Diffusion Study Group #8

11/5/2022

Tanishq Abraham

Perception Prioritized Training of Diffusion Models

Elucidating the Design Space of Diffusion-Based Generative Models

Motivation

Diffusion models are a powerful framework for neural image synthesis and even useful in other domains. How can we further improve the image/distribution quality, training cost, and generation speed?

The literature is dense on theory, design choices are made to ensure the models are on a solid theoretical footing.

However, this approach has a danger of obscuring the available design space — a proposed model may appear as a tightly coupled package where no individual component can be modified without breaking the entire system

Expressing diffusion models in a common framework

Data distribution: $p_{\text{data}}(\boldsymbol{x})$ with standard deviation σ_{data}

Noisy data distributions: $p(\boldsymbol{x}; \sigma)$ (add Gaussian noise of standard deviation σ to the data)

For $\sigma_{\max} \gg \sigma_{\text{data}}$, indistinguishable from noise

Start at $\boldsymbol{x}_0 \sim \mathcal{N}(\mathbf{0}, \sigma_{\max}^2 \mathbf{I})$ and denoise into images $\boldsymbol{x}_i \sim p(\boldsymbol{x}_i; \sigma_i)$ with noise levels $\sigma_0 = \sigma_{\max} > \sigma_1 > \cdots > \sigma_N = 0$

The endpoint $\boldsymbol{x}_N$ is distributed according to the data

The ODE formulation

To treat the process as continuous, we define $\boldsymbol{x}_a \sim p\big(\boldsymbol{x}_a; \sigma(t_a)\big)$ which is a sample at timestep t_a which can be evolved to $\boldsymbol{x}_b \sim p\big(\boldsymbol{x}_b; \sigma(t_b)\big)$

The noise levels are now a continuous function $\sigma(t)$

This evolution can be performed by integrating the ODE:

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

where $\dot{\sigma}(t) = \frac{\mathrm{d}\sigma(t)}{\mathrm{d}t}$ and $\nabla_{\boldsymbol{x}} \log p\left(\boldsymbol{x}; \sigma(t)\right)$ is the score function

A forward step ($t_a \rightarrow t_b$) nudges the sample away from the data, at a rate that depends on the change in noise level. The reverse is true for a backward step ($t_b \rightarrow t_a$).

Denoising score matching

We define $D(\boldsymbol{x};\sigma)$ to be a denoiser function that is trained with the following objective:

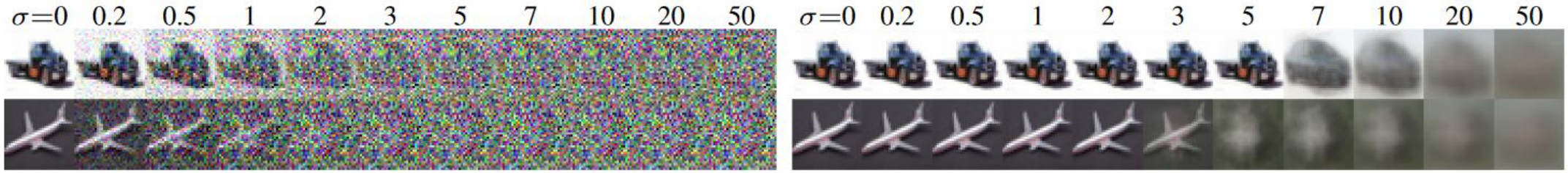
$$\mathbb{E}_{\boldsymbol{y}\sim p_{\text{data}}}\mathbb{E}_{\boldsymbol{n}\sim\mathcal{N}(\mathbf{0},\sigma^2\mathbf{I})}\left\|D(\boldsymbol{y}+\boldsymbol{n};\sigma)-\boldsymbol{y}\right\|_2^2$$

where $\boldsymbol{y}$ is a training image, and $\boldsymbol{n}$ is Gaussian noise. As we've discussed (ex: Tweedie's formula), the following is true:

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

The denoiser function is implemented as a neural network $D_\theta(\boldsymbol{x};\sigma)$

Ideal Denoiser



(a) Noisy images drawn from $p(\boldsymbol{x};\sigma)$ (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.

Time-dependent scaling

Some methods (such as VP ODEs) introduce an additional scale $s(t)$ such that $\boldsymbol{x} = s(t)\hat{\boldsymbol{x}}$ is a scaled down version of the original non-scaled sample $\hat{\boldsymbol{x}}$. The more generalized ODE taking this into account is:

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

Solution by discretization

Plugging the learned score function into the ODE, we can numerically solve the ODE in order to sample from our distribution.

Different ODE solvers (also called integration schemes) and discrete time steps (where $t_N = 0$) can be chosen

Usually Euler's method is used:

$$\boldsymbol{x}_{i+1} = \boldsymbol{x}_i + (t_{i+1} - t_i)d_i$$

where d_i is the right-hand side of the ODE evaluated at timestep t_i

Expressing diffusion models in a common framework - sampling

		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 $u_M = 0$ $u_{j-1} = \sqrt{\frac{u_j^2+1}{\max(\alpha_{j-1}/\alpha_j, C_1)} - 1}$	$\left(\sigma_{max}^{\frac{1}{\rho}} + \frac{i}{N-1}(\sigma_{min}^{\frac{1}{\rho}} - \sigma_{max}^{\frac{1}{\rho}})\right)^{\rho}$
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

Improvements to deterministic sampling

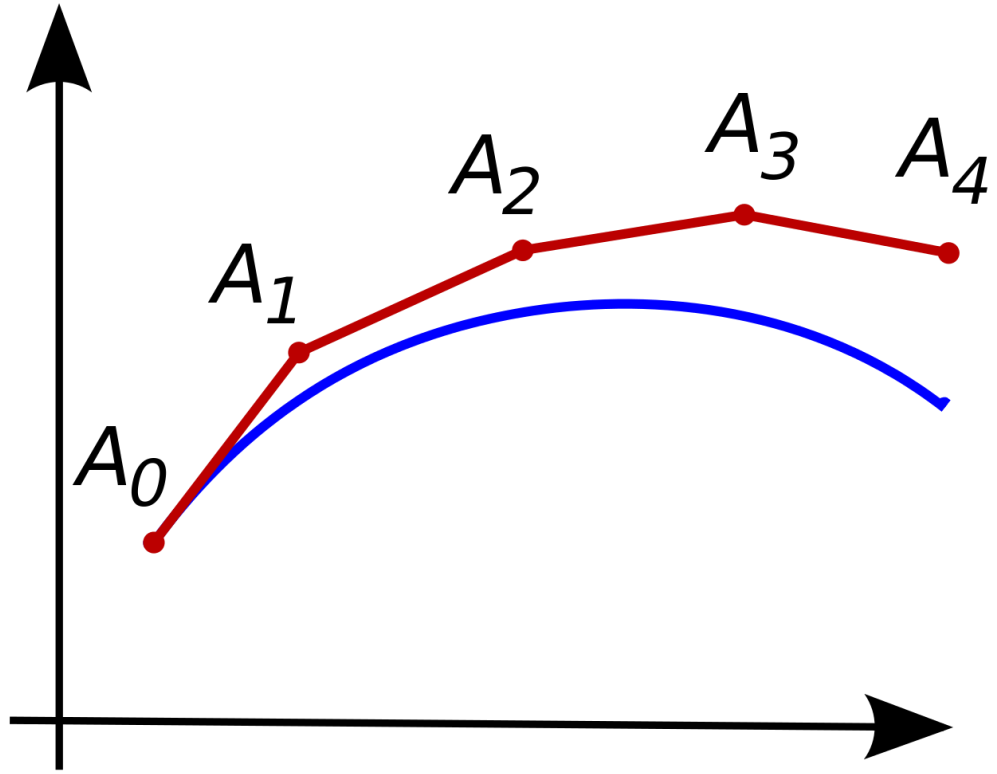
Choices related to the sampling process are largely independent of the other components, such as network architecture and training details.

From the viewpoint of the sampler, D_θ is a black box!

To demonstrate this, authors tried the original sampler, a re-implemented sampler, and a new sampler with a model trained under the VP ODE (Score SDE paper) VE ODE (Score SDE paper), and iDDPM (ADM paper) frameworks.

Discretization and higher-order integrators

Truncation error by ODEs (accumulates over timesteps):



Discretization and higher-order integrators

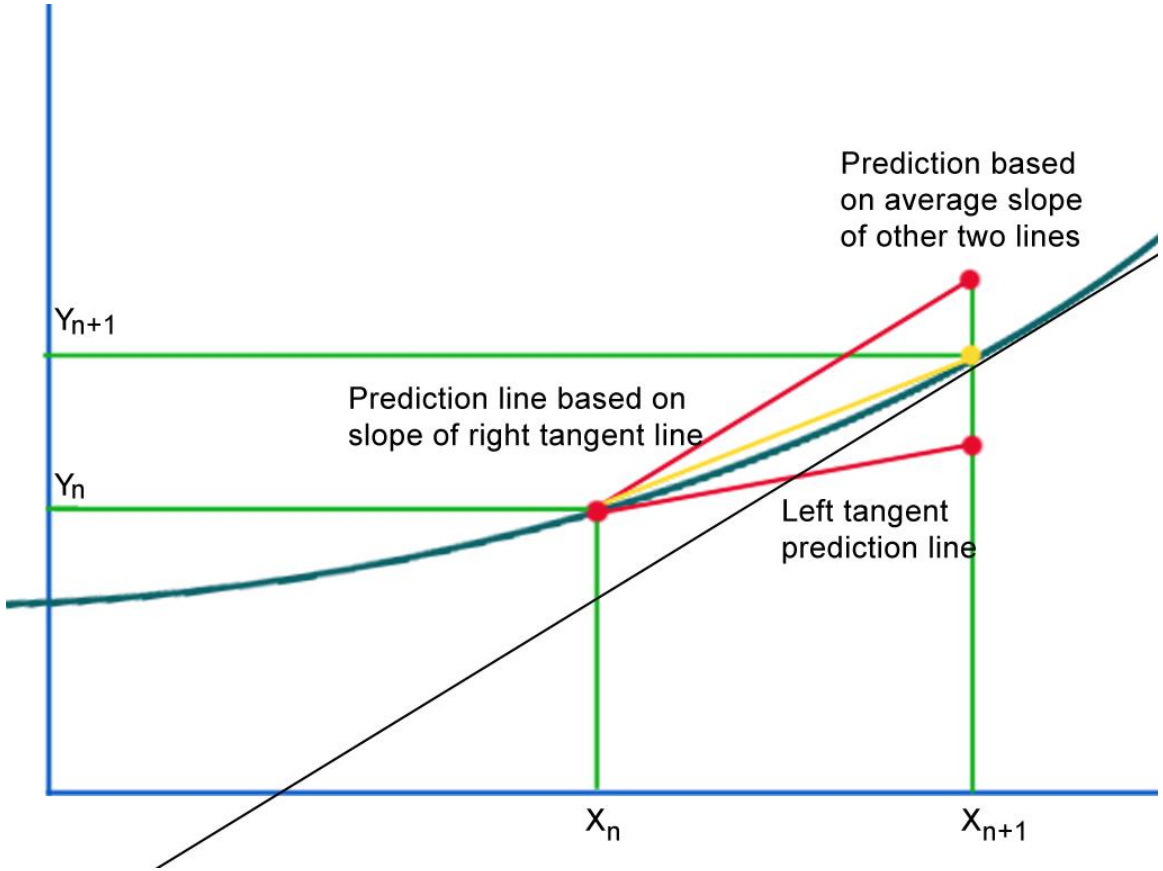
Euler's method - local truncation error scales with $\mathcal{O}(h^2)$

Heun's method (second-order method) – local truncation error scales with $\mathcal{O}(h^3)$

$$\tilde{\boldsymbol{x}}_{i+1} = \boldsymbol{x}_i + (t_{i+1} - t_i) d_i(\boldsymbol{x}_i, t_i)$$
$$d_i' = d_i(\tilde{\boldsymbol{x}}_{i+1}, t_{i+1})$$
$$x_{i+1} = x_i + \frac{(t_{i+1} - t_i)}{2}(d_i + d_i')$$

Adds an additional evaluation of D_θ per step

Heun's method



Heun's method

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

procedure HEUNSAMPLER($D_\theta(\boldsymbol{x};\sigma)$, $\sigma(t)$, $s(t)$, $t_{i\in\{0,\dots,N\}}$)

  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 $i \in \{0, \dots, N-1\}$ **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

Choosing time steps

Truncation error is very large for low noise levels and smaller for high noise levels (confirmed through empirical experiments in the appendix)

This indicates the best selection of step sizes should be if it decreased monotonically with decreasing σ. Therefore, $t_i = \sigma^{-1}(\sigma_i)$ and we set:

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

ρ controls how much the steps near $\sigma_{\min}$ are shortened at the expense of longer steps near $\sigma_{\max}$.

$\rho = 3$ equalizes the truncation error over all steps, but $5 < \rho = 7 < 10$ is better for sampling images. This suggests that errors near $\sigma_{\min}$ have a large impact.

Trajectory curvature and noise schedule

The shape of the ODE solution trajectories is defined by $s(t), \sigma(t)$

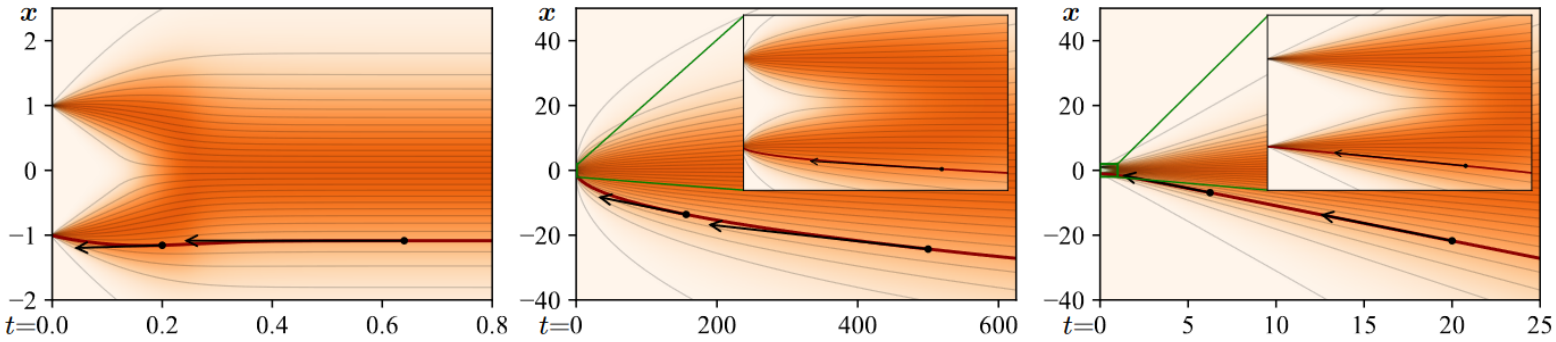
The best choice is $\sigma(t) = t$, $s(t) = 1$, which is also used in DDIM. The ODE becomes:

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

A single Euler step to $t = 0$ is the denoised image. The tangent of the solution trajectory always points to the denoiser output.

Trajectory curvature and noise schedule

This ODE provides largely linear trajectories



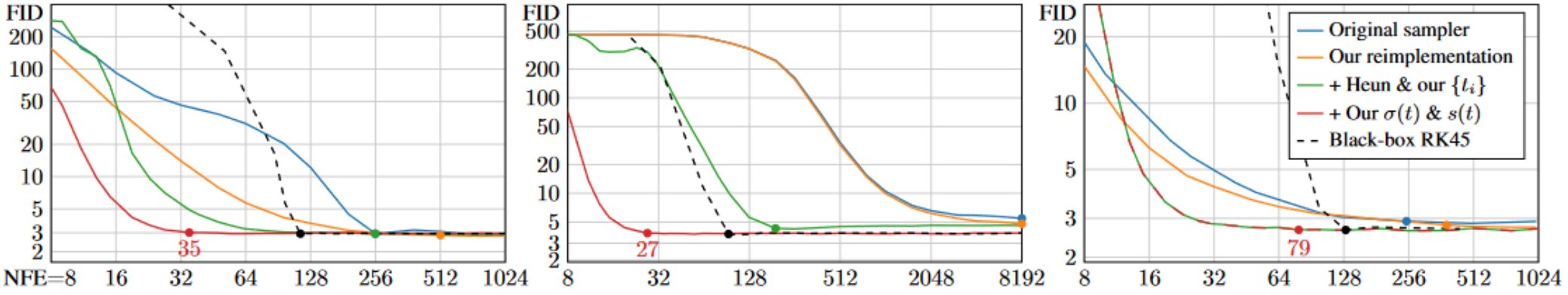
(a) Variance preserving ODE [49] (b) Variance exploding ODE [49] (c) DDIM [47] / Our ODE

Figure 3: A sketch of ODE curvature in 1D where p_{data} is two Dirac peaks at $\boldsymbol{x} = \pm 1$. 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.

Results – improvements to deterministic sampling

The re-implementations of the deterministic samplers in the unified framework resulted in consistently better results.

“The differences are explained by certain oversights in the original implementations as well as our more careful treatment of discrete noise levels in the case of DDIM”



(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.

Stochastic sampling

While they describe the same distributions, ODEs result in lower output quality compared to SDEs, so what is the role of stochasticity in practice?

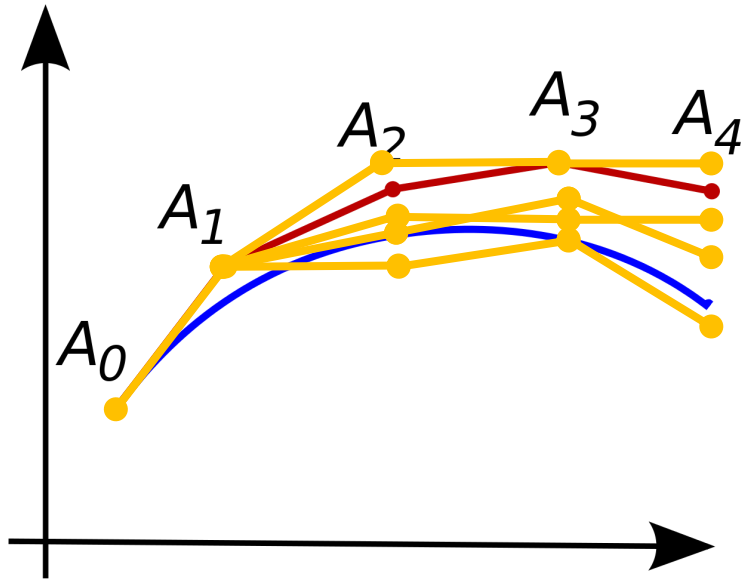
Let's examine the SDE where $\mathrm{d}\boldsymbol{x}_+$ is for the forward process and $\mathrm{d}\boldsymbol{x}_-$ is for the reverse process

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

Song et al. uses $\beta(t) = \dot{\sigma}(t)/\sigma(t)$

Stochastic sampling

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. On the other hand, approximating the Langevin term with discrete SDE solver steps introduces error in itself.



Proposed stochastic sampler

Go to higher noise level $\hat{t}_i = t_i + \gamma_i t_i$ and get a noisier sample

$$\hat{\boldsymbol{x}}_i = x_i + \sqrt{\hat{t}_i^2 - t_i^2}\,\boldsymbol{\epsilon}_i$$

Numerically solve the ODE back to t_{i+1}

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

procedure StochasticSampler($D_\theta(\boldsymbol{x};\sigma)$, $t_{i\in\{0,\dots,N\}}$, $\gamma_{i\in\{0,\dots,N-1\}}$, S_{noise})
  sample $\boldsymbol{x}_0 \sim \mathcal{N}(\mathbf{0},\ t_0^2\,\mathbf{I})$
  for $i \in \{0,\dots,N-1\}$ **do**
    ▷ $\gamma_i = \begin{cases} \min\left(\frac{S_{\text{churn}}}{N}, \sqrt{2}-1\right) & \text{if } t_i \in [S_{\text{tmin}}, S_{\text{tmax}}] \\ 0 & \text{otherwise} \end{cases}$
    sample $\boldsymbol{\epsilon}_i \sim \mathcal{N}(\mathbf{0},\ S_{\text{noise}}^2\,\mathbf{I})$
    $\hat{t}_i \leftarrow t_i + \gamma_i t_i$ ▷ Select temporarily increased noise level $\hat{t}_i$
    $\hat{\boldsymbol{x}}_i \leftarrow \boldsymbol{x}_i + \sqrt{\hat{t}_i^2 - t_i^2}\,\boldsymbol{\epsilon}_i$ ▷ Add new noise to move from t_i to $\hat{t}_i$
    $\boldsymbol{d}_i \leftarrow \left(\hat{\boldsymbol{x}}_i - D_\theta(\hat{\boldsymbol{x}}_i;\hat{t}_i)\right)/\hat{t}_i$ ▷ Evaluate $\mathrm{d}\boldsymbol{x}/\mathrm{d}t$ at $\hat{t}_i$
    $\boldsymbol{x}_{i+1} \leftarrow \hat{\boldsymbol{x}}_i + (t_{i+1} - \hat{t}_i)\boldsymbol{d}_i$ ▷ Take Euler step from $\hat{t}_i$ to t_{i+1}
    if $t_{i+1} \neq 0$ **then**
      $\boldsymbol{d}'_i \leftarrow \left(\boldsymbol{x}_{i+1} - D_\theta(\boldsymbol{x}_{i+1};t_{i+1})\right)/t_{i+1}$ ▷ Apply 2nd order correction
      $\boldsymbol{x}_{i+1} \leftarrow \hat{\boldsymbol{x}}_i + (t_{i+1} - \hat{t}_i)\left(\frac{1}{2}\boldsymbol{d}_i + \frac{1}{2}\boldsymbol{d}'_i\right)$
  return $\boldsymbol{x}_N$

Proposed stochastic sampler

Similar to the predictor-corrector sampler of Song et al.

Comparison to Euler-Maruyama (?)

Increasing the amount of stochasticity is effective in correcting errors made by earlier sampling steps but has drawbacks.

Excessive stochasticity results in gradual loss of detail in generations

There is also a drift toward oversaturated colors at very low and high noise levels.

Degradation is likely due to flaws in D_θ that violate the premises of Langevin diffusion

Proposed stochastic sampler

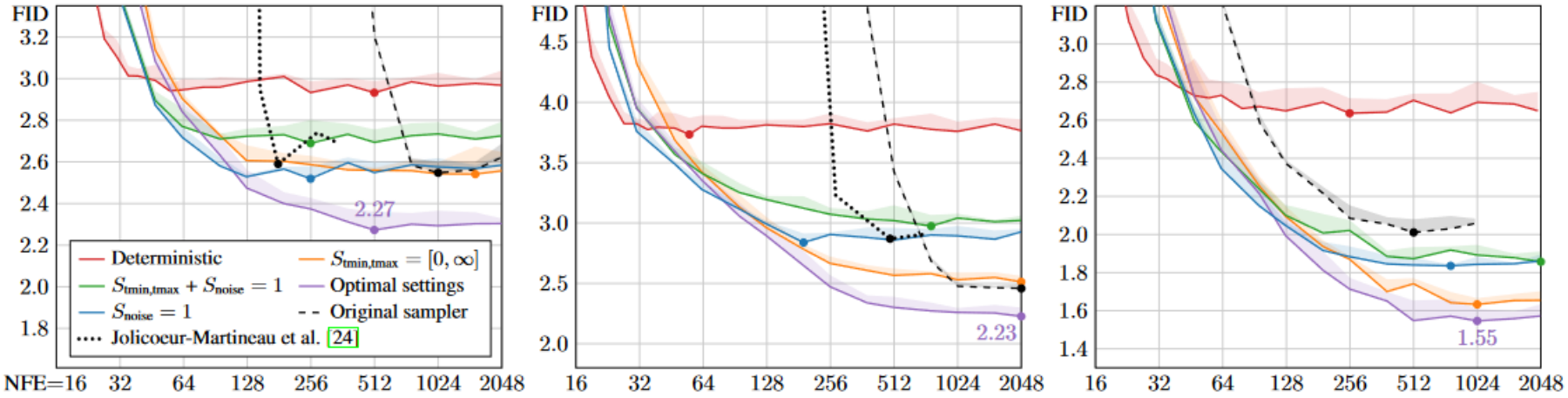
Empirical hacks used like:

- Only enabling stochasticity for $t_i \in [S_{\text{tmin}}, S_{\text{tmax}}]$
- Define $\gamma_i = S_{\text{churn}}/N$ 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 (?)
- Set S_{noise} slightly above 1 (inflate standard deviation of noise added)

The optimal value for these parameters were found by grid search on a case-by-case basis.

“[…]using stochastic sampling as the primary means of evaluating model improvements may inadvertently end up influencing the design choices related to model architecture and training.”

Proposed stochastic sampler



(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.

Preconditioning and Training Objective

It is advisable to keep input and output signal magnitudes fixed e.g. unit variance

Modeling D directly is therefore not ideal as the input $\boldsymbol{y} = \boldsymbol{x} + \boldsymbol{n}$ and output $\boldsymbol{x}$ magnitudes depend on the noise level σ

Most previous approaches train F_θ to predict $\boldsymbol{n}$ scaled to unit variance, giving us $D_\theta(\boldsymbol{x}; \sigma) = \boldsymbol{x} - \sigma F_\theta(\boldsymbol{x}; \sigma)$

However, at large σ, the predicted scaled $\boldsymbol{n}$ needs to be predicted exactly, as any minor errors will be amplified by a factor of σ**.** In this case, directly doing $D_\theta(\boldsymbol{x}; \sigma)$ might be optimal.

Preconditioning and Training Objective

An alternative denoiser that preconditions and post-processes the neural network $F_\theta(\cdot)$ is proposed in this work:

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

This results in the following training objective:

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

Main difference: $\sigma\sim p_{\text{train}}(\sigma)$ and $\lambda(\sigma)$ weights different noise levels in the loss

$c_{\text{in}}(\sigma)$ is selected such that the networks inputs have unit variance, $c_{\text{out}}(\sigma)$ is selected such that the network outputs have unit variance, and $c_{\text{skip}}(\sigma)$ is selected to minimize error amplification

Preconditioning and Training Objective

Comparing over different models:

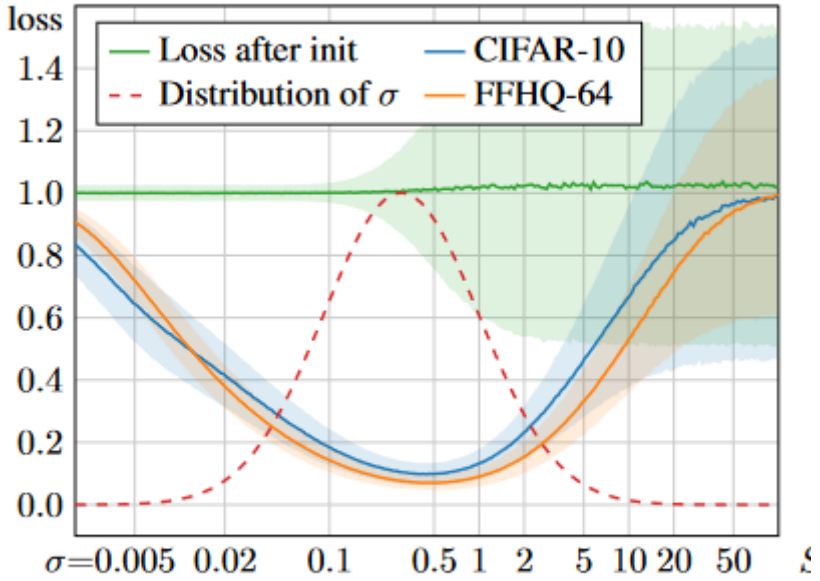
Network and preconditioning (Section 5)				
Architecture of F_θ	DDPM++	NCSN++	DDPM	(any)
Skip scaling $c_{\text{skip}}(\sigma)$	1	1	1	$\sigma_{\text{data}}^2/\left(\sigma^2+\sigma_{\text{data}}^2\right)$
Output scaling $c_{\text{out}}(\sigma)$	$-\sigma$	σ	$-\sigma$	$\sigma\cdot\sigma_{\text{data}}/\sqrt{\sigma_{\text{data}}^2+\sigma^2}$
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	$\sigma^{-1}(\sigma)\sim\mathcal{U}(\epsilon_{\text{t}},1)$	$\ln(\sigma)\sim\mathcal{U}(\ln(\sigma_{\min}),\ln(\sigma_{\max}))$	$\sigma=u_j,\ \ j\sim\mathcal{U}\{0,M-1\}$	$\ln(\sigma)\sim\mathcal{N}(P_{\text{mean}},P_{\text{std}}^2)$
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	$\beta_{\text{d}}=19.9,\beta_{\min}=0.1$ $\epsilon_{\text{s}}=10^{-3},\epsilon_{\text{t}}=10^{-5}$ $M=1000$	$\sigma_{\min}=0.02$ $\sigma_{\max}=100$	$\bar{\alpha}_j=\sin^2(\frac{\pi}{2}\frac{j}{M(C_2+1)})$ $C_1=0.001, C_2=0.008$ $M=1000, j_0=8^\dagger$	$\sigma_{\min}=0.002,\sigma_{\max}=80$ $\sigma_{\text{data}}=0.5,\rho=7$ $P_{\text{mean}}=-1.2, P_{\text{std}}=1.2$

* 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

Loss weighting and sampling

Balance the magnitude of the loss over different noise levels: $\lambda(\sigma) = 1/c_{\text{out}}(\sigma)^2$

The intermediate noise levels matter most so a log-normal distribution is chosen



(a) Loss & noise distribution

Augmentation

Augmentation is often an issue for smaller dataset

Borrow GAN augmentation pipeline (StyleGAN-ADA)

Use 6 geometric transformations, other transformations seem to be harmful for diffusion models.

The model is now conditioned on the augmentation during training and set to no augmentation conditioning during inference

Table 6: Our augmentation pipeline. Each training image undergoes a combined geometric transformation based on 8 random parameters that receive non-zero values with a certain probability. The model is conditioned with an additional 9-dimensional input vector derived from these parameters.

Augmentation	Transformation	Parameters	Prob.	Conditioning	Constants
x-flip	$\textsc{Scale2D}(1-2a_0,\ 1)$	$a_0 \sim \mathcal{U}\{0,1\}$	100%	a_0	$A_{\text{prob}} = 12\%$
y-flip	$\textsc{Scale2D}(1,\ 1-2a_1)$	$a_1 \sim \mathcal{U}\{0,1\}$	A_{prob}	a_1	or 15%
Scaling	$\textsc{Scale2D}\big((A_{\text{scale}})^{a_2},$ $(A_{\text{scale}})^{a_2}\big)$	$a_2 \sim \mathcal{N}(0,1)$	A_{prob}	a_2	$A_{\text{scale}} = 2^{0.2}$
Rotation	$\textsc{Rotate2D}(-a_3)$	$a_3 \sim \mathcal{U}(-\pi,\pi)$	A_{prob}	$\cos a_3 - 1$ $\sin a_3$	
Anisotropy	$\textsc{Rotate2D}(a_4)$ $\textsc{Scale2D}\big((A_{\text{aniso}})^{a_5},$ $1/(A_{\text{aniso}})^{a_5}\big)$ $\textsc{Rotate2D}(-a_4)$	$a_4 \sim \mathcal{U}(-\pi,\pi)$ $a_5 \sim \mathcal{N}(0,1)$	A_{prob}	$a_5\ \cos a_4$ $a_5\ \sin a_4$	$A_{\text{aniso}} = 2^{0.2}$
Translation	$\textsc{Translate2D}\big((A_{\text{trans}})a_6,$ $(A_{\text{trans}})a_7\big)$	$a_6 \sim \mathcal{N}(0,1)$ $a_7 \sim \mathcal{N}(0,1)$	A_{prob}	a_6 a_7	$A_{\text{trans}} = 1/8$

Results – training improvements

Preconditioning doesn't improve FID per se, but makes training more robust

P2 weighting provides improvement but only improves $\lambda(\sigma)$ and not $p_{\text{train}}(\sigma)$

New SOTA on CIFAR10 (VP – DDPM++, VE – NCSN++)

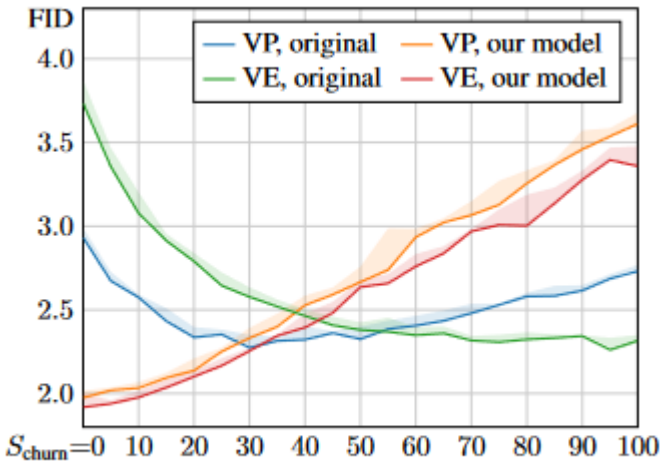
Table 2: Evaluation of our training improvements. The starting point (config A) is VP & VE using our **deterministic** sampler. At the end (configs E,F), VP & VE only differ in the architecture of F_θ.

	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

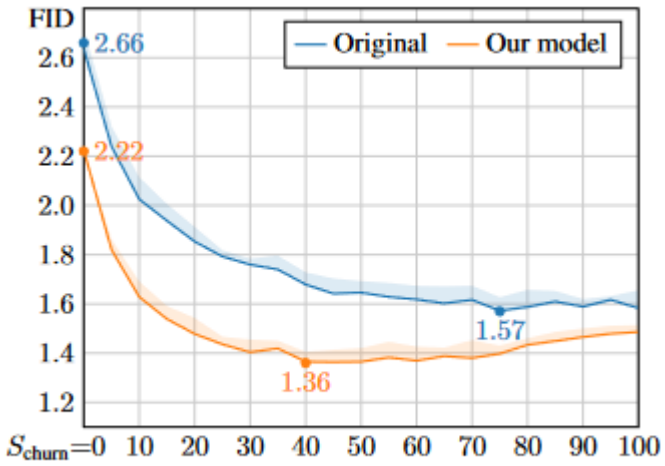
Results – training improvements

On CIFAR10, stochastic sampling worsens results! On ImageNet 64x64 stochastic sampling helps, indicating benefit of stochastic sampling for more diverse datasets

New SOTA on ImageNet 64x64 (ADM architecture, no aug reg)



(b) Stochasticity on CIFAR-10



(c) Stochasticity on ImageNet-64

Generations