▶ Notebook

Lecture 4 at a glance: following Paras Chopra's explainer

Introduction: destruction as creation

0.1 Static in, a cat out ▶
0.2 Destruction as creation ▶

Part 1 · The forward process

1.1 One noising step ▶
1.2 A toy schedule, by hand ▶
1.3 A beautiful shortcut ▶
1.4 The real schedule ▶
1.5 Watch it dissolve ▶

Part 2 · The reverse process

2.1 Run the film backwards ▶
2.2 Small steps back are simple ▶
2.3 Guess the noise ▶
2.4 One step back ▶

Part 3 · What the network learns, and training

3.1 What does the network learn? ▶
3.2 The network ▶
3.3 The training loop ▶
3.4 Training, live ▶
3.5 What the network believes ▶

Part 4 · Sampling: birth of an image

4.1 One step back, by hand ▶
4.2 The toy, all the way back ▶
4.3 The sampling loop ▶
4.4 Watch it ▶
4.5 Why the fresh noise? ▶
4.6 New digits ▶

Part 5 · The score: arrows to the data

5.1 Arrows to the data ▶
5.2 Follow the arrows ▶

Part 6 · Noise schedules and faster sampling

6.1 Noise schedules ▶
6.2 DDIM: fewer steps ▶
6.3 Faster still ▶

Part 7 · Steering: labels, guidance and text

7.1 A 3 on demand ▶
7.2 Classifier-free guidance ▶
7.3 The guidance dial ▶
7.4 From pixels to latents ▶
7.5 Text in ▶
7.6 A sentence in, a picture out ▶

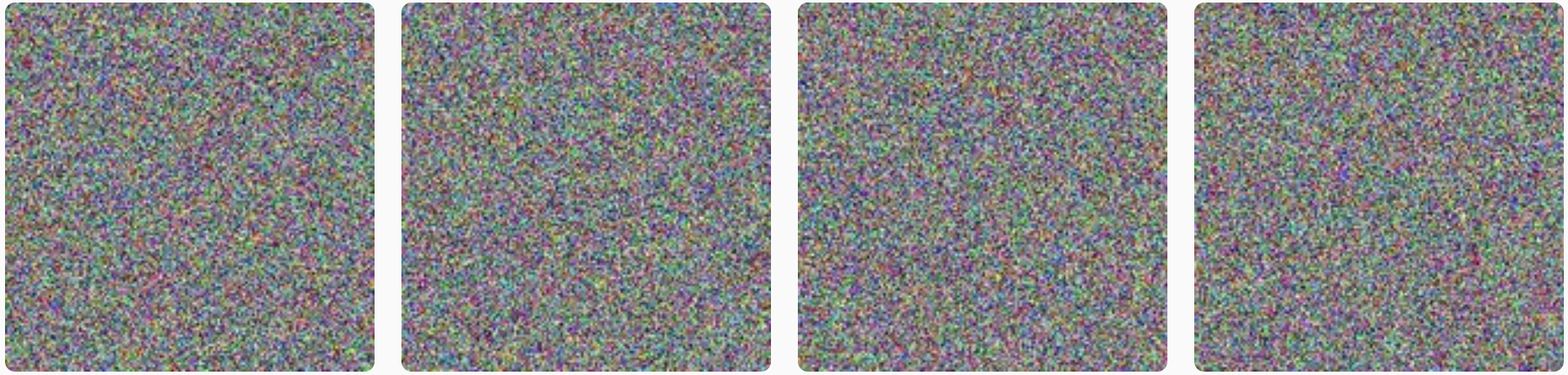
Part 8 · Wrap-up

8.1 Wrap-up ▶

> Two sessions, in the order of Paras Chopra's "Diffusion Models from First Principles". Parts 0–4 build a working diffusion model, following every number by hand. Parts 5–8: the score view, faster sampling, and steering with labels and text. The ▶ Notebook link on every slide opens its section in Colab.

▶ Notebook 0.1

Static in, a cat out



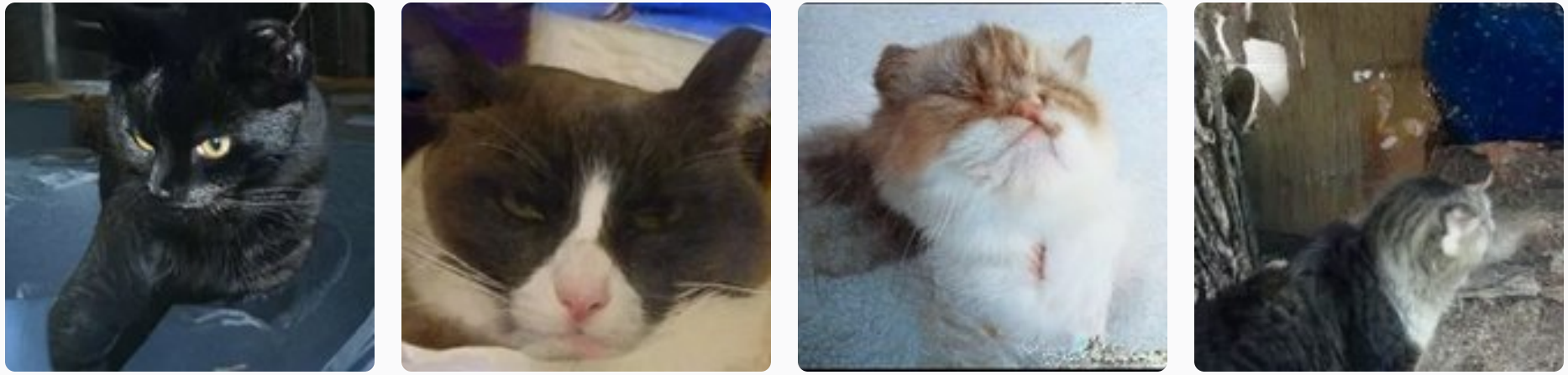
t = 1000 · step 0 of 1,000

> The network of the original DDPM paper, trained on photos of cats, starts from pure static: 196,608 random numbers per image…

Ho, Jain & Abbeel, "Denoising Diffusion Probabilistic Models" (2020) · Companion film "Diffusion models, from noise to images, step by step" (2026): google/ddpm-cat-256 run on the CC0 photo "Chelsea" (scikit-image)

▶ Notebook 0.1

Static in, a cat out



t = 0 · step 1000 of 1,000

the original DDPM model Real samples from the network of the original DDPM paper (Ho, Jain & Abbeel 2020), trained on photos of cats.

> …and a thousand small steps later there is a cat. Today: how it works, built from scratch on handwritten digits, where we can see every number.

Ho, Jain & Abbeel, "Denoising Diffusion Probabilistic Models" (2020) · Companion film "Diffusion models, from noise to images, step by step" (2026): google/ddpm-cat-256 run on the CC0 photo "Chelsea" (scikit-image)

▶ Notebook 0.1

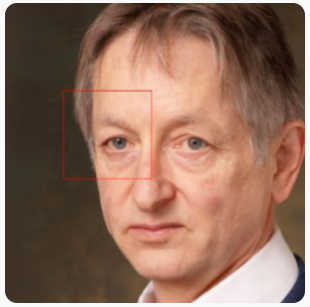
The same loop runs today's image generators



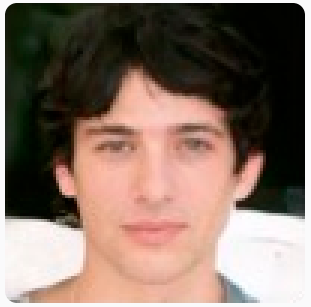
"a basket of cherries"



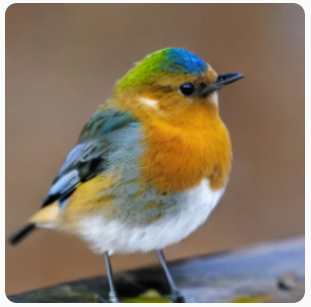
fill a hole



sharpen a blurry face



a sketch → a scene



edges → a bird



your dog, at the Acropolis

> Text to image, filling a hole, sharpening a face, a sketch into a scene, edges into a bird, one dog in a new place. Different inputs, one recipe.

Published samples: LDM (Rombach et al. 2022) · RePaint (Lugmayr et al. 2022) · SR3 (Saharia et al. 2022) · SDEdit (Meng et al. 2022) · ControlNet (Zhang et al. 2023) · DreamBooth (Ruiz et al. 2023). Showcase after MIT 6.S183 "A Practical Introduction to Diffusion Models" (IAP 2025), Lecture 1

▶ Notebook 0.1

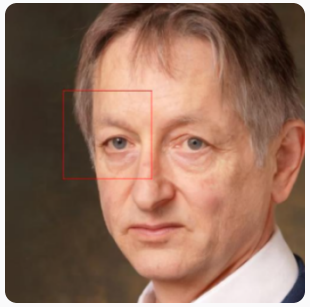
The same loop runs today's image generators



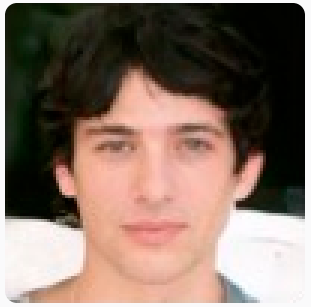
"a basket of cherries"



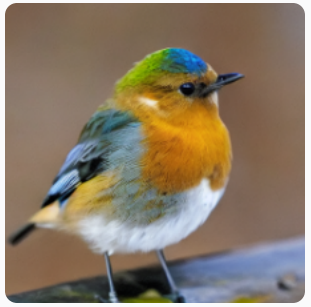
fill a hole



sharpen a blurry face



a sketch → a scene



edges → a bird



your dog, at the Acropolis

published Video, audio, 3-D shapes, protein structures and robot motions are made the same way.

> The recipe goes beyond images: video, audio, proteins, robot policies. By the end you can say what each network was trained to predict.

Published samples: LDM (Rombach et al. 2022) · RePaint (Lugmayr et al. 2022) · SR3 (Saharia et al. 2022) · SDEdit (Meng et al. 2022) · ControlNet (Zhang et al. 2023) · DreamBooth (Ruiz et al. 2023). Showcase after MIT 6.S183 "A Practical Introduction to Diffusion Models" (IAP 2025), Lecture 1

▶ Notebook 0.2

The idea in one sentence

If you learn to undo each tiny step of a destruction process, you can create from scratch.

Paras Chopra, “Diffusion Models from First Principles”

> Destroying is easy; creating is hard. The trick is to connect them.

▶ Notebook 0.2

The idea in one sentence

If you learn to undo each tiny step of a destruction process, you can create from scratch.

Paras Chopra, “Diffusion Models from First Principles”



illustration Generated with Gemini. A drop of ink spreads until the water is uniform; nothing pulls it back together.

> Film a drop of ink spreading through water. Played forwards, structure disappears. Between two neighbouring frames the change is tiny, and tiny changes can be predicted. Learn to undo each one, and you can run the film backwards.

▶ Notebook 0.2

The plan: forwards, backwards, train, sample

q: add noise, step by step. Fixed: nothing to learn.



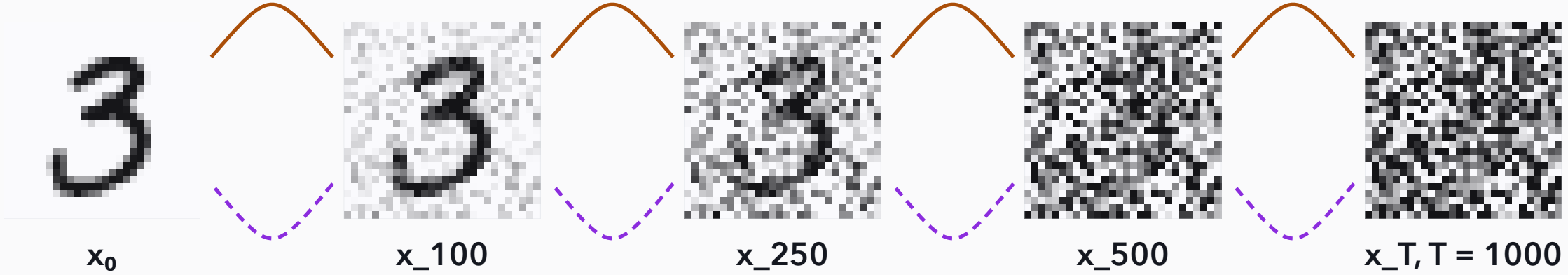
> Left to right: add a little noise, a thousand times, until our 3 is pure static. That direction is fixed and needs no learning (Part 1).

Ho, Jain & Abbeel, "Denoising Diffusion Probabilistic Models" (2020)

▶ Notebook 0.2

The plan: forwards, backwards, train, sample

q: add noise, step by step. Fixed: nothing to learn.



p_θ: remove noise, step by step. Learned: today's network.

> Right to left: remove a little noise at a time. That direction is learned (Parts 2–4). Then the score view (5), faster sampling (6), and steering with labels and text (7).

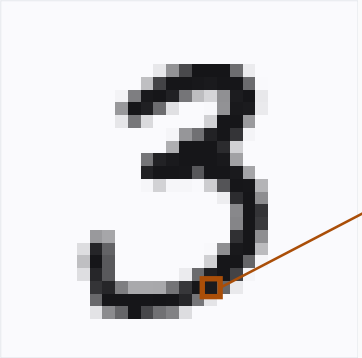
Ho, Jain & Abbeel, "Denoising Diffusion Probabilistic Models" (2020)

▶ Notebook 1.1

One noising step: shrink a little, add a little noise

$$x_t = \sqrt{\alpha_t}\, x_{t-1} + \sqrt{\beta_t}\, \varepsilon_t$$

$$\alpha_t = 1 - \beta_t, \qquad \varepsilon_t \sim \mathcal{N}(0, 1)$$



pixel (22, 16)

its value x_0 = +1: ink

paper is −1, ink is +1

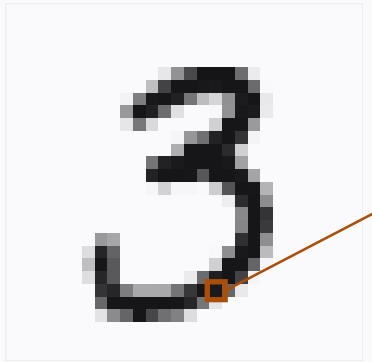
> Our running example: one ink pixel of our 3 (ink is +1, paper is −1). One noising step multiplies the pixel by $\sqrt{\alpha_t}$, a little less than 1, and adds a small Gaussian kick of size $\sqrt{\beta_t}$.

▶ Notebook 1.1

One noising step: shrink a little, add a little noise

$$x_t = \sqrt{\alpha_t}\, x_{t-1} + \sqrt{\beta_t}\, \varepsilon_t$$

$$\alpha_t = 1 - \beta_t, \qquad \varepsilon_t \sim \mathcal{N}(0, 1)$$



pixel (22, 16)

its value x_0 = +1: ink

paper is −1, ink is +1



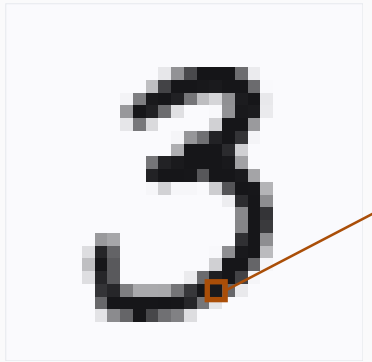
> Toy step 1, with $\beta_1 = 0.1$ and the noise draw $\varepsilon_1 = 0.3$: shrink 1 to 0.949, add 0.095, and get $x_1 = 1.044$.

▶ Notebook 1.1

One noising step: shrink a little, add a little noise

$$x_t = \sqrt{\alpha_t}\, x_{t-1} + \sqrt{\beta_t}\, \varepsilon_t$$

$$\alpha_t = 1 - \beta_t, \qquad \varepsilon_t \sim \mathcal{N}(0, 1)$$



pixel (22, 16)

its value x_0 = +1: ink

paper is −1, ink is +1



β_t is how much noise step t adds. Real models take 1,000 tiny steps; to see every number we first use a toy with five big steps.

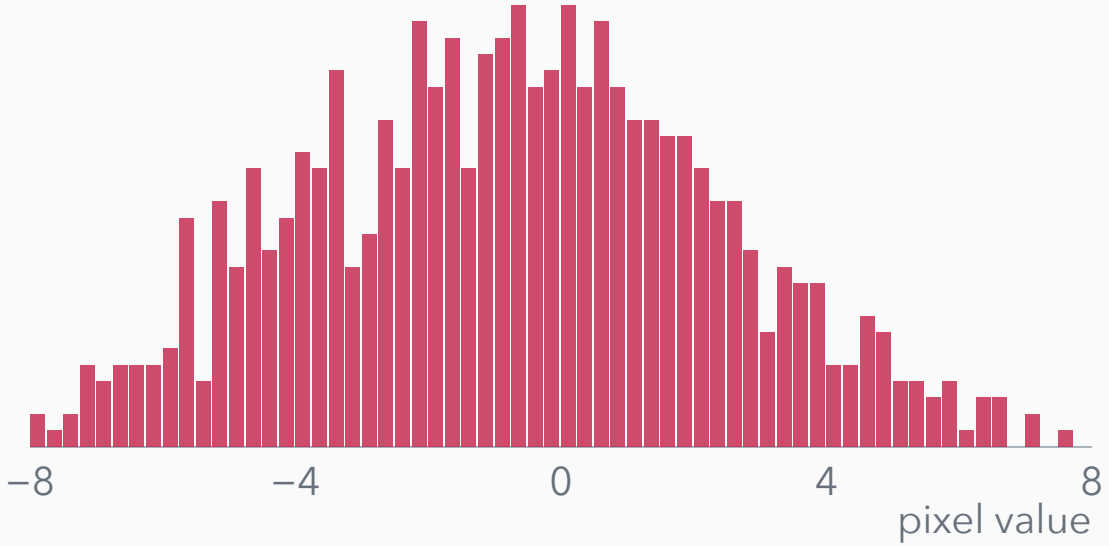
> Repeat with a growing β. But first: why shrink at all?

▶ Notebook 1.1

Keep adding noise: what happens to the spread?

add noise only

spread (sd) = 3.22



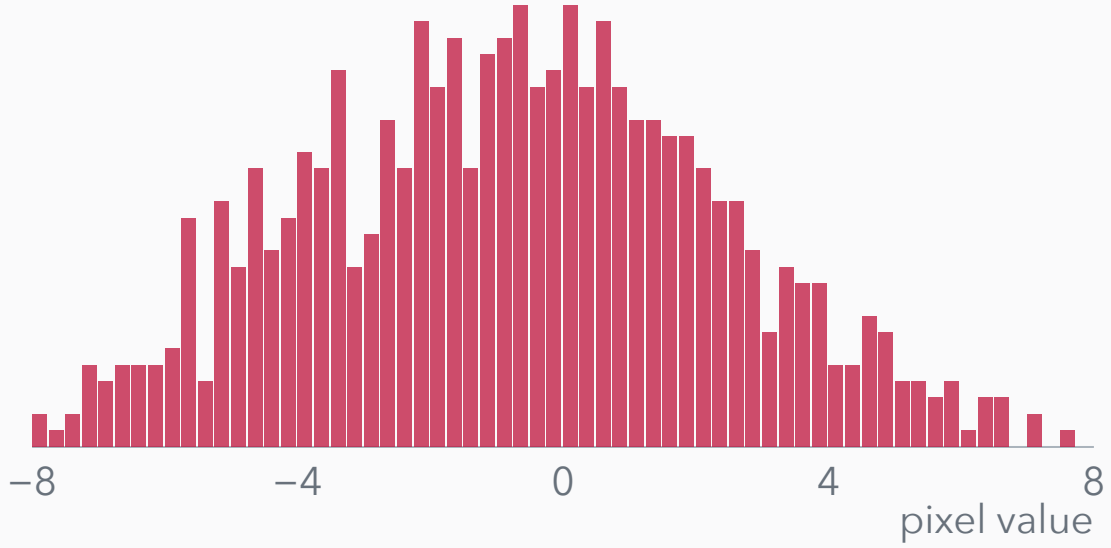
All 784 pixels of our 3, after t steps of adding noise only. Paper and ink blur together, and the spread keeps growing: at t = 1,000 the standard deviation is 3.3, more than three times the image's range.

Deepia, "Diffusion Models: DDPM, Generative AI Animated" (2025)

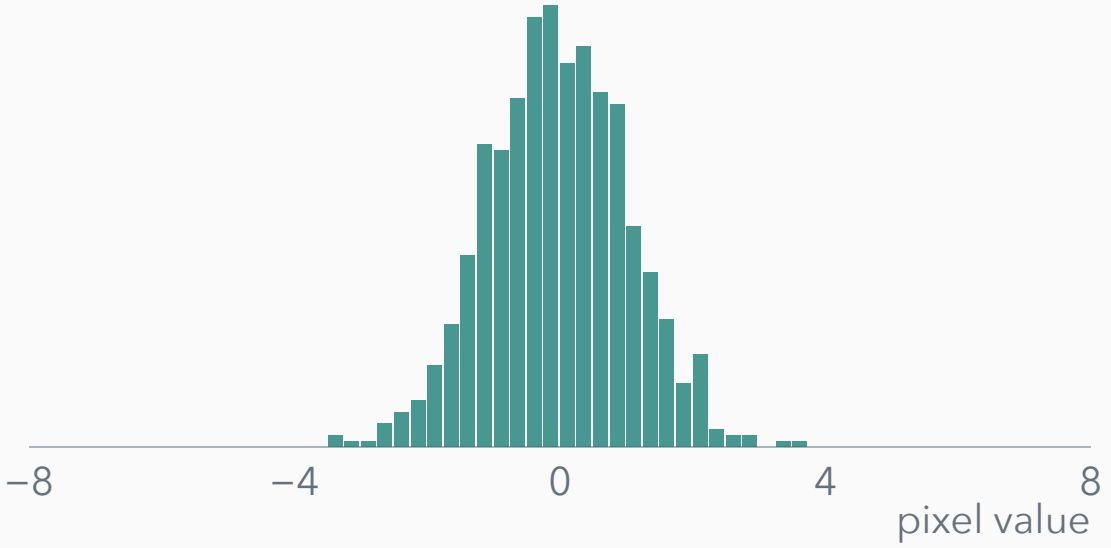
▶ Notebook 1.1

Keep adding noise: what happens to the spread?

add noise only spread (sd) = 3.22



shrink a little, then add spread (sd) = 1.05



> Now shrink a little before adding: multiply by $\sqrt{(1-\beta)}$, then add $\sqrt{\beta}\,\varepsilon$. The spread settles at 1 and stays there.

▶ Notebook 1.1

Shrink a little, then add: the spread stays at 1

$$x_t = \sqrt{1-\beta_t}\, x_{t-1} + \sqrt{\beta_t}\, \varepsilon_t$$

> The real noising step: shrink the previous value by √(1 − β), then add noise of variance β.

Deepia, "Diffusion Models: DDPM, Generative AI Animated" (2025) · Ho, Jain & Abbeel, "Denoising Diffusion Probabilistic Models" (2020)

▶ Notebook 1.1

Shrink a little, then add: the spread stays at 1

$$x_t = \sqrt{1-\beta_t}\, x_{t-1} + \sqrt{\beta_t}\, \varepsilon_t$$

$$\mathrm{Var}(x_t) = (1-\beta_t)\,\mathrm{Var}(x_{t-1}) + \beta_t$$

Rule: Var(aX + bε) = a² Var X + b² Var ε when X and ε are independent, and Var ε = 1.

> Variance of a sum of independent parts: the shrunk part keeps (1 − β) of the old variance; the noise adds β.

Deepia, "Diffusion Models: DDPM, Generative AI Animated" (2025) · Ho, Jain & Abbeel, "Denoising Diffusion Probabilistic Models" (2020)

▶ Notebook 1.1

Shrink a little, then add: the spread stays at 1

$$x_t = \sqrt{1-\beta_t}\, x_{t-1} + \sqrt{\beta_t}\, \varepsilon_t$$

$$\mathrm{Var}(x_t) = (1-\beta_t)\,\mathrm{Var}(x_{t-1}) + \beta_t$$

Rule: Var(aX + bε) = a² Var X + b² Var ε when X and ε are independent, and Var ε = 1.

$$\mathrm{Var}(x_{t-1}) = 1 \;\Rightarrow\; \mathrm{Var}(x_t) = (1-\beta_t) + \beta_t = 1$$

And if the variance is not yet 1, the gap shrinks: $\mathrm{Var}(x_t) - 1 = (1-\beta_t)(\mathrm{Var}(x_{t-1}) - 1)$. Our pixel's first step: $\sqrt{0.9999} \times 1 + 0.01 \times (-0.5) = 0.995$.

> If the variance was 1, it stays exactly 1. If it was not, every step moves it closer to 1.

▶ Notebook 1.2

A toy schedule you can do by hand: five steps

t	β_t	$\alpha_t = 1 - \beta_t$
1	0.1	$1 - 0.1 = 0.9$
2	0.2	$1 - 0.2 = 0.8$
3	0.3	$1 - 0.3 = 0.7$
4	0.5	$1 - 0.5 = 0.5$
5	0.8	$1 - 0.8 = 0.2$

> Five steps with growing β: 0.1, 0.2, 0.3, 0.5, 0.8. Each α is simply 1 − β: the share of the previous value that a step keeps.

▶ Notebook 1.2

A toy schedule you can do by hand: five steps

t	β_t	$\alpha_t = 1 - \beta_t$	$\bar{\alpha}_t = \bar{\alpha}_{t-1} \times \alpha_t$
1	0.1	$1 - 0.1 = 0.9$	$\alpha_1 =$ **0.9**
2	0.2	$1 - 0.2 = 0.8$	$0.9 \times 0.8 =$ **0.72**
3	0.3	$1 - 0.3 = 0.7$	$0.72 \times 0.7 =$ **0.504**
4	0.5	$1 - 0.5 = 0.5$	$0.504 \times 0.5 =$ **0.252**
5	0.8	$1 - 0.8 = 0.2$	$0.252 \times 0.2 =$ **0.0504**

> $\bar{\alpha}_t$ ("alpha-bar") multiplies the α's so far: 0.9, then $0.9 \times 0.8 = 0.72$, then $\times 0.7 = 0.504$, $\times 0.5 = 0.252$, $\times 0.2 = 0.0504$. It measures how much of the original image survives.

▶ Notebook 1.2

A toy schedule you can do by hand: five steps

t	β_t	$\alpha_t = 1 - \beta_t$	$\bar{\alpha}_t = \bar{\alpha}_{t-1} \times \alpha_t$	signal $\sqrt{\bar{\alpha}_t}$	noise $\sqrt{(1 - \bar{\alpha}_t)}$
1	0.1	$1 - 0.1 = 0.9$	$\alpha_1 = \mathbf{0.9}$	0.949	0.316
2	0.2	$1 - 0.2 = 0.8$	$0.9 \times 0.8 = \mathbf{0.72}$	0.849	0.529
3	0.3	$1 - 0.3 = 0.7$	$0.72 \times 0.7 = \mathbf{0.504}$	0.710	0.704
4	0.5	$1 - 0.5 = 0.5$	$0.504 \times 0.5 = \mathbf{0.252}$	0.502	0.865
5	0.8	$1 - 0.8 = 0.2$	$0.252 \times 0.2 = \mathbf{0.0504}$	0.224	0.974

> The two numbers we use all lecture: $\sqrt{\bar{\alpha}_t}$, the size of the signal, and $\sqrt{(1 - \bar{\alpha}_t)}$, the size of the noise. By step 5 they are 0.224 and 0.974: almost pure noise.

▶ Notebook 1.2

Forward, step by step: every number

t	x_{t-1}	ε_t	$\sqrt{a_t} \cdot x_{t-1} + \sqrt{\beta_t} \cdot \varepsilon_t$	x_t
1	1.000	0.3	0.949 × 1.000 + 0.316 × 0.3	**1.044**
2	1.044	−0.5	0.894 × 1.044 + 0.447 × (−0.5)	**0.710**
3				
4				
5				

Start at the ink pixel, $x_0 = 1$. Step 1: 1.044. Step 2 shrinks by $\sqrt{0.8}$ and adds $\sqrt{0.2} \times (-0.5)$: 0.710.

▶ Notebook 1.2

Forward, step by step: every number

t	x_{t-1}	ε_t	$\sqrt{a_t} \cdot x_{t-1} + \sqrt{\beta_t} \cdot \varepsilon_t$	x_t
1	1.000	0.3	0.949 × 1.000 + 0.316 × 0.3	**1.044**
2	1.044	−0.5	0.894 × 1.044 + 0.447 × (−0.5)	**0.710**
3	0.710	0.6	0.837 × 0.710 + 0.548 × 0.6	**0.922**
4	0.922	−1.2	0.707 × 0.922 + 0.707 × (−1.2)	**−0.196**
5				

> Steps 3 and 4 have bigger kicks. After step 4 the ink pixel is negative, −0.196: it looks like paper.

▶ Notebook 1.2

Forward, step by step: every number

t	x_{t-1}	ε_t	$\sqrt{a_t} \cdot x_{t-1} + \sqrt{\beta_t} \cdot \varepsilon_t$	x_t
1	1.000	0.3	0.949 × 1.000 + 0.316 × 0.3	**1.044**
2	1.044	−0.5	0.894 × 1.044 + 0.447 × (−0.5)	**0.710**
3	0.710	0.6	0.837 × 0.710 + 0.548 × 0.6	**0.922**
4	0.922	−1.2	0.707 × 0.922 + 0.707 × (−1.2)	**−0.196**
5	−0.196	0.5	0.447 × (−0.196) + 0.894 × 0.5	**0.359**

From $x_5 = 0.359$ alone you cannot tell the pixel was ink. (A paper pixel with the same draws would give −0.090.)

> Step 5: $x_5 = 0.359$. The original value is almost gone.

▶ Notebook 1.3

Two steps by hand: two noises become one

$$x_2 = \underbrace{\sqrt{\alpha_2\alpha_1}\,x_0}_{\text{signal}} + \underbrace{\sqrt{\alpha_2\beta_1}\,\varepsilon_1 + \sqrt{\beta_2}\,\varepsilon_2}_{\text{two independent Gaussians}}$$

> Training needs millions of noisy examples at random t: must we run every step for each one? Substitute x_1 into the rule for x_2 and open the brackets: a signal part, and two independent noise terms.

Ari Seff, "What are Diffusion Models?" (2022)

▶ Notebook 1.3

Two steps by hand: two noises become one

$$x_2 = \underbrace{\sqrt{\alpha_2\alpha_1}\,x_0}_{\text{signal}} + \underbrace{\sqrt{\alpha_2\beta_1}\,\varepsilon_1 + \sqrt{\beta_2}\,\varepsilon_2}_{\text{two independent Gaussians}}$$

$$\mathrm{Var}(\text{noise}) = \alpha_2\beta_1 + \beta_2 = 1 - \alpha_1\alpha_2$$

> Their variances add (Lecture 1): $\alpha_2\beta_1 + \beta_2$. Since $\beta = 1 - \alpha$, that is exactly $1 - \alpha_1\alpha_2$.

Ari Seff, "What are Diffusion Models?" (2022)

▶ Notebook 1.3

Two steps by hand: two noises become one

$$x_2 = \underbrace{\sqrt{\alpha_2\alpha_1}\,x_0}_{\text{signal}} + \underbrace{\sqrt{\alpha_2\beta_1}\,\varepsilon_1 + \sqrt{\beta_2}\,\varepsilon_2}_{\text{two independent Gaussians}}$$

$$\mathrm{Var}(\text{noise}) = \alpha_2\beta_1 + \beta_2 = 1 - \alpha_1\alpha_2$$

$$x_2 = \sqrt{\alpha_1\alpha_2}\,x_0 + \sqrt{1-\alpha_1\alpha_2}\,\varepsilon, \qquad \varepsilon \sim \mathcal{N}(0,1)$$

> So the two noises are one Gaussian with that variance: $x_2 = \sqrt{(a_1a_2)}\,x_0 + \sqrt{(1 - a_1a_2)}\,\varepsilon$.

Ari Seff, "What are Diffusion Models?" (2022)

▶ Notebook 1.3

Two steps by hand: two noises become one

$$x_2 = \underbrace{\sqrt{\alpha_2\alpha_1}\,x_0}_{\text{signal}} + \underbrace{\sqrt{\alpha_2\beta_1}\,\varepsilon_1 + \sqrt{\beta_2}\,\varepsilon_2}_{\text{two independent Gaussians}}$$

$$\mathrm{Var}(\text{noise}) = \alpha_2\beta_1 + \beta_2 = 1 - \alpha_1\alpha_2$$

$$x_2 = \sqrt{\alpha_1\alpha_2}\,x_0 + \sqrt{1-\alpha_1\alpha_2}\,\varepsilon, \qquad \varepsilon \sim \mathcal{N}(0,1)$$

$$\beta_1 = 0.1,\ \beta_2 = 0.2: \quad x_2 = \sqrt{0.72}\,(1) + \sqrt{0.28}\,(-0.26) = 0.71$$

> Check with two big illustrative steps, $\beta_1 = 0.1$ and $\beta_2 = 0.2$, and the draws $\varepsilon_1 = 0.3$, $\varepsilon_2 = -0.5$: step by step $x_2 = 0.71$, and the formula with the combined $\varepsilon = -0.26$ gives the same.

Ari Seff, "What are Diffusion Models?" (2022)

▶ Notebook 1.3

The shortcut: jump straight to any step

$$x_t = \sqrt{\bar{\alpha}_t}\, x_0 + \sqrt{1-\bar{\alpha}_t}\, \varepsilon, \qquad \bar{\alpha}_t = \alpha_1 \alpha_2 \cdots \alpha_t$$

> The two-step argument works for any t: the signal is shrunk by $\sqrt{\bar{\alpha}_t}$ (the product of all the α's), and all the noises together are one Gaussian with variance $1 - \bar{\alpha}_t$.

▶ Notebook 1.3

The shortcut: jump straight to any step

$$x_t = \sqrt{\bar{\alpha}_t}\, x_0 + \sqrt{1-\bar{\alpha}_t}\, \varepsilon, \qquad \bar{\alpha}_t = \alpha_1 \alpha_2 \cdots \alpha_t$$

$$\text{toy}, t = 2: \quad \sqrt{0.72} \times 1 + \sqrt{0.28} \times (-0.262) = 0.710 \qquad (\text{the same as two steps})$$

> Check on the toy: one jump with the combined draw ε = −0.262 gives x_2 = 0.710, exactly what two steps gave.

▶ Notebook 1.3

The shortcut: jump straight to any step

$$x_t = \sqrt{\bar{\alpha}_t}\, x_0 + \sqrt{1-\bar{\alpha}_t}\, \varepsilon, \qquad \bar{\alpha}_t = \alpha_1 \alpha_2 \cdots \alpha_t$$

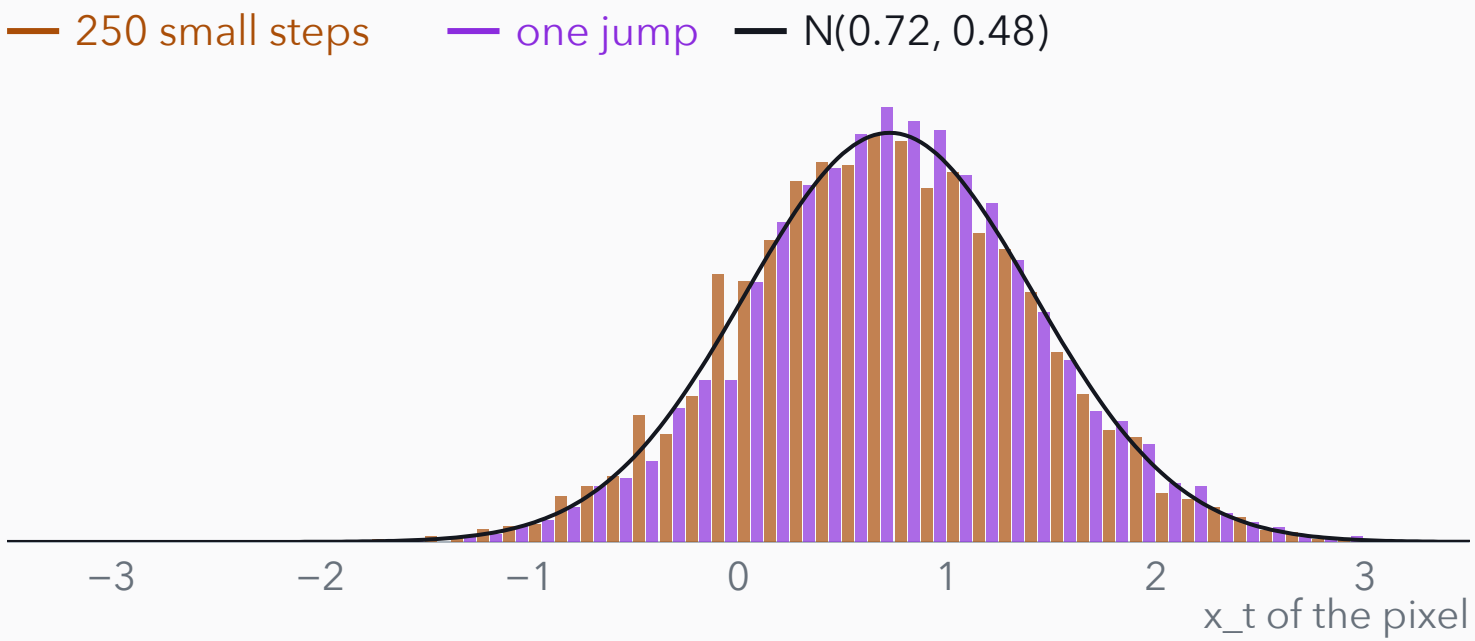
$$\text{toy}, t = 2: \quad \sqrt{0.72} \times 1 + \sqrt{0.28} \times (-0.262) = 0.710 \qquad (\text{the same as two steps})$$

```python
def q_sample(x0, t, eps, alpha_bar):
    a = alpha_bar[t]                          # one number per example
    return a.sqrt() * x0 + (1 - a).sqrt() * eps
```

> In code: one line. Any noise level, any image, in one shot.

▶ Notebook 1.3

Step by step or one jump: the same distribution



t = 250, 4,000 runs each

steps: mean 0.713, sd 0.699

jump: mean 0.735, sd 0.679

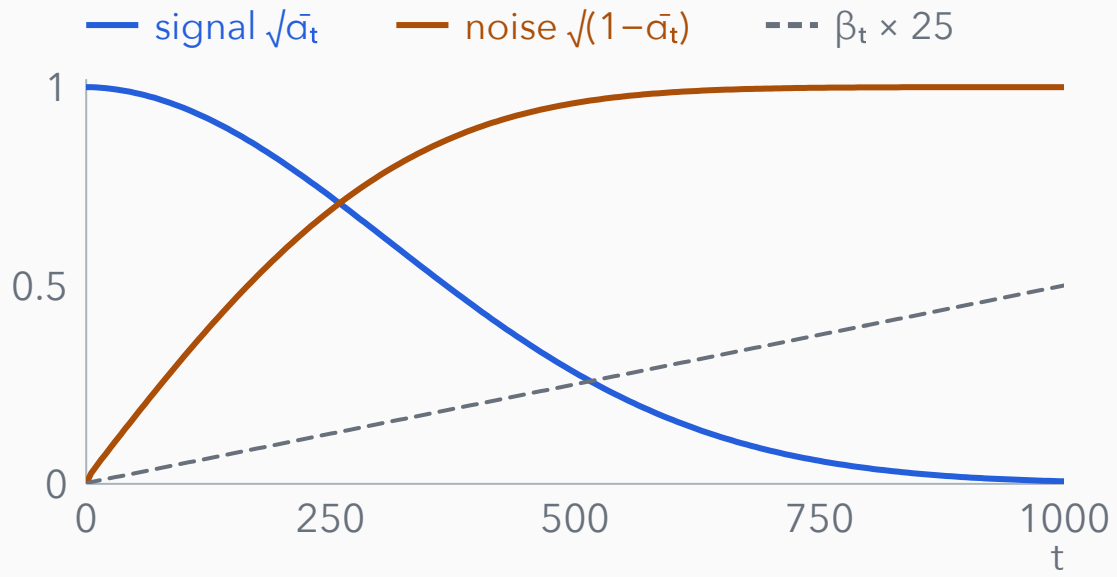
exact: mean 0.724, sd 0.690

250 noise draws per run, or one

> 4,000 runs each way, from our ink pixel: t small steps with their own noise draws, or one jump with one draw. Same histogram, same mean and spread, matching the formula. Move t.

▶ Notebook 1.4

The real schedule: 1,000 small steps

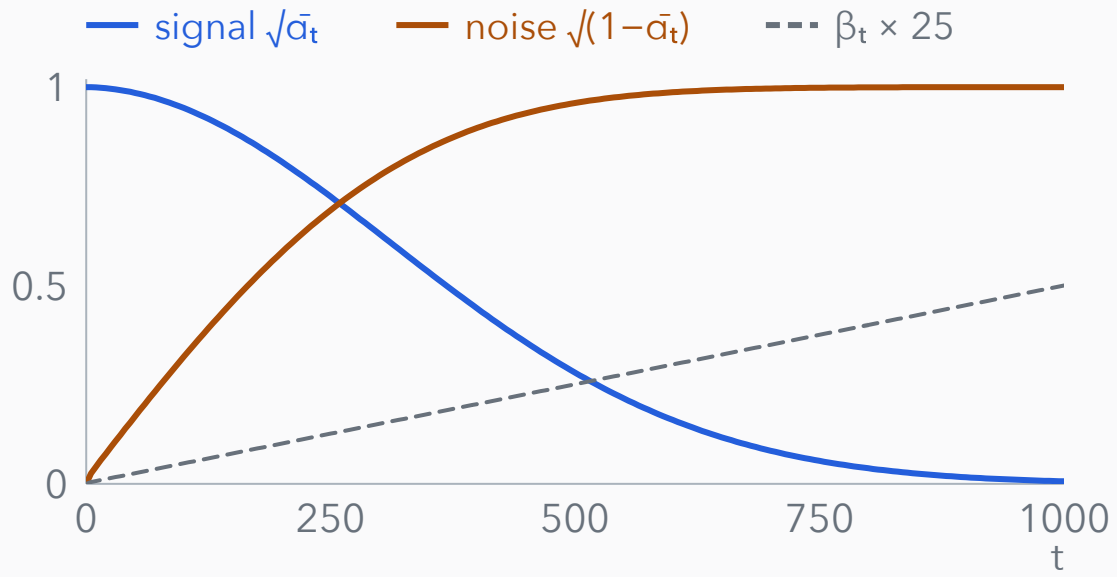


> The schedule of the original DDPM: 1,000 steps, β rising from 0.0001 to 0.02. The signal coefficient $\sqrt{\bar{\alpha}_t}$ falls from 1 to almost 0; the noise coefficient $\sqrt{(1-\bar{\alpha}_t)}$ rises to 1.

Ho, Jain & Abbeel, "Denoising Diffusion Probabilistic Models" (2020) · Nichol & Dhariwal, "Improved Denoising Diffusion Probabilistic Models" (2021)

▶ Notebook 1.4

The real schedule: 1,000 small steps



$$t = 250: \quad \bar{\alpha} = 0.524, \quad x_{250} = 0.724 \times 1 + 0.690 \times (-0.5) = 0.379$$

> Our running example from now on: t = 250, where signal and noise are about equal. The ink pixel with the noise draw ε = −0.5 becomes x_{250} = 0.379.

▶ Notebook 1.4

Try it: every number of a forward jump

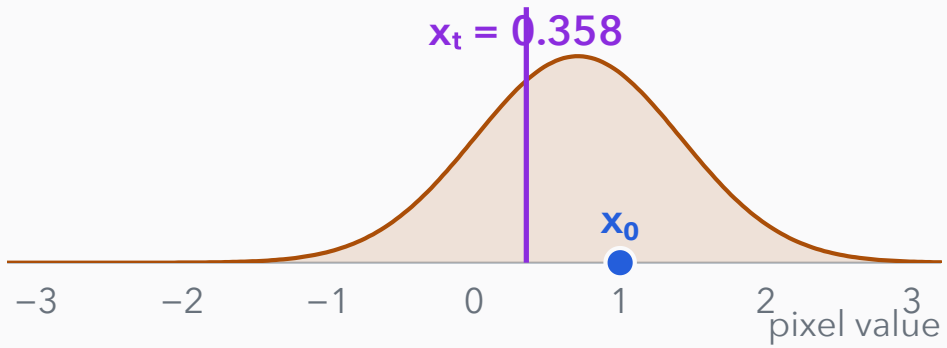
$$\beta_3 = 0.3, \qquad \alpha_3 = 1 - \beta_3 = 0.7$$

$$\bar{\alpha}_3 = \alpha_1 \alpha_2 \alpha_3 = 0.9 \times 0.8 \times 0.7 = 0.5040$$

$$\sqrt{\bar{\alpha}_3} = 0.710, \qquad \sqrt{1 - \bar{\alpha}_3} = 0.704$$

$$x_3 = \sqrt{\bar{\alpha}_3}\, x_0 + \sqrt{1 - \bar{\alpha}_3}\, \varepsilon = 0.710 \times 1.000 + 0.704 \times (-0.500) = \mathbf{0.358}$$

where x_t can land: N(0.71, 0.496)

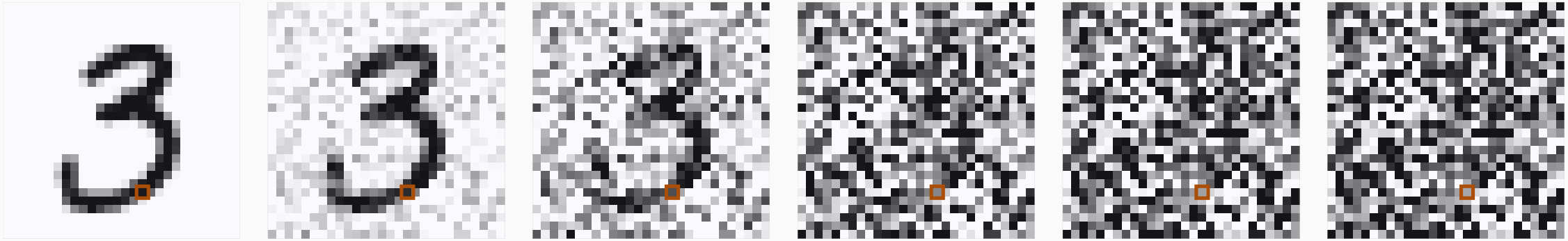


t	β_t	$\alpha_t = 1 - \beta_t$	$\bar{\alpha}_t$	$\sqrt{\bar{\alpha}_t}$	$\sqrt{1 - \bar{\alpha}_t}$
1	0.1	0.9	0.9000	0.949	0.316
2	0.2	0.8	0.7200	0.849	0.529
3	**0.3**	**0.7**	**0.5040**	**0.710**	**0.704**
4	0.5	0.5	0.2520	0.502	0.865
5	0.8	0.2	0.0504	0.224	0.974

Pick the toy or the real schedule, a step t, ink or paper, and a noise draw ε. Every number is shown: β, α, the product ᾱ, the two coefficients and the noisy value, with the bell of all values x_t could take.

▶ Notebook 1.5

Slide t: watch our 3 dissolve



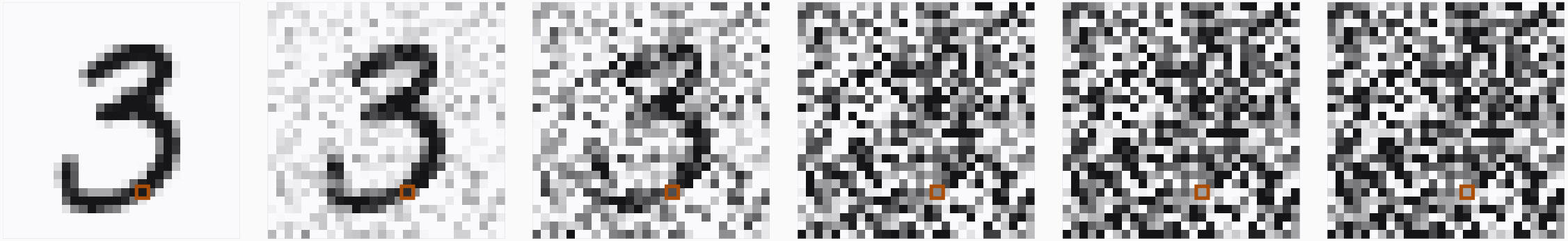
t = 0	t = 100	t = 250	t = 500	t = 750	t = 1000
$\sqrt{\bar{\alpha}} = 1.00$	$\sqrt{\bar{\alpha}} = 0.95$	$\sqrt{\bar{\alpha}} = 0.72$	$\sqrt{\bar{\alpha}} = 0.28$	$\sqrt{\bar{\alpha}} = 0.06$	$\sqrt{\bar{\alpha}} = 0.01$

> Move t. The image is a mix of two things: the signal, our 3 scaled by $\sqrt{\bar{\alpha}_t}$, and noise scaled by $\sqrt{(1 - \bar{\alpha}_t)}$. The pixel we follow is outlined.

Paras Chopra, "Diffusion Models from First Principles" (2025)

▶ Notebook 1.5

Slide t: watch our 3 dissolve



t = 0	t = 100	t = 250	t = 500	t = 750	t = 1000
$\sqrt{\bar{\alpha}} = 1.00$	$\sqrt{\bar{\alpha}} = 0.95$	$\sqrt{\bar{\alpha}} = 0.72$	$\sqrt{\bar{\alpha}} = 0.28$	$\sqrt{\bar{\alpha}} = 0.06$	$\sqrt{\bar{\alpha}} = 0.01$

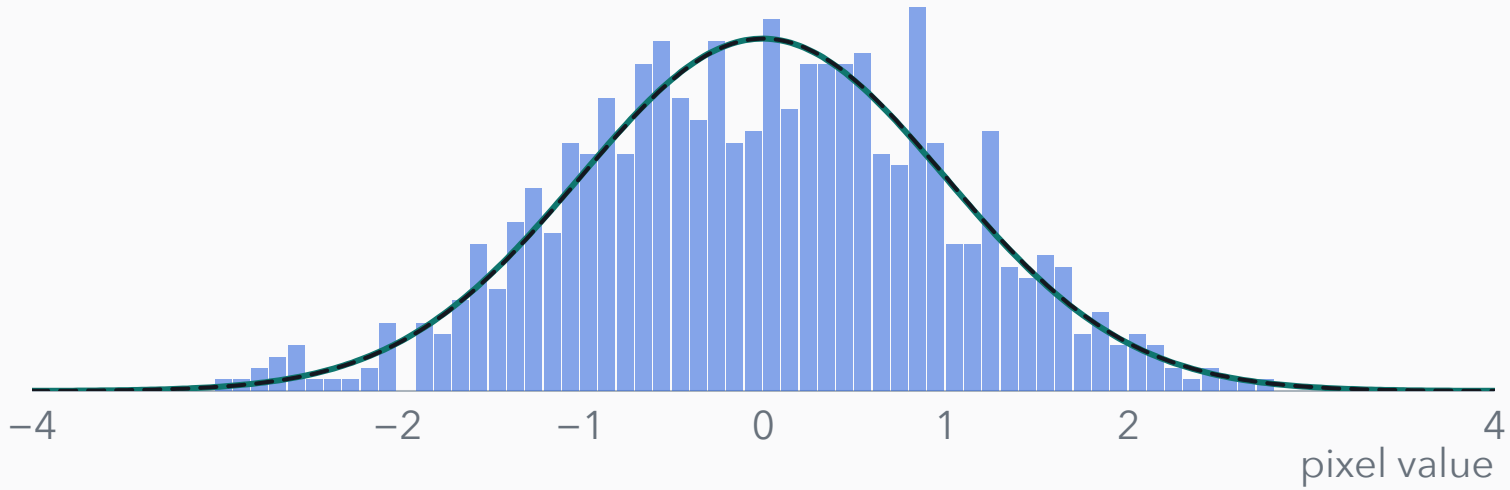
> Around t = 600 the 3 is gone: $\sqrt{\bar{\alpha}}$ has fallen to 0.16. Our ink pixel (+1) reads 0.00 at t = 500: from the value alone you can no longer tell ink from paper.

Paras Chopra, "Diffusion Models from First Principles" (2025)

▶ Notebook 1.5

Every image ends in the same place: N(0, 1)

— our 3: its 784 pixels — all MNIST pixels (exact) --- N(0, 1)



t = 1000

signal $\sqrt{\bar{\alpha}} = 0.006$

noise $\sqrt{(1-\bar{\alpha})} = 1.000$

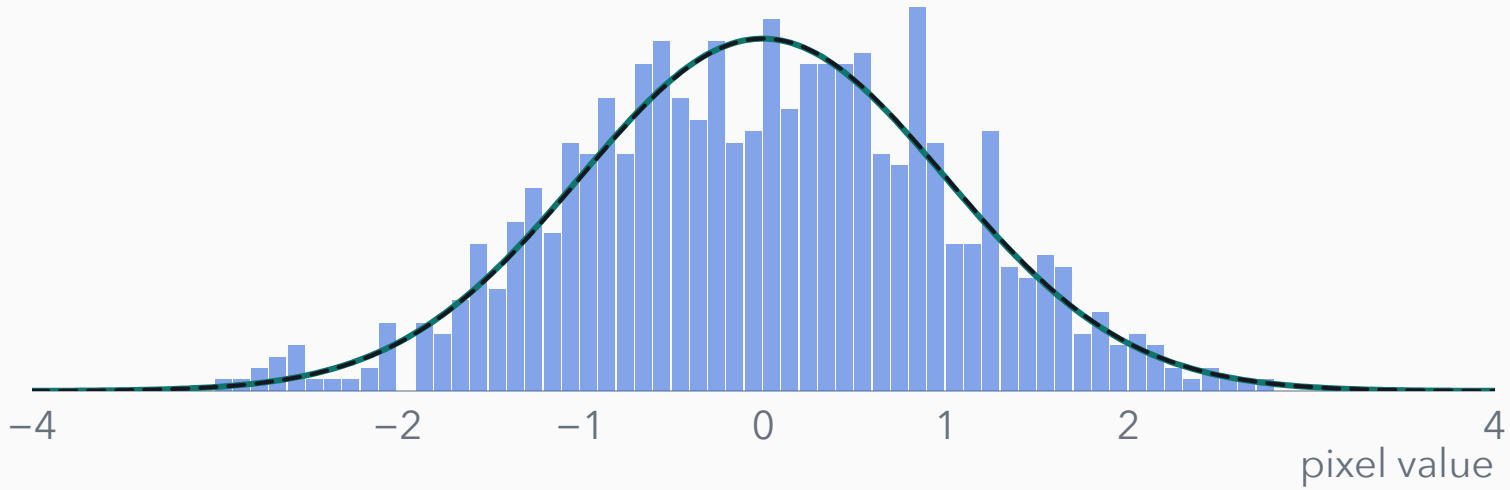
only the noise term is left

> At t = 0 the 784 pixels of our 3 sit at −1 (paper) and +1 (ink). Move t: the two bumps blur and merge, and by t = 1,000 they are the standard bell N(0, 1). Every MNIST pixel does the same (green).

▶ Notebook 1.5

Every image ends in the same place: N(0, 1)

— our 3: its 784 pixels — all MNIST pixels (exact) --- N(0, 1)



t = 1000

signal $\sqrt{\bar{\alpha}} = 0.006$

noise $\sqrt{(1 - \bar{\alpha})} = 1.000$

only the noise term is left

> Why a Gaussian? Not the Central Limit Theorem: the signal part $\sqrt{\bar{\alpha}_t}\, x_0$ is shrunk to zero, and what is left, $\sqrt{(1 - \bar{\alpha}_t)}\, \varepsilon$, is Gaussian because we made it Gaussian.

Paras Chopra, "Diffusion Models from First Principles" (2025) · (endpoint argument corrected)

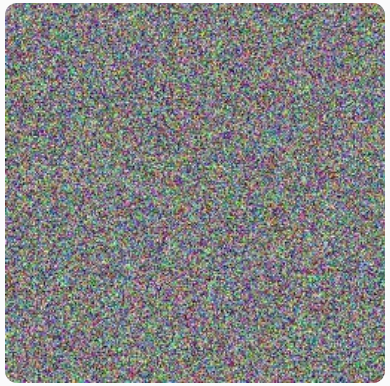
▶ Notebook 1.5

The same jump on a real photograph



the photo x_0

× 0.280 +



one noise draw ε

× 0.960 =



x_t at t = 500

The noise is winning: only the layout survives. $\sqrt{\bar{\alpha}_t} = 0.280$, $\sqrt{(1 - \bar{\alpha}_t)} = 0.960$.

> Nothing in the shortcut is special to digits. A 256 × 256 colour photo is 196,608 numbers; the same two coefficients mix it with one noise image. Move t.

Ho, Jain & Abbeel, "Denoising Diffusion Probabilistic Models" (2020) · Companion film "Diffusion models, from noise to images, step by step" (2026): google/ddpm-cat-256 run on the CC0 photo "Chelsea" (scikit-image)

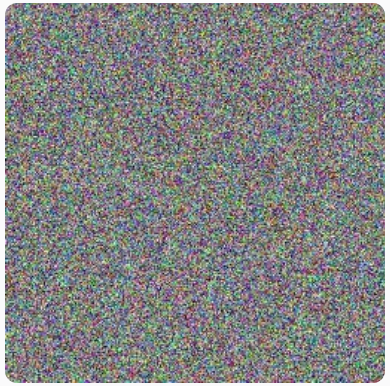
▶ Notebook 1.5

The same jump on a real photograph



the photo x_0

× 0.280 +



one noise draw ε

× 0.960 =



x_t at t = 500

The noise is winning: only the layout survives. $\sqrt{\bar{\alpha}_t} = 0.280$, $\sqrt{(1 - \bar{\alpha}_t)} = 0.960$.

> By t ≈ 600 only the layout is left, light and dark patches; by t = 1,000 the photo is gone. Every photo ends as the same static.

Ho, Jain & Abbeel, "Denoising Diffusion Probabilistic Models" (2020) · Companion film "Diffusion models, from noise to images, step by step" (2026): google/ddpm-cat-256 run on the CC0 photo "Chelsea" (scikit-image)

▶ Notebook 1.5

Destroying is easy, and it hands us labels for free

we have	how
a clean image x_0	the dataset
a noise level t	pick 1 … 1,000 at random
the noise ε	draw N(0, I)
the noisy image x_t	one line: $\sqrt{\bar{\alpha}_t}\, x_0 + \sqrt{(1 - \bar{\alpha}_t)}\, \varepsilon$

> Part 1 in one table. No learning anywhere: every quantity is chosen, drawn or computed.

▶ Notebook 1.5

Destroying is easy, and it hands us labels for free

we have	how
a clean image x_0	the dataset
a noise level t	pick 1 … 1,000 at random
the noise ε	draw N(0, I)
the noisy image x_t	one line: $\sqrt{\bar{\alpha}_t}\, x_0 + \sqrt{(1 - \bar{\alpha}_t)}\, \varepsilon$

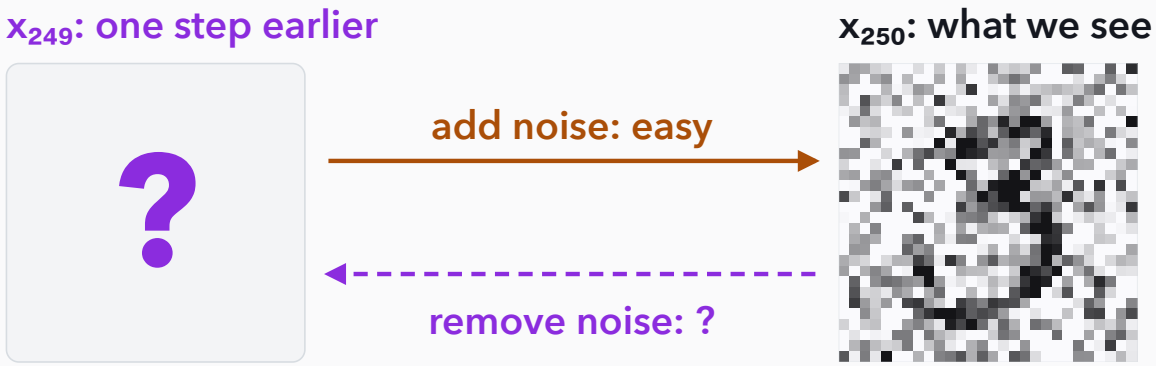
Next: can we go back? Given x_t, where was x_{t-1}?

> The hard direction is the other one. Part 2: the reverse process.

▶ Notebook 2.1

Run the film backwards: where was x_{t-1}?

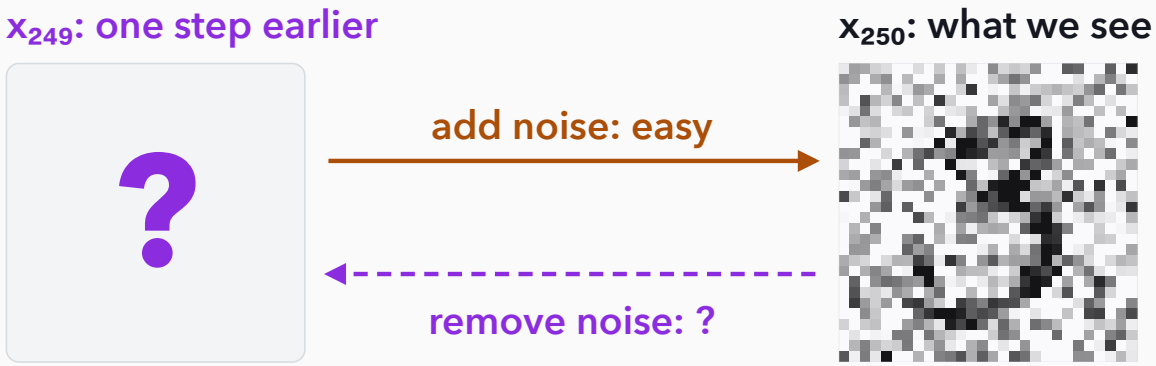
we want: $p(x_{t-1} \mid x_t)$



> Forwards was one line of code. Backwards: we see a noisy image at step t. What did it look like one step earlier? Many images could have produced it.

▶ Notebook 2.1

Run the film backwards: where was x_{t-1}?



we want: $p(x_{t-1} \mid x_t)$

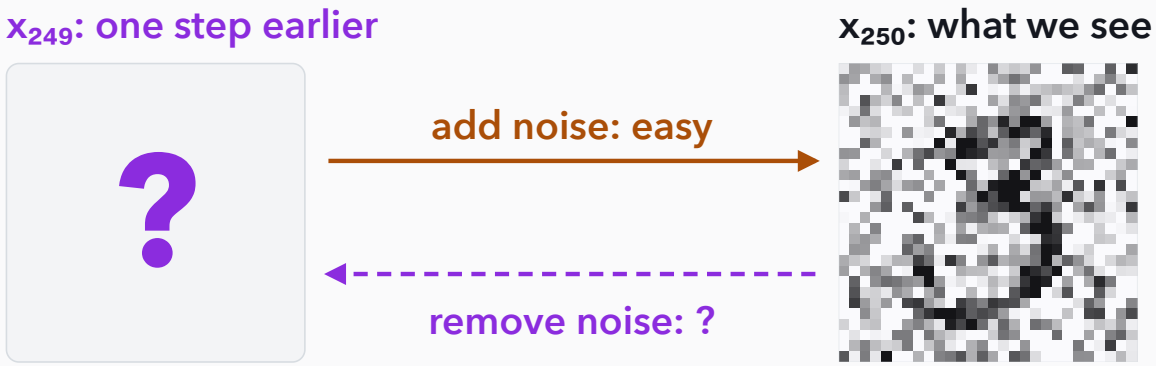
The exact answer needs the distribution of **every possible image**, one step noisier. Nobody has that.

> The exact step back is Bayes' rule, and it needs the distribution of all images at that noise level: as hard as the generative model itself.

Paras Chopra, "Diffusion Models from First Principles" (2025) (V)

▶ Notebook 2.1

Run the film backwards: where was x_{t-1}?



$$\text{we want:} \quad p(x_{t-1} \mid x_t)$$

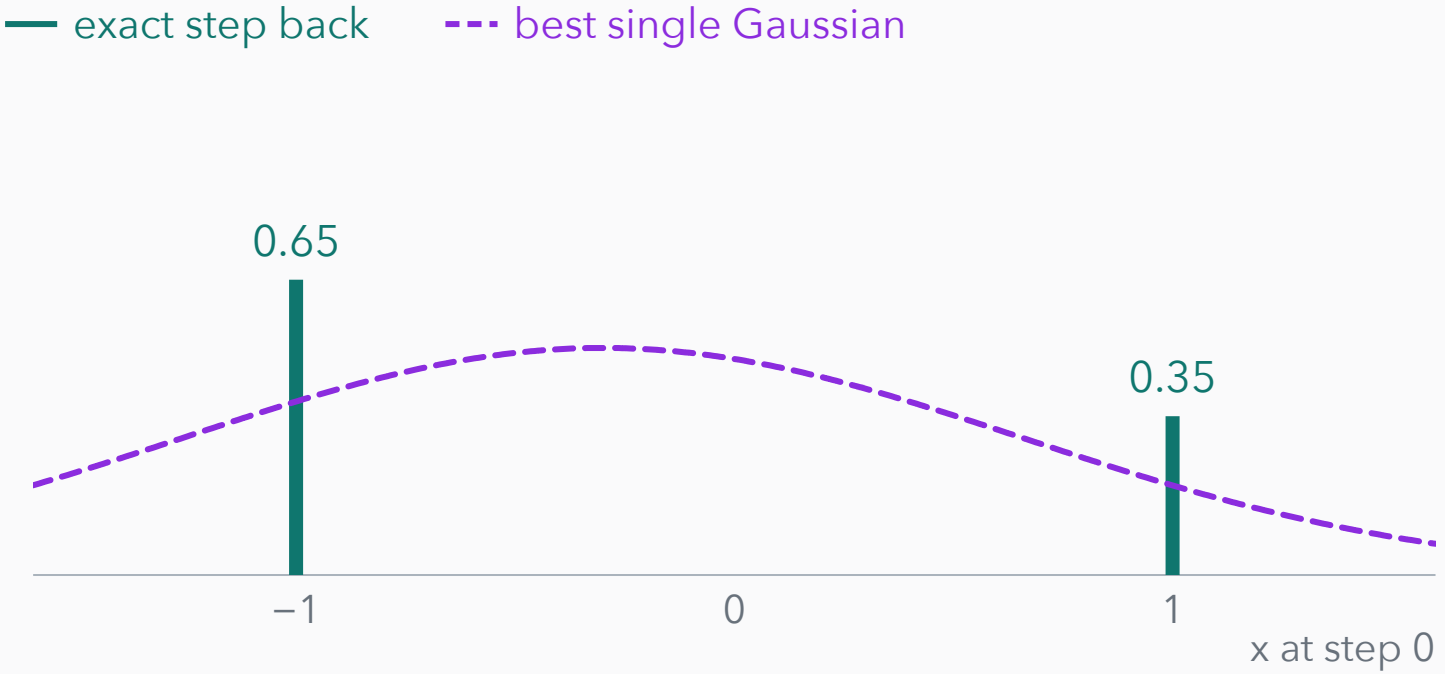
The exact answer needs the distribution of **every possible image**, one step noisier. Nobody has that.

So we **learn** it. When the step is small, the answer is one bell curve: the network only has to find its centre.

> So we learn it, one small step at a time. Next: why a small step back is just a Gaussian, on our one pixel, where we can compute everything exactly.

▶ Notebook 2.2

Shrink the jump: the step back becomes one Gaussian



from t = 500 back to 0

a jump of 500 steps

two spikes: paper or ink

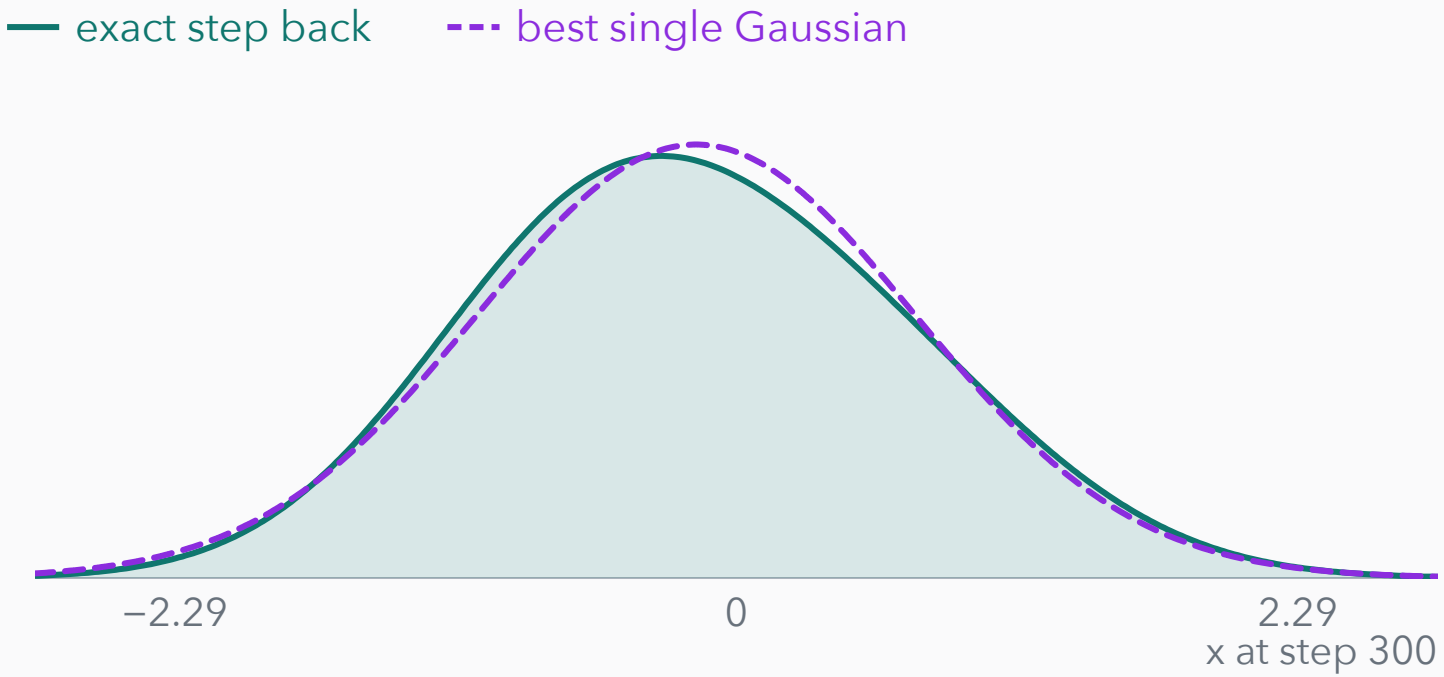
observed x_t = 0.00

> Our pixel is ink in 35% of training digits and paper otherwise: simple enough to compute the step back exactly. From t = 500, given $x_t = 0$, a jump all the way back gives two spikes, ink or paper (green). One Gaussian (purple) fits badly.

Ari Seff, "What are Diffusion Models?" (2022) · (the 1-D picture, here computed exactly)

▶ Notebook 2.2

Shrink the jump: the step back becomes one Gaussian



from t = 500 back to 300

a jump of 200 steps

lopsided: a rough fit

bump centres 1.5 widths apart

observed x_t = 0.00

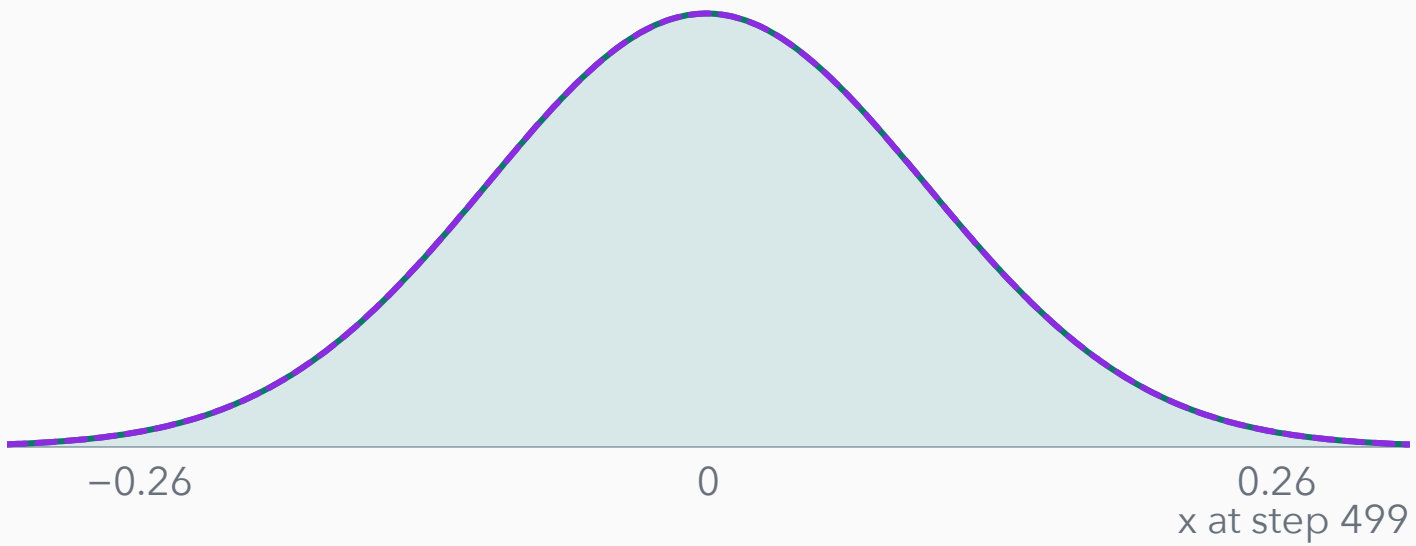
> Back 200 steps, to t = 300: two bumps, 1.5 widths apart. One Gaussian is still a poor fit.

Ari Seff, "What are Diffusion Models?" (2022) · (the 1-D picture, here computed exactly)

▶ Notebook 2.2

Shrink the jump: the step back becomes one Gaussian

— exact step back --- best single Gaussian



from t = 500 back to 499

a jump of 1 step

one bell: a Gaussian fits

bump centres 0.1 widths apart

observed x_t = 0.00

> One step back, 500 → 499: the bumps are 0.06 widths apart, one bell. Press play to shrink the jump continuously.

Ari Seff, "What are Diffusion Models?" (2022) · (the 1-D picture, here computed exactly)

▶ Notebook 2.3

If we knew the noise, we could undo the jump exactly

$$x_t = \sqrt{\bar{\alpha}_t}\, x_0 + \sqrt{1-\bar{\alpha}_t}\, \varepsilon \quad \Longleftrightarrow \quad x_0 = \frac{x_t - \sqrt{1-\bar{\alpha}_t}\, \varepsilon}{\sqrt{\bar{\alpha}_t}}$$

> Solve the shortcut for x_0: if someone told us the noise ε, the clean pixel would follow in one line.

▶ Notebook 2.3

If we knew the noise, we could undo the jump exactly

$$x_t = \sqrt{\bar{\alpha}_t}\, x_0 + \sqrt{1-\bar{\alpha}_t}\, \varepsilon \quad \Longleftrightarrow \quad x_0 = \frac{x_t - \sqrt{1-\bar{\alpha}_t}\, \varepsilon}{\sqrt{\bar{\alpha}_t}}$$

$$\text{toy}, t=2: \quad \frac{0.710 - \sqrt{0.28} \times (-0.262)}{\sqrt{0.72}} = 1.000$$

Check on the toy: $x_2 = 0.710$ with its noise -0.262 gives back $x_0 = 1$ exactly.

▶ Notebook 2.3

If we knew the noise, we could undo the jump exactly

$$x_t = \sqrt{\bar{\alpha}_t}\, x_0 + \sqrt{1-\bar{\alpha}_t}\, \varepsilon \quad \Longleftrightarrow \quad x_0 = \frac{x_t - \sqrt{1-\bar{\alpha}_t}\, \varepsilon}{\sqrt{\bar{\alpha}_t}}$$

$$\text{toy}, t = 2: \quad \frac{0.710 - \sqrt{0.28} \times (-0.262)}{\sqrt{0.72}} = 1.000$$

$$\hat{\varepsilon} = \text{network}(x_t,\, t) \;\approx\; \varepsilon$$

> When generating, nobody tells us ε. So a network guesses it from what it can see: x_t and t. Everything else is arithmetic.

▶ Notebook 2.4

One step back, in one line

$$x_{t-1} = \underbrace{\frac{1}{\sqrt{\alpha_t}}\left(x_t - \frac{\beta_t}{\sqrt{1-\bar{\alpha}_t}}\,\hat{\varepsilon}\right)}_{\mu\text{: a little less noise}} + \underbrace{\sqrt{\beta_t}\,z}_{\text{fresh noise}}, \qquad z \sim \mathcal{N}(0,1)$$

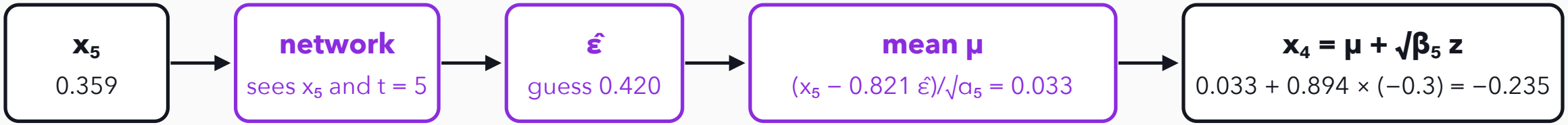
> The reverse step in one line. Inside the bracket: remove a small multiple of the guessed noise. Dividing by $\sqrt{\alpha_t}$ undoes the forward shrink. Then add a little fresh noise.

Ho, Jain & Abbeel, "Denoising Diffusion Probabilistic Models" (2020) (Algorithm 2) · Paras Chopra, "Diffusion Models from First Principles" (2025) (V)

▶ Notebook 2.4

One step back, in one line

$$x_{t-1} = \underbrace{\frac{1}{\sqrt{\alpha_t}}\left(x_t - \frac{\beta_t}{\sqrt{1-\bar{\alpha}_t}}\,\hat{\varepsilon}\right)}_{\mu\text{: a little less noise}} + \underbrace{\sqrt{\beta_t}\,z}_{\text{fresh noise}}, \qquad z \sim \mathcal{N}(0,1)$$



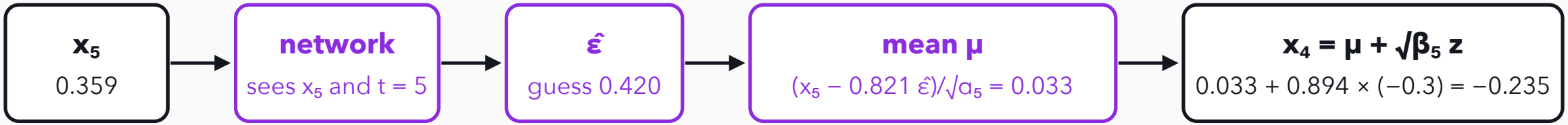
Only the purple box is learned. The fresh noise z is drawn, like ε in the forward step.

> As a picture, toy step 5: from $x_5 = 0.359$ the network guesses $\hat{\varepsilon} = 0.420$; the mean is 0.033; with the fresh draw z = -0.3 we land at $x_4 = -0.235$.

▶ Notebook 2.4

One step back, in one line

$$x_{t-1} = \underbrace{\frac{1}{\sqrt{\alpha_t}}\left(x_t - \frac{\beta_t}{\sqrt{1-\bar{\alpha}_t}}\,\hat{\varepsilon}\right)}_{\mu\text{: a little less noise}} + \underbrace{\sqrt{\beta_t}\,z}_{\text{fresh noise}}, \qquad z \sim \mathcal{N}(0,1)$$



Only the purple box is learned. The fresh noise z is drawn, like ε in the forward step.

> That is the whole generator: start from pure noise and repeat this step from t = T down to 1. Only the network is learned; everything else is the schedule's numbers.

Ho, Jain & Abbeel, "Denoising Diffusion Probabilistic Models" (2020) (Algorithm 2) · Paras Chopra, "Diffusion Models from First Principles" (2025) (V)

▶ Notebook 3.1

Three ways to say the same guess

the network outputs	meaning	our pixel (best guess)
$\hat{\varepsilon}$	the noise: “what is the noise? subtract it”	0.28
$\hat{x}_0$	the clean value: “what is under the noise?”	0.26
$s = \nabla_x \log q_t$	the score: “which way is more likely?” (Part 5)	−0.40

> Three equivalent things a network can output. Knowing one, x_t and t, you can compute the others.

▶ Notebook 3.1

Three ways to say the same guess

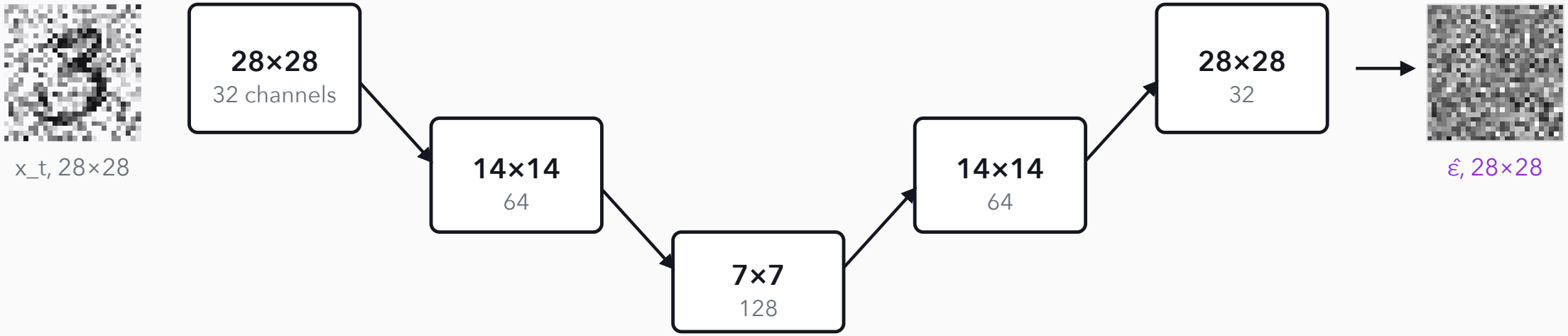
the network outputs	meaning	our pixel (best guess)
$\hat{\varepsilon}$	the noise: "what is the noise? subtract it"	0.28
$\hat{x}_0$	the clean value: "what is under the noise?"	0.26
$s = \nabla_x \log q_t$	the score: "which way is more likely?" (Part 5)	−0.40

Each can be converted into the others with the shortcut. DDPM predicts ε.

> DDPM picks the noise: it has the same scale at every t, so it is an easy target. The network has one job: guess the noise.

▶ Notebook 3.2

For images: a U-Net, so every pixel can see the whole digit

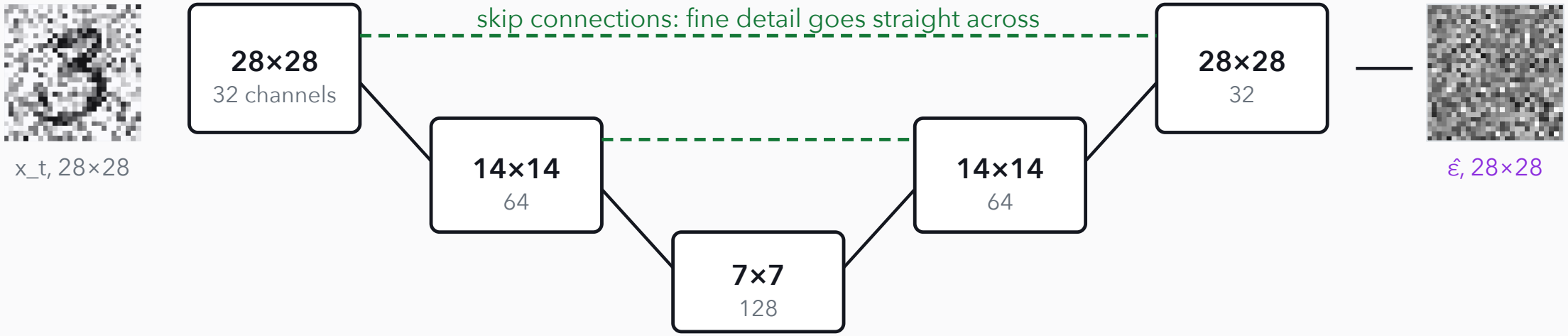


> The output must be the same size as the input: one guessed noise value per pixel. Shrink to 7×7 so that each unit sees the whole digit, then grow back to 28×28.

Ho, Jain & Abbeel, "Denoising Diffusion Probabilistic Models" (2020) · Outlier, "Diffusion Models, Paper Explanation, Math Explained" (2022) (7:01)

▶ Notebook 3.2

For images: a U-Net, so every pixel can see the whole digit

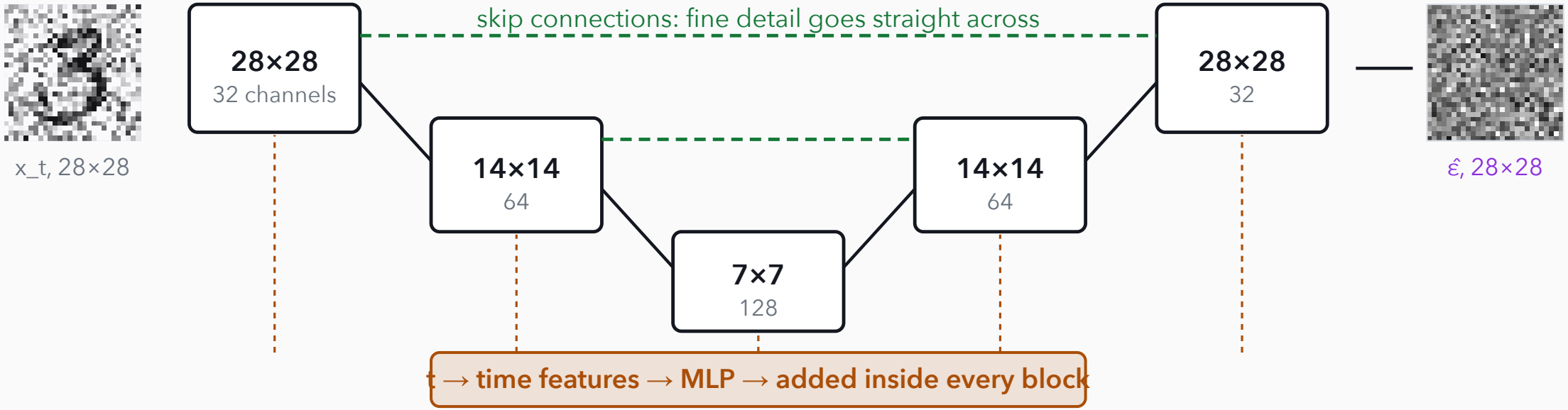


> Skip connections carry fine detail straight across, so the output can be sharp. 814,145 parameters.

Ho, Jain & Abbeel, "Denoising Diffusion Probabilistic Models" (2020) · Outlier, "Diffusion Models, Paper Explanation, Math Explained" (2022) (7:01)

▶ Notebook 3.2

For images: a U-Net, so every pixel can see the whole digit



> The time embedding is added inside every block: the same weights, told how noisy the input is.

▶ Notebook 3.2

One network for every noise level: tell it t

the same input $x_t = 0.38$	best guess of x_0
at t = 50	1.00: ink, surely
at t = 250	0.26: unsure
at t = 900	−0.29: just the prior

> The same pixel value means different things at different noise levels: at t = 50 it is surely ink, at t = 900 it says nothing.

Outlier, "Diffusion Models, Paper Explanation, Math Explained" (2022) (7:01) · MIT 6.S183 "A Practical Introduction to Diffusion Models" (IAP 2025), Lecture 1 (22:21)

▶ Notebook 3.2

One network for every noise level: tell it t

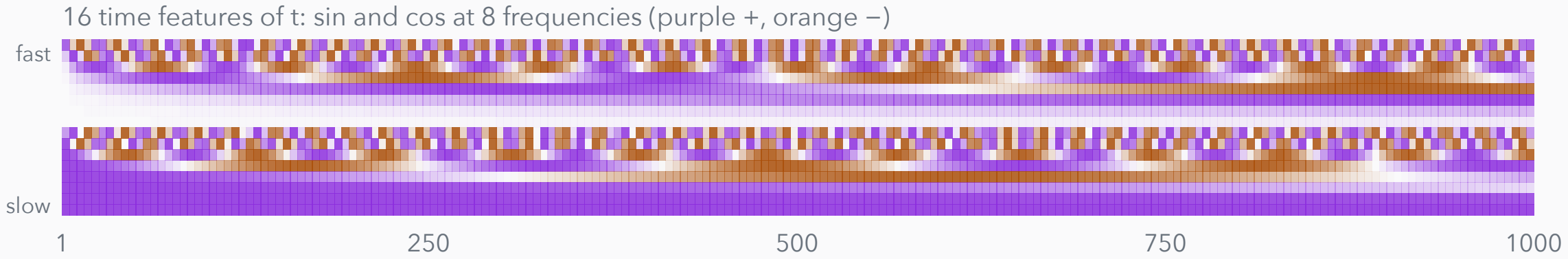
the same input $x_t = 0.38$	best guess of x_0
at t = 50	1.00: ink, surely
at t = 250	0.26: unsure
at t = 900	−0.29: just the prior

> So the network must be told t. One set of weights handles every level; t tells it how much noise to expect.

▶ Notebook 3.2

One network for every noise level: tell it t

the same input $x_t = 0.38$	best guess of x_0
at t = 50	1.00: ink, surely
at t = 250	0.26: unsure
at t = 900	−0.29: just the prior



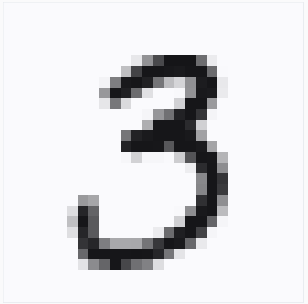
> Like token positions in a transformer, t becomes a vector of sines and cosines at different frequencies, so nearby steps look alike and far ones differ.

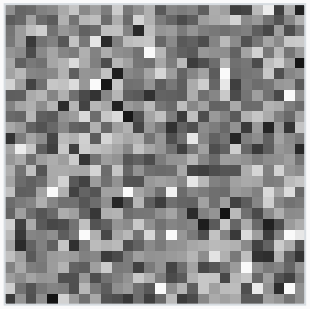
Outlier, "Diffusion Models, Paper Explanation, Math Explained" (2022) (7:01) · MIT 6.S183 "A Practical Introduction to Diffusion Models" (IAP 2025), Lecture 1 (22:21)

▶ Notebook 3.3

A training example costs one line

x_0: a training image $\varepsilon \sim N(0, I)$: fresh noise





t = 250

(a random level)

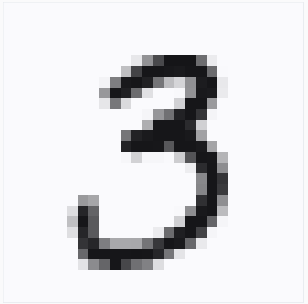
> Ingredients: a training image, a fresh noise image and a random level t.

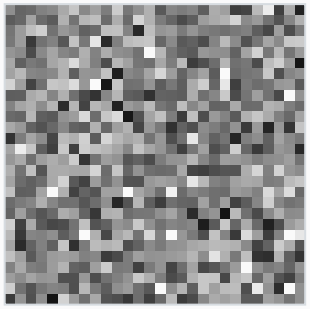
Ho, Jain & Abbeel, "Denoising Diffusion Probabilistic Models" (2020) (Algorithm 1)

▶ Notebook 3.3

A training example costs one line

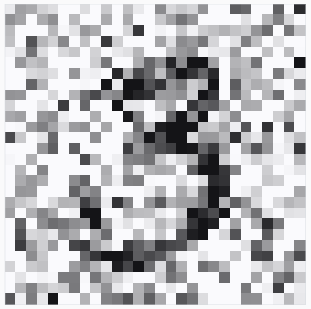
x_0: a training image ε ~ N(0, I): fresh noise $x_t = \sqrt{\bar{\alpha}}\, x_0 + \sqrt{(1-\bar{\alpha})}\, \varepsilon$





t = 250

(a random level)

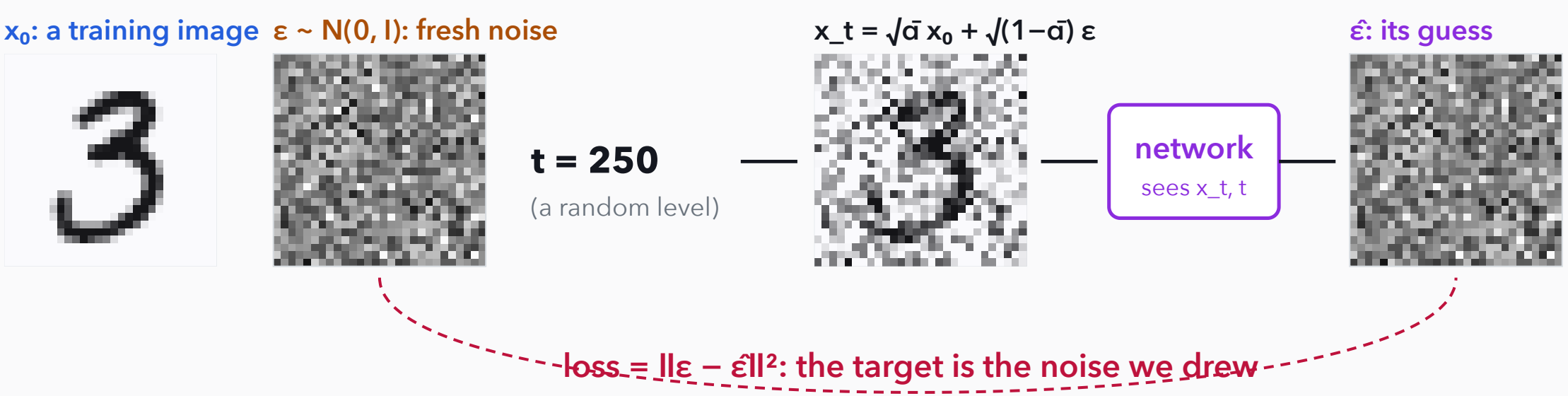


> Make the noisy input with the shortcut: $x_t = \sqrt{\bar{\alpha}_t}\, x_0 + \sqrt{(1 - \bar{\alpha}_t)}\, \varepsilon$.

Ho, Jain & Abbeel, "Denoising Diffusion Probabilistic Models" (2020) (Algorithm 1)

▶ Notebook 3.3

A training example costs one line



> The network sees x_t and t and guesses the noise. The answer is the ε we drew ourselves: labels for free, as many as we like.

Ho, Jain & Abbeel, "Denoising Diffusion Probabilistic Models" (2020) (Algorithm 1)

▶ Notebook 3.3

The loss: squared error between the noise and its guess

$$L(\theta) = \left\| \varepsilon - \hat{\varepsilon}_\theta(x_t, t) \right\|^2 \quad \text{(averaged over pixels and examples)}$$

> The loss: how far the guessed noise is from the noise we drew, squared and averaged.

▶ Notebook 3.3

The loss: squared error between the noise and its guess

$$L(\theta) = \left\| \varepsilon - \hat{\varepsilon}_\theta(x_t, t) \right\|^2 \quad \text{(averaged over pixels and examples)}$$

$$\text{pixel:} \quad (-0.5 - 0.28)^2 = 0.60$$

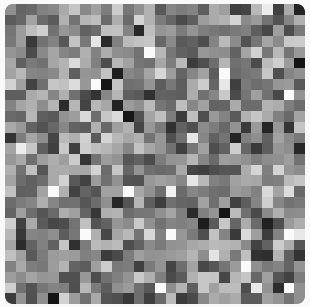
Our pixel with the best possible guess, 0.28: the error is (−0.5 − 0.28)² = 0.60. Even the best guess is wrong for this draw.

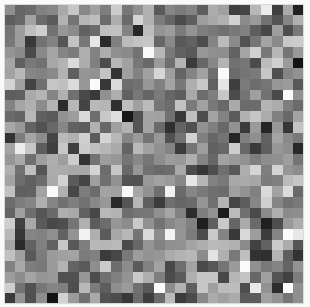
▶ Notebook 3.3

The loss: squared error between the noise and its guess

$$L(\theta) = \left\| \varepsilon - \hat{\varepsilon}_\theta(x_t, t) \right\|^2 \quad \text{(averaged over pixels and examples)}$$

$$\text{pixel:} \quad (-0.5 - 0.28)^2 = 0.60$$





ε: the noise we drew ε̂: the network's guess, t = 250 mean squared error 0.024

> Our 3 at t = 250 through the trained network: the guessed noise looks just like the real one; the average squared error over 784 pixels is 0.024.

▶ Notebook 3.3

The training step in eight lines


```
t   = torch.randint(1, T + 1, (B,))          # a level per image
eps = torch.randn_like(x0)                   # the noise: also the target
a   = alpha_bar[t].view(B, 1, 1, 1)
xt  = a.sqrt() * x0 + (1 - a).sqrt() * eps   # the shortcut
eps_hat = model(xt, t)                       # the guess
loss = ((eps - eps_hat) ** 2).mean()         # squared error
opt.zero_grad(); loss.backward()
opt.step()                                   # one gradient step
```

> Draw a level and a noise image for each training image, jump with the shortcut, guess the noise, take the squared error, and step.

Ho, Jain & Abbeel, "Denoising Diffusion Probabilistic Models" (2020) (Algorithm 1)

▶ Notebook 3.3

The training step in eight lines


```
t   = torch.randint(1, T + 1, (B,))          # a level per image
eps = torch.randn_like(x0)                   # the noise: also the target
a   = alpha_bar[t].view(B, 1, 1, 1)
xt  = a.sqrt() * x0 + (1 - a).sqrt() * eps   # the shortcut
eps_hat = model(xt, t)                       # the guess
loss = ((eps - eps_hat) ** 2).mean()         # squared error
opt.zero_grad(); loss.backward()
opt.step()                                   # one gradient step
```

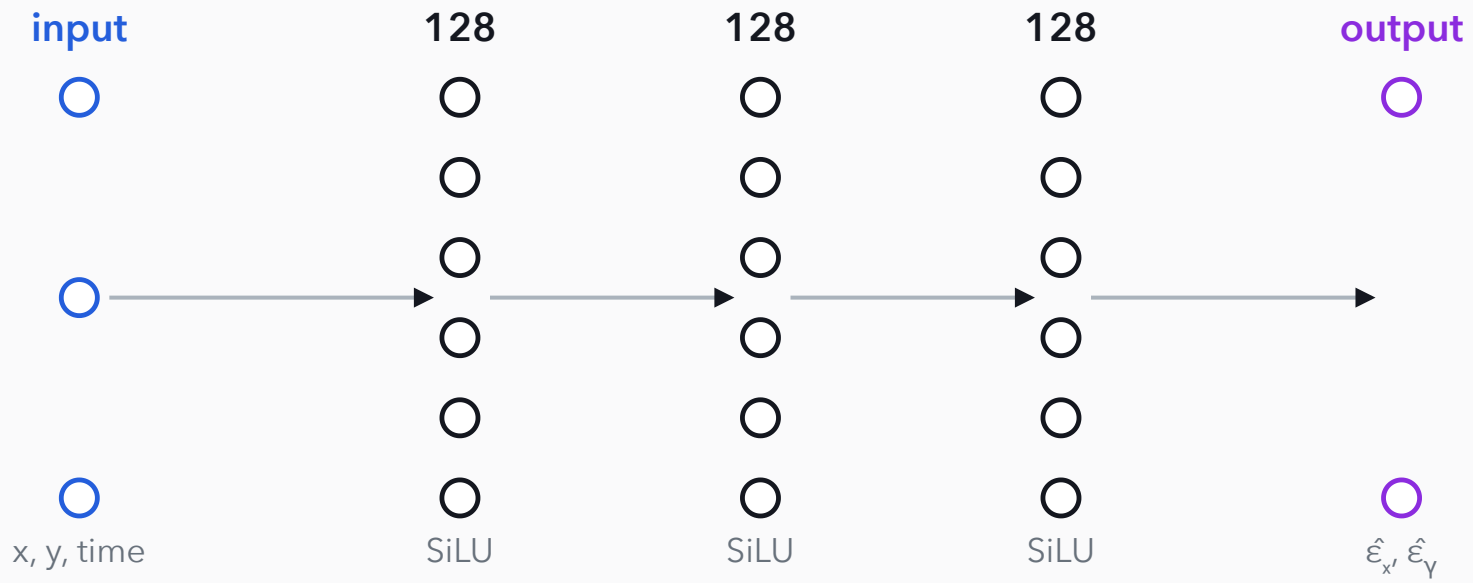
That is all of training: no adversary, no encoder, no sampling. 30,000 such steps trained our MNIST model in about 50 minutes. And the loss is a reweighted ELBO.

> Compare with the GAN of Lecture 3: one network, one loss, no game. And the loss is not arbitrary: it is the ELBO of Lectures 1 and 3 with its weights set to 1. Diffusion pays later, at sampling time.

Ho, Jain & Abbeel, "Denoising Diffusion Probabilistic Models" (2020) (Algorithm 1)

▶ Notebook 3.4

For points in 2-D: a small MLP



the data: 2-D points



Our 2-D example: points on a smiley face. 35,714 parameters; the time features are simply concatenated to the input.

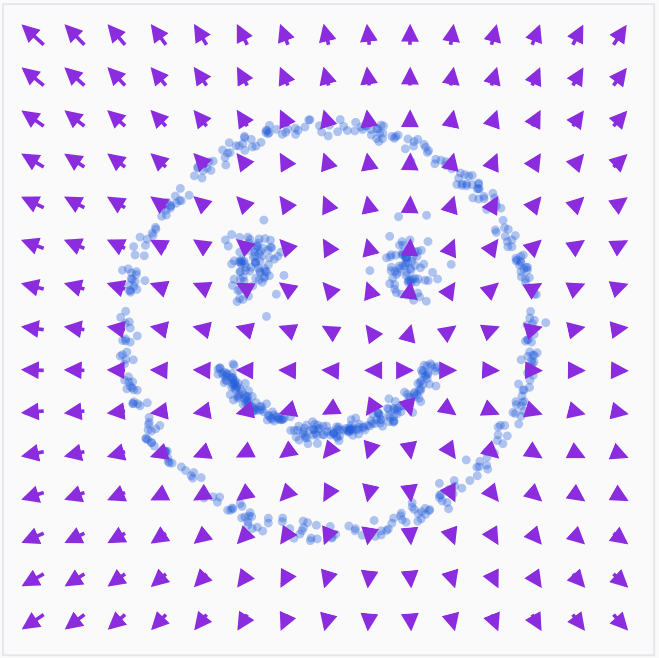
> Before images, a dataset we can see completely: points on a smiley face. The network: the point (x, y) and the 16 time features in, three layers of 128, two numbers out: the guessed noise.

▶ Notebook 3.4

Press play: the network learns the smiley

1 · Forward | 2 · Reverse | **3 · Train** | 4 · Sample | 5 · Score | 6 · Faster | 7 · Steer | 8 · Wrap-up

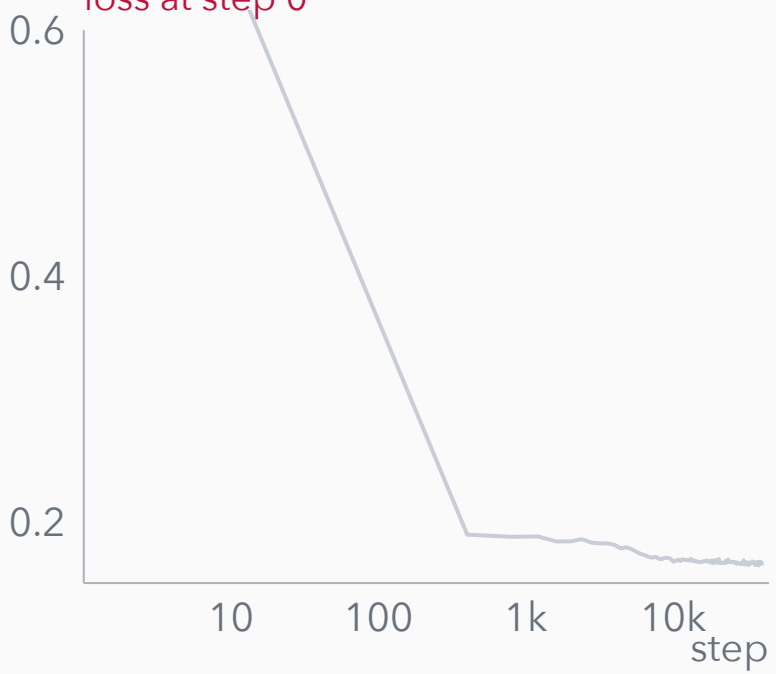
arrows: $x_t \rightarrow$ the network's $\hat{x}_0$ (t = 100)



300 samples from this network

300 of 300 land off this chart

loss at step 0



> Before training: arrows from each grid point to the network's guess of the clean point (at t = 100) point nowhere useful, and its samples are a blob.

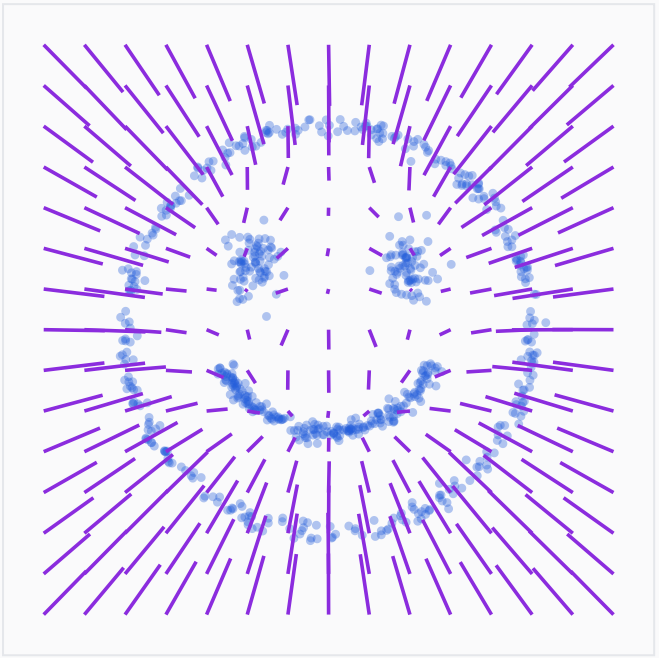
Paras Chopra, "Diffusion Models from First Principles" (2025) ("Training Playground", here with a real network)

▶ Notebook 3.4

Press play: the network learns the smiley

1 · Forward | 2 · Reverse | **3 · Train** | 4 · Sample | 5 · Score | 6 · Faster | 7 · Steer | 8 · Wrap-up

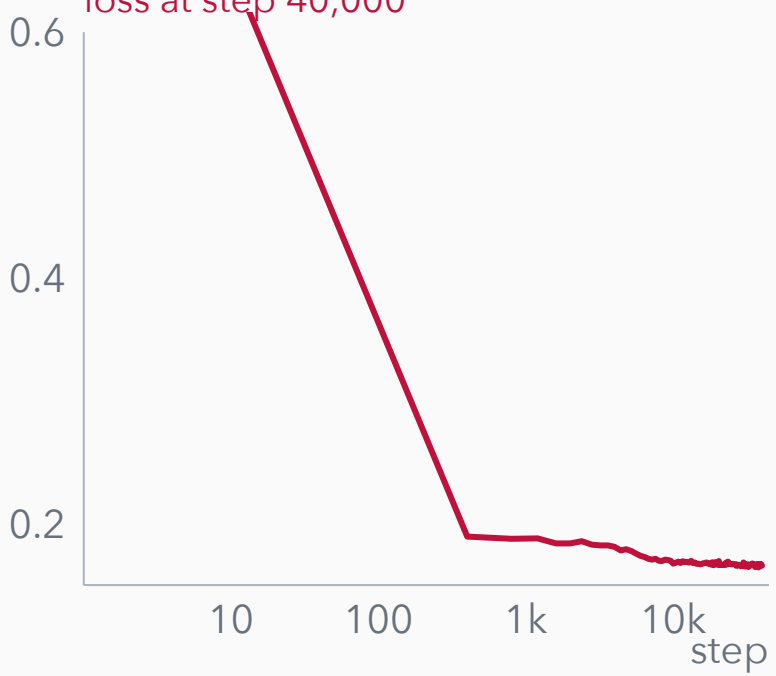
arrows: $x_t \rightarrow$ the network's $\hat{x}_0$ (t = 100)



300 samples from this network



loss at step 40,000

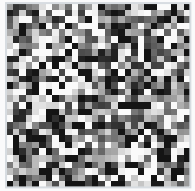


> Press play: as the loss falls the arrows turn towards the face, and the samples, drawn with Part 4's loop, become a smiley.

Paras Chopra, "Diffusion Models from First Principles" (2025) ("Training Playground", here with a real network)

▶ Notebook 3.5

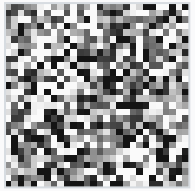
What the network believes our 3 is, at each noise level



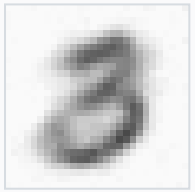
t = 1000



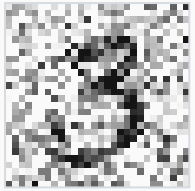
$\hat{x}_0$ error 0.756



t = 600



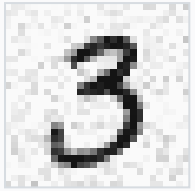
$\hat{x}_0$ error 0.154



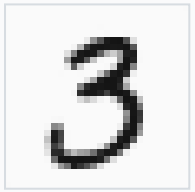
t = 250



$\hat{x}_0$ error 0.021



t = 50

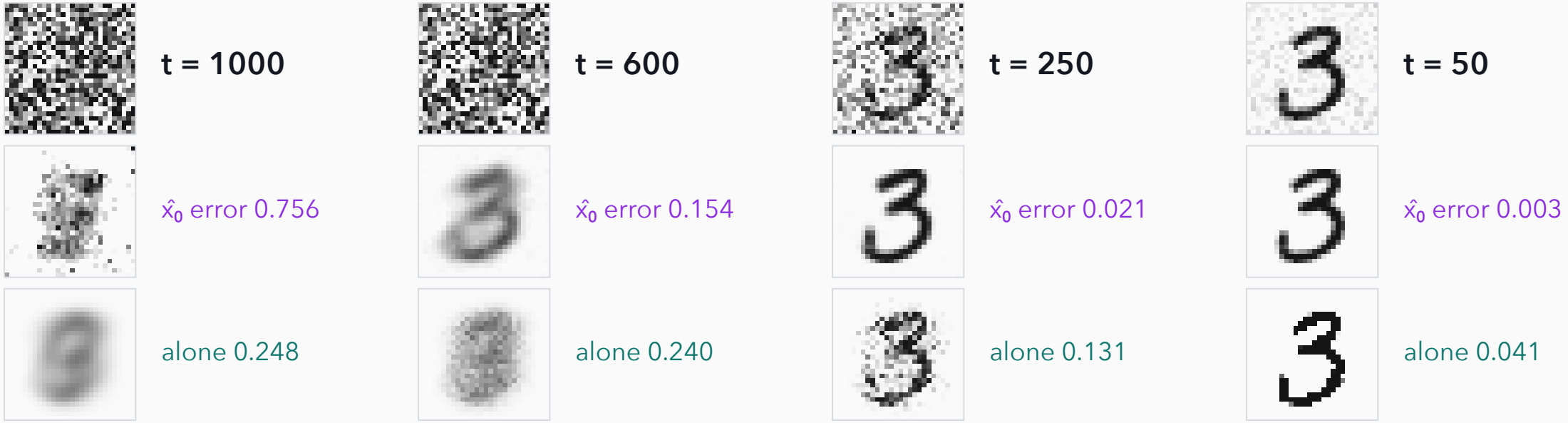


$\hat{x}_0$ error 0.003

> Our 3 at t = 250, its predicted noise, and the clean image that guess implies. Move the slider: at t = 1,000 the guess is a blurry average digit; by t = 500 it is clearly a 3; at small t it is exactly our 3.

▶ Notebook 3.5

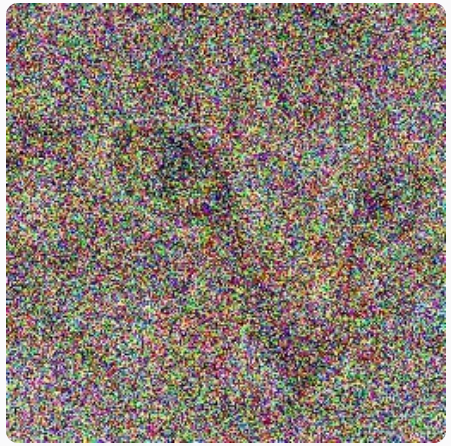
What the network believes our 3 is, at each noise level



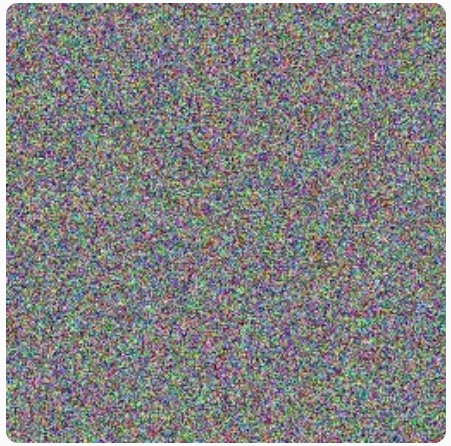
> Each pixel guessed on its own (the best one-pixel guess, from optional depth) stays a grey blur much longer. The network's guesses are far better.

▶ Notebook 3.5

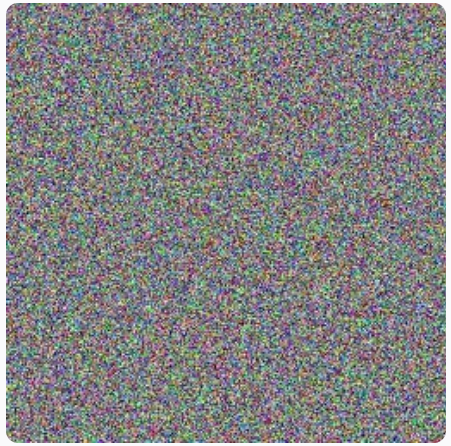
The original DDPM network on a real photograph



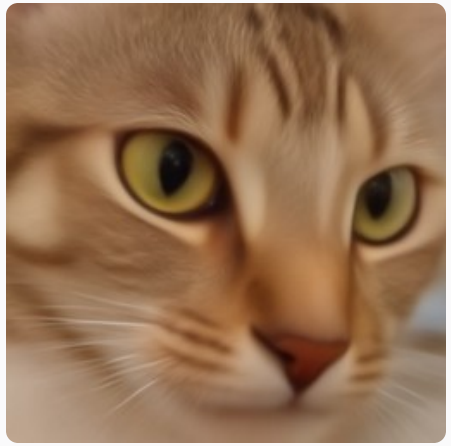
x_{400}: the input



ε: the noise we added



$\hat{\varepsilon}$: the network's guess



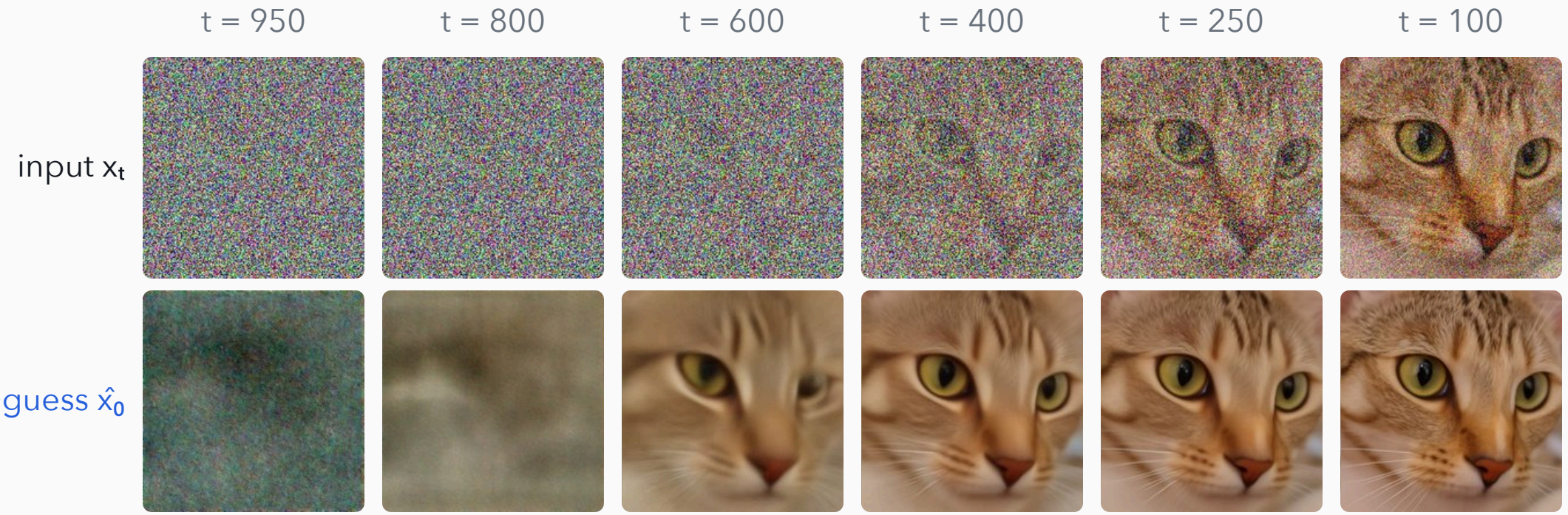
$\hat{x}_0$: the photo it implies

> The network of the original DDPM paper, given our photo at t = 400. Its guess of the noise (third) is almost exactly the noise we added (second): correlation 0.998. Subtract it, and the photo comes back (right).

Ho, Jain & Abbeel, "Denoising Diffusion Probabilistic Models" (2020) · Companion film "Diffusion models, from noise to images, step by step" (2026): google/ddpm-cat-256 run on the CC0 photo "Chelsea" (scikit-image)

▶ Notebook 3.5

The original DDPM network on a real photograph



The same at other noise levels. At t = 950 its guess is a grey blur; by t = 600, a blurry cat; by t = 250, this cat. The same story as our 3, on a photo.

Ho, Jain & Abbeel, "Denoising Diffusion Probabilistic Models" (2020) · Companion film "Diffusion models, from noise to images, step by step" (2026): google/ddpm-cat-256 run on the CC0 photo "Chelsea" (scikit-image)

▶ Notebook 3.5

Training: a regression problem we made ourselves

step	what happens
draw	an image x_0, a level t, a noise ε
make	x_t with the shortcut
guess	$\hat{\varepsilon} = \text{network}(x_t, t)$
learn	minimise $\|\varepsilon - \hat{\varepsilon}\|^2$: the network learns $E[\varepsilon \mid x_t]$, the best average guess

> Part 3 in four rows: draw, make, guess, learn. The network ends up predicting the average noise consistent with what it sees.

▶ Notebook 3.5

Training: a regression problem we made ourselves

step	what happens
draw	an image x_0, a level t, a noise ε
make	x_t with the shortcut
guess	$\hat{\varepsilon} = \text{network}(x_t, t)$
learn	minimise $\|\varepsilon - \hat{\varepsilon}\|^2$: the network learns $E[\varepsilon \mid x_t]$, the best average guess

Next: generate. Start from pure noise and use the guesses to walk back, one small Gaussian step at a time.

> Part 4: put the guess into Part 2's step back, and repeat a thousand times.

▶ Notebook 4.1

One step back, by hand, on the real schedule

$$\hat{\varepsilon} = \text{network}(x_{250} = 0.379,\ t = 250) = 0.276$$

> Our pixel at t = 250, value 0.379. Suppose the trained network says the noise is 0.28 (the best possible guess for this pixel).

Ho, Jain & Abbeel, "Denoising Diffusion Probabilistic Models" (2020) (Algorithm 2)

▶ Notebook 4.1

One step back, by hand, on the real schedule

$$\hat{\varepsilon} = \text{network}(x_{250} = 0.379,\ t = 250) = 0.276$$

$$\mu = \frac{1}{\sqrt{0.9949}}\left(0.379 - \frac{0.0051}{0.690} \times 0.276\right) = 0.378$$

> The mean: remove 0.0051/0.690 of the guessed noise and undo the shrink: 0.378. One step moves the pixel very little.

Ho, Jain & Abbeel, "Denoising Diffusion Probabilistic Models" (2020) (Algorithm 2)

▶ Notebook 4.1

One step back, by hand, on the real schedule

$$\hat{\varepsilon} = \text{network}(x_{250} = 0.379,\ t = 250) = 0.276$$

$$\mu = \frac{1}{\sqrt{0.9949}}\left(0.379 - \frac{0.0051}{0.690} \times 0.276\right) = 0.378$$

$$x_{249} = \mu + \sqrt{\beta_{250}}\, z = 0.378 + 0.071 \times 0.8 = 0.435$$

Add fresh noise of size $\sqrt{\beta} = 0.071$: with $z = 0.8$, $x_{249} = 0.435$. One step done; 249 to go.

Ho, Jain & Abbeel, "Denoising Diffusion Probabilistic Models" (2020) (Algorithm 2)

▶ Notebook 4.1

One step back, by hand, on the real schedule

$$\hat{\varepsilon} = \text{network}(x_{250} = 0.379,\ t = 250) = 0.276$$

$$\mu = \frac{1}{\sqrt{0.9949}}\left(0.379 - \frac{0.0051}{0.690} \times 0.276\right) = 0.378$$

$$x_{249} = \mu + \sqrt{\beta_{250}}\, z = 0.378 + 0.071 \times 0.8 = 0.435$$

$$\text{the network's guess of the clean pixel: } \hat{x}_0 = \frac{0.379 - 0.690 \times 0.276}{0.724} = 0.26$$

> Along the way we can read off the network's guess of the clean pixel: 0.26. Probably ink, not sure yet.

Ho, Jain & Abbeel, "Denoising Diffusion Probabilistic Models" (2020) (Algorithm 2)

▶ Notebook 4.2

The toy, all the way back: every number

t	x_t	$\hat{\varepsilon}$ (network)	mean $\mu = (x_t - c_t\,\hat{\varepsilon}) / \sqrt{\alpha_t}$	fresh noise $\sqrt{\beta_t}\,z$	x_{t-1}
5	0.359	0.420	$(0.359 - 0.821 \times 0.420) / 0.447 = 0.033$	$0.894 \times (-0.3)$	**−0.235**
4					
3					
2					
1					

$c_t = \beta_t/\sqrt{(1 - \bar{\alpha}_t)}$. $\hat{\varepsilon}$ is the best possible guess for this one-pixel dataset (optional depth shows how to compute it).

> Start from $x_5 = 0.359$, where our forward run ended. Step 5: the network guesses 0.420, the mean is 0.033, a big fresh kick $0.894 \times (-0.3)$: $x_4 = -0.235$.

▶ Notebook 4.2

The toy, all the way back: every number

t	x_t	$\hat{\varepsilon}$ (network)	mean $\mu = (x_t - c_t\,\hat{\varepsilon}) / \sqrt{\alpha_t}$	fresh noise $\sqrt{\beta_t}\,z$	x_{t-1}
5	0.359	0.420	$(0.359 - 0.821 \times 0.420) / 0.447 = 0.033$	$0.894 \times (-0.3)$	**−0.235**
4	−0.235	−0.019	$(-0.235 - 0.578 \times (-0.019)) / 0.707 = -0.318$	0.707×0.5	**0.036**
3					
2					
1					

$c_t = \beta_t/\sqrt{(1 - \bar{\alpha}_t)}$. $\hat{\varepsilon}$ is the best possible guess for this one-pixel dataset (optional depth shows how to compute it).

> Step 4: guess −0.019, mean −0.318, kick 0.707×0.5: $x_3 = 0.036$.

▶ Notebook 4.2

The toy, all the way back: every number

t	x_t	$\hat{\varepsilon}$ (network)	mean $\mu = (x_t - c_t\,\hat{\varepsilon}) / \sqrt{\alpha_t}$	fresh noise $\sqrt{\beta_t}\,z$	x_{t-1}
5	0.359	0.420	$(0.359 - 0.821 \times 0.420) / 0.447 = 0.033$	$0.894 \times (-0.3)$	**−0.235**
4	−0.235	−0.019	$(-0.235 - 0.578 \times (-0.019)) / 0.707 = -0.318$	0.707×0.5	**0.036**
3	0.036	0.306	$(0.036 - 0.426 \times 0.306) / 0.837 = -0.113$	0.548×0.2	**−0.003**
2					
1					

$c_t = \beta_t / \sqrt{(1 - \bar{\alpha}_t)}$. $\hat{\varepsilon}$ is the best possible guess for this one-pixel dataset (optional depth shows how to compute it).

> Step 3: $x_2 = -0.003$. The pixel hovers near 0: still undecided.

▶ Notebook 4.2

The toy, all the way back: every number

t	x_t	$\hat{\varepsilon}$ (network)	mean $\mu = (x_t - c_t\,\hat{\varepsilon}) / \sqrt{\alpha_t}$	fresh noise $\sqrt{\beta_t}\,z$	x_{t-1}
5	0.359	0.420	$(0.359 - 0.821 \times 0.420) / 0.447 = 0.033$	$0.894 \times (-0.3)$	**−0.235**
4	−0.235	−0.019	$(-0.235 - 0.578 \times (-0.019)) / 0.707 = -0.318$	0.707×0.5	**0.036**
3	0.036	0.306	$(0.036 - 0.426 \times 0.306) / 0.837 = -0.113$	0.548×0.2	**−0.003**
2	−0.003	0.490	$(-0.003 - 0.378 \times 0.490) / 0.894 = -0.211$	$0.447 \times (-0.4)$	**−0.390**
1					

$c_t = \beta_t / \sqrt{(1 - \bar{\alpha}_t)}$. $\hat{\varepsilon}$ is the best possible guess for this one-pixel dataset (optional depth shows how to compute it).

> Step 2: $x_1 = -0.390$. Now it leans towards paper.

▶ Notebook 4.2

The toy, all the way back: every number

t	x_t	$\hat{\varepsilon}$ (network)	mean $\mu = (x_t - c_t\,\hat{\varepsilon}) / \sqrt{\alpha_t}$	fresh noise $\sqrt{\beta_t}\,z$	x_{t-1}
5	0.359	0.420	(0.359 − 0.821 × 0.420) / 0.447 = 0.033	0.894 × (−0.3)	**−0.235**
4	−0.235	−0.019	(−0.235 − 0.578 × (−0.019)) / 0.707 = −0.318	0.707 × 0.5	**0.036**
3	0.036	0.306	(0.036 − 0.426 × 0.306) / 0.837 = −0.113	0.548 × 0.2	**−0.003**
2	−0.003	0.490	(−0.003 − 0.378 × 0.490) / 0.894 = −0.211	0.447 × (−0.4)	**−0.390**
1	−0.390	1.765	(−0.390 − 0.316 × 1.765) / 0.949 = −0.999	none	**−0.999**

$c_t = \beta_t/\sqrt{(1 - \bar{\alpha}_t)}$. $\hat{\varepsilon}$ is the best possible guess for this one-pixel dataset (optional depth shows how to compute it).

> Step 1, no fresh noise: $x_0 = -0.999$. Paper. Generation does not undo our forward run: it draws a new pixel. Run it many times (next slide) and about 35% come out ink, as in the data.

▶ Notebook 4.2

Try it: walk back from noise, one step at a time

$t = 2, \qquad x_2 = -0.003$

$\hat{\varepsilon} = \text{network}(x_2, 2) = 0.490$

the best possible guess for this one-pixel dataset

$$\mu = \frac{1}{\sqrt{\alpha_2}}\left(x_2 - \frac{\beta_2}{\sqrt{1-\bar{\alpha}_2}}\,\hat{\varepsilon}\right) = \frac{1}{0.894}\left(-0.003 - \frac{0.2}{0.529} \times 0.490\right) = -0.211$$

$$x_1 = \mu + \sqrt{\beta_2}\, z = -0.211 + 0.447\, z$$

the network's guess of the clean pixel: $\hat{x}_0 = \frac{x_2 - \sqrt{1-\bar{\alpha}_2}\,\hat{\varepsilon}}{\sqrt{\bar{\alpha}_2}} = -0.310$

previous step: $x_2 = -0.113 + 0.548 \times 0.200 = -0.003$



> Step back one click at a time and watch every number: the network's guess, the mean, the fresh noise. Start new runs and count how often you end at ink. Then switch to the real schedule: 1,000 tiny steps.

▶ Notebook 4.3

The sampling loop: start from noise, repeat a thousand times


```
x = torch.randn(n, 1, 28, 28)                # start: pure noise
for t in range(T, 0, -1):                    # 1000, 999, ..., 1
    eps_hat = model(x, t)                    # guess the noise
    c = beta[t] / (1 - alpha_bar[t]).sqrt()
    mean = (x - c * eps_hat) / alpha[t].sqrt()   # the mean of 2.4
    z = torch.randn_like(x) if t > 1 else 0  # no noise at the last step
    x = mean + sigma[t] * z                  # sample the Gaussian step
return x
```

> Start from pure noise, which Part 1 showed is where every image ends. Then a thousand times: guess the noise, compute the mean, add a little fresh noise. No noise at the last step.

Ho, Jain & Abbeel, "Denoising Diffusion Probabilistic Models" (2020) (Algorithm 2)

▶ Notebook 4.3

The sampling loop: start from noise, repeat a thousand times


```
x = torch.randn(n, 1, 28, 28)                # start: pure noise
for t in range(T, 0, -1):                    # 1000, 999, ..., 1
    eps_hat = model(x, t)                    # guess the noise
    c = beta[t] / (1 - alpha_bar[t]).sqrt()
    mean = (x - c * eps_hat) / alpha[t].sqrt()   # the mean of 2.4
    z = torch.randn_like(x) if t > 1 else 0  # no noise at the last step
    x = mean + sigma[t] * z                  # sample the Gaussian step
return x
```

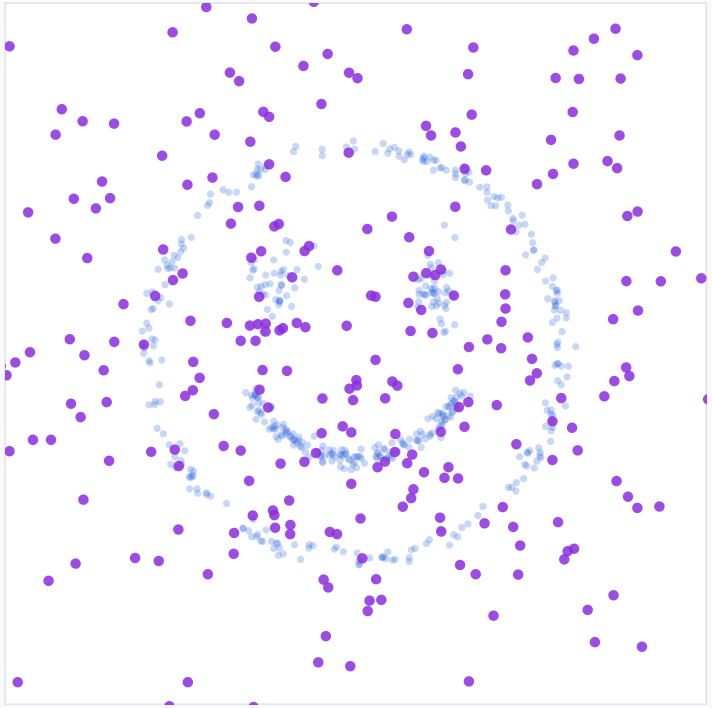
Training uses one random level per example; **sampling** visits every level, in order: 6.1 s for 64 digits on a laptop GPU.

> Training never runs this loop; sampling always does. That is why sampling is slow, and Part 6 makes it faster.

Ho, Jain & Abbeel, "Denoising Diffusion Probabilistic Models" (2020) (Algorithm 2)

▶ Notebook 4.4

Watch 300 particles find the smiley, and drop your own



t = 1000

300 particles, each its own DDPM run

orange: three of their paths

faint blue: the training data

> Three hundred particles, each its own sampling run. At t = 1,000 they are pure noise.

▶ Notebook 4.4

Watch 300 particles find the smiley, and drop your own

t = 0

300 particles, each its own DDPM run

orange: three of their paths

faint blue: the training data

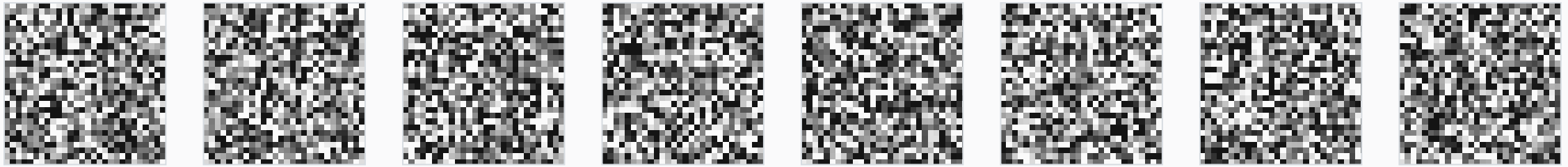
> Press play: the network pulls them onto the face. Click anywhere in the square to drop your own particle: the network, running in this page, carries it home.

Ari Seff, "What are Diffusion Models?" (2022) (5:17) · MIT 6.S183 "A Practical Introduction to Diffusion Models" (IAP 2025), Lecture 1

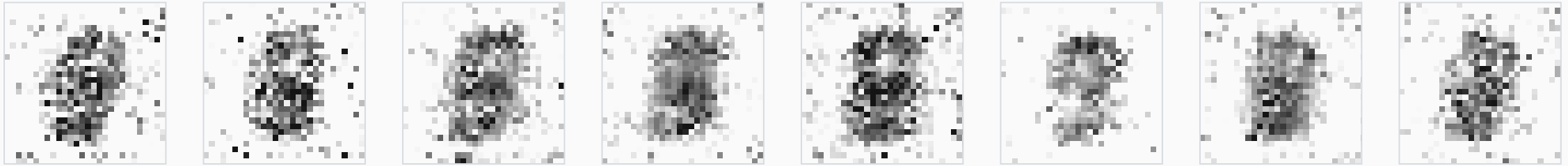
▶ Notebook 4.4

Eight runs from noise: x_t, and the network's guess of the end

x_t at t = 1000



$\hat{x}_0$: the network's guess of the finished digit



> Top: x_t for eight runs; bottom: the network's guess of the finished digit at that moment. Coarse first: a blob, then a shape, then strokes.

▶ Notebook 4.4

Eight runs from noise: x_t, and the network's guess of the end

x_t at t = 0



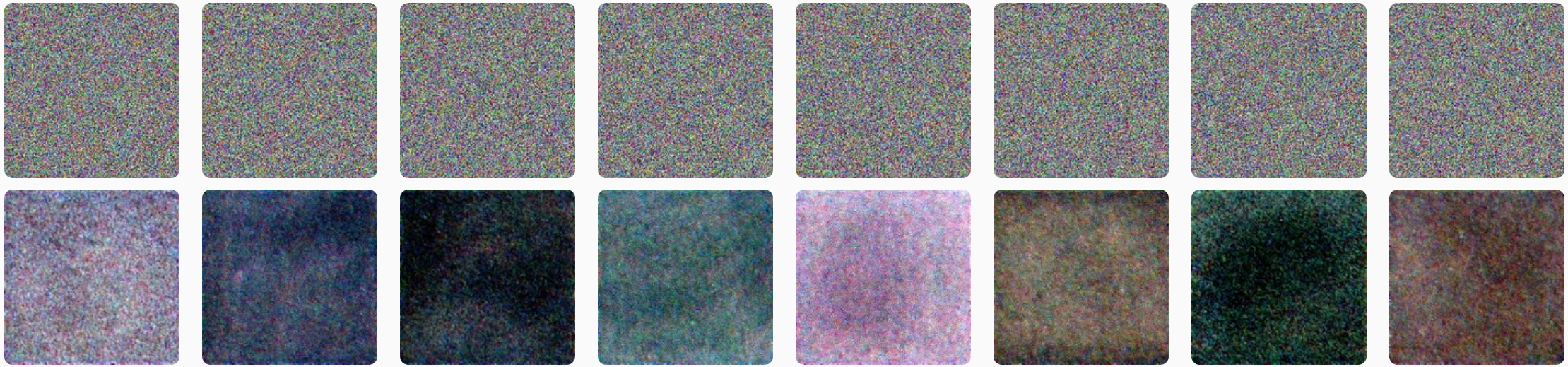
$\hat{x}_0$: the network's guess of the finished digit



> Watch the fourth run: its guess changes its mind, a 0, then a 9, a 5, a 3, and finally a 9. Early steps choose the rough shape; late steps add the details.

▶ Notebook 4.4

Eight runs of the real cat model: x_t, and its guess of the end



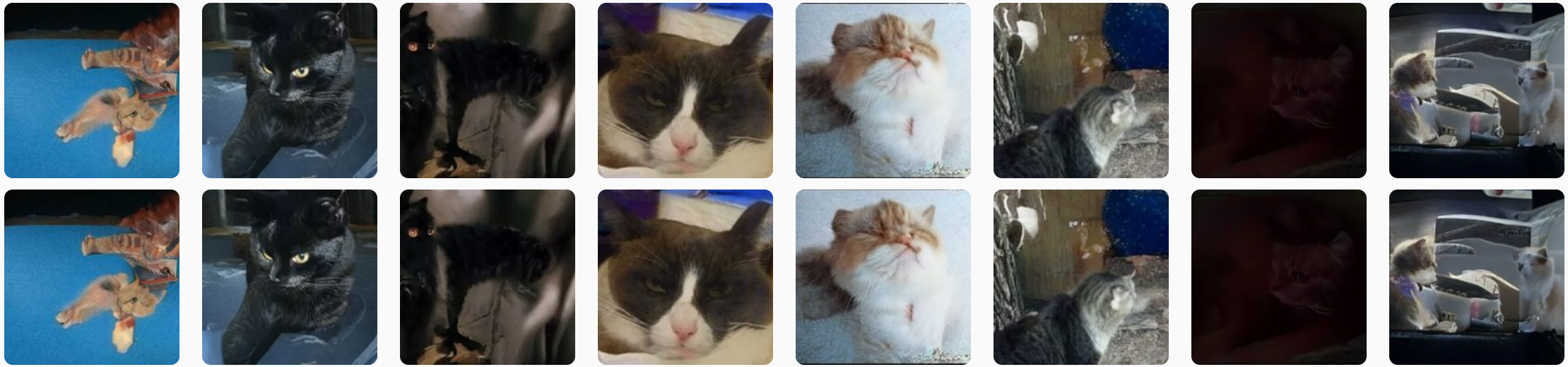
Top: x_t. Bottom: the network's guess of the finished image. t = 1000 · step 0 of 1,000

> The opening film, now with a second row: at every step, the network's guess of the finished photo. Press play.

Ho, Jain & Abbeel, "Denoising Diffusion Probabilistic Models" (2020) · Companion film "Diffusion models, from noise to images, step by step" (2026): google/ddpm-cat-256 run on the CC0 photo "Chelsea" (scikit-image)

▶ Notebook 4.4

Eight runs of the real cat model: x_t, and its guess of the end



Top: x_t. Bottom: the network's guess of the finished image. t = 0 · step 1000 of 1,000

> Coarse to fine, as on MNIST: first blurry light and dark, then a layout, then fur and eyes. Not every run is a good cat: this 2020 model is small by today's standards.

Ho, Jain & Abbeel, "Denoising Diffusion Probabilistic Models" (2020) · Companion film "Diffusion models, from noise to images, step by step" (2026): google/ddpm-cat-256 run on the CC0 photo "Chelsea" (scikit-image)

▶ Notebook 4.5

Take out the fresh noise, and every sample collapses

with fresh noise



> Sixteen runs of the sampler, as designed.

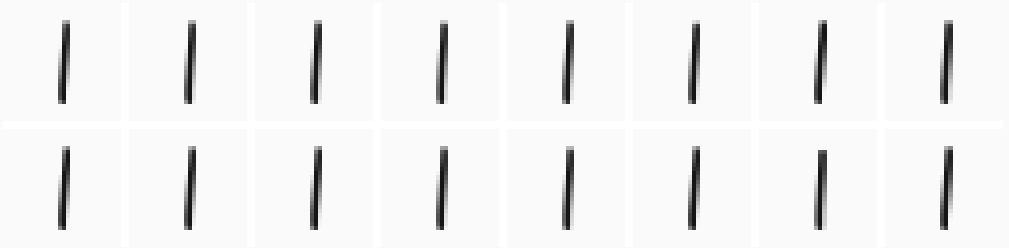
▶ Notebook 4.5

Take out the fresh noise, and every sample collapses

with fresh noise



same starts, no fresh noise



> The same sixteen starting noises, keeping only the mean of each step: sixteen identical thin 1s. Ink covers 5% of the image instead of 14%.

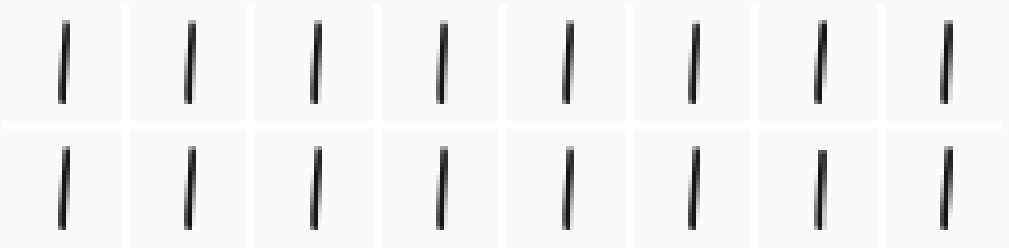
▶ Notebook 4.5

Take out the fresh noise, and every sample collapses

with fresh noise



same starts, no fresh noise

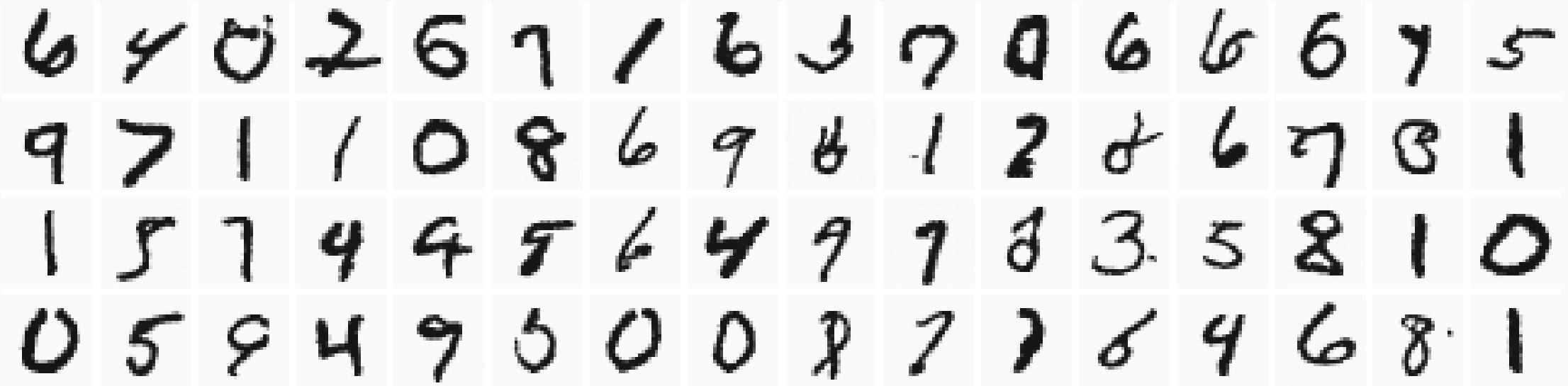


The smiley does the same: without fresh noise all 300 particles end at one point. Each mean step pulls towards an average; only the noise lets a path commit, as our pixel must commit to paper or ink.

> In 2-D every particle collapses to a single point. The step-back mean is a pull towards the average guess; the fresh noise is what keeps the samples spread like the data.

▶ Notebook 4.6

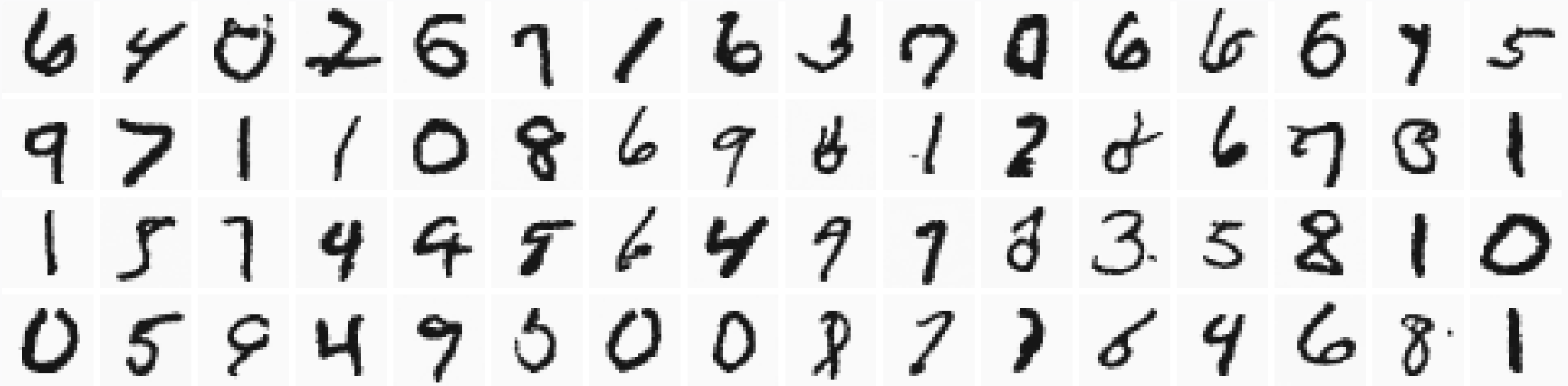
64 new digits



64 samples from our model: each is a thousand steps from its own noise (6.1 s for all 64).

▶ Notebook 4.6

64 new digits



Sharp like the GAN's, trained with one regression loss like the VAE. And none is a copy of a training image (optional depth measures it).

> This is what we were after: sharp digits from a stable, simple training loss.

▶ Notebook 4.6

The whole method on a napkin

Train

1. x_0 from the data, $t \sim 1 \dots T$, $\varepsilon \sim N(0, I)$
2. $x_t = \sqrt{\bar{\alpha}_t}\, x_0 + \sqrt{(1 - \bar{\alpha}_t)}\, \varepsilon$
3. step on $\|\varepsilon - \hat{\varepsilon}_\theta(x_t, t)\|^2$

Sample

1. $x_T \sim N(0, I)$
2. for $t = T \dots 1$: $\hat{\varepsilon} = \hat{\varepsilon}_\theta(x_t, t)$
3. $x_{t-1} = (x_t - \beta_t/\sqrt{(1-\bar{\alpha}_t)}\, \hat{\varepsilon})/\sqrt{\alpha_t} + \sigma_t\, z$

> Everything so far, in six lines. Paras Chopra's test: if you can derive these on a napkin, you have understood diffusion.

Paras Chopra, "Diffusion Models from First Principles" (2025) ("derive it on a napkin")

▶ Notebook 4.6

The whole method on a napkin

Train

1. x_0 from the data, $t \sim 1 \dots T$, $\varepsilon \sim N(0, I)$
2. $x_t = \sqrt{\bar{\alpha}_t}\, x_0 + \sqrt{(1 - \bar{\alpha}_t)}\, \varepsilon$
3. step on $\|\varepsilon - \hat{\varepsilon}_\theta(x_t, t)\|^2$

Sample

1. $x_T \sim N(0, I)$
2. for $t = T \dots 1$: $\hat{\varepsilon} = \hat{\varepsilon}_\theta(x_t, t)$
3. $x_{t-1} = (x_t - \beta_t/\sqrt{(1-\bar{\alpha}_t)}\, \hat{\varepsilon})/\sqrt{\alpha_t} + \sigma_t\, z$

End of the first session. Next: the score view, faster sampling, and steering with labels and text.

> Next session: arrows to the data (the score), faster sampling, and steering with labels and text.

Paras Chopra, "Diffusion Models from First Principles" (2025) ("derive it on a napkin")

▶ Notebook 5.1

The score: an arrow at every point, pointing uphill

$$s(x) = \nabla_x \log p(x)$$

> The score of a distribution: the gradient of its log density, an arrow at every point.

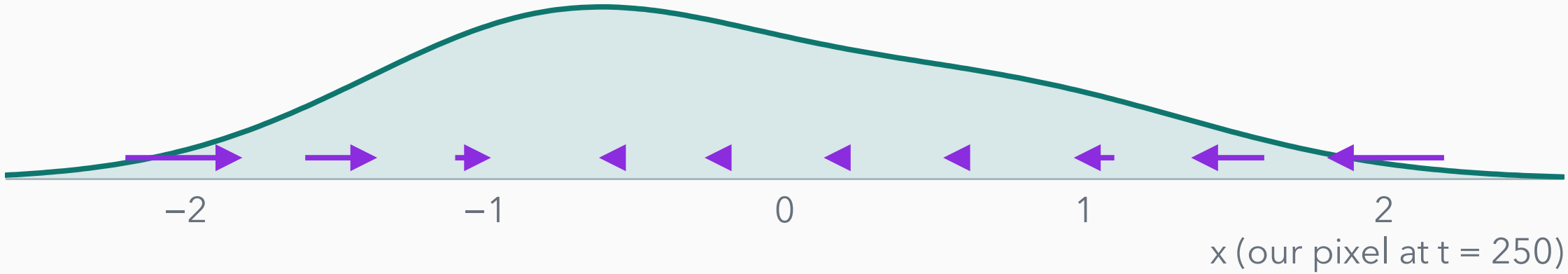
Paras Chopra, "Diffusion Models from First Principles" (2025) (IX) · Vizuara, "Principles of Diffusion Models", Lecture 1 (2025) (39:27) · Song & Ermon, "Generative Modeling by Estimating Gradients of the Data Distribution" (2019)

▶ Notebook 5.1

The score: an arrow at every point, pointing uphill

$$s(x) = \nabla_x \log p(x)$$

density of the noisy pixel (green) and its score (purple arrows: uphill)



> Our noisy pixel at t = 250: two bumps. Every arrow points uphill, towards the nearer bump; between them, towards the more probable one.

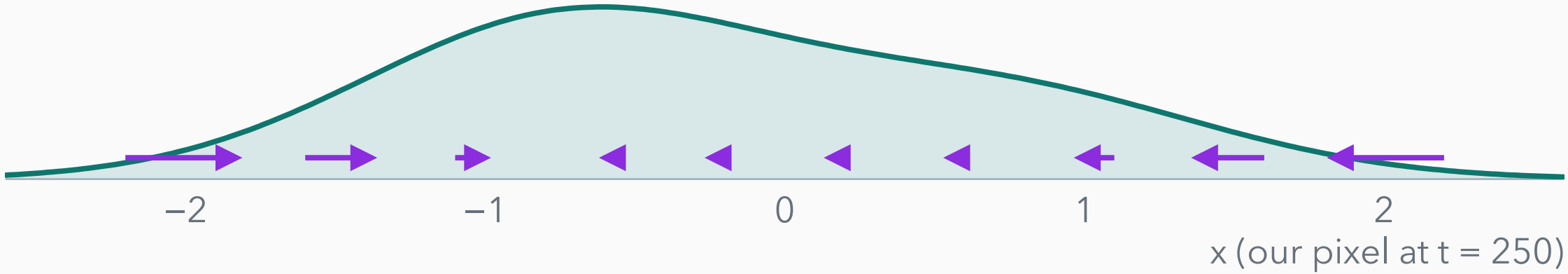
Paras Chopra, "Diffusion Models from First Principles" (2025) (IX) · Vizuara, "Principles of Diffusion Models", Lecture 1 (2025) (39:27) · Song & Ermon, "Generative Modeling by Estimating Gradients of the Data Distribution" (2019)

▶ Notebook 5.1

The score: an arrow at every point, pointing uphill

$$s(x) = \nabla_x \log p(x)$$

density of the noisy pixel (green) and its score (purple arrows: uphill)

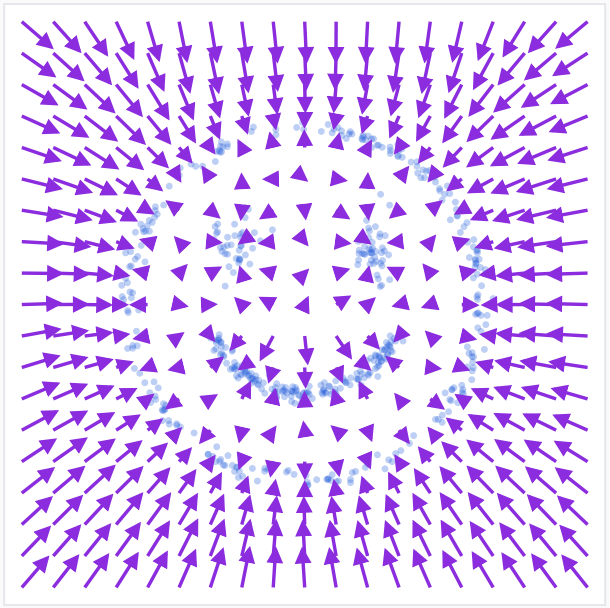


> Why it is useful: the gradient of log p ignores the normalising constant, the integral that made Lecture 1 hard. Arrows are enough to sample.

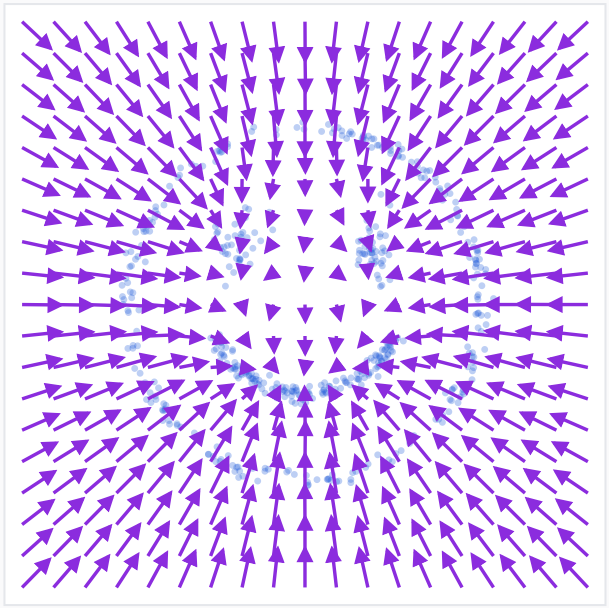
Paras Chopra, "Diffusion Models from First Principles" (2025) (IX) · Vizuara, "Principles of Diffusion Models", Lecture 1 (2025) (39:27) · Song & Ermon, "Generative Modeling by Estimating Gradients of the Data Distribution" (2019)

▶ Notebook 5.1

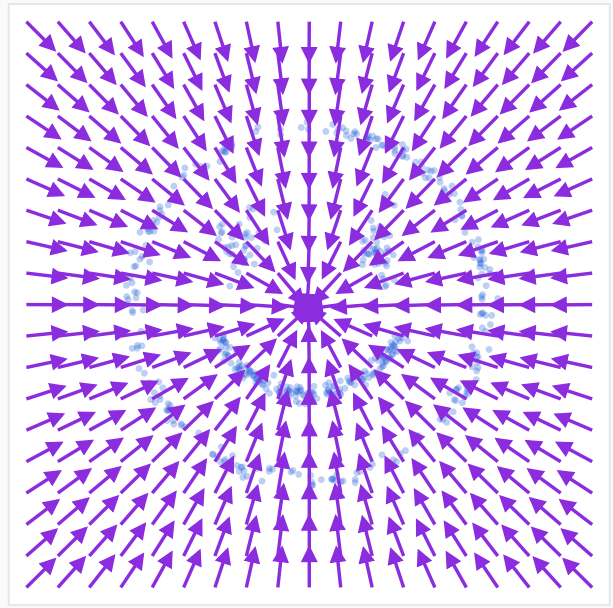
The smiley's arrows, at different noise levels



t = 10



t = 100

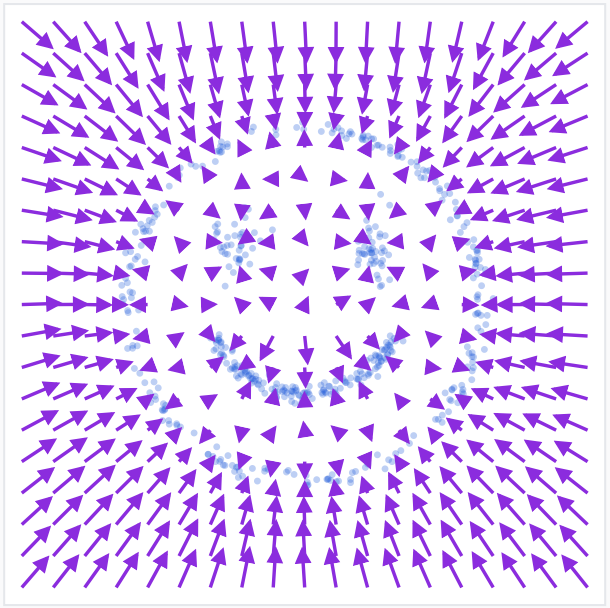


t = 500

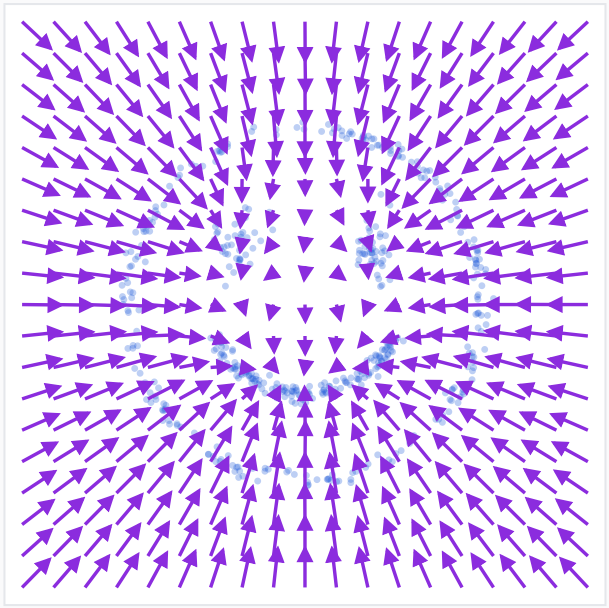
From the trained network: flip the guessed noise and scale it, $s \approx -\hat{\varepsilon}/\sqrt{(1-\bar{\alpha}_t)}$. The noise guess points away from the data; the score points towards it. At t = 100 every arrow points to a slightly blurred, slightly shrunk smiley.

▶ Notebook 5.1

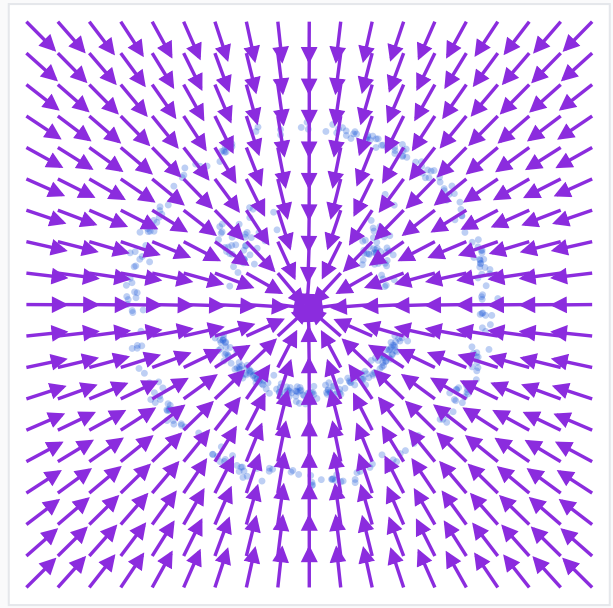
The smiley's arrows, at different noise levels



t = 10



t = 100

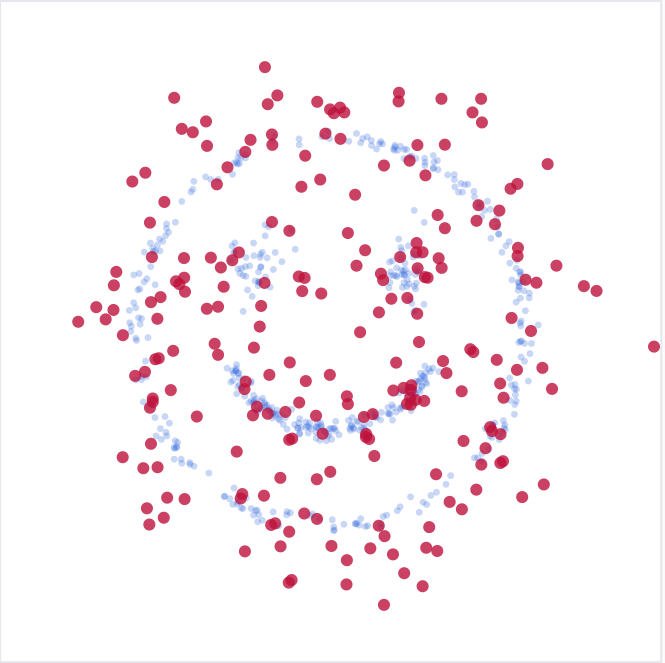


t = 500

Move t. At t = 10 the arrows are sharp near the face; far from it the network saw no training points at this level, and the arrows are less trustworthy. At t = 500 the density is blurred everywhere and the arrows are smooth.

▶ Notebook 5.2

Follow the arrows, with a little noise: Langevin sampling



annealed, t = 1000 → 1

$$x \leftarrow x + (\eta/2) \cdot \text{score}(x) + \sqrt{\eta} \cdot z$$

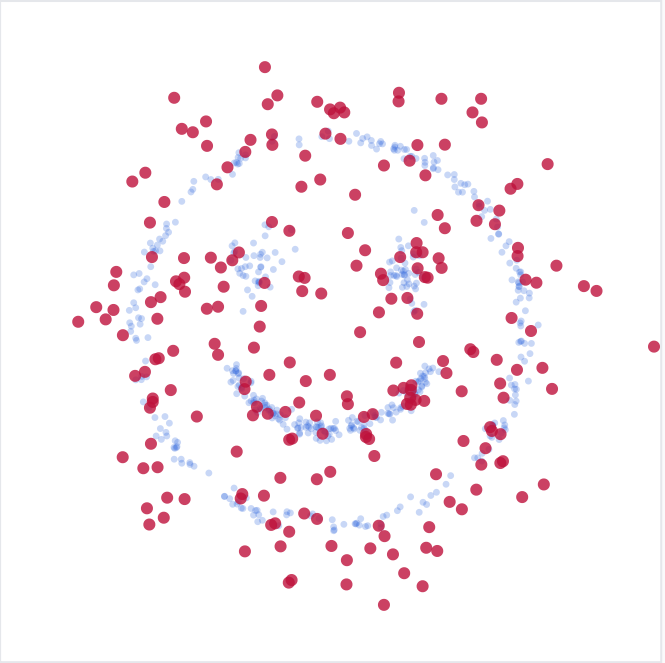
One level: follow the arrows of that noise level.

Annealed: start at t = 1000, lower the noise step by step, as DDPM does.

> Langevin dynamics: a small step uphill along the score, plus a small random kick. Run it at one fixed noise level, starting from pure noise.

▶ Notebook 5.2

Follow the arrows, with a little noise: Langevin sampling



annealed, t = 1000 → 1

$$x \leftarrow x + (\eta/2) \cdot \text{score}(x) + \sqrt{\eta} \cdot z$$

One level: follow the arrows of that noise level.

Annealed: start at t = 1000, lower the noise step by step, as DDPM does.

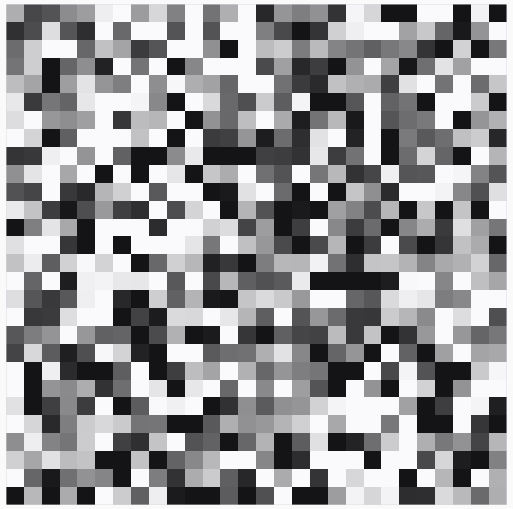
> Now anneal: begin at t = 1,000 where the arrows exist everywhere, and lower the noise as the particles settle. This is what DDPM's sampler does, with a step size set by the schedule.

Song & Ermon, "Generative Modeling by Estimating Gradients of the Data Distribution" (2019) · Paras Chopra, "Diffusion Models from First Principles" (2025) (IX)

▶ Notebook 6.1

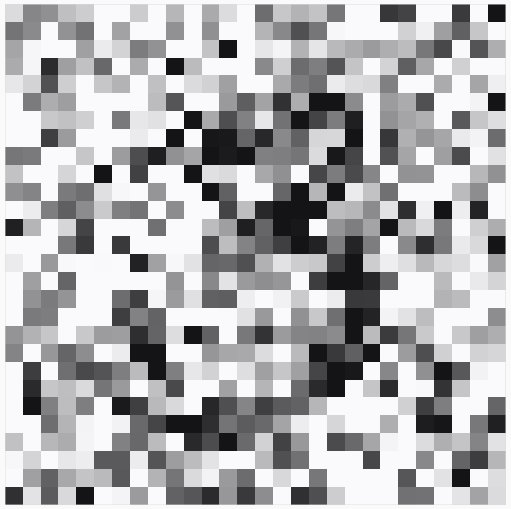
The schedule matters: linear destroys too early

linear

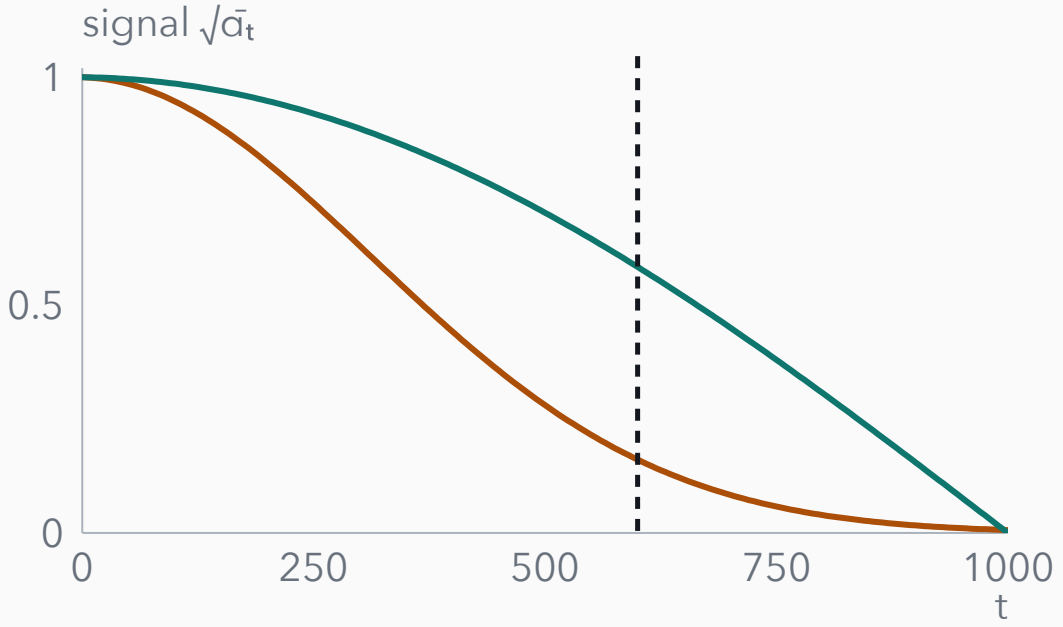


signal $\sqrt{\bar{\alpha}} = 0.161$

cosine



signal $\sqrt{\bar{\alpha}} = 0.584$



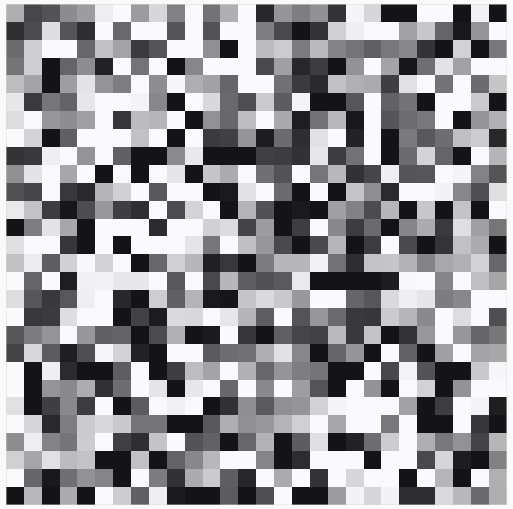
> Our 3 at t = 600 under the linear schedule of DDPM: gone. Under the cosine schedule: still visible. Move t.

Nichol & Dhariwal, "Improved Denoising Diffusion Probabilistic Models" (2021) · Outlier, "Diffusion Models, Paper Explanation, Math Explained" (2022) (6:26)

▶ Notebook 6.1

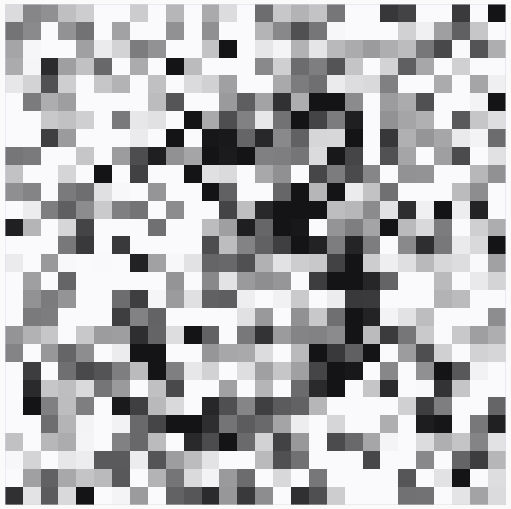
The schedule matters: linear destroys too early

linear

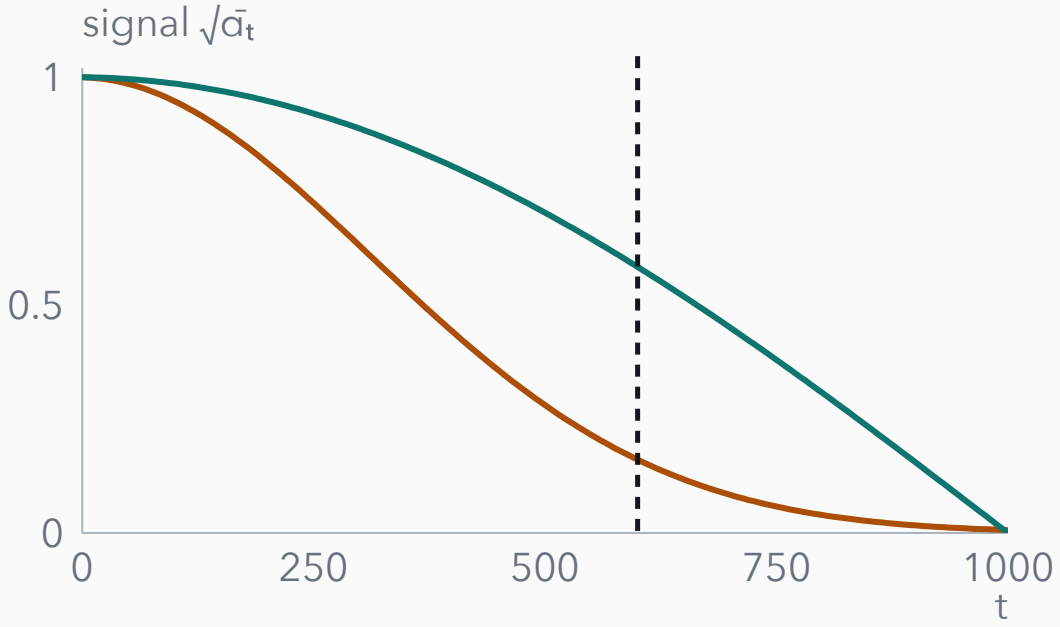


signal $\sqrt{\bar{\alpha}} = 0.161$

cosine



signal $\sqrt{\bar{\alpha}} = 0.584$



> With the linear schedule the last few hundred steps are almost pure noise, so the network spends many of its levels learning very little. The cosine schedule spreads the destruction evenly.

Nichol & Dhariwal, "Improved Denoising Diffusion Probabilistic Models" (2021) · Outlier, "Diffusion Models, Paper Explanation, Math Explained" (2022) (6:26)

▶ Notebook 6.2

A thousand network calls per image

our model, laptop GPU	time
one network call, 8 digits	5 ms
DDPM: 1,000 calls	3.2 s
a 512×512 text-to-image model	about a thousand times more work per call

> Sampling costs one network call per step: a thousand calls for each batch. Fine for 28×28 digits, painful for large images.

▶ Notebook 6.2

A thousand network calls per image

our model, laptop GPU	time
one network call, 8 digits	5 ms
DDPM: 1,000 calls	3.2 s
a 512×512 text-to-image model	about a thousand times more work per call

Can we take fewer, bigger steps with the same trained network?

> Part 2 said big jumps back are not Gaussian. Is there another way to jump?

▶ Notebook 6.2

DDIM: jump to the guess, then re-noise it to an earlier level

$$\hat{x}_0 = \frac{x_t - \sqrt{1-\bar{\alpha}_t}\,\hat{\varepsilon}}{\sqrt{\bar{\alpha}_t}}$$

> Guess the clean image from x_t, as before.

▶ Notebook 6.2

DDIM: jump to the guess, then re-noise it to an earlier level

$$\hat{x}_0 = \frac{x_t - \sqrt{1-\bar{\alpha}_t}\,\hat{\varepsilon}}{\sqrt{\bar{\alpha}_t}}$$

$$x_{t'} = \sqrt{\bar{\alpha}_{t'}}\,\hat{x}_0 + \sqrt{1-\bar{\alpha}_{t'}}\,\hat{\varepsilon} \qquad (t' < t)$$

> Then use the shortcut of 1.3 in reverse: noise the guess back up to an earlier level t′, reusing the same guessed noise instead of a fresh draw.

▶ Notebook 6.2

DDIM: jump to the guess, then re-noise it to an earlier level

$$\hat{x}_0 = \frac{x_t - \sqrt{1-\bar{\alpha}_t}\,\hat{\varepsilon}}{\sqrt{\bar{\alpha}_t}}$$

$$x_{t'} = \sqrt{\bar{\alpha}_{t'}}\,\hat{x}_0 + \sqrt{1-\bar{\alpha}_{t'}}\,\hat{\varepsilon} \qquad (t' < t)$$

$$\text{pixel, } 250 \to 200: \quad \sqrt{0.659} \times 0.26 + \sqrt{0.341} \times 0.28 = 0.373$$

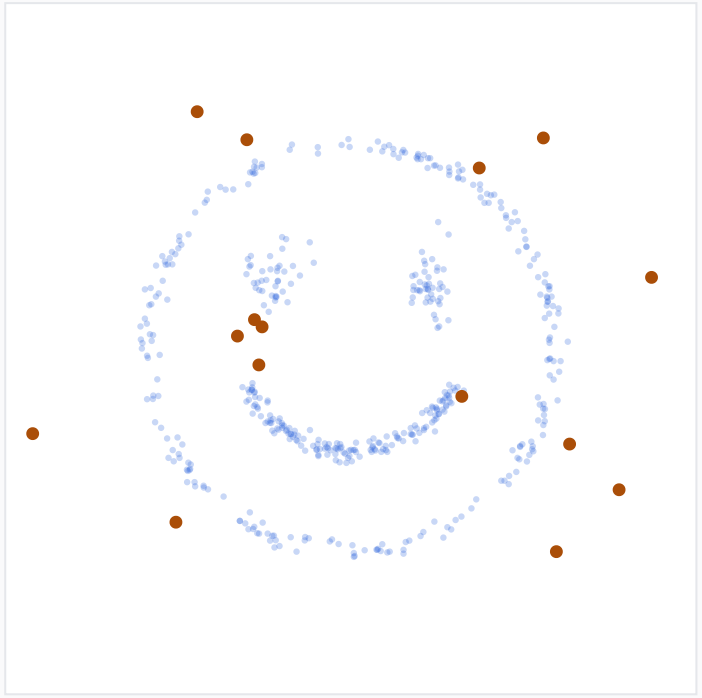
> No fresh noise, so the path is deterministic, and t′ can be far earlier: 50 jumps instead of 1,000, with the same trained network.

Song, Meng & Ermon, "Denoising Diffusion Implicit Models" (2021) · MIT 6.S183 "A Practical Introduction to Diffusion Models" (IAP 2025), Lecture 2 (24:35)

▶ Notebook 6.2

Same network: DDPM paths jitter, DDIM paths glide

DDPM: 1,000 steps with fresh noise



DDIM: 50 steps, no fresh noise



> Forty particles, same starting noise, same trained network. Left: DDPM, a thousand random steps. Right: DDIM, fifty deterministic jumps.

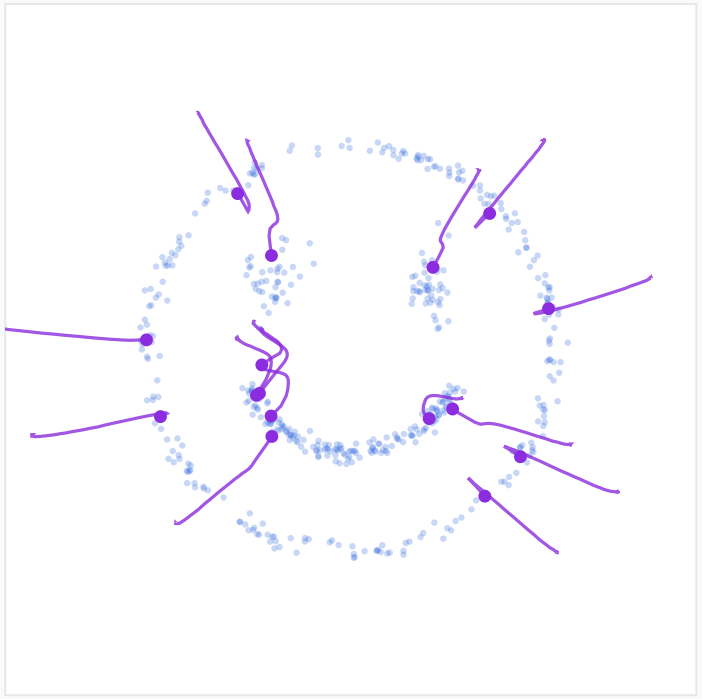
▶ Notebook 6.2

Same network: DDPM paths jitter, DDIM paths glide

DDPM: 1,000 steps with fresh noise



DDIM: 50 steps, no fresh noise

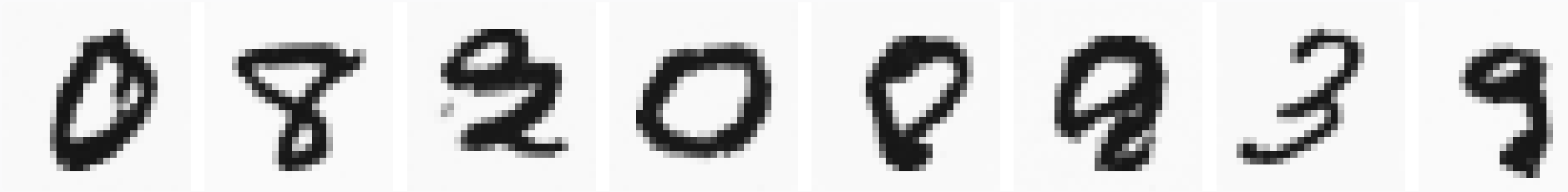


DDPM paths wander and then settle; DDIM paths are nearly straight. Both end on the smiley.

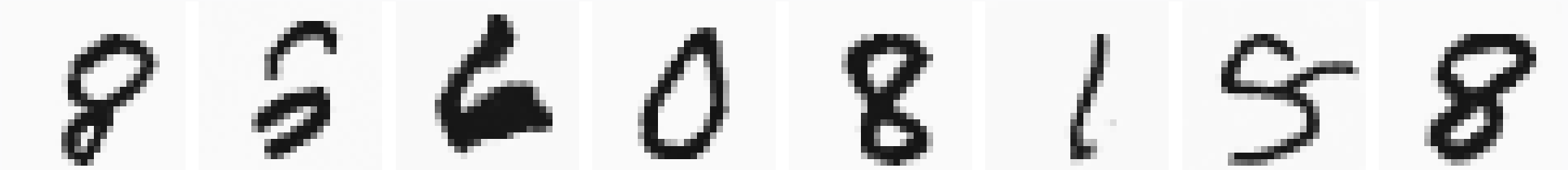
▶ Notebook 6.2

Fewer steps, same network: from 1 to 1,000

DDIM, 20 steps · 0.060 s for 8 digits



DDPM, 1,000 steps with fresh noise (same starting noise)



> Eight digits from the same eight noises. One step gives the blurry average guess of 3.5; about twenty steps already give clean digits, fifty times faster than DDPM.

▶ Notebook 6.3

Stable Diffusion in fewer steps, and in one

DDIM, the same network



1 step | 2 steps | 5 steps | 10 steps | 25 steps | 50 steps

> The real Stable Diffusion: one sentence, one starting noise, fewer and fewer DDIM steps. From about five steps there is a fox; one or two big jumps are not enough. Twenty-five steps look as good as fifty.

Song, Meng & Ermon, "Denoising Diffusion Implicit Models" (2021) · Luo et al., "Latent Consistency Models" (2023) · Stable Diffusion v1.5, run for the companion film "Diffusion models, from noise to images, step by step" (2026)

▶ Notebook 6.3

Stable Diffusion in fewer steps, and in one

DDIM, the same network



1 step | 2 steps | 5 steps | 10 steps | 25 steps | 50 steps

a distilled model (LCM)



1 step | 2 steps | 4 steps | 8 steps | DDIM, 4 steps

A distilled model, a latent consistency adapter trained to jump far, draws a recognisable fox in one step and a sharp one in four. Plain DDIM at four steps (right) is still smeared.

Song, Meng & Ermon, "Denoising Diffusion Implicit Models" (2021) · Luo et al., "Latent Consistency Models" (2023) · Stable Diffusion v1.5, run for the companion film "Diffusion models, from noise to images, step by step" (2026)

▶ Notebook 6.3

Where the field went: same idea, better parts

direction	what changes	examples
transformers as denoisers	the U-Net becomes a ViT on patches	DiT; Stable Diffusion 3; video models
straight paths	predict a velocity along a straight line from noise to data	flow matching; rectified flow
few steps	train a student to jump far	distillation; consistency models (1–4 steps)
beyond images	the same loop on other data	audio, video, 3-D, protein structures, robot policies, weather
text	undo masking instead of Gaussian noise	masked diffusion language models

> Since 2020 the parts changed and the idea stayed: corrupt data with a known process, learn to undo it in small steps, sample by running the steps.

MIT 6.S183 "A Practical Introduction to Diffusion Models" (IAP 2025), Lecture 2 · MIT 6.S183 "A Practical Introduction to Diffusion Models" (IAP 2025), Lecture 6 · Peebles & Xie (2023) · Lipman et al. (2023) · Liu et al. (2023) · Song et al. (2023)

▶ Notebook 7.1

A 3 on demand: give the network the label


```
h = time_mlp(time_features(t))      # how noisy
h = h + label_embedding(c)            # which digit: one added line
eps_hat = unet(x_t, h)
```

Conditioning is one line: a learned embedding of the label is added to the time embedding, so every block knows both how noisy the image is and which digit to make.

▶ Notebook 7.1

A 3 on demand: give the network the label


```
h = time_mlp(time_features(t))      # how noisy
h = h + label_embedding(c)            # which digit: one added line
eps_hat = unet(x_t, h)
```



Asked for a 3: **94%** are classified as 3 by a separate MNIST classifier (98% test accuracy).

Asked for a 3, the conditional model gives 15 of 16 that our classifier calls a 3.

▶ Notebook 7.1

A 3 on demand: give the network the label


```
h = time_mlp(time_features(t))      # how noisy
h = h + label_embedding(c)            # which digit: one added line
eps_hat = unet(x_t, h)
```



Asked for a 3: **94%** are classified as 3 by a separate MNIST classifier (98% test accuracy).

Good, but some are weak or ambiguous. Can we push the samples to be more clearly 3s?

> Not all are convincing. Guidance pushes harder, and it comes from Bayes' rule.

▶ Notebook 7.2

Guidance: ask twice, then exaggerate the difference

$$\hat{\varepsilon}_{\text{guided}} = \hat{\varepsilon}(x_t, t, \varnothing) + w\left(\hat{\varepsilon}(x_t, t, c) - \hat{\varepsilon}(x_t, t, \varnothing)\right)$$

> Ask the same network twice: once with the label, once with it hidden. The difference points towards "more like the label". Move w times along it.

Ho & Salimans, "Classifier-Free Diffusion Guidance" (2022) · Paras Chopra, "Diffusion Models from First Principles" (2025) (XII)

▶ Notebook 7.2

Guidance: ask twice, then exaggerate the difference

$$\hat{\varepsilon}_{\text{guided}} = \hat{\varepsilon}(x_t, t, \varnothing) + w\left(\hat{\varepsilon}(x_t, t, c) - \hat{\varepsilon}(x_t, t, \varnothing)\right)$$

w = 0 ignores the label · w = 1 is the conditional model · w > 1 exaggerates it.

> w = 0 ignores the label, w = 1 is the conditional model itself, and larger w exaggerates the label.

▶ Notebook 7.2

Guidance: ask twice, then exaggerate the difference

$$\hat{\varepsilon}_{\text{guided}} = \hat{\varepsilon}(x_t, t, \varnothing) + w\left(\hat{\varepsilon}(x_t, t, c) - \hat{\varepsilon}(x_t, t, \varnothing)\right)$$

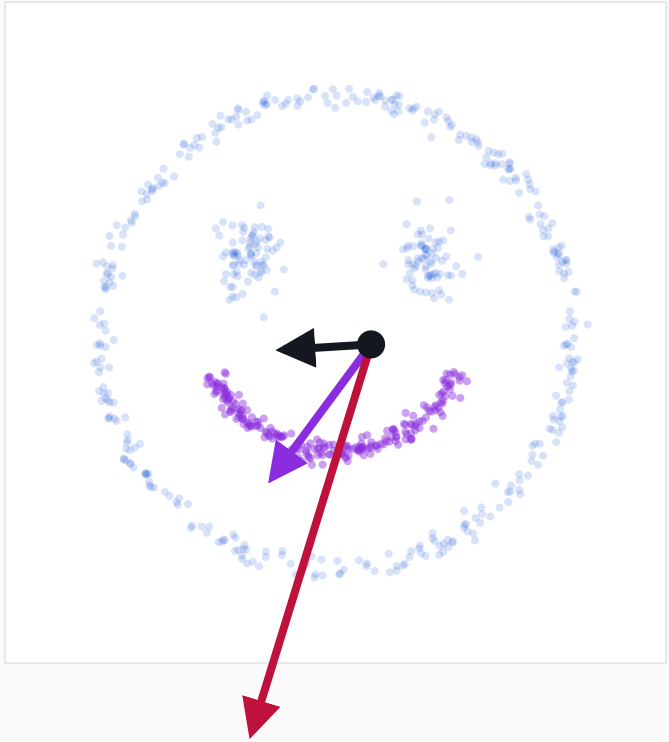
w = 0 ignores the label · w = 1 is the conditional model · w > 1 exaggerates it.

```
e_none = model(x_t, t, NULL)          # the label hidden
e_c    = model(x_t, t, c)             # the label shown
eps_hat = e_none + w * (e_c - e_none)
```

> One network gives both answers: in training, the label is replaced by “no label” 10% of the time. At sampling time: two calls per step.

▶ Notebook 7.2

Guidance as arrows: two guesses, and a push past them



at the black point, t = 300

no label: $\hat{\varepsilon}(\varnothing) = (0.16, 0.01)$

"mouth": $\hat{\varepsilon}(c) = (0.18, 0.24)$

guided: $\hat{\varepsilon}(\varnothing) + 3.0 \times (\hat{\varepsilon}(c) - \hat{\varepsilon}(\varnothing)) = (0.22, 0.70)$

w > 1: the red arrow goes past the purple one,
an extra push away from "any part of the face".

Arrows: the direction $-\hat{\varepsilon}$ in which each guess moves the point, drawn three times longer than $-\sqrt{(1-\bar{\alpha})}\,\hat{\varepsilon}$ to be visible.

Click in the square to move the point.

The part-conditional smiley network at one point. Black: its guess with no label. Purple: its guess for "mouth". Red: the guided guess. Move w: past 1, the red arrow overshoots the purple one, pushing harder towards the mouth. Click to move the point.

▶ Notebook 7.3

The guidance dial: more w, more 3, less variety

16 samples asked for a 3, w = 3



classified as 3

100%

variety (spread)

15.8

ink per image

17%

> Sixteen samples asked for a 3. w = 0 ignores the label (6% are 3s, about the base rate). w = 1: 94%. From w = 2 every sample is a 3, but the variety falls and the strokes get bolder.

▶ Notebook 7.3

The guidance dial: more w, more 3, less variety

16 samples asked for a 3, w = 3



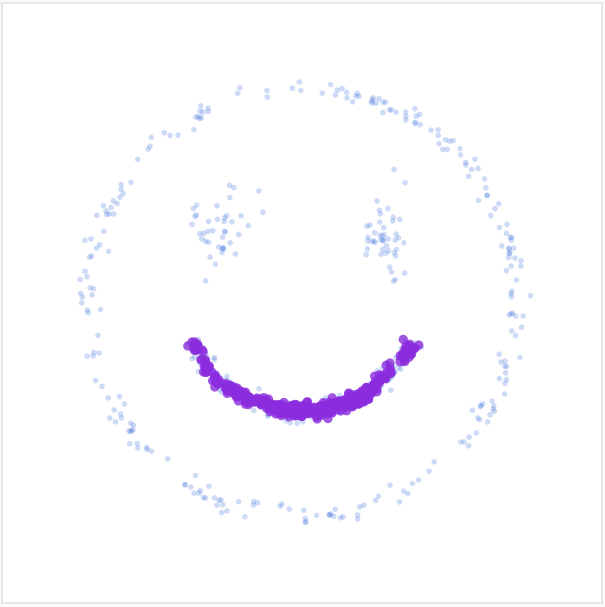
classified as 3

100%
variety (spread)

15.8
ink per image

17%

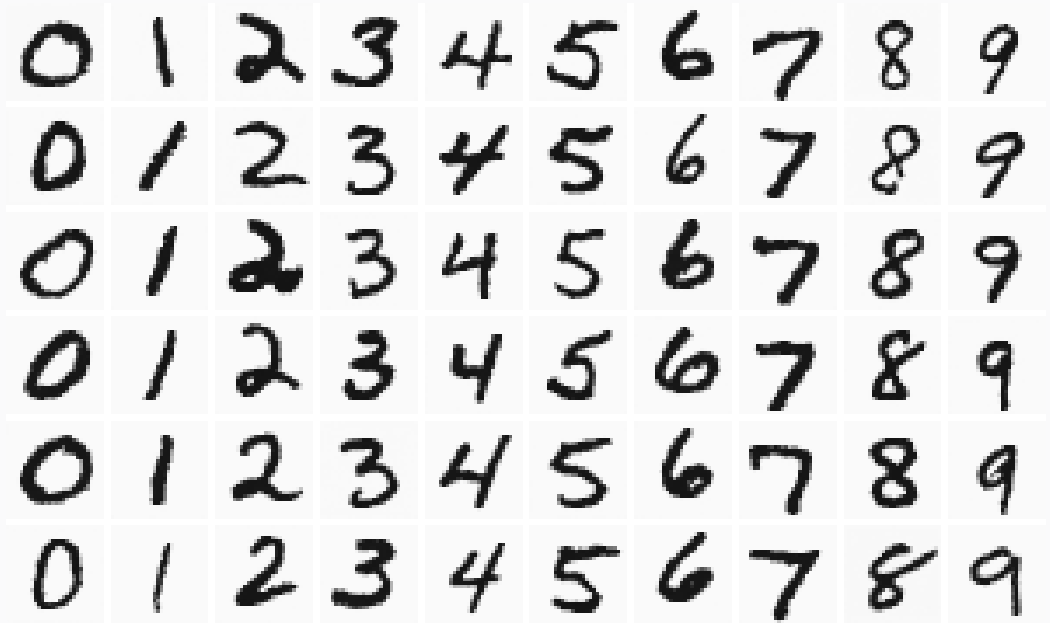
smiley guided to “mouth”, w = 3



> The smiley, guided towards the mouth: higher w pulls the samples into the middle of the mouth. Guidance trades variety for fidelity.

▶ Notebook 7.3

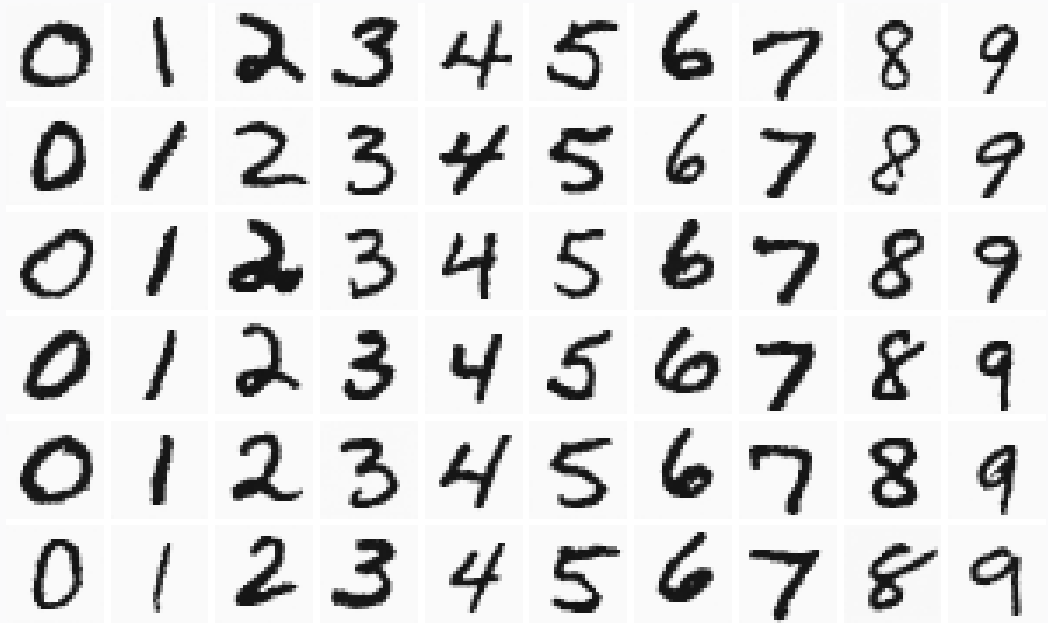
Every digit on demand



Six samples of each digit, guided with w = 3: one network, one label, ten classes.

▶ Notebook 7.3

Every digit on demand



At w = 3 our classifier agrees with the requested digit for **100%** of these 60 samples.

Next: scale it up, from a digit label to a sentence.

> All 60 are classified as the digit we asked for. Next: bigger images (latent diffusion), and a sentence instead of a label.

▶ Notebook 7.4

Pixels are expensive: diffuse a compressed code

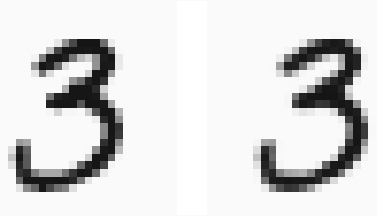
A 512 × 512 colour image: **786,432** numbers through the U-Net, at every one of 50 steps, twice with guidance.

> Running a U-Net on every pixel of a large image, at every step, is the main cost.

▶ Notebook 7.4

Pixels are expensive: diffuse a compressed code

A 512 × 512 colour image: **786,432** numbers through the U-Net, at every one of 50 steps, twice with guidance.



3 → 4 × 7 × 7 code (196 numbers) → decoded



the code: four 7 × 7 channels

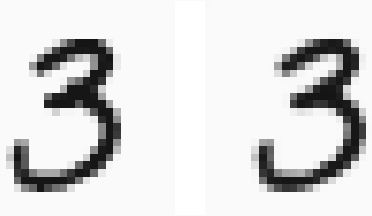
> Lecture 2 to the rescue: an autoencoder compresses our 3 into a 4 × 7 × 7 code and decodes it back almost perfectly.

Rombach et al., "High-Resolution Image Synthesis with Latent Diffusion Models" (2022) · Paras Chopra, "Diffusion Models from First Principles" (2025) (XII)

▶ Notebook 7.4

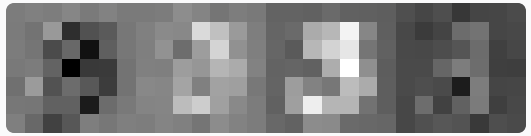
Pixels are expensive: diffuse a compressed code

A 512 × 512 colour image: **786,432** numbers through the U-Net, at every one of 50 steps, twice with guidance.



3 → 4 × 7 × 7 code (196 numbers) → decoded





the code: four 7 × 7 channels

Diffusion on the codes, then the decoder. Stable Diffusion: 512 × 512 × 3 → 64 × 64 × 4, 48 times fewer numbers.

> Train the same diffusion model on the codes, then decode its samples: latent diffusion, the idea behind Stable Diffusion.

Rombach et al., "High-Resolution Image Synthesis with Latent Diffusion Models" (2022) · Paras Chopra, "Diffusion Models from First Principles" (2025) (XII)

▶ Notebook 7.4

Stable Diffusion's autoencoder: 48 times fewer numbers



our photo: 512 × 512 × 3
786,432 numbers

encode →



the code: 4 × 64 × 64
16,384 numbers

> The real autoencoder of Stable Diffusion on our photo: 786,432 numbers in, four 64 × 64 channels out. That is 16,384 numbers, 48 times fewer, and the channels still look like a small cat.

Rombach et al., "High-Resolution Image Synthesis with Latent Diffusion Models" (2022) · Stable Diffusion v1.5, run for the companion film "Diffusion models, from noise to images, step by step" (2026)

▶ Notebook 7.4

Stable Diffusion's autoencoder: 48 times fewer numbers



our photo: 512 × 512 × 3
786,432 numbers

encode →



the code: 4 × 64 × 64
16,384 numbers

decode →



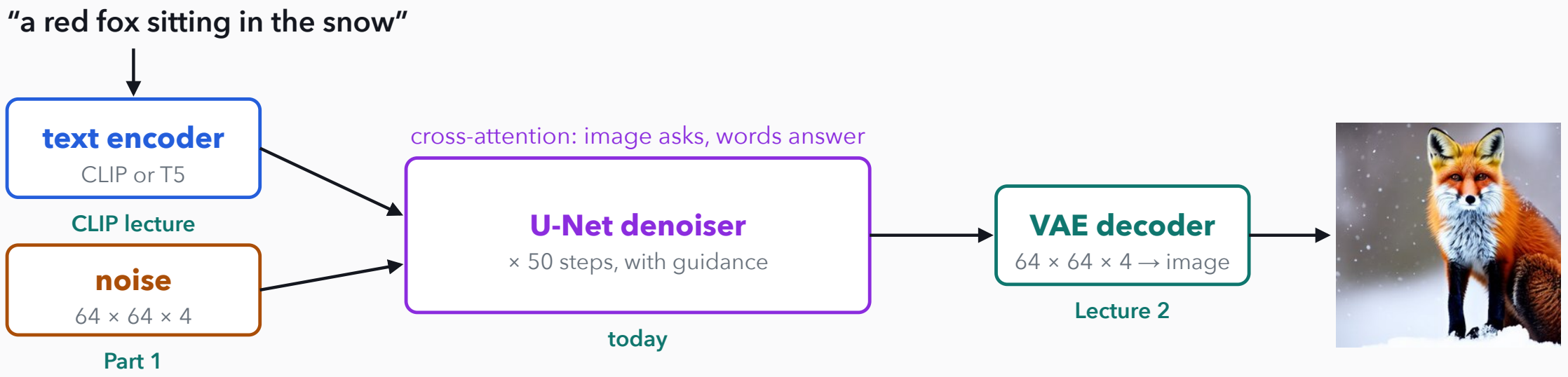
decoded: almost the same
(30.9 dB)

> Decoded back, the photo is almost unchanged (30.9 dB). Stable Diffusion's U-Net works only on these small codes; the decoder runs once, at the very end.

Rombach et al., "High-Resolution Image Synthesis with Latent Diffusion Models" (2022) · Stable Diffusion v1.5, run for the companion film "Diffusion models, from noise to images, step by step" (2026)

▶ Notebook 7.5

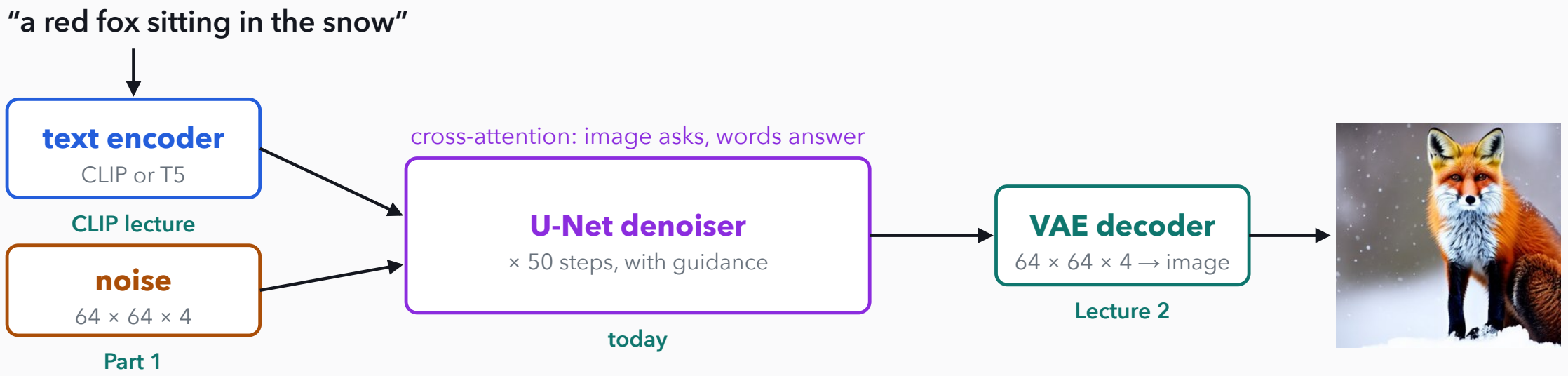
Text in, through cross-attention



> A label was one embedding added everywhere. A sentence is many word vectors, and different parts of the image care about different words: cross-attention lets each position attend to the words, exactly the attention of our attention lectures.

▶ Notebook 7.5

Text in, through cross-attention

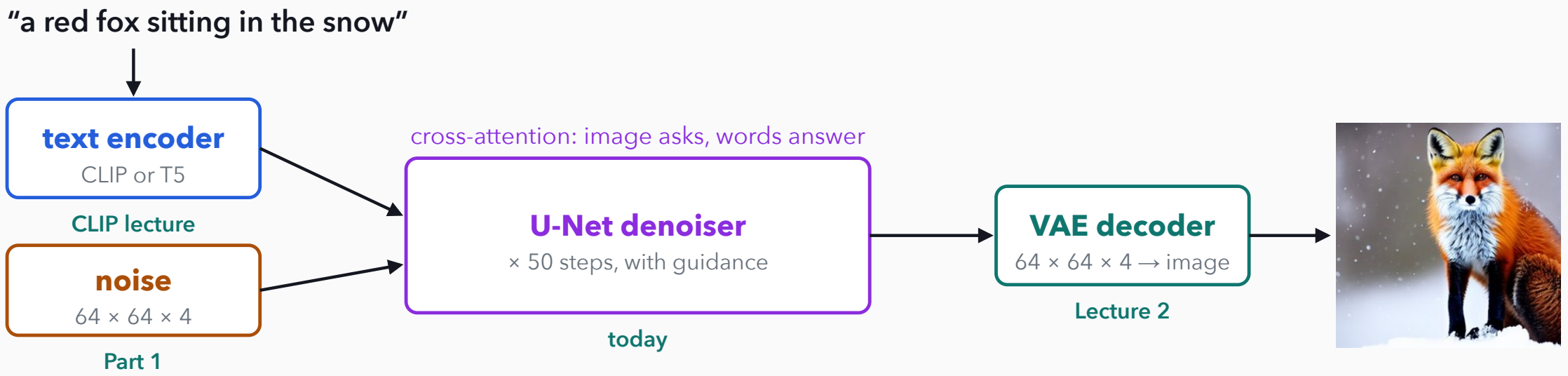


Cross-attention: each image position asks the words a question. Queries come from the U-Net's features; keys and values from the word vectors.

> The whole pipeline: text encoder, a U-Net denoiser on 64 × 64 × 4 latents with guidance, then the VAE decoder. Every box is a lecture you have had.

▶ Notebook 7.5

Text in, through cross-attention



Cross-attention: each image position asks the words a question. Queries come from the U-Net's features; keys and values from the word vectors.

> Nothing new in the training loop: image codes and captions instead of digits and labels.

▶ Notebook 7.5

Which words go where: the real cross-attention



the picture | “red” | “fox” | “sitting” | “snow”

> Stable Diffusion, asked for “a red fox sitting in the snow”. For each word, the parts of the picture that attend to it are lit up. The fox attends to “fox”, “red” and “sitting”.

Hertz et al., “Prompt-to-Prompt Image Editing with Cross Attention Control” (2022) · Stable Diffusion v1.5, run for the companion film “Diffusion models, from noise to images, step by step” (2026)

▶ Notebook 7.5

Which words go where: the real cross-attention



the picture | “red” | “fox” | “sitting” | “snow”

Bright: this part of the picture attends to the word. Each map is averaged over heads, layers and all 50 steps.

> The snow and the background attend to “snow”, and the fox does not. The sentence is not squeezed into one vector: each region reads the words it needs, at every step.

▶ Notebook 7.6

A sentence in, a picture out: 50 steps



"a red fox sitting in fresh snow…"



"a peacock with its tail feathers open…"



"a lighthouse on a rocky cliff at sunset, oil painting"



"a watercolor painting of a hummingbird…"



"glowing jellyfish in the deep ocean…"

step 0 of 50

> The real Stable Diffusion: five sentences, five starting noises, one network and the same 50-step loop with guidance. Press play.

Rombach et al., "High-Resolution Image Synthesis with Latent Diffusion Models" (2022) · Stable Diffusion v1.5, run for the companion film "Diffusion models, from noise to images, step by step" (2026)

▶ Notebook 7.6

A sentence in, a picture out: 50 steps



"a red fox sitting in fresh snow…"



"a peacock with its tail feathers open…"



"a lighthouse on a rocky cliff at sunset, oil painting"



"a watercolor painting of a hummingbird…"



"glowing jellyfish in the deep ocean…"

step 50 of 50

> The same story as our digits and the cats: blotches of colour first (noise in the latent space, decoded), then a layout, then detail. Only the sentence differs.

Rombach et al., "High-Resolution Image Synthesis with Latent Diffusion Models" (2022) · Stable Diffusion v1.5, run for the companion film "Diffusion models, from noise to images, step by step" (2026)

▶ Notebook 7.6

The guidance dial on a sentence: w = 0 to 15



w = 0



w = 1



w = 3



w = 7.5 (default)



w = 15

> "A red fox sitting in fresh snow…", the same starting noise, five guidance strengths. At w = 0 the sentence is ignored: an unrelated picture. At w = 1, the plain conditional model, a dull, greyish fox.

Ho & Salimans, "Classifier-Free Diffusion Guidance" (2022) · Paras Chopra, "Diffusion Models from First Principles" (2025) (XII) · Stable Diffusion v1.5, run for the companion film "Diffusion models, from noise to images, step by step" (2026)

▶ Notebook 7.6

The guidance dial on a sentence: w = 0 to 15



w = 0



w = 1



w = 3



w = 7.5 (default)



w = 15

$$\hat{\varepsilon}_{\text{guided}} = \hat{\varepsilon}(x_t, t, \varnothing) + w\left(\hat{\varepsilon}(x_t, t, \text{sentence}) - \hat{\varepsilon}(x_t, t, \varnothing)\right)$$

> From w = 3 to 7.5, the fox we asked for; 7.5 is the usual default. At w = 15 the colours burn and the picture looks over-processed. The same trade as our 3s (7.3): more of the prompt, less natural.

Ho & Salimans, "Classifier-Free Diffusion Guidance" (2022) · Paras Chopra, "Diffusion Models from First Principles" (2025) (XII) · Stable Diffusion v1.5, run for the companion film "Diffusion models, from noise to images, step by step" (2026)

▶ Notebook 8.1

Three generators, side by side

	VAE (Lecture 2)	GAN (Lecture 3)	Diffusion (today)
training signal	ELBO: reconstruct + KL	a discriminator's verdict	ELBO, reweighted: predict the noise
to sample	decode z ~ N(0, I)	generate from z ~ N(0, I)	denoise x_T ~ N(0, I), step by step
network calls per sample	1	1	20–1,000
likelihood	a lower bound	none	a lower bound
typical failure	blurry samples	unstable training, collapse	slow sampling

> The three generators of this course. Diffusion keeps the VAE's stable, likelihood-based training and gets the GAN's sharpness, and pays in sampling time.

▶ Notebook 8.1

Exit questions

Q1 What does the network see in training, and what is its target?

x_t and t; the noise ε we drew.

Q2 Why must t be an input?

Q3 What happens in one reverse step?

Q4 Why add noise while sampling?

Q5 How does a sentence steer the image?

> Q1: training.

▶ Notebook 8.1

Exit questions

Q1 What does the network see in training, and what is its target?

x_t and t; the noise ε we drew.

Q2 Why must t be an input?

One set of weights serves every noise level; the same x_t means different things at different t.

Q3 What happens in one reverse step?

Q4 Why add noise while sampling?

Q5 How does a sentence steer the image?

> Q2: time.

▶ Notebook 8.1

Exit questions

Q1 What does the network see in training, and what is its target?

x_t and t; the noise ε we drew.

Q2 Why must t be an input?

One set of weights serves every noise level; the same x_t means different things at different t.

Q3 What happens in one reverse step?

Guess $\hat{\varepsilon}$, turn it into $\hat{x}_0$, compute the mean μ, add a little fresh noise.

Q4 Why add noise while sampling?

Q5 How does a sentence steer the image?

> Q3: one step back.

▶ Notebook 8.1

Exit questions

Q1 What does the network see in training, and what is its target?

x_t and t; the noise ε we drew.

Q2 Why must t be an input?

One set of weights serves every noise level; the same x_t means different things at different t.

Q3 What happens in one reverse step?

Guess $\hat{\varepsilon}$, turn it into $\hat{x}_0$, compute the mean μ, add a little fresh noise.

Q4 Why add noise while sampling?

Each step is a draw from a Gaussian; without the noise all samples collapse to one average.

Q5 How does a sentence steer the image?

> Q4: sampling noise.

▶ Notebook 8.1

Exit questions

Q1 What does the network see in training, and what is its target?

x_t and t; the noise ε we drew.

Q2 Why must t be an input?

One set of weights serves every noise level; the same x_t means different things at different t.

Q3 What happens in one reverse step?

Guess $\hat{\epsilon}$, turn it into $\hat{x}_0$, compute the mean μ, add a little fresh noise.

Q4 Why add noise while sampling?

Each step is a draw from a Gaussian; without the noise all samples collapse to one average.

Q5 How does a sentence steer the image?

Its word vectors enter the U-Net through cross-attention; guidance pushes along the difference between the guesses with and without the text.

> Q5: text.

▶ Notebook 8.1

References and credits

Explainer and videos this lecture follows. Paras Chopra, "Diffusion Models from First Principles" (2025) · Ari Seff, "What are Diffusion Models?" (2022) · Deepia, "Diffusion Models: DDPM, Generative AI Animated" (2025) · Outlier, "Diffusion Models, Paper Explanation, Math Explained" (2022) · Vizuara, "Principles of Diffusion Models", Lecture 1 (2025) · MIT 6.S183 "A Practical Introduction to Diffusion Models" (IAP 2025), Lectures 1–6 (Yuan, Becker, Chen, Scarvelis, Lukoianov).

Papers. Sohl-Dickstein et al., "Deep Unsupervised Learning using Nonequilibrium Thermodynamics" (2015) · Ho, Jain & Abbeel, "Denoising Diffusion Probabilistic Models" (2020) · Song & Ermon, "Generative Modeling by Estimating Gradients of the Data Distribution" (2019) · Song et al., "Score-Based Generative Modeling through SDEs" (2021) · Song, Meng & Ermon, "Denoising Diffusion Implicit Models" (2021) · Nichol & Dhariwal, "Improved Denoising Diffusion Probabilistic Models" (2021) · Dhariwal & Nichol, "Diffusion Models Beat GANs on Image Synthesis" (2021) · Ho & Salimans, "Classifier-Free Diffusion Guidance" (2022) · Rombach et al., "High-Resolution Image Synthesis with Latent Diffusion Models" (2022) · Karras et al., "Elucidating the Design Space of Diffusion-Based Generative Models" (2022) · Lipman et al., "Flow Matching for Generative Modeling" (2023) · Liu et al., "Flow Straight and Fast" (2023) · Peebles & Xie, "Scalable Diffusion Models with Transformers" (2023) · Kadkhodaie et al., "Generalization in diffusion models arises from geometry-adaptive harmonic representations" (2024) · Luo et al., "Latent Consistency Models" (2023) · Hertz et al., "Prompt-to-Prompt Image Editing with Cross Attention Control" (2022).

Tutorials. Lilian Weng, "What are Diffusion Models?" (2021) · Calvin Luo, "Understanding Diffusion Models: A Unified Perspective" (2022) · Sander Dieleman's blog (sander.ai).

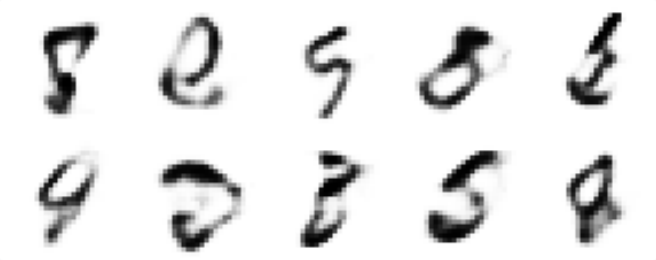
Published images (opening): LDM, RePaint, SR3, SDEdit, ControlNet, DreamBooth, each from the paper's own figures. **Real photos** (0.1, 1.5, 3.5, 4.4): the original DDPM cat checkpoint (google/ddpm-cat-256) and the CC0 photo "Chelsea" from scikit-image; (6.3, 7.4, 7.5, 7.6, 7.6): Stable Diffusion v1.5 with the LCM-LoRA adapter. Both run for the companion film "Diffusion models, from noise to images, step by step" (2026). **Illustration:** ink in water, generated with Gemini.

Our models and data. MNIST (LeCun et al.); the MNIST DDPM, conditional DDPM, autoencoder, latent DDPM and classifier from the Generative Modelling II rebuild; the smiley models trained for this lecture. Gemini (Google) watched the videos and drafted a plan; Claude (Anthropic) built the lecture.

> Every source used, with the slides that use it credited in their footers.

Two generators so far, and a third idea

VAE (Lecture 2)

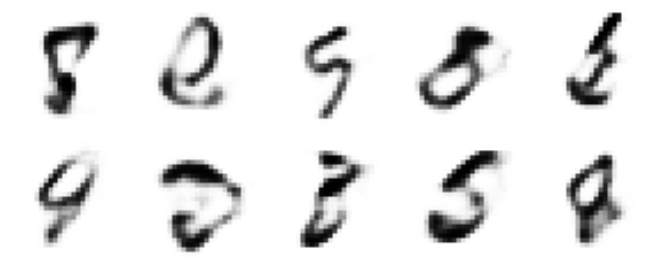


one decoder step from z: honest likelihood, blurry digits

> The VAE decodes a random code in one step: it has a likelihood and trains stably, but its digits are blurry.

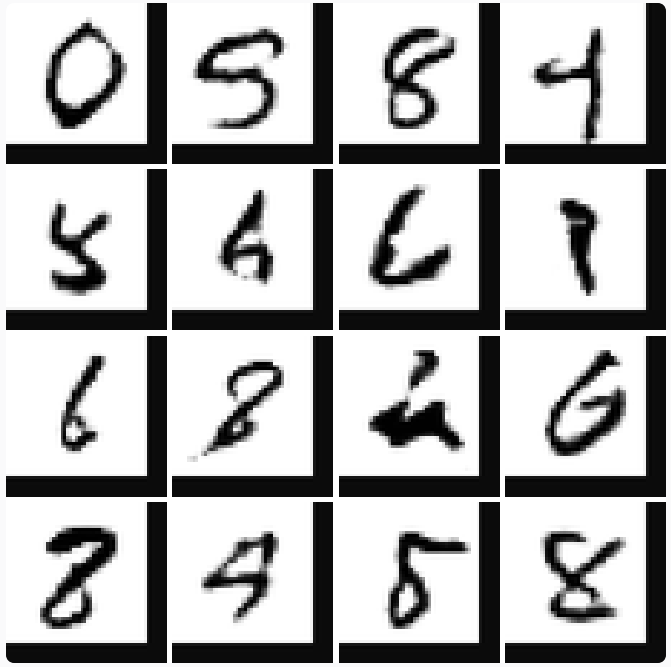
Two generators so far, and a third idea

VAE (Lecture 2)



one decoder step from z: honest likelihood, blurry digits

GAN (Lecture 3)

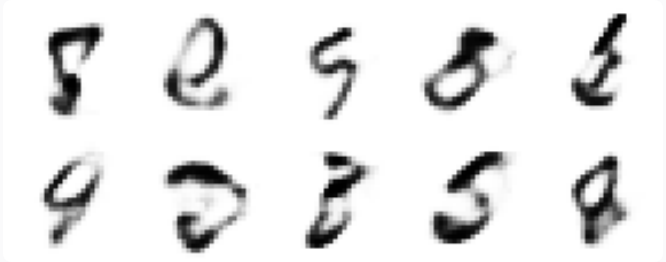


one generator step: sharp, but a fragile game that can collapse

> The GAN also makes a digit in one step: sharp, but training is a game that can collapse.

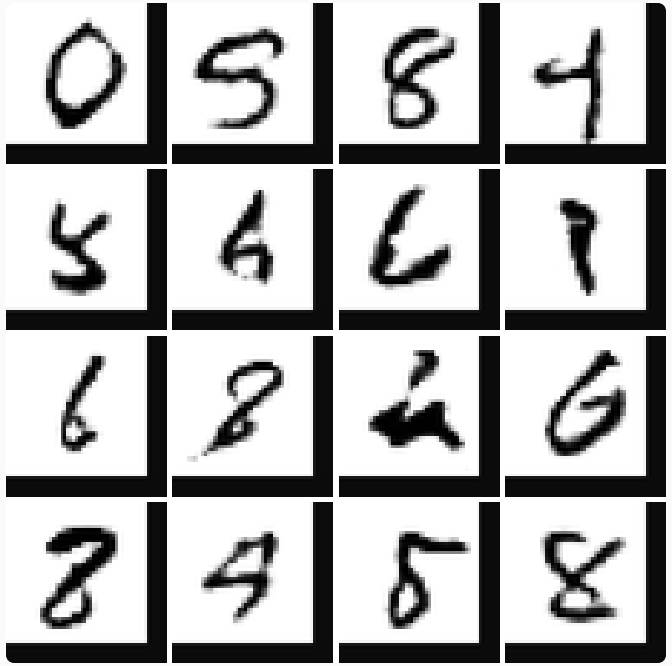
Two generators so far, and a third idea

VAE (Lecture 2)



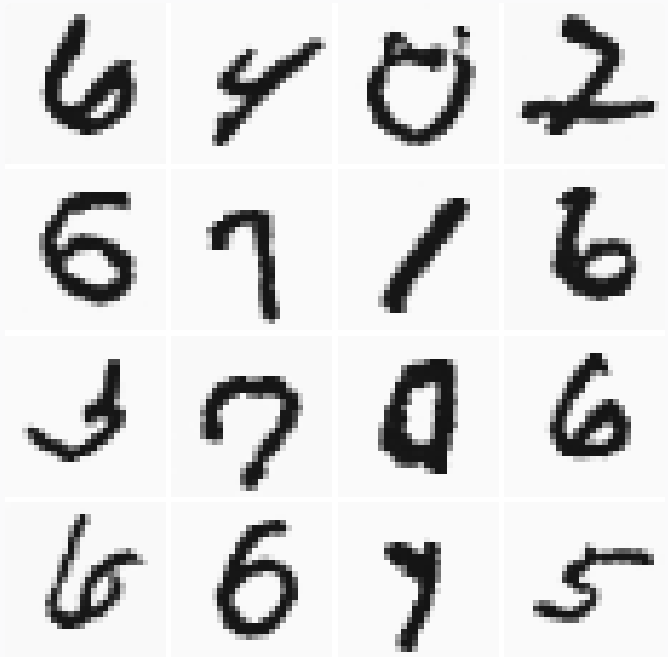
one decoder step from z: honest likelihood, blurry digits

GAN (Lecture 3)



one generator step: sharp, but a fragile game that can collapse

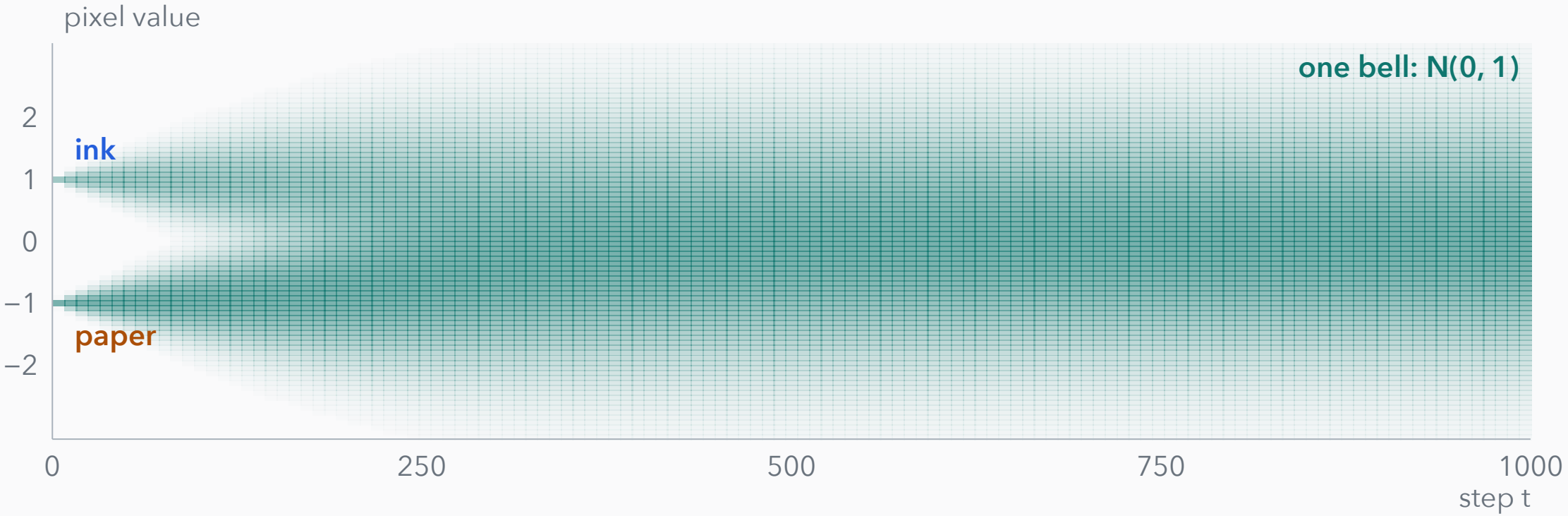
Diffusion (today)



a thousand small denoising steps, each one an easy regression

> Diffusion makes a digit in a thousand small steps. Each step is a simple regression problem, and the result is sharp.

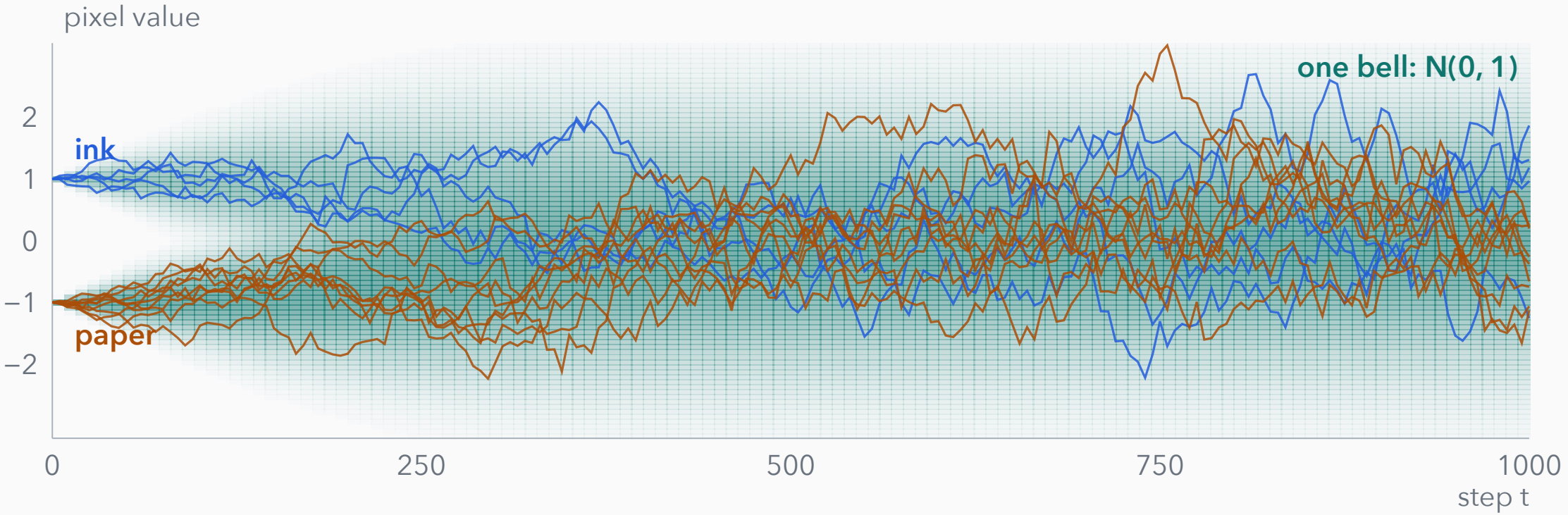
One pixel, a thousand steps: the density flows



> Our pixel across the dataset: ink (+1) with probability 0.35, paper (−1) otherwise. Its density over time, exactly: two thin lines that widen and merge into one bell.

Deepia, “Diffusion Models: DDPM, Generative AI Animated” (2025)

One pixel, a thousand steps: the density flows



> Fourteen noising runs, step by step. Each starts at ink or paper and wanders; by the end you cannot tell where any of them started.

Deepia, “Diffusion Models: DDPM, Generative AI Animated” (2025)

Do we need a 1,000-step loop for one noisy example?

$$x_1 = \sqrt{\alpha_1}\, x_0 + \sqrt{\beta_1}\, \varepsilon_1 \qquad (\alpha_t = 1 - \beta_t)$$

> Name the shrink factor: $\alpha_t = 1 - \beta_t$. One step: $x_1 = \sqrt{\alpha_1}\, x_0 + \sqrt{\beta_1}\, \varepsilon_1$.

Ari Seff, "What are Diffusion Models?" (2022)

Do we need a 1,000-step loop for one noisy example?

$$x_1 = \sqrt{\alpha_1}\, x_0 + \sqrt{\beta_1}\, \varepsilon_1 \qquad (\alpha_t = 1 - \beta_t)$$

$$x_2 = \sqrt{\alpha_2}\left(\sqrt{\alpha_1}\, x_0 + \sqrt{\beta_1}\, \varepsilon_1\right) + \sqrt{\beta_2}\, \varepsilon_2$$

> Two steps: substitute x_1 into the rule for x_2.

Ari Seff, "What are Diffusion Models?" (2022)

Do we need a 1,000-step loop for one noisy example?

$$x_1 = \sqrt{\alpha_1}\, x_0 + \sqrt{\beta_1}\, \varepsilon_1 \qquad (\alpha_t = 1 - \beta_t)$$

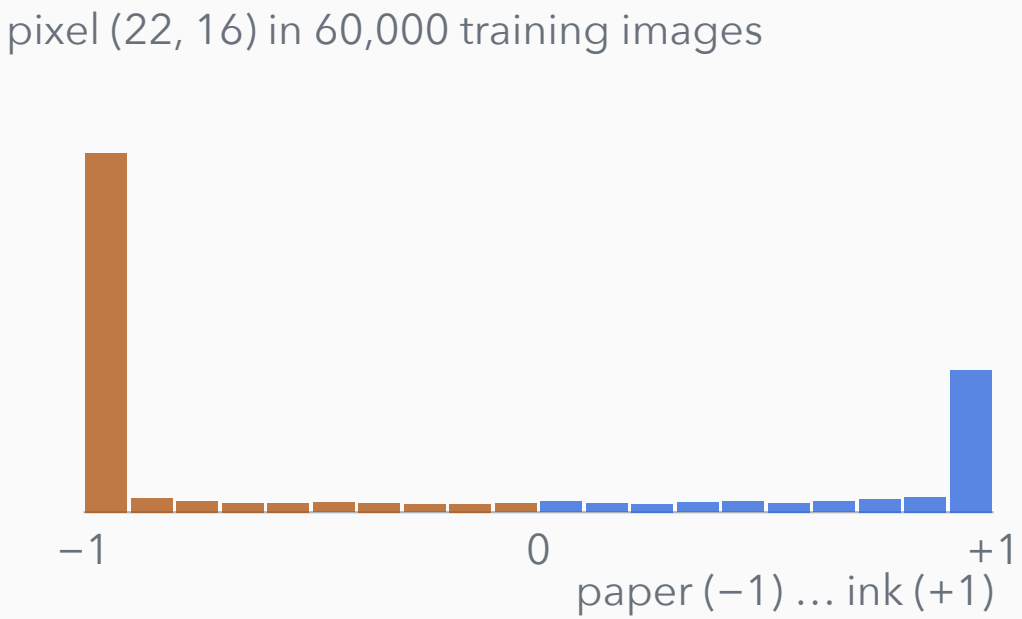
$$x_2 = \sqrt{\alpha_2}\left(\sqrt{\alpha_1}\, x_0 + \sqrt{\beta_1}\, \varepsilon_1\right) + \sqrt{\beta_2}\, \varepsilon_2$$

$$x_3 = \sqrt{\alpha_3}\left(\sqrt{\alpha_2}\left(\sqrt{\alpha_1}\, x_0 + \sqrt{\beta_1}\, \varepsilon_1\right) + \sqrt{\beta_2}\, \varepsilon_2\right) + \sqrt{\beta_3}\, \varepsilon_3 \cdots$$

> Three steps, and it keeps nesting. Training needs millions of noisy examples at random t: we cannot afford this loop each time.

Ari Seff, “What are Diffusion Models?” (2022)

One pixel: paper or ink

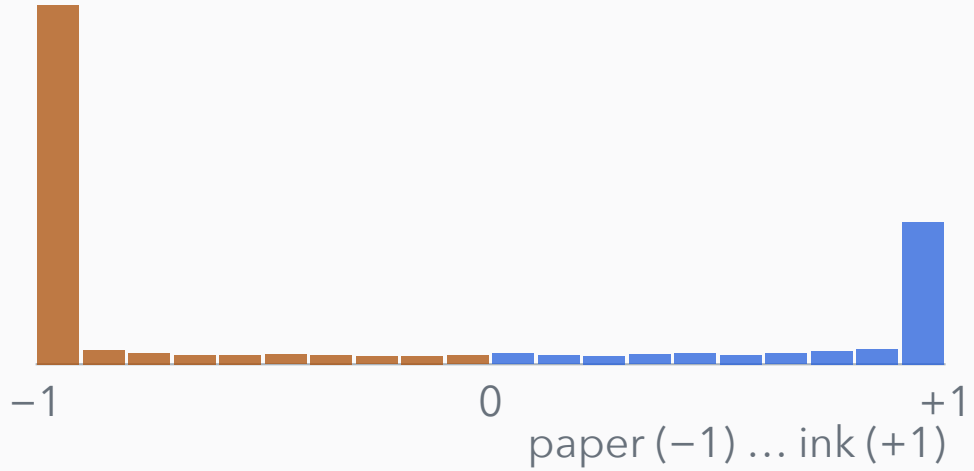


Across the 60,000 training images, our pixel is ink (darker than mid-grey) in 35% of them.

> Our pixel across the whole training set: mostly paper, often ink (35% of images), sometimes in between.

One pixel: paper or ink

pixel (22, 16) in 60,000 training images



Across the 60,000 training images, our pixel is ink (darker than mid-grey) in 35% of them.

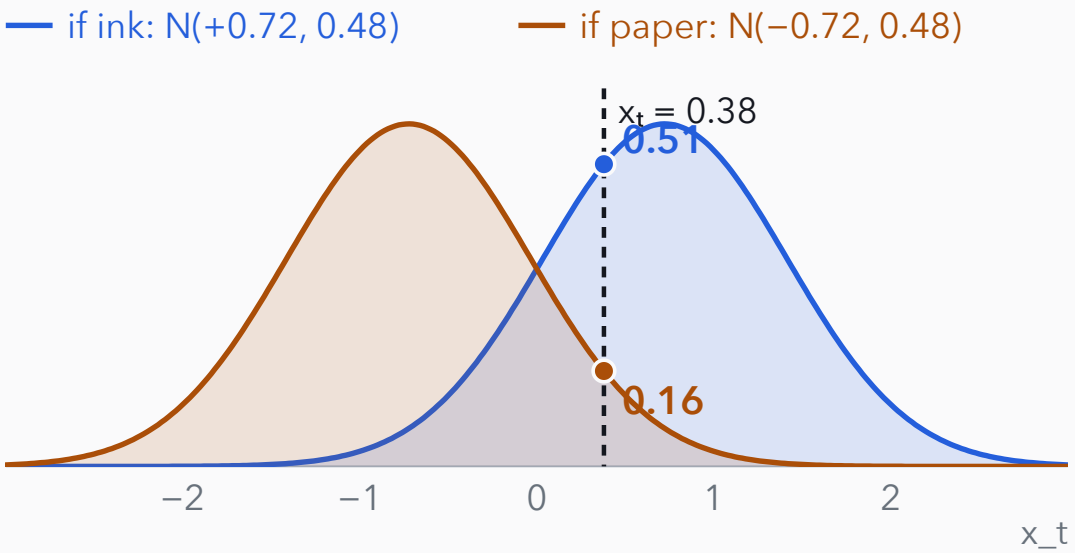
$$x_0 = +1 \text{ (ink) with probability } \pi = 0.35$$

$$x_0 = -1 \text{ (paper) otherwise}$$

A simplification: about 23% of the time this pixel is a grey in between. Ignoring the greys makes every formula exact.

> Model it as a coin: ink with probability π = 0.35, paper otherwise. This is the prior over the clean pixel, and it is all Bayes needs.

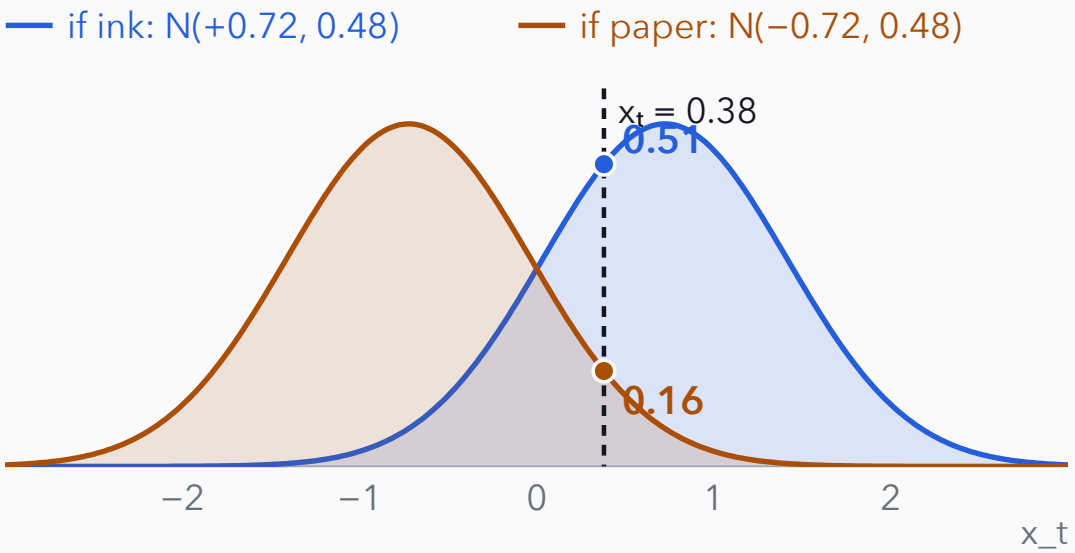
We see $x_{250} = 0.38$. Was the pixel ink?



$$p(x_{250} \mid \text{ink}) = 0.51, \qquad p(x_{250} \mid \text{paper}) = 0.16$$

If the pixel was ink, x_{250} comes from N(+0.72, 0.48); if paper, from N(−0.72, 0.48). The observed 0.38 is three times more likely under ink.

We see $x_{250} = 0.38$. Was the pixel ink?

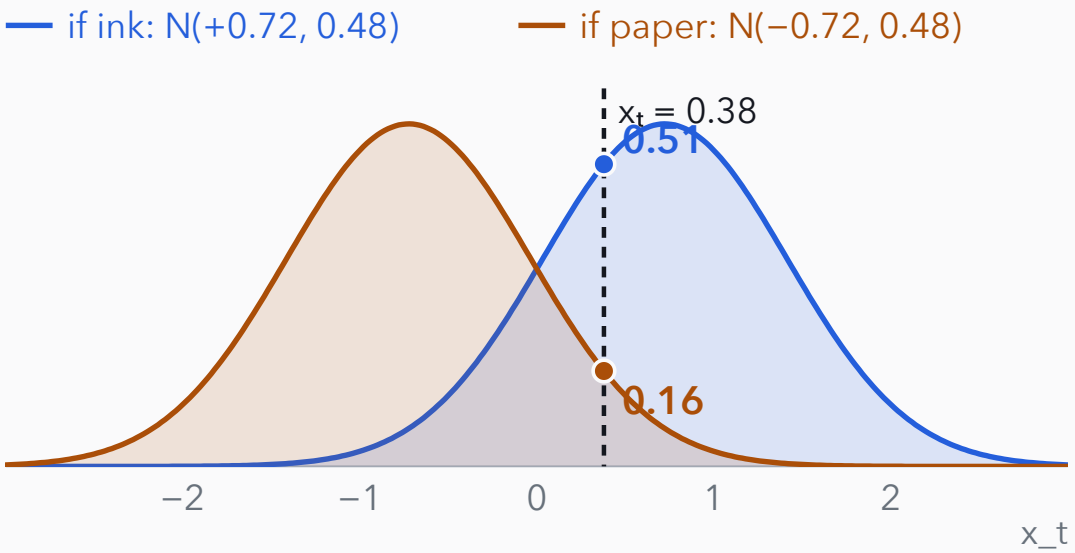


$$p(x_{250} \mid \text{ink}) = 0.51, \qquad p(x_{250} \mid \text{paper}) = 0.16$$

$$P(\text{ink} \mid x_{250}) = \frac{0.35 \times 0.51}{0.35 \times 0.51 + 0.65 \times 0.16} = 0.63$$

Bayes, with the prior 0.35 for ink: P(ink | x_{250}) = 0.63. The prior pulls it down: paper is more common.

We see $x_{250} = 0.38$. Was the pixel ink?



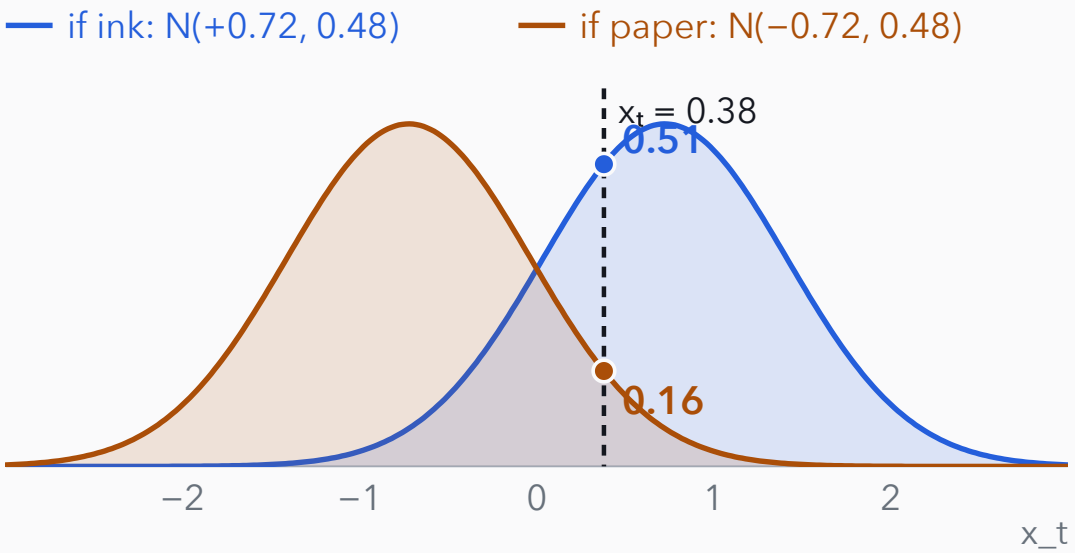
$$p(x_{250} \mid \text{ink}) = 0.51, \qquad p(x_{250} \mid \text{paper}) = 0.16$$

$$P(\text{ink} \mid x_{250}) = \frac{0.35 \times 0.51}{0.35 \times 0.51 + 0.65 \times 0.16} = 0.63$$

$$\hat{x}_0 = \mathbb{E}[x_0 \mid x_{250}] = 0.63\,(+1) + 0.37\,(-1) = 0.26$$

> The best single guess of x_0 under squared error is the posterior average: 0.26. A grey: probably ink, but not surely.

We see $x_{250} = 0.38$. Was the pixel ink?



$$p(x_{250} \mid \text{ink}) = 0.51, \qquad p(x_{250} \mid \text{paper}) = 0.16$$

$$P(\text{ink} \mid x_{250}) = \frac{0.35 \times 0.51}{0.35 \times 0.51 + 0.65 \times 0.16} = 0.63$$

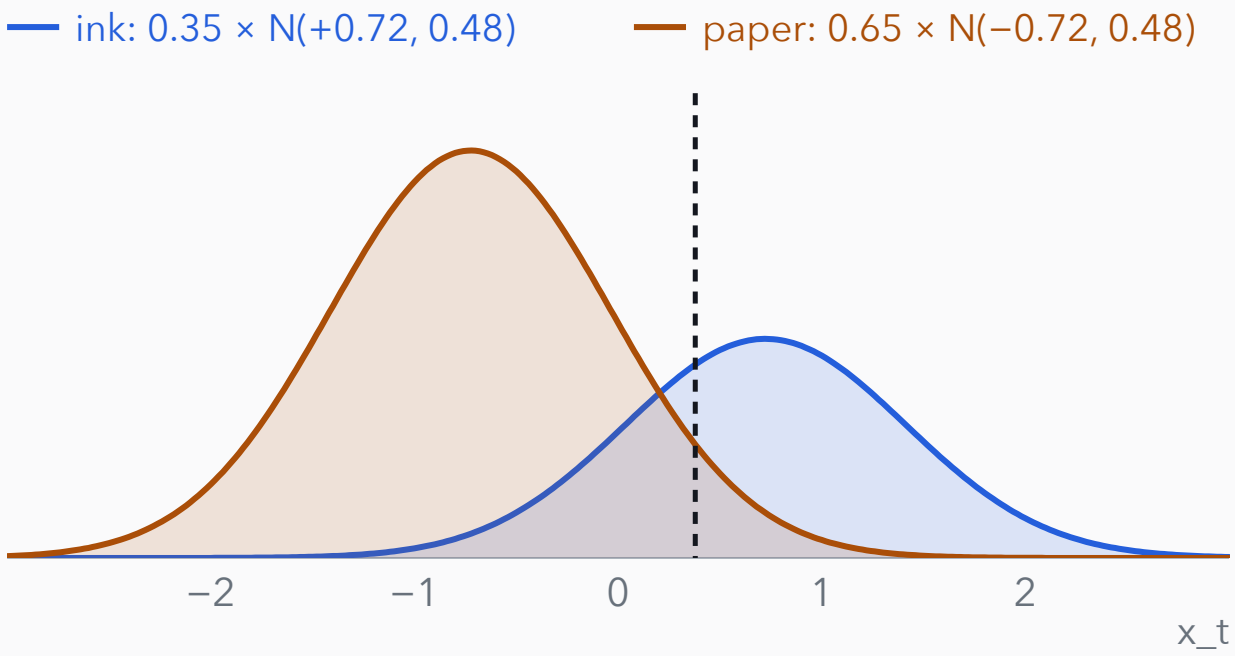
$$\hat{x}_0 = \mathbb{E}[x_0 \mid x_{250}] = 0.63\,(+1) + 0.37\,(-1) = 0.26$$

$$P(\text{ink} \mid x_t) = \sigma\!\left(\log \tfrac{\pi}{1-\pi} + \tfrac{2\sqrt{\bar{\alpha}_t}}{1-\bar{\alpha}_t}\, x_t \right)$$

In general the posterior is a sigmoid of x_t: the log of the likelihood ratio is a straight line in x_t, as in logistic regression.

▶ Notebook

Move t and x_t: the posterior and the best guess



t = 250, x_t = 0.38

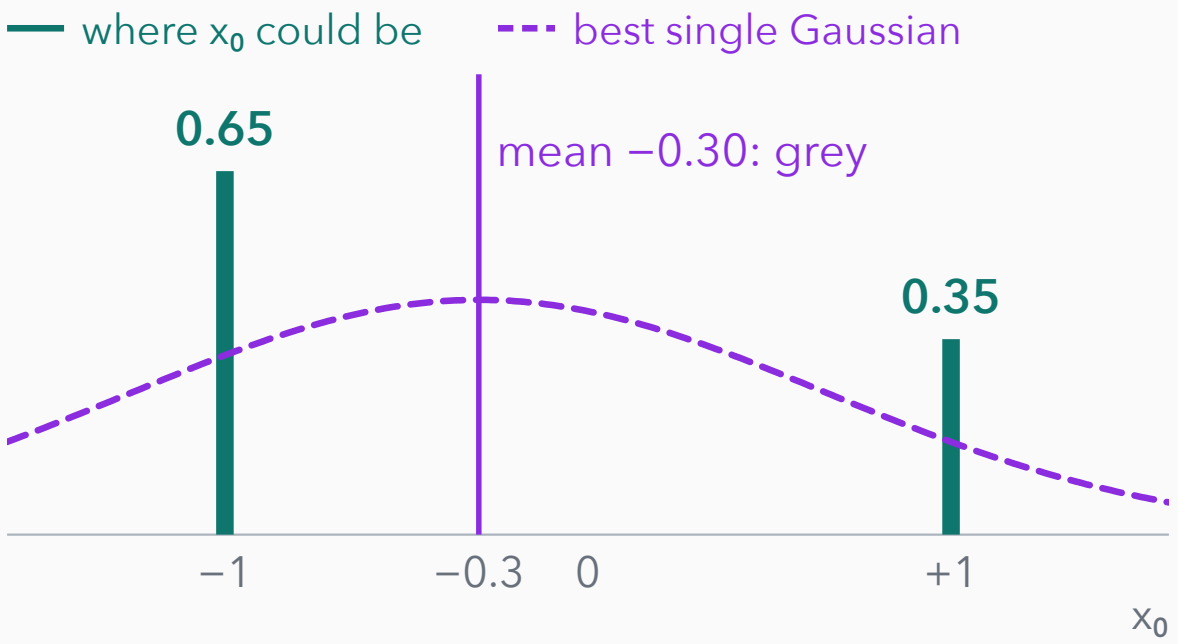
P(ink | x_t) = 0.63 paper 0.37

best guess $E[x_0 \mid x_t] = 0.26$

a grey: neither paper nor ink

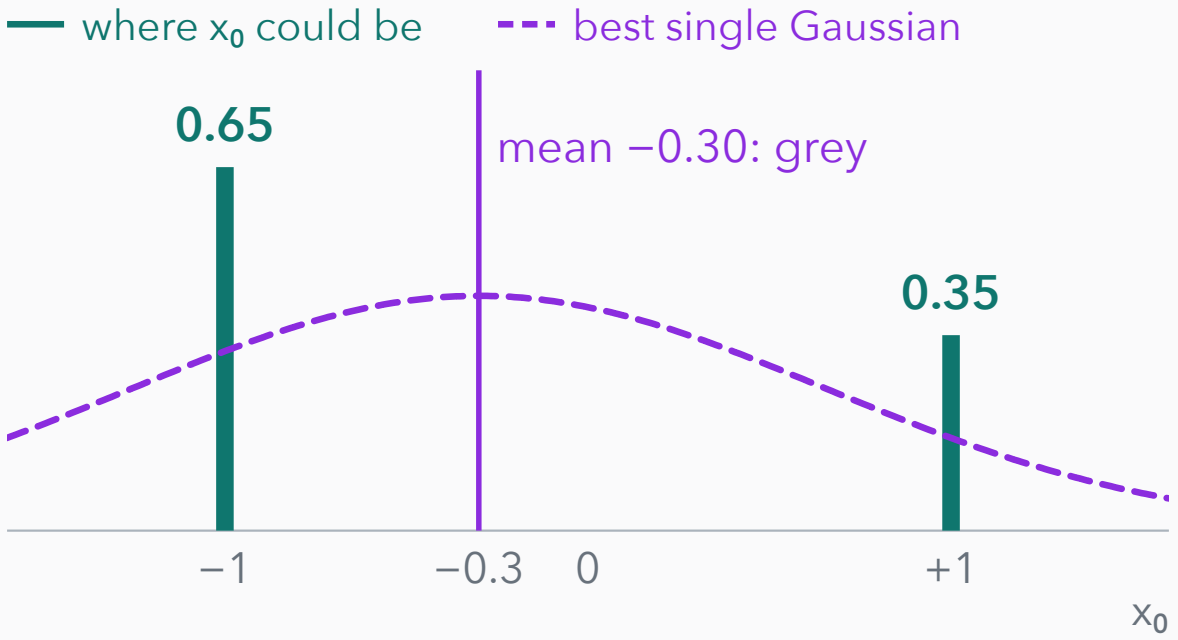
> Move x_t: the posterior slides from paper to ink. Move t: at small t the two bumps are far apart and any x_t is decisive; at large t they overlap and the answer falls back to the prior.

From pure noise straight back to the pixel: two spikes



> From x_T, pure noise, back to x_0 in one jump: x_T says nothing about the pixel, so the answer is the prior: paper with 0.65, ink with 0.35. Two spikes.

From pure noise straight back to the pixel: two spikes

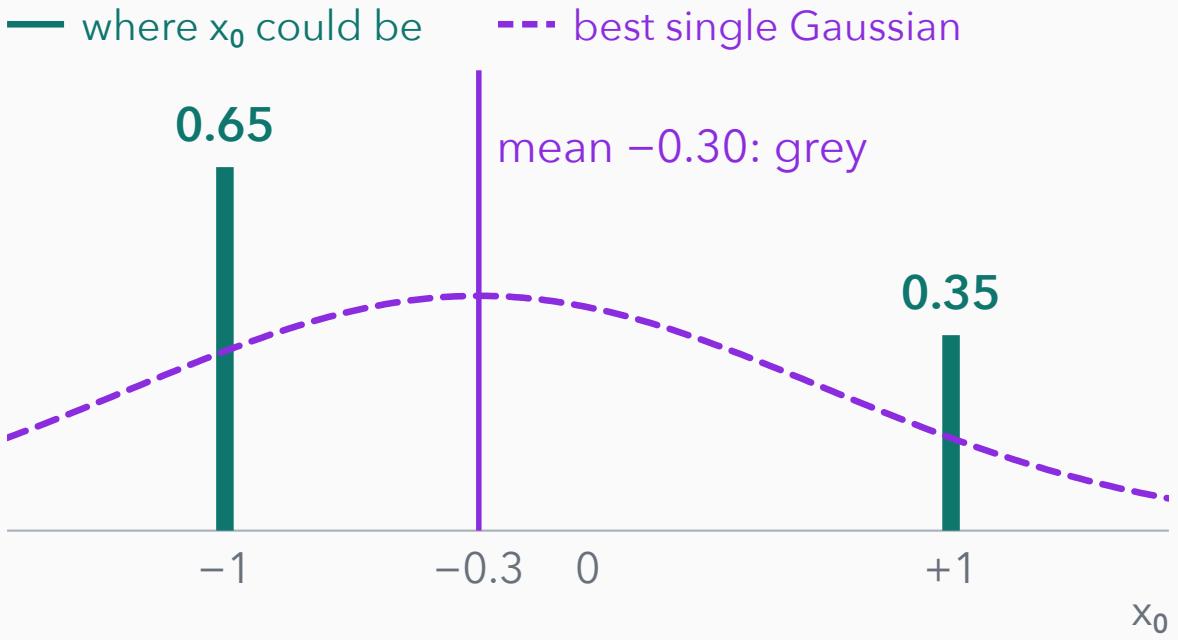


One Gaussian has one bump: its best fit sits at the average, a grey that is neither paper nor ink.

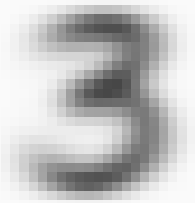
> One Gaussian cannot be two spikes. Its best fit is centred at the average, −0.30: a grey pixel nobody drew.

MIT 6.S183 "A Practical Introduction to Diffusion Models" (IAP 2025), Lecture 2 (the "100 cats" question)

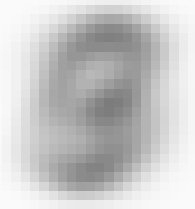
From pure noise straight back to the pixel: two spikes



One Gaussian has one bump: its best fit sits at the average, a grey that is neither paper nor ink.



average of 6,131 training 3s



average of all digits

> For whole images the same thing happens: a one-step denoiser asked for "a digit" returns the average digit, a blur. Average a hundred cats and you get no cat at all.

Why: a narrow likelihood times a smooth prior

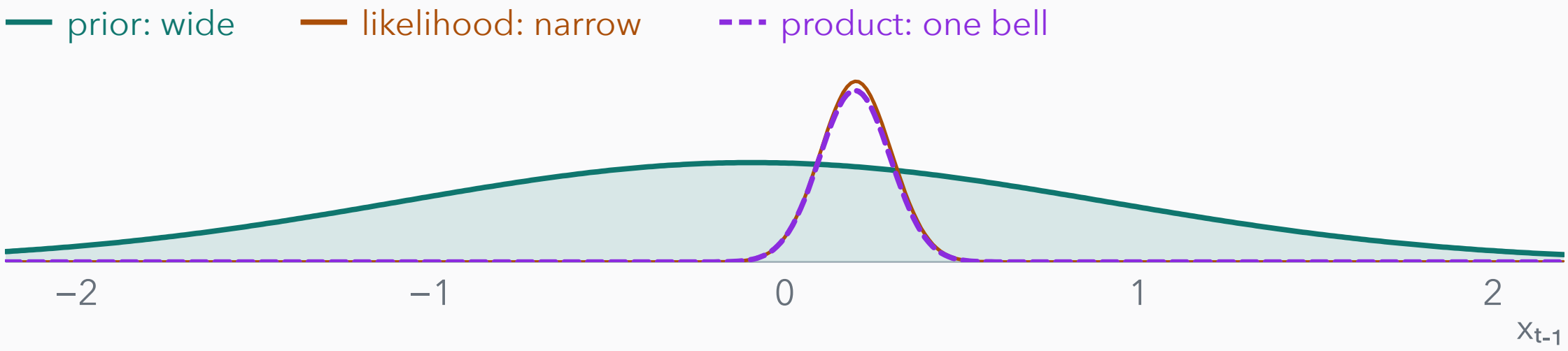
$$q(x_{t-1} \mid x_t) \;\propto\; \underbrace{\mathcal{N}\left(x_t;\, \sqrt{\alpha_t}\, x_{t-1},\, \beta_t\right)}_{\text{narrow: } \beta_t \le 0.02} \;\times\; \underbrace{q(x_{t-1})}_{\text{smooth}}$$

> Bayes again for one step: the likelihood is a narrow bump, because β_t is at most 0.02.

Ari Seff, "What are Diffusion Models?" (2022) · Sohl-Dickstein et al., "Deep Unsupervised Learning using Nonequilibrium Thermodynamics" (2015)

Why: a narrow likelihood times a smooth prior

$$q(x_{t-1} \mid x_t) \propto \underbrace{\mathcal{N}\left(x_t; \sqrt{\alpha_t}\, x_{t-1}, \beta_t\right)}_{\text{narrow: } \beta_t \leq 0.02} \times \underbrace{q(x_{t-1})}_{\text{smooth}}$$

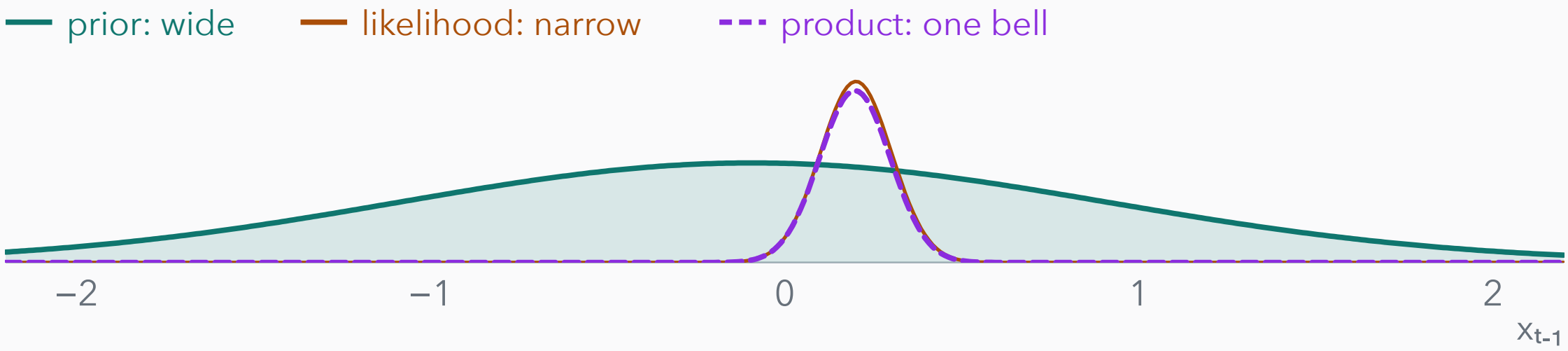


> Over such a narrow range the prior hardly changes, so the product is one narrow bell, centred near x_t and nudged towards where the prior is higher.

Ari Seff, "What are Diffusion Models?" (2022) · Sohl-Dickstein et al., "Deep Unsupervised Learning using Nonequilibrium Thermodynamics" (2015)

Why: a narrow likelihood times a smooth prior

$$q(x_{t-1} \mid x_t) \;\propto\; \underbrace{\mathcal{N}\left(x_t;\, \sqrt{\alpha_t}\, x_{t-1},\, \beta_t\right)}_{\text{narrow: } \beta_t \le 0.02} \;\times\; \underbrace{q(x_{t-1})}_{\text{smooth}}$$



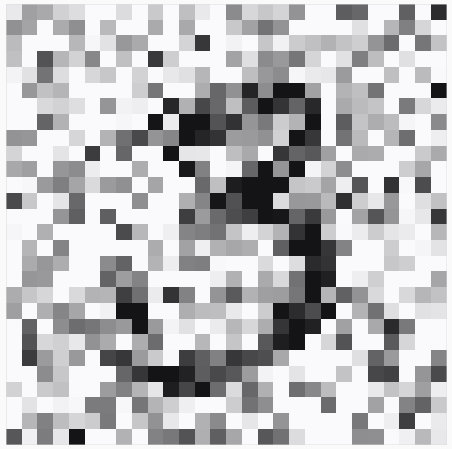
$$p_\theta(x_{t-1} \mid x_t) = \mathcal{N}\left(x_{t-1};\; \mu_\theta(x_t, t),\; \sigma_t^2\right)$$

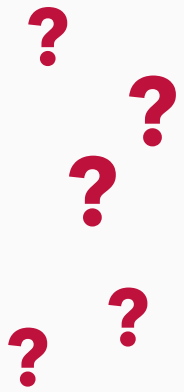
> So we will learn one Gaussian per step: its mean comes from a network μ_θ(x_t, t); its width σ_t is fixed by the schedule. A thousand such steps, and the network is shared.

Ari Seff, "What are Diffusion Models?" (2022) · Sohl-Dickstein et al., "Deep Unsupervised Learning using Nonequilibrium Thermodynamics" (2015)

If we knew the clean image, the step back would be exact

x_{250}: noisy





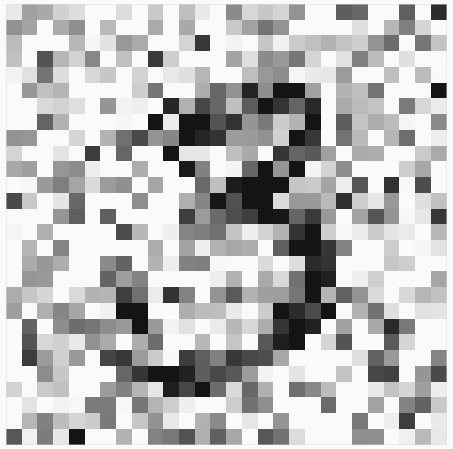
Where was x_{249}?

It could have come from many different images.

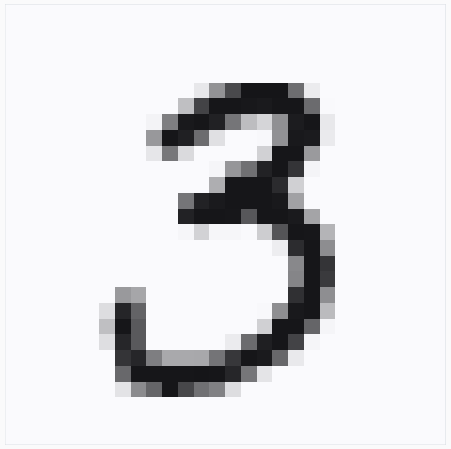
> Given only the noisy 3, the step back is uncertain: many clean images could have produced it.

If we knew the clean image, the step back would be exact

x_{250}: noisy



x_0: known



The step back is pinned: one narrow Gaussian

pulled a little towards x_0

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

> Given the clean 3 as well, the step back is pinned down: one narrow Gaussian. During training we always have x_0, so this is a target we can compute.

Outlier, "Diffusion Models, Paper Explanation, Math Explained" (2022) (18:23)

Bayes with three Gaussians

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

> Bayes' rule with x_0 given throughout: likelihood of x_t, times the prior of x_{t−1} given x_0, over the evidence.

Outlier, "Diffusion Models, Paper Explanation, Math Explained" (2022) · Lilian Weng, "What are Diffusion Models?" (2021)

Bayes with three Gaussians

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

$$= \frac{\mathcal{N}\left(x_t; \sqrt{\alpha_t}\, x_{t-1}, \beta_t\right)\; \mathcal{N}\left(x_{t-1}; \sqrt{\bar{\alpha}_{t-1}}\, x_0, 1 - \bar{\alpha}_{t-1}\right)}{\mathcal{N}\left(x_t; \sqrt{\bar{\alpha}_t}\, x_0, 1 - \bar{\alpha}_t\right)}$$

All three are Gaussians we know. The first is one noising step (x_0 adds nothing once x_{t−1} is known: the Markov property). The other two are the shortcut of 1.3.

Outlier, "Diffusion Models, Paper Explanation, Math Explained" (2022) · Lilian Weng, "What are Diffusion Models?" (2021)

Bayes with three Gaussians

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

$$= \frac{\mathcal{N}\left(x_t; \sqrt{\alpha_t}\, x_{t-1}, \beta_t\right)\; \mathcal{N}\left(x_{t-1}; \sqrt{\bar{\alpha}_{t-1}}\, x_0, 1 - \bar{\alpha}_{t-1}\right)}{\mathcal{N}\left(x_t; \sqrt{\bar{\alpha}_t}\, x_0, 1 - \bar{\alpha}_t\right)}$$

$$\log q(x_{t-1} \mid x_t, x_0) = -\frac{1}{2}\left[\frac{\left(x_t - \sqrt{\alpha_t}\, x_{t-1}\right)^2}{\beta_t} + \frac{\left(x_{t-1} - \sqrt{\bar{\alpha}_{t-1}}\, x_0\right)^2}{1 - \bar{\alpha}_{t-1}}\right] + \text{const}$$

> Take logs. The denominator has no x_{t−1}: a constant. What is left is a quadratic in x_{t−1}, and an exponentiated quadratic is a Gaussian.

Outlier, “Diffusion Models, Paper Explanation, Math Explained” (2022) · Lilian Weng, “What are Diffusion Models?” (2021)

Complete the square: the mean and the width

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

> Match the quadratic with −(x − μ̃)²/(2β̃): the coefficient of x_{t−1}² gives the width. It simplifies because $\beta_t = 1 - \alpha_t$.

Ho, Jain & Abbeel, "Denoising Diffusion Probabilistic Models" (2020) · eq. 6–7

Complete the square: the mean and the width

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

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

> The coefficient of x_{t−1} gives the mean: a weighted mix of the clean value and the current noisy value.

Ho, Jain & Abbeel, "Denoising Diffusion Probabilistic Models" (2020) · eq. 6–7

Complete the square: the mean and the width

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

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

$$t = 250: \quad \tilde{\mu} = 0.0077 \times 1 + 0.9919 \times 0.379 = 0.384, \qquad \sqrt{\tilde{\beta}} = 0.071$$

> Our pixel: the step back keeps 99% of x_t and adds 0.8% of x_0: $\tilde{\mu} = 0.384$, with width 0.071. A tiny step.

Ho, Jain & Abbeel, "Denoising Diffusion Probabilistic Models" (2020) · eq. 6–7

We do not have x_0 when generating, so guess it

$$\mu_\theta(x_t, t) = \tilde{\mu}_t\big(x_t,\ \hat{x}_0(x_t, t)\big)$$

> The plan: replace the unknown x_0 in $\tilde{\mu}$ by a guess $\hat{x}_0$ computed from x_t. That guess is the network.

We do not have x_0 when generating, so guess it

$$\mu_\theta(x_t, t) = \tilde{\mu}_t\big(x_t,\ \hat{x}_0(x_t, t)\big)$$

$$\text{pixel:} \quad \mu = 0.0077 \times 0.26 + 0.9919 \times 0.379 = 0.378 \qquad (\text{exact: } 0.384)$$

> With the best possible guess for our pixel, $\hat{x}_0 = 0.26$ from "We see $x_{250} = 0.38$. Was the pixel ink?", the mean is 0.378: within 0.006 of the exact 0.384. Small errors per step, a thousand steps.

We do not have x_0 when generating, so guess it

$$\mu_\theta(x_t, t) = \tilde{\mu}_t\big(x_t,\ \hat{x}_0(x_t, t)\big)$$

$$\text{pixel:} \quad \mu = 0.0077 \times 0.26 + 0.9919 \times 0.379 = 0.378 \qquad (\text{exact: } 0.384)$$

$$\hat{x}_0 = \frac{x_t - \sqrt{1-\bar{\alpha}_t}\ \hat{\varepsilon}}{\sqrt{\bar{\alpha}_t}} \qquad (\text{the shortcut, solved for } x_0)$$

> Guessing x_0 and guessing the noise are the same thing: solve the shortcut for x_0. DDPM's network guesses the noise $\hat{\varepsilon}$.

Substitute: the mean in terms of the noise

$$\tilde{\mu}_t = a\, x_0 + b\, x_t, \qquad a = \frac{\sqrt{\bar{\alpha}_{t-1}}\,\beta_t}{1-\bar{\alpha}_t}, \quad b = \frac{\sqrt{\alpha_t}\,(1-\bar{\alpha}_{t-1})}{1-\bar{\alpha}_t}$$

> The mean is a mix of x_0 and x_t, with weights a and b from the schedule.

Ho, Jain & Abbeel, "Denoising Diffusion Probabilistic Models" (2020) · eq. 11

Substitute: the mean in terms of the noise

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

$$x_0 = \frac{x_t - \sqrt{1-\bar{\alpha}_t}\,\varepsilon}{\sqrt{\bar{\alpha}_t}} \ \Rightarrow\ \tilde{\mu}_t = \left(\frac{a}{\sqrt{\bar{\alpha}_t}} + b\right) x_t - \frac{a\sqrt{1-\bar{\alpha}_t}}{\sqrt{\bar{\alpha}_t}}\,\varepsilon$$

> Replace x_0 by the shortcut solved for x_0: everything becomes x_t and ε.

Ho, Jain & Abbeel, "Denoising Diffusion Probabilistic Models" (2020) · eq. 11

Substitute: the mean in terms of the noise

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

$$x_0 = \frac{x_t - \sqrt{1-\bar{\alpha}_t}\,\varepsilon}{\sqrt{\bar{\alpha}_t}} \ \Rightarrow\ \tilde{\mu}_t = \left(\frac{a}{\sqrt{\bar{\alpha}_t}} + b\right) x_t - \frac{a\sqrt{1-\bar{\alpha}_t}}{\sqrt{\bar{\alpha}_t}}\,\varepsilon$$

$$\tilde{\mu}_t = \frac{1}{\sqrt{\alpha_t}}\left(x_t - \frac{\beta_t}{\sqrt{1-\bar{\alpha}_t}}\,\varepsilon\right) \qquad \frac{1}{0.9974}\left(0.379 - \frac{0.0051}{0.690}(-0.5)\right) = 0.384$$

> Simplify the two coefficients ($\sqrt{\bar{\alpha}_t} = \sqrt{\alpha_t}\,\sqrt{\bar{\alpha}_{t-1}}$ and $\beta + \alpha = 1$): $\tilde{\mu} = (x_t - \beta_t/\sqrt{(1-\bar{\alpha}_t)}\cdot\varepsilon)/\sqrt{\alpha_t}$. Our pixel with its true $\varepsilon = -0.5$: 0.384 again.

Ho, Jain & Abbeel, "Denoising Diffusion Probabilistic Models" (2020) · eq. 11

ε is all the noise since x_0, not the last step's

$$x_t = \sqrt{\bar{\alpha}_t}\, x_0 + \underbrace{\sqrt{1-\bar{\alpha}_t}\, \varepsilon}_{\text{every step's noise, combined}}$$

> The target ε is the combined noise of all t steps, as in 1.3: not the noise added in the last step.

ε is all the noise since x_0, not the last step's

$$x_t = \sqrt{\bar{\alpha}_t}\, x_0 + \underbrace{\sqrt{1-\bar{\alpha}_t}\, \varepsilon}_{\text{every step's noise, combined}}$$

t = 250	size
all noise added since x_0: $\sqrt{(1-\bar{\alpha})}$	0.690
the last step's noise alone: $\sqrt{\beta_{250}}$	0.071

> At t = 250 the last step's noise has size 0.071; all the noise together, 0.690. The network predicts the whole thing.

The same noisy value can come from different noises

$$x_{250} = 0.38: \quad \underbrace{\text{ink},\ \varepsilon = -0.50}_{P=0.63} \quad \text{or} \quad \underbrace{\text{paper},\ \varepsilon = 1.60}_{P=0.37}$$

> Two stories explain $x_{250} = 0.38$: ink with noise −0.50, or paper with noise +1.60. The network cannot tell which.

The same noisy value can come from different noises

$$x_{250} = 0.38: \quad \underbrace{\text{ink},\ \varepsilon = -0.50}_{P=0.63} \quad \text{or} \quad \underbrace{\text{paper},\ \varepsilon = 1.60}_{P=0.37}$$

$$\hat{\varepsilon}^* = \mathbb{E}[\varepsilon \mid x_t] = 0.63 \times (-0.50) + 0.37 \times 1.60 = 0.28$$

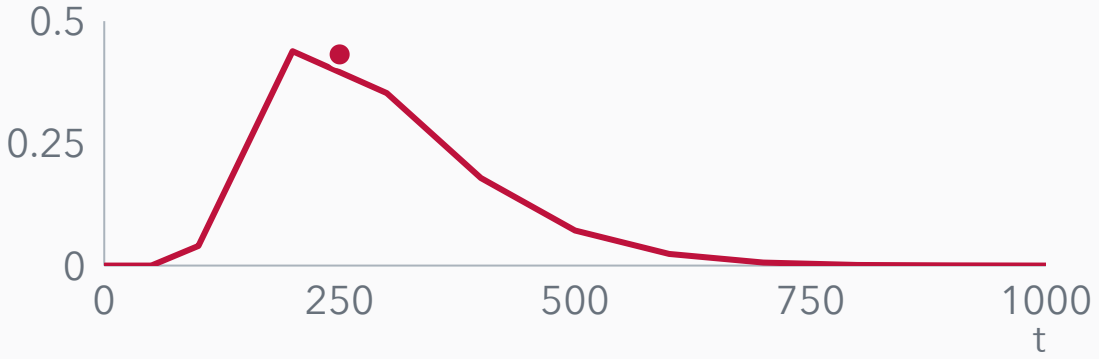
> The squared error is smallest for the average noise, weighted by how likely each story is: 0.28. The network learns this conditional mean.

The same noisy value can come from different noises

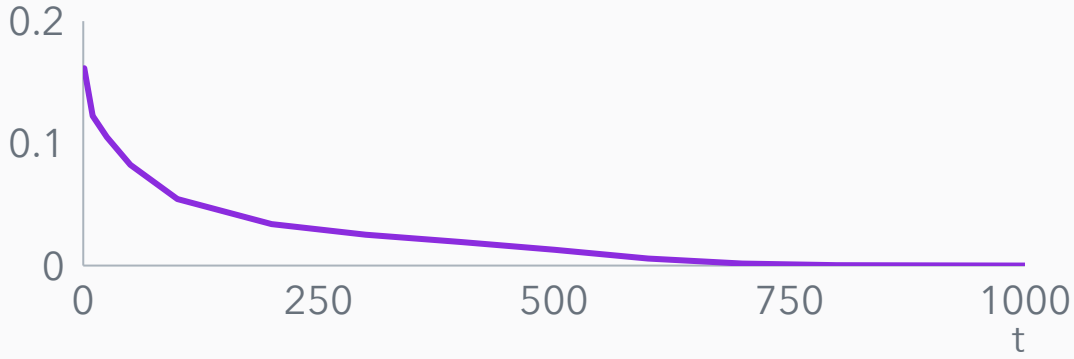
$$x_{250} = 0.38: \quad \underbrace{\text{ink},\ \varepsilon = -0.50}_{P=0.63} \quad \text{or} \quad \underbrace{\text{paper},\ \varepsilon = 1.60}_{P=0.37}$$

$$\hat{\varepsilon}^* = \mathbb{E}[\varepsilon \mid x_t] = 0.63 \times (-0.50) + 0.37 \times 1.60 = 0.28$$

one pixel: the least possible loss



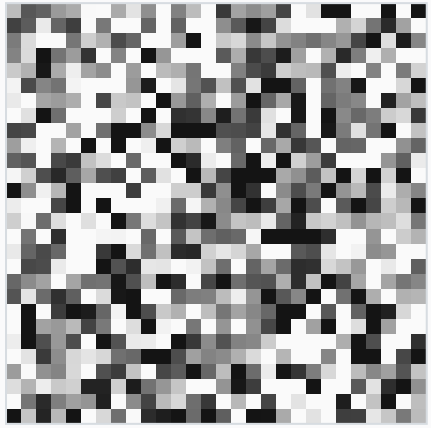
our MNIST network: its loss at each t



> So the loss cannot reach zero. Left: for our pixel the unavoidable loss is largest in the middle, where both stories are plausible. Right: our trained network's loss by noise level.

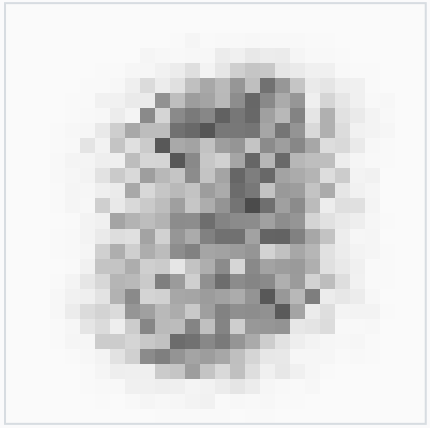
One pixel alone gives a grey; all 784 together give a 3

x_{500}



the input

each pixel alone

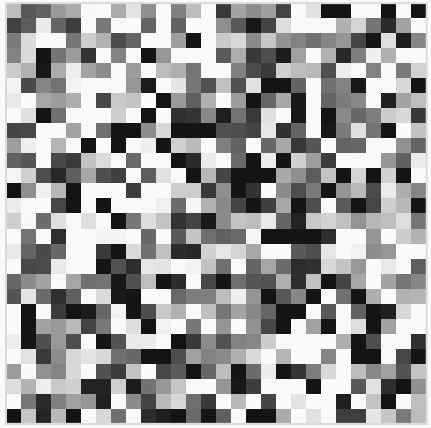


error 0.230

> At t = 500, guessing each pixel on its own from its own noisy value gives a grey cloud: average squared error 0.230.

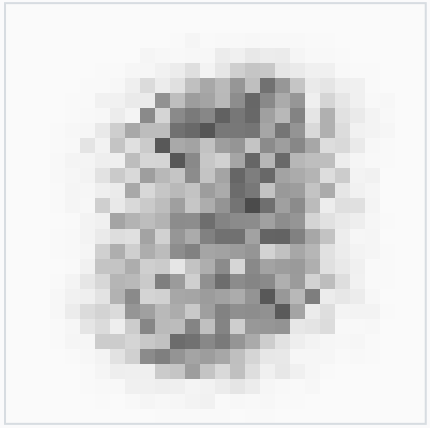
One pixel alone gives a grey; all 784 together give a 3

x_{500}



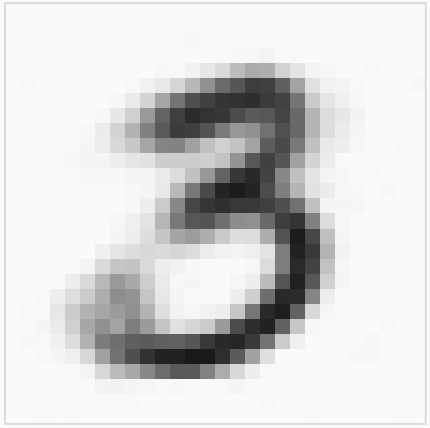
the input

each pixel alone



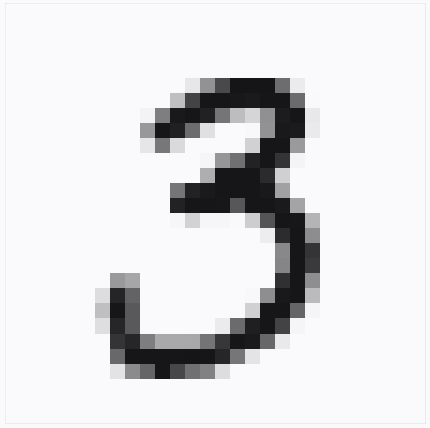
error 0.230

the network



error 0.075

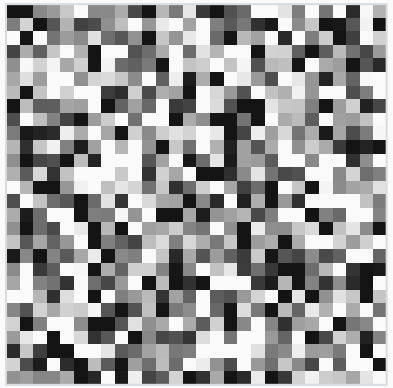
our 3



> The network looks at all of them at once and returns a clean 3: error 0.075, three times smaller. Lecture 2's lesson again: strokes are coordinated choices, and the U-Net can see them.

The same step on an image, at t = 600

x_{600}

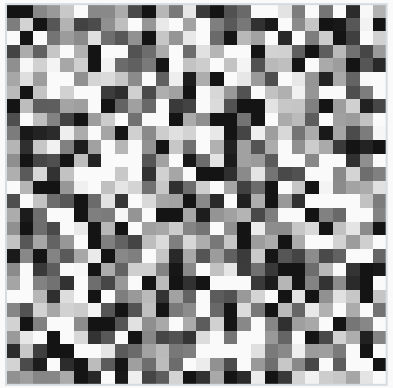


$\beta = 0.012$, $\bar{\alpha} = 0.026$, step width 0.110

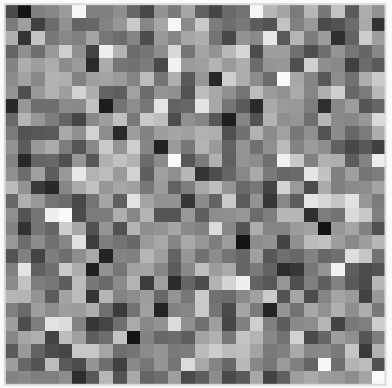
> A sampling run of our model at t = 600: to us, pure static.

The same step on an image, at t = 600

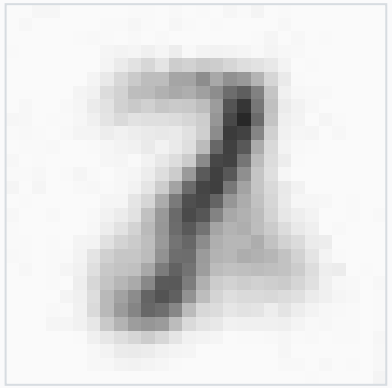
x_{600}



$\hat{\varepsilon}$: guessed noise



$\hat{x_0}$: guessed digit

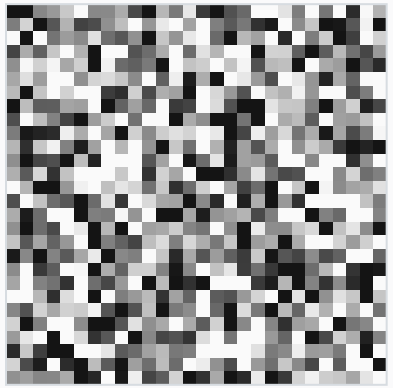


$\beta = 0.012$, $\bar{\alpha} = 0.026$, step width 0.110

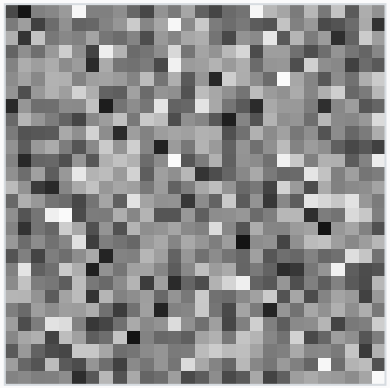
> The network's guessed noise, and the digit it implies: already a 7.

The same step on an image, at t = 600

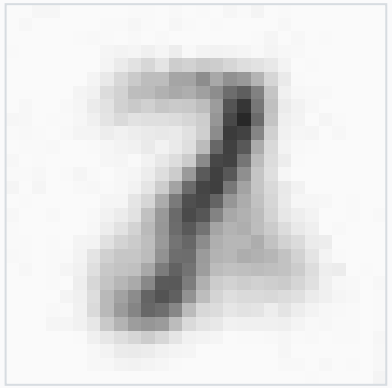
x_{600}



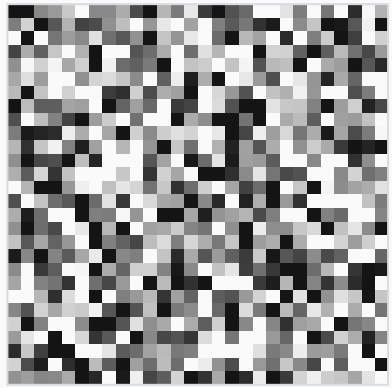
$\hat{\varepsilon}$: guessed noise



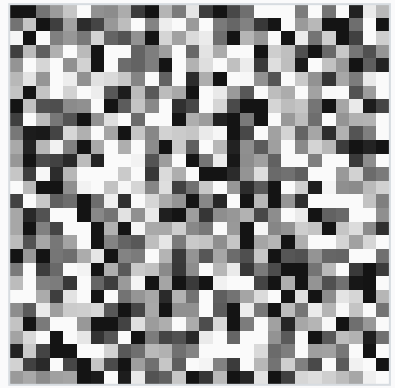
$\hat{x_0}$: guessed digit



μ: the mean



x_{599}



$\beta = 0.012$, $\bar{\alpha} = 0.026$, step width 0.110

> The mean moves the static only very slightly towards that digit, and fresh noise is added. To the eye nothing changes; over 600 more steps the 7 appears.

The noise predictor is a score estimator

$$\text{one bump, } x_0 \text{ known:} \quad \nabla_x \log \mathcal{N}\left(x; \sqrt{\bar{\alpha}_t}\, x_0, 1 - \bar{\alpha}_t\right) = -\frac{x - \sqrt{\bar{\alpha}_t}\, x_0}{1 - \bar{\alpha}_t} = -\frac{\varepsilon}{\sqrt{1 - \bar{\alpha}_t}}$$

> If we knew x_0, the noisy value would come from one Gaussian bump, and its score is the noise, scaled: $-\varepsilon/\sqrt{(1 - \bar{\alpha})}$.

Song & Ermon, "Generative Modeling by Estimating Gradients of the Data Distribution" (2019) · Calvin Luo, "Understanding Diffusion Models: A Unified Perspective" (2022) · Paras Chopra, "Diffusion Models from First Principles" (2025) (IX, corrected)

The noise predictor is a score estimator

$$\text{one bump, } x_0 \text{ known:} \quad \nabla_x \log \mathcal{N}\left(x; \sqrt{\bar{\alpha}_t}\, x_0, 1 - \bar{\alpha}_t\right) = -\frac{x - \sqrt{\bar{\alpha}_t}\, x_0}{1 - \bar{\alpha}_t} = -\frac{\varepsilon}{\sqrt{1 - \bar{\alpha}_t}}$$

$$\text{all bumps:} \quad \nabla_x \log q_t(x) = \mathbb{E}\left[-\frac{\varepsilon}{\sqrt{1 - \bar{\alpha}_t}} \,\middle|\, x \right] = -\frac{\mathbb{E}[\varepsilon \mid x]}{\sqrt{1 - \bar{\alpha}_t}}$$

> The real noisy density mixes one bump per possible x_0. Its score is the average of the bumps' scores, weighted by how likely each x_0 is given x: the average noise (Tweedie's formula).

Song & Ermon, "Generative Modeling by Estimating Gradients of the Data Distribution" (2019) · Calvin Luo, "Understanding Diffusion Models: A Unified Perspective" (2022) · Paras Chopra, "Diffusion Models from First Principles" (2025) (IX, corrected)

The noise predictor is a score estimator

$$\text{one bump, } x_0 \text{ known:} \quad \nabla_x \log \mathcal{N}\left(x; \sqrt{\bar{\alpha}_t}\, x_0, 1-\bar{\alpha}_t\right) = -\frac{x - \sqrt{\bar{\alpha}_t}\, x_0}{1-\bar{\alpha}_t} = -\frac{\varepsilon}{\sqrt{1-\bar{\alpha}_t}}$$

$$\text{all bumps:} \quad \nabla_x \log q_t(x) = \mathbb{E}\left[-\frac{\varepsilon}{\sqrt{1-\bar{\alpha}_t}} \,\middle|\, x \right] = -\frac{\mathbb{E}[\varepsilon \mid x]}{\sqrt{1-\bar{\alpha}_t}}$$

$$\hat{\varepsilon}_\theta(x_t, t) \approx \mathbb{E}[\varepsilon \mid x_t] \quad \Longrightarrow \quad \hat{\varepsilon}_\theta(x_t, t) \approx -\sqrt{1-\bar{\alpha}_t}\, \nabla_x \log q_t(x_t)$$

> Our network learned exactly that average noise ("The same noisy value can come from different noises"). So, up to a scale and a sign, the noise predictor is the score of the noisy data: the denoiser knows the arrows.

Song & Ermon, "Generative Modeling by Estimating Gradients of the Data Distribution" (2019) · Calvin Luo, "Understanding Diffusion Models: A Unified Perspective" (2022) · Paras Chopra, "Diffusion Models from First Principles" (2025) (IX, corrected)

The noise predictor is a score estimator

$$\text{one bump, } x_0 \text{ known:} \quad \nabla_x \log \mathcal{N}\left(x; \sqrt{\bar{\alpha}_t}\, x_0, 1-\bar{\alpha}_t\right) = -\frac{x - \sqrt{\bar{\alpha}_t}\, x_0}{1-\bar{\alpha}_t} = -\frac{\varepsilon}{\sqrt{1-\bar{\alpha}_t}}$$

$$\text{all bumps:} \quad \nabla_x \log q_t(x) = \mathbb{E}\left[-\frac{\varepsilon}{\sqrt{1-\bar{\alpha}_t}} \,\middle|\, x \right] = -\frac{\mathbb{E}[\varepsilon \mid x]}{\sqrt{1-\bar{\alpha}_t}}$$

$$\hat{\varepsilon}_\theta(x_t, t) \approx \mathbb{E}[\varepsilon \mid x_t] \quad \Longrightarrow \quad \hat{\varepsilon}_\theta(x_t, t) \approx -\sqrt{1-\bar{\alpha}_t}\, \nabla_x \log q_t(x_t)$$

$$\text{pixel, } t = 250: \quad -\frac{-0.5}{0.690} = +0.72 \ \text{(if we knew it was ink)}, \qquad -\frac{0.276}{0.690} = -0.40$$

> Our pixel: the one-bump formula would say +0.72 (push up, towards ink) only if we knew it was ink. The true score is −0.40: it points from x_t towards $\sqrt{\bar{\alpha}} \cdot \hat{x}_0 = 0.19$.

Song & Ermon, "Generative Modeling by Estimating Gradients of the Data Distribution" (2019) · Calvin Luo, "Understanding Diffusion Models: A Unified Perspective" (2022) · Paras Chopra, "Diffusion Models from First Principles" (2025) (IX, corrected)

Four ways to run the same trained network

	random (fresh noise)	deterministic
discrete steps	DDPM: Part 4	DDIM: Part 6
continuous time	reverse SDE	probability-flow ODE

> One trained $\hat{\epsilon}$-network, four samplers: random or deterministic, in steps or in continuous time. Optional depth has the continuous versions.

MIT 6.S183 "A Practical Introduction to Diffusion Models" (IAP 2025), Lecture 2 (39:04) · Song et al., "Score-Based Generative Modeling through SDEs" (2021)

Four ways to run the same trained network

	random (fresh noise)	deterministic
discrete steps	DDPM: Part 4	DDIM: Part 6
continuous time	reverse SDE	probability-flow ODE

In two sentences: diffusion is a VAE whose encoder is fixed noising; its ELBO is a weighted sum of noise MSEs, and we train the unweighted one. The noise predictor is the score of the noisy data, so sampling climbs the arrows while the noise level falls.

> The deterministic samplers are the faster ones: Part 6.

MIT 6.S183 "A Practical Introduction to Diffusion Models" (IAP 2025), Lecture 2 (39:04) · Song et al., "Score-Based Generative Modeling through SDEs" (2021)

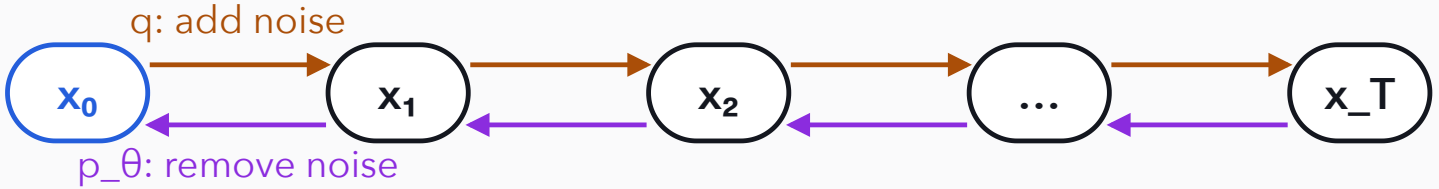
Diffusion is a VAE with a fixed encoder

VAE (Lecture 2)



prior p(z) = N(0, I)

Diffusion



p(x_T) = N(0, I)

> Lecture 2's VAE above, today's chain below. Does diffusion fit the VAE picture?

Ari Seff, "What are Diffusion Models?" (2022) (6:28) · Calvin Luo, "Understanding Diffusion Models: A Unified Perspective" (2022)

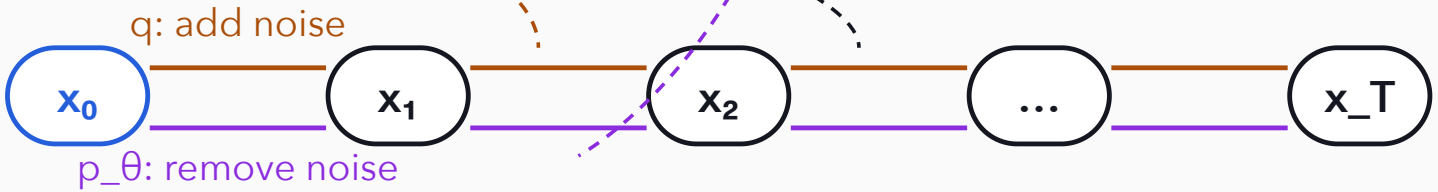
Diffusion is a VAE with a fixed encoder

VAE (Lecture 2)



prior p(z) = N(0, I)

Diffusion



p(x_T) = N(0, I)

encoder: fixed, no φ

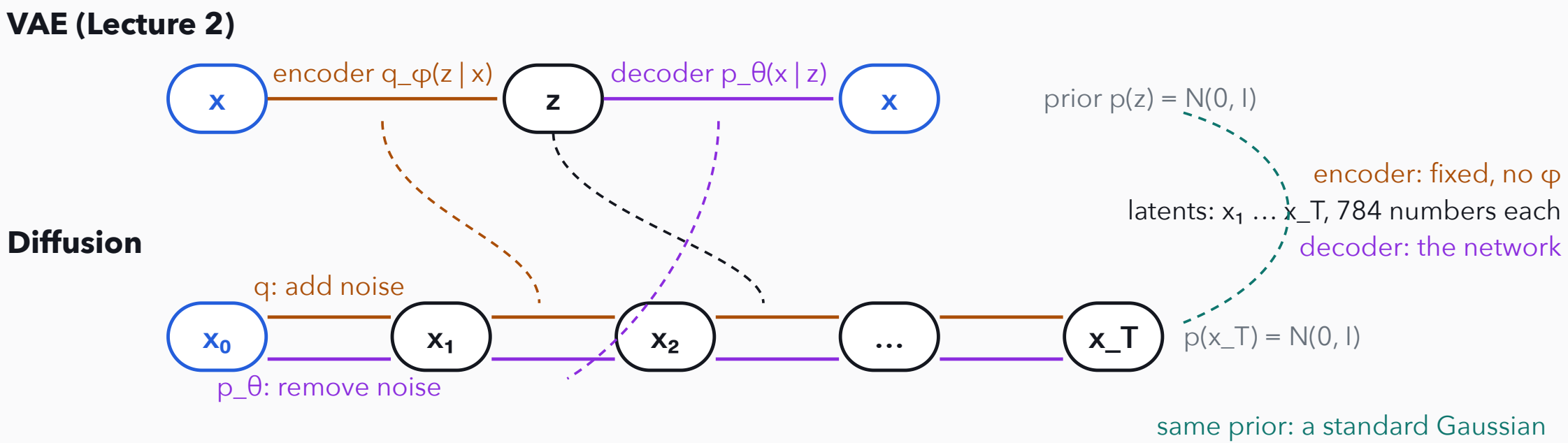
latents: x_1 … x_T, 784 numbers each

decoder: the network

> Yes: the noisy images x_1 … x_T are the latents, the forward noising is the encoder (fixed, nothing to learn) and the reverse chain is the decoder.

Ari Seff, "What are Diffusion Models?" (2022) (6:28) · Calvin Luo, "Understanding Diffusion Models: A Unified Perspective" (2022)

Diffusion is a VAE with a fixed encoder



> Both start sampling from the same prior: a standard Gaussian. So we can train diffusion the way we trained the VAE: maximise the ELBO.

Ari Seff, "What are Diffusion Models?" (2022) (6:28) · Calvin Luo, "Understanding Diffusion Models: A Unified Perspective" (2022)

Its ELBO is a weighted noise error

$$-\mathrm{ELBO} = \sum_t w_t \, \mathbb{E}\left\| \varepsilon - \hat{\varepsilon}_\theta(x_t, t) \right\|^2 + \mathrm{const}$$

> Write the ELBO of this VAE as in Lectures 1 and 3: each step contributes a KL divergence between two Gaussians, and that KL is a weighted squared error between the true noise and the guess.

Ho, Jain & Abbeel, "Denoising Diffusion Probabilistic Models" (2020) · §3.4

Its ELBO is a weighted noise error

$$-\mathrm{ELBO} = \sum_t w_t \, \mathbb{E}\left\|\varepsilon - \hat{\varepsilon}_\theta(x_t, t)\right\|^2 + \mathrm{const}$$

$$L_{\mathrm{simple}} = \mathbb{E}\left\|\varepsilon - \hat{\varepsilon}_\theta(x_t, t)\right\|^2 \qquad (\text{every } w_t = 1)$$

Part 3's loss is this ELBO with its weights changed; Ho et al. found that equal weights give better samples. The derivation, one rule per line, is in optional depth.

> Set every weight to 1 and you get exactly the loss we trained in Part 3. So training is, up to this reweighting, maximum likelihood.

Ho, Jain & Abbeel, "Denoising Diffusion Probabilistic Models" (2020) · §3.4

The same bound as Lecture 1

$$\text{Lecture 1:} \quad \log p(\mathcal{D}) \;\geq\; \mathbb{E}_q\big[\log p(\mathcal{D}, \theta) - \log q(\theta)\big] = \text{ELBO}$$

> Lecture 1: the log evidence is at least the ELBO, an average under q of log joint minus log q.

The same bound as Lecture 1

$$\text{Lecture 1:} \quad \log p(\mathcal{D}) \;\geq\; \mathbb{E}_q\big[\log p(\mathcal{D}, \theta) - \log q(\theta)\big] = \text{ELBO}$$

$$\text{now:} \quad \log p_\theta(x_0) \;\geq\; \mathbb{E}_q\big[\log p_\theta(x_{0:T}) - \log q(x_{1:T} \mid x_0)\big]$$

> Here the data is x_0, the unknowns are x_1 … x_T and q is the forward chain: the same bound, letter for letter.

The same bound as Lecture 1

$$\text{Lecture 1:}\quad \log p(\mathcal{D}) \;\geq\; \mathbb{E}_q\big[\log p(\mathcal{D},\theta) - \log q(\theta)\big] = \text{ELBO}$$

$$\text{now:}\quad \log p_\theta(x_0) \;\geq\; \mathbb{E}_q\big[\log p_\theta(x_{0:T}) - \log q(x_{1:T}\mid x_0)\big]$$

$$p_\theta(x_{0:T}) = p(x_T)\prod_{t=1}^{T} p_\theta(x_{t-1}\mid x_t), \qquad q(x_{1:T}\mid x_0) = \prod_{t=1}^{T} q(x_t\mid x_{t-1})$$

> Both chains factorise into steps: the model's joint is the prior on x_T times a thousand learned steps back; q is a thousand fixed steps forward.

Ho, Jain & Abbeel, "Denoising Diffusion Probabilistic Models" (2020) · Lecture 1 §4.1

Turn each forward step around, given x_0

$$\mathrm{ELBO} = \mathbb{E}_q\left[\log p(x_T) + \sum_{t=1}^{T} \log \frac{p_\theta(x_{t-1} \mid x_t)}{q(x_t \mid x_{t-1})}\right]$$

> Put the factorisations in: one log ratio per step, the network's step back over the fixed step forward.

Outlier, "Diffusion Models, Paper Explanation, Math Explained" (2022) (16:23) · Calvin Luo, "Understanding Diffusion Models: A Unified Perspective" (2022)

Turn each forward step around, given x_0

$$\mathrm{ELBO} = \mathbb{E}_q\left[\log p(x_T) + \sum_{t=1}^{T} \log \frac{p_\theta(x_{t-1} \mid x_t)}{q(x_t \mid x_{t-1})}\right]$$

$$q(x_t \mid x_{t-1}) = q(x_t \mid x_{t-1}, x_0) = \frac{q(x_{t-1} \mid x_t, x_0)\, q(x_t \mid x_0)}{q(x_{t-1} \mid x_0)} \qquad (t \geq 2)$$

> The ratio compares steps in opposite directions. Fix that with Bayes, exactly as in "Bayes with three Gaussians": the forward step equals the step back given x_0, times a ratio of shortcut densities.

Turn each forward step around, given x_0

$$\text{ELBO} = \mathbb{E}_q\left[\log p(x_T) + \sum_{t=1}^{T} \log \frac{p_\theta(x_{t-1} \mid x_t)}{q(x_t \mid x_{t-1})}\right]$$

$$q(x_t \mid x_{t-1}) = q(x_t \mid x_{t-1}, x_0) = \frac{q(x_{t-1} \mid x_t, x_0)\, q(x_t \mid x_0)}{q(x_{t-1} \mid x_0)} \qquad (t \geq 2)$$

$$\prod_{t=2}^{T} \frac{q(x_t \mid x_0)}{q(x_{t-1} \mid x_0)} = \frac{q(x_2 \mid x_0)}{q(x_1 \mid x_0)} \cdot \frac{q(x_3 \mid x_0)}{q(x_2 \mid x_0)} \cdots = \frac{q(x_T \mid x_0)}{q(x_1 \mid x_0)}$$

> Those ratios telescope: every middle term cancels, leaving only the two ends.

Three kinds of term; the middle ones are the training loss

$$-\mathrm{ELBO} = \underbrace{\mathrm{KL}\big(q(x_T \mid x_0) \,\|\, p(x_T)\big)}_{L_T} + \sum_{t=2}^{T} \underbrace{\mathbb{E}_q \,\mathrm{KL}\big(q(x_{t-1} \mid x_t, x_0) \,\|\, p_\theta(x_{t-1} \mid x_t)\big)}_{L_{t-1}} \underbrace{- \mathbb{E}_q \log p_\theta(x_0 \mid x_1)}_{L_0}$$

> Group the logs into KL divergences. Three kinds of term.

Ho, Jain & Abbeel, "Denoising Diffusion Probabilistic Models" (2020) · eq. 5

Three kinds of term; the middle ones are the training loss

$$-\mathrm{ELBO} = \underbrace{\mathrm{KL}\big(q(x_T \mid x_0) \,\|\, p(x_T)\big)}_{L_T} + \sum_{t=2}^{T} \underbrace{\mathbb{E}_q \,\mathrm{KL}\big(q(x_{t-1} \mid x_t, x_0) \,\|\, p_\theta(x_{t-1} \mid x_t)\big)}_{L_{t-1}} \underbrace{- \mathbb{E}_q \log p_\theta(x_0 \mid x_1)}_{L_0}$$

term	what it compares	depends on θ?
L_T	the noised image with the N(0, I) prior	no: and it is ≈ 0
L_0	the very last step back	yes, a small decoder term

> L_T has no parameters, and the shortcut makes it tiny. L_0 is the final step, a small reconstruction term.

Ho, Jain & Abbeel, "Denoising Diffusion Probabilistic Models" (2020) · eq. 5

Three kinds of term; the middle ones are the training loss

$$-\mathrm{ELBO} = \underbrace{\mathrm{KL}\big(q(x_T \mid x_0) \,\|\, p(x_T)\big)}_{L_T} + \sum_{t=2}^{T} \underbrace{\mathbb{E}_q \,\mathrm{KL}\big(q(x_{t-1} \mid x_t, x_0) \,\|\, p_\theta(x_{t-1} \mid x_t)\big)}_{L_{t-1}} \underbrace{- \mathbb{E}_q \log p_\theta(x_0 \mid x_1)}_{L_0}$$

term	what it compares	depends on θ?
L_T	the noised image with the N(0, I) prior	no: and it is ≈ 0
L_0	the very last step back	yes, a small decoder term

Each L_{t-1} compares the exact step back given x_0 (“Complete the square: the mean and the width”) with the network’s step back. For our pixel at t = 250 it is 0.0032 nats.

> The heart of it: a thousand KLs, each between the exact step back given x_0 and the network’s step back. Both are Gaussians.

Ho, Jain & Abbeel, “Denoising Diffusion Probabilistic Models” (2020) · eq. 5

Lecture 1's formula: two Gaussians of the same width

$$\mathrm{KL}\big(\mathcal{N}(\mu_1, \sigma_1^2) \,\|\, \mathcal{N}(\mu_2, \sigma_2^2)\big) = \log \frac{\sigma_2}{\sigma_1} + \frac{\sigma_1^2 + (\mu_1 - \mu_2)^2}{2\sigma_2^2} - \frac{1}{2}$$

> The Gaussian KL derived step by step in Lecture 1, section 2.7.

Lecture 1's formula: two Gaussians of the same width

$$\mathrm{KL}\big(\mathcal{N}(\mu_1, \sigma_1^2) \,\|\, \mathcal{N}(\mu_2, \sigma_2^2)\big) = \log \frac{\sigma_2}{\sigma_1} + \frac{\sigma_1^2 + (\mu_1 - \mu_2)^2}{2\sigma_2^2} - \frac{1}{2}$$

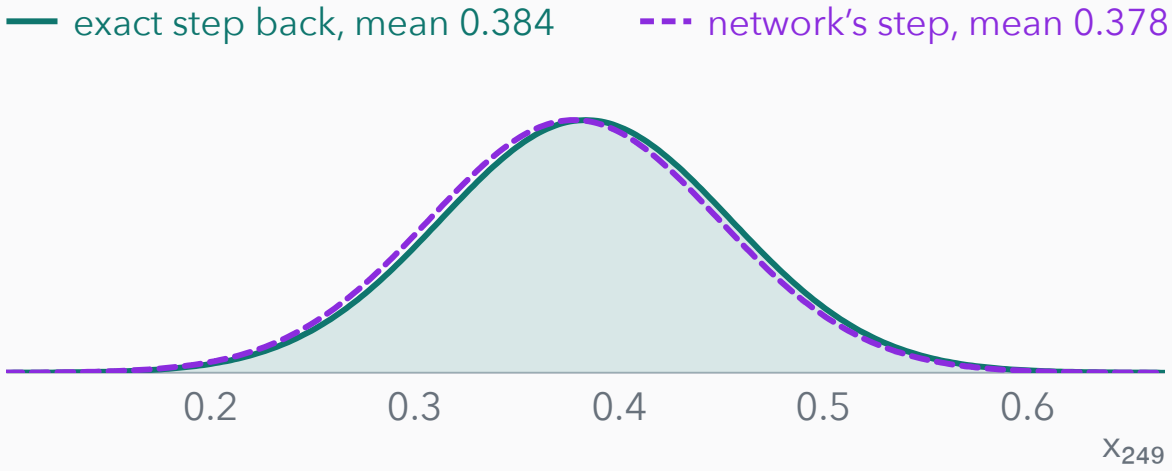
$$\sigma_1 = \sigma_2 = \sigma: \qquad \mathrm{KL} = 0 + \frac{\sigma^2 + (\mu_1 - \mu_2)^2}{2\sigma^2} - \frac{1}{2} = \frac{(\mu_1 - \mu_2)^2}{2\sigma^2}$$

> When the widths are equal the log term vanishes and the variances cancel: only the squared distance between the means is left.

Lecture 1's formula: two Gaussians of the same width

$$\mathrm{KL}\big(\mathcal{N}(\mu_1,\sigma_1^2)\,\|\,\mathcal{N}(\mu_2,\sigma_2^2)\big)=\log\frac{\sigma_2}{\sigma_1}+\frac{\sigma_1^2+(\mu_1-\mu_2)^2}{2\sigma_2^2}-\frac{1}{2}$$

$$\sigma_1=\sigma_2=\sigma:\qquad \mathrm{KL}=0+\frac{\sigma^2+(\mu_1-\mu_2)^2}{2\sigma^2}-\frac{1}{2}=\frac{(\mu_1-\mu_2)^2}{2\sigma^2}$$



$$\frac{(0.3836-0.3779)^2}{2\times 0.00503}=0.0032\ \text{nats}$$

Our pixel at t = 250: the exact step back has mean 0.384, the network's (with the best guess) 0.378, both with width 0.071. KL = 0.0032 nats. Minimising it slides one bell onto the other.

Write both means with noise: the KL is a squared noise error

$$\tilde{\mu}_t = \frac{1}{\sqrt{\alpha_t}}\left(x_t - \frac{\beta_t}{\sqrt{1-\bar{\alpha}_t}}\,\varepsilon\right), \qquad \mu_\theta = \frac{1}{\sqrt{\alpha_t}}\left(x_t - \frac{\beta_t}{\sqrt{1-\bar{\alpha}_t}}\,\hat{\varepsilon}_\theta\right)$$

> From “Substitute: the mean in terms of the noise”: the exact mean uses the true noise ε, the network’s mean its guess $\hat{\varepsilon}$. Everything else is identical.

Ho, Jain & Abbeel, “Denoising Diffusion Probabilistic Models” (2020) · eq. 12

Write both means with noise: the KL is a squared noise error

$$\tilde{\mu}_t = \frac{1}{\sqrt{\alpha_t}}\left(x_t - \frac{\beta_t}{\sqrt{1-\bar{\alpha}_t}}\,\varepsilon\right), \qquad \mu_\theta = \frac{1}{\sqrt{\alpha_t}}\left(x_t - \frac{\beta_t}{\sqrt{1-\bar{\alpha}_t}}\,\hat{\varepsilon}_\theta\right)$$

$$\tilde{\mu}_t - \mu_\theta = \frac{\beta_t}{\sqrt{\alpha_t}\sqrt{1-\bar{\alpha}_t}}\left(\hat{\varepsilon}_\theta - \varepsilon\right)$$

Subtract: x_t cancels, and what remains is a constant times the noise error.

Ho, Jain & Abbeel, "Denoising Diffusion Probabilistic Models" (2020) · eq. 12

Write both means with noise: the KL is a squared noise error

$$\tilde{\mu}_t = \frac{1}{\sqrt{\alpha_t}}\left(x_t - \frac{\beta_t}{\sqrt{1-\bar{\alpha}_t}}\,\varepsilon\right), \qquad \mu_\theta = \frac{1}{\sqrt{\alpha_t}}\left(x_t - \frac{\beta_t}{\sqrt{1-\bar{\alpha}_t}}\,\hat{\varepsilon}_\theta\right)$$

$$\tilde{\mu}_t - \mu_\theta = \frac{\beta_t}{\sqrt{\alpha_t}\sqrt{1-\bar{\alpha}_t}}\left(\hat{\varepsilon}_\theta - \varepsilon\right)$$

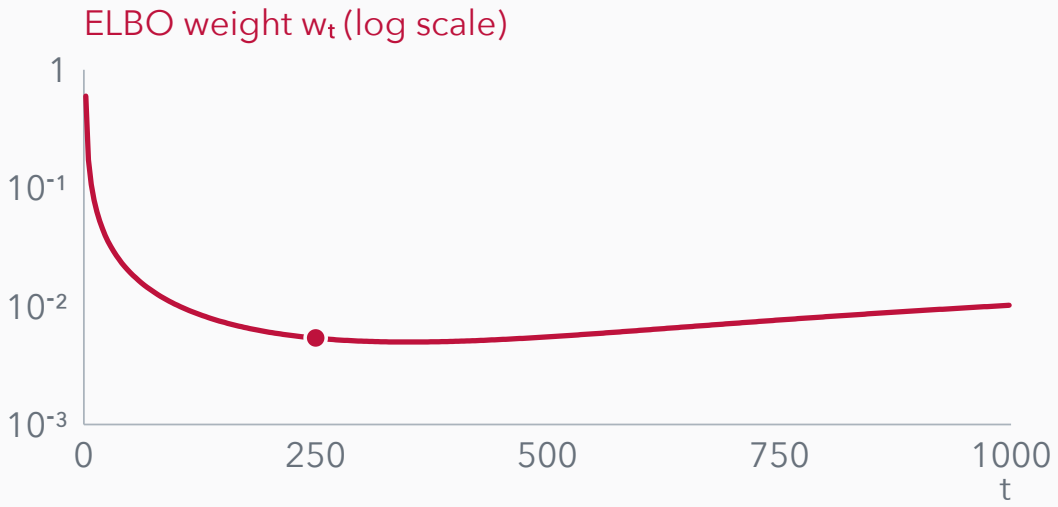
$$L_{t-1} = \frac{(\tilde{\mu}_t - \mu_\theta)^2}{2\tilde{\beta}_t} = \underbrace{\frac{\beta_t^2}{2\tilde{\beta}_t\,\alpha_t(1-\bar{\alpha}_t)}}_{w_t}\left\|\varepsilon - \hat{\varepsilon}_\theta\right\|^2$$

$$w_{250}\,(-0.5 - 0.28)^2 = 0.0054 \times 0.603 = 0.0032$$

> Square and divide by 2β̃: each KL term is a weight w_t times the squared noise error. Our pixel: 0.0054 × 0.603 = 0.0032, the same as the previous slide.

Ho, Jain & Abbeel, "Denoising Diffusion Probabilistic Models" (2020) · eq. 12

Drop the weights: the simple loss we trained in Part 3

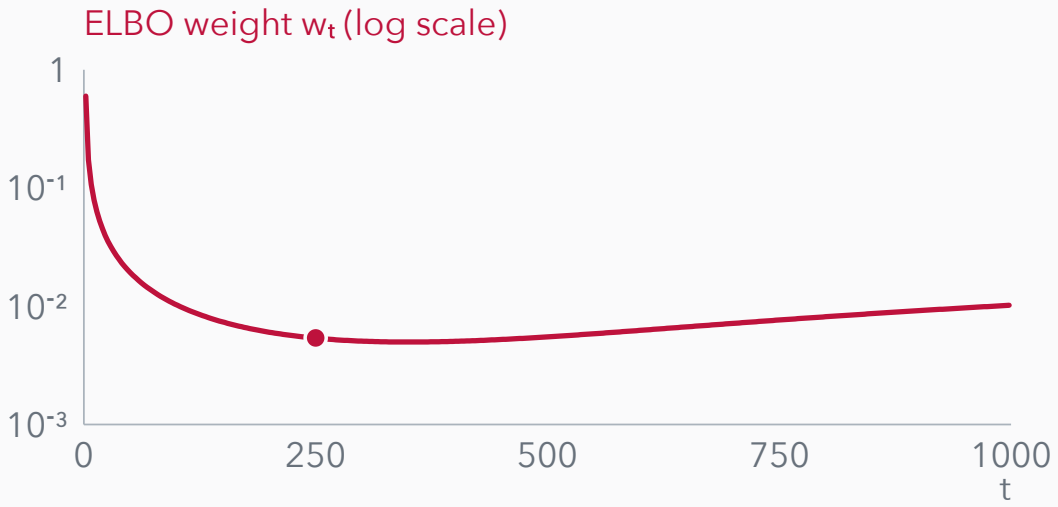


$$L_{\text{simple}} = \mathbb{E}_{t,\, x_0,\, \varepsilon} \left\| \varepsilon - \hat{\varepsilon}_\theta(x_t, t) \right\|^2$$

> The ELBO weights w_t are huge at small t and tiny at large t: the true bound mostly cares about the last, almost-clean steps.

Ho, Jain & Abbeel, "Denoising Diffusion Probabilistic Models" (2020) · §3.4

Drop the weights: the simple loss we trained in Part 3



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

Ho et al. (2020) found that equal weights give **better samples** than the true ELBO weights: they move effort from the easy small-noise steps to the harder large-noise ones.

So Part 3's loss is the ELBO with its weights changed: training is, up to this reweighting, maximum likelihood.

> DDPM drops them: every t counts equally. That is exactly the loss we trained in Part 3, now with a reason.

Ho, Jain & Abbeel, "Denoising Diffusion Probabilistic Models" (2020) · §3.4

The best possible denoiser only copies its training set

$$\hat{x}_0^*(x_t) = \sum_{i=1}^{N} w_i \, x^{(i)}, \qquad w_i \propto \exp\!\left(- \frac{\| x_t - \sqrt{\bar{\alpha}_t}\, x^{(i)} \|^2}{2(1-\bar{\alpha}_t)} \right)$$

> Our pixel formula with N possible clean images instead of two: for a finite training set the loss-minimising guess is a weighted average of training images. It drives the loss to its minimum.

The best possible denoiser only copies its training set

$$\hat{x}_0^*(x_t) = \sum_{i=1}^{N} w_i \, x^{(i)}, \qquad w_i \propto \exp\left(- \frac{\| x_t - \sqrt{\bar{\alpha}_t}\, x^{(i)} \|^2}{2(1-\bar{\alpha}_t)} \right)$$

sampled with the perfect denoiser of 1,000 training images



their nearest training images



Distance to the nearest training image: **0.01, 0.00, 0.00, 0.00, 0.01, 0.00, 0.00, 0.00**. Copies.

> Sample with it: every sample is exactly a training image. A perfect loss is a lookup table.

The best possible denoiser only copies its training set

$$\hat{x}_0^*(x_t) = \sum_{i=1}^{N} w_i \, x^{(i)}, \qquad w_i \propto \exp\Big(- \frac{\| x_t - \sqrt{\bar{\alpha}_t}\, x^{(i)} \|^2}{2(1-\bar{\alpha}_t)} \Big)$$

sampled with our network



their nearest of 60,000 training images



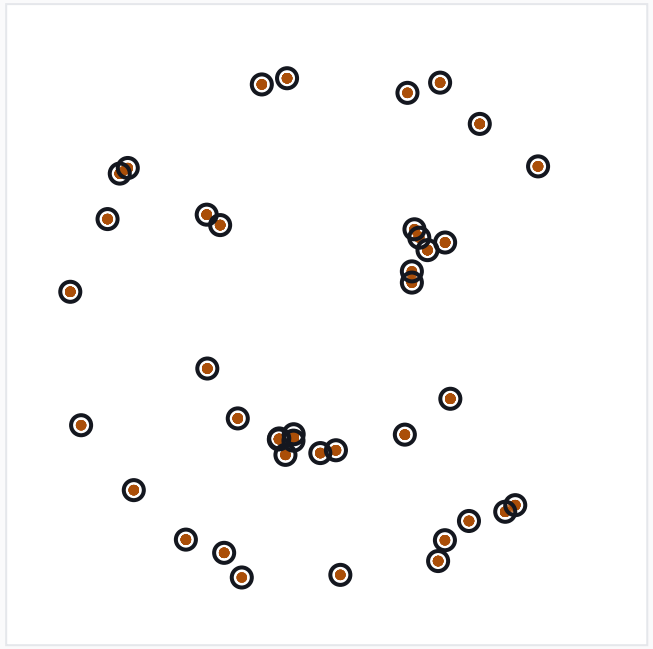
Distances **4.0–13.1**; a real test digit is 8.2 from its nearest training digit. New digits.

> Our network samples digits that are about as far from the training set as fresh test digits are. It generalises, because it cannot represent the lookup table and is smooth instead.

▶ Notebook

In 2-D: the perfect denoiser copies, a network fills in

the best possible denoiser



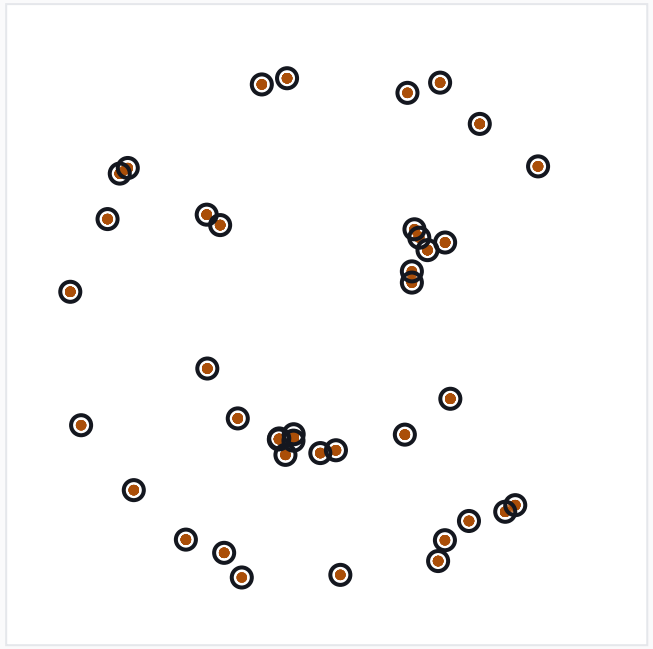
400 samples, mean distance to a training point 0.0006

> Forty training points on the smiley (circles). The perfect denoiser's 400 samples land exactly on them: mean distance 0.0006.

▶ Notebook

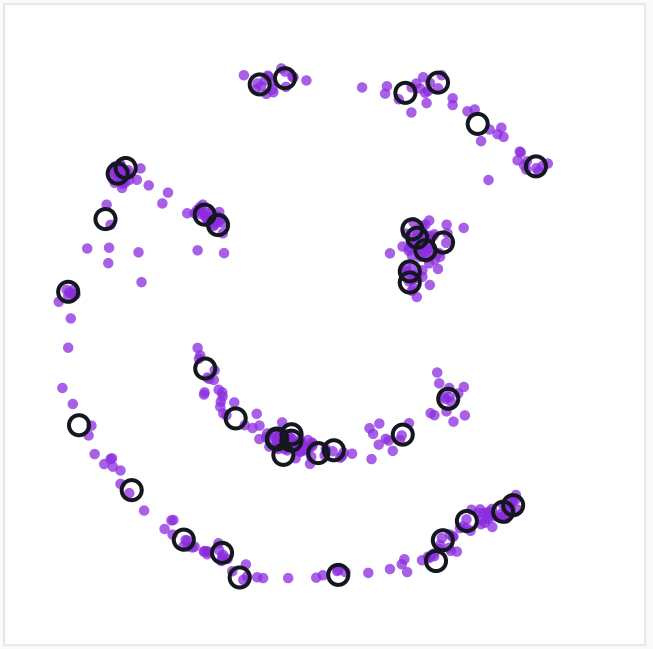
In 2-D: the perfect denoiser copies, a network fills in

the best possible denoiser



400 samples, mean distance to a training point 0.0006

our network, trained on the same 40



mean distance to a training point 0.049; to the face 0.006

> A network trained on the same 40 points: its samples spread along the curves between them, 0.049 from the nearest training point but only 0.006 from the true face. It invented the face between the points.

Classifier guidance: Bayes' rule on scores

$$p(x \mid c) = \frac{p(c \mid x)\, p(x)}{p(c)}$$

Bayes' rule for images of class c: likelihood of the label times the prior over images.

Classifier guidance: Bayes' rule on scores

$$p(x \mid c) = \frac{p(c \mid x)\, p(x)}{p(c)}$$

$$\nabla_x \log p(x \mid c) = \nabla_x \log p(x) + \nabla_x \log p(c \mid x) - \underbrace{\nabla_x \log p(c)}_{=0}$$

> Take the log and the gradient in x. The evidence p(c) has no x, so it vanishes: the same trick that made the ELBO work. The conditional score is the unconditional score plus a classifier's gradient.

Classifier guidance: Bayes' rule on scores

$$p(x \mid c) = \frac{p(c \mid x)\, p(x)}{p(c)}$$

$$\nabla_x \log p(x \mid c) = \nabla_x \log p(x) + \nabla_x \log p(c \mid x) - \underbrace{\nabla_x \log p(c)}_{=0}$$

$$\text{guided score:} \quad \nabla_x \log p(x) + w\, \nabla_x \log p(c \mid x)$$

> Turn up the classifier's voice with a weight w > 1. The catch: we need a classifier that works on noisy images at every t.

Dhariwal & Nichol, "Diffusion Models Beat GANs on Image Synthesis" (2021) · MIT 6.S183 "A Practical Introduction to Diffusion Models" (IAP 2025), Lecture 3 (6:19)

Why guidance works: the classifier cancels out

$$\nabla_x \log p(c \mid x) = \nabla_x \log p(x \mid c) - \nabla_x \log p(x)$$

> Rearrange Bayes once more: the classifier's gradient is the difference between the conditional and the unconditional score.

Why guidance works: the classifier cancels out

$$\nabla_x \log p(c \mid x) = \nabla_x \log p(x \mid c) - \nabla_x \log p(x)$$

$$\nabla_x \log p(x) + w\big(\nabla_x \log p(x \mid c) - \nabla_x \log p(x)\big)$$

> Substitute it into the guided score. No classifier left: only two scores of the same diffusion model.

Why guidance works: the classifier cancels out

$$\nabla_x \log p(c \mid x) = \nabla_x \log p(x \mid c) - \nabla_x \log p(x)$$

$$\nabla_x \log p(x) + w\big(\nabla_x \log p(x \mid c) - \nabla_x \log p(x)\big)$$

$$\tilde{\varepsilon} = \hat{\varepsilon}_\theta(x_t, t, \varnothing) + w\big(\hat{\varepsilon}_\theta(x_t, t, c) - \hat{\varepsilon}_\theta(x_t, t, \varnothing)\big)$$

In noises ("The noise predictor is a score estimator": score = $-\hat{\varepsilon}/\sqrt{(1-\bar{\alpha})}$, the scale cancels): start from the unconditional guess and move w times along the difference. w = 0 ignores the label; w = 1 is the plain conditional model; w > 1 extrapolates.

Why guidance works: the classifier cancels out

$$\nabla_x \log p(c \mid x) = \nabla_x \log p(x \mid c) - \nabla_x \log p(x)$$

$$\nabla_x \log p(x) + w\big(\nabla_x \log p(x \mid c) - \nabla_x \log p(x)\big)$$

$$\tilde{\varepsilon} = \hat{\varepsilon}_\theta(x_t, t, \varnothing) + w\big(\hat{\varepsilon}_\theta(x_t, t, c) - \hat{\varepsilon}_\theta(x_t, t, \varnothing)\big)$$

```
e_uncond = model(x_t, t, NULL)        # label hidden
e_cond   = model(x_t, t, c)
eps_hat  = e_uncond + w * (e_cond - e_uncond)
```

> How many networks? One: during training the label is replaced by “no label” some of the time (10% for MNIST), so the same network gives both guesses. Two calls per step at sampling.

The continuous view: a stochastic differential equation

$$dx = -\tfrac{1}{2}\beta(t)\,x\,dt + \sqrt{\beta(t)}\,dW \qquad \text{(forward: the noising, with many tiny steps)}$$

> Where does the drift −½βx come from?

Song et al., "Score-Based Generative Modeling through SDEs" (2021)

The continuous view: a stochastic differential equation

$$dx = -\tfrac{1}{2}\beta(t)\,x\,dt + \sqrt{\beta(t)}\,dW \qquad \text{(forward: the noising, with many tiny steps)}$$

$$dx = \Big[-\tfrac{1}{2}\beta(t)\,x - \beta(t)\,\nabla_x \log q_t(x)\Big]dt + \sqrt{\beta(t)}\,d\bar{W} \qquad \text{(reverse: needs the score)}$$

> Why does the reverse need the score?

Song et al., "Score-Based Generative Modeling through SDEs" (2021)

Other targets: x_0, v and flow matching

$$\hat{\varepsilon}, \quad \hat{x}_0, \quad \hat{v} = \sqrt{\bar{\alpha}_t}\,\varepsilon - \sqrt{1-\bar{\alpha}_t}\,x_0$$

> Do these targets carry different information?

Other targets: x_0, v and flow matching

$$\hat{\varepsilon}, \quad \hat{x}_0, \quad \hat{v} = \sqrt{\bar{\alpha}_t}\,\varepsilon - \sqrt{1-\bar{\alpha}_t}\,x_0$$

$$\text{flow matching:} \quad x_t = (1-t)\,x_0 + t\,\varepsilon, \qquad \text{predict } v = \varepsilon - x_0, \quad \frac{dx}{dt} = v$$

> Why are flow-matching paths straighter?

A frozen diffusion model as a critic: score distillation

$$\nabla_\psi L_{\mathrm{SDS}} = \mathbb{E}_{t,\varepsilon}\left[w(t)\left(\hat{\varepsilon}_\theta(x_t, t, c) - \varepsilon\right)\frac{\partial x}{\partial \psi}\right], \quad x = g(\psi)$$

Render an image x from 3-D parameters ψ, noise it, ask the frozen text-to-image model for its noise guess, and push ψ so that the guess matches (DreamFusion).

> What is being trained here?