.06	87.27	88.57	**86.30**
PN++	Baseline	85.37	88.25	89.63	87.75
	Our FS	86.35	89.59	89.18	**88.37**
	Our SS	86.02	88.41	89.83	88.09

random rotation around the shape elongation axis. In our analysis we use as reference the standard supervised baseline: the naïve variant of our method obtained by setting $\alpha = 0$ which simply corresponds to turning off the puzzle solver.

Single Domain We start by evaluating the performance of our approach in the most classical single domain scenario and present the results in Table I. We consider as testbed ModelNet40 (C=40) and ScanObjectNN (C=16) and observe that in both cases our multi-task approach consistently outperforms the baseline regardless of the used backbone. These results indicate that, by simply solving the auxiliary self-supervised task, the learned representation is better able to capture the object semantics and provides further discriminative information to the final classifier.

Transfer Learning We also perform two TL experiments considering the availability of an extra unsupervised source $\mathcal{S}'$. The first one has $\mathcal{S}$ = ModelNet40 and $\mathcal{S}'$ = (5K samples from) ShapeNetCore and aims to analyze knowledge transfer among two different synthetic domains. The second one considers $\mathcal{S}$ = ScanObjectNN and $\mathcal{S}'$ = (4K samples from) ModelNet40 and focuses on knowledge transfer from synthetic to real point clouds. The cardinality of $\mathcal{S}'$ was chosen to have a good balance between unsupervised ($\mathcal{S}'$) and annotated ($\mathcal{S}$) data. Results are reported in the third and sixth row of Table I. Overall we observe a further improvement up to 1 pp with respect to the previous results without transfer, with the only exception of ModelNet40 and OBJ_ONLY when using the PN++ backbone. Here the extra synthetic information does not seem to provide useful cues. Differently, for PN the advantage is always visible, indicating that also the backbone choice has a role in the transfer process.

As already mentioned in Section III-D, we highlight that TL and self-supervision have been combined in previous work but with a setting different from ours. In [22] a two-stage pipeline is proposed: first a 3D puzzle solver is learned in an unsupervised manner on the whole ShapeNetCore dataset (> 50k models), than the obtained weights are used to initialize a supervised model. On ModelNet40 this pipeline reaches an accuracy of 92.4%, a negligible gain over the 92.2% accuracy of the baseline with random initialization. Considering the amount of extra unsupervised data used and the different backbone (DGCNN [7]), our 92.1% obtained without extra information appears upstanding. Another interesting comparison can be done with the recently proposed multi-task method [44] which combines classification and segmentation as a possible strategy to deal with real world point clouds. This approach obtains an accuracy of 80.2% on PB_T50_RS_BG over a baseline of 77.9%, but requires an additional costly annotation phase (foreground/background mask) being fully supervised. In contrast, our approach reaches 79.08% accuracy without using any extra label, confirming the effectiveness of the auxiliary self-supervised task.

TABLE III

SHAPE CLASSIFICATION ACCURACY (%) WHEN TRAINING AND TESTING IS DONE ON DIFFERENT DOMAINS (DG). IF THE UNLABELED TARGET DATA IS PROVIDED AT TRAINING TIME (DA), OUR MULTI-TASK IS ABLE TO ADAPT AND REDUCE THE DOMAIN GAP

Domain Generalization and Adaptation							
Method		ModelNet40 →					PB_T50_RS_BG →
		OBJ_ONLY	OBJ_BG	PB_T50_RS	PB_T50_RS_BG	AVG	ModelNet40
PointDAN [45]		56.42	44.84	48.99	34.39	46.16	54.66
PN	Baseline	54.74	43.58	44.96	34.25	44.38	47.43
	Our DG	54.53	49.68	45.22	36.28	46.43	39.30
	Our DA	58.53	47.58	46.70	35.85	47.16	51.54
PN++	Baseline	52.49	44.00	44.83	34.29	43.90	47.66
	Our DG	57.47	52.42	52.84	38.65	50.34	52.88
	Our DA	60.4	53.89	54.66	39.63	**52.14**	**56.07**
3DmFV [6]		30.90	24.00	24.90	16.40	24.05	51.50
PointCNN [3]		32.20	29.50	24.60	19.20	26.37	49.20
PointNet [1]		42.30	41.10	31.10	23.20	34.42	50.90
PointNet++ [2]		43.60	37.70	32.00	22.90	34.05	47.40
SpiderCNN [5]		44.20	42.10	30.90	22.20	34.85	46.60
DGCNN[7]		49.30	46.70	36.80	27.20	40.00	54.70

Few-Shot and Semi-Supervised We focused on ModelNet40 for experiments in these settings. The results in Table II show the performance obtained when the labeled training data reduces up to only 20% of the original amount. We can observe that, despite the overall drop in performance, our multi-task approach in the few-shot (FS) setting maintains its advantage with respect to the baseline. When considering also the unlabeled data for the semi-supervised (SS) setting and the PN backbone, we observe a further increase in performance.

Domain Generalization If training and test data are drawn from two very different distributions the model learned on the former usually fails to generalize to the latter. Being able to maintain a good performance in this challenging condition is crucial in all the cases in which obtaining annotated data of the target domain is not possible. We consider the DG setting when training on ModelNet40 and testing on ScanObjectNN and report results in Table III. Our multi-task approach fully learned only on synthetic data shows a significant improvement with respect to the baseline with gains up to 6 and 8 pp in the OBJ_BG and with a still relevant gain of 2 and 4 pp in the most challenging PB_T50_RS_BG, respectively with PN and PN++. We also consider the inverse generalization direction

from PB_T50_RS_BG to ModelNet40. Here the training data is affected by various sources of noise and adding the self-supervised task on PN seems to backfire, increasing the risk of overfitting. On the other way round, PN++ is more reliable and presents a significant gain of 4 percentage points.

Domain Adaptation We investigated whether our multi-task approach could close the domain gap when unlabeled target data are available at training time. The DA results in Table III provide a positive answer showing a further increase in performance over DG.

The very recent PointDAN method [45] tackles point-cloud domain shifts by combining local and global alignment. Local alignment is obtained trough an attention module that takes into consideration the relationship between close nodes, while global alignment is performed through maximum classifier discrepancy [50]. Table III shows that our multi-task approach largely outperforms this solution. Finally, an overall look at the performance of several recent point cloud networks is provided in the bottom part of Table III. The results indicate that our multi-task approach establishes the new state-of-the-art for classification on real world data from synthetic training. Even in the opposite learning direction from real to synthetic, our model combining supervised and self-supervised learning shows promising results: the ability to adapt provides it with a further way to improve over existing references.

Different Auxiliary Tasks Solving 3D puzzles is just one of the possible auxiliary self-supervised tasks. We investigate 3D reconstruction as alternative solution, in two different settings. We start by focusing on the ModelNet-OBJ_ONLY experiment and implementing reconstruction by introducing an FC Decoder to regress the input shape from the PN extracted global features. However, in this way we do not get any significant improvement (54.88) over the baseline (54.74). Indeed, it is crucial to provide the self-supervised task with both local and global information. Only in this way the 3D shape reconstruction module supports adaptation, and produces a relative improvement (56.42) which is still lower than what we get with 3D puzzle (58.53). By integrating both the puzzle and reconstruction tasks together we obtain a further small improvement (59.02).

For a second evaluation we consider the recent preprint [51], which proposed a multi-task model combining classification and Region Reconstruction (RegRec). It also exploits Point-Cloud-Mixup (PCM), a training procedure that involves mixed instances of labeled input data. A comparison on the 10 shared classes of ModelNet (M), ShapeNet (S) and ScanNet (S*) with all the methods based on PN produces the results in Table IV. Besides confirming that our multi-task approach outperforms PointDAN, they also show that the self-supervised reconstruction task is less powerful than our 3D puzzle solver for adaptation. Moreover, the results seem quite unstable as shown by the large standard deviation for the $M \rightarrow S^*$ and $S^* \rightarrow M$ cases. The only advantage is provided by the PCM method for some of the domain pairs. Still, on average the results remain in favour of our method.

TABLE IV
CROSS-DOMAIN ACCURACY (%) ON MODELNET-10 (*M*), SHAPENET-10 (*S*) AND SCANNET-10 (*S**). POINTDAN AND THE METHOD PROPOSED IN THE RECENT PRE-PRINT [51] SHARE THE SAME POINTNET [1] FEATURE EXTRACTOR.

Method	M→S	M→S*	S→M	S→S*	S*→M	S*→S	Avg.
PointDAN [45]	80.2±0.8	45.3±2.0	71.2±3.0	46.9±3.3	59.8±2.3	66.2±4.8	61.6
RegRec [51]	80.0±0.6	46.0±5.7	68.5±4.8	41.7±1.9	63.0±6.7	68.2±1.1	61.2
RegRec + PCM [51]	81.1±1.1	50.3±2.0	54.3±0.3	52.8±2.0	54.0±5.5	69.0±0.9	60.3
Our DA	81.6±0.6	49.7±1.4	73.6±0.5	41.9±0.9	65.9±0.7	68.1±1.6	**63.5**

D. Part Segmentation Results

The second set of our experiments is dedicated to part segmentation, the problem of assigning each vertex to the shape part to which it belongs to. By following [1], the quality of the predicted part segmentation is evaluated in terms of the mean Intersection-over-Union (mIoU) metric. The mIoU of a shape is defined as the average over its Q parts of the IoU between the ground-truth and predicted segmentations of each part. The mIoU of a category is defined as the average of the mIoUs of the shapes it contains.

Training Details Throughout all the part segmentation experiments we use the PointNet Segmentation backbone from [1]. We slightly modify the network architecture introducing a branch for jointly solving the 3D puzzle task, this branch shares most of the initial network layers with the main segmentation one. Our modifications does not increase the original segmentation branch capacity. We used batch size of 64, Adam optimizer, and an initial learning of rate 0.001, decreased by a factor of 2 every 20 epochs. Data augmentation is applied exactly as in our classification experiments.

Single Domain and Transfer Learning Table V shows the part segmentation results obtained by our method on the chair shapes from two challenging subsets of ScanObjectNN in terms of the evaluation metric used in [44]. In the SD setting the introduction of self-supervision does not improve over the baseline accuracy. In the TL setting, by considering as extra source of knowledge $\mathcal{S}'$ the unlabeled chairs from ModelNet40, our approach proves once again its effectiveness.

Few-Shot and Semi-Supervised By following [23] we randomly sample 1% and 5% of the ShapeNetPart train set to evaluate the point features in a semi-supervised setting. The results in Table VI indicate that our multi-task approach, although not improving over the baseline in the few-shot setting, in the semi-supervised setting outperforms the current state of the art in the 1% case and practically matches it in the 5% case. It is interesting to underline that also our best competitor CCD [23] is a multi-task approach that combines clustering and reconstruction with a self-supervised classification obtained by learning on the clustering auto-defined labels. For a more in-depth analysis of our results we plot some visualizations out of our 1% part segmentation experiment for chairs and lamps. Figure 3 shows that our multi-task approach allows a better recognition of the armrests. Indeed the position of these relative small parts may be better learned thanks to the puzzle solution task. A similar conclusion holds for the lamp basis.

Domain Generalization and Adaptation We focus on the domain shift between synthetic and real chairs with

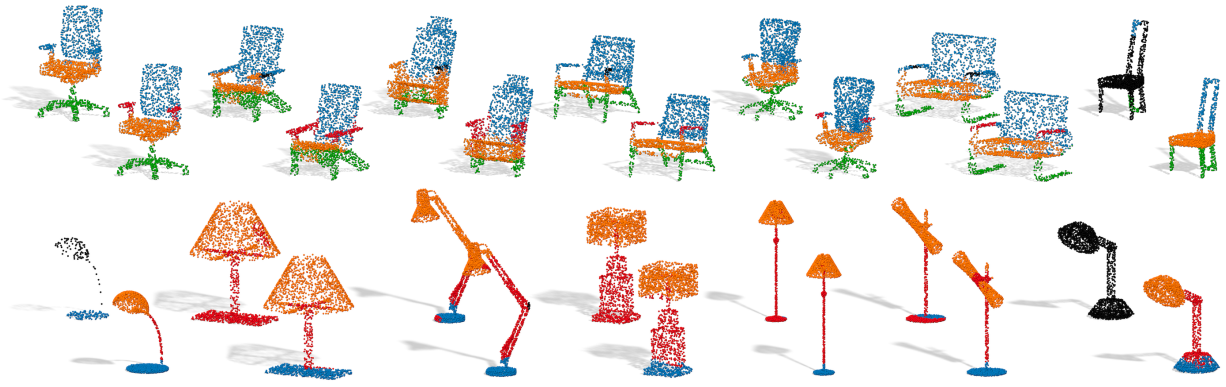
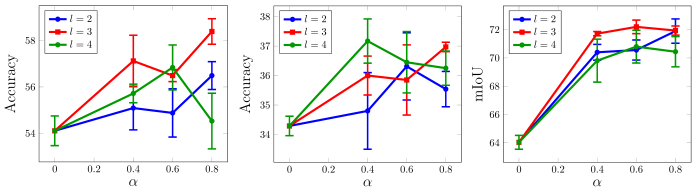


Fig. 3. Part segmentation of chairs and lamps when only 1% of training data are available. Each couple shows the baseline prediction (top left) and our approach (bottom right). The last example show the worst case for the baseline. Black points denotes predictions whose maximum value was not a chair or lamp part.

TABLE V
PART SEGMENTATION OF CHAIRS FROM TWO VARIANTS OF SCANOBJECTNN EVALUATED IN TERMS OF PER PART AVERAGE AND OVERALL ACCURACY (%).

Dataset	Method	Bg	Seat	Back	Base	Arm	Avg.	Overall
OBJ_BG	Baseline	65.14	87.88	89.73	67.16	58.97	73.02	81.62
	Our SD	64.97	87.46	86.27	68.96	57.54	73.04	81.67
	Our TL	69.43	86.59	88.71	72.70	61.37	**75.76**	**82.60**
PB_T50_R_BG	Baseline	82.06	83.71	75.30	54.75	35.11	**66.19**	81.13
	Our SD	82.02	83.50	77.80	51.53	25.50	64.07	81.31
	Our TL	83.08	81.87	79.06	50.71	30.40	64.99	**81.82**



(DA) ModelNet40 →OBJ_ONLY | (DA) ModelNet40 →PB_T50_RS_BG | (SS) ShapeNetPart 1%

Fig. 4. Parameter (α, l) evaluation in case of cross domain Shape Classification (left, center), and 1% semi-supervised Part Segmentation (right). The case $\alpha = 0$ corresponds to the baseline regardless of the l value. Each experiment is repeated three times and we report here the average results with their standard deviation.

ShapeNetPart as source and ScanObjectNN as target. We highlight that ScanObjectNN has some part annotation issue confirmed by the authors through personal communications, thus we prefer to use only the OBJ_BG provided subdomain, neglecting the background which is absent in ShapenNetPart. Table VII collects the results confirming also in this case the advantage of our multi-task approach over the supervised learning baseline.

E. Ablation Analysis

As indicated in sec. III our approach has two main hyper-parameters. One related to the learning model (α) and the other needed to define the puzzled data (l). We analyze how much our method is sensitive to their variation by considering different values of $\alpha = \{0.4, 0.6, 0.8\}$ and three different puzzle decomposition settings with $l = \{2, 3, 4\}$. In particular we focus on the DA shape classification with PN backbone on ModelNet40 as source and OBJ_ONLY, PB_T50_RS_BG as target. We also consider part segmentation SS with PN segmentation backbone on 1% of ShapeNetPart.

TABLE VI
ACCURACY (MIOU) FOR PART SEGMENTATION ON SHAPENETPART WITH LIMITED ANNOTATIONS.

Method	1%	5%
SO-Net [17]	64.00	69.00
PointCapsNet [18]	67.00	70.00
CCD [23]	68.20	**77.70**
Baseline	64.52	75.75
Our FS	64.49	75.07
Our SS	**71.95**	77.42

TABLE VII
PER PART AND AVERAGE ACCURACY (%) OF CHAIR SEGMENTATION. WE USE THE SAME METRIC OF TABLE V.

Part Segmentation - DA/DG					
ShapeNetPart → OBJ_BG					
Method	Seat	Back	Base	Arm	Avg.
Baseline	67.85	45.60	84.89	14.87	53.30
our DG	71.80	42.61	84.57	21.48	55.11
our DA	65.70	49.11	85.91	21.40	**55.53**

Figure 4 confirms that on average $l = 3$ is the best choice: intermediate between the minimum decomposition of an object into $2^3 = 8$ big parts and the very fine decomposition into $4^3 = 64$ parts. On the other hand, for the puzzle loss weight, our standard choice $\alpha = 0.6$ can be improved by passing to $\alpha = 0.8$, which indicates that it is possible to get an even further advantage with an ad hoc finetuned choice of the parameters. Overall there is a trade off between the two considered hyper-parameters: the auxiliary task can have a low weight as far as the number of puzzle part is high and vice-versa. This discussion holds both for DA shape classification and part segmentation results. In the latter case the difficulty of dealing with $l = 4$ is even more evident.

V. CONCLUSIONS

In this work we investigated how to deal with 3D labeled and unlabeled data possibly coming from different domains and with data annotation scarcity. To tackle these real world challenges we proposed a multi-task approach that combines

supervised and self-supervised learning and showed with an extensive evaluation that it produces the new state-of-the-art for both shape classification and part segmentation on cross-domain and few-shot real world settings. We see this work as a first exciting step towards a new family of methods better able to generalize and adapt to novel testing conditions for 3D point clouds.

ACKNOWLEDGMENT

This work was partially supported by the CHIST-ERA BURG Project (TT,AA).

REFERENCES

[1] C. R. Qi, H. Su, K. Mo, and L. J. Guibas, "PointNet: Deep Learning on Point Sets for 3D Classification and Segmentation," in *CVPR*, 2017.
[2] C. R. Qi, L. Yi, H. Su, and L. J. Guibas, "PointNet++: Deep Hierarchical Feature Learning on Point Sets in a Metric Space," in *NIPS*, 2017.
[3] Y. Li, R. Bu, M. Sun, W. Wu, X. Di, and B. Chen, "Pointcnn: Convolution on x-transformed points," in *NIPS*, 2018.
[4] F. Monti, D. Boscaini, J. Masci, E. Rodolà, J. Svoboda, and M. M. Bronstein, "Geometric deep learning on graphs and manifolds using mixture model cnns," in *CVPR*, 2017.
[5] Y. Xu, T. Fan, M. Xu, L. Zeng, and Y. Qiao, "Spidercnn: Deep learning on point sets with parameterized convolutional filters," in *ECCV*, 2018.
[6] Y. Ben-Shabat, M. Lindenbaum, and A. Fischer, "3dmfv: Three-dimensional point cloud classification in real-time using convolutional neural networks," *IEEE Robotics and Automation Letters*, vol. 3, no. 4, pp. 3145–3152, 2018.
[7] Y. Wang, Y. Sun, Z. Liu, S. E. Sarma, M. M. Bronstein, and J. M. Solomon, "Dynamic Graph CNN for Learning on Point Clouds," *ACM Trans. Graph.*, 2019.
[8] D. Boscaini, J. Masci, S. Melzi, M. M. Bronstein, U. Castellani, and P. Vandergheynst, "Learning class-specific descriptors for deformable shapes using localized spectral convolutional networks," *Comput. Graph. Forum*, vol. 34, pp. 13–23, 2015.
[9] C. Wang, B. Samari, and K. Siddiqi, "Local spectral graph convolution for point set feature learning," in *ECCV*, 2018.
[10] L. Yi, H. Su, X. Guo, and L. J. Guibas, "Syncspeccnn: Synchronized spectral cnn for 3d shape segmentation," in *CVPR*, 2017.
[11] D. Pathak, P. Krähenbühl, J. Donahue, T. Darrell, and A. Efros, "Context encoders: Feature learning by inpainting," in *CVPR*, 2016.
[12] R. Zhang, P. Isola, and A. A. Efros, "Colorful image colorization," in *ECCV*, 2016.
[13] G. Larsson, M. Maire, and G. Shakhnarovich, "Colorization as a proxy task for visual understanding," in *CVPR*, 2017.
[14] C. Doersch, A. Gupta, and A. A. Efros, "Unsupervised visual representation learning by context prediction," in *ICCV*, 2015.
[15] M. Noroozi and P. Favaro, "Unsupervised learning of visual representations by solving jigsaw puzzles," in *ECCV*, 2016.
[16] S. Gidaris, P. Singh, and N. Komodakis, "Unsupervised representation learning by predicting image rotations," in *ICLR*, 2018.
[17] J. Li, B. M. Chen, and G. Hee Lee, "So-net: Self-organizing network for point cloud analysis," in *CVPR*, 2018.
[18] Y. Zhao, T. Birdal, H. Deng, and F. Tombari, "3d point capsule networks," in *CVPR*, 2019.
[19] Y. Yang, C. Feng, Y. Shen, and D. Tian, "Foldingnet: Point cloud auto-encoder via deep grid deformation," in *CVPR*, 2018.
[20] Z. Han, X. Wang, Y.-S. Liu, and M. Zwicker, "Multi-angle point cloud-vae: Unsupervised feature learning for 3d point clouds from multiple angles by joint self-reconstruction and half-to-half prediction," in *ICCV*, 2019.
[21] L. Zhang and Z. Zhu, "Unsupervised feature learning for point cloud by contrasting and clustering with graph convolutional neural network," in *CVPR Workshop*, 2019.
[22] J. Sauder and B. Sievers, "Self-supervised deep learning on point clouds by reconstructing space," in *NIPS*, 2019.
[23] K. Hassani and M. Haley, "Unsupervised multi-task feature learning on point clouds," in *ICCV*, 2019.
[24] M. Long, H. Zhu, J. Wang, and M. I. Jordan, "Deep transfer learning with joint adaptation networks," in *ICML*, 2017.
[25] B. Sun and K. Saenko, "Deep coral: Correlation alignment for deep domain adaptation," in *ECCV Workshops*, 2016.
[26] S. Ben-David, J. Blitzer, K. Crammer, A. Kulesza, F. Pereira, and J. Vaughan, "A theory of learning from different domains," *Machine Learning*, vol. 79, pp. 151–175, 2010.
[27] M. Mancini, L. Porzi, S. Rota Bulò, B. Caputo, and E. Ricci, "Boosting domain adaptation by discovering latent domains," in *CVPR*, 2018.
[28] F. M. Carlucci, L. Porzi, B. Caputo, E. Ricci, and S. Rota Bulò, "Autodial: Automatic domain alignment layers," in *ICCV*, 2017.
[29] Y. Ganin, E. Ustinova, H. Ajakan, P. Germain, H. Larochelle, F. Laviolette *et al.*, "Domain-adversarial training of neural networks," *J. Mach. Learn. Res.*, vol. 17, no. 1, pp. 2096–2030, 2016.
[30] E. Tzeng, J. Hoffman, T. Darrell, and K. Saenko, "Adversarial discriminative domain adaptation," in *CVPR*, 2017.
[31] P. Russo, F. M. Carlucci, T. Tommasi, and B. Caputo, "From source to target and back: symmetric bi-directional adaptive gan," in *CVPR*, 2018.
[32] J. Hoffman, E. Tzeng, T. Park, J.-Y. Zhu, P. Isola, K. Saenko *et al.*, "CyCADA: Cycle-consistent adversarial domain adaptation," in *ICML*, 2018.
[33] M. Ghifary, W. B. Kleijn, M. Zhang, and D. Balduzzi, "Domain generalization for object recognition with multi-task autoencoders," in *ICCV*, 2015.
[34] H. Li, S. Jialin Pan, S. Wang, and A. C. Kot, "Domain generalization with adversarial feature learning," in *CVPR*, 2018.
[35] K. Bousmalis, G. Trigeorgis, N. Silberman, D. Krishnan, and D. Erhan, "Domain Separation Networks," in *NIPS*, 2016.
[36] R. Xu, G. Li, J. Yang, and L. Lin, "Larger norm more transferable: An adaptive feature norm approach for unsupervised domain adaptation," in *ICCV*, 2019.
[37] F. M. Carlucci, P. Russo, T. Tommasi, and B. Caputo, "Hallucinating agnostic images to generalize across domains," in *TASK-CV Workshop at ICCV*, 2019.
[38] S. Shankar, V. Piratla, S. Chakrabarti, S. Chaudhuri, P. Jyothi, and S. Sarawagi, "Generalizing across domains via cross-gradient training," in *ICLR*, 2018.
[39] R. Volpi, H. Namkoong, O. Sener, J. Duchi, V. Murino, and S. Savarese, "Generalizing to unseen domains via adversarial data augmentation," in *NIPS*, 2018.
[40] D. Li, Y. Yang, Y. Song, and T. M. Hospedales, "Learning to generalize: Meta-learning for domain generalization," in *AAAI*, 2018.
[41] D. Li, J. Zhang, Y. Yang, C. Liu, Y.-Z. Song, and T. M. Hospedales, "Episodic training for domain generalization," in *ICCV*, 2019.
[42] F. M. Carlucci, A. D'Innocente, S. Bucci, B. Caputo, and T. Tommasi, "Domain generalization by solving jigsaw puzzles," in *CVPR*, 2019.
[43] J. Xu, L. Xiao, and A. M. López, "Self-supervised domain adaptation for computer vision tasks," *Preprint Arxiv:1907.10915*, 2019.
[44] M. A. Uy, Q.-H. Pham, B.-S. Hua, D. T. Nguyen, and S.-K. Yeung, "Revisiting Point Cloud Classification: A New Benchmark Dataset and Classification Model on Real-World Data," in *ICCV*, 2019.
[45] C. Qin, H. You, L. Wang, C.-C. J. Kuo, and Y. Fu, "Pointdan: A multi-scale 3d domain adaption network for point cloud representation," in *NIPS*, 2019.
[46] Z. Wu, S. Song, A. Khosla, F. Yu, L. Zhang, X. Tang *et al.*, "3D ShapeNets: A Deep Representation for Volumetric Shapes," in *CVPR*, 2015.
[47] A. X. Chang, T. Funkhouser, L. Guibas, P. Hanrahan, Q. Huang, Z. Li *et al.*, "ShapeNet: An Information-Rich 3D Model Repository," in *Preprint ArXiv:1512.03012*, 2015.
[48] L. Yi, V. G. Kim, D. Ceylan, I.-C. Shen, M. Yan, H. Su *et al.*, "A Scalable Active Framework for Region Annotation in 3D Shape Collections," in *SIGGRAPH Asia*, 2016.
[49] A. Dai, A. X. Chang, M. Savva, M. Halber, T. Funkhouser, and M. Nießner, "Scannet: Richly-annotated 3d reconstructions of indoor scenes," in *CVPR*, 2017.
[50] K. Saito, K. Watanabe, Y. Ushiku, and T. Harada, "Maximum classifier discrepancy for unsupervised domain adaptation," *CVPR*, 2018.
[51] I. Achituve, H. Maron, and G. Chechik, "Self-supervised learning for domain adaptation on point-clouds," *Preprint ArXiv:2003.12641v1*, 2020.