A Practical Introduction to Diffusion Models

MIT 15.S60, IAP 2026
Session 5b

Peter Hoffman

Diffusion = noising + learned denoising

$$X_0 \sim p_{\text{data}} \quad \longrightarrow \quad X_T \approx \mathcal{N}(0, I) \qquad \text{(Noising process: data to noise, fixed)}$$

$$X_T \approx \mathcal{N}(0, I) \quad \longrightarrow \quad X_0 \sim p_{\text{data}} \qquad \text{(Denoising process: noise to data, learned)}$$

The forward SDE maps data to noise

Definition: An Itô stochastic differential equation has the form

$$dX_t = -\beta(t)\, X_t\, dt + \sqrt{\beta(t)}\, dW_t \tag{F}$$

for $t \in [0, T]$ and noise schedule $\beta(t) \geq 0$.

Fact: the marginal $X_t \sim p_t \to \mathcal{N}(0, I)$ as $t \to T$.

Intuition: the drift shrinks signal while Gaussian noise accumulates.

The reverse SDE maps noise to data

Definition: Let p_t for $t \in [0, T]$ denote the density of the forward process $X_t \sim p_t$.

Definition: The *reverse process* is the SDE

$$dX_t^R = \beta(t)\Big(X_t^R + \nabla_x \ln p_{T-t}\big(X_t^R\big)\Big)\, dt + \sqrt{\beta(t)}\, d\overline{W}_t, \tag{R}$$

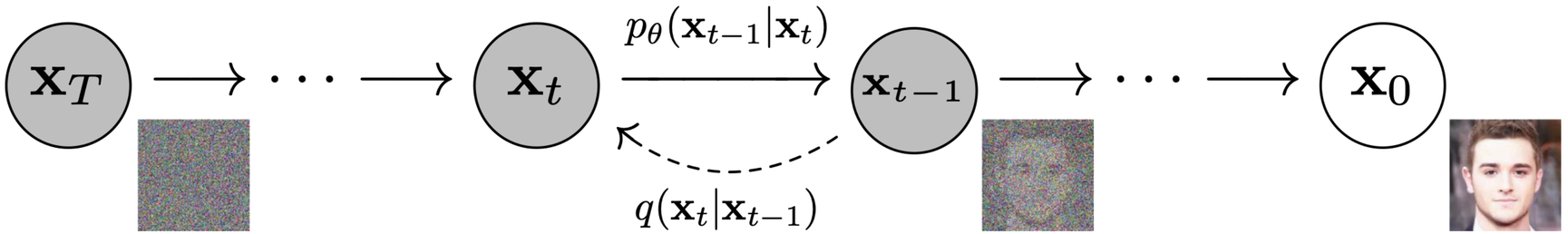
for $t \in [0, T]$

Theorem: [Law of (R), Haussmann & Pardoux (1986)] Given the forward equation (F) and its reverse equation (R), if we initialize (R) with $X_0^R \sim p_T$, then $X_t^R \sim p_{T-t}$ for all $t \in [0, T]$.

The DDPM algorithm

The DDPM algorithm generates new images by:

1. Discretize the forward process
2. Discretize the reverse process
3. Use the forward process to generate training dataset $\mathcal{D} = \{(\text{clean image, noisy image})\}$
4. Learn a denoiser ϵ_θ using the training dataset $\mathcal{D}$
5. Simulate the reverse process in terms of ϵ_θ to generate new images



Step 1: Discretize the forward process

To simulate

$$dX_t = -\beta(t)\, X_t\, dt + \sqrt{\beta(t)}\, dW_t \qquad \text{for noise schedule } \beta(t) \geq 0,$$

use the one-step recursive discretization

$$X_{t_k} = \sqrt{\alpha_k}\, X_{t_{k-1}} + \sqrt{1-\alpha_k}\, \epsilon_k \qquad \epsilon_k \sim \mathcal{N}(0, I)$$

for

$$\alpha_k := \exp\Big(- \int_{t_{k-1}}^{t_k} \beta(u)\, du \Big).$$

Step 1: Discretize the forward process

$$dX_t = -\beta(t)\, X_t\, dt + \sqrt{\beta(t)}\, dW_t,$$

$$X_{t_k} = \sqrt{\alpha_k}\, X_{t_{k-1}} + \sqrt{1-\alpha_k}\, \epsilon_k, \qquad \epsilon_k \sim \mathcal{N}(0, I).$$

Next, use the equivalent one-shot explicit form

$$X_{t_k} = \sqrt{\bar{\alpha}_k}\, X_{t_0} + \sqrt{1-\bar{\alpha}_k}\, \epsilon, \qquad \epsilon \sim \mathcal{N}(0, I)$$

for $\bar{\alpha}_k := \prod_{s=1}^{k} \alpha_s$. This lets us generate (X_{t_k}, t_k) from (X_{t_0}) in one shot.

Step 1: Discretize the forward process

$$dX_t = -\beta(t)\, X_t\, dt + \sqrt{\beta(t)}\, dW_t,$$

$$X_{t_k} = \sqrt{\alpha_k}\, X_{t_{k-1}} + \sqrt{1-\alpha_k}\, \epsilon_k, \qquad \epsilon_k \sim \mathcal{N}(0, I).$$

$$X_{t_k} = \sqrt{\bar{\alpha}_k}\, X_{t_0} + \sqrt{1-\bar{\alpha}_k}\, \epsilon, \qquad \epsilon \sim \mathcal{N}(0, I)$$

These discretizations are *exact*: $\forall t_k$

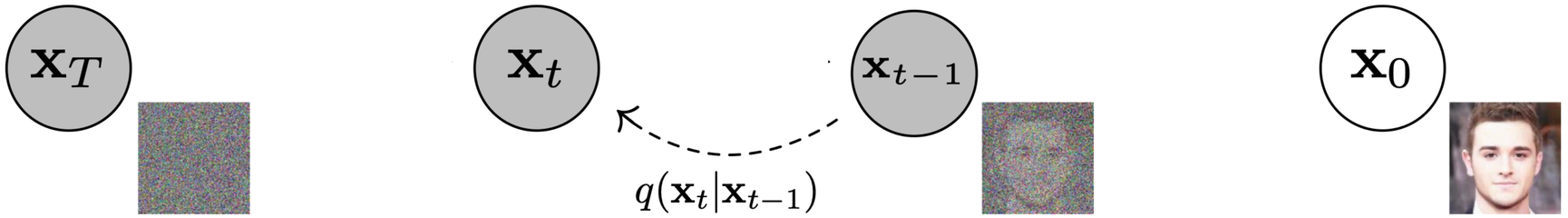
$$X_{t_k}^{\text{one shot}} \overset{d}{=} X_{t_k}^{\text{one step}} \overset{d}{=} X_{t_k}^{\text{SDE}},$$

and in particular

$$q(X_{t_k} \mid X_{t_0}) = \mathcal{N}\big(\sqrt{\bar{\alpha}_k}\, X_{t_0},\ (1-\bar{\alpha}_k) I\big).$$

Step 1: Discretize the forward process

$$X_{t_k} = \sqrt{\bar{\alpha}_k}\, X_{t_0} + \sqrt{1 - \bar{\alpha}_k}\, \epsilon, \qquad \epsilon \sim \mathcal{N}(0, I)$$



Step 2: Discretize the reverse process

Definition: The *reverse process* is the SDE

$$dX_t^R = \beta(t)\Big(X_t^R + \nabla_x \ln p_{T-t}\big(X_t^R\big)\Big)\, dt + \sqrt{\beta(t)}\, d\overline{W}_t, \tag{R}$$

for $t \in [0, T]$.

Because the forward chain is Gaussian, the true reverse conditional given the clean sample is also Gaussian:

$$q(X_{t-1} \mid X_t, X_0) = \mathcal{N}\big(\mu_t(X_t, X_0),\ \tilde{\beta}_t I\big),$$

where

$$\mu_t(X_t, X_0) = \frac{\sqrt{\bar{\alpha}_{t-1}}\,\beta_t}{1-\bar{\alpha}_t} X_0 + \frac{\sqrt{\alpha_t}\,(1-\bar{\alpha}_{t-1})}{1-\bar{\alpha}_t} X_t, \qquad \beta_t := 1-\alpha_t, \quad \tilde{\beta}_t := \frac{1-\bar{\alpha}_{t-1}}{1-\bar{\alpha}_t}\,\beta_t.$$

Step 2: Discretize the reverse process

$$q(X_{t-1} \mid X_t, X_0) = \mathcal{N}\big(\mu_t(X_t, X_0),\ \tilde{\beta}_t I\big),$$

This discretization is *exact* in the sense that initializing from $X_T \sim p_T$,

$$X_t \overset{d}{=} X^R_{T-t} \sim p_t \qquad \text{for all } t = 0, 1, \ldots, T.$$

DDMP uses noise prediction

$$q(X_{t-1} \mid X_t, X_0) = \mathcal{N}\big(\mu_t(X_t, X_0),\ \tilde{\beta}_t I\big),$$

$$\mu_t(X_t, X_0) = \frac{\sqrt{\bar{\alpha}_{t-1}}\,\beta_t}{1-\bar{\alpha}_t}\,X_0 + \frac{\sqrt{\alpha_t}\,(1-\bar{\alpha}_{t-1})}{1-\bar{\alpha}_t}\,X_t,$$

$$\beta_t := 1-\alpha_t, \quad \tilde{\beta}_t := \frac{1-\bar{\alpha}_{t-1}}{1-\bar{\alpha}_t}\,\beta_t.$$

But we do *not* know X_0.

DDPMs handle this by **predicting X_0 via noise prediction**.

Goal: figure out how to simulate the reverse process.

Step 3: Generate the training dataset

Algorithm: (training data generation)

1. Sample a clean image $X_0 \sim p_{\text{data}}$ (i.e., pick a random training example).
2. Sample a timestep $t \sim \text{Unif}\{1, \ldots, T\}$.
3. Sample noise $\epsilon \sim \mathcal{N}(0, I)$.
4. Form the noisy image in one shot:

$$X_t = \sqrt{\bar{\alpha}_t}\, X_0 + \sqrt{1 - \bar{\alpha}_t}\, \epsilon.$$

5. Add the training tuple to the dataset:

$$(X_0,\ X_t,\ \epsilon,\ t)$$

$$\mathcal{D} = \{(\text{clean image, noisy image, noise, time})\}$$

Step 4: Learn a denoising function $\epsilon_\theta(X_t, t)$

Train a neural network $\epsilon_\theta(X_t, t)$ to predict the noise ϵ in a given

$$X_t = \sqrt{\bar{\alpha}_t}\, X_0 + \sqrt{1 - \bar{\alpha}_t}\, \epsilon$$

from the dataset

$$\mathcal{D} = \{(X_0,\ X_t,\ \epsilon,\ t)\}.$$

Formally, use a neural net to solve

$$\min_\theta\ \mathbb{E}_{(X_0, X_t, \epsilon, t) \sim \text{Unif}\{\mathcal{D}\}} \left[\ \|\epsilon - \epsilon_\theta(X_t, t)\|_2^2\ \right]$$

Recall:

Step 2: Discretize the reverse process

Definition: The *reverse process* is the SDE

$$dX_t^R = \beta(t)\Big(X_t^R + \nabla_x \ln p_{T-t}\big(X_t^R\big)\Big)\, dt + \sqrt{\beta(t)}\, d\overline{W}_t, \tag{R}$$

for $t \in [0, T]$.

Because the forward chain is Gaussian, the true reverse conditional given the clean sample is also Gaussian:

$$q(X_{t-1} \mid X_t, X_0) = \mathcal{N}\big(\mu_t(X_t, X_0),\ \tilde{\beta}_t I\big),$$

where

$$\mu_t(X_t, X_0) = \frac{\sqrt{\bar{\alpha}_{t-1}}\,\beta_t}{1-\bar{\alpha}_t}\, X_0 + \frac{\sqrt{\alpha_t}\,(1-\bar{\alpha}_{t-1})}{1-\bar{\alpha}_t}\, X_t, \qquad \beta_t := 1-\alpha_t, \quad \tilde{\beta}_t := \frac{1-\bar{\alpha}_{t-1}}{1-\bar{\alpha}_t}\,\beta_t.$$

Step 5: Simulate the reverse process

Given ϵ_θ, define a point estimate of the clean sample:

$$\widehat{X_0}(X_t, t) = \frac{1}{\sqrt{\bar{\alpha}_t}}\Big(X_t - \sqrt{1-\bar{\alpha}_t}\,\epsilon_\theta(X_t, t)\Big). \tag{2}$$

Now instead of

$$q(X_{t-1} \mid X_t, X_0) = \mathcal{N}\big(\mu_t(X_t, X_0),\ \tilde{\beta}_t I\big),$$

we use

$$p_\theta(X_{t-1} \mid X_t, \widehat{X_0}) = \mathcal{N}\big(\mu_{t,\theta}(X_t, \widehat{X_0}),\ \tilde{\beta}_t I\big),$$

where the Gaussian reverse kernel is

$$\mu_{t,\theta}(X_t, \widehat{X_0}) = \frac{\sqrt{\bar{\alpha}_{t-1}}\,\beta_t}{1-\bar{\alpha}_t}\,\widehat{X_0} + \frac{\sqrt{\alpha_t}\,(1-\bar{\alpha}_{t-1})}{1-\bar{\alpha}_t}\,X_t, \qquad \beta_t := 1-\alpha_t, \quad \tilde{\beta}_t := \frac{1-\bar{\alpha}_{t-1}}{1-\bar{\alpha}_t}\,\beta_t. \tag{3}$$

Diffusion = noising + learned denoising

$$\widehat{X_0}(X_t, t) = \frac{1}{\sqrt{\bar{\alpha}_t}}\Big(X_t - \sqrt{1-\bar{\alpha}_t}\,\epsilon_\theta(X_t, t)\Big). \tag{2}$$

$$\mu_{t,\theta}(X_t, \widehat{X_0}) = \frac{\sqrt{\bar{\alpha}_{t-1}}\,\beta_t}{1-\bar{\alpha}_t}\,\widehat{X_0} + \frac{\sqrt{\alpha_t}\,(1-\bar{\alpha}_{t-1})}{1-\bar{\alpha}_t}\,X_t, \qquad \beta_t := 1-\alpha_t, \quad \tilde{\beta}_t := \frac{1-\bar{\alpha}_{t-1}}{1-\bar{\alpha}_t}\,\beta_t. \tag{3}$$

Algorithm (generation).

1. Sample $X_T \sim \mathcal{N}(0, I)$.
2. For $t = T, T-1, \ldots, 1$:
   (a) Compute $\hat{X}_0(X_t, t)$ using (2)
   (b) Compute $\mu_\theta(X_t, t)$ using (3).
   (c) Sample
   $$X_{t-1} = \mu_\theta(X_t, t) + \sqrt{\tilde{\beta}_t}\,z, \qquad z \sim \mathcal{N}(0, I)$$
3. Output X_0.

Neural network architecture for $\epsilon_\theta(X_t, t)$

- The model $\epsilon_\theta(x_t, t)$ is a denoiser conditioned on noise level.
- For images, the standard baseline is a **U-Net** with a timestep (often plus some attention).
- Transformers also work, especially at large scale and with heavy conditioning

Default recommendation: U-Net denoiser + time embedding.

Colab