BOOT: Data-free Distillation of Denoising Diffusion Models with Bootstrapping

Pramook Khungurn

June 27, 2023

This note was written as I read the paper "BOOT: Data-free Distillation of Denoising Diffusion Models with Bootstrapping" by Gu et al. [GZZ+23].

1 Introduction

- It is a known problem that diffusion models are slow compared other generative models such that GANs or VAEs.
- There are techniques that distills diffusion models into models that can generate data in few model evaluation steps. These include:
  - Straightforward knowledge distillation [LL21].
  - Progressive distillation [SH22, MRG+23].
  - Cosistency models [SDCS23].
- However, these techniques require either (1) generating a lot of synthetic data with the teacher model or (2) using the training dataset in the process of distillation.
- The requirements above make it hard to apply for text-condition diffusion models, which currently requires a very large dataset to train.
- The paper proposes BOOT, which can distill diffusion models without needing any data.
- It is inspired by consistency models [SDCS23].
  - The process of sampling a diffusion model can be thought of as tracing a trajectory of a particle moving through a velocity field defined by the probability flow ODE [SSDK+21].
  - In the consistency model work, Song et al. observes that all points in a specific trajectory form an equivalent class. What we want to do is to find the terminal point from any other points the trajectory.
  - So, a consistency model wants any $\mathbf{x}_t$ in the same trajectory to map to the same $\mathbf{x}_0$.
- On the other hand, BOOT predicts all possible $\mathbf{x}_t$ given the Gaussian noise $\boldsymbol{\xi}$ and time t.
- The name "BOOT" comes from the word "bootstrapping," which is used in the meaning that it becomes easier to predict $\mathbf{x}_t$ if the model has already learned to predict $\mathbf{x}_{t'}$ with $t' > t$.

2 Background

2.1 Diffusion Model

- The paper uses the continuous-time [SSDK+21, KAAL22] and variance-preserving [SH22] formulation.
- A data point is denoted by $\mathbf{x} \in \mathbb{R}^N$.
- The forward process generates a stochastic process $\{\mathbf{x}_t : t \in [0, T]\}$ with $\mathbf{x}_0 = \mathbf{x} \sim p_{\text{data}}$ governed by the **noise schedule** α_t and σ_t with $\alpha_t^2 + \sigma_t^2 = 1$ (because we are using the variance-preserving formulation). The stochastic process has the following properties:

$$q(\mathbf{x}_t|\mathbf{x}) = \mathcal{N}(\mathbf{x}_t; \alpha_t \mathbf{x}, \sigma_t^2 I),$$
$$q(\mathbf{x}_t|\mathbf{x_s}) = \mathcal{N}(\mathbf{x}_t; \alpha_{t|s} \mathbf{x}_s, \sigma_{t|s}^2 I)$$

  where

$$\alpha_{t|s} = \frac{\alpha_t}{\alpha_s}$$
$$\sigma_{t|s}^2 = \sigma_t^2 - \alpha_{t|s}^2 \sigma_s^2$$

  for $s < t$.

- The quality α_t^2/σ_t^2 is called the **signal-to-noise ratio (SNR)**. It decreases monotonically with t.
- A diffusion model is denoted by $\mathbf{f}_{\boldsymbol{\phi}}$. In this paper, it predicts the denoised data $\mathbf{x}$ from $\mathbf{x}_t$ and t. It is trained with the following objective:

$$\mathcal{L}_{\boldsymbol{\phi}}^{\text{diff}} = E_{\mathbf{x} \sim p_{\text{data}}, t \sim [0,T], \mathbf{x}_t \sim q(\mathbf{x}_t|\mathbf{x})} \big[w_t \| \mathbf{f}_{\boldsymbol{\phi}}(\mathbf{x}_t, t) - \mathbf{x} \|^2 \big]$$

  where w_t is the weight used to "balance perceptual quality and diversity."

- Given a diffusion model, a sample can be generated deterministically from a Gaussian noise by using the DDIM sampler [SME20], which has the following update rule:

$$\mathbf{x}_s = \frac{\sigma_s}{\sigma_t} \mathbf{x}_t + \left(\alpha_s - \alpha_t \frac{\sigma_s}{\sigma_t} \right) \mathbf{f}_{\boldsymbol{\phi}}(\mathbf{x}_t, t)$$

  where $s < t$. The process starts with sampling $\mathbf{x}_T = \boldsymbol{\xi} \sim \mathcal{N}(\mathbf{0}, I)$, and then we can obtain $\mathbf{x}_t$ with smaller and smaller t until we reach $t \approx 0$. The catch is that the steps size $\delta = t - s$ must be small enough.

2.2 Knowledge Distillation

- The student model is denoted by $\mathbf{g}_{\boldsymbol{\theta}}$. This is used in constrast with the teacher diffusion model $\mathbf{f}_{\boldsymbol{\phi}}$.
- The most straightforward form of distillation is direct distillation where the student model is trained with the following loss function:

$$\mathcal{L}_{\boldsymbol{\theta}}^{\text{direct}} = E_{\boldsymbol{\xi} \sim \mathcal{N}(\mathbf{0}, I)} \big[\| \mathbf{g}_{\boldsymbol{\theta}}(\boldsymbol{\xi}) - \texttt{ODE-Solver}(\mathbf{f}_{\boldsymbol{\phi}}, \boldsymbol{\xi}, T \to 0) \|^2 \big]$$

  where `ODE-Solver` is any sampler like the DDIM sampler in the last section.

- The drawback of the above approach is that the `ODE-Solver` needs many steps in order to make the student model generate high quality data. This can make the training process slow.
- However, notice that the approach does not require access to training data at all.

- Other approaches such as progressive distillation [SH22], consistency models [SDCS23], and TRACT [BAL+23] avoid running the full `ODE-Solver` from T to 0.

- For the consistency model, the student model is conditioned on time, which means that we want $\mathbf{g}_{\boldsymbol{\theta}}(\mathbf{x}_t, t)$ to predict $\mathbf{x}$. The model is trained with the following loss function:

$$\mathcal{L}_{\boldsymbol{\theta}}^{\mathrm{CM}} = E_{\mathbf{x} \sim p_{\mathrm{data}}, s, t \sim [0,T], s<t, \mathbf{x}_t \sim q(\mathbf{x}_t|\mathbf{x})} \left[\| \mathbf{g}_{\boldsymbol{\theta}}(\boldsymbol{x}_t, t) - \mathbf{g}_{\boldsymbol{\theta}_{\mathrm{EMA}}}(\mathbf{x}_s, s) \|^2 \right]$$

  where $\mathbf{x}_s = \texttt{ODE-Solver}(\mathbf{f}_{\boldsymbol{\phi}}, \mathbf{x}_t, t \to s)$ and $\boldsymbol{\theta}_{\mathrm{EMA}}$ is the exponential moving average of the student parameters $\boldsymbol{\theta}$.

- While training a consistency model does not require executing the `ODE-Solver` from start to finish, it requires access to the training data.

3 Method

- The goal of booth is to train a time-condition model $\mathbf{g}_{\boldsymbol{\theta}}(\boldsymbol{\xi}, t)$ that predicts $\mathbf{x}_t = \texttt{ODE-Solver}(\mathbf{f}_{\boldsymbol{\phi}}, \boldsymbol{\theta}, T \to t)$ when $\boldsymbol{\xi} \sim \mathcal{N}(\mathbf{0}, I)$.

- We can generate a sample by evaluating $\mathbf{g}_{\boldsymbol{\theta}}(\boldsymbol{\xi}, 0)$ after sampling $\boldsymbol{\theta} \sim \mathcal{N}(\mathbf{0}, I)$.

- Since the model always takes a Gaussian noise as input, there is no need for a dataset during the training process.

3.1 Signal-ODE

- Predicting $\mathbf{x}_t$ is hard because it is a noisy image.

- It is much easier to predict $\mathbf{y}_t = (\mathbf{x}_t - \sigma_t \boldsymbol{\theta})/\alpha_t$, which is supposed to represent a predicted denoised image or the "signal component" of $\mathbf{x}_t$.

- Moreover, we know that

$$\begin{aligned}
\mathbf{y}_s &= \frac{\mathbf{x}_s - \sigma_s \boldsymbol{\xi}}{\alpha_s} \\
&= \frac{1}{\alpha_s}\mathbf{x}_s - \frac{\sigma_s}{\alpha_s}\boldsymbol{\xi} \\
&= \frac{1}{\alpha_s}\left[\frac{\sigma_s}{\sigma_t}\mathbf{x}_t + \left(\alpha_s - \alpha_t \frac{\sigma_s}{\sigma_t} \right) \mathbf{f}_{\boldsymbol{\phi}}(\mathbf{x}_t, t) \right] - \frac{\sigma_s}{\alpha_s}\boldsymbol{\xi} \\
&= \left(1 - \frac{\sigma_s}{\alpha_s}\frac{\alpha_t}{\sigma_t} \right) \mathbf{f}_{\boldsymbol{\phi}}(\mathbf{x}_t, t) + \frac{\sigma_s}{\alpha_s}\left(\frac{\mathbf{x}_t}{\sigma_t} - \boldsymbol{\xi} \right) \\
&= \left(1 - \frac{\sigma_s}{\alpha_s}\frac{\alpha_t}{\sigma_t} \right) \mathbf{f}_{\boldsymbol{\phi}}(\mathbf{x}_t, t) + \frac{\sigma_s}{\alpha_s}\left(\frac{\mathbf{x}_t - \sigma_t \boldsymbol{\xi}}{\sigma_t} \right) \\
&= \left(1 - \frac{\sigma_s}{\alpha_s}\frac{\alpha_t}{\sigma_t} \right) \mathbf{f}_{\boldsymbol{\phi}}(\mathbf{x}_t, t) + \frac{\sigma_s}{\alpha_s}\frac{\alpha_t}{\sigma_t}\left(\frac{\mathbf{x}_t - \sigma_t \boldsymbol{\xi}}{\alpha_t} \right) \\
&= \left(1 - \frac{\sigma_s}{\alpha_s}\frac{\alpha_t}{\sigma_t} \right) \mathbf{f}_{\boldsymbol{\phi}}(\mathbf{x}_t, t) + \frac{\sigma_s}{\alpha_s}\frac{\alpha_t}{\sigma_t}\mathbf{y}_t.
\end{aligned}$$

  Define $\lambda_t = -\log(\alpha_t/\sigma_t)$, we have that the above equation can be rewritten as:

$$\mathbf{y}_s = (1 - e^{\lambda_s - \lambda_t})\mathbf{f}_{\boldsymbol{\phi}}(\mathbf{x}_t, t) + e^{\lambda_s - \lambda_t}\mathbf{y}_t.$$

- The above equation can be turned into an ODE.

$$\begin{aligned}\mathbf{y}_s - \mathbf{y}_t &= (1 - e^{\lambda_s - \lambda_t})\mathbf{f}_{\boldsymbol{\phi}}(\mathbf{x}_t, t) - (1 - e^{\lambda_s - \lambda_t})\mathbf{y}_t \\ \mathbf{y}_s - \mathbf{y}_t &= (1 - e^{\lambda_s - \lambda_t})\big[\mathbf{f}_{\boldsymbol{\phi}}(\mathbf{x}_t, t) - \mathbf{y}_t\big].\end{aligned}$$

  Dividing both sizes by $s - t$ and taking the limit as $s \to t^-$, we have that

$$\begin{aligned}\lim_{s \to t^-} \frac{\mathbf{y}_s - \mathbf{y}_t}{s - t} &= \bigg(\lim_{s \to t^-} \frac{1 - e^{\lambda_s - \lambda_t}}{s - t} \bigg) \big[\mathbf{f}_{\boldsymbol{\phi}}(\mathbf{x}_t, t) - \mathbf{y}_t\big] \\ \frac{\mathrm{d}\mathbf{y}_t}{\mathrm{d}t} &= \bigg(\lim_{s \to t^-} \frac{1 - e^{\lambda_s - \lambda_t}}{s - t} \bigg) \big[\mathbf{f}_{\boldsymbol{\phi}}(\mathbf{x}_t, t) - \mathbf{y}_t\big].\end{aligned}$$

  The limit on the RHS is an indeterminate form $0/0$, so we will apply l'Hôpital's rule.

$$\lim_{s \to t^-} \frac{1 - e^{\lambda_s - \lambda_t}}{s - t} = \lim_{s \to t^-} \frac{\{1 - e^{\lambda_s - \lambda_t}\}'}{\{s - t\}'} = \lim_{s \to t^-} \frac{-e^{\lambda_s - \lambda_t}\{\lambda_s - \lambda_t\}'}{1} = \lim_{s \to t^-} \frac{-e^{\lambda_s - \lambda_t}\lambda'_s}{1} = -\lambda'_t.$$

  As a result,

$$\frac{\mathrm{d}\mathbf{y}_t}{\mathrm{d}t} = -\lambda'_t\big[\mathbf{f}_{\boldsymbol{\phi}}(\mathbf{x}_t, t) - \mathbf{y}_t\big].$$

  The above ODE is called the **signal-ODE**. It is supposed to be integrated with the boundary condition $\mathbf{y}_T = \boldsymbol{\xi} \sim \mathcal{N}(\mathbf{0}, I)$ with time from $t = T$ to $t = 0$. Once we get $\mathbf{y}_0$, we can output this as a sampled data because $\mathbf{y}_0 = \mathbf{x}_0$.

3.2 Learning

- We would like to train a neural network $\mathbf{y}_{\boldsymbol{\theta}}(\boldsymbol{\xi}, t)$ that approximate $\mathbf{y}_t$. This network can be trained with the following loss:

$$\mathcal{L}^{\mathrm{DE}}_{\boldsymbol{\theta}} = E_{\boldsymbol{\xi} \sim \mathcal{N}(\mathbf{0}, I), t \sim [0,T]} \bigg[\bigg\| \frac{\mathrm{d}\mathbf{y}_{\boldsymbol{\theta}}(\boldsymbol{\xi}, t)}{\mathrm{d}t} + \lambda'_t(\mathbf{f}_{\boldsymbol{\phi}}(\hat{\mathbf{x}}_t, t) - \mathbf{y}_{\boldsymbol{\theta}}(\boldsymbol{\xi}, t)) \bigg\|^2 \bigg].$$

  Here, we use the estimate $\hat{\mathbf{x}}_t = \alpha_t \mathbf{y}_{\boldsymbol{\theta}}(\boldsymbol{\xi}, t) + \sigma_t \boldsymbol{\xi}$.

- While computing the time derivative $\mathrm{d}\mathbf{y}_{\boldsymbol{\theta}}(\boldsymbol{\xi}, t)/deet$ is possible with forward mode differentiation, computing the gradient through the derivative can be expensive. Hence, we can approximate the derivative with the difference equation:

$$\begin{aligned}\mathbf{y}_{\boldsymbol{\theta}}(\boldsymbol{\xi}, s) &\approx \mathbf{y}_{\boldsymbol{\theta}}(\boldsymbol{\xi}, t) - (s - t)\lambda'_t(\mathbf{f}_{\boldsymbol{\phi}}(\hat{\mathbf{x}}_t, t) - \mathbf{y}_{\boldsymbol{\theta}}(\boldsymbol{\xi}, t)) \\ &= \mathbf{y}_{\boldsymbol{\theta}}(\boldsymbol{\xi}, t) + (t - s)\lambda'_t(\mathbf{f}_{\boldsymbol{\phi}}(\hat{\mathbf{x}}_t, t) - \mathbf{y}_{\boldsymbol{\theta}}(\boldsymbol{\xi}, t)) \\ &= \mathbf{y}_{\boldsymbol{\theta}}(\boldsymbol{\xi}, t) + \delta\lambda'_t(\mathbf{f}_{\boldsymbol{\phi}}(\hat{\mathbf{x}}_t, t) - \mathbf{y}_{\boldsymbol{\theta}}(\boldsymbol{\xi}, t))\end{aligned}$$

  where $\delta = t - s$ is the step size.

- So, we can train the network with the following loss instead:

$$\mathcal{L}^{\mathrm{BS}}_{\boldsymbol{\theta}} = E_{\boldsymbol{\xi} \sim \mathcal{N}(\mathbf{0}, I), t \sim [\delta, T]} \bigg[\frac{\tilde{w}_t}{\delta^2} \Big\| \mathbf{y}_{\boldsymbol{\theta}}(\boldsymbol{\xi}, s) - \mathtt{SG}\big[\mathbf{y}_{\boldsymbol{\theta}}(\boldsymbol{\xi}, t) + \delta\lambda'_t(\mathbf{f}_{\boldsymbol{\phi}}(\hat{\mathbf{x}}_t, t) - \mathbf{y}_{\boldsymbol{\theta}}(\boldsymbol{\xi}, t))\big] \Big\|^2 \bigg].$$

  where $\mathtt{SG}[\cdot]$ is the stop-gradient operator, and $\tilde{w}_t$ is a time-dependent weight, which is implicit in how time t is sampled.

- A challenge in training $\mathbf{y}_{\boldsymbol{\theta}}$ error accumulation: errors in prediction of $\mathbf{y}_t$ might propagate to prediction of $\mathbf{y}_s$ with $s < t$.

- The paper proposes two ways to mitigate error accumulation.
  - Sample time t uniformly, despite potential slow down in convergence.
  - Use higher-order solvers such as Heun's method to compute the expected value of $\mathbf{y}_{\boldsymbol{\theta}}(\boldsymbol{\xi}, s)$ instead of just using the Euler's method like in $\mathcal{L}_{\boldsymbol{\theta}}^{\mathrm{BS}}$.
- Another problem that must be address is numerical issues caused by the fact that λ_t' is unbounded at $t = T$ and $t = 0$.
  - The student model must be trained in the truncated range $t \in [t_{\min}, t_{\max}]$.
  - We must also ensure that the student model behaves in the same way as the teacher model at $t_{\max}$. This is done by minimizing the loss:

$$\mathcal{L}_{\boldsymbol{\theta}}^{\mathrm{BC}} = E_{\boldsymbol{\xi} \sim \mathcal{N}(\mathbf{0}, I)} \Big[\| \mathbf{f}_{\boldsymbol{\phi}}(\boldsymbol{\xi}, t_{\max}) - \mathbf{y}_{\boldsymbol{\theta}}(\boldsymbol{\xi}, t_{\max}) \|^2 \Big]$$

- As a result, the overall training loss for BOOT is $\mathcal{L}_{\boldsymbol{\theta}} = \mathcal{L}_{ves\theta}^{\mathrm{BS}} + \beta \mathcal{L}_{\boldsymbol{\theta}}^{\mathrm{BC}}$ where β is a hyperparameter.

3.3 Adapting to Guided Diffusion Models

- Conditional diffusion models $\mathbf{f}_{\boldsymbol{\phi}}(\mathbf{x}_t, t, \mathbf{c})$ are often trained to be able to perform classifier-free guidance:

$$\tilde{\mathbf{f}}_{\boldsymbol{\phi}}(\mathbf{x}_t, t, \mathbf{c}) = \mathbf{f}_{\boldsymbol{\phi}}(\mathbf{x}_t, t, \emptyset) + w\big(\mathbf{f}_{\boldsymbol{\phi}}(\mathbf{x}_t, t, \mathbf{c}) - \mathbf{f}_{\boldsymbol{\phi}}(\mathbf{x}_t, t, \emptyset)\big)$$

  where w is the guidance scale, and $\emptyset$ denotes a conditioning input that makes the model unconditional.
- It is straightforward to train a (conditional) student model that follows the behavior of a teacher model with classifier free guidance. We can either
  - Fix the guidance scale w and use $\tilde{\mathbf{f}}_{\boldsymbol{\phi}}(\mathbf{x}_t, t, \mathbf{c})$.
  - Like Meng et al. [MRG+23], train a student model that also receives w as input. (However, this requires architecture change.)

4 Experiments

- The authors performed experiments on the following datasets:
  - FFHQ 64×64
  - Class-conditional ImageNet 64×64.
  - LSUN Bedroom 256×256.
- For class-conditional ImageNet, the paper evaluates the student model on random guidance scale in the range $w \in [1, 5]$.
- The authors also distilled two text-to-image models: DeepFloyd-IF and Stable Diffusion.
  - Text prompts were obtrained from DiffusionDB [WMM+23].
- In most experiments, the student model $\mathbf{y}_\theta$ would have the same architecture as the teacher model $\mathbf{f}_\theta$, and the parameters of the student model would be iniitlized to those of the teacher models.
- The exception to the above practice is when the model must also be conditioned on the guidance scale w (i.e., class-conditonal ImageNet). When this is the case, w is taken into account via ADAIN layers.

- When looking at FID scores on the three datasets, BOOT-distilled models does not perform as well as 50-step DDIM sampler but better than 10-step DDIM sampler.
- Without the boundary condition loss, the student model's outputs are consistently sharp across time steps, indicating mode collapse.
- We may train a BOOT model by progressively decreasing time during training. However, the paper found that this progressive-time scheme leads to more artifacts. The authors surmise that progressive-time training tends to accumulate irreversible errors.

References

[BAL+23] David Berthelot, Arnaud Autef, Jierui Lin, Dian Ang Yap, Shuangfei Zhai, Siyuan Hu, Daniel Zheng, Walter Talbott, and Eric Gu. Tract: Denoising diffusion models with transitive closure time-distillation, 2023.

[GZZ+23] Jiatao Gu, Shuangfei Zhai, Yizhe Zhang, Lingjie Liu, and Josh Susskind. Boot: Data-free distillation of denoising diffusion models with bootstrapping, 2023.

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

[LL21] Eric Luhman and Troy Luhman. Knowledge distillation in iterative generative models for improved sampling speed, 2021.

[MRG+23] Chenlin Meng, Robin Rombach, Ruiqi Gao, Diederik P. Kingma, Stefano Ermon, Jonathan Ho, and Tim Salimans. On distillation of guided diffusion models, 2023.

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

[SH22] Tim Salimans and Jonathan Ho. Progressive distillation for fast sampling of diffusion models. *CoRR*, abs/2202.00512, 2022.

[SME20] Jiaming Song, Chenlin Meng, and Stefano Ermon. Denoising diffusion implicit models, 2020.

[SSDK+21] Yang Song, Jascha Sohl-Dickstein, Diederik P Kingma, Abhishek Kumar, Stefano Ermon, and Ben Poole. Score-based generative modeling through stochastic differential equations. In *International Conference on Learning Representations*, 2021.

[WMM+23] Zijie J. Wang, Evan Montoya, David Munechika, Haoyang Yang, Benjamin Hoover, and Duen Horng Chau. Diffusiondb: A large-scale prompt gallery dataset for text-to-image generative models, 2023.