Discrete Point Flow Networks for Efficient Point Cloud Generation

Roman Klokov, Edmond Boyer, Jakob Verbeek

Discrete Point Flow Networks for Efficient Point Cloud Generation

Roman Klokov[1[0000−0001−9592−7009]], Edmond Boyer[1[0000−0002−1182−3729]], and Jakob Verbeek[2[0000−0003−1419−1816]]

[1] Univ. Grenoble Alpes, Inria, CNRS, Grenoble INP, LJK, 38000 Grenoble, France
{firsname.lastname}@inria.fr
[2] Facebook AI Research

Abstract. Generative models have proven effective at modeling 3D shapes and their statistical variations. In this paper we investigate their application to point clouds, a 3D shape representation widely used in computer vision for which, however, only few generative models have yet been proposed. We introduce a latent variable model that builds on normalizing flows with affine coupling layers to generate 3D point clouds of an arbitrary size given a latent shape representation. To evaluate its benefits for shape modeling we apply this model for generation, autoencoding, and single-view shape reconstruction tasks. We improve over recent GAN-based models in terms of most metrics that assess generation and autoencoding. Compared to recent work based on continuous flows, our model offers a significant speedup in both training and inference times for similar or better performance. For single-view shape reconstruction we also obtain results on par with state-of-the-art voxel, point cloud, and mesh-based methods.

Keywords: generative modeling, normalizing flows, 3D shape modeling, point cloud generation, single view reconstruction

1 Introduction

Generative shape models are used in numerous computer vision applications where they allow to encode 3D shape variations with respect to different attributes, such as shape classes or shape deformations, as well as to infer shapes from partial observations, for instance from a single or a few images. Central to shape models is the representation chosen for shapes that can be extrinsic, for example the ubiquitous voxels and octrees, or intrinsic as with meshes and point clouds. While extrinsic representations enable relatively straightforward extensions of 2D deep learning techniques to 3D, they suffer from their inherent trade-off between precision and complexity. This is why 3D shapes are often represented using intrinsic models, among which point clouds are a natural and versatile solution, serving as a basis for many 3D capturing methods, including most multi-view stereo and range sensing methods, *e.g.* kinect.

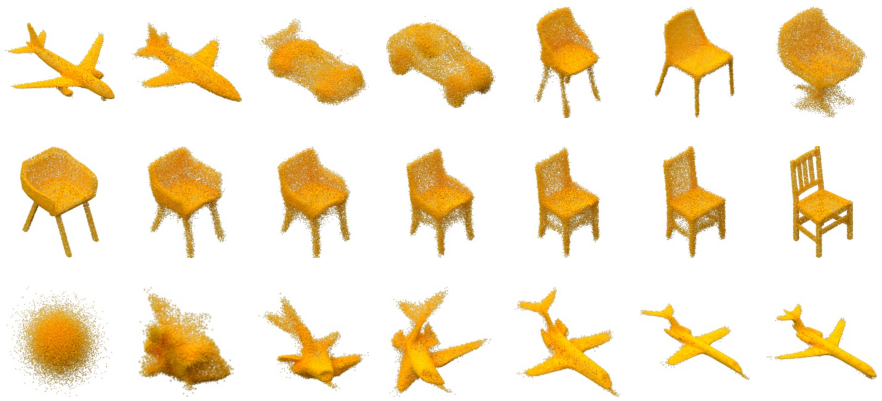

Fig. 1: Top: Point clouds sampled from DPF-Net for the ShapeNet classes *airplane*, *car*, and *chair*. Middle: Latent space interpolation between two point clouds from the test set. Bottom: Deformation of points across the flow steps.

Following the success of CNNs for 2D computer vision problems, many deep learning models have been proposed that can handle 3D data. This includes works on voxel grids [5, 9, 13, 15, 27, 34, 52, 53], octrees [46, 48], meshes [20, 36, 49, 51], point clouds [12, 21, 26, 33, 41, 42, 50], and implicit functions [35, 38]. While they provide effective tools to build predictive models of 3D shapes, *e.g.* from a single image, we investigate in this paper the less extensively explored and more generic problem of probabilistic generative 3D shape modeling.

Significant advances have been made in generative modeling of natural images using deep neural networks with convolutional architectures. Consequently, they can easily be adapted to generative shape models which are based on extrinsic representations, using regular 3D convolutional layers [52, 5, 27]. On the other hand, their extensions to intrinsic representations, such as point clouds and meshes, are less obvious and, to the best of our knowledge, so far, only Yang *et al.* [54] have studied generative models from which arbitrary size point clouds can be sampled without any conditioning information.

We explore a hierarchical latent variable model that treats the points as exchangeable variables, which allows us to model and sample point clouds of arbitrary size. Within this framework, each point cloud is considered as a sample from a shape-specific distribution over the 3D surface of the object, and these distributions are embedded in a latent space. To sample a point cloud, first, a vector is sampled in the latent shape space, and then, any desired number of 3D points can be sampled i.i.d. conditioned on the latent shape representation. Our model shares the high-level structure with PointFlow [54], but differs in the underlying network architectures, reducing the training and sampling time by more than an order of magnitude. In particular, our model builds on discrete normalizing flows with affine coupling layers [11] rather than continuous flows, and FiLM conditioning layers [39] to construct a flexible density on 3D points given

the latent shape representation. In Figure 1 we illustrate diverse point clouds sampled from our class-specific models, interpolation between point clouds in the learned latent shape space, and the sequential process by which the discrete normalizing flow warps the points to obtain the final shape.

We evaluate generative and autoencoding capabilities of our model, as well as its use for single-view shape reconstruction. We obtain similar or better performance compared to GAN-based models in terms of metrics that assess generation and autoencoding. Compared to recent work based on continuous flows, our model offers a significant speedup, for similar or better performance. For single view reconstruction our model performs on par with state-of-the-art methods, yet allows to reconstruct with arbitrarily large point clouds. Moreover, we analyze various design choices regarding data splitting and normalization.

2 Related work

Generative Models. Deep neural networks have sparked significant progress in generative modeling. The most widely adopted models are variational autoencoders (VAEs) [25, 45], generative adversarial networks (GANs) [14, 22], and normalizing flows [10, 11]. All three approaches share the basic principle of defining a latent variable z with a simple prior, *e.g.* unit Gaussian, and construct a complex conditional $p(x|z)$ on data x by means of deep neural networks. Maximum likelihood training of the resulting marginal $p(x)$ is generally intractable due to the non-linearities. To train the model, VAEs rely on an amortized inference network that produces a variational posterior $q(z|x)$. GANs, on the other hand, use a discriminator network to distinguish training examples and model samples, and use it as a signal to train the generative model. Alternatively to previous approaches, normalizing flow models rely on invertible neural network architectures to avoid the intractability of the marginalization altogether. In this case, the likelihood can be computed exactly by the means of the change of variable formula, and latent variables can be inferred deterministically. A variety of different normalizing flows has recently been proposed, see *e.g.* [2, 7, 11, 16, 23, 24, 44]. See [28, 37] for recent comprehensive reviews on normalizing flows.

Affine coupling layers [11] allow for a computation of the inverse in a closed form that is as efficient as the function itself. Within this approach, the activations A^ℓ in layer ℓ are partitioned in two groups, A_1^ℓ and A_2^ℓ. The first group is unchanged, and used to update the other group by scaling and translation, *i.e.* $A_2^{\ell+1} = A_2^\ell \odot s(A_1^\ell) + t(A_1^\ell)$, where $\odot$ denotes element-wise multiplication, and $s(\cdot)$ and $t(\cdot)$ can be arbitrary (non-invertible) neural networks. The inverse is trivially obtained by subtraction and division, since $A_1^{\ell+1} = A_1^\ell$. Many coupling layers with changing variable partitioning can be stacked to construct a complex invertible flow. Training and sampling the model require to compute the flow reverse directions, and since affine coupling layers are equally efficient in both directions, it means that both processes are fast. This is in contrast to some other normalizing flows, such as invertible ResNets [2, 7], or planar and radial flows [44], for which the inverse flow does not have a closed form.

Neural ordinary differential equations were recently proposed as a generalization of deep residual networks (ResNets, [18, 19]) in the limit of infinite depth [7]. Chen *et al*. [7] demonstrated that Neural ODEs can be used to define normalizing flow, which are referred to as "continuous normalizing flows".

Several conditional flow-based models have recently been proposed for vision tasks. In [31] flows conditioned on an input image are used for image segmentation, inpainting, denoising, and depth refinement. Their model is trained directly via maximum likelihood estimation, as their model does not include a global latent variable. Similarly, C-Flow [40] does not involve a global latent variable, and rather than treating point clouds as sets, it sorts the points to a regular pixel grid, and applies 2D normalizing flows for single image point cloud reconstruction. A conditional VAE model, where flow is used to define a flexible distribution on the latent variable given the conditioning data was introduced by [3]. This is similar in structure to our model for single view reconstruction. Their experiments, however, concern the prediction of point trajectories in 2D for hand-written digits, and traffic participants such as pedestrians and cars. The generative image model of [32] is related to ours, as a VAE model with a flow-based decoder. The application contexts, RGB images *vs*. point clouds, and resulting architectures are, however, quite different.

Point Cloud Generating Networks. Deep learning models for point cloud processing have received significant attention in recent years, see *e.g*. [1, 17, 26, 29, 41, 42, 47, 55, 54]. The PointNet architecture of Qi *et al*. [41] was the first to propose a deep network for recognition of point clouds. The points are first processed in an identical and independent manner by an MLP, and global max-pooling is used to aggregate the per-point information. KD-Net [26] and PointNet++ [42] add a notion of spatial proximity to the architecture, replacing global max-pooling with local aggregation. While these models can interpret point clouds, they cannot generate them.

Early point cloud generating networks [1, 12] produce point clouds with a fixed number of points n, by using a network with $n \times 3$ outputs. AtlasNet [17] mitigates this limitation by using a set of k square 2D patches, and deforming each of these non-linearly by using k patch-specific MLPs that takes as input 2D patch coordinates as well as a global shape representation. The shape vector is obtained from a point cloud encoder network (for autoencoding), or from a CNN trained for single-view image reconstruction. The point cloud GAN (PC-GAN, [29]) is related, but uses a single generator that that takes a global shape vector as input together with (arbitrarily many) samples from a unit Gaussian. Similarly to [12], AtlasNet is a conditional model, that generates point clouds *given* another point cloud or an image. In contrast, PC-GAN includes a second generator that models a distribution on the latent shape space, so it can generate point clouds in an unconditional manner.

The high-level hierarchical latent variable structure of PointFlow [54] is similar to PC-GAN. Rather than using adversarial training, however, they train the model using a VAE-like approach in which an inference network produces an approximate posterior on the latent shape representation. Moreover, they use

continuous normalizing flows [7] to define a prior on the shape space, and a conditional distribution on 3D points given a latent shape representation. Our work is based on the same high-level VAE-like structure as PointFlow, but differs in the design of the network components. Most importantly, we make use of efficient "discrete" affine coupling layers, avoiding the use of computationally expensive ODE solvers for training and generation needed for the "continuous" flows, resulting in a significant speed-up to train and sample from the model. We describe our "Discrete Point Flow Networks" in the next section.

3 Discrete Point Flow Networks

In this section we first present the high-level hierarchical latent variable model, followed by a more detailed description of the model components in Section 3.2.

3.1 Hierarchical Latent Variable Model for Point Cloud Generation

Our goal is to define a generative model over point clouds of variable size that represent 3D shapes. The defining characteristics of point clouds are that the number of points may vary from one cloud to another, and that there is no inherent ordering among the points.

Let $X = \{x_1, \ldots, x_n\}$ be a point cloud with $x_i \in \mathbb{R}^d$, where $d = 3$ for point clouds for 3D shapes. The dimension d may be larger in some cases, *e.g.* $d = 6$ when modeling 3D points equipped with surface normals. An exchangeable distribution is one that is invariant to permutations of the data, *i.e.*

$$p(x_1, \ldots, x_n) = p(x_{\pi_1}, \ldots, x_{\pi_n}), \tag{1}$$

where π is a permutation of the integers $1, \ldots, n$. Note that independence implies exchangeability, but the reverse does not hold.

De Finetti's representation theorem states that any exchangeable distribution can be written as a factored distribution, conditioned on a latent variable:

$$p(X) = \int_z p_\psi(z) \prod_{x \in X} p_\theta(x|z) \mathrm{d}z. \tag{2}$$

In the case of 3D point cloud modeling, the latent variable z can be thought of as an element in an abstract shape space, sampled from a prior $p_\psi(z)$. This construction allows for point clouds of different cardinality, since conditioned on the shape representation z, the elements of the point cloud are sampled i.i.d. Given this general framework, also adopted in [29, 54], the challenge is to:

1. Design a flexible model so that the conditional distribution $p_\theta(x|z)$ concentrates around the surface of the object represented by z.
2. Mitigate the intractability of the integral in Eq. (2) during training when using, *e.g.*, deep neural networks to construct $p_\theta(x|z)$.

Before we consider the design of $p_\theta(x|z)$ and $p_\psi(z)$ in Section 3.2, we describe how to deal with the integral in Eq. (2) using the VAE framework [25].

We efficiently approximate the intractable posterior $p(z|X)$ with an amortized inference network $q_\phi(z|X)$. The approximate posterior allows us to define a variational bound on the likelihood in Eq. (2) as using Jensen's inequality [4]:

$$\ln p(X) \geq \sum_{x \in X} \mathbb{E}_{q_\phi(z|X)}[\ln p_\theta(x|z)] - \mathcal{D}_{\mathrm{KL}}(q_\phi(z|X)||p_\psi(z)) \equiv -\mathcal{F}. \tag{3}$$

The first term aims to reconstruct the points $x \in X$ using shape representations sampled from $q_\phi(z|X)$, whereas the second term ensures that the approximate posterior cannot arbitrarily deviate from the prior. Following [25, 45] we use Monte Carlo sampling and the reparametrization trick to jointly minimize the loss $\mathcal{F}$ over θ, ψ and ϕ using stochastic gradient descent. The distributions $q_\phi(z|X)$, $p_\theta(x|z)$, and $p_\psi(z)$ and the underlying network architectures that make up the loss are detailed in the following section.

3.2 Design of Model Components

Shape-conditional Point Distribution. The density on points for a given latent shape, $p_\theta(x|z)$, needs to be flexible enough to concentrate its support around the surface of the 3D shape. To this end we construct a conditional form of normalizing flows based on affine coupling layers [11].

Let $y \in \mathbb{R}^3$ denote a latent variable for each 3D point x, with a Gaussian conditional distribution given by $p_\theta(y|z) = \mathcal{N}(y; \nu_\theta(z), \mathrm{diag}(\omega_\theta(z)))$, where $\nu_\theta(z)$ and $\omega_\theta(z)$ are non-linear functions of z. In the affine coupling layer, we partition the coordinates of y in two groups, y^c and y^u, and update y^u by affine transformation conditioned on y^c and the latent shape representation z, *i.e.* $x^u = y^u \odot s_\theta(y^c, z) + t_\theta(y^c, z)$, while leaving the conditioning coordinates unchanged, *i.e.* $x^c = y^c$. To achieve the desired expressivity, we stack many affine coupling layers, cycling through the six possible partitionings of the three coordinates. Each coupling layer in the resulting flow $f_\theta(x; z)$ is conditioned on the latent shape representation z by the means of the FiLM conditioning mechanism [39] in the scaling and translation functions.

In practice, the scaling and translation functions are implemented by MLPs, which inflate the dimensionality of y^c to D_{inf}, and then deflate it to the dimensionality of y^u. Simultaneously, a separate MLP takes the latent variable z and outputs conditioning coefficients of size D_{inf}, with which we multiply and shift the inflated hidden units in the scaling and translation functions. In Figure 2 we provide an overview of the architecture of our conditional coupling layers.

Using $f_\theta(x; z)$ to denote the invertible flow network that maps x to y, the change of variable formula allows to write the density of 3D points x given z as:

$$p_\theta(x|z) = \mathcal{N}(f_\theta(x; z); \nu_\theta(z), \mathrm{diag}(\omega_\theta(z))) \left| \det \left(\frac{\partial f_\theta(x; z)}{\partial x^\top} \right) \right|, \tag{4}$$

which enters into the loss defined in Eq. (3).

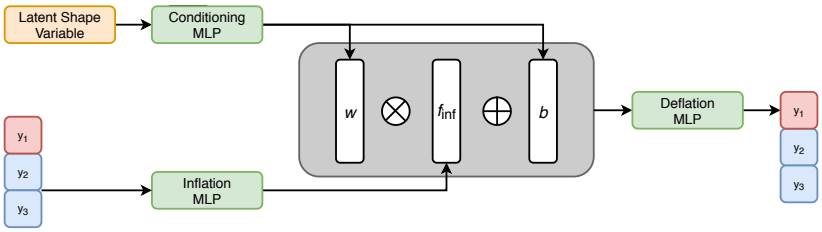

Fig. 2: Architecture of our conditional affine coupling layer applied to a single 3D point, with red dimension of the point being updated given the blue ones.

Amortized Inference Network. The amortized inference network $q_\phi(z|X)$ takes a point cloud and produces a distribution on the latent shape representation. We use a permutation invariant design based on the PointNet architecture for shape classification [41]. As an output, the model produces the mean and diagonal covariance matrix of a Gaussian on $z \in \mathbb{R}^D$, *i.e.* $q_\phi(z|X) = \mathcal{N}\left(z; \mu_\phi(X), \mathrm{diag}\left(\sigma_\phi(X)\right)\right)$.

Latent Shape Prior. Rather than using a unit Gaussian prior in the latent space, as is common in deep generative models, we learn a more expressive prior $p_\psi(z)$ by means of another normalizing flow $g_\psi(z)$ based on affine coupling layers, similar to [8, 54]. In our experiments it reduces the KL divergence in Eq. (3) by adapting the prior to fit the marginal posterior $\sum_X q_\phi(z|X)$, rather than forcing the inference network to induce a unit Gaussian marginal posterior, resulting in improved generative performance. Using this construction, we obtain the KL divergence as:

$$\mathcal{D}_{\mathrm{KL}}(q_\phi(z|X)||p_\psi(z)) = \mathbb{E}_{q_\phi(z|X)}[\ln p_\psi(z)] - \mathcal{H}(q_\phi(z|X)) \quad (5)$$

$$= \mathbb{E}_{q_\phi(z|X)}\left[\ln \mathcal{N}\left(g_\psi(z); \eta, \mathrm{diag}\left(\kappa\right)\right) + \ln \left|\det\left(\frac{\partial g_\psi(z)}{\partial z^\top}\right)\right|\right] - 1^\top \ln \sigma_\phi(X), \quad (6)$$

where we use Monte Carlo sampling to approximate the expectation.

Single-View Reconstruction Architecture. For single-view reconstruction, we follow [27] and define the model as:

$$p(X|v) = \int_z p_\psi(z|v) \prod_{x \in X} p_\theta(x|z) \mathrm{d}z, \quad (7)$$

where we replaced the latent shape prior $p_\psi(z)$ with an image conditioned one, $p_\psi(z|v)$. In this case, the latent shape flow g_ψ does not deform a parametric Gaussian, but rather a Gaussian whose mean and variance are computed from an image v by a CNN encoder. We train the model by optimizing a variational bound similar to Eq. (3), and using the PointNet inference network to obtain an approximate posterior. Figure 3 provides an overview of the data flow between the model components for training and point cloud generation.

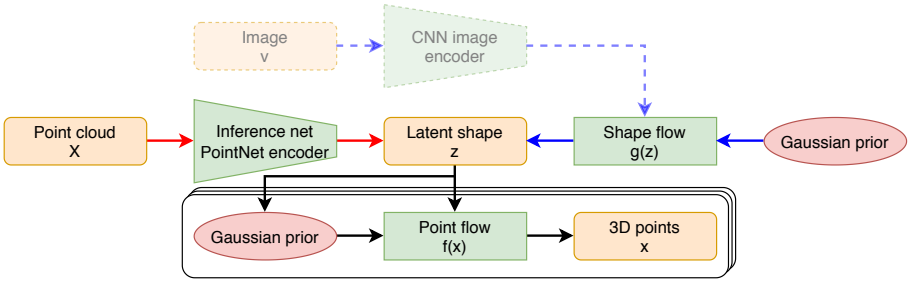

Fig. 3: Overview of DPF-Net: arrows indicate data flow to sample new point clouds (blue) and point cloud autoencoding (red), black arrows are used in both processes. During training flow modules are traversed in the reverse direction. For single-view reconstruction the shape prior is conditioned on the image (dashed).

4 Experiments

Datasets. In order to provide a comparison with prior academic studies on shape generation, we perform experiments on subsets of ShapeNet [6] dataset. For autoencoding we use the ShapeNetCore.v2, containing roughly 55k meshes from 55 classes. In the generative setting, following [54], we use single class subsets (*airplanes*, *cars*, and *chairs*) from the same dataset. For the single-view reconstruction task we used a subset of 13 major classes of ShapeNetCore.v1 from Choy *et al*. [9], which comes with rendered 137×137 images from 24 randomized viewpoints per shape. We substitute the voxel grids provided by Choy *et al*. with the original meshes to sample point clouds for training and evaluation.

Data Split. For autoencoding we use a random split of the data, distributed across train, validation, and test sets in a 70/10/20 proportion per class. In the generative setting, we use single class subsets from the same random split. By using a random data split per class we intentionally deviate from the official ShapeNet data split, used in [54], which splits into significantly different data subsets per class. For example, in case of airplanes, training and validation sets mostly contain regular passenger aircraft, while the test set is populated with fighter jets and spaceships. While such a split could be useful in the context of autoencoding to assess out-of-distribution generalization, a significant mismatch between training and test set is undesirable for evaluation of generative models which are supposed to fit the training distribution. For single-view reconstruction we use the train/test split from [9].

Normalization. The original meshes in the ShapeNet are not normalized for position and scale which negatively affects the reconstruction quality. We therefore, additionally use a normalized version of the dataset, where we preprocess each mesh separately so that the sampled point clouds are (approximately) zero mean, and tightly fit in a unit diameter sphere. For generative experiments we use models trained and evaluated on normalized data. In case of the autoencoding,

we report results for two separate DPF-Nets trained either on non-normalized or normalized data, where in the latter case we rescale point clouds to original scales before evaluation for comparability with the rest of the models. While using non-normalized data, similarly to [54], we perform global normalization across all shapes by translation to the aggregate center of all the training shapes, but do not rescale to unit global variance. For single-view reconstruction we compare models trained on normalized data, but evaluate them in the unit radius sphere scale for comparability with related work.

Point clouds are uniformly sampled from the meshes by sampling polygons with a probability proportional to their area, and then uniformly sampling a point per each selected polygon. Unlike previous works using precomputed point clouds, we perform this procedure on the fly, thus obtaining a different random point cloud each time we process a 3D shape. During both training and evaluation we sample two point clouds for each shape: one is used as input to the inference network, the other for optimization or evaluation of the decoder. We use $2,048$, $2,048$, and $2,500$ points for training and quantitative evaluation for generative, autoencoding, and single-view reconstruction tasks accordingly.

Evaluation Metrics. We follow the standard protocol [1, 12, 54] and use Chamfer distance (CD) and earth mover's distance (EMD) to assess point cloud reconstructions. To measure the generative properties we follow [1, 54], and use metrics to compare equally sized *sets* of generated and reference point clouds:

- The Jensen-Shannon divergence (JSD) compares the marginal distributions obtained by taking the union of all generated (or reference) point clouds, and quantizing the distributions to a voxel grid.
- The Minimum matching distance (MMD) computes the average distance of reference point clouds to their nearest (in CD/EMD) generated point cloud.
- Coverage (COV) is the fraction of reference point clouds matched by minimum CD/EMD distance by at least one generated point cloud.
- 1-nearest neighbour accuracy (1-NNA) classifies generated and reference point clouds as belonging to either of these two sets using a leave-one-out 1-nearest neighbor classifier (using CD/EMD). Ideal accuracy is 50%.

For single-view reconstruction we additionaly report F1-score. For more detailed descriptions of these metrics we refer to [1, 54].

4.1 Experimental Setup

Inference Network. We use a PointNet encoder [41] with the number of features progressing over layers as $3-64-128-256-512$, followed by a max-pooling across points, and two fully-connected layers of sizes 512 and D, where D is the size of the latent space. We use $D = 512$ in the autoencoding and single-view reconstruction experiments, and $D = 128$ for generative modeling.

Latent Shape Prior. We use 14 affine coupling layers to construct the latent space prior. In the coupling layers, we alternate between two orthogonal partitioning schemes: odd and even dimensions are split in different groups; the first

$D/2$ dimension go in one group, and the remaining in the other. For single-view reconstruction we use ResNet18 image encoder, similarly to [17, 51, 50].

Point Decoder. The point decoder $p(x|z)$ starts with a three-dimensional Gaussian with mean and variances computed from z by an MLP with two hidden layers. This Gaussian is transformed by 63 of our conditioned affine coupling layers, each consisting of (i) two fully-connected layers that map z to the FiLM conditioning coefficients, each of size 64, (ii) two fully-connected layers that inflate input dimensionality to 64 hidden units (at which point the FiLM conditioning is performed), (iii) a final fully-connected layer that deflate the dimensionality to compute the scaling and translation functions.

Baselines. We retrained AtlasNet [17], l-GAN-CD/EMD [1], PointFlow [54], and DCG [50] with our split of the ShapeNet dataset, using the implementation provided by the authors and our data processing pipeline. For improved comparability we also modified the point cloud encoders in all models to match each other (except for l-GANs, since it significantly worsened their results). To match other approaches, we used $2,048$ points per cloud for AtlasNet in the autoencoding task and consequently set the number of learned primitives to 16.

Oracles. For all the tasks we also provide an "oracle" to assess the best possible performance values. For autoencoding and single-view reconstruction, the oracle samples a second point cloud from the ground truth mesh, rather than generating a point cloud. For generative modeling, the oracle uses the point clouds from the training set, instead of sampling point clouds from the model.

Our code is publicly available at https://github.com/Regenerator/dpf-nets.

4.2 Generative Modeling Evaluation

Efficiency Comparison with PointFlow. We compare our DPF-Networks in terms of computational efficiency and memory footprint to PointFlow [54]. We train both models for point cloud generation, and report the number of parameters and total training time. To compute the training memory footprint, training time, and generation time per sample (point cloud), we divided total GPU memory occupied during training, batch iteration time, and batch generation time, respectively, by the batch size.

Both models were run on a single TITAN RTX GPU. We estimate the total training time for PointFlow after the initial 100 epochs of training, which took 2 days, and assume that the full training procedure requires $4,000$ epochs, as reported by the authors of PointFlow. We observed that the training procedure

Table 1: Efficiency comparison for DPF-Nets and PointFlow generative models.

Model	Nr. params., 10^6	Mem. footprint, Mb/sample	tr. time, ms/sample	total tr. time, days	gen. time, ms/sample
PointFlow [54]	1.63	470	500	80	150
DPF-Net (Ours)	3.76	370	16	1.1	4

Table 2: Generative modeling results. Oracle results are underlined when the are not the best. JSD and MMD-EMD are multiplied by 10^2, MMD-CD by 10^4.

Category	Model	JSD↓	MMD↓		COV↑, %		1-NNA↓, %	
			CD	EMD	CD	EMD	CD	EMD
Airplane	l-GAN-CD [1]	2.76	**5.69**	5.16	39.5	17.1	72.9	92.1
	l-GAN-EMD [1]	1.77	6.05	**4.15**	39.7	40.4	75.7	73.0
	PointFlow [54]	1.42	6.05	4.32	**44.7**	**48.4**	**70.9**	**68.4**
	DPF-Nets (Ours)	**0.94**	6.07	4.26	**46.8**	**48.4**	**70.6**	**67.0**
	Oracle	0.50	5.97	3.98	51.4	52.7	49.8	48.2
Car	l-GAN-CD [1]	2.65	**8.83**	5.36	41.3	15.9	**62.6**	92.7
	l-GAN-EMD [1]	1.31	9.00	**4.40**	38.3	32.9	65.2	**63.2**
	PointFlow [54]	0.59	9.53	4.71	**42.3**	35.8	70.1	74.2
	DPF-Nets (Ours)	**0.45**	9.59	4.61	**43.4**	**45.7**	70.3	**64.3**
	Oracle	0.37	9.24	4.56	52.8	52.7	50.9	50.5
Chair	l-GAN-CD [1]	3.65	**16.66**	7.91	42.3	17.1	68.5	96.5
	l-GAN-EMD [1]	1.27	**16.78**	**5.75**	44.3	43.8	66.6	67.8
	PointFlow [54]	1.51	17.15	6.20	43.3	46.5	67.0	70.4
	DPF-Nets (Ours)	**1.01**	17.08	6.14	**46.9**	**48.5**	**63.5**	**64.8**
	Oracle	0.49	16.39	5.71	52.8	53.4	49.7	49.6

slowed down over the course of training, because ODE-solver gradually increases the number of iterations to meet the required tolerance. Thus, all timings in Table 1 for PointFlow should be understood as lower bounds.

From the results in Table 1 we see that even though DPF-Networks have more parameters, the associated training memory footprint is lower and, our model is approximately 30 times faster both in training and inference iterations, and can be trained in a single day.

Quantitative Results. We compare to l-GANs and PointFlow models and report oracle performance as a reference. Given the prohibitive computation cost of complete PointFlow training, we provide results obtained after training for four days, which is four times the full training time of DPF-Net in the same setting. In order to account for random sampling every model is evaluated using ten different sets of generated objects, each of the size of the test set. Thus, for each metric we report mean values over ten runs. In addition to the best values, in Table 2 we also write in bold results that are within two standard deviations of the best result.

Overall, DPF-Networks yield the best results in terms of JSD, COV-CD/EMD and 1-NNA-CD/EMD, except for the 1-NNA-CD for *car*. This confirms that our DPF-Network is capable of generating more realistic and diverse sets of point clouds, for random samples from our model see Figure 1.

L-GAN-CD experiences mode collapses, and generates objects with good CD values, but with very poor coverage and 1-NNA in terms of the EMD metric. PointFlow shows performance similar to ours, except for JSD, while being significantly slower in both training and sampling. In contrast to the evaluations performed in [54], based on the official split, in our experiments the oracle obtains the best performances for all metrics, except for MMD (see underlined

Table 3: Autoencoding results. † - results from [54] on the official split, ∗ - results for equal training time as DPF-Net on the random split.

Metric	CD $\times 10^4$	EMD $\times 10^2$
l-GAN-CD [1]	7.07	7.70
l-GAN-EMD [1]	9.18	5.30
AtlasNet [17]	**5.66**	5.81
PointFlow† [54]	7.54	5.18
PointFlow*	10.22	6.58
DPF-Net, orig.	6.85	5.06
DPF-Net, norm.	6.17	**4.37**
Oracle	3.10	3.13

results). We believe that this highlights the fact the MMD metric does not favor diversity in the generated point clouds, but instead favors point clouds with low CD/EMD distances to all the reference shapes. If the generated point clouds contain a subset of high quality modes from the test subset, the metric can yield good results, even better than the oracle. DPF-Nets and PointFlow yield qualitatively similar point cloud samples, we provide a comparison of samples in the supplementary material.

4.3 Autoencoding Evaluation

We compare DPF-networks with other models in terms of autoencoding performance in Table 3. Similarly to generative experiments, we restricted the training time of PointFlow, this time, to match the training time of our approach which was approximately a week. Among models trained on non-normalized data, DPF-Net (orig.) achieve the best results in the EMD metric and second best in the CD metric, outperformed only by the non-generative AtlasNet which is trained by optimization of the CD metric. The DPF-Net outperforms both l-GANs which were specifically optimized for the CD/EMD metrics, while being trained by

Fig. 4: Qualitative comparison of the models from Table 3 for the autoencoding task with sparse (top) and dense (bottom) inputs.

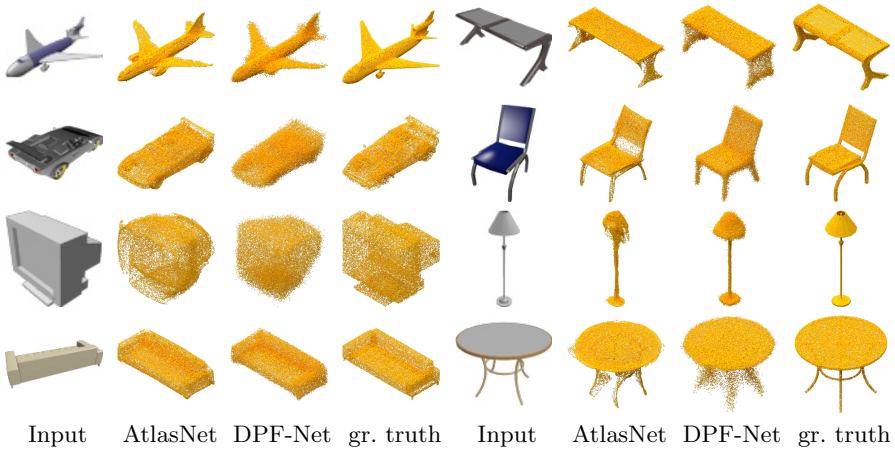

Fig. 5: Qualitative comparison for single-view reconstruction task.

optimization of the likelihood lower bound. Importantly DPF-Nets outperform PointFlow in both metrics under the same and extended computational budget.

When our model is trained on normalized data, results significantly improve, achieving state-of-the-art among generative models for both metrics. This underlines the importance of proper data normalization for shape modeling.

In Figure 4 we qualitatively compare our autoencoding results to l-GANs, AtlasNet, and PointFlow. All approaches can work with arbitrary size inputs, but only AtlasNet, PointFlow, and DPF-Nets can reconstruct with arbitrary density. In this comparison we use 512 and $32,768$ points as sparse and dense inputs, while reconstructing fixed $2,048$ points for l-GANs and $32,768$ point for AtlasNet, PointFlow and DPF-Nets. Models with better CD values (l-GAN-CD, AtlasNet) tend to concentrate points in some regions of reconstructed shapes, while models with better EMD values (l-GAN-EMD, DPF-Nets) distribute points more evenly. While AtlasNet achieves best CD, its reconstructions contain sharp plane-like artifacts. Our DPF-Nets produce overall smoother reconstructions, but, on the other hand, suffer from more noise.

4.4 Single-View Reconstruction

In this section we test DPF-Nets on the inference of 3D point clouds from single images. The architecture used for this specific task is depicted in Figure 3 and detailed in Section 3.2. We compare our results to recent state-of-the-art methods in the field. This includes: the voxel-based PRN [27], point cloud-based approaches of AtlasNet [17] and DCG [50], and the mesh-based Pixel2Mesh [51].

Although convenient, in general, comparison to voxel-based approaches should be taken with a grain of salt, since it is biased. To compute the proposed metrics either ground truth or reconstructed voxelized shapes are fed to the marching

Table 4: Single-view reconstruction results. †: results taken from [51].

Model	CD↓, $\times 10^3$	EMD↓, $\times 10^2$	F1↑, $\tau = 0.001, \%$
PRN [27]	7.56	11.00	**53.1**
AtlasNet [17]	**5.34**	12.54	52.2
DCG [50]	6.35	18.94	45.7
Pixel2Mesh† [51]	5.91	13.80	-
DPF-Nets	5.51	**10.95**	52.4
Oracle	1.10	5.70	84.0

cubes algorithm to obtain final meshes which are used to sample point clouds. Resulting ground truth meshes in that case are crude approximations of the original meshes, used in the evaluation of the point cloud and mesh-based approaches. Moreover, there are cases both in voxelized data and reconstructions, when the marching cubes algorithm fails to output meshes.

The results in Table 4 show that DPF-Net clearly outperforms earlier works in terms of the EMD metric. It also achieves best results in terms of the F1-score among point cloud and mesh-based models. In terms of CD, similarly to autoencoding it is outperformed only by AtlasNet with a small margin. This validates the ability of normalizing flows to capture complex distributions in 3D and to model shape surfaces.

Qualitative single-view reconstruction results can be found in Figure 5. Note that a single reconstruction model has been trained across all 13 classes for both AtlasNet and DPF-Net. Similarly to the autoencoding task, compared to AtlasNet our approach produces more evenly distributed point clouds without sharp dense clusters, but introduces more noise.

5 Conclusion

We presented DPF-Networks, a generative model for point clouds of arbitrary size. DPF-Nets are based on a latent variable model and use normalizing flows with affine coupling layers to construct a flexible, yet tractable, shape conditional density on 3D points, and an expressive latent shape space prior. They are trained akin to VAEs, using a permutation invariant point cloud encoder as approximate posterior distribution over the latent shape space.

The evaluation on the ShapeNet dataset demonstrates that DPF-nets improve generative performance metrics over previous work in most metrics and classes. Compared to a recent related work based on continuous normalizing flows, our model is between one and two orders of magnitude faster to train and sample from. Applied to single view reconstruction, DPF-Nets outperform state-of-the-art methods, hence showing promising capabilities in 3D shape modeling.

Acknowledgements. Work done while Jakob Verbeek was at INRIA.

A Training Details

All the models were trained with AMSGrad optimizer [43] with decoupled weight decay regularization [30] with step-like schedule for the learning rate. All the hyperparameters for all the experiments can be found on the paper webpage https://github.com/Regenerator/dpf-nets.

B Detailed Generative Modeling Performance

In Table 5 we present the same evaluation as in Table 2 of the main paper, but include the standard deviations across the ten sets of point clouds sampled from each model.

C Qualitative Results for Autoencoding

In Figure 6 and Figure 7 we show reconstruction results for l-GANs, AtlasNet, PointFlow, and ours DPF-Net from sparse and dense input point clouds. All the models used 512 and 32,768 points as a sparse or dense input accordingly. Note, that AtlasNet, PointFlow and DPF-Net are able to reconstruct arbitrary densely (32,768 points here), for l-GANs output size is fixed to 2,048.

D Qualitative Results for Generation

In figures 8—10 we provide additional samples from the models trained for the Airplane, Car, and Chair classes. For each point cloud we sample $32,768$ points.

E Additional interpolations

In figures 11—13 we provide latent space interpolations from the models trained for the Airplane, Car, and Chair classes. We sample two shapes from the test data (left- and right-most in the figures), and interpolate between corresponding latent variable samples to obtain a path in the shape space. For each latent shape on the path we then sample a point cloud of size $32,768$.

F Additional flow illustrations

In figures 14—16 we provide examples of the generating flow for sampled point clouds from the classes Airplane, Car, and Chair. We sample a shape from data, obtain initial Gaussian from the corresponding latent variable, sample $32,768$ points by the flow, and then visualize the evolution of the point cloud across the initial Gaussian and layers 32, 48, 56, 60, 62, 63 of the generative flow network.

Table 5: Generative modeling evaluation. JSD and MMD-EMD are multiplied by 10^2, MMD-CD by 10^4. Cases where the oracle does not obtain best results are underlined.

Category	Model	JSD↓	MMD↓		COV↑, %		1-NNA↓, %	
			CD	EMD	CD	EMD	CD	EMD
Airplane	1-GAN-CD [1]	2.76 ± 0.16	**5.69** ± 0.04	5.16 ± 0.02	39.5 ± 0.8	17.1 ± 0.6	72.9 ± 0.8	92.1 ± 0.6
	1-GAN-EMD [1]	1.77 ± 0.13	6.05 ± 0.04	**4.15** ± 0.02	39.7 ± 1.4	40.4 ± 1.2	75.7 ± 0.6	73.0 ± 1.2
	PointFlow [54]	1.42 ± 0.12	6.05 ± 0.05	4.32 ± 0.01	**44.7** ± 1.2	**48.4** ± 1.0	**70.9** ± 1.0	**68.4** ± 1.0
	DPF-Nets (Ours)	**0.94** ± 0.11	6.07 ± 0.04	4.26 ± 0.02	**46.8** ± 1.2	**48.4** ± 0.9	**70.6** ± 1.0	**67.0** ± 1.2
	Oracle	0.50 ± 0.04	5.97 ± 0.09	3.98 ± 0.01	51.4 ± 1.0	52.7 ± 1.3	49.8 ± 1.3	48.2 ± 1.1
Car	1-GAN-CD [1]	2.65 ± 0.07	**8.83** ± 0.06	5.36 ± 0.01	41.3 ± 0.8	15.9 ± 1.3	**62.6** ± 0.6	92.7 ± 0.4
	1-GAN-EMD [1]	1.31 ± 0.10	**9.00** ± 0.08	**4.40** ± 0.01	38.3 ± 1.2	32.9 ± 0.7	65.2 ± 0.4	**63.2** ± 1.0
	PointFlow [54]	0.59 ± 0.02	9.53 ± 0.06	4.71 ± 0.01	**42.3** ± 1.0	35.8 ± 1.3	70.1 ± 0.9	74.2 ± 0.6
	DPF-Nets (Ours)	**0.45** ± 0.02	9.59 ± 0.04	4.61 ± 0.01	**43.4** ± 0.9	**45.8** ± 1.0	70.3 ± 0.6	**64.3** ± 1.5
	Oracle	0.37 ± 0.03	9.24 ± 0.06	4.56 ± 0.01	52.8 ± 1.1	52.7 ± 0.9	50.9 ± 1.1	50.5 ± 1.2
Chair	1-GAN-CD [1]	3.65 ± 0.09	**16.66** ± 0.08	7.91 ± 0.02	42.3 ± 0.5	17.1 ± 0.5	68.5 ± 0.5	96.5 ± 0.1
	1-GAN-EMD [1]	1.27 ± 0.06	**16.78** ± 0.07	**5.75** ± 0.01	44.3 ± 0.9	43.8 ± 1.0	66.6 ± 0.6	67.8 ± 0.7
	PointFlow [54]	1.51 ± 0.11	17.15 ± 0.10	6.20 ± 0.01	43.3 ± 0.8	**46.5** ± 1.0	67.0 ± 0.3	70.4 ± 0.6
	DPF-Nets (Ours)	**1.01** ± 0.06	17.08 ± 0.11	6.14 ± 0.01	**46.9** ± 0.8	**48.5** ± 1.1	**63.5** ± 1.3	**64.8** ± 0.7
	Oracle	0.49 ± 0.01	16.39 ± 0.07	5.71 ± 0.01	52.8 ± 0.8	53.4 ± 1.1	49.7 ± 0.7	49.6 ± 0.9

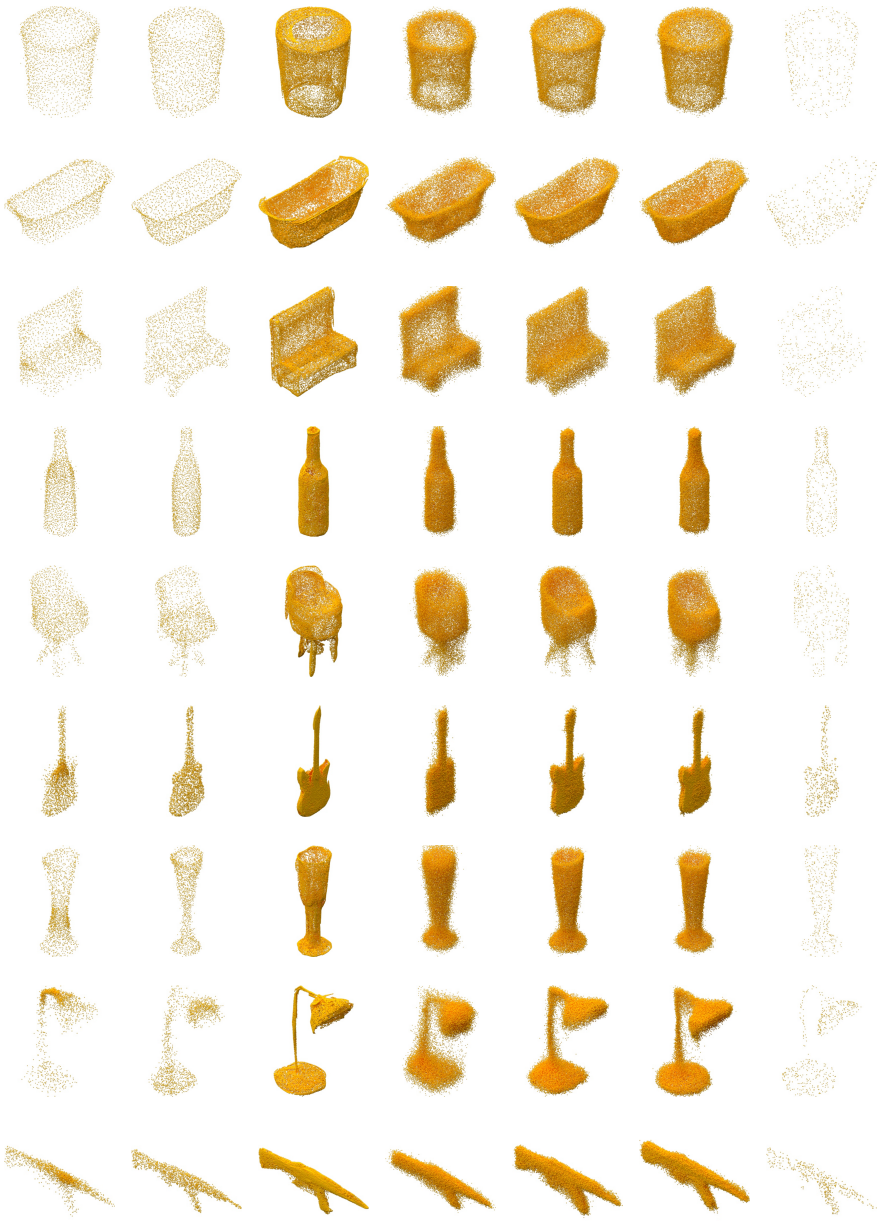

Fig. 6: Qualitative comparison in the autoencoding task with sparse inputs. Left to right: reconstructions from l-GAN-CD, l-GAN-EMD, AtlasNet, PointFlow, DPF-Nets (orig.), DPF-Nets (norm.), and ground-truth.

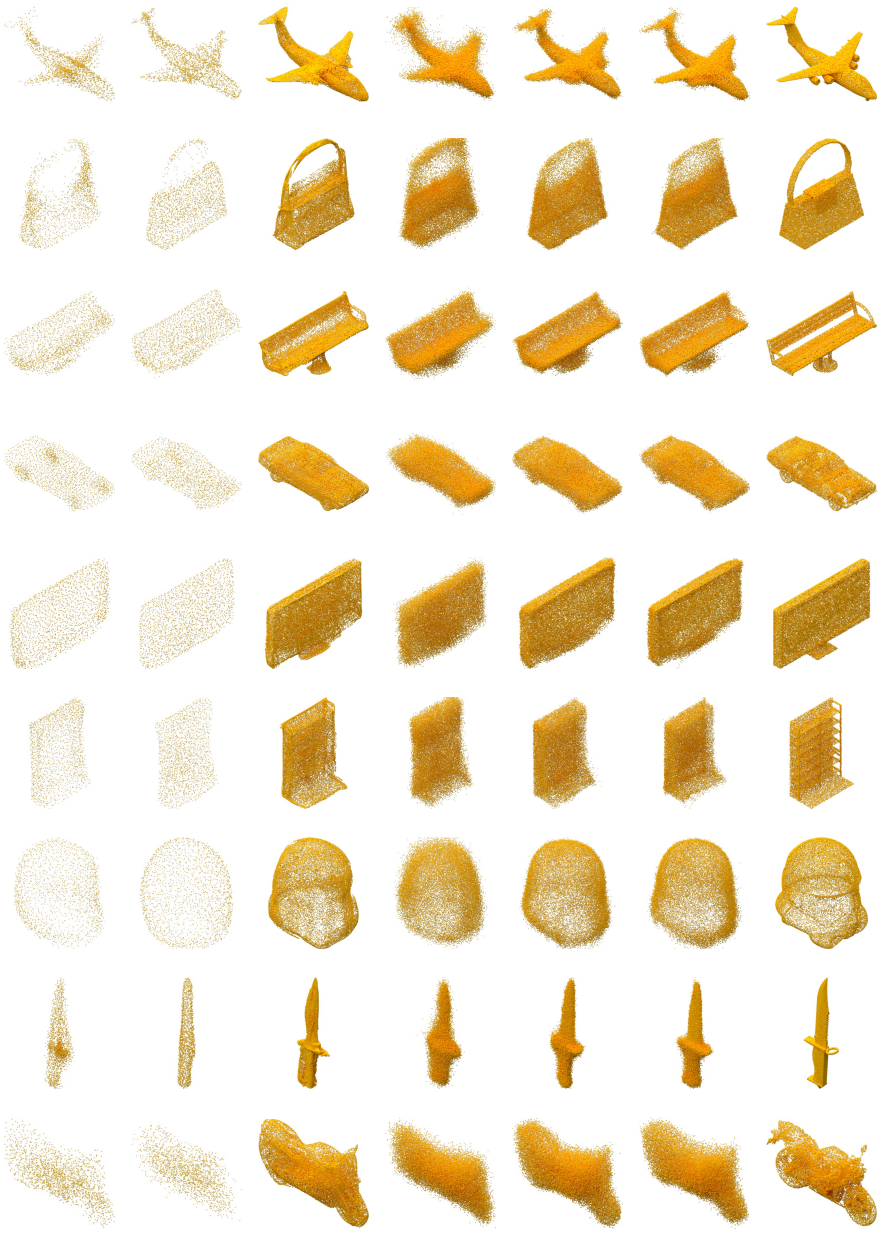

Fig. 7: Qualitative comparison in the autoencoding task with dense inputs. Left to right: reconstructions from l-GAN-CD, l-GAN-EMD, AtlasNet, PointFlow, DPF-Nets (orig.), DPF-Nets (norm.), and ground-truth.

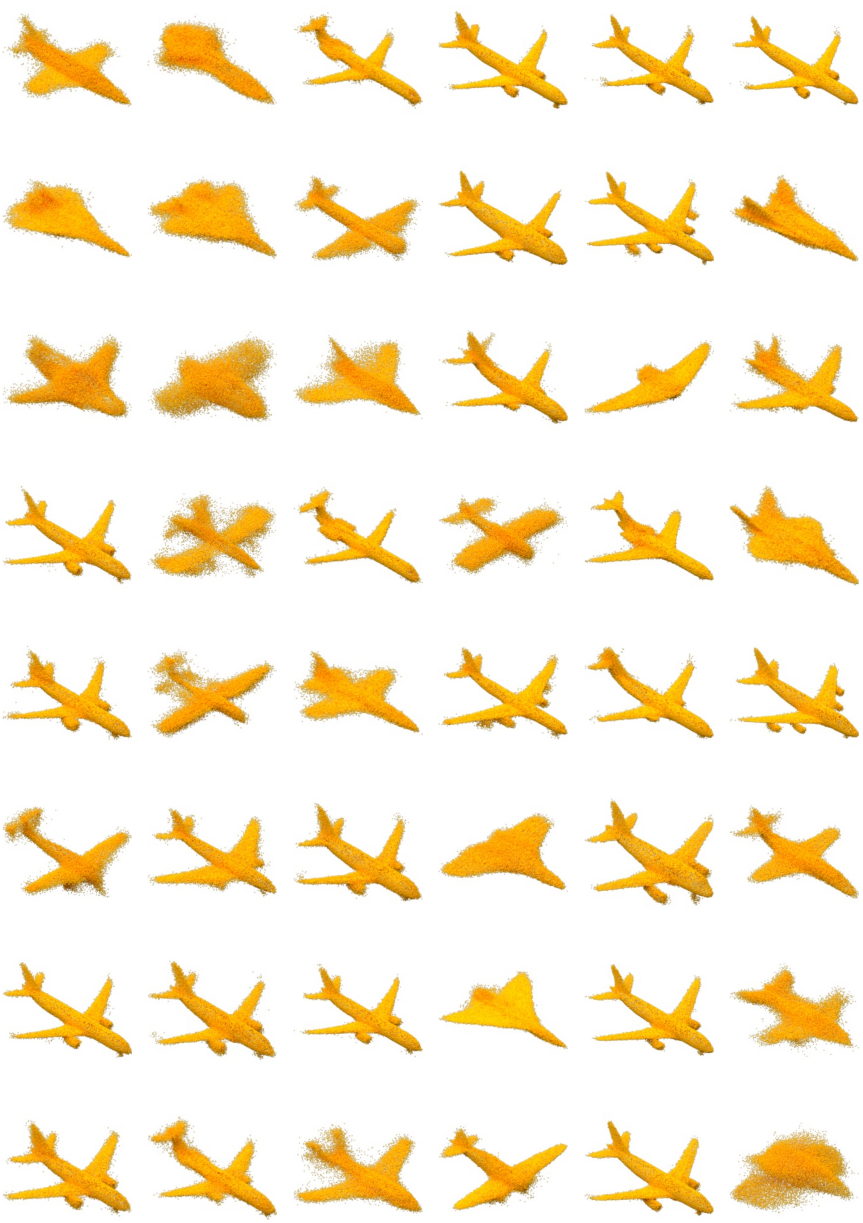

Fig. 8: Random samples from models trained on the Airplane class. Columns 1—3 samples from PointFlow, columns 4—6 samples from DPF-Net.

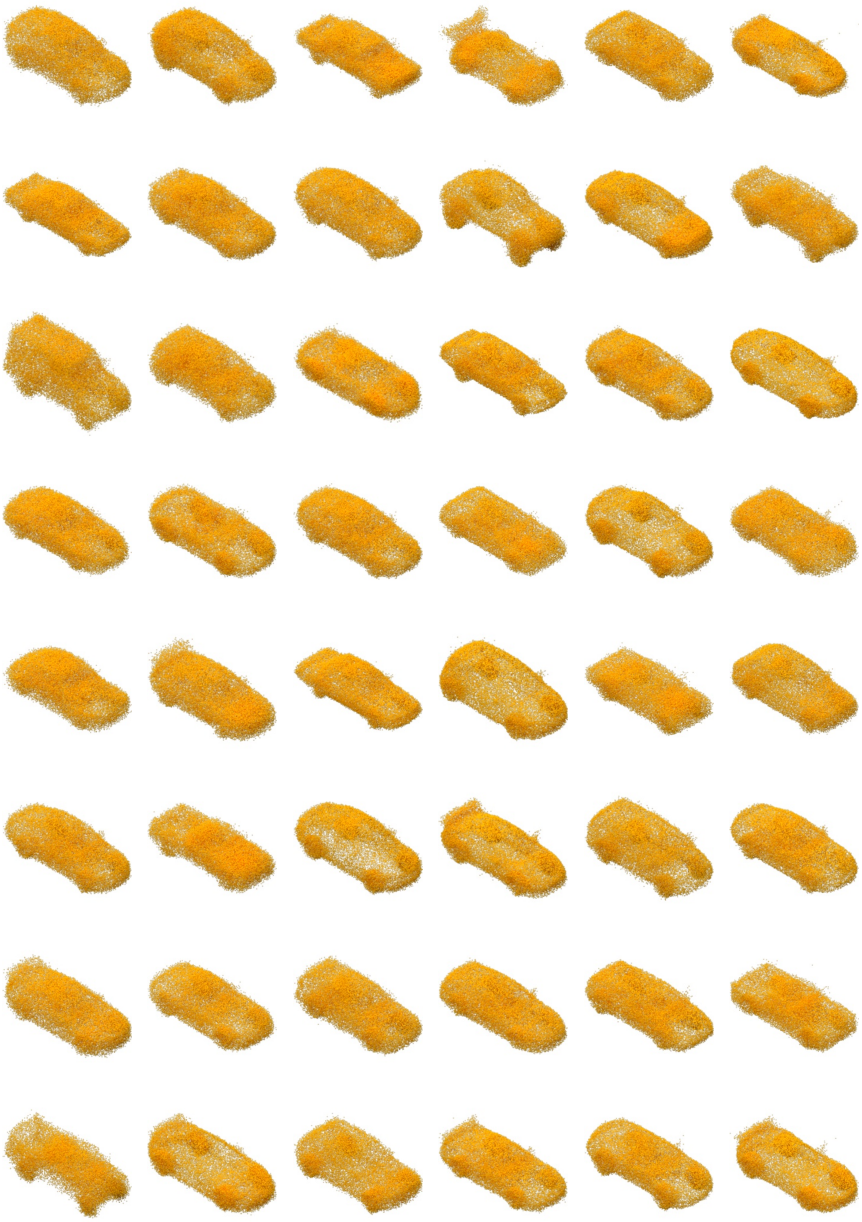

Fig. 9: Random samples from models trained on the Car class. Columns 1—3 samples from PointFlow, columns 4—6 samples from DPF-Net.

Fig. 10: Random samples from models trained on the Chair class. Columns 1—3 samples from PointFlow, columns 4—6 samples from DPF-Net.

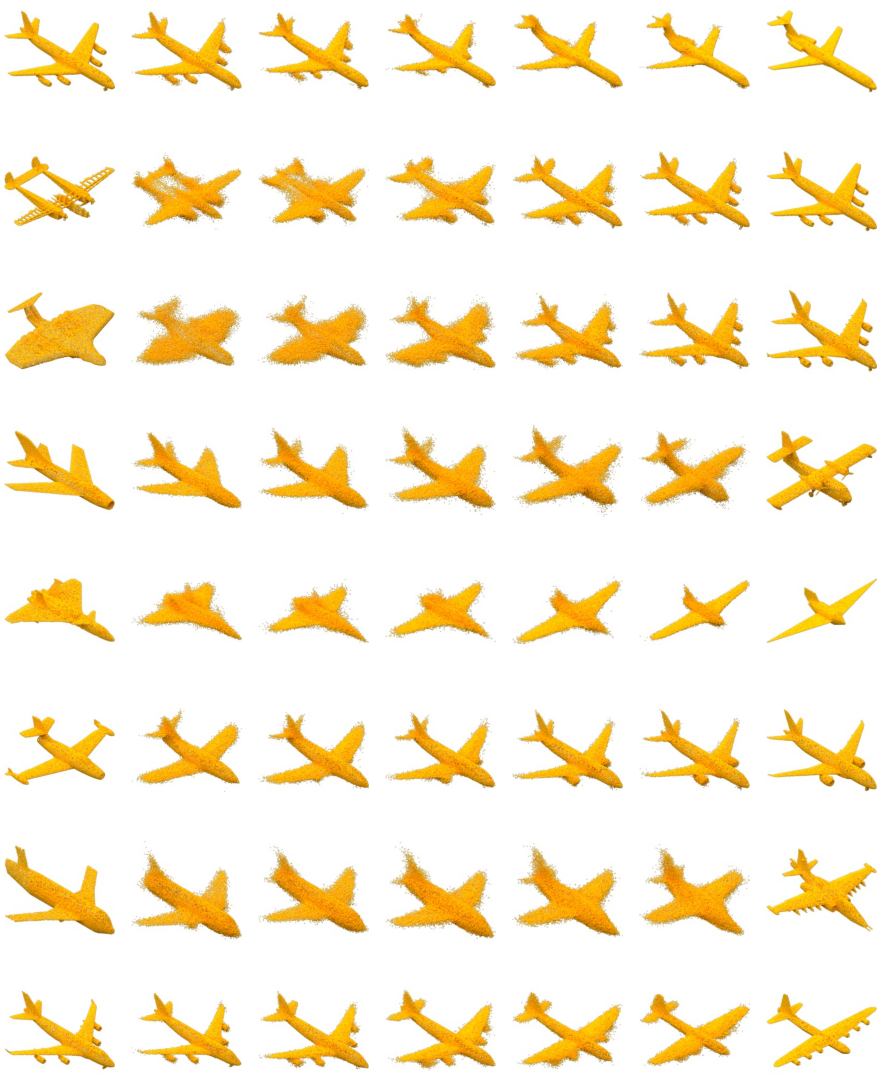

Fig. 11: Interpolations between Airplane data samples.

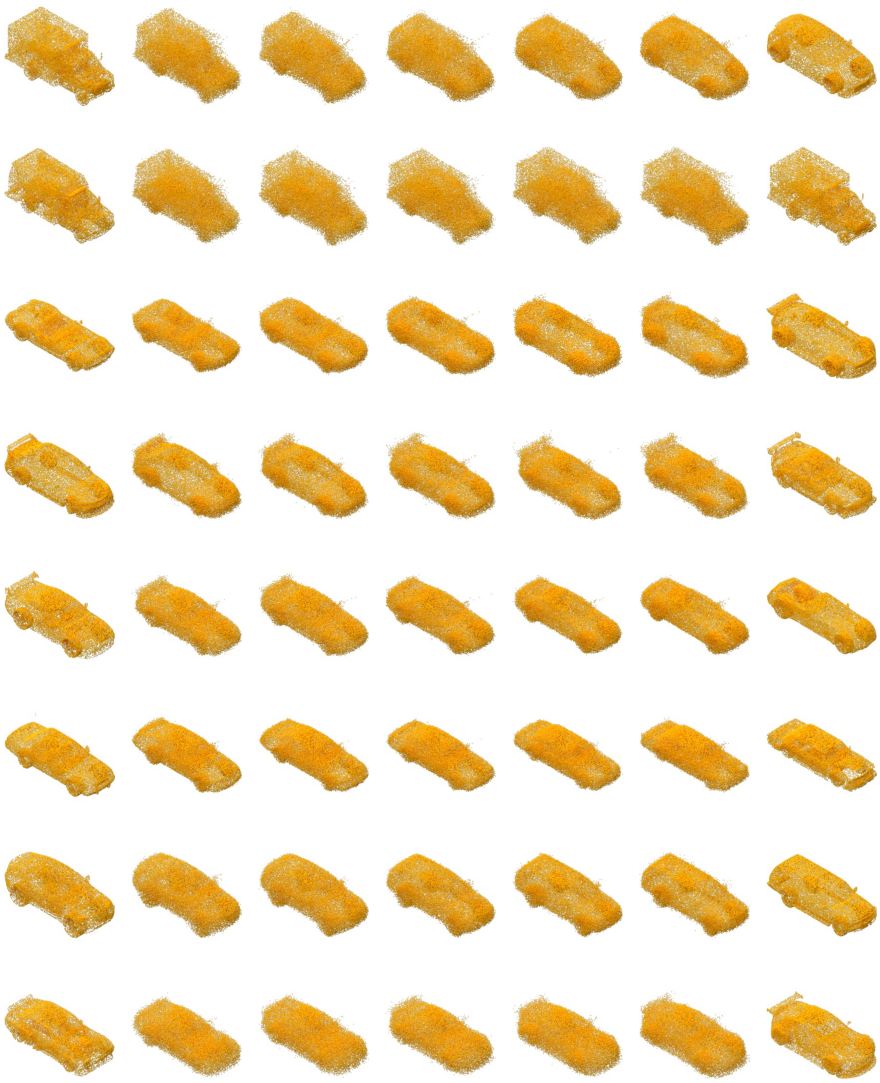

Fig. 12: Interpolations between Car data samples.

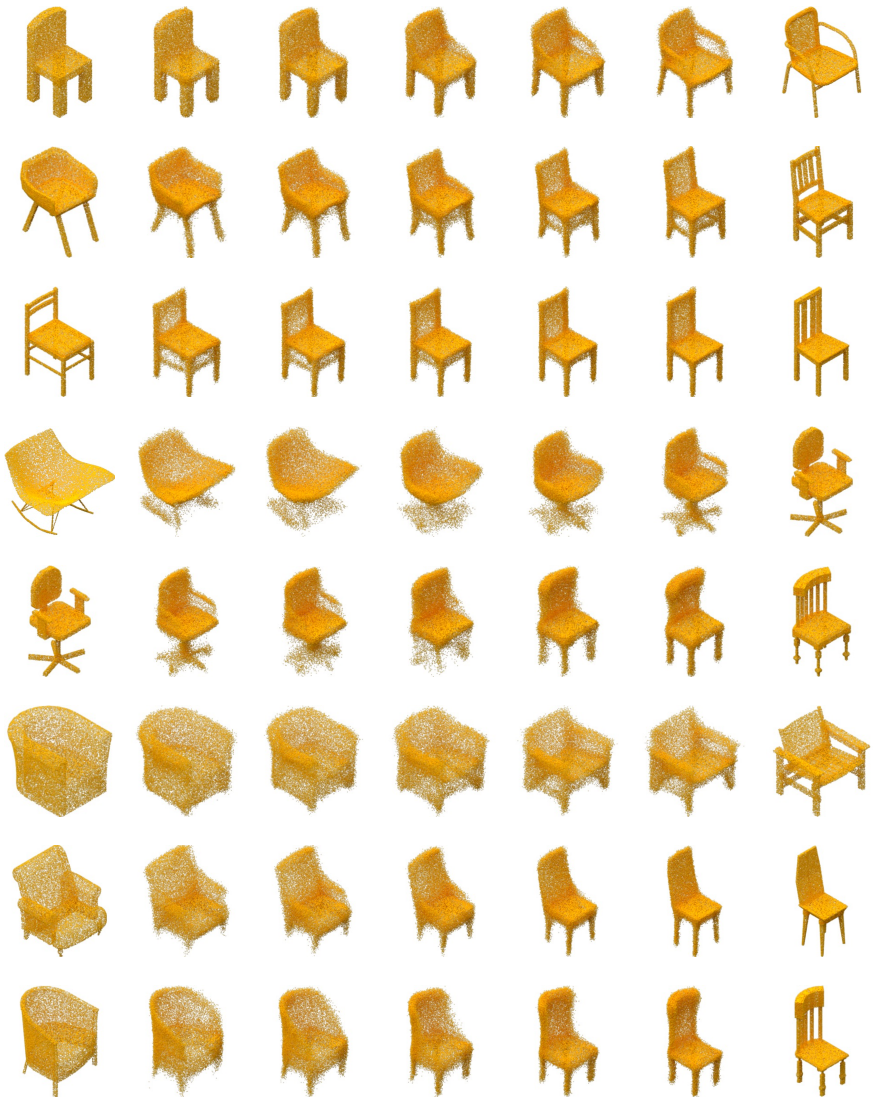

Fig. 13: Interpolations between Chairs data samples.

Fig. 14: Generating flow for sampled point clouds from the Airplane model.

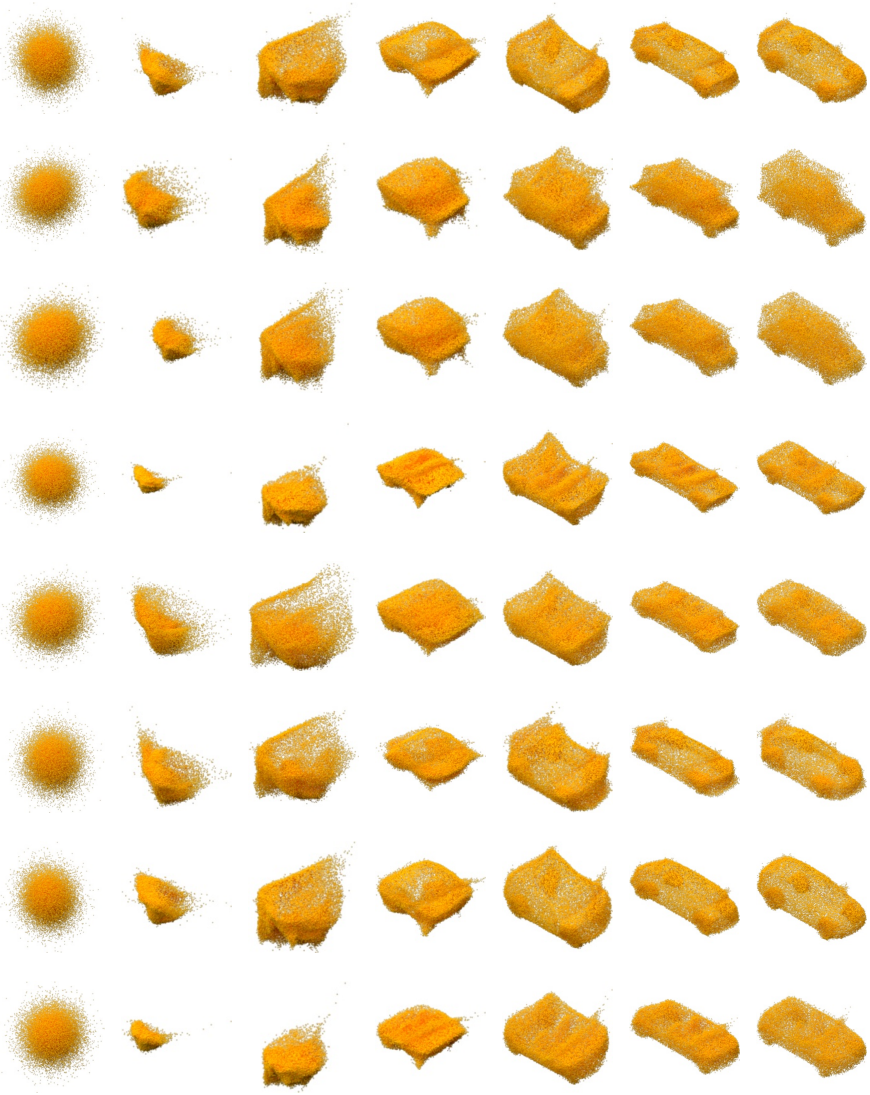

Fig. 15: Generating flow for sampled point clouds from the Car model.

Fig. 16: Generating flow for sampled point clouds from the Chair model.

References

1. Achlioptas, P., Diamanti, O., Mitliagkas, I., Guibas, L.: Learning representations and generative models for 3D point clouds. In: ICML (2018)
2. Behrmann, J., Grathwohl, W., Chen, R., Duvenaud, D., Jacobsen, J.H.: Invertible residual networks. In: ICML (2019)
3. Bhattacharyya, A., Hanselmann, M., Fritz, M., Schiele, B., Straehle, C.N.: Conditional flow variational autoencoders for structured sequence prediction. In: NeurIPS Workshop on Machine Learning for Autonomous Driving (2019)
4. Bishop, C.: Pattern recognition and machine learning. Spinger-Verlag (2006)
5. Brock, A., Lim, T., Ritchie, J., Weston, N.: Generative and discriminative voxel modeling with convolutional neural networks. In: NeurIPS 3D deep learning workshop (2016)
6. Chang, J.R., Chen, Y.S.: Batch-normalized maxout network in network. In: ICML (2016)
7. Chen, T., Rubanova, Y., Bettencourt, J., Duvenaud, D.: Neural ordinary differential equations. In: NeurIPS (2018)
8. Chen, X., Kingma, D., Salimans, T., Duan, Y., Dhariwal, P., Schulman, J., Sutskever, I., Abbeel, P.: Variational lossy autoencoder. In: ICLR (2017)
9. Choy, C., Xu, D., Gwak, J.Y., Chen, K., Savarese, S.: 3D-R2N2: A unified approach for single and multi-view 3D object reconstruction. In: ECCV (2016)
10. Deco, G., Brauer, W.: Higher order statistical decorrelation without information loss. In: NeurIPS (1995)
11. Dinh, L., Sohl-Dickstein, J., Bengio, S.: Density estimation using Real NVP. In: ICLR (2017)
12. Fan, H., Su, H., Guibas, L.: A point set generation network for 3D object reconstruction from a single image. In: CVPR (2017)
13. Girdhar, R., Fouhey, D., Rodriguez, M., Gupta, A.: Learning a predictable and generative vector representation for objects. In: ECCV (2016)
14. Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., Bengio, Y.: Generative adversarial nets. In: NeurIPS (2014)
15. Graham, B., Engelcke, M., van der Maaten, L.: 3D semantic segmentation with submanifold sparse convolutional networks. In: CVPR (2018)
16. Grathwohl, W., Chen, R., Bettencourt, J., Sutskever, I., Duvenaud, D.: FFJORD: Free-form continuous dynamics for scalable reversible generative models. In: ICLR (2019)
17. Groueix, T., Fisher, M., Kim, V., Russell, B., Aubry, M.: A papier-mâché approach to learning 3D surface generation. In: CVPR (2018)
18. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. In: CVPR (2016)
19. He, K., Zhang, X., Ren, S., Sun, J.: Identity mappings in deep residual networks. In: ECCV (2016)
20. Henderson, P., Ferrari, V.: Learning to generate and reconstruct 3D meshes with only 2D supervision. In: BMVC (2018)
21. Insafutdinov, E., Dosovitskiy, A.: Unsupervised learning of shape and pose with differentiable point clouds. In: NeurIPS (2018)
22. Karras, T., Aila, T., Laine, S., Lehtinen, J.: Progressive growing of GANSs for improved quality, stability, and variation. In: ICLR (2018)
23. Kingma, D., Dhariwal, P.: Glow: Generative flow with invertible 1×1 convolutions. In: NeurIPS (2018)

24. Kingma, D., Salimans, T., Jozefowicz, R., Chen, X., Sutskever, I., Welling, M.: Improved variational inference with inverse autoregressive flow. In: NeurIPS (2016)
25. Kingma, D., Welling, M.: Auto-encoding variational Bayes. In: ICLR (2014)
26. Klokov, R., Lempitsky, V.: Escape from cells: Deep Kd-networks for the recognition of 3D point cloud models. In: ICCV (2017)
27. Klokov, R., Verbeek, J., Boyer, E.: Probabilistic reconstruction networks for 3D shape inference from a single image. In: BMVC (2019)
28. Kobyzev, I., Prince, S., Brubaker, M.: Normalizing flows: An introduction and review of current methods. arXiv preprint (2019)
29. Li, C.L., Zaheer, M., Zhang, Y., Poczos, B., Salakhutdinov, R.: Point cloud GAN. arXiv preprint (2018)
30. Loshchilov, I., Hutter, F.: Decoupled weight decay regularization. In: ICLR (2019)
31. Lu, Y., Huang, B.: Structured output learning with conditional generative flows. In: AAAI (2020)
32. Lucas, T., Shmelkov, K., Alahari, K., Schmid, C., Verbeek, J.: Adaptive density estimation for generative models. In: NeurIPS (2019)
33. Mandikal, P., Navaneet, K., Agarwal, M., Babu, R.: 3D-LMNet: Latent embedding matching for accurate and diverse 3D point cloud reconstruction from a single image. In: BMVC (2018)
34. Maturana, D., Scherer, S.: VoxNet: A 3D convolutional neural network for real-time object recognition. In: IROS (2015)
35. Michalkiewicz, M., Pontes, J., Jack, D., Baktashmotlagh, M., Eriksson, A.: Implicit surface representations as layers in neural networks. In: ICCV (2019)
36. Monti, F., Boscaini, D., Masci, J., Rodolà, E., Svoboda, J., Bronstein, M.: Geometric deep learning on graphs and manifolds using mixture model CNNs. In: CVPR (2017)
37. Papamakarios, G., Nalisnick, E., Rezende, D., Mohamed, S., Lakshminarayanan, B.: Normalizing flows for probabilistic modeling and inference. arXiv preprint (2019)
38. Park, J., Florence, P., Straub, J., Newcombe, R., Lovegrove, S.: DeepSDF: Learning continuous signed distance functions for shape representation. In: CVPR (2019)
39. Perez, E., Strub, F., Vries, H.D., Dumoulin, V., Courville, A.: FiLM: Visual reasoning with a general conditioning layer. In: AAAI (2018)
40. Pumarola, A., Popov, S., Moreno-Noguer, F., Ferrari, V.: C-Flow: Conditional generative flow models for images and 3D point clouds. In: CVPR (2020)
41. Qi, C., Su, H., Mo, K., Guibas, L.: Pointnet: Deep learning on point sets for 3D classification and segmentation. In: CVPR (2017)
42. Qi, C., Yi, L., Su, H., Guibas, L.: Pointnet++: Deep hierarchical feature learning on point sets in a metric space. In: NeurIPS (2017)
43. Reddi, S., Kale, S., Kumar, S.: On the convergence of Adam and beyond. In: ICLR (2018)
44. Rezende, D., Mohamed, S.: Variational inference with normalizing flows. In: ICML (2015)
45. Rezende, D., Mohamed, S., Wierstra, D.: Stochastic backpropagation and approximate inference in deep generative models. In: ICML (2014)
46. Riegler, G., Ulusoy, A., Geiger, A.: OctNet: Learning deep 3D representations at high resolutions. In: CVPR (2017)
47. Su, H., Jampani, V., Sun, D., Maji, S., Kalogerakis, E., Yang, M.H., Kautz, J.: SPLATNet: Sparse lattice networks for point cloud processing. In: CVPR (2018)
48. Tatarchenko, M., Dosovitskiy, A., Brox, T.: Octree generating networks: Efficient convolutional architectures for high-resolution 3D outputs. In: ICCV (2017)

49. Verma, N., Boyer, E., Verbeek, J.: FeaStNet: Feature-steered graph convolutions for 3D shape analysis. In: CVPR (2018)
50. Wang, K., Chen, K., Jia, K.: Deep cascade generation on point sets. In: IJCAI (2019)
51. Wang, N., Zhang, Y., Li, Z., Fu, Y., Liu, W., Jiang, Y.: Pixel2Mesh: Generating 3D mesh models from single RGB images. In: ECCV (2018)
52. Wu, J., Zhang, C., Xue, T., Freeman, W., Tenenbaum, J.: Learning a probabilistic latent space of object shapes via 3D generative-adversarial modeling. In: NeurIPS (2016)
53. Wu, Z., Song, S., Khosla, A., Yu, F., Zhang, L., Tang, X., Xiao, J.: 3D ShapeNets: A deep representation for volumetric shapes. In: CVPR (2015)
54. Yang, G., Huang, X., Hao, Z., Liu, M.Y., Belongie, S., Hariharan, B.: PointFlow: 3D point cloud generation with continuous normalizing flows. In: ICCV (2019)
55. Zaheer, M., Kottur, S., Ravanbakhsh, S., Poczos, B., Salakhutdinov, R., Smola, A.: Deep sets. In: NeurIPS (2017)