# Score Jacobian Chaining: Lifting Pretrained 2D Diffusion Models for 3D Generation

Haochen Wang[*1]    Xiaodan Du[*1]    Jiahao Li[*1]    Raymond A. Yeh[2]    Greg Shakhnarovich[1]

[1]TTI-Chicago    [2]Purdue University

A zoomed out high-quality photo of Temple of Heaven

A high quality photo of a delicious burger

A high quality photo of a Victorian style wooden chair with velvet upholstery

A high quality photo of a classic silver muscle car

Figure 1. Results for text-driven 3D generation using Score Jacobian Chaining with Stable Diffusion as the pretrained model.

## Abstract

*A diffusion model learns to predict a vector field of gradients. We propose to apply chain rule on the learned gradients, and back-propagate the score of a diffusion model through the Jacobian of a differentiable renderer, which we instantiate to be a voxel radiance field. This setup aggregates 2D scores at multiple camera viewpoints into a 3D score, and re-purposes a pretrained 2D model for 3D data generation. We identify a technical challenge of distribution mismatch that arises in this application, and propose a novel estimation mechanism to resolve it. We run our algorithm on several off-the-shelf diffusion image generative models, including the recently released Stable Diffusion trained on the large-scale LAION 5B dataset.*

## 1. Introduction

We introduce a method that converts a pretrained 2D diffusion generative model on images into a 3D generative model of radiance fields, without requiring access to any 3D data. The key insight is to interpret diffusion models as

learned predictors of a gradient field, often referred to as the *score function* of the data log-likelihood. We apply the chain rule on the estimated score, hence the name Score Jacobian Chaining (SJC).

Following Hyvärinen [16], the score is defined as the gradient of the log-density function with respect to the data (rather than parameter). Diffusion models of various families [13, 49, 50, 52] can all be interpreted [20, 23, 52] as modeling $\nabla_{\boldsymbol{x}} \log p_\sigma(\boldsymbol{x})$ *i.e.* the denoising score at noise level $\sigma$. For readability, we refer to the denoising score as the score. Generating a sample from a diffusion model involves repeated evaluations of the score function from large to small $\sigma$ level, so that a sample $\boldsymbol{x}$ gradually moves closer to the data manifold. It can be loosely interpreted as gradient descent, with precise control on the step sizes so that data distribution evolves to match the annealed $\sigma$ level (ancestral sampler [13], SDE and probability-flow ODE [52], etc.). While there are other perspectives to a diffusion model [13, 49], here we are primarily motivated from the viewpoint that diffusion models produce a gradient field.

A natural question to ask is whether the chain rule can be applied to the learned gradients. Consider a diffusion model on images. An image $\boldsymbol{x}$ may be parameterized by some

---
\* Equal contribution.

function $f$ with parameters $\boldsymbol{\theta}$, *i.e.*, $\boldsymbol{x} = f(\boldsymbol{\theta})$. Applying the chain rule through the Jacobian $\frac{\partial \boldsymbol{x}}{\partial \boldsymbol{\theta}}$ converts a gradient on image $\boldsymbol{x}$ into a gradient on the parameter $\boldsymbol{\theta}$. There are many potential use cases for pairing a pretrained diffusion model with different choices of $f$. In this work we are interested in exploring the connection between 3D and multiview 2D by choosing $f$ to be a differentiable renderer, thus creating a 3D generative model using only pretrained 2D resources.

Many prior works [2, 58, 60] perform 3D generative modeling by training on 3D datasets [5, 24, 54, 59]. This approach is often as challenging as it is format-ambiguous. In addition to the high data acquisition cost of 3D assets [9], there is no universal data format: point clouds, meshes, volumetric radiance field, etc, all have computational trade-offs. What is common to these 3D assets is that they can be rendered into 2D images. An inverse rendering system, or a differentiable renderer [25, 27, 30, 34, 39], provides access to the Jacobian $\boldsymbol{J}_\pi \triangleq \frac{\partial \boldsymbol{x}_\pi}{\partial \boldsymbol{\theta}}$ of a rendered image $\boldsymbol{x}_\pi$ at camera viewpoint $\pi$ with respect to the underlying 3D parameterization $\boldsymbol{\theta}$. Our method uses differentiable rendering to aggregate 2D image gradients over multiple viewpoints into a 3D asset gradient, and lifts a generative model from 2D to 3D. We parameterize a 3D asset $\boldsymbol{\theta}$ as a radiance field stored on voxels and choose $f$ to be the volume rendering function.

A key technical challenge is that computing the 2D score by directly evaluating a diffusion model on a rendered image $\boldsymbol{x}_\pi$ leads to an out-of-distribution (OOD) problem. Generally, diffusion models are trained as denoisers and have only seen noisy inputs during training. On the other hand, our method requires evaluating the denoiser on non-noisy rendered images from a 3D asset during optimization, and it leads to the OOD problem. To address the issue, we propose *Perturb-and-Average Scoring*, an approach to estimate the score for non-noisy images.

Empirically, we first validate the effectiveness of *Perturb-and-Average Scoring* at solving the OOD problem and explore the hyperparameter choices on a simple 2D image canvas. Here we identify open problems on using unconditioned diffusion models trained on FFHQ and LSUN Bedroom. Next, we use Stable Diffusion, a model pretrained on the web-scale LAION dataset to perform SJC for 3D generation, as shown in Fig. 1. **Our contributions are as follows:**

- We propose a method for lifting a 2D diffusion model to 3D via an application of the chain rule.

- We illustrate the challenge of OOD when using a pretrained denoiser and propose *Perturb-and-Average Scoring* to resolve it.

- We point out the subtleties and open problems on applying *Perturb-and-Average Scoring* as gradient for optimization.

- We demonstrate the effectiveness of SJC for the task of 3D text-driven generation.

## 2. Related Works

**Diffusion models** have recently advanced to image generation on Internet-scale datasets [10, 36, 42, 44–47]. A diffusion model could be interpreted as either a VAE [13, 49] or a denoising score-matcher [50, 52, 56]. Notably, models trained under one regime can be directly used for inference and sampling by the other [20, 52]; they are in practice largely equivalent.

**Neural radiance fields (NeRF)** is a family of inverse rendering algorithms that have excelled at multiview 3D reconstruction tasks including view synthesis and surface geometry estimation [31, 34, 40, 57, 61]. Conceptually, a 3D asset is represented as a dense grid of RGB colors and spatial density $\tau$, and rendered into images in a way analogous to alpha compositing [32]. NeRF parameterizes the $(\text{RGB}, \tau)$ volume with a neural network, but querying the network densely in 3D incurs significant compute costs. Alternatively, Voxel NeRFs [6, 11, 19, 28, 53] store the volume on voxels and observe no loss in end task performance [11, 53]. Querying voxels is a simple memory operation that is much faster than a feedforward pass of a neural network. Here we use a customized voxel radiance field with hyperparameters based on DvGO [53] and TensoRF [6].

**2D-supervised 3D GANs** pioneered [35, 43, 63] the approach of training 3D generative models using only unstructured 2D images, and promise greater scalability in terms of data. Rather than supervising directly on the 3D asset a model generates, these methods supervise the 2D renderings of the generated 3D asset, often using an adversarial loss [3, 4, 38, 48, 64]. In other words, only images are needed as training data. However, training such a 3D generative model from scratch is still challenging [37]. Recent empirical evaluation remains mostly on human and animal faces [3]. Our method does the opposite: we take an image generative model that is *already pretrained* on large amounts of 2D data and use it to guide the iterative optimization of a 3D asset. Optimization-based generation makes it much slower compared to 3D GANs, but it becomes possible to harness powerful off-the-shelf 2D generative models such as Stable Diffusion [45] for greater content diversity.

**CLIP-guided, optimization-based 3D generative models** share a similar philosophy of optimizing 3D assets by guiding on 2D renderings [15, 17, 18, 22, 26, 33]. Among them, DreamFields [17] and PureClipNeRF [26] also use NeRF as their differentiable renderers. In this case, the 2D guidance comes from CLIP [42], a pretrained image-text matching model. These works optimize the 3D assets so that the image renderings match a user-provided text prompt. Since CLIP is not a 2D generative model per se, such a pipeline usually creates some abstract content [29] that looks very different from real images. In contrast, we use diffusion models, which are proper 2D generative models, to create realistic looking 3D

content.

**DreamFusion**. The recently arXived work by Poole et al. [41], *independent and concurrent to our work*, proposes an algorithm that is similar to our approach at the pseudo-code level. Differently, their procedure uses the mathematical setup by Graikos et al. [12] to search for image parametrization that minimizes the training loss of a diffusion model. In contrast, our work is motivated by applying the chain rule to the 2D score. The key differences have been summarized in Sec. 4.3. In terms of implementation, we do not have access to the close-sourced Imagen [46] diffusion model. Instead, we use the pretrained Stable Diffusion model released by Rombach et al. [45]. For a comparison with DreamFusion, we use with a third-party implementation based on the same diffusion model, namely Stable-DreamFusion [55].

## 3. Preliminaries

To establish a common notation, we briefly review the score-based perspective of diffusion models. For readers familiar with VAE literature on diffusion models, we provide a concise score-based formula card and more details in Appendix Sec. **??** to connect these ideas.

**Denoising score matching.** Given a dataset of samples $\mathcal{Y} = \{\boldsymbol{y}_i\}$ drawn from $p_{\text{data}}$, a diffusion model revolves primarily around learning a denoiser $D$ by minimizing the difference between a noised sample $\boldsymbol{y} + \sigma\boldsymbol{n}$ and $\boldsymbol{y}$,

$$\mathbb{E}_{\boldsymbol{y}\sim p_{\text{data}}}\mathbb{E}_{\boldsymbol{n}\sim\mathcal{N}(0,\boldsymbol{I})}\|D(\boldsymbol{y}+\sigma\boldsymbol{n};\sigma)-\boldsymbol{y}\|_2^2, \tag{1}$$

*i.e.* $D$ is denoising the input $\boldsymbol{y} + \sigma\boldsymbol{n}$, for a range of $\sigma$ values. For 2D images, $D$ is commonly chosen to be a ConvNet. Variants such as DDPM [13] parameterized the ConvNet to instead predict a noise residual $\hat{\epsilon}$, and these models can be converted back to the form of a denoiser by [52]

$$D(\boldsymbol{x};\sigma) = \boldsymbol{x} - \sigma\hat{\epsilon}(\boldsymbol{x}). \tag{2}$$

In this paper, we treat all pretrained diffusion models as denoisers, and perform the interface conversion in our implementation when needed.

**Score from denoiser.** Let $p_\sigma(\boldsymbol{x})$ denote the data distribution perturbed by Gaussian noise of standard deviation $\sigma$. It is shown in prior works [16, 50] that the denoiser $D$ trained according to Eq. (1) provides a good approximation to the denoising score:

$$\boldsymbol{\nabla}_{\boldsymbol{x}}\log p_\sigma(\boldsymbol{x}) \approx \frac{D(\boldsymbol{x};\sigma)-\boldsymbol{x}}{\sigma^2}. \tag{3}$$

A denoising diffusion model estimates the *score function* of the noised distribution $p_\sigma(\boldsymbol{x})$ at various $\sigma \in \{\sigma_i\}_{i=1}^T$. To perform sampling, the diffusion model gradually updates a sample through a sequence of noise levels of $\sigma_T > \cdots >$

$\sigma_0 = 0$. $\{\sigma_i\}$ are chosen empirically, with a typical range being $[0.01, 157]$ [13] in the case of DDPM.

**Score as mean-shift.** A helpful intuition is that the score behaves like *mean-shift* [7, 8]. If we simplify $p_{\text{data}}$ to be an empirical data distribution over the i.i.d. samples $\{\boldsymbol{y}_i\}$, then at noise level $\sigma$, $p_\sigma(\boldsymbol{x})$ takes the form of a mixture of Gaussians [51]

$$p_\sigma(\boldsymbol{x}) = \mathbb{E}_{\boldsymbol{y}\sim p_{\text{data}}}\mathcal{N}(\boldsymbol{x};\,\boldsymbol{y},\,\sigma^2\mathbf{I}). \tag{4}$$

In this case there exists a closed-form expression [20, 51] to the optimal denoiser

$$D(\boldsymbol{x};\sigma) = \frac{\sum_i \mathcal{N}(\boldsymbol{x};\,\boldsymbol{y}_i,\,\sigma^2\mathbf{I})\,\boldsymbol{y}_i}{\sum_i \mathcal{N}(\boldsymbol{x};\,\boldsymbol{y}_i,\,\sigma^2\mathbf{I})}. \tag{5}$$

In other words, $D(\boldsymbol{x};\,\sigma)$ is a locally weighted mean of data samples $\{\boldsymbol{y}_i\}$ around $\boldsymbol{x}$ under a Gaussian kernel with bandwidth $\sigma$. The denoising score function can be thought of as a non-parametric guide on how to update $\boldsymbol{x}$ in order to move it towards its *weighted nearest neighbors*.

## 4. Score Jacobian Chaining for 3D Generation

Let $\boldsymbol{\theta}$ denotes the parameters of a 3D asset, *e.g.*, voxel grid of $(\text{RGB}, \tau)$ as in Sec. 4.2. Our goal is to model and sample from the distribution $p(\boldsymbol{\theta})$ to generate a 3D scene. In our setting, only a pretrained 2D diffusion model on images $p(\boldsymbol{x})$ is given and we do not have access to 3D data. To relate the 2D and 3D distributions $p(\boldsymbol{x})$ and $p(\boldsymbol{\theta})$, we assume that the probability density of 3D asset $\boldsymbol{\theta}$ is proportional to the expected probability densities of its multiview 2D image renderings $\boldsymbol{x}_\pi$ over camera poses $\pi$, *i.e.*,

$$p_\sigma(\boldsymbol{\theta}) \propto \mathbb{E}_\pi\big[p_\sigma(\boldsymbol{x}_\pi(\boldsymbol{\theta}))\big], \tag{6}$$

up to a normalization constant $Z = \int \mathbb{E}_\pi\big[p_\sigma(\boldsymbol{x}_\pi(\boldsymbol{\theta}))\big]d\boldsymbol{\theta}$. That is, a 3D asset $\boldsymbol{\theta}$ is as likely as its 2D renderings $\boldsymbol{x}_\pi$.

Next, we establish a lower bound, $\log\tilde{p}_\sigma(\boldsymbol{\theta})$, on the distribution in Eq. (6) using Jensen's inequality:

$$\log p_\sigma(\boldsymbol{\theta}) = \log\big[\mathbb{E}_\pi(p_\sigma(\boldsymbol{x}_\pi))\big] - \log Z \tag{7}$$

$$\geq \mathbb{E}_\pi[\log p_\sigma(\boldsymbol{x}_\pi)] - \log Z \triangleq \log\tilde{p}_\sigma(\boldsymbol{\theta}). \tag{8}$$

Recall that the score is the *gradient* of log probability density of data. By chain rule

$$\boldsymbol{\nabla}_{\boldsymbol{\theta}}\log\tilde{p}_\sigma(\boldsymbol{\theta}) = \mathbb{E}_\pi\left[\boldsymbol{\nabla}_{\boldsymbol{\theta}}\log p_\sigma(\boldsymbol{x}_\pi)\right] \tag{9}$$

$$\frac{\partial\log\tilde{p}_\sigma(\boldsymbol{\theta})}{\partial\boldsymbol{\theta}} = \mathbb{E}_\pi\left[\frac{\partial\log p_\sigma(\boldsymbol{x}_\pi)}{\partial\boldsymbol{x}_\pi}\cdot\frac{\partial\boldsymbol{x}_\pi}{\partial\boldsymbol{\theta}}\right] \tag{10}$$

$$\underbrace{\boldsymbol{\nabla}_{\boldsymbol{\theta}}\log\tilde{p}_\sigma(\boldsymbol{\theta})}_{\text{3D score}} = \mathbb{E}_\pi[\,\underbrace{\boldsymbol{\nabla}_{\boldsymbol{x}_\pi}\log p_\sigma(\boldsymbol{x}_\pi)}_{\text{2D score; pretrained}}\cdot\underbrace{\boldsymbol{J}_\pi}_{\text{renderer Jacobian}}\,].$$
$$\tag{11}$$

We will next discuss how to compute the 2D score in practice using a pretrained diffusion model.
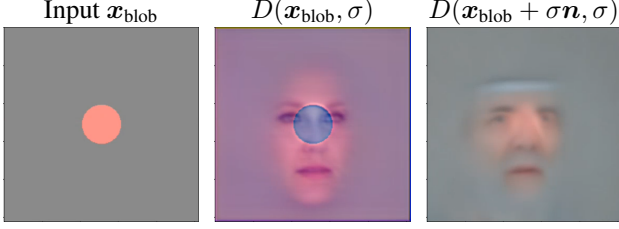
Figure 2. Illustration of denoiser's OOD issue using a denoiser pretrained on FFHQ. When directly evaluating $D(\boldsymbol{x}_{\text{blob}}, \sigma)$ the model did not correct for the orange blob into a face image. Contrarily, evaluating the denoiser on noised input $D(\boldsymbol{x}_{\text{blob}} + \sigma\boldsymbol{n}, \sigma)$ produces an image that successfully merges the blob with the face manifold.

## 4.1. Computing 2D Score on Non-Noisy Images

Computing the 3D score in Eq. (11) requires the 2D score on $\boldsymbol{x}_\pi$. A first attempt would be to directly apply the score from the denoiser in Eq. (3), *i.e.*,

$$\text{score}(\boldsymbol{x}_\pi, \sigma) \triangleq (D(\boldsymbol{x}_\pi; \sigma) - \boldsymbol{x}_\pi)/\sigma^2. \tag{12}$$

Unfortunately, evaluating the pretrained denoiser $D$ on $\boldsymbol{x}_\pi$ causes an out-of-distribution (OOD) problem. From the training objective in Eq. (1), at each noise level $\sigma$, the denoiser $D$ has only seen noisy inputs of the distribution $\boldsymbol{y} + \sigma\boldsymbol{n}$ where $\boldsymbol{y} \sim p_{\text{data}}$ and $\boldsymbol{n} \sim \mathcal{N}(0, \mathbf{I})$. However, a rendered image $\boldsymbol{x}_\pi$ from 3D asset $\boldsymbol{\theta}$ is generally not consistent with such distribution.

We illustrate this OOD situation in Fig. 2. Given a denoiser pretrained on FFHQ [21] by Baranchuk et al. [1], we visualize the output $D(\boldsymbol{x}_{\text{blob}}; \sigma = 6.5)$ where the input $\boldsymbol{x}_{\text{blob}}$ is a non-noisy image showing an orange blob centered on a grey canvas. Under the intuition that $D$ predicts a *weighted nearest neighbor* as reviewed in (5), we expect the denoiser to blend the orange blob with the manifold of faces. However in reality we observe sharp artifacts when updating with this score $(D(\boldsymbol{x}_{\text{blob}}; \sigma) - \boldsymbol{x}_{\text{blob}})/\sigma^2$ and the image becomes further away from the face manifold.

**Perturb-and-Average Scoring.** To address the OOD problem, we propose *Perturb-and-Average Scoring* (PAAS). It computes the score on non-noisy images $\boldsymbol{x}_\pi$ with a denoiser $D$ by adding noise to the input, and then considering the expectation of the predicted scores w.r.t. the random noise,

$$\text{PAAS}(\boldsymbol{x}_\pi, \sqrt{2}\sigma) \tag{13}$$

$$\triangleq \mathbb{E}_{\boldsymbol{n} \sim \mathcal{N}(0, \mathbf{I})} \left[ \text{score}(\boldsymbol{x}_\pi + \sigma\boldsymbol{n}, \sigma) \right] \tag{14}$$

$$= \mathbb{E}_{\boldsymbol{n}} \left[ \frac{D(\boldsymbol{x}_\pi + \sigma\boldsymbol{n}, \sigma) - (\boldsymbol{x}_\pi + \sigma\boldsymbol{n})}{\sigma^2} \right] \tag{15}$$

$$= \mathbb{E}_{\boldsymbol{n}} \left[ \frac{D(\boldsymbol{x}_\pi + \sigma\boldsymbol{n}, \sigma) - \boldsymbol{x}_\pi}{\sigma^2} \right] - \underbrace{\mathbb{E}_{\boldsymbol{n}} \left[ \frac{\boldsymbol{n}}{\sigma} \right]}_{=0}. \tag{16}$$

In practice, we use the Monte Carlo estimate of the expectation in Eq. (16). The algorithm is illustrated in Fig. 3. Given
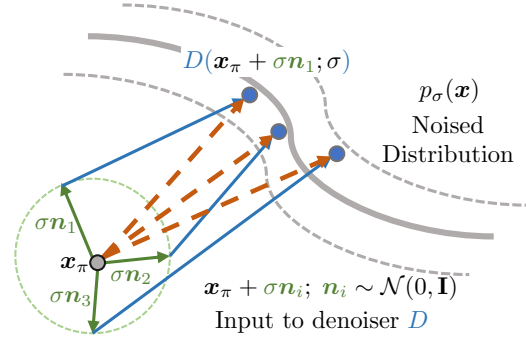


Figure 3. Computing PAAS on 2D renderings $\boldsymbol{x}_\pi$. Directly evaluating $D(\boldsymbol{x}_\pi; \sigma)$ leads to an OOD problem. Instead, we add noise to $\boldsymbol{x}_\pi$, and evaluate $D(\boldsymbol{x}_\pi + \sigma\boldsymbol{n}; \sigma)$ (blue dots). The PAAS is then computed by averaging over the brown dashed arrows, corresponding to multiple samples of $\boldsymbol{n}$. See Sec. 4.1 for details.

a set of sampled noises $\{\boldsymbol{n}_i\}$, each $D(\boldsymbol{x}_\pi + \sigma\boldsymbol{n}_i)$ provides an update direction on the perturbed input $\boldsymbol{x}_\pi + \sigma\boldsymbol{n}_i$. By averaging over the noise perturbations $\{\boldsymbol{n}_i\}$, we obtain an update direction on $\boldsymbol{x}_\pi$ itself.

**Justifying PAAS in Eq. (13).** We show that Perturb-and-Average Scoring provides an approximation to the score on $\boldsymbol{x}_\pi$ at an inflated noise level of $\sqrt{2}\sigma$

$$\text{PAAS}(\boldsymbol{x}_\pi, \sqrt{2}\sigma) \approx \nabla_{\boldsymbol{x}_\pi} \log p_{\sqrt{2}\sigma}(\boldsymbol{x}_\pi). \tag{17}$$

---

**Lemma 1** *Assuming an empirical data distribution $p_\sigma(\boldsymbol{x})$ in Eq. (4), for any $\boldsymbol{x} \in \mathbb{R}^d$*

$$\log p_{\sqrt{2}\sigma}(\boldsymbol{x}) \geq \mathbb{E}_{\boldsymbol{n} \sim \mathcal{N}(0, \mathbf{I})} \log p_\sigma(\boldsymbol{x} + \sigma\boldsymbol{n}). \tag{18}$$

---

*Proof.* Observe that the LHS of Eq. (19) is a convolution of two Gaussians, therefore

$$\mathbb{E}_{\boldsymbol{n} \sim \mathcal{N}(0, \mathbf{I})} \left[ \mathcal{N}(\boldsymbol{x} + \sigma\boldsymbol{n}; \boldsymbol{\mu}, \sigma^2\mathbf{I}) \right] = \mathcal{N}(\boldsymbol{x}; \boldsymbol{\mu}, 2\sigma^2\mathbf{I}) \tag{19}$$

Recall that $p_\sigma(\boldsymbol{x})$ is a mixture of Gaussians per Eq. (4);

$$p_{\sqrt{2}\sigma}(\boldsymbol{x}) = \mathbb{E}_{\boldsymbol{y} \sim p_{\text{data}}} \mathcal{N}(\boldsymbol{x}; \boldsymbol{y}, 2\sigma^2\mathbf{I}) \tag{20}$$

$$= \mathbb{E}_{\boldsymbol{y} \sim p_{\text{data}}} \mathbb{E}_{\boldsymbol{n} \sim \mathcal{N}(0, \mathbf{I})} \mathcal{N}(\boldsymbol{x} + \sigma\boldsymbol{n}; \boldsymbol{y}, \sigma^2\mathbf{I}) \tag{21}$$

$$= \mathbb{E}_{\boldsymbol{n} \sim \mathcal{N}(0, \mathbf{I})} \mathbb{E}_{\boldsymbol{y} \sim p_{\text{data}}} \mathcal{N}(\boldsymbol{x} + \sigma\boldsymbol{n}; \boldsymbol{y}, \sigma^2\mathbf{I}) \tag{22}$$

$$= \mathbb{E}_{\boldsymbol{n} \sim \mathcal{N}(0, \mathbf{I})} p_\sigma(\boldsymbol{x} + \sigma\boldsymbol{n}). \tag{23}$$

Taking the $\log$ on both sides of Eq. (23) and by Jensen's inequality, we arrive at Eq. (18). □

---

**Claim 1** *Assuming a trained denoiser $D$ as in Eq. (3), our PAAS$(\boldsymbol{x}_\pi, \sqrt{2}\sigma)$ in Eq. (13) computes the gradient w.r.t. a lower bound of $\log p_{\sqrt{2}\sigma}(\boldsymbol{x})$.*

---

*Proof.* By taking the gradient of the RHS of Lemma 1,

$$\nabla_{\boldsymbol{x}} \mathbb{E}_{\boldsymbol{n}} \log p_\sigma(\boldsymbol{x} + \sigma\boldsymbol{n}) = \mathbb{E}_{\boldsymbol{n}} \nabla_{\boldsymbol{x} + \sigma\boldsymbol{n}} \log p_\sigma(\boldsymbol{x} + \sigma\boldsymbol{n})$$

$$= \mathbb{E}_{\boldsymbol{n}}[\text{score}(\boldsymbol{x}_\pi + \sigma\boldsymbol{n}, \sigma)]. \tag{24}$$

which is the proposed PAAS algorithm in Eq. (13). □

## 4.2. Inverse Rendering on Voxel Radiance Field

With the computation of the 2D score resolved, the other half of our setup in Eq. (11) requires access to the Jacobian of a differentiable renderer.

**3D Representation.** We represent a 3D asset $\boldsymbol{\theta}$ as a voxel radiance field [6, 11, 53], which is much faster to access and update compared to a vanilla NeRF parameterized by a neural network [34]. The parameters $\boldsymbol{\theta}$ consist of a density voxel grid $\mathbf{V}^{(\text{density})} \in \mathbb{R}^{1 \times N_x \times N_y \times N_z}$ and a voxel grid of appearance features $\mathbf{V}^{(\text{app})} \in \mathbb{R}^{c \times N_x \times N_y \times N_z}$. Conventionally the appearance features are simply the RGB colors and $c = 3$. For simplicity, we do not model view dependencies in this work.

**Inverse Volumetric Rendering.** Image rendering is performed independently along a camera ray through each pixel. We cut a camera light ray into equally distanced segments of length $d$, and at the spatial location corresponding to the beginning of the $i$-th segment we sample a $(\text{RGB}_i, \tau_i)$ tuple from the color and density grids using trilinear interpolation. These values are alpha-composited using volume rendering quadrature [32] into the pixel color $C = \sum_i w_i \cdot \text{RGB}_i$, where

$$w_i = \alpha_i \cdot \prod_{j=0}^{i-1}(1 - \alpha_j); \quad \alpha_i = 1 - \exp(-\tau_i d). \quad (25)$$

Volume rendering of $\boldsymbol{\theta}$ is directly differentiable. At a rendered image $\boldsymbol{x}_\pi$, the Vector-Jacobian product in Eq. (11) between $\text{PAAS}(\boldsymbol{x}_\pi)$ and the Jacobian $J_\pi = \frac{\partial \boldsymbol{x}_\pi}{\partial \boldsymbol{\theta}}$ is computed by back-propagating the score through Eq. (25). This Vector-Jacobian product provides us with the 3D gradient needed for generative modeling on the voxel radiance field.

**Regularization Strategies.** The voxel grid is a very powerful 3D representation for volumetric rendering. Given noisy 2D guidance, the model may cheat by populating the entire grid with small densities such that the combined effect from one view hallucinates a plausible image. We propose several techniques to encourage the formation of a coherent 3D structure.

*Emptiness Loss:* Ideally, the space should be sparse with near zero densities except at the object. We propose an emptiness loss to encourage sparsity on a ray $\mathbf{r}$:

$$\mathcal{L}_{\text{emptiness}}(\mathbf{r}) = \frac{1}{n} \sum_{i=1}^{n} \log(1 + \beta \cdot w_i), \quad (26)$$

where $w_i$ are the alpha-composited weights shown in (25). The shape of the $\log$ function imposes severe penalties at the onset of small weights, but does not grow aggressively if the weights are large. It is consistent with our aim to eliminate small densities. The hyperparameter $\beta$ controls the steepness of the loss function near 0. A larger $\beta$ will put more emphasis on eliminating low-density noise. We set $\beta = 10$.
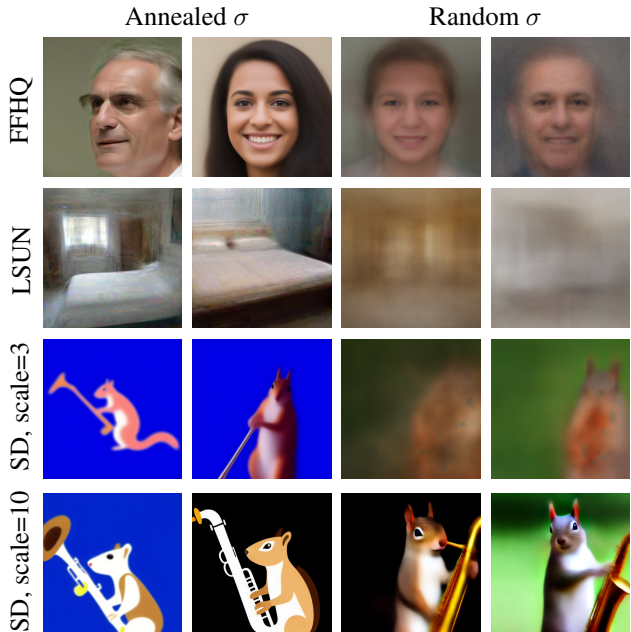


Figure 4. Sampling 2D images with Perturb-and-Average Scoring. We compare Annealed vs Random $\sigma$ schedule against several diffusion models. Row 1 & 2: the random $\sigma$ schedule exhibits strong mode-seeking behavior, and it results in low-quality "mean" images on unconditioned diffusion models trained on FFHQ and LSUN Bedroom. In this case, we need a carefully designed annealed $\sigma$ schedule to produce better, more diverse samples. Row 3 & 4: Stable Diffusion (SD) is conditioned on the prompt "a squirrel holding a saxophone". The use of natural language makes the conditioned distribution much easier to sample from. When the guidance scale is elevated to 10, Random $\sigma$ schedule that fails on FFHQ and LSUN starts to produce crisp, clean images.

*Emptiness Loss Schedule:* We use a hyperparameter $\lambda$ to control the contribution of the emptiness loss. If we apply a large emptiness loss, it will hinder the learning of geometry in the early stage of training. But if the emptiness is too small, there will be floating density artifacts. We adopt a two-stage noise elimination schedule to deal with this problem. In the first $K$ iterations, we use a relatively small weighting factor $\lambda_1$. After the $K^{\text{th}}$ iteration, it is increased to a larger $\lambda_2$. In our experiments $\lambda_1 = 1 \times 10^4$ and $\lambda_2 = 2 \times 10^5$. We provide an ablation study of this technique in Fig. 7 to show its effectiveness.

*Center Depth Loss:* Sometimes the optimization places the object away from the scene center. The object either becomes small or wander around the image boundary. For the few cases when this happens we apply a center depth loss

$$\mathcal{L}_{\text{center}}(\boldsymbol{D}) = -\log\left(\frac{1}{|\mathcal{B}|}\sum_{\mathbf{p} \in \mathcal{B}} \boldsymbol{D}(\mathbf{p}) - \frac{1}{|\mathcal{B}^{\complement}|}\sum_{\mathbf{q} \notin \mathcal{B}} \boldsymbol{D}(\mathbf{q})\right)$$
$$(27)$$

A DSLR photo of a yellow duck

A ficus planted in a pot

A zoomed out photo a small castle

A high quality photo of a toy motorcycle

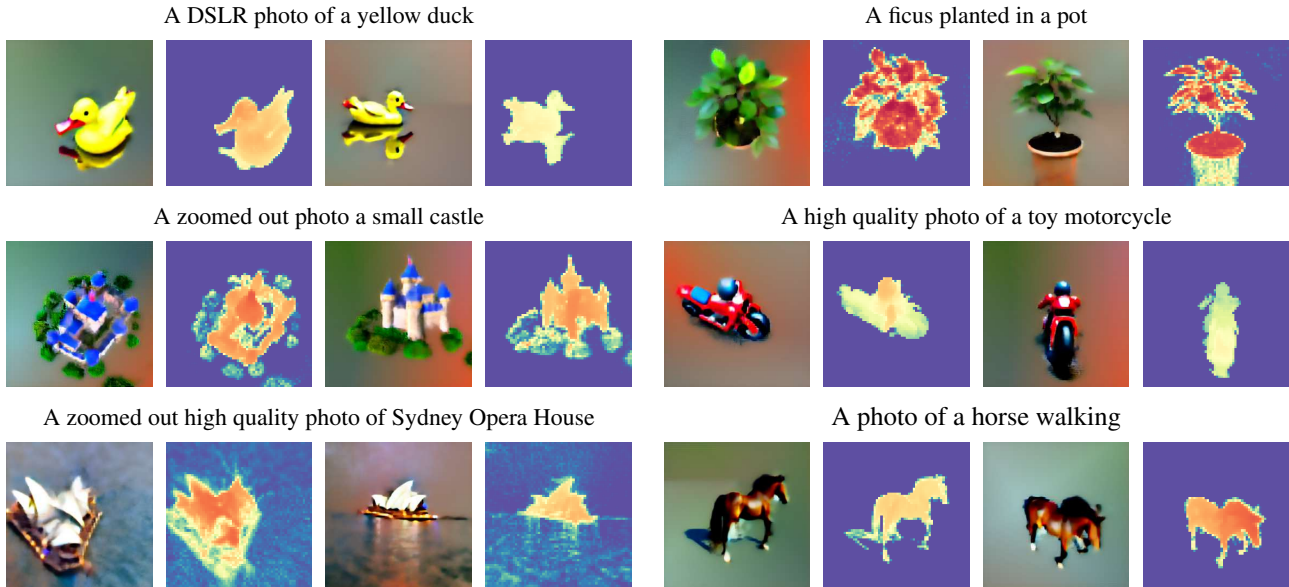A zoomed out high quality photo of Sydney Opera House

A photo of a horse walking

Figure 5. Qualitative results of text-prompted generation of 3D models with SJC, purely from the pretrained Stable Diffusion (2D) image model. Each row shows two views, with associated depth maps (blue is far, red is near), for a single 3D model generated for a given prompt. Note the detailed appearance as well as a sharp, well-defined depth structure.

where $D$ is the depth image, $\mathcal{B}$ is a box (set of pixel locations) at the center of the image, and $\mathcal{B}^{\complement}$ is its complement.

## 4.3. SJC vs. DreamFusion

In this section, we describe the differences and the connections between our SJC and DreamFusion.

**Differences from DreamFusion.** In terms of formulation, DreamFusion's computation of the gradient w.r.t. $\theta$ involves a U-Net Jacobian term (see Eq. 2 in their paper [41]). In practice, they "found that omitting the U-Net Jacobian term" to be more effective. On the other hand, this U-Net Jacobian term *does not appear* in our formulation. Their additional justification in the appendix actually leans more towards our viewpoint. An additional contribution of ours beyond Dream-Fusion [41] is our analysis of the effect that the OOD problem has when using a denoiser on rendered images (Claim 1), and the PAAS method to address it. For the variance reduction technique, namely the use of the Monte-Carlo estimate on Eq. (16), or $\hat{\epsilon} - \epsilon$ (in DreamFusion), vs. on Eq. (15), we observe comparable performance between the two methods empirically for 3D generation.

**Influences by DreamFusion.** At the time of this submission, DreamFusion is a concurrent arXiv paper. However, as we have read the paper, our research was influenced by their reported observations. In particular, we adopted the idea of randomized scheduling of $\sigma$ during 3D optimization for easier hyperparameter tuning, and used view-augmented language prompting that improves the overall 3D quality. For future work we do hope to explore a more general solution than view-dependent prompts.

## 5. Experiments

We conduct experiments on both unconditioned and conditioned diffusion models to have a more comprehensive understanding of the properties of SJC.

**DDPMs trained on FFHQ and LSUN Bedroom** are unconditioned diffusion models with an architecture based on the implementation by Dhariwal and Nichol [10]. They are trained on an image resolution of $256 \times 256$. FFHQ [21] is a dataset of aligned faces with diverse coverage of gender, age, race, facial appearance as well as head poses. LSUN Bedroom [62] includes bedroom images with varied furniture layout plans and rich interior design styles.

**Stable Diffusion** is an expanded work based on Latent Diffusion Model (LDM) developed by Rombach et al. [45]. It is trained on the LAION5B dataset [47]. We use the release version v1.5. Diffusion is performed on a latent space of $4 \times 64 \times 64$, then upsampled to $3 \times 256 \times 256$ by a decoder. The model is natively trained for text-conditioned image generations, and exposes a guidance scale parameter that controls the strength of language conditioning [14]. Intuitively a larger guidance scale makes the conditioned image distribution more faithful to the text prompt by trading off sample diversity.

### 5.1. Validating PAAS on 2D images.

Before directly jumping to 3D generation, we first verify that PAAS provides effective guidance on a simple 2D image canvas. In other words, here $\theta$ is a grid of RGB values and $f$ is an identity function. The hope is that gradient descent
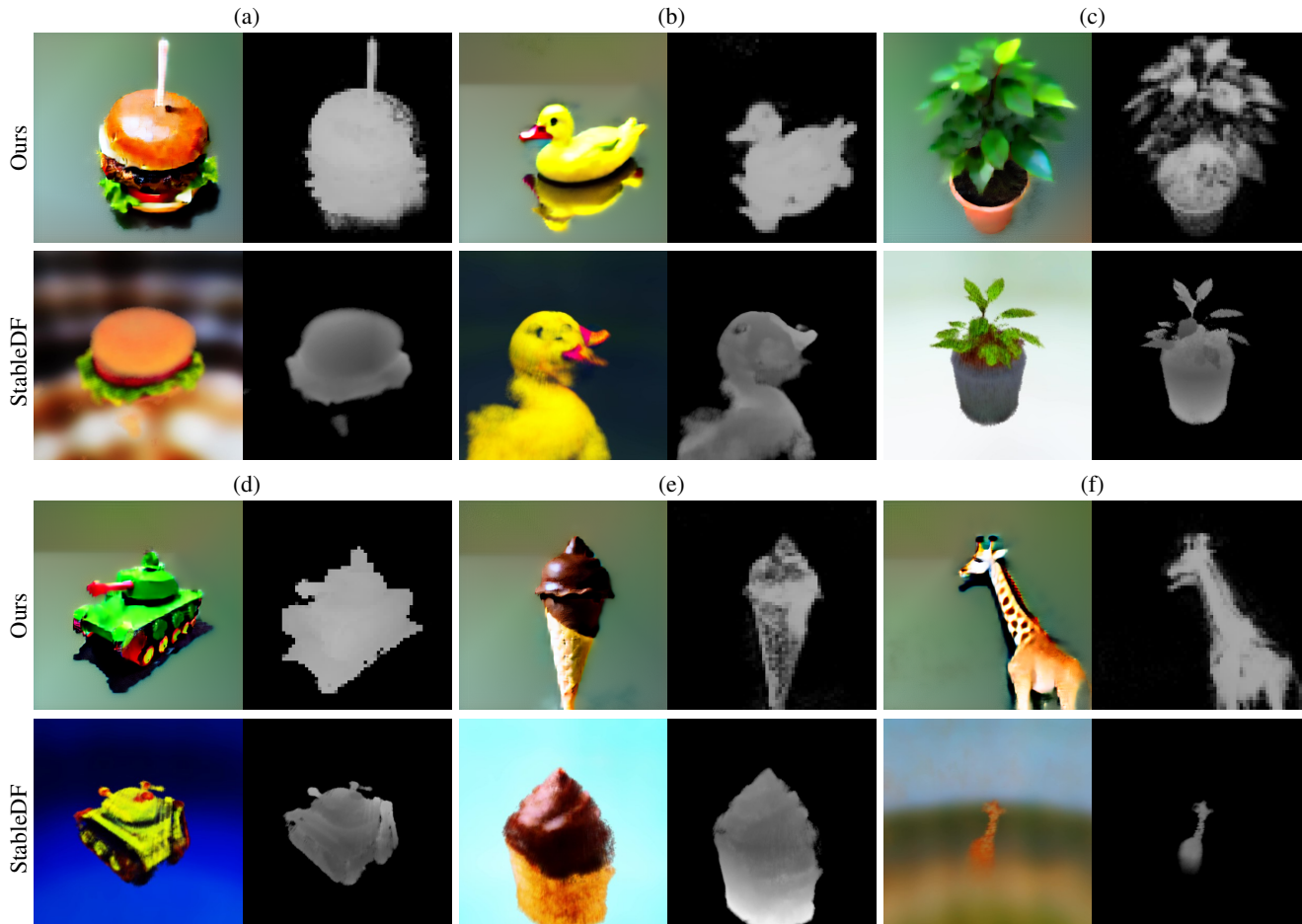
Figure 6. Qualitative comparison between Stable-DreamFusion (StableDF) and Ours. The prompts are: (a) "A high quality photo of a delicious burger"; (b) "a DSLR photo of a yellow duck"; (c) "A ficus planted in a pot"; (d) "A product photo of a toy tank"; (e) "A high quality photo of a chocolate icecream cone"; (f)"A wide angle zoomed out photo of a giraffe". Both methods are run for 10k iterations without per-prompt finetuning on the hyperparameters. The images on the left are rendered RGB images and the images on the right are depth visualization.

on the vector field produced by PAAS creates high quality images. Here an important decision to make is the schedule of $\{\sigma_i\}$ at which we compute PAAS.

We experimented with an annealed schedule (Annealed $\sigma$) vs. a random schedule (Random $\sigma$) as proposed in Dream-Fusion. Under the Annealed $\sigma$ schedule, we start from a large $\sigma$ and gradually decrease it as we update the image canvas $x$. PAAS computed at larger $\sigma$ level attends to high level image structure while smaller $\sigma$ provides stronger guidance on detailed features. The Random $\sigma$ schedule on the contrary uniformly samples a $\sigma$ at every step. We show qualitative comparisons in Fig. 4.

For unconditioned diffusion models trained on FFHQ, we observe that Annealed $\sigma$ performs better than Random $\sigma$, and the image samples have better pose variation and quality. Particularly, the randomized $\sigma$ exhibits severe mode-seeking behavior converging to average faces. In the case of LSUN Bedroom, the mode-seeking behavior results in a

blurry image canvas with no content.

On the other hand, natural language prompting plays a critical role when sampling images with Stable Diffusion. When the language guidance is set to a regular level of 3.0, the observations are broadly consistent with sampling on FFHQ and LSUN. Random $\sigma$ schedule produces blurry outputs. However, when the guidance scale is elevated to 10.0, Random $\sigma$ schedule begins to generate crisp, clean images and outperforms Annealed $\sigma$ schedule. Despite various sophisticated strategies on Annealed $\sigma$ scheduling (see our code for details), at a high language guidance scale Randomized $\sigma$ remains the better option. We hypothesize that stronger language guidance forces the image distribution to be narrower and more beneficial for a mode-seeking algorithm. We acknowledge none of the images in Fig. 4 can match the sample quality of a standard diffusion inference pipeline, and the right way to apply PAAS as gradient for optimization remains an open problem.
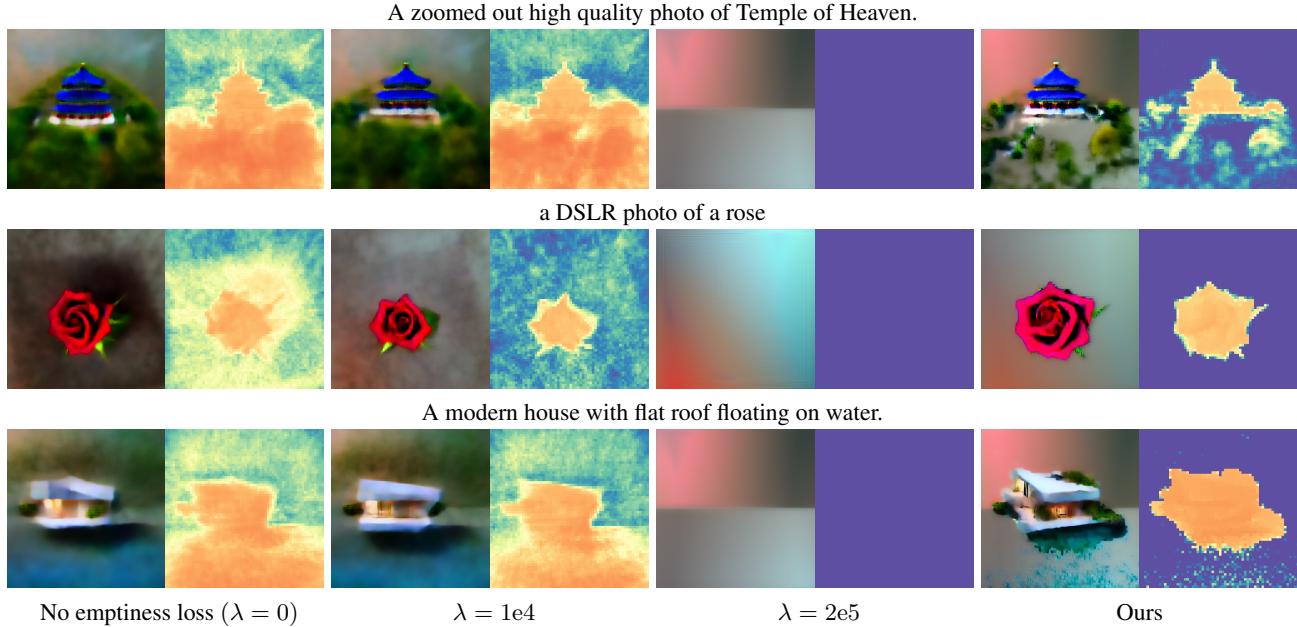
A zoomed out high quality photo of Temple of Heaven.

a DSLR photo of a rose

A modern house with flat roof floating on water.

| No emptiness loss ($\lambda = 0$) | $\lambda = 1e4$ | $\lambda = 2e5$ | Ours |

Figure 7. Ablation experiments on the proposed emptiness loss schedule. For each setting of the loss weight $\lambda$, we show a rendered image and the associated depth map from a randomly sampled viewpoint. Ours incorporates the loss with weight schedule described in 5.2. It leads to better 3D shape, as evidenced in the cleaner depth maps. Setting the loss weight too low yields "cloudy" depth fields. When setting the weight too high, SJC fails to produce meaningful 3D models.

## 5.2. 3D Generation

In this paper, we focus on 3D Generation with the language-conditioned Stable Diffusion model. We found that tuning the Annealed $\sigma$ schedule on FFHQ and LSUN Bedroom in 3D domain is difficult in practice, and leave it as future work. Based on the insights from 2D experiments earlier, we use Random $\sigma$ schedule coupled with a high language guidance scale.

**Rendering with Latent 3D Features.** Stable Diffusion economizes compute by performing diffusion modeling on the latent features of a pretrained AutoEncoder. We therefore choose to render a feature image in this latent space from a features field [3, 38] represented by a voxel grid in $\mathbb{R}^{4 \times N_x \times N_y \times N_z}$.

**Qualitative Comparison.** In Fig. 5, we show text-prompted 3D generation results from SJC, and compare SJC with Stable-DreamFusion, a third-party implementation based on the same pretrained Stable Diffusion v1.5 model. In Fig. 6, we show qualitative comparisons of generated 3D assets given the same prompt. We observe that SJC generates 3D models with better image quality and more sensible structure than Stable-DreamFusion in a significant number of cases. We note that both systems exhibit quality fluctutations over different trials, and the point of this comparison is to show that our overall pipeline is competitive.

**Ablations.** In Fig. 7, we conduct ablations to demonstrate the importance of the proposed emptiness loss and scheduling of its weight $\lambda$ discussed in Sec. 4.2. We show results without the emptiness loss, with constant weight $\lambda$ vs our proposed scheduling of $\lambda$. We observe that our complete method (Ours) improves the quality of generated 3D models, e.g., fewer floating artifacts and better geometry.

## 6. Conclusion

We propose an optimization-based approach to generate 3D assets from pretrained image (2D) diffusion models. The key technical contribution is the derivation of Perturb-and-Average Scoring method which bridges the gap between the denoising-trained diffusion models and the non-noisy images encountered in the process of optimizing a 3D model guided by the diffusion. Working with the large-scale Stable Diffusion model, we demonstrate that our approach can generate compelling 3D models, comparing favorably to available concurrent work. Finally, we investigate an interesting distinction between the effect of noise scheduling regime in unconditional image diffusion models and a text-conditional model, and identify an avenue for future work.

# References

[1] Dmitry Baranchuk, Andrey Voynov, Ivan Rubachev, Valentin Khrulkov, and Artem Babenko. Label-efficient semantic segmentation with diffusion models. In *International Conference on Learning Representations*, 2022. 4

[2] Ruojin Cai, Guandao Yang, Hadar Averbuch-Elor, Zekun Hao, Serge Belongie, Noah Snavely, and Bharath Hariharan. Learning gradient fields for shape generation. In *Eur. Conf. Comput. Vis.*, 2020. 2

[3] Eric R Chan, Connor Z Lin, Matthew A Chan, Koki Nagano, Boxiao Pan, Shalini De Mello, Orazio Gallo, Leonidas J Guibas, Jonathan Tremblay, Sameh Khamis, et al. Efficient geometry-aware 3D generative adversarial networks. In *IEEE Conf. Comput. Vis. Pattern Recog.*, 2022. 2, 8

[4] Eric R Chan, Marco Monteiro, Petr Kellnhofer, Jiajun Wu, and Gordon Wetzstein. pi-GAN: Periodic implicit generative adversarial networks for 3d-aware image synthesis. In *IEEE Conf. Comput. Vis. Pattern Recog.*, 2021. 2

[5] Angel X. Chang, Thomas Funkhouser, Leonidas Guibas, Pat Hanrahan, Qixing Huang, Zimo Li, Silvio Savarese, Manolis Savva, Shuran Song, Hao Su, Jianxiong Xiao, Li Yi, and Fisher Yu. ShapeNet: An Information-Rich 3D Model Repository. Technical Report arXiv:1512.03012 [cs.GR], Stanford University — Princeton University — Toyota Technological Institute at Chicago, 2015. 2

[6] Anpei Chen, Zexiang Xu, Andreas Geiger, Jingyi Yu, and Hao Su. Tensorf: Tensorial radiance fields. In *Eur. Conf. Comput. Vis.*, pages 333–350, 2022. 2, 5

[7] Yizong Cheng. Mean shift, mode seeking, and clustering. 1995. 3

[8] Dorin Comaniciu and Peter Meer. Mean shift analysis and applications. In *Int. Conf. Comput. Vis.*, 1999. 3

[9] Amélie Deltombe. How much does it cost to create 3D models?, Apr 2022. 2

[10] Prafulla Dhariwal and Alexander Nichol. Diffusion models beat gans on image synthesis. *Advances in Neural Information Processing Systems*, 34:8780–8794, 2021. 2, 6

[11] Sara Fridovich-Keil, Alex Yu, Matthew Tancik, Qinhong Chen, Benjamin Recht, and Angjoo Kanazawa. Plenoxels: Radiance fields without neural networks. In *IEEE Conf. Comput. Vis. Pattern Recog.*, pages 5501–5510, 2022. 2, 5

[12] Alexandros Graikos, Nikolay Malkin, Nebojsa Jojic, and Dimitris Samaras. Diffusion models as plug-and-play priors. In *Advances in Neural Information Processing Systems*, volume 35, pages 14715–14728, 2022. 3

[13] Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. In *Adv. Neural Inform. Process. Syst.*, 2020. 1, 2, 3

[14] Jonathan Ho and Tim Salimans. Classifier-free diffusion guidance. In *NeurIPS 2021 Workshop on Deep Generative Models and Downstream Applications*, 2021. 6

[15] Fangzhou Hong, Mingyuan Zhang, Liang Pan, Zhongang Cai, Lei Yang, and Ziwei Liu. Avatarclip: Zero-shot text-driven generation and animation of 3d avatars. *ACM Transactions on Graphics (TOG)*, 41(4):1–19, 2022. 2

[16] Aapo Hyvärinen. Estimation of non-normalized statistical models by score matching. *J. Mach. Learn. Res*, 2005. 1, 3

[17] Ajay Jain, Ben Mildenhall, Jonathan T Barron, Pieter Abbeel, and Ben Poole. Zero-shot text-guided object generation with dream fields. In *IEEE Conf. Comput. Vis. Pattern Recog.*, 2022. 2

[18] Nikolay Jetchev. Clipmatrix: Text-controlled creation of 3D textured meshes. *arXiv preprint arXiv:2109.12922*, 2021. 2

[19] Animesh Karnewar, Tobias Ritschel, Oliver Wang, and Niloy Mitra. Relu fields: The little non-linearity that could. In *ACM SIGGRAPH 2022 Conference Proceedings*, 2022. 2

[20] Tero Karras, Miika Aittala, Timo Aila, and Samuli Laine. Elucidating the design space of diffusion-based generative models. In *Adv. Neural Inform. Process. Syst.*, 2022. 1, 2, 3

[21] Tero Karras, Samuli Laine, and Timo Aila. A style-based generator architecture for generative adversarial networks. In *IEEE Conf. Comput. Vis. Pattern Recog.*, 2019. 4, 6

[22] Nasir Khalid, Tianhao Xie, Eugene Belilovsky, and Tiberiu Popa. Clip-mesh: Generating textured meshes from text using pretrained image-text models. *ACM Trans. Graph.*, 2022. 2

[23] Diederik Kingma, Tim Salimans, Ben Poole, and Jonathan Ho. Variational diffusion models. *Advances in neural information processing systems*, 34:21696–21707, 2021. 1

[24] Sebastian Koch, Albert Matveev, Zhongshi Jiang, Francis Williams, Alexey Artemov, Evgeny Burnaev, Marc Alexa, Denis Zorin, and Daniele Panozzo. ABC: A big cad model dataset for geometric deep learning. In *IEEE Conf. Comput. Vis. Pattern Recog.*, 2019. 2

[25] Christoph Lassner and Michael Zollhofer. Pulsar: Efficient sphere-based neural rendering. In *IEEE Conf. Comput. Vis. Pattern Recog.*, 2021. 2

[26] Han-Hung Lee and Angel X Chang. Understanding pure clip guidance for voxel grid nerf models. *arXiv preprint arXiv:2209.15172*, 2022. 2

[27] Tzu-Mao Li, Miika Aittala, Frédo Durand, and Jaakko Lehtinen. Differentiable Monte Carlo ray tracing through edge sampling. *ACM Trans. Graph.*, 2018. 2

[28] Lingjie Liu, Jiatao Gu, Kyaw Zaw Lin, Tat-Seng Chua, and Christian Theobalt. Neural sparse voxel fields. In *Adv. Neural Inform. Process. Syst.*, 2020. 2

[29] Xingchao Liu, Chengyue Gong, Lemeng Wu, Shujian Zhang, Hao Su, and Qiang Liu. Fusedream: Training-free text-to-image generation with improved clip+ GAN space optimization. *arXiv preprint arXiv:2112.01573*, 2021. 2

[30] Matthew M Loper and Michael J Black. OpenDR: An approximate differentiable renderer. In *Eur. Conf. Comput. Vis.*, 2014. 2

[31] Ricardo Martin-Brualla, Noha Radwan, Mehdi SM Sajjadi, Jonathan T Barron, Alexey Dosovitskiy, and Daniel Duckworth. Nerf in the wild: Neural radiance fields for unconstrained photo collections. In *IEEE Conf. Comput. Vis. Pattern Recog.*, 2021. 2

[32] Nelson Max. Optical models for direct volume rendering. *IEEE Trans. Vis. Comput. Graph.*, 1995. 2, 5

[33] Oscar Michel, Roi Bar-On, Richard Liu, Sagie Benaim, and Rana Hanocka. Text2mesh: Text-driven neural stylization for meshes. In *IEEE Conf. Comput. Vis. Pattern Recog.*, 2022. 2

[34] 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 *Eur. Conf. Comput. Vis.*, pages 405–421, 2020. 2, 5

[35] Thu Nguyen-Phuoc, Chuan Li, Lucas Theis, Christian Richardt, and Yong-Liang Yang. HoloGAN: Unsupervised

learning of 3D representations from natural images. In *Int. Conf. Comput. Vis.*, 2019. 2

[36] Alexander Quinn Nichol, Prafulla Dhariwal, Aditya Ramesh, Pranav Shyam, Pamela Mishkin, Bob Mcgrew, Ilya Sutskever, and Mark Chen. Glide: Towards photorealistic image generation and editing with text-guided diffusion models. In *International Conference on Machine Learning*, pages 16784–16804, 2022. 2

[37] Michael Niemeyer and Andreas Geiger. Campari: Camera-aware decomposed generative neural radiance fields. In *Int. Conf. 3DV*, 2021. 2

[38] Michael Niemeyer and Andreas Geiger. Giraffe: Representing scenes as compositional generative neural feature fields. In *IEEE Conf. Comput. Vis. Pattern Recog.*, 2021. 2, 8

[39] Merlin Nimier-David, Delio Vicini, Tizian Zeltner, and Wenzel Jakob. Mitsuba 2: A retargetable forward and inverse renderer. *ACM Trans. Graph.*, 2019. 2

[40] Michael Oechsle, Songyou Peng, and Andreas Geiger. Unisurf: Unifying neural implicit surfaces and radiance fields for multi-view reconstruction. In *Int. Conf. Comput. Vis.*, 2021. 2

[41] Ben Poole, Ajay Jain, Jonathan T. Barron, and Ben Mildenhall. Dreamfusion: Text-to-3d using 2d diffusion. In *Int. Conf. Learn. Represent.*, 2023. 3, 6

[42] 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*, 2021. 2

[43] Sai Rajeswar, Fahim Mannan, Florian Golemo, David Vazquez, Derek Nowrouzezahrai, and Aaron Courville. Pix2scene: Learning implicit 3d representations from images. *openreview*, 2018. 2

[44] Aditya Ramesh, Prafulla Dhariwal, Alex Nichol, Casey Chu, and Mark Chen. Hierarchical text-conditional image generation with clip latents. *arXiv preprint arXiv:2204.06125*, 2022. 2

[45] Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-resolution image synthesis with latent diffusion models. In *IEEE Conf. Comput. Vis. Pattern Recog.*, 2022. 2, 3, 6

[46] Chitwan Saharia, William Chan, Saurabh Saxena, Lala Li, Jay Whang, Emily L Denton, Kamyar Ghasemipour, Raphael Gontijo Lopes, Burcu Karagol Ayan, Tim Salimans, Jonathan Ho, David J Fleet, and Mohammad Norouzi. Photorealistic text-to-image diffusion models with deep language understanding. In *Adv. Neural Inform. Process. Syst.*, volume 35, pages 36479–36494, 2022. 3

[47] Christoph Schuhmann, Romain Beaumont, Richard Vencu, Cade W Gordon, Ross Wightman, Mehdi Cherti, Theo Coombes, Aarush Katta, Clayton Mullis, Mitchell Wortsman, Patrick Schramowski, Srivatsa R Kundurthy, Katherine Crowson, Ludwig Schmidt, Robert Kaczmarczyk, and Jenia Jitsev. LAION-5b: An open large-scale dataset for training next generation image-text models. In *Adv. Neural Inform. Process. Syst.*, 2022. 2, 6

[48] Katja Schwarz, Yiyi Liao, Michael Niemeyer, and Andreas Geiger. GRAF: Generative radiance fields for 3D-aware image synthesis. In *Adv. Neural Inform. Process. Syst.*, 2020.

[49] Jascha Sohl-Dickstein, Eric Weiss, Niru Maheswaranathan, and Surya Ganguli. Deep unsupervised learning using nonequilibrium thermodynamics. In *International Conference on Machine Learning*, pages 2256–2265, 2015. 1, 2

[50] Yang Song and Stefano Ermon. Generative modeling by estimating gradients of the data distribution. In *Adv. Neural Inform. Process. Syst.*, 2019. 1, 2, 3

[51] Yang Song and Stefano Ermon. Improved techniques for training score-based generative models. In *Adv. Neural Inform. Process. Syst.*, 2020. 3

[52] Yang Song, Jascha Sohl-Dickstein, Diederik P Kingma, Abhishek Kumar, Stefano Ermon, and Ben Poole. Score-based generative modeling through stochastic differential equations. In *Int. Conf. Learn. Represent.*, 2021. 1, 2, 3

[53] Cheng Sun, Min Sun, and Hwann-Tzong Chen. Direct voxel grid optimization: Super-fast convergence for radiance fields reconstruction. In *IEEE Conf. Comput. Vis. Pattern Recog.*, 2022. 2, 5

[54] Xingyuan Sun, Jiajun Wu, Xiuming Zhang, Zhoutong Zhang, Chengkai Zhang, Tianfan Xue, Joshua B Tenenbaum, and William T Freeman. Pix3D: Dataset and methods for single-image 3D shape modeling. In *IEEE Conf. Comput. Vis. Pattern Recog.*, 2018. 2

[55] Jiaxiang Tang. Stable-dreamfusion: Text-to-3d with stable-diffusion, 2022. https://github.com/ashawkey/stable-dreamfusion. 3

[56] Pascal Vincent. A connection between score matching and denoising autoencoders. *Neural computation*, 2011. 2

[57] 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 *Adv. Neural Inform. Process. Syst.*, 2021. 2

[58] Jiajun Wu, Chengkai Zhang, Tianfan Xue, Bill Freeman, and Josh Tenenbaum. Learning a probabilistic latent space of object shapes via 3D generative-adversarial modeling. In *Adv. Neural Inform. Process. Syst.*, 2016. 2

[59] Zhirong Wu, Shuran Song, Aditya Khosla, Fisher Yu, Linguang Zhang, Xiaoou Tang, and Jianxiong Xiao. 3D ShapeNets: A deep representation for volumetric shapes. In *IEEE Conf. Comput. Vis. Pattern Recog.*, 2015. 2

[60] Guandao Yang, Xun Huang, Zekun Hao, Ming-Yu Liu, Serge Belongie, and Bharath Hariharan. PointFlow: 3D point cloud generation with continuous normalizing flows. In *Int. Conf. Comput. Vis.*, 2019. 2

[61] Lior Yariv, Jiatao Gu, Yoni Kasten, and Yaron Lipman. Volume rendering of neural implicit surfaces. In *Adv. Neural Inform. Process. Syst.*, 2021. 2

[62] Fisher Yu, Ari Seff, Yinda Zhang, Shuran Song, Thomas Funkhouser, and Jianxiong Xiao. LSUN: Construction of a large-scale image dataset using deep learning with humans in the loop. *arXiv preprint arXiv:1506.03365*, 2015. 6

[63] Yuxuan Zhang, Wenzheng Chen, Huan Ling, Jun Gao, Yinan Zhang, Antonio Torralba, and Sanja Fidler. Image GANs meet differentiable rendering for inverse graphics and interpretable 3d neural rendering. In *Int. Conf. Learn. Represent.*, 2021. 2

[64] Xiaoming Zhao, Fangchang Ma, David Güera, Zhile Ren, Alexander G Schwing, and Alex Colburn. Generative multi-plane images: Making a 2D GAN 3D-Aware. In *Eur. Conf.*