Point Cloud Instance Segmentation using Probabilistic Embeddings

Biao Zhang
KAUST
biao.zhang@kaust.edu.sa

Peter Wonka
KAUST
pwonka@gmail.com

Abstract

In this paper we propose a new framework for point cloud instance segmentation. Our framework has two steps: an embedding step and a clustering step. In the embedding step, our main contribution is to propose a probabilistic embedding space for point cloud embedding. Specifically, each point is represented as a tri-variate normal distribution. In the clustering step, we propose a novel loss function, which benefits both the semantic segmentation and the clustering. Our experimental results show important improvements to the SOTA, i.e., 3.1% increased average per-category mAP on the PartNet dataset.

1. Introduction

In this paper we tackle the problem of instance segmentation of point clouds. In instance segmentation we would like to assign two labels to each point in a point cloud. The first label is the class label (*e.g.*, leg, back, seat, ... for a chair data set) and the second label is the instance ID (a unique number, *e.g.*, to distinguish the different legs of a chair). While instance segmentation had many recent successes in the image domain [11, 24, 7, 27, 4, 26], we believe that the problem of instance segmentation for point clouds is not sufficiently explored.

We build our work on the idea of embedding-based instance segmentation, that is very popular in the image and volume domain [7, 27, 4, 26, 21] and has also been successfully applied in the point clouds domain [33, 34]. In this approach typically two steps are employed. In the first step, each point (or pixel) is embedded in a feature space such that points belonging to the same instance should be close and points belonging to different instances should be further apart from each other. In the second step points are grouped using a clustering algorithm, such as mean-shift or greedy clustering.

One important design choice in embedding-based methods is the dimensionality of the feature space. Some methods propose to use a high dimensional feature space [4, 20], while others use a low dimensional features space that has
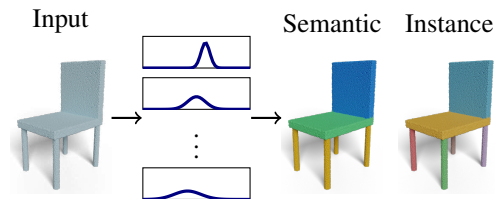

Figure 1: Our method takes a point cloud as input, encodes the points as random variables, and outputs semantic class labels and instance labels.

the same dimensionality as the input data [27, 16, 26], *e.g.*, 2D for images, and 3D for point clouds. Methods with a low dimensional embedding space not only have lower computational complexity, but they also lead to better interpretability, *e.g.*, embeddings are encoded as offset vectors towards instance centers.

Therefore, the main goal of our work is to extend the expressiveness of the embedding space in a way that leads to improved segmentation performance. Our proposed solution is to employ probabilistic embeddings, such that each point in the embedding space is encoded by a distribution. While assessing uncertainty is a popular tool in recent computer vision research [15, 17, 23, 5] and we introduce this idea to the task of instance segmentation. Incorporating uncertainty leads to an important improvement in segmentation performance. For example, on the PartNet [25] fine-grained instance segmentation dataset we can improve the SOTA by 3.1% average per-category mAP.

In the remainder of the paper, we will give more details on the probabilistic embedding algorithm (Sec. 3), explain the embedding step (Sec. 3.1) and the clustering step (Sec. 3.4) in more detail.

Contribution.   Our main contributions are as follows

1. We propose to use probabilistic embeddings for instance segmentation and present a complete framework in the context of point cloud instance segmentation based on probabilistic embeddings.

2. We develop a new loss function for the clustering step

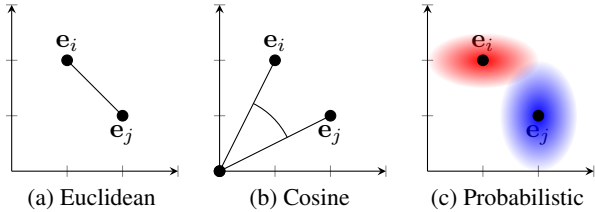(a) Euclidean   (b) Cosine   (c) Probabilistic

Figure 2: **Examples of (dis)similarity measures.**

that is especially suited for high-granularity data sets.

3. We show that the proposed probabilistic embeddings can be incorporated into existing embedding-based methods.

2. Related work

2.1. 2D image instance segmentation

The dominant approaches for image instance segmentation are proposal-based methods [11, 24], which are built upon object detection methods [8, 29]. Typically, they have higher quality, but a slower computation time compared to proposal free methods. The mainstream proposal free approaches are based on metric learning. The basic idea is to learn an embedding space, in which pixels belonging to the same object instance are close to each other and distant to pixels belonging to other object instances [7, 4]. All above works are based on high-dimensional embedding, while more recent works [22, 16, 27, 26] show that 2D spatial embedding is sufficient to achieve the same or even higher performance.

2.2. 3D point cloud instance segmentation

SGPN [33] uses PointNet++ [28] as backbone network and designs a double-hinge loss function to learn a pairwise similarity matrix of points. GSPN [36] produces object proposals with high objectness for point cloud instance segmentation. ASIS [34] is a module capable of making semantic segmentation and instance segmentation take advantage of each other. [25] release a large scale point cloud dataset for part instance segmentation and benchmark their method and SGPN on this dataset. PointGroup [14] and OccuSeg [10] achieved great success in scene datasets by voxelizing point clouds.

2.3. Uncertainty in computer vision

[15] present a unified framework combining model uncertainty with data uncertainty and can estimate uncertainty in classification and regression tasks. We introduce uncertainty estimation to the literature of instance embedding, by modeling points as random variables. Our method is related to recent works in deep generative networks [19, 30]. They use a stochastic encoder to encode a data sample as a set of random variables, while focusing on solving the problem of backpropagation through random variables in deep neural networks. We deal with this problem by using a probabilistic product kernel [13].

3. Method

A training sample is a labeled 3D point cloud. It consists of point coordinates $\{\mathbf{x}_i\}_{i=1}^N$, class labels $\{y_i\}_{i=1}^N$ and instance IDs $\{z_i\}_{i=1}^N$. We want to train a neural network to infer per point class labels and per point instance IDs at the same time.

3.1. Probabilistic spatial embedding

A common approach in the literature of instance segmentation is to learn a function to embed pixels/points into a space where pair-wise similarity can be measured. Usually, this function is a deep neural network f which transforms an unordered point set $\{\mathbf{x}_i\}_{i=1}^N$ to embeddings $\{\mathbf{e}_i\}_{i=1}^N$. Instead of deterministic embeddings used in previous work, here we consider a probabilistic embedding, by modeling $\mathbf{e}_i$ as a random variable, $\mathbf{e}_i \sim p_i(\mathbf{e})$, where p_i is a probability density function. In Section 3.3 we will need to calculate the sum of random variables. In the ideal case, the distribution of a single random variable and the sum of multiple random variables has the same type of distribution that can be described with a few parameters. We choose to work with the tri-variate Gaussian distribution[1] $p_i(\mathbf{e}) = \mathcal{N}(\mathbf{e}; \boldsymbol{\mu}_i, \boldsymbol{\Sigma}_i)$ with mean vector $\boldsymbol{\mu}_i \in \mathbb{R}^3$ and covariance matrix $\boldsymbol{\Sigma}_i \in \mathbb{R}^{3 \times 3}$. For simplicity, let $\boldsymbol{\Sigma}_i$ be a diagonal matrix, $\boldsymbol{\Sigma}_i = \mathrm{diag}(\sigma_i^{(1)2}, \sigma_i^{(2)2}, \sigma_i^{(3)2})$, where $\sigma_i^{(d)2}$ is the square of $\sigma_i^{(d)}$ and $d = 1, 2, 3$.

The network $f(\cdot)$ takes as input a (unordered) point set $\{\mathbf{x}_i\}_{i=1}^N$, and outputs $\{\boldsymbol{\mu}_i, \boldsymbol{\sigma}_i, \mathbf{p}_i\}_{i=1}^N$, $f(\{\mathbf{x}_i\}_{i=1}^N) = \{\boldsymbol{\mu}_i, \boldsymbol{\sigma}_i, \mathbf{p}_i\}_{i=1}^N$, where $\boldsymbol{\sigma}_i = \left[\sigma_i^{(1)}, \sigma_i^{(2)}, \sigma_i^{(3)}\right]^\mathsf{T} \in \mathbb{R}^3$ and $\mathbf{p}_i$ is a probability vector which can be used to infer class label of $\mathbf{x}_i$ and will be explained in Sec. 3.4.

3.2. Similarity measure

In deterministic embeddings, the (dis)similarity between points is usually measured by Euclidean distance $\|\mathbf{e}_i - \mathbf{e}_j\|$, or cosine similarity $\frac{\mathbf{e}_i^\mathsf{T} \mathbf{e}_j}{\|\mathbf{e}_i\|\|\mathbf{e}_j\|}$ (See Figure 2). Since now we are using probabilistic embeddings, a similarity kernel for random variables needs to be selected. Here we describe the Bhattacharyya kernel [13].

Definition. *Let $\mathcal{P}$ be the set of distributions over Ω. The Bhattacharyya kernel on $\mathcal{P}$ is the function $\mathcal{K} : \mathcal{P} \times \mathcal{P} \mapsto \mathbb{R}$ such that, for all $p, q \in \mathcal{P}$, $\mathcal{K}(p, q) = \int_\Omega \sqrt{p(\mathbf{x})}\sqrt{q(\mathbf{x})}\mathrm{d}\mathbf{x}$.*

---

[1] Refer to [22, 16, 27, 26] for a discussion why spatial embedding works.

We choose this kernel as our similarity measure for two reasons, 1) the Bhattacharyya kernel is symmetric, i.e. $\mathcal{K}(p,q) = \mathcal{K}(q,p)$; 2) the Bhattacharyya kernel has values between 0 (no similarity) and 1 (maximal similarity). And $\mathcal{K}(p,q) = 1$ if and only if $p = q$.

Then the similarity $\kappa(\cdot, \cdot)$ between random variables can be represented by the Bhattacharyya kernel of their probability density functions[2],

$$
\begin{aligned}
\kappa(\mathbf{e}_i, \mathbf{e}_j) &= \int \sqrt{\mathcal{N}(\mathbf{e}; \boldsymbol{\mu}_i, \boldsymbol{\Sigma}_i)} \sqrt{\mathcal{N}(\mathbf{e}; \boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j)} d\mathbf{e} \\
&= \beta_{i,j} \exp\left(-\left\| \boldsymbol{\mu}_i - \boldsymbol{\mu}_j \right\|^2_{\boldsymbol{\Sigma}_{i,j}^{-1}} \right),
\end{aligned} \tag{1}
$$

where

$$
\alpha_{i,j}^{(d)} = 4(\sigma_i^{(d)2} + \sigma_j^{(d)2}),
$$

$$
\beta_{i,j} = \left(\prod_{d=1}^{3} \left(\sigma_i^{(d)}/\sigma_j^{(d)} + \sigma_j^{(d)}/\sigma_i^{(d)} \right)/2 \right)^{-\frac{1}{2}},
$$

$$
\boldsymbol{\Sigma}_{i,j} = \mathrm{diag}(\alpha_{i,j}^{(1)}, \alpha_{i,j}^{(2)}, \alpha_{i,j}^{(3)}),
$$

$$
\left\| \boldsymbol{\mu}_i - \boldsymbol{\mu}_j \right\|^2_{\boldsymbol{\Sigma}_{i,j}^{-1}} = (\boldsymbol{\mu}_i - \boldsymbol{\mu}_j)^\intercal \boldsymbol{\Sigma}_{i,j}^{-1} (\boldsymbol{\mu}_i - \boldsymbol{\mu}_j)
$$

$$
= \sum_{d=1}^{3} (\mu_i^{(d)} - \mu_j^{(d)})/\alpha_{i,j}^{(d)}.
$$

- If the uncertainties $\boldsymbol{\sigma}_i$ and $\boldsymbol{\sigma}_j$ have a large difference, $\beta_{i,j}$ will be small, so will be $\kappa(\mathbf{e}_i, \mathbf{e}_j)$. See Fig. 4.

- If the centers $\boldsymbol{\mu}_i$ and $\boldsymbol{\mu}_j$ have a large difference, the exponential term will be small, so will be $\kappa(\mathbf{e}_i, \mathbf{e}_j)$. See Fig. 4.

- The scale term $\beta_{i,j} = 1$ if and only if the uncertainties $\boldsymbol{\sigma}_i$ and $\boldsymbol{\sigma}_j$ are element-wise equal. In this case, $\kappa(\mathbf{e}_i, \mathbf{e}_j)$ becomes an anisotropic Gaussian kernel,

$$
\kappa_{RBF}(\boldsymbol{\mu}_i, \boldsymbol{\mu}_j) = \exp\left(-\left\| \boldsymbol{\mu}_i - \boldsymbol{\mu}_j \right\|^2_{\boldsymbol{\Sigma}_{i,j}^{-1}} \right). \tag{2}
$$

- The exponential term equals 1 if and only if the centers $\boldsymbol{\mu}_i$ and $\boldsymbol{\mu}_j$ are element-wise equal. In this case, $\kappa(\mathbf{e}_i, \mathbf{e}_j)$ becomes $\beta_{i,j}$, i.e., the similarity between uncertainties. This property allows two points that have the same embedding centers to have a low similarity, as long as $\beta_{i,j}$ is small.

Compared to deterministic embedding, $\kappa(\mathbf{e}_i, \mathbf{e}_j) = \exp\left(-\left\| \mathbf{e}_i - \mathbf{e}_j \right\|^2 \right)$ our similarity measure consists not merely of the similarity of spatial distances, but also the similarity of uncertainties.

In the following, we discuss multiple choices of embedding distributions that we will evaluate in Sec 4.2.
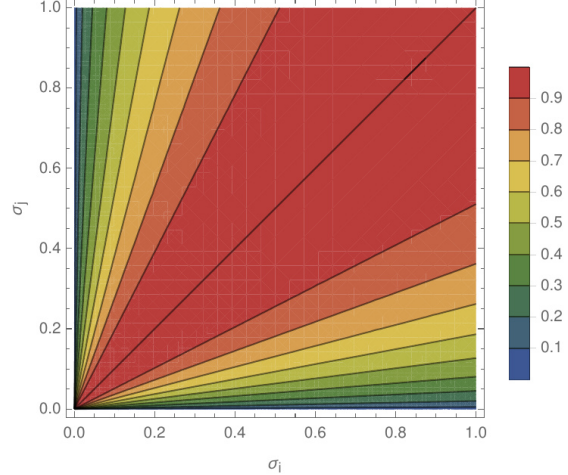
---
[2]Refer to [13] for a derivation.

Figure 3: Contour plot of the 1-D uncertainty similarity $\beta_{i,j} = \left((\sigma_i/\sigma_j + \sigma_j/\sigma_i)/2 \right)^{-1/2}$. The highest value 1 is achieved when $\sigma_i = \sigma_j$. The value goes to 0 when one of the uncertainties is small and the other is large.
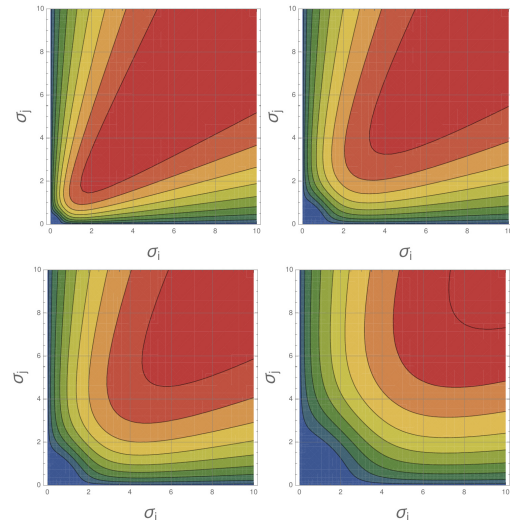

Figure 4: Contour plots of the similarity for different values of $\Delta\mu$ (from left to right, from top to below, $\Delta\mu = 1, 5, 10, 25$) in the case of 1-D probabilistic embedding. The similarity becomes $\beta_{i,j} \exp\left(-\frac{1}{2} \frac{\Delta\mu}{\alpha_{i,j}} \right)$, where $\Delta\mu = (\mu_i - \mu_j)^2$, $\alpha_{i,j} = 4(\sigma_i^2 + \sigma_j^2)$. The legend is the same as in Figure 3.

Homoscedasticity vs. Heteroscedasticity. The embeddings $\{\mathbf{e}_i\}_{i=1}^{N}$ are homoscedastic if they have the same variance $\boldsymbol{\Sigma}$. In this case, for a point cloud $\mathbf{X}$ we learn to predict a single $\boldsymbol{\Sigma}$ instead of point-dependent variances $\{\boldsymbol{\Sigma}_i\}_{i=1}^{N}$. And the similarity kernel becomes $\kappa(\mathbf{e}_i, \mathbf{e}_j) = \exp\left(-\left\| \boldsymbol{\mu}_i - \boldsymbol{\mu}_j \right\|^2_{\boldsymbol{\Sigma}_{i,j}^{-1}} \right)$, which is also the form of the RBF kernel in Eq. (2).

Isotropy vs. Anisotropy. The variance $\mathbf{\Sigma}_i$ is isotropic if its diagonal elements (variances of dimensions) are the same. Then we can write $\mathbf{\Sigma}_i = \sigma_i^2 \mathbf{I}$, where $\mathbf{I}$ is a 3×3 identity matrix. The similarity can be written as

$$\kappa(\mathbf{e}_i, \mathbf{e}_j) = \beta_{i,j} \exp\left(-\frac{\|\boldsymbol{\mu}_i - \boldsymbol{\mu}_j\|^2}{\alpha_{i,j}}\right), \qquad (3)$$

where $\beta_{i,j} = ((\sigma_i/\sigma_j + \sigma_j/\sigma_i)/2)^{-\frac{3}{2}}$ and $\alpha_{i,j} = 4(\sigma_i^2 + \sigma_j^2)$.

3.3. Instance grouping

Let $\{i : z_i = k\}$ be the index set of points having instance ID k. We take an average of these embeddings to get the embedding $\mathbf{c}_k$ of instance k, $\mathbf{c}_k = \frac{1}{|\{i:z_i=k\}|} \sum_{\{i:z_i=k\}} \mathbf{e}_i$. Since the sum of Gaussian random variables is still a Gaussian random variable, we can derive the following:

$$p(\mathbf{c}_k) = \mathcal{N}(\mathbf{c}_k; \hat{\boldsymbol{\mu}}_k, \hat{\mathbf{\Sigma}}_k), \qquad (4a)$$

$$\hat{\boldsymbol{\mu}}_k = \frac{1}{|\{i : z_i = k\}|} \sum_{\{i:z_i=k\}} \boldsymbol{\mu}_i, \qquad (4b)$$

$$\hat{\mathbf{\Sigma}}_k = \frac{1}{|\{i : z_i = k\}|} \sum_{\{i:z_i=k\}} \mathbf{\Sigma}_i. \qquad (4c)$$

Now we can measure the similarity between a point and an instance by using $\kappa(\mathbf{e}_i, \mathbf{c}_k)$.

If $z_i = k$, we want $\kappa(\mathbf{e}_i, \mathbf{c}_k)$ to be close to 1, otherwise 0. We can optimize a binary cross entropy loss function,

$$\mathcal{L}_{InsCE} = \frac{1}{NK} \sum_{k=0}^{K-1} \sum_{i=1}^{N} \begin{cases} -\ln \kappa(\mathbf{e}_i, \mathbf{c}_k), & \text{if } z_i = k, \\ -\ln(1 - \kappa(\mathbf{e}_i, \mathbf{c}_k)), & \text{otherwise.} \end{cases} \qquad (5)$$

However, in practice, this suffers from a serious foreground-background imbalance problem. To remedy this drawback we propose to use the combined log-Dice loss function [35] instead:

$$\mathcal{L}_{Ins} = \mathcal{L}_{InsCE} - \ln \frac{2 \sum_{k=0}^{K-1} \sum_{i=1}^{N} \kappa(\mathbf{e}_i, \mathbf{c}_k) \mathbb{1}_{z_i=k}}{\sum_{k=0}^{K-1} \sum_{i=1}^{N} (\kappa(\mathbf{e}_i, \mathbf{c}_k) + \mathbb{1}_{z_i=k})}, \qquad (6)$$

where $\mathbb{1}_{z_i=k}$ is an indicator function which equals 1 when $z_i = k$, 0 otherwise.

Entropy Regularization. As we can see in Figure 4, when $\sigma_i^{(l)}$ and $\sigma_j^{(l)}$ goes to infinity while keeping $\sigma_i^{(l)} = \sigma_j^{(l)}$, $\beta_{i,j} = 1$ and the similarity equals to 1 no matter what the value $\boldsymbol{\mu}_i - \boldsymbol{\mu}_j$ is. Formally speaking,

$$\lim_{\sigma_i^{(l)} \to \infty, \sigma_j^{(l)} \to \infty, \sigma_i^{(l)} = \sigma_j^{(l)}, l=1,2,3} \kappa(\mathbf{e}_i, \mathbf{e}_j) = 1. \qquad (7)$$
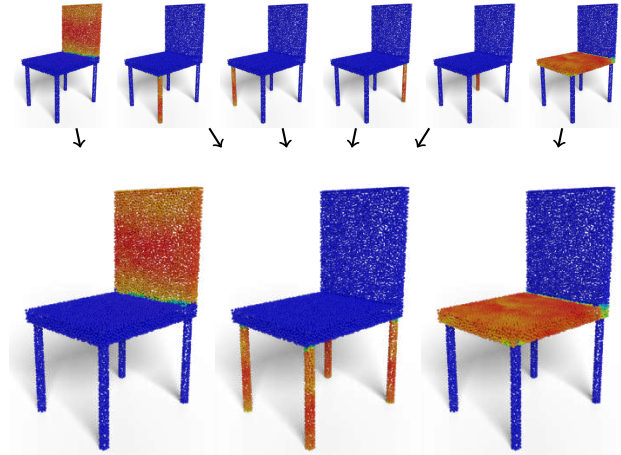

Figure 5: **Top row**: similarity maps for each part instance $\tilde{\mathbf{Q}}[:, k]$. **Bottom row**: probability map $\mathbf{Q}[:, l]$. The arrows show that information of instances of the same class is aggregated in Eq. 11. Red means high similarity, while blue means low.

Consequently, the similarity degenerates to constant 1 for every pair of embeddings. To address this issue, we propose an entropy regularizer,

$$\mathcal{L}_{Reg} = \sum_{i=1}^{N} \mathbb{H}(\mathbf{e}_i) = \sum_{i=1}^{N} \left[\frac{3}{2} \ln(2\pi e) + \frac{1}{2} \ln\left(\sigma_i^{(1)} \sigma_i^{(2)} \sigma_i^{(3)}\right)^2 \right], \qquad (8)$$

where $\mathbb{H}(\mathbf{e}_i)$ is the entropy of multivariate Gaussian variable $\mathbf{e}_i$. This regularizer is not only able to prevent the similarity degeneration by minimizing the variances along all dimensions, but can also penalize large uncertainties, thus increasing the confidence of the network output as in [9] and [32].

3.4. Semantic classification

Neven *et al.* [26] introduces a way to use score maps to find cluster centers. Our main novelty is the new loss function, so our description focuses on this part. We still describe the greedy clustering steps from [26] for completeness. In Section 4.2, we compare our new *center-aware loss* to the previously used MSE loss in [26].

After defining the similarity measure, we can easily find out all points similar to an instance center. However, during the inference phase, we don't have the information of ground-truth instance IDs, thus, it is impossible to use Eqs. 4b and 4c to get instance centers. Therefore, along with distribution parameters $\{\boldsymbol{\mu}_i\}_{i=1}^{N}$ and $\{\boldsymbol{\sigma}_i\}_{i=1}^{N}$, we also predict a score map $\{\mathbf{p}_i\}_{i=1}^{N}$, where $\mathbf{p}_i \in \mathbb{R}^L$ and its l-th entry $\mathbf{p}_i^{(l)}$ indicates the probability of $\mathbf{x}_i$ being an instance center with class label l. Thus we want $\mathbf{P} \in \mathbb{R}^{N \times L}$, the matrix form of $\{\mathbf{p}_i\}_{i=1}^{N}$, to satisfy two conditions:

1. $\mathbf{P}[i,:]$ is a probability vector and can be used to infer class label y_i of point $\mathbf{x}_i$, i.e., $y_i = \arg\max_{l=1}^{L} \mathbf{P}[i,l]$.

2. For foreground class labels $l \in \{1, 2, \ldots, L\}$, $\mathbf{P}[:,l]$ is a score map of being an instance center with class label l.

The first condition is easy to satisfy with the cross entropy loss. Assuming $\mathbf{P}[i,:]$ is the output of a softmax function, we can minimize,

$$\mathcal{L}_{ClsCE} = \frac{1}{N}\sum_{i=1}^{N}\left(\frac{1}{L}\sum_{l=1}^{L} -\mathbb{1}_{y_i=l}\log\mathbf{P}[i,l]\right), \quad (9)$$

where $\mathbb{1}_{y_i=l}$ is an indicator function which equals 1 when $y_i = l$, 0 otherwise.

For the second condition, we take into account $\kappa(\mathbf{e}_i, \mathbf{c}_k)$, which is the similarity between $\mathbf{x}_i$ and an instance k. Consider $\tilde{\mathbf{Q}} \in \mathbb{R}^{N \times K}$,

$$\tilde{\mathbf{Q}}[i,k] = \kappa(\mathbf{e}_i, \mathbf{c}_k)\mathbb{1}_{z_i=k} = \begin{cases} \kappa(\mathbf{e}_i, \mathbf{c}_k) & z_i = k, \\ 0 & \text{otherwise}, \end{cases} \quad (10)$$

where each entry $\tilde{\mathbf{Q}}[i,k]$ can be interpreted as the probability of $\mathbf{x}_i$ being the center of instance k. Upon this we calculate $\mathbf{Q} \in \mathbb{R}^{N \times L}$, where $\mathbf{Q}[i,l]$ gives the probability of $\mathbf{x}_i$ being an instance center with class label l,

$$\mathbf{Q}[i,l] = \max_{\{k:y(k)=l\}} \tilde{\mathbf{Q}}[i,k], \quad (11)$$

where $y(k)$ is the class label of instance k, due to the fact that $\{\mathbf{x}_i : z_i = k\}$ must have the same class label. (See an illustration in Figure 5.) After that, we want both $\mathbf{P}[:,l]$ and $\mathbf{Q}[:,l]$ to achieve local maxima at the same points for all $l \in \{1, 2, \ldots, L\}$. When we are doing inference, these local maxima are chosen as instance centers. Therefore, the first condition can be weakened, and only points which are close to instance centers should be classified correctly.

We design a new loss function to satisfy the two conditions at the same time,

$$\mathcal{L}_{Score} = \frac{1}{NL}\sum_{i=1}^{N}\sum_{l=1}^{L} -\mathbf{Q}[i,l]\log\mathbf{P}[i,l]. \quad (12)$$

Here $\mathbf{Q}$ is fixed as a target when training. We can view $\mathcal{L}$ in two ways,

1. First, we switch the order of summation in Eq. (12),

$$\mathcal{L}_{Score} = \frac{1}{L}\sum_{l=1}^{L}\left(\frac{1}{N}\sum_{i=1}^{N} -\mathbf{Q}[i,l]\log\mathbf{P}[i,l]\right). \quad (13)$$

The value of this quantity $-\mathbf{Q}[i,l]\log\mathbf{P}[i,l]$ is high when weight term $\mathbf{Q}[i,l]$ is high, and if we minimize it,

we are forcing $-\log\mathbf{P}[i,l]$ to be small. Consequently, $\mathbf{P}[i,l]$ would be large. This guarantees local maxima of $\mathbf{Q}[:,l]$ are also local maxima of $\mathbf{P}[:,l]$. And minimizing this loss term is equivalent to minimize the KL-divergence between (unnormalized probability) $\mathbf{Q}[:,l]$ and (unnormalized probability) $\mathbf{P}[:,l]$,

$$\mathbb{KL}(\mathbf{Q}[:,l]|\mathbf{P}[:,l]) = \frac{1}{N}\sum_{i=1}^{N}\mathbf{Q}[i,l]\log\frac{\mathbf{Q}[i,l]}{\mathbf{P}[i,l]}. \quad (14)$$

2. Second, we look at the inner summation of Eq. (12),

$$\mathcal{L}_{Score} = \frac{1}{N}\sum_{i=1}^{N}\left(\frac{1}{L}\sum_{l=1}^{L} -\mathbf{Q}[i,l]\log\mathbf{P}[i,l]\right). \quad (15)$$

The inner summation inside the round bracket is the cross entropy between $\mathbf{Q}[i,:]$ and $\mathbf{P}[i,:]$. And it is equivalent to replacing the one-hot vector in Equation 9 with $\mathbf{Q}[i,:]$. Also, it is the form of label smoothing, a commonly used training trick in image classification [31, 12]. The closer $\mathbf{Q}[i,:]$ is to a one-hot vector, the more confidence we give to the classification loss of point $\mathbf{x}_i$. By definition of $\mathbf{Q}[i,l]$, it can be easily seen that the resulting classifier only classifies near-centers points correctly. Thus we call our new loss function the *center-aware loss*.

The inference process is done with a greedy approach [26]. From foreground score maps $\{\mathbf{P}[:,1], \mathbf{P}[:,2], \ldots, \mathbf{P}[:,L]\}$, we sample a point $\mathbf{x}_{i_0}$ with highest score $\mathbf{P}[i_0, l_0]$, where i_0 is the point index and l_0 is its class label,

$$(i_0, l_0) = \underset{i \in \{1,2,\ldots,N\}, \, l \in \{1,2,\ldots,L\}}{\arg\max} \mathbf{P}[i,l]. \quad (16)$$

The point $\mathbf{x}_{i_0}$ is an anchor and we want to find all similar points. Specifically we find all points $\mathbf{x}_i$ with $\kappa(\mathbf{e}_i, \mathbf{e}_{i_0}) \geq \tau$. As a result, the instance ID of $\mathbf{x}_i$ is 0. After that, all points satisfying the inequality are all masked out. Similarly, we sample $\mathbf{x}_{i_1}$ and mask out points with instance ID 1, sample $\mathbf{x}_{i_2}$ and mask out points with instance ID 2, and so on. We stop this loop if there is no point left. We use the validation set to fit hyperparameter τ, which is 0.35 in our experiments.

3.5. Implementation

For a fair comparison to our main competitor PartNet [25] we keep as much of their structure as possible (Note that PartNet is the name of a dataset as well as an instance segmentation method). We also use PointNet++ [28] as the feature extraction backbone, with the same parameters as [25]. We use 3 output heads for centers, uncertainties, and scores as in $f(\{\mathbf{x}_i\}_{i=1}^{N}) = \{\boldsymbol{\mu}_i, \boldsymbol{\sigma}_i, \mathbf{p}_i\}_{i=1}^{N}$. We list the activation functions for output heads in Table 2.

		Avg	Bag	Bed	Bottle	Bowl	Chair	Clock	Dish	Disp	Door	Ear	Faucet	Hat	Key	Knife	Lamp	Laptop	Micro	Mug	Fridge	Scis	Stora	Table	Trash	Vase
SGPN	1	55.7	38.8	29.8	61.9	56.9	72.4	20.3	72.2	89.3	49.0	57.8	63.2	68.7	20.0	63.2	32.7	**100.0**	50.6	82.2	50.6	71.7	32.9	49.2	56.8	46.6
	2	29.7	-	15.4	-	-	25.4	-	58.1	-	25.4	-	-	-	-	-	21.7	**-**	49.4	-	22.1	-	30.5	18.9	-	-
	3	29.5	-	11.8	45.1	-	19.4	18.2	38.3	78.8	15.4	35.9	37.8	-	-	38.3	14.4	**-**	32.7	-	18.2	-	21.5	14.6	24.9	36.5
	Avg	46.8	38.8	19.0	53.5	56.9	39.1	19.3	56.2	84.1	29.9	46.9	50.5	68.7	20.0	50.8	22.9	100.0	44.2	82.2	30.3	71.7	28.3	27.6	40.9	41.6
PartNet	1	62.6	**64.7**	48.4	**63.6**	59.7	74.4	**42.8**	76.3	93.3	52.9	57.7	69.6	70.9	43.9	58.4	37.2	**100.0**	50.0	86.0	50.0	80.9	45.2	**54.2**	**71.7**	**49.8**
	2	37.4	-	23.0	-	-	35.5	-	62.8	-	**39.7**	-	-	-	-	-	26.9	**-**	47.8	-	35.2	-	35.0	31.0	-	-
	3	36.6	-	15.0	48.6	-	29.0	**32.3**	**53.3**	80.1	17.2	39.4	44.7	-	-	**45.8**	18.7	**-**	34.8	-	26.5	-	27.5	23.9	33.7	**52.0**
	Avg	54.4	**64.7**	28.8	56.1	59.7	46.3	**37.6**	**64.1**	86.7	36.6	48.6	57.2	70.9	43.9	52.1	27.6	100.0	44.2	86.0	37.2	80.9	35.9	**36.4**	52.7	**50.9**
Ours	1	**65.1**	64.6	**51.4**	63.1	**72.0**	**77.1**	41.1	**76.9**	**95.3**	**61.2**	**66.5**	**73.1**	**71.8**	**48.6**	**76.5**	37.1	100.0	**50.5**	**90.9**	**50.5**	**88.6**	**47.3**	40.3	69.0	48.7
	2	**40.4**	-	**31.0**	-	-	**38.6**	-	**64.2**	-	36.9	-	-	-	-	-	**31.0**	**-**	**51.2**	-	**37.3**	-	**42.0**	**31.5**	-	-
	3	**39.8**	-	**26.2**	50.7	-	**34.7**	30.2	50.0	**82.0**	25.7	**43.2**	**55.6**	-	-	44.4	**20.3**	**-**	**37.0**	-	**31.1**	-	**34.2**	25.5	37.7	47.6
	Avg	**57.5**	64.6	**36.2**	56.9	**72.0**	50.1	35.6	63.7	**88.7**	41.3	54.9	**64.4**	**71.8**	**48.6**	**60.5**	29.5	100.0	**46.2**	**90.9**	39.6	**88.6**	**41.2**	32.4	**53.4**	48.1

Table 1: **Instance segmentation results on PartNet (part-category mAP%, IoU threshold 0.5, fine(3), middle(2), and coarse(1)-grained)**.

	Output	Activation
Centers	$\mathbf{o}_i \in \mathbb{R}^3$	$\boldsymbol{\mu}_i = \mathbf{x}_i + \tanh \mathbf{o}_i$
Uncertainties	$\tilde{\boldsymbol{\sigma}}_i \in \mathbb{R}^3$	$\boldsymbol{\sigma}_i = \exp \tilde{\boldsymbol{\sigma}}_i$
Scores	$\tilde{\mathbf{p}}_i \in \mathbb{R}^3$	$\mathbf{p}_i = \text{softmax}(\tilde{\mathbf{p}}_i)$

Table 2: **Activations for different output branches**

The final objective function is

$$\mathcal{L} = \mathcal{L}_{Ins} + \mathcal{L}_{Score} + 0.001 \cdot \mathcal{L}_{Reg} \qquad (17)$$

We use random jittering, translation (between -0.01 and 0.01) and rotation (between $-15°$ and $15°$ for each axis) as data augmentation, and use the Adam [18] optimizer. We use a batch-size of 16 and an initial learning rate of 0.001 for 500 epochs with a decay factor of 0.5 at epoch 50 and epoch 150.

4. Results

PartNet [25] provides coarse-, middle- and fine-grained part instance-level annotations for 3D point clouds from ShapeNet [1]. It contains 24 object categories, but the number of training samples varies greatly from 92 to 5707 for different categories. In contrast to indoor scene point cloud datasets (*e.g.*, ScanNet by [3]), instances (object parts) of PartNet require more context to be classified and are connected. Many visually alike parts have different semantic labels, *e.g.*, ping-pong table's legs and pool table's legs in the category of table. Also, instance masks should have no overlaps. All these make it a very challenging dataset for instance segmentation.

4.1. Quantitative and qualitative results

PartNet. We report per-category mean Average Precision (mAP) scores for the PartNet dataset in Table 1. The IoU threshold is 0.5. We compare our probabilistic embedding algorithm to PartNet [25] and SGPN [33]. The results are averaged over three levels of granularity (fine(3), middle(2), and coarse(1)).

On the complete dataset, our method outperforms the best competitor PartNet by 3.1% average per-category mAP. We can observe that our method has a slightly bigger advantage in fine-grained instance segmentation compared to coarse-grained instance segmentation (3.2% vs. 2.5%). We can also observe consistent improvements in categories with little as well as many training samples. While we beat SOTA in all categories with many training samples (Chair, Table, StorageFurniture, and Lamp), PartNet has better results in some of the categories with fewer training samples.

We also show visualization examples in Figure 6. Compared to PartNet [25], our method shows great improvement especially when there are many instances in a point cloud.

ScanNet. As baseline method we chose a network based on performance and availability of code. Since the best methods, such as OccuSeg, do not release code for ScanNet, we decided to build our own baseline using MinkowskiNet [2] as feature backbone. MinkowskiNet is a sparse tensor network that achieved great results on indoor semantic scene segmentation. In order to adapt the network to instance segmentation, we re-implemented the learnable margin method proposed by [26]. The learnable margin method does well on common image instance segmentation datasets and is well-balanced both in speed and accuracy. This com-
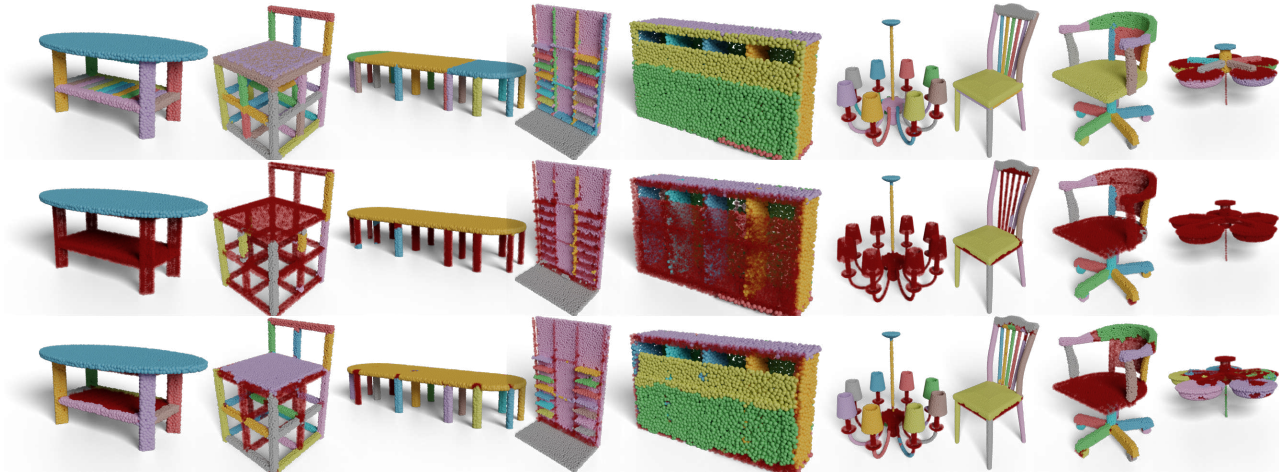
Figure 6: **Top row**: ground-truth. **Middle row**: PartNet. **Bottom row**: Probabilistic Embedding. We show true positives (IoU threshold 0.5) with the same color as ground truth. False detections are shown in *transparent red.*

bination of two recent papers gives a strong baseline, but not state-of-the-art results in the metrics. We compare to this baseline, also using MinkowskiNet as feature backbone to make the results directly comparable.

We report the average precision (AP) in Table 3 and compare our method with other leading results on Scan-Net. Although we do not have the overall state-of-the-art results, the improvement over the baseline [26] verifies the impact of probabilistic embedding and demonstrates that our method can be integrated with any embedding-based method and any backbone network. We can improve the validation mAP by 4.9%. We would also like to note that the main point of the paper is to showcase the benefit of the probabilisitc embedding method. We did not have the re-sources to fine-tune our method for the ScanNet dataset, but nevertheless our results are comparable with the state of the art and in some categories beating state of the art already. We therefore argue that this result underlines the signifi-cance of the proposed embedding method as it is likely that future state of the art methods will also be able to benefit from it.

4.2. Ablation study and analysis

We conduct the ablation study on all categories of Part-Net [25], but we only list detailed values for the four largest categories in Table 4.

Effect of probabilistic embedding. We compare four different versions of probabilistic embedding. The Gaus-sian distribution used in the model can either be isotropic or anisotropic, homoscedastic or heteroscedastic. Thus we have isotropic homoscedastic, anisotropic homoscedastic, isotropic heteroscedastic, and anisotropic heteroscedastic.

The isotropic homoscedastic probabilistic embedding,

	validation			test		
	mAP	AP50	AP25	mAP	AP50	AP25
MTML [21]	20.3	40.2	55.4	28.2	54.9	73.1
3D-MPA [6]	35.3	59.1	72.4	35.5	61.1	73.7
PointGroup [14]	34.8	56.9	71.3	40.7	63.6	**77.8**
OccuSeg [10]	**44.2**	**60.7**	71.9	**44.3**	63.4	73.9
Learnable margin[†][26]	28.1	50.1	70.1	-	-	-
Proposed[†]	33.0	57.1	**73.8**	39.6	**64.5**	77.6

[†] Implemeneted with MinkowskiNet

Table 3: **Results on ScanNet.** We list the results on both validation and hidden test sets of ScanNet. Note that due to the unique submission policy of ScanNet, we are unable to provide the results of learnable margin on the test set. On validation set, we improve mAP by 4.9%.
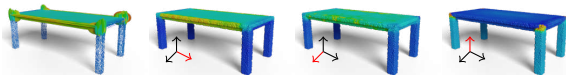

Figure 7: **Learned Uncertainties**. Top left: uncertain-ties are represented as ellipsoids, where directional scaling shows the value of uncertainties along 3 axes. The other 3 subfigures: uncertainties along 3 axes. We represent large values with red colors and smaller values with blue colors.

learns to predict a single scalar representing the uncertainty of a point cloud. We do not see improvements over its deter-minisitc counterpart, but there is a large gap between them in large categories which have much more part instances and classes than others.

Similar cases happen in anisotropic homoscedastic and isotropic homoscedastic embedding. The former learns a 3D uncertainty vector for a single point cloud, while the

Ablation	Model	Center	ExtDim	Prob	Aniso	Hetero	AllAvg	Δ	Large	Δ	Others	Δ	Chair	Δ	Lamp	Δ	Stora	Δ	Table	Δ
Loss				✓	✓	✓	54.3	-0.7	16.2	-7.3	**43.1**	3.1	19.0	-8.4	8.8	-9.6	29.7	6.2	7.1	-17.4
Deterministic	Reference	✓					55.0	0.0	23.4	0.0	40.0	0.0	27.4	0.0	18.5	0.0	23.5	0.0	24.4	0.0
		✓	✓				**56.2**	1.2	27.2	3.7	41.2	1.2	32.8	5.4	**20.2**	1.7	29.8	6.4	**25.8**	1.4
Probabilistic		✓		✓			54.7	-0.3	26.4	3.0	39.9	-0.1	33.7	6.3	17.8	-0.7	30.0	6.5	24.3	-0.2
		✓		✓		✓	53.1	-1.9	**27.4**	3.9	38.1	-1.8	33.6	6.2	20.0	1.5	30.9	7.5	25.0	0.6
		✓		✓	✓		55.6	0.6	27.3	3.9	41.0	1.1	**33.9**	6.5	19.5	1.0	**31.1**	7.6	24.8	0.3
	Full	✓		✓	✓	✓	**57.5**	2.5	**28.7**	5.2	**43.0**	3.0	**34.7**	7.3	**20.3**	1.8	**34.2**	10.7	**25.5**	1.1

Table 4: **Ablation study**. **Center**, **ExtDim**, **Prob** refer to our proposed *center-aware loss for the clustering step, the 6D deterministic embedding, and our proposed probabilistic embedding.* ***Aniso*** and ***Hetero*** *refer to the choice of Gaussian: anisotropic and heteroscedastic.* ***AllAvg*** *means taking all levels of granularity and categories into consideration.* ***Large*** *means fine-grained level of four largest categories.* ***Others*** *means fine-grained level of all the other categories. Here we also list the results on four largest categories of fine-grained level. The top two results are marked bold.*
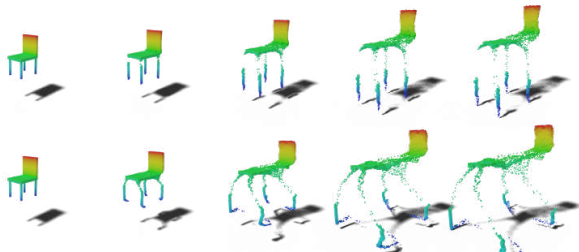

Figure 8: **Comparison of embeddings**. **Top row**: Deterministic embedding. **Bottom row**: Probabilistic embedding. **Left to right**: we show a gradual shape transformation between the original point cloud and the embedded point cloud.

latter learns point-dependent uncertainty scalars. They all show significant improvements over determinisitc embedding on fine-grained categories.

Finally, our full model uses anisotropic heteroscedastic probabilistic embedding, which outputs not only point-dependent but also axis-dependent uncertainties. See Figure 7 for an illustration of learned uncertainties. The points at boundary regions have significantly larger uncertainties compared to others. In summary, the full model achieves the best results among all variations.

Effect of spatial embedding. Since our full model outputs a 3D center vector and 3D uncertainty vector, in a way, we can regard it as a 6D embedding method (with a totally different similarity kernel). One may wonder: how does it compare with the performance of 6D deterministic embedding? The results in Table 4 show, increasing the dimension of deterministic embedding from 3 to 6 shows some improvement, but less than using probabilistic embedding. Thus the performance of our method, cannot be achieved

by simply increasing the dimension of deterministic embedding, which also shows the superiority of the probabilistic embedding. We illustrate the differences between deterministic and probabilistic embedding in 3D in Figure 8. We can observe, that probabilistic embedding introduces much stronger deformations of the geometry.

Effect of center-aware loss. We examine the effect of the center-aware loss in the clustering step. We use the same setup as in our full model except changing the center-aware loss to MSE loss [26]. In Table 4, we can see that our proposed loss function is especially stable on large fine-grained datasets (5.2% vs -7.3%).

5. Conclusion

We build on embedding-based instance segmentation to present a framework of probabilistic embedding and a new loss function for the clustering step. We evaluate our framework on a large scale point cloud dataset, PartNet, and achieve state-of-the-art performance. Moreover, the qualitative results show the new framework is robust to point clouds with many instances. Additionally, it is able to estimate uncertainties while increasing the accuracy of instance segmentation. In future work, we hope that the probabilistic embedding can be further applied to other kinds of data representation, *e.g.*, 2D images, 3D volumes, and meshes.

Acknowledgements

References

[1] Angel X Chang, Thomas Funkhouser, Leonidas Guibas, Pat Hanrahan, Qixing Huang, Zimo Li, Silvio Savarese, Manolis Savva, Shuran Song, Hao Su, et al. Shapenet: An information-rich 3d model repository. *arXiv preprint arXiv:1512.03012*, 2015.

[2] Christopher Choy, JunYoung Gwak, and Silvio Savarese. 4d spatio-temporal convnets: Minkowski convolutional neural networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3075–3084, 2019.

[3] Angela Dai, Angel X Chang, Manolis Savva, Maciej Halber, Thomas Funkhouser, and Matthias Nießner. Scannet: Richly-annotated 3d reconstructions of indoor scenes. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5828–5839, 2017.

[4] Bert De Brabandere, Davy Neven, and Luc Van Gool. Semantic instance segmentation with a discriminative loss function. *arXiv preprint arXiv:1708.02551*, 2017.

[5] Garoe Dorta, Sara Vicente, Lourdes Agapito, Neill D. F. Campbell, and Ivor Simpson. Structured uncertainty prediction networks. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2018.

[6] Francis Engelmann, Martin Bokeloh, Alireza Fathi, Bastian Leibe, and Matthias Nießner. 3d-mpa: Multi-proposal aggregation for 3d semantic instance segmentation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 9031–9040, 2020.

[7] Alireza Fathi, Zbigniew Wojna, Vivek Rathod, Peng Wang, Hyun Oh Song, Sergio Guadarrama, and Kevin P Murphy. Semantic instance segmentation via deep metric learning. *arXiv preprint arXiv:1703.10277*, 2017.

[8] Ross Girshick. Fast r-cnn. In *Proceedings of the IEEE international conference on computer vision*, pages 1440–1448, 2015.

[9] Yves Grandvalet and Yoshua Bengio. Semi-supervised learning by entropy minimization. In *Advances in neural information processing systems*, pages 529–536, 2005.

[10] Lei Han, Tian Zheng, Lan Xu, and Lu Fang. Occuseg: Occupancy-aware 3d instance segmentation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 2940–2949, 2020.

[11] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. Mask r-cnn. In *Proceedings of the IEEE international conference on computer vision*, pages 2961–2969, 2017.

[12] Tong He, Zhi Zhang, Hang Zhang, Zhongyue Zhang, Junyuan Xie, and Mu Li. Bag of tricks for image classification with convolutional neural networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 558–567, 2019.

[13] Tony Jebara, Risi Kondor, and Andrew Howard. Probability product kernels. *Journal of Machine Learning Research*, 5(Jul):819–844, 2004.

[14] Li Jiang, Hengshuang Zhao, Shaoshuai Shi, Shu Liu, Chi-Wing Fu, and Jiaya Jia. Pointgroup: Dual-set point grouping for 3d instance segmentation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4867–4876, 2020.

[15] Alex Kendall and Yarin Gal. What uncertainties do we need in bayesian deep learning for computer vision? In *Advances in neural information processing systems*, pages 5574–5584, 2017.

[16] Alex Kendall, Yarin Gal, and Roberto Cipolla. Multi-task learning using uncertainty to weigh losses for scene geometry and semantics. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 7482–7491, 2018.

[17] Salman Khan, Munawar Hayat, Syed Waqas Zamir, Jianbing Shen, and Ling Shao. Striking the right balance with uncertainty. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019.

[18] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[19] Diederik P Kingma and Max Welling. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013.

[20] 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.

[21] Jean Lahoud, Bernard Ghanem, Marc Pollefeys, and Martin R Oswald. 3d instance segmentation via multi-task metric learning. *arXiv preprint arXiv:1906.08650*, 2019.

[22] Xiaodan Liang, Liang Lin, Yunchao Wei, Xiaohui Shen, Jianchao Yang, and Shuicheng Yan. Proposal-free network for instance-level object segmentation. *IEEE transactions on pattern analysis and machine intelligence*, 40(12):2978–2991, 2017.

[23] Chao Liu, Jinwei Gu, Kihwan Kim, Srinivasa G. Narasimhan, and Jan Kautz. Neural rgb(r)d sensing: Depth and uncertainty from a video camera. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019.

[24] Shu Liu, Lu Qi, Haifang Qin, Jianping Shi, and Jiaya Jia. Path aggregation network for instance segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 8759–8768, 2018.

[25] Kaichun Mo, Shilin Zhu, Angel X Chang, Li Yi, Subarna Tripathi, Leonidas J Guibas, and Hao Su. Partnet: A large-scale benchmark for fine-grained and hierarchical part-level 3d object understanding. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 909–918, 2019.

[26] Davy Neven, Bert De Brabandere, Marc Proesmans, and Luc Van Gool. Instance segmentation by jointly optimizing spatial embeddings and clustering bandwidth. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 8837–8845, 2019.

[27] David Novotny, Samuel Albanie, Diane Larlus, and Andrea Vedaldi. Semi-convolutional operators for instance segmentation. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 86–102, 2018.

[28] Charles Ruizhongtai Qi, Li Yi, Hao Su, and Leonidas J Guibas. Pointnet++: Deep hierarchical feature learning on

point sets in a metric space. In *Advances in neural information processing systems*, pages 5099–5108, 2017.

[29] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. In *Advances in neural information processing systems*, pages 91–99, 2015.

[30] Danilo Jimenez Rezende, Shakir Mohamed, and Daan Wierstra. Stochastic backpropagation and approximate inference in deep generative models. *arXiv preprint arXiv:1401.4082*, 2014.

[31] Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jon Shlens, and Zbigniew Wojna. Rethinking the inception architecture for computer vision. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2818–2826, 2016.

[32] Dequan Wang, Evan Shelhamer, Bruno Olshausen, and Trevor Darrell. Dynamic scale inference by entropy minimization. *arXiv preprint arXiv:1908.03182*, 2019.

[33] Weiyue Wang, Ronald Yu, Qiangui Huang, and Ulrich Neumann. Sgpn: Similarity group proposal network for 3d point cloud instance segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2569–2578, 2018.

[34] Xinlong Wang, Shu Liu, Xiaoyong Shen, Chunhua Shen, and Jiaya Jia. Associatively segmenting instances and semantics in point clouds. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4096–4105, 2019.

[35] Ken CL Wong, Mehdi Moradi, Hui Tang, and Tanveer Syeda-Mahmood. 3d segmentation with exponential logarithmic loss for highly unbalanced object sizes. In *International Conference on Medical Image Computing and Computer-Assisted Intervention*, pages 612–619. Springer, 2018.

[36] Li Yi, Wang Zhao, He Wang, Minhyuk Sung, and Leonidas J Guibas. Gspn: Generative shape proposal network for 3d instance segmentation in point cloud. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3947–3956, 2019.