**SURFIT: Learning to Fit Surfaces Improves Few Shot Learning on Point Clouds**

Gopal Sharma*1  Bidya Dash*1  Matheus Gadelha2  Aruni RoyChowdhury1  Marios Loizou3  Evangelos Kalogerakis1  Liangliang Cao1  Erik Learned-Miller1  Rui Wang1  Subhransu Maji1

1University of Massachusetts Amherst  2Adobe  3University of Cyprus

Abstract

We present **SURFIT**, a simple approach for label efficient learning of 3D shape segmentation networks. **SURFIT** is based on a self-supervised task of decomposing the surface of a 3D shape into geometric primitives. It can be readily applied to existing network architectures for 3D shape segmentation, and improves their performance in the few-shot setting, as we demonstrate in the widely used ShapeNet and PartNet benchmarks. **SURFIT** outperforms the prior state-of-the-art in this setting, suggesting that decomposability into primitives is a useful prior for learning representations predictive of semantic parts. We present a number of experiments varying the choice of geometric primitives and downstream tasks to demonstrate the effectiveness of the method.

1. Introduction

Recent advances in visual recognition have been in part due to supervised training of deep networks on massive collections of images. However, collecting manual supervision on 3D domains is vastly more challenging, especially for detailed properties such as part labels of shapes. To this end we present **SURFIT**, a self-supervised approach for learning 3D shape representations based on decomposing the surface of a 3D shape into geometric primitives, that improves learning part segmentation models from a few labeled examples. Our approach exploits the fact that parts of 3D shapes are often aligned with simple geometric primitives, such as ellipsoids and cuboids. Even if these primitives capture 3D shapes at a rather coarse level, the induced partitions provide a strong prior for learning part segmentation networks, as seen in Fig. 1. This purely geometric task allows us to utilize vast amounts of unlabeled data in existing 3D shape repositories to guide representation learning for part segmentation, which is especially useful in the few-shot setting.

The overall framework for **SURFIT** is based on a point embedding module and a primitive fitting module, as illustrated in Fig. 2. The point embedding module is a deep network that generates per-point embeddings for a 3D shape, represented as point clouds, where the color indicates the parts such as wings and engines. The induced partitions and shape reconstruction obtained by fitting ellipsoids to each shape using our approach are shown in the middle row and bottom row respectively. The induced partitions have a significant overlap with semantic parts.

Figure 1. **SURFIT** uses primitive fitting as a self-supervised task for learning 3D shape representations. Top row: 3D shapes represented as point clouds, where the color indicates the parts such as wings and engines. The induced partitions and shape reconstruction obtained by fitting ellipsoids to each shape using our approach are shown in the middle row and bottom row respectively. The induced partitions have a significant overlap with semantic parts.
ric primitives based on an Atlas [18], as an alternative surface primitive representation.

Our method achieves 63.4% part IoU performance in ShapeNet segmentation dataset [7] with just one labeled example per-class, outperforming the prior state-of-the-art [13] by 1.6%. We also present results on the PartNet dataset [32] where our self-supervision provides 2.1% improvement compared to a baseline approach while using 10 labeled examples per-class. We also perform extensive analysis of the impact of various design choices and primitive types on the resulting shape segmentations. Our experiments indicate that the use of ellipsoids as geometric primitives provide the best performance, followed by cuboids, then AtlasNet patches.

2. Related Work

We are interested in learning per-point representations of 3D shapes in a self-supervised manner given a large number of unlabeled shapes and only a few labeled examples. To this end, we briefly review the literature on geometric primitive fitting and shape decomposition, few-shot learning, and deep primitive fitting. We also discuss the limitations of prior work and how we address them.

Geometric primitives and shape decomposition Biederman’s recognition-by-components theory [3] attempts to explain object recognition in humans by the ability to assemble basic shapes such as cylinders and cones, called geons, into the complex objects encountered in the visual world. Early work in cognitive science [20] shows that humans are likely to decompose a 3D shape along regions of maximum concavity, resulting in parts that tend to be convex, often referred to as the “minima rule”. Classical approaches in computer vision have modeled 3-D shapes as a composition of simpler primitives, e.g. work by Binford [4, 5] and Marr [29]. More recent work in geometric processing has developed shape decomposition techniques that generate different types of primitives which are amenable to tasks like editing, grasping, tracking and animation. Those have explored primitives like 3D curves [15, 17, 30], cages [50], sphere-meshes [46], generalized cylinders [55], radial basis functions [6, 26] and simple geometric primitives [40]. This motivates the use of our geometric primitive fitting as a self-supervised task for learning representations.

Unsupervised learning for 3D data. Several previous techniques have been proposed to learn 3D representations without relying on extra annotations. Many such techniques rely on reconstruction approaches [14, 18, 52–54]. FoldingNet [53] uses an auto-encoder trained with permutation invariant losses to reconstruct the point cloud. Their decoder consists of a neural network representing a surface parametrized on a 2D grid. AtlasNet [18] proposes using several such decoders that result in the reconstructed surface being represented as a collection of surface patches. Li et al. [24] presents SO-Net that models spatial distribution of point cloud by constructing a self-organizing map, which is used to extract hierarchical features. The proposed architecture trained in auto-encoder fashion learns representation useful for classification and segmentation. Chen et al. [8] proposes an auto-encoder with multiple branches, where each branch is used to reconstruct the shape by producing implicit fields instead of point clouds. However, this requires one decoder for separate part, which restricts its use to category-specific models. Several techniques also proposed models for generating implicit functions from point clouds [9, 16, 31], but it is unclear how well the representations learned by those methods perform in recognition tasks.

Several works use reconstruction losses along with other self supervised tasks. Hassani et al. propose multiple tasks: reconstruction, clustering and classification to learn point representation for shapes. 3D jigsaw puzzles [2, 39] has been used as a self-supervised task to learn shape representation for classification and part segmentation. Thabet et al. [45] propose a self-supervision task of predicting the next point in a space filling curve (Morton-order curve) using RNN. The output features from the RNN are used for semantic segmentation tasks. Several works have proposed learning point representation using noisy labels and semantic tags available from various shape repositories. Sharma et al. [42] learn point representations using noisy part hierarchies and designer-labeled semantic tags for few-shot semantic segmentation. Muralikrishnan et al. [34] design a U-Net to learn point representations that predicts user-prescribed shape-level tags by first predicting intermediate semantic segmentation. More recently, Xie et al. [51] learn per-point representation for 3D scenes, where point embeddings of matched points from two different views of a scene are pushed closer than un-matched points under a contrastive learning framework.

Closely related to our work, Gadelha et al. [13] use approximate convex decomposition of watertight meshes as source of self-supervision by training a metric over point clouds that respect the decomposition. Our approach directly operates on point clouds and integrates the decomposition objectives in a unified and end-to-end trainable manner. Empirically we show that this improves performance and removes the need for the black-box approach.

Deep primitive fitting. Several approaches have investigated the use of deep learning models for shape decomposition. The overall idea is to use a 3D discriminative model to compute a latent representation which will be later decoded by a neural network specifically designed to generate primitive parameters. Several primitive types have been
used, including superquadrics [35,36], cuboids [12,47] and radial basis functions [16]. However, all these approaches have focused on generative tasks with the goal of editing or manipulating a 3D shape. Our insight is that reconstructing a shape by assembling simpler components improves representation learning for discriminative tasks, especially when only a few labeled training examples are available.

3. Method

Our method assumes that one is provided with a small set of labeled shapes $X_l$ and a large set of unlabeled shapes $X_u$. Each shape $X \in \{X_l, X_u\}$ is represented as a point cloud with $N$ points, i.e., $X = \{x_i\}$ where $x_i \in \mathbb{R}^3$. The shapes in $X_l$ additionally come with part label $Y = \{y_i\}$ for each point. In our experiments we use the entire set of shapes from the ShapeNet core dataset [7] and few labeled examples from the ShapeNet semantic segmentation dataset and PartNet dataset.

The architecture of SURFit consists of a point embedding module $\Phi$ and a primitive fitting module $\Psi$. The point embedding module $\Phi(X)$ maps the shape into embeddings corresponding to each point $\{\Phi(x_i)\} \in \mathbb{R}^D$. The primitive fitting module $\Psi$ maps the set of point embeddings to a set of primitives $\{P_i\} \in \mathcal{P}$. Thus $\Psi \circ \Phi : X \rightarrow \mathcal{P}$ is a mapping from point clouds to primitives. In addition the point embeddings can be mapped to point labels via a classification function $\Theta$ and thus, $\Theta \circ \Phi : X \rightarrow Y$. We follow a joint training approach where shapes from $X_l$ are used to compute a supervised loss and the shapes from $X_u$ are used to compute a self-supervised loss for learning by minimizing the following objective:

$$\min_{\Phi, \Psi, \Theta} \mathcal{L}_{ssl} + \mathcal{L}_{sl},$$

where

$$\mathcal{L}_{ssl} = \mathbb{E}_{X \sim X_l} [\ell_{ssl}(X, \Psi \circ \Phi(X))],$$

and

$$\mathcal{L}_{sl} = \mathbb{E}_{(X,Y) \sim X_l} [\ell_{sl}(Y, \Theta \circ \Phi(X))].$$

Here $\ell_{ssl}$ is defined as a reconstruction error between the point cloud and a set of primitives, while $\ell_{sl}$ is the cross entropy loss between predicted and true labels. We describe the details of the point embedding module in Sec. 3.1 and the primitive fitting module in Sec. 3.2. Finally in Sec. 3.3 we describe various loss functions used to train SURFit.

3.1. Point embedding module

This module produces an embedding of each point in a point cloud. While any point cloud architecture [37,38,49] can be used, we experiment with PointNet++ [38] and DGCNN [49], two popular architectures for point cloud segmentation. These architectures have also been used in prior work on few-shot semantic segmentation making a comparison easier.

3.2. Primitive fitting module

The primitive fitting module is divided into a decomposition step that groups the set of points into clusters in the embedding space, and a fitting step that estimates the parameters of the primitive for each cluster.

Decomposing a point cloud. These point embeddings $\Phi(x_i) \in \mathbb{R}^D$ are grouped into $M$ clusters using a differentiable mean-shift clustering. The motivation behind the
choice of mean-shift over other clustering approaches such as k-means is that it allows the number of clusters to vary according to a kernel bandwidth. In general we expect that different shapes require different number of clusters. We use recurrent mean-shift updates in a differentiable manner as proposed by [22]. Specifically, we initialize seed points as \( Y^{(0)} = Z \in \mathbb{R}^{N \times D} \) and update them as follows:

\[
Y^{(t)} = K Z D^{-1} \tag{4}
\]

We use the von Mises-Fisher kernel [28] \( K = \exp(Y^{(t-1)}Z^T/b^2) \), where \( D = \text{diag}(K1) \) and \( b \) is the bandwidth. \( K \) is updated after every iteration. The embeddings are normalized to unit norm, i.e., \( ||z_i||_2 = 1 \), after each iteration. We perform fixed number of iterations during training. After these updates, a non-max suppression step yields \( M \) cluster centers \( c_m, m = \{1, \ldots, M\} \) while making sure number of clusters are bounded. Having updated the embeddings with the mean-shift iterations, we can now define a soft membership \( W \) for each point \( x_i \), represented by the embedding vector \( y_i \), to the cluster center \( c_m \):

\[
w_{i,m} = \frac{\exp(e_m^T y_i)}{\sum_m \exp(e_m^T y_i)} \tag{5}
\]

where \( w_{i,m} = 1 \) represents the full membership of the \( i \)th point to the \( m \)th cluster. The supplementary material contains more details about non-max suppression and bandwidth computation.

Ellipsoid fitting. Given the clustering, we then fit an ellipsoid to each of the clusters. Traditionally, fitting an ellipsoid to a point cloud is formulated as a minimum volume enclosing ellipsoid [10] and solved using the Khachiyan algorithm. However, this is harder to incorporate in an end-to-end training pipeline and is also susceptible to outliers (see supplementary materials for details). We instead use a simpler procedure based on singular value decomposition (SVD). Given the membership of the point to \( m \)th cluster we first center the points and compute the SVD as:

\[
\mu = W_m X Z^{-1} \tag{6}
\]

\[
X = X - \mu \tag{7}
\]

\[
U, S, V = \text{SVD}(X^T W_m X Z^{-1}) \tag{8}
\]

where \( W_m \) is the diagonal matrix with \( w_{i,m} \) as its diagonal entries \( (W_m[i, i] = w_{i,m} \) for \( m \)th cluster) and \( Z = \text{trace}(W_m) \). The orientation of an ellipsoid is given by \( V \). The length of the principal axes can be computed from the singular values as \( a_i = \kappa \sqrt{S_i} \). We select \( \kappa \) by cross validation. The matrix \( W_m \) selects the points with membership to the \( m \)th cluster in a ‘soft’ or weighted fashion, and the SVD in Eq. 8 gives us the parameters of the ellipsoid that fits these weighted points.

Discussion — alternate choices for primitives. Our approach can be used to fit cuboids instead of ellipsoids by considering the bounding box of the fitted ellipsoids instead. This may induce different partitions over the point clouds, and we empirically compare it’s performance in the Sec. 4. Another choice is to represent the surface using an Atlas – a collection of parameterized patches. We use the technique proposed in AtlasNet [18] where neural networks \( f_\theta \) parameterize the coordinate charts \( f_\theta : [0, 1]^2 \rightarrow (x, y, z) \) conditioned on a latent code computed from the point cloud. The decoders are trained along with the encoder using gradient descent to minimize Chamfer distance between input points and output points across all decoders. An encoder trained in this fashion learns to decompose input points into complex primitives, i.e. via arbitrary deformations of the 2D plane. Point representations learnt in this fashion by the encoder can be used for downstream few-shot semantic segmentation task as shown in Sec. 4 and Tab. 2. However, this approach requires adding multiple decoder neural networks. Our ellipsoid fitting approach does not require significant architecture changes and avoids the extra parameters required by AtlasNet.

3.3. Loss functions

Reconstruction loss. This is computed as the Chamfer distance of input point clouds from predicted primitives. For the distance of a point on the input surface to the predicted surface we use

\[
L_1 = \sum_{i=1}^{N} \min_{m} D_m^2(x_i), \tag{9}
\]

where \( M \) is the number of clusters and \( D_m(x_i) \) is the distance of input point \( x_i \) from the \( m \)th primitive. This is one side of Chamfer distance, ensuring that the predicted primitives cover the input surface. \( D_m \) is approximated using analytic distance function [1] which performs better than a sampling based approach (see the supplementary material for more details). To ensure that the input surface covers the predicted primitives we minimize the following loss:

\[
L_2 = \sum_{m=1}^{M} \sum_{p \sim E_m} \min_{i=1}^{N} ||x_i - p||_2^2 \tag{10}
\]

where \( E_m \) is the \( m \)th fitted primitive. We sample 10k points over all ellipsoids, weighted by the surface area of each primitive. We uniformly sample each primitive surface.
Please refer to Supplementary material for more information on uniformly sampling an ellipsoid surface. We use the two-sided loss to minimize reconstruction error:

$$\ell_{recon} = L_1 + L_2$$  \hspace{1cm} (11)

The hypothesis is that for a small number of primitives the above losses encourage the predicted primitives to fit the input surface. Since the fitting is done using a union of convex primitives, each diagonal entry of the matrix $W_m$ in Eq. 8 should have higher weights to sets of points that belong to convex regions, thereby resulting in a convex (or approximately convex) segmentation of a point cloud. The point representations learnt in this manner are helpful for point cloud segmentation as shown in Table 2.

**Intersection loss.** To encourage spatially compact clusters we introduce a loss function that penalizes overlap between ellipsoids. Note that the clustering objective does not guarantee this as they operate on an abstract embedding space. Specifically, for each point $p$ sampled inside the surface of predicted shape should be contained inside a single primitive. Alternatively the corresponding primitive should have negative signed distance $S_m(p)$ at that point $p$, whereas the the signed distance (SD) should be positive for the remaining primitives. Let $\mathcal{V}_m$ be the set of points sampled inside the primitive $m$. Then intersection loss is defined as

$$\ell_{inter} = \sum_m \sum_{p \in \mathcal{V}_m} \sum_{j \neq m} [S_j(p)]_+^2,$$  \hspace{1cm} (12)

where $[S_m(p)]_+ = \min(S_m(p), 0)$ includes only the points with negative SD, as points with positive SD are outside the primitive and do not contribute in intersection. We use a differentiable approximation of the SD function of an ellipsoid proposed in [1].

**Similarity loss.** We found that all the per-point embedding are similar at initialization resulting in a mode collapse of the clusters. This local minima can be avoided by spreading the point embeddings across the space. We add a small penalty only at early stage of training that minimizes the similarity of output point embeddings $Y$ from mean-shift iterations (Eq. 4) as follows:

$$\ell_{sym} = \sum_{i \neq j} (1 + y_i y_j^T)^2$$  \hspace{1cm} (13)

3.4 Training details

We train our network jointly using both a self-supervised loss and a supervised loss. We alternate between our self-supervised training while sampling point clouds from the entire unlabeled $\mathcal{X}_u$ dataset and supervised training while taking limited samples from the labeled $\mathcal{X}_l$ set.

$$L = \sum_{\mathcal{X} \sim \mathcal{X}_u} \ell_{recon} + \lambda_1 \ell_{inter} + \lambda_2 \ell_{sym} + \sum_{\mathcal{X} \sim \mathcal{X}_l} \ell_{recon}$$

where $\ell_{ce}$ is a cross entropy loss, $\lambda_1$ and $\lambda_2$ are constants.

**Back-propagation and numerical stability**

- To back-propagate the gradients through SVD computation we use analytic gradients derived by Ionescu et al. [27]. When back-propagating gradients through SVD, gradients can go to infinity when singular values are indistinct. This happens when membership weights in a cluster are concentrated on a line, point or sphere. We implemented a custom Pytorch layer following [25], that changes the term $K = \frac{1}{\sigma_i - \sigma_j}$ from Eq: 13 in [21] to $K = \frac{1}{\sign(\sigma_i - \sigma_j)(\max(|\sigma_i - \sigma_j|, \epsilon))}$ with $\epsilon = 1e - 6$. SVD computation can still be unstable when the condition number of the input matrix is large. In this case, we remove that cluster from the backward pass when condition number is greater than $1e5$.

- **Differentiability of mean shift clustering procedure:** The membership matrix $W \in \mathbb{R}^{N \times M}$ is constructed by doing non-max suppression (NMS) over output $Y$ of mean shift iteration. The derivative of NMS w.r.t embeddings $Y$ is either zero or undefined, thereby making it non-differentiable. Thus we remove NMS from the computation graph and back-propagate through the rest of the graph, which is differentiable through Eq. 5. This can be seen as a straight-through estimator [41], which has been used in previous shape parsing works [25, 43].

We will release code of our implementation upon acceptance of the manuscript.

4. Experiments

4.1 Datasets

As a source of unlabeled data for the task of self supervision, we use the Shapenet Core dataset [7], which consists of 55 categories with 55,447 meshes in total. We sample these meshes uniformly to get 2048 points per shape. For the task of few-shot semantic segmentation, we use the Shapenet Semantic Segmentation dataset, which consists of 16,881 labeled point clouds across 16 shape categories with total 50 part categories.
Figure 3. Visualization of predicted semantic labels and ellipsoids on the Shapenet dataset. **Top:** Ground truth point clouds, **middle:** predicted labels using our fitting approach, trained using \( k = 10 \) labeled examples per category, **bottom:** predicted ellipsoids. **SURFit** predicts variable number of ellipsoids to approximate the input point cloud while maintaining correspondence with semantic parts.

| Samples/cls. | \( k=1 \) | \( k=3 \) | \( k=5 \) | \( k=10 \) | \( k=20 \) | \( k=50 \) | \( k=100 \) | \( k=max \) |
|--------------|----------|----------|----------|----------|----------|----------|----------|----------|
| Baseline     | 53.15 ± 2.4 | 59.54 ± 1.4 | 68.14 ± 0.9 | 71.32 ± 0.5 | 75.22 ± 0.8 | 78.79 ± 0.4 | 79.67 ± 0.3 | **81.40 ± 0.4** |
| **SURFit**   | **63.14 ± 3.4** | **71.24 ± 1.3** | **73.75 ± 0.7** | **75.03 ± 0.9** | **76.73 ± 0.5** | **79.28 ± 0.2** | **80.16 ± 0.2** | **80.40 ± 0.1** |

Table 1. **Few-shot segmentation on the ShapeNet dataset** (class avg. IoU over 5 rounds). The number of shots or samples per class is denoted by \( k \) for each of the 16 ShapeNet shape categories used for supervised training. Our proposed method **SURFit** consistently outperforms the baselines.

Figure 4. **Visualization of various primitive fitting approaches.** **a)** input point cloud. **b)** Ellipsoid fitting using our approach. **c)** cuboid fitting using our approach. **d)** different primitives from AtlasNet. Different colors are used to depict different primitives. For AtlasNet we visualize each chart with a unique color. Notice that geometric primitives are better localized and approximate the shape in fewer primitives in comparison to AtlasNet.

We also evaluate our method on the PartNet dataset [33]. This dataset provides fine-grained semantic segmentation annotation for various 3D shape categories, unlike the more coarse-level shape parts in the ShapeNet dataset. We use 12 categories from “level-3”, which denotes the finest level of segmentation. For training different approaches in few-shot framework, we remove test shapes of labeled dataset \( \mathcal{X}_l \) from our self-supervision dataset \( \mathcal{X}_u \). This avoids train-test set overlap.

### 4.2. Few-shot part segmentation on Shapenet

For each category from the ShapeNet part segmentation dataset, we randomly sample \( k \) labeled shapes (16 × \( k \) in total) and use these for training the semantic segmentation component of our model. We use the entire training set from the ShapeNet core dataset for self-supervision task. We train a single model on all 16 shape categories of ShapeNet.

**Baselines.** Our first baseline takes the PointNet++ as the base architecture and trains it from scratch on only labeled training examples, using \( k \) labeled shapes per category in a few-shot setup. Second, we create a reconstruction baseline – we use a PointNet++ as a shared feature extractor, which extracts a global shape encoding that is input to an AtlasNet decoder [18] with 25 charts. A separate decoder is used to predict per-point semantic labels. This network is trained using a Chamfer distance-based reconstruction loss using the entire unlabeled training set and using \( k \) labeled training examples. We use 5 different rounds with sampled labeled sets at various values of \( k \) and report their average
Figure 5. **Analysis of clustering.** We analyze two clustering approaches, 1) **SURFit** and 2) directly clustering **points** using K-Means. Top: normalized mutual information (NMI) and bottom: precision vs recall between predicted clusters and semantic part labels. **SURFit** gives higher average NMI (54.3 vs 35.4) and higher precision than clustering with only points as features.

**Discussion of results.** Table 1 shows results on few-shot segmentation at different number of labeled examples. Our method **SURFit** performs better than the supervised baseline showing the effectiveness of our method as self-supervision task.

In Table 2 we compare our approach with previous methods using instance IOU and class IOU [38], using 1% and 5% of the labeled training set to train different methods. Note that instance IOU is highly influenced by the shape categories with large number of testing shapes e.g. Chair. **SURFit**, on the other hand gives equal importance to all categories, hence it is a more robust evaluation metric. **SURFit** with ellipsoid as primitive outperforms previous methods including **SURFit** baseline with AtlasNet. Interestingly, **SURFit** with AtlasNet baseline outperforms all previous baselines except ACD. We speculate that since primitives predicted by AtlasNet are highly overlapped and less localized in comparison to our ellipsoid and cuboid fitting approaches as shown in Figure 4, this results in worse performance of AtlasNet. **SURFit** also performs better than ACD. This shows that our approach of primitive fitting in an end-to-end trainable manner is better than training a network using contrastive learning guided by approximate convex decomposition of water-tight meshes as proposed by ACD.

In Figure 3 we show predicted semantic labels using **SURFit** along with fitted ellipsoids. In Figure 4 we show outputs of various self supervision techniques using primitive fitting. We experimented with both cuboid and ellipsoid fitting as a self supervision task. We observed that both performs similar qualitatively and quantitatively. The fitted primitives using ellipsoid/cuboid fitting approaches are more aligned with different parts of the shape in comparison to the outputs of AtlasNet.

**Effect of the size of unlabeled dataset.** In the Table 3 we show the effect of size of unlabeled dataset used for self-supervision. We observe improvement in performance of 5-shot semantic segmentation task with increase in unlabeled dataset.

**Effect of similarity loss.** Similarity loss is only used in the initial stage of training as it prevents the network converging to a local minima at this early phase. Without this procedure, our performance is similar to Baseline (training from scratch). However, using only similarity loss (without reconstruction loss) leads to worse results (44.2 mIOU) than Baseline (68.1 mIOU) on few-shot k=5 setting.

**Analysis of learned point embeddings.** In Figure 5 we quantitatively analyze the performance of clustering induced by **SURFit** and compare it with clustering obtained by running the K-Means algorithm directly on point clouds. We take 340 shapes from the Airplane category of ShapeNet for this experiment and show the histogram of Normalized Mutual Information (NMI) [44] and the precision-recall curve [11] between predicted clusters and ground truth part labels. Our approach produces clusters with higher NMI (54.3 vs 35.4), which shows better alignment of our predicted point clusters with the ground truth part labels. Our approach also produces higher precision clusters in comparison to K-Means at equivalent recall, which shows the tendency of our algorithm to over-segment a shape. To further analyze the consistency of learned point embedding across shapes, we use TSNE [48] to visualize point embedding by projecting them to 3D color space. We use a fixed number of shapes for each category and run TSNE on each category separately. Figure 6 shows points belonging to same semantic parts are consistently projected to similar colors, further confirming the consistency of learned embeddings.

**4.3. Few-shot semantic segmentation on PartNet**

Here we experiment on the PartNet dataset for the task of few-shot semantic segmentation. For each category from this dataset, we randomly sample k labeled shapes and use these for training semantic segmentation part of the architecture. Similar to our previous experiment, we use the complete training shapes from the ShapeNet Core dataset.
Figure 6. **TSNE visualization of learned embeddings.** For each shape category, we take a fixed number of shapes and extract point embeddings from SURFIT. We run TSNE on each category separately to project the 128-D embeddings to 3D color space. Notice that points belonging to same semantic parts are colored similarly, which indicates the consistency of learned embeddings.

| Method                  | 1% IoU | 5% IoU | 1% cls. IoU | 5% cls. IoU |
|-------------------------|--------|--------|-------------|-------------|
| SO-Net [23]             | 64.0   | 69.0   | -           | -           |
| PointCapsNet [54]       | 67.0   | 70.0   | -           | -           |
| MortonNet [45]          | -      | 77.1   | -           | -           |
| JointSSL [2]            | 71.9   | 77.4   | -           | -           |
| Multi-task [19]         | 68.2   | 77.7   | -           | 72.1        |
| ACD [13]                | 75.1   | 78.6   | 74.6        | 77.5        |
| SURFIT w/ Atlas         | 73.8   | 78.6   | 74.5        | 78.9        |
| SURFIT w/ Cuboid        | 75.2   | 78.6   | 74.6        | 78.6        |
| SURFIT w/ Ellipsoid     | **75.4** | **78.7** | **75.3** | **79.0**   |

Table 2. **Comparison with state-of-the-art few-shot part segmentation methods on ShapeNet.** Performance is evaluated using instance-averaged and class-averages IoU.

| Method                  | Shapenet | PartNet |
|-------------------------|----------|---------|
|                          | cls. IoU | part avg. IoU |
| k=1%                    | k=5%     | k=10    | k=20     |
| SURFIT w/ Ellipsoids    | 75.3     | 79.0    | 28.9     | 32.1     |
| SURFIT w/ Ellipsoids+inter | 75.0   | 79.0    | **29.4** | **32.6** |

Table 3. **Effect of the size of unlabeled dataset used for self-supervision on 5-shot semantic segmentation on ShapeNet.**

Table 4 shows part avg. IoU for the different methods. The AtlasNet method shows improvement over the purely-supervised baseline. SURFIT shows improvement over both AtlasNet and Baseline, indicating the effectiveness of our approach in the fine-grained semantic segmentation setting for the self-supervision task. We choose DGCNN as a backbone architecture for this experiment. Unlike our Shapenet experiments, in the PartNet experiment we train a separate model for each category, as done in the original paper [33].

Similar to our previous experiment, we create two baselines – 1) we train a network from scratch providing only k labeled examples, and 2) we train the AtlasNet on the entire unlabeled training set using the self-supervised reconstruction loss and only k labeled examples as supervision.

Table 4 shows part avg. IoU for the different methods. The AtlasNet method shows improvement over the purely-supervised baseline. SURFIT shows improvement over both AtlasNet and Baseline, indicating the effectiveness of our approach in the fine-grained semantic segmentation setting as well. We further experiment with cuboids as the primitive, which achieves similar performance as using ellipsoid primitives, consistent with our previous ShapeNet results.

**Effect of intersection loss.** We also experiment with adding intersection loss while training SURFIT with ellipsoid primitive in Table 5. Intersection loss gives improvement only in PartNet dataset. We speculate that since PartNet contains fine-grained segmentation of shapes, minimizing overlap between primitives here is more helpful than in Shapenet dataset which contains only coarse level of segmentation. Table 2 and 4 shows best performing approach.
5. Conclusion

We have proposed a simple self-supervision task for learning point embeddings – learning to fit an unlabeled point-cloud using a set of geometric primitives – both complex such as deformed planes as used by AtlasNet, and simple such as ellipsoids and cuboids. We provide an end-to-end trainable framework for incorporating this task into standard network architectures for point cloud segmentation. Our method can be readily applied to existing architectures for semantic segmentation and shows improvements over fully-supervised baselines and other self-supervised approaches on the well-known ShapeNet and PartNet datasets, demonstrating that learning to reconstruct a shape using a set of primitives can indeed induce representations useful for discriminative downstream tasks. On the limitation side, we primarily rely on basic primitives i.e. ellipsoid and cuboids, but other primitives could be useful to capture part variability etc.

References

[1] Ellipsoid SDF. https://www.iquilezles.org/www/articles/ellipsoids/ellipsoids.htm. Accessed: 2020-11-15. 4, 5
[2] Antonio Allegro, Davide Boscaini, and Tatiana Tomas. Joint supervised and self-supervised learning for 3d real-world challenges, 2020. 2, 8
[3] Irving Biederman. Recognition-by-components: a theory of human image understanding. Psychological review, 94(2):115, 1987. 2
[4] I Binford. Visual perception by computer. In IEEE Conference of Systems and Control, 1971. 2
[5] Thomas O Binford, Tod S Levitt, and Wallace B Mann. Bayesian inference in model-based machine vision. In Proceedings of the Third Conference on Uncertainty in Artificial Intelligence, pages 86–97, 1987. 2
[6] Jonathan C Carr, Richard K Beatson, Jon B Cherrie, Tim J Mitchell, W Richard Fright, Bruce C McCallum, and Tim R Evans. Reconstruction and representation of 3d objects with radial basis functions. In Proceedings of the 28th annual conference on Computer graphics and interactive techniques, pages 67–76, 2001. 2
[7] Angel X. Chang, Thomas A. Funkhouser, Leonidas J. Guibas, Pat Hanrahan, Qi-Xing 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. CoRR, abs/1512.03012, 2015. 2, 3, 5
[8] Zhiqin Chen, Kangxue Yin, Matthew Fisher, Siddhartha Chaudhuri, and Hao Zhang. BAE-NET: branched autoencoder for shape co-segmentation. In Proceedings of the IEEE International Conference on Computer Vision, pages 8490–8499, 2019. 2
[9] Boyang Deng, Kyle Genova, Soroosh Yazdani, Sofien Bouaziz, Geoffrey Hinton, and Andrea Tagliasacchi. Cvxnet: Learnable convex decomposition. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2020. 2
[10] Shu-Cherng Fang and Sarat Puthenpura. Linear Optimization and Extensions: Theory and Algorithms. Prentice-Hall, Inc., USA, 1993. 4
[11] E. B. Fowlkes and C. L. Mallows. A method for comparing two hierarchical clusterings. Journal of the American Statistical Association, 78(383):553–569, 1983. 7
[12] Matheus Gadela, Giorgio Gori, Duygu Ceylan, Radomir Mech, Nathan Carr, Tamy Boubekeur, Rui Wang, and Subhransu Maji. Learning generative models of shape handles. In IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2020. 3
[13] Matheus Gadela, 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. 2, 8
[14] Matheus Gadela, Rui Wang, and Subhransu Maji. Multiresolution Tree Networks for 3D Point Cloud Processing. In ECCV, 2018. 2
[15] Ran Gal, Olga Sorkine, Niloy J. Mitra, and Daniel Cohen-Or. iwires: An analyze-and-edit approach to shape manipulation. ACM Transactions on Graphics (Siggraph), 28(3):#33, 1–10, 2009. 2
[16] Kyle Genova, Forrester Cole, Daniel Vlasic, Aaron Sarna, William T Freeman, and Thomas Funkhouser. Learning shape templates with structured implicit functions. In International Conference on Computer Vision, 2019. 2, 3
[17] Giorgio Gori, Alla Sheffer, Nicholas Vining, Enrique Rosales, Nathan Carr, and Tao Ju. Flowrep: Descriptive curve networks for free-form design shapes. ACM Transaction on Graphics, 36(4), 2017. 2
[18] Thibault Groueix, Matthew Fisher, Vladimir G. Kim, Bryan Russell, and Mathieu Aubry. AtlasNet: A Papier-Mâché Approach to Learning 3D Surface Generation. In Proceedings IEEE Conf. on Computer Vision and Pattern Recognition (CVPR), 2018. 2, 4, 6
[19] Kaveh Hassani and Mike Haley. Unsupervised multi-task feature learning on point clouds. In Proceedings of the IEEE International Conference on Computer Vision, pages 8160–8171, 2019. 8
[20] Donald D Hoffman and Whitman Richards. Parts of recognition. 1983. 2
[21] C. Ionescu, O. Vantzos, and C. Sminchisescu. Matrix backpropagation for deep networks with structured layers. In 2015 IEEE International Conference on Computer Vision (ICCV), pages 2965–2973, 2015. 5
[22] Shu Kong and Charless C Fowlkes. Recurrent pixel embedding for instance grouping. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pages 9018–9028, 2018. 4
[23] Jiaxin Li, Ben M Chen, and Gim Hee Lee. So-net: Self-organizing network for point cloud analysis. In Proceedings of the IEEE conference on computer vision and pattern recognition, pages 9397–9406, 2018. 8
[24] Jiaxin Li, Ben M Chen, and Gim Hee Lee. So-net: Self-organizing network for point cloud analysis. arXiv preprint arXiv:1803.04249, 2018. 2
[55] Yang Zhou, Kangxue Yin, Hui Huang, Hao Zhang, Minglun Gong, and Daniel Cohen-Or. Generalized cylinder decomposition. *ACM Trans. Graph.*, 34(6), 2015.