EDM (Elucidating the Design Space of Diffusion-Based Generative Models)

Suk-Hwan Lee

Artificial Intelligence

Creating the Future

Dong-A University

Division of Computer Engineering & Artificial Intelligence

References

EDM : Tero Karras, Miika Aittala, Timo Aila, Samuli Laine, (nvidia)

"**Elucidating the Design Space of Diffusion-Based Generative Models**," NeurIPS 2022.

https://arxiv.org/abs/2206.00364

https://github.com/NVlabs/edm

https://www.youtube.com/watch?v=T0Qxzf0eaio

https://research.nvidia.com/publication/2022-11_elucidating-design-space-diffusion-based-generative-models

Blog

https://sang-yun-lee.notion.site/Elucidating-the-Design-Space-of-Diffusion-Based-Generative-Models-a81b14bf297743ec90a72c11f0fbce57

Elucidating the Design Space of Diffusion-Based Generative Models

Abstract

- Argue that the ***theory*** and ***practice*** of **diffusion-based generative models** are currently **unnecessarily convoluted** and seek to remedy the situation by presenting **a design space that clearly separates the concrete design choices**.
- **Changes** to both the **sampling and training processes**, as well as **preconditioning of the score networks**.
- New state-of-the-art **FID of 1.79** for CIFAR-10 in a **class-conditional setting** and **1.97 in an unconditional setting**, with much **faster sampling** (35 network evaluations per image) than prior designs.

1. Introduction

1) First contribution

- Look at the theory behind these models from a **practical standpoint**, **focusing on the "tangible" objects and algorithms** that appear in the ***training*** **and** ***sampling*** phases, and **less on the statistical processes**.
- Focus on the broad class of models where **a neural network is used to model the score [22] of a noise level dependent marginal distribution of the training data corrupted by Gaussian noise**. Thus, our work is in the context of ***denoising score matching*** [54].

2) Second contribution

- ***Sampling processes*** used to synthesize images using diffusion models.
- Identify the best-performing **time discretization for sampling**, apply a **higher order Runge–Kutta method** for the sampling process, evaluate different sampler schedules, and analyze the usefulness of stochasticity in the sampling process.
- A **significant drop in the number of sampling steps** required during synthesis

Elucidating the Design Space of Diffusion-Based Generative Models

3) Third contribution

- Focus on the **training of the score-modeling neural network**.
- While continue to rely on the commonly used network architectures (**DDPM** [16], **NCSN** [48]), we provide the first principled analysis of the **preconditioning of the networks' inputs, outputs, and loss functions in a diffusion model setting** and derive best practices for improving the *training dynamics*.
- Also suggest an **improved distribution of noise levels during training**, and note that **non-leaking augmentation** [25]—typically used with GANs—is beneficial for diffusion models as well.

T. Karras, M. Aittala, J. Hellsten, S. Laine, J. Lehtinen, and T. Aila. Training generative adversarial networks with limited data. In Proc. NeurIPS, 2020.

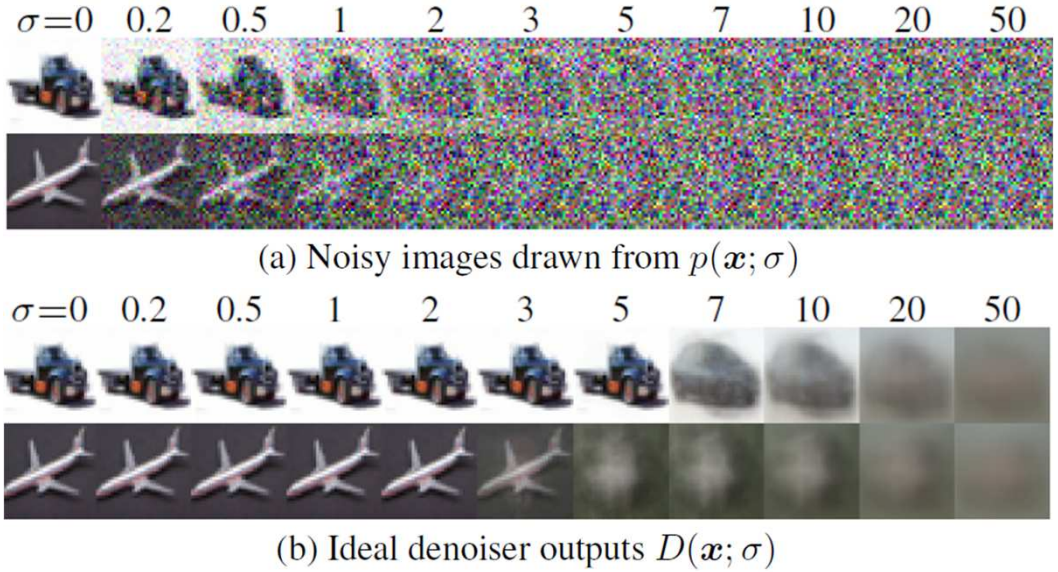


Figure 1: **Denoising score matching on CIFAR-10**. (a) Images from the training set corrupted with varying levels of additive Gaussian noise. High levels of noise lead to oversaturated colors; we normalize the images for cleaner visualization. (b) Optimal denoising result from minimizing Eq. 2 analytically (see Appendix B.3). With increasing noise level, the result approaches dataset mean.

2. Expressing diffusion models in a common framework

Definition

- $p_{data}(\boldsymbol{x})$: **Data distribution**, with standard deviation σ_{data}
- $p(\boldsymbol{x};\sigma)$: Family of **mollified distributions** obtained by **adding i.i.d. Gaussian noise** of standard deviation σ to the data.
- For $\sigma_{max} \gg \sigma_{data}$, $p(\boldsymbol{x};\sigma_{max})$ is practically indistinguishable from pure Gaussian noise.

- The idea of diffusion models
  - ✓ Randomly sample a noise image $\boldsymbol{x}_0 \sim \mathcal{N}(0, \sigma_{max}^2 \mathbf{I})$ and sequentially denoise it into images $\boldsymbol{x}_i$ with noise levels $\sigma_0 = \sigma_{max} > \sigma_1 > \cdots > \sigma_N = 0$ so that at each noise level $\boldsymbol{x}_i \sim p(\boldsymbol{x}_i;\sigma_i)$.
  - ✓ The endpoint $\boldsymbol{x}_N$ is distributed according to the data.

- Song et al. [49] ;
- ✓ Present a **stochastic differential equation (SDE)** that maintains the desired distribution p as sample $\boldsymbol{x}$ evolves over time.
- ✓ This allows the above process to be implemented using **a stochastic solver** that **both removes and adds noise at each iteration**.
- ✓ They also give a corresponding "**probability flow" ordinary differential equation (ODE)** where the only source of randomness is the initial noise image $\boldsymbol{x}_0$.

➢ Next page : Examining the **ODE**, as it **offers a fruitful setting for analyzing sampling trajectories and their discretizations**. The insights carry over to *stochastic sampling*.

2. Expressing diffusion models in a common framework

Y. Song, et al. Score-based generative modeling through stochastic differential equations. In Proc. ICLR, 2021.

Score-based Generative Modeling through SDE

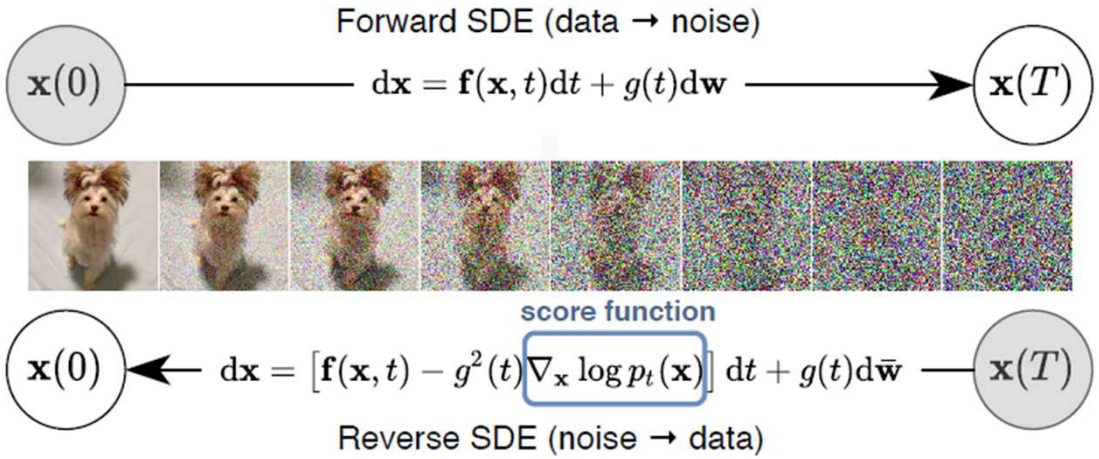


Figure 1: **Solving a reverse-time SDE yields a score-based generative model**. Transforming data to a simple noise distribution can be accomplished with a continuous-time SDE. This SDE can be reversed if we know the score of the distribution at each intermediate time step, $\nabla_x \log p_t(x)$.

$\mathbf{w}$: The standard Wiener process (a.k.a., Brownian motion),
$f(\cdot, t), \mathbb{R}^d \to \mathbb{R}^d$: A vector-valued function called the ***drift*** coefficient of $x(t)$
$g(\cdot), \mathbb{R} \to \mathbb{R}$: A scalar function known as the ***diffusion*** coefficient of $x(t)$.

2. Expressing diffusion models in a common framework

Y. Song, et al. Score-based generative modeling through stochastic differential equations. In Proc. ICLR, 2021.

Score-based Generative Modeling through SDE

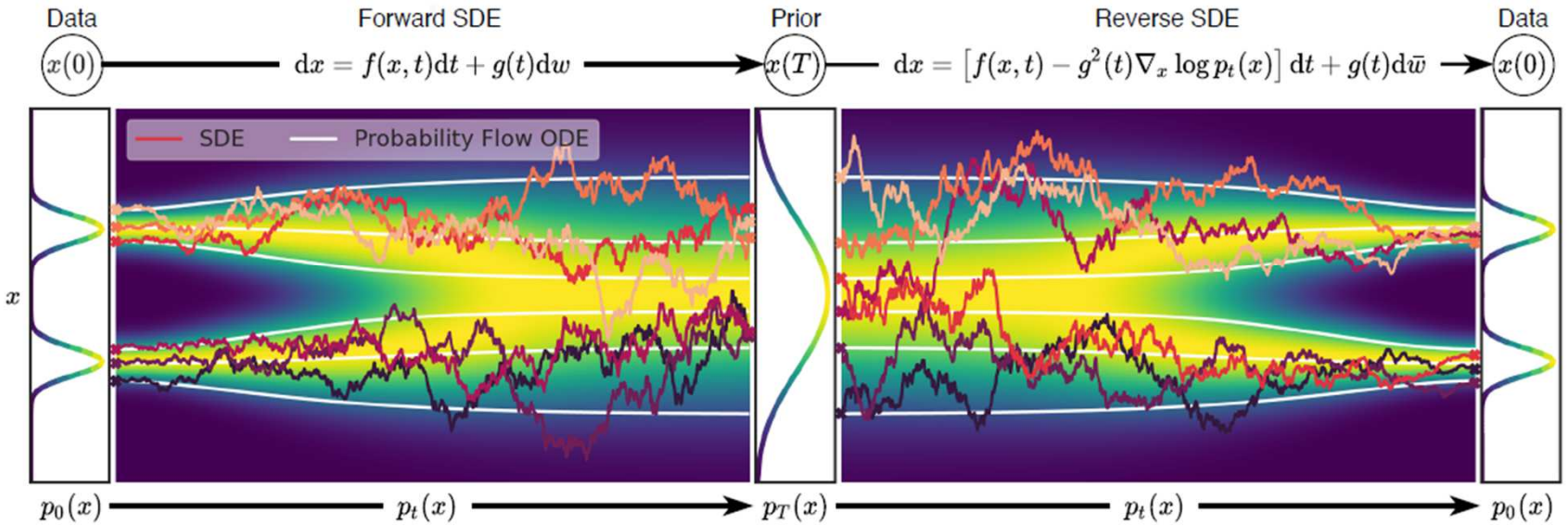


Figure 2: **Overview of score-based generative modeling through SDEs**. We can map **data** to **a noise distribution (the prior) with an SDE** (Section 3.1), and **reverse this SDE for generative modeling** (Section 3.2). We can also **reverse the associated probability flow ODE** (Section 4.3), which yields a **deterministic process that samples from the same distribution as the SDE**. **Both the reverse-time SDE and probability flow ODE can be obtained by estimating the score, $\nabla_x \log p_t(x)$** (Section 3.3).

2. Expressing diffusion models in a common framework

Y. Song, et al. Score-based generative modeling through stochastic differential equations. In Proc. ICLR, 2021.

ODE formulation

- **A probability flow ODE [49] continuously increases or reduces noise level of the image when moving forward or backward in time**, respectively.
- To specify the ODE, we must first **choose a schedule $\sigma(t)$** that defines the desired noise level at time t. ; Ex; $\sigma(t) \propto \sqrt{t}$, as it corresponds to constant-speed heat diffusion [12].

➢ The defining characteristic of the **probability flow ODE**

- Evolving a sample $\boldsymbol{x}_a \sim p(\boldsymbol{x}_a; \sigma(t_a))$ from time t_a to t_b (either forward or backward in time) yields a sample $\boldsymbol{x}_b \sim p(\boldsymbol{x}_b; \sigma(t_b))$
- This requirement is satisfied by

$$\mathrm{d}\boldsymbol{x} = -\dot{\sigma}(t)\,\sigma(t)\,\nabla_{\boldsymbol{x}} \log p\big(x; \sigma(t)\big)\,\mathrm{d}t, \qquad (1)$$

✓ Dot : time derivative

✓ $\nabla_x \log p(\boldsymbol{x}; \sigma(t))$: the ***score function*** [22], **a vector field that points towards higher density of data at a given noise level.**

- An **forward step** of this ODE nudges **the sample away from the data**, at a rate that depends on the change in noise level.
- Equivalently, a **backward step** nudges the **sample towards the data distribution.**

2. Expressing diffusion models in a common framework

Denoising score matching

- **Denoiser function** $D(x;\sigma)$: Minimize the expected L_2 denoising error for samples drawn from p_{data} separately for every σ,

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

$$\text{then } \nabla_{\boldsymbol{x}}\log p(\boldsymbol{x};\sigma)=\big(D(\boldsymbol{x};\sigma)-\boldsymbol{x}\big)/\sigma^2, \quad (3)$$

$\boldsymbol{y}$: a training image, $\boldsymbol{n}$: noise

- The **score function** isolates the noise component from the signal in $\boldsymbol{x}$, and Eq. 1 amplifies (or diminishes) it over time.

- The key observation in diffusion models is that $D(x;\sigma)$ can be implemented as **a neural network** $D_\theta(x;\sigma)$ **trained** according to Eq. 2.
- D_θ may include **additional pre- and post-processing steps**, such as scaling x to an appropriate dynamic range

- Figure 1 illustrates the behavior of ideal D in practice.

(b) Ideal denoiser outputs $D(\boldsymbol{x};\sigma)$

Figure 1: Denoising score matching on CIFAR-10. (a) Images from the training set corrupted with varying levels of additive Gaussian noise. High levels of noise lead to oversaturated colors; we normalize the images for cleaner visualization. (b) **Optimal denoising result** from minimizing Eq. 2 analytically (see Appendix B.3). With increasing noise level, the result approaches dataset mean.

2. Expressing diffusion models in a common framework

Time-dependent signal scaling

- Some methods (see Appendix C.1) introduce an additional **scale schedule** $s(t)$ and consider $\boldsymbol{x} = s(t)\hat{\boldsymbol{x}}$ to be a scaled version of the original, non-scaled variable $\hat{\boldsymbol{x}}$.
- This **changes** the **time-dependent probability density**, and consequently also the **ODE solution trajectories**.
- The resulting ODE is a generalization of Eq. (1).

$$\mathrm{d}\boldsymbol{x} = -\dot{\sigma}(t)\,\sigma(t)\,\nabla_{\boldsymbol{x}} \log p\big(x;\sigma(t)\big)\,\mathrm{d}t,$$

$$\mathrm{d}\boldsymbol{x} = \left[\frac{\dot{s}(t)}{s(t)}\,x - s(t)^2\,\dot{\sigma}(t)\,\sigma(t)\,\nabla_{\boldsymbol{x}} \log p\left(\frac{x}{s(t)};\sigma(t)\right)\right]\,\mathrm{d}t. \qquad (4)$$

- Note that : Explicitly undo the scaling of $\boldsymbol{x}$ when evaluating the score function to keep the definition of $p(\boldsymbol{x};\sigma)$ independent of $s(t)$.

Solution by discretization

- The ODE to be solved is obtained by substituting Eq. 3 into Eq. 4 to define the point-wise gradient

$$\nabla_{\boldsymbol{x}} \log p(\boldsymbol{x};\sigma) = \big(D(\boldsymbol{x};\sigma) - \boldsymbol{x}\big)/\sigma^2,$$

$$\mathrm{d}\boldsymbol{x} = \left[\frac{\dot{s}(t)}{s(t)}\,\boldsymbol{x} - s(t)^2\,\dot{\sigma}(t)\,\sigma(t)\,\nabla_{\boldsymbol{x}} \log p\left(\frac{\boldsymbol{x}}{s(t)};\sigma(t)\right)\right]\,\mathrm{d}t.$$

- The solution can be found by **numerical integration**, i.e., taking finite steps over discrete time intervals.
- This requires choosing both the **integration scheme** (e.g., **Euler** or a **variant of Runge–Kutta**), as well as the **discrete sampling times** $\{t_0, t_1, \cdots, t_N\}$.
- Many prior works rely on Euler's method, but we show in Section 3 that a **2nd order solver** offers a **better computational tradeoff**.

2. Expressing diffusion models in a common framework

Putting it together

Table 1: **Specific design choices** employed by different model families.

N is the number of ODE solver iterations that we wish to execute during *sampling*.

The corresponding sequence of time steps is $\{t_0, t_1, \cdots, t_N\}$, where $t_N = 0$.

If the model was originally trained for specific choices of N and $\{t_i\}$, the originals are denoted by M and $\{\mu_i\}$, respectively.

The **denoiser** is defined as

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

F_θ : the raw neural network layers

		VP [49]	VE [49]	iDDPM [37] + DDIM [47]	Ours ("EDM")
Sampling (Section 3)					
ODE solver		Euler	Euler	Euler	2nd order Heun
Time steps	$t_{i<N}$	$1 + \frac{i}{N-1}(\epsilon_s - 1)$	$\sigma_{\max}^2 \left(\sigma_{\min}^2/\sigma_{\max}^2\right)^{\frac{i}{N-1}}$	$u_{\lfloor j_0 + \frac{M-1-j_0}{N-1} i + \frac{1}{2} \rfloor}$, where $u_M = 0$, $u_{j-1} = \sqrt{\frac{u_j^2+1}{\max(\bar{\alpha}_{j-1}/\bar{\alpha}_j, C_1)} - 1}$	$\left(\sigma_{\max}^{\frac{1}{\rho}} + \frac{i}{N-1}(\sigma_{\min}^{\frac{1}{\rho}} - \sigma_{\max}^{\frac{1}{\rho}})\right)^\rho$
Schedule	$\sigma(t)$	$\sqrt{e^{\frac{1}{2}\beta_d t^2 + \beta_{\min} t} - 1}$	$\sqrt{t}$	t	t
Scaling	$s(t)$	$1/\sqrt{e^{\frac{1}{2}\beta_d t^2 + \beta_{\min} t}}$	1	1	1
Network and preconditioning (Section 5)					
Architecture of F_θ		DDPM++	NCSN++	DDPM	(any)
Skip scaling	$c_{\text{skip}}(\sigma)$	1	1	1	$\sigma_{\text{data}}^2/\left(\sigma^2 + \sigma_{\text{data}}^2\right)$
Output scaling	$c_{\text{out}}(\sigma)$	$-\sigma$	σ	$-\sigma$	$\sigma \cdot \sigma_{\text{data}}/\sqrt{\sigma_{\text{data}}^2 + \sigma^2}$
Input scaling	$c_{\text{in}}(\sigma)$	$1/\sqrt{\sigma^2+1}$	1	$1/\sqrt{\sigma^2+1}$	$1/\sqrt{\sigma^2 + \sigma_{\text{data}}^2}$
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		$\sigma^{-1}(\sigma) \sim \mathcal{U}(\epsilon_t, 1)$	$\ln(\sigma) \sim \mathcal{U}(\ln(\sigma_{\min}), \ln(\sigma_{\max}))$	$\sigma = u_j,\ \ j \sim \mathcal{U}\{0, M-1\}$	$\ln(\sigma) \sim \mathcal{N}(P_{\text{mean}}, P_{\text{std}}^2)$
Loss weighting	$\lambda(\sigma)$	$1/\sigma^2$	$1/\sigma^2$	$1/\sigma^2$ (note: *)	$\left(\sigma^2 + \sigma_{\text{data}}^2\right)/(\sigma \cdot \sigma_{\text{data}})^2$
Parameters		$\beta_d = 19.9, \beta_{\min} = 0.1$ $\epsilon_s = 10^{-3}, \epsilon_t = 10^{-5}$ $M = 1000$	$\sigma_{\min} = 0.02$ $\sigma_{\max} = 100$	$\bar{\alpha}_j = \sin^2(\frac{\pi}{2}\frac{j}{M(C_2+1)})$ $C_1 = 0.001, C_2 = 0.008$ $M = 1000, j_0 = 8^\dagger$	$\sigma_{\min} = 0.002, \sigma_{\max} = 80$ $\sigma_{\text{data}} = 0.5, \rho = 7$ $P_{\text{mean}} = -1.2, P_{\text{std}} = 1.2$

* iDDPM also employs a second loss term L_{vlb} † In our tests, $j_0 = 8$ yielded better FID than $j_0 = 0$ used by iDDPM

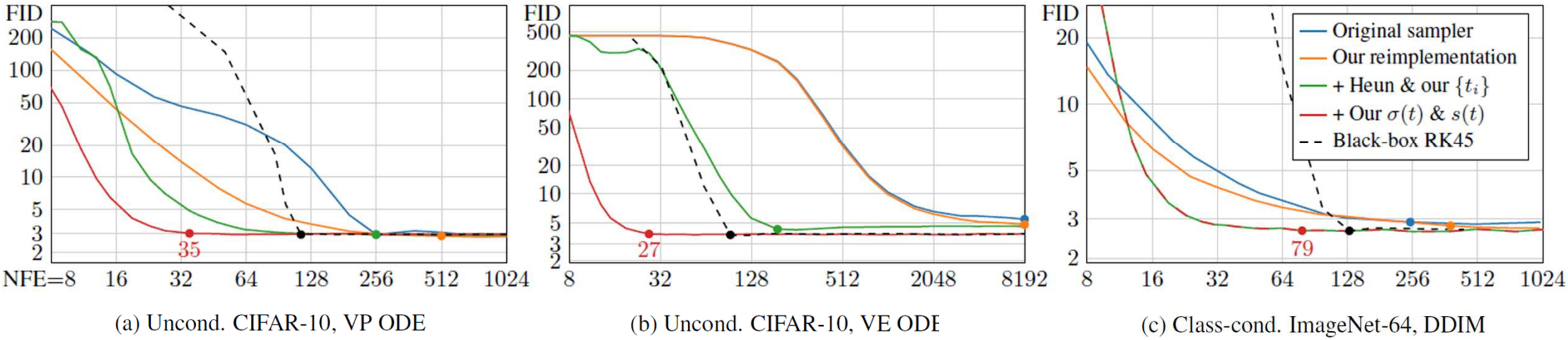
3. Improvements to deterministic sampling

- Topics in **diffusion model** research : **Improving the output quality** and/or **Decreasing the computational cost of sampling**

- Our hypothesis : **Choices related to the sampling process** are **largely independent of the other components**; network architecture and training details.
  - ✓ In other words, the training procedure of D_θ should not dictate $\sigma(t)$, $s(t)$, and $\{t_i\}$, nor vice versa
  - ✓ From the sampler viewpoint, D_θ is simply a black box [55, 56].

- We evaluate the "**DDPM++ cont. (VP)**" and "**NCSN++ cont. (VE)**" models by Song et al. [49] **trained** on **unconditional CIFAR-10** [29] at 32x32, corresponding to the variance preserving (VP) and variance exploding (VE) formulations [49], originally inspired by DDPM [16] and SMLD [48].
- We also evaluate the "**ADM (dropout)**" model by Dhariwal and Nichol [9] **trained on class-conditional ImageNet** [8] at 64x64, corresponding to the improved DDPM (**iDDPM**) formulation [37]. This model was trained using a discrete set of **M = 1000 noise levels**.

3. Improvements to deterministic sampling



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

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.

- Figure 2 shows **FID** as a function of **neural function evaluations (NFE)**, how many times D_θ is evaluated to produce a single image.
- Given that the **sampling process** is **dominated** entirely by the cost of D_θ, improvements in NFE translate directly to sampling speed.
- The reimplementations of these methods in our unified framework (orange) yield similar but consistently better results

3. Improvements to deterministic sampling

Discretization and higher-order integrators.

- **Solving an ODE numerically** is necessarily an **approximation of following the true solution trajectory**.
- At each step, the solver introduces ***truncation error*** that accumulates over the course of *N* steps. The local error generally scales *superlinearly* with respect to step size, and thus increasing *N* improves the accuracy of the solution.

- **Euler**'s method : the first order ODE solver with $O(h^2)$ local error with respect to step size h.
- **Higher-order Runge–Kutta** methods [50] scale more favorably but require multiple evaluations of D_θ per step.
- Heun's 2nd order method [2] (a.k.a. **improved Euler, trapezoidal rule**) to provide an excellent tradeoff between truncation error and NFE.

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

procedure HEUNSAMPLER($D_\theta(x;\sigma)$, $\sigma(t)$, $s(t)$, $t_{i\in\{0,\dots,N\}}$)
  sample $\boldsymbol{x}_0 \sim \mathcal{N}(\mathbf{0},\ \sigma^2(t_0)\, s^2(t_0)\, \mathbf{I})$ ▷ Generate initial sample at t_0
  for $i \in \{0,\dots,N-1\}$ **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}
      $\boldsymbol{x}_{i+1} \leftarrow \boldsymbol{x}_i + (t_{i+1}-t_i)\left(\frac{1}{2}\boldsymbol{d}_i + \frac{1}{2}\boldsymbol{d}'_i\right)$ ▷ Explicit trapezoidal rule at t_{i+1}
  return $\boldsymbol{x}_N$ ▷ Return noise-free sample at t_N

- Algorithm 1 introduces an additional **correction step** for $\boldsymbol{x}_{i+1}$ to account for change in dx/dt between t_i and t_{i+1}.
- This correction leads to $O(h^3)$ local error at the cost of one additional evaluation of D_θ per step.
- Note that stepping to $\sigma = 0$ would result in a division by zero, so we revert to Euler's method in this case.

3. Improvements to deterministic sampling

Discretization and higher-order integrators.

❖ **Time steps** $\{t_i\}$

- The time steps $\{t_i\}$ determine how the **step sizes** and thus **truncation errors** are distributed between **different noise levels**.
- We adopt a parameterized scheme where the time steps are defined according to a sequence of noise levels $\{\sigma_i\}$, i.e.,.

$$t_i = \sigma^{-1}(\sigma_i)$$

- Set $\sigma_{i<N} = (A_i + B)^\rho$ and select the constants A and B so that $\sigma_0 = \sigma_{max}$ and $\sigma_{N-1} = \sigma_{min}$, which gives

$$\sigma_{i<N} = \left(\sigma_{\max}{}^{\frac{1}{\rho}} + \frac{i}{N-1}\left(\sigma_{\min}{}^{\frac{1}{\rho}} - \sigma_{\max}{}^{\frac{1}{\rho}}\right)\right)^\rho \quad \text{and} \quad \sigma_N = 0. \tag{5}$$

- ρ controls how much the steps near σ_{min} are shortened at the expense of longer steps near σ_{max}.
- Suggest $\rho = 7$

- Results for **Heun's method and Eq. 5** are shown as the green curves in Figure 2. Heun's method reaches the same FID as Euler's method with considerably lower NFE.

3. Improvements to deterministic sampling

Trajectory curvature and noise schedule.

- **Schedule $\sigma(t)$** and **Scale $s(t)$** defines the shape of the ODE solution trajectories.
- The choice of $\sigma(t)$ and $s(t)$ offers a way to reduce the truncation errors, as their magnitude can be expected to scale proportional to the curvature of dx/dt.
- Best choice : $\sigma(t) = t$ and $s(t) = 1$, made in DDIM [47]. (Red curves in Figure 2)
  - The ODE of Eq. 4 simplifies to

$$\mathrm{d}\boldsymbol{x} = \left[\frac{\dot{s}(t)}{s(t)}\,\boldsymbol{x} - s(t)^2\,\dot{\sigma}(t)\,\sigma(t)\,\nabla_{\boldsymbol{x}} \log p\left(\frac{\boldsymbol{x}}{s(t)};\sigma(t)\right)\right]\,\mathrm{d}t.$$

$$\mathrm{d}\boldsymbol{x}/\mathrm{d}t = \big(\boldsymbol{x} - D(\boldsymbol{x};t)\big)/t$$

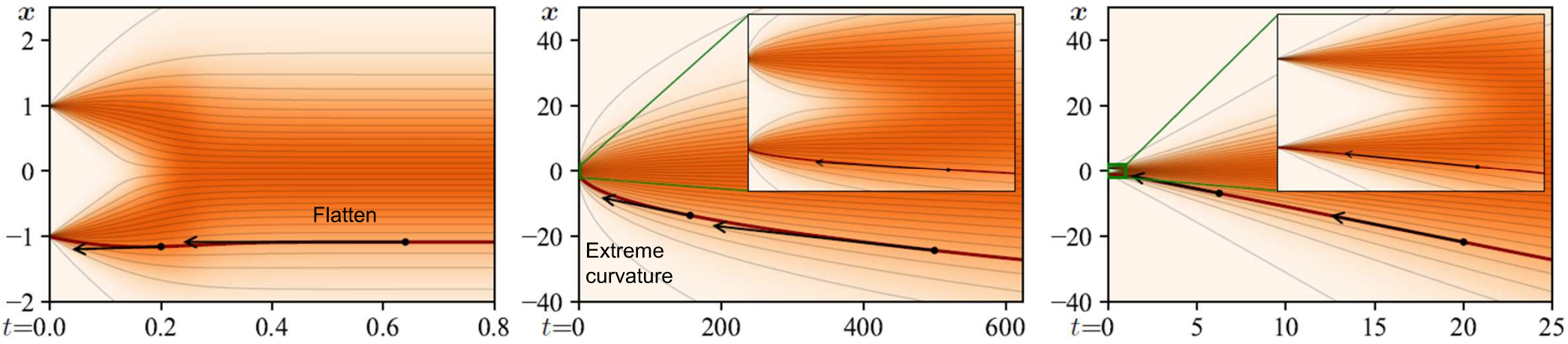
  - σ and s become interchangeable.

- An immediate consequence is that at any x and t, a single Euler step to $t = 0$ yields the denoised image $D_\theta(x;t)$.
- The tangent of the solution trajectory therefore always points towards the denoiser output. This can be expected to change only slowly with the noise level, which corresponds to largely linear solution trajectories.

- The 1D ODE sketch of Figure 3c
- Solution trajectories approach linear at both large and small noise levels, and have substantial curvature in only a small region in between.
- In Figure 1(b), the change between different denoiser targets occurs in a relatively narrow σ range. With the advocated schedule, this corresponds to high ODE curvature being limited to this same range.

Fig. 1(b) Ideal deniser outputs $D(x;\sigma)$

3. Improvements to deterministic sampling



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

Figure 3: A sketch of **ODE curvature in 1D where p_{data} is two Dirac peaks at $x = \pm 1$**. Horizontal t axis is chosen to show $\sigma \in [0,25]$ in each plot, with insets showing $\sigma \in [0,1]$ near the data. Example local gradients are shown with black arrows.

(a) **Variance preserving ODE** of Song et al. [49] has solution trajectories that flatten out to horizontal lines at large σ. Local gradients start pointing towards data only at small σ.

(b) **Variance exploding** variant has extreme curvature near data and the solution trajectories are curved everywhere.

(c) With **the schedule** used by **DDIM** [47] and us, as σ increases, the solution trajectories approach straight lines that point towards the mean of data. As $\sigma \to 0$, the trajectories become linear and point towards the data manifold.

4. Stochastic Sampling

- **Deterministic sampling** offers many benefits,
  - ✓ e.g., the ability to turn real images into their corresponding latent representations by inverting the ODE.
  - ✓ However, lead to **worse output quality** [47, 49] **than stochastic sampling that injects fresh noise into the image in each step.**
- Given that ODEs and SDEs recover the same distributions in theory, what exactly is the role of stochasticity

Backgrounds

- The **SDEs** of Song et al. [49] can be **generalized** [20, 58] as a **sum** of the **probability flow ODE** of Eq. 1 and a **time-varying *Langevin diffusion* SDE** [14] (see Appendix B.5):

- ω_t : the standard Wiener process.
- dx_+ dx_- : Separate SDEs for moving forward and backward in time
- **Langevin term** : Combination of a **deterministic score-based denoising term** and a **stochastic noise injection term**, whose net noise level contributions cancel out.
- $\boldsymbol{\beta(t)}$**; Noise replacement schedue** : Expresses **the relative rate at which existing noise is replaced with new noise**. The SDEs of Song et al. [49] are recovered with the choice $\boldsymbol{\beta(t) = \dot{\sigma}(t)/\sigma(t)}$, whereby **the score vanishes from the forward SDE.**

$$\mathrm{d}\boldsymbol{x}_{\pm} = \underbrace{-\dot{\sigma}(t)\sigma(t)\nabla_{\boldsymbol{x}} \log p\big(\boldsymbol{x};\sigma(t)\big)\,\mathrm{d}t}_{\text{probability flow ODE (Eq. 1)}} \pm \underbrace{\underbrace{\beta(t)\sigma(t)^2\nabla_{\boldsymbol{x}} \log p\big(\boldsymbol{x};\sigma(t)\big)\,\mathrm{d}t}_{\text{deterministic noise decay}} + \underbrace{\sqrt{2\beta(t)}\sigma(t)\,\mathrm{d}\omega_t}_{\text{noise injection}}}_{\text{Langevin diffusion SDE}}, \quad (6)$$

The implicit Langevin diffusion drives the sample towards the desired marginal distribution at a given time, actively correcting for any errors made in earlier sampling steps.

4. Stochastic Sampling

Our stochastic sampler

- A stochastic sampler that combines our **2nd order deterministic ODE integrator** with **explicit Langevin-like "churn" of adding and removing noise**

- At each step i, given the sample x_i at noise level t_i (= $\sigma(t_i)$),

  1) First, we add noise to the sample according to a factor $\gamma_i \geq 0$ to reach a higher noise level $\hat{t}_i = t_i + \gamma_i t_i$

  2) Second, from the increased-noise sample $\hat{x}_i$, we solve the ODE backward from $\hat{t}_i$ to t_{i+1} with a single step. This yields a sample x_{i+1} with noise level t_{i+1}, and the iteration continues.

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

1: **procedure** StochasticSampler($D_\theta(x;\sigma)$, $t_{i\in\{0,\dots,N\}}$, $\gamma_{i\in\{0,\dots,N-1\}}$, S_{noise})
2: sample $x_0 \sim \mathcal{N}(0,\ t_0^2\,\mathbf{I})$
3: **for** $i \in \{0,\dots,N-1\}$ **do**
4: sample $\epsilon_i \sim \mathcal{N}(0,\ S_{\text{noise}}^2\,\mathbf{I})$
5: 1) $\hat{t}_i \leftarrow t_i + \gamma_i t_i$
6: $\hat{x}_i \leftarrow x_i + \sqrt{\hat{t}_i^2 - t_i^2}\,\epsilon_i$ ▷ Add new noise to move from t_i to $\hat{t}_i$
7: 2) $d_i \leftarrow (\hat{x}_i - D_\theta(\hat{x}_i;\hat{t}_i))/\hat{t}_i$ ▷ Evaluate $\mathrm{d}x/\mathrm{d}t$ at $\hat{t}_i$
8: $x_{i+1} \leftarrow \hat{x}_i + (t_{i+1} - \hat{t}_i)d_i$ ▷ Take Euler step from $\hat{t}_i$ to t_{i+1}
9: **if** $t_{i+1} \neq 0$ **then**
10: $d'_i \leftarrow (x_{i+1} - D_\theta(x_{i+1};t_{i+1}))/t_{i+1}$ ▷ Apply 2nd order correction
11: $x_{i+1} \leftarrow \hat{x}_i + (t_{i+1} - \hat{t}_i)\left(\frac{1}{2}d_i + \frac{1}{2}d'_i\right)$
12: **return** x_N

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

- In our method, the parameters used to evaluate D_θ on line 7 of Algorithm 2 correspond to the state after noise injection, whereas an Euler–Maruyama -like method would use $x_i; t_i$ instead of $\hat{x}_i; \hat{t}_i$.
- In the limit of Δ_t approaching zero, there may be no difference between these choices, but the distinction appears to become significant when **pursuing low NFE with large steps**.

4. Stochastic Sampling

Practical Considerations

- **Increasing** the **amount of *stochasticity*** is **effective in correcting errors made by earlier sampling steps**, but it has its own **drawbacks**. (Appendix E.1)
  - ✓ **Excessive Langevin-like addition and removal of noise** results in **gradual loss** of detail in the generated images with all datasets and denoiser networks.
  - ✓ There is **a drift toward oversaturated colors at very low and high noise levels**.

- Suspect that **practical denoisers induce a slightly nonconservative vector field in Eq. 3, violating the premises of Langevin diffusion and causing these detrimental effects**.
- Notably, our experiments with **analytical denoisers** (such as the one in Figure 1b) have **not shown such degradation**.

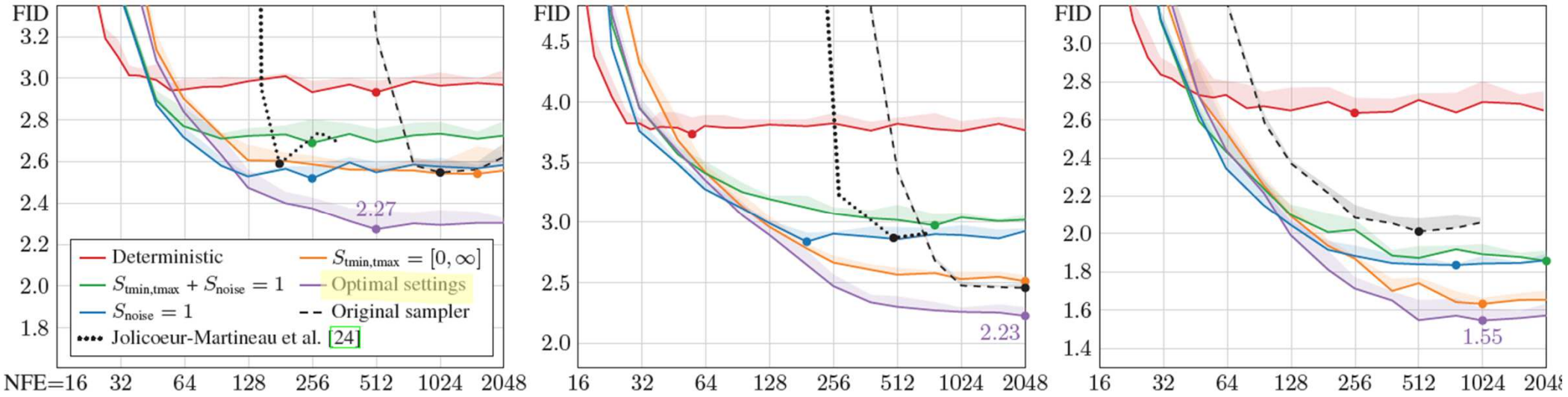
- If the degradation is caused by flaws in $D_\theta(x;t)$, they can only be remedied using **heuristic means during sampling**.

- We address the **drift toward oversaturated colors by only enabling stochasticity within a specific range of noise levels** $t_i \in [S_{tmin}, S_{tmax}]$.
  - ✓ For these noise levels, we define $\gamma_i = S_{churn}/N$, where S_{churn} controls the overall amount of stochasticity. Clamp γ_i to never introduce more new noise than what is already present in the image

- The loss of detail can be partially counteracted by **setting S_{noise} slightly above 1 to inflate the standard deviation for the newly added noise**.
  - ✓ This suggests that a major component of the hypothesized non-conservativity of $D_\theta(x;t)$ is a tendency to remove slightly too much noise—most likely due to **regression toward the mean** that can be expected to happen with **any L2-trained denoiser**

4. Stochastic Sampling

Evaluation



(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.

Elucidating the Design Space of Diffusion-Based Generative Models

5. Preconditioning and training

Practices for training neural networks in a supervised fashion

- **Training a neural network to model D directly would be far from ideal**
  - ✓ For example, as the input $\boldsymbol{x} = \boldsymbol{y} + \boldsymbol{n}$ is a combination of clean signal $\boldsymbol{y}$ and noise $\boldsymbol{n} \sim \mathcal{N}(\boldsymbol{0}; \sigma^2 \boldsymbol{I})$, its magnitude varies immensely depending on noise level σ.
  - ✓ For this reason, the common practice is to not represent D_θ as a neural network directly, but instead **train a different network $\boldsymbol{F_\theta}$ from which $\boldsymbol{D_\theta}$ is derived**.

Previous methods [37,47,49]

- the input scaling via a σ-dependent normalization factor and precondition the output by training F_θ to predict $\boldsymbol{n}$ scaled to unit variance, from which the signal is then reconstructed via $\boldsymbol{D_\theta(x;\sigma)} = \boldsymbol{x - \sigma F_\theta(\cdot)}$. $D_\theta(\boldsymbol{x};\sigma) = \boldsymbol{x} - \sigma F_\theta(\cdot)$
- *Drawback* : **At large $\boldsymbol{\sigma}$, the network needs to fine-tune its output carefully to cancel out the existing noise $\boldsymbol{n}$ exactly and give the output at the correct scale**; note that any errors made by the network are amplified by a factor of σ.
- In this situation, it would seem **much easier to predict the expected output $\boldsymbol{D_\theta(x;\sigma)}$ directly**.

- Propose to precondition the neural network with a σ-dependent skip connection that allows it to estimate either $\boldsymbol{y}$ or $\boldsymbol{n}$, or something in between

$$D_\theta(\boldsymbol{x};\sigma) = c_{\text{skip}}(\sigma)\,\boldsymbol{x} + c_{\text{out}}(\sigma)\,F_\theta\big(c_{\text{in}}(\sigma)\,\boldsymbol{x};\; c_{\text{noise}}(\sigma)\big) \tag{7}$$

  - ✓ F_θ : the neural network to be trained
  - ✓ $c_{\text{skip}}(\sigma)$: Modulates the skip connection
  - ✓ $c_{\text{in}}(\sigma)$, $c_{\text{out}}(\sigma)$: Scale the input and output magnitudes
  - ✓ $c_{\text{noise}}(\sigma)$: Map noise level σ into a conditioning input for F_θ.
  - ✓ Taking a weighted expectation of Eq. 2 over the noise levels gives the overall training loss

$$\mathbb{E}_{\sigma,\boldsymbol{y},\boldsymbol{n}}\left[\lambda(\sigma)\,\|D(\boldsymbol{y}+\boldsymbol{n};\sigma) - \boldsymbol{y}\|_2^2\right]$$

where $\sigma \sim p_{\text{train}}$, $\boldsymbol{y} \sim p_{\text{data}}$, and $\boldsymbol{n} \sim \mathcal{N}(\boldsymbol{0}, \sigma^2\boldsymbol{I})$

- The probability of sampling a given noise level σ is given by $p_{\text{train}}(\sigma)$ and the corresponding weight is given by $\lambda(\sigma)$. We can equivalently express this loss with respect to the raw network output F in Eq. 7:

5. Preconditioning and training

Loss Function

- Equivalently express the loss with respect to the raw network output F_θ in Eq. 7

$$\mathbb{E}_{\sigma,\boldsymbol{y},\boldsymbol{n}}\left[\lambda(\sigma)\,\|D(\boldsymbol{y}+\boldsymbol{n};\sigma)-\boldsymbol{y}\|_2^2\right] \quad \text{where } \sigma\sim p_{\text{train}},\ \boldsymbol{y}\sim p_{\text{data}}, \text{ and } \boldsymbol{n}\sim\mathcal{N}(\mathbf{0},\sigma^2\mathbf{I})$$

$$D_\theta(\boldsymbol{x};\sigma) = c_{\text{skip}}(\sigma)\,\boldsymbol{x} + c_{\text{out}}(\sigma)\,F_\theta\big(c_{\text{in}}(\sigma)\,\boldsymbol{x};\,c_{\text{noise}}(\sigma)\big) \qquad (7)$$

$$\mathbb{E}_{\sigma,\boldsymbol{y},\boldsymbol{n}}\Big[\underbrace{\lambda(\sigma)\,c_{\text{out}}(\sigma)^2}_{\text{effective weight}}\,\Big\|\underbrace{F_\theta\big(c_{\text{in}}(\sigma)\cdot(\boldsymbol{y}+\boldsymbol{n});c_{\text{noise}}(\sigma)\big)}_{\text{network output}}-\underbrace{\tfrac{1}{c_{\text{out}}(\sigma)}\big(\boldsymbol{y}-c_{\text{skip}}(\sigma)\cdot(\boldsymbol{y}+\boldsymbol{n})\big)}_{\text{effective training target}}\Big\|_2^2\Big] \qquad (8)$$

- This form reveals the *effective training target* of F_θ, allowing us to determine suitable choices for the preconditioning functions from first principles

- Best choices in Table 1 : Requiring network inputs and training targets to have unit variance (c_{in}, c_{out}), and amplifying errors in F_θ as little as possible (c_{skip}).
- The formula for c_{noise} is chosen empirically

Architecture of F_θ		(any)
Skip scaling	$c_{\text{skip}}(\sigma)$	$\sigma_{\text{data}}^2/\left(\sigma^2+\sigma_{\text{data}}^2\right)$
Output scaling	$c_{\text{out}}(\sigma)$	$\sigma\cdot\sigma_{\text{data}}/\sqrt{\sigma_{\text{data}}^2+\sigma^2}$
Input scaling	$c_{\text{in}}(\sigma)$	$1/\sqrt{\sigma^2+\sigma_{\text{data}}^2}$
Noise cond.	$c_{\text{noise}}(\sigma)$	$\frac{1}{4}\ln(\sigma)$

Elucidating the Design Space of Diffusion-Based Generative Models

5. Preconditioning and training

Table 2: Evaluation of our training improvements. The starting point (config A) is VP & VE using our **deterministic** sampler. At the end (configs E,F), VP & VE only differ in the architecture of F_θ.

	CIFAR-10 [29] at 32×32				FFHQ [27] 64×64		AFHQv2 [7] 64×64	
	Conditional		Unconditional		Unconditional		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

- Config A : Baseline : Y. Song, J. Sohl-Dickstein, D. P. Kingma, A. Kumar, S. Ermon, and B. Poole. Score-based generative modeling through stochastic differential equations. In Proc. ICLR, 2021
- Config B : Re-adjust the basic hyperparameters
- Config C : Improve the expressive power of the model by removing the lowest-resolution layers and doubling the capacity of the highest-resolution layers instead; see Appendix F.3
- Config D : We then replace the original choices of $(c_{in}, c_{out}, c_{noise}, c_{skip})$ with our preconditioning, which keeps the results largely unchanged—except for VE that improves considerably at 64x64 resolution.
- Instead of improving FID per se, the main benefit of our preconditioning is that it makes the training more robust, enabling us to turn our focus on redesigning the loss function without adverse effects

5. Preconditioning and training

Loss weighting and sampling (config E in Table 2)

- training F_θ as preconditioned in Eq. 7 incurs an effective per-sample loss weight of $\lambda(\sigma)c_{\text{out}}(\sigma)^2$

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

- To balance the *effective loss weights*, we set $\lambda(\sigma)=1/c_{\text{out}}(\sigma)^2$, which also equalizes the initial training loss over the entire σ range as shown in Figure 5a (green curve).
- Finally, we need to **select $p_{\text{train}}(\sigma)$**, i.e., **how to choose noise levels during training.**
  - ✓ Inspecting the per- σ loss after training (blue and orange curves) reveals that **a significant reduction is possible only at intermediate noise levels**;
  - ✓ At very low levels : it is both difficult and irrelevant to discern the vanishingly small noise component,
  - ✓ At high levels : the training targets are always dissimilar from the correct answer that approaches dataset average.

Architecture of F_θ		(any)
Skip scaling	$c_{\text{skip}}(\sigma)$	$\sigma_{\text{data}}^2/\left(\sigma^2+\sigma_{\text{data}}^2\right)$
Output scaling	$c_{\text{out}}(\sigma)$	$\sigma\cdot\sigma_{\text{data}}/\sqrt{\sigma_{\text{data}}^2+\sigma^2}$
Input scaling	$c_{\text{in}}(\sigma)$	$1/\sqrt{\sigma^2+\sigma_{\text{data}}^2}$
Noise cond.	$c_{\text{noise}}(\sigma)$	$\frac{1}{4}\ln(\sigma)$

- Therefore, we target the training efforts to the relevant range using a **simple log-normal distribution for $p_{\text{train}}(\sigma)$** as detailed in Table 1 and illustrated in Figure 5a (red curve).
- Table 2 shows that our proposed $p_{\text{train}}(\sigma)$ and $\lambda(\sigma)$ (config E) lead to a dramatic improvement in FID in all cases when used in conjunction with our preconditioning (config D).

5. Preconditioning and training

Augmentation regularization (config F in Table 2)

- To **prevent potential overfitting** that often plagues diffusion models with smaller datasets, we borrow an **augmentation pipeline** from the **GAN literature** [25].
- The pipeline consists of **various geometric transformations** (see Appendix F.2) that we **apply to a training image prior to adding noise.**
- To prevent the augmentations from leaking to the generated images, we provide the **augmentation parameters as a conditioning input to F_θ; during inference we set the them to zero** to guarantee that only non-augmented images are generated.

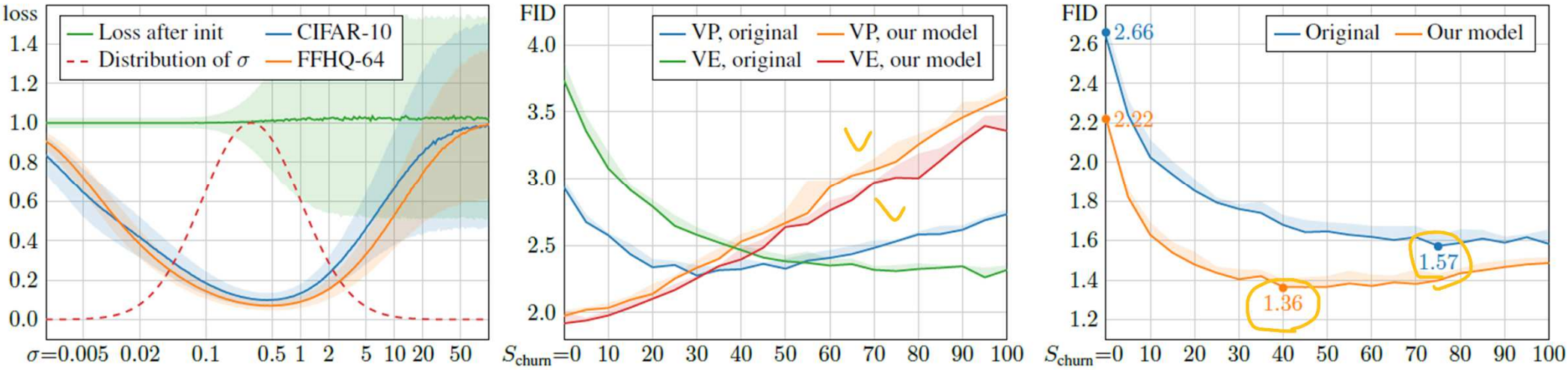
- Table 2 shows that data augmentation provides a consistent improvement (config F) that yields new state-of-the-art FIDs of 1.79 and 1.97 for conditional and unconditional CIFAR-10, beating the previous records of 1.85 [45] and 2.10 [53].

[25] T. Karras, M. Aittala, J. Hellsten, S. Laine, J. Lehtinen, and T. Aila. Training generative adversarial networks with limited data. In Proc. NeurIPS, 2020

Stochastic sampling revisited

- Interestingly, the relevance of stochastic sampling appears to diminish as the model itself improves, as shown in Figure 5b,c.
- When using our training setup in CIFAR-10 (Figure 5b), the **best results** were obtained with **deterministic sampling**, and any amount of stochastic sampling was detrimental.

5. Preconditioning and training



(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.

Appendix A. Additional Results

Deterministic, Original sampler (DDIM); FID 2.91 NFE 250

Deterministic, Our sampler (Alg. 1); FID 2.66 NFE 79

Stochastic, Original sampler (iDDPM); FID 2.01 NFE 512

Stochastic, Our sampler (Alg. 2); FID 1.55 NFE 1023

Figure 6: Results for different samplers on class-conditional ImageNet [8] at 64x64 resolution, using the pre-trained ADM model by Dhariwal and Nichol [9]. The cases correspond to dots in Figures 2c and 4c.

Appendix A. Additional Results

Deterministic, Our sampler & training configuration; FID 2.23 NFE 79

Stochastic, Our sampler & training configuration; FID 1.36 NFE 511

Figure 7: Results for our training configuration on class-conditional ImageNet [8] at 64x64 resolution, using our deterministic and stochastic samplers.

Appendix A. Additional Results

FFHQ, Original training (config A), VP; FID 3.39 NFE 79

FFHQ, Original training (config A), VE; FID 25.95 NFE 79

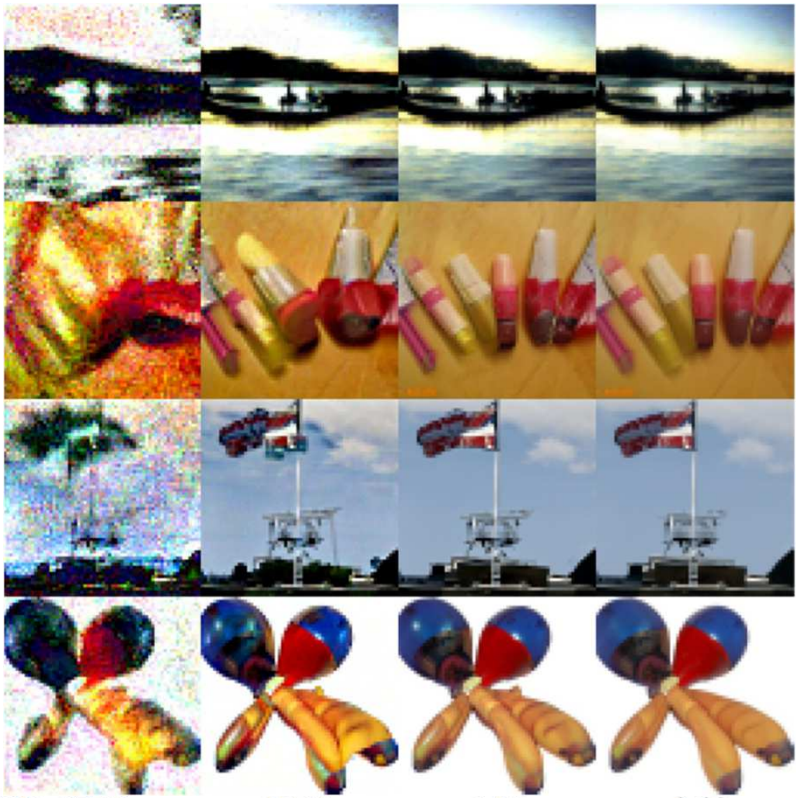
FFHQ, Our training (config F), VP; FID 2.39 NFE 79

FFHQ, Our training (config F), VE; FID 2.53 NFE 79

Figure 11: Results for different training configurations on FFHQ [27] at 64x64 resolution, using our deterministic sampler with the same set of latent codes (x_0) in each case.

Appendix A. Additional Results

Class-conditional ImageNet-64, Pre-trained



Class-conditional CIFAR-10, Our training, VP

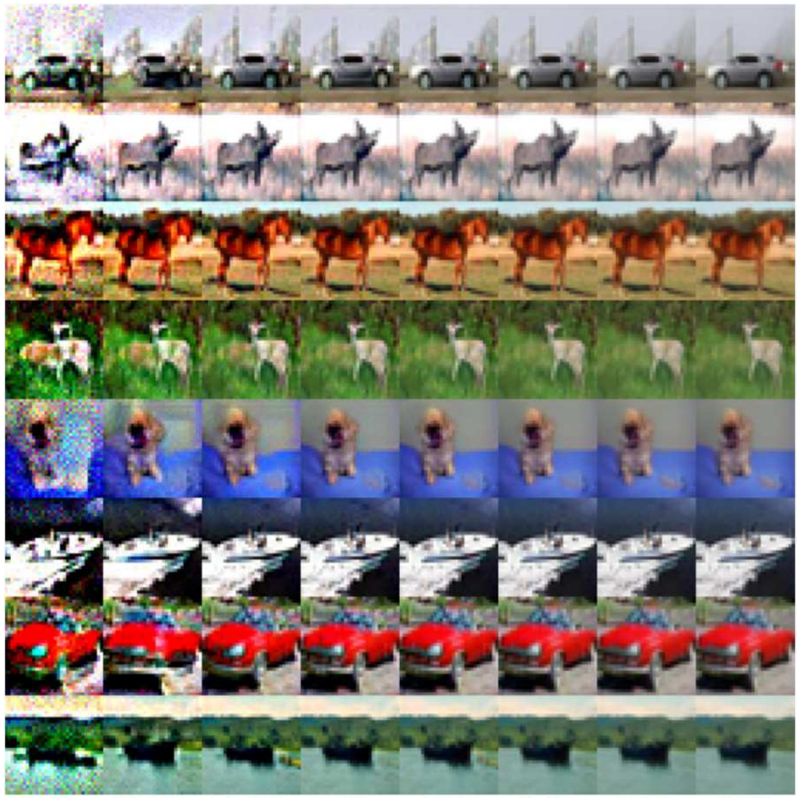


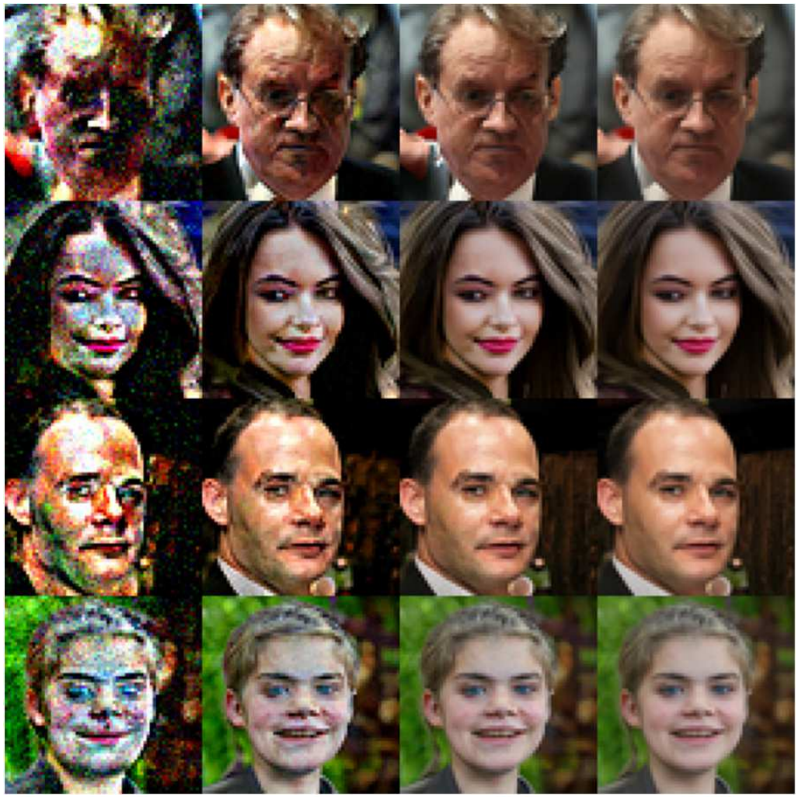
Figure 12: Image quality and FID as a function of NFE using our deterministic sampler.

At 32x32 resolution, reasonable image quality is reached around NFE = 13, but FID keeps improving until NFE = 35.

At 64x64 resolution, reasonable image quality is reached around NFE = 19, but FID keeps improving until NFE = 79

Appendix A. Additional Results

Unconditional FFHQ, Our training, VP



FID 142.34 29.22 5.13 **2.39**
NFE 7 11 19 79

Unconditional AFHQv2, Our Training, VP

61.57 13.68 3.00 **1.96**
7 11 19 79

Figure 12: Image quality and FID as a function of NFE using our deterministic sampler.

At 32x32 resolution, reasonable image quality is reached around NFE = 13, but FID keeps improving until NFE = 35.

At 64x64 resolution, reasonable image quality is reached around NFE = 19, but FID keeps improving until NFE = 79

Appendix A. Additional Results

Table 3: **Evaluation of our improvements to deterministic sampling**. The values correspond to the curves shown in Figure 2. We summarize each curve with two key values: **the lowest observed FID for any NFE ("FID")**, **and the lowest NFE whose FID is within 3% of the lowest FID ("NFE")**. The values marked with "–" are identical to the ones above them, because our sampler uses the same $\sigma(t)$ and $s(t)$ as DDIM.

	Unconditional CIFAR-10 at 32×32				Class-conditional ImageNet-64	
	VP		VE			
Sampling method	FID ↓	NFE ↓	FID ↓	NFE ↓	FID ↓	NFE ↓
Original sampler [49, 9]	2.85	256	5.45	8192	2.85	250
Our Algorithm 1	**2.79**	512	4.78	8192	2.73	384
+ Heun & our t_i	2.88	255	4.23	191	**2.64**	**79**
+ Our $\sigma(t)$ & $s(t)$	2.93	**35**	3.73	**27**	–	–
Black-box RK45	2.94	115	**3.69**	93	2.66	131

Table 4: **Evaluation and ablations of our improvements to stochastic sampling**. The values correspond to the curves shown in Figure 4.

	Unconditional CIFAR-10 at 32×32				Class-conditional ImageNet-64	
	VP		VE			
Sampling method	FID ↓	NFE ↓	FID ↓	NFE ↓	FID ↓	NFE ↓
Deterministic baseline (Alg. 1)	2.93	**35**	3.73	**27**	2.64	**79**
Alg. 2, $S_{\text{tmin,tmax}} = [0, \infty]$, $S_{\text{noise}} = 1$	2.69	95	2.97	383	1.86	383
Alg. 2, $S_{\text{tmin,tmax}} = [0, \infty]$	2.54	127	2.51	511	1.63	767
Alg. 2, $S_{\text{noise}} = 1$	2.52	95	2.84	191	1.84	255
Alg. 2, Optimal settings	**2.27**	383	**2.23**	767	**1.55**	511
Previous work [49, 9]	2.55	768	2.46	1024	2.01	384

Appendix B. Derivation of formulas

B1. Original ODE / SDE formulation from previous work (Song et al.)

However, f and g are of little practical interest in themselves, whereas **the marginal distributions are of utmost importance in terms of training the model in the first place, bootstrapping the sampling process, and understanding how the ODE behaves in practice**.

Given that **the idea of the probability flow ODE is to match a particular set of marginal distributions**, it makes sense to **treat the marginal distributions as first-class citizens and define the ODE directly based on $\sigma(t)$ and $s(t)$, eliminating the need for $f(t)$ and $g(t)$.**

Song et al. [49] define their forward SDE (Eq. 5 in [49]) as

$$\mathrm{d}\boldsymbol{x} = \boldsymbol{f}(\boldsymbol{x}, t)\,\mathrm{d}t + g(t)\,\mathrm{d}\omega_t, \tag{9}$$

where ω_t is the standard Wiener process and $\boldsymbol{f}(\cdot, t) : \mathbb{R}^d \to \mathbb{R}^d$ and $g(\cdot) : \mathbb{R} \to \mathbb{R}$ are the drift and diffusion coefficients, respectively, where d is the dimensionality of the dataset. These coefficients are selected differently for the variance preserving (VP) and variance exploding (VE) formulations, and $\boldsymbol{f}(\cdot)$ is always of the form $f(\boldsymbol{x}, t) = f(t)\,\boldsymbol{x}$, where $f(\cdot) : \mathbb{R} \to \mathbb{R}$. Thus, the SDE can be equivalently written as

$$\mathrm{d}\boldsymbol{x} = f(t)\,\boldsymbol{x}\,\mathrm{d}t + g(t)\,\mathrm{d}\omega_t. \tag{10}$$

The perturbation kernels of this SDE (Eq. 29 in [49]) have the general form

$$p_{0t}\big(\boldsymbol{x}(t) \mid \boldsymbol{x}(0)\big) = \mathcal{N}\big(\boldsymbol{x}(t);\ s(t)\,\boldsymbol{x}(0),\ s(t)^2\,\sigma(t)^2\,\mathbf{I}\big), \tag{11}$$

where $\mathcal{N}(\boldsymbol{x}; \boldsymbol{\mu}, \boldsymbol{\Sigma})$ denotes the probability density function of $\mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$ evaluated at $\boldsymbol{x}$,

$$s(t) = \exp\left(\int_0^t f(\xi)\,\mathrm{d}\xi\right), \quad \text{and} \quad \sigma(t) = \sqrt{\int_0^t \frac{g(\xi)^2}{s(\xi)^2}\,\mathrm{d}\xi}. \tag{12}$$

The marginal distribution $p_t(\boldsymbol{x})$ is obtained by integrating the perturbation kernels over $\boldsymbol{x}(0)$:

$$p_t(\boldsymbol{x}) = \int_{\mathbb{R}^d} p_{0t}(\boldsymbol{x} \mid \boldsymbol{x}_0)\,p_{\text{data}}(\boldsymbol{x}_0)\,\mathrm{d}\boldsymbol{x}_0. \tag{13}$$

Song et al. [49] define the probability flow ODE (Eq. 13 in [49]) so that it obeys this same $p_t(\boldsymbol{x})$:

$$\mathrm{d}\boldsymbol{x} = \left[f(t)\,\boldsymbol{x} - \tfrac{1}{2}\,g(t)^2\,\nabla_{\boldsymbol{x}} \log p_t(\boldsymbol{x})\right]\,\mathrm{d}t. \tag{14}$$

Appendix B. Derivation of formulas

B2. Our ODE formulation (Eq. 1 and Eq. 4)

Let us start by expressing the marginal distribution of Eq. 13 in closed form:

$$p_t(\boldsymbol{x}) = \int_{\mathbb{R}^d} p_{0t}(\boldsymbol{x} \mid \boldsymbol{x}_0)\, p_{\text{data}}(\boldsymbol{x}_0)\, \mathrm{d}\boldsymbol{x}_0 \tag{15}$$

$$= \int_{\mathbb{R}^d} p_{\text{data}}(\boldsymbol{x}_0) \left[\mathcal{N}\big(\boldsymbol{x};\ s(t)\,\boldsymbol{x}_0,\ s(t)^2\,\sigma(t)^2\,\mathbf{I}\big)\right] \mathrm{d}\boldsymbol{x}_0 \tag{16}$$

$$= \int_{\mathbb{R}^d} p_{\text{data}}(\boldsymbol{x}_0) \left[s(t)^{-d}\,\mathcal{N}\big(\boldsymbol{x}/s(t);\ \boldsymbol{x}_0,\ \sigma(t)^2\,\mathbf{I}\big)\right] \mathrm{d}\boldsymbol{x}_0 \tag{17}$$

$$= s(t)^{-d} \int_{\mathbb{R}^d} p_{\text{data}}(\boldsymbol{x}_0)\, \mathcal{N}\big(\boldsymbol{x}/s(t);\ \boldsymbol{x}_0,\ \sigma(t)^2\,\mathbf{I}\big)\, \mathrm{d}\boldsymbol{x}_0 \tag{18}$$

$$= s(t)^{-d} \left[p_{\text{data}} * \mathcal{N}\big(\mathbf{0},\ \sigma(t)^2\,\mathbf{I}\big)\right]\big(\boldsymbol{x}/s(t)\big), \tag{19}$$

where $p_a * p_b$ denotes the convolution of probability density functions p_a and p_b. The expression inside the brackets corresponds to a mollified version of p_{data} obtained by adding i.i.d. Gaussian noise to the samples. Let us denote this distribution by $p(\boldsymbol{x};\sigma)$:

$$p(\boldsymbol{x};\sigma) = p_{\text{data}} * \mathcal{N}\big(\mathbf{0},\ \sigma(t)^2\,\mathbf{I}\big) \quad \text{and} \quad p_t(\boldsymbol{x}) = s(t)^{-d}\, p\big(\boldsymbol{x}/s(t);\sigma(t)\big). \tag{20}$$

We can now express the probability flow ODE (Eq. 14) using $p(\boldsymbol{x};\sigma)$ instead of $p_t(\boldsymbol{x})$:

$$\mathrm{d}\boldsymbol{x} = \big[f(t)\boldsymbol{x} - \tfrac{1}{2}\, g(t)^2\, \nabla_{\boldsymbol{x}} \log\big[p_t(\boldsymbol{x})\big]\big]\ \mathrm{d}t \tag{21}$$

$$= \big[f(t)\boldsymbol{x} - \tfrac{1}{2}\, g(t)^2\, \nabla_{\boldsymbol{x}} \log\big[s(t)^{-d}\, p\big(\boldsymbol{x}/s(t);\sigma(t)\big)\big]\big]\ \mathrm{d}t \tag{22}$$

$$= \big[f(t)\boldsymbol{x} - \tfrac{1}{2}\, g(t)^2\, \big[\nabla_{\boldsymbol{x}} \log s(t)^{-d} + \nabla_{\boldsymbol{x}} \log p\big(\boldsymbol{x}/s(t);\sigma(t)\big)\big]\big]\ \mathrm{d}t \tag{23}$$

$$= \big[f(t)\boldsymbol{x} - \tfrac{1}{2}\, g(t)^2\, \nabla_{\boldsymbol{x}} \log p\big(\boldsymbol{x}/s(t);\sigma(t)\big)\big]\ \mathrm{d}t. \tag{24}$$

Appendix B. Derivation of formulas

B2. Our ODE formulation (Eq. 1 and Eq. 4)

Next, let us rewrite $f(t)$ in terms of $s(t)$ based on Eq. 12:

$$\exp\left(\int_0^t f(\xi)\,\mathrm{d}\xi\right) = s(t) \tag{25}$$

$$\int_0^t f(\xi)\,\mathrm{d}\xi = \log s(t) \tag{26}$$

$$\mathrm{d}\left[\int_0^t f(\xi)\,\mathrm{d}\xi\right]/\mathrm{d}t = \mathrm{d}\left[\log s(t)\right]/\mathrm{d}t \tag{27}$$

$$f(t) = \dot{s}(t)/s(t). \tag{28}$$

Similarly, we can also rewrite $g(t)$ in terms of $\sigma(t)$:

$$\sqrt{\int_0^t \frac{g(\xi)^2}{s(\xi)^2}\,\mathrm{d}\xi} = \sigma(t) \tag{29}$$

$$\int_0^t \frac{g(\xi)^2}{s(\xi)^2}\,\mathrm{d}\xi = \sigma(t)^2 \tag{30}$$

$$\mathrm{d}\left[\int_0^t \frac{g(\xi)^2}{s(\xi)^2}\,\mathrm{d}\xi\right]/\mathrm{d}t = \mathrm{d}\left[\sigma(t)^2\right]/\mathrm{d}t \tag{31}$$

$$g(t)^2/s(t)^2 = 2\,\dot{\sigma}(t)\,\sigma(t) \tag{32}$$

$$g(t)/s(t) = \sqrt{2\,\dot{\sigma}(t)\,\sigma(t)} \tag{33}$$

$$g(t) = s(t)\,\sqrt{2\,\dot{\sigma}(t)\,\sigma(t)}. \tag{34}$$

Appendix B. Derivation of formulas

B2. Our ODE formulation (Eq. 1 and Eq. 4)

Finally, substitute f (Eq. 28) and g (Eq. 34) into the ODE of Eq. 24:

$$\begin{aligned}
\mathrm{d}\boldsymbol{x} &= \left[[f(t)]\,\boldsymbol{x} - \tfrac{1}{2}\,[g(t)]^2\,\nabla_{\boldsymbol{x}} \log p\big(\boldsymbol{x}/s(t);\sigma(t)\big)\right]\,\mathrm{d}t && (35)\\
&= \left[\big[\dot{s}(t)/s(t)\big]\,\boldsymbol{x} - \tfrac{1}{2}\left[s(t)\sqrt{2\,\dot{\sigma}(t)\,\sigma(t)}\right]^2 \nabla_{\boldsymbol{x}} \log p\big(\boldsymbol{x}/s(t);\sigma(t)\big)\right]\,\mathrm{d}t && (36)\\
&= \left[\big[\dot{s}(t)/s(t)\big]\,\boldsymbol{x} - \tfrac{1}{2}\left[2\,s(t)^2\,\dot{\sigma}(t)\,\sigma(t)\right] \nabla_{\boldsymbol{x}} \log p\big(\boldsymbol{x}/s(t);\sigma(t)\big)\right]\,\mathrm{d}t && (37)\\
&= \left[\frac{\dot{s}(t)}{s(t)}\,\boldsymbol{x} - s(t)^2\,\dot{\sigma}(t)\,\sigma(t)\,\nabla_{\boldsymbol{x}} \log p\left(\frac{\boldsymbol{x}}{s(t)};\sigma(t)\right)\right]\,\mathrm{d}t. && (38)
\end{aligned}$$

Thus we have obtained Eq. 4 in the main paper, and Eq. 1 is recovered by setting $s(t) = 1$:

$$\mathrm{d}\boldsymbol{x} = -\dot{\sigma}(t)\,\sigma(t)\,\nabla_{\boldsymbol{x}} \log p\big(\boldsymbol{x};\sigma(t)\big)\,\mathrm{d}t. \qquad (39)$$

Our formulation (Eq. 4) highlights the fact that every realization of the probability flow ODE is simply a reparameterization of the same canonical ODE; changing $\sigma(t)$ corresponds to reparameterizing t, whereas changing $s(t)$ corresponds to reparameterizing $\boldsymbol{x}$.

Appendix B. Derivation of formulas

B.3 Denoising score matching (Eq. 2 and Eq. 3)

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

then $\nabla_{\boldsymbol{x}}\log p(\boldsymbol{x};\sigma) = \big(D(\boldsymbol{x};\sigma)-\boldsymbol{x}\big)/\sigma^2$, (3)

$\boldsymbol{y}$: a training image, $\boldsymbol{n}$: noise

For the sake of completeness, we derive the connection between score matching and denoising for a finite dataset. For a more general treatment and further background on the topic, see Hyvärinen [22] and Vincent [54].

Let us assume that our training set consists of a finite number of samples $\{\boldsymbol{y}_1,\ldots,\boldsymbol{y}_Y\}$. This implies $p_{\text{data}}(\boldsymbol{x})$ is represented by a mixture of Dirac delta distributions:

$$p_{\text{data}}(\boldsymbol{x}) = \frac{1}{Y}\sum_{i=1}^{Y}\delta(\boldsymbol{x}-\boldsymbol{y}_i), \quad (40)$$

which allows us to also express $p(\boldsymbol{x};\sigma)$ in closed form based on Eq. 20:

$$\begin{aligned}
p(\boldsymbol{x};\sigma) &= p_{\text{data}} * \mathcal{N}\big(\mathbf{0},\,\sigma(t)^2\,\mathbf{I}\big) && (41)\\
&= \int_{\mathbb{R}^d} p_{\text{data}}(\boldsymbol{x}_0)\,\mathcal{N}\big(\boldsymbol{x};\,\boldsymbol{x}_0,\,\sigma^2\,\mathbf{I}\big)\,\mathrm{d}\boldsymbol{x}_0 && (42)\\
&= \int_{\mathbb{R}^d}\left[\frac{1}{Y}\sum_{i=1}^{Y}\delta(\boldsymbol{x}_0-\boldsymbol{y}_i)\right]\mathcal{N}\big(\boldsymbol{x};\,\boldsymbol{x}_0,\,\sigma^2\,\mathbf{I}\big)\,\mathrm{d}\boldsymbol{x}_0 && (43)\\
&= \frac{1}{Y}\sum_{i=1}^{Y}\int_{\mathbb{R}^d}\mathcal{N}\big(\boldsymbol{x};\,\boldsymbol{x}_0,\,\sigma^2\,\mathbf{I}\big)\,\delta(\boldsymbol{x}_0-\boldsymbol{y}_i)\,\mathrm{d}\boldsymbol{x}_0 && (44)\\
&= \frac{1}{Y}\sum_{i=1}^{Y}\mathcal{N}\big(\boldsymbol{x};\,\boldsymbol{y}_i,\,\sigma^2\,\mathbf{I}\big). && (45)
\end{aligned}$$

Appendix B. Derivation of formulas

B.3 Denoising score matching (Eq. 2 and Eq. 3)

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

then $\nabla_{\boldsymbol{x}}\log p(\boldsymbol{x};\sigma) = \big(D(\boldsymbol{x};\sigma)-\boldsymbol{x}\big)/\sigma^2$, (3)

$\boldsymbol{y}$: a training image, $\boldsymbol{n}$: noise

Let us now consider the denoising score matching loss of Eq. 2. By expanding the expectations, we can rewrite the formula as an integral over the noisy samples $\boldsymbol{x}$:

$$\begin{aligned}
\mathcal{L}(D;\sigma) &= \mathbb{E}_{\boldsymbol{y}\sim p_{\text{data}}}\,\mathbb{E}_{\boldsymbol{n}\sim\mathcal{N}(0,\sigma^2\mathbf{I})}\,\big\|D(\boldsymbol{y}+\boldsymbol{n};\sigma)-\boldsymbol{y}\big\|_2^2 && (46)\\
&= \mathbb{E}_{\boldsymbol{y}\sim p_{\text{data}}}\,\mathbb{E}_{\boldsymbol{x}\sim\mathcal{N}(\boldsymbol{y},\sigma^2\mathbf{I})}\,\big\|D(\boldsymbol{x};\sigma)-\boldsymbol{y}\big\|_2^2 && (47)\\
&= \mathbb{E}_{\boldsymbol{y}\sim p_{\text{data}}}\int_{\mathbb{R}^d}\mathcal{N}(\boldsymbol{x};\,\boldsymbol{y},\,\sigma^2\,\mathbf{I})\,\big\|D(\boldsymbol{x};\sigma)-\boldsymbol{y}\big\|_2^2\,\mathrm{d}\boldsymbol{x} && (48)\\
&= \frac{1}{Y}\sum_{i=1}^{Y}\int_{\mathbb{R}^d}\mathcal{N}(\boldsymbol{x};\,\boldsymbol{y}_i,\,\sigma^2\,\mathbf{I})\,\big\|D(\boldsymbol{x};\sigma)-\boldsymbol{y}_i\big\|_2^2\,\mathrm{d}\boldsymbol{x} && (49)\\
&= \int_{\mathbb{R}^d}\underbrace{\frac{1}{Y}\sum_{i=1}^{Y}\mathcal{N}(\boldsymbol{x};\,\boldsymbol{y}_i,\,\sigma^2\,\mathbf{I})\,\big\|D(\boldsymbol{x};\sigma)-\boldsymbol{y}_i\big\|_2^2}_{=:\,\mathcal{L}(D;\boldsymbol{x},\sigma)}\,\mathrm{d}\boldsymbol{x}. && (50)
\end{aligned}$$

Eq. 50 means that we can minimize $\mathcal{L}(D;\sigma)$ by minimizing $\mathcal{L}(D;\boldsymbol{x},\sigma)$ independently for each $\boldsymbol{x}$:

$$D(\boldsymbol{x};\sigma) = \arg\min_{D(\boldsymbol{x};\sigma)}\mathcal{L}(D;\boldsymbol{x},\sigma). \quad (51)$$

Appendix B. Derivation of formulas

B.3 Denoising score matching (Eq. 2 and Eq. 3)

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

then $\nabla_{\boldsymbol{x}}\log p(\boldsymbol{x};\sigma) = \big(D(\boldsymbol{x};\sigma)-\boldsymbol{x}\big)/\sigma^2$, (3)

$\boldsymbol{y}$: a training image, $\boldsymbol{n}$: noise

This is a convex optimization problem; its solution is uniquely identified by setting the gradient w.r.t. $D(\boldsymbol{x};\sigma)$ to zero:

$$\mathbf{0} = \nabla_{D(\boldsymbol{x};\sigma)}\Big[\mathcal{L}(D;\boldsymbol{x},\sigma)\Big] \quad (52)$$

$$\mathbf{0} = \nabla_{D(\boldsymbol{x};\sigma)}\left[\frac{1}{Y}\sum_{i=1}^{Y}\mathcal{N}(\boldsymbol{x};\,\boldsymbol{y}_i,\,\sigma^2\,\mathbf{I})\,\big\|D(\boldsymbol{x};\sigma)-\boldsymbol{y}_i\big\|_2^2\right] \quad (53)$$

$$\mathbf{0} = \sum_{i=1}^{Y}\mathcal{N}(\boldsymbol{x};\,\boldsymbol{y}_i,\,\sigma^2\,\mathbf{I})\,\nabla_{D(\boldsymbol{x};\sigma)}\Big[\big\|D(\boldsymbol{x};\sigma)-\boldsymbol{y}_i\big\|_2^2\Big] \quad (54)$$

$$\mathbf{0} = \sum_{i=1}^{Y}\mathcal{N}(\boldsymbol{x};\,\boldsymbol{y}_i,\,\sigma^2\,\mathbf{I})\,\Big[2\,D(\boldsymbol{x};\sigma)-2\,\boldsymbol{y}_i\Big] \quad (55)$$

$$\mathbf{0} = \left[\sum_{i=1}^{Y}\mathcal{N}(\boldsymbol{x};\,\boldsymbol{y}_i,\,\sigma^2\,\mathbf{I})\right]D(\boldsymbol{x};\sigma)-\sum_{i=1}^{Y}\mathcal{N}(\boldsymbol{x};\,\boldsymbol{y}_i,\,\sigma^2\,\mathbf{I})\,\boldsymbol{y}_i \quad (56)$$

$$D(\boldsymbol{x};\sigma) = \frac{\sum_i\mathcal{N}(\boldsymbol{x};\,\boldsymbol{y}_i,\,\sigma^2\,\mathbf{I})\,\boldsymbol{y}_i}{\sum_i\mathcal{N}(\boldsymbol{x};\,\boldsymbol{y}_i,\,\sigma^2\,\mathbf{I})}, \quad (57)$$

which gives a closed-form solution for the ideal denoiser $D(\boldsymbol{x};\sigma)$. Note that Eq. 57 is feasible to compute in practice for small datasets — we show the results for CIFAR-10 in Figure 1b.

Appendix B. Derivation of formulas

B.3 Denoising score matching (Eq. 2 and Eq. 3)

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

then $\nabla_{\boldsymbol{x}}\log p(\boldsymbol{x};\sigma)=\big(D(\boldsymbol{x};\sigma)-\boldsymbol{x}\big)/\sigma^2$, (3)

$\boldsymbol{y}$: a training image, $\boldsymbol{n}$: noise

Next, let us consider the score of the distribution $p(\boldsymbol{x};\sigma)$ defined in Eq. 45:

$$\begin{aligned}
\nabla_{\boldsymbol{x}}\log p(\boldsymbol{x};\sigma) &= \frac{\nabla_{\boldsymbol{x}}p(\boldsymbol{x};\sigma)}{p(\boldsymbol{x};\sigma)} && (58)\\
&= \frac{\nabla_{\boldsymbol{x}}\left[\frac{1}{Y}\sum_i\mathcal{N}(\boldsymbol{x};\,\boldsymbol{y}_i,\,\sigma^2\,\mathbf{I})\right]}{\left[\frac{1}{Y}\sum_i\mathcal{N}(\boldsymbol{x};\,\boldsymbol{y}_i,\,\sigma^2\,\mathbf{I})\right]} && (59)\\
&= \frac{\sum_i\nabla_{\boldsymbol{x}}\mathcal{N}(\boldsymbol{x};\,\boldsymbol{y}_i,\,\sigma^2\,\mathbf{I})}{\sum_i\mathcal{N}(\boldsymbol{x};\,\boldsymbol{y}_i,\,\sigma^2\,\mathbf{I})}. && (60)
\end{aligned}$$

We can simplify the numerator of Eq. 60 further:

$$\begin{aligned}
\nabla_{\boldsymbol{x}}\mathcal{N}(\boldsymbol{x};\,\boldsymbol{y}_i,\,\sigma^2\,\mathbf{I}) &= \nabla_{\boldsymbol{x}}\left[(2\pi\sigma^2)^{-\frac{d}{2}}\exp\frac{\|\boldsymbol{x}-\boldsymbol{y}_i\|_2^2}{-2\,\sigma^2}\right] && (61)\\
&= (2\pi\sigma^2)^{-\frac{d}{2}}\nabla_{\boldsymbol{x}}\left[\exp\frac{\|\boldsymbol{x}-\boldsymbol{y}_i\|_2^2}{-2\,\sigma^2}\right] && (62)\\
&= \left[(2\pi\sigma^2)^{-\frac{d}{2}}\exp\frac{\|\boldsymbol{x}-\boldsymbol{y}_i\|_2^2}{-2\,\sigma^2}\right]\nabla_{\boldsymbol{x}}\left[\frac{\|\boldsymbol{x}-\boldsymbol{y}_i\|_2^2}{-2\,\sigma^2}\right] && (63)\\
&= \mathcal{N}(\boldsymbol{x};\,\boldsymbol{y}_i,\,\sigma^2\,\mathbf{I})\,\nabla_{\boldsymbol{x}}\left[\frac{\|\boldsymbol{x}-\boldsymbol{y}_i\|_2^2}{-2\,\sigma^2}\right] && (64)\\
&= \mathcal{N}(\boldsymbol{x};\,\boldsymbol{y}_i,\,\sigma^2\,\mathbf{I})\left[\frac{\boldsymbol{y}_i-\boldsymbol{x}}{\sigma^2}\right]. && (65)
\end{aligned}$$

Appendix B. Derivation of formulas

B.3 Denoising score matching (Eq. 2 and Eq. 3)

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

then $\nabla_{\boldsymbol{x}}\log p(\boldsymbol{x};\sigma) = \big(D(\boldsymbol{x};\sigma)-\boldsymbol{x}\big)/\sigma^2$, (3)

$\boldsymbol{y}$: a training image, $\boldsymbol{n}$: noise

Let us substitute the result back to Eq. 60:

$$\nabla_{\boldsymbol{x}}\log p(\boldsymbol{x};\sigma) = \frac{\sum_i \nabla_{\boldsymbol{x}}\mathcal{N}(\boldsymbol{x};\,\boldsymbol{y}_i,\,\sigma^2\mathbf{I})}{\sum_i \mathcal{N}(\boldsymbol{x};\,\boldsymbol{y}_i,\,\sigma^2\mathbf{I})} \quad (66)$$

$$= \frac{\sum_i \mathcal{N}(\boldsymbol{x};\,\boldsymbol{y}_i,\,\sigma^2\mathbf{I})\left[\frac{\boldsymbol{y}_i-\boldsymbol{x}}{\sigma^2}\right]}{\sum_i \mathcal{N}(\boldsymbol{x};\,\boldsymbol{y}_i,\,\sigma^2\mathbf{I})} \quad (67)$$

$$= \left(\frac{\sum_i \mathcal{N}(\boldsymbol{x};\,\boldsymbol{y}_i,\,\sigma^2\mathbf{I})\boldsymbol{y}_i}{\sum_i \mathcal{N}(\boldsymbol{x};\,\boldsymbol{y}_i,\,\sigma^2\mathbf{I})} - \boldsymbol{x}\right)/\sigma^2. \quad (68)$$

Notice that the fraction in Eq. 68 is identical to Eq. 57. We can thus equivalently write Eq. 68 as

$$\nabla_{\boldsymbol{x}}\log p(\boldsymbol{x};\sigma) = \big(D(\boldsymbol{x};\,\sigma)-\boldsymbol{x}\big)/\sigma^2, \quad (69)$$

which matches Eq. 3 in the main paper.

Appendix B. Derivation of formulas

B4. Evaluating our ODE in practice (Algorithm 1)

$$\nabla_{\boldsymbol{x}} \log p(\boldsymbol{x};\sigma) = \big(D(\boldsymbol{x};\sigma) - \boldsymbol{x}\big)/\sigma^2, \quad (3)$$

Time-dependent signal scaling

$$\mathrm{d}\boldsymbol{x} = \left[\frac{\dot{s}(t)}{s(t)}\,\boldsymbol{x} - s(t)^2\,\dot{\sigma}(t)\,\sigma(t)\,\nabla_{\boldsymbol{x}} \log p\left(\frac{\boldsymbol{x}}{s(t)};\sigma(t)\right)\right]\,\mathrm{d}t. \quad (4)$$

$$\mathrm{d}\boldsymbol{x} = -\dot{\sigma}(t)\,\sigma(t)\,\nabla_{\boldsymbol{x}} \log p\big(\boldsymbol{x};\sigma(t)\big)\,\mathrm{d}t, \quad \text{for } s(t) = 1$$

Let us consider $\boldsymbol{x}$ to be a scaled version of an original, non-scaled variable $\hat{\boldsymbol{x}}$ and substitute $\boldsymbol{x} = s(t)\,\hat{\boldsymbol{x}}$ into the score term that appears in our scaled ODE (Eq. 4):

$$\begin{aligned}
& \nabla_{\boldsymbol{x}} \log p\big(\boldsymbol{x}/s(t);\sigma(t)\big) && (70)\\
= \;& \nabla_{[s(t)\hat{\boldsymbol{x}}]} \log p\big([s(t)\,\hat{\boldsymbol{x}}]/s(t);\sigma(t)\big) && (71)\\
= \;& \nabla_{s(t)\hat{\boldsymbol{x}}} \log p\big(\hat{\boldsymbol{x}};\sigma(t)\big) && (72)\\
= \;& \tfrac{1}{s(t)}\nabla_{\hat{\boldsymbol{x}}} \log p\big(\hat{\boldsymbol{x}};\sigma(t)\big). && (73)
\end{aligned}$$

We can further rewrite this with respect to $D(\cdot)$ using Eq. 3:

$$\nabla_{\boldsymbol{x}} \log p\big(\boldsymbol{x}/s(t);\sigma(t)\big) = \tfrac{1}{s(t)\sigma(t)^2}\Big(D\big(\hat{\boldsymbol{x}};\sigma(t)\big) - \hat{\boldsymbol{x}}\Big). \quad (74)$$

Appendix B. Derivation of formulas

B4. Evaluating our ODE in practice (Algorithm 1)

$$\nabla_{\boldsymbol{x}} \log p(\boldsymbol{x};\sigma) = \big(D(\boldsymbol{x};\sigma) - \boldsymbol{x}\big)/\sigma^2, \quad (3)$$

Time-dependent signal scaling

$$\mathrm{d}\boldsymbol{x} = \left[\frac{\dot{s}(t)}{s(t)}\,\boldsymbol{x} - s(t)^2\,\dot{\sigma}(t)\,\sigma(t)\,\nabla_{\boldsymbol{x}} \log p\left(\frac{\boldsymbol{x}}{s(t)};\sigma(t)\right)\right] \mathrm{d}t. \quad (4)$$

$$\mathrm{d}\boldsymbol{x} = -\dot{\sigma}(t)\,\sigma(t)\,\nabla_{\boldsymbol{x}} \log p\big(\boldsymbol{x};\sigma(t)\big)\,\mathrm{d}t, \quad \text{for } s(t) = 1$$

Let us now substitute Eq. 74 into Eq. 4, approximating the ideal denoiser $D(\cdot)$ with our trained model $D_\theta(\cdot)$:

$$\begin{aligned}
\mathrm{d}\boldsymbol{x} &= \left[\dot{s}(t)\,\boldsymbol{x}/s(t) - s(t)^2\,\dot{\sigma}(t)\,\sigma(t)\left[\tfrac{1}{s(t)\sigma(t)^2}\Big(D_\theta\big(\hat{\boldsymbol{x}};\sigma(t)\big) - \hat{\boldsymbol{x}}\Big)\right]\right]\mathrm{d}t && (75)\\
&= \left[\tfrac{\dot{s}(t)}{s(t)}\,\boldsymbol{x} - \tfrac{\dot{\sigma}(t)s(t)}{\sigma(t)}\Big(D_\theta\big(\hat{\boldsymbol{x}};\sigma(t)\big) - \hat{\boldsymbol{x}}\Big)\right]\mathrm{d}t. && (76)
\end{aligned}$$

Finally, backsubstitute $\hat{\boldsymbol{x}} = \boldsymbol{x}/s(t)$:

$$\begin{aligned}
\mathrm{d}\boldsymbol{x} &= \left[\tfrac{\dot{s}(t)}{s(t)}\,\boldsymbol{x} - \tfrac{\dot{\sigma}(t)s(t)}{\sigma(t)}\Big(D_\theta\big([\hat{\boldsymbol{x}}];\sigma(t)\big) - [\hat{\boldsymbol{x}}]\Big)\right]\mathrm{d}t && (77)\\
&= \left[\tfrac{\dot{s}(t)}{s(t)}\,\boldsymbol{x} - \tfrac{\dot{\sigma}(t)s(t)}{\sigma(t)}\Big(D_\theta\big([\boldsymbol{x}/s(t)];\sigma(t)\big) - [\boldsymbol{x}/s(t)]\Big)\right]\mathrm{d}t && (78)\\
&= \left[\tfrac{\dot{s}(t)}{s(t)}\,\boldsymbol{x} - \tfrac{\dot{\sigma}(t)s(t)}{\sigma(t)}D_\theta\big(\boldsymbol{x}/s(t);\sigma(t)\big) + \tfrac{\dot{\sigma}(t)}{\sigma(t)}\,\boldsymbol{x}\right]\mathrm{d}t && (79)\\
&= \left[\Big(\tfrac{\dot{\sigma}(t)}{\sigma(t)} + \tfrac{\dot{s}(t)}{s(t)}\Big)\,\boldsymbol{x} - \tfrac{\dot{\sigma}(t)s(t)}{\sigma(t)}D_\theta\big(\boldsymbol{x}/s(t);\sigma(t)\big)\right]\mathrm{d}t. && (80)
\end{aligned}$$

We can equivalenty write Eq. 80 as

$$\mathrm{d}\boldsymbol{x}/\mathrm{d}t = \left(\frac{\dot{\sigma}(t)}{\sigma(t)} + \frac{\dot{s}(t)}{s(t)}\right)\boldsymbol{x} - \frac{\dot{\sigma}(t)s(t)}{\sigma(t)}D_\theta\left(\frac{\boldsymbol{x}}{s(t)};\sigma(t)\right), \quad (81)$$

matching lines 4 and 7 of Algorithm 1.

Appendix B. Derivation of formulas

B5. Our SDE formulation (Eq. 6)

$$\mathrm{d}\boldsymbol{x}_{\pm} = \underbrace{-\dot{\sigma}(t)\sigma(t)\nabla_{\boldsymbol{x}} \log p\big(\boldsymbol{x};\sigma(t)\big)\,\mathrm{d}t}_{\text{probability flow ODE (Eq. 1)}} \pm \underbrace{\underbrace{\beta(t)\sigma(t)^2\nabla_{\boldsymbol{x}} \log p\big(\boldsymbol{x};\sigma(t)\big)\,\mathrm{d}t}_{\text{deterministic noise decay}} + \underbrace{\sqrt{2\beta(t)}\sigma(t)\,\mathrm{d}\omega_t}_{\text{noise injection}}}_{\text{Langevin diffusion SDE}}, \tag{6}$$

Derive the SDE of Eq. 6 by the following strategy;

- The desired marginal densities $p(x;\sigma(t))$ are convolutions of the data density p_{data} and an isotropic Gaussian density with standard deviation $\sigma(t)$ (see Eq. 20). Hence, considered as a function of the time t, the density evolves according to a heat diffusion PDE with time-varying diffusivity. As a first step, we find this PDE.
- We then use the Fokker–Planck equation to recover a family of SDEs for which the density evolves according to this PDE. Eq. 6 is obtained from a suitable parametrization of this family.

B.5.1 Generating the marginals by heat diffusion

We consider the time evolution of a probability density $q(\boldsymbol{x},t)$. Our goal is to find a PDE whose solution with the initial value $q(\boldsymbol{x},0) := p_{\text{data}}(\boldsymbol{x})$ is $q(\boldsymbol{x},t) = p\big(\boldsymbol{x},\sigma(t)\big)$. That is, the PDE should reproduce the marginals we postulate in Eq. 20.

The desired marginals are convolutions of p_{data} with isotropic normal distributions of time-varying standard deviation $\sigma(t)$, and as such, can be generated by the heat equation with time-varying diffusivity $\kappa(t)$. The situation is most conveniently analyzed in the Fourier domain, where the marginal densities are simply pointwise products of a Gaussian function and the transformed data density. To find the diffusivity that induces the correct standard deviations, we first write down the heat equation PDE:

$$\frac{\partial q(\boldsymbol{x},t)}{\partial t} = \kappa(t)\Delta_{\boldsymbol{x}} q(\boldsymbol{x},t). \tag{82}$$

The Fourier transformed counterpart of Eq. 82, where the transform is taken along the $\boldsymbol{x}$-dimension, is given by

$$\frac{\partial \hat{q}(\boldsymbol{\nu},t)}{\partial t} = -\kappa(t)|\boldsymbol{\nu}|^2 \hat{q}(\boldsymbol{\nu},t). \tag{83}$$

The target solution $q(\boldsymbol{x},t)$ and its Fourier transform $\hat{q}(\boldsymbol{\nu},t)$ are given by Eq. 20:

$$q(\boldsymbol{x},t) = p\big(\boldsymbol{x};\sigma(t)\big) = p_{\text{data}}(\boldsymbol{x}) * \mathcal{N}\big(\mathbf{0},\ \sigma(t)^2\,\mathbf{I}\big) \tag{84}$$

$$\hat{q}(\boldsymbol{\nu},t) = \hat{p}_{\text{data}}(\boldsymbol{\nu})\ \exp\Big(-\tfrac{1}{2}\,|\boldsymbol{\nu}|^2\,\sigma(t)^2\Big). \tag{85}$$

Appendix B. Derivation of formulas

B5. Our SDE formulation (Eq. 6)

$$\mathrm{d}\boldsymbol{x}_{\pm} = \underbrace{-\dot{\sigma}(t)\sigma(t)\nabla_{\boldsymbol{x}} \log p\big(\boldsymbol{x};\sigma(t)\big)\,\mathrm{d}t}_{\text{probability flow ODE (Eq. 1)}} \pm \underbrace{\underbrace{\beta(t)\sigma(t)^2\nabla_{\boldsymbol{x}} \log p\big(\boldsymbol{x};\sigma(t)\big)\,\mathrm{d}t}_{\text{deterministic noise decay}} + \underbrace{\sqrt{2\beta(t)}\sigma(t)\,\mathrm{d}\omega_t}_{\text{noise injection}}}_{\text{Langevin diffusion SDE}}, \tag{6}$$

B.5.1

- The desired marginal densities $p(x;\sigma(t))$ are convolutions of the data density p_{data} and an isotropic Gaussian density with standard deviation $\sigma(t)$ (see Eq. 20). Hence, considered as a function of the time t, the density evolves according to a heat diffusion PDE with time-varying diffusivity. As a first step, we find this PDE.

B.5.1 Generating the marginals by heat diffusion

Differentiating the target solution along the time axis, we have

$$\frac{\partial \hat{q}(\boldsymbol{\nu},t)}{\partial t} = -\dot{\sigma}(t)\sigma(t)\,|\boldsymbol{\nu}|^2\,\hat{p}_{\text{data}}(\boldsymbol{\nu})\,\exp\left(-\tfrac{1}{2}\,|\boldsymbol{\nu}|^2\,\sigma(t)^2\right) \tag{86}$$

$$= -\dot{\sigma}(t)\sigma(t)\,|\boldsymbol{\nu}|^2\,\hat{q}(\boldsymbol{\nu},t). \tag{87}$$

Eqs. 83 and 87 share the same left hand side. Equating them allows us to solve for $\kappa(t)$ that generates the desired evolution:

$$-\kappa(t)|\boldsymbol{\nu}|^2\hat{q}(\boldsymbol{\nu},t) = -\dot{\sigma}(t)\sigma(t)\,|\boldsymbol{\nu}|^2\,\hat{q}(\boldsymbol{\nu},t) \tag{88}$$

$$\kappa(t) = \dot{\sigma}(t)\sigma(t). \tag{89}$$

To summarize, the desired marginal densities corresponding to noise levels $\sigma(t)$ are generated by the PDE

$$\frac{\partial q(\boldsymbol{x},t)}{\partial t} = \dot{\sigma}(t)\sigma(t)\Delta_{\boldsymbol{x}} q(\boldsymbol{x},t) \tag{90}$$

from the initial density $q(\boldsymbol{x},0) = p_{\text{data}}(\boldsymbol{x})$.

Appendix B. Derivation of formulas

B5. Our SDE formulation (Eq. 6)

$$\mathrm{d}\boldsymbol{x}_{\pm} = \underbrace{-\dot{\sigma}(t)\sigma(t)\nabla_{\boldsymbol{x}} \log p\big(\boldsymbol{x};\sigma(t)\big)\,\mathrm{d}t}_{\text{probability flow ODE (Eq. 1)}} \qquad (6)$$

$$\pm \underbrace{\underbrace{\beta(t)\sigma(t)^2\nabla_{\boldsymbol{x}} \log p\big(\boldsymbol{x};\sigma(t)\big)\,\mathrm{d}t}_{\text{deterministic noise decay}} + \underbrace{\sqrt{2\beta(t)}\sigma(t)\,\mathrm{d}\omega_t}_{\text{noise injection}}}_{\text{Langevin diffusion SDE}},$$

B.5.2

- We then use the Fokker–Planck equation to recover a family of SDEs for which the density evolves according to this PDE. Eq. 6 is obtained from a suitable parametrization of this family.

B.5.2 Derivation of our SDE

Given an SDE

$$\mathrm{d}\boldsymbol{x} = \boldsymbol{f}(\boldsymbol{x},t)\,\mathrm{d}t + \boldsymbol{g}(\boldsymbol{x},t)\,\mathrm{d}\omega_t, \qquad (91)$$

the Fokker–Planck PDE describes the time evolution of its solution probability density $r(\boldsymbol{x},t)$ as

$$\frac{\partial r(\boldsymbol{x},t)}{\partial t} = -\nabla_{\boldsymbol{x}} \cdot \big(\boldsymbol{f}(\boldsymbol{x},t)\,r(\boldsymbol{x},t)\big) + \tfrac{1}{2}\nabla_{\boldsymbol{x}}\nabla_{\boldsymbol{x}} : \big(\mathbf{D}(\boldsymbol{x},t)\,r(\boldsymbol{x},t)\big), \qquad (92)$$

where $\mathbf{D}_{ij} = \sum_k \boldsymbol{g}_{ik}\boldsymbol{g}_{jk}$ is the *diffusion tensor*. We consider the special case $\boldsymbol{g}(\boldsymbol{x},t) = g(t)\,\mathbf{I}$ of $\boldsymbol{x}$-independent white noise addition, whereby the equation simplifies to

$$\frac{\partial r(\boldsymbol{x},t)}{\partial t} = -\nabla_{\boldsymbol{x}} \cdot \big(\boldsymbol{f}(\boldsymbol{x},t)\,r(\boldsymbol{x},t)\big) + \tfrac{1}{2}\,g(t)^2\,\Delta_{\boldsymbol{x}} r(\boldsymbol{x},t). \qquad (93)$$

We are seeking an SDE whose solution density is described by the PDE in Eq. 90. Setting $r(\boldsymbol{x},t) = q(\boldsymbol{x},t)$ and equating Eqs. 93 and 90, we find the sufficient condition that the SDE must satisfy

$$-\nabla_{\boldsymbol{x}} \cdot \big(\boldsymbol{f}(\boldsymbol{x},t)\,q(\boldsymbol{x},t)\big) + \tfrac{1}{2}\,g(t)^2\,\Delta_{\boldsymbol{x}} q(\boldsymbol{x},t) = \dot{\sigma}(t)\,\sigma(t)\,\Delta_{\boldsymbol{x}} q(\boldsymbol{x},t) \qquad (94)$$

$$\nabla_{\boldsymbol{x}} \cdot \big(\boldsymbol{f}(\boldsymbol{x},t)\,q(\boldsymbol{x},t)\big) = \Big(\tfrac{1}{2}\,g(t)^2 - \dot{\sigma}(t)\,\sigma(t)\Big)\,\Delta_{\boldsymbol{x}} q(\boldsymbol{x},t). \qquad (95)$$