Progressive Seed Generation Auto-encoder for Unsupervised Point Cloud Learning (Supplementary)

1. Implementation Details

Here we provide some specific implementation details of our architecture in Section 3.2 of the paper. We set the number of input points N to 2048, which has been commonly used in previous studies. We implement PointNet++ [17] as our encoder, with the same structure as the original paper. Therefore, the encoder samples the input point cloud into 512 and 128 points and finally transforms it to create a 1024-dimensional feature vector. We added one fc layer to convert this into a 512-dimensional θ'. Our decoder consists of one SGM and three SFPMs, followed by point generation layers, which finally generates a $(32 \times 64 = 2048) \times 3$ point cloud as the output. Each convolutional layer in the SGM has output channel dimensions of 512, 256, 128, 64, and 48. The size of the feature map becomes $(512 \times 1 \times 1)$ (θ') $\rightarrow$ $(512 \times 3 \times 3)$ $\rightarrow$ $(256 \times 6 \times 6)$ (Seed 1) $\rightarrow$ $(128 \times 12 \times 12)$ (Seed 2) $\rightarrow$ $(64 \times 24 \times 24)$ (Seed 3) $\rightarrow$ $(48 \times 48 \times 48)$ (Seed 4). The 1×1 conv layers in the SFPM have an output channel dimension of 256, and in the point generation layers, the interpolation layer changes the spatial size of (48×48) to (32×64). The 1×1 conv layer outputs 256-channel features and the last fc layer outputs 3D point coordinates.

2. Point Cloud Upsampling

In decoder ablation, we perform point cloud upsampling by setting the number of output point clouds to 6400 and show the examples with the original output results in Figure 1. It can be seen that Decoder C_{32} produces a result that is the closest to the ground truth than other decoders and produces a more clear, uniform output.

3. Ablation Models

For better understanding, we visualize architectures of the decoders used in analytical study. Figure 2 depicts four Decoders, A, B, C_2, and C_{32}. We ensure that the analysis was performed under the same conditions by unifying the structure of the encoder and decoder except for the part that generates U'.

4. Qualitative Comparison

We present additional qualitative results on point cloud reconstruction. For comparison, we visualize the results of 3D-PointCapsNet [34] and our results in Figure 3. To create the results of 3D-PointCapsNet, we used publicly available code and pre-trained weights provided by the authors[1]. Figure 3 shows that our method generally produces output with lower noise and better detail than 3D-PointCapsNet.

[1]https://github.com/yongheng1991/3D-point-capsule-networks

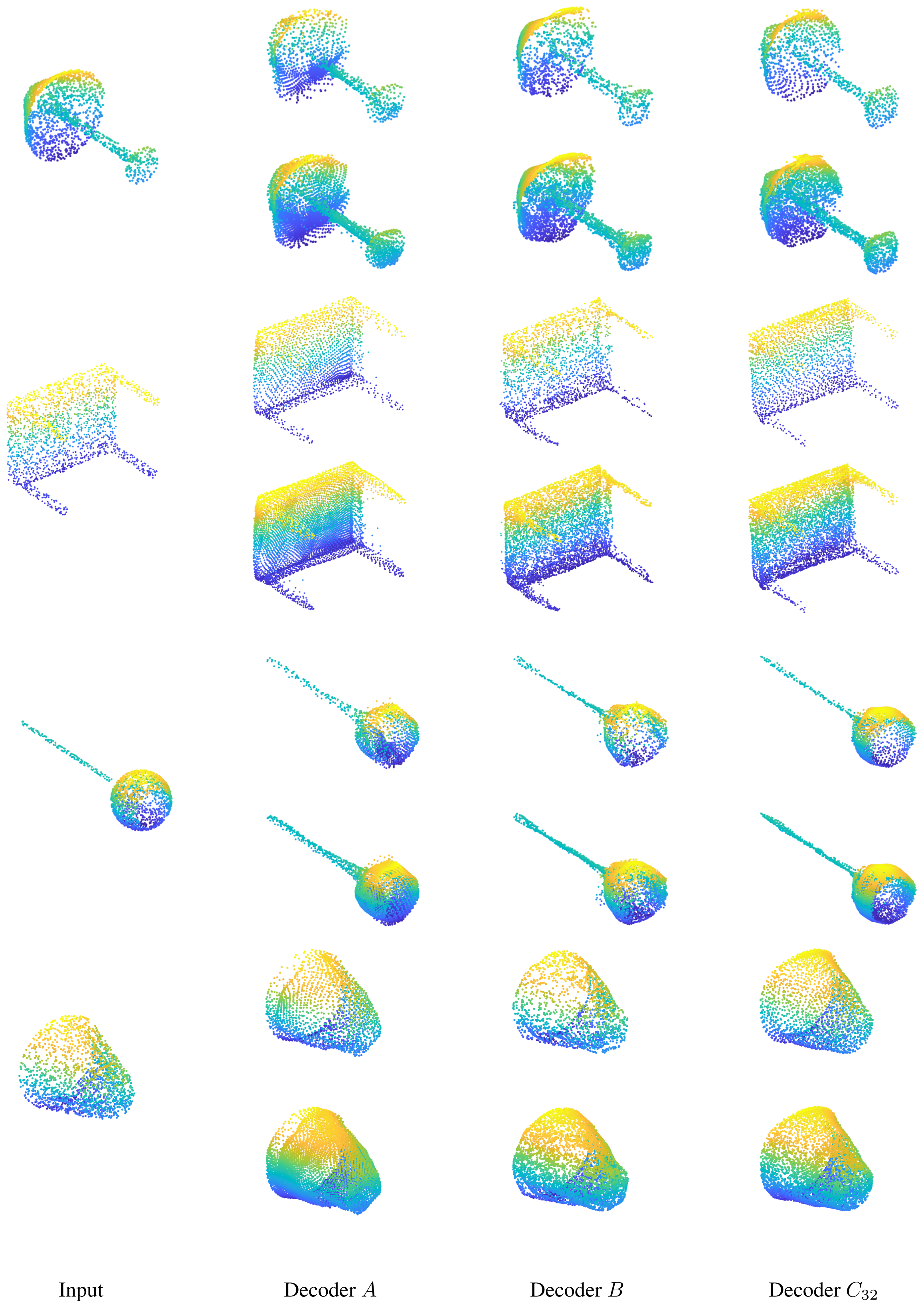

Figure 1. Examples of point cloud reconstruction and upsampling on ShapeNetCorev2 with various decoders for analysis. The images in the first, third, fifth, and seventh lines show the reconstruction results, and the images in the second, fourth, sixth, eighth lines show the results of the upsampling.

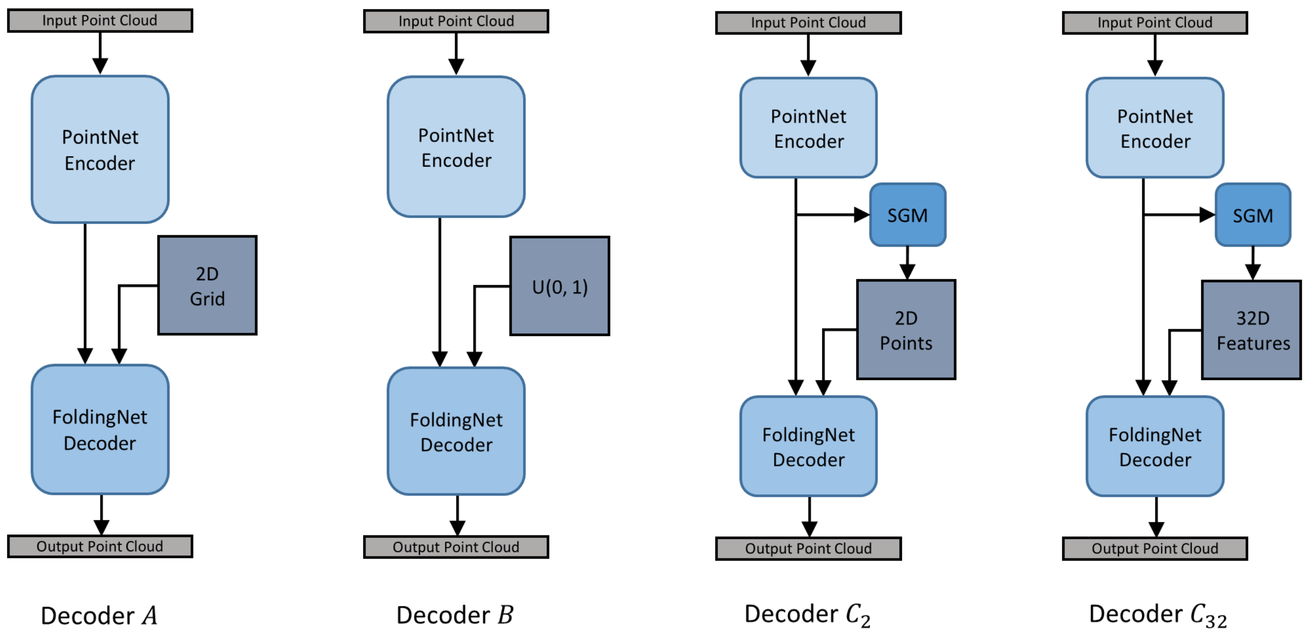

Figure 2. Overall architectures of the trained networks for analysis.

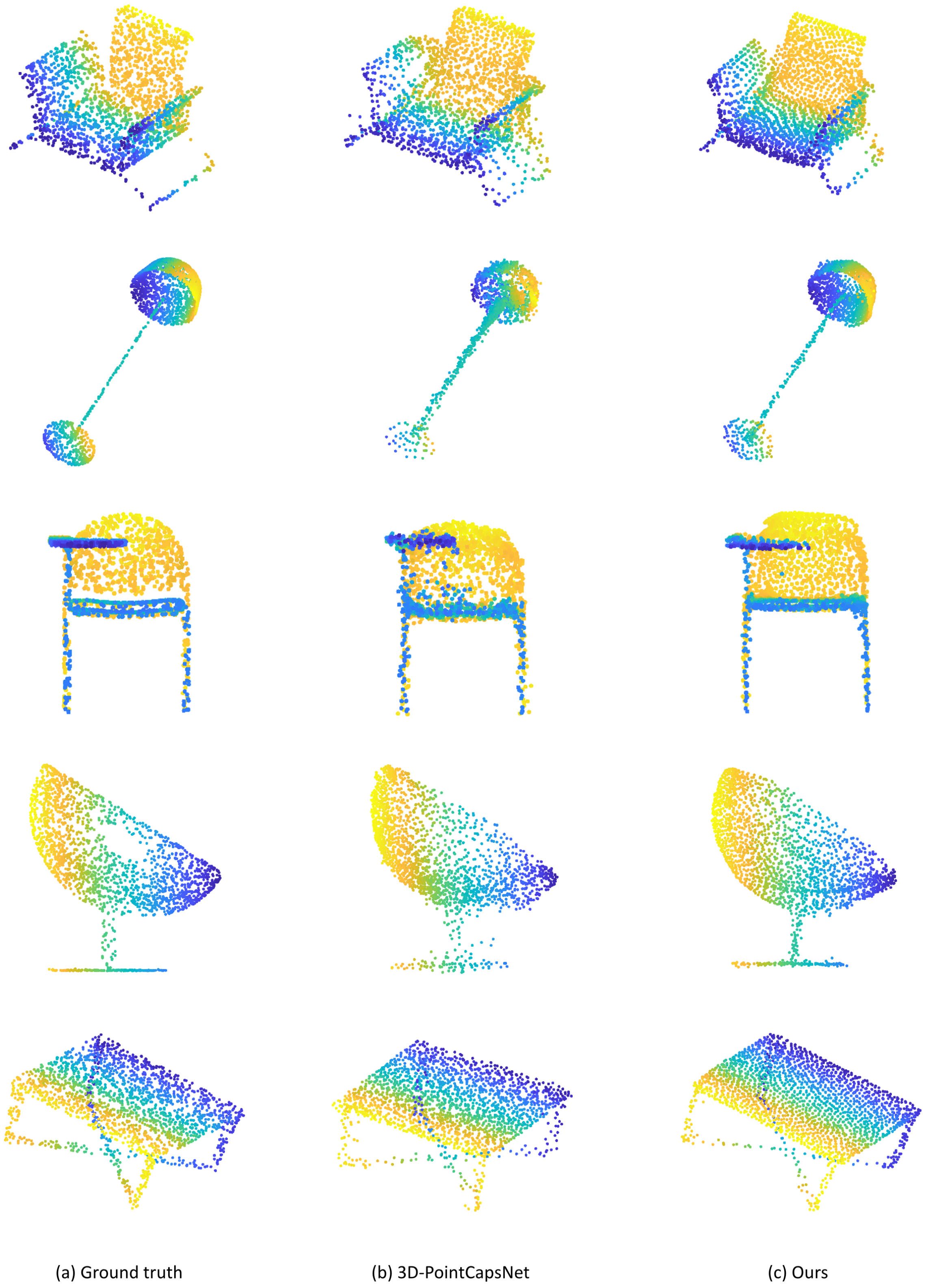

Figure 3. Examples of point cloud reconstruction results on ShapeNetCore13.