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 the datasets 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. We perform additional experiments (adjusting network capacity, adding regularization) for supervised methods training on 10% train data to address the possible overfitting issues in Table 2. We also train supervised methods on 10 random training examples per category, repeated 10 times, and report the results in Table 2. Besides, we show the 1-exemplar results of 2/3/4/5-layer versions of our model in Table 3 and Figure 7 8 9 10. We use { 1024-n }, { 1024-256-n } (our default model in the paper), { 1024-256-256-n } and { 1024-256-256-256-n } for the 2-layer, 3-layer, 4-layer and 5-layer models, respectively, and train them on 8 randomly selected exemplars for each category.

5. One-shot segmentation results

We show in Figure 11 12 13 additional results of our one-shot segmentation with 1, 2 and 3 segmented exemplar(s).

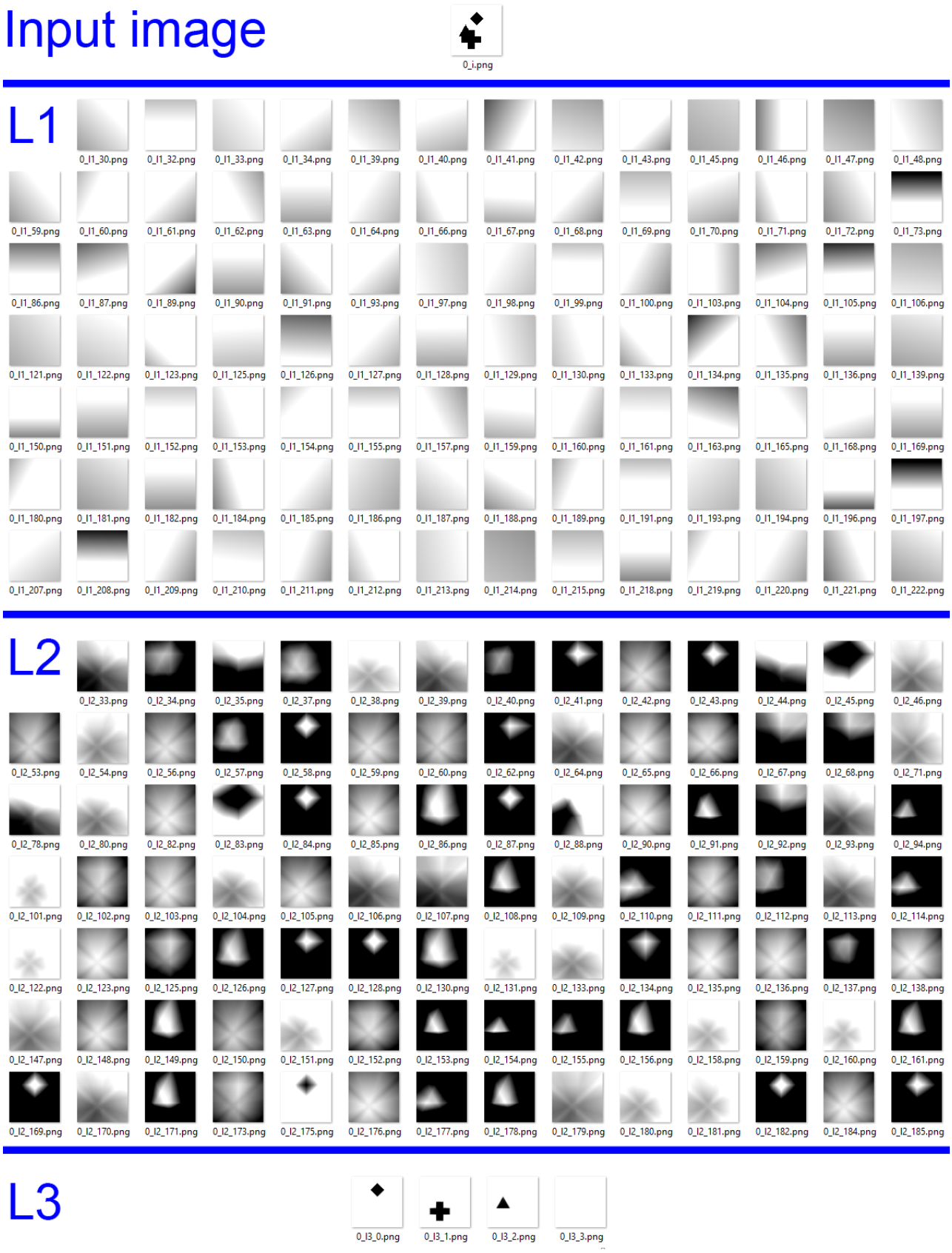

Figure 1. Neuron activations for the model trained on “elements”.

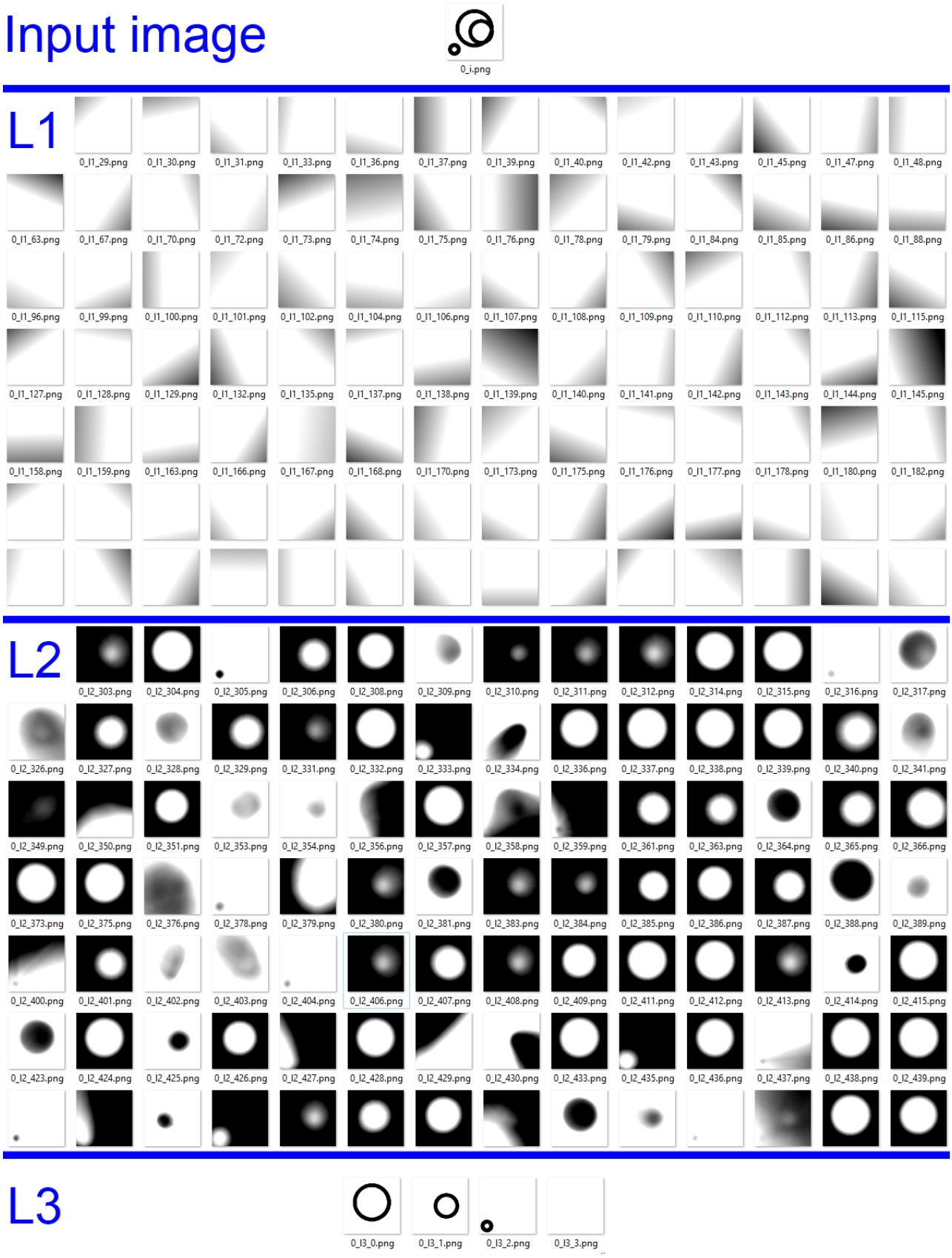

Figure 2. Neuron activations for the model trained on “triple rings”.

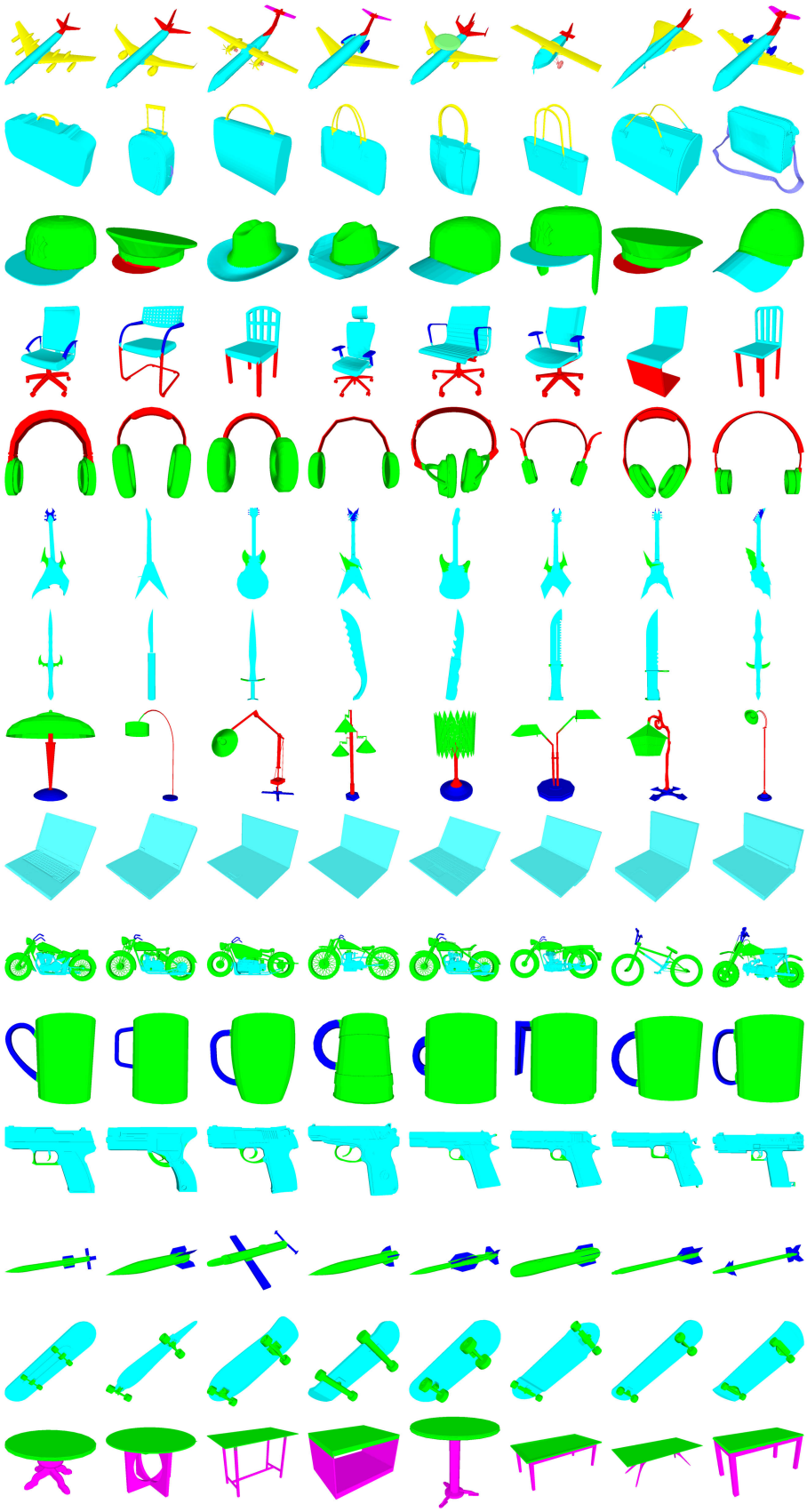

Figure 3. Unsupervised segmentation results on ShapeNet Part dataset.

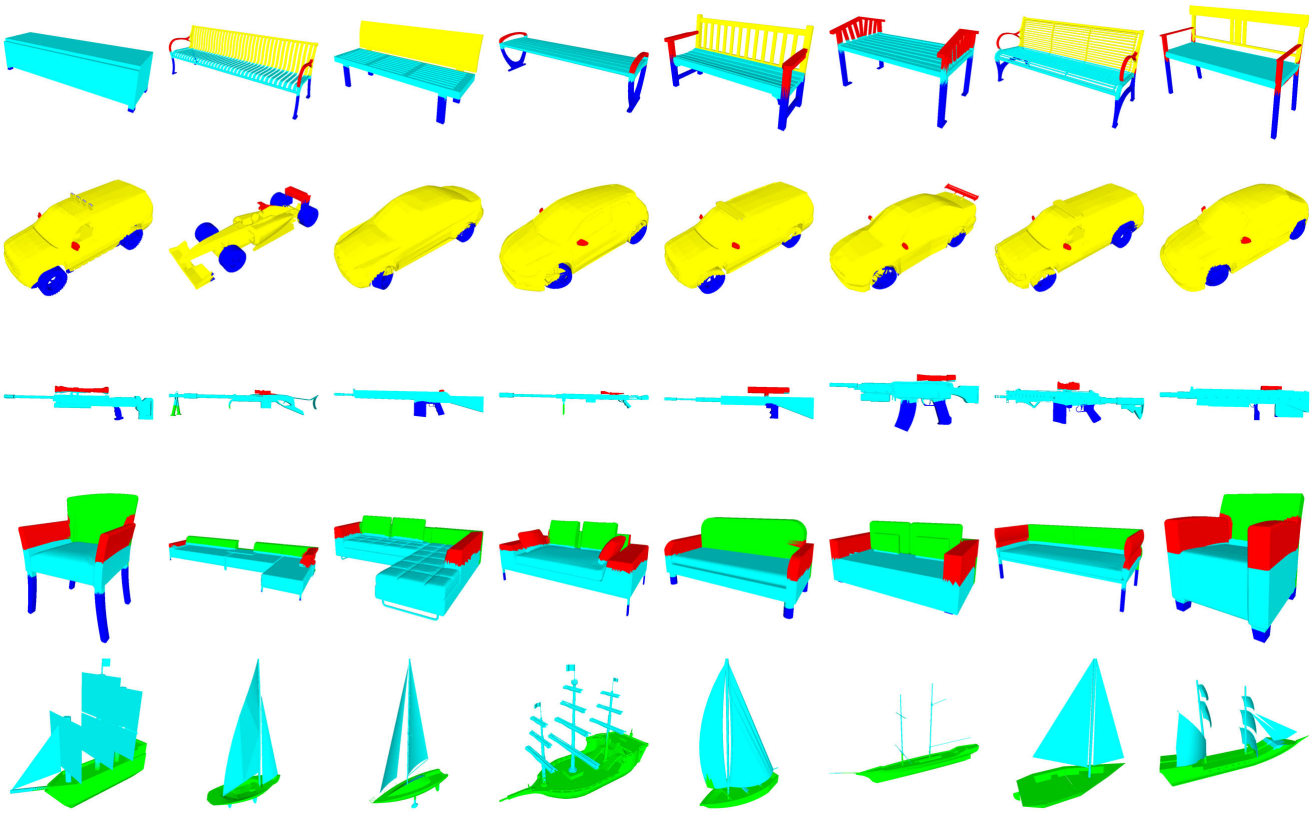

Figure 4. Unsupervised segmentation results on five ShapeNet categories which are not included in Figure 3.

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

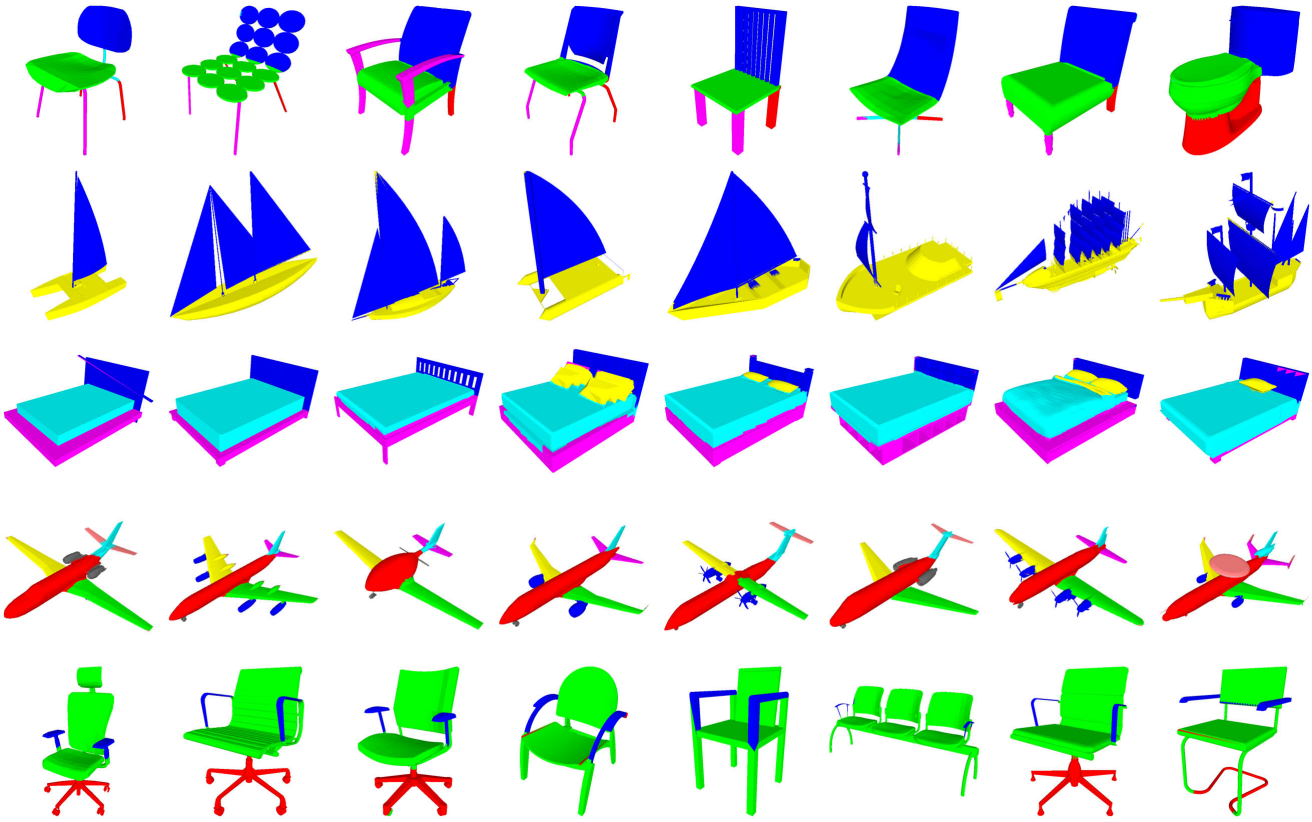

Figure 6. Weakly-supervised segmentation results on the dataset of Tags2parts.

	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	73.2	73.1	74.7	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	85.4	83.5	83.9	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	82.0	85.5	85.5	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	85.4	85.7	86.0	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.	72.3	70.3	76.2	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	87.0	89.0	87.8	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	83.9	83.0	83.6	77.2	79.7	79.2	81.3	28.4	77.4	79.6	81.6	24.9	80.6	83.8	84.4	67.7	76.0	80.4	80.9
lamp	70.1	71.8	70.1	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	94.6	94.7	94.8	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.	54.2	63.4	64.6	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	94.9	95.2	94.8	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	78.3	77.6	78.7	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	42.0	45.7	52.1	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.	73.0	72.6	74.2	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	73.2	72.9	73.3	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	76.6	77.6	78.7	65.6	72.1	73.0	74.6	57.9	73.5	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.

	10-exemplar		(10% train data) Adjust network capacity							(10% train data) Add regularization							
	mean	max	1/8	2/8	3/8	4/8	5/8	6/8	7/8	1e-3	1e-4	1e-5	1e-6	1e-7	1e-8	1e-9	1e-10
PointNet	67.9	73.4	70.5	72.0	**73.8**	72.8	73.4	71.6	70.5	69.4	69.9	71.4	71.5	72.2	71.9	71.7	71.5
PointNet++	68.8	73.0	65.2	69.4	71.5	72.2	73.7	73.5	73.7	70.2	73.7	73.8	73.6	**74.0**	73.2	73.7	73.7
PointCNN	31.2	37.4	58.3	57.4	56.3	57.4	56.7	57.2	56.5	44.3	56.3	56.4	57.5	**58.6**	56.3	57.0	56.4
SSCN	48.7	56.2	54.6	56.5	55.4	55.9	56.8	56.4	56.3	**57.8**	55.7	55.7	56.7	57.3	55.6	56.7	57.2

Table 2. Results of additional experiments, including adjusting network capacity and adding regularization for supervised methods training on 10% train data, and training supervised methods on 10 random training examples per category (repeated 10 times). In "10-exemplar", "mean" is the average IOU on the average result for each category, "max" is the average IOU on the best result for each category. In "Adjust network capacity", the numbers in the second row indicate the adjusted network capacity. In "Add regularization", the numbers in the second row indicate the scale for the regularization term. The best number for each method is shown in boldface.

	plane	bag	cap	chair	earph.	guitar	knife	lamp	laptop	motor.	mug	pistol	rocket	skate.	table	Mean
2-layer	62.2	78.2	80.7	84.7	**78.9**	54.1	32.0	**73.5**	94.5	29.5	92.8	75.1	30.4	63.6	75.5	67.0
3-layer	73.2	**85.4**	82.0	85.4	72.3	87.0	**83.9**	70.1	**94.6**	54.2	94.9	**78.3**	42.0	73.0	73.2	76.6
4-layer	**73.7**	81.6	**82.4**	**85.9**	78.2	**88.4**	82.4	72.9	94.5	63.1	**95.0**	77.8	**50.3**	**75.0**	**76.1**	**78.5**
5-layer	72.7	84.6	81.6	**85.9**	77.6	88.1	83.0	61.8	94.3	**63.5**	94.5	77.8	43.3	74.5	75.6	77.3

Table 3. 1-exemplar results of 2/3/4/5-layer versions of our model. The best number for each category is shown in boldface.

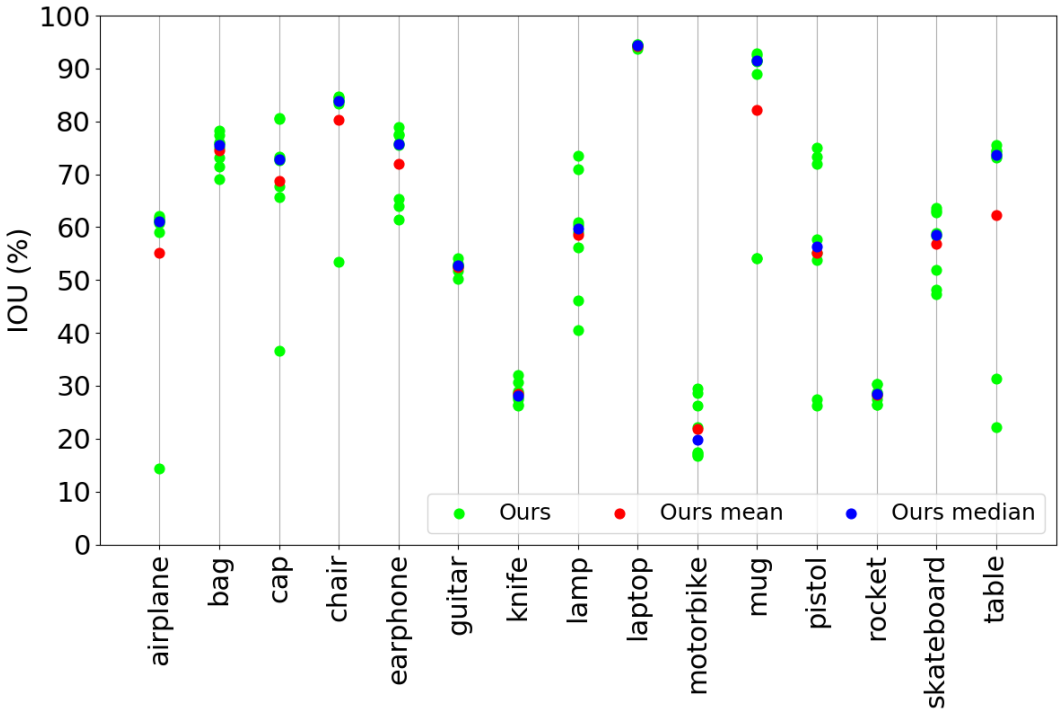

Figure 7. 1-exemplar results of ours 2-layer model ($\{$ 1024-n $\}$) for each shape category on 8 randomly selected exemplars.

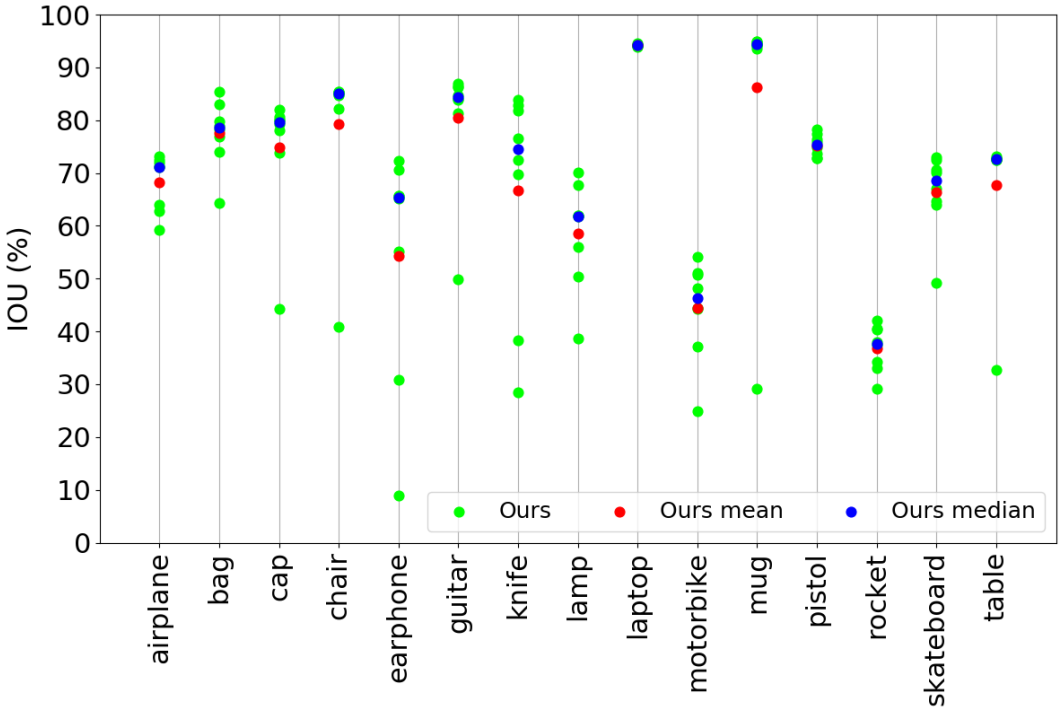

Figure 8. 1-exemplar results of ours 3-layer model ({ 1024-256-n }) for each shape category on 8 randomly selected exemplars.

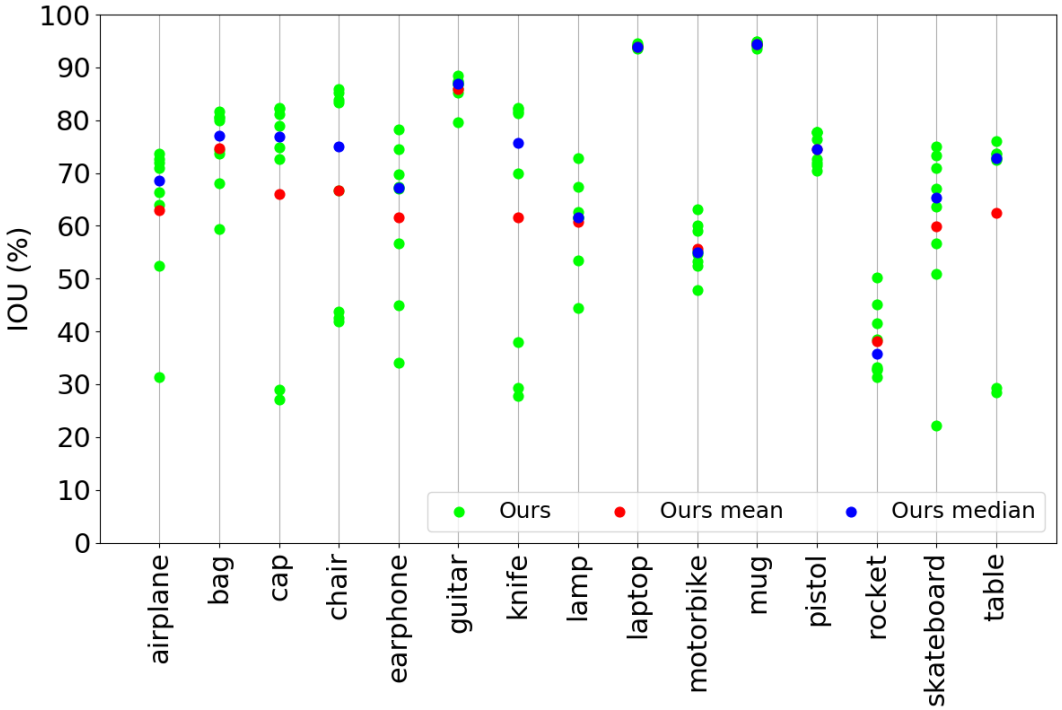

Figure 9. 1-exemplar results of ours 4-layer model ({ 1024-256-256-n }) for each shape category on 8 randomly selected exemplars.

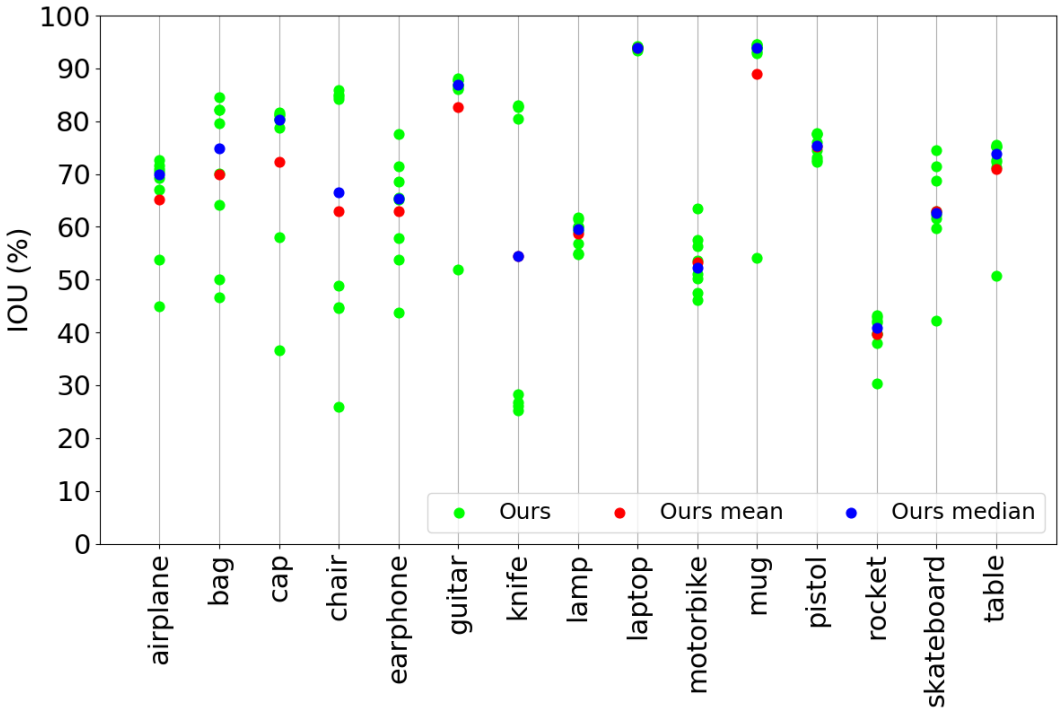

Figure 10. 1-exemplar results of ours 5-layer model ({ 1024-256-256-256-n }) for each shape category on 8 randomly selected exemplars.

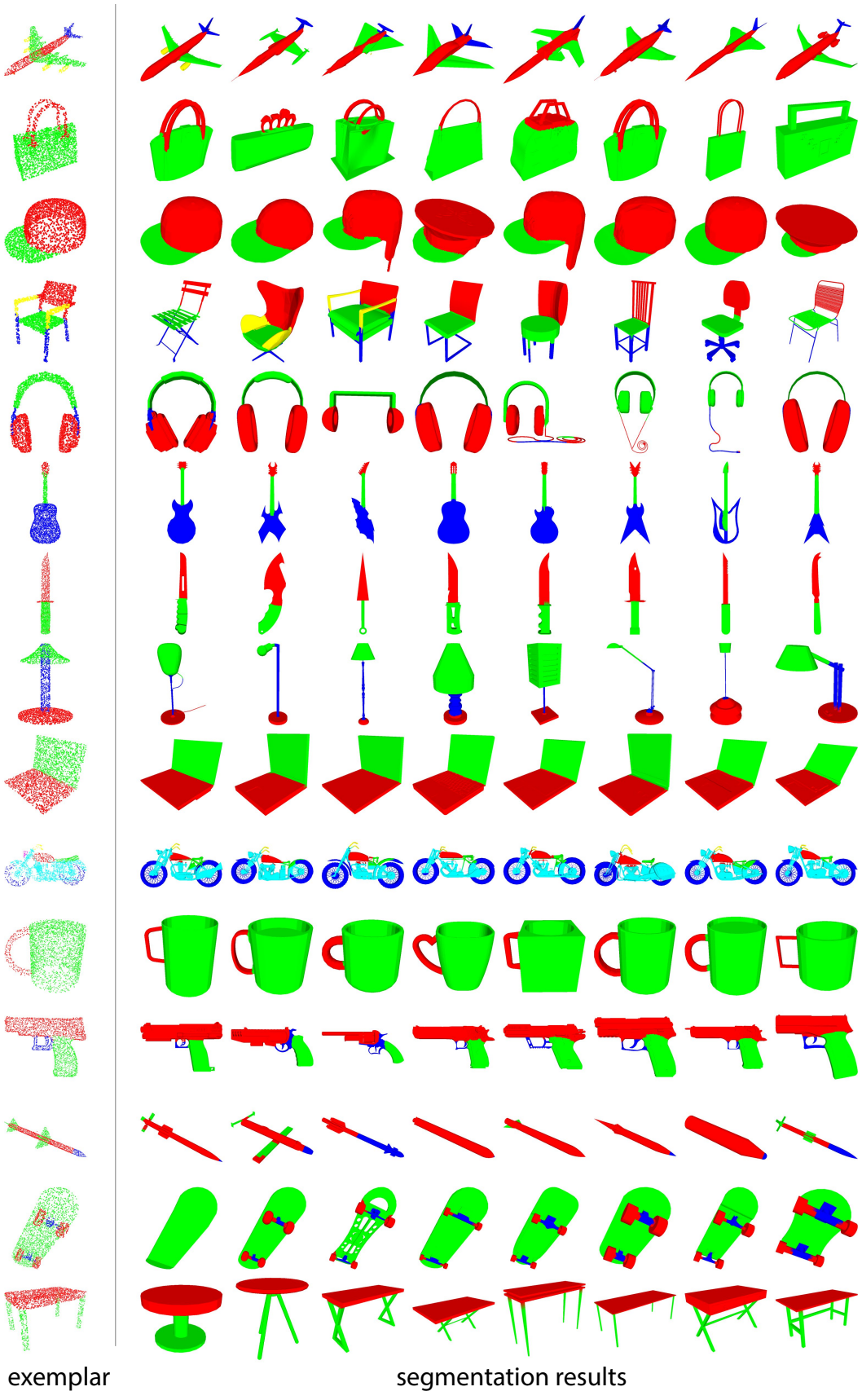

Figure 11. One-shot segmentation results by BAE-NET, with one segmented exemplar.

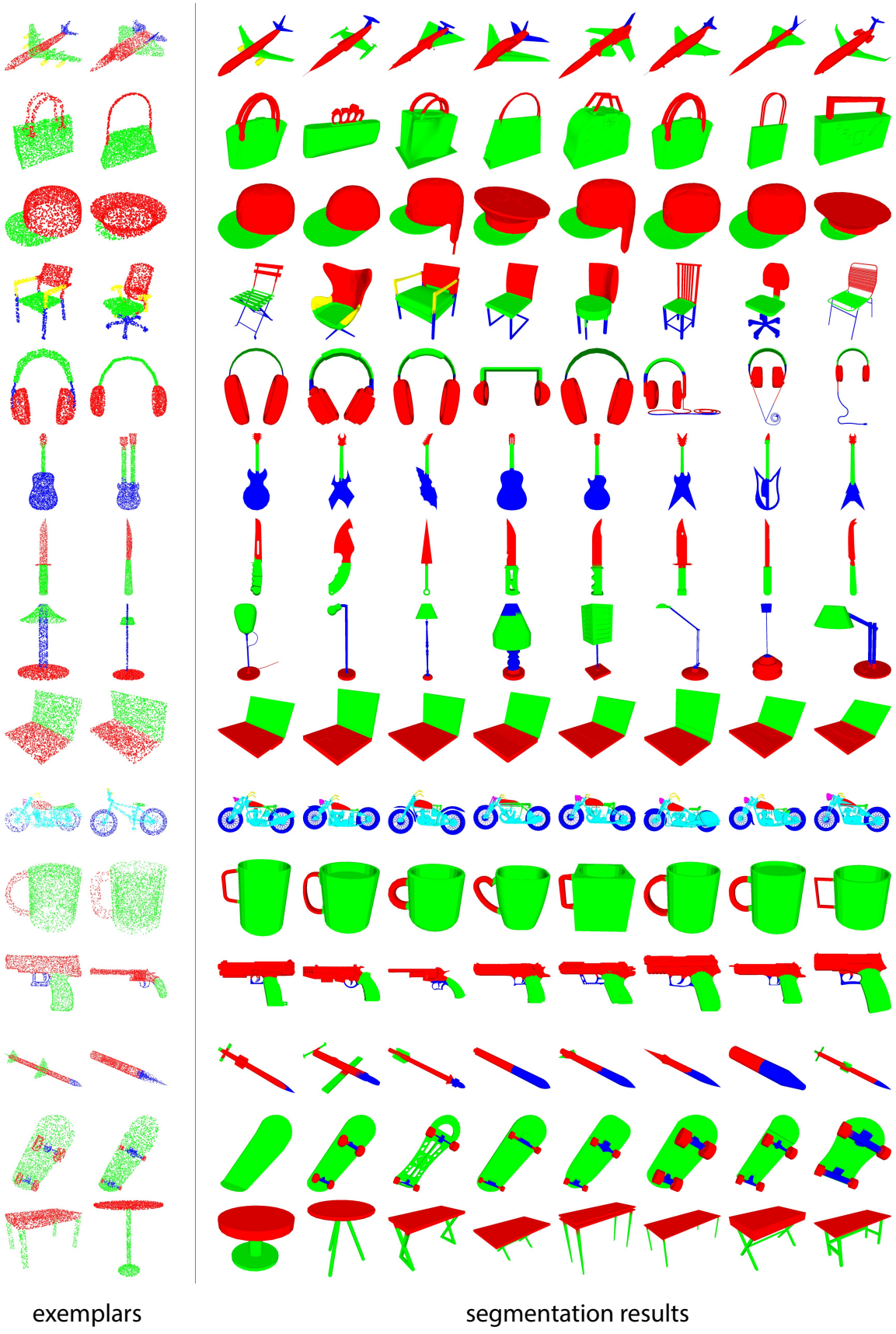

Figure 12. One-shot segmentation results by BAE-NET, with two segmented exemplars.

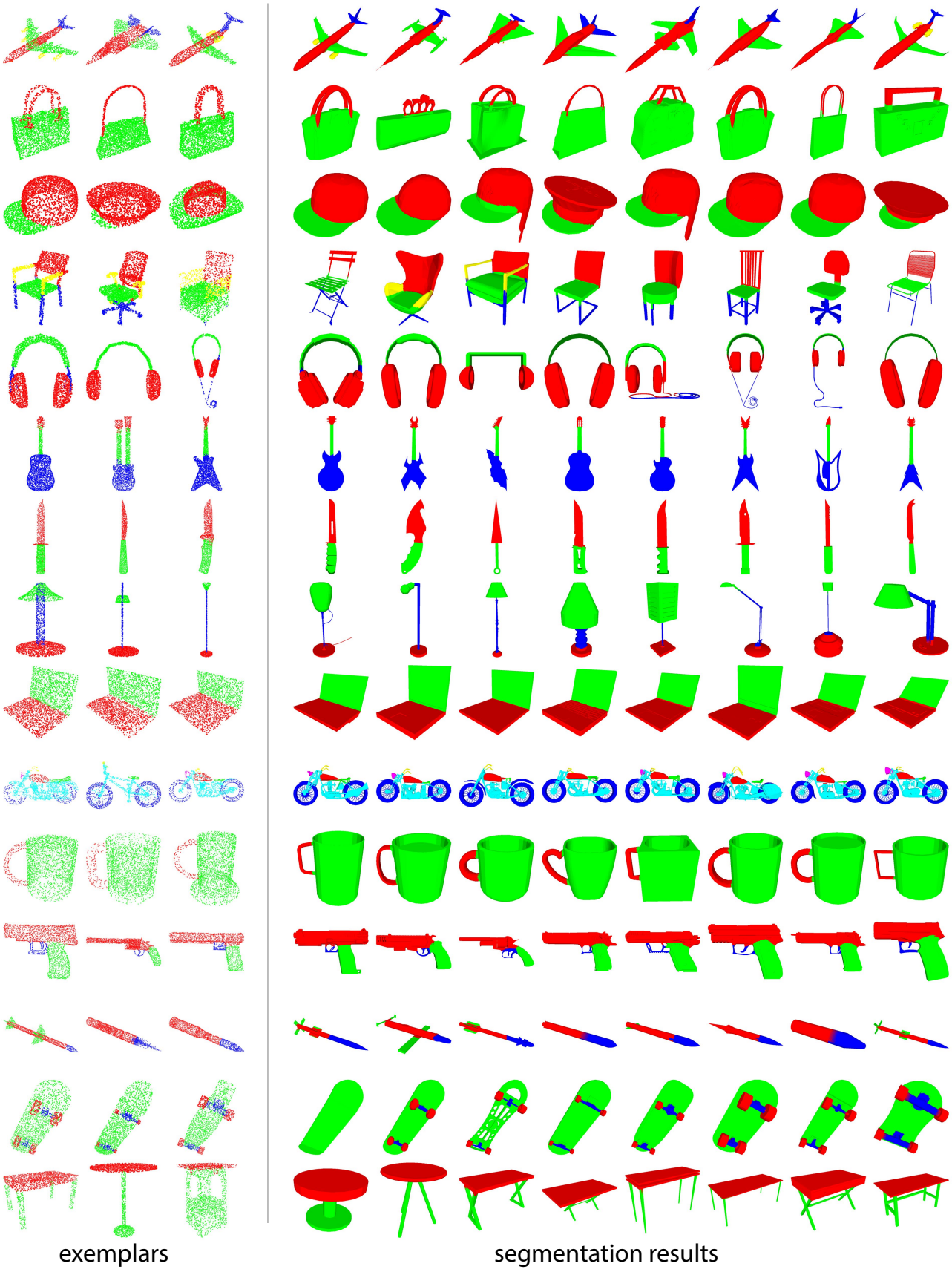

Figure 13. One-shot segmentation results by BAE-NET, with three segmented exemplars.

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:

Layer	*Kernel size*	*Stride*	*Activation function*	*Output shape (d1, d2, channel)*
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:

Layer	*Kernel size*	*Stride*	*Activation function*	*Output shape (d1, d2, channel)*
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:

Layer	*Kernel size*	*Stride*	*Activation function*	*Output shape (d1, d2, channel)*
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:

Layer	*Kernel size*	*Stride*	*Activation function*	*Output shape (d1, d2, channel)*
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:

Layer	*Input shape*	*Activation*	*Output shape*
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:

Layer	*Input shape*	*Activation*	*Output shape*
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:

Layer	*Kernel size*	*Stride*	*Activation function*	*Output shape (d1,d2,d3, channel)*
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:

Layer	*Input shape*	*Activation*	*Output shape*
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

Layer	*Input shape*	*Activation*	*Output shape*
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.