

Diffusion Models and Accelerating Their Sampling

Pravin Nair

Department of Electrical Engineering

Indian Institute of Technology Madras

`pravinnair@iitm.ac.in`

17 August 2026

Joint work with Anuska Roy, MS Research Scholar

Why diffusion models?

High-quality synthesis

- State-of-the-art visual fidelity on large-scale image-generation benchmarks.
- Strong sample diversity and improved coverage of the data distribution.
- No adversarial min-max optimization is required during training.

Why diffusion models?

High-quality synthesis

- State-of-the-art visual fidelity on large-scale image-generation benchmarks.
- Strong sample diversity and improved coverage of the data distribution.
- No adversarial min-max optimization is required during training.

A flexible generative framework

- Applicable to images, audio, video and multimodal generation.
- Supports conditional generation, editing and controllable synthesis.
- Provides powerful learned priors for restoration and inverse problems.

Why diffusion models?

High-quality synthesis

- State-of-the-art visual fidelity on large-scale image-generation benchmarks.
- Strong sample diversity and improved coverage of the data distribution.
- No adversarial min-max optimization is required during training.

A flexible generative framework

- Applicable to images, audio, video and multimodal generation.
- Supports conditional generation, editing and controllable synthesis.
- Provides powerful learned priors for restoration and inverse problems.

Diffusion and flow models combine high-quality synthesis, strong distribution coverage and a broadly reusable generative framework.

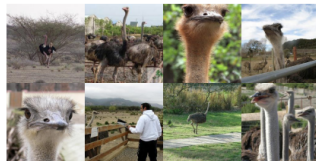
Diffusion models surpass GANs in image synthesis

ImageNet 256 × 256:

BigGAN-deep FID 6.95

versus

Guided diffusion FID 4.59



BigGAN-deep

Guided diffusion

Ground truth

Diffusion models produces samples with improved distribution coverage.

Application of diffusion models in Inverse problems



Measurement
Blur + noise



Ground truth

Application of diffusion models in inverse problems



IRCNN

Runtime: 0.07 s



Restormer

Runtime: 0.26 s



DDRM

Runtime: 2.56 s

Zhang et al., CVPR 2017; Zamir et al., CVPR 2022; Kawar et al., NeurIPS 2022.

Application of diffusion models in Inverse problems



Measurement



Ground Truth



DDRM

Why do we require faster sampling?

Sampling is an iterative computation

- Generation starts from noise and follows a learned reverse-time dynamics.

Why do we require faster sampling?

Sampling is an iterative computation

- Generation starts from noise and follows a learned reverse-time dynamics.
- Every numerical step requires evaluating a large neural network.

Why do we require faster sampling?

Sampling is an iterative computation

- Generation starts from noise and follows a learned reverse-time dynamics.
- Every numerical step requires evaluating a large neural network.
- Consequently, the computational cost is approximately proportional to the number of network function evaluations (NFE):

$$\text{sampling cost} \approx \text{NFE} \times \text{cost of one network evaluation.}$$

Why do we require faster sampling?

Sampling is an iterative computation

- Generation starts from noise and follows a learned reverse-time dynamics.
- Every numerical step requires evaluating a large neural network.
- Consequently, the computational cost is approximately proportional to the number of network function evaluations (NFE):

$$\text{sampling cost} \approx \text{NFE} \times \text{cost of one network evaluation.}$$

After training, image generation becomes a numerical-analysis problem: produce an accurate sample using as few network evaluations as possible.

Quality–runtime trade-off with increasing NFE



6 NFEs
7.38 s



10 NFEs
10.71 s



15 NFEs
16.05 s



20 NFEs
20.48 s

Prompt: *"Two kids in pink and purple jackets standing by a fence."*

Increasing the sampling budget progressively improves image quality, but at a nearly proportional increase in wall-clock time.

Diffusion Models: From Noise to Images

- A diffusion model learns how to gradually remove noise from an image.
- At sampling time, the process is reversed: the model starts from noise and iteratively moves toward a realistic image.

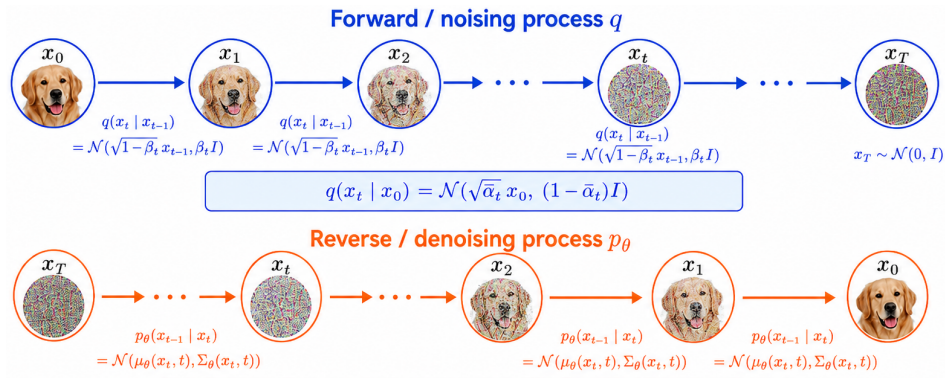


Figure is generated by text-to-image generation diffusion model DALL-E used by GPT 5.5.

Forward process

Let

$$\mathbf{x}_0 \sim p_{\text{data}}, \quad \epsilon \sim \mathcal{N}(\mathbf{0}, \mathbf{I}_d),$$

- $\mathbf{x}_0 \in \mathbb{R}^d$ is a clean data sample.
- p_{data} is the unknown data distribution.
- $\epsilon \in \mathbb{R}^d$ is independent standard Gaussian noise:

$$\mathbb{E}[\epsilon] = \mathbf{0}, \quad \text{Cov}(\epsilon) = \mathbf{I}_d.$$

- $t \in [0, T]$ denotes the corruption time or noise level.

The forward process is designed in advance through the two schedules s_t and σ_t ; it is not learned from data.

The affine Gaussian corruption

Choose two known scalar schedules, $s_t \geq 0$ and $\sigma_t \geq 0$.

s_t : amount of clean signal retained at time t ,

σ_t : standard deviation of the Gaussian noise at time t .

For every time $t \in [0, T]$, define

$$\mathbf{x}_t = s_t \mathbf{x}_0 + \sigma_t \epsilon.$$

The affine Gaussian corruption

Choose two known scalar schedules, $s_t \geq 0$ and $\sigma_t \geq 0$.

s_t : amount of clean signal retained at time t ,

σ_t : standard deviation of the Gaussian noise at time t .

For every time $t \in [0, T]$, define

$$\mathbf{x}_t = s_t \mathbf{x}_0 + \sigma_t \epsilon.$$

Conditioned on the clean sample $\mathbf{x}_0$, only ϵ is random. Therefore,

$$\mathbb{E}[\mathbf{x}_t \mid \mathbf{x}_0] = s_t \mathbf{x}_0$$

and

$$\text{Cov}(\mathbf{x}_t \mid \mathbf{x}_0) = \sigma_t^2 \mathbf{I}_d.$$

The affine Gaussian corruption

Choose two known scalar schedules, $s_t \geq 0$ and $\sigma_t \geq 0$.

s_t : amount of clean signal retained at time t ,

σ_t : standard deviation of the Gaussian noise at time t .

For every time $t \in [0, T]$, define

$$\mathbf{x}_t = s_t \mathbf{x}_0 + \sigma_t \epsilon.$$

Conditioned on the clean sample $\mathbf{x}_0$, only ϵ is random. Therefore,

$$\mathbb{E}[\mathbf{x}_t \mid \mathbf{x}_0] = s_t \mathbf{x}_0$$

and

$$\text{Cov}(\mathbf{x}_t \mid \mathbf{x}_0) = \sigma_t^2 \mathbf{I}_d.$$

Hence,

$$q_{0t}(\mathbf{x}_t \mid \mathbf{x}_0) = \mathcal{N}(s_t \mathbf{x}_0, \sigma_t^2 \mathbf{I}_d).$$

The affine Gaussian corruption

The schedules are normally chosen so that

$t = 0$: $s_0 \approx 1$, $\sigma_0 \approx 0$, $\mathbf{x}_0$ is approximately clean,

$t = T$: $s_T \approx 0$, $\sigma_T \approx 1$, $\mathbf{x}_T$ is approximately Gaussian noise.

- Given a data point, the schedules decide the trajectory of the forward process.

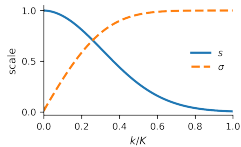
Forward diffusion gradually replaces structured data by Gaussian noise while keeping the corruption distribution known.

Common choices of s_t and σ_t

DDPM (discrete variance-preserving path)

$$s_k = \sqrt{\bar{\alpha}_k}, \quad \sigma_k = \sqrt{1 - \bar{\alpha}_k}, \quad \bar{\alpha}_k = \prod_{j=1}^k (1 - \beta_j).$$

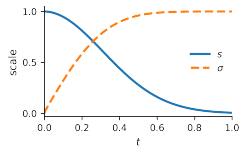
$$\beta_k = \beta_{\min} + \frac{k-1}{K-1} (\beta_{\max} - \beta_{\min}).$$



VP-SDE (continuous variance-preserving path)

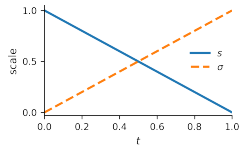
$$s_t = e^{-B(t)/2}, \quad \sigma_t = \sqrt{1 - e^{-B(t)}}, \quad B(t) = \int_0^t \beta(u) du.$$

$$\beta(t) = \beta_{\min} + \frac{t}{T} (\beta_{\max} - \beta_{\min}).$$



Rectified flow (linear interpolation)

$$s_t = 1 - t, \quad \sigma_t = t, \quad t \in [0, 1].$$



From corruption to generation: we need dynamics

So far, the schedules $\mathbf{s}_t$ and σ_t specify

$$\mathbf{x}_t = \mathbf{s}_t \mathbf{x}_0 + \sigma_t \epsilon, \quad q_{0t}(\mathbf{x}_t \mid \mathbf{x}_0) = \mathcal{N}(\mathbf{s}_t \mathbf{x}_0, \sigma_t^2 \mathbf{I}_d).$$

Hence, for any chosen t , we know how to move directly from a clean sample to a noisy sample:

$$\mathbf{x}_0 \xrightarrow{q_{0t}} \mathbf{x}_t.$$

From corruption to generation: we need dynamics

So far, the schedules s_t and σ_t specify

$$\mathbf{x}_t = s_t \mathbf{x}_0 + \sigma_t \epsilon, \quad q_{0t}(\mathbf{x}_t \mid \mathbf{x}_0) = \mathcal{N}(s_t \mathbf{x}_0, \sigma_t^2 \mathbf{I}_d).$$

Hence, for any chosen t , we know how to move directly from a clean sample to a noisy sample:

$$\mathbf{x}_0 \xrightarrow{q_{0t}} \mathbf{x}_t.$$

But generation asks for the opposite direction:

$$\mathbf{x}_T \sim p_T \longrightarrow \mathbf{x}_0 \sim p_{\text{data}}.$$

From corruption to generation: we need dynamics

So far, the schedules $\mathbf{s}_t$ and σ_t specify

$$\mathbf{x}_t = \mathbf{s}_t \mathbf{x}_0 + \sigma_t \epsilon, \quad q_{0t}(\mathbf{x}_t \mid \mathbf{x}_0) = \mathcal{N}(\mathbf{s}_t \mathbf{x}_0, \sigma_t^2 \mathbf{I}_d).$$

Hence, for any chosen t , we know how to move directly from a clean sample to a noisy sample:

$$\mathbf{x}_0 \xrightarrow{q_{0t}} \mathbf{x}_t.$$

But generation asks for the opposite direction:

$$\mathbf{x}_T \sim p_T \longrightarrow \mathbf{x}_0 \sim p_{\text{data}}.$$

Why not learn one direct map from $\mathbf{x}_T$ to $\mathbf{x}_0$?

From corruption to generation: we need dynamics

So far, the schedules $\mathbf{s}_t$ and σ_t specify

$$\mathbf{x}_t = \mathbf{s}_t \mathbf{x}_0 + \sigma_t \epsilon, \quad q_{0t}(\mathbf{x}_t | \mathbf{x}_0) = \mathcal{N}(\mathbf{s}_t \mathbf{x}_0, \sigma_t^2 \mathbf{I}_d).$$

Hence, for any chosen t , we know how to move directly from a clean sample to a noisy sample:

$$\mathbf{x}_0 \xrightarrow{q_{0t}} \mathbf{x}_t.$$

But generation asks for the opposite direction:

$$\mathbf{x}_T \sim p_T \longrightarrow \mathbf{x}_0 \sim p_{\text{data}}.$$

Why not learn one direct map from $\mathbf{x}_T$ to $\mathbf{x}_0$?

- At large noise levels, $\mathbf{x}_T$ contains almost no information about the particular clean sample that should be produced.

From corruption to generation: we need dynamics

So far, the schedules s_t and σ_t specify

$$\mathbf{x}_t = s_t \mathbf{x}_0 + \sigma_t \epsilon, \quad q_{0t}(\mathbf{x}_t | \mathbf{x}_0) = \mathcal{N}(s_t \mathbf{x}_0, \sigma_t^2 \mathbf{I}_d).$$

Hence, for any chosen t , we know how to move directly from a clean sample to a noisy sample:

$$\mathbf{x}_0 \xrightarrow{q_{0t}} \mathbf{x}_t.$$

But generation asks for the opposite direction:

$$\mathbf{x}_T \sim p_T \longrightarrow \mathbf{x}_0 \sim p_{\text{data}}.$$

Why not learn one direct map from $\mathbf{x}_T$ to $\mathbf{x}_0$?

- At large noise levels, $\mathbf{x}_T$ contains almost no information about the particular clean sample that should be produced.
- Image generation therefore proceeds by gradually removing noise, moving through a sequence of intermediate states:

$$\mathbf{x}_T \longrightarrow \mathbf{x}_{T-\tau} \longrightarrow \mathbf{x}_{T-2\tau} \longrightarrow \cdots \longrightarrow \mathbf{x}_0.$$

From corruption to generation: we need dynamics

- Hence, rather than requiring a direct map from $\mathbf{x}_T$ to $\mathbf{x}_0$, we need a mechanism that tells us how to generate the next less-noisy state

$\mathbf{x}_{t-\tau}$ given the current state $\mathbf{x}_t$.

Repeating this transition allows us to move through the corresponding intermediate distributions

$$p_T \longrightarrow p_{T-\tau} \longrightarrow p_{T-2\tau} \longrightarrow \cdots \longrightarrow p_0.$$

From corruption to generation: we need dynamics

- Hence, rather than requiring a direct map from $\mathbf{x}_T$ to $\mathbf{x}_0$, we need a mechanism that tells us how to generate the next less-noisy state

$\mathbf{x}_{t-\tau}$ given the current state $\mathbf{x}_t$.

Repeating this transition allows us to move through the corresponding intermediate distributions

$$p_T \longrightarrow p_{T-\tau} \longrightarrow p_{T-2\tau} \longrightarrow \cdots \longrightarrow p_0.$$

To describe this evolution, knowing only $q_{0t}(\mathbf{x}_t | \mathbf{x}_0)$ is not enough. We also need a consistent rule for moving between intermediate times,

$$q_{rt}(\mathbf{x}_t | \mathbf{x}_r), \quad 0 \leq r < t.$$

From corruption to generation: we need dynamics

- Hence, rather than requiring a direct map from $\mathbf{x}_T$ to $\mathbf{x}_0$, we need a mechanism that tells us how to generate the next less-noisy state

$\mathbf{x}_{t-\tau}$ given the current state $\mathbf{x}_t$.

Repeating this transition allows us to move through the corresponding intermediate distributions

$$p_T \longrightarrow p_{T-\tau} \longrightarrow p_{T-2\tau} \longrightarrow \cdots \longrightarrow p_0.$$

To describe this evolution, knowing only $q_{0t}(\mathbf{x}_t | \mathbf{x}_0)$ is not enough. We also need a consistent rule for moving between intermediate times,

$$q_{rt}(\mathbf{x}_t | \mathbf{x}_r), \quad 0 \leq r < t.$$

We choose these transitions so that the forward process is Markov:

$$q(\mathbf{x}_t | \mathbf{x}_r, \mathbf{x}_{r_1}, \dots, \mathbf{x}_0) = q(\mathbf{x}_t | \mathbf{x}_r).$$

From one-time corruptions to a stochastic process

- For every time t , we have

$$q_{0t}(\mathbf{x}_t \mid \mathbf{x}_0) = \mathcal{N}(\mathbf{s}_t \mathbf{x}_0, \sigma_t^2 \mathbf{I}_d) .$$

From one-time corruptions to a stochastic process

- For every time t , we have

$$q_{0t}(\mathbf{x}_t \mid \mathbf{x}_0) = \mathcal{N}(s_t \mathbf{x}_0, \sigma_t^2 \mathbf{I}_d) .$$

- However, it does not yet specify how two intermediate states $\mathbf{x}_r$ and $\mathbf{x}_t$, with $0 \leq r < t$, are related.

From one-time corruptions to a stochastic process

- For every time t , we have

$$q_{0t}(\mathbf{x}_t \mid \mathbf{x}_0) = \mathcal{N}(\mathbf{s}_t \mathbf{x}_0, \sigma_t^2 \mathbf{I}_d) .$$

- However, it does not yet specify how two intermediate states $\mathbf{x}_r$ and $\mathbf{x}_t$, with $0 \leq r < t$, are related.
- We need a transition distribution $q_{rt}(\mathbf{x}_t \mid \mathbf{x}_r)$ and for a Markov process, this transition must satisfy

$$q(\mathbf{x}_t \mid \mathbf{x}_r, \mathbf{x}_{r_1}, \dots, \mathbf{x}_0) = q_{rt}(\mathbf{x}_t \mid \mathbf{x}_r).$$

From one-time corruptions to a stochastic process

- For every time t , we have

$$q_{0t}(\mathbf{x}_t \mid \mathbf{x}_0) = \mathcal{N}(\mathbf{s}_t \mathbf{x}_0, \sigma_t^2 \mathbf{I}_d).$$

- However, it does not yet specify how two intermediate states $\mathbf{x}_r$ and $\mathbf{x}_t$, with $0 \leq r < t$, are related.
- We need a transition distribution $q_{rt}(\mathbf{x}_t \mid \mathbf{x}_r)$ and for a Markov process, this transition must satisfy

$$q(\mathbf{x}_t \mid \mathbf{x}_r, \mathbf{x}_{r_1}, \dots, \mathbf{x}_0) = q_{rt}(\mathbf{x}_t \mid \mathbf{x}_r).$$

- Equivalently, find a Gaussian transition of the form

$$\mathbf{x}_t = a_{t|r} \mathbf{x}_r + b_{t|r} \xi_{r,t},$$

where $\xi_{r,t} \sim \mathcal{N}(\mathbf{0}, \mathbf{I}_d)$ is independent of $\mathbf{x}_r$ and all states before time r .

Determine the transition by matching the marginals

- Recall from forward process formulation that $\mathbf{x}_r \mid \mathbf{x}_0 \sim \mathcal{N}(s_r \mathbf{x}_0, \sigma_r^2 \mathbf{I}_d)$, $\mathbf{x}_t \mid \mathbf{x}_0 \sim \mathcal{N}(s_t \mathbf{x}_0, \sigma_t^2 \mathbf{I}_d)$

Determine the transition by matching the marginals

- Recall from forward process formulation that $\mathbf{x}_r \mid \mathbf{x}_0 \sim \mathcal{N}(s_r \mathbf{x}_0, \sigma_r^2 \mathbf{I}_d)$, $\mathbf{x}_t \mid \mathbf{x}_0 \sim \mathcal{N}(s_t \mathbf{x}_0, \sigma_t^2 \mathbf{I}_d)$
- From defined transition form, conditioned on $\mathbf{x}_0$,

$$\mathbb{E}[\mathbf{x}_t \mid \mathbf{x}_0] = a_{t|r} s_r \mathbf{x}_0.$$

Determine the transition by matching the marginals

- Recall from forward process formulation that $\mathbf{x}_r \mid \mathbf{x}_0 \sim \mathcal{N}(s_r \mathbf{x}_0, \sigma_r^2 \mathbf{I}_d)$, $\mathbf{x}_t \mid \mathbf{x}_0 \sim \mathcal{N}(s_t \mathbf{x}_0, \sigma_t^2 \mathbf{I}_d)$
- From defined transition form, conditioned on $\mathbf{x}_0$,

$$\mathbb{E}[\mathbf{x}_t \mid \mathbf{x}_0] = a_{t|r} s_r \mathbf{x}_0.$$

- To recover the required mean $s_t \mathbf{x}_0$, we must choose $a_{t|r} = \frac{s_t}{s_r}$.

Determine the transition by matching the marginals

- Recall from forward process formulation that $\mathbf{x}_r \mid \mathbf{x}_0 \sim \mathcal{N}(s_r \mathbf{x}_0, \sigma_r^2 \mathbf{I}_d)$, $\mathbf{x}_t \mid \mathbf{x}_0 \sim \mathcal{N}(s_t \mathbf{x}_0, \sigma_t^2 \mathbf{I}_d)$
- From defined transition form, conditioned on $\mathbf{x}_0$,

$$\mathbb{E}[\mathbf{x}_t \mid \mathbf{x}_0] = a_{t|r} s_r \mathbf{x}_0.$$

- To recover the required mean $s_t \mathbf{x}_0$, we must choose $a_{t|r} = \frac{s_t}{s_r}$.
- The conditional covariance is

$$\text{Cov}(\mathbf{x}_t \mid \mathbf{x}_0) = \left(a_{t|r}^2 \sigma_r^2 + b_{t|r}^2 \right) \mathbf{I}_d.$$

Matching it with $\sigma_t^2 \mathbf{I}_d$ gives

$$b_{t|r}^2 = \sigma_t^2 - \frac{s_t^2}{s_r^2} \sigma_r^2.$$

Determine the transition by matching the marginals

- Recall from forward process formulation that $\mathbf{x}_r \mid \mathbf{x}_0 \sim \mathcal{N}(\mathbf{s}_r \mathbf{x}_0, \sigma_r^2 \mathbf{I}_d)$, $\mathbf{x}_t \mid \mathbf{x}_0 \sim \mathcal{N}(\mathbf{s}_t \mathbf{x}_0, \sigma_t^2 \mathbf{I}_d)$
- From defined transition form, conditioned on $\mathbf{x}_0$,

$$\mathbb{E}[\mathbf{x}_t \mid \mathbf{x}_0] = a_{t|r} \mathbf{s}_r \mathbf{x}_0.$$

- To recover the required mean $\mathbf{s}_t \mathbf{x}_0$, we must choose $a_{t|r} = \frac{s_t}{s_r}$.
- The conditional covariance is

$$\text{Cov}(\mathbf{x}_t \mid \mathbf{x}_0) = \left(a_{t|r}^2 \sigma_r^2 + b_{t|r}^2 \right) \mathbf{I}_d.$$

Matching it with $\sigma_t^2 \mathbf{I}_d$ gives

$$b_{t|r}^2 = \sigma_t^2 - \frac{s_t^2}{s_r^2} \sigma_r^2.$$

Therefore,

$$\mathbf{x}_t = \frac{s_t}{s_r} \mathbf{x}_r + \sqrt{\sigma_t^2 - \frac{s_t^2}{s_r^2} \sigma_r^2} \boldsymbol{\xi}_{r,t}.$$

When is the Gaussian transition well-defined?

- The transition covariance must be nonnegative:

$$v_{t|r} = \sigma_t^2 - \frac{s_t^2}{s_r^2} \sigma_r^2 \geq 0.$$

When is the Gaussian transition well-defined?

- The transition covariance must be nonnegative:

$$v_{t|r} = \sigma_t^2 - \frac{s_t^2}{s_r^2} \sigma_r^2 \geq 0.$$

- For $s_r, s_t > 0$, this is equivalent to

$$\frac{\sigma_t^2}{s_t^2} \geq \frac{\sigma_r^2}{s_r^2}.$$

When is the Gaussian transition well-defined?

- The transition covariance must be nonnegative:

$$v_{t|r} = \sigma_t^2 - \frac{s_t^2}{s_r^2} \sigma_r^2 \geq 0.$$

- For $s_r, s_t > 0$, this is equivalent to

$$\frac{\sigma_t^2}{s_t^2} \geq \frac{\sigma_r^2}{s_r^2}.$$

- Now define the noise-to-signal ratio $\rho_t := \frac{\sigma_t}{s_t}$. Then

$$v_{t|r} \geq 0 \iff \rho_t \geq \rho_r \quad \text{for } r < t.$$

When is the Gaussian transition well-defined?

- The transition covariance must be nonnegative:

$$v_{t|r} = \sigma_t^2 - \frac{s_t^2}{s_r^2} \sigma_r^2 \geq 0.$$

- For $s_r, s_t > 0$, this is equivalent to

$$\frac{\sigma_t^2}{s_t^2} \geq \frac{\sigma_r^2}{s_r^2}.$$

- Now define the noise-to-signal ratio $\rho_t := \frac{\sigma_t}{s_t}$. Then

$$v_{t|r} \geq 0 \iff \rho_t \geq \rho_r \quad \text{for } r < t.$$

- Equivalently, the signal-to-noise ratio $\text{SNR}(t) := \frac{s_t^2}{\sigma_t^2} = \frac{1}{\rho_t^2}$ must be nonincreasing.

Connecting the Gaussian Markov process to an SDE

From the Gaussian Markov construction, for $\tau > 0$,

$$\mathbf{x}_{t+\tau} = \frac{s_{t+\tau}}{s_t} \mathbf{x}_t + \sqrt{\sigma_{t+\tau}^2 - \frac{s_{t+\tau}^2}{s_t^2} \sigma_t^2} \boldsymbol{\xi}_t, \quad \boldsymbol{\xi}_t \sim \mathcal{N}(\mathbf{0}, \mathbf{I}_d).$$

Connecting the Gaussian Markov process to an SDE

From the Gaussian Markov construction, for $\tau > 0$,

$$\mathbf{x}_{t+\tau} = \frac{s_{t+\tau}}{s_t} \mathbf{x}_t + \sqrt{\sigma_{t+\tau}^2 - \frac{s_{t+\tau}^2}{s_t^2} \sigma_t^2} \boldsymbol{\xi}_t, \quad \boldsymbol{\xi}_t \sim \mathcal{N}(\mathbf{0}, \mathbf{I}_d).$$

A linear SDE describes the infinitesimal limit of this transition as $\tau \rightarrow 0$:

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

where $\mathbf{W}_t$ is standard Brownian motion.

Connecting the Gaussian Markov process to an SDE

From the Gaussian Markov construction, for $\tau > 0$,

$$\mathbf{x}_{t+\tau} = \frac{s_{t+\tau}}{s_t} \mathbf{x}_t + \sqrt{\sigma_{t+\tau}^2 - \frac{s_{t+\tau}^2}{s_t^2} \sigma_t^2} \boldsymbol{\xi}_t, \quad \boldsymbol{\xi}_t \sim \mathcal{N}(\mathbf{0}, \mathbf{I}_d).$$

A linear SDE describes the infinitesimal limit of this transition as $\tau \rightarrow 0$:

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

where $\mathbf{W}_t$ is standard Brownian motion.

What does one SDE step do?

For a small forward step $\tau > 0$,
Brownian increments satisfy

$$\Delta \mathbf{W}_t := \mathbf{W}_{t+\tau} - \mathbf{W}_t \sim \mathcal{N}(\mathbf{0}, \tau \mathbf{I}_d).$$

Hence we may write

$$\Delta \mathbf{W}_t = \sqrt{\tau} \boldsymbol{\xi}, \quad \boldsymbol{\xi} \sim \mathcal{N}(\mathbf{0}, \mathbf{I}_d).$$

Euler–Maruyama therefore gives

$$\begin{aligned} \mathbf{x}_{t+\tau} &\approx \mathbf{x}_t + f(t)\mathbf{x}_t \tau + g(t)\Delta \mathbf{W}_t \\ &= \mathbf{x}_t + f(t)\mathbf{x}_t \tau + g(t)\sqrt{\tau} \boldsymbol{\xi}. \end{aligned}$$

Connecting the Gaussian Markov process to an SDE

From the Gaussian Markov construction, for $\tau > 0$,

$$\mathbf{x}_{t+\tau} = \frac{s_{t+\tau}}{s_t} \mathbf{x}_t + \sqrt{\sigma_{t+\tau}^2 - \frac{s_{t+\tau}^2}{s_t^2} \sigma_t^2} \boldsymbol{\xi}_t, \quad \boldsymbol{\xi}_t \sim \mathcal{N}(\mathbf{0}, \mathbf{I}_d).$$

A linear SDE describes the infinitesimal limit of this transition as $\tau \rightarrow 0$:

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

where $\mathbf{W}_t$ is standard Brownian motion.

Thus we need to choose $f(t)$ and $g(t)$ to match

$$\mathbb{E}[\mathbf{x}_{t+\tau} \mid \mathbf{x}_t], \quad \text{Cov}(\mathbf{x}_{t+\tau} \mid \mathbf{x}_t)$$

What does one SDE step do?

For a small forward step $\tau > 0$,
Brownian increments satisfy

$$\Delta \mathbf{W}_t := \mathbf{W}_{t+\tau} - \mathbf{W}_t \sim \mathcal{N}(\mathbf{0}, \tau \mathbf{I}_d).$$

Hence we may write

$$\Delta \mathbf{W}_t = \sqrt{\tau} \boldsymbol{\xi}, \quad \boldsymbol{\xi} \sim \mathcal{N}(\mathbf{0}, \mathbf{I}_d).$$

Euler–Maruyama therefore gives

$$\begin{aligned} \mathbf{x}_{t+\tau} &\approx \mathbf{x}_t + f(t)\mathbf{x}_t \tau + g(t)\Delta \mathbf{W}_t \\ &= \mathbf{x}_t + f(t)\mathbf{x}_t \tau + g(t)\sqrt{\tau} \boldsymbol{\xi}. \end{aligned}$$

Connecting the Gaussian Markov process to an SDE

From the Gaussian Markov construction, for $\tau > 0$,

$$\mathbf{x}_{t+\tau} = \frac{s_{t+\tau}}{s_t} \mathbf{x}_t + \sqrt{\sigma_{t+\tau}^2 - \frac{s_{t+\tau}^2}{s_t^2} \sigma_t^2} \boldsymbol{\xi}_t, \quad \boldsymbol{\xi}_t \sim \mathcal{N}(\mathbf{0}, \mathbf{I}_d).$$

A linear SDE describes the infinitesimal limit of this transition as $\tau \rightarrow 0$:

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

where $\mathbf{W}_t$ is standard Brownian motion.

Thus we need to choose $f(t)$ and $g(t)$ to match

$$\mathbb{E}[\mathbf{x}_{t+\tau} \mid \mathbf{x}_t], \quad \text{Cov}(\mathbf{x}_{t+\tau} \mid \mathbf{x}_t)$$

$$f(t) = \frac{\dot{s}_t}{s_t}, \quad g(t)^2 = \frac{d}{dt} \sigma_t^2 - 2 \frac{\dot{s}_t}{s_t} \sigma_t^2 = 2\sigma_t \left(\dot{\sigma}_t - \frac{\dot{s}_t}{s_t} \sigma_t \right) = 2s_t^2 \rho_t \dot{\rho}_t.$$

What does one SDE step do?

For a small forward step $\tau > 0$,
Brownian increments satisfy

$$\Delta \mathbf{W}_t := \mathbf{W}_{t+\tau} - \mathbf{W}_t \sim \mathcal{N}(\mathbf{0}, \tau \mathbf{I}_d).$$

Hence we may write

$$\Delta \mathbf{W}_t = \sqrt{\tau} \boldsymbol{\xi}, \quad \boldsymbol{\xi} \sim \mathcal{N}(\mathbf{0}, \mathbf{I}_d).$$

Euler–Maruyama therefore gives

$$\begin{aligned} \mathbf{x}_{t+\tau} &\approx \mathbf{x}_t + f(t)\mathbf{x}_t \tau + g(t)\Delta \mathbf{W}_t \\ &= \mathbf{x}_t + f(t)\mathbf{x}_t \tau + g(t)\sqrt{\tau} \boldsymbol{\xi}. \end{aligned}$$

Forward corruption is known; generation requires reverse dynamics

- The forward process is now completely specified. From the affine Gaussian path, we constructed transitions of the form $q_{t,t+\tau}(\mathbf{x}_{t+\tau} \mid \mathbf{x}_t)$, or equivalently the forward SDE

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

Thus, given $\mathbf{x}_t$, we know how to move to a more noisy state $\mathbf{x}_{t+\tau}$.

Forward corruption is known; generation requires reverse dynamics

- The forward process is now completely specified. From the affine Gaussian path, we constructed transitions of the form $q_{t,t+\tau}(\mathbf{x}_{t+\tau} | \mathbf{x}_t)$, or equivalently the forward SDE

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

Thus, given $\mathbf{x}_t$, we know how to move to a more noisy state $\mathbf{x}_{t+\tau}$.

- Generation requires exactly the opposite direction. We start from $\mathbf{x}_T \sim p_T$ and need to repeatedly generate a less noisy state:

$$\mathbf{x}_t \longrightarrow \mathbf{x}_{t-\tau} \longrightarrow \mathbf{x}_{t-2\tau} \longrightarrow \cdots \longrightarrow \mathbf{x}_0.$$

Hence we need either the reverse conditional distribution $p(\mathbf{x}_{t-\tau} | \mathbf{x}_t)$, or an equivalent continuous-time dynamics that moves from T to 0.

Forward corruption is known; generation requires reverse dynamics

- The forward process is now completely specified. From the affine Gaussian path, we constructed transitions of the form $q_{t,t+\tau}(\mathbf{x}_{t+\tau} | \mathbf{x}_t)$, or equivalently the forward SDE

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

Thus, given $\mathbf{x}_t$, we know how to move to a more noisy state $\mathbf{x}_{t+\tau}$.

- Generation requires exactly the opposite direction. We start from $\mathbf{x}_T \sim p_T$ and need to repeatedly generate a less noisy state:

$$\mathbf{x}_t \longrightarrow \mathbf{x}_{t-\tau} \longrightarrow \mathbf{x}_{t-2\tau} \longrightarrow \cdots \longrightarrow \mathbf{x}_0.$$

Hence we need either the reverse conditional distribution $p(\mathbf{x}_{t-\tau} | \mathbf{x}_t)$, or an equivalent continuous-time dynamics that moves from T to 0.

- By Bayes' rule,

$$p(\mathbf{x}_{t-\tau} | \mathbf{x}_t) = \frac{q(\mathbf{x}_t | \mathbf{x}_{t-\tau}) p_{t-\tau}(\mathbf{x}_{t-\tau})}{p_t(\mathbf{x}_t)}.$$

Although the forward transition $q(\mathbf{x}_t | \mathbf{x}_{t-\tau})$ is known, the marginal densities $p_{t-\tau}$ and p_t are generally unknown.

How to reverse a SDE? First we will reverse an ODE

- Consider a deterministic dynamical system

$$\frac{d\mathbf{x}_t}{dt} = \mathbf{v}(\mathbf{x}_t, t), \quad \mathbf{x}_0 \sim p_0.$$

How to reverse a SDE? First we will reverse an ODE

- Consider a deterministic dynamical system

$$\frac{d\mathbf{x}_t}{dt} = \mathbf{v}(\mathbf{x}_t, t), \quad \mathbf{x}_0 \sim p_0.$$

- Starting from $\mathbf{x}_0$, forward integration gives

$$\mathbf{x}_0 \longrightarrow \mathbf{x}_{t_1} \longrightarrow \cdots \longrightarrow \mathbf{x}_T \quad \mathbf{x}_{t_i} = \mathbf{x}_{t_{i-1}} + \int_{t_{i-1}}^{t_i} \mathbf{v}(\mathbf{x}_t, t) dt.$$

How to reverse a SDE? First we will reverse an ODE

- Consider a deterministic dynamical system

$$\frac{d\mathbf{x}_t}{dt} = \mathbf{v}(\mathbf{x}_t, t), \quad \mathbf{x}_0 \sim p_0.$$

- Starting from $\mathbf{x}_0$, forward integration gives

$$\mathbf{x}_0 \longrightarrow \mathbf{x}_{t_1} \longrightarrow \cdots \longrightarrow \mathbf{x}_T \quad \mathbf{x}_{t_i} = \mathbf{x}_{t_{i-1}} + \int_{t_{i-1}}^{t_i} \mathbf{v}(\mathbf{x}_t, t) dt.$$

- The trajectory can be recovered by starting from $\mathbf{x}_T$ and integrating the same ODE backwards:

$$\frac{d\mathbf{x}_t}{dt} = \mathbf{v}(\mathbf{x}_t, t), \quad t : T \longrightarrow 0.$$

How to reverse a SDE? First we will reverse an ODE

- Consider a deterministic dynamical system

$$\frac{d\mathbf{x}_t}{dt} = \mathbf{v}(\mathbf{x}_t, t), \quad \mathbf{x}_0 \sim p_0.$$

- Starting from $\mathbf{x}_0$, forward integration gives

$$\mathbf{x}_0 \longrightarrow \mathbf{x}_{t_1} \longrightarrow \cdots \longrightarrow \mathbf{x}_T \quad \mathbf{x}_{t_i} = \mathbf{x}_{t_{i-1}} + \int_{t_{i-1}}^{t_i} \mathbf{v}(\mathbf{x}_t, t) dt.$$

- The trajectory can be recovered by starting from $\mathbf{x}_T$ and integrating the same ODE backwards:

$$\frac{d\mathbf{x}_t}{dt} = \mathbf{v}(\mathbf{x}_t, t), \quad t : T \longrightarrow 0.$$

- Introduce increasing reverse time $\tau := T - t$, $\mathbf{y}_\tau := \mathbf{x}_{T-\tau}$. Then $\frac{d\mathbf{y}_\tau}{d\tau} = -\mathbf{v}(\mathbf{y}_\tau, T - \tau)$.

How to reverse a SDE? First we will reverse an ODE

- Consider a deterministic dynamical system

$$\frac{d\mathbf{x}_t}{dt} = \mathbf{v}(\mathbf{x}_t, t), \quad \mathbf{x}_0 \sim p_0.$$

- Starting from $\mathbf{x}_0$, forward integration gives

$$\mathbf{x}_0 \longrightarrow \mathbf{x}_{t_1} \longrightarrow \cdots \longrightarrow \mathbf{x}_T \quad \mathbf{x}_{t_i} = \mathbf{x}_{t_{i-1}} + \int_{t_{i-1}}^{t_i} \mathbf{v}(\mathbf{x}_t, t) dt.$$

- The trajectory can be recovered by starting from $\mathbf{x}_T$ and integrating the same ODE backwards:

$$\frac{d\mathbf{x}_t}{dt} = \mathbf{v}(\mathbf{x}_t, t), \quad t : T \longrightarrow 0.$$

- Introduce increasing reverse time $\tau := T - t$, $\mathbf{y}_\tau := \mathbf{x}_{T-\tau}$. Then $\frac{d\mathbf{y}_\tau}{d\tau} = -\mathbf{v}(\mathbf{y}_\tau, T - \tau)$.
- Starting from $\mathbf{x}_T$, forward integration gives

$$\mathbf{x}_T \longrightarrow \mathbf{x}_{t_{N-1}} \longrightarrow \cdots \longrightarrow \mathbf{x}_0 \quad \mathbf{x}_{t_{i-1}} = \mathbf{x}_{t_i} - \int_{t_{i-1}}^{t_i} \mathbf{v}(\mathbf{x}_t, t) dt.$$

Reverse dynamics for SDE require the score

- The forward SDE is

$$d\mathbf{x}_t = f(t)\mathbf{x}_t dt + g(t) d\mathbf{W}_t, \quad t : 0 \rightarrow T.$$

Reverse dynamics for SDE require the score

- The forward SDE is

$$d\mathbf{x}_t = f(t)\mathbf{x}_t dt + g(t) d\mathbf{W}_t, \quad t : 0 \rightarrow T.$$

- Let $p_t(\mathbf{x})$ denote the marginal density of $\mathbf{x}_t$. Its *score function* is

$$\mathbf{s}_\star(\mathbf{x}, t) := \nabla_{\mathbf{x}} \log p_t(\mathbf{x}).$$

It points in the direction in which the log-density p_t increases.

Reverse dynamics for SDE require the score

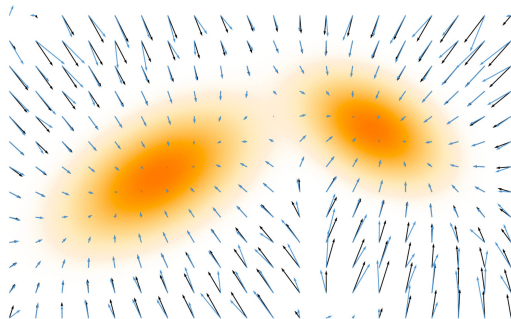
- The forward SDE is

$$d\mathbf{x}_t = f(t)\mathbf{x}_t dt + g(t) d\mathbf{W}_t, \quad t : 0 \rightarrow T.$$

- Let $p_t(\mathbf{x})$ denote the marginal density of $\mathbf{x}_t$. Its *score function* is

$$\mathbf{s}_\star(\mathbf{x}, t) := \nabla_{\mathbf{x}} \log p_t(\mathbf{x}).$$

It points in the direction in which the log-density p_t increases.



Reverse dynamics for SDE require the score

- The forward SDE is

$$d\mathbf{x}_t = f(t)\mathbf{x}_t dt + g(t) d\mathbf{W}_t, \quad t : 0 \rightarrow T.$$

- Let $p_t(\mathbf{x})$ denote the marginal density of $\mathbf{x}_t$. Its *score function* is

$$\mathbf{s}_\star(\mathbf{x}, t) := \nabla_{\mathbf{x}} \log p_t(\mathbf{x}).$$

It points in the direction in which the log-density p_t increases.

- A family of reverse-time dynamics, integrated from T to 0, is

$$d\mathbf{x}_t = \left[f(t)\mathbf{x}_t - \frac{1 + \eta(t)^2}{2} g(t)^2 \mathbf{s}_\star(\mathbf{x}_t, t) \right] dt + \eta(t)g(t) d\tilde{\mathbf{W}}_t.$$

Reverse dynamics for SDE require the score

- The forward SDE is

$$d\mathbf{x}_t = f(t)\mathbf{x}_t dt + g(t) d\mathbf{W}_t, \quad t : 0 \rightarrow T.$$

- Let $p_t(\mathbf{x})$ denote the marginal density of $\mathbf{x}_t$. Its *score function* is

$$\mathbf{s}_\star(\mathbf{x}, t) := \nabla_{\mathbf{x}} \log p_t(\mathbf{x}).$$

It points in the direction in which the log-density p_t increases.

- A family of reverse-time dynamics, integrated from T to 0, is

$$d\mathbf{x}_t = \left[f(t)\mathbf{x}_t - \frac{1 + \eta(t)^2}{2} g(t)^2 \mathbf{s}_\star(\mathbf{x}_t, t) \right] dt + \eta(t)g(t) d\tilde{\mathbf{W}}_t.$$

- Unlike an ODE, an SDE cannot in general be reversed simply by integrating the forward drift backward in time. The reverse drift acquires the score-dependent correction

$$-g(t)^2 \nabla_{\mathbf{x}} \log p_t(\mathbf{x}) \quad (\eta = 1).$$

Reverse dynamics for SDE require the score

With the *exact* score, all of the dynamics varying with η have the same marginal distributions p_t .

$\eta(t)$	Reverse dynamics	Interpretation
1	$d\mathbf{x}_t = [f(t)\mathbf{x}_t - g(t)^2\mathbf{s}_\star(\mathbf{x}_t, t)]dt + g(t)d\tilde{\mathbf{W}}_t$	reverse-time SDE
0	$\frac{d\mathbf{x}_t}{dt} = f(t)\mathbf{x}_t - \frac{1}{2}g(t)^2\mathbf{s}_\star(\mathbf{x}_t, t)$	probability-flow ODE
$0 < \eta(t) < 1$	intermediate score correction and diffusion	partially stochastic

Reverse dynamics for SDE require the score

With the *exact* score, all of the dynamics varying with η have the same marginal distributions p_t .

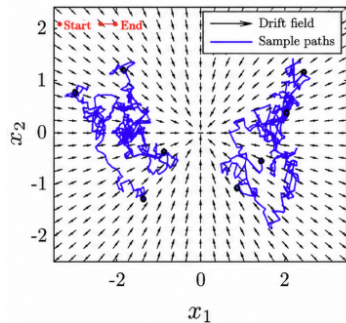
$\eta(t)$	Reverse dynamics	Interpretation
1	$d\mathbf{x}_t = [f(t)\mathbf{x}_t - g(t)^2\mathbf{s}_\star(\mathbf{x}_t, t)]dt + g(t)d\widetilde{\mathbf{W}}_t$	reverse-time SDE
0	$\frac{d\mathbf{x}_t}{dt} = f(t)\mathbf{x}_t - \frac{1}{2}g(t)^2\mathbf{s}_\star(\mathbf{x}_t, t)$	probability-flow ODE
$0 < \eta(t) < 1$	intermediate score correction and diffusion	partially stochastic

Starting from $\mathbf{x}_{t_0} = \mathbf{x}_T \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$, choose $t_i = T - i\tau$ with $\tau = T/100$ and apply Euler–Maruyama

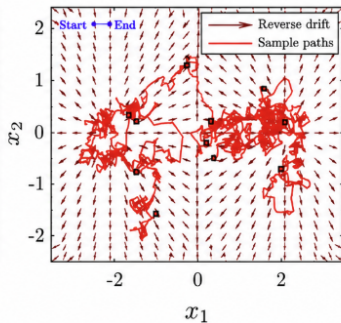
$$\mathbf{x}_{t_{i+1}} = \mathbf{x}_{t_i} - \tau [f(t_i)\mathbf{x}_{t_i} - g(t_i)^2\mathbf{s}_\star(\mathbf{x}_{t_i}, t_i)] + g(t_i)\sqrt{\tau}\xi_i,$$

where, $\xi_i \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$. Then looping over $0, \dots, 99$, yields $\mathbf{x}_{t_{100}} = \mathbf{x}_0$.

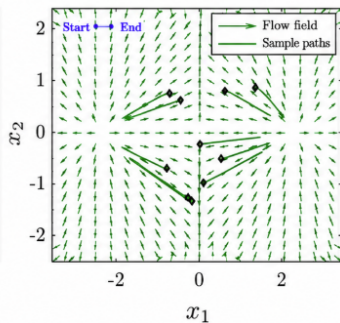
Forward SDE, reverse SDE, and probability-flow ODE



Forward SDE

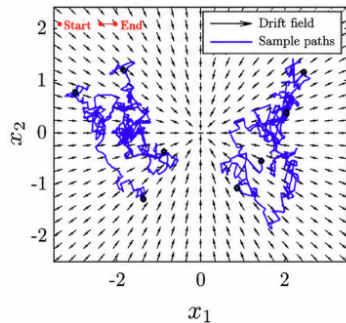


Reverse-time SDE

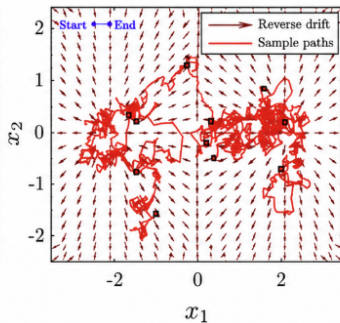


PF-ODE

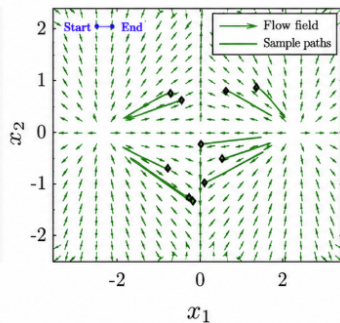
Forward SDE, reverse SDE, and probability-flow ODE



Forward SDE



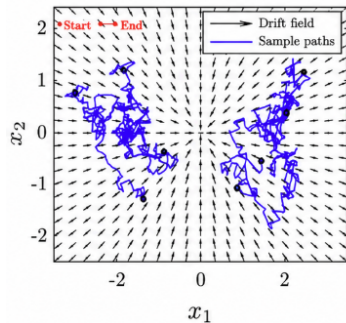
Reverse-time SDE



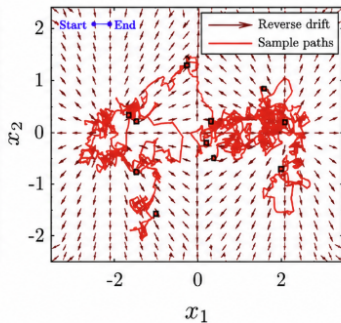
PF-ODE

- “Same marginals” does *not* mean the same trajectories: the reverse-time SDE is stochastic, whereas the probability-flow ODE is deterministic for a fixed terminal sample.

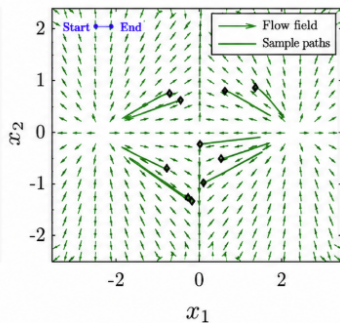
Forward SDE, reverse SDE, and probability-flow ODE



Forward SDE



Reverse-time SDE



PF-ODE

- “Same marginals” does *not* mean the same trajectories: the reverse-time SDE is stochastic, whereas the probability-flow ODE is deterministic for a fixed terminal sample.
- With the exact score, the reverse-time SDE and probability-flow ODE produce the same one-time marginals p_t , despite following different sample paths.

The obstacle: the marginal score is not tractable

- The reverse-time dynamics require the marginal score

$$\mathbf{s}_\star(\mathbf{x}_t, t) := \nabla_{\mathbf{x}_t} \log p_t(\mathbf{x}_t).$$

For example, the probability-flow ODE is

$$\frac{d\mathbf{x}_t}{dt} = f(t)\mathbf{x}_t - \frac{1}{2}g(t)^2 \mathbf{s}_\star(\mathbf{x}_t, t).$$

The obstacle: the marginal score is not tractable

- The reverse-time dynamics require the marginal score

$$\mathbf{s}_\star(\mathbf{x}_t, t) := \nabla_{\mathbf{x}_t} \log p_t(\mathbf{x}_t).$$

For example, the probability-flow ODE is

$$\frac{d\mathbf{x}_t}{dt} = f(t)\mathbf{x}_t - \frac{1}{2}g(t)^2 \mathbf{s}_\star(\mathbf{x}_t, t).$$

- However, the noisy marginal is

$$p_t(\mathbf{x}_t) = \int q_{0t}(\mathbf{x}_t \mid \mathbf{x}_0) p_0(\mathbf{x}_0) d\mathbf{x}_0.$$

Although we have samples from the data distribution p_0 , its density $p_0(\mathbf{x}_0)$ is not available analytically.

The obstacle: the marginal score is not tractable

- The reverse-time dynamics require the marginal score

$$\mathbf{s}_\star(\mathbf{x}_t, t) := \nabla_{\mathbf{x}_t} \log p_t(\mathbf{x}_t).$$

For example, the probability-flow ODE is

$$\frac{d\mathbf{x}_t}{dt} = f(t)\mathbf{x}_t - \frac{1}{2}g(t)^2 \mathbf{s}_\star(\mathbf{x}_t, t).$$

- However, the noisy marginal is

$$p_t(\mathbf{x}_t) = \int q_{0t}(\mathbf{x}_t \mid \mathbf{x}_0) p_0(\mathbf{x}_0) d\mathbf{x}_0.$$

Although we have samples from the data distribution p_0 , its density $p_0(\mathbf{x}_0)$ is not available analytically.

- Hence neither

$$p_t(\mathbf{x}_t) \quad \text{nor} \quad \nabla_{\mathbf{x}_t} \log p_t(\mathbf{x}_t)$$

can be evaluated directly.

Conditioning on a clean sample gives a tractable score

- During training, clean samples

$$\mathbf{x}_0 \sim p_0$$

are available from the dataset, and the forward corruption distribution is completely known:

$$q_{0t}(\mathbf{x}_t \mid \mathbf{x}_0) = \mathcal{N}(\mathbf{s}_t \mathbf{x}_0, \sigma_t^2 \mathbf{I}_d) .$$

Conditioning on a clean sample gives a tractable score

- During training, clean samples

$$\mathbf{x}_0 \sim p_0$$

are available from the dataset, and the forward corruption distribution is completely known:

$$q_{0t}(\mathbf{x}_t \mid \mathbf{x}_0) = \mathcal{N}(\mathbf{s}_t \mathbf{x}_0, \sigma_t^2 \mathbf{I}_d) .$$

- The marginal score can be written as an expectation of these conditional scores. Indeed,

$$\nabla_{\mathbf{x}_t} \log p_t(\mathbf{x}_t) = \frac{1}{p_t(\mathbf{x}_t)} \nabla_{\mathbf{x}_t} p_t(\mathbf{x}_t) = \frac{1}{p_t(\mathbf{x}_t)} \int \nabla_{\mathbf{x}_t} q_{0t}(\mathbf{x}_t \mid \mathbf{x}_0) p_0(\mathbf{x}_0) d\mathbf{x}_0$$

Conditioning on a clean sample gives a tractable score

- During training, clean samples

$$\mathbf{x}_0 \sim p_0$$

are available from the dataset, and the forward corruption distribution is completely known:

$$q_{0t}(\mathbf{x}_t \mid \mathbf{x}_0) = \mathcal{N}(\mathbf{s}_t \mathbf{x}_0, \sigma_t^2 \mathbf{I}_d) .$$

- The marginal score can be written as an expectation of these conditional scores. Indeed,

$$\begin{aligned} \nabla_{\mathbf{x}_t} \log p_t(\mathbf{x}_t) &= \frac{1}{p_t(\mathbf{x}_t)} \nabla_{\mathbf{x}_t} p_t(\mathbf{x}_t) = \frac{1}{p_t(\mathbf{x}_t)} \int \nabla_{\mathbf{x}_t} q_{0t}(\mathbf{x}_t \mid \mathbf{x}_0) p_0(\mathbf{x}_0) d\mathbf{x}_0 \\ &= \frac{1}{p_t(\mathbf{x}_t)} \int q_{0t}(\mathbf{x}_t \mid \mathbf{x}_0) \nabla_{\mathbf{x}_t} \log q_{0t}(\mathbf{x}_t \mid \mathbf{x}_0) p_0(\mathbf{x}_0) d\mathbf{x}_0 \end{aligned}$$

Conditioning on a clean sample gives a tractable score

- During training, clean samples

$$\mathbf{x}_0 \sim p_0$$

are available from the dataset, and the forward corruption distribution is completely known:

$$q_{0t}(\mathbf{x}_t | \mathbf{x}_0) = \mathcal{N}(\mathbf{s}_t \mathbf{x}_0, \sigma_t^2 \mathbf{I}_d) .$$

- The marginal score can be written as an expectation of these conditional scores. Indeed,

$$\begin{aligned} \nabla_{\mathbf{x}_t} \log p_t(\mathbf{x}_t) &= \frac{1}{p_t(\mathbf{x}_t)} \nabla_{\mathbf{x}_t} p_t(\mathbf{x}_t) = \frac{1}{p_t(\mathbf{x}_t)} \int \nabla_{\mathbf{x}_t} q_{0t}(\mathbf{x}_t | \mathbf{x}_0) p_0(\mathbf{x}_0) d\mathbf{x}_0 \\ &= \frac{1}{p_t(\mathbf{x}_t)} \int q_{0t}(\mathbf{x}_t | \mathbf{x}_0) \nabla_{\mathbf{x}_t} \log q_{0t}(\mathbf{x}_t | \mathbf{x}_0) p_0(\mathbf{x}_0) d\mathbf{x}_0 \\ &= \int \nabla_{\mathbf{x}_t} \log q_{0t}(\mathbf{x}_t | \mathbf{x}_0) \frac{q_{0t}(\mathbf{x}_t | \mathbf{x}_0) p_0(\mathbf{x}_0)}{p_t(\mathbf{x}_t)} d\mathbf{x}_0 \end{aligned}$$

Conditioning on a clean sample gives a tractable score

- During training, clean samples

$$\mathbf{x}_0 \sim p_0$$

are available from the dataset, and the forward corruption distribution is completely known:

$$q_{0t}(\mathbf{x}_t \mid \mathbf{x}_0) = \mathcal{N}(\mathbf{s}_t \mathbf{x}_0, \sigma_t^2 \mathbf{I}_d).$$

- The marginal score can be written as an expectation of these conditional scores. Indeed,

$$\begin{aligned}\nabla_{\mathbf{x}_t} \log p_t(\mathbf{x}_t) &= \frac{1}{p_t(\mathbf{x}_t)} \nabla_{\mathbf{x}_t} p_t(\mathbf{x}_t) = \frac{1}{p_t(\mathbf{x}_t)} \int \nabla_{\mathbf{x}_t} q_{0t}(\mathbf{x}_t \mid \mathbf{x}_0) p_0(\mathbf{x}_0) d\mathbf{x}_0 \\ &= \frac{1}{p_t(\mathbf{x}_t)} \int q_{0t}(\mathbf{x}_t \mid \mathbf{x}_0) \nabla_{\mathbf{x}_t} \log q_{0t}(\mathbf{x}_t \mid \mathbf{x}_0) p_0(\mathbf{x}_0) d\mathbf{x}_0 \\ &= \int \nabla_{\mathbf{x}_t} \log q_{0t}(\mathbf{x}_t \mid \mathbf{x}_0) \frac{q_{0t}(\mathbf{x}_t \mid \mathbf{x}_0) p_0(\mathbf{x}_0)}{p_t(\mathbf{x}_t)} d\mathbf{x}_0 \\ &= \int \nabla_{\mathbf{x}_t} \log q_{0t}(\mathbf{x}_t \mid \mathbf{x}_0) p(\mathbf{x}_0 \mid \mathbf{x}_t) d\mathbf{x}_0.\end{aligned}$$

The forward process provides an unbiased training target

- Therefore,

$$\nabla_{\mathbf{x}_t} \log p_t(\mathbf{x}_t) = \mathbb{E}[\nabla_{\mathbf{x}_t} \log q_{0t}(\mathbf{x}_t \mid \mathbf{x}_0) \mid \mathbf{x}_t].$$

The forward process provides an unbiased training target

- Therefore,

$$\nabla_{\mathbf{x}_t} \log p_t(\mathbf{x}_t) = \mathbb{E}[\nabla_{\mathbf{x}_t} \log q_{0t}(\mathbf{x}_t \mid \mathbf{x}_0) \mid \mathbf{x}_t].$$

- Thus, for a fixed noisy state $\mathbf{x}_t$, the conditional score is a random quantity whose conditional mean is exactly the desired marginal score.

The forward process provides an unbiased training target

- Therefore,

$$\nabla_{\mathbf{x}_t} \log p_t(\mathbf{x}_t) = \mathbb{E}[\nabla_{\mathbf{x}_t} \log q_{0t}(\mathbf{x}_t | \mathbf{x}_0) | \mathbf{x}_t] .$$

- Thus, for a fixed noisy state $\mathbf{x}_t$, the conditional score is a random quantity whose conditional mean is exactly the desired marginal score.
- For Gaussian corruption,

$$q_{0t}(\mathbf{x}_t | \mathbf{x}_0) = \mathcal{N}(s_t \mathbf{x}_0, \sigma_t^2 \mathbf{I}_d) ,$$

so its score is known exactly:

$$\nabla_{\mathbf{x}_t} \log q_{0t}(\mathbf{x}_t | \mathbf{x}_0) = -\frac{\mathbf{x}_t - s_t \mathbf{x}_0}{\sigma_t^2} .$$

The forward process provides an unbiased training target

- Therefore,

$$\nabla_{\mathbf{x}_t} \log p_t(\mathbf{x}_t) = \mathbb{E}[\nabla_{\mathbf{x}_t} \log q_{0t}(\mathbf{x}_t | \mathbf{x}_0) | \mathbf{x}_t].$$

- Thus, for a fixed noisy state $\mathbf{x}_t$, the conditional score is a random quantity whose conditional mean is exactly the desired marginal score.
- For Gaussian corruption,

$$q_{0t}(\mathbf{x}_t | \mathbf{x}_0) = \mathcal{N}(s_t \mathbf{x}_0, \sigma_t^2 \mathbf{I}_d),$$

so its score is known exactly:

$$\nabla_{\mathbf{x}_t} \log q_{0t}(\mathbf{x}_t | \mathbf{x}_0) = -\frac{\mathbf{x}_t - s_t \mathbf{x}_0}{\sigma_t^2}.$$

- If we have access to clean images, we can pick one sample $\mathbf{x}_0$, generate

$$t \sim \mathcal{U}[0, T], \quad \epsilon \sim \mathcal{N}(\mathbf{0}, \mathbf{I}_d),$$

and construct $\mathbf{x}_t = s_t \mathbf{x}_0 + \sigma_t \epsilon$. Therefore,

$$\nabla_{\mathbf{x}_t} \log q_{0t}(\mathbf{x}_t | \mathbf{x}_0) = -\frac{\epsilon}{\sigma_t}.$$

The forward process provides an unbiased training target

- Combining this with the previous identity,

$$\mathbb{E}\left[-\frac{\epsilon}{\sigma_t} \middle| \mathbf{x}_t\right] = \nabla_{\mathbf{x}_t} \log p_t(\mathbf{x}_t).$$

Hence, for a chosen data sample $\mathbf{x}_0$ and time t , if $\mathbf{x}_t$ is constructed as $\mathbf{s}_t\mathbf{x}_0 + \sigma_t\epsilon$ for some randomly generated $\epsilon \sim \mathcal{N}(\mathbf{0}, \mathbf{I}_d)$, then $-\frac{\epsilon}{\sigma_t}$ is an unbiased approximation for the marginal score, conditioned on $\mathbf{x}_t$.

The forward process provides an unbiased training target

- Combining this with the previous identity,

$$\mathbb{E}\left[-\frac{\epsilon}{\sigma_t} \mid \mathbf{x}_t\right] = \nabla_{\mathbf{x}_t} \log p_t(\mathbf{x}_t).$$

Hence, for a chosen data sample $\mathbf{x}_0$ and time t , if $\mathbf{x}_t$ is constructed as $\mathbf{s}_t\mathbf{x}_0 + \sigma_t\epsilon$ for some randomly generated $\epsilon \sim \mathcal{N}(\mathbf{0}, \mathbf{I}_d)$, then $-\frac{\epsilon}{\sigma_t}$ is an unbiased approximation for the marginal score, conditioned on $\mathbf{x}_t$.

- During generation, the clean sample $\mathbf{x}_0$ is precisely what we are trying to synthesize. Therefore neither $\mathbf{x}_0$ nor the conditional score

$$\nabla_{\mathbf{x}_t} \log q_{0t}(\mathbf{x}_t \mid \mathbf{x}_0)$$

is available.

Learn the score during training, use it during generation

- During training, both $\mathbf{x}_0$ and the injected noise ϵ are known, so we can train a neural network to regress the approximate score using

$$\mathcal{L}_{\text{score}} = \mathbb{E} \left[\lambda(t) \left\| \mathbf{s}_{\theta}(\mathbf{x}_t, t) + \frac{\epsilon}{\sigma_t} \right\|^2 \right].$$

Learn the score during training, use it during generation

- During training, both $\mathbf{x}_0$ and the injected noise ϵ are known, so we can train a neural network to regress the approximate score using

$$\mathcal{L}_{\text{score}} = \mathbb{E} \left[\lambda(t) \left\| s_{\theta}(\mathbf{x}_t, t) + \frac{\epsilon}{\sigma_t} \right\|^2 \right].$$

- Then during generation, we evaluate the trained network

$$s_{\theta}(\mathbf{x}_t, t) \approx \nabla_{\mathbf{x}_t} \log p_t(\mathbf{x}_t)$$

at the current noisy state and substitute it into the reverse dynamics:

$$d\mathbf{x}_t = \left[f(t)\mathbf{x}_t - \frac{1 + \eta_t^2}{2} g(t)^2 s_{\theta}(\mathbf{x}_t, t) \right] dt + \eta_t g(t) d\tilde{\mathbf{W}}_t, \quad t : T \rightarrow 0.$$

Starting from $\mathbf{x}_T \sim p_T$, numerical integration then produces

$$\mathbf{x}_T \longrightarrow \mathbf{x}_{t_{N-1}} \longrightarrow \cdots \longrightarrow \mathbf{x}_0.$$

Training a diffusion model: score prediction vs. noise prediction

Both formulations use the same synthetic corruption pair

$$\mathbf{x}_0 \sim p_0, \quad t \sim \mathcal{U}[0, T], \quad \epsilon \sim \mathcal{N}(\mathbf{0}, \mathbf{I}_d), \quad \mathbf{x}_t = s_t \mathbf{x}_0 + \sigma_t \epsilon.$$

Score prediction

Network: $s_\theta(\mathbf{x}_t, t) \approx \nabla_{\mathbf{x}_t} \log p_t(\mathbf{x}_t)$.

- 1: Sample $\mathbf{x}_0 \sim p_0$
- 2: Sample $t \sim \mathcal{U}[0, T]$
- 3: Sample $\epsilon \sim \mathcal{N}(\mathbf{0}, \mathbf{I}_d)$
- 4: Construct $\mathbf{x}_t \leftarrow s_t \mathbf{x}_0 + \sigma_t \epsilon$
- 5: Compute the score loss

$$\mathcal{L}_{\text{score}} = \lambda(t) \left\| s_\theta(\mathbf{x}_t, t) + \frac{\epsilon}{\sigma_t} \right\|^2$$

6: Update

$$\theta \leftarrow \theta - \eta \nabla_\theta \mathcal{L}_{\text{score}}$$

Noise prediction

Network: $\epsilon_\theta(\mathbf{x}_t, t) \approx \mathbb{E}[\epsilon \mid \mathbf{x}_t]$.

- 1: Sample $\mathbf{x}_0 \sim p_0$
- 2: Sample $t \sim \mathcal{U}[0, T]$
- 3: Sample $\epsilon \sim \mathcal{N}(\mathbf{0}, \mathbf{I}_d)$
- 4: Construct $\mathbf{x}_t \leftarrow s_t \mathbf{x}_0 + \sigma_t \epsilon$
- 5: Compute the noise loss

$$\mathcal{L}_\epsilon = w(t) \|\epsilon_\theta(\mathbf{x}_t, t) - \epsilon\|^2$$

6: Update

$$\theta \leftarrow \theta - \eta \nabla_\theta \mathcal{L}_\epsilon$$

Inference is primarily an ODE problem

From the same trained diffusion model, the latest works sample using Probability-flow ODE

$$\frac{d\mathbf{x}_t}{dt} = f(t)\mathbf{x}_t - \frac{1}{2}g(t)^2 s_\theta(\mathbf{x}_t, t).$$

- deterministic once $\mathbf{x}_T$ is fixed;
- permits standard high-order ODE solvers;

Inference is primarily an ODE problem

From the same trained diffusion model, the latest works sample using Probability-flow ODE

$$\frac{d\mathbf{x}_t}{dt} = f(t)\mathbf{x}_t - \frac{1}{2}g(t)^2 s_\theta(\mathbf{x}_t, t).$$

- deterministic once $\mathbf{x}_T$ is fixed;
- permits standard high-order ODE solvers;

With noise prediction,

$$\frac{d\mathbf{x}_t}{dt} = f(t)\mathbf{x}_t + \frac{g(t)^2}{2\sigma_t} \epsilon_\theta(\mathbf{x}_t, t).$$

Inference is primarily an ODE problem

From the same trained diffusion model, the latest works sample using Probability-flow ODE

$$\frac{d\mathbf{x}_t}{dt} = f(t)\mathbf{x}_t - \frac{1}{2}g(t)^2 s_\theta(\mathbf{x}_t, t).$$

- deterministic once $\mathbf{x}_T$ is fixed;
- permits standard high-order ODE solvers;

With noise prediction,

$$\frac{d\mathbf{x}_t}{dt} = f(t)\mathbf{x}_t + \frac{g(t)^2}{2\sigma_t} \epsilon_\theta(\mathbf{x}_t, t).$$

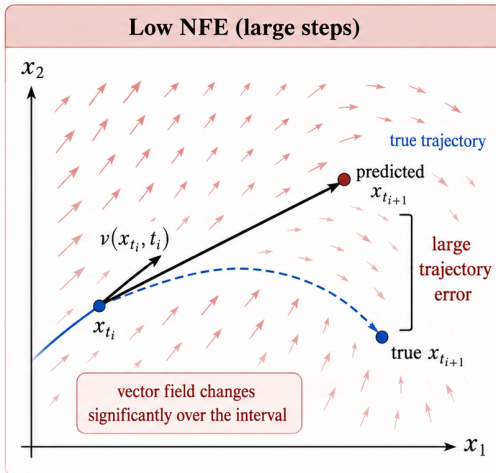
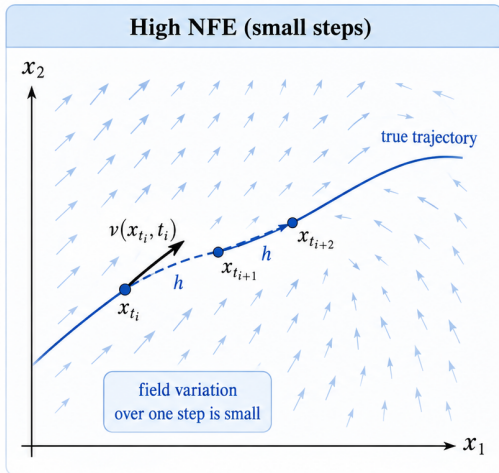
For fast training-free sampling (6-10 iterations), we focus on solving the reverse-time ODE accurately with very few neural-network evaluations.

Fast samplers: practical trade-offs

Method	NFE	Low-NFE behavior	Implementation details	Difficulty to implement and adapt
DDIM	20+	Stable and simple; first-order accuracy	Estimate $\hat{\mathbf{x}}_0$ and reuse the same noise prediction for the next update	Low
DPM-Solver++	10–20	Strong low-NFE performance; guidance can affect stability	Specialized exponential-integrator updates; higher orders require stored model evaluations and order-dependent coefficients	Medium
UniPC	6–20	Higher-order predictor–corrector; effective at few steps	Construct predictor and corrector coefficients; maintain model-history terms and perform order-dependent correction	Medium–High
STORK	6–20	Stability-oriented for aggressive low-step integration	Taylor-based derivative approximation with stabilized Runge–Kutta / virtual-stage constructions	High

Reducing NFE generally requires more sophisticated numerical solvers and additional solver-specific design choices.

What makes very-low-NFE sampling difficult?



Low NFE means larger step sizes. A solver that uses only the local field $v(x_{t_i}, t_i)$ can become inaccurate because the vector field varies substantially along the interval.

What makes very-low-NFE sampling difficult?

- **Each network evaluation must cover a large interval.**

The exact evolution satisfies

$$\mathbf{x}_{t_{i+1}} = \mathbf{x}_{t_i} + \int_{t_i}^{t_{i+1}} \mathbf{v}_{\theta}(\mathbf{x}_t, t) dt.$$

With few sampling points, $|t_{i+1} - t_i|$ is large, so accurately approximating this integral from only a few field evaluations is difficult.

What makes very-low-NFE sampling difficult?

- **Each network evaluation must cover a large interval.**

The exact evolution satisfies

$$\mathbf{x}_{t_{i+1}} = \mathbf{x}_{t_i} + \int_{t_i}^{t_{i+1}} \mathbf{v}_{\theta}(\mathbf{x}_t, t) dt.$$

With few sampling points, $|t_{i+1} - t_i|$ is large, so accurately approximating this integral from only a few field evaluations is difficult.

- **Errors accumulate along the reverse trajectory.**

Each numerical update is computed from an already approximate state. Hence, an error made at one step affects the network evaluation at the next step and can propagate through all subsequent updates.

What makes very-low-NFE sampling difficult?

- **Each network evaluation must cover a large interval.**

The exact evolution satisfies

$$\mathbf{x}_{t_{i+1}} = \mathbf{x}_{t_i} + \int_{t_i}^{t_{i+1}} \mathbf{v}_{\theta}(\mathbf{x}_t, t) dt.$$

With few sampling points, $|t_{i+1} - t_i|$ is large, so accurately approximating this integral from only a few field evaluations is difficult.

- **Errors accumulate along the reverse trajectory.**

Each numerical update is computed from an already approximate state. Hence, an error made at one step affects the network evaluation at the next step and can propagate through all subsequent updates.

- **At low NFE, we cannot afford an additional network evaluation to correct this error.**

A fresh model evaluation could provide new information about the current trajectory, but it would increase both NFE and sampling time. Therefore, any correction must rely only on the network outputs that have already been computed and stored.

CAB: rectify, predict, and correct

Rectify $\longrightarrow$ **Predict** $\longrightarrow$ **Correct**

- **Simple numerical structure.**

CAB consists of three transparent operations: change coordinates, make a higher-order prediction, and correct the prediction using information already available.

CAB: rectify, predict, and correct

Rectify $\longrightarrow$ Predict $\longrightarrow$ Correct

- **Simple numerical structure.**

CAB consists of three transparent operations: change coordinates, make a higher-order prediction, and correct the prediction using information already available.

- **Designed for the challenging 6–10 NFE regime.**

The method specifically targets large-step sampling, where conventional first- and higher-order approximations can accumulate substantial trajectory error.

CAB: rectify, predict, and correct

Rectify $\longrightarrow$ Predict $\longrightarrow$ Correct

- **Simple numerical structure.**

CAB consists of three transparent operations: change coordinates, make a higher-order prediction, and correct the prediction using information already available.

- **Designed for the challenging 6–10 NFE regime.**

The method specifically targets large-step sampling, where conventional first- and higher-order approximations can accumulate substantial trajectory error.

- **No additional neural-network evaluations.**

The correction is constructed entirely from previously computed model outputs; therefore, the corrector introduces *zero extra NFE*.

CAB: rectify, predict, and correct

Rectify $\longrightarrow$ Predict $\longrightarrow$ Correct

- **Simple numerical structure.**

CAB consists of three transparent operations: change coordinates, make a higher-order prediction, and correct the prediction using information already available.

- **Designed for the challenging 6–10 NFE regime.**

The method specifically targets large-step sampling, where conventional first- and higher-order approximations can accumulate substantial trajectory error.

- **No additional neural-network evaluations.**

The correction is constructed entirely from previously computed model outputs; therefore, the corrector introduces *zero extra NFE*.

- **Very small implementation overhead.**

Only a few previous network predictions need to be stored, followed by simple linear combinations in the solver update.

Step 1: rectify the reverse-time ODE

- Write the probability-flow ODE in noise-prediction form as

$$\frac{d\mathbf{x}_t}{dt} = \frac{\dot{s}_t}{s_t} \mathbf{x}_t + s_t \dot{\rho}_t \epsilon_{\theta}(\mathbf{x}_t, t), \quad \rho_t := \frac{\sigma_t}{s_t}.$$

Step 1: rectify the reverse-time ODE

- Write the probability-flow ODE in noise-prediction form as

$$\frac{d\mathbf{x}_t}{dt} = \frac{\dot{s}_t}{s_t} \mathbf{x}_t + s_t \dot{\rho}_t \epsilon_{\theta}(\mathbf{x}_t, t), \quad \rho_t := \frac{\sigma_t}{s_t}.$$

- Introduce the rescaled state $\mathbf{y}_t := \frac{\mathbf{x}_t}{s_t}$, we get

$$\frac{d\mathbf{y}_t}{dt} = \dot{\rho}_t \epsilon_{\theta}(s_t \mathbf{y}_t, t).$$

Step 1: rectify the reverse-time ODE

- Write the probability-flow ODE in noise-prediction form as

$$\frac{d\mathbf{x}_t}{dt} = \frac{\dot{s}_t}{s_t} \mathbf{x}_t + s_t \dot{\rho}_t \epsilon_{\theta}(\mathbf{x}_t, t), \quad \rho_t := \frac{\sigma_t}{s_t}.$$

- Introduce the rescaled state $\mathbf{y}_t := \frac{\mathbf{x}_t}{s_t}$, we get

$$\frac{d\mathbf{y}_t}{dt} = \dot{\rho}_t \epsilon_{\theta}(s_t \mathbf{y}_t, t).$$

- Reparameterizing the trajectory by ρ gives

$$\frac{d\widehat{\mathbf{y}}_{\rho}}{d\rho} = \epsilon_{\theta}\left(s_{t(\rho)} \widehat{\mathbf{y}}_{\rho}, t(\rho)\right).$$

Step 1: rectify the reverse-time ODE

- Write the probability-flow ODE in noise-prediction form as

$$\frac{d\mathbf{x}_t}{dt} = \frac{\dot{s}_t}{s_t} \mathbf{x}_t + s_t \dot{\rho}_t \epsilon_{\theta}(\mathbf{x}_t, t), \quad \rho_t := \frac{\sigma_t}{s_t}.$$

- Introduce the rescaled state $\mathbf{y}_t := \frac{\mathbf{x}_t}{s_t}$, we get

$$\frac{d\mathbf{y}_t}{dt} = \dot{\rho}_t \epsilon_{\theta}(s_t \mathbf{y}_t, t).$$

- Reparameterizing the trajectory by ρ gives

$$\frac{d\hat{\mathbf{y}}_{\rho}}{d\rho} = \epsilon_{\theta}(s_{t(\rho)} \hat{\mathbf{y}}_{\rho}, t(\rho)).$$

- Thus the schedule-induced linear term has disappeared completely. The numerical solver now integrates only the learned nonlinear field.

Step 1: rectify the reverse-time ODE

- Write the probability-flow ODE in noise-prediction form as

$$\frac{d\mathbf{x}_t}{dt} = \frac{\dot{s}_t}{s_t} \mathbf{x}_t + s_t \dot{\rho}_t \epsilon_{\theta}(\mathbf{x}_t, t), \quad \rho_t := \frac{\sigma_t}{s_t}.$$

- Introduce the rescaled state $\mathbf{y}_t := \frac{\mathbf{x}_t}{s_t}$, we get

$$\frac{d\mathbf{y}_t}{dt} = \dot{\rho}_t \epsilon_{\theta}(s_t \mathbf{y}_t, t).$$

- Reparameterizing the trajectory by ρ gives

$$\frac{d\hat{\mathbf{y}}_{\rho}}{d\rho} = \epsilon_{\theta}\left(s_{t(\rho)} \hat{\mathbf{y}}_{\rho}, t(\rho)\right).$$

- Thus the schedule-induced linear term has disappeared completely. The numerical solver now integrates only the learned nonlinear field.
- Importantly, even if $t_0, t_1, \dots, t_N$ are uniformly spaced, the transformed points $\rho_{t_0}, \rho_{t_1}, \dots, \rho_{t_N}$ are generally nonuniformly spaced. Hence we define $h_i := \rho_{t_{i+1}} - \rho_{t_i}$.

Step 2: use a higher-order Adams–Bashforth predictor ($t_i \rightarrow t_{i+1}$)

$$\text{Define } \epsilon_i := \epsilon_{\theta}(\mathbf{s}_{t_i} \mathbf{y}_{t_i}, t_i), \quad h_i := \rho_{t_{i+1}} - \rho_{t_i}, \quad r_i := \frac{h_i}{h_{i-1}}.$$

- A first-order method assumes that the current field ϵ_i represents the dynamics over the entire next interval: $\mathbf{y}_{t_{i+1}} \approx \mathbf{y}_{t_i} + h_i \epsilon_i$.

Step 2: use a higher-order Adams–Bashforth predictor ($t_i \rightarrow t_{i+1}$)

$$\text{Define } \epsilon_i := \epsilon_\theta(\mathbf{s}_{t_i} \mathbf{y}_{t_i}, t_i), \quad h_i := \rho_{t_{i+1}} - \rho_{t_i}, \quad r_i := \frac{h_i}{h_{i-1}}.$$

- A first-order method assumes that the current field ϵ_i represents the dynamics over the entire next interval: $\mathbf{y}_{t_{i+1}} \approx \mathbf{y}_{t_i} + h_i \epsilon_i$.
- AB2 instead uses the two already available evaluations ϵ_{i-1} and ϵ_i :

$$\mathbf{y}_{t_{i+1}}^{\text{pred}} = \mathbf{y}_{t_i} + h_i \left[\left(1 + \frac{r_i}{2}\right) \epsilon_i - \frac{r_i}{2} \epsilon_{i-1} \right].$$

Step 2: use a higher-order Adams–Bashforth predictor ($t_i \rightarrow t_{i+1}$)

$$\text{Define } \epsilon_i := \epsilon_\theta(\mathbf{s}_{t_i} \mathbf{y}_{t_i}, t_i), \quad h_i := \rho_{t_{i+1}} - \rho_{t_i}, \quad r_i := \frac{h_i}{h_{i-1}}.$$

- A first-order method assumes that the current field ϵ_i represents the dynamics over the entire next interval: $\mathbf{y}_{t_{i+1}} \approx \mathbf{y}_{t_i} + h_i \epsilon_i$.
- AB2 instead uses the two already available evaluations ϵ_{i-1} and ϵ_i :

$$\mathbf{y}_{t_{i+1}}^{\text{pred}} = \mathbf{y}_{t_i} + h_i \left[\left(1 + \frac{r_i}{2}\right) \epsilon_i - \frac{r_i}{2} \epsilon_{i-1} \right].$$

- AB2 assumes that the learned field continues approximately along the linear trend,

$$\epsilon_{i-1} \longrightarrow \epsilon_i.$$

It extrapolates this trend over the next interval and integrates the resulting line. The area under this linear approximation gives exactly the coefficients above.

Step 2: use a higher-order Adams–Bashforth predictor ($t_i \rightarrow t_{i+1}$)

$$\text{Define } \epsilon_i := \epsilon_\theta(\mathbf{s}_{t_i} \mathbf{y}_{t_i}, t_i), \quad h_i := \rho_{t_{i+1}} - \rho_{t_i}, \quad r_i := \frac{h_i}{h_{i-1}}.$$

- A first-order method assumes that the current field ϵ_i represents the dynamics over the entire next interval: $\mathbf{y}_{t_{i+1}} \approx \mathbf{y}_{t_i} + h_i \epsilon_i$.
- AB2 instead uses the two already available evaluations ϵ_{i-1} and ϵ_i :

$$\mathbf{y}_{t_{i+1}}^{\text{pred}} = \mathbf{y}_{t_i} + h_i \left[\left(1 + \frac{r_i}{2}\right) \epsilon_i - \frac{r_i}{2} \epsilon_{i-1} \right].$$

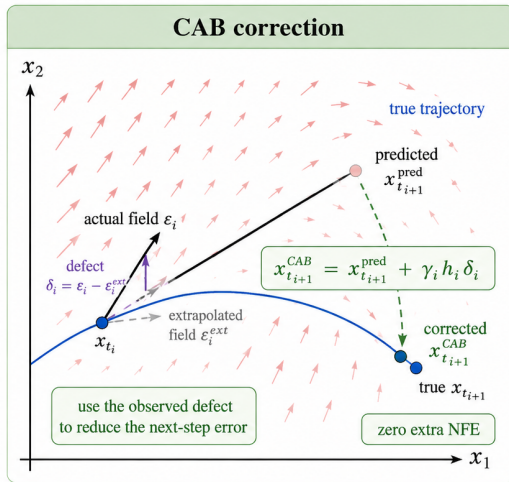
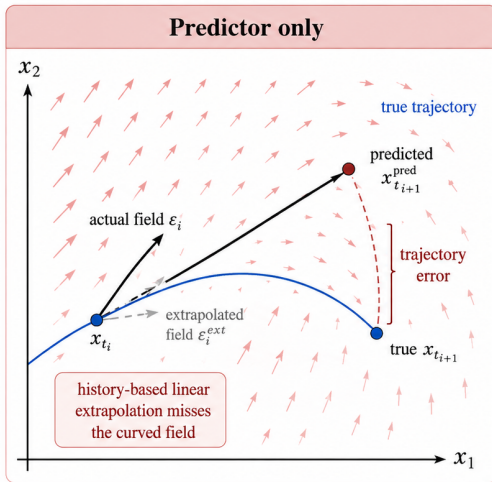
- AB2 assumes that the learned field continues approximately along the linear trend,

$$\epsilon_{i-1} \longrightarrow \epsilon_i.$$

It extrapolates this trend over the next interval and integrates the resulting line. The area under this linear approximation gives exactly the coefficients above.

- AB2 has second-order global accuracy, compared with first-order accuracy for Euler. Therefore, its discretization error decreases faster as the sampling grid is refined: halving the step size reduces the leading error by roughly 2× for Euler, but by roughly 4× for AB2.

Step 3: correct the error revealed by the field history



CAB does not assume the field is exactly linear; it measures the failure of linear extrapolation through the observed defect and feeds that information into the next large-step update.

Step 3: correct the error revealed by the field history

- AB2 assumes that the recent trend in the learned field continues approximately linearly. At very low NFE, however, the steps are large, so this extrapolation can become inaccurate.

Step 3: correct the error revealed by the field history

- AB2 assumes that the recent trend in the learned field continues approximately linearly. At very low NFE, however, the steps are large, so this extrapolation can become inaccurate.
- Using the two previous field values, we can ask, What would the past history of ϵ_{i-1} and ϵ_{i-2} have predicted for the field at the current point t_i ?

Step 3: correct the error revealed by the field history

- AB2 assumes that the recent trend in the learned field continues approximately linearly. At very low NFE, however, the steps are large, so this extrapolation can become inaccurate.
- Using the two previous field values, we can ask, What would the past history of ϵ_{i-1} and ϵ_{i-2} have predicted for the field at the current point t_i ? The linear extrapolation gives

$$\epsilon_i^{\text{ext}} = (1 + r_{i-1})\epsilon_{i-1} - r_{i-1}\epsilon_{i-2}.$$

Step 3: correct the error revealed by the field history

- AB2 assumes that the recent trend in the learned field continues approximately linearly. At very low NFE, however, the steps are large, so this extrapolation can become inaccurate.
- Using the two previous field values, we can ask, What would the past history of ϵ_{i-1} and ϵ_{i-2} have predicted for the field at the current point t_i ? The linear extrapolation gives

$$\epsilon_i^{\text{ext}} = (1 + r_{i-1})\epsilon_{i-1} - r_{i-1}\epsilon_{i-2}.$$

- But the actual network output at the current point is $\epsilon_i = \epsilon_\theta(\mathbf{s}_{t_i}\mathbf{y}_{t_i}, t_i)$. Therefore, we can directly measure how wrong the historical extrapolation was:

$$\delta_i := \epsilon_i - \epsilon_i^{\text{ext}}.$$

Step 3: correct the error revealed by the field history

- AB2 assumes that the recent trend in the learned field continues approximately linearly. At very low NFE, however, the steps are large, so this extrapolation can become inaccurate.
- Using the two previous field values, we can ask, What would the past history of ϵ_{i-1} and ϵ_{i-2} have predicted for the field at the current point t_i ? The linear extrapolation gives

$$\epsilon_i^{\text{ext}} = (1 + r_{i-1})\epsilon_{i-1} - r_{i-1}\epsilon_{i-2}.$$

- But the actual network output at the current point is $\epsilon_i = \epsilon_\theta(\mathbf{s}_{t_i}\mathbf{y}_{t_i}, t_i)$. Therefore, we can directly measure how wrong the historical extrapolation was:

$$\delta_i := \epsilon_i - \epsilon_i^{\text{ext}}.$$

- A small δ_i means that the recent history predicted the field well; a large δ_i indicates that the trajectory has begun to deviate from that extrapolated trend.

Step 3: correct the error revealed by the field history

- AB2 assumes that the recent trend in the learned field continues approximately linearly. At very low NFE, however, the steps are large, so this extrapolation can become inaccurate.
- Using the two previous field values, we can ask, What would the past history of ϵ_{i-1} and ϵ_{i-2} have predicted for the field at the current point t_i ? The linear extrapolation gives

$$\epsilon_i^{\text{ext}} = (1 + r_{i-1})\epsilon_{i-1} - r_{i-1}\epsilon_{i-2}.$$

- But the actual network output at the current point is $\epsilon_i = \epsilon_\theta(\mathbf{s}_{t_i}\mathbf{y}_{t_i}, t_i)$. Therefore, we can directly measure how wrong the historical extrapolation was:

$$\delta_i := \epsilon_i - \epsilon_i^{\text{ext}}.$$

- A small δ_i means that the recent history predicted the field well; a large δ_i indicates that the trajectory has begun to deviate from that extrapolated trend.
- CAB uses this already observed error at t_i to correct the next AB2 prediction at t_{i+1} :

$$\mathbf{y}_{t_{i+1}} = \mathbf{y}_{t_{i+1}}^{\text{pred}} + \gamma_i h_i (\epsilon_i - \epsilon_i^{\text{ext}}).$$

Accuracy guarantee for CAB-2

Informal Theorem. Consider CAB-2 applied to the rectified ODE

$$\frac{d\mathbf{y}}{d\rho} = \epsilon_{\theta}(\mathbf{y}, \rho)$$

on a possibly nonuniform ρ -grid. Assume that

- the rectified learned field is sufficiently smooth and Lipschitz in $\mathbf{y}$;
- neighboring step-size ratios remain well behaved;
- the correction weights γ_i remain bounded; and
- the starting values are sufficiently accurate.

Accuracy guarantee for CAB-2

Informal Theorem. Consider CAB-2 applied to the rectified ODE

$$\frac{d\mathbf{y}}{d\rho} = \epsilon_{\theta}(\mathbf{y}, \rho)$$

on a possibly nonuniform ρ -grid. Assume that

- the rectified learned field is sufficiently smooth and Lipschitz in $\mathbf{y}$;
- neighboring step-size ratios remain well behaved;
- the correction weights γ_i remain bounded; and
- the starting values are sufficiently accurate.

Then CAB-2 remains a **second-order method**. More precisely, if

$$h_{\max} := \max_i |\rho_{t_{i+1}} - \rho_{t_i}|,$$

then

$$\max_i \|\mathbf{y}_{t_i} - \widehat{\mathbf{y}}(\rho_{t_i})\| \leq C h_{\max}^2,$$

where C does not depend on the step size. Equivalently, CAB-2 has **third-order local truncation error** and **second-order global error**.

CAB-2 sampling algorithm

Input: $\mathbf{x}_{t_0} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$, reverse grid $T = t_0 > t_1 > \dots > t_N = 0$, schedules (s_t, σ_t) , noise predictor ϵ_θ , correction weights $\{\gamma_i\}$.

1. Startup

Evaluate the model

$$\epsilon_0 = \epsilon_\theta(\mathbf{x}_{t_0}, t_0), \quad \mathbf{y}_{t_0} = \frac{\mathbf{x}_{t_0}}{s_{t_0}}.$$

Move to the ρ -coordinate

$$\rho_{t_0} = \frac{\sigma_{t_0}}{s_{t_0}}, \quad \rho_{t_1} = \frac{\sigma_{t_1}}{s_{t_1}}, \\ h_0 = \rho_{t_1} - \rho_{t_0}.$$

Euler startup

$$\mathbf{y}_{t_1} = \mathbf{y}_{t_0} + h_0 \epsilon_0, \\ \mathbf{x}_{t_1} = s_{t_1} \mathbf{y}_{t_1}.$$

Stored information

$$\epsilon_0, \quad \mathbf{y}_{t_1}, \quad h_0.$$

2. For $i = 1, \dots, N-1$

Evaluate and compute the step

$$\epsilon_i = \epsilon_\theta(\mathbf{x}_{t_i}, t_i), \quad h_i = \rho_{t_{i+1}} - \rho_{t_i}, \quad r_i = \frac{h_i}{h_{i-1}}.$$

AB2 predictor

$$\mathbf{y}_{t_{i+1}}^{\text{pred}} = \mathbf{y}_{t_i} + h_i \left[\left(1 + \frac{r_i}{2}\right) \epsilon_i - \frac{r_i}{2} \epsilon_{i-1} \right].$$

Correct for $i \geq 2$

$$\epsilon_i^{\text{ext}} = (1 + r_{i-1}) \epsilon_{i-1} - r_{i-1} \epsilon_{i-2}, \\ \mathbf{y}_{t_{i+1}} = \mathbf{y}_{t_{i+1}}^{\text{pred}} + \gamma_i h_i (\epsilon_i - \epsilon_i^{\text{ext}}).$$

For $i = 1$, simply use

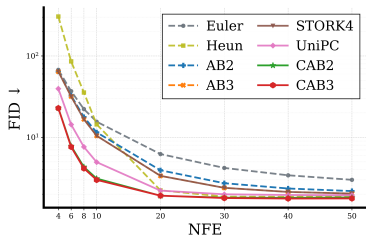
$$\mathbf{y}_{t_2} = \mathbf{y}_{t_2}^{\text{pred}}.$$

Map back to the original variable

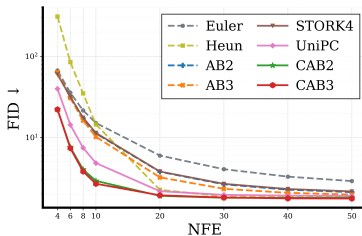
$$\mathbf{x}_{t_{i+1}} = s_{t_{i+1}} \mathbf{y}_{t_{i+1}}.$$

Output: $\mathbf{x}_{t_N}$

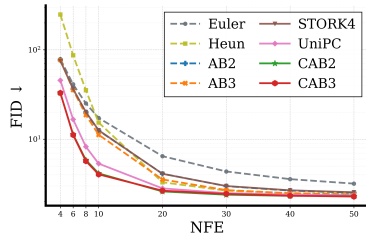
The advantage is concentrated in the low-NFE regime



CIFAR-10, VP



CIFAR-10, VE

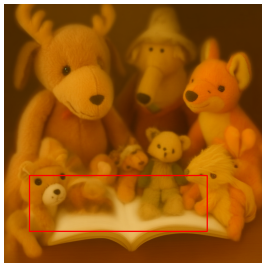


ImageNet, VP

- CAB improves most strongly where the numerical problem is hardest: the low-NFE regime.
- As NFE increases, all methods approach the same accurately resolved trajectory and the differences naturally shrink.

CAB-2 preserves semantics at only 8 NFEs

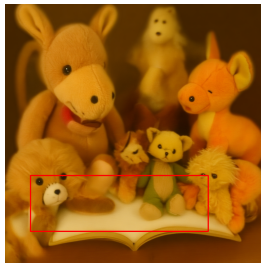
Euler



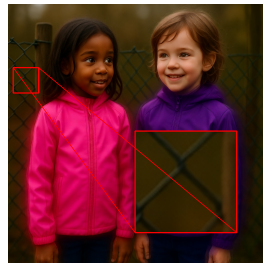
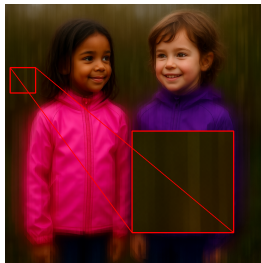
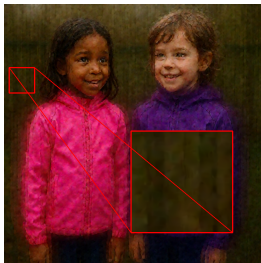
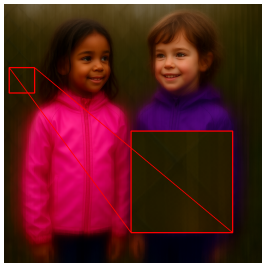
DPM-Solver++



STORK



CAB-2



“Stuffed animals looking at pictures of other animals in a book.” “Two kids in pink and purple jackets standing by a fence.”

Qualitative comparison



Prompt: “Two kids in pink and purple jackets standing by a fence.”

DiT/ImageNet: CAB-2 is strongest from 6 to 20 NFEs

NFE	Euler	DPM++	STORK	AB2	CAB-2
4	120.83	75.26	116.97	116.99	88.04
6	73.20	26.64	51.52	51.52	15.13
8	39.71	24.90	20.70	20.70	9.21
10	24.22	22.76	12.39	12.39	7.99
20	9.94	12.19	7.44	7.44	6.86

- FID ↓, DiT/ImageNet 256×256 , 10K samples.
- Comparing AB2 and CAB-2 isolates the effect of the correction: same predictor, same NFE, same grid.
- At the extreme 4-NFE point DPM-Solver++ remains stronger; from 6–20 NFEs CAB-2 gives the lowest FID in this comparison.

The improvement is not caused by Adams–Bashforth alone: the defect correction produces a large AB2-to-CAB-2 gap at equal NFE.

The same gain appears in diffusion sampling

DDIM



DPM-Solver++



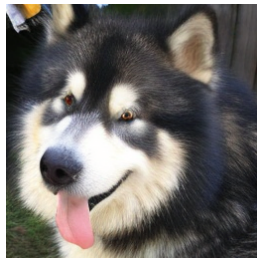
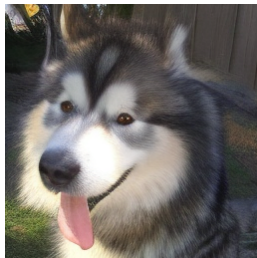
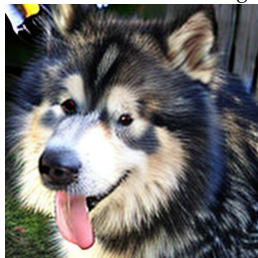
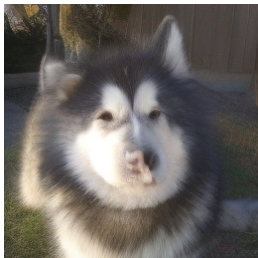
STORK



CAB-2



DiT/ImageNet at 8 NFEs



DiT/ImageNet at 10 NFEs

Conclusion

- A broad class of diffusion and flow models can be described through the affine Gaussian path

$$\mathbf{x}_t = \mathbf{s}_t \mathbf{x}_0 + \sigma_t \epsilon,$$

which leads to a common reverse-time ODE formulation for deterministic sampling.

Conclusion

- A broad class of diffusion and flow models can be described through the affine Gaussian path

$$\mathbf{x}_t = \mathbf{s}_t \mathbf{x}_0 + \sigma_t \epsilon,$$

which leads to a common reverse-time ODE formulation for deterministic sampling.

- At very low NFE, sampling becomes a challenging *large-step numerical integration problem*: a small number of network evaluations must accurately track a rapidly changing learned field.

Conclusion

- A broad class of diffusion and flow models can be described through the affine Gaussian path

$$\mathbf{x}_t = \mathbf{s}_t \mathbf{x}_0 + \sigma_t \epsilon,$$

which leads to a common reverse-time ODE formulation for deterministic sampling.

- At very low NFE, sampling becomes a challenging *large-step numerical integration problem*: a small number of network evaluations must accurately track a rapidly changing learned field.
- CAB addresses this through three simple operations:

$$\underbrace{\mathbf{y}_t = \frac{\mathbf{x}_t}{\mathbf{s}_t}, \quad \rho_t = \frac{\sigma_t}{\mathbf{s}_t}}_{\text{rectify}} \longrightarrow \underbrace{\text{AB2}}_{\text{predict}} \longrightarrow \underbrace{\gamma_i h_i (\epsilon_i - \epsilon_i^{\text{ext}})}_{\text{correct}}.$$

Conclusion

- A broad class of diffusion and flow models can be described through the affine Gaussian path

$$\mathbf{x}_t = \mathbf{s}_t \mathbf{x}_0 + \sigma_t \epsilon,$$

which leads to a common reverse-time ODE formulation for deterministic sampling.

- At very low NFE, sampling becomes a challenging *large-step numerical integration problem*: a small number of network evaluations must accurately track a rapidly changing learned field.
- CAB addresses this through three simple operations:

$$\underbrace{\mathbf{y}_t = \frac{\mathbf{x}_t}{\mathbf{s}_t}, \quad \rho_t = \frac{\sigma_t}{\mathbf{s}_t}}_{\text{rectify}} \longrightarrow \underbrace{\text{AB2}}_{\text{predict}} \longrightarrow \underbrace{\gamma_i h_i (\epsilon_i - \epsilon_i^{\text{ext}})}_{\text{correct}}.$$

- CAB-2 requires only **one new network evaluation per step**, requires no retraining, preserves second-order convergence, and improves sample quality particularly in the low-NFE regime.

- **Extremely coarse steps remain difficult.**

CAB corrects errors in multistep extrapolation, but when the learned trajectory has very strong curvature or rapidly changing dynamics, $|\rho_{t_{i+1}} - \rho_{t_i}|$ may still be too large for accurate explicit integration.

- **Extremely coarse steps remain difficult.**

CAB corrects errors in multistep extrapolation, but when the learned trajectory has very strong curvature or rapidly changing dynamics, $|\rho_{t_{i+1}} - \rho_{t_i}|$ may still be too large for accurate explicit integration.

- **The corrector strength is currently tuned per model.**

The weight γ_i controls how strongly the observed defect $\epsilon_i - \epsilon_i^{\text{ext}}$ is fed back into the update. A fixed value cannot necessarily be optimal across all models, timesteps, guidance strengths, and NFE budgets.

Limitations and future work

- **Extremely coarse steps remain difficult.**

CAB corrects errors in multistep extrapolation, but when the learned trajectory has very strong curvature or rapidly changing dynamics, $|\rho_{t_{i+1}} - \rho_{t_i}|$ may still be too large for accurate explicit integration.

- **The corrector strength is currently tuned per model.**

The weight γ_i controls how strongly the observed defect $\epsilon_i - \epsilon_i^{\text{ext}}$ is fed back into the update. A fixed value cannot necessarily be optimal across all models, timesteps, guidance strengths, and NFE budgets.

- **The sampling grid is not adapted to the learned dynamics.**

Large field variation may be concentrated in only a few regions of ρ -space. This suggests allocating steps according to the local difficulty of the trajectory rather than using a predetermined grid.

Selected references



J. Ho, A. Jain, P. Abbeel. Denoising Diffusion Probabilistic Models. NeurIPS, 2020.



Y. Song et al. Score-Based Generative Modeling through Stochastic Differential Equations. ICLR, 2021.



J. Song, C. Meng, S. Ermon. Denoising Diffusion Implicit Models. ICLR, 2021.



A. Nichol, P. Dhariwal. Improved Denoising Diffusion Probabilistic Models. ICML, 2021.



P. Dhariwal, A. Nichol. Diffusion Models Beat GANs on Image Synthesis. NeurIPS, 2021.



T. Karras et al. Elucidating the Design Space of Diffusion-Based Generative Models. NeurIPS, 2022.



C. Lu et al. DPM-Solver: A Fast ODE Solver for Diffusion Probabilistic Model Sampling. NeurIPS, 2022.



W. Zhao et al. UniPC: A Unified Predictor-Corrector Framework for Fast Sampling of Diffusion Models. NeurIPS, 2023.



Y. Lipman et al. Flow Matching for Generative Modeling. ICLR, 2023.



X. Liu, C. Gong, Q. Liu. Flow Straight and Fast: Learning to Generate and Transfer Data with Rectified Flow. ICLR, 2023.



A. Roy, P. Nair. CAB: Accelerating Flow and Diffusion Sampling via Rectification and Corrected Adams–Bashforth. arXiv:2605.16736v2, 2026.