Supplementary Material for
Temporal Consistent 3D LiDAR Representation Learning for Semantic Perception in Autonomous Driving

Abstract

This supplementary material provides more detailed information for the proposed pre-training method and fine-tuning, with further ablations and results. Appendix A provides diagrams of the backbone used in our experiments and the fine-tuning pipelines for the three downstream tasks. Appendix B shows examples of the sampled temporal views from the batches during pre-training. Next, we show ablations regarding the individual modules of our method in Appendix C. Finally, we provide further results on the fine-tuned downstream tasks in Appendix D and qualitative results for semantic and panoptic segmentation in Appendix E. Furthermore, we provide our code within this supplementary material.

A. Diagrams

This section shows the diagrams of the backbone used in our experiments and the pipeline for the fine-tuned downstream tasks. Appendix A.1 shows the diagram of the backbone model used in our evaluation. Appendix A.2 presents the network diagrams during fine-tuning, where Appendix A.2.1 shows the diagram for the semantic segmentation task, Appendix A.2.2 for the panoptic segmentation, and Appendix A.2.3 for the object detection task.

A.1. Backbone network

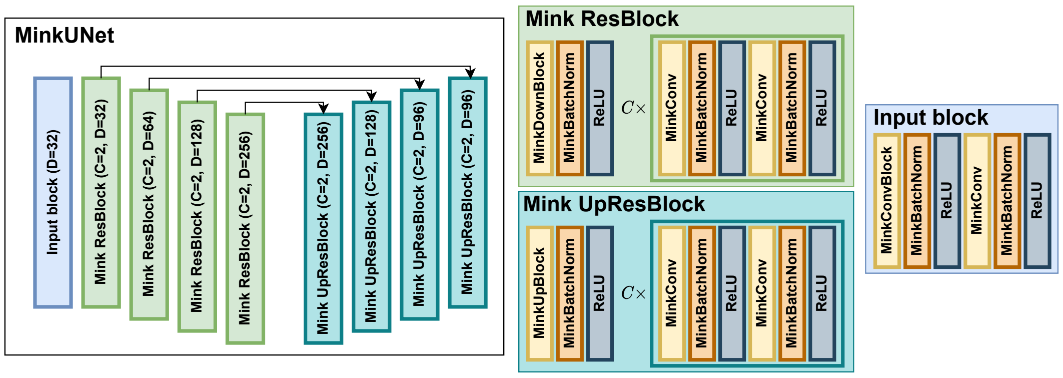


Figure 1. Diagram of the MinkUNet backbone used in the main paper. The backbone used a ResUNet architecture implemented with sparse operations implemented by the Minkowski Engine [4] library. The Mink ResBlock layers are downsampling layers with C residual blocks and D output features. The Mink UpResBlock are upsampling layers also with C residual blocks and D output features.

We use the ResUNet implemented with the Minkowski Engine [4] library (MinkUNet) as the backbone to evaluate our method. This library implements the commonly used deep neural network layers as sparse operations. This network voxelizes the point cloud to then apply sparse convolutions over the voxelized point clouds to extract features. Fig. 1 depicts the used backbone with output feature sizes for each layer, where D is the output feature size, and C is the number of residual blocks for each layer.

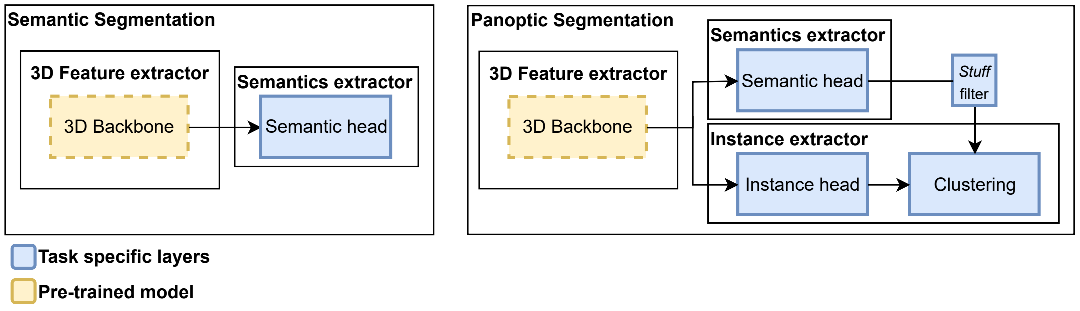
A.2. Downstream tasks

To evaluate the pre-training methods, we fine-tune the pre-trained models to three tasks: semantic segmentation, panoptic segmentation, and object detection. Also, to measure the amount of information extracted during pre-training, we evaluate each downstream task with the full training set and fewer labels and compare the performance with the model trained without pre-training. In all the tasks, we use the MinkUNet as the 3D backbone (see Appendix A.1), which is pre-trained with the different self-supervised methods and compare the performance on each of the downstream tasks.

Percentage of labels	0.1%	1%	10%	50%	100%
Number of epochs	300	120	40	20	15

Table 1. Number of training epochs during fine-tuning to the different percentage of scans for semantic segmentation.

Training subsets. For semantic and panoptic segmentation on SemanticKITTI [2, 6], we use the 0.1%, 1%, 10%, 50%, and 100% subsets of scans used by Nunes *et al.* [7]. Those subsets are randomly sampled scans from the entire training sequences such that all classes are present. Besides that, the number of iterations decreases when training with fewer scans for the same amount of epochs. To overcome that, Nunes *et al.* [7] increase the number of training epochs as the amount of labels decreases. Therefore, as in SegContrast [7], we train for more epochs when fine-tuning with fewer labels on semantic segmentation. The amount of epochs for each percentage of the training subset is listed in Tab. 1. For panoptic segmentation, we use the same number of training epochs for all the label subsets. For object detection, we use the KITTI [6] dataset. In this case, we use the subset of scans samples used by Zhang *et al.* [11]. In their evaluation, they randomly sample three subsets of scans from the full training sequences and report the mean performance over the three subsets. We use the same subsets to evaluate when training with 10% and 20% of scans. Lastly, for nuScenes [3, 9], we use the full training set and the provided mini training subset. On all the datasets, we evaluate the fine-tuned model performance on the full validation set.



(a) Semantic segmentation diagram.

(b) Panoptic segmentation diagram.

Figure 2. Diagrams for semantic and panoptic segmentation used to fine-tune the pre-trained models to evaluate the different self-supervised pre-training methods.

A.2.1 Semantic segmentation

For semantic segmentation, the pipeline is composed of a 3D point-wise feature extractor in which we used the MinkUNet. Then the second module is a semantics extractor. In the pipeline provided by Nunes *et al.* [7], the semantics extractor is a single linear layer to predict a semantic class for each point with the features extracted by the backbone. Fig. 2a depicts this semantic segmentation pipeline.

A.2.2 Panoptic segmentation

The panoptic segmentation uses a 3D point-wise feature extractor in which we also use the MinkUNet. Together with the 3D backbone, we use a semantics extractor, the same as the one used in the semantic segmentation pipeline, and an instance extractor. The instance extractor comprises an instance head layer, followed by a clustering step. The clustering step uses the semantic predictions to filter *stuff* points, leaving only points belonging to *things* instances such as cars, pedestrians, and bicyclists. After filtering out *stuff* points, the remaining points are clustered to identify the instances from the instance head features. For clustering, we use mean shift [5] with bandwidth set to 1.2 and the minimum number of points per cluster as 50. The pipeline diagram is shown in Fig. 2b.

A.2.3 Object detection

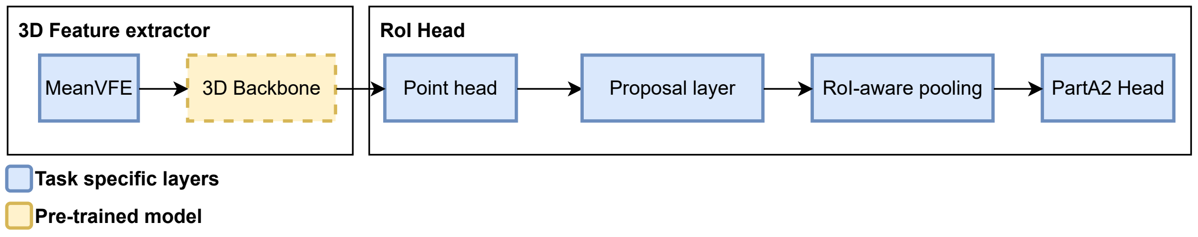


Figure 3. Diagram for object detection pipeline used for fine-tuning the pre-trained models to evaluate the different self-supervised pre-training methods.

The object detection uses the same MinkUNet as a 3D point-wise feature extractor. However, first the voxel features are replaced by the mean feature of the points inside the voxel. Next, the region of interest (RoI) module uses point-wise features to extract object proposals from the full point cloud and uses the PartA2 head [8] to predict the final objects and their classes. The object detection pipeline diagram is depicted in Fig. 3.

B. Temporal object views examples

This section presents examples of the sampled batches $\mathcal{B}_b$ sampled during pre-training. As explained in the main paper, we segment the ground over the individual scans, aggregate the scans and cluster the remaining points to extract the temporal object views. We segment the ground over the individual scans instead of the aggregated scans to save the ground labels in a pre-processing step and save computation time during training. Figs. 4 to 8 show the aggregated scans $\mathcal{P}$ together with the first scan $\mathcal{P}^1$, and the last scan $\mathcal{P}^{12}$, from the batch with $n = 12$. In the examples, it is also possible to see how moving and static objects are associated in the different scans. Each extracted segment is colored with different colors. We highlight in red the moving objects and in green the static objects. In those examples, the structures in the environment could be segmented using our approach. And by keeping the index of the points we could extract views of the same object with very different appearances, ensuring that both are from the same object. These views are also extracted from moving objects, tracking those views even in high-speed scenarios such as a highway (see Fig. 6).

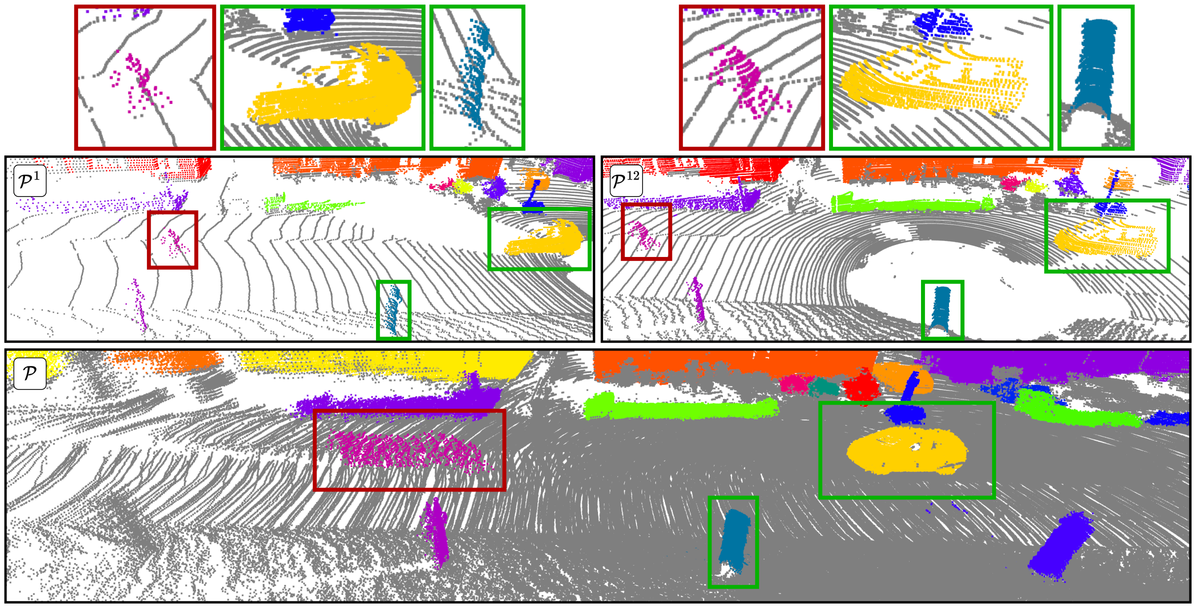


Figure 4. Example of the aggregated scans $\mathcal{P}$ and the first and last scan from the interval, *i.e.*, $\mathcal{P}^1$ and $\mathcal{P}^{12}$. Moving objects are highlighted in red, static objects highlighted in green. (Best viewed in color)

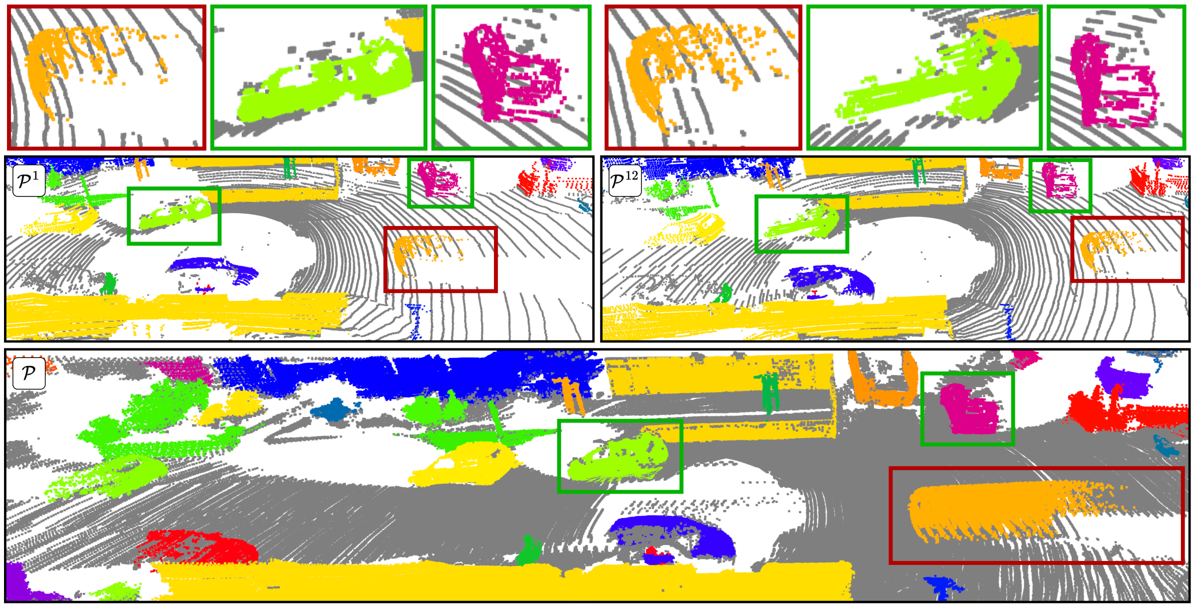


Figure 5. Example of the aggregated scans $\mathcal{P}$ and the first and last scan from the interval, *i.e.*, $\mathcal{P}^1$ and $\mathcal{P}^{12}$. Moving objects are highlighted in red, static objects highlighted in green. (Best viewed in color)

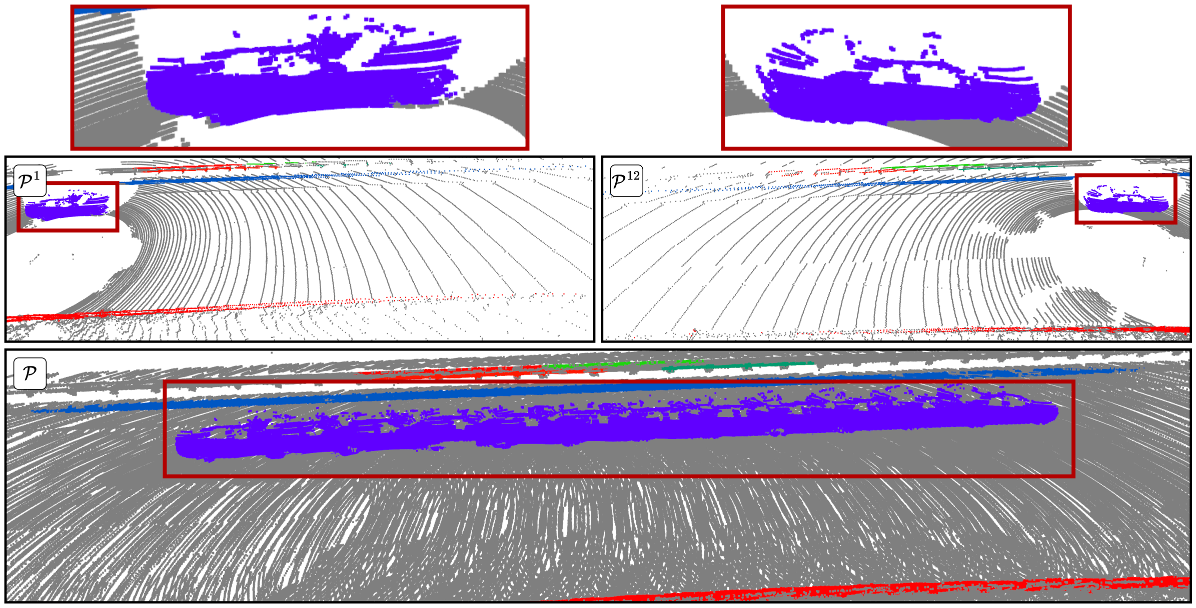


Figure 6. Example of the aggregated scans $\mathcal{P}$ and the first and last scan from the interval, *i.e.*, $\mathcal{P}^1$ and $\mathcal{P}^{12}$. Moving objects are highlighted in red, static objects highlighted in green. (Best viewed in color)

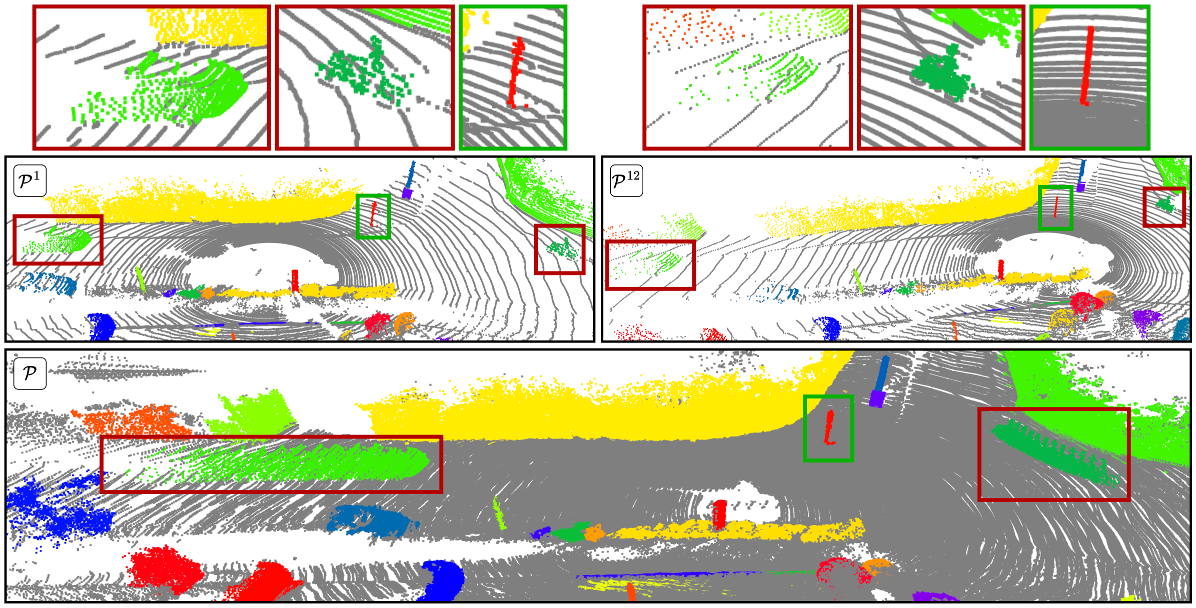


Figure 7. Example of the aggregated scans $\mathcal{P}$ and the first and last scan from the interval, *i.e.*, $\mathcal{P}^1$ and $\mathcal{P}^{12}$. Moving objects are highlighted in red, static objects highlighted in green. (Best viewed in color)

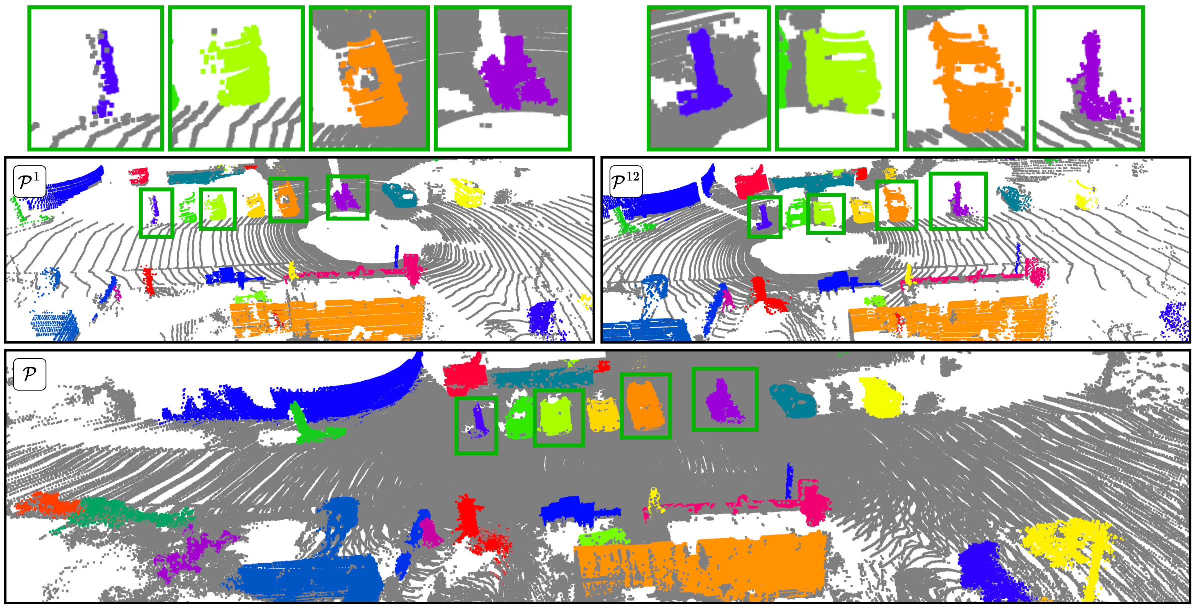


Figure 8. Example of the aggregated scans $\mathcal{P}$ and the first and last scan from the interval, *i.e.*, $\mathcal{P}^1$ and $\mathcal{P}^{12}$. Moving objects are highlighted in red, static objects highlighted in green. (Best viewed in color)

C. Ablations

This section presents further ablations regarding the pre-training step and its modules. Appendix C.1 compares the results when using temporal views of the objects as augmented pairs compared with using only data augmentation to generate the pairs. Next, Appendix C.2 reports results comparing the pre-training using the commonly used non-linear projection head with the Transformer projection head. Lastly, Appendix C.3 presents the results using a different number of scans n to extract the temporal views of the objects.

In these ablations, we evaluate the network by fine-tuning the pre-trained network on semantic segmentation. We compare the results in the lowest label regimes, *i.e.*, 0.1%, 1%, and 10%, since those are regimes with more significant performance differences between the methods. Besides, we compare the linear evaluation results to measure the amount of semantic information embedded in the different ablations.

C.1. Temporal views

	0.1%	1%	10%	Linear eval
D.A. only	38.40	50.08	59.98	28.50
Temporal views + D.A.	**38.59**	**51.42**	**60.34**	**35.90**

Table 2. Ablation comparing the pre-training performance using only data augmentation (D.A.) and our proposed temporal views extraction together with the data augmentation.

In this ablation, we want to compare the performance of our method pre-trained with temporal views of an object as natural augmented pairs instead of using only data augmented to generate artificial augmented pairs from the same scan. Tab. 2 shows the results when generating augmented pairs using only data augmentation and when extracting temporal views of one object and then applying data augmentation to generate the augmented pairs. As can be seen, extracting natural augmentations of one object using the temporal views can boost the network performance when fine-tuning with different amounts of labels. Also, the linear evaluation results suggest that by learning from views of the same object at different points in time, the representation can extract more semantic knowledge during pre-training.

C.2. Transformer projection head

	0.1%	1%	10%	Linear eval
Non-linear projection head	38.19	50.22	56.54	32.27
Transformer projection head	**38.59**	**51.42**	**60.34**	**35.90**

Table 3. Ablation comparing the pre-training performance using a non-linear projection head and a Transformer encoder as a projection head.

In this ablation, we compare the pre-training performance using the common non-linear projection head with the Transformer projection head. In this experiment, we use a non-linear projection head composed of one linear layer, a batch normalization and a ReLU layer, and another linear layer. Tab. 3 shows that the network could achieve better performance with the Transformer projection head when fine-tuning with different label percentages. Also, similarly to the results reported in Appendix C.1, the linear evaluation suggests that the network pre-trained using the Transformer projection head could learn a representation that embeds more semantic information, achieving better mIoU.

C.3. Different number of aggregated scans

Number of scans	0.1%	1%	10%	Linear eval
30	36.36	44.14	56.41	32.27
18	38.19	45.76	58.39	32.50
15	38.15	50.44	57.84	33.52
12	38.59	51.42	**60.34**	35.90
9	38.30	**51.82**	57.42	35.97
6	**38.75**	51.37	59.57	**36.88**

Table 4. Ablation comparing the pre-training performance using a different number of n scans interval to extract temporal views of the objects.

In this ablation, we pre-train the network with different n values for the number of scans in the interval to extract the object views. We report the results in Tab. 4 to compare our chosen $n = 12$ with different sizes of the scans intervals. From $n = 6$ to $n = 12$, the results have just marginal differences. However, when fine-tuning with 10% of labels, we can see that the pre-training with $n = 12$ achieves the best performance, also performing considerably better than the network trained from scratch with 100% of labels (as reported in the main paper). Furthermore, when using bigger n the network performance decreases. Therefore, we decided to stick with the $n = 12$ as reported in the main paper.

C.4. Positional encoding

	0.1%	1%	10%	Linear eval
With positional encoding	**38.81**	49.79	59.68	35.89
Without positional encoding	38.59	**51.42**	**60.34**	**35.90**

Table 5. Ablation comparing the pre-training performance with and without positional encoding.

For Transformer encoders, positional encoding is usually used together with the input embeddings [1]. However, we do not use the positional encoding in our method. In this section, we report the results with and without the positional encoding. As shown in Tab. 5, the network trained without positional encoding achieves better performance than with positional encoding. We hypothesize that the backbone needs to extract less information from the input point clouds when inserting positional information in the Transformer projection head, performing worst when fine-tuning the pre-trained model. Therefore, we decided to remove the positional encoding from our pre-training scheme.

D. Further results

This section reports further results on all three downstream tasks in which we fine-tuned the pre-trained models. First, we discuss the baselines adaptations for pre-training on LiDAR data in Appendix D.1. Second, we present in Appendix D.2 the IoU for each class in both SemanticKITTI [2, 6] and nuScenes [3, 9] datasets. Next, Appendix D.3 presents the recognition quality (RQ) and segmentation quality (SQ) results. Lastly, in Appendix D.4, we present the results for object detection in KITTI dataset [6] fine-tuned with the subsets of labels used by Zhang *et al.* [11]. In all the experiments, we compare our method with the same baselines used in the main paper, *i.e.*, PointContrast (PC), DepthContrast (DC), and SegContrast (SC).

D.1. Baseline adaptations

Given that the PointContrast baseline was thought for indoor data, some adaptations were necessary to pre-train the model using LiDAR data. To pre-train the model using PointContrast, it was necessary to adapt the pre-processing step to find correspondences between points in sequential scans. To do so, we have used the poses provided in SemanticKITTI to aggregate the scans. Then, using the code available in the original PointContrast repo, we have associated points with their nearest neighbor in the aggregated point cloud to define the point-wise positive pairs to be used during pre-training. For the other baselines, SegContrast and DepthContrast, no adaptation was necessary.

D.2. Semantic classes IoU

In this section, we report the IoU for the individual classes after fine-tuning to SemanticKITTI [2, 6] and nuScenes [3, 9] datasets. For SemanticKITTI, we report the results after fine-tuning the network with the linear evaluation setup and fine-tuning with the 10% subset labels used by Nunes *et al.* [7]. For nuScenes, we report the results also after the linear evaluation setup training and after fine-tuning with the mini training subset.

	Car	Bicycle	Motorcycle	Truck	Other-vehicle	Person	Bicyclist	Motorcyclist	Road	Parking	Sidewalk	Other-ground	Building	Fence	Vegetation	Trunk	Terrain	Pole	Traffic-sign	mIoU
Scratch	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	30.10	0.0	1.53	0.0	3.60	0.0	42.67	0.0	0.0	0.0	0.0	4.10
PC [10]	**90.76**	0.0	0.0	0.02	1.82	0.0	0.0	0.0	**66.94**	0.0	**41.61**	0.0	**85.99**	11.02	77.45	41.19	**58.76**	9.53	0.02	25.53
DC [11]	37.36	0.0	0.0	0.0	0.01	0.0	0.0	0.0	45.00	0.01	17.73	0.0	44.02	10.09	59.94	14.88	20.25	13.14	0.0	13.87
SC [7]	82.18	0.0	**0.81**	**0.26**	**2.09**	8.01	0.41	0.0	55.83	0.74	27.11	0.0	80.02	26.39	**81.16**	46.65	40.33	49.91	19.13	27.42
TARL	87.12	0.0	0.20	0.06	0.06	**36.47**	**53.53**	0.0	57.70	**3.18**	25.14	0.0	82.84	**29.38**	80.61	**55.76**	48.21	**50.87**	**36.85**	**35.90**

Table 6. Linear evaluation results after pre-training on SemanticKITTI and training a linear head on top of the frozen backbone also on SemanticKITTI, reporting IoU for each class, where PC stands for PointContrast, DC for DepthContrast and SC for SegContrast.

Tab. 6 shows the IoU for each class in SemanticKITTI after training only a linear head on top of the frozen pre-trained backbone. In the classes where our method does not have the best performance, we can notice that our results are close to the best ones. However, in some classes, such as person, bicyclist, and traffic sign, all the methods achieve poor performance, even close to 0.0% IoU. In those classes, our method outperforms the baselines by a large margin.

	Car	Bicycle	Motorcycle	Truck	Other-vehicle	Person	Bicyclist	Motorcyclist	Road	Parking	Sidewalk	Other-ground	Building	Fence	Vegetation	Trunk	Terrain	Pole	Traffic-sign	mIoU
Scratch	94.56	3.40	33.55	65.68	39.40	42.31	65.98	0.03	90.54	40.10	75.33	0.17	90.54	53.44	88.49	63.38	75.48	57.22	45.56	53.96
PC [10]	95.08	26.96	55.71	70.01	45.42	**62.37**	**81.37**	0.51	91.17	40.88	74.90	**1.30**	90.58	48.72	86.64	60.36	72.46	60.28	50.17	58.68
DC [11]	94.67	9.81	51.06	64.15	37.16	54.86	77.89	0.0	91.25	38.16	76.40	0.91	90.73	53.05	88.86	**66.54**	75.67	60.65	**53.28**	57.11
SC [7]	94.55	9.89	54.31	58.98	39.46	54.48	74.71	0.0	90.15	37.71	73.75	0.05	90.18	53.67	**89.08**	60.86	**76.61**	60.10	51.35	56.31
TARL	**95.42**	**29.28**	**57.23**	**70.48**	**50.58**	60.78	79.89	**2.54**	**91.77**	**42.04**	**76.67**	0.28	**91.02**	**55.77**	88.92	63.74	75.91	**60.99**	53.09	**60.34**

Table 7. Results after pre-training on SemanticKITTI and fine-tuning to it with 10% of the labels, reporting IoU for each class from SemanticKITTI dataset, where PC stands for PointContrast, DC for DepthContrast and SC for SegContrast.

Tab. 7 shows the individual classes IoU after fine-tuning the pre-trained networks to the 10% labels subset. In these results, we can see that after fine-tuning with 10% of the labeled scans, our method achieved the best performance in most of the classes compared to the previous methods. When computing the mIoU, we can see that those improvements brought by our network achieve an outstanding performance compared to the baselines.

	Barrier	Bicycle	Bus	Car	Construction-vehicle	Motorcycle	Pedestrian	Traffic cone	Trailer	Truck	Drivable surface	Other-flat	Sidewalk	Terrain	Manmade	Vegetation	mIoU
Scratch	0.08	0.0	0.20	4.01	0.0	0.14	0.0	0.0	0.0	0.0	47.60	0.0	0.0	0.31	34.41	3.25	5.62
PC [10]	4.97	0.0	0.0	59.23	0.01	0.03	1.40	0.52	0.01	9.30	73.08	0.0	**12.55**	36.47	56.91	60.75	19.70
DC [11]	0.14	0.0	0.08	15.94	0.0	0.0	0.59	0.0	0.04	2.16	61.60	0.0	0.06	3.34	43.36	27.13	9.65
SC [7]	10.05	0.0	0.37	63.18	**0.06**	15.43	16.18	0.70	**0.74**	20.33	**72.42**	**0.31**	9.34	35.12	**67.05**	68.77	23.75
TARL	**12.03**	**0.10**	**5.16**	**63.44**	0.02	**23.31**	**31.73**	**7.36**	0.43	**20.95**	72.18	0.12	5.92	**39.80**	65.70	**68.98**	**26.08**

Table 8. Linear evaluation results after pre-training on SemanticKITTI and training a linear head on top of the frozen backbone on nuScenes, reporting IoU for each class from nuScenes dataset, where PC stands for PointContrast, DC for DepthContrast and SC for SegContrast.

In Tab. 8, we report the results after pre-training on SemanticKITTI and training a linear head on top of the frozen pre-trained backbone on the nuScenes dataset. In this transfer learning setup, our method could better transfer the knowledge extracted from one dataset to the other. In most classes, our method outperforms the baselines achieving the best overall performance (mIoU).

	Barrier	Bicycle	Bus	Car	Construction-vehicle	Motorcycle	Pedestrian	Traffic cone	Trailer	Truck	Drivable surface	Other-flat	Sidewalk	Terrain	Manmade	Vegetation	mIoU
Scratch	20.49	0.0	6.75	53.66	2.28	0.0	10.35	0.36	7.02	17.78	86.29	13.46	33.22	40.65	66.24	72.41	26.94
Sup. pre-train	**29.63**	0.0	19.93	**76.57**	**7.52**	11.15	**44.23**	10.69	**8.45**	33.01	**89.99**	**21.00**	**47.43**	**60.73**	**76.68**	77.17	38.39
PC [10]	21.73	0.0	15.64	71.68	5.08	0.0	20.86	2.29	7.03	16.94	88.60	15.60	40.31	55.99	73.22	75.70	31.92
DC [11]	20.03	0.0	8.82	62.35	4.71	0.63	13.37	2.74	8.40	20.07	85.51	13.45	33.26	37.07	64.07	72.09	27.81
SC [7]	23.50	0.0	10.73	71.70	5.10	0.22	28.80	8.47	8.16	18.72	84.39	16.55	33.27	45.10	69.89	75.66	31.27
TARL	24.59	0.0	**21.95**	74.27	6.97	**41.69**	42.70	**11.18**	8.18	**37.50**	88.54	19.39	41.68	55.64	76.36	**79.05**	**39.36**

Table 9. Results after pre-training on SemanticKITTI and fine-tuning on nuScenes mini training subset, reporting IoU for each class from nuScenes dataset, where PC stands for PointContrast, DC for DepthContrast and SC for SegContrast.

Lastly, we evaluate the network pre-trained on SemanticKITTI by fine-tuning it to nuScenes mini training subset, reporting the results in Tab. 9. In this evaluation, we also compare the network fully supervised pre-trained on SemanticKITTI, and fine-tuned to nuScenes. The table shows that the supervised pre-trained network achieves the best performance in most of the classes, which is expected since it has been trained for semantic segmentation even though using a different dataset. However, our method achieves a performance close to the supervised pre-trained network in those classes and surpasses it by a large margin in some classes, *e.g.*, motorcycle and truck. In the overall performance (mIoU), our method outperforms both the baselines and the fully supervised pre-trained network.

D.3. Panoptic segmentation

In this section we report the recognition quality (RQ) and segmentation quality (SQ) in addition to the panoptic quality and mIoU reported in the main paper.

	0.1%		1%		10%		50%		100%	
Method	RQ	SQ	RQ	SQ	RQ	SQ	RQ	SQ	RQ	SQ
Scratch	7.70	15.17	30.57	57.34	57.31	73.81	65.52	77.11	55.40	77.43
PC [10]	9.29	18.48	36.83	58.93	57.62	75.12	64.37	77.34	66.04	78.02
DC [11]	11.70	19.45	36.35	55.76	56.94	70.92	64.61	73.97	66.43	80.89
SC [7]	11.79	22.10	34.79	58.91	57.01	71.41	65.75	73.75	**67.20**	82.08
TARL	**15.36**	**22.75**	**38.02**	**60.25**	**61.38**	**77.24**	**66.53**	**77.66**	66.38	**83.21**

Table 10. Results (RQ and SQ) when fine-tuning the pre-trained models to panoptic segmentation with different percentage of labels on SemanticKITTI, where PC stands for PointContrast, DC for DepthContrast and SC for SegContrast.

Similarly to the results reported in the main paper, the RQ and SQ reported in Tab. 10 show that our method outperforms the baselines in those metrics. These results suggest that our method can extract semantic and instance-level information during pre-training.

	Mini		Full	
Method	RQ	SQ	RQ	SQ
Scratch	30.82	58.82	63.25	69.23
Sup. pre-train	31.82	60.90	**63.49**	69.31
PC [10]	34.20	60.46	61.06	68.98
DC [11]	36.64	62.16	61.73	68.91
SC [7]	36.48	62.52	62.57	69.09
TARL	**40.47**	**63.84**	63.36	**69.57**

Table 11. Results (RQ and SQ) when fine-tuning the pre-trained models to panoptic segmentation on the full and mini training sets from nuScenes, where PC stands for PointContrast, DC for DepthContrast and SC for SegContrast.

Tab. 11 reports recognition and segmentation quality results for the model pre-trained on SemanticKITTI and then fine-tuned to nuScenes. As in the results reported in the main paper, our method outperforms the baselines and performs better than the fully supervised pre-training in almost all the evaluations. These results suggest that our approach could replace the supervised pre-training on LiDAR data domain.

D.4. Object detection subsets

In this section, we report the object detection results using the training subsets used by Zhang *et al*. [11] in addition to the results reported in the main paper. We pre-train the network with SemanticKITTI and report in this section the fine-tuning using the KITTI training with 10% and 20% of labels. For both 10% and 20% subsets, three random subsets from the 100% scans are sampled. Then, the network is pre-trained in each random subset, and the reported results are the average performance on the subsets. As in the main paper, we report the average precision (AP) with 40 recall positions for car, pedestrian and cyclist, on all three difficulty levels.

	Car			Pedestrian			Cyclist		
	Easy	Moderate	Hard	Easy	Moderate	Hard	Easy	Moderate	Hard
Scratch	**86.86**	73.17	69.73	70.34	62.87	55.98	87.94	69.12	64.96
PC [10]	85.92	73.62	69.48	**71.04**	**64.67**	**58.09**	87.28	67.51	63.29
DC [11]	86.72	72.80	69.58	69.52	63.36	56.51	87.77	68.95	64.85
SC [7]	86.61	**74.29**	**70.56**	69.98	64.19	57.24	**88.95**	**70.22**	**65.81**
TARL	85.66	72.08	67.53	70.12	62.63	55.51	87.06	67.20	62.94

Table 12. Results (AP) when fine-tuning the pre-trained models to object detection on 10% training subsets from KITTI for car, pedestrian and cyclist classes on the easy, moderate and hard difficult levels, where PC stands for PointContrast, DC for DepthContrast and SC for SegContrast.

Tab. 12 presents the average of the results after fine-tuning the network with the three 10% label subsets. Unlike the other downstream tasks, neither pre-training boosts the network performance compared with the model trained from scratch. Instead, the differences between the baselines and the network trained from scratch are minor. Therefore, we report the best performance in bold and the second best in gray. As can be seen, the network trained from scratch achieves the best or the second best most of the time, showing that none of the pre-trainings could improve the performance when training with fewer labels. Unlike the results reported in the main paper, our method does not achieve the best performance training with few labels, however, all the methods achieve a performance similar to the network trained from scratch.

	Car			Pedestrian			Cyclist		
	Easy	Moderate	Hard	Easy	Moderate	Hard	Easy	Moderate	Hard
Scratch	89.06	77.07	72.91	67.86	61.24	55.58	88.70	70.26	65.58
PC [10]	88.93	77.14	73.84	69.01	63.42	57.07	**90.92**	71.73	**67.60**
DC [11]	89.07	77.22	**74.68**	68.22	62.70	56.57	89.80	71.49	67.33
SC [7]	**89.62**	**78.05**	**75.16**	**69.68**	**64.00**	**58.17**	89.50	**71.93**	**68.11**
TARL	**89.12**	**77.34**	74.26	**69.31**	**63.46**	**57.33**	**90.41**	**71.74**	67.55

Table 13. Results (AP) when fine-tuning the pre-trained models to object detection on 20% training subsets from KITTI for car, pedestrian and cyclist classes on the easy, moderate and hard difficult levels, where PC stands for PointContrast, DC for DepthContrast and SC for SegContrast.

In Tab. 13, our network achieves the second best in most columns. However, similarly to Tab. 12, all the methods achieve similar performance, with minor differences compared to the network trained from scratch. For object detection, we could notice the most significant difference when training the model with the full labels, reported in the main paper, where our method achieves the best performance.

E. Qualitative results

In this section, we provide qualitative results comparing the pre-training methods when fine-tuning them to semantic and panoptic segmentation on SemanticKITTI dataset.

E.1. Semantic segmentation

In this section, we compare the semantic segmentation results between the network trained from scratch and after fine-tuning with the different self-supervised pre-training methods with only 0.1% of the labeled scans for semantic segmentation. Figs. 9 to 12 show the results comparison between the different methods and our proposed pre-training approach.

E.2. Panoptic segmentation

In this section, we compare the instance predictions from panoptic segmentation results between the network trained from scratch and after fine-tuning with the different self-supervised pre-training methods with only 10% of the labeled scans for panoptic segmentation. Figs. 13 to 16 shows the comparison of the results between the different methods and our proposed pre-training approach.

References

[1] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Proc. of the Conference on Neural Information Processing Systems (NeurIPS)*, 2017. 7

[2] Jens Behley, Martin Garbade, Andres Milioto, Jan Quenzel, Sven Behnke, Cyrill Stachniss, and Juergen Gall. SemanticKITTI: A Dataset for Semantic Scene Understanding of LiDAR Sequences. In *Proc. of the IEEE/CVF Intl. Conf. on Computer Vision (ICCV)*, 2019. 2, 8

[3] Holger Caesar, Varun Bankiti, Alex H. Lang, Sourabh Vora, Venice Erin Liong, Qiang Xu, Anush Krishnan, Yu Pan, Giancarlo Baldan, and Oscar Beijbom. nuScenes: A Multimodal Dataset for Autonomous Driving. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020. 2, 8

[4] Christopher Choy, JunYoung Gwak, and Silvio Savarese. 4D Spatio-Temporal ConvNets: Minkowski Convolutional Neural Networks. In *Proc. of the IEEE/CVF Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2019. 1

[5] Dorin Comaniciu and Peter Meer. Mean Shift: A Robust Approach Toward Feature Space Analysis. *IEEE Trans. on Pattern Analalysis and Machine Intelligence (TPAMI)*, 24:603–619, 2002. 3

[6] Andreas Geiger, Philip Lenz, and Raquel Urtasun. Are we ready for Autonomous Driving? The KITTI Vision Benchmark Suite. In *Proc. of the IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, pages 3354–3361, 2012. 2, 8

[7] Lucas Nunes, Rodrigo Marcuzzi, Xieyuanli Chen, Jens Behley, and Cyrill Stachniss. SegContrast: 3D Point Cloud Feature Representation Learning through Self-supervised Segment Discrimination. *IEEE Robotics and Automation Letters (RA-L)*, 2022. 2, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20

[8] Shaoshuai Shi, Zhe Wang, Jianping Shi, Xiaogang Wang, and Hongsheng Li. From Points to Parts: 3D Object Detection from Point Cloud with Part-aware and Part-aggregation Network. *IEEE Trans. on Pattern Analalysis and Machine Intelligence (TPAMI)*, 43:2647–2664, 2021. 3

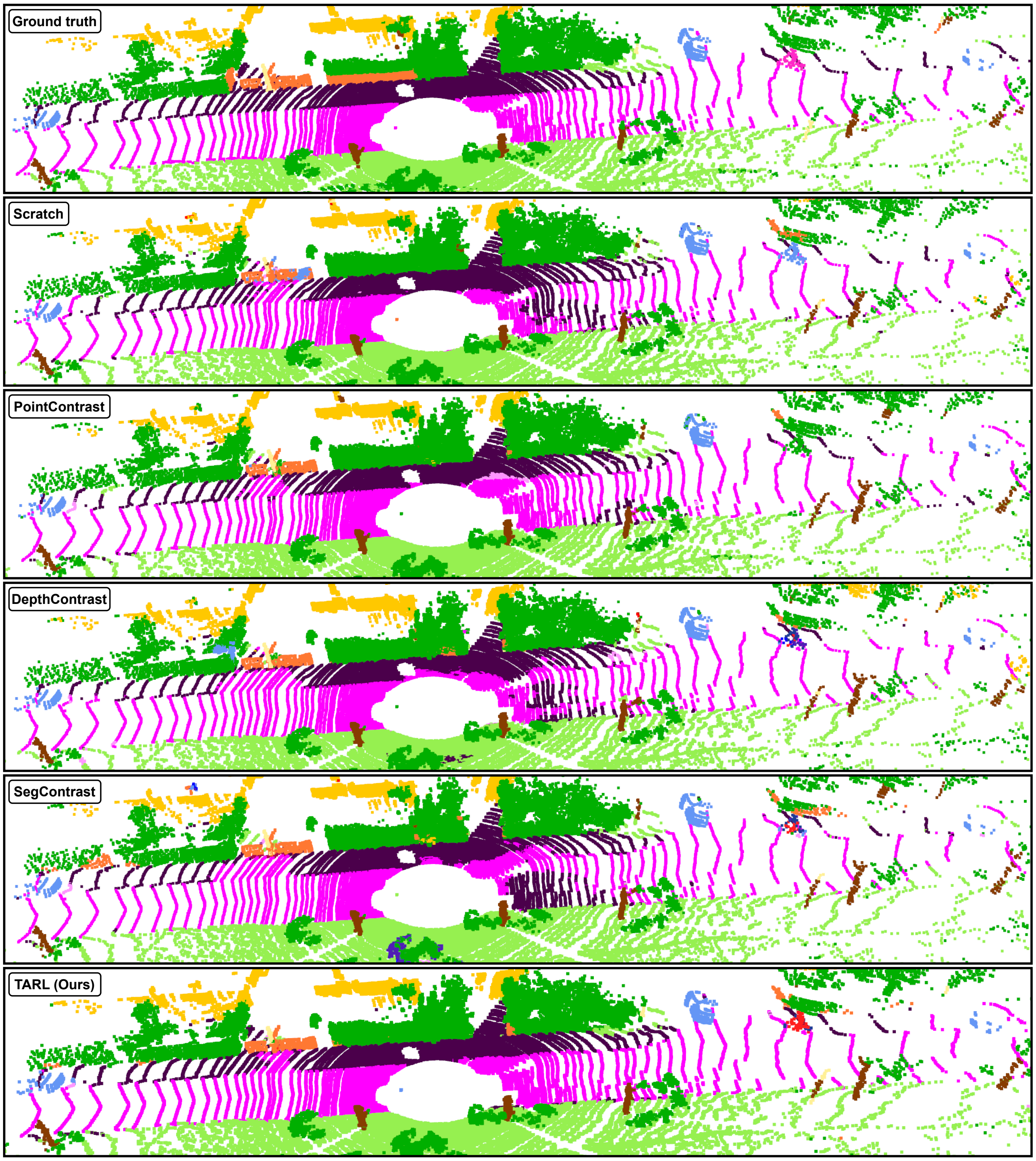


Figure 9. Qualitative comparison between the network trained from scratch and after pre-training with the different pre-training methods for semantic segmentation. We compare the baselines, PointContrast [10], DepthContrast [11], SegContrast [7] and our proposed method TARL.

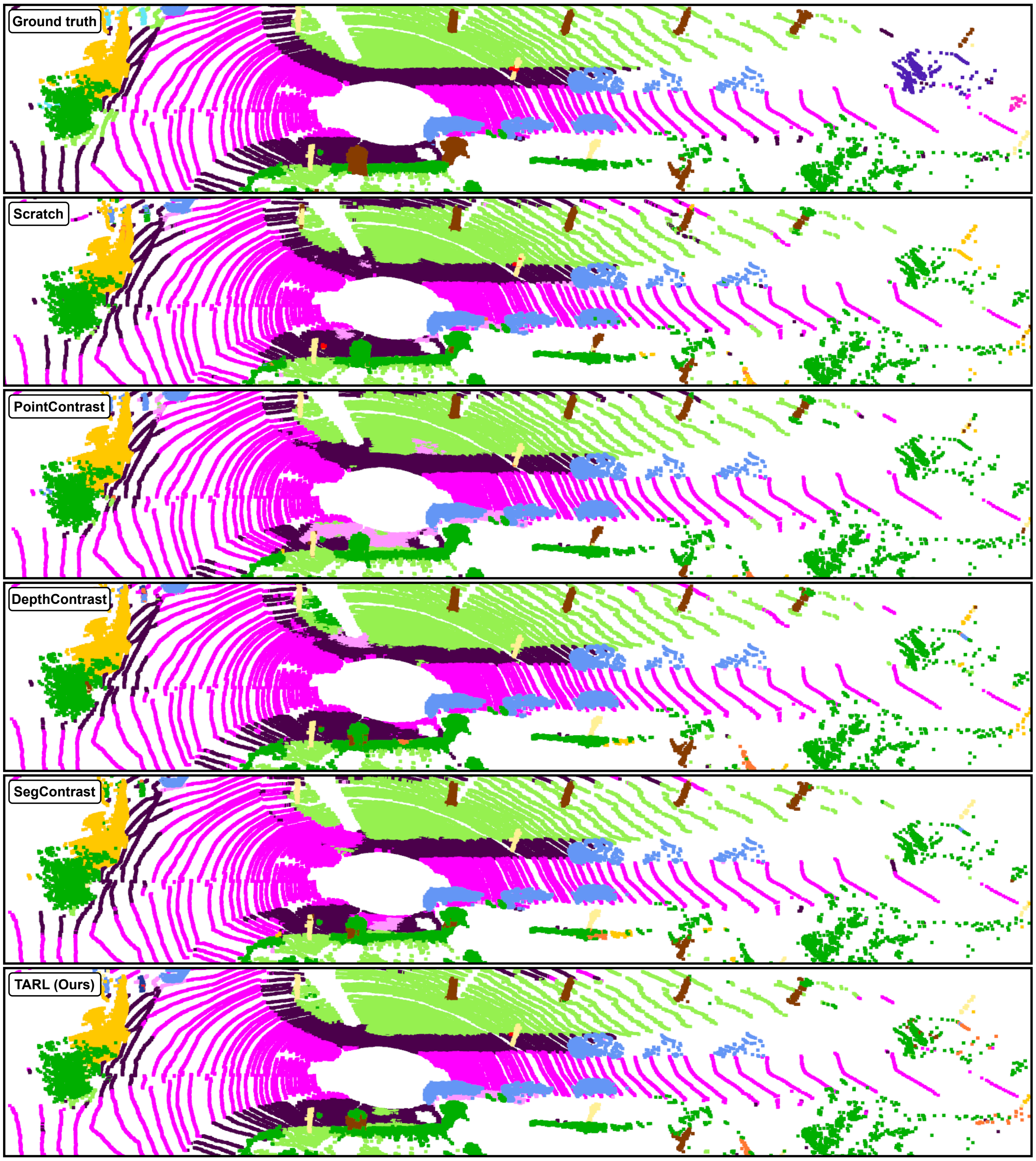


Figure 10. Qualitative comparison between the network trained from scratch and after pre-training with the different pre-training methods for semantic segmentation. We compare the baselines, PointContrast [10], DepthContrast [11], SegContrast [7] and our proposed method TARL.

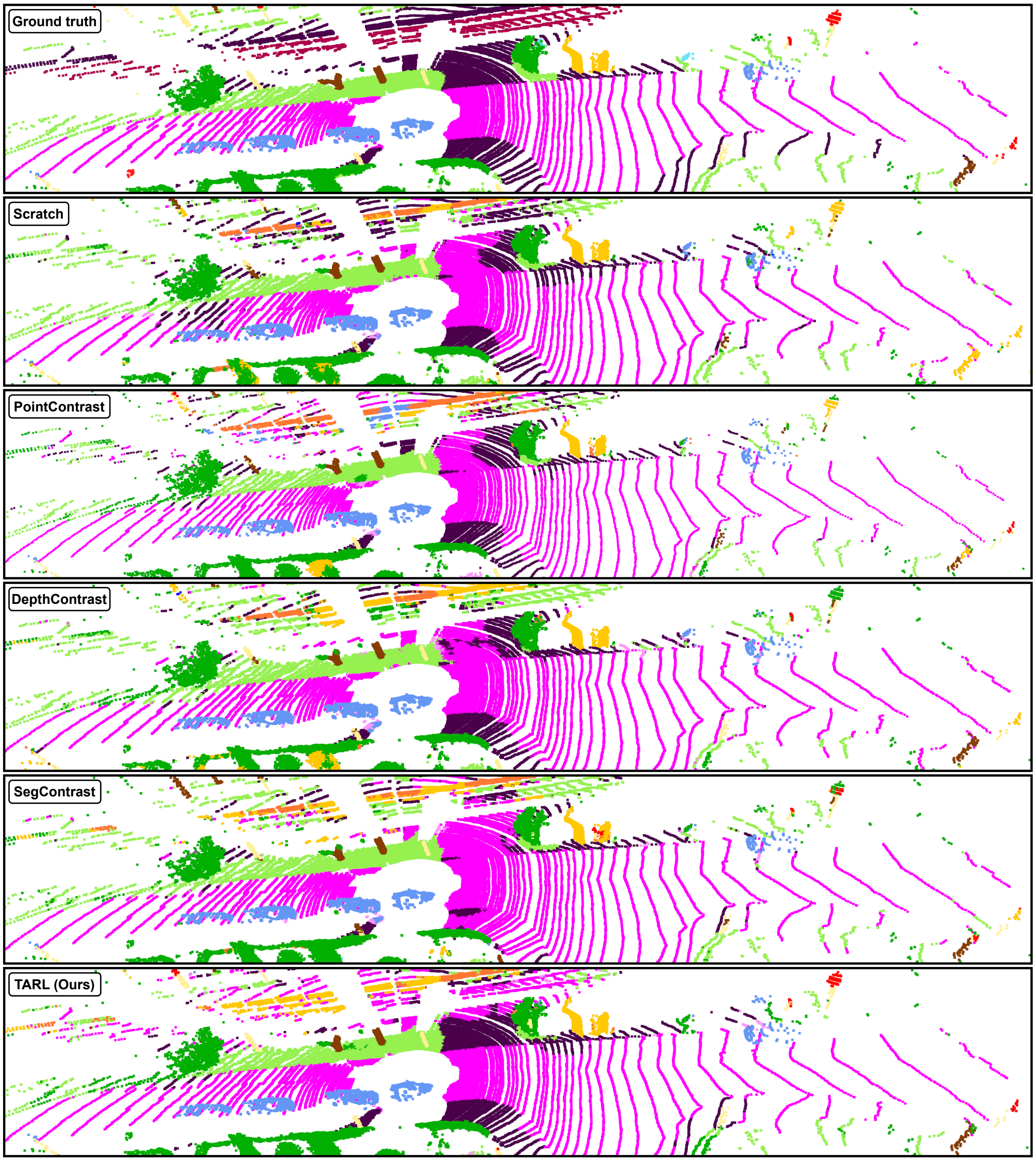


Figure 11. Qualitative comparison between the network trained from scratch and after pre-training with the different pre-training methods for semantic segmentation. We compare the baselines, PointContrast [10], DepthContrast [11], SegContrast [7] and our proposed method TARL.

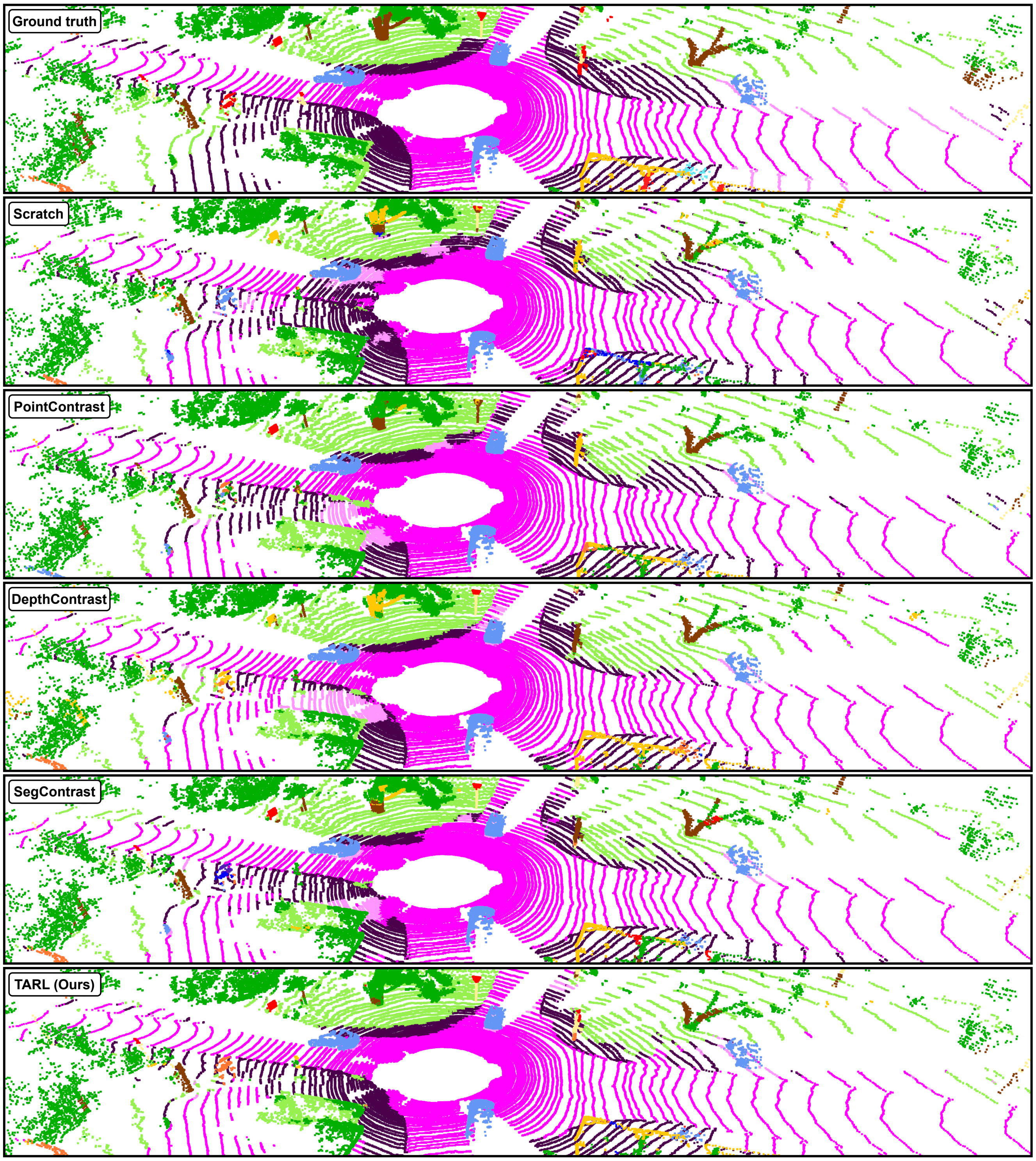


Figure 12. Qualitative comparison between the network trained from scratch and after pre-training with the different pre-training methods for semantic segmentation. We compare the baselines, PointContrast [10], DepthContrast [11], SegContrast [7] and our proposed method TARL.

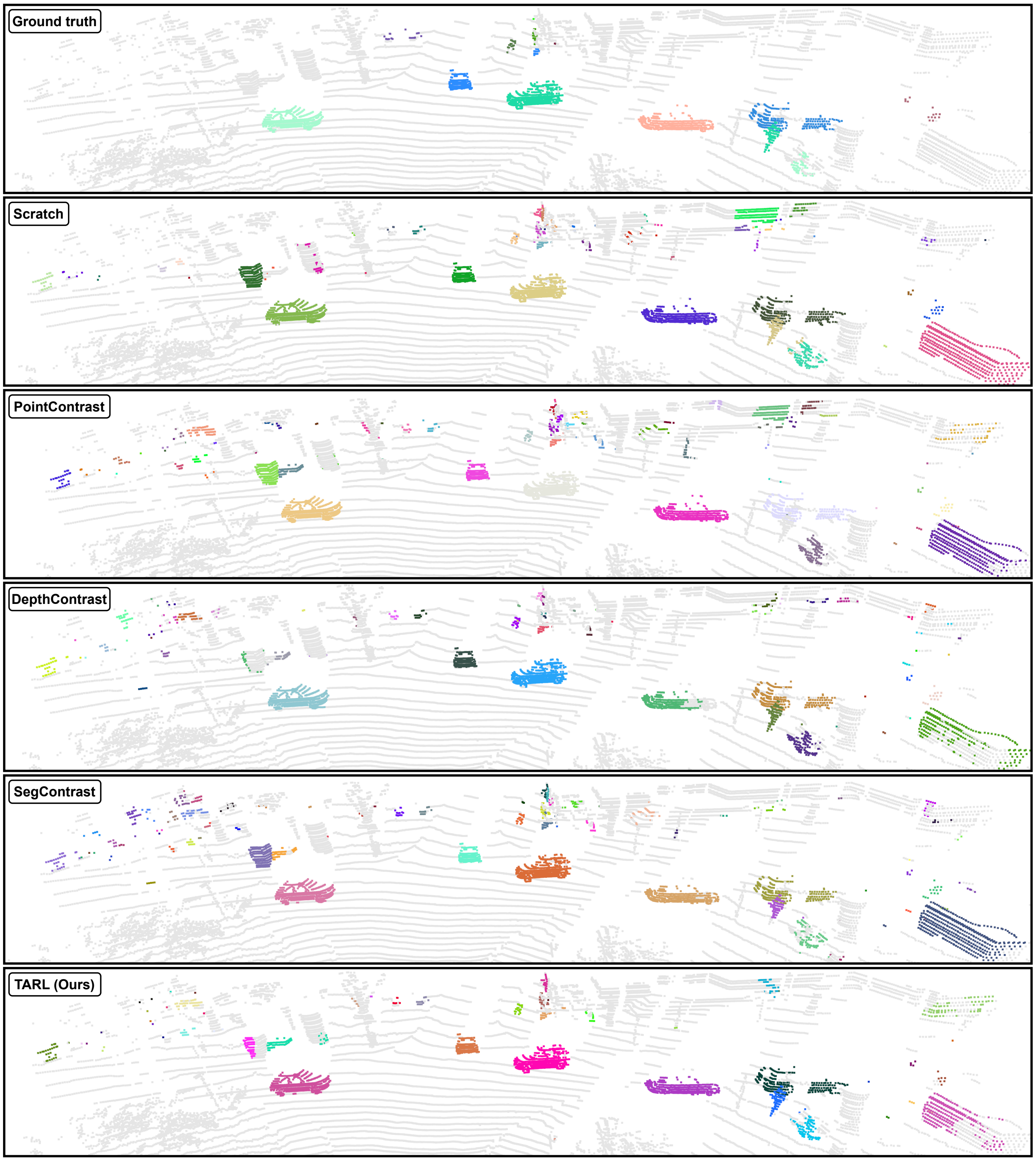


Figure 13. Qualitative comparison between the network trained from scratch and after pre-training with the different pre-training methods. The image shows the instance prediction from the panoptic segmentation. We compare the baselines, PointContrast [10], DepthContrast [11], SegContrast [7] and our proposed method TARL.

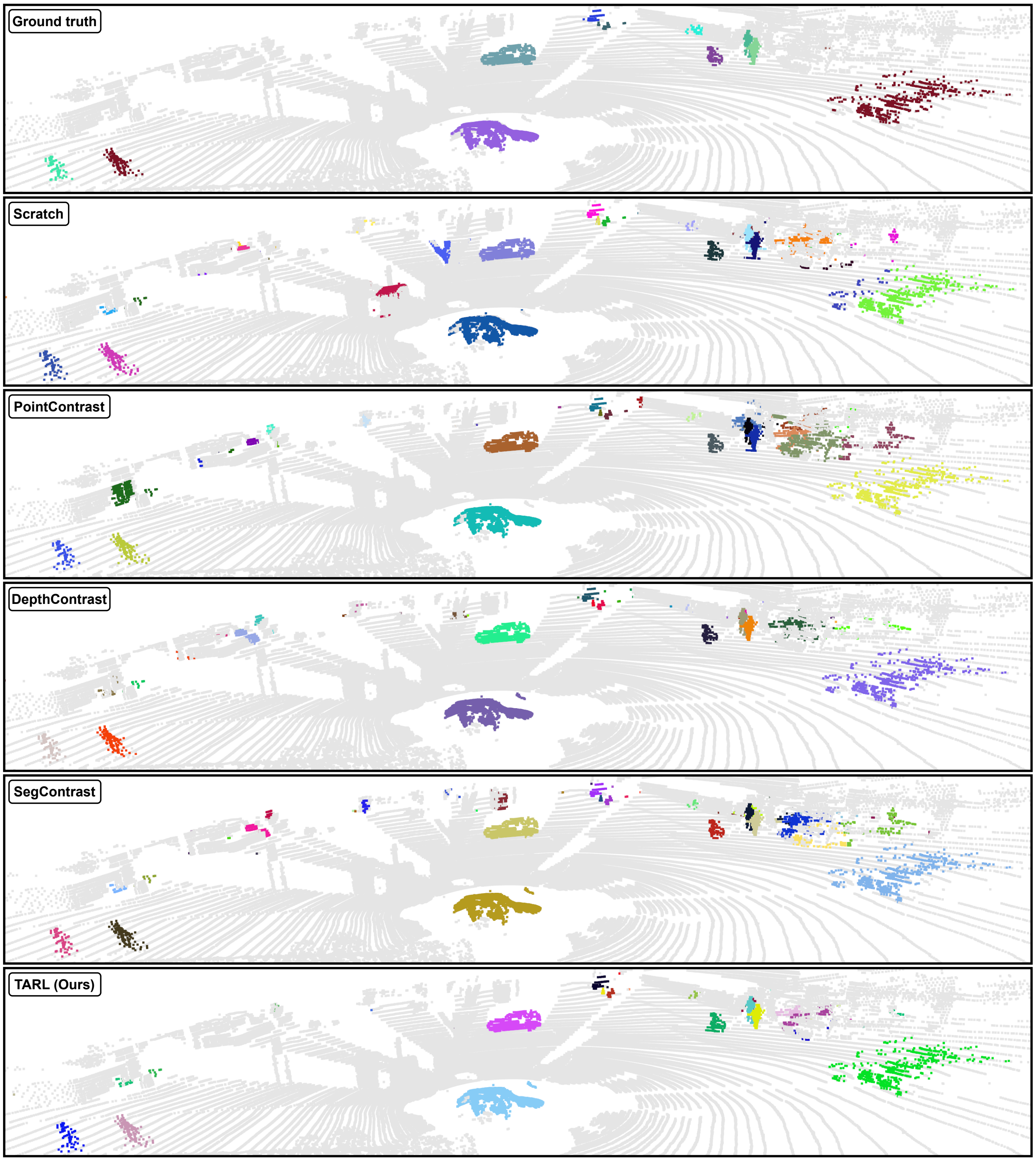


Figure 14. Qualitative comparison between the network trained from scratch and after pre-training with the different pre-training methods. The image shows the instance prediction from the panoptic segmentation. We compare the baselines, PointContrast [10], DepthContrast [11], SegContrast [7] and our proposed method TARL.

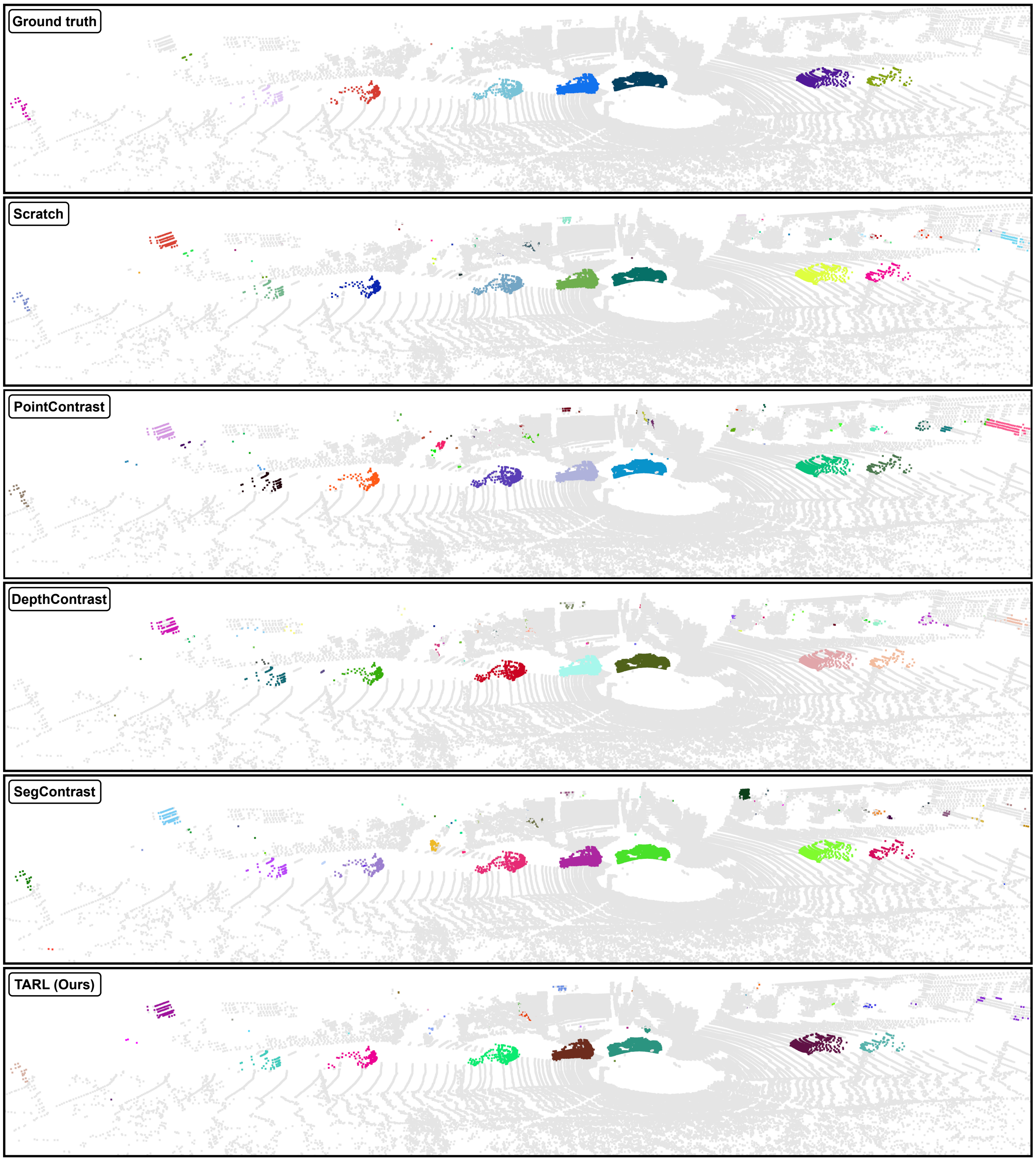


Figure 15. Qualitative comparison between the network trained from scratch and after pre-training with the different pre-training methods. The image shows the instance prediction from the panoptic segmentation. We compare the baselines, PointContrast [10], DepthContrast [11], SegContrast [7] and our proposed method TARL.

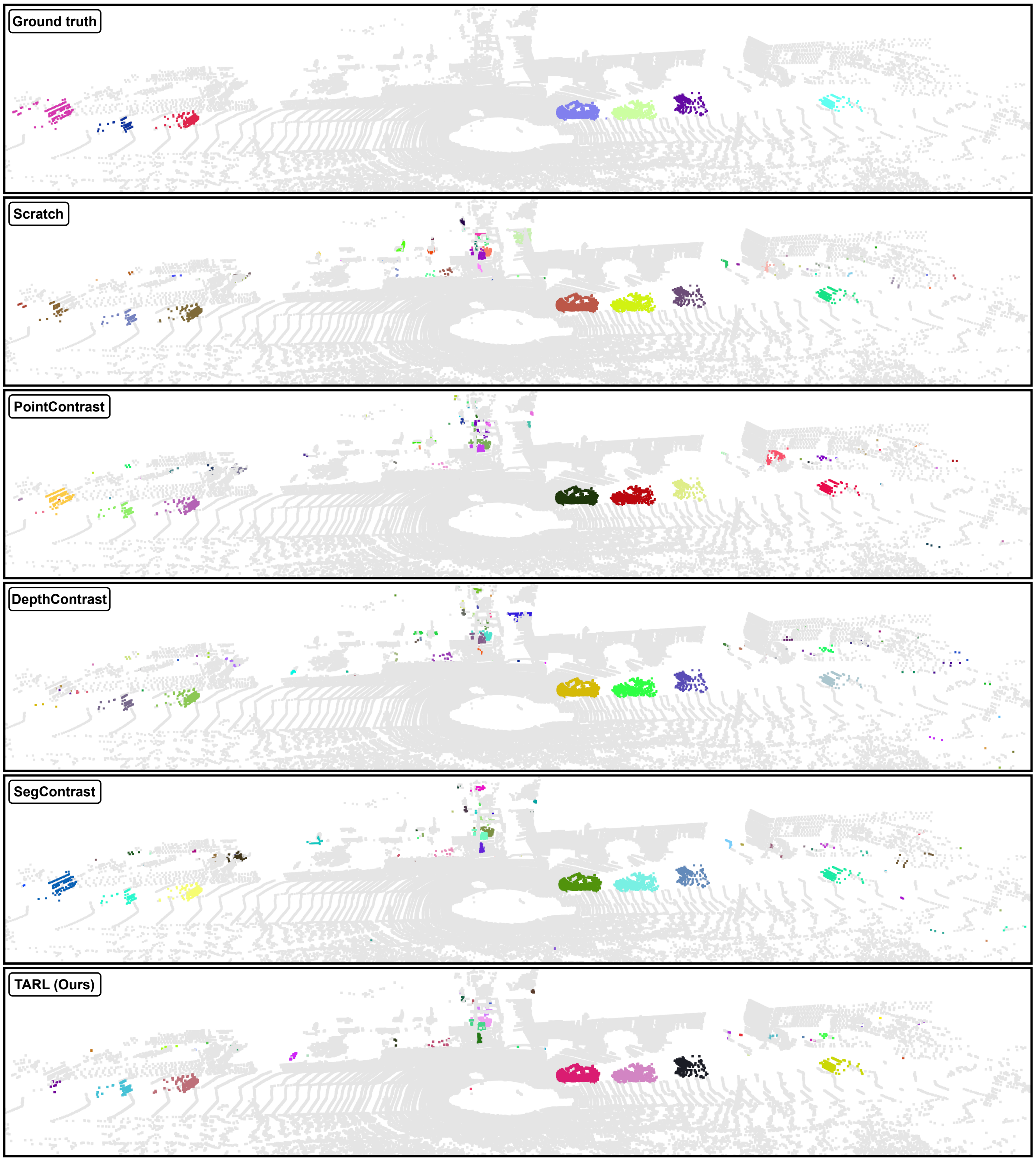


Figure 16. Qualitative comparison between the network trained from scratch and after pre-training with the different pre-training methods. The image shows the instance prediction from the panoptic segmentation. We compare the baselines, PointContrast [10], DepthContrast [11], SegContrast [7] and our proposed method TARL.

[9] Whye Kit Fong, Rohit Mohan, Juana Valeria Hurtado, Lubing Zhou, Holger Caesar, Oscar Beijbom, Abhinav Valada. Panoptic nuscenes: A large-scale benchmark for lidar panoptic segmentation and tracking. *IEEE Robotics and Automation Letters (RA-L)*, 7(2):3795–3802, 2022. 2, 8

[10] Saining Xie, Jiatao Gu, Demi Guo, Charles R. Qi, Leonidas Guibas, and Or Litany. PointContrast: Unsupervised Pre-training for 3D Point Cloud Understanding. In *Proc. of the Europ. Conf. on Computer Vision (ECCV)*, 2020. 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20

[11] Zaiwei Zhang, Rohit Girdhar, Armand Joulin, and Ishan Misra. Self-Supervised Pretraining of 3D Features on any Point-Cloud. In *Proc. of the IEEE/CVF Intl. Conf. on Computer Vision (ICCV)*, 2021. 2, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20