

# Primitive3D: 3D Object Dataset Synthesis from Randomly Assembled Primitives

Xinke Li<sup>1,\*</sup> Henghui Ding<sup>2,3,\*</sup> Zekun Tong<sup>1</sup> Yuwei Wu<sup>1,†</sup> Yeow Meng Chee<sup>1</sup>

<sup>1</sup>National University of Singapore <sup>2</sup>ByteDance <sup>3</sup>ETH Zürich

{xinke.li, zekuntong}@u.nus.edu henghui.ding@vision.ee.ethz.ch {wyw, ymchee}@nus.edu.sg

## Abstract

Numerous advancements in deep learning can be attributed to the access to large-scale and well-annotated datasets. However, such a dataset is prohibitively expensive in 3D computer vision due to the substantial collection cost. To alleviate this issue, we propose a cost-effective method for automatically generating a large amount of 3D objects with annotations. In particular, we synthesize objects simply by assembling multiple random primitives. These objects are thus auto-annotated with part labels originating from primitives. This allows us to perform multi-task learning by combining the supervised segmentation with unsupervised reconstruction. Considering the large overhead of learning on the generated dataset, we further propose a dataset distillation strategy to remove redundant samples regarding a target dataset. We conduct extensive experiments for the downstream tasks of 3D object classification. The results indicate that our dataset, together with multi-task pretraining on its annotations, achieves the best performance compared to other commonly used datasets. Further study suggests that our strategy can improve the model performance by pretraining and fine-tuning scheme, especially for the dataset with a small scale. In addition, pretraining with the proposed dataset distillation method can save 86% of the pretraining time with negligible performance degradation. We expect that our attempt provides a new data-centric perspective for training 3D deep models.

## 1. Introduction

Deep learning has been shown to surpass prior state-of-the-art machine learning techniques in applications of 2D computer vision in the past several years [17–19]. Such success is often attributed to the ease of acquiring large-scale, richly-annotated and diverse 2D image datasets, e.g., ImageNet [16] and COCO [35]. In the field of 3D, a broad variety of applications, such as self-driving vehicles [9], augmented reality [22], and urban construction [7], can ben-




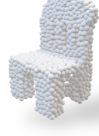
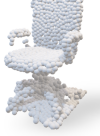
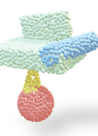
|                 | CAD Model  | Real Scan   | Primitive3D   |
|-----------------|--|---|---|
| 3D Object       |  |  |  |
| Point Cloud     |  |  |  |
| Annotation      | Instance-wise  | Instance-wise   | Point-wise  |
| Collection Cost | Moderate   | High  | Very low  |
| Diversity       | Moderate   | Low   | High  |

Figure 1. Compared to the CAD model and real-world data, the Primitive3D dataset has richer annotations, lower collection cost, and higher sample diversity. These advantages help facilitate the learning of general knowledge for understanding 3D objects.

efit from advancing 3D object understanding, which has long awaited high-quality datasets. Indeed, unlike the 2D counterparts, existing 3D object datasets are often limited in scale or lack of variety in annotation and instance diversity, thus hindering the advances of many applications.

The main reason for the limitation of 3D object datasets is the substantial cost related to data collection and annotation. In the current practice of real-world 3D object data acquisition, LiDAR or RGB-D scans often necessitate considerable labour and device costs [10, 14, 28], while image-derived data requires much computation efforts [33, 51]. On the other hand, although synthesis datasets often consist of 3D CAD models with rich online data sources [6, 31, 63], they still involve manual design works and also have limited generalization for real-world objects recognition [55]. Ultimately, tremendous human efforts have to be made in the annotation and maintenance of large-scale labeled datasets (e.g., voxel or point based label), especially considering the bulky size and the extra dimension of 3D data compared to 2D. To alleviate these problems, attempts have been made to automatically generate 3D objects via generative models built from existing datasets [2, 29, 40, 60]. Although these approaches are capable of generating high-quality 3D data

\*Equal contribution, Henghui was Xinke’s internship mentor.

†Corresponding author.

with fewer human interactions, they are task-oriented, computationally intensive, and may potentially incorporate bias from the training dataset. Therefore, it is critical to find a cheap 3D object data source as an alternative, which can derive large-scale, diverse and richly-annotated datasets, to push forward the development of 3D deep learning.

In our work, to obtain annotated 3D data at low cost, we introduce a learning-free method for the synthesis of pseudo 3D objects. Motivated by Constructive Solid Geometry (CSG) scheme [48], we build a vast number of random objects from basic 3D shapes, *i.e.*, primitives. In a typical CSG scheme, 3D solids are often constructed with a tree representation, where the leaves are primitives and internal nodes are boolean set operations. We randomize this tree-based construction by setting the parameters including tree structure, primitive parameters, boolean operations and rigid transformations as random variables. By uniformly sampling such variables, we derive a random tree, and a random object can be obtained by executing this tree from bottom to top. By tracking the original primitives after the construction, the object is automatically annotated with part-based labels. Furthermore, our analysis shows that such a method can generate objects with sufficient diversity, which allows the deep models to learn generalized representations from the resulting large-scale 3D dataset.

To exploit the abundance and annotations of the generated objects, we provide a multi-task learning method for the point cloud of such objects. To grasp the object geometry on a local and global level, the method combines two learning tasks: *supervised segmentation* of point clouds and *unsupervised reconstruction* of the original points. Regarding the tremendous size of the generated dataset, we then present a method, dataset distillation, that is integrated into the learning process to relieve the computational burden. This method is accomplished by eliminating certain samples from the generated dataset to shrink its maximum mean discrepancy (MMD) to a target dataset.

In experiments, we evaluate the validity of our method by multiple object classification benchmarks. The results of the cross-dataset classification show that the features learned from our dataset can surpass those learned from other widely used 3D object datasets. Furthermore, pre-training by our method combined with fine-tuning can consistently improve the performance of downstream tasks with varying data size. Additionally, pretraining with dataset distillation can obtain comparable or even better performance than that with full dataset, reducing 86% of the pretraining time. In summary, our contributions are:

1. We present a cost-efficient method to generate a large amount of valid and diverse random 3D objects with part annotations automatically. The generated dataset as well as generation scripts will be released publicly.
2. We provide a multi-task learning method to train fea-

ture encoders on our generated dataset with dense annotations. We also propose an approach, dataset distillation, that can be optionally employed in the learning process to lower the computation cost.

3. Experiments show that our dataset serves as the best pretraining data for multiple downstream classification tasks in comparison to other commonly used datasets. Our pretraining method can also consistently help the downstream tasks in achieving higher performance.

## 2. Related Work

### 2.1. Dataset for 3D Object Understanding

In general, 3D object datasets can be divided into two categories: (1) *synthesis dataset*: the datasets of synthetic objects are often derived from CAD models, which can be manually designed or collected from websites. The most commonly used and general datasets are ModelNet [63] and ShapeNet [6], as well as others for the particular usage, *e.g.*, [30, 31] for mechanical design. Despite the relatively large scale of these synthesis datasets, research indicates that models built from them may not necessarily generalize to recognize real objects [55]. (2) *real-world dataset*: there are a few object datasets from real-world scans [14, 55], but most are miniature in scale due to the high cost of the scanning and reconstruction process. Besides the synthesis and real-world datasets, many works generate 3D shapes by deep models, like Deep Belief Networks [63], GAN [2, 60], VAE [40, 64] and Flow [66]. However, the generative models often suffer from considerable learning costs and the limitation of sample diversity. In addition, most of these datasets are not point-wisely labeled, except for [39, 69]. Overall, point-wise annotations are difficult and costly to acquire, and the demand for high-quality 3D object datasets with extensive annotations and diversified data is still great.

### 2.2. Deep Learning for 3D Object Understanding

**Deep Models.** Countless attempts have been made to apply deep learning models on a variety of 3D modalities, including voxels [38], mesh [25], multi-view [52], and point cloud [43, 44]. Among these works, the point cloud-based method has received the most attention for its simplicity. Recently an increasing number of such models [5, 34, 36, 59, 62] have shown their powers for 3D object understanding.

**Model Pretraining.** Recent researches on deep model pretraining for understanding 3D objects mainly focus on unsupervised learning methods. Typical unsupervised tasks include self-reconstruction [2, 20, 24, 26, 67] and self-supervised learning pretext tasks [21, 46, 49, 57, 65]. Most of these works rely on delicately designed pretraining schemes rather than unlocking the potential of the pretraining dataset. In contrast, we pay more attention to the automatic generation and annotation of pretraining datasets.

**Domain Adaption.** Considering the domain gaps between different 3D object datasets, our work is also related to the field of domain adaption (DA) [4, 11, 12, 37]. Indeed, there are efforts to implement domain adaption in 3D objects understanding, such as PointDAN [45] and SSL-DA [1]. However, their approaches are proposed for model-centric DA, while we design our DA method from the perspective of the dataset, as specified in Section 4.2.

### 2.3. Primitives in 3D Deep Learning

Some latest researches have investigated decomposing or fitting 3D objects into primitive representations by deep learning approach [15, 32, 41, 50, 54, 68]. Other methods related to primitives focus on deep shape generation models, where the primitives act as intermediate representations or building blocks [29, 61, 70]. These works demonstrate the representation ability of assembled primitives, which inspires our usage of primitives to generate 3D object data.

## 3. Dataset Construction

### 3.1. Data Generation with Random Primitives

We start with the building blocks of the CSG scheme, namely, the primitives. In CSG-based modeling, the primitive is defined by a type and a set of parameters. The primitive type  $\Psi_i$  is a basic shape, such as a box, sphere, cylinder, cone, and etc [48]. Each type  $\Psi_i$  contains a collection of objects  $\psi_\theta$  with the same shape in the canonical form, say centered at an original point and bounded by a unit ball. Specifically, a canonical instance  $\psi_\theta$  is parameterized by  $\theta \in \Theta^{\Psi_i}$ , for example, the height, width, and depth of a box. Moreover, a particular primitive instance  $\psi'_\theta$  can be obtained from  $\psi_\theta$  by

$$\psi'_\theta = \lambda(\Phi\psi_\theta) + \delta, \quad (1)$$

where  $\Phi \in SO(3)$  is a rotation transformation,  $\delta \in \mathbb{R}^3$  is a translation vector and  $\lambda \in \mathbb{R}^+$  is a scaling factor.

Based on the primitive instances, we design a randomized method to create complicated pseudo objects via the CSG scheme. CSG construction is often represented as a binary tree in which the leaf nodes are primitive instances, while the internal nodes are boolean set operations applied to the immediate children [48]. To obtain random objects, our method is to randomize such a construction process as *Randomized Constructive Tree (RCT)*. The RCT treats each parameter, including the tree structure, as a random variable, such as a uniform distribution, see Algorithm 1. Particularly, we first specify the domain of each RCT parameter, including a finite set of primitive type  $\mathbb{P} = \{\Psi_1, \dots, \Psi_p\}$ , a collection of parameter sets  $\{\Theta^{\Psi_1}, \dots, \Theta^{\Psi_p}, \Lambda\}$  where each element is compact, and a set of boolean operations  $\mathbb{O}$ . Such setups allow us to sample each parameter uniformly. To sample the tree structure with  $l$  leaves uniformly, we obtain the binary tree  $\tau_{2l-1}$  by

Rémy’s algorithm [3]. In addition, a sampled translation  $\delta$  is applied to the left child of the internal node, ensuring that the boolean operation can hardly return an *empty* object.

Compared to learning-based data generation approaches, we simply sample each parameter uniformly within the feasible set, avoiding learning costs and bias from the learned dataset. In this randomized manner, we could generate an unlimited number of distinct pseudo 3D objects at low cost. We refer to the set of these objects as *Primitive3D*. Moreover, because it is easy to track each part of the object to the original primitive, part-based annotations can be created automatically based on the primitive types and instances.

---

#### Algorithm 1: Data Generation

---

**Input:** leaf number  $l$ , type set  $\mathbb{P}$ , parameters set  $\{\Theta\}$ , scale range  $\Lambda$   
boolean operation set  $\mathbb{O}$   
**Output:** Object  $\psi_l$

```

1:  $\tau_{2l-1} \leftarrow \text{RandomBinaryTree}(2n - 1)$ 
2: for  $E_i$  in leaves of  $\tau_{2n-1}$  do
3:   sample primitive:  $\Psi_i \sim \text{Uniform}(\mathbb{P})$ ,  $\theta_i \sim \text{Uniform}(\Theta^{\Psi_i})$ 
4:   sample transform:  $\Phi_i \sim \text{Uniform}(SO(3))$ ,  $\lambda_i \sim \text{Uniform}(\Lambda)$ 
5:    $\psi_{\theta_i} \leftarrow \text{generate}(\Psi_i, \theta_i)$ 
6:    $E_i \leftarrow \lambda_i(\Phi_i\psi_{\theta_i})$ 
7: end for
8: for  $I_j$  in internal nodes of  $\tau_{2n-1}$  by bottom-up do
9:   sample operation:  $\circ \sim \text{Uniform}(\mathbb{O})$ 
10:  sample point from child:  $p_j^l \sim \text{Uniform}(\psi_j^l)$ ,  $p_j^r \sim \text{Uniform}(\psi_j^r)$ 
11:   $\delta_j \leftarrow p_j^r - p_j^l$ 
12:   $I_j \leftarrow \text{execute}(\psi_j^l \circ (\psi_j^r + \delta_j))$ 
13: end for
14:  $\psi_l \leftarrow \text{root node of } \tau_{2l-1}$ 

```

---

### 3.2. Analysis of Randomized Constructive Tree

In this subsection, we illustrate the capability of RCT by its closure property and approximability to arbitrary 3D objects. 3D objects in solid modeling are often defined as bounded, closed, regular, and semi-analytic subsets of  $\mathbb{R}^3$ , and their sets are referred to as *r-sets* [47]. Furthermore, the *regularized boolean operations* are also adapted from the conventional ones for processing r-sets [47]. We note the following property of RCT with above definitions.

**LEMMA 1 (Closure Property of RCT)** *If the leaf nodes of an RCT are r-sets and the internal nodes are regularized boolean operations, the sample of such RCT always corresponds to an r-set.*

The Lemma 1 ensures that every RCT sample can produce a valid 3D object when using r-set primitives, which is because the class of r-set is closed under regularized operations [27]. Additionally, RCT samples associated with properly defined primitive types can approximate any complex object. Such property suggests the sufficient diversity of RCT samples when conducting random sampling. We note that Appendix A.2 has a detailed description of this approximability as well as more definitions related to the r-sets theory. The above properties of RCT enable us to generate massive Primitive3D objects in a random fashion while meeting both the volume and diversity demands.

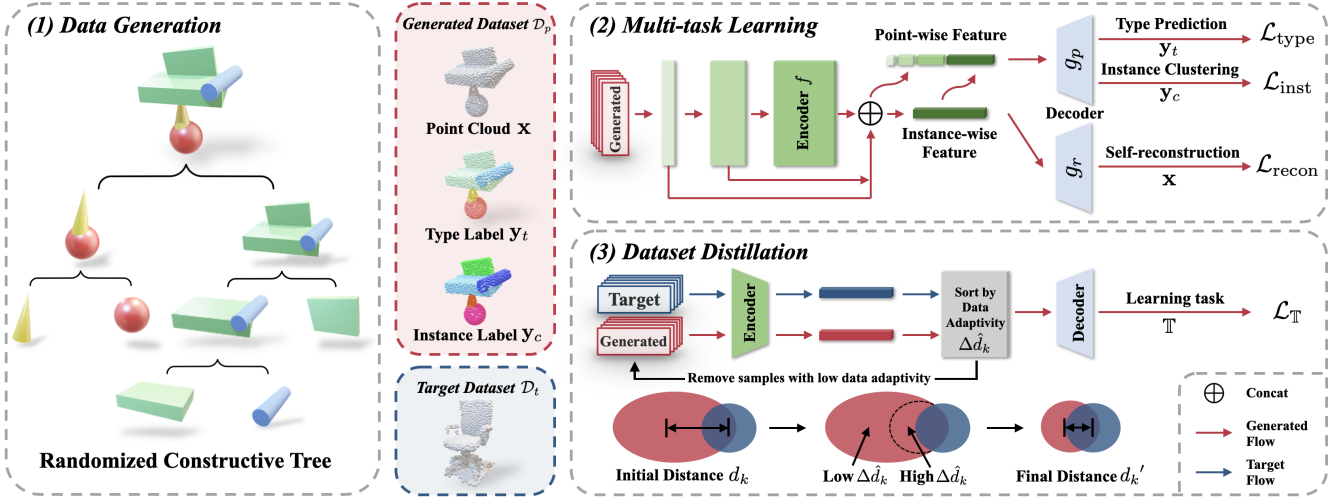


Figure 2. Data generation of Primitive3D and its learning.

## 4. Learning from Primitive3D

To take advantage of Primitive3D’s annotation and enormous volume, we provide a multi-task learning method on the point clouds of Primitive3D objects, as shown in Figure 2. Also, a dataset distillation method is designed for incorporation into the learning.

### 4.1. Multi-task Learning

A point cloud dataset  $\mathcal{D}_p$  can be constructed by collecting  $N$  sampled points  $\mathbf{x} \in \mathbb{X} = \mathbb{R}^{N \times 3}$  from the surface of each Primitive3D object. Each point in the point cloud is associated with a semantic label  $\mathbf{y}_t$  and an instance label  $\mathbf{y}_c$ , which describe the primitive type and instance information, respectively. On the basis of such labels, we propose a multi-task learning that combines multiple levels of information to train a shared encoder  $f$ . The learning procedure is depicted in Figure 2, while the associated tasks are summarized in Table 1. Then we detail each learning task.

| Training task         | Data description         | Groundtruth    |
|-----------------------|--------------------------|----------------|
| Semantic segmentation | Primitive type label     | $\mathbf{y}_t$ |
| Instance segmentation | Primitive instance label | $\mathbf{y}_c$ |
| Reconstruction        | Point cloud              | $\mathbf{x}$   |

Table 1. Learning tasks on Primitive3D.

We define two supervised segmentation tasks based on the point-wise labels  $\mathbf{y}_t$  and  $\mathbf{y}_c$ . Specifically, a point-based decoder  $g_p$  is first employed on the point-wise feature output by the encoder. The segmentation tasks are conducted on the embeddings of  $g_p$ , with the loss being the summation of the losses of two branching tasks as:

$$\mathcal{L}_{\text{seg}} = \mathcal{L}_{\text{type}} + \alpha \cdot \mathcal{L}_{\text{inst}}, \quad (2)$$

where  $\alpha$  is a tunable weight,  $\mathcal{L}_{\text{type}}$  is the cross-entropy loss for semantic segmentation of predicting  $\mathbf{y}_t$ , and  $\mathcal{L}_{\text{inst}}$  is the discriminative loss for instance segmentation used by [42, 58]. Instead of predicting the instance masks,  $\mathcal{L}_{\text{inst}}$

aims to learn point-based embeddings for the class-agnostic clustering, where the ground truth of each cluster is a primitive instance.

Self-reconstruction is implemented as the unsupervised task based on autoencoder structure, with the goal of reconstructing the original points. Particularly, a folding-based decoder [67]  $g_r$  is adapted to deform the canonical 2D grid into new 3D point coordinates  $\mathbf{x}'$  conditioned on the global feature output from the encoder  $f$ . The task loss is defined as the augmented Chamfer distance between the input  $\mathbf{x}$  and the reconstructed  $\mathbf{x}'$  in Equation (3), which is more robust under some ill cases [8] than the original version.

$$\mathcal{L}_{\text{recon}} = \max \left( \sum_{x_i \in \mathbf{x}} \min_{x'_j \in \mathbf{x}'} \|x_i - x'_j\|_2, \sum_{x'_j \in \mathbf{x}'} \min_{x_i \in \mathbf{x}} \|x_i - x'_j\|_2 \right). \quad (3)$$

Overall, the total loss is the weighted sum of all task losses

$$\mathcal{L} = \mathcal{L}_{\text{seg}} + \beta \cdot \mathcal{L}_{\text{recon}}. \quad (4)$$

We note that the segmentation task enables supervised training of the model’s ability to comprehend parts of Primitive3D object, while the unsupervised learning guarantees that the model retains global geometric knowledge. Their combination allows the learning of generalized feature representations from the Primitive3D dataset that can be used for both local and global understanding of 3D objects.

### 4.2. Dataset Distillation

Although learning on a large-scale dataset conceivably produces more general feature representations, the computational cost of such a learning process is also considerable. To improve the learning efficiency, we propose a method for reducing the Primitive3D data in order to learn only the features associated with a target dataset. This process of reducing the learning data according to the target dataset is referred to as *dataset distillation*.



**A Bound from Domain Adaption.** To motivate our method of dataset distillation, we first introduce a learning bound from the domain adaption theory. Let  $\mathcal{D}_p$  and  $\mathcal{D}_t$  denote the generated dataset for the feature learning task and the target dataset, in which the data are sampled *i.i.d* from domain  $\mathcal{P}$  and  $\mathcal{T}$ , respectively. When the data domains  $\mathcal{P}$  and  $\mathcal{T}$  are distinct but relevant, according to the theory in [4, 37], by defining a hypothesis  $h \in \mathcal{H}$ , the expected risk of  $h$  on target domain  $\epsilon_{\mathcal{T}}(h)$  can be bounded by the expected risk  $\epsilon_{\mathcal{P}}(h)$  on the  $\mathcal{P}$  and the distance between  $\mathcal{P}$  and  $\mathcal{T}$ , *i.e.*,

$$\epsilon_{\mathcal{T}}(h) \leq \epsilon_{\mathcal{P}}(h) + 2d_k(\mathcal{P}, \mathcal{T}) + C_0, \quad (5)$$

where  $C_0$  is a constant associated with  $\mathcal{H}$ , and  $d_k$  is the maximum mean discrepancy (MMD) between two domains. As is well-known in [56], decreasing the scale of the dataset  $\mathcal{D}_p$  will increase the empirical upper bound of expected risk  $\epsilon_{\mathcal{P}}(h)$  generally, thus raise the bound of  $\epsilon_{\mathcal{T}}(h)$ . To compensate such growth, the second term  $d_k$  in Equation (5) should be suppressed when removing samples from  $\mathcal{D}_p$ , which motivates our strategy of dataset distillation. In the following, we define a new metric, *data adaptivity*, to remove samples from  $\mathcal{D}_p$  in order to reduce the distance  $d_k$ . Then we propose our dataset distillation based on it.

**Distillation by Data Adaptivity.** Our definition of data adaptivity is simply the difference between the dataset distances before and after removing a subset of samples

**DEFINITION 1 (Data Adaptivity)** Let  $\mathcal{D}$  and  $\mathcal{D}'$  denote two datasets, and  $d(\cdot, \cdot)$  is a metric that measures the distance between two datasets. The  $d$ -based data adaptivity of a subset  $X \subseteq \mathcal{D}$  with respect to  $\mathcal{D}'$  is defined as

$$\Delta_d(X; \mathcal{D}, \mathcal{D}') \triangleq d(\mathcal{D} \setminus X, \mathcal{D}') - d(\mathcal{D}, \mathcal{D}'). \quad (6)$$

This definition quantifies the impact of subtracting  $X$  from  $\mathcal{D}$  on the value of  $d(\mathcal{D}, \mathcal{D}')$ . Therefore, we define the dataset distillation problem that is to find a subset  $X$  that

$$\max_{X \subseteq \mathcal{D}} \Delta_d(X; \mathcal{D}, \mathcal{D}') \text{ s.t. } |X| = n', \quad (7)$$

where  $n'$  is the desired size of the distilled dataset. In general, solving this problem is very difficult because of the large searching space. In practice, we consider an approximated approach such that the importance of each sample  $\mathbf{x} \in \mathcal{D}$  is measured by  $\Delta_d(\mathbf{x}; \mathcal{D}, \mathcal{D}')$ <sup>\*</sup> and we remove the samples with less relative importance.

Now we introduce an efficient method to sort data adaptivity in practice when  $d$  is specified as MMD. Suppose  $\mathbf{x}^p \in \mathcal{D}_p$  and  $\mathbf{x}^t \in \mathcal{D}_t$ , according to [23], an empirical estimation of  $d_k^2(\mathcal{P}, \mathcal{T})$  is given by

$$\begin{aligned} \hat{d}_k^2(\mathcal{D}_p, \mathcal{D}_t) \triangleq & \frac{1}{m^2} \sum_{i,j} k(\mathbf{x}_i^p, \mathbf{x}_j^p) - \frac{2}{mn} \sum_{i,j} k(\mathbf{x}_i^p, \mathbf{x}_j^t) \\ & + \frac{1}{n^2} \sum_{i,j} k(\mathbf{x}_i^t, \mathbf{x}_j^t) \end{aligned} \quad (8)$$

<sup>\*</sup>With an abuse of notation,  $\Delta_d(\mathbf{x}; \mathcal{D}, \mathcal{D}')$  is indeed  $\Delta_d(\{\mathbf{x}\}; \mathcal{D}, \mathcal{D}')$

where  $k$  is the combination of multiple Gaussian Radial Basis Function (RBF) kernels, while  $m$  and  $n$  are the size of  $\mathcal{D}_p$  and  $\mathcal{D}_t$ . By substituting it to Equation (6), the MMD-based data adaptivity  $\Delta_{\hat{d}_k}(\mathbf{x}^p; \mathcal{D}_p, \mathcal{D}_t)$  is properly defined and can be explicitly computed for given  $\mathcal{D}_p$  and  $\mathcal{D}_t$ . However, naively computing and sorting  $\Delta_{\hat{d}_k}(\mathbf{x}^p; \mathcal{D}_p, \mathcal{D}_t)$  for all  $\mathbf{x}^p$  is costly, thus we propose in Lemma 2 a proxy that can be calculated efficiently and achieve the same sorting result for  $\Delta_{\hat{d}_k}$  under a mild condition.

**LEMMA 2** Let  $\mathcal{D} = \{\mathbf{x}_i\}_{i=1}^m$  and  $\mathcal{D}' = \{\mathbf{x}'_j\}_{j=1}^n$  denote two distinct datasets, and  $\Delta_{\hat{d}_k}(\mathbf{x}; \mathcal{D}, \mathcal{D}')$  denotes the MMD-based data adaptivity of one data  $\mathbf{x} \in \mathcal{D}$ . With the assumption that  $|\Delta_{\hat{d}_k}(\mathbf{x}; \mathcal{D}, \mathcal{D}')| \ll \hat{d}_k(\mathcal{D}, \mathcal{D}')$ , sorting the data adaptivity  $\Delta_{\hat{d}_k}(\mathbf{x}; \mathcal{D}, \mathcal{D}')$  can be achieved by sorting the following proxy quantity

$$\frac{1}{n} \sum_{\mathbf{x}_j \in \mathcal{D}'} k(\mathbf{x}, \mathbf{x}'_j) - \frac{1}{m-1} \sum_{\mathbf{x}_i \in \mathcal{D}} k(\mathbf{x}, \mathbf{x}_i). \quad (9)$$

The idea of this lemma is intuitive: data adaptivity of  $\mathbf{x}^p$  depends on how close it is to  $\mathcal{D}_t$  and how far it is from  $\mathcal{D}_p$  indicated by  $k$ . By calculating such proxy, the computation complexity for sorting data adaptivity is reduced dramatically from  $O(2m(m+n)^2)$  to  $O(m(m+n))$ .

---

#### Algorithm 2: Data Distillation

---

**Input:** model  $f$ , learning task  $\mathbb{T}$ , learning task loss  $\mathcal{L}_{\mathbb{T}}$ , ratio  $r$   
threshold  $size_t$ , epoch  $L$ , generated dataset  $\mathcal{D}_p$ , target dataset  $\mathcal{D}_t$   
**Output:** trained model  $f_L$   
1:  $size \leftarrow |\mathcal{D}_p|$ , Initialize  $f_0$   
2: **for**  $i = 1, \dots, L$  **do**  
3:  $f_i \leftarrow$  train  $f_{i-1}$  on  $\mathcal{D}_p$  by task  $\mathbb{T}$  with loss  $\mathcal{L}_{\mathbb{T}}$   
4: **if**  $size > size_t$  **then**  
5:  $size \leftarrow \max(\lfloor r \cdot size \rfloor, size_t)$   
6: Sort  $\mathcal{D}_p$  by  $\Delta_{\hat{d}_k}(\mathbf{x}^p; \mathcal{D}_p, \mathcal{D}_t)$  in descending order  
7:  $\mathcal{D}_p \leftarrow \mathcal{D}_p[1, \dots, size]$   
8: **end if**  
9: **end for**  
10: **return**  $f_L$

---

Putting everything together, we propose a workflow of progressively pruning the data of lower data adaptivity from  $\mathcal{D}_p$  as illustrated in Algorithm 2, where  $\mathbb{T}$  is our learning task on Primitive3D,  $size_t$  is a threshold deciding the size of the distilled dataset and  $r$  is the retention ratio in each distillation step. The parameter setup for  $r$  and  $size_t$  is specified in Section 5.3. Similar to previous work [53], we calculate the kernel function using the global features learned from self-reconstruction rather than the original  $\mathbf{x}$ . With the proposed method, the learning data from Primitive3D can be substantially less than the original, markedly cutting off the computational overhead without losing the efficacy of features for the target dataset as experiments show.

| Pretraining dataset | ModelNet40  |             |             | ScanObjectNN |             |             | ScanNet10   |             |             |
|---------------------|-------------|-------------|-------------|--------------|-------------|-------------|-------------|-------------|-------------|
|                     | Uns         | Sup         | Uns & Sup   | Uns          | Sup         | Uns&Sup     | Uns         | Sup         | Uns&Sup     |
| ModelNet40          | -           | -           | -           | 68.9         | 70.1        | 72.5        | 71.9        | 64.4        | 70.0        |
| ScanObjectNN        | 85.3        | 85.2        | 84.6        | -            | -           | -           | 67.4        | 62.0        | 68.0        |
| ScanNet10           | 86.3        | 85.3        | 85.6        | 70.7         | 68.5        | 70.2        | -           | -           | -           |
| ShapeNet            | <u>87.1</u> | 87.3        | 88.0        | <u>72.2</u>  | 73.2        | 74.5        | <u>72.3</u> | 66.3        | 69.3        |
| Primitive3D (Ours)  | 86.1        | <u>88.9</u> | <b>89.4</b> | 70.8         | <u>76.7</u> | <b>78.5</b> | 70.0        | <u>72.0</u> | <b>72.9</b> |

Table 2. The comparison of cross-dataset classification accuracy (%) on various 3D datasets. “Sup” and “Uns” denotes the pretraining with only supervised task and unsupervised task, respectively, while “Uns& Sup” denotes the pretraining with the combination of them.

## 5. Experiments

### 5.1. Experiments Setup

**Dataset Preparation.** We first generate a set of 150,000 point clouds extracted from Primitive3D objects sampled by the RCT approach proposed in Section 3.1. The point cloud dataset is named Primitive3D in this section, which is generated once and fixed throughout the experiments except for the ablation study on dataset size in Section 5.3. To achieve a good trade-off between object complexity and generation time, the number of leaves in RCTs is chosen in the range of 1 to 6 for our generation as detailed in Section 5.3. In our implementation, the boolean operations set  $\mathbb{O}$  contains only the regularized union, which is sufficient to generate complex objects and is easy to implement. The primitive types  $\mathbb{P}$  of RCTs include five shapes, namely, sphere, box, cylinder, cone, and torus.

| Dataset           | Initialism | Type | #Class | Split         |
|-------------------|------------|------|--------|---------------|
| ModelNet40 [63]   | MN40       | CAD  | 40     | 9,840 / 2,468 |
| ScanObjectNN [55] | SONN       | Real | 15     | 2,309 / 581   |
| ScanNet10 [45]    | SN10       | Real | 10     | 6,110 / 1,769 |
| ShapeNet [6]      | SN         | CAD  | 55     | 52,472 / -    |

Table 3. 3D object datasets for classification task.

**Network Architecture.** Unless otherwise specified, we apply the widely used DGCNN [59] as the encoder-decoder network in the experiment. The unsupervised reconstruction task employs a decoder adapted from [67] which takes the 1024-dimensional encoded feature as input, while the two supervised segmentation tasks employ a shared point-wise decoder with the same architecture as the official implementation of point cloud semantic segmentation. The number of input points is 1024 except for tasks involving object part segmentation, which uses the number of 2048.

**Preraining Setting.** We employ the multi-task learning described in Section 3.1 to pretrain deep encoders on Primitive3D to obtain feature representations. The training optimizer is set to Adam without weight decay. We train the model for 50 epochs, beginning with a learning rate of 0.001, and decaying it by 0.7 every 10 epochs. For all training processes in the following content, the batch size is set

to 32. In our multi-task learning, the parameters of discriminative loss for the instance segmentation are identical to the original configuration [13], while the weights  $\alpha = 0.05$  and  $\beta = 0.2$ , respectively. For dataset distillation, we set  $r = 0.7$  and  $size_t = 10000$  except for the ablation study.

### 5.2. Main Results

We first pretrain the models by the multi-task learning on Primitive3D. The output feature representations are then evaluated on multiple object classification benchmarks and compared to the models pretrained on other datasets. We consider four commonly used 3D object datasets for our experiments, namely, ModelNet40 [63], ScanObjectNN [55], ScanNet10 [45] and ShapeNet [6], with the statistics summarized in Table 3. The first three datasets are utilized for both benchmarking and pretraining, while the ShapeNet is included only to serve the pretraining purpose. It is worth mentioning that, compared to the synthesis dataset ModelNet40, the classifications on ScanObjectNN and ScanNet are more challenging due to the presence of real-world noise such as the occlusion of objects. In our comparison, we use the same unsupervised pretraining task on the compared datasets, while different supervised pretraining tasks based on their own annotations. Finally, identical settings are applied to all pretraining tasks.

**Cross Dataset Evaluation.** We conduct cross data evaluation on the benchmark datasets. Specifically, we pretrain the feature encoders using supervised and/or unsupervised learning methods on the pretraining dataset. We test the efficacy of the output features by using a linear SVM trained on the benchmark datasets to perform classification. The results in Table 2 show that the supervised pretraining itself on Primitive3D is enough to outperform other commonly used datasets, except that the performance on ScanNet10 is slightly behind. Moreover, adding in unsupervised pretraining guarantees the superiority of Primitive3D against all compared datasets. We also note that the unsupervised and supervised pretraining on Primitive3D always enhance each other, while on the other datasets their combination sometimes hurts the performance. This suggests that our part-based annotations might be more appropriate to guide the geometry learning of 3D objects.

| Model Pretraining          | Pretraining time | # Training Samples of ModelNet40 |             |             |             | # Training Samples of ScanNet10 |             |             |             |
|----------------------------|------------------|----------------------------------|-------------|-------------|-------------|---------------------------------|-------------|-------------|-------------|
|                            |                  | 400                              | 800         | 1200        | All         | 400                             | 800         | 1200        | All         |
| Random Init                | -                | 73.2                             | 79.4        | 84.3        | 92.0        | 66.4                            | 69.9        | 71.8        | 77.7        |
| Primitive3D (Random Drop)  | 12%              | 76.2                             | 80.0        | 83.2        | 91.4        | 67.3                            | 67.8        | 69.9        | 76.9        |
| Primitive3D (Distillation) | 14%              | <b>78.0</b>                      | <b>82.1</b> | 84.7        | 92.0        | <b>68.9</b>                     | 70.9        | 72.6        | 77.9        |
| Primitive3D (Full)         | 100%             | 77.1                             | 82.0        | <b>85.7</b> | <b>92.1</b> | 68.4                            | <b>71.8</b> | <b>73.1</b> | <b>78.1</b> |

Table 4. Classification accuracy (%) on ModelNet40 and ScanNet10 test sets with various training sample numbers. The comparisons are based on different model initialization: **Random Init**: randomly initialized; **Random Drop**: pretrained on randomly dropped Primitive3D which is indeed Algorithm 2 with random shuffling in step 6; **Distillation**: pretrained with the proposed dataset distillation on Primitive3D; **Full**: pretrained on full Primitive3D. The pretraining times are reported as the relative ratio of full Primitive3D pretraining time.

**Fine-tuning with Varying Data Size.** To check whether our method could boost the downstream task performance, we use the pretrained weights by our learning method to initialize the deep classifier and fine-tune it on benchmark datasets. Besides using the full benchmark datasets, the fine-tuning is also conducted on partial training data to evaluate the data efficiency of models. We imply the same training settings as the official implementation [59] for all runs, except that a lower initial learning rate of 0.01 is applied during fine-tuning procedure. It can be observed from Table 4 that our method surpasses the train-from-scratch method for either synthesis or real-world datasets, especially when the training samples are limited. Furthermore, we report the result of dataset distillation. For the dataset distillation method, the time reported is for the ModelNet40 dataset of 1200 samples, which is the maximum among all six pretraining times. Compared to the full Primitive3D pretraining, our distillation method saves 86% of the time, with at most 1% performance degradation for all tasks. Additionally, dataset distillation substantially outperform the train-from-scratch and the random drop method, especially when the data scale is less than 800 it also exceeds full Primitive3D pretraining.

### 5.3. More Studies

**Visualization of Pretraining Result.** We provide the visualizations in Figure 3 to show the benefits of Primitive3D annotations on the object understanding. As it shows, the primitive segmentation task may aid deep learning models in decomposing objects. Thus, it can serve as a low-level task, reducing the effort required for downstream semantic learning. However, certain unstructured and noisy objects in the real-world dataset, such as *lamp* in the last column of Figure 3, might cause the pretrained model to fail.

**Study of Dataset Generation.** We consider how the RCT structure and size of the Primitive3D pretraining dataset influence the benchmark classification result. The experiment setting is the same as the cross dataset evaluation in Section 5.2 but with different Primitive3D datasets.

In Figure 4 (left), we depict how the accuracy changes as the number of RCT leaves  $l$  varies. It indicates that the

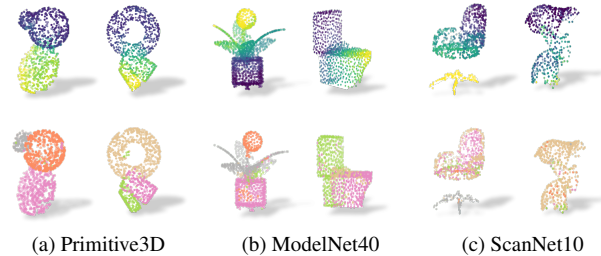


Figure 3. Instance segmentation (up) and semantic segmentation (down) results of various datasets by the pretrained model on Primitive3D. Specifically, instance embedding has been transformed into a 1-d vector by Principal Component Analysis (PCA).

accuracy gain of increasing  $l$  would get saturated once  $l$  exceeds 6. On the other hand, the time to generate an RCT sample rises considerably as  $l$  increases due to the growing complexity in the execution of the boolean operations. To strike a balance between performance and time, we limit  $l$  of RCTs in our implementation to be within [1, 6]. More time profiles of data generation can be found in Appendix.

In Figure 4 (right), we generate Primitive3D datasets with sizes ranging from 2,000 to 300,000 and same RCT settings, and perform the benchmark classification. The result suggests that increasing the size of the dataset, whether dataset distillation is applied or not, improves the accuracy; however, the performance gain of increasing size is minor once it goes beyond 150,000. For this reason, we set the dataset size to be 150,000 in our experiments.

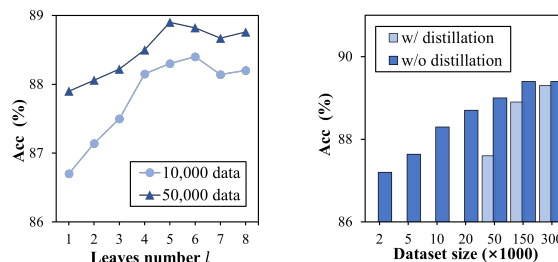


Figure 4. Effect of leaves number (left) and effect of dataset size (right) of Primitive3D on ModelNet40 classification accuracy.

**Study of Dataset Distillation.** We perform ablation studies on how to choose the parameters for dataset distil-

lation, *i.e.*, the retention ratio  $r$  and the dataset size threshold  $size_t$ . The experiment setting is the same as the cross dataset evaluation in Section 5.2 but with the incorporation of dataset distillation. In Figure 5, we see that both the pretraining time and classification accuracy grow as the increases of  $r$  and  $size_t$ , while the growth rate against  $r$  is more subtle than the rate of  $size_t$ . Notably, when  $size_t$  is set to 1,000, the accuracy drops to 87% due to the excessive dropping of pretraining data, yet the consumption time remains more than 0.13. It is also seen that a  $size_t$  of 50,000 can reach a performance comparable to full Primitive3D. In our implementation, we let  $r = 0.7$  and  $size_t = 10,000$  to achieve a compromise between downstream task performance and time efficiency.

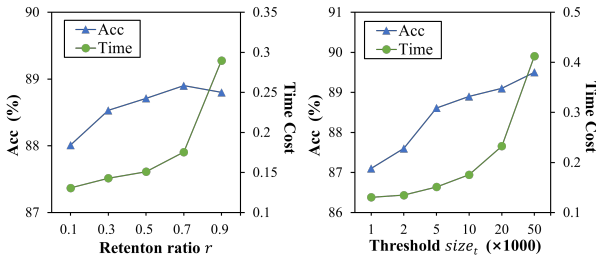


Figure 5. Effect of retention rate (left) and threshold (right) in dataset distillation on the results of ModelNet40 classification.

To visualize the dataset distillation, Figure 6 displays some retained and removed samples in the first dataset distillation stage. Particularly, the third column includes the nearest neighbors of the target data in the feature space from the removed samples. As can be observed, data distillation removes samples from the Primitive3D set that are not similar to the target data, so as to reduce the cost of learning irrelevant data.

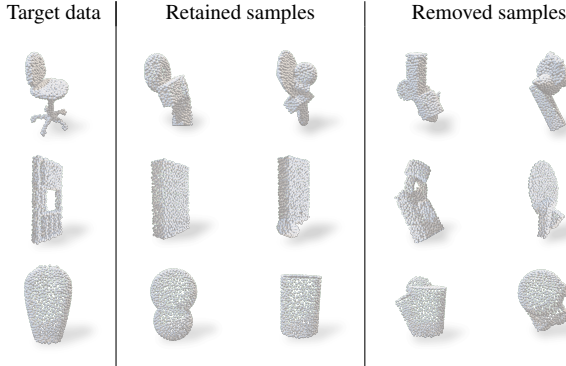


Figure 6. Visualization of dataset distillation.

**Study of Pretraining Tasks.** In Table 5, we study the effect of different pretraining tasks on the quality of the output features. Particularly, we consider four tasks, namely, unsupervised reconstruction with the original Chamfer distance, unsupervised reconstruction with the augmented Chamfer distance, supervised semantic segmentation and supervised instance segmentation.

| $U^{CD}$ | $U^{ACD}$ | $S^t$ | $S^i$ | Accuracy (%) |
|----------|-----------|-------|-------|--------------|
| ✓        |           |       |       | 85.6         |
|          | ✓         |       |       | 86.1         |
|          |           |       | ✓     | 87.1         |
|          |           | ✓     |       | 88.5         |
|          |           | ✓     | ✓     | 88.9         |
|          | ✓         | ✓     | ✓     | <b>89.4</b>  |

Table 5. Accuracy of ModelNet40 classification.  $U^{CD}$  and  $U^{ACD}$  represent the implementation of  $\mathcal{L}_{inst}$  with the Chamfer distance and the augmented Chamfer distance, respectively.  $S^t$  and  $S^i$  stand for applying the tasks of  $\mathcal{L}_{type}$  and  $\mathcal{L}_{inst}$ , respectively.

We also test their combinations to see the compositional effect. The results suggest that semantic segmentation is the most effective individual task among all, while combined tasks can further enhance learning performance.

**Other Downstream Tasks & Comparisons.** Finally, we investigate more downstream tasks and comparisons to other pretraining methods. We perform object part segmentation and unaligned object classification tasks. The former is a fine-grained shape recognition task based on ShapNet-Part [69] dataset, on which our pretraining technique enables DGCNN to improve the mean IoU from 85.0% to 85.3%. Unaligned object classification is performed on the unaligned ModelNet40 dataset, *i.e.*, randomly rotated objects. The feature representations obtained by our method outperform the supervised learned features by a large margin. Moreover, by fixing the pretraining dataset, we compare our technique with other SOTA pretraining methods. Experiments show that the highest performance is obtained when our strategy is used with Primitive3D. Note that Appendix C details the aforementioned experiments.

## 6. Conclusion

We propose an efficient approach to generate 3D objects with part-based annotations based on a randomized construction process. The efficiency allow us to synthesize a large-scale and densely-annotated 3D object dataset. To take advantage of it, we introduce a learning process that comprises multiple tasks and, optionally, a dataset distillation strategy. Combined with our generated dataset, the suggested learning produces well-generalized representations of 3D objects. The result of the experiments indicates that our dataset allows for better pretraining of models than other commonly used datasets. We expect our attempt to provide a new data source for training 3D deep models.

## Acknowledgment

This research is supported by the NUS Research Scholarship. The authors also sincerely thank Yongxing Dai and Chenxi Ma for their contributions to the paper.



## 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. 3
- [2] Panos Achlioptas, Olga Diamanti, Ioannis Mitliagkas, and Leonidas Guibas. Learning representations and generative models for 3d point clouds. In *International conference on machine learning*, pages 40–49. PMLR, 2018. 1, 2
- [3] Laurent Alonso, Jean-Luc Remy, and René Schott. A linear-time algorithm for the generation of trees. *Algorithmica*, 17(2):162–182, 1997. 3
- [4] Shai Ben-David, John Blitzer, Koby Crammer, Fernando Pereira, et al. Analysis of representations for domain adaptation. *Advances in neural information processing systems*, 19:137, 2007. 3, 5
- [5] Alexandre Boulch. Convpoint: Continuous convolutions for point cloud processing. *Computers & Graphics*, 88:24–34, 2020. 2
- [6] Angel X. Chang, Thomas Funkhouser, Leonidas Guibas, Pat Hanrahan, Qixing Huang, Zimo Li, Silvio Savarese, Manolis Savva, Shuran Song, Hao Su, Jianxiong Xiao, Li Yi, and Fisher Yu. ShapeNet: An Information-Rich 3D Model Repository. Technical Report arXiv:1512.03012 [cs.GR], Stanford University — Princeton University — Toyota Technological Institute at Chicago, 2015. 1, 2, 6
- [7] Jingdao Chen, Yihai Fang, and Yong K Cho. Performance evaluation of 3d descriptors for object recognition in construction applications. *Automation in Construction*, 86:44–52, 2018. 1
- [8] Siheng Chen, Chaojing Duan, Yaoqing Yang, Duanshun Li, Chen Feng, and Dong Tian. Deep unsupervised learning of 3d point clouds via graph topology inference and filtering. *IEEE Transactions on Image Processing*, 29:3183–3198, 2019. 4
- [9] Xiaozhi Chen, Kaustav Kundu, Yukun Zhu, Andrew G Berneshawi, Huimin Ma, Sanja Fidler, and Raquel Urtasun. 3d object proposals for accurate object class detection. In *Advances in Neural Information Processing Systems*, pages 424–432. Citeseer, 2015. 1
- [10] Angela Dai, Angel X. Chang, Manolis Savva, Maciej Halber, Thomas Funkhouser, and Matthias Nießner. Scannet: Richly-annotated 3d reconstructions of indoor scenes. In *Proc. Computer Vision and Pattern Recognition (CVPR), IEEE*, 2017. 1
- [11] Yongxing Dai, Jun Liu, Yan Bai, Zekun Tong, and Ling-Yu Duan. Dual-refinement: Joint label and feature refinement for unsupervised domain adaptive person re-identification. *IEEE Transactions on Image Processing*, 30:7815–7829, 2021. 3
- [12] Yongxing Dai, Jun Liu, Yifan Sun, Zekun Tong, Chi Zhang, and Ling-Yu Duan. Idm: An intermediate domain module for domain adaptive person re-id. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 11864–11874, 2021. 3
- [13] Bert De Brabandere, Davy Neven, and Luc Van Gool. Semantic instance segmentation with a discriminative loss function. *arXiv preprint arXiv:1708.02551*, 2017. 6
- [14] Mark De Deuge, Alastair Quadros, Calvin Hung, and Bertrand Douillard. Unsupervised feature learning for classification of outdoor 3d scans. In *Australasian Conference on Robotics and Automation*, volume 2, page 1, 2013. 1, 2
- [15] Boyang Deng, Kyle Genova, Soroosh Yazdani, Sofien Bouaziz, Geoffrey Hinton, and Andrea Tagliasacchi. Cvxnet: Learnable convex decomposition. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 31–44, 2020. 3
- [16] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pages 248–255. Ieee, 2009. 1
- [17] Henghui Ding, Xudong Jiang, Ai Qun Liu, Nadia Magnenat Thalmann, and Gang Wang. Boundary-aware feature propagation for scene segmentation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 6819–6829, 2019. 1
- [18] Henghui Ding, Xudong Jiang, Bing Shuai, Ai Qun Liu, and Gang Wang. Context contrasted feature and gated multi-scale aggregation for scene segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2393–2402, 2018. 1
- [19] Henghui Ding, Xudong Jiang, Bing Shuai, Ai Qun Liu, and Gang Wang. Semantic correlation promoted shape-variant context for segmentation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8885–8894, 2019. 1
- [20] Matheus Gadelha, Rui Wang, and Subhransu Maji. Multiresolution tree networks for 3d point cloud processing. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 103–118, 2018. 2
- [21] Xiang Gao, Wei Hu, and Guo-Jun Qi. Graphter: Unsupervised learning of graph transformation equivariant representations via auto-encoding node-wise transformations. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 7163–7172, 2020. 2
- [22] Iryna Gordon and David G Lowe. What and where: 3d object recognition with accurate pose. In *Toward category-level object recognition*, pages 67–82. Springer, 2006. 1
- [23] Arthur Gretton, Karsten M Borgwardt, Malte J Rasch, Bernhard Schölkopf, and Alexander Smola. A kernel two-sample test. *The Journal of Machine Learning Research*, 13(1):723–773, 2012. 5
- [24] Zhizhong Han, Xiyang Wang, Yu-Shen Liu, and Matthias Zwicker. Multi-angle point cloud-vae: Unsupervised feature learning for 3d point clouds from multiple angles by joint self-reconstruction and half-to-half prediction. In *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 10441–10450. IEEE, 2019. 2
- [25] Rana Hanocka, Amir Hertz, Noa Fish, Raja Giryes, Shachar Fleishman, and Daniel Cohen-Or. Meshcnn: a network with an edge. *ACM Transactions on Graphics (TOG)*, 38(4):1–12, 2019. 2
- [26] Kaveh Hassani and Mike Haley. Unsupervised multi-task feature learning on point clouds. In *Proceedings of the*

- IEEE/CVF International Conference on Computer Vision*, pages 8160–8171, 2019. 2
- [27] Heisuke Hironaka. Triangulations of algebraic sets. In *Algebraic geometry (Proc. Sympos. Pure Math., Vol. 29, Humboldt State Univ., Arcata, Calif., 1974)*, volume 29, pages 165–185, 1975. 3
- [28] Binh-Son Hua, Quang-Hieu Pham, Duc Thanh Nguyen, Minh-Khoi Tran, Lap-Fai Yu, and Sai-Kit Yeung. Scenenn: A scene meshes dataset with annotations. In *2016 Fourth International Conference on 3D Vision (3DV)*, pages 92–101. IEEE, 2016. 1
- [29] Salman H Khan, Yulan Guo, Munawar Hayat, and Nick Barnes. Unsupervised primitive discovery for improved 3d generative modeling. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 9739–9748, 2019. 1, 3
- [30] Sangpil Kim, Hyung-gun Chi, Xiao Hu, Qixing Huang, and Karthik Ramani. A large-scale annotated mechanical components benchmark for classification and retrieval tasks with deep neural networks. In *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part XVIII 16*, pages 175–191. Springer, 2020. 2
- [31] Sebastian Koch, Albert Matveev, Zhongshi Jiang, Francis Williams, Alexey Artemov, Evgeny Burnaev, Marc Alexa, Denis Zorin, and Daniele Panozzo. Abc: A big cad model dataset for geometric deep learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 9601–9611, 2019. 1, 2
- [32] Lingxiao Li, Minhyuk Sung, Anastasia Dubrovina, Li Yi, and Leonidas J Guibas. Supervised fitting of geometric primitives to 3d point clouds. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 2652–2660, 2019. 3
- [33] Xinke Li, Chongshou Li, Zekun Tong, Andrew Lim, Junsong Yuan, Yuwei Wu, Jing Tang, and Raymond Huang. Campus3d: A photogrammetry point cloud benchmark for hierarchical understanding of outdoor scene. In *Proceedings of the 28th ACM International Conference on Multimedia*, pages 238–246, 2020. 1
- [34] Yangyan Li, Rui Bu, Mingchao Sun, Wei Wu, Xinhan Di, and Baoquan Chen. Pointcnn: Convolution on x-transformed points. *Advances in neural information processing systems*, 31:820–830, 2018. 2
- [35] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. Microsoft coco: Common objects in context. In *European conference on computer vision*, pages 740–755. Springer, 2014. 1
- [36] 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. 2
- [37] Mingsheng Long, Yue Cao, Jianmin Wang, and Michael Jordan. Learning transferable features with deep adaptation networks. In *International conference on machine learning*, pages 97–105. PMLR, 2015. 3, 5
- [38] Daniel Maturana and Sebastian Scherer. Voxnet: A 3d convolutional neural network for real-time object recognition. In *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 922–928. IEEE, 2015. 2
- [39] Kaichun Mo, Shilin Zhu, Angel X Chang, Li Yi, Subarna Tripathi, Leonidas J Guibas, and Hao Su. Partnet: A large-scale benchmark for fine-grained and hierarchical part-level 3d object understanding. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 909–918, 2019. 2
- [40] Charlie Nash and Christopher KI Williams. The shape variational autoencoder: A deep generative model of part-segmented 3d objects. In *Computer Graphics Forum*, volume 36, pages 1–12. Wiley Online Library, 2017. 1, 2
- [41] Despoina Paschalidou, Ali Osman Ulusoy, and Andreas Geiger. Superquadrics revisited: Learning 3d shape parsing beyond cuboids. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10344–10353, 2019. 3
- [42] Quang-Hieu Pham, Thanh Nguyen, Binh-Son Hua, Gemma Roig, and Sai-Kit Yeung. Jsis3d: Joint semantic-instance segmentation of 3d point clouds with multi-task pointwise networks and multi-value conditional random fields. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8827–8836, 2019. 4
- [43] Charles R Qi, Hao Su, Kaichun Mo, and Leonidas J Guibas. Pointnet: Deep learning on point sets for 3d classification and segmentation. In *CVPR*, pages 652–660, 2017. 2
- [44] 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. 2
- [45] 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. *Advances in Neural Information Processing Systems*, 32:7192–7203, 2019. 3, 6
- [46] Yongming Rao, Jiwen Lu, and Jie Zhou. Global-local bidirectional reasoning for unsupervised representation learning of 3d point clouds. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5376–5385, 2020. 2
- [47] Aristides Requicha. Mathematical models of rigid solid objects. 1977. 3
- [48] Aristides AG Requicha and Herbert B Voelcker. Constructive solid geometry. 1977. 2, 3
- [49] Jonathan Sauder and Bjarne Sievers. Self-supervised deep learning on point clouds by reconstructing space. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d’Alché-Buc, E. Fox, and R. Garnett, editors, *NeurIPS*, volume 32, 2019. 2
- [50] Gopal Sharma, Rishabh Goyal, Difan Liu, Evangelos Kalogerakis, and Subhransu Maji. Csgnet: Neural shape parser for constructive solid geometry. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5515–5523, 2018. 3
- [51] Arjun Singh, James Sha, Karthik S Narayan, Tudor Achim, and Pieter Abbeel. Bigbird: A large-scale 3d database of object instances. In *2014 IEEE international conference on robotics and automation (ICRA)*, pages 509–516. IEEE, 2014. 1

- [52] Hang Su, Subhransu Maji, Evangelos Kalogerakis, and Erik Learned-Miller. Multi-view convolutional neural networks for 3d shape recognition. In *Proceedings of the IEEE international conference on computer vision*, pages 945–953, 2015. [2](#)
- [53] Ben Tan, Yu Zhang, Sinno Pan, and Qiang Yang. Distant domain transfer learning. In *Proceedings of the AAAI conference on artificial intelligence*, volume 31, 2017. [5](#)
- [54] Shubham Tulsiani, Hao Su, Leonidas J Guibas, Alexei A Efros, and Jitendra Malik. Learning shape abstractions by assembling volumetric primitives. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2635–2643, 2017. [3](#)
- [55] Mikaela Angelina Uy, Quang-Hieu Pham, Binh-Son Hua, Thanh Nguyen, and Sai-Kit Yeung. Revisiting point cloud classification: A new benchmark dataset and classification model on real-world data. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 1588–1597, 2019. [1](#), [2](#), [6](#)
- [56] V Vapnik. Statistical learning theory new york. NY: Wiley, 1:2, 1998. [5](#)
- [57] Hanchen Wang, Qi Liu, Xiangyu Yue, Joan Lasenby, and Matt J Kusner. Unsupervised point cloud pre-training via occlusion completion. In *CVPR*, 2021. [2](#)
- [58] Xinlong Wang, Shu Liu, Xiaoyong Shen, Chunhua Shen, and Jiaya Jia. Associatively segmenting instances and semantics in point clouds. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4096–4105, 2019. [4](#)
- [59] 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. [2](#), [6](#), [7](#)
- [60] Jiajun Wu, Chengkai Zhang, Tianfan Xue, William T Freeman, and Joshua B Tenenbaum. Learning a probabilistic latent space of object shapes via 3d generative-adversarial modeling. In *Proceedings of the 30th International Conference on Neural Information Processing Systems*, pages 82–90, 2016. [1](#), [2](#)
- [61] Rundi Wu, Yixin Zhuang, Kai Xu, Hao Zhang, and Baoquan Chen. Pq-net: A generative part seq2seq network for 3d shapes. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 829–838, 2020. [3](#)
- [62] Wenxuan Wu, Zhongang Qi, and Li Fuxin. Pointconv: Deep convolutional networks on 3d point clouds. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 9621–9630, 2019. [2](#)
- [63] Zhirong Wu, Shuran Song, Aditya Khosla, Fisher Yu, Linguang Zhang, Xiaoou Tang, and Jianxiong Xiao. 3d shapenets: A deep representation for volumetric shapes. In *CVPR*, pages 1912–1920, 2015. [1](#), [2](#), [6](#)
- [64] Zhijie Wu, Xiang Wang, Di Lin, Dani Lischinski, Daniel Cohen-Or, and Hui Huang. Sagnet: Structure-aware generative network for 3d-shape modeling. *ACM Transactions on Graphics (TOG)*, 38(4):1–14, 2019. [2](#)
- [65] Saining Xie, Jiatao Gu, Demi Guo, Charles R Qi, Leonidas Guibas, and Or Litany. Pointcontrast: Unsupervised pre-training for 3d point cloud understanding. In *European Conference on Computer Vision*, pages 574–591. Springer, 2020. [2](#)
- [66] Guandao Yang, Xun Huang, Zekun Hao, Ming-Yu Liu, Serge Belongie, and Bharath Hariharan. Pointflow: 3d point cloud generation with continuous normalizing flows. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 4541–4550, 2019. [2](#)
- [67] Yaoqing Yang, Chen Feng, Yiru Shen, and Dong Tian. Foldingnet: Point cloud auto-encoder via deep grid deformation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 206–215, 2018. [2](#), [4](#), [6](#)
- [68] Mohsen Yavartanoo, Jaeyoung Chung, Reyhaneh Neshatavar, and Kyoung Mu Lee. 3dias: 3d shape reconstruction with implicit algebraic surfaces. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 12446–12455, 2021. [3](#)
- [69] 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 Transactions on Graphics (ToG)*, 35(6):1–12, 2016. [2](#), [8](#)
- [70] Chuhan Zou, Ersin Yumer, Jimei Yang, Duygu Ceylan, and Derek Hoiem. 3d-prnn: Generating shape primitives with recurrent neural networks. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 900–909, 2017. [3](#)