DreamFusion: Text-To-3D Using 2D Diffusion

Monday, November 14, 2022 10:29 PM

- This note is written as I read the paper "DreamFusion: Text-To-3D Using 2D Diffusion" by Poole et al.
- Paper link: https://arxiv.org/abs/2209.14988.

- What the paper achieves, I think, is quite remarkable.
  - It allows one to generated 3D models, represented as NeRFs, given a text prompt.
  - It achieves this by using a pretrained denoising diffusion probabilistic model (DDPM).
  - So, there's no need to prepare any 3D dataset to create 3D models.
- There are many similar previous works (which I should read later…)
  - Those that use NeRF-like 3D representations as parts of a generative models.
    - **GRAF: Generative Radiance Fields for 3D-Aware Image Synthesis**
      - NeurIPS 2020
      - https://arxiv.org/abs/2007.02442
    - **pi-GAN: Periodic Implicit Generative Adversarial Networks for 3D-Aware Image Synthesis**
      - CVPR 2021
      - https://arxiv.org/abs/2012.00926
    - **StyleNeRF: A Style-based 3D-Aware Generator for High-resolution Image Synthesis**
      - ICLR 2022
      - https://arxiv.org/abs/2110.08985
  - Those that try to generate 3D models from text.
    - **Towards implicit text-guided 3d shape generation**
      - CVPR 2022
      - https://arxiv.org/abs/2203.14622
    - **Zero-Shot Text-Guided Object Generation with Dream Fields**
      - CVPR 2022
      - https://arxiv.org/abs/2112.01455
    - **CLIP-Forge: Towards zero-shot text-to-shape generation**
      - CVPR 2022
      - https://arxiv.org/abs/2110.02624
    - **ClipMatrix: Text-controlled creation of 3d textured meshes**
      - CVPR 2022
      - https://arxiv.org/abs/2110.02624
    - **CLIP-NeRF: Text-and-Image Driven Manipulation of Neural Radiance Fields**
      - CVPR 2022
      - https://arxiv.org/abs/2112.05139
- The paper's approach.
  - Optimize a NeRF using a generative model as part of the loss function.
  - Same approach as the Dream Fields paper
    - Uses CLIP to guide training of a NeRF.
  - Unlike Dream Fields, it replaces CLIP with a text-conditioned DDPM.
  - The paper also proposes an optimization trick called **score distillation sampling (SDS)**.
    - It is based on "probability density distillation," which will be explained later.
  - DreamFusion = SDS + NeRF

Background on diffusion models

- The paper uses notations similar to the "variational diffusion models" paper by Kingma et al (2021).
  ↳ There are differences, however.

- A data item from data distribution is denoted by x.

- A time step is denoted by t. It goes from 0 to 1.

- A degraded sample (a latent) is denoted by z_t.

- The forward process is denoted by q. We have that

$$q(z_t | x) = N(z_t ; \alpha_t x, \sigma_t^2 I)$$

where $\alpha_t^2 = 1 - \sigma_t^2$. The standard deviation must be such that

① $\sigma_0 \approx 0$ (beginning)
② $\sigma_1 \approx 1$ (end)
③ σ_t is monotonically increasing.

- The backward process is denoted by p_ϕ.

  ⇒ It starts with $p_\phi(z_1) = N(z_1; 0, I)$

  ⇒ Transition from z_t to $z_{t-\Delta t}$ is given by

$$p(z_{t-\Delta t} | z_t) = q(z_{t-\Delta t} ; z_t, x = x_\phi(z_t, t))$$

  where $x_\phi(z_t, t)$ is a denoising model that predicts x from z_t.

- Following Ho et al.'s 2020 paper, we train a noise prediction model $\xi_\phi(z_t, t)$ that predicts noise $\xi \sim N(0, I)$ that is used to create z_t from x

according to $z_t = \alpha_t x + \sigma_t \xi$.

- When ξ_ϕ is trained well enough, it approximates the score of z_t:

$$\xi_\phi(z_t, t) \approx -\sigma_t \nabla \log q(z_t)$$

- Let $\hat{s}_\phi(z_t, t)$ be the score prediction model. That is,

$$\hat{s}_\phi(z_t, t) = \frac{\hat{\xi}_\phi(z_t, t)}{\sigma_t}.$$

- $\hat{\xi}_\phi(z_t, t)$ is trained with a (weighted) evidence lower bound, which simplifies to:

$$\mathcal{L}_{Diff}(\phi, x)$$
$$= E_{t \sim \underbrace{U(0,1)}_{\text{uniform distribution}}, \xi \sim N(0,1)} \left[w(t) \| \xi - \hat{\xi}_\phi(\overbrace{z_t}^{\alpha_t x + \sigma_t \xi}, t) \|^2 \right]$$

- Let $p_\phi(z_t, t)$ denote the approximate marginal distribution of z_t whose score funtion is $s_\phi(z_t, t)$.

- The paper relies on a version of DDPM that is conditioned on text embedding y.
  - ↳ This means that we have $\xi_\phi(z_t, t, y)$ in addition to $\xi_\phi(z_t, t)$

- When generating a sample, we use classifier guidance to compute the noise

$$\hat{\xi}_\phi(z_t, t, y) = (1+w)\,\xi_\phi(z_t, t, w) - w\,\xi_\phi(z_t, t)$$

where $w \geq 0$ is a hyperparameter.

- Let $\hat{p}_\phi(z_t, t, y)$ denotes the approximate marginal whose score function is $\hat{s}_\phi(z_t, t, y) = \dfrac{-\hat{\xi}_\phi(z_t, t, y)}{\sigma_t}$.

Sampling in parameter space

- A differentiable image parameterization (DIP) is a representation of image x by a parameter vector θ subjected to the constraint that $x = g(\theta)$ where g is a differentiable function.

  called a "generator"

  ↳ NeRF is a special case of this.

- With a pretrained DDPM, we want to sample images in term of parameters θ instead of the raw image x.

  ↳ We want to do this because, when g is a NeRF rendering function, we have that θ is the parameters of a NeRF, which means we get a representation of a 3D object instead of just a picture.

- A standard approach to get θ is to optimize $\mathcal{L}_{\text{Diff}}(\phi, x = g(\theta))$ with respect to θ while holding ϕ

$\mathcal{L}_{Diff}(\phi, x = g(\theta))$ with respect to θ while holding ϕ fixed.

- The authors tried the above approach, but found that it did not produce realistic examples.
  - ↳ A paper by Graikos et al. showed that this approach can be made to work.
  - ↳ However, one needs to customize the timestep schedule and the optimizer to make it work.

- The paper explains why the above approach does not work. Consider the gradient of $\mathcal{L}_{Diff}$.

$$\nabla_\theta \mathcal{L}_{Diff}(\phi, x = g(\theta))$$
$$= E_{t \sim U(0,1), \xi \sim N(0,I)} \left[2w(t) (\hat{\xi}_\phi(z_t,t,y) - \xi) \frac{\partial \hat{\xi}_\phi(z_t,t,y)}{\partial z_t} \frac{\partial z_t}{\partial \theta} \right]$$

because $z_t = \alpha_t x(\theta) + \sigma_t \xi$, we have $\frac{\partial z_t}{\partial \theta} = \alpha_t \frac{\partial x}{\partial \theta}$

so,

$$\nabla_\theta \mathcal{L}_{Diff}(\phi, x = g(\theta))$$
$$= E_{t \sim U(0,1), \xi \sim N(0,I)} \left[2\alpha_t w(t) \underbrace{(\hat{\xi}_\phi(z_t,t,y) - \xi)}_{\text{noise residual}} \underbrace{\frac{\partial \hat{\xi}_\phi}{\partial z_t}}_{\text{noise model Jacobian}} \overbrace{\frac{\partial x}{\partial \theta}}^{\text{generator Jacobian}} \right]$$

The noise model Jacobian is expensive to compute and is poorly conditioned for small noise levels. (Note: only offers a simple explanation and not evidences.)

only offers a simple explanation and not evidences.)

- The paper then say that dropping the noise model gradient term altogether leads to better gradient.

$$\nabla_\theta \mathcal{L}_{\underbrace{SDS}}(\phi, x = g(\theta))$$

$$= E_{t \sim U(0,1), \xi \sim N(0,I)} \left[2\alpha_t w(t) \left(\hat{\xi}_\phi(z_t, t, y) - \xi \right) \frac{\partial x}{\partial t} \right]$$

"score distillation sampling"

Probability density distillation

- The score distillation sampling is inspired by a technique called "probability density distillation."

- This comes from the paper "Parallel WaveNet: Fast High-Fidelity Speech Synthesis" by van den Oord et al.
  - ↳ Paper link: https://arxiv.org/pdf/1711.10433.pdf

- WaveNet is an autoregressive model for speech synthesis.
  - ⇒ $x \in \mathbb{R}^d$ is a one-dimensional speech signal. $x = (x_1, x_2, \ldots, x_d)$.
  - ⇒ The paper model the probability of a sample x by

$$p(x) = p(x_1 | \theta) \prod_{t=2}^{d} p(x_t | x_{t-1}, \theta)$$

where θ is the model parameter.

  - ⇒ So, when generating a sample, you need to

⇒ So, when generating a sample, you need to generate the component of x sequentially from x_1, then x_2, then x_3, and so on.

⇒ As a result, generating a sample of length d takes $O(d)$ deep network evaluation.

- The paper aims to come up with a much faster sampling algorithm.

- Its solution is to distill WaveNet into another model called Inverse-Autoregressive Flows (IAF), which is the student model.

- IAF is a normalizing flow model. It models the distribution of x, $p_X(x)$, with the following process:

① Sample a latent variable $z \in \mathbb{R}^d$ with a fixed distribution $p_Z(z)$.

↳ The paper uses the multi-dimensional logistic distribution.

https://en.wikipedia.org/wiki/Logistic_distribution

↳ $p_Z(z) = \mathbb{L}(0, I)$

② Apply a deterministic transformation to z:

$$x_t = z_t \cdot s(z_{<t}, \theta) + \mu(z_{<t}, \theta)$$

where $\underbrace{z_{<t} = (z_1, z_2, \dots, z_{t-1})}_{} \rightarrow f(z_{\le t})$

- It follows that

$$\log p_X(x) = \log p_Z(z) - \log \left| \frac{\partial x}{\partial z} \right|$$

$$= \log p_Z(z) - \underbrace{\sum_t \log \frac{f(z_{\le t})}{\partial z_t}}_{\text{because Jacobian is lower triangular}}$$

- Moreover, we have that

$$p(x_t \mid z_{<t}, \theta) = \mathbb{L}(x_t ; \mu(z_{<t}, \theta), s(z_{<t}, \theta)).$$

- Given an IAF student $p_S(x)$ and a WaveNet teacher $p_T(x)$, we seek to minimize the KL divergence

$$D_{KL}(p_S \| p_T) = \underbrace{H(p_S, p_T)}_{\text{cross entropy}} - \underbrace{H(p_S)}_{\text{entropy of } p_S}$$

between the two.

- The two terms can be written as follows:

$$H(p_s) = E_{z \sim \mathbb{L}(0,I)} \left[-\log p_S(x(z)) \right]$$

$$= E_{z \sim \mathbb{L}(0,I)} \left[-\log p_Z(z) + \sum_t \log \frac{\partial f(z_{<t})}{\partial z_t} \right]$$

$$= E_{z \sim \mathbb{L}(0,I)} \left[\sum_t \log s(z_{<t}, \theta) \right] + \underbrace{2d.}_{\rightarrow H(\mathbb{L}(0,I))}$$

$$H(p_s, p_T) = \sum^{d} E \left[H(p_S(x_t \mid x_{<t}), p_T(x_t \mid x_{<t})) \right]$$

$$H(p_S, p_T) = \underbrace{\sum_{t=1}^{d} E_{x_{<t} \sim p_S(x_{<t})} \left[H\left(p_S(x_t | x_{<t}),\ p_T(x_t | x_{<t}) \right) \right]}_{\substack{\text{See proof in paper.} \\ \text{We will not prove this}}}$$

Applying probability density distribution to DDPM

- The teacher model is a pretrained text-conditional DDPM. Thinking of time as continuous between 0 and 1, it gives rise to probability distribution of z_t, which is denoted by

$$\hat{p}_\phi(z_t, t, y)$$

- The student model is the forward process that starts from $x = g(\theta)$. The distribution of z_t is given by

$$q(z_t | x = g(\theta)).$$

- Applying probability density distillation, we seek to minimize the KL divergence between the teacher and student distributions.

$$D_{KL}\left(q(z_t | x = g(\theta)) \,\|\, \hat{p}_\phi(z_t, t, y) \right)$$

$$= E_{\xi \sim N(0, I)} \left[\log q(z_t | x = g(\theta)) - \log \hat{p}_\phi(z_t, t, y) \right]$$

- As a result,

$$\nabla_\theta D_{KL}\left(q(z_t | x = g(\theta)) \,\|\, \hat{p}_\phi(z_t, t, y) \right)$$

$$= E_{\xi \sim N(0, I)} \left[\underbrace{\nabla_\theta \log q(z_t | x = g(\theta))}_{\downarrow} - \underbrace{\nabla_\theta \log \hat{p}_\phi(z_t, t, y)}_{} \right]$$

$\xi \sim N(0,I)$ [illegible]

entropy of Gaussian with variance σ_t^2, so does not depend on $\theta \rightarrow$ so zero

$= \left(\nabla_{z_t} \log \hat{p}_\phi(z_t, t, y) \right) \frac{\partial z_t}{\partial \theta}$

$$= E_{\xi \sim N(0,I)} \left[-\hat{s}_\phi(z_t, t, y) \frac{\partial z_t}{\partial \theta} \right]$$

recall $z_t = \alpha_t x + \sigma_t \xi$

$$= E_{\xi \sim N(0,I)} \left[\frac{\hat{\xi}_\phi(z_t, t, y)}{\sigma_t} \alpha_t \frac{\partial x}{\partial \theta} \right]$$

$$= E_{\xi \sim N(0,I)} \left[\frac{\alpha_t}{\sigma_t} \hat{\xi}_\phi(z_t, t, y) \frac{\partial x}{\partial \theta} \right].$$

- Now, since $\xi \sim N(0,I)$, we have that

$$0 = E_{\xi \sim N(0,I)} [\xi]$$
$$= E_{\xi \sim N(0,I)} \left[\frac{\alpha_t}{\sigma_t} \xi \frac{\partial x}{\partial \theta} \right]$$

So,

$$\nabla_\theta D_{KL}\left(q(z_t | x = g(\theta)) \,\|\, \hat{p}_\phi(z_t, t, y) \right)$$
$$= E_{\xi \sim N(0,I)} \left[\frac{\alpha_t}{\sigma_t} \left(\hat{\xi}_\phi(z_t, t, y) - \xi \right) \frac{\partial x}{\partial \theta} \right]$$

- The paper explains that it is better to include ξ in the expression because it acts like a control variate and reduces the variance of the gradient.

  ↳ Read more about this in the "Stick the landing" paper by Roeder et al.

  https://arxiv.org/abs/1703.09194

- The score distillation sampling gradient arises from trying to minimize the expected weighted KL-divergence with respected to t.

$$\nabla_\theta \mathcal{L}_{SDS} = E_{t\sim U(0,1)}\left[2\sigma_t w(t) \nabla_\theta D_{KL}(q \,\|\, \hat{p}_\phi) \right]$$

$$= E_{t\sim U(0,1),\, \xi\sim N(0,I)}\left[2\alpha_t w(t) \left(\hat{\xi}_\phi(z_t,t,y) - \xi\right) \frac{\partial x}{\partial \theta} \right].$$

The SDS algorithm

- We have justified the SDS gradient. Now it's time to specify the algorithm to find θ.

Input:
1. DDPM noise model $\hat{p}_\phi$
2. generator function g
3. conditioning info y

Initialize θ
while not satisfied **do**
  Sample $t \sim \text{Uniform}(0,1)$
  Sample $\xi \sim N(0, I)$
  $x \leftarrow g(\theta)$
  $z_t \leftarrow \alpha_t x + \sigma_t \xi$
  $\xi_t \leftarrow \hat{\xi}_\phi(z_t, t, y)$
  $G \leftarrow 2\alpha_t w(t) \nabla_\theta \left(\text{stopgradient}(\hat{\xi}_t - \xi) \cdot x \right)$
  Update θ with G
end while

$\hat{\xi}_t$ has dependency on θ, but we don't want this to appear in this calculation so stop the gradient.

$\nabla_\theta (a \cdot x) = a \frac{\partial x}{\partial \theta}$

NeRF details

- DreamFusion applies SDS to g that is a NeRF.

- The paper uses mip-NeRF 360 variation of NeRF.
  ↳ Paper link: https://arxiv.org/abs/2111.12077

- The paper modifies the NeRT to output albedo instead of radiance.

- The surface normal at each point is computed by the negative gradient with respect to the position of the volumetric density τ.

$$n = -\nabla_x \tau / \| \nabla_x \tau \|.$$

- The 3D geometry is shaded with an ambient light term and a point light term, assuming that the surface is Lambertian.

- Apart from the NeRF, there's another MLP that computes background environmental light.

- The paper composites the rendered object with environment radiance using the alpha values computed from visibility computed from volumetric density.

- The paper employs a number of geometric regularizers.
  ↳ Penalty on opacity along rays from Dream Fields paper.
  https://arxiv.org/abs/2112.01455
  ↳ Orientation loss from Ref-NeRF
  https://arxiv.org/abs/2112.03907

More details in the paper though.

Dream Fusion algorithm

DreamFusion algorithm

- Inputs: ① A pretrained text-to-image DDPM (Imagen)
  ② A NeRF parameterization $g(\theta)$
  ③ Conditioning text
  ↳ Used to derive conditioning info y.

- The algorithm is basically score distillation sampling (SDS) with specification on how to perform the $x \leftarrow g(\theta)$ step.

- Details on how to render is as follows:
  - ↳ We sample the camera by sampling its:
    ① position
    ② look-at point around the origin
    ③ the up vector
  - ↳ Then, we sample a point light source position close to the camera. The paper specifies nothing about how to sample lights.
  - ↳ Use the NeRF to render a 64×64 image to feed the DDPM.
  - ↳ There are 3 modes of rendering
    ① Normal mode: All features are turned on.
    ② Texture-less mode: The albedo of all points is set to white
    ↳ The paper says it helps avoid degenerate cases such as rendering a flat surface with an image on it.
    ③ Albedo-only mode: Outputs albedo without shading.
  - ↳ Texts indicating viewing configuration is added to

↳ Texts indicating viewing configuration is added to the text prompt.

- In $\mathcal{L}_{SDS}$, the authors use
  - ↳ $w(t) = \sigma_t^2$
  - ↳ $t \sim \text{Uniform}(0.02, 0.98)$ to avoid numerial instabilities.

- Classifier guidance weight was set to 100 (very high)
  ↳ higher guidance → improved sample quality.

- Uses the Distributed Shampoo optimizer, batch size of 4.

- Sampling was done on 4 TPUs with batch size 4.
  The authors optimized the NeRF for 15,000 iterations, taking about 1.5 hours.