Implicit Autoencoder for Point Cloud Self-supervised Representation Learning

Siming Yan[1] Zhenpei Yang[1] Haoxiang Li[2] Li Guan[2] Hao Kang[2]
Gang Hua[2] Qixing Huang[1]

[1]The University of Texas at Austin [2]Wormpex AI Research

Abstract. Many 3D representations (e.g., point clouds) are discrete samples of the underlying continuous 3D surface. This process inevitably introduces sampling variations on the underlying 3D shapes. In learning 3D representation, the variations should be disregarded while transferable knowledge of the underlying 3D shape should be captured. This becomes a grand challenge in existing representation learning paradigms. This paper studies autoencoding on point clouds. The standard autoencoding paradigm forces the encoder to capture such sampling variations as the decoder has to reconstruct the original point cloud that has sampling variations. We introduce Implicit Autoencoder(IAE), a simple yet effective method that addresses this challenge by replacing the point cloud decoder with an implicit decoder. The implicit decoder outputs a continuous representation that is shared among different point cloud sampling of the same model. Reconstructing under the implicit representation can prioritize that the encoder discards sampling variations, introducing more space to learn useful features. We theoretically justify this claim under a simple linear autoencoder. Moreover, the implicit decoder offers a rich space to design suitable implicit representations for different tasks. We demonstrate the usefulness of IAE across various self-supervised learning tasks for both 3D objects and 3D scenes. Experimental results show that IAE consistently outperforms the state-of-the-art in each task. Our code will be available at https://github.com/SimingYan/IAE.

1 Introduction

Point cloud provides a natural and flexible representation of 3D objects. The rapid development of 3D scanning devices and techniques enables the capture and access of massive amounts of point cloud data. With the emergence of powerful deep learning models, we are now able to obtain promising performance on many point cloud tasks, ranging from object-level understanding, including shape classification [5] and part segmentation [55], to scene-level understanding, such as 3D object detection [8, 12, 43] and 3D semantic segmentation [2]. While these applications are important, manually annotating large-scale point cloud data can be very costly due to difficulties in designing 3D interfaces and visualizing point clouds. Because of this, there are growing interests in exploring self-supervised representation learning on point cloud data.

Generally speaking, self-supervised representation learning studies how to effectively utilize raw and unlabeled data to pre-train deep neural networks. The

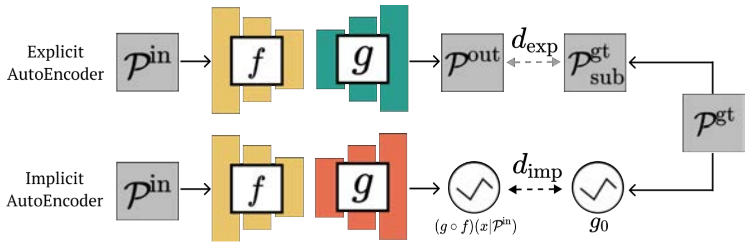
Explicit AutoEncoder
Implicit AutoEncoder
$\mathcal{P}^{\text{in}}$
f
g
$\mathcal{P}^{\text{out}}$
d_{exp}
$\mathcal{P}^{\text{gt}}_{\text{sub}}$
$\mathcal{P}^{\text{gt}}$
d_{imp}
$(g \circ f)(x|\mathcal{P}^{\text{in}})$
g_0

Fig. 1: **Difference between Explicit Autoencoder and Implicit Autoencoder (Ours)**. $\mathcal{P}^{\text{gt}}$ is the complete point cloud. $\mathcal{P}^{\text{gt}}_{\text{sub}}$ is a sub-sampled version of the complete point cloud, that typically contains a fixed number of point (e.g., 2048). $\mathcal{P}^{\text{in}}$ is the input point cloud, which equals to $\mathcal{P}^{\text{gt}}_{\text{sub}}$ unless data augmentation such as cropping(see Section 4.2) is applied. Unlike Explicit Autoencoder that trains the decoder g to recover sub-sampled point cloud $\mathcal{P}^{\text{gt}}_{\text{sub}}$, Implicit Autoencoder forces the decoder g to recover the implicit representation of the complete point cloud $\mathcal{P}^{\text{gt}}$.

pre-trained weights are then transferred and fine-tuned on small-scale annotated data for downstream tasks such as classification and segmentation. The network weights initialized in this way tend to avoid weak local minimums and increase the network's performance stability [10]. Substantial effort has been devoted to self-supervised learning methods for 2D images [6, 9, 25, 34, 51]. Among this line, autoencoder is one of the most classical methods [3, 18, 34, 45, 46]. Typically, it has an encoder that transforms the input into a latent code and a decoder that expands the latent code to reconstruct the input. The latent code usually has a much lower dimension than the input. By training with the reconstruction loss, the autoencoder is forced to learn multi-scale features repeating among the inputs. The performance of such an approach normally depends on the network architecture. Unlike the conventional structured data (e.g., images), point clouds are unordered collections of points. There is increasing literature addressing suitable network architecture and learning algorithms for autoencoding on point clouds. For example, Yang et al. [54] proposed a point cloud autoencoder with a novel folding-based decoder. Wang et al. [47] designed a denoising autoencoder using a standard point cloud completion model. However, these works exclusively follow the design paradigm for image autoencoding, i.e., the decoder and the encoder share the same representation (point cloud in this case) as the input.

We argue that there is a fundamental drawback in using point cloud as the decoder representation for autoencoding. Point clouds are spatially unstructured discretized representations of 3D shapes. Many different point clouds could represent a single 3D shape. The resulting *sampling variations* do not capture information that is useful for 3D understanding. However, a point-based autoencoder forces the encoder to remember such useless variations since the decoder has to reconstruct the original point clouds.

This paper proposes a non-symmetric point cloud autoencoder scheme that uses the implicit function as the output surface representation, dubbed IAE(Implicit Autoencoder). IAE enjoys multiple advantages over traditional point cloud autoencoders. First, the sampling variations problem is addressed using the implicit surface representation. In other words, the implicit decoder outputs a continuous representation that is shared among different point cloud sampling of the same model. Reconstructing under the implicit representation can prioritize that the encoder discards sampling variations, introducing more space to learn useful features. Second, the learning process of IAE is guided by minimizing the discrepancy of two implicit functions, which do not require the explicit computation of data association(e.g., Earth Mover Distance (EMD) [41] or Chamfer Distance (CD) [11]). Moreover, without the need to decode the whole point cloud, our model is smaller and more resource-efficient. Our model can process up to 40k input points in a single Tesla V100 GPU, making it possible to keep necessary details while pre-training on a large real-world point cloud. Our IAE scheme is illustrated in Figure 1.

To demonstrate the usefulness of IAE, we verify that the learned representation from our pre-trained model can be successfully adapted to both object and scene-level understanding tasks, including 3D shape classification, 3D object detection, and indoor scene semantic segmentation. Experimental results show that IAE consistently outperforms the state-of-the-art in each task. Our experiments also show that the performance gains of IAE are persistent when varying the sampling resolution, although the relative numbers slightly drop when increasing the sampling resolution. On the one hand, this is expected as sampling variations reduce when increasing the sampling resolution. On the other hand, the persistent performance gain is attributed to the fact that there are computational challenges in matching point clouds, e.g., convergence and local minimum issues under both EMD and CD, for autoencoding. Such computational challenges amplify for larger point clouds, as the search spaces become more complex.

2 Related Work

Self-supervised Representation Learning on 2D image Self-supervised learning is a well-studied task in computer vision [9, 15, 34, 57, 60]. Most relevant methods are motivated by the observation that high-level semantic features are implicitly correlated with a wide variety of non-semantic "proxy" features. Many these kind of proxy features are accessible by simple image manipulation, including image inpainting [34], colorization [57], jigsaw puzzles [30], rotate prediction [15], and etc. By learning explicitly to predict the proxy feature, the learned representations create representations that implicitly capture higher-level features. Recently, another family of self-supervised learning approaches, contrastive learning [4, 6, 19, 51, 61], has emerged with substantially improved transfer performance. These contrastive embedding objectives optimize networks to discriminate the embeddings of each instance. Typically, they aim at embedding augmented versions of the same sample close to each other while trying to push away embeddings from different samples.

Autoencoding An autoencoder typically contains two parts: an encoder and a decoder. Generally, they work by compressing the input into a low-dimensional latent code and then reconstructing the output from it. The latent code is usually constrained by a much smaller dimension than the input. By training the autoencoder, the latent code is forced to drop input redundancies and preserve useful features. Autoencoding is a classical method for representation learning [25,46], which has been out-performed by contrastive learning approaches for years. However, the recent work in this line, He et al. [18], has reclaimed state-of-the-art performance.

Self-supervised Representation Learning on Point Cloud Unlike conventional structured data (e.g., images), point clouds are unordered sets of vectors, which pose extra challenges to representation learning. Most recent methods focus on learning representations from a single 3D object [1,17,21,36,42,47,54]. These methods mainly pre-train their models on ShapeNet [5]. However, the resulting models are found to have limited transferability to a real scene-level dataset, which is previously attributed to the domain gap [53]. In contrast, [40,53,58] directly apply representation learning on real-world scene-level point clouds. With the success of contrastive learning in 2D images, Xie et al. [53] proposed a point-level contrastive learning method that computes point-wise correspondences between the different views of a point cloud. Rao et al. [40] learned the 3D representation by applying object-level contrastive learning on two random scenes generated from the same set of synthetic objects. Inspired by instance discrimination [51], Zhang et al. [58] proposed a contrastive learning approach on point cloud instances. These methods require a strategy to define positive and negative pairs. They require substantial computational resources for large-scale batch size training. Unlike exploring different learning methods, this paper focuses on addressing the impact of sampling variations of point clouds.

Autoencoding Our work falls into the paradigm of point cloud autoencoding. In the same spirit of image autoencoding, point cloud autoencoding seeks to jointly train an encoder and a decoder that can recover the input point cloud or a complete point cloud. Probably the most similar work to ours are FoldingNet [54] and OcCo [47]. FoldingNet designed a point cloud autoencoder with a novel folding-based decoder. OcCo proposed a pipeline first to mask point cloud from camera viewpoints and then reconstruct the complete point cloud by a standard point cloud completion model [56]. Both works demonstrate promising gains of such pre-training. However, using point cloud as decoder representation has several drawbacks. First, the discrete surface representation forces the autoencoder to learn irrelevant input sampling information. Second, unlike grid-structured images, which can be easily decoded using convolution, symmetrically training a point cloud decoder is considerably harder. In contrast, we propose to replace the point-cloud generation by implicit representation generation. As described in Section 1, the implicit representation decoder enjoys multiple benefits compared to the point cloud decoder and addresses the above problems effectively.

Implicit Representations Recent works have investigated the implicit representation of continuous 3D shapes by optimizing deep networks that map 3D coordinates to signed distance [27,32] or occupancy grids [7,26,35]. In contrast to explicit representations (e.g., point cloud, voxel, and triangle mesh) that possess discretization errors, implicit models represent shapes continuously and can handle complicated shapes with varying topologies. Implicit representations have been successfully adapted in various 3D tasks ranging from 3D reconstruction from images [23,24,29], primitive-based 3D reconstruction [13,14,33], 4D reconstruction [28], and representing continuous texture [31]. However, no prior work has integrated implicit representations into self-supervised representation learning on point clouds to the best of our knowledge.

3 Motivation

The motivation of IAE can be summarized as follows. Point clouds sampled from continuous 3D models have sampling variations that do not capture useful information about the underlying 3D geometry. In the context of representation learning via autoencoding, if both the encoder and the decoder use point cloud as the surface representation, they will be forced to capture such variations to reconstruct the original inputs. On the other hand, if the decoder uses the implicit representation, which is invariant to such sampling variations, then the encoder and the decoder do not need to capture such variations. Instead, the encoder is encouraged to disregard the irrelevant sampling variations to capture the underlying 3D geometry features. For simplicity, we provide an analysis under a linear autoencoding model, which already offers valuable insights.

Specifically, suppose we have $N > n$ points $\boldsymbol{x}_i \in R^n, 1 \leq i \leq N$. Without losing generality, we assume $\boldsymbol{x}_i$ lies on a low-dimensional linear space $\{\,L\,\}$ of dimension $m < n$. These data points are used to model the underlying 3D models. Now let us perturb each point $\boldsymbol{x}'_i = \boldsymbol{x}_i + \epsilon_i$, where ϵ_i is used to model sampling variations. We assume $\epsilon_i \in \{\,L\,\}^\perp$, meaning they encode variations that are orthogonal to variations of the underlying 3D models. Denote $X := (\boldsymbol{x}_1, \cdots, \boldsymbol{x}_N)$ and $X' := (\boldsymbol{x}'_1, \cdots, \boldsymbol{x}'_N)$.

We consider two linear autoencoding models. The first one, which is analogous to IAE, takes $\boldsymbol{x}'_i$ as input and seeks to reconstruct $\boldsymbol{x}_i$:

$$A^\star, Q^\star = \underset{A', Q' \in \mathbb{R}^{n\times m}}{\operatorname{argmin}} \sum_{k=1}^{N} \|A' {Q'}^T \boldsymbol{x}'_k - \boldsymbol{x}_k\|^2 \qquad s.t. \quad {Q'}^T Q' = I_m,\ Q' \in \{\,C\,\}(X') \tag{1}$$

Here Q' is the encoder, and A' is the decoder. $\{\,C\,\}(X')$ denotes the column space of matrix X'. The constraints ${Q'}^T Q' = I_m$ and $Q' \in \{\,C\,\}(X')$ ensure that the encoder-decoder pair is unique up to an unitary transformation in $O(m)$. The following proposition characterizes that $Q^\star$ is independent of $\epsilon_k, 1 \leq k \leq n$.

Proposition 1. *Let $Q \in \mathbb{R}^{n\times m}$ collect the top-m eigenvectors of the convariance matrix $C = \sum\limits_{k=1}^{N} \boldsymbol{x}_k \boldsymbol{x}'_k$. Then under the assumption that $\epsilon_k \in \{\,L\,\}^\perp, 1 \leq k \leq N$,*

$$Q^\star = Q. \tag{2}$$

In other words, $Q^\star$ does not encode sampling variations.

Now consider the second autoencoding model where we force the encoder-decoder pair to reconstruct the original inputs:

$$\hat{A}^\star, \hat{Q}^\star = \underset{A', Q' \in \mathbb{R}^{n\times m}}{\operatorname{argmin}} \sum_{k=1}^{N} \|A' {Q'}^T \boldsymbol{x}'_k - \boldsymbol{x}'_k\|_{\mathcal{F}}^2 \qquad s.t. \quad {Q'}^T Q' = I_m,\ Q' \in \{\,C\,\}(X') \tag{3}$$

In this case, $\hat{A}^\star = \hat{Q}^\star$, and both of them are given by the top m eigenvectors of the covariance matrix $C' = \sum \boldsymbol{x}_k' {\boldsymbol{x}_k'}^T$.

To quantitatively compare encoders $\hat{Q}^\star$ and Q, we need the following definition.

Definition 1. *Consider two unitary matrices $Q_1, Q_2 \in \mathbb{R}^{n\times m}$ where $Q_i^T Q_i = I_m$. We define the deviation between them as*

$$\mathcal{D}(Q_1, Q_2) := Q_1 - Q_2 R^\star, \qquad R^\star = \underset{R\in O(m)}{\operatorname{argmin}} \|Q_1 - Q_2 R\|_{\mathcal{F}}^2 \tag{4}$$

The following proposition specifies the derivatives between $\hat{Q}^\star$ and ϵ_k.

Proposition 2. *Under the assumption that $\epsilon_k \in \{L\}^\perp, 1 \le k \le N$, we have*

$$\frac{\partial \mathcal{D}(\hat{Q}^\star, Q)}{\partial \epsilon_{ki}} = (I_n - QQ^T)(\boldsymbol{e}_i \boldsymbol{x}_k^T) Q \Lambda^+ \tag{5}$$

where $\Lambda = \operatorname{diag}(\lambda_1, \cdots, \lambda_m)$ is a diagonal matrix that collects the top eigenvalues of C that correspond to Q. $\boldsymbol{e}_k$ is the k-th basis vector.

In other word, $\hat{Q}^\star$ is sensitive to ϵ_k. Therefore, it encodes sampling variations.

4 Approach

This section introduces the details of IAE. We begin with describing the underlying principles of IAE in Section 4.1. Next, We describe the technical details of IAE in Section 4.2. Then, we show the implementation details in Section 4.3.

4.1 Explicit Autoencoder v.s. Implicit Autoencoder

Explicit Autoencoder. Given an input point cloud $\mathcal{P}^{\text{in}} \in \mathcal{R}^{n_0\times 3}$ and a target point cloud $\mathcal{P}^{\text{gt}}$, an Explicit Autoencoder jointly trains an encoder f_Θ and a decoder g_Φ using a distance metric $d_{\text{exp}}(\cdot,\cdot) : \mathcal{R}^{n_0\times 3} \times \mathcal{R}^{n_1\times 3} \to \mathcal{R}$ between the input point cloud and the sub-sampled version of target point cloud $\mathcal{P}^{\text{gt}}_{\text{sub}} \in \mathcal{R}^{n_1\times 3}$:

$$\min_{\Theta,\Phi} d_{\text{exp}}((g_\Phi \circ f_\Theta)(\mathcal{P}^{\text{in}}), \mathcal{P}^{\text{gt}}_{\text{sub}}) \tag{6}$$

This paper differentiates two settings of autoencoding. In the case of pure autoencoding, $\mathcal{P}^{\text{gt}}_{\text{sub}} = \mathcal{P}^{\text{in}}$. In contrast, the second setting considers point cloud completion from partial inputs. In this setting, $\mathcal{P}^{\text{gt}}_{\text{sub}}$ is typically different than $\mathcal{P}^{\text{in}}$. Common choices of d_{exp} include Earth Mover Distance (EMD) and Chamfer Distance (CD). Computing EMD requires computing an optimal bijective mapping between two point clouds. Computing CD is comparably less expensive but still requires finding correspondence between two point clouds.

Implicit Autoencoder. In contrast to Explicit Autoencoder which needs to predict the full explicit representation using f_Θ, Implicit Autoencoder outputs an implicit function $(g_\Phi \circ f_\Theta)(x|\mathcal{P}^{\text{in}}) : \mathcal{R}^3 \to \mathcal{R}$, where $x \in \mathcal{R}^3$ denotes the query

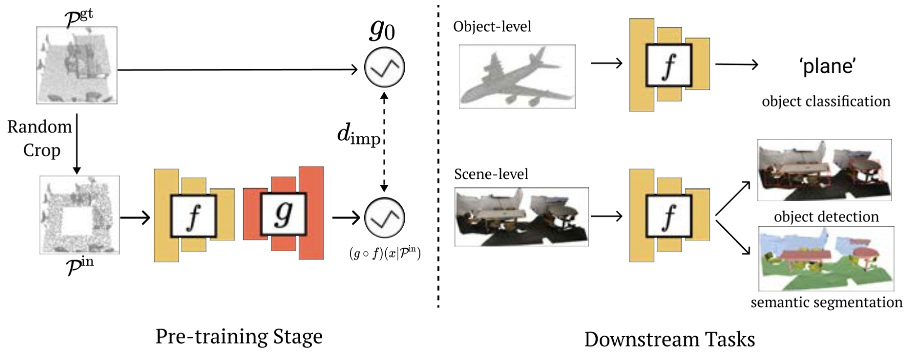

Fig. 2: **Detailed pipeline of Implicit AutoEncoder.** Left: **Pre-training stage.** Take ScanNet data as an example. Given a point cloud $\mathcal{P}^{\text{gt}}$, we apply a random center crop for the input point cloud $\mathcal{P}^{\text{in}}$. Backbone module f encodes the input. Prediction module g takes embedding features from f and query point x as input and outputs implicit prediction[1]. g_0 is ground truth implicit function obtained from $\mathcal{P}^{\text{gt}}$. After pre-training, we only keep f for further fine-tuning. Right: **Downstream tasks** contain two parts. For object-level tasks, we evaluate on object classification task. For scene-level tasks, we evaluate on object detection and semantic segmentation.

point. This function is conditioned on the input point cloud $\mathcal{P}^{\text{in}}$. We further define the ground truth implicit function as $g_0(x) : \mathcal{R}^3 \to \mathcal{R}$. In practice, g_0 can be chosen as the signed distance function, occupancy grid, among others. The training objective is to match these two functions through a distance metric d_{imp} between implicit surfaces:

$$\min_{\Theta,\Phi} d_{\text{imp}}((g_\Phi \circ f_\Theta)(x|\mathcal{P}^{\text{in}}), g_0(x)) \tag{7}$$

Here, the choice of d_{imp} and g_0 are typically coupled. For example, when g_0 is a signed distance function we choose d_{imp} to be L_1 distance. On the other hand, we choose d_{imp} to be a cross-entropy when g_0 is the occupancy grid.

4.2 Design Space for Implicit Autoencoder

Network Architecture. The network structure of our paradigm includes an encoder f_Θ and a decoder g_Φ, as shown in Figure 2. Our Implicit Autoencoder trains these two modules together. After pre-training, g_Φ is discarded and the encoder module f_Θ is further fine-tuned for downstream tasks. Note that f_Θ can use different backbones for different tasks. Details are shown in Section 5.2.

The network architecture of g_Φ can take different implicit representations. Our experiments explore two different designs of g_Φ: Occupancy Network Style and Convolutional Occupancy Network style. Detailed analysis is shown in Section 5.3.

[1] For simplification, we denote f as f_Θ, g as g_Φ, and do not show the query point x in Figure 2.

Experimental results show that the Convolutional Occupancy Network style prediction module g_Φ exhibits better performance.

Output Encoding. The formulation of Eq 7 gives the flexibility of defining g_0 by choosing a suitable implicit encoding of the output. We experimented with different ways to define g_0. The first one defines g_0 as the signed distance to the point cloud $\mathcal{P}^{\text{gt}}$. We also experimented with the unsigned distance field (e.g., when normal information is unavailable). Moreover, we also experimented with different occupancy functions.

Loss Function. A simple way to minimize the distance between $g_\Phi \circ f_\Theta$ and g_0 is to minimize the evaluation on a sample set of the ambient space. For example, in the case of the unsigned distance field, the evaluation is defined using the L_1 norm:

$$\mathcal{L} = \frac{1}{N} \sum_{i=0}^{N} \|(g_\Phi \circ f_\Theta)(x_i|\mathcal{P}^{\text{in}}) - g_0(x_i)\| \tag{8}$$

where $x_i \in \mathcal{S}$ is uniformly sampled inside the bounding box of $\mathcal{P}^{\text{in}}$.

Partial Point Cloud Input. To help the model capture high-level semantic features, we randomly center-crop a part of the input point cloud (See Figure 2). We show that our model is able to reconstruct the missing parts and the resulting encoder can achieve better performance in downstream tasks.

4.3 Implementation Details

Data Generation. Given a point cloud $\mathcal{P}^{\text{gt}}$, we first randomly choose a removing ratio from 0% to 50% and apply a center-cropping on it to get the input point cloud $\mathcal{P}^{\text{in}}$. To build the ground truth label of the implicit function g_0, we use different strategies on synthetic and real datasets. First, we uniformly sample the query points within the volume of interest. Then, for the real dataset, to obtain the unsigned distance value, we directly compute the distance d between the query point and nearest point from $\mathcal{P}^{\text{gt}}$. And because the point cloud from the real dataset is usually incomplete, we do not define the sign distance values. For the occupancy value, we set the label to be 1 if the distance $d < 0.005m$, and 0 if $d \geq 0.005m$. For the synthetic dataset, we obtain the true signed distance, unsigned distance, and occupancy values from the underlying water-tight meshes.

Pre-training. We implement all models in PyTorch and use Adam optimizer with no weight decay. The learning rate is set to 10^{-4} for all datasets. For ShapeNet, we pre-train the models for 600 epochs. And for ScanNet, we pre-train the models for 1000 epochs.

5 Experiments

In this section, we first introduce the pre-training setting of IAE on different datasets in Section 5.1. Next, we evaluate our models on various downstream tasks in Section 5.2. At length, we present a series of ablation studies and experiment analysis in Section 5.3 and 5.4.

Method	**ModelNet40**
3D-GAN [49]	83.3%
Latent-GAN [1]	85.7%
SO-Net [21]	87.3%
MAP-VAE [16]	88.4%
Jigsaw* [42]	84.1%
FoldingNet* [54]	90.1%
Orientation* [36]	90.7%
STRL* [20]	90.9%
OcCo* [47]	89.7%
IAE(ours)	**92.1%**

Table 1: **Linear evaluation for shape classification on ModelNet40**. Note that to make a fair comparison, different * methods use the same DGCNN encoder backbone.

Category	**Method**	**ModelNet40**
Supervised	PointNet [38]	89.2%
	PointNet++ [39]	90.7%
	PointCNN [22]	92.2%
	KPConv [44]	92.9%
	DGCNN [48]	92.9%
	PointTransform [59]	93.7%
Self-Supervised	FoldingNet [54]	93.1%
	STRL [20]	93.1%
	OcCo [47]	93.0%
	IAE(ours)	**93.7%**

Table 2: **Shape classification fine-tuned results on ModelNet40.** Supervised learning methods train the model from scratch. Self-supervised methods use the pre-trained models as the initial weight for supervised fine-tuning. All the self-supervised methods shown here use the same DGCNN encoder backbone.

5.1 Pre-Training Dataset

We use two datasets for pre-training. ShapeNet is used for shape classification. ScanNet is used for indoor 3D object detection and 3D semantic segmentation.
ShapeNet [5] contains 57,748 synthetic 3D shapes from 55 categories. We follow the procedure of [26, 32] to generate the signed distance, unsigned distance, and occupancy grid labels for each point cloud. Note that we need water-tight meshes in this step to generate the sign. During training, we apply the same data augmentation methods as FoldingNet [54].
ScanNet [8] contains more than 1500 real indoor scenes. We apply a sliding window strategy and crop each scene into small cubes with the size of $d \times d \times d$. We set $d = 3.0m$ in this paper. Following the train/val split from [37], we extract around 8K/2.5K point clouds in the training/validation set. For each point cloud, we randomly sample 10000 points as the input. Due to the lack of water-tight meshes, we cannot easily define signed distances and occupancy values. Therefore, we follow the procedure described in Section 4.3 to obtain the true labels. During training, we force the network to generate unsigned distance values by taking absolute values at the output layer.

5.2 Downstream Tasks

One of the most important motivations for representation learning is to learn features that can transfer well to different downstream tasks. In this section, we present the experiment settings and results for each downstream tasks.

Shape Classification. Following the standard protocols from previous work, we evaluate the shape feature learning of our model on ModelNet40 benchmark [50]. ModelNet dataset has two variants, i.e., ModelNet40 and ModelNet10. ModelNet40 consists of 9832 training objects and 2468 test objects in 40 classes.

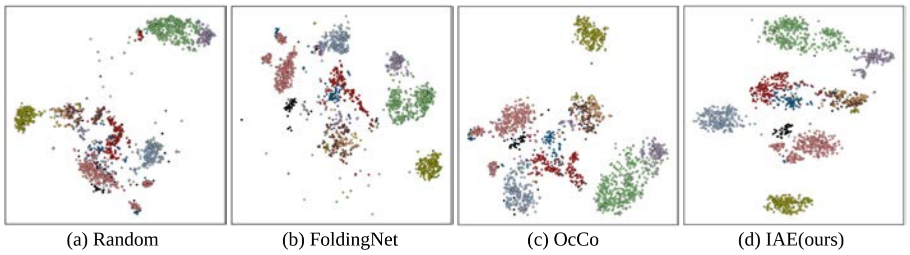

Fig. 3: **Visualization of learned features.** We visualize the learned features for each sample in ModelNet10 using t-SNE. All the models use DGCNN as the encoder backbone. (a) uses random initialization. (b),(c),(d) are pre-trained on ShapeNet.

ModelNet10 consists of 3991 training objects, 908 test objects in 10 classes. We pre-process the training datasets by following [38]. Each shape is sampled to 10,000 points.

Linear SVM Evaluation. In this experiment, we train a linear Support Vector Machine (SVM) classifier using the latent code obtained from our backbone module f_Θ, which is pre-trained on ShapeNet. To make a fair comparison, we use DGCNN [48] as the encoder backbone, following the practice of previous approaches [20, 36, 47]. We randomly sample 2048 points from each shape for both pre-training and SVM training. Table 1 shows the classification results. IAE achieves the state-of-the-art performance of 92.1% accuracy on ModelNet40, while the runner-up method only has 90.9% accuracy. Since the pre-training of the encoder and the training of linear SVM are on different datasets, such results also demonstrate the transferability of our model.

Supervised Fine-tuning. In this experiment, we fine-tune our pre-trained model using supervised methods. Specifically, we use our pre-trained model as the initialization weights of the DGCNN encoder and then fine-tune it on the ModelNet40 dataset. The results are shown in Table 2. We can see that IAE shows the best performance (93.7%) among other self-supervised approaches under the same encoder backbone (DGCNN).

Embedding Visualization. We visualize the learned features of our model and baseline approaches in Figure 3. We compare with FoldingNet [54], OcCo [47], and a sanity-check baseline, random initialization. Random initialization use randomly initialized network weight to obtain the embedding, and its performance explains the network prior. The embeddings for different categories in the ModelNet10 dataset are shown using t-SNE dimension reduction. Empirically, we observe that our pre-trained model provides a cleaner separation between different shape categories than FoldingNet [54], OcCo [47], and random initialization.

Indoor 3D Object Detection. Self-supervised pre-training for real-world 3D object detection is considered more challenging than shape classification. As observed in PointContrast [53], pre-training on synthetic datasets, such as ShapeNet, usually does not generalize well to real-world tasks. However, pre-training on real-world datasets turns out to be difficult as well. Since real-world point clouds are usually very noisy, complicated, and incomplete, previous

Method	**ScanNet**		**SUN RGB-D**	
	AP$_{50}$	**AP$_{25}$**	**AP$_{50}$**	**AP$_{25}$**
VoteNet [37]	33.5	58.6	32.9	57.7
STRL [20]	38.4	59.5	35.0	58.2
RandomRooms [40]	36.2	61.3	35.4	59.2
PointContrast [53]	38.0	59.2	34.8	57.5
DepthContrast[2] [58]	39.1	**62.1**	35.4	**60.4**
IAE (Ours)	**39.8**	61.5	**36.0**	**60.4**

Table 3: **3D object detection results.** We fine-tuned our pre-trained model on ScanNetV2 and SUN-RGBD validation set using a popular detection framework, VoteNet [37]. We show mean of average precision(mAP) across all semantic classes with 3D IoU threshold 0.25 and 0.5. Our method outperforms prior work across most metrics.

approaches [36, 42, 47] failed to achieve considerable performance gains via self-supervised pre-training.

IAE takes advantage of convolutional occupancy network to generate implicit functions as output, making it easier to handle complex point clouds [35]. Specifically, we use VoteNet-style PointNet++ [37] as our encoder module f_Θ and pre-train the model on ScanNet. Next, we use the pre-trained weights as the initialization and further fine-tune them for detection tasks. Table 3 shows the results. Our model shows 18.8% and 4.9% improvements compared with training from scratch on mAP 0.5 and 0.25, respectively. Furthermore, we also fine-tune our pre-trained model on a more challenging dataset, SUN RGB-D [43]. It contains 10,335 single-view RGB-D images, split into 5,285 training samples and 5,050 validation samples. As shown in Table 3, our model performs the best on SUN RGB-D. The difference between the pre-training dataset and fine-tuning dataset further demonstrates the transferability of our pre-training method.

Indoor 3D Semantic Segmentation. We further evaluate our model on the indoor semantic segmentation task. We use Stanford Large-Scale 3D Indoor Spaces (S3DIS) [2] which consists of 3D point cloud data from 6 large-scale indoor areas with per-point categorical annotation. Our model is pre-trained on ScanNet and fine-tuned on S3DIS. We report the results in Table 4. IAE consistently outperforms other methods. It outperforms the state-of-the-art method by 0.8% and 3.8% on OA and mIoU.

Label Efficiency Training. We study the label efficiency of our model on 3D object detection by varying the portion of supervised training data. Results can be found in Figure 4. We use 20%, 40%, 60%, and 80% of the training data from ScanNet and SUN RGB-D dataset. We can observe that our pre-training method gives larger gains when the labeled data is less. And with only 60% training data on ScanNet/SUN RGB-D, our model can get similar performance compared with using all training data from scratch. This suggests our pre-training can help the downstream task to obtain better results with fewer data.

[2] In the DepthContrast paper, they used a slightly larger model than Votenet. For a fair comparison, we reproduce DepthContrast with the original Votenet model.

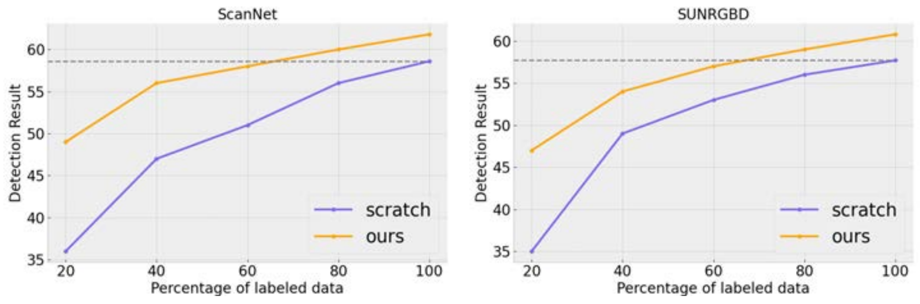

Fig. 4: **Label efficiency training.** We pre-train our model on ScanNet and then fine-tune on ScanNet and SUN RGB-D separately. During fine-tuning, different percentages of labeled data are used. Our pre-training model outperforms training from scratch and achieves nearly the same result with only 60% labeled data.

Method	**OA**	**mIoU**
DGCNN [48]	84.1	56.1
Jigsaw [42]	84.4	56.6
OcCo [47]	85.1	58.5
IAE(ours)	**85.9**	**60.7**

Table 4: **Semantic segmentation results on S3DIS.** We show overall accuracy(OA) and intersection of union(mIoU) across six folds.

Task	**Pre-train**	**Acc/AP$_{25}$**
Object detection	ScanNet	**60.4**
	ShapeNet	59.4
MN40 Linear	ScanNet	91.1%
	ShapeNet	**92.1%**

Table 5: **Cross-domain generalizability** between ShapeNet and ScanNet. For 3D object detection task, we report mAP at IoU=0.25 on SUN RGB-D dataset. For ModelNet40 Linear evaluation task, we report classification accuracy.

Cross-domain generalizability. Utilizing synthetic CAD object models to help the learning of 3D real data tasks(e.g., object detection) remains an open problem in 3D computer vision. Xie et al. [53] provides a failure object detection case when pre-training the backbone model on ShapeNet and fine-tuning on ScanNet. However, recently, Huang et al. [20] reported an opposite observation and attributed the failure reason of [53] to the simple encoder architecture. Since it is much easier to access a large number of synthetic data, it is still desirable to explore the possibility of whether the learned model on synthetic data can have good generalizability to real data. To elucidate this problem, we pre-train the model on the real ScanNet dataset and the synthetic ShapeNet dataset, and test their cross-domain generalizability. Table 5 summarizes the results. For 3D object detection, the model pre-trained on ShapeNet can achieve 59.4 mAP, which is lower than the one pre-trained on ScanNet. However, it still shows an improvement over training from scratch (57.7). This observation is consistent with the conclusion from Huang et al. [20] and demonstrates the effective cross-domain generalizability of our model.

Conversely, we also report linear evaluation results on the ModelNet40 benchmark. It is interesting to observe the transferability from natural scenes to the synthetic shape domain. Surprisingly, pre-training on ScanNet achieves a

Decoder	Method	ModelNet40
Explicit	FoldingNet [54]	90.1%
	OcCo [47]	89.7%
	SnowflakeNet [52]	89.9%
Implicit	OccNet [26]	91.5%
	Conv-OccNet [35]	**92.1%**

Decoder	Functions	ModelNet40
Explicit	Point Cloud	90.1%
Implicit	Occ Value	91.3%
	UDF	91.7%
	SDF	**92.1%**

Table 6: Left: **Ablation study on different decoder model.** On ModelNet40, we show linear evaluation results. Our implicit auto-encoder formulations can be improved upon explicit counterpart under various decoder models. Right: **Ablation study on implicit function.** For explicit representation, we use FoldingNet as the decoder. For implicit representation, we experimented with Occupancy Value(Occ Value), Unsigned Distance Function(UDF), and Signed Distance Function(SDF) and find consistent improvement over explicit representation.

comparable result with 91.1% accuracy. This result outperforms previous state-of-the-art methods and demonstrates the strong transferability of our model.

5.3 Ablation Study

In this section, we discuss a series of ablation studies to understand the benefit of each design choice of IAE.
Explicit Decoder v.s. Implicit Decoder. In this experiment, we use the same encoder model and experimented with different decoder models from both categories. Specifically, we study three state-of-the-art explicit decoder models of FoldingNet [54], OcCo [47], and SnowflakeNet [52], and two implicit decoder models of Occupancy Network [26] and Convolutional Occupancy Network [35]. Note that while Convolutional Occupancy Network is a volumetric implicit representation, Occupancy Network does not contain any volumetric representation. We pre-train on ShapeNet and evaluate models on ModelNet40 benchmark using linear SVM. The results are shown in Table 6. Surprisingly, we found the implicit decoders achieve consistently better performance than all explicit decoders.
Different Implicit Functions. We also investigate several different implicit representations, including signed distance function, unsigned distance function, and occupancy. Encouragingly, as shown in Table 6 right, all of them show better results than the explicit representation. Among those implicit representations, we found signed distance function works the best.
Completion v.s. No Completion. Instead of taking complete point cloud as input, an alternative approach is to take a partial part. Possessing the ability to recover the missing part, the model should be able to learn structural and contextual information, especially on real data. To study the influence of completion, we tried several different cropping settings. The results are reported in Table 7. We conduct all the experiments here on ScanNet and fine-tune the pre-trained models on 3D objection detection. First, we try different maximum cropping sizes, including 0%, 20%, 50%, 70% of the input point cloud. 0% means the input point cloud is complete. We find that all the models outperform training from scratch. The model with a maximum cropping size of 50% achieves the

cs=0	cs=0.2	cs=0.5	cs=0.7	gt	SUNRGBD
-	-	-	-	-	57.7
√	-	-	-	-	60.0
-	√	-	-	-	60.2
-	-	√	-	-	60.4
-	-	-	√	-	59.8
-	-	-	-	√	**60.8**

Table 7: **Different setting of data augmentation.** 'cs' denotes cropping size. 'gt' means using ground truth bounding box to guide partial point cloud generation. The first line shows the result with no pre-training.

Decoder	Method	SUN RGBD
Explicit	FoldingNet [54]	58.2
	OcCo [47]	58.4
	SnowflakeNet [52]	58.1
Implicit	Conv-OccNet [35]	**60.4**

Table 8: **Comparison between explicit approaches and our model** on real data completion task. Our model built upon convolutional occupancy network shows consistent improvements.

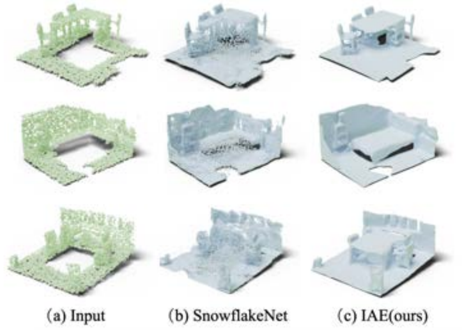

Fig. 5: **Qualitative completion results on ScanNet.** We show the results of SnowflakeNet [52] and our model.

best transfer learning performance on the SUN RGB-D dataset. Note that the cropping size=0 model can also get 60.0 mIoU, which is very close to the prior state-of-the-art method. These results further demonstrate the effectiveness of using implicit function as the output representation.

One interesting question to ask is that if we use ground truth bounding boxes of objects to guide partial point cloud generation, will the model learn better representation? To answer this question, we conduct the following experiment. For each input, we locate one target object around the center point($\pm 0.5m$). Then we randomly choose one-half part of the center cube($1.5m \times 1.5m$) and remove it. Note that other objects near the target within the range are likely cropped as well. Using this dataset, we pre-train another model and fine-tune it on SUN RGB-D detection. The result is shown in the last line of Table 7. Our approach achieved 60.8 mIoU, which is superior to our best setting. We hypothesis that this is because this setting forces the model to pay more attention to object-related contextual and semantic features and hence learn better representation for related high-level tasks.

Completion Result. To demonstrate the effectiveness of using implicit functions on the real data completion task, we compare with three explicit methods. In the same completion data setting, we pre-train FoldingNet, OcCo, and the current state-of-the-art point cloud completion approach SnowflakeNet [52] on ScanNet. Then we fine-tune models on SUN RGB-D detection. We show the result in

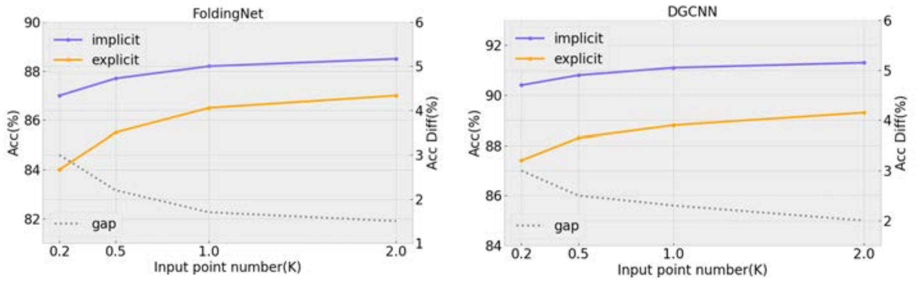

Fig. 6: **Comparison between explicit and implicit auto-encoder.** 'Acc Diff' denotes the accuracy difference between two models on ModelNet40 linear evaluation. With the increase of input point number, the gap between explicit and implicit model decrease. Please note that both models show slightly worse performance compared to Table 1 because we do not add data augmentations in this experiment.

Table 8. Our model outperforms these three explicit approaches consistently. Also, We show qualitative results of SnowflakeNet and our model on ScanNet. As shown in Figure 5, the explicit model failed to complete the missing part of the point cloud from ScanNet, while IAE gives a plausible completion.

5.4 Experiment Analysis

As discussed in Section 3, we argue that explicit autoencoders are forced to capture sampling variations in order to reconstruct the original point cloud. Intuitively, such sampling variations drop when increasing the sampling resolution.

To further study this problem, we conduct the following experiment. According to the definition in Section 4.1, first, on ShapeNet, we generate four datasets by sampling different number points of the input point cloud, ranging from $n_0 = 256$ to 2048 points. Then, we pre-train both explicit and implicit models and evaluate them on ModelNet40 using linear SVM. More precisely, consider $n_0 = 256$. For explicit models, the output is consistent with the input, which means $n_1 = 256$. Therefore, the output number of explicit models varies across different datasets. We use the FoldingNet-based decoder due to its flexibility and accuracy. For implicit models, the only difference among datasets is the input point cloud. The ground truth implicit function values keep the same. We use the convolutional occupancy network-based decoder and take SDF as the implicit representation. The result is illustrated in Figure 6. For fair comparisons, we train the models with two types of encoders, FoldingNet-based and DGCNN.

Under both settings, we notice that when increasing the point cloud resolution, the gap between explicit and implicit models narrows (gray dashed line). Our hypothesis is, for the explicit model with coarse point clouds and large sampling variations, it is forced to learn the sampling bias to reconstruct the ground-truth point clouds, which is not part of the generalizable knowledge. Therefore, it obtains the best performance when the input number increases to 2048. In contrast, for the implicit model, the ground truth label never changes. The learning supervision is consistent by minimizing the discrepancy of two implicit functions. So the change across different datasets is not that significant. We empirically validate that by taking implicit function as the decoder representation. In this case, the encoder is able to disregard the irrelevant sampling variation, which is why IAE can achieve better performance across different tasks.

In summary, while there are some reductions in performance gains of IAE when increasing the sampling resolution (gray dashed line), the performance gains are still considerable. One explanation is that point clouds are unordered sets, and there are computational challenges in matching point clouds, e.g., convergence and local minimum issues under both EMD and CD. Such computational challenges amplify for larger point clouds, as the search spaces become more complex.

6 Conclusions and Limitations

In this paper, we propose IAE, a simple yet effective self-supervised learning framework for the point cloud. Unlike the conventional autoencoder for point cloud which reconstructs input point cloud explicitly, we reconstruct the implicit function representation. We argue that IAE can prioritize that the encoder discards sampling variations, introducing more space to learn useful features. We found such simple change already enables the pre-training model to learn better representation and achieve considerable improvement over a wide range of downstream tasks, including 3D shape classification, 3D object detection, and 3D semantic segmentation. We also demonstrate how completing partial input can further boost the pre-training gains by learning mid to high level semantic concepts. One limitation of our work is that we require an additional pre-processing step of the raw point cloud to get implicit representation training labels. Another limitation is that we only experimented with hand-crafted implicit function targets, such as signed distance function, while jointly learning implicit function targets might bring more improvements.

Acknowledgements. Part of this work was initiated when Siming Yan was a summer research intern at Wormpex AI Research. We would like to thank Bo Sun and Haitao Yang for the helpful discussions. Qixing Huang would like to acknowledge the support from NSF Career IIS-2047677 and NSF HDR TRIPODS-1934932. Gang Hua is partly supported by National Key R&D Program of China Grant 2018AAA0101400, NSFC Grants 61629301, 61773312, and 61976171.

References

1. Achlioptas, P., Diamanti, O., Mitliagkas, I., Guibas, L.: Learning representations and generative models for 3d point clouds. In: International conference on machine learning. pp. 40–49. PMLR (2018) 4, 9
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: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 1534–1543 (2016) 1, 11
3. Bengio, Y., Courville, A., Vincent, P.: Representation learning: A review and new perspectives. IEEE transactions on pattern analysis and machine intelligence **35**(8), 1798–1828 (2013) 2

4. 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). pp. 132–149 (2018) 3
5. Chang, A.X., Funkhouser, T., Guibas, L., Hanrahan, P., Huang, Q., Li, Z., Savarese, S., Savva, M., Song, S., Su, H., et al.: Shapenet: An information-rich 3d model repository. arXiv preprint arXiv:1512.03012 (2015) 1, 4, 9
6. Chen, T., Kornblith, S., Norouzi, M., Hinton, G.: A simple framework for contrastive learning of visual representations. In: International conference on machine learning. pp. 1597–1607. PMLR (2020) 2, 3
7. Chen, Z., Zhang, H.: Learning implicit fields for generative shape modeling. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 5939–5948 (2019) 4
8. Dai, A., Chang, A.X., Savva, M., Halber, M., Funkhouser, T., Nießner, M.: Scannet: Richly-annotated 3d reconstructions of indoor scenes. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 5828–5839 (2017) 1, 9
9. Doersch, C., Gupta, A., Efros, A.A.: Unsupervised visual representation learning by context prediction. In: Proceedings of the IEEE international conference on computer vision. pp. 1422–1430 (2015) 2, 3
10. Erhan, D., Courville, A., Bengio, Y., Vincent, P.: Why does unsupervised pre-training help deep learning? In: Proceedings of the thirteenth international conference on artificial intelligence and statistics. pp. 201–208. JMLR Workshop and Conference Proceedings (2010) 2
11. Fan, H., Su, H., Guibas, L.J.: A point set generation network for 3d object reconstruction from a single image. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 605–613 (2017) 3
12. Geiger, A., Lenz, P., Urtasun, R.: Are we ready for autonomous driving? the kitti vision benchmark suite. In: 2012 IEEE conference on computer vision and pattern recognition. pp. 3354–3361. IEEE (2012) 1
13. Genova, K., Cole, F., Sud, A., Sarna, A., Funkhouser, T.: Local deep implicit functions for 3d shape. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 4857–4866 (2020) 4
14. Genova, K., Cole, F., Vlasic, D., Sarna, A., Freeman, W.T., Funkhouser, T.: Learning shape templates with structured implicit functions. In: Proceedings of the IEEE/CVF International Conference on Computer Vision. pp. 7154–7164 (2019) 4
15. Gidaris, S., Singh, P., Komodakis, N.: Unsupervised representation learning by predicting image rotations. arXiv preprint arXiv:1803.07728 (2018) 3
16. Han, Z., Wang, X., Liu, Y.S., Zwicker, M.: Multi-angle point cloud-vae: Unsupervised feature learning for 3d point clouds from multiple angles by joint self-reconstruction and half-to-half prediction. In: 2019 IEEE/CVF International Conference on Computer Vision (ICCV). pp. 10441–10450. IEEE (2019) 9
17. Hassani, K., Haley, M.: Unsupervised multi-task feature learning on point clouds. In: Proceedings of the IEEE/CVF International Conference on Computer Vision. pp. 8160–8171 (2019) 4
18. He, K., Chen, X., Xie, S., Li, Y., Dollár, P., Girshick, R.: Masked autoencoders are scalable vision learners. arXiv preprint arXiv:2111.06377 (2021) 2, 4
19. He, K., Fan, H., Wu, Y., Xie, S., Girshick, R.: Momentum contrast for unsupervised visual representation learning. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 9729–9738 (2020) 3
20. 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) 9, 10, 11, 12
21. Li, J., Chen, B.M., Lee, G.H.: So-net: Self-organizing network for point cloud analysis. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 9397–9406 (2018) 4, 9

22. Li, Y., Bu, R., Sun, M., Wu, W., Di, X., Chen, B.: Pointcnn: Convolution on x-transformed points. Advances in neural information processing systems **31**, 820–830 (2018) 9
23. Liu, S., Zhang, Y., Peng, S., Shi, B., Pollefeys, M., Cui, Z.: Dist: Rendering deep implicit signed distance function with differentiable sphere tracing. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 2019–2028 (2020) 4
24. Liu, S., Saito, S., Chen, W., Li, H.: Learning to infer implicit surfaces without 3d supervision. arXiv preprint arXiv:1911.00767 (2019) 4
25. Masci, J., Meier, U., Cireşan, D., Schmidhuber, J.: Stacked convolutional auto-encoders for hierarchical feature extraction. In: International conference on artificial neural networks. pp. 52–59. Springer (2011) 2, 4
26. Mescheder, L., Oechsle, M., Niemeyer, M., Nowozin, S., Geiger, A.: Occupancy networks: Learning 3d reconstruction in function space. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 4460–4470 (2019) 4, 9, 13
27. Michalkiewicz, M., Pontes, J.K., Jack, D., Baktashmotlagh, M., Eriksson, A.: Implicit surface representations as layers in neural networks. In: Proceedings of the IEEE/CVF International Conference on Computer Vision. pp. 4743–4752 (2019) 4
28. Niemeyer, M., Mescheder, L., Oechsle, M., Geiger, A.: Occupancy flow: 4d reconstruction by learning particle dynamics. In: Proceedings of the IEEE/CVF International Conference on Computer Vision. pp. 5379–5389 (2019) 4
29. Niemeyer, M., Mescheder, L., Oechsle, M., Geiger, A.: Differentiable volumetric rendering: Learning implicit 3d representations without 3d supervision. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 3504–3515 (2020) 4
30. Noroozi, M., Favaro, P.: Unsupervised learning of visual representations by solving jigsaw puzzles. In: European conference on computer vision. pp. 69–84. Springer (2016) 3
31. Oechsle, M., Mescheder, L., Niemeyer, M., Strauss, T., Geiger, A.: Texture fields: Learning texture representations in function space. In: Proceedings of the IEEE/CVF International Conference on Computer Vision. pp. 4531–4540 (2019) 4
32. Park, J.J., Florence, P., Straub, J., Newcombe, R., Lovegrove, S.: Deepsdf: Learning continuous signed distance functions for shape representation. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 165–174 (2019) 4, 9
33. Paschalidou, D., Gool, L.V., Geiger, A.: Learning unsupervised hierarchical part decomposition of 3d objects from a single rgb image. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 1060–1070 (2020) 4
34. Pathak, D., Krahenbuhl, P., Donahue, J., Darrell, T., Efros, A.A.: Context encoders: Feature learning by inpainting. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 2536–2544 (2016) 2, 3
35. Peng, S., Niemeyer, M., Mescheder, L., Pollefeys, M., Geiger, A.: Convolutional occupancy networks. In: Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part III 16. pp. 523–540. Springer (2020) 4, 11, 13, 14
36. Poursaeed, O., Jiang, T., Qiao, H., Xu, N., Kim, V.G.: Self-supervised learning of point clouds via orientation estimation. In: 2020 International Conference on 3D Vision (3DV). pp. 1018–1028. IEEE (2020) 4, 9, 10, 11
37. Qi, C.R., Litany, O., He, K., Guibas, L.J.: Deep hough voting for 3d object detection in point clouds. In: Proceedings of the IEEE/CVF International Conference on Computer Vision. pp. 9277–9286 (2019) 9, 11

38. 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) 9, 10
39. Qi, C.R., Yi, L., Su, H., Guibas, L.J.: Pointnet++: Deep hierarchical feature learning on point sets in a metric space. arXiv preprint arXiv:1706.02413 (2017) 9
40. Rao, Y., Liu, B., Wei, Y., Lu, J., Hsieh, C.J., Zhou, J.: Randomrooms: Unsupervised pre-training from synthetic shapes and randomized layouts for 3d object detection. In: Proceedings of the IEEE/CVF International Conference on Computer Vision. pp. 3283–3292 (2021) 4, 11
41. Rubner, Y., Tomasi, C., Guibas, L.J.: The earth mover's distance as a metric for image retrieval. International journal of computer vision **40**(2), 99–121 (2000) 3
42. Sauder, J., Sievers, B.: Self-supervised deep learning on point clouds by reconstructing space. arXiv preprint arXiv:1901.08396 (2019) 4, 9, 11, 12
43. 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) 1, 11
44. 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/CVF International Conference on Computer Vision. pp. 6411–6420 (2019) 9
45. Tschannen, M., Bachem, O., Lucic, M.: Recent advances in autoencoder-based representation learning. arXiv preprint arXiv:1812.05069 (2018) 2
46. Vincent, P., Larochelle, H., Bengio, Y., Manzagol, P.A.: Extracting and composing robust features with denoising autoencoders. In: Proceedings of the 25th international conference on Machine learning. pp. 1096–1103 (2008) 2, 4
47. Wang, H., Liu, Q., Yue, X., Lasenby, J., Kusner, M.J.: Unsupervised point cloud pre-training via view-point occlusion, completion. arXiv preprint arXiv:2010.01089 (2020) 2, 4, 9, 10, 11, 12, 13, 14
48. Wang, Y., Sun, Y., Liu, Z., Sarma, S.E., Bronstein, M.M., Solomon, J.M.: Dynamic graph cnn for learning on point clouds. Acm Transactions On Graphics (tog) **38**(5), 1–12 (2019) 9, 10, 12
49. Wu, J., Zhang, C., Xue, T., Freeman, W.T., Tenenbaum, J.B.: Learning a probabilistic latent space of object shapes via 3d generative-adversarial modeling. In: Proceedings of the 30th International Conference on Neural Information Processing Systems. pp. 82–90 (2016) 9
50. Wu, Z., Song, S., Khosla, A., Yu, F., Zhang, L., Tang, X., Xiao, J.: 3d shapenets: A deep representation for volumetric shapes. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 1912–1920 (2015) 9
51. Wu, Z., Xiong, Y., Yu, S.X., Lin, D.: Unsupervised feature learning via non-parametric instance discrimination. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 3733–3742 (2018) 2, 3, 4
52. Xiang, P., Wen, X., Liu, Y.S., Cao, Y.P., Wan, P., Zheng, W., Han, Z.: Snowflakenet: Point cloud completion by snowflake point deconvolution with skip-transformer. In: Proceedings of the IEEE/CVF International Conference on Computer Vision. pp. 5499–5509 (2021) 13, 14
53. Xie, S., Gu, J., Guo, D., Qi, C.R., Guibas, L., Litany, O.: Pointcontrast: Unsupervised pre-training for 3d point cloud understanding. In: European Conference on Computer Vision. pp. 574–591. Springer (2020) 4, 10, 11, 12
54. Yang, Y., Feng, C., Shen, Y., Tian, D.: Foldingnet: Point cloud auto-encoder via deep grid deformation. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 206–215 (2018) 2, 4, 9, 10, 13, 14
55. 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. ACM Transactions on Graphics (ToG) **35**(6), 1–12 (2016) 1

56. Yuan, W., Khot, T., Held, D., Mertz, C., Hebert, M.: Pcn: Point completion network. In: 2018 International Conference on 3D Vision (3DV). pp. 728–737. IEEE (2018) 4
57. Zhang, R., Isola, P., Efros, A.A.: Colorful image colorization. In: European conference on computer vision. pp. 649–666. Springer (2016) 3
58. Zhang, Z., Girdhar, R., Joulin, A., Misra, I.: Self-supervised pretraining of 3d features on any point-cloud. arXiv preprint arXiv:2101.02691 (2021) 4, 11
59. Zhao, H., Jiang, L., Jia, J., Torr, P.H., Koltun, V.: Point transformer. In: Proceedings of the IEEE/CVF International Conference on Computer Vision. pp. 16259–16268 (2021) 9
60. Zhuang, C., Yan, S., Nayebi, A., Schrimpf, M., Frank, M.C., DiCarlo, J.J., Yamins, D.L.: Unsupervised neural network models of the ventral visual stream. Proceedings of the National Academy of Sciences **118**(3) (2021) 3
61. Zhuang, C., Zhai, A.L., Yamins, D.: Local aggregation for unsupervised learning of visual embeddings. In: Proceedings of the IEEE/CVF International Conference on Computer Vision. pp. 6002–6012 (2019) 3

A Proof of Propositions in Section 3

Denote
$$X = (\boldsymbol{x}_1, \cdots, \boldsymbol{x}_N), \qquad X' = (\boldsymbol{x}'_1, \cdots, \boldsymbol{x}'_N).$$

To obtain closed-form expressions of the linear auto-encoding problem, we reformulate the optimization problem as

$$\min_{R,B\in\mathbb{R}^{n\times m}, R^TR=I_m} \sum_{k=1}^{N} \|RB^T\boldsymbol{x}'_k - \boldsymbol{x}_k\|^2 \tag{9}$$

The following proposition specifies the optimal solution to (9).

Lemma 1. *The optimal solution $(R^\star, B^\star), R, B \in \mathbb{R}^{n\times m}$ to (9) satisfies that the columns of $R^\star$ are the leading m eigenvectors with the largest eigenvalues of*

$$(X'X)^T(X'X'^T)^+(X'X^T).$$

Moreover,
$$B^\star = (X'X'^T)^+(X'X^T)R^\star.$$

Proof. Denote $R = (\boldsymbol{r}_1, \cdots, \boldsymbol{r}_m)$ and $B = (\boldsymbol{b}_1, \cdots, \boldsymbol{b}_m)$. Then

$$\begin{aligned}\sum_{k=1}^{N}\|RB^T\boldsymbol{x}'_k - \boldsymbol{x}_k\|^2 &= \sum_{k=1}^{N}\Big({\boldsymbol{x}'_k}^TBB^T\boldsymbol{x}'_k - 2(R^T\boldsymbol{x}_k)^T(R^T\boldsymbol{x}'_k) + \|\boldsymbol{x}_k\|^2\Big)\\ &= \sum_{k=1}^{N}\Big(\sum_{j=1}^{m}\big((\boldsymbol{b}_j^T\boldsymbol{x}'_k)^2 - 2(\boldsymbol{r}_j^T\boldsymbol{x}_k)(\boldsymbol{b}_j^T\boldsymbol{x}'_k)\big) + \|\boldsymbol{x}_k\|^2\Big)\\ &= \sum_{j=1}^{m}\Big(\boldsymbol{b}_j^T\big(\sum_{k=1}^{N}\boldsymbol{x}'_k{\boldsymbol{x}'_k}^T\big)\boldsymbol{b}_j - 2\big(\sum_{k=1}^{N}\boldsymbol{x}'_k{\boldsymbol{x}_k}^T\boldsymbol{r}_j\big)^T\boldsymbol{b}_j\Big)\\ &\quad + \sum_{k=1}^{N}\|\boldsymbol{x}_k\|^2\end{aligned} \tag{10}$$

Therefore, define

$$\{\boldsymbol{b}_j^\star, 1\le j\le m\} = \underset{\boldsymbol{b}_j}{\operatorname{argmin}} \sum_{k=1}^{N}\|RB^T\boldsymbol{x}'_k - \boldsymbol{x}_k\|^2.$$

Since $\boldsymbol{b}_j$ lies in the column space of X', it is clear that the optimal solution $\boldsymbol{b}_j^\star$ is given by

$$\begin{aligned}\boldsymbol{b}_j^\star &= \Big(\sum_{k=1}^{N}\boldsymbol{x}'_k{\boldsymbol{x}'_k}^T\Big)^+\Big(\sum_{k=1}^{N}\boldsymbol{x}'_k{\boldsymbol{x}_k}^T\Big)\boldsymbol{r}_j\\ &= (X'X'^T)^+(X'X^T)\boldsymbol{r}_j, \qquad 1\le j\le m.\end{aligned} \tag{11}$$

Substituting (11) into (10), we have that the objective function becomes

$$\begin{aligned}
&\sum_{k=1}^{N} \|R{B^\star}^T \boldsymbol{x}'_k - \boldsymbol{x}_k\|^2 \\
=&\sum_{j=1}^{m} \boldsymbol{r}_j^T \left(X{X'}^T\right)\left(X'{X'}^T\right)^+\left(X'X^T\right)\boldsymbol{r}_j \\
&-2\sum_{j=1}^{m} \boldsymbol{q}_j^T \left(X{X'}^T\right)^T\left(X'{X'}^T\right)^+\left(X'X^T\right)\boldsymbol{r}_j + \sum_{k=1}^{N}\|\boldsymbol{x}_k\|^2 \\
=&-\sum_{j=1}^{m} \boldsymbol{r}_j^T \left(X'X^T\right)^T\left(X'{X'}^T\right)^+\left(X'X^T\right)\boldsymbol{r}_j + \sum_{k=1}^{N}\|\boldsymbol{x}_k\|^2
\end{aligned} \tag{12}$$

Therefore, the optimization problem in (9) reduces to

$$\max_{R\in\mathbb{R}^{n\times m}, R^TR=I_m} \text{Trace}\left(R^T\left(X'X^T\right)^T\left(X'{X'}^T\right)^+\left(X'X^T\right)R\right) \tag{13}$$

It is easy to see that the optimal solution $R^\star = (\boldsymbol{r}_1^\star, \cdots, \boldsymbol{r}_m^\star)$ to (9) satisfies that $\boldsymbol{r}_i^\star, i \in [m]$ are the leading m eigenvectors of $C_{X',X} := \left(X'X^T\right)^T\left(X'{X'}^T\right)^+\left(X'X^T\right)$. □

A.1 Proof of Proposition 1

We show that when $\epsilon_k \in \{L\}^\perp, k \in [N]$,

$$R^\star = B^\star = Q.$$

Therefore, the formulation of (1) is identical to that of (9). In fact, consider the singular value decomposition (SVD) of

$$X' = U\Sigma V^T.$$

First,

$$\begin{aligned}
X^TX =& X^TX' \\
=& X^TU\Sigma V^T.
\end{aligned}$$

Therefore, V is an orthnormal basis of the column space of X^T. This means we can write out the SVD of $X = U'\Sigma'V^T$. Again using $X^TX = X^TU\Sigma V^T$, we have

$$V\Sigma'^2V^T = V\Sigma'{U'}^TU\Sigma V^T.$$

It follows that

$$\Sigma'^2 = \Sigma'{U'}^TU\Sigma.$$

In other words,

$$U'\Sigma' = U\Sigma.$$

This means we can arrange the SVD of X so that

$$U' = U, \Sigma' = \Sigma.$$

We proceed to show that $(XX'^T)(X'X'^T)^+(X'X^T) = XX^T$. In fact,

$$\begin{aligned}
&(XX'^T)(X'X'^T)^+(X'X^T) \\
&= XV\Sigma U^T\big(U\Sigma V^T V\Sigma U^T\big)^+(U\Sigma V^T X^T) \\
&= XV\Sigma U^T\big(U\Sigma^2 U^T\big)^+(U\Sigma V^T X^T) \\
&= XV\Sigma U^T U\Sigma^{-2}U^T U\Sigma V^T X^T \\
&= XVV^T X^T \\
&= XV\Sigma U^T(U\Sigma^{-2}U^T)U\Sigma V^T X^T \\
&= XV\Sigma U^T(U\Sigma^2 U^T)^+ U\Sigma V^T X^T \\
&= XX^T(XX^T)^+ XX^T = XX^T.
\end{aligned}$$

Moreover,

$$\begin{aligned}
&(X'X'^T)^+(X'X^T) \\
&= (U\Sigma VV^T\Sigma U^T)^+(U\Sigma V^T V\Sigma U) \\
&= (U\Sigma^2 U^T)^+(U\Sigma^2 U) \\
&= I_n.
\end{aligned}$$

$\square$

A.2 Proof of Proposition 2

Let us first consider the case where the corresponding eigenvalues of Q are distinctive. Let $\boldsymbol{q}_j, m+1 \leq j \leq n$ expand the columns of Q to form an orthonormal basis of $\mathbb{R}^n$. In this case, applying the derivative formula of eigenvectors to each eigenvector and the fact that $\epsilon_k \in \{ L \}^\perp$, we obtain

$$d\boldsymbol{q}_i = -\sum_{j\neq i} \frac{\boldsymbol{q}_j^T \sum_{k=1}^{N} (\boldsymbol{x}_k \epsilon_k^T + \epsilon_k \boldsymbol{x}_k^T)\boldsymbol{q}_i}{\lambda_j - \lambda_i} \boldsymbol{q}_j \tag{14}$$

$$= -\sum_{j\neq i, j=1}^{m} \frac{\boldsymbol{q}_j^T \sum_{k=1}^{N} (\boldsymbol{x}_k \epsilon_k^T + \epsilon_k \boldsymbol{x}_k^T)\boldsymbol{q}_i}{\lambda_j - \lambda_i} \boldsymbol{q}_j - \sum_{j=m+1}^{n} \frac{\boldsymbol{q}_j^T \sum_{k=1}^{N} (\boldsymbol{x}_k \epsilon_k^T + \epsilon_k \boldsymbol{x}_k^T)\boldsymbol{q}_i}{\lambda_j - \lambda_i} \boldsymbol{u}_j$$

$$= -\sum_{j=m+1}^{n} \frac{\boldsymbol{q}_j^T \sum_{k=1}^{N} \epsilon_k \boldsymbol{x}_k^T \boldsymbol{q}_i}{\lambda_j - \lambda_i} \boldsymbol{q}_j$$

$$= \big(\sum_{j=m+1}^{n} \boldsymbol{q}_j \boldsymbol{q}_j^T \big) \big(\sum_{k=1}^{N} \epsilon_k \boldsymbol{x}_k^T \big) \boldsymbol{q}_i \lambda_i^{-1} = \left(I_n - QQ^T \right) \sum_{k=1}^{N} \epsilon_k \boldsymbol{x}_k^T \boldsymbol{q}_i \lambda_i^{-1}. \tag{15}$$

It is easy to check that

$$\begin{aligned} d\boldsymbol{q}_i^T \boldsymbol{q}_i &= 0 \qquad 1 \leq i \leq m \\ d\boldsymbol{q}_i^T \boldsymbol{q}_j + d\boldsymbol{q}_j^T \boldsymbol{q}_i &= 0 \qquad 1 \leq i \neq j \leq m \end{aligned}$$

Therefore, $Q^T \hat{Q}^\star \approx I_2$ up to second-order errors $O(\{\|\epsilon_k^2\|\})$. Therefore, the rotation matrix used to calibrate $\hat{Q}^\star$ and Q when defining $\mathcal{D}(\hat{Q}^\star, Q)$ is the identity matrix up to second-order errors $O(\{\|\epsilon_k^2\|\})$. Applying (15), we have

$$\begin{aligned} \frac{\partial \{ D \} (\hat{Q}^\star, Q)}{\partial \epsilon_{ki}} &= \left(I_n - QQ^T \right) \left(\boldsymbol{e}_k \boldsymbol{x}_k^T \boldsymbol{q}_1 \lambda_1^{-1}, \cdots, \boldsymbol{e}_k \boldsymbol{x}_k^T \boldsymbol{q}_m \lambda_m^{-1} \right) \\ &= (I_n - QQ^T) \boldsymbol{e}_k \boldsymbol{x}_k^T Q \Lambda^{-1}. \end{aligned}$$

The proof under the case where the eigenvalues of Q are repeating is similar, except that the summation in (14) shall discard (i, j) pairs where $\lambda_i = \lambda_j$. On the other hand, the uncertainties in eigenvectors when having repeating eigenvalues are addressed by the calibration rotation matrix in $\mathcal{D}(\hat{Q}^\star, Q)$.

□