Generative PointNet: Energy-Based Learning on Unordered Point Sets for 3D Generation, Reconstruction and Classification

Jianwen Xie[1], Yifei Xu[2], Zilong Zheng[2] Song-Chun Zhu[2], and Ying Nian Wu[2]

[1] Cognitive Computing Lab, Baidu Research, USA
jianwenxie@baidu.com
[2] University of California, Los Angeles, USA
{fei960922, zilongzheng0318}@ucla.edu, {sczhu, ywu}@stat.ucla.edu

Abstract. We propose a generative model of unordered point sets, such as point clouds, in the forms of an energy-based model, where the energy function is parameterized by an input-permutation-invariant bottom-up neural network. The energy function learns a coordinate encoding of each point and then aggregates all individual point features into energy for the whole point cloud. We show that our model can be derived from the discriminative PointNet. The model can be trained by MCMC-based maximum likelihood learning (as well as its variants), without the help of any assisting networks like those in GANs and VAEs. Unlike most point cloud generator that relys on hand-crafting distance metrics, our model does not rely on hand-crafting distance metric for point cloud generation, because it synthesizes point clouds by matching observed examples in terms of statistical property defined by the energy function. Furthermore, we can learn a short-run MCMC toward the energy-based model as a flow-like generator for point cloud reconstruction and interpretation. The learned point cloud representation can be also useful for point cloud classification. Experiments demonstrate the advantages of the proposed generative model of point clouds.

1 Introduction

1.1 Background and motivation

Point clouds, as a standard 3D acquisition format used by devices like Lidar on autonomous vehicles, Kinect for Xbox game console and face identification sensor on phones, are getting increasingly popular for 3D representation in computer vision. Moreover, compared to other 3D formats such as 3D voxel grid and 3D mesh, point cloud can provide a compact and detailed 3D representation because it represents a 3D object with densely placed points along its surface, rather than characterizing it as a regular 3D voxel grid with binary voxel values indicating opacity or constructing its surface through elementary shapes like triangles.

Learning a generative model of 3D point clouds is a fundamental problem for 3D computer vision because it is beneficial to 3D point cloud synthesis and analysis tasks, by providing an explicit probability distribution of point clouds. Despite the enormous advance of discriminative models for the tasks of 3D point cloud classification and segmentation, e.g., [23,24,42], the progress in developing generative models for 3D point clouds has been lagging behind. A major challenge in generative modeling of point clouds is that unlike images, videos and volumetric shapes, point clouds are not regular structures but unordered point sets, which makes extending existing paradigms intended for structured data not straightforward. That is why the majority of existing works on 3D generative models are based on volumetric data, e.g., [31,30,37,15].

With the recent success of a variety of generation tasks such as image generation [25,36,35,2,13,22] and video generation [29,38,35,27,32,33], researchers have become increasingly interested in point cloud generation, e.g., [7,43,28,1,20,40]. Most of them are based on frameworks of GAN [9] (e.g., [43,28,1,20]), VAE [18] (e.g., [7,40]), or encoder-decoder with hand-crafting distance metrics, such as Chamfer distance or earth mover's distance [6] for measuring two point clouds (e.g., [7,43]). In this paper, we propose a principled generative model for probabilistic modeling of 3D point clouds. Specifically, the model is a probability density function directly defined on unordered point sets, and it is in the form of a deep energy-based model with the energy function parameterized by an input-permutation-invariant bottom-up deep network that is suitable for defining energy on an unordered point set. We call the proposed model the generative PointNet, because it can be derived from the discriminative PointNet [23]. The maximum likelihood estimation of our model follows what Grenander [11] called "analysis by synthesis" scheme in pattern theory [10]. Specifically, within each learning iteration, "fake" 3D point cloud examples are generated by Langevin dynamics sampling, which is a gradient-based Markov chain Monte Carlo (MCMC) method, from the current model, and then the model parameters are updated based on the difference between the "fake" examples and the observed examples in order to match the "fake" examples to the observed examples in terms of some permutation-invariant statistical properties defined by the energy function.

Instead of implicitly modeling the distribution of points as a top-down generator model [9,18] (implicit because the marginal probability density of a generator model requires integrating out the latent noise vector, which is analytically intractable) or indirectly learning the model by an adversarial learning scheme where a discriminator is recruited and simultaneously trained with the generator in a minimax two-player game, or a variational inference scheme where an encoder is recruited as an inference model to approximate the intractable posterior distribution, we explicitly model this distribution as an energy-based model and directly learn the model by MCMC-based maximum likelihood (as well as its variants) without the aid of any extra network. The maximum likelihood learning, in general, does not suffer from mode collapse and instability issues existing in GANs due to the unbalanced joint training of two models.

Models using encoder-decoder structures for point cloud generation rely on hand-crafting distance metrics to measure the dissimilarity between two point sets. However, the maximum likelihood learning of our model corresponds to a statistical property matching of the observed and the generated point clouds, where the statistical property is defined by the derivative of the energy function with respect to the parameters. Therefore, our model doesn't rely on hand-crafting distance metrics.

About the learning algorithm, as mentioned above, the maximum likelihood learning algorithm follows an "analysis by synthesis" scheme, which iterates the following two steps. Synthesis step: generate the "fake" synthesized examples from the current model. Analysis step: update the model parameters based on the difference between the observed examples and the synthesized examples. See the recent paper [21] for a thorough investigation of various implementation schemes for learning the energy-based model. The following are two different implementations of the synthesis step. (1) Persistent chain [36], which runs a finite-step MCMC such as Langevin dynamics from the synthesized examples generated from the previous learning epoch. (2) Contrastive divergence chain [14], which runs a finite step MCMC from the observed examples. (3) Non-persistent short-run MCMC [22], which runs a finite step MCMC from Gaussian white noise. It is possible to learn an unbiased model using scheme (1), but the learning can be time-consuming. Scheme (2) learns a biased model that usually cannot generate realistic synthesized examples. (3) has been recently proposed by [22]. Even though the learned model may still be biased, the learning is very efficient, and the short-run MCMC initialized from noise can generate realistic synthesized examples. Moreover, the noise-initialized short-run Langevin dynamics may be viewed as a flow-like model [4,5,19] or a generator-like model [9,18] that transforms the initial noise to the synthesized example. Interestingly, the learned short-run dynamics is capable of reconstructing the observed examples and interpolating different examples, similar to the flow model and the generator model.

In our paper, we study energy-based generative modeling of unordered point sets by proposing the generative PointNet model. We adopt the non-persistent short-run MCMC sampling scheme to learn the model for the sake of unifying point cloud generation, reconstruction and classification. Experiments show that the learned short-run MCMC can generate realistic point cloud patterns, and it can reconstruct observed point clouds and interpolate between point clouds. Moreover, even though it learns a biased model, the learned energy function and features are capable of classification.

1.2 Related work

Energy-based generative ConvNets [36] aims to learn an explicit probability distribution of data in the form of the energy-based model, in which the energy function is parametrized by modern convolutional neural networks and the MCMC sampling is based on Langevin dynamics. Compelling results on learning complex data distributions with the energy-based generative ConvNets [36] have been shown on images [36], videos [38,39,12] and 3D voxels [37]. Some alternative sampling strategies to make the training of the models more effective have been studied. For example, [8] proposes a multi-grid method for learning energy-based generative ConvNet models. It learns an ensemble of generative ConvNet models for different scales of images with finite-step Langevin sampling. Synthesized images from smaller scales are used to initialize the Langevin sampling at larger scales. [35,34] propose to train a generative ConvNet model with the help of an extra generator network as an approximate direct sampler. Recently, [22] propose to learn non-convergent, non-mixing, and non-persistent short-run MCMC, and treat this short-run MCMC as a learned generator model or a flow model for image generation and reconstruction. However, the models in the works mentioned above are only suitable for data with a regular structure. Learning such an energy-based model for 3D point cloud, which is an unordered point set, has not been investigated prior to this paper.

Deep learning methods have been successfully applied to point clouds for discriminative tasks including classification and segmentation, such as [23,24,42]. PointNet [23] is a pioneering discriminative deep net that directly processes point clouds for classification, by designing permutation invariant network architecture to deal with unordered point sets. As to generative models of point clouds, [7] uses variational auto-encoders (VAEs) and [43] uses adversarial auto-encoders with heuristic loss functions measuring the dissimilarity between two point sets, e.g., Chamfer distance (CD) or earth mover's distance (EMD), for the point cloud generation. Generative adversarial networks (GANs) for point clouds are explored in [20,1,28]. For example, [20] and [1] learn GANs on raw point cloud data. [20] also learns GANs on the latent space of an auto-encoder that is pre-trained with CD or EMD loss on raw data. [28] proposes to generate point clouds via a GAN with graph convolution that extracts localized information from point clouds. [40] studies point cloud generation using continuous normalizing flows trained with variational inference. Our paper learns energy-based generative models of point clouds via MCMC-based MLE. The proposed model, which we call generative PointNet, can be derived from the discriminative PointNet. Our model enables us to get around the complexities of training GANs or VAEs, or the troubles of crafting distance metrics for measuring similarity between two point sets.

1.3 Contributions

The key contributions of our work are as follows.

- **Modeling**: We propose a novel deep energy-based model to explicitly represent the probability distribution of an unordered point set, such as 3D point cloud, by adopting a permutation invariant bottom-up network as the energy function. This is the first generative model with explicit density function for point cloud data.
- **Learning**: Under the proposed energy-based models, we propose to use an unconventional short-run MCMC to learn our model and treat the MCMC process as a flow-based generator model, such that it can be used for point cloud reconstruction and generation simultaneously. Unusually energy-based model is unable to reconstruct data. This is the first EBM model that can perform point cloud reconstruction.

- **Uniqueness**: Compared with existing point cloud generative model, the proposed model has the following unique properties: (1) It doesn't rely on extra assiting network for training; (2) It can be derived from the discriminative PointNet; (3) It unifies synthesis and reconstruction in a single energy-based framework; (4) It unified explicit density (i.e., EBM) and implicit density (i.e., short run MCMC as a latent variable model) of point cloud in a single framework.
- **Performance**: Our framework obtains competitive performance with much less parameters when comparing with the state-of-art point cloud generative models, such as GAN-based and VAE-based approaches, in the tasks of reconstruction, synthesis, and classification.

2 Proposed model

2.1 Generalizing the exponential family model

Suppose we observe a set of 3D shapes $\{X_i, i = 1, ..., N\}$ from a particular category of object. Each shape is represented by a set of 3D points $X = \{x_k, k = 1, ..., M\}$, where each point x is a vector of its 3D coordinate plus optional extra information such as RGB color etc. In this paper, unless otherwise stated, the points we discuss only contains 3D coordinate information for simplicity.

We define an explicit probability distribution of shape, each shape itself being a 3D point cloud, by the following energy-based model

$$p_\theta(X) = \frac{1}{Z(\theta)} \exp\left[f_\theta(X)\right], \tag{1}$$

where the energy function $-f_\theta(X)$ with parameters θ defines the energy of the point cloud X, and the point cloud X with a low energy is assigned a high probability. $Z(\theta) = \int \exp(f_\theta(X))dX$ is the analytically intractable normalizing constant, which ensures the sum of all the probabilities in the distribution is equal to 1.

Since each point cloud input X is a set of unordered points, the negative energy function, $f_\theta(X)$, defined on a point set needs to be invariant to $M!$ permutations of the point set in point feeding order. We parameterize $f_\theta(X)$ by an input-permutation-invariant bottom-up deep network with parameters θ. Specifically, we design $f_\theta(X)$ by applying a symmetric function on non-linearly transformed points in the set, i.e., $f_\theta(\{x_1, ..., x_M\}) = g(\{h(x_1), .., h(x_M)\})$, where h is parameterized by a multi-layer perceptron network and g is a symmetric function, which is a composition of an average pooling function and a multi-layer perceptron network. The network architecture of the energy function is visualized in Figure 1. Please read the caption for the details of the network.

2.2 Maximum likelihood

Suppose we observe a set of 3D shapes $\mathcal{X} = \{X_i, i = 1, ..., N\}$ from a particular category of object. Let q_{data} be the distribution that generates the observed

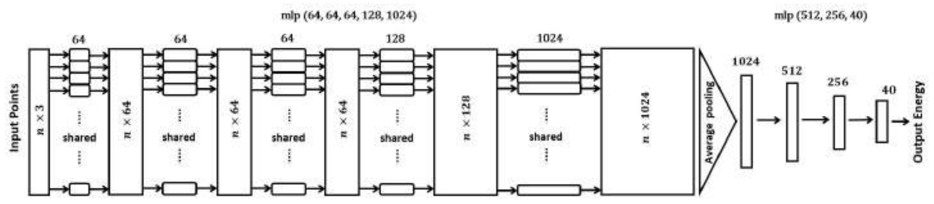

Fig. 1. Architecture of the energy function of the generative PointNet. The energy function is an input-permutation-invariant bottom-up deep network, which takes n unordered points as input, encodes each point into features by multilayer perceptron (MLP) with numbers of channels 64, 64, 64, 128, and 1024 at each layer respectively, and then aggregates all point features to a global feature by average pooling, and eventually outputs a scalar energy by multilayer perceptron with numbers of channels 512, 256 and 40 at each layer respectively. BatchNorm is used with ReLU for layers before average pooling, while only ReLU is used for layers after average pooling.

examples. The goal of learning the energy-based model p_θ is to estimate the parameter θ from the observations $\mathcal{X}$. For large N, the maximum likelihood estimation of θ,

$$\max_\theta \left[\frac{1}{N}\sum_{i=1}^{N}\log p_\theta(X_i)\right] \approx \max_\theta \mathrm{E}_{q_{\mathrm{data}}}[\log p_\theta(X)]$$

is equivalently to minimize the Kullback-Leibler divergence $\mathrm{KL}(q_{\mathrm{data}}\|p_\theta)$ over θ, where the KL divergence is defined as $\mathrm{KL}(q|p) = \mathrm{E}_q[\log(q(x)/p(x))]$. We can update θ by gradient ascent. The gradient of the log-likelihood or, equivalently, the negative KL divergence is computed by

$$\begin{aligned} &-\frac{\partial}{\partial\theta}\mathrm{KL}(q_{\mathrm{data}}(X)\|p_\theta(X)) \\ &= \mathrm{E}_{q_{\mathrm{data}}}\left[\frac{\partial}{\partial\theta}f_\theta(X)\right] - \mathrm{E}_{p_\theta}\left[\frac{\partial}{\partial\theta}f_\theta(X)\right] \quad (2) \\ &\approx \frac{1}{n}\sum_{i=1}^{n}\left[\frac{\partial}{\partial\theta}f_\theta(X_i)\right] - \frac{1}{n}\sum_{i=1}^{n}\left[\frac{\partial}{\partial\theta}f_\theta(\tilde{X}_i)\right], \quad (3) \end{aligned}$$

where $\{\tilde{X}_i, i = 1, ..., n\}$ are n point clouds generated from the current distribution p_θ by Markov chain Monte Carlo (MCMC) method, such as Langevin dynamics. Equation 3 refers to the MCMC approximation of the analytically intractable gradient due to the intractable expectation term $\mathrm{E}_{p_\theta}[\cdot]$ in Equation 3, and leads to the mini-batch "analysis by synthesis" learning algorithm. At iteration t, we randomly sample a batch of observed examples from the training data set $\{X_i, i = 1, ..., n\} \sim q_{\mathrm{data}}$, and generate a batch of synthesized examples from the current distribution $\{\tilde{X}_i, i = 1, ..., n\} \sim p_\theta$ by MCMC sampling. Then we compute the gradient $\Delta(\theta_t)$ according to Equation 3 and update the model parameter θ by $\theta_{t+1} = \theta_t + \gamma_t\Delta(\theta_t)$ with learning rate θ_t.

2.3 MCMC sampling with Langevin Dynamics

To sample point clouds from the distribution $p_\theta(X)$ by Langevin dynamics, we iterate the following steps:

$$X_{\tau+1} = X_\tau + \frac{\delta^2}{2}\frac{\partial}{\partial X} f_\theta(X_\tau) + \delta U_\tau, \tag{4}$$

where τ indexes the time step, δ is the step size, and $U_\tau \sim \mathrm{N}(0, I)$ is a Gaussian white noise. Since f_θ is a differentiable function, the term of gradient of $f_\theta(X_\tau)$ with respect to X can be efficiently computed via back-propagation. A momentum update and Metropolis-Hastings step can be added to correct for the finiteness of δ, but most authors find that these can be ignored in practice if δ is set to be small enough. As to MCMC initialization, we can initialize long-run non-persistent MCMC from noise point clouds, or initializes persistent MCMC from noise point clouds, and within each subsequent learning iteration, run a finite-step MCMC starting from the synthesized point cloud generated in the previous learning iteration. Following Contrastive Divergence [14], one may initialize the MCMC from the training examples sampled from the training data set within each learning iteration.

3 Short-run MCMC as generator model

Learning p_θ requires MCMC sampling to generate synthesized point clouds. The learned p_θ is multi-modal because p_{data} is usually multi-modal, which is due to the complexity of the point cloud patterns and the large scale of the data set. The property of multimodality is likely to cause the different chains to get trapped by the local modes. Thus the MCMC sampling of p_θ may take a long time to converge, regardless of the initial distribution and the length of the Markov chain. Following the recent work on learning EBM [22], instead of running a long-run convergent MCMC to sample from p_θ, we only run non-convergent, non-persistent short-run MCMC toward p_θ for a fixed number of steps K, starting from a fixed initial distribution, such as Gaussian white noise distribution p_0.

We use M_θ to denote the transition kernel of the K steps of MCMC toward $p_\theta(X)$. For a given initial probability distribution p_0, the resulting marginal distribution of the sample X after running K steps of MCMC starting from p_0 is denoted by

$$q_\theta(X) = M_\theta p_0(X) = \int p_0(Z) M_\theta(X|Z) dZ \tag{5}$$

Since $q_\theta(X)$ is not convergent, the X is highly dependent to Z. $q_\theta(X)$ can be considered a generator model, flow-based model, or a latent variable model with Z being the continuous latent variable in the following form

$$\begin{aligned} &Z \sim p_0(Z), \\ &X = M_\theta(Z, \xi), \end{aligned} \tag{6}$$

where Z and X have the same dimension, Z follows a known prior distribution p_0, i.e., Gaussian white noise distribution. M_θ is a short-run Langevin dynamics including K Langevin steps in Equation 4, which can be considered a K-layer residual network with noise injected into each layer and weight sharing in each layer. Let ξ be all the randomness in M_θ due to the layer-wise injected noise. Generally, parameter θ of generator model can be trained by adversarial learning (where an extra discriminator is recruited), variational inference (where an extra inference model is recruited), cooperative learning (where an extra energy-based model is recruited), or alternative back-propogation (where MCMC-based inference is required). However, as a special case, the generator model represented by a short-run MCMC shown in Equation 6 can be trained by the "analysis by synthesis" scheme, where we update θ according to Equation 3 and synthesize $\{\tilde{X}\}$ according to Equation 6. Training θ with short-run MCMC is no longer a maximum likelihood estimator but a moment matching estimator (MME) that solves equation

$$\mathrm{E}_{q_{\text{data}}}\left[\frac{\partial}{\partial\theta}f_\theta(X)\right] = \mathrm{E}_{p_\theta}\left[\frac{\partial}{\partial\theta}f_\theta(X)\right].$$

Even though the learned p_θ based on short-run MCMC is $p_{\hat{\theta}_{\text{MEE}}}$ rather than $p_{\bar{\theta}_{\text{MLE}}}$, the $q_{\bar{\theta}_{\text{MEE}}}$ is still a valid generator that is useful for 3D point cloud generator and reconstruction. As to reconstruction, given a testing 3D point cloud X, we can reconstruct X by finding Z to minimize the reconstruction error $L(Z) = ||X - M_\theta(Z)||^2$, where $M_\theta(Z)$ is a noise-disabled version of $M_\theta(Z, \xi)$ (after learning, the noise term is negligible compared to the gradient term). This can be easily achieved by running gradient descent on $L(Z)$, with Z initialized from $Z_0 \sim p_0$. Even though we abandon p_θ and keep q_θ eventually, p_θ is crucial because q is derived from p and we learn q under p. In other works, p serves as an incubator of $q_{\bar{\theta}_{\text{MEE}}}$.

When the model p_θ is learned from large scale data set and only limited budge of MCMC can be affordable, learning a short-run MCMC as a generator model toward p_θ for point cloud generation and construction will be a tradeoff between MCMC efficiency and MLE accuracy.

The learning method based on noise-initialized short-run MCMC is similar to contrastive divergence [14], which initializes a finite-step MCMC from each observed example within each learning iteration. Contrastive divergence also learns a bias model, but the learned model is usually incapable of synthesis, much less reconstruction and interpolation. For noise-initialized short-run Langevin, it is possible to optimize tuning parameters such as step sizes to minimize the bias caused by short-run MCMC. We leave this to future work.

4 Understanding generative PointNet

Proposition 1: *Generative PointNet corresponds to discriminative PointNet.* Suppose, besides noise category $p_0(X)$, there are K categories of 3D shapes, each of which is represented by the generative PoinNet defined in Equation 1, i.e., $p_{\theta_k}(X) = \frac{1}{Z}(\theta_k)\exp[f_{\theta_k}(X)], k = 1, ..., K$. Let ρ_k be the prior probability of category $k, k = 0, ..., K$. According to the Bayes' theorem, the posterior distribution that classifies an point cloud example X to the category $k \in \{0, ..., K\}$ is the discriminative PointNet, which is a softmax multi-class point cloud classifier,

$$p(k|X) = \frac{\exp(f_{\theta_k}(X) + b_k)}{\sum_{k=0}^{K} \exp(f_{\theta_k}(X) + b_k)}, \tag{7}$$

with $b_k = \log \rho_k - \log \rho_0 - \log Z(\theta_k)$.

Proposition 2: *Generative PointNet can be derived from discriminative PointNet.* Suppose a base category $k = 0$ is generated by $p_0(X)$, and suppose we fix $f_{\theta_0}(X) = 0$ and $b_0 = 0$ for the base category. If $p(k|X)$ is represented by discriminative PointNet in Equation 7, then according to the Bayes' theorem, the probability distribution of each category $p_\theta(X|k) = p_{\theta_k}(X)$ is of the form of model 1, with $b_k = \log \rho_k - \log \rho_0 + \log Z(\theta_k)$, where ρ_k is the prior probability of category k.

Proposition 3: Adversarial learning. Define the value function V with respect to θ and $\{\tilde{X}_i\}$ as

$$V(\theta, \{\tilde{X}_i\}) = \frac{1}{n} \sum_{i=1}^{n} \left[f_\theta(X_i) - f_\theta(\tilde{X}_i) \right]. \tag{8}$$

The learning step in Equation 3 and the sampling step in Equation 4 play a minimax game

$$\min_{\{\tilde{X}_i\}} \max_{\theta} V(\theta, \{\tilde{X}_i\}).$$

The learning step seeks to increase V by minimizing the KL divergence between data distribution and the model because $\frac{\partial}{\partial \theta} V = -\frac{\partial}{\partial \theta} \mathrm{KL}(q_{\mathrm{data}} \| p_\theta)$, while the sampling step seeks to decrease V by minimizing $-f_\theta(\tilde{X}_i)$ because the Langevin dynamics searches for the minima of the energy function.

Proposition 4: Moment matching estimation. Maximum likelihood learning of the generative PointNet with short-run or long-run MCMC solves the equation $-\frac{\partial}{\partial \theta} \mathrm{KL}(q_{\mathrm{data}}(X) \| p_\theta(X)) = 0$. At the convergence of the learning algorithm, we have

$$\frac{1}{n} \sum_{i=1}^{n} \left[\frac{\partial}{\partial \theta} f_\theta(X_i) \right] = \frac{1}{n} \sum_{i=1}^{n} \left[\frac{\partial}{\partial \theta} f_\theta(\tilde{X}_i) \right],$$

where the synthesized point clouds under the learned energy function match the observed point clouds on average in terms of the statistical features of the energy function, instead of raw point clouds.

Neural tangent kernel and metric: The learning algorithm seeks to match the expectations of $\Phi_\theta(X) = \frac{\partial}{\partial \theta} f_\theta(X)$ over the observed data and synthesized data. In the recent literature on the theoretical understanding of deep neural networks, the expectation of $\langle \Phi_\theta(X), \Phi_\theta(X') \rangle$, where the expectation is with respect to the random initialization of θ, is called the neural tangent kernel [16], and it plays a central role in understanding the optimization and generalization of deep and wide networks. It is possible to define a metric based on such a kernel. We shall study this issue in future work.

5 Experiments

5.1 Synthesis

We evaluate our model on generating 3D point clouds. We first create a point cloud dataset by sampling points uniformly from the mesh surface of all categories of objects in the ShapeNet dataset, and then normalize all points to have zero-mean and unit variance per axis. We train one single model for each category of point clouds. The number of training examples in each category ranges from 100 to 700. Each point cloud contains 1,024 points.

The network structure of the energy function is visualized in Figure 1. It first encodes each 3-dimensional point coordinate in Euclidean space to a 1024-dimensional point feature by multilayer perceptron (MLP), then uses an average pooling layer to aggregate information from all the points to a single 1024-dimensional global point cloud feature, and maps it to an energy by another multilayer perceptron. The energy function is permutationinvariant because the MLP for point encoding is shared by all unordered points and the output of the average pooling layer, followed by an MLP, serves as a symmetric function that is unaffected by the input order. All these components can be easily implemented by a bottom-up deep neural network, where the point encoder h is a 5-layer MLP with output dimensions 64, 64, 64, 128 and 1024 at each layer respectively and the symmetric function g is an average pooling followed by a 5-layer MLP with output dimensions 512, 256, 40 and 1. We use BatchNorm and ReLU operations in all layers in h, but only ReLU operation in all layers in g

To quantitatively evaluate the performance of generative models, we adopt the following three metrics that are also used in [1,40]:

(1) Jensen-Shannon Divergence (JSD) measures the distance between the marginal distributions of points in the reference set, $p_r(x)$, and the one in the generated set, $p_g(x)$, by

$$\mathrm{JSD}(p_g, p_r) = \frac{1}{2}[\mathrm{KL}(p_r||M) + \mathrm{KL}(p_g||M)],$$

where $M = \frac{1}{2}(p_g + p_r)$, and the empirical distributions of p_g and p_r can be obtained by voxelizing the point cloud space to a $28 \times 28 \times 28$ voxel grid and counting the number of points within each voxel across all point clouds in the set. This metric can only evaluate the marginal distribution of points rather than the distribution of point clouds. A model that fails to capture the diversity of shapes of point clouds in a category can still generate point clouds that can obtain a satisfactory JSD score.

(2) Coverage (COV) is defined as the fraction of point clouds in the reference set that are matched to at least one point cloud in the generated set, in which the matching of two point clouds is measured by either Chamfer distance (CD) or earth mover's distance (EMD) [6]. CD and EMD are two permutation-invariant metrics to measure the dissimilarity of two point clouds. Given two equally sized point clouds A and B, the CD metric is given by

$$D_{CD}(A, B) = \sum_{a \in A} \min_{b \in B} ||a - b||_2^2 + \sum_{b \in B} \min_{a \in A} ||a - b||_2^2,$$

while the EMD metric is computed by

$$D_{EMD}(A,B) = \min_{\phi: A \Rightarrow B} \sum_{a \in A} ||a - \phi(a)||_2,$$

where ϕ is a bijection. Let $\mathcal{X}_r$ and $\mathcal{X}_g$ be the reference set and generated set of point clouds respectively. The COV metric is computed by

$$\mathrm{COV}(\mathcal{X}_g, \mathcal{X}_r) = \frac{|\{\arg\min_{X_r \in \mathcal{X}_r} D(X_g, X_r) | X_g \in \mathcal{X}_g\}|}{|\mathcal{X}_r|},$$

(3) Minimum Matching Distance (MMD) measures the average of distances in the matching where every point cloud in the reference set is matched to the one in the generated set with the minimum distance. Either CD or EMD metrics can be used to compute the distance between two point clouds. The MMD metric is computed by

$$\mathrm{MMD}(\mathcal{X}_g, \mathcal{X}_r) = \frac{1}{|\mathcal{X}_r|} \sum_{X_r \in \mathcal{X}_r} \min_{X_g \in \mathcal{X}_g} D(X_g, X_r).$$

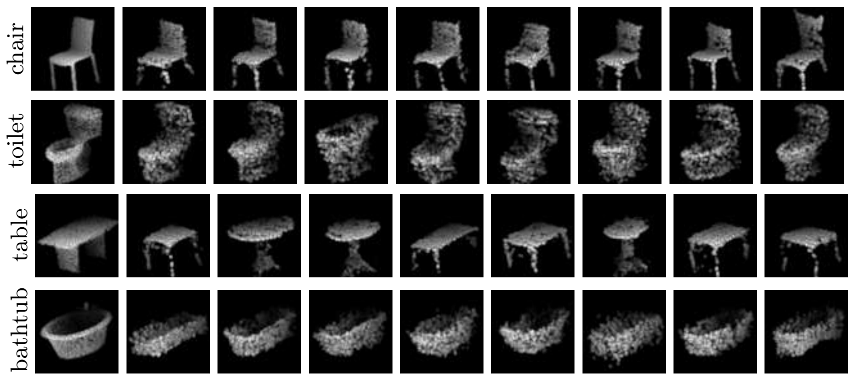

Fig. 2. Generating 3D point clouds. Each row shows one experiment, where the first point cloud is an example randomly selected from the training set. The rest are synthesized point clouds sampled from the learned model by Langevin dynamics. The number of points in each example is 1,024. From top to bottom: chair, toilet, table, and bathtub.

We compare our model with some baseline generative models for 3D point clouds, including PointFlow [40], l-GAN, and r-GAN, in Table 1 in terms of Jensen-Shannon Divergence (JSD), Coverage (COV), Minimum matching distance (MMD). We also compare the numbers of parameters of different models in Table 3. Due to the usage of extra networks in learning, models based on GANs and VAEs have different sizes of parameters in training and generation stage. Our model does not rely on auxiliary networks. We report the performance of the baseline methods using their official codes. Our method with fewer parameters outperforms all baselines across all four categories. Figure 2 displays some examples of point clouds generated by our model for categories chair, toilet, table, and bathtub.

Table 1. Comparison of synthesis quality

	Model	JSD	MMD		Coverage	
			CD	EMD	CD	EMD
Monitor	r-GAN	45.48	1.73	23.58	2.37	1.08
	l-GAN	10.27	1.28	14.12	9.89	3.01
	Flow	7.82	0.99	15.50	14.19	4.95
	Ours	**2.72**	**0.73**	**10.93**	**27.10**	**34.84**
	Truth	0.37	0.18	3.04	65.59	65.59
Dresser	r-GAN	25.58	2.08	23.42	6.00	4.50
	l-GAN	8.59	1.34	15.53	15.00	10.50
	Flow	4.45	0.93	12.18	29.00	34.50
	Ours	**3.33**	**0.87**	**11.68**	**37.00**	**35.50**
	Truth	1.07	0.24	3.30	64.00	64.00
Desk	r-GAN	43.42	1.90	35.28	4.50	1.50
	l-GAN	16.63	1.56	22.12	10.50	6.00
	Flow	7.10	**1.32**	19.18	17.00	23.00
	Ours	**4.51**	**1.32**	**15.27**	**27.50**	**29.00**
	Truth	1.17	0.40	4.75	62.50	62.50
Bathtub	r-GAN	49.09	1.36	30.37	5.66	2.83
	l-GAN	17.79	1.17	17.98	10.38	3.77
	Flow	9.98	1.05	15.29	17.93	14.15
	Ours	**4.99**	**0.86**	**11.87**	**32.08**	**29.25**
	Truth	1.40	0.26	3.55	63.21	63.21

	Model	JSD	MMD		Coverage	
			CD	EMD	CD	EMD
Table	r-GAN	59.69	2.75	34.17	1.79	0.51
	l-GAN	27.52	2.14	20.64	7.14	4.34
	Flow	7.73	1.15	15.82	15.82	12.27
	Ours	**3.02**	**0.99**	**12.63**	**31.89**	**36.22**
	Truth	0.59	0.23	3.67	61.99	61.99
Bed	r-GAN	35.05	1.12	29.46	5.63	1.55
	l-GAN	11.75	0.89	16.25	10.68	6.80
	Flow	4.85	0.74	12.71	18.64	17.48
	Ours	**2.10**	**0.78**	**11.34**	**25.63**	**30.87**
	Truth	0.36	0.21	3.46	61.94	61.94
Toilet	r-GAN	47.45	1.79	30.48	2.91	0.87
	l-GAN	15.50	1.55	19.82	11.34	3.78
	Flow	2.45	**0.87**	12.46	29.65	32.27
	Ours	**2.42**	1.05	**12.25**	**31.11**	**34.59**
	Truth	0.53	0.25	3.40	65.70	65.70
Chair	r-GAN	54.68	1.70	28.33	2.36	0.68
	l-GAN	15.57	1.31	18.93	4.16	1.12
	Flow	5.86	**0.88**	**12.55**	19.69	20.81
	Ours	**1.68**	0.91	12.68	**31.27**	**32.51**
	Truth	0.24	0.25	3.77	62.88	62.88

5.2 Reconstruction

We demonstrate the reconstruction ability of the proposed energy-based model for 3D point clouds. We learn our model with short-run MCMC as a generator. Given a testing point cloud object, we reconstruct it with the learned generator by minimizing the reconstruction error as we discussed in Section 3. Figure 3 displays some examples of reconstructing unobserved examples. The first row displays the original point clouds to reconstruct, the second row shows the corresponding reconstruction point clouds obtained by our model, and the third row shows the result obtained by PointFlow [40], which is a VAE-based framework as a baseline method. For VAE, the reconstruction can be easily achieved by first inferring the latent variable of the input example and then mapping the inferred latent variable to the point cloud space via the generator. Table 2 shows a quantitative comparison of the performance of different methods for point cloud

Table 2. A comparison of performances in point cloud reconstruction. (EMD: earth mover's distance. CD: Chamfer distance.)

Category	Model	EMD	CD
monitor	Ours	**10.71**	**0.77**
	PointFlow	16.55	1.44
dresser	Ours	**13.86**	**1.47**
	PointFlow	19.42	1.87
desk	Ours	**13.51**	**1.15**
	PointFlow	21.05	2.16
bathtub	Ours	**8.81**	**0.58**
	PointFlow	16.61	1.22
table	Ours	**17.76**	**2.60**
	PointFlow	18.75	1.75
bed	Ours	**11.05**	**0.83**
	PointFlow	18.06	1.27
toilet	Ours	**10.77**	**0.81**
	PointFlow	19.79	1.64
chair	Ours	**12.77**	**0.93**
	PointFlow	19.22	1.59

Table 3. Parameter Number (Million)

Method	Full	Generation
r-GAN	7.22	6.91
l-GAN	1.97	1.71
PointFlow	1.61	1.06
Ours	**0.82**	**0.82**

Table 4. A comparison of 3D object classification on ModelNet10 dataset

Method	Accuracy
SPH [17]	79.8%
LFD [3]	79.9%
VConv-DAE [26]	80.5%
3D-GAN [30]	91.0%
3D-WINN [15]	91.9%
3D-DescriptorNet [37]	92.4%
FoldingNet [41]	94.4%
l-GAN [1]	95.4%
PointFlow [40]	93.7%
Ours	92.4%

reconstruction. CD and EMD metrics are adopted to measure the quality of the reconstruction. Our method outperforms the baseline.

5.3 Interpolation

We demonstrate the interpolation ability of our model. We learn our model with short-run MCMC as a generator. We first sample two noise point clouds Z_1 and Z_2 from Gaussian distribution as two samples from the latent space. Then we perform linear interpolation in the latent space $Z_\rho = (1-\rho) \cdot Z_1 + \rho \cdot Z_2$, with ρ discretized into 8 values within $[0, 1]$. We generate the point clouds by $X_\rho = M_\theta(Z_\rho)$. Figure 4 shows two results of interpolation between Z_1 and Z_2 by displaying the sequence of generated point clouds $\{X_\rho\}$. Smooth transition and physically plausible intermediate generated examples suggest that the generator M_θ learns a smooth latent space for point cloud embedding.

5.4 Classification

The learned point encoder $h(x)$ in the energy function $f_\theta(X)$ can be useful for point cloud feature extraction, and the features can be applied to supervised learning. We evaluate h by performing a classification experiment on the ModelNet10 dataset. We first train a single generative PointNet on the training examples from all categories in an unsupervised manner. The network architecture and learning configuration are the same as the one used in the previous

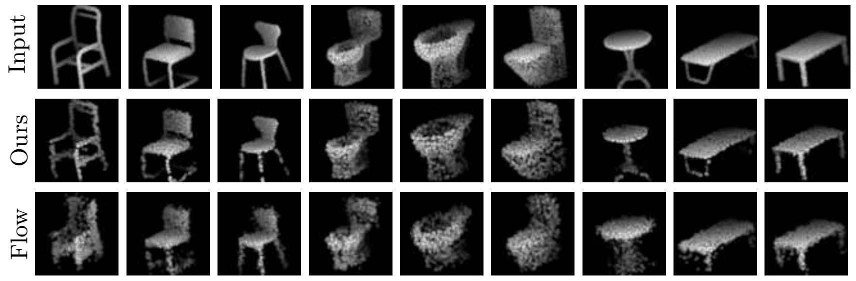

Fig. 3. Point cloud reconstruction. A short-run MCMC toward the energy-based generative PointNet as a generator is learned from categories chair, toilet, and table respectively. The learned generator is applied to point cloud reconstruction by inferring the latent noise point cloud Z to minimize the reconstruction error. The first row display some point clouds to reconstruct, the second row shows the corresponding reconstruction by our model, and the third row shows the corresponding reconstruction obtained by PointFlow, which is a VAE-based model and serves as a baseline.

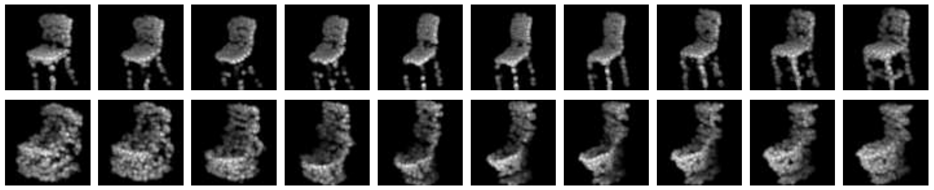

Fig. 4. Point cloud interpolation between generated examples at two ends. The transition in each row displays the sequence $M_\theta(Z_\rho)$ with linear interpolated latent variable $Z_\rho = \rho Z_1 + (1-\rho)Z_2$ where $\rho \in [0,1]$.

sections. We replace the average pooling layer by a max-pooling layer in the learned energy function and use the output of the max-pooling as point cloud features. Such as point cloud feature extractor is also permutation-invariant. We train an SVM classifier from labeled data based on the extracted features for classification. We evaluate the classification accuracy of the SVM on the testing data using the one-versus-all rule. Table 4 reports 7 published results on this dataset obtained by other baseline methods. Our method is on a par with other methods in terms of classification accuracy on this dataset.

We further conduct experiments to test the robustness of the classifier. We consider the following three types of data corruptions: (1) Type 1: missing points, where we randomly delete points from each point cloud. (2) Type 2: added points, where we add extra points that are uniformly distributed in the unit sphere into each point cloud. (3) point perturbation, where we perturb each point of each point cloud by adding a Gaussian noise. We report classification accuracy of the classifier on the corrupted version of ModelNet10 test set. Figure 5 shows the results. The classification performances decrease as the corruption levels (e.g., missing point ratio, added point ration, and standard deviation of point perturbation) increase. In the case of missing points, even though 80% points are deleted in each testing point cloud, the classifier can still perform with an accuracy rate 90.96%. In the extreme case where we only keep 10 points (10%) in each point cloud, the accuracy becomes 31.05%.

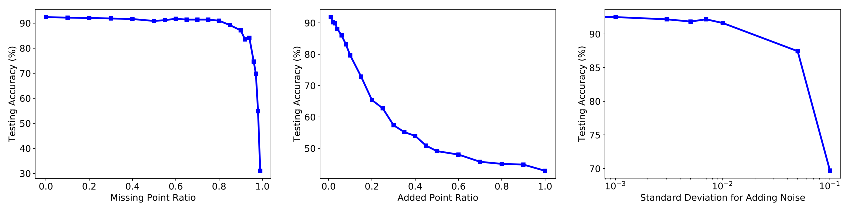

Fig. 5. Robustness Test. The model is tested on ModelNet10 test set with three types of point corruptions. Classification accuracies are reported across different levels of corruptions. Left: missing points. Middle: added points. Right: point perturbation.

5.5 Feature visualization

The learned energy function, which is in the form of an input-permutation-invariant bottom-up neural network, can serve as a feature extractor of point clouds for the sake of classification. To obtain more insights on the learned features of point clouds, we adopt t-SNE to visualize the 1024-dimensional point cloud feature extracted from our learned energy function in a 2D space. Figure 6 displays the extracted features for the point clouds in the ModelNet10 test split. As shown in Figure 6, the learned point cloud feature extractor performs reasonably well, because, with the learned features, similar shapes are clustered together.

5.6 Visualization of point encoding function

The energy function in our model learns a coordinate encoding of each point and then aggregates all individual point codes into energy for the point set. The coordinate encoding function is implemented by a multilayer perceptron (MLP), which learns to encode each 3-dimensional point via a 1024-dimensional vector. To better understand what the encoding function learned, we visualize each filter at different layers of the MLP by showing the points in the point cloud domain that give positive filter responses. In Figure 7, we randomly show 20 filter visualizations for each layer. The results suggest that different filters at different layers learn to detect points in different shapes of regions. Filters at a higher layer usually detect points in regions with more complicated shapes than those at a lower layer.

6 Conclusions

This paper studies the energy-based modeling of 3D point clouds. We propose a probability density of point clouds, which is unordered point sets, in the form of the energy-based model where the energy function is parameterized by a permutation-invariant function. The model can be trained via MCMC-based maximum likelihood learning, without the need of recruiting any other assisting network. The learning process follows "analysis by synthesis" scheme. Experiments show that the model can be useful for 3D generation, reconstruction, interpretation, and classification.

References

1. Achlioptas, P., Diamanti, O., Mitliagkas, I., Guibas, L.: Learning representations and generative models for 3D point clouds. In: International Conference on Machine Learning. pp. 40–49 (2018)
2. Brock, A., Donahue, J., Simonyan, K.: Large scale gan training for high fidelity natural image synthesis. arXiv preprint arXiv:1809.11096 (2018)
3. Chen, D.Y., Tian, X.P., Shen, Y.T., Ouhyoung, M.: On visual similarity based 3d model retrieval. In: Computer graphics forum. vol. 22, pp. 223–232. Wiley Online Library (2003)
4. Dinh, L., Krueger, D., Bengio, Y.: Nice: Non-linear independent components estimation. arXiv preprint arXiv:1410.8516 (2014)
5. Dinh, L., Sohl-Dickstein, J., Bengio, S.: Density estimation using real nvp. arXiv preprint arXiv:1605.08803 (2016)
6. Fan, H., Su, H., Guibas, L.J.: A point set generation network for 3d object reconstruction from a single image. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 605–613 (2017)
7. Gadelha, M., Wang, R., Maji, S.: Multiresolution tree networks for 3D point cloud processing. In: Proceedings of the European Conference on Computer Vision. pp. 103–118 (2018)
8. Gao, R., Lu, Y., Zhou, J., Zhu, S.C., Nian Wu, Y.: Learning generative ConvNets via multi-grid modeling and sampling. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 9155–9164 (2018)
9. Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., Bengio, Y.: Generative adversarial nets. In: Advances in Neural Information Processing Systems. pp. 2672–2680 (2014)
10. Grenander, U.: A unified approach to pattern analysis. In: Advances in Computers, vol. 10, pp. 175–216. Elsevier (1970)
11. Grenander, U., Miller, M.I.: Pattern theory: from representation to inference. Oxford University Press (2007)
12. Han, T., Nijkamp, E., Fang, X., Hill, M., Zhu, S.C., Wu, Y.N.: Divergence triangle for joint training of generator model, energy-based model, and inference model (2018)
13. Han, T., Nijkamp, E., Fang, X., Hill, M., Zhu, S.C., Wu, Y.N.: Divergence triangle for joint training of generator model, energy-based model, and inferential model. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 8670–8679 (2019)

14. Hinton, G.E.: Training products of experts by minimizing contrastive divergence. Neural computation **14**(8), 1771–1800 (2002)
15. Huang, W., Lai, B., Xu, W., Tu, Z.: 3d volumetric modeling with introspective neural networks (2019)
16. Jacot, A., Gabriel, F., Hongler, C.: Neural tangent kernel: Convergence and generalization in neural networks. In: Advances in neural information processing systems. pp. 8571–8580 (2018)
17. Kazhdan, M., Funkhouser, T., Rusinkiewicz, S.: Rotation invariant spherical harmonic representation of 3 d shape descriptors. In: Symposium on geometry processing. vol. 6, pp. 156–164 (2003)
18. Kingma, D.P., Welling, M.: Auto-encoding variational bayes. In: International Conference on Learning Representations (2014)
19. Kingma, D.P., Dhariwal, P.: Glow: Generative flow with invertible 1x1 convolutions. In: Advances in Neural Information Processing Systems. pp. 10215–10224 (2018)
20. Li, C.L., Zaheer, M., Zhang, Y., Poczos, B., Salakhutdinov, R.: Point cloud GAN. arXiv preprint arXiv:1810.05795 (2018)
21. Nijkamp, E., Hill, M., Han, T., Zhu, S.C., Wu, Y.N.: On the anatomy of mcmc-based maximum likelihood learning of energy-based models. arXiv preprint arXiv:1903.12370 (2019)
22. Nijkamp, E., Hill, M., Zhu, S.C., Wu, Y.N.: On learning non-convergent short-run MCMC toward energy-based model (2019)
23. Qi, C.R., Su, H., Mo, K., Guibas, L.J.: Pointnet: Deep learning on point sets for 3D classification and segmentation. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 652–660 (2017)
24. Qi, C.R., Yi, L., Su, H., Guibas, L.J.: Pointnet++: Deep hierarchical feature learning on point sets in a metric space. In: Advances in neural information processing systems. pp. 5099–5108 (2017)
25. Radford, A., Metz, L., Chintala, S.: Unsupervised representation learning with deep convolutional generative adversarial networks. arXiv preprint arXiv:1511.06434 (2015)
26. Sharma, A., Grau, O., Fritz, M.: Vconv-dae: Deep volumetric shape learning without object labels. In: European Conference on Computer Vision. pp. 236–250. Springer (2016)
27. Tulyakov, S., Liu, M.Y., Yang, X., Kautz, J.: Mocogan: Decomposing motion and content for video generation. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 1526–1535 (2018)
28. Valsesia, D., Fracastoro, G., Magli, E.: Learning localized generative models for 3D point clouds via graph convolution. In: Proceedings of International Conference on Learning Representations (2019)
29. Vondrick, C., Pirsiavash, H., Torralba, A.: Generating videos with scene dynamics. In: Advances in neural information processing systems. pp. 613–621 (2016)
30. Wu, J., Zhang, C., Xue, T., Freeman, B., Tenenbaum, J.: Learning a probabilistic latent space of object shapes via 3d generative-adversarial modeling. In: Advances in neural information processing systems. pp. 82–90 (2016)
31. Wu, Z., Song, S., Khosla, A., Yu, F., Zhang, L., Tang, X., Xiao, J.: 3d shapenets: A deep representation for volumetric shapes. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 1912–1920 (2015)
32. Xie, J., Gao, R., Zheng, Z., Zhu, S.C., Wu, Y.N.: Learning dynamic generator model by alternating back-propagation through time. In: Proceedings of the AAAI Conference on Artificial Intelligence. vol. 33, pp. 5498–5507 (2019)

33. Xie, J., Gao, R., Zheng, Z., Zhu, S.C., Wu, Y.N.: Motion-based generator model: Unsupervised disentanglement of appearance, trackable and intrackable motions in dynamic patterns. arXiv preprint arXiv:1911.11294 (2019)
34. Xie, J., Lu, Y., Gao, R., Wu, Y.N.: Cooperative learning of energy-based model and latent variable model via MCMC teaching. In: Thirty-Second AAAI Conference on Artificial Intelligence (2018)
35. Xie, J., Lu, Y., Gao, R., Zhu, S.C., Wu, Y.N.: Cooperative training of descriptor and generator networks. IEEE Transactions on Pattern Analysis and Machine Intelligence (2018)
36. Xie, J., Lu, Y., Zhu, S.C., Wu, Y.: A theory of generative ConvNet. In: International Conference on Machine Learning. pp. 2635–2644 (2016)
37. Xie, J., Zheng, Z., Gao, R., Wang, W., Zhu, S.C., Nian Wu, Y.: Learning descriptor networks for 3D shape synthesis and analysis. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 8629–8638 (2018)
38. Xie, J., Zhu, S.C., Nian Wu, Y.: Synthesizing dynamic patterns by spatial-temporal generative ConvNet. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 7093–7101 (2017)
39. Xie, J., Zhu, S.C., Nian Wu, Y.: Learning energy-based spatial-temporal generative ConvNets for dynamic patterns. IEEE Transactions on Pattern Analysis and Machine Intelligence (2019)
40. Yang, G., Huang, X., Hao, Z., Liu, M.Y., Belongie, S., Hariharan, B.: PointFlow: 3D point cloud generation with continuous normalizing flows. In: Proceedings of the IEEE International Conference on Computer Vision. pp. 4541–4550 (2019)
41. Yang, Y., Feng, C., Shen, Y., Tian, D.: Foldingnet: Point cloud auto-encoder via deep grid deformation. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 206–215 (2018)
42. Zaheer, M., Kottur, S., Ravanbakhsh, S., Poczos, B., Salakhutdinov, R.R., Smola, A.J.: Deep sets. In: Advances in neural information processing systems. pp. 3391–3401 (2017)
43. Zamorski, M., Zięba, M., Nowak, R., Stokowiec, W., Trzciński, T.: Adversarial autoencoders for generating 3D point clouds. arXiv preprint arXiv:1811.07605 (2018)

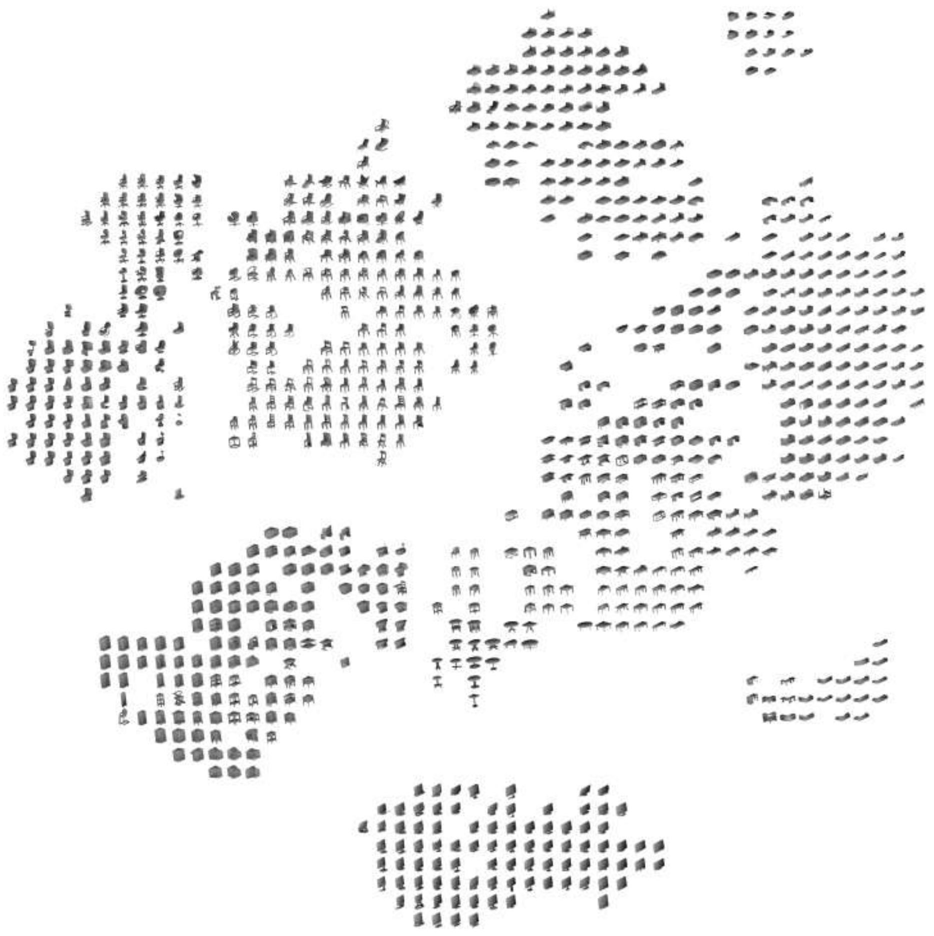

Fig. 6. t-SNE visualization of the learned features for the point clouds in ModelNet10 test split. Features are extracted by a bottom-up neural network that serves as the energy function in our model. In the context of maximum likelihood learning of the model, the network is trained in an unsupervised manner in the sense that no class labels are required during training.

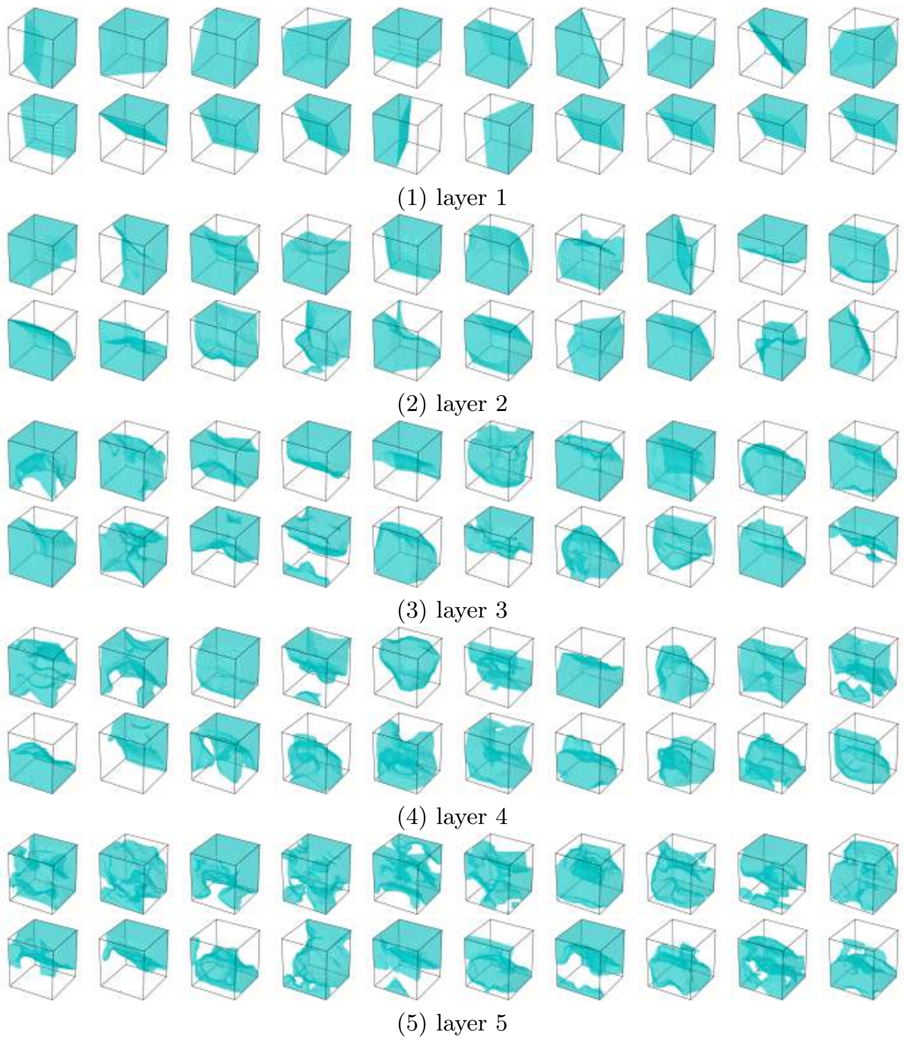

Fig. 7. Visualization of point encoding function. The point encoding function is implemented by a multilayer perceptron (MLP). We visualize each filter at different layers of the MLP by showing the points that have positive filter responses. We randomly display 20 filter visualizations for each layer.