PQ-NET: A Generative Part Seq2Seq Network for 3D Shapes Supplementary Materials

Rundi Wu[1] Yixin Zhuang[1] Kai Xu[2] Hao Zhang[3] Baoquan Chen[1]
[1]Center on Frontiers of Computing Studies, Peking University
[2]National University of Defense Technology
[3]Simon Fraser University

A. Overview

This supplementary material contains six parts:

- Sec.B describes the implementation detailed of our PQ-NET.
- Sec.C describes the data preparation details.
- Sec.D explains the metrics used for the evaluation of shape generation.
- Sec.E provides comparison results to 3D-PRNN on generation task.
- Sec.F provides more visual results of partial shape completion, random shape generation.

B. Implementation Details

In this section, we describe the detailed design and implementation of our PQ-NET architecture along with the training configuration.

Table 1 and Table 2 list the detailed architecture with specific parameters of our PQ-NET, divided into part geometry autoencoder and Seq2Seq autoencoder. For part geometry autoencoder, we use similar design as IM-NET [1], with skip connection in the implicit decoder. For Seq2Seq autoencoder, we also employ dropout regularization with drop rate 0.2 in the middle of GRU to reduce overfitting.

As mentioned in the main paper, we train our PQ-NET in two separate steps. We adopt a progressive strategy to train our part geometry autoencoder by increasing the resolution of part volum. Practically, we use resolution of 16^3, 32^3 and 64^3 with batch size 40 and learning rate 5e-4 in our experiments. Then the part geometry autoencoder is fixed and used to train the Seq2Seq autoencoder on the resolution of 64^3 with batch size as 64 and learning rate as 1e-3. We use PyTorch [4] framework to implement our PQ-NET and conduct all the experiments.

CNN Encoder			
Layer	Kernel Size	Stride	Output Shape
input voxel	-	-	(1,64,64,64)
Conv3D+BN+lReLU	(4,4,4)	(2,2,2)	(32,32,32,32)
Conv3D+BN+lReLU	(4,4,4)	(2,2,2)	(64,16,16,16)
Conv3D+BN+lReLU	(4,4,4)	(2,2,2)	(128,8,8,8)
Conv3D+BN+lReLU	(4,4,4)	(2,2,2)	(256,4,4,4)
Conv3D+Sigmoid	(4,4,4)	(1,1,1)	(128,1,1,1)
Implicit Decoder			
Layer	Dropout	Input Shape	Output Shape
feature+coordinates	-	(128 + 3)	(131)
FC+lReLU	0.2	(131)	(2048)
FC+lReLU	0.2	(2048 + 131)	(1024)
FC+lReLU	0.2	(1024 + 131)	(512)
FC+lReLU	0.2	(512 + 131)	(256)
FC+lReLU	-	(256 + 131)	(128)
FC+Sigmoid	-	(128)	(1)

Table 1. Architecture of our part geometry autoencoder. Conv3D: 3D Convolutional Layer, BN: Batch Normalization, lReLU: leaky ReLU, FC: Fully Connected Layer.

Seq2Seq Encoder RNN				
Type	#layers	input size	hidden size	bidirectional
GRU	2	(128+6+K)	256	True
Seq2Seq Decoder RNN				
Type	#layers	input size	hidden size	bidirectional
GRU	2	(128+6)	512	False
Seq2Seq Decoder FC				
Type		input size	hidden size	output size
FC-lReLU-FC		512	256	128
FC-ReLU-FC		512	128	6
FC-ReLU-FC		512	128	1

Table 2. Architecture of our Seq2Seq autoencoder. K is the maximum number of parts of all shaeps in the dataset.

C. Data Preparation Details

For all experiments in our paper, we mainly use three largest categories of PartNet [3], that is, chair, table and lamp. Since each shape in PartNet is partitioned into small elements and then grouped following a hierarchical structure with each node a semantical label, we use the nodes in the second layer as our part geometry. The part label we used appears in

the file "partnet-dataset\stats\after_merging_label_ids\xxx-label-2.txt", where "xxx" correspondes to the number of each shape categories. We remove shapes that contain more than 10 parts, resulting in 6305 chairs, 7357 tables and 1188 lamps, which are further divided into training, validation and test sets using official data splits of PartNet. Note that this upper bound of number of parts can be increased. Table 3 shows the statistics result about the number of parts per shape in our dataset.

	Chair	Table	Lamp
avg #parts	5.59	6.06	2.94
min #parts	2	2	2
max #parts	9	10	7

Table 3. Average, minimum and maximum number of parts per shape for each category in our dataset.

To prepare the training data for our network, we first voxelize the original shape mesh into voxel representation at 64^3 resolution and fill the interior of the shape voxel using classic flood filling algorithm. Each part is then scaled to 64^3 resolution within its bounding box. The resulting 64^3 part volums are downsampled to 32^3 and 16^3 resolution. With voxelized part geometry at differ resolutions, we follow the sampling approach as in IM-NET to progressively sample points near the surface with each point a signed distance to the surface. Specifically, we sample 4096 points in 16^3 volum, and with higher resolutions, such as 32^3 and 64^3, we sample 8192 and 32768, respectively. The sampled points together with signed distance values are used to train our part geometry autoencoder. The box parameters used in Seq2Seq autoencoder training are produced by calculating bounding box of each part in original shape voxel. Intuitively, the box parameters indicate the deviation and translation from part local frame to the shape coordinate system.

D. Metrics

We explain the quantitative metrics adopt for generation task in our paper, *i.e*. Coverage (COV), Minimum Matching Distance (MMD) and Jensen-Shannon Divergence (JSD) []. In their calculation, chamfer distance is used when comparing our method to IM-NET [1] and StructureNet [2] while IoU is used when comparing to 3D-PRNN [5].

Let $\mathcal{G}$ be a set of generated shapes and $\mathcal{S}$ be the ground truth test set.

COV To compute COV, for each shape in $\mathcal{G}$ we find its nearest neighbor in $\mathcal{S}$ and mark it as matched. COV is the fraction of matched shapes in $\mathcal{S}$ over the total size of $\mathcal{S}$. COV roughly represents the diversity of the generated shapes. A high COV score suggests most of shapes in $\mathcal{S}$ can be roughly represented by shapes in $\mathcal{G}$.

Metrics	Method	Chair	Table	Lamp	Avg
COV-IoU	Ours	**58.94**	**59.80**	**76.34**	**65.03**
	3D-PRNN	51.03	36.44	58.93	48.80
MMD-IoU	Ours	**0.259**	**0.271**	**0.298**	**0.276**
	3D-PRNN	0.275	0.357	0.347	0.326

Table 4. Quantitative comparison to 3D-PRNN for 3D shape generation on three categories: chair, table, lamp.

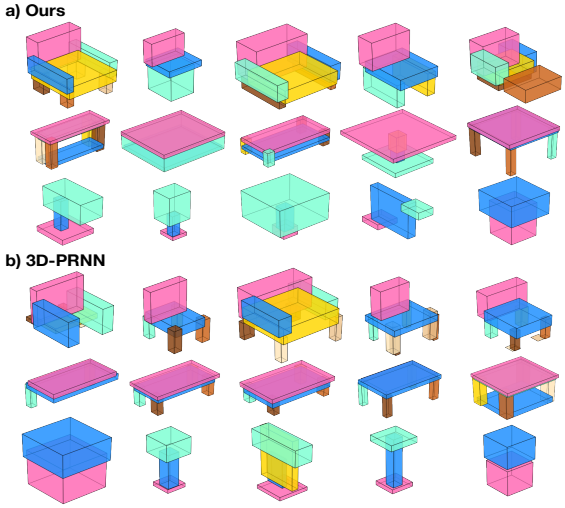

Figure 1. More visual comparison of random generated 3D primitives between ours and 3D-PRNN.

MMD To compute MMD, for each shape in $\mathcal{S}$, we calculate the distance to its nearest neighbor in $\mathcal{G}$. Then MMD is defined as the average of all these distances. MMD roughly represents the fidelity of the generated shapes.

JSD In a predefined voxel gird, for each shape of point cloud form in $\mathcal{G}$, we count the number of points lying inside each voxel, and do the same thing for $\mathcal{S}$. Then we get two distribution in Euclidean 3D space $P_{\mathcal{G}}$ and $P_{\mathcal{S}}$. JSD is defined as the Jensen-Shannon Divergence between the two distribution.

We use the code from `https://github.com/optas/latent_3d_points` for the calculation of the above metrics.

E. Comparison to 3D-PRNN on Shape Generation Task

In this section, we demonstrate detailed comparison results to 3D-PRNN [5] on shape generation task, which are not fully shown in the main paper due to paper length limitation. Unlike our PQ-NET that generates new shapes from random noise, 3D-PRNN samples new structure within a constraint region. For a fair comparison, we follow the setting described in their paper, by sampling the first input feature of RNN from training data.

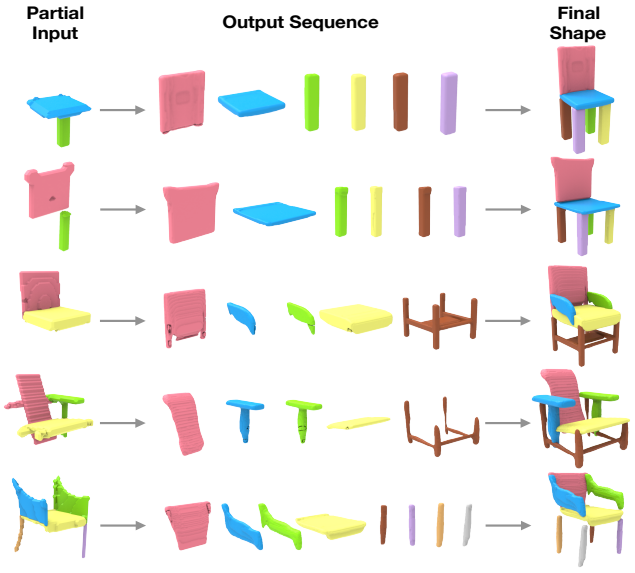

Figure 2. Visual results of partial shape completion.

Results of quantitative comparison on are shown in and Table 4. We sampled 2000 random generated shapes for chair and table, 800 for lamp, to compute the coverage(COV) and minimum matching distance(MMD) between the generated set and ground truth test set. We use $1 - \text{IoU}$ as the distance measure when comparing two shapes. It can be seen that our network outperforms 3D-PRNN in all of the measurements, which means that our generation results are more diverse and plausible. More visual results are shown in Figure 1.

F. More Results

Figure 2 shows visual results for partial shape completion and Figure 3 at the last page shows more results of our generated shapes.

References

[1] Z. Chen and H. Zhang. Learning implicit fields for generative shape modeling. *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019.

[2] K. Mo, P. Guerrero, L. Yi, H. Su, P. Wonka, N. Mitra, and L. J. Guibas. Structurenet: Hierarchical graph networks for 3d shape generation. *ACM Trans. on Graph. (SIGGRAPH Asia)*, 2019.

[3] K. Mo, S. Zhu, A. X. Chang, L. Yi, S. Tripathi, L. J. Guibas, and H. Su. PartNet: A large-scale benchmark for fine-grained and hierarchical part-level 3D object understanding. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019.

[4] A. Paszke, S. Gross, S. Chintala, G. Chanan, E. Yang, Z. DeVito, Z. Lin, A. Desmaison, L. Antiga, and A. Lerer. Automatic differentiation in PyTorch. In *NIPS Autodiff Workshop*, 2017.

[5] C. Zou, E. Yumer, J. Yang, D. Ceylan, and D. Hoiem. 3D-PRNN: Generating shape primitives with recurrent neural networks. *2017 IEEE International Conference on Computer Vision (ICCV)*, Oct 2017.

Figure 3. More visual results of our generated shapes (row 1-6), along with two latent space interpolation (row 7-8). All shapes are sampled at resolution 256^3 and reconstructed using Marching Cubes.