DreamDiffusion

A Preprint

1 Main

3D gen models can be trained on voxels or point clouds, but data is relatively scarce compared to plentiful 2D images.

2 Introduction

Text-conditioned diffusion models achieve 2D synthesis by learning to reverse a noising process, but they do not directly produce 3D geometry. DreamFusion overcomes this by using a frozen 2D diffusion network as a loss to sculpt a Neural Radiance Field (NeRF) to match a text prompt, without any paired 3D data.

3 Diffusion Models and Score Distillation Sampling

3.1 2D Text-to-Image Diffusion

A diffusion model defines a forward process

$$q(z_t \mid x) = \mathcal{N}\big(\alpha_t x,\ \sigma_t^2 I\big),$$

and learns a denoiser ϵ_ϕ via

$$\mathcal{L}_{\text{Diff}}(\phi, x) = \mathbb{E}_{t,\epsilon}\Big[w(t)\,\|\epsilon_\phi(\alpha_t x + \sigma_t \epsilon;\, t, y) - \epsilon\|^2\Big]. \tag{1}$$

3.2 Score Distillation Sampling

We wish to optimize generator parameters θ so that $x = g(\theta)$ lies in a high-likelihood region of the diffusion model. Define the SDS loss gradient as

$$\nabla_\theta \mathcal{L}_{\text{SDS}} = \mathbb{E}_{t,\epsilon}\Big[w(t)\,\big(\hat{\epsilon}_\phi(z_t; y, t) - \epsilon\big)\,\frac{\partial g(\theta)}{\partial \theta}\Big], \quad z_t = \alpha_t\, g(\theta) + \sigma_t\, \epsilon, \tag{2}$$

omitting the expensive U-Net Jacobian and absorbing constant factors into $w(t)$.

3.3 Sampling in Parameter Space

Existing diffusion sampling operates in pixel space, matching the data the model was trained on. Instead, we parameterize images via a differentiable generator $x = g(\theta)$ (e.g. a NeRF) and seek

$$\theta^* = \arg\min_\theta \mathcal{L}_{\text{Diff}}\big(\phi,\ g(\theta)\big),$$

but found this unstable in practice. By dropping the U-Net Jacobian term, we obtain the practical update of Eq. (2), which follows the learned score function to guide $g(\theta)$ toward regions the diffusion model deems plausible.

4 Neural Rendering of a 3D Model

A NeRF MLP maps each point $\mu \in \mathbb{R}^3$ to density τ and color ρ:

$$(\tau, \rho) = \mathrm{MLP}(\mu; \theta).$$

Colors accumulate along rays via

$$C = \sum_i w_i\, c_i, \quad w_i = \alpha_i \prod_{j<i}(1-\alpha_j), \quad \alpha_i = 1 - \exp(-\tau_i\, \delta_i),$$

and each sample's shaded color is

$$c_i = \rho_i \circ \big(\ell_\rho\ \max(0,\, n_i \cdot \tfrac{\ell-\mu_i}{\|\ell-\mu_i\|}) + \ell_a\big),$$

with normals $n_i = -\nabla_\mu \tau_i / \|\nabla_\mu \tau_i\|$.

5 Text-to-3D Optimization

1. Initialize $\theta \sim \mathcal{N}(0, I)$.
2. Repeat for $k = 1, \ldots, N$:
   (a) Sample a random camera and lighting.
   (b) Render $x = g(\theta)$ at low resolution.
   (c) Draw $t \sim U[0,1],\ \epsilon \sim \mathcal{N}(0, I)$.
   (d) Compute $z_t = \alpha_t x + \sigma_t \epsilon$.
   (e) Query diffusion model: $\hat{\epsilon} = \epsilon_\phi(z_t; y, t)$.
   (f) Compute gradient $\nabla_\theta \mathcal{L}_{\mathrm{SDS}}$ via Eq. (2).
   (g) Update $\theta \leftarrow \theta - \eta\, \nabla_\theta \mathcal{L}_{\mathrm{SDS}}$.

6 Algorithm (Pseudocode)

Algorithm 1 DreamFusion Optimization Loop

Input: text prompt y, frozen diffusion ϵ_ϕ, renderer $g(\theta)$ Initialize $\theta \sim \mathcal{N}(0, I)$ $k = 1$ N Sample camera pose, light source $x \leftarrow g(\theta)$ $t \sim U(0,1),\ \epsilon \sim \mathcal{N}(0, I)$ $z_t \leftarrow \alpha_t\, x + \sigma_t\, \epsilon$ $\hat{\epsilon} \leftarrow \epsilon_\phi(z_t;\, y, t)$ $g_\theta \leftarrow w(t)\, (\hat{\epsilon} - \epsilon)\, \nabla_\theta x$ $\theta \leftarrow \theta - \eta\, g_\theta$ return θ

A Appendix A.4: Equivalence to Density Distillation

Here we show that the SDS gradient arises from a weighted KL divergence between the true noising distribution and the diffusion model's predicted distribution. Define

$$\mathcal{L}_{\mathrm{SDS}}(\phi, \theta) \triangleq \mathbb{E}_t\big[\tfrac{\sigma_t}{\alpha_t} w(t)\, \mathrm{KL}\big(q(z_t \mid g(\theta); y, t) \,\|\, p_\phi(z_t; y, t)\big)\big].$$

Then one can verify

$$\nabla_\theta \mathcal{L}_{\mathrm{SDS}} = \nabla_\theta \mathbb{E}_{t,\epsilon}\Big[\tfrac{\sigma_t}{\alpha_t} w(t)\, \big\langle \nabla_{z_t} \log q(z_t \mid x),\, \tfrac{\partial g(\theta)}{\partial \theta} \big\rangle\Big],$$

and using the score-matching identity $\nabla_{z_t} \log q(z_t \mid x) = -(\hat{\epsilon}_\phi(z_t; y, t) - \epsilon)/\sigma_t$, this yields exactly the practical gradient of Eq. (2) up to constant factors.