# Generative PointNet: Deep 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 [2,3,4], Ying Nian Wu [2]
[1] Cognitive Computing Lab, Baidu Research, Bellevue, WA, USA
[2] University of California, Los Angeles (UCLA), CA, USA
[3] Tsinghua University, Beijing, China
[4] Peking University, Beijing, China
{jianwen, fei960922, z.zheng}@ucla.edu, {sczhu, ywu}@stat.ucla.edu

## Abstract

*We propose a generative model of unordered point sets, such as point clouds, in the form 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 an energy for the whole point cloud. We call our model the Generative PointNet because it can be derived from the discriminative PointNet. Our 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 generators that rely on hand-crafted distance metrics, our model does not require any hand-crafted distance metric for the point cloud generation, because it synthesizes point clouds by matching observed examples in terms of statistical properties 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 interpolation. The learned point cloud representation can be 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 and face identification sensor on phones, are getting increasingly popular for 3D representation in computer vision. Moreover, compared to other 3D formats such as voxel grids and 3D meshes, point clouds can provide a compact and detailed representation of a 3D object.

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., PointNet [31], PointNet++ [32], DeepSet [52], ShapeContextNet [49], PointGrid [24], DynamicGCN [37], and SampleNet [23], the progress in developing generative models for 3D point clouds has been lagging. 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., 3D ShapeNet [39], 3D GAN [38], Generative VoxelNet [44, 45], 3D-INN [16], etc.

With the recent success of a variety of generation tasks such as image generation and video generation, researchers have become increasingly interested in point cloud generation, e.g., [8, 53, 35, 1, 25, 50]. Most of them are based on well-established frameworks of GAN [11] (e.g., [53, 35, 1, 25]), VAE [22] (e.g., [8, 50]), or encoder-decoder with hand-crafted distance metrics, such as Chamfer distance or earth mover's distance [7] for measuring the dissimilarity of two point clouds (e.g., [8, 53]). 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 (EBM) [42] with the energy function parameterized by an input-permutation-invariant bottom-up deep network that is suitable for defining energy on an unordered point

*Equal contributions.

set. We call the proposed model the Generative PointNet because, following the theory presented in [42], such a model can be derived from the discriminative PointNet [31]. The maximum likelihood estimation (MLE) of our model follows what Grenander [13] called "analysis by synthesis" scheme in pattern theory [12]. 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 [26, 3] (MCMC) method, from the current model, and then the model parameters are updated based on the difference between the "fake" examples and the "real" observed examples in order to match the "fake" examples to the "real" 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 [11, 22] (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 used as an inference model to approximate the intractable posterior distribution, we explicitly model this distribution as an EBM and directly learn the model by MCMC-based MLE (as well as its variants) without the aid of any extra network. The MLE, in general, does not suffer from mode collapse and instability issues, which exist in GANs due to the unbalanced joint training of two models.

Models using encoder-decoders for point cloud generation typically rely on hand-crafted distance metrics to measure the dissimilarity between two point sets. However, the MLE learning of our model corresponds to a statistical matching between the observed and the generated point clouds, where the statistical properties are defined by the derivatives of the energy function with respect to the learning parameters. Therefore, our model does not rely on hand-crafted distance metrics.

About the learning algorithm, as mentioned above, the MLE 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 "real" observed examples and the "fake" synthesized examples. See the recent paper [29] for a thorough investigation of various implementation schemes for learning the EBM. The following are different implementations of the synthesis step. (i) Persistent chain [42], which runs a finite-step MCMC such as Langevin dynamics [27] from the synthesized examples generated from the previous learning iteration. (ii) Contrastive divergence chain [15], which runs a finite step MCMC from the observed examples. (iii) Non-persistent short-run MCMC [30], which runs a finite-step MCMC from Gaussian white noise. It is possible to learn an unbiased model using scheme (i), but the learning can be time-consuming. Scheme (ii) learns a biased model that usually cannot generate realistic synthesized examples. (iii) has been recently proposed by [30]. Even though the learned model may still be biased, similar to contrastive divergence, 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 [5, 6, 21] or a generator-like model [11, 22] 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 [30].

In our work, we adopt the learning scheme (iii). We 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 still useful for classification.

### 1.2. Related work

**Energy-based modeling and learning**. Energy-based generative ConvNets [42] aim to learn an explicit probability distribution of data in the form of the EBM, in which the energy function is parametrized by a modern convolutional neural network and the MCMC sampling is based on Langevin dynamics. Compelling results on learning complex data distributions with the energy-based generative ConvNets [42] have been shown on images [42], videos [47, 48, 14] and 3D voxels [44, 45]. Some alternative sampling strategies to make the training of the models more effective have been studied. For example, [9] proposes a multi-grid method for learning energy-based generative ConvNet models. Cooperative learning or CoopNets [41, 40, 43] trains a generative ConvNet with a generator as an amortized sampler via MCMC teaching. [30] proposes to learn a non-convergent, non-mixing, and non-persistent short-run MCMC, and treats this short-run MCMC as a learned generator model. Recent advances show that the generative ConvNet can be trained with a VAE, e.g., [14, 46] or a flow-based model, e.g., [10, 28]. However, the models in the works mentioned above are only suitable for data with regular structures. Learning EBMs for 3D point clouds, which are unordered point sets, has not been investigated prior to our paper.

**Deep learning for point clouds**. Deep learning methods have been successfully applied to point clouds for discriminative tasks including classification and segmentation, such as [31, 32, 52]. PointNet [31] is a pioneering dis-

criminative 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, [8] uses VAEs and [53] 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. GANs for point clouds are explored in [25, 1, 35]. For example, [25] and [1] learn a GAN on raw point cloud data, while [25] learns a GAN on the latent space of an auto-encoder that is pre-trained with CD or EMD loss on raw data. [35] proposes to generate point clouds via a GAN with graph convolution that extracts localized information from point clouds. [50] studies point cloud generation using continuous normalizing flows trained with variational inference. Our paper learns an EBM of point clouds via MCMC-based MLE. The proposed model, which we call *Generative PointNet* (or GPointNet), 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 EBM to explicitly represent the probability distribution of an unordered point set, e.g., a 3D point cloud, by designing a input-permutation-invariant bottom-up network as the energy function. This is the first generative model that provides an explicit density function for point cloud data. It will shed a new light not only on the area of 3D deep learning but also in the study of unordered set modeling.

**Learning**: Under the proposed EBMs, we propose to adopt an unconventional short-run MCMC to learn our model and treat the MCMC as a flow-based generator model, such that it can be used for point cloud reconstruction and generation simultaneously. Usually EBM is unable to reconstruct data. This is the first EBM that can perform point cloud reconstruction and interpolation.

**Uniqueness**: Compared with existing point cloud generative models, our model has the following unique properties: (1) It does not rely on an extra assisting network for training; (2) It can be derived from the discriminative PointNet; (3) It unifies synthesis and reconstruction in a single framework; (4) It unifies an explicit density (i.e., EBM) and an implicit density (i.e., short-run MCMC as a latent variable model) of the point cloud in a single framework.

**Performance**: Our energy-based framework obtains competitive performance with much fewer parameters compared with the state-of-art point cloud generative models, such as GAN-based and VAE-based approaches, in the tasks of synthesis, reconstruction and classification.

## 2. Generative PointNet

### 2.1. Energy-based model for unordered point sets

Suppose we observe a set of 3D shapes $\{X_i, i = 1, ..., N\}$ from a particular category of objects. 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, the points we discuss only contain 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

$$p_\theta(X) = \frac{1}{Z(\theta)} \exp\left[f_\theta(X)\right] p_0(X), \tag{1}$$

where $f_\theta(X)$ is a scoring function that maps the input $X$ to a score and is parameterized by a bottom-up neural network, $p_0(X) \propto \exp(-||X||^2/2s^2)$ is the Gaussian white noise reference distribution ($s$ is a hyperparameter and set to be 0.3 in our paper), $Z(\theta) = \int \exp[f_\theta(X)]p_0(X)dX$ is the analytically intractable normalizing constant, which ensures the sum of all the probabilities in the distribution is equal to 1. The energy function $\mathcal{E}_\theta(X) = -f_\theta(X) + ||X||^2/2s^2$ containing parameters $\theta$ defines the energy of the point cloud $X$, and the point cloud $X$ with a low energy is assigned a high probability.

Since each point cloud input $X$ is a set of unordered points, the energy function, $\mathcal{E}_\theta(X)$, defined on a point set needs to be invariant to $M!$ permutations of the point set in point feeding order. Because $||X||^2/2s^2$ is already naturally invariant to the point permutation, we only need to parameterize $f_\theta(X)$ by an input-permutation-invariant bottom-up deep network in order to obtain a proper $\mathcal{E}_\theta(X)$ that can handle unordered points. 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 an average pooling function followed by a multi-layer perceptron network. The network architecture of the scoring function $f_\theta$ is illustrated in Figure 1. Please read the caption for the details of the network.

### 2.2. Maximum likelihood

Suppose we observe a collection of 3D point clouds $\mathcal{X} = \{X_i, i = 1, ..., N\}$ from a particular category of object. Let $q_{\text{data}}$ be the distribution that generates the observed examples. The goal of learning $p_\theta$ is to estimate the parameter $\theta$ from the observations $\mathcal{X}$. For a large $N$, the maximum likelihood estimation of $\theta$,

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

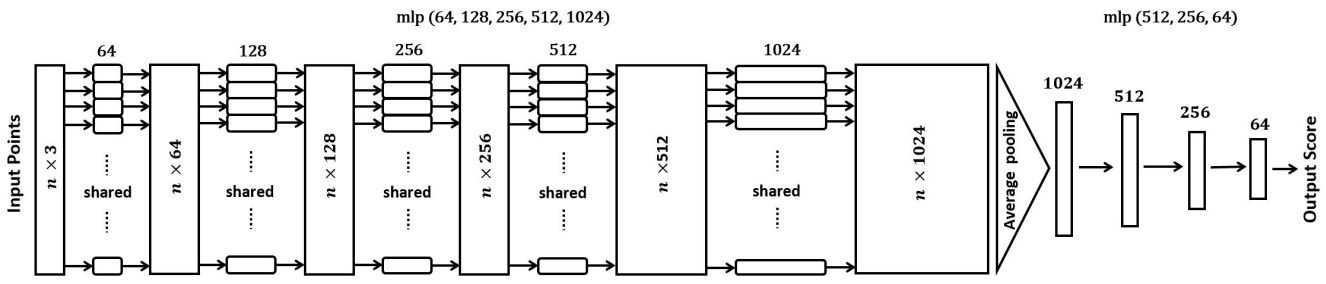


Figure 1: Architecture of the scoring function of the Generative PointNet. The scoring function $f_\theta(X)$ is an input-permutation-invariant bottom-up deep network, which takes $n$ unordered points as input, encodes each point into features by multilayer perceptron (MLP) with numbers of channels 64, 128, 256, 512 and 1,024 at each layer respectively, and then aggregates all point features to a global feature by average pooling, and eventually outputs scalar energy by multilayer perceptron with numbers of channels 512, 256, 64 and 1 at each layer respectively. Layer Normalization [2] is used with ReLU for layers before average pooling, while only ReLU is used for layers after average pooling.

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

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

where $\{\tilde{X}_i, i = 1, ..., n\}$ are $n$ point clouds generated from the current distribution $p_\theta$ by an MCMC method, such as Langevin dynamics. Eq.(3) refers to the MCMC approximation of the analytically intractable gradient due to the intractable expectation term $\mathrm{E}_{p_\theta}[\cdot]$ in Eq.(2), 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 Eq.(3) and update the model parameter $\theta$ by $\theta_{t+1} = \theta_t + \gamma_t\Delta(\theta_t)$ with a learning rate $\gamma_t$.

### 2.3. MCMC sampling with Langevin dynamics

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

$$X_{\tau+1} = X_\tau - \frac{\delta^2}{2}\frac{\partial}{\partial X}\mathcal{E}_\theta(X_\tau) + \delta U_\tau, \qquad (4)$$

where $\tau$ indexes the time step, $\delta$ is the step size, and $U_\tau \sim \mathcal{N}(0, I)$ is the Gaussian white noise. Since $f_\theta$ is a differentiable function, the term of gradient of $\mathcal{E}_\theta(X_\tau)$ with respect to $X$ can be efficiently computed via back-propagation. As to MCMC initialization, the following are three options. (1) Initialize long-run non-persistent MCMC from noise point clouds. (2) Initialize 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. (3) Following Contrastive Divergence [15], 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 $p_\theta$ requires MCMC sampling to generate synthesized point clouds. The learned $p_\theta$ 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 different MCMC chains to get trapped by the local modes. Thus the MCMC sampling of $p_\theta$ may take a long time to mix, regardless of the initial distribution and the length of the Markov chain. Following the recent work on learning EBM [30], instead of running a long-run convergent MCMC to sample from $p_\theta$, we only run non-convergent, non-persistent short-run MCMC toward $p_\theta$ for a fixed number of steps $K$, starting from a fixed initial distribution, such as Gaussian white noise distribution $p_0$.

We use $M_\theta$ 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

$$q_\theta(X) = M_\theta p_0(X) = \int p_0(Z)M_\theta(X|Z)dZ \qquad (5)$$

Since $q_\theta(X)$ is not convergent, the $X$ is highly dependent to $Z$. $q_\theta(X)$ can be considered a generator model, a

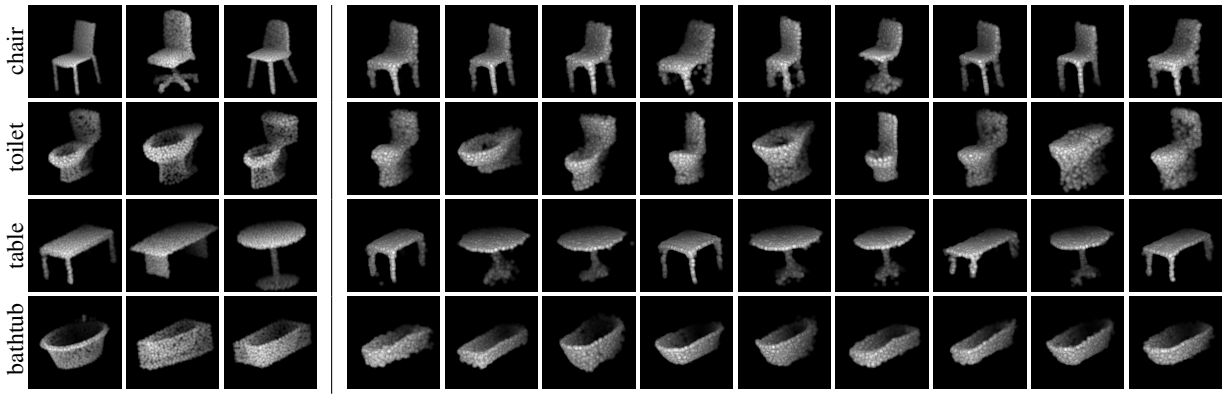


Figure 2: Generating 3D point clouds of objects. Each row shows one experiment, where the first three point clouds are three examples randomly selected from the training set. The rest are synthesized point clouds sampled from the short-run Langevin dynamics. The number of points in each example is 2,048. From top to bottom: chair, toilet, table, and bathtub.

flow-based model, or a latent variable model with $Z$ being the continuous latent variables in the following form

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

where $Z$ and $X$ have the same number of dimensions, and $Z$ follows a known prior (Gaussian) distribution $p_0$. $M_\theta$ is a short-run Langevin dynamics including $K$ Langevin steps in Eq.(4), which can be considered a $K$-layer residual network with noise injected into each layer and weight sharing at each layer. Let $\xi$ be all the randomness in $M_\theta$ due to the layer-wise injected noise. The model represented by a short-run MCMC shown in Eq.(6) can be trained by the "analysis by synthesis" scheme, where we update $\theta$ according to Eq.(3) and synthesize $\{\tilde{X}\}$ according to Eq.(6). Training $\theta$ with a short-run MCMC is no longer a maximum likelihood estimator but a moment matching estimator (MME) that solves the following estimating equation

$$\mathrm{E}_{q_{\mathrm{data}}}\left[\frac{\partial}{\partial \theta} f_\theta(X)\right] = \mathrm{E}_{p_\theta}\left[\frac{\partial}{\partial \theta} f_\theta(X)\right]. \tag{7}$$

Even though the learned $p_\theta$ based on short-run MCMC is $p_{\hat{\theta}_{\mathrm{MEE}}}$ rather than $p_{\hat{\theta}_{\mathrm{MLE}}}$, the $q_{\hat{\theta}_{\mathrm{MEE}}}$ is still a valid generator that is useful for 3D point cloud generation 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 $p_\theta$ in Eq.(1) and keep $q_\theta$ in Eq.(5) eventually, $p_\theta$ is crucial because $q$ is derived from $p$ and we learn $q$ under $p$. In other works, $p$ serves as an incubator of $q_{\hat{\theta}_{\mathrm{MEE}}}$.

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

The learning method based on noise-initialized short-run MCMC is similar to contrastive divergence [15], 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 size $\delta$ to minimize the bias caused by short-run MCMC. Also, the learning algorithm of our model 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 $\theta$, is called the neural tangent kernel [17], 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 these issues in our future work.

## 4. Experiments

We conduct experiments to test the proposed GPointNet model for point cloud modeling on a variety of tasks below. The code and more results can be found at: http://www.stat.ucla.edu/~jxie/GPointNet.

### 4.1. Synthesis

We evaluate our model for 3D point cloud synthesis on the ModelNet10, a 10-category subset of ModelNet [39] which is commonly used as a benchmark for 3D object analysis. We first create a point cloud dataset by sampling points uniformly from the mesh surface of each object in the Mod-

| | Model | JSD (↓) | MMD (↓) | | Coverage (↑) | |
|---|---|---|---|---|---|---|
| | | | CD | EMD | CD | EMD |
| night stand | r-GAN | 2.679 | 1.163 | 2.394 | 50.00 | 38.37 |
| | l-GAN | 1.000 | 0.746 | 1.563 | 44.19 | 39.53 |
| | PointFlow | **0.240** | 0.888 | 1.451 | 55.81 | 39.53 |
| | Ours | 0.590 | **0.692** | **1.148** | **59.30** | **61.63** |
| | Training Set | 0.263 | 0.793 | 1.096 | 60.40 | 52.32 |
| toilet | r-GAN | 3.180 | 2.995 | 2.891 | 17.00 | 16.00 |
| | l-GAN | 1.253 | 1.258 | 1.481 | 21.00 | 28.00 |
| | PointFlow | **0.362** | 0.965 | 1.513 | 39.00 | 33.00 |
| | Ours | 0.386 | **0.816** | **1.265** | **44.00** | **37.00** |
| | Training Set | 0.249 | 0.823 | 1.116 | 48.00 | 51.00 |
| monitor | r-GAN | 2.936 | 1.524 | 2.021 | 21.00 | 24.00 |
| | l-GAN | 1.653 | 0.915 | 1.349 | 28.00 | 27.00 |
| | PointFlow | **0.326** | 0.831 | 1.288 | 37.00 | 32.00 |
| | Ours | 0.780 | **0.803** | **1.213** | **40.00** | **38.00** |
| | Training Set | 0.283 | 0.554 | 0.938 | 48.00 | 53.00 |
| chair | r-GAN | 2.772 | 1.709 | 2.164 | 23.00 | 28.00 |
| | l-GAN | 1.358 | 1.419 | 1.480 | 23.00 | 26.00 |
| | PointFlow | **0.278** | 0.965 | 1.322 | 42.00 | 51.00 |
| | Ours | 0.563 | **0.889** | **1.280** | **56.00** | **57.00** |
| | Training Set | 0.365 | 0.858 | 1.190 | 54.00 | 59.00 |
| bathtub | r-GAN | 3.014 | 2.478 | 2.536 | 26.00 | 30.00 |
| | l-GAN | 0.928 | 0.865 | 1.324 | 32.00 | 38.00 |
| | PointFlow | **0.350** | **0.593** | 1.320 | 50.00 | 44.00 |
| | Ours | 0.460 | 0.660 | **1.108** | **58.00** | **50.00** |
| | Training Set | 0.344 | 0.652 | 0.980 | 56.00 | 52.00 |

| | Model | JSD (↓) | MMD (↓) | | Coverage (↑) | |
|---|---|---|---|---|---|---|
| | | | CD | EMD | CD | EMD |
| sofa | r-GAN | 1.866 | 2.037 | 2.247 | 13.00 | 23.00 |
| | l-GAN | 0.681 | 0.631 | **1.028** | **43.00** | 44.00 |
| | PointFlow | **0.244** | 0.585 | 1.313 | 34.00 | 33.00 |
| | Ours | 0.647 | **0.547** | 1.089 | 39.00 | **45.00** |
| | Training Set | 0.185 | 0.467 | 0.904 | 56.00 | 56.00 |
| bed | r-GAN | 1.973 | 1.250 | 2.441 | 27.00 | 21.00 |
| | l-GAN | 0.646 | **0.539** | **0.992** | 48.00 | 44.00 |
| | PointFlow | **0.219** | 0.544 | 1.230 | **50.00** | 35.00 |
| | Ours | 0.461 | 0.552 | 1.004 | **50.00** | **50.00** |
| | Training Set | 0.169 | 0.516 | 0.927 | 57.00 | 55.00 |
| table | r-GAN | 3.801 | 3.714 | 2.625 | 8.00 | 14.00 |
| | l-GAN | 4.254 | 1.232 | 2.166 | 14.00 | 9.00 |
| | PointFlow | 1.044 | 1.630 | 1.535 | 16.00 | 29.00 |
| | Ours | **0.869** | **0.640** | **1.000** | **44.00** | **37.00** |
| | Training Set | 0.703 | 1.218 | 1.182 | 31.00 | 38.00 |
| desk | r-GAN | 3.575 | 2.712 | 3.678 | 22.09 | 22.09 |
| | l-GAN | 2.233 | **1.139** | 2.345 | 38.37 | 25.58 |
| | PointFlow | **0.327** | 1.254 | **1.548** | 38.37 | 46.51 |
| | Ours | 0.454 | 1.223 | 1.567 | **56.98** | **52.33** |
| | Training Set | 0.329 | 1.055 | 1.332 | 53.48 | 50.00 |
| dresser | r-GAN | 1.726 | 1.299 | 1.675 | 36.05 | 30.23 |
| | l-GAN | 0.648 | 0.642 | 1.010 | 45.35 | 43.02 |
| | PointFlow | **0.270** | 0.715 | 1.349 | 46.51 | 37.21 |
| | Ours | 0.457 | **0.485** | **0.988** | **53.49** | **52.33** |
| | Training Set | 0.215 | 0.551 | 0.882 | 56.98 | 54.65 |

Table 1: Comparison of quality of point cloud synthesis on the ModelNet10. ↓: the lower the better, ↑: the higher the better. MMD-CD scores are multiplied by 100; MMD-EMD scores and JSDs are multiplied by 10.

elNet10 dataset, and then scale them into a range of [-1,1]. We train one single model for each category of point clouds. The number of training examples in each category ranges from 100 to 900. Each point cloud contains 2,048 points.

The network structure of the scoring function $f_\theta(X)$ is visualized in Figure 1. It first encodes each 3-dimensional point coordinate in Euclidean space to a 1,024-dimensional point feature by an MLP, then uses an average pooling layer to aggregate information from all the points to a single 1,024-dimensional global point cloud feature, and maps it to the score by another MLP. The scoring function is input-permutation-invariant because the MLP for point encoding is shared by all unordered points and also the output of the symmetric function, which is an average pooling layer followed by an MLP, is not affected by the point feeding order.

| Model | Category | CD | EMD | Category | CD | EMD |
|---|---|---|---|---|---|---|
| Ours | night stand | **0.378** | **0.685** | sofa | 0.427 | **0.703** |
| PointFlow | | 0.464 | 0.990 | | **0.389** | 0.888 |
| Ours | toilet | **0.396** | **0.708** | bed | **0.361** | **0.670** |
| PointFlow | | 0.456 | 0.992 | | 0.372 | 0.914 |
| Ours | monitor | **0.371** | **0.705** | table | **0.318** | **0.621** |
| PointFlow | | 0.441 | 0.957 | | 0.581 | 1.008 |
| Ours | chair | **0.337** | **0.719** | desk | **0.391** | **0.697** |
| PointFlow | | 0.510 | 1.028 | | 0.500 | 1.063 |
| Ours | bathtub | 0.321 | **0.612** | dresser | **0.329** | **0.645** |
| PointFlow | | **0.289** | 0.825 | | 0.415 | 0.942 |

Table 2: Comparison of performance in reconstruction on the ModelNet10. CD scores are multiplied by 100 and EMD scores are multiplied by 10. The lower the better.

We use Adam [20] for optimization with an initial learning rate 0.005, $\beta_1 = 0.9$ and $\beta_2 = 0.999$. We decay the learning rate by 0.985 for every 50 iterations. The mini-batch size is 128. The number of paralleled MCMC chains is 128. We run $K$ = 64 Langevin steps, with the step size $\delta = 0.005$. To avoid exploding gradients in MCMC, we clip the gradient values to a range [-1,1] at each Langevin step. We run 2,000 iterations for training. To further improve training, we inject additive Gaussian noises with stand deviation 0.01 to the observed examples at each iteration.

To quantitatively evaluate the performance of generative models of point clouds, we adopt three metrics that are also used in [1, 50], i.e., Jensen-Shannon Divergence (JSD), Coverage (COV) and Minimum Matching Distance (MMD). When evaluating COV and MMD, two point

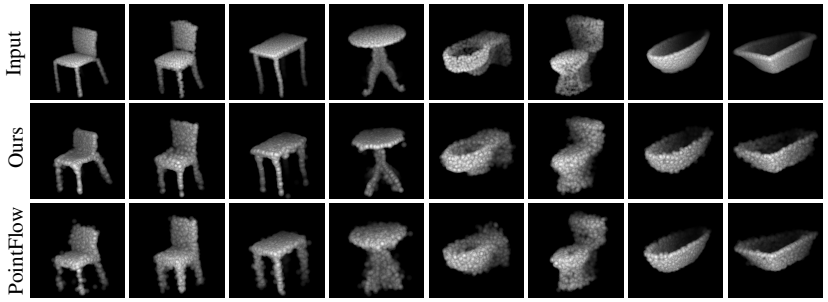


Figure 3: Point cloud reconstruction. A short-run MCMC as a generator is learned from chair, table, toilet and bathtub, respectively. The learned generator is applied to reconstruction by inferring the latent $Z$ to minimize the reconstruction error.

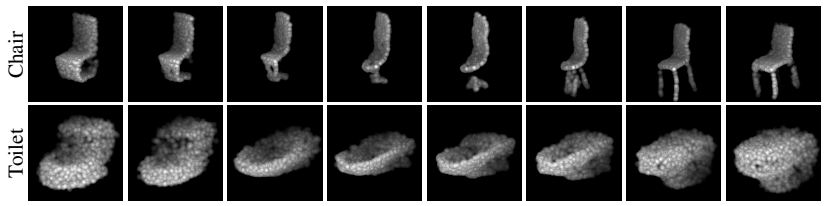


Figure 4: Point cloud interpolation between the generated examples at two ends. The transition in each row displays the sequence of $M_\theta(Z_\rho)$ with the linear interpolated latent variable $Z_\rho = \rho Z_1 + (1-\rho)Z_2$, where $\rho \in [0, 1]$. The left and right point clouds are $M_\theta(Z_1)$ and $M_\theta(Z_2)$, respectively.

| Method | Full | Generation |
|---|---|---|
| r-GAN | 7.22 | 6.91 |
| l-GAN | 1.97 | 1.71 |
| PointFlow | 1.61 | 1.06 |
| Ours | 1.39 | |

Table 3: A comparison of model sizes. Our method has only one network for both learning and generation.(Million)

| Method | Accuracy |
|---|---|
| SPH [18] | 79.8% |
| LFD [4] | 79.9% |
| PANORAMA-NN [33] | 91.1% |
| VConv-DAE [34] | 80.5% |
| 3D-GAN [38] | 91.0% |
| 3D-WINN [16] | 91.9% |
| 3D-DescriptorNet [44] | 92.4% |
| Primitive GAN [19] | 92.2% |
| FoldingNet [51] | 94.4% |
| l-GAN [1] | 95.4% |
| PointFlow [50] | 93.7% |
| Ours | 93.7% |

Table 4: A comparison of accuracy of 3D object classification on the 10-category ModelNet10 dataset.

clouds are measured by either Chamfer distance (CD) or earth mover's distance (EMD). We compare our model with some baseline generative models for point clouds, including PointFlow [50], l-GAN, and r-GAN, in Table 1. We report the performance of the baselines using their official codes. Figure 2 displays some examples of point clouds generated by our model for categories chair, toilet, table, and bathtub.

### 4.2. Reconstruction

We demonstrate the reconstruction ability of the GPoint-Net model for 3D point clouds. We learn our model with a 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 reconstructed point clouds obtained by the learned model, and the third row shows the results obtained by a baseline, PointFlow [50], which is a VAE-based framework. For VAE, the reconstruction can be easily achieved by first inferring the latent variables of the input example and then mapping the inferred latent variables back to the point cloud space via the generator. Table 2 shows a quantitative comparison of our method with PointFlow for point cloud reconstruction. CD and EMD metrics are adopted to measure the quality of the reconstruction. On the whole, our method outperforms the baseline.

As to model complexity, 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 GAN and VAE have different sizes of parameters in training and generation stages. Our model does not use an auxiliary network, thus it has less parameters.

### 4.3. Interpolation

We demonstrate the interpolation ability of our model. We learn the model with short-run MCMC. 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 $\rho$ discretized into 8 values within $[0, 1]$. We generate point clouds by $X_\rho = M_\theta(Z_\rho)$. Figure 4 shows two results of interpolation between $Z_1$ and $Z_2$ by showing the sequences of generated point clouds $\{X_\rho\}$. Smooth transition and physically plausible intermediate generated examples suggest that the generator learns a smooth latent space for point cloud embedding.

### 4.4. Representation learning for classification

The learned point encoder $h(x)$ in the scoring 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 GPointNet on the training examples from all categories in an unsupervised manner. The network $f_\theta(X)$ is the same as the one used in the previous sections, except that we add one layer with 2,048 channels before the average pooling and one layer with 1,024 channels after the average pooling. We replace the average pooling layer by a max-pooling layer in the learned scoring function and use the output of the max-pooling as point cloud features. Such a point cloud feature extractor is also permutation-invariant. We train an SVM [36] 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 11 published results on this dataset obtained by other baselines. Our method is on a par with other methods in terms of classification accuracy on this dataset.

We 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 cube $[-1, 1]^3$ into each point cloud. (3) Type 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 performance decreases as the corruption level (e.g., missing point ratio, added point ratio, and standard deviation of point perturbation) increases. In the case of missing points, even though $94\%$ points are deleted in each testing example, the classifier can still perform with an accuracy $90.20\%$. In the extreme case where we only keep 20 points ($1\%$) in each point cloud, the accuracy becomes $53.19\%$.

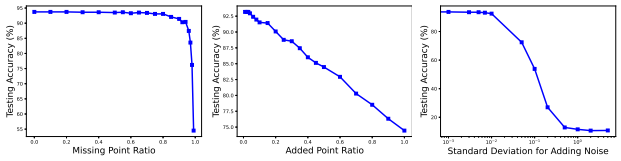


Figure 5: Robustness Test. The model is tested on ModelNet10 test set with three types of point corruptions. Classification accuracies are reported across different levels of corruptions. Left: missing points. Middle: added points. Right: point perturbation.

### 4.5. Visualization of point encoding function

The scoring function learns a coordinate encoding of each point and then aggregates all individual point codes into a score for the point set. The coordinate encoding function is implemented by an MLP, learning to encode each 3-dimensional point to a 2,048-dimensional vector in the model that we use for classification. To better understand what each encoding function learns, 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 6, we randomly visualize 4 filters at 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.

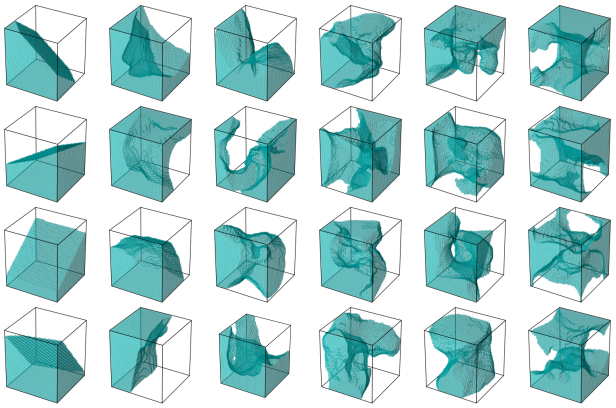


Figure 6: Visualization of point encoding functions. The point encoding function is implemented by a MLP. Each filter at different layers of the MLP is visualized by showing the points that have positive filter responses. Four filters randomly selected at each layer are visualized.

## 5. Conclusion

This paper studies the deep energy-based modeling and learning of unordered 3D point clouds. We propose a probability density of 3D point clouds, which is unordered point sets, in the form of the energy-based model where the energy function is parameterized by an input-permutation-invariant deep neural network. 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.

## Acknowledgment

The work is supported by NSF DMS-2015577, DARPA XAI project N66001-17-2-4029, ARO project W911NF1810296, ONR MURI project N00014-16-1-2007, and XSEDE grant ASC180018. We thank Erik Nijkamp for insightful discussions about short-run MCMC for EBM and neural tangent kernel.

## 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 (ICML)*, pages 40–49, 2018. 1, 3, 6, 7

[2] Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E Hinton. Layer normalization. *arXiv preprint arXiv:1607.06450*, 2016. 4

[3] Adrian Barbu and Song-Chun Zhu. *Monte Carlo Methods*. Springer, 2020. 2

[4] 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. 7

[5] Laurent Dinh, David Krueger, and Yoshua Bengio. NICE: non-linear independent components estimation. In Yoshua Bengio and Yann LeCun, editors, *International Conference on Learning Representations (ICLR) Workshop*, 2015. 2

[6] Laurent Dinh, Jascha Sohl-Dickstein, and Samy Bengio. Density estimation using real NVP. In *International Conference on Learning Representations (ICLR)*, 2017. 2

[7] Haoqiang Fan, Hao Su, and Leonidas J Guibas. A point set generation network for 3D object reconstruction from a single image. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 605–613, 2017. 1

[8] Matheus Gadelha, Rui Wang, and Subhransu Maji. Multiresolution tree networks for 3D point cloud processing. In *European Conference on Computer Vision (ECCV)*, pages 103–118, 2018. 1, 3

[9] Ruiqi Gao, Yang Lu, Junpei Zhou, Song-Chun Zhu, and Ying Nian Wu. Learning generative ConvNets via multi-grid modeling and sampling. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 9155–9164, 2018. 2

[10] Ruiqi Gao, Erik Nijkamp, Diederik P Kingma, Zhen Xu, Andrew M Dai, and Ying Nian Wu. Flow contrastive estimation of energy-based models. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 7518–7528, 2020. 2

[11] 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 (NIPS)*, pages 2672–2680, 2014. 1, 2

[12] Ulf Grenander. A unified approach to pattern analysis. In *Advances in Computers*, volume 10, pages 175–216. Elsevier, 1970. 2

[13] Ulf Grenander and Michael I Miller. *Pattern Theory: From Representation to Inference*. Oxford University Press, 2007. 2

[14] Tian Han, Erik Nijkamp, Xiaolin Fang, Mitch Hill, Song-Chun Zhu, and Ying Nian Wu. Divergence triangle for joint training of generator model, energy-based model, and inferential model. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 8670–8679, 2019. 2

[15] Geoffrey E Hinton. Training products of experts by minimizing contrastive divergence. *Neural Computation*, 14(8):1771–1800, 2002. 2, 4, 5

[16] Wenlong Huang, Brian Lai, Weijian Xu, and Zhuowen Tu. 3D volumetric modeling with introspective neural networks. In *The Thirty-Third AAAI Conference on Artificial Intelligence (AAAI)*, pages 8481–8488, 2019. 1, 7

[17] Arthur Jacot, Franck Gabriel, and Clément Hongler. Neural tangent kernel: Convergence and generalization in neural networks. In *Advances in Neural Information Processing Systems (NeurIPS)*, pages 8571–8580, 2018. 5

[18] Michael Kazhdan, Thomas Funkhouser, and Szymon Rusinkiewicz. Rotation invariant spherical harmonic representation of 3D shape descriptors. In *Symposium on Geometry Processing*, volume 6, pages 156–164, 2003. 7

[19] Salman H Khan, Yulan Guo, Munawar Hayat, and Nick Barnes. Unsupervised primitive discovery for improved 3d generative modeling. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 9739–9748, 2019. 7

[20] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *International Conference on Learning Representations (ICLR)*, 2015. 6

[21] Durk P Kingma and Prafulla Dhariwal. Glow: Generative flow with invertible 1x1 convolutions. In *Advances in Neural Information Processing Systems (NIPS)*, pages 10215–10224, 2018. 2

[22] Diederik P. Kingma and Max Welling. Auto-encoding variational bayes. In *International Conference on Learning Representations (ICLR)*, 2014. 1, 2

[23] Itai Lang, Asaf Manor, and Shai Avidan. Samplenet: differentiable point cloud sampling. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 7578–7588, 2020. 1

[24] Truc Le and Ye Duan. Pointgrid: A deep network for 3d shape understanding. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 9204–9214, 2018. 1

[25] Chun-Liang Li, Manzil Zaheer, Yang Zhang, Barnabas Poczos, and Ruslan Salakhutdinov. Point cloud GAN. *arXiv preprint arXiv:1810.05795*, 2018. 1, 3

[26] Jun S Liu. *Monte Carlo Strategies in Scientific Computing*. Springer Science & Business Media, 2008. 2

[27] Radford M Neal et al. MCMC using hamiltonian dynamics. *Handbook of Markov Chain Monte Carlo*, 2(11):2, 2011. 2

[28] Erik Nijkamp, Ruiqi Gao, Pavel Sountsov, Srinivas Vasudevan, Bo Pang, Song-Chun Zhu, and Ying Nian Wu. Learning energy-based model with flow-based backbone by neural transport mcmc. *arXiv preprint arXiv:2006.06897*, 2020. 2

[29] Erik Nijkamp, Mitch Hill, Tian Han, Song-Chun Zhu, and Ying Nian Wu. On the anatomy of MCMC-based maximum likelihood learning of energy-based models. In *The Thirty-Fourth AAAI Conference on Artificial Intelligence (AAAI)*, pages 5272–5280, 2020. 2

[30] Erik Nijkamp, Mitch Hill, Song-Chun Zhu, and Ying Nian Wu. Learning non-convergent non-persistent short-run MCMC toward energy-based model. In *Advances in Neural Information Processing Systems (NeurIPS)*, pages 5233–5243, 2019. 2, 4

[31] Charles R Qi, Hao Su, Kaichun Mo, and Leonidas J Guibas. Pointnet: Deep learning on point sets for 3D classification and segmentation. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 652–660, 2017. 1, 2

[32] 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 (NIPS)*, pages 5099–5108, 2017. 1, 2

[33] Konstantinos Sfikas, Theoharis Theoharis, and Ioannis Pratikakis. Exploiting the PANORAMA representation for convolutional neural network classification and retrieval. In *10th Eurographics Workshop on 3D Object Retrieval*, 2017. 7

[34] Abhishek Sharma, Oliver Grau, and Mario Fritz. VConv-DAE: Deep volumetric shape learning without object labels. In *European Conference on Computer Vision (ECCV)*, pages 236–250. Springer, 2016. 7

[35] Diego Valsesia, Giulia Fracastoro, and Enrico Magli. Learning localized generative models for 3D point clouds via graph convolution. In *International Conference on Learning Representations (ICLR)*, 2019. 1, 3

[36] Vladimir Vapnik. *The Nature of Statistical Learning Theory*. Springer, 2000. 8

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

[38] 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 (NIPS)*, pages 82–90, 2016. 1, 7

[39] Zhirong Wu, Shuran Song, Aditya Khosla, Fisher Yu, Linguang Zhang, Xiaoou Tang, and Jianxiong Xiao. 3D shapenets: A deep representation for volumetric shapes. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1912–1920, 2015. 1, 5

[40] Jianwen Xie, Yang Lu, Ruiqi Gao, and Ying Nian Wu. Cooperative learning of energy-based model and latent variable model via MCMC teaching. In *Thirty-Second AAAI Conference on Artificial Intelligence (AAAI)*, pages 4292–4301, 2018. 2

[41] Jianwen Xie, Yang Lu, Ruiqi Gao, Song-Chun Zhu, and Ying Nian Wu. Cooperative training of descriptor and generator networks. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 42(1):27–45, 2020. 2

[42] Jianwen Xie, Yang Lu, Song-Chun Zhu, and Yingnian Wu. A theory of generative ConvNet. In *International Conference on Machine Learning (ICML)*, pages 2635–2644, 2016. 1, 2

[43] Jianwen Xie, Zilong Zheng, Xiaolin Fang, Song-Chun Zhu, and Ying Nian Wu. Cooperative training of fast thinking initializer and slow thinking solver for conditional learning. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 2021. 2

[44] Jianwen Xie, Zilong Zheng, Ruiqi Gao, Wenguan Wang, Song-Chun Zhu, and Ying Nian Wu. Learning descriptor networks for 3D shape synthesis and analysis. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 8629–8638, 2018. 1, 2, 7

[45] Jianwen Xie, Zilong Zheng, Ruiqi Gao, Wenguan Wang, Song-Chun Zhu, and Ying Nian Wu. Generative VoxelNet: learning energy-based models for 3D shape synthesis and analysis. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 2020. 1, 2

[46] Jianwen Xie, Zilong Zheng, and Ping Li. Learning energy-based model with variational auto-encoder as amortized sampler. In *The Thirty-Fifth AAAI Conference on Artificial Intelligence (AAAI)*, 2021. 2

[47] Jianwen Xie, Song-Chun Zhu, and Ying Nian Wu. Synthesizing dynamic patterns by spatial-temporal generative ConvNet. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 7093–7101, 2017. 2

[48] Jianwen Xie, Song-Chun Zhu, and Ying Nian Wu. Learning energy-based spatial-temporal generative ConvNets for dynamic patterns. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 43(2):516–531, 2021. 2

[49] Saining Xie, Sainan Liu, Zeyu Chen, and Zhuowen Tu. Attentional shapecontextnet for point cloud recognition. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4606–4615, 2018. 1

[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 *IEEE International Conference on Computer Vision (ICCV)*, pages 4541–4550, 2019. 1, 3, 6, 7

[51] Yaoqing Yang, Chen Feng, Yiru Shen, and Dong Tian. Foldingnet: Point cloud auto-encoder via deep grid deformation. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 206–215, 2018. 7

[52] Manzil Zaheer, Satwik Kottur, Siamak Ravanbakhsh, Barnabas Poczos, Ruslan R Salakhutdinov, and Alexander J Smola. Deep sets. In *Advances in Neural Information Processing Systems (NIPS)*, pages 3391–3401, 2017. 1, 2

[53] Maciej Zamorski, Maciej Zięba, Rafał Nowak, Wojciech Stokowiec, and Tomasz Trzciński. Adversarial autoencoders for generating 3D point clouds. *arXiv preprint arXiv:1811.07605*, 2018. 1, 3