Conditional Invertible Flow for Point Cloud Generation

Michał Stypułkowski [12†], **Maciej Zamorski** [13], **Maciej Zięba** [13], **Jan Chorowski** [24]

[1] Tooploox
[2] University of Wrocław, Poland
[3] Wrocław University of Science and Technology, Poland
[4] NavAlgo
[†]`michal.stypulkowski@tooploox.com`

Abstract

This paper focuses on a novel generative approach for 3D point clouds that makes use of invertible flow-based models. The main idea of the method is to treat a point cloud as a probability density in 3D space that is modeled using a cloud-specific neural network. To capture the similarities between point clouds we rely on parameter sharing among networks, with each cloud having only a small embedding vector that defines it. We use invertible flows networks to generate the individual point clouds, and to regularize the embedding space. We evaluate the generative capabilities of the model both in qualitative and quantitative manner.

1 Introduction

In this paper we propose a novel method for generating 3D point clouds that utilize the flow-based models. Conceptually, we treat each point in 3D space as an instance in data space and we represent a set of points that constitutes the point cloud using a flow-based neural network. However, to capture the similarities between point clouds, each of the networks shares the parameters and each point cloud is described with an individual embedding that is used as conditional factor in the flow-based model. In order to generate the point cloud we sample an embedding from the assumed prior distribution, then we sample points from the flow assuming given embedding value. The prior distribution for the embedding space is achieved with an additional embedding-level flow.

The proposed model can be interpreted as a meta-learner, where each point cloud is treated as a separate training set that is used to train a single network. However, instead of retraining a model for each point cloud we use an additional embedding space that controls the shape of the point cloud. Harnessing meta-learning techniques enable us to discover the embedding for new point cloud with gradient-based approach by keeping the parameters of flow-based model unchanged. Because of the point-level generative procedure our model is order-invariant and insensitive to the different numbers of points among point clouds.

We identify the following contribution of our work: (i) a novel generative model for 3D point clouds, coherent in terms of mathematics thanks to direct likelihood optimization and change of variable formula, (ii) point-level generative approach that controls the sampling process in every step, and is insensitive to the ordering and different numbers of the points, (iii) a new meta-learning approach, where each point cloud (as separate training subset) is represented by unique embedding that can be parameterized.
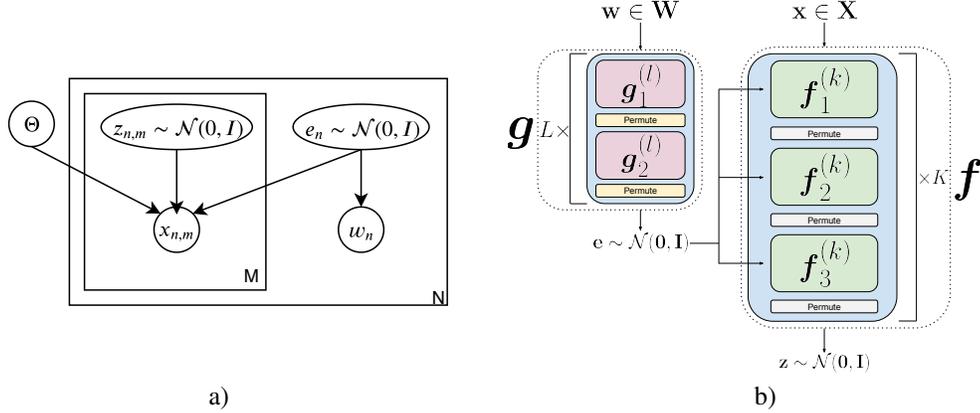
Figure 1: a) Our model generates individual 3D points $\mathbf{x}_{n,m}$ conditioned on a point-cloud embeddings $\mathbf{e}_n$. To regularize the embedding space we introduce additional observed variables $\mathbf{w}_n$ set to the location of the point cloud given by MDS. b) We implement our model using invertible normalizing flows $\mathbf{f}$ and $\mathbf{g}$.

2 Model description

Our model is composed of two invertible normalizing flows, $\mathbf{f}$ and $\mathbf{g}$ (see Figure 1). The model $\mathbf{f}$ is responsible for transforming single-point data space X to the normally distributed Z space. The model $\mathbf{f}$ is conditioned by the latent factor $\mathbf{e}$ that can be seen as embedding representation of the point cloud. The role of $\mathbf{g}$ is to map some observable point cloud representation $\mathbf{w}$ to the normally distributed latent space E to achieve latent point cloud representation from a given prior distribution.

For generative purposes, we first sample point cloud embedding $\mathbf{e}$ from the assumed prior. Next, for the sampled $\mathbf{e}$ we utilize model $\mathbf{f}$ to generate a point cloud by sampling desired number of points $\mathbf{z}$ from Z space and using invertible transformation $\mathbf{f}^{-1}(\mathbf{z}, \mathbf{e})$ to generate points from the data space X.

2.1 Flow-based model

The main idea of flow-based models is to find invertible transformation between some simple distribution p_Z and complex one p_X. We assume that there exists a bijection $\mathbf{f}$ such that $\mathbf{z} = \mathbf{f}(\mathbf{x})$, where $\mathbf{z} \sim p_Z$ and $\mathbf{x} \sim p_X$. We make use of the change of variables formula for density functions

$$p_X(\mathbf{x}) = p_Z(\mathbf{f}(\mathbf{x})) \left| \det \frac{\partial \mathbf{f}(\mathbf{x})}{\partial \mathbf{x}^T} \right|, \tag{1}$$

where $\left| \det \frac{\partial \mathbf{f}(\mathbf{x})}{\partial \mathbf{x}^T} \right|$ is the absolute value of the determinant of the Jacobian of the transformation $\mathbf{f}$ at point $\mathbf{x}$. With proper definition of $\mathbf{f}$ we are able to train model with direct likelihood of the data. We follow [2] in construction of our flows and define $\mathbf{f}$ as combinations of simple flows $\mathbf{f}_i^{(k)}$ each followed by permutation of the dimensions, where $\mathbf{f}_i^{(k)}$ is defined as:

$$\begin{aligned} \mathbf{h}_{1:d} &= \mathbf{x}_{1:d} \\ \mathbf{h}_{d+1:D} &= \mathbf{x}_{d+1:D} \odot \exp(M(\mathbf{x}_{1:d})) + A(\mathbf{x}_{1:d}), \end{aligned} \tag{2}$$

where D is dimension of $\mathbf{x}$ and $d < D$. This transformation is easily invertible and the logarithm of the Jacobian equals $\sum_i M(\mathbf{x}_{1:d})_i$. Note that there are no constraints on either M or A thus we choose them to be neural networks.

2.2 Conditional Flow

Instead of treating $\mathbf{x}$ as a whole point cloud, we take p_X as a distribution of points over the surface of selected point cloud. This specified point cloud is represented by its embedding $\mathbf{e}$. If we add $\mathbf{e}$ as a second argument to $\mathbf{f}$, we are able to choose the distribution from which we want to sample points.

We introduce second flow $\mathbf{g}$ which is responsible for mapping vectors $\mathbf{w}$ into vectors $\mathbf{e} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$. $\mathbf{w}$ is observable representation of single point cloud. It is obtained using Multidimensional Scaling algorithm, which transforms matrix of Chamfer Distances between every pair of point clouds in training set into lower dimensional space W.

Given the data point $\mathbf{x}$ and the corresponding vector $\mathbf{w}$, our main goal is to find the joint distribution $p_{X,W}$. Using the same approach as in the single-flow model and the fact that Z and E are independent, we get

$$p_{X,W}(\mathbf{x}, \mathbf{w}) = p_{Z,E}(\mathbf{f}(\mathbf{x}, \mathbf{e}), \mathbf{g}(\mathbf{w})) \cdot \left| \det \begin{pmatrix} \frac{\partial \mathbf{f}(\mathbf{x},\mathbf{e})}{\partial \mathbf{x}^T} & \frac{\partial \mathbf{f}(\mathbf{x},\mathbf{e})}{\partial \mathbf{w}^T} \\ 0 & \frac{\partial \mathbf{g}(\mathbf{w})}{\partial \mathbf{w}^T} \end{pmatrix} \right| \tag{3}$$

$$= p_Z(\mathbf{f}(\mathbf{x}, \mathbf{e})) \cdot p_E(\mathbf{g}(\mathbf{w})) \cdot \left| \det \frac{\partial \mathbf{f}(\mathbf{x}, \mathbf{e})}{\partial \mathbf{x}^T} \right| \cdot \left| \det \frac{\partial \mathbf{g}(\mathbf{w})}{\partial \mathbf{w}^T} \right|,$$

where $\mathbf{z} = \mathbf{f}(\mathbf{x}, \mathbf{e})$ and $\mathbf{e} = \mathbf{g}(\mathbf{w})$.

The dataset is composed of set of pairs $(\mathbf{x}_{n,m}, \mathbf{w}_n)$, where $\mathbf{x}_{n,m}$ is the m-th point from the n-th point cloud and $\mathbf{w}_n$ is the vector representing the n-th point cloud. We are able to train our model optimizing log-likelihood of the data and the vectors $\mathbf{w}$

$$\sum_n \sum_m \log \big(p_{X,W}(\mathbf{x}_{n,m}, \mathbf{w}_n) \big) = \sum_n \sum_m \Bigg[\log \big(p_Z(\mathbf{f}(\mathbf{x}_{n,m}, \mathbf{e}_n)) \big) + \log \big(p_E(\mathbf{g}(\mathbf{w}_n)) \big) \tag{4}$$

$$+ \log \left| \det \frac{\partial \mathbf{f}(\mathbf{x}_{n,m}, \mathbf{e}_n)}{\partial \mathbf{x}^T} \right| + \log \left| \det \frac{\partial \mathbf{g}(\mathbf{w}_n)}{\partial \mathbf{w}^T} \right| \Bigg].$$

In practice, embedding $\mathbf{e}$ as a second argument of $\mathbf{f}$ is passed to both M and A in every simple flow $\mathbf{f}_i^k$. Due to the fact that $\mathbf{x}$ is 3-dimensional vector, we are able to transform one dimension in every simple flow $\mathbf{f}_i^k$, i.e. d in Eq. 2 is equal to 2. We shift data dimensions one place right after each simple flow to transform all of the dimensions. Inverse function $\mathbf{f}^{-1}$ is defined as inverse function of $\mathbf{f}$ with respect to its first argument, i.e. $\mathbf{z} = \mathbf{f}(\mathbf{x}, \mathbf{e})$ and $\mathbf{x} = \mathbf{f}^{-1}(\mathbf{z}, \mathbf{e})$. We define $\mathbf{g}$ analogously as in 2.1. We split dimensions in half and swap these halves after each $\mathbf{g}_j^l$.

3 Related work

The application of deep generative models to 3D shapes were initially studied in [5] and [4]. The former work focuses on application of Convolutional Deep Belief Network on 3D point grids. The later uses an adversarial training to learn the underlying 3D voxels distribution. In [1] authors provide a two stage approach with simple auto-encoder and GAN model trained in stacked mode. Authors of [7] provide interesting variation of adversarial autoencoder for generating 3D point clouds. In [3] the end-to-end GAN for point clouds is provided. One of the recent papers [6] utilizes VAE together with continuous normalizing flows to generate 3D point clouds.

In contrary to the reference approaches our model operates on single points rather than on entire point clouds, delivers coherent likelihood optimization rather then ELBO and is invariant to the number of the points stored in the point cloud.

4 Experiments

The goal of the experiments is to provide quantitative and qualitative analysis of generative capabilities of the model in terms of the quality of generated samples, interpolation and coverage and minimum matching distance metrics. For all experiments we used point clouds from ShapeNet dataset limited to objects representing chairs and we split the data to train and test sets with 90%-10% proportions.

4.1 Architecture

We implemented both $\mathbf{f}$ and $\mathbf{g}$ as normalizing flows. $\mathbf{f}$ is a combination of 10 segments $\mathbf{f}^{(k)}$ for $k = 1, ..., 10$ with 3 blocks $\mathbf{f}_i^{(k)}$ for $i = 1, 2, 3$ each. Every block is defined by two ResNets. Analogically $\mathbf{g}$ has 5 segments with 2 blocks each. We chose dimensions of $\mathbf{w}$ and $\mathbf{e}$ to equal
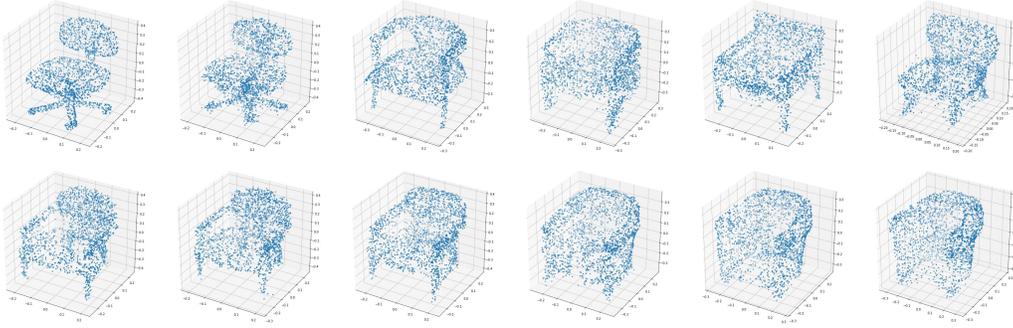
Figure 2: Top row (from the left): chair from the training set and its reconstruction, chair from the test set and its reconstruction, two samples generated randomly from prior distributions. Bottom row: Interpolation between two reconstructed point clouds.

Table 1: MMD and Coverage results compared to reference models

Model	S-CIF	CIF	l-GAN [1]
COV-CD	29.79%	48.23%	59.4%
MMD-CD	0.0025	0.0018	0.0020

64. Priors p_Z and p_E are d-dimensional standard normal distributions, where d equals to 3 and 64 respectively. We used Adam optimizer with default parameters and learning rate 10^{-4} decaying every 40 epochs by factor 0.8.

4.1.1 Results

Qualitative results of reconstruction, sampling and interpolation are shown on Figure 2.

Reconstruction Reconstructions of training set point cloud are made with known embeddings $\mathbf{e}$ (or equivalently $\mathbf{g}(\mathbf{w})$) used during training. This experiment shows $\mathbf{f}$'s ability of sampling points from desired distribution. Generated point clouds are sharp and they are good reconstructions of the training set.

Reconstruction of the test set object requires finding the proper vector $\mathbf{e}$ in the embedding space E. In order to do that, we initialize $\mathbf{e} \sim p_E$ and update it by maximizing the likelihood

$$p_X(\mathbf{x}; \mathbf{e}) = p_Z(\mathbf{f}(\mathbf{x}; \mathbf{e})) \cdot \left| \det \frac{\partial \mathbf{f}(\mathbf{x}; \mathbf{e})}{\partial \mathbf{x}^T} \right|. \tag{5}$$

We perform optimization only with respect to $\mathbf{e}$, having $\mathbf{f}$'s weights frozen. Quality of the reconstruction varies based on the uniqueness of the object. Best results are achieved on simpler shapes but on more complicated ones results are still satisfying.

Sampling To sample a new point cloud, we first sample embedding $\mathbf{e}$ from prior distribution p_E to obtain the conditioning term for $\mathbf{f}$. Next we sample desired number of points $\mathbf{z}$ from p_Z. We pass $\mathbf{e}$ and $\mathbf{z}$ samples to $\mathbf{f}^{-1}$ in order to generate various shapes of chairs.

Interpolation We performed interpolation in embedding space E between two clouds from training set. Results show that embeddings are arranged in natural and logical way. Transitions between each sample is smooth and there are no huge visual differences.

Quantitative We follow evaluations made in [1] to check overall quality of samples and its generalization. We calculated Minimum Matching Distance (MMD) and Coverage (COV) of sampled point clouds with respect to the test set using Chamfer Distance (CD). We report the results in Table 1 comparing our Conditional Invertible Flow (CIF) model with Single Conditional Invertible Flow (S-CIF) model that uses simple MLP instead of $\mathbf{g}$ flow and KL divergence to enforce $\mathbf{e}$ to be normally distributed and with l-GAN [1] trained with CD. Our approach achieves the best results in terms of MMD but l-GAN outperforms our approach with respect to COV. However, no specific model selection was performed with respect to our method on current stage of the development.

4

5 Conclusion and future work

In this paper we provide the novel point-level order-invariant generative model for 3D point clouds that is coherent in terms of mathematics thanks to direct likelihood optimization. We provide some quantitative and qualitative results that are promising for the future works. In next steps we are going to stabilize the training between two flow models, search for better flow-based architectures and make the deeper qualitative analysis with respect to log-likelihood values.

Acknowledgments

References

[1] P. Achlioptas, O. Diamanti, I. Mitliagkas, and L. Guibas. Learning representations and generative models for 3d point clouds. *arXiv preprint arXiv:1707.02392*, 2017.

[2] L. Dinh, J. Sohl-Dickstein, and S. Bengio. Density estimation using real nvp. *arXiv preprint arXiv:1605.08803*, 2016.

[3] C.-L. Li, M. Zaheer, Y. Zhang, B. Poczos, and R. Salakhutdinov. Point cloud gan. *arXiv preprint arXiv:1810.05795*, 2018.

[4] J. Wu, C. Zhang, T. Xue, B. Freeman, and J. Tenenbaum. Learning a probabilistic latent space of object shapes via 3d generative-adversarial modeling. In *Advances in neural information processing systems*, pages 82–90, 2016.

[5] Z. Wu, S. Song, A. Khosla, F. Yu, L. Zhang, X. Tang, and J. Xiao. 3d shapenets: A deep representation for volumetric shapes. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1912–1920, 2015.

[6] G. Yang, X. Huang, Z. Hao, M.-Y. Liu, S. Belongie, and B. Hariharan. Pointflow: 3d point cloud generation with continuous normalizing flows. *arXiv preprint arXiv:1906.12320*, 2019.

[7] M. Zamorski, M. Zieba, P. Klukowski, R. Nowak, K. Kurach, W. Stokowiec, and T. Trzcinski. Adversarial autoencoders for compact representations of 3d point clouds. *arXiv preprint arXiv:1811.07605*, 2018.