Diffusion Models

Xingche Guo
Department of Statistics University of Connecticut · Fall 2026

Resources for this talk

My Generative Modeling course Github repo:
https://github.com/xingcheg/Statistical-Generative-Modeling

Key references

1. **DDPM.** Ho, Jain, and Abbeel (2020).
   Denoising Diffusion Probabilistic Models. *NeurIPS*.

2. **DDIM.** J. Song, Meng, and Ermon (2021).
   Denoising Diffusion Implicit Models. *ICLR*.

3. **SDE and probability-flow ODE.** Y. Song et al. (2021).
   Score-Based Generative Modeling through Stochastic Differential Equations. *ICLR*.

4. **Diffusion principles.** Lai et al. (2025).
   The Principles of Diffusion Models. arXiv:2510.21890.

5. **Textbook.** Murphy (2023).
   Probabilistic Machine Learning: Advanced Topics. MIT Press.

Evolution of image generation

Faces across eight milestones: from early latent-variable models to current image generators

2013 - 2026

Published examples | Updated 22 Oct 2026

2013 | VAE



Frey face | Latent-variable decoder

Kingma & Welling, Fig. 4a [1]

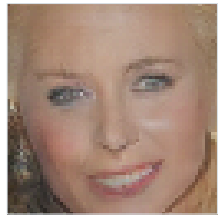
2014 | GAN



TFD face | Adversarial learning

Goodfellow et al., Fig. 2b [2]

2015 | DCGAN



Face sample | Convolutional generator

Radford et al., Fig. 10 [3]

2018 | StyleGAN



Synthetic portrait | Style control

Karras et al., official gallery [4]

2020 | DDPM



CelebA-HQ face | Iterative denoising

Ho et al., Fig. 1 [5]

2021 | Latent Diffusion



CelebA-HQ face | Denoising in latent space

Rombach et al., Fig. 4 [6]

2024 | Stable Diffusion 3.5



Text-guided portrait | Flow + transformers

Stability AI, release gallery [7]

2026 | FLUX 3 Image Klein



Reference-guided generation + precise editing

Black Forest Labs, release gallery [8]



Learning to generate images

An image is represented by $X \in \mathbb{R}^{H \times W \times 3}$. We imagine an unknown image distribution p_{data}.

1. Learning: fit a distribution

Given observed images $X^{(1)}, \ldots, X^{(n)} \overset{\text{i.i.d.}}{\sim} p_{\text{data}}$, estimate θ so that

$$p_{\widehat{\theta}}(x) \approx p_{\text{data}}(x).$$

2. Sampling: generate a new image

$$\text{Unconditional generation:} \qquad X_{\text{new}} \sim p_{\widehat{\theta}}(x),$$

$$\text{Conditional generation:} \qquad X_{\text{new}} \sim p_{\widehat{\theta}}(x \mid c).$$

For **text-to-image generation**, c is a text prompt, such as *"a cat sitting by a window."* For conditional diffusion generation, see classifier-free guidance (CFG) (Ho and Salimans, 2022).

This talk focuses on unconditional generation.

The diffusion idea

Learning a complex image distribution directly is difficult. **Instead, learn to undo noise one small step at a time.**

Forward (fixed): gradually add noise



Reverse (learned): recover structure step by step

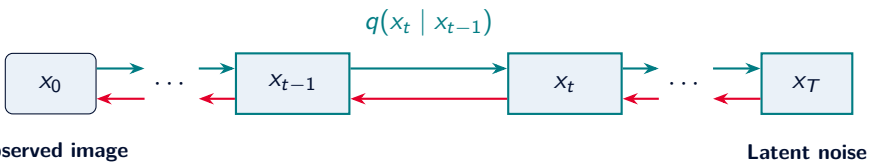
Learning: Corrupt real images at different noise levels and train a network to perform the corresponding denoising steps.

Generation: Start from **fresh Gaussian noise** and repeatedly apply the learned reverse steps to produce a **new image**.

The DDPM idea

Consider the forward diffusion process $x_0 \longrightarrow x_1 \longrightarrow \cdots \longrightarrow x_T$, where x_0 is the observed image and each step adds noise. Under the **Markov assumption**, the joint density factorizes as

$$q(x_{0:T}) = q_0(x_0) \prod_{t=1}^{T} q(x_t \mid x_{t-1}).$$



By Bayes' rule, we can show that **the exact reverse process is also Markovian.** The same joint distribution admits the reverse factorization:

$$q(x_{0:T}) = q_T(x_T) \prod_{t=1}^{T} q(x_{t-1} \mid x_t).$$

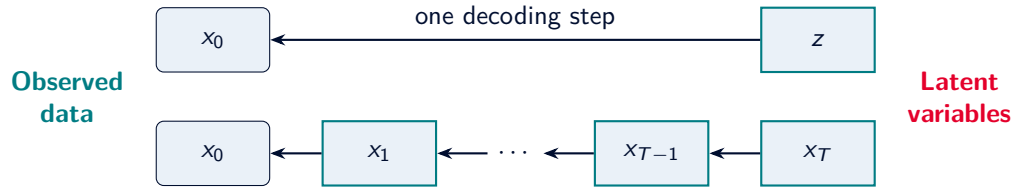
DDPM as a hierarchical VAE

VAE: one latent variable and one decoding step:

$$p_\theta(x_0, z) = p(z)\, p_\theta(x_0 \mid z).$$

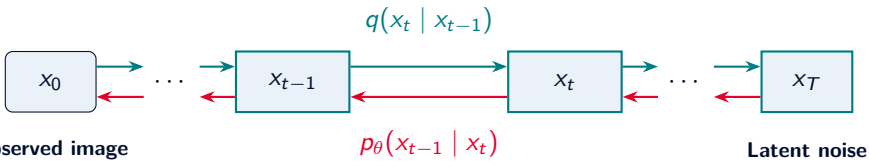
DDPM: a chain of latent states with learned reverse transitions:

$$p_\theta(x_{0:T}) = \underbrace{p(x_T)}_{\approx q_T(x_T)} \prod_{t=1}^{T} \underbrace{p_\theta(x_{t-1} \mid x_t)}_{\approx q(x_{t-1}|x_t)}.$$



Generation: Draw $x_T \sim \mathcal{N}(0, I_D)$, then generate an image x_0 through a sequence of learned denoising steps.

DDPM: learning to denoise



Forward process (fixed): Gradually add Gaussian noise using a schedule $\{\beta_t\}$.

$$q(x_t \mid x_{t-1}) = \mathcal{N}\Big(\sqrt{1-\beta_t}\, x_{t-1},\, \beta_t I_D\Big).$$

Reverse process (learning): Given a noisy state x_t, learn a distribution over plausible, less noisy states x_{t-1}:

$$\boxed{p_\theta(x_{t-1} \mid x_t) \approx q(x_{t-1} \mid x_t).}$$

Every state has the same dimension: $x_0, x_1, \ldots, x_T \in \mathbb{R}^D$.

The forward process

Choose a fixed noise schedule $0 < \beta_t < 1$ for $t = 1, \dots, T$, and let

$$\alpha_t = 1 - \beta_t.$$

Each step shrinks the current state and adds Gaussian noise:

$$\boxed{q(x_t \mid x_{t-1}) = \mathcal{N}(x_t; \sqrt{\alpha_t}\, x_{t-1}, \beta_t I_D)\,.}$$

Equivalently, generate the next state by

$$x_t = \underbrace{\sqrt{\alpha_t}\, x_{t-1}}_{\text{retain part of the signal}} + \underbrace{\sqrt{\beta_t}\, \epsilon_t}_{\text{add noise}}, \qquad \epsilon_t \overset{\text{iid}}{\sim} \mathcal{N}(0, I_D),$$

where ϵ_t is independent of the previous states.

The scaling preserves the standard Gaussian distribution:

$$x_{t-1} \sim \mathcal{N}(0, I_D) \quad \Longrightarrow \quad x_t \sim \mathcal{N}(0, (\alpha_t + \beta_t) I_D) = \mathcal{N}(0, I_D).$$

The forward marginal

Condition on a fixed image x_0, each x_t is a deterministic term plus a linear combination of independent Gaussian noises, so $q(x_t \mid x_0)$ is Gaussian:

$$\boxed{q(x_t \mid x_0) = \mathcal{N}\Big(x_t; \sqrt{\bar{\alpha}_t}\, x_0,\, (1 - \bar{\alpha}_t) I_D\Big)}$$

with $\bar{\alpha}_t = \prod_{s=1}^{t} \alpha_s$.
At any time step t, sample x_t directly from the clean image x_0:

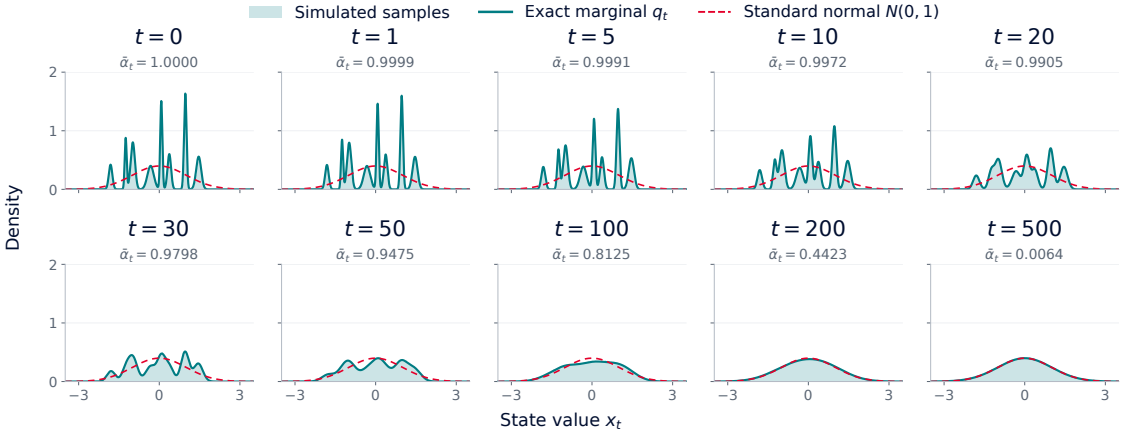
$$\boxed{x_t = \sqrt{\bar{\alpha}_t}\, x_0 + \sqrt{1 - \bar{\alpha}_t}\, \epsilon, \qquad \epsilon \sim \mathcal{N}(0, I_D).}$$

No intermediate states are needed.
As $\bar{\alpha}_t \to 0$, its mean vanishes and its covariance approaches I_D, so

$$q(x_t \mid x_0) \longrightarrow \mathcal{N}(0, I_D).$$

An one-dimensional example

The forward process smooths an irregular, eight-component Gaussian mixture.



Python simulation: 150,000 trajectories, $T = 500$, and β_t increases linearly from 10^{-4} to 0.02. Initially, $\mathbb{E}[x_0] = 0$ and $\mathrm{Var}(x_0) = 1$. All panels use the same axes.

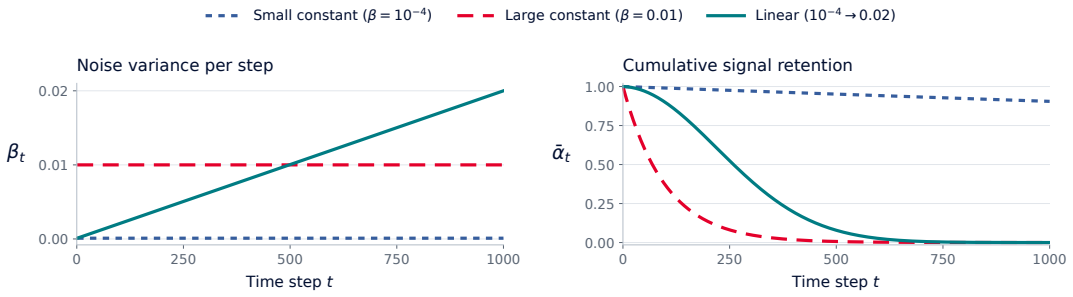
Choices of noise schedules

Constant: $\beta = 10^{-4}$ or 0.01.

$$\beta_t = \beta, \qquad \bar{\alpha}_t = (1-\beta)^t.$$

Linear: β_t increases with t.

$$\beta_t = \beta_{\text{min}} + \frac{t-1}{T-1}(\beta_{\text{max}} - \beta_{\text{min}}).$$



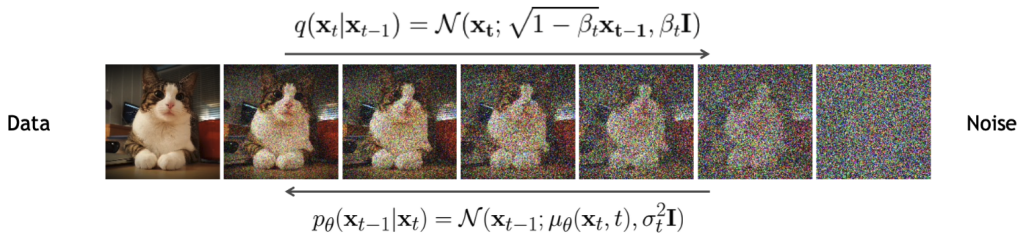
Small constant: small steps, but $\bar{\alpha}_{1000} \approx 0.905$; substantial signal remains.

Large constant: nearly pure noise by $T = 1000$, but the first noise SD is 10 times larger.

Linear: small initial steps and enough total noise: $\bar{\alpha}_{1000} \approx 4 \times 10^{-5}$.

Learning the reverse process

The forward transitions are known. Our goal is to **learn the reverse transitions**.



Define a **generative model** that starts from Gaussian noise:

$$p_\theta(x_{0:T}) = p(x_T) \prod_{t=1}^{T} p_\theta(x_{t-1} \mid x_t), \qquad p(x_T) = \mathcal{N}(0, I_D).$$

Here, x_0 is observed, while $x_{1:T}$ are **latent variables**.

How can we learn θ from observed images?

The Gaussian reverse process

Starting from Gaussian noise, define the **generative model**:

$$p_\theta(x_{0:T}) = p(x_T) \prod_{t=1}^{T} p_\theta(x_{t-1} \mid x_t), \qquad p(x_T) = \mathcal{N}(0, I_D).$$

For $t = T, \ldots, 2$, model each **reverse transition** as Gaussian:

$$p_\theta(x_{t-1} \mid x_t) = \mathcal{N}\Big(x_{t-1}; \mu_\theta(x_t, t), \sigma_t^2 I_D\Big).$$

- **Learned mean:** A shared neural network uses the noisy image x_t and time t to determine $\mu_\theta(x_t, t)$.
- **Fixed variance:** σ_t^2 follows a prespecified schedule and is not learned.

We will specify the mean parameterization and variance schedule later.

Variational inference for DDPM

Use the **fixed forward process** as the variational distribution for the latent path $x_{1:T}$:

$$q(x_{1:T} \mid x_0) = \prod_{t=1}^{T} q(x_t \mid x_{t-1}).$$

For an observed image x_0, lower-bound its log-likelihood:

$$\begin{aligned}
\log p_\theta(x_0) &= \log \int p_\theta(x_{0:T})\, dx_{1:T} \\
&= \log \mathbb{E}_{q(x_{1:T}|x_0)} \left[\frac{p_\theta(x_{0:T})}{q(x_{1:T} \mid x_0)} \right] \\
&\geq \underbrace{\mathbb{E}_{q(x_{1:T}|x_0)} \left[\log \frac{p_\theta(x_{0:T})}{q(x_{1:T} \mid x_0)} \right]}_{\text{ELBO}}.
\end{aligned}$$

Learning: maximize the ELBO over θ. The forward distribution q has **no trainable parameters**.

The three parts of the ELBO

Maximizing the ELBO is equivalent to **minimizing its negative**:

$$
\begin{aligned}
-\mathcal{L}_{\mathrm{ELBO}}(\theta; x_0) = & \underbrace{-\mathbb{E}_{q(x_1|x_0)}\left[\log p_\theta(x_0 \mid x_1)\right]}_{L_0(x_0)\text{: Reconstruction}} \\
& + \underbrace{\mathrm{KL}\left(q(x_T \mid x_0) \,\|\, p(x_T)\right)}_{L_T(x_0)\text{: Prior matching}} \\
& + \sum_{t=2}^{T} \underbrace{\mathbb{E}_{q(x_t|x_0)}\left[\mathrm{KL}\left(q(x_{t-1} \mid x_t, x_0) \,\|\, p_\theta(x_{t-1} \mid x_t)\right)\right]}_{L_{t-1}(x_0)\text{: Denoising matching}}.
\end{aligned}
$$

Learning: Improve reconstruction and reduce the denoising KL divergences. The prior-matching term is **constant in** θ.

Three reverse distributions

For $t \geq 2$, distinguish the **true reverse transition**, the **training target**, and the **learned model**.

Distribution	Role	Key property
$q(x_{t-1} \mid x_t)$	True reverse transition	Usually intractable; not generally Gaussian
$q(x_{t-1} \mid x_t, x_0)$	Training target given the clean image x_0	Exactly Gaussian, with known mean and variance
$p_\theta(x_{t-1} \mid x_t)$	Learned approximation to the true reverse transition	Modeled as Gaussian; used for generation

The network receives only x_t and t in both training and generation. During training, x_0 supplies the denoising target.

The training target

For $t \geq 2$, Bayes' rule gives

$$q(x_{t-1} \mid x_t, x_0) = \frac{q(x_t \mid x_{t-1})\, q(x_{t-1} \mid x_0)}{q(x_t \mid x_0)}.$$

Both factors in the numerator are Gaussian. This gives us a Gaussian distribution:

$$\boxed{q(x_{t-1} \mid x_t, x_0) = \mathcal{N}\Big(x_{t-1}; \widetilde{\mu}_t(x_t, x_0), \widetilde{\beta}_t I_D\Big), \quad t \geq 2.}$$

where

$$\tilde{\beta}_t = \frac{1}{a_t} = \frac{1 - \bar{\alpha}_{t-1}}{1 - \bar{\alpha}_t}\, \beta_t,$$

$$\tilde{\mu}_t(x_t, x_0) = \frac{b_t}{a_t} = \frac{\sqrt{\bar{\alpha}_{t-1}}\, \beta_t}{1 - \bar{\alpha}_t}\, x_0 + \frac{\sqrt{\alpha_t}(1 - \bar{\alpha}_{t-1})}{1 - \bar{\alpha}_t}\, x_t.$$

From Gaussian KL to squared-error loss

For $t \geq 2$, the target and learned reverse transition are Gaussian:

$$q(x_{t-1} \mid x_t, x_0) = \mathcal{N}\Big(x_{t-1}; \widetilde{\mu}_t(x_t, x_0), \widetilde{\beta}_t I_D\Big),$$

$$p_\theta(x_{t-1} \mid x_t) = \mathcal{N}\Big(x_{t-1}; \mu_\theta(x_t, t), \sigma_t^2 I_D\Big).$$

Suppressing the mean arguments, the Gaussian KL formula gives

$$\mathrm{KL}(q \,\|\, p_\theta) = \frac{D}{2}\left[\frac{\widetilde{\beta}_t}{\sigma_t^2} - 1 + \log\frac{\sigma_t^2}{\widetilde{\beta}_t}\right] + \frac{\|\widetilde{\mu}_t - \mu_\theta\|^2}{2\sigma_t^2}.$$

If we choose $\sigma_t^2 = \widetilde{\beta}_t$, the covariance terms cancel:

$$\boxed{\mathrm{KL}(q \,\|\, p_\theta) = \frac{1}{2\sigma_t^2}\,\|\widetilde{\mu}_t - \mu_\theta\|^2\,.}$$

Interpretation: Matching the reverse transitions reduces to a **weighted least squares problem** for the predicted mean.

From Gaussian KL to squared-error loss

The mean of $q(x_{t-1} \mid x_t, x_0)$ is a known linear combination of x_0 and x_t:

$$\widetilde{\mu}_t(x_t, x_0) = \frac{\sqrt{\bar{\alpha}_{t-1}}\,\beta_t}{1-\bar{\alpha}_t}\, x_0 + \frac{\sqrt{\alpha_t}(1-\bar{\alpha}_{t-1})}{1-\bar{\alpha}_t}\, x_t.$$

Use the **same form** for the learned mean, replacing x_0 with a **network prediction** $\widetilde{x}_\theta(x_t, t)$:

$$\mu_\theta(x_t, t) = \frac{\sqrt{\bar{\alpha}_{t-1}}\,\beta_t}{1-\bar{\alpha}_t}\, \widetilde{x}_\theta(x_t, t) + \frac{\sqrt{\alpha_t}(1-\bar{\alpha}_{t-1})}{1-\bar{\alpha}_t}\, x_t.$$

For $t \geq 2$ with $\sigma_t^2 = \widetilde{\beta}_t$, the common x_t term cancels when subtracting the means:

$$\begin{aligned} \mathrm{KL}(q \,\|\, p_\theta) &= \frac{1}{2\sigma_t^2} \left\| \widetilde{\mu}_t - \mu_\theta \right\|^2 \\ &= \frac{\bar{\alpha}_{t-1}\beta_t^2}{2\sigma_t^2(1-\bar{\alpha}_t)^2} \left\| x_0 - \widetilde{x}_\theta(x_t, t) \right\|^2 . \end{aligned}$$

From image prediction to noise prediction

We create a noisy training image by sampling $\epsilon \sim \mathcal{N}(0, I_D)$:

$$x_t = \underbrace{\sqrt{\bar{\alpha}_t}\, x_0}_{\text{signal}} + \underbrace{\sqrt{1-\bar{\alpha}_t}\, \epsilon}_{\text{noise}}.$$

If we knew the added noise, we could recover the clean image:

$$x_0 = \frac{x_t - \sqrt{1-\bar{\alpha}_t}\, \epsilon}{\sqrt{\bar{\alpha}_t}}.$$

DDPM learns to predict this noise. A neural network $\widetilde{\epsilon}_\theta(x_t, t)$ takes the noisy image and its noise level as inputs, giving

$$\widetilde{x}_\theta(x_t, t) = \frac{x_t - \sqrt{1-\bar{\alpha}_t}\, \widetilde{\epsilon}_\theta(x_t, t)}{\sqrt{\bar{\alpha}_t}}.$$

Thus, predicting the noise also provides an estimate of the clean image. At high noise levels, this estimate can be uncertain.

From image prediction to noise prediction

The two prediction errors differ only by a known scale:

$$\|x_0 - \widetilde{x}_\theta(x_t, t)\|^2 = \frac{1-\bar{\alpha}_t}{\bar{\alpha}_t} \|\epsilon - \widetilde{\epsilon}_\theta(x_t, t)\|^2 .$$

Thus, the clean-image loss weight w_t becomes

$$\lambda_t = w_t \frac{1-\bar{\alpha}_t}{\bar{\alpha}_t} = \frac{\beta_t^2}{2\sigma_t^2 \alpha_t (1-\bar{\alpha}_t)}.$$

The negative ELBO becomes a weighted noise-prediction loss:

$$-\mathcal{L}_{\mathrm{ELBO}}(\theta; x_0) = \sum_{t=1}^{T} \lambda_t \, \mathbb{E}_{\epsilon \sim \mathcal{N}(0, I_D)} \left[\|\epsilon - \widetilde{\epsilon}_\theta(x_t, t)\|^2 \right] + \text{Const.},$$

where $x_t = \sqrt{\bar{\alpha}_t}\, x_0 + \sqrt{1-\bar{\alpha}_t}\, \epsilon$.

Learning: Use one shared network to predict the sampled noise at every noise level. The input t tells the network how much corruption to expect.

Simplified training loss

The ELBO gives a weighted noise-prediction objective:

$$\mathcal{L}_{\text{weighted}}(\theta) = \frac{1}{T}\sum_{t=1}^{T} \lambda_t \, \mathbb{E}_{x_0,\epsilon}\left[\left\|\epsilon - \widetilde{\epsilon}_\theta(x_t, t)\right\|^2\right], \qquad x_t = \sqrt{\bar{\alpha}_t}\, x_0 + \sqrt{1-\bar{\alpha}_t}\, \epsilon.$$

1. Give all time steps equal weight. Set $\lambda_t = 1$. Empirically, this reweighting improved sample quality in the original DDPM experiments.

2. Express the average as an expectation. Let $t \sim \text{Uniform}\{1, \ldots, T\}$:

$$\boxed{\mathcal{L}_{\text{simple}}(\theta) = \mathbb{E}_{x_0,t,\epsilon}\left[\left\|\epsilon - \widetilde{\epsilon}_\theta(x_t, t)\right\|^2\right].}$$

Here $x_0 \sim q(x_0)$ and $\epsilon \sim \mathcal{N}(0, I_D)$, independently of t.

Practical benefit: Each training image needs only one randomly sampled time step to obtain an unbiased estimate of the loss.

DDPM Training

Repeat until convergence:

1. **Sample a minibatch of clean images:**

$$x_0^{(1)}, \ldots, x_0^{(B)} \quad \text{from the training data.}$$

2. **Sample a timestep and noise independently for each image:**

$$t_i \sim \text{Uniform}\{1, \ldots, T\}, \qquad \epsilon^{(i)} \sim \mathcal{N}(0, I_D), \quad i = 1, \ldots, B.$$

3. **Construct noisy images:**

$$x_{t_i}^{(i)} = \sqrt{\bar{\alpha}_{t_i}}\, x_0^{(i)} + \sqrt{1 - \bar{\alpha}_{t_i}}\, \epsilon^{(i)}.$$

4. **Take gradient descent step on:**

$$\nabla_\theta \frac{1}{B} \sum_{i=1}^{B} \left\| \epsilon^{(i)} - \epsilon_\theta\left(x_{t_i}^{(i)}, t_i\right) \right\|^2.$$

Key idea: One timestep per image. No need to simulate the entire forward chain.

DDPM sampling

1. **Start from pure Gaussian noise:** $x_T \sim \mathcal{N}(0, I_D)$.
2. **Apply the learned reverse steps.** For $t = T, T-1, \ldots, 1$, use $\widetilde{\epsilon}_\theta(x_t, t)$ to predict the noise in x_t and compute

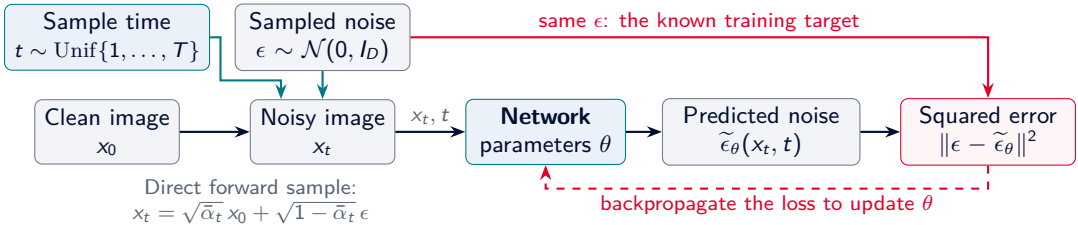
$$x_{t-1} = \underbrace{\frac{1}{\sqrt{\alpha_t}}\left(x_t - \frac{\beta_t}{\sqrt{1-\bar{\alpha}_t}}\widetilde{\epsilon}_\theta(x_t, t)\right)}_{\text{learned reverse mean } \mu_\theta(x_t, t)} + \underbrace{\sigma_t z_t}_{\text{sampling noise}},$$

   where $z_t \sim \mathcal{N}(0, I_D)$ for $t > 1$ and $z_1 = 0$.
3. **Return x_0 as the generated image.**

Intuition: The learned mean guides each denoising step. The added noise reflects uncertainty about the previous state.

Training and generation

Training: create a noisy input and a known target



Generation: build a new image through the reverse chain

Keep the trained θ fixed



Every step reuses $\widetilde{\epsilon}_\theta(x_t, t)$. Add fresh Gaussian noise for $t > 1$, and none at $t = 1$.

Try DDPM on the CelebA data

The CelebA data

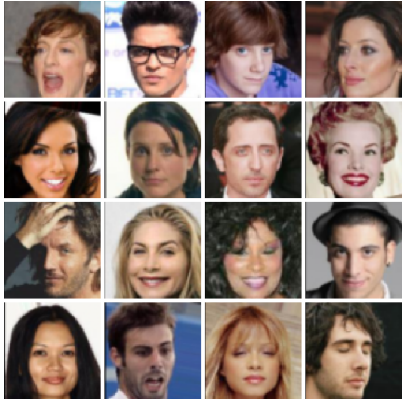
- CelebA contains **202,599 face images**.
- We uses **50,000 training**, 2,000 validation, and 2,000 test images.
- Crop to 148×148; resize to 64×64 RGB; normalize to $[-1, 1]$.

The task: unconditional generation

Learn the distribution of faces, then generate **new images from Gaussian noise**.

$$x_0 \in \mathbb{R}^{3\times 64\times 64}.$$

Real held-out CelebA examples

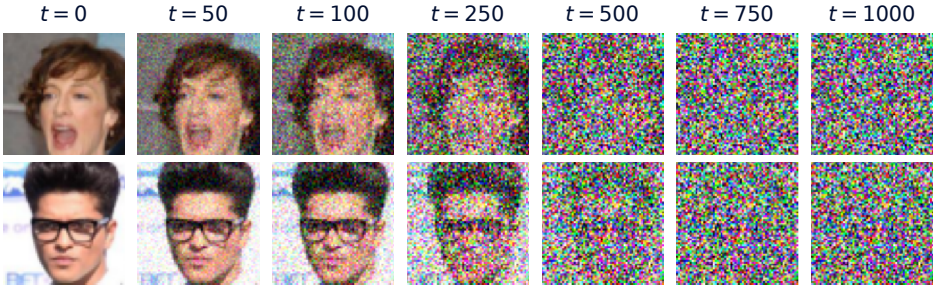


Data: Liu et al., CelebA (2015).

Forward process: image to noise

We choose $T = 1000$. For any chosen timestep, sample directly from the forward marginal:

$$x_t = \sqrt{\bar{\alpha}_t}\, x_0 + \sqrt{1 - \bar{\alpha}_t}\, \epsilon, \qquad \epsilon \sim \mathcal{N}(0, I).$$

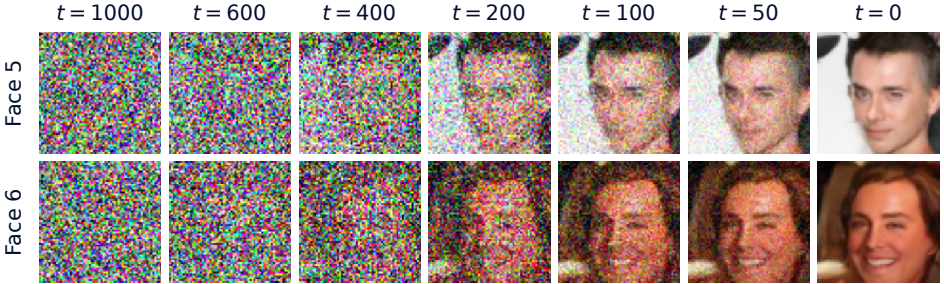


- A linear noise schedule is used: β_t increases linearly from 10^{-4} to 0.02.
- At $t = 1000$, $\sqrt{\bar{\alpha}_t} \approx 0.00635$: very little original signal remains.

Reverse process: noise to faces

Start with **fresh Gaussian noise**; use the trained U-Net to sample backward.

$$x_{1000} \sim \mathcal{N}(0, I) \quad \longrightarrow \quad x_{999} \quad \longrightarrow \quad \cdots \quad \longrightarrow \quad x_0.$$



- All 1,000 reverse steps are executed.
- Fresh noise is added for $t > 1$, and none at $t = 1$.

Generalizing DDPM

Training uses one noisy image at a sampled time.

$$x_t = \sqrt{\bar{\alpha}_t}\, x_0 + \sqrt{1-\bar{\alpha}_t}\, \epsilon, \qquad \epsilon \sim \mathcal{N}(0, I_D).$$

The network learns to predict the added noise.

$$\mathcal{L}_{\text{simple}}(\theta) = \mathbb{E}_{x_0,t,\epsilon}\left[\left\|\epsilon - \widetilde{\epsilon}_\theta(x_t, t)\right\|^2\right].$$

- Each training example uses a clean image and its noisy version.
- We never need to simulate the full path $x_1, \ldots, x_T$.

The training objective depends only on the marginals $q(x_t \mid x_0)$.

Can we keep these marginals but change the paths between them?

Same marginals, different paths

The DDPM joint admits a backward factorization:

$$q(x_{1:T} \mid x_0) = q(x_T \mid x_0) \prod_{t=2}^{T} q(x_{t-1} \mid x_t, x_0).$$

Generalize by choosing different conditional factors:

$$\boxed{q_\sigma(x_{1:T} \mid x_0) := q_\sigma(x_T \mid x_0) \prod_{t=2}^{T} q_\sigma(x_{t-1} \mid x_t, x_0).}$$

Preserve the same marginals. Keep the terminal distribution fixed:

$$q_\sigma(x_T \mid x_0) = q(x_T \mid x_0) = \mathcal{N}\left(\sqrt{\bar{\alpha}_T} x_0, (1 - \bar{\alpha}_T) I_D\right).$$

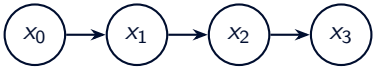
Choose the conditional factors so that

$$q_\sigma(x_t \mid x_0) = q(x_t \mid x_0), \qquad t = 1, \ldots, T-1.$$

The marginals can stay fixed while the dependence across time changes.

DDPM and generalized DDPM

DDPM

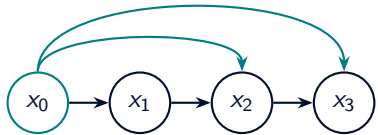


$q(x_t \mid x_{t-1})$

The next state depends only on the previous state.

Markovian

Generalized DDPM



$q_\sigma(x_t \mid x_{t-1}, x_0)$

The next state can also depend on the clean image.

Non-Markovian in general

Different dependence across time,

the same noisy-image distributions $q(x_t \mid x_0)$.

A conditional with adjustable randomness

Extract the noise already present in x_t:

$$\epsilon_t = \frac{x_t - \sqrt{\bar{\alpha}_t}\, x_0}{\sqrt{1-\bar{\alpha}_t}}.$$

A sample from $q_\sigma(x_{t-1} \mid x_t, x_0)$, for $t \geq 2$, is

$$x_{t-1} = \underbrace{\sqrt{\bar{\alpha}_{t-1}}\, x_0}_{\text{clean image}} + \underbrace{\sqrt{1-\bar{\alpha}_{t-1}-\tilde{\sigma}_t^2}\, \epsilon_t}_{\text{reused noise}} + \underbrace{\tilde{\sigma}_t\, \xi_t}_{\text{fresh noise}}, \quad \xi_t \sim \mathcal{N}(0, I_D).$$

The total noise level stays fixed as the split between reused and fresh noise changes.

It recovers DDPM when

$$\tilde{\sigma}_t^2 = \widetilde{\beta}_t = \frac{1-\bar{\alpha}_{t-1}}{1-\bar{\alpha}_t}\, \beta_t.$$

DDIM: a deterministic special case

Denoising Diffusion Implicit Models (DDIM) use zero sampling variance:

$$\boxed{\tilde{\sigma}_t^2 = 0, \qquad t = 2, \dots, T.}$$

For $t \geq 2$, $q_\sigma(x_{t-1} \mid x_t, x_0)$ **degenerates to a point mass** at

$$x_{t-1} = \sqrt{\bar{\alpha}_{t-1}}\, x_0 + \sqrt{1 - \bar{\alpha}_{t-1}}\, \frac{x_t - \sqrt{\bar{\alpha}_t}\, x_0}{\sqrt{1 - \bar{\alpha}_t}}.$$

Thus, given (x_t, x_0), the previous state x_{t-1} is determined.

The **terminal marginal remains unchanged**:

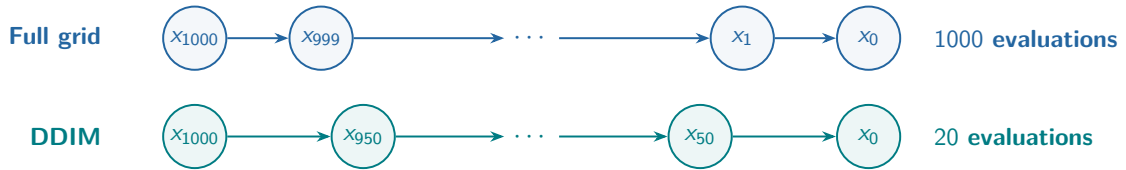
$$q_\sigma(x_T \mid x_0) = \mathcal{N}\Big(\sqrt{\bar{\alpha}_T}\, x_0,\, (1 - \bar{\alpha}_T) I_D\Big).$$

Given x_0, drawing x_T therefore determines the entire path $x_{T-1}, \dots, x_1$ through the **deterministic conditionals**.

Conditional on x_0, all randomness in the path comes from x_T.

Song, Meng & Ermon (2021), Denoising Diffusion Implicit Models, .

Accelerated sampling by DDIM



For DDIM (given x_0):
$$x_{t-1} = \sqrt{\bar{\alpha}_{t-1}}\, x_0 + \sqrt{1-\bar{\alpha}_{t-1}}\, \frac{x_t - \sqrt{\bar{\alpha}_t}\, x_0}{\sqrt{1-\bar{\alpha}_t}}.$$

Hence the standardized noise is unchanged along the path:
$$\epsilon = \frac{x_t - \sqrt{\bar{\alpha}_t}\, x_0}{\sqrt{1-\bar{\alpha}_t}} = \cdots = \frac{x_s - \sqrt{\bar{\alpha}_s}\, x_0}{\sqrt{1-\bar{\alpha}_s}}, \qquad 0 < s < t.$$

During sampling, estimate the unknowns at each selected time t:
$$\epsilon \leftarrow \widehat{\epsilon} = \widetilde{\epsilon}_\theta(x_t, t), \qquad x_0 \leftarrow \widehat{x}_0 = \frac{x_t - \sqrt{1-\bar{\alpha}_t}\, \widehat{\epsilon}}{\sqrt{\bar{\alpha}_t}}.$$

Substitute these estimates into the expression for x_s, then repeat at the next selected time.

DDIM: key advantages

Consistency across grids

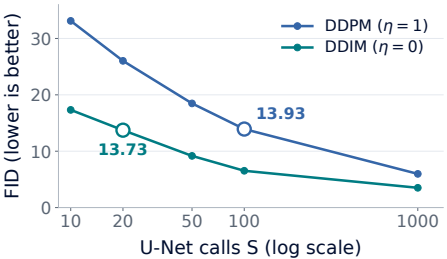
LSUN bedrooms



Same x_T in each column. Details change with S.

Quality with fewer steps

CelebA 64×64



Similar FID: DDIM 20 steps vs. DDPM 100.

- **Same training:** reuse the trained DDPM.
- **Deterministic:** fixed x_T and grid determine x_0.

Song, Meng & Ermon (2021), Denoising Diffusion Implicit Models, Left: crop of Fig. 5. Right: Table 1.

DDPM in continuous time

Rescale the DDPM step. For $t \in [0,1]$, use noise variance $\beta(t)\Delta t$:

$$X_{t+\Delta t} = \sqrt{1-\beta(t)\Delta t}\, X_t + \sqrt{\beta(t)\Delta t}\, \epsilon,$$

where $\epsilon \sim \mathcal{N}(0, I_D)$ is independent at each step.

By Taylor's expansion, $\sqrt{1-u} \approx 1 - u/2$,

$$X_{t+\Delta t} - X_t \approx -\frac{1}{2}\beta(t) X_t\, \Delta t + \sqrt{\beta(t)}\, \underbrace{\sqrt{\Delta t}\, \epsilon}_{\Delta W_t}.$$

As $\Delta t \to 0$: the variance-preserving (VP) SDE

$$\mathrm{d}X_t = \underbrace{-\frac{1}{2}\beta(t) X_t\, \mathrm{d}t}_{\text{signal decay}} + \underbrace{\sqrt{\beta(t)}\, \mathrm{d}W_t}_{\text{noise injection}}.$$

W_t is D-dimensional Brownian motion, with $\Delta W_t \sim \mathcal{N}(0, \Delta t\, I_D)$.

A general diffusion framework

Forward SDE: data to noise $(t : 0 \to 1)$. With a scalar noise scale $g(t)$,

$$\mathrm{d}X_t = \underbrace{f(X_t, t)}_{\text{drift}}\,\mathrm{d}t + \underbrace{g(t)}_{\text{noise scale}}\,\mathrm{d}W_t.$$

DDPM is a discretization of the VP choice:

$$f(x, t) = -\frac{1}{2}\beta(t)x, \qquad g(t) = \sqrt{\beta(t)}.$$

Reverse SDE: noise to data $(t : 1 \to 0)$. Let q_t denote the density of X_t.

$$\mathrm{d}X_t = \left[f(X_t, t) - g(t)^2 \underbrace{\nabla_x \log q_t(X_t)}_{\text{score}}\right]\mathrm{d}t + g(t)\,\mathrm{d}\overline{W}_t.$$

Here $\mathrm{d}t < 0$, and $\overline{W}_t$ is Brownian motion in reverse time.

Learn the score $s_\theta(x_t, t) \approx \nabla_x \log q_t(x_t)$, then simulate backward from noise.

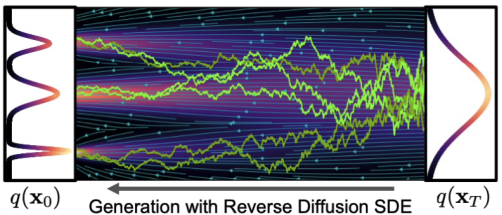
The probability-flow ODE

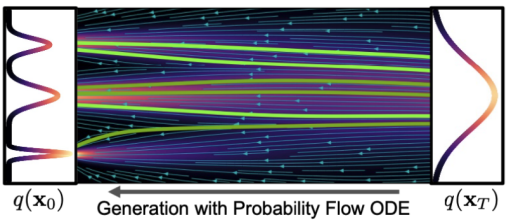
Idea: Move noise samples toward data using a deterministic velocity field.

$$\boxed{\frac{\mathrm{d}X_t}{\mathrm{d}t} = f(X_t, t) - \frac{1}{2} g(t)^2 \nabla_x \log q_t(X_t).}$$

For generation, integrate backward from $t = 1$ to $t = 0$.

Why it works: The score correction makes the ODE move probability mass in the same way as the SDE. With the true score and the same starting distribution, both have the same marginals q_t.





Thank You!