Building a Strong Pre-Training Baseline for Universal 3D Large-Scale Perception

Supplementary Material

Contents

A. Implementation Details

Pre-training Pipeline. Overall, we have two phases, the pre-training phase and the fine-tuning phase. To provide clear pre-training phase of CSC, we summarize our framework in pseudo-code style as shown Algorithm 1. Then, we will introduce the implementation details of fine-tuning phase on three downstream perception tasks below.

A.1. Semantic Segmentation and Object Detection

As done in previous works [2, 5], we choose MinkUNet as the 3D backbone and fine-tune the pre-trained MinkUNet to perform semantic segmentation task. For object detection, we use VoxelNet as the 3D backbone. The fine-tuning process on the two downstream tasks, semantic segmentation and object detection, is out of the scope of our contribution. Therefore, we recommend that readers refer to the paper and code in SLidR[5] for details of the implementation.

A.2. Panoptic Segmentation by Cylinder3D

To the best of our knowledge, we are the first to report the results of fine-tuning the pre-trained 3D backbone on panoptic segmentation, where the backbone is obtained from the multi-modality pre-training. Considering that the off-the-shelf pipeline of pre-training and fine-tuning is not available, we draw lessons from the uni-modality point cloud pre-training work [3] and the study of panoptic segmentation method [8, 10], and finally choose the Panoptic-PolarNet [10] with Cylinder3D [11] as our panoptic segmentation network. Specifically, our panoptic segmentation network includes a 3D encoding network implemented by Cylinder3D, a 2D network implemented by UNet, and two heads for the prediction of semantic and instance. In the implementation, we use the LCPS [8] codebase, then remove the image branch to evaluate the 3D pre-training methods.

Algorithm 1 Multi-Modality 3D Pre-Training Pipeline for our Coherent Semantic Cues Framework (CSC)

Input: (i) the point cloud frame $\mathcal{P}$ and (ii) the surrounding images $\mathcal{I}$.
Backbones: (1) the 2D embedding network $\mathbf{\Theta}_{\boldsymbol{I}}$, (2) the 3D embedding network $\mathbf{\Theta}_{\boldsymbol{P}}$, and (3) the 2D mask network $\mathbf{\Theta}_{\boldsymbol{F}}$.
Loss: superpixel-superpoint constrastive loss $\mathcal{L}_{\text{sp}}$ and prototype-based loss $\mathcal{L}_{\text{pro}}$.
Hyper-parameters: (a) the starting epoch of using prototype-based loss λ, (b) the temperature of superpixel-superpoint loss τ_{sp}, and (c) the temperature of prototype-based loss τ_{pro}.
Pipeline:
Obtain superpixels $\mathcal{S}_{\text{2D}}$ and semantic cues $\mathcal{C}_{\text{sem}}$ from $\mathcal{I}$ by $\mathbf{\Theta}_{\boldsymbol{F}}$.
for n in 1:$MaxEpochs$ **do**
  if $n \bmod \lambda == 0$ **then**
    for i in 1:$iterations$ **do**
      Extract 2D/3D features from $\mathcal{I}$/$\mathcal{P}$ by $\mathbf{\Theta}_{\boldsymbol{I}}$/$\mathbf{\Theta}_{\boldsymbol{P}}$.
      According to 2D/3D features, $\mathcal{S}_{\text{2D}}$, and $\mathcal{C}_{\text{sem}}$, to produce the superpixels & superpoints features $\mathcal{F}_{\text{2D}}$ & $\mathcal{F}_{\text{3D}}$.
      Maintain two separate prototype features $\mathcal{P}_{\text{2D}}$ & $\mathcal{P}_{\text{3D}}$ by $\mathcal{F}_{\text{2D}}$, $\mathcal{F}_{\text{3D}}$, and $\mathcal{C}_{\text{sem}}$.
      $\mathcal{P}_{\text{2D}} \xrightarrow{\text{Linear Layers}} \bar{\mathcal{P}}_{\text{2D}}$,
      $\mathcal{P}_{\text{3D}} \xrightarrow{\text{Linear Layers}} \bar{\mathcal{P}}_{\text{3D}}$.
      $\{\bar{\mathcal{P}}_{\text{2D}}, \bar{\mathcal{P}}_{\text{3D}}\} \xrightarrow{\text{Linear Layers}} \mathcal{P}_{\text{mix}}$.
    end for
  end if
  for i in 1:$iterations$ **do**
    Extract 2D/3D features from $\mathcal{I}$/$\mathcal{P}$ by $\mathbf{\Theta}_{\boldsymbol{I}}$/$\mathbf{\Theta}_{\boldsymbol{P}}$.
    According to 2D/3D features, $\mathcal{S}_{\text{2D}}$, and $\mathcal{C}_{\text{sem}}$, to generate the superpixels & superpoints features $\mathcal{F}_{\text{2D}}$ & $\mathcal{F}_{\text{3D}}$.
    $\mathcal{L}_{\text{sp}} = -\sum_{i=0}^{Q} \log \frac{\exp\left(\langle \boldsymbol{F}_{\text{3D}}^{i}, \boldsymbol{F}_{\text{2D}}^{i} \rangle / \tau_{sp}\right)}{\sum_{j=0}^{Q} \exp\left(\langle \boldsymbol{F}_{\text{3D}}^{i}, \boldsymbol{F}_{\text{2D}}^{j} \rangle / \tau_{sp}\right)}$
    if $n < \lambda$ **then**
      $\mathcal{L}_{\text{total}} = \mathcal{L}_{\text{sp}}$.
    else
      $\mathcal{L}_{\text{pro}} = -\log \frac{\exp\left(\langle \boldsymbol{F}_{\text{3D}}, \boldsymbol{P}_{\text{mix}}^{+} \rangle / \tau_{pro}\right)}{\sum_{i=0}^{|\mathcal{C}_{\text{sem}}|} \exp\left(\langle \boldsymbol{F}_{\text{3D}}, \boldsymbol{P}_{\text{mix}}^{i} \rangle / \tau_{pro}\right)}$.
      $\mathcal{L}_{\text{total}} = \mathcal{L}_{\text{sp}} + \mathcal{L}_{\text{pro}}$.
    end if
    Update $\mathbf{\Theta}_{\boldsymbol{P}}$ by $\mathcal{L}_{\text{total}}$.
  end for
end for

B. Complementary Results

B.1. Semantic segmentation result of 1% labels

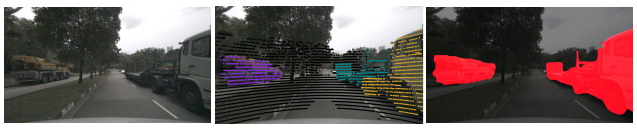
To further detailed analysis in annotation-efficient semantic segmentation, we show the average per-class performance in Tab. 1. We observe that 3D representations learned

Method	barrier	bicycle	bus	car	const. veh.	motorcycle	pedestrian	traffic cone	trailer	truck	driv. surf.	other flat	sidewalk	terrain	manmade	vegetation	**mIoU**
Random Init.	0.0	0.0	8.1	65.0	0.1	6.6	21.0	9.0	9.3	25.8	89.5	14.8	41.7	48.7	72.4	73.3	30.3
Point Con. [6]	0.0	1.0	5.6	67.4	0.0	3.3	31.6	5.6	12.1	30.8	91.7	21.9	48.4	50.8	75.0	74.6	32.5
Depth Con. [7]	0.0	0.6	6.5	64.7	0.2	5.1	29.0	9.5	12.1	29.9	90.3	17.8	44.4	49.5	73.5	74.0	31.7
SLidR [5]	0.0	3.1	15.2	72.0	0.9	18.8	43.2	12.5	14.7	33.3	92.8	29.4	54.0	61.0	80.2	81.9	38.3
ST-SLidR [2]	0.0	2.7	16.0	74.5	3.2	25.4	50.9	20.0	17.7	40.2	92.0	30.7	54.2	61.1	80.5	82.9	40.7
TriCC [4]	0.0	2.6	20.7	73.6	0.3	18.9	49.2	22.0	16.9	33.4	**94.5**	43.1	57.2	62.1	82.3	82.6	41.2
Seal [1]	0.0	**9.4**	32.6	**77.5**	**10.4**	28.0	53.0	25.0	**30.9**	**49.7**	94.0	**33.7**	**60.1**	59.6	**83.9**	83.4	45.8
Ours	0.0	0.0	**58.7**	74.0	0.1	**40.9**	**58.9**	**31.8**	23.7	45.1	92.5	33.0	56.4	**62.4**	81.6	**84.2**	**47.0**

Table 1. Per-class IoU on nuScenes when fine-tunning with 1% labels

by CSC significantly improve performance in the partial classes, such as bus, motorcycle, pedestrian, and traffic cone. Specifically, we observe a remarkable increase of ↑ 26.1% IoU for the bus class, ↑ 12.9% IoU for the motorcycle class, and ↑ 5.9% IoU for the pedestrian class, all of which are encompassed in the prototype semantics of the VFM (see Fig. 2). However, we observe a decrease in the IoUs for the construction vehicle, trailer, and truck categories, which we attribute to the fact that DINOV2 fine-tuned on the ADE20K dataset [9] misclassifies these three classes as the truck class. More specifically, This is because the categories in ADE20K and nuScenes do not match, while we emply DINOv2 fine-tuned on ADE20K during pre-training phase, which is misaligned with the 3D semantic label during fine-tuning phase. As shown in Fig. 1, we assign semantic labels to pixels via 3D semantic labels and 2D-3D relations, and find those pixels categorized as trucks by DINOv2 (c), which could be construction vehicle, trailer, or truck in nuScenes (b).

Nevertheless, our CSC framework still outperforms existing pre-training methods and outperforms the current SOTA method Seal by up to 1.2 mIoU. The above experimental results demonstrate that, compared to the existing 3D pre-training methods, our proposed CSC more effectively transfers 2D knowledge into the 3D network, thereby dramatically reducing the dependency on expensive 3D annotations.



(a) Image (b) 3D semantic label (c) DINOv2 semantic mask

Figure 1. (a) an input image from nuScenes; (b) const. veh., trailer and truck in DINOv2; (c) truck in nuScenes.

B.2. More Image-LIDAR Datasets.

As shown in Tab. 2, we conduct more semantic segmentation experiments on ScribbleKITTI and SemanticSTF, and observe that our CSC outperform SOTA methods on these dataset.

Method	KITTI	ScribbleKITTI (New)		SemanticSTF (New)
	1% (mIoU)	1% (mIoU)	10% (mIoU)	50% (mIoU)
Random Init.	39.5	23.8	47.6	48.0
PPKT	44.0	36.5	51.7	50.9
SLidR	44.6	39.6	50.5	52.0
Seal	46.6	40.6	52.8	53.5
Ours (CSC)	**47.2** (+0.6)	**41.6** (+1.0)	**53.2** (+0.4)	**53.8** (+0.3)

Table 2. CSC vs. previous methods on more datasets.

B.3. More Ablation Study

In this section, we present experimental results on the impact of hyper-parameters (includes λ, τ_{pro} in semantic segmentation and the number of pre-training epochs in panoptic segmentation) and variants of using multi-modality prototype. Note that all experiments are performed on the nuScenes dataset.

Study of λ. As shown in Algorithm 1, we use λ to control the beginning epoch of using our proposed prototype-based loss. Meanwhile, the λ denotes the epoch interval of updating the mixed prototypes. We examine the influences of different λ and tabulate the results in Tab. 3. We observe that when we decrease λ (*e.g.*, $10 \rightarrow 5 \rightarrow 2$), there is almost always exist a performance gain in all settings of different percentage labels. In addition, we list the GPU hour required for the pre-training phase at different λ. Considering the oversized hyper-parameter search space and the time cost of pre-training phase, we take 5 as the default value for λ. Also, this demonstrates the potential of using our CSC framework as the multi-modality pre-training baseline.

λ	GPU Hours	Semantic Seg.			
		1%	5%	10%	25%
2	**29 h**	**47.4**	**57.4**	63.2	**69.4**
5	24 h	47.0	57.0	**63.3**	68.6
10	22 h	46.1	56.6	62.7	68.2

Table 3. Results (mIoU) of modifying the starting epoch λ. The row with the gray background represent our default value for λ.

Study of τ_{pro}. In Tab. 4, we show the impact of various temperature τ_{pro} in our prototype-based loss. The τ_{pro} play a role in scaling the similarity scores between superpoint and prototype embeddings. According to the experimental results, we observe that the pre-trained 3D network achieves the best performance at three different labeling setting (1%, 10%, and 25%), when we set τ_{pro} as 1.0. When $\tau_{pro} = 2.0$, the 3D network achieves the best result at 5% annotation.

τ_{pro}	Semantic Seg.			
	1%	5%	10%	25%
0.5	46.9	56.9	62.3	68.2
1.0	**47.0**	57.0	**63.3**	**68.6**
1.5	45.7	56.7	62.0	67.3
2.0	46.0	**58.4**	62.8	68.1

Table 4. Results (mIoU) of modifying the prototype-based loss's temperature τ_{pro}. The row with the gray background represent our default value for τ_{pro}.

Study of the pre-training epoch. In Tab. 5, we show the results about evaluation metrics of PQ, SQ, and RQ at different pre-training epoch, *i.e.*, 10, 20, and 30. We can observe that the PQ metric shows a steady upward trend when we increase the pre-training epoch. However, we find a counter-intuitive experimental phenomenon when observing SQ metric, which is that SQ at $epoch = 10$ (74.0%) or 20 (74.5%) is significantly higher than SQ at $epoch = 50$ (68.0%). To explore the reason behind such phenomenon, we compare the segmentation quality (SQ) metric and the SQ changes (Δ) for partial classes at $epoch = 10$ and 50 in Tab. 6. From this table, we can clearly find that as the epoch increases, the 3D network's semantic segmentation ability for bicycle class decreases significantly ($\downarrow$ 95.0% SQ), where the bicycle class is almost absent from the superpixel semantics obtained by DINOV2 (please refer to the semantic distribution of superpixel obtained from DINOV2 in Fig. 2).

Pre-Training Epoch	Panoptic Segmentation					
	1%			5%		
	PQ	SQ	RQ	PQ	SQ	RQ
10	18.5	74.0	23.8	22.5	76.4	27.9
20	19.3	**74.5**	24.6	23.1	**76.9**	28.5
50	**19.7**	68.0	**25.0**	**23.4**	70.4	**28.7**

Table 5. Results (PQ, SQ, and RQ) of modifying the number of pre-training epoch. The row with the gray background represent the default number of pre-training epoch.

Classes	Epoch 10 (%)	Epoch 50 (%)	Δ (%)
SQ	74.0	68.0	$\downarrow$ 6.0
bicycle	95.0	0.0	$\downarrow$ 95.0
bus	73.0	83.7	$\uparrow$ 10.7
car	82.3	86.9	$\uparrow$ 4.6
motorcycle	90.3	81.3	$\downarrow$ 9.0
trailer	85.6	65.0	$\downarrow$ 20.6

Table 6. Results for SQ and partial class-wise SQ on nuScenes, when fine-tuning the pre-trained 3D backbone to panoptic segmentation with 1% lables. We compare two experimental settings with epochs of 10 and 50, respectively.

Study about variants of using multi-modality prototype. In Tab. 5 of main paper, we have shown the two results of using only 3D prototypes and employing MMPB. To further analyze, we conduct more variants about how to utilize the multi-modality prototypes. Observing the results in Tab 7, we find that (1) simply using raw 2D/3D prototypes, including #(1), #(2), and #(3), does not lead to performance improvement but rather to destructive effects; (2) When using our proposed MMPB #(5) or the modified version of MMPB without Modality-Specific Prototype Projection #(4), it significantly enhances the quality of the 3D pre-trained backbone and improves the performance on downstream task. The experimental results above demonstrate that our MMPB plays a key role in obtaining a high-quality 3D backbone, via effectively projecting and fusing 2D and 3D modality prototypes.

#	Variants	Semantic Seg.		
		1%	5%	10%
(1)	2D Prototype	40.3	54.5	61.1
(2)	3D Prototype	40.3	53.3	60.5
(3)	2D Pro. + 3D Pro.	41.5	54.2	60.7
(4)	MMPB w/o Proj.	45.7	56.7	62.7
(5)	MMPB	**47.0**	**57.0**	**63.3**

Table 7. Results (mIoU) of diverse variants of using multi-modality prototypes. #(1), #(2), and #(3) are three straightforward ways that employing raw 2D/3D prototypes. #(4) is the modified MMPB version without Modality-Specific Prototype Projection. #(5) denotes our complete MMPB.

B.4. 2D/3D Prototype Generation

. Here, we try other ways than average operation for generating 2D/3D prototypes. In Tab. 8, we try more operations on nuScenes, and observe that average operation is the best.

#	Operation	1% (mIoU)	5% (mIoU)	10% (mIoU)
(1)	Average (Default)	**47.0**	**57.0**	**63.3**
(2)	Max	46.7	56.7	62.2
(3)	Sum	14.2	16.6	18.6

Table 8. Results of three operations for prototype generation

B.5. Semantic Distribution of superpixel

To explore the superpixels generated by the VFM (DINOV2), we provide both a pie chart and a statistical table of the semantic distribution of superpixels in Fig. 2. We observe that only one-third of the superpixels, which are generated by DINOV2 and located within the view of point cloud, are relevant to our desired autonomous driving scenarios. Despite having only these related semantic categories, *i.e.*, scene-relevant semantic prototypes, our CSC model also achieves consistency enhancement of the 3D pre-trained model for the three downstream tasks. This result states that VFM-Assisted Semantic Prototype Generation and Coherent Semantic Consistency are effective in building a strong pre-training baseline for universal 3D large-scale perception.

C. Public Resources Used

We acknowledge the use of the following public resources, during the course of this work:

- nuScenes[1] ..... CC BY-NC-SA 4.0
- nuScenes-devkit[2] ..... Apache License 2.0
- SemanticKITTI[3] ..... CC BY-NC-SA 4.0
- MinkowskiEngine[4] ..... MIT License
- SLidR[5] ..... Apache License 2.0
- Cylinder3D[6] ..... Apache License 2.0
- PyTorch-Lightning[7] ..... Apache License 2.0
- OpenPCDet[8] ..... Apache License 2.0
- DINOv2[9] ..... Apache License 2.0
- OneFormer[10] ..... MIT License
- SpConv[11] ..... Apache License 2.0

References

[1] Youquan Liu, Lingdong Kong, Jun Cen, Runnan Chen, Wenwei Zhang, Liang Pan, Kai Chen, and Ziwei Liu. Segment any point cloud sequences by distilling vision foundation models. *arXiv preprint arXiv:2306.09347*, 2023. 2

[2] Anas Mahmoud, Jordan SK Hu, Tianshu Kuai, Ali Harakeh, Liam Paull, and Steven L Waslander. Self-supervised image-to-point distillation via semantically tolerant contrastive loss. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 7102–7110, 2023. 1, 2

[3] Lucas Nunes, Louis Wiesmann, Rodrigo Marcuzzi, Xieyuanli Chen, Jens Behley, and Cyrill Stachniss. Temporal consistent 3d lidar representation learning for semantic perception in autonomous driving. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5217–5228, 2023. 1

[4] Bo Pang, Hongchi Xia, and Cewu Lu. Unsupervised 3d point cloud representation learning by triangle constrained contrast for autonomous driving. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5229–5239, 2023. 2

[5] Corentin Sautier, Gilles Puy, Spyros Gidaris, Alexandre Boulch, Andrei Bursuc, and Renaud Marlet. Image-to-lidar self-supervised distillation for autonomous driving data. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 9891–9901, 2022. 1, 2

[6] Saining Xie, Jiatao Gu, Demi Guo, Charles R Qi, Leonidas Guibas, and Or Litany. Pointcontrast: Unsupervised pre-training for 3d point cloud understanding. In *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part III 16*, pages 574–591. Springer, 2020. 2

[7] Zaiwei Zhang, Rohit Girdhar, Armand Joulin, and Ishan Misra. Self-supervised pretraining of 3d features on any point-cloud. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 10252–10263, 2021. 2

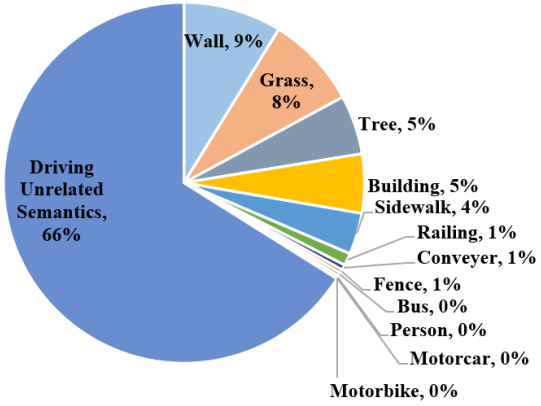
[8] Zhiwei Zhang, Zhizhong Zhang, Qian Yu, Ran Yi, Yuan Xie, and Lizhuang Ma. Lidar-camera panoptic segmentation via geometry-consistent and semantic-aware alignment. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 3662–3671, 2023. 1

[9] Bolei Zhou, Hang Zhao, Xavier Puig, Tete Xiao, Sanja Fidler, Adela Barriuso, and Antonio Torralba. Semantic understanding of scenes through the ade20k dataset. *International Journal of Computer Vision*, 127(3):302–321, 2019. 2, 5

[10] Zixiang Zhou, Yang Zhang, and Hassan Foroosh. Panoptic-polarnet: Proposal-free lidar point cloud panoptic segmentation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 13194–13203, 2021. 1

[11] Xinge Zhu, Hui Zhou, Tai Wang, Fangzhou Hong, Yuexin Ma, Wei Li, Hongsheng Li, and Dahua Lin. Cylindrical and asymmetrical 3d convolution networks for lidar segmentation. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 9939–9948, 2021. 1

[1] https://www.nuscenes.org/nuscenes
[2] https://github.com/nutonomy/nuscenes-devkit
[3] http://semantic-kitti.org
[4] https://github.com/NVIDIA/MinkowskiEngine
[5] https://github.com/valeoai/SLidR
[6] https://github.com/xinge008/Cylinder3D
[7] https://github.com/Lightning-AI/lightning
[8] https://github.com/open-mmlab/OpenPCDet
[9] https://github.com/facebookresearch/dinov2
[10] https://github.com/SHI-Labs/OneFormer
[11] https://github.com/traveller59/spconv



Superpixel Semantic	Counts	Percent (‰)
Wall	80.4K	88‰
Grass	75.8K	83‰
Tree	48.5K	53‰
Building	48.3K	52‰
Sidewalk	33.9K	37‰
Railing	10.1K	11‰
Conveyer	4.3K	4‰
Fence	2.9K	3‰
Bus	2.8K	3‰
Person	2.5K	2‰
Motorcar	110	0‰
Motorbike	28	0‰
Driving Unrelated Classes	603K	661‰

Figure 2. Semantic classes and number of superpixels generated by DINOV2 on the nuScenes dataset. We provide both the pie chart (left half) and the statistical table (right half) to present the semantic distribution of the superpixels. Note that since we use DINOV2 fine-tuned on the ADE20K dataset [9], the network classifies each superpixel in 150 categories, most of which are categories that are unrelated to autonomous driving scenes, such as bed, bench, book, etc. We denote these classes as Driving Unrelated Semantics, which amounted to 603K and accounted for 661‰ of the total.