Generative Shape Synthesis with Variational Auto-Decoder

Baris Zöngür
baris.zoenguer@tum.de

Munzer Dwedari
munzer.dwedari@tum.de

Abstract

3D shape synthesis is a fundamental task that benefits multiple research fields in computer vision. Like in any research field, implementing new methods paves the way for further advancements in the future. In this project, we prove that a Variational Auto-Decoder [14] works for the task of 3D shape generation using a significantly lighter model and hereby achieving comparable results when compared to previous methods.

1. Introduction

3D shape generation is a standing challenge in computer vision and has room for significant improvement. Areas like vision, graphics, and robotics benefits extensively from the advancements in 3D shape generation. A good generative model has high quality and diversity. By high quality, we mean that generated results look realistic to humans. In other words, while not being a part of the original data, generated shapes should be in the same distribution. We propose a model that follows these principles and is not implemented for the 3D generation task before. This model generates new 3D objects from random vectors without the guidance of an initial or a partial object.

We are using a model that learns the distribution of 3D implicit representations for a class. Instead of using a Variational Auto-Encoder, we are using a Variational Auto-Decoder [14]. In this way, we can represent each 3D shape directly as a latent vector. Operating directly on the latent vector space yields a linear space that can be interpolated.

Our contribution is proving that a latent space, which is acquired with a Variational Auto-Decoder, is suitable for 3D shape generation tasks. Therefore, we are using a lightweight model that yields comparable results to recent work on 3D generation while having significantly fewer parameters and training time. In this way, we show that Variational Auto-Decoders give good generative results with 3D data.

2. Related Work

Previous works use different data representations for the 3D shape generation task such as voxels [11] [12] [5] and point clouds [13] [1] [6] [7]. In our case, we are using an implicit representation in voxel space. Implicit representations are often used for different tasks in 3D computer vision [4] [9]. Therefore, we based our decoder on an implicit network called 3D-EPN [4] that is originally used for shape completion. We modified the 3D-EPN model by removing the encoder and the skip connections, adding more filters to the 3D transposed convolutional layers, and decreasing the bottleneck layer size (see section 3.1).

Auto-Decoders proved to be useful for encoding 3D data in a latent space and can be used for tasks like shape completion [10]. Variational Auto-Decoders use the idea of an Auto-Decoder with a probabilistic latent space [14]. In that way, it can be used for the generative task. However, the Variational Auto-Decoder method is originally implemented for 2D images. That said, we are using the latent space formulation of Variational Auto-Decoder for 3D shape representation.

3. Method

Our model can be used for two tasks: the first one is 3D shape generation and the second one is encoding objects in a linear latent space and generating new ones using linear interpolation. For shape generation, we train a Variational Auto-Decoder (VAD) model, while for shape encoding, we train a normal Auto-Decoder (AD). In the former case (VAD), a mean μ and a standard deviation Σ vector are assigned to each object, and a latent input is sampled using the reparameterization trick. In the Auto-Decoder case, only a single latent vector is assigned to each object.

3.1. Formulation

We are using the function that is proposed for training a Variational Auto-Decoder model. [14] The formulation of this function is:

$$\mathcal{V}\left(q_\theta, \theta | x_i\right) = \mathcal{L}_R - \mathcal{L}_{KL} \tag{1}$$

where:

$$\mathcal{L}_R = \mathbb{E}_{q_\phi(z|x_i)}[\ln(p_\theta(x|z))] \tag{2}$$

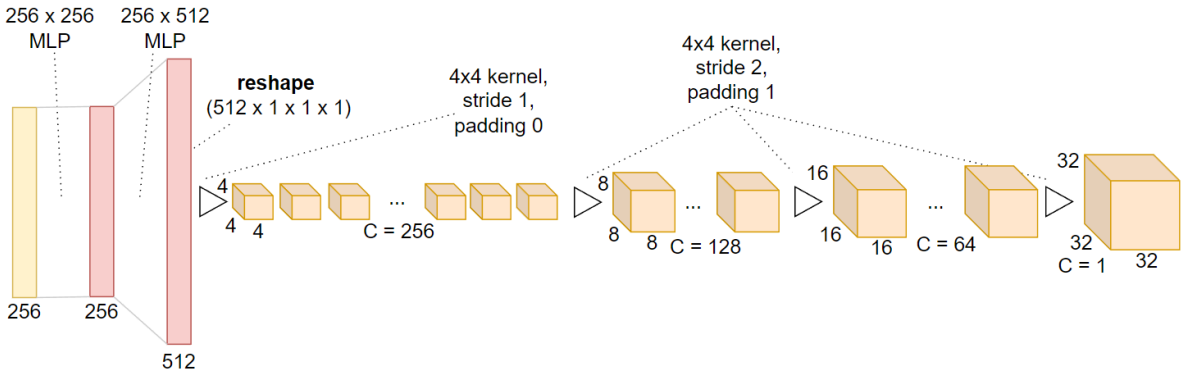

Figure 1. Detailed representation of the model. C corresponds to the number of channels.

and:

$$\mathcal{L}_{KL} = KL\left(q_\phi(z) \| p_\theta(z)\right) \tag{3}$$

Term $\mathcal{L}_R$ encourages minimizing the distance between real and generated posterior distributions. In other words, maximizing the $\mathcal{L}_R$ term maximizes the quality of the reconstruction. Term $\mathcal{L}_{KL}$ is the Kullback-Leibler divergence between latent samples and latent prior. In our case, the latent prior is a multivariate Gaussian distribution in which a differentiable sampling is possible with a reparameterization trick. Similar to the Variational Auto-Decoder paper [14], we define a multivariate Gaussian approximate posterior as:

$$q_\phi(z|x_i) := \mathcal{N}(z; \mu_i, \Sigma_i) \tag{4}$$

where $\phi = \{\mu_i, \Sigma_i\}$ is the representation for the posterior distribution. The reparameterization of the posterior latent code z is given by $z = \mu_i + \epsilon \cdot \Sigma_i$, where $\epsilon \sim \mathcal{N}(0, I)$. For practical purposes, we are using the loss function:

$$\mathcal{L}_{VAD} = -\mathcal{L}_R + \mathcal{L}_{KL} \tag{5}$$

which is $\mathcal{V}$ in Equation 1, multiplied by -1. This way, we can minimize the loss function $\mathcal{L}_{VAD}$. To minimize the term $-\mathcal{L}_R$, we are using the L_1 distance between generated distance field values and corresponding ground truth.

For the non-variational model, we are not using the $\mathcal{L}_{KL}$ term and solely optimize for the reconstruction loss. Instead of optimizing for vectors μ and Σ, we are optimizing for a single latent vector assigned for each object.

3.2. Model

Our model gets a latent vector as an input and outputs a 3D implicit representation in $\mathbb{R}^{32\times32\times32}$. We constructed this model by modifying an existing Auto-Encoder model that is proposed in [4] . This 3D encoder-predictor network is originally used for shape completion. Our modified version can be seen in Figure 1. We are using the same model for both variational and non-variational cases. The input of the model is a latent vector is $\mathbb{R}^{256}$. There are two main parts in our model: The first one is the bottleneck layer consists of 2 MLP layers, where the first one has a shape of 256×256 and the second one of 256×512. We are reshaping the output of the bottleneck layer to $1 \times 1 \times 1$ voxel with 512 channels. The second part is a convolutional decoder that consists of 4 3D transposed convolutional layers. Each layer has $4 \times 4 \times 4$ kernels while reducing the number of channels from 512 to 1. The first layer has a stride of 1 with no padding and the rest have a stride of 2 with padding 1. We get the logarithm of the output in each voxel to get a distance field representation where 1 corresponds to surface level.

3.3. Data

We use the ShapeNet [3] data with implicit representation in voxel space. We train on several categories, including airplanes, cars, and chairs. We split the data in 80% training, 10% validation and 10% test sets. For calculating our 1-NN score (see section 4.1), we generate meshes from our implicit representation using the marching cubes algorithm [8] and sample 2048 points from the object surface to get a point cloud representation.

3.4. Training

For training we used a learning rate of 0.01 for the decoder, latent codes and standard deviations with a scheduler that reduces the learning rate by half every 100 epochs. In total, each model is trained for 1000 epochs with a batch size of 64. We found out that a high kl-weight causes the model to overfit to certain distributions that are not similar to the distributions of a reference test set. A too low kl-weight causes our model to fit a linear distribution to the latent space instead of a probabilistic one. So, sampling from the prior distribution does not yield reasonable generated shapes. The optimal kl-weight we trained with is 0.03.

4. Evaluation

4.1. Quantitative Analysis

Model	Score (chair, training)
Occupancy Networks [9]	88.00
Ours	**92.00** (86.00 on validation)

Table 1. IOU results for reconstruction compared to [9] on chairs.

To evaluate our model on the reconstruction task, we follow the paper [9] and calculate the Intersection-Over-Union (IOU) of our distance field output on the training and validation sets as a sanity check (1). We therefore transform our output and ground truth into an occupancy grid. As can be seen in fig. 1, our model is able to achieve good reconstruction on the training chair dataset in comparison to [9]. To test the metric on validation, we freeze the decoder and optimize for the latent codes of the validation set only. The model also achieves a good score for the validation set and doesn't overfit in that matter.

Model	Airplanes	Chairs	Cars
r-GAN [1]	98.40	83.69	94.46
l-GAN (CD) [1]	87.30	68.58	66.49
l-GAN (EMD) [1]	89.49	71.90	71.16
PointFLow [13]	75.68	62.84	58.10
SoftFlow [6]	76.05	59.21	64.77
DPF-Net [7]	75.18	62.00	62.35
Shape-GF [2]	80.00	68.96	63.20
PVD [15]	**73.82**	**56.26**	**54.55**
Ours (whole)	76.33 (±1.3)	65.00 (±1.4)	77.56 (±1.1)
Ours (200)	73.05 (±1.1)	62.95 (±1.9)	72.25 (±1.2)
Ours (100)	69.25 (±2.8)	58.1 (±1.8)	65.5 (±1.2)

Table 2. 1-NN scores compared to the results presented in [15]. We report our scores on 3 different reference set sizes (the whole test set, 200, and 100 samples). The reported numbers of our model are the mean values after we calculate the score 10 times for each category. We also indicate the standard deviations next to them. The lower the value the better.

The 1-NN metric measures how well the distribution of the generated samples matches the distribution of a reference set. We use the exact implementation in [13]. Note that the smaller the score the better.

We calculate the 1-NN score of our model and compare it to the results of multiple models from [15]. Since the authors don't mention what size they use for their reference set, we calculate our score on 3 different sizes. We observe that the smaller the reference set, the better the final score is, as a bigger size effects the diversity of the generated shapes. We believe this happens because the capacity of our model is too small to keep generating diverse shapes at a large quantity. So as a part of future work, we can experiment with a bigger model and bigger latent codes. Nevertheless, overall our model achieves comparable good results. Even with the highest reference set size, our model outperforms some of the other models on airplanes and chairs. For cars however, our model does not perform as well. It is also important to mention that since we input random vectors every time we generate a shape, our 1-NN scores vary a little bit every time we calculate them. Thus, we run every model 10 times and note down the mean score value and the standard deviation (2).

4.2. Qualitative Analysis

4.2.1 Shape Synthesis

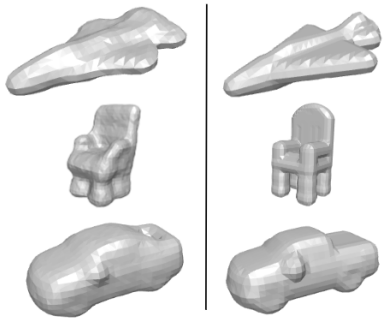

Figure 2. Samples of generated shapes (left) for the classes airplanes, chairs, and cars next to their nearest neighbor in the reference set (right).

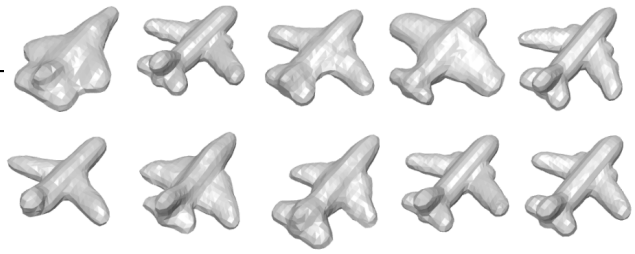

Figure 3. 10 generated shapes of airplanes, showing the diversity of our model.

In fig. 2 we show 3 samples of our generated results and compare them with the nearest neighbor in the reference set using the Chamfer distance. The comparison shows that the generated shapes are similar to the distributions in the reference set. Nevertheless, the generated shapes show different (new) properties, like the flat back side and more round wings on the airplane. The same can be observed with the chair, where the generated shape seems slightly different with wider narrow legs and a square shaped back. As for

the car, the main difference to be seen is the trunk where it is smaller and has an opening in the generated one. To show the diversity of our model, we generate 10 random samples of airplanes (fig. 3).

4.2.2 Intra-Class Interpolation

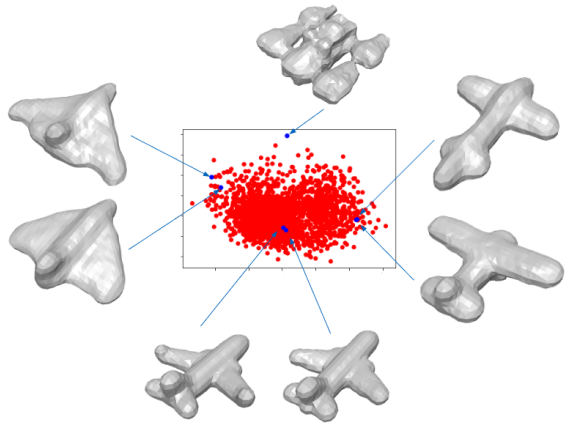

Figure 4. 2D latent space of the trained latent codes on airplanes. Non-variational model trained.

For intra-class interpolation, we train a non-variational decoder on different classes individually. We visualize the trained latent codes of airplanes (fig. 4) using Principal Component Analysis (PCA) in a 2 dimensional space and see that similar shapes are clustered together. With this representation, we can also see that it is possible to linearly interpolate within the latent space with our model.

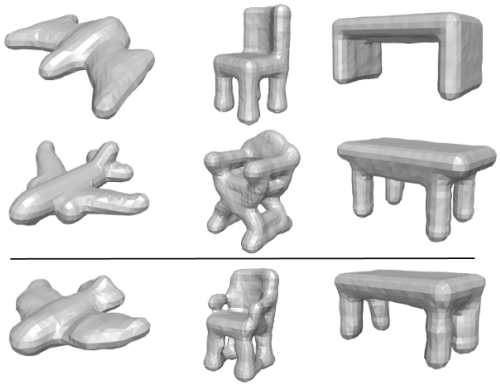

Figure 5. Examples of intra-class interpolation within the classes airplanes, chairs, and tables. The interpolated result is at the bottom below the line. Note that for the airplanes and tables we use the weights 0.5 for both objects while for the chairs we use weights of 0.59 (top) and 0.41 (bottom) since they offer a better result.

Fig. 5 shows interpolation examples from 3 classes. The interpolated results look meaningful and show properties from both parent objects. For example the resulting chair has arms like the bottom one but a back and legs like the top one. At 0.5/0.5 weights, the model does not always give a 'mean' shape as can be seen in the table example where the resulting table looks more similar to the bottom one. Note that not every pair of samples gives meaningful interpolation results, especially for hard and complex objects.

4.2.3 Inter-Class Interpolation

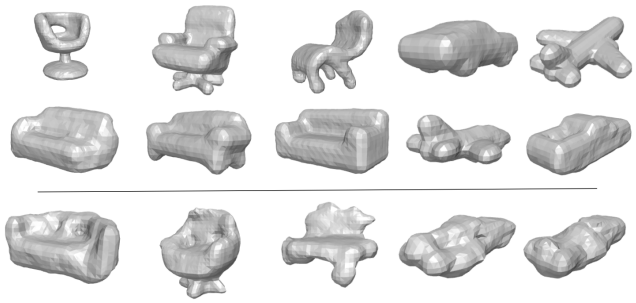

Figure 6. Example of inter-class interpolation between chairs and sofas, and cars and airplanes.

For the inter class interpolation we trained one (non-variational) model on the joint dataset of 2 classes. As can be seen in fig. 6, the inter class interpolation does not yield consistent results as the intra-class interpolation. Yet, again, we can create a linear latent space using multiple classes together.

5. Conclusion

In this project, we implement a new method for the 3D shape synthesis task. With the achieved comparable results (5), we proved that a Variational Auto-Decoder model can be used for 3D shape synthesis and outperform other methods even with a significantly more lightweight decoder. We show that our Auto-Decoder method can encode a single or multiple classes of 3D objects in a latent space, in which linear interpolation yields reasonable results.

Our main limitation of our model is the low capacity that comes from using too few parameters on the latent codes and decoder. One possible direction to improve performance is to use a deeper model. Also, the main formulation can be altered with different data representation, like point clouds, as can be found in several related papers.

References

[1] Panos Achlioptas, Olga Diamanti, Ioannis Mitliagkas, and Leonidas Guibas. Learning representations and generative models for 3d point clouds. In *International conference on machine learning*, pages 40–49. PMLR, 2018. 1, 3

[2] Ruojin Cai, Guandao Yang, Hadar Averbuch-Elor, Zekun Hao, Serge Belongie, Noah Snavely, and Bharath Hariharan. Learning gradient fields for shape generation. In *European Conference on Computer Vision*, pages 364–381. Springer, 2020. 3

[3] Angel X Chang, Thomas Funkhouser, Leonidas Guibas, Pat Hanrahan, Qixing Huang, Zimo Li, Silvio Savarese, Manolis Savva, Shuran Song, Hao Su, et al. Shapenet: An information-rich 3d model repository. *arXiv preprint arXiv:1512.03012*, 2015. 2

[4] Angela Dai, Charles Ruizhongtai Qi, and Matthias Nießner. Shape completion using 3d-encoder-predictor cnns and shape synthesis. In *Proc. Computer Vision and Pattern Recognition (CVPR), IEEE*, 2017. 1, 2

[5] Wenlong Huang, Brian Lai, Weijian Xu, and Zhuowen Tu. 3d volumetric modeling with introspective neural networks. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 8481–8488, 2019. 1

[6] Hyeongju Kim, Hyeonseung Lee, Woo Hyun Kang, Joun Yeop Lee, and Nam Soo Kim. Softflow: Probabilistic framework for normalizing flow on manifolds. *Advances in Neural Information Processing Systems*, 33:16388–16397, 2020. 1, 3

[7] Roman Klokov, Edmond Boyer, and Jakob Verbeek. Discrete point flow networks for efficient point cloud generation. In *European Conference on Computer Vision*, pages 694–710. Springer, 2020. 1, 3

[8] William E Lorensen and Harvey E Cline. Marching cubes: A high resolution 3d surface construction algorithm. *ACM siggraph computer graphics*, 21(4):163–169, 1987. 2

[9] Lars Mescheder, Michael Oechsle, Michael Niemeyer, Sebastian Nowozin, and Andreas Geiger. Occupancy networks: Learning 3d reconstruction in function space. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4460–4470, 2019. 1, 3

[10] Jeong Joon Park, Peter Florence, Julian Straub, Richard Newcombe, and Steven Lovegrove. Deepsdf: Learning continuous signed distance functions for shape representation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 165–174, 2019. 1

[11] Jiajun Wu, Chengkai Zhang, Tianfan Xue, William T. Freeman, and Joshua B. Tenenbaum. Learning a probabilistic latent space of object shapes via 3d generative-adversarial modeling. *CoRR*, abs/1610.07584, 2016. 1

[12] Jianwen Xie, Zilong Zheng, Ruiqi Gao, Wenguan Wang, Song-Chun Zhu, and Ying Nian Wu. Learning descriptor networks for 3d shape synthesis and analysis. *CoRR*, abs/1804.00586, 2018. 1

[13] Guandao Yang, Xun Huang, Zekun Hao, Ming-Yu Liu, Serge Belongie, and Bharath Hariharan. Pointflow: 3d point cloud generation with continuous normalizing flows. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 4541–4550, 2019. 1, 3

[14] Amir Zadeh, Yao-Chong Lim, Paul Pu Liang, and Louis-Philippe Morency. Variational auto-decoder: A method for neural generative modeling from incomplete data. *arXiv preprint arXiv:1903.00840*, 2019. 1, 2

[15] Linqi Zhou, Yilun Du, and Jiajun Wu. 3d shape generation and completion through point-voxel diffusion. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 5826–5835, October 2021. 3