PoinTr: Diverse Point Cloud Completion with Geometry-Aware Transformers
Supplementary Material

A. Implementation Details

Our proposed method PointTr is implemented with PyTorch [3]. We utilize AdamW optimizer [2] to train the network with initial learning rate as 0.0005 and weight decay as 0.0005. In all of our experiments, we set the depth of the encoder and decoder in our transformer to 6 and 8 and set k of kNN operation to 16 and 8 for the DGCNN feature extractor and the geometry-aware block respectively. We use 6 head attention for all transformer blocks and set their hidden dimensions to 384. On the PCN dataset, the network takes 2048 points as inputs and is required to complete the other 14336 points. We set the batch size to 54 and train the model for 300 epochs with the continuous learning rate decay of 0.9 for every 20 epochs. We set N to 128 and M to 224. On ShapeNet-55/34, the model takes 2048 points as inputs and is required to complete the other 6144 points. We set the batch size to 128 and train the model for 200 epochs with the continuous learning rate decay of 0.76 for every 20 epochs. We set N to 128 and M to 96.

We employ a lightweight DGCNN [6] model to extract the point proxy features. To reduce the computational cost, we hierarchically downsample the original input point cloud to $N = 128$ center points and use several DGCNN layers to capture local geometric relationships. The detailed network architecture is: $\texttt{Linear}(C_{in} = 3, C_{out} = 8) \rightarrow \texttt{DGCNN}(C_{in} = 8, C_{out} = 32, K = 8, N_{out} = 2048) \rightarrow \texttt{DGCNN}(C_{in} = 32, C_{out} = 64, K = 8, N_{out} = 512) \rightarrow \texttt{DGCNN}(C_{in} = 64, C_{out} = 64, K = 8, N_{out} = 512) \rightarrow \texttt{DGCNN}(C_{in} = 64, C_{out} = 128, K = 8, N_{out} = 128)$, where C_{in} and C_{out} are the numbers of channels of input and output features, N_{out} is the number of points after FPS.

B. Technical Details on Transformers

Encoder-Decoder Architecture. The overall architecture of the transformer encoder-decoder networks is illustrated in Figure 1. The point proxies are passed through the transformer encoder with N multi-head self-attention layers and feed-forward network layers. Then, the decoder receives the generated query embeddings and encoder memory, and produces the final set of predicted point proxies that represents the missing part of the point cloud through N multi-head self-attention layers, decoder-encoder attention layers and feed-forward network layers. We set N to 6 in all our experiments following common practice [5].

Multi-head Attention. Multi-head attention mechanism allows the network to jointly attend to information from different representation subspaces at different positions [5]. Speciacally, given the input values V, keys K and queries Q, the multi-head attention is computed by:

$$\text{MultiHead}(Q, K, V) = W^O \text{Concat}(\text{head}_1, ..., \text{head}_h),$$

where W^O the weights of the output linear layer and each head feature can be obtained by:

$$\text{head}_i = \text{softmax}(\frac{QW_i^Q(KW_i^K)^T}{\sqrt{d_k}})VW_i^V$$

where W_i^Q, W_i^K and W_i^V are the linear layers that project the inputs to different subspaces and d_k is the dimension of the input features.

Feed-forward network (FFN). Following [5], we use two linear layers with ReLU activations and dropout as the feed-forward network.

C. Detailed Experimental Results

Detailed results on ShapeNet-55: In Table 1, we report the detailed results for FoldingNet [8], PCN [9], TopNet [4], PFNet [1], GRNet [7] and the proposed method on ShapeNet-55. Each row in the table stands for a category of object. We test each method under three settings: simple, moderate and hard.

Detailed results on ShapeNet-34: In Table 2, we report the detailed results for the novel objects from 21 categories in ShapeNet-34. Each row in the table stands for a category of object. We test each method under the three settings: simple, moderate and hard.

D. Complexity Analysis

Our method achieves the best performance on both our newly proposed diverse benchmarks and the existing benchmarks. We provide the detailed complexity analysis of our

Table 1: Detailed results on ShapeNet-55. *S.*, *M.* and *H.* stand for the simple, moderate and hard settings.

CD-ℓ_2($\times$ 1000)	FoldingNet [8]			PCN [9]			TopNet [4]			PFNet [1]			GRNet [7]			Ours-PoinTr		
	S.	M.	H.	S.	M.	H.	S.	M.	H.	S.	M.	H.	S.	M.	H.	S.	M.	H.
airplane	1.36	1.28	1.7	0.9	0.89	1.32	1.02	0.99	1.48	1.35	1.44	2.69	0.87	0.87	1.27	**0.27**	**0.38**	**0.69**
trash bin	2.93	2.9	5.03	2.16	2.18	5.15	2.51	2.32	5.03	4.03	3.39	9.63	1.69	2.01	3.48	**0.8**	**1.15**	**2.15**
bag	2.31	2.38	3.67	2.11	2.04	4.44	2.36	2.23	4.21	3.63	3.66	7.6	1.41	1.7	2.97	**0.53**	**0.74**	**1.51**
basket	2.98	2.77	4.8	2.21	2.1	4.55	2.62	2.43	5.71	4.74	3.88	8.47	1.65	1.84	3.15	**0.73**	**0.88**	**1.82**
bathtub	2.68	2.66	4.0	2.11	2.09	3.94	2.49	2.25	4.33	3.64	3.5	5.74	1.46	1.73	2.73	**0.64**	**0.94**	**1.68**
bed	4.24	4.08	5.65	2.86	3.07	5.54	3.13	3.1	5.71	4.44	5.36	9.14	1.64	2.03	3.7	**0.76**	**1.1**	**2.26**
bench	1.94	1.77	2.36	1.31	1.24	2.14	1.56	1.39	2.4	2.17	2.16	4.11	1.03	1.09	1.71	**0.38**	**0.52**	**0.94**
birdhouse	4.06	4.18	5.88	3.29	3.53	6.69	3.73	3.98	6.8	3.96	5.0	9.66	1.87	2.4	4.71	**0.98**	**1.49**	**3.13**
bookshelf	3.04	3.03	3.91	2.7	2.7	4.61	3.11	2.87	4.87	3.19	3.47	5.72	1.42	1.71	2.78	**0.71**	**1.06**	**1.93**
bottle	1.7	1.91	4.02	1.25	1.43	4.61	1.56	1.66	4.02	2.37	2.89	10.03	1.05	1.44	2.67	**0.37**	**0.74**	**1.5**
bowl	2.79	2.6	4.23	2.05	1.83	3.66	2.33	1.98	4.82	4.3	3.97	8.76	1.6	1.77	2.99	**0.68**	**0.78**	**1.44**
bus	1.47	1.42	2.0	1.2	1.14	2.08	1.32	1.21	2.29	2.06	1.88	3.75	1.06	1.16	1.48	**0.42**	**0.55**	**0.79**
cabinet	2.0	1.86	2.79	1.6	1.49	3.47	1.91	1.65	3.36	2.72	2.37	4.73	1.27	1.41	2.09	**0.55**	**0.66**	**1.16**
camera	5.5	6.04	8.87	4.05	4.54	8.27	4.75	4.98	9.24	6.57	8.04	13.11	2.14	3.15	6.09	**1.1**	**2.03**	**4.34**
can	2.84	2.68	5.71	2.02	2.28	6.48	2.67	2.4	5.5	5.65	4.05	16.29	1.58	2.11	3.81	**0.68**	**1.19**	**2.14**
cap	4.1	4.04	5.87	1.82	1.76	4.2	3.0	2.69	5.59	10.92	9.04	20.3	1.17	1.37	3.05	**0.46**	**0.62**	**1.64**
car	1.81	1.81	2.31	1.48	1.47	2.6	1.71	1.65	3.17	2.06	2.1	3.43	1.29	1.48	2.14	**0.64**	**0.86**	**1.25**
cellphone	1.04	1.06	1.87	0.8	0.79	1.71	1.01	0.96	1.8	1.25	1.37	3.65	0.82	0.91	1.18	**0.32**	**0.39**	**0.6**
chair	2.37	2.46	3.62	1.7	1.81	3.34	1.97	2.04	3.59	2.94	3.48	6.34	1.24	1.56	2.73	**0.49**	**0.74**	**1.63**
clock	2.56	2.41	3.46	2.1	2.01	3.98	2.48	2.16	4.03	3.15	3.27	6.03	1.46	1.66	2.67	**0.62**	**0.84**	**1.65**
keyboard	1.21	1.18	1.32	0.82	0.82	1.04	0.88	0.83	1.15	0.83	1.06	1.97	0.74	0.81	1.09	**0.3**	**0.39**	**0.45**
dishwasher	2.6	2.17	3.5	1.93	1.66	4.39	2.43	1.74	4.64	4.57	3.23	6.39	1.43	1.59	2.53	**0.55**	**0.69**	**1.42**
display	2.15	2.24	3.25	1.56	1.66	3.26	1.84	1.85	3.48	2.27	2.83	5.52	1.13	1.38	2.29	**0.48**	**0.67**	**1.33**
earphone	6.37	6.48	9.14	3.13	2.94	7.56	4.36	4.47	8.36	15.07	17.5	33.37	1.78	2.18	5.33	**0.81**	**1.38**	**3.78**
faucet	4.46	4.39	7.2	3.21	3.48	7.52	3.61	3.59	7.25	5.68	6.79	14.29	1.81	2.32	4.91	**0.71**	**1.42**	**3.49**
filecabinet	2.59	2.48	3.76	2.02	1.97	4.14	2.41	2.12	4.12	3.72	3.57	7.13	1.46	1.71	2.89	**0.63**	**0.84**	**1.69**
guitar	0.65	0.6	1.25	0.42	0.38	1.23	0.57	0.47	1.42	0.74	0.89	5.41	0.44	0.48	0.76	**0.14**	**0.21**	**0.42**
helmet	5.39	5.37	7.96	3.76	4.18	7.53	4.36	4.55	7.73	9.55	8.41	15.44	2.33	3.18	6.03	**0.99**	**1.93**	**4.22**
jar	3.65	3.87	6.51	2.57	2.82	6.0	3.03	3.17	7.03	5.44	5.56	11.87	1.72	2.37	4.37	**0.77**	**1.33**	**2.87**
knife	1.29	0.87	1.21	0.94	0.62	1.37	0.84	0.68	1.44	2.11	1.53	3.89	0.72	0.66	0.96	**0.2**	**0.33**	**0.56**
lamp	3.93	4.23	6.87	3.1	3.45	7.02	3.03	3.39	8.15	6.82	7.61	14.22	1.68	2.43	5.17	**0.64**	**1.4**	**3.58**
laptop	1.02	1.04	1.96	0.75	0.79	1.59	0.8	0.85	1.66	1.04	1.21	2.46	0.83	0.87	1.28	**0.32**	**0.34**	**0.6**
loudspeaker	3.21	3.15	4.55	2.5	2.45	5.08	3.1	2.76	5.32	4.32	4.19	7.6	1.75	2.08	3.45	**0.78**	**1.16**	**2.17**
mailbox	2.44	2.61	4.98	1.66	1.74	5.18	2.16	2.1	5.1	3.82	4.2	10.51	1.15	1.59	3.42	**0.39**	**0.78**	**2.56**
microphone	4.42	5.06	7.04	3.44	3.9	8.52	2.83	3.49	6.87	6.58	7.56	16.74	2.09	2.76	5.7	**0.7**	**1.66**	**4.48**
microwaves	2.67	2.48	4.43	2.2	2.01	4.65	2.65	2.15	5.07	4.63	3.94	6.52	1.51	1.72	2.76	**0.67**	**0.83**	**1.82**
motorbike	2.63	2.55	3.52	2.03	2.01	3.13	2.29	2.25	3.54	2.17	2.48	5.09	1.38	1.52	2.26	**0.75**	**1.1**	**1.92**
mug	3.66	3.67	5.7	2.45	2.48	5.17	2.89	2.56	5.43	4.76	4.3	8.37	1.75	2.16	3.79	**0.91**	**1.17**	**2.35**
piano	3.86	4.04	6.04	2.64	2.74	4.83	2.99	2.89	5.64	4.57	5.26	9.26	1.53	1.82	3.21	**0.76**	**1.06**	**2.23**
pillow	2.33	2.38	3.87	1.85	1.81	3.68	2.31	2.26	4.19	4.21	3.82	7.89	1.42	1.67	3.04	**0.61**	**0.82**	**1.56**
pistol	1.92	1.62	2.52	1.25	1.17	2.65	1.5	1.3	2.62	2.27	2.09	7.2	1.11	1.06	1.76	**0.43**	**0.66**	**1.3**
flowerpot	4.53	4.68	6.46	3.32	3.39	6.04	3.61	3.45	6.28	4.83	5.51	10.68	2.02	2.48	4.19	**1.01**	**1.51**	**2.77**
printer	3.66	4.01	5.34	2.9	3.19	5.84	3.04	3.19	5.84	5.56	6.06	9.29	1.56	2.38	4.24	**0.73**	**1.21**	**2.47**
remote	1.14	1.2	1.98	0.99	0.97	2.04	1.14	1.17	2.16	1.74	2.37	4.61	0.89	1.05	1.29	**0.36**	**0.53**	**0.71**
rifle	1.27	1.02	1.37	0.98	0.8	1.31	0.98	0.86	1.46	1.72	1.45	3.02	0.83	0.77	1.16	**0.3**	**0.45**	**0.79**
rocket	1.37	1.18	1.88	1.05	1.04	1.87	1.04	1.0	1.93	1.65	1.61	3.82	0.78	0.92	1.44	**0.23**	**0.48**	**0.99**
skateboard	1.58	1.58	2.07	1.04	0.94	1.68	1.08	1.05	1.84	1.43	1.6	3.09	0.82	0.87	1.24	**0.28**	**0.38**	**0.62**
sofa	2.22	2.09	3.14	1.65	1.61	2.92	1.93	1.76	3.39	2.65	2.53	4.84	1.35	1.45	2.32	**0.56**	**0.67**	**1.14**
stove	2.69	2.63	3.99	2.07	2.02	4.72	2.44	2.16	4.84	4.03	3.71	7.15	1.46	1.72	3.22	**0.63**	**0.92**	**1.73**
table	2.23	2.15	3.21	1.56	1.5	3.36	1.78	1.65	3.21	3.03	3.11	5.74	1.15	1.33	2.33	**0.46**	**0.64**	**1.31**
telephone	1.07	1.06	1.75	0.8	0.8	1.67	1.02	0.95	1.78	1.3	1.47	3.37	0.81	0.89	1.18	**0.31**	**0.38**	**0.59**
tower	2.46	2.45	3.91	1.91	1.97	4.47	2.15	2.05	4.51	3.13	3.54	9.87	1.26	1.69	3.06	**0.55**	**0.9**	**1.95**
train	1.86	1.68	2.32	1.5	1.41	2.37	1.59	1.44	2.51	2.01	2.03	4.1	1.09	1.14	1.61	**0.5**	**0.7**	**1.12**
watercraft	1.85	1.69	2.49	1.46	1.39	2.4	1.53	1.42	2.67	2.1	2.13	4.58	1.09	1.12	1.65	**0.41**	**0.62**	**1.07**
washer	3.47	3.2	4.89	2.42	2.31	6.08	2.92	2.53	6.53	5.55	4.11	7.04	1.72	2.05	4.19	**0.75**	**1.06**	**2.44**
mean	2.68	2.66	4.06	1.96	1.98	4.09	2.26	2.17	4.31	3.84	3.88	8.03	1.35	1.63	2.86	**0.58**	**0.88**	**1.8**

Table 2: Detailed results for the novel objects on ShapeNet-34. *S.*, *M.* and *H.* stand for the simple, moderate and hard settings.

CD-ℓ_2 ($\times$ 1000)	FoldingNet [8]			PCN [9]			TopNet [4]			PFNet [1]			GRNet [7]			Ours-PoinTr		
	S.	M.	H.	S.	M.	H.	S.	M.	H.	S.	M.	H.	S.	M.	H.	S.	M.	H.
bag	2.15	2.27	3.99	2.48	2.46	3.94	2.08	1.95	4.36	3.88	4.42	9.67	1.47	1.88	3.45	**0.96**	**1.34**	**2.08**
basket	2.37	2.2	4.87	2.79	2.51	4.78	2.46	2.11	5.18	4.47	4.55	14.46	1.78	1.94	4.18	**1.04**	**1.4**	**2.9**
birdhouse	3.27	3.15	5.62	3.53	3.47	5.31	3.17	2.97	5.89	3.9	4.65	9.88	1.89	2.34	5.16	**1.22**	**1.79**	**3.45**
bowl	2.61	2.3	4.55	2.66	2.35	3.97	2.46	2.16	4.84	4.35	5.0	14.59	1.77	1.97	3.9	**1.05**	**1.32**	**2.4**
camera	4.4	4.78	7.85	4.84	5.3	8.03	4.24	4.43	8.11	6.78	8.04	13.91	2.31	3.38	7.2	**1.63**	**2.67**	**4.97**
can	1.95	1.73	5.86	1.95	1.89	5.21	2.02	1.7	5.82	2.95	3.47	23.02	1.53	1.8	3.08	**0.8**	**1.17**	**2.85**
cap	6.07	5.98	11.49	7.21	7.14	10.94	4.68	4.23	9.17	14.11	14.86	28.23	3.29	4.87	13.02	**1.4**	**2.74**	**8.35**
keyboard	0.98	0.96	1.35	1.07	1.0	1.23	0.79	0.77	1.55	1.13	1.16	2.58	0.73	0.77	1.11	**0.43**	**0.45**	**0.63**
dishwasher	2.09	1.8	4.55	2.45	2.09	3.53	2.51	1.77	4.72	3.44	3.78	9.31	1.79	1.7	3.27	**0.93**	**1.05**	**2.04**
earphone	6.86	6.96	12.77	7.88	6.59	16.53	5.33	4.83	11.67	20.31	23.21	39.49	4.29	**4.16**	**10.3**	**2.03**	5.1	10.69
helmet	4.86	5.04	8.86	6.15	6.41	9.16	4.89	4.86	8.73	8.78	10.07	21.2	3.06	4.38	10.27	**1.86**	**3.3**	**6.96**
mailbox	2.2	2.29	4.49	2.74	2.68	4.31	2.35	2.2	4.91	5.2	5.33	10.94	1.52	1.9	4.33	**1.03**	**1.47**	**3.34**
microphone	2.92	3.27	8.54	4.36	4.65	8.46	3.03	3.2	7.15	6.39	7.99	19.41	2.29	3.23	8.41	**1.25**	**2.27**	**5.47**
microwaves	2.29	2.12	5.17	2.59	2.35	4.47	2.67	2.12	5.41	3.89	4.08	9.01	1.74	1.81	3.82	**1.01**	**1.18**	**2.14**
pillow	2.07	2.11	3.73	2.09	2.16	3.54	2.08	2.05	4.01	4.15	4.29	12.01	1.43	1.69	3.43	**0.92**	**1.24**	**2.39**
printer	3.02	3.23	5.53	3.28	3.6	5.56	2.9	2.96	6.07	5.38	5.94	10.29	1.82	2.41	5.09	**1.18**	**1.76**	**3.1**
remote	0.89	0.92	1.85	0.95	1.08	1.58	0.89	0.89	2.28	1.51	1.75	6.0	0.82	1.02	1.29	**0.44**	**0.58**	**0.78**
rocket	1.28	1.09	2.0	1.39	1.22	2.01	1.14	0.96	2.03	1.84	1.51	4.01	0.97	0.79	1.6	**0.39**	**0.72**	**1.39**
skateboard	1.53	1.42	1.99	1.97	1.78	2.45	1.23	1.2	2.01	2.43	2.53	4.25	0.93	1.07	1.83	**0.52**	**0.8**	**1.31**
tower	2.25	2.25	4.74	2.37	2.4	4.35	2.2	2.17	5.47	3.38	4.15	13.11	1.35	1.8	3.85	**0.82**	**1.35**	**2.48**
washer	2.58	2.34	5.5	2.77	2.52	4.64	2.63	2.14	6.57	4.53	4.27	9.23	1.83	1.97	5.28	**1.04**	**1.39**	**2.73**
mean	2.79	2.77	5.49	3.22	3.13	5.43	2.65	2.46	5.52	5.37	5.95	13.55	1.84	2.23	4.95	**1.05**	**1.67**	**3.45**

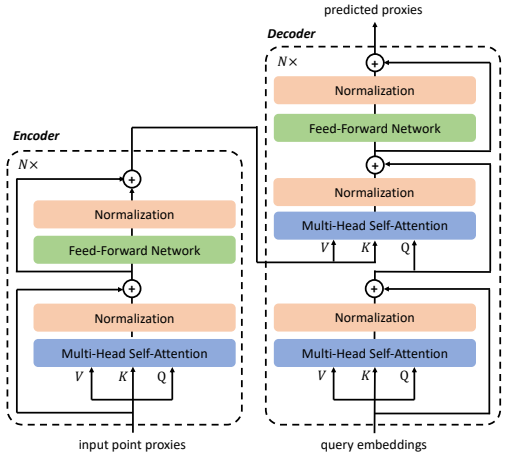

Figure 1: The overall architecture of the transformer encoder-decoder networks.

Table 3: Complexity analysis. We report the the number of parameter (Params) and theoretical computation cost (FLOPs) of our method and five existing methods. We also provide the average Chamfer distances of all categories in ShapeNet-55 (CD_{55}) and unseen categories in ShapeNet34 (CD_{34}) as references.

Models	Params	FLOPs	CD_{55}	CD_{34}
FoldingNet [8]	2.30 M	27.58 G	3.12	3.62
PCN [9]	5.04 M	15.25 G	2.66	3.85
TopNet [4]	5.76 M	6.72 G	2.91	3.50
PFNet [1]	73.05 M	4.96 G	5.22	8.16
GRNet [7]	73.15 M	40.44 G	1.97	2.99
PoinTr	30.9 M	10.41 G	1.07	2.05

method in Table 3. We report the number of parameters and theoretical computation cost (FLOPs) of our method and other five methods. We also provide the average Chamfer distances of all categories in ShapeNet-55 and unseen categories in ShapeNet34 as references. We can see our method achieves the best performance while using relatively low parameters and FLOPs among the methods in the table, which shows our method offers a decent trade-off between cost and performance.

E. Visualization of the Predicted Centers

We visualize the local center prediction results on ShapeNet-55. We adopt a coarse-to-fine strategy to recover the point cloud. Our method starts with the prediction of local centers, then we can obtain the final results by adding the points around the centers. As shown in Figure 2, Line (a) shows the input partial point cloud and the predicted point centers. Line (b) is the predicted point clouds. We see the predicted point proxies can successfully represent the overall structure of the point cloud and the details then are added in the final predictions.

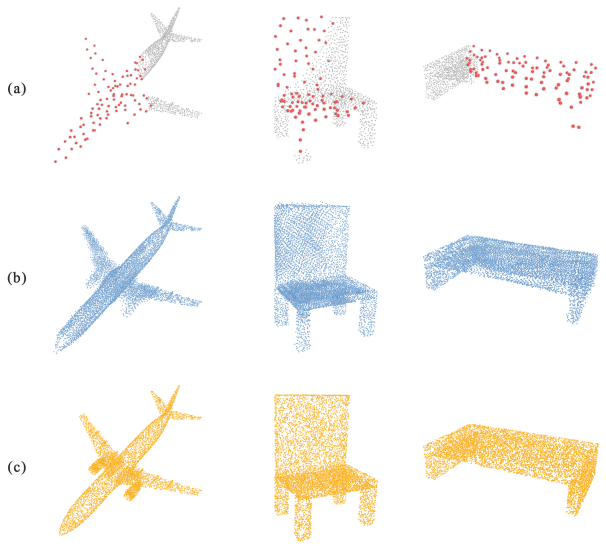

Figure 2: Visualization of predicted points proxies. In Line (a), we show the input partial point clouds and the predicted centers. Based on predicted point proxies, we can easily predicted the accurate point centers and then complete the point clouds, as shown in Line (b). We show the ground-truth point cloud in Line (c) for comparisons.

F. Qualitative Results

In Figure 3, we provide more qualitative results on ShapeNet-55. We see our results are much better than baseline methods visually.

References

[1] Zitian Huang, Yikuan Yu, Jiawen Xu, Feng Ni, and Xinyi Le. Pf-net: Point fractal network for 3d point cloud completion. In *CVPR*, pages 7659–7667, 2020. 1, 2, 3

[2] Ilya Loshchilov and Frank Hutter. Fixing weight decay regularization in adam. 2018. 1

[3] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. Pytorch: An imperative style, high-performance deep learning library. In *NeurIPS*, pages 8026–8037, 2019. 1

[4] Lyne P. Tchapmi, Vineet Kosaraju, Hamid Rezatofighi, Ian D. Reid, and Silvio Savarese. Topnet: Structural point cloud decoder. In *CVPR*, pages 383–392, 2019. 1, 2, 3

[5] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. In Isabelle Guyon, Ulrike von Luxburg, Samy Bengio, Hanna M. Wallach, Rob Fergus, S. V. N. Vishwanathan, and Roman Garnett, editors, *NeurIPS*, pages 5998–6008, 2017. 1

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

[7] Haozhe Xie, Hongxun Yao, Shangchen Zhou, Jiageng Mao, Shengping Zhang, and Wenxiu Sun. Grnet: Gridding residual network for dense point cloud completion. In Andrea Vedaldi, Horst Bischof, Thomas Brox, and Jan-Michael Frahm, editors, *ECCV*, pages 365–381, 2020. 1, 2, 3

[8] Yaoqing Yang, Chen Feng, Yiru Shen, and Dong Tian. Foldingnet: Interpretable unsupervised learning on 3d point clouds. *CoRR*, abs/1712.07262, 2017. 1, 2, 3

[9] Wentao Yuan, Tejas Khot, David Held, Christoph Mertz, and Martial Hebert. PCN: point completion network. In *3DV*, pages 728–737, 2018. 1, 2, 3

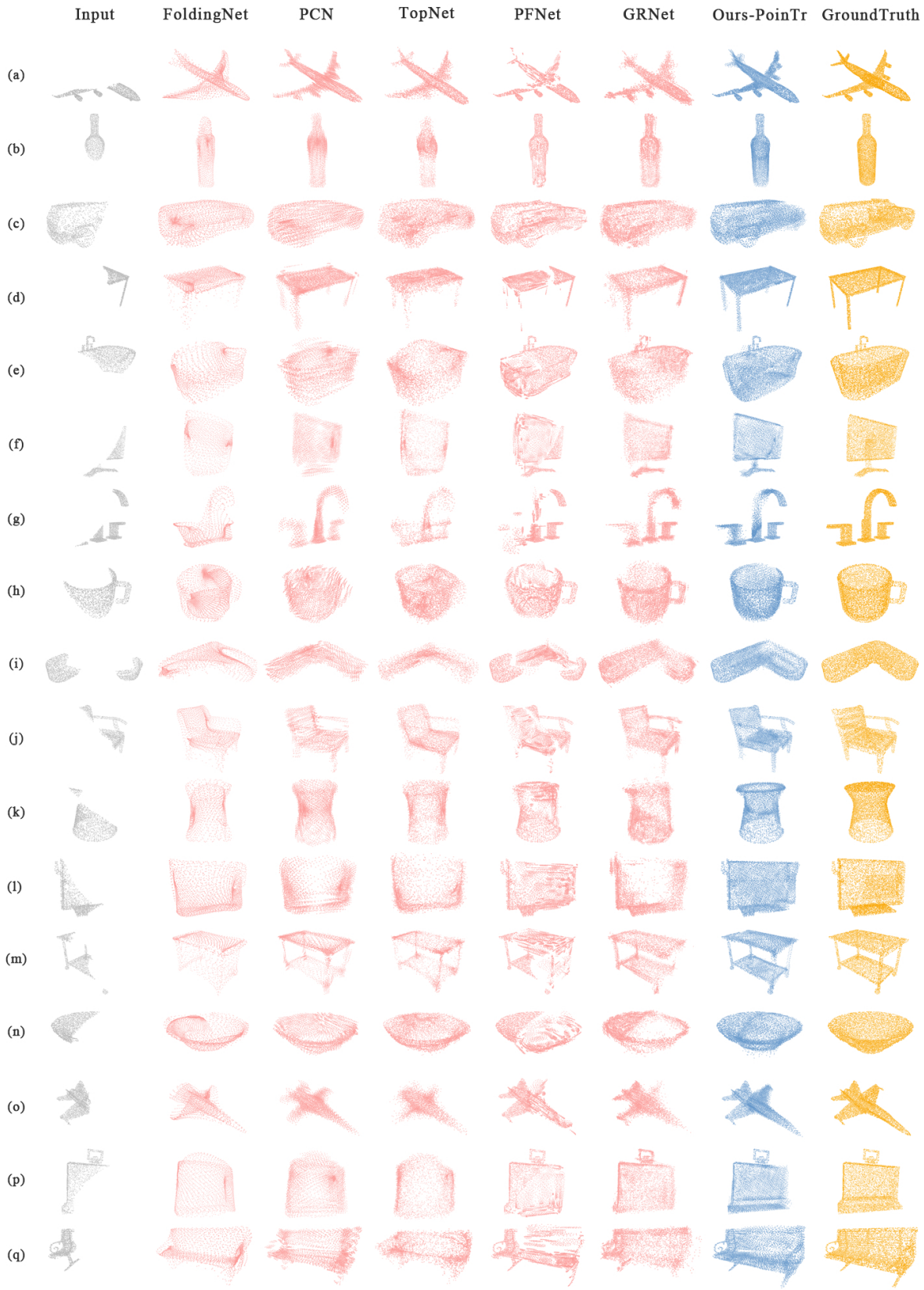

Figure 3: More qualitative results on ShapeNet-55.