# Supplemental Material for The Neurally-Guided Shape Parser: Grammar-based Labeling of 3D Shape Regions with Approximate Inference

## A. Data Details

### A.1. Data Preprocessing

Ground-truth shape regions for instances in our datasets are created from PartNet [9]. Specifically, we create a shape region for each connected mesh-component of a given shape in PartNet. This guarantees the region decomposition is at minimum an instance segmentation, as each leaf part instance must correspond with at least 1 connected mesh-component, but it many cases it creates an instance over-segmentation, as a single leaf part instance will be represented by more than 1 mesh-component. To determine the semantic label of a region, we find the terminal label in the shape grammar that is a parent of the label assigned to the shape region by PartNet. Note that there will only ever be one such parent (because of the unique path assumption from Section 3). There are some shape instances where there are no parent labels that meet this criteria, in which case we do not include the shape in our datasets. Details of the shape grammars we use can be found in Section F. Additionally we filter out any shape instances with very small part regions that would be hard to reason over with point clouds of size 10000 (the number of points used by all point cloud consuming networks we consider). If any shape region has an area that makes up less than 0.1% of the total shape area, the entire shape instance is ignored.

### A.2. Creating Dataset Splits

We used shapes from the chair, lamp, table, vase, knife, and storage furniture categories of PartNet. These were all the categories that had at least 250 valid shape instances (as defined by criteria in Section A.1) and had shape grammar definitions at a fine granularity. For each of these datasets, we created train/val/test splits so that no set had more than 400 shapes or less than 50 shapes. Additionally, when making the splits we employed a greedy strategy where we tried to keep each terminal label of the grammar evenly represented (both across and within sets), whenever possible. Exact numbers of shapes in each split, for each category, can be found in Table 1. For all experiments, validation and test sets remain constant. In the experiment where the number of training set examples is varied, each training set at a smaller size is a proper subset of the corresponding larger training set.

| Set | Chair | Lamp | Table | Storage | Vase | Knife |
|---|---|---|---|---|---|---|
| Train | 400 | 400 | 400 | 400 | 400 | 239 |
| Validation | 400 | 157 | 400 | 55 | 54 | 50 |
| Test | 400 | 157 | 400 | 55 | 54 | 50 |

Table 1. The number of shapes in the train, validation, and test set splits for each category.

### A.3. Region Corruption

In Section 4.5 of the main paper, we describe an experiment where we analyze the robustness of NGSP to corruptions of the input regions. We experiment with two levels of corruptions, the 2X paradigm, where each region is split into 2 regions, and the 4X paradigm, where each region is split into 4 regions. The corruptions for the 2X paradigm are produced in the following manner. For each region, a random mesh face, $f_0$, from that region is sampled. Then, treating each face as the average of its vertices, the furthest mesh face from $f_0$, $f_1$, is found. We then find the furthest mesh face from $f_1$ within the region, $f_2$. For all other faces within the region, we record their distance from both $f_1$ and $f_2$, and assign each face to the cluster it is closest to. Regions containing a single mesh face are not further split. To generate the 4X splits, the 2X split procedure is applied twice in succession.

## B. Training Details

### B.1. Training Hyperparameters

The geometry, layout, and guide networks are variants of PointNet++'s written in PyTorch [10–12]. All PointNet++'s use 3 set abstraction layers with 1024, 256, 64 grouping points, 0.1, 0.2, 0.4 radius size and 64, 128, 256 feature size respectively, with the global pooling step done at a feature size of 1024. ReLU activations are used throughout. All multilayer perceptron (MLP) heads for per-shape or per-point classification use 2 hidden layers with dimensions of 256 and

64 with leaky ReLU activations (slope 0.02). We train with batch size of 16, without any batch normalization, but with dropout in the MLPs of 0.4 within the guide network and 0.2 for the geometry and layout networks. For the guide network, batch members are randomly sampled from the data. For the geometry and layout network, each batch contains one positive example, and the rest of the batch contains negative examples mined with respect to the positive one. The binary classification loss is computed with a mean operation independently for the positive and negative examples, and then summed together. We convert mesh data into 100000 points sampled uniformly from the surface of each shape, in order to be able to reason about regions of small, fine-grained parts. The guide network uses point clouds with 10000 points while the geometry and layout networks uses 4096 points. If there are less than the expected number of points corresponding to the union of regions needed while querying the geometry or layout networks, we repeat points in order to facilitate batch training and evaluation. All networks receive data augmentation in the form of random non-uniform scales and point-wise perturbations.

The region network uses a graph neural network architecture that operates over a graph of shape regions; we describe the graph generation process in Section B.4, that constructs the graph topology and initializes the per-node and per-edge features. The region network performs 4 cycles of gated graph convolution [3, 6], with residual node connections, a batch size of 16, and no batch normalization, After the graph convolutions, a global readout max-pooling operation is applied across all node features within each graph, so that there is a single feature representation for each graph. For the label region network, a linear layer takes this feature representation and predicts a probability distribution over terminals in the grammar. For the area region network, a linear layer takes this feature representation and predicts a single scalar value.

## B.2. Semantic Label Negative Sampling Strategies

As mentioned in the main paper, positive examples for the geometry and layout networks are sourced directly from the ground-truth label assignments to regions of the input shapes. Negative examples are sourced by finding label assignments different from the canonical version, that would change the assigned regions, or child labels of the assigned regions, for the label of interest. However, negative examples that are too similar to their corresponding positive examples are not used. To check this, we measure the percentage of region area that is unchanged between the positive and negative example, if this percentage is above .95, then we ignore that negative example. We employ a set of negative sampling strategies to find suitable 'incorrect' label assignments, detailed as follows:

**Label Assignment Perturbations** One way to source negative examples is to find 'incorrect' label assignments that are deviations from the original label assignment to the entire shape. We do this by creating 9999 perturbed label assignments for each shape, where 100 have 1 label change, 200 and 2 label changes, 300 have 3 label changes, 400 have 4 label changes, 500 have 5 label changes, 500 have 10% label changes, 1000 have 20% label changes, 1500 have 30% label changes, 2000 have 40% label changes, 2500 have 50% label changes and 999 have all labels changed. When sampling label changes, we make it more likely that labels will be switched to other labels in the grammar they are closer to in the hierarchy of labels implied by the grammar.

**Adding Regions** In this method to source negative examples for a given shape and label, we first identify all regions that are assigned to this label, and all regions that are not assigned to this label, under the canonical labeling. Then some regions (randomly sampled) that are not assigned to this label are switched to be assigned to the label of interest.

**Removing Regions** In this method to source negative examples for a given shape and label, we first identify all regions that are assigned to this label under the canonical labeling. Then some regions (randomly sampled) that are assigned to this label are removed and are no longer assigned to the label of interest.

**Using Regions from different Parts** In this method to source negative examples for a given shape and label, we first identify all regions that are not assigned to this label under the canonical labeling. Then a subset of these regions (randomly sampled) are assigned to the label of interest (with all other regions in the shape unassigned to this label).

**Using Regions from a different Shape** In this method to source negative examples for a given shape and label, we first find all other shapes in the dataset where the label of interest was not seen, and sample one such shape. Then a subset of regions (randomly sampled) from this sampled shape is assigned to the label as a negative example.

**Changing the Child Label of Regions** This method to source negative examples for a given shape and label is only used by the layout networks. No regions of the input shape are changed, instead for each region there is a 50% chance to change the assigned child label of the region to a different random value (consistent with the available options defined by the grammar).

For all other negative sampling strategies, when negative examples are generated for the layout network and a new

region is added, a random child label is assigned in the same fashion.

### B.3. Differentiating Geometry and Layout Networks

The differences between the geometry and layout networks come from the input features, the labels in the shape grammar they cover, and the types of negative examples they learn from. Both types of networks principally operate over points that come from regions assigned to the label of interest. The layout network additionally receives the assigned child label of each point with respect to the corresponding label as a onehot vector concatenated to the XYZ position of each point. As the terminal labels of the shape grammar have no children labels, they are not assigned any layout networks. As the root label of the shape grammar always encompasses all shape regions, there is no geometry network assigned to it. The types of negative examples seen by each network are sampled at different rates. For the geometry network, the sampling probabilities for each negative example are (corresponding to the strategies listed in B.2): 50% label assignment perturbations, 15% adding regions, 15% removing regions, 15% using regions from different parts, 5% using regions from a different shape and 0% changing the child label of regions. For the layout network, the sampling probabilities for each negative example are (corresponding to the strategies listed in B.2): 50% label assignment perturbations, 7.5% adding regions, 7.5% removing regions, 7.5% using regions from different parts, 2.5% using regions from a different shape and 25% changing the child label of regions.

### B.4. Region Graph Creation

The region network takes as input a collection of shape regions represented as a graph. The nodes of this graph correspond to shape regions, and the edges are created such that the graph is fully connected. To populate the initial node and edge features we use point cloud auto-encoders. The point cloud auto-encoders are trained on a collection of chair shapes we gather from PartNet; we make use of their shape region decompositions but do not use their label information. For the node feature, we train a PointNet++ to consume a 1024 dimensional point cloud sampled from one shape region, project it into a 64 dimensional latent space, and then decode the 64 dimensional vector with a 3-layer MLP into a 1024 x 3 vector; encouraging the input and output point clouds to match with a Chamfer distance loss. For the edge feature, we train a point cloud auto-encoder to consume two 1024 dimensional point clouds sampled from two shape regions. The point clouds are distinguished from one another with a one hot encoding appended to each XYZ position. These point clouds are concatenated together to form a 2048 x 5 input vector. A PointNet++ module consumes this input and projects it into a 64 dimensional latent space. This 64 dimensional vector is then run through a 3-layer MLP to form a 2048 x 3 vector. We interpret the first 1024 of these points to correspond to the first region and the last 1024 of these points to correspond to the second region, and encourage each decoded point cloud to match its target with a Chamfer distance loss. For both paradigms, all shape regions are centered and scaled to lie within the unit sphere, and training is done with the Adam optimizer, a learning rate of 0.0001, and a batch size of 32. Both the region point cloud auto-encoder and the paired-region point cloud auto-encoder are pretrained and frozen; they are used across categories and labeled data training set sizes. Finally, when creating region graphs for the region network, per-node features are created by running the region through the region point cloud auto-encoder, and per-edge features are created by running pairs of regions through the paired-region point cloud auto-encoder. The position and scale of each region are also concatenated onto each per-node feature.

## C. Perceptual Study

As described in Section 4.6 of the main paper, we ran a perceptual user-study to determine semantic segmentation performance for 'in the wild' shape instances that lacked ground-truth label annotations. We recruited 12 college students to participate in our study. Each participant was shown a sequence of 46 shape segmentation examples. Each example compared the shape segmentations produced by two different methods (either NGSP and PartNet or NGSP and LHSS). We visualized each labeling by expanding the label hierarchy of the grammar. For each label of the grammar, when the two label assignments agreed on which parts were assigned to that label, that label was colored purple. When the two methods differed on which parts were assigned to that label, each method's parts were depicted side-by-side and given different colors (orange or blue). Figure 1 shows an example prompt, where the orange labeling is predicted by NGSP and the blue labeling is predicted by LHSS. For each example, the participants were asked to pick the label assignment (orange or blue) that better matched the given shape; we reported quantitative results of this study in Table 4 of the main paper. For this quantitative analysis we included all participants who spent between 5 minutes and 30 minutes on this task; excluding 2 outliers who took 2 minutes and 4 hours respectively to complete the survey.

As described in the main paper, we sourced the 'in the wild' shapes from the ShapeNet dataset [4]. We used shape instances from the chair category, and collected a set of 26 meshes whose given connected-components roughly corresponded to part-instance over-segmentations. For each of these 26 meshes, we produced 2 potential label assignment comparisons (for comparisons against both PartNet and LHSS). The choice of which method should be colored blue or orange was randomized for every rendering. To not

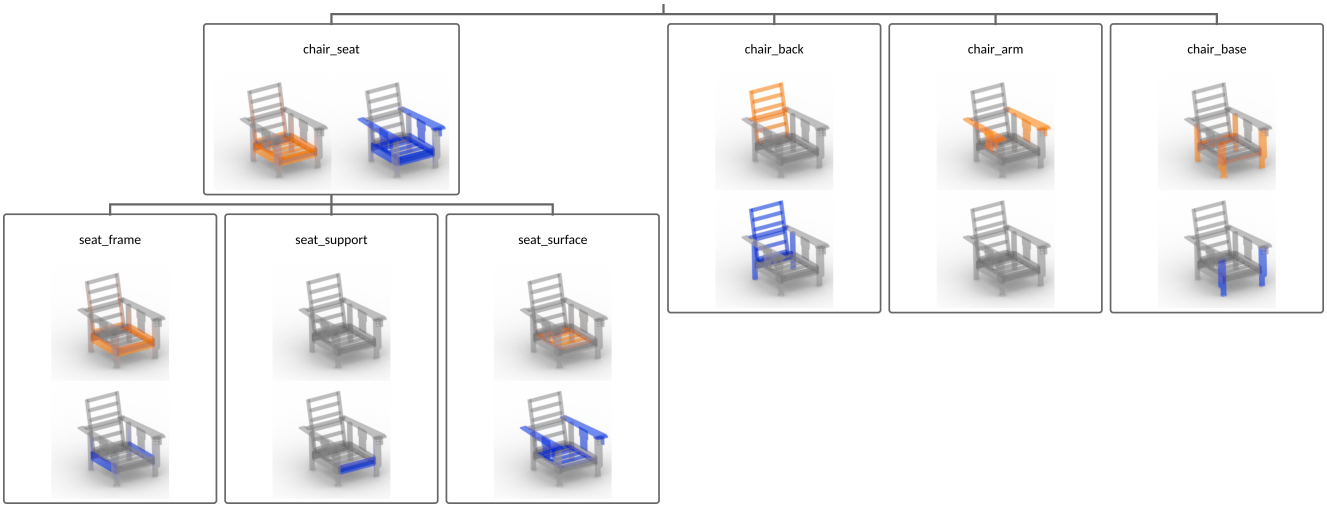



Figure 1. Example comparison from our perceptual study visualizing two label assignments to shape regions sourced from a ShapeNet mesh [4]. The orange labeling is from NGSP and the blue labeling is from LHSS. The supplemental includes additional examples.

overwhelm each participant, instead of expanding the entire chair grammar hierarchy, for each example we always show all the children of the root node (chair back, base, seat, arm, head, footrest), and we randomly choose to show the full expansion of exactly one child. Depending on the given label assignments, some root children expansion views did not present a substantial qualitative difference between the two semantic segmentation methods (either because of very small and hard to perceive shape regions or because the chosen label assignments were the same for the expanded nodes); we manually removed such expansions from the experiment, reducing the number of examples each participant was asked to make judgement on from 52 -> 46. In the supplemental zip-file, we include all of the comparison renders that participants were shown in the experiment. In Figure 2 we include qualitative comparisons of predicted labelings from different methods.

# D. Comparison Method Implementation Details

## D.1. BAE-NET

We follow the implementation provided by the BAE-NET author's whenever possible [5]. To produce voxelizations that BAE-NET takes as input, we take the following steps. First we create a manifold version of the mesh [8]. Then we compute inside-outside values for query points that lie along a grid, using the fast winding number algorithm [1]. We use logic from the BAE-NET code to turn these query point inside-outside values into a voxelization for each shape (used as input to an encoder) and paired (point, value) data used to train the implicit decoder.

BAE-NET has two training modes: supervised and unsupervised training. Supervised training can only be run on shapes that come with semantic labels. In the original BAE-NET implementation, 3000 warm-up epochs are run over the full supervised learning set (all shapes that contain labels), then unsupervised shapes are integrated. After the warm-up phase, after every 4 unsupervised training updates, BAE-NET makes a supervised learning update for every shape instance in the supervised set. We employ this paradigm in the low-data regimes (10/40 labeled data instances). In plentiful labeled data regimes, this strategy is prohibitively slow, so we instead make one full pass through both the unsupervised and supervised examples for each epoch after the warm-up period.

## D.2. LEL

Our label-efficient learning variant utilizes the shape region based self-supervised training scheme proposed in [7]. In their experiments, they source shape region decompositions from an ACD method. In our experiments, we use shape region decompositions provided by PartNet. Following their method, we modify the PointNet++ used in the PartNet variant to include an additional linear layer, that consumes the last per-point feature and outputs a 128 dimensional per-point embedding. The self-supervised loss encouraging per-point embeddings to cluster to similar parts of space is then implemented with code from the label-efficient learning paper.

## D.3. LHSS

We follow the implementation provided by the method's authors whenever possible [13]. We directly use their Julia code that consumes input meshes in order to generate the per-point features. We re-implement their neural network in PyTorch, following all hyper-parameters as described in their released torch code. To solve the MRF formed by per-shape-region unary terms and paired terms that correlate

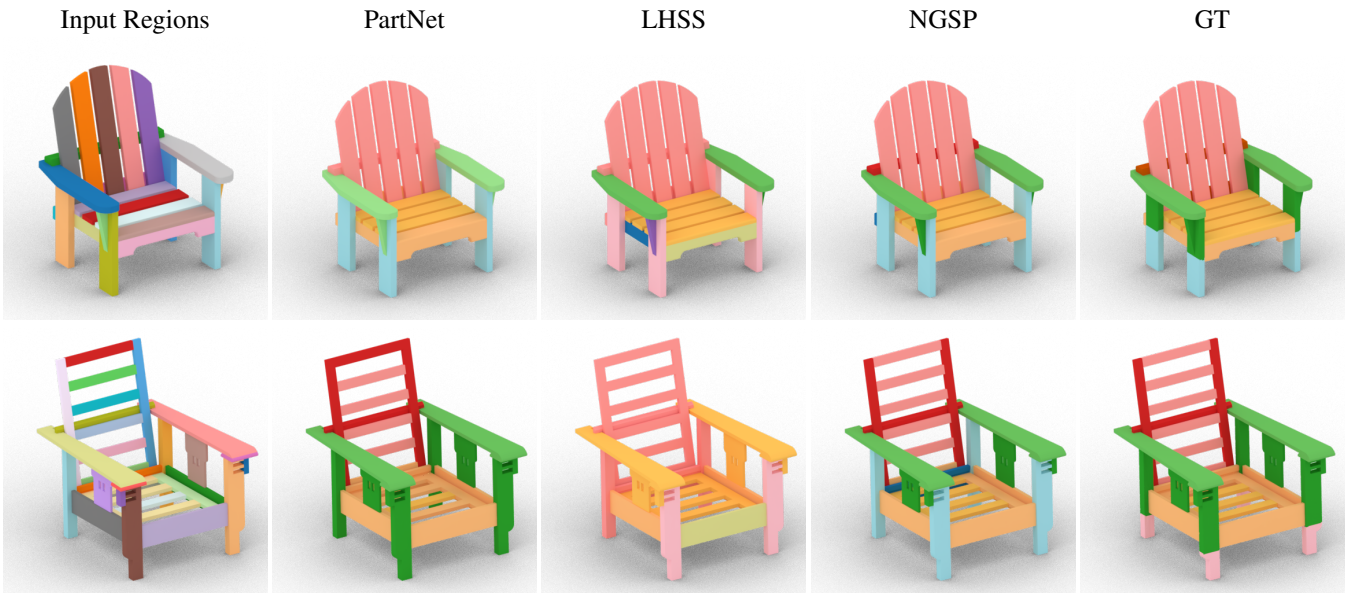



Figure 2. Additional qualitative results for 'in the wild' shapes.

to label distances within the grammar hierarchy, we use the alpha-expansion algorithm from the publically available GCO package [2].

### D.4. Converting Per-Atom Predictions to Per-Region Predictions

As mentioned in the main paper, some methods make per-point predictions, e.g. they predict a semantic label for each point in the input point cloud (PartNet, BAE-NET, LEL). For a fair comparison against NGSP and LHSS, which assign labels to shape region, for methods appended by (R) we group per-atom predictions into per-region predictions. To do this, we first compute the probability distribution over labels for each point (e.g. send the logits through a softmax). Then we group points based on the shape regions, and for each shape region we find the region label probability distribution by average all of the per-point label probability distributions. We then take the arg max of the region label probability distribution as the chosen label assignment for that shape region.

### D.5. Unlabeled Additional Shape Instances

Some comparison methods (BAE-NET and LEL) are able to use shape instances that lack label annotations, but contain shape region decompositions. For fair comparison, we allow these methods to train on up to 1000 additional shapes where the shape region decomposition is provided, but the semantic label annotations are withheld. We source these unlabeled shapes from PartNet instances that do not show up in the labeled training, validation or test sets. For some number of labeled training data + category combinations, there are not enough shape instances in PartNet to reach 1000, in which case we use as many shapes as there are available. Table 2 contains how many unlabeled shape instances are used by BAE-NET and LEL for each number of labeled training data + category combination.

| # Train | Chair | Lamp | Table | Storage | Vase | Knife |
|---|---|---|---|---|---|---|
| 10 Labeled | 1000 | 1000 | 1000 | 1000 | 1000 | 404 |
| 40 Labeled | 1000 | 1000 | 1000 | 1000 | 1000 | 374 |
| 400 Labeled | 1000 | 1000 | 1000 | 1000 | 652 | 175 |

Table 2. Number of additional unlabeled shape instances used by BAE-Net and LEL comparison methods, for each category, and each number of labeled training data used during training.

## E. Inference Run-Time

As demonstrated, NGSP outperforms comparison methods on the task of semantic segmentation. This performance improvement comes at the cost of an increase in the time it takes to produce semantic segmentations; NGSP does not operate in an end-to-end fashion, but rather performs approximate MAP inference. This search is directed by the neural guide network, which proposes a constrained set of label assignments that are then considered under the full likelihood model; evaluating more proposals will result in a better MAP estimate, but incurs more computation time. This trade-off is controlled with the hyper-parameter $k$, the number of proposals generated from the guide network. When $k$ is set to 1, the performance is equivalent to just using the guide network (from the ablation table main paper). The time it takes NGSP to generate a semantic segmentation for an average chair is 0.2 seconds when $k = 10$, 0.3 seconds when $k = 100$, 0.8 seconds when $k = 1000$, and 4 seconds when $k = 10000$. For all results, we set $k$ to 10000, allowing us

to well-approximate the MAP, while keeping computational time manageable.

## F. Additional Qualitative Results

We present additional qualitative results comparing different methods on the task of fine-grained semantic segmentation of Partnet shapes for Chairs (Figure 3), Tables (Figure 4), Lamps (Figure 5), Vases (Figure 6), Knives (Figure 7) and Storage Furniture (Figure 8).

Each figure additionally contains the semantic grammar we use. All shape grammars we use are derived from the hierarchies defined by PartNet. In most cases these labels corresponds to the *level-2* granularity in PartNet. For some shapes, where level-2 was not defined, level-3 was substituted. For all terminal labels that are present in the depicted qualitative examples, we color the background text of the terminal label to match the color of semantic part in the qualitative renders. The input region column is given purely random colors, such that each shape region is given a unique color.

## References


[1] Gavin Barill, Neil Dickson, Ryan Schmidt, David I.W. Levin, and Alec Jacobson. Fast winding numbers for soups and clouds. *ACM Transactions on Graphics*, 2018. 4

[2] Yuri Boykov, Olga Veksler, and Ramin Zabih. Fast approximate energy minimization via graph cuts. *IEEE Trans. Pattern Anal. Mach. Intell.*, 23(11):1222–1239, nov 2001. 5

[3] Xavier Bresson and Thomas Laurent. Residual gated graph convnets. *arXiv preprint arXiv:1711.07553*, 2017. 2

[4] Angel X. Chang, Thomas Funkhouser, Leonidas Guibas, Pat Hanrahan, Qixing Huang, Zimo Li, Silvio Savarese, Manolis Savva, Shuran Song, Hao Su, Jianxiong Xiao, Li Yi, and Fisher Yu. ShapeNet: An Information-Rich 3D Model Repository. *arXiv:1512.03012*, 2015. 3, 4

[5] Zhiqin Chen, Kangxue Yin, Matthew Fisher, Siddhartha Chaudhuri, and Hao Zhang. Bae-net: Branched autoencoder for shape co-segmentation. *Proceedings of International Conference on Computer Vision (ICCV)*, 2019. 4

[6] Vijay Prakash Dwivedi, Chaitanya K Joshi, Thomas Laurent, Yoshua Bengio, and Xavier Bresson. Benchmarking graph neural networks. *arXiv preprint arXiv:2003.00982*, 2020. 2

[7] Matheus Gadelha, Aruni RoyChowdhury, Gopal Sharma, Evangelos Kalogerakis, Liangliang Cao, Erik Learned-Miller, Rui Wang, and Subhransu Maji. Label-efficient learning on point clouds using approximate convex decompositions. In *European Conference on Computer Vision (ECCV)*, 2020. 4

[8] Jingwei Huang, Hao Su, and Leonidas Guibas. Robust watertight manifold surface generation method for shapenet models. *arXiv preprint arXiv:1802.01698*, 2018. 4

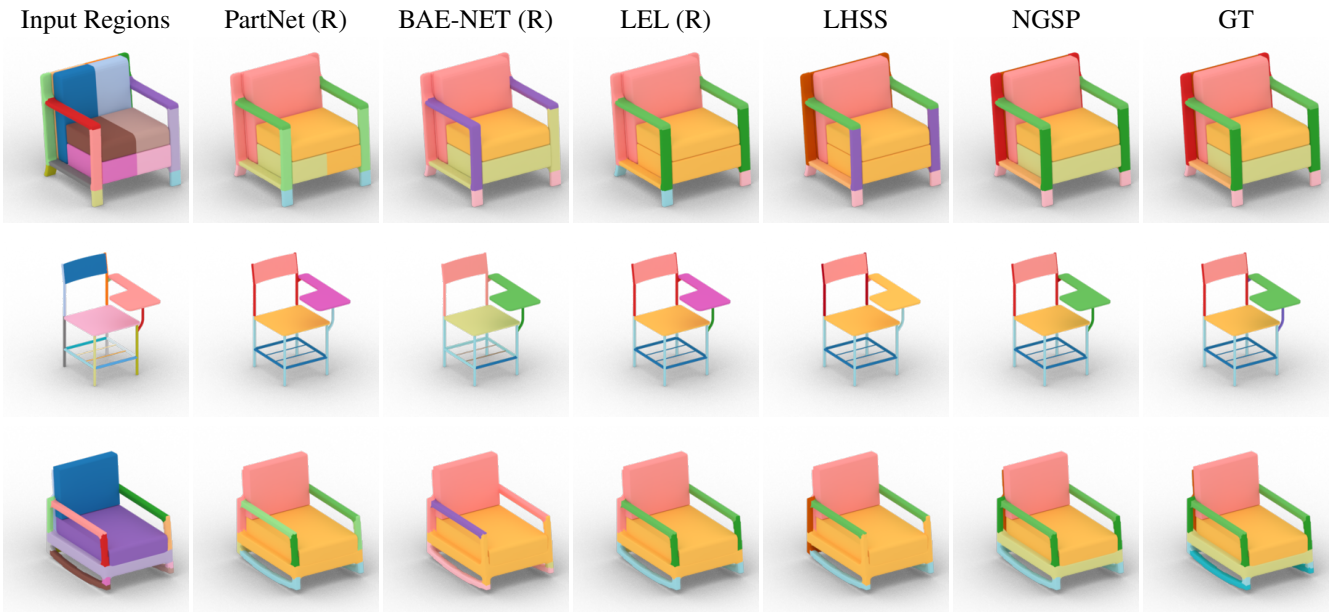
[9] Kaichun Mo, Shilin Zhu, Angel X. Chang, Li Yi, Subarna Tripathi, Leonidas J. Guibas, and Hao Su. PartNet: A large-scale benchmark for fine-grained and hierarchical part-level 3D object understanding. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019. 1

[10] Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. Automatic differentiation in pytorch. 2017. 1

[11] 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

[12] Erik Wijmans. Pointnet++ pytorch. `https://github.com/erikwijmans/Pointnet2_PyTorch`, 2018. 1

[13] Li Yi, Leonidas Guibas, Aaron Hertzmann, Vladimir G. Kim, Hao Su, and Ersin Yumer. Learning hierarchical shape segmentation and labeling from online repositories. *SIGGRAPH*, 2017. 4




Chair → arm; back; base; head; seat; footrest
Arm → connector; holistic frame; horizontal bar; near vertical bar; sofa style; writing table
Back → connector; frame; support; surface
Base → foot base; pedestal base; regular leg base; star leg base
Head → connector; headrest
Seat → frame; support; surface
Footrest → base; footrest seat
Foot Base → foot
Pedestal Base → central support; pedestal
Regular Leg Base → bar stretcher; foot; leg; rocker; runner
Star Leg Base → central support; mechanical control; star leg set
Footrest Seat → support ; surface

Figure 3. Qualitative Results and Grammar for the Chair category.

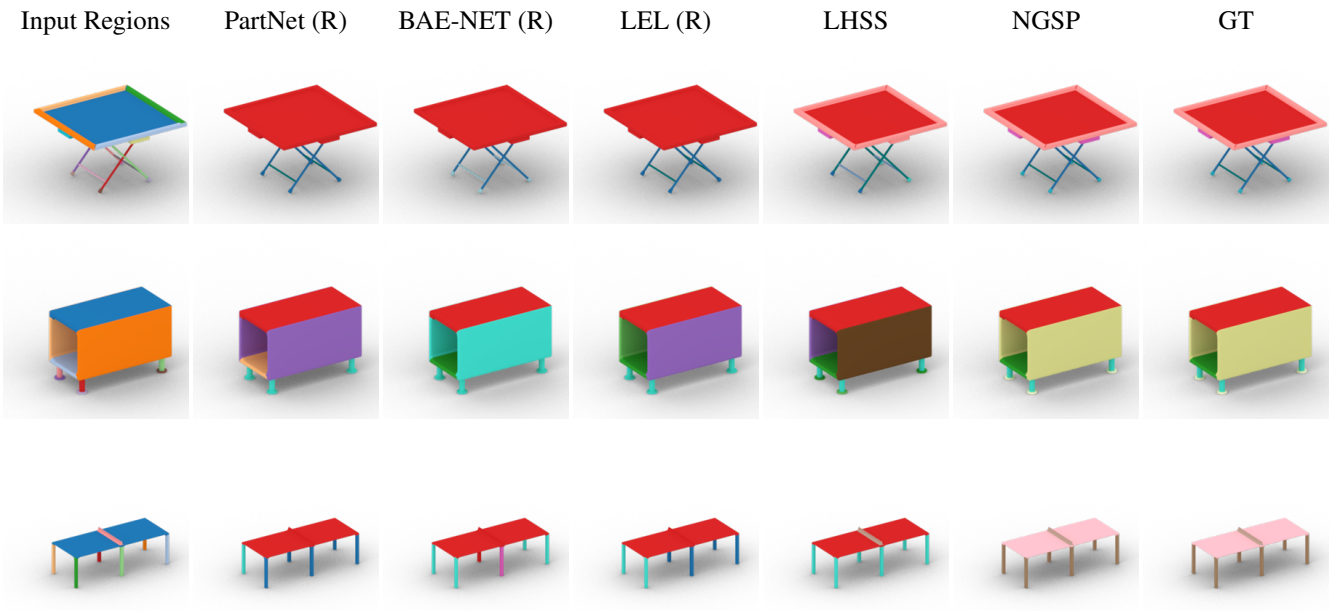
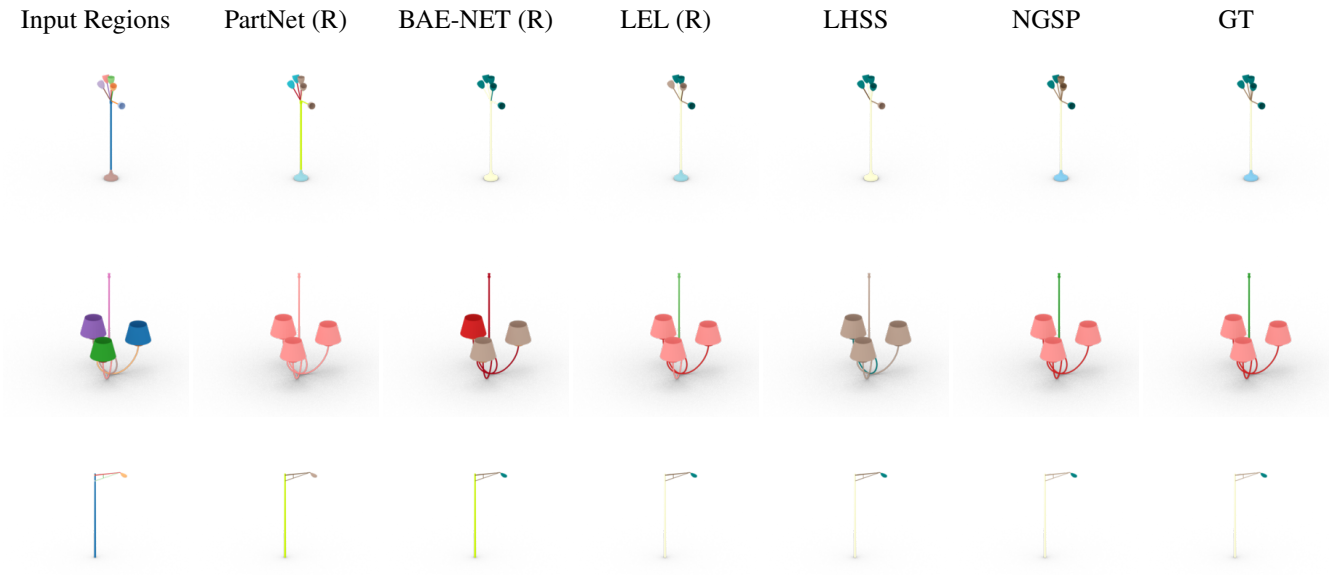


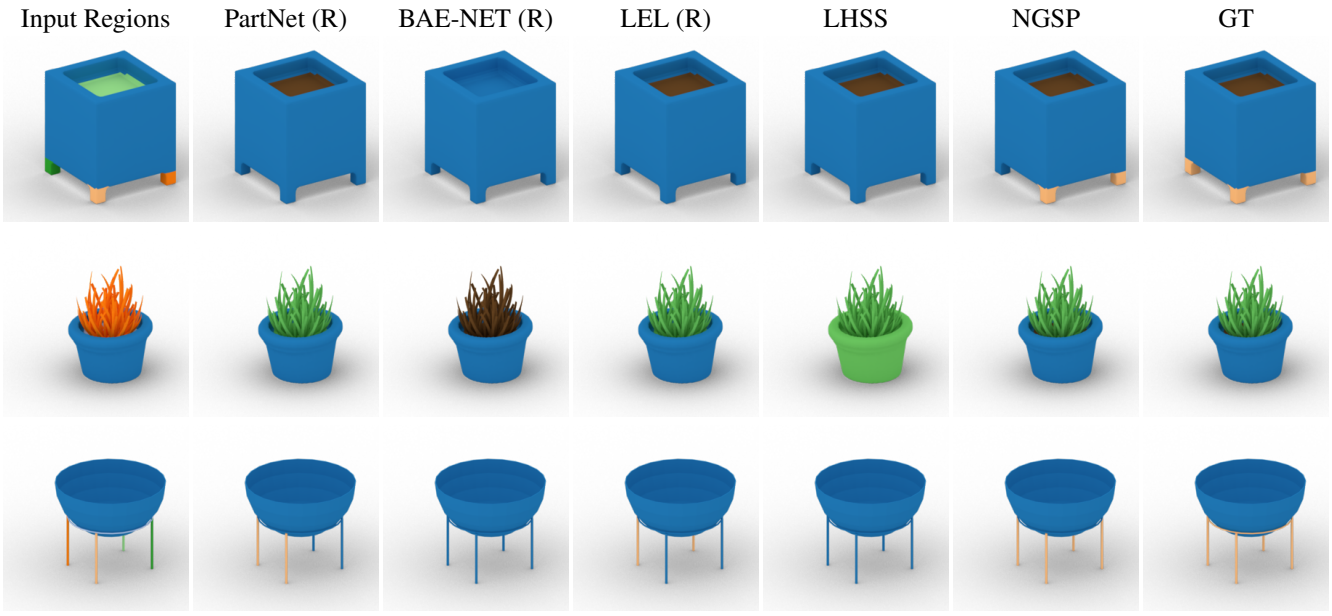

Table → game table; picnic table; regular table
Game table → ping pong table; pool table
Picnic table → bench; bench connector; regular picnic table
Regular table → regular base; regular tabletop
Ping pong table → net; ping pong base; ping pong tabletop
Pool table → ball; pool base; pool tabletop
Regular picnic table → picnic base; picnic tabletop
Regular base → drawer base; pedestal base; regular leg base; star leg base;
Regular tabletop → dropleaf; frame; surface
Ping pong base → ping pong regular leg base
Ping pong tabletop → ping pong surface
Pool base → pool regular leg base
Pool tabletop → frame; surface
Picnic base → picnic regular leg base
Picnic tabletop → surface
Drawer base → back panel; bar stretcher; bottom panel; cabinet door; caster; drawer; foot; keyboard tray; shelf; leg; tabletop connector; vertical divider panel; vertical front panel; vertical side panel
Pedestal base → central support; pedestal; tabletop connector
Regular leg base → bar stretcher; caster; circular stretcher; foot; leg; runner; tabletop connector
Star leg base → central support; star leg set
Ping pong regular leg base → bar stretcher; leg
Picnic regular leg base → leg
Pool regular leg base → leg

Figure 4. Qualitative Results and Grammar for the Table category.



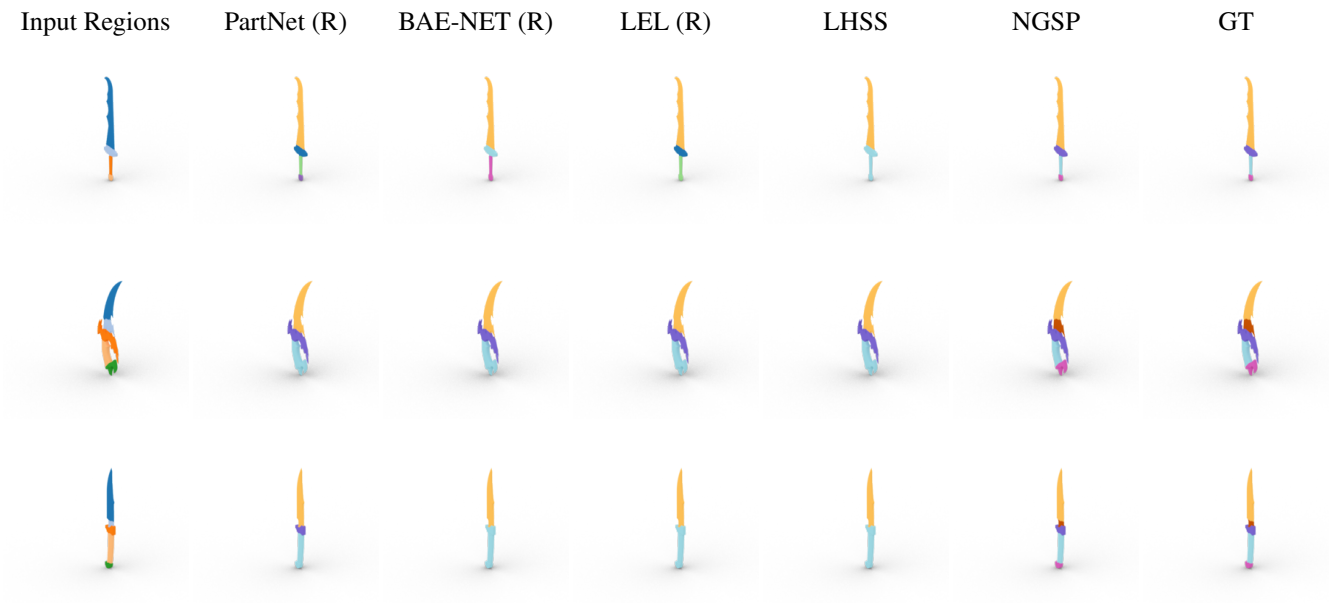

Lamp → ceiling lamp; street lamp; table or floor lamp; wall lamp
Ceiling lamp → chandelier; pendant lamp
Street lamp → post ; street unit; street base
Table or floor lamp → ToF base; ToF body; ToF unit; ToF power cord
Wall lamp → wall base; body; wall unit
Chandelier → chain; chandelier base; body ; chandelier unit group
Pendant lamp → pendant base; pendant unit; pendant power cord
Street unit → arm; head
ToF base → ToF holistic base; ToF leg base
ToF body → jointed ; solid ; pole ; vertical panel
ToF unit → connector; arm; head
ToF power cord → cord
Wall base → wall holistic base
Wall unit → arm; head
Chandelier base → chandelier holistic base
Chandelier unit group → chandelier unit
Pendant base → pendant holistic base
Pendant unit → chain; head
Pendant power cord → cord
ToF holistic base → base part
ToF leg base → leg
Wall holistic base → base part
Chandelier holistic base → base part
Chandelier unit → arm; head
Pendant holistic base → base part

Figure 5. Qualitative Results and Grammar for the Lamp category.



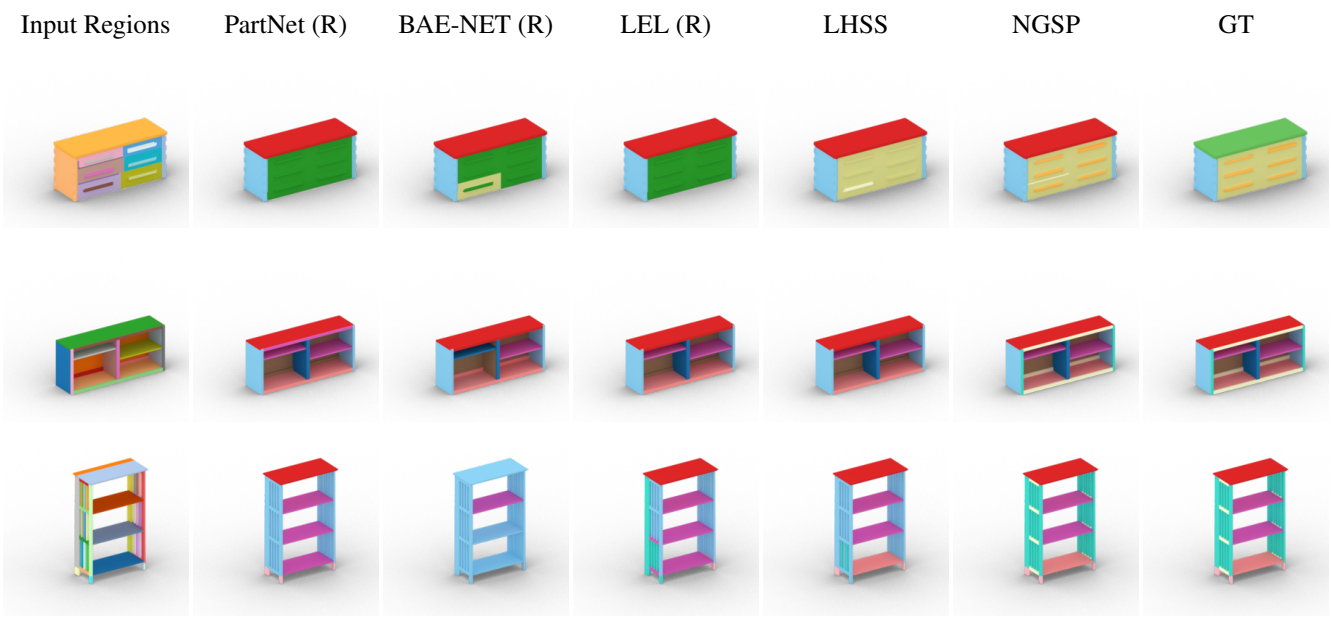

Vase → base; body; containing things
Base → foot base
Body → container; lid
Containing things → liquid or soil; plant
Foot base foot

Figure 6. Qualitative Results and Grammar for the Vase category.




Knife → dagger; cutting instrument
Dagger → dagger blade side; dagger handle side
Cutting instrument → cutting instrument blade side; cutting instrument handle side
Dagger blade side → blade
Dagger handle side → butt ; guard ; handle
Cutting instrument blade side → blade ; bolster
Cutting instrument handle side → butt ; guard ; handle

Figure 7. Qualitative Results and Grammar for the Knife category.




Storage Furniture → base; door; frame; countertop; drawer; shelf
Base → foot base; panel base
Frame → back panel; bottom panel; horizontal bar; vertical bar; top panel; vertical divider panel; vertical front panel; vertical side panel
Drawer → drawer box; handle
Drawer Box → back; bottom; front; side

Figure 8. Qualitative Results and Grammar for the Storage Furniture category.