Accelerating Diffusion Models

EECE695D: Efficient ML Systems

Spring 2025

Recall: Diffusion model

- A generative model, i.e., method to model the probability density $p(\mathbf{x})$
  - "denoising diffusion probabilistic model"

- Like many other generative models, $p(\mathbf{x})$ is generated as a **pushforward** of some easy-to-sample density (e.g., Gaussian)

$$\mathbf{z} \sim \mathcal{N}(0, \mathbf{I}_d) \longrightarrow \boxed{\text{Generator } f(\,\cdot\,)} \longrightarrow \mathbf{x} = f(\mathbf{z}) \sim p(\mathbf{x})$$

- **Goal.** Learn a good $f(\,\cdot\,)$ from samples of $\mathbf{x}$

Recall: Diffusion model

$$\mathbf{z} \sim \mathcal{N}(0,\mathbf{I}_d) \longrightarrow \boxed{\text{Generator } f(\,\cdot\,)} \longrightarrow \mathbf{x} = f(\mathbf{z}) \sim p(\mathbf{x})$$

- **Challenge.**
  - How should we generate the corresponding $\mathbf{z}^{(i)}$ for some sample $\mathbf{x}^{(i)}$?
  - Also, $f(\,\cdot\,)$ is likely to be very complicated
- **Idea.** There is a straightforward way to model f^{-1} (use $\mathbf{z}^{(i)} = f^{-1}(\mathbf{x}^{(i)})$)
  - Plus, this f^{-1} can be decomposed into many sub-functions

Recall: Diffusion model

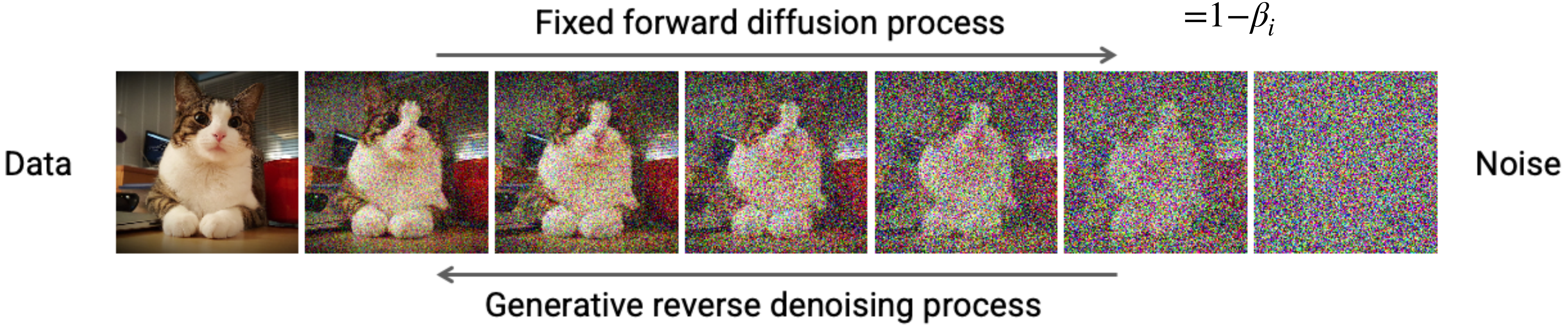
- **Forward diffusion.** Adds Gaussian noise gradually

$$q(\mathbf{x}_t \,|\, \mathbf{x}_{t-1}) = \mathcal{N}(\mathbf{x}_t \,|\, \sqrt{1-\beta_t}\mathbf{x}_{t-1}, \beta_t \mathbf{I})$$

β_t: analogous to "time" between t and $t-1$

  - Given $\mathbf{x}_0$, we can sample $\mathbf{x}_t$ as

$$\mathbf{x}_t = \sqrt{\bar{\alpha}_t}\mathbf{x}_0 + \sqrt{1-\bar{\alpha}_t}\epsilon, \qquad \bar{\alpha}_t = \prod_{i=1}^{t} \underbrace{\alpha_i}_{=1-\beta_i}$$



$$\mathbf{x}_0 \longrightarrow \mathbf{x}_1 \longrightarrow \mathbf{x}_2 \longrightarrow \quad (\cdots) \quad \longrightarrow \mathbf{z} = \mathbf{x}_T$$

Recall: Diffusion model

- **Reverse denoising**. Want to model $q(\mathbf{x}_{t-1} | \mathbf{x}_t)$
  - After some math*, one can realize that we can approximate

$$q(\mathbf{x}_{t-1} | \mathbf{x}_t) = \mathcal{N}(\mathbf{x}_{t-1} | \mu_\theta(\mathbf{x}_t), \sigma_t^2 \mathbf{I}_d)$$

where the mean $\mu_\theta(\mathbf{x}_t)$ can be written as:

$$\mu_\theta(\mathbf{x}_t, t) = \frac{1}{\sqrt{\alpha_t}} \left(\mathbf{x}_t - \frac{1 - \alpha_t}{\sqrt{1 - \bar{\alpha}_t}} \epsilon_\theta(\mathbf{x}_t, t) \right)$$

Noise model; to be trained from the data

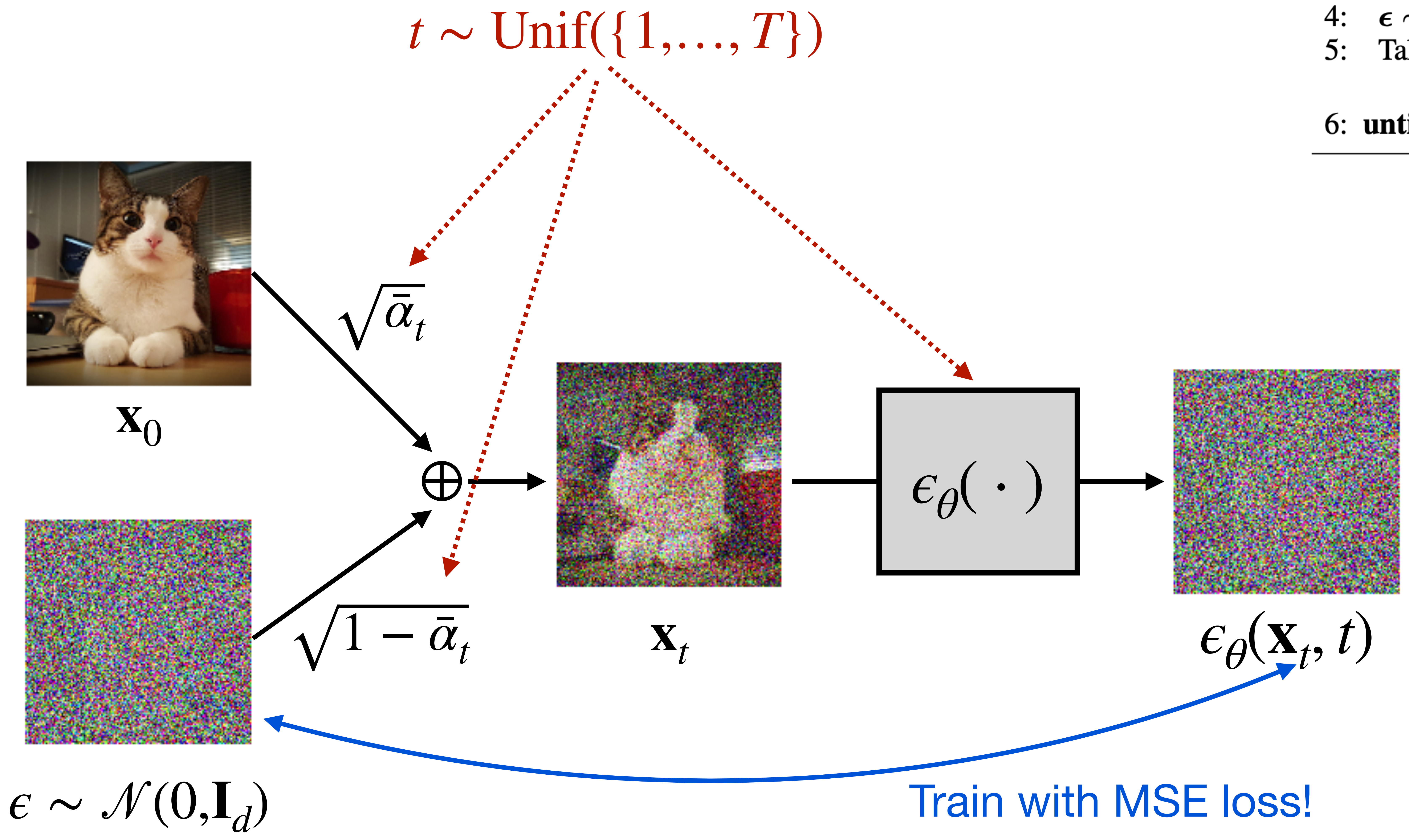
  - As the model is Gaussian, fitting the distribution is simply training with the squared loss

Recall: Diffusion model

- **Training**. Train by noise prediction

Algorithm 1 Training

1: **repeat**
2: $\mathbf{x}_0 \sim q(\mathbf{x}_0)$
3: $t \sim \text{Uniform}(\{1, \ldots, T\})$
4: $\boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$
5: Take gradient descent step on
$$\nabla_\theta \left\| \boldsymbol{\epsilon} - \boldsymbol{\epsilon}_\theta(\sqrt{\bar{\alpha}_t}\mathbf{x}_0 + \sqrt{1-\bar{\alpha}_t}\boldsymbol{\epsilon}, t) \right\|^2$$
6: **until** converged



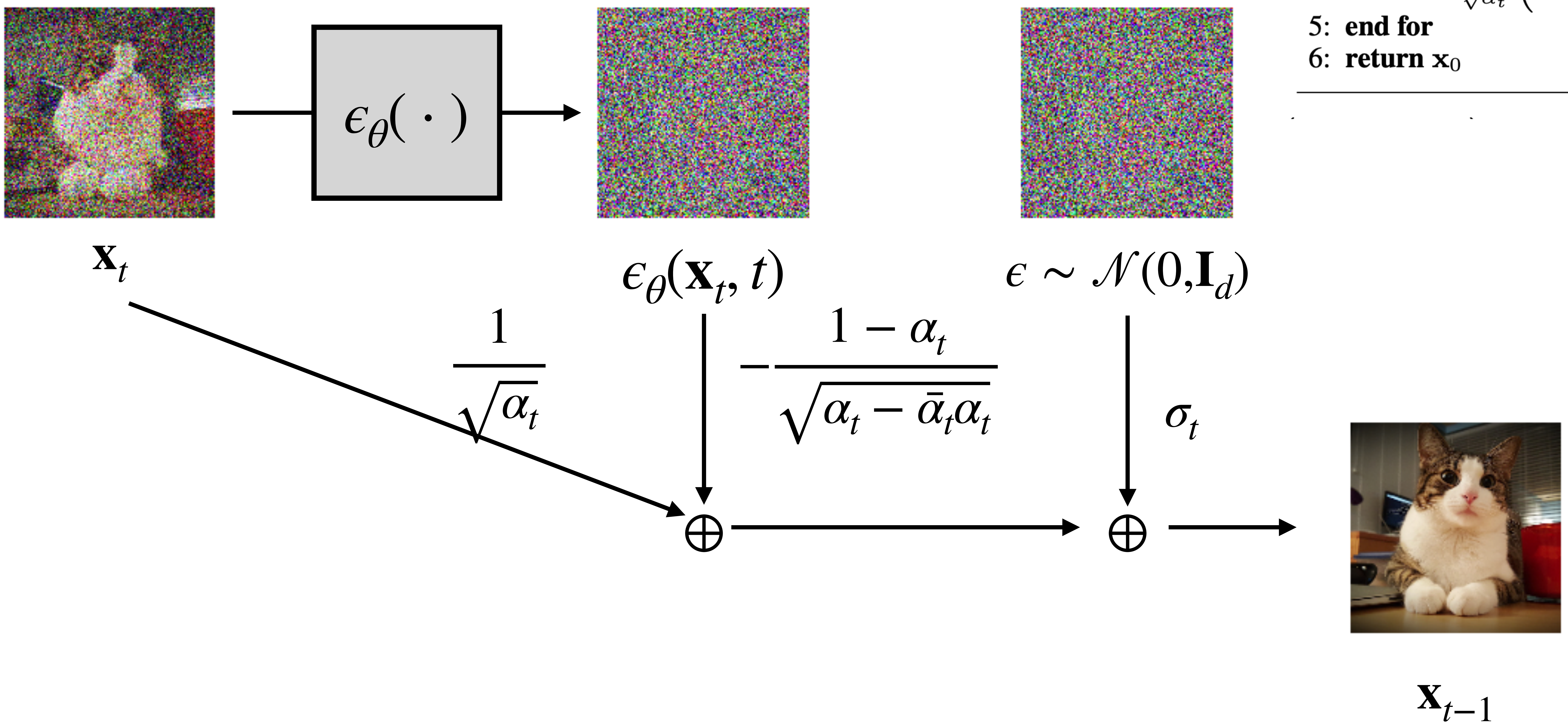
Ho et al., "Denoising Diffusion Probabilistic Models," NeurIPS 2020

Recall: Diffusion model

- **Sampling**. Step-by-step denoising

Algorithm 2 Sampling

1: $\mathbf{x}_T \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$
2: **for** $t = T, \ldots, 1$ **do**
3: $\quad \mathbf{z} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ if $t > 1$, else $\mathbf{z} = \mathbf{0}$
4: $\quad \mathbf{x}_{t-1} = \frac{1}{\sqrt{\alpha_t}}\left(\mathbf{x}_t - \frac{1-\alpha_t}{\sqrt{1-\bar{\alpha}_t}}\boldsymbol{\epsilon}_\theta(\mathbf{x}_t, t)\right) + \sigma_t \mathbf{z}$
5: **end for**
6: **return** $\mathbf{x}_0$



Ho et al., "Denoising Diffusion Probabilistic Models," NeurIPS 2020

Problem

- **Goal.** We want **fast & on-device** generation
  - Hopefully video editing as well

Approaches

- Many different approaches
  - SDE/ODE solvers
  - Reduce the number of denoising steps
    - Deterministic sampler (e.g., DDIM)
    - Distillation
  - Reduce the computational cost of denoising model
    - Compress the model (Presentation 1)
    - Re-use computed values (Presentation 2)
  - Parallel sampling (Presentation 3)

ODE solvers

- **Idea.** Consider infinitesimal time intervals
  - Recall that the forward diffusion is

$$q(\mathbf{x}_t \mid \mathbf{x}_{t-1}) = \mathcal{N}(\mathbf{x}_t \mid \sqrt{1-\beta_t}\mathbf{x}_{t-1}, \beta_t \mathbf{I})$$

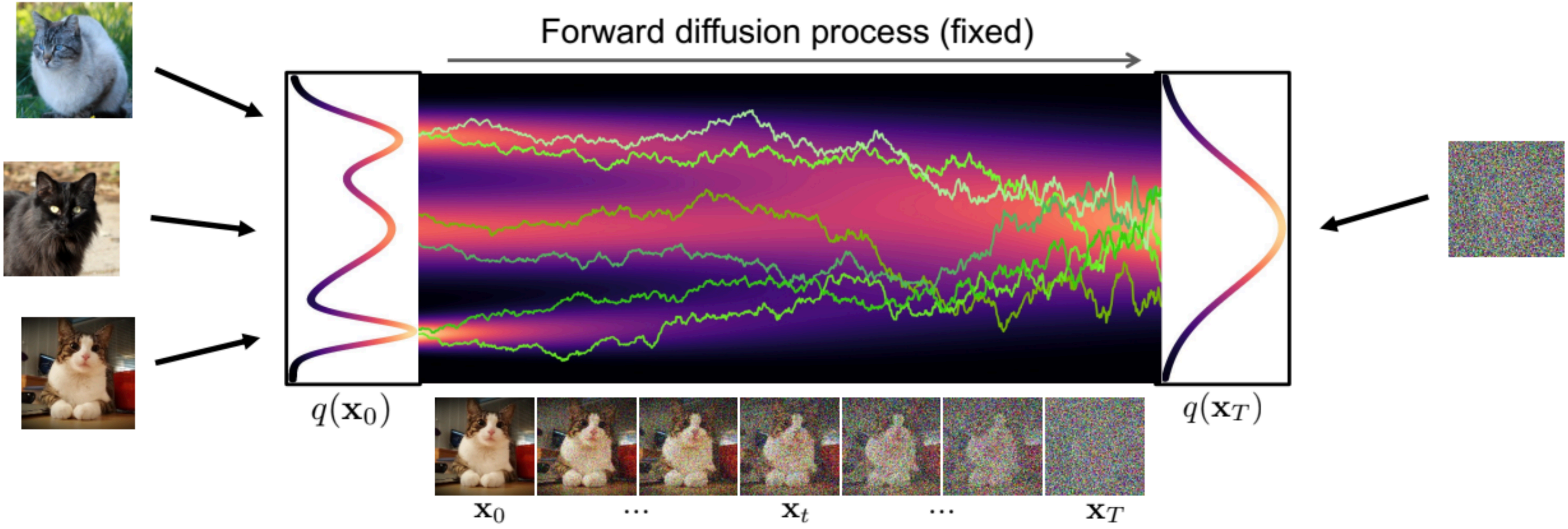
  - This becomes an **stochastic differential equation**

$$\begin{aligned} \mathbf{x}_t &= \sqrt{1-\beta(t)\Delta t}\,\mathbf{x}_{t-1} + \sqrt{\beta(t)\Delta t}\,\mathcal{N}(0,\mathbf{I}) \\ &\approx \mathbf{x}_{t-1} - \frac{\beta(t)\Delta t}{2}\mathbf{x}_{t-1} + \sqrt{\beta(t)\Delta t}\,\mathcal{N}(0,\mathbf{I}) \end{aligned}$$

$$\Rightarrow \qquad \mathrm{d}\mathbf{x}_t = -\frac{1}{2}\beta(t)\mathbf{x}_t \mathrm{d}t + \sqrt{\beta(t)}\,\mathrm{d}\omega_t$$

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

ODE solvers



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

ODE solvers

- The reverse diffusion process can be written as:

$$\mathrm{d}\mathbf{x}_t = -\frac{1}{2}\beta(t)\big(\mathbf{x}_t + 2\,\nabla_{\mathbf{x}_t}\log q_t(\mathbf{x}_t)\big)\mathrm{d}t + \sqrt{\beta(t)}\mathrm{d}\bar{\omega}_t$$

- We can train a neural network which approximates this "score function"
  - Use $q_t(\mathbf{x}_t\,|\,\mathbf{x}_0)$ for tractibility
  - The reverse can then be expressed as:

$$\mathrm{d}\mathbf{x}_t = -\frac{1}{2}\beta(t)\big(\mathbf{x}_t + 2\mathbf{s}_\theta(\mathbf{x}_t, t)\big)\mathrm{d}t + \sqrt{\beta(t)}\mathrm{d}\bar{\omega}_t$$

  - Use off-the-shelf SDE solvers.
    - Can also come up with ODE version, which is very fast!

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

Deterministic sampler

- **Idea.** We can play with the noise-adding procedure

$$q(\mathbf{x}_t \mid \mathbf{x}_{t-1}) = \mathcal{N}(\mathbf{x}_t \mid \sqrt{1-\beta_t}\mathbf{x}_{t-1}, \beta_t \mathbf{I})$$

  - Markov property (is there a reason why it should be so?)
  - Fixed (can we introduce learnable components?)

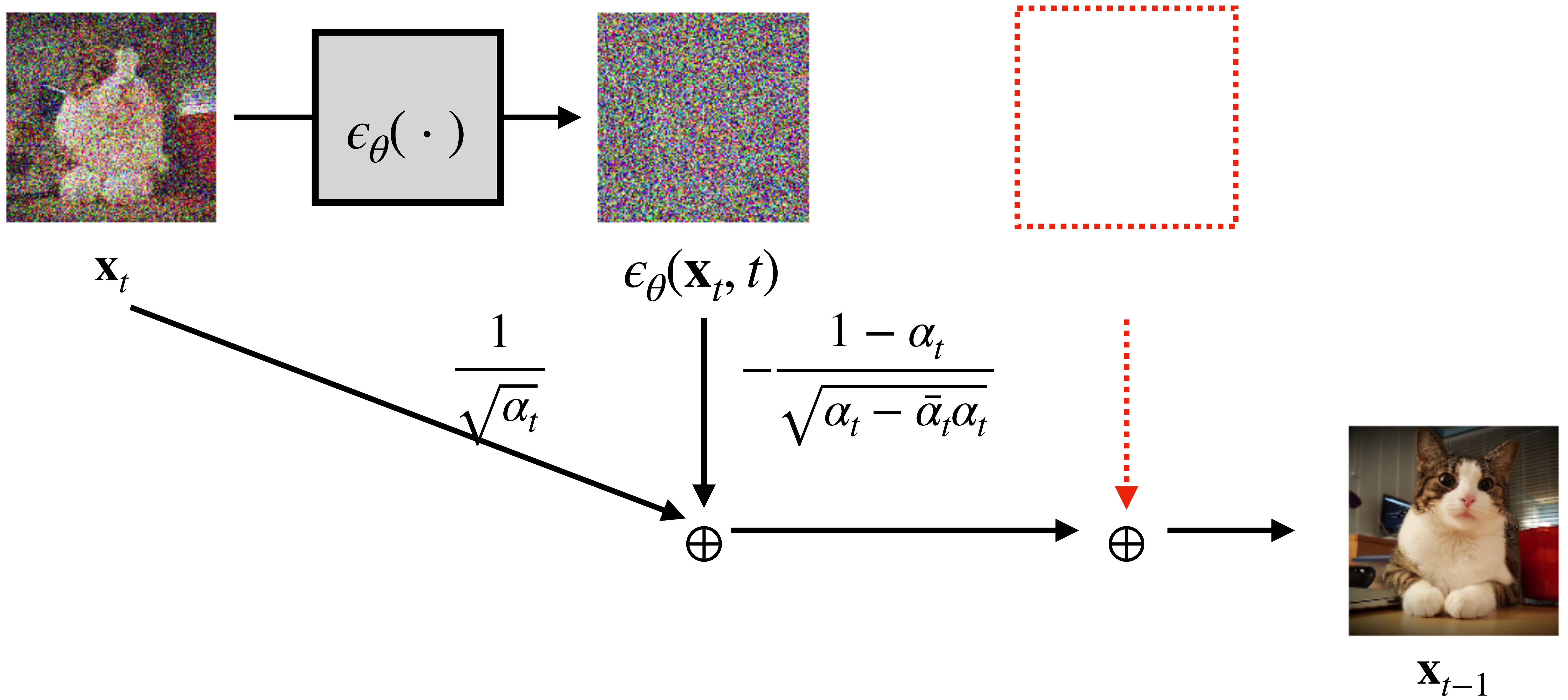
- **Desired.** Want faster "mixing" during forward
  - so that reverting them can be done in fewer steps

Data  Noise

Song et al., "Denoising Diffusion Implicit Models," ICLR 2021

Deterministic sampler

- **DDIM.** No noise-adding during the sampling
  - Theoretical motivations from approximating $q(\mathbf{x}_{t-1} \mid \mathbf{x}_t, \mathbf{x}_0)$ (not $q(\mathbf{x}_{t-1} \mid \mathbf{x}_t)$)
  - Same training procedure
  - Faster sampling (1000 → 25~40), but slightly weaker image diversity



Song et al., "Denoising Diffusion Implicit Models," ICLR 2021

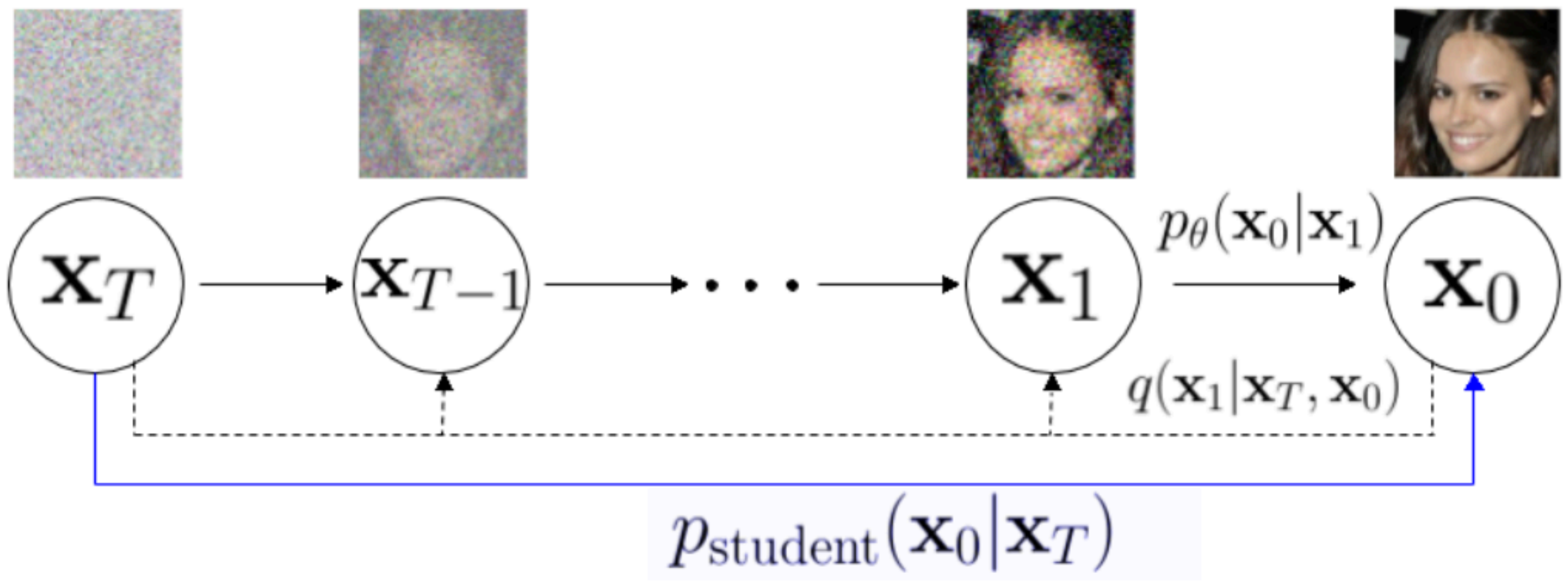
Distillation

- **Idea.** Simply distill a multi-step denoiser from a single-step one
  - Luhman & Luhman (2021) distills with the loss

$$L = \frac{1}{2}\mathbb{E}_{\mathbf{x}_T} \| f_{\text{stu}}(\mathbf{x}_T) - f_{\text{tea}}(\mathbf{x}_T) \|_2^2$$

    where $f(\cdot)$ denotes the mean of the estimated Gaussian.

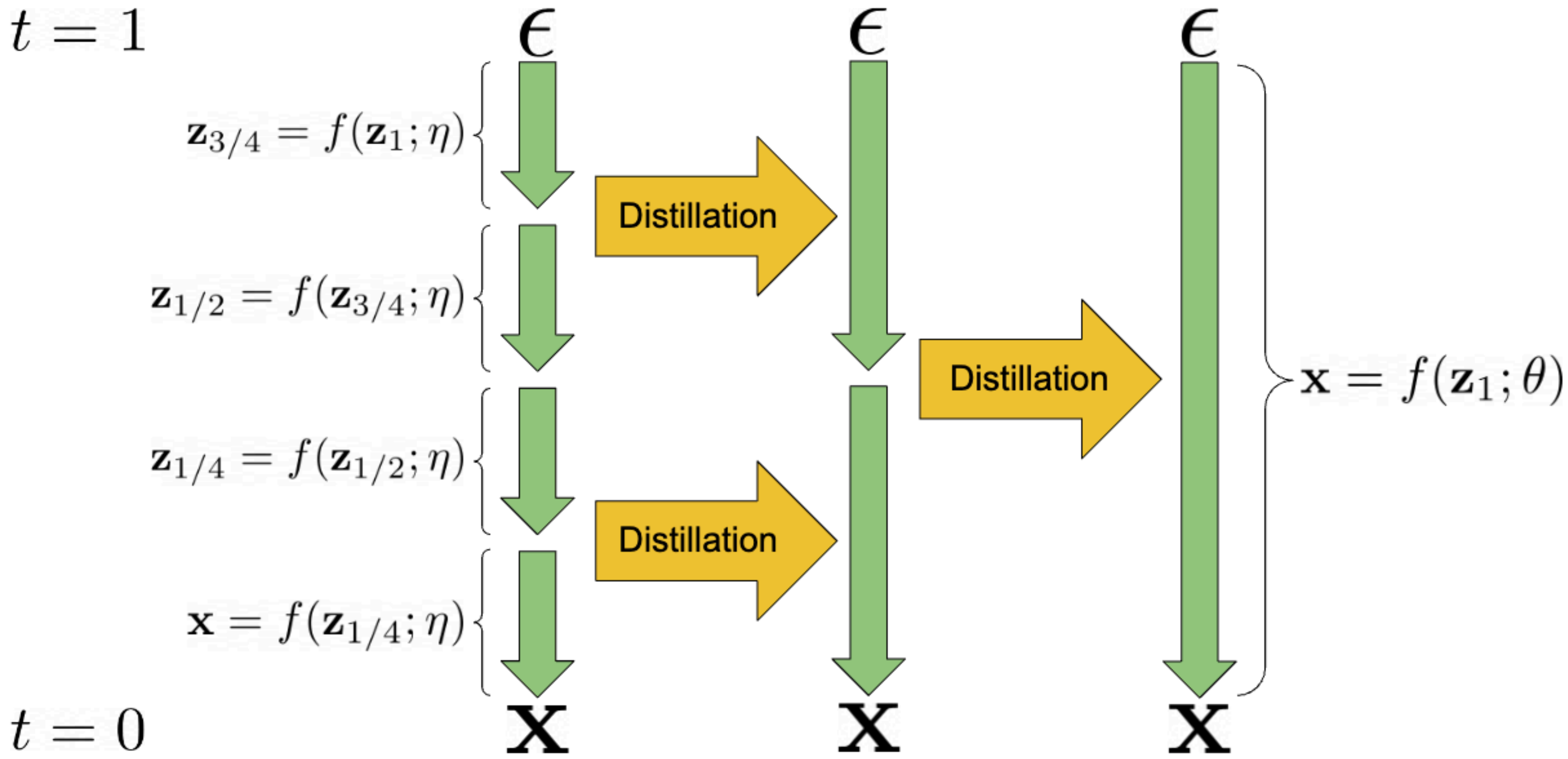
    - f_{tea} is generated by multi-step diffusion



Luhman and Luhman, "Knowledge Distillation in Iterative Generative Models for Improved Sampling Speed," arXiv 2021

Distillation

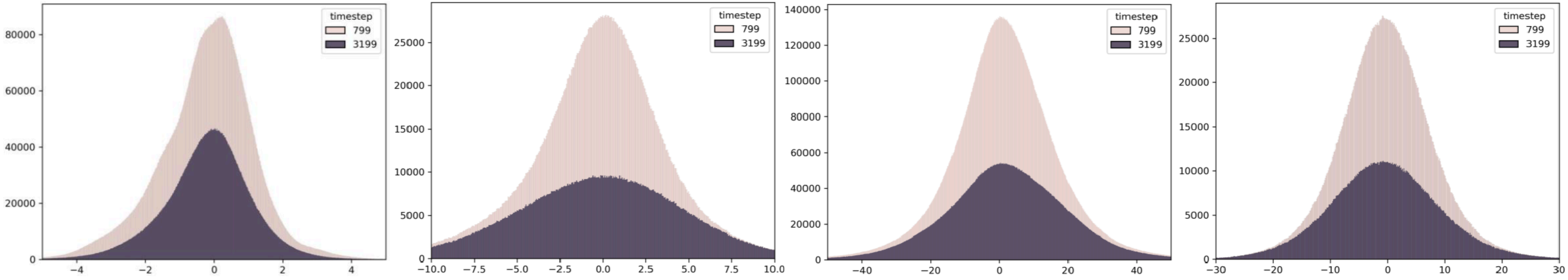
- Later works find that progressive distillation is beneficial, in general
  - No need to run full number of sampling steps with the original model



Salimans and Ho, "Progressive Distillation for Fast Sampling of Diffusion Models," ICLR 2022

Considerations in model compression

- A noteworthy characteristic of diffusion models is their time-dependency
  - The **activation distribution** changes from timestep to timestep
    - Requires a careful calibration of quantization range

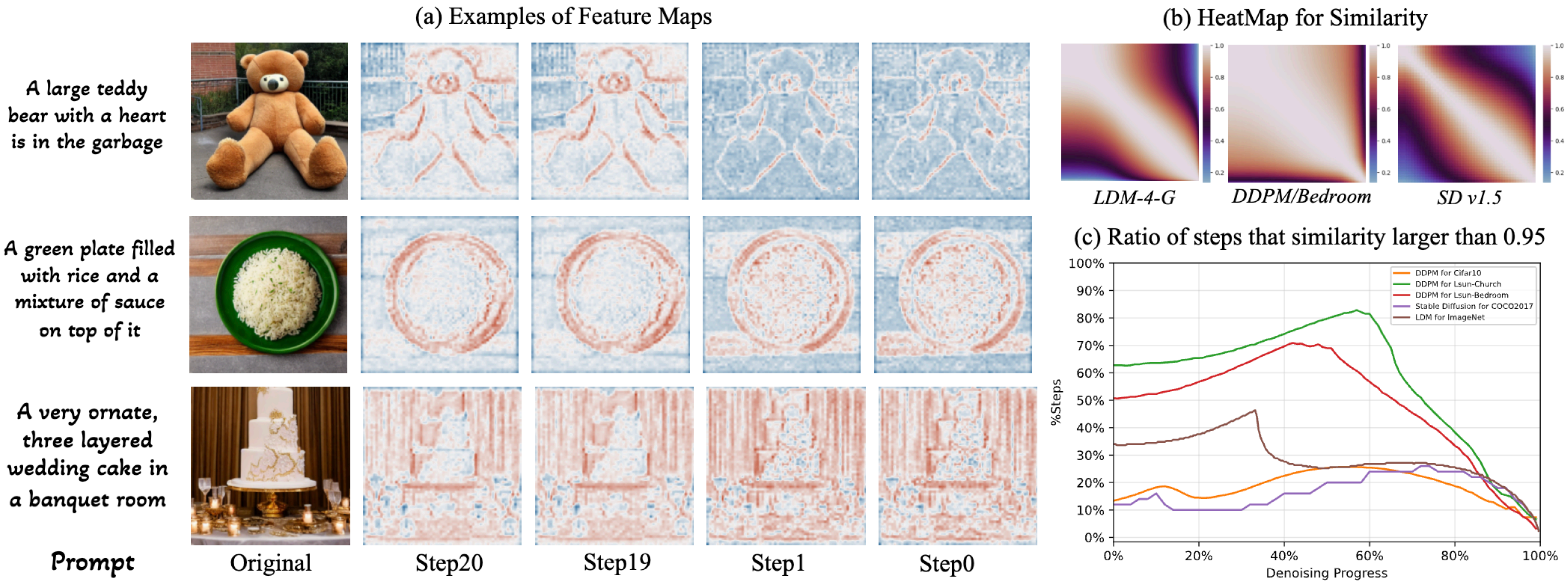


1st Layer 8th Layer 15th Layer 22ed Layer

Shang et al., "Post-training Quantization on Diffusion Models," CVPR 2023

Considerations in feature reuse

- There seems to be much **feature redundancy** across timesteps
  - Caching and reusing high-level features or attention can save computations at the expense of minimal quality degradation



Ma et al., "DeepCache: Accelerating Diffusion Models for Free," CVPR 2024

Further readings

- Consistency models
  - https://arxiv.org/abs/2303.01469
- Parallel sampling
  - https://arxiv.org/abs/2305.16317
- Early stopping
  - https://arxiv.org/abs/2205.12524

That’s it for today 🙌