# Supplementary Material

## 1. Implementation Details: Classification

We train the models on the standard 9,843 training set and evaluate on the 2,468 test set. Following the practice in [1], we add small random perturbations and rotations on both the input point clouds and the vertices on the highest fractal level. To project n points of size $n \times 3$ to $N$ vertex features of size $N \times C$, we firstly find the $k$-nearest points for each vertex according to the angles between vertex and points to get $N$ groups of neighboring points of size $k \times 3$ ($N \times k \times 3$ in total). Then, we feed each group of neighboring points of size $k \times 3$ to a small PointNet model described in our paper to obtain the feature of length $C$ for each vertex. The projection model consists of 3 fully-connected layers, where a Non-Local layer [2] is used before the last layer and we apply maxpool operation on the outputs of these 3 layers to summarize the features of $k$ sampled points for each vertex. Note that we only apply the Non-Local operation on the $k$ points that are assigned to the same vertex, and keep the features of different vertices independent. We use the global max pooling operation to summarize the output features of each stage, and concatenate these summarized features to form the global representation of the input point cloud. There are 3 fully-connected layers in the classifier, and we randomly dropout features followed by the last layer with 80% probability. For models that are trained with the invertibility constraint, we further add a module $f$ to map the features on lattices to the input points. Specifically, we map feature on each vertex to 4 3D points using the idea of [3], where we concatenate a 2D coordinate for each feature to generate 4 different points from the same feature (in our case, $(-\alpha, -\alpha)$, $(-\alpha, \alpha)$, $(\alpha, -\alpha)$ and $(\alpha, \alpha)$ are used and we set $\alpha$ to 0.3). Then, the reconstruction error of input points $X$ and the reconstructed points $\hat{X} = \bigcup_i f(F_i)$ can be computed following:

$$d_{CH}(X, \hat{X}) = \max\{\frac{1}{|X|}\sum_{x \in X} \min_{\hat{x} in \hat{X}} ||x - \hat{x}||_2, \\ \frac{1}{|\hat{X}|}\sum_{\hat{x} \in \hat{X}} \min_{x \in X} ||x - \hat{x}||_2\}. \quad (1)$$

The detailed architectures of projection modules and $f$ are summarized in Table 1 and Table 2 respectively.

Table 1. The detailed architecture of projection module. $N$ is the number of points and $c$ the dimension of original feature of each point (for example, $c = 3$ means xyz input).

| input size | output size | layer type |
|---|---|---|
| $N \times c$ | $N_4 \times k \times c$ | assign points |
| $N_4 \times k \times c$ | $N_4 \times k \times 8K$ | fully-connected |
| $N_4 \times k \times 8K$ | $N_4 \times k \times 8K$ | fully-connected |
| $N_4 \times k \times 8K$ | $N_4 \times k \times 8K$ | Non-Local |
| $N_4 \times k \times 8K$ | $N_4 \times k \times 16K$ | fully-connected |
| $N_4 \times k \times 16K$ | $N_4 \times 16K$ | maxpool |

Table 2. The detailed architecture of $f$.

| input size | output size | layer type |
|---|---|---|
| $N_4 \times 4 \times (16K + 2)$ | $N_4 \times 4 \times 8K$ | fully-connected |
| $N_4 \times 4 \times 8K$ | $N_4 \times 4 \times 4K$ | fully-connected |
| $N_4 \times 4 \times 4K$ | $N_4 \times 4 \times 3$ | fully-connected |

## 2. Implementation Details: Retrieval

We train the models for 3D shape retrieval with arbitrary rotations to improve the performance when perturbed 3D point clouds are used for evaluation. During training, we randomly sample $b/4$ point clouds from the training set and add 3 other positive examples for each of these point clouds to form a mini-batch, where $b$ is the size of mini-batch. We perform the online hard example mining for both positive examples and negative examples. More specifically, for each sample $x$, we find the the positive example $x_p$ that has the lowest similarity with $x$ and the negative example $x_n$ that has the highest similarity with $x$ in the mini-batch. Then, the triplet loss can be defined as:

$$L_{triplet} = \max(s(x, x_n) - s(x, x_p) + \beta, 0), \quad (2)$$

where $s(x, y)$ is the cosine similarity between $x$ and $y$, and $\beta = 0.3$ is the margin value.

## 3. Implementation Details: Part Segmentation

For part segmentation task, an encoder-decoder network is used to predict per-point labels. We use the classification network as the encoder network. For the decoder net-

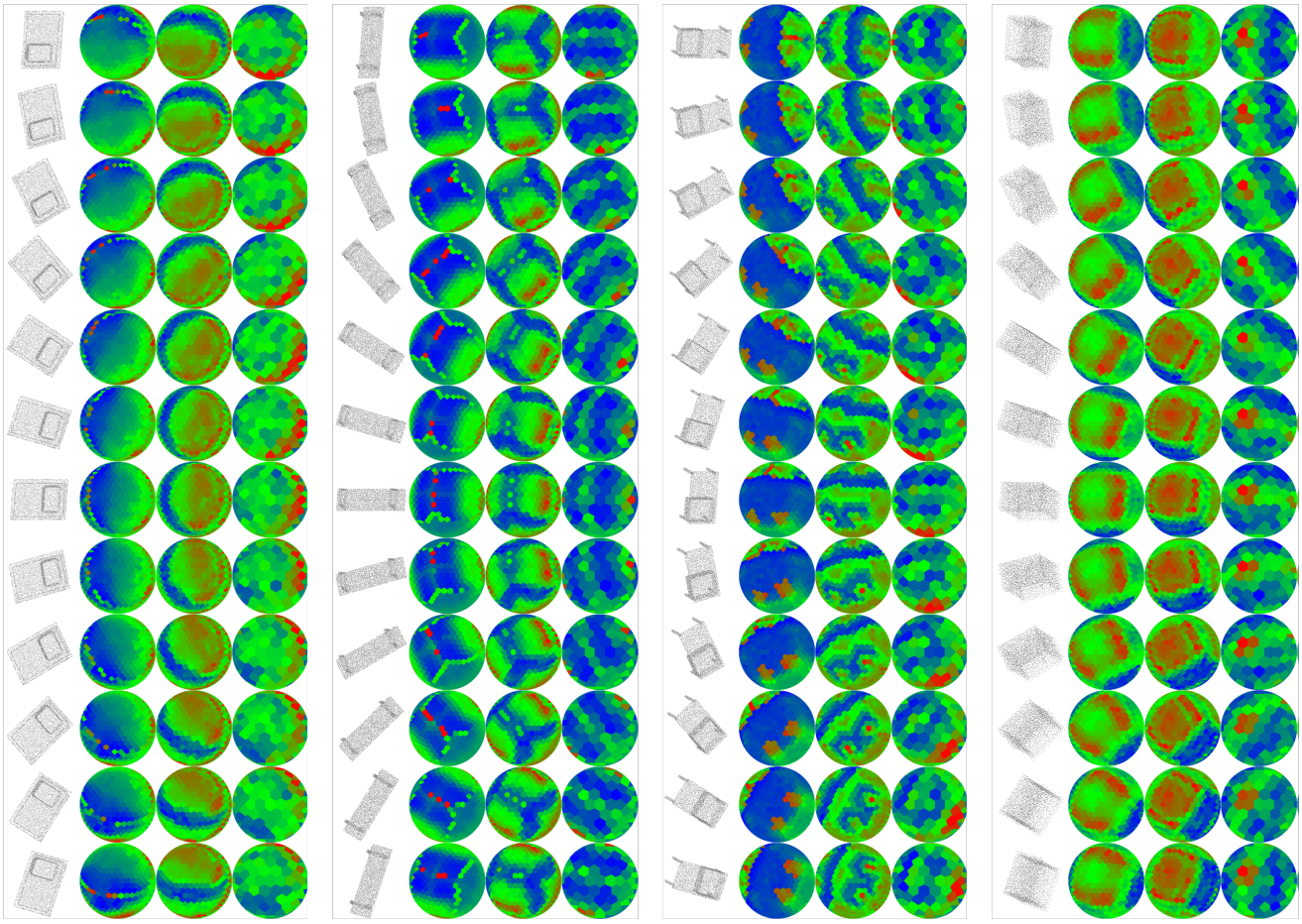

Figure 1. Visualization of the input point clouds and the features from different hierarchies. The 4 columns of each group represent input point clouds, outputs of the projection module, low-level features and high-level features respectively. Best viewed in color.

work, one symmetry convolution block is added after the up-sample layer for each fractal level. For each points, we concatenate 3D coordinate with features from nearest vertex of different fractal levels to form the final feature of each point. Then, a 3-layer MLP (512, 256, C) is used to obtain the final predictions. Similar to classification task, we add a dropout layer before the last fully-connected layer. Other details are similar with [1].

## 4. Visualization

The visualization of input point clouds, projection features, low-level features and high-level features with different rotations is presented in Figure 1. We can see that the activations from different hierarchies are all rotation invariant, thus our model is resistant to arbitrary rotations on input point clouds.

## References


[1] Charles Ruizhongtai Qi, Li Yi, Hao Su, and Leonidas J Guibas. Pointnet++: Deep hierarchical feature learning on point sets in a metric space. In *Advances in Neural Information Processing Systems*, pages 5099–5108, 2017. 1, 2

[2] Xiaolong Wang, Ross Girshick, Abhinav Gupta, and Kaiming He. Non-local neural networks. *arXiv preprint arXiv:1711.07971*, 10, 2017. 1

[3] Yaoqing Yang, Chen Feng, Yiru Shen, and Dong Tian. Foldingnet: Point cloud auto-encoder via deep grid deformation. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, volume 3, 2018. 1