Elucidating the Design Space of Diffusion-Based Generative Models

Junoh Kang

Computer Vision Laboratory
ECE, Seoul National University
junoh.kang@snu.ac.kr

The Reasons of Reviewing This Paper

- EDM has limited theoretical novelty, and its contributions are mainly engineering-oriented.
- In deep learning, practical implementation is just as important as theoretical support.
- When proposing new paradigms without any baseline codes, engineering skills are essential to bring the paradigms into the real world.
- The suggestions in the paper may not be optimal, but they are theoretically or empirically supported.
- I am not an engineering-oriented person and I want to learn such reasonings from the paper.

Contents

Reformulate Diffusion Process

Reformulate Diffusion Process

Previous formulation [6]

Song et al. [5] defines forward SDE as

$$d\mathbf{x}_t = f(t)\mathbf{x}_t dt + g(t) d\mathbf{w}_t. \tag{1}$$

As a consequence, the marginal distirbution at time t becomes

$$p(\mathbf{x}_t|\mathbf{x}_0) = \mathcal{N}(\mathbf{x}_t; s(t)\mathbf{x}_0, s(t)^2\sigma(t)^2\mathrm{I}), \tag{2}$$

where $s(t) = \exp(\int_0^t f(\xi)d\xi)$ and $\sigma(t) = \sqrt{\int_0^t \frac{g(\xi)^2}{s(\xi)^2} d\xi}$.

Song et al. [5] **indirectly defines** the marginal distribution by $f(\cdot)$ and $g(\cdot)$. However, the marginal distribution is the most important factor for training diffusion models.

Reformulate Diffusion Process

Suggested formulation

Instead of defining $f(\cdot)$ and $g(\cdot)$, EDM **directly defines** the marginal distribution by setting $s(\cdot)$ and $\sigma(\cdot)$:

$$p(\mathbf{x}_t) = s(t)^{-d} p(\mathbf{x}_t / s(t); \sigma(t)), \tag{3}$$

where $p(\mathbf{x}; \sigma) = [p_{\text{data}} * \mathcal{N}(0, \sigma^2 \mathrm{I})](\mathbf{x})$.

Then, the corresponding probability flow ODE is

$$d\mathbf{x}_t = [\dot{s}(t)/s(t) - s(t)^2 \dot{\sigma}(t)\sigma(t) \nabla_{\mathbf{x}} \log p(\mathbf{x}_t / s(t); \sigma(t))]\, dt. \tag{4}$$

Design Space of Diffusion Models

Design Space of Diffusion Models

Role of components

1. Generation by diffusion models can be interpreted as solving

$$d\mathbf{x}_t = f(\mathbf{x}_t, s(t), \sigma(t))dt. \tag{5}$$

2. Since $f(\mathbf{x}_t, s(t), \sigma(t))$ is not known, it is parametrized by a network $f_\theta(\mathbf{x}_t, s(t), \sigma(t))$. The inaccurate approximation on the target causes degradation. $\rightarrow$ **Better training.**
3. The solution at $t = 0$ given boundary condition at $t = T$ is

$$\mathbf{x}_0 = \mathbf{x}_T + \int_0^T f(\mathbf{x}_t, s(t), \sigma(t))dt. \tag{6}$$

   The integral is numerically calculated, which causes truncation error. $\rightarrow$ **Truncation-error-reducing ODE / Truncation-error-reducing algorithms, $\Phi(\cdot)$ / Distributing truncation error properly.**

Design Space of Diffusion Models

Classification of independent components

- Components in Training
  - Parametrization
  - Network preconditioning: $c_{\text{skip}}(\sigma)$, $c_{\text{out}}(\sigma)$, $c_{\text{in}}(\sigma)$, and $c_{\text{noise}}(\sigma)$
  - Loss weighting: $\lambda(t)$
  - Noise distribution
  - Augmentation
- Components in Deterministic Sampling
  - Truncation-error-reducing ODE: $s(t)$, $\sigma(t)$
  - Truncation-error-reducing algorithms: Higher-order inegrators, $\Phi(\cdot)$
  - Distributing truncation error properly: Discretizations $\{t_i\}_0^N$
- Components in Stochastic Sampling
  - Rate of replaced noises $\beta(t)$
  - Heuristics: $S_{\text{tmin}}, S_{\text{tmax}}, S_{\text{noise}}, S_{\text{churn}}$.

Improvements to Training

Training

Parametrization

For $s(t) = 1$, $D(\mathbf{x}_t, \sigma)$ is a denoiser which minimizes ℓ_2-norm with $\mathbf{x}_0$:

$$\mathbb{E}_{\mathbf{y} \sim p_{\text{data}}} \mathbb{E} || D(\mathbf{y} + \mathbf{n}) - \mathbf{y} ||_2^2. \tag{7}$$

The relation between a score function and the ideal denoiser is

$$\nabla_{\mathbf{x}} \log p(\mathbf{x}; \sigma) = (D(\mathbf{x}; \sigma) - \mathbf{x}) / \sigma^2. \tag{8}$$

EDM parametirzes the denoiser with a network.

Training

Parametrization

Benny et al. [1] observes that predicting the denoised output, $D(\mathbf{x}; \sigma)$, is eaiser for high noise level, while predicting the noise, $\mathbf{n}$, is eaiser for low noise level. Previous methods [3, 6, 2] usally predicts $\mathbf{n}$.

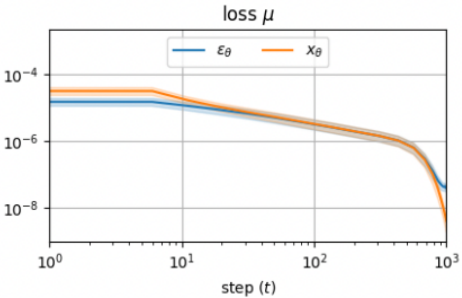


Figure 1: Loss comparison between predicting the denoised output or the added noise.

To predict $D(\mathbf{x};\sigma)$ or $\mathbf{n}$, or something in between according to the noise level, EDM parametrizes the denoiser function by

$$D_\theta(\mathbf{x};\sigma) = c_{\text{skip}}(\sigma)\mathbf{x} + c_{\text{out}}(\sigma)F_\theta(c_{\text{in}}(\sigma)\mathbf{x}; c_{\text{noise}}(\sigma)), \tag{9}$$

where F_θ is a neural network.

- c_{skip} modulates skip connection.
- c_{in} controls input scale.
- c_{out} controls output scale.
- c_{noise} maps noise level into a conditioning input.

Training

Network preconditioning

Then, the loss function (7) is

$$\mathbb{E}_{\sigma,\mathbf{y},\mathbf{b}}\left[\underbrace{\lambda(\sigma)c_{\text{out}}(\sigma)^2}_{\text{effective weight}}\,||\,\underbrace{F_\theta(c_{\text{in}}(\sigma)(\mathbf{y}+\mathbf{n};c_{\text{noise}}(\sigma)))}_{\text{network output}}-\underbrace{\frac{1}{c_{\text{out}}(\sigma)}(\mathbf{y}-c_{\text{skip}}(\sigma)(\mathbf{y}+\mathbf{n}))}_{\text{effective training target}}\,||_2^2\right]. \tag{10}$$

Training

Network preconditioning

1. $c_{\text{noise}}(\sigma) = \frac{1}{4}\log(\sigma)$ is chosen empirically.
2. Inputs should have unit variance.

$$\operatorname{Var}_{\mathbf{y},\mathbf{n}}\left[c_{\text{in}}(\sigma)(\mathbf{y}+\mathbf{n})\right] = 1 \tag{11}$$

$$\Leftrightarrow \quad c_{\text{in}}(\sigma) = 1/\sqrt{\sigma^2 + \sigma_{\text{data}}^2}. \tag{12}$$

3. Effective training target should have unit variance.

$$\operatorname{Var}_{\mathbf{y},\mathbf{n}}\left[\frac{1}{c_{\text{out}}}(\mathbf{y} - c_{\text{skip}}(\sigma)(\mathbf{y}+\mathbf{n}))\right] = 1 \tag{13}$$

$$\Leftrightarrow \quad c_{\text{out}}(\sigma)^2 = (1 - c_{\text{skip}}(\sigma))^2\sigma_{\text{data}}^2 + c_{\text{skip}}(\sigma)^2\sigma^2. \tag{14}$$

Training

Network preconditioning

4. $c_{\text{out}}(\sigma)$ should be small so that errors would not amplified.

$$c_{\text{skip}}(\sigma) = \underset{c_{\text{skip}}(\sigma)}{\arg\min}\, c_{\text{out}}(\sigma) \tag{15}$$

$$\Leftrightarrow \quad c_{\text{skip}}(\sigma) = \sigma_{\text{data}}^2 / (\sigma^2 + \sigma_{\text{data}}^2), \tag{16}$$

$$c_{\text{out}}(\sigma) = \sigma \cdot \sigma_{\text{data}} / \sqrt{\sigma^2 + \sigma_{\text{data}}^2}. \tag{17}$$

5. Effecitve weight should be uniform.

$$\lambda(\sigma) c_{\text{out}}(\sigma)^2 = 1 \tag{18}$$

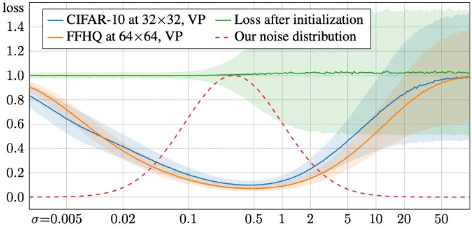
$$\Leftrightarrow \quad \lambda(\sigma) = (\sigma^2 + \sigma_{\text{data}}^2) / (\sigma \cdot \sigma_{\text{data}})^2. \tag{19}$$

6. Finally, the expected value of the loss at each noise level is 1. Moreover, the change of effective training target according to σ coincides to the observation of Benny et al. [1].

Training

Netowrk preconditioning & noise distribution

Noise distribution is chosen practically; at low noise levels, separating the small noise component is difficult, whereas at high noise levels, the correct answer approaches to dataset average.



(a) Training loss & noise distribution

Figure 2: Observed initial (green) and final loss per noise level. The shaded regions represent the standard deviation over 10k rondaom samples. EDM's proposed training sample density is shown by the dashed red curve.

Training

Augmentation

EDM uses augmentation pipeline from the GAN [4].

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 y-flip	$\text{SCALE2D}(1-2a_0,\ 1)$ $\text{SCALE2D}(1,\ 1-2a_1)$	$a_0 \sim \mathcal{U}\{0,1\}$ $a_1 \sim \mathcal{U}\{0,1\}$	100% A_{prob}	a_0 a_1	$A_{\text{prob}} = 12\%$ or 15%
Scaling	$\text{SCALE2D}\big((A_{\text{scale}})^{a_2},\ (A_{\text{scale}})^{a_2}\big)$	$a_2 \sim \mathcal{N}(0,1)$	A_{prob}	a_2	$A_{\text{scale}} = 2^{0.2}$
Rotation	$\text{ROTATE2D}(-a_3)$	$a_3 \sim \mathcal{U}(-\pi,\pi)$	A_{prob}	$\cos a_3 - 1$ $\sin a_3$	
Anisotropy	$\text{ROTATE2D}(a_4)$ $\text{SCALE2D}\big((A_{\text{aniso}})^{a_5},\ 1/(A_{\text{aniso}})^{a_5}\big)$ $\text{ROTATE2D}(-a_4)$	$a_4 \sim \mathcal{U}(-\pi,\pi)$ $a_5 \sim \mathcal{N}(0,1)$	A_{prob}	$a_5 \cos a_4$ $a_5 \sin a_4$	$A_{\text{aniso}} = 2^{0.2}$
Translation	$\text{TRANSLATE2D}\big((A_{\text{trans}})a_6,\ (A_{\text{trans}})a_7\big)$	$a_6 \sim \mathcal{N}(0,1)$ $a_7 \sim \mathcal{N}(0,1)$	A_{prob}	a_6 a_7	$A_{\text{trans}} = 1/8$

Training

Augmentation

1. Each augmentation is enabled with A_{prob}.
2. Draw a_i from each enabled augmentation and construct transformation matrix.
3. Pass data through $2\times$ supersampled high-quality Wavelet filters.
4. Construct a 9-dimensional conditioning input vector for non-leaking augmentation. This vector makes the network to perform auxiliary tasks.

Improvements to Determinisitic Sampling

Improvements to Determinisitic Sampling

Higher-order integrators & Discretization

For $s(t) = 1$ the ODE becomes

$$d\mathbf{x}_t = [-\dot{\sigma}(t)\sigma(t)\nabla_{\mathbf{x}} \log p(\mathbf{x}_t; \sigma(t))] \, dt. \tag{20}$$

With $\sigma(t) = t$ and denoiser, the ODE simplifies into

$$\begin{aligned} d\mathbf{x}_t/dt &= (\mathbf{x}_t - D(\mathbf{x}_t; t))/t && (21) \\ &:= f(\mathbf{x}_t, t) && (22) \end{aligned}$$

Improvements to Determinisitic Sampling

Truncation-error-reducing algorithms: Higher-order inegrators

Euler method approximates the integral by

$$\int_{t_i}^{t_{i-1}} f(\mathbf{x}_t, t)dt = (t_{i-1} - t_i)f(\mathbf{x}_{t_i}, t_i) + O(|t_{i-1} - t_i|^2). \tag{23}$$

Therefore, the total truncation error is $O(\max |t_{i-1} - t_i|)$.
Let $\hat{\mathbf{x}}_{t_{i-1}}$ is a solution obtained by Euler method. Then, Heun's method approximates the integral by

$$\int_{t_i}^{t_{i-1}} f(\mathbf{x}_t, t)dt = (t_{i-1} - t_i)(f(\mathbf{x}_{t_i}, t_i) + f(\hat{\mathbf{x}}_{t_{i-1}}, t_{i-1}))/2 + O(|t_{i-1} - t_i|^3). \tag{24}$$

Therefore, the total truncation error is $O(\max |t_{i-1} - t_i|^2)$. Huen's method decreases truncation error at the cost of one additional evaluation of the network.

Improvements to Determinisitic Sampling

Truncation-error-reducing algorithms: Higher-order inegrators

Algorithm 1 Deterministic sampling using Heun's 2[nd] 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 2[nd] 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

Improvements to Determinisitic Sampling

Distributing truncation error properly: Discretizations $\{t_i\}_0^N$

As long as using numerical integrators with limited computational resources, **truncation errors are inevitable.** In terms of obtaining ODE trajectories accurately, it is important to minimize total truncation errors. However, the interests of diffusion models at generation are **only the solutions at low noise levels**, it is reasonable to **focus on low noise levels.**

For discretization

$$\sigma_{i<N} = (\sigma_{\max}^{1/\rho} + \frac{i}{N-1}(\sigma_{\min}^{1/\rho} - \sigma_{\max}^{1/\rho}))^{\rho} = t_{N-i}, \sigma_N = 0, \tag{25}$$

increasing ρ results dense discretizations at low noise levels.

Improvements to Determinisitic Sampling

Distributing truncation error properly: Discretizations $\{t_i\}_0^N$

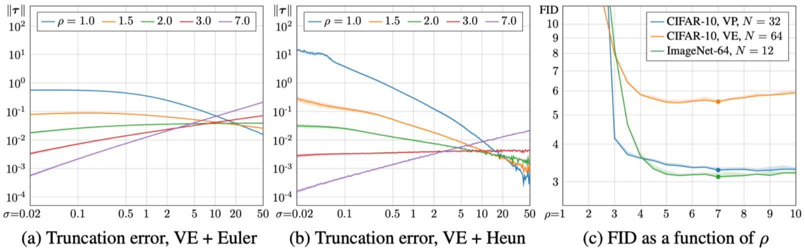


Figure 3: (a),(b) Local truncation error at different noise levels. (c) FID as a function of ρ.

$\rho = 3$ nearly equalizes the truncation error at each step as in Figure 3(a),(b). However, $\rho = 7$ generates better samples as in Figure 3(c).

Proper value of ρ may change according to the tasks. *e.g.*, Equalized truncation error is needed for solving ODE in both directions.

Improvements to Determinisitic Sampling

Truncation-error-reducing ODE: $s(t)$, $\sigma(t)$

Many integrators including Euler and Heun's method have small truncation errors if $f(\mathbf{x}_t, t)$ has **small curvature**, or is close to linear function.

$$\int_{t_i}^{t_{i-1}} f(\mathbf{x}_t, t)dt \approx \begin{cases} (t_{i-1} - t_i) f(\mathbf{x}_{t_i}, t_i) & \text{Euler method} \\ (t_{i-1} - t_i)(f(\mathbf{x}_{t_i}, t_i) + f(\hat{\mathbf{x}}_{t_{i-1}}, t_{i-1}))/2 & \text{Heun's method} \end{cases}$$

$s(t)$ and $\sigma(t)$ determine the shape of the ODE solution trajectories, which is closely related to linearity of the $f(\cdot)$.

Improvements to Determinisitic Sampling

Truncation-error-reducing ODE: $s(t)$, $\sigma(t)$

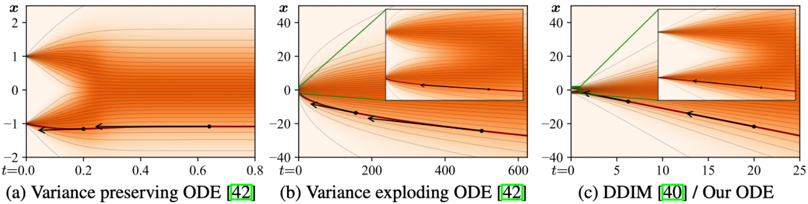


Figure 4: A sketch of ODE curvature in 1D where p_{data} is two Dirac peaks at $\mathbf{x} = \pm 1$. Axis is chosen to show $\sigma in[0, 25]$ and zoom in $\sigma in[0, 1]$.

$s(t) = 1$ and $\sigma(t) = t$ shows small curvature, while the tangent directs to the datapoints.

Stochastic Sampling

Stochastic Sampling

SDE formulation

EDM reformulates forward and backward SDE as a sum of the probability flow ODE and a varying-rate *Langevin diffusion* SDE:

$$d\mathbf{x}_{\pm} = \underbrace{-\dot{\sigma}(t)\sigma(t)\nabla_{\mathbf{x}} \log p(\mathbf{x}; \sigma(t))dt}_{\text{probability flow ODE}} \tag{26}$$

$$\pm \underbrace{\underbrace{\beta(t)\sigma(t)^2\nabla_{\mathbf{x}} \log p(\mathbf{x}; \sigma(t))dt}_{\text{deterministic noise decay}} + \underbrace{\sqrt{2\beta(t)}\sigma(t)d\mathbf{w}_t}_{\text{noise injection}}}_{\text{Langevin diffusion SDE}} \tag{27}$$

Stochastic Sampling

Role of stochasticity

In theory, ODE and SDE have the same marginal distributions.
However, in practice stochasticity often enhances the sample quality.

Authors try to explain the role of stochasticity as the followings:

1. $\mathbf{x}_t$ deviates from the ideal marginal distribution, because of the training error and truncation error.
2. The Langevin diffusion drives the sample towards the ideal marginal distribution.

Stochastic Sampling

Algorithm

Authors suggest their stochastic sampling algorithms:

1. Add noise to the sample according to a factor $\gamma_i \leq 0$ to reach a higher noise level.
2. Solve the ODE backward from increased noise level to desired noise level.

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

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

  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 $\hat{t}_i$

    $\boldsymbol{d}_i \leftarrow \big(\hat{\boldsymbol{x}}_i - D_\theta(\hat{\boldsymbol{x}}_i;\hat{t}_i)\big)/\hat{t}_i$ ▷ 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**

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

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

  return $\boldsymbol{x}_N$

Stochastic Sampling

Algorithm in real world

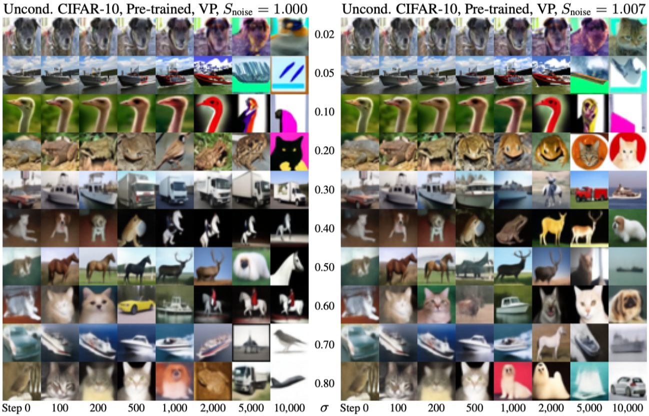


Figure 5: Gradual image degradation with repeated addition and removal of noise. A random image is drawn from $p(\mathbf{x}; \sigma)$ and run Algorithm 2 for a certain number of steps with $\gamma_i = \sqrt{2} - 1$.

Stochastic Sampling

Algorithm in real world

Figure 5 shows the observation of the **effect of Langevin diffusion**. It is supposed to drive the sample towards the true data distribution, however...

1. For low noise levels, images drift toward **oversaturated colors**.
2. For high noise levels, images become **abstract** when $S_{\text{noise}} = 1$.

Authors suspect that **non-conservative vector field** generated by parametrized denoiser **violates the premises of Langevin diffusion** since their analytical denoisers have not shown such degradation.

$\rightarrow$ Fix flaws of $D_\theta(\mathbf{x}; \sigma)$ with heuristic!

Stochastic Sampling

Algorithm in real world

Fix flaws of $D_\theta(\mathbf{x}; \sigma)$ with heuristic!

1. For low noise levels, images drift toward **oversaturated colors**.
   $\rightarrow$ Enable stochasticity within $t_i \in [S_{\text{tmin}}, \underline{S_{\text{tmax}}}]$.
2. For high noise levels, images become **abstract** when $S_{\text{noise}} = 1$.
   $\rightarrow$ $D_\theta(\cdot)$ removes too much noise because of regression towards the mean, which often happens when ℓ_2 trained.
   $\rightarrow$ Inflate the standard deviation of newly added noise: $S_{\text{noise}} > 1$.
3. New noise never exceeds the noise already in the image.
   $\rightarrow$ Clamp γ_i.
4. Controls the overal stochasticity by S_{churn}.

Results

Determinisitic Sampling

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 [28] at 32×32				FFHQ [26] 64×64		AFHQv2 [7] 64×64	
	Conditional		Unconditional		Unconditional		Unconditional	
Training configuration	VP	VE	VP	VE	VP	VE	VP	VE
A Baseline [42] (*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

Stochastic Sampling

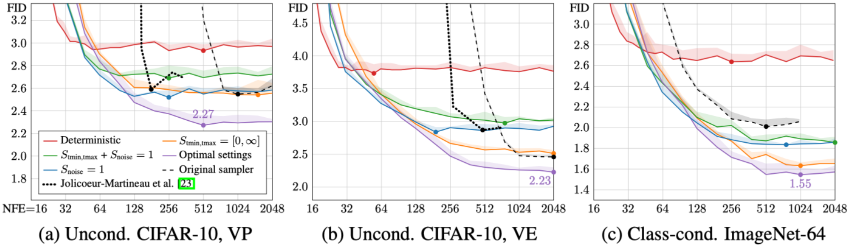


Figure 6: Evaluation of stochastic sampler. Red line is deterministic sampler while purple line is optimal stochastic sampler.

Reference I

Yaniv Benny and Lior Wolf.
Dynamic dual-output diffusion models.
CVPR, 2022.

Prafulla Dhariwal and Alexander Nichol.
Diffusion models beat GANs on image synthesis.
In *NeurIPS*, 2021.

Jonathan Ho, Ajay Jain, and Pieter Abbeel.
Denoising diffusion probabilistic models.
In *NeurIPS*, 2020.

Tero Karras, Miika Aittala, Janne Hellsten, Samuli Laine, Jaakko Lehtinen, and Timo Aila.
Training generative adversarial networks with limited data.
In *NeurIPS*, 2020.

Jiaming Song, Chenlin Meng, and Stefano Ermon.
Denoising diffusion implicit models.
In *ICLR*, 2021.

Yang Song, Jascha Sohl-Dickstein, Diederik P Kingma, Abhishek Kumar, Stefano Ermon, and Ben Poole.
Score-based generative modeling through stochastic differential equations.
In *ICLR*, 2021.