

BAE-NET: Branched Autoencoder for Shape Co-Segmentation - Supplementary Material

Please find the detailed **network structures and parameters** at the end of this document.

1. Visualization of neuron activations

We show in Figure 1 2 visualization of neuron activations in the first, second and third layer of our 3-layer network. Since L1 and L2 have hundreds of neurons, we randomly select a few to show here. Trivial pure-color images are omitted.

2. Unsupervised segmentation results

We show in Figure 3 4 additional unsupervised segmentation results on ShapeNet Part dataset and four ShapeNet categories which are not included in ShapeNet Part dataset. We show in Figure 5 additional unsupervised segmentation results on a joint set of chairs and tables from ShapeNet Part dataset.

3. Weakly-supervised segmentation results

We show in Figure 6 additional results of weakly-supervised segmentation on dataset of Tags2parts.

4. One-shot training vs supervised methods

We provide in Table 1 the detailed results of the comparison experiment shown in section 4.4 of the paper. We use the original codes provided by the authors of the supervised methods on Github. To obtain the results, we use their default network parameters and train their networks on each data category for two hundred epochs.

5. One-shot segmentation results

We show in Figure 7 8 9 additional results of our one-shot segmentation with 1, 2 and 3 segmented exemplar(s).

	Ours (one-shot training)			PointNet				PointNet++				pointCNN				SSCN			
	1-exem.	2-exem.	3-exem.	5%	10%	20%	30%	5%	10%	20%	30%	5%	10%	20%	30%	5%	10%	20%	30%
plane	71.2	72.9	74.6	70.6	76.1	79.8	81.0	72.2	76.4	80.2	81.2	71.7	73.6	77.0	78.9	68.1	74.2	76.9	77.8
bag	73.9	81.8	75.2	69.7	69.8	64.4	64.5	38.3	43.4	57.0	67.6	44.8	44.6	44.7	45.4	44.8	50.9	62.1	60.5
cap	81.6	84.7	82.0	0.2	62.6	56.9	64.8	77.6	77.8	74.8	76.9	36.6	36.5	13.6	15.5	37.9	46.2	69.6	75.3
chair	85.6	85.0	85.1	83.7	86.0	86.0	86.0	86.1	87.5	89.2	89.3	83.4	86.1	86.3	88.0	81.8	84.5	86.2	87.0
earph.	62.6	64.1	76.1	53.5	62.1	55.8	56.4	51.0	67.7	63.2	59.3	34.9	35.1	35.1	15.2	34.6	58.6	61.0	51.7
guitar	86.4	88.2	88.3	86.3	86.2	88.5	89.1	22.6	87.4	88.6	89.2	87.2	87.0	89.5	90.1	84.0	86.2	87.9	88.7
knife	82.1	81.3	82.4	77.2	79.7	79.2	81.3	28.4	18.2	79.6	81.6	24.9	80.6	83.8	84.4	67.7	76.0	80.4	80.9
lamp	64.0	62.9	66.4	71.9	73.6	75.1	77.5	65.2	71.4	75.7	79.0	70.0	75.9	77.4	76.4	52.6	59.6	59.1	64.5
laptop	94.5	94.4	94.3	92.0	93.3	93.8	94.2	23.0	94.1	94.8	95.4	65.4	94.6	95.0	94.8	56.7	53.6	65.6	64.5
motor.	52.8	59.4	63.7	59.6	59.1	61.0	62.7	57.5	61.3	58.9	63.3	16.9	16.9	57.4	64.8	24.1	25.3	27.0	29.1
mug	94.7	94.1	94.6	77.8	83.4	90.2	91.5	91.1	90.4	91.4	89.8	48.7	48.6	92.2	93.5	46.0	46.0	46.8	76.1
pistol	73.8	76.4	77.3	71.0	75.9	78.7	74.5	70.8	72.8	75.3	75.8	23.8	52.9	79.5	80.2	31.7	42.7	45.5	52.9
rocket	39.8	48.2	47.2	41.4	41.8	47.8	49.1	40.4	51.4	54.3	51.3	22.8	22.7	22.7	24.4	27.3	25.1	30.6	39.8
skate.	70.4	72.7	72.8	56.4	57.7	66.9	65.1	70.5	68.7	70.9	70.7	43.8	43.8	55.6	58.8	26.1	44.3	37.1	41.0
table	78.1	80.5	80.7	72.5	74.8	71.2	80.8	74.1	75.3	77.8	78.1	62.5	71.2	74.2	75.4	76.1	77.0	78.8	79.7
Mean	74.1	76.4	77.4	65.6	72.1	73.0	74.6	57.9	69.6	75.4	76.6	49.2	58.0	65.6	65.7	50.6	56.7	61.0	64.6

Table 1. Our one-shot training result vs. supervised methods. To visualize the comparison between our results and the results of supervised methods, we highlight those IOUs that are higher than our 3-exemplar results with yellow color. The IOU's which are lower than or equal to our 3-exemplar results are shown in blue.

Input image

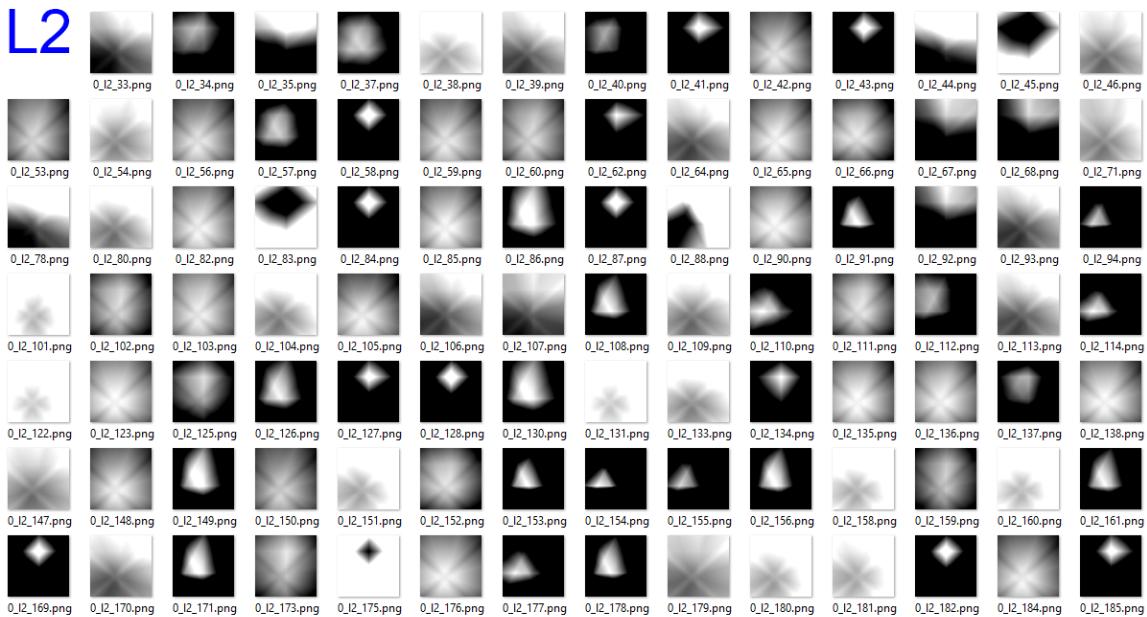


0_I.png

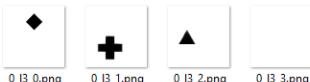
L1



L2



L3



0_I3_0.png

0_I3_1.png

0_I3_2.png

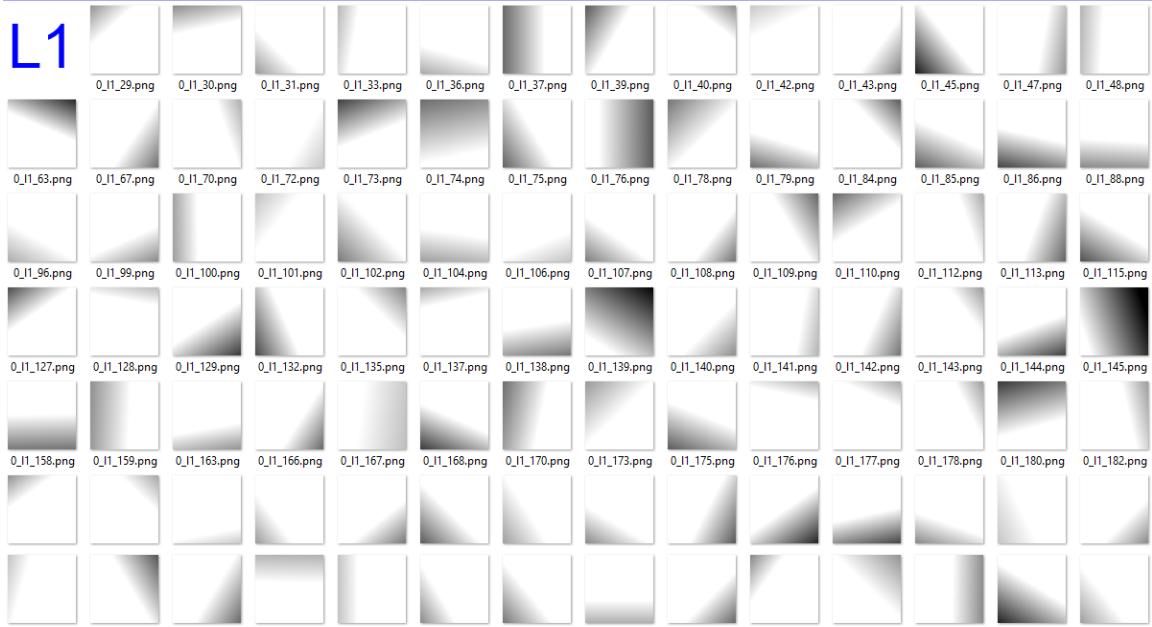
0_I3_3.png

Figure 1. Neuron activations for the model trained on “elements”.

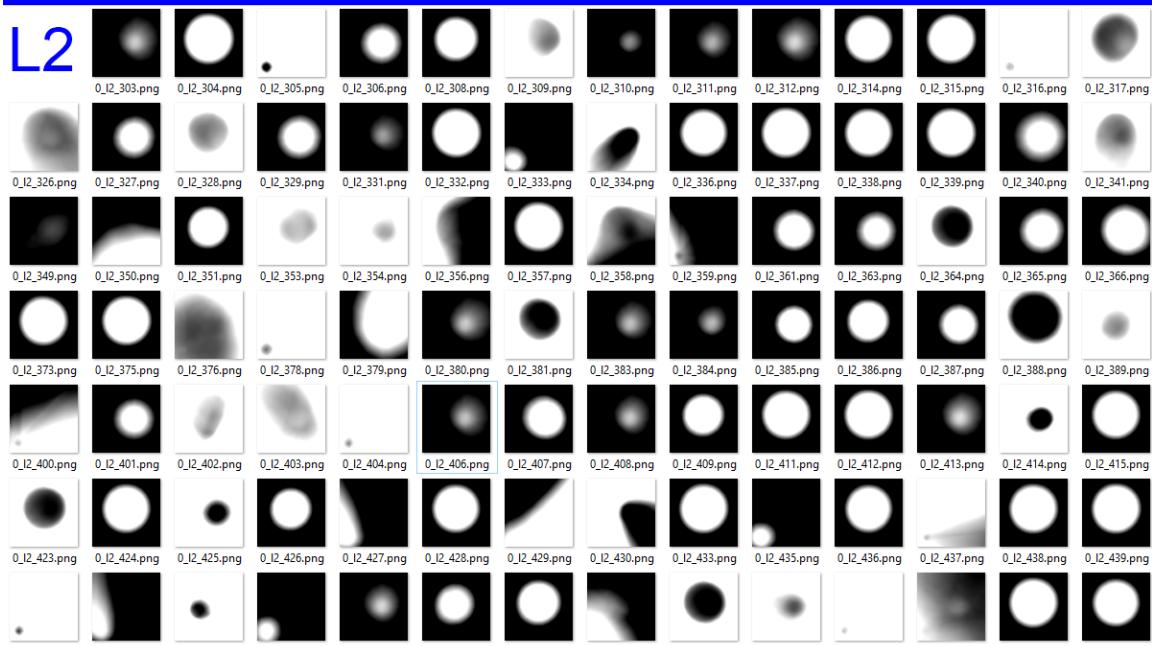
Input image



L1



L2



L3

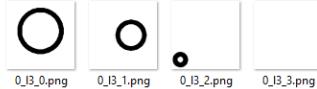


Figure 2. Neuron activations for the model trained on “triple rings”.

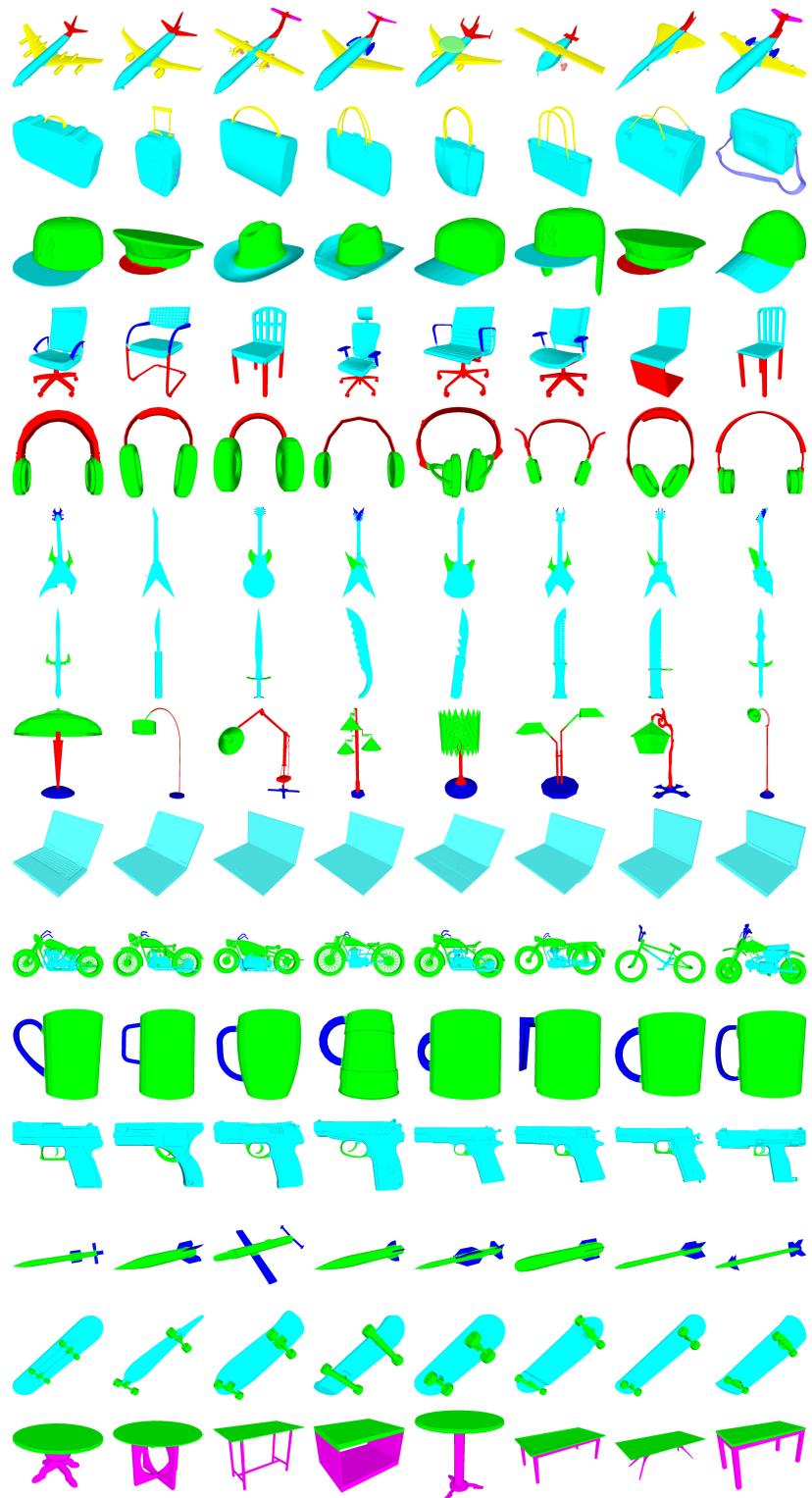


Figure 3. Unsupervised segmentation results on ShapeNet Part dataset.

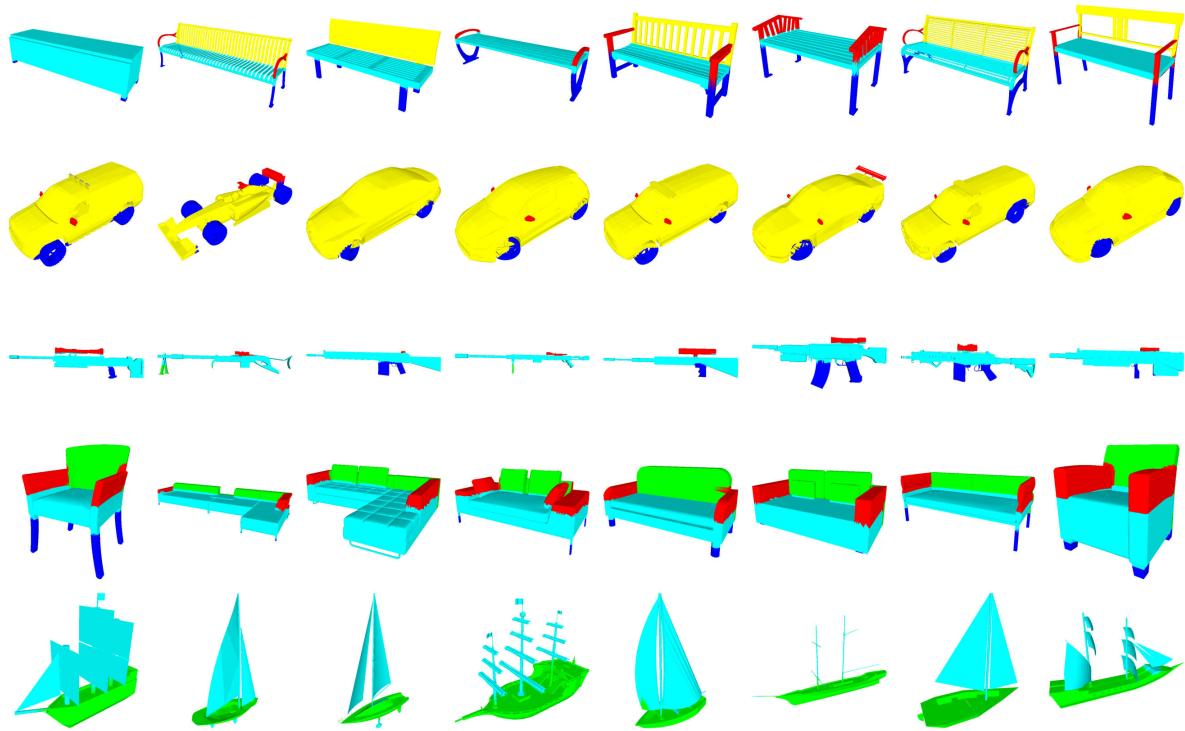


Figure 4. Unsupervised segmentation results on five ShapeNet categories which are not included in Figure 3.



Figure 5. Unsupervised segmentation results on a joint set of chairs and tables from ShapeNet Part dataset.

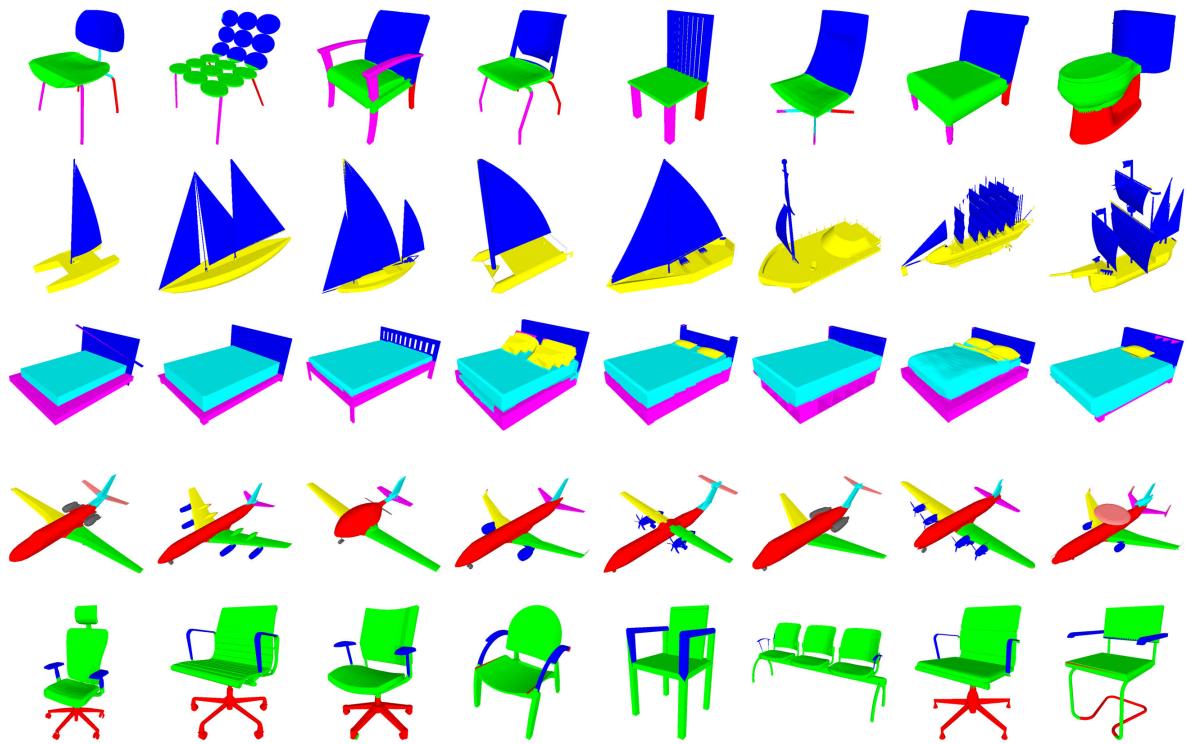


Figure 6. Weakly-supervised segmentation results on the dataset of Tags2parts.

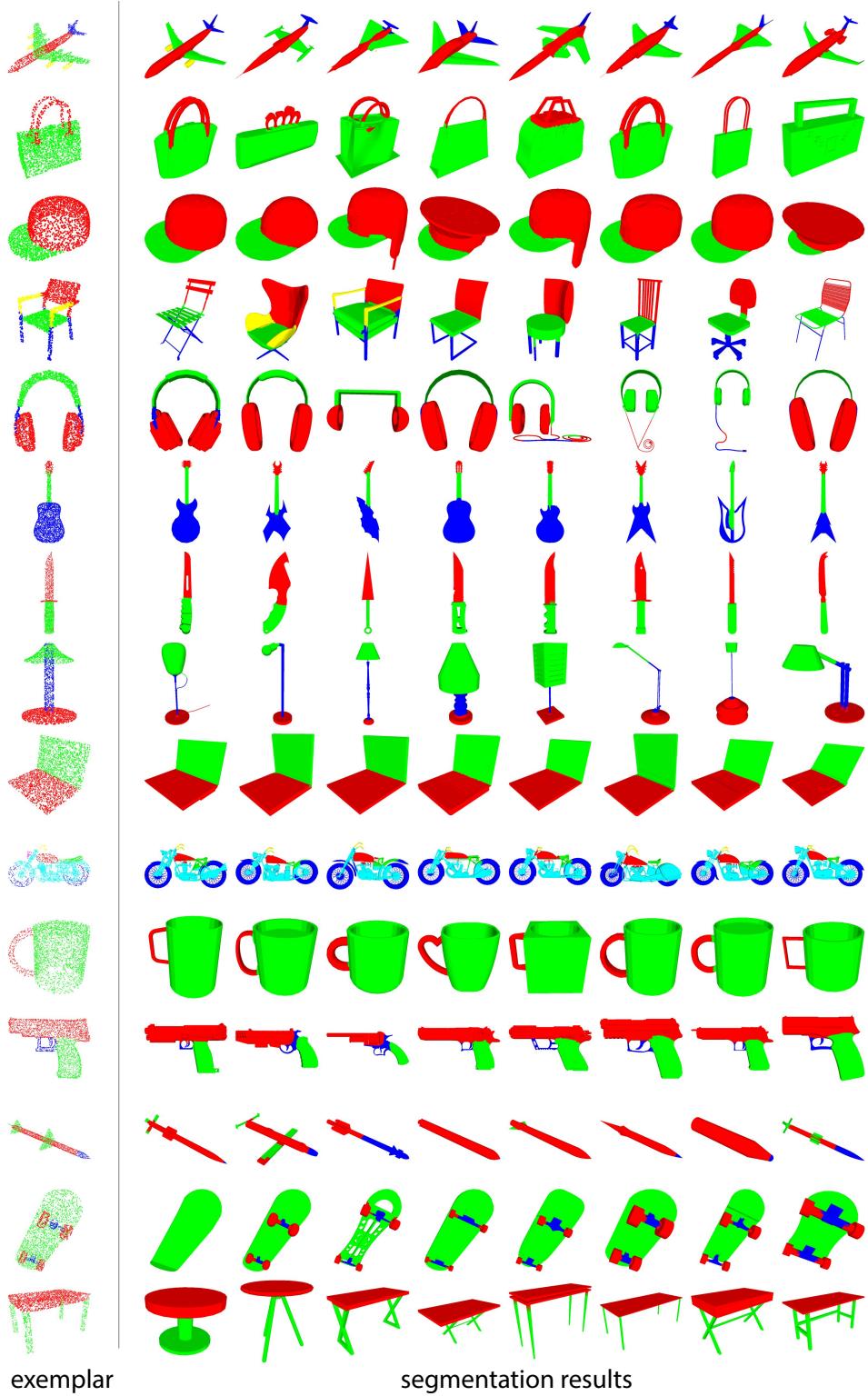


Figure 7. One-shot segmentation results by BAE-NET, with one segmented exemplar.

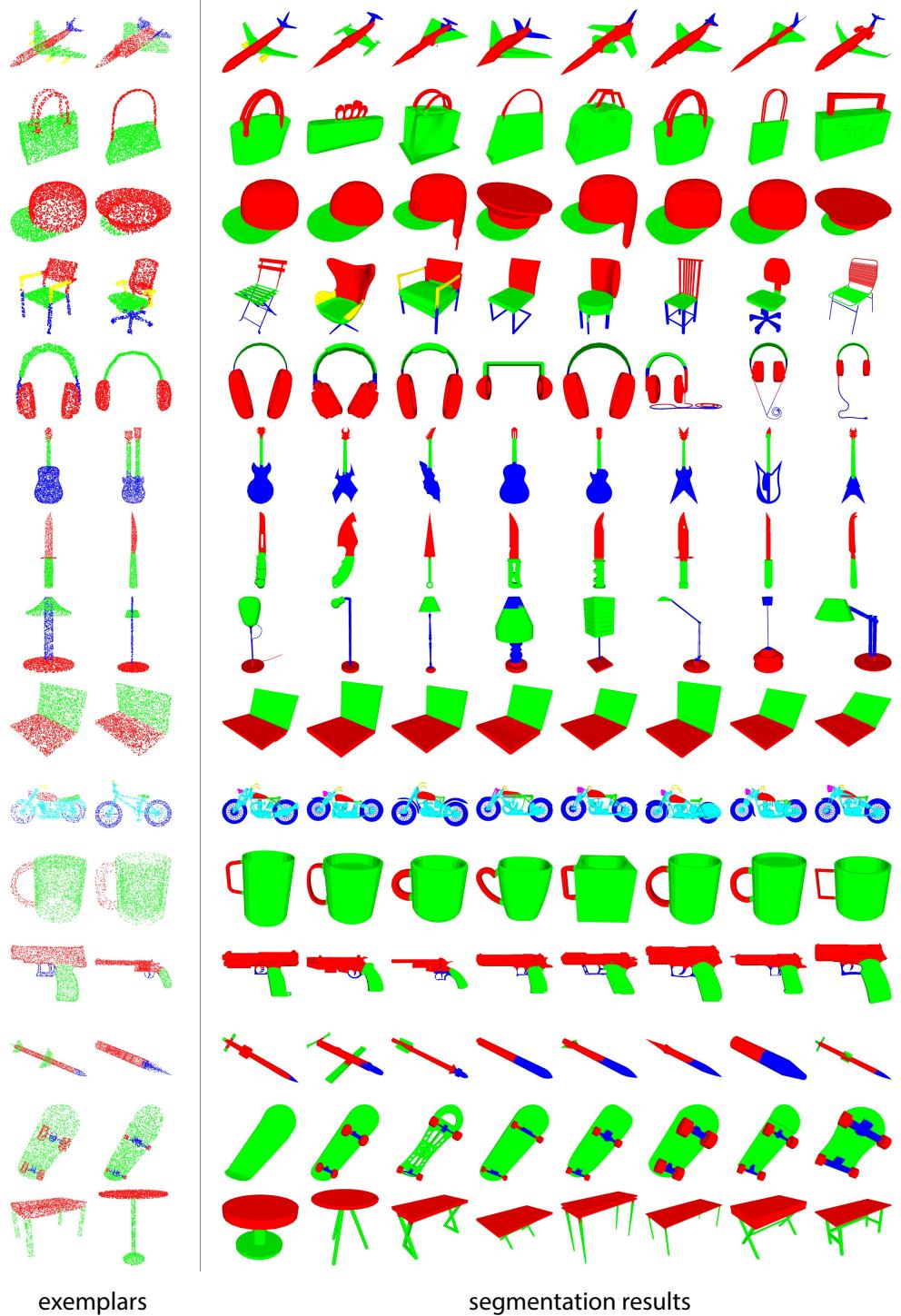


Figure 8. One-shot segmentation results by BAE-NET, with two segmented exemplars.

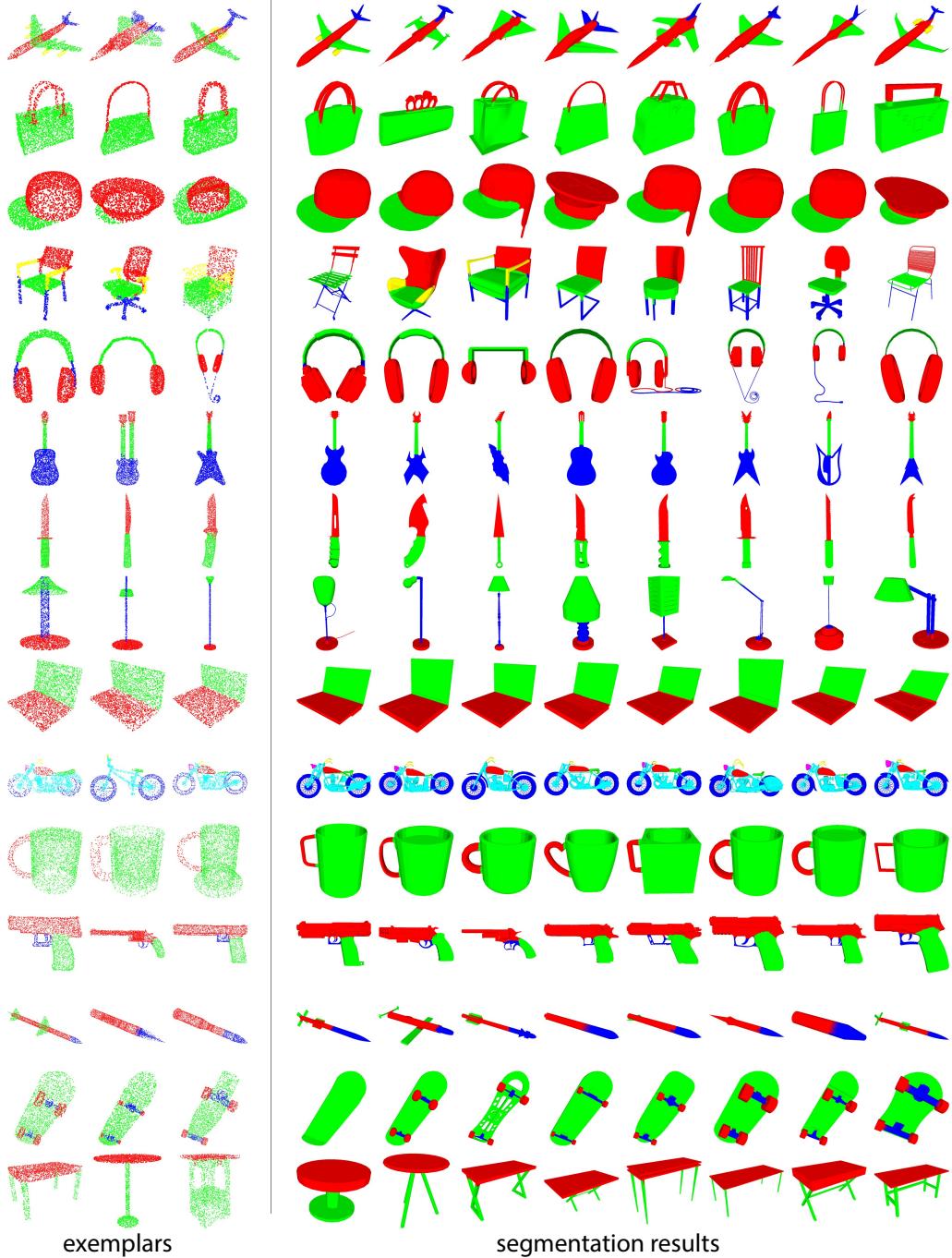


Figure 9. One-shot segmentation results by BAE-NET, with three segmented exemplars.

NETWORK STRUCTURE

This document provides more details about network structures and training configurations.

* IN stands for instance normalization.

A. Autoencoders for 2D shapes

(1) Encoders

*CNN model and our model share the same type of CNN encoder

Encoder for 64^3 input:

<i>Layer</i>	<i>Kernel size</i>	<i>Stride</i>	<i>Activation function</i>	<i>Output shape (d1, d2, channel)</i>
input pixels	-	-	-	(64,64,1)
conv2d	(4,4)	(2,2)	IN LReLU	(32,32,64)
conv2d	(4,4)	(2,2)	IN LReLU	(16,16,128)
conv2d	(4,4)	(2,2)	IN LReLU	(8,8,256)
conv2d	(4,4)	(2,2)	IN LReLU	(4,4,512)
conv2d	(4,4)	-	Sigmoid	(1,1,16)

Encoder for 128^3 input:

<i>Layer</i>	<i>Kernel size</i>	<i>Stride</i>	<i>Activation function</i>	<i>Output shape (d1, d2, channel)</i>
input pixels	-	-	-	(128,128,1)
conv2d	(4,4)	(2,2)	IN LReLU	(64,64,32)
conv2d	(4,4)	(2,2)	IN LReLU	(32,32,64)
conv2d	(4,4)	(2,2)	IN LReLU	(16,16,128)
conv2d	(4,4)	(2,2)	IN LReLU	(8,8,256)
conv2d	(4,4)	(2,2)	IN LReLU	(4,4,512)
conv2d	(4,4)	-	Sigmoid	(1,1,16)

(2) CNN decoders

Decoder for 64^3 input:

<i>Layer</i>	<i>Kernel size</i>	<i>Stride</i>	<i>Activation function</i>	<i>Output shape (d1, d2, channel)</i>
feature code	-	-	-	(1,1,16)
deconv2d	(4,4)	-	IN LReLU	(4,4,512)
deconv2d	(4,4)	(2,2)	IN LReLU	(8,8,256)
deconv2d	(4,4)	(2,2)	IN LReLU	(16,16,128)
deconv2d	(4,4)	(2,2)	IN LReLU	(32,32,64)
deconv2d	(4,4)	(2,2)	Sigmoid	(64,64,4)
max-pooling	(1,1)	(1,1)	-	(64,64,1)

Decoder for 128^3 input:

<i>Layer</i>	<i>Kernel size</i>	<i>Stride</i>	<i>Activation function</i>	<i>Output shape (d1, d2, channel)</i>
feature code	-	-	-	(1,1,16)
deconv2d	(4,4)	-	IN LReLU	(4,4, 512)
deconv2d	(4,4)	(2,2)	IN LReLU	(8,8,256)
deconv2d	(4,4)	(2,2)	IN LReLU	(16,16,128)
deconv2d	(4,4)	(2,2)	IN LReLU	(32,32,64)
deconv2d	(4,4)	(2,2)	IN LReLU	(64,64,32)
deconv2d	(4,4)	(2,2)	Sigmoid	(128,128,4)
max-pooling	(1,1)	(1,1)	-	(128, 128,1)

(3) Our 3-Layer model

Decoder for 64^3 input:

<i>Layer</i>	<i>Input shape</i>	<i>Activation</i>	<i>Output shape</i>
feature code + coordinates	(16+2)	-	(18)
fully-connected	(18)	LReLU	(256)
fully-connected	(256)	LReLU	(256)
fully-connected	(256)	Sigmoid	(4)
max-pooling	(4)	-	(1)

Decoder for 128^3 input:

<i>Layer</i>	<i>Input shape</i>	<i>Activation</i>	<i>Output shape</i>
feature code + coordinates	(16+2)	-	(18)
fully-connected	(18)	LReLU	(512)
fully-connected	(512)	LReLU	(512)
fully-connected	(512)	Sigmoid	(4)
max-pooling	(4)	-	(1)

B. Autoencoders for 3D shapes

Encoder:

<i>Layer</i>	<i>Kernel size</i>	<i>Stride</i>	<i>Activation function</i>	<i>Output shape (d1,d2,d3, channel)</i>
input voxels	-	-	-	(64,64,64,1)
conv3d	(4,4,4)	(2,2,2)	IN LReLU	(32,32,32,32)
conv3d	(4,4,4)	(2,2,2)	IN LReLU	(16,16,16,64)
conv3d	(4,4,4)	(2,2,2)	IN LReLU	(8,8,8,128)
conv3d	(4,4,4)	(2,2,2)	IN LReLU	(4,4,4,256)
conv3d	(4,4,4)	-	Sigmoid	(1,1,1,128)

Decoder for unsupervised tasks:

<i>Layer</i>	<i>Input shape</i>	<i>Activation</i>	<i>Output shape</i>
feature code + coordinates	(128+3)	-	(131)
fully-connected	(131)	LReLU	(3072)
fully-connected	(3072)	LReLU	(384)
fully-connected	(384)	Sigmoid	(12)
max-pooling	(12)	-	(1)

Decoder for one-shot training:

*n is the number of ground truth parts

<i>Layer</i>	<i>Input shape</i>	<i>Activation</i>	<i>Output shape</i>
feature code + coordinates	(128+3)	-	(131)
fully-connected	(131)	LReLU	(1024)
fully-connected	(1024)	LReLU	(256)
fully-connected	(256)	Sigmoid	(n)
max-pooling	(n)	-	(1)

E. Training configurations

The networks were implemented with Tensorflow and using Adam optimizer (learning_rate=5e-5, beta1=0.5, beta2=0.999, epsilon=1e-8).

For leaky ReLU, alpha=0.02.

The batch size for all network is 1 (shape). The actual number of points varies according to the input shapes.