Label-Efficient Learning on Point Clouds using Approximate Convex Decompositions

Matheus Gadelha*, Aruni RoyChowdhury*‡, Gopal Sharma, Evangelos Kalogerakis, Liangliang Cao, Erik Learned-Miller, Rui Wang, Subhransu Maji

University of Massachusetts Amherst
{mgadelha,aruni,gopal,kalo,llcao,elm,ruiwang,smaji}@cs.umass.edu

Abstract. The problems of shape classification and part segmentation from 3D point clouds have garnered increasing attention in the last few years. Both of these problems, however, suffer from relatively small training sets, creating the need for statistically efficient methods to learn 3D shape representations. In this paper, we investigate the use of Approximate Convex Decompositions (ACD) as a self-supervisory signal for label-efficient learning of point cloud representations. We show that using ACD to approximate ground truth segmentation provides excellent self-supervision for learning 3D point cloud representations that are highly effective on downstream tasks. We report improvements over the state-of-the-art for unsupervised representation learning on the ModelNet40 shape classification dataset and significant gains in few-shot part segmentation on the ShapeNetPart dataset. Our source code is publicly available.[1]

1 Introduction

The performance of current neural network models on tasks such as classification and semantic segmentation of point cloud data is limited by the amount of labeled training data. Since collecting high quality annotations on 3D shapes is usually expensive and time consuming, there have been increasing efforts to train on noisy or weakly labeled datasets [42, 53, 83], or via completely unsupervised training [11, 19, 25, 75, 76]. An alternative strategy is to train the network on one task through *self-supervision* with automatically generated labels to initialize its parameters, then fine-tune it on the final task. Examples of such self-supervised approaches for network initialization include clustering [6, 7], solving jigsaw puzzles [44], and image colorization [78]. A key question in our problem setting is *"What makes for a good self-supervision task in the case of 3D shapes?"* That is, what tasks induce inductive biases that are beneficial to the downstream shape understanding tasks.

[1] https://github.com/matheusgadelha/PointCloudLearningACD
* equal contribution.
‡ Now at Amazon, work done prior to joining.

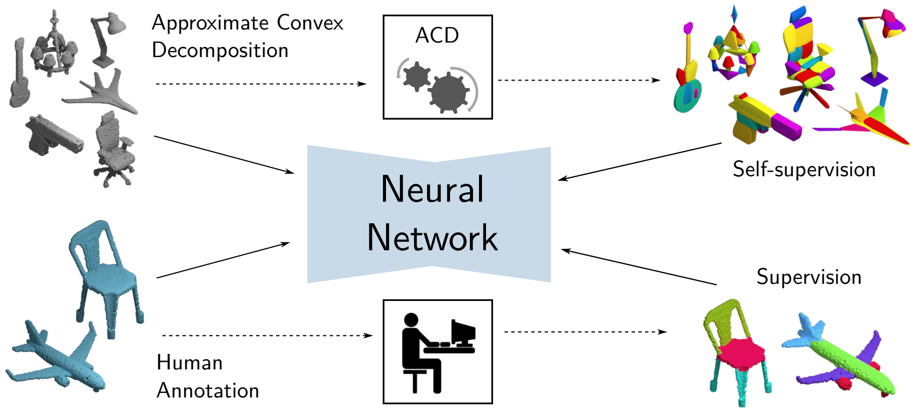

Fig. 1. Overview of our method versus a fully-supervised approach. ***Top:*** Approximate Convex Decomposition (ACD) can be applied on a large repository of unlabeled point clouds, yielding a *self-supervised* training signal for the neural network without involving any human annotators. ***Bottom:*** the usual *fully-supervised* setting, where human annotators label the semantic parts of shapes, which are then used as supervision for the neural network. The unsupervised ACD task results in learning useful representations from unlabeled data, significantly improving performance in shape classification and semantic segmentation, especially when labeled data is scarce or unavailable.

We posit that decomposing shapes into geometrically simple constituent parts provides an excellent self-supervisory learning signal for downstream 3D segmentation and classification tasks. Specifically, we propose using a classical shape decomposition method, Approximate Convex Decomposition (ACD), as the self-supervisory signal to pre-train neural networks for processing 3D data. Our approach is motivated by the observation that convexity provides cues to capture structural components related to human perception [30, 37], and object manufacturability [14, 38]. However, strict convex decomposition often leads to highly over-segmented shapes, leading to our choice of *approximate* convex decomposition [37]. As shown in the Figure 2, ACD decomposes shapes into segments that roughly align with instances of different parts. For example, two wings of an airplane are decomposed into two separate, approximately convex parts.

Our approach is illustrated in Figure 1. The main idea is to automatically generate training data by decomposing unlabeled 3D shapes into convex components. Since ACD relies solely on geometric information to perform its decomposition, the process does not require any human intervention. We formulate ACD as a metric learning problem on point embeddings and train the model using a contrastive loss [12, 24]. We demonstrate the effectiveness of our approach on standard 3D shape classification and segmentation benchmarks. In classification, we show that the representation learned from performing shape decomposition leads to features that achieve state-of-the-art performance for *unsupervised shape classification* on ModelNet40 [73] (**89.8**%). For *few-shot shape segmentation* on

ShapeNet [8], our model outperforms the state-of-the-art by **7.5%** mIoU when using 1% of the available labeled training data. Moreover, differently from other unsupervised approaches, our method can be applied to any of the well-known neural network backbones for point cloud processing. Finally, we provide thorough experimental analysis and visualizations demonstrating the role of the ACD self-supervision on representations learned by neural networks.

2 Related Work

Learning Representations on 3D data. Shape representations using neural networks have been widely studied in computer vision and graphics. An early approach was the use of *occupancy grids* to represent shapes for classification or segmentation tasks [40, 73]; however these representations suffered from computational and memory inefficiency. These problems were mitigated by architectures that use spatial partitioning data structures [32, 49, 66, 67]. *Multi-view approaches* [27, 31, 56, 58, 60] learn representations by using order invariant pooling of features from multiple rendered views of a shape. Another class of methods take *point cloud representations* (*i.e.*, a set of (x, y, z) coordinate triples) as input, and learn permutation invariant representations [18, 19, 25, 47, 48, 55, 70, 75]. Using point sets as a 3D representation does not suffer from the memory constraints of volumetric representations nor the self-occlusion issues of multi-view approaches. Still, all the above approaches rely on massive amounts of labeled 3D data. In this paper, we develop a technique to enable the learning of label-efficient representations from point clouds. Our approach is architecture-agnostic and relies on learning from approximate convex decompositions, which can be automatically computed from a 3D shapes.

Approximate Convex Decompositions. Studies in the cognitive science literature have argued that humans tend to reason about 3D shapes as the union of convex components [26]. However, performing exact convex decomposition is an NP-Hard problem that leads to an impractically high number of shape components [5]. An alternative class of decomposition techniques, named *Approximate Convex Decomposition* (ACD) [30, 37, 39, 82], compute approximate decompositions up to a concavity tolerance ϵ. This tolerance makes the computation significantly more efficient and leads to shape approximations containing a smaller number of components. Apart from shape segmentation [4, 30], these approximations are useful for a variety of tasks like mesh generation [37] and collision detection [71]. Furthermore, ACD-based decompositions has been shown to have a high degree of similarity to human segmentations, as indicated by segmentation evaluation measures, such as the Rand Index in the PSB benchmark [9, 30]. This indicates that ACD can provide useful cues for discovering shape parts.

There have been various techniques to compute ACD for shapes based on either geometric analysis [30,37,39,82], and recently with deep networks [10,15]. In this work, we used a particular type of ACD named Volumetric Hierachical Approximate Convex Decomposition (V-HACD) [39] (more details in Section 3.1.)

Unlike previous methods, our goal is to apply ACD to automatically supervise a convex decomposition task as an initialization step for learning point cloud representations. We show that the training signal provided by ACD leads to improvements in semantic segmentation as well as unsupervised shape classification.

Self-supervised learning. In many situations, unlabeled images or videos contain useful information that can be leveraged to automatically create a training loss for learning useful representations. *Self-supervised learning* explores this idea, using unlabeled data to train deep networks by solving tasks that do not require any human annotation effort.

Learning to colorize grayscale images was among the first approaches to training modern deep neural networks in a self-supervised fashion [35, 78, 79]. Estimating the color for a pixel from a black-and-white image requires some understanding of a pixel's meaning (*e.g.*, skies are blue, grass is green, etc.). Thus training a network to estimate pixel colors leads to the learning of representations that are useful in downstream tasks like object classification. The contextual information in an image also lends itself to the design of proxy tasks. These include learning to estimate the relative positions of cropped image patches [16], the similarity of patches tracked across videos [68, 69], the appearance of missing patches in an image [46, 63], or learning from image modalities in RGBD data [23, 58]. Motion from unlabeled videos also provides a useful pre-training signal. Pathak *et al.* [45] used motion segmentation to learn how to segment images, and Jiang *et al.* [29] estimate relative depth as a proxy task for pre-training a network for scene understanding tasks. Other approaches include solving jigsaw puzzles with permuted image patches [44], and training a generative adversarial model [17]. An empirical comparison of various self-supervised tasks may be found in [21, 33]. In the case of limited samples, *i.e.*, the *few-shot classification* setting, including self-supervised losses along with the usual supervised training is shown to be beneficial [59]. Recent work has also focused on learning unsupervised representations for 3D shapes using tasks such as clustering [25] and reconstruction [52, 76], which we compare against in our experiments.

Label-efficient representation learning on point clouds. Several recent approaches [11, 25, 43, 53, 83] have been proposed to alleviate expensive labeling of shapes. Muralikrishnan *et al.* [43] learn a per-point representation by training a network to predict shape-level tags. Yi et al. [77] embed pre-segmented parts in descriptor space by jointly learning a metric for clustering parts, assigning tags to them, and building a consistent part hierarchy. Chen *et al.* [11] proposed a branched auto-encoder, where each branch learns coarse part level features, which are further used to reconstruct the shape by producing implicit fields for each part. However, this approach requires one decoder for every different part, which restricts their experiments to category-specific models. On the other hand, our approach can be directly applied to any of the well known point-based architectures, is capable of handling multiple categories at once for part segmentation, and additionally learns useful features for unsupervised shape classification. Furthermore, Chen *et al.* [11] show experiments on single shot se-

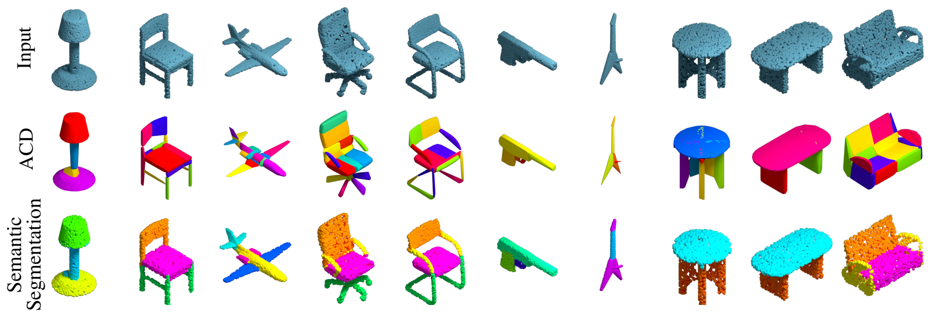

Fig. 2. Input point clouds (first row), convex components *automatically* computed by ACD (second row) and *human-labeled* point clouds (last row) from the ShapeNet [8] part segmentation benchmark. Note – ***(i)*** different colors for the ACD components only signify different parts– no semantic meaning or inter-shape correspondence is inferred by this procedure; ***(ii)*** for the human labels, colors do convey semantic meaning: *e.g.*, the backs of chairs are always orange; ***(iii)*** while the ACD decompositions tend to oversegment the shapes, they contain most of the boundaries present in the human annotations, suggesting that the model has similar criteria for decomposing objects into subparts; *e.g.*, the chair's legs are separated from the seat, wings and engines are separated from the airplane boundary, pistol trigger is separated from the rest.

mantic segmentation on manually selected shapes, whereas we show results on randomly selected training shapes in a few-shot setting. Most similar to our work, Hassani *et al.* [25] propose a novel architecture for point clouds, which is trained on multiple tasks at the same time: clustering, classification and reconstruction. In our experiments, we demonstrate that we outperform their method on few-shot segmentation by **7.5**% IoU and achieve the same performance on unsupervised ModelNet40 classification by using *only ACD as a proxy task*. If we further add a reconstruction term, our method achieves state-of-the-art performance in unsupervised shape classification. Finally, Sharma *et al.* [53] proposed learning point embeddings by using noisy part labels and semantic tags available on the 3DWarehouse dataset [62]. Their model is used for few-shot semantic segmentation. In this work, we instead gather part labels using approximate convex decomposition, whose computation is automatic and can be applied to any mesh regardless of the existence of semantic tags.

3 Method

3.1 Approximate Convex Decomposition

In this subsection, we provide an overview of the shape decomposition approach used to generate the training data for our self-supervised task. A detailed description of the method used in this work can be found in [39].

Given a polyhedron P, the goal is to compute the smallest set of convex polyhedra $\mathcal{C} = \{C_k | k = 1, ..., K\}$, such that the union $\cup_{k=1}^{K} C_i$ corresponds to P. Exact convex decomposition of polyhedra is an NP-Hard problem [5] and leads to decompositions containing too many components, rendering it impractical for use in most applications (ours included). This can be circumvented by Approximate Convex Decomposition (ACD) techniques. ACD relaxes the convexity constraint of exact convex decomposition by allowing every component to be *approximately convex* up to a concavity ϵ. The way concavity is computed and how the components are split varies according to different methods [20,30,37,39,82]. In this work, we use an approach called Volumetric Hierarchical Approximate Convex Decomposition (V-HACD) [39]. The reasons for using this approach are threefold. First, as the name suggests, V-HACD performs computations using volumetric representations, which can be easily computed from dense point cloud data or meshes and lead to good results without having to resort to costly mesh decimation and remeshing procedures. Second, the procedure is reasonably fast and can be applied to open surfaces of arbitrary genus. Third, V-HACD guarantees that no two components overlap, which means that there is no part of the surface that is approximated by more than one component. In the next paragraph, we describe V-HACD in detail.

V-HACD. Since the method operates on volumetric representations, the first step is to convert a shape into an occupancy grid. If the shape is represented as a point cloud, one can compute an occupancy grid by selecting which cells are occupied by the points and filling its interior. In our case, since our training shapes are from ShapeNet [8] that includes meshes, we chose to compute the occupancy grid by voxelizing the meshes using [28]. Once the voxelization is computed, the algorithm proceeds on computing convex components by recursively splitting the volume into two parts. First, the volume is centered and aligned in the coordinate system according to its principal axis. Then, one of the three axis aligned planes is selected as a splitting plane that separates the volume in two different parts. This procedure is applied multiple times until we reach the maximum number of desired components or the concavity tolerance is reached. The concavity $\eta(\mathcal{C})$ of a set of components $\mathcal{C}$ is computed as

$$\eta(\mathcal{C}) = \max_{k=1,...K} d(C_k, \texttt{CH}(C_k)), \tag{1}$$

where $d(X, Y)$ is the difference between the volumes X and Y; $\texttt{CH}(X)$ is the convex hull of X; and C_k is the kth element of the set $\mathcal{C}$. The splitting plane selection is performed by choosing the axis-aligned plane that minimizes an energy $E(V, \mathbf{p})$, where V is the volume we are aiming to split and $\mathbf{p}$ is the splitting plane. This energy is defined as:

$$E(V, \mathbf{p}) = E_{con}(V, \mathbf{p}) + \alpha E_{bal}(V, \mathbf{p}) + \beta E_{sym}(V, \mathbf{p}), \tag{2}$$

where E_{con} is a component connectivity term, which measures the sum of the normalized concavities between both sides of volume; E_{bal} is a balance component term, which measures the dissimilarity between both sides; and E_{sym} is a symmetry component term, which penalizes planes that are orthogonal to a potential revolution axis. The parameters α and β are weights for the last two terms. In all our experiments, we used the default values of $\alpha = \beta = 0.05$. We refer the reader to [39] for a detailed description of the components in the energy term.

Assigning component labels to point clouds. The output of ACD for every shape is a set of approximately convex components represented as meshes. For each shape, we sample points on the original ShapeNet mesh and on the mesh of every ACD component. We then propagate component labels to every point in the original point cloud by using nearest neighbor matching with points used in the decomposition. More precisely, given an unlabeled point cloud $\{p_i\}_{i=1}^N$, this assigns a component label $\Gamma(p_i, \mathcal{C})$ to each point p_i via:

$$\Gamma(p_i, \mathcal{C}) = \operatorname*{argmin}_{k=1\ldots|\mathcal{C}|} \left[\min_{p_j \in C_k} ||p_i - p_j|| \right]. \tag{3}$$

3.2 Self-supervision with ACD

The component labels generated by the ACD algorithm are not consistent across point clouds, *i.e.*, "component 5" may refer to the *seat* of a chair in one point cloud but the *leg* of the chair in another. Therefore, the usual cross-entropy loss, which is generally used to train networks for tasks such as semantic part labeling, is not applicable in our setting. We formulate the learning of ACDs as a metric learning problem on point embeddings via a *pairwise* or *contrastive loss* [24].

We assume that each point $p_i = (x_i, y_i, z_i)$ in a point cloud $\mathbf{x}$ is encoded as $\Phi(\mathbf{x})_i$ in some embedding space by a neural network encoder $\Phi(\cdot)$, *e.g.* PointNet [57] or PointNet++ [48]. Let the embeddings of a pair of points p_i and p_j from a shape be $\Phi(\mathbf{x})_i$ and $\Phi(\mathbf{x})_j$, normalized to unit length (*i.e.* $||\Phi(\mathbf{x})_i|| = 1$), and the set of convex components as described above be $\mathcal{C}$. The pairwise loss is then defined as

$$\mathcal{L}^{pair}(\mathbf{x}, p_i, p_j, \mathcal{C}) = \begin{cases} 1 - \Phi(\mathbf{x})_i^\top \Phi(\mathbf{x})_j, & \text{if } [\Gamma(p_i, \mathcal{C}) = \Gamma(p_j, \mathcal{C})] \\ \max(0, \Phi(\mathbf{x})_i^\top \Phi(\mathbf{x})_j - m), & \text{if } [\Gamma(p_i, \mathcal{C}) \neq \Gamma(p_j, \mathcal{C})]. \end{cases} \tag{4}$$

This loss encourages points belonging to the same component to have a high similarity $\Phi(\mathbf{x})_i^\top \Phi(\mathbf{x})_j$, and encourages points from different components to have low similarity, subject to a margin m (set to $m = 0.5$ as in [34]). $[\cdot]$ denotes the Iverson bracket.

Joint training with ACD. Formally, let us consider samples $\mathcal{X} = \{\mathbf{x}_i\}_{i \in [n]}$, divided into two parts: $\mathcal{X}^\mathcal{L}$ and $\mathcal{X}^\mathcal{U}$ of sizes l and u respectively. Now $\mathcal{X}^\mathcal{L} := \{\mathbf{x}_1, ..., \mathbf{x}_l\}$ consist of point clouds that are provided with human-annotated labels $\mathcal{Y}^\mathcal{L} := \{\mathbf{y}_1, ..., \mathbf{y}_l\}$, while we do not know the labels of the samples $\mathcal{X}^\mathcal{U} := \{\mathbf{x}_{l+1}, ..., \mathbf{x}_{l+u}\}$. By running ACD on the samples in $\mathcal{X}^\mathcal{U}$, we can obtain a set of components for each shape. The pairwise contrastive loss $\mathcal{L}^{pair}$

(Eq. 4) can then be defined over $\mathbf{x}_i \in \mathcal{X}^{\mathcal{U}}$ as a self-supervised objective. For the samples $\mathbf{x}_i \in \mathcal{X}^{\mathcal{L}}$, we have access to their ground-truth labels $\mathcal{Y}^{\mathcal{L}}$, which may for example, be semantic part labels. In that case, the standard choice of training objective is the *cross-entropy loss* $\mathcal{L}^{CE}$, defined over the points in an input point cloud. Thus, we can train a network on both $\mathcal{X}^{\mathcal{L}}$ and $\mathcal{X}^{\mathcal{U}}$ via a ***joint loss*** that combines both the supervised ($\mathcal{L}^{CE}$) and self-supervised ($\mathcal{L}^{pair}$) objectives,

$$\mathcal{L} = \mathcal{L}^{CE} + \lambda \cdot \mathcal{L}^{pair}. \tag{5}$$

The scalar hyper-parameter λ controls the relative strength between the supervised and self-supervised training signals. In the ***pretraining*** scenario, when we *only* have the unlabeled dataset $\mathcal{X}^{\mathcal{U}}$ available, we can train a neural network purely on the ACD parts by optimizing the $\mathcal{L}^{pair}$ objective.

4 Experiments

We demonstrate the effectiveness of the ACD-based self-supervision across a range of experimental scenarios. For all the experiments in this section we use ACDs computed on all shapes from the ShapeNetCore data [8], which contains 57,447 shapes across 55 categories. The decomposition was computed using a concavity tolerance of 1.5×10^{-3} and a volumetric grid of resolution 128^3. All the other parameters are set to their default values according to a publicly available implementation[2] of [39]. The resulting decompositions have an average of 17 parts per shape. The ACD computation takes 1.6s per shape on an Intel i7-2600 3.4GHz using 8 cores.

4.1 Shape classification on ModelNet

In this set of experiments, we show that the representations learned by a network trained on ACD are useful for discriminative downstream tasks such as classifying point clouds into shape categories.

Dataset. We report results on the ModelNet40 shape classification benchmark, which consists of 12,311 shapes from 40 shape categories in a train/test split of 9,843/2,468. A linear SVM is trained on the features extracted on the training set of ModelNet40. This setup mirrors other approaches for unsupervised learning on point clouds, such as FoldingNet [76] and Hassani *et al.* [25].

Experimental setup. A PointNet++ network is trained on the unlabeled ShapeNet-Core data using the pairwise contrastive loss on the ACD task, using the Adam optimizer, initial learning rate of 10^{-3} and halving the learning rate every epoch. This network architecture creates an embedding for each of the N points in an input shape, while for the shape classification task we require a single global descriptor for the entire point cloud. Therefore, we aggregate the per-point features of PointNet++ at the first two set aggregation layers (`SA1`

[2] `https://github.com/kmammou/v-hacd`

and `SA2`) and the last fully connected layer (`fc`), resulting in 128, 256 and 128 dimensional feature vectors, respectively. Since features from different layers may have different scales, we normalize each vector to unit length before concatenating them, and apply element-wise signed square-rooting [50], resulting in a final 512-dim descriptor for each point cloud. The results are presented in Table 1.

Comparison with baselines. As an initial naïve baseline, we use a PointNet++ network with random weights as our feature extractor, and then perform the usual SVM training. This gives 78% accuracy on ModelNet40 – while surprisingly good, the performance is not entirely unexpected: randomly initialized convolutional neural networks are known to provide useful features by virtue of their architecture, as studied in Saxe *et al.* [51]. Training this network with ACD, on the other hand, gives a significant boost to performance (78% $\rightarrow$ **89.1**%), demonstrating the effectiveness of our proposed self-supervision task. This indicates some degree of generalization across datasets and tasks – from distinguishing convex components on ShapeNet to classifying shapes on ModelNet40. Inspired by [25], we also investigated if adding a reconstruction component to the loss would further improve accuracy. Reconstruction is done by simply adding an AtlasNet [22] decoder to our model and using Chamfer distance as reconstruction loss. Without the reconstruction term (i.e. trained only to perform ACD using contrastive loss), our accuracy (89.1%) is the same as the multi-task learning approach presented in [25]. After adding a reconstruction term, we achieve an improved accuracy of **89.8**%. On the other hand, having just reconstruction without ACD yields an accuracy of 86.2%. This shows not only that ACD is a useful task when learning representations for shape classification, but that it can also be combined with shape reconstruction to yield complementary benefits.

Comparison with previous work. Approaches for *unsupervised* or *self-supervised* learning on point clouds are listed in the upper portion of Table 1. Our method achieves **89.1%** classification accuracy from purely using the ACD loss, which is met only by the unsupervised multi-task learning method of Hassani *et al.* [25] (adding a reconstruction loss to our method slightly improves over the state-of-the-art: 89.1% $\rightarrow$ **89.8%**). We note that our method merely adds a contrastive loss to a standard architecture (PointNet++), without requiring a custom architecture and multiple pretext tasks as in [25], which uses clustering, pseudo-labeling and reconstruction.

4.2 Few-shot segmentation on ShapeNet

Dataset. We report results on the **ShapeNetSeg** part segmentation benchmark [8], which is a subset of the ShapeNetCore database with manual annotations (train/val/test splits of 12,149/1,858/2,874). It consists of 16 man-made shape categories such as airplanes, chairs, and tables, with manually labeled semantic parts (50 in total), such as wings, tails, and engines for airplanes; legs, backs, and seats for chairs, and so on. Given a point cloud at test time, the goal is to assign each point its correct part label out of the 50 possible parts.

Table 1. Unsupervised shape classification on the ModelNet40 dataset. The representations learned in the intermediate layers by a network trained for the ACD task on ShapeNet data are general enough to be useful for discriminating between shape categories on ModelNet40.

Method	**Accuracy (%)**
VConv-DAE [52]	75.5
3D-GAN [72]	83.3
Latent-GAN [1]	85.7
MRTNet [19]	86.4
PointFlow [75]	86.8
FoldingNet [76]	88.4
PointCapsNet [80]	88.9
Multi-task [25]	89.1
Our baseline (with Random weights)	78.0
With reconstruction term only	86.2
Ours with ACD	89.1
Ours with ACD + Reconstruction	**89.8**

Table 2. Few-shot segmentation on the ShapeNet dataset (*class avg. IoU* over 5 rounds). The number of shots or samples per class is denoted by k for each of the 16 ShapeNet categories used for supervised training. Jointly training with the ACD task reduces overfitting when labeled data is scarce, leading to significantly better performance over a purely supervised baseline.

Samples/cls.	**k=1**	**k=3**	**k=5**	**k=10**
Baseline	53.15 ± 2.49	59.54 ± 1.49	68.14 ± 0.90	71.32 ± 0.52
w/ ACD	61.52 ± 2.19	69.33 ± 2.85	72.30 ± 1.80	74.12 ± 1.17
	k=20	**k=50**	**k=100**	**k=inf**
Baseline	75.22 ± 0.82	78.79 ± 0.44	79.67 ± 0.33	81.40 ± 0.44
w/ ACD	76.19 ± 1.18	78.67 ± 0.72	78.76 ± 0.61	81.57 ± 0.68

Few-shot learning tasks are typically described in terms of "n-way k-shot" – the task is to discriminate among n classes and k samples per class are provided as training data. We modify this approach to our setup as follows – we select k samples from each of the $n = 16$ shape categories as the labeled training data, while the task remains semantic part labeling over the 50 part categories.

Experimental setup. For this task, we perform *joint training* with two losses – the usual cross-entropy loss over labeled parts for the training samples from ShapeNetSeg, and an additional contrastive loss over the ACD components for the samples from ShapeNetCore (Eq. 5), setting $\lambda = 10$. In our initial experiments, we found joint training to be more helpful than pre-training on ACD and then fine-tuning on the few-shot task (an empirical phenomenon also noted in [74]), and thereafter consistently used joint training for the few-shot experi-

Table 3. Comparison with state-of-the-art semi-supervised part segmentation methods on ShapeNet. Performance is evaluated using *instance-averaged IoU*.

Method	1% labeled **IoU**	5% labeled **IoU**
SO-Net [36]	64.0	69.0
PointCapsNet [80]	67.0	70.0
MortonNet [61]	-	77.1
JointSSL [2][3]	71.9	77.4
Multi-task [25]	68.2	77.7
ACD (*ours*)	**75.7**	**79.7**

ments. All overlapping point clouds between the human-annotated ShapeNetSeg and the unlabeled ShapeNetCore were removed from the self-supervised training set. The (x, y, z) coordinates of the points in each point cloud are used an the input to the neural network; we do not include any additional information such as normals or category labels in these experiments.

Comparison with baselines. Table 2 shows the few-shot segmentation performance of our method, versus a fully-supervised baseline. Especially in the cases of very few labeled training samples ($k = 1, \ldots, 10$), having the ACD loss over a large unlabeled dataset provides a consistent and significant gain in performance over purely training on the labeled samples. As larger amounts of labeled training samples are made available, naturally there is limited benefit from the additional self-supervised loss – *e.g.* when using all the labeled data, our method is within standard deviation of the purely supervised baseline. Qualitative results are shown in Fig. 3.

Comparison with previous work. The performance of recent *unsupervised* and *self-supervised* methods on ShapeNet segmentation are listed in Table 3. Consistent with the protocol followed by the multi-task learning approach of Hassani *et al.* [25], we provide 1% and 5% of the training samples of ShapeNet-Seg as the labeled data and report instance-average IoU. Our method clearly outperforms the state-of-the-art unsupervised learning approaches, improving over [25] at both the 1% and 5% settings ($68.2 \rightarrow$ **75.7**% and $77.7 \rightarrow$ **79.7**%, respectively).

4.3 Analysis of ACD

Effect of backbone architectures. Differently from [11,25,76], the ACD self-supervision does not require any custom network design and should be easily applicable across various backbone architectures. To this end, we use two recent high-performing models – *PointNet++* (with multi-scale grouping [48])

[3] Concurrent work.

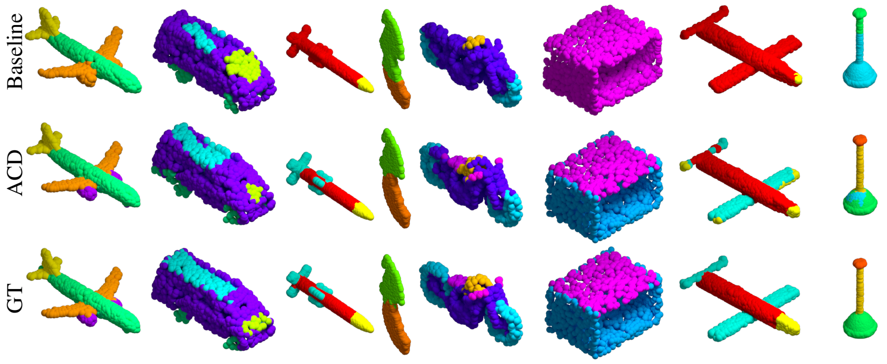

Fig. 3. Qualitative comparison on 5-shot ShapeNet [8] part segmentation. The baseline method in the first row corresponds to training using only 5 examples per class, whereas the ACD results in the second row were computed by performing joint training (cross-entropy from 5 examples + contrastive loss over ACD components from ShapeNetCore). The network backbone architecture is the same for both approaches – PointNet++ [48]. The baseline method merges parts that should be separated, *e.g.*, engines of the airplane, details of the rocket, top of the table, and seat of the motorcycle.

and *DGCNN* [70] – as the backbones, reporting results on ModelNet40 shape classification and few-shot segmentation ($k = 5$) on ShapeNetSeg (Table 4). On shape classification, both networks show large gains from ACD pre-training: 11% for PointNet++ (as reported earlier) and 14% for DGCNN. On few-shot segmentation with 5 samples per category (16 shape categories), PointNet++ improves from 68.14% IoU to 72.3% with the inclusion of the ACD loss. The baseline DGCNN performance with only 5 labeled samples per class is relatively lower (64.14%), however with the additional ACD loss on unlabeled samples, the model achieves 73.11% IoU, which is comparable to the corresponding PointNet++ performance (72.30%).

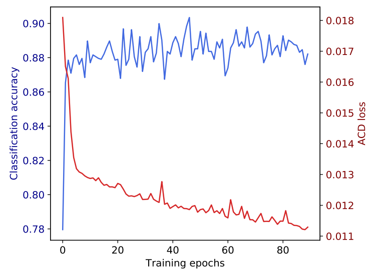

Fig. 4. Classification accuracy of a linear SVM on the ModelNet40 *validation set* v.s. the ACD *validation loss* over training epochs.

Role of ACD in shape classification. Fig. 4 shows the reduction in validation loss on learning ACD (red curve) as training progresses on the unlabeled ShapeNet data. Note that doing well on ACD (in terms of the validation loss) also leads to learning representations that are useful for the downstream tasks of shape classification (in terms of SVM accuracy on a validation subset of ModelNet40 data, shown in blue). However, the correlation between the two quantities is not very strong (Pearson $\rho = 0.667$) – from the plots it appears that after the initial epochs, where we observe a

Table 4. Comparing embeddings from PointNet++ [48] and DGCNN [70] backbones: shape classification accuracy on ModelNet40 (*Class./MN40*) and few-shot part segmentation performance in terms of class-averaged IoU on ShapeNet (*Part Seg./ShapeNet*).

Task / Dataset	**Method**	**PointNet++**	**DGCNN**
Class./MN40	Baseline	77.96	74.11
	w/ ACD	**89.06**	**88.21**
5-shot Seg./ShapeNet	Baseline	68.14 ± 0.90	64.14 ± 1.43
	w/ ACD	**72.30** ± 1.80	**73.11** ± 0.95

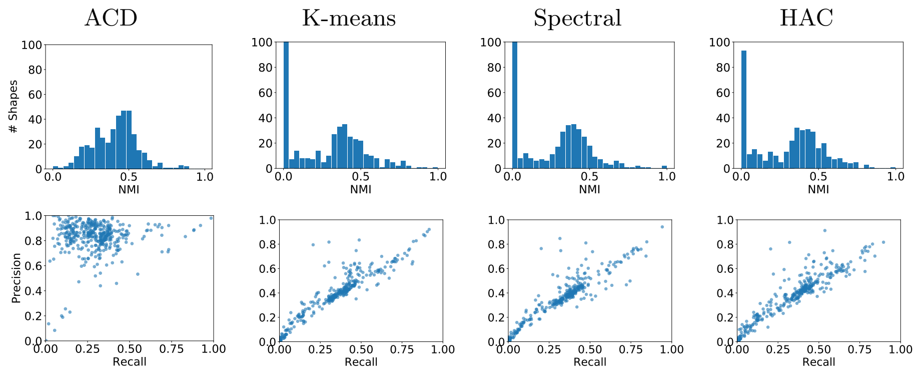

Fig. 5. Correspondence between human part labels and shape decompositions: comparing ACD with basic clustering algorithms – K-means, spectral clustering and hierarchical agglomerative clustering (HAC). ***Row-1:*** histogram of *normalized mutual information* (NMI) between human labels and clustering – ACD is closer to the ground-truth parts than others (y-axes clipped at 100 for clarity). ***Row-2:*** plotting *precision v.s. recall* for each input shape, ACD has high precision and moderate recall (tendency to over-segment parts), while other methods are usually lower in both metrics.

large gain in classification accuracy as well as a large reduction in ACD loss, continuing to be better at the pretext task does not lead to any noticeable gains in the ability to classify shapes: training with ACD gives the model some useful notion of grouping and parts, but it is not intuitively obvious if *perfectly* mimicking ACD will improve representations for classifying point-clouds into shape categories.

Comparison with clustering algorithms. We quantitatively analyse the connection between convex decompositions and semantic object parts by comparing ACD with human part annotations on 400 shapes from ShapeNet, along with simple clustering baselines – K-means [3], spectral clustering [54,65] and hierarchical agglomerative clustering (HAC) [41] on (x, y, z) coordinates of the point clouds. For the baselines, we set the number of clusters to be the number of ground-truth parts in each shape. For each sample shape, given the set of M part categories $\Omega = \{\omega_1, \omega_2, \dots \omega_M\}$ and the set of N clusters $\mathcal{C} = \{C_1, C_2, \dots C_N\}$, clustering performance is evaluated using *normalized mutual information* (N-

MI) [64], defined as

$$\mathrm{NMI}(\Omega, \mathcal{C}) = \frac{I(\Omega; \mathcal{C})}{[H(\Omega) + H(\mathcal{C})]/2}, \tag{6}$$

where $I(\cdot;\cdot)$ denotes the mutual information between classes Ω and clusters $\mathcal{C}$, and $H(\cdot)$ is the entropy [13]. A better clustering results in *higher* NMI w.r.t. the ground-truth part labels. The first row of Fig. 5 shows the histograms of NMI between cluster assignments and human part annotations: ACD, though not exactly aligned to human notions of parts, is significantly better than other clustering methods, which have very low NMI in most cases.

We also plot the *precision* and *recall* of clustering for each of the 400 shapes on the second row of Fig. 5. The other baseline methods show that a naïve clustering of points does not correspond well to semantic parts. ACD has high precision and moderate recall on most of the shapes – this agrees with the visual impression that the decompositions contain most of the boundaries present in the human annotations, even if ACD tends to oversegment the shapes. For example, ACD typically segments the legs of a chair into four separate components. On the other hand, the part annotations in ShapeNet label all the legs of a chair with the same label, since the benchmark does not distinguish between the individual legs of a chair. We note that the correspondence of ACD to human part labels is not perfect, and this opens an interesting avenue for further work – exploring other decomposition methods like generalized cylinders [81] that may correspond more closely to human-defined parts, and in turn could lead to improved downstream performance on discriminative tasks.

5 Conclusion

Self-supervision using approximate convex decompositions (ACD) has been shown to be effective across multiple tasks and datasets – few-shot part segmentation on ShapeNet and shape classification on ModelNet, consistently surpassing existing self-supervised and unsupervised methods in performance. A simple pairwise contrastive loss is sufficient for introducing the ACD task into a network training framework, without dependencies on any custom architectures. The method can be easily integrated into existing state-of-the-art architectures operating on point clouds such as PointNet++ and DGCNN, yielding significant improvements in both cases. Given the demonstrated effectiveness of ACD in self-supervision, this opens the door to incorporating other shape decomposition methods from the classical geometry processing literature into deep neural network models operating on point clouds.

Acknowledgements The project is supported in part by the National Science Foundation (NSF) through grants #1908669, #1749833, #1617333. Our experiments were performed in the UMass GPU cluster obtained under the Collaborative Fund managed by the Massachusetts Technology Collaborative.

References

1. Achlioptas, P., Diamanti, O., Mitliagkas, I., Guibas, L.: Representation learning and adversarial generation of 3d point clouds. arXiv preprint arXiv:1707.02392 (2017) 10
2. Alliegro, A., Boscaini, D., Tommasi, T.: Joint supervised and self-supervised learning for 3d real-world challenges (2020), `https://arxiv.org/abs/2004.07392` 11
3. Arthur, D., Vassilvitskii, S.: k-means++: The advantages of careful seeding. Tech. rep. (2007) 13
4. Au, O.K.C., Zheng, Y., Chen, M., Xu, P., Tai, C.L.: Mesh Segmentation with Concavity-Aware Fields. IEEE Trans. Visual. Comput. Graphics **18**(7), 1125–1134 (Jul 2011). https://doi.org/10.1109/TVCG.2011.131 3
5. Bernard, C.: Convex partitions of polyhedra: a lower bound and worst-case optimal algorithm. SIAM J. Comput. (Jul 1984), `https://dl.acm.org/doi/10.1137/0213031` 3, 6
6. Caron, M., Bojanowski, P., Joulin, A., Douze, M.: Deep clustering for unsupervised learning of visual features. In: Proceedings of the European Conference on Computer Vision (ECCV). pp. 132–149 (2018) 1
7. Caron, M., Bojanowski, P., Mairal, J., Joulin, A.: Unsupervised pre-training of image features on non-curated data. In: Proceedings of the IEEE International Conference on Computer Vision. pp. 2959–2968 (2019) 1
8. Chang, A.X., Funkhouser, T.A., Guibas, L.J., Hanrahan, P., Huang, Q.X., Li, Z., Savarese, S., Savva, M., Song, S., Su, H., Xiao, J., Yi, L., Yu, F.: Shapenet: An information-rich 3d model repository. CoRR **abs/1512.03012** (2015) 3, 5, 6, 8, 9, 12
9. Chen, X., Golovinskiy, A., Funkhouser, T.: A benchmark for 3D mesh segmentation. ACM Transactions on Graphics (Proc. SIGGRAPH) **28**(3) (Aug 2009) 3
10. Chen, Z., Tagliasacchi, A., Zhang, H.: Bsp-net: Generating compact meshes via binary space partitioning. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (2019) 3
11. Chen, Z., Yin, K., Fisher, M., Chaudhuri, S., Zhang, H.: Bae-net: branched autoencoder for shape co-segmentation. In: Proceedings of the IEEE International Conference on Computer Vision. pp. 8490–8499 (2019) 1, 4, 11
12. Chopra, S., Hadsell, R., LeCun, Y.: Learning a similarity metric discriminatively, with application to face verification. In: 2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05). vol. 1, pp. 539–546. IEEE (2005) 2
13. Cover, T.M., Thomas, J.A.: Elements of information theory. John Wiley & Sons (2012) 14
14. lke Demir, Aliaga, D.G., Benes, B.: Near-convex decomposition and layering for efficient 3d printing. Additive Manufacturing **21**, 383 – 394 (2018). https://doi.org/https://doi.org/10.1016/j.addma.2018.03.008, `http://www.sciencedirect.com/science/article/pii/S2214860417300386` 2
15. Deng, B., Genova, K., Yazdani, S., Bouaziz, S., Hinton, G., Tagliasacchi, A.: Cvxnet: Learnable convex decomposition. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (2020) 3
16. Doersch, C., Gupta, A., Efros, A.A.: Unsupervised visual representation learning by context prediction. In: Proceedings of the IEEE International Conference on Computer Vision. pp. 1422–1430 (2015) 4

17. Donahue, J., Simonyan, K.: Large scale adversarial representation learning. In: Advances in Neural Information Processing Systems. pp. 10541–10551 (2019) 4
18. Gadelha, M., Maji, S., Wang, R.: 3d shape generation using spatially ordered point clouds. In: British Machine Vision Conference (BMVC) (2017) 3
19. Gadelha, M., Wang, R., Maji, S.: Multiresolution Tree Networks for 3D Point Cloud Processing. In: ECCV (2018) 1, 3, 10
20. Ghosh, M., Amato, N.M., Lu, Y., Lien, J.M.: Fast approximate convex decomposition using relative concavity. Compututer Aided Deisgn. **45**, 494–504 (Feb 2013) 6
21. Goyal, P., Mahajan, D., Gupta, A., Misra, I.: Scaling and benchmarking self-supervised visual representation learning. arXiv preprint arXiv:1905.01235 (2019) 4
22. Groueix, T., Fisher, M., Kim, V.G., Russell, B., Aubry, M.: AtlasNet: A Papier-Mâché Approach to Learning 3D Surface Generation. In: Proceedings IEEE Conf. on Computer Vision and Pattern Recognition (CVPR) (2018) 9
23. Gupta, S., Hoffman, J., Malik, J.: Cross modal distillation for supervision transfer. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 2827–2836 (2016) 4
24. Hadsell, R., Chopra, S., LeCun, Y.: Dimensionality reduction by learning an invariant mapping. In: 2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'06). vol. 2, pp. 1735–1742. IEEE (2006) 2, 7
25. Hassani, K., Haley, M.: Unsupervised multi-task feature learning on point clouds. In: Proceedings of the IEEE International Conference on Computer Vision. pp. 8160–8171 (2019) 1, 3, 4, 5, 8, 9, 10, 11
26. Hoffman, D.D., Richards, W.: Parts of recognition (1983) 3
27. Huang, H., Kalogerakis, E., Chaudhuri, S., Ceylan, D., Kim, V.G., Yumer, E.: Learning local shape descriptors from part correspondences with multiview convolutional networks. ACM Transactions on Graphics **37**(1) (2018) 3
28. Huang, J., Yagel, R., Filippov, V., Kurzion, Y.: An accurate method for voxelizing polygon meshes. IEEE Symposium on Volume Visualization (Oct 1998) 6
29. Jiang, H., Larsson, G., Maire Greg Shakhnarovich, M., Learned-Miller, E.: Self-supervised relative depth learning for urban scene understanding. In: Proceedings of the European Conference on Computer Vision (ECCV). pp. 19–35 4
30. Kaick, O.V., Fish, N., Kleiman, Y., Asafi, S., Cohen-OR, D.: Shape segmentation by approximate convexity analysis. ACM Trans. Graph. **34**(1) (2014) 2, 3, 6
31. Kalogerakis, E., Averkiou, M., Maji, S., Chaudhuri, S.: 3D shape segmentation with projective convolutional networks. In: Proc. CVPR (2017) 3
32. Klokov, R., Lempitsky, V.: Escape from cells: Deep Kd-Networks for the recognition of 3D point cloud models. In: Proc. ICCV (2017) 3
33. Kolesnikov, A., Zhai, X., Beyer, L.: Revisiting self-supervised visual representation learning. arXiv preprint arXiv:1901.09005 (2019) 4
34. Kong, S., Fowlkes, C.C.: Recurrent pixel embedding for instance grouping. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 9018–9028 (2018) 7
35. Larsson, G., Maire, M., Shakhnarovich, G.: Learning representations for automatic colorization. In: European Conference on Computer Vision. pp. 577–593. Springer (2016) 4
36. Li, J., Chen, B.M., Hee Lee, G.: So-net: Self-organizing network for point cloud analysis. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 9397–9406 (2018) 11

37. Lien, J.M., Amato, N.M.: Approximate convex decomposition of polyhedra. In: Proceedings of the 2007 ACM Symposium on Solid and Physical Modeling. SPM '07 (2007) 2, 3, 6
38. Luo, L., Baran, I., Rusinkiewicz, S., Matusik, W.: Chopper: Partitioning models into 3d-printable parts. ACM Trans. Graph. **31**(6) (2012) 2
39. Mamou, K.: Volumetric approximate convex decomposition. In: Lengyel, E. (ed.) Game Engine Gems 3, chap. 12, pp. 141–158. A K Peters / CRC Press (2016) 3, 5, 6, 7, 8
40. Maturana, D., Scherer, S.: 3D convolutional neural networks for landing zone detection from LiDAR. In: Proc. ICRA (2015) 3
41. Müllner, D., et al.: fastcluster: Fast hierarchical, agglomerative clustering routines for r and python. Journal of Statistical Software **53**(9), 1–18 (2013) 13
42. Muralikrishnan, S., Kim, V.G., Chaudhuri, S.: Tags2parts: Discovering semantic regions from shape tags. CoRR **abs/1708.06673** (2017) 1
43. Muralikrishnan, S., Kim, V.G., Chaudhuri, S.: Tags2Parts: Discovering semantic regions from shape tags. In: Proc. CVPR. IEEE (2018) 4
44. Noroozi, M., Favaro, P.: Unsupervised learning of visual representations by solving jigsaw puzzles. In: European Conference on Computer Vision. pp. 69–84. Springer (2016) 1, 4
45. Pathak, D., Girshick, R., Dollár, P., Darrell, T., Hariharan, B.: Learning features by watching objects move. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 2701–2710 (2017) 4
46. Pathak, D., Krahenbuhl, P., Donahue, J., Darrell, T., Efros, A.A.: Context encoders: Feature learning by inpainting. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 2536–2544 (2016) 4
47. Qi, C.R., Su, H., Mo, K., Guibas, L.J.: PointNet: Deep learning on point sets for 3D classification and segmentation. In: Proc. CVPR (2017) 3
48. Qi, C.R., Yi, L., Su, H., Guibas, L.: PointNet++: Deep hierarchical feature learning on point sets in a metric space. In: Proc. NIPS (2017) 3, 7, 11, 12, 13
49. Riegler, G., Ulusoys, A.O., Geiger, A.: Octnet: Learning deep 3D representations at high resolutions. In: Proc. CVPR (2017) 3
50. Sánchez, J., Perronnin, F., Mensink, T., Verbeek, J.: Image classification with the fisher vector: Theory and practice. International journal of computer vision **105**(3), 222–245 (2013) 9
51. Saxe, A.M., Koh, P.W., Chen, Z., Bhand, M., Suresh, B., Ng, A.Y.: On random weights and unsupervised feature learning. In: ICML. vol. 2, p. 6 (2011) 9
52. Sharma, A., Grau, O., Fritz, M.: Vconv-dae: Deep volumetric shape learning without object labels. In: European Conference on Computer Vision. pp. 236–250. Springer (2016) 4, 10
53. Sharma, G., Kalogerakis, E., Maji, S.: Learning point embeddings from shape repositories for few-shot segmentation. In: 2019 International Conference on 3D Vision (3DV). pp. 67–75 (2019) 1, 4, 5
54. Shi, J., Malik, J.: Normalized cuts and image segmentation. IEEE Transactions on pattern analysis and machine intelligence **22**(8), 888–905 (2000) 13
55. Su, H., Jampani, V., Sun, D., Maji, S., Kalogerakis, E., Yang, M.H., Kautz, J.: SPLATNet: Sparse lattice networks for point cloud processing. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (2018) 3
56. Su, H., Maji, S., Kalogerakis, E., Learned-Miller, E.G.: Multi-view convolutional neural networks for 3d shape recognition. In: Proc. ICCV (2015) 3
57. Su, H., Qi, C., Mo, K., Guibas, L.: PointNet: Deep Learning on Point Sets for 3D Classification and Segmentation. In: CVPR (2017) 7

58. Su, J.C., Gadelha, M., Wang, R., Maji, S.: A deeper look at 3d shape classifiers. In: Proceedings of the European Conference on Computer Vision Workshops (ECCV) (2018) 3, 4
59. Su, J.C., Maji, S., Hariharan, B.: When does self-supervision improve few-shot learning? arXiv preprint arXiv:1910.03560 (2019) 4
60. Tatarchenko, M., Park, J., Koltun, V., Zhou., Q.Y.: Tangent convolutions for dense prediction in 3D. CVPR (2018) 3
61. Thabet, A., Alwassel, H., Ghanem, B.: MortonNet: Self-Supervised Learning of Local Features in 3D Point Clouds. arXiv (Mar 2019), https://arxiv.org/abs/1904.00230 11
62. Trimble Inc.: Trimble 3D Warehouse (2008), https://3dwarehouse.sketchup.com/ 5
63. Trinh, T.H., Luong, M.T., Le, Q.V.: Selfie: Self-supervised pretraining for image embedding. arXiv preprint arXiv:1906.02940 (2019) 4
64. Vinh, N.X., Epps, J., Bailey, J.: Information theoretic measures for clusterings comparison: Variants, properties, normalization and correction for chance. Journal of Machine Learning Research **11**(Oct), 2837–2854 (2010) 14
65. Von Luxburg, U.: A tutorial on spectral clustering. Statistics and computing **17**(4), 395–416 (2007) 13
66. Wang, P.S., Liu, Y., Guo, Y.X., Sun, C.Y., Tong, X.: O-CNN: Octree-based convolutional neural networks for 3D shape analysis. ACM Trans. Graph. **36**(4) (2017) 3
67. Wang, P.S., Sun, C.Y., Liu, Y., Tong, X.: Adaptive o-cnn: A patch-based deep representation of 3d shapes. ACM Trans. Graph. **37**(6) (2018) 3
68. Wang, X., Gupta, A.: Unsupervised learning of visual representations using videos. In: Proceedings of the IEEE International Conference on Computer Vision. pp. 2794–2802 (2015) 4
69. Wang, X., He, K., Gupta, A.: Transitive invariance for self-supervised visual representation learning. In: Proceedings of the IEEE international conference on computer vision. pp. 1329–1338 (2017) 4
70. Wang, Y., Sun, Y., Liu, Z., Sarma, S.E., Bronstein, M.M., Solomon, J.M.: Dynamic graph cnn for learning on point clouds. ACM Transactions on Graphics (TOG) **38**(5), 1–12 (2019) 3, 12, 13
71. Weller, R.: A Brief Overview of Collision Detection. SpringerLink pp. 9–46 (2013) 3
72. Wu, J., Zhang, C., Xue, T., Freeman, B., Tenenbaum, J.: Learning a probabilistic latent space of object shapes via 3d generative-adversarial modeling. In: Advances in neural information processing systems. pp. 82–90 (2016) 10
73. Wu, Z., Song, S., Khosla, A., Yu, F., Zhang, L., Tang, X., Xiao, J.: 3d shapenets: A deep representation for volumetric shapes. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 1912–1920 (2015) 2, 3
74. Xie, Q., Hovy, E., Luong, M.T., Le, Q.V.: Self-training with noisy student improves imagenet classification. arXiv preprint arXiv:1911.04252 (2019) 10
75. Yang, G., Huang, X., Hao, Z., Liu, M.Y., Belongie, S., Hariharan, B.: Pointflow: 3d point cloud generation with continuous normalizing flows. In: Proceedings of the IEEE International Conference on Computer Vision. pp. 4541–4550 (2019) 1, 3, 10
76. Yang, Y., Feng, C., Shen, Y., Tian, D.: Foldingnet: Point cloud auto-encoder via deep grid deformation. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 206–215 (2018) 1, 4, 8, 10, 11

77. Yi, L., Guibas, L., Hertzmann, A., Kim, V.G., Su, H., Yumer, E.: Learning hierarchical shape segmentation and labeling from online repositories. ACM Trans. Graph. **36** (Jul 2017) 4
78. Zhang, R., Isola, P., Efros, A.A.: Colorful image colorization. In: European conference on computer vision. pp. 649–666. Springer (2016) 1, 4
79. Zhang, R., Isola, P., Efros, A.A.: Split-brain autoencoders: Unsupervised learning by cross-channel prediction. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 1058–1067 (2017) 4
80. Zhao, Y., Birdal, T., Deng, H., Tombari, F.: 3d point capsule networks. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 1009–1018 (2019) 10, 11
81. Zhou, Y., Yin, K., Huang, H., Zhang, H., Gong, M., Cohen-Or, D.: Generalized cylinder decomposition. ACM Trans. Graph. **34**(6) (2015) 14
82. Zhou Ren, Junsong Yuan, Chunyuan Li, Wenyu Liu: Minimum near-convex decomposition for robust shape representation. In: 2011 International Conference on Computer Vision (Nov 2011) 3, 6
83. Zhu, C., Xu, K., Chaudhuri, S., Yi, L., Guibas, L.J., Zhang, H.: Adacoseg: Adaptive shape co-segmentation with group consistency loss. CoRR **abs/1903.10297** (2019), `http://arxiv.org/abs/1903.10297` 1, 4