Supplementary Material for Spatio-temporal Self-Supervised Representation Learning for 3D Point Clouds

Siyuan Huang[1,*], Yichen Xie[2,*], Song-Chun Zhu[3,4,5], Yixin Zhu[3,4]
[1] University of California, Los Angeles [2] Shanghai Jiao Tong University
[3] Beijing Institute for General Artificial Intelligence [4] Peking University [5] Tsinghua University
https://siyuanhuang.com/STRL

1. Experimental Details

In this section, we describe additional details about network architectures (Sect. 1.1) and training methods (Sect. 1.2).

1.1. Network Architecture of STRL

As shown in Fig. 2 in the main text, the proposed spatio-temporal representation learning (STRL) framework consists of three components: online network, target network, and predictor. Online and target networks are both composed of an encoder and a projector. The encoders follow different backbone architectures with minimal modifications, detailed in Sect. 1.2. Below, we clarify the structure of the projector and predictor.

Projector The projector contains two fully connected (FC) layer with output size *hidden size* s_h and *projection size* s_p, respectively. A batch normalization and a ReLU activation layer are inserted between the two FC layers. The input of the first FC layer is the encoder's output, whose dimension differs on the basis of downstream tasks.

Predictor The predictor has a similar structure to the projector. It also consists of two FC layers with an output size of s_h and s_p, respectively. The batch normalization layer and ReLU activation layer are structured in the same fashion as the ones in the projector. The projector's output of the target network serves as the predictor's input.

We follow [2] to set the hyper-parameters $s_h = 4096$ and $s_p = 256$.

1.2. Training Details

Below, we specify the training details for each downstream tasks described in the main text, including shape classification, semantic segmentation, and indoor/outdoor object detection. For all downstream tasks, we adopt Adam optimizer [3] and LARS wrapper [8].

* indicates equal contribution.

1.2.1 Shape Classification

Backbone We adopt two practical backbones—PointNet [5] and DGCNN [7]—for shape classification task. We extract the global feature after the last max-pooling layer as the encoder output.

Linear Classification During pre-training on ShapeNet, we set the learning rate as 0.001 with a cosine decay. For PointNet, we train the model with a batch size of 48 for 50 epochs. For DGCNN, we train the model with a batch size of 32 for 100 epochs. Afterward, we fit a linear Support Vector Machine (SVM) on the representation of the ModelNet training set and evaluate it on the test set. The linear SVM has the default parameters of $C = 1.0$ and $tol = 1e-3$. For both models, we randomly select 2,048 points for each shape in both pre-training and training.

Supervised Fine-tuning In the pre-training process, we set the learning rate as 0.001 with a cosine decay. Only the DGCNN model is utilized in this task; we pre-train it on ShapeNet with a batch size of 32 for 100 epochs. During the fine-tuning process, we follow all the parameters in [7] except reducing the training epoch from 250 to 125 since our pre-trained weight helps to accelerate the supervised training. For pre-training and fine-tuning, we select 1,024 points for each shape.

1.3. Semantic Segmentation

We adopt the DGCNN network as the backbone. Like shape classification, we extract the 1024-d embedding after the last max-pooling layer as the encoder's output.

We pre-train the network on processed ScanNet dataset for 100 epochs with a batch size of 28, setting the learning rate as 0.001 with a cosine decay. We extract a key frame per 10 frames and select a window size of 10 key frames when choosing adjacent frames.

When fine-tuning the pre-trained model, we follow the settings in [7] to train the model on each area of the S3DIS dataset for 100 epochs. We use the SGD optimizer with a learning rate of 0.1 (cosine decay) and a batch size of 32. We randomly select 4,096 points for each frame during pre-training and each block during fine-tuning.

1.4. Indoor 3D Object Detection

Backbone We adopt the VoteNet model with PointNet++ backbone. By adding a max-pooling layer at the end of the backbone, we obtain a global feature of 256-d embedding with the encoder, which is further passed into the projector.

Training Parameter Same as semantic segmentation, we extract a key frame every 10 frames to process the ScanNet dataset for pre-training. Next, a window size of 10 key frames is chosen to find adjacent frames. We use a learning rate of 0.001 with a batch size of 32 for 100 epochs in the pre-training process. When fine-tuning the pre-trained model, we follow the settings in [4] to train the model for 180 epochs. The learning rate is set as 0.001 and decayed by 0.1 at the step of $80, 120, 160$. We use a batch size of 8. In both processes of pre-training and fine-tuning, we randomly select 20,000 points for each scene.

1.5. Outdoor 3D Object Detection

Backbone The PV-RCNN model is adopted in this task, together with the Sparse Convolution backbone. Additionally, we also add a max-pooling layer at the end of the backbone. A 128-d global feature is obtained as the output of the encoder.

Training Parameter We pre-train the model on KITTI raw data with a learning rate of 0.004 (cosine decay) and a batch size of 32 for 50 epochs. We sub-sample the point cloud frames per second as key frames and use a window size of 5 key frames. In the fine-tuning process, we keep the same settings as in [6]. We train the model with a learning rate of 0.01 for 80 epochs on the KITTI object detection benchmark training set. Since the input is voxelized in both pre-training and fine-tuning, we pass all points to the model without random sampling.

2. Generalizability Analysis

In Sect. 5.3 of the main text, we have described some cross-domain experiments to analyze the generalizability of pre-training between synthetic shapes and natural scenes. Here, we supplement an extra experiment to transfer the ShapeNet pre-trained DGCNN model to the 3D semantic segmentation task. We follow the setting in Sect. 5.2.2 of the main text to fine-tune the pre-trained model on one of Area 1-5 of S3DIS dataset each time and evaluate the model on Area 6. Table 1 summarizes the main results.

Consistent with the conclusion detailed in Sect. 5.3 of the main text, the DGCNN model, pre-trained on ShapeNet, achieves comparable performance set by the ScanNet pre-trained ones, which echoes our hypothesis mentioned in the main text: The model benefits from more diverse and cleaner shapes in ShapeNet to master basic spatial structures. These learned low-level knowledge help boost performance in downstream tasks, despite these downstream tasks being carried out on more complicated scenes.

Table 1: **Ablation Study: Cross-domain Generalizability**. We transfer the ShapeNet pre-trained DGCNN model to the 3D semantic segmentation task on S3DIS.

Fine-tuning Area	Method	Acc.	mIoU
Area 1 (3687 *samples*)	*from scratch*	84.57%	57.85
	STRL (ScanNet)	**85.28%**	**59.15**
	STRL (ShapeNet)	84.85%	59.11
Area 2 (4440 *samples*)	*from scratch*	70.56%	38.86
	STRL (ScanNet)	**72.37%**	**39.21**
	STRL (ShapeNet)	70.45%	38.66
Area 3 (1650 *samples*)	*from scratch*	77.68%	49.49
	STRL (ScanNet)	**79.12%**	**51.88**
	STRL (ShapeNet)	78.96%	51.03
Area 4 (3662 *samples*)	*from scratch*	73.55%	38.50
	STRL (ScanNet)	73.81%	39.28
	STRL (ShapeNet)	**74.42%**	**40.58**
Area 5 (6852 *samples*)	*from scratch*	76.85%	48.63
	STRL (ScanNet)	77.28%	49.53
	STRL (ShapeNet)	**78.53%**	**50.55**

3. Representation Robustness

We disturb the input of the ModelNet40 data and apply the linear SVM on the representations extracted by Pre-trained. The results with different disturbances: (1) Cutout: 86.91; (2) Crop: 74.59; (3) Jitter the points: 87.97; (4) Add noisy points: 82.33.

4. Alternative Framework

We design our STRL framework based on [2]. In comparison, our spatio-temporal self-supervised representation learning can also well fit other contrastive methods. Below, we present results by adopting an alternative framework—SimCLR [1]—on the linear shape classification tasks with PointNet backbone. This task is representative as it can directly reflect the efficacy of learned representations. We experimented with different batch sizes during pre-training. Table 2 tabulates main results. It reveals that a comparable performance (0.1% - 0.5% performance drop) is also reached with SimCLR framework, shown the compatibility of our proposed STRL. We also find that our method is stable on different batch sizes in the range between 32 and 1024 and achieve the best performance between 64 and 512.

Table 2: **Alternative Frameworks: SimCLR *v.s.* BYOL**. We pre-train the PointNet model separately with SimCLR and BYOL framework. The results are evaluated with a linear SVM classifier on the ModelNet40 dataset. We pre-train the model with different batch sizes.

Framework	Different batch sizes during pre-training						
	32	48	64	128	256	512	1024
BYOL	88.0%	88.1%	88.4%	88.2%	88.1%	88.4%	87.8%
SimCLR	87.9%	87.9%	88.1%	88.0%	87.6%	88.2%	87.6%

References

[1] Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. A simple framework for contrastive learning of visual representations. *arXiv preprint arXiv:2002.05709*, 2020. 2

[2] Jean-Bastien Grill, Florian Strub, Florent Altché, Corentin Tallec, Pierre H Richemond, Elena Buchatskaya, Carl Doersch, Bernardo Avila Pires, Zhaohan Daniel Guo, Mohammad Gheshlaghi Azar, et al. Bootstrap your own latent: A new approach to self-supervised learning. *arXiv preprint arXiv:2006.07733*, 2020. 1, 2

[3] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014. 1

[4] Charles R Qi, Or Litany, Kaiming He, and Leonidas J Guibas. Deep hough voting for 3d object detection in point clouds. In *Proceedings of International Conference on Computer Vision (ICCV)*, 2019. 2

[5] Charles R Qi, Hao Su, Kaichun Mo, and Leonidas J Guibas. Pointnet: Deep learning on point sets for 3d classification and segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017. 1

[6] Shaoshuai Shi, Chaoxu Guo, Li Jiang, Zhe Wang, Jianping Shi, Xiaogang Wang, and Hongsheng Li. Pv-rcnn: Point-voxel feature set abstraction for 3d object detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020. 2

[7] Yue Wang, Yongbin Sun, Ziwei Liu, Sanjay E Sarma, Michael M Bronstein, and Justin M Solomon. Dynamic graph cnn for learning on point clouds. *ACM Transactions on Graphics (TOG)*, 38(5):1–12, 2019. 1

[8] Yang You, Igor Gitman, and Boris Ginsburg. Scaling sgd batch size to 32k for imagenet training. *arXiv preprint arXiv:1708.03888*, 2017. 1