Primitive3D: 3D Object Dataset Synthesis from Randomly Assembled Primitives Supplementary Material

Xinke Li[1,*] Henghui Ding[2,3,*] Zekun Tong[1] Yuwei Wu[1,†] Yeow Meng Chee[1]
[1]National University of Singapore [2]ByteDance [3]ETH Zürich
{xinke.li, zekuntong}@u.nus.edu henghui.ding@vision.ee.ethz.ch {wyw, ymchee}@nus.edu.sg

A. Theoretical Analysis

A.1. Proof of Lemma 2

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 $\left|\Delta_{\hat{d}_k}(\mathbf{x}; \mathcal{D}, \mathcal{D}')\right| \ll \hat{d}_k(\mathcal{D}, \mathcal{D}')$, sorting the data adaptivity $\Delta_{\hat{d}_k}(\mathbf{x}; \mathcal{D}, \mathcal{D}')$ can be achieved by sorting the following proxy quantity*

$$\tilde{\Delta}(\mathbf{x}) \triangleq \frac{1}{n}\sum_{\mathbf{x}_j \in \mathcal{D}'} k\left(\mathbf{x}, \mathbf{x}'_j\right) - \frac{1}{m-1}\sum_{\mathbf{x}_i \in \mathcal{D}} k\left(\mathbf{x}, \mathbf{x}_i\right). \tag{1}$$

Proof. Let $\hat{d}_k^2$ denote $\hat{d}_k^2(\mathcal{D}, \mathcal{D}')$, $\mathbf{x}_{i^*}$ denote one removed sample from $\mathcal{D}$, and $\hat{d}_{k,\bar{i}^*}^2$ denote $\hat{d}_k^2\left(\mathcal{D}\backslash\mathbf{x}_{i^*}, \mathcal{D}'\right)$. By substituting the explicit definition of $\hat{d}_k^2$, we have

$$\begin{aligned}
\hat{d}_k^2 &= \frac{1}{m^2}\sum_{i,j} k\left(\mathbf{x}_i, \mathbf{x}_j\right) - \frac{2}{mn}\sum_{i,j} k\left(\mathbf{x}_i, \mathbf{x}'_j\right) + \frac{1}{n^2}\sum_{i,j} k\left(\mathbf{x}'_i, \mathbf{x}'_j\right), \\
\hat{d}_{k,\bar{i}^*}^2 &= \frac{1}{(m-1)^2}\sum_{i\neq i^*, j\neq i^*} k\left(\mathbf{x}_i, \mathbf{x}_j\right) - \frac{2}{(m-1)n}\sum_{i\neq i^*, j} k\left(\mathbf{x}_i, \mathbf{x}'_j\right) \\
&\quad + \frac{1}{n^2}\sum_{i,j} k\left(\mathbf{x}'_i, \mathbf{x}'_j\right).
\end{aligned} \tag{2}$$

Here, we define the following notation:

$$\begin{aligned}
A &= \sum_{i,j} k\left(\mathbf{x}_i, \mathbf{x}_j\right), \quad B = \sum_{i,j} k\left(\mathbf{x}_i, \mathbf{x}'_j\right), \\
C &= \sum_{i,j} k\left(\mathbf{x}'_i, \mathbf{x}'_j\right), \quad D = k\left(\mathbf{x}_{i^*}, \mathbf{x}_{i^*}\right),
\end{aligned} \tag{3}$$

It is known that when $\mathcal{D}$ and $\mathcal{D}'$ are given, A, B, C, and D are fixed constants. Then it is easy to check that the following equalities hold.

$$\begin{aligned}
\hat{d}_k^2 &= \frac{1}{m^2}A - \frac{2}{mn}B + \frac{1}{n^2}C, \\
\hat{d}_{k,\bar{i}^*}^2 &= \frac{1}{(m-1)^2}\left(A - 2\sum_i k\left(\mathbf{x}_i, \mathbf{x}_{i^*}\right) + D\right) \\
&\quad - \frac{2}{(m-1)n}\left(B - \sum_j k\left(\mathbf{x}_{i^*}, \mathbf{x}'_j\right)\right) + \frac{1}{n^2}C.
\end{aligned} \tag{4}$$

By Equation (4), we can calculate the difference between $\hat{d}_{k,\bar{i}^*}^2$ and $\hat{d}_k^2$ as

$$\begin{aligned}
&\hat{d}_{k,\bar{i}^*}^2 - \hat{d}_k^2 \\
&= \frac{2}{m-1}\left(-\frac{1}{(m-1)}\sum_i k\left(\mathbf{x}_i, \mathbf{x}_{i^*}\right) + \frac{1}{n}\sum_j k\left(\mathbf{x}_{i^*}, \mathbf{x}'_j\right)\right) + E, \\
&= \frac{2}{m-1}\tilde{\Delta}(\mathbf{x}_{i^*}) + E
\end{aligned} \tag{5}$$

where E is a constant defined by

$$E = \left(\frac{1}{(m-1)^2} - \frac{1}{m^2}\right)A + \left(\frac{2}{(m-1)n} - \frac{2}{mn}\right)B + \frac{D}{(m-1)^2}. \tag{6}$$

Therefore, sorting difference of $\hat{d}_{k,\bar{i}^*}^2 - \hat{d}_k^2$ for all $\mathbf{x}_{i^*} \in \mathcal{D}$ can be achieved by sorting the $\tilde{\Delta}(\mathbf{x}_{i^*})$ in Equation (5). Then we check the relationship between the data adaptivity $\Delta_{\hat{d}_k}(\mathbf{x}_{i^*}; \mathcal{D}, \mathcal{D}')$ and difference $\hat{d}_{k,\bar{i}^*}^2 - \hat{d}_k^2$. From the definition of data adaptivity in the main script [26], we have the following equation

$$\Delta_{\hat{d}_k}(\mathbf{x}_{i^*}; \mathcal{D}, \mathcal{D}') = \hat{d}_{k,\bar{i}^*} - \hat{d}_k. \tag{7}$$

Let $\Delta_{\hat{d}_k,\bar{i}^*} = \Delta_{\hat{d}_k}(\mathbf{x}_{i^*}; \mathcal{D}, \mathcal{D}')$, we can obtain,

$$\hat{d}_{k,\bar{i}^*}^2 - \hat{d}_k^2 = 2\hat{d}_k\Delta_{\hat{d}_k,\bar{i}^*} + \Delta_{\hat{d}_k,\bar{i}^*}^2 \tag{8}$$

With the assumption of $|\Delta_{\hat{d}_k,\bar{i}^*}| \ll \hat{d}_k$, the second term of Equation (8) is negligible. As $\hat{d}_k$ is a positive constant, we can obtain

$$\Delta_{\hat{d}_k,\bar{i}^*} \propto \hat{d}_{k,\bar{i}^*}^2 - \hat{d}_k^2. \tag{9}$$

Thus, it is sufficient to sort the data adaptivity for all $\mathbf{x}_{i^*} \in \mathcal{D}$ by sorting the proxy quantity $\tilde{\Delta}(\mathbf{x}_{i^*})$. □

*Equal contribution
†Corresponding author

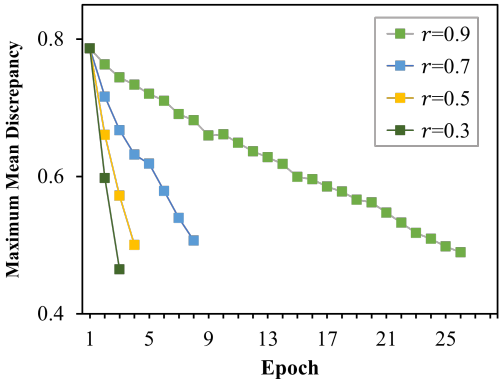

Figure 1. MMD between Primitive3D and target dataset ModelNet40, *i.e.* $\hat{d}_k(\mathcal{D}_p, \mathcal{D}_t)$ during training with dataset distillation.

Remark. Indeed, the condition $|\Delta_{\hat{d}_k,\bar{i}^*}| \ll \hat{d}_k$ means that removing $\mathbf{x}_{i^*}$ from $\mathcal{D}$ poses trivial effect on $\hat{d}_k(\mathcal{D}, \mathcal{D}')$. We note that it can be easily fulfilled when the size of dataset $\mathcal{D}$ is large. On the other hand, to verify the assumption condition exactly, one still needs to calculate $\hat{d}_k$ and $\hat{d}_{k,\bar{i}^*}$ in practice, which is against the purpose of computation reduction.

To check whether our dataset distillation method with $\tilde{\Delta}(\mathbf{x})$ sorting leads to a decrease in $\hat{d}_k$, we record the MMD between the generated dataset $\mathcal{D}_p$ and the target dataset $\mathcal{D}_t$, *i.e.* $\hat{d}_k(\mathcal{D}_p, \mathcal{D}_t)$, during the distillation process in Figure 1. It can be observed that $\hat{d}_k(\mathcal{D}_p, \mathcal{D}_t)$ is constantly reduced by removing samples with smaller proxy quantities by Lemma 2.

A.2. Analysis of RCT Samples

In this analysis, we first refine the definition of r-set and Boolean set operations in RCT. Then, we present three examples to demonstrate how any 3D object can be represented or approximated by the RCT sample, when $\mathbb{P}$ contains certain primitive types, like spheres, boxes, and tetrahedrons. Examples are shown in Figure 2.

Definition of r-set. Here we present the theoretical definition of the r-set in solid modeling. The sets we are concerned with are subsets of $\mathbb{R}^3$.

DEFINITION 1 (Regular Set [18]). *The regularization of a set X, $reg(X)$, is the closure of the interior of X,* i.e., *$reg(X) = cl(int(X))$. A set Z is regular if $reg(Z) = Z$.*

The regularized set is the standard model used in solid modeling. The primary benefit of employing it to describe solids is that "dangling" edges or surfaces are eliminated [4]. With the above definition, we refer to the r-set as a a regularized set of the form $\psi = \{(x, y, z) : g(x, y, z) \leq 0\}$ for some function $g : \mathbb{R}^3 \to \mathbb{R}$, of which the boundary $\{(x, y, z) : g(x, y, z) = 0\}$ is semi-analytic. More associated theoretical results can be found in [4].

Regularized Operation. Considering the definition of the r-set, we are able to define *regularized boolean set operations* as

$$X \cup^* Y \triangleq reg(X \cup Y),$$
$$X \cap^* Y \triangleq reg(X \cap Y),$$
$$X -^* Y \triangleq reg(X - Y),$$

where $\cup$, $\cap$, and $-$ are the set operations of union, intersection, and subtract, respectively. These definitions, when combined with the definition of a regular set, result in Lemma 1 in the main script [26], due to the fact that the class of regular sets is closed under regularized set operations [9].

Union of spheres. A well-implemented representation of 3D solids is the medial axis transform [3]. Instead of using the boundary to represent a solid, it uses the union of infinite maximal spheres completely contained in the interior. The centers of these spheres are located on the medial axis of the solid. Formally, the medial axes of a solid is *homotopy* equivalent to the 3D solid [6], which suggests the ability of the union of spheres to represent complex objects.

Union of boxes. The union of voxels, *i.e.*, cubic boxes, is a powerful and widely used representation of 3D solids [10]. Taking into account a voxel with edge length a, we can fill bounded 3D objects with finite voxels by stacking them. Additionally, we may anticipate a higher approximation accuracy for the given object when the voxel edge a is lower.

Union of tetrahedrons. It is proved that any topological 3-manifold is *homemorphic* to a union of tetrahedrons [13, 14]. Based on the above theory, a method to triangulate 3D manifold solids into multiple tetrahedrons is presented in [12]. Furthermore, any tetrahedron can be obtained from primitives by taking the intersection of four primitives with planar boundaries, *e.g.*, the box, cone and cylinder in our primitive types. For this reason, our RCT samples can generate tetrahedrons for representing 3D solids.

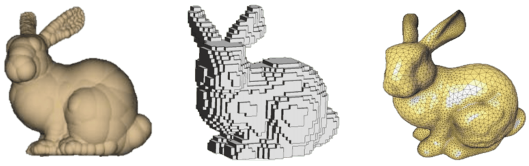

Figure 2. The approximated representation of 3D object by the union of spheres (left) [21], the union of boxes (middle) [11] and the union of tetrahedrons (right) [2].

# Primitive	1	2	3	4	5	6
# Sample	2000	6000	12000	30000	50000	50000

Table 1. Statistics of the generated Primitive3D dataset.

B. More Implementation Details

B.1. Dataset Construction

In this subsection, we describe the process of generating Primitive3D data in more depth, including the set of primitive types and the generation time.

Primitive Type Set. The sphere, box, cylinder, torus, and cone are the primitive types utilized in our RCT-based data generation, which formally can be represented as

$$\mathbb{P} = \{\Psi_{Sphere}, \Psi_{Box}, \Psi_{Cylinder}, \Psi_{Torus}, \Psi_{Cone}\},$$

where each type Ψ_i has parameters Θ^{Ψ_i}, as seen in Figure 3. In particular, because the canonical form of primitives is bounded within the unit ball and centered at the original point, the actual number of adjustable parameters for Ψ_i is $|\Theta^{\Psi_i}| - 1$. We also present the statistics of our Primitive3D dataset generated in Table 1, which summarizes the number of primitives and RCTs in the dataset.

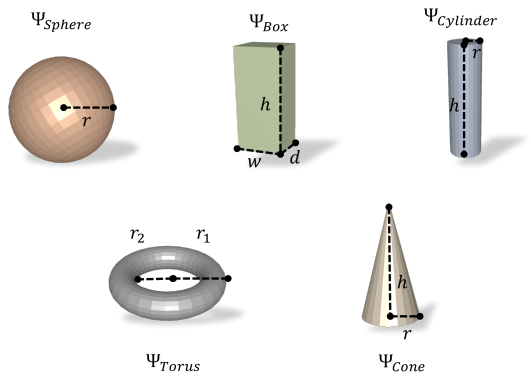

Figure 3. Primitive types set $\mathbb{P}$ and the parameters per type.

Generation Time Profile. To demonstrate the cost efficiency of our dataset collection, we present the time profile for the mesh-based generation of Primitive3D data. The results in Figure 4 demonstrate that the generation time of an RCT sample is proportional to its number of leaf nodes l, which is because the executions of $l-1$ boolean operations consume more than 90% of the generation time. When l is equal to 5, the largest leaf number that we use, the generation time is 0.075 s. To obtain more data in a short time, we further explore parallel computing settings. As seen in Figure 4, multithreading can help reduce running time, but the benefit becomes minor as the number of processors increases. Finally, when parallel computing is allowed, we can produce 10,000 RCT samples with 6 leaves in only 2 minutes. The above time profiles are based on Intel Xeon CPU E5-2680 v4 @ 2.40GHz and Ubuntu 18.04 operation system.

B.2. Segmentation Loss Function

We define two supervised segmentation tasks based on the generated labels, namely, predicting the primitive type and instance which each point belongs to. Specifically, a decoder is first employed on the point-wise feature output from the encoder. The segmentation tasks are conducted on the decoded embeddings, with the loss being the summation of the losses of two branch tasks as:

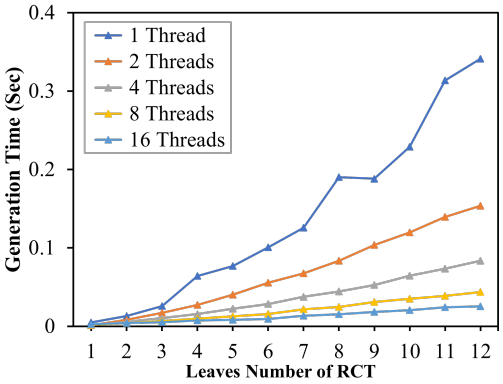

Figure 4. RCT sample generation time *vs*. leaves number l.

$$\mathcal{L}_{seg} = \mathcal{L}_{type} + \alpha \cdot \underbrace{(\mathcal{L}_{var} + \mathcal{L}_{dist} + \gamma \cdot \mathcal{L}_{reg})}_{\mathcal{L}_{inst}}. \quad (10)$$

where α, γ are the tunable weights, $\mathcal{L}_{type}$ is the cross-entropy loss of predicting $\mathbf{y}_t$ as usual, and $\mathcal{L}_{inst}$ is the clustering loss of point embedding from [15, 23, 28]. It consists of three components, in which $\mathcal{L}_{var}$ pull the embedding toward the cluster center, $\mathcal{L}_{dist}$ separates the distinct cluster centers, and $\mathcal{L}_{reg}$ is the regularization term, with their formulations as follows,

$$\mathcal{L}_{var} = \frac{1}{P}\sum_{p=1}^{P}\frac{1}{N_p}\sum_{i=1}^{N_p}[\|\mu_p - \mathbf{e}_i\|_2 - \delta_{var}]_+^2 \quad (11)$$

$$\mathcal{L}_{dist} = \frac{1}{P(P-1)}\sum_{p=1}^{P}\sum_{q\neq p}^{P}[2\delta_{dist} - \|\mu_p - \mu_q\|_2]_+^2 \quad (12)$$

$$\mathcal{L}_{reg} = \frac{1}{P}\sum_{p=1}^{P}\|\mu_p\|_2 \quad (13)$$

where P is the number of primitives in an object, N_p is the number of points in p-th primitive, $\mathbf{e}_i \in \mathcal{R}^{n_e}$ is the embedding of i-th point in $\mathbf{x}$ and μ_p is the centroid of point embeddings in p-th primitive. Also, $[x]_+ = \max(x, 0)$ is the hinge function, δ_{dist} and δ_{var} are the margins for their losses. We set $\alpha = 0.05$ and $\gamma = 0.001$ in practice.

B.3. Network Architecture

As the recent development in graph-based neural networks [20], we imply dynamic graph neural network (DGCNN) for our experiments. The DGCNN architecture used follows the original implementation [24]. The encoder-decoder network of DGCNN is illustrated in Figure 5. The global feature output from the encoder is a 1024-dimensional vector, while the point-wise feature is a 1536-dimensional vector.

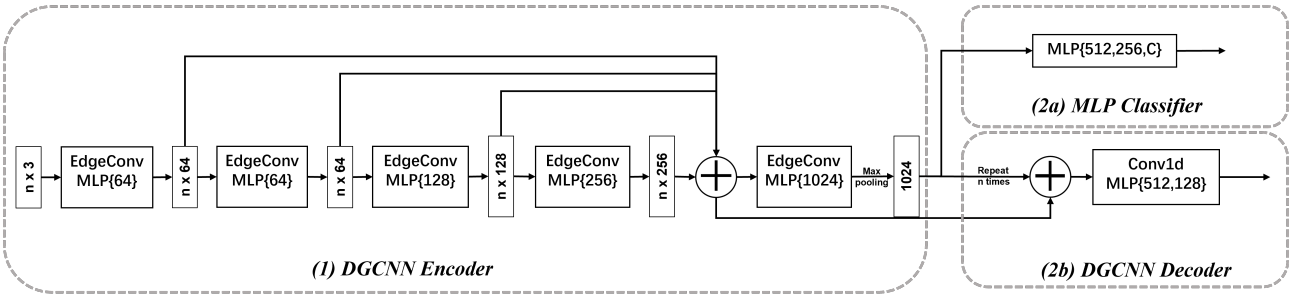

Figure 5. Our model with DGCNN.

B.4. Setup for Additional Tasks

Part Segmentation. The part segmentation is a fine-grained shape recognition task based on ShapeNetPart [27] that contains 16881 shapes with 16 categories and 50 types of parts. We use the same optimization setting as in [24]. The optimizer is chosen as Adam with the initial learning rate of 0.003 and no weight decay. The learning rate decays by 0.5 every 20 epochs until it reaches 0.00001. The total number of epochs and batch size are set to 200 and 16. We employ the pretraining & fine-tuning strategy to evaluate our method, and these settings are consistent for both the train-from-scratch and fine-tuning process. We also note that the input point number of 2048 is set for all runs including the train-from-scratch, pretraining, and fine-tuning processes.

Unaligned Object Classification. Unaligned object classification is to evaluate the sensitivity of learned features to the rotation transformation as the real-world objects are often unaligned. The used unaligned object dataset is derived from the aligned ModelNet40 dataset [25]. In particular, we independently sample rotation matrices from $SO(3)$ and employ them on aligned ModelNet40 objects. To control the magnitude of the sampled rotation angle in practice, we apply the axis-angle representation of rotation and limit the rotation angle in a range $[0, \omega_{max}]$. When the parameter ω_{max} is set to π, the rotation matrices would be uniformly sampled from $SO(3)$. To evaluate the effectiveness of pretraining methods on this task, the feature encoders are first pretrained on the pretraining dataset. The pretrained feature encoders are utilized to extract features from the training split of the unaligned object dataset. A linear SVM is trained on these extracted features and performs classification on the unaligned test split.

C. More Experimental Results

This section illustrates our method's performance on a variety of tasks. We begin by supplementing the main script with more object classification results. Following that, we present the results of object part segmentation and unaligned object classification in further detail.

Pretraining dataset	Pretraining method	Classification dataset MN40	SONN	SN10
ShapeNet [5]	JigSaw3D [19]	88.6	72.7	69.8
ShapeNet [5]	GraphTER [7]	88.7	74.7	71.0
ShapeNet [5]	OcCo [22]	89.2	78.3	71.2
Primitive3D	JigSaw3D [19]	87.4	71.8	70.1
Primitive3D	GraphTER [7]	89.3	75.3	68.5
Primitive3D	OcCo [22]	89.0	77.4	71.6
Primitive3D	MT (ours)	**89.4**	**78.5**	**72.9**

Table 2. The comparison of cross-dataset classification accuracy (%) on various 3D datasets.

C.1. Object Classification

Comparisons of Pretraining Method. Our comparison in Section 5.2 of the main script [26] focuses on the comparison of pretraining datasets, while this section compares our multi-task learning method with other pretraining approaches by fixing the dataset. In particular, the comparison includes our implementation of three self-supervised methods: the jigsaw reconstruction task [19] (JigSaw3D), the transformation equivariance task [7] (GraphTER) and the occlusion completion task [22] (OcCo). The pretraining settings of these methods follow the official implementation. The setting of experiments follows the *Cross Dataset Evaluation* in Section 5.2 of the main script [26]. For a fair comparison, we apply the same model architecture (DGCNN) and input point number (1024) for all runs.

The results in Table 2 demonstrate that our multi-task learning constantly outperforms other pretraining methods on the Primitive3D dataset. While the OcCo method can perform well when utilized on both ShapeNet and Primitive3D, there remains some gap between the best performance of OcCo and the results of multi-task pretraining on Primitive3D.

Training Curve of Fine-tuning. In this experiment, we compare the training curves for three initializations of the classifier weights: randomly initialized without pretraining; multi-task pretraining on ShapeNet; multi-task pretraining on Primitive3D. The setting of the experiment follows the fine-tuning with all training samples in Section 5.2 of the

Method	OA	AA	bench 20	bowl 20	cone 20	cup 20	curtain 20	door 20	flower_pot 20	keyboard 20
PointNet+RI	89.1	85.6	75	100	90.0	55.0	95.0	90.0	25.0	95.0
DGCNN+RI	92.0	87.9	80.0	95.0	100	45.0	95.0	85.0	45.0	100
DGCNN+Ours	**92.1**	**89.1**	80.0	95.0	100	**60.0**	90.0	**95.0**	**65.0**	100

	lamp 20	laptop 20	person 20	radio 20	sink 20	stairs 20	stool 20	tent 20	wardrobe 20	xbox 20
	85.0	100	85.0	75.0	85.0	80.0	60.0	95.0	70.0	70.0
	85.0	100	85.0	70.0	85.0	90.0	75.0	95.0	75.0	80.0
	85.0	100	**95.0**	**85.0**	**90.0**	90.0	75.0	95.0	75.0	75.0

Table 3. Overall, average and class-wise accuracy (%) on ModelNet40. Numbers under class names indicate the number of instances. We select 18 classes with the fewest samples among all 40 classes. **RI** stands for the random initialization.

Method	OA	AA	bathhub 26	bed 85	bookshelf 146	cabinet 149	chair 801	lamp 41	monitor 61	plant 25	sofa 134	table 301
PointNet+RI	76.4	62.0	50.0	54.1	**55.5**	62.4	91.9	**58.5**	63.9	52.0	54.5	77.7
DGCNN+RI	77.7	62.0	53.9	58.9	47.3	71.2	**95.0**	43.9	63.9	60.0	45.5	**80.1**
DGCNN+Ours	**78.1**	**64.3**	**57.7**	**65.9**	47.3	**73.2**	93.5	39.0	**68.9**	**64.0**	**55.2**	78.7

Table 4. Overall, average and class-wise Accuracy (%) on ScanNet10. Numbers under class names indicate the number of instances.

Method	OA	AA	bag 17	bin 40	box 28	cabinet 75	chair 78	desk 30	display 42	door 42	shelf 49	table 54	bed 22	pillow 21	sink 24	sofa 42	toilet 17
PointNet+RI	76.6	70.9	70.6	75.0	39.3	76.0	98.7	73.3	83.3	90.5	77.6	77.8	59.1	52.4	50.0	92.9	47.1
DGCNN+RI	82.1	78.4	70.6	80.0	46.4	84.0	**100**	66.7	85.7	95.2	79.6	79.6	77.3	81.0	**58.3**	**95.2**	76.5
DGCNN+Ours	**83.3**	**80.2**	70.6	**82.5**	46.4	84.0	94.9	**73.3**	**90.5**	95.2	**83.7**	**88.9**	**86.4**	81.0	54.0	88.1	**82.4**

Table 5. Overall, average, and class-wise Accuracy (%) on ScanObjectNN. Numbers under class names indicate the number of instances.

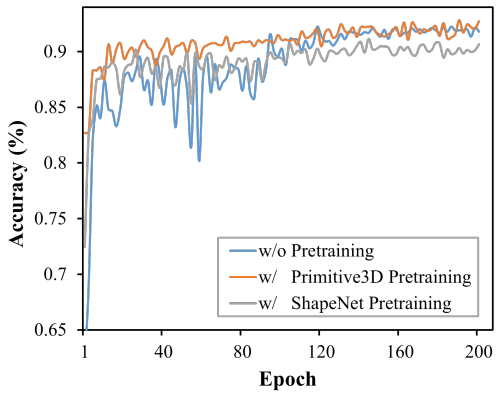

Figure 6. Test accuracy curve on ModelNet40 using random weights and pretrained weights from different datasets.

main script [26]. The training curves record the test accuracy per two epochs, which are depicted in Figure 6.

As seen in Figure 6, our pretraining strategy leads to faster convergence of the classifier's training. In particular, our method obtains 90% accuracy in 10 epochs, whereas randomly initialized weights require about 90 epochs.

Class-wise Results of Fine-tuning. In Table 3, Table 4, and Table 5, we provide the per category classification results of the full training data fine-tuning experiment. As can be seen, our fine-tuning is usually more accurate than training-from-scratch on categories with fewer samples, such as *flowerpot* in ModelNet40, *plant* in ScanNet10, and *toilet* in ScanObjectNN. This demonstrates the potential of our technique for dealing with undersampled categories in object classification. However, the categories with noisy and unstructured objects are difficult to learn with our methods, such as *lamp* in ScanNet10 and *sofa* in ScanObjectNN.

C.2. Unaligned Object Classification

We perform the unaligned object classification task to verify the robustness of our learning method to rotation transformation. We first train a feature encoder using supervised learning on the aligned ModelNet40 dataset. Using the feature encoder as a baseline, we report the accuracy gain over the baseline by various pretraining methods on Primitive3D in Table 6. Furthermore, we alter the upper bound on the rotation angle, *i.e.* ω_{max}, to control the magnitude of random alignment.

As can be observed, our strategy, whether uses multi-task learning or supervised tasks alone, learns rotation-robust feature representations that obviously outperform other learning methods. As the degree of alignment drops,

Pretraining method	ω_{max} for random rotation $\frac{\pi}{2}$	$\frac{2\pi}{3}$	$\frac{5\pi}{6}$	π
JigSaw3D [19]	3.1	3.9	4.6	5.1
OcCo [22]	2.8	3.7	6.4	8.6
GraphTER [7]	-0.8	-3.1	-1.1	-2.8
Primitive3D+Sup	5.3	**7.1**	**8.6**	9.1
Primitive3D+Uns&Sup	**5.4**	6.0	8.4	**10.9**

Table 6. Accuracy gain (%) in unaligned ModelNet40 over supervised learned features on ModelNet40. The compared features are obtained by learning Primitive3D with various methods.

namely, ω_{max} increases, the accuracy margin of our method over the baseline continues to improve. Our feature learning method, in particular, obtains an accuracy of 67.6 % on the fully unaligned dataset ($\omega_{max} = \pi$), while the supervised baseline only achieves 56.7%.

C.3. Part Segmentation

We perform pretraining on Primitive3D and fine-tuning on ShapeNetPart for the object part segmentation task. As seen in the results of Table 7, the performance of our pretraining method can beat either the train-from-scratch or the unsupervised counterparts. This illustrates that our pretraining strategy is successful not only for instance-level tasks but also for point-level feature learning, as the objective of our supervised segmentation task is to learn the local geometry information of objects.

D. Limitations of Our Work

Our work mainly focuses on constructing a larege-scale 3D object dataset with low cost. However, we restrict our discussion to only 3D object understanding, whereas other scene-level 3D tasks such as semantic scene segmentation are not covered in this study. It will be interesting in future work to construct random scenes consisting of multiple generated objects to serve a broader range of applications.

E. Visualization of Dataset

In Figure 7, we visualize some selected examples in Primitive3D with semantic annotations (primitive type label) and instance annotations (primitive instance label).

References

[1] 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. 8

[2] Pierre Alliez, David Cohen-Steiner, Mariette Yvinec, and Mathieu Desbrun. Variational tetrahedral meshing. In *ACM SIGGRAPH 2005 Papers*, pages 617–625. 2005. 2

[3] Nina Amenta, Sunghee Choi, and Ravi Krishna Kolluri. The power crust. In *Proceedings of the sixth ACM symposium on Solid modeling and applications*, pages 249–266, 2001. 2

[4] Suzanne Fox Buchele. *Three-dimensional binary space partitioning tree and constructive solid geometry tree construction from algebraic boundary representations*. The University of Texas at Austin, 1999. 2

[5] 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. 4

[6] Hyeong In Choi, Sung Woo Choi, and Hwan Pyo Moon. Mathematical theory of medial axis transform. *pacific journal of mathematics*, 181(1):57–88, 1997. 2

[7] 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. 4, 6, 8

[8] 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. 8

[9] 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. 2

[10] GJ Jense. Voxel-based methods for cad. *Computer-aided design*, 21(8):528–533, 1989. 2

[11] Nilanjana Karmakar, Arindam Biswas, Partha Bhowmick, and Bhargab B Bhattacharya. A combinatorial algorithm to construct 3d isothetic covers. *International Journal of Computer Mathematics*, 90(8):1571–1606, 2013. 2

[12] Sai Huen Lo. Volume discretization into tetrahedra—ii. 3d triangulation by advancing front approach. *Computers & Structures*, 39(5):501–511, 1991. 2

[13] Edwin E Moise. Affine structures in 3-manifolds: V. the triangulation theorem and hauptvermutung. *Annals of mathematics*, pages 96–114, 1952. 2

[14] Edwin E Moise. Geometric topology in dimensions 2 and 3. 1977. 2

[15] 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. 3

[16] 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. 8

[17] 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. 8

[18] Aristides Requicha and Robert Tilove. Mathematical foundations of constructive solid geometry: General topology of closed regular sets. 1978. 2

[19] 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. 4, 6

[20] Zekun Tong, Yuxuan Liang, Changsheng Sun, Xinke Li, David Rosenblum, and Andrew Lim. Digraph inception convolutional networks. *Advances in neural information processing systems*, 33:17907–17918, 2020. 3

[21] Dangxiao Wang, Xin Zhang, Yuru Zhang, and Jing Xiao. Configuration-based optimization for six degree-of-freedom haptic rendering for fine manipulation. *IEEE transactions on haptics*, 6(2):167–180, 2012. 2

[22] Hanchen Wang, Qi Liu, Xiangyu Yue, Joan Lasenby, and Matt J Kusner. Unsupervised point cloud pre-training via occlusion completion. In *CVPR*, 2021. 4, 6

[23] 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. 3

[24] 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. 3, 4, 8

[25] 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. 4

[26] Li Xinke, Ding Henghui, Tong Zekun, Wu Yuwei, and Yeow Meng Chee. Primitive3d: 3d object dataset synthesis from randomly assembled primitives, 2022. 1, 2, 4, 5

[27] 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. 4

[28] Hui Zhang and Henghui Ding. Prototypical matching and open set rejection for zero-shot semantic segmentation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 6974–6983, 2021. 3

Shapes	Supervised			Unsupervised			
	PointNet [16]	PointNet++ [17]	DGCNN [24]	LGAN [1]	MAP-VAE [8]	GraphTER [7]	Ours
Aero	83.4	82.4	84.2	54.1	62.7	81.7	**84.0**
Bag	78.7	79.0	83.7	48.7	67.1	68.1	**84.8**
Cap	82.5	87.7	84.4	62.6	73.0	83.7	**83.7**
Car	74.9	77.3	77.1	43.2	58.5	74.6	**77.8**
Chair	89.6	90.8	90.9	68.4	77.1	88.1	**91.1**
Earphone	73.0	71.8	**78.5**	58.3	67.3	68.9	78.3
Guitar	91.5	91.0	91.5	74.3	84.8	90.6	**91.5**
Knife	85.9	85.9	87.3	68.4	77.1	86.6	**88.2**
Lamp	80.8	**83.7**	82.9	53.4	60.9	80.0	83.4
Laptop	95.3	95.3	96.0	82.6	90.8	95.6	**95.8**
Motor	65.2	71.6	67.8	18.6	35.8	56.3	65.2
Mug	93.0	**94.1**	93.3	75.1	87.7	90.0	93.8
Pistol	81.2	81.3	82.6	54.7	64.2	80.8	**82.9**
Rocket	57.9	58.7	**59.7**	37.2	45.0	55.2	57.9
Skateboard	72.8	**76.4**	75.5	46.7	60.4	70.7	75.2
Table	80.6	82.6	81.7	66.4	74.8	79.1	**82.8**
mean	83.7	85.1	85.0	57.0	68.0	81.9	**85.3**

Table 7. Detailed results on part segmentation task on ShapeNetPart.

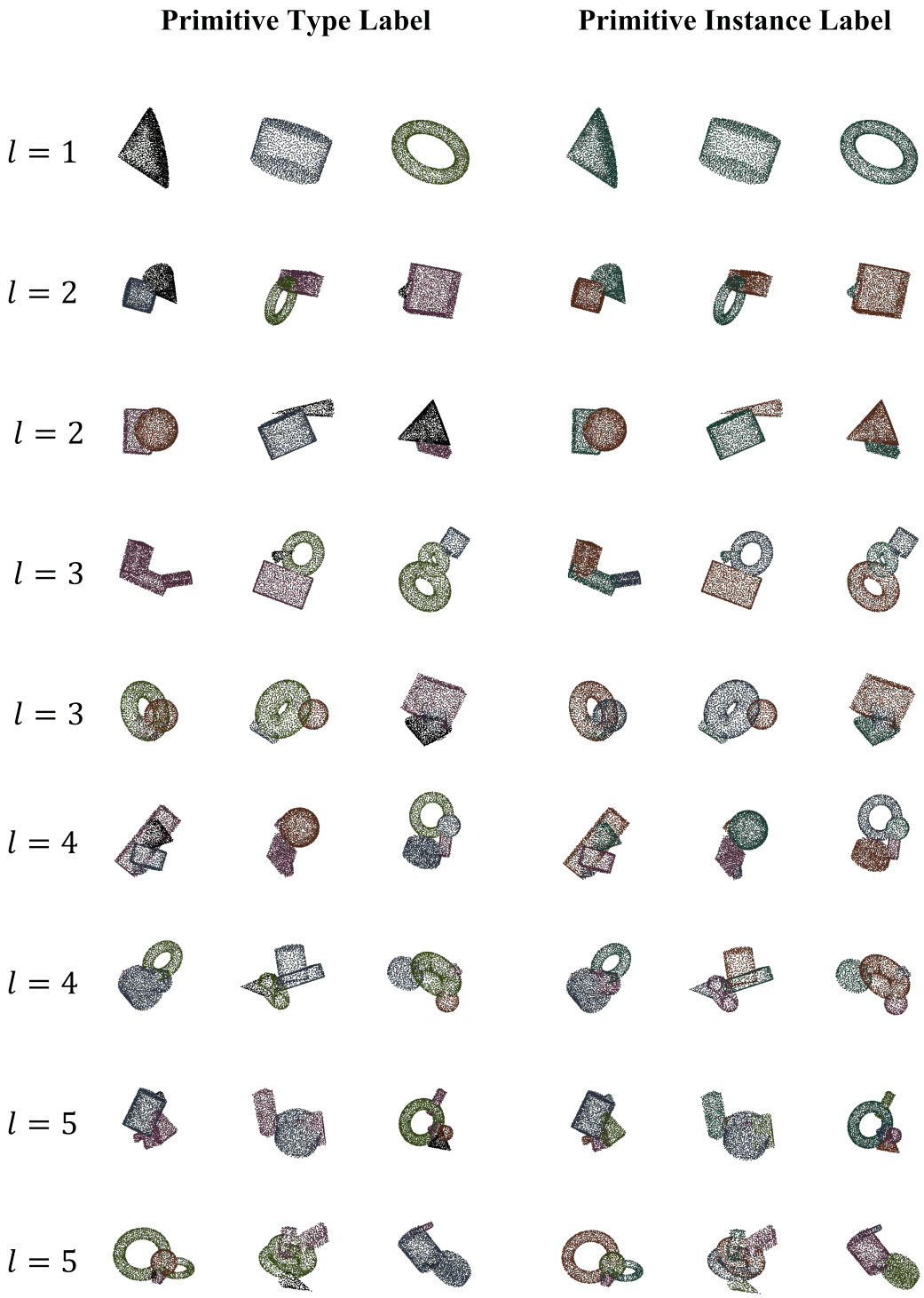

Figure 7. Visualization of Primitve3D samples with various leaf number l