Self-Supervised Learning on 3D Point Clouds by Learning Discrete Generative Models

Benjamin Eckart[1] Wentao Yuan[2] Chao Liu[1] Jan Kautz[1]
[1]NVIDIA [2]University of Washington

Abstract

While recent pre-training tasks on 2D images have proven very successful for transfer learning, pre-training for 3D data remains challenging. In this work, we introduce a general method for 3D self-supervised representation learning that 1) remains agnostic to the underlying neural network architecture, and 2) specifically leverages the geometric nature of 3D point cloud data. The proposed task softly segments 3D points into a discrete number of geometric partitions. A self-supervised loss is formed under the interpretation that these soft partitions implicitly parameterize a latent Gaussian Mixture Model (GMM), and that this generative model establishes a data likelihood function. Our pretext task can therefore be viewed in terms of an encoder-decoder paradigm that squeezes learned representations through an implicitly defined parametric discrete generative model bottleneck. We show that any existing neural network architecture designed for supervised point cloud segmentation can be repurposed for the proposed unsupervised pretext task. By maximizing data likelihood with respect to the soft partitions formed by the unsupervised point-wise segmentation network, learned representations are encouraged to contain compositionally rich geometric information. In tests, we show that our method naturally induces semantic separation in feature space, resulting in state-of-the-art performance on downstream applications like model classification and semantic segmentation.

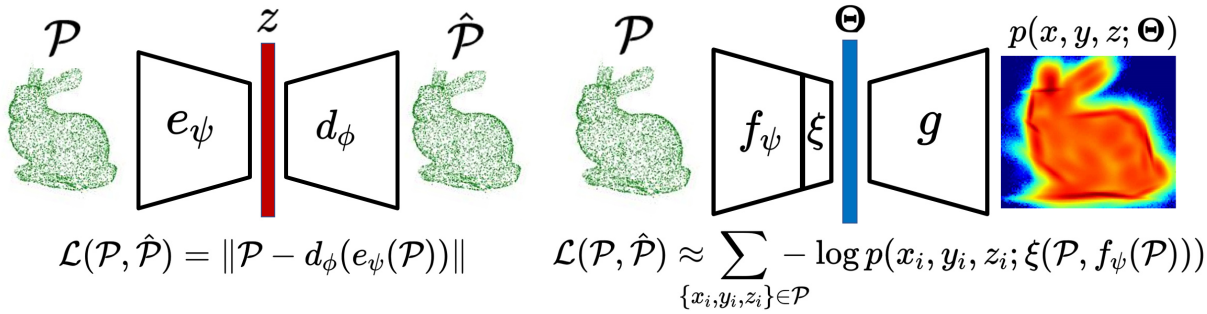
1. Introduction

There has been a growing emergence of increasingly effective self-supervised learning methods developed in the form of unsupervised pretext tasks. Instead of human-annotated supervision, the pretext task itself is designed to create its own supervisory signal. For example, learning to predict or discriminate data augmentations that preserve the semantics of the input have recently been shown to yield rich latent representations for downstream tasks [18, 6, 34, 33, 14]. One of the longstanding goals of unsupervised learning has been to improve transfer learning to the point where unsupervised pre-training combined with downstream supervision outperforms the traditional fully supervised training pipeline. In recent years, we've seen this come to fruition in several domains, with methods like BERT [11] for NLP, and BYOL [18], SimCLR [6, 7], CPC [34, 22], and others claiming top performance on image classification benchmarks.

Compared to deep learning for NLP and 2D computer vision, deep learning for 3D perception remains a relatively nascent field and self-supervised 3D learning even more so. One could argue there are several reasons for this, but the most salient is perhaps the lack of a common representation: while NLP has word embeddings and 2D computer vision has 2D images, 3D data enjoys no such universal and obvious data structure. The basic representation for 3D data is extremely fractured: for 3D content creation, triangular meshes are the *de facto* standard, yet most 3D sensors produce raw data in the form of 3D point clouds. Traditional 3D vision algorithms leverage structures like Octrees [23] or Hashed Voxel Lists [32], but deep learning approaches favor structures more amenable to differentiability and/or efficient neural processing, like sparse voxel networks [8, 44], implicit functions [30], graphs [27], or point-based networks [36, 37, 46]. Thus, it is still an active area of research to find the proper universal "3D backbone" that can become as ubiquitous as ResNet [21] is for learning representations of 2D images.

In terms of self-supervised learning on 3D data, the lack of standardization of basic 3D data processing further magnifies these issues since any technique designed for a particular architecture or data representation might have limited long-term utility. Furthermore, though it is relatively easy to obtain large amounts of unlabeled 3D data given the recent proliferation of self-driving cars and cheap commodity 3D sensors [53, 25], it can be difficult and time-consuming to produce accurate ground truth 3D annotations. Thus, it could be argued that the relative need for strong performing self-supervised methods for 3D data is much higher than in the 2D regime.

In this work, our goal is to try to devise a self-supervised



(a) **Traditional Autoencoder Bottleneck** (b) **Proposed "Parametric" Bottleneck**

Figure 1. In a traditional autoencoder paradigm, the input data $\mathcal{P}$ is passed through a latent bottleneck z from which the original input is explicitly reconstructed. That is, $\hat{\mathcal{P}} = d_\phi(z)$ and $z = e_\psi(\mathcal{P})$. The autoencoding task is then defined by a loss according to a predefined distance measure between the input $\mathcal{P}$ and output point cloud $\hat{\mathcal{P}}$ (*e.g.* Chamfer Loss). In our proposed setup, the bottleneck layer has an interpretable form as the constituent parameters $\boldsymbol{\Theta}$ of a discrete generative model (GMM). Given any point-wise classification network f_ψ, we design a fixed parameter-less function ξ such that $\boldsymbol{\Theta} = \xi(\mathcal{P}, f_\psi(\mathcal{P}))$. The PDF over 3D space is defined by $g(\cdot; \boldsymbol{\Theta})$. Instead of explicitly generating $\hat{\mathcal{P}}$ by *iid* sampling from the generative model g, we can use the negative log likelihood of the input $\mathcal{P}$ to directly impose a loss.

3D representation learning method that 1) remains agnostic to the specific choice of 3D representation or the underlying neural network architecture, and 2) specifically leverages the geometric nature of 3D point cloud data. To this end, we propose a pre-training task that can be applied to any off-the-shelf network architecture that outputs point-wise classification scores (*e.g.* logits), which we connect to the geometric nature of 3D point clouds by re-interpreting these classification scores in the context of probabilistic geometric spatial partition assignments. Any network designed for common point-wise classification tasks like semantic segmentation [3, 2] or part segmentation [4] can be leveraged without modification for our proposed 3D representation learning task, regardless of whether the architecture's underlying representation uses voxels, SDFs, graphs, unstructured points, etc.

Unlike a traditional supervised semantic segmentation paradigm, however, we have no supervision in the form of per-point class labels for ground truth partition assignment. Thus, if we wish to utilize segmentation networks to learn representations in an unsupervised fashion, we need to create something like "pseudo-labels" automatically from the data itself that we can compare against during training in order to develop a self-supervised loss function.

One way to solve this problem is by the so-called jigsaw type approaches (*e.g.* [33, 41]) that coarsely discretize the input space and create this pseudo-label directly from the voxel-id that each particular point falls into. Then, the voxels are randomly permuted and the neural network's task is to classify the voxel-id of each point. Though these have seen success in the 3D domain [41], the voxel-wise permutation operation leads to a destruction of the point cloud's original overall global geometry, even when its global geometry is arguably the strongest semantic cue. The jigsaw method therefore must rely on learning local (intra-voxel) features in order to position points globally, such that the global layout is only implicitly learned as a byproduct of local feature learning.

Compared to jigsaw tasks that learn to reassemble pictures or point clouds from a set of permuted disjoint partitions/voxels, our proposed pretext task learns the partitioning function itself to softly assign point clouds into geometrically coherent overlapping clusters. In doing so, we avoid any augmentation of the data that might degrade its geometric coherency and thus its semantic information. However, this leads to the question: what differentiates a "good" partitioning from a "bad" partitioning? Here, we take inspiration from recent work on deep learning for point cloud registration [52], where given two point clouds offset by an unknown rotation and translation, a point cloud segmentation network is used to implicitly infer a pair of latent and transformation-equivariant Gaussian Mixture Models (GMMs). In contrast to this work, however, we remove the registration objective completely and instead adapt this method to work only on a single point cloud without any spatial transformation or data augmentation necessary. To do this, we directly utilize the data likelihood of the implicitly defined GMM.

Given N points and J "pseudo-classes" and the $N \times J$ matrix of logits $\mathcal{S}$ from an underlying point cloud segmentation network, we propose dual interpretations of $\mathcal{S}$:
1) $\mathcal{S}$ partitions 3D space: $\mathcal{S}$ defines a soft spatial partitioning in 3D space of the input data into J discrete partitions.
2) $\mathcal{S}$ predicts latent posteriors: $\mathcal{S}$ calculates the posterior log probabilities of a set of latent binary variables that correspond each point to one of J components of a latent gen-

erative mixture model.

If we allow the latent model in the second interpretation to be implicitly defined through the first interpretation, we can assert "goodness-of-fit" evaluations by calculating the total data likelihood of the input point cloud with respect to the spatial density defined by the mixture model. In doing so, our self-supervision becomes the likelihood of this discrete generative model with respect to the input. One can view this proposed design as learning to probabilistically autoencode through a parametric discrete generative model bottleneck (termed "parametric bottleneck" elsewhere in this paper), where instead of a learned decoder, as is normally the case when autoencoding, we directly utilize the 3D PDF induced by the specific values of the bottleneck layer. Refer to Figure 1 for a graphical depiction of these differences. In summary, for a given point cloud, our pretext task is to learn the parameters of a point-wise classification network such that its output $\mathcal{S}$ produces the most likely spatial partitioning of these 3D points with respect to the discrete generative model implicitly defined by $\mathcal{S}$.

2. Related Work

Contrastive Approaches in 2D: Contrastive learning operates under the idea that a well-structured latent space can be trained in a metric learning sense by pushing together learned representations derived from inputs that represent the same semantic content and pushing away learned representations from inputs that have different semantics [28, 34, 22, 18, 31]. In practice, this involves hand-designing a set of data augmentations that do not degrade the semantic content in the input data such that positive input data pairs can be formed by repeatedly applying these augmentations on the input (*e.g.*, color distortion, blur, or rotation), and negative pairs can be mined from the mini-batch [6, 7] or a memory bank [20].

Contrastive Approaches in 3D: Recognizing the superior performance of contrastive learning approaches in 2D, PointContrast [49] by Xie *et al*. was the first to systematically research the efficacy of the contrastive paradigm for 3D representation learning. Using FCGF [9] by Choy *et al*. as their backbone, their data augmentation scheme consists of taking different views of the same 3D scene (by taking different frames of an RGBD video stream provided by the ScanNet dataset [10]) and applying scaling, rotation, and translation to the points within these views. Positive pairs are constructed at the point level by corresponding points between frames using nearest neighbor search.

In all these approaches, care must be made to hand-design a set of data augmentations that don't degrade the semantics of the input, otherwise the ability to learn a feature space that captures input semantics while being invariant to data augmentation will be lost. In 3D, it is somewhat less clear what constitutes a semantics-preserving data augmentation given that a 3D point cloud is defined solely by the locations of its constituent points, and so any disturbance of the original geometry (*e.g*. via cropping, scaling, or view-based occlusions) could be construed to potentially degrade its semantics. While contrastive approaches therefore need a carefully hand-designed set of transformations, our proposed method needs no transformation set or data augmentation procedure at all in order to learn useful representations.

Transformation/Context Prediction in 2D: Another class of methods devise pretext tasks to directly predict the contextual augmentation of transformation. In 2D images, such approaches include Jigsaw puzzles [33], estimating rotations [14], context prediction by relative paired patch orientation prediction [12], and predicting transformations directly in a VAE-like context [38, 26].

Transformation/Context Prediction in 3D: Sauder and Sievers [41] were the first to adapt the jigsaw puzzle pretext task for learning representations on 3D point clouds. These approaches also interpret point-wise classifications as the predictions of point-to-partition correspondence and enjoy a similar benefit to our proposed approach in that any point-wise classification network can be used as the backbone. However, unlike the jigsaw puzzle task, which uses static and regular hard spatial partitions that are set *a priori*, our pretext task *learns* the placement of these spatial partitions via self-training. Furthermore, the partitions in our task can be arbitrarily shaped and are defined with soft probabilistic boundaries. Lastly, we have no need for data augmentation (*e.g*. by permuting space) since we these partition predictions are used to implicitly define a latent generative model and therefore spatial density from which data likelihood can be directly calculated. Other notable methods in 3D include learning shared features between part-whole hierarchies [39], a point cloud completion task [45], and orientation/rotation estimation [35]. We compare our method against these latter two techniques in Section 4.

3D Representation Learning by Learning Generative Models: Our pretext task can be considered in the context of learning a generative model for the given input data. Previous work on learning generative models for point cloud data have also observed its potential for use as a general purpose representation learning paradigm. Yuan *et al*. learn generative models of latent isotropic GMMs for the purpose of global point cloud registration [52]. Achlioptas *et al*. design an autoencoding architecture for point clouds and then train GANs [16] or mixture models to represent the latent space distribution [1]. PointGrow [43] is an autoregressive model for point cloud generation that allows inter-point correlations to be learned and modeled. Pointflow [50] creates a two-level hierarchy of Continuous Normalizing Flows [40, 17], where the first level encodes the high-dimensional shape space and the second level, condi-

tioned on the first, encodes the flow mapping a 3D Gaussian into a uniform distribution of points over the surface manifold of a 3D object. We compare against all these representations in Section 4.3.

3. Pretext Task

We derive the proposed pretext task from a simple insight: *A mathematical equivalence between a common component of both supervised and unsupervised learning exists that can be exploited to convert the former to the latter.* This common component is the *softmax* operation. In a typical supervised application, when class labels are known, row-wise softmax is how classification probabilities are calculated. In an unsupervised learning, this *same operation* is performed when Bayes Rule is applied to calculate the posterior of an underlying discrete latent variable of a generative model describing the joint data likelihood.

We exploit this mathematical equivalence to turn a supervised point-wise classification network (*e.g.* PointNet segmentation network) into an unsupervised maximum likelihood network that optimizes the parameters of a discrete latent variable model (GMM). This softmax equivalence enables our proposed unsupervised pre-training method.

3.1. Overview

To introduce our pretext task, we start with a neural network f_ψ with parameters ψ that takes as input a point cloud $\mathcal{P}$ consisting of a set of N 3D points $\boldsymbol{p}_i = \{x_i, y_i, z_i\}$ and that can output N different point-wise classification score vectors $\boldsymbol{s}_i$ for each $\boldsymbol{p}_i$ over a set of J possible classes. Thus, each score (logit) vector $\boldsymbol{s}_i = \{s_{ij}\}_{j=1}^{J}$ is of size J with index j indicating the class. The total logit predictions can be summarized by the score matrix $\mathcal{S}$ that has size $N \times J$. This output is common to many different architectures and is the exact output of something like a semantic segmentation network in which each point in a point cloud is assigned logits to one of J possible semantic categories (*e.g.*, ground, building, car, road, etc).

3.2. Probabilistic Spatial Partitioning

The first step is to reinterpret $\mathcal{S}$, the $N \times J$ matrix of logits that form the output of a point cloud segmentation network, as a set of N joint log probabilities between the points $\boldsymbol{p}_i$ and a set of J latent binary correspondence variables $\mathbf{c}_i = \{c_{ij}\}_{j=1}^{J}$ where exactly one entry in each latent correspondence vector $\mathbf{c}_i$ is 1 and the rest are 0. One can think of this binary correspondence vector as assigning each point in $\mathcal{P}$ to one of J discrete spatial partitions. Mathematically we can write,

$$s_{ij} \stackrel{\text{def}}{=} \log p(\boldsymbol{p}_i, c_{ij} = 1) \tag{1}$$

From this joint log likelihood interpretation, we can then calculate the posterior over each binary correspondence c_{ij}. Using Bayes Rule to calculate this posterior reduces to a row-wise *softmax* operation over $\mathcal{S}$,

$$p(c_{ij} = 1|\boldsymbol{p}_i) = \frac{p(\boldsymbol{p}_i|c_{ij} = 1)p(c_{ij} = 1)}{p(\boldsymbol{p}_i)} \tag{2}$$

$$= \frac{\exp\left(\log p(\boldsymbol{p}_i, c_{ij} = 1)\right)}{\sum_{j'=1}^{J} \exp\left(\log p(\boldsymbol{p}_i, c_{ij'} = 1)\right)} \tag{3}$$

$$= \frac{\exp\left(s_{ij}\right)}{\sum_{j'=1}^{J} \exp\left(s_{ij'}\right)} \tag{4}$$

Thus, computing a row-wise *softmax* of the $N \times J$ matrix $\mathcal{S}$, as is commonly done to predict class probabilities in a supervised paradigm, is exactly equivalent to computing the posterior probabilities for a set of latent correspondence variables that associate points to one of J spatial partitions. As shorthand, we define $\gamma_{ij} \stackrel{\text{def}}{=} p(c_{ij} = 1|\boldsymbol{p}_i)$ and the $N \times J$ matrix of all γ_{ij} as $\Gamma \stackrel{\text{def}}{=} \{\gamma_{ij}\}_{i,j}^{N,J}$. Note that if we had direct supervision of the values for c_{ij} (*e.g.* through data augmentation or generation of synthetic data), optimizing over the log posterior would generalize to contrastive learning.

This softmax equivalence sets up the connection from point-wise segmentation networks to probabilistic spatial partitioning. Suppose we have a generative likelihood model $g : \mathbb{R}^3 \to \mathbb{R}$ characterized by a finite set of parameters $\boldsymbol{\Theta}$. These $\boldsymbol{\Theta}$ parametrically define a spatial probability distribution over the entire 3D Euclidean domain $\mathbb{R}^3$ such that $\iiint g(x, y, z; \boldsymbol{\Theta})\, dx\, dy\, dz = 1$. Given a point cloud $\mathcal{P} = \{\boldsymbol{p}_i\}_{i=1}^{N}$, if we interpret each individual point as being an *iid* sample of g, this induces a factorized total probability $p(\mathcal{P})$ of the form $p(\mathcal{P}) = \prod_{i=1}^{N} g(\boldsymbol{p}_i; \boldsymbol{\Theta})$. Further, we choose g to be constructed as a convex combination of a discrete set of J probability distributions, each with their own individual set of parameters $\boldsymbol{\Theta}_j$. That is, we can define $\boldsymbol{\Theta} = \{\pi_j, \boldsymbol{\Theta}_j\}_{j=1}^{J}$ where $\{\pi_j\}_{j=1}^{J}$ are the convex weights of each sub-component, making g a mixture model with J sub-components.

3.3. Training as Latent Model Optimization

If we interpret the latent correspondence posterior derivation from Equations 2-4 as the probabilistic association of points to g's J sub-components, we arrive at the two different interpretations of $\mathcal{S}$ explained in Section 1:

1. $\mathcal{S}$ represents a soft assignment of each point in $\mathcal{P}$ to J discrete spatial partitions.

2. $\mathcal{S}$ represents a probabilistic association prediction of each point in $\mathcal{P}$ with g's J mixture components.

We tie these two interpretations together by letting the spatial partitions themselves determine the specific values of g's sub-component parameters $\boldsymbol{\Theta}_j$. If the form of each of g's J sub-components is a 3D Gaussian (*i.e.*, g defines a

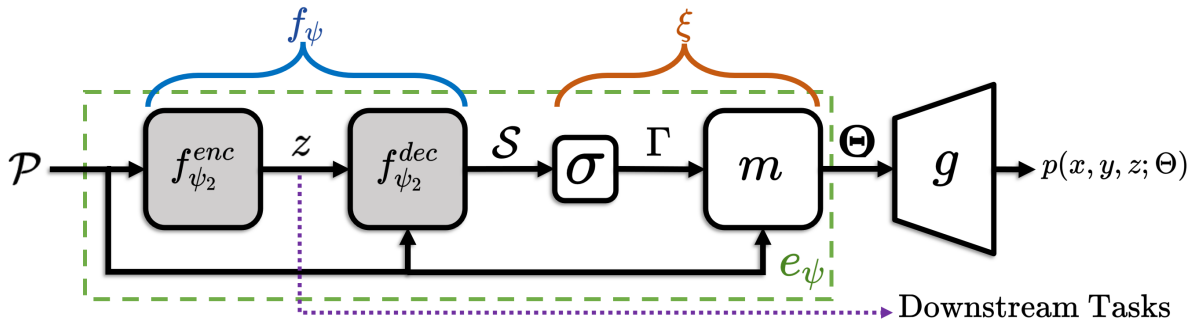


Figure 2. **Parametric Generative Model Bottleneck:** Using an off-the-shelf point-wise classification network, f_ψ, we can construct the parametric encoder denoted by e_ψ from the concatenation of f_ψ with ξ. The function ξ consists of a row-wise softmax that converts $\mathcal{S}$ into the set of posterior predictions Γ and a parameter-less fixed function compute block m that computes $\mathbf{\Theta}$ from $\mathcal{P}$ and Γ. The output $\mathbf{\Theta}$ is then used in conjunction with g to establish a dense PDF over 3D space. The learned representations z can be transferred to downstream tasks and everything after f^{enc}_ψ can be discarded.

GMM), then each $\mathbf{\Theta}_j$ is completely defined by its 3D mean $\boldsymbol{\mu}_j$ and 3×3 covariance matrix $\mathbf{\Sigma}_j$. Using the soft assignments of each point to J spatial partitions, we can calculate the mean and covariance of the points in each of these J partitions as follows,

$$\pi_j = \frac{1}{N} \sum_{i=1}^{N} \frac{\exp\left(s_{ij}\right)}{\sum_{j'=1}^{J} \exp\left(s_{ij'}\right)} \tag{5}$$

$$\boldsymbol{\mu}_j = \frac{1}{N\pi_j} \sum_{i=1}^{N} \frac{\exp\left(s_{ij}\right) \boldsymbol{p}_i}{\sum_{j'=1}^{J} \exp\left(s_{ij'}\right)} \tag{6}$$

$$\mathbf{\Sigma}_j = \frac{1}{N\pi_j} \sum_{i=1}^{N} \frac{\exp\left(s_{ij}\right) \left(\boldsymbol{p}_i - \boldsymbol{\mu}_j\right)\left(\boldsymbol{p}_i - \boldsymbol{\mu}_j\right)^\top}{\sum_{j'=1}^{J} \exp\left(s_{ij'}\right)} \tag{7}$$

Equation 5 calculates the relative proportion of points residing in each given partition, and Equations 6 and 7 calculated the softly weighted means and covariances of the points assigned to these spatial partitions. Note that only $\mathcal{S}$ and the original points $\mathcal{P}$ are required to calculate the parameters π_j, $\boldsymbol{\mu}_j$, and $\mathbf{\Sigma}_j$ for all $j = 1..J$. These calculated partition statistics directly form the parameter set of g as $\mathbf{\Theta} \leftarrow \{\pi_j, \boldsymbol{\mu}_j, \mathbf{\Sigma}_j\}_{j=1}^{J}$. For convenience, we define the deterministic, differentiable, and parameter-less function $\xi : \mathbb{R}^{N\times 3} \times \mathbb{R}^{N\times J} \rightarrow \mathbb{R}_+^{J\times 1} \times \mathbb{R}^{J\times 3} \times S_{++}^{3,\ J\times 1}$, such that $\xi(\mathcal{P}, \mathcal{S}) = \mathbf{\Theta}$ and where S_{++}^3 denotes the set of symmetric positive definite matrices of dimension equal to 3. Our encoder e_ψ is therefore completely defined by the parameters ψ of the point-wise classification network f_ψ through the composition $e_\psi \stackrel{\text{def}}{=} \xi(\mathcal{P}, f_\psi(\mathcal{P}))$.

The connection between $\mathcal{S}$ and g allows us to directly calculate the negative log likelihood of an input point cloud $\mathcal{P}$ under the assumption that g parameterized by $\mathbf{\Theta}$ defines a generative probability for all $\boldsymbol{p}_i \in \mathcal{P}$ as *iid* samples of g,

$$-\log p(\mathcal{P}; \mathbf{\Theta}) = -\sum_{i=1}^{N} \log g(\boldsymbol{p}_i; \mathbf{\Theta}) \tag{8}$$

$$= -\sum_{i=1}^{N} \log \sum_{j=1}^{J} p(\boldsymbol{p}_i, c_{ij} = 1; \mathbf{\Theta}) \tag{9}$$

$$= -\sum_{i=1}^{N} \log \sum_{j=1}^{J} \pi_j \mathcal{N}(\boldsymbol{p}_i; \boldsymbol{\mu}_j, \mathbf{\Sigma}_j) \tag{10}$$

Finally, our proposed unsupervised pretext task can be summarized mathematically as follows,

$$\psi^* = \underset{\psi}{\operatorname{argmin}}\, \mathbb{E}_{\mathcal{P}\sim\mathcal{D}} \left[-\sum_{i=1}^{N} \log g(\boldsymbol{p}_i; e_\psi(\mathcal{P})) \right] \tag{11}$$

The proposed unsupervised pre-training task is shown diagrammatically in Figure 2. In summary, the task is to learn the parameters ψ of the point-wise classification network f_ψ that minimize the expected negative log likelihood under the interpretation that $\xi(\mathcal{P}, f_\psi(\mathcal{P}))$ defines an encoder network e_ψ that subsequently determines the specific form of a latent generative model g.

The training loss is formed by assuming that every $\boldsymbol{p}_i \in \mathcal{P}$ can be viewed as *iid* samples of the latent generative model g having the PDF $p(x, y, z; \mathbf{\Theta})$. Given an empirical data distribution $\mathcal{D}$ over point clouds $\mathcal{P}$, such that $\mathcal{P} \sim \mathcal{D}$, a Monte Carlo optimization of the objective in Equation 11 over ψ amortizes the loss over the entire dataset.

These equations calculate the optimal parameters of a latent GMM in a minimum KL-Divergence sense with respect to the backbone network f_ψ. For the theoretical justification behind this choice, refer to the Supplementary.

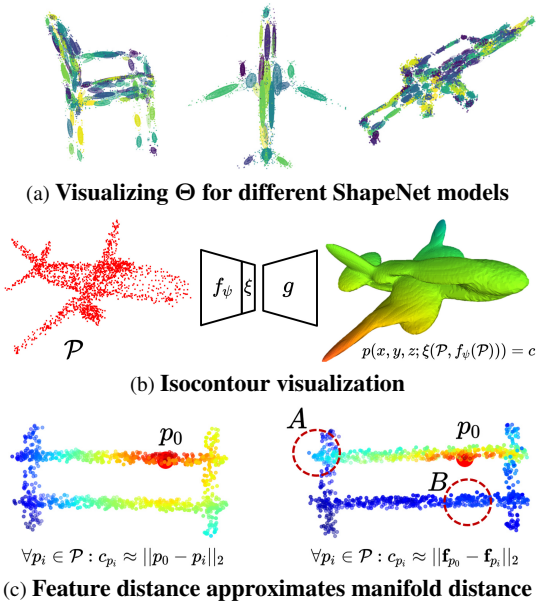
3.4. Autoencoding through a Parametric Bottleneck

One instructive way to view the proposed approach is through the lens of a traditional autoencoding task (See

Figure 1). A traditional autoencoder (Figure 1a) learns to compress its high dimensional input by pushing the input through an information bottleneck into a much smaller dimensional latent space (z), which is then decompressed from the bottleneck dimension back to its original dimension. After training, the learned autoencoder will compress its input in such a way as to be able to decompress it as faithfully as possible.

Our proposed method is similar to an autoencoder in the sense that we are also trying to encode our high dimensional input data into a smaller dimension that we can then use to reconstruct the original data as faithfully as possible. In this paradigm, our encoder e_ψ is the concatenation of f_ψ and ξ, and our decoder is g (Figure 1b). However, our method is different in the sense that the bottleneck layer has a standalone *parametric model* interpretation with respect to some underlying latent generative model defined by the decoding process g. Furthermore, in this view the decoder g is completely and deterministically defined by the specific values of the bottleneck layer (*i.e.*, $\mathbf{\Theta}$). This is in contrast to something like a Variational Autoencoder (VAE) [26], whose bottleneck layer also has a parametric interpretation with respect to an underlying generative model (diagonalized multivariate Gaussian), but for which this generative model is used only to seed a stochastic input into a learnable decoder.

Given that g is parameter-less and defines a total generative probability of $\mathcal{P}$ as the product of individual point probabilities, we can skip an explicit decoding or sampling step when calculating the loss and instead compute the exact likelihood of reconstructing the original N points if g were to be sampled N times. Backpropagating through a loss defined from this reconstruction likelihood will therefore produce better and better generative descriptions of the input data. That is, the segmentation network can only decrease its loss by inferring different spatial partitions using f_ψ such that the spatial density predicted by the model more closely fits the input data. For this reason, our proposed method can be seen as a kind of generalization to the jigsaw puzzle task since the network must learn something holistic about the global shape of its input in order to adequately partition it into geometrically compact and well-fitted partitions. See Figures 3 and 4 for different visualizations of $\mathbf{\Theta}$ and learned feature spaces using f_ψ trained from ShapeNet (Section 4.2).



(a) **Visualizing $\mathbf{\Theta}$ for different ShapeNet models**

(b) **Isocontour visualization**

(c) **Feature distance approximates manifold distance**

Figure 3. **Self-Supervised Pre-Training on ShapeNet Models** **Top:** We visualize $g(\cdot; \mathbf{\Theta})$ by plotting ellipsoid meshes around each Gaussian component's 1-sigma isocontour. The pretext task learns to partition the input such that these implicitly defined Gaussian components well-model their local geometry. **Middle:** We visualize an isocontour $g(\cdot; \mathbf{\Theta}) = c$ for a constant c and render the resulting isosurface using Marching Cubes. **Bottom:** Points are colored according to the distance from a random point p_0. The left point cloud is colored according to the 3D distance of p_0 with every point $p_i \in \mathcal{P}$. Similarly, the right point cloud is colored according to distance from p_0 but with respect to the learned point-level features. Point-level feature distance differs from 3D distance and tends toward local manifold distance. In region B, the bottom shelf is very far away from p_0 in feature space since it is unlikely that these points will belong to the same partition. In region A, the feature distance remains small since this area is located on the same planar segment.

4. Experimental Results

4.1. Implementation Details

Point Segmentation Backbone: We directly compare our implementation against Sauder and Sievers' 3D jigsaw task [41] ("Jigsaw3D"), Poursaeed *et al.*'s orientation estimation task [35] ("Rotation3D"), and Wang *et al.*'s occlusion completion task [45] ("OcCo"). In order to provide fair apples-to-apples comparison with these methods, we implement our proposed task using the same two 3D backbone networks that both these works use: PointNet [36] and Dynamic Graph CNN (DGCNN) [46]. In all cases, and to match prior work, our latent dimension is set to 1024.

Latent Feature Extraction: When performing feature extraction for use in downstream tasks, we don't use $\mathbf{\Theta}$ directly as our learned latent feature. Instead, we extract the global feature $\boldsymbol{z}$ embedded inside the point-wise classification backbone itself (note $\boldsymbol{z}$ in Figure 2). In the case of PointNet, we extract the global feature vector directly after the point-wise max-pool operation. Similarly, for DGCNN, we extract the global feature after the pooling layer after the fifth EdgeConv layer. One consequence of this is that our choice of the number of partitions J has no direct effect

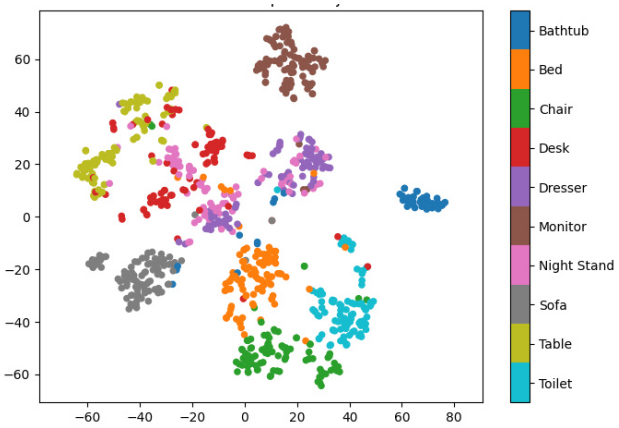


Figure 4. **Unsupervised features learned from ShapeNet:** t-SNE [29] visualization (perplexity of 10, 1000 iterations) of the 1024-dimensional feature vectors from the ModelNet10 test split, color coded by class label. Even though the model was trained on ShapeNet and never had access to any label information, the pretext task produces a feature space with strong semantic and class separation.

on the downstream task. For example, though the S3DIS dataset has 13 semantic classes (see Section 4.4), we do not need to pre-train our network to infer $J = 13$ spatial partitions.

Choice of Hyperparameter J: The most notable hyperparameter in our method is the choice of J, which sets the number of spatial partitions to infer during pre-training. Empirically, we found that increasing J yields better performance (see ablation results in Supplementary). However, given that our method requires the calculation of the full $N \times J$ matrix $\mathcal{S}$, this limits our ability to choose a large J if N is also large. In our experiments we set $J = 32$, which seems to be a good compromise between representational richness and training efficiency.

4.2. Representation Learning on ShapeNet

We apply our pretext task on the ShapeNet dataset, which consists of 57448 meshed models from 55 different household categories. We pre-train for 25 epochs, using point clouds consisting of $N = 1024$ uniformly random point samples from the original meshes (we found better results could be had with $N = 2048$, but to remain consistent with the methodology in [41] and elsewhere, we show results for $N = 1024$ only).

In addition to the common dimensionality reduction techniques for visualizing the feature space (*e.g.* see Figure 4), we can also readily interpret the parametric bottleneck itself, since each $\boldsymbol{\Theta}$ output predicts a unique PDF over 3D space. Furthermore, since $\boldsymbol{\Theta}$ is itself comprised of J mixture components $\boldsymbol{\Theta} = \{\pi_j, \boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j\}_{j=1}^{J}$, we can visualize each Gaussian component individually to see the interaction between $\mathcal{S}$, the partitions defined by it, and the parameterization it implies. In Figure 3a, we plot a mesh of each individual Gaussian's 1-sigma isocontour. One can see that the network learns to partition points in such a way as to keep each Gaussian as compactly supported as possible by its corresponding softly assigned point partition.

Learned Features + Linear SVM	Accuracy%
SPH [24]	68.2
LFD [5]	75.5
T-L Network [15]	74.4
VConv-DAE [42]	75.5
3D-GAN [47]	83.3
Latent-GAN [1]	85.7
PointGrow [43]	85.7
MRTNet-VAE [13]	86.4
PointFlow [50]	86.8
FoldingNet [51]	88.4
VIP-GAN [19]	90.2
PointNet + Jigsaw3D [41]	87.3
PointNet + Rotation3D [35]	88.6
PointNet + OcCo [45]	88.7
PointNet + ParAE (ours)	**90.3**
DGCNN + Jigsaw3D [41]	90.6
DGCNN + Rotation3D [35]	90.8
DGCNN + OcCo [45]	89.2
DGCNN + ParAE (ours)	**91.6**

Table 1. **ModelNet40 Classification using Linear SVM:** All models listed pre-train on ShapeNet in a self-supervised fashion, then fit a linear SVM to the features from the training split of ModelNet40 and report classification accuracy on the test split. Our method is denoted "ParAE" (Parametric Autoencoder), and we show performance both with a PointNet backbone and a DGCNN backbone.

4.3. ModelNet40 Classification

Linear SVM Performance: To test the efficacy of our learned representations, we apply the standard procedure of fitting a simple linear SVM classifier to the features from our unsupervised model [41, 1, 47, 51]. First, we pre-train on ShapeNet [4] as described in Section 4.2. Then, we freeze the network weights and extract global features from each point cloud in the training split of ModelNet40 [48]. ModelNet40 consists of 12311 meshed models from 40 object categories, split into 9843 training meshes and 2468 testing meshes. We train the linear SVM on the features from ModelNet40's training split and then report test accuracy on the test split.

Our linear SVM classification accuracy is compared in Table 1. We compare against a set of methods consisting of previous hand-crafted or generative modeling approaches, as well as PointNet and DGCNN models with various pretext tasks. Our basic PointNet backbone yields higher classification accuracy (90.3) than even a modern GAN approach [19] (90.2), and also yields better accuracy than

Model	Accuracy
PointNet + Random Initialization [36]	89.2
PointNet + Jigsaw3D [41]	89.8†
PointNet + OcCo [45]	90.2
PointNet + ParAE (ours)	**90.5**
DGCNN + Random Initialization [46]	92.2
DGCNN + Jigsaw3D [41]	92.4
DGCNN + OcCo [45]	**93.1**
DGCNN + ParAE (ours)	92.9

† as reported in [45]

Table 2. **Fully Supervised ModelNet40 Classification:** We compare random weight initialization against initializing models with pre-trained weights from various self-supervision tasks. Our pretext task outperforms the others using a PointNet backbone, and falls slightly behind the Occlusion Completion pretext task when using a DGCNN backbone.

Supervised	JigSaw3D [41]		**ParAE (ours)**	
Train Area	ΔmIOU	ΔAcc	ΔmIOU	ΔAcc
Area 1	+1.1	+0.6	**+9.9**	**+8.9**
Area 2	+0.3	0	**+3.9**	**+1.1**
Area 3	+2.5	+1.2	**+8.5**	**+6.7**
Area 4	+0.5	+0.1	**+5.6**	**+5.4**
Area 6	0	+0.2	**+5.3**	**+3.3**

Table 3. **Semantic Segmentation with S3DIS:** We look at the delta performance improvement when comparing supervised training performance under random weight initialization vs. pre-trained weights learned from an unsupervised pre-training task. Compared to pre-training with JigSaw3D [41], our proposed task provides a significantly larger boost to mIOU and accuracy. In all cases, the results are reported by supervised training on a single Area and calculating test performance on Area 5.

other proposed pretext tasks for learning on 3D point clouds (the next best is Occupancy Completion [45] at 88.7). Interestingly, we find that our linear SVM performance *even exceeds the performance of a fully supervised PointNet itself*, which trains to 89.2 test accuracy from random initialization [36]. With DGCNN, we see additional performance gains given that it is a more substantial and modern backbone with hierarchical graph-based features. Our DGCNN-based model obtains 91.6 overall test accuracy, while the next best is the Rotation3D, the pretext task based on orientation estimation (90.8).

Semi-Supervised Performance: We additionally test semi-supervised performance by limiting the amount of labeled ModelNet data. With only 50% of the training data labels, our method outperforms Jigsaw3D even if trained on 100% of the data (90.9 vs. 90.6). Surprisingly, with only 20% of labels our method outperforms a fully supervised FoldingNet [51] (88.6 vs. 88.4). For more details on these experiments, refer to the Supplementary.

ModelNet Fine-Tuning: In this experiment, we pre-train using our unsupervised pretext task on ShapeNet and then fine tune using ModelNet labels. Refer to Table 2. We can see that initializing the weights of the model using the weights learned from the pretext task unilaterally improves performance, even in the case of full supervision. Our classification accuracy in PointNet is state-of-the-art (90.5), however our performance with DGCNN (92.9) falls slightly below that of Occupancy Completion (93.1).

4.4. Semantic Segmentation

In this section, we show our method's downstream performance for semantic segmentation on the Stanford Large-Scale 3D Indoor Spaces (S3DIS) dataset [2]. This dataset consists of 3D scans from 6 different indoor spaces, totalling 271 rooms. Each point is labeled in terms of 13 different semantic categories. Our self-supervision experiment uses DGCNN and mirrors the set-up of [41]:

Baseline Models: We train 5 different supervised models from scratch on Areas 1-4 and 6. Then we test each of the 5 models on Area 5.

Pre-Trained Model: We pre-train a single model on all Areas 1-4 and 6 using the self-supervised pretext task. Then we use this single pre-trained model for weight initialization before fine tuning 5 different models according to the baseline scenario.

Our results are summarized in Table 3. For clarity of presentation, we omit absolute numbers and show the delta in performance that a given pre-training task provides over training the network from randomly initialized weights. Both self-supervised pretext tasks produce overall improvement in mIOU and accuracy compared to the baseline of random initialization without pre-training. However, our proposed task produces much large relative improvement. For example, pre-training on Areas 1-4 and 6 using our unsupervised pretext task produces a network with nearly *10 more mIOU* than the baseline, compared to JigSaw3D's relative performance increase of +1.1 mIOU.

5. Conclusion

Even with the recent successes in representation and transfer learning for 2D images, it yet remains unclear how to apply these techniques in the 3D realm where the data is much less structured. While recent works have shown that straightforward 3D adaption of jigsaw puzzles [41] and rotation estimation [35] can be beneficial, we propose a new pretext task tailored specifically for 3D data. We do this by exploiting the computational connection between supervised point-wise classification and the unsupervised calculation of a latent posterior with respect to a discrete generative model. This allows us to adapt any 3D architecture for point-wise classification to implicitly learn a discrete generative model of point density. In general, we hope this inspires further research into novel pretext tasks designed specifically to the idiosyncrasies of 3D data.

References

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

[2] 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*, pages 1534–1543, 2016.

[3] Jens Behley, Martin Garbade, Andres Milioto, Jan Quenzel, Sven Behnke, Cyrill Stachniss, and Jurgen Gall. Semantickitti: A dataset for semantic scene understanding of lidar sequences. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 9297–9307, 2019.

[4] Angel X. Chang, Thomas Funkhouser, Leonidas Guibas, Pat Hanrahan, Qixing Huang, Zimo Li, Silvio Savarese, Manolis Savva, Shuran Song, Hao Su, Jianxiong Xiao, Li Yi, and Fisher Yu. ShapeNet: An Information-Rich 3D Model Repository. Technical Report arXiv:1512.03012 [cs.GR], Stanford University — Princeton University — Toyota Technological Institute at Chicago, 2015.

[5] Ding-Yun Chen, Xiao-Pei Tian, Yu-Te Shen, and Ming Ouhyoung. On visual similarity based 3d model retrieval. In *Computer graphics forum*, volume 22, pages 223–232. Wiley Online Library, 2003.

[6] Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. A simple framework for contrastive learning of visual representations. *arXiv preprint arXiv:2002.05709*, 2020.

[7] Ting Chen, Simon Kornblith, Kevin Swersky, Mohammad Norouzi, and Geoffrey Hinton. Big self-supervised models are strong semi-supervised learners. *arXiv preprint arXiv:2006.10029*, 2020.

[8] Christopher Choy, JunYoung Gwak, and Silvio Savarese. 4d spatio-temporal convnets: Minkowski convolutional neural networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3075–3084, 2019.

[9] Christopher Choy, Jaesik Park, and Vladlen Koltun. Fully convolutional geometric features. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 8958–8966, 2019.

[10] Angela Dai, Angel X. Chang, Manolis Savva, Maciej Halber, Thomas Funkhouser, and Matthias Nießner. ScanNet: Richly-annotated 3D reconstructions of indoor scenes. 2017.

[11] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

[12] C. Doersch, A. Gupta, and Alexei A. Efros. Unsupervised visual representation learning by context prediction. *2015 IEEE International Conference on Computer Vision (ICCV)*, pages 1422–1430, 2015.

[13] Matheus Gadelha, Rui Wang, and Subhransu Maji. Multiresolution tree networks for 3d point cloud processing. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 103–118, 2018.

[14] Spyros Gidaris, Praveer Singh, and Nikos Komodakis. Unsupervised representation learning by predicting image rotations. *arXiv preprint arXiv:1803.07728*, 2018.

[15] Rohit Girdhar, David F Fouhey, Mikel Rodriguez, and Abhinav Gupta. Learning a predictable and generative vector representation for objects. In *European Conference on Computer Vision*, pages 484–499. Springer, 2016.

[16] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In *Advances in neural information processing systems*, pages 2672–2680, 2014.

[17] Will Grathwohl, Ricky TQ Chen, Jesse Bettencourt, Ilya Sutskever, and David Duvenaud. Ffjord: Free-form continuous dynamics for scalable reversible generative models. *arXiv preprint arXiv:1810.01367*, 2018.

[18] Jean-Bastien Grill, Florian Strub, Florent Altché, C. Tallec, Pierre H. Richemond, Elena Buchatskaya, C. Doersch, Bernardo Avila Pires, Zhaohan Daniel Guo, Mohammad Gheshlaghi Azar, B. Piot, K. Kavukcuoglu, Rémi Munos, and Michal Valko. Bootstrap your own latent: A new approach to self-supervised learning. *ArXiv*, abs/2006.07733, 2020.

[19] Zhizhong Han, Mingyang Shang, Yu-Shen Liu, and Matthias Zwicker. View inter-prediction gan: Unsupervised representation learning for 3d shapes by learning global shape memories to support local view predictions. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 8376–8384, 2019.

[20] Kaiming He, Haoqi Fan, Yuxin Wu, Saining Xie, and Ross B. Girshick. Momentum contrast for unsupervised visual representation learning. *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 9726–9735, 2020.

[21] Kaiming He, X. Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 770–778, 2016.

[22] Olivier J. Hénaff, A. Srinivas, J. Fauw, Ali Razavi, C. Doersch, S. Eslami, and A. Oord. Data-efficient image recognition with contrastive predictive coding. *ArXiv*, abs/1905.09272, 2019.

[23] Armin Hornung, Kai M Wurm, Maren Bennewitz, Cyrill Stachniss, and Wolfram Burgard. Octomap: An efficient probabilistic 3d mapping framework based on octrees. *Autonomous robots*, 34(3):189–206, 2013.

[24] Michael Kazhdan, Thomas Funkhouser, and Szymon Rusinkiewicz. Rotation invariant spherical harmonic representation of 3 d shape descriptors. In *Symposium on geometry processing*, volume 6, pages 156–164, 2003.

[25] Leonid Keselman, John Iselin Woodfill, Anders Grunnet-Jepsen, and Achintya Bhowmik. Intel realsense stereoscopic depth cameras. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pages 1–10, 2017.

[26] Diederik P. Kingma and M. Welling. Auto-encoding variational bayes. *CoRR*, abs/1312.6114, 2014.

[27] Thomas N Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907*, 2016.

[28] Phuc H Le-Khac, Graham Healy, and Alan F Smeaton. Contrastive representation learning: A framework and review. *arXiv preprint arXiv:2010.05113*, 2020.

[29] L. V. D. Maaten and Geoffrey E. Hinton. Visualizing data using t-sne. *Journal of Machine Learning Research*, 9:2579–2605, 2008.

[30] Ben Mildenhall, Pratul P Srinivasan, Matthew Tancik, Jonathan T Barron, Ravi Ramamoorthi, and Ren Ng. Nerf: Representing scenes as neural radiance fields for view synthesis. *arXiv preprint arXiv:2003.08934*, 2020.

[31] Ishan Misra and Laurens van der Maaten. Self-supervised learning of pretext-invariant representations. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 6707–6717, 2020.

[32] Matthias Nießner, Michael Zollhöfer, Shahram Izadi, and Marc Stamminger. Real-time 3d reconstruction at scale using voxel hashing. *ACM Transactions on Graphics (ToG)*, 32(6):1–11, 2013.

[33] M. Noroozi and P. Favaro. Unsupervised learning of visual representations by solving jigsaw puzzles. In *ECCV*, 2016.

[34] A. Oord, Y. Li, and Oriol Vinyals. Representation learning with contrastive predictive coding. *ArXiv*, abs/1807.03748, 2018.

[35] Omid Poursaeed, Tianxing Jiang, Quintessa Qiao, Nayun Xu, and Vladimir G Kim. Self-supervised learning of point clouds via orientation estimation. *2020 International Conference on 3D Vision*, 2020.

[36] Charles R Qi, Hao Su, Kaichun Mo, and Leonidas J Guibas. Pointnet: Deep learning on point sets for 3d classification and segmentation. *Proc. Computer Vision and Pattern Recognition (CVPR), IEEE*, 1(2):4, 2017.

[37] Charles Ruizhongtai Qi, Li Yi, Hao Su, and Leonidas J Guibas. Pointnet++: Deep hierarchical feature learning on point sets in a metric space. In *Advances in neural information processing systems*, pages 5099–5108, 2017.

[38] Guo-Jun Qi, Liheng Zhang, Chang Wen Chen, and Qi Tian. Avt: Unsupervised learning of transformation equivariant representations by autoencoding variational transformations. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 8130–8139, 2019.

[39] Yongming Rao, Jiwen Lu, and Jie Zhou. Global-local bidirectional reasoning for unsupervised representation learning of 3d point clouds. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5376–5385, 2020.

[40] Danilo Jimenez Rezende and Shakir Mohamed. Variational inference with normalizing flows. *arXiv preprint arXiv:1505.05770*, 2015.

[41] Jonathan Sauder and Bjarne Sievers. Self-supervised deep learning on point clouds by reconstructing space. In *Advances in Neural Information Processing Systems*, pages 12962–12972, 2019.

[42] Abhishek Sharma, Oliver Grau, and Mario Fritz. Vconv-dae: Deep volumetric shape learning without object labels. In *European Conference on Computer Vision*, pages 236–250. Springer, 2016.

[43] Yongbin Sun, Yue Wang, Ziwei Liu, Joshua Siegel, and Sanjay Sarma. Pointgrow: Autoregressively learned point cloud generation with self-attention. In *The IEEE Winter Conference on Applications of Computer Vision*, pages 61–70, 2020.

[44] Haotian Tang, Zhijian Liu, Shengyu Zhao, Yujun Lin, Ji Lin, Hanrui Wang, and Song Han. Searching efficient 3d architectures with sparse point-voxel convolution. *arXiv preprint arXiv:2007.16100*, 2020.

[45] Hanchen Wang, Qi Liu, Xiangyu Yue, Joan Lasenby, and Matthew J Kusner. Pre-training by completing point clouds. *arXiv preprint arXiv:2010.01089*, 2020.

[46] 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):146, 2019.

[47] Jiajun Wu, Chengkai Zhang, Tianfan Xue, Bill Freeman, and Josh Tenenbaum. Learning a probabilistic latent space of object shapes via 3d generative-adversarial modeling. In *Advances in neural information processing systems*, pages 82–90, 2016.

[48] 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*, pages 1912–1920, 2015.

[49] Saining Xie, Jiatao Gu, Demi Guo, Charles R Qi, Leonidas J Guibas, and Or Litany. Pointcontrast: Unsupervised pre-training for 3d point cloud understanding. *arXiv preprint arXiv:2007.10985*, 2020.

[50] Guandao Yang, Xun Huang, Zekun Hao, Ming-Yu Liu, Serge Belongie, and Bharath Hariharan. Pointflow: 3d point cloud generation with continuous normalizing flows. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 4541–4550, 2019.

[51] 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*, pages 206–215, 2018.

[52] Wentao Yuan, Ben Eckart, Kihwan Kim, Varun Jampani, Dieter Fox, and Jan Kautz. Deepgmr: Learning latent gaussian mixture models for registration. *arXiv preprint arXiv:2008.09088*, 2020.

[53] Zhengyou Zhang. Microsoft kinect sensor and its effect. *IEEE multimedia*, 19(2):4–10, 2012.