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

Antonio Alliegro[1], Davide Boscaini[2], and Tatiana Tommasi[1]

[1]Politecnico di Torino, Italy
[2]Fondazione Bruno Kessler, Trento, Italy
{antonio.alliegro,tatiana.tommasi}@polito.it, d.boscaini@fbk.eu

Abstract. Point cloud processing and 3D shape understanding are very 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, where the amount of annotated data may be limited and integrating new sources of knowledge becomes crucial to support autonomous learning. Here we consider several possible 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 for 3D shape classification and part segmentation.

1 Introduction

Artificial intelligence made great strides in recent years with terrific results in the area of computer vision. This naturally reflects also 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 for many reasons, ranging from the inherent difficulty of dealing with a three-dimensional space, as well as time and domain variations which makes it difficult 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 large entities such as 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

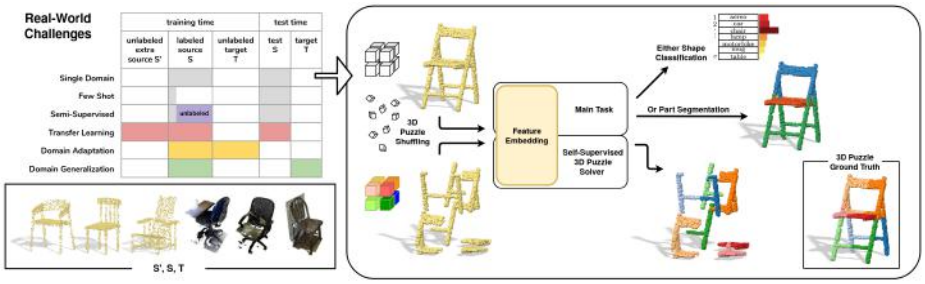


Fig. 1. Overview of the considered real world challenges and of the proposed multi-task approach. We deal with different domains covering real and synthetic 3D point clouds as well as several learning settings across domains and scarce annotations

properties make typical convolutional neural networks (CNN) unsuitable for point clouds. Possible solutions consists 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:

1. we design a new research landscape by *tackling at once several aspects of cross-domain adaptive 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 2 for an overview). We investigate how to follow this trend for 3D tasks.

2. we propose a *new multi-task end-to-end deep learning model for point clouds that combines supervised and self-supervised learning* (see Fig. 1). Specifically we define a deep architecture that solves 3D puzzles while jointly training a main recognition 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.

3. we present extensive experiments across three 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.

2 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 roots back to PointNet [31], a MLP architecture combining symmetry and spatial transform functions to learn point features which are then aggregated into global shape representations. PointNet++ [32] extended the previous model by hierarchically combining multiple PointNet modules. PointCNN [24] 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 [28,47,3,43] or in the spectral domain [4,42,50].

Self-supervised learning is a framework recently achieving 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 [30], colorization [54,18], patch reordering [11,29] and rotation recognition [14]. The solution of the pretext task captures 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 [22,55] and methods to deform a 2D grid onto the underlying 3D object surface [48]. In [15] 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. A network to verify whether two randomly sampled parts from the dataset belong or not to the same object is presented in [53], while [36] proposes to reconstruct point clouds whose parts have been randomly rearranged. Finally, reconstruction, clustering, and self-supervised classification are combined together in [16], 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 [25,26,38,2], ad-hoc network layers [27,7] or adversarial learning [12,39]. Several recent works also involve

generative style transfer methods [34,17], reconstruction penalties [13,21,5] or feature norms constraints [46]. 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 [13,21,23,8], data augmentation [37,41], and meta-learning [19,20]. Most recently, self-supervised learning has also shown promising results in the DA and DG scenarios [6,45,51].

Our approach connects all the described frameworks in a novel way. Instead of using one [15,36] or multiple [53,16] 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 [40]. Our analysis largely extends that recently presented in [33] 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.

3 Method

The Intuition. At the basis of our work there is the intuition that 3D point cloud understanding can still be extremely challenging even 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 recognition task and the self-supervised puzzle task jointly learn a shared data representation.

More Formally. Let us assume to observe data $S = \{(\mathbf{x}_i, \mathbf{y}_i)\}_{i=1}^{N}$ where each sample $\mathbf{x}_i = \{\mathbf{p}_1^i, \mathbf{p}_2^i, \dots, \mathbf{p}_K^i\}$ 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*, $\mathbf{y}_i = y_i \in \{1, \dots, C\}$ is a scalar denoting one out of C object categories. For *part segmentation*, instead, $\mathbf{y}_i$ is a K-dimensional vector with each component $y_{ik} \in \{1, \dots, Q\}$ 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 can be described as the combination of two parametric non-linear functions: $\mathbf{\Phi}_{\theta_f,\theta_m}$ and $\mathbf{\Psi}_{\theta_f,\theta_p}$, where the subscripts of the parameters

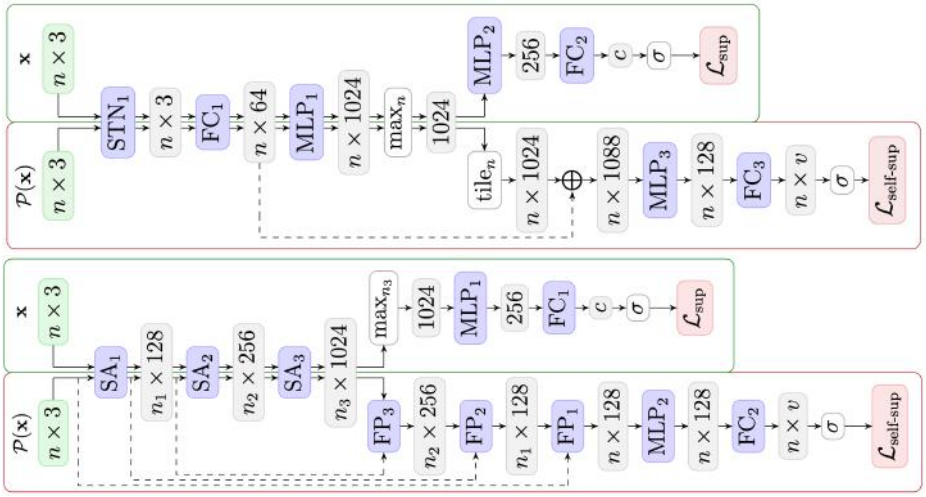


Fig. 2. Our multi-task architecture with PN (top) and PN++ (bottom) backbone used for shape classification. We refer the interested reader respectively to [31] and [32] for the details of each component. Color scheme: green = input data, blue = parametric layer, white = non-parametric layer, grey = output features, red = loss function

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 the output of the feature extractor and final fully connected part of the network. It is finally compared to its corresponding ground-truth label through the loss function $\mathcal{L}_m(\mathbf{\Phi}_{\theta_f,\theta_m}(\mathbf{x}), \mathbf{y})$ which measures the prediction error on the main task.

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 by minimizing 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 difference between the assigned voxel label and the correct one per point. Overall we train the network to obtain the optimal model through

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

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.

Hyper-parameters and Implementation Choices. The described learning problem has one main hyper-parameters 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. 4.5. 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. By investigating both of these backbones we can highlight the effect of the context information learned by the puzzle to different ways of dealing with point clouds. Indeed the first architecture basically learn on each point independently and only accumulates the final features, while the second follows a multi-scale strategy with a heuristic point grouping at separate layers.

4 Experiments

4.1 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 $\mathcal{S}$ and $\mathcal{S}'$. We list them below:

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 setting is analogous to SD but differs in the evaluation phase, where 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.

4.2 Datasets

Synthetic data from ModelNet: ModelNet40 [44] 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 [31], 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 variants of it depending on the annotations required by the main task we aim to solve. ShapeNetCore [9] contains 51300 clean 3D CAD models from 55 different classes, each annotated with the object category. ShapeNetPart [49] 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 [40] is a recent dataset containing 2902 3D scans of real-world objects from 15 categories (mostly furniture) originated from ScanNet [10]. 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, making it suitable for investigating the domain shift between the two datasets in terms of Domain Generalization (DG) and Domain Adaptation (DA) experiments.

4.3 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.

Table 1. Shape classification accuracy (%) of our multi-task approach with respect to the main classification baseline ($\alpha = 0$) implemented on two different backbones (PN [31], PN++ [32]). 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
PN	Baseline	88.65	75.22	70.22	71.37	62.56
	Our SD	89.71	75.04	**71.26**	73.39	65.20
	Our TL	**90.72**	**77.45**	**71.26**	**73.49**	**65.61**
PN++	Baseline	91.93	84.17	83.99	78.66	77.90
	Our SD	**92.10**	**85.89**	83.13	79.22	78.00
	Our TL	91.58	84.68	**84.33**	**80.46**	**79.08**

Training details. Throughout all the classification experiments we used the following parameters. When using the PN backbone, we considered a batch size of 64, Adam optimizer, and an initial learning rate of 0.001 decreased by a factor of 4 every 20 epochs. When using PN++ we considered a 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 [31], *i.e.* random vertex jittering drawn from $\mathcal{N}(0, 0.01)$, and random rotation around the shape elongation axis.

Baselines. For all our experiments we use as reference the standard supervised baseline. It is a 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 scenario when a single domain is available and present the results in Table 1. 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. The advantage appears particularly evident on the real world dataset ScanObjectNN with a gain of about 3 percentage points (pp) for the difficult `PB_T50_RS_BG` on PN.

Transfer Learning. We also perform two TL experiments considering the availability of an extra unsupervised source $\mathcal{S}'$. The first one considers $\mathcal{S}$ = 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 targets 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 1. Overall we observe a further improvement up to 1 pp with respect to the previous results without transfer,

Table 2. Shape classification accuracy (%) of our multi-task approach when the training set contains a limited amount of annotated data

Backbone	Method	ModelNet40			
		20%	40%	60%	100%
PN	Baseline	82.94	85.49	87.11	88.65
	Our FS	82.09	87.03	88.13	**89.71**
	Our SS	**83.06**	**87.27**	**88.57**	-
PN++	Baseline	85.37	88.25	89.63	91.93
	Our FS	**86.35**	**89.59**	89.18	**92.10**
	Our SS	86.02	88.41	**89.83**	-

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 information. Differently, for PN the advantage is always visible, indicating that also the backbone choice has a role in the transfer process.

We highlight that, although previous works discussed TL in combination with self-supervised tasks, their settings differs from ours. In [36] 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 [43]), our 92.1% obtained without extra information appears upstanding. Another interesting comparison can be done with the recently proposed multi-task method [40] 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.

Few-Shot and Semi-Supervised. We focused on ModelNet40 for experiments in these settings. The results in Table 2 show the performance obtained when the amount of 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. When training and test data are drawn from two very different distributions the model learned on the former ones 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 3. Our multi-task approach fully learned only on synthetic data shows a

Table 3. Shape classification accuracy (%) of our multi-task 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	ModelNet40
PointDAN [33]		56.42	44.84	48.99	34.39	54.66
PN	Baseline	54.74	43.58	44.96	34.25	47.43
	Our DG	54.53	49.68	45.22	36.28	39.30
	Our DA	58.53	47.58	46.70	35.85	51.54
PN++	Baseline	52.49	44.00	44.83	34.29	47.66
	Our DG	57.47	52.42	52.84	38.65	52.88
	Our DA	**60.4**	**53.89**	**54.66**	**39.63**	**56.07**
3DmFV [3]		30.90	24.00	24.90	16.40	51.50
PointCNN [24]		32.20	29.50	24.60	19.20	49.20
PointNet [31]		42.30	41.10	31.10	23.20	50.90
PointNet++ [32]		43.60	37.70	32.00	22.90	47.40
SpiderCNN [47]		44.20	42.10	30.90	22.20	46.60
DGCNN[43]		49.30	46.70	36.80	27.20	54.70

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 pp.

Domain Adaptation. We also investigated whether our multi-task approach could close the domain gap when unlabeled target data are available at training time. DA results in Table 3 provide a positive answer showing a further increase in performance over the DG results (with few exceptions with PN). A detailed per-class analysis of both DA and DG results with the PN++ backbone for the ModelNet40→OBJ_ONLY case is shown in Figure 3. This visualization helps to understand which are the most difficult cases across domains.

The very recent PointDAN method [33] proposed to solve 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 [35]. Table 3 shows that our multi-task approach largely outperforms this solution[1]. Finally, an overall look at the performance of several recent point cloud networks is provided in the bottom part of Table 3. The results

[1] See the Appendix for an even more extensive comparison.

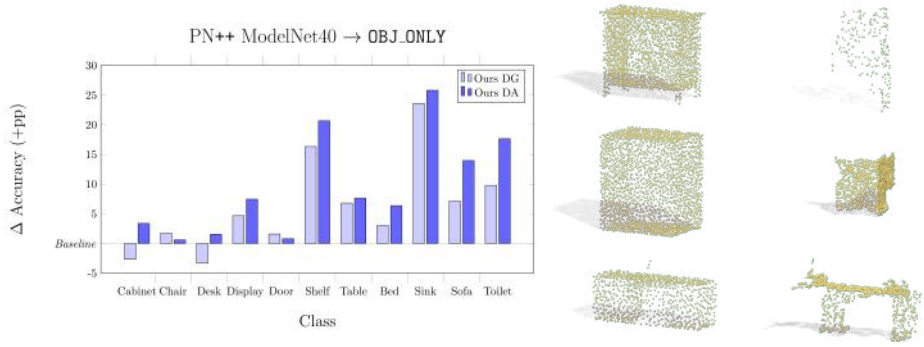


Fig. 3. *left*: Per-class accuracy improvements (w.r.t. baseline): our approach tackles domain shift with a gain up to 25 pp. *right*: Real-World shapes from classes 'Cabinet' and 'Desk' are very difficult to classify across domains due to noise and occlusion. Here the first column show examples from ModelNet, while the second column from ScanObject. First two rows are cabinet, the last row is desk

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.

4.4 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 [31], 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 used the PointNet Segmentation backbone from [31]. We slightly modified 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 4 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 [40]. In the SD setting the introduction of self-supervision in the learning process does not improve over the baseline accuracy. In the TL setting, by considering as extra source of

Table 4. Part segmentation of chairs from two variants of ScanObjectNN evaluated in terms of per part average accuaracy (%) and overall accuracy (%)

Dataset	Method	Bg	Seat	Back	Base	Arm	Avg. Acc.	Overall Acc.
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**

Fig. 4. 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

knowledge $\mathcal{S}'$ the unlabeled chairs from ModelNet40, our approach proves once again its effectiveness.

Few-Shot and Semi-Supervised. By following [16] 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 5 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 [16] 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 in Figure 4 for chairs and lamps. Regarding chairs, our multi-task approach seems to allow a better recognition of the armrests. Indeed the position of these relative small parts of the chair may be better learned thanks to the puzzle solution task. A similar consideration may be done for the lamp basis.

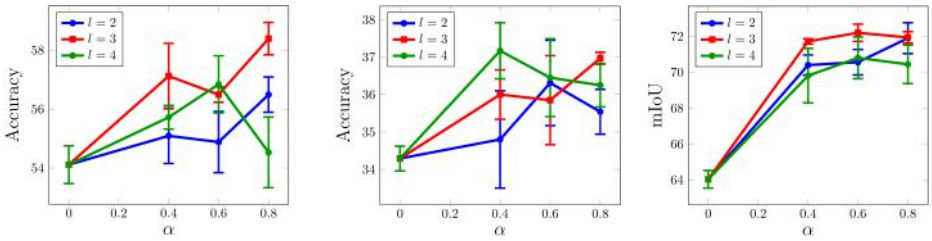
Domain Generalization and Adaptation. We focus on the domain shift between synthetic and real chairs with ShapeNetPart as source and ScanObjectNN as target. We highlight that ScanObjectNN has some part annotation issue confirmed by the authors through personal communications, thus we prefer to use only the OBJ_BG provided subdomain, neglecting the background which is absent

Table 5. Accuracy (mIoU) for part segmentation on ShapeNet-Part with limited annotations.

Method	1%	5%
SO-Net [22]	64.00	69.00
PointCapsNet [55]	67.00	70.00
CCD [16]	68.20	**77.70**
Baseline	64.52	75.75
Our FS	64.49	75.07
Our SS	**71.95**	77.42

Table 6. Per part and average accuracy (%) of chair segmentation. We use the same metric of Table 4

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**



(a) DA ModelNet40→`OBJ_ONLY` (b) DA ModelNet40→`PB_T50_RS_BG` (c) SS ShapeNetPart 1%

Fig. 5. Parameter (α, l) evaluation in case of cross domain Shape Classification (a,b), and 1% semi-supervised Part Segmentation (c). 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

in ShapenNetPart. Table 6 collects the results confirming also in this case the advantage of our multi-task approach over the supervised learning baseline.

4.5 Ablation Analysis

As indicated in sec. 3 our approach has two main hyperparameters. 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. Figure 5 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.

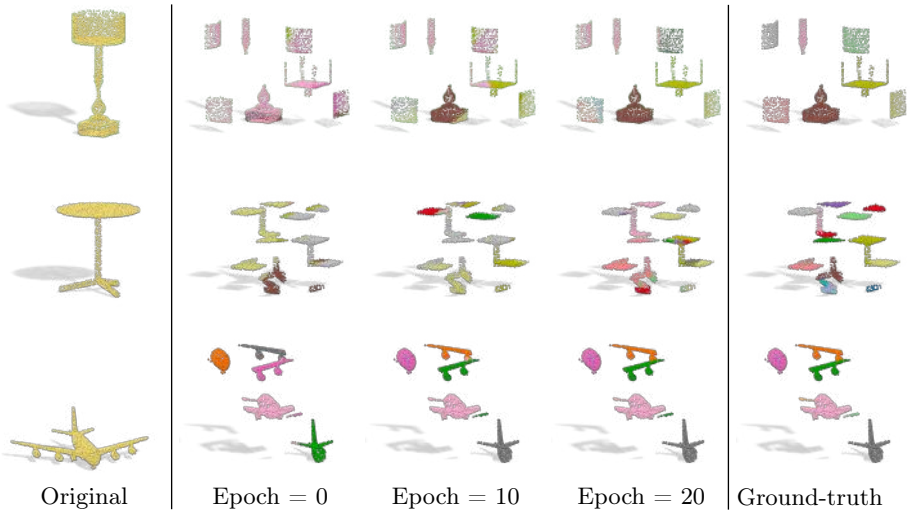


Fig. 6. Visualization of 3D puzzle progressively solved by our model. *First row*: the shuffled parts of the lamp are almost uniformly wrong at the beginning. The voxels are then identified during training. *Second row*: annotating the correct voxel appears particularly difficult for this table due to multiple similar subparts. *Third row*: for the airplane it seems easy to get coherent predictions but the voxel identity is initially mistaken and progressively corrected in the following epochs

4.6 Qualitative Analysis of Puzzle Solution

For a qualitative analysis of the learning process we show in Figure 6 how the puzzle is progressively solved at different epochs. Initially the annotation is quite confused, but then the model quickly improves by identifying to which decomposed voxel the sample part should belong to, and colors the points accordingly.

5 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. Our choice of the specific self-supervised task of solving 3D puzzle is indeed just one of the many possible that deserve attention for future work.

References

1. Achituve, I., Maron, H., Chechik, G.: Self-supervised learning for domain adaptation on point-clouds. Preprint ArXiv:2003.12641 (2020)
2. Ben-David, S., Blitzer, J., Crammer, K., Kulesza, A., Pereira, F., Vaughan, J.: A theory of learning from different domains. Machine Learning **79**, 151–175 (2010)
3. Ben-Shabat, Y., Lindenbaum, M., Fischer, A.: 3dmfv: Three-dimensional point cloud classification in real-time using convolutional neural networks. IEEE Robotics and Automation Letters **3**(4), 3145–3152 (2018)
4. Boscaini, D., Masci, J., Melzi, S., Bronstein, M.M., Castellani, U., Vandergheynst, P.: Learning class-specific descriptors for deformable shapes using localized spectral convolutional networks. Comput. Graph. Forum **34**, 13–23 (2015)
5. Bousmalis, K., Trigeorgis, G., Silberman, N., Krishnan, D., Erhan, D.: Domain Separation Networks. In: NIPS (2016)
6. Carlucci, F.M., D'Innocente, A., Bucci, S., Caputo, B., Tommasi, T.: Domain generalization by solving jigsaw puzzles. In: CVPR (2019)
7. Carlucci, F.M., Porzi, L., Caputo, B., Ricci, E., Rota Bulò, S.: Autodial: Automatic domain alignment layers. In: ICCV (2017)
8. Carlucci, F.M., Russo, P., Tommasi, T., Caputo, B.: Hallucinating agnostic images to generalize across domains. In: TASK-CV Workshop at ICCV (2019)
9. Chang, A.X., Funkhouser, T., Guibas, L., Hanrahan, P., Huang, Q., Li, Z., Savarese, S., Savva, M., Song, S., Su, H., Xiao, J., Yi, L., Yu, F.: ShapeNet: An Information-Rich 3D Model Repository. In: Preprint ArXiv:1512.03012 (2015)
10. Dai, A., Chang, A.X., Savva, M., Halber, M., Funkhouser, T., Nießner, M.: Scannet: Richly-annotated 3d reconstructions of indoor scenes. In: CVPR (2017)
11. Doersch, C., Gupta, A., Efros, A.A.: Unsupervised visual representation learning by context prediction. In: ICCV (2015)
12. Ganin, Y., Ustinova, E., Ajakan, H., Germain, P., Larochelle, H., Laviolette, F., Marchand, M., Lempitsky, V.: Domain-adversarial training of neural networks. J. Mach. Learn. Res. **17**(1), 2096–2030 (2016)
13. Ghifary, M., Kleijn, W.B., Zhang, M., Balduzzi, D.: Domain generalization for object recognition with multi-task autoencoders. In: ICCV (2015)
14. Gidaris, S., Singh, P., Komodakis, N.: Unsupervised representation learning by predicting image rotations. In: ICLR (2018)
15. Han, Z., Wang, X., Liu, Y.S., Zwicker, M.: 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)
16. Hassani, K., Haley, M.: Unsupervised multi-task feature learning on point clouds. In: ICCV (2019)
17. Hoffman, J., Tzeng, E., Park, T., Zhu, J.Y., Isola, P., Saenko, K., Efros, A., Darrell, T.: CyCADA: Cycle-consistent adversarial domain adaptation. In: ICML (2018)
18. Larsson, G., Maire, M., Shakhnarovich, G.: Colorization as a proxy task for visual understanding. In: CVPR (2017)
19. Li, D., Yang, Y., Song, Y., Hospedales, T.M.: Learning to generalize: Meta-learning for domain generalization. In: AAAI (2018)
20. Li, D., Zhang, J., Yang, Y., Liu, C., Song, Y.Z., Hospedales, T.M.: Episodic training for domain generalization. In: ICCV (2019)
21. Li, H., Jialin Pan, S., Wang, S., Kot, A.C.: Domain generalization with adversarial feature learning. In: CVPR (2018)

22. Li, J., Chen, B.M., Hee Lee, G.: So-net: Self-organizing network for point cloud analysis. In: CVPR (2018)
23. Li, Y., Tian, X., Gong, M., Liu, Y., Liu, T., Zhang, K., Tao, D.: Deep domain generalization via conditional invariant adversarial networks. In: ECCV (2018)
24. Li, Y., Bu, R., Sun, M., Wu, W., Di, X., Chen, B.: Pointcnn: Convolution on x-transformed points. In: NIPS (2018)
25. Long, M., Cao, Y., Wang, J., Jordan, M.I.: Learning transferable features with deep adaptation networks. In: ICML (2015)
26. Long, M., Zhu, H., Wang, J., Jordan, M.I.: Deep transfer learning with joint adaptation networks. In: ICML (2017)
27. Mancini, M., Porzi, L., Rota Bulò, S., Caputo, B., Ricci, E.: Boosting domain adaptation by discovering latent domains. In: CVPR (2018)
28. Monti, F., Boscaini, D., Masci, J., Rodolà, E., Svoboda, J., Bronstein, M.M.: Geometric deep learning on graphs and manifolds using mixture model cnns. In: CVPR (2017)
29. Noroozi, M., Favaro, P.: Unsupervised learning of visual representations by solving jigsaw puzzles. In: ECCV (2016)
30. Pathak, D., Krähenbühl, P., Donahue, J., Darrell, T., Efros, A.: Context encoders: Feature learning by inpainting. In: CVPR (2016)
31. Qi, C.R., Su, H., Mo, K., Guibas, L.J.: PointNet: Deep Learning on Point Sets for 3D Classification and Segmentation. In: CVPR (2017)
32. Qi, C.R., Yi, L., Su, H., Guibas, L.J.: PointNet++: Deep Hierarchical Feature Learning on Point Sets in a Metric Space. In: NIPS (2017)
33. 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: NIPS (2019)
34. Russo, P., Carlucci, F.M., Tommasi, T., Caputo, B.: From source to target and back: symmetric bi-directional adaptive gan. In: CVPR (2018)
35. Saito, K., Watanabe, K., Ushiku, Y., Harada, T.: Maximum classifier discrepancy for unsupervised domain adaptation. CVPR (2018)
36. Sauder, J., Sievers, B.: Self-supervised deep learning on point clouds by reconstructing space. In: NIPS (2019)
37. Shankar, S., Piratla, V., Chakrabarti, S., Chaudhuri, S., Jyothi, P., Sarawagi, S.: Generalizing across domains via cross-gradient training. In: ICLR (2018)
38. Sun, B., Saenko, K.: Deep coral: Correlation alignment for deep domain adaptation. In: ECCV Workshops (2016)
39. Tzeng, E., Hoffman, J., Darrell, T., Saenko, K.: Adversarial discriminative domain adaptation. In: CVPR (2017)
40. Uy, M.A., Pham, Q.H., Hua, B.S., Nguyen, D.T., Yeung, S.K.: Revisiting Point Cloud Classification: A New Benchmark Dataset and Classification Model on Real-World Data. In: ICCV (2019)
41. Volpi, R., Namkoong, H., Sener, O., Duchi, J., Murino, V., Savarese, S.: Generalizing to unseen domains via adversarial data augmentation. In: NIPS (2018)
42. Wang, C., Samari, B., Siddiqi, K.: Local spectral graph convolution for point set feature learning. In: ECCV (2018)
43. Wang, Y., Sun, Y., Liu, Z., Sarma, S.E., Bronstein, M.M., Solomon, J.M.: Dynamic Graph CNN for Learning on Point Clouds. ACM Trans. Graph. (2019)
44. Wu, Z., Song, S., Khosla, A., Yu, F., Zhang, L., Tang, X., Xiao, J.: 3D ShapeNets: A Deep Representation for Volumetric Shapes. In: CVPR (2015)
45. Xu, J., Xiao, L., López, A.M.: Self-supervised domain adaptation for computer vision tasks. Preprint Arxiv:1907.10915 (2019)

46. Xu, R., Li, G., Yang, J., Lin, L.: Larger norm more transferable: An adaptive feature norm approach for unsupervised domain adaptation. In: ICCV (2019)
47. Xu, Y., Fan, T., Xu, M., Zeng, L., Qiao, Y.: Spidercnn: Deep learning on point sets with parameterized convolutional filters. In: ECCV (2018)
48. Yang, Y., Feng, C., Shen, Y., Tian, D.: Foldingnet: Point cloud auto-encoder via deep grid deformation. In: CVPR (2018)
49. Yi, L., Kim, V.G., Ceylan, D., Shen, I.C., Yan, M., Su, H., Lu, C., Huang, Q., Sheffer, A., Guibas, L.: A Scalable Active Framework for Region Annotation in 3D Shape Collections. In: SIGGRAPH Asia (2016)
50. Yi, L., Su, H., Guo, X., Guibas, L.J.: Syncspeccnn: Synchronized spectral cnn for 3d shape segmentation. In: CVPR (2017)
51. Zhai, X., Oliver, A., Kolesnikov, A., Beyer, L.: S4l: Self-supervised semi-supervised learning. In: ICCV (2019)
52. Zhang, H., Cisse, M., Dauphin, Y.N., Lopez-Paz, D.: mixup: Beyond empirical risk minimization. In: ICLR (2018)
53. Zhang, L., Zhu, Z.: Unsupervised feature learning for point cloud by contrasting and clustering with graph convolutional neural network. In: CVPR Workshop (2019)
54. Zhang, R., Isola, P., Efros, A.A.: Colorful image colorization. In: ECCV (2016)
55. Zhao, Y., Birdal, T., Deng, H., Tombari, F.: 3d point capsule networks. In: CVPR (2019)

Appendix

A Relation to other Puzzle Solvers

We designed our puzzle solver with the aim of making it easily compatible with the main supervised objective in a multi-task model specifically tailored for 3D problems. Indeed our learning architecture is trained by jointly optimizing both the puzzle and the main supervised task. This makes our approach different from the recently published work [36] 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 [6] where, however, the puzzle task is defined in 2D with a completely different logic with respect to that proposed in our work. Specifically in [6] the whole puzzled sample is described by a single index which identifies the voxels' permutation and the puzzle task is formalized as a classification problem to predict that index. This implementation does not have a straightforward extension to 3D because it does not deal properly with the non-Euclidean nature of point clouds, and makes the puzzle particularly difficult in case of empty voxels. A practical case would be that of having $l = 3$ or $l = 4$ as discussed in sec. 4.5, where among the large number of voxels, only a limited amount will contain some point. In a preliminary set of experiments we tested this global classification strategy observing poor results. This happens because the self-supervised task is not able to provide local auxiliary information and the multi-task model does not show any advantage wrt the single-task supervised baseline. 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

Table 7. Cross-Domain accuracy (%) on PointDAN [33] dataset. We compare Our approach against PointDAN and the method proposed in the recent pre-print [1] on top of the same PointNet [31] feature extractor. We refer to ModelNet-10 as *M*, ShapeNet-10 as *S* and ScanNet-10 as *S**

Method	M→S	M→S*	S→M	S→S*	S*→M	S*→S	Avg.
PointDAN [33]	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 [1]	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 [1]	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	**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**

in that voxel. After shuffling, the puzzle solver performs a per-point voxel label prediction. Since the focus is on the points and not on the voxels, the issue of empty voxels does not affect our approach.

B Further Experimental Comparisons in the DA Setting

In sec. 4.3 we discussed the experimental comparison of our approach against PointDAN [33], which is, at the time of writing, the only publication focusing on 3D pointcloud classification across domains. We extend here the evaluation on the dataset proposed in [33], built upon 10 overlapping object classes between ModelNet-10 (M), ShapeNet (S) and ScanNet [10] (S*). The first two are synthetic data collections, while the last is a real-world dataset from RGB-D scans.

We consider each 3D object aligned with respect to the upward direction and unit-cube scaled. During training we perform the standard data augmentation described in previous experiments: random angle y-axis rotation and random vertex jittering drawn from $\mathcal{N}(0, 0.01)$. We also compare our result with a very recent pre-print [1] which presents a domain adaptation approach based on the combination of region reconstruction (*RegRec*) and a training procedure based on the Mixup method [52] (Point Cloud Mixup, *PCM*). The first is indeed a self-supervised strategy, while the second is a training procedure that exploits linear combination of samples.

As shown in Table 7, even in this setting our multi-task approach outperforms PointDAN. With respect to the method in [1], the direct comparison with RegRec shows that this self-supervised reconstruction task is less powerful with respect to 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.