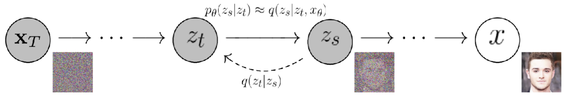
Data-free Distillation of Diffusion Models with Bootstrapping

Jiatao Gu, Chen Wang, Shuangfei Zhai, Yizhe Zhang, Lingjie Liu, Josh Susskind
ICML 2024 · Apple Inc., UPenn

Introduction and Motivation

TLDR: We introduce a distillation method that can reduce diffusion models into one-step inference without using any training data.

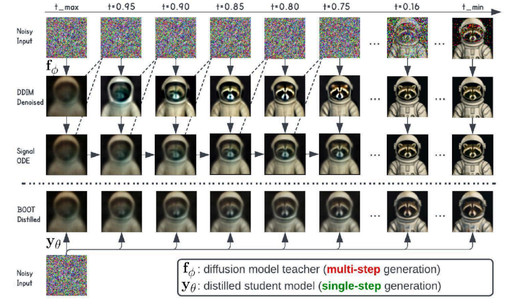
The reverse process of diffusion models.



DDIM Sampling: multiple step denoising

$$\boldsymbol{x}_s = (\sigma_s/\sigma_t)\,\boldsymbol{x}_t + (\alpha_s - \alpha_t\sigma_s/\sigma_t)\,\boldsymbol{f}_\phi(\boldsymbol{x}_t, t), \quad s < t,$$

Our method: predict image from random noise **in one step** with $\mathbf{y}_\theta$

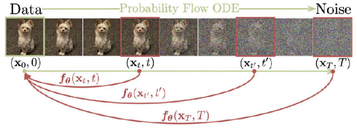


Problems with Existing Distillation Methods

1). Direct distillation approaches need to generate noise-image targets through multi-step DDIM sampling, which is expensive.

$$\mathcal{L}_\theta^{\text{Direct}} = \mathbb{E}_{\boldsymbol{\epsilon}\sim\mathcal{N}(0,I)} \|\boldsymbol{g}_\theta(\boldsymbol{\epsilon}) - \texttt{ODE-Solver}(\boldsymbol{f}_\phi, \boldsymbol{\epsilon}, T \to 0)\|_2^2$$

2). Consistency models removed the requirements of synthetic dataset by learning from "bootstrapping", but they need to access the real data and requires EMA to avoid trivial solutions.



Our Method

Overview

We introduce singal-ODE, a modeling technique focused exclusively on signals and its corresponding distillation process. We also adopt our method on text-to-image diffusion models.

Signal-ODE

In this work, we propose to learn from a "signal-ODE" derived from the original diffusion model by predict the "signal" part of the original DDIM path:

$$\boldsymbol{y}_t = (\boldsymbol{x}_t - \sigma_t\boldsymbol{\epsilon})/\alpha_t$$

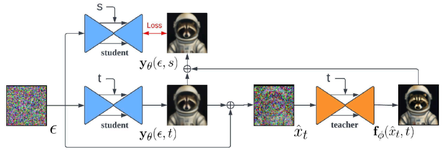
We can rewrite DDIM sampling as:

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

$$\frac{\mathrm{d}\boldsymbol{y}_t}{\mathrm{d}t} = -\lambda_t' \cdot (\boldsymbol{f}_\phi(\boldsymbol{x}_t, t) - \boldsymbol{y}_t) \quad \text{Signal-ODE}$$

Learning with Bootstrapping

The training object is to match the student output at two different timesteps with the help of teacher ODE solvers. Through bootstrapping, the student model can learn to generate samples from noise directly.



Matching two sides of the signal ODE:

$$\mathcal{L}_\theta^{\text{DE}} = \mathbb{E}_{\boldsymbol{\epsilon}\sim\mathcal{N}(0,I), t\sim[0,T]} \left\| \frac{\mathrm{d}\boldsymbol{y}_\theta(\boldsymbol{\epsilon}, t)}{\mathrm{d}t} + \lambda_t' \cdot (\boldsymbol{f}_\phi(\hat{\boldsymbol{x}}_t, t) - \boldsymbol{y}_\theta(\boldsymbol{\epsilon}, t)) \right\|_2^2$$

Taking finite difference for the ODE:

$$\begin{aligned}
\mathcal{L}_\theta &= \mathbb{E}_{\boldsymbol{\epsilon},t}\left[\tilde{\omega}_t \left\| \frac{\mathrm{d}\boldsymbol{y}_\theta(\boldsymbol{\epsilon}, t)}{\mathrm{d}t} + \lambda_t' \cdot (\boldsymbol{f}_\phi(\hat{\boldsymbol{x}}_t, t) - \boldsymbol{y}_\theta(\boldsymbol{\epsilon}, t)) \right\|_2^2\right] \\
&\approx \mathbb{E}_{\boldsymbol{\epsilon},t}\left[\tilde{\omega}_t \| \frac{\boldsymbol{y}_\theta(\boldsymbol{\epsilon}, s) - \boldsymbol{y}_\theta(\boldsymbol{\epsilon}, t)}{\delta} - \lambda_t' (\boldsymbol{f}_\phi(\hat{\boldsymbol{x}}_t, t) - \boldsymbol{y}_\theta(\boldsymbol{\epsilon}, t)) \|_2^2\right] \\
&= \mathbb{E}_{\boldsymbol{\epsilon},t}\left[\frac{\tilde{\omega}_t}{\delta^2} \|\boldsymbol{y}_\theta(\boldsymbol{\epsilon}, s) - \text{SG}\left[\boldsymbol{y}_\theta(\boldsymbol{\epsilon}, t) + \delta\lambda_t' \underbrace{(\boldsymbol{f}_\phi(\hat{\boldsymbol{x}}_t, t) - \boldsymbol{y}_\theta(\boldsymbol{\epsilon}, t))}_{\text{incremental improvement}}\right] \|_2^2\right]
\end{aligned}$$

No ema needed

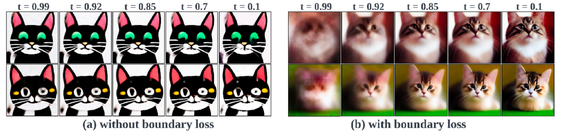
Error Accumulation

Imperfect predictions at large time steps might propagate to subsequent timesteps during bootstrapping. We employ two methods: (1) we sample timesteps uniformly throughout training; (2) We use high-order solvers such as Heun's Method.

Boundary Condition

In theory, the boundary can be arbitrary values since $\alpha_T = 0$. We make the student model learn in a truncated range $t \in [t_{\min}, t_{\max}]$ and add an auxiliary boundary loss:

$$\mathcal{L}_\theta^{\text{BC}} = \mathbb{E}_{\boldsymbol{\epsilon}\sim\mathcal{N}(0,I)}\left[\|\boldsymbol{f}_\phi(\boldsymbol{\epsilon}, t_{\max}) - \boldsymbol{y}_\theta(\boldsymbol{\epsilon}, t_{\max})\|_2^2\right]$$



(a) without boundary loss (b) with boundary loss

Distillation of Text-to-Image Models

Our method can be readily applied for distilling conditional diffusion models in either pixel space or latent space.

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

Our Algorithm

Algorithm 1 Distillation using BOOT for Conditional Diffusion Models.

Require: pretrained diffusion model $\boldsymbol{f}_\phi$, initial student parameter from the teacher $\theta \leftarrow \phi$, step size δ, learning rate η, CFG weight w, context dataset $\mathcal{D}$, negative condition $\boldsymbol{n} = \emptyset, t_{\min}, t_{\max}, \beta$.

while not converged **do**
  Sample noise input $\boldsymbol{\epsilon} \sim \mathcal{N}(0, I)$
  Sample context input $\boldsymbol{c} \sim \mathcal{D}$
  Sample $t \sim (t_{\min}, t_{\max}), s = \min(t - \delta, t_{\min}))$
  Compute noise schedule $\alpha_t, \sigma_t, \alpha_s, \sigma_s$
  Compute $\lambda_t' \approx (1 - \frac{\alpha_t\sigma_s}{\sigma_t\alpha_s})/\delta$
  Generate the model predictions:
    $\boldsymbol{y}_t = \boldsymbol{y}_\theta(\boldsymbol{\epsilon}, t, \boldsymbol{c}), \quad \boldsymbol{y}_s = \boldsymbol{y}_\theta(\boldsymbol{\epsilon}, s, \boldsymbol{c}), \quad \boldsymbol{y}_{t_{\max}} = \boldsymbol{y}_\theta(\boldsymbol{\epsilon}, t_{\max}, \boldsymbol{c})$
  Generate the noisy sample $\hat{\boldsymbol{x}}_t = \alpha_t\boldsymbol{y}_t + \sigma_t\boldsymbol{\epsilon}$
  Compute the denoised target:
    $\tilde{\boldsymbol{f}}_t = \boldsymbol{f}_\phi(\hat{\boldsymbol{x}}_t, t, \boldsymbol{n}) + w \cdot (\boldsymbol{f}_\phi(\hat{\boldsymbol{x}}_t, t, \boldsymbol{c}) - \boldsymbol{f}_\phi(\hat{\boldsymbol{x}}_t, t, \boldsymbol{n}))$
    $\tilde{\boldsymbol{f}}_{t_{\max}} = \boldsymbol{f}_\phi(\boldsymbol{\epsilon}, t_{\max}, \boldsymbol{n}) + w \cdot (\boldsymbol{f}_\phi(\boldsymbol{\epsilon}, t_{\max}, \boldsymbol{c}) - \boldsymbol{f}_\phi(\boldsymbol{\epsilon}, t_{\max}, \boldsymbol{n}))$
  Compute the bootstrapping loss $\mathcal{L}_\theta^{\text{BS}} = \frac{1}{(\delta\lambda_t')^2}\|\boldsymbol{y}_s - \text{SG}(\boldsymbol{y}_t + \delta\lambda_t'(\tilde{\boldsymbol{f}}_t - \boldsymbol{y}_t))\|_2^2$
  Compute the boundary loss $\mathcal{L}_\theta^{\text{BC}} = \|\boldsymbol{y}_{t_{\max}} - \tilde{\boldsymbol{f}}_{t_{\max}}\|_2^2$
  Update model parameters $\theta \leftarrow \theta - \eta \cdot \nabla_\theta\left(\mathcal{L}_\theta^{\text{BS}} + \beta\mathcal{L}_\theta^{\text{BC}}\right)$
end while
return Trained model parameters θ

Quantitative Results

Method	NFE	FID
Diffusion Model+Solver		
DDPM (Ho et al., 2020)	1000	3.17
DDIM (Song et al., 2021)	50	4.67
DPM-solver-2 (Lu et al., 2022)	10	5.94
DEIS (Zhang & Chen, 2022)	10	4.17
EDM (Karras et al., 2022)	35	2.04
Distillation		
Direct* (Luhman & Luhman, 2021)	1	9.36
DFNO* (Zheng et al., 2023)	1	4.12
ReFlow* (Liu et al., 2022)	1	6.18
PD (Salimans & Ho, 2022)	1	8.34
CM (Song et al., 2023)	1	3.55
Data-free Distillation		
Ours (L2 loss)	1	6.88
Ours (LPIPS loss)	1	4.38

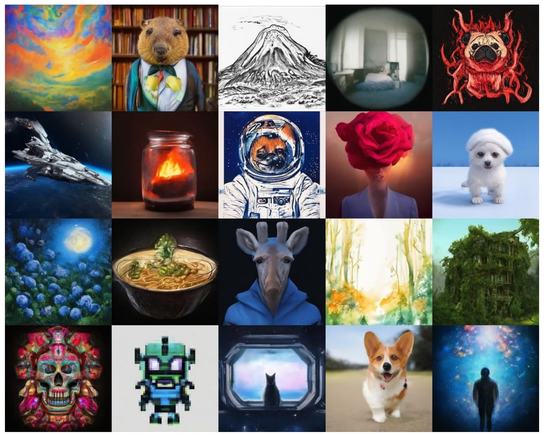
Results on CIFAR-10

	Steps	FFHQ 64 × 64 FID	FFHQ 64 × 64 fps	ImageNet 64 × 64 FID	ImageNet 64 × 64 fps
DDPM	250	5.4	0.2	11.0	0.1
DDIM	50	7.6	1.2	13.7	0.6
	10	18.3	5.3	18.3	3.3
	1	225	54	237	34
Ours	1	9.0	54	12.3	34

Results on FFHQ and ImageNet

Qualitative Results

Comparison with other baselines



Random Samples from our single-step student model distilled from DeepFloyd-IF

Links

SCAN ME

https://machinelearning.apple.com/research/boot