# Self-Supervised Pretraining for Large-Scale Point Clouds

**Zaiwei Zhang**
AWS AI
Santa Clara, CA 95054
zaiweiz@amazon.com

**Min Bai**
AWS AI
Santa Clara, CA 95054
baimin@amazon.com

**Erran Li**
AWS AI
Santa Clara, CA 95054
lilimam@amazon.com

## Abstract

Pretraining on large unlabeled datasets has been proven to improve the downstream task performance on many computer vision tasks, such as 2D object detection and video classification. However, for large-scale 3D scenes, such as outdoor LiDAR point clouds, pretraining is not widely used. Due to the special data characteristics of large 3D point clouds, approaches for 2D pretraining frameworks tend to not generalize well to this domain. In this paper, we propose a new self-supervised pretraining method that targets large-scale 3D scenes. We pretrain commonly used point-based and voxel-based model architectures and show the transfer learning performance on 3D object detection and semantic segmentation. We demonstrate the effectiveness of our approach on both dense 3D indoor point clouds and sparse outdoor LiDAR point clouds.

## 1 Introduction

Accurate and reliable perception is core to the capabilities and safety of autonomous and robotic systems such as self-driving vehicles and indoor robotics. To accurately measure the structure of their surroundings, such platforms usually rely on LiDARs and depth cameras for outdoor and indoor scenes, respectively. Given the 3D point clouds, models are built for a variety of perception tasks, such as object detection, scene segmentation, localization, and mapping.

Modern deep learning models have shown great capabilities when large quantities of annotated data are given. However, generating the required amount of annotations is a laborious and expensive manual process [13; 59; 17]. This is especially true for labeled datasets targeting object detection and semantic segmentation of large real-world scenes, which can contain dozens of independently moving objects and irregularly shaped background material. Moreover, the 3D data is difficult to clearly visualize, manipulate, and annotate on a 2D screen. In the case of outdoor LiDAR scans, the annotation tasks are further complicated by the sparsity of the typical point cloud. However, the cost of acquiring a large quantity of unlabeled sensor data is usually much lower compared to manual annotation. Hence, developing algorithms to automatically discover useful patterns and strengthen the performance of models with few annotated examples is of great value.

Recently, researchers have explored self-supervised learning (SSL) to bootstrap model training in various domains. In computer vision, [26; 8; 22; 25] make use of datasets such as ImageNet [52] with carefully designed pretext tasks to learn useful features for downstream tasks such as object detection and segmentation. These techniques rely on the enormous ImageNet dataset, which depicts

36th Conference on Neural Information Processing Systems (NeurIPS 2022).

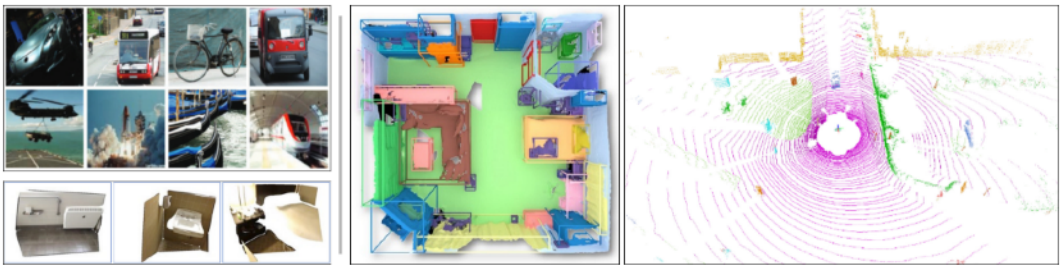

**Figure 1:** Complexities of datasets. Other techniques pretrain on simpler scenes like ImageNet [52] (upper left) where each image prominently features one foreground object or single ScanNet [13] frames (lower left). On the other hand, we target highly complex indoor scenes like aggregated complete ScanNet scenes (middle) and LiDAR scans of road scenes like SemanticKITTI [5] (right) with dozens of different objects (colored).

tens of thousands of object types. Most images prominently feature a single foreground object, which can be well-described by a global feature vector for each sample. In the 3D point cloud domain, several works [74; 67] have proposed pretraining techniques to learn useful features for downstream tasks. However, these methods likewise primarily focus on much simpler scenes, such as single objects or a limited indoor scene captured by a depth camera from a single viewpoint.

On the other hand, we target complex 3D scenes with a multitude of objects and background material. We visualize the differences between the subject datasets of existing approaches and our method in Figure 1. We first consider large-scale, complete 3D indoor scene models generated from combined depth images. Next, we explore outdoor environments with LiDAR points up to 100m away on dozens of objects and complex background regions. We refer to our method as SSPL (**s**elf **s**upervised **p**retraining for **l**arge-scale point clouds). Unlike most of the prior work, we propose a technique which performs contrastive learning on local features extracted from sub-regions (volumes) within large-scale point clouds. Our algorithm makes use of a local contrastive loss within a scene and as well as a global contrastive loss across different scenes. As we will demonstrate, our method involving local feature reasoning greatly outperforms the existing methods in our settings.

In practice, [73] has noted that there are significant domain gaps attributable to the LiDAR sensor due to its sparsity, even when the observed scenes and downstream tasks are highly similar. Motivated by this, we explore the setting of in-domain self-supervised pre-training for LiDAR point clouds. As we will demonstrate, existing 3D SSL methods [74; 67] are unsuitable for highly complex outdoor scenes, motivating the need for new algorithms.

In this paper, we make the following contributions:

- We propose a self-supervised pretraining method which leverages reasoning over local volume level features for 3D point clouds.
- We introduce a novel ClusterInfoNCE loss to efficiently perform contrastive learning globally for a large set of local volume features.
- We show that our pretraining approach outperforms baselines on multiple downstream tasks on both single-view outdoor LiDAR and large-scale indoor point clouds, with up to +6.2% higher AP25 in object detection and +3.6% mIoU in semantic segmentation.

## 2 Related work

Our method builds on and extends the work from the self-supervised learning literature, with a focus on large-scale 3D point clouds. In this section, we give an overview of the recent advances in self-supervised learning, as well as perception and representation learning in 3D.

**Self-supervised learning for images** Self-supervised learning has been studied extensively in machine learning and computer vision [63; 45; 51; 53; 40]. There are many approaches for 2D representation learning, such as clustering [6; 7; 32], GANs [15; 41], various pretext tasks [14; 43; 64] *etc*. Recent advances [42; 26; 28; 8; 10; 61; 36; 19; 22; 25] have shown that self-supervised pretraining exhibits great advantages in transfer learning, especially for benchmark with limited labels. [9; 4] both use similar local and global contrastive learning approaches but [9] uses domain specific

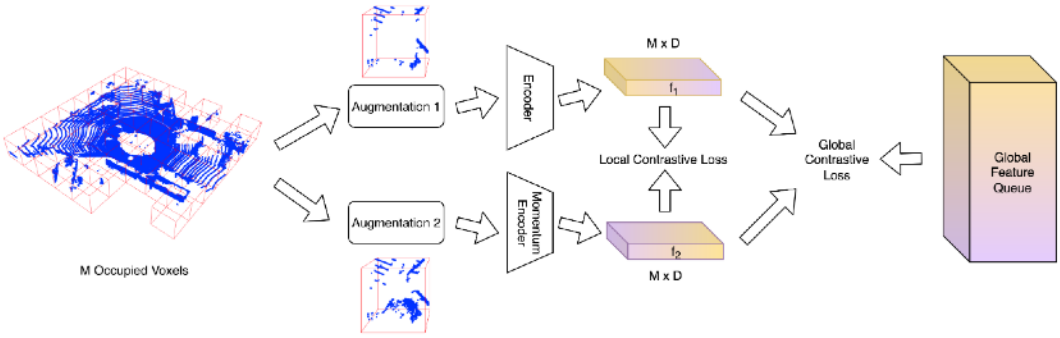


**Figure 2:** Overview of SSPL- a 3D representation learning method for large-scale point clouds pretraining. We split a 3D scene point cloud into $M$ occupied volumes and apply two different data augmentations for point clouds in each volume. In this figure, we show data augmentation examples of a single volume. We use a neural network encoder and a momentum encoder to extract spatial features which are pooled and projected to obtain local volume-level features. With paired volume features, SSPL optimizes the local contrastive loss between paired features and optimizes a global contrastive loss between the local features and the features in the global feature queue.

clustering for their global feature learning and [4] mainly studies the local feature affinities while we explicitly conduct similarity measures across different instances. [27] conducts data augmentation on local features but not global reasoning across different images. Our work adopts the momentum encoder from *MoCo* [26] to extract features for instance discrimination, instead of a memory bank.

**Self-supervised learning for 3D data** Studies on unsupervised feature learning for 3D data have mainly been focused on single 3D object representation with applications on reconstruction, classification or part segmentation [2; 16; 23; 24; 38; 54; 65; 71; 34; 1]. However, due to the domain gap issues, features learned from single 3D object representations does not transfer well to scene level understanding tasks [67]. Recently, there has been a growing interest in designing self-supervised methods to build representations of scene level point clouds [67; 31; 74]. Xie *et al.* [67] proposed a self-supervised method based on enforcing contrastive learning on point correspondences between different views of 3D scenes. These point-wise correspondences can be acquired after aligning different depth maps into a unified 3D scene, which requires time-consuming 3D registration algorithms or specific human labeling efforts on identifying static correspondences. In the case of sparse outdoor LiDAR point clouds, temporal correspondences are often not well defined. Similarly, another line of work [31; 18] rely on multiple scene scans' alignments. On the other hand, our method only requires single frame 3D scans, making it more generally applicable. Zhang *et al.* [74] conducted an instance discrimination task on global scene-level features while our method focuses on reasoning across local patches.

**Supervised 3D perception** With the significant applications of robotics and self-driving vehicles, 3D perception has been a widely studied topic. Among the many perception tasks, supervised object detection [48; 56; 69; 55; 57; 37; 72; 11; 70; 75] and semantic segmentation [76; 62; 68; 29; 30; 33] have been explored by a large number of prior works. Because of the popularity and usefulness of these tasks, we select them as the downstream tasks to analyze the effectiveness of our approach. However, we note that as our self-supervision is task agnostic, it can be applied to other downstream tasks. To achieve these tasks, a number of neural network backbones have been proposed to process 3D point cloud data, which is usually given as an unordered and irregular set of points [49; 20; 60]. In our work, we use both the popular PointNet++ [50] based on pointwise reasoning and 3D U-Net [68] based on sparse voxel convolutions to demonstrate the general effectiveness of our approach. Recently, [44] has proposed a similar local contrastive learning approach on clustered point clouds but they did not conduct global reasoning for those local point clusters.

## 3 Method

We introduce our self-supervised pretraining framework: SSPL as outlined in Algorithm 1, and visualized in Figure 2. As shown in Figure 1, we focus on large-scale point clouds, which contain numerous objects and distinct structures within each scene. In these cases, a global vector encoding requires a combinatorial and intractable level of expressiveness to describe the attributes and

**Algorithm 1** Training Framework for SSPL

**Input** Backbone architecture NN; Dataset $\mathcal{D} = \{\mathbf{X}\}_{i=1}^{N}$; Global feature queues $F_g \in \mathrm{R}^{C \times D}$;
**Output** Pre-trained weight for the NN
**for** $x_i$ in $X$ **do**
- Split $x_i$ into $M$ occupied volumes: $x_i = \{V\}_{i=1}^{M}$
- From $x_i$, generate two augmented versions $x_i^1$ and $x_i^2$
- Generate *local* crops $v_i^1$ in $x_i^1$ and *global* crops $v_i^2$ in $x_i^2$
- Compute volume features $f^1, f^2 \in \mathrm{R}^{M \times D}$
- Compute local contrastive loss $L_c(f^1, f^2)$ and global contrastive loss $L_g(f^1, f^2, F_g)$
- Update $F_g$ with $f^2$ and update NN

**end for**

locations of the various objects. Therefore, we propose to perform representation learning on local features. We first describe local volume features extraction in Section 3.1, and introduce the local contrastive loss in Section 3.2. Finally, we discuss the formulation of the global contrastive reasoning in Section 3.3.

### 3.1 Local feature extraction

Given a large scale 3D point cloud instance $x_i$, we split it into $M$ occupied volumes. Similar to patches in 2D images, we can predefine a volume size and sample up to $M$ occupied volumes based on typical scale of input scenes. $x_i$ is then transformed into two augmented views $x_i^1$ and $x_i^2$ by applying random rotation and scaling. We then sample points in each occupied volume and generate two crops with different sizes for two augmented views. The smaller crops $v_i^1$, generated from $x_i^1$, are defined as *local* crops while larger ones $v_i^2$, generated from $x_i^2$, are *global* crops. This scheme creates natural local-global correspondences, which enables the network to predict global features from local information in feature learning. $x_i^1$ and $x_i^2$ are mapped to feature vectors by applying a neural network. After acquiring the per-point/per-voxel features, we group the features by their corresponding volumes and apply max-pooling to the last layer's features. We then apply a two layer MLP as in [10] and L2 normalization to compute the volume features $f^1$ and $f^2$.

**Momentum encoder** Adapting the method of He *et al*. [26], we use an encoder $\theta_s$ to extract features for $x_i^1$ and a momentum encoder $\theta_t$ to extract features for the augmented view $x_i^2$. The weights of the momentum encoder are updated with an exponential moving average (EMA) from the encoder's weights. The update rule is $\theta_t \leftarrow \lambda\theta_t + (1-\lambda)\theta_s$ with a fixed $\lambda = 0.999$.

### 3.2 Local volume reasoning

Traditionally, given a pair of volume features $f^1, f^2 \in \mathrm{R}^{M \times D}$ from two augmented versions of $x_i$ instance, feature consistency is usually applied as a pre-text task. For contrastive learning approaches, InfoNCE [26] is usually enforced, where volume features $f_1$ and $f_2$ are pushed to be similar and volume features $f_1$ are dissimilar to all other volume features from other instances. However, computation costs increase linearly as the number of feature embedding increases for each batch. Traditionally, InfoNCE assumes that there is one feature embedding for each data instance. Since in average there are about 300 occupied volumes for each scene in our experiments, it is computationally infeasible to perform InfoNCE on each volume feature with a large set of global features as negative examples.

Therefore, we adapt the PointInfoNCE [67] loss to conduct contrastive learning between different volumes of the same point cloud. Compared to PointInfoNCE, we reason about volume-level information, where the loss is defined as:

$$L_c = -\sum_{j \in M} \log \frac{\exp(f_j^1 \cdot f_j^2/\tau)}{\sum_{k \in M} \exp(f_j^1 \cdot f_k^2/\tau)} \tag{1}$$

Here, $M$ is the set of all occupied volumes for the $x_i$ instance, and $\tau$ is a temperature parameter [66]. The positive examples are the two augmented crops from the same volume, while negative examples

are from other volumes under the same instance. In practice, we find that performing this local volume reasoning alone as a pre-text task gives comparable or better pretrained models compared to baseline methods. Please refer to Section 5.4 for experimental evidences.

Although applying BYOL [22] as a pretext task does not yield computational issues, we find that it results in minimal performance gains in our settings. Please see Section 5.3 for more details.

### 3.3 Global volume reasoning

One natural objective for global contrastive reasoning is to use instance discrimination on volume features. As mentioned before, this objective does not scale well with large numbers of occupied volumes. Across different large scale scene datasets, we observe that there are common objects appearing in almost every scene, *e.g.* road and car in outdoor scenes or chair and sofa in indoor scenes. Inspired by this, we propose to conduct global volume reasoning through a fixed number of global clusters. This reduces the computational burden and adds regularization during pretraining.

**Preliminary on unsupervised feature clustering** Traditional methods on feature clustering, such as k-means and DBSCAN have proven to be effective but time-consuming to apply on large dataset. Recently, a line of methods [3; 8] show that pseudo-label assignment can be formulated as an optimal transport problem, where clustering can be done efficiently by the Sinkhorn-Knopp algorithm [12]. We follow a similar approach for our feature clustering steps.

**Global feature queue** We first build a global feature queue $F_g$ to store features of occupied volumes over multiple data instances. Adapting from He *et al*. [26], the global feature queue is created with volume features $f_2$ from the momentum encoder. The following steps are then used during training.

**Global feature clustering** For building global clusters, we follow the same method proposed by Asano *et al*. [3]. Given global feature queue $F_g \in \mathrm{R}^{C \times D}$, where $C$ is the queue size and $D$ is the feature dimension, we apply a classification head $h_g : \mathrm{R}^D \rightarrow \mathrm{R}^K$ to convert the global feature vectors to class (cluster ID) scores. We then map them to class probabilities via a softmax operator: $P = softmax(F_g \cdot h_g)$. We find the class label assignments $Q$ by performing iterative Sinkhorn-Knopp updates [12].

**Global contrastive loss** Given a pair of volume features $f^1, f^2$, we query the global feature queue with local features $f^2$ to extract the corresponding cluster labels for each volume. We use kNN on $l2$ feature distances when querying, and we set $k = 1$ for clustering.

With volume level class labels, it is natural to use a classification pretext task for pretraining. However, compared with pretraining on object-level features (*e.g.* ImageNet), our volume-level features usually lack class diversity and feature diversity. For indoor scenes, there are repeating wall and floor volumes, while a majority of volumes are roads and buildings in outdoors scenes. We observe that such pretraining strategy provides minimal improvements. (Please see section 5.4 for details.) To directly apply feature representation learning across scenes, we propose a **ClusterInfoNCE** loss:

$$L_g = -\frac{1}{K}\sum_{i=1}^{K} \log \frac{\exp(f_i^1 \cdot knn(f_i^2, F_g)/\tau)}{\sum_{j!=i} \exp(f_i^1 \cdot F_g^j/\tau)} \tag{2}$$

where $K$ is the number of clusters and $f_i^1$ and $f_i^2$ are drawn from $f^1$ and $f^2$ if their corresponding class labels are $i$. $knn(f_i^2, F_g)$ extracts the most similar features of $f_i^2$ from $F_g$, which serve as positive examples. For negative examples, $F_g^j$ are sampled from $F_g$ if their corresponding class labels are $j$. In our experiments, we find that using 20k negative examples works well. Since the runtime of **ClusterInfoNCE** loss depends only on number of clusters used, it scales well with more occupied volumes. By optimizing this loss, we are explicitly enforcing feature consistency across scenes and encouraging the volume features to form clusters globally. As shown in Section 7.4, we observe that adding the **ClusterInfoNCE** loss results in improvements on various downstream tasks. As well, we compare it with different negative sampling strategies in Section 5.4.

### 3.4 Implementation details

In our experiments, we set the size of global feature queue to be 300K. We use a temperature value of 0.1 while computing the non-parametric softmax in Eq 1 and 3. The local and global contrastive

**Table 1:** Fine-tuning performance on ScanNet Dataset and SUN RGB-D Dataset (AP25)

| | % of Scan Used for Fine-Tuning | | | | % of SUN Used for Fine-Tuning | | | |
|---|---|---|---|---|---|---|---|---|
| Self-Supervision Method | 20% | 50% | 70% | 100% | 20% | 50% | 70% | 100% |
| None | 46.1 | 52.3 | 54.0 | 58.6 | 45.2 | 55.2 | 55.6 | 57.4 |
| BYOL [22] | 48.2 | 54.0 | 55.5 | 59.1 | 47.7 | 55.6 | 56.4 | 58.0 |
| PointContrast [67] | - | - | - | 59.2 | - | - | - | 57.5 |
| DepthContrast [74] | 50.1 | 56.0 | 57.0 | 60.1 | 45.5 | 56.5 | 57.5 | 59.1 |
| Ours | **53.0** | **58.5** | **60.6** | **63.0** | **47.5** | **57.8** | **58.5** | **60.1** |

loss are equally weighted. For training, we use a standard SGD optimizer with momentum 0.9, and we use a cosine learning rate scheduler [39] which decreases from 0.06 to 0.00006 and train the model for 500 epochs with a batch size of 96.

# 4 Experimental results

In this section, we thoroughly analyze the efficiency of our proposed method. First, we describe the tasks and datasets involved in our experiments. Then, we present quantitative results, demonstrating the increase in performance relative to our comparison baselines. Finally, we take a closer look at the contributions of each of our proposed components.

The key goal of self-supervised learning is to automatically learn feature extractors that are useful for actual downstream tasks. Moreover, it is beneficial if the learned features are useful for different tasks, as this improves the usability of the technique and reduces the need for task-specific redesign. Our approach is chiefly designed for large scale scenes that are not dominated by a single, well defined foreground object. We demonstrate the effectiveness of our self-supervised learning approach by showing that downstream models can achieve significantly higher performance using a small fraction of available annotations. Next, we dive into the features learned by our model to gain additional insights, followed by various ablation studies to justify the design choices in our approach. As mentioned in Section 1, for LiDAR point clouds, the sensor configuration has a tremendous impact on the structure of perceived 3D points. Therefore, we opt to pretrain and finetune our models in each domain individually for outdoor datasets.

## 4.1 Baseline methods

We limit our input data to measurements taken at single instants in time (instead of temporal sequences) for generalization. As such, we use the recent BYOL [22] and DepthContrast [74] (based on MoCo [26]) as baselines. In both cases, we use the same full scene augmentation scheme as proposed in DepthContrast. For indoor scene pretraining, to introduce more scene complexity, we use the point clouds of fully reconstructed scenes built by aggregating multiple sequential measurements of the scene. Therefore, we also use PointContrast [67] as a comparison for the indoor setting.

## 4.2 Large scale dense point clouds

First, we evaluate our feature learning framework on point clouds produced with common depth cameras. In our work, we choose the ScanNet [13] dataset for pretraining. ScanNet contains 1500 large scale indoor scenes, with each scene consisting of a variety of objects. Additionally, it contains more than 10 different room types with diverse room scales. We believe that ScanNet covers a diverse set of indoor object distributions.

Compared to DepthContrast [74] and PointContrast [67], we only utilize the full scans from ScanNet training dataset for pretraining. We aim to learn sharp local features from large scale scenes, and thus we are only interested in feature learning with full views instead of partial views. The pretraining dataset contains 1201 full view scans with 50k points per scan. We use a PointNet++ [50] backbone network for pretraining, as it is widely used for different 3D perception benchmarks.

To evaluate the transfer learning performance of our pretrained models, we use two commonly used indoor 3D object detection benchmarks: ScanNet (SCAN) [13] and SUN RGB-D (SUN) [58]. We

finetune our pretrained model by using it as the backbone initialization of VoteNet [46] (a well known 3D object detector), and report the detection performance using the mean Average Precision at IoU=0.25 (AP25) metric. To study the label efficiency of our pretrained models, we also subsample different sets of the training data used for finetuning. We follow the setup in VoteNet [46] for finetuning.

Based on the results in Table 1, our approach significantly improves downstream object detection performance across all settings, with up to 4.4% improvement on ScanNet and 2.7% on SUN RGB-D. Moreover, on ScanNet, our pretraining approach together with 20% of labeled data allows the downstream detection task to achieve the equivalent performance of using 50% of labeled data and training from scratch. Practically, this can lead to significant savings in labeling cost. Our approach also outperforms the top-performing baseline method DepthContrast. (Note that we reproduce DepthContrast with the original Votenet model and pretrain only on the 1200 instances for fair comparison.) We see a more prominent performance gain on ScanNet compared to on SUN RGB-D, as the former contains only 1201 training scenes while the latter contains more than 5k.

### 4.3 Large scale sparse point clouds

Next, we further increase the size and complexity of the 3D scene by using outdoor scenes collected for autonomous vehicles using LiDAR units. These scenes consist of a large number of different objects and regions, spanning dozens of semantic classes. As we will show, it is critical to consider local regions separately in the self-supervised learning framework.

Our algorithm aims to learn useful features for all regions in a point cloud. While detectable individual objects cover a large fraction of the 3D structures present in the previous indoor datasets, compact and countable foreground objects suitable for detection such as cars and pedestrians consist of only 10% of a typical outdoor scene. As such, we select the semantic segmentation task to validate our method, as it aims to give a meaningful label to every 3D point captured by the sensor.

We use two large scale, well-established public datasets for this task, which contain a large variety of scenes. SemanticKITTI (SK) [5] is based on the original KITTI [17] dataset, which is one of the original autonomous driving datasets that propelled a tremendous amount of progress in the perception field. This dataset contains 19k training frames over 10 sequences with approximately 122k points per scene. The more recent Waymo Open Dataset (WOD) [59] provides a further leap in the scale and variability of data by providing 24k training frames sampled from 850 driving sequences, with a higher point density of 161k points per scene. Both datasets define approximately 20 semantic classes. For these experiments, we use the sparse 3D convolution backbone based on U-Net [77] as it is used in current state of the art semantic segmentation models [68; 76]. As well, we largely follow the training settings of [68].

To show that our approach extracts meaningful representations, we present semantic segmentation results on subsampled sets of the training data provided by SemanticKITTI and WOD. The samples are uniformly and randomly selected from the overall training datasets. To simulate the scenario where different levels of human annotation effort is available, the samples are drawn such that any smaller fractional subset is contained within any larger subset. The results are shown in Table 2. We add a light-weight decoder with two convolutional layers to the pretrained point-wise feature encoding backbone network, and finetune until convergence. In each case, we report the average performance over the checkpoints of the last 15% of the training process to reduce noise.

Our approach significantly improves downstream semantic segmentation when the annotation is limited. For example, we observe 3.6% and 2.3% improvements in mIoU on SemanticKITTI and WOD, respectively, when annotations for 1% subsets of the full datasets are provided. Moreover, these improvements are much more significant than those gained from pre-training with BYOL and DepthContrast, suggesting that beneficial pre-training on large scale point clouds require designs with specific considerations. On the other hand, we observe that the gains are reduced when additional annotated data is provided, as seen in the case with 5% and 10% annotations, as expected.

### 4.4 Feature analysis

Here, we examine the features learned by our self-supervision scheme. As with the other analysis in our work, we compare with the BYOL and DepthContrast baselines, and use the SemanticKITTI

**Table 2:** Fine-tuning performance on SemanticKITTI Dataset and Waymo Open Dataset (mIoU)

| | % of SK Used for Fine-Tuning | | | | % of WOD Used for Fine-Tuning | | | |
|---|---|---|---|---|---|---|---|---|
| Self-Supervision Method | 1% | 2% | 5% | 10% | 1% | 2% | 5% | 10% |
| None | 38.9 | 44.0 | **51.7** | 53.4 | 42.5 | 45.8 | 50.4 | 52.8 |
| BYOL [22] | 38.8 | 43.4 | 51.0 | 52.3 | 42.3 | 46.5 | 49.9 | 53.2 |
| DepthContrast [74] | 39.2 | 44.7 | 49.9 | 52.3 | 42.7 | 45.8 | 50.7 | 53.0 |
| Ours | **42.5** | **46.4** | 51.0 | **53.6** | **44.8** | **47.3** | **51.3** | **53.5** |

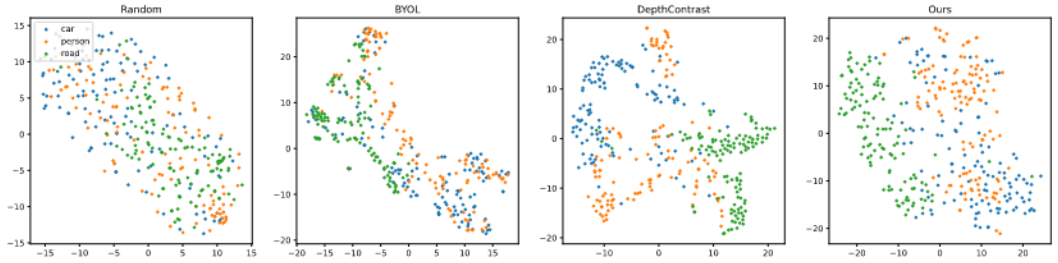


**Figure 3:** t-SNE visualization of features generated by backbone network with various weight settings.

dataset for the study. We visualize the final per-point features from the 3D UNet model using the t-SNE dimensionality reduction and color the points according to their semantic class. For clearer interpretability, we limit the semantic classes to the commonly seen and important classes of *road, person, and car*. The points are subsampled from the validation set.

We see that the embeddings produced using DepthContrast and our approach are both able to separate the three classes in feature space to a much stronger degree than the BYOL baseline, as well as the random initialization which serves as a sanity check. In comparison with DepthContrast, our embeddings have larger variance within the feature space while still being able to separate the semantic classes, suggesting that they can possibly preserve more distinctive geometric information while extracting meaning. This hypothesis is supported by the added performance gain of pretraining using our method compared with DepthContrast as seen in Table 2.

## 5 Ablation study

In this section, we demonstrate the importance of each component in our training framework. We conduct our ablation studies with models pretrained on the widely accepted SemanticKITTI dataset with complex and large-scale scenes in autonomous driving.

### 5.1 Volume cropping ratios

In the left side of Figure 4, we report the impact of applying different cropping ratios for the *local* crop. For the *global* crop, we randomly crop approximately 50% of the original volume so that most of the geometry feature is still preserved in the momentum encoder. We observe that smaller *local* crops produce marginal improvements in finetuning, and that a 10% crop ratio is optimal.

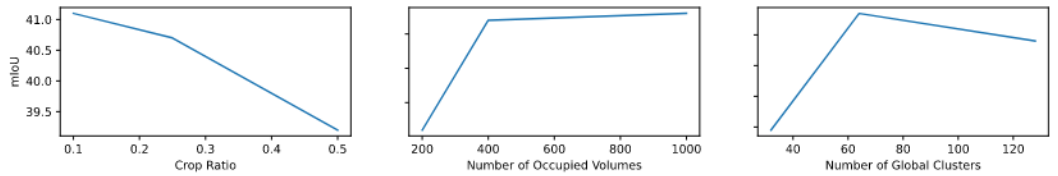


**Figure 4:** Segmentation ablation study on SemanticKITTI using 1% data for fine-tuning. Left: mIoU vs number of occupied volumes, middle: mIoU vs volume crop ratios, right: mIoU vs number of global clusters.

**Table 3:** BYOL vs contrastive methods

| | Scratch | BYOL | Contrastive |
|---|---|---|---|
| 1% SK | 38.9 | 39.8 | **41.7** |

**Table 4:** Different global volume reasoning strategies

| | Local | Cls | Hard | Rand | Our |
|---|---|---|---|---|---|
| 1% SK | 41.7 | 40.8 | 41.0 | 40.5 | **42.9** |

### 5.2 Number of occupied volumes

For local volume reasoning, the number of occupied volumes plays an important role as it directly affects the number of negatives used for local contrastive loss. Intuitively, by increasing the number of occupied volumes, the local contrastive loss forces the network to learn more distinct local features. However, with a fixed model size, the expressiveness of local features saturates after it forms a certain number of clusters, which directly correlates with number of occupied volumes in our training approach. As shown in the middle of Figure 4, we see strong performance with 400 occupied volumes with little gain for further increases.

### 5.3 BYOL vs contrastive method

Instead of contrastive methods, Grill *et al.* [22] proposed a new approach for self-supervised learning **BYOL**, which directly enforces feature consistency between a student network and a teacher network. To better understand our framework, we reproduce the BYOL loss formulation to our best effort to replace the PointInfoNCE in the local volume reasoning. Formally, we use the following mean square error between the normalized predictions and target projections:

$$L_c^{'} = \sum_{j \in M} ||z_\theta(f_j^1) - f_j^2||^2 \tag{3}$$

Similar to Equation 2, $M$ is the set of all occupied volumes, and $z_\theta$ is a predictor for the student network. Based on the results from Table 3, the features learned with BYOL performs significantly worse than the contrastive approach in 1% SemanticKITTI finetuning. We hypothesize that within the same instance, some volumes share very similar feature patterns.

Without explicitly encouraging dispersion within the feature space (e.g. negative examples), BYOL learns less distinguishing features in our setting.

### 5.4 Different strategies for global volume reasoning

As mentioned in Section 3, a natural pretext task for global volume reasoning is classfication. Given a pair of volume features $f^1, f^2$ and their corresponding cluster labels $Q^v$, we apply a linear reprojection $h_l$ to map the volume features to class scores, and optimize the classification loss below:

$$L_{cls} = -\sum_{j \in M} Q_j^v \log P_j^1 - \sum_{j \in M} Q_j^v \log P_j^2 \tag{4}$$

where $P_j^1 = softmax(f^1 \cdot h_l)$ and $P_j^2 = softmax(f^2 \cdot h_l)$. As shown in Table 4 where **Cls** represents the model pretrained on the classification loss, we see that it does not improve the finetuning performance, compared to using local contrastive loss alone (**Local**). We also compare different negative sampling strategies for the ClusterInfoNCE loss. In our approach, after we find the positive feature example for cluster $i$, we sample negative feature examples from other clusters. In Table 4, we compare with only sampling negative examples from cluster $i$ (**Hard**) (which can be seen as hard negative sampling) as well as random sampling from all clusters (**Rand**). Based on the results, our approach (**Our**) provides the largest performance boost.

### 5.5 Number of global clusters

We explore the impact of the number of global clusters used in our global feature queue to compute the contrastive loss (right plot of Figure 4). This hyperparameter is shown to be impactful, where 64 is the best setting for the SemanticKITTI LiDAR scans. We believe that the optimal number is influenced by the complexities of the 3D scenes in question.

## 6 Limitations, future work, and broader impact

Lastly, we discuss the limitations of our model. As the characteristics of 3D sensors vary widely (dense depth scans vs sparse LiDAR point clouds), our approach relies on in-domain unlabeled data as opposed to producing a general purpose foundation model like those used in 2D images and language. Moreover, we do not make use of any temporal information, which can provide significant cues for feature consistency. We believe that these are possible areas for future research. We believe that increasing 3D perception capabilities of autonomous agents will allow them to operate more safely and enrich our lives. As self-supervised learning is able to extract useful representation from unlabeled datasets, it can be less impacted by biases that may influence annotation processes, and possibly improve the reliability and fairness of perception models.

## References


[1] Achituve, I., Maron, H., Chechik, G.: Self-supervised learning for domain adaptation on point-clouds. arXiv preprint arXiv:2003.12641 (2020)

[2] Achlioptas, P., Diamanti, O., Mitliagkas, I., Guibas, L.: Learning representations and generative models for 3d point clouds. In: Proceedings of the International Conference on Machine Learning (ICML). pp. 40–49. PMLR (2018)

[3] Asano, Y.M., Rupprecht, C., Vedaldi, A.: Self-labelling via simultaneous clustering and representation learning. International Conference on Learning Representations (ICLR) (2020)

[4] Bai, Y., Chen, X., Kirillov, A., Yuille, A., Berg, A.C.: Point-level region contrast for object detection pre-training. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 16061–16070 (2022)

[5] Behley, J., Garbade, M., Milioto, A., Quenzel, J., Behnke, S., Stachniss, C., Gall, J.: SemanticKITTI: A Dataset for Semantic Scene Understanding of LiDAR Sequences. In: Proc. of the IEEE/CVF International Conf. on Computer Vision (ICCV) (2019)

[6] Bojanowski, P., Joulin, A.: Unsupervised learning by predicting noise. In: Proceedings of the International Conference on Machine Learning (ICML) (2017)

[7] Caron, M., Bojanowski, P., Joulin, A., Douze, M.: Deep clustering for unsupervised learning of visual features. In: Proceedings of the European Conference on Computer Vision (ECCV) (2018)

[8] Caron, M., Misra, I., Mairal, J., Goyal, P., Bojanowski, P., Joulin, A.: Unsupervised learning of visual features by contrasting cluster assignments. In: NeurIPS (2020)

[9] Chaitanya, K., Erdil, E., Karani, N., Konukoglu, E.: Contrastive learning of global and local features for medical image segmentation with limited annotations. Advances in Neural Information Processing Systems **33**, 12546–12558 (2020)

[10] Chen, T., Kornblith, S., Norouzi, M., Hinton, G.: A simple framework for contrastive learning of visual representations. arXiv preprint arXiv:2002.05709 (2020)

[11] Chen, Y., Liu, S., Shen, X., Jia, J.: Fast point r-cnn. In: Proceedings of the IEEE International Conference on Computer Vision. pp. 9775–9784 (2019)

[12] Cuturi, M.: Sinkhorn distances: Lightspeed computation of optimal transport. Advances in neural information processing systems **26** (2013)

[13] Dai, A., Chang, A.X., Savva, M., Halber, M., Funkhouser, T., Niessner, M.: Scannet: Richly-annotated 3d reconstructions of indoor scenes. In: The IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (July 2017)

[14] Doersch, C., Gupta, A., Efros, A.A.: Unsupervised visual representation learning by context prediction. In: Proceedings of the International Conference on Computer Vision (ICCV) (2015)

[15] Donahue, J., Krahenbühl, P., Darrell, T.: Adversarial feature learning. In: International Conference on Learning Representations (ICLR) (2016)

[16] Gadelha, M., Wang, R., Maji, S.: Multiresolution tree networks for 3d point cloud processing. In: Proceedings of the European Conference on Computer Vision (ECCV). pp. 103–118 (2018)

[17] Geiger, A., Lenz, P., Stiller, C., Urtasun, R.: Vision meets robotics: The kitti dataset. International Journal of Robotics Research (IJRR) (2013)

[18] Gojcic, Z., Litany, O., Wieser, A., Guibas, L.J., Birdal, T.: Weakly supervised learning of rigid 3d scene flow. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition. pp. 5692–5703 (2021)

[19] Goyal, P., Mahajan, D., Gupta, A., Misra, I.: Scaling and benchmarking self-supervised visual representation learning. Proceedings of the International Conference on Computer Vision (ICCV) (2019)

[20] Graham, B., Engelcke, M., van der Maaten, L.: 3d semantic segmentation with submanifold sparse convolutional networks. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 9224–9232 (2018)

[21] Graham, B., Engelcke, M., van der Maaten, L.: 3d semantic segmentation with submanifold sparse convolutional networks. CVPR (2018)

[22] Grill, J.B., Strub, F., Altché, F., Tallec, C., Richemond, P., Buchatskaya, E., Doersch, C., Avila Pires, B., Guo, Z., Gheshlaghi Azar, M., et al.: Bootstrap your own latent-a new approach to self-supervised learning. Advances in Neural Information Processing Systems **33**, 21271–21284 (2020)

[23] Groueix, T., Fisher, M., Kim, V.G., Russell, B.C., Aubry, M.: A papier-mâché approach to learning 3d surface generation. In: Proceedings of the Conference on Computer Vision and Pattern Recognition (CVPR). pp. 216–224 (2018)

[24] Hassani, K., Haley, M.: Unsupervised multi-task feature learning on point clouds. In: Proceedings of the International Conference on Computer Vision (ICCV). pp. 8160–8171 (2019)

[25] He, K., Chen, X., Xie, S., Li, Y., Dollár, P., Girshick, R.: Masked autoencoders are scalable vision learners. arXiv preprint arXiv:2111.06377 (2021)

[26] He, K., Fan, H., Wu, Y., Xie, S., Girshick, R.: Momentum contrast for unsupervised visual representation learning. arXiv preprint arXiv:1911.05722 (2019)

[27] Hénaff, O.J., Koppula, S., Alayrac, J.B., Van den Oord, A., Vinyals, O., Carreira, J.: Efficient visual pretraining with contrastive detection. In: Proceedings of the IEEE/CVF International Conference on Computer Vision. pp. 10086–10096 (2021)

[28] Hénaff, O.J., Razavi, A., Doersch, C., Eslami, S., Oord, A.v.d.: Data-efficient image recognition with contrastive predictive coding. arXiv preprint arXiv:1905.09272 (2019)

[29] Hu, Q., Yang, B., Xie, L., Rosa, S., Guo, Y., Wang, Z., Trigoni, N., Markham, A.: Randla-net: Efficient semantic segmentation of large-scale point clouds. Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (2020)

[30] Hu, Q., Yang, B., Xie, L., Rosa, S., Guo, Y., Wang, Z., Trigoni, N., Markham, A.: Learning semantic segmentation of large-scale point clouds with random sampling. IEEE Transactions on Pattern Analysis and Machine Intelligence (2021)

[31] Huang, S., Xie, Y., Zhu, S.C., Zhu, Y.: Spatio-temporal self-supervised representation learning for 3d point clouds. In: Proceedings of the IEEE/CVF International Conference on Computer Vision. pp. 6535–6545 (2021)

[32] Ji, X., Henriques, J.F., Vedaldi, A.: Invariant information clustering for unsupervised image classification and segmentation. In: Proceedings of the International Conference on Computer Vision (ICCV) (2019)

[33] Jiang, L., Shi, S., Tian, Z., Lai, X., Liu, S., Fu, C.W., Jia, J.: Guided point contrastive learning for semi-supervised point cloud segmentation. Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (2021)

[34] Jing, L., Chen, Y., Zhang, L., He, M., Tian, Y.: Self-supervised modal and view invariant feature learning. arXiv preprint arXiv:2005.14169 (2020)

[35] Kingma, D.P., Ba, J.: Adam: A method for stochastic optimization. In: 3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings (2015), `http://arxiv.org/abs/1412.6980`

[36] Kolesnikov, A., Zhai, X., Beyer, L.: Revisiting self-supervised visual representation learning. arXiv preprint arXiv:1901.09005 (2019)

[37] Li, B., Ouyang, W., Sheng, L., Zeng, X., Wang, X.: Gs3d: An efficient 3d object detection framework for autonomous driving. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 1019–1028 (2019)

[38] Li, J., Chen, B.M., Hee Lee, G.: So-net: Self-organizing network for point cloud analysis. In: Proceedings of the Conference on Computer Vision and Pattern Recognition (CVPR). pp. 9397–9406 (2018)

[39] Loshchilov, I., Hutter, F.: Sgdr: Stochastic gradient descent with warm restarts. arXiv preprint arXiv:1608.03983 (2016)

[40] Masci, J., Meier, U., Cires, D., Schmidhuber, J.: Stacked convolutional auto-encoders for hierarchical feature extraction. In: ICANN. pp. 52–59 (2011)

[41] Mescheder, L., Nowozin, S., Geiger, A.: Adversarial variational Bayes: Unifying variational autoencoders and generative adversarial networks. In: Proceedings of the International Conference on Machine Learning (ICML) (2017)

[42] Misra, I., van der Maaten, L.: Self-supervised learning of pretext-invariant representations. arXiv preprint arXiv:1912.01991 (2019)

[43] Noroozi, M., Favaro, P.: Unsupervised learning of visual representations by solving jigsaw puzzles. In: Proceedings of the European Conference on Computer Vision (ECCV) (2016)

[44] Nunes, L., Marcuzzi, R., Chen, X., Behley, J., Stachniss, C.: Segcontrast: 3d point cloud feature representation learning through self-supervised segment discrimination. IEEE Robotics and Automation Letters **7**(2), 2116–2123 (2022)

[45] Olshausen, B.A., Field, D.J.: Emergence of simple-cell receptive field properties by learning a sparse code for natural images. Nature **381**(6583), 607 (1996)

[46] Qi, C.R., Litany, O., He, K., Guibas, L.J.: Deep hough voting for 3d object detection in point clouds. arXiv preprint arXiv:1904.09664 (2019)

[47] Qi, C.R., Litany, O., He, K., Guibas, L.J.: Deep hough voting for 3d object detection in point clouds. arXiv preprint arXiv:1904.09664 (2019)

[48] Qi, C.R., Liu, W., Wu, C., Su, H., Guibas, L.J.: Frustum pointnets for 3d object detection from rgb-d data. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 918–927 (2018)

[49] Qi, C.R., Su, H., Mo, K., Guibas, L.J.: Pointnet: Deep learning on point sets for 3d classification and segmentation. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 652–660 (2017)

[50] Qi, C.R., Yi, L., Su, H., Guibas, L.J.: Pointnet++: Deep hierarchical feature learning on point sets in a metric space. In: Advances in neural information processing systems. pp. 5099–5108 (2017)

[51] Ranzato, M., Huang, F.J., Boureau, Y.L., LeCun, Y.: Unsupervised learning of invariant feature hierarchies with applications to object recognition. In: CVPR (2007)

[52] Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., Huang, Z., Karpathy, A., Khosla, A., Bernstein, M., Berg, A.C., Fei-Fei, L.: ImageNet Large Scale Visual Recognition Challenge. IJCV **115** (2015)

[53] Salakhutdinov, R., Hinton, G.: Deep Boltzmann machines. In: AI-STATS. pp. 448–455 (2009)

[54] Sauder, J., Sievers, B.: Self-supervised deep learning on point clouds by reconstructing space. In: Advances in Neural Information Processing Systems (NeurIPS). pp. 12962–12972 (2019)

[55] Shi, S., Guo, C., Jiang, L., Wang, Z., Shi, J., Wang, X., Li, H.: Pv-rcnn: Point-voxel feature set abstraction for 3d object detection. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 10529–10538 (2020)

[56] Shi, S., Wang, X., Li, H.: Pointrcnn: 3d object proposal generation and detection from point cloud. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 770–779 (2019)

[57] Shi, S., Wang, Z., Shi, J., Wang, X., Li, H.: From points to parts: 3d object detection from point cloud with part-aware and part-aggregation network. arXiv preprint arXiv:1907.03670 (2019)

[58] Song, S., Lichtenberg, S.P., Xiao, J.: Sun rgb-d: A rgb-d scene understanding benchmark suite. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 567–576 (2015)

[59] Sun, P., Kretzschmar, H., Dotiwalla, X., Chouard, A., Patnaik, V., Tsui, P., Guo, J., Zhou, Y., Chai, Y., Caine, B., et al.: Scalability in perception for autonomous driving: Waymo open dataset. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 2446–2454 (2020), this publication was made using the Waymo Open Dataset, provided by Waymo LLC under license terms available at waymo.com/open.

[60] Thomas, H., Qi, C.R., Deschaud, J.E., Marcotegui, B., Goulette, F., Guibas, L.J.: Kpconv: Flexible and deformable convolution for point clouds. In: Proceedings of the IEEE International Conference on Computer Vision. pp. 6411–6420 (2019)

[61] Tian, Y., Sun, C., Poole, B., Krishnan, D., Schmid, C., Isola, P.: What makes for good views for contrastive learning? arXiv preprint arXiv:2005.10243 (2020)

[62] Unal, O., Dai, D., Van Gool, L.: Scribble-supervised lidar semantic segmentation. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) (2022)

[63] Vincent, P., Larochelle, H., Bengio, Y., Manzagol, P.A.: Extracting and composing robust features with denoising autoencoders. In: ICML (2008)

[64] Wang, X., Gupta, A.: Unsupervised learning of visual representations using videos. In: Proceedings of the International Conference on Computer Vision (ICCV) (2015)

[65] Wang, Y., Solomon, J.M.: Deep closest point: Learning representations for point cloud registration. In: Proceedings of the International Conference on Computer Vision (ICCV). pp. 3523–3532 (2019)

[66] Wu, Z., Xiong, Y., Yu, S.X., Lin, D.: Unsupervised feature learning via non-parametric instance discrimination. In: Proceedings of the Conference on Computer Vision and Pattern Recognition (CVPR) (2018)

[67] Xie, S., Gu, J., Guo, D., Qi, C.R., Guibas, L.J., Litany, O.: Pointcontrast: Unsupervised pretraining for 3d point cloud understanding. In: Proceedings of the European Conference on Computer Vision (ECCV) (2020)

[68] Yan, X., Gao, J., Li, J., Zhang, R., Li, Z., Huang, R., Cui, S.: Sparse single sweep lidar point cloud segmentation via learning contextual shape priors from scene completion. In: Proceedings of the AAAI Conference on Artificial Intelligence. vol. 35, pp. 3101–3109 (2021)

[69] Yan, Y., Mao, Y., Li, B.: Second: Sparsely embedded convolutional detection. Sensors **18**(10), 3337 (2018)

[70] Yang, B., Liang, M., Urtasun, R.: Hdnet: Exploiting hd maps for 3d object detection. In: Proceedings of Conference on Robot Learning (2018)

[71] Yang, Y., Feng, C., Shen, Y., Tian, D.: Foldingnet: Point cloud auto-encoder via deep grid deformation. In: Proceedings of the Conference on Computer Vision and Pattern Recognition (CVPR). pp. 206–215 (2018)

[72] Yang, Z., Sun, Y., Liu, S., Shen, X., Jia, J.: Std: Sparse-to-dense 3d object detector for point cloud. In: Proceedings of the IEEE International Conference on Computer Vision. pp. 1951–1960 (2019)

[73] Yi, L., Gong, B., Funkhouser, T.: Complete & label: a domain adaptation approach to semantic segmentation of lidar point clouds. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition (2021)

[74] Zhang, Z., Girdhar, R., Joulin, A., Misra, I.: Self-supervised pretraining of 3d features on any point-cloud. In: Proceedings of the IEEE/CVF International Conference on Computer Vision. pp. 10252–10263 (2021)

[75] Zhang, Z., Sun, B., Yang, H., Huang, Q.: H3dnet: 3d object detection using hybrid geometric primitives. arXiv preprint arXiv:2006.05682 (2020)

[76] Zhu, X., Zhou, H., Wang, T., Hong, F., Ma, Y., Li, W., Li, H., Lin, D.: Cylindrical and asymmetrical 3d convolution networks for lidar segmentation. arXiv preprint arXiv:2011.10033 (2020)

[77] Çiçek, , Abdulkadir, A., Lienkamp, S.S., Brox, T., Ronneberger, O.: 3d u-net: Learning dense volumetric segmentation from sparse annotation. MICCAI (2016)

## Checklist

1. For all authors...
   (a) Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope? [Yes]
   (b) Did you describe the limitations of your work? [Yes]
   (c) Did you discuss any potential negative societal impacts of your work? [Yes]
   (d) Have you read the ethics review guidelines and ensured that your paper conforms to them? [Yes]
2. If you are including theoretical results...
   (a) Did you state the full set of assumptions of all theoretical results? [N/A]
   (b) Did you include complete proofs of all theoretical results? [N/A]
3. If you ran experiments...
   (a) Did you include the code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL)? [No] We provide the detailed information about our model architecture in the supplementary material. Our codes are still under internal review and we will release the codes as soon as it's approved internally.
   (b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? [Yes] We provide the detailed
   (c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [No] We try to fix the random seed during pretraining, and we always measure the finetuning performance after the loss converges.
   (d) Did you include the total amount of compute and the type of resources used (e.g., type of GPUs, internal cluster, or cloud provider)? [Yes] We have provided this information in the supplementary material.
4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets...
   (a) If your work uses existing assets, did you cite the creators? [Yes]
   (b) Did you mention the license of the assets? [N/A] We used widely tackled academic datasets with clear license information in their respective publications as cited in our work.
   (c) Did you include any new assets either in the supplemental material or as a URL? [N/A]
   (d) Did you discuss whether and how consent was obtained from people whose data you're using/curating? [N/A]
   (e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? [N/A]
5. If you used crowdsourcing or conducted research with human subjects...
   (a) Did you include the full text of instructions given to participants and screenshots, if applicable? [N/A]
   (b) Did you describe any potential participant risks, with links to Institutional Review Board (IRB) approvals, if applicable? [N/A]
   (c) Did you include the estimated hourly wage paid to participants and the total amount spent on participant compensation? [N/A]

# 7 Appendix

We first provide the model architecture details in Section 7.1. We then discuss the hyper-parameters used in training and finetuning for all our downstream tasks in Section 7.2. We also provide the implementation details on the baseline methods in Section 7.3, and we show additional results, such as per-class detection and segmentation performance in Section 7.4. Finally, we show more feature visualizations in Section 7.5.

## 7.1 Model architecture

**Point-based model** We use PointNet++ model for 3D object detection with dense point clouds. For ScanNet and SUN RGB-D, we use the same model. As shown in Table **??**, PointNet++ is consisted of four set abstraction layers and two feature up-sampling layers, as designed in [46]. Each SA layer is specified by (n, r, [$c_1$, ..., $c_k$]), where n represents number of output points, r represents the ball-region radius of the reception field, $c_i$ represents the feature channel size of the i-th layer in the MLP. Each feature up-sampling (FP) layer upsamples the point features by interpolating the features on input points to output points. Each FP layer is specified by [$c_1$, ..., $c_k$] where $c_i$ is the output of the i-th layer in the MLP.

**Voxel-based model** We use the popular 3D U-Net with sparse computations for efficiency improvements [21; 77; 68; 76]. The input point cloud is voxelized using a regular voxel grid (at a resolution of $0.1 \times 0.1 \times 0.1m$ for SemanticKITTI and $0.1 \times 0.1 \times 0.15m$ for Waymo Segmentation). To eliminate issues with intensity calibration and scaling, we use a binary occupancy grid. The model consists of one input convolution followed by 6 layers of U-Net blocks with the standard skip links between the encode and decode branches as proposed by [77] and used by [68]. Each block consists of a 3D convolution with a 2x downsampling, followed by one sub-block (batch normalization, ReLU activation, and 3D convolution layer), and subsequently two similar sub-blocks to merge the higher layer features. The final output of the network is reprojected to yield the 64-dimensional per-point features which are used during pretraining. In the finetuning or inference mode, we further add a two layer per-point MLP as an output head. The details are listed in Table 6.

## 7.2 Finetuning details

**Object detection for ScanNet and SUN RGB-D** We use the same configurations in VoteNet [47] for finetuning. We apply Adam optimizer [35] and use a base learning rate 0.001 with a 0.1× weight decrease at 80, 120 and 160 epochs. The model is trained for 180 epochs in total. We use a batch size of 8 for both ScanNet and SUN RGB-D. We use the same configuration for training from scratch and finetuning, and we only load the pretrained PointNet++ backbone during fine-tuning.

**Semantic segmentation for KITTI and Waymo** We use largely the same configurations as in JS3C [68] for their segmentation network, with small modifications in number of epochs to account for the greatly reduced training dataset sizes in our finetuning experiments. We train the model for 800, 600, 600, and 500 epochs on the 1%, 2%, 5%, and 10% splits, respectively, with a base learning rate 0.001 and a 0.7x weight decay at every tenth of the training process. We use a batch size of 32 for both datasets, and as before use the same configurations for training from scratch and finetuning experiments.

| layer name | input layer | type | output size | layer params |
|---|---|---|---|---|
| sa1 | point cloud (xyz) | SA | (2048, 3+128) | (2048, 0.2, [64, 64, 128]) |
| sa2 | sa1 | SA | (1024, 3+256) | (1024, 0.4, [128, 128, 256]) |
| sa3 | sa2 | SA | (512, 3+256) | (512, 0.8, [128, 128, 256]) |
| sa4 | sa3 | SA | (256, 3+256) | (256, 1.2, [128, 128, 256]) |
| fp1 | sa3,sa4 | FP | (256, 3+256) | [256, 256] |
| fp2 | sa2,sa3 | FP | (256, 3+256) | [256, 256] |

**Table 5: PointNet++ Network Architecture** used in Section 7.4

### 7.3 Baseline methods implementations

For this paper, we want to demonstrate that representation learning in local patch/volume level is better than representation learning in global level. Therefore, we consider the general setting for global-level feature learning, and we are not evaluating the benefits from multi-modality (2D-3D), multi-representation (point-voxel), and etc.

**DepthContrast [74]** As mentioned before, we only focus on the general setting. Thus, we choose to only use the within-format setting from DepthContrast to clearly compare the benefits between local and global representation learning. Our proposed approach can be adapted to across-format learning in a similar fashion. We will address that in future work. For pretraining in both dense and sparse point clouds, we use the following data augmentations: RandomCuboidCropping, RandomLocalDrop, RandomRotation, RandomFlipping, RandomScaling. We use the same configurations as in the original DepthContrast. Since there are only 1201 training instances in ScanNet, we pretrain the PointNet++ on ScanNet for 3K epochs, with a standard SGD optimizer with momentum 0.9. We also use a cosine learning rate scheduler [39] which decreases from 0.06 to 0.00006 and train the model with a batch size of 96. On KITTI ans Waymo, we use the same pretraining settings except that we pretrain for 500 epochs each.

**BYOL [22]** For BYOL baseline, we use a voxel-based model for pretraining, and we adopt the same data augmentation from DepthContrast. For the BYOL formulation, we changed the momentum encoder from DepthContrast to a teacher network and we add a predictor module to the student network, which is the encoder from DepthContrast. We use different initialization for the teacher and student network, and the weights of the teacher network are updated with an exponential moving average (EMA) from the student's weights. The update rule is $\theta_t \leftarrow \lambda\theta_t + (1-\lambda)\theta_s$ with a fixed $\lambda = 0.999$. We use the following loss formulation to pretrain the model:

$$L_c^{'} = \sum_{i \in N} ||z_\theta(F_i^1) - F_i^2||^2 \tag{5}$$

$N$ is the total number of training instances and $F_i^1$, $F_i^2$ is the i-th global encoding from the student and teacher network, respectively. $z_\theta$ is a predictor for the student network. We use the same pretraining parameters as in DepthContrast.

### 7.4 More detailed results

**Per-category results for ScanNet** We provide the detailed per-category object detection results for 100% annotated data setting in Table 7. We see that our pretraining improves the downstream performance on most of the categories, especially on cabinet, door, and shower curtain. With our local reasoning approach, we are able to extract more distinct features even with the planar surfaces, which boosts the performance of those categories. We observe similar behaviors in SUN RGB-D.

| input | layer name | output dims |
|---|---|---|
| occupancy grid | conv_in | 16 |
| conv_in | block_1 | 32 |
| block_1 | block_2 | 48 |
| block_2 | block_3 | 64 |
| block_3 | block_4 | 80 |
| block_4 | block_5 | 96 |
| block_5 | block_6 | 112 |
| block_5, block_6 | upblock_5 | 96 |
| block_4, upblock_5 | upblock_4 | 80 |
| block_3, upblock_4 | upblock_3 | 64 |
| block_2, upblock_3 | upblock_2 | 48 |
| block_1, upblock_2 | upblock_1 | 32 |
| conv_in, upblock_1 | features | 64 |
| features | logits | num_classes |

**Table 6: 3D U-Net Network Architecture** used in Section 7.4

**Per-category results for SUN RGB-D** Similar to the ScanNet dataset, we see notable improvements on the SUN RGB-D dataset of our pretraining scheme as compared with training from scratch and the other self-supervised learning baselines. The difference is particularly notable in the *bathtub*, *sofa*, and *bookshelf* classes, which is similar to the case in ScanNet.

**Per-category results for SemanticKITTI** We provide the detailed per-category semantic segmentation results for the 1% annotated data setting in Table 9. We see that our pretraining improves the downstream performance after finetuning on the great majority of object classes, especially on many small or rare classes including *bicyclist*, *person*, *truck*, and *traffic-signs*, where we see up to 14% improvement over training from scratch or the baseline methods.

**Per-category results for Waymo Open Dataset Semantic Segmentation** As with SemanticKITTI, we continue to observe significant improvements across the different semantic classes on the Waymo dataset when 1% of annotations are used. The improvements are particularly notable in the *bicyclist*, *pedestrian*, and *bicycle* categories.

### 7.5 More feature visualizations

In this section, we examine the evolution of the learned features of our approach through the learning process at different epochs. In Figure 5, we see that the learned features gradually spread out in feature space while maintaining significant separation between different classes. This demonstrates the impact of our locally contrastive scheme.

| Pretraining | None | BYOL [22] | DepthCon [74] | Ours |
|---|---|---|---|---|
| all | 58.6 | 59.1 | 60.1 | **63.0** |
| cabinet | 36.3 | 37.4 | 35.8 | **41.2** |
| bed | 87.9 | 88.8 | 87.3 | **89.2** |
| chair | 88.7 | 88.3 | 88.8 | **90.1** |
| sofa | **89.6** | 88.4 | 88.1 | 86.8 |
| table | 58.8 | 61.4 | 63.3 | **65.4** |
| door | 47.3 | 50.1 | 50.1 | **55.1** |
| window | 38.1 | 38.7 | 41.3 | **47.3** |
| bookshelf | 44.6 | **58.9** | 57.7 | 52.9 |
| picture | 7.8 | 6.3 | 7.7 | **9.6** |
| counter | **56.1** | 50.2 | 47.1 | 49.7 |
| desk | **71.7** | 62.4 | 64.5 | 64.1 |
| curtain | 47.2 | 48.5 | 49.0 | **59.3** |
| refrigerator | 45.3 | 45.2 | 49.1 | **56.0** |
| shower curtain | 57.1 | 56.1 | 61.6 | **75.7** |
| toilet | 94.9 | 95.0 | **96.6** | 95.7 |
| sink | 54.7 | 50.9 | 55.4 | **56.0** |
| bathtub | 92.1 | 92.3 | 88.8 | **92.4** |
| garbage bin | 37.2 | 47.2 | **48.5** | 48.1 |

**Table 7: Detailed results on 100% ScanNet**

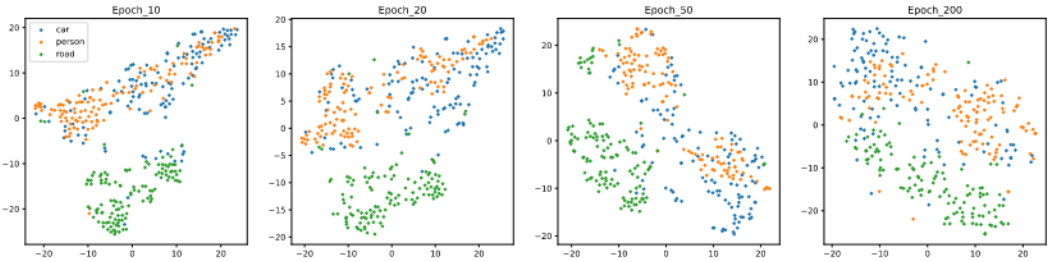


**Figure 5:** t-SNE visualization of features generated by backbone network with various weight settings.

| Pretraining | None | BYOL [22] | DepthCon [74] | Ours |
|---|---|---|---|---|
| all | 57.4 | 58.0 | 59.1 | **60.1** |
| bed | 83.6 | 84.6 | 84.4 | **85.2** |
| table | 49.9 | 49.2 | 50.5 | **51.9** |
| sofa | 64.4 | 63.8 | 63.7 | **67.1** |
| chair | 74.8 | 74.9 | **75.5** | 75.3 |
| toilet | **89.8** | 88.9 | 89.7 | 88.8 |
| desk | 24.1 | 26.2 | 25.8 | **26.2** |
| dresser | 28.9 | 29.4 | **32.1** | 31.9 |
| night_stand | 59.6 | 62.4 | **63.1** | 62.1 |
| bookshelf | 30.7 | 32.1 | 34.4 | **34.5** |
| bathhub | 71.2 | 70.8 | 71.9 | **77.5** |

**Table 8: Detailed results on 100% SUN RGB-D**

**Table 9:** Detailed semantic segmentation results on 1% SemanticKITTI (mIoU)

| Pretraining | None | BYOL [22] | DepthCon [74] | Ours |
|---|---|---|---|---|
| all | 38.9 | 38.8 | 39.2 | **42.5** |
| car | 90.9 | 90.4 | 90.6 | **91.5** |
| bicycle | **3.2** | 1.8 | 1.6 | 1.1 |
| motorcycle | **5.1** | 4.4 | 4.0 | 3.7 |
| truck | 15.8 | 22.1 | 16.6 | **29.9** |
| other-vehicle | 13.1 | 15.4 | 14.3 | **23.2** |
| person | 27.7 | 24.2 | 24.2 | **34.9** |
| bicyclist | 10.7 | 9.6 | 12.6 | **17.5** |
| motorcyclist | 0.0 | 0.0 | 0.0 | 0.0 |
| road | 87.5 | 85.3 | 86.8 | **88.8** |
| parking | 22.0 | 18.3 | 22.3 | **23.2** |
| sidewalk | 66.3 | 65.0 | 65.6 | **69.5** |
| other-ground | **1.1** | 0.1 | 0.3 | 0.2 |
| building | 84.1 | 85.2 | 86.1 | **86.2** |
| fence | 33.8 | 37.5 | 39.9 | **40.5** |
| vegetation | 82.8 | 82.0 | 83.5 | **84.4** |
| trunk | 46.6 | 50.7 | 48.1 | **51.9** |
| terrain | 66.3 | 67.4 | 67.5 | **69.5** |
| pole | 43.4 | 45.2 | 49.9 | **51.5** |
| traffic-sign | 38.3 | 33.4 | 31.5 | **40.8** |

**Table 10:** Detailed semantic segmentation results on 1% Waymo Open Dataset (mIoU)

| Pretraining | None | BYOL [22] | DepthCon [74] | Ours |
|---|---|---|---|---|
| all | 42.5 | 42.3 | 42.7 | **44.8** |
| car | 88.1 | 88.8 | 89.1 | **89.8** |
| truck | 40.9 | **45.2** | 44.5 | 44.3 |
| bus | 34.2 | 37.4 | **41.1** | 37.8 |
| other vehicle | 4.5 | 5.3 | 3.5 | **6.3** |
| motorcyclist | 0.1 | **0.3** | 0.0 | 0.0 |
| bicyclist | 17.5 | 16.7 | 15.3 | **21.3** |
| pedestrian | 68.3 | 70.0 | 70.5 | **71.8** |
| sign | 49.0 | 48.7 | 49.6 | **50.0** |
| traffic light | 23.4 | **23.8** | 20.3 | 23.2 |
| pole | 54.5 | 53.6 | 53.9 | **55.1** |
| construction cone | 28.4 | 27.8 | 24.2 | **29.4** |
| bicycle | 9.9 | 7.7 | 9.4 | **18.5** |
| motorcycle | 21.2 | 19.0 | 23.5 | **24.2** |
| building | 88.9 | 88.6 | 89.1 | **89.7** |
| vegetation | 81.2 | 81.3 | 81.8 | **82.5** |
| tree trunk | 53.8 | 53.8 | 53.3 | **55.3** |
| curb | 48.4 | 47.2 | 47.9 | **50.7** |
| road | 82.2 | 83.2 | 83.1 | **85.2** |
| lane marker | 19.3 | 18.9 | 19.7 | **23.1** |
| other ground | 10.6 | 8.0 | 10.8 | **11.1** |
| walkable | 58.4 | 56.7 | 58.3 | **61.1** |
| sidewalk | 51.4 | 49.6 | 51.1 | **54.3** |