Learning Point Embeddings from Shape Repositories for Few-Shot Segmentation Supplementary Material

Gopal Sharma Evangelos Kalogerakis Subhransu Maji
University of Massachusetts, Amherst
{gopalsharma,kalo,smaji}@cs.umass.edu

1. Dataset

Our dataset is a subset of ShapeNetCore where we focus on 16 categories from ShapeNet part segmentation dataset. Note that the semantic segmentation dataset contains $16.6k$ shapes of these categories compared to $28k$ in the ShapeNet core. We first start by downloading collada file for shapes in ShapenetCore dataset from the 3D Warehouse website, but constraining to 16 categories mentioned above. Samples from the dataset are shown in the Figure 2 (left). The Collada format stores the meshes in hierarchical structure, starting from the root node, recursively applying transformation until the leaf nodes that correspond to different parts of the 3D shape.

Note that we only use a small number of the segmentation labels provided in the ShapeNet segmentation benchmark for training in our few-shot segmentation experiments. We also make sure that the there is no overlap between any of our training set (embedding training, tag training, semantic segmentation training) and the evaluation set.

Generating segments from meshes. The number of segments in meshes from Collada files can vary from 1-4000. These range from ones where all the parts are grouped together to others where parts are vastly over segmented. A possible way to control the number of segments is to select the depth of the tree that gives reasonable number of segments. Lower level in the hierarchy gives smaller number of segments as shown in Figure-3 (main paper). We select the depth of the tree such that the number of segments are at least k, where k is the number of semantic parts present in the semantic part-segmentation dataset for that category. This is done to avoid favoring cases where semantically different parts are merged. We further, select the depth of the tree such that maximum number of segments is less than 500 to avoid large over-segmentation of shape and to keep high ratio of number of points vs number of segments. Figure 1 shows the distribution of segments in our pruned dataset.

These meshes have inconsistent orientation, thus we preprocess these meshes to align in a canonical orientation of the Shapenent core dataset. The alignment is done by first sampling points from source and target meshes, then rotating the source point cloud along all the three-axis by from 0 to 180 degrees at the interval of 30 degrees and finally by selecting the orientation which gives least Chamfer distance between the source and the target shape points. The coarse search is sufficient to align most models. We preprocess the meshes by uniformly sampling 10k points from the surface using stratified sampling where sampling is weighted by the area of the segment, i.e. we sample more points from the segments with larger surface area in comparison to segments with smaller surface area.

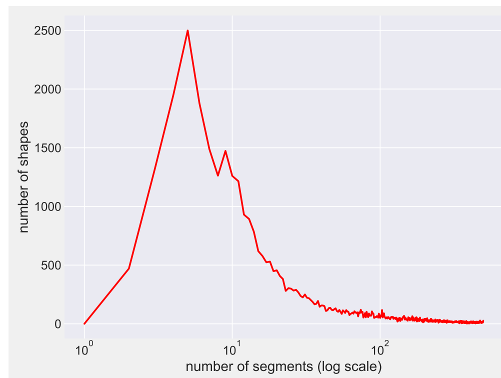

Figure 1: **Distribution of number of segments.**

2. Network Architectures

The details about our point embedding network used for various experiments are shown in Table 1. The PEN is a variant of PointNet that produces a per-point embedding. For classification tasks (segmentation, tag prediction) we add two addtional layers to predict labels.

3. Visualization of Semantic Segmentation

Figure 3 compares the segmentation models pretrained with Hierarchy meta data, trained from scratch, and autoencoder pretrained for training size 4 and 8.

Figure 2: **Visualization of the meta data.** (Left) Parts of various objects shown in different colors. Notice that segmentations vary in their number and granularity across instances. (Right) A word cloud of the raw tags collected from the dataset. The font size is proportional to the square root of frequency in the dataset.

(a) PEN Hierarcy

	Layers	Output
1	Input shape	$3 \times N$
2	Relu(FC(1, 64))	$64 \times N$
3	Relu(FC(64, 128))	$128 \times N$
4	Relu(FC(128, 512))	$512 \times N$
5	Relu(FC(512, 1024))	$1024 \times N$
6	Max-pool(1xN)	1024×1
7	Concat(2, 6)	$1088 \times N$
8	Relu(FC(1088, 512))	$512 \times N$
9	Relu(FC(512, 256))	$256 \times N$
10	Relu(FC(256, 128))	$128 \times N$
11	FC(128, 64)	$64 \times N$

(b) PEN Segmentation

	Layers	Output
1	Input shape	$3 \times N$
2	Relu(FC(1, 64))	$64 \times N$
3	Relu(FC(64, 128))	$128 \times N$
4	Relu(FC(128, 512))	$512 \times N$
5	Relu(FC(512, 1024))	$1024 \times N$
6	Max-pool(1xN)	1024×1
7	Concat(2, 6)	$1088 \times N$
8	Relu(FC(1088, 512))	$512 \times N$
9	Relu(FC(512, 256))	$256 \times N$
10	Relu(FC(256, 128))	$128 \times N$
11	Relu(FC(128, 64))	$64 \times N$
12	Relu(FC(64, 64))	$64 \times N$
13	Softmax(FC(64, C))	$C \times N$

(c) PEN Tags

	Layers	Output
1	Input shape	$3 \times N$
2	Relu(FC(1, 64))	$64 \times N$
3	Relu(FC(64, 128))	$128 \times N$
4	Relu(FC(128, 512))	$512 \times N$
5	Relu(FC(512, 1024))	$1024 \times N$
6	Max-pool(1xN)	1024×1
7	Concat(2, 6)	$1088 \times N$
8	Relu(FC(1088, 512))	$512 \times N$
9	Relu(FC(512, 256))	$256 \times N$
10	Relu(FC(256, 128))	$128 \times N$
11	Relu(FC(128, 64))	$64 \times N$
12	Relu(FC(64, 64))	$64 \times N$
13	Sigmoid(FC(64, T))	$T \times N$

Table 1: **Architecture details.** Network architecture for (a) point embeddings trained with hierachy data, (b) semantic segmentation, and (c) tag prediction. The difference between (b) and (c) is that the latter is trained on noisy tag data, which we do not know to be mutually exclusive and sparse. This motivates training using per-tag binary classification (sigmoid vs. softmax for the semantic segmentation task on ShapeNet). For transfer learning of *Hierarchy* in (b), first 11 layers are initialized using (a). Tag training in (c) can be done either with no initialization for *Tags* or with initializing first 11 layers using (a) for *Hierarchy+Tags*. For transfer learning in case of *Tags* or *Hierarchy+Tags* to predict semantic labels, pre-trained network with tag supervision in (c) is used where the last layer of (c) is replaced by Softmax(FC(64, C)). ReLU denotes $\max(0, x)$, FC: Fully Connected layer, Max-pool computes dimensionwise maximum across all points, Concat(i,j) concatenates the ouputs of layer i and j, C: number of semantic classes and T: number of tags.

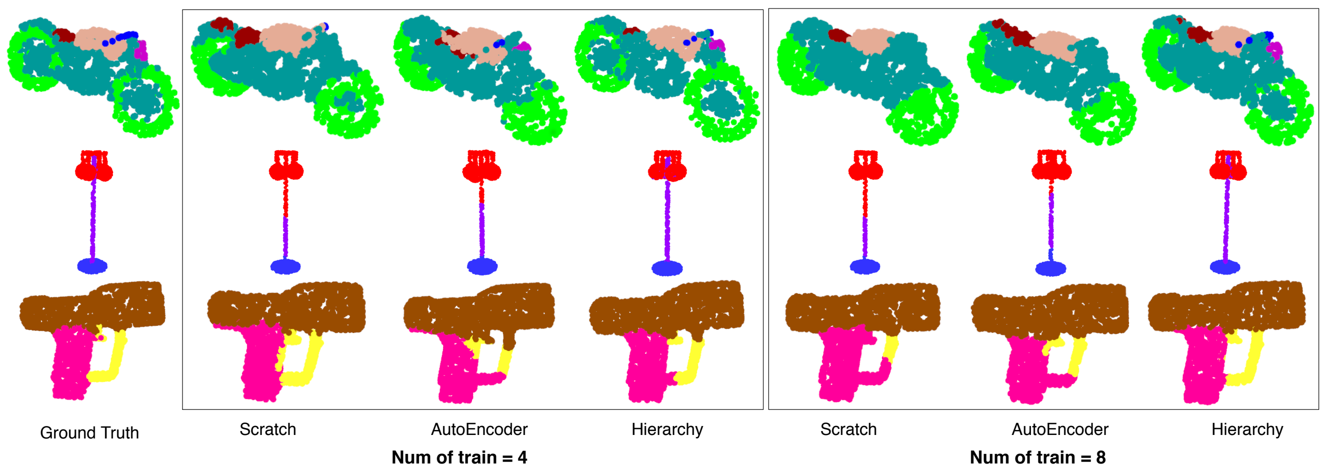

Figure 3: **Segmentation results.** Visualization of segmentations produced by various models (scratch, autoencoder, hierarchy) when the number of training shapes is 4 (Left) and 8 (Right). The boundaries between parts are better delinated (as seen in the ground truth) by the models trained on hierarchy meta data.