Self-Distillation for Unsupervised 3D Domain Adaptation

(Article begins on next page)

26 April 2024

Self-Distillation for Unsupervised 3D Domain Adaptation

Adriano Cardace   Riccardo Spezialetti   Pierluigi Zama Ramirez
Samuele Salti   Luigi Di Stefano
Department of Computer Science and Engineering (DISI)
University of Bologna, Italy
{adriano.cardace2, riccardo.spezialetti, pierluigi.zama}@unibo.it

Abstract

Point cloud classification is a popular task in 3D vision. However, previous works, usually assume that point clouds at test time are obtained with the same procedure or sensor as those at training time. Unsupervised Domain Adaptation (UDA) instead, breaks this assumption and tries to solve the task on an unlabeled target domain, leveraging only on a supervised source domain. For point cloud classification, recent UDA methods try to align features across domains via auxiliary tasks such as point cloud reconstruction, which however do not optimize the discriminative power in the target domain in feature space. In contrast, in this work, we focus on obtaining a discriminative feature space for the target domain enforcing consistency between a point cloud and its augmented version. We then propose a novel iterative self-training methodology that exploits Graph Neural Networks in the UDA context to refine pseudo-labels. We perform extensive experiments and set the new state-of-the art in standard UDA benchmarks for point cloud classification. Finally, we show how our approach can be extended to more complex tasks such as part segmentation.

1. Introduction

In recent years, point cloud classification has received a lot of attention due to its relevance to many practical applications such as scene understanding, augmented/mixed reality, robotics and autonomous driving [20, 52]. Deep learning brings the promise of data-driven solutions to this problem and a variety of deep architectures have arisen in response to this challenge [43, 42, 35, 69, 68, 34, 26, 61, 54].

The success of these approaches goes hand in hand with the availability of large datasets containing labelled shapes [64, 7]. However, most existing annotated datasets concern clean and occlusions-free CAD shapes, but deep models trained on these objects drastically fail when facing data with different characteristics. This is the case, in particular,
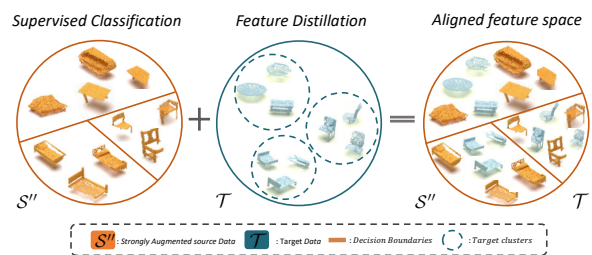
Figure 1. Proposed UDA method. We combine a supervised training of strongly augmented source data with a self-distillation approach that aims at clustering target shapes unsupervisedly. The combination of these two approaches leads to an alignment in feature space across domains.

of models trained on synthetic CAD data and then tested on point clouds obtained with real sensors, where parts of the object may be missing due to occlusions and measurements are corrupted by noise. Here comes to help Unsupervised Domain Adaptation (UDA), which pursues solving a supervised learning task in a *Target* domain, $\mathcal{T}$, where data come without labels, by leveraging on labeled data available in a *Source* domain, $\mathcal{S}$. In the last couple of years, an increasing number of papers [4, 43, 1, 2, 73] have addressed UDA for point cloud classification, with popular synthetic datasets of CAD models, such as ModelNet40 [65] or ShapeNet [7], and real datasets such as ScanNet [11]. The main line of research focuses on learning an effective feature space for the target domain by means of auxiliary tasks such as point cloud reconstruction [1, 46], 3D puzzle sorting [2] and rotation prediction [73]. These tasks are refereed as auxiliary since they do not directly solve the main task, but at the same time, they are useful to learn features for the target domain without the need of annotations. Although such techniques considerably improve over the baseline (i.e., training only on source data), the design of such tasks is not trivial and typically lead to sub-optimal solutions. It requires identifying one that can drive the network to learn repre-

sentations *discriminative* enough to perform classification in the target domain effectively. Despite the fact they force some degree of alignment between the features computed on objects from the two domains, such auxiliary tasks do not explicitly steer the network to learn discriminative representations amenable to classification in the target domain. For instance, if we train a network to reconstruct shapes, we will get similar point cloud embeddings for similar 3D shapes. However, two point clouds could represent objects that, though similar in shape, do belong to different categories, *e.g.*, a cabinet and a bookshelf. As a consequence, relying on reconstruction to perform domain adaptation can align features between the two domains, with similar shapes embedded close one to each other regardless of their domain, but the decision boundaries learnable from the labeled source samples may not discriminate effectively between target samples belonging to different classes. This is also shown in [1], where a simple denoising auto encoder for point clouds only slightly improves performance over the baseline. We reckon that similar considerations apply to the other auxiliary tasks proposed in the literature as they pursue cross-domain feature alignment based on a learning objective that does not ensure cross-domain class discriminability. We support this claim by comparing our proposal with previous works in the experimental section.

In this paper instead, we take inspiration from a recent self-supervised approach, DINO [6], to learn more discriminative representations for the target domain by constraining a sample and a strongly augmented version of itself to be classified similarly. This is typically achieved trough self-distillation, a methodology where the output of a neural network is compared with the output obtained from a mean teacher *i.e.* a temporal exponential moving average of the weights of the network itself (EMA) [53]. As shown in [6] this training methodology allows for clustering together samples of the same class. However, differently from DINO, we apply self-distillation for the first time in the 3D UDA context, where samples are point clouds, and the main goal is to reduce the gap between representations of two different domains rather than only learning a well-clustered feature space for a single domain. We believe that self-distillation is particularly suited for point cloud domain adaptation due to peculiar 3D data augmentations such as translation, occlusion and point-wise noise that can easily bridge the gap between source and target domain. By exploiting such augmentations to strongly augment source data and by enforcing inter-class discriminability for the target domain via self-distillation, we are able to obtain a shared aligned feature space across domains. The overall idea is illustrated in Fig. 1). Moreover, one major limitation of DINO that hinders its wide adoption is mode collapse [6], and previous works usually adopt multiple tricks and hyper-parameters such as clustering constraints [5], predic-

tor [21] and contrastive losses [66] that are difficult to apply and tune in other contexts. In this work, we show how this paradigm can be applied without such tricks to UDA for point cloud classification, where mode collapse is prevented by simultaneously training a classifier on labelled source data that inherently separates the features space according to semantic categories.

In the second step of our proposal, following recently published works in the field [4, 73, 46, 17], we make use of *self-training*, an iterative methodology that exploits the predictions of a pre-trained model (*pseudo-labels*) to provide partial supervision on the target domain as well. However, pseudo-labels are noisy and their naive use typically leads to overfitting of the dominant classes of the source domain as shown in [71, 75]. The strategy proposed by [4] to refine them requires offline training of an additional network for this purpose and the definition of hand-crafted rules based on k-NN queries, limiting its general applicability, while [73] adopted a standard procedure borrowed from the 2D world [74]. As a further contribution of our work, we take a different path, and propose to use Graph Neural Networks (GNNs) [63] to refine pseudo-labels online during self-training. Our main intuition is that by using a GNN, pseudo-labels are obtained by considering relationships between all target samples in the dataset rather than on single samples in isolation. This allows for reasoning at the dataset-level and enables to correct misclassified samples and thus refine pseudo-labels. Moreover, the target feature space is clustered thanks to the self-distillation, thus each node of the graph is likely to be connected to samples of the same category. Hence, the GNN can improve the pseudo-labels by reasoning on a neighborhood of samples sharing the same class. This procedure can be done online during training with the graph structure evolving over time, thus avoiding pseudo-labels overfitting.

Project Page at `https://cvlab-unibo.github.io/FeatureDistillation/`. In short, our contributions can be summarised as follows:

1. we propose the first approach in UDA for point clouds classification that exploits self-distillation to learn effective representations to classify point clouds in the target domain;

2. we show a novel strategy to use GNNs in UDA for point clouds classification. It enables online refinement of pseudo-labels, which reduces risks of overfitting, and it is conducive to effective self-training;

3. we extensively test our framework on standard benchmarks [43] and establish new state-of-the art results. Furthermore, we show how our approach can be generalized to the challenging task of part segmentation.

2. Related Work

Unsupervised 3D Domain Adaptation. Unsupervised Domain Adaptation emerged in the last years as a technique able to alleviate the domain shift when training a neural network on a source domain (e.g., synthetic simulations) and test in an unlabeled different, but related target domain (e.g., real data). UDA has a rich literature in the 2D world, and a noticeable amount of work as been conducted for image classification [19, 3, 37, 36, 51, 56], semantic segmentation [33, 55, 8, 25] and object detection [9, 60, 60]. PointDAN [43] has been the first work to address point cloud classification in the UDA context; they leverage on the well known Maximum Classifier Discrepancy (MCD) [45] to achieve alignment in feature space. Differently, [2, 1, 73] exploit Self-Supervised Learning (SSL) to run additional tasks on both domains. [4] also leverages on point cloud reconstruction, but uses it to refine pseudo-labels. Despite its effectiveness, their pipeline is rather complex and based on ad-hoc k-NN queries. The main difference with these works is in the way we exploit 3D transformations: while they use transformations such as rotations or point-wise jittering to solve an additional task in a self-supervised fashion, we use these augmentations in input space to design a novel distillation approach that pushes the network to learn a discriminative feature space for the target domain.

Self-training. Self-training [74] is a common technique used in Domain Adaptation to assign noisy annotations to target samples *i.e.* pseudo-labels [30], so that partial supervision can be provided to learn the distributions of the target domain. Pseudo-labels are often fairly inaccurate and many methods have been proposed to address this issue for UDA for image classification [22, 10, 48], semantic segmentation [40, 33, 27], and object detection [28, 59] by either filtering or refining pseudo labels. The potential of self-training has also been showed for point clouds classification in [4] where pseudo-labels are refined by an auxiliary reconstruction task. We also leverage on this powerful technique and propose for the first time to refine pseudo-labels using Graph Neural Networks in the UDA context.

Knowledge distillation. The case in which soft pseudo-labels rather than hard labels are used is typically denoted as *Knowledge distillation* [24]. Although distillation has been originally introduced to boost performance of small neural networks, recent works revisited knowledge distillation as way to learn robust features for better initialization or image retrieval [6]. In particular, DINO [6] proposed a novel framework able to learn robust features exploiting augmented versions of the same images of a given domain. Inspired by DINO[6], we propose to apply such paradigm to tackle the UDA scenario for 3D objects. Indeed we aim at showing that self-distillation can be applied exploiting 3D augmentations, and more importantly, that we can design such learning protocol to reduce the gap between a source and a target domain.

Graph Neural Networks (GNNs). Recent GNNs models [29, 58, 14] have emerged as powerful architectures for graph-structured data, covering a large spectrum of applications: social analysis [31, 44], drug discovery [16] and recommendation systems [18, 62]. The rich literature on 2D semi-supervised learning [29, 58, 12, 32, 47, 44] already provides many works that leverage on GNNs to assign labels on unlabelled nodes. However, all these works, assume a small amount of *perfectly* labelled nodes in the graph for each class, while this assumption does not hold in the UDA scenario. To the best of our knowledge, [39] is the only paper that addresses UDA for image classification using GNNs. They focus on extracting complementary features through a GNN to be combined with features obtained with a classical Convolutional Neural Network (CNN). Other works such as [13] and [15] instead, exploit graph structures (not GNNs) with hand-crafted label propagation algorithms to achieve adaptation. Differently, we propose the usage of GNNs to obtain new pseudo-labels while self-training, to avoid overfitting and to allow an iterative refinement of them to converge to better adaptation performances. Moreover, we are the first to show their effectiveness in the case of UDA for 3D shape classification.

3. Method

Our framework is divided into two main steps: self-distillation (Sec. 3.2) and *self-training with pseudo-labels refinement* (Sec. 3.3 and Sec. 3.4). The overall pipeline is depicted in Fig. 2. We start introducing the notation and a brief review of the basic concepts about GNNs.

3.1. Preliminaries

Notation. In this paper, we consider UDA for point cloud classification, *i.e.* given a point cloud with N elements $x \in \mathbb{R}^{N \times 3}$ we aim at learning a neural network $\Omega : x \to [0,1]^K$ that takes an input example x and produces a K-dimensional vector representing the confidence scores for K classes. Such a point cloud classifier consists of two components: $\Omega = \Phi \circ \Psi$. The first is a feature extractor network, $\Phi : \mathbb{R}^3 \to \mathbb{R}^D$, producing $g \in \mathbb{R}^D$, *i.e.* a D-dimensional global feature descriptor for the shape, the second is small MLP $\Psi : \mathbb{R}^D \to \mathbb{R}^K$ followed by a softmax operator which maps g to a vector of confidence scores $\hat{p} \in [0,1]^K$. Finally, the class predictions can be obtain by the argmax operator $\Lambda : \mathbb{R}^K \to \mathcal{Y}$. As it is peculiar in UDA settings, we have at our disposal a source domain with labels $\mathcal{S} = \{(x_s^i \in \mathcal{X}_s, y_s^i \in \mathcal{Y}_s)\}_{i=1}^{n_s}$, and a target domain $\mathcal{T} = \{x_t^j \in \mathcal{X}_t\}_{j=1}^{n_t}$, where the point clouds are unlabeled. Our objective is to obtain a classifier able to make correct predictions on $\mathcal{T}$.

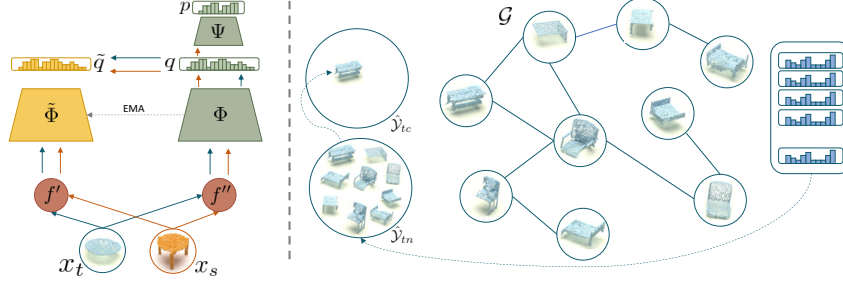**Background on GNNs.** Graph Neural Networks (GNNs) are models designed to process graphs, *i.e.* sets of

Figure 2. Illustration of our framework. **Left**: weakly and strongly augmented point clouds are generated with two transformation functions f' and f'' for both domains. The weakly augmented shapes are fed to an exponential moving average (EMA) encoder, the teacher $\tilde{\Phi}$, while the strongly augmented are processed by the student Φ. A consistency loss is applied between the corresponding embeddings. **Right**: the whole target dataset is processed by a GCN $\mathcal{G}$ online during self-training to iteratively refine and update pseudo-labels

nodes that are optionally joined one to another by edges representing relationships. GNNs are a powerful tool to process unstructured data thanks to their ability of updating the representations of each node by aggregation of information from the neighbouring nodes. An undirected graph $\mathcal{G}$ is represented as a tuple $(\mathcal{V}, \mathcal{E})$, where $\mathcal{V}$ is the set of N vertices $v_i \in \mathcal{V}$, and $\mathcal{E}$ is the set of edges. The graph topology is determined by the adjacency matrix $A \in \mathbb{R}^{N \times N}$, with $A_{i,j} = 1$ if two nodes i and j are connected. Among the many architectures of GNNs [63], in this work we adopt the Graph Convolutional Networks (GCNs) [29], which propagate the information between each layer according to the following propagation rule:

$$H^{(l+1)} = \sigma\left(\tilde{D}^{-\frac{1}{2}} \tilde{A} \tilde{D}^{-\frac{1}{2}} H^{(l)} W^{(l)}\right) \tag{1}$$

where $\tilde{A} = A + I$ represents the adjacency matrix with self-connections, I is the identity matrix, $\tilde{D}_{ii} = \sum_j \tilde{A}_{ij}$ acts as a scaling factor and $W^{(l)}$ is a layer-specific trainable weight matrix. The aggregation rule is followed by a non-linear activation function $\sigma(\cdot)$ such as ReLU. Note that matrix $H^{(l)}$ deals with the l-th layer of the network, each row i representing the feature vector of a node $v_i \in \mathcal{V}$ in that layer. We refer the reader to [63] for a more detailed discussion.

3.2. Self-distillation

In this section, we present the self-distillation module that we use in both steps of our pipeline. The purpose of this component is to distill good features for the target domain unsupervisedly, so that a discriminative feature space directly useful for classification can be learned even though no direct supervision is available in $\mathcal{T}$. Our main intuition, is that learning a clustered feature space that minimises the distance among variations of the same cloud from the target domain, while simultaneously learning decision boundaries amenable to classification thanks to a carefully augmented source domain, is key to obtain good pseudo-labels to be

deployed in the self-training process. Indeed, without enforcing compactness in the feature space, it is more likely that, due to the domain gap, target samples are spread across the different categories defined by the decision boundaries of the classifier. This is undesirable since it would lead to excessive noise in pseudo-labels.

To achieve our goal, we use two data augmentation functions $f', f'' : \mathbb{R}^{N \times 3} \to \mathbb{R}^{N \times 3}$ that take as input a point cloud x and return a weakly augmented (x') and a strongly augmented point cloud (x'') respectively. Then, we adopt a self-distillation paradigm, where we train a student encoder Φ to match the output of a teacher encoder $\tilde{\Phi}$. In particular, we match two global shape descriptors, $\tilde{g} = \tilde{\Phi}(x')$ and $g = \Phi(x'')$, computed by feeding a weakly augmented point cloud x' to the teacher and the strongly augmented version x'' to the student.

By taking inspiration from [6], we design the student and the teacher to output probability distributions over D dimensions, denoted by q and $\tilde{q}$, respectively. These probabilities can be obtained by normalizing the output of the two encoders, *i.e.* g and $\tilde{g}$, with a softmax function:

$$q(g, \tau) = \frac{\exp(g/\tau)}{\sum_{d=1}^{D} \exp(g^{(d)}/\tau)},$$
$$\tilde{q}(\tilde{g}, \tilde{\tau}) = \frac{\exp(\tilde{g}/\tilde{\tau})}{\sum_{d=1}^{D} \exp(\tilde{g}^{(d)}/\tilde{\tau})} \tag{2}$$

where $\tau > 0$ and $\tilde{\tau} > 0$ are the two temperature parameters which control the sharpness of the output distributions for the student and the teacher, respectively. Differently from [6], we don't require any complex scheduling for the temperature parameters, and we just empirically set them to 0.5 by observing the model performance on the source domain for the ModelNet→ScanNet experiment and set it to the same value for all the others. To force the embedding of the augmented point cloud to match that computed for the original one, we minimize the cross-entropy:

$$\mathcal{L}_{sd}(\tilde{g}, g) = -\tilde{q}(\tilde{g}, \tilde{\tau}) \log q(g, \tau) \tag{3}$$

by running backprop on the student network Φ, while the weights of the teacher are updated by computing an exponential moving average of those of the student. Please note that both networks share the same architecture but have different weights. We employ an EMA as a teacher network since it is a convenient way to provide robust and stable features throughout the training process without the need of training another network [53, 23].

Data augmentation and transformation functions. To implement f' and f'', we use a set of common data augmentation techniques for point clouds such as: jittering, elastic deformation [72], scaling along the three axis. More specifically, to obtain the weakly augmented point cloud x', we only use jittering, while for the the strongly augmented x'', we employ all the above transformations. Additionally, when performing synthetic-to-real adaptation, we also include random point removal [50]. We refer to the supplementary material for some qualitative examples.

Interestingly, the same 3D transformations can be used to simulate the target distribution given source data. In fact, although it is not possible to exactly predict the shift between two domains, one can approximate the nuisances that affect the target data through aggressive data augmentation. For example, when performing UDA between different synthetic domains, shapes may have similar geometric elements but with different style [38, 67], which can be mimicked by object distortions or elongation and scaling. Similarly, when moving from a synthetic domain to a real one, it is reasonable to assume that shapes within the same class will appear similar to CAD models but will have missing components due to occlusions, and point coordinates will be affected by the noise originated in the acquisition process. Therefore, as shown in Fig. 2 (left), at training time we augment the source data re-utilizing the transformation function f'', with the goal of minimising the gap between the two domains in the input space and seamlessly obtain a better alignment also in the feature space. Applying such well designed augmentations to source data combined with our distillation technique is beneficial to the student model. Intuitively, by distillation we aim at clustering target samples, while by data augmentation we force source clusters, naturally obtained with a classification loss, to be aligned with the target ones.

3.3. Pseudo-labels initialization

In the first step of our method, we exploit the self-distillation module presented in the previous section to obtain an initial set of pseudo-labels for the target domain. In particular, as shown in Fig. 2 (left), we train a classifier $\Omega = \Phi \circ \Psi$ on top of the student feature extractor and fed with augmented source data. We use the cross entropy loss:

$$\mathcal{L}_{ce}(x''_s, y_s) = -y_s \log \Omega(x''_s) \tag{4}$$

Simultaneously, we feed to the encoder Φ batches of source and target point clouds strongly augmented with the transformation function f'', while $\tilde{\Phi}$ receives the weakly augmented versions, and minimise Eq. (3) to learn the desired clustered feature space for the data of the target domain. After training, the initial set of pseudo-labels is computed by feeding each target sample x^i_t into Ω and selecting the class with the highest confidence score: $\hat{y}^i_t = \Lambda(\Omega(x^i_t))$.

3.4. Self-training and pseudo-labels refinement

In the second step, we exploit and refine the previously obtained pseudo-labels. We do this by alternating self-training and refinement in an iterative procedure.

Self-training. In this step we train our classifier $\Omega = \Phi \circ \Psi$ leveraging pseudo-labels, starting from scratch if it is the first iteration. To do so, we first split the pairs of target samples and associated pseudo-labels $(x^i_t, \hat{y}^i_t)$ into two disjoint sets, *i.e.* $\hat{\mathcal{Y}}_{tc}$ and $\hat{\mathcal{Y}}_{tn}$, associated with confident and non-confident pseudo-labels, respectively, and with the former initialized to the empty set. The sets will be useful to realize the iterative procedure outlined at the end of the section. We then train Φ and Ψ using self-distillation and supervision for both domains, with supervision for the target coming from pseudo-labels:

$$\mathcal{L} = -\mathcal{L}_{ce}(x''_s, y_s) - \lambda \mathcal{L}_{ce}(x'_t, \hat{y}_t) - \mathcal{L}_{sd}(x'', x')$$

$$\text{where} \quad \lambda = \begin{cases} 1, & \hat{y}_t \in \hat{\mathcal{Y}}_{tc} \\ 0.2, & \hat{y}_t \in \hat{\mathcal{Y}}_{tn} \end{cases}. \tag{5}$$

Note that, as in the previous step, $\mathcal{L}_{sd}$ acts on both domains. We provide a sensitivity analysis in the supplementary material for λ when $\hat{y}_t \in \hat{\mathcal{Y}}_{tn}$ to show that our framework is not sensitive to this hyper-parameter.

Refinement. Naively using pseudo-labels as done in the previous step typically leads to ignoring the classes that are underrepresented in the source domain and to obtain suboptimal performance on the target domain due to noise in the pseudo-labels [40, 49]. Hence, we run self-training only for a few epochs and then refine the pseudo-labels exploiting a GCN. Our intuition is that, by leveraging on a global view of the target dataset, the GCN can better disambiguate hard cases compared to the initial pseudo-labels provided by the classifier, that, on the other hand, takes its decision on each input sample in isolation. For instance, even if few samples of a rare class are tightly connected (i.e. node with high degrees), it is likely for their confidence to be high as in their neighbourhood only nodes with the same class are present. The role of the GCN is therefore twofold: it corrects pseudo-labels; it decides which pseudo-labels should be considered confident and thereby moved from $\hat{\mathcal{Y}}_{tn}$ into $\hat{\mathcal{Y}}_{tc}$. We obtain the graph $\mathcal{G}$ by considering all samples in the target domain, as shown in Fig. 2 (right), and we build the adjacency matrix A based on the cosine similarity between

the global shape embeddings g:

$$A_{i,j} = \begin{cases} 1, & \frac{\langle g_i, g_j \rangle}{\|g_i\| \|g_j\|} > \epsilon \\ 0, & otherwise \end{cases} \qquad (6)$$

with ϵ being a similarity threshold empirically set to 0.95 so that the node degree (the average number of neighbours for each node of the graph) is roughly 10. We provide in the supplementary material a sensitivity study of this hyper-parameter, showing that our framework is insensitive w.r.t. to the node degree. This is necessary for memory con-straints, as the required memory to train a GCN is highly affected by this hyper-parameter. Inspired by [47], we equip each node in $\mathcal{G}$ with the embedding g as well as with the pre-diction provided by the classifier Ω, *i.e.* the vector $\hat{p}$. These two pieces of information provide the GCN with cues con-cerning both the geometric structure as well as the semantic class of the object. For example, it may be the case that two point clouds have similar embeddings and yet belong to different classes. This occurs frequently when consider-ing a real domain, where an occluded chair with a missing back could easily be misclassified as a table or the back it-self with missing legs can be confused for a monitor. Hence, providing the GCN with the additional information on the probability distribution among the K classes can help it at-taining more accurate pseudo-labels for target samples fea-turing similar embeddings. Then, we compute the input to the GCN as

$$H^{(0)} = \Phi(X_t) + \Omega(X_t)W_D \qquad (7)$$

where X_t is the set of all target samples and $W_D \in R^{K \times D}$ is a learnable projection matrix that projects the output dis-tribution over K classes in a D-dimensional space.

Afterwards, following Eq. (1), we stack three graph con-volutional layers where the last acts as a node classifier that returns a matrix of size $n_t \times K$. The GCN is optimized with a classical cross-entropy loss computed over all target samples, $\hat{\mathcal{Y}}_{tn} \cup \hat{\mathcal{Y}}_{tc}$, without taking into account the confi-dence on their pseudo-labels. It is worth noticing that, the predictions $\Omega(X_t)$, *i.e.* part of the input to the GCN, do not necessarily match the $\hat{\mathcal{Y}}_{tn} \cup \hat{\mathcal{Y}}_{tc}$ pseudo-labels. However, the GCN can just learn to output the same probability vec-tor $\Omega(X_t)$, discarding part of the input features [47], and consequently failing in generalizing at test time due to label leakage. Hence, we randomly mask (*i.e.* set to zero) 20% of the inputs $\Omega(X_t)$ at training time.

Finally, after training, we exploit the GCN to extract con-fident samples, *i.e.* the top θ predictions for each class, up-date the corresponding pseudo-labels with the output of the GCN, and move them from $\hat{\mathcal{Y}}_{tn}$ into $\hat{\mathcal{Y}}_{tc}$.

Iterative training. We argue that the topology of the graph highly influences the output of the GCN. As the en-coder improves its embeddings with multiple rounds of self-training, also thanks to the self-distillation process, pseudo-labels become better and better since the graph structure im-proves. Hence, we plug the previous steps into an iterative learning process, where we repeat:

a) self-train with Eq. (5) Φ and Ψ for e epochs using self-distillation and supervision for both domains, with su-pervision for the target coming from pseudo-labels;

b) build $\mathcal{G}$ and train the GCN to refine pseudo-labels;

c) update current pseudo-labels, moving the top θ predic-tions of the GCN for each class from $\hat{\mathcal{Y}}_{tn}$ to $\hat{\mathcal{Y}}_{tc}$.

To gradually increase the size of $\hat{\mathcal{Y}}_{tc}$, θ starts from 0 and grows to 1 to include more and more samples during train-ing. At test time, the GCN as well as the teacher encoder $\tilde{\Phi}$ can be simply discarded, with Ω being the only network re-quired to perform inference. Although the GCN can poten-tially be used at test time to obtain better performance, we discard it as this would introduce additional requirements such as keeping the whole training set in memory, and com-puting the neighborhood of each test sample.

4. Experiments

To show the effectiveness of out method, we compare against state-of-the-art methods for UDA for point cloud classification such as [4, 46, 17], using two different back-bones for our feature extractors: PointNet [41] and DGCNN [61]. Furthermore, we compare with a baseline *i.e.* a simple model trained only on the source domain without any adap-tation, and an oracle model, which instead assumes to have all target data available. The former constitutes the lower-bound in terms of performance, while the latter is consid-ered as the upper-bound since all target data can be utilized. Finally, we also conducted an experiment on the challeng-ing task of part segmentation to show how our method can be extended to different tasks than point cloud classifica-tion. In this case, we adopt the setting introduced in [2], which is the only method performing adaptation on such task for the synthetic-to-real scenario.

Datasets. The standard dataset used for UDA for point cloud classification is PointDA-10 [43], which con-sists of three subsets that share the same ten classes of three popular point clouds classification datasets: ShapeNet [7], ModelNet40 [64] and ScanNet [11]. This allows to define six different scenarios that involve synthetic-to-synthetic, synthetic-to-real and real-to-synthetic adaptation. ModelNet-10 consists of 4,183 training and 856 testing point clouds, that are extracted from synthetic 3D CAD models. Similarly, ShapeNet-10 features synthetic data only. It is the largest and most varied among the three datasets, and it comprises 17,378 training and 2,492 test-ing samples. Lastly, ScanNet-10 is the only real datasets,

Method	ModelNet to ShapeNet	ModelNet to ScanNet	ShapeNet to ModelNet	ShapeNet to ScanNet	ScanNet to ModelNet	ScanNet to ShapeNet	Avg
No Adaptation	80.5	41.6	75.8	40.0	60.5	63.6	60.3
PointDAN [43]	80.2	45.3	71.2	46.9	59.8	66.2	61.6
DefRec+PCM [1]	81.1	50.3	54.3	52.8	54.0	69.0	60.3
3D Puzzle [2]	81.6	49.7	73.6	41.9	65.9	68.1	63.5
RefRec [4]	81.4	56.5	**85.4**	53.3	73.0	73.1	70.5
(Ours)	**83.4**	**61.6**	77.3	**57.7**	**78.6**	**79.8**	**73.1**
Oracle	93.2	66.2	95	66.2	95.0	93.2	

Table 1. Shape classification accuracy (%) on the PointDA-10 dataset with PointNet. For each method, we report the average results on three runs. Best result on each column is in bold.

Method	ModelNet to ShapeNet	ModelNet to ScanNet	ShapeNet to ModelNet	ShapeNet to ScanNet	ScanNet to ModelNet	ScanNet to ShapeNet	Avg
No Adaptation	83.3	43.8	75.5	42.5	63.8	64.2	62.2
PointDAN [43]	83.9	44.8	63.3	45.7	43.6	56.4	56.3
DefRec+PCM [1]	81.7	51.8	78.6	54.5	73.7	71.1	68.6
GAST [†] [73]	**84.8**	59.8	**80.8**	56.7	81.1	74.9	73.0
GLRV [17]	85.4	60.4	78.8	57.7	77.8	76.2	72.7
ImplicitPCDA [46]	86.2	58.6	81.4	56.9	81.5	74.4	73.2
(Ours)	83.9	**61.1**	80.3	**58.9**	**85.5**	**80.9**	**75.1**
Oracle	93.9	78.4	96.2	78.4	96.2	93.9	80.5

Table 2. Shape classification accuracy (%) on the PointDA-10 dataset with DGCNN. For each method, we report the average results on three runs. Best result on each column is in bold. † Denotes a more powerful variant of DGCNN and results are obtained by performing checkpoint selection on the test set.

and it consists of 6,110 and 1,769 training and testing point clouds, respectively. It has been obtained from multiple real RGB-D scans. For this reason, it exhibits several forms of noise such as errors in the registration process and occlusions. Differently from point classification, there is no established setting for part segmentation in the literature and we refer to [2] as a reference since it is the only work performing synthetic-to-real adaptation from ShapeNetPart [57] to ScanOBJ-BG [70]. The task is solved only for the *chair* class, which comprises 4 components to segment: *Seat, Back, Base, Arm.*

4.1. Results

Classification. We report in Tab. 1 and Tab. 2 our results with PointNet and DGCNN, respectively. For PointNet, we establish overall the new state-of-the-art with 73.1% in terms of accuracy. We also note that our framework achieves the best results in 5 out of 6 settings, with a big gap in ModelNet→ScanNet and ShapeNet→ScanNet (+5.1% and +4.4%) that are the most challenging scenarios as they involve synthetic-to-real UDA. In particular, we highlight the result obtained in ModelNet→ScanNet (61.6%), which is, roughly, only 5% less than the oracle. We also observe remarkable improvements when addressing the opposite case *i.e.* real-to-synthetic (last two columns). This demonstrates the capability of our framework to deal with large domain shifts. As regards as synthetic-to-synthetic UDA, we observe good performance in ModelNet→ShapeNet, while we are the second best model in ShapeNet→ModelNet. We attribute the gap with RefRec to the peculiarity of ShapeNet→ModelNet, where the source domain is a com-

Part Segmentation: ShapeNetPart → ScanOBJ_BG					
Method	Seat	Back	Base	Arm	Avg.
Source only	67.85	45.60	84.89	14.87	53.30
3D Puzzle [2]	65.70	49.11	85.91	21.40	55.53
Self-dist (ours)	71.1	79.3	65.2	37.0	63.2
(ours)	**74.7**	**82.7**	67.9	37.7	**65.7**

Table 3. Per part and average mIoU (%) of chair segmentation for ShapeNetPart to ScanOBJ-BG.

plex dataset while the target is a simple one with shapes clearly distinguishable among classes *i.e.* objects with similar shapes do belong to the same class. In such specific scenarios, reconstruction-based approaches such as RefRec shine since the auxiliary task of reconstructing a point cloud naturally tends to form well-shaped clusters in feature space that are amenable for classification.

Furthermore, we repeat the same experiments using DGCNN as our main backbone. We achieve again state-of-the-art result (75.1%), showing the generality of our approach towards other architectures. Overall, we observe a similar trend w.r.t. Tab. 1, with an increase in performance in almost all configurations w.r.t. previous works.

Part Segmentation. Although our main goal is to propose a method that aims at solving UDA for point cloud classification, our method can be easily extended to more challenging tasks such as part segmentation, which consists in assigning to each vertex of the shape one object category. As done for point cloud classification, we perform a first step of self-distillation to distill good features for the target domain unsupervisedly. We then simply adapt the self-training step by considering each vertex of the input shape as a node in the graph. In this case, the node representation consists of a local feature vector extracted from the main backbone, which is a PointNet as in [2]. The whole graph is theoretically composed of all points of all shapes in the dataset. However, keeping all vertices in memory would be impractical and we perform the procedure explained in Sec. 3.2 by considering 20000 points of the whole dataset for each refinement iteration. Results are reported in Tab. 3. The evaluation metric is the mean Intersection-over-Union (mIoU), which is computed for each part Q for all the samples of the *chair* class. Then, the average across parts is reported. First, we observe that our full framework (last row) surpasses by more than 10% the previous method (second row). Furthermore, we highlight the effectiveness of self-distillation for the part segmentation task. Indeed, when only performing the first step of our pipeline (third row of Tab. 3), we already overcome [2] by 7.7%.

Self-distillation vs knowledge distillation. In Tab. 4, we ablate our self-distillation strategy and also compare it to an obvious alternative, *i.e.* applying Eq. (3) in output space. In this case, the self-distillation loss in Eq. (3) is applied on the output of the classifier rather than the feature vector of the backbone. As explained in Sec. 2, this protocol is similar to the knowledge distillation paradigm [24] that

Step	ce	sd	kd	ModelNet to ShapeNet	ModelNet to ScanNet	ShapeNet to ModelNet	ShapeNet to ScanNet	ScanNet to ModelNet	ScanNet to ShapeNet	Avg
	✓			80.5	41.6	75.8	40.0	60.5	63.6	60.3
PL init	✓	✓		**82.1**	**57.2**	77.6	**55.0**	**71.0**	**72.1**	**69.2**
	✓		✓	79.6	54.0	**79.2**	53.2	53.9	70.0	65.0

Table 4. Ablation study for the first step of our framework. ce: cross-entropy loss on source domain sd: self-distillation loss Eq. (3) in feature space used to train the pseudo-labels model; kd: standard knowledge distillation loss [24] in output space. We report the average results on three runs.

Step	st	ref	sd	ModelNet to ShapeNet	ModelNet to ScanNet	ShapeNet to ModelNet	ShapeNet to ScanNet	ScanNet to ModelNet	ScanNet to ShapeNet	Avg
	✓			82.7	59.3	74.9	56.4	77.1	77.8	71.4
Adaptation	✓	✓		**83.4**	60.9	**78.2**	56.3	77.9	79.4	72.7
	✓	✓	✓	**83.4**	**61.6**	77.3	**57.7**	**78.6**	**79.8**	**73.1**

Table 5. Ablation for the second step of our algorithm. st: self-training with pseudo-labels of the last row of Tab. 4 model; sd: self-distillation loss in the adaptation step; ref: refinement of pseudo-labels with GCN. We average results on three runs.

uses soft pseudo-labels. While we observe in both cases an improvement over the baseline trained only on source data (first row), the improvement is twice as large when self-distillation is deployed, which demonstrates the importance of working in feature space. Moreover, the large improvement in absolute terms (+8.9% on average) attained by using self-distillation shows its effectiveness in reducing the domain gap, validating our intuition to use it to tackle UDA. Interestingly, we observe a different behaviour for ShapeNet→ModelNet. This is again likely due to the peculiarity of the setting. With the source domain being much larger and richer than the target one, it is plausible that pseudo-labels in output space are quite accurate, end therefore more effective in this case. The model trained with self-distillation is used to extract the initial set of pseudo-labels for our method, as well as for all the self-training variants compared in Tab. 5. Finally, we highlight how the results obtained with self-distillation are clearly superior in all scenarios on average to those attained by competitors based on self-supervised learning tasks, *e.g.* row 2 (DefRec) and 3 (3D puzzle) of Tab. 1, that are based on a reconstruction and a 3D puzzle pretext task, respectively. This provides empirical support for our claim on the higher effectiveness of self-distillation with respect to auxiliary tasks for 3D UDA.

Self-training strategies. In Tab. 5, we perform an ablation study on the second step of our pipeline. We start by applying the simplest strategy to perform self-training (first row), *i.e.* using all pseudo-labels of the target domain together with the labels from the source domain to train a single classifier. This provides competitive results (71.4%), that is already better than the previous state-of-the-art model (70.5%), again showcasing the effectiveness of self-distillation to obtain pseudo-labels for UDA. When activating also the proposed online refinement that iteratively improves pseudo-labels thanks to the global reasoning of the GCN (second row), we appreciate another large improvement compared to the naive self-training, which validates the importance of the proposed iterative refinement.
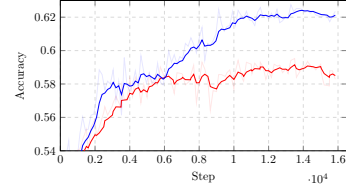

Figure 3. Test accuracy on target domain during training on ModelNet→ScanNet. Our model (Blue) consistently improves pseudo-labels during training differently from a simple self-training strategy in which pseudo-labels are fixed (Red).

Finally, in the last row, we report the results attained by activating self-distillation also in the adaptation step, which leads to the best performance and is the model used in all other experiments. As a further validation of the importance of the design decisions in our framework, we plot the training curves of the synthetic-to-real ModelNet→ScanNet in Fig. 3. The curves represent the test accuracy on the target domain during training. The red plot shows the behaviour of the naive self-training, which correspond to row 1 of Tab. 5. On the other hand, the blue lines represent the training curves obtained with our full model, *i.e.* last row of Tab. 5. We can appreciate that, after a certain number of steps, the blue line is always above the red line. This is a clear evidence that in our full model, pseudo-labels are improved over time, while in the naive case the model starts to overfit, leading to a plateau. We also wish to point out that such behaviour is key to a good UDA method because, in absence of target labels to perform validation, it is basically impossible to decide when to stop the training process.

5. Limitations

The main limitation of the proposed approach is the hand-crafted data augmentation functions used to augment both source and target data. To this end, we would like to investigate the possibility to learn a transformation able to automatically model the gap between the two domains. This would allow to handle dynamically cases where less augmentation is needed, such as in ShapeNet→ModelNet.

6. Conclusion

In this work, we explored a novel strategy to learn features on the target domain without the need of annotations. We first proposed to guide the network to learn a clustered feature space for the target domain and preserve discriminability suitable for classification. In addition, we introduced a novel refinement strategy that is able to globally reason on the target domain by means of GNN and to correct misclassified samples during training. Combining the two contributions, allowed to establish the state-of-the-art in the reference benchmarks. Finally, we showed how these contributions can be used for more challenging tasks such as part segmentation.

References

[1] Idan Achituve, Haggai Maron, and Gal Chechik. Self-supervised learning for domain adaptation on point clouds. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pages 123–133, 2021.

[2] Antonio Alliegro, Davide Boscaini, and Tatiana Tommasi. Joint supervised and self-supervised learning for 3d real world challenges. In *2020 25th International Conference on Pattern Recognition (ICPR)*, pages 6718–6725, 2021.

[3] Konstantinos Bousmalis, George Trigeorgis, Nathan Silberman, Dilip Krishnan, and Dumitru Erhan. Domain separation networks. In *Proceedings of the 30th International Conference on Neural Information Processing Systems*, page 343–351, Red Hook, NY, USA, 2016. Curran Associates Inc.

[4] Adriano Cardace, Riccardo Spezialetti, Pierluigi Zama Ramirez, Samuele Salti, and Luigi Di Stefano. Refrec: Pseudo-labels refinement via shape reconstruction for unsupervised 3d domain adaptation. In *2021 International Conference on 3D Vision (3DV)*. IEEE, 2021.

[5] Mathilde Caron, Piotr Bojanowski, Armand Joulin, and Matthijs Douze. Deep clustering for unsupervised learning of visual features. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 132–149, 2018.

[6] Mathilde Caron, Hugo Touvron, Ishan Misra, Hervé Jégou, Julien Mairal, Piotr Bojanowski, and Armand Joulin. Emerging properties in self-supervised vision transformers. In *Proceedings of the International Conference on Computer Vision (ICCV)*, 2021.

[7] Angel X Chang, Thomas Funkhouser, Leonidas Guibas, Pat Hanrahan, Qixing Huang, Zimo Li, Silvio Savarese, Manolis Savva, Shuran Song, Hao Su, et al. Shapenet: An information-rich 3d model repository. *arXiv preprint arXiv:1512.03012*, 2015.

[8] Minghao Chen, Hongyang Xue, and Deng Cai. Domain adaptation for semantic segmentation with maximum squares loss. *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*, Oct 2019.

[9] Yuhua Chen, Wen Li, Christos Sakaridis, Dengxin Dai, and Luc Van Gool. Domain adaptive faster r-cnn for object detection in the wild. In *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 3339–3348, 2018.

[10] Yining Chen, Colin Wei, Ananya Kumar, and Tengyu Ma. Self-training avoids using spurious features under domain shift. In Hugo Larochelle, Marc'Aurelio Ranzato, Raia Hadsell, Maria-Florina Balcan, and Hsuan-Tien Lin, editors, *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*, 2020.

[11] Angela Dai, Angel X Chang, Manolis Savva, Maciej Halber, Thomas Funkhouser, and Matthias Nießner. Scannet: Richly-annotated 3d reconstructions of indoor scenes. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 5828–5839, 2017.

[12] Enyan Dai, Charu Aggarwal, and Suhang Wang. Nrgnn: Learning a label noise resistant graph neural network on sparsely and noisily labeled graphs. In *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining*, KDD '21, page 227–236, New York, NY, USA, 2021. Association for Computing Machinery.

[13] Debasmit Das and C. S. George Lee. Graph matching and pseudo-label guided deep unsupervised domain adaptation. In Věra Kůrková, Yannis Manolopoulos, Barbara Hammer, Lazaros Iliadis, and Ilias Maglogiannis, editors, *Artificial Neural Networks and Machine Learning – ICANN 2018*, pages 342–352, Cham, 2018. Springer International Publishing.

[14] Michaël Defferrard, Xavier Bresson, and Pierre Vandergheynst. Convolutional neural networks on graphs with fast localized spectral filtering. *Advances in neural information processing systems*, 29:3844–3852, 2016.

[15] Zhengming Ding, Sheng Li, Ming Shao, and Yun Raymond Fu. Graph adaptive knowledge transfer for unsupervised domain adaptation. In *ECCV*, 2018.

[16] David K Duvenaud, Dougal Maclaurin, Jorge Iparraguirre, Rafael Bombarell, Timothy Hirzel, Alan Aspuru-Guzik, and Ryan P Adams. Convolutional networks on graphs for learning molecular fingerprints. In C. Cortes, N. Lawrence, D. Lee, M. Sugiyama, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 28. Curran Associates, Inc., 2015.

[17] Hehe Fan, Xiaojun Chang, Wanyue Zhang, Yi Cheng, Ying Sun, and Mohan Kankanhalli. Self-supervised global-local structure modeling for point cloud domain adaptation with reliable voted pseudo labels. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 6377–6386, June 2022.

[18] Wenqi Fan, Yao Ma, Qing Li, Yuan He, Eric Zhao, Jiliang Tang, and Dawei Yin. Graph neural networks for social recommendation. *The World Wide Web Conference on - WWW '19*, 2019.

[19] Yaroslav Ganin and Victor Lempitsky. Unsupervised domain adaptation by backpropagation. In *Proceedings of the 32nd International Conference on International Conference on Machine Learning - Volume 37*, ICML'15, page 1180–1189. JMLR.org, 2015.

[20] Andreas Geiger, Philip Lenz, and Raquel Urtasun. Are we ready for autonomous driving? the kitti vision benchmark suite. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2012.

[21] Jean-Bastien Grill, Florian Strub, Florent Altché, Corentin Tallec, Pierre H Richemond, Elena Buchatskaya, Carl Doersch, Bernardo Avila Pires, Zhaohan Daniel Guo, Mohammad Gheshlaghi Azar, et al. Bootstrap your own latent: A new approach to self-supervised learning. *arXiv preprint arXiv:2006.07733*, 2020.

[22] Xiang Gu, Jian Sun, and Zongben Xu. Spherical space domain adaptation with robust pseudo-label loss. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2020.

[23] Kaiming He, Haoqi Fan, Yuxin Wu, Saining Xie, and Ross Girshick. Momentum contrast for unsupervised visual representation learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 9729–9738, 2020.

[24] Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. Distilling the knowledge in a neural network, 2015.

[25] Judy Hoffman, Eric Tzeng, Taesung Park, Jun-Yan Zhu, Phillip Isola, Kate Saenko, Alexei Efros, and Trevor Darrell. CyCADA: Cycle-consistent adversarial domain adaptation. In Jennifer Dy and Andreas Krause, editors, *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pages 1989–1998. PMLR, 10–15 Jul 2018.

[26] Binh-Son Hua, Minh-Khoi Tran, and Sai-Kit Yeung. Point-wise convolutional neural networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 984–993, 2018.

[27] Myeongjin Kim and Hyeran Byun. Learning texture invariant representation for domain adaptation of semantic segmentation. *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, Jun 2020.

[28] Seunghyeon Kim, Jaehoon Choi, Taekyung Kim, and Changick Kim. Self-training and adversarial background regularization for unsupervised domain adaptive one-stage object detection. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, October 2019.

[29] Thomas N Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907*, 2016.

[30] D. Lee. Pseudo-label: The simple and efficient semi-supervised learning method for deep neural networks. In *International Conference on Machine Learning (ICML) Workshop*, 2013.

[31] Chang Li and Dan Goldwasser. Encoding social information with graph convolutional networks forPolitical perspective detection in news media. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 2594–2604, Florence, Italy, July 2019. Association for Computational Linguistics.

[32] Qimai Li, Xiao-Ming Wu, Han Liu, Xiaotong Zhang, and Zhichao Guan. Label efficient semi-supervised learning via graph filtering. *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, Jun 2019.

[33] Yunsheng Li, Lu Yuan, and Nuno Vasconcelos. Bidirectional learning for domain adaptation of semantic segmentation. *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, Jun 2019.

[34] Yongcheng Liu, Bin Fan, Shiming Xiang, and Chunhong Pan. Relation-shape convolutional neural network for point cloud analysis. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8895–8904, 2019.

[35] Ze Liu, Han Hu, Yue Cao, Zheng Zhang, and Xin Tong. A closer look at local aggregation operators in point cloud analysis. In *European Conference on Computer Vision*, pages 326–342. Springer, 2020.

[36] Mingsheng Long, Han Zhu, Jianmin Wang, and Michael I Jordan. Unsupervised domain adaptation with residual transfer networks. In D. Lee, M. Sugiyama, U. Luxburg, I. Guyon, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 29. Curran Associates, Inc., 2016.

[37] Mingsheng Long, Han Zhu, Jianmin Wang, and Michael I. Jordan. Deep transfer learning with joint adaptation networks. In *Proceedings of the 34th International Conference on Machine Learning - Volume 70*, ICML'17, page 2208–2217. JMLR.org, 2017.

[38] Zhaoliang Lun, Evangelos Kalogerakis, and Alla Sheffer. Elements of style: learning perceptual shape style similarity. *ACM Transactions on graphics (TOG)*, 34(4):1–14, 2015.

[39] Xinhong Ma, Tianzhu Zhang, and Changsheng Xu. Gcan: Graph convolutional adversarial network for unsupervised domain adaptation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019.

[40] Ke Mei, Chuang Zhu, Jiaqi Zou, and Shanghang Zhang. Instance adaptive self-training for unsupervised domain adaptation. *Lecture Notes in Computer Science*, page 415–430, 2020.

[41] Charles R Qi, Hao Su, Kaichun Mo, and Leonidas J Guibas. Pointnet: Deep learning on point sets for 3d classification and segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 652–660, 2017.

[42] Charles Ruizhongtai Qi, Li Yi, Hao Su, and Leonidas J Guibas. Pointnet++: Deep hierarchical feature learning on point sets in a metric space. In *NIPS*, 2017.

[43] Can Qin, Haoxuan You, Lichen Wang, C.-C. Jay Kuo, and Yun Fu. Pointdan: A multi-scale 3d domain adaption network for point cloud representation. In *Advances in Neural Information Processing Systems*, 2019.

[44] Jiezhong Qiu, Jian Tang, Hao Ma, Yuxiao Dong, Kuansan Wang, and Jie Tang. Deepinf: Social influence prediction with deep learning. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, KDD '18, page 2110–2119, New York, NY, USA, 2018. Association for Computing Machinery.

[45] Kuniaki Saito, Kohei Watanabe, Y. Ushiku, and T. Harada. Maximum classifier discrepancy for unsupervised domain adaptation. *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 3723–3732, 2018.

[46] Yuefan Shen, Yanchao Yang, Mi Yan, He Wang, Youyi Zheng, and Leonidas J. Guibas. Domain adaptation on point clouds via geometry-aware implicits. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 7223–7232, June 2022.

[47] Yunsheng Shi, Zhengjie Huang, Shikun Feng, Hui Zhong, Wenjing Wang, and Yu Sun. Masked label prediction: Unified message passing model for semi-supervised classification. In *Proceedings of the Thirtieth International Joint Conference on Artificial Intelligence (IJCAI)*, pages 1548–1554, 2021.

[48] Inkyu Shin, Sanghyun Woo, Fei Pan, and In So Kweon. Two-phase pseudo label densification for self-training based domain adaptation. In Andrea Vedaldi, Horst Bischof, Thomas Brox, and Jan-Michael Frahm, editors, *Computer Vision - ECCV 2020 - 16th European Conference, Glasgow, UK, August 23-28, 2020, Proceedings, Part XIII*, volume 12358 of *Lecture Notes in Computer Science*, pages 532–548. Springer, 2020.

[49] Inkyu Shin, Sanghyun Woo, Fei Pan, and In So Kweon. Two-phase pseudo label densification for self-training based domain adaptation. *Lecture Notes in Computer Science*, page 532–548, 2020.

[50] Riccardo Spezialetti, Federico Stella, Marlon Marcon, Luciano Silva, Samuele Salti, and Luigi Di Stefano. Learning to orient surfaces by self-supervised spherical cnns. *Advances in Neural Information Processing Systems*, 33, 2020.

[51] Baochen Sun and Kate Saenko. Deep coral: Correlation alignment for deep domain adaptation. In *ECCV Workshops*, 2016.

[52] Pei Sun, Henrik Kretzschmar, Xerxes Dotiwalla, Aurelien Chouard, Vijaysai Patnaik, Paul Tsui, James Guo, Yin Zhou, Yuning Chai, Benjamin Caine, et al. Scalability in perception for autonomous driving: Waymo open dataset. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 2446–2454, 2020.

[53] Antti Tarvainen and Harri Valpola. Mean teachers are better role models: Weight-averaged consistency targets improve semi-supervised deep learning results. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, NIPS'17, page 1195–1204, Red Hook, NY, USA, 2017. Curran Associates Inc.

[54] Hugues Thomas, Charles R Qi, Jean-Emmanuel Deschaud, Beatriz Marcotegui, François Goulette, and Leonidas J Guibas. Kpconv: Flexible and deformable convolution for point clouds. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 6411–6420, 2019.

[55] Yi-Hsuan Tsai, Wei-Chih Hung, Samuel Schulter, Kihyuk Sohn, Ming-Hsuan Yang, and Manmohan Chandraker. Learning to adapt structured output space for semantic segmentation. *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, Jun 2018.

[56] Eric Tzeng, Judy Hoffman, Kate Saenko, and Trevor Darrell. Adversarial discriminative domain adaptation. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2962–2971, 2017.

[57] Mikaela Angelina Uy, Quang-Hieu Pham, Binh-Son Hua, Duc Thanh Nguyen, and Sai-Kit Yeung. Revisiting point cloud classification: A new benchmark dataset and classification model on real-world data, 2019.

[58] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. Graph Attention Networks. *International Conference on Learning Representations*, 2018.

[59] He Wang, Yezhen Cong, Or Litany, Yue Gao, and Leonidas J. Guibas. 3dioumatch: Leveraging iou prediction for semi-supervised 3d object detection. *2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, Jun 2021.

[60] Tao Wang, Xiaopeng Zhang, Li Yuan, and Jiashi Feng. Few-shot adaptive faster r-cnn. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019.

[61] Yue Wang, Yongbin Sun, Ziwei Liu, Sanjay E Sarma, Michael M Bronstein, and Justin M Solomon. Dynamic graph cnn for learning on point clouds. *Acm Transactions On Graphics (tog)*, 38(5):1–12, 2019.

[62] Shu Wu, Yuyuan Tang, Yanqiao Zhu, Liang Wang, Xing Xie, and Tieniu Tan. Session-based recommendation with graph neural networks. *Proceedings of the AAAI Conference on Artificial Intelligence*, 33:346–353, Jul 2019.

[63] Zonghan Wu, Shirui Pan, Fengwen Chen, Guodong Long, Chengqi Zhang, and Philip S. Yu. A comprehensive survey on graph neural networks. *IEEE Transactions on Neural Networks and Learning Systems*, 32(1):4–24, Jan 2021.

[64] Zhirong Wu, Shuran Song, Aditya Khosla, Fisher Yu, Linguang Zhang, Xiaoou Tang, and Jianxiong Xiao. 3d shapenets: A deep representation for volumetric shapes. In *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1912–1920, 2015.

[65] Zhirong Wu, Shuran Song, Aditya Khosla, Fisher Yu, Linguang Zhang, Xiaoou Tang, and Jianxiong Xiao. 3d shapenets: A deep representation for volumetric shapes. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1912–1920, 2015.

[66] Zhirong Wu, Yuanjun Xiong, Stella X Yu, and Dahua Lin. Unsupervised feature learning via non-parametric instance discrimination. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3733–3742, 2018.

[67] Kai Xu, Honghua Li, Hao Zhang, Daniel Cohen-Or, Yueshan Xiong, and Zhi-Quan Cheng. Style-content separation by anisotropic part scales. In *ACM SIGGRAPH Asia 2010 papers*, pages 1–10, 2010.

[68] Qiangeng Xu, Xudong Sun, Cho-Ying Wu, Panqu Wang, and Ulrich Neumann. Grid-gcn for fast and scalable point cloud learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5661–5670, 2020.

[69] Xu Yan, Chaoda Zheng, Zhen Li, Sheng Wang, and Shuguang Cui. Pointasnl: Robust point clouds processing using nonlocal neural networks with adaptive sampling. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5589–5598, 2020.

[70] Li Yi, Vladimir G. Kim, Duygu Ceylan, I-Chao Shen, Mengyan Yan, Hao Su, Cewu Lu, Qixing Huang, Alla Sheffer, and Leonidas Guibas. A scalable active framework for region annotation in 3d shape collections. *ACM Trans. Graph.*, 35(6), nov 2016.

[71] Zhedong Zheng and Yi Yang. Unsupervised scene adaptation with memory regularization in vivo. *Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence*, Jul 2020.

[72] Qian-Yi Zhou, Jaesik Park, and Vladlen Koltun. Open3D: A modern library for 3D data processing. *arXiv:1801.09847*, 2018.

[73] Longkun Zou, Hui Tang, Ke Chen, and Kui Jia. Geometry-aware self-training for unsupervised domain adaptation on object point clouds. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 6403–6412, October 2021.

[74] Yang Zou, Zhiding Yu, BVK Vijaya Kumar, and Jinsong Wang. Unsupervised domain adaptation for semantic segmentation via class-balanced self-training. In *Proceedings*

of the European Conference on Computer Vision (ECCV), pages 289–305, 2018.

[75] Yang Zou, Zhiding Yu, Xiaofeng Liu, B. V. K. Vijaya Kumar, and Jinsong Wang. Confidence regularized self-training. *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*, Oct 2019.