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

Siyuan Huang1,*, Yichen Xie2,*, Song-Chun Zhu3,4,5, Yixin Zhu3,4
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

Abstract

To date, various 3D scene understanding tasks still lack practical and generalizable pre-trained models, primarily due to the intricate nature of 3D scene understanding tasks and their immense variations introduced by camera views, lighting, occlusions, etc. In this paper, we tackle this challenge by introducing a spatio-temporal representation learning (STRL) framework, capable of learning from unlabeled 3D point clouds in a self-supervised fashion. Inspired by how infants learn from visual data in the wild, we explore the rich spatio-temporal cues derived from the 3D data. Specifically, STRL takes two temporally-correlated frames from a 3D point cloud sequence as the input, transforms it with the spatial data augmentation, and learns the invariant representation self-supervisedly. To corroborate the efficacy of STRL, we conduct extensive experiments on three types (synthetic, indoor, and outdoor) of datasets. Experimental results demonstrate that, compared with supervised learning methods, the learned self-supervised representation facilitates various models to attain comparable or even better performances while capable of generalizing pre-trained models to downstream tasks, including 3D shape classification, 3D object detection, and 3D semantic segmentation. Moreover, the spatio-temporal contextual cues embedded in 3D point clouds significantly improve the learned representations.

1. Introduction

Point cloud is a quintessential 3D representation for visual analysis and scene understanding. It differs from alternative 3D representations (e.g., voxel, mesh) as it is ubiquitous: Entry-level depth sensors (even on cellphones) directly produce point clouds before triangulating into meshes or converting to voxels, making it mostly applicable to 3D scene understanding tasks such as 3D shape analysis [5], 3D object detection and segmentation [58, 10]. Despite its omnipresence in 3D representation, however, annotating 3D point cloud data is proven to be much more difficult compared with labeling conventional 2D image data; this obstacle precludes its potentials in 3D visual tasks. As such, properly leveraging the colossal amount of unlabeled 3D point cloud data is a sine qua non for the success of large-scale 3D visual analysis and scene understanding.

Meanwhile, self-supervised learning from unlabeled images [11, 45, 24, 22, 6, 19, 7] and videos [54, 80, 34, 51] becomes a nascent direction in representation learning with great potential in downstream tasks.

In this paper, we fill in the absence by exploiting self-supervised representation learning for 3D point clouds to address a long-standing problem in our community—the supervised training struggles at producing practical and generalizable pre-trained models due to the supervision-starved nature of the 3D data. Specifically, we consider the following three principles in model design and learning:

Simplicity Although self-supervised learning approaches for 3D point clouds exist, they rely exclusively on spatial analysis by reconstructing the 3D point clouds [1, 75, 53, 20]. This static perspective of self-supervised learning is designed explicitly with complex operations, architectures, or losses, making it difficult to train and generalize to diversified downstream tasks. We believe such intricate designs are artificially introduced and unnecessary, and could be diminished or eliminated by complementing the missing temporal contextual cues, akin to how infants may understand this world [18, 57].

* indicates equal contribution.
Invariance  Learning data invariance via data augmentation and contrasting has shown promising results on images and videos [22, 6, 19]. A natural question arises: How could we introduce and leverage the invariance in 3D point clouds for self-supervised learning?

Generalizability  Prior literature [1, 75, 53, 20] has only verified the self-supervisedly learned representations in shape classification on synthetic datasets [5], which possesses dramatically different characteristics compared with the 3D data of natural indoor [58, 10] or outdoor [16] environments, thus failed to demonstrate sufficient generalizability to higher-level tasks (e.g., 3D object detection).

To adhere to the above principles and tackle the challenges introduced thereby, we devise a spatio-temporal representation learning (STRL) framework to learn from unlabeled 3D point clouds. Of note, STRL is remarkably simple by learning only from the positive pairs, inspired by the BYOL [19]. Specifically, STRL uses two neural networks, referred to as online and target networks, that interact and learn from each other. By augmenting one input, we train the online network to predict the target network representation of another temporally correlated input, obtained by a separate augmentation process.

To learn the invariant representation [12, 68], we explore the inextricably spatio-temporal contextual cues embedded in 3D point clouds. In our approach, the online network’s and target network’s inputs are temporally correlated, sampled from a point cloud sequence. Specifically, for natural images/videos, we sample two frames with a natural viewpoint change in depth sequences as the input pair. For synthetic data like 3D shape, we augment the original input by rotation, translation, and scaling to emulate the viewpoint change. The temporal difference between the inputs avails models of capturing the randomness and invariance across different viewpoints. Additional spatial augmentations further facilitate the model to learn 3D spatial structures of point clouds; see examples in Fig. 1 and Sect. 3.

To generalize the learned representation, we adopt several practical networks as backbone models. By pre-training on large datasets, we verify that the learned representations can be readily adapted to downstream tasks directly or with additional feature fine-tuning. We also demonstrate that the learned representation can be generalized to distant domains, different from the pre-trained domains; e.g., the representation learned from ScanNet [10] can be generalized to shape classification tasks on ShapeNet [5] and 3D object detection task on SUN RGB-D [58].

We conduct extensive experiments on various domains and test the performance by applying the pre-trained representation to downstream tasks, including 3D shape classification, 3D object detection, and 3D semantic segmentation. Next, we summarize our main findings.

Our method outperforms prior arts.  By pre-training with STRL and applying the learned models to downstream tasks, it (i) outperforms the state-of-the-art unsupervised methods on ModelNet40 [71] and reaches 90.9% 3D shape classification accuracy with linear evaluation, (ii) shows significant improvements in semi-supervised learning with limited data, and (iii) boosts the downstream tasks by transferring the pre-trained models, e.g., it improves 3D object detection on SUN RGB-D [58] and KITTI dataset [16], and 3D semantic segmentation on S3DIS [2] via fine-tuning.

Simple learning strategy leads to the satisfying performance of learned 3D representation.  Through the ablative study in Tables 7 and 8, we observe that STRL can learn the self-supervised representations with simple augmentations; it robustly achieves a satisfying accuracy (about 85%) on ModelNet40 linear classification, which echoes recent findings [46] that simply predicting the 3D orientation helps learn good representation for 3D point clouds.

The spatio-temporal cues boost the performance of learned representation.  Relying on spatial or temporal augmentation alone only yield relatively low performance as shown in Tables 7 and 8. In contrast, we achieve an improvement of 3% accuracy by learning the invariant representations combining both spatial and temporal cues.

Pre-training on synthetic 3D shapes is indeed helpful for real-world applications.  Recent study [73] shows the representation learned from ShapeNet is not well-generalized to the downstream tasks. Instead, we report an opposite observation in Table 6, showing the representation pre-trained on ShapeNet can achieve comparable and even better performance while applying to downstream tasks that tackle complex data obtained in the physical world.

2. Related Work

Representation Learning on Point Clouds  Unlike conventional representations of structured data (e.g., images), point clouds are unordered sets of vectors. This unique nature poses extra challenges to the learning of representations. Although deep learning methods on unordered sets [66, 77, 42] could be applied to point clouds [52, 77], these approaches do not leverage spatial structures.

Taking spatial structures into consideration, modern approaches like PointNet [48] directly feed raw point clouds into neural networks; these networks ought to be permutation invariant as point clouds are unordered sets. PointNet achieves this goal by using the max-pooling operation to form a single feature vector representing the global context from a set of points. Since then, researches have proposed alternative representation learning methods with hierarchy [49, 33, 13], convolution-based structure [25, 74, 39, 59, 78, 70, 61], or graph-based information aggregation [13, 64, 55, 67]. Operating directly on raw point clouds, these neural networks naturally provide per-point embedding, particularly effective for point-based tasks. Since the proposed STRL is flexible and compatible with various neural models serving as the backbone, our design of STRL leverages the efficacy introduced by per-point embedding.
Figure 2: Illustration of our self-supervised learning framework. Given two spatio-temporal correlated 3D point clouds, the online network predicts the target network’s representation via a predictor. Parameters of the target network are updated by the online network’s moving average.

3. Spatio-temporal Representation Learning

We devise the proposed spatio-temporal representation learning (STRL) based on BYOL [19] and extend its simplicity to the learning of 3D point cloud representation. Fig. 2 illustrates the proposed method.

3.1. Building Temporal Sequence of Point Clouds

To learn a simple, invariant, and generalizable representation for 3D point clouds, we formulate the representation learning as training with sequences of potentially partial and cluttered 3D point clouds of objects or scenes. Given a sequence of potentially non-uniformly sampled time steps, we denote the corresponding point cloud sequence as \( \mathcal{P} = \{p_t\}_{t=1}^T \). We devise two approaches to generating the training point cloud sequences to handle various data sources.

**Natural Sequence**
Natural sequences refer to the data sequences captured by RGB-D sensors, wherein each depth image \( I_t \) is a projected view of the scene. Given the camera pose (extrinsic parameters) at each time step \( c_t \), we back-project depth images with intrinsic parameters \( c_{in} \) and obtain a sequence of point clouds \( \{p_t\} \) in world coordinate:

\[
p_t = \text{Backproj}(I_t, c_{in}, c_{ex}), \quad t = 1, \cdots, T. \tag{1}
\]

**Synthetic Sequence**
Static point clouds are intrinsically spatial, missing the crucial temporal dimension compared to natural sequences. Given a point cloud \( p_0 \), we solve this problem by generating a synthetic sequence. Specifically, we consecutively rotate, translate, and scale the original point cloud to construct a sequence of point clouds \( \{p_t\} \):

\[
p_t = R_t(p_{t-1}), \quad t = 1, \cdots, T, \tag{2}
\]

where \( t \) is the index of transformations, and \( R_t \) the sampled transformation, emulating temporal view changes.

3.2. Representation Learning

We design STRL to unsupervisedly learn the representations through the interactions of two networks: the online network and target network. Here, the essence of self-supervised learning is to train the online network to accurately predict the target network’s representation.

Specifically, the online network parameterized by \( \theta \) consists of two components: a backbone encoder \( e_\theta \) and a feature projector \( f_\theta \). Similarly, the target network parameterized by \( \phi \) has a backbone encoder \( e_\phi \) and feature projector \( f_\phi \). In addition, a predictor \( r \) with parameters regresses the target presentation: The target network serves as regression targets to train the online network, and its parameters \( \phi \) are an exponential moving average of the online parameters \( \theta \):

\[
\phi \leftarrow \tau \phi + (1 - \tau) \theta, \tag{3}
\]

where \( \tau \in [0, 1] \) is the decay rate of the moving average.
Given a sequence of point clouds \( P \), we sample two frames of point clouds \( p^u, p^v \in P \) by a temporal sampler \( T \). With a set of spatial augmentations \( A \) (see details in Sect. 4), STRL generates two inputs \( x^u = a^u(p^v) \) and \( x^v = a^v(p^v) \), where \( a^u, a^v \in A \). For each input, the online network and target network generate \( z_\theta = f_\theta(e_\theta(x^u)), z_\phi = f_\phi(e_\phi(x^v)) \), respectively. With the additional predictor \( r \), the goal of STRL is to minimize the mean squared error between the normalized predictions and target projections:

\[
L_{u \rightarrow v} = \frac{1}{\|r(z_\theta)\|_2^2} \cdot \frac{z_\theta - \langle r(z_\theta), z_\theta \rangle}{\|z_\theta\|_2^2} = 2 - 2 \cdot \frac{\|r(z_\theta)\|_2^2 \cdot \|z_\theta\|_2^2}{\|r(z_\theta)\|_2^2} \quad (4)
\]

Finally, we symmetrize the loss in Eq. (4) to compute \( L_{v \rightarrow u} \) by separately feeding \( x^v \) to the online network and \( x^u \) to the target network. The total loss is defined as:

\[
L_{\text{total}} = L_{u \rightarrow v} + L_{v \rightarrow u}. \quad (5)
\]

Within each training step, only the parameters of the online network and predictor are updated. The target network’s parameters are updated after each training step by Eq. (3). Similar to [22, 19], we only keep the backbone encoder of the online network \( e_\theta \) at the end of the training as the learned model. Algorithm 1 details the proposed STRL.

### 4. Implementation Details

**Synthetic Sequence Generation** We sample the combination of following transformations to construct the function \( R() \) in Eq. (2); see an illustration in Fig. 3b:

- Random rotation. For each axis, we draw random angles within \( 15^\circ \) and rotate around it.
- Random translation. We translate the point cloud globally within 10% of the point cloud dimension.
- Random scaling. We scale the point cloud with a factor \( s \in [0.8, 1.25] \).

To further increase the randomness, each transformation is sampled and applied with a probability of 0.5.

**Spatial Augmentation** The spatial augmentation transforms the input by changing the point cloud’s local geometry, which helps STRL to learn a better spatial structure representation of point clouds. Specifically, we apply the following operations, similar to the image data augmentation; see an illustration in Fig. 3a:

- Random cropping. A random 3D cuboid patch is cropped with a volume uniformly sampled between 60% and 100% of the original point cloud. The aspect ratio is controlled within \([0.75, 1.33]\).
- Random cutout. A random 3D cuboid is cut out. Each dimension of the 3D cuboid is within \([0.1, 0.4]\) of the original dimension.
- Random jittering. Each point’s 3D locations are shifted by a uniformly random offset within \([0, 0.05]\).
- Random drop-out. We randomly drop out 3D points by a drop-out ratio within \([0, 0.7]\).

**Algorithm 1: STRL of 3D point clouds**

**Input:**
1. \( \{P\} \): a set of 3D point cloud sequences;
2. \( T, A \): temporal sampler and spatial augmentations;
3. \( e_\theta, f_\theta \): online encoder and projector with parameter \( \theta \);
4. \( e_\phi, f_\phi \): target encoder and projector with parameter \( \phi \);
5. \( r \): predictor;
6. \( K \): number of optimization steps;
7. \( N \): batch size.

**Output:** online encoder \( e_\theta \).

**for** \( k = 1 \) to \( K \) **do**

\[
\begin{align*}
&\text{/* sample batches of temporal-correlated point clouds */} \\
&B \leftarrow \{p^i, p^i \in T(\{P\})\}_{i=1}^N \\
&\text{/* sample spatial augmentations */} \\
&\text{/* generate inputs */} \\
&x^u = a^u(p^v), x^v = a^v(p^v) \\
&\text{/* project */} \\
&z_\theta = f_\theta(g_\theta(x^u)), z_\phi = f_\phi(g_\phi(x^v)) \\
&\text{/* compute loss */} \\
&L_{u \rightarrow v} = -2 \cdot \frac{\langle r(z_\theta), z_\theta \rangle}{\|r(z_\theta)\|_2^2 \cdot \|z_\theta\|_2^2} \\
&\text{/* compute total & symmetric loss */} \\
&L_{\text{total}} = L_{u \rightarrow v} + L_{v \rightarrow u} \\
&\text{/* update online network & predictor */} \\
&\theta, r = \text{optimize}(\theta, r, L_{\text{total}}) \\
&\text{/* update target network */} \\
&\phi = \tau \phi + (1 - \tau) \theta \\
\end{align*}
\]

- Down-sampling. We down-sample point clouds based on the encoder’s input dimension by randomly picking the necessary amount of 3D points.
- Normalization. We normalize the point cloud to fit a unit sphere while training on synthetic data [5].

Among these augmentations, cropping and cutout introduce more evident changes to the point clouds’ spatial structures. As such, we apply them with a probability of 0.5.

**Training** We use the LARS optimizer [76] with a cosine decay learning rate schedule [40], with a warm-up period of 10 epochs but without restarts. For the target network, the exponential moving average parameter starts with \( \tau_{\text{start}} = 0.996 \) and is gradually increased to 1 during the training. Specifically, we set \( \tau = 1 - (1 - \tau_{\text{start}}) \cdot (\cos(\pi k/K) + 1)/2 \) with \( k \) being the current training step and \( K \) the maximum number of training steps.

STRL is favorable and generalizable to different backbone encoders; see details about the encoder structure for each specific experiment in Sect. 5. The projector and predictor are implemented as multi-layer perceptions (MLPs) with activation [43] and batch normalization [29]; see the supplementary materials for detailed network structures.

We use a batch size ranging from 64 to 256 split over 8 TITAN RTX GPUs for most of the pre-trained models.
5. **Experiment**

We start by introducing how to pre-train STRL on various data sources in Sect. 5.1. Next, we evaluate these pre-trained models on various downstream tasks in Sect. 5.2. At length, in Sect. 5.3, we analyze the effects of different modules and parameters in our model, with additional analytic experiments and discussions of open problems.

### 5.1. Pre-training

To recap, as detailed in Sect. 3.1, we build the sequences of point clouds and perform the pre-training of STRL to learn the spatio-temporal invariance of point cloud data. For synthetic shapes and natural indoor/outdoor scenes, we generate temporal sequences of point clouds and sample input pairs using different strategies detailed below.

#### 5.1.1 Synthetic Shapes

**ShapeNet** We learn the self-supervised representation model from the ShapeNet [5] dataset. It consists of 57,448 synthetic objects from 55 categories. We pre-process the point clouds following Yang et al. [75]. By augmenting each point cloud into two different views with temporal transformations defined in Eq. (2), we generate two temporal-corrected point clouds. The spatial augmentations are further applied to produce the pair of point clouds as the input.

#### 5.1.2 Natural Indoor and Outdoor Scenes

We also learn the self-supervised representation models from natural indoor and outdoor scenes, in which sequences of point clouds are readily available. Using RGB-D sensors, sequences of depth images can be captured by scanning over different camera poses. Since most scenes are captured smoothly, we learn the temporal invariance from the temporal correlations between the adjacent frames.

**ScanNet** For indoor scenes, we pre-train on the ScanNet dataset [10]. It consists of 1,513 reconstructed meshes for 707 unique scenes. In experiments, we find that increasing the frame-sampling frequency only makes a limited contribution to the performance. Hence, we sub-sample the raw depth sequences every 100 frames as the keyframes for each scene, resulting in 1,513 sequences and roughly 25 thousand frames in total. During pre-training, we generate fixed-length sliding windows based on the keyframes of each sequence and sample two random frames within each window. By back projecting the two frames with Eq. (1), we generate point clouds in the world coordinate. We use the camera position to translate the two point clouds into the same world coordinate; the camera center of the first frame is the origin.

**KITTI** For outdoor scenes, we pre-train on the KITTI dataset [15]. It includes 100+ sequences divided into 6 categories. For each scene, images and point clouds are recorded at roughly 10 FPS. We only use the sequences of point clouds captured by the Velodyne lidar sensor. On average, each frame has about 120,000 points. Similar to ScanNet, we sub-sample the keyframes and sample frame pairs within sliding windows as training pairs.

For pre-training on natural scenes, we further enhance the data diversity by applying synthetic temporal transformations in Eq. (2) to the two point clouds. At length, the spatial data augmentation is applied to both point clouds.

### 5.2. Downstream Tasks

For each downstream task below, we present the model structures, experimental settings, and results. Please refer to the supplementary materials for training details.

#### 5.2.1 Shape Understanding

We adopt the protocols presented in prior work [1, 53, 69, 75] to evaluate the shape understanding capability of our pre-trained model using the ModelNet40 [71] benchmark. It contains 12,331 objects (9,843 for training and 2,468 for testing) from 40 categories. We pre-process the data following Qi et al. [48], such that each shape is sampled to 10,000 points in unit space.

As detailed in Sect. 5.1, we pre-train the backbone models on ShapeNet dataset. We measure the learned representations using the following evaluation metrics.


**Linear Evaluation for Shape Classification** To classify 3D shapes, we append a linear Support Vector Machine (SVM) on top of the encoded global feature vectors. Following Sauder et al. [53], these global features are constructed by extracting the activation after the last pooling layer. Our STRL is flexible to work with various backbones; we select two practical ones—PointNet [48] and DGCNN [67]. The SVM is trained with the extracted global features from the training sets of ModelNet40 datasets. We randomly sample 2,048 points from each shape during both pre-training and SVM training.

Table 1 tabulates the classification results on test sets. The proposed STRL outperforms all the state-of-the-art unsupervised and self-supervised methods on ModelNet40.

Table 1: **Comparisons of the linear evaluation for shape classification on ModelNet40.** A linear classifier is trained on the representations learned by different self-supervised approaches on the ShapeNet dataset.

<table>
<thead>
<tr>
<th>Method</th>
<th>ModelNet40</th>
</tr>
</thead>
<tbody>
<tr>
<td>3D-GAN [69]</td>
<td>83.3%</td>
</tr>
<tr>
<td>Latent-GAN [1]</td>
<td>85.7%</td>
</tr>
<tr>
<td>SO-Net [38]</td>
<td>87.3%</td>
</tr>
<tr>
<td>FoldingNet [75]</td>
<td>88.4%</td>
</tr>
<tr>
<td>MRTNet [21]</td>
<td>86.4%</td>
</tr>
<tr>
<td>3D-PointCapsNet [75]</td>
<td>88.9%</td>
</tr>
<tr>
<td>MAP-VAE [75]</td>
<td>88.4%</td>
</tr>
<tr>
<td>Sauder et al. + PointNet [53]</td>
<td>87.3%</td>
</tr>
<tr>
<td>Sauder et al. + DGCNN [53]</td>
<td>90.6%</td>
</tr>
<tr>
<td>Poursaeed et al. + PointNet [46]</td>
<td>88.6%</td>
</tr>
<tr>
<td>Poursaeed et al. + DGCNN [46]</td>
<td>90.7%</td>
</tr>
<tr>
<td>STRL + PointNet (ours)</td>
<td>88.3%</td>
</tr>
<tr>
<td>STRL + DGCNN (ours)</td>
<td><strong>90.9%</strong></td>
</tr>
</tbody>
</table>

**Supervised Fine-tuning for Shape Classification** We also evaluate the learned representation models by supervised fine-tuning. The pre-trained model serves as the point cloud encoder’s initial weight, and we fine-tune the DGCNN network given the labels on the ModelNet40 dataset. Our STRL leads to a marked performance improvement of up to 0.9% on the final classification accuracy; see Table 2a. This improvement is more significant than previous methods; it even attains a comparable performance set by the state-of-the-art supervised learning method [78].

Furthermore, we show that our pre-trained models can significantly boost the classification performance in semi-supervised learning where limited labeled training data is provided. Specifically, we randomly sample the training data with different proportions and ensure at least one sample for each category is selected. Next, we fine-tune the pre-trained model on these limited samples with supervision and evaluate its performance on full test sets. Table 2b summarizes the results measured by accuracy. It shows that the proposed model obtains 2.1% and 1.6% performance gain when 1% and 20% of the training samples are available; our self-supervised models would better facilitate downstream tasks when fewer training samples are available.

Table 2: **Shape classification fine-tuned on ModelNet40.** The self-supervised pre-trained model serves as the initial weight for supervised learning methods.

(a) Fine-tuned on Full Training Set

<table>
<thead>
<tr>
<th>Category</th>
<th>Method</th>
<th>Accuracy</th>
</tr>
</thead>
<tbody>
<tr>
<td>Supervised</td>
<td>PointNet [48]</td>
<td>89.2%</td>
</tr>
<tr>
<td></td>
<td>PointNet++ [49]</td>
<td>90.7%</td>
</tr>
<tr>
<td></td>
<td>DGCNN [67]</td>
<td>92.2%</td>
</tr>
<tr>
<td></td>
<td>ShellNet [78]</td>
<td>93.1%</td>
</tr>
<tr>
<td>Self-supervised</td>
<td>Sauder et al. + DGCNN [53]</td>
<td>92.4%</td>
</tr>
<tr>
<td></td>
<td>STRL + DGCNN (ours)</td>
<td><strong>93.1%</strong></td>
</tr>
</tbody>
</table>

(b) Fine-tuned on Few Training Samples

<table>
<thead>
<tr>
<th>Method</th>
<th>1%</th>
<th>5%</th>
<th>10%</th>
<th>20%</th>
</tr>
</thead>
<tbody>
<tr>
<td>DGCNN</td>
<td>60.5%</td>
<td>82.7%</td>
<td>86.5%</td>
<td>89.7%</td>
</tr>
<tr>
<td>STRL + DGCNN (ours)</td>
<td>90.9%</td>
<td>92.4%</td>
<td>92.2%</td>
<td>93.1%</td>
</tr>
</tbody>
</table>

Figure 4: **Visualization of learned features.** We visualize the extracted features for each sample in ModelNet10 test set using t-SNE. Both models are pre-trained on ShapeNet.

**Embedding Visualization** We visualize the learned features of PointNet and DGCNN model with our self-supervised method in Fig. 4; it displays the embedding for samples of different categories in the ModelNet10 test set. t-SNE [41] is adopted for dimension reduction. We observe that both pre-trained models well separate most samples based on categories, except dressers and night stands; they usually look similar and are difficult to distinguish.

### 5.2.2 Indoor Scene Understanding

Our proposed STRL learns representations based on view transformation, suitable for both synthetic shapes and natural scenes. Consequently, unlike prior work that primarily performs transfer learning to shape understanding, our method can also boost the indoor/outdoor scene understanding tasks. We start with the indoor scene understanding in this section. We first pre-train our STRL self-supervisedly on the ScanNet dataset as described in Sect. 5.1. Next, we evaluate the performance of 3D object detection and semantic segmentation through fine-tuning with labels.

**3D Object Detection** 3D object detection requires the model to predict the 3D bounding boxes with their object categories based on the input 3D point cloud. After pre-training, we fine-tune and evaluate the model on the SUN
RGB-D [58] dataset. It contains 10,335 single-view RGB-D images, split into 5,285 training samples and 5,050 validation samples. Objects are annotated with 3D bounding boxes and category labels. We conduct this experiment with VoteNet [47], which is a widely-used model with 3D point clouds as input. During pre-training, we slightly modify its PointNet++ [49] backbone by adding a max-pooling layer at the end to obtain the global features. Table 3 summarizes the results. The pre-training improves the detection performance by 1.2 mAP compared against training VoteNet from scratch, demonstrating that the representation learned from a large dataset, i.e., ScanNet, can be successfully transferred to a different dataset and improve the performances of high-level tasks via fine-tuning. It also outperforms the state-of-the-art self-supervised learning method [73] by 0.7 mAP.¹

Table 3: 3D object detection fine-tuned on SUN RGB-D

<table>
<thead>
<tr>
<th>Model</th>
<th>Method</th>
<th>Input</th>
<th>mAP@0.25 IoU</th>
</tr>
</thead>
<tbody>
<tr>
<td>VoteNet</td>
<td>from scratch</td>
<td>Geo+Height Geo</td>
<td>57.7</td>
</tr>
<tr>
<td>SR-UNet [9]</td>
<td>PointContrast [73]</td>
<td>Geo Geo</td>
<td>57.5</td>
</tr>
<tr>
<td>VoteNet</td>
<td>STRL (ours)</td>
<td>Geo</td>
<td>58.2</td>
</tr>
</tbody>
</table>

3D Semantic Segmentation We transfer the pre-trained model to the 3D semantic segmentation task on the Stanford Large-Scale 3D Indoor Spaces (S3DIS) [2] dataset. This dataset contains 3D point clouds scanned from 272 rooms in 6 indoor areas, with each point annotated into 13 categories. We follow the setting in Qi et al. [48] and Wang et al. [67] and split each room into 1m × 1m blocks. Different from them, we use 4,096 points with only geometric features (XYZ coordinates) as the model input. In this experiment, the DGCNN network is firstly pre-trained on ScanNet with STRL. Here, we focus on semi-supervised learning with only limited labeled data. As such, we fine-tune the pre-trained model on one area in Area 1-5 each time and test the model on Area 6. As shown in Table 4, the pre-trained models consistently outperform the models trained from scratch, especially with a small training set.

5.2.3 Outdoor Scene Understanding

Compared with indoor scenes, point clouds captured in outdoor environments are much sparser due to the long-range nature of Lidar sensors, posing additional challenges. In this section, we evaluate the performance of the proposed STRL by transferring the learned visual representations to the 3D object detection task for outdoor scenes.

As described in Sect. 5.1, we pre-train the model on the KITTI dataset with PV-RCNN [56]—the state-of-the-art model for 3D object detection. Similar to VoteNet, we modify the backbone network of PV-RCNN for pre-training by adding a max-pooling layer to obtain the global features.

ⁱThe model pre-trained on ShapeNet achieves better results as 59.2 mAP, which is analyzed and explained in Sect. 5.3

Table 4: 3D semantic segmentation fine-tuned on S3DIS.

<table>
<thead>
<tr>
<th>Fine-tuning Area</th>
<th>Method</th>
<th>Acc.</th>
<th>mIoU</th>
</tr>
</thead>
<tbody>
<tr>
<td>Area 1 (3687 samples)</td>
<td>from scratch</td>
<td>84.57%</td>
<td>57.85</td>
</tr>
<tr>
<td></td>
<td>STRL</td>
<td>85.28%</td>
<td>59.15</td>
</tr>
<tr>
<td>Area 2 (4440 samples)</td>
<td>from scratch</td>
<td>70.56%</td>
<td>38.86</td>
</tr>
<tr>
<td></td>
<td>STRL</td>
<td>72.37%</td>
<td>39.21</td>
</tr>
<tr>
<td>Area 3 (1650 samples)</td>
<td>from scratch</td>
<td>77.68%</td>
<td>49.49</td>
</tr>
<tr>
<td></td>
<td>STRL</td>
<td>79.12%</td>
<td>51.88</td>
</tr>
<tr>
<td>Area 4 (3662 samples)</td>
<td>from scratch</td>
<td>73.55%</td>
<td>38.50</td>
</tr>
<tr>
<td></td>
<td>STRL</td>
<td>73.81%</td>
<td>39.28</td>
</tr>
<tr>
<td>Area 5 (6852 samples)</td>
<td>from scratch</td>
<td>76.85%</td>
<td>48.63</td>
</tr>
<tr>
<td></td>
<td>STRL</td>
<td>77.28%</td>
<td>49.53</td>
</tr>
</tbody>
</table>

We fine-tune the pre-trained model on KITTI 3D object detection benchmark [16], a subset of the KITTI raw data. In this benchmark, each point cloud is annotated with 3D object bounding boxes. The subset includes 3,712 training samples, 3,769 validation samples, and 7,518 test samples. Table 5 tabulates results. On all three categories, models pre-trained with STRL outperform the model trained from scratch. In particular, for the cyclist category where the least training samples are available, the proposed STRL generates a marked performance elevation. We further freeze the backbone model while fine-tuning; the results reveal that models with the pre-trained backbone reach a comparable performance compared with training models from scratch.

Table 5: 3D object detection fine-tuned on KITTI. We report 3D detection performance with moderate difficulty on the val set of KITTI dataset. Performances below are evaluated by mAP with 40 recall positions.

<table>
<thead>
<tr>
<th>Method</th>
<th>Car (IoU=0.7)</th>
<th>Pedestrian</th>
<th>Cyclist</th>
</tr>
</thead>
<tbody>
<tr>
<td>PV-RCNN (from scratch)</td>
<td>84.50</td>
<td>90.53</td>
<td>57.06</td>
</tr>
<tr>
<td>STRL + PV-RCNN (frozen backbone)</td>
<td>81.63</td>
<td>87.84</td>
<td>39.62</td>
</tr>
<tr>
<td>STRL + PV-RCNN</td>
<td>84.70</td>
<td>90.75</td>
<td>57.80</td>
</tr>
</tbody>
</table>

5.3. Analytic Experiments and Discussions

Generalizability: ScanNet vs ShapeNet Pre-training

What kind of data would endow the learned model with better generalizability to other data domains remains an open problem in 3D computer vision. To elucidate this problem, we pre-train the model on the existing largest natural dataset ScanNet and synthetic data ShapeNet, and test their generalizability to different domains. Table 6 tabulates our cross-domain experimental settings and results, demonstrating the successful transfer from models pre-trained on natural scenes to synthetic shape domain, achieving comparable shape classification performance under linear evaluation.
Additionally, we report an opposite observation in contrast to a recent study [73]. Specifically, the VoteNet model pre-trained on the ShapeNet dataset achieves better performance than ScanNet pre-training in SUN RGB-D object detection, demonstrating better generalizability of ShapeNet data. We believe three potential reasons lead to such conflicting results: (i) The encoder adapted to learn the point cloud features in Xie et al. [73] is too simple such that it fails to capture sufficient information from the pre-trained ShapeNet dataset. (ii) The ShapeNet dataset provides point clouds with clean spatial structures and fewer noises, which benefits the pre-trained model to learn effective representations. (iii) Although the amount of sequence data in ScanNet is large, the modality might still be limited as it only has 707 scenes. This last hypothesis is further backed by our experiments in Data Efficiency below.

**Temporal Transformation** As described in Sect. 3.1 and 5.1, we learn from synthetic view transformation on object shapes and natural view transformation on physical scenes. To study their effects, we disentangle the combinations by removing certain transformations to generate training data of synthetic shapes when pre-training on the ShapeNet dataset; Table 7a summarizes results. For physical scenes, we pre-train PV-RCNN on the KITTI dataset and compare the models trained with and without sampling input data from natural sequences; Table 7b summarizes the results. Temporal transformation introduces substantial performance gains in both cases.

**Spatial Data Augmentation** We investigate the effects of spatial data augmentations by turning off certain types of augmentation; see Table 8. By augmenting the point clouds into different shapes and dimensions, random crop boosts the performance, whereas random cutout hurts the performance as it breaks the point cloud’s structural continuity, crucial for point-wise feature aggregation from neighbors.

**Data Efficiency** To further analyze how the size of training data affects our model, we pre-train the DGCNN model with a subset of ScanNet dataset by sampling 25,000 frames depth images from the entire 1,513 sequences. Evaluated on ModelNet40, the model’s performance only drops about 0.5% for both linear evaluation and fine-tuning compared with training on the whole set with 0.4 million frames; such results are similar to 2D image pre-training [22]. We hypothesize that increasing the data diversity, but not sampling density, would improve performances in self-supervised 3D representation learning.

**Robustness** We observe that the proposed STRL can learn the self-supervised representations by simple augmentations; it robustly achieves a satisfying accuracy (about 85%) on ModelNet40 linear classification. Nevertheless, it differs from the results shown in 2D image pre-training [6, 19], where data augmentations affect the ImageNet linear evaluation by up to 10%. We hypothesize that this difference might be ascribed to the general down-sampling process performed on the point clouds, which introduces structural noises and helps the invariant feature learning.

### 6. Conclusion

In this paper, we devise a spatio-temporal self-supervised learning framework for learning 3D point cloud representations. Our method has a simple structure and demonstrates promising results on transferring the learned representations to various downstream 3D scene understanding tasks. In the future, we hope to explore how to extend current methods to holistic 3D scene understanding [28, 27, 26, 8, 30, 50, 31] and how to bridge the domain gap by joint training of unlabeled data from various domains.
References


