Learning Gradient Fields for Shape Generation

Ruojin Cai(✉), Guandao Yang, Hadar Averbuch-Elor, Zekun Hao, Serge Belongie, Noah Snavely, and Bharath Hariharan

Cornell University, Ithaca, USA
rc844@cornell.edu

Abstract. In this work, we propose a novel technique to generate shapes from point cloud data. A point cloud can be viewed as samples from a distribution of 3D points whose density is concentrated near the surface of the shape. Point cloud generation thus amounts to moving randomly sampled points to high-density areas. We generate point clouds by performing stochastic gradient ascent on an unnormalized probability density, thereby moving sampled points toward the high-likelihood regions. Our model directly predicts the gradient of the log density field and can be trained with a simple objective adapted from score-based generative models. We show that our method can reach state-of-the-art performance for point cloud auto-encoding and generation, while also allowing for extraction of a high-quality implicit surface. Code is available at https://github.com/RuojinCai/ShapeGF.

Keywords: 3D generation · Generative models

1 Introduction

Point clouds are becoming increasingly popular for modeling shapes, as many modern 3D scanning devices process and output point clouds. As such, an increasing number of applications rely on the recognition, manipulation, and synthesis of point clouds. For example, an autonomous vehicle might need to detect cars in sparse LiDAR point clouds. An augmented reality application might need to scan in the environment. Artists may want to further manipulate scanned objects to create new objects and designs. A *prior* for point clouds would be useful for these applications as it can densify LiDAR clouds, create additional training data for recognition, complete scanned objects or synthesize new ones. Such a prior requires a powerful generative model for point clouds.

R. Cai and G. Yang—Equal contribution.

Electronic supplementary material The online version of this chapter (https://doi.org/10.1007/978-3-030-58580-8_22) contains supplementary material, which is available to authorized users.

© Springer Nature Switzerland AG 2020
A. Vedaldi et al. (Eds.): ECCV 2020, LNCS 12348, pp. 364–381, 2020.
https://doi.org/10.1007/978-3-030-58580-8_22

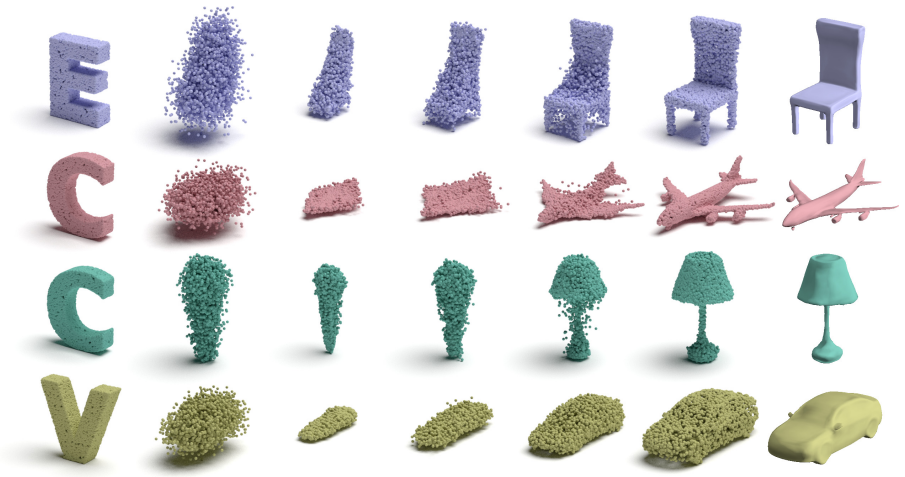

Fig. 1. To generate shapes, we sample points from an arbitrary prior (depicting the letters "E", "C", "C", "V" in the examples above) and move them stochastically along a learned gradient field, ultimately reaching the shape's surface. Our learned fields also enable extracting the surface of the shape, as demonstrated on the right.

In this work, we are interested in learning a generative model that can sample shapes represented as point clouds. A key challenge here is that point clouds are sets of arbitrary size. Prior work often generates a fixed number of points instead [1,16,17,48,64]. This number, however, may be insufficient for some applications and shapes, or too computationally expensive for others. Instead, following recent works [31,52,60], we consider a point cloud as a set of *samples* from an underlying distribution of 3D points. This new perspective not only allows one to generate an arbitrary number of points from a shape, but also makes it possible to model shapes with varying topologies. However, it is not clear how to best parameterize such a distribution of points, and how to learn it using *only* a limited number of sampled points.

Prior research has explored modeling the distribution of points that represent the shape using generative adversarial networks (GANs) [31], flow-based models [60], and autoregressive models [52]. While substantial progress has been made, these methods have some inherent limitations for modeling the distribution representing a 3D shape. The training procedure can be unstable for GANs or prohibitively slow for invertible models, while autoregressive models assume an ordering, restricting their flexibility for point cloud generation. Implicit representations such as DeepSDF [44] and OccupancyNet [36] can be viewed as modeling this probability density of the 3D points directly, but these models require ground truth signed distance fields or occupancy fields, which are difficult to obtain from point cloud data alone without corresponding meshes.

In this paper, we take a different approach and focus on the end goal – being able to draw an arbitrary number of samples from the distribution of points.

Working backward from this goal, we observe that the sampling procedure can be viewed as moving points from a generic prior distribution towards high likelihood regions of the shape (i.e., the surface of the shape). One way to achieve that is to move points gradually, following the gradient direction, which indicates where the density grows the most [57]. To perform such sampling, one only needs to model the gradient of log-density (known as the *Stein score function* [34]). In this paper, we propose to model a shape by learning the gradient field of its log-density. To learn such a gradient field from a set of sampled points from the shape, we build upon a *denoising score matching* framework [27,50]. Once we learn a model that outputs the gradient field, the sampling procedure can be done using a variant of stochastic gradient ascent (i.e. *Langevin dynamics* [50,57]).

Our method offers several advantages. First, our model is trained using a simple L_2 loss between the predicted and a "ground-truth" gradient field estimated from the input point cloud. This objective is much simpler to optimize than adversarial losses used in GAN-based techniques. Second, because it models the gradient directly and does not need to produce a normalized distribution, it imposes minimal restrictions on the model architecture in comparison to flow-based or autoregressive models. This allows us to leverage more expressive networks to model complicated distributions. Because the partition function need not be estimated, our model is also much *faster* to train. Finally, our model is able to furnish an implicit surface of the shape, as shown in Fig. 1, without requiring ground truth surfaces during training. We demonstrate that our technique can achieve state-of-the-art performance in both point cloud auto-encoding and generation. Moreover, our method can retain the same performance when trained with much sparser point clouds.

Our key contributions can be summarized as follows:

- We propose a novel point cloud generation method by extending score-based generative models to learn conditional distributions.
- We propose a novel algorithm to extract high-quality implicit surfaces from the learned model without the supervision from ground truth meshes.
- We show that our model can achieve state-of-the-art performance for point cloud auto-encoding and generation.

2 Related Work

Point Cloud Generative Modeling. Point clouds are widely used for representing and generating 3D shapes due to their simplicity and direct relation to common data acquisition techniques (LiDARs, depth cameras, etc.). Earlier generative models either treat point clouds as a fixed-dimensional matrix (i.e. $N \times 3$ where N is predefined) [1,16,17,48,52,55,63,64], or relies on heuristic set distance functions such as Chamfer distance and Earth Mover Distance [5,11,17,23,61]. As pointed out in Yang *et al.* [60] and Sect. 1, both of these approaches lead to several drawbacks. Alternatively, we can model the point cloud as samples from a distribution of 3D points. Toward this end, Sun *et al.* [52] applies an autoregressive model to model the distribution of points, but it requires assuming an

ordering while generating points. Li *et al.* [31] applies a GAN [3,24] on both this distribution of 3D points as well as the distribution of shapes. PointFlow [60] applies normalizing flow [43] to model such distribution, so sampling points amounts to moving them to the surface according to a learned vector field. In addition to modeling the movement of points, PointFlow also tracks the change of volume in order to normalize the learned distribution, which is computationally expensive [8]. While our work applies a GAN to learn the distribution of latent code similar to Li *et al.* and Achilioptas *et al.*, we take a different approach to model the distribution of 3D points. Specifically, we predict the gradient of log-density field to model the non-normalized probability density, thus circumventing the need to compute the partition function and achieves faster training time with a simple L2 loss.

Generating Other 3D Representations. Common representations emerged for deep generative 3D modeling include voxel-based [21,59], mesh-based [2,18,25,33,45,53], and assembly-based techniques [32,38]. Recently, implicit representations are gaining increasing popularity, as they are capable of representing shapes with high level of detail [10,36,37,44]. They also allow for learning a structured decomposition of shapes, representing local regions with Gaussian functions [19,20] or other primitives [26,49,54]. In order to reconstruct the mesh surface from the learned implicit field, these methods require finding the zero isosurface of the learned occupancy field (e.g. using the Marching Cubes algorithm [35]). Our learned gradient field also allows for high-quality surface reconstruction using similar methods. However, we do not require prior information on the shape (e.g., signed distance values) for training, which typically requires a watertight input mesh. Recently, SAL [4] learns a signed distance field using only point cloud as supervision. Different from SAL, our model directly outputs the gradients of the log-density instead field of the signed distance, which allows our model to use arbitrary network architecture without any constraints. As a result, our method can scale to more difficult settings such as train on larger dataset (e.g. ShapeNet [6]) or train with sparse scanned point clouds.

Energy-Based Modeling. In contrast to flow-based models [8,12,22,28,46,60] and auto-regressive models [40–42,52], energy-based models learn a non-normalized probability distribution [29], thus avoid computation to estimate the partition function. It has been successfully applied to tasks such as image segmentation [14,15], where a normalized probability density function is hard to define. Score matching was first proposed for modeling energy-based models in [27] and deals with "matching" the model and the observed data log-density gradients, by minimizing the squared distance between them. To improve its performance and scalability, various extensions have been proposed, including denoising score matching [56] and sliced score matching [51]. Most recently, Song and Ermon [50] introduced data perturbation and annealed Langevin dynamics to the original denoising score matching method, providing an effective way to model data embedded on a low dimensional manifold. Their method was applied to the image generation task, achieving performance comparable to GANs. In this work,

we extend this method to model conditional distributions and demonstrate its suitability to the task of point cloud generation, viewing point clouds as samples from the 2D manifold (shape surface) in 3D space.

3 Method

In this work, we are interested in learning a generative model that can sample shapes represented as point clouds. Therefore, we need to model two distributions. First, we need to model the distribution of shapes, which encode how shapes vary across an entire collection of shapes. Once we can sample a particular shape of interest, then we need a mechanism to sample a point clouds from its surface. As previously discussed, a point cloud is best viewed as samples from a distribution of 3D (or 2D) points, which encode a particular shape. To sample point clouds of arbitrary size for this shape, we also need to model this distribution of points.

Specifically, we assume a set of shapes $\mathcal{X} = \{X^{(i)}\}_{i=1}^{N}$ are provided as input. Each shape in $\mathcal{X}$ is represented as a point cloud sampled from its surface, defined by $X^{(i)} = \{x_j^i\}_{j=1}^{M_i}$. Our goal is to learn both the distribution of shapes and the distribution of points, conditioned on a particular shape from the data. To achieve that, we first propose a model to learn the distribution of points encoding a shape from a set of points $X^{(i)}$ (Sect. 3.1– 3.5). Then we describe how to model the distribution of shapes from the set of point clouds (i.e. $\mathcal{X}$) in Sect. 3.6.

3.1 Shapes as a Distribution of 3D Points

We would like to define a distribution of 3D points $P(x)$ such that sampling from this distribution will provide us with a surface point cloud of the object. Thus, the probability density encoding the shape should concentrate on the shape surface. Let S be the set of points on the surface and $P_S(x)$ be the uniform distribution over the surface. Sampling from $P_S(x)$ will create a point cloud uniformly sampled from the surface of interest. However, this distribution is hard to work with: for all points that are not in the surface $x \notin S$, $P_S(x) = 0$. As a result, $P_S(x)$ is discontinuous and has usually zero support over its ambient space (i.e. $\mathbb{R}^3$), which impose challenges in learning and modeling. Instead, we approximate $P_S(x)$ by smoothing the distribution with a Gaussian kernel:

$$Q_{\sigma,S}(x) = \int_{s \in \mathbb{R}^3} P_S(s)\mathcal{N}(x; s, \sigma^2 I)ds. \tag{1}$$

As long as the standard deviation σ is small enough, $Q_{\sigma,S}(x)$ will approximate the true data distribution $P_S(x)$ whose density concentrates near the surface. Therefore, sampling from $Q_{\sigma,S}(x)$ will yield points near the surface S.

As discussed in Sect. 1, instead of modeling $Q_{\sigma,S}$ directly, we will model the gradient of the logarithmic density (i.e. $\nabla_x \log Q_{\sigma,S}(x)$). Sampling can then be performed by starting from a prior distribution and performing gradient ascent on this field, thus moving points to high probability regions.

In particular, we will model the gradient of the log-density using a neural network $g_\theta(x, \sigma)$, where x is a location in 3D (or 2D) space. We will first analyze several properties of this gradient field $\nabla_x \log Q_{\sigma,S}(x)$. Then we describe how we train this neural network and how we sample points using the trained network.

3.2 Analyzing the Gradient Field

In this section we provide an interpretation of how $\nabla_x \log Q_{\sigma,S}(x)$ behaves with different σ's. Computing a Monte Carlo approximation of $Q_{\sigma,S}(x)$ using a set of observations $\{x_i\}_{i=1}^m$, we obtain a mixture of Gaussians with modes centered at x_i and radially-symmetric kernels:

$$Q_{\sigma,S}(x) = \mathbb{E}_{s\sim P_S}\left[\mathcal{N}(x; s, \sigma^2 I)\right] \approx \frac{1}{m}\sum_{i=1}^{m}\mathcal{N}(x; x_i, \sigma^2 I) \triangleq A_\sigma(x, \{x_i\}_{i=1}^m).$$

The gradient field can thus be approximated by the gradient of the logarithmic of this Gaussian mixture:

$$\nabla_x \log A_\sigma(x, \{x_i\}_{i=1}^m) = \frac{1}{\sigma^2}\left(-x + \sum_{i=1}^{m} x_i w_i(x, \sigma)\right), \tag{2}$$

where the weight $w_{ij}(x, \sigma)$ is computed from a softmax with temperature $2\sigma^2$:

$$w_i(x, \sigma) = \frac{\exp\left(-\frac{1}{2\sigma^2}\|x - x_i\|^2\right)}{\sum_{j=1}^{m}\exp\left(-\frac{1}{2\sigma^2}\|x - x_j\|^2\right)}. \tag{3}$$

Since $\sum_i w_i(x, \sigma) = 1$, $\sum_i x_i w_i(x, \sigma)$ falls within the convex hull created by the sampled surface points $\{x_i\}_{i=1}^m$. Therefore, the direction of this gradient of the logarithmic density field points from the sampled location towards a point inside the convex hull of the shape. When the temperature is high (i.e. σ is large), then the weights $w_i(x, \sigma)$ will be roughly the same and $\sum_i x_i w_i(x, \sigma)$ behaves like averaging all the x_i's. Therefore, the gradient field will point to a coarse shape that resembles an average of the surface points. When the temperature is low (i.e. σ is small), then $w_i(x, \sigma)$ will be close to 0 except when x_i is the closest to x. As a result, $\sum_i x_i w_i(x, \sigma)$ will behave like an $\text{argmin}_{x_i}\|x_i - x\|$. The gradient direction will thus point to the nearest point on the surface. In this case, the norm of the gradient field approximates a distance field of the surface up to a constant σ^{-2}. This allows the gradient field to encode fine details of the shape and move points to the shape surface more precisely. Figure 2 shows a visualization of the field in the 2D case for a series of different σ's.

3.3 Training Objective

As mentioned in Sect. 3.1, we would like to train a deep neural network $g_\theta(x, \sigma)$ to model the gradient of log-density: $\nabla_x \log Q_{\sigma,S}(x)$. One simple objective achieving this is minimizing the L2 loss between them [27]:

$$\ell_{\text{direct}}(\sigma, S) = \mathbb{E}_{x\sim Q_{\sigma,S}(x)}\left[\frac{1}{2}\left\|g_\theta(x, \sigma) - \nabla_x \log Q_{\sigma,S}(x)\right\|_2^2\right]. \tag{4}$$

However, optimizing such an objective is difficult as it is generally hard to compute $\nabla_x \log Q_{\sigma,S}(x)$ from a finite set of observations.

Inspired by *denoising score matching* methods [50,56], we can write $Q_{\sigma,S}(x)$ as a perturbation of the data distribution $P_S(x)$, produced with a Gaussian noise with standard deviation σ. Specifically, $Q_{\sigma,S}(x) = \int P_S(s) q_\sigma(\tilde{x}|s) dx$, where $q_\sigma(\tilde{x}|s) = \mathcal{N}(\tilde{x}|s, \sigma^2 I)$. As such, optimizing the objective in Eq. 4 can be shown to be equivalent to optimizing the following [56]:

$$\ell_{\mathsf{denoising}}(\sigma, S) = \mathbb{E}_{s\sim P_S, \tilde{x}\sim q_\sigma(\tilde{x}|s)} \left[\frac{1}{2} \| g_\theta(\tilde{x}, \sigma) - \nabla_{\tilde{x}} \log q_\sigma(\tilde{x}|s) \|_2^2 \right]. \tag{5}$$

Since $\nabla_{\tilde{x}} \log q_\sigma(\tilde{x}|s) = \frac{s-\tilde{x}}{\sigma^2}$, this loss can be easily computed using the observed point cloud $X = \{x_j\}_{j=1}^m$ as following:

$$\ell(\sigma, X) = \frac{1}{|X|} \sum_{x_i \in X} \| g_\theta(\tilde{x_i}, \sigma) - \frac{x_i - \tilde{x}_i}{\sigma^2} \|_2^2, \quad \tilde{x_i} \sim \mathcal{N}(x_i, \sigma^2 I). \tag{6}$$

Multiple Noise Levels. One problem with the abovementioned objective is that most $\tilde{x_i}$ will concentrate near the surface if σ is small. Thus, points far away from the surface will not be supervised. This can adversely affect the sampling quality, especially when the prior distribution puts points to be far away from the surface. To alleviate this issue, we follow Song and Ermon [50] and train g_θ for multiple σ's, with $\sigma_1 \geq \cdots \geq \sigma_k$. Our final model is trained by jointly optimizing $\ell(\sigma_i, X)$ for all σ_i. The final objective is computed empirically as:

$$\mathcal{L}(\{\sigma_i\}_{i=1}^k, X) = \sum_{i=1}^k \lambda(\sigma_i) \ell(\sigma_i, X), \tag{7}$$

where $\lambda(\sigma_i)$ are parameters weighing the losses $\ell(\sigma_i, X)$. $\lambda(\sigma_i)$ is chosen so that the weighted losses roughly have the same magnitude during training.

3.4 Point Cloud Sampling

Sampling a point cloud from the distribution is equivalent to moving points from a prior distribution to the surface (i.e. the high-density region). Therefore, we can perform stochastic gradient ascent on the logarithmic density field. Since $g_\theta(x, \sigma)$ approximates the gradient of the log-density field (i.e. $\nabla_x \log Q_{\sigma,S}(x)$), we could thus use $g_\theta(x, \sigma)$ to update the point location x. In order for the points to reach all the local maxima, we also need to inject random noise into this process. This amounts to using Langevin dynamics to perform sampling [57].

Specifically, we first sample a point x_0 from a prior distribution π. The prior is usually chosen to be simple distribution such as a uniform or a Gaussian distribution. We empirically demonstrate that the sampling performance won't be affected as long as the prior points are sampled from places where the perturbed points would reach during training. We then perform the following recursive update with step size $\alpha > 0$:

$$x_{t+1} = x_t + \frac{\alpha}{2} g_\theta(x_t, \sigma) + \sqrt{\alpha} \epsilon_t, \quad \epsilon_t \sim \mathcal{N}(0, I). \tag{8}$$

Under mild conditions, $p(x_T)$ converges to the data distribution $Q_{\sigma,S}(x)$ as $T \to \infty$ and $\epsilon \to 0$ [57]. Even when such conditions fail to hold, the error in Eq. 8 is usually negligible when α is small and T is large [9,13,39,50].

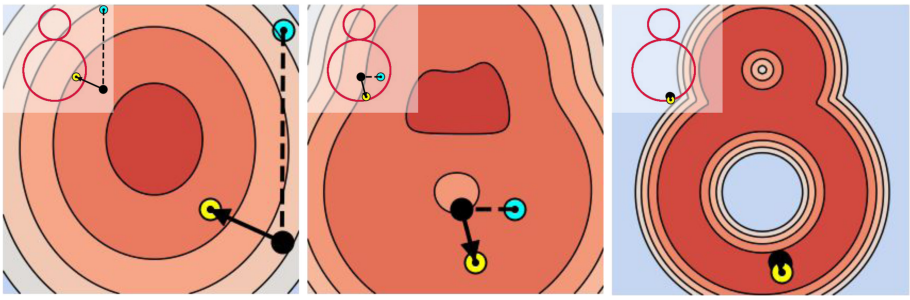

Fig. 2. Log density field with different σ (biggest to smallest) and a Langevin Dynamic point update step with that σ. Deeper color indicates higher density. The ground truth shape is shown in the upper left corner. Dotted line indicated Gaussian noise and solid arrows indicates gradient step. As sigma decreases, the log-density field changes from coarse to fine, and points are moved closer to the surface.

Prior works have observed that a main challenge for using Langevin dynamics is its slow mixing time [50,58]. To alleviate this issue, Song and Ermon [50] propose an annealed version of Langevin dynamics, which gradually anneals the noise for the score function. Specifically, we first define a list of σ_i with $\sigma_1 \geq \cdots \geq \sigma_k$, then train one single denoising score matching model that could approximate q_{σ_i} for all i. Then, annealed Langevin dynamics will recursively compute the x_t while gradually decreasing σ_i:

$$x'_{t+1} = x_t + \frac{\sqrt{\alpha}\sigma_i \epsilon_t}{\sigma_k}, \quad \epsilon_t \sim \mathcal{N}(0, I), \tag{9}$$

$$x_{t+1} = x'_{t+1} + \frac{\alpha \sigma_i^2}{2\sigma_k^2} g_\theta(x'_{t+1}, \sigma_i). \tag{10}$$

Figure 2 demonstrates the sampling across the annealing process in a 2D point cloud. As discussed in Sect. 3.3, larger σ's correspond to coarse shapes while smaller σ's correspond to fine shape. Thus, this annealed Langevin dynamics can be thought of as a coarse-to-fine refinement of the shape. Note that we make the noise perturnbation step before the gradient update step, which leads to cleaner point clouds. The supplementary material contains detailed hyperparameters.

3.5 Implicit Surface Extraction

Next we show that our learned gradient field (e.g. g_θ) also allows for obtaining an implicit surface. The key insight here is that the surface is defined as the set

of points that reach the maximum density in the data distribution $P_S(x)$, and thus these points have *zero* gradient. Another interpretation is that when σ is small enough (i.e. $Q_{\sigma,S}(x)$ approximates the true data distribution $p(x)$), the gradient for points near the surface will point to its nearest point on the surface, as described in Sect. 3.2:

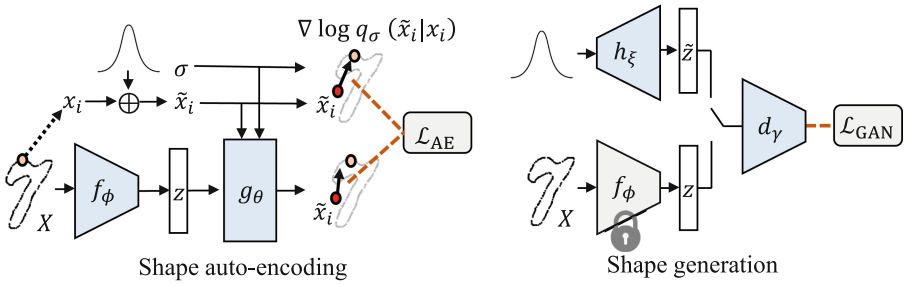

Fig. 3. Illustration of training pipe for shape auto-encoding and generation.

$$g_\theta(x,\sigma) \approx \frac{1}{\sigma^2}\left(-x + \text{argmin}_{s\in S}\,\|x-s\|\right). \tag{11}$$

Thus, for a point near the surface, its norm equals *zero* if and only if $x \in S$ (provided the arg min is unique). Therefore, the shape can be approximated by the zero iso-surface of the gradient norm:

$$S \approx \{x \mid \|g_\theta(x,\sigma)\| = \delta\}, \tag{12}$$

for some $\delta > 0$ that is sufficiently small. One caveat is that points for which the arg min in Eq. 11 is not unique may also have a zero gradient. These correspond to *local minimas* of the likelihood. In practice, this is seldom a problem for surface extraction, and it is possible to discard these regions by conducting the second partial derivative test.

Also as mentioned in Sect. 3.2, when the σ is small, the norm of the gradient field approximates a distance field of the surface, scaled by a constant σ^{-2}. This allows us to retrieval the surface S efficiently using an off-the-shelf ray-casting technique [47] (see Figs. 1, 4 and 5).

3.6 Generating Multiple Shapes

In the previous sections, we focused on learning the distribution of points that represent a single shape. Our next goal is to model the distribution of shapes. We, therefore, introduce a latent code z to encode which specific shape we want to sample point clouds from. Furthermore, we adapt our gradient decoder to be conditional on the latent code z (in addition to σ and the sampled point).

As illustrated in Fig. 3, the training is conducted in two stages. We first train an auto-encoder with an encoder f_ϕ that takes a point cloud and outputs the latent code z. The gradient decoder is provided with z as input and produces a gradient field with noise level σ. The auto-encoding loss is thus:

$$\mathcal{L}_{AE}(\mathcal{X}) = \mathop{\mathbb{E}}_{X\sim\mathcal{X}} \left[\frac{1}{2|X|} \sum_{x\in X, \sigma_i} \lambda(\sigma_i) \left\| g_\theta(\tilde{x}, f_\phi(X), \sigma_i) - \frac{x - \tilde{x}}{\sigma_i^2} \right\|_2^2 \right], \tag{13}$$

where each $\tilde{x_j}$ is drawn from a $\mathcal{N}(x_j, \sigma_i^2 I)$ for a corresponding σ_i. This first stage provides us with a network that can model the distribution of points representing the shape encoded in the latent variable z. Once the auto-encoder is fully trained, we apply a latent-GAN [1] to learn the distribution of the latent code $p(z) = p(f_\phi(X))$, where X is a point cloud sampled from the data distribution. Doing so provides us with a generator h_ξ that can sample a latent code from $p(z)$, allowing us control over which shape will be generated. To sample a novel shape, we first sample a latent code $\tilde{z}$ using h_ξ. We can then use the trained gradient decoder g_θ to sample point clouds or extract an implicit surface from the shape represented as z. For more details about hyperparameters and model architecture, please refer to the supplementary material.

4 Experiments

In this section, we will evaluate our model's performance in point cloud auto-encoding (Sect. 4.1), up-sampling (Sect. 4.1), and generation (Sect. 4.2) tasks. Finally, we present an ablation study examining our model design choices (Sect. 4.3). Implementation details will be shown in the supplementary materials.

Datasets. Our experiments focus mainly on two datasets: MNIST-CP and ShapeNet. MNIST-CP was recently proposed by Yifan *et al.* [62] and consists of 2D contour points extracted from the MNIST [30] dataset, which contains 50K and 10K training and testing images. Each point cloud contains 800 points. The ShapeNet [7] dataset contains 35708 shapes in training set and 5158 shapes in test set, capturing 55 categories. For our method, we normalize all point clouds by centering their bounding boxes to the origin and scaling them by a constant such that all points range within the cube $[-1, 1]^3$ (or the square in the 2D case).

Evaluation Metrics. Following prior works [1,23,60], we use the symmetric Chamfer Distance (CD) and the Earth Mover's Distance (EMD) to evaluate the reconstruction quality of the point clouds. To evaluate the generation quality, we use metrics in Yang *et al.* [60] and Achlioptas *et al.* [1]. Specifically, Achlioptas *et al.* [1] suggest using Minimum Matching Distance (MMD) to measure fidelity of the generated point cloud and Coverage (COV) to measure whether the set of generated samples cover all the modes of the data distribution. Yang *et al.* [60]

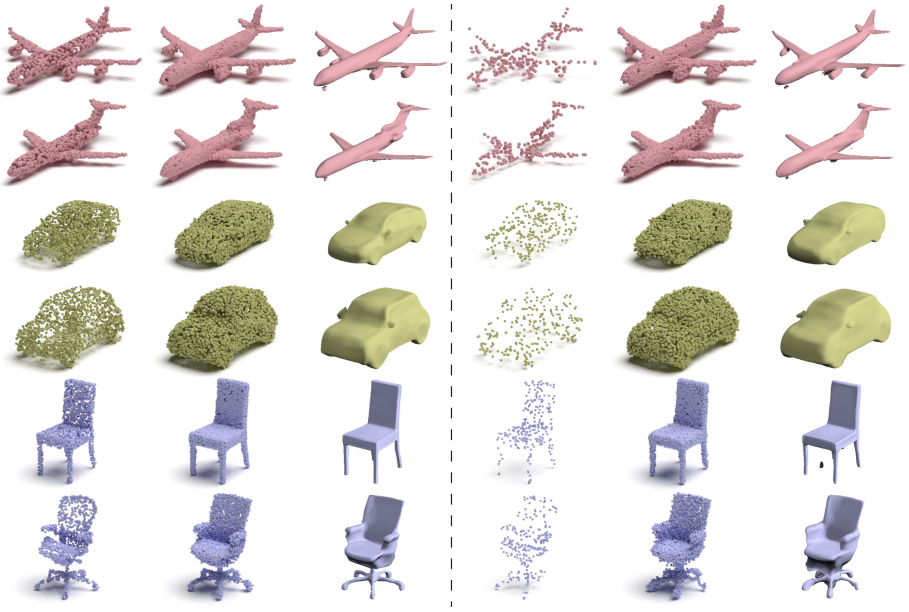

Fig. 4. Shape auto-encoding test results. Our model can accurately reconstruct shapes given 2048 points (left) or only 256 points (right) describing the shape. Output point clouds are illustrated in the center and implicit surfaces on the left.

propose to use the accuracy of a k-NN classifier performing two-sample tests. The idea is that if the sampled shapes seem to be drawn from the actual data distribution, then the classifier will perform like a random guess (i.e. results in 50% accuracy). To evaluate our results, we first conduct per-shape normalization to center the bounding box of the shape and scale its longest length to be 2, which allows the metric to focus on the geometry of the shape and not the scale.

4.1 Shape Auto-encoding

In this section, we evaluate how well our model can learn the underlying distribution of points by asking it to auto-encode a point cloud. We conduct the auto-encoding task for five settings: all 2D point clouds in MNIST-CP, 3D point clouds on the whole ShapeNet, and three categories in ShapeNet (Airplane, Car, Chair). In this experiment, our method is compared with the current state-of-the-art AtlasNet [23] with patches and with sphere. Furthermore, we also compare against Achilioptas *et al.* [1] which predicts point clouds as a fixed-dimensional array, and PointFlow [60] which uses a flow-based model to represent the distribution. We follow the experiment set-up in PointFlow to report performance in both CD and EMD in Table 1. Since these two metrics depend on the scale of the point clouds, we also report the upper bound in the "oracle" column. The upper

Table 1. Shape auto-encoding on the MNIST-CP and ShapeNet datasets. The best results are highlighted in bold. CD is multiplied by 10^4 and EMD is multiplied by 10^2.

Dataset	Metric	l-GAN [1]		AtlasNet [23]		PF [60]	Ours	Oracle
		CD	EMD	Sphere	Patches			
MNIST-CP	CD	8.204	–	7.274	4.926	17.894	**2.669**	1.012
	EMD	40.610	–	19.920	15.970	8.705	**7.341**	4.875
Airplane	CD	1.020	1.196	1.002	0.969	1.208	**0.96**	0.837
	EMD	4.089	2.577	2.672	2.612	2.757	**2.562**	2.062
Chair	CD	9.279	11.21	6.564	6.693	10.120	**5.599**	3.201
	EMD	8.235	6.053	5.790	5.509	6.434	**4.917**	3.297
Car	CD	5.802	6.486	5.392	5.441	6.531	**5.328**	3.904
	EMD	5.790	4.780	4.587	4.570	5.138	**4.409**	3.251
ShapeNet	CD	7.120	8.850	5.301	**5.121**	7.551	5.154	3.031
	EMD	7.950	5.260	5.553	5.493	5.176	**4.603**	3.103

bound is produced by computing the error between two different point clouds with the same number of points sampled from the same underlying meshes.

Our method consistently outperforms all other methods on the EMD metric, which suggests that our point samples follow the distribution or they are more uniformly distributed among the surface. Note that our method outperforms PointFlow in both CD and EMD for all datasets, but requires much less time to train. Our training for the Airplane category can be completed in about less than 10 h, yet reproducing the results for PointFlow's pretrained model takes at least two days. Our method can even sometimes outperform Achilioptas *et al.*and AtlasNet in CD, which is the loss they are directly optimizing at.

Table 2. Auto-encoding sparse point clouds. We randomly sample N points from each shape (in the Airplane dataset). During training, the model is provided with M points (the columns). CD is multiplied by 10^4 and EMD is multiplied by 10^2.

N	CD					EMD				
	2048	1024	512	256	128	2048	1024	512	256	128
$10K$	0.993	1.057	0.999	1.136	1.688	2.463	2.608	2.589	3.042	3.715
$3K$	1.080	1.059	1.003	1.142	1.753	2.533	2.586	2.557	2.997	3.878
$1K$	–	–	1.021	1.149	1.691	–	–	2.565	2.943	3.633

Point Cloud Upsampling. We conduct a set of experiments on subsampled ShapeNet point clouds. These experiments are primarily focused on showing that

(i) our model can learn from sparser datasets, and that (ii) we can infer a dense shape from a sparse input. In the regular configuration (reported above), we learn from $N = 10K$ points which are uniformly sampled from each shape mesh model. During training, we sample $M = 2048$ points (from the $10K$ available in total) to be the input point cloud. To evaluate our model, we perform the Langevin dynamic procedure (described in Sect. 3.4) over 2048 points sampled from the prior distribution and compare these to 2048 points from the reference set.

To evaluate whether our model can effectively upsample point clouds and learn from a sparse input, we train models with $N = [1K, 3K, 10K]$ and $M = [128, 256, 512, 1024, 2048]$ on the Airplane dataset. To allow for a fair comparison, we evaluate all models using the same number of output points (i.e. 2048 points are sampled from the prior distribution in all cases). As illustrated in Table 2, we obtain comparable auto-encoding performance while training with significantly sparser shapes. Interestingly, the number of points available from the model (i.e. N) does not seem to affect performance, suggesting that we can indeed learn from sparser datasets. Several qualitative examples auto-encoding shapes from the regular and sparse configurations are shown in Fig. 4. We also demonstrate that our model can also provide a smooth iso-surface, even when only a sparse point cloud (i.e. 256 points) is provided as input.

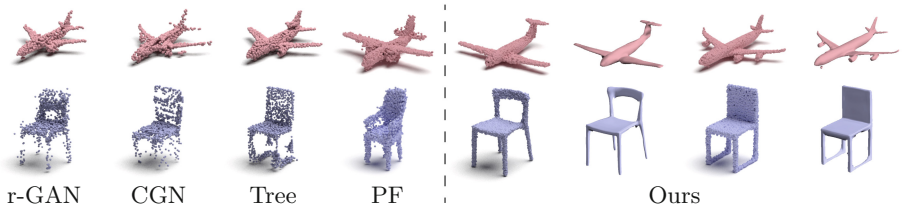

Fig. 5. Generation results. We shown results from r-GAN, GCN, TreeGAN (Tree), and PointFlow (PF) are illustrated on the left for comparison. Generated point clouds are illustrated alongside the corresponding implicit surfaces.

4.2 Shape Generation

We quantitatively compare our method's performance on shape generation with r-GAN [1], GCN-GAN [55], TreeGAN [48], and PointFlow [60]. We use the same experiment setup as PointFlow except for the data normalization before the evaluation. The generation results are reported in Table 3. Though our model requires a two-stage training, the training can be done within one day with a 1080 Ti GPU, while reproducing PointFlow's results requires training for at least two days on the same hardware. Despite using much less training time, our model achieves comparable performance to PointFlow, the current state-of-the-art. As demonstrated in Fig. 5, our generated shapes are also visually cleaner.

Table 3. Shape generation results. ↑ means the higher the better, ↓ means the lower the better. MMD-CD is multiplied by 10^3 and MMD-EMD is multiplied by 10^2.

Category	Model	MMD (↓)		COV (%, ↑)		1-NNA (%, ↓)	
		CD	EMD	CD	EMD	CD	EMD
Airplane	r-GAN [1]	1.657	13.287	38.52	19.75	95.80	100.00
	GCN [55]	2.623	15.535	9.38	5.93	95.16	99.12
	Tree [48]	1.466	16.662	44.69	6.91	95.06	100.00
	PF [60]	1.408	7.576	39.51	**41.98**	**83.21**	**82.22**
	Ours	**1.285**	**7.364**	**47.65**	**41.98**	85.06	83.46
	Train	1.288	7.036	45.43	45.43	72.10	69.38
Chair	r-GAN [1]	18.187	32.688	19.49	8.31	84.82	99.92
	GCN [55]	23.098	25.781	6.95	6.34	86.52	96.48
	Tree [48]	16.147	36.545	40.33	8.76	74.55	99.92
	PF [60]	15.027	19.190	40.94	44.41	67.60	72.28
	Ours	**14.818**	**18.791**	**46.37**	**46.22**	**66.16**	**59.82**
	Train	15.893	18.472	50.45	52.11	53.93	54.15

Table 4. Ablation study comparing auto-encoding performance on the Airplane dataset. CD is multiplied by 10^4 and EMD is multiplied by 10^2.

Metric	Single noise level			Prior distribution		
	0.1	0.05	0.01	Uniform	Fixed	Gaussian
CD	2.545	1.573	1009.357	0.993	0.993	0.996
EMD	4.400	8.455	36.715	2.463	2.476	2.475

4.3 Ablation Study

We conduct an ablation study quantifying the importance of learning with multiple noise levels. As detailed in Sects. 3.3–3.4, we train s_θ for multiple σ's. During inference, we sample point clouds using an annealed Langevin dynamics procedure, using the same σ's seen during training. In Table 4 we show results for models trained with a single noise level and tested without annealing. As illustrated in the table, the model does not perform as well when learning using a single noise level only. This is especially noticeable for the model trained on the smallest noise level in our model ($\sigma = 0.01$), as large regions in space are left unsupervised, resulting in significant errors.

We also demonstrate that our model is insensitive to the choice of the prior distribution. We repeat the inference procedure for our auto-encoding experiment, initializing the prior points with a Gaussian distribution or in a fixed location (using the same trained model). Results are reported on the right side of Table 4. Different prior configurations don't affect the performance, which is expected due to the stochastic nature of our solution. We further demonstrate

our model's robustness to the prior distribution in Fig. 1, where the prior depicts 3D letters.

5 Conclusions

In this work, we propose a generative model for point clouds which learns the gradient field of the logarithmic density function encoding a shape. Our method not only allows sampling of high-quality point clouds, but also enables extraction of the underlying surface of the shape. We demonstrate the effectiveness of our model on point cloud auto-encoding, generation, and super-resolution. Future work includes extending our work to model texture, appearance, and scenes.

Acknowledgment. This work was supported in part by grants from Magic Leap and Facebook AI, and the Zuckerman STEM leadership program.

References

1. Achlioptas, P., Diamanti, O., Mitliagkas, I., Guibas, L.: Learning representations and generative models for 3D point clouds. In: ICML (2018)
2. Anguelov, D., Srinivasan, P., Koller, D., Thrun, S., Rodgers, J., Davis, J.: Scape: shape completion and animation of people. In: ACM SIGGRAPH 2005 Papers, pp. 408–416 (2005)
3. Arjovsky, M., Chintala, S., Bottou, L.: Wasserstein generative adversarial networks. In: ICML (2017)
4. Atzmon, M., Lipman, Y.: SAL: sign agnostic learning of shapes from raw data. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 2565–2574 (2020)
5. Ben-Hamu, H., Maron, H., Kezurer, I., Avineri, G., Lipman, Y.: Multi-chart generative surface modeling. ACM Trans. Graph. (TOG) **37**(6), 1–15 (2018)
6. Chang, A.X., et al.: ShapeNet: an information-rich 3D model repository. Technical report. arXiv:1512.03012 [cs.GR], Stanford University — Princeton University — Toyota Technological Institute at Chicago (2015)
7. Chang, A.X., et al.: ShapeNet: an information-rich 3D model repository. arXiv preprint arXiv:1512.03012 (2015)
8. Chen, T.Q., Rubanova, Y., Bettencourt, J., Duvenaud, D.K.: Neural ordinary differential equations. In: NeurIPS (2018)
9. Chen, T., Fox, E., Guestrin, C.: Stochastic gradient Hamiltonian Monte Carlo. In: International Conference on Machine Learning, pp. 1683–1691 (2014)
10. Chen, Z., Zhang, H.: Learning implicit fields for generative shape modeling. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 5939–5948 (2019)
11. Deprelle, T., Groueix, T., Fisher, M., Kim, V., Russell, B., Aubry, M.: Learning elementary structures for 3D shape generation and matching. In: Advances in Neural Information Processing Systems, pp. 7433–7443 (2019)
12. Dinh, L., Krueger, D., Bengio, Y.: Nice: non-linear independent components estimation. CoRR abs/1410.8516 (2014)
13. Du, Y., Mordatch, I.: Implicit generation and generalization in energy-based models. arXiv preprint arXiv:1903.08689 (2019)

14. Fan, A., Fisher III, J.W., Kane, J., Willsky, A.S.: MCMC curve sampling and geometric conditional simulation. In: Computational Imaging VI, vol. 6814, p. 681407. International Society for Optics and Photonics (2008)
15. Fan, A.C., Fisher, J.W., Wells, W.M., Levitt, J.J., Willsky, A.S.: MCMC curve sampling for image segmentation. In: Ayache, N., Ourselin, S., Maeder, A. (eds.) MICCAI 2007. LNCS, vol. 4792, pp. 477–485. Springer, Heidelberg (2007). https://doi.org/10.1007/978-3-540-75759-7_58
16. Fan, H., Su, H., Guibas, L.J.: A point set generation network for 3D object reconstruction from a single image. In: CVPR (2017)
17. Gadelha, M., Wang, R., Maji, S.: Multiresolution tree networks for 3D point cloud processing. In: Ferrari, V., Hebert, M., Sminchisescu, C., Weiss, Y. (eds.) ECCV 2018. LNCS, vol. 11211, pp. 105–122. Springer, Cham (2018). https://doi.org/10.1007/978-3-030-01234-2_7
18. Gao, L., Yang, J., Wu, T., Yuan, Y.J., Fu, H., Lai, Y.K., Zhang, H.: SDM-NET: deep generative network for structured deformable mesh. ACM Trans. Graph. (TOG) **38**(6), 1–15 (2019)
19. Genova, K., Cole, F., Sud, A., Sarna, A., Funkhouser, T.: Deep structured implicit functions. arXiv preprint arXiv:1912.06126 (2019)
20. Genova, K., Cole, F., Vlasic, D., Sarna, A., Freeman, W.T., Funkhouser, T.: Learning shape templates with structured implicit functions. In: Proceedings of the IEEE International Conference on Computer Vision, pp. 7154–7164 (2019)
21. Girdhar, R., Fouhey, D.F., Rodriguez, M., Gupta, A.: Learning a predictable and generative vector representation for objects. In: Leibe, B., Matas, J., Sebe, N., Welling, M. (eds.) ECCV 2016. LNCS, vol. 9910, pp. 484–499. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-46466-4_29
22. Grathwohl, W., Chen, R.T.Q., Bettencourt, J., Sutskever, I., Duvenaud, D.: FFJORD: free-form continuous dynamics for scalable reversible generative models. In: ICLR (2019)
23. Groueix, T., Fisher, M., Kim, V.G., Russell, B., Aubry, M.: AtlasNet: a Papier-Mâché approach to learning 3D surface generation. In: CVPR (2018)
24. Gulrajani, I., Ahmed, F., Arjovsky, M., Dumoulin, V., Courville, A.C.: Improved training of Wasserstein GANs. In: NeurIPS (2017)
25. Hanocka, R., Hertz, A., Fish, N., Giryes, R., Fleishman, S., Cohen-Or, D.: MeshCNN: a network with an edge. ACM Trans. Graph. (TOG) **38**(4), 1–12 (2019)
26. Hao, Z., Averbuch-Elor, H., Snavely, N., Belongie, S.: DualSDF: semantic shape manipulation using a two-level representation. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 7631–7641 (2020)
27. Hyvärinen, A.: Estimation of non-normalized statistical models by score matching. J. Mach. Learn. Res. **6**, 695–709 (2005)
28. Kingma, D.P., Dhariwal, P.: Glow: generative flow with invertible 1x1 convolutions. In: NeurIPS (2018)
29. LeCun, Y., Chopra, S., Hadsell, R., Ranzato, M., Huang, F.: A tutorial on energy-based learning. Predict. Struct. Data **1** (2006)
30. LeCun, Y., Cortes, C., Burges, C.: MNIST handwritten digit database (2010)
31. Li, C.L., Zaheer, M., Zhang, Y., Poczos, B., Salakhutdinov, R.: Point cloud GAN. arXiv preprint arXiv:1810.05795 (2018)
32. Li, J., Xu, K., Chaudhuri, S., Yumer, E., Zhang, H., Guibas, L.: GRASS: generative recursive autoencoders for shape structures. ACM Trans. Graph. (TOG) **36**(4), 1–14 (2017)

33. Litany, O., Bronstein, A., Bronstein, M., Makadia, A.: Deformable shape completion with graph convolutional autoencoders. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 1886–1895 (2018)
34. Liu, Q., Lee, J., Jordan, M.: A kernelized stein discrepancy for goodness-of-fit tests. In: International Conference on Machine Learning, pp. 276–284 (2016)
35. Lorensen, W.E., Cline, H.E.: Marching cubes: a high resolution 3D surface construction algorithm. ACM SIGGRAPH Comput. Graph. **21**(4), 163–169 (1987)
36. Mescheder, L., Oechsle, M., Niemeyer, M., Nowozin, S., Geiger, A.: Occupancy networks: learning 3D reconstruction in function space. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 4460–4470 (2019)
37. Michalkiewicz, M., Pontes, J.K., Jack, D., Baktashmotlagh, M., Eriksson, A.: Deep level sets: implicit surface representations for 3D shape inference. arXiv preprint arXiv:1901.06802 (2019)
38. Mo, K., Guerrero, P., Yi, L., Su, H., Wonka, P., Mitra, N., Guibas, L.J.: StructureNet: hierarchical graph networks for 3D shape generation. arXiv preprint arXiv:1908.00575 (2019)
39. 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)
40. Van den Oord, A., Kalchbrenner, N., Espeholt, L., Vinyals, O., Graves, A., et al.: Conditional image generation with pixelcnn decoders. In: NeurIPS (2016)
41. Oord, A.v.d., et al.: WaveNet: a generative model for raw audio. arXiv preprint arXiv:1609.03499 (2016)
42. Oord, A.v.d., Kalchbrenner, N., Kavukcuoglu, K.: Pixel recurrent neural networks. In: ICML (2016)
43. Papamakarios, G., Nalisnick, E., Rezende, D.J., Mohamed, S., Lakshminarayanan, B.: Normalizing flows for probabilistic modeling and inference. arXiv preprint arXiv:1912.02762 (2019)
44. Park, J.J., Florence, P., Straub, J., Newcombe, R., Lovegrove, S.: DeepSDF: learning continuous signed distance functions for shape representation. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 165–174 (2019)
45. Pons-Moll, G., Romero, J., Mahmood, N., Black, M.J.: Dyna: a model of dynamic human shape in motion. ACM Trans. Graph. (TOG) **34**(4), 1–14 (2015)
46. Rezende, D.J., Mohamed, S.: Variational inference with normalizing flows. In: ICML (2015)
47. Roth, S.D.: Ray casting for modeling solids. Comput. Graph. Image Process. **18**(2), 109–144 (1982)
48. Shu, D.W., Park, S.W., Kwon, J.: 3D point cloud generative adversarial network based on tree structured graph convolutions. In: Proceedings of the IEEE International Conference on Computer Vision, pp. 3859–3868 (2019)
49. Smirnov, D., Fisher, M., Kim, V.G., Zhang, R., Solomon, J.: Deep parametric shape predictions using distance fields. arXiv preprint arXiv:1904.08921 (2019)
50. Song, Y., Ermon, S.: Generative modeling by estimating gradients of the data distribution. In: Advances in Neural Information Processing Systems, pp. 11895–11907 (2019)
51. Song, Y., Garg, S., Shi, J., Ermon, S.: Sliced score matching: a scalable approach to density and score estimation. arXiv preprint arXiv:1905.07088 (2019)

52. Sun, Y., Wang, Y., Liu, Z., Siegel, J.E., Sarma, S.E.: PointGrow: autoregressively learned point cloud generation with self-attention. arXiv preprint arXiv:1810.05591 (2018)
53. Tan, Q., Gao, L., Lai, Y.K., Xia, S.: Variational autoencoders for deforming 3D mesh models. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 5841–5850 (2018)
54. Tulsiani, S., Su, H., Guibas, L.J., Efros, A.A., Malik, J.: Learning shape abstractions by assembling volumetric primitives. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 2635–2643 (2017)
55. Valsesia, D., Fracastoro, G., Magli, E.: Learning localized generative models for 3d point clouds via graph convolution (2018)
56. Vincent, P.: A connection between score matching and denoising autoencoders. Neural Comput. **23**(7), 1661–1674 (2011)
57. Welling, M., Teh, Y.W.: Bayesian learning via stochastic gradient Langevin dynamics. In: Proceedings of the 28th International Conference on Machine Learning (ICML-11), pp. 681–688 (2011)
58. Wenliang, L., Sutherland, D., Strathmann, H., Gretton, A.: Learning deep kernels for exponential family densities. arXiv preprint arXiv:1811.08357 (2018)
59. 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)
60. 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)
61. Yang, Y., Feng, C., Shen, Y., Tian, D.: FoldingNet: point cloud auto-encoder via deep grid deformation. In: CVPR (2018)
62. Yifan, W., Wu, S., Huang, H., Cohen-Or, D., Sorkine-Hornung, O.: Patch-based progressive 3D point set upsampling. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 5958–5967 (2019)
63. Zamorski, M., Zieba, M., Nowak, R., Stokowiec, W., Trzcinski, T.: Adversarial autoencoders for generating 3D point clouds. arXiv preprint arXiv:1811.07605 2 (2018)
64. Zamorski, M., Zieba, M., Nowak, R., Stokowiec, W., Trzciński, T.: Adversarial autoencoders for generating 3D point clouds. arXiv preprint arXiv:1811.07605 (2018)