Diffusion model JC
Introduction 1: SDE view

Manuel Gloeckler

02.10.24

Elucidating the Design Space of Diffusion-Based Generative Models

Tero Karras NVIDIA
Miika Aittala NVIDIA
Timo Aila NVIDIA
Samuli Laine NVIDIA

SDE view: Introduction

SDE view: Introduction

- All common diffusion models are based on *linear time varying SDEs*

$$d\mathbf{x} = f(t)\mathbf{x}dt + g(t)d\mathbf{w}_t$$

SDE view: Introduction

- All common diffusion models are based on *linear time varying SDEs*

$$d\mathbf{x} = f(t)\mathbf{x}dt + g(t)d\mathbf{w}_t$$

- Convenient because the perturbation kernels are known

$$p_{0t}(\mathbf{x}(t)|\mathbf{x}(0)) = \mathcal{N}(\mathbf{x}(t); s(t)\mathbf{x}(0), s(t)^2\sigma^2(t)^2 I)$$

$$s(t) = \exp\left(\int_0^t f(t)dt\right) \qquad \sigma^2(t) = \int_0^t g(t)^2/s(t)^2 dt$$

SDE view: Introduction

- All common diffusion models are based on *linear time varying SDEs*

$$d\mathbf{x} = f(t)\mathbf{x}dt + g(t)d\mathbf{w}_t$$

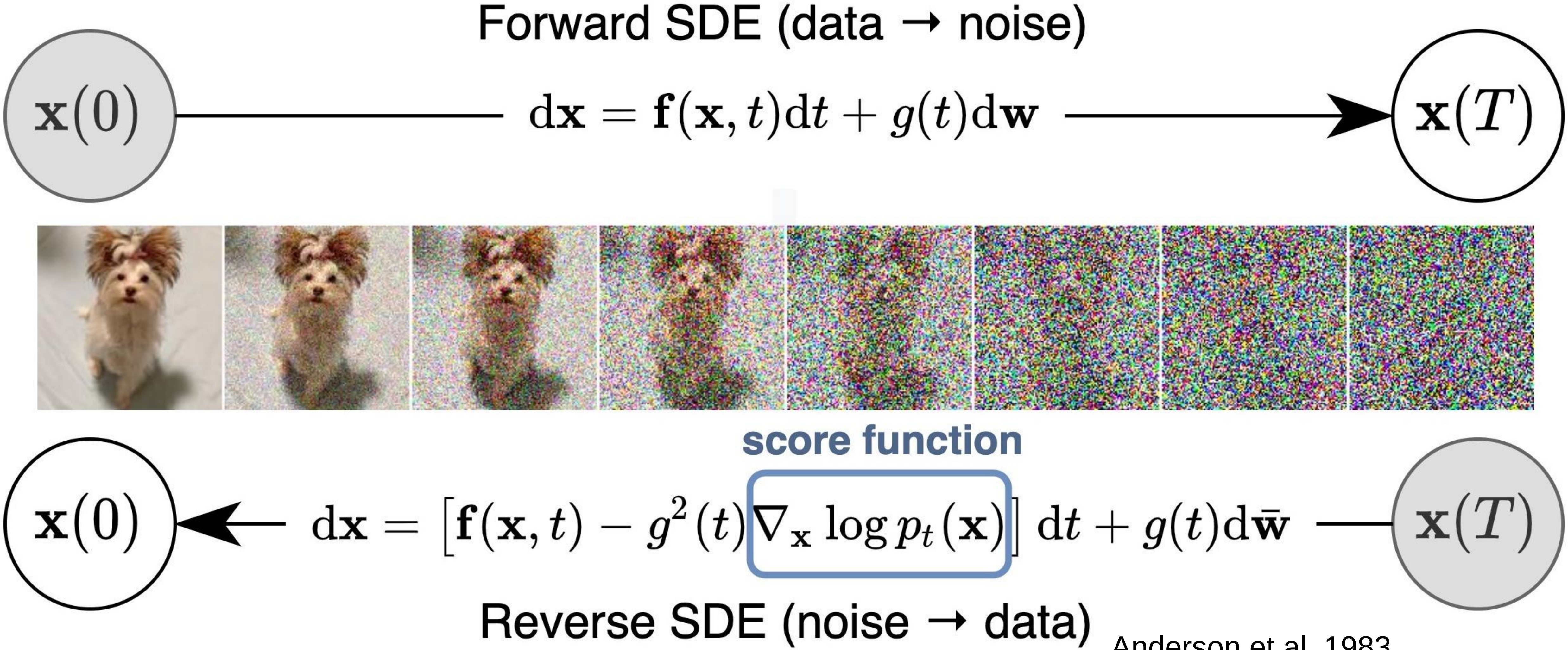
- Convenient because the perturbation kernels are known

$$p_{0t}(\mathbf{x}(t)|\mathbf{x}(0)) = \mathcal{N}(\mathbf{x}(t); s(t)\mathbf{x}(0), s(t)^2\sigma^2(t)^2 I)$$

$$s(t) = \exp\left(\int_0^t f(t)dt\right) \qquad \sigma^2(t) = \int_0^t g(t)^2/s(t)^2 dt$$

- But intractable marginals (in general)

$$\boxed{p_t(\mathbf{x}) = \int p_{0t}(\mathbf{x}|\mathbf{x}_0)p_{data}(\mathbf{x}_0)d\mathbf{x}_0}$$



Anderson et.al. 1983

But all we need for that are the marginals…

- We actually don't care to much about f and g, the marginals are fully defined by the $s(t), \sigma(t)$!
- *More intuitive:*
  - $s(t)$ is how we scale inputs i.e. usually constant or decays to zero.
  - $\sigma(t)$ is the amount of noise we add to the inputs i.e. exploding or preserving.

But all we need for that are the marginals…

- We actually don't care to much about f and g, the marginals are fully defined by the $s(t), \sigma(t)$!
- *More intuitive:*
  - $s(t)$ is how we scale inputs i.e. usually constant or decays to zero.
  - $\sigma(t)$ is the amount of noise we add to the inputs i.e. exploding or preserving.

- **So instead:** Start from $s(t), \sigma(t)$ and derive the rest...

$$f(t) = \dot{s}(t)/s(t) \qquad g(t) = s(t)\sqrt{2\dot{\sigma}(t)\sigma(t)}$$

$$p_t(\mathbf{x}) = s(t)^{-d}\left[p_{data} * \mathcal{N}(0, \sigma^2(t)I)\right](\mathbf{x}/s(t))$$

revere SDE $d\mathbf{x} = (f(t)\mathbf{x} - 0.5g^2(t)\nabla_{\mathbf{x}} \log p_t(x))dt + g(t)d\mathbf{w}_t$

revere ODE $d\mathbf{x} = (f(t)\mathbf{x} - 0.5g^2(t)\nabla_{\mathbf{x}} \log p_t(x))dt$

Song et. al. 2020

revere SDE

$$d\mathbf{x} = (f(t)\mathbf{x} - 0.5g^2(t)\nabla_{\mathbf{x}} \log p_t(x))dt + g(t)d\mathbf{w}_t$$

revere ODE

$$d\mathbf{x} = (f(t)\mathbf{x} - 0.5g^2(t)\nabla_{\mathbf{x}} \log p_t(x))dt$$

Song et. al. 2020

Karras et. al. 2022

revere SDE

$$dx_{\pm} = -\dot{\sigma}(t)\sigma(t)\nabla_x \log p(x; \sigma(t))\, dt \pm \beta(t)\sigma(t)^2\nabla_x \log p(x; \sigma(t))\, dt + \sqrt{2\beta(t)\sigma(t)}\, d\omega_t$$

revere ODE

$$d\mathbf{x} = \left[\frac{\dot{s}(t)}{s(t)}\mathbf{x} - s(t)^2\dot{\sigma}(t)\sigma(t)\nabla_{\mathbf{x}} \log p(\mathbf{x}/s(t), \sigma(t))\right] dt$$

$$d\mathbf{x} = -\dot{\sigma}(t)\sigma(t)\nabla_{\mathbf{x}} \log p(\mathbf{x}, \sigma(t))dt \quad \text{if } s(t) = 1$$

revere SDE $$d\mathbf{x} = (f(t)\mathbf{x} - 0.5g^2(t)\nabla_{\mathbf{x}} \log p_t(x))dt + g(t)d\mathbf{w}_t$$

revere ODE $$d\mathbf{x} = (f(t)\mathbf{x} - 0.5g^2(t)\nabla_{\mathbf{x}} \log p_t(x))dt$$

Song et. al. 2020

Karras et. al. 2022

revere SDE
$$\begin{aligned} dx_{\pm} = -\dot{\sigma}(t)\sigma(t)\nabla_x \log p(x;\sigma(t))\, dt \pm \beta(t)\sigma(t)^2\nabla_x \log p(x;\sigma(t))\, dt \\ +\sqrt{2\beta(t)\sigma(t)}\, d\omega_t \end{aligned}$$

(Appendix B5 i.e. finding a family of SDEs with same marginals using Fokker-Planck equations)

revere ODE
$$d\mathbf{x} = \left[\frac{\dot{s}(t)}{s(t)}\mathbf{x} - s(t)^2\dot{\sigma}(t)\sigma(t)\nabla_{\mathbf{x}} \log p(\mathbf{x}/s(t), \sigma(t))\right] dt$$

$$d\mathbf{x} = -\dot{\sigma}(t)\sigma(t)\nabla_{\mathbf{x}} \log p(\mathbf{x}, \sigma(t))dt \qquad \text{if } s(t) = 1$$

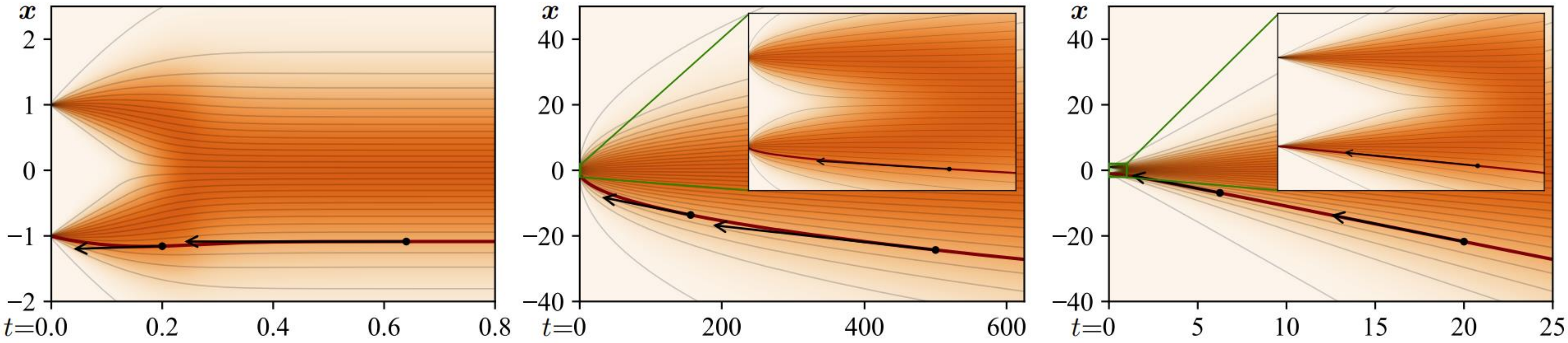
(Appendix B1, B2 plugin in the new expressions for drift and diffusion in previous reverse ODE)

SDEs considered

		VP [49]	VE [49]	iDDPM [37] + DDIM [47]	Ours ("EDM")
Schedule	$\sigma(t)$	$\sqrt{e^{\frac{1}{2}\beta_\mathrm{d}t^2+\beta_\mathrm{min}t}-1}$	$\sqrt{t}$	t	t
Scaling	$s(t)$	$1/\sqrt{e^{\frac{1}{2}\beta_\mathrm{d}t^2+\beta_\mathrm{min}t}}$	1	1	1

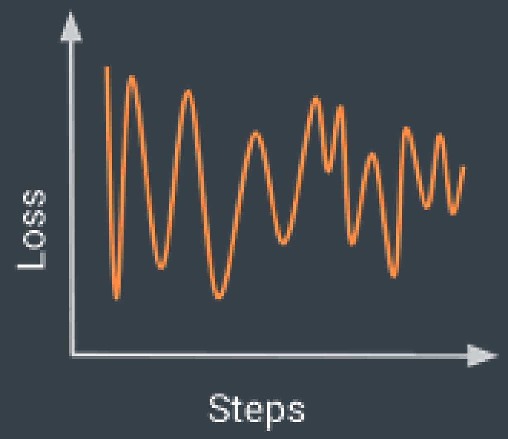
SDEs considered

		VP [49]	VE [49]	iDDPM [37] + DDIM [47]	Ours ("EDM")
Schedule	$\sigma(t)$	$\sqrt{e^{\frac{1}{2}\beta_\mathrm{d}t^2+\beta_\mathrm{min}t}-1}$	$\sqrt{t}$	t	t
Scaling	$s(t)$	$1/\sqrt{e^{\frac{1}{2}\beta_\mathrm{d}t^2+\beta_\mathrm{min}t}}$	1	1	1



(a) Variance preserving ODE [49] (b) Variance exploding ODE [49] (c) DDIM [47] / Our ODE

Training diffusion models…



Problem with training diffusion models

- *Multitask learning* problem: We need a denoiser for each noise level

$$\mathcal{L}(D) = \mathbb{E}_{y \sim p_{\text{data}}} \mathbb{E}_{n \sim \mathcal{N}(0, \sigma^2 I)} \|D(y+n;\sigma) - y\|_2^2$$

- **Problem**: We actually don't care about the denoise but the score

$$\nabla_x \log p(x;\sigma) = \frac{(D(x;\sigma) - x)}{\sigma^2}$$

As $\sigma \rightarrow 0$ we need a **much** more accurate denoise at it is scale !

As $\sigma \rightarrow \infty$ we don't need to learn anything…

Problem with training diffusion models

- *Multitask learning* problem: We need a denoiser for each noise level

$$\mathcal{L}(D) = \mathbb{E}_{y\sim p_{\mathrm{data}}}\mathbb{E}_{n\sim\mathcal{N}(0,\sigma^2 I)}\|D(y+n;\sigma)-y\|_2^2$$

- **Problem**: We actually don't care about the denoise but the score

$$\nabla_x \log p(x;\sigma) = \frac{(D(x;\sigma)-x)}{\sigma^2}$$

Tweedie's formula
Robbins et.al. 1956

$$\mathbb{E}[x|\tilde{x}] = \tilde{x} + \sigma^2\nabla_{\tilde{x}}\log p(\tilde{x})$$

As $\sigma \to 0$ we need a **much** more accurate denoise at it is scale !

As $\sigma \to \infty$ we don't need to learn anything…

Problem with training diffusion models

- Actual denoising score matching loss

$$\mathcal{L}(S) = \mathbb{E}_{y \sim p_{data}} \mathbb{E}_{n \sim \mathcal{N}(0, \sigma^2 I)} \left[||S(y+n, \sigma) - \frac{n}{\sigma}||_2^2 \right]$$

Problem with training diffusion models

- Actual denoising score matching loss

$$\mathcal{L}(S) = \mathbb{E}_{y \sim p_{data}} \mathbb{E}_{n \sim \mathcal{N}(0, \sigma^2 I)} \left[||S(y+n, \sigma) - \frac{n}{\sigma}||_2^2 \right]$$

- **Problem**: Hard to optimize

  As $\sigma \to 0$ variance explodes to infintiy

  As $\sigma \to \infty$ we don't need to learn anything…

Problem with training diffusion models

- Actual denoising score matching loss

$$\mathcal{L}(S) = \mathbb{E}_{y\sim p_{data}} \mathbb{E}_{n\sim\mathcal{N}(0,\sigma^2 I)} \left[||S(y+n,\sigma) - \frac{n}{\sigma}||_2^2 \right]$$

- **Problem**: Hard to optimize

  As $\sigma \to 0$ variance explodes to infintiy

  As $\sigma \to \infty$ we don't need to learn anything…

- Usually "noise matching":

Problem with training diffusion models

- Actual denoising score matching loss

$$\mathcal{L}(S) = \mathbb{E}_{y \sim p_{data}} \mathbb{E}_{n \sim \mathcal{N}(0, \sigma^2 I)} \left[||S(y+n, \sigma) - \frac{n}{\sigma}||_2^2 \right]$$

- **Problem**: Hard to optimize

  As $\sigma \rightarrow 0$ variance explodes to infintiy

  As $\sigma \rightarrow \infty$ we don't need to learn anything…

- Usually "noise matching": $\mathcal{L}(S) = \mathbb{E}_y \mathbb{E}_n \left[\frac{1}{\sigma^2} ||\sigma S(y+n, \sigma) - n||_2^2 \right]$

Problem with training diffusion models

- Actual denoising score matching loss

$$\mathcal{L}(S) = \mathbb{E}_{y \sim p_{data}} \mathbb{E}_{n \sim \mathcal{N}(0, \sigma^2 I)} \left[||S(y+n, \sigma) - \frac{n}{\sigma}||_2^2 \right]$$

- **Problem**: Hard to optimize

  As $\sigma \to 0$ variance explodes to infintiy

  As $\sigma \to \infty$ we don't need to learn anything…

- Usually "noise matching": $\mathcal{L}(S) = \mathbb{E}_y \mathbb{E}_n \left[\frac{1}{\sigma^2} ||\sigma S(y+n, \sigma) - n||_2^2 \right]$

NOTE: Problem of *denoising* score matching, not score matching in general
Score matching (Hyvärin, 2005), Sliced Score matching (Song et. al. 2019)

$$\underbrace{D_\theta(x;\sigma)}_{\text{Denoiser}} = c_{skip}(\sigma)x + c_{out}(\sigma)F_\theta(c_{in}(\sigma)x; c_{noise}(\sigma))$$

$$\underbrace{D_\theta(x;\sigma)}_{\text{Denoiser}} = c_{skip}(\sigma)x + c_{out}(\sigma)F_\theta(c_{in}(\sigma)x; c_{noise}(\sigma))$$

$$\mathcal{L}(\theta) = \mathbb{E}_{\sigma,y,n}\left[\underbrace{\lambda(\sigma)c_{\text{out}}(\sigma)^2}_{\text{effective weight}} \| \underbrace{F_\theta(c_{\text{in}}(\sigma)\cdot(y+n); c_{\text{noise}}(\sigma))}_{\text{network output}} - \underbrace{\frac{1}{c_{\text{out}}(\sigma)}(y - c_{\text{skip}}(\sigma)\cdot(y+n))}_{\text{effective training target}} \|_2^2\right]$$

$$\underbrace{D_\theta(x;\sigma)}_{\text{Denoiser}} = c_{skip}(\sigma)x + c_{out}(\sigma)F_\theta(\underbrace{c_{in}(\sigma)x}_{\text{Should be standardized}}; c_{noise}(\sigma))$$

$$c_{\mathrm{in}}(\sigma) = \frac{1}{\sqrt{\sigma^2 + \sigma_{\mathrm{data}}^2}}$$

$$\mathcal{L}(\theta) = \mathbb{E}_{\sigma,y,n}\left[\underbrace{\lambda(\sigma)c_{\mathrm{out}}(\sigma)^2}_{\text{effective weight}} \parallel \underbrace{F_\theta(c_{\mathrm{in}}(\sigma)\cdot(y+n); c_{\mathrm{noise}}(\sigma))}_{\text{network output}} - \underbrace{\frac{1}{c_{\mathrm{out}}(\sigma)}\left(y - c_{\mathrm{skip}}(\sigma)\cdot(y+n)\right)}_{\text{effective training target}} \parallel_2^2\right]$$

$$\underbrace{D_\theta(x;\sigma)}_{\text{Denoiser}} = c_{skip}(\sigma)x + \underbrace{c_{out}(\sigma)}_{}F_\theta(\underbrace{c_{in}(\sigma)}_{}x; c_{noise}(\sigma))$$

Denoiser

Effective target should have **variance of one**

Should be standardized

$$c_{\text{in}}(\sigma) = \frac{1}{\sqrt{\sigma^2 + \sigma^2_{\text{data}}}}$$

$$c_{\text{out}}(\sigma)^2 = (1 - c_{\text{skip}}(\sigma))^2\,\sigma^2_{\text{data}} + c_{\text{skip}}(\sigma)^2\sigma^2$$

$$\mathcal{L}(\theta) = \mathbb{E}_{\sigma,y,n}\left[\underbrace{\lambda(\sigma)c_{\text{out}}(\sigma)^2}_{\text{effective weight}} \| \underbrace{F_\theta(c_{\text{in}}(\sigma)\cdot(y+n); c_{\text{noise}}(\sigma))}_{\text{network output}} - \boxed{\underbrace{\frac{1}{c_{\text{out}}(\sigma)}(y - c_{\text{skip}}(\sigma)\cdot(y+n))}_{\text{effective training target}}} \|_2^2\right]$$

$$D_\theta(x;\sigma) = c_{skip}(\sigma)x + c_{out}(\sigma)F_\theta(c_{in}(\sigma)x; c_{noise}(\sigma))$$

Denoiser

Minimize c_{out}

$$c_{\text{skip}}(\sigma) = \frac{\sigma_{\text{data}}^2}{\sigma^2 + \sigma_{\text{data}}^2}$$

Effective target should have **variance of one**

Should be standardized

$$c_{\text{in}}(\sigma) = \frac{1}{\sqrt{\sigma^2 + \sigma_{\text{data}}^2}}$$

$$c_{\text{out}}(\sigma)^2 = (1 - c_{\text{skip}}(\sigma))^2\,\sigma_{\text{data}}^2 + c_{\text{skip}}(\sigma)^2\sigma^2$$

$$\mathcal{L}(\theta) = \mathbb{E}_{\sigma,y,n}\left[\underbrace{\lambda(\sigma)c_{\text{out}}(\sigma)^2}_{\text{effective weight}}\,\|\,\underbrace{F_\theta(c_{\text{in}}(\sigma)\cdot(y+n); c_{\text{noise}}(\sigma))}_{\text{network output}} - \underbrace{\frac{1}{c_{\text{out}}(\sigma)}\,(y - c_{\text{skip}}(\sigma)\cdot(y+n))}_{\text{effective training target}}\,\|_2^2\right]$$

$$\underbrace{D_\theta(x;\sigma)}_{\text{Denoiser}} = \underbrace{c_{skip}(\sigma)}_{\text{Minimize } c_{out}} x + \underbrace{c_{out}(\sigma)}_{\textbf{Effective target} \text{ should have } \textbf{variance of one}} F_\theta(\underbrace{c_{in}(\sigma)x}_{\text{Should be standardized}}; c_{noise}(\sigma))$$

$$c_{\text{skip}}(\sigma) = \frac{\sigma^2_{\text{data}}}{\sigma^2 + \sigma^2_{\text{data}}}$$

$$c_{\text{out}}(\sigma) = \frac{\sigma \cdot \sigma_{\text{data}}}{\sqrt{\sigma^2 + \sigma^2_{\text{data}}}}$$

$$c_{\text{in}}(\sigma) = \frac{1}{\sqrt{\sigma^2 + \sigma^2_{\text{data}}}}$$

$$\mathcal{L}(\theta) = \mathbb{E}_{\sigma,y,n}\left[\underbrace{\lambda(\sigma)c_{\text{out}}(\sigma)^2}_{\text{effective weight}} \| \underbrace{F_\theta(c_{\text{in}}(\sigma)\cdot(y+n); c_{\text{noise}}(\sigma))}_{\text{network output}} - \underbrace{\frac{1}{c_{\text{out}}(\sigma)}(y - c_{\text{skip}}(\sigma)\cdot(y+n))}_{\text{effective training target}} \|_2^2\right]$$

$$\underbrace{D_\theta(x;\sigma)}_{\text{Denoiser}} = \underbrace{c_{skip}(\sigma)}_{\text{Minimize } c_{out}} x + \underbrace{c_{out}(\sigma)}_{\text{Effective target should have variance of one}} F_\theta(\underbrace{c_{in}(\sigma)}_{\text{Should be standardized}} x; \underbrace{c_{noise}(\sigma)}_{\text{Kinda arbirary}}))$$

Denoiser

Minimize c_{out}

$$c_{\text{skip}}(\sigma) = \frac{\sigma_{\text{data}}^2}{\sigma^2 + \sigma_{\text{data}}^2}$$

Effective target should have **variance of one**

$$c_{\text{out}}(\sigma) = \frac{\sigma \cdot \sigma_{\text{data}}}{\sqrt{\sigma^2 + \sigma_{\text{data}}^2}}$$

Should be standardized

$$c_{\text{in}}(\sigma) = \frac{1}{\sqrt{\sigma^2 + \sigma_{\text{data}}^2}}$$

Kinda arbirary

Effective target weight of **one**

$$\lambda(\sigma) = \frac{(\sigma^2 + \sigma_{\text{data}}^2)}{(\sigma \cdot \sigma_{\text{data}})^2}$$

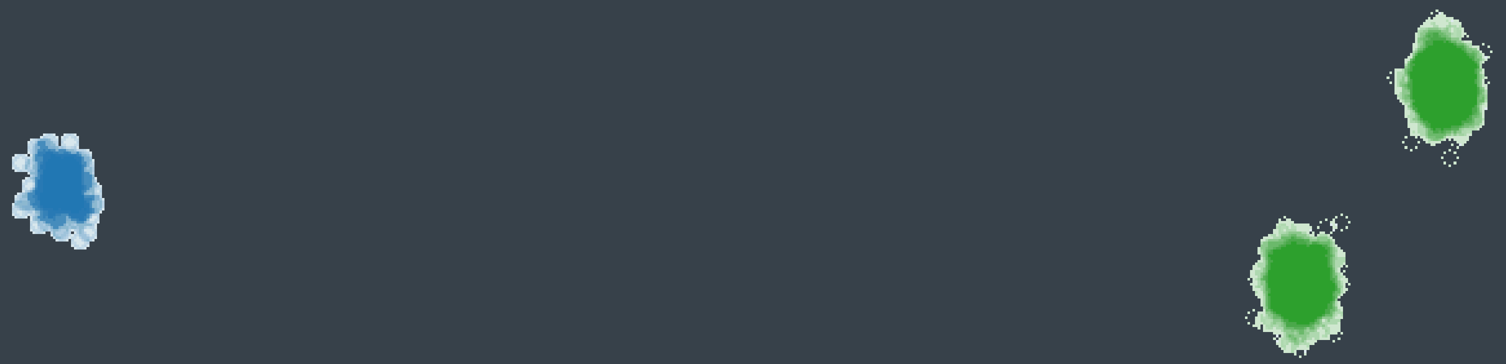
$$\mathcal{L}(\theta) = \mathbb{E}_{\sigma,y,n}\left[\underbrace{\lambda(\sigma) c_{\text{out}}(\sigma)^2}_{\text{effective weight}} \left\| \underbrace{F_\theta(c_{\text{in}}(\sigma)\cdot(y+n); c_{\text{noise}}(\sigma))}_{\text{network output}} - \underbrace{\frac{1}{c_{\text{out}}(\sigma)}(y - c_{\text{skip}}(\sigma)\cdot(y+n))}_{\text{effective training target}} \right\|_2^2\right]$$

Improve training…

	VP [49]	VE [49]	iDDPM [37] + DDIM [47]	Ours ("EDM")
Network and preconditioning (Section 5)				
Architecture of F_θ	DDPM++	NCSN++	DDPM	(any)
Skip scaling $c_{\text{skip}}(\sigma)$	1	1	1	$\sigma^2_{\text{data}} / \left(\sigma^2 + \sigma^2_{\text{data}}\right)$
Output scaling $c_{\text{out}}(\sigma)$	$-\sigma$	σ	$-\sigma$	$\sigma \cdot \sigma_{\text{data}} / \sqrt{\sigma^2_{\text{data}} + \sigma^2}$
Input scaling $c_{\text{in}}(\sigma)$	$1/\sqrt{\sigma^2+1}$	1	$1/\sqrt{\sigma^2+1}$	$1/\sqrt{\sigma^2 + \sigma^2_{\text{data}}}$
Noise cond. $c_{\text{noise}}(\sigma)$	$(M-1)\,\sigma^{-1}(\sigma)$	$\ln(\frac{1}{2}\sigma)$	$M-1-\arg\min_j \lvert u_j - \sigma \rvert$	$\frac{1}{4}\ln(\sigma)$
Training (Section 5)				
Noise distribution	$\sigma^{-1}(\sigma) \sim \mathcal{U}(\epsilon_t, 1)$	$\ln(\sigma) \sim \mathcal{U}(\ln(\sigma_{\text{min}}), \ln(\sigma_{\text{max}}))$	$\sigma = u_j,\ \ j \sim \mathcal{U}\{0, M-1\}$	$\ln(\sigma) \sim \mathcal{N}(P_{\text{mean}}, P^2_{\text{std}})$
Loss weighting $\lambda(\sigma)$	$1/\sigma^2$	$1/\sigma^2$	$1/\sigma^2$ (note: *)	$\left(\sigma^2 + \sigma^2_{\text{data}}\right) / (\sigma \cdot \sigma_{\text{data}})^2$

	CIFAR-10 [29] at 32×32 Conditional		CIFAR-10 [29] at 32×32 Unconditional		FFHQ [27] 64×64 Unconditional		AFHQv2 [7] 64×64 Unconditional	
Training configuration	VP	VE	VP	VE	VP	VE	VP	VE
A Baseline [49] (*pre-trained)	2.48	3.11	3.01*	3.77*	3.39	25.95	2.58	18.52
B + Adjust hyperparameters	2.18	2.48	2.51	2.94	3.13	22.53	2.43	23.12
C + Redistribute capacity	2.08	2.52	2.31	2.83	2.78	41.62	2.54	15.04
D + Our preconditioning	2.09	2.64	2.29	3.10	2.94	3.39	2.79	3.81
E + Our loss function	1.88	1.86	2.05	1.99	2.60	2.81	2.29	2.28
F + Non-leaky augmentation	**1.79**	**1.79**	**1.97**	**1.98**	**2.39**	**2.53**	**1.96**	**2.16**
NFE	35	35	35	35	79	79	79	79

Sampling diffusion models…



Deterministic sampling

- One extra evaluation for $O\left(dt^3\right)$ local error

Algorithm 1 Deterministic sampling using Heun's 2nd order method with arbitrary $\sigma(t)$ and $s(t)$.

procedure HEUNSAMPLER($D_\theta(\boldsymbol{x};\sigma),\ \sigma(t),\ s(t),\ t_{i\in\{0,\ldots,N\}}$)	
sample $\boldsymbol{x}_0 \sim \mathcal{N}\big(\mathbf{0},\ \sigma^2(t_0)\ s^2(t_0)\ \mathbf{I}\big)$	▷ Generate initial sample at t_0
for $i \in \{0,\ldots,N-1\}$ **do**	▷ Solve Eq. 4 over N time steps
$\boldsymbol{d}_i \leftarrow \left(\frac{\dot\sigma(t_i)}{\sigma(t_i)} + \frac{\dot s(t_i)}{s(t_i)}\right)\boldsymbol{x}_i - \frac{\dot\sigma(t_i)s(t_i)}{\sigma(t_i)} D_\theta\left(\frac{\boldsymbol{x}_i}{s(t_i)}; \sigma(t_i)\right)$	▷ Evaluate $\mathrm{d}\boldsymbol{x}/\mathrm{d}t$ at t_i
$\boldsymbol{x}_{i+1} \leftarrow \boldsymbol{x}_i + (t_{i+1} - t_i)\boldsymbol{d}_i$	▷ Take Euler step from t_i to t_{i+1}
if $\sigma(t_{i+1}) \neq 0$ **then**	▷ Apply 2nd order correction unless σ goes to zero
$\boldsymbol{d}'_i \leftarrow \left(\frac{\dot\sigma(t_{i+1})}{\sigma(t_{i+1})} + \frac{\dot s(t_{i+1})}{s(t_{i+1})}\right)\boldsymbol{x}_{i+1} - \frac{\dot\sigma(t_{i+1})s(t_{i+1})}{\sigma(t_{i+1})} D_\theta\left(\frac{\boldsymbol{x}_{i+1}}{s(t_{i+1})}; \sigma(t_{i+1})\right)$	▷ Eval. $\mathrm{d}\boldsymbol{x}/\mathrm{d}t$ at t_{i+1}
$\boldsymbol{x}_{i+1} \leftarrow \boldsymbol{x}_i + (t_{i+1} - t_i)\left(\tfrac{1}{2}\boldsymbol{d}_i + \tfrac{1}{2}\boldsymbol{d}'_i\right)$	▷ Explicit trapezoidal rule at t_{i+1}
return $\boldsymbol{x}_N$	▷ Return noise-free sample at t_N

$$t_i = \sigma^{-1}(\sigma_i) \qquad \sigma_{i<N} = \left({\sigma_{\max}}^{\frac{1}{\rho}} + \tfrac{i}{N-1}\left({\sigma_{\min}}^{\frac{1}{\rho}} - {\sigma_{\max}}^{\frac{1}{\rho}}\right)\right)^{\rho} \quad \text{and} \quad \sigma_N = 0.$$

step size should decrease with σ

Sophisticated high order samplers are not worth the cost…

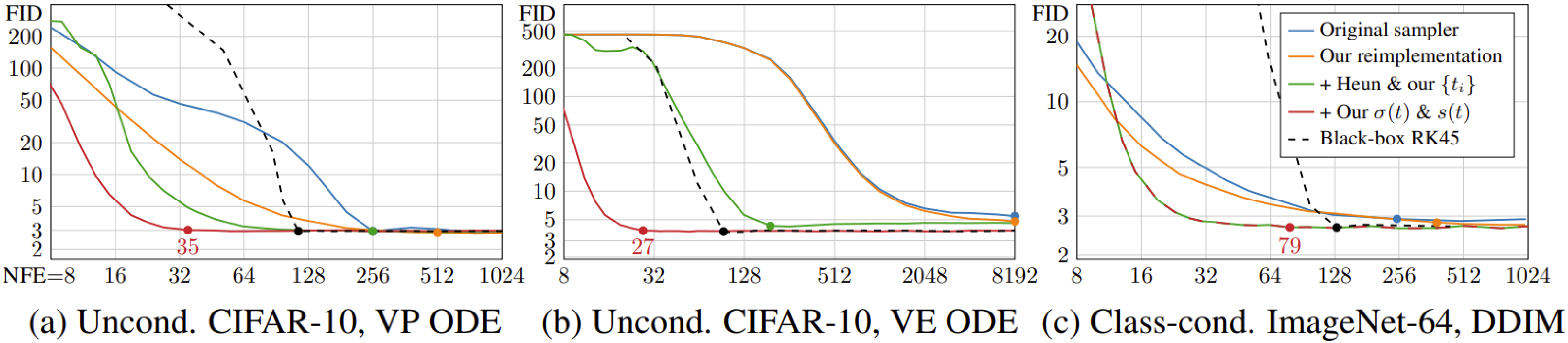


Figure 2: Comparison of deterministic sampling methods using three pre-trained models. For each curve, the dot indicates the lowest NFE whose FID is within 3% of the lowest observed FID.

Stochastic sampler

- Not a SDE solver, but an ODE integrator with Langevin-like steps according to some schedule.

Algorithm 2 Our stochastic sampler with $\sigma(t) = t$ and $s(t) = 1$.

procedure $\textsc{StochasticSampler}(D_\theta(\boldsymbol{x};\sigma),\ t_{i\in\{0,\ldots,N\}},\ \gamma_{i\in\{0,\ldots,N-1}\},\ S_{\text{noise}})$

  sample $\boldsymbol{x}_0 \sim \mathcal{N}(\mathbf{0},\ t_0^2\ \mathbf{I})$

  for $i \in \{0, \ldots, N-1\}$ **do** $\quad \triangleright\ \gamma_i = \begin{cases} \min\left(\frac{S_{\text{churn}}}{N}, \sqrt{2}-1\right) & \text{if } t_i \in [S_{\text{tmin}}, S_{\text{tmax}}] \\ 0 & \text{otherwise} \end{cases}$

    sample $\boldsymbol{\epsilon}_i \sim \mathcal{N}(\mathbf{0},\ S_{\text{noise}}^2\ \mathbf{I})$

    $\hat{t}_i \leftarrow t_i + \gamma_i t_i$ $\quad \triangleright$ Select temporarily increased noise level $\hat{t}_i$

    $\hat{\boldsymbol{x}}_i \leftarrow \boldsymbol{x}_i + \sqrt{\hat{t}_i^2 - t_i^2}\ \boldsymbol{\epsilon}_i$ $\quad \triangleright$ Add new noise to move from t_i to $\hat{t}_i$

    $\boldsymbol{d}_i \leftarrow \big(\hat{\boldsymbol{x}}_i - D_\theta(\hat{\boldsymbol{x}}_i; \hat{t}_i)\big)/\hat{t}_i$ $\quad \triangleright$ Evaluate $\mathrm{d}\boldsymbol{x}/\mathrm{d}t$ at $\hat{t}_i$

    $\boldsymbol{x}_{i+1} \leftarrow \hat{\boldsymbol{x}}_i + (t_{i+1} - \hat{t}_i)\boldsymbol{d}_i$ $\quad \triangleright$ Take Euler step from $\hat{t}_i$ to t_{i+1}

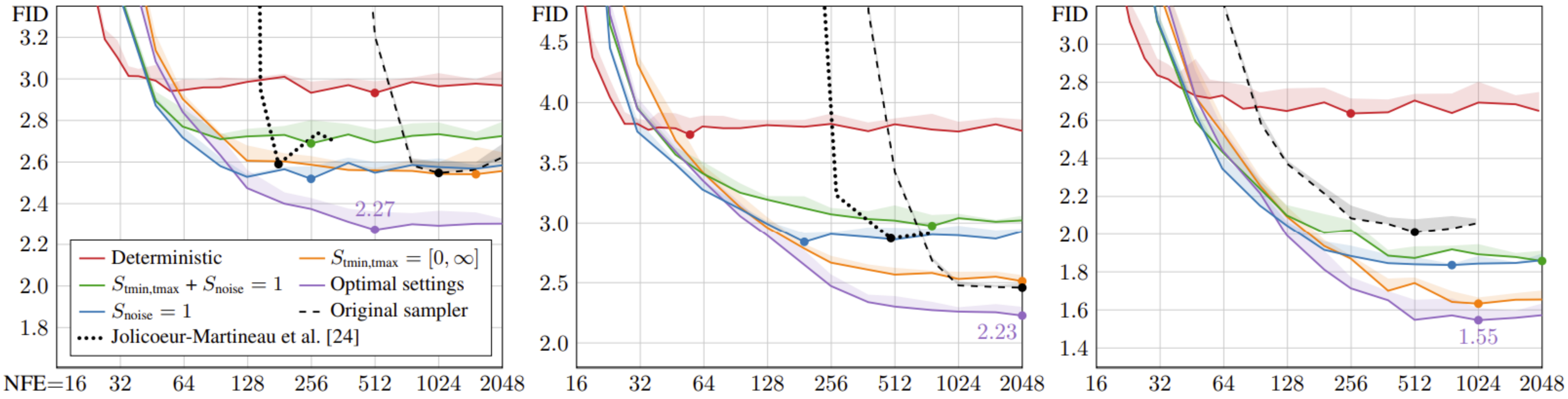
    if $t_{i+1} \neq 0$ **then**

      $\boldsymbol{d}'_i \leftarrow \big(\boldsymbol{x}_{i+1} - D_\theta(\boldsymbol{x}_{i+1}; t_{i+1})\big)/t_{i+1}$ $\quad \triangleright$ Apply 2nd order correction

      $\boldsymbol{x}_{i+1} \leftarrow \hat{\boldsymbol{x}}_i + (t_{i+1} - \hat{t}_i)\big(\tfrac{1}{2}\boldsymbol{d}_i + \tfrac{1}{2}\boldsymbol{d}'_i\big)$

  return $\boldsymbol{x}_N$

Stochastic sampler > Deterministic samplers



(a) Uncond. CIFAR-10, VP (b) Uncond. CIFAR-10, VE (c) Class-cond. ImageNet-64

Figure 4: Evaluation of our stochastic sampler (Algorithm 2). The purple curve corresponds to optimal choices for $\{S_{\text{churn}}, S_{\text{tmin}}, S_{\text{tmax}}, S_{\text{noise}}\}$; orange, blue, and green correspond to disabling the effects of $S_{\text{tmin,tmax}}$ and/or S_{noise}. The red curves show reference results for our deterministic sampler (Algorithm 1), equivalent to setting $S_{\text{churn}} = 0$. The dashed black curves correspond to the original stochastic samplers from previous work: Euler–Maruyama [49] for VP, predictor-corrector [49] for VE, and iDDPM [37] for ImageNet-64. The dots indicate lowest observed FID.

Stochastic sampler > Deterministic samplers

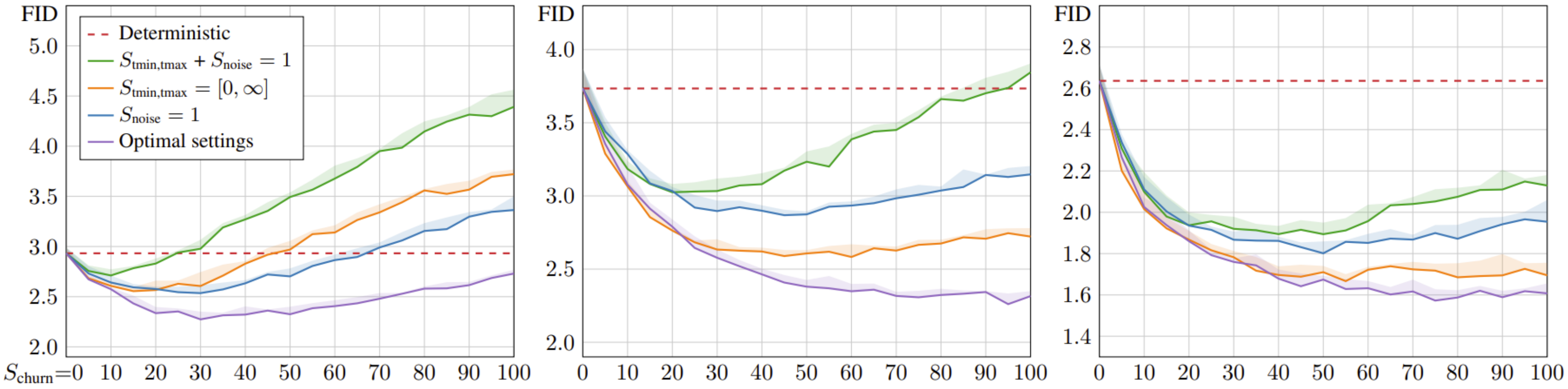
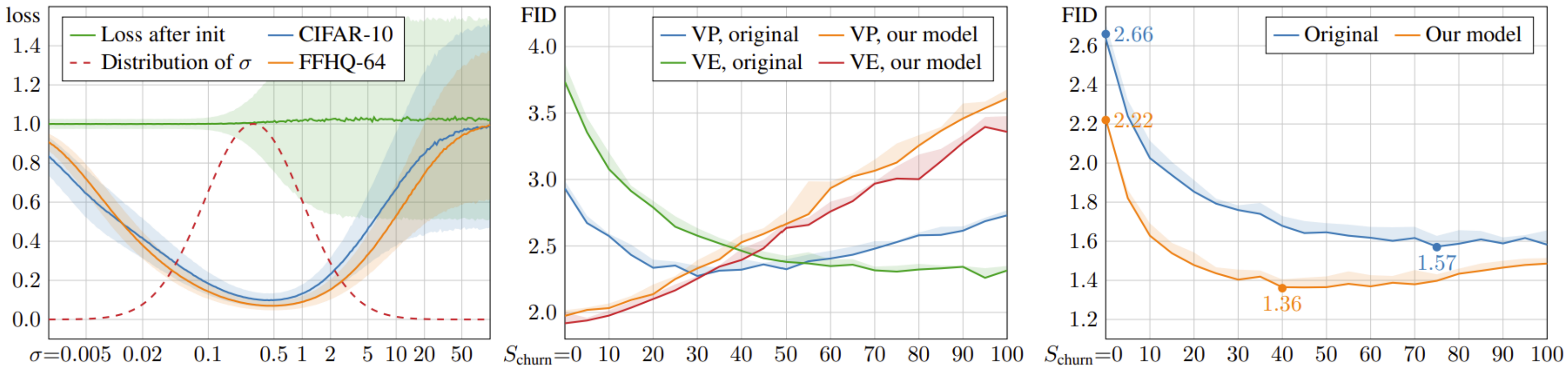


Figure 15: Ablations of our stochastic sampler (Algorithm 2) parameters using pre-trained networks of Song et al. [49] and Dhariwal and Nichol [9]. Each curve shows FID (y-axis) as a function of S_{churn} (x-axis) for $N = 256$ steps (NFE = 511). The dashed red lines correspond to our deterministic sampler (Algorithm 1), equivalent to setting $S_{\text{churn}} = 0$. The purple curves correspond to optimal choices for $\{S_{\text{tmin}}, S_{\text{tmax}}, S_{\text{noise}}\}$, found separately for each case using grid search. Orange, blue, and green correspond to disabling the effects of $S_{\text{tmin,tmax}}$ and/or S_{noise}. The shaded regions indicate the range of variation between the lowest and highest observed FID.

$$\gamma_i = \begin{cases} \min\left(\frac{S_{\text{churn}}}{N}, \sqrt{2}-1\right) & \text{if } t_i \in [S_{\text{tmin}}, S_{\text{tmax}}] \\ 0 & \text{otherwise} \end{cases}$$

If trained good, deterministic samplers can be good

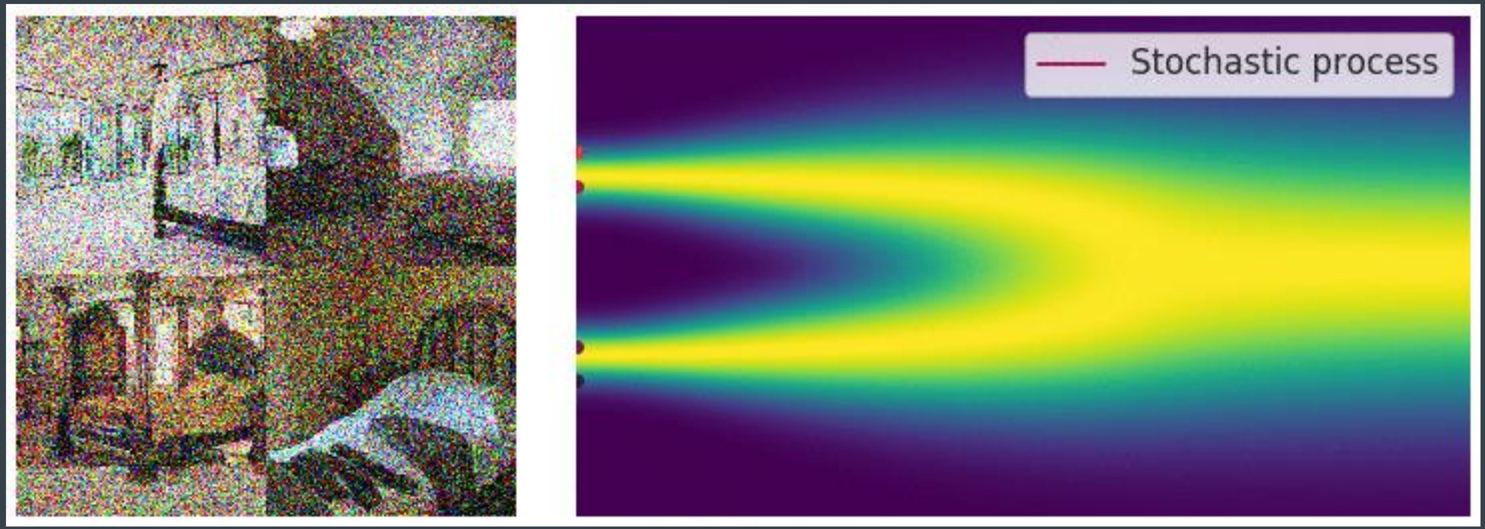


(a) Loss & noise distribution (b) Stochasticity on CIFAR-10 (c) Stochasticity on ImageNet-64

Figure 5: **(a)** Observed initial (green) and final loss per noise level, representative of the the 32×32 (blue) and 64×64 (orange) models considered in this paper. The shaded regions represent the standard deviation over 10k random samples. Our proposed training sample density is shown by the dashed red curve. **(b)** Effect of S_{churn} on unconditional CIFAR-10 with 256 steps (NFE $=$ 511). For the original training setup of Song et al. [49], stochastic sampling is highly beneficial (blue, green), while deterministic sampling ($S_{\text{churn}} = 0$) leads to relatively poor FID. For our training setup, the situation is reversed (orange, red); stochastic sampling is not only unnecessary but harmful. **(c)** Effect of S_{churn} on class-conditional ImageNet-64 with 256 steps (NFE $=$ 511). In this more challenging scenario, stochastic sampling turns out to be useful again. Our training setup improves the results for both deterministic and stochastic sampling.

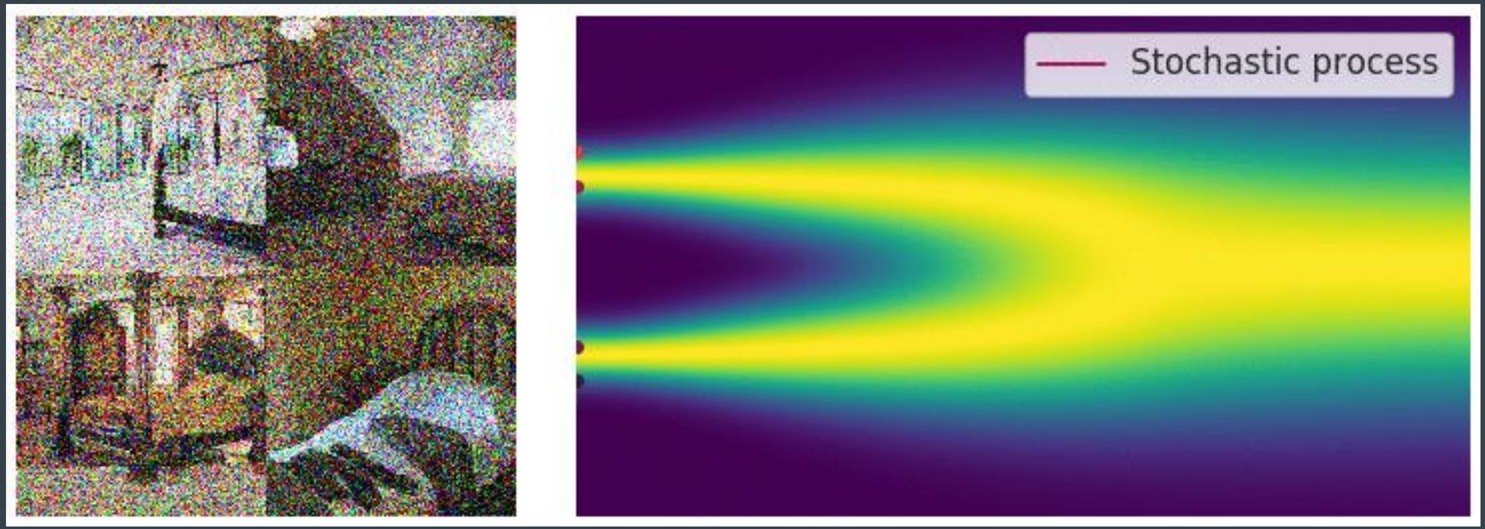
THE END

$\mathbf{x}(0) \rightarrow \mathbf{x}(T)$



$\mathbf{x}(0)$ $\mathbf{x}(T)$

$$\mathbf{x}(0) \longrightarrow \mathbf{x}(T)$$



$$\mathbf{x}(0) \longleftarrow \mathbf{x}(T)$$