

SALAD: Part-Level Latent Diffusion for 3D Shape Generation and Manipulation

Juil Koo* Seungwoo Yoo* Minh Hieu Nguyen* Minhyuk Sung
KAIST

{63days,dreamy1534,hieurstics,mhsung}@kaist.ac.kr

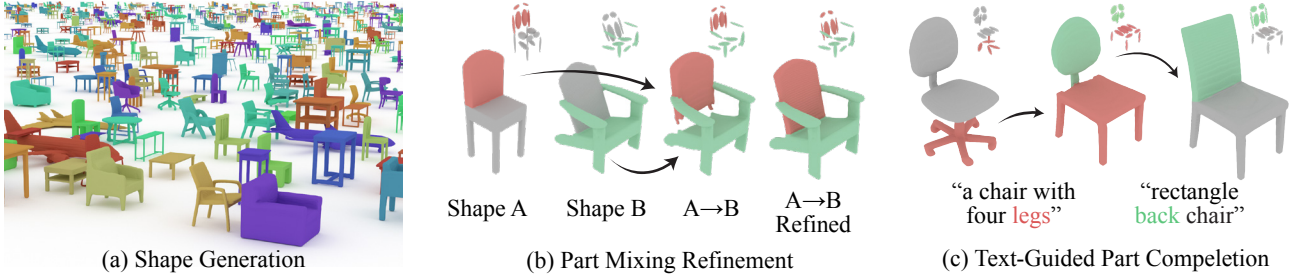


Figure 1: **An overview of SALAD.** (a) Our cascaded diffusion model trained on part-level 3D representations produces high-quality 3D shapes of different classees. Although trained for *unconditional* generation, SALAD hints its zero-shot capability in various manipulation scenarios, including (b) part mixing and refinement, and (c) text-guided part completion.

Abstract

We present a cascaded diffusion model based on a part-level implicit 3D representation. Our model achieves state-of-the-art generation quality and also enables part-level shape editing and manipulation without any additional training in conditional setup. Diffusion models have demonstrated impressive capabilities in data generation as well as zero-shot completion and editing via a guided reverse process. Recent research on 3D diffusion models has focused on improving their generation capabilities with various data representations, while the absence of structural information has limited their capability in completion and editing tasks. We thus propose our novel diffusion model using a part-level implicit representation. To effectively learn diffusion with high-dimensional embedding vectors of parts, we propose a cascaded framework, learning diffusion first on a low-dimensional subspace encoding extrinsic parameters of parts and then on the other high-dimensional subspace encoding intrinsic attributes. In the experiments, we demonstrate the outperformance of our method compared with the previous ones both in generation and part-level completion and manipulation tasks. Our project page is <https://salad3d.github.io>.

1. Introduction

The staggering rise of the recent image generative model such as DALL-E 2 [56], StableDiffusion [57], and Midjourney [41] has drawn great attention to the diffusion mod-

els. With the state-of-the-art performance in generating data [14, 21, 57, 56, 41], diffusion models have quickly replaced existing generative models in many applications. Besides the quality of the generated data, another key advantage of the diffusion models is the zero-shot capability in completion and editing. Recent research [11, 36, 39] has shown that diffusion models trained without any conditions can be applied to completion and editing tasks by starting the reverse process from partial data and properly guiding the process.

Such capabilities of the diffusion models have prompted attempts to apply them to 3D generation [4, 37, 48, 73, 70, 33, 46, 27], although likewise the other neural 3D generation and reconstruction work, the key challenge in applying diffusion to 3D is to find an appropriate representation of 3D data. Particularly, to take full advantage of the diffusion models, both producing realistic data and being leveraged to editing and manipulation, a careful design of the 3D data representation is needed. A naive adaption of the 2D image diffusion models to the 3D voxels is impractical due to the order of magnitude more computation time and memory. Hence, the earlier attempt to apply diffusion or score-based models to 3D (which has also been continued until recently) was to use point clouds as 3D representation [4, 37, 48], although the fine details of shapes could not be reproduced since the training computation is still too heavy to increase the resolution — $2k$ points are used in training. Later, some hybrid representations have been explored, such as points and voxels [73], points and features [70], voxels and features [33], although these were still limited in being trained

*Equal contribution.

with low-resolution 3D data. Implicit representation has been proven to be the best to capture fine details in 3D generation and reconstruction [49, 8, 40]. Hence, concurrent work [33, 46] introduced latent diffusion methods generating codes that can be decoded into implicit functions of 3D shapes. However, then the diffusion in a latent space cannot be used for the *guided* reverse process – e.g., filling a missing part of a shape while preserving the others, and thus the model cannot be exploited for manipulation. Neural wavelet [27] is a notable exception that improves efficiency in training without a latent space but by learning diffusion in spectral wavelet space. While it succeeded in producing local details, it is still nontrivial to specify a local region to be modified in the spectral space, thus limiting the model to be used in the manipulation tasks.

As a 3D diffusion model feeding two birds with one seed, achieving high-quality *generation* and enabling *manipulation*, we present our novel Shape PArt-Level LATent Diffusion Model, dubbed **SALAD**. Our work is inspired by recent work [17, 26, 34, 20] introducing disentangled implicit representations into *parts*. The advantages of the part-level disentangled representation are in the *efficiency* allocating the memory capacity of the latent code effectively to multiple parts, and also in the *locality* allowing each part to be edited independently, thus best fitted to our purpose. We specifically base our work on SPAGHETTI [20] that learns the part decomposition in a self-supervised way. Each part is described with an independent embedding vector describing the extrinsics and intrinsics of the part as shown in Figure 1, and thus the parts that need to be edited or replaced can be easily chosen. It is a crucial difference from latent diffusion where the latent codes do not explicitly express any spatial and structural information and voxel diffusion where the region to be modified can only be specified in the 3D space, not in the shape.

Our technical contribution is the diffusion neural network designed to properly handle the characteristics of the part-level implicit representation, which is a *set* of *high-dimensional* embedding vectors. To cope with the set data and achieve permutation invariance while allowing global communications across the parts, we employ Transformer [65] and condition each self-attention block with the timestep in the diffusion process. The challenge is also in learning diffusion in the high-dimensional embedding space, which is known to be hard to train [69]. To get around the issue, we introduce a *two-phase cascaded* diffusion model. We leverage the fact that the part embedding vector is split into a small set of *extrinsic* parameters approximating the shape of a part and a high-dimensional *intrinsic* latent supplementing the detailed geometry information. Hence, our cascaded pipeline learns two diffusions, one generating extrinsic parameters first and the other producing an intrinsic latent conditioned on the extrinsics, ef-

fectively improving the generation quality with the same computation resources.

Our quantitative and qualitative assessments on SALAD demonstrate its outperformance compared with SotA methods in shape generation as shown in Section 5.1. We further demonstrate zero-shot manipulation capability of our SALAD, trained solely for unconditional generation, by conducting extensive experiments on downstream tasks, including part completion (Section 5.2), part mixing and refinement (Section 5.3). Last but not least, we showcase the versatility of SALAD in modeling multi-modal distributions such as text-guided generation (Section 5.4) and completion (Section 5.5). To summarize, our contributions are:

- We propose SALAD, a novel diffusion model capable of generating part-level 3D implicit representations.
- We propose a *two-phase cascaded* diffusion model, effective for handling high-dimensional latent spaces, that sets a new SotA in shape generation.
- We demonstrate the importance of orchestrating diffusion models and part-level implicit representation for the zero-shot capability of SALAD in shape editing.
- We further extend our SALAD to text-guided generation and editing that can synergize with text-driven part segmentation network.

2. Related Work

3D Generative Models. The first 3D generative models are based on GAN, learning a distribution of latents that can be decoded into various 3D representations such as point clouds [1, 64, 59] and implicit representations [29, 19, 8, 28, 72]. Later research [5, 16] also proposed to leverage a 2D discriminator in the 3D GAN training while projecting the 3D shape to 2D via differentiable rendering [31, 42]. Autoregressive models for 3D data have also been introduced to produce meshes [47], point clouds [62], or (ir)regular feature grids [71, 66], which have also been extended to handle conditional inputs in the completion [66] and multi-modal generation [43, 15] tasks. Recent work focused on exploiting the better generation capabilities of diffusion and score-based models. Cai *et al.* [4], Luo and Hu [37], and Zhou *et al.* [73] were the first proposing score-based [4] or diffusion-based [37, 73] frameworks learning distributions of point clouds. Hui *et al.* [27] proposed to learn diffusion over wavelet coefficients of truncated signed distance functions. The recent success of latent diffusion models (LDMs) [57] for 2D images also prompted to develop diffusion models operating on latent vectors of either the entire 3D shapes [10, 46] or each point [70], voxel [33], and triplane [60] (note that all of them are *concurrent* work except for LION [70]). Conditional models taking texts [48] or multimodal data [33, 9] are also concurrently introduced with our work.

The advances in 3D generative models have shown significant improvement in the quality of produced shapes, although, in our work, we focus on introducing a more *versatile* 3D generative model that can be used not only for shape generation but also for shape editing and completion *without* any additional training for the conditional setups (yet also achieving the SotA generation results). We aim to fully utilize the manipulation capabilities of the diffusion model with a compact part-level implicit representation of 3D shapes.

Part-Level Implicit 3D Representations. There is a large body of work exploring part-level 3D decomposition, although most of which focuses on segmenting or abstracting a supervised [68, 53, 54, 45, 44] and unsupervised [63, 61, 67, 51, 7, 13, 50] ways. Recent work coupled the part-level structure with the implicit shape representation to enable shape manipulation with the part representation parameters. SIF [18] and LDIF [17] first introduced the idea of combining a set of Gaussians in the 3D space to local implicit functions corresponding to each of them. NeuralTemplate [26] instead used a set of convexes as the part-level extrinsics and connected each of them with a latent vector decoded into a local implicit function. SPAGHETTI [20] employed 3D Gaussians again but trained the network so that the Gaussians can not only approximate the shape but also transform a local region with its mean and covariance parameters. We base our work on SPAGHETTI and present a framework of learning diffusion on the SPAGHETTI representation. While SPAGHETTI also provided an auto-decoding-based shape generation pipeline, we demonstrate that our cascaded model diffusing on extrinsics and intrinsics sequentially produces shapes with much better quality while learning the exact data distributions on both spaces.

3. Diffusion Models and Part-Level Shape Representation

3.1. Background on Diffusion Models

We first briefly overview the technical background of diffusion models. Diffusion models [21] are latent variable models that approximate a data distribution $q(\mathbf{x}^{(0)})$ with a Markov chain, which is also called a *reverse process*:

$$p_\theta(\mathbf{x}^{(0)}) := \int p_\theta(\mathbf{x}^{(0:T)}) d\mathbf{x}^{(1:T)}, \quad (1)$$

where $p_\theta(\mathbf{x}^{(0:T)}) = p(\mathbf{x}^{(T)}) \prod_{t=1}^T p_\theta(\mathbf{x}^{(t-1)}|\mathbf{x}^{(t)})$. Here, $p(\mathbf{x}^{(T)}) = \mathcal{N}(\mathbf{x}^{(T)}; \mathbf{0}, \mathbf{I})$ is the standard normal prior enabling tractable sampling.

The conditional probabilities $\{p_\theta(\mathbf{x}^{(t-1)}|\mathbf{x}^{(t)})\}_{t=1}^T$ are parameterized by a neural network whose weights are denoted by θ . The weights are optimized through the *forward* diffusion process $q(\mathbf{x}^{(1:t)}|\mathbf{x}^{(0)})$ that sequentially adds Gaus-

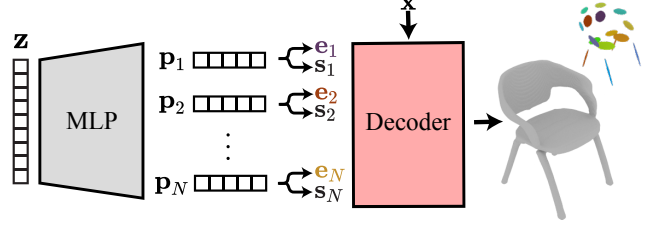


Figure 2: **Part-Level implicit representation by Hertz *et al.* [20].** A latent vector \mathbf{z} encoding global geometry is first mapped to a set of part latents $\{\mathbf{p}_i\}_{i=1}^N$, each of which is decomposed into extrinsic parameters $\{\mathbf{e}_i\}_{i=1}^N$ and intrinsic latents $\{\mathbf{s}_i\}_{i=1}^N$. The decoder, conditioned on $\{(\mathbf{e}_i, \mathbf{s}_i)\}_{i=1}^N$, outputs an occupancy value given a query point \mathbf{x} .

sian noises to the data $\mathbf{x}^{(0)} \sim q(\mathbf{x}^{(0)})$:

$$q(\mathbf{x}^{(1:t)}|\mathbf{x}^{(0)}) := \prod_{s=1}^t q(\mathbf{x}^{(s)}|\mathbf{x}^{(s-1)}), \quad (2)$$

where $q(\mathbf{x}^{(s)}|\mathbf{x}^{(s-1)}) := \mathcal{N}(\mathbf{x}^{(s)}; \sqrt{1 - \beta(s)}\mathbf{x}^{(s-1)}, \beta(s)\mathbf{I})$,

and $\beta(s)$ is an element of a monotonically increasing sequence $\beta^{(1:T)} \in (0, 1]^T$. By choosing Gaussians as forward diffusion kernels, the conditional densities $q(\mathbf{x}^{(t)}|\mathbf{x}^{(0)})$ at $t = 1, \dots, T$ can be expressed in the closed form:

$$q(\mathbf{x}^{(t)}|\mathbf{x}^{(0)}) = \mathcal{N}(\mathbf{x}^{(t)}; \sqrt{\alpha(t)}\mathbf{x}^{(0)}, (1 - \alpha(t))\mathbf{I}), \quad (3)$$

where $\alpha(t) := 1 - \beta(t)$ and $\bar{\alpha}(t) := \prod_{s=1}^t \alpha(s)$. Over the forward process dissipating a sample $\mathbf{x}^{(0)} \sim q(\mathbf{x}^{(0)})$ toward $q(\mathbf{x}^{(T)}) = \mathcal{N}(\mathbf{0}, \mathbf{I})$, the weights θ parameterizing the reverse process $p_\theta(\mathbf{x}^{(0)})$ are learned by optimizing the following variational bound on negative log likelihood:

$$\mathbb{E}_{q(\mathbf{x}^{(0)})} [-\log p_\theta(\mathbf{x}^{(0)})] \leq \mathbb{E}_{q(\mathbf{x}^{(0)}, \dots, \mathbf{x}^{(T)})} \left[-\log \frac{p_\theta(\mathbf{x}^{(0:T)})}{q(\mathbf{x}^{(1:T)}|\mathbf{x}^{(0)})} \right]. \quad (4)$$

Following Ho *et al.* [21], we parameterize our reverse process $p_\theta(\mathbf{x}^{(t-1)}|\mathbf{x}^{(t)})$ as:

$$p_\theta(\mathbf{x}^{(t-1)}|\mathbf{x}^{(t)}) := \mathcal{N}(\mathbf{x}^{(t-1)}; \boldsymbol{\mu}_\theta(\mathbf{x}^{(t)}, t), \beta^{(t)}\mathbf{I}). \quad (5)$$

In particular, we use the parameterization $\boldsymbol{\mu}_\theta(\mathbf{x}^{(t)}, t) = 1/\sqrt{\alpha^{(t)}}(\mathbf{x}^{(t)} - \beta^{(t)}/\sqrt{1 - \bar{\alpha}^{(t)}}\boldsymbol{\epsilon}_\theta(\mathbf{x}^{(t)}, t))$ and optimize its parameters θ with a training objective that encourages a network $\boldsymbol{\epsilon}_\theta$ to predict the noise $\boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ present in the given data:

$$\mathcal{L}(\theta) := \mathbb{E}_{t, \mathbf{x}^{(0)}, \boldsymbol{\epsilon}} \left[\left\| \boldsymbol{\epsilon} - \boldsymbol{\epsilon}_\theta \left(\sqrt{\bar{\alpha}^{(t)}}\mathbf{x}^{(0)} + \sqrt{1 - \bar{\alpha}^{(t)}}\boldsymbol{\epsilon}, t \right) \right\|^2 \right]. \quad (6)$$

3.2. Part-Level Shape Representation

Neural implicit representations [8, 49, 40] have been widely exploited in 3D shape generation and reconstruction due to their advantages in capturing fine details without

limitation in resolutions even with a small memory footprint. However, their disadvantage of not supporting intuitive editing and manipulation has been a hindrance to increasing their utilization. To remedy the drawback, recent works [18, 17, 19, 26, 20] introduced *dual* representations combining explicit and implicit representations, taking advantage of both of them. Among them, Hertz *et al.* [20], which our work is based on, was the first introducing a hybrid representation integrating two types of disentanglements simultaneously into an implicit representation: 1) part-level disentanglement, representing each local region separately, and 2) extrinsic-intrinsic disentanglement, describing extrinsic properties (i.e., the approximate shape and transformations) with parameters in the 3D space while encoding intrinsic properties (i.e., geometric details) using a latent code. This novel representation, called SPAGHETTI [20], is learned in an auto-decoding setup without any supervision of the part decomposition.

In SPAGHETTI, a 3D shape is first mapped to a global latent \mathbf{z} and then further encoded into a set of part embedding vectors $\{\mathbf{p}_i\}_{i=1}^N$, where N denotes the number of parts. Each part embedding vector \mathbf{p}_i is again mapped into both a set of extrinsic parameters \mathbf{e}_i and an intrinsic latent \mathbf{s}_i through an MLP. The set of extrinsic parameters $\mathbf{e}_i = \{\mathbf{c}_i, \Sigma_i, \pi_i\}$ of each part represents a Gaussian in the 3D space with mean $\mathbf{c}_i \in \mathbb{R}^3$ and covariance $\Sigma_i \in \mathbb{R}^{3 \times 3}$, depicting an approximate shape of a part. $\pi_i \in \mathbb{R}$ is the blending weight for the Gaussian mixture representation of the entire shape: $\sum_i \pi_i \mathcal{N}(\mathbf{x}|\mathbf{c}_i, \Sigma_i)$, describing the volume of the shape as a probability distribution. Since $\{\mathbf{e}_i\}_{i=1}^N$ can only encode the part-level structural information, the intrinsic latents $\{\mathbf{s}_i\}_{i=1}^N$ supplement the detailed geometry information so that the pairs of the extrinsic parameters and intrinsic latents can be decoded back to the original shape in an implicit form. Specifically, an implicit decoder \mathcal{D} is trained to predict an occupancy value at point \mathbf{x} :

$$o = \mathcal{D}\left(\mathbf{x} \mid \{\mathbf{e}_i\}_{i=1}^N, \{\mathbf{s}_i\}_{i=1}^N\right), \quad (7)$$

where occupancy value $o \in [0, 1]$ is 1 when the query point is inside the shape, and 0 otherwise. The keys to achieving both the part-level and extrinsic-intrinsic disentanglements in the training of decoder \mathcal{D} are the regularizations forcing a single pair $(\mathbf{e}_i, \mathbf{s}_i)$ of a part to determine the occupancy of each point, and the Gaussian parameters in \mathbf{e}_i to transform the corresponding local region. See the original paper [20] for the details of the decoder training.

The extrinsic vector \mathbf{e}_i is precisely represented as a 16-dimensional vector $\{\mathbf{c}_i, \lambda_i^1, \lambda_i^2, \lambda_i^3, \mathbf{u}_i^1, \mathbf{u}_i^2, \mathbf{u}_i^3, \pi_i\}$, where $\lambda_i^j \in \mathbb{R}$ and $\mathbf{u}_i^j \in \mathbb{R}^3$ are eigenvalues and eigenvectors of the covariance matrix Σ_i , while the intrinsic vector \mathbf{s}_i is a 512-dimensional vector. Note that the much smaller extrinsic vector contains the approximate shape information of the part; we leverage this fact in our effective cascaded

diffusion model.

Also, note that SPAGHETTI is trained in an auto-decoding setup while regularizing the global latent code $\mathbf{z} \in \mathbb{R}^{512}$ to follow the unit Gaussian. Thus, the shapes can be simply generated by sampling a latent code \mathbf{z} from the unit Gaussian in the \mathbf{z} space, although we demonstrate that diffusion in the extrinsic and intrinsic embedding spaces can produce much more plausible shapes (Section 5.1).

4. SALAD – Part-Level Cascaded Diffusion

Here we introduce our cascaded diffusion framework generating the part-level implicit shape representation. In the shape representation introduced in Section 3.2, note that there are multiple *layers* of representations all of which can be decoded into the original shape, such as the global latent \mathbf{z} , the set of part latents $\{\mathbf{p}_i\}$, and the set of extrinsic and intrinsic vectors $\{(\mathbf{e}_i, \mathbf{s}_i)\}$. Below, we first introduce some preliminary approaches to learning diffusion for each representation, and then we propose our final cascaded framework for learning diffusions in two phases.

Diffusion of \mathbf{z} . Learning diffusion in the space of the global shape latent \mathbf{z} is straightforward; the noise prediction network ϵ_θ (in Equation 6) can be simply modeled as an MLP. In the network ϵ_θ , the timestep t is generally first transformed by a positional encoding $\gamma(\cdot)$ [65] and then fed as the scale and translation factors to the adaptive normalization layers such as AdaLN [52]. In our experiments (Section 5.1), we show that this simple diffusion already outperforms the quality of generation by sampling \mathbf{z} from the unit Gaussian since it can learn the exact distribution of \mathbf{z} , although the improvement is marginal.

Diffusion of $\{\mathbf{p}_i\}_{i=1}^N$. To improve the quality of generation, one can instead consider diffusing the set of part latents $\{\mathbf{p}_i\}_{i=1}^N$. A simple MLP taking the concatenation of the part latents as input, however, results in diffusion in a very high-dimensional space and also does not address the order invariance of the set data. We employ Transformer [65] to properly handle the set data while also promoting communications across parts. Each self-attention block is equipped with a post-MLP, where the positional-encoded timestep $\gamma(t)$ is fed to the AdaLN layer. This part-level latent diffusion can better reproduce the details of each part, while it still suffers from the difficulty in diffusing in a high-dimensional latent space.

Cascaded Diffusion of $\{\mathbf{e}_i\}_{i=1}^N$ and $\{\mathbf{s}_i\}_{i=1}^N$. Inspired by Ho *et al.* [22] introducing *cascaded* diffusion for images, diffusing low-resolution images first and then diffusing high-resolution images conditioned on the low-resolution outputs, we propose a *two-phase* framework for learning diffusion. We observe that the extrinsic and intrinsic attributes $\{\mathbf{e}_i\}_{i=1}^N$ and $\{\mathbf{s}_i\}_{i=1}^N$ play similar roles to low-

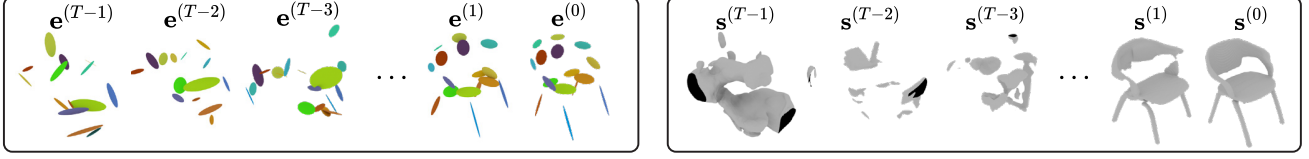


Figure 3: **Pipeline overview.** SALAD consists of two diffusion models for extrinsic and intrinsic vectors, respectively. During phase 1 (left), it generates extrinsic vectors representing structures of shapes. Phase 2 (right) takes these outputs as conditions and produces intrinsic vectors encoding local geometry information.

and high-resolution images; the former describes the approximate of the data, while the latter captures fine details. Also importantly, the extrinsic vector \mathbf{e}_i is much lower-dimensional, thus easier to make the noise prediction converge. Thus, in our first phase, we learn the diffusion of $\{\mathbf{e}_i\}_{i=1}^N$ with the same Transformer-based noise prediction network ϵ_θ above. Then, in the second phase, we use another Transformer-based network ϵ_ϕ to model a conditional distribution $p(\{\mathbf{s}_i\}_{i=1}^N | \{\mathbf{e}_i\}_{i=1}^N)$ given $\{\mathbf{e}_i\}_{i=1}^N$. Specifically, in the post-MLP of the self-attention block, for each \mathbf{s}_i , now the AdaLN layer takes as input a concatenation of the positional-encoded timestep $\gamma(t)$ and a feature vector $\mathcal{E}(\mathbf{e}_i)$ learned from the corresponding extrinsic parameters \mathbf{e}_i . The features $\{\mathcal{E}(\mathbf{e}_i)\}_{i=1}^N$ are learned from an additional stack of the self-attention modules encoding $\{\mathbf{e}_i\}_{i=1}^N$. Both of the noise prediction networks ϵ_θ and ϵ_ϕ are trained with the same variational bound loss with Equation 6 as follows:

$$\mathcal{L}_e(\theta) := \mathbb{E}_{t, \{\mathbf{e}\}_{i=1}^N, \epsilon} \left[\left\| \epsilon - \epsilon_\theta \left(\{\mathbf{e}^{(t)}\}_{i=1}^N, \gamma(t) \right) \right\|^2 \right] \quad (8)$$

$$\mathcal{L}_s(\phi) := \mathbb{E}_{t, \{\mathbf{s}\}_{i=1}^N, \epsilon} \left[\left\| \epsilon - \epsilon_\phi \left(\{\mathbf{s}^{(t)}\}_{i=1}^N, \gamma(t), \{\mathbf{e}^{(0)}\}_{i=1}^N \right) \right\|^2 \right] \quad (9)$$

where $\mathbf{e}^{(t)}$ and $\mathbf{s}^{(t)}$ are the extrinsic and intrinsic attributes after t -step forward process of adding Gaussian noise, respectively. Refer to the **supplementary material** for more implementation details.

5. Experiment

In this section, we demonstrate that SALAD outperforms other baselines in shape *generation* (Section 5.1) and enables intuitive *manipulation*, such as part completion (Section 5.2) and part mixing and refinement (Section 5.3), where the combination of part-level representation and diffusion models is essential. Lastly, we also demonstrate that SALAD outperforms other baselines in text-guided shape generation (Section 5.4) and can leverage part-level representation for text-guided part completion (Section 5.5).

5.1. Shape Generation

Evaluation Setup. For evaluation and comparison, we follow the settings of Hui *et al.* [27]. We use *air-*

plane and *chair* classes from the ShapeNet [6] dataset and the train-test split from Chen *et al.* [8]. The model is trained for each class. At inference time, we sample 2000 shapes for each class, and measure three evaluation metrics [1, 35] to assess quality and diversity of the generated shapes: Coverage (COV), Minimum Matching Distance (MMD), and 1-Nearest Neighbor Accuracy (1-NNA). We compare SALAD with existing 3D generative models [8, 29, 37, 20, 27].

Results. The quantitative and qualitative results, including ablation studies, are summarized in Table 1 and Figure 4. For more results, refer to the **supplementary material**. We reproduced the results of SPAGHETTI [20] and Neural Wavelet [27] using the official code, and the other quantitative results are directly borrowed from Hui *et al.* [27], marked with “*” in Table 1. (We also display the results of SPAGHETTI [20] and Neural Wavelet [27] reported by Hui *et al.* [27] in the gray-colored rows. Note that SPAGHETTI results are similar, while there is a gap in the Neural Wavelet results.) To ease qualitative comparisons in Figure 4, we retrieve the generated shapes using the same query ground truth shape and compare them.

As shown in Table 1, SALAD achieves SotA results or is on par with the baselines. In particular, we outperform Neural Wavelet [27], which is a SotA diffusion-based 3D generative model, on 1-NNA by a large margin: 65.04 vs. 57.82 for *chair* CD, and 75.77 vs. 73.92 for *airplane* CD (lower is better).

Qualitatively, SALAD produces clean high-resolution meshes with fine details as shown in Figure 4. When comparing “Diffusion of \mathbf{z} ” (in Section 4) with SPAGHETTI [20], we demonstrate that our simple latent diffusion already produces much better quality shapes than sampling \mathbf{z} from the unit Gaussian distribution as SPAGHETTI does. “Diffusion of $\{\mathbf{p}_i\}_{i=1}^N$ ” uses Transformer [65] instead of simple MLPs and outperforms “Diffusion of \mathbf{z} ”, clearly showing how our Transformer-based architecture is the key to learning the distribution of high-dimensional latents represented as a set.

When comparing our final model SALAD with “Diffusion of $\{\mathbf{p}_i\}_{i=1}^N$ ”, SALAD outperforms “Diffusion of $\{\mathbf{p}_i\}_{i=1}^N$ ” by a large margin across all metrics. It shows that our cascaded diffusion training is crucial to improve shape generation quality.

Table 1: **Quantitative comparison of shape generation.** The numbers directly from Hui *et al.* [27] are marked with *. MMD-CD scores and MMD-EMD scores are scaled by 10^3 and 10^2 , respectively. The best results are highlighted without considering the gray-colored rows. The ablation study results are presented in rows 8-9.

Id	Method	COV \uparrow		Chair MMD \downarrow		1-NNA \downarrow		COV \uparrow		Airplane MMD \downarrow		1-NNA \downarrow	
		CD	EMD	CD	EMD	CD	EMD	CD	EMD	CD	EMD	CD	EMD
1	IM-NET* [8]	56.49	54.50	11.79	14.52	61.98	63.45	61.55	62.79	3.320	8.371	76.21	76.08
2	Voxel-GAN* [29]	43.95	39.45	15.18	17.32	80.27	81.16	38.44	39.18	5.937	11.69	93.14	92.77
3	DPM* [37]	51.47	55.97	12.79	16.12	61.76	63.72	60.19	62.30	3.543	9.519	74.60	72.31
4	SPAGHETTI* [20]	49.19	51.92	14.90	15.90	70.72	68.95	58.34	58.38	4.062	8.887	78.24	77.01
5	Neural Wavelet* [27]	58.19	55.46	11.70	14.31	61.47	61.62	64.78	64.40	3.230	7.756	71.69	66.74
6	SPAGHETTI	49.48	50.22	14.7	15.85	72.34	69.46	56.86	58.83	4.260	8.930	79.36	78.86
7	Neural Wavelet	49.63	50.15	12.12	14.25	65.04	62.87	60.94	59.09	3.528	7.964	75.77	72.93
8	Diff. of \mathbf{z}	49.71	48.75	11.71	14.12	62.72	61.25	54.88	59.33	3.877	8.958	82.20	80.35
9	Diff. of $\{\mathbf{p}_i\}_{i=1}^N$	50.96	51.40	13.57	15.41	66.19	67.04	58.59	61.80	4.264	9.230	78.80	76.14
10	SALAD (Ours)	56.42	55.16	11.69	14.29	57.82	58.41	63.16	65.39	3.636	8.238	73.92	71.08

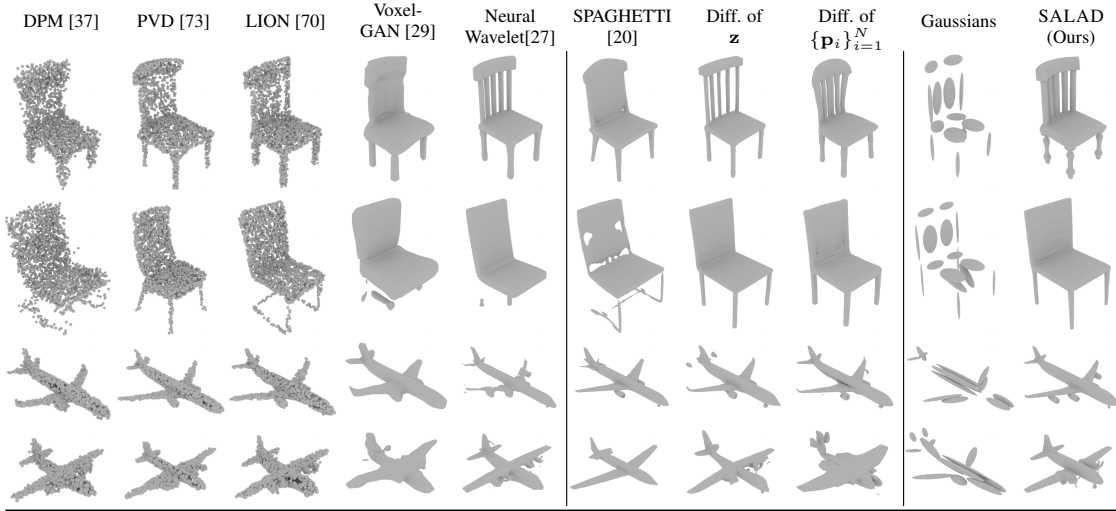


Figure 4: **Qualitative comparison of the shape generation.** Given a query ground truth shape, we retrieve the closest generated shape by measuring EMD in each method. SALAD produces highly detailed 3D shapes compared to the baselines.

5.2. Part Completion

Here, we describe how SALAD, which was trained in an *unconditional* setup, can be employed to part completion. We compare the results against the most recent diffusion model, Neural Wavelet [27] and the SotA of shape completion, ShapeFormer [66].

Experiment Setup. For completion using diffusion models, we run *guided* reverse process proposed by Meng *et al.* [39]. Specifically, given the input data $\mathbf{x} \in \mathbb{R}^d$ and a mask of the region to be reconstructed $m \in [0, 1]^d$, each step of the reverse process of the diffusion is performed as follows:

$$\begin{aligned}
 \mathbf{x}_{\text{unmasked}}^{(t-1)} &\sim \mathcal{N}(\sqrt{\bar{\alpha}^{(t)}}\mathbf{x}^{(0)}, (1 - \bar{\alpha}^{(t)})\mathbf{I}) \\
 \mathbf{x}_{\text{masked}}^{(t-1)} &\sim \mathcal{N}(\mu_{\theta}(\mathbf{x}^{(t)}, t), \beta^{(t)}\mathbf{I}) \\
 \mathbf{x}^{(t-1)} &= m \odot \mathbf{x}_{\text{unmasked}}^{(t-1)} + (1 - m) \odot \mathbf{x}_{\text{masked}}^{(t-1)}.
 \end{aligned} \tag{10}$$

Unlike previous methods such as ShapeFormer [66], this approach guarantees to preserve the unmasked region. In our experiments, we randomly remove and regenerate a semantic part of *chairs* and *airplanes*. While we can simply select $(\mathbf{e}_i, \mathbf{s}_i)$ pairs of parts we want to remove in SALAD, in feature-voxel representation like Neural Wavelet [27], it is not trivial to specify the regions that would include the completed part. This limits their generation output to only occupy the masked voxels, while a larger mask could interfere with or even break unwanted parts leading to seams in the final output. For the guided reverse process of Neural Wavelet [27] in our experiments, we use the axis-aligned bounding box of a part as a mask and transform the mask to the wavelet domain. Refer to the **supplementary material** for more details on mask construction.

We first randomly choose 100 shapes from our training set. Then, for all methods, we randomly select a se-

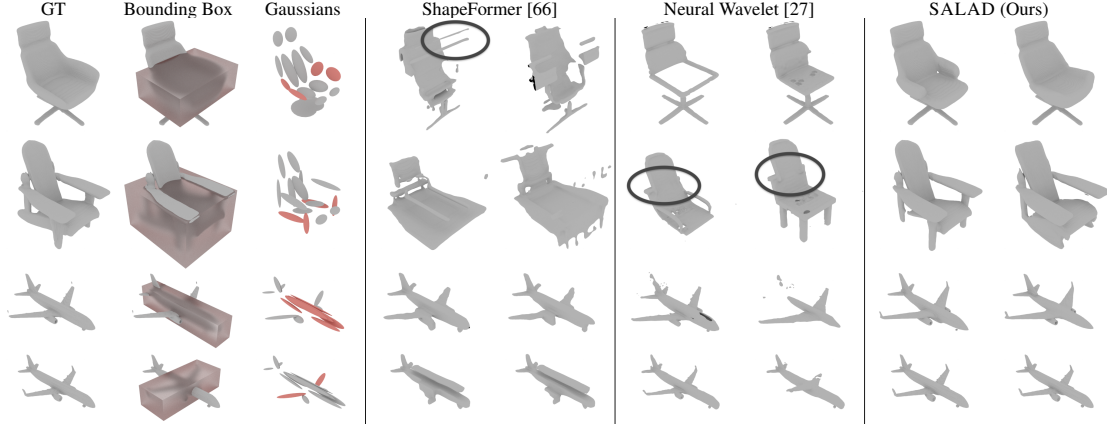


Figure 5: **Qualitative comparison of the part completion.** We examine SALAD and other baselines in part completion after ablating semantic parts or regions, highlighted in red in columns 2 and 3. SALAD produces realistic completions for missing parts. The baselines fail to preserve observed parts or introduce noticeable seams at bounding box boundaries.

mantic part from each shape and generate five variations. For quantitative comparisons, we report the reconstruction loss, MMD and FPD (Fréchet PointNet Distance) [59] indicating the quality and diversity of completions. Note that we measure MMD *from* completions *to* groundtruth shapes to quantify the proximity of the completed shapes to the groundtruth shapes. We use the official pre-trained models for ShapeFormer [66] and Neural Wavelet [27]. We also report the results from Neural Wavelet trained by ourselves.

Results. The quantitative results and qualitative results are summarized in Table 2 and Figure 5, respectively. For more results, refer to the **supplementary material**. As shown in Table 2, SALAD, trained solely for *unconditional* shape generation, outperforms the baselines in most of the metrics by large margins, especially in FPD which is the metric of how plausible the shapes are.

The qualitative results presented in Figure 5 further manifests the advantages of employing a part-level 3D representation in SALAD. In row 1 of Figure 5, ShapeFormer [66] introduces noticeable artifacts at the back of the chair that lies outside the binary mask (column 2). In contrast, SALAD completes the seat seamlessly while preserving the other parts, benefiting from the spatial correspondence between the binary mask and the shape representation. Even with such spatial correspondences, the limitation of specifying regions instead of parts persists in Neural Wavelet [27]. In particular, the row 2 of Figure 5 shows visible seams at the bounding box boundary while SALAD generates the missing part consistent with the surrounding parts.

5.3. Part Mixing and Refinement

While Hertz *et al.* [20] demonstrates creating new shapes by combining parts from existing shapes, naively mixing part representations is prone to produce failure cases

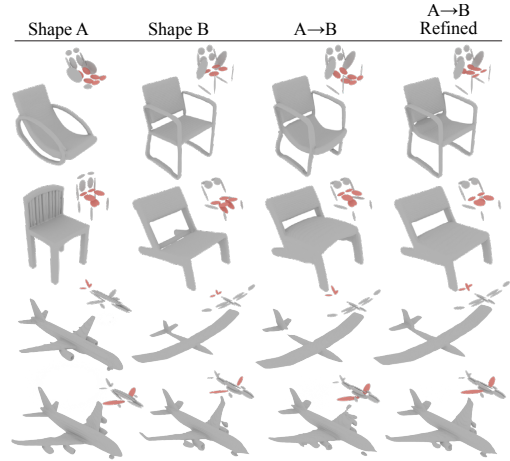


Figure 6: **Qualitative results of Part mixing and refinement.** SALAD improves quality of part mixing outputs.

as illustrated in Figure 6 and Figure 1. Cracks or discontinuities at joint regions are one type of failure case as shown in row 3 of Figure 6 and (b) of Figure 1. Another type of failures is the dissonance between combined parts that results in undesired distortions or the vanishing of parts. SALAD can remedy this issue by refining both the extrinsic and intrinsic vectors through the guided reverse process. Refer to the **supplementary material** for more qualitative results and **quantitative comparisons**.

5.4. Text-Guided Shape Generation

We further demonstrate SALAD can perform *conditional* generation, specifically generating 3D shapes given an input text. To condition a text to the model, we concatenate a language feature and an input of AdaLN, $\gamma(t)$, and optionally $\mathcal{E}(\mathbf{e}_i)$. We experiment with the text and shape pair dataset from ShapeGlot [2] and compare the gener-

Table 2: **Quantitative comparison of part completion.** The metrics based on CD and EMD are scaled by 10^3 and 10^2 , respectively. The result from the pre-trained Neural Wavelet is marked with *.

Method	<i>reverse</i> -MMD ↓		Chair Reconstruction ↓		FPD ↓	<i>reverse</i> -MMD ↓		Airplane Reconstruction ↓		FPD ↓
	CD	EMD	CD	EMD		CD	EMD	CD	EMD	
ShapeFormer [66]	32.83	22.8	55.05	25.49	83.56	5.43	10.87	10.83	11.81	79.18
Neural Wavelet* [27]	13.46	15.65	8.72	12.92	18.83	3.81	9.07	3.84	8.85	31.38
Neural Wavelet	11.87	15.07	8.93	12.44	18.78	3.56	8.79	3.90	9.02	36.17
SALAD (Ours)	12.1	14.56	5.45	9.22	16.75	3.55	8.68	2.12	6.53	29.44









Text	AutoSDF [43]		SALAD (Ours)	
“chair has round arms and wheels.”				
“its the one with gaps in the back.”				

Figure 7: **Qualitative comparison of text-guided generation.** SALAD generates high-quality 3D shapes conforming to the input texts compared to AutoSDF [43].

Table 3: **Quantitative comparison of text-guided generation.** Overall, SALAD achieves better performance than AutoSDF. Specifically, it improves FPD by a large margin.

Methods	CLIP-S ↑	NEP ↑	FPD ↓
AutoSDF [43]	30.98	38.98	31.53
SALAD (Ours)	30.92	42.22	4.043

ation quality of our text-conditioned model with the one by AutoSDF [43], which is the SotA text-to-shape generative model. The train-test split used in AutoSDF is used. Also, following AutoSDF, we measure the following three metrics for the evaluation: CLIP-Similarity-Score (CLIP-S) [55], Neural-Evaluator-Preference (NEP), and Fréchet Point Cloud Distance (FPD) [59].

NEP proposed by Mittal *et al.* [43] is a preference rate obtained from a neural evaluator. The neural evaluator is pre-trained on a text-conditioned binary classification task where the model distinguishes the target shape corresponding to the input text. Since the neural evaluator used in AutoSDF has not publicly been released, we train our neural evaluator based on PartGlott [30], a simpler architecture trained only on point clouds without images. More details of the experiment setup is in the **supplementary material**.

As shown in Table 3, our generated shapes are more preferred by the neural evaluator over the shapes generated by AutoSDF. Also, Figure 7 and FPD results reflect that SALAD produces more plausible shapes, and our generated shapes conform to given texts more than the shapes of AutoSDF.

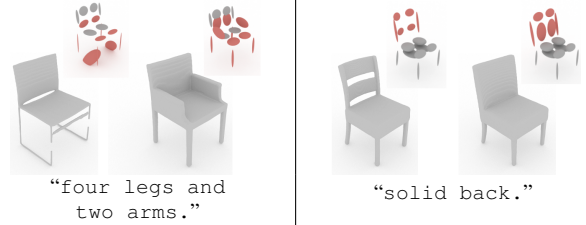


Figure 8: **Qualitative results of text-guided part completion.** The part of the left mesh selected by GAUSSGLOT, highlighted by red, is completed to fit a given text by a reverse process of text-guided SALAD.

5.5. Text-Guided Part Completion

We further demonstrate how SALAD can be integrated with a text-driven semantic part segmentation network to aid user interactive shape editing. Following PartGlott [30] architecture, we design GAUSSGLOT, a model that uses $\{e_i\}_{i=1}^N$ as a part representation and predicts semantic part labels of those from texts. More details of GAUSSGLOT architecture and training results can be found in the **supplementary material**. Figure 8 shows examples that the parts of input shapes selected by GAUSSGLOT are completed according to given texts by a reverse process of text-conditioned SALAD introduced in Section 5.4. It demonstrates that users can freely manipulate 3D shapes with texts in an end-to-end manner by leveraging SALAD with GAUSSGLOT.

6. Conclusion

We presented SALAD, a cascaded 3D diffusion model for part-level implicit representation. Compared with other 3D diffusion models, our model achieves the best quality in shape generation and also is versatile to be exploited in diverse part-level shape manipulation tasks such as completing, mixing, and text-guided editing. Diffusion on the disentangled representation that allows picking individual parts without specifying a bounding region in the 3D space was the key to fully utilizing the zero-shot manipulation capability of the diffusion models. In future work, we plan to further investigate the diffusion models on part-level representations with different primitives and parametrization for parts.

References

- [1] Panos Achlioptas, Olga Diamanti, Ioannis Mitliagkas, and Leonidas Guibas. Learning representations and generative models for 3d point clouds. In *ICML*, 2018.
- [2] Panos Achlioptas, Judy Fan, Robert Hawkins, Noah Goodman, and Leonidas Guibas. ShapeGlot: Learning language for shape differentiation. In *ICCV*, 2019.
- [3] Valentin De Bortoli, Emile Mathieu, Michael John Hutchinson, James Thornton, Yee Whye Teh, and Arnaud Doucet. Riemannian score-based generative modelling. In *NeurIPS*, 2022.
- [4] Ruojin Cai, Guandao Yang, Hadar Averbuch-Elor, Zekun Hao, Serge Belongie, Noah Snively, and Bharath Hariharan. Learning gradient fields for shape generation. In *ECCV*, 2020.
- [5] Eric R. Chan, Connor Z. Lin, Matthew A. Chan, Koki Nagano, Boxiao Pan, Shalini De Mello, Orazio Gallo, Leonidas Guibas, Jonathan Tremblay, Sameh Khamis, Tero Karras, and Gordon Wetzstein. Efficient geometry-aware 3D generative adversarial networks. In *CVPR*, 2022.
- [6] 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.
- [7] Zhiqin Chen, Andrea Tagliasacchi, and Hao Zhang. Bsp-net: Generating compact meshes via binary space partitioning. In *CVPR*, 2020.
- [8] Zhiqin Chen and Hao Zhang. Learning implicit fields for generative shape modeling. In *CVPR*, 2019.
- [9] Yen-Chi Cheng, Hsin-Ying Lee, Sergey Tulyakov, Alexander Schwing, and Liangyan Gui. SDFusion: Multimodal 3d shape completion, reconstruction, and generation. *arXiv preprint arXiv:2212.04493*, 2022.
- [10] Gene Chou, Yuval Bahat, and Felix Heide. Diffusionsdf: Conditional generative modeling of signed distance functions. *arXiv preprint arXiv:2211.13757*, 2022.
- [11] Hyungjin Chung, Byeongsu Sim, and Jong Chul Ye. Improving diffusion models for inverse problems using manifold constraints. 2022.
- [12] Albert Cohen. *Biorthogonal Wavelets*. 1993.
- [13] Boyang Deng, Kyle Genova, Soroosh Yazdani, Sofien Bouaziz, Geoffrey Hinton, and Andrea Tagliasacchi. Cvxnet: Learnable convex decomposition. In *CVPR*, 2020.
- [14] Prafulla Dhariwal and Alexander Nichol. Diffusion models beat gans on image synthesis. *NeurIPS*, 2021.
- [15] Rao Fu, Xiao Zhan, Yiwen Chen, Daniel Ritchie, and Srinath Sridhar. Shapecrafter: A recursive text-conditioned 3d shape generation model. *arXiv preprint arXiv:2207.09446*, 2022.
- [16] Jun Gao, Tianchang Shen, Zian Wang, Wenzheng Chen, Kangxue Yin, Daiqing Li, Or Litany, Zan Gojcic, and Sanja Fidler. GET3D: A generative model of high quality 3d textured shapes learned from images. In *NeurIPS*, 2022.
- [17] Kyle Genova, Forrester Cole, Avneesh Sud, Aaron Sarna, and Thomas Funkhouser. Local deep implicit functions for 3d shape. In *CVPR*, 2020.
- [18] Kyle Genova, Forrester Cole, Daniel Vlasic, Aaron Sarna, William T Freeman, and Thomas Funkhouser. Learning shape templates with structured implicit functions. In *ICCV*, 2019.
- [19] Zekun Hao, Hadar Averbuch-Elor, Noah Snively, and Serge Belongie. DualSDF: Semantic shape manipulation using a two-level representation. In *CVPR*, 2020.
- [20] Amir Hertz, Or Perel, Raja Giryes, Olga Sorkine-Hornung, and Daniel Cohen-Or. SPAGHETTI: Editing implicit shapes through part aware generation. *ACM TOG*, 2022.
- [21] Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. In *NeurIPS*, 2019.
- [22] Jonathan Ho, Chitwan Saharia, William Chan, David J Fleet, Mohammad Norouzi, and Tim Salimans. Cascaded diffusion models for high fidelity image generation. *JMLR*, 2022.
- [23] Jonathan Ho and Tim Salimans. Classifier-free diffusion guidance. In *NeurIPS 2021 Workshop on Deep Generative Models and Downstream Applications*, 2021.
- [24] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 1997.
- [25] Jingyu Hu, Ka-Hei Hui, Zhengzhe Liu, Ruihui Li, and Chi-Wing Fu. Neural wavelet-domain diffusion for 3d shape generation, inversion, and manipulation. *arXiv preprint arXiv:2302.00190*, 2023.
- [26] Ka-Hei Hui, Ruihui Li, Jingyu Hu, and Chi-Wing Fu. Neural template: Topology-aware reconstruction and disentangled generation of 3d meshes. In *CVPR*, 2022.
- [27] Ka-Hei Hui, Ruihui Li, Jingyu Hu, and Chi-Wing Fu. Neural wavelet-domain diffusion for 3d shape generation. In *SIG-GRAPH ASIA*, 2022.
- [28] Moritz Ibing, Isaak Lim, and Leif Kobbelt. 3d shape generation with grid-based implicit functions. In *CVPR*, 2021.
- [29] Marian Kleineberg, Matthias Fey, and Frank Weichert. Adversarial generation of continuous implicit shape representations. *Eurographics - Short Papers*, 2020.
- [30] Juil Koo, Ian Huang, Panos Achlioptas, Leonidas J Guibas, and Minhyuk Sung. PartGlot: Learning shape part segmentation from language reference games. In *CVPR*, 2022.
- [31] Samuli Laine, Janne Hellsten, Tero Karras, Yeongho Seol, Jaakko Lehtinen, and Timo Aila. Modular primitives for high-performance differentiable rendering. *ACM TOG*, 2020.
- [32] Adam Leach, Sebastian M Schmon, Matteo T. Degiacomi, and Chris G. Willcocks. Denoising diffusion probabilistic models on SO(3) for rotational alignment. In *ICLR 2022 Workshop on Geometrical and Topological Representation Learning*, 2022.
- [33] Muheng Li, Yueqi Duan, Jie Zhou, and Jiwen Lu. Diffusion-sdf: Text-to-shape via voxelized diffusion. *arXiv preprint arXiv:2212.03293*, 2022.
- [34] Connor Z. Lin, Niloy J. Mitra, Gordon Wetzstein, Leonidas Guibas, and Paul Guerrero. NeuForm: Adaptive overfitting for neural shape editing. In *NeurIPS*, 2022.
- [35] David Lopez-Paz and Maxime Oquab. Revisiting classifier two-sample tests. In *ICLR*, 2017.
- [36] Andreas Lugmayr, Martin Danelljan, Andres Romero, Fisher Yu, Radu Timofte, and Luc Van Gool. Repaint: Inpainting

- using denoising diffusion probabilistic models. In *CVPR*, 2022.
- [37] Shitong Luo and Wei Hu. Diffusion probabilistic models for 3d point cloud generation. In *CVPR*, 2021.
- [38] Prasanta Chandra Mahalanobis. On the generalised distance in statistics. In *Proceedings of the National Institute of Sciences of India*, 1936.
- [39] Chenlin Meng, Yutong He, Yang Song, Jiaming Song, Jiajun Wu, Jun-Yan Zhu, and Stefano Ermon. SDEdit: Guided image synthesis and editing with stochastic differential equations. In *ICLR*, 2022.
- [40] Lars Mescheder, Michael Oechsle, Michael Niemeyer, Sebastian Nowozin, and Andreas Geiger. Occupancy Networks: Learning 3d reconstruction in function space. In *CVPR*, 2019.
- [41] Midjourney. Midjourney. <https://www.midjourney.com/>.
- [42] Ben Mildenhall, Pratul P. Srinivasan, Matthew Tancik, Jonathan T. Barron, Ravi Ramamoorthi, and Ren Ng. NeRF: Representing scenes as neural radiance fields for view synthesis. In *ECCV*, 2020.
- [43] Paritosh Mittal, Yen-Chi Cheng, Maneesh Singh, and Shubham Tulsiani. Autosdf: Shape priors for 3d completion, reconstruction and generation. In *CVPR*, 2022.
- [44] Kaichun Mo, Paul Guerrero, Li Yi, Hao Su, Peter Wonka, Niloy Mitra, and Leonidas Guibas. Structurenets: Hierarchical graph networks for 3d shape generation. *ACM TOG*, 2019.
- [45] Kaichun Mo, Shilin Zhu, Angel X Chang, Li Yi, Subarna Tripathi, Leonidas J Guibas, and Hao Su. Partnet: A large-scale benchmark for fine-grained and hierarchical part-level 3d object understanding. In *CVPR*, 2019.
- [46] Gimin Nam, Mariem Khelifi, Andrew Rodriguez, Alberto Tono, Linqi Zhou, and Paul Guerrero. 3d-ldm: Neural implicit 3d shape generation with latent diffusion models. *arXiv preprint arXiv:2212.00842*, 2022.
- [47] Charlie Nash, Yaroslav Ganin, SM Ali Eslami, and Peter Battaglia. Polygen: An autoregressive generative model of 3d meshes. In *ICML*, 2020.
- [48] Alex Nichol, Heewoo Jun, Prafulla Dhariwal, Pamela Mishkin, and Mark Chen. Point-e: A system for generating 3d point clouds from complex prompts. *arXiv preprint arXiv:2212.08751*, 2022.
- [49] Jeong Joon Park, Peter Florence, Julian Straub, Richard Newcombe, and Steven Lovegrove. DeepSDF: Learning continuous signed distance functions for shape representation. In *CVPR*, 2019.
- [50] Despoina Paschalidou, Angelos Katharopoulos, Andreas Geiger, and Sanja Fidler. Neural parts: Learning expressive 3d shape abstractions with invertible neural networks. In *CVPR*, 2021.
- [51] Despoina Paschalidou, Ali Osman Ulusoy, and Andreas Geiger. Superquadrics revisited: Learning 3d shape parsing beyond cuboids. In *CVPR*, 2019.
- [52] Ethan Perez, Florian Strub, Harm de Vries, Vincent Dumoulin, and Aaron C. Courville. Film: Visual reasoning with a general conditioning layer. In *AAAI*, 2018.
- [53] Charles R Qi, Hao Su, Kaichun Mo, and Leonidas J Guibas. Pointnet: Deep learning on point sets for 3d classification and segmentation. In *CVPR*, 2017.
- [54] Charles R Qi, Li Yi, Hao Su, and Leonidas J Guibas. Pointnet++: Deep hierarchical feature learning on point sets in a metric space. In *NeurIPS*, 2017.
- [55] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, et al. Learning transferable visual models from natural language supervision. In *ICML*, 2021.
- [56] Aditya Ramesh, Prafulla Dhariwal, Alex Nichol, Casey Chu, and Mark Chen. Hierarchical text-conditional image generation with clip latents. *arXiv preprint arXiv:2204.06125*, 2022.
- [57] Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-resolution image synthesis with latent diffusion models. In *CVPR*, 2022.
- [58] Peter H Schönemann. A generalized solution of the orthogonal procrustes problem. *Psychometrika*, 1966.
- [59] Dong Wook Shu, Sung Woo Park, and Junseok Kwon. 3d point cloud generative adversarial network based on tree structured graph convolutions. In *ICCV*, 2019.
- [60] J. Ryan Shue, Eric Ryan Chan, Ryan Po, Zachary Ankner, Jiajun Wu, and Gordon Wetzstein. 3d neural field generation using triplane diffusion. *arXiv preprint arXiv:2211.16677*, 2022.
- [61] Chun-Yu Sun, Qian-Fang Zou, Xin Tong, and Yang Liu. Learning adaptive hierarchical cuboid abstractions of 3d shape collections. *ACM TOG*, 2019.
- [62] Yongbin Sun, Yue Wang, Ziwei Liu, Joshua Siegel, and Sanjay Sarma. Pointgrow: Autoregressively learned point cloud generation with self-attention. In *WACV*, 2020.
- [63] Shubham Tulsiani, Hao Su, Leonidas J. Guibas, Alexei A. Efros, and Jitendra Malik. Learning shape abstractions by assembling volumetric primitives. In *CVPR*, 2017.
- [64] Diego Valsesia, Giulia Fracastoro, and Enrico Magli. Learning localized generative models for 3d point clouds via graph convolution. In *ICLR*, 2019.
- [65] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *NeurIPS*, 2017.
- [66] Xingguang Yan, Liqiang Lin, Niloy J Mitra, Dani Lischinski, Daniel Cohen-Or, and Hui Huang. Shapeformer: Transformer-based shape completion via sparse representation. In *CVPR*, 2022.
- [67] Kaizhi Yang and Xuejin Chen. Unsupervised learning for cuboid shape abstraction via joint segmentation from point clouds. *ACM TOG*, 2021.
- [68] 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 TOG*, 2016.
- [69] Sihyun Yu, Kihyuk Sohn, Subin Kim, and Jinwoo Shin. Video probabilistic diffusion models in projected latent space. *arXiv preprint arXiv:2302.07685*, 2023.

- [70] 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 *NeurIPS*, 2022.
- [71] Biao Zhang, Matthias Nießner, and Peter Wonka. 3dirlg: Irregular latent grids for 3d generative modeling. *arXiv preprint arXiv:2205.13914*, 2022.
- [72] Xin-Yang Zheng, Yang Liu, Peng-Shuai Wang, and Xin Tong. SDF-StyleGAN: Implicit sdf-based StyleGAN for 3d shape generation. In *Comput. Graph. Forum (SGP)*, 2022.
- [73] Linqi Zhou, Yilun Du, and Jiajun Wu. 3d shape generation and completion through point-voxel diffusion. In *ICCV*, 2021.

Appendix

A.1. Quantitative Results of Part Mixing and Refinement

Experiment Setup. To demonstrate the refinement capability of SALAD, we conduct quantitative comparisons between the part mixing outputs from SPAGHETTI [20] and the refined outputs from SALAD. For evaluation, we randomly select 100 pairs of shapes from the training set and swap a semantic part, for all parts that two shapes in a pair have in common. Swapping a part between two shapes results in two mixed shapes for each pair. The numbers of the shapes resulting from part mixing are 606, 670, and 400 for *chair*, *airplane*, and *table* classes, respectively. The mixed shapes are refined using the method introduced in Section 5.2 with diffusion timestep $t = 10$. We evaluate the same metrics used in Section 5.1 using the test set provided by Chen *et al.* [8] and report the results in Table A4.

Results. As indicated in the metrics reported in Table A4, the quality of mixed shapes are further improved after the refinement step. We particularly observe noticeable gaps in 1-NNA across all shape classes. More *qualitative* results are reported in Section A.6.

Table A4: **Quantitative comparison of part mixing.** After combining parts from two different shapes, our SALAD further refines the outputs by adjusting mixed parts. The refinement step brings noticeable improvements in 1-NNA.

Method	Chair						Airplane						Table					
	COV \uparrow		MMD \downarrow		1-NNA \downarrow		COV \uparrow		MMD \downarrow		1-NNA \downarrow		COV \uparrow		MMD \downarrow		1-NNA \downarrow	
	CD	EMD	CD	EMD	CD	EMD	CD	EMD	CD	EMD	CD	EMD	CD	EMD	CD	EMD	CD	EMD
SPAGHETTI [20]	42.24	44.06	18.18	17.53	73.18	74.26	39.85	42.09	5.34	10.05	80.22	78.88	31.50	32.25	19.68	18.02	86.62	87.62
SALAD (Ours)	40.59	43.89	17.21	16.96	69.97	68.23	40.15	40.75	5.24	9.72	77.61	76.27	44.25	43.25	17.27	16.98	66.25	69.62

A.2. SALAD Implementation Details

As discussed in Section 3.2, an extrinsic vector \mathbf{e}_i is represented by $\{\mathbf{c}_i, \lambda_i^1, \lambda_i^2, \lambda_i^3, \mathbf{u}_i^1, \mathbf{u}_i^2, \mathbf{u}_i^3, \pi_i\}$, where the eigenvectors $\{\mathbf{u}_i^j\}_{j=1}^3$ must be orthogonal to each other. Therefore, the diffusion processes for $\{\mathbf{e}_i\}_{i=1}^N$ need to model distributions in a product space of an orthogonal group $O(3)$ and Euclidean group, not in the Euclidean space. Recent work [32, 3] introduce diffusion models on Lie group or its product space, however, we empirically find that learning diffusion without considering the orthogonality also performs well. It is ensured only at the test time by taking the projection of the generated eigenvectors $\mathbf{U}_i = [\mathbf{u}_i^1, \mathbf{u}_i^2, \mathbf{u}_i^3]$ to $O(3)$ space. We follow Schönemann [58] and project \mathbf{U}_i as

$$\tilde{\mathbf{U}}_i = [\tilde{\mathbf{u}}_i^1, \tilde{\mathbf{u}}_i^2, \tilde{\mathbf{u}}_i^3] = \mathbf{A}\mathbf{B}^T, \quad (11)$$

where $\mathbf{U}_i = \mathbf{A}\mathbf{\Sigma}\mathbf{B}^T$ is a singular value decomposition of \mathbf{U}_i . We also clip negative eigenvalues in $\{\lambda_i^j\}_{j=1}^3$ to 1×10^{-4} since the covariance matrix is positive-definite.

We normalize elements of \mathbf{e}_i to avoid arbitrary high-variance latent space. Specifically, during the training of “Diffusion of $\{\mathbf{e}_i\}_{i=1}^N$ ”, we normalize π_i and $\{\lambda_i^j\}_{j=1}^3$ using element-wise means and standard deviations pre-computed from all training data. At test time, we re-scale these elements by the means and the standard deviations. We do not apply normalization to the others.

The Transformer-based network of SALAD introduced in Section 4 consists of an embedding layer, which maps an input to 512-dimensional embeddings, and 6 Transformer blocks. Each Transformer block is a stack of a self-attention block and an MLP, each of which is followed by an AdaLN layer. We set the dimension of the output of the positional encoding $\gamma(\cdot)$ to 128.

As SALAD consists of two diffusion models, each trained for 5000 epochs, we train the baselines for 10,000 epochs for a fair comparison. We use a batch size of 64 and an initial learning rate 10^{-4} with a polynomial decaying scheduler (power=0.999). The diffusion process is configured with $T = 1000$, $\beta^{(1)} = 10^{-4}$, and $\beta^{(T)} = 0.05$.

A.3. Experiment Details

In this section, we provide details of the experiments whose results are reported in the main paper.

A.3.1 Details on Part Completion Experiment Setup — Section 5.2

As mentioned in Section 5.2, part completion via a *guided* reverse process [39] requires binary masks indicating the parts to be ablated. We describe how such masks are constructed for SALAD and Neural Wavelet [27] in this section.

SALAD. We define a binary mask $m \in \{0, 1\}^N$ for pairs $\{(\mathbf{e}_i, \mathbf{s}_i)\}_{i=1}^N$ to have value 0 at completed parts, 1 otherwise. To this end, we first *transfer* the part labels of the annotated point clouds from ShapeNet [6] dataset to each $(\mathbf{e}_i, \mathbf{s}_i)$. Assume a point cloud $\{(\mathbf{x}_j, l_j)\}_{j=1}^K$ of K points where $\mathbf{x}_j \in \mathbb{R}^3$ and $l_j \in \{1, 2, \dots, L\}$, denote 3D coordinate and part label of j -th point, respectively. Each $(\mathbf{e}_i, \mathbf{s}_i)$ is assigned a part label $l_i \in \{1, 2, \dots, L\}$ based on the proximity of \mathbf{e}_i to the points $\{\mathbf{x}_j\}_{j=1}^K$. Since \mathbf{e}_i parameterizes a Gaussian distribution in 3D space, we employ Mahalanobis distance [38] as a distance measure. For each Gaussian represented by \mathbf{e}_i , we compute the distance to every point \mathbf{x}_j and select the closest 100 points. We then count the number of part label occurrences over the points and assign the most frequently occurred label to the pair.

Having assigned the part labels to each of $\{(\mathbf{e}_i, \mathbf{s}_i)\}_{i=1}^N$, we define a mask m selecting a part whose label is l as

$$m_i = \begin{cases} 0 & \text{if } l_i = l \\ 1 & \text{otherwise} \end{cases}, \quad (12)$$

where m_i denotes the i -th element of m .

Neural Wavelet [27]. Note that there is neither a publicly available official code nor detailed instructions for shape manipulation using Neural Wavelet [27]. Although a concurrent work of ours, Hu *et al.* [25], demonstrates shape manipulation using Neural Wavelet, it does not provide a detailed implementation.

Following Hui *et al.* [27], we derive the wavelet coefficients of the shapes in our training set. We compute signed distance functions (SDFs) of the shapes and truncate their values into $[-0.1, 0.1]$. We denote S the resulting truncated signed distance function (TSDF) of a shape. We leverage Biorthogonal wavelet-6-8 filter [12] to decompose S into a coarse wavelet coefficient volume at a scale 3 (C^3) and a detail wavelet coefficient volume at a scale 2 (D^2). Refer to Hui *et al.* [27] for details on preprocessing.

We then aim to derive binary masks for C^3 , necessary for leveraging pre-trained Neural Wavelet [27] for part completion. Note that selecting a part to complete is a *nontrivial* task for a voxel-based representation adapted by Neural Wavelet, as opposed to SALAD where we can define binary masks for $\{(\mathbf{e}_i, \mathbf{s}_i)\}_{i=1}^N$ to select parts directly. As one solution, we compute bounding boxes enclosing semantic parts of 3D shapes, and use them to designate the *regions* to complete. Such bounding boxes are used to compute binary masks for C^3 via a heuristic based on the property of wavelet transforms extracting local spectral information. Through experiments, we empirically find a set of wavelet coefficients that vary when the TSDF values in a 3D volume are set to 0.1 (i.e., outside of a shape). For instance, we set the TSDF values in the bounding box enclosing the back of a chair to 0.1 to discover a set of wavelet coefficients corresponding to the part. We assign 0 to the coefficients whose amount of change is above a threshold δ and 1 to the others.

Rigorously, let $M \in \{0, 1\}^{256^3}$ denote a binary voxel grid of the same resolution as S with 0 indicating the semantic part of interest and 1 otherwise. Such M is derived from a bounding box enclosing a semantic part of a 3D shape, and is used to derive a *masked* TSDF S^* defined as

$$S_v^* = \begin{cases} 0.1 & \text{if } M_v = 0 \\ S_v & \text{otherwise} \end{cases}, \quad (13)$$

for all $v \in \{(0, 0, 0), (0, 0, 1), \dots, (255, 255, 255)\}$. After marking all values inside a bounding box as *outside*, we obtain the wavelet coefficients C^{3*} via forward wavelet transform. A mask m for C^3 is then defined as

$$m_{v'} = \begin{cases} 0 & \text{if } |C_{v'}^{3*} - C_{v'}^3| > \delta \\ 1 & \text{otherwise} \end{cases}. \quad (14)$$

for all $v' \in \{(0, 0, 0), (0, 0, 1), \dots, (47, 47, 47)\}$. Here, we use $\delta = 0.001$.

ShapeFormer [66]. As discussed in Section 5.2, after constructing the axis-aligned bounding box of a part, we make a partial point cloud by masking out the points inside the bounding box, and pass it to ShapeFormer [66] as an input.

A.3.2 Details on Text-Guided Shape Generation — Section 5.4

Implementation Details of Text-Conditioned SALAD. We impose text conditions on both the first and the second phase models by feeding text features from our text encoder. We use LSTM [24] for the text encoder and train it jointly with the first and the second phase models. We also apply the classifier-free guidance [23]. More precisely, we jointly train a conditional diffusion model $\epsilon_\theta(\mathbf{x}^{(t)}, t, \mathbf{c})$ and an unconditional diffusion model $\epsilon_\theta(\mathbf{x}^{(t)}, t, \emptyset)$, where \mathbf{c} denotes a condition feature vector and \emptyset is a null condition vector. We randomly set \mathbf{c} to \emptyset with a 20% dropout probability during training. To make \emptyset , we feed an empty sequence as an input text and zero vectors for $\{\mathcal{E}(\mathbf{e}_i)\}_{i=1}^N$. \mathbf{c} is solely a text feature for the first phase model. For the second phase model conditioned on the features from extrinsic vectors $\{\mathbf{e}_i\}_{i=1}^N$, we use the concatenation of the features and a text feature as a condition.

At sampling time, the noise prediction is adjusted by an extrapolation between the noise prediction of the conditional diffusion model and the unconditional diffusion model as follows:

$$\tilde{\epsilon}_t = (1 + w)\epsilon_\theta(\mathbf{x}^{(t)}, t, \mathbf{c}) - w\epsilon_\theta(\mathbf{x}^{(t)}, t, \emptyset), \quad (15)$$

where $\tilde{\epsilon}_t$ is the noise prediction with the classifier-free guidance applied, and w is a hyperparameter controlling guidance strength. We use $w = 2$ for sampling.

Experiment Setup. To measure Neural-Evaluator-Preference (NEP) discussed in Section 5.4, we leverage a modified PartGlott [30] for a neural evaluator. The modified architecture takes point clouds as inputs instead of super-segments. Refer to the PartGlott [30] paper for more details. We adapt the training and test set of PartGlott [30] to create binary classification examples. The modified PartGlott achieves 73.98% test accuracy on the binary classification. Following Mittal *et al.* [43], we consider an example to be confused if the absolute difference between the neural evaluator’s confidence is ≤ 0.2 .

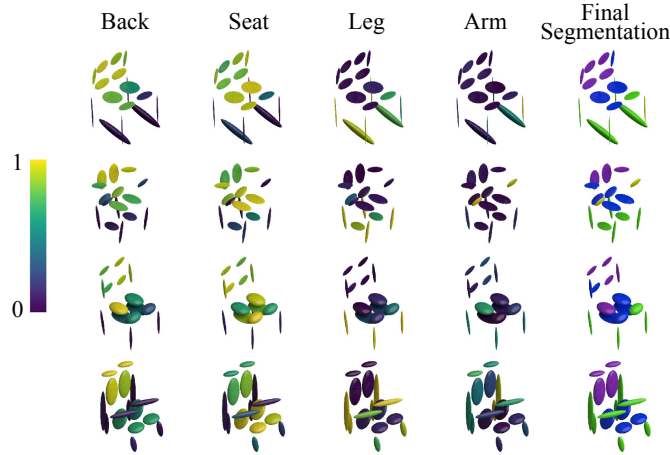


Figure A9: **GAUSSGLOT qualitative results.** The attention maps for each semantic part achieved by GAUSSGLOT are shown in the left columns of the figure. The colors of the attention maps change from dark blue to yellow as the attention weights increase from 0 to 1. The final part segmentation results are depicted in the rightmost column of the figure, where purple, blue, green, and yellow indicate *back*, *seat*, *leg*, and *arm*, respectively.

A.3.3 Details on GaussGlot — Section 5.5

Inspired by Koo *et al.* [30], we design a text-driven self-supervised semantic part segmentation network, GAUSSGLOT, where a set of Gaussian primitives is employed as super-segments. As discussed in Section 5.4, PartGlott is a neural evaluator that classifies shapes from a query text. While solving this text-conditioned shape classification, PartGlott learns semantic part

segmentation in an unsupervised manner by learning the attention maps between the input text and the super-segments. Refer to the PartGlott [30] paper for more details. Specifically, we train GAUSSGLOT with $\{\mathbf{e}_i\}_{i=1}^N$ excluding π_i elements which is inessential to define 3D Gaussian primitives. Based on the architecture of PartGlott, 15-dimensional Gaussian parameters are mapped to 256-dimensional features through MLPs. We embed text tokens into 128 dimensions and use LSTM as a text encoder with 256-dimensional hidden states. Our trained GAUSSGLOT achieves 76.03% test accuracy and 56.85% mIoU. Qualitative part segmentation examples and the attention maps of each semantic part from GAUSSGLOT can be found in Figure A9.

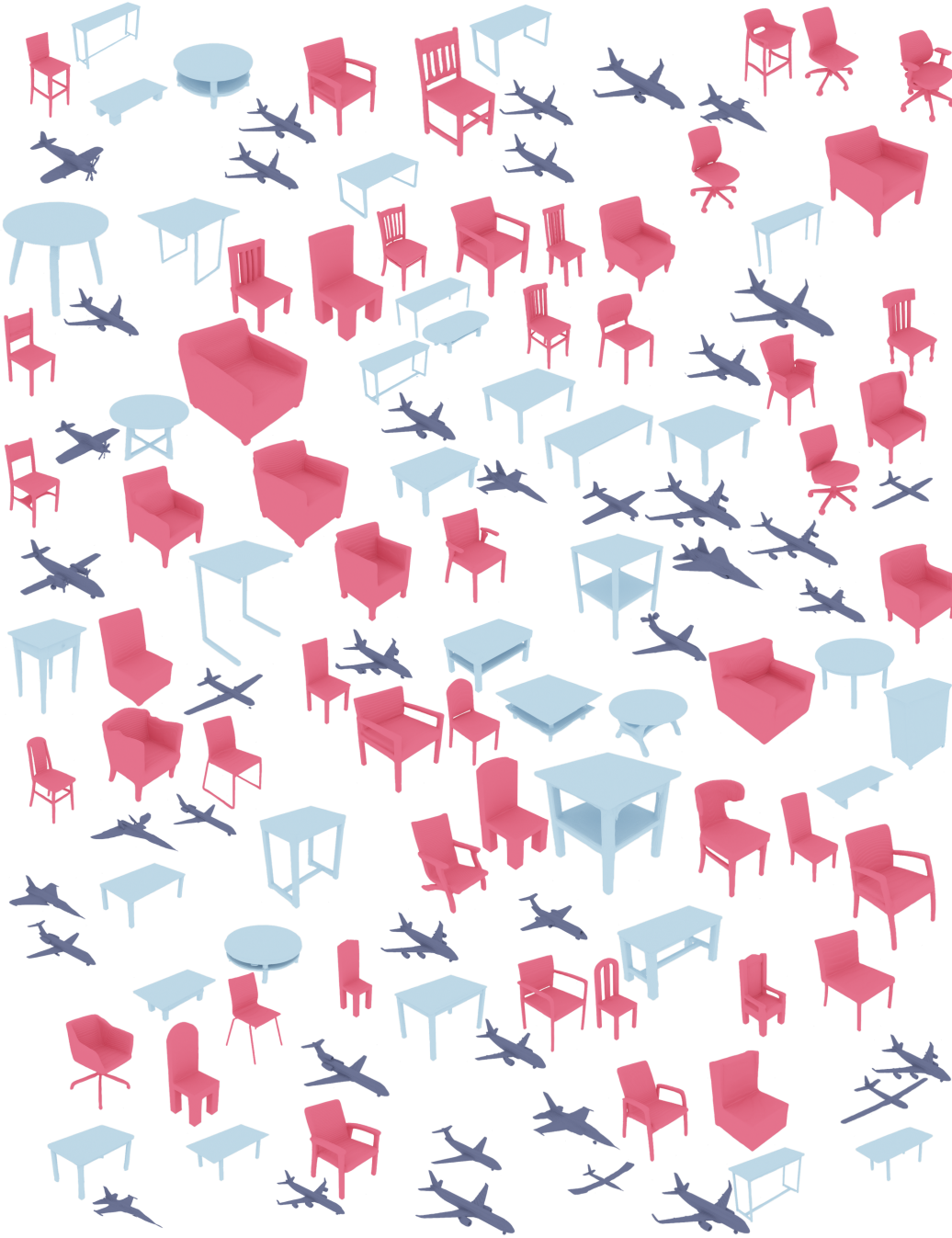




















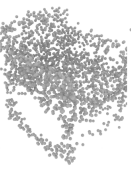
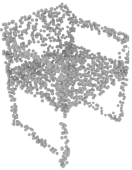
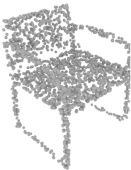








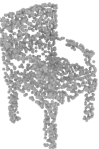








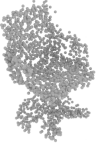
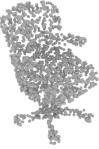
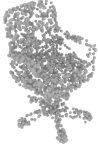














































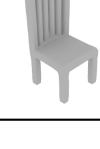
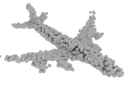
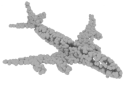
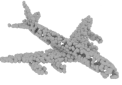







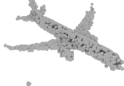
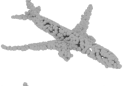
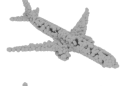







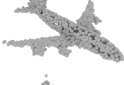









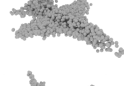
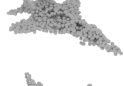
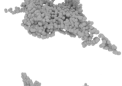







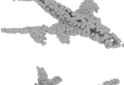
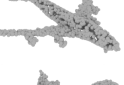
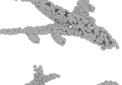







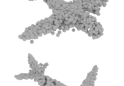
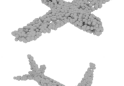








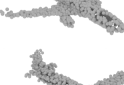
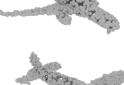
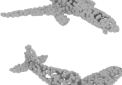







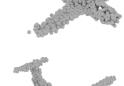
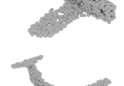
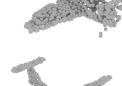







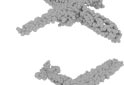
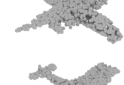
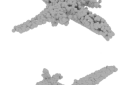







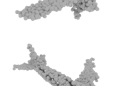
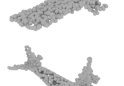








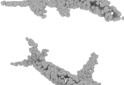
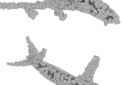









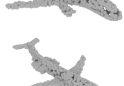










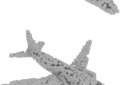


















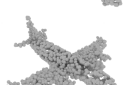










Figure A10: A visual gallery of *airplanes*, *chairs*, and *tables* generated by SALAD.

A.4. More Qualitative Comparisons on Shape Generation

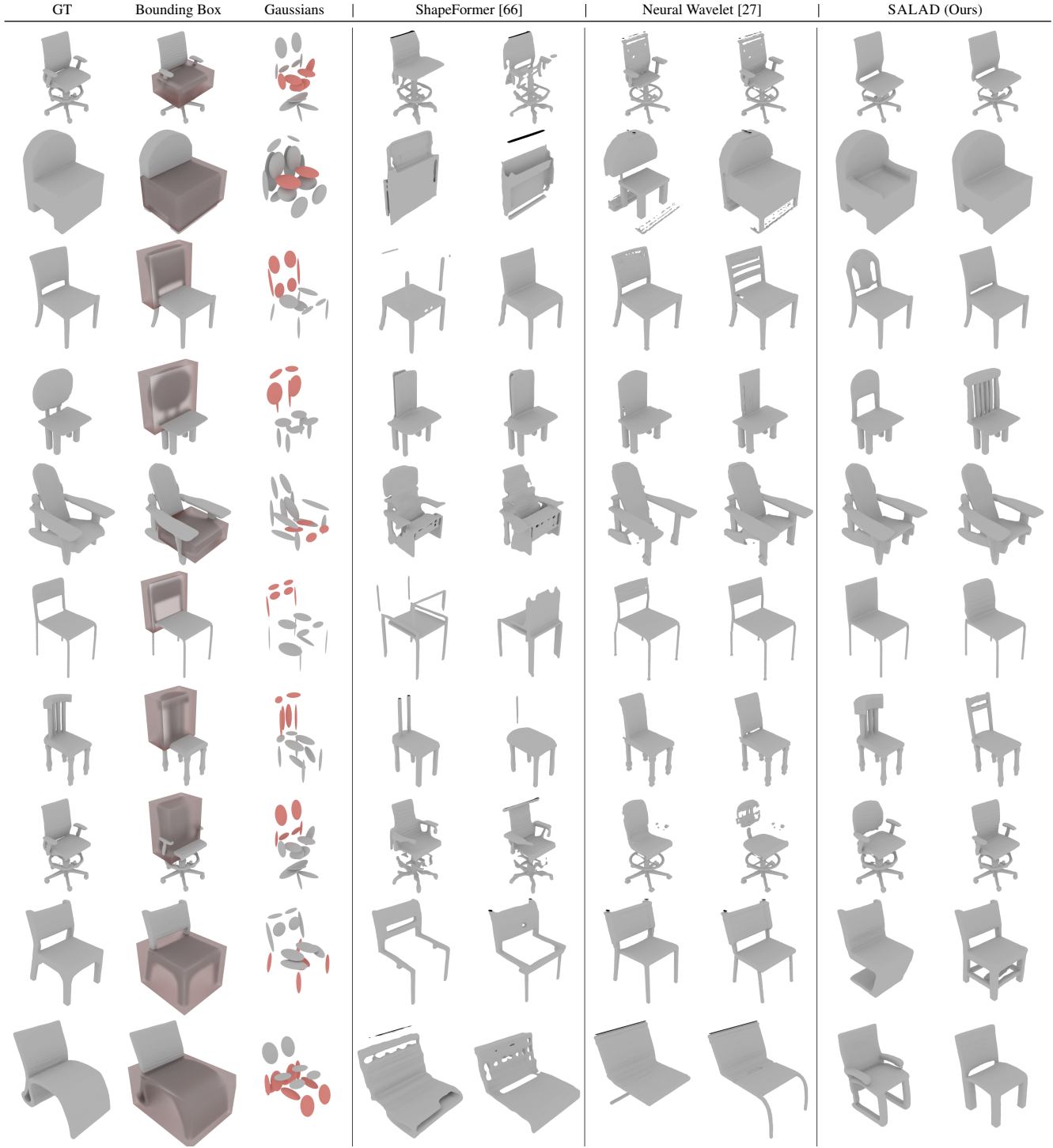
In the following, we provide more qualitative comparisons on shape generation with *chair* and *airplane* classes, as shown in Figure 4.


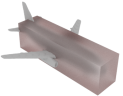





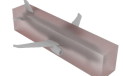

















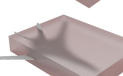

















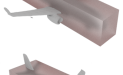

















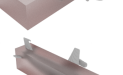
















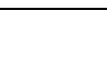
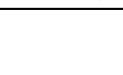

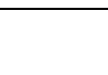
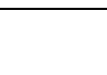
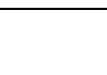
























DPM [37]	PVD [73]	LION [70]	Voxel-GAN [29]	Neural Wavelet[27]	SPAGHETTI [20]	Diff. of \mathbf{z}	Diff. of $\{\mathbf{p}_i\}_{i=1}^N$	Gaussians	SALAD (Ours)
									
									
									
									
									
									
									
									
									

DPM [37]	PVD [73]	LION [70]	Voxel-GAN [29]	Neural Wavelet[27]	SPAGHETTI [20]	Diff. of \mathbf{z}	Diff. of $\{\mathbf{p}_i\}_{i=1}^N$	Gaussians	SALAD (Ours)
									
									
									
									
									
									
									
									
									
									
									
									
									
									
									

A.5. More Qualitative Comparisons on Part Completion

We report more qualitative comparisons on part completion with *chair* and *airplane* classes, as shown in Figure 5.















































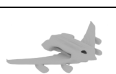
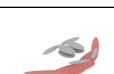

























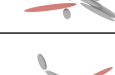




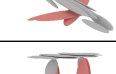








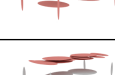






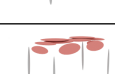

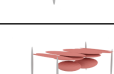




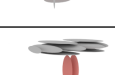




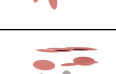














GT	Bounding Box	Gaussians	ShapeFormer [66]	Neural Wavelet [27]	SALAD (Ours)
					
					
					
					
					
					
					
					
					
					
					
					
					
					
					
					
					
					

A.6. More Qualitative Results on Part Mixing and Refinement


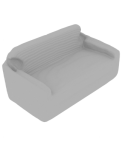









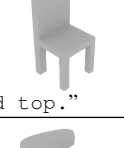






We report more qualitative results on part mixing and refinement with *chair*, *airplane* and *table* classes, as shown in Figure 6.

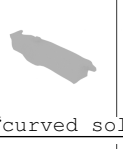




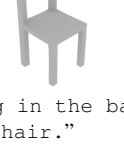












Shape A		Shape B		A→B		A→B Refined
						
						
						
						
						
						
						
						
						
						
						
						
						
						
						
						
						

A.7. More Qualitative Comparisons on Text-Guided Shape Generation









































We report more qualitative comparisons on text-guided shape generation between AutoSDF [43] and SALAD.

AutoSDF [43]	SALAD (Ours)
	
“fat no legs.”	
	
“thin/skinny legs with chair arms.”	
	
“the target has very tiny arms.”	
	
“with a narrow slat across my back.”	
	
“round chair with round back.”	
	
“curved top.”	
	
“oval footrest.”	
	
“wrap around curved back narrow legs.”	
	
“this chair is very tall with skinny legs on it.”	

AutoSDF [43]	SALAD (Ours)
	
“curved solid back.”	
	
“rounded back.”	
	
“has an opening in the back of the chair.”	
	
“the one with the oval shaped back.”	
	
“the one that look most like a lawn chair. net-like back.”	
	
“dining room chair with fancy holes in back.”	
	
“regular looking back, no arms.”	
	
“5 lines, with curve.”	

A.8. More Qualitative Results on Text-Guided Part Completion

We report more qualitative results on text-guided part completion leveraging SALAD and GAUSSGLOT. In the figure below, the parts selected by GAUSSGLOT from the text are highlighted by red. Text-conditioned SALAD completes the selected parts to match the text via the guided reverse process.

Input Mesh	Input Gaussians	Output Mesh	Output Gaussians	Input Mesh	Input Gaussians	Output Mesh	Output Gaussians
							
“four legs and a straight back”				“straight rectangular back”			
							
“chair with no arms”				“swivel legs”			
							
“solid base and no leg”				“round seat has arms and a circle base”			
							
“thick legs and arms”				“circular back”			
							
“four thin legs”				“it only has two legs”			