

Designing Diffusion Models in Real World

Junoh Kang

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



Computer Vision Lab
Seoul National University

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

The reasons of reviewing the 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

1. Revisit Diffusion Models
2. Improvements to Training
3. Improvements to Deterministic Sampling
4. Stochastic Sampling

Revisit Diffusion Models

Revisit Diffusion Models

Reformulate Diffusion Models

Song et al. [3] defines forward SDE as

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

As a consequence, the marginal distribution at time t becomes

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

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

Revisit Diffusion Models

Reformulate Diffusion Models

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

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

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

Then, the corresponding probability flow ODE is

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

Revisit Diffusion Models

Obstacles in diffusion models

1. Generation by diffusion models can be interpreted as solving

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

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. → **Better training**.
3. The solution at $t = 0$ given boundary condition at $t = T$ is

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

The integral is numerically calculated, which causes truncation error. → **Reduce truncation errors, focus on important region.**

Revisit Diffusion Models

Design Space of Diffusion Models

- ▶ Components regarding Training
 - ▶ Parametrization
 - ▶ Network preconditioning: $c_{\text{skip}}(\sigma)$, $c_{\text{out}}(\sigma)$, $c_{\text{in}}(\sigma)$, $c_{\text{noise}}(\sigma)$
 - ▶ Loss weighting: $\lambda(t)$
 - ▶ Noise distribution: $\sigma \sim p_{\text{noise}}$
 - ▶ Augmentation
- ▶ Components regarding Deterministic Sampling
 - ▶ Truncation-error-reducing ODE: $s(t)$, $\sigma(t)$
 - ▶ Truncation-error-reducing algorithms: Higher-order integrators
 - ▶ Distributing truncation errors properly: Discretization $\{t_i\}_0^N$
- ▶ Components regarding Stochastic Sampling
 - ▶ Rate of replaced noises $\beta(t)$
 - ▶ Heuristics: S_{tmin} , S_{tmax} , S_{noise} , S_{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{y} :

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

The relation between a score function and the ideal denoiser is

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

Networks in many baselines predicts either $D(\mathbf{x}, \sigma)$ or n .

Training

Parametrization

Benny et al. [1] observes that predicting the denoised output, $D(\mathbf{x}; \sigma)$, is easier for high noise level, while predicting the noise, \mathbf{n} , is easier for low noise level.

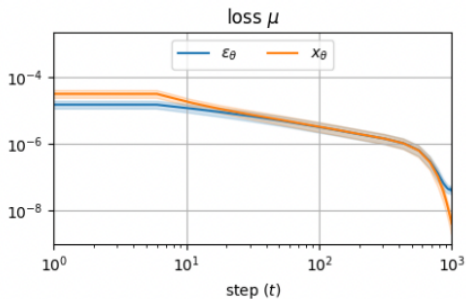


Figure 1: Loss comparison between predicting the denoised output or the added noise.

Training

Network preconditioning

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

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

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

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

Training

Network preconditioning

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

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

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

3. Effective training target should have unit variance.

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

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

Training

Network preconditioning

4. Errors of network should not be amplified.

$$c_{\text{skip}}(\sigma) = \arg \min_{c_{\text{skip}}(\sigma)} c_{\text{out}}(\sigma) \quad (15)$$

$$\Rightarrow \begin{cases} c_{\text{skip}}(\sigma) = \sigma_{\text{data}}^2 / (\sigma^2 + \sigma_{\text{data}}^2) \\ c_{\text{out}}(\sigma) = \sigma \cdot \sigma_{\text{data}} / \sqrt{\sigma^2 + \sigma_{\text{data}}^2} \end{cases} \quad (16)$$

5. Effective weight should be uniform.

$$\lambda(\sigma) c_{\text{out}}(\sigma)^2 = 1 \quad (17)$$

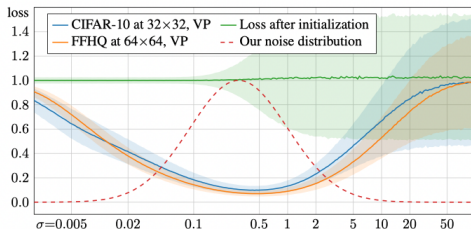
$$\Rightarrow \lambda(\sigma) = (\sigma^2 + \sigma_{\text{data}}^2) / (\sigma \cdot \sigma_{\text{data}})^2. \quad (18)$$

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

Network preconditioning & noise distribution

At low noise levels, separating the small noise components is difficult and irrelevant, whereas at high noise levels, the correct answer approaches to dataset average; EDM focuses on middle range noise levels for training: $\log(\sigma) \sim \mathcal{N}(-1.2, 1.2)$.



(a) Training loss & noise distribution

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

EDM uses augmentation pipeline from the GAN literature [2].

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

Improvements to Deterministic Sampling

Higher-order integrators & Discretization

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

$$d\mathbf{x}_t = [-\dot{\sigma}(t)\sigma(t)\nabla_{\mathbf{x}} \log p(\mathbf{x}_t; \sigma(t))] dt. \quad (19)$$

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

$$d\mathbf{x}_t/dt = (\mathbf{x}_t - D(\mathbf{x}_t; t))/t \quad (20)$$

$$:= f(\mathbf{x}_t, t) \quad (21)$$

Improvements to Deterministic Sampling

Truncation-error-reducing algorithms: Higher-order integrators

Euler method approximates the integral by

$$\int_{t_i}^{t_{i-1}} f(\mathbf{x}_t, t) dt = (t_{i-1} - t_i) f(\mathbf{x}_{t_i}, t_i) + O(|t_{i-1} - t_i|^2). \quad (22)$$

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

$$\int_{t_i}^{t_{i-1}} f(\mathbf{x}_t, t) dt = (t_{i-1} - t_i) (f(\mathbf{x}_{t_i}, t_i) + f(\hat{\mathbf{x}}_{t_{i-1}}, t_{i-1})) / 2 + O(|t_{i-1} - t_i|^3). \quad (23)$$

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

Improvements to Deterministic Sampling

Truncation-error-reducing algorithms: Higher-order integrators

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

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

Improvements to Deterministic 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

$$t_{N-i} = \sigma_{i < N} = \left(\sigma_{\max}^{1/\rho} + \frac{i}{N-1} (\sigma_{\min}^{1/\rho} - \sigma_{\max}^{1/\rho}) \right)^\rho, \sigma_N = 0, \quad (24)$$

increasing ρ results dense discretizations at low noise levels.

Improvements to Deterministic Sampling

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

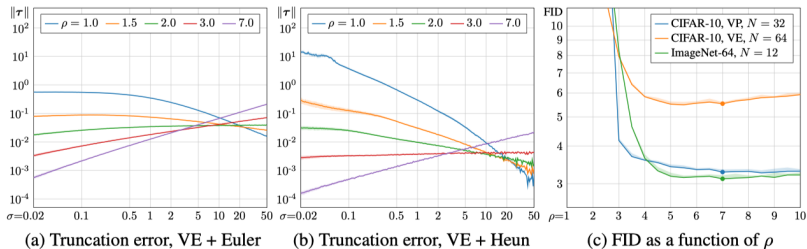


Figure 3: (a),(b) Local truncation error at different noise levels. (c) FID as a function of ρ .

$\rho = 3$ 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 Deterministic 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. $s(t)$ and $\sigma(t)$ determine the shape of the ODE solution trajectories, which is closely related to linearity of the $f(\cdot)$.

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

Improvements to Deterministic Sampling

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

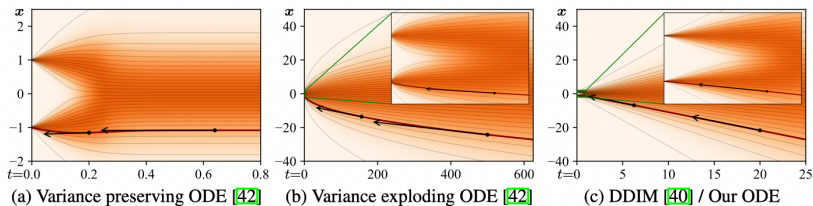


Figure 4: A sketch of ODE curvature in 1D where p_{data} is two Dirac peaks at $\mathbf{x} = \pm 1$. Axis is chosen to show $\sigma \in [0, 25]$ and zoom in $\sigma \in [0, 1]$. (c) sketches the curvature when $s(t) = 1$ and $\sigma(t) = t$. It has small curvature, while the tangent directs to the datapoints.

Improvements to Deterministic Sampling

Results

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 .

Training configuration	CIFAR-10 [28] at 32×32				FFHQ [26] 64×64		AFHQv2 [7] 64×64	
	Conditional		Unconditional		Unconditional		Unconditional	
	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

- ▶ Config B changes basic hyperparameters such as batch size, learning rate, dropout, *etc*; it disable gradient clipping
- ▶ Config C improves the expressive power of the model.
- ▶ Configs D, E, and F are explained in the previous context.

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:

$$d\mathbf{x}_{\pm} = \underbrace{-\dot{\sigma}(t)\sigma(t)\nabla_{\mathbf{x}} \log p(\mathbf{x}; \sigma(t))dt}_{\text{probability flow ODE}} \quad (25)$$

$$\pm \underbrace{\beta(t)\sigma(t)^2\nabla_{\mathbf{x}} \log p(\mathbf{x}; \sigma(t))dt}_{\text{deterministic noise decay}} + \underbrace{\sqrt{2\beta(t)}\sigma(t)d\mathbf{w}_t}_{\text{noise injection}} \quad (26)$$

Langevin diffusion SDE

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

Stochastic sampling algorithm in EDM is executed in two steps:

1. **Noise injection:** integrate noise into samples according to $\gamma_i \geq 0$.
2. **Noise decay with probability flow:** solve the ODE from heightened noise level to desired level.

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

```
1: procedure STOCHASTICSAMPLER( $D_\theta(\mathbf{x}; \sigma)$ ,  $t_i \in \{0, \dots, N\}$ ,  $\gamma_i \in \{0, \dots, N-1\}$ ,  $S_{\text{noise}}$ )
2:   sample  $\mathbf{x}_0 \sim \mathcal{N}(\mathbf{0}, t_0^2 \mathbf{I})$ 
3:   for  $i \in \{0, \dots, N-1\}$  do
4:     sample  $\boldsymbol{\epsilon}_i \sim \mathcal{N}(\mathbf{0}, S_{\text{noise}}^2 \mathbf{I})$ 
5:      $\hat{t}_i \leftarrow t_i + \gamma_i t_i$ 
6:      $\hat{\mathbf{x}}_i \leftarrow \mathbf{x}_i + \sqrt{\hat{t}_i^2 - t_i^2} \boldsymbol{\epsilon}_i$ 
7:      $\mathbf{d}_i \leftarrow (\hat{\mathbf{x}}_i - D_\theta(\hat{\mathbf{x}}_i; \hat{t}_i)) / \hat{t}_i$ 
8:      $\mathbf{x}_{i+1} \leftarrow \hat{\mathbf{x}}_i + (t_{i+1} - \hat{t}_i) \mathbf{d}_i$ 
9:     if  $t_{i+1} \neq 0$  then
10:       $\mathbf{d}'_i \leftarrow (\mathbf{x}_{i+1} - D_\theta(\mathbf{x}_{i+1}; t_{i+1})) / t_{i+1}$ 
11:       $\mathbf{x}_{i+1} \leftarrow \hat{\mathbf{x}}_i + (t_{i+1} - \hat{t}_i) (\frac{1}{2} \mathbf{d}_i + \frac{1}{2} \mathbf{d}'_i)$ 
12:   return  $\mathbf{x}_N$ 
```

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

\triangleright Select temporarily increased noise level \hat{t}_i

\triangleright Add new noise to move from t_i to \hat{t}_i

\triangleright Evaluate $d\mathbf{x}/dt$ at \hat{t}_i

\triangleright Take Euler step from \hat{t}_i to t_{i+1}

\triangleright Apply 2nd order correction

Stochastic Sampling

Algorithm in real world

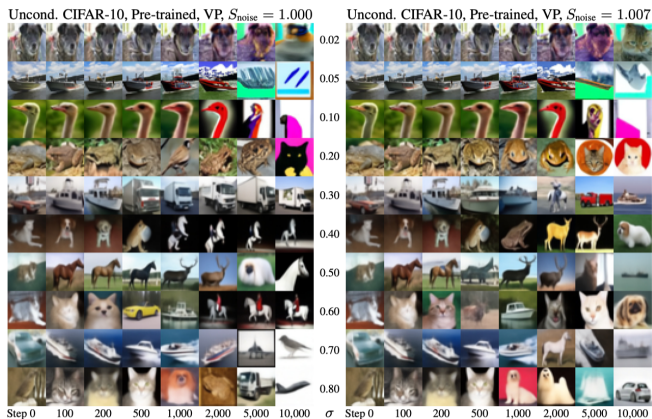


Figure 5: Observe the effect of Langevin diffusion in real world: there is gradual image degradation with the repeated addition and removal of noise. A random image is drawn from $p(x; \sigma)$ and Algorithm 2 is run for a certain number of steps with $\gamma_i = \sqrt{2} - 1$.

Stochastic Sampling

Algorithm in real world

Langevin diffusion 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.

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

Stochastic Sampling

Results

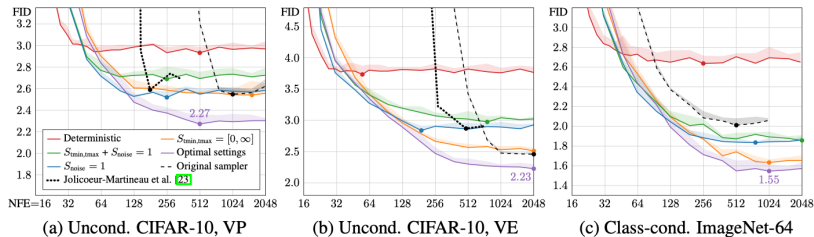


Figure 6: Evaluation of stochastic sampler with ablations. Red line is deterministic sampler while purple line is optimal stochastic sampler.

Reference I



Yaniv Benny and Lior Wolf.

Dynamic dual-output diffusion models.
CVPR, 2022.



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.