ChartPointFlow for Topology-Aware 3D Point Cloud Generation

Takumi Kimura
Graduate School of System
Informatics, Kobe University
Kobe, Japan
kimura@ai.cs.kobe-u.ac.jp

Takashi Matsubara
Graduate School of Engineering
Sciences, Osaka University
Toyonaka, Japan
matsubara@sys.es.osaka-u.ac.jp

Kuniaki Uehara
Faculty of Business Administration,
Osaka Gakuin University
Suita, Japan
kuniaki.uehara@ogu.ac.jp

ABSTRACT

A point cloud serves as a representation of the surface of a three-dimensional (3D) shape. Deep generative models have been adapted to model their variations typically using a map from a ball-like set of latent variables. However, previous approaches did not pay much attention to the topological structure of a point cloud, despite that a continuous map cannot express the varying numbers of holes and intersections. Moreover, a point cloud is often composed of multiple subparts, and it is also difficult to express. In this study, we propose ChartPointFlow, a flow-based generative model with multiple latent labels for 3D point clouds. Each label is assigned to points in an unsupervised manner. Then, a map conditioned on a label is assigned to a continuous subset of a point cloud, similar to a chart of a manifold. This enables our proposed model to preserve the topological structure with clear boundaries, whereas previous approaches tend to generate blurry point clouds and fail to generate holes. The experimental results demonstrate that ChartPointFlow achieves state-of-the-art performance in terms of generation and reconstruction compared with other point cloud generators. Moreover, ChartPointFlow divides an object into semantic subparts using charts, and it demonstrates superior performance in case of unsupervised segmentation.

CCS CONCEPTS

• Computing methodologies → Point-based models.

KEYWORDS

point clouds, generative model, manifold

ACM Reference Format:

Takumi Kimura, Takashi Matsubara, and Kuniaki Uehara. 2021. ChartPointFlow for Topology-Aware 3D Point Cloud Generation. In *Proceedings of the 29th ACM International Conference on Multimedia (MM '21), October 20–24, 2021, Virtual Event, China*. ACM, New York, NY, USA, 25 pages. <https://doi.org/10.1145/3474085.3475589>

1 INTRODUCTION

A three-dimensional (3D) point cloud, which is a set of 3D locations in a Euclidean space, has gained popularity as a representation of a geometric shape [25, 34, 35, 40, 41, 44–46, 49, 51] (see the survey [12] for more details). Specifically, the point cloud of an

Permission to make digital or hard copies of part or all of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for third-party components of this work must be honored. For all other uses, contact the owner/author(s).

MM '21, October 20–24, 2021, Virtual Event, China

© 2021 Copyright held by the owner/author(s).

ACM ISBN 978-1-4503-8651-7/21/10.

<https://doi.org/10.1145/3474085.3475589>

Figure 1: Conceptual comparison of existing methods (top) and the proposed method (bottom).

Figure 2: Transformation from simple latent distributions (left) into an object (right). Each row corresponds to a chart.

object's surface is easily acquired using sensors such as LiDARs and Kinects. Point clouds can capture a much higher resolution than voxels, and can be processed using simpler manipulations than meshes. By leveraging the flexibility of deep learning, a deep generative model of point clouds enables a variety of synthesis tasks such as generation, reconstruction, and super-resolution [1, 2, 13, 18, 26, 36, 39, 43, 47]. Because it is difficult to measure the quality of a generated point cloud numerically, most studies employ flow-based generative models [6, 10, 20] or generative adversarial networks (GANs) [9]. These methods learn a map that transforms a latent distribution into an object in the data space, and then they evaluate the object without a heuristic distance.

As a representation of an object's surface, a point cloud often has a thin, circular, or hollow structure [30]. Flow-based generative models encounter a difficulty in expressing such manifold-like structures because a bijective map that is necessary for these models does not exist between a Euclidean space and a manifold with holes, as shown in the top panel of Fig. 1. To express a point cloud X

lying on the one-dimensional (1D) circle S^1 , a map φ modeled using a neural network squashes a two-dimensional (2D) ball in a latent space and stretches it to trace an arc, resulting in a discontinuity and outliers. Several existing methods address a similar issue using a flow on a manifold or a dynamic chart method [29, 38]. However, such methods are applicable only when the geometric property of the target manifold is known and fixed. This assumption does not always hold for point cloud datasets of a variety of shapes. Moreover, a point cloud is often composed of multiple subparts, some of which can be disconnected; additionally, it is difficult to express. This is true for methods based on GANs and autoencoders (AEs) as well, as long as their neural networks are continuous.

Considering these drawbacks, we propose *ChartPointFlow*, a generative model for 3D point clouds with latent labels. Each label is assigned to points in unsupervised manner. Then, a map conditioned on a label is assigned to a continuous subset of a given point cloud, similar to a chart of a manifold, and a set of charts forms an atlas that covers the entire point cloud. Taking Fig. 1 as an example, *ChartPointFlow* with two charts, namely, φ_α and φ_β , generates two arcs separately and concatenates them in the data space, thereby generating a continuous and hollow circle. For a more complex object, each chart is assigned to a semantic subpart, e.g., the airframe, right wing, nose, and left wing of an airplane, as shown in Fig. 2. From the perspective of the generative model, *ChartPointFlow* with n labels provides a mixture of n distributions.

Furthermore, we evaluate *ChartPointFlow* through its performance on synthetic datasets and ShapeNet dataset [4] of point clouds. The experiments demonstrate that *ChartPointFlow* preserves the topological structure in detail, whereas previous approaches tend to generate blurry point clouds and fail to generate holes. Numerical results demonstrate that *ChartPointFlow* outperforms other state-of-the-art point cloud generators, such as r-GAN [1], l-GAN [1], PC-GAN [26], ShapeGF [3], PointFlow [47], SoftFlow [18], AtlasNet [11], AtlasNet V2 [5], tree-GAN [39], and GCN-GAN [43]. In terms of reconstruction and unsupervised semantic segmentation, *ChartPointFlow* outperforms AtlasNet [11] and AtlasNet V2 [5], which are based on AEs and share the concept of charts and atlases.

2 RELATED WORK

Deep Learning on Point Clouds: A point cloud is composed of points in no particular order. PointNet takes each point separately and performs a permutation-invariant operation (max-pooling), thereby obtaining the global feature [34]. Following PointNet, many studies focused on classification and segmentation tasks [25, 34, 35, 40, 41, 44–46, 49, 51].

Likelihood-based Point Cloud Generation: One of the earliest models for point cloud generation is MR-VAE [8], which is based on a variational AE (VAE). A VAE is a probabilistic model that is implemented using two neural networks, namely a decoder that generates a sample and an encoder that performs the variational inference of the latent variable [37]. MR-VAE was trained to minimize a heuristic distance between real and generated point clouds. Zamorski et al. [50] employed an adversarial AE to regularize the latent variables. Liu et al. [27] employed a recurrent neural network to generate a point cloud step-by-step. Instead of an AE, Cai et

al. [3] proposed ShapeGF, which used an implicit function defined using a neural network.

Yang et al. [47] proposed PointFlow, which is a combination of a permutation-invariant encoder and a point-wise flow-based generative model. A flow-based generative model is a neural network that forms a bijective map and obtains a likelihood using the change of variables without a heuristic distance [6, 10, 20]. Moreover, this model can accept and generate an arbitrary number of points.

Because no bijective map exists between manifolds of different topologies, a flow-based generative model tends to be destabilized when modeling zero-width structures, such as a surface. This is often the case with point clouds. Kim et al. [18] proposed SoftFlow to address this issue by adding perturbations to points at the training phase. SoftFlow emphasizes the importance of the topology, but remains inapplicable to general topological structures, such as holes, intersections, and disconnections. *ChartPointFlow* addresses this problem by using charts.

Likelihood-free Point Cloud Generation: Another group of models for point cloud generation involves those based on GANs. A GAN comprises a pair of neural networks, namely, a generator that outputs artificial samples and a discriminator that evaluates their similarity to real samples without a heuristic distance or an explicit likelihood [9]. r-GAN generates all the points of a point cloud simultaneously [1]. l-GAN applies a GAN to the feature vector extracted by a pretrained AE [1]. PC-GAN employs a permutation-invariant generator [26]. Spectral-GAN handles point clouds in the spectral domain [36].

Other GAN-based approaches can be regarded as recursive super-resolutions. Each model first generates a sparse point cloud, and then it adds more points to interpolate the existing ones repeatedly [2, 13, 39, 43]. Valsesia et al. [43] found that the points close to each other have similar feature vectors. Shu et al. [39] also found that each point generated at the first step may be associated with a semantic subpart of the point cloud. These results demonstrate the importance of semantic subparts. However, the above-mentioned studies do not deal with subparts explicitly.

Generative Model with Labels: For modeling samples of multiple categories, deep learning-based generative models have been extended to mixture distributions, such as conditional VAEs [21], conditional GANs [31], and conditional flow-based generative models [6, 20, 24]. The condition represents the class label that an image or object belongs to. In contrast, *ChartPointFlow* divides each point in a single object into a class. As shown in Fig. 1, existing generative models encounter a difficulty in expressing a single cluster if the cluster has a different topology.

AtlasNet [11] and AtlasNet V2 [5] share the concept of charts and atlases with *ChartPointFlow*. However, they assume to express all objects in the same category using a fixed number of fixed-size charts. This assumption is unnatural when the objects' shapes vary widely. For example, the topology of a chair with armrests is different from that of a chair without armrests. In contrast, *ChartPointFlow* resizes charts and discards unnecessary charts by inferring the occurrence probability of each chart from a given object shape. Although AtlasNets aim to reconstruct point clouds, they cannot generate point clouds without modification. AtlasNets are based on ordinary neural networks, which approximate arbitrary functions.

Figure 3: Point clouds generated using the proposed *ChartPointFlow*, *PointFlow* [47], and *SoftFlow* [18]. Color represents the chart that the point belongs to.

ChartPointFlow employs a flow-based generative model, which approximates only bijective functions [42]. Compared with *AtlasNets*, *ChartPointFlow* has an architecture that is more consistent with the definition of charts. Luo and Hu [30] also introduced a similar concept for denoising.

3 BACKGROUND

To clarify the issues with existing methods, this section provides preliminary results. We prepared synthetic datasets, namely, the circle [10], 2sines [18], four-circle [32], and double-moon [6], each of which has only one object X comprising many points $\{x_j\}$, as shown in Fig. 3. The leftmost column shows the datasets. The second and third columns show the generation results of *PointFlow* [47] and *SoftFlow* [18], for which we employed Glow [20] as the backbone. The generated circles, 2sines, and four-circle show the discontinuities and blurred intersections. The generated double-moons show the string-shaped artifacts. Using FFJORD as the backbone, *PointFlow* and *SoftFlow* suppressed the undesired discontinuities for circle and 2sines but not for four-circle (see the fourth and fifth columns). Moreover, they still show the string-shaped artifacts that ruin the desired disconnections. FFJORD is a flow-based generative model inspired by a differential equation, and it learns a bijective map as a vector flow [10]. Because of numerical integration, FFJORD involves significantly high computational costs. *SoftFlow* did not employ FFJORD for 3D point cloud generation.

A flow-based generative model always learns a continuous deformation. Therefore, a generated point cloud X always lies on a connected manifold with no hole, as long as the latent space is a Euclidean space. *PointFlow* and *SoftFlow* squashed 2D latent distributions to express thin structures, stretched them to trace arcs, and failed in expressing holes, intersections, and disconnections. See Appendix A for more details on flow-based models. The same is true for methods based on GANs and AEs because a neural network

is continuous in general. This limitation is more problematic in practical tasks, as demonstrated by the results in Section 6.4. These results motivate this study.

4 FLOW-BASED MODEL WITH CHARTS

Prior to *ChartPointFlow*, we propose a flow-based model with charts, which is a generator of a single point cloud X .

Network Structure: A point generator F is a flow-based generative model of a point $x \in X$ conditioned on a label y . The point generator F conditioned on a label y is regarded as a chart, and a set of n charts forms an atlas that covers the entire point cloud X . The conditional log-likelihood of the point x is obtained using the change of variables, as follows:

$$\log p_F(x|y) = \log p(z) + \log \left| \det \frac{\partial F^{-1}(x;y)}{\partial x} \right|, \quad (1)$$

where z denotes the latent variable $z = F^{-1}(x;y)$, and its prior $p(z)$ denotes the standard Gaussian distribution. One can obtain the marginal log-likelihood $\log p_F(x)$ as the sum of all possible labels $\log p_F(x) = \log \sum_y p_F(x|y)p(y)$. Instead, we employed a variational inference model $q_C(y|x)$, which was implemented as a neural network called a chart predictor C . The evidence lower bound (ELBO) $\mathcal{L}_{ELBO}(F, C; x)$ is then calculated as,

$$\begin{aligned} \log p_F(x) &= \mathbb{E}_{q_C(y|x)} \left[\log \frac{p_F(x,y)}{p_F(y|x)} \frac{q_C(y|x)}{q_C(y|x)} \right] \\ &\geq \mathbb{E}_{q_C(y|x)} \left[\log \frac{p_F(x|y)p(y)}{q_C(y|x)} \right] \\ &= \mathbb{E}_{q_C(y|x)} \left[\log p(z) + \log \left| \det \frac{\partial F^{-1}(x;y)}{\partial x} \right| \right] \\ &\quad - H[q_C(y|x)|p(y)] + H[q_C(y|x)] \\ &=: \mathcal{L}_{ELBO}(F, C; x), \end{aligned} \quad (2)$$

where $H[q_C(y|x)|p(y)]$ and $H[q_C(y|x)]$ denote the cross-entropy and entropy, respectively. We assume that the label prior $p(y)$ is the uniform distribution, which implies that the cross-entropy $H[q_C(y|x)|p(y)]$ has a constant value. We emphasize that the label y is inferred to maximize the ELBO in an unsupervised manner.

Training: The label y is represented by a one-hot vector, and the ELBO $\mathcal{L}_{ELBO}$ is given by the weighted average over all possible labels. This approach requires the computational cost to be proportional to the number of labels. To avoid this issue, we employed the Gumbel-Softmax approach [17]. Specifically,

$$\begin{aligned} \tilde{y} &= \text{softmax}((\log \pi_C(x) + \mathbf{g})/\tau), \\ \mathbf{g} &\sim \text{Gumbel}(0, 1), \end{aligned} \quad (3)$$

where $\mathbf{g}$ denotes a vector, each of whose elements follows the Gumbel distribution $\text{Gumbel}(0, 1)$, $\tau \in (0, \infty)$ denotes the temperature of the softmax function, and $\pi_C(x)$ denotes the vector of the label posterior $q_C(y|x)$, i.e., $(\pi_C(x))_j = q_C(y_j = 1|x)$. This approach allows us to apply the Monte Carlo sampling to the label $\tilde{y}$ in a differentiable manner. One uses a sufficiently small temperature τ , draws an almost one-hot vector $\tilde{y}$, substitutes it into the ELBO $\mathcal{L}_{ELBO}$, and trains neural networks using gradient descent algorithms. The ELBO $\mathcal{L}_{ELBO}$ is approximated as,

$$\begin{aligned} \mathcal{L}_{ELBO}(F, C; x) &\approx \tilde{\mathcal{L}}_{ELBO}(F, C; x) \\ &:= \log p(z) + \log \left| \det \frac{\partial F^{-1}(x;\tilde{y})}{\partial x} \right| - H[q_C(y|x)|p(y)] + H[q_C(y|x)] \end{aligned} \quad (4)$$

Figure 4: Results without (left) and with (right) the regularization term $\mathcal{L}_{MI}(x, y)$. Color represents the chart that the point belongs to.

where the vector $\tilde{y}$ is given by Eq. (3). Owing to the Gumbel-Softmax approach, we emphasize that the computational cost is constant regardless of the number of charts.

When maximizing the approximated ELBO $\tilde{\mathcal{L}}_{ELBO}$, the entropy $H[q_C(y|x)]$ is maximized, resulting in each point belonging to all labels with the same probabilities and the charts overlapping with each other. To assign each chart to a specific connected region of a manifold, i.e., a point cloud, we introduce a regularization term $\mathcal{L}_{MI}(x, y)$, which is based on the mutual information $I(x; y)$, as follows.

$$\begin{aligned} I(y; x) &= H[q_C(y)] - H[q_C(y|x)] \\ &\approx H\left[\frac{1}{|X|} \sum_{\tilde{x} \in X} q_C(y|\tilde{x})\right] - H[q_C(y|x)] \\ &=: \mathcal{L}_{MI}(C; x). \end{aligned} \quad (5)$$

The maximization of the regularization term $\mathcal{L}_{MI}$ cancels out the maximization of the entropy $H[q_C(y|x)]$ in the ELBO $\tilde{\mathcal{L}}(F, C; X)$, and it additionally maximizes the entropy $H[q_C(y)]$. Thus, each sample belongs to only one chart, and all charts are used with uniform probabilities.

For the i.i.d. assumption, the objective function to be maximized for the entire point cloud X is defined using the sum over the points x , as follows.

$$\mathcal{L}(F, C; X, \lambda) = \sum_{x \in X} [\tilde{\mathcal{L}}_{ELBO} + \lambda \mathcal{L}_{MI}], \quad (6)$$

where λ adjusts the regularization term $\mathcal{L}_{MI}(x, y)$.

Experiments on Synthetic Data: As shown in Fig. 3, we conducted preliminary experiments on 2D synthetic datasets to prove the concept of the proposed method. We employed Glow [20] as the backbone of the point generator F . We used $n = 4$ charts for the circle and 2sines datasets, $n = 8$ charts for the four-circle dataset, and $n = 2$ charts for the double-moon dataset. We set λ to 1.1 and τ to 0.1. For other experimental settings, we followed SoftFlow [18], such as Adam optimizer [19] with a batch size of 100 for 36K iterations. Following FFJORD [10], the learning rate was set to 10^{-4} for Glow and to 10^{-3} for FFJORD. After training, each point x was drawn using the point generator F , as follows.

$$x = F(z; y) \text{ for } y \sim p(y) \text{ and } z \sim p(z). \quad (7)$$

PointFlow [47] and SoftFlow [18] were trained under the same experimental settings. Note that the proposed method with only a single chart is the same as PointFlow.

The generated point clouds are summarized in Fig. 3. PointFlow and SoftFlow generated point clouds suffering from discontinuities, blurs, and artifacts, as mentioned in Section 3. In contrast, the proposed method generated a circle without any discontinuity, intersections free from a severe blur, and two arcs clearly separated without any artifacts, even though the backbone was Glow. Color represents the chart that the point belongs to. In the circle,

2sines, and four-circle datasets, subparts are connected smoothly and form the manifold with holes. The intersection is expressed as the intersection of the subparts. In the double-moon, each chart is assigned to one of the arcs exclusively, and thereby expresses the disconnected manifold without artifacts. These results imply that the proposed concept of charts works well for various topological structures, even with the same latent variable distribution $p(z)$.

The left panel of Fig. 4 shows the results without the regularization term $\mathcal{L}_{MI}(x, y)$. Each label is then assigned to the entire point cloud overlapping with each other, and the model generates the discontinuity. This is because the maximization of the entropy $H[q_C(y|x)]$ results in the uniform posterior $q_C(y|x)$, and each label works similarly.

5 CHARTPOINTFLOW

In this section, we extend the model proposed in Section 4 and apply it to 3D point cloud datasets. We name it *ChartPointFlow*. Figure 5 shows a conceptual diagram of ChartPointFlow. We assume that a point cloud dataset $\mathcal{X}$ is composed of N objects $\{X_1, X_2, \dots, X_N\}$, and each object X_i is represented by a cloud of M_i points $\{x_1, x_2, \dots, x_{M_i}\}$.

Network Structure: The feature encoder E is the same as those used in PointFlow [47] and SoftFlow [18]. The feature encoder E is a permutation-invariant neural network that accepts a point cloud X consisting of M points and encodes it to a posterior $q_E(s_X|X)$ of a feature vector s_X using the reparameterization trick [23]. The feature vector s_X is considered a representation of the entire shape of the point cloud X . With a Gaussian prior, the reparameterization trick is known to suffer from posterior collapse, where the output s_X ignores the input X [22, 47]. To make the prior more expressive, the feature encoder E is combined with a flow-based generative model called a prior flow G , which maps the feature vector s_X to the latent variable w . The trainable prior $p_G(s_X)$ of the feature vector s_X is then given by,

$$\log p_G(s_X) = \log p(w) + \int_{t_0}^{t_1} \log \left| \det \frac{\partial G^{-1}(s_X)}{\partial s_X} \right|, \quad (8)$$

where $w = G^{-1}(s_X)$, and the prior $p(w)$ is set to the standard Gaussian distribution. Thereby, the prior flow G learns the distribution of point clouds.

In addition to the architectures in the previous studies, ChartPointFlow has a chart predictor $q_C(y|x, s_X)$, which is introduced in Section 4. The chart predictor $q_C(y|x, s_X)$ is conditioned on the feature vector s_X . It accepts a point $x \in X$ and infers the label y that corresponds to the chart that the point x belongs to. The condition on s_X implies that different point clouds have different atlases, even in the same dataset. For example, points in the same location can be part of the engine or the airframe depending on the airplane's width. Moreover, the posterior of the label $\tilde{y}$ is $q_C(y|X, s_X) = \mathbb{E}_{x \in X} \sum_j q_C(y|x, s_X)$, indicating that the size of each chart depends on point cloud X . A zero posterior implies that the corresponding chart is discarded. In this way, ChartPointFlow differs significantly from AtlasNets, whose charts (patches) have the same size [5, 11].

Figure 5: Architectures and data flow during the training phase (left) and the generation phase (right).

The point generator F was the same as that used in SoftFlow [18] except that ours is conditioned on the label y , whereas that of SoftFlow is conditioned on the injected noise's intensity. The conditional log-likelihood of a point x is given by

$$\log p_F(x|y, s_X) = \log p(z) + \log \left| \det \frac{\partial F^{-1}(x; y, s_X)}{\partial x} \right|. \quad (9)$$

For the generation task, we further propose a neural network called the chart generator K , which accepts a feature vector s_X and gives the posterior $p_K(y|s_X)$ of the label y .

Objective Function: Let Y denote a set of labels, each of which corresponds to a point x of the given point cloud X . Owing to the i.i.d. assumption, $p(Y) = \prod_j p(y_j)$, $q_C(Y|X, s_X) = \prod_j q_C(y_j|X, s_X)$, and $p_F(X|Y, s_X) = \prod_j p_F(x_j|y_j, s_X)$. Given the above, the ELBO $\mathcal{L}_{ELBO}$ is given by,

$$\begin{aligned} \log p(X) &\geq \mathbb{E}_{q_E(s_X|X)} q_C(Y|X, s_X) \left[\log \frac{p_F(X|Y, s_X) p(Y) p_G(s_X)}{q_C(Y|X, s_X) q_E(s_X|X)} \right] \\ &= \mathbb{E}_{q_E(s_X|X)} \left[\sum_j \left\{ \mathbb{E}_{q_C(y_j|X, s_X)} \left[\log p_F(x_j|y_j, s_X) \right] \right. \right. \\ &\quad \left. \left. + H[q_C(y_j|X, s_X)] - H[q_C(y_j|X, s_X)|p(y_j)] \right\} \right] \\ &\quad - D_{KL}(q_E(s_X|X) \| p_G(s_X)) \\ &=: \mathcal{L}_{ELBO}(F, C, E, G; X). \end{aligned} \quad (10)$$

In practice, the expectation $\mathbb{E}_{q_C(y_j|X, s_X)}$ over the inferred label y_j is approximated using the Gumbel-Softmax approach [17] (see Eq. (3)), and the expectation $\mathbb{E}_{q_E(s_X|X)}$ over the feature vector s_X is approximated using Monte Carlo sampling [23]. The approximated ELBO is denoted by $\tilde{\mathcal{L}}_{ELBO}(F, C, E, G; X)$.

The first term of the regularization term in Eq. (5) forces each object to use all charts equivalently. However, each chart may have a different size in practice. To achieve a flexible adjustment, we introduce the coefficient terms μ and λ as

$$\mathcal{L}_{MI}(C; X, \mu, \lambda) := \sum_j \left\{ \mu H \left[\frac{1}{|X|} \sum_{\tilde{x} \in X} q_C(y_j|\tilde{x}) \right] - \lambda H[q_C(y_j|x_j)] \right\}. \quad (11)$$

The objective function to be maximized is given by,

$$\mathcal{L}(F, C, E, G, K; X, \mu, \lambda) = \sum_{X \in \mathcal{X}} [\tilde{\mathcal{L}}_{ELBO} + \mathcal{L}_{MI}]. \quad (12)$$

In addition, the chart generator K is trained separately to estimate the label posterior $q_C(y|X)$ by maximizing the objective function

$$\mathcal{L}_{CP}(K; X) = -\sum_{X \in \mathcal{X}} D_{KL}(p_K(y|s_X) \| q_C(y|X)). \quad (13)$$

Usage and Tasks: For the generation tasks, one can follow the right panel of Fig. 5. First, draw a latent variable w from the prior $p(w)$ and feed it to the prior flow G , obtaining a feature vector $s_X = G(w)$. By feeding the feature vector s_X to the chart generator K , the label posterior $p_K(y|s_X)$ is obtained as a categorical distribution. Repeat the following step M times for M points: draw a label y_j from the posterior $p_K(y|s_X)$ and a latent variable z_j from the prior $p(z_j)$, feed the pair to the point generator F , and obtain a point $x_j = F(z_j; y_j, s_X)$. The set of the obtained points is the generated point cloud X . Formally, $p(X) = \int_{s_X} p_G(s_X) \prod_j \int_{y_j} p_F(x_j|y_j, s_X) p_K(y_j|s_X)$.

For the reconstruction or super-resolution task, feed a given point cloud X to the feature encoder E and obtain a feature vector s_X instead of drawing the feature vector s_X from the prior flow G . Drawing the same number of points is called reconstruction, and adding drawn points to a given point cloud is called super-resolution.

The computational cost of ChartPointFlow is almost the same as that of the comparison methods, PointFlow [47] and SoftFlow [18], when the same backbones are used. Recall that the computational cost of the proposed method is constant regardless of the number of charts owing to the Gumbel-Softmax approach. The chart predictor C is used only during the training phase. The computational cost of the chart generator K is negligible because it is proportional to the number of point clouds (i.e., objects), whereas other components E , F , and C require a computational cost that is proportional to the number of points in all point clouds. In previous studies, PointFlow employed FFJORD as the backbone, but SoftFlow employed Glow. We employed Glow as the backbone of ChartPointFlow. Therefore, its computational cost is at the same level as that of SoftFlow and significantly smaller than that of PointFlow.

Figure 6: Generation examples by ChartPointFlow.

6 EXPERIMENTS AND RESULTS

6.1 Experimental Settings

We evaluated the performance of ChartPointFlow using the Core.v2 of ShapeNet dataset [4]. The dataset is composed of 513,000 unique 3D objects of 55 categories. We selected three different categories: airplane, chair, and car, following Yang et al. [47].

We followed the experimental settings presented in SoftFlow’s release code [18]. We trained 15K epochs in each category using the Adam optimizer [19] with a batch size of 128, an initial learning rate of 2.0×10^{-3} , $\beta_1 = 0.9$, and $\beta_2 = 0.999$. We decayed the learning rate by quarter after every 5K epochs. We obtained $M = 2,048$ points randomly from each object X .

We set $\tau = 0.1$ for the Gumbel-Softmax approach, set $\mu = 0.05$ and $\lambda = 1.0$ for the regularization term $\mathcal{L}_{MI}$, and searched the number n of charts from a range of $\{4, 8, 12, 16, 20, 24, 28, 32\}$. The architectures of the neural networks followed those of SoftFlow [18]. The detailed architectures are summarized in Appendix B.

6.2 Evaluation Metrics

To measure the distance between a pair of point clouds X_1 and X_2 , we employed the earth mover’s distance (EMD) [1, 18, 47]. The EMD is the minimum of the total travel distance of points to deform a point cloud to the other. Specifically, the EMD is defined as

$$EMD(X_1, X_2) = \min_{\phi: X_1 \rightarrow X_2} \sum_{x \in X_1} \|x - \phi(x)\|_2. \quad (14)$$

where both point clouds X_1 and X_2 are composed of the same number of points, ϕ denotes a bijective map from the point cloud X_1 to the other X_2 , and $\|\cdot\|_2$ denotes the Euclidean distance on $\mathbb{R}^3$. While Chamfer distance (CD) has also been used, recent studies have pointed out that it yields misleading results [1]. CD focuses on populated regions (e.g., a chair’s seat cushion) and ignores sparsely placed points (e.g., a chair’s mesh backrest).

To evaluate the similarity between a pair of sets X_1 and X_2 of point clouds, we employed the 1-nearest neighbor accuracy (1-NNA) [18, 28, 47], which aims to evaluate whether two distributions are identical in two-sample tests. 1-NNA is obtained as

$$1\text{-NNA}(X_1, X_2) = \frac{\sum_{X_1 \in \mathcal{X}_1} \mathbb{1}[N_{X_1} \in \mathcal{X}_1] + \sum_{X_2 \in \mathcal{X}_2} \mathbb{1}[N_{X_2} \in \mathcal{X}_2]}{|X_1| + |X_2|}, \quad (15)$$

Model	Airplane	Chair	Car
r-GAN [1]	99.51	99.47	99.86
l-GAN (CD) [1]	97.28	85.27	88.07
l-GAN (EMD) [1]	85.68	65.56	68.32
PC-GAN [26]	92.32	78.37	90.87
ShapeGF [3]	81.44	59.60	60.31
PointFlow [47]	75.06	59.89	62.36
SoftFlow [18]	69.44	63.51	64.71
ChartPointFlow	65.08	58.31	58.68

Table 1: Generation performances. Closer to 50% is better.

where both sets X_1 and X_2 are composed of the same number of point clouds, $N_{X_\bullet}$ denotes the nearest neighbor of $X_\bullet$ in $X_1 \cup X_2 - \{X_\bullet\}$, and $\mathbb{1}[\cdot]$ denotes the indicator function. Roughly speaking, a 1-nearest neighbor classifier classifies a given point cloud X into X_1 or X_2 according to the nearest sample N_X in terms of the EMD. The closer to 50% the accuracy of the 1-NNA is, the more similar the distributions X_1 and X_2 are. Previous studies also used Jensen-Shannon divergence (JSD), minimum matching distance (MMD), and coverage (COV). However, recent studies have revealed that they may give good scores to poor models [18, 47]. For example, JSD gives a good score to a model that generates an average shape without considering individual shapes [47].

6.3 Generation Task

For the generation task, we compared ChartPointFlow with point clouds generators, namely r-GAN [1], l-GAN [1], PC-GAN [26], ShapeGF [3], PointFlow [47], and SoftFlow [18].

For ChartPointFlow, we took the average results of 16 runs to suppress the variance due to the randomness in the generation, and summarized the results in Table 1. The results of ShapeGF were obtained using the official release code¹ under the same experimental settings, and those of the other methods for comparison were obtained from [47] and [18]. The top four methods are based on GANs, ShapeGF is based on the implicit function theorem, and the others are flow-based models. One can see that ChartPointFlow outperforms the other methods in all categories. We provided

¹<https://github.com/RuojinCai/ShapeGF>

Figure 7: Generation examples nearest to the references taken from the datasets.

the results of ChartPointFlow with 28, 20, and 24 charts for the airplane, chair, and car categories, respectively. The results with different numbers of charts are summarized in Appendix C.1. ChartPointFlow achieved state-of-the-art results with 16–28 charts for all categories. The results of the other metrics are summarized in Appendix C.2 just for reference.

Figure 6 shows the samples generated by ChartPointFlow, each of which is composed of 10,000 points. Each protruding subpart of an object, such as the airplane’s horizontal tails, the chair’s legs, and the cars’ wheels, is expressed using a different chart. The same subparts of different objects are expressed by the same charts. The chairs in Results A, C, and D do not have armrests and do not use the charts assigned to the armrests of the chairs in Results B and E (see the red arrows). ChartPointFlow assigned several charts to the chair’s seat and armrests only when needed, thereby expressing the varying topologies. Other results are summarized in Appendix C.3.

For comparison, we took reference samples from the evaluation subsets and chose the nearest samples in terms of EMD from the samples generated by each model, as shown in Fig. 7. We used pretrained models of PointFlow² and SoftFlow³ distributed by the original authors. ChartPointFlow generated samples more similar than others, suggesting that it generated a variety of shapes.

The other GAN-based methods [36, 39, 43] used different experimental settings. Under the same experimental settings, we confirmed that ChartPointFlow outperformed these methods (see Appendix C.4).

²<https://github.com/stevenygd/PointFlow>

³<https://github.com/ANLGBOY/SoftFlow>

Model	Airplane	Chair	Car
ShapeGF [3]	2.55	5.22	4.63
AtlasNet [11]	2.95	6.68	4.75
AtlasNet V2 (PD) [5]	3.28	5.67	4.51
AtlasNet V2 (PT) [5]	3.57	5.97	5.13
PointFlow [47]	2.77	6.42	5.16
SoftFlow [18]	2.60	6.60	5.08
ChartPointFlow	2.23	4.62	3.96

Table 2: Reconstruction errors. Smaller is better.

6.4 Reconstruction Task

For the reconstruction (or super-resolution) task, we measured the EMD between a reference point cloud and a reconstructed one, and summarized the results averaged over five trials (see Table 2). In this section, we used the pretrained model of PointFlow, which was trained only on the reconstruction task, whereas ChartPointFlow and SoftFlow were trained only on the generation task. We also evaluated AtlasNet [11] and AtlasNet V2 (patch deformation (PD) and point translation (PT)) [5] with 25 patches (P25), which are specialized for the reconstruction task. Using the original codes^{4,5}, we trained AtlasNets ourselves under the same experimental settings. We also evaluated ShapeGF [3].

ChartPointFlow outperformed all the comparison methods in all categories. The improvement from the performances of PointFlow and SoftFlow is the most significant for the chair category. This may be because the chair category shows the varying shapes of armrests and legs and the varying number of holes in the backrest, i.e., the varying topologies. Figure 8 shows that ChartPointFlow reconstructed such shapes clearly.⁶ Because of the same reason, AtlasNet V2 outperformed PointFlow and SoftFlow in the chair category, but not in other categories. Moreover, ChartPointFlow reconstructed even the airplane’s front wheel and the car’s mirrors. PointFlow and SoftFlow generate shapes with different topologies only for simple target domains (e.g., 2D synthetic datasets, as shown in Fig. 3), and they suffer from blurs and artifacts in practice. AtlasNet V2 reconstructed objects that are sharper than input objects; in other words, they have difficulty in expressing small subparts with accurate densities. This is because AtlasNet V2 deformed the fixed number of fixed-size 2D patches.

See Appendices C.1 and C.3 for more results.

6.5 Unsupervised Segmentation

ChartPointFlow and AtlasNets assign each point to one of the charts (or patches [5, 11]). This process can be regarded as clustering or unsupervised segmentation. We evaluated the performances of ChartPointFlow and AtlasNets on the unsupervised part segmentation task. The PartDataset of ShapeNet dataset contains labels corresponding to semantic parts for part segmentation, such as wings of an airplane [48]. In particular, each of the three used categories is divided into four parts.

⁴<https://github.com/ThibaultGROUEIX/AtlasNet>

⁵<https://github.com/TheoDEPRELLE/AtlasNetV2>

⁶AtlasNet V2 (PT) deals with a fixed number of points; thus, it is unavailable for reconstruction of a point cloud more dense than that used at the training phase.

Figure 8: Reconstruction examples.

Model	Airplane	Chair	Car
AtlasNet [11]	0.22 / 0.76	0.23 / 0.74	0.11 / 0.71
AtlasNet V2 (PD) [5]	0.25 / 0.79	0.24 / 0.75	0.13 / 0.72
AtlasNet V2 (PT) [5]	0.27 / 0.80	0.24 / 0.74	0.17 / 0.73
ChartPointFlow	0.30 / 0.80	0.35 / 0.86	0.19 / 0.79

Table 3: Segmentation performances (NMI/purity) with 25 clusters. Larger is better.

After training, we fed all the unseen objects to a model to assign points to charts, and we obtained the purity (PUR) and normalized mutual information (NMI). They are defined as,

$$\text{PUR}(Y, \hat{Y}) = \frac{1}{|\hat{Y}|} \sum_l \max_k \#\{y_j | y_j = k \text{ for } j \text{ such that } \hat{y}_j = l\},$$

$$\text{NMI}(Y, \hat{Y}) = \frac{2 I(Y, \hat{Y})}{H(Y) + H(\hat{Y})}, \quad (16)$$

where $y_j \in Y$ and $\hat{y}_j \in \hat{Y}$ denote the ground truth label and the estimated chart of a point $x_j \in X$, respectively. l denotes the l -th cluster estimated by a model, and k denotes the k -th ground truth label.

We evaluated ChartPointFlow and AtlasNets with 25 clusters (called charts or patches), which is the default number for AtlasNets. AtlasNets do not have a chart predictor. Instead, we performed a reconstruction task and a 1-nearest neighbor classification. Specifically, we assigned a given point to the chart that the nearest reconstructed point belongs to. ChartPointFlow outperformed AtlasNets for both criteria in all categories, except for the purity for airplane, as summarized in Table 3. See Appendix C.1 for the results with different numbers of charts. We obtained the results of the part segmentation by assigning a label to each cluster so as to maximize the purity, as shown in Fig. 9. ChartPointFlow segmented the tail wing of an airplane, the legs of a chair, and the wheels of a car more clearly than AtlasNets, which contaminated the leg part of a chair with the seat part and the backrest part. Because AtlasNets employed fixed-size patches, a patch used for a leg of a chair was used for different parts of other chairs when their legs were much smaller.

7 CONCLUSION

In this study, we proposed ChartPointFlow, which is a flow-based generative model of point clouds that employs multiple charts. Each

Figure 9: Results of unsupervised part segmentation.

chart is assigned to a semantic subpart of a point cloud, thereby expressing a variety of shapes with different topologies. Owing to Monte Carlo sampling, the computational cost is of the same order as that of the case without charts. The performance was evaluated using four 2D synthetic datasets and three 3D practical datasets, and the results demonstrated that ChartPointFlow generates various point clouds of various shapes with better accuracies than the comparison methods.

ACKNOWLEDGMENTS

This work was partially supported by the MIC/SCOPE #172107101, JST-CREST (JPMJCR1914), and JSPS KAKENHI (19H04172, 19K20344).

REFERENCES

- [1] Panos Achlioptas, Olga Diamanti, Ioannis Mitliagkas, and Leonidas Guibas. 2018. Learning representations and generative models for 3d point clouds. In *International Conference on Machine Learning (ICML)*.
- [2] M. Arshad and William J. Beksi. 2020. A Progressive Conditional Generative Adversarial Network for Generating Dense and Colored 3D Point Clouds. In *International Conference on 3D Vision (3DV)*.
- [3] Ruojin Cai, Guandao Yang, Hadar Averbuch-Elor, Zekun Hao, Serge Belongie, Noah Snavely, and Bharath Hariharan. 2020. Learning Gradient Fields for Shape Generation. In *European Conference on Computer Vision (ECCV)*.
- [4] 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. 2015. ShapeNet: An information-rich 3d model repository. *arXiv preprint arXiv:1512.03012* (2015).
- [5] Theo Deprelle, Thibault Groueix, Matthew Fisher, Vladimir G. Kim, Bryan C. Russell, and Mathieu Aubry. 2019. Learning elementary structures for 3D shape generation and matching. In *Advances in Neural Information Processing Systems (NeurIPS)*.
- [6] Laurent Dinh, Jascha Sohl-Dickstein, and Samy Bengio. 2017. Density estimation using real NVP. In *International Conference on Learning Representations (ICLR)*.
- [7] Harrison Edwards and Amos Storkey. 2016. Towards a Neural Statistician. In *International Conference on Learning Representations (ICLR)*.
- [8] Matheus Gadelha, Rui Wang, and Subhransu Maji. 2018. Multiresolution Tree Networks for 3D Point Cloud Processing. In *European Conference on Computer Vision (ECCV)*.
- [9] Ian J Goodfellow, Jean Pouget-abadie, Mehdi Mirza, Bing Xu, and David Warde-farley. 2014. Generative Adversarial Nets. In *Advances in Neural Information Processing Systems (NIPS)*.
- [10] Will Grathwohl, Ricky T. Q. Chen, Jesse Bettencourt, Ilya Sutskever, and David Duvenaud. 2019. FJORD: Free-form Continuous Dynamics for Scalable Reversible Generative Models. In *International Conference on Learning Representations (ICLR)*.
- [11] Thibault Groueix, Matthew Fisher, Vladimir G. Kim, Bryan C. Russell, and Mathieu Aubry. 2018. AtlasNet: A Papier-Mâché Approach to Learning 3D Surface Generation. In *Computer Vision and Pattern Recognition (CVPR)*.
- [12] Yulan Guo, Hanyun Wang, Qingyong Hu, Hao Liu, Li Liu, and Mohammed Bennamoun. 2019. Deep learning for 3D point clouds: A survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence* (2019).
- [13] Le Hui, Rui Xu, Jin Xie, Jianjun Qian, and Jian Yang. 2020. Progressive Point Cloud Deconvolution Generation Network. In *European Conference on Computer Vision (ECCV)*.
- [14] M.F. Hutchinson. 1989. A Stochastic Estimator of the Trace of the Influence Matrix for Laplacian Smoothing Splines. *Communications in Statistics - Simulation and Computation* 18 (1989), 1059–1076.
- [15] Sergey Ioffe and Christian Szegedy. 2015. Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift. In *International Conference on Machine Learning (ICML)*.
- [16] Jörn-Henrik Jacobsen, Arnold Smeulders, and Edouard Oyallon. 2018. i-RevNet: Deep Invertible Networks. In *International Conference on Learning Representations (ICLR)*.
- [17] Eric Jang, Shixiang Gu, and Ben Poole. 2017. Categorical reparameterization with gumbel-softmax. In *International Conference on Learning Representations (ICLR)*.
- [18] Hyeonju Kim, Hyeonseung Lee, Woo Hyun Kang, Joun Yeop Lee, and Nam Soo Kim. 2020. SoftFlow: Probabilistic Framework for Normalizing Flow on Manifolds. In *Advances in Neural Information Processing Systems (NeurIPS)*.
- [19] Diederik P. Kingma and Jimmy Ba. 2015. Adam: A Method for Stochastic Optimization. In *International Conference on Learning Representations (ICLR)*.
- [20] Diederik P. Kingma and Prafulla Dhariwal. 2018. Glow: Generative flow with invertible 1x1 convolutions. In *Advances in Neural Information Processing Systems (NeurIPS)*.
- [21] Diederik P. Kingma, Danilo J. Rezende, and Max Welling. 2014. Semi-supervised Learning with Deep Generative Models. In *Advances in Neural Information Processing Systems (NIPS)*.
- [22] Diederik P. Kingma, Tim Salimans, Rafal Jozefowicz, Xi Chen, Ilya Sutskever, and Max Welling. 2016. Improved variational inference with inverse autoregressive flow. In *Advances in Neural Information Processing Systems (NIPS)*.
- [23] Diederik P. Kingma and Max Welling. 2014. Auto-encoding variational bayes. In *International Conference on Learning Representations (ICLR)*.
- [24] Roman Klokov, Edmond Boyer, and Jakob Verbeek. 2020. Discrete Point Flow Networks for Efficient Point Cloud Generation. In *European Conference on Computer Vision (ECCV)*.
- [25] Roman Klokov and Victor Lempitsky. 2017. Escape from Cells: Deep Kd-Networks for the Recognition of 3D Point Cloud Models. In *International Conference on Computer Vision (ICCV)*.
- [26] Chun Liang Li, Manzil Zaheer, Yang Zhang, Barnabás Póczos, and Ruslan Salakhutdinov. 2019. Point cloud gan. In *Deep Generative Models for Highly Structured Data, International Conference on Learning Representations (ICLR) Workshop*.
- [27] Xinhai Liu, Zhizhong Han, Xin Wen, Yu-Shen Liu, and Matthias Zwicker. 2019. L2G Auto-Encoder: Understanding Point Clouds by Local-to-Global Reconstruction with Hierarchical Self-Attention. In *ACM International Conference on Multimedia (MM)*.
- [28] David Lopez-Paz and Maxime Quab. 2017. Revisiting classifier two-sample tests. In *International Conference on Learning Representations (ICLR)*.
- [29] Aaron Lou, Derek Lim, Isay Katsman, Leo Huang, Qingxuan Jiang, Ser-Nam Lim, and Christopher De Sa. 2020. Neural Manifold Ordinary Differential Equations. In *Advances in Neural Information Processing Systems (NeurIPS)*.
- [30] Shitong Luo and Wei Hu. 2020. Differentiable Manifold Reconstruction for Point Cloud Denoising. In *ACM International Conference on Multimedia (MM)*.
- [31] Mehdi Mirza and Simon Osindero. 2014. Conditional Generative Adversarial Nets. *arXiv preprint arXiv:1411.1784* (2014).
- [32] Didrik Nielsen, Priyank Jaini, Emiel Hoogetboom, Ole Winther, and Max Welling. 2020. SurVAE Flows: Surjections to Bridge the Gap between VAEs and Flows. In *Advances in Neural Information Processing Systems (NeurIPS)*.
- [33] George Papamakarios, Theo Pavlakou, and Iain Murray. 2017. Masked Autoregressive Flow for Density Estimation. In *Advances in Neural Information Processing Systems (NIPS)*.
- [34] Charles R. Qi, Hao Su, Kaichun Mo, and Leonidas J. Guibas. 2017. PointNet: Deep learning on point sets for 3D classification and segmentation. In *Computer Vision and Pattern Recognition (CVPR)*.
- [35] Charles R. Qi, Li Yi, Hao Su, and Leonidas J. Guibas. 2017. PointNet++: Deep hierarchical feature learning on point sets in a metric space. In *Advances in Neural Information Processing Systems (NeurIPS)*.
- [36] Sameera Ramasinghe, Salman Khan, Nick Barnes, and Stephen Gould. 2020. Spectral-GANs for High-Resolution 3D Point-cloud Generation. *arXiv preprint arXiv:1912.01800* (2020).
- [37] Danilo Jimenez Rezende and Shakir Mohamed. 2015. Variational inference with normalizing flows. In *International Conference on Machine Learning (ICML)*.
- [38] Danilo Jimenez Rezende, George Papamakarios, Sébastien Racanière, Michael S. Albergo, Gurtej Kanwar, Phiala E. Shanahan, and Kyle Cranmer. 2020. Normalizing Flows on Tori and Spheres. In *International Conference on Machine Learning (ICML)*.
- [39] Dongwook Shu, Sung Woo Park, and Junseok Kwon. 2019. 3D point cloud generative adversarial network based on tree structured graph convolutions. In *International Conference on Computer Vision (ICCV)*.
- [40] Hang Su, Varun Jampani, Deqing Sun, Subhransu Maji, Evangelos Kalogerakis, Ming Hsuan Yang, and Jan Kautz. 2018. SPLATNet: Sparse Lattice Networks for Point Cloud Processing. In *Computer Vision and Pattern Recognition (CVPR)*.
- [41] Xiao Sun, Zhouhui Lian, and Jianguo Xiao. 2019. SRINet: Learning Strictly Rotation-Invariant Representations for Point Cloud Classification and Segmentation. In *ACM International Conference on Multimedia (MM)*.
- [42] Takeshi Teshima, Isao Ishikawa, Koichi Tojo, Kenta Oono, Masahiro Ikeda, and Masashi Sugiyama. 2020. Coupling-based Invertible Neural Networks Are Universal Diffeomorphism Approximators. In *Advances in Neural Information Processing Systems (NeurIPS)*.
- [43] Diego Valsesia, Giulia Fracastoro, and Enrico Magli. 2019. Learning localized generative models for 3D point clouds via graph convolution. In *International Conference on Learning Representations (ICLR)*.
- [44] Lei Wang, Yuchun Huang, Yaolin Hou, Shenman Zhang, and Jie Shan. 2019. Graph Attention Convolution for Point Cloud Semantic Segmentation. In *Computer Vision and Pattern Recognition (CVPR)*.
- [45] Panqu Wang and Ulrich Neumann. 2020. Grid-GCN for Fast and Scalable Point Cloud Learning. In *Computer Vision and Pattern Recognition (CVPR)*.
- [46] Xu Yan, Chaoda Zheng, Zhen Li, Sheng Wang, and Shuguang Cui. 2020. PointASNL: Robust Point Clouds Processing Using Nonlocal Neural Networks With Adaptive Sampling. In *Computer Vision and Pattern Recognition (CVPR)*.
- [47] Guandao Yang, Xun Huang, Zekun Hao, Ming Yu Liu, Serge Belongie, and Bharath Hariharan. 2019. Pointflow: 3D point cloud generation with continuous normalizing flows. In *International Conference on Computer Vision (ICCV)*.
- [48] Li Yi, Vladimir G. Kim, Duygu Ceylan, I-Chao Shen, Mengyan Yan, Hao Su, Cewu Lu, Qixing Huang, Alla Sheffer, and Leonidas Guibas. 2016. A Scalable Active Framework for Region Annotation in 3D Shape Collections. *SIGGRAPH Asia*.
- [49] Manzil Zaheer, Satwik Kottur, Siamak Ravanbakhsh, Barnabás Póczos, Ruslan Salakhutdinov, and Alexander J. Smola. 2017. Deep sets. In *Advances in Neural Information Processing Systems (NeurIPS)*.
- [50] Maciej Zamorski, Maciej Zięba, Piotr Klukowski, Rafał Nowak, Karol Kurach, Wojciech Stokowiec, and Tomasz Trzcinski. 2020. Adversarial autoencoders for compact representations of 3D point clouds. *Computer Vision and Image Understanding (CVIU)* 193 (2020).
- [51] Na Zhao. 2018. End2End Semantic Segmentation for 3D Indoor Scenes. In *ACM International Conference on Multimedia (MM)*.