Score Jacobian Chaining

Tuesday, December 6, 2022 12:34 AM

- Paper title: "Score Jacobian Chanining: Lifting Pretrained 2D Diffusion Models for 3D generation"
- Paper link: https://pals.ttic.edu/p/score-jacobian-chaining

---

Introduction

- This is another paper that attempts to use a DDPM trained on 2D images to generate 3D models.
- They call their approach "score Jacobian training"
- A DDPM is trained to predict the denoising score

$$\nabla_x \log p_\sigma(x)$$

where σ is the noise level, and $p_\sigma(x)$ is the probability distribution of

$$x + \sigma_t \xi \quad \text{(variance exploding form)}$$

where $\xi \sim N(0, I)$ and $x \sim p_{data}$.

- Now, an image x might be parameterized as $f(\theta)$
- Question: Can the score above be used to optimize θ?
- If we apply the chain rule and multiply

$\nabla_x \log p_\sigma(x)$ with the Jacobian $\frac{\partial x}{\partial \theta}$, then

$\nabla_x \log p_\sigma(x)$ with the Jacobian $\frac{\partial x}{\partial \theta}$, then we should be able to obtain $\frac{\partial \log p_\sigma(x)}{\partial \theta}$.

This means that we should be able to solve the optimization problem

$$\arg\min_\theta \log p_\sigma(x) = \arg\min_\theta \log p_\sigma\big(f(\theta)\big).$$

- When f is a differentiable 3D-to-2D renderer and θ is the parameters of a 3D model, we should be able to generate 3D data with a DDPM.

- The paper sets
  - $\Rightarrow$ θ to be radiance fields stored in voxels
  - $\Rightarrow$ f to be volume rendering function.

- The rendering algorithm allows us to compute

$$J_\pi = \frac{\partial x_\pi}{\partial \theta}$$

where x_π is an image rendered from viewpoint π.

- By aggregating $J_\pi \nabla_x \log p_\sigma(x)$ over multiple viewpoints, we should be able to optimized for a voxelized radiance field that represents a 3D object.

- However, doing so directly leads to an out-of-distribution (OOD) problem.

$\Rightarrow$ DDPMs have only seen noisy images.

$\Rightarrow$ However, $f(\theta)$ produces a clean image.

It seems that the gradient $J_\pi \nabla_x \log p_\sigma(x)$ would only work when σ is low.

- The paper proposes Perturb-and-Average scoring (PAAS) as a solution to the OOD problem.

Background

- A DDPM learns a denoiser D that minimizes

$$E_{x \sim p_{data}, \xi \sim N(0,I)} \left[\| D(x + \sigma\xi; \sigma) - x \|^2 \right]$$

- A popular variant instead learns to predicts the noise.

So, it has a noise prediction network $\hat{\xi}$ that is trained to minimize

$$E_{x \sim p_{data}, \xi \sim N(0,I)} \left[\| \hat{\xi}(x + \sigma\xi; \sigma) - \xi \|^2 \right].$$

- The noise prediction network is related to the denoiser as follows:

$$D(x;\sigma) = x - \sigma\hat{\xi}(x;\sigma).$$

- Let $p_\sigma(x)$ denote the distribution of $x + \sigma\xi$ where $x \sim p_{data}$, $\xi \sim N(0, I)$.

- It follows that, when D is optimized well,

$$\underbrace{\nabla_x \log p_\sigma(x)}_{\text{score}} \approx \frac{-\hat{\xi}(x;\sigma)}{\sigma} = \frac{D(x;\sigma) - x}{\sigma^2}$$

- The noise prediction model is trained to predict the score at several noise levels $\sigma_T > \sigma_{T-1} > \dots > \sigma_0 = 0$

- For the Ho et al. paper, the σ_t's are in the range $[0.01, 157]$.

↳ Note, however, that the Ho et al. paper uses the variance preserving formulation. So, σ_t is not $\sqrt{1-\bar{\alpha}_t}$. Rather, $\sigma_t = 1/\sqrt{\bar{\alpha}_t}$

↳ My expository article
https://pkhungurn.github.io/notes/notes/ml/ddpm-expository/index.html
estimates that $\bar{\alpha}_{1000} \approx 10^{-2.192} \approx \frac{1}{156}$, so the paper's estimate of the range of σ_t seems accurate.

- Score also behaves like mean-shift.

⇒ Let us simplify p_{data} to be an empirical data distribution over i.i.d samples $\{y_i\}$

⇒ At noise level σ, we have that

$$p_\sigma(x) = E_{y \sim p_{data}}\left[N(x; y, \sigma^2 I) \right]$$

⇒ In this case, there's a closed form expression to the optimal denoiser.

$$D(x, \sigma) = \frac{\sum_i N(x; y_i, \sigma^2 I)\, y_i}{\sum_i N(x; y_i, \sigma^2 I)}$$

⇒ So, $D(x, \sigma)$ is a weighted combination of y_i's under Gaussian kernel with variance $\sigma^2 I$.

⇒ The score function thus tells us how to update x so that it moves towards its weighted nearest neighbors.

Score Jacobian Chaining

- Let θ be parameters of a 3D model.
  ⇒ We shall specify what it exactly is later.

- Let $p(\theta)$ denote the probability distribution of θ.

- To relate the probability of $p(x)$ of 2D images to the probability $p(\theta)$ of 3D data, we assume that

$$p_\sigma(\theta) \propto E_\pi \left[p_\sigma(x_\pi(\theta)) \right]$$

$$p_\sigma(\theta) \propto \mathbb{E}_\pi [p_\sigma (x_\pi (\theta))]$$

That is, $p_\sigma(\theta)$ is the expected probability of renderings, and the expectation is taken with respect to the camera viewpoint π.

- Now, $$\log p_\sigma(\theta) = \log \left(\mathbb{E}_\pi [p_\sigma (x_\pi (\theta))] \right) - \underbrace{\log Z}_{\text{normalization const}}$$
$$\geq \underbrace{\mathbb{E}_\pi [\log p_\sigma (x_\pi (\theta))] - \log Z}_{\text{Denoted as } \tilde{S}_\sigma(\theta)}$$

through Jensen's inequality.

- Now, $$\underbrace{\nabla_\theta \tilde{S}_\sigma (\theta)}_{\text{"3D score"}} = \mathbb{E}_\pi \left[\frac{\partial \log p_\sigma (x_\pi)}{\partial x_\pi} \cdot \frac{\partial x_\pi}{\partial \theta} \right]$$

$$= \mathbb{E}_\pi \left[\underbrace{\nabla_{x_\pi} \log p_\sigma (x_\pi)}_{\text{2D score pre-trained}} \cdot \underbrace{J_\pi}_{\text{Jacobian.}} \right]$$

(Not a score per se, but a ELBO of the score)

- As mentioned earlier, we have an OOD problem when trying to compute the 2D score.

- More concretely, if we approximate the 2D score with

$$\text{score}(x_\pi, \sigma) \approx \frac{D(x_\pi; \sigma) - x_\pi}{\sigma^2}$$

The problem is that the denoise has only seen images of the form $x = y + \sigma \xi$ where $y \sim p_{data}$, ξ. However, $x_\pi = f(\theta)$ is a clean image that is not corrupted by noise.

- To address the problem, the paper proposes "perturb-and-average scoring" (PAAS)

⇒ This is done by adding noise to the rendered image, and then consider the expected score.

⇒ More concretely,

$$\mathrm{PAAS}(x_\pi, \sqrt{2}\sigma)$$

$$= \mathbb{E}_{\xi \sim N(0,I)}\left[\mathrm{score}(x_\pi + \sigma\xi, \sigma)\right]$$

$$= \mathbb{E}_{\xi \sim N(0,I)}\left[\frac{D(x_\pi + \sigma\xi, \sigma) - (x_\pi + \sigma\xi)}{\sigma^2}\right]$$

$$= \mathbb{E}_{\xi \sim N(0,I)}\left[\frac{D(x_\pi + \sigma\xi, \sigma) - x_\pi}{\sigma^2}\right] - \cancelto{0}{\mathbb{E}_{\xi \sim N(0,I)}\left[\frac{\xi}{\sigma}\right]}$$

- It can be shown that $\boxed{\mathrm{PAAS}(x_\pi, \sqrt{2}\sigma) \approx \nabla_{x_\pi} \log p_{\sqrt{2}\sigma}(x_\pi).}$

- Lemma 1 $\log p_{\sqrt{2}\sigma}(x) \geq \mathbb{E}_{\xi \sim N(0,I)}\left[\log p_\sigma(x + \sigma\xi)\right].$

Proof: $p_{\sqrt{2}\sigma}(x) = \mathbb{E}_{y \sim p_{data}}\left[N(x; y, 2\sigma^2 I)\right]$

$$= \mathbb{E}_{y \sim p_{data}}\left[\mathbb{E}_{\xi \sim N(0,I)}\left[N(x + \sigma\xi; y, \sigma^2 I)\right]\right]$$

$$= \mathbb{E}_{\xi \sim N(0,I)}\left[\mathbb{E}_{y \sim p_{data}}\left[N(x + \sigma\xi; y, \sigma^2 I)\right]\right]$$

$$= \mathbb{E}_{\xi \sim N(0,I)}\left[p_\sigma(x + \sigma\xi)\right]$$

Now, take log of both sides and apply Jensen's inequality □

Now, take log of both sides and apply Jensen's inequality $\square$

- Claim 1: Assuming a trained denoiser D, then PAAS computes the gradient of a lower bound of $\log p_{\sqrt{2}\sigma}(x)$.

Proof: $\nabla_x \left(\mathbb{E}_{\xi \sim N(0,I)} \left[\log p_\sigma (x + \sigma \xi) \right] \right)$

$$= \mathbb{E}_{\xi \sim N(0,I)} \left[\nabla_x \log p_\sigma (x + \sigma\xi) \right]$$

$$= \mathbb{E}_{\xi \sim N(0,I)} \left[\nabla_{x+\sigma\xi} \log p_\sigma (x+\sigma\xi) \cdot \cancelto{I}{\frac{\partial (x+\sigma\xi)}{\partial x}} \right]$$

$$= \mathbb{E}_{\xi \sim N(0,I)} \left[\text{score}(x + \sigma\xi, \sigma) \right]$$

$$\approx \text{PAAS}(x, \sqrt{2}\sigma).$$

We finish the proof by noting that $\mathbb{E}_{\xi \sim N(0,I)} \left[\log p_\sigma (x+\sigma\xi) \right]$ is a lower bound of $\log p_{\sqrt{2}\sigma}(x)$. $\square$

θ and f

- θ here is a voxel radiance field. It has 2 components.
  - $\Rightarrow$ A density voxel grid $V^{(\text{density})} \in \mathbb{R}^{1 \times N_x \times N_y \times N_z}$
  - $\Rightarrow$ A color voxel grid $V^{(\text{color})} \in \mathbb{R}^{3 \times N_x \times N_y \times N_z}$

    RGB

- Given a ray that passes through the volume, we partition the ray into equally long segments of length d.
- At the beginning of the ith segment, we evaluate (RGB_i, τ_i) through trilinear sampling of the voxel grid.
- The color of the ray is given by:

$$C = \sum_i w_i \cdot RGB_i$$

where

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

- The formulation above allows us to find $J_\pi = \frac{\partial x_\pi}{\partial \theta}$ by backpropagating through C.
- Given noisy 2D distance, the model may cheat by populating the entire grid with small densities.
- To avoid the above problem, we add several losses.
- The first loss is the <u>emptiness loss</u>

⇒ We want the density to be sparse: mostly zero except when inside an object.

⇒ To encorage sparsity, we add the following loss:

$$\mathcal{L}_{\text{emptiness}} = \frac{1}{N} \sum_{i=1}^{N} \log(1 + \beta \cdot w_i)$$

$$\mathcal{L}_{emptiness} = \frac{1}{N}\sum_{i=1} \log(1 + \beta \cdot w_i)$$

where w_i is the weight above.

=> The shape of the log function leads to large penalty at the onset of small weights, but it doesn't grow quickly as w_i becomes large.

=> Larger β puts more emphasis on eliminating small densities.

=> The paper uses $\beta = 10$.

=> The emptiness loss is weighted by a hyperparameter λ.

- If λ is too high, it will prevent learning of geometry in early stage of the optimization.

- If λ is too low, there will be floating artifacts.

=> To make λ conducive to high quality outputs, the paper sets $\lambda = 1 \times 10^4$ in the first K iterations.
After that, λ is increased to 2×10^5.

- The 2^{nd} loss is called the center depth loss

=> This is used to constrain the object to be at the center of the picture.

=> $$\mathcal{L}_{center}(D) = -\log\left(\frac{1}{|B|}\sum_{p \in B} D(p) - \frac{1}{|B^C|}\sum_{q \notin B} D(q) \right)$$

- D = depth image
- B = a box (set of pixels) at the center of the image.
- B^C = B's complement.

$\Rightarrow$ I'm uneasy about this loss because it can become undefined when the argument of the logarithm is negative.

Putting it all together

- The algorithm for optimizing θ is given as follows.

for $i \leftarrow 1$ **to** #iterations **do**

Sample a viewpoint π

$x_\pi \leftarrow f(\theta)$; save computation tree to later compute Jacobian-vector product.

Sample σ according to some algorithm.

Sample $\xi \sim N(0, I)$

Compute $\tilde{s} = \dfrac{D(x_\pi + \sigma\xi, \sigma) - x_\pi}{\sigma^2}$

Compute $g_1 = J_\pi \tilde{s}$.

Compute $g_2 = \nabla_\theta \left(\lambda \mathcal{L}_{emptiness} + \hat{\lambda} \mathcal{L}_{center} \right)$

Compute $g_2 = \nabla_\theta(\lambda \mathcal{L}_{emptiness} + \lambda \mathcal{L}_{center})$

Use $g_1 + g_2$ to update θ

end for

- There are two strategies for sampling σ.

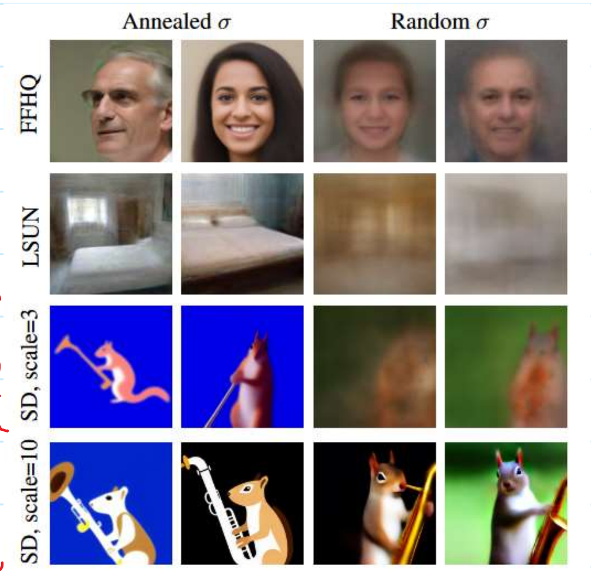
  ⇒ Annealed σ : Start from a high σ and gradually decrease it.

  ⇒ Random σ : Uniformly randomly sample from a range. Much like what is done in the DreamFushion paper.

Validating PAAS on 2D images

- The paper tries to validate its algorithm by setting θ = grid of pixels (an image) and
  f = identity function

-



weak guidance

strong guidance

unconditional DDPM trained on small datasets

text-conditioned DDPM

- For unconditional DDPM where guidance cannot be done out of the box, we have that

① Random σ leads to blurry images

② Annealed σ leads to crisper images

- However, for text-conditioned DDPM, we have that:

① Large guidance weight leads to sharper images for both annealed σ and random σ.

② Random σ can generate sharp images when guidance weight $= 10$. These images seem to be more diverse than the annealed counterpart

- In general, images generated with PAAS are of lower quality than that of normal ancestral sampling.

<u>3D generation</u>

- They conducted an experiment using Stable Diffusion as the DDPM.

- They used random σ and perhaps a high guidance weight.

- Their model was able to generate objects such as animals and Sydney operă house.

- The paper claims its method is competitive with DreamFushion, but no quantitative evidence is provided.

Code

- https://github.com/pals-ttic/sjc/blob/main/run_sjc.py

- One thing to note is that Stable Diffusion, which uses the variance preserving formulation, needs to be adapted so that it works with score Jacobian chaining, which uses the variance exploding formulation.
  - ↳ The adaptation can be found in this file:

    https://github.com/pals-ttic/sjc/blob/ade9a1a81dea638dbdfbc0f5edefdfb94e9642c0/adapt_sd.py