# DiffGS: Functional Gaussian Splatting Diffusion

**Anonymous Author(s)**
Affiliation
Address
email

## Abstract

3D Gaussian Splatting (3DGS) has shown convincing performance in rendering speed and fidelity, yet the generation of Gaussian Splatting remains a challenge due to its discreteness and unstructural nature. In this work, we propose DiffGS, a general Gaussian generator based on latent diffusion models. DiffGS is a powerful and efficient 3D generative model which is capable of generating Gaussian primitives at arbitrary numbers for high-fidelity rendering with rasterization. The key insight is to disentangledly represent Gaussian Splatting via three novel functions to model Gaussian probabilities, colors and transforms. Through the novel disentangling of 3DGS, we represent the discrete and unstructual 3DGS with continuous Gaussian Splatting functions, where we then train a latent diffusion model with the target of generating these Gaussian Splatting functions both unconditionally and conditionally. Meanwhile, we introduce a discretization algorithm to extract Gaussians at arbitrary numbers from the generated functions via octree-guided sampling and optimization. We explore DiffGS for various tasks, including unconditional generation, conditional generation from text, image, and partial 3DGS, as well as Point-to-Gaussian generation. We believe that DiffGS provides a new direction for flexibly modeling and generating Gaussian Splatting.

## 1 Introduction

3D content creation is a vital task in computer graphics and 3D computer vision, which shows great potential in real-world applications such as virtual reality, game design, film production, and robotics. Previous 3D generative models usually take Neural Radiance Field (NeRF) [30, 2, 47] as the representation. However, the volumetric rendering for NeRF requires considerable computational cost, leading to sluggish rendering speeds and significant memory burden. Recent advances of 3D Gaussian Splatting (3DGS) [21, 52, 19] demonstrating its potential to serve as the next-generation 3D representation by enabling both real-time rendering and high-fidelity appearance modeling. Designing 3D generative models for 3DGS provides a scheme for real-time interaction with 3D creations.

The core challenge in generative 3DGS modeling lies in its discreteness and unstructural nature, which prevents the well-studied frameworks in structural image/voxel/video generation from transferring to directly generate 3DGS. Concurrent works [56, 15] alternatively transport Gaussians into structural voxel grids with volume generation models [9] for generating Gaussians. However, these methods lead to 1) abundant computational cost for high-resolution voxels, 2) struggles in preserving high-quality Gaussian reconstructions due to the information loss during voxelization and 3) limited number of generated Gaussians constrained by the voxel resolutions.

To address these challenges, we present DiffGS, a novel diffusion-based generative model for general 3D Gaussian Splatting which is capable of efficiently generating high-quality Gaussian primitives at arbitrary numbers. The key insight of DiffGS is to disentangledly represent Gaussian Splatting via three novel functions: Gaussian probability function (GauPF), Gaussian color function (GauCF) and Gaussian transform function (GauTF). GauPF indicates the geometry of 3DGS by modeling the

Submitted to 38th Conference on Neural Information Processing Systems (NeurIPS 2024). Do not distribute.

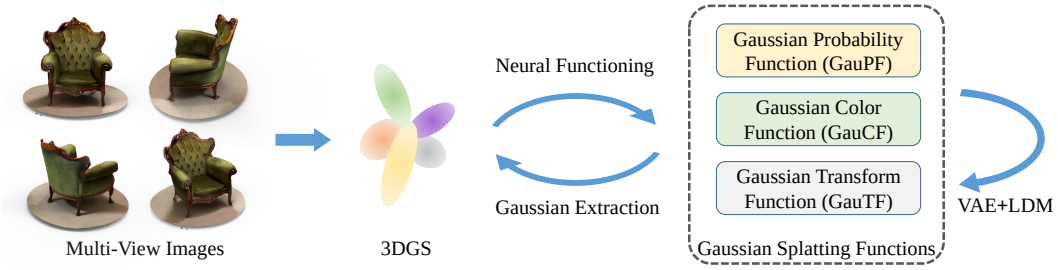



Figure 1: Illustration of DiffGS. We fit 3DGS from multi-view images and then disentangle it into three Gaussian Splatting Functions. We train a Gaussian VAE with a latent diffusion model for generating these functions, followed by a Gaussian extraction algorithm to obtain the final generated Gaussians.

probabilities of each sampled 3D location to be a Gaussian location. GauCF and GauTF predict the Gaussian attributes of appearances and transformations given a 3D location as input. Through the novel disentangling of 3DGS, we represent the discrete and unstructual 3DGS with three continuous Gaussian Splatting Functions.

With the disentangled and powerful representation, the next step is to design a generative model with the target of generating these Gaussian Splatting functions. We propose a Gaussian VAE model for creating compressed representation for the Gaussian Splatting functions. The Gaussian VAE learns a regularized latent space which maps the Gaussians Splatting functions of each shape into one latent vector. A latent diffusion model (LDM) is simultaneously trained at the latent space for generating novel 3DGS shapes. With the powerful LDM, We explore DiffGS to generate diverse 3DGS both conditionally and unconditionally. Finally, we introduce a discretization algorithm to extract Gaussians at arbitrary numbers from the generated functions via octree-guided sampling and optimization. The key idea is to first extract 3D Gaussian geometry from GauPF as the 3D locations sampled at the 3D spaces with largest Gaussian probabilities, and then predict the Gaussian attributes with GauCF and GauTF. We illustrate the overview of DiffGS in Fig. 1.

We systematically summarize the superiority of DiffGS in terms of: 1) Efficiency, we design DiffGS based on Gaussian Splatting and Latent Diffusion Models, which shows significant efficiency in model training, inference and shape rendering. 2),3) Generality and quality, we generate native 3DGS without processes like voxelization, leading to unimpaired quality and generality in applying to downstream 3DGS applications. 4) Scalability, we scalably generate Gaussian primitives at arbitrary numbers. We conduct comprehensive experiments on both synthetic ShapeNet dataset and real-world DeepFashion3D dataset, which demonstrate our non-trivial improvements over the state-of-the-art methods.

In summary, our contributions are:

- We propose DiffGS, a novel diffusion-based generative model for general 3D Gaussian Splatting which is capable of efficiently generating high-quality Gaussian primitives at arbitrary numbers.
- We introduce a novel schema to disentangledly represent Gaussian Splatting via three functions to model Gaussian probabilities, Gaussian colors and Gaussian transforms. We simultaneously propose a discretization algorithm to extract Gaussians from these functions via octree-guided sampling and optimization.
- DiffGS achieves remarkable performances under various tasks including unconditional generation, conditional generation from text, image, and partial 3DGS, as well as Point-to-Gaussian generation.

## 2 Related Work

### 2.1 Rendering-Guided 3D Representation

Remarkable progress have been achieved in the field of novel view synthesis (NVS) [30, 34, 47, 2, 32], since the proposal of Neural Radiance Field (NeRF) [30]. NeRF implicitly represents scene

appearance and geometries using MLP-based neural networks, optimized through volume rendering to achieve outstanding novel view synthesis quality. Some variants [1, 12, 36] show promising performance by advancing NeRF in terms of rendering quality, scalability and view-consistency. Additionally, more recent methods [32, 6, 11, 49] explore the training and rendering efficiency of NeRF by introducing feature-grids based 3D representations. Instant-NGP [32] highly accelerates NeRF learning by introducing multiresolution feature grids based on hash table with fully-fused CUDA kernel implementations. However, the NeRF representations which require expensive neural network inferences during volume rendering, still struggles in the applications where real-time rendering speed is required.

Recently, the emergence of 3D Gaussian Splatting (3DGS) [21, 44, 23, 52, 55] has showcased impressive real-time results in novel view synthesis (NVS). 3DGS [21] has led to revolutions in the NVS field by demonstrating superior performances in multiple domains. However, the generation of Gaussian Splatting remains a challenge due to its discreteness and unstructural nature. We introduce a novel schema to represent the discrete and unstructual 3DGS with three continuous Gaussian Splatting Functions, thus ingeniously tackle the challenge by designing generative models for the functions.

### 2.2 3D Generative Models

The field of creating 3D contents with generative models has emerged as a particularly captivating research direction. A series of works [35, 24, 50, 39, 29, 27, 7, 43, 53, 45] focus on optimization-based frameworks based on Score Distillation Sampling (SDS), which achieve convincing generation performances by distilling 3D geometry and appearance of the radiance fields with pretrained 2D diffusion models [33, 17] as the prior. However, these works entail significant computational costs due to time-consuming per-scene optimization. Going beyond optimization-based 3D generation, recent works [31, 46, 48, 20] explore 3D generative methods based on diffusion models to directly learn priors from 3D datasets for generative radiance fields modeling. These methods typically represent radiance fields as structural triplanes [48, 42, 14] or voxels [46, 31, 9]. DiffRF [31] leverage a voxel based NeRF representation with 3D U-Nets as the backbone to train a diffusion model.

With the recent advances in 3DGS [21], designing a powerful 3D generative model for generating 3DGS is expected to be a popular research topic and also brings significant challenges due to the discreteness and unstructural nature of 3DGS, which prevents the well-studied frameworks in structural image/voxel/video generation from transferring to directly generate 3DGS. A series of works [54, 59, 18, 57] follow the schema of image-based reconstruction without generative modeling, which lake the ability to generate diverse shapes. Concurrent works [15, 56] follow the voxel-based representations to transport Gaussians into structural voxel grids with volume generation models for generating Gaussians. However, these methods come with several drawbacks, including abundant computational costs for high-resolution voxels, struggles in preserving high-quality Gaussian reconstructions due to information loss during voxelization, and a restricted number of generated Gaussians constrained by voxel resolutions. In contrast, our proposed DiffGS explore a new perspective to directly represent the discrete and unstructual 3DGS with three continuous Gaussian Splatting Functions. Though the insight, we design a latent diffusion model for efficiently generating high-quality Gaussian primitives by learning to generate the Gaussian Splatting Functions. DiffGS generates general Gaussians at arbitrary numbers with a specially designed octree-based extraction algorithm.

## 3 Method

We introduce DiffGS, a novel diffusion-based generative model for general 3D Gaussian Splatting which is capable of efficiently generating high-quality Gaussian primitives at arbitrary numbers. The overview of DiffGS is shown in Fig. 2. We first preview Gaussian Splatting in Sec. 3.1 and present the novel functional schema for representing Gaussian Splatting with three disentangled Gaussian Splatting Functions in Sec. 3.2. We then introduce the Gaussian Variational Auto-encoder and the Latent Diffusion model for compressing and generative modeling on Gaussian Splatting Functions, as shown in Sec. 3.3. A novel discretization algorithm is proposed in Sec. 3.4 to extract Gaussians at arbitrary numbers from the generated functions via octree-guided sampling and optimization.

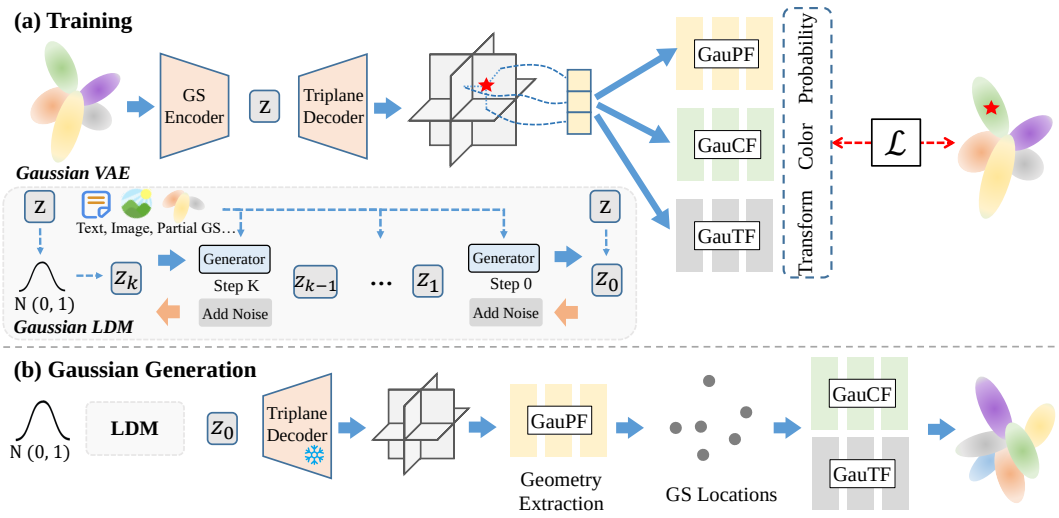



Figure 2: Overview of DiffGS. (a) We disentangle the fitted 3DGS into three Gaussian Splatting Functions to model the Gaussian probability, colors and transforms. We then train a Gaussian VAE with a conditional latent diffusion model for generating these functions. (b) During generation, we first extract Gaussian geometry from the generated GauPF, followed by the GauCF and GauTF to obtain the Gaussian attributes.

### 3.1 Preview Gaussian Splatting

3D Gaussian Splatting (3DGS) [21] represents a 3D shape or scene as a set of Gaussians with attributes to model the geometries and view-dependent appearances. For a 3DGS $G = \{g_i\}_{j=1}^{N}$ containing $N$ Gaussians, the geometry of j-$th$ Gaussian is explicitly parameterized via 3D covariance matrix $\Sigma_i$ and its center $\sigma_i \in \mathbb{R}^3$, fomulated as:

$$g_i(x) = \exp\left(-\frac{1}{2}(x-\sigma_i)^T\Sigma^{-1}(x-\sigma_i)\right), \tag{1}$$

where the covariance matrix $\Sigma_i = r_i s_i s_i^T r_i^T$ is factorized into a rotation matrix $r_i \in \mathbb{R}^4$ and a scale matrix $s_i \in \mathbb{R}$. The appearance of the Gaussian $g_i$ is controlled by an opacity value $o_j \in \mathbb{R}$ and a color value $c_i \in \mathbb{R}^3$. Note that the color is represented as a series of sphere harmonics coefficients in practice of 3DGS, yet we still keep its definition as three-dimension color $c_i$ in our paper for a clear understanding on our method. To this end, the 3DGS $G$ is defined as $\{g_i = \{\sigma_i, r_i, s_i, o_i, c_i\} \in \mathbb{R}^K\}_{j=1}^{N}$, where $K$ is dimension of the combined attributes in each Gaussian.

### 3.2 Functional Gaussian Splatting Representation

The key challenge in generative 3DGS modeling lies in its discreteness and unstructural nature, which prevents the well-studied generative frameworks from transferring to directly generate 3DGS. We address this challenge by introducing to disentangledly represent Gaussian Splatting via three novel functions: Gaussian probability function (GauPF), Gaussian color function (GauCF) and Gaussian transform function (GauTF). Through the novel disentangling of 3DGS, we represent the discrete and unstructual 3DGS with three continuous Gaussian Splatting Functions.

**Gaussian Probability Function.** Gaussian Probability Function (GauPF) indicates the geometry of 3DGS by modeling the probabilities of each sampled 3D location to be a Gaussian location. Given a set of 3D query location $Q = \{q_j \in \mathbb{R}^3\}_{i=1}^{M}$ sampled in 3D space around a fitted 3DGS $G = \{g_i \in \mathbb{R}^3\}_{j=1}^{N}$, the GauPF of $G$ predicts the probabilities $p$ of queries $\{q_j\}_{i=1}^{M}$ to be a Gaussian location in $G$, fomulated as:

$$p_j = \text{GauPF}(q_j) \in [0, 1]. \tag{2}$$

The idea of Gaussian probability modeling comes from the observation that the further a 3D location $q_j$ is from all Gaussians, the lower the probability that any Gaussian occupies the space at $q_j$.

Therefore the ground truth Gaussian probability of $q_j$ is defined as:

$$\text{GauPF}(q_j) = \tau(\lambda(\underset{i\in[1,N}{min}||\hat{q}_j - \sigma_i||_2)), \tag{3}$$

where $\underset{i\in[1,N}{min}||\hat{q}_j - \sigma_i||_2$ indicates the distance from $q_j$ to the nearest Gaussian center in $\{\sigma_i\}_{i=1}^N$, $\lambda$ is a truncation function which filters the extremely large values and $\tau$ is a continuous function which maps the query-to-Gaussian distances to probabilities in the range of [0,1].

A learned GauPF implicitly models the locations of 3D Gaussian centers, which is the key factor for generating high-quality 3DGS. The extraction of 3DGS centers from GauPF is then achieved with our designed Gaussian extraction algorithm which will be introduced in Sec. 3.4.

**Gaussian Color and Transform Modeling.** Gaussian Color Function (GauCF) and Gaussian Transform Function (GauTF) predict the Gaussian attributes of appearances and transformations from Gaussian geometries. Specifically, given the center $\sigma_i$ of a Gaussian $g_i$ in $G$ as input, GauCF predicts the color attribute $c_i$ and GauTF predicts the rotation $r_i$, scale $s_i$ and opacity $o_i$, formulated as:

$$\{c_i\} = \text{GauCF}(\sigma_i); \;\; \{r_i, s_i, o_i\} = \text{GauTF}(\sigma_i). \tag{4}$$

Note that GauCF and GauTF mainly focus on predicting the Gaussian colors and transforms from 3D Gaussian centers. This is different from the GauPF which models the probabilities of query samples in the 3D space. The reason is that GauPF focuses on exploring the geometry of 3DGS from the 3D space, while GauCF and GauTF learn to predict the Gaussian attributes from the known geometries.

Through the novel disentangling of 3DGS, we represent the discrete and unstructual 3DGS with three continuous Gaussian Splatting Functions. The functional representation is a general and flexible term for 3DGS which has no restrictions on the Gaussian numbers, densities, geometries, etc.

### 3.3 Gaussian Variational Auto-encoder and Latent Diffusion

With the disentangled and powerful representation, the next step is to design a generative model with the target of generating these Gaussian Splatting functions. We follow the common schema to design a Gaussian Variational Auto-encoder (VAE) [22] with a Latent Diffusion Model (LDM) [33] as the generative model. The detailed framework and the training pipeline of DiffGS are illustrated in Fig. 1(a).

**Gaussian VAE.** The Gaussian VAE compresses the Gaussian Splatting Functions into a regularized latent space by mapping the Gaussian Splatting Functions of each 3DGS shape into a latent vector, from which we can also recover the Gaussian Splatting Functions. Specifically, the Gaussian VAE consists of 1) a GS encoder $\phi_{en}$ to learn representations from 3DGS and encodes each 3DGS into a latent vector $z$, 2) a triplane decoder $\phi_{de}$ which decodes the latent $z$ into a feature triplane, and 3) three neural predictors $\psi_{pf}, \psi_{cf}$ and $\psi_{tf}$ which serve as the implementation of GauPF, GauCF and GauTF to predict Gaussian probabilities, colors and transforms.

Given a fitted 3DGS $G = \{g_i\}_{j=1}^N$ as input, the GS encoder $\phi_{en}$ extracts a global latent feature $z$ from $G$, which is then decoded into a feature triplane $t \in \mathbb{R}^{H\times W\times C\times 3}$ with the decoder $\phi_{de}$, formulated as:

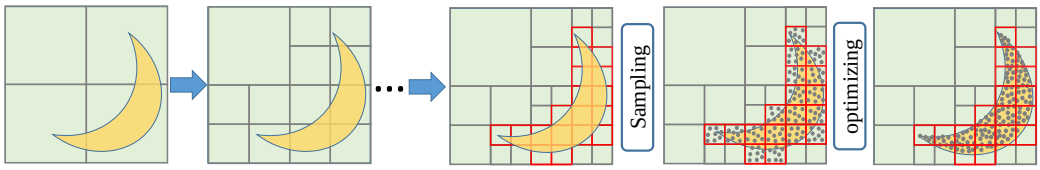
$$z = \phi_{en}(G); \;\; t = \phi_{de}(z). \tag{5}$$

The triplane $t$ consists of three orthogonal feature planes $\{t_{XY}, t_{XZ}, t_{YZ}\}$ which are aligned to the axex. For a 3D location $q_j$, we obtain its corresponding feature $f_j = interp(t, q_j)$ from the triplane $t$ by projecting $q_j$ onto the orthogonal feature planes and concatenating the tri-linear interpolated features at the three planes. We then predict the Gaussian probability, color and transform of $q_j$ with the neural Gaussian Function predictors as:

$$\{\hat{p}_i\} = \psi_{pf}(f_j); \;\; \{\hat{c}_i\} = \psi_{cf}(f_j); \;\; \{\hat{r}_i, \hat{s}_i, \hat{o}_i\} = \psi_{tf}(f_j). \tag{6}$$

The Gaussian VAE is trained with the target of accurately predicting Gaussian attributes and robustly regularizing the latent space. In practice, the training objective is formulated as:

$$\mathcal{L}_{\text{VAE}} = \|\{\hat{p}, \hat{c}, \hat{r}, \hat{s}, \hat{o}\} - \{p, c, r, s, o\}\|_1 + \beta\left(D_{KL}\left(\mathcal{Q}_\phi(z|G)\|\mathcal{P}(z)\right)\right). \tag{7}$$




(a) Octree depth 1 (b) Octree depth 2 (c) Octree depth L (d) Proxy Gaussian centers (e) Final Gaussian centers

Figure 3: Gaussian geometry extractions from generated GauPF. The yellow and green regions indicate the high probability area and the low probability area judged by GauPF. (a),(b) and (c) show the progressively octree build process at depth 1,2 and $L$. (d) We sample proxy Gaussian centers from the octree at final depth $L$. (e) We optimize proxy centers to the exact geometry indicated in GauPF.

The first loss term indicates the $\mathcal{L}_1$ loss between the predicted Gaussian attributes in Eq. (6) and the target ones defined by the ground truth Gaussian Splatting Functions in Eq. (2) and Eq. (4). The second term in Eq. (7) is the KL-divergence loss with a factor of $\beta$, which constrains on the regularization of the learned latent space of $z$. Specifically, we define the inferred posterior of $z$ as the distribution $\mathcal{Q}_\phi(z|G)$, which is regularized to align with the Gaussian distribution prior $\mathcal{P}(z) = \mathcal{N}(0, I)$, where $I$ is the standard deviation.

**Gaussian LDM.** With the trained Gaussian VAE in place, we are now able to encode any 3DGS into a compact 1D latent vector $z$. We then train a latent diffusion model (LDM) [33] efficiently on the latent space. A diffusion model is trained to generate samples from a target distribution by reversing a process that incrementally introduces noise. We define $\{z_0, z_1, ..., z_K\}$ as the forward process $\gamma(z_{0:K})$ which gradually transforms a real data $z_0$ into Gaussian noise $(z_T)$ by adding noises. The backward process $\mu(z_{0:K})$ leverages a neural generator $\mu$ to denoise $z_K$ into a real data sample.

To achieve controlled generation of 3DGS, we introduce a conditioning mechanism [41] into the diffusion process with cross-attention. Given an input condition $y$ (e.g. text, image, partial 3DGS), we leverage a custom encoder $\delta$ to project $y$ into the condition embedding $\delta(y)$. The embedding is then fused into the generator $\mu$ with cross attention modules. Following DDPM, we simply adopt the optimizing objective to train the generator for predicting noises $\epsilon_\sigma$, formulated as:

$$\mathcal{L}_{\text{LDM}} = \mathbb{E}_{z_0, t, \epsilon \sim \mathcal{N}(0,I)} \left[ \|\epsilon - \epsilon_\sigma \left( z_t, \delta(y), t \right)\|^2 \right], \tag{8}$$

where $t$ is a time step and $\epsilon$ is a noise latent sampled from the Gaussian distribution $\mathcal{N}(0, I)$. We adopt the well-studied architecture DALLE-2 [40] as the LDM implementation.

### 3.4 Gaussian Extraction Algorithm

The final step for the generation process of DiffGS is to extract 3DGS from the generated Gaussian Splatting Functions, similar to the effect of Marching Cubes algorithm [25] which extracts meshes from Signed Distance Functions. The key factor is to extract the geometries of 3DGS, i.e., Gaussian locations and the appearances of 3DGS, i.e., colors and transforms. The full generation pipeline is shown in Fig. 2(b).

**Octree-Guided Geometry Sampling.** The locations of 3D Gaussian centers indicate the geometry of the represented 3DGS. We aim to design a discretization algorithm to obtain the discrete 3D locations from the learned continuous Gaussian Probability Function parameterized with the neural network $\psi_{pf}$, which models the probability of each query sampled in the 3D space to be a 3D Gaussian location. To achieve this, we design an octree based sampling and optimization algorithm which generates accurate center locations of 3D Gaussians at arbitrary numbers.

We show the 2D illustration of the algorithm in Fig. 3. Assume that the 3D space is divided into the high probability area (the yellow region) and the low probability area (the green region) by the generated GauPF. We aim to extract the geometry as the locations with high probabilities. A naive implementation is to densely sample queries in the 3D space and keep the ones with large probabilities as outputs. However, it will lead to high computational cost for inferencing and the discrete sampling also struggles to accurately reach the locations with largest probabilities in the continuous GauPF. We get inspiration from octree [28, 51] to design a progressive strategy which only explores the 3D regions with large probabilities in current octree depth for further subdivision in the next octree depth. After $L$ layers of octree subdivision, we reach the local regions with largest

Table 1: Comparisons of unconditional generation under ShapeNet [5] dataset.

| Method | Airplane | | Chair | |
|---|---|---|---|---|
| | FID-50K ↓ | KID-50K (‰) ↓ | FID-50K ↓ | KID-50K (‰) ↓ |
| GET3D [13] | — | — | 59.51 | 2.414 |
| DiffTF [4] | 110.8 | 9.173 | 93.02 | 6.708 |
| Ours | **47.03** | **3.436** | **35.28** | **2.148** |


Figure 4: Visual comparisons with state-of-the-arts on unconditional generation of ShapeNet Chairs.

probabilities, from where we uniformly sample $N$ 3D points as the proxy points $\{\rho_i\}_{i=1}^N$ representing coarse locations of Gaussian centers.

**Optimizing Geometry with GauPF.** To further refine the proxy points to the exact locations of Gaussian centers with largest probabilities in GauPF, we propose to further optimize the proxy points with the supervision from learned GauPF $\psi_{pf}$. Specifically, we set the position of proxy points $\{\rho_i = \{\rho x_i, \rho y_i, \rho z_i\}\}_{i=1}^N$ to be learnable and optimize them to reach the positions $\{\hat{\sigma}_i\}_{i=1}^N$ with largest probabilities of $\psi_{pf}$. The optimization target is formulated as:

$$\mathcal{L}_{\text{Geo}} = \frac{1}{N}\sum_{i=1}^{N} \psi_{pf}(\rho_i). \tag{9}$$

Note that we can set $N$ to arbitrary numbers, enabling DiffGS to generate 3DGS with no limits on the density and resolution.

**Extracting Gaussian Attributes.** We now obtain the estimated geometry indicating the predicted Gaussian centers $\{\hat{\sigma}_i\}_{i=1}^N$. We then extract the appearances and transforms from the generated triplane $t$, Gaussian Color Function $\psi_{cf}$ and Gaussian Transform Function $\psi_{tf}$ as $\{\hat{c}_i\} = \psi_{cf}(interp(t, \hat{\sigma}_i))$ and $\{\hat{r}_i, \hat{s}_i, \hat{o}_i\} = \psi_{tf}(interp(t, \hat{\sigma}_i))$. Finally, the general 3DGS is now generated as $\hat{G} = \{\hat{\sigma}_i, \hat{c}_i, \hat{r}_i, \hat{s}_i, \hat{o}_i\}_{i=1}^N$.

# 4 Experiment

## 4.1 Unconditional Generation

**Dataset and Metrics.** For unconditional generation of 3D Gaussian Splatting, we conduct experiments under the airplane and chair classes of ShapeNet [5] dataset. Following previous works [31, 4], we report two widely-used image generation metrics Fréchet Inception Distance (FID) [16] and Kernel Inception Distance (KID) [3] for evaluating the rendering quality of our proposed DiffGS and previous state-of-the-art works. The metrics are evaluated between 50K renderings of the generated shapes and 50K renderings of the ground turth ones, both at the resolution of 1024×1024.

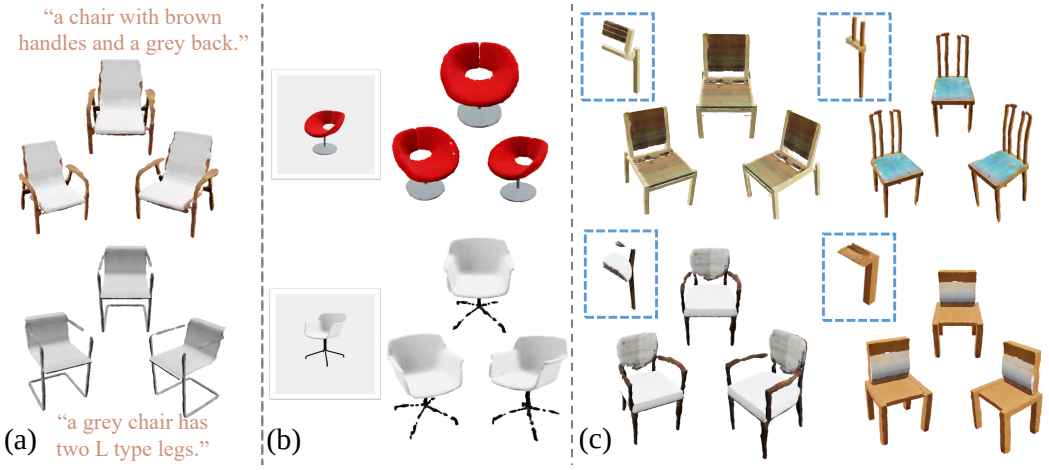



Figure 5: Visualization of conditional 3DGS generation results on ShapeNet. (a) Text conditional generation. (b) Image conditional generation. (c) Gaussian Splatting completion.

**Comparisons** We compare DiffGS with the state-of-the-art methods in terms of the rendering quality of generated shapes, including the GAN-based methods GET3D [13] and the diffusion based method DiffTF [4]. The quantitative comparison is shown in Tab. 1, where DiffGS achieves the best performance over all the baselines. We further show the visual comparison on the renderings of some generated shapes in Fig. 4, where the GAN-based GET3D struggle in generating complex shapes and the generations of DiffTF is blurry with poor textures. In contrast, DiffGS produces significantly more visual-appealing and high-fidelity generations in terms of rendering and geometry qualities.

### 4.2 Conditional Generation

We explore the conditional generation ability of DiffGS given texts, images and partial 3DGS as the input conditions. All the experiments are conducted under the chair class of ShapeNet [5] dataset with commonly used data splits in previous methods [8, 26].

**Text/Image-conditional Gaussian Splatting Generation** For introducing texts/images as the conditions for controllable Gaussian Splatting generation, we leverage the frozen text and image encoder from the pretrained CLIP [38] model as the implementation of custom text encoder $\gamma_{text}$ and $\gamma_{image}$ for achieving text/image embeddings. We then train DiffGS with the conditional optimization objective in Eq.(8). We show the visualization of some text/image conditional generations produced by DiffGS in Fig. 5(a) and Fig. 5(b). The results show that DiffGS accurately recovers the semantics and geometries described in the text prompts and the images, demonstrating the powerful capability of DiffGS in generating high-fidelity 3DGS from text descriptions or vision signals.

**Gaussian Splatting Completion** Additionally, we explore an interesting task of Gaussian Splatting completion. To the best of our knowledge, we are the first to focus and introduce solutions for this task. Specifically, the Gaussian Splatting completion task is to recover the complete 3DGS from a partial 3DGS which contains large occlusions. In real-world applications, having only sparse views with limited viewpoint movement available for optimizing 3DGS often results in a partial 3DGS. Solving Gaussian Splatting completion task enables us to infer the complete and dense 3DGS from the partial ones for improving the rendering quality at invisible viewpoints.

We introduce DiffGS with partial 3DGS as the conditions for solving this task. Specifically, we simply leverage a modified PointNet [37] as the custom encoder $\gamma_{partial}$ for partial 3DGS. Fig. 5(c) presents the visualization of Gaussian Splatting completion results produced by DiffGS. The results show that DiffGS is capable of recovering complex geometries and detailed appearances from highly occluded 3DGS. Please refer to the appendix for implementation details on Gaussian Splatting completion.

### 4.3 Point-to-Gaussian Generation

We further introduce DiffGS for another challenging and vital task of Point-to-Gaussian generation. This task aims to generate the Gaussian attributes given a 3D point cloud as input. The task serves as

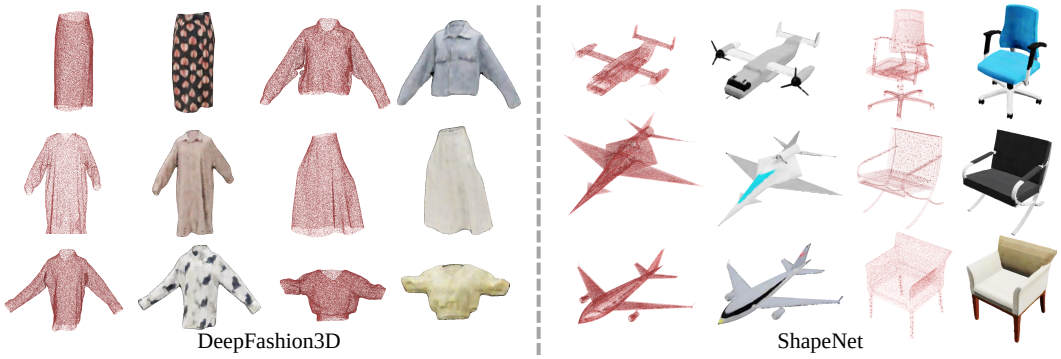



Figure 6: Visualization of Point-to-Gaussian fittings on Deepfashion3D and generations on ShapeNet.

the bridge between the easily accessible point clouds and the powerful 3DGS representation which efficiently models high-quality 3D appearances.

**Dataset and Implementation.** We conduct experiments under the chair and airplane classes of ShapeNet and also the widely-used garment dataset DeepFashion3D [58]. The DeepFashin3D dataset is a real-captured 3D dataset containing complex textures. For implementing Point-to-Gaussian, we simply train the Gaussain VAE with the three-dimension point clouds as inputs, instead of the 3DGS with attributes. Please refer to the appendix for more details on data preparation and implementation.

**Performances.** We provide the visualization of some Point-to-Gaussian fitting and generation results in Fig. 6. We shown the fitting results for Deepfashion3D [58] dataset and the generation results for the test set of airplane and chair classes in ShapeNet [5]. DiffGS produces visual-appealing 3DGS generations given only 3D point cloud geometries as inputs. The results demonstrate that DiffGS can accurately predict Gaussian attributes for 3D point clouds. We believe DiffGS provides a new direction for 3DGS content generation by connecting 3DGS with point clouds.

### 4.4 Ablation Study

To evaluate some major designs and important hyper-parameters in DiffGS, we conduct ablation studies under the chair class of ShapeNet dataset. We report the performance in terms of PSNR, SSIM and LPIPS of the reconstructed 3DGS with Gaussian VAE.

**Framework Designs.** We first evaluate some major designs of our framework in Tab. 2. We justify the effectiveness of introducing the truncation function $\lambda$ when modeling GauPF and report the results without $\lambda$ as 'w/o truncation'. We then explore implementing the projection function $\tau$ either as $\tau(x) = e^{-x}$ (as shown in 'Exponent') or as a linear projection (as shown in 'Ours'). We also show the results without the optimization process during Gaussian extraction as 'w/o Optimization', which demonstrates the effectiveness of optimizing Gaussians to the exact locations.

Table 2: Ablations on framework designs.

| Method | PSNR | SSIM | LPIPS |
|---|---|---|---|
| w/o Trunction | 29.39 | 0.9792 | 0.0173 |
| Exponent | 29.74 | 0.9765 | 0.0188 |
| w/o Optimization | 30.34 | 0.9875 | 0.0152 |
| Ours | **34.01** | **0.9879** | **0.0149** |

**Gaussian Numbers.** One significant advantage of DiffGS lies in the ability of generating high-quality Gaussians at arbitrary numbers. To explore how the number of Gaussians affects the rendering quality, we conduct ablations on the Gaussian numbers as shown in Fig. 3. The results demonstrate that denser Gaussians lead to better quality.

Table 3: Ablations on Gaussian number.

| Num | PSNR | SSIM | LPIPS |
|---|---|---|---|
| 50K | 28.61 | 0.9787 | 0.0251 |
| 100K | 30.35 | 0.9838 | 0.0151 |
| 350K | **34.01** | **0.9879** | **0.0149** |

## 5 Conclusion

In this paper, we introduce DiffGS for generative modeling of 3DGS. DiffGS disentangled represent 3DGS via three novel functions to model Gaussian probabilities, colors and transforms. We then train a latent diffusion model with the target of generating these functions both conditionally and unconditionally. DiffGS generates 3DGS with arbitrary numbers by an octree-guided extraction algorithm. The experimental results on various tasks demonstrate the superiority of DiffGS.

# References


[1] Jonathan T Barron, Ben Mildenhall, Matthew Tancik, Peter Hedman, Ricardo Martin-Brualla, and Pratul P Srinivasan. Mip-nerf: A multiscale representation for anti-aliasing neural radiance fields. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 5855–5864, 2021.

[2] Jonathan T Barron, Ben Mildenhall, Dor Verbin, Pratul P Srinivasan, and Peter Hedman. Mip-nerf 360: Unbounded anti-aliased neural radiance fields. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5470–5479, 2022.

[3] Mikołaj Bińkowski, Danica J Sutherland, Michael Arbel, and Arthur Gretton. Demystifying mmd gans. *arXiv preprint arXiv:1801.01401*, 2018.

[4] Ziang Cao, Fangzhou Hong, Tong Wu, Liang Pan, and Ziwei Liu. Large-vocabulary 3d diffusion model with transformer. *arXiv preprint arXiv:2309.07920*, 2023.

[5] 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.

[6] Anpei Chen, Zexiang Xu, Andreas Geiger, Jingyi Yu, and Hao Su. Tensorf: Tensorial radiance fields. In *European Conference on Computer Vision*, pages 333–350. Springer, 2022.

[7] Zilong Chen, Feng Wang, and Huaping Liu. Text-to-3d using gaussian splatting. *arXiv preprint arXiv:2309.16585*, 2023.

[8] Zhiqin Chen and Hao Zhang. Learning implicit fields for generative shape modeling. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5939–5948, 2019.

[9] Yen-Chi Cheng, Hsin-Ying Lee, Sergey Tulyakov, Alexander G Schwing, and Liang-Yan Gui. Sdfusion: Multimodal 3d shape completion, reconstruction, and generation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4456–4465, 2023.

[10] Gene Chou, Yuval Bahat, and Felix Heide. Diffusion-sdf: Conditional generative modeling of signed distance functions. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 2262–2272, 2023.

[11] Sara Fridovich-Keil, Alex Yu, Matthew Tancik, Qinhong Chen, Benjamin Recht, and Angjoo Kanazawa. Plenoxels: Radiance fields without neural networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5501–5510, 2022.

[12] Qiancheng Fu, Qingshan Xu, Yew-Soon Ong, and Wenbing Tao. Geo-Neus: Geometry-consistent neural implicit surfaces learning for multi-view reconstruction. *Advances in Neural Information Processing Systems (NeurIPS)*, 2022.

[13] 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. *Advances In Neural Information Processing Systems*, 35:31841–31854, 2022.

[14] Anchit Gupta, Wenhan Xiong, Yixin Nie, Ian Jones, and Barlas Oğuz. 3dgen: Triplane latent diffusion for textured mesh generation. *arXiv preprint arXiv:2303.05371*, 2023.

[15] Xianglong He, Junyi Chen, Sida Peng, Di Huang, Yangguang Li, Xiaoshui Huang, Chun Yuan, Wanli Ouyang, and Tong He. Gvgen: Text-to-3d generation with volumetric representation. *arXiv preprint arXiv:2403.12957*, 2024.

[16] Martin Heusel, Hubert Ramsauer, Thomas Unterthiner, Bernhard Nessler, and Sepp Hochreiter. Gans trained by a two time-scale update rule converge to a local nash equilibrium. *Advances in neural information processing systems*, 30, 2017.

[17] Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. *Advances in neural information processing systems*, 33:6840–6851, 2020.

[18] Yicong Hong, Kai Zhang, Jiuxiang Gu, Sai Bi, Yang Zhou, Difan Liu, Feng Liu, Kalyan Sunkavalli, Trung Bui, and Hao Tan. Lrm: Large reconstruction model for single image to 3d. *arXiv preprint arXiv:2311.04400*, 2023.

[19] Binbin Huang, Zehao Yu, Anpei Chen, Andreas Geiger, and Shenghua Gao. 2d gaussian splatting for geometrically accurate radiance fields. *arXiv preprint arXiv:2403.17888*, 2024.

[20] Heewoo Jun and Alex Nichol. Shap-e: Generating conditional 3d implicit functions. *arXiv preprint arXiv:2305.02463*, 2023.

[21] Bernhard Kerbl, Georgios Kopanas, Thomas Leimkühler, and George Drettakis. 3d gaussian splatting for real-time radiance field rendering. *ACM Transactions on Graphics*, 42(4):1–14, 2023.

[22] Diederik P Kingma and Max Welling. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013.

[23] Jiahe Li, Jiawei Zhang, Xiao Bai, Jin Zheng, Xin Ning, Jun Zhou, and Lin Gu. Dngaussian: Optimizing sparse-view 3d gaussian radiance fields with global-local depth normalization. *arXiv preprint arXiv:2403.06912*, 2024.

[24] Chen-Hsuan Lin, Jun Gao, Luming Tang, Towaki Takikawa, Xiaohui Zeng, Xun Huang, Karsten Kreis, Sanja Fidler, Ming-Yu Liu, and Tsung-Yi Lin. Magic3d: High-resolution text-to-3d content creation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 300–309, 2023.

[25] William E Lorensen and Harvey E Cline. Marching cubes: A high resolution 3D surface construction algorithm. *ACM Siggraph Computer Graphics*, 21(4):163–169, 1987.

[26] Shitong Luo and Wei Hu. Diffusion probabilistic models for 3d point cloud generation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 2837–2845, 2021.

[27] Baorui Ma, Haoge Deng, Junsheng Zhou, Yu-Shen Liu, Tiejun Huang, and Xinlong Wang. Geodream: Disentangling 2d and geometric priors for high-fidelity and consistent 3d generation. *arXiv preprint arXiv:2311.17971*, 2023.

[28] Donald Meagher. Geometric modeling using octree encoding. *Computer graphics and image processing*, 19(2):129–147, 1982.

[29] Gal Metzer, Elad Richardson, Or Patashnik, Raja Giryes, and Daniel Cohen-Or. Latent-nerf for shape-guided generation of 3d shapes and textures. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 12663–12673, 2023.

[30] B Mildenhall, PP Srinivasan, M Tancik, JT Barron, R Ramamoorthi, and R Ng. NeRF: Representing scenes as neural radiance fields for view synthesis. In *European Conference on Computer Vision*, 2020.

[31] Norman Müller, Yawar Siddiqui, Lorenzo Porzi, Samuel Rota Bulo, Peter Kontschieder, and Matthias Nießner. Diffrf: Rendering-guided 3d radiance field diffusion. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4328–4338, 2023.

[32] Thomas Müller, Alex Evans, Christoph Schied, and Alexander Keller. Instant neural graphics primitives with a multiresolution hash encoding. *ACM Transactions on Graphics (ToG)*, 41(4):1–15, 2022.

[33] Alexander Quinn Nichol and Prafulla Dhariwal. Improved denoising diffusion probabilistic models. In *International Conference on Machine Learning*, pages 8162–8171. PMLR, 2021.

[34] Keunhong Park, Utkarsh Sinha, Jonathan T Barron, Sofien Bouaziz, Dan B Goldman, Steven M Seitz, and Ricardo Martin-Brualla. Nerfies: Deformable neural radiance fields. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 5865–5874, 2021.

[35] Ben Poole, Ajay Jain, Jonathan T Barron, and Ben Mildenhall. Dreamfusion: Text-to-3d using 2d diffusion. *arXiv preprint arXiv:2209.14988*, 2022.

[36] Albert Pumarola, Enric Corona, Gerard Pons-Moll, and Francesc Moreno-Noguer. D-nerf: Neural radiance fields for dynamic scenes. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10318–10327, 2021.

[37] Charles R Qi, Hao Su, Kaichun Mo, and Leonidas J Guibas. Pointnet: Deep learning on point sets for 3d classification and segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 652–660, 2017.

[38] 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 *International conference on machine learning*, pages 8748–8763. PMLR, 2021.

[39] Amit Raj, Srinivas Kaza, Ben Poole, Michael Niemeyer, Nataniel Ruiz, Ben Mildenhall, Shiran Zada, Kfir Aberman, Michael Rubinstein, Jonathan Barron, et al. Dreambooth3d: Subject-driven text-to-3d generation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 2349–2359, 2023.

[40] Aditya Ramesh, Prafulla Dhariwal, Alex Nichol, Casey Chu, and Mark Chen. Hierarchical text-conditional image generation with clip latents, 2022.

[41] Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-resolution image synthesis with latent diffusion models. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 10684–10695, 2022.

[42] J Ryan Shue, Eric Ryan Chan, Ryan Po, Zachary Ankner, Jiajun Wu, and Gordon Wetzstein. 3d neural field generation using triplane diffusion. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 20875–20886, 2023.

[43] Jingxiang Sun, Bo Zhang, Ruizhi Shao, Lizhen Wang, Wen Liu, Zhenda Xie, and Yebin Liu. Dreamcraft3d: Hierarchical 3d generation with bootstrapped diffusion prior. *arXiv preprint arXiv:2310.16818*, 2023.

[44] Jiaxiang Tang, Jiawei Ren, Hang Zhou, Ziwei Liu, and Gang Zeng. Dreamgaussian: Generative gaussian splatting for efficient 3d content creation. *arXiv preprint arXiv:2309.16653*, 2023.

[45] Junshu Tang, Tengfei Wang, Bo Zhang, Ting Zhang, Ran Yi, Lizhuang Ma, and Dong Chen. Make-it-3d: High-fidelity 3d creation from a single image with diffusion prior. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 22819–22829, 2023.

[46] Zhicong Tang, Shuyang Gu, Chunyu Wang, Ting Zhang, Jianmin Bao, Dong Chen, and Baining Guo. Volumediffusion: Flexible text-to-3d generation with efficient volumetric encoder. *arXiv preprint arXiv:2312.11459*, 2023.

[47] Peng Wang, Lingjie Liu, Yuan Liu, Christian Theobalt, Taku Komura, and Wenping Wang. NeuS: Learning neural implicit surfaces by volume rendering for multi-view reconstruction. *Advances in Neural Information Processing Systems*, 34:27171–27183, 2021.

[48] Tengfei Wang, Bo Zhang, Ting Zhang, Shuyang Gu, Jianmin Bao, Tadas Baltrusaitis, Jingjing Shen, Dong Chen, Fang Wen, Qifeng Chen, et al. Rodin: A generative model for sculpting 3d digital avatars using diffusion. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 4563–4573, 2023.

[49] Yiming Wang, Qin Han, Marc Habermann, Kostas Daniilidis, Christian Theobalt, and Lingjie Liu. Neus2: Fast learning of neural implicit surfaces for multi-view reconstruction. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 3295–3306, 2023.

[50] Zhengyi Wang, Cheng Lu, Yikai Wang, Fan Bao, Chongxuan Li, Hang Su, and Jun Zhu. Prolificdreamer: High-fidelity and diverse text-to-3d generation with variational score distillation. *Advances in Neural Information Processing Systems*, 36, 2024.

[51] Jane Wilhelms and Allen Van Gelder. Octrees for faster isosurface generation. *ACM Transactions on Graphics (TOG)*, 11(3):201–227, 1992.

[52] Guanjun Wu, Taoran Yi, Jiemin Fang, Lingxi Xie, Xiaopeng Zhang, Wei Wei, Wenyu Liu, Qi Tian, and Xinggang Wang. 4d gaussian splatting for real-time dynamic scene rendering. *arXiv preprint arXiv:2310.08528*, 2023.

[53] Jiale Xu, Xintao Wang, Weihao Cheng, Yan-Pei Cao, Ying Shan, Xiaohu Qie, and Shenghua Gao. Dream3d: Zero-shot text-to-3d synthesis using 3d shape prior and text-to-image diffusion models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 20908–20918, 2023.

[54] Yinghao Xu, Zifan Shi, Wang Yifan, Hansheng Chen, Ceyuan Yang, Sida Peng, Yujun Shen, and Gordon Wetzstein. Grm: Large gaussian reconstruction model for efficient 3d reconstruction and generation. *arXiv preprint arXiv:2403.14621*, 2024.

[55] Taoran Yi, Jiemin Fang, Guanjun Wu, Lingxi Xie, Xiaopeng Zhang, Wenyu Liu, Qi Tian, and Xinggang Wang. Gaussiandreamer: Fast generation from text to 3d gaussian splatting with point cloud priors. *arXiv preprint arXiv:2310.08529*, 2023.

[56] Bowen Zhang, Yiji Cheng, Jiaolong Yang, Chunyu Wang, Feng Zhao, Yansong Tang, Dong Chen, and Baining Guo. Gaussiancube: Structuring gaussian splatting using optimal transport for 3d generative modeling. *arXiv preprint arXiv:2403.19655*, 2024.

[57] Kai Zhang, Sai Bi, Hao Tan, Yuanbo Xiangli, Nanxuan Zhao, Kalyan Sunkavalli, and Zexiang Xu. Gs-lrm: Large reconstruction model for 3d gaussian splatting. *arXiv preprint arXiv:2404.19702*, 2024.

[58] Heming Zhu, Yu Cao, Hang Jin, Weikai Chen, Dong Du, Zhangye Wang, Shuguang Cui, and Xiaoguang Han. Deep fashion3d: A dataset and benchmark for 3d garment reconstruction from single images. In *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part I 16*, pages 512–530. Springer, 2020.

[59] Zi-Xin Zou, Zhipeng Yu, Yuan-Chen Guo, Yangguang Li, Ding Liang, Yan-Pei Cao, and Song-Hai Zhang. Triplane meets gaussian splatting: Fast and generalizable single-view 3d reconstruction with transformers. *arXiv preprint arXiv:2312.09147*, 2023.

# A Source Codes

We provide our demonstration code as a part of our supplementary materials. We will release the source code, data, pretrained models and instructions upon acceptance.

# B More Experimental Details

## B.1 Gaussian Splatting Data Preparing

DiffGS takes fitted 3DGS as input for learning generative modeling. To prepare the 3DGS dataset of ShapeNet, we uniformly render 100 views from the ground truth meshes with blender to first obtain the dense multi-view images for each 3D shape in the chair and airplane classes of ShapeNet dataset. After that, we leverage the vanilla 3D Gaussian Splatting [21] method for fitting the 3DGS for each shape with the rendered multi-view images.

For achieving more stable and regularized 3DGS data for better generative modeling, we design some strategies for better initialization and optimization of 3DGS. (1) Since we fit 3DGS from existing 3D datasets with known geometries, we can simply sample dense point clouds uniformly from the surfaces as the perfect initialization for 3DGS optimization, instead of initializing with COLMAP points. The sampled point number is set to 100K. (2) We observe that optimizing 3DGS freely may often lead to some extremly large Gaussians. This will lead to unstable training of the Gaussian VAE and latent diffusion models, further affecting the generative modeling results. Therefore, we clip the scales at a maximum size of 0.01 to avoid the abnormal Gaussians.

## B.2 Gaussian Splatting Completion

We explore the task of Gaussian Splatting completion. Specifically, the Gaussian Splatting completion task is to recover the complete 3DGS from a partial 3DGS which contains large occlusions. In real-world applications, having only sparse views with limited viewpoint movement available for optimizing 3DGS often results in a partial 3DGS. Solving Gaussian Splatting completion task enables us to infer the complete and dense 3DGS from the partial ones for improving the rendering quality at invisible viewpoints.

**Data preparation.** We generate partial 3D Gaussian Splatting data from the complete datasets in a straightforward manner. First, we randomly divide each 3DGS into 8 chunks. Then, we occlude 7 chunks, leaving the remaining chunk as the partial 3DGS. This method allows us to prepare partial-complete 3DGS pairs for training and testing.

**Implementation.** To leverage DiffGS with partial 3DGS as the conditions for Gaussian Splatting completion, we leverage a modified PointNet [37] as the custom encoder $\gamma_{partial}$ for partial 3DGS, which projects the partial 3DGS with $K$ channels into a global partial 3DGS embedding. The DiffGS for Gaussian Splatting completion is trained with the target of Eq.(8) by introducing partial 3DGS embeddings through the cross-attention module.

## B.3 Point-to-Gaussian Generation

**Data preparation.** For the task of Point-to-Gaussian generation, we first prepare the training/testing data as point cloud-3DGS pairs obtained through the Gaussian Splatting Fitting process described in Sec. B.1. Specifically, the point clouds are generated by densely sampling 100K points on the ground truth meshes, while the paired 3DGS are obtained by optimizing with multi-view images rendered around the ground truth meshes. The data preparation for the DeepFashion [58] dataset follows the same process as for the ShapeNet [5] dataset.

Note that we train the Point-to-Gaussian DiffGS models for fitting the DeepFashion3D dataset, which contains only 563 garment instances. In contrast, we split the airplane and chair classes of the ShapeNet dataset into train/test sets to learn generalizable representations that enables DiffGS to predict novel appearances for unknown point cloud geometries. The results shown in Fig. 6 of the main paper include both the fitting results on the DeepFashion3D dataset and the generation results from the point clouds in the test set of the airplane and chair classes in the ShapeNet dataset.

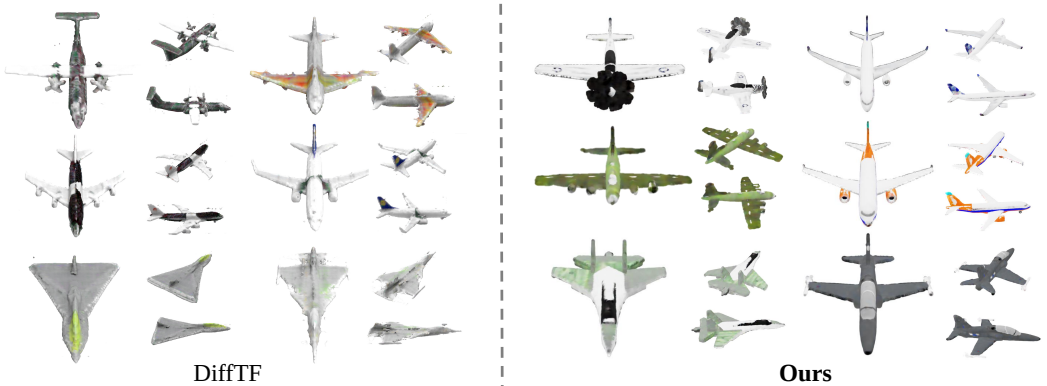



Figure 7: Visual comparisons with state-of-the-art method DiffTF [4] on unconditional generation of ShapeNet airplanes.

**Implementation.** For implementing Point-to-Gaussian, we train the Gaussian VAE using three-dimensional point clouds as inputs instead of 3DGS with attributes. Specifically, we replace the GS encoder in the Gaussian VAE with a PointNet-based network to learn representations from three-dimensional point clouds and recover Gaussian attributes from them. All architectures, optimization targets, and Gaussian extraction processes remain the same as in the Gaussian VAE, except for the encoder. Note that for the Point-to-Gaussian task, the Gaussian LDM is not trained, as we focus solely on decoding and extracting Gaussians from the point cloud inputs.

## C Implement Details

We implement DiffGS with Pytorch Lightning. We leverage the Adam optimizer with a learning rate of 0.0001. We train DiffGS with eight 3090 GPUs and the convergence in each class of ShapeNet dataset takes around 5 days. The Guassian encoder and Triplane decoder are implemented based on the SDF-VAE encoder and decoder of DiffusionSDF [10], where we modify the PointNet [37] in the SDF-VAE encoder to receive $K$ dimension 3DGS as the inputs.

## D Additional Ablation Study on Octree Depth

The depth of the octree in our Gaussian extraction algorithm is an important hyper-parameter in the framework. To explore how the octree depth affects the rendering quality, we conduct ablations on the octree depth as shown in Fig. 4. The results demonstrate that a larger octree depth leads to better quality by capturing more geometry details.

Table 4: Ablations on Gaussian numbers.

| Depth | PSNR | SSIM | LPIPS |
|---|---|---|---|
| 7 | 29.77 | 0.9772 | 0.0254 |
| 8 | 31.70 | 0.9834 | 0.0156 |
| 9 | 32.70 | 0.9842 | 0.0162 |
| 10 | **34.01** | **0.9879** | **0.0149** |

## E More Visualization Comparisons

We further compare DiffGS with DiffTF [4] on the airplane class of the ShapeNet dataset. The visual comparison is shown in Fig. 7, where our proposed DiffGS achieves more visually appealing results than DiffTF. DiffTF often produces shape generations with blurry textures, resulting in poor rendering quality. In contrast, our method produces high-fidelity renderings with the generated high-quality 3DGS, accurately capturing both geometry and appearances.

## F Limitation

One limitation of our method is that it sometimes produces overly creative color schemes. We illustrate this issue with two examples in Fig. 8. As shown, the failure cases of DiffGS result in excessively colorful appearances. For instance, the chair shown exhibits a color transition from

Figure 8: Failure cases of our method. DiffGS sometimes generates overly creative shapes with colorful appearances.

yellow to blue to green and finally to orange. These overly creative generations may not accurately reflect real-world shapes.

## G Video

We provide a video as part of our supplementary materials, which includes visualizations of the unconditional generations, image/text conditional generations, Gaussian Splatting completion results, and Point-to-Gaussian generations.

# NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: Our main claims made in the abstract and introduction accurately reflect the paper's contributions and scope.

   Guidelines:

   - The answer NA means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A No or NA answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: We analysis the limitations of our method in Sec.F of the appendix.

   Guidelines:

   - The answer NA means that the paper has no limitation while the answer No means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory Assumptions and Proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

   Answer: [NA]

Justification: The paper does not include theoretical results.

Guidelines:

- The answer NA means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental Result Reproducibility**

Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

Answer: [Yes]

Justification: We provide the detailed information in reproducing our methods in Sec.3, Sec.4 of the main paper and the appendix. We also provide a demonstration code of our method in the supplementary materials.

Guidelines:

- The answer NA means that the paper does not include experiments.
- If the paper includes experiments, a No answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
- If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
- Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
- While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
  (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
  (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
  (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
  (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

Answer: [Yes]

Justification: We provide our demonstration code as a part of our supplementary materials. We will release the source code, data and instructions upon acceptance.

Guidelines:

- The answer NA means that paper does not include experiments requiring code.
- Please see the NeurIPS code and data submission guidelines (https://nips.cc/public/guides/CodeSubmissionPolicy) for more details.
- While we encourage the release of code and data, we understand that this might not be possible, so "No" is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
- The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (https://nips.cc/public/guides/CodeSubmissionPolicy) for more details.
- The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
- The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
- At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
- Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental Setting/Details**

Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer, etc.) necessary to understand the results?

Answer: [Yes]

Justification: We provide the training and testing details in the experiment section (Sec.4) and the appendix.

Guidelines:

- The answer NA means that the paper does not include experiments.
- The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
- The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment Statistical Significance**

Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

Answer: [No]

Justification: We report the average performance as the experimental results.

Guidelines:

- The answer NA means that the paper does not include experiments.
- The authors should answer "Yes" if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
- The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).

- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g. negative error rates).
- If error bars are reported in tables or plots, The authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments Compute Resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: The computer resources needed to reproduce the experiments are provided in Sec.C of the appendix.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code Of Ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics https://neurips.cc/public/EthicsGuidelines?

   Answer: [Yes]

   Justification: The research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics.

   Guidelines:

   - The answer NA means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer No, they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader Impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [Yes]

    Justification: We discuss the applications and potential impacts of our method in the introduction.

    Guidelines:

    - The answer NA means that there is no societal impact of the work performed.
    - If the authors answer NA or No, they should explain why their work has no societal impact or why the paper does not address societal impact.

- Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pretrained language models, image generators, or scraped datasets)?

    Answer: [NA]

    Justification: The paper poses no such risks.

    Guidelines:

    - The answer NA means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
    - We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes]

    Justification: We use the open-sourced datasets under their licenses.

    Guidelines:

    - The answer NA means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.
    - For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
    - If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.

- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New Assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [NA]

    Justification: The paper does not release new assets.

    Guidelines:

    - The answer NA means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and Research with Human Subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [NA]

    Justification: The paper does not involve crowdsourcing nor research with human subjects.

    Guidelines:

    - The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
    - According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional Review Board (IRB) Approvals or Equivalent for Research with Human Subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [NA]

    Justification: The paper does not involve crowdsourcing nor research with human subjects.

    Guidelines:

    - The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
    - Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
    - We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
    - For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.