Consistency Trajectory Models

Pramook Khungurn

October 30, 2023

This note is written as I read the "Consistency Trajectory Models: Learning Probability Flow ODE Trajectory for Diffusion" [KLL+23] paper.

1 Introduction

- Consistency trajectory model (CTM) is a follow-up work of consistency model [SDCS23].
- It is a generalization where it allows evaluating both the end point of a trajectory and the rate of change.
- There are two tricks used:
  - Extending the model so that it has the start time t and the end time s for the trajectory.
  - Reparameterize the model so that it becomes something equilavent to the rate of change when $s \rightarrow t$.

  The formulation is interesting.
- However, while the formulation is interesting, the training procedure is complicated and, IMHO, ugly. The paper piles losses after losses to improve the training results. These includes LPIPS and adversarial loss. The resulting improvement is marginal in CIFAR-10 and ImageNet 64, and the differences are so small that you might be able to attribute them initialization.

2 Preliminary

- The forward process of a diffusion model is defined by the forward diffusion process.

  $$\mathrm{d}\mathbf{x}_t = \sqrt{2t}\,\mathrm{d}\mathbf{w}_t$$

  with $\mathbf{x}_0 \sim p_{\mathrm{data}}$. This is the linear noise schedule use in the EDM paper [KAAL22].
- The backward process is modeled by the reverse-time SDE [And82]

  $$\mathrm{d}\mathbf{x}_t = -2t\nabla \log p_t(\mathbf{x}_t)\,\mathrm{d}t + \sqrt{2t}\,\mathrm{d}\overline{\mathbf{w}}_t$$

  where $\mathrm{d}\overline{\mathbf{w}}_t$ is the standard Weiner process whose time runs backward, and $p_t(\mathbf{x}_t)$ is the probability distribution of $\mathbf{x}_t$ according to the forward process. The backward process, of course, runs backward in time. It is initialized with $\mathbf{x}_T \sim p_T$.
- The deterministic counterpart of the reverse-time ODE is the probability flow (PF) ODE:

  $$\frac{\mathrm{d}\mathbf{x}_t}{\mathrm{d}t} = -t\nabla \log p_t(\mathbf{x}_t) = \frac{\mathbf{x}_t - E_{p_{t\rightarrow 0}(\mathbf{x}|\mathbf{x}_t)}[\mathbf{x}|\mathbf{x}_t]}{t}$$

  where $p_{t\rightarrow 0}(\mathbf{x}|\mathbf{x}_t)$ is the probability distribution of the solution of the reverse-time stochastic process from time t to zero, initiated from $\mathbf{x}_t$.

- The expression $E_{p_{t\to 0}(\mathbf{x}|\mathbf{x}_t)}[\mathbf{x}|\mathbf{x}_t]$ is referred to by the paper as the "denoiser." By Tweedie's formula, we have that

$$E_{p_{t\to 0}(\mathbf{x}|\mathbf{x}_t)}[\mathbf{x}|\mathbf{x}_t] = \mathbf{x} + t^2 \nabla \log p_t(\mathbf{x}_t).$$

- The denoiser is modeled by a neural network $D_{\boldsymbol{\phi}}$, which is trained with the denoising score matching loss:

$$\mathcal{L}(\boldsymbol{\phi}) = E_{\substack{\mathbf{x}_0 \sim p_{\text{data}},\\ t\sim p_{\text{time}},\\ \mathbf{x}\sim p_{0\to t}(\mathbf{x}|\mathbf{x}_0)}} \Big[\|\mathbf{x}_0 - D_{\boldsymbol{\phi}}(\mathbf{x}, t)\|^2 \Big]$$

  where $p_{0\to t}(\mathbf{x}|\mathbf{x}_t)$ is the conditional probability of the forward process that is initiated at a particular data point $\mathbf{x}_0$.

- The empirical ODE that is used for sampling is given by

$$\frac{\mathrm{d}\mathbf{x}_t}{\mathrm{d}t} = \frac{\mathbf{x}_t - D_{\boldsymbol{\phi}}(\mathbf{x}_t, t)}{t}.$$

  Sampling thus involves computing the integral

$$\int_T^0 \frac{\mathrm{d}\mathbf{x}_t}{\mathrm{d}t}\,\mathrm{d}t,$$

  which is equivalent to

$$\mathbf{x}_0 = \mathbf{x}_t + \int_T^0 \frac{\mathbf{x}_t - D_{\boldsymbol{\phi}}(\mathbf{x}_t, t)}{t}\,\mathrm{d}t.$$

3 Method

3.1 The Model

- CTM predicts both infinitestimal changes and intermediate points of the PF ODE.
- Let $G(\mathbf{x}_t, t, s)$ be the solution of the PF ODE from the initial condition $\mathbf{x}_t$ at time t to the final time $s \le t$

$$G(\mathbf{x}_t, t, s) := \mathbf{x}_t + \int_t^s \frac{\mathbf{x}_u - E_{p_{u\to 0}}[\mathbf{x}|\mathbf{x}_u]}{u}\,\mathrm{d}u$$

- Now, we may rewrite G with something closer to the DDIM update rule:

$$\mathbf{x}_s \approx \frac{\sigma_s}{\sigma_t}\mathbf{x}_t + \bigg(\alpha_s - \alpha_t \frac{\sigma_s}{\sigma_t}\bigg) E[\mathbf{x}|\mathbf{x}_t].$$

  In our case, $\sigma_t = t$ and $\alpha_t = 1$. So, the write would be something like this:

$$G(\mathbf{x}_t, t, s) = \frac{s}{t}\mathbf{x}_t + \bigg(1 - \frac{s}{t}\bigg) \times \text{something}.$$

  where the "something" should approach $E[\mathbf{x}|\mathbf{x}_t]$ as $s \to t$.

- The expression for "something" in this paper is as in the following lemma.

Lemma 1. *Suppose that the score satisfies* $\sup_{\mathbf{x}} \int_0^T \|\nabla \log p_u(\mathbf{x})\| \,\mathrm{d}u < \infty$. *Then,* $G(\mathbf{x}_t, t, s)$ *can be expressed as:*

$$G(\mathbf{x}_t, t, s) = \frac{s}{t}\mathbf{x}_t + \left(1 - \frac{s}{t}\right) g(\mathbf{x}_t, t, s)$$

where

$$g(\mathbf{x}_t, t, s) = \mathbf{x}_t + \frac{t}{t-s}\int_t^s \frac{\mathbf{x}_u - E_{p_{u\to 0}}[\mathbf{x}|\mathbf{x}_u]}{u}\,\mathrm{d}u$$

Here, g satisfies:

- *When $s = 0$, we have that $G(\mathbf{x}_t, t, 0) = g(\mathbf{x}_t, t, 0)$.*
- *As $s \to t$, we have that $g(\mathbf{x}_t, t, s) \to E_{p_{t\to 0}(\mathbf{x}|\mathbf{x}_t)}[\mathbf{x}|\mathbf{x}_t]$.*

- CTM seeks to approximate the little g function with a neural network $g_{\boldsymbol{\theta}}$ so that we have a neural network

$$G_{\boldsymbol{\theta}}(\mathbf{x}_t, t, s) := \frac{s}{t}\mathbf{x}_t + \left(1 - \frac{s}{t}\right) g_{\boldsymbol{\theta}}(\mathbf{x}_t, t, s).$$

This parametermization gives $G_{\boldsymbol{\theta}}(\mathbf{x}_t, t, t) = \mathbf{x}_t$ for free, which helps with training stability.

3.2 Training

- Our goal is to make sure that the neural network should match the groundtruth integrator G.

$$G_{\boldsymbol{\theta}}(\mathbf{x}_t, t, s) \sim G(\mathbf{x}_t, t, s)$$

for any $s \leq t$.

- We don't have access directly to G. However, if we have a teacher diffusion model $D_{\boldsymbol{\phi}}$ we can approximate it with an ODE solver:

$$G_{\boldsymbol{\theta}}(\mathbf{x}_t, s) \approx \textsc{Solver}(\mathbf{x}_t, t, s; \boldsymbol{\phi}). \tag{1}$$

- Equivalently, we can also instead relax the goal and try to match local consistency:

$$G_{\boldsymbol{\theta}}(\mathbf{x}_t, t, s) \approx G_{\widetilde{\boldsymbol{\theta}}}(\textsc{Solver}(\mathbf{x}_t, t, t - \Delta; \boldsymbol{\phi}), t - \Delta t, s) \tag{2}$$

where $\Delta t \in [0, t-s]$ and $\widetilde{\boldsymbol{\theta}}$ is the slow varying parameters of G computed by EMA:

$$\widetilde{\boldsymbol{\theta}}^{(k+1)} := \mu \widetilde{\boldsymbol{\theta}}^{(k+1)} + (1-\mu)\boldsymbol{\theta}.$$

The paper says their models work well with $\mu = 0.999$ or $\mu = 0.9999$.

- We note that the goals (1) and (2) can be summarized into one same goal the paper call "soft matching":

$$G_{\boldsymbol{\theta}}(\mathbf{x}_t, t, s) \approx G_{\widetilde{\boldsymbol{\theta}}}(\textsc{Solver}(\mathbf{x}_t, t, u; \boldsymbol{\phi}), u, s) \tag{3}$$

for any $u \in [t, s)$. This is because, when $u = s$, we have that

$$G_{\boldsymbol{\theta}}(\textsc{Solver}(\mathbf{x}_t, t, s; \boldsymbol{\phi}), s, s) = \textsc{Solver}(\mathbf{x}_t, t, s; \boldsymbol{\phi})$$

by how $G_{\boldsymbol{\theta}}$ is parameterized.

- We now need to turn the goal (3) into a loss function. The paper chooses to slap $G_{\widetilde{\boldsymbol{\theta}}}(\cdot, s, 0)$ to both sides of the equation.

$$\begin{aligned}\mathbf{x}_{\mathrm{est}}(\mathbf{x}_t, t, s) &= G_{\mathrm{sg}(\boldsymbol{\theta})}(G_{\boldsymbol{\theta}}(\mathbf{x}_t, t, s), t, s)\\ \mathbf{x}_{\mathrm{target}}(\mathbf{x}_t, t, u, s) &= G_{\mathrm{sg}(\boldsymbol{\theta})}(G_{\widetilde{\boldsymbol{\theta}}}(\textsc{Solver}(\mathbf{x}_t, t, u; \boldsymbol{\phi}), u, s), t, s)\end{aligned}$$

  Then it computes some kind of distance between $\mathbf{x}_{\mathrm{est}}$ and $\mathbf{x}_{\mathrm{target}}$. The paper cites several reasons for this:

  - The diffusion model here has the variance exploding formulation. If we compare the values at time s, the loss may overemphasize differences at large times because of the scale of the values.
  - Values at time $s > 0$ are noisy. It is hard to use losses that are specialized to images such as perceptual loss, LPIPS, or adversarial loss on these values.

- So, to learn consistent trajectory, we use the following loss:

$$\mathcal{L}_{\mathrm{CTM}}(\boldsymbol{\theta}) = E_{\substack{\mathbf{x}_0 \sim p_{\mathrm{data}},\\ t \in [0,T],\\ s \in [0,t],\\ u \in [s,t),\\ \mathbf{x}_t \sim p_{0\to t}(\mathbf{x}_t|\mathbf{x}_0)}} \big[d(\mathbf{x}_{\mathrm{est}}(\mathbf{x}_t, t, s)), \mathbf{x}_{\mathrm{target}}(\mathbf{x}_t, t, u, s)\big]$$

- Training with the above loss, however, might lead to inaccurate estimiation of $g_{\boldsymbol{\theta}}$ when s approaches t because its weight, $1 - s/t$ approaches 0. To mitigate this problem, we use the fact that $g_{\boldsymbol{\theta}}$ should approximate $E[\mathbf{x}_0|\mathbf{x}_t]$ when $s = t$. This leads to the denoising score matching (DSM) loss:

$$\mathcal{L}_{\mathrm{DSM}}(\boldsymbol{\theta}) = E_{\substack{\mathbf{x}_0 \sim p_{\mathrm{data}},\\ t \in [0,T],\\ \mathbf{x}_t \sim p_{0\to t}(\mathbf{x}_t|\mathbf{x}_0)}} \big[\|\mathbf{x}_0 - g_{\boldsymbol{\theta}}(\mathbf{x}_t, t, t)\|^2\big].$$

- In order to enhance CTM's performance beyond the teacher $D_{\boldsymbol{\phi}}$, we can also slap adversarial loss on $\mathbf{x}_{\mathrm{est}}$:

$$\mathcal{L}_{\mathrm{GAN}}(\boldsymbol{\theta}, \boldsymbol{\eta}) = E_{\mathbf{x}_0 \sim p_{\mathrm{data}}}\big[\log D_{\boldsymbol{\eta}}(\mathbf{x}_0)\big] + E_{\substack{\mathbf{x}_0 \sim p_{\mathrm{data}},\\ t \in [0,T],\\ \mathbf{x}_t \sim p_{0\to t}(\mathbf{x}_t|\mathbf{x}_0)}} \big[\log(1 - D_{\boldsymbol{\eta}}(\mathbf{x}_{\mathrm{est}}(\mathbf{x}_t, t, 0)))\big]$$

  where $D_{\boldsymbol{\eta}}$ is a discriminator network.

- The overall loss function is given by

$$\mathcal{L}(\boldsymbol{\theta}, \boldsymbol{\eta}) = \mathcal{L}_{\mathrm{CTM}}(\boldsymbol{\theta}) + \lambda_{\mathrm{DSM}}\mathcal{L}_{\mathrm{DSM}}(\boldsymbol{\theta}) + \lambda_{\mathrm{GAN}}\mathcal{L}_{\mathrm{GAN}}(\boldsymbol{\theta}, \boldsymbol{\eta}).$$

- The training algorithm is as follows:

```
repeat
    Sample $\mathbf{x} \sim p_{\mathrm{data}}$.
    Sample $\boldsymbol{\xi} \sim \mathcal{N}(\mathbf{0}, I)$.
    Sample $t \in [0,T]$, $s \in [0,T]$, and $u \in [s,t]$.
    Calculate $\mathbf{x}_t \leftarrow \mathbf{x}_0 + \boldsymbol{\xi}$.
    Calculate $\textsc{Solver}(\mathbf{x}_t, t, u; \boldsymbol{\phi})$.
    Update $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} - \alpha\frac{\partial}{\partial\boldsymbol{\theta}}\mathcal{L}(\boldsymbol{\theta}, \boldsymbol{\eta})$.
    Update $\boldsymbol{\eta} \leftarrow \boldsymbol{\eta} + \alpha\frac{\partial}{\partial\boldsymbol{\theta}}\mathcal{L}_{\mathrm{GAN}}(\boldsymbol{\theta}, \boldsymbol{\eta})$.
    Update $\widetilde{\boldsymbol{\theta}} \leftarrow \mu\widetilde{\boldsymbol{\theta}} + (1-\mu)\boldsymbol{\theta}$.
until converge
```

  Here, α is the learning rate.

3.3 Training from Scratch

- We can replace $D_{\boldsymbol{\phi}}(\mathbf{x}_t, t)$ with $\mathbf{g}_{\boldsymbol{\theta}}(\mathbf{x}_t, t, t)$.
- So, $\mathbf{x}_{\mathrm{target}}$ now becomes $G_{\widetilde{\boldsymbol{\theta}}}(G_{\boldsymbol{\theta}}(\textsc{Solver}(\mathbf{x}_t, t, u; \widetilde{\boldsymbol{\theta}}), u, s), s, 0)$.

3.4 Implementation Details

- Datasets: CIFAR-10 and ImageNet 64.
- Hardware.
  - CIFAR-10: Four V100 GPUS, each with 16G RAM.
  - ImageNet 64: Eight A100 GPUs, each with 40G RAM.

  It seems that they used ABCI?
- The paper parametermizes $g_{\boldsymbol{\theta}}$ with the EDM conditioning:

$$g_{\boldsymbol{\theta}(\mathbf{x}_t, t, s)} = \frac{\sigma_{\mathrm{data}}^2}{t^2 + \sigma_{\mathrm{data}}^2}\mathbf{x}_t + \frac{t\sigma_{\mathrm{data}}}{\sqrt{t^2 + \sigma_{\mathrm{data}}^2}}\mathrm{NN}_{\boldsymbol{\theta}}(\mathbf{x}_t, t, s).$$

  where $\mathrm{NN}_{\boldsymbol{\theta}}$ is a neural network, and $\sigma_{\mathrm{data}} = 0.5$.
- I'm not so sure whether they scale $\mathbf{x}_t$ with $1/(t^2 + \sigma_{\mathrm{data}}^2)$ like the EDM paper does. Maybe the authors just forgot to mention this.
- $g_{\boldsymbol{\theta}}$ is initalized to the parameters of the teacher $D_{\boldsymbol{\phi}}$.
- Training has two phases based on whether the adversarial loss is used or not.
  - For CIFAR-10, λ_{GAN} is set to 0 for the first 50K iterations. It then gratually increases 1.
  - For ImageNet 64, λ_{GAN} is set to 0 for the first 10K iterations. It then gratually increases 1.
- Batch sizes for non-adversarial losses:
  - CIFAR-10: 256
  - ImageNet 64: 2048
- Batch sizes for adversarial loss:
  - CIFAR-10: 64 in the first phase (16 for each GPU), and 44 in the second phase (11 for each GPU) after $\mathcal{L}_{\mathrm{GAN}}$ is activate.
  - ImageNet 64: 128 in the first phase (16 for each GPU), and 88 in the second phase (11 for each GPU).
- The paper uses LPIPS loss for the difference function d in L_{CTM}.
- Sampling of t and s are based on N discretized time steps used in the consistency model paper [SDCS23].
  - For CIFAR-10, the paper sets $N = 18$.
  - For ImageNet 64, the paper sets $N = 40$.
- Evaluating the Solver function.
  - For CIFAR-10, the paper fixes the maximum number of ODE steps to 17.
  - For ImageNet 64, the paper fixes the maximum number of ODE steps to 20.

If we increase the number of maximum ODE steps, the results would be better. The paper does not specify what ODE solver it uses, but I would assume it's Heun's method.

- $\mathcal{L}_{\text{DSM}}$ calculation.
  - For 50% of the time, the samper sample t from the EDM's original scheme $\log t \sim \mathcal{N}(-1.2, 1.2^2)$.
  - For 50% of the time, the paper samples $\xi \sim \mathcal{U}([0, 0.7])$, and then transforms it to

    $$t \leftarrow (\sigma_{\text{max}}^{1/\rho} + \xi(\sigma_{\text{min}}^{1/\rho} - \sigma_{\text{max}}^{1/\rho}))^\rho.$$

    So that the training can work on large times. Here, $\sigma_{\text{min}} = 0.002$, $\sigma_{\text{max}} = 80$, and $\rho = 7$.
- $\mathcal{L}_{\text{GAN}}$ calculation.
  - Use two feature extractors.
    * EfficientNet.
    * DeIT-base.
  - Before obtaining the features, upscale the images to 224×224 with bilinear interpolation.
  - Apply cross-channel mixing and cross-scale mixing like the StyleGAN-XL paper.
  - Cross-scale mixing results in a feature pyramid which has four different resolutions.
  - Hence, there are 8 discriminators: 4 for EfficientNet, and 4 for DeIT-base.
- Learning reates
  - CIFAR-10: 4×10^{-4} for the generator and 2×10^{-3} for the discriminators.
  - ImageNet 64: 8×10^{-6} for the generator and 2×10^{-3} for the discriminators.
- EMA
  - CIFAR-10: $\mu = 0.9999$
  - ImageNet 64: $\mu = 0.999$.
- Training length:
  - CIFAR-10: 100K iterations.
  - ImageNet 64: 30K iterations.

4 Sampling

- CTM enables normal score evaluation through $\mathbf{g}_{\boldsymbol{\theta}}(\mathbf{x}_t, t, t)$.
- CTM also enable long jumps using an algorithm called γ-sampling.
  - Suppose the sampling time steps are $T = t_0 > t_1 > \dots > t_N = 0$.
  - We first sample $\mathbf{x}_{t_0} \sim \mathcal{N}(0, \sigma_{\text{max}}^2 I)$.
  - Suppose we have sampled $\mathbf{x}_{t_i}$.
    * Use $G_{\boldsymbol{\theta}}(\mathbf{x}_{t_0}, t_0, \sqrt{1-\gamma^2} t_1)$ to get a sample, overshooting t_1 by a little.
    * Then, we add $\gamma t_1 \boldsymbol{\xi}$ to the sample where $\xi \in \mathcal{N}(\mathbf{0}, I)$ to get back to the noise level at t_1.
  - We repeat the process until we get to $t_N = 0$. (Of course, we have to be careful of the boundary case.)

5 Results

- CIFAR-10
  - 1 NFE: 1.98 (unconditional), 1.73 (conditional), 2.39 (from scratch unconditional)
  - 2 NFEs: 1.87 (unconditional), 1.63 (conditional)
- ImageNet 64
  - 1 NFE: 2.06
  - 2 NFEs: 1.90

References

[And82] Brian D.O. Anderson. Reverse-time diffusion equation models. *Stochastic Processes and their Applications*, 12(3):313–326, 1982.

[KAAL22] Tero Karras, Miika Aittala, Timo Aila, and Samuli Laine. Elucidating the design space of diffusion-based generative models, 2022.

[KLL$^+$23] Dongjun Kim, Chieh-Hsin Lai, Wei-Hsiang Liao, Naoki Murata, Yuhta Takida, Toshimitsu Uesaka, Yutong He, Yuki Mitsufuji, and Stefano Ermon. Consistency trajectory models: Learning probability flow ode trajectory of diffusion, 2023.

[SDCS23] Yang Song, Prafulla Dhariwal, Mark Chen, and Ilya Sutskever. Consistency models, 2023.