PointFlow: 3D Point Cloud Generation with Continuous Normalizing Flows

Guandao Yang[1,2*], Xun Huang[1,2*], Zekun Hao[1,2], Ming-Yu Liu[3], Serge Belongie[1,2], Bharath Hariharan[1]
[1]Cornell University [2]Cornell Tech [3]NVIDIA

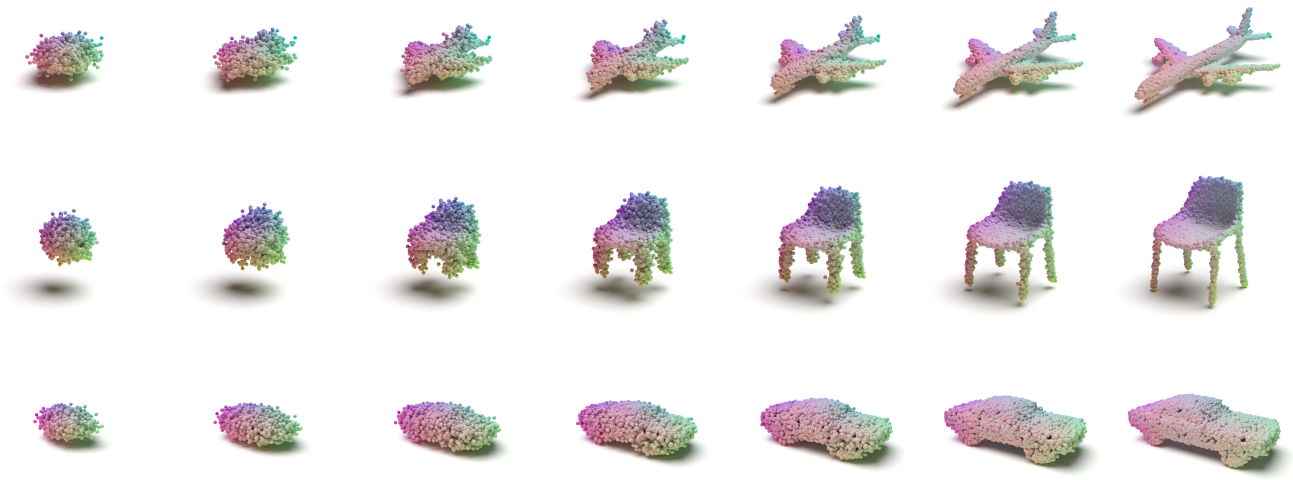

Figure 1: Our model transforms points sampled from a simple prior to realistic point clouds through continuous normalizing flows. The videos of the transformations can be viewed on our project website: https://www.guandaoyang.com/PointFlow/.

Abstract

As 3D point clouds become the representation of choice for multiple vision and graphics applications, the ability to synthesize or reconstruct high-resolution, high-fidelity point clouds becomes crucial. Despite the recent success of deep learning models in discriminative tasks of point clouds, generating point clouds remains challenging. This paper proposes a principled probabilistic framework to generate 3D point clouds by modeling them as a distribution of distributions. Specifically, we learn a two-level hierarchy of distributions where the first level is the distribution of shapes and the second level is the distribution of points given a shape. This formulation allows us to both sample shapes and sample an arbitrary number of points from a shape. Our generative model, named PointFlow, learns each level of the distribution with a continuous normalizing flow. The invertibility of normalizing flows enables the computation of the likelihood during training and allows us to train our model in the variational inference framework. Empirically, we demonstrate that PointFlow achieves state-of-the-art performance in point cloud generation. We additionally show that our model can faithfully reconstruct point clouds and learn useful representations in an unsupervised manner. The code is available at https://github.com/stevenygd/PointFlow.

1. Introduction

Point clouds are becoming popular as a 3D representation because they can capture a much higher resolution than voxel grids and are a stepping stone to more sophisticated representations such as meshes. Learning a generative model of point clouds could benefit a wide range of point cloud synthesis tasks such as reconstruction and super-resolution, by providing a better *prior* of point clouds. However, a major roadblock in generating point clouds is the complexity of the space of point clouds. A cloud of points corresponding to a chair is best thought of as sam-

*Equal contribution.

ples from a distribution that corresponds to the surface of the chair, and the chair itself is best thought of as a sample from a distribution of chair shapes. As a result, in order to generate a chair according to this formulation, we need to characterize a *distribution of distributions*, which is under-explored by existing generative models.

In this paper, we propose PointFlow, a principled generative model for 3D point clouds that learns a distribution of distributions: the former being the distribution of shapes and the latter being the distribution of points given a shape. Our key insight is that instead of directly parametrizing the distribution of points in a shape, we model this distribution as an *invertible parameterized transformation* of 3D points from a prior distribution (e.g., a 3D Gaussian). Intuitively, under this model, generating points for a given shape involves sampling points from a generic Gaussian prior, and then *moving* them according to this parameterized transformation to their new location in the target shape, as illustrated in Figure 1. In this formulation, a given shape is then simply the variable that parametrizes such transformation, and a category is simply a distribution of this variable. Interestingly, we find that representing this distribution too as a transformation of a prior distribution leads to a more expressive model of shapes. In particular, we use the recently proposed continuous normalizing flow framework to model both kinds of transformations [38, 5, 16].

This parameterization confers several advantages. The invertibility of these transformations allows us to not just sample but also estimate probability densities. The ability to estimate probability densities in turn allows us to train these models in a principled manner using the variational inference framework [26], where we maximize a variational lower bound on the log-likelihood of a training set of point clouds. This probabilistic framework for training further lets us avoid the complexities of training GANs or hand-crafting good distance metrics for measuring the difference between two sets of points. Experiments show that PointFlow outperforms previous state-of-the-art generative models of point clouds, and achieves compelling results in point cloud reconstruction and unsupervised feature learning.

2. Related work

Deep learning for point clouds. Deep learning has been introduced to improve performance in various point cloud discriminative tasks including classification [36, 37, 49, 53], segmentation [36, 41], and critical points sampling [10]. Recently, substantial progress has been made in point cloud synthesis tasks such as auto-encoding [1, 49, 17], single-view 3D reconstruction [12, 21, 28, 30, 13], stereo reconstruction [43], and point cloud completion [52, 51]. Many point cloud synthesis works convert a point distribution to a $N \times 3$ matrix by sampling N (N is pre-defined) points from the distribution so that existing generative models are readily applicable. For example, Gadelha *et al*. [13] apply variational auto-encoders (VAEs) [26] and Zamorski *et al*. [54] apply adversarial auto-encoders (AAEs) [32] to point cloud generation. Achlioptas *et al*. [1] explore generative adversarial networks (GANs) [15, 2, 19] for point clouds in both raw data space and latent space of a pre-trained auto-encoder. In the above methods, the auto-encoders are trained with heuristic loss functions that measure the distance between two point sets, such as Chamfer distance (CD) or earth mover's distance (EMD). Sun *et al*. [42] apply auto-regressive models [45] with a discrete point distribution to generate one point at a time, also using a fixed number of points per shape.

However, treating a point cloud as a fixed-dimensional matrix has several drawbacks. First, the model is restricted to generate a fixed number of points. Getting more points for a particular shape requires separate up-sampling models such as [52, 51, 50]. Second, it ignores the permutation invariance property of point sets, which might lead to suboptimal parameter efficiency. Heuristic set distances are also far from ideal objectives from a generative modeling perspective since they make the original probabilistic interpretation of VAE/AAE no longer applicable when used as the reconstruction objective. In addition, exact EMD is slow to compute while approximations could lead to biased or noisy gradients. CD has been shown to incorrectly favor point clouds that are overly concentrated in the mode of the marginal point distribution [1].

Some recent works introduce sophisticated decoders consisting of a cascade [49] or a mixture [17] of smaller decoders to map one (or a mixture of) 2-D uniform distribution(s) to the target point distribution, overcoming the shortcomings of using a fixed number of points. However, they still rely on heuristic set distances that lack a probabilistic guarantee. Also, their methods only learn the distribution of points for each shape, but not the distribution of shapes. Li *et al*. [29] propose a "sandwiching" reconstruction objective that combines a variant of WGAN [2] loss with EMD. They also train another GAN in the latent space to learn shape distribution, similar to Achlioptas *et al*. [1]. In contrast, our method is simply trained end-to-end by maximizing a variational lower bound on the log-likelihood, does not require multi-stage training, and does not have any instability issues common for GAN-based methods.

Generative models. There are several popular frameworks of deep generative models, including generative adversarial networks [15, 2, 22], variational auto-encoders [26, 39], auto-regressive models [33, 45], and flow-based models [8, 38, 9, 24]. In particular, flow-based models and auto-regressive models can both perform exact likelihood evaluation, while flow-based models are much more efficient to sample from. Flow-based models have been successfully applied to a variety of generation tasks such as

image generation [24, 9, 8], video generation [27], and voice synthesis [35]. Also, there has been recent work that combines flows with other generative models, such as GANs [18, 7], auto-regressive models [20, 34, 25], and VAEs [25, 44, 6, 38, 44, 5, 16].

Most existing deep generative models aim at learning the distribution of fixed-dimensional variables. Learning the *distribution of distributions*, where the data consists of a *set of sets*, is still underexplored. Edwards and Storkey [11] propose a hierarchical VAE named Neural Statistician that consumes a set of sets. They are mostly interested in the few-shot case where each set only has a few samples. Also, they are focused on classifying sets or generating new samples from a given set. While our method is also applicable to these tasks, our focus is on learning the distribution of sets and generating new sets (point clouds in our case). In addition, our model employs a tighter lower bound on the log-likelihood, thanks to the use of normalizing flow in modeling both the reconstruction likelihood and the prior.

3. Overview

Consider a set of shapes $\mathcal{X} = \{X_i\}_{i=1}^N$ from a particular class of object, where each shape is represented as a set of 3D points $X_i = \{x_j^i\}_{j=1}^{M_i}$. As discussed in Section 1, each point $x_j^i \in \mathbb{R}^3$ is best thought of as being sampled from a point distribution $Q^i(x)$, usually a uniform distribution over the surface of an object X_i. Each shape X_i is itself a sample from a distribution over shapes $Q(X)$ that captures what shapes in this category look like.

Our goal is to learn the distribution of shapes, each shape itself being a distribution of points. In other words, our generative model should be able to both sample shapes and sample an arbitrary number of points from a shape.

We propose to use continuous normalizing flows to model the distribution of points given a shape. A continuous normalizing flow can be thought of as a vector field in the 3-D Euclidean space, which induces a distribution of points through transforming a generic prior distribution (*e.g.*, a standard Gaussian). To sample points from the induced distribution, we simply sample points from the prior and move them according to the vector field. Moreover, the continuous normalizing flow is invertible, which means we can move data points back to the prior distribution to compute the exact likelihood. This model is highly intuitive and interpretable, allowing a close inspection of the generative process as shown in Figure 1.

We parametrize each continuous normalizing flow with a latent variable that represents the shape. As a result, modeling the distribution of shapes can be reduced to modeling the distribution of the latent variable. Interestingly, we find continuous normalizing flow also effective in modeling the latent distribution. Our full generative model thus consists of two levels of continuous normalizing flows, one modeling the shape distribution by modeling the distribution of the latent variable, and the other modeling the point distribution given a shape.

In order to optimize the generative model, we construct a variational lower bound on the log-likelihood by introducing an inference network that infers a latent variable distribution from a point cloud. Here, we benefit from the fact that the invertibility of the continuous normalizing flow enables likelihood computation. This allows us to train our model end-to-end in a stable manner, unlike previous work based on GANs that requires two-stage training [1, 29]. As a side benefit, we find the inference network learns a useful representation of point clouds in an unsupervised manner.

In Section 4 we introduce some background on continuous normalizing flows and variational auto-encoders. We then describe our model and training in detail in Section 5.

4. Background

4.1. Continuous normalizing flow

A normalizing flow [38] is a series of invertible mappings that transform an initial known distribution to a more complicated one. Formally, let $f_1, \ldots, f_n$ denote a series of invertible transformations we want to apply to a latent variable y with a distribution $P(y)$. $x = f_n \circ f_{n-1} \circ \cdots \circ f_1(y)$ is the output variable. Then the probability density of the output variable is given by the change of variables formula:

$$\log P(x) = \log P(y) - \sum_{k=1}^{n} \log \left| \det \frac{\partial f_k}{\partial y_{k-1}} \right|, \qquad (1)$$

where y can be computed from x using the inverse flow: $y = f_1^{-1} \circ \cdots \circ f_n^{-1}(x)$. In practice, $f_1, \ldots, f_n$ are usually instantiated as neural networks with an architecture that makes the determinant of the Jacobian $\left| \det \frac{\partial f_k}{\partial y_{k-1}} \right|$ easy to compute. The normalizing flow has been generalized from a discrete sequence to a continuous transformation [16, 5] by defining the transformation f using a continuous-time dynamic $\frac{\partial y(t)}{\partial t} = f(y(t), t)$, where f is a neural network that has an unrestricted architecture. The continuous normalizing flow (CNF) model for $P(x)$ with a prior distribution $P(y)$ at the start time can be written as:

$$x = y(t_0) + \int_{t_0}^{t_1} f(y(t), t) dt, \quad y(t_0) \sim P(y)$$

$$\log P(x) = \log P(y(t_0)) - \int_{t_0}^{t_1} \operatorname{Tr}\left(\frac{\partial f}{\partial y(t)} \right) dt \qquad (2)$$

and $y(t_0)$ can be computed using the inverse flow $y(t_0) = x + \int_{t_1}^{t_0} f(y(t), t) dt$. A black-box ordinary differential equation (ODE) solver can been applied to estimate the outputs and the input gradients of a continuous normalizing flow [16, 5].

4.2. Variational auto-encoder

Suppose we have a random variable X that we are building generative models for. The variational auto-encoder (VAE) is a framework that allows one to learn $P(X)$ from a dataset of observations of X [26, 39]. The VAE models the data distribution via a latent variable z with a prior distribution $P_\psi(z)$, and a decoder $P_\theta(X|z)$ which captures the (hopefully simpler) distribution of X given z. During training, it additionally learns an inference model (or encoder) $Q_\phi(z|X)$. The encoder and decoder are jointly trained to maximize a lower bound on the log-likelihood of the observations

$$\begin{aligned}\log P_\theta(X) &\geq \log P_\theta(X) - D_{KL}(Q_\phi(z|X)||P_\theta(z|X)) \\ &= \mathbb{E}_{Q_\phi(z|x)}\left[\log P_\theta(X|z)\right] - D_{KL}\left(Q_\phi(z|X)||P_\psi(z)\right) \\ &\triangleq \mathcal{L}(X;\phi,\psi,\theta)\,, \end{aligned} \tag{3}$$

which is also called the evidence lower bound (ELBO). One can interpret ELBO as the sum of the negative reconstruction error (the first term) and a latent space regularizer (the second term). In practice, $Q_\phi(z|X)$ is usually modeled as a diagonal Gaussian $\mathcal{N}(z|\mu_\phi(X), \sigma_\phi(X))$ whose mean and standard deviation are predicted by a neural network with parameters ϕ. To efficiently optimize the ELBO, sampling from $Q_\phi(z|X)$ is done by reparametrizing z as $z = \mu_\phi(X) + \sigma_\phi(X) \cdot \epsilon$, where $\epsilon \sim \mathcal{N}(0, I)$.

5. Model

We now have the paraphernalia needed to define our generative model of point clouds. Using the terminology of the VAE, we need three modules: the encoder $Q_\phi(z|X)$ that encodes a point cloud into a shape representation z, a prior $P_\psi(z)$ over shape representations, and a decoder $P_\theta(X|z)$ that models the distribution of points given the shape representation. We use a simple permutation-invariant encoder to predict $Q_\phi(z|X)$, following the architecture in Achlioptas *et al*. [1]. We use continuous normalizing flows for both the prior $P_\psi(z)$ and the generator $P_\theta(X|z)$, which are described below.

5.1. Flow-based point generation from shape representations

We first decompose the reconstruction log-likelihood of a point set into the sum of log-likelihood of each point

$$\log P_\theta(X|z) = \sum_{x \in X} \log P_\theta(x|z)\,. \tag{4}$$

We propose to model $P_\theta(x|z)$ using a conditional extension of CNF. Specifically, a point x in the point set X is the result of transforming some point $y(t_0)$ in the prior distribution $P(y) = \mathcal{N}(0, I)$ using a CNF conditioned on z:

$$x = G_\theta(y(t_0); z) \triangleq y(t_0) + \int_{t_0}^{t_1} g_\theta(y(t), t, z)dt, y(t_0) \sim P(y)\,,$$

where g_θ defines the continuous-time dynamics of the flow G_θ conditioned on z. Note that the inverse of G_θ is given by $G_\theta^{-1}(x; z) = x + \int_{t_1}^{t_0} g_\theta(y(t), t, z)dt$ with $y(t_1) = x$. The reconstruction likelihood of follows equation (2):

$$\log P_\theta(x|z) = \log P(G_\theta^{-1}(x; z)) - \int_{t_0}^{t_1} \mathrm{Tr}\left(\frac{\partial g_\theta}{\partial y(t)}\right) dt\,. \tag{5}$$

Note that $\log P(G_\theta^{-1}(x; z))$ can be computed in closed form with the Gaussian prior.

5.2. Flow-based prior over shapes

Although it is possible to use a simple Gaussian prior over shape representations, it has been shown that a restricted prior tends to limit the performance of VAEs [6]. To alleviate this problem, we use another CNF to parametrize a learnable prior. Formally, we rewrite the KL divergence term in Equation 3 as

$$D_{KL}(Q_\phi(z|x)||P_\psi(z)) = -\mathbb{E}_{Q_\phi(z|x)}\left[\log P_\psi(z)\right] - H[Q_\phi(z|X)]\,, \tag{6}$$

where H is the entropy and $P_\psi(z)$ is the prior distribution with learnable parameters ψ, obtained by transforming a simple Gaussian $P(w) = \mathcal{N}(0, I)$ with a CNF:

$$z = F_\psi(w(t_0)) \triangleq w(t_0) + \int_{t_0}^{t_1} f_\psi(w(t), t)dt, w(t_0) \sim P(w)\,,$$

where f_ψ defines the continuous-time dynamics of the flow F_ψ. Similarly as described above, the inverse of F_ψ is given by $F_\psi^{-1}(z) = z + \int_{t_1}^{t_0} f_\psi(w(t), t)dt$ with $w(t_1) = z$. The log probability of the prior distribution can be computed by:

$$\log P_\psi(z) = \log P\left(F_\psi^{-1}(z)\right) - \int_{t_0}^{t_1} \mathrm{Tr}\left(\frac{\partial f_\psi}{\partial w(t)}\right) dt\,. \tag{7}$$

5.3. Final training objective

Plugging Equation 4, 5, 6, 7 into Equation 3, the ELBO of a point set X can be finally written as

$$\begin{aligned}\mathcal{L}(X;\phi,\psi,\theta) &= \mathbb{E}_{Q_\phi(z|x)}\left[\log P_\psi(z) + \log P_\theta(X|z)\right] + H[Q_\phi(z|X)] \\ &= \mathbb{E}_{Q_\phi(z|X)}[\log P\left(F_\psi^{-1}(z)\right) - \int_{t_0}^{t_1} \mathrm{Tr}\left(\frac{\partial f_\psi}{\partial w(t)}\right) dt \\ &+ \sum_{x \in X}(\log P(G_\theta^{-1}(x; z)) - \int_{t_0}^{t_1} \mathrm{Tr}\left(\frac{\partial g_\theta}{\partial y(t)}\right) dt)] \\ &+ H[Q_\phi(z|X)]\,. \end{aligned} \tag{8}$$

Our model is trained end-to-end by maximizing the ELBO of all point sets in the dataset

$$\phi^*, \psi^*, \theta^* = \arg\max_{\phi,\psi,\theta} \sum_{X \in \mathcal{X}} \mathcal{L}(X;\phi,\psi,\theta). \tag{9}$$

We can interpret this objective as the sum of three parts:

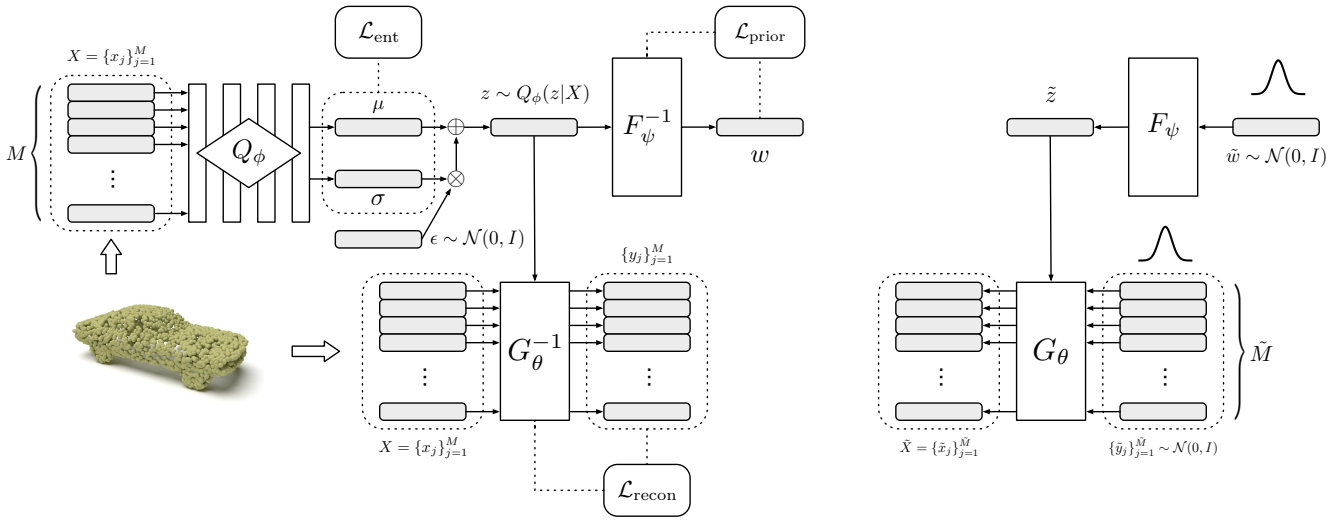

Figure 2: Model architecture. (a) At training time, the encoder Q_ϕ infers a posterior over shape representations given an input point cloud X, and samples a shape representation z from it. We then compute the probability of z in the prior distribution ($\mathcal{L}_{\text{prior}}$) through a inverse CNF F_ψ^{-1}, and compute the reconstruction likelihood of X ($\mathcal{L}_{\text{recon}}$) through another inverse CNF G_θ^{-1} conditioned on z. The model is trained end-to-end to maximize the evidence lower bound (ELBO), which is the sum of $\mathcal{L}_{\text{prior}}$, $\mathcal{L}_{\text{recon}}$, and $\mathcal{L}_{\text{ent}}$ (the entropy of the posterior $Q_\phi(z|X)$). (b) At test time, we sample a shape representation $\tilde{z}$ by sampling $\tilde{w}$ from a Gaussian prior and transforming it with F_ψ. To sample points from the shape represented by $\tilde{z}$, we first sample points from the 3-D Gaussian prior and then move them according to the CNF parameterized by $\tilde{z}$.

1. **Prior:** $\mathcal{L}_{\text{prior}}(X; \psi, \phi) \triangleq \mathbb{E}_{Q_\phi(z|x)}[\log P_\psi(z)]$ encourages the encoded shape representation to have a high probability under the prior, which is modeled by a CNF as described in Section 5.2. We use the reparameterization trick [26] to enable a differentiable Monte Carlo estimate of the expectation:

$$\mathbb{E}_{Q_\phi(z|x)}[\log P_\psi(z)] \approx \frac{1}{L} \sum_{l=1}^{L} \log P_\psi(\mu + \epsilon_l \odot \sigma) \,,$$

   where μ and σ are mean and standard deviation of the isotropic Gaussian posterior $Q_\phi(z|x)$ and L is simply set to 1. ϵ_i is sampled from the standard Gaussian distribution $\mathcal{N}(0, I)$.

2. **Reconstruction likelihood:** $\mathcal{L}_{\text{recon}}(X; \theta, \phi) \triangleq \mathbb{E}_{Q_\phi(z|x)}[\log P_\theta(X|z)]$ is the reconstruction log-likelihood of the input point set, computed as described in Section 5.1. The expectation is also estimated using Monte Carlo sampling.

3. **Posterior Entropy:** $\mathcal{L}_{\text{ent}}(X; \phi) \triangleq H[Q_\phi(z|X)]$ is the entropy of the approximated posterior:

$$H[Q_\phi(z|X)] = \tfrac{d}{2}(1 + \ln(2\pi)) + \textstyle\sum_{i=1}^{d} \ln \sigma_i \,.$$

All the training details (*e.g.*, hyper-parameters, model architectures) are included in the supplementary materials.

5.4. Sampling

To sample a shape representation, we first draw $\tilde{w} \sim \mathcal{N}(0, I)$ then pass it through F_ψ to get $\tilde{z} = F_\psi(\tilde{w})$. To generate a point given a shape representation $\tilde{z}$, we first sample a point $\tilde{y} \in \mathbb{R}^3$ from $\mathcal{N}(0, I)$, then pass $\tilde{y}$ through G_θ conditioned on $\tilde{z}$ to produce a point on the shape : $\tilde{x} = G_\theta(\tilde{w}; z)$. To sample a point cloud with size $\tilde{M}$, we simply repeat it for $\tilde{M}$ times. Combining these two steps allows us to sample a point cloud with $\tilde{M}$ points from our model:

$$\tilde{X} = \{G_\theta(\tilde{y}_j; F_\psi(\tilde{w}))\}_{1 \le j \le \tilde{M}}, \tilde{w} \sim \mathcal{N}(0, I), \forall j, \tilde{y}_j \sim \mathcal{N}(0, I) \,.$$

6. Experiments

In this section, we first introduce existing metrics for evaluating point cloud generation, discuss their limitations, and introduce a new metric that overcomes these limitations. We then compare the proposed method with previous state-of-the-art generative models of point clouds, using both previous metrics and the proposed one. We additionally evaluate the reconstruction and representation learning ability of the auto-encoder part of our model.

6.1. Evaluation metrics

Following prior work, we use Chamfer distance (CD) and earth mover's distance (EMD) to measure the similarity

between point clouds. Formally, they are defined as follows:

$$\text{CD}(X, Y) = \sum_{x \in X} \min_{y \in Y} \|x - y\|_2^2 + \sum_{y \in Y} \min_{x \in X} \|x - y\|_2^2,$$

$$\text{EMD}(X, Y) = \min_{\phi: X \to Y} \sum_{x \in X} \|x - \phi(x)\|_2,$$

where X and Y are two point clouds with the same number of points and ϕ is a bijection between them. Note that most previous methods use either CD or EMD in their training objectives, which tend to be favored if evaluated under the same metric. Our method, however, does not use CD or EMD during training.

Let S_g be the set of generated point clouds and S_r be the set of reference point clouds with $|S_r| = |S_g|$. To evaluate generative models, we first consider the three metrics introduced by Achlioptas *et al*. [1]:

- **Jensen-Shannon Divergence (JSD)** are computed between the marginal point distributions:

  $$\text{JSD}(P_g, P_r) = \frac{1}{2} D_{KL}(P_r || M) + \frac{1}{2} D_{KL}(P_g || M)\,,$$

  where $M = \frac{1}{2}(P_r + P_g)$. P_r and P_g are marginal distributions of points in the reference and generated sets, approximated by discretizing the space into 28^3 voxels and assigning each point to one of them. However, it only considers the marginal point distributions but not the distribution of individual shapes. A model that always outputs the "average shape" can obtain a perfect JSD score without learning any meaningful shape distributions.

- **Coverage (COV)** measures the fraction of point clouds in the reference set that are matched to at least one point cloud in the generated set. For each point cloud in the generated set, its nearest neighbor in the reference set is marked as a match:

  $$\text{COV}(S_g, S_r) = \frac{|\{\arg\min_{Y \in S_r} D(X, Y) | X \in S_g\}|}{|S_r|},$$

  where $D(\cdot, \cdot)$ can be either CD or EMD. While coverage is able to detect mode collapse, it does not evaluate the quality of generated point clouds. In fact, it is possible to achieve a perfect coverage score even if the distances between generated and reference point clouds are arbitrarily large.

- **Minimum matching distance (MMD)** is proposed to complement coverage as a metric that measures quality. For each point cloud in the reference set, the distance to its nearest neighbor in the generated set is computed and averaged:

  $$\text{MMD}(S_g, S_r) = \frac{1}{|S_r|} \sum_{Y \in S_r} \min_{X \in S_g} D(X, Y)\,,$$

  where $D(\cdot, \cdot)$ can be either CD or EMD. However, MMD is actually very insensitive to low-quality point clouds in S_g, since they are unlikely to be matched to real point clouds in S_r. In the extreme case, one can imagine that S_g consists of mostly very low-quality point clouds with one additional point cloud in each mode of S_r, yet has a reasonably good MMD score.

As discussed above, all existing metrics have their limitations. As will be shown later, we also empirically find all these metrics sometimes give generated point clouds even better scores than real point clouds, further casting doubt on whether they can ensure a fair model comparison. We therefore introduce another metric that we believe is better suited for evaluating generative models of point clouds:

- **1-nearest neighbor accuracy (1-NNA)** is proposed by Lopez-Paz and Oquab [31] for two-sample tests, assessing whether two distributions are identical. It has also been explored as a metric for evaluating GANs [48]. Let $S_{-X} = S_r \cup S_g - \{X\}$ and N_X be the nearest neighbor of X in S_{-X}. 1-NNA is the leave-one-out accuracy of the 1-NN classifier:

  $$\text{1-NNA}(S_g, S_r) = \frac{\sum_{X \in S_g} \mathbb{I}[N_X \in S_g] + \sum_{Y \in S_r} \mathbb{I}[N_Y \in S_r]}{|S_g| + |S_r|},$$

  where $\mathbb{I}[\cdot]$ is the indicator function. For each sample, the 1-NN classifier classifies it as coming from S_r or S_g according to the label of its nearest sample. If S_g and S_r are sampled from the same distribution, the accuracy of such a classifier should converge to 50% given a sufficient number of samples. The closer the accuracy is to 50%, the more similar S_g and S_r are, and therefore the better the model is at learning the target distribution. In our setting, the nearest neighbor can be computed using either CD or EMD. Unlike JSD, 1-NNA considers the similarity between shape distributions rather than between marginal point distributions. Unlike COV and MMD, 1-NNA directly measures distributional similarity and takes both diversity and quality into account.

6.2. Generation

We compare our method with three existing generative models for point clouds: raw-GAN [1], latent-GAN [1], and PC-GAN [29], using their official implementations that are either publicly available or obtained by contacting the authors. We train each model using point clouds from one of the three categories in the ShapeNet [3] dataset: *airplane*, *chair*, and *car*. The point clouds are obtained by sampling points uniformly from the mesh surface. All points in each category are normalized to have zero-mean per axis

Table 1: Generation results. ↑: the higher the better, ↓: the lower the better. The best scores are highlighted in bold. Scores of the real shapes that are worse than some of the generated shapes are marked in gray. MMD-CD scores are multiplied by 10^3; MMD-EMD scores are multiplied by 10^2; JSDs are multiplied by 10^2.

Category	Model	# Parameters (M)		JSD (↓)	MMD (↓)		COV (%, ↑)		1-NNA (%, ↓)	
		Full	Gen		CD	EMD	CD	EMD	CD	EMD
Airplane	r-GAN	7.22	6.91	7.44	0.261	5.47	42.72	18.02	93.58	99.51
	l-GAN (CD)	1.97	1.71	4.62	0.239	4.27	43.21	21.23	86.30	97.28
	l-GAN (EMD)	1.97	1.71	**3.61**	0.269	3.29	**47.90**	**50.62**	87.65	85.68
	PC-GAN	9.14	1.52	4.63	0.287	3.57	36.46	40.94	94.35	92.32
	PointFlow (ours)	**1.61**	**1.06**	4.92	**0.217**	**3.24**	46.91	48.40	**75.68**	**75.06**
	Training set	-	-	6.61	0.226	3.08	42.72	49.14	70.62	67.53
Chair	r-GAN	7.22	6.91	11.5	2.57	12.8	33.99	9.97	71.75	99.47
	l-GAN (CD)	1.97	1.71	4.59	2.46	8.91	41.39	25.68	64.43	85.27
	l-GAN (EMD)	1.97	1.71	2.27	2.61	**7.85**	40.79	41.69	64.73	65.56
	PC-GAN	9.14	1.52	3.90	2.75	8.20	36.50	38.98	76.03	78.37
	PointFlow (ours)	**1.61**	**1.06**	**1.74**	**2.42**	7.87	**46.83**	**46.98**	**60.88**	**59.89**
	Training set	-	-	1.50	1.92	7.38	57.25	55.44	59.67	58.46
Car	r-GAN	7.22	6.91	12.8	1.27	8.74	15.06	9.38	97.87	99.86
	l-GAN (CD)	1.97	1.71	4.43	1.55	6.25	38.64	18.47	63.07	88.07
	l-GAN (EMD)	1.97	1.71	2.21	1.48	5.43	39.20	39.77	69.74	68.32
	PC-GAN	9.14	1.52	5.85	1.12	5.83	23.56	30.29	92.19	90.87
	PointFlow (ours)	**1.61**	**1.06**	**0.87**	**0.91**	**5.22**	**44.03**	**46.59**	**60.65**	**62.36**
	Training set	-	-	0.86	1.03	5.33	48.30	51.42	57.39	53.27

and unit-variance globally. Following prior convention [1], we use 2048 points for each shape during both training and testing, although our model is able to sample an arbitrary number of points. We additionally report the performance of point clouds sampled from the training set, which is considered as an upper bound since they are from the target distribution.

In Table 1, we report the performance of different models, as well as their number of parameters in total (full) or in the generative pathways (gen). We first note that all the previous metrics (JSD, MMD, and COV) sometimes assign a better score to point clouds generated by models than those from the training set (marked in gray). The 1-NNA metric does not seem to have this problem and always gives a better score to shapes from the training set. Our model outperforms all baselines across all three categories according to 1-NNA and also obtains the best score in most cases as evaluated by other metrics. Besides, our model has the fewest parameters among compared models. In the supplementary materials, we perform additional ablation studies to show the effectiveness of different components of our model. Figure 3 shows some examples of novel point clouds generated by our model. Figure 4 shows examples of point clouds reconstructed from given inputs.

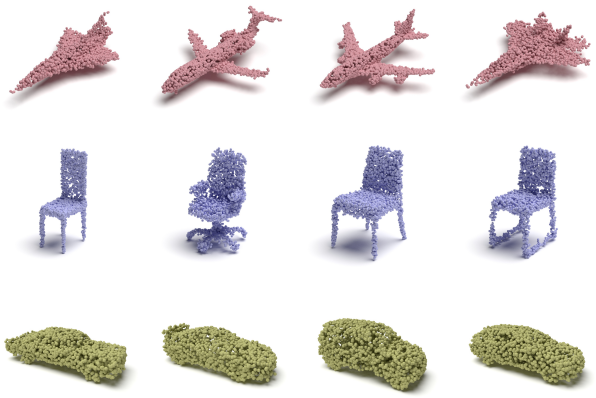

Figure 3: Examples of point clouds generated by our model. From top to bottom: airplane, chair, and car.

6.3. Auto-encoding

We further quantitatively compare the reconstruction ability of our flow-based auto-encoder with l-GAN [1] and AtlasNet [17]. Following the setting of AtlasNet, the state-of-the-art in this task, we train our auto-encoder on all shapes in the ShapeNet dataset. The auto-encoder is trained with the reconstruction likelihood objective $\mathcal{L}_{\text{recon}}$ only. At

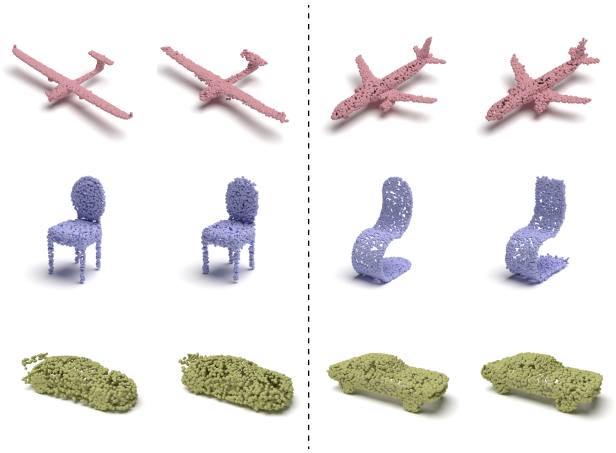

Figure 4: Examples of point clouds reconstructed from inputs. From top to bottom: airplane, chair, and car. On each side of the figure we show the input point cloud on the left and the reconstructed point cloud on the right.

Table 2: Unsupervised feature learning. Models are first trained on ShapeNet to learn shape representations, which are then evaluated on ModelNet40 (MN40) and ModelNet10 (MN10) by comparing the accuracy of off-the-shelf SVMs trained using the learned representations.

Method	MN40 (%)	MN10 (%)
SPH [23]	68.2	79.8
LFD [4]	75.5	79.9
T-L Network [14]	74.4	-
VConv-DAE [40]	75.5	80.5
3D-GAN [46]	83.3	91.0
l-GAN (EMD) [1]	84.0	**95.4**
l-GAN (CD) [1]	84.5	**95.4**
PointGrow [42]	85.7	-
MRTNet-VAE [13]	86.4	-
FoldingNet [49]	**88.4**	94.4
l-GAN (CD) [1] †	**87.0**	92.8
l-GAN (EMD) [1] †	86.7	92.2
PointFlow (ours)	86.8	**93.7**

† We run the official code of l-GAN on our pre-processed dataset using the same encoder architecture as our model.

test time, we sample 4096 points per shape and split them into an input set and a reference set, each consisting of 2048 points. We then compute the distance (CD or EMD) between the reconstructed input set and the reference set [1].

[1]We use a separate reference set because we expect the auto-encoder to learn the point distribution. Exactly reproducing the input points is acceptable behavior, but should not be given a higher score than randomly sampling points from the underlying point distribution.

Table 3: Auto-encoding performance evaluated by CD and EMD. AtlasNet is trained with CD and l-GAN is trained on CD or EMD. Our method is not trained on CD or EMD. CD and EMD scores are multipled by 10^4 and 10^2 respectively.

Model	# Parameters (M)	CD	EMD
l-GAN (CD) [1]	1.77	7.12	7.95
l-GAN (EMD) [1]	1.77	8.85	5.26
AtlasNet [17]	44.9	**5.13**	5.97
PointFlow (ours)	**1.30**	7.54	**5.18**

Although our model is not directly trained with EMD, it obtains the best EMD score, even higher than l-GAN trained with EMD and AtlasNet which has more than 40 times more parameters.

6.4. Unsupervised representation learning

We finally evaluate the representation learning ability of our auto-encoders. Specifically, we extract the latent representations of our auto-encoder trained in the full ShapeNet dataset and train a linear SVM classifier on top of it on ModelNet10 or ModelNet40 [47]. Only for this task, we normalize each individual point cloud to have zero-mean per axis and unit-variance globally, following prior works [53, 1]. We also apply random rotations along the gravity axis when training the auto-encoder.

A problem with this task is that different authors have been using different encoder architectures with a different number of parameters, making it hard to perform an apples-to-apples comparison. In addition, different authors may use different pre-processing protocols (as also noted by Yang *et al*. [49]), which could also affect the numbers.

In Table 2, we still show the numbers reported by previous papers, but also include a comparison with l-GAN [1] trained using the same encoder architecture and the exact same data as our model. On ModelNet10, the accuracy of our model is 1.5% and 0.9% higher than l-GAN (EMD) and l-GAN (CD), respectively. On ModelNet40, the performance of the three models is very close.

7. Conclusion and future works

In this paper, we propose PointFlow, a generative model for point clouds consisting of two levels of continuous normalizing flows trained with variational inference. Future work includes applications to other tasks such as point cloud reconstruction from a single image.

8. Acknowledgment

This work was supported in part by a research gift from Magic Leap. Xun Huang was supported by NVIDIA Graduate Fellowship.

References

[1] Panos Achlioptas, Olga Diamanti, Ioannis Mitliagkas, and Leonidas Guibas. Learning representations and generative models for 3d point clouds. In *ICML*, 2018. 2, 3, 4, 6, 7, 8

[2] Martin Arjovsky, Soumith Chintala, and Léon Bottou. Wasserstein generative adversarial networks. In *ICML*, 2017. 2

[3] Angel X. Chang, Thomas Funkhouser, Leonidas Guibas, Pat Hanrahan, Qixing Huang, Zimo Li, Silvio Savarese, Manolis Savva, Shuran Song, Hao Su, Jianxiong Xiao, Li Yi, and Fisher Yu. ShapeNet: An Information-Rich 3D Model Repository. Technical Report arXiv:1512.03012 [cs.GR], Stanford University — Princeton University — Toyota Technological Institute at Chicago, 2015. 6

[4] Ding-Yun Chen, Xiao-Pei Tian, Edward Yu-Te Shen, and Ming Ouhyoung. On visual similarity based 3d model retrieval. *Comput. Graph. Forum*, 22:223–232, 2003. 8

[5] Tian Qi Chen, Yulia Rubanova, Jesse Bettencourt, and David K Duvenaud. Neural ordinary differential equations. In *NeurIPS*, 2018. 2, 3

[6] Xi Chen, Diederik P Kingma, Tim Salimans, Yan Duan, Prafulla Dhariwal, John Schulman, Ilya Sutskever, and Pieter Abbeel. Variational lossy autoencoder. In *ICLR*, 2016. 3, 4

[7] Ivo Danihelka, Balaji Lakshminarayanan, Benigno Uria, Daan Wierstra, and Peter Dayan. Comparison of maximum likelihood and gan-based training of real nvps. *arXiv preprint arXiv:1705.05263*, 2017. 3

[8] Laurent Dinh, David Krueger, and Yoshua Bengio. Nice: Non-linear independent components estimation. *CoRR*, abs/1410.8516, 2014. 2, 3

[9] Laurent Dinh, Jascha Sohl-Dickstein, and Samy Bengio. Density estimation using real nvp. In *ICLR*, 2017. 2, 3

[10] Oren Dovrat, Itai Lang, and Shai Avidan. Learning to sample. *arXiv preprint arXiv:1812.01659*, 2018. 2

[11] Harrison A Edwards and Amos J. Storkey. Towards a neural statistician. In *ICLR*, 2017. 3

[12] Haoqiang Fan, Hao Su, and Leonidas J Guibas. A point set generation network for 3d object reconstruction from a single image. In *CVPR*, 2017. 2

[13] Matheus Gadelha, Rui Wang, and Subhransu Maji. Multiresolution tree networks for 3d point cloud processing. In *ECCV*, 2018. 2, 8

[14] Rohit Girdhar, David F. Fouhey, Mikel Rodriguez, and Abhinav Gupta. Learning a predictable and generative vector representation for objects. In *ECCV*, 2016. 8

[15] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In *NeurIPS*, 2014. 2

[16] Will Grathwohl, Ricky T. Q. Chen, Jesse Bettencourt, Ilya Sutskever, and David Duvenaud. Ffjord: Free-form continuous dynamics for scalable reversible generative models. In *ICLR*, 2019. 2, 3

[17] Thibault Groueix, Matthew Fisher, Vladimir G. Kim, Bryan Russell, and Mathieu Aubry. AtlasNet: A Papier-Mâché Approach to Learning 3D Surface Generation. In *CVPR*, 2018. 2, 7, 8

[18] Aditya Grover, Manik Dhar, and Stefano Ermon. Flow-gan: Combining maximum likelihood and adversarial learning in generative models. In *AAAI*, 2018. 3

[19] Ishaan Gulrajani, Faruk Ahmed, Martin Arjovsky, Vincent Dumoulin, and Aaron C Courville. Improved training of wasserstein gans. In *NeurIPS*, 2017. 2

[20] Chin-Wei Huang, David Krueger, Alexandre Lacoste, and Aaron C. Courville. Neural autoregressive flows. In *ICML*, 2018. 3

[21] Li Jiang, Shaoshuai Shi, Xiaojuan Qi, and Jiaya Jia. Gal: Geometric adversarial loss for single-view 3d-object reconstruction. In *ECCV*, 2018. 2

[22] Tero Karras, Samuli Laine, and Timo Aila. A style-based generator architecture for generative adversarial networks. In *CVPR*, 2019. 2

[23] Michael M. Kazhdan, Thomas A. Funkhouser, and Szymon Rusinkiewicz. Rotation invariant spherical harmonic representation of 3d shape descriptors. In *Symposium on Geometry Processing*, 2003. 8

[24] Diederik P. Kingma and Prafulla Dhariwal. Glow: Generative flow with invertible 1x1 convolutions. In *NeurIPS*, 2018. 2, 3

[25] Diederik P. Kingma, Tim Salimans, and Max Welling. Improving variational inference with inverse autoregressive flow. In *NeurIPS*, 2016. 3

[26] Diederik P Kingma and Max Welling. Auto-encoding variational bayes. In *ICLR*, 2014. 2, 4, 5

[27] Manoj Kumar, Mohammad Babaeizadeh, Dumitru Erhan, Chelsea Finn, Sergey Levine, Laurent Dinh, and Durk Kingma. Videoflow: A flow-based generative model for video. *arXiv preprint arXiv:1903.01434*, 2019. 3

[28] Andrey Kurenkov, Jingwei Ji, Animesh Garg, Viraj Mehta, JunYoung Gwak, Christopher B. Choy, and Silvio Savarese. Deformnet: Free-form deformation network for 3d shape reconstruction from a single image. In *WACV*, 2018. 2

[29] Chun-Liang Li, Manzil Zaheer, Yang Zhang, Barnabas Poczos, and Ruslan Salakhutdinov. Point cloud gan. *arXiv preprint arXiv:1810.05795*, 2018. 2, 3, 6

[30] Kejie Li, Trung Pham, Huangying Zhan, and Ian D. Reid. Efficient dense point cloud object reconstruction using deformation vector fields. In *ECCV*, 2018. 2

[31] David Lopez-Paz and Maxime Oquab. Revisiting classifier two-sample tests. In *ICLR*, 2017. 6

[32] Alireza Makhzani, Jonathon Shlens, Navdeep Jaitly, Ian Goodfellow, and Brendan Frey. Adversarial autoencoders. *arXiv preprint arXiv:1511.05644*, 2015. 2

[33] Aaron van den Oord, Nal Kalchbrenner, and Koray Kavukcuoglu. Pixel recurrent neural networks. In *ICML*, 2016. 2

[34] George Papamakarios, Theo Pavlakou, and Iain Murray. Masked autoregressive flow for density estimation. In *NeurIPS*, 2017. 3

[35] Ryan Prenger, Rafael Valle, and Bryan Catanzaro. Waveglow: A flow-based generative network for speech synthesis. *CoRR*, abs/1811.00002, 2018. 3

[36] Charles R Qi, Hao Su, Kaichun Mo, and Leonidas J Guibas. Pointnet: Deep learning on point sets for 3d classification and segmentation. In *CVPR*, 2017. 2

[37] Charles Ruizhongtai Qi, Li Yi, Hao Su, and Leonidas J Guibas. Pointnet++: Deep hierarchical feature learning on point sets in a metric space. In *NeurIPS*, 2017. 2

[38] Danilo Jimenez Rezende and Shakir Mohamed. Variational inference with normalizing flows. In *ICML*, 2015. 2, 3

[39] Danilo Jimenez Rezende, Shakir Mohamed, and Daan Wierstra. Stochastic backpropagation and approximate inference in deep generative models. In *ICML*, 2014. 2, 4

[40] Abhishek Sharma, Oliver Grau, and Mario Fritz. Vconv-dae: Deep volumetric shape learning without object labels. In *ECCV Workshops*, 2016. 8

[41] Matan Shoef, Sharon Fogel, and Daniel Cohen-Or. Pointwise: An unsupervised point-wise feature learning network. *arXiv preprint arXiv:1901.04544*, 2019. 2

[42] Yongbin Sun, Yue Wang, Ziwei Liu, Joshua E Siegel, and Sanjay E Sarma. Pointgrow: Autoregressively learned point cloud generation with self-attention. *arXiv preprint arXiv:1810.05591*, 2018. 2, 8

[43] Vladyslav Usenko, Jakob Engel, Jörg Stückler, and Daniel Cremers. Reconstructing street-scenes in real-time from a driving car. In *3DV*, 2015. 2

[44] Rianne van den Berg, Leonard Hasenclever, Jakub M. Tomczak, and Max Welling. Sylvester normalizing flows for variational inference. In *UAI*, 2018. 3

[45] Aaron Van den Oord, Nal Kalchbrenner, Lasse Espeholt, Oriol Vinyals, Alex Graves, et al. Conditional image generation with pixelcnn decoders. In *NeurIPS*, 2016. 2

[46] Jiajun Wu, Chengkai Zhang, Tianfan Xue, William T Freeman, and Joshua B Tenenbaum. Learning a Probabilistic Latent Space of Object Shapes via 3D Generative-Adversarial Modeling. In *NeurIPS*, 2016. 8

[47] Zhirong Wu, Shuran Song, Aditya Khosla, Fisher Yu, Linguang Zhang, Xiaoou Tang, and Jianxiong Xiao. 3d shapenets: A deep representation for volumetric shapes. In *CVPR*, 2015. 8

[48] Qiantong Xu, Gao Huang, Yang Yuan, Chuan Guo, Yu Sun, Felix Wu, and Kilian Weinberger. An empirical study on evaluation metrics of generative adversarial networks. *arXiv preprint arXiv:1806.07755*, 2018. 6

[49] Yaoqing Yang, Chen Feng, Yiru Shen, and Dong Tian. Foldingnet: Point cloud auto-encoder via deep grid deformation. In *CVPR*, 2018. 2, 8

[50] Wang Yifan, Shihao Wu, Hui Huang, Daniel Cohen-Or, and Olga Sorkine-Hornung. Patch-based progressive 3d point set upsampling. *arXiv preprint arXiv:1811.11286*, 2018. 2

[51] Lequan Yu, Xianzhi Li, Chi-Wing Fu, Daniel Cohen-Or, and Pheng-Ann Heng. Ec-net: an edge-aware point set consolidation network. In *ECCV*, 2018. 2

[52] Lequan Yu, Xianzhi Li, Chi-Wing Fu, Daniel Cohen-Or, and Pheng-Ann Heng. Pu-net: Point cloud upsampling network. In *CVPR*, 2018. 2

[53] Manzil Zaheer, Satwik Kottur, Siamak Ravanbakhsh, Barnabas Poczos, Ruslan R Salakhutdinov, and Alexander J Smola. Deep sets. In *NeurIPS*, 2017. 2, 8

[54] Maciej Zamorski, Maciej Zieba, Rafał Nowak, Wojciech Stokowiec, and Tomasz Trzciński. Adversarial autoencoders for generating 3d point clouds. *arXiv preprint arXiv:1811.07605*, 2018. 2