

SGPN: Similarity Group Proposal Network for 3D Point Cloud Instance Segmentation

Weiyue Wang¹ Ronald Yu²
¹University of Southern California
 Los Angeles, California
 {weiyuewa, qianguih, uneumann}@usc.edu

Qiangui Huang¹ Ulrich Neumann¹
²University of California, San Diego
 San Diego, California
 ronaldiscool@gmail.com

Abstract

We introduce *Similarity Group Proposal Network* (SGPN), a simple and intuitive deep learning framework for 3D object instance segmentation on point clouds. SGPN uses a single network to predict point grouping proposals and a corresponding semantic class for each proposal, from which we can directly extract instance segmentation results. Important to the effectiveness of SGPN is its novel representation of 3D instance segmentation results in the form of a similarity matrix that indicates the similarity between each pair of points in embedded feature space, thus producing an accurate grouping proposal for each point. To the best of our knowledge, SGPN is the first framework to learn 3D instance-aware semantic segmentation on point clouds. Experimental results on various 3D scenes show the effectiveness of our method on 3D instance segmentation, and we also evaluate the capability of SGPN to improve 3D object detection and semantic segmentation results. We also demonstrate its flexibility by seamlessly incorporating 2D CNN features into the framework to boost performance.

1. Introduction

Instance segmentation on 2D images have achieved promising results recently [18, 10, 29, 23]. With the rise of autonomous driving and robotics applications, the demand for 3D scene understanding and the availability of 3D scene data has rapidly increased in recently. Unfortunately, the literature for 3D instance segmentation and object detection lags far behind its 2D counterpart; scene understanding with Convolutional Neural Networks (CNNs) [42, 43, 11] on 3D volumetric data is limited by high memory and computation cost. Recently, deep learning frameworks PointNet/Pointnet++ [31, 33] on point clouds open up more efficient and flexible ways to handle 3D data.

Following the pioneering works in 2D scene understanding, our goal is to develop a novel deep learning framework trained *end-to-end* for 3D instance-aware semantic segmen-

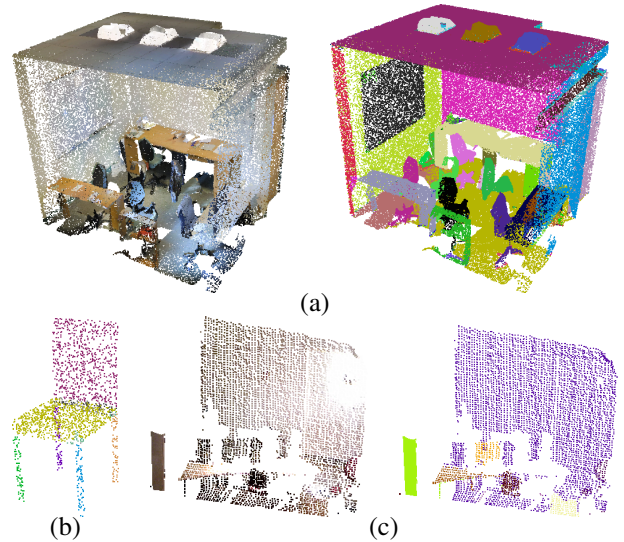


Figure 1: Instance segmentation for point clouds using SGPN. Different colors represent different instances. (a) Instance segmentation on complete real scenes. (b) Single object part instance segmentation. (c) Instance segmentation on point clouds obtained from partial scans.

tation on point clouds that, like established baseline systems for 2D scene understanding tasks, is *intuitive, simple, flexible, and effective*.

An important consideration for instance segmentation on a point cloud is how to represent output results. Inspired by the trend of predicting proposals for tasks with a variable number of outputs, we introduce a *Similarity Group Proposal Network* (SGPN), which formulates *group proposals* of object instances by learning a novel 3D instance segmentation representation in the form of a *similarity matrix*.

Our pipeline first uses PointNet/PointNet++ to extract a descriptive feature vector for *each point* in the point cloud. As a form of similarity metric learning, we enforce the idea that points belonging to the same object instance should have very similar features; hence we measure the distance between the features of each pair of points in order to form

a *similarity matrix* that indicates whether any given pair of points belong to the same object instance.

The rows in our similarity matrix can be viewed as instance candidates, which we combine with learned confidence scores in order to generate plausible *group proposals*. We also learn a semantic segmentation map in order to classify each object instance obtained from our group proposals. We are also able to directly derive tight 3D bounding boxes for object detection.

By simply measuring the distance between overdetermined feature representations of each pair of points, our *similarity matrix* simplifies our pipeline in that we remain in the natural point cloud representation of defining our objects by the relationships between points.

In summary, SGPN has three output branches for instance segmentation on point clouds: a *similarity matrix* yielding point-wise group proposals, a *confidence map* for pruning these proposals, and a *semantic segmentation map* to give the class label for each group. SGPN is to the best of our knowledge the first deep learning framework to learn 3D instance segmentation on point clouds.

We evaluate our framework on both 3D shapes (ShapeNet [4]) and real 3D scenes (Stanford Indoor Semantic Dataset [1] and NYUV2 [40]) and demonstrate that SGPN achieves state-of-the-art results on 3D instance segmentation. We also conduct comprehensive experiments to show the capability of SGPN on achieving high performance on 3D semantic segmentation and 3D object detection on point clouds. Although a minimalistic framework with no bells and whistles already gives visually pleasing results (Figure 1), we also demonstrate the flexibility of SGPN as we boost performance even more by seamlessly integrating CNN features from RGBD images.

2. Related Works

2.1. Object Detection and Instance Segmentation

Recent advances in object detection [37, 14, 24, 35, 36, 26, 13, 25] and instance segmentation [23, 10, 9, 30, 29] on 2D images have achieved promising results. R-CNN [15] for 2D object detection established a baseline system by introducing region proposals as candidate object regions. Faster R-CNN [37] leveraged a CNN learning scheme and proposed Region Proposal Networks(RPN). YOLO [35] divided the image into grids and each grid cell produced an object proposal. Many 2D instance segmentation approaches are based on segment proposals. DeepMask [29] learns to generate segment proposals each with a corresponding object score. Dai et al. [10] predict segment candidates from bounding box proposals. Mask R-CNN [18] extended Faster R-CNN by adding a branch on top of RPN to produce object masks for instance segmentation.

Following these pioneering 2D works, 3D bounding box

detection frameworks have emerged [38, 42, 43, 11, 5]. Song and Xiao [43] use a volumetric CNN to create 3D RPN on a voxelized 3D scene and then use both the color and depth data of the image in a joint 3D and 2D object recognition network on each proposal. Deng and Latecki [11] regress class-wise 3D bounding box models based on RGBD image appearance features only. Armeni et al [1] use a sliding shape method with CRF to perform 3D object detection on point cloud. To the best of our knowledge, no previous work exists that learns 3D instance segmentation.

2.2. 3D Deep Learning

Convolutional neural networks generalize well to 3D by performing convolution on voxels for certain tasks such as object classification [32, 46, 27, 49, 39], shape reconstruction [47, 17, 8] of simple objects, and 3D object detection as mentioned in Section 2.1. However, volumetric representation carry a high memory and computational cost and have strong limitations dealing with 3D scenes [7, 1, 44]. Octree-based CNNs [39, 45, 46] have been introduced recently, but they are less flexible than volumetric CNNs and still suffer from memory efficiency problems.

A point cloud is an intuitive, memory-efficient 3D representation well-suited for representing detailed, large scenes for 3D instance segmentation using deep learning. PointNet/Pointnet++ [31, 33] recently introduce deep neural networks on 3D point clouds, learning successful results for tasks such as object classification and part and semantic scene segmentation. We base our network architecture off of PointNet/PointNet++, achieving a novel method that learns 3D instance segmentation on point clouds.

2.3. Similarity Metric Learning

Our work is also closely related to similarity metric learning, which has been widely used in deep learning on various tasks such as person re-identification [50], matching [16], image retrieval [12, 48] and face recognition [6]. Siamese CNNs [6, 41, 3] are used on tasks such as tracking [22] and one-shot learning [20] by measuring the similarity of two input images. Alejandro et. al [28] introduced an associative embedding method to group similar pixels for multi-person pose estimation and 2D instance segmentation by enforcing that pixels in the same group should have similar values in their embedding space without actually enforcing what those exact values should be. Our method exploits metric learning in a different way in that we regress the likelihood of two points belonging to the same group and formulate the similarity matrix as group proposals to handle variable number of instances.

3. Method

The goal of this paper is to take a 3D point cloud as input and produce an object instance label for each point and

a class label for each instance. Utilizing recent developments in deep learning on point clouds [31, 33], we introduce a Similarity Group Proposal Network (SGPN), which consumes a 3D point cloud and outputs a set of instance proposals that each contain the group of points inside the instance as well as its class label. Section 3.1 introduces the design and properties of SGPN. Section 3.2 proposes an algorithm to merge similar groups and give each point an instance label. Section 3.3 gives implementation details. Figure 2 depicts the overview of our system.

3.1. Similarity Group Proposal Network

SGPN is a very *simple* and *intuitive* framework. As shown in Figure 2, it first passes a point cloud P of size N_p through a feed-forward feature extraction network inspired by PointNets [31, 33], learning both global and local features in the point cloud. This feature extraction network produces a matrix F . SGPN then diverges into three branches that each pass F through a single PointNet layer to obtain sized $N_p \times N_f$ feature matrices F_{SIM}, F_{CF}, F_{SEM} , which we respectively use to obtain a *similarity matrix*, a *confidence map* and a *semantic segmentation map*. The i th row in a $N_p \times N_f$ feature matrix is a N_f -dimensional vector that represents point P_i in an embedded feature space. Our loss L is given by the sum of the losses from each of these three branches: $L = L_{SIM} + L_{CF} + L_{SEM}$. Our network architecture can be found in the supplemental.

Similarity Matrix We propose a novel *similarity matrix* S from which we can formulate group proposals to directly recover accurate instance segmentation results. S is of dimensions $N_p \times N_p$, and element S_{ij} classifies whether or not points P_i and P_j belong to the same object instance. Each row of S can be viewed as a proposed grouping of points that form a candidate object instance.

We leverage that points belonging to the same object instance should have similar features and lie very close together in feature space. We obtain S by, for each pair of points $\{P_i, P_j\}$, simply subtracting their corresponding feature vectors $\{F_{SIM_i}, F_{SIM_j}\}$ and taking the L_2 norm such that $S_{ij} = \|F_{SIM_i} - F_{SIM_j}\|_2$. This reduces the problem of instance segmentation to learning an embedding space where points in the same instance are close together and those in different object instances are far apart.

For a better understanding of how SGPN captures correlation between points, in Figure 3(a) we visualize the similarity (euclidean distance in feature space) between a given point and the rest of the points in the point cloud. Points in different instances have greater euclidean distances in feature space and thus smaller similarities even though they have the same semantic labels. For example, in the bottom-right image of Figure 3(a), although the given table leg point has greater similarity with the other table leg points than the

table top, it is still distinguishable from the other table leg.

We believe that a *similarity matrix* is a more natural and simple representation for 3D instance segmentation on a point cloud compared to traditional 2D instance segmentation representations. Most state-of-the-art 2D deep learning methods for instance segmentation first localize the image into patches, which are then passed through a neural network and segment a binary mask of the object.

While learning a binary mask in a bounding box is a more natural representation for *space-centric* structures such as images or volumetric grids where features are largely defined by which positions in a grid have strong signals, point clouds can be viewed as *shape-centric* structures where information is encoded by the relationship between the points in the cloud, so we would prefer to also define instance segmentation output by the relationship between points without working too much in grid space.

Hence we expect that a deep neural network could better learn our *similarity matrix*, which compared to traditional representations is a *more natural and straightforward* representation for instance segmentation in a point cloud.

Double-Hinge Loss for Similarity Matrix As is the case in [28], in our similarity matrix we do not need to precisely regress the exact values of our features; we only optimize the simpler objective that similar points should be close together in feature space. We define three potential similarity classes for each pair of points $\{P_i, P_j\}$:

1. P_i and P_j belong to the same object instance.
2. P_i and P_j share the same semantic class but do not belong to the same object instance.
3. P_i and P_j do not share the same semantic class.

Pairs of points should lie progressively further away from each other in feature space as their similarity class increases. We define our loss as:

$$L_{SIM} = \sum_i \sum_j^{N_p} l(i, j)$$

$$l(i, j) = \begin{cases} \|F_{SIM_i} - F_{SIM_j}\|_2 & C_{ij} = 1 \\ \alpha \max(0, K_1 - \|F_{SIM_i} - F_{SIM_j}\|_2) & C_{ij} = 2 \\ \max(0, K_2 - \|F_{SIM_i} - F_{SIM_j}\|_2) & C_{ij} = 3 \end{cases}$$

where C_{ij} indicates which of the similarity classes defined above does the pair of points $(\{P_i, P_j\})$ belong to and α, K_1, K_2 are constants such that $\alpha > 1, K_2 > K_1$.

Although the second and third similarity class are treated equivalently for the purposes of instance segmentation, distinguishing between them in L_{SIM} using our double-hinge loss allows our similarity matrix output branch and our semantic segmentation output branch to mutually assist each

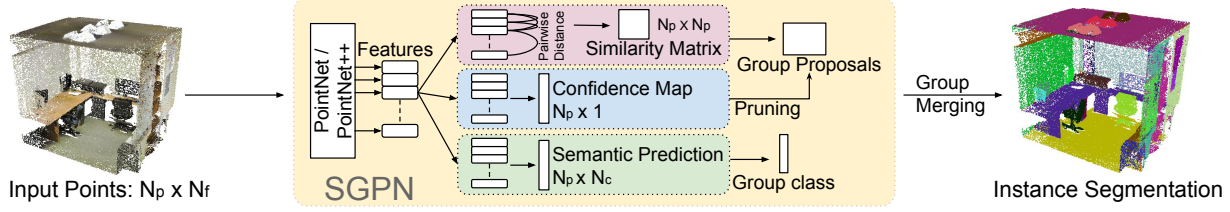


Figure 2: Pipeline of our system for point cloud instance segmentation.

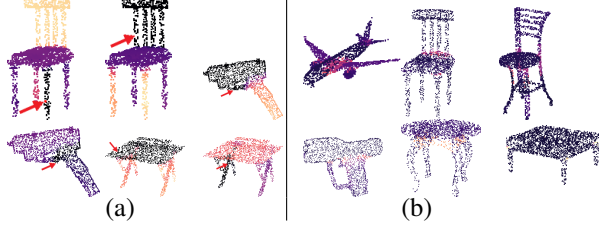


Figure 3: (a) Similarity (euclidean distance in feature space) between a given point (indicated by red arrow) and the rest of points. A darker color represents lower distance in feature space thus higher similarity. (b) Confidence map. A darker color represents higher confidence.

other for increased accuracy and convergence speed. Since the semantic segmentation network is actually wrongly trying to bring pairs of points in our second similarity class closer together in feature space, we also add an $\alpha > 1$ term to increase the weight of our loss to dominate the gradient from the semantic segmentation output branch.

At test time if $S_{ij} < Th_S$ where $Th_S < K_1$, then points pair P_i and P_j are in the same instance group.

Similarity Confidence Network SGPN also feeds F_{CF} through an additional PointNet layer to predict a $N_p \times 1$ confidence map CM reflecting how confidently the model believes that each grouping candidate is indeed a correct object instance. Figure 3(b) provides a visualization of the confidence map; points located in the boundary area between parts have lower confidence.

We regress confidence scores based on ground truth groups G represented as a $N_p \times N_p$ matrix identical in form to our similarity matrix. If P_i is a background point that does not belong to any object in the ground truth then the row G_i will be all zeros. For each row in S_i , we expect the ground-truth value in the confidence map CM_i to be the intersection over union (IoU) between the set of points in the predicted group S_i and the ground truth group G_i . Our loss L_{CF} is the L2 loss between the inferred and expected CM .

Although the loss L_{CF} depends on the similarity matrix output branch during training, at test time we run the branches in parallel and only groups with confidence greater than a threshold Th_C are considered valid group proposals.

Semantic Segmentation Map The semantic segmentation map acts as a point-wise classifier. SGPN passes F_{SEM}

through an additional PointNet layer whose architecture depends on the number of possible semantic classes, yielding the final output M_{SEM} , which is a $N_p \times N_C$ sized matrix where N_C is the number of possible object categories. $M_{SEM_{ij}}$ corresponds to the probability that point P_i belongs to class C_j .

The loss L_{SEM} is a weighted sum of the cross entropy softmax loss for each row in the matrix. We use median frequency balancing [2] and the weight assigned to a category is $ac = medianfreq / freq(c)$, where $freq(c)$ is the total number of points of class c divided by the total number of points in samples where c is present, and $medianfreq$ is the median of these $freq(c)$.

At test time, the class label for a group instance is assigned by calculating the mode of the semantic labels of the points in that group.

3.2. Group Proposal Merging

The similarity matrix S produces N_p group proposals, many of which are noisy or represent the same object. We first discard proposals with predicted confidence less than Th_C or cardinality less than Th_{M2} . We further prune our proposals into clean, non-overlapping object instances by applying Non-Maximum Suppression; groups with IoU greater than Th_{M1} are merged together by selecting the group with the maximum cardinality.

Each point is then assigned to the group proposal that contains it. In the rare case (2%) that after the merging stage a point belongs to more than one final group proposal, this usually means that the point is at the boundary between two object instances, which means that the effectiveness of our network would be roughly the same regardless of which group proposal the point is assigned to. Hence, with minimal loss in accuracy we randomly assign the point to any one of the group proposals that contains it. We refer to this process as *GroupMerging* throughout the rest of the paper.

3.3. Implementation Details

We use an ADAM [19] optimizer with initial learning rate 0.0005, momentum 0.9 and batch size 4. The learning rate is divided by 2 every 20 epochs. The network is trained with only the L_{SIM} loss for the first 5 epochs. In our experiment, α is set to 2 initially and is increased by 2 every 5 epochs. This design makes the network more focused on separating features of points that belong to different object

instances but have the same semantic labels. K_1, K_2 are set to 10 and 80, respectively. We use per-category histogram thresholding to get the threshold point Th_s for each testing sample. Th_{M1} is set to 0.6 and Th_{M2} is set to 200. Th_C is set to 0.1. Our network is implemented with Tensorflow and a single Nvidia GTX1080 Ti GPU. It takes 16-17 hours to converge. At test time, SGPN takes 40ms on an input point cloud with size 4096×9 with PointNet++ as our baseline architecture. Further runtime analysis can be found in Section 4.2.

4. Experiments

We evaluate SGPN on 3D instance segmentation on the following datasets:

- Stanford 3D Indoor Semantics Dataset (S3DIS) [1]: This dataset contains 3D scans in 6 areas including 271 rooms. The input is a complete point cloud generated from scans fused together from multiple views. Each point has semantic labels and instance annotations.
- NYUV2 [40]: Partial point clouds are generated from single view RGBD images. The dataset is annotated with 3D bounding boxes and 2D semantic segmentation masks. We use the improved annotation in [11]. Since both 3D bounding boxes and 2D segmentation masks annotations are given, ground truth 3D instance segmentation labels for point clouds can be easily generated. We follow the standard split with 795 training images and 654 testing images.
- ShapeNet [4, 51] Part Segmentation: ShapeNet contains 16,881 shapes annotated with 50 types of parts in total. Most object categories are labeled with two to five parts. We use the official split of 795 training samples and 654 test percentage samples in our experiments.

We also show the capability of SGPN to improve semantic segmentation and 3D object detection. To validate the flexibility of SGPN, we also seamlessly incorporate 2D CNN features into our network to boost performance on the NYUV2 dataset.

4.1. S3DIS Instance Segmentation and 3D Object Detection

We perform experiments on Stanford 3D Indoor Semantic Dataset to evaluate our performance on large real scene scans. Following experimental settings in PointNet [31], points are uniformly sampled into blocks of area $1m \times 1m$. Each point is labeled as one of 13 categories (chair, table, floor, wall, clutter etc.) and represented by a 9D vector (XYZ, RGB, and normalized location as to the room). At train time we uniformly sample 4096 points in each block, and at test time we use all points in the block as input.

SGPN uses PointNet as its baseline architecture for this experiment.¹ Figure 5 shows instance segmentation results on S3DIS with SGPN. Different colors represent different instances. Point colors of the same group are not necessarily the same as their counterparts in the ground truth since object instances are unordered. To visualize instance classes, we also add semantic segmentation results. SGPN achieves good performance on various room types.

We also compare instance segmentation performance with the following method (which we call Seg-Cluster): Perform semantic segmentation using our network and then select all points as seeds. Starting from a seed point, BFS is used to search neighboring points with the same label. If a cluster with more than 200 points has been found, it is viewed as a valid group. Our *GroupMerging* algorithm is then used to merge these valid groups.

We calculate the IoU on points between each predicted and ground truth group. A detected instance is considered as true positive if the IoU score is greater than a threshold. The average precision (AP) is further calculated for instance segmentation performance evaluation. Table 1 shows the AP for every category with IoU threshold 0.5. To the best of our knowledge, there are no existing instance segmentation method on point clouds for arbitrary object categories, so we further demonstrate the capability of SGPN to handle various objects by adding the 3D detection results of Armeni et al. [1] on S3DIS to Table 1. The difference in evaluation metrics between our method and [1] is that the IoU threshold of [1] is 0.5 on a 3D bounding box and the IoU calculation of our method is on points. Despite this difference in metrics, we can still see our superior performance on both large and small objects.

We see that a naive method like Seg-Cluster tends to properly separate regions far away for large objects like the ceiling and floor. However for small object, Seg-Cluster fails to segment instances with the same label if they are close to each other. Mean APs with different IoU thresholds (0.25, 0.5, 0.75) are also evaluated in Table 2. Figure 4 shows qualitative comparison results.

Once we have instance segmentation results, we can compute the bounding box for every instance and thus produce 3D object detection predictions. In Table 3, we compare our method with the 3D object detection system introduced in PointNet [31], which to the best of our knowledge is the state-of-the-art method for 3D detection on S3DIS. Detection performance is evaluated over 4 categories AP with IoU threshold 0.5.

The method introduced in PointNet clusters points given semantic segmentation results and uses a binary classifica-

¹PointNet [31] proposed a 3D detection system while PointNet++ [33] does not. To make fair comparison, we use PointNet as our baseline architecture for this experiment while using PointNet++ in Sections 4.2 and 4.3.

| | Mean | ceiling | floor | wall | beam | column | window | door | table | chair | sofa | bookcase | board |
|-------------------|--------------|--------------|--------------|--------------|--------------|--------------|--------------|--------------|--------------|--------------|-------------|--------------|--------------|
| Armeni et al. [1] | 49.93 | 71.61 | 88.70 | 72.86 | 66.67 | 91.77 | 25.92 | 54.11 | 46.02 | 16.15 | 6.78 | 54.71 | 3.91 |
| Seg-Cluster | 20.39 | 43.58 | 35.52 | 16.64 | 12.59 | 15.90 | 23.86 | 15.75 | 22.63 | 10.33 | 3.92 | 43.33 | 10.71 |
| SGPN | 54.35 | 79.44 | 66.29 | 88.77 | 77.98 | 60.71 | 66.62 | 56.75 | 46.90 | 40.77 | 6.38 | 47.61 | 11.05 |

Table 1: Results on instance segmentation in S3DIS scenes. The metric is AP(%) with IoU threshold 0.5. To the best of our knowledge, there are no existing instance segmentation methods on point clouds for arbitrary object categories. The result of Armeni et al. [1] is on 3D object detection and IoU is calculated on 3D bounding boxes, while Seg-Cluster and SGPN are on points.

| | $AP_{0.25}$ | $AP_{0.5}$ | $AP_{0.75}$ |
|-------------|--------------|--------------|--------------|
| Seg-Cluster | 25.56 | 20.39 | 16.08 |
| SGPN | 59.85 | 54.35 | 43.09 |

Table 2: Comparison results on instance segmentation with different IoU thresholds in S3DIS scenes. Metric is mean AP(%) over 13 categories.

| | Mean | table | chair | sofa | board |
|---------------|--------------|--------------|--------------|-------------|--------------|
| PointNet [31] | 24.24 | 46.67 | 33.80 | 4.76 | 11.72 |
| Seg-Cluster | 18.72 | 33.44 | 22.8 | 5.38 | 13.07 |
| SGPN | 30.20 | 49.90 | 40.87 | 6.96 | 13.28 |

Table 3: Comparison results on 3D detection in S3DIS scenes. SGPN uses PointNet as baseline. The metric is AP with IoU threshold 0.5.

| | Mean IoU | Accuracy |
|---------------|--------------|--------------|
| PointNet [31] | 49.76 | 79.66 |
| SGPN | 50.37 | 80.78 |

Table 4: Results on semantic segmentation in S3DIS scenes. SGPN uses PointNet as baseline. Metric is mean IoU(%) over 13 classes (including clutter).

tion network for each category to separate close objects with same categories. Our method outperforms it by a large margin, and unlike PointNet does not require an additional network, which unnecessarily introduces additional complexity during both train and test time and local minima during train time. SGPN can effectively separate the difficult cases of objects of the same semantic class but different instances (c.f. Figure 4) since points in different instances are far apart in feature space even though they have the same semantic label. We further compare our semantic segmentation results with PointNet in Table 4. SGPN outperforms its baseline with the help of its similarity matrix.

4.2. NYUV2 Object Detection and Instance Segmentation Evaluation

We evaluate the effectiveness of our approach on partial 3D scans on the NYUV2 dataset. In this dataset, 3D point clouds are lifted from a single RGBD image. An image of size $H \times W$ can produce $H \times W$ points. We subsample this point cloud by resizing the image to $\frac{H}{4} \times \frac{W}{4}$ and get the corresponding points using a nearest neighbor search. Both our training and testing experiments are conducted on such a point cloud. PointNet++ is used as our baseline.

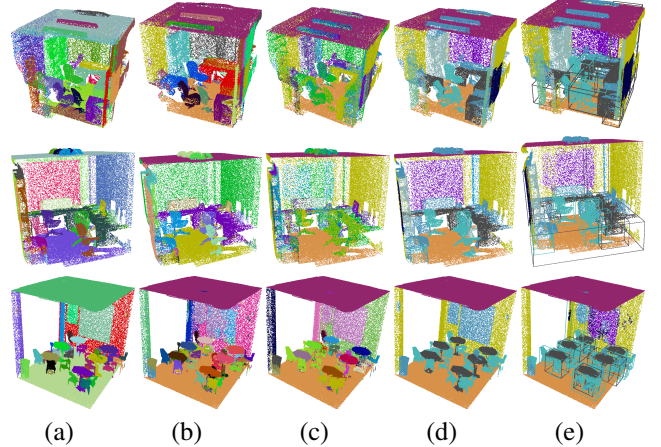


Figure 4: Comparison results on S3DIS. (a) Ground Truth for instance segmentation. Different colors represents different instances. (b) SGPN instance segmentation results. (c) Seg-Cluster instance segmentation results. (d) Ground Truth for semantic segmentation. (e) Semantic Segmentation and 3D detection results of SGPN. The color of the detected bounding box for each object category is the same as the semantic labels.

In [34], 2D CNN features are combined 3D point cloud for RGBD semantic segmentation. By leveraging the *flexibility* of SGPN, we also seamlessly integrate 2D CNN features from RGB images to boost performance. A 2D CNN consumes an RGBD map and extracts feature maps F_2 with size $\frac{H}{4} \times \frac{W}{4} \times N_{F2}$. Since there are $\frac{H}{4} \times \frac{W}{4}$ sub-sampled points for every image, a feature vector of size N_{f2} can be extracted from F_2 at each pixel location. Every feature vector is concatenated to F (a $N_p \times N_F$ feature matrix produced by PointNet/PointNet++ as mentioned in Section 3.1) for each corresponding point, yielding a feature map of size $N_p \times (N_F + N_{F2})$, which we then feed to our output branches. Figure 6 illustrates this procedure; we call this pipeline SGPN-CNN. In our experiments, we use a pre-trained AlexNet model [21] (with the first layer stride 1) and extract F_2 from the conv5 layer. We use $H \times W = 316 \times 415$ and $N_p = 8137$. The 2D CNN and SGPN are trained jointly.

Evaluation is performed on 19 object categories. Figure 7 shows qualitative results on instance segmentation of SGPN. Table 5 shows comparisons between Seg-Cluster




















| | Mean |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|-------------|-------------|---|---|---|---|---|---|---|---|---|---|--|---|---|---|---|---|---|---|---|
| Seg-Cluster | 38.8 | 43.3 | 83.9 | 28.2 | 2.9 | 53.6 | 43.0 | 41.4 | 5.4 | 49.0 | 56.4 | 24.4 | 3.1 | 30.9 | 36.1 | 68.4 | 49.3 | 32.1 | 12.2 | 74.1 |
| SGPN | 40.1 | 46.4 | 84.1 | 30.9 | 5.8 | 54.6 | 44.8 | 40.1 | 6.0 | 51.4 | 56.1 | 27.6 | 4.1 | 30.9 | 35.1 | 67.1 | 50.1 | 34.9 | 15.0 | 76.3 |
| SGPN-CNN | 43.5 | 54.4 | 83.2 | 45.9 | 7.7 | 56.6 | 43.6 | 42.0 | 5.2 | 50.1 | 54.3 | 35.4 | 5.3 | 37.8 | 40.3 | 66.6 | 59.1 | 43.6 | 18.1 | 77.6 |

Table 5: Results on instance segmentation in NYUV2. The metric is AP with IoU 0.25.



Figure 5: SGPN instance segmentation results on S3DIS. The first row is the prediction results. The second row is ground truths. Different colors represent different instances. The third row is the predicted semantic segmentation results. The fourth row is the ground truths for semantic segmentation.

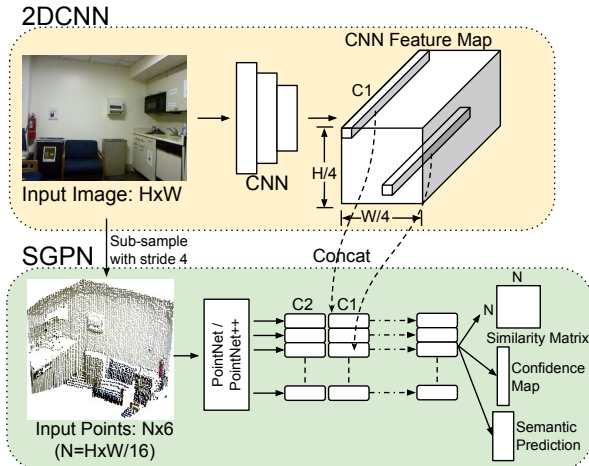


Figure 6: Incorporating CNN features in SGPN.

and our SGPN and CNN-SGPN frameworks on instance segmentation. The evaluation metric is average precision (AP) with IoU threshold 0.25.

The margin of improvement for SGPN compared to Seg-Cluster is not as high as it is on S3DIS, because in this

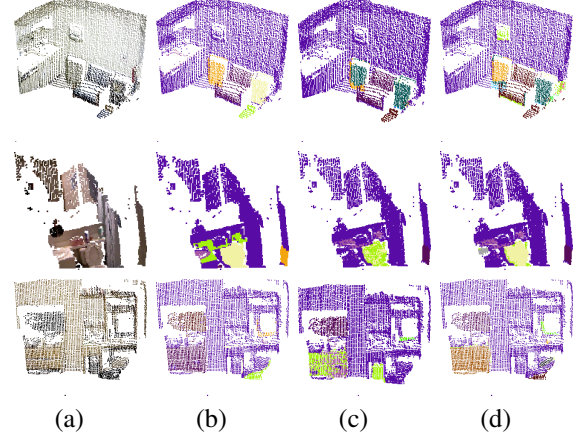


Figure 7: SGPN instance segmentation results on NYUV2. (a) Input point clouds. (b) Ground truths for instance segmentation. (c) Instance segmentation results with SGPN. (d) Instance segmentation results with SGPN-CNN.





| | Mean |  |  |  |  |
|--------------------------|--------------|---|---|---|---|
| Deep Sliding Shapes [43] | 37.55 | 58.2 | 36.1 | 27.2 | 28.7 |
| Deng and Latecki [11] | 35.55 | 46.4 | 33.1 | 33.3 | 29.4 |
| SGPN | 36.25 | 44.4 | 30.4 | 46.1 | 24.4 |
| SGPN-CNN | 41.30 | 50.8 | 34.8 | 49.4 | 30.2 |

Table 6: Comparison results on 3D detection (AP with IoU 0.25) in NYUV2. Please note we use point groups as inference while [43, 11] use large bounding box with invisible regions as ground truth. Our prediction is the tight bounding box on points which makes the IoU much smaller than [43, 11].

dataset objects with the same semantic label are usually far apart in Euclidean space. Additionally, naive methods like Seg-Cluster benefit since it is easy to separate a single instance into parts since the points are not connected due to occlusion in partial scanning. Table 5 also illustrates that SGPN can effectively utilize CNN features. Instead of concatenating fully-connected layer of 2D and 3D networks as in [43], we combine 2D and 3D features by considering their geometric relationships.

We further calculate bounding boxes with instance segmentation results. Table 6 compares our work with the state-of-the-art works [43, 11] on NYUV2 3D object detection. Following the evaluation metric in [42], AP is calculated with IoU threshold 0.25 on 3D bounding boxes. The

| | Mean | air-plane | bag | cap | car | chair | head phone | guitar | knife | lamp | laptop | motor | mug | pistol | rocket | skate board | table |
|------|-------------|-------------|-------------|-------------|-------------|-------------|-------------|-------------|-------------|-------------|-------------|-------------|-------------|-------------|-------------|-------------|-------------|
| [33] | 84.6 | 80.4 | 80.9 | 60.0 | 76.8 | 88.1 | 83.7 | 90.2 | 82.6 | 76.9 | 94.7 | 68.0 | 91.2 | 82.1 | 59.9 | 78.2 | 87.5 |
| SGPN | 85.8 | 80.4 | 78.6 | 78.8 | 71.5 | 88.6 | 78.0 | 90.9 | 83.0 | 78.8 | 95.8 | 77.8 | 93.8 | 87.4 | 60.1 | 92.3 | 89.4 |

Table 7: Semantic segmentation results on ShapeNet part dataset. Metric is mean IoU(%) on points.

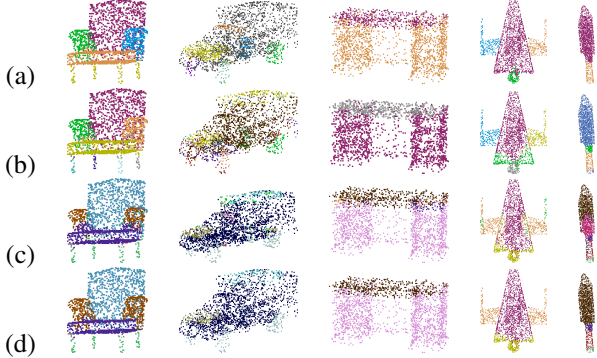


Figure 8: Qualitative results on ShapeNet Part Dataset. (a) Generated ground truth for instance segmentation. (b) SGPN instance segmentation results. (c) Semantic segmentation results of PointNet++. (d) Semantic segmentation results of SGPN.

NYUV2 dataset provides ground truth 3D bounding boxes that encapsulate the whole object including the part that is invisible in the depth image. Both [43] and [11] use these large ground truth bounding boxes for inference. In our method, we infer point groupings, which lack information of the invisible part of the object. Our output is the derived tight bounding box around the grouped points in the partial scan, which makes our IoUs much smaller than [43, 11]. However, we can still see the effectiveness of SGPN on the task of 3D object detection on partial scans as our method achieves better performance on small objects.

Computation Speed To benchmark the testing time with [43, 11] and make fair comparison, we run our framework on an Nvidia K40 GPU. SGPN takes 170ms and around 400M GPU memory per sample. CNN-SGPN takes 300ms and 1.4G GPU memory per sample. GroupMerging takes 180ms on an Intel i7 CPU. However, the detection net in [11] takes 739ms on an Nvidia Titan X GPU. In [43], RPN takes 5.62s and ORN takes 13.93s per image on an Nvidia K40 GPU. Our model improves the efficiency and reduces GPU memory usage by a large margin.

4.3. ShapeNet Part Instance Segmentation

Following the settings in [33], point clouds are generated by uniformly sampling shapes from Shapenet [4]. In our experiments we sample each shape into 2048 points. The XYZ of points are fed into network as input with size 2048×3 . To generate ground truth labels for part instance segmentation from semantic segmentation results, we perform DBSCAN clustering on each part category of an object

to group points into instances. This experiment is conducted as a toy example to demonstrate the effectiveness of our approach on instance segmentation for pointclouds.

We use Pointnet++ as our baseline. Figure 8(b) illustrates the instance segmentation results. For instance results, we again use different colors to represent different instances, and point colors of the same group are not necessarily the same as the ground truth. Since the generated ground truths are not “real” ground truths, only qualitative results are provided. SGPN achieves good results even under challenging conditions. As we can see from the Figure 8, SGPN is able to group the chair legs into four instances even though even in the ground truth DBSCAN can not separate the chair legs apart.

The similarity matrix can also help the semantic segmentation branch training. We compare SGPN to PointNet++ (i.e. our framework with solely a semantic segmentation branch) on semantic segmentation in Table 7. The inputs of both networks are point clouds of size 2048. Evaluation metric is mIoU on points of each shape category. Our model performs better than PointNet++ due to the similarity matrix. Qualitative results are shown in Figure 8. Some false segmentation prediction is refined with the help of SGPN.

5. Conclusion

We present SGPN, an intuitive, simple, and flexible framework for 3D instance segmentation on point clouds. With the introduction of the similarity matrix as our output representation, group proposals with class predictions can be easily generated from a single network. Experiments show that our algorithm can achieve good performance on instance segmentation for various 3D scenes and facilitate the tasks of 3D object detection and semantic segmentation.

Future Work While a similarity matrix provides an intuitive representation and an easily defined loss function, one limitation of SGPN is that the size of the similarity matrix scales quadratically as N_p increases. Thus, although much more memory efficient than volumetric methods, SGPN cannot process extremely large scenes on the order 10^5 or more points. Future research directions can consider generating groups using seeds that are selected based on SGPN to reduce the size of the similarity matrix. Currently trained in a fully supervised manner using a hinge loss, SGPN can also be extended in future works to learn in a more unsupervised setting or to learn more different kinds of data representations beyond instance segmentation.

A. Network Architecture

In our experiments, we use both PointNet and PointNet++ as our baseline architectures. For the S3DIS dataset, we use PointNet as our baseline for fair comparison with the 3D object detection system described in the PointNet paper [31]. The network architecture is the same as the semantic segmentation network as stated in PointNet except for the last two layers. Our F is the last 1×1 conv layer with BatchNorm and ReLU in PointNet with 256 output channels. F_{SIM}, F_{CF}, F_{SEM} are 1×1 conv layers with output channels (128, 128, 128), respectively.

For the NYUV2 dataset, we use PointNet++ as our baseline. We use the same notations as PointNet++ to describe our architecture:

$SA(K, r, [l_1, \dots, l_d])$ is a set abstraction (SA) level with K local regions of ball radius r using a PointNet architecture of d 1×1 conv layers with output channels $l_i (i = 1, \dots, d)$. $FP(l_1, \dots, l_d)$ is a feature propagation (FP) level with d 1×1 conv layers. Our network architecture is:

$SA(1024, 0.1, [32, 32, 64]),$
 $SA(256, 0.2, [64, 64, 128]),$
 $SA(128, 0.4, [128, 128, 256]),$
 $SA(64, 0.8, [256, 256, 256]),$
 $SA(16, 1.2, [256, 256, 512]),$
 $FP(512, 256),$
 $FP(256, 256),$
 $FP(256, 256),$
 $FP(256, 128),$
 $FP(128, 128, 128, 128).$

F_{SIM}, F_{CF}, F_{SEM} are 1×1 conv layers with output channels (128, 128, 128) respectively.

For our experiments on the ShapeNet part dataset, PointNet++ is used as our baseline. We use the same network architecture as in the PointNet++ paper [33]. F_{SIM}, F_{CF}, F_{SEM} are 1×1 conv layers with output channels (64, 64, 64), respectively.

B. Experiment Settings

B.1. S3DIS Dataset

Block Merging We divide each scene into $1m \times 1m$ blocks with overlapping sliding windows in a snake pattern of stride $0.5m$ as is shown in Figure 9. The entire scene is also divided into a $400 \times 400 \times 400$ grid V . V_k is used to indicate the instance label of cell k where $k \in [0, 400 \times 400 \times 400)$. Given V and point instance labels for each block PL where PL_{ij} represents the instance label of j th point in block i , a *BlockMerging* algorithm (refer

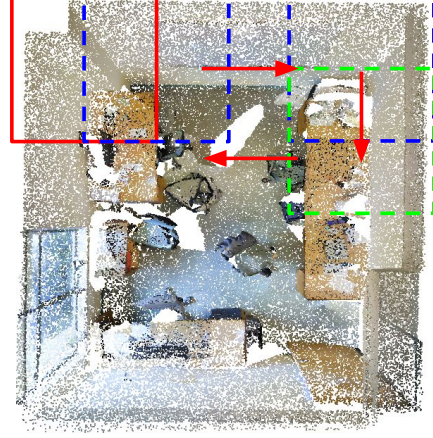


Figure 9: Dividing scene into blocks with overlap (top view).

Algorithm 1: BlockMerging

Input : V, PL
Output: Point instance labels for the whole scene L

```

1 Initialize  $V$  with all elements  $-1$ ;
2  $GroupCount \leftarrow 0$ ;
3 for every block  $i$  do
4   if  $i$  is the 1st block then
5     for every point  $P_j$  in block  $i$  do
6       Define  $k$  where  $P_j$  is located in the  $k$ th
       cell of  $V$ ;
7        $V_k \leftarrow PL_{1j}$ ;
8     end
9   else
10    for every instance  $I_j$  in block  $i$  do
11      Define  $V_{I_j}$  points in  $I_j$  are located in cells
       $V_{I_j}$ ;
12       $V_t \leftarrow$  the cells in  $V_{I_j}$  that do not have
      value  $-1$ ;
13      if the frequency of the mode in  $V_t < 30$ 
14        then
15        |  $V_{I_j} \leftarrow GroupCount$ ;
16        |  $GroupCount \leftarrow GroupCount + 1$ ;
17      else
18      |  $V_{I_j} \leftarrow$  the mode of  $V_t$ ;
19      end
20    end
21  end
22 for every point  $P_j$  in the whole scene do
23   Define  $k$  where  $P_j$  is located in the  $k$ th cell of  $V$ ;
24    $L_j \leftarrow V_k$ ;
25 end

```

| | Mean | wall | floor | chair | table | desk | bed | book shelf | sofa | sink | bath tub | toilet | curtain | counter | door | window | shower curtain | fridge | picture | cabinet |
|-------------|-------------|-------------|-------------|-------------|-------------|-------------|-------------|-------------|-------------|-------------|-------------|-------------|-------------|-------------|------|--------|----------------|-------------|---------|-------------|
| Seg-Cluster | 30.9 | 49.3 | 77.1 | 57.1 | 38.8 | 17.6 | 39.4 | 17.2 | 37.0 | 29.4 | 40.0 | 43.1 | 52.9 | 26.7 | 0.0 | 0.0 | 18.0 | 15.7 | 0.0 | 28.8 |
| SGPN | 35.1 | 46.9 | 79.0 | 63.6 | 40.7 | 22.8 | 43.8 | 22.4 | 36.8 | 35.8 | 46.2 | 60.5 | 61.1 | 26.9 | 0.0 | 0.0 | 21.7 | 24.5 | 0.0 | 34.1 |

Table 8: Instance segmentation results on ScanNet. The metric is AP (%) with IoU threshold 0.5. We observe 0 percent AP on items that appear on the wall (door, window, picture) as they contain very little depth information and are almost all incorrectly semantically labeled as the wall. Future works can explore addressing this problem.

to Algorithm 1) is derived to merge object instances from different blocks.

In Figure 10, we show more qualitative results of instance segmentation with SGPN.

C. More Experiments

C.1. ScanNet

We provide more experimental results on ScanNet [7]. This dataset contains 1513 scanned and reconstructed indoor scenes. We use the official split with 1201 scenes for training and 312 for testing. Following the same *Block-Merging* procedure, each scene is divided into $1.5m \times 1.5m$ blocks and each block is uniformly sampled into 4096 points for training. All points in the block are used at test time. Each point is represented by a 9D vector (XYZ, RGB, and normalized location with respect to the room scene). PointNet++ is used as the baseline. The network architecture is:

$SA(1024, 0.1, [32, 32, 64]),$
 $SA(256, 0.2, [64, 64, 128]),$
 $SA(64, 0.4, [128, 128, 256]),$
 $SA(16, 0.8, [256, 256, 512]),$
 $FP(256, 256),$
 $FP(256, 256),$
 $FP(256, 128),$
 $FP(128, 128, 128, 128).$

And F_{SIM}, F_{CF}, F_{SEM} are 1×1 conv layers with output channels (128, 128, 128) respectively. Table 8 illustrates the quantitative comparison results with Seg-Cluster. The metric is average precision (AP) with IoU threshold 0.5. Figure 11 shows instance segmentation results on ScanNet.

References

- [1] I. Armeni, O. Sener, A. R. Zamir, H. Jiang, I. Brilakis, M. Fischer, and S. Savarese. 3d semantic parsing of large-scale indoor spaces. In *CVPR*, 2016. 2, 5, 6
- [2] V. Badrinarayanan, A. Kendall, and R. Cipolla. Segnet: A deep convolutional encoder-decoder architecture for image segmentation. *TPAMI*, 2017. 4
- [3] L. Bertinetto, J. Valmadre, J. F. Henriques, A. Vedaldi, and P. H. S. Torr. Fully-convolutional siamese networks for object tracking. In *ECCV Workshops*, 2016. 2
- [4] A. X. Chang, T. Funkhouser, L. Guibas, P. Hanrahan, Q. Huang, Z. Li, S. Savarese, M. Savva, S. Song, H. Su, et al. Shapenet: An information-rich 3d model repository. *arXiv preprint arXiv:1512.03012*, 2015. 2, 5, 8
- [5] X. Chen, H. Ma, J. Wan, B. Li, and T. Xia. Multi-view 3d object detection network for autonomous driving. In *CVPR*, 2017. 2
- [6] S. Chopra, R. Hadsell, and Y. LeCun. Learning a similarity metric discriminatively, with application to face verification. In *CVPR*, 2005. 2
- [7] A. Dai, A. X. Chang, M. Savva, M. Halber, T. Funkhouser, and M. Nießner. Scannet: Richly-annotated 3d reconstructions of indoor scenes. In *CVPR*, 2017. 2, 10
- [8] A. Dai, C. R. Qi, and M. Nießner. Shape completion using 3d-encoder-predictor cnns and shape synthesis. In *CVPR*, 2017. 2
- [9] J. Dai, K. He, Y. Li, S. Ren, and J. Sun. Instance-sensitive fully convolutional networks. In *ECCV*, 2016. 2
- [10] J. Dai, K. He, and J. Sun. Instance-aware semantic segmentation via multi-task network cascades. In *CVPR*, 2016. 1, 2
- [11] Z. Deng and L. J. Latecki. Amodal detection of 3d objects: Inferring 3d bounding boxes from 2d ones in rgb-depth images. In *CVPR*, 2017. 1, 2, 5, 7, 8
- [12] A. Frome, Y. Singer, F. Sha, and J. Malik. Learning globally-consistent local distance functions for shape-based image retrieval and classification. In *ICCV*, 2007. 2
- [13] C.-Y. Fu, W. Liu, A. Ranga, A. Tyagi, and A. C. Berg. Dssd: Deconvolutional single shot detector. *arXiv preprint arXiv:1701.06659*, 2017. 2
- [14] R. Girshick. Fast r-cnn. In *ICCV*, 2015. 2
- [15] R. Girshick, J. Donahue, T. Darrell, and J. Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In *CVPR*, 2014. 2
- [16] X. Han, T. Leung, Y. Jia, R. Sukthankar, and A. C. Berg. Matchnet: Unifying feature and metric learning for patch-based matching. In *CVPR*, 2015. 2
- [17] X. Han, Z. Li, H. Huang, E. Kalogerakis, and Y. Yu. High-resolution shape completion using deep neural networks for global structure and local geometry inference. In *ICCV*, 2017. 2
- [18] K. He, G. Gkioxari, P. Dollar, and R. Girshick. Mask r-cnn. In *ICCV*, 2017. 1, 2



Figure 10: Instance segmentation results on S3DIS with SGPN. Different colors represent different instances. The colors of the same object in ground truth and prediction are not necessarily the same.

- [19] D. Kingma and J. Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014. 4
- [20] G. Koch, R. Zemel, and R. Salakhutdinov. Siamese neural networks for one-shot image recognition. In *ICML Deep Learning Workshop*, 2015. 2
- [21] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In *NIPS*, 2012. 6
- [22] L. Leal-Taix, C. Canton-Ferrer, and K. Schindler. Learning by tracking: siamese cnn for robust target association. *CVPR DeepVision Workshops*, 2016. 2
- [23] Y. Li, H. Qi, J. Dai, X. Ji, and Y. Wei. Fully convolutional

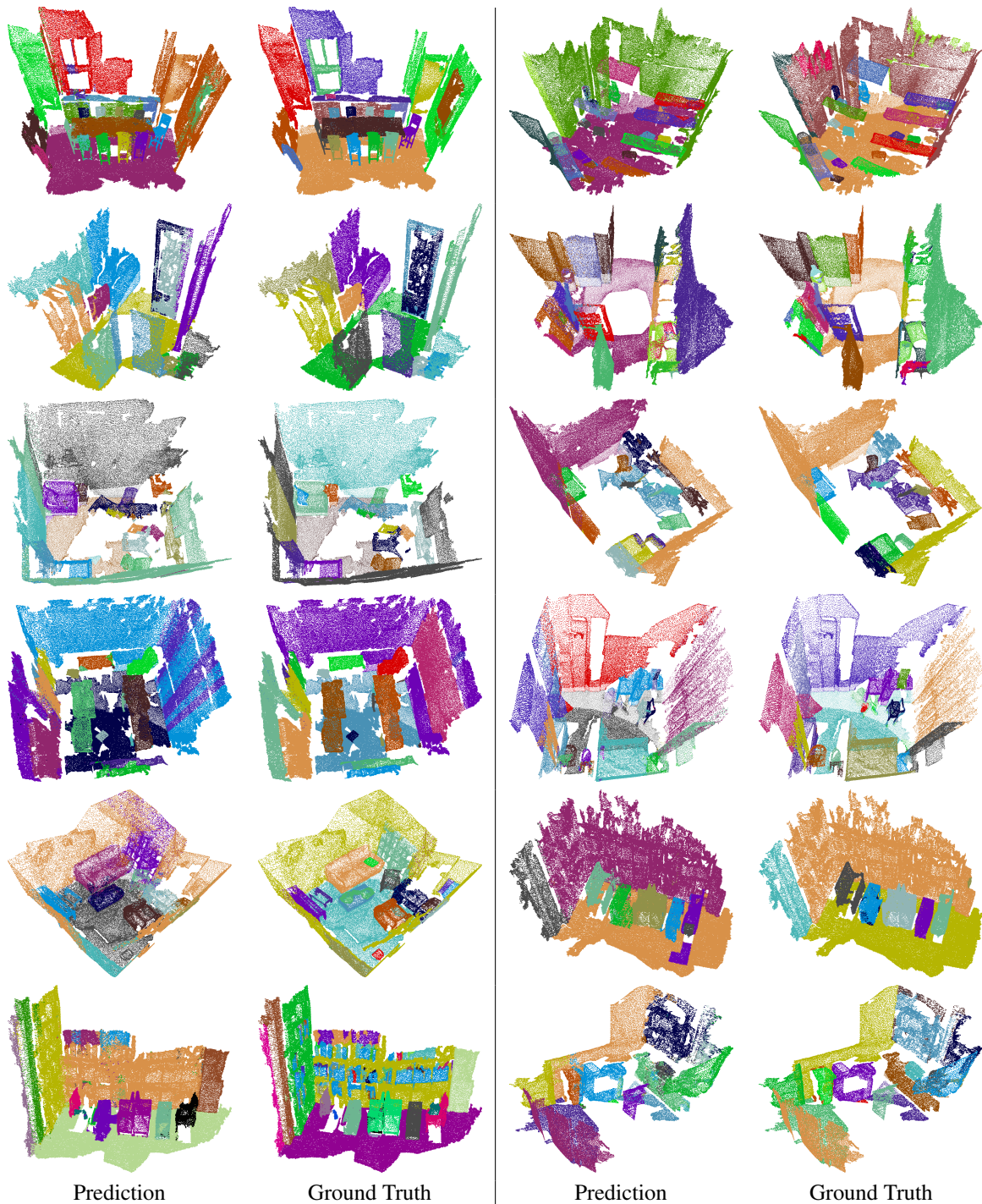


Figure 11: Instance segmentation results on ScanNet with SGPN. Different colors represent different instances. The colors of the same object in ground truth and prediction are not necessarily the same.

- instance-aware semantic segmentation. In *CVPR*, 2017. 1, 2
- [24] T.-Y. Lin, P. Dollar, R. Girshick, K. He, B. Hariharan, and S. Belongie. Feature pyramid networks for object detection. In *CVPR*, 2017. 2
- [25] T.-Y. Lin, P. Goyal, R. Girshick, K. He, and P. Dollar. Focal loss for dense object detection. In *CVPR*, 2017. 2
- [26] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu, and A. C. Berg. SSD: Single shot multibox detector. In *ECCV*, 2016. 2
- [27] D. Maturana and S. Scherer. Voxnet: A 3d convolutional neural network for real-time object recognition. In *IROS*, 2015. 2

- [28] A. Newell and J. Deng. Associative embedding: End-to-end learning for joint detection and grouping. In *NIPS*, 2016. 2, 3
- [29] P. O. Pinheiro, R. Collobert, and P. Dollár. Learning to segment object candidates. In *NIPS*, 2015. 1, 2
- [30] P. O. Pinheiro, T.-Y. Lin, R. Collobert, and P. Dollár. Learning to refine object segments. In *ECCV*, 2016. 2
- [31] C. R. Qi, H. Su, K. Mo, and L. J. Guibas. Pointnet: Deep learning on point sets for 3d classification and segmentation. *CVPR*, 2017. 1, 2, 3, 5, 6, 9
- [32] C. R. Qi, H. Su, M. Nießner, A. Dai, M. Yan, and L. J. Guibas. Volumetric and multi-view cnns for object classification on 3d data. In *CVPR*, 2016. 2
- [33] C. R. Qi, L. Yi, H. Su, and L. J. Guibas. Pointnet++: Deep hierarchical feature learning on point sets in a metric space. In *NIPS*, 2017. 1, 2, 3, 5, 8, 9
- [34] X. Qi, R. Liao, J. Jia, S. Fidler, and R. Urtasun. 3d graph neural networks for rgb-d semantic segmentation. In *CVPR*, 2017. 6
- [35] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi. You only look once: Unified, real-time object detection. In *CVPR*, 2016. 2
- [36] J. Redmon and A. Farhadi. Yolo9000: Better, faster, stronger. In *CVPR*, 2017. 2
- [37] S. Ren, K. He, R. Girshick, and J. Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. In *NIPS*, 2015. 2
- [38] Z. Ren and E. B. Sudderth. Three-dimensional object detection and layout prediction using clouds of oriented gradients. In *CVPR*, 2016. 2
- [39] G. Riegler, A. O. Ulusoy, and A. Geiger. Octnet: Learning deep 3d representations at high resolutions. In *CVPR*, 2017. 2
- [40] N. Silberman, D. Hoiem, P. Kohli, and R. Fergus. Indoor segmentation and support inference from rgb-d images. *ECCV*, 2012. 2, 5
- [41] E. Simo-Serra, E. Trulls, L. Ferraz, I. Kokkinos, P. Fua, and F. Moreno-Noguer. Discriminative learning of deep convolutional feature point descriptors. In *ICCV*, 2015. 2
- [42] S. Song and J. Xiao. Sliding shapes for 3d object detection in depth images. In *ECCV*, 2014. 1, 2, 7
- [43] S. Song and J. Xiao. Deep Sliding Shapes for amodal 3D object detection in RGB-D images. In *CVPR*, 2016. 1, 2, 7, 8
- [44] S. Song, F. Yu, A. Zeng, A. X. Chang, M. Savva, and T. Funkhouser. Semantic scene completion from a single depth image. *arXiv preprint arXiv:1611.08974*, 2016. 2
- [45] M. Tatarchenko, A. Dosovitskiy, and T. Brox. Octree generating networks: Efficient convolutional architectures for high-resolution 3d outputs. In *ICCV*, 2017. 2
- [46] P.-S. Wang, Y. Liu, Y.-X. Guo, C.-Y. Sun, and X. Tong. O-cnn: Octree-based convolutional neural networks for 3d shape analysis. *ACM Transactions on Graphics (TOG)*, 2017. 2
- [47] W. Wang, Q. Huang, S. You, C. Yang, and U. Neumann. Shape inpainting using 3d generative adversarial network and recurrent convolutional networks. In *ICCV*, 2017. 2
- [48] K. Q. Weinberger and L. K. Saul. Distance metric learning for large margin nearest neighbor classification. *Journal of Machine Learning Research*, 2009. 2
- [49] Z. Wu, S. Song, A. Khosla, F. Yu, L. Zhang, X. Tang, and J. Xiao. 3d shapenets: A deep representation for volumetric shapes. In *CVPR*, 2015. 2
- [50] D. Yi, Z. Lei, S. Liao, and S. Z. Li. Deep metric learning for person re-identification. In *ICPR*, 2014. 2
- [51] L. Yi, V. G. Kim, D. Ceylan, I. Shen, M. Yan, H. Su, A. Lu, Q. Huang, A. Sheffer, L. Guibas, et al. A scalable active framework for region annotation in 3d shape collections. *ACM Transactions on Graphics (TOG)*, 2016. 5