

# StableDreamer: Taming Noisy Score Distillation Sampling for Text-to-3D

Pengsheng Guo Hans Hao Adam Caccavale Zhongzheng Ren Edward Zhang  
Qi Shan Aditya Sankar Alexander G. Schwing Alex Colburn Fangchang Ma  
Apple



Figure 1. StableDreamer generates high-quality 3D geometry and appearance, represented as anisotropic 3D Gaussians, from the input text prompts. StableDreamer reduces the commonly seen multi-face Janus problem, improves local details, and converges robustly without requiring a mesh representation, modifying the SDS loss, or using any additional 3D or multi-view priors.

## Abstract

In text-to-3D generation, utilizing 2D diffusion models through score distillation sampling (SDS) [25] frequently leads to issues such as blurred appearances and multi-faced geometry, primarily due to the intrinsically noisy nature of the SDS loss. Our analysis identifies the core of these challenges as the interaction among noise levels in the 2D diffusion process, the architecture of the diffusion network, and the 3D model representation. To overcome these limitations, we present StableDreamer, a methodology incorporating three advances. First, inspired by Instruct-NeRF2NeRF [7], we formalize the equivalence of the SDS generative prior and a simple supervised L2 reconstruction loss. This finding provides a novel tool to debug SDS, which we use to show the impact of time-annealing noise levels on reducing multi-faced geometries. Second, our analysis shows that while image-space diffusion contributes to

geometric precision, latent-space diffusion is crucial for vivid color rendition. Based on this observation, StableDreamer introduces a two-stage training strategy that effectively combines these aspects, resulting in high-fidelity 3D models. Third, we adopt an anisotropic 3D Gaussians representation, replacing NeRFs, to enhance the overall quality, reduce memory usage during training, and accelerate rendering speeds, and better capture semi-transparent objects. StableDreamer reduces multi-face geometries, generates fine details, and converges stably.

## 1. Introduction

Recent advances in Generative AI have marked a paradigm shift across various domains, with notable progress in dialogue generation (e.g., ChatGPT [22]), image generation [27, 29, 30] and video synthesis [9, 39]. However, despite its immense potential, 3D generation still lags behind

in these developments. A critical obstacle in 3D generation is the limited size of available datasets, which pale in comparison to the extensive databases used in language [38] and image fields [31]. To circumvent this lack of 3D datasets, recent efforts such as DreamFusion [25] leverage 2D text-to-image models by using Score Distillation Sampling to generate 3D models from text prompts, showing exciting results with compelling appearance and geometry.

However, these text-to-3D approaches are far from perfect. Several critical issues persist. First, the generated 3D assets frequently exhibit over-saturated colors and blurry appearance. Fine local details are often omitted, giving results a somewhat “toy-like” quality. Second, the generated 3D asset’s geometry tends to be oversimplified, lacking the ability to faithfully represent thin or intricate shapes. Furthermore, these approaches are notorious for exhibiting the “Janus problem”, where the generated 3D object contains multiple canonical views seen from different viewpoints. Lastly, the optimization and rendering speed are hampered by the nature of test-time optimization and the underlying NeRF representation.

In response to the aforementioned challenges, we introduce a simple text-to-3D framework *StableDreamer*. We start with an empirical analysis that yields two pivotal insights: first, SDS loss can be conceptualized as a supervised reconstruction task using denoised images as ground truth, paving the way for a visualization tool to inspect the training dynamics, and motivating a noise-level annealing to stabilize SDS training. Second, we observe that image-space diffusion excels in geometric accuracy but falls short in color vibrancy. In contrast, latent-space diffusion enhances color at the expense of geometric fidelity. This leads us to develop a dual-phase training scheme, leveraging distinct diffusion architectures to optimize overall generation quality. Notably, we establish that these observations are agnostic to the underlying 3D representations with broad applicability. A third noteworthy innovation within StableDreamer is the adoption of 3D Gaussians [11] as the fundamental 3D representation. This choice offers a host of distinct advantages, including high fidelity for local details and fast rendering speed. However, directly substituting this representation into existing SDS frameworks leads to low-quality results and artifacts, likely due to the mismatch between noisy SDS loss and the localized nature of 3D Gaussians. To mitigate this, we implement strategies on initialization and density control, achieving a robust convergence to high-quality 3D Gaussians. In summary, our contributions are threefold:

- Interpreting SDS as a reparametrized supervised reconstruction problem, leading to new visualization that motivates the use of an annealing schedule for noise levels.
- A two-stage training framework that combines image and latent diffusion for enhanced geometry and color quality.
- Integration of 3D Gaussians for text-to-3D generation,

with novel regularization techniques for improved quality and convergence, to further improve fidelity and details. With these simple changes, StableDreamer reduces the multi-face geometry problem and produces a high level of fidelity and details in the synthesized models. StableDreamer is stable in training, without the need for switching between different 3D representations [15], modification of the SDS loss [42], or additional 3D or multi-view *a priori* [4, 43]. Our experiments establish StableDreamer’s improvements over leading text-to-3D models.

## 2. Related Work

**Text-to-3D.** Since the advent of large vision-language models [26, 29, 30], the research community has increasingly focused on the generation of 3D assets from textual input. Early approaches [18] utilize the CLIP embedding [26] for alignment between rendered images and text prompts. The seminal work DreamFusion [25] and SJC [40] distill the score of learned text-to-image diffusion models [29, 30] into optimizing neural 3D models (*e.g.*, NeRF [20]). These works demonstrate more realistic and high-fidelity results and subsequently became the de facto solutions in this field.

Following the success of DreamFusion/SJC, numerous follow-up works have advanced the field. These approaches encompass a spectrum of ideas including multi-stage refinement [15], geometry and appearance disentanglement [2], and improved the score distillation loss [42]. In this work, we study strategies that would enable stable training of a single 3D representation under the SDS framework, without having to convert to meshes (*e.g.*, Magic3D [15] and ProlificDreamer [42]), designing a different loss (*e.g.*, ProlificDreamer [42], NFSD [10]), or relying on other 3D or multi-view *a priori* that is trained on additional datasets (*e.g.*, GSGEN [4]).

**Neural 3D Representations.** Neural 3D representations originated in the context of 3D reconstruction [3, 17, 23], where neural networks implicitly learned signed distance functions and occupancy functions. This implicit modeling was then extended to the task of novel-view synthesis [16, 19, 32], yielding remarkable rendering outcomes. Subsequent works [1, 28, 41, 45] continued refining neural 3D representations from diverse perspectives; for a comprehensive overview, readers are directed to Tewari et al. [37]. A noteworthy trend [21, 35, 44] involves the adoption of hybrid implicit-explicit representations, inducing more spatially localized gradient changes for faster training and improved quality. Most recently, Kerbl et al. [11] popularized 3D Gaussians as an innovative, explicit scene representation. In this work, we incorporate a 3D Gaussians representation and regularized score distillation sampling (SDS) during training. This integration promotes fast convergence and enhances the overall quality of the generated

scenes. We diverge in a few details, such as using diffuse color without the spherical harmonics, and we adopt a customized initialization and density control strategy. Parallel efforts such as GSGEN [4], DreamGaussian [36] and GaussianDreamer [43] have concurrently chosen 3D Gaussians as the representation. However, GSGEN [4] and GaussianDreamer [43] both require an additional 3D prior during training. DreamGaussian [36] uses 3D Gaussians only as coarse initialization for extracting a mesh, whereas we produce high quality 3D Gaussians directly.

**Image Generative Models.** Generative models for images have been an active area of research, leading to significant advances in the generation of realistic and high-quality 2D content. Early approaches like Variational Autoencoders (VAEs) [12], Generative Adversarial Networks (GANs) [5], and Normalizing Flows [13] laid the foundation for this field. In recent years, diffusion models [8, 33, 34] have demonstrated exceptional capabilities in generating large-scale, high-fidelity images with precise textual control over content and style. In this work, we aim to ensure a robust and stable training process with the SDS loss. To accomplish this, we incorporate both an image-space diffusion model, DeepFloyd IF [14], and a latent-space diffusion model, Stable Diffusion [29]. This strategic combination is employed due to the distinct yet complementary guidance these models offer in the context of text-to-3D generation.

### 3. Preliminaries and Notation

In this section we briefly introduce the background on both Score Distillation Sampling (SDS) and 3D Gaussians.

**Score Distillation Sampling (SDS).** SDS is a loss introduced in DreamFusion [25] for generating a 3D scene model (such as a NeRF [20]) from a text prompt  $y$  using a pretrained 2D diffusion model. Starting with a randomly initialized scene model, parameterized by  $\theta$ , we iteratively sample random viewpoints  $\pi$  facing the object, and render an RGB image  $\mathbf{x}$  using differentiable rendering, i.e.  $\mathbf{x} = g(\theta, \pi)$ . This rendered RGB image  $\mathbf{x}$  is treated as an image to be denoised with a pretrained 2D diffusion model to obtain an improved image that better aligns with the text prompt. The image  $\mathbf{x}$  is perturbed with additive Gaussian noise  $\epsilon \sim \mathcal{N}(0, 1)$  such that

$$\mathbf{x}_t = \sqrt{\bar{\alpha}_t} \mathbf{x} + \sqrt{1 - \bar{\alpha}_t} \epsilon, \quad (1)$$

where the noise hyperparameter  $t$  determines the magnitude of  $\bar{\alpha}_t$ , predefined through a fixed variance schedule. The diffusion network typically predicts the added noise  $\hat{\epsilon}$ . While the diffusion process is iterative, Eq. (1) suggests a

one-step prediction of the denoised image as

$$\hat{\mathbf{x}}(\mathbf{x}_t; t, y) = \frac{\mathbf{x}_t - \sqrt{1 - \bar{\alpha}_t} \hat{\epsilon}(\mathbf{x}_t; t, y)}{\sqrt{\bar{\alpha}_t}}. \quad (2)$$

Note that these equations are based on Equations 4 and 15 in the DDPM paper [8].

The DreamFusion authors find that omitting the poorly conditioned diffusion network Jacobian term from the typical diffusion training loss gradient gives a more stable gradient for backpropagation to the current scene model, resulting in the SDS loss gradient

$$\nabla_{\theta} \ell_{\text{SDS}} (\mathbf{x} = g(\theta, \pi)) \triangleq \mathbb{E}_{t, \epsilon} \left[ w_t (\hat{\epsilon}(\mathbf{x}_t; y, t) - \epsilon) \frac{\partial \mathbf{x}}{\partial \theta} \right]. \quad (3)$$

In DreamFusion, this is shown to be the gradient of a weighted probability density distillation loss. In Sec. 4.1, we explore a more intuitive interpretation of the SDS loss that leads to a natural tool for visualization.

**3D Gaussian Splatting** 3D Gaussians is an explicit 3D representation popularized by [11], where the scene is comprised of a large set of semitransparent anisotropic 3D Gaussians. These Gaussian primitives are geometrically parameterized by covariance (or equivalently scale and rotation) and position, with appearance parameterized by color and opacity. This representation has been shown to achieve remarkable results in the area of novel-view synthesis, with significantly higher quality and rendering speed compared to previous volumetric methods based on radiance fields. To render 3D Gaussians, each primitive is projected into a screen space 2D Gaussian and sequentially rasterized in a back-to-front manner using alpha-blending. For screen-space positions  $\mu_i$ , screen-space covariances  $\Sigma_i$ , colors  $c_i$ , and opacities  $\sigma_i$ , the per-primitive alpha values and the final composited rendered color at pixel position  $x$  are

$$\begin{aligned} \alpha_i(x) &= \sigma_i e^{-\frac{1}{2}(x - \mu_i)^T \Sigma_i^{-1} (x - \mu_i)} \\ C(x) &= \sum_i c_i \alpha_i(x) \prod_{j < i} (1 - \alpha_j(x)) \end{aligned}$$

This rendering process is fully differentiable (given a differentiable sorting subroutine), enabling its use as a representation for text-to-3D generation.

### 4. StableDreamer

In a nutshell, StableDreamer addresses both the common blurry appearance and the multi-face geometry problems in SDS training with three conceptually simple modifications: (1) time-annealing of noise levels for 2D diffusion, which reduces multi-face geometries; (2) a dual-phase training that utilizes image-space diffusion for accurate geometry

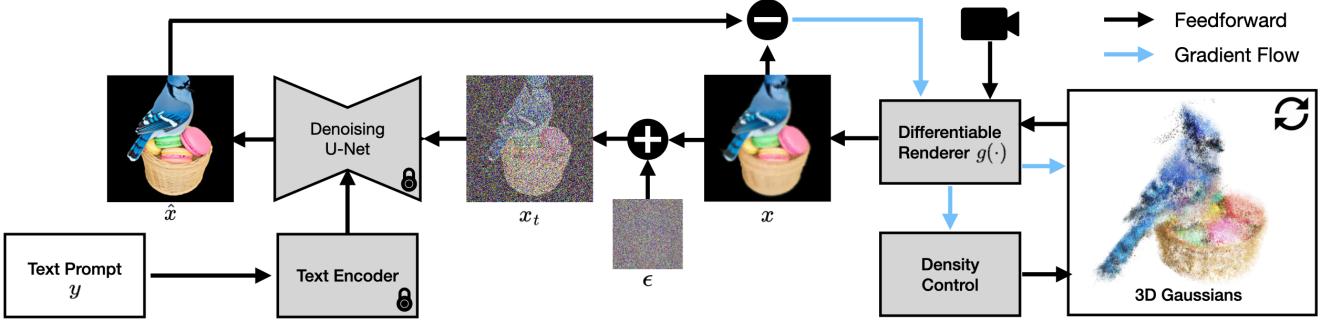


Figure 2. Our pipeline, StableDreamer, is an iterative optimization framework for creating anisotropic 3D Gaussians from text prompts. It begins with a text prompt as input, which is then processed by a fixed, pretrained text encoder to generate a text embedding. This embedding serves as conditioning input for our pretrained diffusing U-nets. During each iteration, we randomly sample a viewpoint and render the 3D Gaussians into an RGB image, which is subsequently input into the U-net for denoising and enhancement. The discrepancies between the denoised images and the originally rendered images are utilized as gradients for updating the anisotropic 3D Gaussians.

and subsequently a latent-space diffusion for vibrant and sharp appearances; and (3) integration of 3D Gaussians with regularization and density control that aims to improve model capacity for local details and transparent objects.

#### 4.1. Inspecting and Taming SDS Loss

A key challenge of optimization with the SDS loss is the noisy gradients inherent in the formulation. To address this, we first propose a novel interpretation that links it to NeRF reconstruction (specifically, Instruct-NeRF2NeRF [7]). This theoretical connection leads to two practical benefits: an annealing strategy for noise levels to improve convergence and a new visualization tool for inspecting the training dynamics of SDS.

**The SDS Generative Prior and NeRF Reconstruction.** In the DreamFusion training paradigm, the 3D scene representation is treated as an image generator while the SDS loss is treated as a prior over the generated images. While this probability-based interpretation allows the use of statistical tools (e.g. [42]), a more practical lens is suggested in a different related work. Instruct-NeRF2NeRF [7] is a recent work that also uses generative 2D models, albeit for a style transfer application rather than text-to-3D generation. In this work, the usual supervised reconstruction framework is used where a set of ground truth images is compared against a rendering from the current scene model. During training, Instruct-NeRF2NeRF uses the generative model to iteratively replace individual ground truth images with results from the 2D image generator (which may not be multiview-consistent) based on the current rendering result from that viewpoint. The authors note that their training process can be interpreted as a variant of SDS. Here we make this connection explicit:

**Proposition 1.** *Training a 3D scene representation with the*

*SDS generative prior is mathematically equivalent (up to scale) to using L2 reconstruction loss against images generated from the 2D generator.*

*Proof.* Without loss of generality, consider the SDS loss with an image-space diffusion model without classifier-free guidance. We use Eqs. (1) and (2) to expand the noise residual:

$$\begin{aligned} \hat{\epsilon}(\mathbf{x}_t; t, y) - \epsilon &= \frac{\mathbf{x}_t - \sqrt{\bar{\alpha}_t} \hat{\mathbf{x}}(\mathbf{x}_t; y, t)}{\sqrt{1 - \bar{\alpha}_t}} - \epsilon \\ &= \frac{\sqrt{\bar{\alpha}_t} \mathbf{x} + \sqrt{1 - \bar{\alpha}_t} \epsilon - \sqrt{\bar{\alpha}_t} \hat{\mathbf{x}}(\mathbf{x}_t; y, t)}{\sqrt{1 - \bar{\alpha}_t}} - \epsilon \\ &= \frac{\sqrt{\bar{\alpha}_t}}{\sqrt{1 - \bar{\alpha}_t}} (\mathbf{x} - \hat{\mathbf{x}}(\mathbf{x}_t; y, t)) \end{aligned}$$

Then, the gradient of the SDS loss is implemented as

$$\begin{aligned} \nabla_{\theta} \ell_{\text{SDS}}(\mathbf{x} = g(\theta, \pi)) &\triangleq w(t) (\hat{\epsilon}(\mathbf{x}_t; y, t) - \epsilon) \frac{\partial \mathbf{x}}{\partial \theta} \\ &= w(t) \frac{\sqrt{\bar{\alpha}_t}}{\sqrt{1 - \bar{\alpha}_t}} (\mathbf{x} - \hat{\mathbf{x}}(\mathbf{x}_t; y, t)) \frac{\partial \mathbf{x}}{\partial \theta}, \end{aligned}$$

which is exactly the gradient of a scaled L2 loss  $\ell_{L2}(\mathbf{x}, \hat{\mathbf{x}}) = \frac{\beta(t)}{2} \|\mathbf{x} - \hat{\mathbf{x}}\|^2$  between the current rendering  $\mathbf{x}$  and ground truth image  $\hat{\mathbf{x}}(\mathbf{x}_t; y, t)$ , with  $\beta(t) = \frac{w(t)\sqrt{\bar{\alpha}_t}}{\sqrt{1 - \bar{\alpha}_t}}$ . For latent-space diffusion models, a similar line of reasoning shows that SDS loss is instead equivalent to a latent-space L2 loss.  $\square$

**Annealing of Noise Level.** The above discussion establishes a novel perspective where the one-step denoised image  $\hat{\mathbf{x}}$ , as defined in Eq. (2), is conceptualized as the ground truth image in the context of NeRF reconstruction. This insight yields significant implications for the noise level scheduling in the 2D diffusion process. Particularly, to ensure effective convergence during SDS training, it is crucial

that the variance of these ground truth images starts large and decreases as training advances. To achieve this, we dynamically adjust the noise distribution’s upper and lower limits, progressively narrowing the range with training iterations. We use a piecewise linear schedule for the upper and lower bounds that converge by the end of the training. Guiding this noise magnitude is critical, since excessive noise leads to larger gradient magnitudes (equivalent to having a changing ground truth), which can lead to worse model convergence as shown later in Sec. 5.3. Incidentally, ProlificDreamer [42] proposes a similar but simpler annealing strategy, reducing noise level after initial iteration steps.

**Visualization of Supervision Signals.** A second advantage of implementing the proposed SDS loss reparameterization lies in the enhanced interpretability of the training process. Through the visualization of the pseudo-ground-truth image  $\hat{x}$  throughout the training phase, we gain insights into the direct influence of different hyperparameters on target images. This capability empowers us to devise a more robust training scheme, effectively taming the inherent noise in SDS loss for text-to-3D tasks.

A common challenge for 3D generation from text is the tendency for these systems to form objects with multiple faces. By examining the latent images we find a relationship between the multi-face problem and the SDS noise parameter. Figure 3 shows the predicted original images  $\hat{x}$  from two training runs with different noise levels. For the run with larger noise the system is more likely to hallucinate a face on the back of the dog’s head. Since each iteration is conditioned on the previous state, repeated selection of large noise values can cause the model to converge to a geometry with many faces. On the flip side, using lower noise levels reduces the signal to the optimization as the latent images do not change between iterations. Taken together, these results suggest we should use an annealing strategy for the added noise where it begins with a larger range and narrows as the training progresses.

Similarly, the visualizations of the one-step denoised image  $\hat{x}$  for various guidance scales in Fig. 4 provide insight into the effect of the guidance scale hyperparameter. Lower values lead to smooth images lacking fine details, while larger values hallucinate high-frequency details and oversaturated colors. This can lead to fake-looking images as shown in Sec. 5.3. While the effect this parameter is already understood, this simple example highlights the insights made possible by this reparameterization.

## 4.2. A Tale of Two Diffusions: Image vs. Latent

The current landscape of diffusion models in the literature bifurcates into two categories: image-space diffusion and latent-space diffusion. Image-space models, such as DeepFloyd [14] and Imagen [30], directly apply noise to the im-

ages. In contrast, latent-space models like Stable Diffusion [24, 29] necessitate an encoder-decoder pair that maps between the image and latent spaces, applying noise only in the latent domain. Our empirical analysis reveals that these two model types exhibit different guidance directions for text-to-3D. We propose an effective two-stage training framework that leverages their distinct properties. As shown in Fig. 5, the proposed framework can produce sharp texture and detailed geometry. Incidentally, Magic3D [15] arrives at a similar training strategy but mainly for reasons of speed and resolution, rather than quality.

### Image-space diffusion for geometry reconstruction.

For the first stage of training, we propose to use the image-space model, DeepFloyd [14], to train the 3D model. The primary goal at this stage is to converge to a reasonable rough geometry, so that a detailed appearance can be learned later in the optimization, as shown in the first row of Fig. 5. Therefore, in this stage, we only use the coarse DeepFloyd model, operating at  $64 \times 64$  resolution. At this stage, all the parameters of the 3D models are learnable. A low learning rate is used for the geometry as it converges (see Appendix C for more detailed analysis).

### Latent-space diffusion for appearance enhancement.

While the coarse reconstruction successfully yields a 3D model with satisfactory geometric accuracy, it tends to fall short in terms of visual quality due to its use of low-resolution 2D image supervision at  $64 \times 64$  resolution. The primary objective of the refinement stage is to significantly enhance the visual fidelity of the 3D model, as shown in the second row of Fig. 5. To achieve this, we employ a latent-space diffusion model, Stable Diffusion (SDv2.1-base) [29] trained with  $512 \times 512$  resolution images. As shown in Appendix D, the image-space diffusion models are not suitable to get the detailed appearance for the 3D model (even for a high-resolution model like DeepFloyd with super-resolution modules). We hypothesize that this is due to view-inconsistent pixel-level guidance, resulting in a blurred model and the loss of appearance detail. In contrast, the guidance from the latent-space diffusion model is less sensitive to this issue, since the loss is calculated in the latent space after feature compression from the image encoder. As a result, with the guidance from Stable Diffusion at the second stage, we largely increase model fidelity for both appearance and geometry.

## 4.3. Integrating 3D Gaussians

The aforementioned training scheme provides stabilized training with NeRF, yet there is potential for further enhancement in the finer details. 3D Gaussians offer advantages such as rapid rendering speeds and enhanced local representation over other NeRF representations. However, they

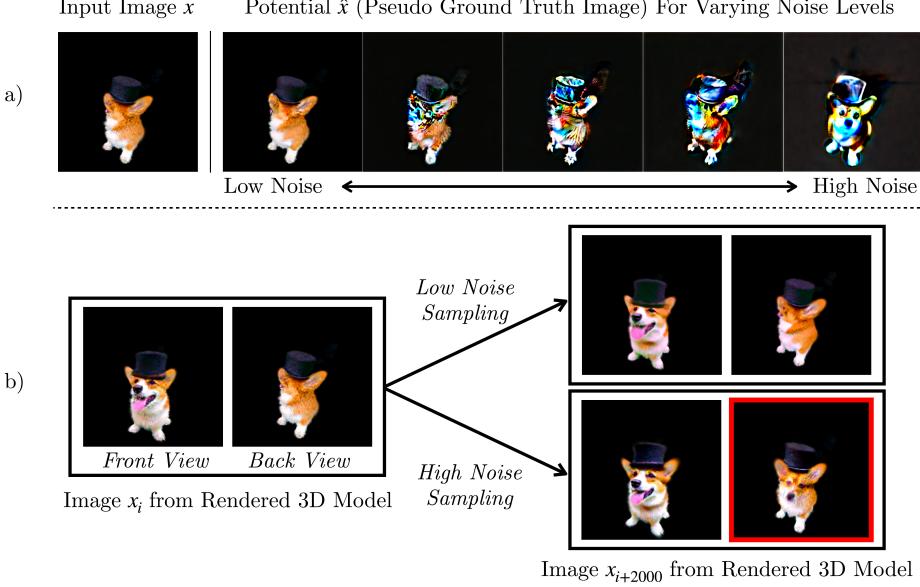


Figure 3. a): Per Proposition 1, the reformulated loss equation enables visualization of the one-step denoised image  $\hat{x}$  which allows us to observe the effect of modifying the level of noise being injected into  $x_t$  in Eq. (1) and subsequently  $\hat{x}$  in Eq. (2). Less noise produces images closer to the input image  $x$  while larger noise levels produce more variation.

b): Two training runs are compared, one biased to sample lower noise (top) and one biased to sample higher noise (bottom). Two views are rendered at both an early iteration  $i$  and later iteration  $i + 2000$ . From a), high noise samples are associated with a face incorrectly hallucinated on the back of the dogs head. Unsurprisingly, the model with larger noise ends up converging to a multi-faced dog.



Figure 4. Understanding the impact of guidance scale on the appearance via visualizing the one-step denoised images  $\hat{x}$  during training. Left-to-right: Guidance scale 10, 20, 35, and 100. As the guidance scale increases, so does the high frequency detail and color, eventually leading to an unrealistic image.



Figure 5. Results from two training stages. Stage 1 (top): image-space diffusion (DeepFloyd) produces accurate geometry at the cost of muted colors. Stage 2 (bottom): we finetune with latent-space diffusion (Stable Diffusion) to enhance the appearance.

are sensitive to the hyper-parameters and training strategies. In fact, directly substituting this representation into our existing training frameworks leads to low-quality results and artifacts, likely due to the mismatch between noisy SDS loss and the localized nature of 3D Gaussians. Specifically, we observe that despite having on average 10x larger gradient magnitude compared to other learnable parameters (*e.g.*, colors, scales, rotation), the position variables exhibit a "random walk" behavior without converging to a high-quality geometry. This observation motivates specialized 3D Gaussians training strategies around initialization and density control.

**Initialization.** In 3DGs [11], Structure-from-Motion (SfM) is used to initialize the Gaussian locations for scene reconstruction. However, this method cannot be used in text-to-3D generation. Thus, we use a simple alternate approach that has proved compatible with a wide range of text prompts. To start, the centers of the Gaussian primitives are randomly sampled with a uniform distribution over a volume. While the positions are uniformly distributed, the opacity of each point is initialized relative to its proximity to the center of the volume. More specifically, the initial opacity linearly decays with distance from the origin. This simple heuristic helps with convergence since the majority of generated objects have most of their density closer to the center of the scene.

**Density control.** Our experiments show that the position learning of 3D Gaussians is hard to tune and easily diverges with large learning rates due to the noisy signal from SDS loss. To stabilize training, a small learning rate is required for the position variables to avoid moving too far from their initial locations. Consequently, we cannot solely rely on position learning to produce fine geometry. Therefore, we turn to density control for geometry construction. Specifically, after initialization, we apply periodic densification and pruning, gradually adding new points in order to produce finer geometry and appearance. Additionally, we find that resetting the opacities to near zero at early training stages helps reduce floaters and bad geometry. Please refer to [Appendix B](#) for details of our implementation.

## 5. Experiments

We compare StableDreamer against several state-of-the-art text-to-3d methods on the overall quality of the synthesized 3D geometry and appearance as well as memory usage during training and rendering speed. More ablation studies can be found in our appendices.

### 5.1. Comparison To Prior Methods

As shown in Fig. 6, StableDreamer achieves state-of-the-art results compared to baseline works including DreamFusion [25], Magic3D [15], GSGen [4], and ProlificDreamer [42]. StableDreamer’s initial coarse geometric optimization converges to accurate geometry, greatly reducing the occurrence of multi-faced geometry commonly seen in the baseline methods. Tab. 1 presents an efficiency analysis of our method in comparison to baseline approaches. Our method, employing 3D Gaussians, renders at > 30FPS while maintaining reasonable training time and minimal GPU memory usage. Notably, Magic3D tends to produce over-saturated color while ProlificDreamer and GS-Gen achieve similar detailed textures but consistently produce multi-faced or otherwise incorrect geometries (additional visualization in [Appendix A](#)).

### 5.2. Generalization Across 3D Representations

We showcase the efficacy of 3D Gaussians compared to volumetric radiance fields, specifically iNGP [21]. iNGP [21] was widely adopted in previous work [2, 15, 42] thanks to its speed compared to classical MLP-based implicit neural representations [25]. To ensure an equitable evaluation, both 3D Gaussians and iNGP were trained with the proposed training scheme. The qualitative results are reported in the two rightmost columns in Fig. 6. Our training scheme is generalizable beyond 3D Gaussians and works well on iNGP. Overall, 3D Gaussians still produce better local details than iNGP, supporting our choice of 3D representation. For detailed structures (*e.g.* hairs from corgi and bunny),

iNGP typically produces either blurry or noisy surface textures, while 3D Gaussians generate realistic detailed structures. iNGP also results in temporal aliasing and flickering, which is visible only in videos.

Quantitative efficiency measurements, presented in Tab. 1, indicate the advantages of 3D Gaussians. With a similar parameter count, 3D Gaussians utilize 82% less GPU memory and render 6 times faster faster than iNGP [21]. Interestingly, training time between the two methods remained comparable, largely owing to the fact that the 2D diffusion models constitute the dominant time-consuming component in the forward process, especially in the coarse stage when rendering resolution is low.

	Training Time (min)	Peak Memory Usage (GB)	Render Speed (fps)
DreamFusion-iNGP (12.6M) [25]	<b>40</b>	17.6	14.0
Magic3D (12.6M) [15]	75	16.6	9.4
ProlificDreamer (12.6M) [42]	277	31.8	10.8
GSGen (4.7M) [4]	228	9.9	<b>52.5</b>
Ours-iNGP (12.6M)	81	31.9	7.38
Ours-3DGS (14M)	97	<b>5.7</b>	46.0

Table 1. Comparison of parameter count, training time, memory usage, and render speed. The evaluations are performed on a single NVIDIA V100 GPU. DreamFusion and Magic3D are not open-sourced so we use the Threestudio implementation [6].

### 5.3. Ablation on SDS Annealing

A critical aspect of the optimization processes described in Fig. 2 is the addition of noise to the image generated by the 2D diffusion model. Noisy gradients are a common issue with SDS loss and, as shown in Sec. 4.1, crafting a schedule for the noise bounds is important for consistently converging to good results. Our results shown in Fig. 7 match what we find in our analysis of the visualizations of the one-step denoised images and demonstrate that high noise levels during training tend to produce artifacts and multi-faced geometry. Intuitively, as the model converges, less noise should be added each step once the optimization has settled into a single local minimum.

## 6. Failure Analysis

While our strategies are shown to reduce multi-face geometry, there remain scenarios where these methods do not yield satisfactory results, as illustrated in Fig. 8. For instance, some failures originate from the 2D diffusion model’s inability to accurately interpret the prompt, while others produce floating or blurry geometries. Multi-face geometry also still exists for certain prompts.

## 7. Conclusion

In this work, we introduce StableDreamer, a text-to-3D framework that addresses the blurry appearance and multi-faced geometry problems that are commonly seen in prior

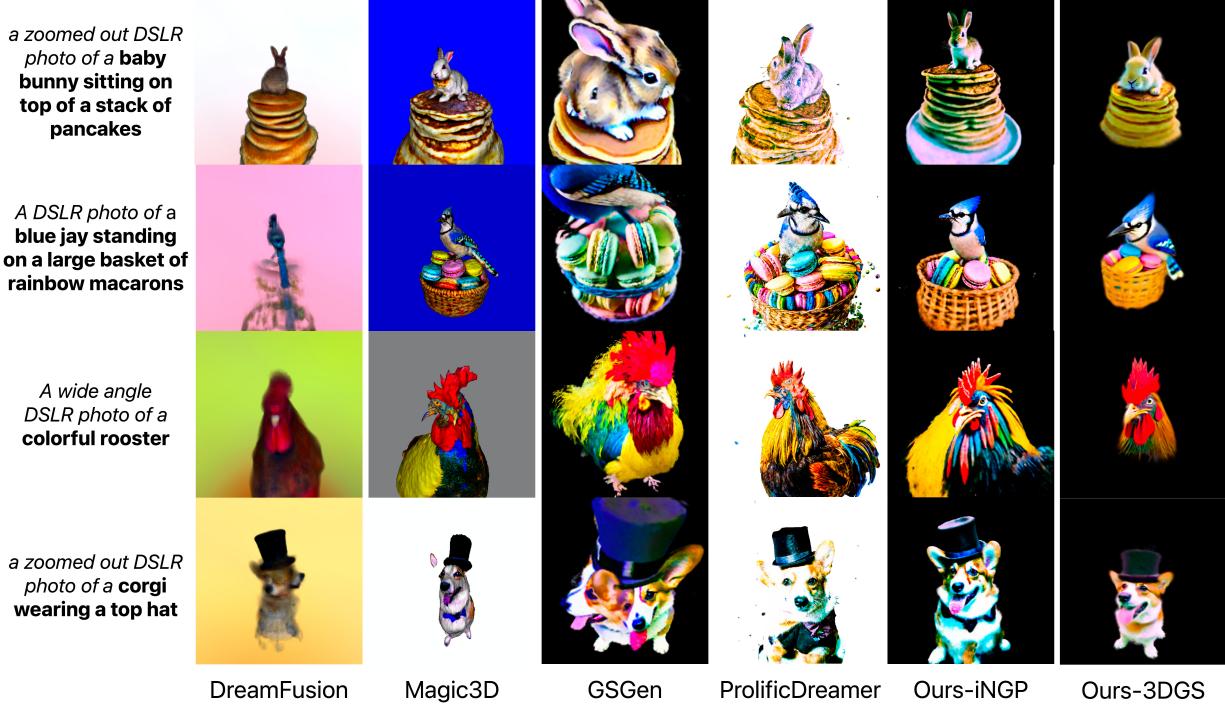


Figure 6. Comparison against prior methods. Prior methods typically have problems such as blurriness (DreamFusion [25]), multi-face geometry (Magic3D [15], GSgen [4], and ProlificDreamer), over-saturation in color (Magic3D [15]), cartoony appearances, or mismatch between content and text prompts. StableDreamer (including both iNGP [21] and 3D Gaussians [11] geometry primitives) achieves accurate geometry representation with fine details while preserving a realistic appearance. Results for DreamFusion and Magic3D use the open-source ThreeStudio implementation [6] since the authors have not released their code. Additional visualization are shown in [Appendix A](#).

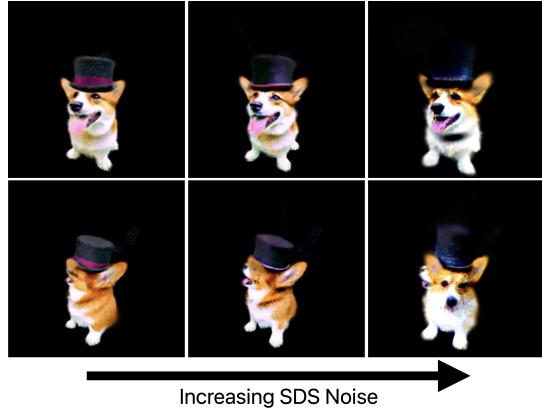


Figure 7. The upper and lower bounds of the noise being injected into  $\mathbf{x}_t$  in Eq. (1) change as a function of the training iteration. Larger noise levels give more high-frequency texture detail, but also more artifacts including multiple faces. The converged model is shown from the front (top row) and back (bottom row), with increasing levels of noise left-to-right.

methods. Our analysis reveals that the Score Distillation Sampling loss can be reparametrized as a supervised reconstruction loss using denoised images as pseudo-ground-truth. This finding leads to intuitive ways to visually inspect the training dynamics and the formulation noise level annealing strategies that reduce the occurrence of multi-



Figure 8. Failure cases: “An astronaut riding a kangaroo” with the astronaut being erroneously merged in; “A teddy bear pushing a shopping cart full of fruits and vegetables” with floaters; and “Michelangelo style statue of dog reading news on a cellphone” with multi-face and blurry geometries.

face artifacts. Empirical results show that image-space diffusion assists in generating better geometry while latent-space diffusion produces vibrant and detailed colors, inspiring our dual-phase training scheme. Notably, both the reparametrization and training schemes are agnostic to the underlying 3D representations and generalize beyond 3D Gaussians. However, to enhance detail and construction fidelity, we adopt a 3D Gaussians as our core 3D representation, including a number of strategies involving initialization and density control to enhance the robustness and convergence speed toward accurate geometric representations. Our empirical study demonstrates the superior quality of our method in comparison to previous approaches.

## References

- [1] Anpei Chen, Zexiang Xu, Andreas Geiger, Jingyi Yu, and Hao Su. Tensorf: Tensorial radiance fields. In *ECCV*, 2022. 2
- [2] Rui Chen, Yongwei Chen, Ningxin Jiao, and Kui Jia. Fantasia3d: Disentangling geometry and appearance for high-quality text-to-3d content creation. In *ICCV*, 2023. 2, 7
- [3] Zhiqin Chen and Hao Zhang. Learning implicit fields for generative shape modeling. *CVPR*, 2019. 2
- [4] Zilong Chen, Feng Wang, and Huaping Liu. Text-to-3d using gaussian splatting. *arXiv preprint arXiv:2309.16585*, 2023. 2, 3, 7, 8
- [5] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In *NeurIPS*, 2014. 3
- [6] Yuan-Chen Guo, Ying-Tian Liu, Ruizhi Shao, Christian Laforte, Vikram Voleti, Guan Luo, Chia-Hao Chen, Zi-Xin Zou, Chen Wang, Yan-Pei Cao, and Song-Hai Zhang. threestudio: A unified framework for 3d content generation. <https://github.com/threestudio-project/threestudio>, 2023. 7, 8, 2
- [7] Ayaan Haque, Matthew Tancik, Alexei Efros, Aleksander Holynski, and Angjoo Kanazawa. Instruct-nerf2nerf: Editing 3d scenes with instructions. In *ICCV*, 2023. 1, 4
- [8] Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. *NeurIPS*, 2020. 3
- [9] Jonathan Ho, William Chan, Chitwan Saharia, Jay Whang, Ruiqi Gao, Alexey Gritsenko, Diederik P Kingma, Ben Poole, Mohammad Norouzi, David J Fleet, et al. Imagen video: High definition video generation with diffusion models. *arXiv preprint arXiv:2210.02303*, 2022. 1
- [10] Oren Katzir, Or Patashnik, Daniel Cohen-Or, and Dani Lischinski. Noise-free score distillation. *arXiv preprint arXiv:2310.17590*, 2023. 2
- [11] Bernhard Kerbl, Georgios Kopanas, Thomas Leimkühler, and George Drettakis. 3d gaussian splatting for real-time radiance field rendering. *ACM Transactions on Graphics*, 2023. 2, 3, 6, 8, 1
- [12] Diederik P Kingma and Max Welling. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013. 3
- [13] Ivan Kobyzev, Simon JD Prince, and Marcus A Brubaker. Normalizing flows: An introduction and review of current methods. *IEEE TPAMI*, 2020. 3
- [14] Mikhail Konstantinov, Alex Shonenkov, Daria Bakshandaeva, and Ksenia Ivanova. Deepfloyd ai research band. <https://www.deepfloyd.ai/deepfloyd-if>, 2023. 3, 5
- [15] 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 *CVPR*, 2023. 2, 5, 7, 8
- [16] Stephen Lombardi, Tomas Simon, Jason Saragih, Gabriel Schwartz, Andreas Lehrmann, and Yaser Sheikh. Neural volumes: Learning dynamic renderable volumes from images. *ACM TOG*, 2019. 2
- [17] Lars Mescheder, Michael Oechsle, Michael Niemeyer, Sebastian Nowozin, and Andreas Geiger. Occupancy networks: Learning 3d reconstruction in function space. In *CVPR*, 2019. 2
- [18] Oscar Michel, Roi Bar-On, Richard Liu, Sagie Benaim, and Rana Hanocka. Text2mesh: Text-driven neural stylization for meshes. In *CVPR*, 2022. 2
- [19] 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. 2
- [20] 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. 2, 3
- [21] Thomas Müller, Alex Evans, Christoph Schied, and Alexander Keller. Instant neural graphics primitives with a multiresolution hash encoding. *ACM Trans. Graph.*, 2022. 2, 7, 8
- [22] OpenAI. Chatgpt: A pre-trained language model. <https://www.openai.com/research/chatgpt>, 2021. 1
- [23] Jeong Joon Park, Peter Florence, Julian Straub, Richard Newcombe, and Steven Lovegrove. Deepsdf: Learning continuous signed distance functions for shape representation. In *CVPR*, 2019. 2
- [24] Dustin Podell, Zion English, Kyle Lacey, Andreas Blattmann, Tim Dockhorn, Jonas Müller, Joe Penna, and Robin Rombach. Sdxl: Improving latent diffusion models for high-resolution image synthesis. *arXiv:2307.01952*, 2023. 5
- [25] Ben Poole, Ajay Jain, Jonathan T Barron, and Ben Mildenhall. Dreamfusion: Text-to-3d using 2d diffusion. *ICLR*, 2022. 1, 2, 3, 7, 8
- [26] 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. 2
- [27] Aditya Ramesh, Mukul Goyal, Oren Dovrat, Jitendra Ke, Shweta Lu, Alessandro Sordoni, Linjie Kang, Adam Ng, Tim Smith, Xi Choi, et al. Dall-e: Creating images from text. *arXiv preprint arXiv:2102.12092*, 2021. 1
- [28] Christian Reiser, Songyou Peng, Yiyi Liao, and Andreas Geiger. Kilonerf: Speeding up neural radiance fields with thousands of tiny mlps, 2021. 2
- [29] Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-resolution image synthesis with latent diffusion models. In *CVPR*, 2022. 1, 2, 3, 5
- [30] Chitwan Saharia, William Chan, Saurabh Saxena, Lala Li, Jay Whang, Emily L Denton, Kamyar Ghasemipour, Raphael Gontijo Lopes, Burcu Karagol Ayan, Tim Salimans, et al. Photorealistic text-to-image diffusion models with deep language understanding. *NeurIPS*, 2022. 1, 2, 5
- [31] Christoph Schuhmann, Romain Beaumont, Richard Vencu, Cade Gordon, Ross Wightman, Mehdi Cherti, Theo

- Coombes, Aarush Katta, Clayton Mullis, Mitchell Wortsman, et al. Laion-5b: An open large-scale dataset for training next generation image-text models. In *NeurIPS*, 2022. 2
- [32] Vincent Sitzmann, Michael Zollhoefer, and Gordon Wetzstein. Scene representation networks: Continuous 3d-structure-aware neural scene representations. In *NeurIPS*, 2019. 2
- [33] Jascha Sohl-Dickstein, Eric Weiss, Niru Maheswaranathan, and Surya Ganguli. Deep unsupervised learning using nonequilibrium thermodynamics. In *ICML*, 2015. 3
- [34] Yang Song and Stefano Ermon. Generative modeling by estimating gradients of the data distribution. *NeurIPS*, 2019. 3
- [35] Cheng Sun, Min Sun, and Hwann-Tzong Chen. Direct voxel grid optimization: Super-fast convergence for radiance fields reconstruction. In *CVPR*, 2022. 2
- [36] 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. 3
- [37] Ayush Tewari, Justus Thies, Ben Mildenhall, Pratul P. Srinivasan, Edgar Treitschke, Yifan Wang, Christoph Lassner, Vincent Sitzmann, Ricardo Martin-Brualla, Stephen Lombardi, Tomas Simon, Christian Theobalt, Matthias Nießner, Jonathan T. Barron, Gordon Wetzstein, Michael Zollhöfer, and Vladislav Golyanik. Advances in neural rendering. *CGF*, 2022. 2
- [38] Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, et al. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*, 2023. 2
- [39] Ruben Villegas, Mohammad Babaeizadeh, Pieter-Jan Kindermans, Hernan Moraldo, Han Zhang, Mohammad Taghi Saffar, Santiago Castro, Julius Kunze, and Dumitru Erhan. Phenaki: Variable length video generation from open domain textual description. *arXiv preprint arXiv:2210.02399*, 2022. 1
- [40] Haochen Wang, Xiaodan Du, Jiahao Li, Raymond A Yeh, and Greg Shakhnarovich. Score jacobian chaining: Lifting pretrained 2d diffusion models for 3d generation. In *CVPR*, 2023. 2
- [41] 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. In *NeurIPS*, 2021. 2
- [42] 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. In *NeurIPS*, 2023. 2, 4, 5, 7
- [43] Taoran Yi, Jiemin Fang, Guanjun Wu, Lingxi Xie, Xiaopeng Zhang, Wenyu Liu, Qi Tian, and Xinggang Wang. Gaussian-dreamer: Fast generation from text to 3d gaussian splatting with point cloud priors. *arxiv:2310.08529*, 2023. 2, 3
- [44] Alex Yu, Rui long Li, Matthew Tancik, Hao Li, Ren Ng, and Angjoo Kanazawa. PlenOctrees for real-time rendering of neural radiance fields. In *ICCV*, 2021. 2
- [45] Alex Yu, Vickie Ye, Matthew Tancik, and Angjoo Kanazawa. pixelNeRF: Neural radiance fields from one or few images. In *CVPR*, 2021. 2

# StableDreamer: Taming Noisy Score Distillation Sampling for Text-to-3D

## Supplementary Material

### Appendix A. Additional Visualization

Fig. app-1 shows additional result comparison with different view of angles. StableDreamer is able to generate the 3D model with both detailed texture and geometry compared to the baseline methods.

### Appendix B. Density Control Setup

Fig. app-2 shows an illustration of our density control setup. To assist with the convergence of the geometry of the scenes, we use the following schedule to modify the 3D Gaussians. Firstly, we randomly initialize 1000 points based on the aforementioned initialization scheme. As shown in Appendix C, we intend to use less starting points to reduce the floaters and produce better geometry. Then, for every 500 iterations we apply a densification process based on the original Gaussian splatting method [11]. More specifically, we split and clone the Gaussians when the magnitude of the position gradient is over a threshold. By doing so, we can allow the representation to better capture fine details. Please refer to the original paper [11] for more details of the densification algorithm. Note that we start this densification process after 100 iterations. This is to make sure the averaged positional gradients get stabilized. Similar to the original method, we also apply periodic pruning immediately after densification to remove the Guassians with smaller opacities or large 2D projected area. In addition, as shown in the ablation study in Appendix C, we found that resetting the opacities at the early training stage can help to reduce the floaters in the final result. In our setup, we choose to reset the opacities at the 1000th iteration. This is due to the positions and other attributes of the primitives have begin to converge before 1000 iteration, and resetting this parameters allows for a more robust convergence by preventing the optimization from getting caught in the initial local minima (e.g., floaters or bad geometry). The density control process ends at 12000 iterations; we then proceed with 3000 fine-tuning iterations with a fixed number of 3D Gaussians to smooth out the spiky artifacts introduced by densification.

### Appendix C. Ablation on Density Control

As shown in Figure app-2, to assist with the convergence of the geometry of the scenes, we use the following schedule to modify the 3D Gaussians. Firstly, we randomly initialize 1000 points based on the aforementioned initialization scheme. Then, every 500 iterations we apply a densification process based on the original Gaussian splatting

method [11]. More specifically, we split and clone the Gaussians when the magnitude of the position gradient is over a threshold. By doing so, we can allow the representation to better capture fine details. Please refer to the original paper [11] for more details of the densification algorithm. Note that we start this densification process after 100 iterations. This is to make sure the averaged positional gradients get stabilized.

**Initialization.** As shown in Fig. app-3, starting with fewer points and annealing the initial opacity of the Gaussians results in the best geometry. More specifically, comparing the results from the same row, the results with opacity decay in the right column (*i.e.*, linearly decaying opacity based on the distance to the origin) have less floaters. Furthermore, comparing the results from the same column, with more starting points (from top to bottom), there are more floaters and the training become unstable if we initialize with a large amount of points due to the noisy signal from SDS loss (see the figure on the bottom left).

**Density control and position learning** In our experiments, we found that resetting opacity for all of the Gaussians during densification can help to reduce floaters. As shown in Figure app-4, with opacity reset, there are much less floaters in the final result (bottom) compared with the case without opacity reset (top). Note that, in our experiment, we choose to reset the opacity to 0.005 at the iteration of 1000 based on grid search.

Besides opacity reset, we also found the representation of 3D Gaussians is very sensitive to the learning rate of the positions (*i.e.*, xyz coordinates). As shown in Figure app-5, with a slightly large learning rate (0.0064), the geometry gets diverged due to the diversification process. This is aligned with the result from original 3D Gaussians paper [11]. Even under their reconstruction task, which has more regularization (*i.e.* image supervision) comparing with our generation task, the original method still uses a really small position learning rate as 0.00064, which essentially does not allow the centroids of the 3D Gaussians moving much. Instead, the fine geometry is forced to be learned by density control (densification and pruning).

### Appendix D. Ablation on Two-Stage Training

**Benefit from the coarse-to-fine training paradigm.** Fig. app-6 shows the first stage result (*i.e.*, training from scratch) using Stable Diffusion model (left) and DeepFloyd

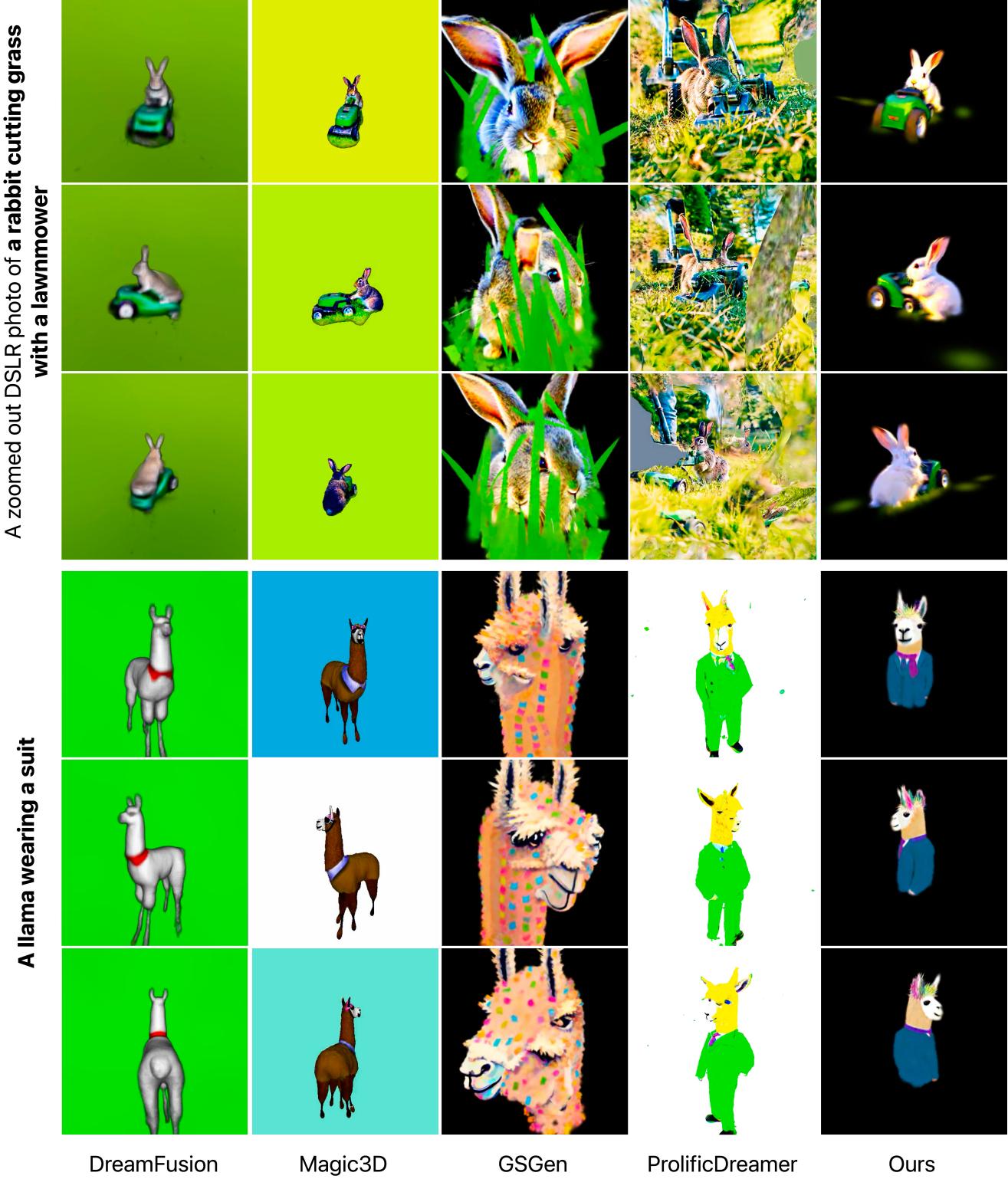


Figure app-1. Multi-view comparison against prior methods. Each column shows the generated object from 3 different views roughly equally spaced about the vertical axis. GSGen and ProlificDreamer struggle to produce 3D view-consistent geometry. DreamFusion and Magic3D do not have released code so we use the open-source Threeestudio implementation [6].

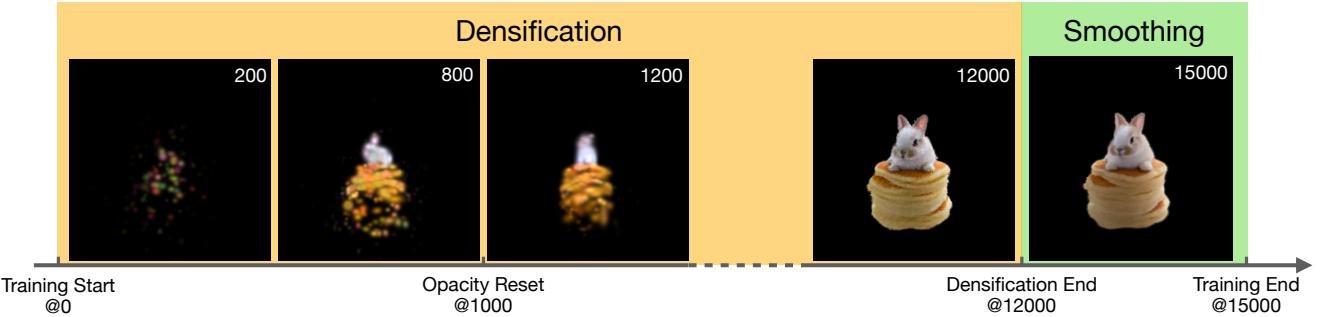


Figure app-2. Density control schedule. We randomly initialize points and apply density control (densification and pruning) to obtain the coarse geometry and texture. Then an additional smoothing step is followed in order to remove the spiky artifacts as introduced by densification.

model (right) for both of the geometry primitives 3D Gaussians and iNGP. Although there is a sharper texture from the high-resolution Stable Diffusion model, the overall geometry is worse than the result from the coarse DeepFloyd model. As shown in Fig. app-7, after finetuning with the diffusion models trained with high resolution images (Stable Diffusion or DeepFloyd with super-resolution module), we can get a 3D model with much higher fidelity, while also keeps the good geometry that is learned from the first stage.

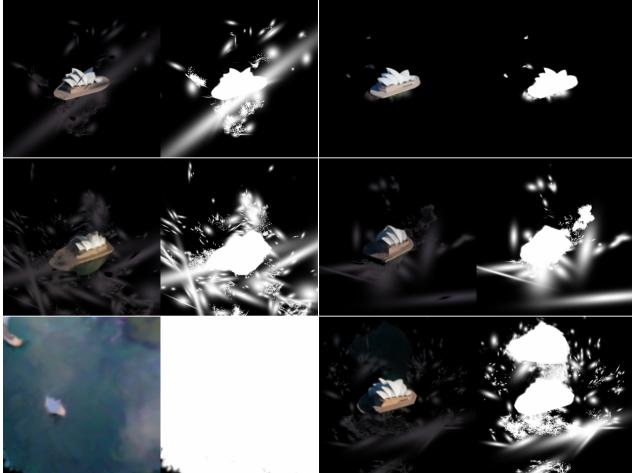


Figure app-3. Ablation study for 3D Gaussians initialization schemes with prompt: *a zoomed out DSLR photo of the Sydney opera house, aerial view*. Left Column: Fix initial opacity levels. Right Column: Opacity initialization based on distance to center of scene. Top Row: 1K starting points. Middle Row: 10K starting points. Bottom Row: 100K starting points.

**Benefit of the use of latent-space diffusion model in the second stage learning.** As shown in Fig. app-7, when finetuning from the first stage model trained with the coarse DeepFloyd model, both Stable Diffusion and DeepFloyd with super-resolution module can achieve better geometry and texture, as they are trained with high resolution images. However, if we compare the resulting images, (*e.g.*, the texture of basket and the fine hairs from bunny) the Deep-

Floyd result is lacking details, while the Stable Diffusion model can produce both better texture and sharper geometry. As mentioned earlier, this is due to the image-based guidance (*i.e.*, DeepFloyd) has more adverse effect to the view consistency of the 3D model, while the guidance from the latent-space diffusion model (*i.e.*, Stable Diffusion) is less sensitive due to the feature compression from its image encoder.



Figure app-4. Resetting opacity during densification can help reduce floaters as shown in the opacity renderings on the right. Top: without opacity reset; bottom: with opacity reset.



Figure app-5. Using an inappropriate learning rate for position updates can readily lead to geometric divergence.

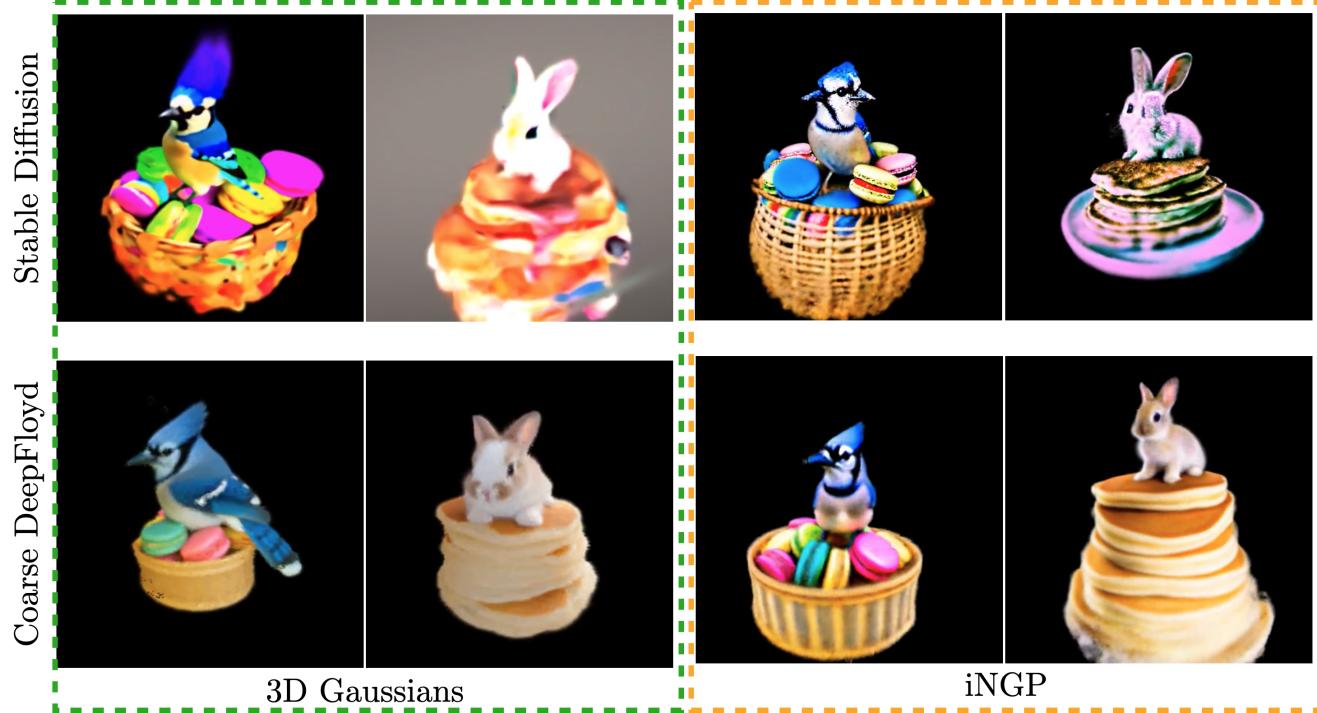


Figure app-6. Result from different diffusion models when training from scratch.

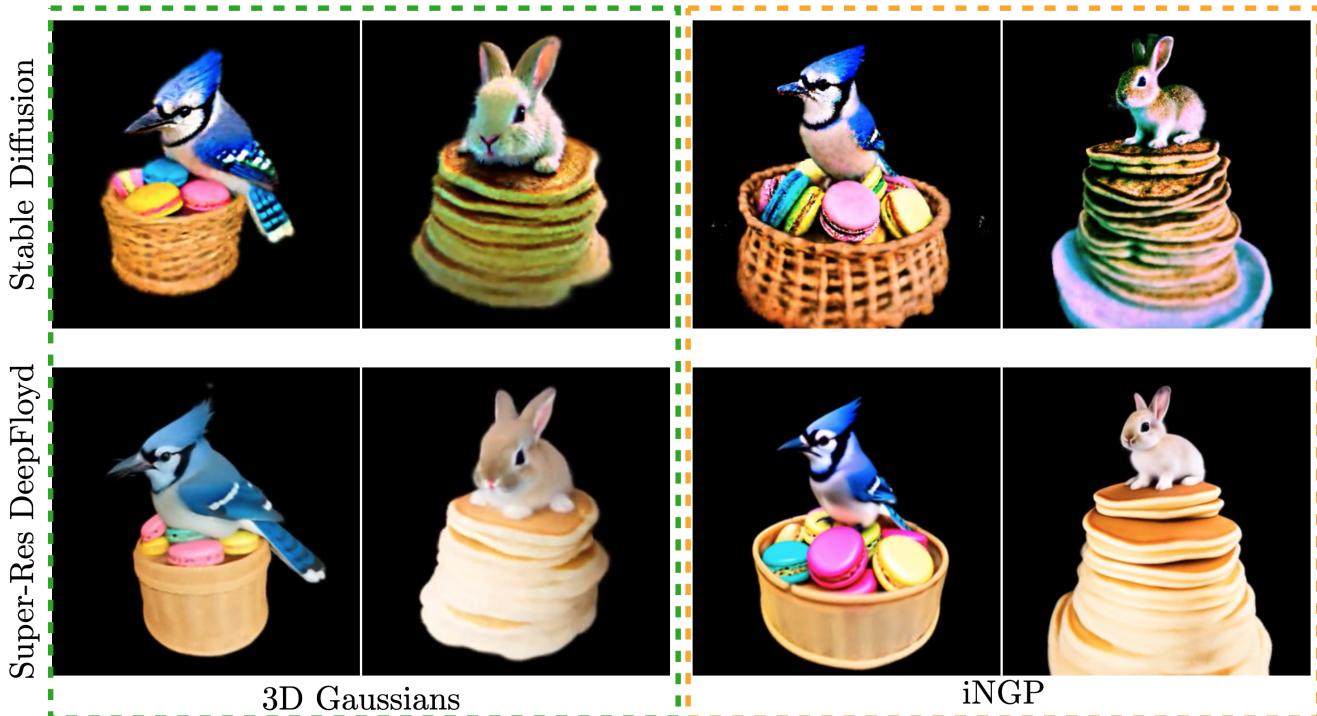


Figure app-7. Result from different diffusion models when finetuning from the first stage model.