SHRED: 3D Shape Region Decomposition with Learned Local Operations

R. KENNY JONES, Brown University, USA
AALIA HABIB, Brown University, USA
DANIEL RITCHIE, Brown University, USA

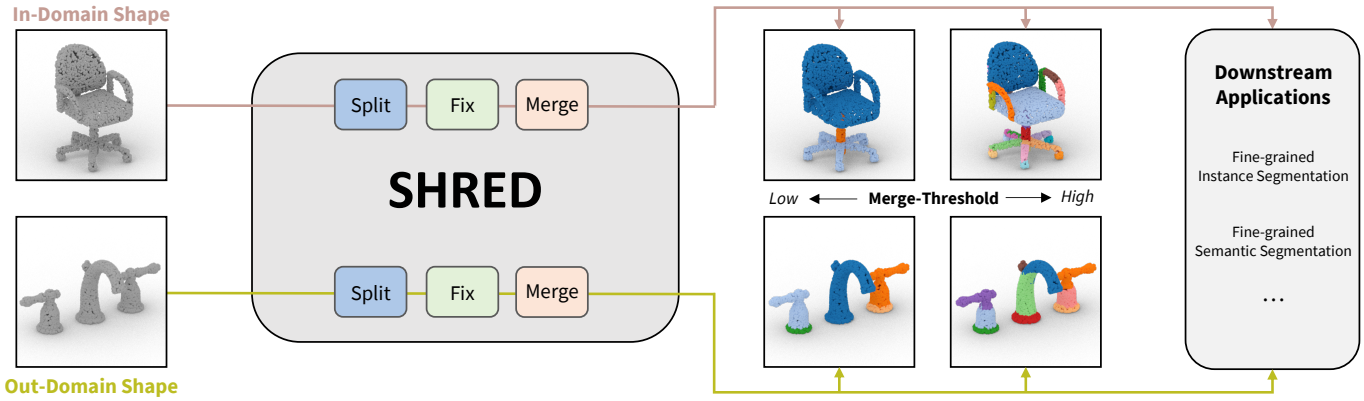

Fig. 1. SHRED decomposes 3D shapes into regions by learning to locally split, fix, and merge segments. A merge-threshold parameter can be adjusted to change decomposition granularity depending on the target downstream application.

We present SHRED, a method for 3D SHape REgion Decomposition. SHRED takes a 3D point cloud as input and uses learned local operations to produce a segmentation that approximates fine-grained part instances. We endow SHRED with three decomposition operations: splitting regions, fixing the boundaries between regions, and merging regions together. Modules are trained independently and locally, allowing SHRED to generate high-quality segmentations for categories not seen during training. We train and evaluate SHRED with fine-grained segmentations from PartNet; using its merge-threshold hyperparameter, we show that SHRED produces segmentations that better respect ground-truth annotations compared with baseline methods, at any desired decomposition granularity. Finally, we demonstrate that SHRED is useful for downstream applications, out-performing all baselines on zero-shot fine-grained part instance segmentation and few-shot fine-grained semantic segmentation when combined with methods that learn to label shape regions.

CCS Concepts: • **Computing methodologies** → Shape analysis; Neural networks.

Additional Key Words and Phrases: shape analysis, shape segmentation, fine-grained components

ACM Reference Format:
R. Kenny Jones, Aalia Habib, and Daniel Ritchie. 2022. SHRED: 3D Shape Region Decomposition with Learned Local Operations. *ACM Trans. Graph.* 41, 6, Article 186 (December 2022), 11 pages. https://doi.org/10.1145/3550454.3555440

Authors' addresses: R. Kenny Jones, russell_jones@brown.edu, Brown University, USA; Aalia Habib, aalia_habib@brown.edu, Brown University, USA; Daniel Ritchie, daniel_ritchie@brown.edu, Brown University, USA.

1 INTRODUCTION

3D segmentation is a fundamental problem within computer graphics and vision. Many applications in these fields require methods capable of decomposing 3D shapes and scenes into meaningful regions: establishing correspondences, skeleton extraction, guiding semantic labeling, and allowing autonomous agents to interact with objects, to name a few [Abbatematteo et al. 2019; Landrieu and Simonovsky 2018; Liu and Zhang 2004].

While there is a long history of computer graphics research using geometric and visual cues to segment 3D shapes into regions, most recent work has explored the use of data-driven neural methods. Learning-based approaches have produced impressive results on coarse decomposition tasks where training annotations are plentiful, but often struggle when either of these conditions is not met. Most methods that operate within this paradigm learn to segment shapes globally, which contributes to both of these limitations. While coarse decompositions are often fairly globally consistent, fine-grained decomposition are highly varied, even within the same category (e.g. chairs almost always have a back, a seat, and a base, but only a small subset of chairs have wheel casters). Moreover, these methods are often unable to produce sensible segmentations for shapes outside of their training distribution, as their networks specialize to the global patterns of training shapes.

Some recent approaches have aimed to develop learning-based systems capable of producing fine-grained segmentations in a way that generalizes across domains. Within this paradigm, the hope is to train networks on categories of shapes with abundant annotations, and then show these networks have learned decomposition policies that can be directly applied to shapes from categories that lack annotations. Many of these methods use at least one network that

reasons locally, with the idea that forcing the network to ignore global context might encourage better generalization on out-domain shapes [Luo et al. 2020; Wang et al. 2021a].

Following this general framework, SHRED learns a collection of region decomposition operations that all reason in a local fashion: splitting a region into sub-regions, fixing the boundaries between regions, and deciding when neighboring regions should be merged together. Each of these operations is modeled with a neural module that operates over regions, represented as point-clouds, and trained with ground-truth part annotations from a dataset of shapes. The modules of SHRED are applied sequentially to generate a region decomposition. An input shape first undergoes a naive decomposition using farthest-point sampling, then these regions are passed through the split, fix and merge operators to produce the final segmentation. Of note, the merge operator exposes a merge-threshold hyperparameter that can be toggled to control the granularity of the output decomposition.

We train a version of SHRED on three data abundant categories of PartNet: chairs, lamps and storage furniture (*in-domain* shapes), and evaluate its ability to produce region decompositions for test-set in-domain and out-domain shapes. We compare SHRED against other shape segmentation methods, including learned and non-learned approaches, that operate both globally and locally. We find that SHRED produces better region decompositions than baseline methods, for both in-domain and out-domain categories. Analyzing the trade-off between decomposition quality and granularity, we vary SHRED's merge-threshold to create a Pareto frontier of solutions that strictly dominates all comparisons. We then demonstrate that SHRED's decompositions can be treated as fine-grained part instance segmentations that outperform comparison methods. Finally, we evaluate how SHRED can improve fine-grained semantic segmentation, using the output of SHRED as the input to a method that learns to semantically label shape regions, and find that using SHRED results in the best performance.

In summary, our contributions are:

(i) SHRED, a method for 3D SHape REgion Decomposition with learned local split, fix and merge operations.
(ii) Demonstrations on collections of manufactured shapes that SHRED outperforms baseline methods in terms of fine-grained segmentation performance and finding better trade-offs between decomposition quality and granularity, for both in-domain and out-domain categories.

Code for our method and experiments can be found at found at https://github.com/rkjones4/SHRED .

2 RELATED WORK

Shape segmentation with geometric cues. There is a long history of computer graphics research dedicated to segmenting a 3D shape into meaningful regions according to geometric properties. Most such methods are not data-driven, but rather analyze heuristic properties of 3D meshes to produce shape decompositions useful for various downstream applications. These include approaches such as: normalized cuts, symmetry cues [Wang et al. 2011], fuzzy clustering [Katz and Tal 2003], spectral methods [Asafi et al. 2013; Liu and Zhang 2004], and approximate convex decomposition [Kaick et al. 2014; Lien and Amato 2008]. Please refer to [Shamir 2008] for a survey on the topic.

Learning to approximate shapes with primitives. A great body of recent research has been devoted to learning methods that aim to coarsely approximate 3D shapes with a union of primitive structures. Methods differ by the type of primitive used, for instance cuboids [Sun et al. 2019; Tulsiani et al. 2017; Yang and Chen 2021], superquadrics [Paschalidou et al. 2019], convex solids [Chen et al. 2019a; Deng et al. 2020], and more general local neural functions [Chen et al. 2019b; Genova et al. 2019; Kawana et al. 2020; Paschalidou et al. 2021]. These methods can train on 3D shapes that lack annotations and produce segmentations with paired correspondences across different shape instances. Without annotations, these approaches rely on global reconstruction-based losses, resulting in decompositions that well-represent coarse structures, but often ignore fine-grained regions of interest. Relatedly, some methods use primitive decompositions to formulate self-supervised losses that augment training to improve few-shot semantic labeling [Gadelha et al. 2020; Sharma et al. 2021].

Shape decomposition with supervised learning. Most learning-based shape decomposition methods operate within a supervised learning paradigm by consuming a training dataset that contains annotated regions. Many of these works learn to globally decompose a shape in a category specific fashion for a particular type of region-annotation (e.g. part instances) [Qi et al. 2017a,b; Wang et al. 2018, 2019; Yi et al. 2019; Yu et al. 2019]. These approaches achieve state-of-the-art performance when the desired decomposition is coarse and training data is plentiful for the category of interest, but do not perform as well with fine-grained segmentations or out-of-domain inputs.

Recent work has tried to address these issues by developing learning-based solutions that work locally. For instance, [Han et al. 2020] propose a pipeline for single-image shape reconstruction via cuboid proxies that can generalize to out-of-domain shapes. Some methods have been designed with fine-grained regions in mind. [Wang et al. 2021a] use a learned clustering approach to perform local split operations, and then aggregate sub-regions with a global merge step. [Luo et al. 2020] use policy gradient reinforcement learning to train a network that takes an over-segmented 3D shape and merges segments together to approximate a part instance segmentation. SHRED shares similar motivations to these last two methods: learn local operations on shape categories with abundant annotations, and then show this promotes strong generalization capabilities on novel shape categories. We compare SHRED against these approaches, and find that SHRED's collection of learned local operations improves the quality of its region decompositions.

Related to this problem of zero-shot generalization, some methods frame shape region decomposition as a co-segmentation problem, such as AdaCoSeg [Zhu et al. 2020]. In this paradigm, a single labeled exemplar is provided as template to indicate how the rest of the distribution should be decomposed. The fix operator of SHRED takes inspiration from the part prior network in AdaCoSeg.

Learning to semantically label. Some applications desire region decompositions where part instances are grouped by semantic properties. Learning-based methods that perform global category specific

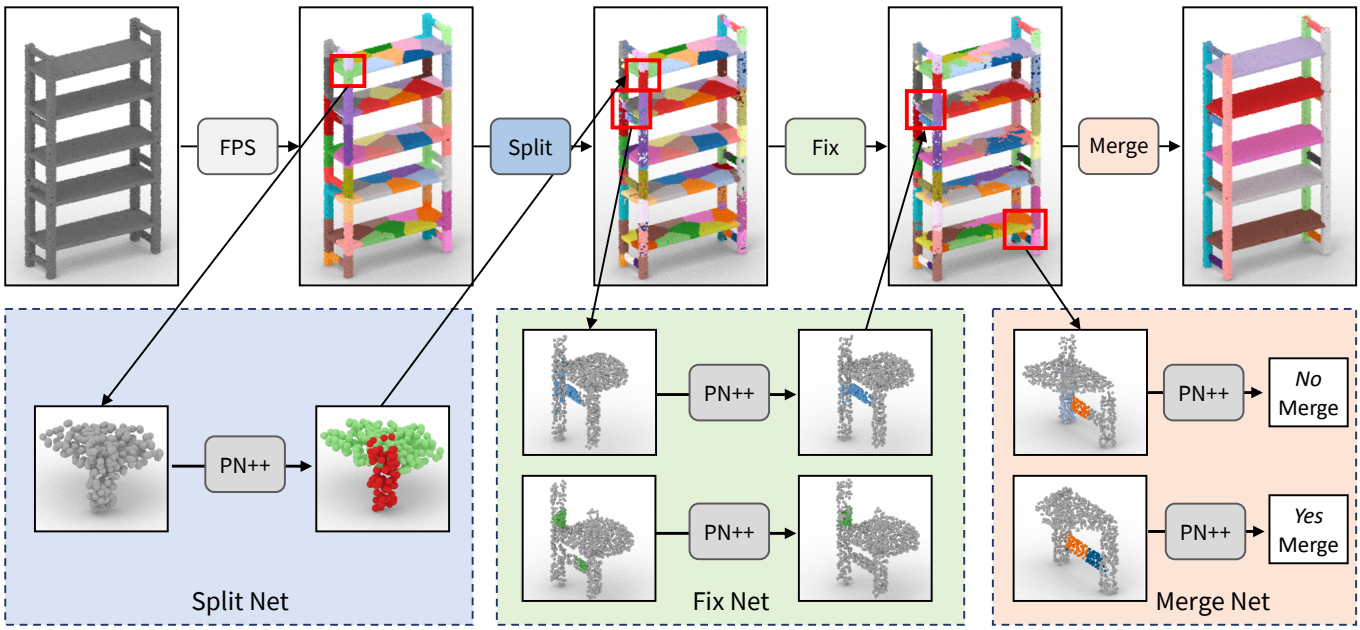

Fig. 2. The modules of SHRED. From left to right, input shapes are naively decomposed by farthest-point sampling (FPS), regions are split into sub-regions, boundaries are fixed, and neighbors are merged together. Bottom-row cut-outs visualize network input-outputs.

segmentations can be used for this task, although they struggle with fine-grained semantic decompositions or paradigms when labeled data is limited. Recent approaches have investigated how to address this latter problem for coarse semantic labels. When the number of shapes that contain any labels at all is limited, methods have investigated how to improve few-shot segmentation performance by incorporating self-supervised training objectives over large amounts of unlabeled data [Chen et al. 2019b; Gadelha et al. 2020; Sharma et al. 2019; Sun et al. 2022; Wang et al. 2021b; Xie et al. 2020], learning to morph shapes into matched templates from a small labeled collection [Wang et al. 2020], and framing few-shot segmentation under a meta-leaning paradigm [Hao and Fang 2021; Huang et al. 2021]. Related methods have focused on improving segmentation performance when the number of labels provided for each shape is limited [Liu et al. 2021; Xu and Lee 2020].

Another line of investigation has looked into approaches that learn to assign semantic labels to regions of a 3D shape [Jones et al. 2022; Yi et al. 2017]. When a good region decomposition is provided, these methods have been shown to outperform global semantic segmentation approaches, especially for fine-grained semantic labels and when labeled data is limited. We will show that the region decompositions created by SHRED can be combined with these approaches to improve few-shot fine-grained semantic segmentation performance.

3 METHOD

SHRED takes a 3D shape as input and outputs a fine-grained region decomposition. We define a region decomposition for shape S as a set of regions $R = \{r_0, r_1, ..., r_n\}$ s.t $S = \bigcup_{i \in N} r_i$. Given a ground-truth region decomposition R^* (e.g. fine-grained part instance annotations produced by a human), we desire two properties of R. First, each $r_i \in R$ should be a subset of some $r_j^* \in R^*$: $r_i \subseteq r_j^*$. Second, the number of regions in R should not exceed the number of regions in R^*: $|R| \leq |R^*|$. The first property states a desire that regions in R do not cross the part boundaries defined by R^*, while the second property states that the granularity of R's decomposition should not exceed the granularity of R^*'s decomposition. Notice that these properties work against each other; decreasing $\|R\|$ increases the probability that some $r_i \in R$ is no longer a subset of some region of R^*, while ensuring that $r_i \subseteq r_j^* \; \forall \; i \in N$ requires that $|R| \geq |R^*|$. These two properties are only simultaneously met when $R = R^*$, but we can use them to evaluate the goodness of any given R.

We provide a visual overview of our method in Figure 2. SHRED contains three learned modules that perform region modifications: a split module that splits a region (Section 3.1, blue box) , a fix module that fixes part boundaries between regions (Section 3.2, green box), and a merge module that decides when neighboring regions should be merged together (Section 3.3, orange box). As we are interested in using SHRED to generate decompositions for out-of-distribution shapes, we design each module to operate on only region-local information. This encourages the operations to learn region-specific principals that generalize better than category-specific patterns [Luo et al. 2020]. Following [Wang et al. 2021a], we represent 3D shapes as high-resolution point-clouds (100k points) sampled from a mesh-surface, in order to capture fine-grained geometry. SHRED first applies a naive strategy to decompose the shape into noisy regions, and then sequentially applies the split, fix, and merge modules to produce a high-quality part-respecting region decomposition.

In all experiments, we create the naive region decomposition with a simple farthest-point sampling (FPS) procedure with K centroids (default $K = 64$). The split, fix and merge networks are all variants of PointNet++'s written in PyTorch [Paszke et al. 2017; Qi et al. 2017b; Wijmans 2018]. Further network training details and hyper-parameters are provided in the supplemental material.

3.1 Split Module

The split module is tasked with deciding if a region should be further decomposed into multiple sub-regions. In SHRED, the split operation is applied over the naive region decompositions produced by FPS clustering, as each FPS-produced region might be an under-segmentation with respect to $R*$. We model the split operation with an instance segmentation network that considers each region individually; it consumes a point cloud representing a region and predicts an instance label for each point of the region.

Model Details. The split network uses a PointNet++ instance segmentation back-bone, with a per-point MLP head that predicts into 10 maximum part slots. The back-bone uses 4 set abstraction layers with 1024, 256, 64, 16 grouping points, 0.1, 0.2, 0.4, 0.8 radius size and 64, 128, 256, 512 feature size respectively. A series of 4 per-point feature propagation modules converts each 512 feature to size 128. Batch normalization and ReLU activations are used throughout. A MLP head for per-point predictions uses a hidden layer with dimension of 64 to produce a prediction into 10 slots. The MLP uses ReLU activations and a 0.1 dropout.

The input region point clouds have 6 features (xyz positions and normals), are sub-sampled to 512 points, and are normalized to the unit-sphere. The split network is trained with cross-entropy loss, which requires finding an alignment between predicted instance slots and the target instance slots. We use a variant of the Hungarian matching algorithm of Mo et al. [2019] to dynamically find the best alignment during training. As we want the split network to remove all under-segmentation, our matching algorithm greedily encourages over-segmentations that better respect part boundaries, at the cost of using more prediction slots. We further describe this matching in the supplemental material.

Data Preparation. Training examples for the split network are sourced from part instance labeled training shapes. For each shape, we produce a naive region decomposition using FPS clustering. Then, each region in the naive decomposition contributes one input point cloud for training, where ground-truth target instances are supplied by the fine-grained annotated labels.

3.2 Fix Module

The fix module is responsible for improving region boundaries. It consumes the region decomposition output by the split module. This decomposition is fine-grained, but might contain errors on part boundaries, as the split network has no information about the shape outside of each region. While the fix module also operates over regions, unlike the split module, it also receives nearby points from outside the region to help to contextualize the region within the shape. The input to the fix network is formed by a concatenation of two points clouds, one coming from the region of interest, one coming from outside the region of interest (colored versus grey points in Figure 2, bottom-middle). The fix network then makes a binary prediction for each input point, deciding whether or not it should be inside or outside the region of interest. SHRED applies this inside-outside prediction to the local neighborhood of points around each region, and these per-region decisions are propagated into a global region decomposition through an argmax operation.

Model Details. The fix network uses a PointNet++ instance segmentation back-bone, with a per-point MLP that predicts a binary logit. The back-bone is the same as the one for the split network, except that batch norm is not used. A MLP head for per-point predictions uses 2 hidden layers with dimensions of 64 and 32 to produce a binary prediction logit. The MLP uses ReLU activations and a 0.1 dropout.

Input point clouds are made up of 2048 inside points and 2048 outside points, where each point has 7 features (xyz positions, normals, inside-outside flag), and are normalized to the unit-sphere. The network is trained with Binary Cross Entropy loss, where 1.0 (0.0) indicates the point is inside (outside) the region of interest.

Data Preparation. We employ a synthetic perturbation process to generate training data for the fix network. To generate a training example, we first sample a random ground-truth (GT) part instance from a random training set shape. We then corrupt this region by randomly adding outside points into the region or removing points from the region. Inside points from this corrupted region and nearby outside points (within 0.1 radius) are then combined to create an input point cloud. Finally, we take the corrupted region, find the GT region it has the highest overlap with, and use the inside-outside values of this best-matching GT region to produce the target labels for the sampled points of the training example.

3.3 Merge Module

The merge module decides when neighboring regions should be combined together. It consumes the output of the fix module, so typically the region decomposition contains little under-segmentation but has $|R| >> |R^*|$. Within our problem formulation, two regions should be merged if the GT region they share the most overlap with is the same: merging these regions would decrease $|R|$ without increasing the level of under-segmentation. The merge network learns to solve this problem by locally operating over pairs of neighboring regions, r_i and r_j, and making binary predictions whether or not the two regions should be merged together. It receives a point cloud as input, where 1/4 of the points come from r_i, 1/4 of the points come from r_j, and 1/2 come from nearby outside points within a 0.1 radius (Figure 2, bottom-right).

Within the merging module, the merge network is applied in an iterative procedure. Each round of this procedure begins by finding all neighboring regions (using region-to-region minimum point distance) and adding them into a queue to be evaluated by the merge network. The merge network then predicts a merge probability for each pair of regions and sorts these predictions from most-to-least confident. SHRED then iterates through these predictions, merging together neighboring regions if their merge probability is above a user-defined merge-threshold; once a region has been merged, it

cannot be further merged in the same round. The procedure repeats until a round ends with no merges.

The merge-threshold, MT, defines the granularity of the output decomposition. By default, we set MT to 0.5, which encourages decompositions to match the granularity of R^* from the training distribution. When set to 1, no regions are merged from the output of the fix module; when set to 0, all regions are combined together. The merge-threshold can be used to explore tradeoffs between region granularity and region under-segmentation, as seen in Figure 1.

Model Details. We model the merge network with a PointNet++ classification back-bone, with a MLP head that predicts a binary logit. The back-bone uses 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. Batch normalization and ReLU activations are used throughout. A MLP head for per-shape predictions uses 2 hidden layers with dimensions of 256 and 64.

The input point clouds contain 2048 points, with 8 features (xyz position, normal, one hot flags for region i and region j), and are normalized to the unit-sphere. The network is trained with Binary Cross Entropy loss, where 1.0 (0.0) indicates the regions should (should not) be merged together.

Data Preparation. Training data for the merge network is produced with a synthetic perturbation procedure. This procedure begins by sampling a shape from our training set. It then decomposes the shape into regions using our FPS procedure. Each region in this segmentation then undergoes further decomposition in an annotation-aware fashion. For every region, each GT part instance in that region is randomly split into sub-parts. After this split, each sub-part is then randomly assigned to either its own region or grouped with other sub-parts in its parent region. We empirically find this process creates decompositions that broadly cover the distribution of outputs created by the split and fix modules.

After we have created this region decomposition, we find all neighboring regions and begin sampling merges between neighbors at random. We record each sampled merge as a training example. Points from the two neighboring regions, r_i and r_j, and nearby points outside both regions, form the input point cloud. The label of the merge is determined by checking the parts in R^* that have the best match with the sampled regions: if r_i and r_j best match to the same r_k^* part, then the merge should happen, otherwise the merge should not happen. While sampling merges, we also update the region decomposition. If the merge should happen, we execute the merge with 75% chance, while if the merge should not happen, we execute the merge with 25% chance. This process repeats until all remaining neighbors have been considered.

4 EXPERIMENTS

We evaluate SHRED's ability to produce high-quality region decompositions using fine-grained part annotations from the PartNet dataset (Section 4.1). We describe our training procedure in Section 4.2. In Section 4.3, we compare SHRED against related methods, analyzing the trade-off between decomposition quality and decomposition granularity. We then examine how SHRED can be used

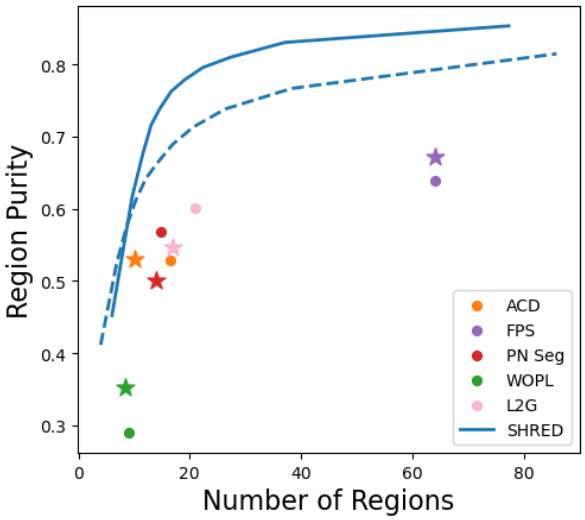

Fig. 3. Comparing segmentation granularity (X-axis, lower is better) and quality (Y-axis, higher is better). In(out)-domain averages are shown with solid (dotted) lines and circles (stars).

to improve performance on downstream tasks such as zero-shot fine-grained part instance segmentation (Section 4.4) and few-shot fine-grained semantic segmentation (Section 4.5). Finally, we analyze the importance of SHRED's various components through a series of ablation studies in Section 4.6.

4.1 Dataset Details

We use data from the finest-grained part annotations of PartNet [Mo et al. 2019] to train and evaluate our method. We train SHRED on a subset of categories with abundant data (in-domain), and show that SHRED is able to learn patterns that generalize well to novel categories (out-domain) at test time. Our in-domain categories are: chair, lamp, and storage. Our out-domain categories are: bed, display, earphone, faucet, knife, refrigerator and table. We evenly sample shapes from our in-domain categories to form train/validation/test splits of 6000/600/600 shapes, while each out-domain category contributes ~200 shapes to the test set.

4.2 Training Details

SHRED requires training 3 networks: the split network, the fix network, and the merge network. We use the Adam optimizer [Kingma and Ba 2014] with learning rates of $1e^{-3}$, $1e^{-4}$, $1e^{-4}$ and batch sizes of 64, 64, 128 for the split, fix, and merge networks respectively. For the merge network, a learning rate scheduler is employed to drop the learning by a factor of 0.25 every time the train loss does not reach a new minimum over a patience of 10000 iterations. Training is performed on a machine with a GeForce RTX 3090 Ti GPU and an Intel i7-11700K CPU. The split network trained for 70 epochs (18 hours), the fix network trained for 200 epochs (18 hours), and the

Table 1. Fine-grained instance segmentation performance on in-domain (left) and out domain (right) test-set shapes (metric is AIoU). SHRED outperforms all baseline methods, and can be further improved by setting the merge-threshold to 0.8.

	In Domain				Out Domain							
Method	*Avg*	*Chair*	*Lamp*	*Storage*	*Avg*	*Bed*	*Display*	*Earphone*	*Faucet*	*Knife*	*Fridge*	*Table*
FPS	0.237	0.278	0.245	0.186	0.173	0.199	0.115	0.162	0.202	0.128	0.151	0.255
WOPL	0.178	0.173	0.248	0.114	0.235	0.074	0.234	0.267	0.290	0.461	0.134	0.181
PN Seg	0.377	0.381	0.361	0.389	0.318	0.167	0.427	0.274	0.306	0.396	0.253	0.404
L2G	0.425	0.456	0.477	0.342	0.392	0.253	0.466	0.366	0.451	0.504	0.247	0.455
ACD	0.352	0.407	0.471	0.179	0.393	0.186	0.481	0.377	0.469	**0.656**	0.158	0.425
SHRED ($MT = .5$)	0.614	0.610	0.633	0.601	0.524	0.426	**0.568**	0.408	0.584	0.606	0.430	0.644
SHRED ($MT = .8$)	**0.631**	**0.626**	**0.647**	**0.618**	**0.534**	**0.455**	0.540	**0.447**	**0.592**	0.626	**0.435**	**0.645**

merge network trained for 700k iterations (~3 days). All networks perform early stopping on the in-domain validation set.

4.3 Decomposition Quality vs Granularity

We compare SHRED against baseline methods on their ability to navigate the trade-off between decomposition quality and decomposition granularity. Given shape S with GT decomposition R^*, the objective is to produce a decomposition R such that (i) $r_i \subseteq r_j^* \; \forall \; i \in N$ and (ii) $|R| \leq |R^*|$. As these conditions are infeasible to meet in practice, we evaluate the goodness of a region decomposition R by analyzing the degree to which it violates these properties. We capture violations of (i) with a region purity metric, explained below. Adherence to property (ii) can be easily captured by $|R|$, where the goal is to minimize this value.

Region Purity Metric. Given a shape S, region decomposition R and ground-truth decomposition R^*, the region purity metric aims to capture the quality of R irrespective of R's granularity. By quality we refer to the degree of under-segmentation present in R w.r.t R^*, i.e. the degree to which property (i), $r_i \subseteq r_j^* \; \forall \; i \in N$, is violated .

The region purity metric takes values from 0 to 1: 1 indicates (i) is not violated, while lower values indicate it has been violated to a greater degree. Region purity is calculated by the following procedure. First, we find an optimal assignment A from the regions of R to the regions of R^*. A is a region decomposition that keeps track of the best $r_k^* \in R^*$ for each $r_i \in R$. For each $r_i \in R$, we calculate the ground-truth region r_k^* that r_i best matches: $r_k^* = \max_{r_j^* \in R^*} IoU(r_i, r_j^*)$. Then, we find all points in S assigned to r_i under R, and set their label to r_k^* under A. Notice that as R is a valid region decomposition, we are guaranteed that A will also be a valid region decomposition. Once A has been computed, we calculate how well A matches R^*. For each ground-truth region, $r_j^* \in R^*$, we find all points in S assigned to r_j^* and calculate the percentage of those points assigned to r_j^* under A. The final region purity metric for the (S, R, R^*) triplet is then the average of this value across all ground-truth regions of R^*.

Comparisons. We evaluate SHRED against a suite of baseline methods:

- **FPS**: Farthest-point sampling, the naive region decomposition that initializes SHRED.
- **ACD**: Approximate Convex Decomposition, a popular non-learning based method [Mamou 2016]
- **PN Seg**: Learning-based method that predicts global fine-grained instance segmentations [Mo et al. 2019].
- **WOPL**: Learning-based method that locally splits and globally merges regions with clustering [Wang et al. 2021a].
- **L2G**: Learning-based method that uses a policy gradient trained network to locally merge regions [Luo et al. 2020].

WOPL and PN Seg use the same training data as SHRED. L2G is trained on a superset of data that SHRED uses (same in-domain categories, more shapes per category). Please see the supplemental for implementation details.

We plot the trade-off that each method makes between region quality (region purity, Y-axis) and region granularity (number of regions, X-axis) in Figure 3. The solid line and circles indicate in-domain averages, while the dotted line and stars indicate out-domain averages. Methods that produce better region decompositions will be closer to the top-left corner.

SHRED can vary the granularity of its output decomposition by modulating the merge-threshold (Section 3.3); we leverage this property to plot SHRED decomposition results as a curve, varying the merge-threshold from 0.01 to 0.99. Notice that this SHRED curve forms a Pareto frontier, dominating all other comparison methods in terms of region purity (Y-axis) at any level of decomposition granularity (X-axis). This demonstrates that, irrespective of the desired granularity, SHRED finds region decompositions that better respect the GT part regions compared with any baseline methods. Importantly, this trend holds on both in-domain and out-domain categories, indicating that SHRED has learned decomposition policies that generalize well to novel types of shapes. We provide additional results for different stages of baseline methods in the supplemental material.

4.4 Fine-grained Part Instance Segmentation

A straightforward application of SHRED is to treat its region decompositions as fine-grained part instance segmentations. We evaluate SHRED against comparison methods on this task, splitting results for in-domain test-set shapes (generalization) and out-domain test-set shapes (zero-shot). Following [Wang et al. 2021a], we use AIoU as our part instance segmentation metric. For each GT part, we find the predicted part with the maximum IoU. The AIoU is then the

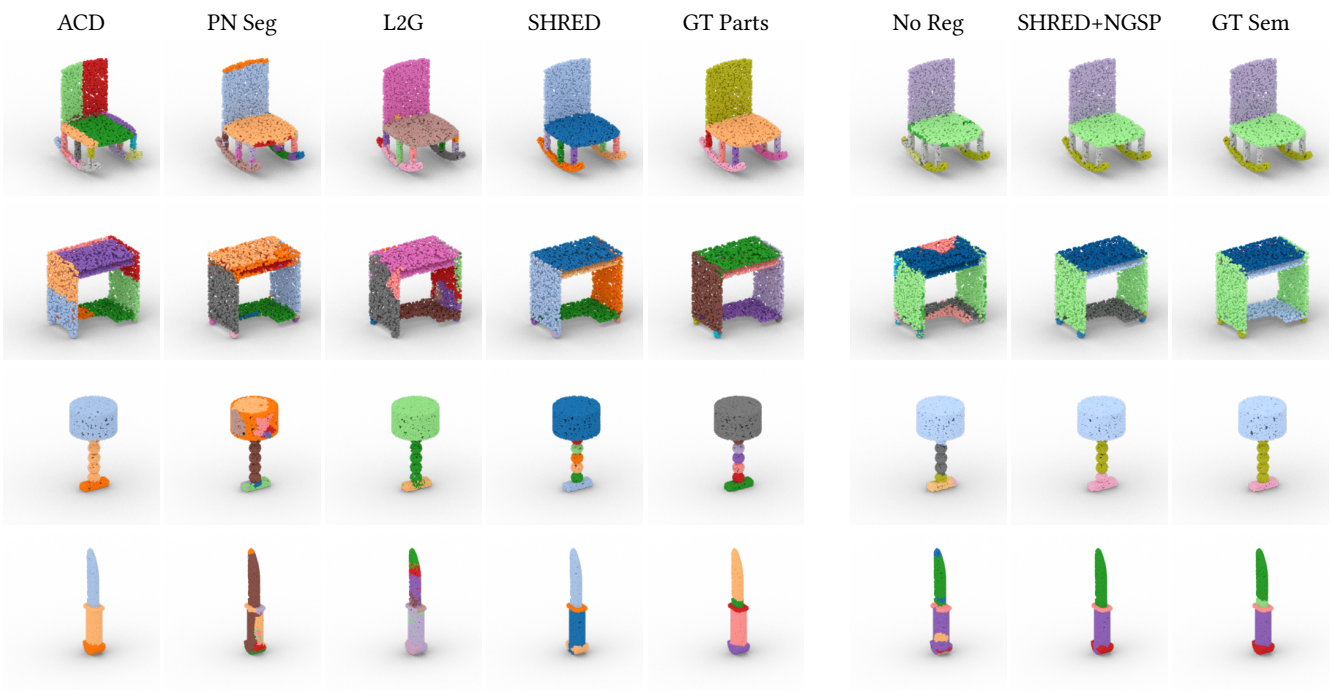

Fig. 4. *Left:* Qualitative region decomposition outputs, where regions are randomly colored. SHRED's predictions most closely match ground-truth part annotations (GT Parts). *Right:* Using NGSP to label SHRED's regions creates semantic segmentations that are more similar to ground-truth semantic annotations (GT Sem) compared with region-agnostic baselines (No Reg), when labeled data is limited.

average of this value across all GT part instances, where higher AIoU values are better.

We present quantitative results of our experiments in Table 1 and qualitative results in Figure 4, left. We find SHRED outperforms comparison methods on fine-grained instance segmentations for both in-domain and out-domain categories. For in-domain categories SHRED with a 0.5 merge-threshold achieves a 44% boost over the next best method (L2G), while for out-domain categories SHRED provides a 33% boost over the next best method (ACD).

We analyze how the merge-threshold hyperparameter affects SHRED's fine-grained instance segmentation performance. In Figure 5, we plot region decomposition granularity (X-axis) versus instance segmentation AIoU (Y-axis). As in other plots, we are able to represent SHRED results as a curve by modulating the merge-threshold from 0.01 to 0.99. When the merge-threshold is set to very-low or very-high values, the AIoU performance deteriorates. We plot the performance of the default merge-threshold value, 0.5, as a blue circle (star) for the in-domain (out-domain) average. The default merge-threshold value achieves close to optimal performance out of all merge-threshold values, but the AIoU performance can be slightly improved by increasing the merge-threshold to 0.8 (see Table 1, last row).

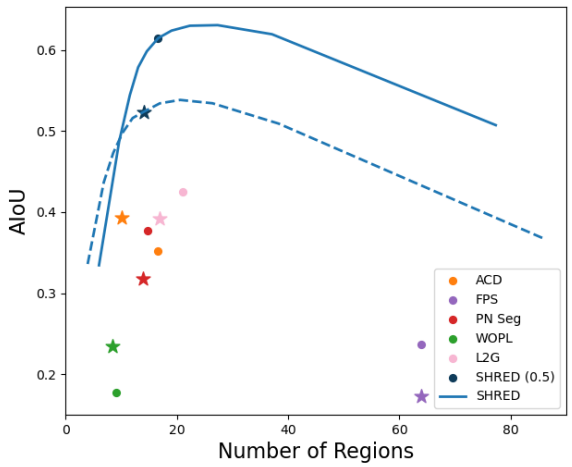

Fig. 5. We plot fine-grained instance segmentation performance (AIoU) as a function of the number of predicted regions. SHRED with the default merge-threshold is shown in dark-blue, while we also vary the merge-threshold from 0.01 to 0.99 to form a curve of SHRED results (blue).

4.5 Fine-grained Semantic Segmentation

We examine how SHRED can be used to improve fine-grained semantic segmentation performance when access to semantic label annotations are limited. The Neurally-Guided Shape Parser (NGSP) is a method that learns to assign semantic labels to regions of a 3D shape; it has demonstrated advantages over globally learned semantic segmentation networks when it receives well-structured

Table 2. Semantic segmentation results in a few-shot paradigm (# Train) with no regions (No Reg) and combining NGSP with region decomposition methods. SHRED+NGSP achieves the best mIoU performance averaged across categories.

			In Domain			Out Domain	
# Train	**Method**	*Avg*	*Chair*	*Lamp*	*Storage*	*Table*	*Knife*
10 shapes	No Reg	0.148	0.188	0.104	0.231	0.068	**0.335**
	PN Seg	0.203	0.190	0.210	0.287	0.123	0.075
	L2G	0.240	0.276	**0.239**	0.263	0.180	0.132
	ACD	0.154	0.193	0.187	0.115	0.119	0.151
	SHRED	**0.277**	**0.311**	0.229	**0.365**	**0.203**	0.205
40 shapes	No Reg	0.276	0.345	0.177	0.312	0.268	**0.417**
	PN Seg	0.298	0.305	0.324	0.332	0.232	0.128
	L2G	0.328	0.407	0.331	0.316	0.259	0.334
	ACD	0.237	0.304	0.302	0.168	0.174	0.254
	SHRED	**0.375**	**0.431**	**0.344**	**0.415**	**0.311**	0.355

(e.g. human-produced) region decompositions as input [Jones et al. 2022]. We evaluate the performance of NGSP when SHRED is used to produce the input region decomposition compared with alternative methods. For each category, NGSP requires training two types of models: a guide network and likelihood networks. We train a guide network separately for each region decomposition method. As likelihood network training is expensive, we train a single likelihood network using ground-truth part instance annotations, and share this network across experimental conditions. Please see the supplemental material for additional experiment details.

Combining SHRED and NGSP. We present results of this experiment in Table 2 using the intersection of categories studied by our method and NGSP. The rows in the top (bottom) half of the table correspond to training the guide and likelihood networks on 10 (40) shapes with semantic label annotations. The metric used is semantic mIoU, where higher-values indicate a better semantic segmentation. As seen, using SHRED produced regions allows NGSP to achieve the best semantic segmentation (last-rows) compared with using baseline methods to produce shape regions (middle-rows), or using no regions (top-row, PartNet semantic segmentation network [Mo et al. 2019]). This result generally holds for both individual in-domain and out-of-domain categories. Interestingly, for the knife category, the no region method outperforms all NGSP variants. We attribute this result to the fact that the knife grammar from PartNet is relatively coarse and contains a top-level binary split into two sub-categories of cutting instruments (knives versus daggers) with virtually identical sub-trees. We share qualitative results in Figure 4, right. As demonstrated, combining SHRED with NGSP typically produces fine-grained semantic labels that better match GT annotations. One limitation of NGSP is that it is not able to correct under-segmentations provided by region decomposition methods. For instance, the blade (dark-green) and the bolster (light-green) of the knife in the last row of Figure 4, are grouped into the same region by SHRED, so NGSP is unable to accurately segment out the bolster. To improve performance in these cases, it may be useful to run iterative rounds of SHRED and NGSP, so that SHRED can further decompose regions that NGSP believes may be under-segmented.

Combining SHRED and Guide. As NGSP guide network training is fast, we include additional semantic segmentation results in Table 3, where a partial version of NGSP, with just the guide network and no likelihood networks, is used to create semantic segmentations. We include results for additional categories (all in-domain and out-domain categories studied in Section 4). We also include additional region decomposition methods: FPS, WOPL (with just the prior network), and SHRED with different merge-threshold values (0.2, 0.5, 0.8). The guide network benefits most from SHRED generated regions for both in-domain and out-domain averages. When semantic labeling grammars are more coarse, and when more labeled training shapes are present, the gap between SHRED and alternative approaches shrinks. The chair, lamp, storage and table categories all have more than 18 nodes in their semantic grammars, while the bed, display, earphone, faucet, knife and refrigerator categories all have less than 11 nodes in their semantic grammars.

4.6 SHRED Ablations

In this section, we consider the performance of SHRED under different hyperparameter settings and ablations conditions. We quantitatively evaluate each modified version in Table 4, by measuring the achieved fine-grained instance segmentation AIoU over in-domain and out-domain category averages. In the rest of this section, we describe the different ablation conditions that populate the rows of the table.

Removing local operations. SHRED uses three locally learned modules to perform region splits, fixes and merges. We evaluate ablated version of SHRED, where only two modules are employed to produce region decompositions, in the *No Split*, *No Fix*, and *No Merge* rows. As demonstrated, removing any of SHRED's local operations leads to worse region decompositions.

Modified Hungarian Matching Algorithm. As discussed in Section 3.1, split network training requires a dynamically computed matching between predicted and target instance slots. SHRED uses a variant of the typical Hungarian matching algorithm for this procedure, where over-segmentation is explicitly encouraged. In the *No Hung OS match* row, we ablate this design decision, replacing our over-segmentation biased Hungarian matching algorithm with the typical exact instance segmentation formulation. As seen, changing to the default formulation slightly decreases SHRED's performance. Please see the supplemental material for a detailed explanation of our matching algorithm.

Training with less data. By default, the local operators of SHRED are trained over multiple in-domain shape categories (chairs, lamps, and storage furniture) with each category contributing 2000 shape instances. In the *Chair only*, *Lamp only*, and *Storage only* ablation rows, we evaluate how SHRED is able to generalize when trained over 2000 instances from a single shape category. In the *Limited data* row, we present a version of SHRED that is trained over the same shape categories (chairs, lamps and storage furniture), but where each category contributes only 200 shape instances. From these results, we can observe that SHRED benefits from both (a) more in-domain shape categories and (b) more shape instances per category, but (a) should be prioritized over (b). In fact, SHRED with

Table 3. Semantic segmentation mIoU performance using the NGSP guide network (no likelihood networks) to assign semantic labels to shape regions produced by different decomposition methods. We show how the guide network performs under different settings of SHRED, varying the merge-threshold from 0.2 to 0.5 to 0.8 .

		In Domain				Out Domain							
# Train	**Method**	*Avg*	*Chair*	*Lamp*	*Storage*	*Avg*	*Bed*	*Display*	*Earphone*	*Faucet*	*Knife*	*Fridge*	*Table*
	No Reg	0.174	0.188	0.103	0.230	0.331	0.267	0.666	0.263	0.346	0.335	0.373	0.068
	FPS	0.221	0.250	0.205	0.207	0.336	0.241	0.607	**0.274**	**0.390**	0.327	0.367	0.147
	WOPL Prior	0.207	0.231	0.149	0.241	0.336	0.259	0.590	0.265	0.357	**0.338**	0.362	0.180
	PN SEG	0.213	0.230	0.180	0.230	0.285	0.232	0.579	0.193	0.263	0.216	0.395	0.117
10 shapes	L2G	0.233	0.247	0.205	0.247	0.293	0.256	0.593	0.218	0.262	0.210	0.354	0.157
	ACD	0.188	0.260	0.188	0.117	0.296	0.219	0.669	0.218	0.348	0.265	0.229	0.127
	SHRED (MT = .2)	0.259	0.251	**0.220**	0.307	0.315	0.250	0.623	0.174	0.322	0.227	0.471	0.141
	SHRED (MT = .5)	**0.279**	0.287	0.193	**0.357**	0.356	0.303	**0.690**	0.190	0.380	0.279	**0.494**	0.159
	SHRED (MT = .8)	0.265	**0.297**	0.183	0.316	**0.360**	**0.309**	0.653	0.242	0.341	0.328	0.466	**0.181**
	No Reg	0.278	0.345	0.177	0.312	0.443	**0.452**	0.714	0.371	0.390	**0.417**	0.487	0.268
	FPS	0.276	0.275	0.292	0.261	0.399	0.326	0.654	**0.419**	0.436	0.355	0.353	0.247
	WOPL Prior	0.316	0.322	0.312	0.314	0.416	0.355	0.678	0.408	0.415	0.410	0.381	0.266
	PN SEG	0.299	0.314	0.239	0.344	0.367	0.293	0.622	0.331	0.357	0.317	0.425	0.226
40 shapes	L2G	0.332	0.380	0.302	0.313	0.383	0.327	0.679	0.372	0.382	0.297	0.397	0.229
	ACD	0.266	0.328	0.296	0.173	0.361	0.248	0.702	0.356	0.373	0.412	0.248	0.190
	SHRED (MT = .2)	0.343	0.365	0.298	0.367	0.421	0.395	0.709	0.298	0.390	0.338	0.510	0.304
	SHRED (MT = .5)	**0.368**	**0.399**	**0.325**	0.379	0.443	0.409	0.722	0.344	0.418	0.396	**0.522**	0.287
	SHRED (MT = .8)	**0.368**	0.398	0.300	**0.407**	**0.465**	0.408	**0.736**	0.405	**0.468**	0.400	0.504	**0.331**

Table 4. SHRED instance segmentation performance under different hyper-parameter settings and ablation conditions. Metric is AIoU, averaged for both the in-domain and out-domain categories. Best , second-best , and third-best conditions are highlighted.

Condition	**In Domain AIoU**	**Out Domain AIoU**
No Split	0.470	0.440
No Fix	0.574	0.492
No Merge	0.324	0.225
No Hung OS match	0.599	0.515
Chair only	0.484	0.485
Lamp only	0.329	0.387
Storage only	0.444	0.414
Limited data (10%)	0.543	0.496
Split with naive SDC	0.490	0.448
Align with naive SDC	0.561	0.503
Merge with naive SDC	0.488	0.425
Cascade training	0.434	0.413
SHRED (K = 32)	0.594	0.512
SHRED (K = 64)	0.614	0.524
SHRED (K = 128)	0.617	0.541

just 10% of the training data still significantly outperforms all of the baseline methods from Table 1.

Naive Synthetic Data Creation. As described throughout Section 3, SHRED employs synthetic data creation (SDC) strategies to produce training data for each local operator. We designed these SDC strategies to roughly match the types of input each operator might expect during the course of the split-fix-merge sequence. In the *Split with naive SDC*, *Align with naive SDC*, and *Merge with naive SDC* rows we ablate our SDC procedure for one operator at a time, by replacing the default SDC with a naive SDC. Please see the supplemental material for details on the naive synthetic data creation procedures. In all cases, replacing the default SDC with the naive SDC leads to worse AIoU. This supports our claim that having a synthetic data creation strategy that broadly matches the expected input for each local operator improves SHRED's overall performance.

Cascade Training of Operators. SHRED trains each operator independently with synthetic data creation procedures. When a single operator sequence is desired, it is also possible to train the operators of SHRED in a cascading fashion: taking the predictions from previous stages to produce the training data for later stages. We present a version of SHRED trained within this paradigm in the *Cascade training* row. In this paradigm, the fix network is trained on predictions from the split network, and the merge network is trained on predictions from the split and fix networks. The performance of this condition is much worse compared with the default version of SHRED, likely because in the cascade approach the amount of operator training data is bottle-necked by the number of training shapes, whereas the synthetic data creation procedure can generate infinite amounts of operator training data, even with limited training shapes. Further note that training over synthetic data allows operators to be trained in parallel and provides greater flexibility in how the operators can be applied during an inference procedure.

Initial Region Splits. SHRED applies its learned local operators over regions produced by a naive decomposition method. By default,

we use a farthest-point sampling (FPS) procedure to produce the naive decomposition with $K = 64$ centroids. In the last rows of Table 4 we evaluate SHRED's instance segmentation performance under different values of K. SHRED is not overly sensitive to the initial decomposition granularity, performing well under all K values. The best AIoU value, for both in-domain and out-domain categories, is achieved with $K = 128$. The downside of starting with a finer initial region decomposition is that SHRED inference is slower; SHRED inference with $K = 64$ takes under 6 seconds per shape, while SHRED inference with $K = 128$ takes over 10 seconds per shape.

5 CONCLUSION

We introduced SHRED, a method that performs 3D SHape REgion Decomposition by learning to locally split, fix and merge. We trained SHRED on part annotations from three data-abundant PartNet categories (chairs, lamps, storage) and experimentally validated its ability to produce high-quality region decompositions for out-domain categories of manufactured shapes. In comparisons against baseline methods, we used SHRED's merge-threshold hyperparameter to demonstrate that it offers the best trade-off between decomposition quality and granularity. Finally, we evaluated SHRED on downstream applications involving fine-grained parts, zero-shot instance segmentation and few-shot semantic labeling, finding that SHRED improves performance over baselines.

Future Work. While SHRED has demonstrated an impressive capacity to generalize to out-of-distribution instances, all of the shapes we have experimented with come from the same meta-distribution (manufactured objects). It would be interesting to investigate how SHRED generalizes to more diverse domains, including those that share locally similar properties (partial shape scans, 3D scenes) or those with hardly any similarities (organic bodies). Along another direction, while SHRED's merge-threshold allows some exploration over a range of decomposition granularities, the segmentations that SHRED produces are always flat. Developing a procedure to convert this series of flat decompositions into a shared hierarchical segmentation may prove useful for various downstream applications (e.g. collision detection or structure-based generative modeling).

Finally, while we find that SHRED's sequential application of the split, fix and merge operations results in region decompositions that outperform previous approaches, it is unlikely that this sequence is optimal for every shape instance. It is easy to imagine that some input shapes might benefit from repeated applications of a particular operation, by running this iterative procedure multiple times, or by only applying the operation to certain regions. Wrapping the operations of SHRED with a more advanced outer-loop search would be one way to accomplish this per-shape tailoring goal. Moreover, if this search was guided by a global likelihood function, then application-dependant terms could even be added into consideration (e.g. encouraging convex regions or respecting bilateral symmetries). Alternatively, a human-in-the-loop could provide constraints to guide the decomposition process in an interactive fashion, by for instance, indicating patches to be split or merged, or boundaries to be fixed, with sketched-based controls. We believe that these paradigms, where human-specified objectives help guide local data-driven modules, are a promising way forward towards generating human-quality fine-grained region decompositions of 3D objects from distributions that lack abundant annotations.

ACKNOWLEDGMENTS

We would like to thank the anonymous reviewers for their helpful suggestions. This work was funded in parts by NSF award #1941808 and a Brown University Presidential Fellowship. Daniel Ritchie is an advisor to Geopipe and owns equity in the company. Geopipe is a start-up that is developing 3D technology to build immersive virtual copies of the real world with applications in various fields, including games and architecture

REFERENCES

Ben Abbatematteo, Stefanie Tellex, and George Konidaris. 2019. Learning to Generalize Kinematic Models to Novel Objects. In *Proceedings of the Third Conference on Robot Learning*.

Shmuel Asafi, Avi Goren, and Daniel Cohen-Or. 2013. Weak convex decomposition by lines-of-sight. In *Computer graphics forum*, Vol. 32. Wiley Online Library, 23–31.

Zhiqin Chen, Andrea Tagliasacchi, and Hao Zhang. 2019a. BSP-Net: Generating Compact Meshes via Binary Space Partitioning. arXiv:1911.06971 [cs.CV]

Zhiqin Chen, Kangxue Yin, Matthew Fisher, Siddhartha Chaudhuri, and Hao Zhang. 2019b. BAE-NET: Branched Autoencoder for Shape Co-Segmentation. *Proceedings of International Conference on Computer Vision (ICCV)* (2019).

Boyang Deng, Kyle Genova, Soroosh Yazdani, Sofien Bouaziz, Geoffrey Hinton, and Andrea Tagliasacchi. 2020. CvxNet: Learnable Convex Decomposition. (June 2020).

Matheus Gadelha, Aruni RoyChowdhury, Gopal Sharma, Evangelos Kalogerakis, Liangliang Cao, Erik Learned-Miller, Rui Wang, and Subhransu Maji. 2020. Label-Efficient Learning on Point Clouds using Approximate Convex Decompositions. In *European Conference on Computer Vision (ECCV)*.

Kyle Genova, Forrester Cole, Daniel Vlasic, Aaron Sarna, William T Freeman, and Thomas Funkhouser. 2019. Learning shape templates with structured implicit functions. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 7154–7164.

Songfang Han, Jiayuan Gu, Kaichun Mo, Li Yi, Siyu Hu, Xuejin Chen, and Hao Su. 2020. Compositionally Generalizable 3D Structure Prediction. (2020). arXiv:arXiv:2012.02493

Yu Hao and Yi Fang. 2021. Meta-Learning 3D Shape Segmentation Functions. https://doi.org/10.48550/ARXIV.2110.03854

Hao Huang, Xiang Li, Lingjing Wang, and Yi Fang. 2021. 3D-MetaConNet: Meta-learning for 3D Shape Classification and Segmentation. In *2021 International Conference on 3D Vision (3DV)*. 982–991. https://doi.org/10.1109/3DV53792.2021.00106

R. Kenny Jones, Aalia Habib, Rana Hanocka, and Daniel Ritchie. 2022. The Neurally-Guided Shape Parser: Grammar-based Labeling of 3D Shape Regions with Approximate Inference. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*.

Oliver Van Kaick, Noa Fish, Yanir Kleiman, Shmuel Asafi, and Daniel Cohen-Or. 2014. Shape segmentation by approximate convexity analysis. *ACM Transactions on Graphics (TOG)* 34, 1 (2014), 1–11.

Sagi Katz and Ayellet Tal. 2003. Hierarchical mesh decomposition using fuzzy clustering and cuts. *ACM transactions on graphics (TOG)* 22, 3 (2003), 954–961.

Yuki Kawana, Yusuke Mukuta, and Tatsuya Harada. 2020. Neural Star Domain as Primitive Representation. In *NeurIPS 2020*.

Diederik P. Kingma and Jimmy Ba. 2014. Adam: A Method for Stochastic Optimization. *CoRR* abs/1412.6980 (2014).

Loic Landrieu and Martin Simonovsky. 2018. Large-scale point cloud semantic segmentation with superpoint graphs. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 4558–4567.

Jyh-Ming Lien and Nancy M Amato. 2008. Approximate convex decomposition of polyhedra and its applications. *Computer Aided Geometric Design* 25, 7 (2008), 503–522.

Rong Liu and Hao Zhang. 2004. Segmentation of 3D meshes through spectral clustering. In *12th Pacific Conference on Computer Graphics and Applications, 2004. PG 2004. Proceedings.* 298–305.

Zhengzhe Liu, Xiaojuan Qi, and Chi-Wing Fu. 2021. One Thing One Click: A Self-Training Approach for Weakly Supervised 3D Semantic Segmentation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 1726–1736.

Tiange Luo, Kaichun Mo, Zhiao Huang, Jiarui Xu, Siyu Hu, Liwei Wang, and Hao Su. 2020. Learning to Group: A Bottom-Up Framework for 3D Part Discovery in Unseen Categories. In *International Conference on Learning Representations*. https://openreview.net/forum?id=rkl8dlHYvB

Khaled Mamou. 2016. Volumetric Hierarchical Approximate Convex Decomposition. In *Game Engine Gems 3*, Eric Lengyel (Ed.). A K Peters, 141–158.

Kaichun Mo, Shilin Zhu, Angel X. Chang, Li Yi, Subarna Tripathi, Leonidas J. Guibas, and Hao Su. 2019. 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)*.

Despoina Paschalidou, Angelos Katharopoulos, Andreas Geiger, and Sanja Fidler. 2021. Neural Parts: Learning Expressive 3D Shape Abstractions with Invertible Neural Networks. In *Proceedings IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*.

Despoina Paschalidou, Ali Osman Ulusoy, and Andreas Geiger. 2019. Superquadrics Revisited: Learning 3D Shape Parsing beyond Cuboids. In *Proceedings IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*.

Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. 2017. Automatic differentiation in PyTorch. (2017).

Charles R Qi, Hao Su, Kaichun Mo, and Leonidas J Guibas. 2017a. Pointnet: Deep learning on point sets for 3D classification and segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 652–660.

Charles Ruizhongtai Qi, Li Yi, Hao Su, and Leonidas J Guibas. 2017b. Pointnet++: Deep hierarchical feature learning on point sets in a metric space. In *Advances in neural information processing systems*. 5099–5108.

Ariel Shamir. 2008. A survey on mesh segmentation techniques. In *Computer graphics forum*, Vol. 27. Wiley Online Library, 1539–1556.

Gopal Sharma, Bidya Dash, Matheus Gadelha, Aruni RoyChowdhury, Marios Loizou, Evangelos Kalogerakis, Liangliang Cao, Erik Learned-Miller, and Rui Wang and-Subhransu Maji. 2021. SurFit: Learning to Fit Surfaces Improves Few Shot Learning on Point Clouds. https://doi.org/10.48550/ARXIV.2112.13942

Gopal Sharma, Evangelos Kalogerakis, and Subhransu Maji. 2019. Learning Point Embeddings from Shape Repositories for Few-Shot Segmentation. In *2019 International Conference on 3D Vision, 3DV 2019, Québec City, QC, Canada, September 16-19, 2019*. IEEE, 67–75. https://doi.org/10.1109/3DV.2019.00017

Chunyu Sun, Yiqi Yang, Haoxiang Guo, pengshuai Wang, Xin Tong, Yang Liu, and Shum Heung-Yeung. 2022. Semi-Supervised 3D Shape Segmentation with Multilevel Consistency and Part Substitution. *Computational Visual Media* (2022).

Chun-Yu Sun, Qian-Fang Zou, Xin Tong, and Yang Liu. 2019. Learning Adaptive Hierarchical Cuboid Abstractions of 3D Shape Collections. *ACM Trans. Graph.* 38, 6, Article 241 (Nov. 2019), 13 pages. https://doi.org/10.1145/3355089.3356529

Shubham Tulsiani, Hao Su, Leonidas J. Guibas, Alexei A. Efros, and Jitendra Malik. 2017. Learning Shape Abstractions by Assembling Volumetric Primitives. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

Lingjing Wang, Xiang Li, and Yi Fang. 2020. Few-Shot Learning of Part-Specific Probability Space for 3D Shape Segmentation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*.

Peng-Shuai Wang, Yu-Qi Yang, Qian-Fang Zou, Zhirong Wu, Yang Liu, and Xin Tong. 2021b. Unsupervised 3D Learning for Shape Analysis via Multiresolution Instance Discrimination. *Proceedings of the AAAI Conference on Artificial Intelligence* 35, 4 (May 2021), 2773–2781. https://ojs.aaai.org/index.php/AAAI/article/view/16382

Weiyue Wang, Ronald Yu, Qiangui Huang, and Ulrich Neumann. 2018. SGPN: Similarity Group Proposal Network for 3D Point Cloud Instance Segmentation. In *CVPR*.

Xiaogang Wang, Xun Sun, Xinyu Cao, Kai Xu, and Bin Zhou. 2021a. Learning Fine-Grained Segmentation of 3D Shapes Without Part Labels. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 10276–10285.

Yue Wang, Yongbin Sun, Ziwei Liu, Sanjay E. Sarma, Michael M. Bronstein, and Justin M. Solomon. 2019. Dynamic Graph CNN for Learning on Point Clouds. *ACM Transactions on Graphics (TOG)* (2019).

Yanzhen Wang, Kai Xu, Jun Li, Hao Zhang, Ariel Shamir, Ligang Liu, Zhiquan Cheng, and Yueshan Xiong. 2011. Symmetry hierarchy of man-made objects. In *Computer graphics forum*, Vol. 30. Wiley Online Library, 287–296.

Erik Wijmans. 2018. Pointnet++ Pytorch. *https://github.com/erikwijmans/Pointnet2_PyTorch* (2018).

Saining Xie, Jiatao Gu, Demi Guo, Charles R. Qi, Leonidas Guibas, and Or Litany. 2020. PointContrast: Unsupervised Pre-training for 3D Point Cloud Understanding. In *Computer Vision – ECCV 2020*, Andrea Vedaldi, Horst Bischof, Thomas Brox, and Jan-Michael Frahm (Eds.). Springer International Publishing, Cham, 574–591.

Xun Xu and Gim Hee Lee. 2020. Weakly Supervised Semantic Point Cloud Segmentation: Towards 10x Fewer Labels. In *CVPR*.

Kaizhi Yang and Xuejin Chen. 2021. Unsupervised Learning for Cuboid Shape Abstraction via Joint Segmentation from Point Clouds. *ACM Trans. Graph.* 40, 4, Article 152 (jul 2021), 11 pages. https://doi.org/10.1145/3450626.3459873

Li Yi, Leonidas Guibas, Aaron Hertzmann, Vladimir G. Kim, Hao Su, and Ersin Yumer. 2017. Learning Hierarchical Shape Segmentation and Labeling from Online Repositories. *SIGGRAPH* (2017).

Li Yi, Wang Zhao, He Wang, Minhyuk Sung, and Leonidas J. Guibas. 2019. GSPN: Generative Shape Proposal Network for 3D Instance Segmentation in Point Cloud. In *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 3942–3951. https://doi.org/10.1109/CVPR.2019.00407

Fenggen Yu, Kun Liu, Yan Zhang, Chenyang Zhu, and Kai Xu. 2019. PartNet: A Recursive Part Decomposition Network for Fine-grained and Hierarchical Shape Segmentation. In *CVPR*. to appear.

Chenyang Zhu, Kai Xu, Siddhartha Chaudhuri, Li Yi, Leonidas J. Guibas, and Hao Zhang. 2020. AdaCoSeg: Adaptive Shape Co-Segmentation With Group Consistency Loss. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*.