

SeaLion: Semantic Part-Aware Latent Point Diffusion Models for 3D Generation

Supplementary Material

Contents

| | |
|--|----------|
| 1. Evaluation Metrics | 1 |
| 1.1. Calculation Formulas | 1 |
| 1.2. More Discussions about Part-aware Metrics . | 1 |
| 2. Pseudo-code of Part-aware 3D Editing | 2 |
| 3. Additional Experimental Details and Results | 2 |
| 3.1. Two-step Method on Intra Dataset | 2 |
| 3.2. Impact of the Segmentation Branch | 3 |
| 3.3. Data Augmentation based on DiffFacto and SeaLion | 3 |
| 3.4. Visualization of Generated Point Clouds from SeaLion | 3 |
| 3.5. Examples of Implausible Inter-part Coherence within the Generated Point Clouds from DiffFacto | 4 |
| 3.6. More Experimental Details and Hyper-parameters | 4 |

1. Evaluation Metrics

1.1. Calculation Formulas

Given a generated set $\mathcal{G} = \{x^g | x^g \in \mathbb{R}^{n \times 3}\}$ and a real dataset $\mathcal{R} = \{x^r | x^r \in \mathbb{R}^{n \times 3}\}$, both consist of point clouds with n points. In practice, \mathcal{R} is the test set unseen during the training of SeaLion, while \mathcal{G} is the set of samples generated during the inference. $D(\cdot)$ is the Chamfer distance or earth mover’s distance to measure the distance between two point clouds. The calculation formulas for metrics such as coverage (COV), minimum matching distance (MMD) [1], 1-nearest neighbor accuracy (1-NNA) [8], and snapping score (SNAP) [5] are listed as follows:

Coverage (COV) measures the ratio of overlap between \mathcal{R} and \mathcal{G} relative to the size of \mathcal{R} . It first constructs a subset by selecting the nearest neighbor in \mathcal{R} for each x_g , and then computes the ratio of the cardinality of this subset to the cardinality of \mathcal{R} ,

$$\text{COV}(\mathcal{G}, \mathcal{R}) = \frac{|\{\arg \min_{x_r \in \mathcal{R}} D(x_g, x_r) | x_g \in \mathcal{G}\}|}{|\mathcal{R}|}. \quad (1)$$

Minimum matching distance (MMD) computes the average distance between each x_r in \mathcal{R} and its nearest neighbor in \mathcal{G} ,

$$\text{MMD}(\mathcal{G}, \mathcal{R}) = \frac{1}{|\mathcal{R}|} \sum_{x_r \in \mathcal{R}} \min_{x_g \in \mathcal{G}} D(x_g, x_r). \quad (2)$$

1-nearest neighbor accuracy (1-NNA) measures the similarity between \mathcal{R} and \mathcal{G} by calculating the proportion of

samples in \mathcal{R} or \mathcal{G} whose nearest neighbors belong to the same set.

$$1\text{-NNA}(\mathcal{G}, \mathcal{R}) = \frac{\sum_{x_g \in \mathcal{G}} \mathbb{1}(N_{x_g} \in \mathcal{G}) + \sum_{x_r \in \mathcal{R}} \mathbb{1}(N_{x_r} \in \mathcal{R})}{|\mathcal{G}| + |\mathcal{R}|}, \quad (3)$$

where $\mathbb{1}[\cdot]$ is the indicator function, N_{x_g} is the nearest neighbor of x_g in the set $\mathcal{R} \cup \mathcal{G} \setminus \{x_g\}$, with the same applying to N_{x_r} . **If \mathcal{G} is very similar to \mathcal{R} , it becomes difficult to determine whether the nearest neighbor of x_g belongs to \mathcal{G} or \mathcal{R} , and vice versa.** In such cases, the 1-NNA score approaches **50%**.

Inter-part score (snapping metric, SNAP) [5] measures the connection tightness between two contacting parts in a object by computing the Chamfer distance between their closet N_{SNAP} points, e.g. $N_{\text{SNAP}} = 30$. For the point cloud x , the score $\text{SNAP}(x)$ is calculated by

$$\frac{1}{|P|} \sum_{p_1 \in P} \min_{x_{p_2} \in \mathcal{X}_{p_1}} \text{Chamfer}\{N_{x_{p_2}}^{(N_{\text{SNAP}})}(x_{p_1}), N_{x_{p_1}}^{(N_{\text{SNAP}})}(x_{p_2})\}, \quad (4)$$

where \mathcal{X}_{p_1} denotes the connected parts to x_{p_1} , e.g. if x_{p_1} is the car body, \mathcal{X}_{p_1} represents the contacting parts to the car body, {roof, hood, wheel}. $N_{x_{p_2}}^{(N_{\text{SNAP}})}(x_{p_1})$ refers to the N_{SNAP} nearest points in part x_{p_1} to part x_{p_2} .

1.2. More Discussions about Part-aware Metrics

In Section 3.4 of the main paper, we introduced novel metrics for evaluating the generation of segmentation-labeled point clouds, including 1-NNA (p-CD). The formula for 1-NNA is presented at (3), while the part-aware Chamfer distance p-CD (x_1, x_2) between point clouds x_1 and x_2 is computed as follows:

$$\sum_{p \in P} \left\{ \frac{1}{|x_p^1|} \sum_{q_1 \in x_p^1} \min_{q_2 \in x_p^2} \|q_1 - q_2\|_2^2 + \frac{1}{|x_p^2|} \sum_{q_2 \in x_p^2} \min_{q_1 \in x_p^1} \|q_1 - q_2\|_2^2 \right\}, \quad (5)$$

where x_p^1 and x_p^2 denote part p of the point clouds x^1 and x^2 , respectively, and $q_1, q_2 \in \mathbb{R}^3$ represent individual points. For point clouds composed of different parts, we define the pairwise distance as infinity. Here, we present a more comprehensive discussion on the development and rationale behind 1-NNA (p-CD), as detailed below:

- **Argument 1:** The core of evaluation for generation tasks is to measure the similarity between the generated set \mathcal{G} and the real dataset \mathcal{R} . If the two sets cannot be easily distinguished, the performance of the generative model is considered good. The assessment of this distinction incorporates both micro and macro factors: the instance-wise similarity between individual generated and real

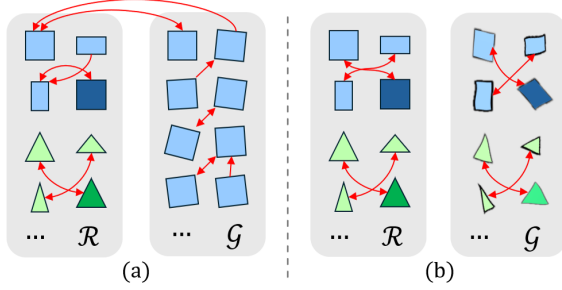


Figure 1. The generated set \mathcal{G} , which either (a) exhibits poor mode coverage compared to the real dataset \mathcal{R} or (b) contains poor-quality samples, cannot achieve a good 1-NNA score. The arrows indicate the nearest neighbors of samples. In both cases, most samples and their nearest neighbors belong to the same set, indicating a significant dissimilarity between \mathcal{G} and \mathcal{R} .

samples, and the overall distributional similarity between \mathcal{G} and \mathcal{R} , i.e. similar mode coverage. In cases where \mathcal{G} consists of high-quality samples but exhibits poor mode coverage (as shown in Figure 1 (a)), or where it has similar mode coverage but includes low-quality samples (as shown in Figure 1 (b)), the generative model cannot achieve a good 1-NNA score (close to 50%), since the samples in either \mathcal{R} or \mathcal{G} tend to have their nearest neighbors within the same set.

- **Argument 2:** For the unlabeled generative tasks [11], the calculation of 1-NNA is based on Chamfer Distance (CD), which quantifies the shape distance between two point clouds. Thus, the overall quality of x_g is represented by $CD(x_g, x_r)$: a low value of $CD(x_g, x_r)$ indicates a ideal quality of x_g , and vice versa. However, we need to consider two key factors in our task:
 - i. the overall quality of the generated point clouds,
 - ii. the accuracy (or rationality) of segmentation.

Due to the lack of “ground truth” segmentation for the generated point clouds, explicitly evaluating segmentation accuracy, such as using mIoU, becomes infeasible.

- **Argument 3:** The limitations of metrics such as 1-NNA-p [5], which obtain final results by averaging part-wise evaluations, in measuring inter-part plausibility are already discussed in Sec. 3.4 of the main paper. In contrast, our novel metric, 1-NNA (p-CD), can **explicitly evaluate shape quality and implicitly assess the rationality of segmentation**. The reasoning is as follows: given x_g and x_r with a very small part-aware Chamfer Distance, i.e. $p\text{-CD}(x_g, x_r) \rightarrow 0$, two facts are implied:
 - i. All parts of x_g are of good quality.
 - ii. All parts of x_g align well with the corresponding parts of x_r . Since the parts of x_r are assembled in a reasonable way, the corresponding parts of x_g also form a coherent and reasonable whole. In other words, x_g is segmented well.

| Metric | Model | Aneurysm |
|--------------------------------------|-------------------|--------------|
| 1-NNA (p-CD) ↓ (%) | Lion & PointNet++ | 74.57 |
| | Lion & SPoTr | 73.91 |
| | DiffFacto | 71.74 |
| | SeaLion | 65.22 |
| COV (p-CD) ↑ (%) | Lion & PointNet++ | 42.65 |
| | Lion & SPoTr | 30.43 |
| | DiffFacto | 39.13 |
| | SeaLion | 60.87 |
| MMD (p-CD) ↓ ($\times 10^{-2}$) | Lion & PointNet++ | 8.23 |
| | Lion & SPoTr | 19.68 |
| | DiffFacto | 8.05 |
| | SeaLion | 7.37 |

Table 1. Evaluation on Intra [9].

| Metric | Model | Airplane |
|------------------------------------|----------------|--------------|
| 1-NNA (CD) ↓ (%) | Lion | 65.66 |
| | SeaLion | 66.27 |
| COV (CD) ↑ (%) | Lion | 46.04 |
| | SeaLion | 46.63 |
| MMD (CD) ↓ ($\times 10^{-3}$) | Lion | 3.90 |
| | SeaLion | 4.07 |

Table 2. Impact of SeaLion’s segmentation branch on unlabeled generation tasks.

Therefore, 1-NNA (p-CD) effectively measures the similarity of \mathcal{G} and \mathcal{R} from the perspective of overall shape quality and segmentation accuracy.

2. Pseudo-code of Part-aware 3D Editing

As discussed in Section 3.3 of the main paper, SeaLion can serve as a tool for part-aware 3D shape editing. The related pseudo code is provided in Algorithm 1.

3. Additional Experimental Details and Results

3.1. Two-step Method on Intra Dataset

Since Lion [11] only released the pretrained weights for airplane, car, and chair classes from ShapeNet [10], we retrain Lion on the Intra [9] dataset to evaluate the two-step method on this dataset. Additionally, we train PointNet++ [7] and the state-of-the-art segmentation model, SPoTr [6], to assign pseudo labels on the generated point clouds, respectively. The experimental results presented in Table 1 demonstrate that SeaLion outperforms DiffFacto and the two-step method across all metrics, aligning with the trends observed in the main paper.

Algorithm 1 Part-aware 3D shape editing using SeaLion.

```
1: Input: Point cloud  $x$  consisting of  $n$  points, segmentation labels  $y$ , desired fix-shape part  $p$ .
2: Output: Novel generated point cloud  $x_0$  with preserved fix-shape part  $p$  and variation in the remaining parts, along with the updated segmentation labels  $y_0$ .
3:  $\text{mask}_p \leftarrow (y == p)$  ▷ Define a boolean mask to select points belonging to part  $p$ 
4:  $z_0 \leftarrow \phi_z(x)$ 
5:  $h_0 \leftarrow \phi_h(x, y, z_0)$ 
6:  $y_\tau \leftarrow y$  ▷  $\tau < T$ 
7: Perturb  $h_0$  for  $\tau$  steps to  $h_\tau$ 
8: for  $t \leftarrow \tau$  to 1 do
9:    $h_{t-1}, y_{t-1} \leftarrow \epsilon_h(h_t, t, z_0)$ 
10:   $y_{t-1} \leftarrow \alpha \cdot y_{t-1} + (1 - \alpha) \cdot y_t$  ▷ EMA smooth
11:   $\text{mask}_p^{t-1} \leftarrow ((1 - \text{mask}_p) \odot y_{t-1}) == p$ 
12:   $n_p^{t-1} \leftarrow \sum \text{mask}_p^{t-1}$ 
13:  if  $n_p^{t-1} > 0$  then ▷ Substitute the latent points in the remaining part but predicted as fix-shape part  $p$ 
14:     $\text{mask}_{\text{others}}^{t-1} \leftarrow ((1 - \text{mask}_p) \odot y_{t-1}) \neq p$ 
15:    Extract non-zero indices in  $\text{mask}_{\text{others}}^{t-1}$ , randomly sample  $n_p^{t-1}$  elements and then create a boolean mask for substitution  $\text{mask}_{\text{resample}}^{t-1}$ 
16:     $h_{t-1}[\text{mask}_p^{t-1}] \leftarrow h_{t-1}[\text{mask}_{\text{resample}}^{t-1}]$ 
17:     $y_{t-1}[\text{mask}_p^{t-1}] \leftarrow y_{t-1}[\text{mask}_{\text{resample}}^{t-1}]$ 
18:  end if
19:  Perturb  $h_0$  for  $t$  steps to  $h_t^*$ 
20:   $h_{t-1} \leftarrow \text{mask}_p \odot h_{t-1}^* + (1 - \text{mask}_p) \odot h_{t-1}$ 
21:   $y_{t-1} \leftarrow \text{mask}_p \odot y + (1 - \text{mask}_p) \odot y_{t-1}$ 
22: end for
23:  $x_0 \leftarrow \xi_h(h_0, y_0, z_0)$ 
24: Return  $x_0, y_0$ 
```

3.2. Impact of the Segmentation Branch

Although DDPMs are increasingly used as representation learners for various downstream tasks [3], such as image classification and segmentation, we are particularly interested in the impact of the segmentation branch on the original point cloud generation. If the representations for segmentation prediction and 3D noise prediction lie in entirely different distributions, combining both prediction tasks within a unified model could be detrimental. To investigate this, we ignore the predicted segmentation labels of the point clouds generated by SeaLion and re-evaluate them using metrics designed for unlabeled generative tasks, such as 1-NNA (CD). It is worth noting that the official weights of Lion [11] are trained on a larger dataset [2] compared to the segmentation-labeled subset [10]. To ensure a fair comparison, we retrain Lion on the smaller segmentation-labeled subset [10]. The experimental results presented in Table 2 illustrate that SeaLion achieves performance comparable to Lion in the evaluation of unlabeled generation task. This indicates that the representations for noise and segmentation predictions align well in the feature space. Therefore, incorporating the segmentation prediction branch and its associated training objective do not degrade

the generative performance.

3.3. Data Augmentation based on DiffFacto and SeaLion

Table 5 of the main paper presents the results of generative data augmentation using SeaLion for the segmentation task, where point clouds from six categories generated by SeaLion are incorporated to expand the training set of SPoTr [6]. We test on car class using SPoTr with the train set augmented by samples generated by DiffFacto [5] and SeaLion for comparison. The results are **78.23%** and **81.43%** on mIoU.

3.4. Visualization of Generated Point Clouds from SeaLion

Some of the generated point clouds of airplane, car, chair, guitar, lamp, and table categories from SeaLion are demonstrated in Figure 2, 3, 4, 5, 6, and 7, respectively. These generated point clouds demonstrate high-quality on overall shapes and exhibit diverse modalities. A video vividly showcasing the point clouds is submitted along with this paper. Furthermore, Figure 8 presents a visual comparison of cars generated by SeaLion, Lion & SPoTr [6, 11], and

DiffFacto [5].

3.5. Examples of Implausible Inter-part Coherence within the Generated Point Clouds from DiffFacto

An extreme case of implausible inter-part coherence within a shape is demonstrated in Figure 4 of the main paper. More realistic examples generated from DiffFacto [5] are shown in Figure 9.

3.6. More Experimental Details and Hyper-parameters

Hyper-parameters of the architecture of SeaLion. Details about the hyper-parameters of global encoder ϕ_z , global diffusion module ϵ_z , point-level encoder ϕ_h , point-level decoder ξ_h , and point-level diffusion module ϵ_h are listed in Table 3, 4, 5, 6, and 7, respectively. PVConv, SA, GA, and FP refer to point-voxel convolutions modules [4], set abstraction layers [7], global attention layers, and feature propagation layers [7], respectively.

More training details. The training of SeaLion includes two stages. We train the VAE model for 8k epochs in the first stage and the latent diffusion model for 24k epochs in the second stage. For these two stages, we use an Adam optimizer with a learning rate of $1e-3$. We conduct the experiments using an NVIDIA RTX 3090 GPU with 24GB of VRAM. For the experiments on ShapeNet [10], the training process takes an average of 5.4 hours for the first stage and 45 hours for the second stage across six categories.

Details of traditional data augmentation. In the experiment of generative data augmentation in the main paper, the traditional data augmentation methods including random rescaling (0.8, 1.2), random transfer (-0.1, 0.1), jittering (-0.005, 0.005), random flipping, random rotation around the x/y/z axis within a small range (-5° , $+5^\circ$).

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
- [2] 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. 3
- [3] Michael Fuest, Pingchuan Ma, Ming Gui, Johannes S Fischer, Vincent Tao Hu, and Bjorn Ommer. Diffusion models and representation learning: A survey. *arXiv preprint arXiv:2407.00783*, 2024. 3
- [4] Zhijian Liu, Haotian Tang, Yujun Lin, and Song Han. Point-voxel cnn for efficient 3d deep learning. *Advances in Neural Information Processing Systems*, 32, 2019. 4
- [5] George Kiyohiro Nakayama, Mikaela Angelina Uy, Jiahui Huang, Shi-Min Hu, Ke Li, and Leonidas Guibas. DiffFacto: Controllable part-based 3d point cloud generation with cross diffusion. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 14257–14267, 2023. 1, 2, 3, 4, 7, 8
- [6] Jinyoung Park, Sanghyeok Lee, Sihyeon Kim, Yunyang Xiong, and Hyunwoo J Kim. Self-positioning point-based transformer for point cloud understanding. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 21814–21823, 2023. 2, 3, 7
- [7] Charles Ruizhongtai Qi, Li Yi, Hao Su, and Leonidas J Guibas. Pointnet++: Deep hierarchical feature learning on point sets in a metric space. *Advances in neural information processing systems*, 30, 2017. 2, 4
- [8] 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
- [9] Xi Yang, Ding Xia, Taichi Kin, and Takeo Igarashi. Intra: 3d intracranial aneurysm dataset for deep learning. In *CVPR*, pages 2656–2666, 2020. 2
- [10] Li Yi, Vladimir G Kim, Duygu Ceylan, I-Chao Shen, Mengyan Yan, Hao Su, Cewu Lu, Qixing Huang, Alla Sheffer, and Leonidas Guibas. A scalable active framework for region annotation in 3d shape collections. *ACM Transactions on Graphics (ToG)*, 35(6):1–12, 2016. 2, 3, 4
- [11] Xiaohui Zeng, Arash Vahdat, Francis Williams, Zan Gojcic, Or Litany, Sanja Fidler, and Karsten Kreis. Lion: Latent point diffusion models for 3d shape generation. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2022. 2, 3, 7

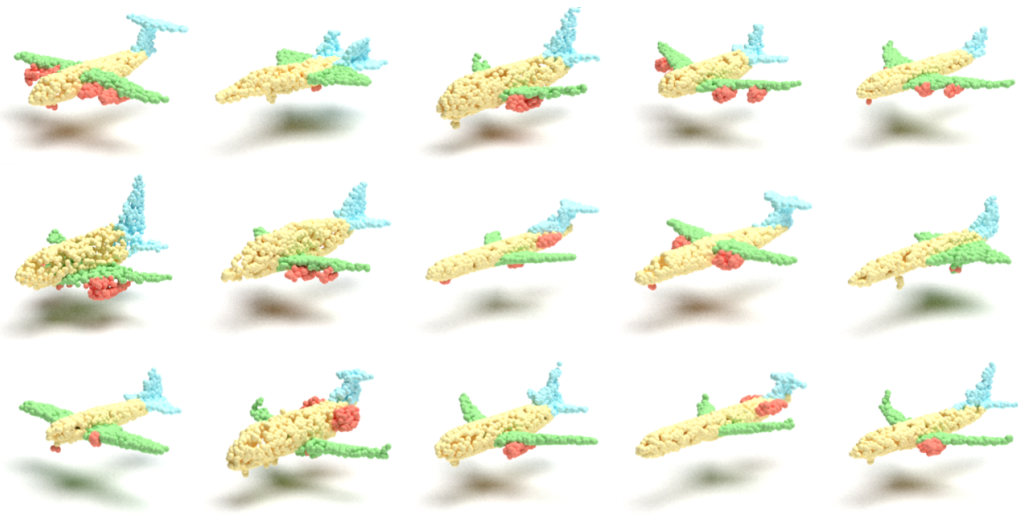


Figure 2. Generated point clouds of airplane class from SeaLion.

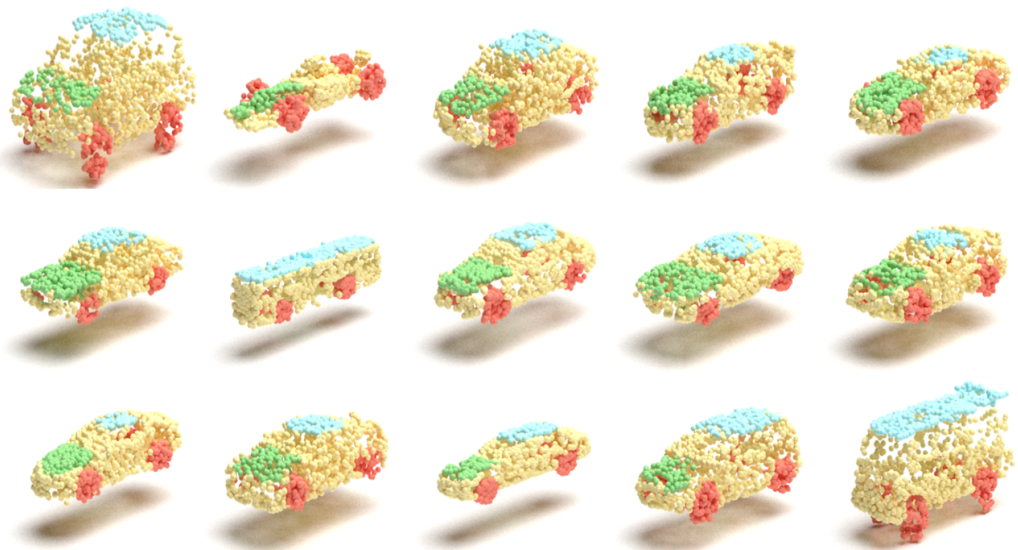


Figure 3. Generated point clouds of car class from SeaLion.

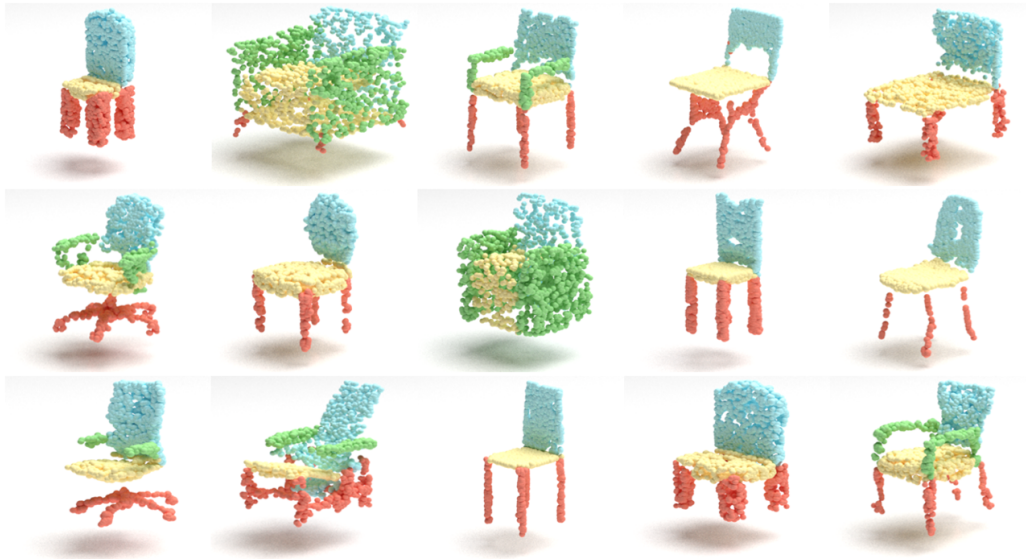


Figure 4. Generated point clouds of chair class from SeaLion.

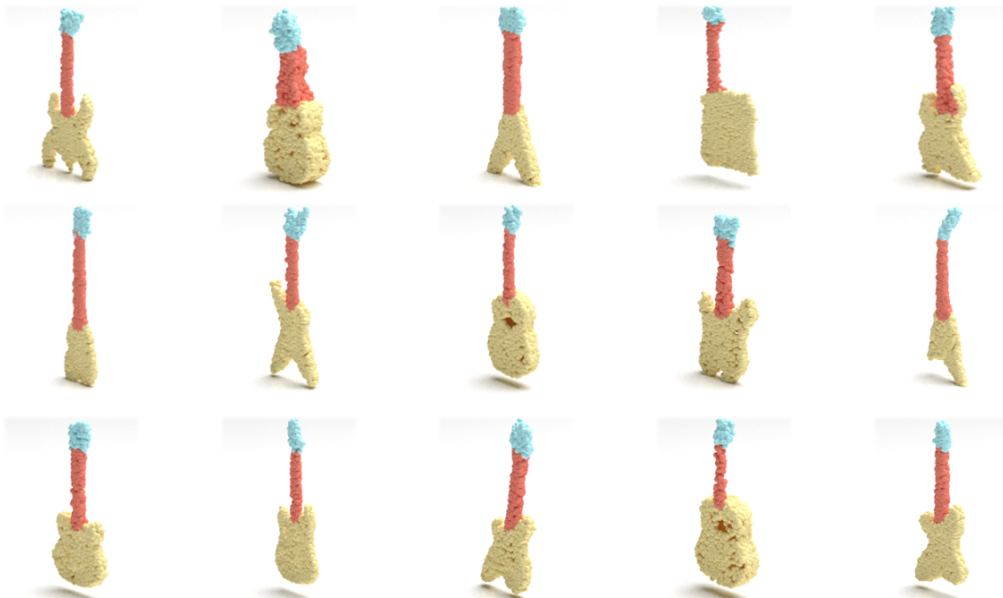


Figure 5. Generated point clouds of guitar class from SeaLion.



Figure 6. Generated point clouds of lamp class from SeaLion.

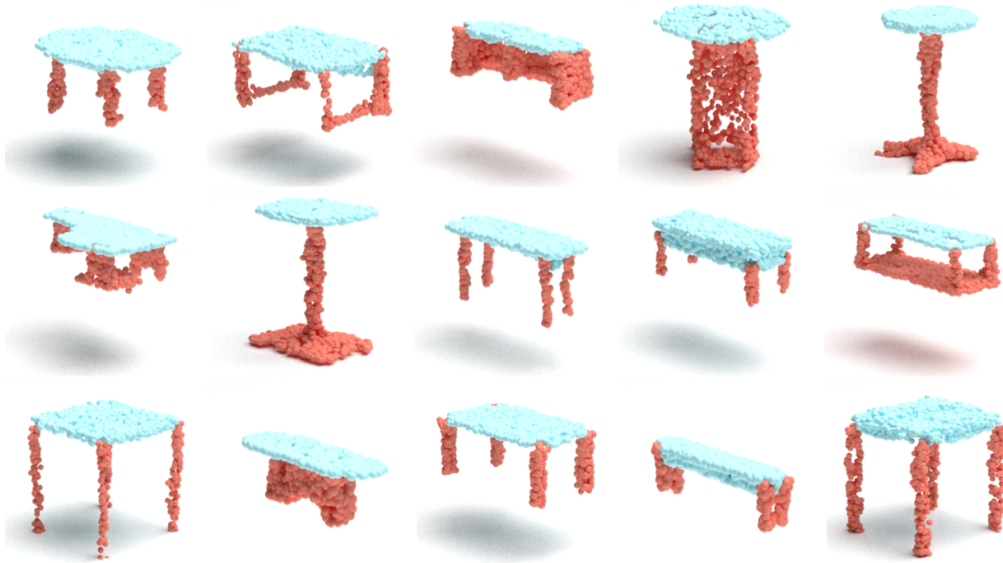


Figure 7. Generated point clouds of table class from SeaLion.

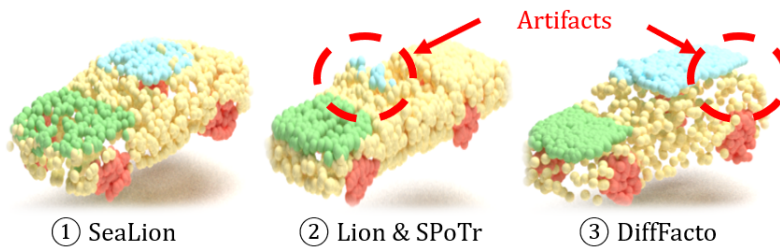


Figure 8. Comparison of cars generated by SeaLion, Lion & SPoTr [6, 11], and DiffFacto [5]. In ②, the tip of the front grass (yellow) is misclassified as the roof (blue), while in ③, the roof part is excessively large and incompatible with the body.

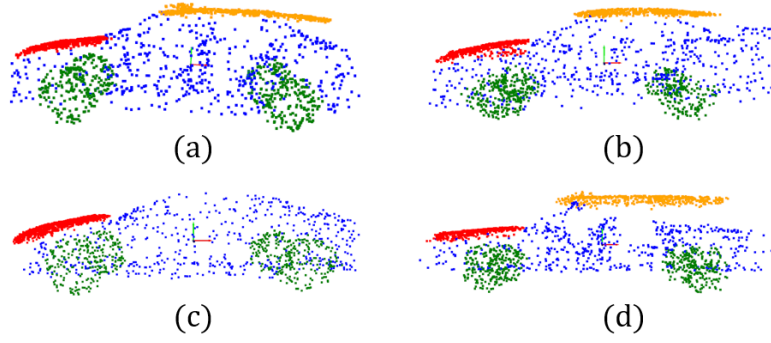


Figure 9. Examples of implausible inter-part coherence in shapes generated by DiffFacto [5]. (a) & (b) Too long roof. (c) Hood at an improper position. (d) The convertible car are not supposed to have a flat roof.

| | | | |
|--------------|----------------------------------|----------|---------|
| Input | point clouds (2048×3) | | |
| Output | global latent (1×128) | | |
| | | Layer 1 | Layer 2 |
| PVConv | layers | 2 | 1 |
| | hidden dimensions | 32 | 32 |
| | voxel grid size | 32 | 16 |
| SA | grouper center | 1024 | 256 |
| | grouper radius | 0.1 | 0.2 |
| | grouper neighbors | 32 | 32 |
| | MLP layers | 2 | 2 |
| | MLP output dimensions | 32, 32 | 32, 64 |
| Output layer | MLP layers | 2 | |
| | MLP output dimensions | 128, 128 | |

Table 3. Hyper-parameters of the global encoder ϕ_z .

| | | |
|----------------------|---|------------|
| Input | global latent (1×128), diffusion time step t | |
| Output | predicted noise on global latent (1×128) | |
| Input linear layer | output dimension | 2048 |
| Time embedding layer | sinusoidal embedding dimension | 128 |
| | MLP layers | 2 |
| | MLP output dimensions | 512, 2048 |
| Stacked ResNet | MLP layers | 2 |
| | MLP output dimensions | 2048, 2048 |
| | SE MLP layers | 2 |
| | SE MLP output dimensions | 256, 2048 |
| Output linear layer | output dimension | 128 |

Table 4. Hyper-parameters of the global diffusion ϵ_z .

| | | | | | |
|--------|--|--------------|----------|----------|---------------|
| Input | point clouds (2048×3), segmentation labels ($2048 \times c$), global latent (1×128) | | | | |
| Output | point-level latent (2048×4) | | | | |
| | | Layer 1 | Layer 2 | Layer 3 | Layer 4 |
| PVConv | layers | 2 | 1 | 1 | - |
| | hidden dimensions | 32 | 64 | 128 | - |
| | voxel grid size | 32 | 16 | 8 | - |
| SA | grouper center | 1024 | 256 | 64 | 16 |
| | grouper radius | 0.1 | 0.2 | 0.4 | 0.8 |
| | grouper neighbors | 32 | 32 | 32 | 32 |
| | MLP layers | 2 | 2 | 2 | 3 |
| | MLP output dimensions | 32, 32 | 64, 128 | 128, 256 | 128, 128, 128 |
| GA | hidden dimensions | 32 | 128 | 256 | 128 |
| | attention heads | 8 | 8 | 8 | 8 |
| FP | MLP layers | 3 | 2 | 2 | 2 |
| | MLP output dimensions | 128, 128, 64 | 128, 128 | 128, 128 | 128, 128 |
| PVConv | layers | 2 | 2 | 3 | 3 |
| | hidden dimensions | 64 | 128 | 128 | 128 |
| | voxel grid size | 32 | 16 | 8 | 8 |

Table 5. Hyper-parameters of the point-level encoder ϕ_h . Note: layer 1 refers to the shallowest layer and layer 4 refers to the deepest layer, c denotes the number of parts.

| | | | | | |
|--------------|--|--------------|----------|----------|---------------|
| Input | point-level latent (2048×4), segmentation labels ($2048 \times c$), global latent (1×128) | | | | |
| Output | point cloud (2048×3) | | | | |
| | | Layer 1 | Layer 2 | Layer 3 | Layer 4 |
| PVConv | layers | 2 | 1 | 1 | - |
| | hidden dimensions | 32 | 64 | 128 | - |
| | voxel grid size | 32 | 16 | 8 | - |
| SA | grouper center | 1024 | 256 | 64 | 16 |
| | grouper radius | 0.1 | 0.2 | 0.4 | 0.8 |
| | grouper neighbors | 32 | 32 | 32 | 32 |
| | MLP layers | 2 | 2 | 2 | 3 |
| | MLP output dimensions | 32, 64 | 64, 128 | 128, 256 | 128, 128, 128 |
| GA | hidden dimensions | 64+c | 128+c | 256+c | 128+c |
| | attention heads | 8 | 8 | 8 | 8 |
| FP | MLP layers | 3 | 2 | 2 | 2 |
| | MLP output dimensions | 128, 128, 64 | 128, 128 | 128, 128 | 128, 128 |
| PVConv | layers | 2 | 2 | 3 | 3 |
| | hidden dimensions | 64 | 128 | 128 | 128 |
| | voxel grid size | 32 | 16 | 8 | 8 |
| Output layer | MLP layers | 2 | | | |
| | MLP output dimensions | 128, 3 | | | |

Table 6. Hyper-parameters of the point-level decoder ξ_h . Note: layer 1 refers to the shallowest layer and layer 4 refers to the deepest layer, c denotes the number of parts.

| | | | | | |
|-----------------------------|---|--------------|----------|----------|---------------|
| Input | point-level latent (2048×4), diffusion time step t , global latent (1×128) | | | | |
| Output | predicted noise on point-level latent (2048×4), predicted segmentation labels ($2048 \times c$) | | | | |
| Time embedding | sinusoidal dimensions | 64 | | | |
| | MLP layers | 2 | | | |
| | MLP output dimensions | 64, 64 | | | |
| | | Layer 1 | Layer 2 | Layer 3 | Layer 4 |
| PVConv | layers | 2 | 1 | 1 | - |
| | hidden dimensions | 32 | 64 | 128 | - |
| | voxel grid size | 32 | 16 | 8 | - |
| SA | grouper center | 1024 | 256 | 64 | 16 |
| | grouper radius | 0.1 | 0.2 | 0.4 | 0.8 |
| | grouper neighbors | 32 | 32 | 32 | 32 |
| | MLP layers | 2 | 2 | 2 | 3 |
| | MLP output dimensions | 32, 64 | 64, 128 | 128, 256 | 128, 128, 128 |
| GA | hidden dimensions | 64 | 128 | 256 | 128 |
| | attention heads | 8 | 8 | 8 | 8 |
| FP (noise) | MLP layers | 3 | 2 | 2 | 2 |
| | MLP output dimensions | 128, 128, 64 | 128, 128 | 128, 128 | 128, 128 |
| PVConv (noise) | layers | 2 | 2 | 3 | 3 |
| | hidden dimensions | 64 | 128 | 128 | 128 |
| | voxel grid size | 32 | 16 | 8 | 8 |
| Output layer (noise) | MLP layers | 2 | | | |
| | MLP output dimensions | 128, 4 | | | |
| FP (segmentation) | MLP layers | 3 | 2 | 2 | 2 |
| | MLP output dimensions | 128, 128, 64 | 128, 128 | 128, 128 | 128, 128 |
| PVConv (segmentation) | layers | 2 | 2 | 3 | 3 |
| | hidden dimensions | 64 | 128 | 128 | 128 |
| | voxel grid size | 32 | 16 | 8 | 8 |
| Output layer (segmentation) | MLP layers | 2 | | | |
| | MLP output dimensions | 128, c | | | |

Table 7. Hyper-parameters of the point-level diffusion ϵ_h . Note: layer 1 refers to the shallowest layer and layer 4 refers to the deepest layer, c denotes the number of parts.