# Point Cloud Generation via Variational Auto-encoder

Wen Xiao
Department of Computer Science
University of British Columbia
xiaowen3@cs.ubc.ca

Zhenan Fan
Department of Computer Science
University of British Columbia
zhenanf@cs.ubc.ca

Qiuyan Liu
Department of Computer Science
University of British Columbia
floraliu@cs.ubc.ca

## Abstract

*In this project, the problem of generating point clouds is examined using VAEs. The proposed models use permutation invariant encoder and fully connected layers as decoders. Different loss functions are defined in our models, including Chamfer Distance and Earth Movers Distance. To balance the reconstruction loss and KL divergence, we used the idea of $\beta$-VAE. Then, we generate new samples in three ways, using standard Gaussians, mixture of Gaussians, and directly from the output distribution from encoder. By the result of experiments, we can successfully generate reasonable point cloud instance.*

## 1. Introduction

3D representation of objects in real-life is widely used in computer graphics, and there are many kinds of traditional 3D geometric data, such as view-based projections. Recently, many researchers start working with point cloud data, which represent an object as a set of data points in some coordinate system. Some of them used to transform the point cloud data into 3D voxel grid data or bunch of images, such as VoxNet proposed by Maturana et al. in [9]. And others tried to build deep learning models on point cloud data directly for both classification and segmentation, as in [10][11][12]. In this paper, we propose to build a Variational Auto-Encoder model directly using the point cloud data to generate point cloud representation of objects.

In Achlioptas et al.'s paper[2], they have shown that it is reasonable to generate point cloud from low-dimensional latent space. In our project, we want to use VAE to do the generation. The reason is that ideally VAE can directly learn the distribution of latent variable and compared with GAN, VAEs are usually easier to train.

There are three main challenges of building the generative model for point could data: the first one is that we need to make the model permutation invariant,the second is to find an appropriate loss to measure how similar the generated point cloud and the target point cloud are, and the last one is to balance the reconstruction loss and KL divergence.

Our main contributions are:

- we are the first to apply a simple VAE on point cloud generation task,
- we use the idea from $\beta$-VAE to balance the reconstruction loss and KL divergence,
- instead of sampling from the standard normal distribution, we tried to sample from a mixture of Gaussian.

We will show some related work in this area in Section 2, the architecture and key insights of our model is shown in Section 4. In Section **??**, we will discuss our experiments and results, and Section 7 shows our conclusion.

## 2. Related Work

Recently, there are many works related to 3D shape data. We will focus on those most recent works related to point cloud generation and object recognition.

AtlasNet, which is composed of a union of learnable parametrizations transform a set of 2D squares to the surface [5], is used for learning to generate the surface of 3D shapes. The auto-encoder part in AtlasNet is an encoder based on PointNet [10], where the decoder is 4 fully-connected layers of different sizes with ReLU

non-linearities on the first three layers and tanh on the final output layer.

Based on the Neural Network model for point cloud classification and segmentation PointNet [10], Yaoqing Y. et al. [13] proposed a graph-based enhancement on top of PointNet to promote local structures, where the encoder is a concatenation of multi-layer perceptrons (MLP) and graph-based maxpooling layers. And a novel folding-based decoder is used to deform a canonical 2D grid onto the underlying 3D object surface of a point cloud, which consist of two consecutive 3-layer perceptrons. In practice, in terms of classification accuracy and reconstruction loss, the decoder has better performance in extracting features than the fully connected decoder proposed in [1].

PointNetVLAD is a combination/modification of the existing PointNet [10] and NetVLAD [4], which allows end-to-end training and inference to extract the global descriptor from a given 3D point cloud [3]. The permutation invariant feature of PointNetVLAD ensures the model to output the same global descriptor for a given point cloud regardless of the order in which the points are arranged. It is necessary since the points in point clouds are unordered, which has been shown effective in several supervised and semi-supervised learning [12]. To be specific, the network takes the first part of PointNet, cropped just before the maxpool aggregation layer, feed the output local feature descriptors from PointNet to the NetVLAD layer and finally a fully connected layer to have a compact output feature vector.

However, instead of using permutation equivalent layer, recent study [6] shows a different design - Pointwise Convolutional Neural Network. By inputting points that are sorted in a specific order to the network, the model can still achieve competitive performance in the object recognition task, where the order of the points only affects the final global feature vector used to predict the object category.

For loss functions, Lequan Y. et al. introduces a data-driven point cloud upsampling technique and point out that EMD loss can better capture the object shape than CD [14]. Haque I. et al. proposed that learning features by optimizing a triplet loss on the mean vectors of VAE in conjunction with standard evidence lower bound (ELBO) of VAE allows us to capture more fine-grained information in the latent embedding[8].

As for the point cloud generation, in the paper[2], Achlioptas et al. proposed a new model which is a combination of auto-encoder and GAN. And they have shown that it can generate reasonable point clouds. But they need to first train an auto-encoder, and then train the generator and discriminator at the same time, which takes too much time. So we would like to use a simple VAE model to generate point cloud objects in relatively less time.

## 3. Background on Variational autoencoders

Variational autoencoders (VAEs) were defined in 2013 by Kingma et al. and Rezende et al. A variational autoencoder consists of an encoder, a decoder, and a loss function. The encoder is a neural network. Its input is a datapoint $x$ and it outputs parameters to the probability distribution $q_\theta(z|x)$ of the latent variable $z$, where $\theta$ is the weights and biases. Note that lower-dimensional space is stochastic: we then sample $z$ from $q_\theta(z|x)$.
The decoder is another neural network. Its input is the representation $z$, and it outputs parameters to the probability distribution $p_\phi(x|z)$ of the data, where $\phi$ is the weights and biases. Then we can generate new sample $x^*$ from $p_\phi(x|z)$. The loss function of the variational autoencoder is the negative log-likelihood with a regularizer. Assume we have $N$ data points $x_1, \ldots, x_N$, the loss function $l_i$ is defined to be:

$$l_i(\theta, \phi) = -\mathbb{E}_{z\sim q_\theta(z|x_i)}[\log(p_\phi(x|z))] + D_{KL}(q_\theta(z|x_i)|p(z)) \tag{1}$$

## 4. Model

We are using VAE to do the generation. The full structure of our model is shown in figure1. The details for our model will be described in the following subsections.

### 4.1. Permutation Invariant Encoder

We will first introduce the encoders we used in our model. The key idea is that the encoder should be invariant to input order, which means, no matter how we permute the order of points, the output of encoder should remain the same.Our basic idea comes from Pointnet, which is proposed by Charles R. et al in [10]. The structure is shown in Figure 1, and the main idea in this model is to approximate a general function defined on a point set by applying a symmetric function on transformed elements in the set:

$$f(x_1, \ldots, x_n) = g(h(x_1), \ldots, h(x_n))$$

where $f : \mathbb{R}^3 \times \cdots \times \mathbb{R}^3 \to \mathbb{R}^K$, $h : \mathbb{R}^3 \to \mathbb{R}^M$ and $g : \mathbb{R}^M \times \cdots \times \mathbb{R}^M \to \mathbb{R}^K$ is a symmetric function. In practice, we used 1D convolutional network with filter size 1 as function h, which is equivalant to a 'fully connected layer in 2D'. As for the symmetric function $g$, we chose $maxpool$. Then we use two fully connected layers $fc1$ and $fc2$ to get the mean and variance for the latent Gaussian distribution.

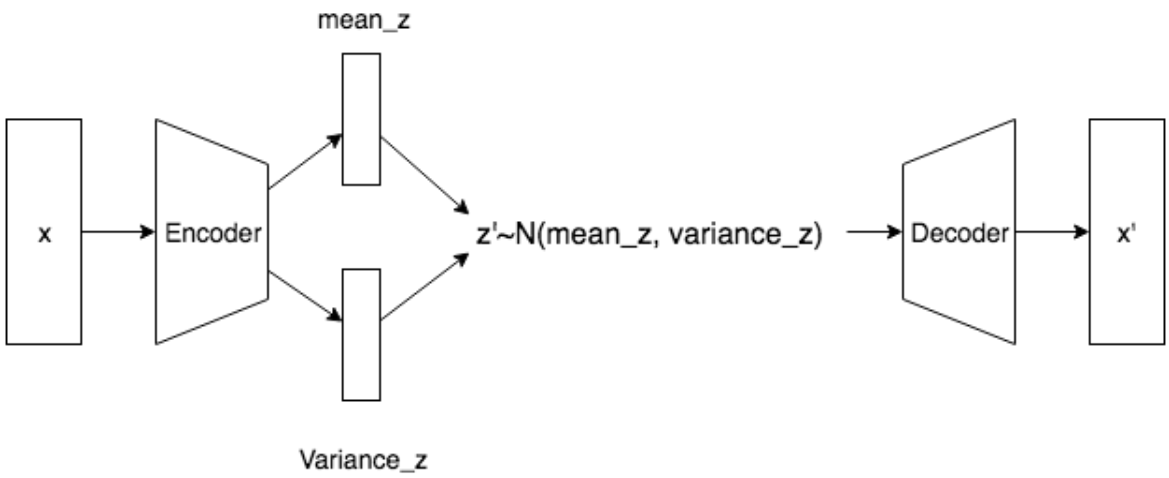


Figure 1. Structure of VAE

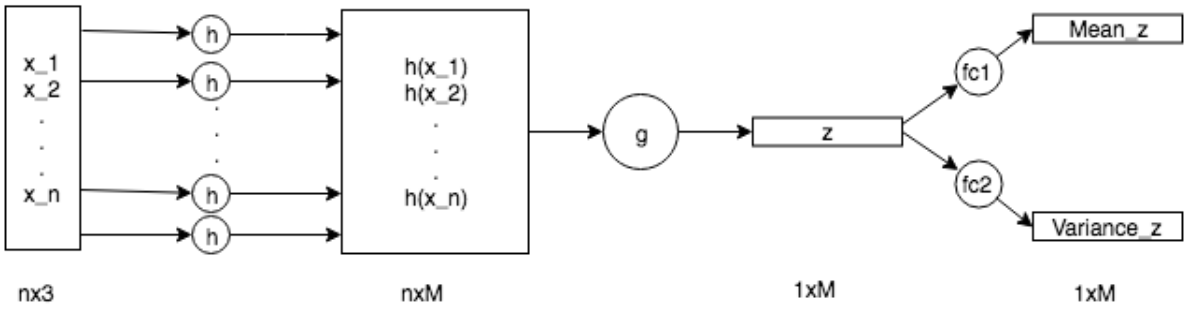


Figure 2. Structure of Permutation Invariant Encoder

### 4.2. Sampling

Let $\mu_z(\mathbf{x})$ and $\Sigma_z(\mathbf{x})$ denote the output from the encoder with input $\mathbf{x}$, then we sample $z^*$ from $q(z|\mathbf{x}) = \mathcal{N}(\mu_z(\mathbf{x}), \Sigma_z(\mathbf{x}))$ and pass it to the decoder.

### 4.3. Decoder

For the decoder, it takes the sampled latent variable $z^*$ as input and the output $\mathbf{x}^*$ is a $n \times 3$ point cloud matrix. Since the loss we use is permutation invariant distance, we don't need to consider the order. In our experiment, we use two fully connected layers with ReLU as activation function. We didn't make the network to be very deep, because many recent papers have observed that VAEs trained with powerful decoders will ignore the latent variables (Chen et al., 2017; Tomczak & Welling, 2017).

### 4.4. Loss function

#### 4.4.1 Chamfer distance

The (extended) Chamfer distance between two sets $S_1, S_2$ of point clouds is defined to be:

$$d_{CD}(S_1, S_2) = \sum_{x \in S_1} \min_{y \in S_2} ||x - y||_2^2 + \sum_{y \in S_2} \min_{x \in S_1} ||x - y||_2^2 \tag{2}$$

For each point, the algorithm of Chamfer distance finds the nearest neighbor in the other set and sums the squared distances up. The advantage of this distance is that it can be easily computed. And since the search for each point is independent, we can calculate it parallel. The disadvantage is that it can not fully capture the difference between two point clouds, since it doesn't take orientation into account. For example, let $A$ be a point cloud of an airplane, and let $A'$ be a copy of $A$ after rotation. Ideally, we want the distance between $A$ and $A'$ to be zero, but the Chamfer distance can be very large.

#### 4.4.2 Earth Movers distance

Consider two sets $S_1, S_2$ of point clouds with equal size, the Earth Movers distance between $S_1$ and $S_2$ is defined to be:

$$d_{EMD}(S_1, S_2) = \min_{\phi: S_1 \to S_2} \sum_{x \in S_1} ||x - \phi(x)||_2 \quad (3)$$

where $\phi$ is a bijection between $S_1$ and $S_2$.
Assume $S_1 = \{x_1, \ldots, x_n\}$ and $S_2 = \{y_1, \ldots, y_n\}$, the EMD distance can be viewed as solving a weighted bipartite matching problem, i.e.

$$\begin{aligned} \text{minimize} \quad & \sum_{i,j=1}^{n} W_{i,j} X_{i,j} \\ \text{subject to} \quad & \sum_{i=1}^{n} X_{i,j} = 1, j = 1, ..., n \\ & \sum_{j=1}^{n} X_{i,j} = 1, i = 1, ..., n \\ & 0 \leq X_{i,j} \leq 1, i, j = 1, ..., n \end{aligned}$$

where $W_{i,j} = ||x_i - y_j||_2$, $\forall i, j$. And by integrality, we know that extreme optimal solution $X$ has entries $X_{i,j} \in \{0, 1\}$. Then we can use many developed algorithms for weighted bipartite matching problem to calculate the Earth Movers distance.
Compared with Chamfer distance, the Earth Mover's distance can describing the difference between two point clouds more accurately, but it also needs more running time.

#### 4.4.3 $\beta$-VAE

For the traditional VAE, the objective(loss) function is defined to be:

$$\mathcal{L}(\mathbf{x}) = -\mathbb{E}_{z \in q_\theta(z|\mathbf{x})}[\log(p(\mathbf{x}|z))] + D_{KL}(q(z|\mathbf{x})|p(z)) \quad (4)$$

where the first term, i.e. the negative log likelihood is also known as the reconstruction loss between $\mathbf{x}$ and $\mathbf{x}^*$. And the second term, i.e. the KL divergence between $q(z|\mathbf{x})$ and $p(z)$ can be viewed as a regularization term.

In the paper [7], they have shown that if we set a specified constraint $\epsilon$ on the regularization term, then the task can be viewed as an optimization problem:

$$\begin{aligned} \text{minimize} \quad & d(\mathbf{x}, \mathbf{x}^*) \\ \text{subject to} \quad & D_{KL}(q(z|\mathbf{x})|p(z)) \leq \epsilon \end{aligned}$$

Rewrite this in the Lagrangian form, we will have:

$$\mathcal{L}(\mathbf{x}) = d(\mathbf{x}, \mathbf{x}^*) + \beta * (D_{KL}(q(z|\mathbf{x})|p(z)) - \epsilon) \quad (5)$$

Here the Lagrangian coefficient $\beta$ controls the degree of applied learning pressure during training, and different $\beta$ can generates different representations.

## 5. Generation

After training, we now have a encoder $E$ and a decoder $D$. Then we use two different methods to generate new samples.

### 5.1. Sample from the output distribution of Encoder

Every time we randomly select $x_i$ from data set $\{x_1, \ldots, x_n\}$. Next, pass it to the encoder and get the latent distribution $q(z|x_i)$. Then, we sample $z^*$ from $q(z|x_i)$ and pass it to the decoder. And the decoder will give us the generated data $x^*$.

### 5.2. Sample from standard Gaussian

Every time we sample $z^*$ from $\mathcal{N}(0, I)$. And then pass it to the decoder, and get the generated new sample $x^* = D(z^*)$.

### 5.3. Sample from mixture of Gaussian

Assume we have $n$ data, $x_1, \ldots, x_n$. After encoding, we will have $n$ different Gaussian distributions, $\mathcal{N}(\mu_1, \Sigma_1), \ldots, \mathcal{N}(\mu_n, \Sigma_n)$. Next, we random pick $m$ Gaussian distribution among them, and sample $\alpha = (\alpha_1, \ldots, \alpha_m) \in \Delta_m$, where $\Delta_m$ is the probability simplex and sample $z_i \in \mathcal{N}(\mu_i, \Sigma_i)$, $\forall i = 1, \ldots, m$. Then we get a new latent variable $z$ as a convex combination of all these $z_i$'s, namely $z^* = \sum_{i=1}^{m} \alpha_i z_i$, and get the generated new sample $x^* = D(z^*)$.

## 6. Experiments

### 6.1. Dataset

We used two datasets for the experiment, both of them are sampled from ShapeNet, and we name them 10000-Point dataset and 2500-Point dataset.
Up to the current stage, we only used airplanes for the experiment. The 10000-Point dataset has only 625 instances of airplane, but it has 10000 points for each instance. While the 2500-Point dataset has around 2800 instances of airplanes, but only 2500 points for each instance. By observation, the 2500-Point dataset is more variant than the 10000-Point dataset. And we showed some instances from the two dataset in Figure 3 and Figure 4. Besides, the 2500-Point dataset takes much less time to train.
To save time, we only trained the final models on 2500-Point dataset, so the results shown below are all from that dataset.

### 6.2. Auto-Encoder

Before we start training the variational auto-encoder model, we trained an auto-encoder to check if the model can

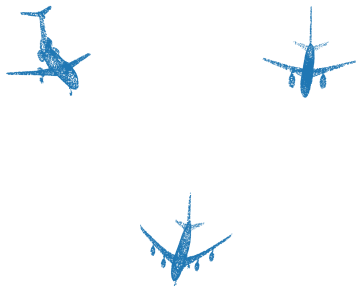

Figure 3. Some instance from 10000-Point dataset

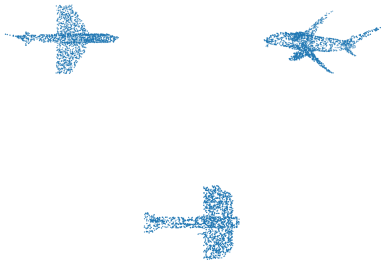

Figure 4. Some instance from 2500-Point dataset

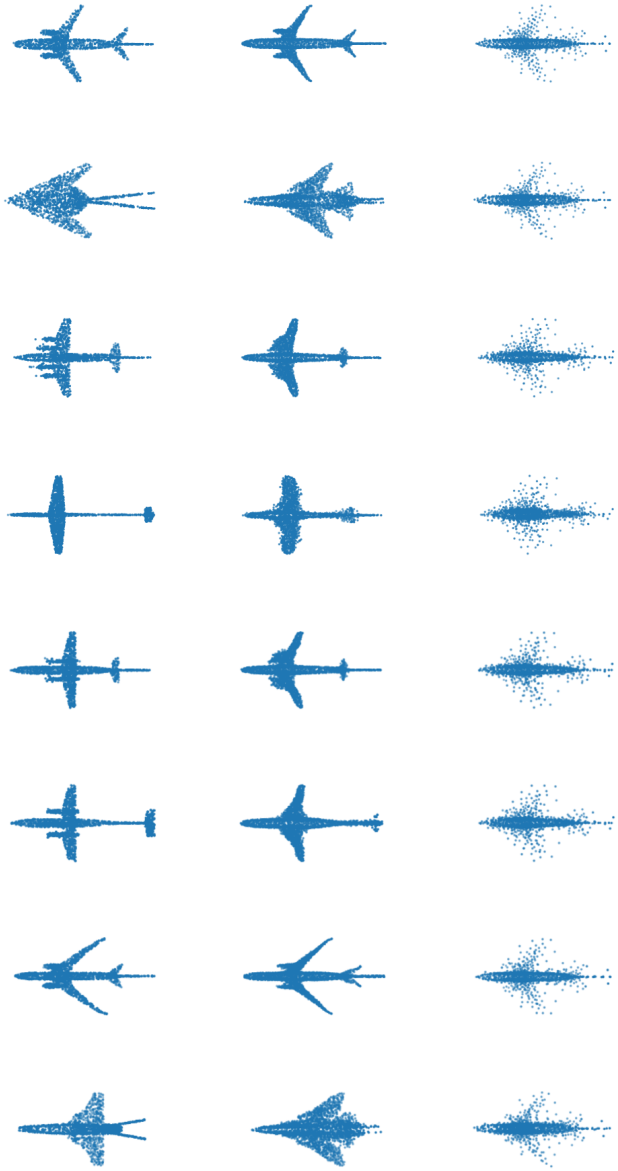

Figure 5. Results of Auto-encoder: the first column is ground truth, the second column is auto-encoder with EMD distance, and the third column is auto-encoder with CD distance, both of the models are trained 100 epochs.

be reconstructed using the current structure and loss function. The only difference here is that we remove the sampling step in our model. For the reconstruction loss, we tried both Chamfer distance and Earth Mover's Distance. Some examples are shown in Figure 5, both of them are trained for 100 epochs and with learning rate $1e-5$. Although CD distance takes less time to compute, it seems like the auto-encoder with the CD distance is harder to converge. So if we train the model for more epochs, it may give us a more accurate reconstruction. But at the current stage, it is obvious that auto-encoder with EMD distance has a better performance than the suto-encoder with CD distance.

### 6.3. Comparison of different sampling method

For the following experiments on sampling, we used the $\beta$-VAE model with EMD loss, $\beta = 10000$ and $\epsilon = 3$, and trained for 500 epochs with learning rate $1e-5$

**Sample $z$ from the output distribution of Encoder** We directly sampled latent vairable $z$ from the output distribution of Encoder, which is also equivalent to sample from the mixture of Gaussian with sample number 1.

**Sample $z$ from standard Gaussian distribution**
We directly sampled latent variable $z$ from the standard normal distribution.

**Sample $z$ from the mixture of Gaussian**
We random pick $N$ point cloud data, and sampled latent variable $z$ based on the method introduced in Section 5.2. Some examples are shown in Figure 6.

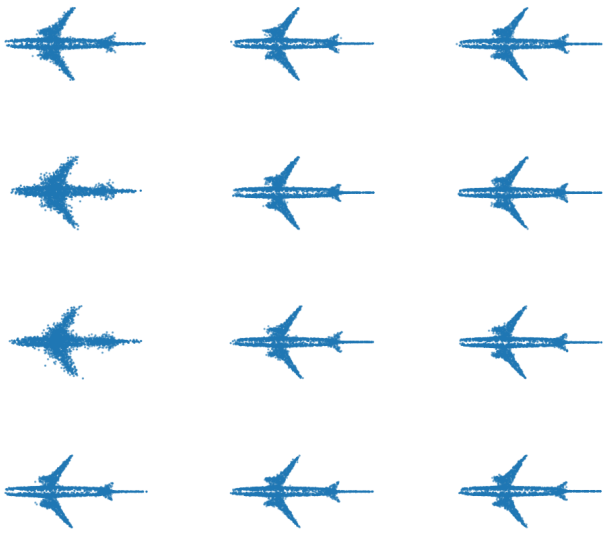

Figure 6. Results of different sampling methods: the first column is sampling from standard Gaussian distribution, the second column is sampling from the output distribution of Encoder, and the third column is sampling from the mixture of Gaussian, both of the models are trained 100 epochs.

As we can see from the graph, the samples generated by sampling $z$ from output distribution of Encoder and mixture of Gaussian have better quality than the ones generated by sampling $z$ from standard Gaussian distribution. This is reasonable, because compared with other two distributions, the standard Gaussian distribution has large variance.

However, all the generated samples are very similar in shape. We think it is probably because the decoder is so powerful that it has already stored critical information for generating the points cloud, which means no matter what inputs we put in, it will always us points clouds with similar shape. And how to improve this is one of our future work.

### 6.4. Comparison of different $\beta$ and $\epsilon$

In this experiment, we want to compare the generated results with different $\beta$ and $\epsilon$. In Figure 7, we show results with different $\epsilon$ and in Figure 8, we show results with different $\beta$.

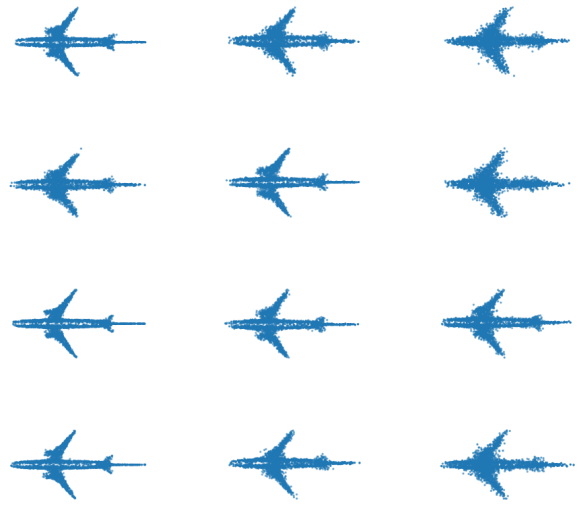

Figure 7. Results of different $\epsilon$ : the first column is $\epsilon = 0$, the second column is $\epsilon = 1$, and the third column is $\epsilon = 3$, all of the models are trained 500 epochs with $\beta = 10000$, and sampled from standard Gaussian distribution.

As we can see from the graph, with $\epsilon$ getting smaller, the quality of generated samples are getting better. It is reasonable because when $\beta \rightarrow$, the -VAE is the same as a strictly constrained optimization problem. And since here we sample $z$ from standard Gaussian distribution, the closer we set $\epsilon$ to 0, the better results we will get. However, it also has the problem that all the generated samples are quite similar.

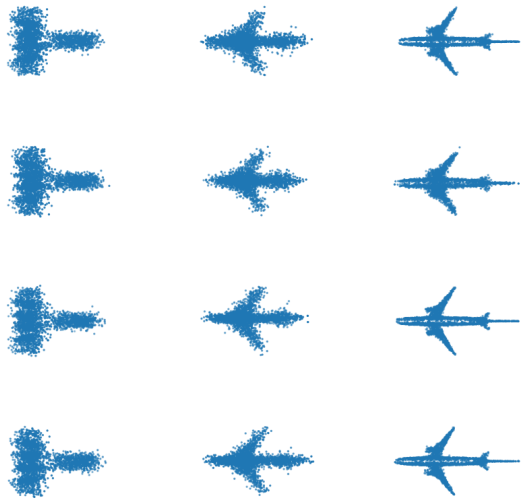

Figure 8. Results of different $\beta$ : the first column is $\beta = 1$, the second column is $\beta = 100$, and the third column is $\beta = 10000$, all of the models are trained 500 epochs with $\epsilon = 0$, and sampled from standard normal distribution.

As we can see from the graph, with $\beta$ getting bigger, the quality of generated samples are getting better. It is reasonable because the $\beta$ actually accounts for the hardness of the constraints. Therefore, when we set it to be big enough, we are actually forcing the output of the encoder to be standard Gaussian, regardless of the input. And this explains why bigger $\beta$ gives better results.

## 7. Conclusion

Based on the results of experiment, we can conclude that VAE does work for the point cloud generation task, but if we want to get better generation sampling from the standard normal distribution, we need to fine tune the hyperparameters to well balance the reconstruction loss and KL divergence. Furthermore, based on the result of experiments, the CD distance takes less time to compute but more epochs to converge, so we may try to train the models with CD distance for more epochs. And also, the generated samples are very similar, which is another problem we need to solve in the future, we may try to use the structure of AE-VAE, which is to generate samples in a latent space.

## References


[1] P. Achlioptas, O. Diamanti, I. Mitliagkas, and L. Guibas. Learning Representations and Generative Models for 3D Point Clouds. *ArXiv e-prints*, July 2017.

[2] P. Achlioptas, O. Diamanti, I. Mitliagkas, and L. J. Guibas. Representation learning and adversarial generation of 3d point clouds. *CoRR*, abs/1707.02392, 2017.

[3] M. Angelina Uy and G. H. Lee. PointNetVLAD: Deep Point Cloud Based Retrieval for Large-Scale Place Recognition. *ArXiv e-prints*, Apr. 2018.

[4] R. Arandjelović, P. Gronat, A. Torii, T. Pajdla, and J. Sivic. NetVLAD: CNN architecture for weakly supervised place recognition. *ArXiv e-prints*, Nov. 2015.

[5] T. Groueix, M. Fisher, V. G. Kim, B. Russell, and M. Aubry. AtlasNet: A Papier-Mâché Approach to Learning 3D Surface Generation. June 2018.

[6] B.-S. Hua, M.-K. Tran, and S.-K. Yeung. Pointwise Convolutional Neural Networks. *ArXiv e-prints*, Dec. 2017.

[7] H. Irina, M. Loic, P. Arka, B. Christopher, G. Xavier, B. Matthew, M. Shakir, and L. Alexander. Beta-VAE: LEARNING BASIC VISUAL CONCEPTS WITH A CONSTRAINED VARIATIONAL FRAMEWORK.

[8] H. Ishfaq, A. Hoogi, and D. Rubin. TVAE: Triplet-Based Variational Autoencoder using Metric Learning. *ArXiv e-prints*, Feb. 2018.

[9] D. Maturana and S. Scherer. Voxnet: A 3d convolutional neural network for real-time object recognition. pages 922–928, 09 2015.

[10] C. R. Qi, H. Su, K. Mo, and L. J. Guibas. Pointnet: Deep learning on point sets for 3d classification and segmentation. *CoRR*, abs/1612.00593, 2016.

[11] C. R. Qi, L. Yi, H. Su, and L. J. Guibas. Pointnet++: Deep hierarchical feature learning on point sets in a metric space. *CoRR*, abs/1706.02413, 2017.

[12] S. Ravanbakhsh, J. Schneider, and B. Poczos. Deep Learning with Sets and Point Clouds. *ArXiv e-prints*, Nov. 2016.

[13] Y. Yang, C. Feng, Y. Shen, and D. Tian. Foldingnet: Interpretable unsupervised learning on 3d point clouds. *CoRR*, abs/1712.07262, 2017.

[14] L. Yu, X. Li, C.-W. Fu, D. Cohen-Or, and P.-A. Heng. PU-Net: Point Cloud Upsampling Network. *ArXiv e-prints*, Jan. 2018.