Complete & Label: A Domain Adaptation Approach to Semantic Segmentation of LiDAR Point Clouds — Supplementary Material

Li Yi Boqing Gong Thomas Funkhouser
Google Research
{ericyi, bgong, tfunkhouser}@google.com

This document provides a list of supplemental materials to support the main paper.

- **Additional Ablation Studies** - We provide additional ablation studies in a more diverse set of domain adaptation directions in Section A. Specifically, we examine the correlation between scene completion and domain adaptation performance, and we also compare our method with handcrafted sampling aligning baselines.
- **Loss Function for Training SVCN** - We describe the loss function for training the sparse voxel completion network (SVCN) in detail in Section B.
- **Label Transfer to and from the Canonical Domain** - We explain how to propagate the source-domain labels to the dense, complete point clouds in the canonical domain and how to project segmentation results from the canonical domain to the target domain in Section C.
- **Implementation Details** - We provide additional implementation details of our whole pipeline in Section D.

A. Additional Ablation Studies

To evaluate the correlation between the quality of scene completion and the performance of domain adaptation, we provided ablation studies using different variants of our method between Waymo and nuScenes-lidarseg in the main submission. Here we provide additional domain adaptation directions including cases between nuScenes-lidarseg and SemanticKitti and between Waymo and SemanticKitti. The settings are exactly the same as Table 3 in the main submission where we replace SVCN in our method with its variants and report the resulting segmentation results in Table 1. It again shows that better scene completion qualities lead to better domain transfer performances, indicating the importance of high-quality surface completion in our method.

In addition, we also provide comparisons with handcrafted sampling aligning baselines regarding more domain adaptation directions in Table 2 to complement Table 4 in the main submission. The setting is the same as Table 4 in the main submission but we also include domain adaptation results between nuScenes-lidarseg and SemanticKitti as well as those between Waymo and SemanticKitti. Notice the adaptation between nuScenes-lidarseg and SemanticKitti includes 10 categories. Our method outperforms both B1 and B2 as well as the no adaptation baseline by large margins, demonstrating the importance of our learning based approach and the SVCN network.

Table 1. The segmentation mIoU of our approach when using different scene completion methods. N denotes nuScenes-lidarseg dataset, W denotes Waymo dataset and K denotes SemanticKITTI dataset.

Source→ Target	Ours w/o refinement	Ours w/o adv.	Ours-full
N→K	30.1	32.4	**33.7**
K→N	29.6	30.7	**31.6**
W→K	58.8	59.5	**60.4**
K→W	50.3	51.2	**52.0**

Table 2. Comparison with handcrafted sampling aligning baselines. N denotes nuScenes-lidarseg dataset, W denotes Waymo dataset and K denotes SemanticKITTI dataset. No DA denotes no adaptation, B1 analytically downsamples or upsamples LiDAR beams, and B2 linearly interpolates LiDAR points to densify the point cloud.

Src→Tgt	No DA	B1	B2	Ours
N→K	23.5	28.1	26.8	**33.7**
K→N	27.9	30.3	29.7	**31.6**
W→K	55.0	-	56.6	**60.4**
K→W	46.3	-	49.5	**52.0**

B. Loss Function for Training SVCN

Figure 3 in the main text shows the architectures of the structure generation network and the structure refinement network, respectively. Both networks contain 7 resolution levels. For any input-output point clouds pairs, we have the ground truth voxel existence probability (0 or 1) at each of the 7 levels. In particular, we set the groud truth voxel existence probability for a voxel to be 1 if the voxel contains one or more 3D points of the output point cloud.

To train the structure generation network, we use a binary cross entropy loss $\mathcal{L}_{\text{bce}}(c^l_{\text{gen}}, \hat{c}^l_{\text{gen}})$ between the ground truth voxel existence probability c^l_{gen} and the predicted voxel existence probability $\hat{c}^l_{\text{gen}}$, leading to a loss function $\mathcal{L}_{\text{gen}} = \sum_l \mathcal{L}_{\text{bce}}(c^l_{\text{gen}}, \hat{c}^l_{\text{gen}})$, where l indexes the l-th level of the decoder.

To train the structure refinement network, we first pretrain the structure generation network and then fix it but switch to the inference mode where we use the predicted voxel existence probability to prune voxels. A binary cross entropy loss $\mathcal{L}_{\text{refine}} = \mathcal{L}_{\text{bce}}(c^0_{\text{refine}}, \hat{c}^0_{\text{refine}})$ at level 0 between the ground truth voxel existence probability c^0_{refine} and the predicted voxel existence probability $\hat{c}^0_{\text{refine}}$ is used to supervise the network.

Local adversarial loss to model the prior over surfaces. We have a strong prior on the completed scene, namely the recovered voxels should lie on 3D surfaces. Previously, researchers have investigated a lot about how to inject high level prior knowledge to get a better loss landscape and a higher model performance. Among them adversarial learning is a successful attempt [3, 5, 4]. Inspired by this, we introduce local adversarial learning to further inject the 3D surface prior to our SVCN. In addition to the binary cross entropy loss we mentioned before, we add adversarial losses into $\mathcal{L}_{\text{gen}}$ and $\mathcal{L}_{\text{refine}}$, which we will detail below.

We treat SVCN as a generator which could estimate for a given incomplete LiDAR point cloud its corresponding complete counterpart and output a set of voxel existence predictions $\hat{c}^l_{\text{gen}}$ and $\hat{c}^0_{\text{refine}}$ on different resolution levels. We use g^l to represent a set of voxels on resolution level l from a real complete scene where each voxel is associated with an existence probability 1. Following [1], we introduce discriminator networks D^l_{gen} and D^0_{refine} to differentiate $\hat{c}^l$ and g^l, and optimize SVCN together with the discriminators in an adversarial manner.

Instead of using a global discriminator encoding the whole scene which usually contains too much information besides the surface prior and could easily introduce complex noise for learning, we use local discriminators whose receptive field is restricted. This is achieved by using fully-convolutional architectures to retain the spatial information in the discriminator. We use the same fully-convolutional architecture for discriminators on all resolution levels. Specifically, we adopt 4 convolution layers with kernel size 3 and stride 2 followed by a linear layer in the end, where the output channel numbers are $\{32, 64, 64, 128, 1\}$. We do not use batch normalization for the discriminators. We use $D(\hat{c}^l)_i$ to represent the confidence value predicted by D for the generator output $\hat{c}^l$ on each output voxel i. Similarly we use $D(g^l)_j$ to represent the confidence value predicted by D for the real samples g^l on each output voxel j. To train a discriminator on resolution level l, we use binary cross entropy to classify each output voxel into either real or fake and the loss can be written as:

$$\mathcal{L}^l_d = -\sum_i \log(1 - D(\hat{c}^l)_i) - \sum_j \log D(g^l)_j \qquad (1)$$

The adversarial loss for SVCN encourages the generator to generate voxel existence predictions fooling the discriminator and can be written as $\mathcal{L}_{\text{adv}}(\hat{c}^l) = -\sum_i \log D(\hat{c}^l)_i$ on resolution level l. After adding the adversarial loss into $\mathcal{L}_{\text{gen}}$ and $\mathcal{L}_{\text{refine}}$, our final loss functional for SVCN is:

$$\mathcal{L}_{\text{gen}} = \sum_l \mathcal{L}_{\text{bce}}(c^l_{\text{gen}}, \hat{c}^l_{\text{gen}}) + \lambda \mathcal{L}_{\text{adv}}(\hat{c}^l_{\text{gen}}) \qquad (2)$$

$$\mathcal{L}_{\text{refine}} = \mathcal{L}_{\text{bce}}(c^0_{\text{refine}}, \hat{c}^0_{\text{refine}}) + \lambda \mathcal{L}_{\text{adv}}(\hat{c}^0_{\text{refine}}) \qquad (3)$$

Confidence-aware convolution in the discriminators. It is worth noticing that $\hat{c}^l$ contains continuous probability values lying on densely upsampled voxels. On the other hand, g^l lies on voxels from real complete scenes where each voxel is associated with an existence probability 1. Even if SVCN predicts perfect existence scores, it is still very easy for a discriminator to tell its difference from realistic scenes using sparse convolution operations. This is to say, the gradients from discriminator will not necessarily push SVCN toward better predictions, which is against our hope. To cope with this issue, we introduce confidence-aware sparse convolution operation to replace the normal sparse convolution in all the discriminators. Recall that the sparse convolution operation proposed in [2] resembles normal convolution operation but restricts the computation to only active sites. To be specific, assuming a represents an active voxel site, $\mathcal{N}(a)$ represents its neighboring active sites. For each $b \in \mathcal{N}(a)$, $\mathbf{f}_b$ represents the corresponding input voxel features, and W_b represents the corresponding convolution kernel matrix. The output feature $\mathbf{f}'_a$ on site a after sparse convolution is $\mathbf{f}'_a = \sum_{b \in \mathcal{N}(a)} W_b \mathbf{f}_b$. In confidence-aware sparse convolution, we have an additional confidence value c_b associated with each voxel b ranging from 0 to 1 and the output feature after each convolution operation becomes $\mathbf{f}'_a = \sum_{b \in \mathcal{N}(a)} c_b W_b \mathbf{f}_b$. When applying such confidence-aware sparse convolution to $\hat{c}^l$ and g^l, $\hat{c}^l$ and g^l will act as both input features and confidence values. It can be seen that when SVCN generates perfect voxel existence probability in either 0 or 1, the discriminator using confidence-aware sparse convolution will not be able to differentiate it from realistic scenes. Therefore confidence-aware sparse convolution is more suitable for our discriminators. To further reduce the difference between $\hat{c}^l$ and g^l so that trivial solutions can be avoided and learning could start smoothly, we sharpen the predicted existence probability $\hat{c}^l$ from SVCN by replacing the sigmoid activation with a sharpened sigmoid activation $s(x) = \frac{1}{1+e^{-kx}}$ where $k \geq 1$ is a sharpening factor.

C. Label Transfer to and from the Canonical Domain

In order to learn a segmentation network in the canonical domain using source domain labels while being able to infer the target domain point labels, we need two operations Prop$(\cdot)$ and Proj$(\cdot)$. Prop$(\cdot)$ propagates labels $\mathbf{y}_i^s$ in the source domain to the canonical domain and Proj$(\cdot)$ projects predicted labels in the canonical domain back to the target domain, resulting in predicted labels $\hat{\mathbf{y}}_j^t$. In this work, we simply adopt nearest neighbor based Prop$(\cdot)$ and Proj$(\cdot)$ operations. To be specific, we first voxelize input source domain point clouds $\mathbf{x}_i^s$ and conduct majority-voting within each voxel to determine the voxel labels, and then for each voxel we propagate its label to its nearest neighbor voxel in the SVCN output $\psi^s(\mathbf{x}_i^s)$. In the loss function, we mask out voxels without any propagated labels in $\psi^s(\mathbf{x}_i^s)$ during training. At inference time, we voxelize input target domain point clouds $\mathbf{x}_j^t$, fetch the voxel labels from the segmentation network predictions $\phi(\psi^t(\mathbf{x}_j^t))$ through nearest neighbor search, and assign the fetched label to all the points from $\mathbf{x}_j^t$ within each voxel.

D. Implementation Details

The structure generation network, structure refinement network and the semantic segmentation network all contain 7 levels in their encoder-decoder architecture and adopt the same number of convolution filters on different levels. The numbers of filters from level 0 to level 6 of the encoder are $(24, 24), (24, 32), (32, 48), (48, 64), (64, 80), (80, 96), (96, 112)$ where each $(\cdot)$ corresponds to one level. The numbers of filters from level 5 to level 0 of the decoder are $(112, 96), (80, 80), (64, 64), (48, 48), (32, 32), (16, 16)$. In all our experiments, we use a voxel size of $d = 20$cm. To obtain the ground truth voxel existence probability c_{gen}^l on level l for structure generation network training, we voxelize the ground truth complete point cloud with a voxel size of $2^l d$ and the voxel existence probability is set to be 1 for a voxel as long as there is one point falls into it. We use only LiDAR point positions as inputs without considering the color or intensity information. While training the segmentation network, we augment the input point clouds through randomly rotating them around z-axis and randomly flipping them with respect to the x-axis and y-axis. For both SVCN and semantic segmentation network training, we use a batch size of 2. We use Adam optimizer where the momentum is set as 0.9 and 0.99. And we use an initial learning rate of 10^{-3}, which is decayed with a factor of 0.7 after every 200k training steps. The learning rate of the discriminator for adversarial learning is set to be 10^{-4} initially and also decays with a factor of 0.7 after every 200k training steps.

References

[1] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In *Advances in neural information processing systems*, pages 2672–2680, 2014.

[2] Benjamin Graham and Laurens van der Maaten. Submanifold sparse convolutional networks. *arXiv preprint arXiv:1706.01307*, 2017.

[3] Christian Ledig, Lucas Theis, Ferenc Huszár, Jose Caballero, Andrew Cunningham, Alejandro Acosta, Andrew Aitken, Alykhan Tejani, Johannes Totz, Zehan Wang, et al. Photo-realistic single image super-resolution using a generative adversarial network. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4681–4690, 2017.

[4] Ruihui Li, Xianzhi Li, Chi-Wing Fu, Daniel Cohen-Or, and Pheng-Ann Heng. Pu-gan: a point cloud upsampling adversarial network. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 7203–7212, 2019.

[5] Xintao Wang, Ke Yu, Shixiang Wu, Jinjin Gu, Yihao Liu, Chao Dong, Yu Qiao, and Chen Change Loy. Esrgan: Enhanced super-resolution generative adversarial networks. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 0–0, 2018.