IAE: IMPLICIT AUTOENCODER FOR POINT CLOUD SELF-SUPERVISED REPRESENTATION LEARNING

Anonymous authors
Paper under double-blind review

ABSTRACT

Autoencoding has been a popular topic across many fields and recently emerged in the 3D domain. However, many 3D representations (e.g., point clouds) are discrete samples of the underlying continuous 3D surface which makes them different from other data modalities. This process inevitably introduces sampling variations on the underlying 3D shapes. In learning 3D representation, a desirable goal is to disregard such sampling variations while focusing on capturing transferable knowledge of the underlying 3D shape. This aim poses a grand challenge to existing representation learning paradigms. For example, the standard autoencoding paradigm forces the encoder to capture such sampling variations as the decoder has to reconstruct the original point cloud. In this paper, we introduce the Implicit Autoencoder (IAE). This simple yet effective method addresses this challenge by replacing the point cloud decoder with an implicit decoder. The implicit decoder can output a continuous representation that is shared among different point cloud samplings of the same model. Reconstructing under the implicit representation can prioritize that the encoder discards sampling variations, introducing appropriate inductive bias to learn more generalizable feature representations. We validate this claim experimentally and show a theoretical analysis under a simple linear autoencoder. Moreover, our implicit decoder offers excellent flexibility in designing 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.

1 INTRODUCTION

Point cloud provides a natural and flexible representation of 3D objects. The rapid development of 3D scanning devices and techniques enable 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 results on many tasks based on the point cloud representation, ranging from object-level understanding, including shape classification (Chang et al., 2015) and part segmentation (Yi et al., 2016), to scene-level understanding, such as 3D object detection (Dai et al., 2017; Song et al., 2015; Geiger et al., 2012) and 3D semantic segmentation (Armeni et al., 2016).

While these applications are essential, 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 aims to effectively utilize raw and unlabeled data to pre-train deep neural networks. The 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 (Erhan et al., 2010).

Substantial effort has been devoted to self-supervised learning methods for 2D images (Pathak et al., 2016; Wu et al., 2018; Doersch et al., 2015; Masci et al., 2011; Chen et al., 2020). Among this line, autoencoder is one of the most classical methods (Pathak et al., 2016; Bengio et al., 2013; Tschannen et al., 2018; Vincent et al., 2008; He et al., 2021). Autoencoder 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. The encoder is forced to learn multi-scale invariant features among the training data by training with the reconstruction loss. The performance of such an approach typically depends on the network architecture.

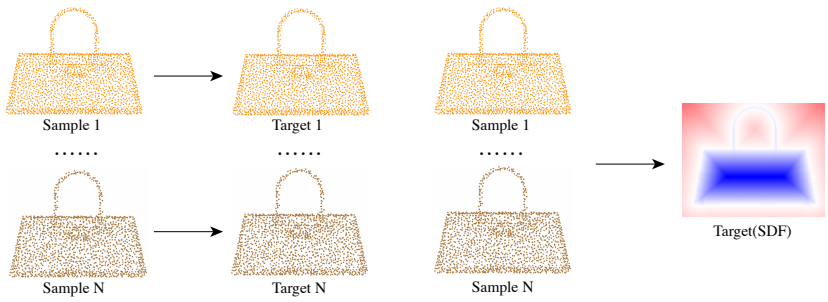


Figure 1: **Explicit Autoencoder versus Implicit Autoencoder (Ours)**. For the traditional explicit autoencoder, the model is forced to reconstruct the input one-to-one. For the implicit autoencoder, the reconstruction targets are the same.

Unlike the conventional structured data (*e.g.*, images), point clouds are unordered collections of points. There is increasing literature attempting to design suitable network architecture and learning algorithms for autoencoding on point clouds. For example, Yang et al. (Yang et al., 2018) proposed a point cloud autoencoder with a novel folding-based decoder. Wang et al. (Wang et al., 2020) designed a denoising autoencoder using a standard point cloud completion model. These works follow the design paradigm for image autoencoding, i.e., the decoder and the encoder presume the exact representation (point cloud in this case) as the input.

However, we observe a potential drawback in using point cloud as the decoder representation for autoencoding. Point clouds are spatially unstructured, discretized, and noisy representations of 3D shapes. In other words, the same 3D shape could be sampled into many different point clouds (See Figure 1), subject to noises induced from various sources such as the intrinsic noises from the sensors and/or the interference from the environment. When a point-cloud-based autoencoder is trained to capture this 3D shape, the encoder is forced to model these *sampling variations*, which we argue, are distractive information in understanding the underlying 3D shape.

To address this issue, this paper, for the first time, combines the implicit surface representation in point cloud self-supervised learning. This novel paradigm nicely addresses the issue of sampling variations. Specifically, we propose 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, which is a continuous representation shared among different point clouds sampled from the same model. Reconstructing under the implicit representation can prioritize that the encoder discards sampling variations, leaving more capacity to learn more generalizable features. Second, the learning process of IAE is guided by minimizing the discrepancy of two implicit functions, bypassing the computationally intensive and unstable explicit data association(*e.g.*, Earth Mover Distance (EMD) (Rubner et al., 2000) or Chamfer Distance (CD) (Fan et al., 2017)). Moreover, without the need to decode the whole point cloud, IAE is smaller and more resource-efficient. It 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.

To demonstrate the usefulness of IAE, we verify that the learned representation from our pre-trained model can be successfully adapted to various 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. Specifically, IAE achieves 92.1% classification accuracy on ModelNet40 linear evaluation(**1.2%** absolute improvement), and 94.2% accuracy on ModelNet40 fine-tuned evaluation(**1.1%** absolute improvement). Moreover, IAE, for the first time, successfully extends the autoencoding paradigm into scene-level tasks. For example, it achieves +6.3/+3.1 absolute improvements on mAP 0.5 compared with training from scratch on ScanNet/SUN RGB-D object detection tasks.

2 Related Work

Self-supervised Representation Learning on Point Clouds 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 (Wang et al., 2020; Sauder & Sievers, 2019; Yang et al., 2018; Poursaeed et al., 2020; Achlioptas et al., 2018; Hassani & Haley, 2019; Li et al., 2018; Chen et al., 2021; Yu et al., 2021). These methods mainly

pre-train on ShapeNet (Chang et al., 2015). However, the resulting models usually exhibit limited transferability to a real scene-level dataset, due to the synthetic-real domain gap (Xie et al., 2020).

With the success of contrastive learning in 2D images, some initial effort has been spent on this direction (Xie et al., 2020; Rao et al., 2021; Zhang et al., 2021). These methods generally require a careful strategy to define positive and negative pairs of the instances. Also, 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 as 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 works to ours are FoldingNet (Yang et al., 2018), OcCo (Wang et al., 2020), and ParAE (Eckart et al., 2021). FoldingNet designed a point cloud autoencoder with a novel folding-based decoder. OcCo proposed a pipeline to mask a point cloud from camera viewpoints and reconstruct the complete point cloud with a standard point cloud completion model (Yuan et al., 2018). Both works demonstrated promising gains from such pre-training. However, using point clouds as the decoder representation has several drawbacks. First, the discrete surface representation forces the autoencoder to learn irrelevant input sampling variations. 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 with an 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. ParAE (Eckart et al., 2021) proposed a pretext task to learn data distribution through discrete generative models. Their supervision is built on a point-wise partitioning matrix, while ours utilizes a simple implicit function that is easier to train.

Implicit Representations Recent works have investigated the implicit representation of continuous 3D shapes by optimizing deep networks that map 3D coordinates to signed distance (Michalkiewicz et al., 2019; Park et al., 2019) or occupancy grids (Chen & Zhang, 2019; Mescheder et al., 2019; Peng et al., 2020). 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 (Liu et al., 2020; 2019; Niemeyer et al., 2020), primitive-based 3D reconstruction (Genova et al., 2020; 2019; Paschalidou et al., 2020), 4D reconstruction (Niemeyer et al., 2019), and representing continuous texture (Oechsle et al., 2019). However, no prior work has integrated implicit representations into self-supervised representation learning on point clouds to the best of our knowledge.

3 3D Representation Learning with Implicit Autoencoding

This section introduces the details of IAE. We begin with describing the underlying principles of IAE in Section 3.1. Next, We describe the technical details of IAE in Section 3.2.

3.1 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 $d_{\text{exp}}(\cdot,\cdot) : \mathcal{R}^{n_0 \times 3} \times \mathcal{R}^{n_1 \times 3} \rightarrow \mathcal{R}$ between the input point cloud and the sub-sampled version of target point cloud $\mathcal{P}^{\text{gt}}_{\text{sub}} \in \mathcal{R}^{n_1 \times 3}$:

$$\min_{\Theta,\Phi} d_{\text{exp}}((g_\Phi \circ f_\Theta)(\mathcal{P}^{\text{in}}), \mathcal{P}^{\text{gt}}_{\text{sub}}). \tag{1}$$

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, where $\mathcal{P}^{\text{gt}}_{\text{sub}}$ 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 correspondences between two point clouds.

Implicit Autoencoder. In contrast to an Explicit Autoencoder which needs to predict the full explicit representation using f_Θ, an Implicit Autoencoder outputs an implicit function $(g_\Phi \circ f_\Theta)(x|\mathcal{P}^{\text{in}}) : \mathcal{R}^3 \rightarrow \mathcal{R}$, where $x \in \mathcal{R}^3$ denotes the query point. This function is conditioned on the input point cloud $\mathcal{P}^{\text{in}}$. We further define the ground truth implicit function as $g_0(x) : \mathcal{R}^3 \rightarrow \mathcal{R}$. In practice, g_0

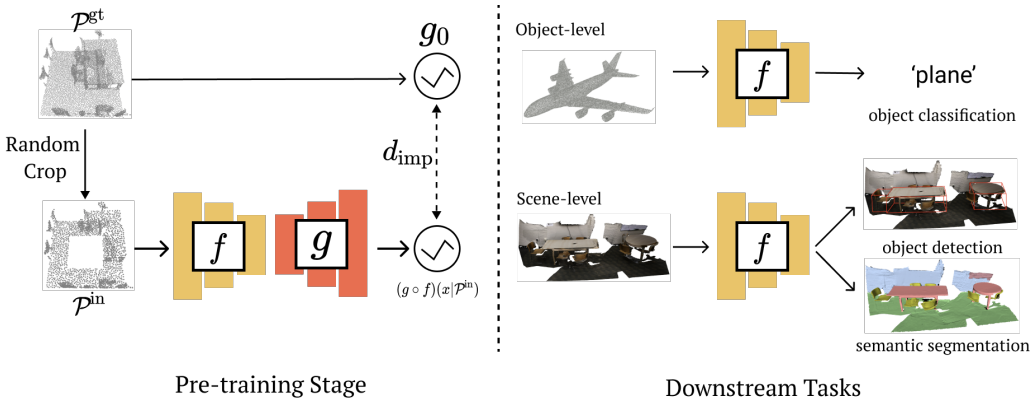


Figure 2: **Pipeline of the Implicit AutoEncoder.** Left: **Pre-training stage.** Given a point cloud $\mathcal{P}^{\text{gt}}$, we apply a random center crop for the input point cloud $\mathcal{P}^{\text{in}}$. Backbone module f encodes the input. Prediction module g takes embedding features from f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

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:

$$\min_{\Theta,\Phi} d_{\text{imp}}((g_\Phi \circ f_\Theta)(x|\mathcal{P}^{\text{in}}), g_0(x)) \tag{2}$$

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 the L_1 distance. On the other hand, we choose d_{imp} to be a cross-entropy loss when g_0 is the occupancy grid.

3.2 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 4.2.

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

Output Encoding. The formulation of Eq 2 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 experimented with different occupancy functions as well.

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:

$$\mathcal{L} = \frac{1}{N}\sum_{i=0}^{N} \|(g_\Phi \circ f_\Theta)(x_i|\mathcal{P}^{\text{in}}) - g_0(x_i)\| \tag{3}$$

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

Data Augmentation. To help the model capture high-level semantic features, we design the data augmentation by randomly center-cropping 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 in Section 4.3.

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

Method	ModelNet40
3D-GAN	83.3%
Latent-GAN	85.7%
SO-Net	87.3%
MAP-VAE	88.4%
Jigsaw*	84.1%
FoldingNet*	90.1%
Orientation*	90.7%
STRL*	90.9%
OcCo*	89.7%
IAE(ours)	**92.1%**

Table 1: **Linear evaluation for shape classification on ModelNet40**. For fair comparisons, different * methods use the same DGCNN encoder backbone.

Category	Method	ModelNet40
Supervised	PointNet	89.2%
	PointNet++	90.7%
	PointCNN	92.2%
	KPConv	92.9%
	DGCNN	92.9%
	PointTransform	93.7%
Self-Supervised	FoldingNet	93.1%
	STRL	93.1%
	OcCo	93.0%
	IAE(ours)	**94.2%**

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

4 EXPERIMENTS

We begin with the pre-training setting of IAE on different datasets in Section 4.1. Next, we evaluate our models on various downstream tasks in Section 4.2. Section 4.3 and 4.4 present an ablation study and an experiment analysis of IAE, respectively.

4.1 PRE-TRAINING DATASET

ShapeNet (Chang et al., 2015) contains 57,748 synthetic 3D shapes from 55 categories. We follow the procedure of (Park et al., 2019; Mescheder et al., 2019) 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 (Yang et al., 2018). ShapeNet is used for evaluating shape classification.

ScanNet (Dai et al., 2017) 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 (Qi et al., 2019), we extract 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 design a nearest neighbor strategy to obtain the true labels. During training, we force the network to generate unsigned distance values by taking absolute values at the output layer. Supplementary material provides more details. ScanNet is used for evaluating indoor 3D object detection and 3D semantic segmentation.

4.2 DOWNSTREAM TASKS

In this section, we present the experiment settings and results for each downstream task.

4.2.1 SHAPE CLASSIFICATION

Following the standard protocols from previous work (Qi et al., 2017), we evaluate the shape feature learning of our model on ModelNet40 (Wu et al., 2015). ModelNet has two variants, i.e., ModelNet40 and ModelNet10. ModelNet40 consists of 9832 training objects and 2468 test objects in 40 classes. ModelNet10 consists of 3991 training objects, 908 test objects in 10 classes. We pre-process the training datasets by following (Qi et al., 2017). 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 (Wang et al., 2019) as the encoder backbone, following the practice of previous approaches (Huang et al., 2021; Wang et al., 2020; Poursaeed et al., 2020). We randomly sample 2048 points from each shape for both pre-training and SVM training. Table 1 shows the results. IAE achieves 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 demonstrate the transferability of our model.

Supervised Fine-tuning. We proceed to evaluate fine-tuning results from our pre-trained model. Specifically, we use our pre-trained model as the initialization weights of the DGCNN encoder and

Method	ScanNet		SUN RGB-D	
	AP_{50}	AP_{25}	AP_{50}	AP_{25}
VoteNet (Qi et al., 2019)	33.5	58.6	32.9	57.7
STRL(Huang et al., 2021)	38.4	59.5	35.0	58.2
RandomRooms(Rao et al., 2021)	36.2	61.3	35.4	59.2
PointContrast(Xie et al., 2020)	38.0	59.2	34.8	57.5
DepthContrast[2](Zhang et al., 2021)	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-tune our pre-trained model on ScanNetV2 and SUN-RGBD validation set using VoteNet (Qi et al., 2019). 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.

Method	OA	mIoU
DGCNN	84.1	56.1
Jigsaw	84.4	56.6
OcCo	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, we report the classification accuracy.

then fine-tune it on ModelNet40. The results are shown in Table 2. We can see that IAE shows the best accuracy (94.2%) among other self-supervised methods under the same DGCNN backbone.

4.2.2 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 (Xie et al., 2020), 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 approaches (Sauder & Sievers, 2019; Poursaeed et al., 2020; Wang et al., 2020) 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 (Peng et al., 2020). Specifically, we use VoteNet (Qi et al., 2019) 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(Song et al., 2015). 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.

4.2.3 Indoor 3D Semantic Segmentation

We further evaluate our model on the indoor semantic segmentation task. We use Stanford Large-Scale 3D Indoor Spaces (S3DIS)(Armeni et al., 2016) 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 0.9% and 3.8% on OA and mIoU.

4.2.4 Cross-domain generalizability

Utilizing synthetic CAD object models to help 3D real data tasks (e.g., object detection) remains an open problem in 3D computer vision. Xie et al. (Xie et al., 2020) provided a failure object detection

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

	Method	MN40
EAE	FoldingNet	90.1%
	OcCo	89.7%
	Snowflake	89.9%
IAE	OccNet	91.5%
	ConvONet	**92.1%**

(a)

	Function	MN40
EAE	PC	90.1%
IAE	Occ	91.3%
	UDF	91.7%
	SDF	**92.1%**

(b)

	Method	SUN
EAE	FoldingNet	58.2
	OcCo	58.4
	Snowflake	58.1
IAE	ConvOccNet	**60.4**

(c)

Table 6: (a).**Ablation study on different decoder models.** On ModelNet40, we show linear evaluation results. Our implicit autoencoders(IAE) consistently outperform explicit counterparts(EAE). (b). **Ablation study on implicit function.** For explicit representation, we use FoldingNet as the decoder and point cloud(PC) at the output. For implicit representations, we experimented with Occupancy Value (Occ Value), Unsigned Distance Function (UDF), and Signed Distance Function (SDF), which show consistent improvements over the explicit representation. (c). **Comparison between explicit approaches and our model** on real data completion task. Our approach using the convolutional occupancy network shows consistent improvements.

Data Aug	scratch	cs=0	cs=0.2	cs=0.5	cs=0.7
SUN RGB-D	57.7	60.0	60.2	**60.4**	59.8

Table 7: **Different setting of data augmentation.** 'cs' denotes cropping size. 'scratch' means training from scratch.

case when pre-training on ShapeNet and fine-tuning on ScanNet. However, recently, Huang et al. (Huang et al., 2021) reported an opposite observation and attributed the failure reason of (Xie et al., 2020) to the simple encoder architecture. Since it is much easier to access a large-scale synthetic data, it is 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. (Huang et al., 2021) and demonstrates the effective cross-domain generalizability of our model.

Conversely, we also report linear evaluation results on the ModelNet40. It is interesting to observe the transferability from natural scenes to the synthetic shape domain. Surprisingly, pre-training on ScanNet achieves a comparable result with 91.1% accuracy. This result still outperforms previous state-of-the-art methods and demonstrates the strong transferability of our model.

4.3 Ablation Study

This section presents an ablation study 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 experiment with different decoder models from both categories. Specifically, we study three state-of-the-art explicit decoder models of FoldingNet (Yang et al., 2018), OcCo (Wang et al., 2020), and SnowflakeNet (Xiang et al., 2021), and two implicit decoder models of Occupancy Network (Mescheder et al., 2019) and Convolutional Occupancy Network (Peng et al., 2020). 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(MN40) using linear SVM. The results are shown in Table 6(a). Interestingly, 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(b), all of them show better results than the explicit representation. Among those implicit representations, signed distance function offers the best performance.

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 conducted all the experiments on ScanNet and fine-tuned 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 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.

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 (Xiang et al., 2021) on ScanNet. Then we fine-tune models on SUN RGB-D detection. We show the result in Table 6(c). Still, our model outperforms these three explicit approaches consistently.

4.4 Further Experimental Analysis

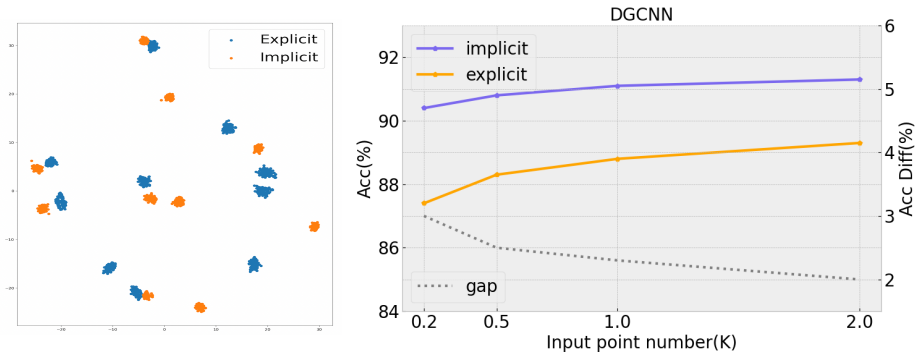


Figure 3: **Comparison between explicit and implicit auto-encoder.** Left: T-SNE visualization for encoder features of EAE and IAE. Right: 'Acc Diff' denotes the accuracy difference between two models on ModelNet40 Linear. When increasing the input resolution, the gap between explicit and implicit models decreases.[3].

We show two experiments to further study why IAE shows better result than explicit autoencoders. First, we randomly choose 10 shapes from ShapeNet. And for each shape, we randomly sample 100 point clouds. Then we take these samples as the input to the pretrained explicit autoencoder and the implicit one. Then we visualize the output global features of the encoders. As shown in the Figure 3 Left, we can see that IAE shows tighter feature clusters of each shape while EAE is more scatter. This experiment validates that under IAE, the learned encoder is less sensitive to sampling variations than the one under explicit representations.

As discussed in Section 1, 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 3.1, first, on ShapeNet, we generate four datasets by sampling different number of 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 four 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 3 Right. 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 resolutions is not that significant.

On the other hand, 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.

[3]Both models show slightly worse performance compared to Table 1, as we do not add data augmentations.

5 ANALYSIS OF IAE UNDER A LINEAR AE MODEL

We proceed to gain more insights of IAE under a linear AE model. 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 among 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 $\boldsymbol{x}_i$:

$$A^\star, Q^\star = \underset{A', Q' \in \mathbb{R}^{n \times m}}{\operatorname{argmin}} \sum_{k=1}^{N} \| A' {Q'}^T \boldsymbol{x}'_k - \boldsymbol{x}_k \|^2 \quad s.t.\ {Q'}^T Q' = I_m,\ Q' \in \{ C \}(X') \tag{4}$$

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 $O(m)$. Below we show that $Q^\star$ is independent of ϵ_k.

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

Prop. 1 indicates that $Q^\star$ does not encode sampling variations.

Now consider the second model where we force the encoder-decoder pair to reconstruct the original inputs, which is analogous to standard autoencoding:

$$\hat{A}^\star, \hat{Q}^\star = \underset{A', Q' \in \mathbb{R}^{n \times m}}{\operatorname{argmin}} \sum_{k=1}^{N} \| A' {Q'}^T \boldsymbol{x}'_k - \boldsymbol{x}'_k \|_{\mathcal{F}}^2 \quad s.t.\ {Q'}^T Q' = I_m,\ Q' \in \{ C \}(X') \tag{5}$$

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

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*

$$\mathcal{D}(Q_1, Q_2) := Q_1 - Q_2 R^\star, \qquad R^\star = \underset{R \in O(m)}{\operatorname{argmin}} \| Q_1 - Q_2 R \|_{\mathcal{F}}^2 \tag{6}$$

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

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

$$\frac{\partial \mathcal{D}(\hat{Q}^\star, Q)}{\partial \epsilon_{ki}} = (I_n - Q Q^T)(\boldsymbol{e}_i \boldsymbol{x}_k^T) Q \Lambda^+ \tag{7}$$

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 standard basis vector.

In other word, $\hat{Q}^\star$ is sensitive to ϵ_k. Therefore, it encodes sampling variations. This theoretical analysis under a simplified setting suggests that IAE may potentially learn more robust representations.

6 CONCLUSIONS AND LIMITATIONS

In this paper, we propose IAE, a simple yet effective self-supervised learning framework for point clouds. Unlike the conventional autoencoder for point cloud which reconstructs input point clouds explicitly, we reconstruct the implicit representation. We argue that IAE can prioritize that the encoder discards sampling variations, introducing more capacity to learn more generalizable features. We find 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. One limitation is that we require a pre-processing step on point clouds to get implicit representations. Another limitation is that we only experimented with hand-crafted implicit function targets, such as signed distance function. Jointly learning implicit function targets may bring more improvements.

REFERENCES

Panos Achlioptas, Olga Diamanti, Ioannis Mitliagkas, and Leonidas Guibas. Learning representations and generative models for 3d point clouds. In *International conference on machine learning*, pp. 40–49. PMLR, 2018.

Iro Armeni, Ozan Sener, Amir R Zamir, Helen Jiang, Ioannis Brilakis, Martin Fischer, and Silvio Savarese. 3d semantic parsing of large-scale indoor spaces. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 1534–1543, 2016.

Yoshua Bengio, Aaron Courville, and Pascal Vincent. Representation learning: A review and new perspectives. *IEEE transactions on pattern analysis and machine intelligence*, 35(8):1798–1828, 2013.

Angel X Chang, Thomas Funkhouser, Leonidas Guibas, Pat Hanrahan, Qixing Huang, Zimo Li, Silvio Savarese, Manolis Savva, Shuran Song, Hao Su, et al. Shapenet: An information-rich 3d model repository. *arXiv preprint arXiv:1512.03012*, 2015.

Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. A simple framework for contrastive learning of visual representations. In *International conference on machine learning*, pp. 1597–1607. PMLR, 2020.

Ye Chen, Jinxian Liu, Bingbing Ni, Hang Wang, Jiancheng Yang, Ning Liu, Teng Li, and Qi Tian. Shape self-correction for unsupervised point cloud understanding. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 8382–8391, 2021.

Zhiqin Chen and Hao Zhang. Learning implicit fields for generative shape modeling. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 5939–5948, 2019.

Angela Dai, Angel X Chang, Manolis Savva, Maciej Halber, Thomas Funkhouser, and Matthias Nießner. Scannet: Richly-annotated 3d reconstructions of indoor scenes. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 5828–5839, 2017.

Carl Doersch, Abhinav Gupta, and Alexei A Efros. Unsupervised visual representation learning by context prediction. In *Proceedings of the IEEE international conference on computer vision*, pp. 1422–1430, 2015.

Benjamin Eckart, Wentao Yuan, Chao Liu, and Jan Kautz. Self-supervised learning on 3d point clouds by learning discrete generative models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 8248–8257, 2021.

Dumitru Erhan, Aaron Courville, Yoshua Bengio, and Pascal Vincent. 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.

Haoqiang Fan, Hao Su, and Leonidas J Guibas. 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.

Andreas Geiger, Philip Lenz, and Raquel Urtasun. 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.

Kyle Genova, Forrester Cole, Daniel Vlasic, Aaron Sarna, William T Freeman, and Thomas Funkhouser. Learning shape templates with structured implicit functions. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 7154–7164, 2019.

Kyle Genova, Forrester Cole, Avneesh Sud, Aaron Sarna, and Thomas Funkhouser. Local deep implicit functions for 3d shape. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 4857–4866, 2020.

Kaveh Hassani and Mike Haley. Unsupervised multi-task feature learning on point clouds. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 8160–8171, 2019.

Kaiming He, Xinlei Chen, Saining Xie, Yanghao Li, Piotr Dollár, and Ross Girshick. Masked autoencoders are scalable vision learners. *arXiv preprint arXiv:2111.06377*, 2021.

Siyuan Huang, Yichen Xie, Song-Chun Zhu, and Yixin Zhu. 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.

Jiaxin Li, Ben M Chen, and Gim Hee Lee. 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.

Shaohui Liu, Yinda Zhang, Songyou Peng, Boxin Shi, Marc Pollefeys, and Zhaopeng Cui. 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.

Shichen Liu, Shunsuke Saito, Weikai Chen, and Hao Li. Learning to infer implicit surfaces without 3d supervision. *arXiv preprint arXiv:1911.00767*, 2019.

Jonathan Masci, Ueli Meier, Dan Cireşan, and Jürgen Schmidhuber. Stacked convolutional auto-encoders for hierarchical feature extraction. In *International conference on artificial neural networks*, pp. 52–59. Springer, 2011.

Lars Mescheder, Michael Oechsle, Michael Niemeyer, Sebastian Nowozin, and Andreas Geiger. 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.

Mateusz Michalkiewicz, Jhony K Pontes, Dominic Jack, Mahsa Baktashmotlagh, and Anders Eriksson. Implicit surface representations as layers in neural networks. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 4743–4752, 2019.

Michael Niemeyer, Lars Mescheder, Michael Oechsle, and Andreas Geiger. Occupancy flow: 4d reconstruction by learning particle dynamics. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 5379–5389, 2019.

Michael Niemeyer, Lars Mescheder, Michael Oechsle, and Andreas Geiger. 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.

Michael Oechsle, Lars Mescheder, Michael Niemeyer, Thilo Strauss, and Andreas Geiger. Texture fields: Learning texture representations in function space. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 4531–4540, 2019.

Jeong Joon Park, Peter Florence, Julian Straub, Richard Newcombe, and Steven Lovegrove. 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.

Despoina Paschalidou, Luc Van Gool, and Andreas Geiger. 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.

Deepak Pathak, Philipp Krahenbuhl, Jeff Donahue, Trevor Darrell, and Alexei A Efros. Context encoders: Feature learning by inpainting. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 2536–2544, 2016.

Songyou Peng, Michael Niemeyer, Lars Mescheder, Marc Pollefeys, and Andreas Geiger. 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.

Omid Poursaeed, Tianxing Jiang, Han Qiao, Nayun Xu, and Vladimir G Kim. Self-supervised learning of point clouds via orientation estimation. In *2020 International Conference on 3D Vision (3DV)*, pp. 1018–1028. IEEE, 2020.

Charles R Qi, Hao Su, Kaichun Mo, and Leonidas J Guibas. Pointnet: Deep learning on point sets for 3d classification and segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 652–660, 2017.

Charles R Qi, Or Litany, Kaiming He, and Leonidas J Guibas. 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.

Yongming Rao, Benlin Liu, Yi Wei, Jiwen Lu, Cho-Jui Hsieh, and Jie Zhou. 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.

Yossi Rubner, Carlo Tomasi, and Leonidas J Guibas. The earth mover's distance as a metric for image retrieval. *International journal of computer vision*, 40(2):99–121, 2000.

Jonathan Sauder and Bjarne Sievers. Self-supervised deep learning on point clouds by reconstructing space. *arXiv preprint arXiv:1901.08396*, 2019.

Shuran Song, Samuel P Lichtenberg, and Jianxiong Xiao. 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.

Michael Tschannen, Olivier Bachem, and Mario Lucic. Recent advances in autoencoder-based representation learning. *arXiv preprint arXiv:1812.05069*, 2018.

Pascal Vincent, Hugo Larochelle, Yoshua Bengio, and Pierre-Antoine Manzagol. Extracting and composing robust features with denoising autoencoders. In *Proceedings of the 25th international conference on Machine learning*, pp. 1096–1103, 2008.

Hanchen Wang, Qi Liu, Xiangyu Yue, Joan Lasenby, and Matthew J Kusner. Unsupervised point cloud pre-training via view-point occlusion, completion. *arXiv preprint arXiv:2010.01089*, 2020.

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

Zhirong Wu, Shuran Song, Aditya Khosla, Fisher Yu, Linguang Zhang, Xiaoou Tang, and Jianxiong Xiao. 3d shapenets: A deep representation for volumetric shapes. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 1912–1920, 2015.

Zhirong Wu, Yuanjun Xiong, Stella X Yu, and Dahua Lin. Unsupervised feature learning via non-parametric instance discrimination. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 3733–3742, 2018.

Peng Xiang, Xin Wen, Yu-Shen Liu, Yan-Pei Cao, Pengfei Wan, Wen Zheng, and Zhizhong Han. 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.

Saining Xie, Jiatao Gu, Demi Guo, Charles R Qi, Leonidas Guibas, and Or Litany. Pointcontrast: Unsupervised pre-training for 3d point cloud understanding. In *European Conference on Computer Vision*, pp. 574–591. Springer, 2020.

Yaoqing Yang, Chen Feng, Yiru Shen, and Dong Tian. 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.

Li Yi, Vladimir G Kim, Duygu Ceylan, I-Chao Shen, Mengyan Yan, Hao Su, Cewu Lu, Qixing Huang, Alla Sheffer, and Leonidas Guibas. A scalable active framework for region annotation in 3d shape collections. *ACM Transactions on Graphics (ToG)*, 35(6):1–12, 2016.

Xumin Yu, Lulu Tang, Yongming Rao, Tiejun Huang, Jie Zhou, and Jiwen Lu. Point-bert: Pre-training 3d point cloud transformers with masked point modeling. *arXiv preprint arXiv:2111.14819*, 2021.

Wentao Yuan, Tejas Khot, David Held, Christoph Mertz, and Martial Hebert. Pcn: Point completion network. In *2018 International Conference on 3D Vision (3DV)*, pp. 728–737. IEEE, 2018.

Zaiwei Zhang, Rohit Girdhar, Armand Joulin, and Ishan Misra. Self-supervised pretraining of 3d features on any point-cloud. *arXiv preprint arXiv:2101.02691*, 2021.

A SUPPLEMENTARY MATERIALS

Our supplementary material provides proof of propositions in Section A.1, additional implementation details on data generation and pre-training Section A.2, as well as more experiment results and analysis in Section A.3.

A.1 PROOF OF PROPOSITIONS IN SECTION 5

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

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

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

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

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

Therefore, define

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

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

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

Therefore, the optimization problem in (9) reduces to

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

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 $C_{X',X} := \Big(X'X^T\Big)^T\Big(X'X'^T\Big)^+\Big(X'X^T\Big)$. □

A.1.1 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 orthonormal 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,

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

Moreover,

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

□

A.1.2 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

$$\begin{aligned}
d\boldsymbol{q}_i &= -\sum_{j\neq i}\frac{\boldsymbol{q}_j^T\sum_{k=1}^{N}(\boldsymbol{x}_k\epsilon_k^T+\epsilon_k\boldsymbol{x}_k^T)\boldsymbol{q}_i}{\lambda_j-\lambda_i}\boldsymbol{q}_j \qquad (14)\\
&= -\sum_{j\neq i,j=1}^{m}\frac{\boldsymbol{q}_j^T\sum_{k=1}^{N}(\boldsymbol{x}_k\epsilon_k^T+\epsilon_k\boldsymbol{x}_k^T)\boldsymbol{q}_i}{\lambda_j-\lambda_i}\boldsymbol{q}_j-\sum_{j=m+1}^{n}\frac{\boldsymbol{q}_j^T\sum_{k=1}^{N}(\boldsymbol{x}_k\epsilon_k^T+\epsilon_k\boldsymbol{x}_k^T)\boldsymbol{q}_i}{\lambda_j-\lambda_i}\boldsymbol{u}_j\\
&= -\sum_{j=m+1}^{n}\frac{\boldsymbol{q}_j^T\sum_{k=1}^{N}\epsilon_k\boldsymbol{x}_k^T\boldsymbol{q}_i}{\lambda_j-\lambda_i}\boldsymbol{q}_j\\
&= \big(\sum_{j=m+1}^{n}\boldsymbol{q}_j\boldsymbol{q}_j^T\big)\big(\sum_{k=1}^{N}\epsilon_k\boldsymbol{x}_k^T\big)\boldsymbol{q}_i\lambda_i^{-1} = \big(I_n-QQ^T\big)\sum_{k=1}^{N}\epsilon_k\boldsymbol{x}_k^T\boldsymbol{q}_i\lambda_i^{-1}. \qquad (15)
\end{aligned}$$

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

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

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 $\mathcal{D}(\hat{Q}^\star, Q)$.

□

A.2 Implementation Details

A.2.1 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

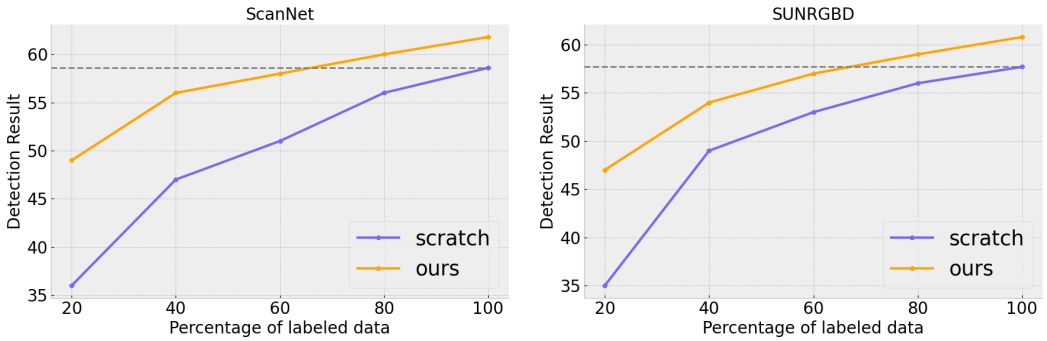
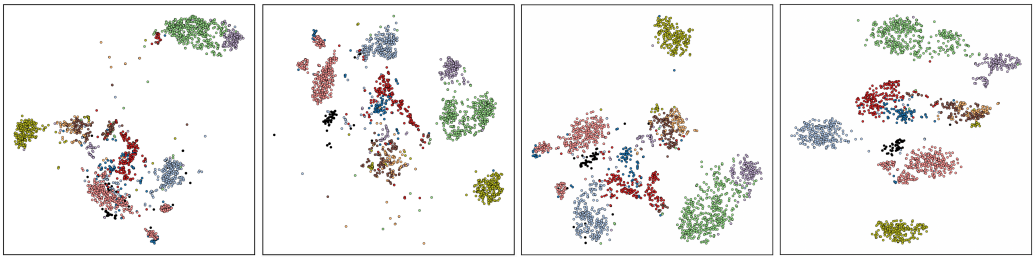


Figure 6: **Label efficiency training.** We pre-train our model on ScanNet and then fine-tune on ScanNet and SUN RGB-D separately. During fine-tuning, different percentages of labeled data are used. Our pre-training model outperforms training from scratch and achieves nearly the same result with only 60% labeled data.



(a) Random (b) FoldingNet (c) OcCo (d) IAE(ours)

Figure 7: **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.

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.

A.2.2 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.

A.3 More Results

A.3.1 Label Efficiency Training

Pretraining helps models to be fine-tuned with small amount of labeled data. 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 6. 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.

A.3.2 Embedding Visualization.

We visualize the learned features of our model and baseline approaches in Figure 7. We compare with FoldingNet (Yang et al., 2018), OcCo (Wang et al., 2020), 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 (Yang et al., 2018), OcCo (Wang et al., 2020), and random initialization.