Paper Title: Learning Object Bounding Boxes for 3D Instance Segmentation on Point Clouds

We would like to thank all reviewers for their very insightful comments and address them in the following.

1. Comparison of computation efficiency. Table 1 compares the time consumption of four existing approaches using their released codes on the validation split (312 scenes) of ScanNet(v2) dataset. SGPN, ASIS, GSPN and our 3D-BoNet are implemented by Tensorflow 1.4, 3D-SIS by Pytorch 0.4. All approaches are running on a single Titan X GPU and the pre/post-processing steps on an i7 CPU core with a single thread. Note that 3D-SIS automatically uses CPU for computing when some large scenes are unable to be processed by the single GPU. Overall, our approach is much more computationally efficient than existing methods, even achieving up to $20\times$ faster than ASIS.

Table 1: Time consumption of different approaches on the validation split (312 scenes) of ScanNet(v2) (seconds).

	SGPN	ASIS	GSPN	3D-SIS	**3D-BoNet(Ours)**
	network(GPU): 650 group merging(CPU): 46562 block merging(CPU): 2221	network(GPU): 650 mean shift(CPU): 53886 block merging(CPU): 2221	network(GPU): 500 point sampling(GPU): 2995 neighbour search(CPU): 468	voxelization, projection, network, etc. (GPU+CPU): 38841	network(GPU): 650 *SCN (GPU parallel): 208* block merging(CPU): 2221
total	49433	56757	3963	38841	**2871**

2. Gradient estimation of Hungarian algorithm. There are many ways to estimate the gradient of the bouding box assignemnt. In our implementation we use a very simple approach and finding a better estimator is the scope of future work. Given the predicted bounding box parameters as a stack vector of all the boxes, $\mathbf{B}$, and ground-truth boxes, $\bar{\mathbf{B}}$, we compute the assignment cost matrix, $\mathbf{C}$. This matrix is converted to a permutation matrix, $\mathbf{A}$, using the Hungarian algorithm. Here we focus on the euclidean distance component of the loss, $\mathbf{C}^{ed}$. The derivative of our loss component w.r.t the network parameters, θ, in matrix form is: $\frac{\partial \mathbf{C}^{ed}}{\partial \theta} = -2(\mathbf{A}\mathbf{B}-\bar{\mathbf{B}})\left[\mathbf{A}+\frac{\partial \mathbf{A}}{\partial \mathbf{C}}\frac{\partial \mathbf{C}}{\partial \mathbf{B}}\mathbf{B}\right]^T\frac{\partial \mathbf{B}}{\partial \theta}$ (1) . The components are easily computable except for $\frac{\partial \mathbf{A}}{\partial \mathbf{C}}$ which is the gradient of the permutation w.r.t the assignment cost matrix which is zero nearly everywhere. We found that training the model works when setting this term to zero in our experiments. However, convergence can be sped up using the straight-through-estimator [1], which assumes that the gradient of the rounding is identity (or a small constant), $\frac{\partial \mathbf{A}}{\partial \mathbf{C}} = \mathbb{1}$. This speeds up convergence as it allows both the error in the bounding box alignment (1st term of Eq. (1)) to be backpropagated and the assignment to be reinforced (2nd term of Eq. (1)). This approach has been shown to work well in practice for many problems including for differentiating through permutations for solving combinatorial optimization problems [4] and for training binary neural networks [2]. Other, more complex approaches could also be used in our framework for computing the gradient of the assignment such as [3] which uses a Plackett-Luce distribution over permutations and a reparameterized gradient estimator.

3. One-to-one mapping vs. Many-to-one mapping. The primary advantage of one-to-one mapping between predicted boxes and ground truth is the computation efficiency during testing. Nevertheless, many-to-one mapping may bring higher precision with sacrificing the speed. We agree that it is an interesting direction to integrate a greedy algorithm to solve the one-to-one mapping problem, but it is non-trivial to make it differentiable.

4. Discussion about what has been learnt. Fundamentally, the designed multi-criteria loss functions for 3D bounding box prediction enable the network to learn key vertices to include dense point clusters, thereby inferring an overall boundary for the object. This general idea can indeed be extended to 2D images, as long as we are able to find good metrics to measure the valid object pixels within a predicted box.

5. Clarification of Algorithm 1. w.r.t step 3 of Algorithm 1, the probability $\boldsymbol{p}_{xyz} = \frac{1}{1+\exp(-\boldsymbol{\Delta}_{xyz})}$ is a vector, e.g., (p_x, p_y, p_z), indicating the probability of a point being inside of the box from x-y-z axes. Eventually, the minimum value of (p_x, p_y, p_z) determines the final probability of that point being inside of the box. This approximate probability is indeed biased towards slightly larger boxes, and we agree that a normalized distance is worthwhile for future exploration and may benefit the bounding box prediction.

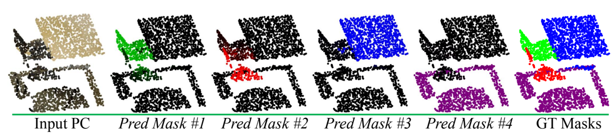

Figure 1: Visualization of predicted instance masks.

6. Visualization of bounding boxes and point masks. Figure 1 visualizes the predicted instance masks, where the black points have ~ 0 probability and the brighter points have ~ 1 probability to be an instance within each predicted mask. Predicted bounding boxes are visualized in the appendix (Section B) of our submission.

References

[1] Bengio, Y., Léonard, N., & Courville, A., *Estimating or propagating gradients through stochastic neurons* ..... arXiv preprint arXiv:1308.3432 (2013).
[2] Yin, Penghang, et al., *Understanding Straight-Through Estimator in Training Activation Quantized Neural Nets*. ICLR (2019).
[3] Grover, Aditya, et al., *Stochastic Optimization of Sorting Networks via Continuous Relaxations*. ICLR (2019).
[4] Emami, P. and Ranka, S., *Learning permutations with sinkhorn policy gradient*. arXiv preprint arXiv:1805.07010 (2019).