Diffusion Models: Theory

From a known noising process back to data — predict the noise, then reverse it

Prof. Nipun Batra

ES 667 · Deep Learning · IIT Gandhinagar

The generative problem becomes denoising

Why a new family of generative models?

Q QUESTION

We have met two ways to generate data:

Why a new family of generative models?

Q QUESTION

We have met two ways to generate data:

VAE (L19) — organize a latent space, decode a latent in **one** pass. Trained by a likelihood bound.

GAN (L20) — learn a generator against an **adversarial critic**. One pass, but unstable minimax training.

Why a new family of generative models?

Q QUESTION

We have met two ways to generate data:

VAE (L19) — organize a latent space, decode a latent in **one** pass. Trained by a likelihood bound.

GAN (L20) — learn a generator against an **adversarial critic**. One pass, but unstable minimax training.

Diffusion starts from a corruption process we design — then generates by iterating a learned reverse process

Why a new family of generative models?

Q QUESTION

We have met two ways to generate data:

VAE (L19) — organize a latent space, decode a latent in **one** pass. Trained by a likelihood bound.

GAN (L20) — learn a generator against an **adversarial critic**. One pass, but unstable minimax training.

Diffusion starts from a corruption process we design — then generates by iterating a learned reverse process

the key move: if we **know** how to add noise, we can train a network to **remove** it

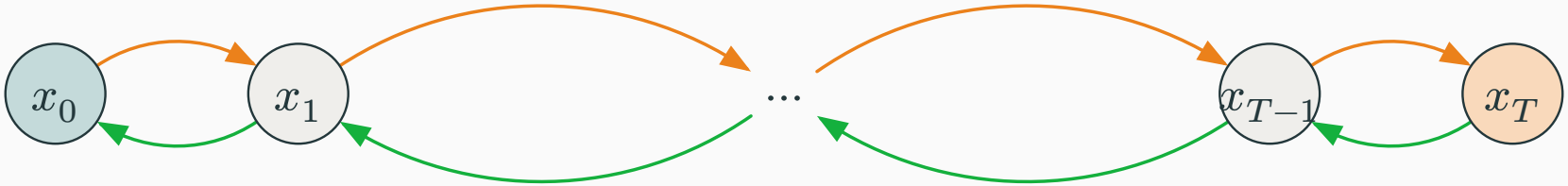
The one idea of this lecture

a diffusion model gradually turns **data** into **Gaussian noise**,
then learns to run that sequence **backwards**, one small denoising step at a time.

The one idea of this lecture

a diffusion model gradually turns **data** into **Gaussian noise**, then learns to run that sequence **backwards**, one small denoising step at a time.

forward (fixed): add noise $q(x_t \mid x_{t-1})$

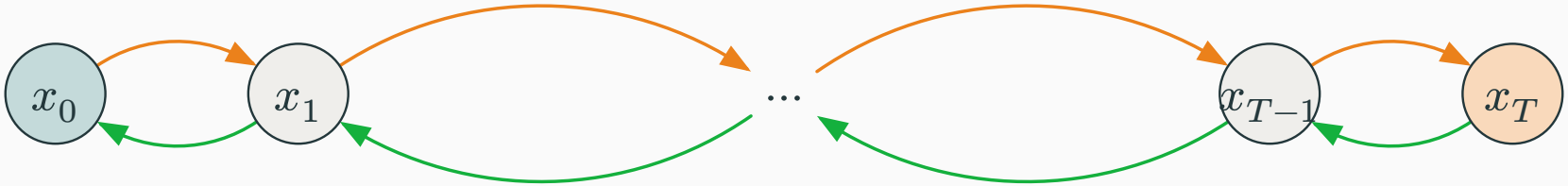


reverse (learned): denoise $p_{\theta(x_{t-1} \mid x_t)}$

The one idea of this lecture

a diffusion model gradually turns **data** into **Gaussian noise**, then learns to run that sequence **backwards**, one small denoising step at a time.

forward (fixed): add noise $q(x_t \mid x_{t-1})$



reverse (learned): denoise $p_\theta(x_{t-1} \mid x_t)$

forward is **fixed and known**; only the **reverse** transitions are **learned**

Visual hook: the noising filmstrip

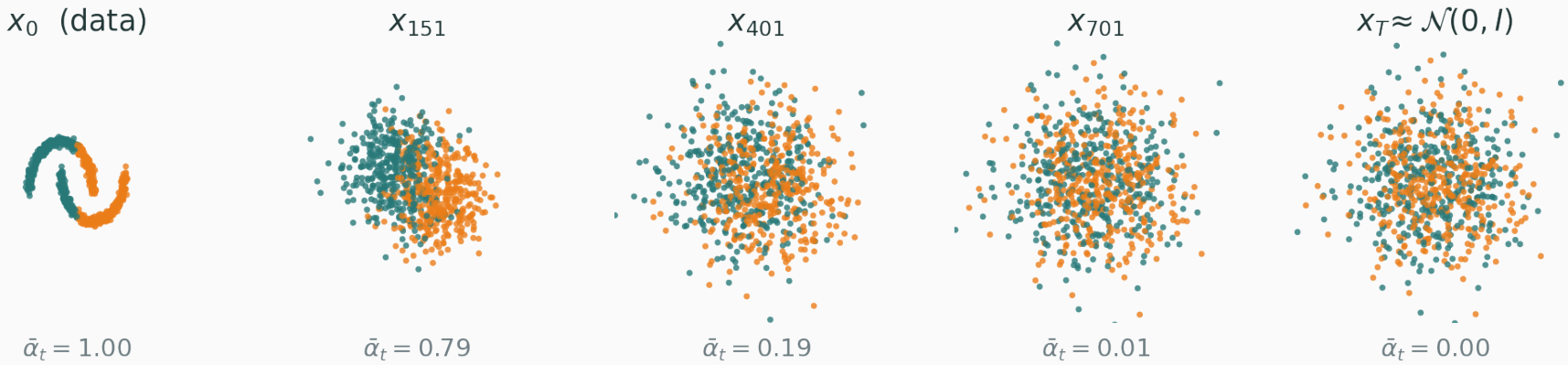
V VISUAL

Take a data point x_0 and add a little Gaussian noise, over and over:

Visual hook: the noising filmstrip

V VISUAL

Take a data point x_0 and add a little Gaussian noise, over and over:

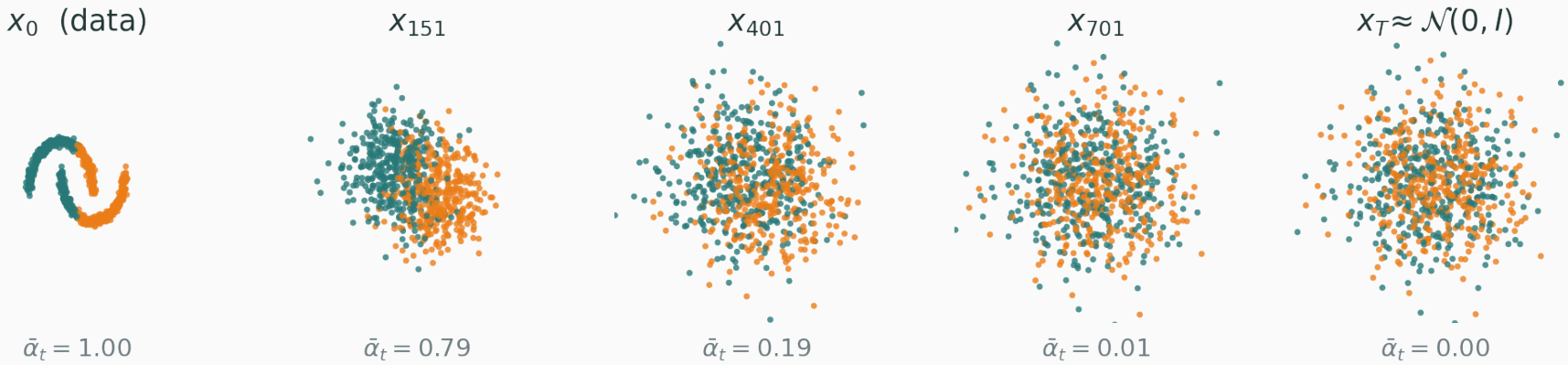


forward process $q(x_t \mid x_{t-1})$ — **add a little Gaussian noise each step** ⟶

Visual hook: the noising filmstrip

V VISUAL

Take a data point x_0 and add a little Gaussian noise, over and over:



forward process $q(x_t \mid x_{t-1})$ — **add a little Gaussian noise each step** ⟶

structure dissolves smoothly; by $t = T$ the sample is **indistinguishable from pure noise**

The forward process

D DERIVATION

Each forward step is a **fixed** Gaussian that shrinks the signal and injects noise:

The forward process

D DERIVATION

Each forward step is a **fixed** Gaussian that shrinks the signal and injects noise:

$$q(x_t \mid x_{t-1}) = \mathcal{N}\left(\sqrt{1-\beta_t}\, x_{t-1}, \beta_t I\right)$$

The forward process

D DERIVATION

Each forward step is a **fixed** Gaussian that shrinks the signal and injects noise:

$$q(x_t \mid x_{t-1}) = \mathcal{N}\left(\sqrt{1-\beta_t}\, x_{t-1}, \beta_t I\right)$$

- $\beta_t \in (0, 1)$ is a small **variance schedule** — how much noise this step adds;
- nothing here is learned: the forward chain is **chosen in advance**.

The forward process

D DERIVATION

Each forward step is a **fixed** Gaussian that shrinks the signal and injects noise:

$$q(x_t \mid x_{t-1}) = \mathcal{N}\left(\sqrt{1-\beta_t}\, x_{t-1}, \beta_t I\right)$$

- $\beta_t \in (0, 1)$ is a small **variance schedule** — how much noise this step adds;
- nothing here is learned: the forward chain is **chosen in advance**.

$\sqrt{1-\beta_t}$ **scales down** the previous sample; $\beta_t I$ **adds** fresh noise

A notation that makes the algebra clean

Define, for each step and cumulatively:

A notation that makes the algebra clean

D DERIVATION

Define, for each step and cumulatively:

$$\alpha_t = 1 - \beta_t, \qquad \overline{\alpha}_t = \prod_{s=1}^{t} \alpha_s$$

A notation that makes the algebra clean

D DERIVATION

Define, for each step and cumulatively:

$$\alpha_t = 1 - \beta_t, \qquad \overline{\alpha}_t = \prod_{s=1}^{t} \alpha_s$$

- α_t is the **fraction of variance kept** in one step;
- $\overline{\alpha}_t$ is the **fraction of the original signal kept** after t steps.

A notation that makes the algebra clean

D DERIVATION

Define, for each step and cumulatively:

$$\alpha_t = 1 - \beta_t, \qquad \overline{\alpha}_t = \prod_{s=1}^{t} \alpha_s$$

- α_t is the **fraction of variance kept** in one step;
- $\overline{\alpha}_t$ is the **fraction of the original signal kept** after t steps.

$\overline{\alpha}_t$ falls from ≈ 1 (early) toward ≈ 0 (late) — it is the whole story of the schedule

One-step intuition

D DERIVATION

Read the forward step as two operations:

One-step intuition

D DERIVATION

Read the forward step as two operations:

$$x_t = \underbrace{\sqrt{\alpha_t}\, x_{t-1}}_{\text{shrink the signal}} + \underbrace{\sqrt{\beta_t}\, \varepsilon}_{\text{add a little noise}}, \quad \varepsilon \sim \mathcal{N}(0, I)$$

One-step intuition

D DERIVATION

Read the forward step as two operations:

$$x_t = \underbrace{\sqrt{\alpha_t}\, x_{t-1}}_{\text{shrink the signal}} + \underbrace{\sqrt{\beta_t}\, \varepsilon}_{\text{add a little noise}}, \quad \varepsilon \sim \mathcal{N}(0, I)$$

each step barely changes the sample — many **tiny** steps accumulate into total corruption

The closed form that makes it all practical

D DERIVATION

Because Gaussians compose, we can jump straight to **any** noise level:

The closed form that makes it all practical

D DERIVATION

Because Gaussians compose, we can jump straight to **any** noise level:

$$q(x_t \mid x_0) = \mathcal{N}\left(\sqrt{\overline{\alpha}_t}\, x_0, (1 - \overline{\alpha}_t) I\right)$$

The closed form that makes it all practical

D DERIVATION

Because Gaussians compose, we can jump straight to **any** noise level:

$$q(x_t \mid x_0) = \mathcal{N}\left(\sqrt{\overline{\alpha}_t}\, x_0, (1 - \overline{\alpha}_t) I\right)$$

no need to simulate steps $1, 2, \ldots, t-1$ — one formula reaches x_t directly

The closed form that makes it all practical

D DERIVATION

Because Gaussians compose, we can jump straight to **any** noise level:

$$q(x_t \mid x_0) = \mathcal{N}\left(\sqrt{\overline{\alpha}_t}\, x_0, (1 - \overline{\alpha}_t) I\right)$$

no need to simulate steps $1, 2, \ldots, t-1$ — one formula reaches x_t directly

mean $\sqrt{\overline{\alpha}_t}\, x_0$ is the **surviving signal**; variance $(1 - \overline{\alpha}_t) I$ is the **accumulated noise**

Sampling a noisy training input

D DERIVATION

The closed form is a one-line recipe with the **reparameterization trick**:

Sampling a noisy training input

D DERIVATION

The closed form is a one-line recipe with the **reparameterization trick**:

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

Sampling a noisy training input

D DERIVATION

The closed form is a one-line recipe with the **reparameterization trick**:

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

- draw a clean x_0, draw a noise ε, pick a timestep t;
- the pair (x_t, ε) is a **ready-made supervised example** — input x_t, target ε.

Sampling a noisy training input

D DERIVATION

The closed form is a one-line recipe with the **reparameterization trick**:

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

- draw a clean x_0, draw a noise ε, pick a timestep t;
- the pair (x_t, ε) is a **ready-made supervised example** — input x_t, target ε.

> we **know the noise** because we drew it ourselves — this is the crux of the whole method

Board calculation: build x_t by hand

D DERIVATION

Let $x_0 = 1$, $\overline{\alpha}_t = 0.64$, and a drawn noise $\varepsilon = -0.5$.

Board calculation: build x_t by hand

D DERIVATION

Let $x_0 = 1$, $\overline{\alpha}_t = 0.64$, and a drawn noise $\varepsilon = -0.5$.

$$x_t = \sqrt{0.64}\,(1) + \sqrt{1 - 0.64}\,(-0.5)$$

Board calculation: build x_t by hand

D DERIVATION

Let $x_0 = 1$, $\overline{\alpha}_t = 0.64$, and a drawn noise $\varepsilon = -0.5$.

$$x_t = \sqrt{0.64}\,(1) + \sqrt{1 - 0.64}\,(-0.5)$$

$$x_t = \underbrace{0.8}_{\text{signal}} + \underbrace{0.6 \cdot (-0.5)}_{\text{noise}} = 0.8 - 0.3 = \mathbf{0.5}$$

Board calculation: build x_t by hand

D DERIVATION

Let $x_0 = 1$, $\overline{\alpha}_t = 0.64$, and a drawn noise $\varepsilon = -0.5$.

$$x_t = \sqrt{0.64}\,(1) + \sqrt{1 - 0.64}\,(-0.5)$$

$$x_t = \underbrace{0.8}_{\text{signal}} + \underbrace{0.6 \cdot (-0.5)}_{\text{noise}} = 0.8 - 0.3 = \mathbf{0.5}$$

signal $\sqrt{\overline{\alpha}_t}\,x_0 = 0.8$ — most of the clean value survives here

noise $\sqrt{1 - \overline{\alpha}_t}\,\varepsilon = -0.3$ — a moderate perturbation at this t

At the end of the chain

D DERIVATION

As $t \to T$, the product $\overline{\alpha}_t$ collapses toward zero:

At the end of the chain

D DERIVATION

As $t \to T$, the product $\overline{\alpha}_t$ collapses toward zero:

$$\overline{\alpha}_T \approx 0 \quad \Longrightarrow \quad q(x_T \mid x_0) \approx \mathcal{N}(0, I)$$

At the end of the chain

D DERIVATION

As $t \to T$, the product $\overline{\alpha}_t$ collapses toward zero:

$$\overline{\alpha}_T \approx 0 \quad \Longrightarrow \quad q(x_T \mid x_0) \approx \mathcal{N}(0, I)$$

x_T **is (almost) standard Gaussian noise — trivial to sample from**

At the end of the chain

D DERIVATION

As $t \to T$, the product $\overline{\alpha}_t$ collapses toward zero:

$$\overline{\alpha}_T \approx 0 \quad \Longrightarrow \quad q(x_T \mid x_0) \approx \mathcal{N}(0, I)$$

x_T is (almost) standard Gaussian noise — trivial to sample from

so generation can **start from pure noise** and only needs to learn the way **back**

Learning the reverse process

The direction that looks impossible

Q QUESTION

From one noisy image, **many** clean images are plausible:

The direction that looks impossible

Q QUESTION

From one noisy image, **many** clean images are plausible:

given a noisy x_t, which x_{t-1} did it come from?

The direction that looks impossible

Q QUESTION

From one noisy image, **many** clean images are plausible:

given a noisy x_t, which x_{t-1} did it come from?

The reverse of a random corruption is **not a function** — it is a **distribution**. There is genuine uncertainty about what was destroyed.

The direction that looks impossible

Q QUESTION

From one noisy image, **many** clean images are plausible:

given a noisy x_t, which x_{t-1} did it come from?

The reverse of a random corruption is **not a function** — it is a **distribution**. There is genuine uncertainty about what was destroyed.

so we model the reverse step **probabilistically**, and learn it from data

A parameterized reverse step

D DERIVATION

Model each reverse transition as a Gaussian whose mean (and variance) the network predicts:

A parameterized reverse step

D DERIVATION

Model each reverse transition as a Gaussian whose mean (and variance) the network predicts:

$$p_{\theta(x_{t-1} \mid x_t)} = \mathcal{N}\left(\mu_{\theta(x_t,t)}, \Sigma_{\theta(x_t,t)}\right)$$

A parameterized reverse step

D DERIVATION

Model each reverse transition as a Gaussian whose mean (and variance) the network predicts:

$$p_{\theta(x_{t-1} \mid x_t)} = \mathcal{N}\left(\mu_{\theta(x_t,t)}, \Sigma_{\theta(x_t,t)}\right)$$

- if steps are small, the **true** reverse step is **approximately Gaussian** too — so this form is well-motivated;
- all the difficulty is pushed into predicting the **mean** μ_θ.

A parameterized reverse step

D DERIVATION

Model each reverse transition as a Gaussian whose mean (and variance) the network predicts:

$$p_{\theta(x_{t-1} \mid x_t)} = \mathcal{N}\left(\mu_{\theta(x_t,t)}, \Sigma_{\theta(x_t,t)}\right)$$

- if steps are small, the **true** reverse step is **approximately Gaussian** too — so this form is well-motivated;
- all the difficulty is pushed into predicting the **mean** μ_θ.

the network must turn a slightly-noisier x_t into a slightly-cleaner x_{t-1}

The task we can actually supervise

D DERIVATION

During training we **know both** x_0 and the noise ε we injected. So flip the problem:

The task we can actually supervise

D DERIVATION

The task we can actually supervise

D DERIVATION

During training we **know both** x_0 and the noise ε we injected. So flip the problem:

don’t predict x_{t-1} directly — predict the noise ε that was added

The task we can actually supervise

D DERIVATION

The task we can actually supervise

D DERIVATION

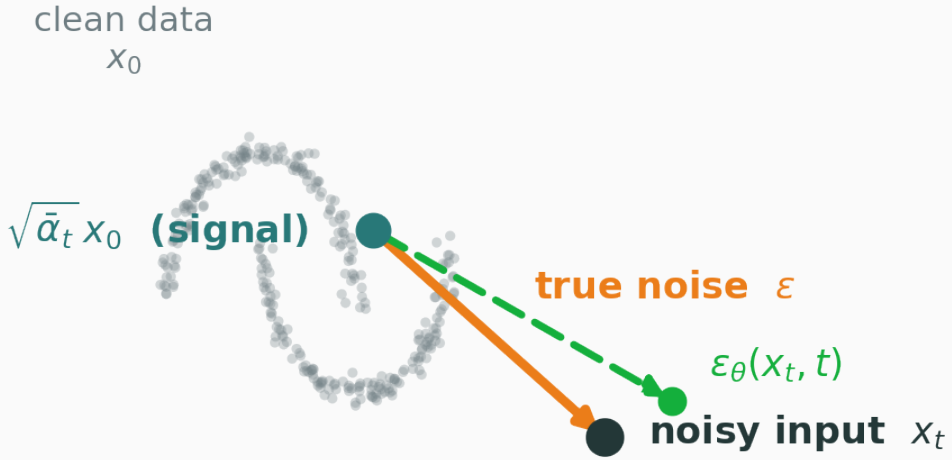
During training we **know both** x_0 and the noise ε we injected. So flip the problem:

don’t predict x_{t-1} directly — predict the noise ε that was added

The task we can actually supervise

D DERIVATION

the supervised target is the injected noise; train $\varepsilon_\theta(x_t, t) \approx \varepsilon$



The noise-prediction network

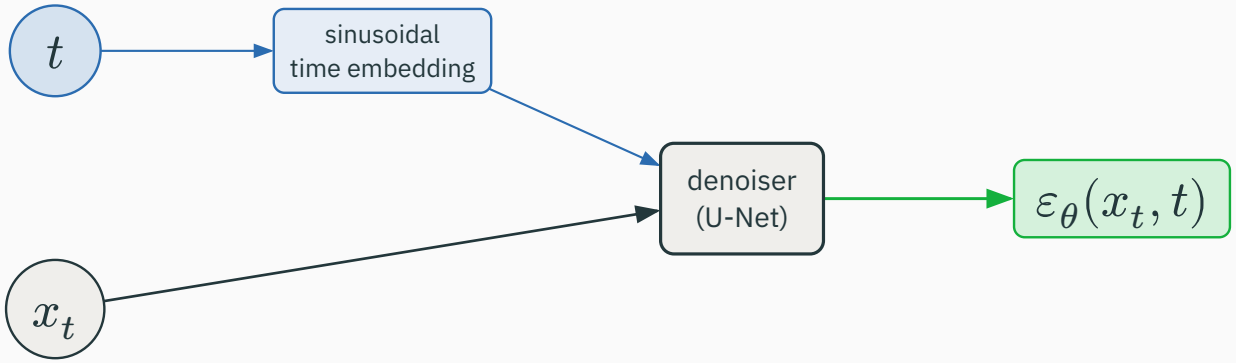
V VISUAL

A single network $\varepsilon_{\theta(x_t,t)}$ estimates the injected noise from the noisy input and the timestep:

The noise-prediction network

V VISUAL

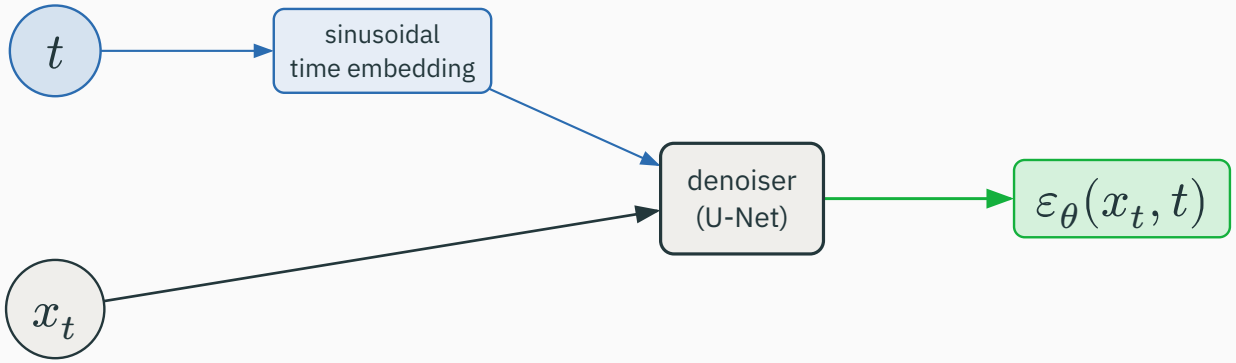
A single network $\varepsilon_{\theta(x_t,t)}$ estimates the injected noise from the noisy input and the timestep:



The noise-prediction network

V VISUAL

A single network $\varepsilon_{\theta(x_t,t)}$ estimates the injected noise from the noisy input and the timestep:



it **must** see t: “remove a little noise” and “remove almost all noise” are different jobs

The practical DDPM objective

D DERIVATION

Train by simple regression of predicted noise onto true noise:

The practical DDPM objective

D DERIVATION

Train by simple regression of predicted noise onto true noise:

$$L_{\text{simple}} = \mathbb{E}_{x_0,\, \varepsilon,\, t}\left[\left\| \varepsilon - \varepsilon_{\theta(x_t, t)} \right\|_2^2 \right]$$

The practical DDPM objective

D DERIVATION

Train by simple regression of predicted noise onto true noise:

$$L_{\text{simple}} = \mathbb{E}_{x_0,\,\varepsilon,\,t}\left[\left\|\varepsilon - \varepsilon_{\theta(x_t,t)}\right\|_2^2\right]$$

- expectation over data x_0, injected noise ε, and timestep t;
- this is **supervised regression** with **self-generated** targets — no labels, no adversary.

The practical DDPM objective

D DERIVATION

Train by simple regression of predicted noise onto true noise:

$$L_{\text{simple}} = \mathbb{E}_{x_0,\,\varepsilon,\,t}\left[\,\left\|\varepsilon - \varepsilon_{\theta(x_t,t)}\right\|_2^2\,\right]$$

- expectation over data x_0, injected noise ε, and timestep t;
- this is **supervised regression** with **self-generated** targets — no labels, no adversary.

the whole of DDPM training is: **sample noise, predict it, minimize the MSE**

A useful reframe: self-supervised denoising

D DERIVATION

Every clean example is an **infinite** source of training pairs:

A useful reframe: self-supervised denoising

D DERIVATION

Every clean example is an **infinite** source of training pairs:

one clean x_0 $\rightarrow$ pick any t, draw any ε $\rightarrow$ a fresh noisy input x_t.

exact target — the noise ε is **known by construction**, so the label is free and precise.

A useful reframe: self-supervised denoising

D DERIVATION

Every clean example is an **infinite** source of training pairs:

one clean x_0 $\rightarrow$ pick any t, draw any ε $\rightarrow$ a fresh noisy input x_t.

exact target — the noise ε is **known by construction**, so the label is free and precise.

diffusion is denoising self-supervision at **every** noise level at once

From predicted noise to a clean estimate

D DERIVATION

Invert the closed form: if we can guess ε, we can guess the clean signal.

From predicted noise to a clean estimate

D DERIVATION

Invert the closed form: if we can guess ε, we can guess the clean signal.

$$\hat{x}_0 = \frac{x_t - \sqrt{1-\overline{\alpha}_t}\,\varepsilon_{\theta(x_t,t)}}{\sqrt{\overline{\alpha}_t}}$$

From predicted noise to a clean estimate

D DERIVATION

Invert the closed form: if we can guess ε, we can guess the clean signal.

$$\hat{x}_0 = \frac{x_t - \sqrt{1-\overline{\alpha}_t}\,\varepsilon_{\theta(x_t,t)}}{\sqrt{\overline{\alpha}_t}}$$

predicting the noise and predicting the clean image are **two views of the same quantity**

The reverse sampling loop

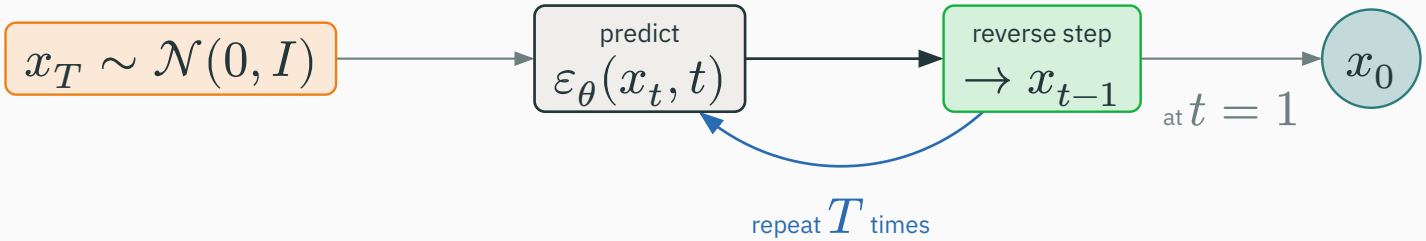
V VISUAL

Start at pure noise; peel off a little noise at a time until a sample emerges:

The reverse sampling loop

V VISUAL

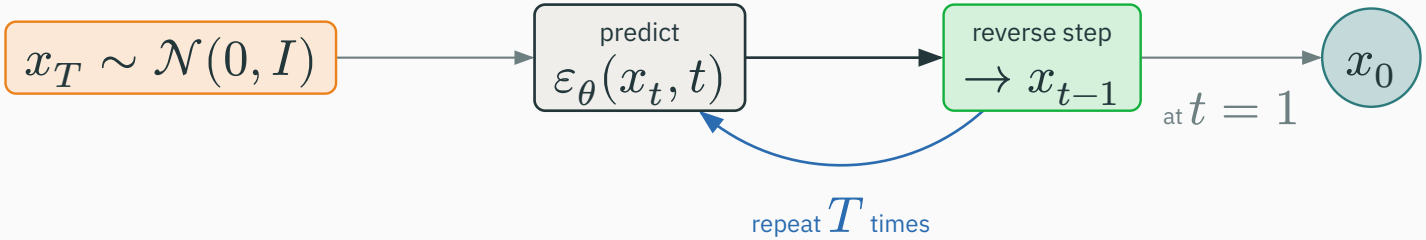
Start at pure noise; peel off a little noise at a time until a sample emerges:



The reverse sampling loop

V VISUAL

Start at pure noise; peel off a little noise at a time until a sample emerges:




```
x_T ~ N(0, I)                          # start from pure Gaussian noise
for t = T, ..., 1:
    eps_hat = eps_theta(x_t, t)        # predict the noise
    x_{t-1} = reverse_step(x_t, eps_hat, t)   # sample / estimate x_{t-1}
return x_0
```

The reverse step, written out

D DERIVATION

`reverse_step` is one Gaussian sample: a mean built from the predicted noise, plus a little fresh noise.

The reverse step, written out

D DERIVATION

`reverse_step` is one Gaussian sample: a mean built from the predicted noise, plus a little fresh noise.

$$x_{t-1} = \frac{1}{\sqrt{\alpha_t}}\left(x_t - \frac{1-\alpha_t}{\sqrt{1-\bar{\alpha}_t}}\,\varepsilon_{\theta(x_t,t)}\right) + \sigma_t\, z, \qquad z \sim \mathcal{N}(0, I)$$

The reverse step, written out

D DERIVATION

`reverse_step` is one Gaussian sample: a mean built from the predicted noise, plus a little fresh noise.

$$x_{t-1} = \frac{1}{\sqrt{\alpha_t}}\left(x_t - \frac{1-\alpha_t}{\sqrt{1-\bar{\alpha}_t}}\,\varepsilon_{\theta(x_t,t)}\right) + \sigma_t\, z, \qquad z \sim \mathcal{N}(0, I)$$

The **mean** is just $\hat{x}_0$'s information re-expressed as a small step toward $t-1$; $\sigma_t^2 = \beta_t$ (or the exact posterior variance) sets how much fresh noise to re-inject. Setting $z = 0$ gives the deterministic (DDIM-style) step.

Interactive: forward and reverse

I INTERACTIVE

INTERACTIVE

↗ nipunbatra.github.io/interactive-articles/diffusion-denoise

Slide the noise level to watch $x_t = \sqrt{\bar{\alpha}_t}\, x_0 + \sqrt{1-\bar{\alpha}_t}\, \varepsilon$ corrupt an example, then run the reverse chain step-by-step and see a sample condense out of noise.

Why generation is slow

D DERIVATION

The cost of quality is **many** forward passes:

Why generation is slow

D DERIVATION

The cost of quality is **many** forward passes:

GAN — one z, one generator pass, one image. Fast.

Diffusion — the **same** denoiser is called **hundreds to thousands** of times per sample.

Why generation is slow

D DERIVATION

The cost of quality is **many** forward passes:

GAN — one z, one generator pass, one image. Fast.

Diffusion — the **same** denoiser is called **hundreds to thousands** of times per sample.

stable training and broad coverage — paid for in **inference** time (L22 attacks this)

Interpreting the geometry

Denoising happens at many scales

D DERIVATION

Different reverse steps do qualitatively different work:

Denoising happens at many scales

D DERIVATION

Different reverse steps do qualitatively different work:

- **early** reverse steps (near pure noise) — recover **global layout** and coarse structure;
- **late** reverse steps (nearly clean) — restore **fine texture** and detail.

Denoising happens at many scales

D DERIVATION

Different reverse steps do qualitatively different work:

- **early** reverse steps (near pure noise) — recover **global layout** and coarse structure;
- **late** reverse steps (nearly clean) — restore **fine texture** and detail.

one network, but the **job changes** along the trajectory — hence the time conditioning

The noise schedule is a curriculum

D DERIVATION

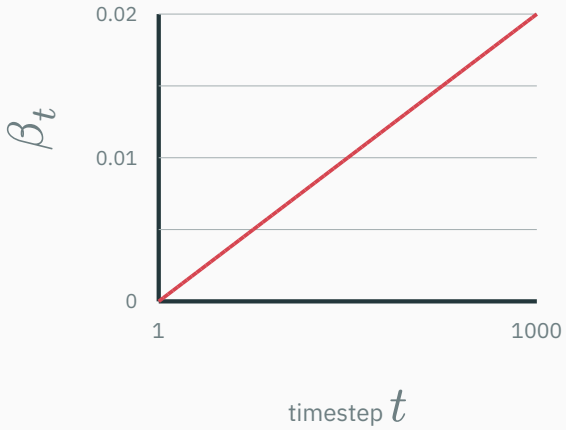
Training touches **every** difficulty, from trivial to nearly hopeless:

The noise schedule is a curriculum

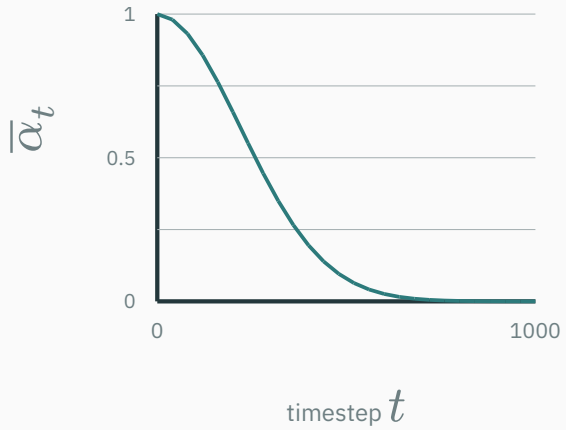
D DERIVATION

Training touches **every** difficulty, from trivial to nearly hopeless:

β_t: $10^{-4} \rightarrow 0.02$ **(linear)**



$\overline{\alpha}_t$: **signal retained**

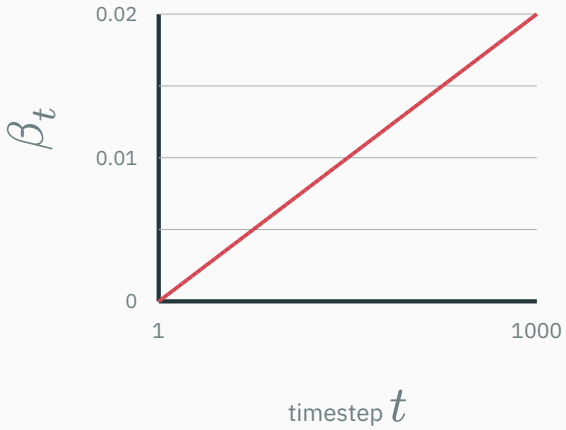


The noise schedule is a curriculum

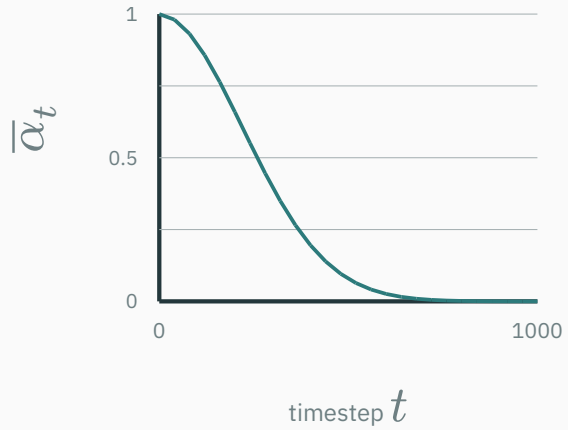
D DERIVATION

Training touches **every** difficulty, from trivial to nearly hopeless:

β_t: $10^{-4} \rightarrow 0.02$ **(linear)**



$\overline{\alpha}_t$**: signal retained**



β_t (how much noise each step adds) sets $\overline{\alpha}_t$ (how much signal survives)

The score connection — one conceptual slide

D DERIVATION

For a noisy distribution p_t, the **score** is the gradient of its log-density:

The score connection — one conceptual slide

D DERIVATION

For a noisy distribution p_t, the **score** is the gradient of its log-density:

$\nabla_x \log p_t(x)$ (points toward **higher-density** regions)

The score connection — one conceptual slide

D DERIVATION

For a noisy distribution p_t, the **score** is the gradient of its log-density:

$\nabla_x \log p_t(x)$ (points toward **higher-density** regions)

Predicting Gaussian noise is **equivalent** (up to a schedule factor) to estimating this score:

$$\varepsilon_{\theta(x_t,t)} \approx -\sqrt{1-\overline{\alpha}_t}\,\nabla_x \log p_t(x_t).$$

The score connection — one conceptual slide

D DERIVATION

For a noisy distribution p_t, the **score** is the gradient of its log-density:

$\nabla_x \log p_t(x)$ (points toward **higher-density** regions)

Predicting Gaussian noise is **equivalent** (up to a schedule factor) to estimating this score:

$$\varepsilon_{\theta(x_t,t)} \approx -\sqrt{1-\overline{\alpha}_t}\,\nabla_x \log p_t(x_t).$$

a bridge to **score-based** models — we stop here, no SDE derivation

What the network actually learns

Q QUESTION

Not a lookup table of clean images:

What the network actually learns

Q QUESTION

Not a lookup table of clean images:

it learns, from any noisy point, a local direction toward where the data density is high

What the network actually learns

Q QUESTION

Not a lookup table of clean images:

it learns, from any noisy point, a local direction toward where the data density is high

a **vector field of denoising directions** — generalized across inputs, timesteps, and conditions

Conditional generation, previewed

D DERIVATION

Feed the denoiser extra information c so the trajectory follows a desired target:

Conditional generation, previewed

D DERIVATION

Feed the denoiser extra information c so the trajectory follows a desired target:

- a **class label**; a **text embedding**; a **segmentation mask**; a **low-resolution image**;

Conditional generation, previewed

D DERIVATION

Feed the denoiser extra information c so the trajectory follows a desired target:

- a **class label**; a **text embedding**; a **segmentation mask**; a **low-resolution image**;

$\varepsilon_{\theta(x_t,t,c)}$ (same network, now told what to aim for)

Conditional generation, previewed

D DERIVATION

Feed the denoiser extra information c so the trajectory follows a desired target:

- a **class label**; a **text embedding**; a **segmentation mask**; a **low-resolution image**;

$\varepsilon_{\theta(x_t,t,c)}$ (same network, now told what to aim for)

the denoising **question** is unchanged — only the **conditioning** differs (full story in L22)

Three generative families so far

V VISUAL

Family	Training signal	Sampling
VAE	likelihood (ELBO) bound	one latent decode
GAN	adversarial critic	one generator pass
Diffusion	predict **known** injected noise	repeated denoising

Three generative families so far

V VISUAL

Family	Training signal	Sampling
VAE	likelihood (ELBO) bound	one latent decode
GAN	adversarial critic	one generator pass
Diffusion	predict **known** injected noise	repeated denoising

diffusion trades a **hard sampler** for an **easy, stable** learning signal

What diffusion gains and what it pays

D DERIVATION

What diffusion gains and what it pays

D DERIVATION

gains — stable supervised-style training; broad mode coverage; flexible conditioning.

pays — many inference steps; substantial compute per sample.

What diffusion gains and what it pays

D DERIVATION

gains — stable supervised-style training; broad mode coverage; flexible conditioning.

pays — many inference steps; substantial compute per sample.

the central trade of the method: **training ease** against **sampling cost**

Knowledge check

Q QUESTION

If t is **large** (a very noisy input), should the model rely more on the **noisy input** or on its **learned data prior**? Why?

Knowledge check

Q QUESTION

If t is **large** (a very noisy input), should the model rely more on the **noisy input** or on its **learned data prior**? Why?

At large t, x_t carries **almost no signal** ($\overline{\alpha}_t \approx 0$). The denoiser must lean on the **prior** it learned — the input mostly tells it **which** broad direction, not the fine details.

Closing bridge to practice

D DERIVATION

Theory says: **predict the noise, repeatedly**.

Closing bridge to practice

Theory says: **predict the noise, repeatedly**.

Practice then asks three questions:

Closing bridge to practice

D DERIVATION

Theory says: **predict the noise, repeatedly**.

Practice then asks three questions:

1. **what architecture** predicts ε_θ? (U-Net / diffusion Transformer)
2. **how does text steer it**? (conditioning + guidance)
3. **can we avoid thousands of steps**? (fast samplers, latent space)

Closing bridge to practice

D DERIVATION

Theory says: **predict the noise, repeatedly**.

Practice then asks three questions:

1. **what architecture** predicts ε_θ? (U-Net / diffusion Transformer)
2. **how does text steer it**? (conditioning + guidance)
3. **can we avoid thousands of steps**? (fast samplers, latent space)

L22 answers all three

Diffusion theory in depth

The forward chain factorizes

D DERIVATION

The forward process is a **Markov chain** — each step depends only on the last:

The forward chain factorizes

D DERIVATION

The forward process is a **Markov chain** — each step depends only on the last:

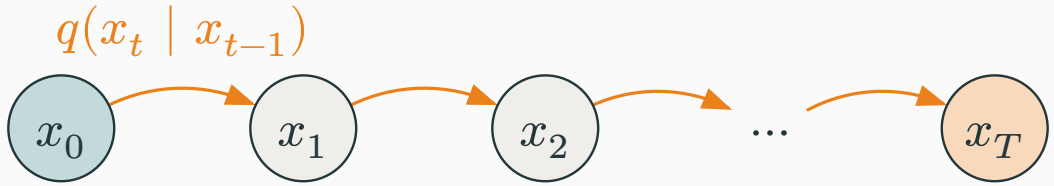
$$q(x_{1:T} \mid x_0) = \prod_{t=1}^{T} q(x_t \mid x_{t-1})$$

The forward chain factorizes

D DERIVATION

The forward process is a **Markov chain** — each step depends only on the last:

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

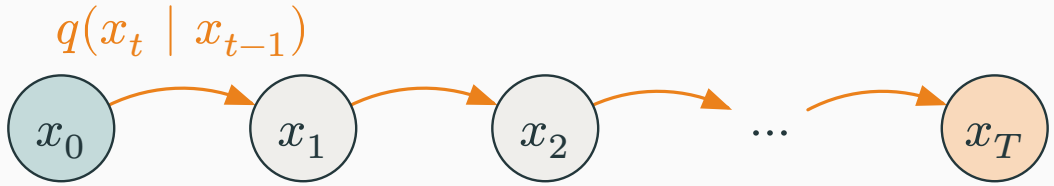


The forward chain factorizes

D DERIVATION

The forward process is a **Markov chain** — each step depends only on the last:

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



a simple, fixed structure — even when the data distribution is wildly complicated

The reverse model factorizes too

D DERIVATION

Sampling runs the learned chain from the easy prior downward:

The reverse model factorizes too

D DERIVATION

Sampling runs the learned chain from the easy prior downward:

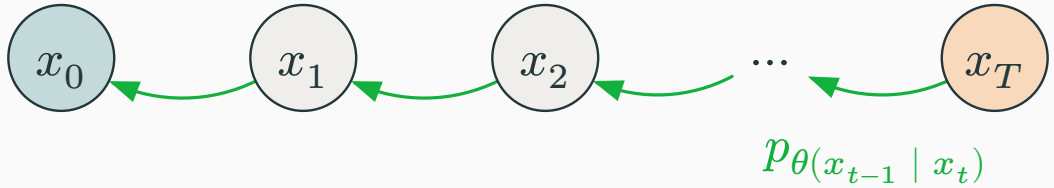
$$p_{\theta(x_{0:T})} = p(x_T) \prod_{t=1}^{T} p_{\theta(x_{t-1} \mid x_t)}, \qquad p(x_T) = \mathcal{N}(0, I)$$

The reverse model factorizes too

D DERIVATION

Sampling runs the learned chain from the easy prior downward:

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

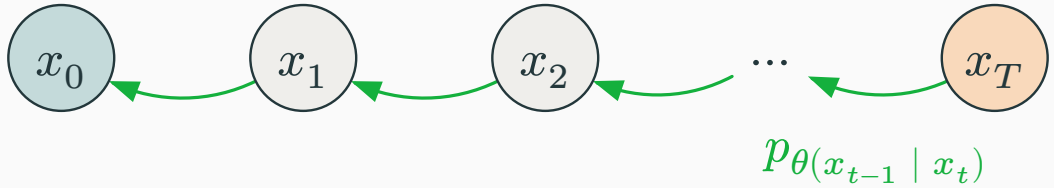


The reverse model factorizes too

D DERIVATION

Sampling runs the learned chain from the easy prior downward:

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



start at $\mathcal{N}(0, I)$, apply T learned reverse transitions, arrive at x_0

Why the closed form matters computationally

D DERIVATION

Without it, one training pair at timestep t would need t **sequential** noising steps:

Why the closed form matters computationally

D DERIVATION

Without it, one training pair at timestep t would need t **sequential** noising steps:

naive — to get x_t, simulate $x_1 \to x_2 \to \ldots \to x_t$. Costly, and different for every t.

closed form — $x_t = \sqrt{\overline{\alpha}_t}x_0 + \sqrt{1-\overline{\alpha}_t}\varepsilon$ in **one** operation, for **any** t.

Why the closed form matters computationally

D DERIVATION

Without it, one training pair at timestep t would need t **sequential** noising steps:

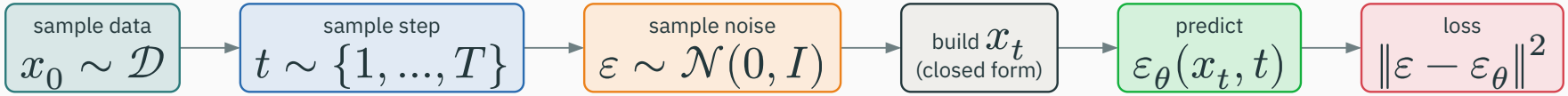
naive — to get x_t, simulate $x_1 \to x_2 \to \ldots \to x_t$. Costly, and different for every t.

closed form — $x_t = \sqrt{\overline{\alpha}_t}x_0 + \sqrt{1-\overline{\alpha}_t}\varepsilon$ in **one** operation, for **any** t.

one random t gives one training example in a **single** step — this is what makes training feasible

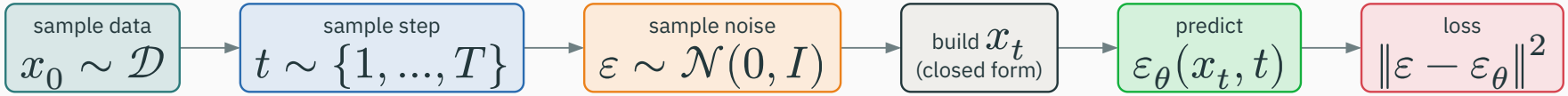
Training touches every noise difficulty

V VISUAL



Training touches every noise difficulty

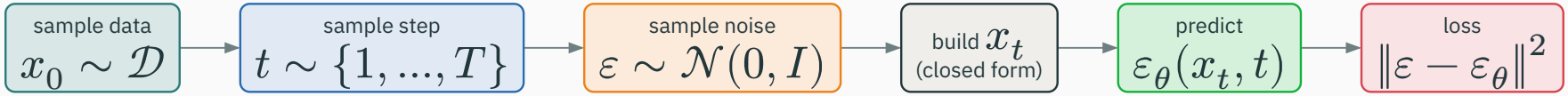
V VISUAL



The five operations of one training step:

Training touches every noise difficulty

V VISUAL

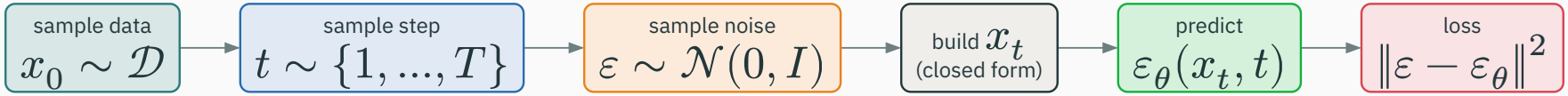


The five operations of one training step:

sample x_0 $\longrightarrow$ sample t $\longrightarrow$ sample ε $\longrightarrow$ build x_t $\longrightarrow$ predict $\varepsilon_{\theta(x_t,t)}$

Training touches every noise difficulty

V VISUAL



The five operations of one training step:

sample x_0 $\longrightarrow$ sample t $\longrightarrow$ sample ε $\longrightarrow$ build x_t $\longrightarrow$ predict $\varepsilon_{\theta(x_t,t)}$

the same data point spawns a **continuum** of self-supervised denoising tasks

Reading the noise schedule

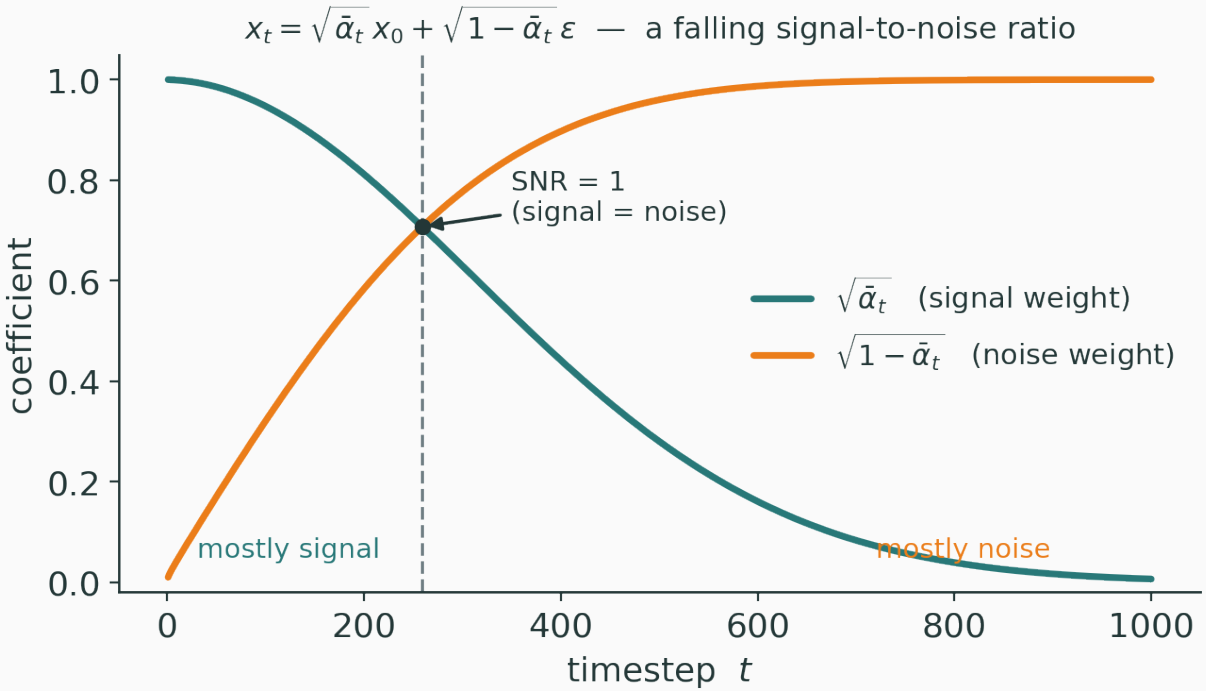
V VISUAL

Plot the signal and noise coefficients against t:

Reading the noise schedule

V VISUAL

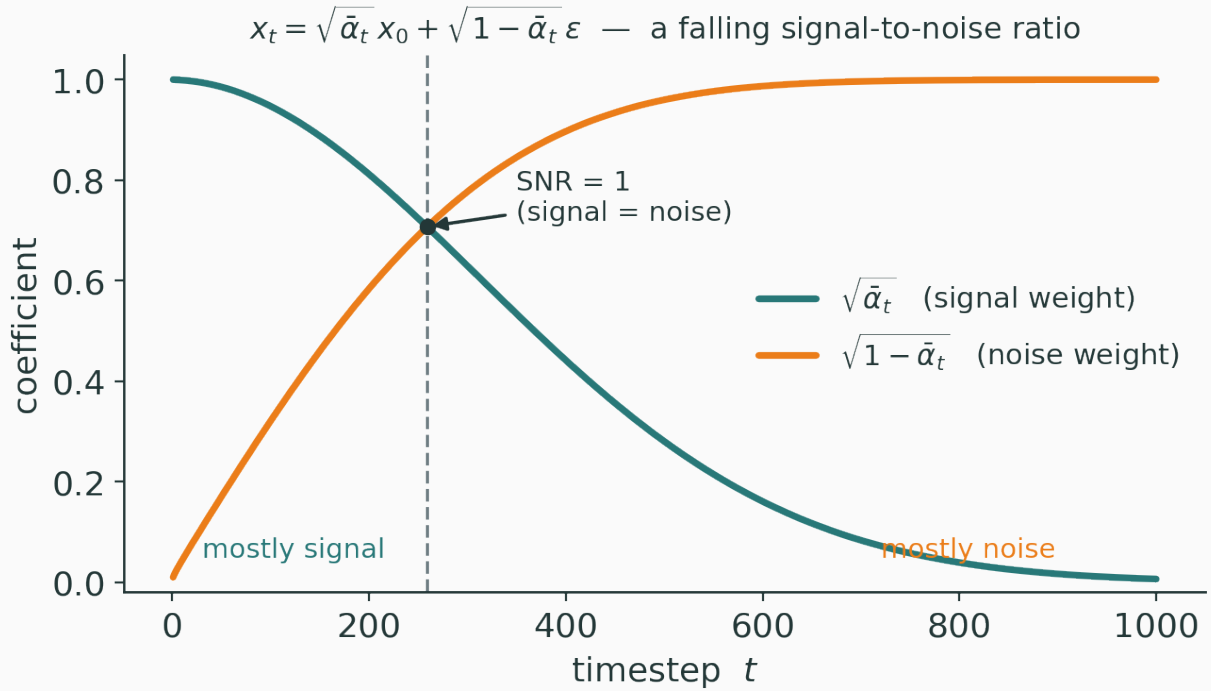
Plot the signal and noise coefficients against t:



Reading the noise schedule

V VISUAL

Plot the signal and noise coefficients against t:



$\sqrt{\bar{\alpha}_t}$ falls from 1; $\sqrt{1-\bar{\alpha}_t}$ rises to 1 — a steadily **falling signal-to-noise ratio**

Different timesteps are different tasks

V VISUAL

Timestep	Input appearance	Main denoising demand
small t	mostly clean	remove fine perturbations
middle t	partly structured	recover objects / layout
large t	near Gaussian noise	rely strongly on the learned prior

Different timesteps are different tasks

V VISUAL

Timestep	Input appearance	Main denoising demand
small t	mostly clean	remove fine perturbations
middle t	partly structured	recover objects / layout
large t	near Gaussian noise	rely strongly on the learned prior

the time embedding tells the network **which** of these regimes it is in

Why predict noise rather than the image?

D DERIVATION

Several equivalent parameterizations exist:

Why predict noise rather than the image?

D DERIVATION

Several equivalent parameterizations exist:

$$\varepsilon \quad \text{(noise)}, \qquad x_0 \quad \text{(clean image)}, \qquad v \quad \text{(velocity)}$$

Why predict noise rather than the image?

D DERIVATION

Several equivalent parameterizations exist:

$$\varepsilon \quad (\text{noise}), \qquad x_0 \quad (\text{clean image}), \qquad v \quad (\text{velocity})$$

- **noise prediction** gives a target with a **known, unit-scale** Gaussian across all t — nicely conditioned;
- practical systems may use x_0 or v; the underlying diffusion principle is **unchanged**.

Why predict noise rather than the image?

D DERIVATION

Several equivalent parameterizations exist:

$$\varepsilon \quad (\text{noise}), \qquad x_0 \quad (\text{clean image}), \qquad v \quad (\text{velocity})$$

- **noise prediction** gives a target with a **known, unit-scale** Gaussian across all t — nicely conditioned;
- practical systems may use x_0 or v; the underlying diffusion principle is **unchanged**.

the parameterization is a **modeling choice**, not the essence of diffusion

MSE is a conditional regression loss

D DERIVATION

From an ambiguous x_t, the network cannot recover the **exact** unknowable noise draw:

MSE is a conditional regression loss

D DERIVATION

From an ambiguous x_t, the network cannot recover the **exact** unknowable noise draw:

$$\varepsilon_{\theta(x_t,t)} \quad \rightarrow \quad \mathbb{E}[\varepsilon \mid x_t, t]$$

MSE is a conditional regression loss

D DERIVATION

From an ambiguous x_t, the network cannot recover the **exact** unknowable noise draw:

$$\varepsilon_{\theta(x_t,t)} \quad \rightarrow \quad \mathbb{E}[\,\varepsilon \mid x_t, t\,]$$

Minimizing MSE makes ε_θ approach the **conditional mean** of the noise given x_t. It blends the evidence in x_t with the **data prior** — exactly what a good denoiser should do.

Time must enter the network

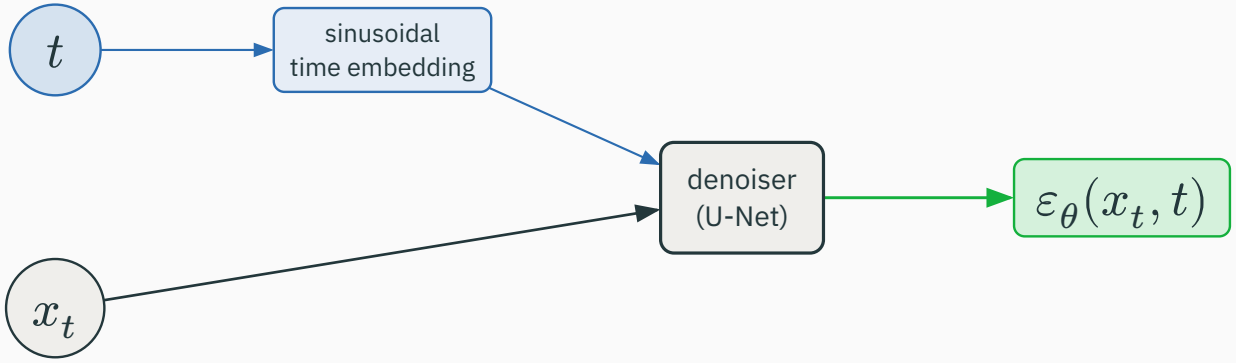
V VISUAL

The **same** tensor can occur at many noise levels — the model needs to know which:

Time must enter the network

V VISUAL

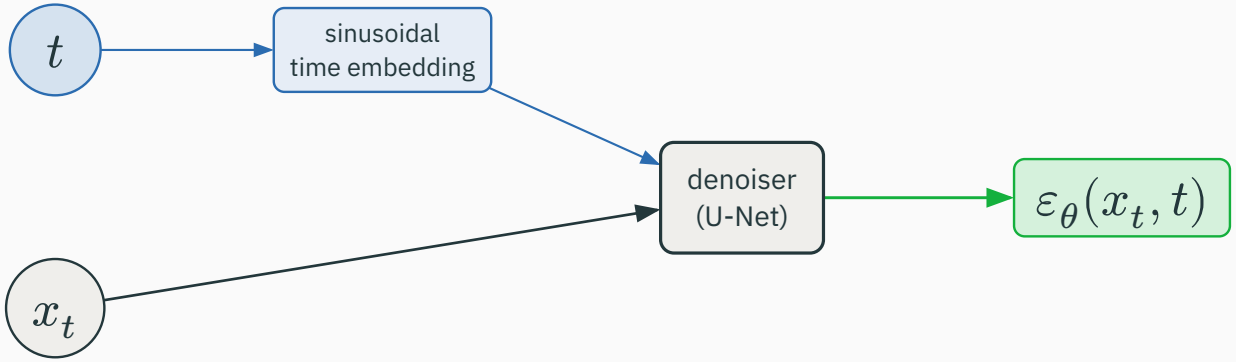
The **same** tensor can occur at many noise levels — the model needs to know which:



Time must enter the network

V VISUAL

The **same** tensor can occur at many noise levels — the model needs to know which:



$t \longrightarrow$ sinusoidal / learned embedding $\longrightarrow$ injected into the denoiser blocks

Reverse-mean intuition

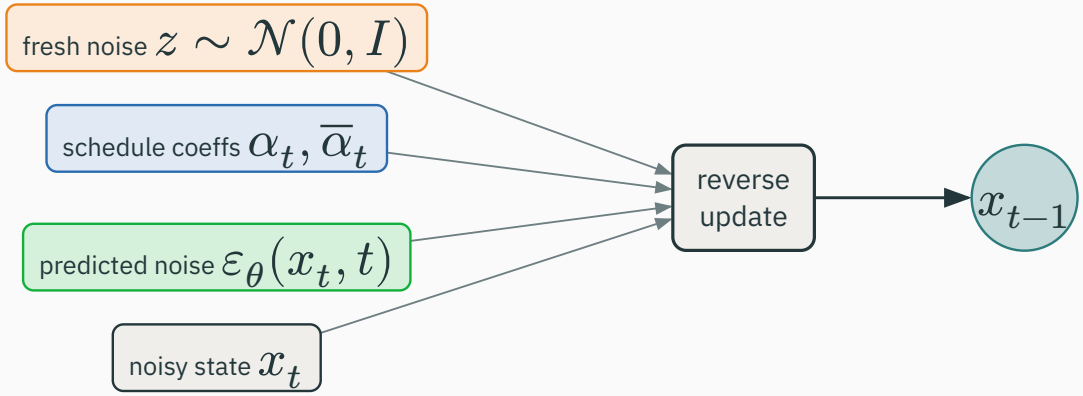
V VISUAL

One reverse step **combines** four ingredients — no need to memorize the coefficients:

Reverse-mean intuition

V VISUAL

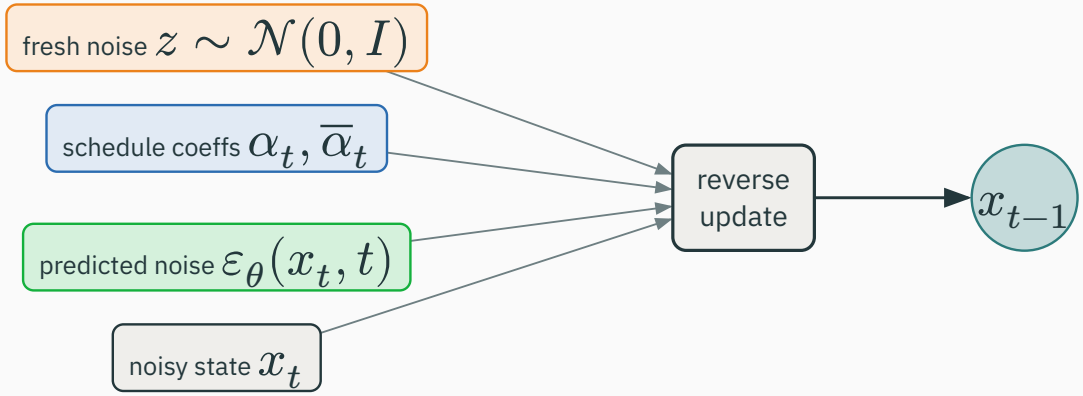
One reverse step **combines** four ingredients — no need to memorize the coefficients:



Reverse-mean intuition

V VISUAL

One reverse step **combines** four ingredients — no need to memorize the coefficients:



current state, predicted noise, schedule, and (optionally) fresh randomness $\longrightarrow x_{t-1}$

Stochastic vs deterministic sampling

D DERIVATION

Once the denoiser is trained, **how** you step back is a separate choice:

Stochastic vs deterministic sampling

D DERIVATION

Once the denoiser is trained, **how** you step back is a separate choice:

stochastic (e.g. DDPM) — add fresh noise z at each reverse step; more **diverse** samples.

deterministic (e.g. DDIM) — no added noise once x_T is fixed; **reproducible**, often **fewer** steps.

Stochastic vs deterministic sampling

D DERIVATION

Once the denoiser is trained, **how** you step back is a separate choice:

stochastic (e.g. DDPM) — add fresh noise z at each reverse step; more **diverse** samples.

deterministic (e.g. DDIM) — no added noise once x_T is fixed; **reproducible**, often **fewer** steps.

this changes diversity, reproducibility, and speed — **not** the learned denoiser itself

One sample, many intermediate states

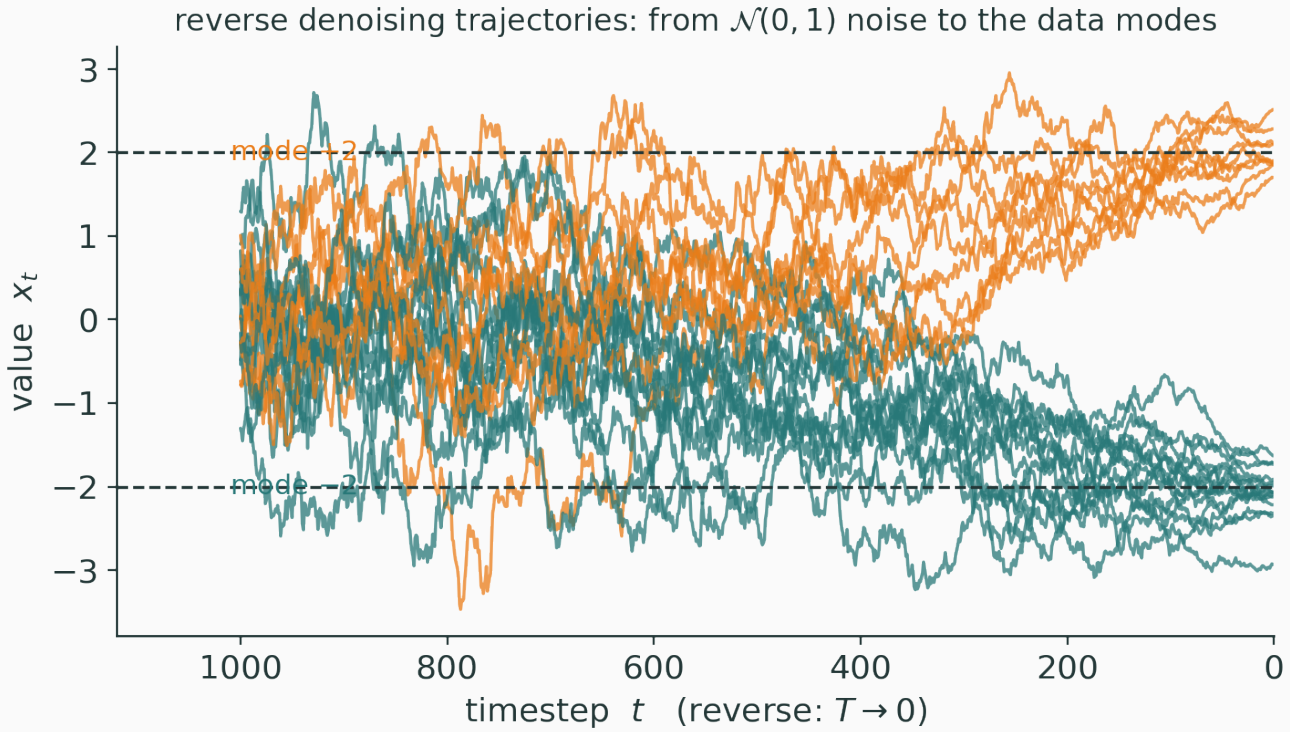
V VISUAL

Run the reverse chain and watch a value settle onto a data mode:

One sample, many intermediate states

V VISUAL

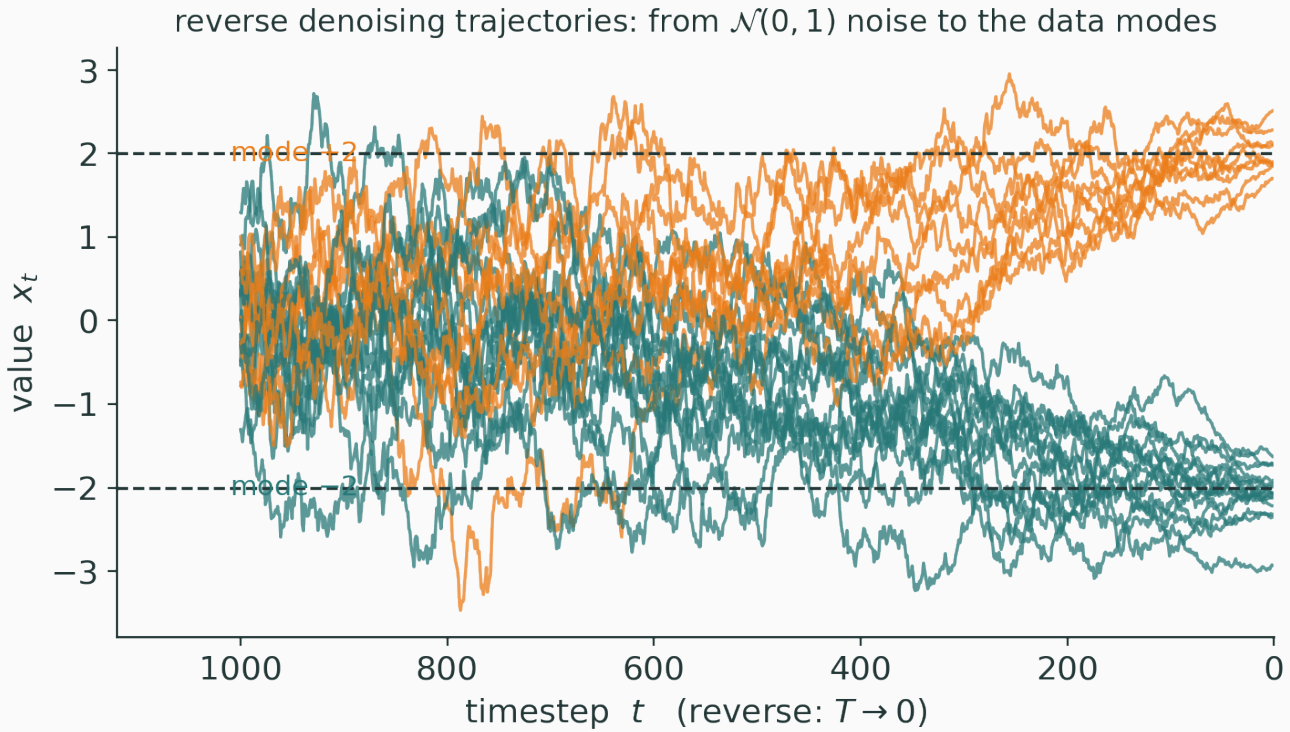
Run the reverse chain and watch a value settle onto a data mode:



One sample, many intermediate states

V VISUAL

Run the reverse chain and watch a value settle onto a data mode:



global choice (“which mode?”) is made **early**; fine position is refined **late** — an observation, not a law

Relation to denoising autoencoders

D DERIVATION

A denoising autoencoder learns one corruption level:

Relation to denoising autoencoders

D DERIVATION

A denoising autoencoder learns one corruption level:

$$\text{corrupted input} \quad \rightarrow \quad \text{clean input}$$

Relation to denoising autoencoders

D DERIVATION

A denoising autoencoder learns one corruption level:

$$\text{corrupted input} \quad \rightarrow \quad \text{clean input}$$

diffusion is a DAE at every calibrated noise level, plus a generative reverse chain

Relation to denoising autoencoders

D DERIVATION

A denoising autoencoder learns one corruption level:

$$\text{corrupted input} \quad \rightarrow \quad \text{clean input}$$

diffusion is a DAE at every calibrated noise level, plus a generative reverse chain

the extra ingredients: a **sequence** of levels, **time conditioning**, and **sampling from noise**

The variational perspective, placed

D DERIVATION

The original DDPM objective comes from a **variational bound** on the data likelihood:

The variational perspective, placed

D DERIVATION

The original DDPM objective comes from a **variational bound** on the data likelihood:

$-\log p_{\theta(x_0)} \leq \mathbb{E}_q[\dots]$ (a sum of per-step KL terms — like L19's ELBO)

The variational perspective, placed

D DERIVATION

The original DDPM objective comes from a **variational bound** on the data likelihood:

$-\log p_{\theta(x_0)} \leq \mathbb{E}_q[\,\dots\,]$ (a sum of per-step KL terms — like L19's ELBO)

The simple noise-prediction MSE L_{simple} is a **reweighted simplification** of that likelihood bound — it drops constants and per-timestep weights that hurt sample quality.

The variational perspective, placed

D DERIVATION

The original DDPM objective comes from a **variational bound** on the data likelihood:

$$-\log p_{\theta(x_0)} \leq \mathbb{E}_q[\,\dots\,] \qquad \text{(a sum of per-step KL terms — like L19's ELBO)}$$

The simple noise-prediction MSE L_{simple} is a **reweighted simplification** of that likelihood bound — it drops constants and per-timestep weights that hurt sample quality.

connect to the ELBO from L19 — we do **not** re-derive the whole bound here

Score intuition, revisited

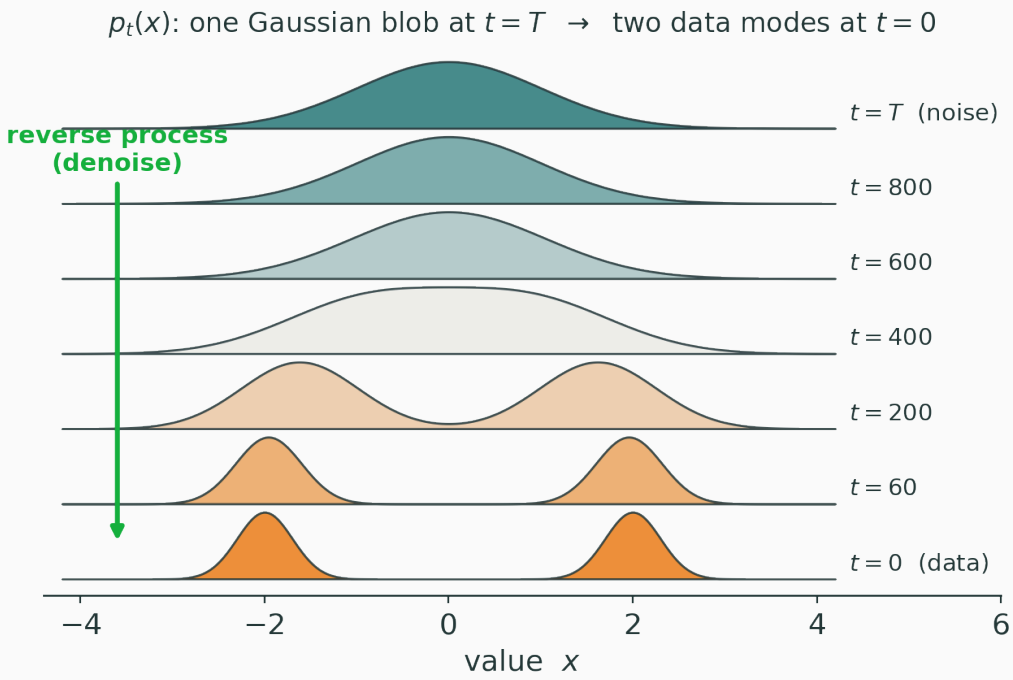
V VISUAL

For a noisy point x_t, the score points toward denser regions of p_t:

Score intuition, revisited

V VISUAL

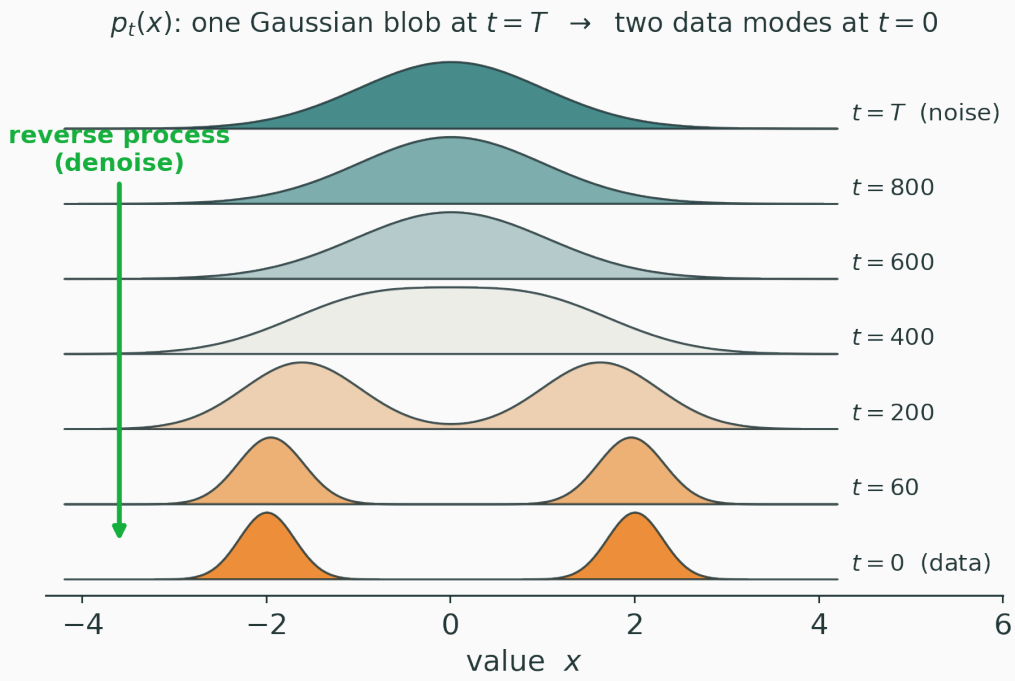
For a noisy point x_t, the score points toward denser regions of p_t:



Score intuition, revisited

V VISUAL

For a noisy point x_t, the score points toward denser regions of p_t:



the reverse process rides these directions from one blurry blob back to the data modes

Conditioning changes the reverse distribution

D DERIVATION

A condition c reshapes **which** reverse paths are favored:

Conditioning changes the reverse distribution

D DERIVATION

A condition c reshapes **which** reverse paths are favored:

$$p_\theta(x_{t-1} \mid x_t, c)$$

Conditioning changes the reverse distribution

D DERIVATION

A condition c reshapes **which** reverse paths are favored:

$$p_\theta(x_{t-1} \mid x_t, c)$$

text, class labels, masks, low-res images — all change the **same** conditional denoising question

Conditioning changes the reverse distribution

D DERIVATION

A condition c reshapes **which** reverse paths are favored:

$$p_\theta(x_{t-1} \mid x_t, c)$$

text, class labels, masks, low-res images — all change the **same** conditional denoising question

INTERACTIVE

↗ nipunbatra.github.io/interactive-articles/classifier-guidance

A conditioning widget: watch the reverse trajectory bend toward a chosen class.

Classifier guidance, previewed

D DERIVATION

An external classifier can push the trajectory toward a target class:

Classifier guidance, previewed

D DERIVATION

An external classifier can push the trajectory toward a target class:

$\nabla_{x_t} \log p(c \mid x_t)$ (gradient that says "look more like class c")

Classifier guidance, previewed

D DERIVATION

An external classifier can push the trajectory toward a target class:

$\nabla_{x_t} \log p(c \mid x_t)$ (gradient that says "look more like class c")

add this to the score to **steer** sampling — L22 teaches the practical **classifier-free** version

Architectural preview: the denoiser

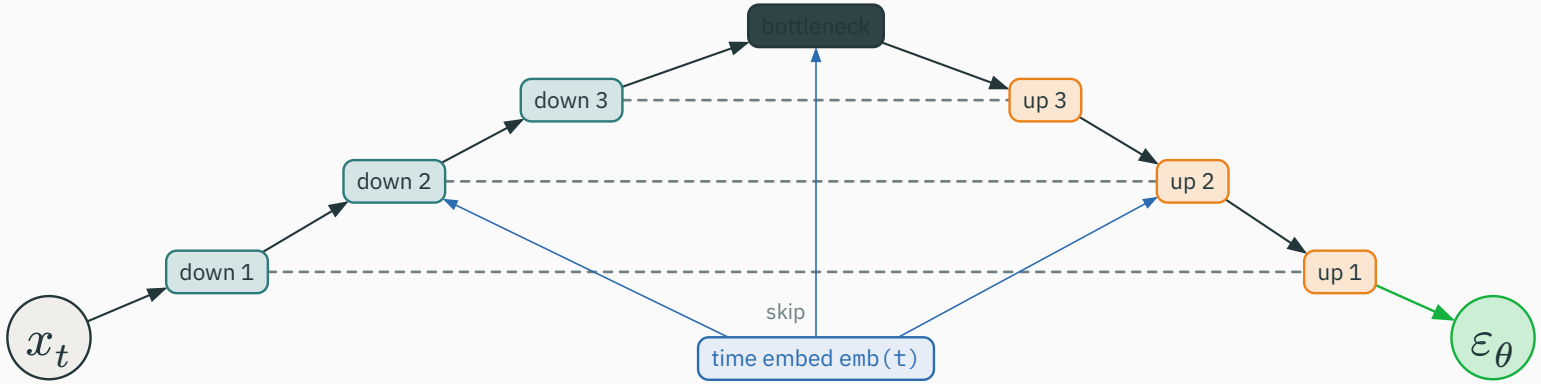
V VISUAL

The loss says **what** to predict; the architecture says **how** to compute it:

Architectural preview: the denoiser

V VISUAL

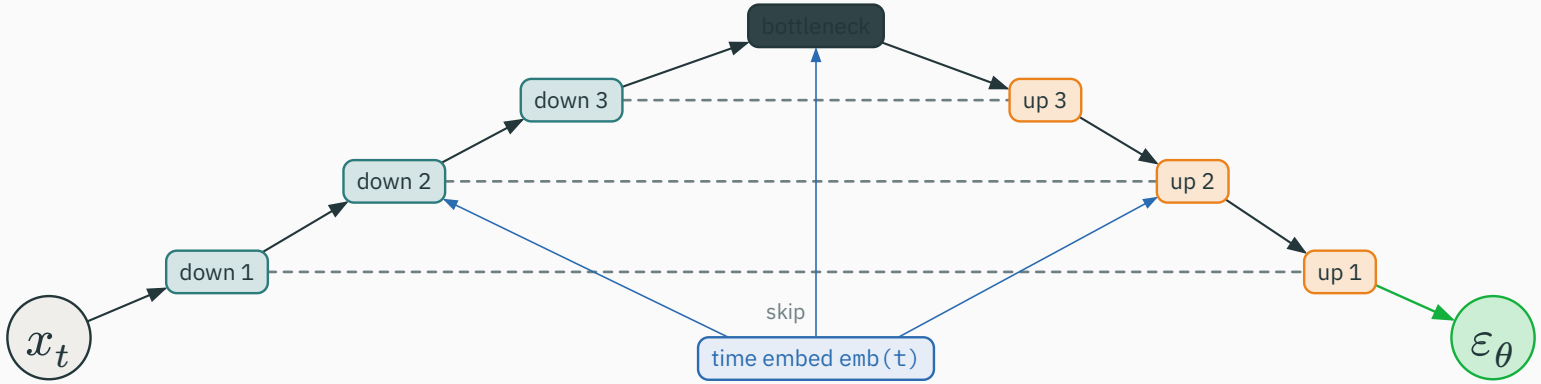
The loss says **what** to predict; the architecture says **how** to compute it:



Architectural preview: the denoiser

V VISUAL

The loss says **what** to predict; the architecture says **how** to compute it:



a **U-Net** (or diffusion Transformer): multi-scale, skip connections, time embedding in every block

Why diffusion training is comparatively stable

D DERIVATION

Every example has a **fixed, known** target:

Why diffusion training is comparatively stable

D DERIVATION

Every example has a **fixed, known** target:

$$\text{target} = \varepsilon \qquad \text{(the noise we drew — it never moves)}$$

Why diffusion training is comparatively stable

D DERIVATION

Every example has a **fixed, known** target:

$$\text{target} = \varepsilon \qquad \text{(the noise we drew — it never moves)}$$

Unlike a GAN, **no learned opponent** shifts the target between batches — there is no minimax game to diverge.

Why diffusion training is comparatively stable

D DERIVATION

Every example has a **fixed, known** target:

$$\text{target} = \varepsilon \qquad \text{(the noise we drew — it never moves)}$$

Unlike a GAN, **no learned opponent** shifts the target between batches — there is no minimax game to diverge.

optimization is not **trivial**, but the central instability of adversarial training is **gone**

Why diffusion sampling is expensive

V VISUAL

Model family	**Typical generation structure**
VAE	one decode
GAN	one generator pass
diffusion	**repeated** denoiser calls (T of them)

Why diffusion sampling is expensive

V VISUAL

Model family	Typical generation structure
VAE	one decode
GAN	one generator pass
diffusion	**repeated** denoiser calls (T of them)

the model trades **inference work** for a **tractable, stable** training signal

Common misconceptions

Q QUESTION

Misconception	**Correction**
diffusion memorizes a reverse “noise table”	a network **generalizes** denoising across inputs and conditions
the forward process is learned	in standard DDPMs it is **chosen and fixed**
the network only predicts the clean image	noise, clean image, or velocity can be parameterized
more steps always mean a better image	schedules and samplers set a quality / latency **trade-off**

Retrieval questions

Q QUESTION

1. Why can x_t be sampled **directly** from x_0?

Retrieval questions

Q QUESTION

1. Why can x_t be sampled **directly** from x_0?
2. What target is available during diffusion training **without labels**?

Retrieval questions

Q QUESTION

1. Why can x_t be sampled **directly** from x_0?
2. What target is available during diffusion training **without labels**?
3. Why **must** the denoiser know t?

Retrieval questions

Q QUESTION

1. Why can x_t be sampled **directly** from x_0?
2. What target is available during diffusion training **without labels**?
3. Why **must** the denoiser know t?
4. Why is generation **slower** than GAN sampling?

Retrieval questions

Q QUESTION

1. Why can x_t be sampled **directly** from x_0?
2. What target is available during diffusion training **without labels**?
3. Why **must** the denoiser know t?
4. Why is generation **slower** than GAN sampling?

if all four are quick to answer, the core of the lecture is secure

The β_t schedule is a design choice

D DERIVATION

The schedule sets the **sequence of learning problems**:

The β_t schedule is a design choice

D DERIVATION

The schedule sets the **sequence of learning problems**:

too little noise — x_T stays structured; the prior $\mathcal{N}(0, I)$ is a poor starting point.

too abrupt noise — intermediate denoising tasks become needlessly hard.

The β_t schedule is a design choice

D DERIVATION

The schedule sets the **sequence of learning problems**:

too little noise — x_T stays structured; the prior $\mathcal{N}(0, I)$ is a poor starting point.

too abrupt noise — intermediate denoising tasks become needlessly hard.

smooth schedules (linear, cosine) spread difficulty **evenly** across timesteps

Variance in reverse sampling

D DERIVATION

The learned reverse Gaussian has a mean **and** a variance:

Variance in reverse sampling

D DERIVATION

The learned reverse Gaussian has a mean **and** a variance:

$$p_{\theta(x_{t-1} \mid x_t)} = \mathcal{N}(\mu_\theta, \Sigma_\theta)$$

Variance in reverse sampling

D DERIVATION

The learned reverse Gaussian has a mean **and** a variance:

$$p_{\theta(x_{t-1} \mid x_t)} = \mathcal{N}(\mu_\theta, \Sigma_\theta)$$

- Σ_θ may be **fixed** to a schedule value or **learned**;
- it controls how much randomness each reverse step injects.

Variance in reverse sampling

D DERIVATION

The learned reverse Gaussian has a mean **and** a variance:

$$p_{\theta(x_{t-1} \mid x_t)} = \mathcal{N}(\mu_\theta, \Sigma_\theta)$$

- Σ_θ may be **fixed** to a schedule value or **learned**;
- it controls how much randomness each reverse step injects.

variance is a **modeling knob**, not an arbitrary visual-effect dial

Loss weighting across timesteps

D DERIVATION

An “MSE loss” still hides modeling choices:

Loss weighting across timesteps

D DERIVATION

An “MSE loss” still hides modeling choices:

- not every t need contribute **equally** to the objective;
- parameterization (ε / x_0 / v) and schedule change **which** SNR regimes dominate learning.

Loss weighting across timesteps

D DERIVATION

An “MSE loss” still hides modeling choices:

- not every t need contribute **equally** to the objective;
- parameterization (ε / x_0 / v) and schedule change **which** SNR regimes dominate learning.

reweighting timesteps trades off **coarse structure** against **fine detail** quality

Data modalities beyond images

D DERIVATION

The same recipe travels — only the denoiser and conditioning change:

Data modalities beyond images

D DERIVATION

The same recipe travels — only the denoiser and conditioning change:

- **audio** waveforms or spectrograms; **video** with temporal structure;
- **molecular** coordinates; **trajectories** and actions; continuous **latent** representations.

Data modalities beyond images

D DERIVATION

The same recipe travels — only the denoiser and conditioning change:

- **audio** waveforms or spectrograms; **video** with temporal structure;
- **molecular** coordinates; **trajectories** and actions; continuous **latent** representations.

for standard continuous-data DDPMs, forward = Gaussian corruption; the **architecture** must respect the data's structure

Why direct pixel diffusion is demanding

D DERIVATION

Noise is added independently to a **very high-dimensional** tensor:

Why direct pixel diffusion is demanding

D DERIVATION

Noise is added independently to a **very high-dimensional** tensor:

a good denoiser must fix texture, object identity, global layout, and condition alignment at once

Why direct pixel diffusion is demanding

D DERIVATION

Noise is added independently to a **very high-dimensional** tensor:

a good denoiser must fix texture, object identity, global layout, and condition alignment at once

this pressure motivates L22′s **latent-space** diffusion and multi-scale architectures

A tiny one-dimensional thought experiment

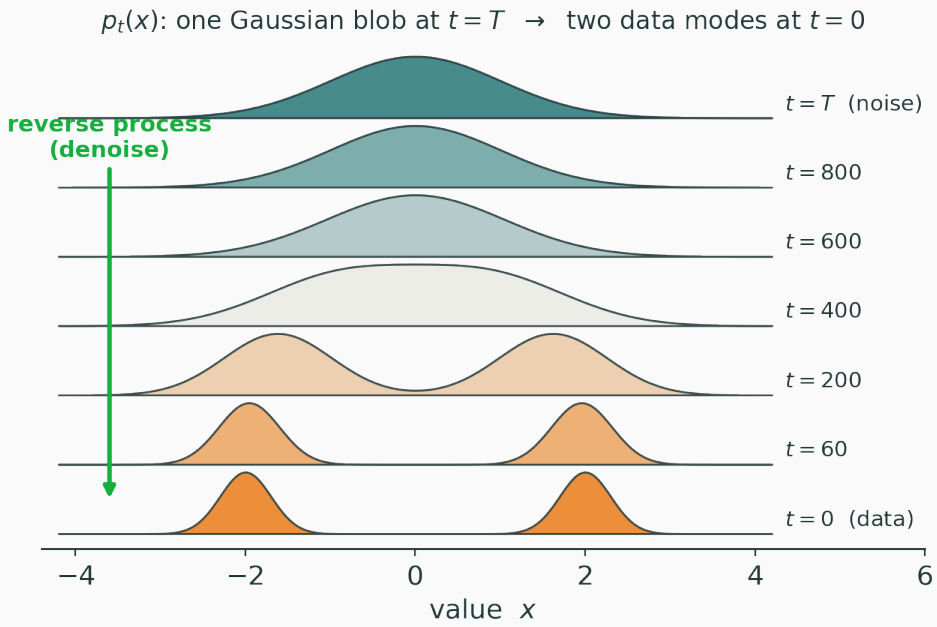
V VISUAL

Data lives near -2 and $+2$ on a line; at high noise the two modes **blend**:

A tiny one-dimensional thought experiment

V VISUAL

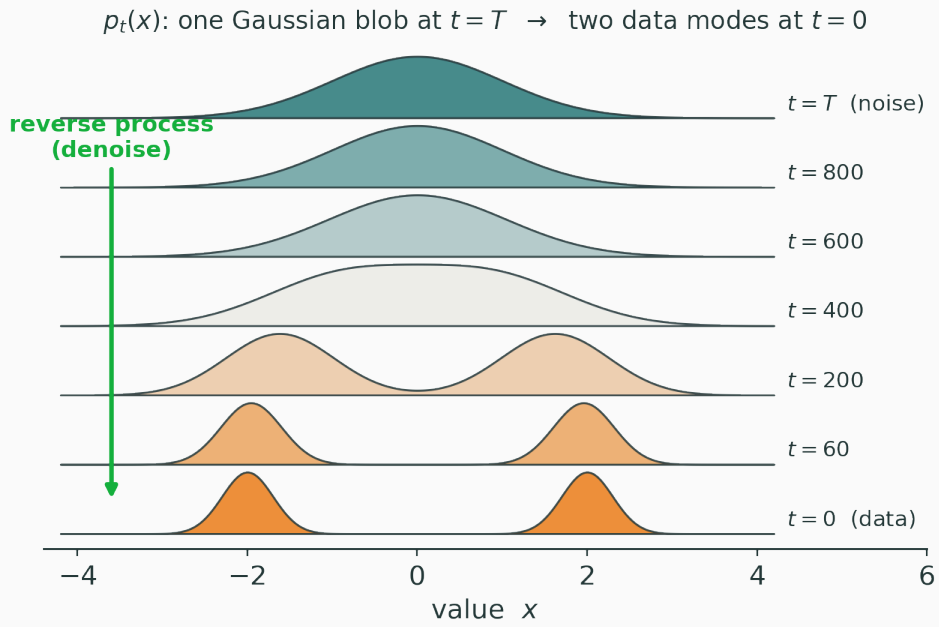
Data lives near -2 and $+2$ on a line; at high noise the two modes **blend**:



A tiny one-dimensional thought experiment

V VISUAL

Data lives near -2 and $+2$ on a line; at high noise the two modes **blend**:



the reverse model must **decide which mode** a noisy point should approach — generation is inherently probabilistic

Denoising is not a nearest-neighbour lookup

D DERIVATION

For a noisy point between modes, the model uses **learned structure**, not a database:

Denoising is not a nearest-neighbour lookup

D DERIVATION

For a noisy point between modes, the model uses **learned structure**, not a database:

- the result may resemble **interpolation**, **recombination**, or a genuinely **new** sample;

Denoising is not a nearest-neighbour lookup

D DERIVATION

For a noisy point between modes, the model uses **learned structure**, not a database:

- the result may resemble **interpolation**, **recombination**, or a genuinely **new** sample;

Separately: **memorization** remains an empirical risk to **test for** on small or narrow datasets.

A diffusion evaluation lens

Q QUESTION

No single grid or scalar settles quality — evaluate along five axes:

A diffusion evaluation lens

Q QUESTION

No single grid or scalar settles quality — evaluate along five axes:

- **fidelity**; **coverage / diversity**; **conditioning adherence**;
- **sampling cost**; **robustness and harmful failure modes**.

A diffusion evaluation lens

Q QUESTION

No single grid or scalar settles quality — evaluate along five axes:

- **fidelity**; **coverage / diversity**; **conditioning adherence**;
- **sampling cost**; **robustness and harmful failure modes**.

FID and a caption match tell you **something** — never **everything**

Theory-to-practice checklist

V VISUAL

Before reading any implementation, locate it with five questions:

Theory-to-practice checklist

V VISUAL

Before reading any implementation, locate it with five questions:

1. data space — **pixels or latents**?
2. **denoiser architecture**?
3. **time / noise parameterization**?
4. **condition interface**?
5. **sampler** and step budget?

Theory-to-practice checklist

V VISUAL

Before reading any implementation, locate it with five questions:

1. data space — **pixels or latents**?
2. **denoiser architecture**?
3. **time / noise parameterization**?
4. **condition interface**?
5. **sampler** and step budget?

these five coordinates place nearly **every** diffusion system in the course map

Common equations, one page

V VISUAL

Idea	Equation
forward step	$q(x_t \mid x_{t-1}) = \mathcal{N}\left(\sqrt{1-\beta_t}\, x_{t-1}, \beta_t I\right)$
closed form	$x_t = \sqrt{\overline{\alpha}_t}\, x_0 + \sqrt{1-\overline{\alpha}_t}\, \varepsilon$
training loss	$L = \mathbb{E}\left\|\varepsilon - \varepsilon_{\theta(x_t,t)}\right\|_2^2$
clean estimate	$\hat{x}_0 = \left(x_t - \sqrt{1-\overline{\alpha}_t}\, \varepsilon_\theta\right) / \sqrt{\overline{\alpha}_t}$

Common equations, one page

V VISUAL

Idea	Equation
forward step	$q(x_t \mid x_{t-1}) = \mathcal{N}\left(\sqrt{1-\beta_t}\, x_{t-1}, \beta_t I\right)$
closed form	$x_t = \sqrt{\overline{\alpha}_t}\, x_0 + \sqrt{1-\overline{\alpha}_t}\, \varepsilon$
training loss	$L = \mathbb{E}\left\|\varepsilon - \varepsilon_{\theta(x_t,t)}\right\|_2^2$
clean estimate	$\hat{x}_0 = \left(x_t - \sqrt{1-\overline{\alpha}_t}\, \varepsilon_\theta\right) / \sqrt{\overline{\alpha}_t}$

a final reference — **not** another derivation

Retrieval questions, extended

Q QUESTION

1. Which part of diffusion is **fixed** and which part is **learned**?

Retrieval questions, extended

Q QUESTION

1. Which part of diffusion is **fixed** and which part is **learned**?
2. How does the closed form turn one image into **many** training examples?

Retrieval questions, extended

Q QUESTION

1. Which part of diffusion is **fixed** and which part is **learned**?
2. How does the closed form turn one image into **many** training examples?
3. Why is the reverse direction **conditional and probabilistic**?

Retrieval questions, extended

Q QUESTION

1. Which part of diffusion is **fixed** and which part is **learned**?
2. How does the closed form turn one image into **many** training examples?
3. Why is the reverse direction **conditional and probabilistic**?
4. What trade-off does **iterative sampling** introduce?

Next lecture

D DERIVATION

noise-prediction objective

$+$ latent space, U-Net / Transformer, and conditioning

$=$ a **practical** diffusion system

Next lecture

D DERIVATION

noise-prediction objective

+ latent space, U-Net / Transformer, and conditioning

= a **practical** diffusion system

L22 builds that system component by component — including **Stable Diffusion**

If you know how to **add** noise, you can train a network to **remove** it.

Next: **Diffusion in practice** → latent diffusion, U-Nets, guidance, and Stable Diffusion.