# Supplementary

## A. Overview

This supplementary document provides more technical details and experimental results to the main paper. Shape retrieval experiments are demonstrated with ShapeNet Core55 dataset in Sec. B. The time and space complexity is analyzed in Sec. C, followed by detailed illustration of our permutation invariant SOM training algorithms in Sec. D. More experiments and results are presented in Sec. E.

## B. Shape Retrieval

Our object classification network can be easily extended to the task of 3D shape retrieval by regarding the classification score as the feature vector. Given a query shape and a shape library, the similarity between the query and the candidates can be computed as their feature vector distances.

### B.1. Dataset

We perform 3D shape retrieval task using the ShapeNet Core55 dataset, which contains 51,190 shapes from 55 categories and 204 subcategories. Specifically, we adopt the dataset split provided by the 3D Shape Retrieval Contest 2016 (SHREC16), where 70% of the models are used for training, 10% for validation and 20% for testing. Since the 3D shapes are represented by CAD models, i.e., vertices and faces, we sample 5,000 points and surface normal vectors from each CAD model. Data augmentation is identical with the previous classification and segmentation experiments - random jitter and scale.

### B.2. Procedures

We train a classification network on the ShapeNet Core55 dataset using identical configurations as our ModelNet40 classification experiment, i.e. a SOM of size $8\times8$ and $k = 3$. For simplicity, the softmax loss is minimized with only the category labels (without any subcategory information). The classification score vector of length 55 is used as the feature vector. We calculate the L2 feature distance between each shape in the test set and all shapes in the same predicted category from the test set (including itself). The corresponding retrieval list is constructed by sorting these shapes according to the feature distances.

### B.3. Performance

SHREC16 provides several evaluation metrics including Precision-Recall curve, F-score, mean average precision (mAP), normalized discounted cumulative gain (NDCG). These metrics are computed under two contexts - macro and micro. Macro metric is a simple average across all categories while micro metric is a weighted average according to the number of shapes in each category. As shown in Table 3, our SO-Net out-performs state-of-the-art approaches with most metrics. The precision-recall curves are illustrated in Fig. 9, where SO-Net demonstrates the largest area under curve (AUC). Some shape retrieval results are visualized in Fig. 11.

## C. Time and Space Complexity

We evaluate the model size, forward (inference) time and training time of several point cloud based networks in the task of ModelNet40 classification, as shown in Table 4. The forward timings are acquired with a batch size of 8 and input point cloud size of 1024. In the comparison, we choose the networks with the best classification accuracy among various configurations of PointNet and PointNet++, i.e., PointNet with transformations and PointNet++ with multi-scale grouping (MSG). Because of the parallelizability and simplicity of our network design, our model size is smaller and the training speed is significantly faster compared to PointNet and its successor PointNet++. Meanwhile, our inference time is around 1/3 of that of PointNet++.

## D. Permutation Invariant SOM

We apply two methods to ensure that the SOM is invariant to the permutation of the input points - fixed initialization and deterministic training rule.

### D.1. Initialization for SOM Training

In addition to permutation invariance, the initialization should be reasonable so that the SOM training is less prone to local minima. Suboptimal SOM may lead to many isolated nodes outside the coverage of the input points. For simplicity, we use fixed initialization for any point cloud input although there are other initialization approaches that are permutation invariant, *e.g.*, principal component initialization. We generate a set of node coordinates that are uniformly distributed in an unit ball to serve as a reasonable initialization because the input point clouds are in various shapes. Unfortunately, as shown in Fig. 10, isolated nodes are inevitable even with uniform initialization. Isolated nodes may not be associated during the kNN search, and their corresponding node features will be set to zero, i.e. the node features are invalid. Nevertheless, our SO-Net is robust to small amount of invalid nodes as demonstrated in the experiments.

We propose a simple algorithm based on potential field methods to generate the initialization as shown in Algorithm 1. $S = \{s_j \in \mathbb{R}^3, j = 0, \cdots, M-1\}$ represents the SOM nodes and $\eta$ is the learning rate. The key idea is to apply a repulsion force between any pair of nodes, and external forces to attract nodes toward the origin. The parameter $\lambda$ is used to control the weighting between the re-

| Method | Micro | | | | | Macro | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | P@N | R@N | F1@N | mAP | NDCG@N | P@N | R@N | F1@N | mAP | NDCG@N |
| Tatsuma | 0.427 | 0.689 | 0.472 | 0.728 | 0.875 | 0.154 | 0.730 | 0.203 | 0.596 | 0.806 |
| Wang_CCMLT | 0.718 | 0.350 | 0.391 | 0.823 | 0.886 | 0.313 | 0.536 | 0.286 | 0.661 | 0.820 |
| Li_ViewAggregation | 0.508 | **0.868** | 0.582 | 0.829 | 0.904 | 0.147 | **0.813** | 0.201 | 0.711 | 0.846 |
| Bai_GIFT [3] | 0.706 | 0.695 | 0.689 | 0.825 | 0.896 | 0.444 | 0.531 | 0.454 | 0.740 | 0.850 |
| Su_MVCNN [33] | 0.770 | 0.770 | 0.764 | 0.873 | 0.899 | 0.571 | 0.625 | 0.575 | **0.817** | 0.880 |
| Kd-Net [18] | 0.760 | 0.768 | 0.743 | 0.850 | 0.905 | 0.492 | 0.676 | 0.519 | 0.746 | 0.864 |
| O-CNN [36] | 0.778 | 0.782 | 0.776 | **0.875** | 0.905 | - | - | - | - | - |
| Ours | **0.799** | 0.800 | **0.795** | 0.869 | **0.907** | **0.615** | 0.673 | **0.622** | 0.805 | **0.888** |

Table 3. 3D shape retrieval results with SHREC16. Our SO-Net out-performs state-of-the-art deep networks with most metrics.

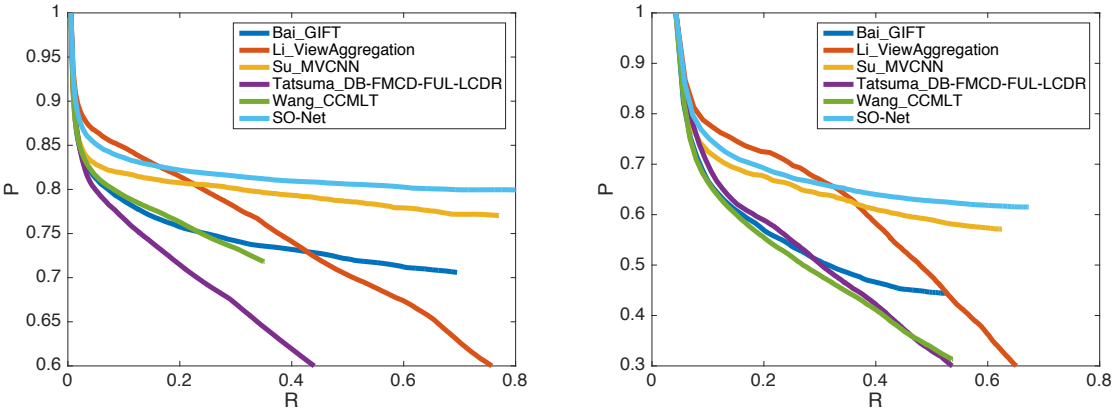


Figure 9. Precision-recall curve for micro (a) and macro (b) metrics in the 3D shape retrieval task. In both curves, the SO-Net demonstrates the largest AUC.

| | Size / MB | Forward / ms | Train / h |
|---|---|---|---|
| PointNet [26] | 40 | **25.3** | 3-6 |
| PointNet++ [28] | 12 | 163.2 | 20 |
| Kd-Net [18] | - | - | 16 |
| Ours | **11.5** | 59.6 | **1.5** |

Table 4. Time and space complexity of point cloud based networks in ModelNet40 classification.

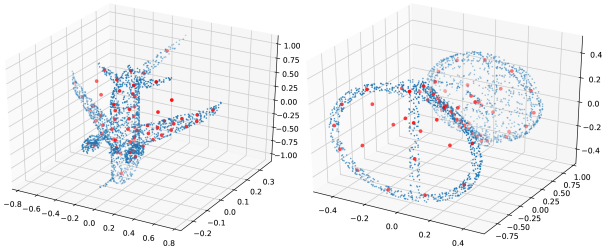

Figure 10. Results of SOM training with uniform initialization. Isolated nodes are inevitable even with uniform initialization.

pulsion and attraction force, so that the resulting nodes are within the unit ball.

**Algorithm 1** Potential field method for SOM initialization

Set random seed.
Random initialization: $S \leftarrow \mathcal{U}(-1, 1)$
**repeat**
  **for all** $s_j \in S$ **do**
    $f_j^{wall} \leftarrow -s_j$
    $f_j^{node} \leftarrow 0$
    **for all** $s_k \in S, k \neq j$ **do**
      $f_j^{node} \leftarrow f_j^{node} + \lambda \frac{s_j - s_k}{\|s_j - s_k\|_2^2}$
    **end for**
  **end for**
  **for all** $s_j \in S$ **do**
    $s_j \leftarrow s_j + \eta(f_j^{wall} + f_j^{node})$
  **end for**
**until** converge

## D.2. Batch Update Training

Instead of updating the SOM once per point, the batch update rule conducts one update after accumulating the effect of all points in the point cloud. As a result, each SOM

update iteration is unrelated to the order of point, i.e., permutation invariant. During SOM training, each training sample affects the winner node and all its neighbors. We define the neighborhood function as a Gaussian distribution as follows:

$$
\begin{aligned}
w_{xy}(x, y|p, q, \sigma_x, \sigma_y) &= \frac{\exp\big(-\frac{1}{2}(x-\mu)^T\Sigma^{-1}(x-\mu)\big)}{\sqrt{(2\pi)^2|\Sigma|}} \\
\mu &= \begin{bmatrix} p & q \end{bmatrix}^T \\
\Sigma &= \begin{bmatrix} \sigma_x^2 & 0 \\ 0 & \sigma_y^2 \end{bmatrix}.
\end{aligned} \tag{6}
$$

The pseudo code of the training scheme is shown in Algorithm 2. $P = \{p_i \in \mathbb{R}^3, i = 0, \cdots, N-1\}$ and $S = \{s_j \in \mathbb{R}^3, j = 0, \cdots, M-1\}$ represent the input points and SOM nodes respectively. The learning rate $\eta_t$ and neighborhood parameter $(\sigma_x, \sigma_y)$ should be decreased slowly during training. In addition, Algorithm 2 can be easily implemented as matrix operations which are highly efficient on GPU.

**Algorithm 2** SOM batch update rule

Initialize $m \times m$ SOM $S$ with Algorithm 1
**for** $t <$ MaxIter **do**
  ▷ Set update vectors to zero
  **for all** $s_{xy} \in S$ **do**
    $D_{xy} \leftarrow 0$
  **end for**
  ▷ Accumulate the effect of all points
  **for all** $p_i \in P$ **do**
    Obtain nearest neighbor coordinate $p, q$
    **for all** $s_{xy} \in S$ **do**
      $w_{xy} \leftarrow$ Eq. (6)
      $D_{xy} \leftarrow D_{xy} + w_{xy}(p_i - s_{xy})$
    **end for**
  **end for**
  ▷ Conduct one update
  **for all** $s_{xy} \in S$ **do**
    $s_{xy} \leftarrow s_{xy} + \eta_t D_{xy}$
  **end for**
  $t \leftarrow t + 1$
  Adjust $\sigma_x$, $\sigma_y$ and $\eta_t$
**end for**

# E. More Experiments

## E.1. MNIST Classification

We evaluate our network using the 2D MNIST dataset, which contains 60,000 $28 \times 28$ images for training and 10,000 images for testing. 2D coordinates are extracted from the non-zero pixels of the images. In order to upsample these 2D coordinates into point clouds of a certain size, *e.g.*, 512 in our experiment, we augment the original pixel coordinates with Gaussian noise $\mathcal{N}(0, 0.01)$. Other than the acquisition of point clouds, the data augmentation is exactly the same as other experiments using ModelNet or ShapeNetPart. We reduce the SOM size to $4 \times 4$ and set $k = 4$ because the point clouds are in 2D and the cloud size is relatively small. The neurons in the shared fully connected layers are reduced as well: 2-64-64-128-128 during point feature extraction and (128+2)-256-512-512-1024 during node feature extraction.

Similar to 3D classification tasks, our network outperforms existing point cloud based deep networks although the best performance is still from the well engineered 2D ConvNets as shown in Table 6. Despite using point cloud representation instead of images, our network demonstrates better results compared with ConvNets such as Network in Network [22], LeNet5 [20].

## E.2. Classification with SOM Only

There are two sources of information utilized by the SO-Net - the point cloud and trained SOM. The information from SOM is explicitly used when the nodes are concatenated with the node features at the beginning of node feature extraction. Additionally, the SOM is implicitly utilized because point normalization, kNN search and the max pooling are based on the nodes. We perform classification using the SOM nodes without the point coordinates of the point cloud to analyze the contribution of the SOM. We feed the SOM nodes into a 3-layer MLP with MNIST, ModelNet10 and ModelNet40 dataset. Similarly in the Kd-Net [18], experiments are conducted using the kd-tree split directions without point information, i.e. feeding directions of the splits into a MLP. The results are shown in Table 5.

It is interesting that we can achieve reasonable performance in the classification tasks by combining SOM and a simple MLP. But there is still a large gap between this variant and the full SO-Net, which suggests that the integration of SOM and point clouds is important. Another intriguing phenomenon is that the SOM based MLP achieves better results than split-based MLP. It suggests that maybe SOM is more expressive than kd-trees in the context of classification.

## E.3. Result Visualization

To visualize the shape retrieval results, we present the top 5 retrieval results for a few shapes as shown in Fig. 11

For the point cloud autoencoder, we present results from two networks. The first network consumes 1024 points and reconstructs 1280 points with the ShapeNetPart dataset (Fig. 12), while the second one consumes 5000 points

| Method | Input | MNIST | ModelNet10 | ModelNet40 |
|---|---|---|---|---|
| Kd-Net split based MLP [18] | splits | 82.40 | 83.4 | 73.2 |
| Kd-Net depth 10 [18] | point | 99.10 | 93.3 | 90.6 |
| Ours - SOM based MLP | SOM nodes | 91.37 | 88.9 | 75.7 |
| Ours | point | 99.56 | 94.5 | 92.3 |

Table 5. Classification results using structure information - SOM nodes and kd-tree split directions.

| Method | Error rate (%) |
|---|---|
| Multi-column DNN [10] | **0.23** |
| Network in Network [22] | 0.47 |
| LeNet5 [20] | 0.80 |
| Multi-layer perceptron [31] | 1.60 |
| PointNet [26] | 0.78 |
| PointNet++ [28] | 0.51 |
| Kd-Net [18] | 0.90 |
| ECC [32] | 0.63 |
| Ours | 0.44 |

Table 6. MNIST classification results.

and reconstructs 4608 points using the ModelNet40 dataset (Fig. 13). We present one instance for each category.

For results of object part segmentation using ShapeNet-Part dataset, we visualize one instance per category in Fig. 14. The inputs to the network are point clouds of size 1024 and the corresponding surface normal vectors.

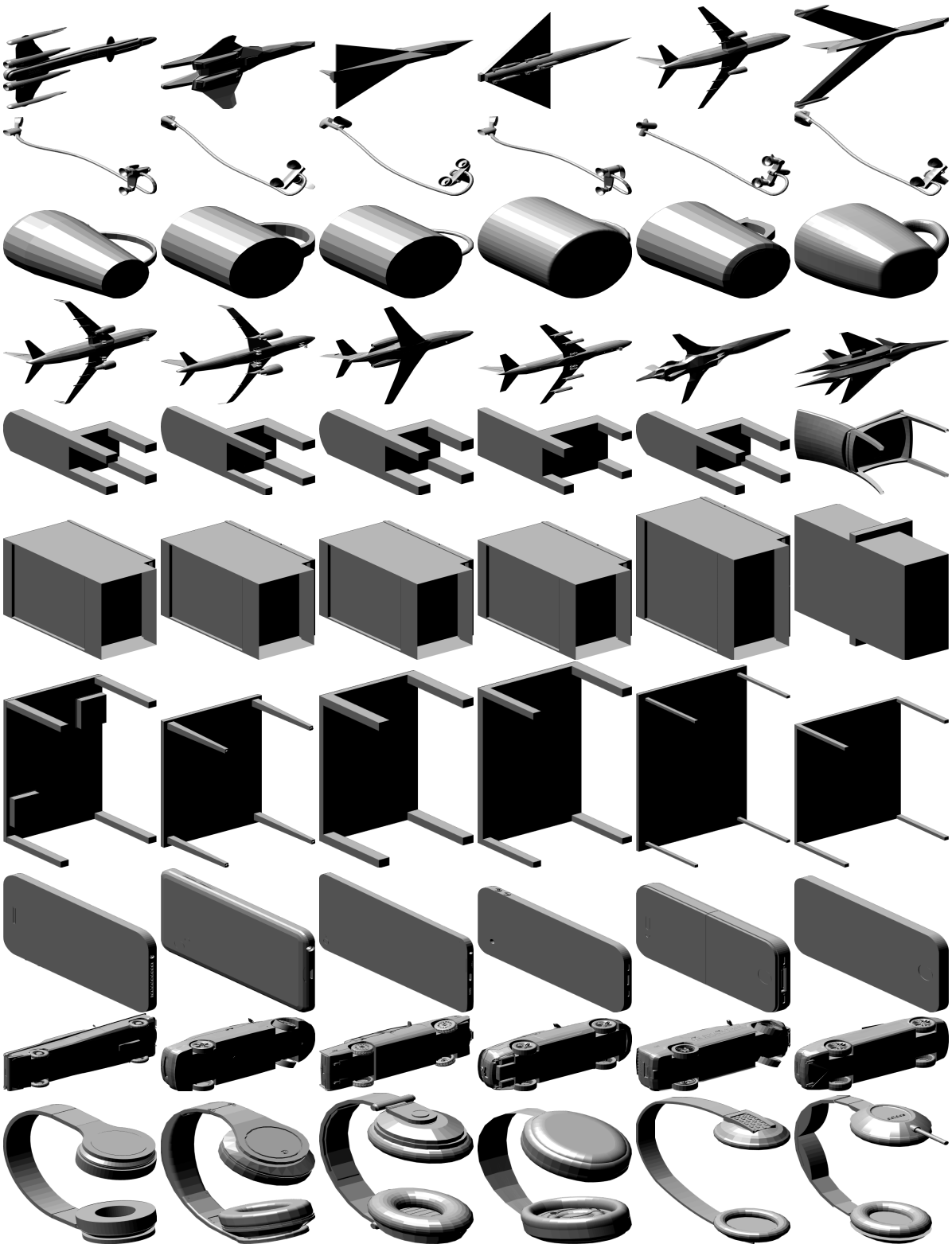

Figure 11. Top 5 retrieval results. First column: query shapes. Column 2-6: retrieved shapes ordered by feature similarity.

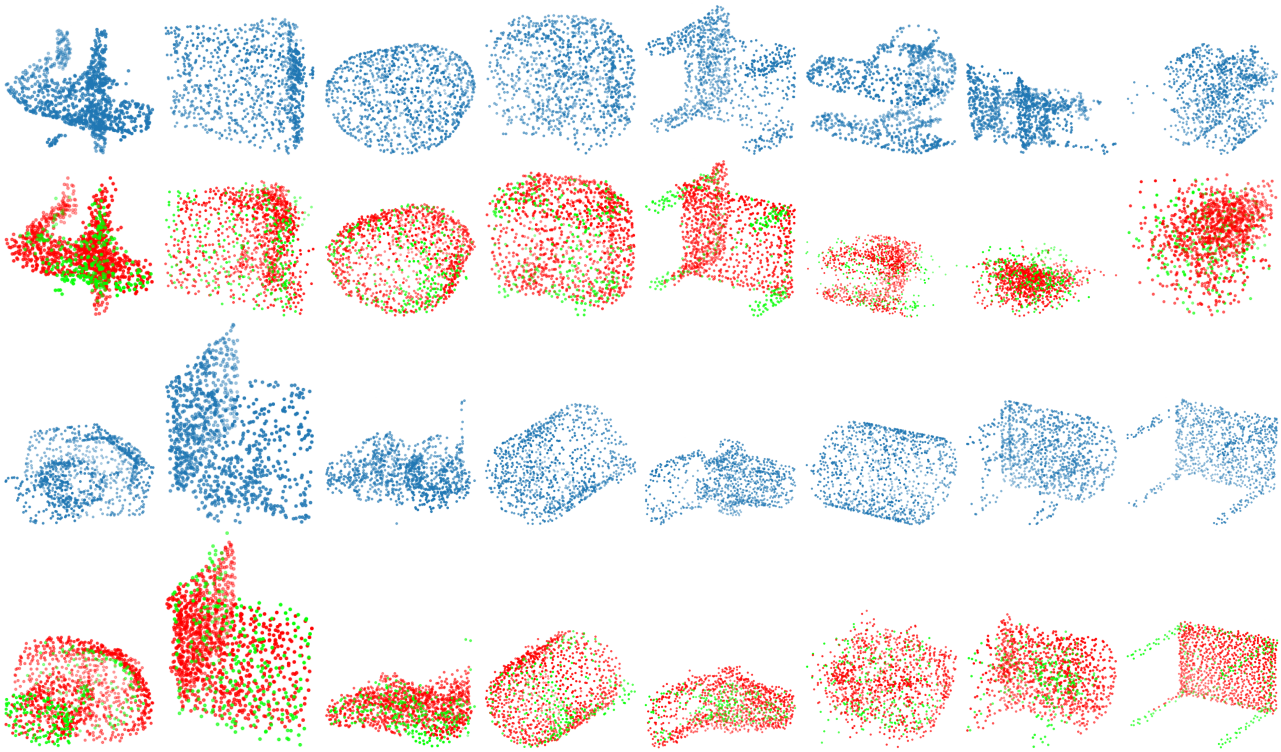

Figure 12. Results of our ShapeNetPart autoencoder. Red points are recovered by the convolution branch and green ones are by the fully connected branch. Odd rows: input point clouds. Even rows: reconstructed point clouds.

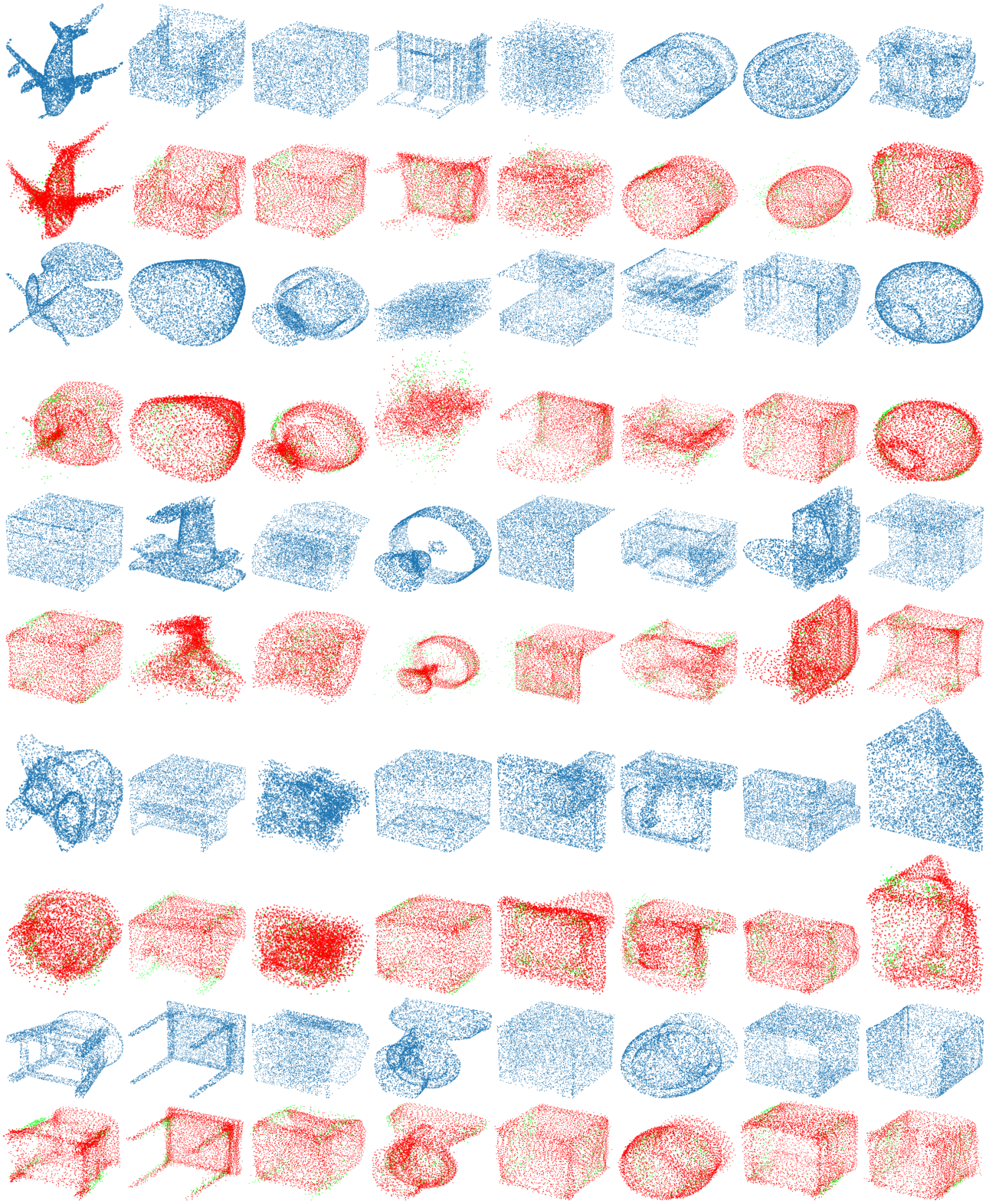

Figure 13. Results of our ModelNet40 autoencoder. Red points are recovered by the convolution branch and green ones are by the fully connected branch. Odd rows: input point clouds. Even rows: reconstructed point clouds.

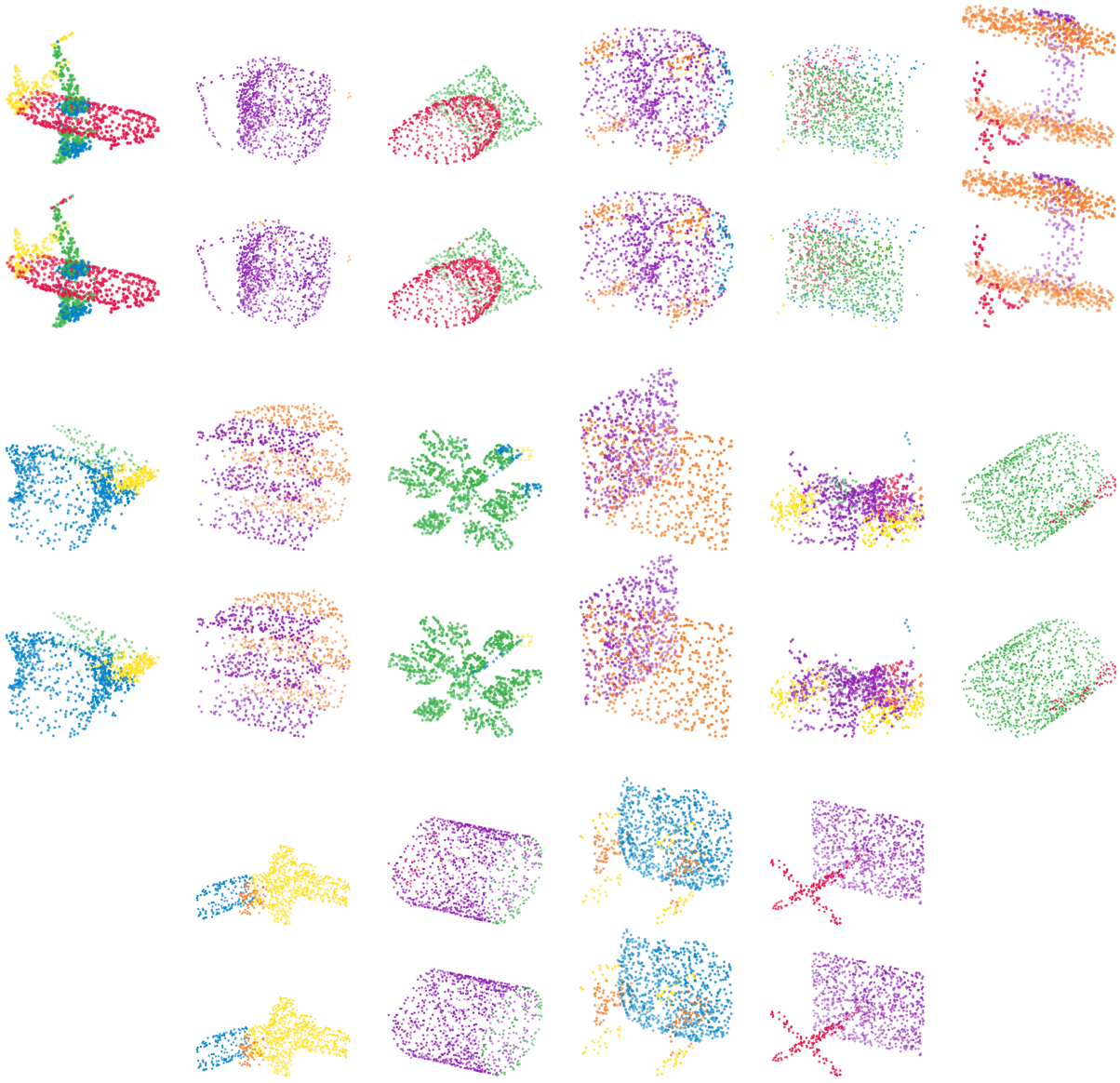

Figure 14. Results of object part segmentation. Odd rows: ground truth segmentation. Even rows: predicted segmentation.