PointContrast: Unsupervised Pre-training for 3D Point Cloud Understanding

Saining Xie[1], Jiatao Gu[1], Demi Guo⋆, Charles R. Qi⋆,
Leonidas Guibas[2]⋆, and Or Litany[2]⋆

[1] Facebook AI Research
[2] Stanford University

Abstract. Arguably one of the top success stories of deep learning is transfer learning. The finding that pre-training a network on a rich source set (*e.g.*, ImageNet) can help boost performance once fine-tuned on a usually much smaller target set, has been instrumental to many applications in language and vision. Yet, very little is known about its usefulness in 3D point cloud understanding. We see this as an opportunity considering the effort required for annotating data in 3D. In this work, we aim at facilitating research on 3D representation learning. Different from previous works, we focus on high-level scene understanding tasks. To this end, we select a suite of diverse datasets and tasks to measure the effect of unsupervised pre-training on a large source set of 3D scenes. Our findings are extremely encouraging: using a unified triplet of architecture, source dataset, and contrastive loss for pre-training, we achieve improvement over recent best results in segmentation and detection across 6 different benchmarks for indoor and outdoor, real and synthetic datasets – demonstrating that the learned representation can generalize across domains. Furthermore, the improvement was similar to supervised pre-training, suggesting that future efforts should favor scaling data collection over more detailed annotation. We hope these findings will encourage more research on unsupervised pretext task design for 3D deep learning.

Keywords: Unsupervised Learning, Point Cloud Recognition, Representation Learning, 3D Scene Understanding

1 Introduction

Representation learning is one of the main driving forces of deep learning research. In 2D vision, the finding that pre-training a network on a rich source set (*e.g.* ImageNet classification) can help boost performance once fine-tuned on the usually much smaller target set, has been key to the success of many applications. A particularly important setting is when the pre-training stage is unsupervised, as this opens up the possibility to utilize a practically infinite train set size. Unsupervised pre-training has been remarkably successful in natural language processing [49, 13], and has recently attracted increasing attention in 2D vision [42, 3, 27, 63, 23, 42, 3, 40, 27, 69, 28, 87, 8].

⋆ Work done while at Facebook AI Research.

In the past few years, the field of 3D deep learning has witnessed much progress with an ever-increasing number of 3D representation learning schemes [1, 16, 74, 21, 36, 67, 22, 15, 81, 12, 9]. However, it still falls behind compared to its 2D counterpart as evidently, in all 3D scene understanding tasks, ad-hoc training *from scratch* on the target data is still the dominant approach. Notably, all existing representation learning schemes are tested either on single objects or low-level tasks (e.g. registration). This status quo can be attributed to multiple reasons: 1) Lack of large-scale and high-quality data: compared to 2D images, 3D data is harder to collect, more expensive to label, and the variety of sensing devices may introduce drastic domain gaps; 2) Lack of unified backbone architectures: in contrast to 2D vision where architectures such as ResNets have proven successful as backbone networks for pre-training and fine-tuning, point cloud network architecture designs are still evolving; 3) Lack of a comprehensive set of datasets and high-level tasks for evaluation.

The purpose of this work is to move the needle by initiating research on *unsupervised pre-training* with *supervised fine-tuning* in deep learning for 3D scene understanding. To do so, we cover four important ingredients: 1) Selecting a large dataset to be used at pre-training; 2) identifying a backbone architecture that can be shared across many different tasks; 3) evaluating two unsupervised objectives for pre-training the backbone network, and 4) defining an evaluation protocol on a set of diverse downstream datasets and tasks.

Specifically, we choose ScanNet [11] as our source set on which the pre-training takes place, and utilize a sparse residual U-Net [51, 9] as the backbone architecture in all our experiments and focus on the point cloud representation of 3D data. For the pre-training objective, we evaluate two different contrastive losses: Hardest-contrastive loss [10], and PointInfoNCE – an extension of InfoNCE loss [42] used for pre-training in 2D vision. Next, we choose a broad set of target datasets and downstream tasks that includes: semantic segmentation on S3DIS [2], ScanNetV2 [11], ShapeNetPart [77] and Synthia 4D [52]; and object detection on SUN RGB-D [57, 55, 32, 70] and ScanNetV2. Remarkably, our results indicate improved performance across all datasets and tasks (See Table 1 for a summary of the results). In addition, we found a relatively small advantage to pre-training with supervision. This implies that future efforts in collecting data for pre-training should favor scale over precise annotations.

Our contributions can be summarized as follows:

- We evaluate, for the first time, the transferability of learned representation in 3D point clouds to high-level scene understanding.
- Our results indicate that *unsupervised pre-training* improves performance across downstream tasks and datasets, while using a single unified architecture, source set and objective function.
- Powered by unsupervised pre-training, we achieve a new state-of-the-art performance on 6 different benchmarks.
- We believe these findings would encourage a change of paradigm on how we tackle 3D recognition and drive more research on 3D representation learning.

2 Related work

Representation learning in 3D Deep neural networks are notoriously data hungry. This renders the ability to transfer learned representations between datasets and tasks extremely powerful. In 2D vision it has led to a surge of interest in finding optimal pretext unsupervised tasks [43, 83, 84, 14, 41, 18, 5, 42, 3, 40, 27, 69, 28, 87, 8, 10]. We note that while many of these tasks are *low-level* (*e.g.* pixel or patch level reconstruction), they are evaluated based on their transferability to *high-level* tasks such as object detection. Being much harder to annotate, 3D tasks are potentially the biggest beneficiaries of unsupervised- and transfer-learning. This was shown in several works on single object tasks like reconstruction, classification and part segmentation [1, 16, 74, 21, 36, 67, 22, 53]. Yet, generally much less attention has been devoted to representation learning in 3D that extends beyond the single-object level. Further, in the few cases that did study it, the focus was on low-level tasks like registration [15, 81, 12]. In contrast, here we wish to push forward research in 3D representation learning by focusing on transferability to more high-level tasks on more complex scenes.

Deep architectures for point cloud processing In this work, we focus on learning useful representation for point cloud data. Inspired by the success in 2D domain, we conjecture that an important ingredient in enabling such progress is the evident standardization of neural architectures. Canonical examples include VGGNet [56] and ResNet/ResNeXt [26, 71]. In contrast, point cloud neural network design is much less mature, as is apparent by the abundance of new architectures that have been recently proposed. This has multiple reasons. First, is the challenge of processing unordered sets [47, 50, 80, 39]. Second, is the choice of neighborhood aggregation mechanism which could either be hierarchical [48, 33, 82, 16, 35], spatial CNN-like [30, 73, 37, 85, 59], spectral [78, 62, 65] or graph-based [72, 64, 68, 54]. Finally, since the points are discrete samples of an underlying surface, continuous convolutions have also been considered [66, 4, 75]. Recently Choy et al. proposed the Minkowski Engine [9], an extension of submanifold sparse convolutional networks [20] to higher dimensions. In particular, sparse convolutional networks facilitate the adoption of common deep architectures from 2D vision, which in turn can help standardize deep learning for point cloud. In this work, we use a unified U-Net [51] architecture built with Minkowski Engine as the backbone network in all experiments and show it can gracefully transfer between tasks and datasets.

3 PointContrast Pre-training

In this section, we introduce our unsupervised pre-training pipeline. First, to motivate the necessity of a new pre-training scheme, we conduct a pilot study to understand the limitations of existing practice (pre-training on ShapeNet) in 3D deep learning (Section 3.1). After briefly reviewing an inspirational local feature learning work *Fully Convolutional Geometric Features (FCGF)* (Section 3.2), we introduce our unsupervised pre-training solution, *PointContrast*, in terms

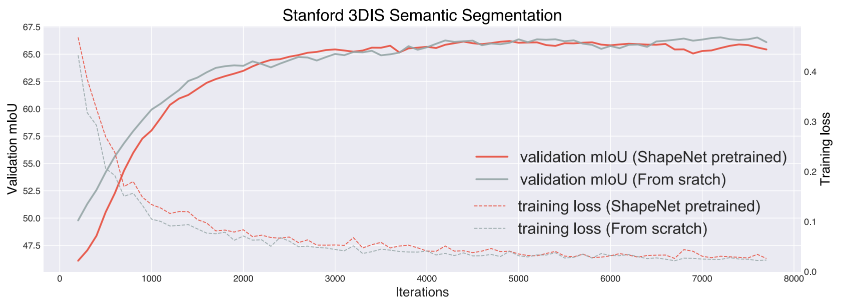


Fig. 1: Training from scratch *vs.* fine-tuning with ShapeNet pre-trained weights.

of pretext task (Section 3.3), loss function (Section 3.4), network architecture (Section 3.5) and pre-training dataset (Section 3.6).

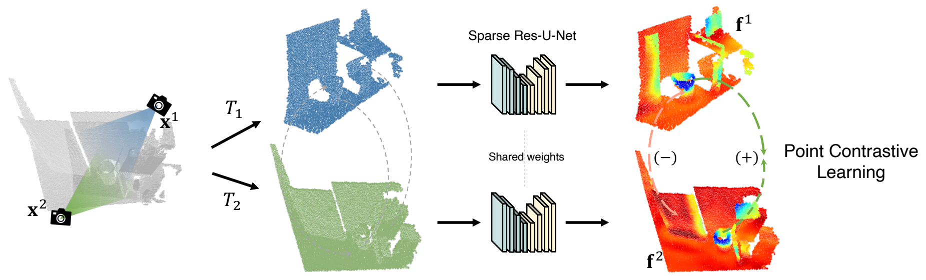
3.1 Pilot Study: is Pre-training on ShapeNet Useful?

Previous works on unsupervised 3D representation learning [1, 16, 74, 21, 36, 67, 22, 53] mainly focused on ShapeNet [7], a dataset of single-object CAD models. One underlying assumption is that by adopting ShapeNet as the ImageNet counterpart in 3D, features learned on *synthetic single objects* could transfer to other real-world applications. Here we take a step back and reassess this assumption by studying a straightforward supervised pre-training setup: we simply pre-train an encoder network on ShapeNet with *full supervision*, and fine-tune it with a U-Net on a downstream task (S3DIS semantic segmentation). Following the practice in 2D representation learning, we use full supervision here as an upper bound to what could be gained from pre-training. We train a sparse ResNet-34 model (details to follow in Section 3.5) for 200 epochs. The model achieves a high validation accuracy of 85.4% on ShapeNet classification task. In Figure 1, we show the downstream task training curves for (a) training from scratch and (b) fine-tuning with ShapeNet pre-trained weights. Critically, one can observe that ShapeNet pre-training, even in the supervised fashion, *hampers* downstream task learning. Among many potential explanations, we highlight two major concerns:

- **Domain gap between source and target data:** Objects in ShapeNet are synthetic, normalized in scale, aligned in pose, and lack scene context. This makes pre-training and fine-tuning data distributions drastically different.
- **Point-level representation matters:** In 3D deep learning, the local geometric features, *e.g.* those encoded by a point and its neighbors, have proven to be discriminative and critical for 3D tasks [47, 48]. Directly training on *object instances* to obtain a global representation might be insufficient.

This led us to rethink the problem: if the goal of pre-training is to boost performance across many real-world tasks, exploring pre-training strategies on

Fig. 2: **PointContrast: Pretext task for 3D pre-training.**



PointContrast: Downstream Tasks for Fine-tuning

Datasets	Real / Synth.	Complexity	Env.	Task	Rel. gain
S3DIS	Real	Entire floor, office	Indoor	Segmentation	(+2.7%) mIoU
SUN RGB-D	Real	Medium-sized cluttered rooms	Indoor	Detection	(+3.1%) mAP0.5
ScanNetV2	Real	Large rooms	Indoor	Segmentation Detection	(+1.9%) mIoU (+2.6%) mAP0.5
ShapeNet	Synth.	Single objects	Indoor & outdoor	Classification	(+4.0%) Acc.*
ShapeNetPart	Synth.	Object parts	Indoor & outdoor	Segmentation	(+2.2%) mIoU*
Synthia 4D	Synth.	Street scenes, driving envs.	Outdoor	Segmentation	(+3.3%) mIoU

Table 1: **Summary of downstream fine-tuning tasks.** Compared to the baseline learning paradigm of training from scratch, which is dominant in 3D deep learning, our unsupervised pre-training method PointContrast boosts the performance across the board when finetuning on a diverse set of high-level 3D understanding tasks.
$*$ indicates results trained using only 1% of the training data.

single objects might offer limited potential. (1) To address the domain gap concern, it might be beneficial to directly pre-train the network on complex scenes with multiple objects, to better match the target distributions; (2) to capture point-level information, we need to design a pretext task and corresponding network architecture that is not only based on instance-level/global representations, but instead can capture dense/local features at the point level.

3.2 Revisiting Fully Convolutional Geometric Features (FCGF)

Here we revisit a previous approach FCGF [10] designed to learn geometric features for *low-level* tasks (*e.g.* registration) as our work is mainly inspired by FCGF. FCGF is a deep learning based algorithm that learns local feature descriptors on correspondence datasets via metric learning. FCGF has two major ingredients that help it stand out and achieve impressive results in registration

Algorithm 1 General Framework of PointContrast

```
Input: Backbone architecture NN; Dataset $X = \{\mathbf{x}_i \in \mathbb{R}^{N \times 3}\}$; Point feature dimension $D$;
Output: Pre-trained weights for NN.
for each point cloud $\mathbf{x}$ in $X$ do
    - From $\mathbf{x}$, generate two views $\mathbf{x}^1$ and $\mathbf{x}^2$.
    - Compute correspondence mapping (matches) $M$ between points in $\mathbf{x}^1$ and $\mathbf{x}^2$.
    - Sample two transformations $\mathbf{T}_1$ and $\mathbf{T}_2$.
    - Compute point features $\mathbf{f}^1, \mathbf{f}^2 \in \mathbb{R}^{N \times D}$ by
    $\mathbf{f}^1 = \mathrm{NN}(\mathbf{T}_1(\mathbf{x}^1))$ and $\mathbf{f}^2 = \mathrm{NN}(\mathbf{T}_2(\mathbf{x}^2))$.
    - Backprop. to update NN with contrastive loss $\mathcal{L}_c(\mathbf{f}^1, \mathbf{f}^2)$ on the matched points.
end
```

recall: (1) **a fully-convolutional design** and (2) **point-level metric learning**. With a fully-convolutional network (FCN) [38] design, FCGF operates on the entire input point cloud (*e.g.* full indoor or outdoor scenes) without having to crop the scene into patches as done in previous works; this way the local descriptors can aggregate information from a large number of neighboring points (up to the extent of receptive field size). As a result, point-level metric learning becomes natural. FCGF uses a U-Net architecture that has a full-resolution output (*i.e.* for N points, the network outputs N associated feature vectors), and positive/negative pairs for metric learning are defined at the point level.

Despite having a fundamentally different goal in mind, FCGF offers inspirations that might address the pretext task design challenges: A fully-convolutional design will allow us to pre-train on the target data distributions that involve complex scenes with a large number of points, and we could define the pretext task directly on points. Under this perspective, we pose the question: *Can we repurpose FCGF as the pretext task for high-level 3D understanding?*

3.3 PointContrast as a Pretext Task

FCGF focuses on local descriptor learning for low-level tasks only. In contrast, a good pretext task for pre-training aims to learn *network weights* that are universally applicable and useful to many high-level 3D understanding tasks. To take the inspiration of FCGF and create such pretext tasks, several design choices need to be revisited. In terms of *architecture*, since inference speed is a major concern in registration tasks, the network used in FCGF is very light-weight; Contrarily, the success of pre-training relies on over-parameterized networks, as clearly evidenced in other domains [13, 8]. In terms of *dataset*, FCGF uses domain-specific registration datasets such as 3DMatch [81] and KITTI odometry [17], which lack both scale and generality. Finally, in terms of *loss design*, contrastive losses explored in FCGF are tailored for registration and it is interesting to explore other alternatives.

In Algorithm 1, we summarize the overall pretext task framework explored in this work. We name the framework *PointContrast*, since the high-level strategy of this pretext task is, contrasting—at the point level—between two transformed point clouds. Conceptually, given a point cloud $\mathbf{x}$ sampled from a certain distribution, we first generate two views $\mathbf{x}^1$ and $\mathbf{x}^2$ that are aligned in the same world coordinates. We then compute the correspondence mapping M between these two views. If $(i, j) \in M$ then point $\mathbf{x}_i^1$ and point $\mathbf{x}_j^2$ are a pair of matched points across two views. We then sample two random geometric transformations T_1 and T_2 to further transform the point clouds into two views. The transformation is what could make the pretext task challenging as the network needs to learn certain *equivariance* to the geometric transformation imposed. In this work, we mainly consider rigid transformation including rotation, translation and scaling. Further details are provided in Appendix. Finally, a contrastive loss is defined over points in two views: we minimize the distance for matched points and maximize the distance of unmatched points. This framework, though coming from a very different motivation (metric learning for geometric local descriptors), shares a strikingly similar pipeline with recent contrastive-based methods for 2D unsupervised visual representation learning [69, 23, 8]. The key difference is that most work for 2D focuses on contrasting instances/images, while in our work the contrastive learning is done densely at the point level.

3.4 Contrastive Learning Loss Design

Hardest-Contrastive Loss The first loss function, hardest-contrastive loss we try, is borrowed from the best-performing loss design proposed in FCGF [10], which adopts a hard negative mining scheme in traditional margin-based contrastive learning formulation,

$$\mathcal{L}_c = \sum_{(i,j)\in\mathcal{P}} \Big\{ [d(\mathbf{f}_i, \mathbf{f}_j) - m_p]_+^2 / |\mathcal{P}| + 0.5[m_n - \min_{k\in\mathcal{N}} d(\mathbf{f}_i, \mathbf{f}_k)]_+^2 / |\mathcal{N}_i| + 0.5[m_n - \min_{k\in\mathcal{N}} d(\mathbf{f}_j, \mathbf{f}_k)]_+^2 / |\mathcal{N}_j| \Big\}$$

Here $\mathcal{P}$ is a set of matched (positive) pairs of points $\mathbf{x}_i^1$ and $\mathbf{x}_j^2$ from two views $\mathbf{x}^1$ and $\mathbf{x}^2$, and $\mathbf{f}_i^1$ and $\mathbf{f}_j^2$ are associated point features for the matched pair. $\mathcal{N}$ is a randomly sampled set of non-matched (negative) points which is used for the hardest negative mining, where the hardest sample is defined as the closest point in the $\mathcal{L}_2$ normalized feature space to a positive pair. $[x]_+$ denotes function $\max(0, x)$. $m_p = 0.1$ and $m_n = 1.4$ are margins for positive and negative pairs.

PointInfoNCE Loss Here we propose an alternative loss design for PointContrast. InfoNCE proposed in [42] is widely used in recent unsupervised representation learning approaches for 2D visual understanding. By modeling the contrastive learning framework as a dictionary look-up process [23], InfoNCE poses contrastive learning as a classification problem and is implemented with a Softmax loss. Specifically, the loss encourages a query q to be similar to its positive key k^+ and dissimilar to, typically many, negative keys k^-. One challenge in 2D is to scale the number of negative keys [23].

However, in the domain of 3D, we have a different problem: usually the real-world 3D datasets are much smaller in terms of instance count, but the number of points for each instance (*e.g.* a indoor or outdoor scene) can be huge, *i.e.* 100K+ points even from one RGB-D frame. This unique property of 3D data property, together with the original motivation to modelling point level information, inspire us to propose the following PointInfoNCE loss:

$$\mathcal{L}_c = -\sum_{(i,j)\in\mathcal{P}} \log \frac{\exp(\mathbf{f}_i \cdot \mathbf{f}_j/\tau)}{\sum_{(\cdot,k)\in\mathcal{P}} \exp(\mathbf{f}_i \cdot \mathbf{f}_k/\tau)}$$

Here $\mathcal{P}$ is the set of all the positive matches from two views. In this formulation, we only consider points that have at least one match and do not use additional non-matched points as negatives. For a matched pair $(i, j) \in \mathcal{P}$, point feature $\mathbf{f}_i^1$ will serve as the query and $\mathbf{f}_j^2$ will serve as the positive key k^+. We use point feature $\mathbf{f}_k^2$ where $\exists(\cdot, k) \in \mathcal{P}$ and $k \neq j$ as the set of negative keys. In practice, we sample a subset of 4096 matched pairs from $\mathcal{P}$ for faster training.

Compared to hardest-contrastive loss, the PointInfoNCE loss has a simpler formulation with fewer hyperparameters. Perhaps more importantly, due to a large number of negative distractors, it is more robust against *mode collapsing* (features collapsed to a single vector) than the hardest-contrastive loss. In our experiments, we find that hard-contrastive loss is unstable and hard to train: the representation often collapses with extended training epochs (which is also observed in FCGF [10]).

3.5 A Sparse Residual U-Net as Shared Backbone

We use a Sparse Residual U-Net (SR-UNet) architecture in this work. It is a 34-layer U-Net [51] architecture that has an encoder network of 21 convolution layers and a decoder network of 13 convolution/deconvolution layers. It follows the 2D ResNet basic block design and each conv/deconv layer in the network is followed by Batch Normalization (BN) [31] and ReLU activation. The overall U-Net architecture has 37.85M parameters. We provide more information and a visualization of the network in Appendix. The SR-UNet architecture was originally designed in [9] that achieved significant improvement over prior methods on the challenging ScanNet semantic segmentation benchmark. In this work, we explore if we can use this architecture as a unified design for both the pre-training task and a diverse set of fine-tuning tasks.

3.6 Dataset for Pre-training

For local geometric feature learning approaches, including FCGF [10], training and evaluation are typically conducted on domain and task-specific datasets such as KITTI odometry [17] or 3DMatch [81]. Common registration datasets are typically constrained in either scale (training samples collected from just dozens of scenes), or generality (focusing on one specific application scenario, *e.g.* indoor scenes or LiDAR scans for self-driving cars), or both. To facilitate future research on 3D unsupervised representation learning, in our work we utilize the ScanNet dataset for pre-training, aiming to address the scale issue. ScanNet is a collection of ∼1500 indoor scenes. Created with a light-weight RGB-D scanning procedure, ScanNet is currently the largest of its kind.[3]

[3] Admittedly, ScanNet is still much smaller in scale compared to 2D datasets.

Here we create a point cloud pair dataset on top of ScanNet for the pre-training framework shown in Figure 2. Given a scene $\mathbf{x}$, we extract pairs of partial scans $\mathbf{x}^1$ and $\mathbf{x}^2$ from different views. More precisely, for each scene, we first sub-sample RGB-D scans from the raw ScanNet videos every 25 frames, and align the 3D point clouds in the same world coordinates (by utilizing estimated camera poses for each frame). Then we collect point cloud pairs from the sampled frames and require that two point clouds in a pair have at least a 30% overlap. We sample a total number of 870K point cloud pairs. Since the partial views are aligned in ScanNet scenes, it is straightforward to compute the correspondence mapping M between two views with nearest neighbor search.

Although ScanNet only captures indoor data distributions, as we will see in Section 4.4, surprisingly it can generalize to other target distributions. We provide additional visualizations for the pre-training dataset in Appendix.

4 Fine-tuning on Downstream Tasks

The most important motivation for representation learning is to learn features that can transfer well to different downstream tasks. There could be different evaluation protocols to measure the usefulness of the learned representation. For example, probing with a linear classifier [19], or evaluating in a semi-supervised setup [27]. The *supervised fine-tuning* strategy, where the pre-trained weights are used as the initialization and are further refined on the target downstream task, is arguably the most practically meaningful way of evaluating feature transferability. with this setup, good features could directly lead to performance gains in downstream tasks.

Under this perspective, in this section we perform extensive evaluations of the effectiveness of PointContrast framework by fine-tuning the pre-trained weights on multiple downstream tasks and datasets. We aim to cover a diverse suite of high-level 3D understanding tasks of different natures such as semantic segmentation, object detection and classification. In all experiment, we use the same backbone network, pre-trained on the proposed ScanNet pair dataset (Section 3.6) using both PointInfoNCE and Hardest-Constrastive objectives.

4.1 ShapeNet: Classification and Part Segmentation

Setup. In Section 3.1 we have observed that weights learned on supervised ShapeNet classification are not able to transfer well to scene-level tasks. Here we explore the opposite direction: Are PointContrast features learned on ScanNet useful for tasks on ShapeNet? To recap, ShapeNet [7] is a dataset of synthetic 3D objects of 55 common categories. It was curated by collecting CAD models from online open-sourced 3D repositories. In [77], part annotations were added to a subset of ShapeNet models segmenting them into 2-5 parts. In order to provide a comparison with existing approaches, here we utilize the ShapeNetCore dataset (SHREC 15 split) for classification, and the ShapeNet part dataset for part segmentation, respectively. We uniformly sample point clouds of 1024 points

evaluating on all 55 classes	1% data	10% data	100% data
Trained from scratch	62.2	77.9	85.1
PointContrast (Hardest-Contrastive)	**66.2** (+4.0)	**79.0** (+1.1)	**85.7** (+0.6)
PointContrast (PointInfoNCE)	**65.8** (+3.6)	**78.8** (+0.9)	**85.7** (+0.6)
using 100% training data	10 tail classes	30 tail classes	all 55 classes
Train from scratch	65.0	70.9	85.1
PointContrast (Hardest-Contrastive)	**70.9** (+5.9)	**72.9** (+2.0)	**85.7** (+0.6)
PointContrast (PointInfoNCE)	**67.8** (+2.8)	**72.0** (+1.1)	**85.7** (+0.6)

Table 2: **ShapeNet classification.** Top: classification accuracy with limited labeled training data for finetuning. Bottom: classification accuracy on the least represented classes in the data (tail-classes). In all cases, PointContrast boosts performance. Relative improvement increases with scarcer training data and on less frequent classes.

methods	IoU (1% data)	IoU (5% data)	IoU (100% data)
SO-Net[36]	64.0	69.0	-
PointCapsNet[86]	67.0	70.0	-
Multitask Unsupervised[22]	68.2	77.7	-
Train from scratch	71.8	79.3	84.7
PointContrast (Hardest-Contrastive)	**74.0** (+2.2)	**79.9** (+0.6)	**85.1** (+0.4)
PointContrast (PointInfoNCE)	**73.1** (+1.3)	**79.9** (+0.6)	**85.1** (+0.4)

Table 3: **ShapeNet part segmentation.** Replacing the backbone architecture with SR-UNet already boosts performance. PointContrast pre-training further adds a significant gain, and outshines where labels are most limited.

from each model for classification and 2048 points for part segmentation. Albeit containing overlapping indoor object categories with ScanNet, this dataset is substantially different as it is synthetic and contains only single objects. We also follow recent works on 3D unsupervised representation learning [22] to explore a more challenging setup: using a very small percentage (*e.g.* 1%-10%) of training data to fine-tune the pre-trained model.

Results. As shown in Table 2 and Table 3, for both datasets, the effectiveness of pre-training are correlated with the availability of training data. In the ShapeNet classification task (Table 2), pre-training helps most where less training data is available, achieving a 4.0% improvement over the training-from-scratch baseline with the hardest-negative objective. We also note that ShapeNet is a class-imbalanced dataset and the minority (tail) classes are very infrequent. When using 100% of the training data, pre-training provides a class-balancing effect, as it boosts performance more on underrepresented (tail) classes. Table 3 shows a similar effects of pre-training on part segmentation performance. Notably, using SR-UNet backbone architecture already boosts performance; yet, pre-training is able to provide further gains, especially when training data is scarce.

4.2 S3DIS Segmentation

methods	mIoU	mAcc
PointNet [47]	41.1	49.0
PointCNN [37]	57.3	63.9
MinkowskiNet32 [9]	65.4	71.7
Train from scratch	68.2	75.5
PointConstrast (Hardest-Contrastive)	**70.9** (+2.7)	**77.0** (+1.5)
PointConstrast (PointInfoNCE)	70.3 (+2.1)	76.9 (+1.4)

Table 4: **Stanford Area 5 Test (Fold 1) (S3DIS).** Replacing the backbone network with SR-UNet improves upon prior art. Using PointContrast adds further significant boost with a mild preference for Hardest-contrastive over the PointInfoNCE objective. See Appendix for more methods in comparison.

Setup. Stanford Large-Scale 3D Indoor Spaces (S3DIS) [2] dataset comprises 3D scans of 6 large-scale indoor areas collected from 3 office buildings. The scans are represented as point clouds and annotated with semantic labels of 13 object categories. Among the datasets used here for evaluation S3DIS is probably the most similar to ScanNet. Transferring features to S3DIS represents a typical scenario for fine-tuning: the downstream task dataset is similar yet much smaller than the pre-training dataset. For the commonly used benchmark split ("Area 5 test"), there are only about 240 samples in the training set. We follow [9] for pre-processing, and use standard data augmentations. See Appendix for details.

Results. Results are summarized in Table 4. Again, merely switching the SR-UNet architecture, training from scratch already improves upon prior art. Yet, fine-tuning the features learned by PointContrast achieves markedly better segmentation results in mIoU and mAcc. Notably, the effect persists across both loss types, achieving a 2.7% mIoU gain using Hardest-Contrastive loss and an on-par improvement of 2.1% mIoU for the PointInfoNCE variant.

4.3 SUN RGB-D Detection

Setup. We now attend to a different high-level 3D understanding task: object detection. Compared to segmentation tasks that estimate point labels, 3D object detection predicts 3D bounding boxes (localization) and their corresponding object labels (recognition). This calls for an architectural modification as the SR-UNet architecture does not directly output bounding box coordinates. Among many different choices [79, 29, 46, 44], we identify the recently proposed VoteNet [45] as a good candidate for three main reasons. First, VoteNet is designed to work directly on point clouds with no additional input (e.g. images). Second, VoteNet originally uses PointNet++ [48] as the backbone architecture for feature extraction. Replacing this with a SR-UNet requires a minimal modification, keeping the proposal pipeline intact. In particular, we reuse the same hyperparameters. Third, VoteNet is the current state-of-the-art method that uses geometric features only, rendering an improvement markedly useful. We

methods	input	mAP@0.5	mAP@0.25
VoteNet [45]	Geo	-	57.0
VoteNet [45]	Geo+Height	32.9	57.7
Train from scratch	Geo	31.7	55.6
PointContrast(Hardest-Contrastive)	Geo	34.5 (+2.8)	**57.5** (+1.9)
PointContrast(PointInfoNCE)	Geo	**34.8** (+3.1)	**57.5** (+1.9)

Table 5: **SUN RGB-D detection results.** PointContrast demonstrates a substantial boost compared to training from scratch. We observe a larger improvement in localization as manifested by the ΔmAP being larger for @0.5 than @0.25.

methods	mIoU	mAcc
MinkowskiNet32 [9]	78.7	91.5
Train from scratch	79.8	91.5
PointContrast (Hardest-Contrastive)	82.6 (+2.8)	**93.7** (+2.2)
PointContrast (PointInfoNCE)	**83.1** (+3.3)	**93.7** (+2.2)

Table 6: **Segmentation results on the 4D Synthia test set.** All networks here are SR-UNet with 3D kernels, trained on individual 3D frames without temporal modeling.

evaluate the detection performance on the SUN RGB-D dataset [57], a collection of single view RGB-D images. The train set contains 5K images annotated with amodal, 3D oriented bounding boxes for objects from 37 categories.

Results. We summarize the results in Table 5. We find that by simply switching in the backbone network, our baseline result (training from scratch) with the SR-UNet architecture achieves worse results (-1.4% mAP@0.25). This may be attributed to the fact that VoteNet design and hyperparameter settings were tailored to its PointNet++ backbone. However, PointContrast gracefully closes the gap by showing a +3.1% gain on mAP@0.5, which also sets a new state-of-the-art in this metric. The performance gain with a harder evaluation metric (mAP@0.5) suggests that PointContrast pre-training can greatly help localization.

4.4 Synthia4D Segmentation

Setup. Synthia4D [52] is a large synthetic dataset designed to facilitate the training of deep neural networks for visual inference in driving scenarios. Photo-realistic renderings are generated from a virtual city, allowing dense and precise annotations of 13 semantic classes, together with pixel-accurate depth. We follow the train/val/test split as prescribed by [9] in the clean setting. In the context of this work, Synthia4D is especially interesting since it is probably the most distant from our pre-training set (outdoor v.s. indoor, synthetic v.s. real). We test the segmentation performance using 3D SR-UNet on a per-frame basis.

Results. PointContrast pre-training brings substantial improvement over the baseline model trained from scratch (+2.3% mIoU) as seen in Table 6. PointInfoNCE performs noticeably better than the hardest-contrastive loss. With unsupervised pre-training, the overall results are much better than the previous

methods	mIoU	mAcc
Train from scratch	72.2	80.7
PointContrast(Hardest-Contrastive)	73.3 (+1.1)	81.0 (+0.3)
PointContrast(PointInfoNCE)	**74.1** (+1.9)	**81.6** (+0.9)

Table 7: **Segmentation results on ScanNet validation set.** PointContrast boosts performance on the "in-domain" transfer task where the pre-training and fine-tuning datasets come from a common source, showing the usefulness of pre-training even when labels are available.

methods	input	mAP@0.5	mAP@0.25
DSS [58, 29]	Geo+RGB	6.8	15.2
3D-SIS [29]	Geo+RGB (5 Views)	22.5	40.2
VoteNet [45]	Geo+Height	33.5	58.6
Train from scratch	Geo	35.4	56.7
PointContrast(Hardest-Contrastive)	Geo	37.3 (+1.9)	**59.2** (+2.5)
PointContrast(PointInfoNCE)	Geo	**38.0** (+2.6)	58.5 (+1.8)

Table 8: **3D object detection results on ScanNet validation set.** Similarly to in-domain *segmentation* task, here as well PointContrast boost performance on *detection*, setting a new best result over prior art. See Appendix for more methods in comparison.

state-of-the-art reported in [9]. Note that in [9] it has been shown that adding the temporal learning (*i.e.* using a 4D network instead of a 3D one) brings additional benefit. To use 3D pre-trained weights for a 4D network with an additional temporal dimension, we can simply inflate the convolutional kernels, following the standard practice in 2D video recognition [6]. We leave it as future work.

4.5 ScanNet: Segmentation and Detection

Setup. Although typically the source dataset for pre-training and the target dataset for fine-tuning are different, because of the specific multi-view contrastive learning pipeline for pre-training, PointContrast can likely learn different representations (*e.g.* invariance/equivariance to rigid transformations or robustness to noise) compared to directly training with supervision. Thus it is interesting to see whether the pre-trained weights can further improve the results on ScanNet itself. We use ScanNet semantic segmentation and object detection tasks to test our hypothesis. For the segmentation experiment, we use the SR-UNet architecture to directly predict point labels. For the detection experiment, we again follow VoteNet [45] and simply switch the original backbone network with the SR-UNet without other modifications to the detection head (See Appendix for details).

Results. Results are summarized in Table 7 and Table 8. Remarkably, on both detection and segmentation benchmark, models pre-trained with PointContrast outperform those trained from scratch. Notably, PointInfoNCE objective performs better than the Hardest-contrastive one, achieving a relative improvement of +1.9% in terms of segmentation mIoU and +2.6% in terms of detection mAP@0.5. Similar to SUN RGB-D detection, here we also observe that PointContrast features help most for localization as indicated by the larger margin of improvement for mAP@0.5 than mAP@0.25.

4.6 Analysis Experiments and Discussions

In this section, we show additional experiments to provide more insights on our pre-training framework. We use S3DIS segmentation for the experiments below.

Supervised pre-training. While the focus of this work is unsupervised pre-training, a natural baseline is to compare against supervised pre-training. To this end, we use the training-from-scratch baseline for the segmentation task on ScanNetV2 and fine-tune the network on S3DIS. This yields an mIoU of 71.2%, which is only 0.3% better than PointContrast unsupervised pre-training. We deem this a very *encouraging signal* that suggests that the gap between supervised and unsupervised representation learning in 3D has been mostly closed (*cf.* years of effort in 2D). One might argue that this is due to the limited quality and scale of ScanNet, but even at this scale the amount of labor involved in annotating thousands of rooms is large. The outcome of this complements the conclusion we had so far: not only should we put resources into creating large-scale 3D datasets for pre-training; but if facing a trade-off between scaling the data size and annotating it, we should favor the former.

Fine-tuning vs from-scratch under longer training schedule. A recent study in 2D vision [24] suggests that simply by training from scratch for more epochs might close the gap from ImageNet pre-training. We conduct additional experiments to train the network from scratch with 2× and 3× schedules on S3DIS, relative to the 1× schedule of our default setup (10K iterations with batch size 48). We found that validation mIoU does not improve with longer training. In fact, the model exhibits overfitting due to the small dataset size, achieving 66.7% and 66.1% mIoU at 20K and 30K iteration, respectively. This suggests that potentially many of the 3D datasets could fall into the "breakdown regime"[24] where network pre-training is essential for good performance.

Holistic scene as a single view for PointContrast. To show that the multi-view design in PointContrast is important, we try a different variant where instead of having partial views $\mathbf{x}^1$ and $\mathbf{x}^2$, we directly use the reconstructed point cloud $\mathbf{x}$ (a full scene in ScanNet) PointContrast. We still apply independent transformations T_1 and T_2 to the same $\mathbf{x}$. We tried different variants and augmentations such as random cropping, point jittering, and dropout. We also tried different transformations for T_1 and T_2 of different degrees of freedom. However, with the best configuration we can get a validation mIoU on S3DIS of 68.35, which is just slightly better than the training from scratch baseline of 68.17. This suggests that the multi-view setup in PointContrast is critical. Potential reasons include: much more abundant and diverse training samples; natural noise due to camera instability as good regularization, as also observed in [81].

5 Conclusions

We have demonstrated an extensive evaluation of the transferability of learned representations in 3D point clouds to high-level 3D understanding tasks. With the help of our unsupervised pre-training framework PointContrast, we achieve state-of-the-art results across 6 different benchmarks and demonstrate that the learned representation can generalize across domains. We hope these findings will encourage more research on 3D representation learning.

Acknowledgements. The authors would like to thank Chris Choy for his help in setting up the segmentation experiments with MinkowskiEngine. O.L. and L.G. were supported in part by NSF grant IIS-1763268, a Vannevar Bush Faculty Fellowship, and a grant from the SAIL-Toyota Center for AI Research.

References

1. Achlioptas, P., Diamanti, O., Mitliagkas, I., Guibas, L.: Learning representations and generative models for 3D point clouds. arXiv preprint arXiv:1707.02392 (2017)
2. Armeni, I., Sener, O., Zamir, A.R., Jiang, H., Brilakis, I., Fischer, M., Savarese, S.: 3D semantic parsing of large-scale indoor spaces. In: ICCV (2016)
3. Bachman, P., Hjelm, R.D., Buchwalter, W.: Learning representations by maximizing mutual information across views. In: NeurIPS (2019)
4. Boulch, A.: Convpoint: continuous convolutions for point cloud processing. Computers & Graphics (2020)
5. Caron, M., Bojanowski, P., Joulin, A., Douze, M.: Deep clustering for unsupervised learning of visual features. In: ECCV (2018)
6. Carreira, J., Zisserman, A.: Quo vadis, action recognition? a new model and the kinetics dataset. In: CVPR (2017)
7. Chang, A.X., Funkhouser, T., Guibas, L., Hanrahan, P., Huang, Q., Li, Z., Savarese, S., Savva, M., Song, S., Su, H., Xiao, J., Yi, L., Yu, F.: ShapeNet: An Information-Rich 3D Model Repository. arXiv preprint arXiv:1512.03012 (2015)
8. Chen, T., Kornblith, S., Norouzi, M., Hinton, G.: A simple framework for contrastive learning of visual representations. arXiv preprint arXiv:2002.05709 (2020)
9. Choy, C., Gwak, J., Savarese, S.: 4D spatio-temporal convnets: Minkowski convolutional neural networks. In: CVPR (2019)
10. Choy, C., Park, J., Koltun, V.: Fully convolutional geometric features. In: ICCV (2019)
11. Dai, A., Chang, A.X., Savva, M., Halber, M., Funkhouser, T., Nießner, M.: Scannet: Richly-annotated 3D reconstructions of indoor scenes. In: CVPR (2017)
12. Deng, H., Birdal, T., Ilic, S.: Ppf-foldnet: Unsupervised learning of rotation invariant 3D local descriptors. In: ECCV (2018)
13. Devlin, J., Chang, M.W., Lee, K., Toutanova, K.: BERT: Pre-training of deep bidirectional transformers for language understanding. In: NAACL (2019)
14. Doersch, C., Gupta, A., Efros, A.A.: Unsupervised visual representation learning by context prediction. In: ICCV (2015)
15. Elbaz, G., Avraham, T., Fischer, A.: 3D point cloud registration for localization using a deep neural network auto-encoder. In: CVPR (2017)
16. Gadelha, M., Wang, R., Maji, S.: Multiresolution tree networks for 3D point cloud processing. In: ECCV (2018)

17. Geiger, A., Lenz, P., Stiller, C., Urtasun, R.: Vision meets robotics: The kitti dataset. The International Journal of Robotics Research **32**(11), 1231–1237 (2013)
18. Gidaris, S., Singh, P., Komodakis, N.: Unsupervised representation learning by predicting image rotations. In: ICLR (2018)
19. Goyal, P., Mahajan, D., Gupta, A., Misra, I.: Scaling and benchmarking self-supervised visual representation learning. In: ICCV (2019)
20. Graham, B., Engelcke, M., van der Maaten, L.: 3d semantic segmentation with submanifold sparse convolutional networks. In: CVPR (2018)
21. Groueix, T., Fisher, M., Kim, V.G., Russell, B.C., Aubry, M.: A papier-mâché approach to learning 3D surface generation. In: CVPR (2018)
22. Hassani, K., Haley, M.: Unsupervised multi-task feature learning on point clouds. In: ICCV (2019)
23. He, K., Fan, H., Wu, Y., Xie, S., Girshick, R.: Momentum contrast for unsupervised visual representation learning. In: CVPR (2020)
24. He, K., Girshick, R., Dollár, P.: Rethinking imagenet pre-training. In: ICCV (2019)
25. He, K., Gkioxari, G., Dollár, P., Girshick, R.: Mask R-CNN. In: ICCV (2017)
26. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. In: CVPR (2016)
27. 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)
28. Hjelm, R.D., Fedorov, A., Lavoie-Marchildon, S., Grewal, K., Bachman, P., Trischler, A., Bengio, Y.: Learning deep representations by mutual information estimation and maximization. ICLR (2019)
29. Hou, J., Dai, A., Nießner, M.: 3D-SIS: 3D semantic instance segmentation of RGB-D scans. In: CVPR (2019)
30. Hua, B.S., Tran, M.K., Yeung, S.K.: Pointwise convolutional neural networks. In: CVPR (2018)
31. Ioffe, S., Szegedy, C.: Batch normalization: Accelerating deep network training by reducing internal covariate shift. In: ICML (2015)
32. Janoch, A., Karayev, S., Jia, Y., Barron, J.T., Fritz, M., Saenko, K., Darrell, T.: A category-level 3d object dataset: Putting the kinect to work. In: Consumer depth cameras for computer vision (2013)
33. Klokov, R., Lempitsky, V.: Escape from cells: Deep kd-networks for the recognition of 3d point cloud models. In: ICCV (2017)
34. Landrieu, L., Simonovsky, M.: Large-scale point cloud semantic segmentation with superpoint graphs. In: CVPR (2018)
35. Lei, H., Akhtar, N., Mian, A.: Spherical convolutional neural network for 3d point clouds. arXiv preprint arXiv:1805.07872 (2018)
36. Li, J., Chen, B.M., Hee Lee, G.: So-net: Self-organizing network for point cloud analysis. In: CVPR (2018)
37. Li, Y., Bu, R., Sun, M., Wu, W., Di, X., Chen, B.: Pointcnn: Convolution on x-transformed points. In: NeurIPS (2018)
38. Long, J., Shelhamer, E., Darrell, T.: Fully convolutional networks for semantic segmentation. In: CVPR (2015)
39. Maron, H., Litany, O., Chechik, G., Fetaya, E.: On learning sets of symmetric elements. arXiv preprint arXiv:2002.08599 (2020)
40. Misra, I., van der Maaten, L.: Self-supervised learning of pretext-invariant representations. In: CVPR (2020)
41. Noroozi, M., Favaro, P.: Unsupervised learning of visual representations by solving jigsaw puzzles. In: ECCV (2016)

42. Oord, A.v.d., Li, Y., Vinyals, O.: Representation learning with contrastive predictive coding. arXiv preprint arXiv:1807.03748 (2018)
43. Pathak, D., Krahenbuhl, P., Donahue, J., Darrell, T., Efros, A.A.: Context encoders: Feature learning by inpainting. In: CVPR (2016)
44. Qi, C.R., Chen, X., Litany, O., Guibas, L.J.: Imvotenet: Boosting 3D object detection in point clouds with image votes. In: CVPR (2020)
45. Qi, C.R., Litany, O., He, K., Guibas, L.J.: Deep hough voting for 3d object detection in point clouds. ICCV (2019)
46. Qi, C.R., Liu, W., Wu, C., Su, H., Guibas, L.J.: Frustum pointnets for 3d object detection from rgb-d data. In: CVPR (2018)
47. Qi, C.R., Su, H., Mo, K., Guibas, L.J.: Pointnet: Deep learning on point sets for 3d classification and segmentation. CVPR (2017)
48. Qi, C.R., Yi, L., Su, H., Guibas, L.J.: Pointnet++: Deep hierarchical feature learning on point sets in a metric space. NeurIPS (2017)
49. Radford, A., Wu, J., Child, R., Luan, D., Amodei, D., Sutskever, I.: Language models are unsupervised multitask learners. OpenAI Blog **1**(8), 9 (2019)
50. Ravanbakhsh, S., Schneider, J., Poczos, B.: Deep learning with sets and point clouds. arXiv preprint arXiv:1611.04500 (2016)
51. Ronneberger, O., Fischer, P., Brox, T.: U-net: Convolutional networks for biomedical image segmentation. In: International Conference on Medical image computing and computer-assisted intervention. pp. 234–241. Springer (2015)
52. Ros, G., Sellart, L., Materzynska, J., Vazquez, D., Lopez, A.M.: The synthia dataset: A large collection of synthetic images for semantic segmentation of urban scenes. In: CVPR (2016)
53. Sauder, J., Sievers, B.: Self-supervised deep learning on point clouds by reconstructing space. In: NeurIPS (2019)
54. Shen, Y., Feng, C., Yang, Y., Tian, D.: Mining point cloud local structures by kernel correlation and graph pooling. In: CVPR (2018)
55. Silberman, N., Hoiem, D., Kohli, P., Fergus, R.: Indoor segmentation and support inference from RGB-D images. ECCV (2012)
56. Simonyan, K., Zisserman, A.: Very deep convolutional networks for large-scale image recognition. arXiv preprint arXiv:1409.1556 (2014)
57. Song, S., Lichtenberg, S.P., Xiao, J.: Sun rgb-d: A rgb-d scene understanding benchmark suite. In: CVPR (2015)
58. Song, S., Xiao, J.: Deep sliding shapes for amodal 3d object detection in rgb-d images. In: CVPR (2016)
59. Su, H., Jampani, V., Sun, D., Maji, S., Kalogerakis, E., Yang, M.H., Kautz, J.: Splatnet: Sparse lattice networks for point cloud processing. In: CVPR (2018)
60. Tatarchenko, M., Park, J., Koltun, V., Zhou, Q.Y.: Tangent convolutions for dense prediction in 3d. In: CVPR (2018)
61. Tchapmi, L., Choy, C., Armeni, I., Gwak, J., Savarese, S.: Segcloud: Semantic segmentation of 3d point clouds. In: 3DV (2017)
62. Te, G., Hu, W., Zheng, A., Guo, Z.: Rgcnn: Regularized graph cnn for point cloud segmentation. In: ACM Multimedia (2018)
63. Tian, Y., Krishnan, D., Isola, P.: Contrastive multiview coding. arXiv preprint arXiv:1906.05849 (2019)
64. Verma, N., Boyer, E., Verbeek, J.: Feastnet: Feature-steered graph convolutions for 3D shape analysis. In: CVPR (2018)
65. Wang, C., Samari, B., Siddiqi, K.: Local spectral graph convolution for point set feature learning. In: ECCV (2018)

66. Wang, S., Suo, S., Ma, W.C., Pokrovsky, A., Urtasun, R.: Deep parametric continuous convolutional neural networks. In: CVPR (2018)
67. Wang, Y., Solomon, J.M.: Deep closest point: Learning representations for point cloud registration. In: ICCV (2019)
68. Wang, Y., Sun, Y., Liu, Z., Sarma, S.E., Bronstein, M.M., Solomon, J.M.: Dynamic graph cnn for learning on point clouds. ACM TOG **38**(5), 1–12 (2019)
69. Wu, Z., Xiong, Y., Yu, S.X., Lin, D.: Unsupervised feature learning via non-parametric instance discrimination. In: CVPR (2018)
70. Xiao, J., Owens, A., Torralba, A.: SUN3D: A database of big spaces reconstructed using sfm and object labels. In: ICCV (2013)
71. Xie, S., Girshick, R., Dollár, P., Tu, Z., He, K.: Aggregated residual transformations for deep neural networks. In: CVPR (2017)
72. Xie, S., Liu, S., Chen, Z., Tu, Z.: Attentional shapecontextnet for point cloud recognition. In: CVPR (2018)
73. Xu, Y., Fan, T., Xu, M., Zeng, L., Qiao, Y.: Spidercnn: Deep learning on point sets with parameterized convolutional filters. In: ECCV (2018)
74. Yang, Y., Feng, C., Shen, Y., Tian, D.: Foldingnet: Point cloud auto-encoder via deep grid deformation. In: CVPR (2018)
75. Yang, Z., Litany, O., Birdal, T., Sridhar, S., Guibas, L.: Continuous geodesic convolutions for learning on 3D shapes. In: arXiv preprint arXiv:2002.02506 (2020)
76. Ye, X., Li, J., Huang, H., Du, L., Zhang, X.: 3d recurrent neural networks with context fusion for point cloud semantic segmentation. In: ECCV (2018)
77. Yi, L., Kim, V.G., Ceylan, D., Shen, I.C., Yan, M., Su, H., Lu, C., Huang, Q., Sheffer, A., Guibas, L.: A scalable active framework for region annotation in 3D shape collections. SIGGRAPH Asia (2016)
78. Yi, L., Su, H., Guo, X., Guibas, L.J.: Syncspeccnn: Synchronized spectral cnn for 3D shape segmentation. In: CVPR (2017)
79. Yi, L., Zhao, W., Wang, H., Sung, M., Guibas, L.: GSPN: Generative shape proposal network for 3D instance segmentation in point cloud. In: CVPR (2019)
80. Zaheer, M., Kottur, S., Ravanbakhsh, S., Poczos, B., Salakhutdinov, R.R., Smola, A.J.: Deep sets. In: NeurIPS (2017)
81. Zeng, A., Song, S., Nießner, M., Fisher, M., Xiao, J., Funkhouser, T.: 3DMatch: Learning local geometric descriptors from RGB-D reconstructions. In: CVPR (2017)
82. Zeng, W., Gevers, T.: 3DContextNet: KD tree guided hierarchical learning of point clouds using local and global contextual cues. In: ECCV (2018)
83. Zhang, R., Isola, P., Efros, A.A.: Colorful image colorization. In: ECCV (2016)
84. Zhang, R., Isola, P., Efros, A.A.: Split-brain autoencoders: Unsupervised learning by cross-channel prediction. In: CVPR (2017)
85. Zhang, Z., Hua, B.S., Yeung, S.K.: Shellnet: Efficient point cloud convolutional neural networks using concentric shells statistics. In: ICCV (2019)
86. Zhao, Y., Birdal, T., Deng, H., Tombari, F.: 3D point capsule networks. In: CVPR (2019)
87. Zhuang, C., Zhai, A.L., Yamins, D.: Local aggregation for unsupervised learning of visual embeddings. In: ICCV (2019)

A Visualization of the SR-UNet Architecture

Here we show the SR-UNet architecture that is used as a shared backbone in our paper for both the pre-training and the fine-tuning phases. This U-Net architecture was originally proposed in [9] for ScanNet semantic segmentation.

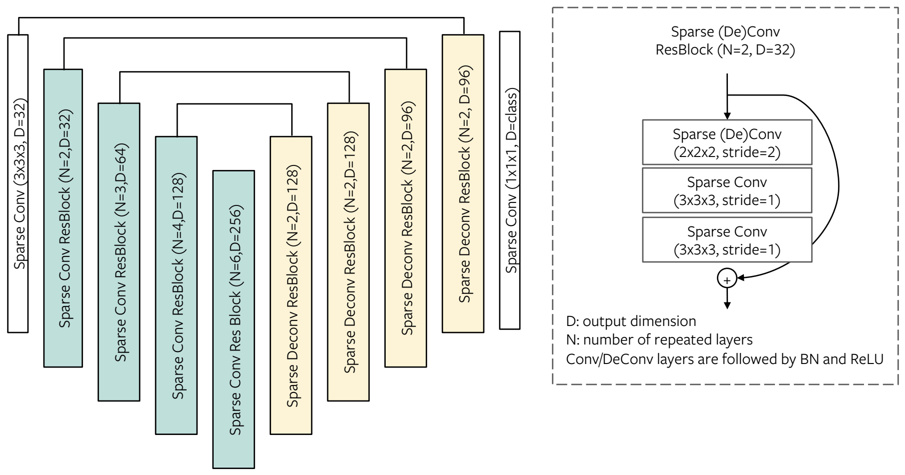

Sparse Conv (3x3x3, D=32)
Sparse Conv ResBlock (N=2,D=32)
Sparse Conv ResBlock (N=3,D=64)
Sparse Conv ResBlock (N=4,D=128)
Sparse Conv Res Block (N=6,D=256)
Sparse Deconv ResBlock (N=2,D=128)
Sparse Deconv ResBlock (N=2,D=128)
Sparse Deconv ResBlock (N=2,D=96)
Sparse Deconv ResBlock (N=2, D=96)
Sparse Conv (1x1x1, D=class)
Sparse (De)Conv ResBlock (N=2, D=32)
Sparse (De)Conv (2x2x2, stride=2)
Sparse Conv (3x3x3, stride=1)
Sparse Conv (3x3x3, stride=1)
+
D: output dimension
N: number of repeated layers
Conv/DeConv layers are followed by BN and ReLU

Fig. 3: SR-UNet architecture we used as a shared backbone network for pre-training and fine-tuning tasks. For segmentation and detection tasks, both the encoder and decoder weights are fine-tuned; for classification downstream tasks, only the encoder network is kept and fine-tuned.

B Visualization of the ScanNet Point Cloud Pair Dataset

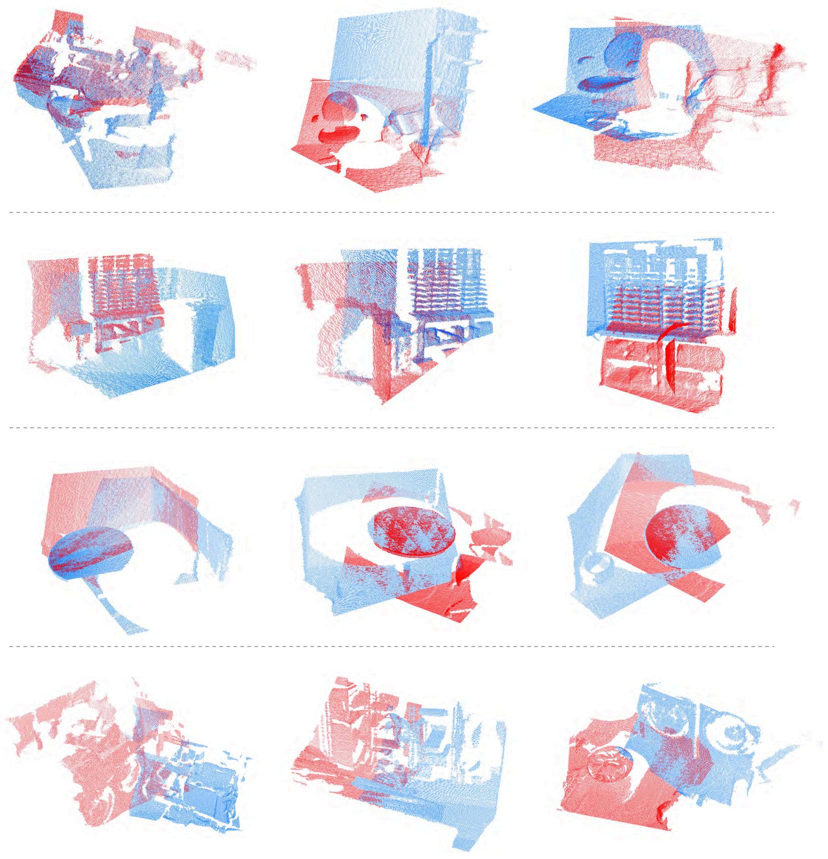


Fig. 4: Visualization of the ScanNet point cloud pair dataset used for pre-training. Each row is a randomly sampled scene. Each column is a different pair of point clouds sampled from the same scene. Different colors are corresponding to two different views (partial scans). At least 30% of the points are overlapping in two views.

C ShapeNet Supervised Training Details

We use a sparse ResNet network that has an identical structure to the encoder part of the SR-UNet in Appendix A. We use Adam optimizer, and add standard data augmentations including rotation, scaling and translation, following [47, 48]. We perform a grid search over the learning rate, weight decay, voxel size (for sparse convolution), and the number of input points. The best performing model configuration is learning rate 0.004, voxel size 0.01, weight decay 1e-5, batch size

512 and 2048 input points. The 85.4% accuracy is to our knowledge the best results that have been reported on this SHREC benchmark split. We use 8 Titan-V100 GPU with data parallelism to train the model. We train the model for 200 epochs and the training takes around 8 hours.

D Details on PointContrast Pre-training

D.1 Details on Transformations

The transformations $\mathbf{T}_1$ and $\mathbf{T}_2$ applied to two views $\mathbf{x}^1$ and $\mathbf{x}^2$ in our experiments involves a random rotation (0 to 360°) along an arbitrary axis (applied independently to both views). We apply scale augmentation to both views (0.8× to 1.2× of the input scale). We have experimented with other augmentations such as translation, point coordinate jittering, and point dropout and did not find noticeable difference in fine-tuning performances.

D.2 Details on Loss Functions

For the hardest-contrastive loss, the positive sample size is 1024 and the hardest negative sample size is 256. More details can be found in [10]. For the PointInfoNCE loss, we provide a detailed PyTorch-like pseudo-code (and explanatory comments) in Algorithm 2.

Algorithm 2 Pseudocode of PointInfoNCE Loss implementation.

```
# f_v1, f_v2: features for matched points (in a minibatch) between view 1 and view 2: NxC
# NN: shared backbone network
# t: temperature
# Ns: subsampling size for point features.

f_v1,inds = random.choice(f_v1, Ns, dim=0) # subsample view 1 point features
f_v2 = f_v2[inds,:] # subsample view 2 point features

logits = torch.mm(f_v1, f_v2.transpose(1, 0)) # Ns by Ns
labels = torch.arange(Ns) # for k-th row, the positive sample is at the k-th position.
loss = CrossEntropyLoss(logits/t, labels)

# SGD update: shared backbone network
loss.backward()
update(NN.params)
```

`mm`: matrix multiplication;

E S3DIS Segmentation Experimental Details

Here we provide training details for S3DIS semantic segmentation task. We use the widely adopted Area 5 Test (Fold 1) split for training and testing. For all the PointContrast variants (Training from scratch, Hardest-contrastive Pretrained, and PointInfoNCE Pretrained) we use the same hyperparameter settings. Specifically we train the model with 8 V100 GPUs with data parallelism for 10,000 iterations. Batch size is 48. Batch normalization is applied independently on each GPU. We use SGD+momentum optimizer with an initial learning rate 0.8. We use Polynomial LR scheduler with a power factor of 0.9. Weight decay is 0.0001 and voxel size is 0.05 (5cm). We use the same data augmentation techniques in [9] such as color hue/saturation augmentation and jittering, as well as scale augmentations (0.9× to 1.1×). In Table 9 we show detailed per-category performance breakdown for our models and previous approaches.

Method	ceiling	floor	wall	beam	clmn	windw	door	chair	table	bkcase	sofa	board	clutter	mIOU	mAcc
PointNet [47]	88.80	97.33	69.80	0.05	3.92	46.26	10.76	52.61	58.93	40.28	5.85	26.38	33.22	41.09	48.98
SegCloud [61]	90.06	96.05	69.86	0.00	18.37	38.35	23.12	75.89	70.40	58.42	40.88	12.96	41.60	48.92	57.35
TangentConv [60]	90.47	97.66	73.99	0.0	20.66	38.98	31.34	77.49	69.43	57.27	38.54	48.78	39.79	52.8	60.7
3D RNN [76]	95.2	98.6	77.4	0.8	9.83	52.7	27.9	76.8	78.3	58.6	27.4	39.1	51.0	53.4	71.3
PointCNN [37]	92.31	98.24	79.41	0.00	17.60	22.77	62.09	80.59	74.39	66.67	31.67	62.05	56.74	57.26	63.86
SuperpointGraph [34]	89.35	96.87	78.12	0.0	42.81	48.93	61.58	84.66	75.41	69.84	52.60	2.1	52.22	58.04	66.5
MinkowskiNet20 [9]	91.55	98.49	84.99	0.8	26.47	46.18	55.82	88.99	80.52	71.74	48.29	62.98	57.72	62.60	69.62
MinkowskiNet32 [9]	91.75	98.71	86.19	0.0	34.06	48.90	62.44	89.82	81.57	**74.88**	47.21	74.44	58.57	65.35	71.71
PntContrast(Scratch)	91.47	98.56	84.08	0.00	33.03	56.88	63.94	90.11	81.67	72.46	76.45	77.89	59.63	68.17	75.45
PntContrast(Hardest-Ctr)	**94.82**	**98.72**	**86.06**	0.00	42.84	**58.00**	**73.72**	**91.73**	**82.38**	74.74	74.58	81.42	**62.66**	**70.90**	**77.00**
PntContrast(Pnt-InfoNCE)	93.26	98.67	85.56	0.11	**45.90**	54.41	67.87	91.56	80.09	74.66	**78.20**	**81.49**	62.32	70.32	76.94

Table 9: **Stanford Area 5 Test (Fold 1).** Per-category IOU performance.

F Synthia4D Segmentation Experimental Details

Here we provide training details for Synthia4D semantic segmentation task. As mentioned in the main paper, we only use 3D sparse convnet without any temporal aggregation mechanisms such as 4D kernels and temporal CRF. For all the PointContrast variants (Training from scratch, Hardest-contrastive Pretrained, and PointInfoNCE Pretrained) we use the same hyperparameter settings, and those are mostly identical the S3DIS experiments. Specifically we train the model with 8 V100 GPUs with data parallelism for 15,000 iterations. Batch size is 72. Batch normalization is applied independently on each GPU. We use SGD+momentum optimizer with an initial learning rate 0.8. We use Polynomial LR scheduler with a power factor of 0.9. Weight decay is 0.0001 and voxel size is 0.05 (5cm). We also use the same data augmentation techniques in [9] in color space and point coordinate space. In Table 10 we show detailed per-category performance breakdown for our models and results reported in [9].

G ScanNet Segmentation Experimental Details

For ScanNet segmentation task, we train the model with 8 V100 GPUs with data parallelism for 15,000 iterations. Batch size is 48. We use SGD+momentum

Method	Bldn	Road	Sdwlk	Fence	Vegittn	Pole	Car	T. Sign	Pedstrn	Bicycl	Lane	T. Light	mIoU	mAcc
MinkNet20 + TA [9]	88.096	97.790	78.329	87.088	96.540	97.486	94.203	78.831	92.489	0.000	46.407	67.071	77.03	89.198
4D MinkNet32 + TS-CRF [9]	89.694	98.632	86.893	87.801	98.776	97.284	94.039	80.292	92.300	0.000	49.299	69.060	78.67	90.51
PntContrast(Train from scratch)	92.237	98.619	90.217	86.863	99.346	96.848	95.085	75.526	88.596	0.000	72.173	62.060	79.797	91.492
PntContrast(Hardest-Contrastive)	**92.518**	**99.040**	93.309	87.331	99.384	97.500	**96.174**	81.627	92.007	0.000	**80.257**	71.764	82.576	93.650
PntContrast(PointInfoNCE)	92.238	99.006	**93.993**	**87.368**	**99.657**	**97.755**	95.648	**83.446**	**93.279**	0.000	79.002	**76.364**	**83.146**	**93.707**

Table 10: **Synthia4D segmentation test results** Per-category IOU performance.

optimizer with an initial learning rate 0.8. We use Polynomial LR scheduler with a power factor of 0.9. Weight decay is 0.0001 and voxel size is 0.025 (2.5cm). We also use the same data augmentation techniques in [9] in color space and point coordinate space. In Table 10 we show detailed per-category performance breakdown for our models and results reported in [9].

Method	bath	bed	bookshelf	cab	chair	counter	curtain	desk	door	floor	other	pic	ref	shower	sink	sofa	tab	toilet	wall	wind	mIoU	mAcc
Scratch	84.866	96.412	64.271	80.928	90.910	**85.954**	73.890	61.767	59.935	80.753	30.896	**68.571**	62.109	75.252	54.603	67.236	90.084	**68.720**	85.936	60.279	72.169	80.718
Hardest-Ct	85.773	96.442	**66.883**	**81.525**	**91.829**	84.708	74.030	65.149	62.835	**82.645**	32.113	66.451	64.408	77.507	53.049	**69.762**	94.269	67.934	**88.601**	60.190	73.305	81.025
PntInfoNCE	**86.540**	**96.456**	66.630	81.294	91.301	84.281	**77.393**	**68.031**	**65.168**	81.600	**33.530**	66.037	**67.639**	**77.803**	**56.853**	69.398	**95.202**	68.329	88.303	**60.924**	**74.136**	**81.623**

Table 11: **ScanNet segmentation results on val set** Per-category IOU performance.

H ScanNet and SUN RGB-D Detection Details

For the 3D object detection experiments, we mostly follow the configurations in VoteNet [45] framework after switching in the SR-UNet backbone architecture. We train the model on 1 GPU, with batch size 64 for SUN RGB-D and 32 for ScanNet. Learning rate is 0.001 and we use Adam optimizer. The input points are subsampled before voxelization, we use 20000 points for SUN RGB-D and 40000 points for ScanNet. The voxel size is 2.5cm for ScanNet and 5cm for SUN RGB-D. In Table 15 we show more results reported by previous methods. In Table 12 and Table 13, we show per-category AP performance for PointContrast models agains training from scratch results, under AP@0.5 metric.

Methods	bed	table	sofa	chair	toil	desk	dress	night	book	bath	mAP@0.5
Train from Scratch	47.8	19.6	48.1	54.6	60.0	6.3	15.8	27.3	5.4	32.1	31.7
Hardest-Contrastive	**52.0**	**20.1**	**52.3**	**55.8**	**60.0**	**7.5**	14.7	36.8	**10.0**	35.6	34.5
PointInfoNCE	50.5	19.4	51.8	54.9	57.4	**7.5**	**16.2**	**37.0**	5.9	**47.6**	**34.8**

Table 12: **SUN RGB-D detection results** Per-category AP@0.5 performance.

Methods	cabinet	bed	chair	sofa	table	door	wind	bkshlf	pic	cntr	desk	curtain	refrig	shower	toilet	sink	bath	garbage	mAP@0.5
Train from Scratch	9.9	70.5	70.0	60.5	43.4	**21.8**	10.5	33.3	0.8	15.4	33.3	26.6	**39.3**	9.7	74.7	23.7	75.8	18.1	35.4
Hardest-Contrastive	10.5	68.4	**75.6**	59.1	43.1	19.6	9.6	**35.0**	**2.1**	15.6	**34.3**	**32.8**	37.8	13.6	76.9	**28.8**	**82.4**	25.8	37.3
PointInfoNCE	**13.1**	**74.7**	75.4	**61.3**	**44.8**	19.8	**12.9**	32.0	0.9	**21.9**	31.9	27.0	32.6	**17.5**	**87.4**	23.2	80.8	**26.7**	**38.0**

Table 13: **ScanNet detection results** Per-category AP@0.5 performance.

I PointContrast vs FCGF for low- and high-level tasks

We take the best performing FCGF model released in [10] that achieves a high registration feature matching recall (FMR) of: 0.958. However, this model does not perform well for S3DIS segmentation. On the other hand, the PointContrast model that performs best for segmentation achieves a lower FMR when applied to the registration task. We conclude that low-level tasks and high-level tasks in 3D might require different design choices.

methods	Registration FMR	S3DIS mIoU
FCGF[9]	**0.958**	63.06
PointContrast	0.912	**70.90**

Table 14: **FCGF vs PointContrast.** FCGF achieves a much higher registration feature matching recall, while PointContrast achieves higher mIoU for segmentation.

methods	input	mAP@0.5	mAP@0.25
DSS [58, 29]	Geo+RGB	6.8	15.2
MRCNN 2D-3D [25, 29]	Geo+RGB	10.5	17.3
F-PointNet [46, 29]	Geo+RGB	10.8	19.8
GSPN [79]	Geo+RGB	17.7	30.6
3D-SIS [29]	Geo+RGB (5 Views)	22.5	40.2
VoteNet [45]	Geo+Height	33.5	58.6
Training from scratch	Geo	35.4	56.7
PointContrast(Hardest-Contrastive)	Geo	37.3	**59.2**
PointContrast(PointInfoNCE)	Geo	**38.0**	58.5

Table 15: **3D object detection results on ScanNet dataset.** More methods in comparison.