Generative Models for 3D Point Clouds

Lingjie Kong [*]
Department of Computer Science
Stanford University
ljkong@stanford.edu

Pankaj Rajak
Department of Computer Science
Stanford University
prajak7@stanford.edu

Siamak Shakeri
Department of Computer Science
Stanford University
siamaks@stanford.edu

1 Introduction

Point clouds are rich geometric data structures, where their three dimensional structure offers an excellent domain for understanding the representation learning and generative modeling in 3D space. Learning generative models for point clouds is an emerging field, where the models are trained to be able to generate a set of points in 3D space that resemble the object which is has been previously trained on. Point clouds as a representation are permutation invariant. Any reordering of the rows of the point cloud matrix should yield a point cloud that represents the same shape. This is currently achieved using *maxpooling*, which is a *symmetrictly* permutation invariant function during encoding. However, this ignores point relationships which are achieved similarly through Convolution Neural Networks in image application. Transformer encoders [1], [2] can be used to encode the relationship between points of a point cloud while preserving permutation invariant property. We hypothesize that using such encoder enables capturing more complex point relationships, thus improving reconstruction and training performance of the models. Another issue with current point cloud models is the assumption of approximating the posterior by mixture of Gaussian. This could hinder the generative performance of the model if the true latent-space distribution of the model is non-Gaussian. We propose using normalizing flow models [3] in the latent space by transforming simple Gaussian latent space representations into more complex ones [4].

Another aspect of point clouds that we explore is improving the decoding process from the latent space code. A multi-layer perceptron decoder, such as the one used in [5], might not be powerful enough, as it generates all the point clouds jointly at once. We propose using a progressive decoder model [6]. Through decoding 3D point cloud in a progressive manner, we can generate the points in the point cloud by conditioning on the already generated points. We hypothesize that this would facilitate generating more complex objects.

The rest of this paper is organized as follows: We discuss the related work in the next section, which is followed by problem statement and technical approach. In section 4, the dataset used and the experimental results are mentioned. This section is followed by analysis of the results and future work.

2 Related Works

The initial interest on 3D point cloud was on classification, segmentation, and object detection [7], [8], [9]. In order to make sure the input size is consistent, the input is either chosen from a fixed number of points sampling from a point cloud or Voxel representation. Authors in [10] have proposed

[*]All authors contributed equally

transformation autoencoder networks, where they keep point clouds permutation invariant through maxpooling. Studies are performed on designing more complicated permutation-invariant functions such as deep set [11] with specific neural network architectures. Others have explored spherical harmonic functions [12]. Besides designing more powerful permutation-invariant functions, there has been work done on developing a hybrid GAN-VAE model to better represent the latent space [5]. Flow models have been explored to solve 3D point generation in [4].

The current state of art method [1] to build generative model for point cloud is based on a hybrid approach consisting of Variational autoencoder (VAE) and Generative adversarial network (GAN), where the latent space of VAE is used as an input to the GAN model.

As previously mentioned, encoders used in point cloud deep models should be permutation invariant. Previous works on generative models of point clouds consist of a variant of multi-layer perceptron (MLP) [1], which is shared among all points present inside the point set. Recently, a new method has been proposed to design filters for 3D point clouds, which is not only translation and permutation invariant, but also equivalent to rotation [3]. These filters are based on the product of radial basis functions and spherical harmonics. The parameters of spherical harmonics in these filters captures local neighborhood information of a point in hierarchical manner. For example, these filters can encode radial and angular local neighborhood of each points separately.

3 Problem Statement and Technical Approach

3.1 Definition

A point cloud is a set of points S, where each $s \in R^3$ is a tuple that determines the x,z, and z coordinates of the surface of a 3D object in the Euclidean space. We assume $|S| = N$.

Density Estimation: Given a collection of point clouds $\{S_i\}$ of a certain object, such as airplane, learn a parameterized density function $p_\theta(S)$.

Generation: Create N tuples of point that jointly represent the surface of a 3D object.
Reconstruction: Given a set of N points representing the surface of an object, conditionally generate N tuples that resemble the same object.

Figure 1 depicts all of our proposed improvements to a baseline VAE model when using transformer as encoder, flow model for latent space, and autoregressive decoder in which :

- x is the 3D point clouds input data.
- $q_\phi(z|x)$ is the encoder approximated posterior distribution.
- $p(z)$ is the prior distribution.
- z is the original latent representation from encoder as mixture of Gaussian.
- z_T is the transformed latent representation after applying flow model.
- $p_\theta(x|z_T)$ is the decoder likelihood distribution.
- x' is the output reconstruction/sample 3d point clouds.

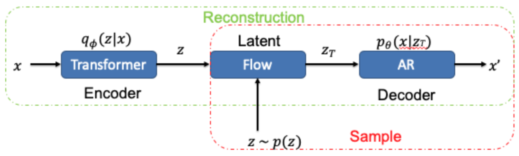


Figure 1: VAE Architecture

3.2 VAE: Variational Autoencoder Model

Baseline VAE which contains 4 conv layers, 1 max pooling, and 3 fully connected layers for the encoder and 3 fully connected layers for the decoder is shown in Figure 2.

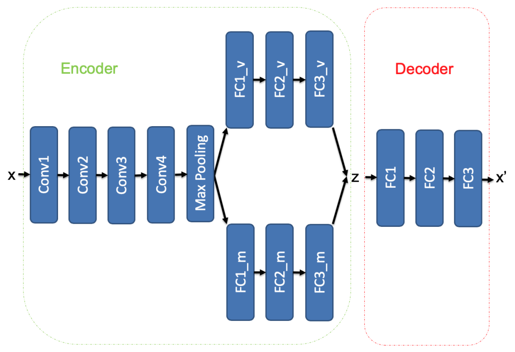


Figure 2: Baseline VAE Network

The prior is modeled as uniform Gaussian $p(\mathbf{z})$

$$p(\mathbf{z}) = \mathcal{N}(\mathbf{x}|0, I)$$

Lower bound can be maximized to the marginal log-likelihood to obtain an expression known as the **evidence lower bound** (ELBO) with encoder network $q_\phi(\mathbf{z}|\mathbf{x})$ and decoder network $p_\theta(\mathbf{x}|\mathbf{z})$.

$$\log p_\theta(\mathbf{x}) \geq \text{ELBO}(\mathbf{x}; \theta, \phi) = \mathcal{L}(\mathbf{x}; \theta, \phi)$$

$$= \mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})}[\log p_\theta(\mathbf{z}, \mathbf{x}) - \log(q_\phi(\mathbf{z}|\mathbf{x}))]$$

$$= \mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})}[\log p_\theta(\mathbf{x}|\mathbf{z})] - D_{\text{KL}}(q_\phi(\mathbf{z}|\mathbf{x})||p(\mathbf{z})))$$

During training, the original 3D points are feed as input for the encoder network to obtain mean $(\mu_\phi(\mathbf{x}))$ and covaraince $(\text{diag}(\sigma^2_\phi(\mathbf{x}))$. Then z is sample as below.

$$q_\phi(\mathbf{z}|\mathbf{x}) = \mathcal{N}(\mathbf{z}|\mu_\phi(\mathbf{x}), \text{diag}(\sigma^2_\phi(\mathbf{x})))$$

Assuming $p_\theta(\mathbf{x}|\mathbf{z})$ is a Gaussian distribution with output as mean and constant covariance, then the reconstruction loss can be written as the chamber's distance loss between the mean of sample $x' \sim p_\theta(\mathbf{x}|\mathbf{z})$ and ground truth x.

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

During Sampling, one can sample from $p(\mathbf{z})$ and feed the sample as input for the decoder network $p_\theta(\mathbf{x}|\mathbf{z})$ to generate 3D points.

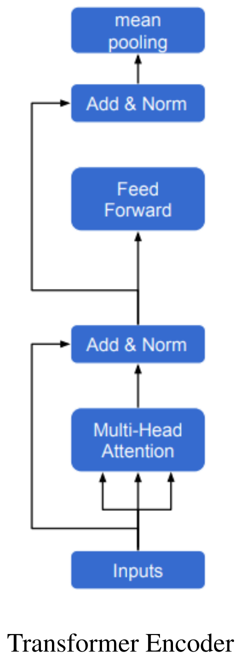
3.3 Transformer Encoder

The baseline VAE uses 1D convolution layers to encode the input points. Therefore, encoding of a point does not depend on the rest of the points in the network. This could potentially degrade the performance of the models by not providing rich encodings.

We propose using self-attention encoder on top of the convolution layers to augment the encoder of the baseline VAE. We use the transformer encoder of [13], which is very popular in natural language processing.

Figure 3 shows the transformer encoder, and its application to the VAE model.

Different from [13], we do not employ positional encoding in the transformer encoder. This is due to the fact that absolute position of points fed to the model does not imply their distance.



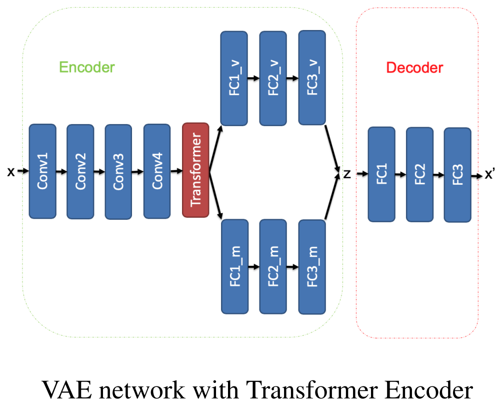


Figure 3: Transformer Encoder design for Point Clouds

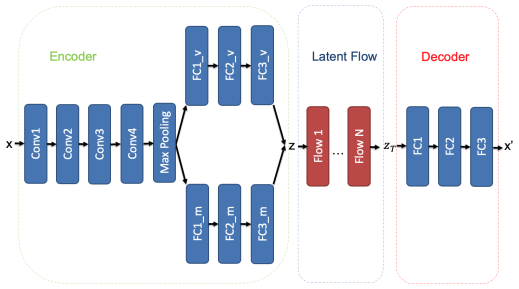


Figure 4: Flow Latent Space VAE Network

3.4 VAE with Flow Latent Space using IAF

An inverse autoregressive flow model (IAF) [14] is used to model the posterior $q(z|X)$, where the output of the encoder is feed into an invertible IAF layer. The output of the IAF layer is given as an input to the decoder. Its structure can be seen in Figure 4.

The VAE loss function using flow in the latent space that is optimized as follows. Recall the ELBO is written as below for baseline VAE.

$$\mathcal{L}(\mathbf{x};\theta,\phi) = \mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})}[\log p_\theta(\mathbf{z},\mathbf{x}) - \log(q_\phi(\mathbf{z}|\mathbf{x}))]$$

For latent flow VAE, the loss function will be.

$$\mathcal{L}(\mathbf{x};\theta,\phi) = \mathbb{E}_{q_\phi(\mathbf{z}_T|\mathbf{x})}[\log p_\theta(\mathbf{z}_T,\mathbf{x}) - \log(q_\phi(\mathbf{z}_T|\mathbf{x}))]$$
$$= \mathbb{E}_{q_\phi(\mathbf{z}_T|\mathbf{x})}[\log p_\theta(\mathbf{x}|\mathbf{z}_T) + \log(p_\theta(\mathbf{z}_T)) - \log(q_\phi(\mathbf{z}_T|\mathbf{x}))]$$

in which the prior $p_\theta(\mathbf{z}_T))$ is modeled by a Gaussian distribution and log posterior $\log(q_\phi(\mathbf{z}_T|\mathbf{x}))$ according to [14] is as below

$$\log(q_\phi(\mathbf{z}_T|\mathbf{x})) = -\sum_{i=1}^{D}(\frac{1}{2}\epsilon_i^2 + \frac{1}{2}\log(2\pi) + \sum_{t=0}^{T}(log(\sigma_t, i)))$$

3.5 VAE with Autoregressive Decoder using NADE

The decoder is modeled in an autoregressive manner following a Neural Autoregressive Distribution Estimation (NADE) [15] structure. Its structure can be shown in Figure 5.

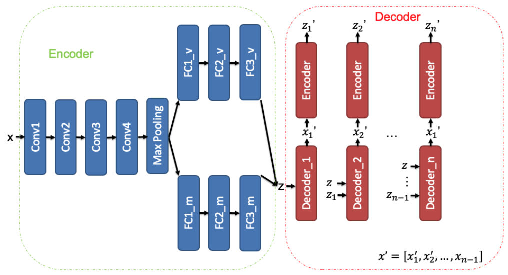


Figure 5: Autoregressive decoder VAE Network

The conditional probability of our decoder can be written as

$$P(X|z) = P_1(x_1|z)P_2(x_2|z, z_{x_1})P_3(x_3|z, z_{x_1}, z_{x_2})...P_n(x_n|z, z_{x_1}, z_{x_2}, ..., z_{x_{n-1}})$$

A point cloud X is partitioned into n groups, in which each conditional probability, $P_n(x_i|z, z_{x_1}, z_{x_2}, ..., z_{x_{i-1}})$ is modeled by a multi-layer perceptron,which joint decode x_i given $z, z_{x_1}, z_{x_2}, ..., z_{x_{i-1}}$. Here, z is sampled from posterior encoder distribution $q(z|X)$. z_{x_i} is m-dimension latent representation of points from group x_i using PointNet architecture,which is shared amoung all groups. For our model, $n = 8$ and a total number of 2048 points is divided into 8 groups of 256 points. Latent space dimension m is 128. Furthermore, the auto regressive decoding of points requires an ordering in which the points needs to generated. Hence, during decoding points are sorted according to one of the x,y,z axis and then partitioned into n groups, where each group is decoded by one NADE block.

3.6 Point Cloud generation using Masked Autoregressive Density Estimation (MADE)

Autoregressive models (AR) offer a flexible way to estimate the density of a data distribution, which can later be used not only to compute log likelihood of new data points but also generate samples from it. However, these models require an ordering of the input data. To build an AR model for point cloud objects, we have first sorted the objects along one of its axes and then used the same axis during the generation. Here, each point is modeled by a Gaussian distribution, where its parameters - μ_i and σ_i- are modeled using 5 MADE layers. The log-likelihood of each object is given as

$$logP(X) = P(x_1)P(x_2|x_1)P(x_3|x_2, x_1)...P(x_n|x_1, x_2, ...x_{n-1})$$

where,

$$P(x_i|x_1, x_2, ...x_{i-1}) = \frac{1}{(2\pi\sigma_i^2)^{\frac{1}{2}}} exp^{(-(x_i - \mu_i)^2/(2\sigma_i^2))}$$

During training, we have also constrained the values of σ_i by adding an L1 regularization such that $\sum \sigma_i$ is minimized as it has shown to increase the sample quality during generation in our. Figure XX shows the samples generated by our MADE model after training for airplane. It can be observed that the quality of the samples generated are not as well as baseline VAE, but they still captures some of the basic features of airplanes such as wings.

As a future research direction our goal is to combine VAE with MADE decoder. Here, the auto-regressive property of MADE will allow tractable computation of P(X|z) using mixture of Gaussian and the latent variable z from VAE will provide global property of the objects to the decoder.

4 Experiment and Result

4.1 Dateset

ShapeNet ([16]) includes a repository of 3D shapes. ShapeNet Core which covers 55 common object categories with about 51,300 unique 3D models are used as the input. Point clouds are created by uniformly sampling from these shapes to convert them to point clouds, as suggested in [5]. This dataset is used for experimentations.

Besides ShapeNet, QM9 dataset ([17]) which is comprised of 134,000 molecules and MD17 molecular dynamics forces ([18]) are other 3D datasets that can be employed in our work.

Another dataset for point clouds is KITTI benchmark suite, which includes 3D Velodyne point clouds.

4.2 Evaluation

One of the challenges when evaluating point clouds is that the metrics need to be permutation invariant. The following distance metrics are proposed in literature for point clouds:

- Earth Mover's Distance(EMD):

$$d_{EMD}(S_1, S_2) = min_{\phi : S_1 \to S_2} \sum_{x_1} ||x - \phi(x)||_2$$

- Chamfer's Distance:

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

Both distance metrics are differentiable. Chamfer's Distance is more computationally efficient than EMD. We have used CD in our work.

[5] proposes a number of evaluation metrics to measure the similarity of two points clouds A and B using the distance metrics mentioned above. They are as follows:

- Jensen-Shannon Divergence: This metric measures how close the points clouds of one set are to the point clouds in another set in the Euclidean 3D space. The closer JSD is to 0, the less divergence there is.
- Coverage: It measures the fraction of points in point cloud B that are closest point to a point in A. This metrics intuitively measures how many points in B are represented in A. The closer coverage is to 1, the better coverage there is.
- Minimum Matching Distance(MMD): This is similar to Coverage, except relying on the average of the minimum distances of matched pairs. The smaller MMD is, the better matching there is.

4.3 Result

- **VAE:** Baseline VAE
- **VAE+Trx: Tr**ansformer Encoder
- **VAE+AR: A**utoregressive **D**ecoder
- **VAE+Flow:** Latent-Space **Flow**

Table 1 Chair, Table 2 Car, and Table 3 airplane show reconstruction and sample results for VAE, VAE+Trx, VAE+Flow, and VAE+AR models separately during both train and test. Baseline VAE model exhibits competitive performance while there is not a specific augmented model consistently out-performing others among all category using JSD, coverage, and MMD as metrics.

Figure 6 shows the reconstruction and ground truth result using airplane and car dataset during training. Baseline VAE model, VAE+Flow, and VAE+AR can reconstruct both the overall shape and details while VAE+Trx reconstructs the overall shape, but fail to reconstruct the tailplane in the airplane dataset.

Table 1: Reconstruction and Generation Evaluation Metrics of Chair

Samples	Model	Split	JSD↓	Coverage↑	MMD↓	NELBO↓	KL↓	Reconst Loss↓
Reconstruction	Train	VAE	0.08127	**0.85061**	**1.56782**	73.32367	17.90308	**55.42059**
		VAE+Trx	0.14206	0.49935	2.22028	102.69266	14.69619	87.99647
		VAE+AR	**0.06992**	0.84866	1.7212	80.10754	18.6372	61.47034
	Test	VAE	0.09089	0.79271	**2.32123**	101.79727	18.02342	**83.77386**
		VAE+Trx	0.15491	0.51025	2.91711	126.22491	14.67373	111.55118
		VAE+AR	**0.07546**	**0.79575**	2.43235	107.27012	18.69086	88.57926
		VAE+Flow	0.10518	0.76765	2.40423	103.23891	17.17888	86.06003
Generation	Train	VAE	0.10043	**0.45642**	3.0537			
		VAE+Trx	0.15628	0.3931	**2.84244**			
		VAE+AR	**0.09164**	0.42736	3.24288			
	Test	VAE	0.10925	**0.43508**	4.11022			
		VAE+Trx	0.16539	0.41989	**3.64259**			
		VAE+AR	**0.10296**	0.40015	4.2808			
		VAE+Flow	0.24696	0.20881	5.38606			

Table 2: Reconstruction and Generation Evaluation Metrics of Car

Samples	Model	Split	JSD↓	Coverage↑	MMD↓	NELBO↓	KL↓	Reconst Loss↓
Reconstruction	Train	VAE	0.05157	**0.40155**	**1.18744**	53.80332	8.42543	**45.37789**
		VAE+Trx	0.06903	0.31448	1.29159	58.98346	7.61572	51.36774
		VAE+AR	**0.04167**	0.3991	1.23474	55.81075	8.33354	47.47721
	Test	VAE	0.05594	0.46733	1.79432	78.51968	9.43197	69.08771
		VAE+Trx	0.09031	0.33807	1.97529	91.12982	7.79197	83.33785
		VAE+AR	**0.047**	**0.47301**	1.85738	79.25379	9.25665	69.99714
		VAE+Flow	0.06515	0.46591	**1.76742**	77.22068	9.42193	**67.79875**
Generation	Train	VAE	0.05342	**0.34418**	**1.36901**			
		VAE+Trx	0.07256	0.29251	1.39957			
		VAE+AR	**0.04651**	0.33157	1.44784			
	Test	VAE	0.07998	0.34801	2.28533			
		VAE+Trx	0.0972	0.31392	2.2712			
		VAE+AR	**0.06792**	**0.34943**	**2.24429**			
		VAE+Flow	0.14089	0.22443	2.79232			

Figure 7 shows the sample test results for chair and table. Baseline VAE model, VAE+Trx, VAE+Flow, VAE+AR can capture the overall shape for both chair and table. However, details such as chair and tables legs cannot be formed clearly during sample.

Figure 8 shows autoregresive decoding result to decode 2048 points for a progressive manner with 256 points per step for 8 steps in total.

Figure 9 shows the VAE model compared to the MADE. Unlike VAE models trained with 2048 points, MADE models are trained with 512 points to speed up the speed.

5 Analysis

- Baseline VAE model exhibits competitive performance versus the augmented models.
- Baseline VAE performs better on most metrics on training set versus the test set. This could be due to overfitting to the training set. Therefore, we need to implement better regularization techniques. It is even more evident by looking at the generative metrics on the test set, which shows more degradation than reconstruction metrics on the same test set.
- Transformer Encoder in general under-performs the rest of the models. One reason is that we currently apply self-attention across all of the input points. This might not be the best

Table 3: Reconstruction and Generation Evaluation Metrics of Airplane

Samples	Model	Split	JSD↓	Coverage↑	MMD↓	NELBO↓	KL↓	Reconst Loss↓
Reconstruction	Train	VAE	**0.03264**	**0.661**	**0.38827**	39.61	12.09	**27.52**
		VAE+Trx	0.05587	0.53355	0.48241	46.057	10.53	35.52
		VAE+AR	0.08704	0.59887	0.51608	48.90747	11.68637	37.22
	Test	VAE	**0.03775**	**0.56436**	**1.14429**	97.01	14.52	**82.49**
		VAE+Trx	0.0791	0.40842	1.38352	142.63245	11.00	131.63
		VAE+AR	0.08432	0.55446	1.15334	97.82669	14.14	83.68
		VAE+Flow	0.03858	0.54703	1.15739	97.73643	14.13	83.59
Generation	Train	VAE	**0.03899**	**0.4541**	0.7485			
		VAE+Trx	0.0616	0.441	**0.68**			
		VAE+AR	0.10037	0.411	0.80			
	Test	VAE	**0.08838**	**0.36634**	1.71			
		VAE+Trx	0.10734	**0.36634**	**1.62**			
		VAE+AR	0.15394	0.34035	1.76			
		VAE+Flow	0.14517	0.25	2.20			

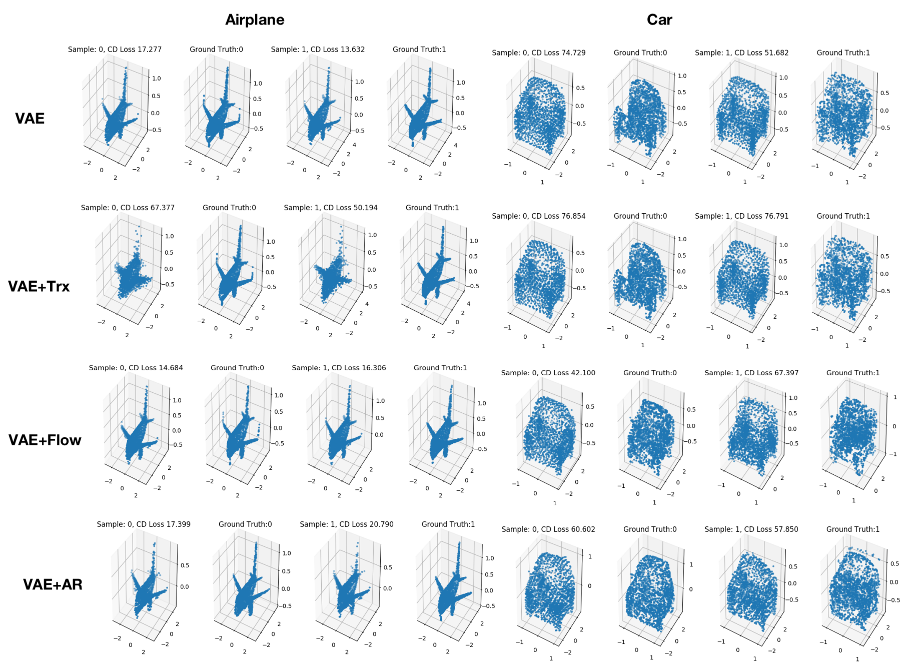


Figure 6: Reconstruction Result

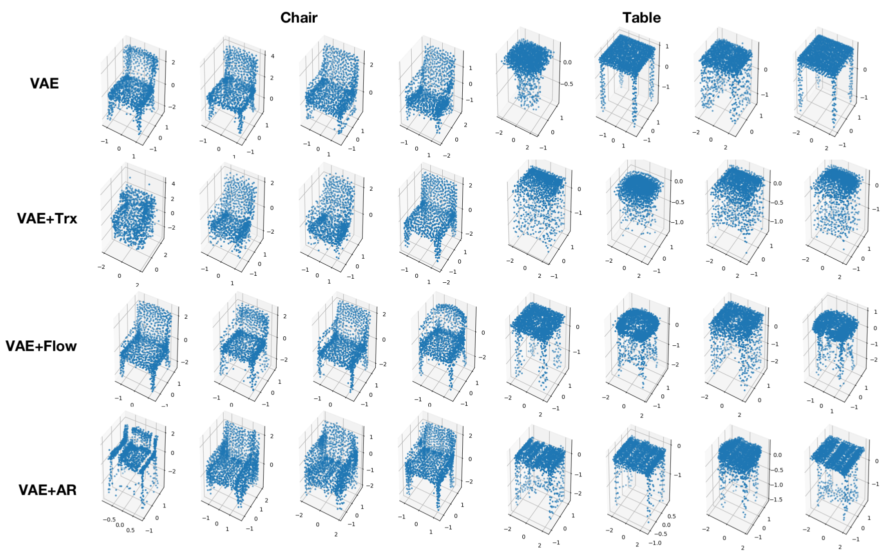


Figure 7: Sample Result

strategy because a point is usually only strong correlated to its neighbours instead of all points.

- Autoregressive decoding improves the generative performance w.r.t to **JSD**. We currently generate **256** points in each progressive step along a specific order. Better tuning this hyper-parameter of number of points per group and specific decoding order, it could better exhibit the merits of this design.
- Latent Space Flow model has competitive *reconstruction* performance, however its *generative* metrics are low. We believe the optimization should be improved for this model, as its objective is different from the rest of models.

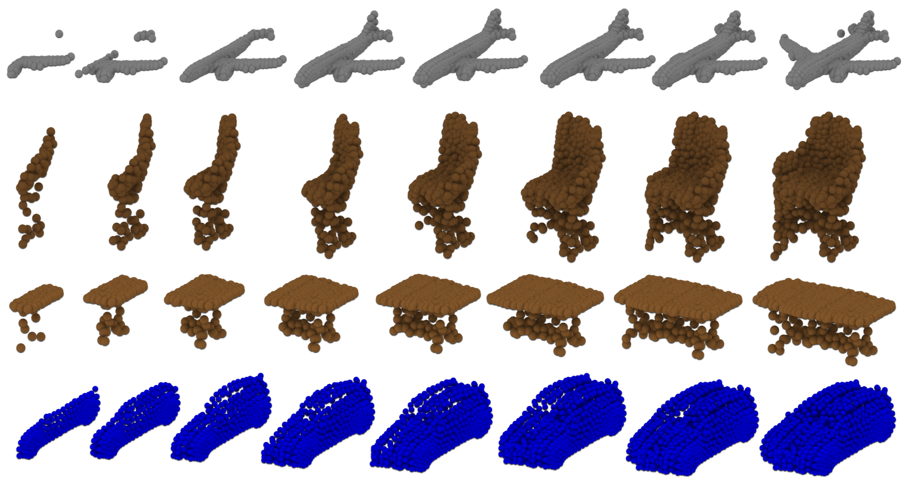


Figure 8: Autoregressive Decoding Result

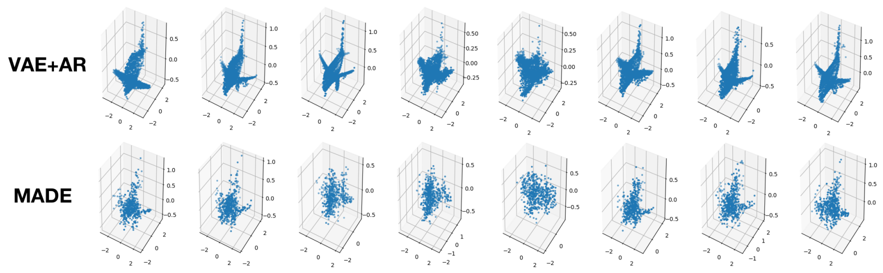


Figure 9: VAE vs. MADE (VAE models are trained with 2048 points whereas MADE models are trained with 512 points)

- Comparing the metrics result among different dataset, it can be observed that chair > airplane > car. One reason can be that car category has more unique 3D shapes. Therefore, if mode collapses into a single one during training, it will definitely perform worse compared to the rest of the dataset during test sample.
- There is not a model that perform consistently well across all. One reason can be that each object has its own unique geometric structure, which is better utilized by one of the models.

6 Conclusion

In this project, we proposed using various ideas to improve the generation and reconstruction performance of VAE models for 3D point clouds. Encoder augmentation by using transformer encoder, latent representation improvement by normalizing flow, and autoregressive decoding were experimented with. We showed that using such techniques could improve the generation and reconstruction performance of the vanilla VAE in some of the metrics. We are planning to continue this work by following the below ideas:

- For encoder, context aware transformer should be used as encoder where each point will look only into its k-nearest neighbors to learn its feature.
- For flow, different number of flow layers should be taken into consideration as well as other flow models beside IAF.
- For decoder, we should experiment with different number of points per decoding step for the NADE architecture, and also model the decoder using MADE, where each output node of MADE will represents a single point.

- Combining all encoder, latent space, and decoder augmentations.

We would like to thank Yang Song, who advised us in this project. We certainly used his insights, and hope to continue to do so. Our code is available at https://github.com/lingjiekong/CS236Project.

References

[1] Wentao Yuan, David Held, Christoph Mertz, and Martial Hebert. Iterative transformer network for 3d point cloud. *arXiv preprint arXiv:1811.11209*, 2018.

[2] Jiancheng Yang, Qiang Zhang, Bingbing Ni, Linguo Li, Jinxian Liu, Mengdie Zhou, and Qi Tian. Modeling point clouds with self-attention and gumbel subset sampling. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3323–3332, 2019.

[3] Danilo Jimenez Rezende and Shakir Mohamed. Variational inference with normalizing flows. *ArXiv*, abs/1505.05770, 2015.

[4] Guandao Yang, Xun Huang, Zekun Hao, Ming-Yu Liu, Serge Belongie, and Bharath Hariharan. Pointflow: 3d point cloud generation with continuous normalizing flows. *arXiv preprint arXiv:1906.12320*, 2019.

[5] Panos Achlioptas, Olga Diamanti, Ioannis Mitliagkas, and Leonidas Guibas. Learning representations and generative models for 3d point clouds. *arXiv preprint arXiv:1707.02392*, 2017.

[6] Ari Heljakka, Arno Solin, and Juho Kannala. Pioneer networks: Progressively growing generative autoencoder. In *Asian Conference on Computer Vision*, pages 22–38. Springer, 2018.

[7] Charles R Qi, Hao Su, Kaichun Mo, and Leonidas J Guibas. Pointnet: Deep learning on point sets for 3d classification and segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 652–660, 2017.

[8] Charles Ruizhongtai Qi, Li Yi, Hao Su, and Leonidas J Guibas. Pointnet++: Deep hierarchical feature learning on point sets in a metric space. In *Advances in neural information processing systems*, pages 5099–5108, 2017.

[9] Yin Zhou and Oncel Tuzel. Voxelnet: End-to-end learning for point cloud based 3d object detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4490–4499, 2018.

[10] Junier B Oliva, Avinava Dubey, Manzil Zaheer, Barnabás Póczos, Ruslan Salakhutdinov, Eric P Xing, and Jeff Schneider. Transformation autoregressive networks. *arXiv preprint arXiv:1801.09819*, 2018.

[11] Manzil Zaheer, Satwik Kottur, Siamak Ravanbakhsh, Barnabas Poczos, Ruslan R Salakhutdinov, and Alexander J Smola. Deep sets. In *Advances in neural information processing systems*, pages 3391–3401, 2017.

[12] Nathaniel Thomas, Tess Smidt, Steven Kearnes, Lusann Yang, Li Li, Kai Kohlhoff, and Patrick Riley. Tensor field networks: Rotation-and translation-equivariant neural networks for 3d point clouds. *arXiv preprint arXiv:1802.08219*, 2018.

[13] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in neural information processing systems*, pages 5998–6008, 2017.

[14] Durk P Kingma, Tim Salimans, Rafal Jozefowicz, Xi Chen, Ilya Sutskever, and Max Welling. Improved variational inference with inverse autoregressive flow. In *Advances in neural information processing systems*, pages 4743–4751, 2016.

[15] Benigno Uria, Marc-Alexandre Côté, Karol Gregor, Iain Murray, and Hugo Larochelle. Neural autoregressive distribution estimation. *The Journal of Machine Learning Research*, 17(1):7184–7220, 2016.

[16] Angel X. Chang, Thomas A. Funkhouser, Leonidas J. Guibas, Pat Hanrahan, Qi-Xing Huang, Zimo Li, Silvio Savarese, Manolis Savva, Shuran Song, Hao Su, Jianxiong Xiao, Li Yi, and Fisher Yu. Shapenet: An information-rich 3d model repository. *CoRR*, abs/1512.03012, 2015. URL `http://arxiv.org/abs/1512.03012`.

[17] Raghunathan Ramakrishnan, Pavlo O. Dral, Matthias Rupp, and O Anatole von Lilienfeld. Quantum chemistry structures and properties of 134 kilo molecules. In *Scientific data*, 2014.

[18] Stefan Chmiela, Alexandre Tkatchenko, Huziel E. Sauceda, Igor Poltavsky, Kristof T. Schütt, and Klaus-Robert Müller. Machine learning of accurate energy-conserving molecular force fields. *Science Advances*, 3(5), 2017. doi: 10.1126/sciadv.1603015. URL `https://advances.sciencemag.org/content/3/5/e1603015`.