DDA4220 · Fall 2026

Lecture 17

Deep Learning and Applications

Diffusion models

Zhen Liu

CUHK-Shenzhen

Lecture outline

01 Why generate in small steps

02 Turning data into noise

03 Learning to remove noise

04 Where the loss comes from

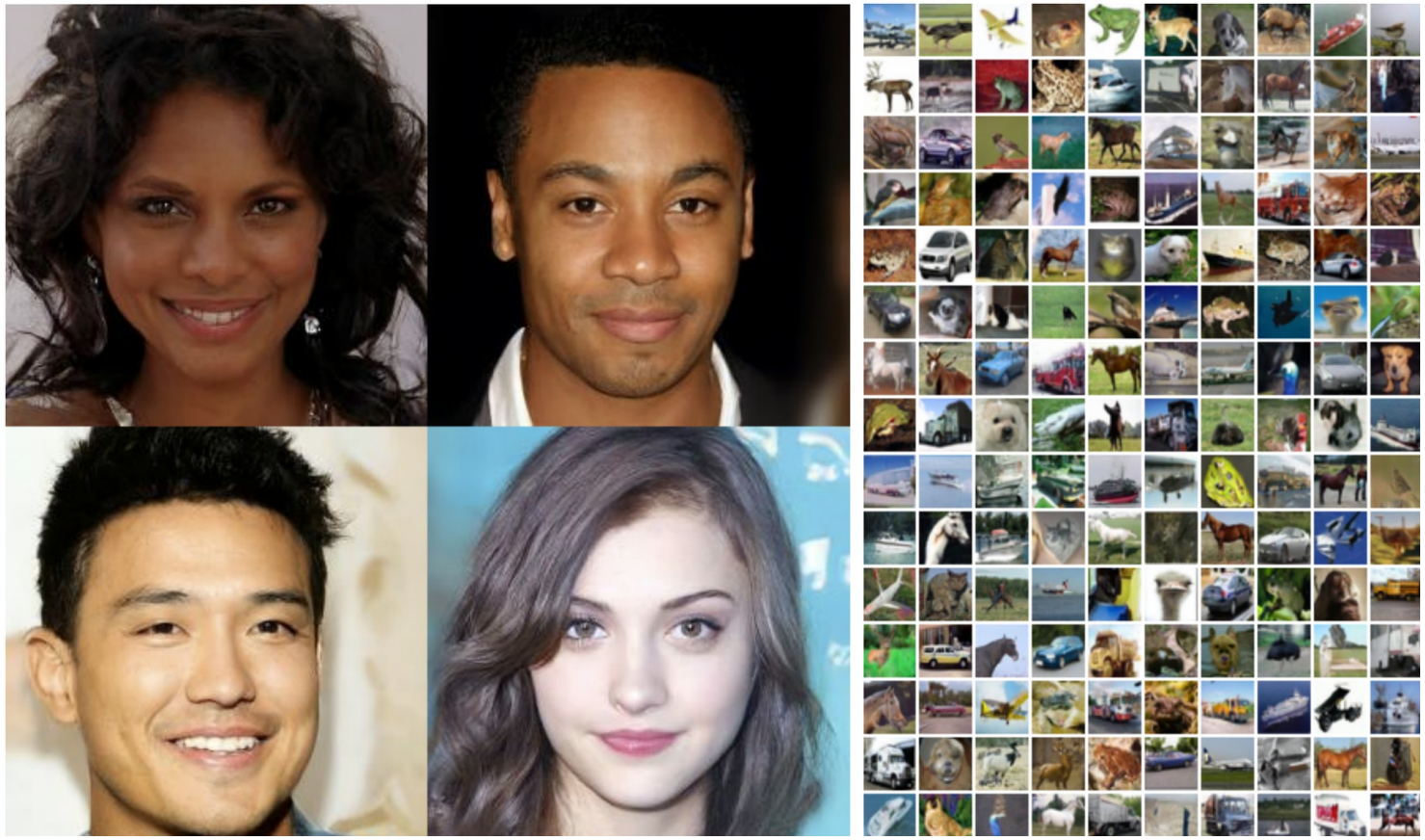
05 The denoising network

06 Scores and faster sampling

07 Results and comparison

* denotes optional slides

Image generation with diffusion



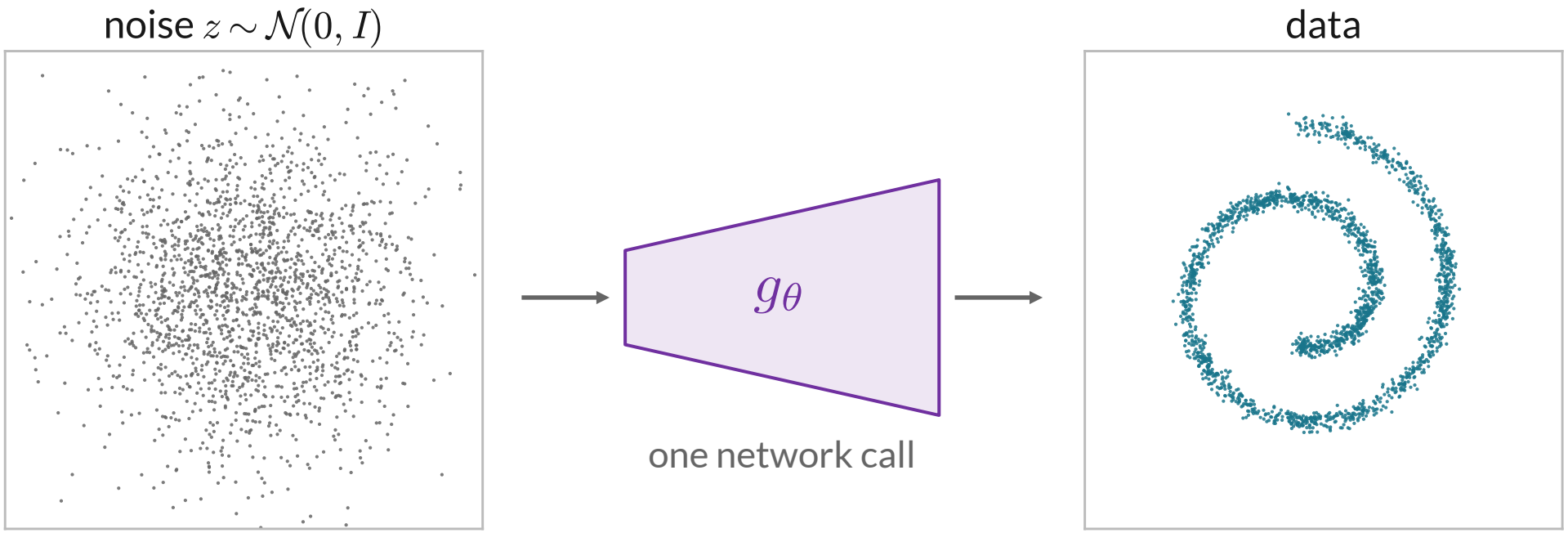
Trained only to remove noise

Generation: start from pure noise, remove a little at a time

Left: 256×256 faces. Right: 32×32 CIFAR-10 images.

Source: DDPM (2020)

Generating means mapping noise to data



VAE decoder (L15) and GAN generator (L16): one network call turns z into a sample x

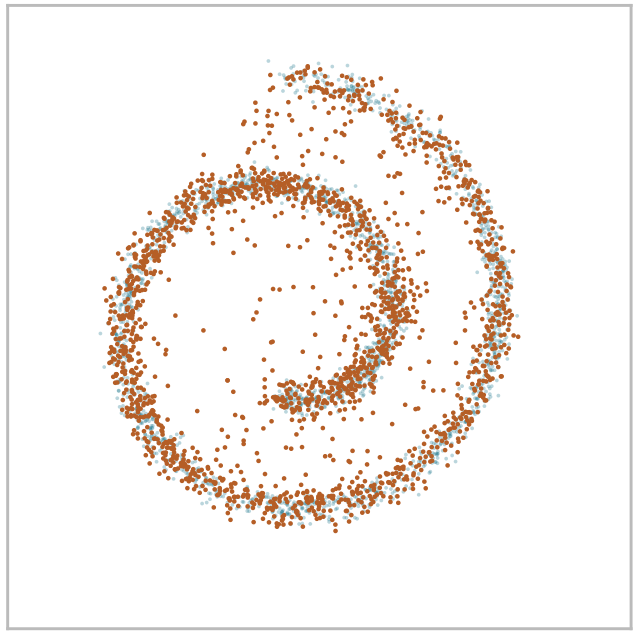
That one map must create all of the data's structure at once

How well does a single step work?

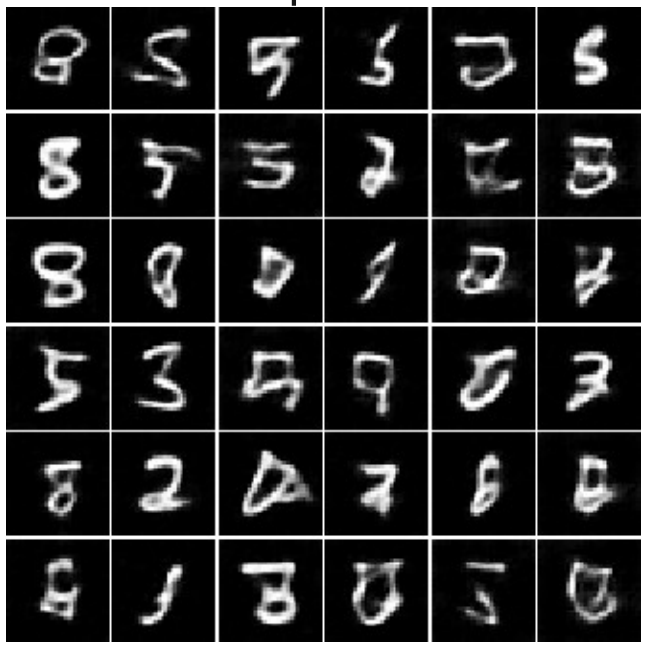
VAE: one Gaussian step blurs

$$p_\theta(x \mid z) = \mathcal{N}\big(\underbrace{\mu_\theta(z)}_{\text{one mean per code}}, \underbrace{\sigma^2 I}_{\text{fixed spread}}\big)$$

VAE samples: spiral



VAE samples: MNIST



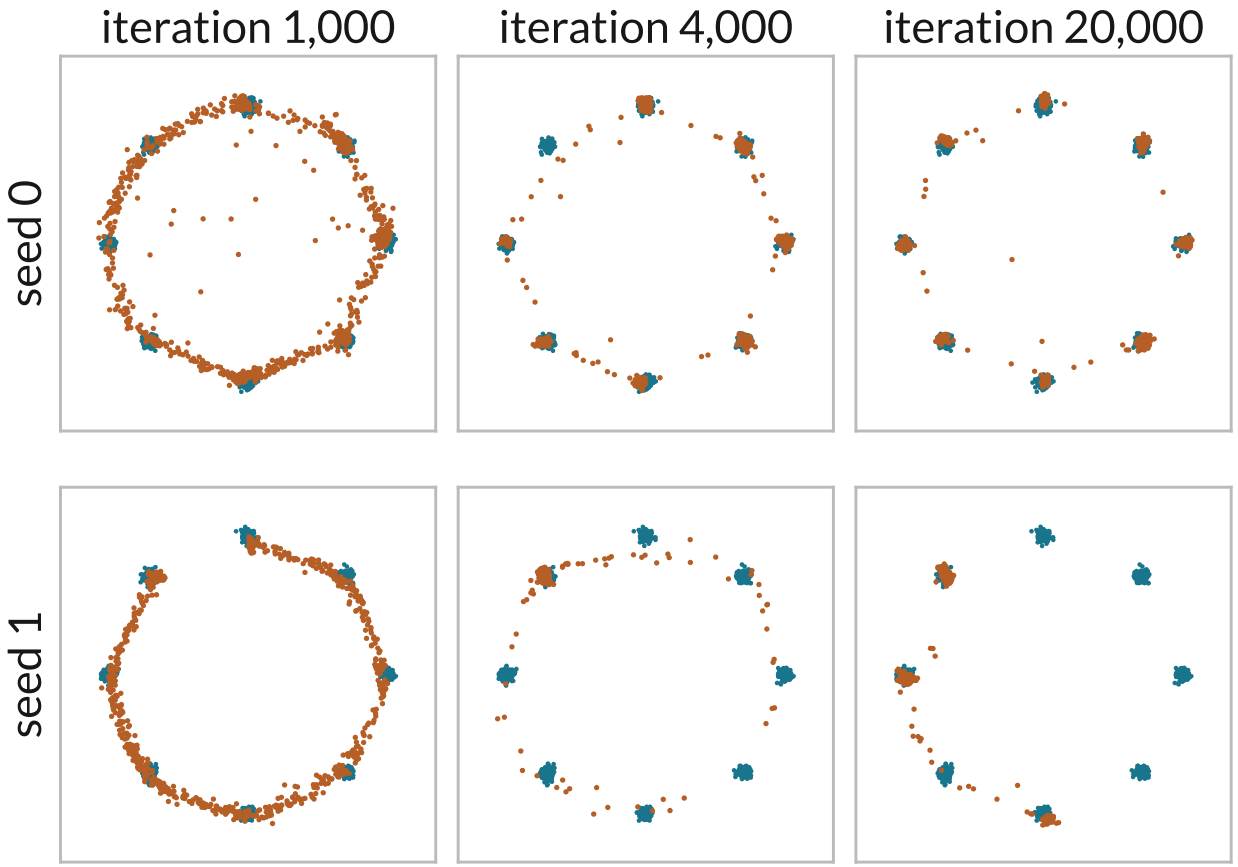
Trained with the ELBO (L15): stable

Spiral: 11% of samples fall between the arms (real data: 0.3%)

MNIST: blurry digits, broken strokes

One Gaussian around one mean is too simple for the whole map

GAN: sharp samples, unstable training



From L16: real data teal, generated orange

No Gaussian step: a discriminator judges the samples (L16)

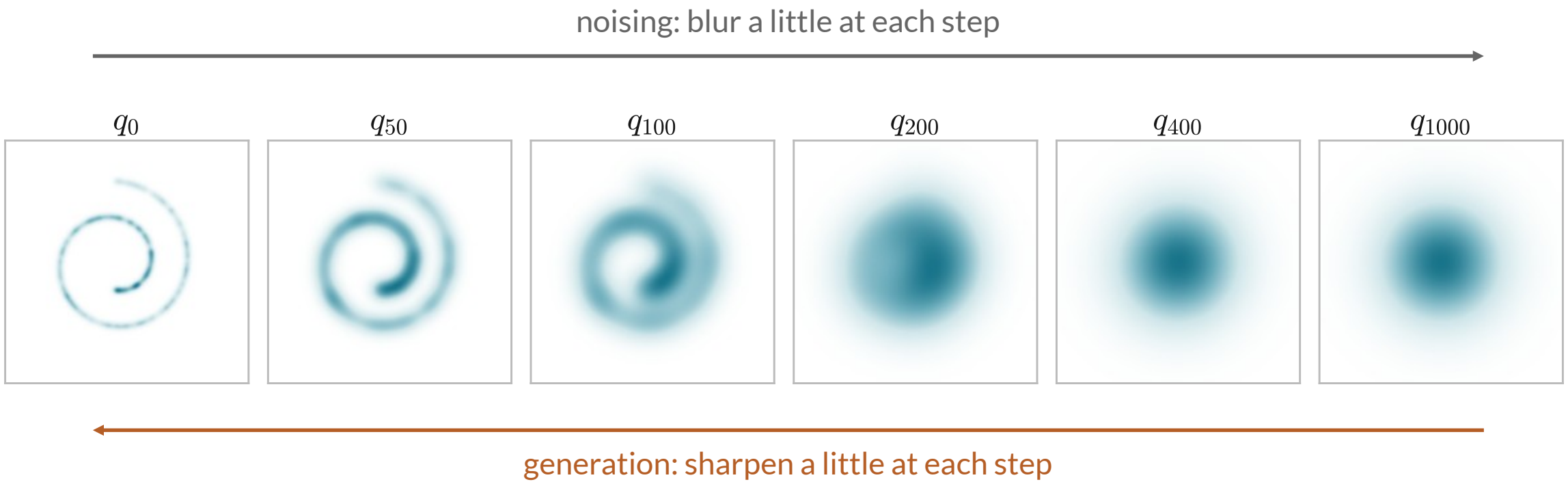
Training is a two-player game

Same settings, two seeds: seed 1 has lost modes by iteration 20,000

Wanted: the VAE's stable training without its one-step Gaussian

A path from data to noise

Many small steps instead of one: add a little Gaussian noise, many times



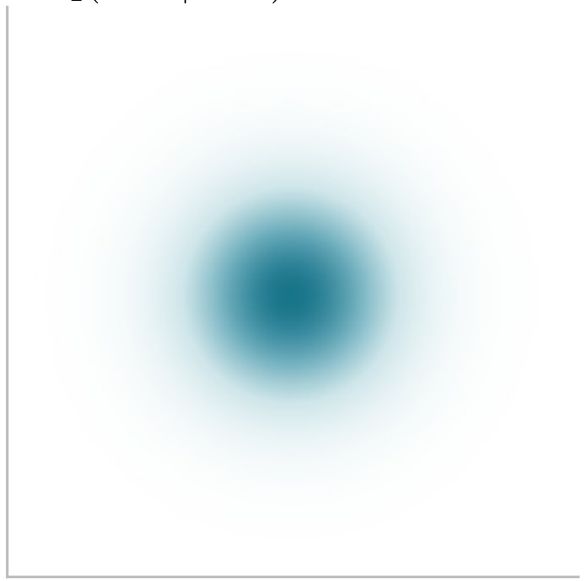
q_t: the exact density of x_t, the spiral after t noising steps

We choose this path, so noisy training examples are free to make

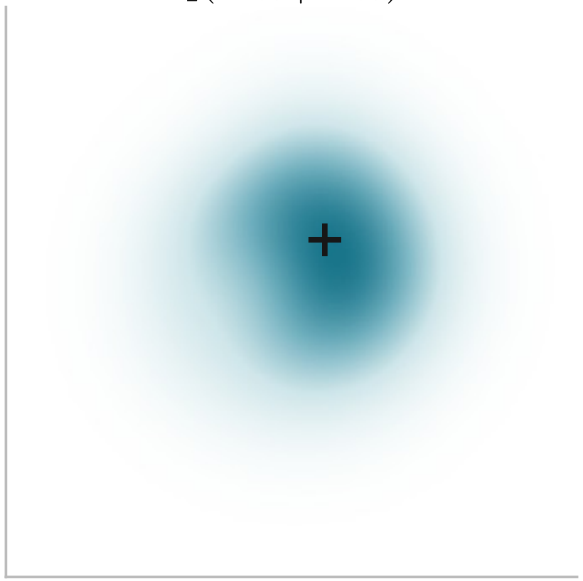
Small steps back are nearly Gaussian

Fix a noisy point x_{400} (+). Where was it earlier? (exact, 2-D spiral)

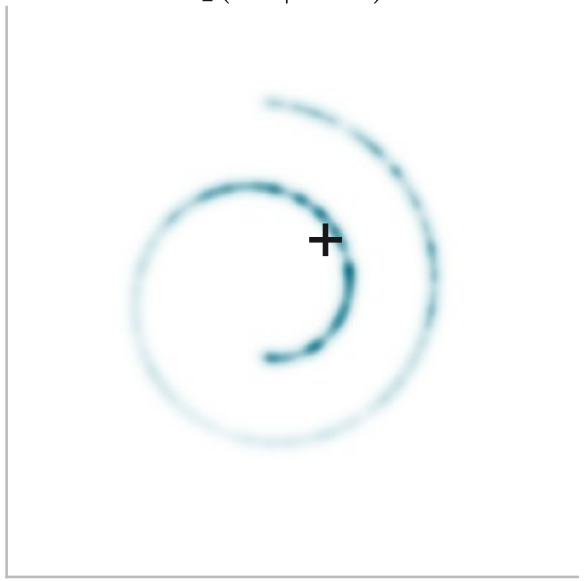
$q(x_{399} \mid x_{400})$ (zoomed 9x)



$q(x_{200} \mid x_{400})$



$q(x_0 \mid x_{400})$



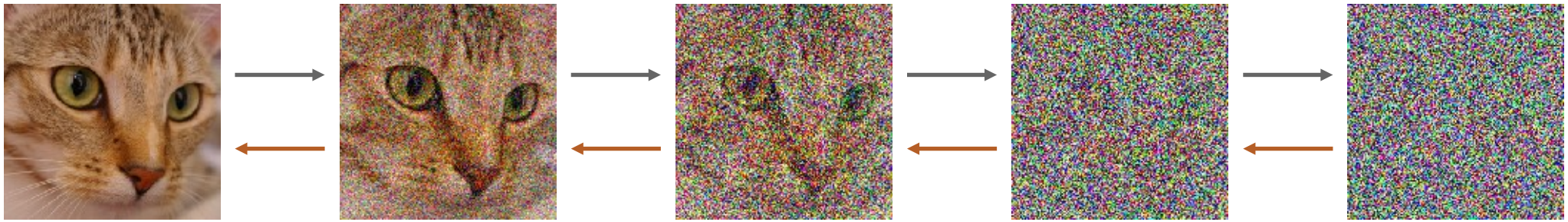
One step back: a small round blob, close to a Gaussian

All the way back: spread along the spiral, the one-step VAE's problem

A Gaussian fits one small step, not a big jump

The idea: add noise slowly, learn to undo it

Forward: fixed, adds Gaussian noise $\rightarrow$



$\leftarrow$ Reverse: learned, removes a little noise per step

Forward process: no parameters; any noise level in one line of code

Reverse process: many small Gaussian steps, one network trained by regression

Price: one network call per step

Source: photo: scikit-image (CC0)

Lecture outline

01 Why generate in small steps

02 Turning data into noise

03 Learning to remove noise

04 Where the loss comes from

05 The denoising network

06 Scores and faster sampling

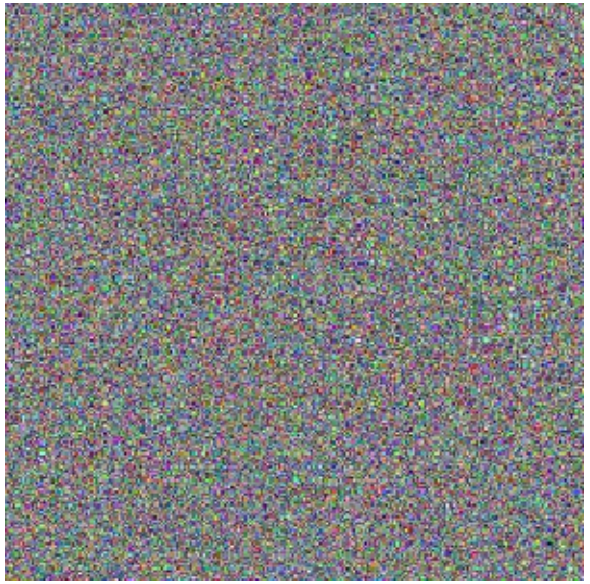
07 Results and comparison

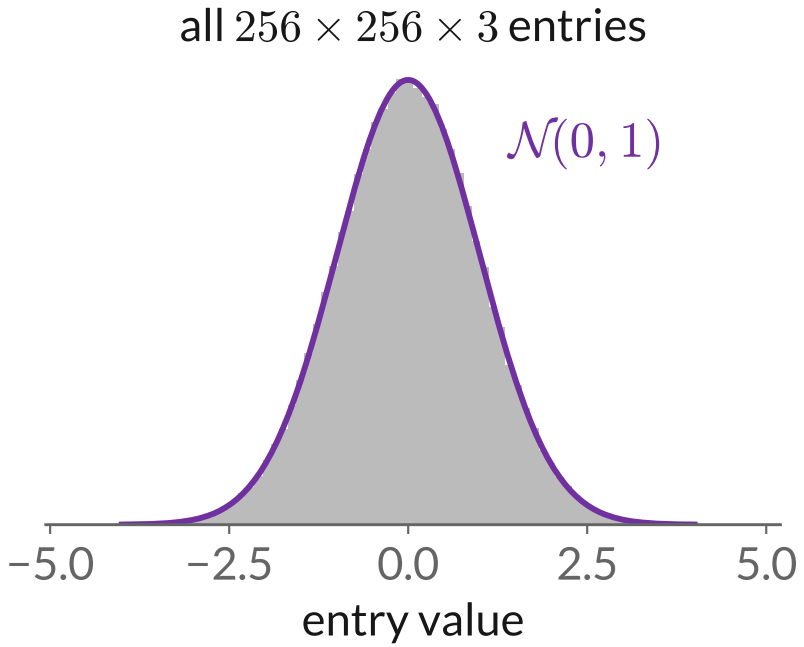
Gaussian noise on an image

image x



noise ϵ, same shape





$\epsilon \sim \mathcal{N}(0, I)$: every entry is an independent draw from a standard normal

Mean 0, variance 1, same shape as the image

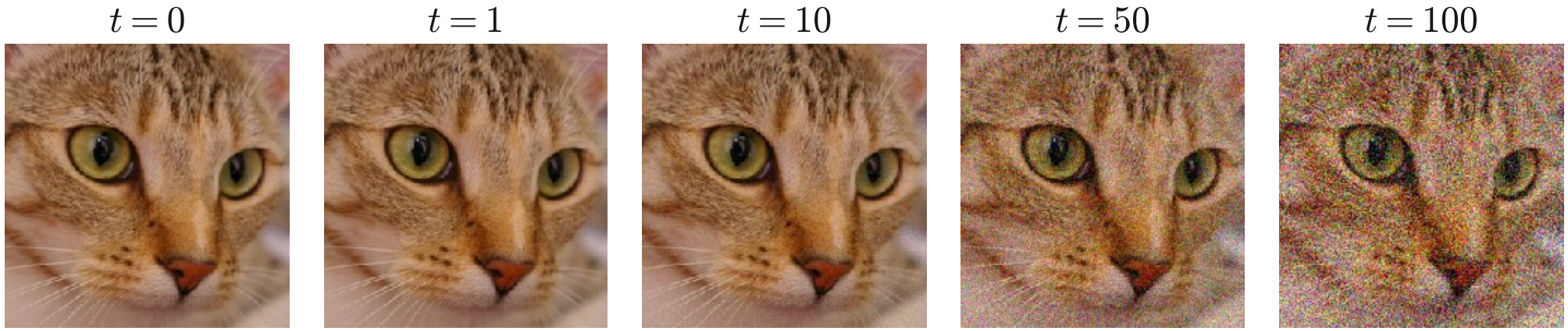
Source: photo: scikit-image (CC0)

One noising step

$$x_t = \underbrace{\sqrt{1-\beta_t}\, x_{t-1}}_{\text{shrink the previous image}} + \underbrace{\sqrt{\beta_t}\, \epsilon_t}_{\text{add fresh noise}}$$

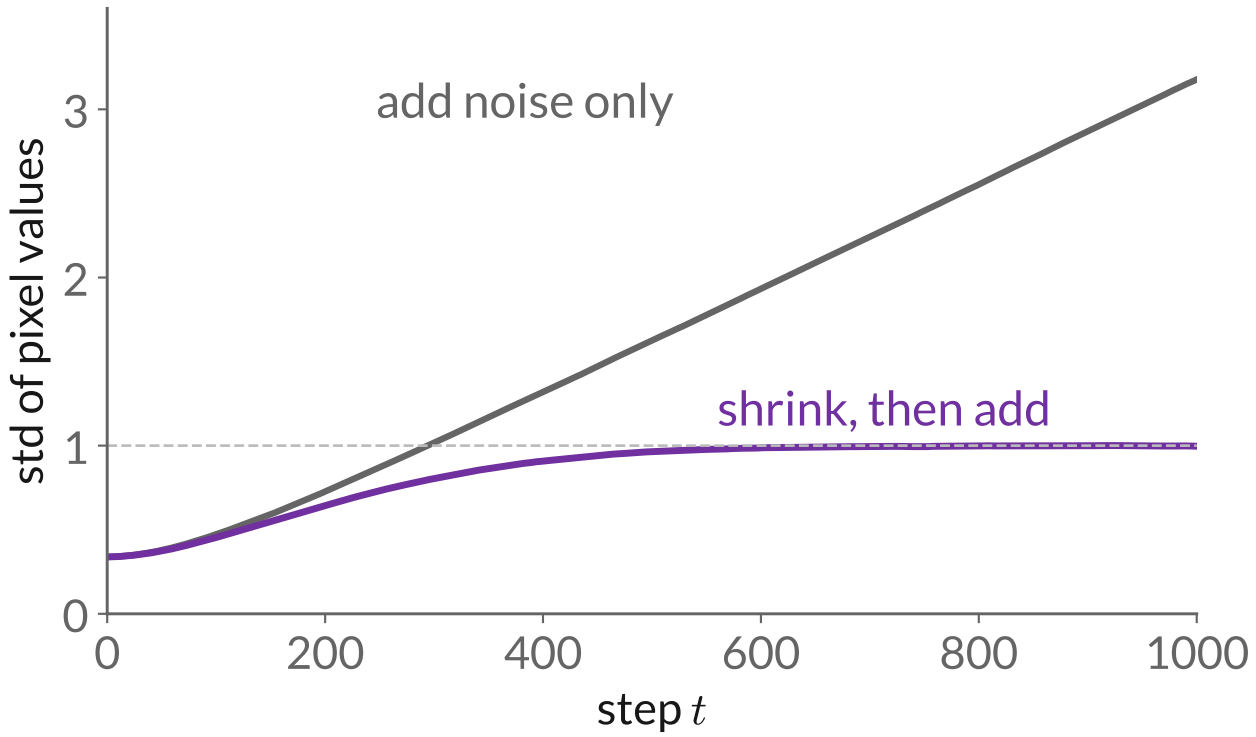
β_t : variance of the noise added at step t, small (from 10^{-4} to 0.02)

$\epsilon_t \sim \mathcal{N}(0, I)$, a new draw at every step



Source: photo: scikit-image (CC0)

Why shrink the signal?



If x_{t-1} has variance 1:

$$\mathrm{Var}(x_t) = (1 - \beta_t) \cdot 1 + \beta_t = 1$$

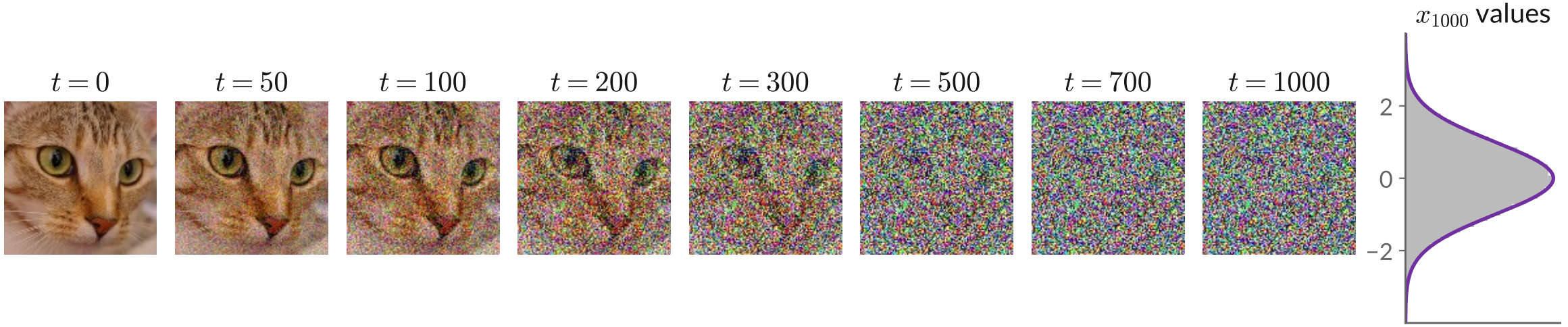
Noise only: spread grows to 3.2

Shrink, then add: spread settles at 1.0

End point $x_T \approx \mathcal{N}(0, I)$, whatever image we start from

Source: photo: scikit-image (CC0)

A thousand small steps



$T = 1000$ steps; β_t rises linearly from 10^{-4} to 0.02 (the DDPM choice)

Right: the values of x_{1000} follow $\mathcal{N}(0, 1)$; nothing of the photo remains

Source: photo: scikit-image (CC0)

Two steps combine into one

Write $\alpha_t = 1 - \beta_t$ and substitute the previous step:

$$
\begin{aligned}
x_t &= \sqrt{\alpha_t}\, x_{t-1} + \sqrt{1-\alpha_t}\, \epsilon_t \\
&= \sqrt{\alpha_t}\left(\sqrt{\alpha_{t-1}}\, x_{t-2} + \sqrt{1-\alpha_{t-1}}\, \epsilon_{t-1}\right) + \sqrt{1-\alpha_t}\, \epsilon_t \\
&= \sqrt{\alpha_t \alpha_{t-1}}\, x_{t-2} + \underbrace{\sqrt{\alpha_t(1-\alpha_{t-1})}\, \epsilon_{t-1} + \sqrt{1-\alpha_t}\, \epsilon_t}_{\text{two independent Gaussians}}
\end{aligned}
$$

Variances of independent Gaussians add:

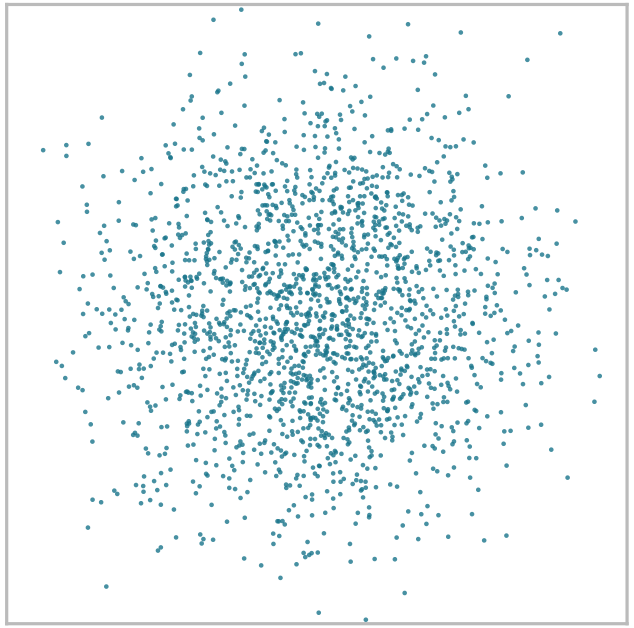
$$\alpha_t(1-\alpha_{t-1}) + (1-\alpha_t) = 1 - \alpha_t\alpha_{t-1}$$

$$\Rightarrow \quad x_t = \sqrt{\alpha_t\alpha_{t-1}}\, x_{t-2} + \sqrt{1-\alpha_t\alpha_{t-1}}\, \epsilon, \qquad \epsilon \sim \mathcal{N}(0, I)$$

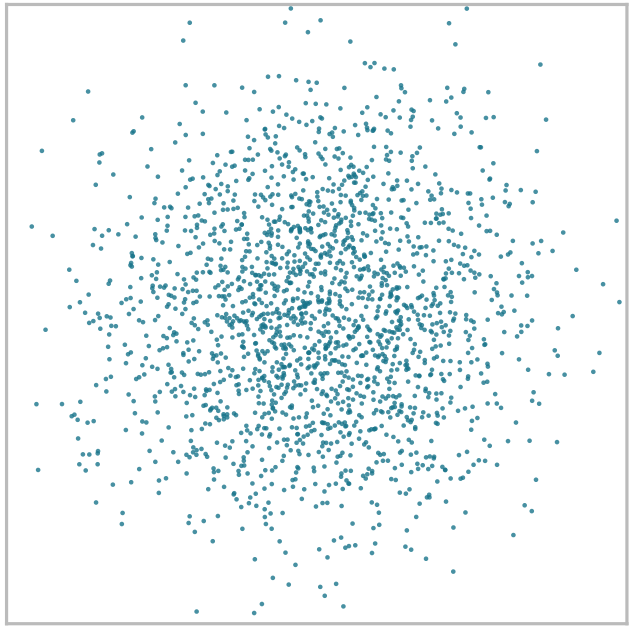
Jump to any noise level

$$x_t = \underbrace{\sqrt{\bar{\alpha}_t}\, x_0}_{\text{signal}} + \underbrace{\sqrt{1-\bar{\alpha}_t}\, \epsilon}_{\text{noise}}, \qquad \bar{\alpha}_t = \alpha_1 \alpha_2 \cdots \alpha_t$$

300 single steps



one jump



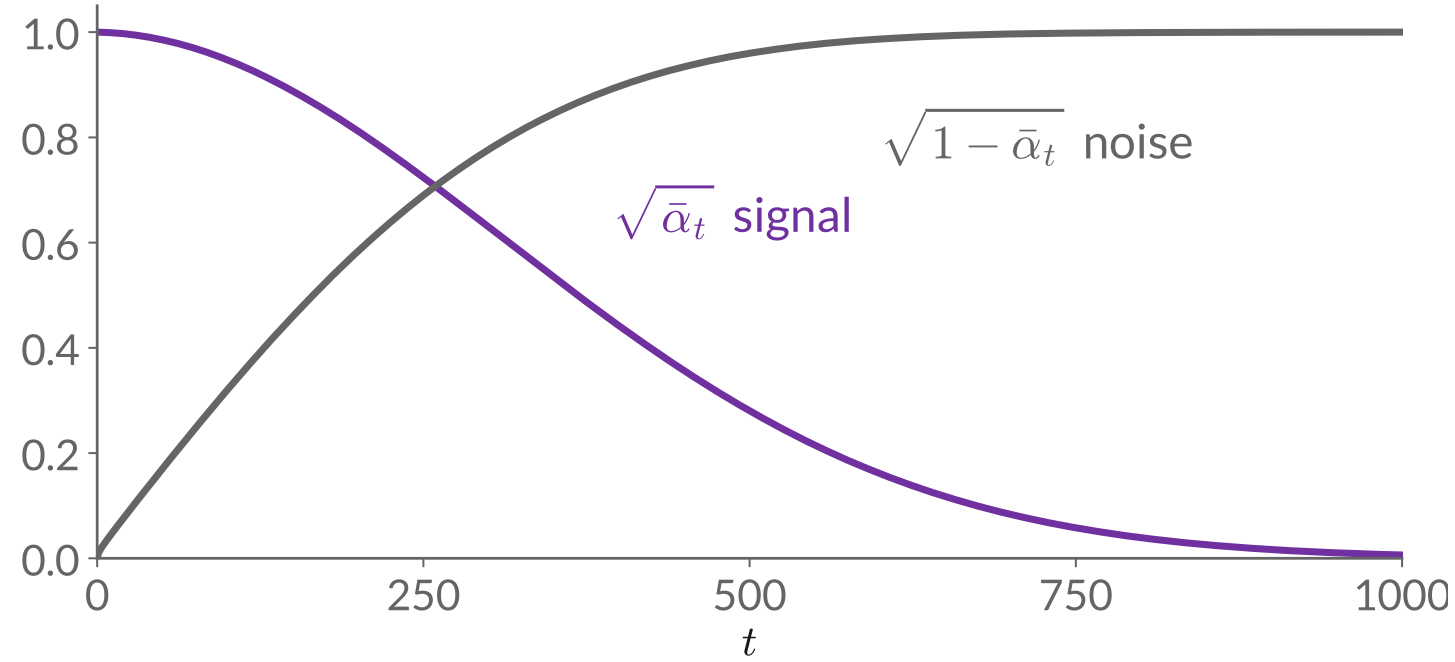
Repeat the two-step argument t times

$\bar{\alpha}_t$: fraction of the signal variance that remains

2-D check: 300 single steps and one jump give the same distribution

Training never simulates the chain

How much signal remains





Signal and noise coefficients for the linear schedule

The signal amplitude halves by $t \approx 370$

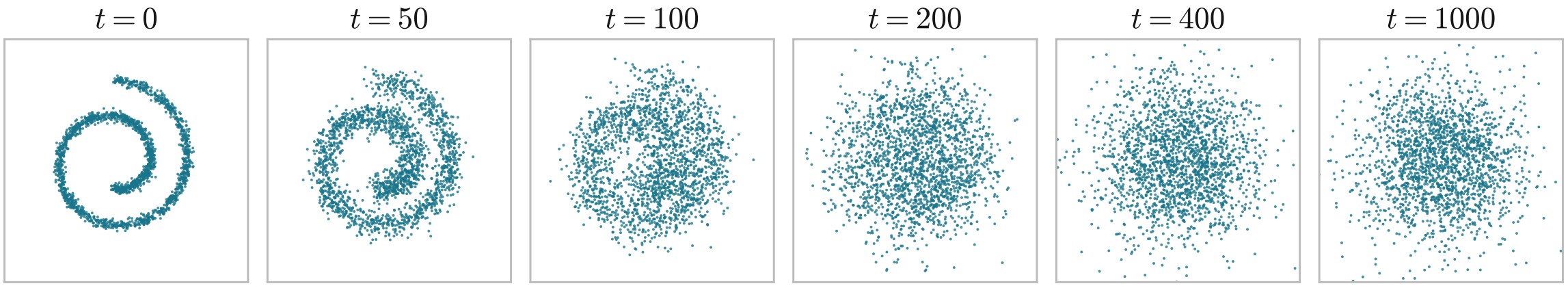
The last few hundred steps are nearly pure noise

Section 05: other schedules spend the steps differently

Source: photo: scikit-image (CC0)

The data distribution dissolves

2-D example: 2000 points from a spiral (teal), each noised to level t



Each point: a shrunk data point plus Gaussian noise, made with the jump formula

By $t = 1000$ the spiral is a standard Gaussian blob

The densities of these clouds: slide 7

Noise hides fine detail first

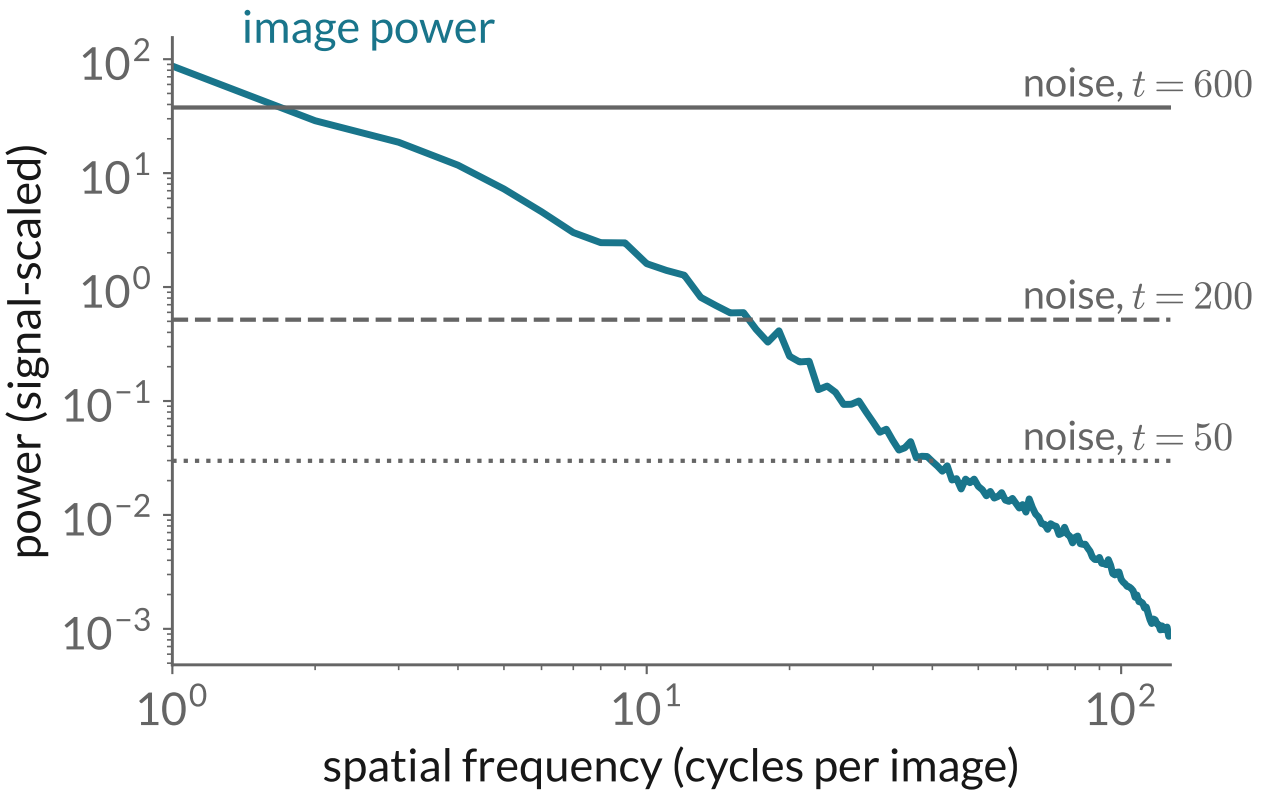


Photo: most power at low spatial frequencies (coarse shapes)

White noise: equal power at every frequency

Above a line, noise wins: high frequencies are hidden first

So generation will form coarse structure first

Source: photo: scikit-image (CC0)

Lecture outline

01 Why generate in small steps

02 Turning data into noise

03 Learning to remove noise

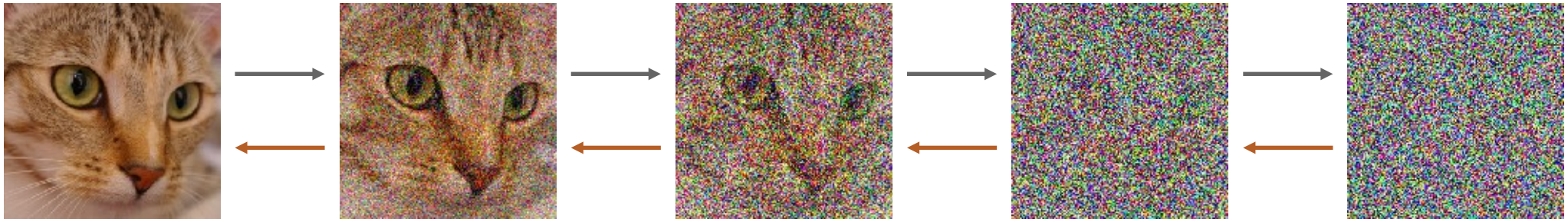
04 Where the loss comes from

05 The denoising network

06 Scores and faster sampling

07 Results and comparison

Generation runs the chain backward



Each orange arrow: one learned step $p_\theta(x_{t-1} \mid x_t)$

Draw $x_T \sim \mathcal{N}(0, I)$, then apply T learned steps to reach x_0

θ: the network's parameters; one network serves every step t

What should one step predict, and how do we train it?

Source: photo: scikit-image (CC0)

The learned reverse step

$$p_\theta(x_{t-1} \mid x_t) = \mathcal{N}\big(\underbrace{\mu_\theta(x_t, t)}_{\text{mean: from a network}}, \underbrace{\sigma_t^2 I}_{\text{variance: fixed}}\big)$$

Slide 8: one small step back is nearly Gaussian

$\mathcal{N}(\mu, \sigma^2 I)$: a Gaussian centered at μ with spread σ in every pixel

DDPM fixes $\sigma_t^2 = \beta_t$; only the mean is learned

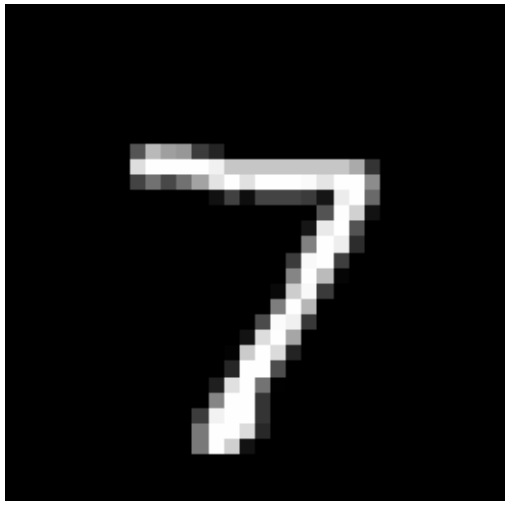
One reverse step: $x_{t-1} = \mu_\theta(x_t, t) + \sigma_t z$, with $z \sim \mathcal{N}(0, I)$

What should the network output? We need a target.

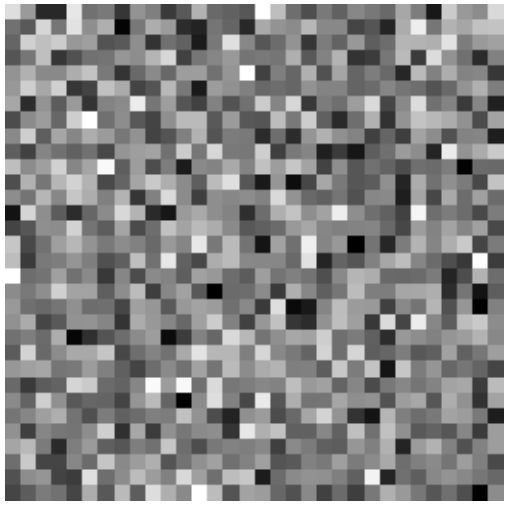
A training pair we can make ourselves

Take a training image, choose t and the noise: we know the right answer

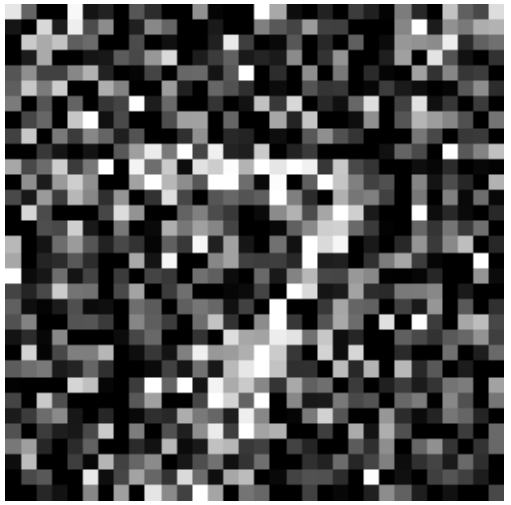
training image x_0



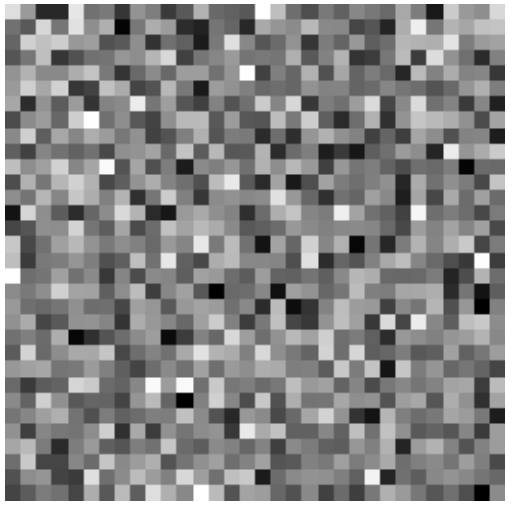
noise ϵ (we choose it)



input x_{300}



prediction $\epsilon_\theta(x_{300}, 300)$



Network input: x_t and t. Target: the noise ϵ we used

Right: prediction of our trained model; the rest of this section uses it

The training loss

$$L_{\text{simple}}(\theta) = \mathbb{E}_{x_0,\, t,\, \epsilon}\left[\left\| \underbrace{\epsilon}_{\text{noise we added}} - \epsilon_\theta\Big(\underbrace{\sqrt{\bar{\alpha}_t}\, x_0 + \sqrt{1-\bar{\alpha}_t}\, \epsilon}_{\text{the noisy input } x_t} ,\, t \Big) \right\|^2 \right]$$

x_0 : a training image; t: uniform on $\{1, \ldots, T\}$; $\epsilon \sim \mathcal{N}(0, I)$

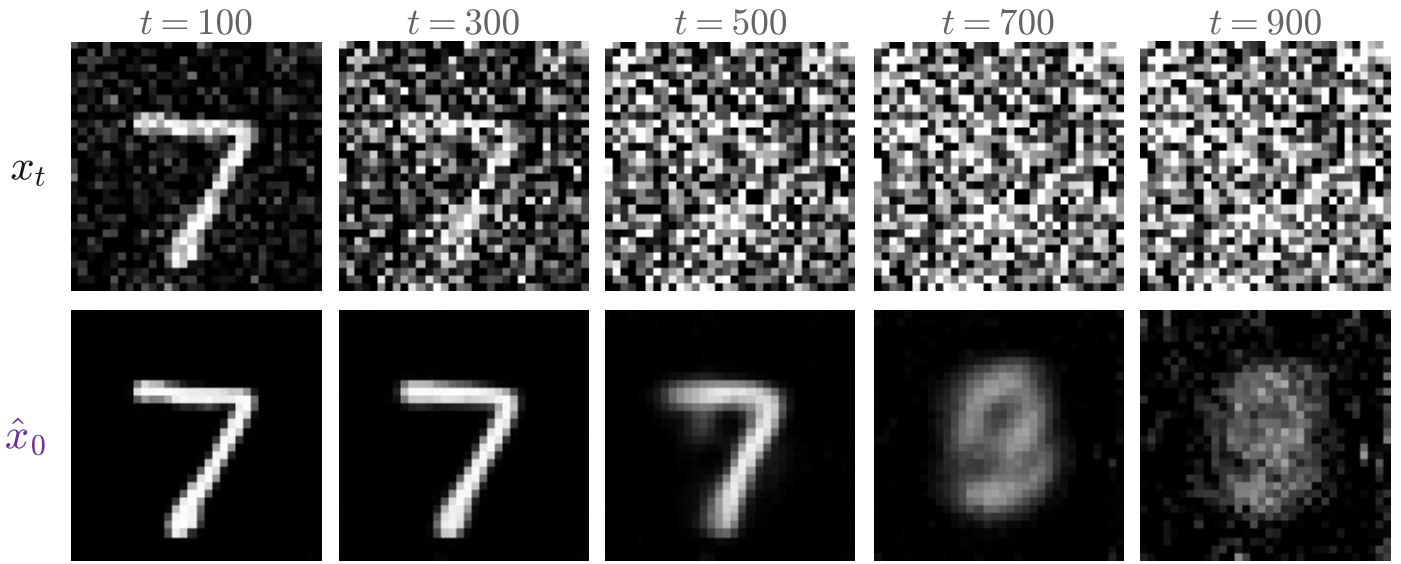
$\epsilon_\theta(x_t, t)$: the network's guess of the noise

Plain regression: no game, no chain during training

Section 04: this is a reweighted likelihood bound

Predicted noise gives a clean-image estimate

$$\hat{x}_0 = \underbrace{\frac{x_t - \sqrt{1-\bar{\alpha}_t}\,\epsilon_\theta(x_t, t)}{\sqrt{\bar{\alpha}_t}}}_{\text{solve the jump formula for } x_0}$$



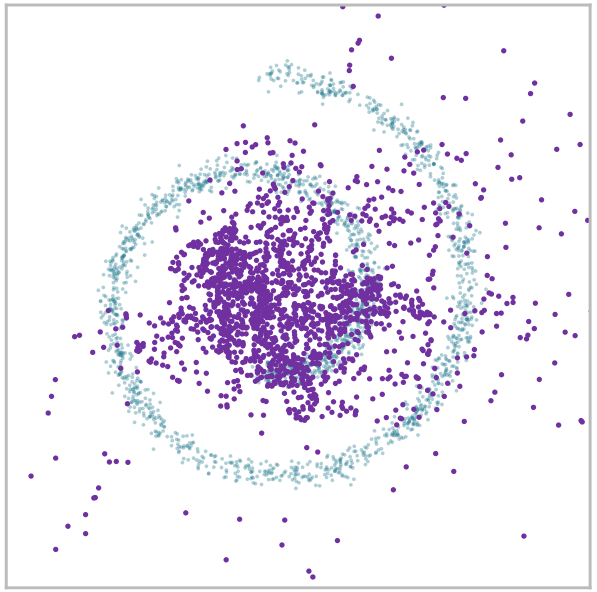
Low noise: a sharp estimate

High noise: blurry, an average of the digits that fit

$\hat{x}_0$ **is an estimate, not a sample**

Why not denoise in one jump?

one jump: $\hat{x}_0$ from x_{1000}

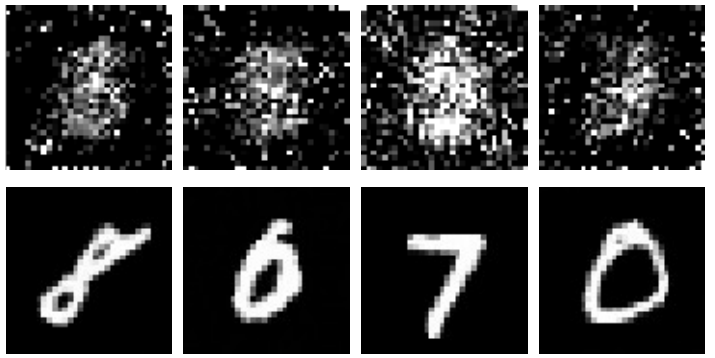


1000 small steps



one jump $\hat{x}_0$

1000 steps



The best single guess is an average of all possibilities, not one of them

Purple: $\hat{x}_0$ from pure noise sits in the middle of the data, not on it

The one-step problem of slides 5–8, now in our trained network

Training algorithm

Training (DDPM, Algorithm 1)

1 repeat

2 $x_0 \sim$ training data

3 $t \sim \mathrm{Uniform}\{1, \ldots, T\}$

4 $\epsilon \sim \mathcal{N}(0, I)$

5 gradient step on $\left\| \epsilon - \epsilon_\theta \left(\sqrt{\bar{\alpha}_t}\, x_0 + \sqrt{1 - \bar{\alpha}_t}\, \epsilon,\ t \right) \right\|^2$

6 until converged

One random noise level per example

Every t trains the same network

No chain is simulated

Source: DDPM (2020)

One reverse step

$$x_{t-1} = \underbrace{\frac{1}{\sqrt{\alpha_t}}}_{\text{undo the shrink}} \left(x_t - \underbrace{\frac{\beta_t}{\sqrt{1-\bar{\alpha}_t}} \epsilon_\theta(x_t, t)}_{\text{remove part of the predicted noise}} \right) + \underbrace{\sigma_t z}_{\text{fresh noise}}$$

The first part is the mean $\mu_\theta(x_t, t)$, written with the predicted noise

$z \sim \mathcal{N}(0, I)$, $\sigma_t^2 = \beta_t$; no fresh noise at the last step

Why exactly these coefficients: Section 04

Sampling algorithm

Sampling (DDPM, Algorithm 2)

1 $x_T \sim \mathcal{N}(0, I)$

2 for $t = T, \ldots, 1$:

3 $\quad z \sim \mathcal{N}(0, I)$ if $t > 1$, else $z = 0$

4 $\quad x_{t-1} = \frac{1}{\sqrt{\alpha_t}}\left(x_t - \frac{\beta_t}{\sqrt{1-\bar{\alpha}_t}}\epsilon_\theta(x_t, t)\right) + \sigma_t z$

5 return x_0

T network calls, one after another

Fresh noise at every step except the last

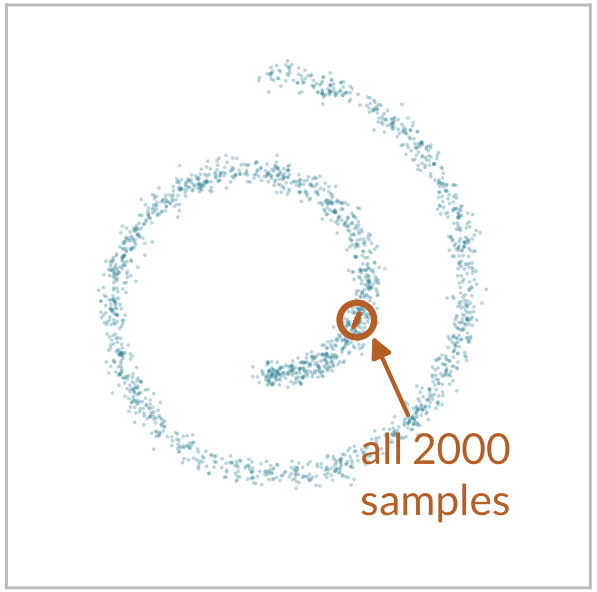
Source: DDPM (2020)

Do not forget the fresh noise

with fresh noise

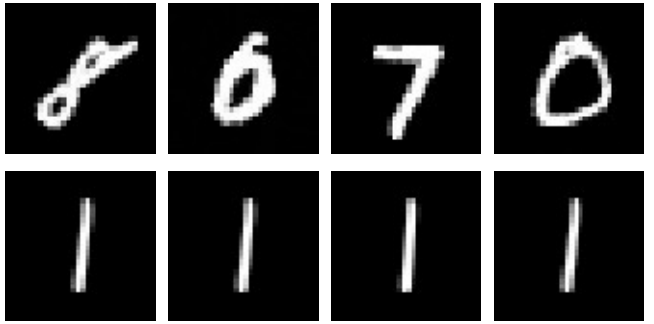


without ($z = 0$)



with fresh noise

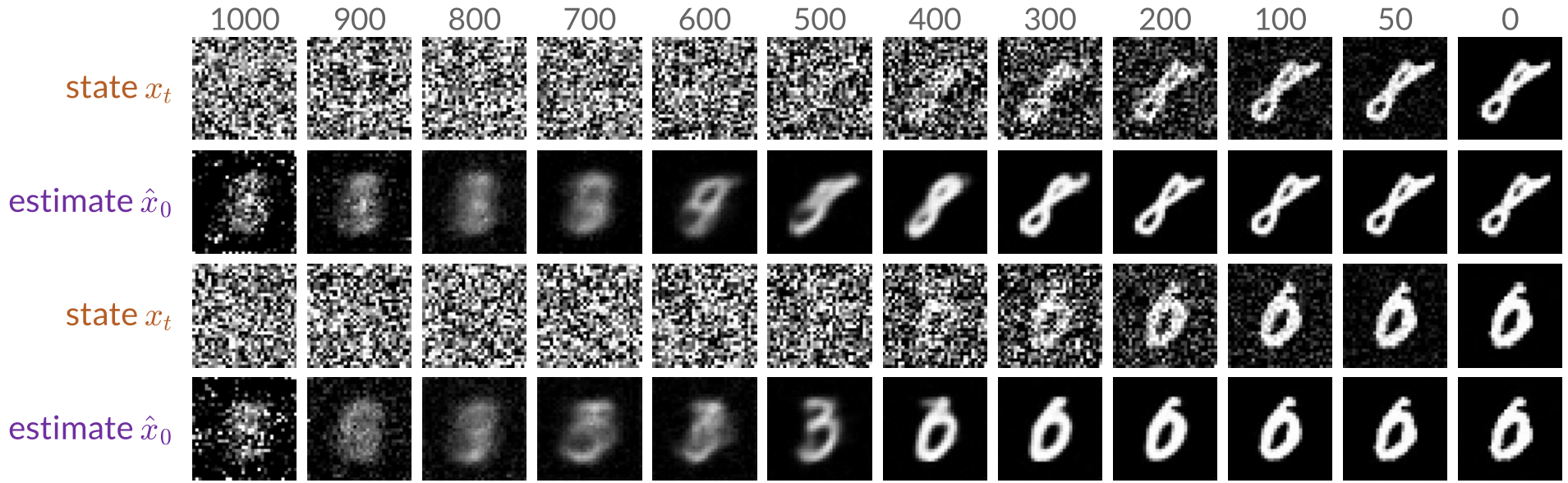
without ($z = 0$)



Same model, same starting noise; only z is set to zero

Without fresh noise, samples collapse: one spiral point, the same thin 1 from every start

Watching one sample form



Top: the sampled states x_t, noisy until the very end

Bottom: the estimate $\hat{x}_0$ at each step; never fed back as a state

The digit's identity settles early; strokes sharpen late

Coarse structure first, details last



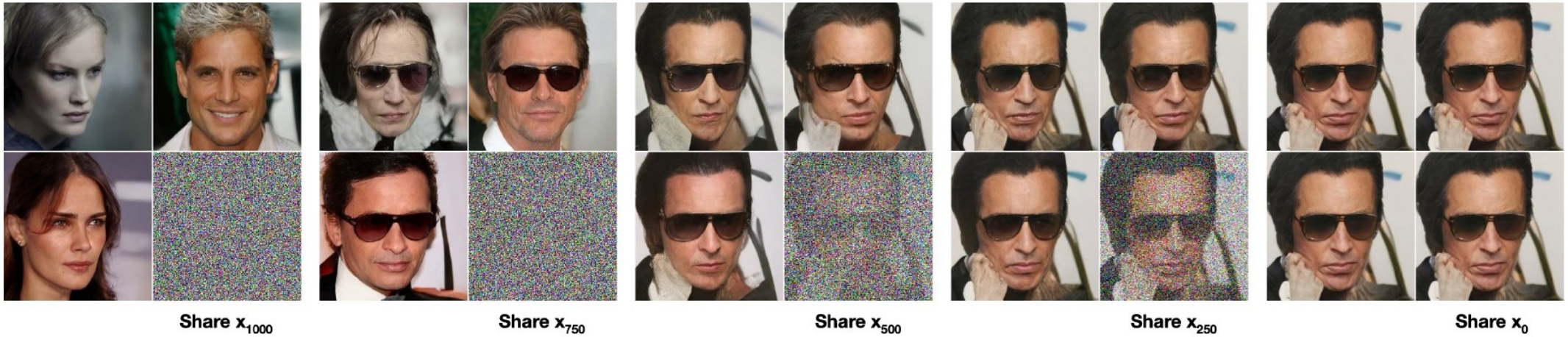
CIFAR-10: $\hat{x}_0$ at successive steps, left to right, for several samples

Overall layout and color appear first; edges and texture last

Source: DDPM (2020)

What is decided when

Keep x_t from one run, then rerun the remaining steps with new noise



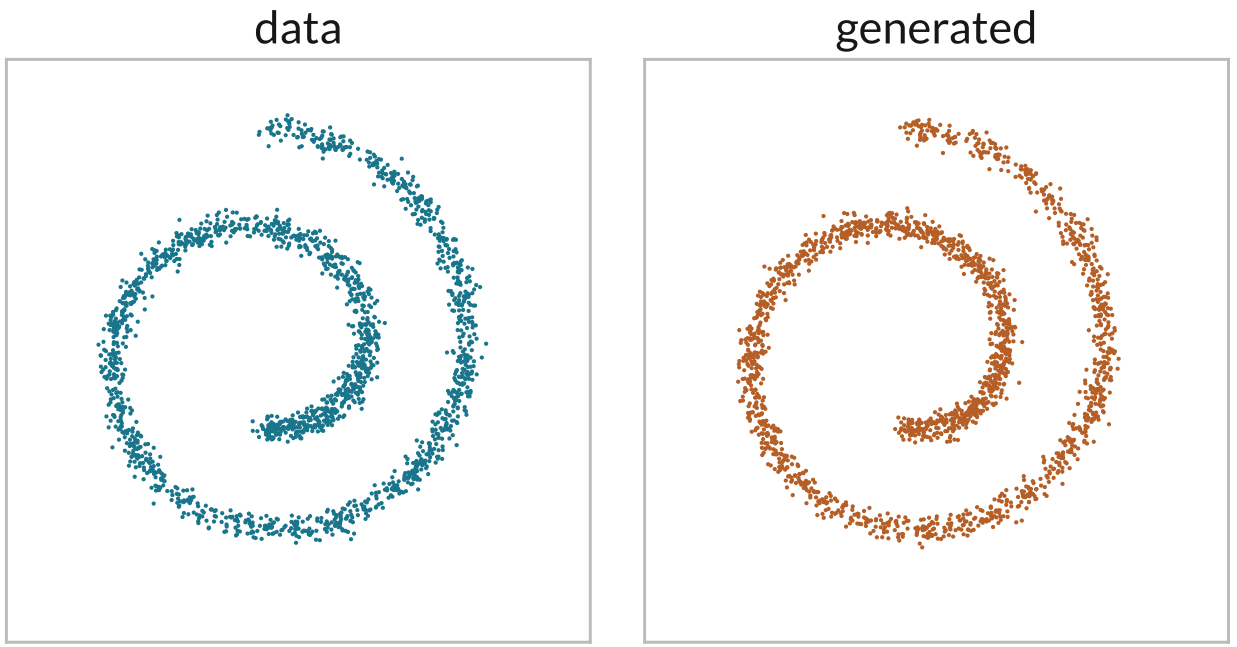
Shared x_{1000} : unrelated faces. Shared x_{500} : same pose and glasses

Shared x_{250} : same face, small differences in detail

Source: DDPM (2020)

Samples from a small model





MNIST: 1.6M-parameter U-Net, 16k training steps (50 min on a laptop GPU), 1000 sampling steps

Spiral: 4-layer MLP, 30k steps

The cost: one network call per step

Our MNIST model on a laptop GPU

One training step, 128 images	**170 ms**	1 call
Generate 64 images, DDPM	**18 s**	1000 calls
Generate 64 images, after Section 06	**0.9 s**	50 calls

NFE (number of function evaluations): network calls per sample

Original DDPM: 1000 NFE per image. A GAN: 1

Lecture outline

01 Why generate in small steps

02 Turning data into noise

03 Learning to remove noise

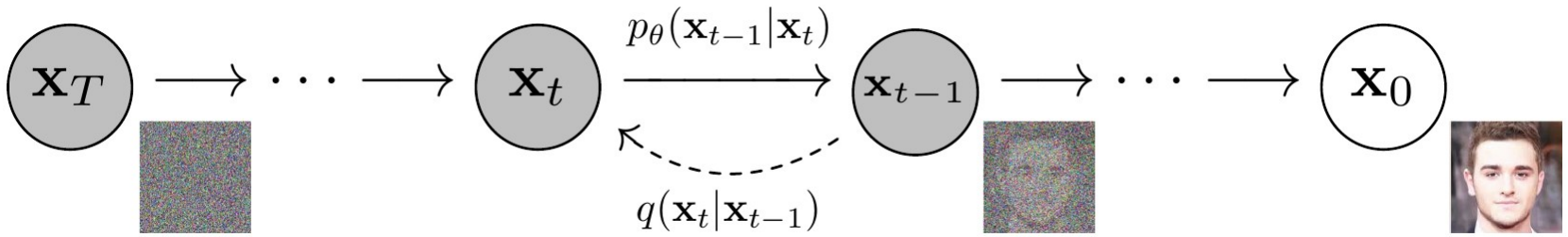
04 Where the loss comes from

05 The denoising network

06 Scores and faster sampling

07 Results and comparison

Diffusion as a latent-variable model



VAE (L15)

Latent z: a short code

Encoder $q_\phi(z \mid x)$: learned

Decoder $p_\theta(x \mid z)$: learned

Diffusion

Latents $x_{1:T} = (x_1, \ldots, x_T)$: all the noisy images

Encoder $q(x_{1:T} \mid x_0)$: fixed noising, no parameters

Decoder p_θ: the learned reverse chain

So the VAE's likelihood bound applies

Source: DDPM (2020)

A bound on the likelihood

$$\log p_\theta(x_0) \geq \underbrace{\mathbb{E}_q\left[\log \frac{p_\theta(x_{0:T})}{q(x_{1:T} \mid x_0)}\right]}_{\text{ELBO, as in L15}} =: -L$$

$$p_\theta(x_{0:T}) = \underbrace{p(x_T)\prod_{t=1}^{T} p_\theta(x_{t-1} \mid x_t)}_{\text{generation chain: start from noise}}$$

$$q(x_{1:T} \mid x_0) = \underbrace{\prod_{t=1}^{T} q(x_t \mid x_{t-1})}_{\text{noising chain: start from data}}$$

$\mathbb{E}_q$: average over noisy chains produced from a training image x_0

Train by minimizing L; next: split it into one term per step

Reverse a forward step with Bayes' rule

$$q(x_t \mid x_{t-1}) = \underbrace{q(x_t \mid x_{t-1}, x_0)}_{\text{the step ignores } x_0} = \underbrace{\frac{q(x_{t-1} \mid x_t, x_0)\, q(x_t \mid x_0)}{q(x_{t-1} \mid x_0)}}_{\text{Bayes' rule with } x_0 \text{ fixed}}$$

Training knows x_0, so we may condition on it

$q(x_{t-1} \mid x_t)$ alone is the spiral-shaped distribution (slide 8)

$q(x_{t-1} \mid x_t, x_0)$: one Gaussian (slide 45)

Substituting this into the bound makes most terms cancel

The bound splits into one term per step

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

L_T : no parameters

L_{t-1} : each reverse step vs. a known target

L_0 : last step

$D_{\mathrm{KL}}(q \,\|\, p)$: how far p is from q; zero only when they match (L15)

Each learned step is compared with a Gaussian we can compute

The algebra: next slide (optional)

The telescoping algebra

Write out the log ratio, then use Bayes' rule (previous slide) for $t \geq 2$:

$$
\begin{aligned}
-L &= \mathbb{E}_q\left[\log p(x_T) + \sum_{t=1}^{T} \log \frac{p_\theta(x_{t-1} \mid x_t)}{q(x_t \mid x_{t-1})}\right] \\
&= \mathbb{E}_q\left[\log p(x_T) + \sum_{t=2}^{T} \log \frac{p_\theta(x_{t-1} \mid x_t)}{q(x_{t-1} \mid x_t, x_0)} + \underbrace{\sum_{t=2}^{T} \log \frac{q(x_{t-1} \mid x_0)}{q(x_t \mid x_0)}}_{= \log q(x_1|x_0) - \log q(x_T|x_0)} + \log \frac{p_\theta(x_0 \mid x_1)}{q(x_1 \mid x_0)}\right] \\
&= \mathbb{E}_q\left[\log \frac{p(x_T)}{q(x_T \mid x_0)} + \sum_{t=2}^{T} \log \frac{p_\theta(x_{t-1} \mid x_t)}{q(x_{t-1} \mid x_t, x_0)} + \log p_\theta(x_0 \mid x_1)\right]
\end{aligned}
$$

Each expected log ratio is minus a KL divergence, which gives the previous slide

With one step, the bound is the VAE's

Set $T = 1$: jump from data x_0 to noise x_1, then one learned step back

$$L = \mathbb{E}_q \Big[\underbrace{D_{\mathrm{KL}}\big(q(x_1 \mid x_0) \,\|\, p(x_1)\big)}_{\text{code vs. prior}} - \underbrace{\log p_\theta(x_0 \mid x_1)}_{\text{reconstruction}} \Big]$$

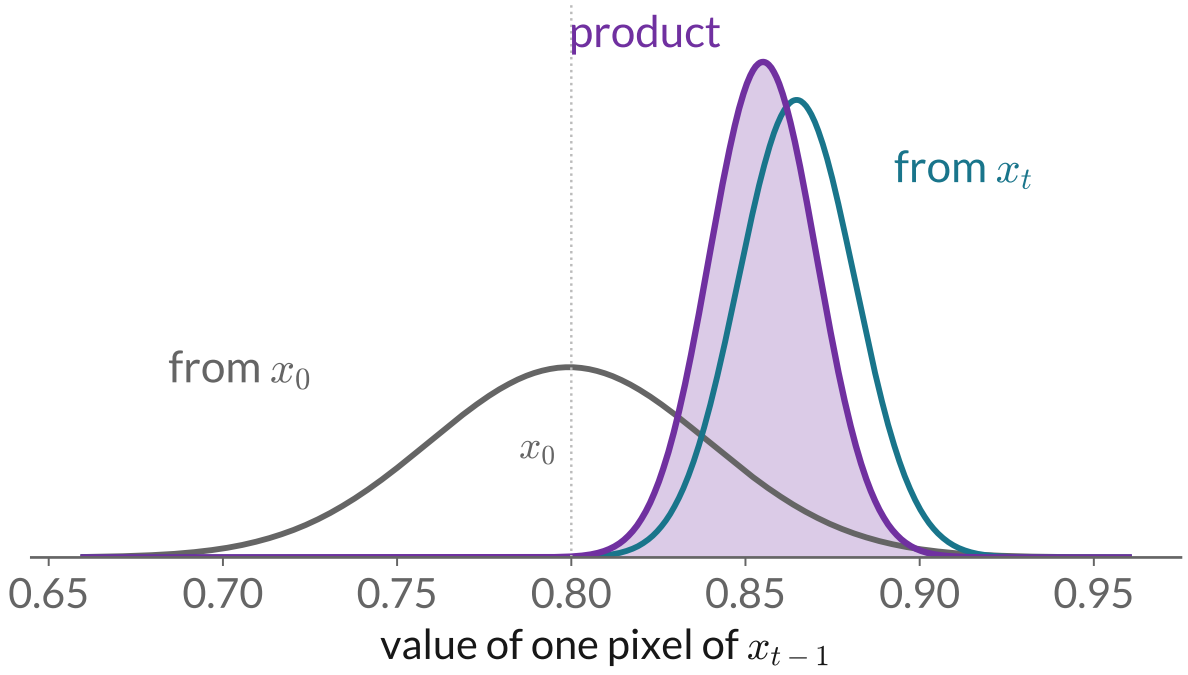
The VAE loss of L15: a fixed encoder and one Gaussian decoder step (the blur of slide 5)

$T = 1000$: the same decoder, split into small, nearly Gaussian steps

Diffusion: a VAE whose decoder takes many small steps

The target is a Gaussian

$$q(x_{t-1} \mid x_t, x_0) \propto \underbrace{q(x_t \mid x_{t-1})}_{\text{one forward step}} \cdot \underbrace{q(x_{t-1} \mid x_0)}_{\text{jump from } x_0}$$



One pixel at $t = 10$ (exact)

Both factors: Gaussians in x_{t-1}

Their product is a narrower Gaussian, between the two

Its center and width have closed forms

Completing the square

For one pixel x of x_{t-1}, add the two exponents:

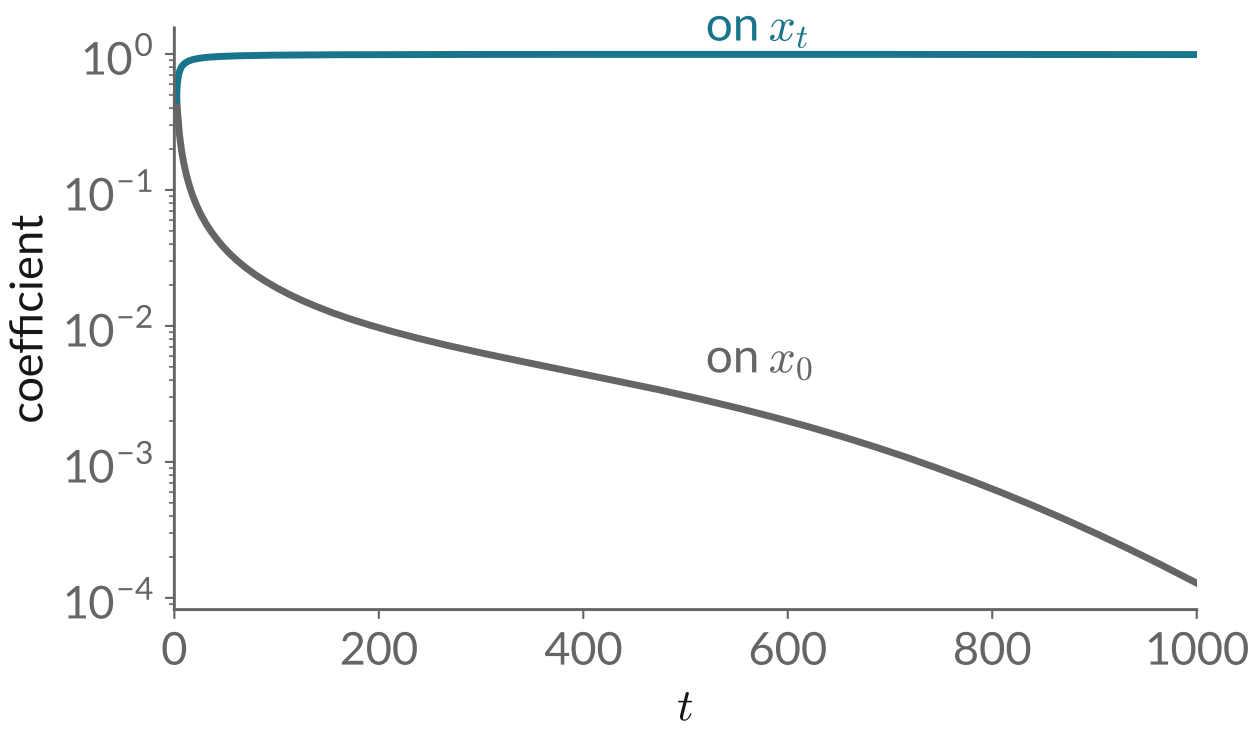
$$-\log q(x \mid x_t, x_0) = \frac{(x_t - \sqrt{\alpha_t}\, x)^2}{2\beta_t} + \frac{(x - \sqrt{\bar{\alpha}_{t-1}}\, x_0)^2}{2(1-\bar{\alpha}_{t-1})} + \text{const} = \frac{(x - \tilde{\mu}_t)^2}{2\tilde{\beta}_t} + \text{const}$$

Match the coefficients of x^2 and x:

$$\underbrace{\tilde{\beta}_t = \frac{1-\bar{\alpha}_{t-1}}{1-\bar{\alpha}_t}\,\beta_t}_{\text{variance}} \qquad \tilde{\mu}_t(x_t, x_0) = \underbrace{\frac{\sqrt{\bar{\alpha}_{t-1}}\,\beta_t}{1-\bar{\alpha}_t}\, x_0}_{\text{weight on } x_0} + \underbrace{\frac{\sqrt{\alpha_t}\,(1-\bar{\alpha}_{t-1})}{1-\bar{\alpha}_t}\, x_t}_{\text{weight on } x_t}$$

The same holds for every pixel independently

How the target mixes its two inputs



Coefficients in $\tilde{\mu}_t$ for the linear schedule

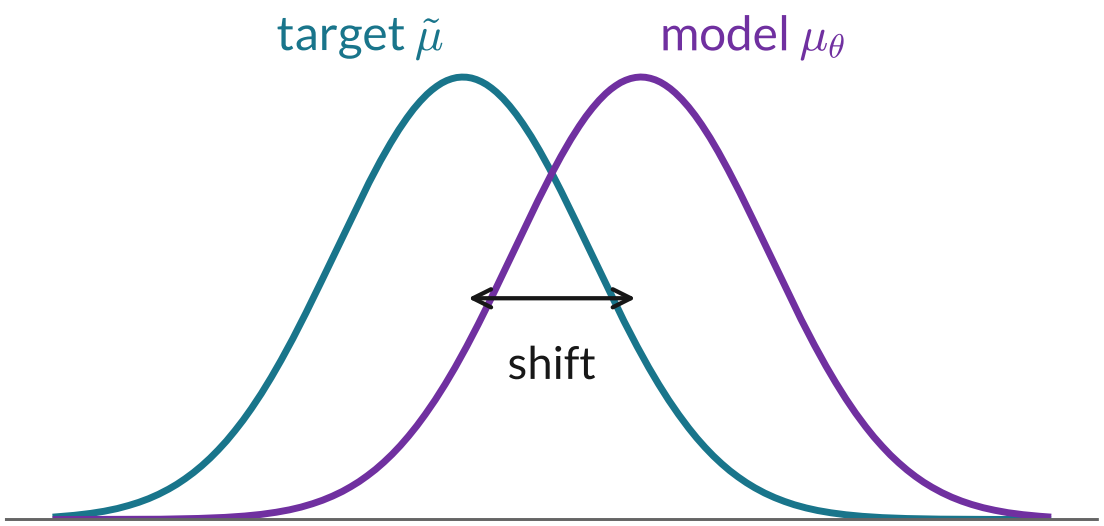
On x_t : close to 1 at every step

On x_0 : a small nudge that shrinks as t grows

Each reverse step moves only a little toward x_0

Matching two Gaussians

$$D_{\mathrm{KL}}\left(\mathcal{N}(\tilde{\mu}, \sigma^2 I) \,\|\, \mathcal{N}(\mu_\theta, \sigma^2 I)\right) = \underbrace{\frac{\|\tilde{\mu} - \mu_\theta\|^2}{2\sigma^2}}_{\text{squared distance of the means}}$$



Same fixed variance: only the centers differ

Target $\tilde{\mu}_t$: needs x_0, known in training

Model $\mu_\theta(x_t, t)$: must work without x_0

Each step becomes a regression on means

Write the target mean with the noise

From the jump formula, $x_0 = \left(x_t - \sqrt{1-\bar{\alpha}_t}\,\epsilon\right)/\sqrt{\bar{\alpha}_t}$. Substitute into $\tilde{\mu}_t$:

$$\tilde{\mu}_t = \underbrace{\frac{1}{\sqrt{\alpha_t}}\left(x_t - \frac{\beta_t}{\sqrt{1-\bar{\alpha}_t}}\,\epsilon\right)}_{\text{uses the true noise}} \qquad \mu_\theta(x_t, t) = \underbrace{\frac{1}{\sqrt{\alpha_t}}\left(x_t - \frac{\beta_t}{\sqrt{1-\bar{\alpha}_t}}\,\epsilon_\theta(x_t, t)\right)}_{\text{same form, predicted noise}}$$

Choose the model's mean in the same form, with ϵ replaced by the network's guess

This is exactly the reverse step of slide 29

The network only has to predict the noise

The KL term becomes a noise error

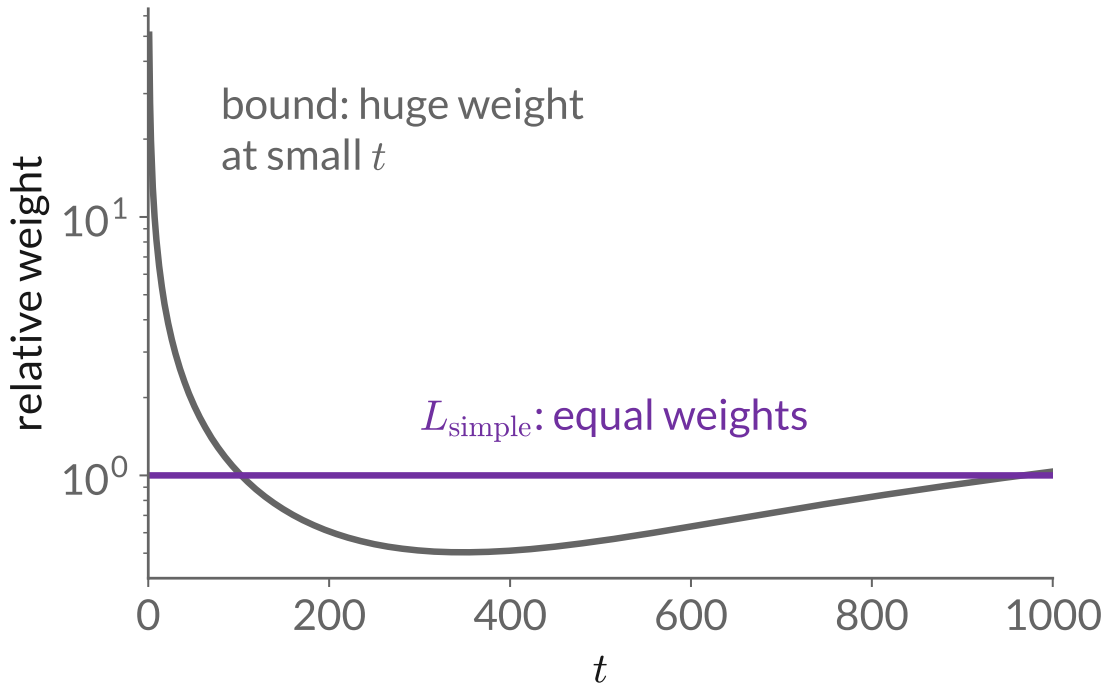
$$L_{t-1} = \underbrace{\frac{\beta_t^2}{2\sigma_t^2\,\alpha_t\,(1-\bar{\alpha}_t)}}_{\text{weight } w_t}\ \underbrace{\mathbb{E}\left\|\epsilon - \epsilon_\theta(x_t, t)\right\|^2}_{\text{noise regression}} + C$$

Subtract the two means: everything except the noise terms cancels

C: does not depend on θ

Minimizing the bound = noise regression, weighted by w_t

Dropping the weights



Objective	IS	FID
$\tilde{\mu}$ prediction (baseline)		
L, learned diagonal Σ	7.28 ± 0.10	23.69
L, fixed isotropic Σ	8.06 ± 0.09	13.22
$\|\|\tilde{\mu} - \tilde{\mu}_\theta\|\|^2$	–	–
ϵ prediction (ours)		
L, learned diagonal Σ	–	–
L, fixed isotropic Σ	7.67 ± 0.13	13.51
$\|\|\tilde{\epsilon} - \epsilon_\theta\|\|^2$ (L_{simple})	$\mathbf{9.46 \pm 0.11}$	**3.17**

With $\sigma_t^2 = \beta_t$: the bound weights $t = 1$ about 49 times more than $t = T$

L_{simple} : equal weights, more effort on the noisier steps

DDPM ablation (right): ϵ prediction with L_{simple} **gives the best FID, 3.17**

Source: DDPM (2020)

The derivation in one view

1	**Bound over the whole chain**	the VAE's ELBO with latents $x_{1:T}$	slide 40
2	**One KL term per step**	against $q(x_{t-1} \mid x_t, x_0)$	slide 42
3	**That target is Gaussian**	mean $\tilde{\mu}_t(x_t, x_0)$, variance $\tilde{\beta}_t$	slide 45–46
4	**Same fixed variance**	KL = squared error between means	slide 48
5	**Write both means with noise**	weighted noise regression	slide 49–50
6	**Drop the weights**	L_{simple}, the loss of Section 03	slide 51

Training with a simple MSE fits a likelihood bound, up to the weights

The two end terms

$$L_T = D_{\mathrm{KL}}\big(q(x_T \mid x_0) \,\|\, \mathcal{N}(0, I)\big)$$

No parameters; nearly zero since $\bar{\alpha}_T \approx 4 \times 10^{-5}$

Ignored in training

$$L_0 = -\log p_\theta(x_0 \mid x_1)$$

The last step outputs pixel values. DDPM uses a Gaussian integrated over each of the 256 pixel bins

Needed for likelihoods in bits per dimension; L_{simple} treats it like any other step

Source: DDPM (2020)

Lecture outline

01 Why generate in small steps

02 Turning data into noise

03 Learning to remove noise

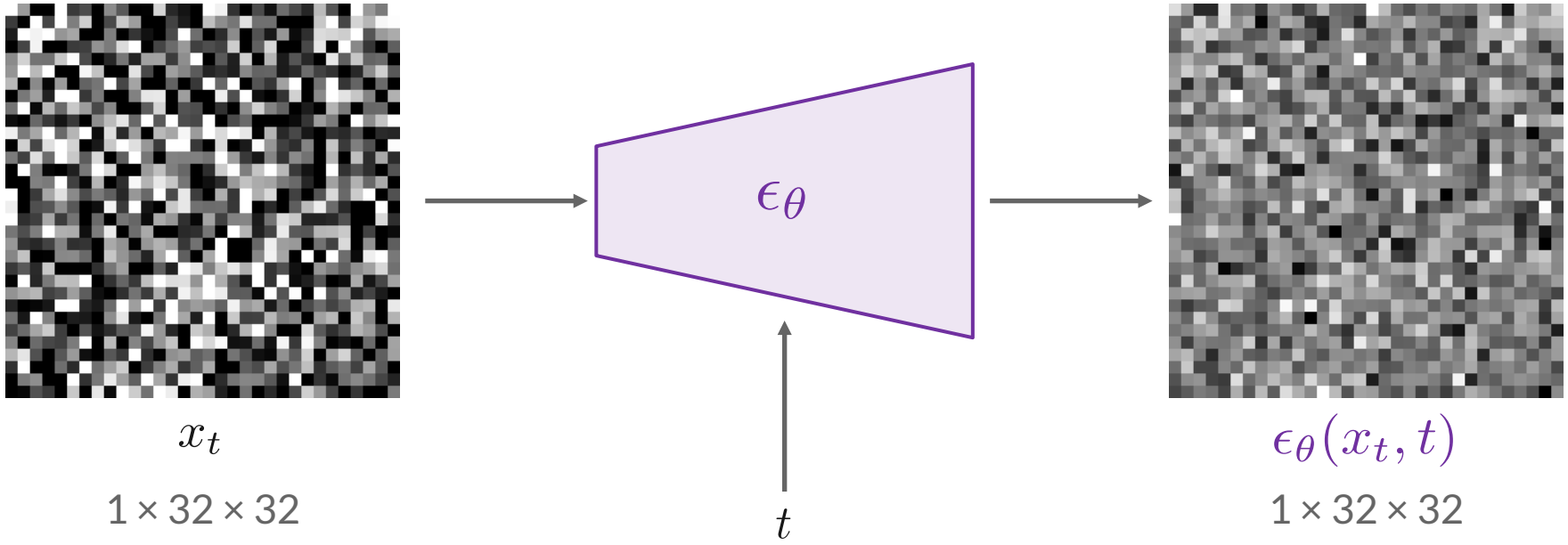
04 Where the loss comes from

05 The denoising network

06 Scores and faster sampling

07 Results and comparison

What the network must do



Output: one noise value per pixel

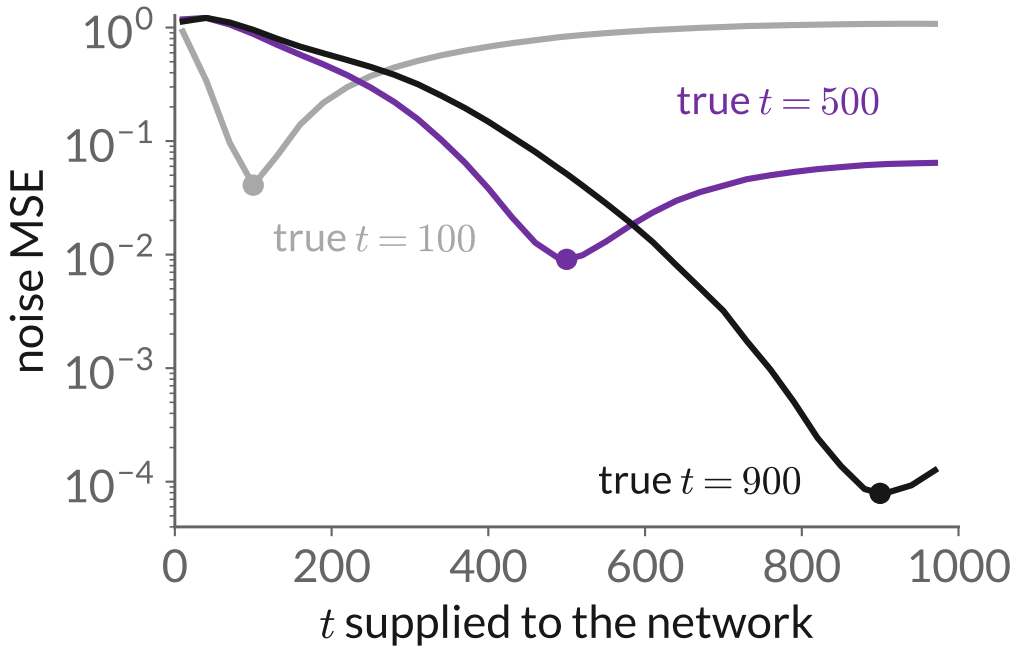
Same shape as the input

Needs local detail (which pixels carry noise) and global context (which digit this is)

A U-Net provides both (L08), plus an input for t

Any network with this input and output works (L18: a Transformer)

The network must know the noise level





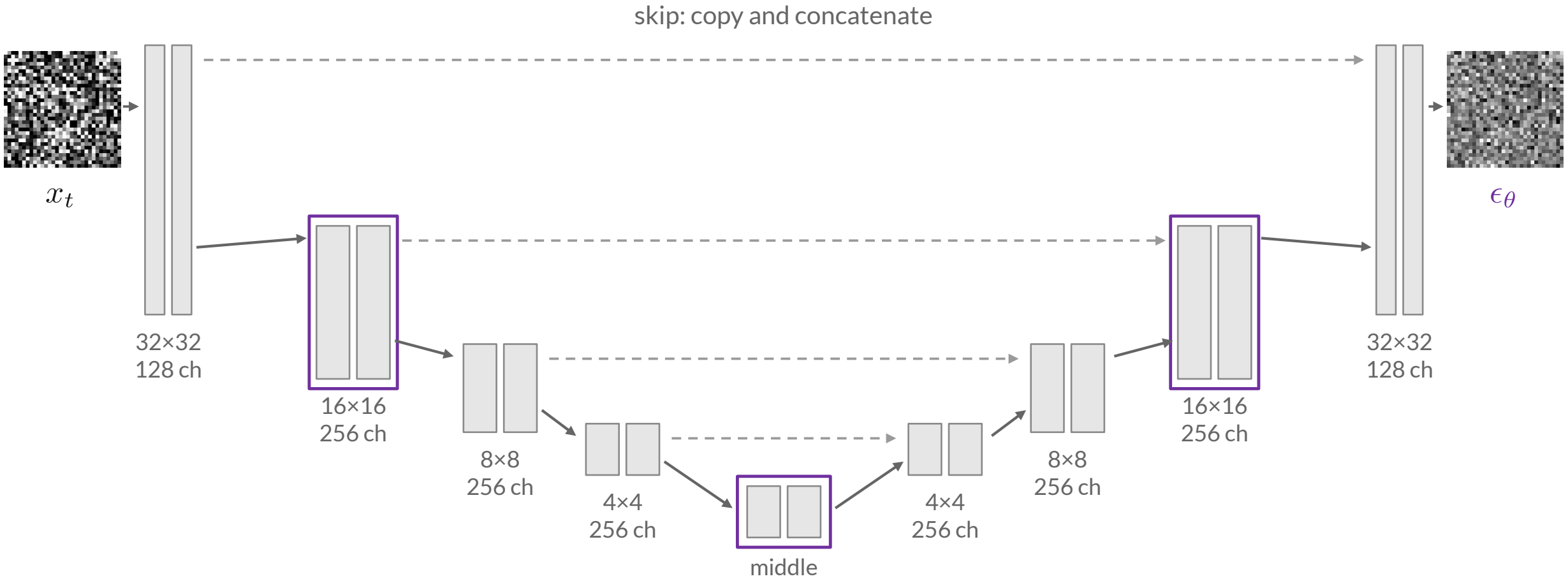
estimate $\hat{x}_0$ when the network is told t

Same noisy input, different t supplied to the network

Error is lowest when the supplied t is the true one

Told too small a t: noise is left in. Too large: the digit fades to gray

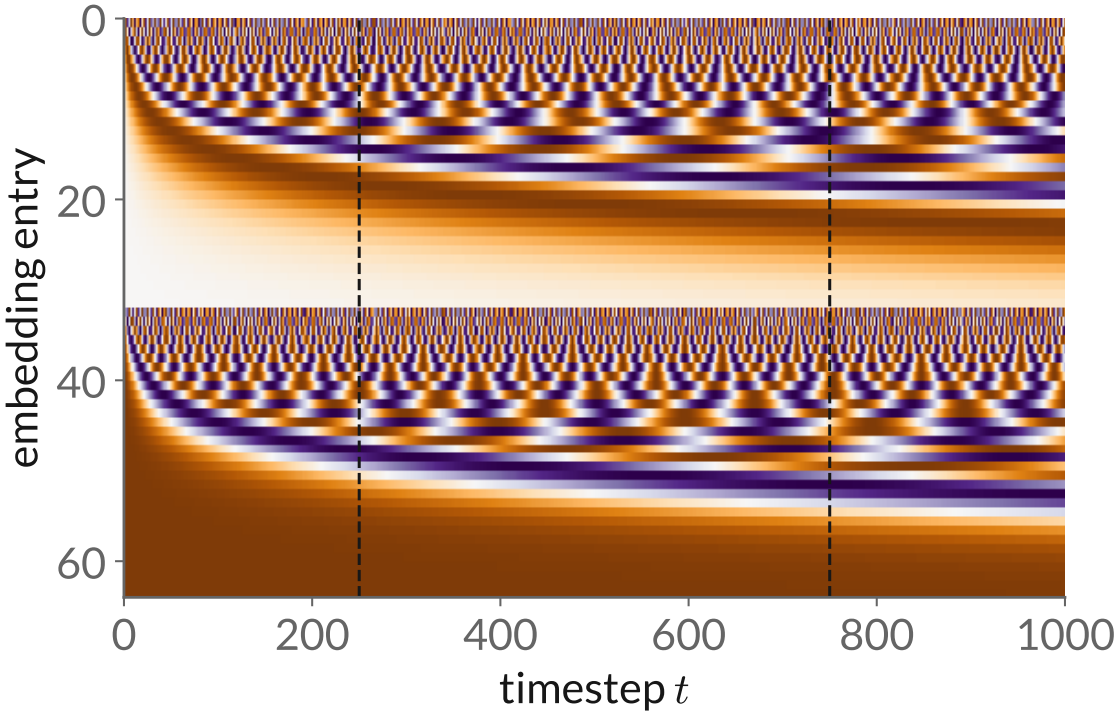
U-Net: context and detail



DDPM's CIFAR-10 network, drawn to scale: height = resolution, width = channels; two residual blocks per level
Purple: self-attention (L12), at 16×16 and in the middle. 35.7M parameters

Source: DDPM (2020)

Feeding the noise level to the network



$$e(t) = \big(\sin(\omega_k t),\ \cos(\omega_k t)\big)_{k=1,\ldots,32}$$

Frequencies ω_k from fast to slow; nearby t give similar vectors (as in L12)

A small MLP turns $e(t)$ into a scale $\gamma(t)$ and a shift $\beta(t)$ for every channel

$$h \leftarrow \underbrace{\gamma(t)}_{\text{scale}} \odot \mathrm{GroupNorm}(h) + \underbrace{\beta(t)}_{\text{shift}}$$

Inside every residual block; the same values at every pixel

DDPM: shift only. ADM: scale and shift, FID 15.08 to 13.06

Source: Diffusion Models Beat GANs (2021)

Architecture changes that helped

Channels	Depth	Heads	Attention resolutions	BigGAN up/downsample	Rescale resblock	FID 700K	FID 1200K
160	2	1	16	✗	✗	15.33	13.21
128	4					-0.21	-0.48
		4				-0.54	-0.82
			32,16,8			-0.72	-0.66
				✓		-1.20	-1.21
					✓	0.16	0.25
160	2	4	32,16,8	✓	✗	**-3.14**	**-3.00**

ADM, ImageNet 128×128 : FID change vs. a DDPM-style baseline

More heads, attention at three resolutions, BigGAN-style resampling

Together: FID 3 points lower

Source: Diffusion Models Beat GANs (2021)

Data and noise on the same scale

pixels in $[0, 255] + \epsilon$



pixels in $[-1, 1] + \epsilon$



Same noise $\epsilon \sim \mathcal{N}(0, I)$ in both

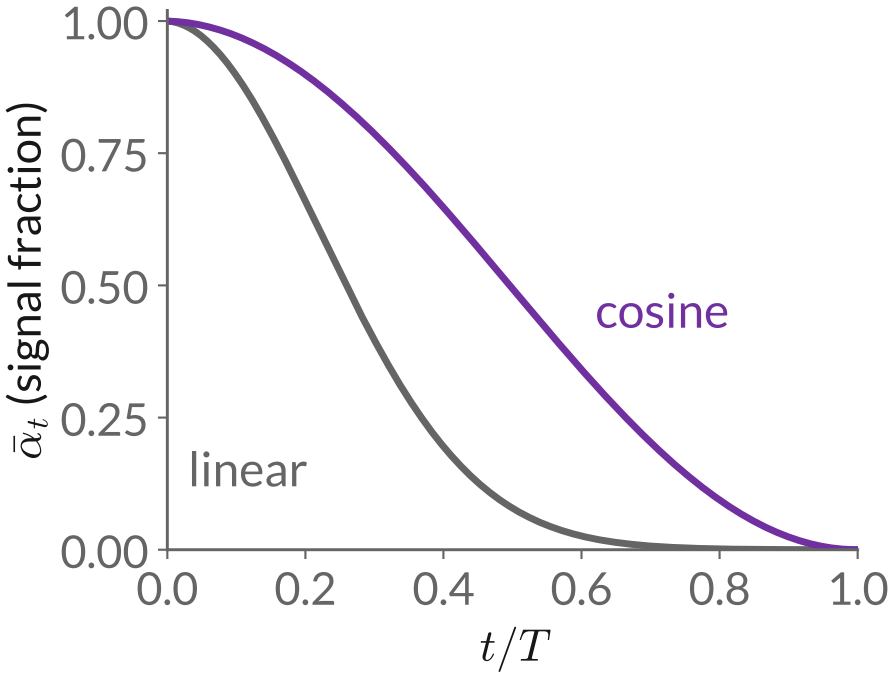
Pixels in $[0, 255]$: noise of size 1 is invisible

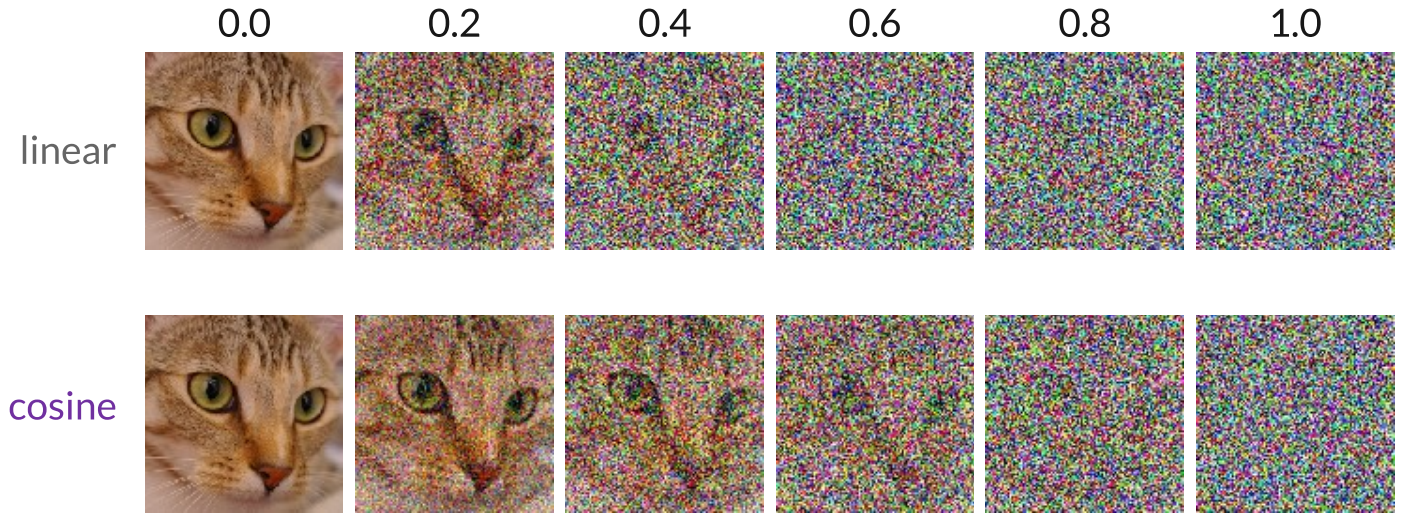
Pixels in $[-1, 1]$: noise and image are comparable

DDPM scales data to $[-1, 1]$ so the schedule means what it says

Source: photo: scikit-image (CC0)

Choosing a noise schedule





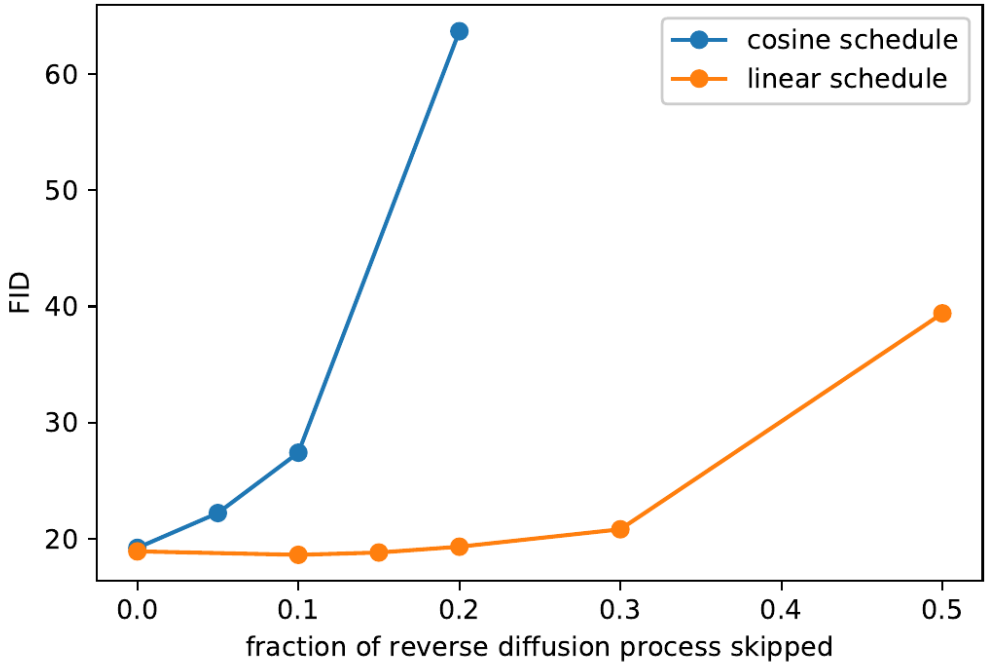
$$\bar{\alpha}_t = \frac{f(t)}{f(0)}, \qquad f(t) = \cos^2\left(\frac{t/T + 0.008}{1.008} \cdot \frac{\pi}{2}\right)$$

Linear: the last quarter is almost pure noise

Cosine: the signal fades more evenly

Source: Improved DDPM (2021) · photo: scikit-image (CC0)

The linear schedule wastes steps



Start sampling below $t = T$, skipping the first part of the reverse process

Linear: skipping 20% barely changes FID; those steps do little

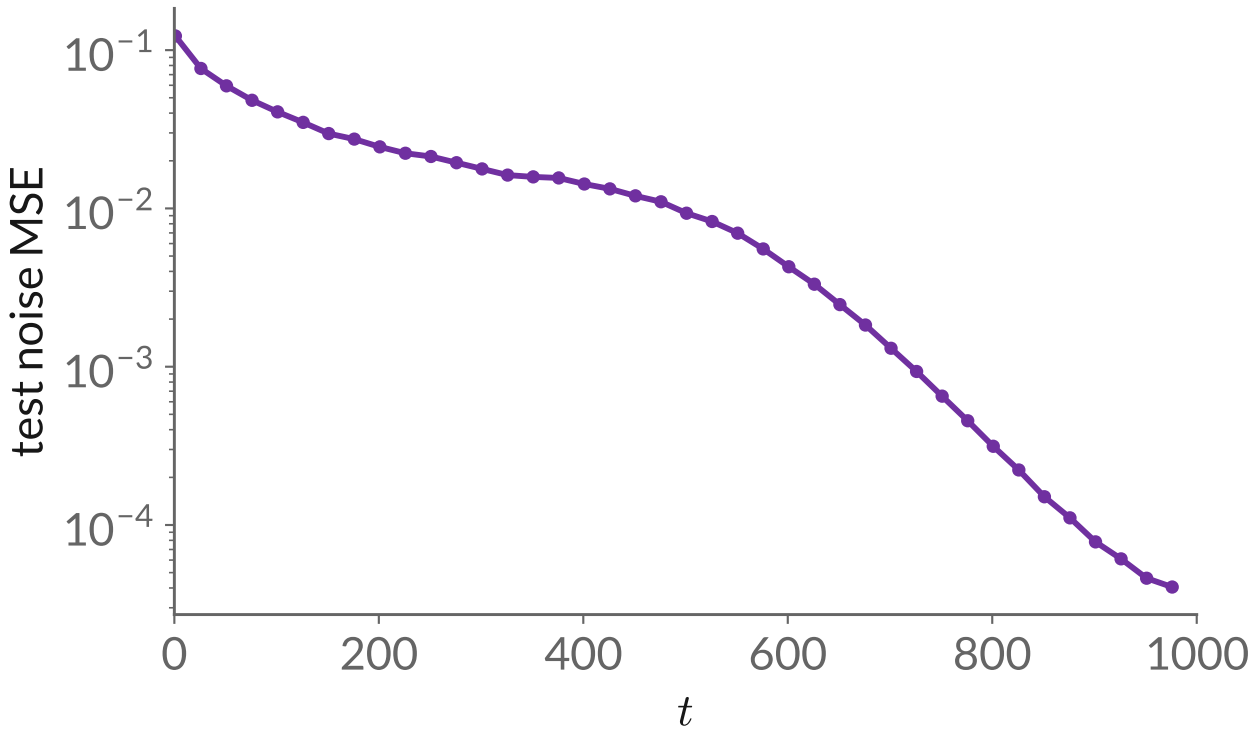
Cosine: every part matters

ImageNet 64: cosine lowers FID from 31.3 to 27.0

(CIFAR-10: FID about the same; likelihood better on both)

Source: Improved DDPM (2021)

The loss differs across noise levels



Test MSE of our model at each t

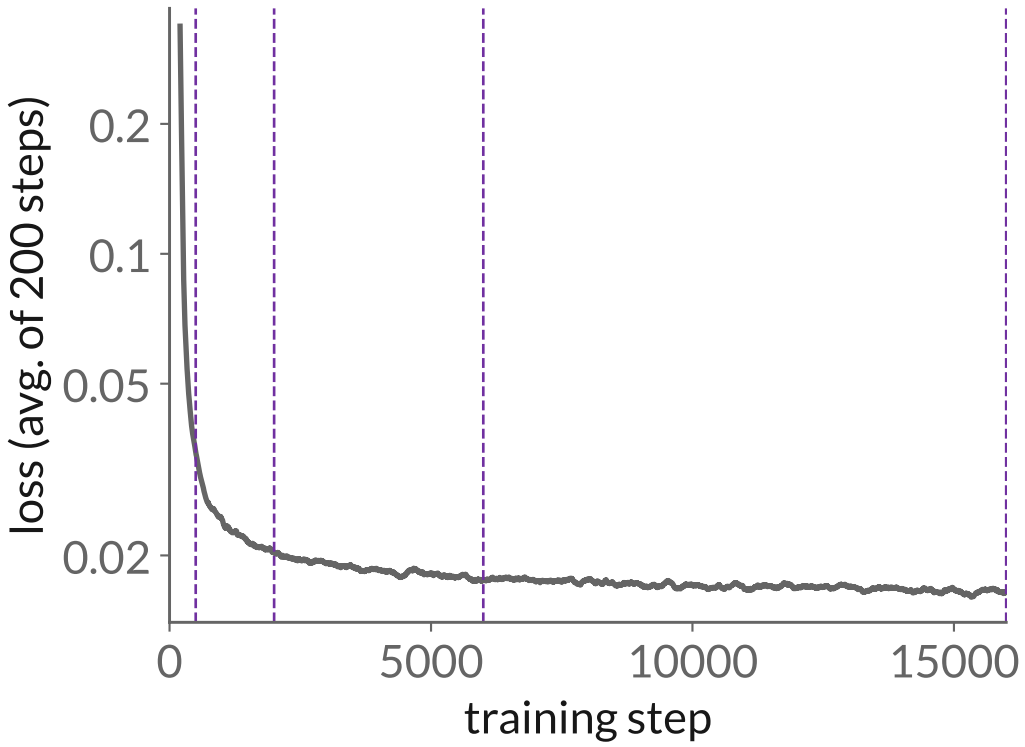
Small t: 0.12. Large t: 4×10^{-5}

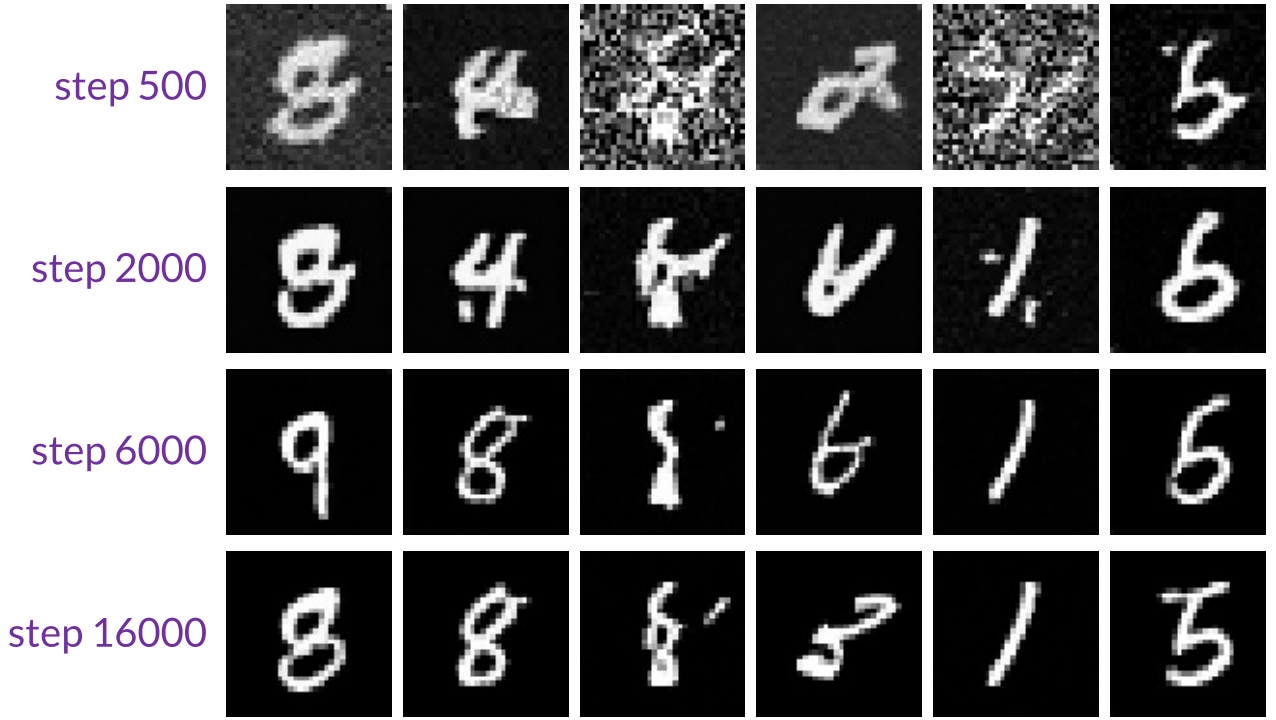
Small t: the noise is a tiny part of x_t, hard to separate from the digit

Large t: x_t is almost all noise, so $\epsilon \approx x_t$

One loss value mixes very different tasks

The loss curve and the samples





Same starting noise for every checkpoint (DDIM, 100 steps)

The loss flattens early; samples keep improving

Averaging the weights (EMA)

$$\theta_{\text{EMA}} \leftarrow d\,\theta_{\text{EMA}} + \underbrace{(1-d)\,\theta}_{\text{current weights}}$$

Updated after every optimizer step

Samples use θ_{EMA}

$d = 0.999$ here; DDPM uses 0.9999

Step 2000: worse, still averaging early weights (test loss 0.026 vs 0.020)

Step 16000: slightly better (0.0157 vs 0.0162)

Not the learning-rate or the noise schedule

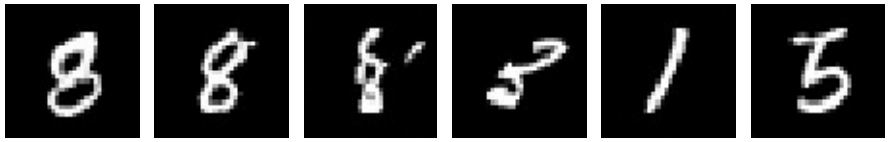
step 2000, raw



step 2000, EMA



step 16000, raw



step 16000, EMA



Lecture outline

01 Why generate in small steps

02 Turning data into noise

03 Learning to remove noise

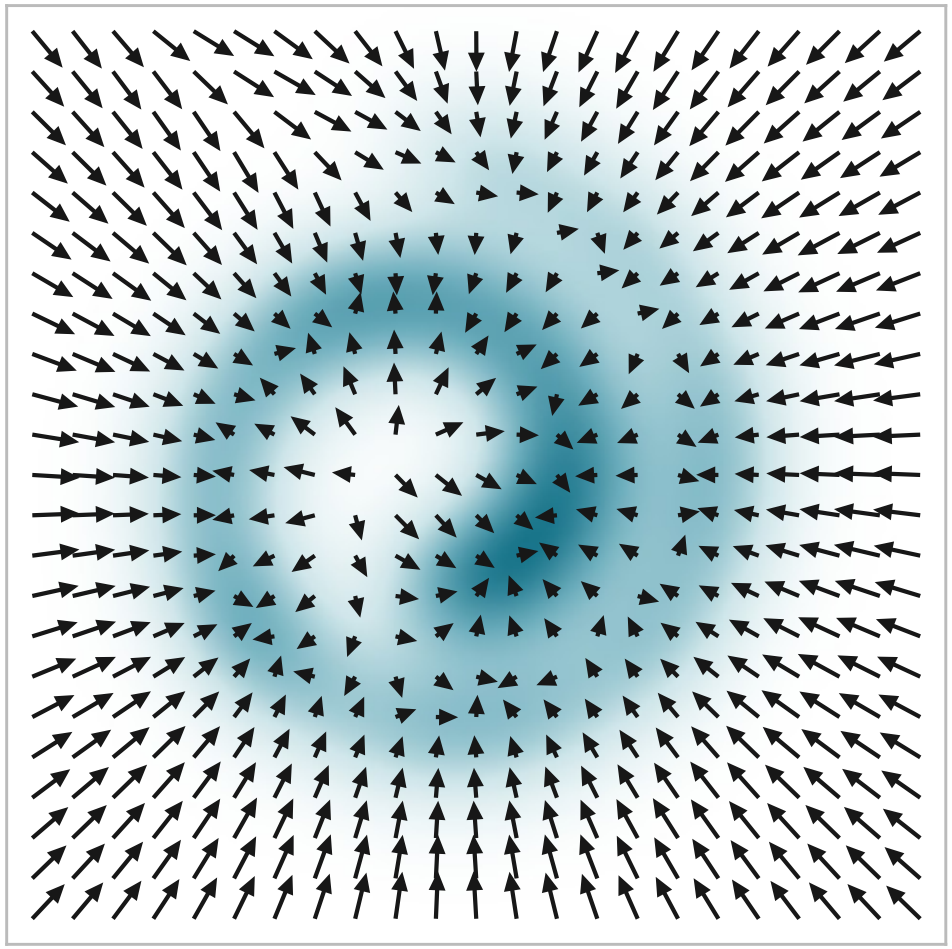
04 Where the loss comes from

05 The denoising network

06 Scores and faster sampling

07 Results and comparison

The score: which way is more likely?



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

Arrows: the direction in which $\log q_{100}$ increases fastest

A gradient with respect to the sample x, not the parameters θ

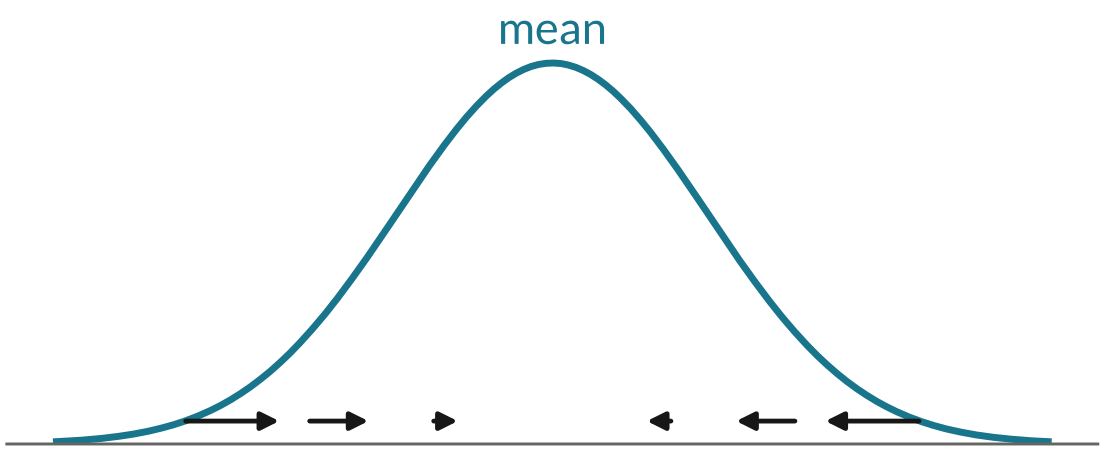
Scaling q by a constant does not change it: no normalization needed

Follow the arrows to reach likely points

For one training image, the noise is the score

Given x_0, the noisy image is Gaussian: $q(x_t \mid x_0) = \mathcal{N}\left(\sqrt{\bar{\alpha}_t}\, x_0,\ (1 - \bar{\alpha}_t) I\right)$

$$\nabla_{x_t} \log q(x_t \mid x_0) = \underbrace{-\frac{x_t - \sqrt{\bar{\alpha}_t}\, x_0}{1 - \bar{\alpha}_t}}_{\text{points back to the center}} = \underbrace{-\frac{\epsilon}{\sqrt{1 - \bar{\alpha}_t}}}_{\text{the noise, rescaled}}$$



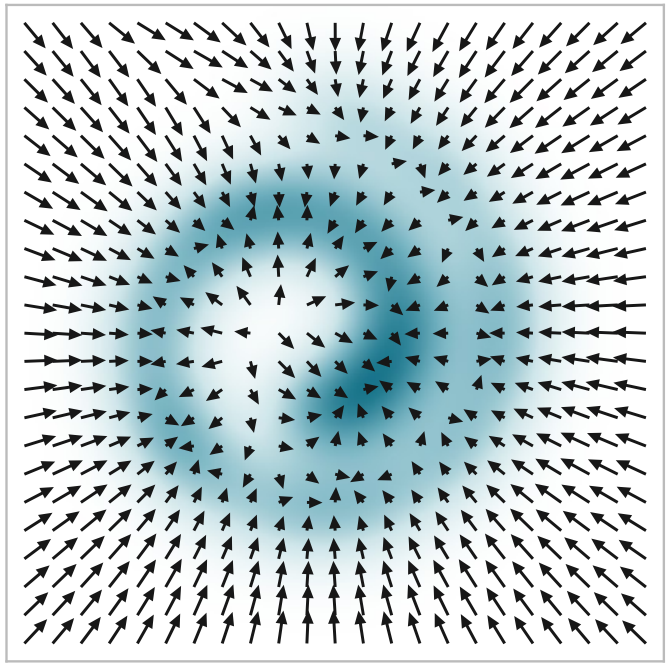
Gaussian score: toward the mean, stronger farther away

Here the mean is the shrunk clean image

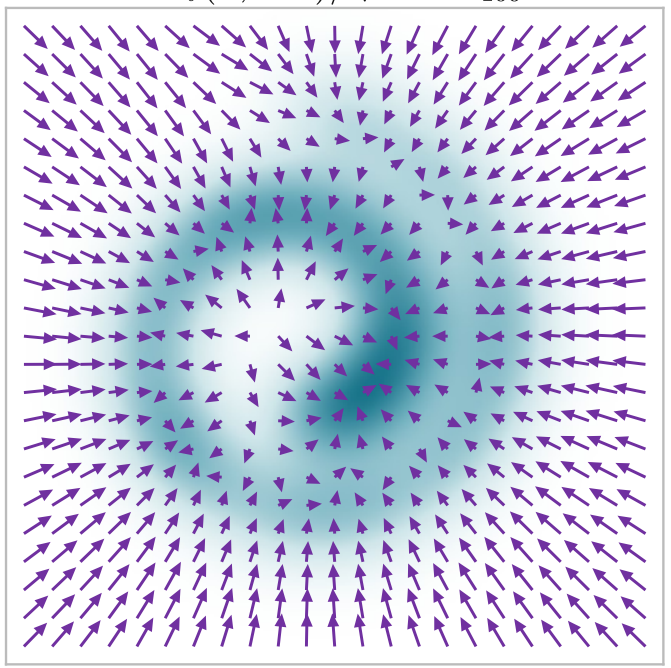
Noise prediction estimates the score

$$-\frac{\epsilon_\theta(x_t, t)}{\sqrt{1-\bar{\alpha}_t}} \approx \nabla_{x_t} \log q_t(x_t)$$

exact score of q_{100}



$-\epsilon_\theta(x, 100)/\sqrt{1-\bar{\alpha}_{100}}$

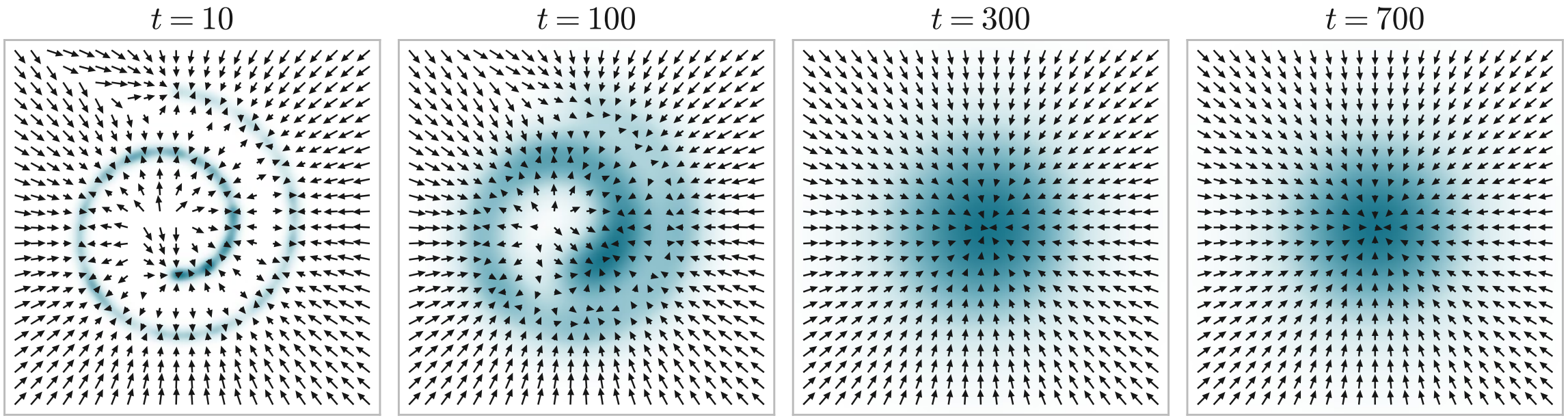


Regression returns the average noise over all x_0 that fit x_t

Averaging per-image scores gives the score of q_t

Spiral, $t = 100$: learned and exact arrows agree (median cosine 0.9997)

Scores at different noise levels



Low noise: sharp arrows onto the spiral, but little data far away to learn from

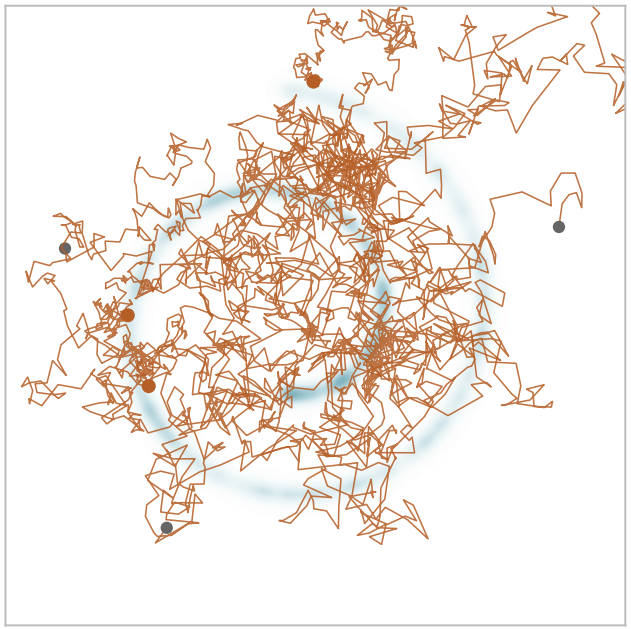
High noise: smooth arrows toward the center, reliable everywhere

Sampling uses the smooth scores first, then sharper ones

Sampling follows the score, plus noise

The DDPM step, rewritten with $s_\theta = -\epsilon_\theta / \sqrt{1-\bar{\alpha}_t}$:

$$x_{t-1} = \frac{1}{\sqrt{\alpha_t}} \left(x_t + \underbrace{\beta_t \, s_\theta(x_t, t)}_{\text{a small step uphill}} \right) + \underbrace{\sigma_t z}_{\text{fresh noise}}$$

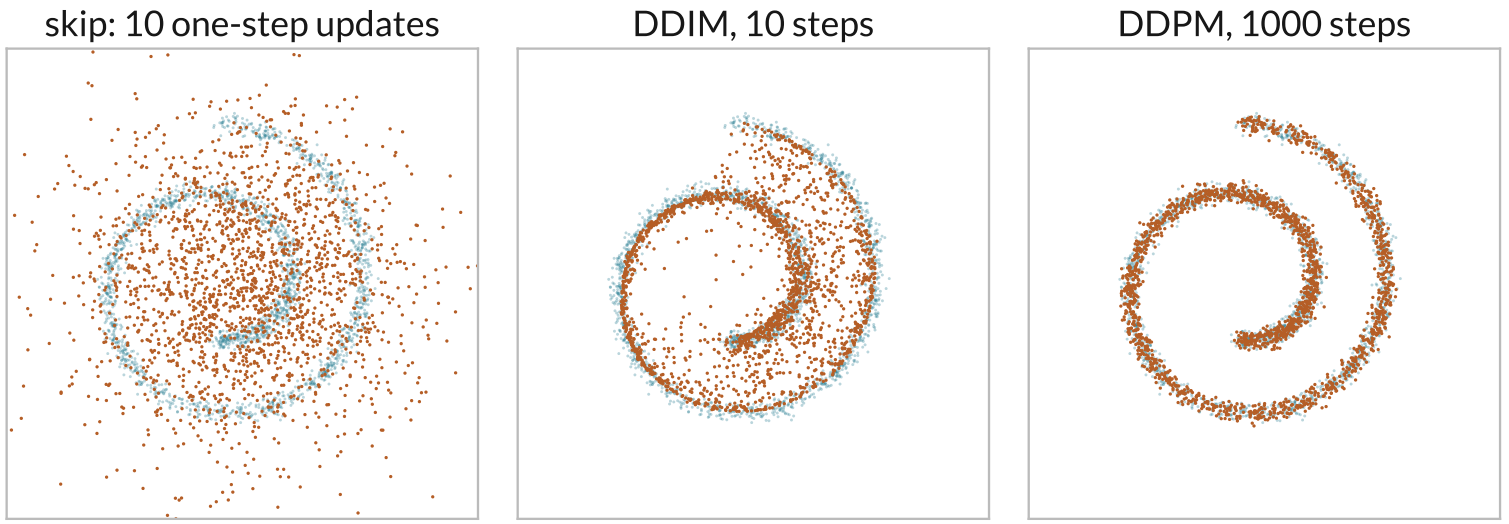


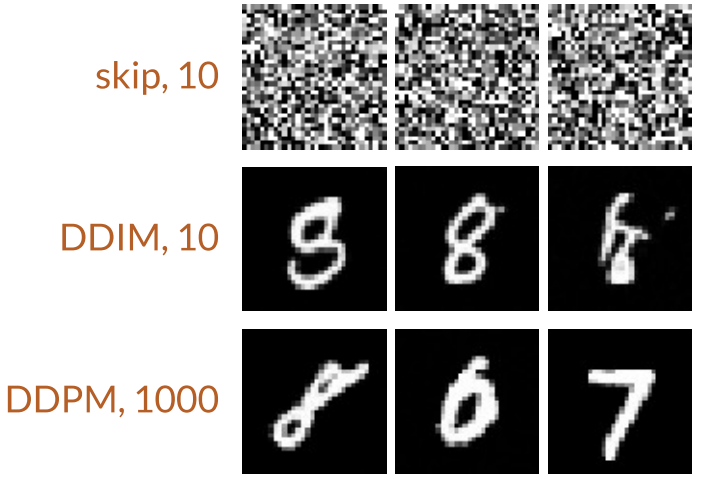
Three DDPM paths on the spiral: noisy, but drifting onto the data

Score: where to go. Noise: keeps samples spread out (slide 31)

The update is exact algebra, not an approximation

Can we simply skip steps?





One-step update applied at only 10 of the 1000 times: most noise remains

Its coefficients assume one small step of variance β_t

We need an update designed for large jumps

DDIM: estimate, then re-noise

From level t to any lower level $s < t$, reusing the network's two outputs:

$$x_s = \underbrace{\sqrt{\bar{\alpha}_s}\ \hat{x}_0}_{\text{clean estimate, at level } s} + \underbrace{\sqrt{1-\bar{\alpha}_s}\ \epsilon_\theta(x_t, t)}_{\text{same predicted noise, at level } s}$$

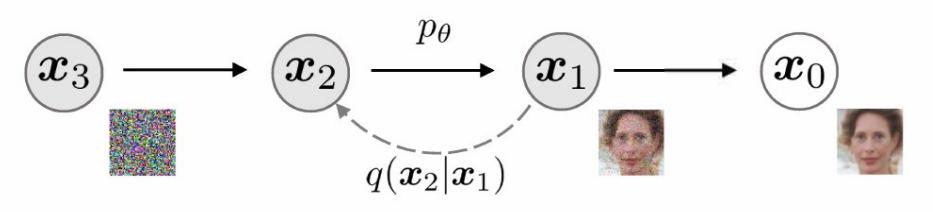
$\hat{x}_0 = \big(x_t - \sqrt{1-\bar{\alpha}_t}\,\epsilon_\theta(x_t, t)\big)/\sqrt{\bar{\alpha}_t}$, as on slide 25

The jump formula, used to go back down to a lower noise level

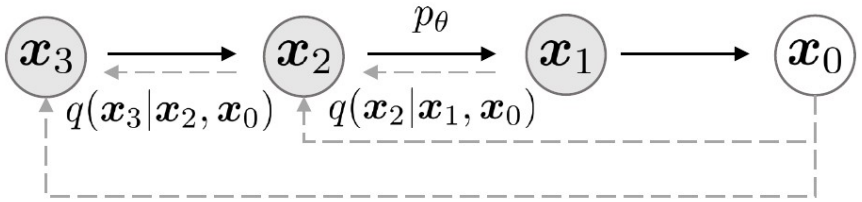
No fresh noise: the update is deterministic

Large jumps are allowed: s can be far below t

Why the trained model still applies



DDPM's noising chain



DDIM's process

L_{simple} only uses $q(x_t \mid x_0)$, one t at a time

DDIM defines a different process with the same $q(x_t \mid x_0)$

So the trained ϵ_θ serves both samplers: no retraining

Source: DDIM (2021)

DDIM sampling

Sampling (DDIM, deterministic)

choose S times $T = \tau_1 > \cdots > \tau_S = 0$, e.g. $S = 50$

$x \sim \mathcal{N}(0, I)$

for each consecutive pair (t, s):

$\hat{\epsilon} = \epsilon_\theta(x, t)$

$\hat{x}_0 = \left(x - \sqrt{1 - \bar{\alpha}_t}\,\hat{\epsilon}\right) / \sqrt{\bar{\alpha}_t}$

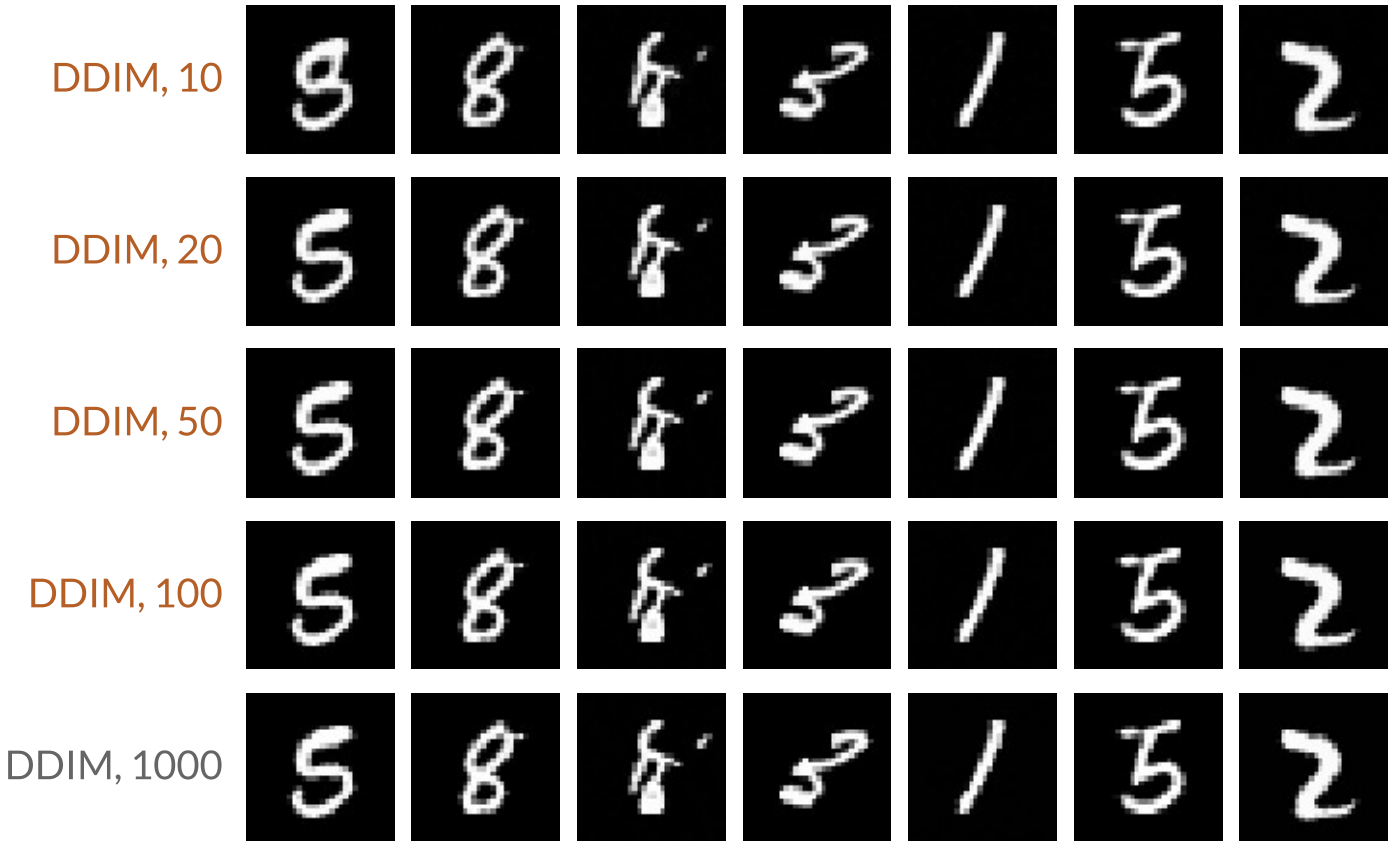
$x = \sqrt{\bar{\alpha}_s}\,\hat{x}_0 + \sqrt{1 - \bar{\alpha}_s}\,\hat{\epsilon}$

return x

- S network calls, not T
- Same trained model as DDPM
- Evenly spaced times work well

Source: DDIM (2021)

Fewer steps on our model



Same starting noise in every row

Bottom row: 1000 DDIM steps, the reference

20 and 50 steps: nearly the same images

10 steps: still clean; a digit can change shape (first column)

20–50× fewer network calls

How much fresh noise?

$$x_s = \sqrt{\bar{\alpha}_s}\,\hat{x}_0 + \sqrt{1-\bar{\alpha}_s-\sigma^2}\,\hat{\epsilon} \;+\; \sigma z\;, \qquad \sigma = \eta\sqrt{\tfrac{1-\bar{\alpha}_s}{1-\bar{\alpha}_t}}\sqrt{1-\tfrac{\bar{\alpha}_t}{\bar{\alpha}_s}}$$

$\eta = 0$: DDIM. $\eta = 1$: the DDPM step for the chosen times

	S	CIFAR10 (32×32) 10	20	50	100	1000	CelebA (64×64) 10	20	50	100	1000
η	0.0	**13.36**	**6.84**	**4.67**	**4.16**	4.04	**17.33**	**13.73**	**9.17**	**6.53**	3.51
	0.2	14.04	7.11	4.77	4.25	4.09	17.66	14.11	9.51	6.79	3.64
	0.5	16.66	8.35	5.25	4.46	4.29	19.86	16.06	11.01	8.09	4.28
	1.0	41.07	18.36	8.01	5.78	4.73	33.12	26.03	18.48	13.93	5.98
	$\hat{\sigma}$	367.43	133.37	32.72	9.99	**3.17**	299.71	183.83	71.71	45.20	**3.26**

CIFAR-10 FID with S steps (lower is better): 13.4 vs 41.1 at 10 steps

$\hat{\sigma}$ row: DDPM's larger variance choice, poor with few steps

Source: DDIM (2021)

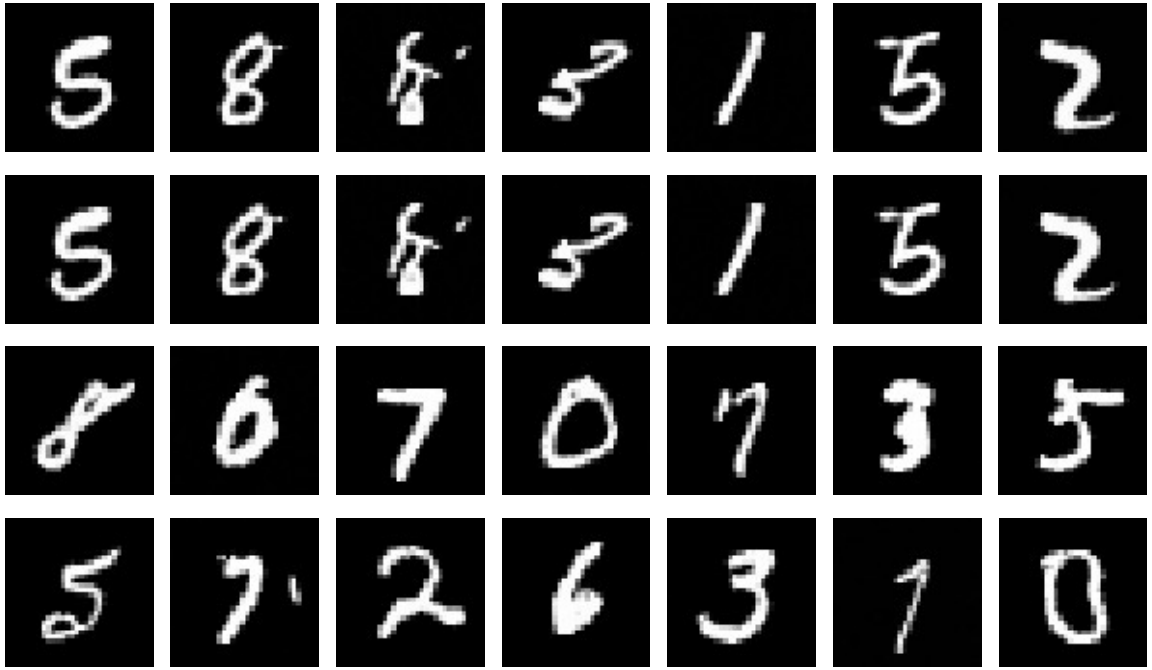
Same start, same image

DDIM, run 1

DDIM, run 2

DDPM, run 1

DDPM, run 2



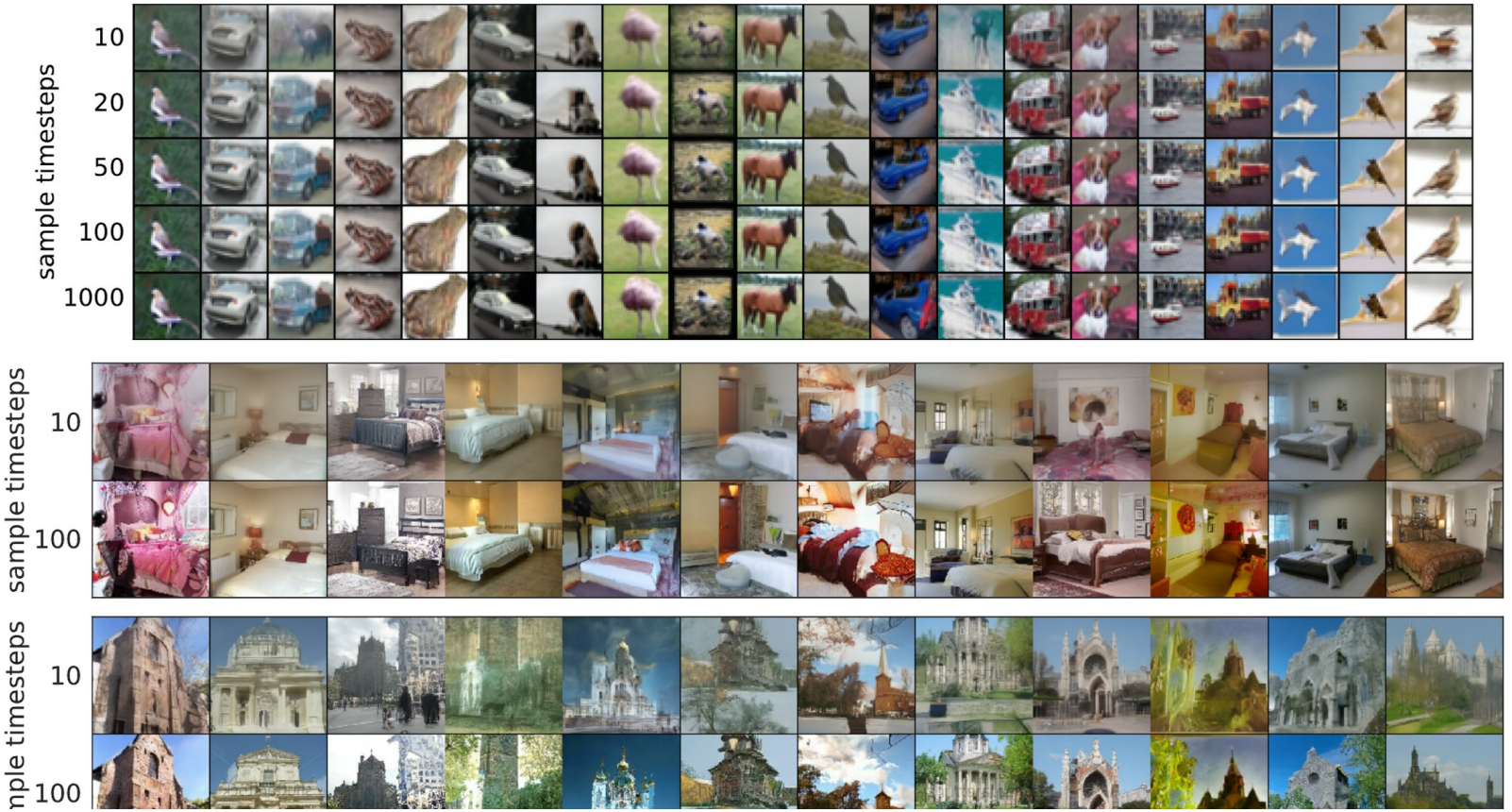
Two runs from the same x_T

DDIM: identical images

DDPM: different images, due to fresh noise

With DDIM, x_T decides the image

Same start, different step counts



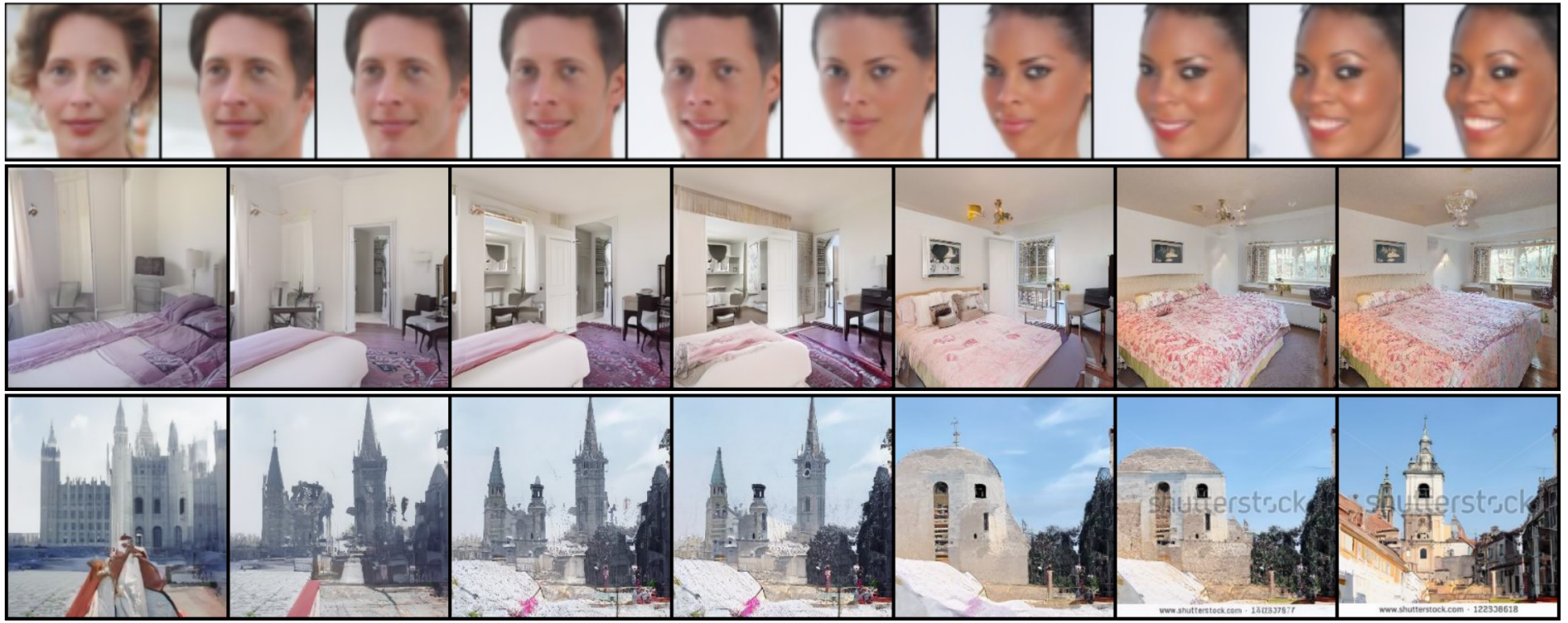
Rows: 10 to 1000 DDIM steps from one x_T

Content and layout are already fixed with 10 steps

More steps add detail

Source: DDIM (2021)

Interpolating the starting noise



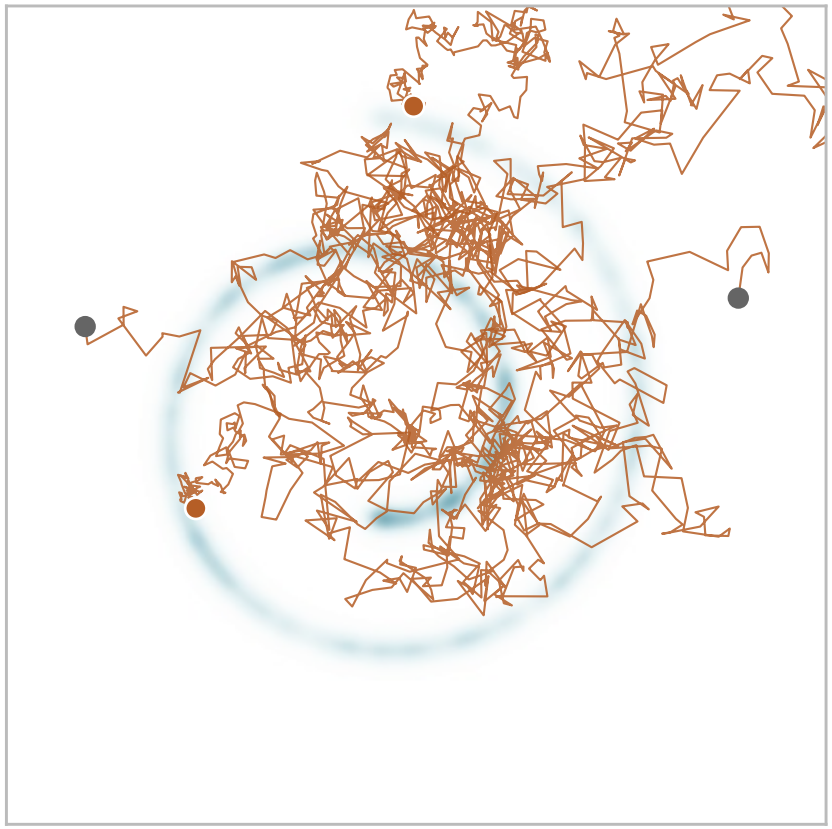
Blend two starting noises gradually; generate each with DDIM (50 steps)

The images change smoothly: x_T acts like a latent code

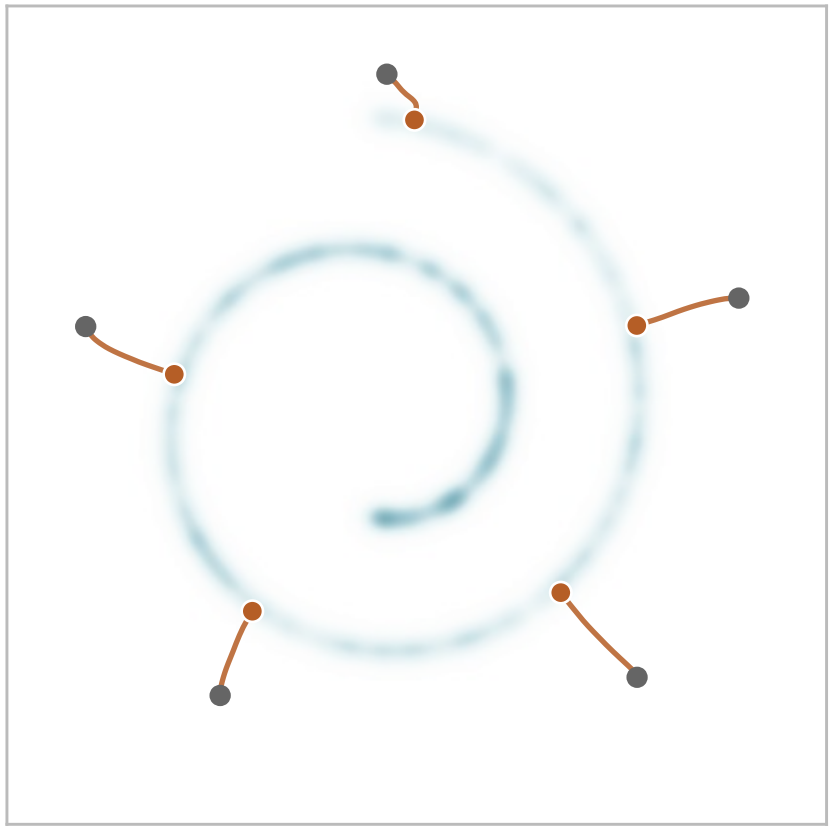
Source: DDIM (2021)

Deterministic paths in 2-D

DDPM: 1000 noisy steps



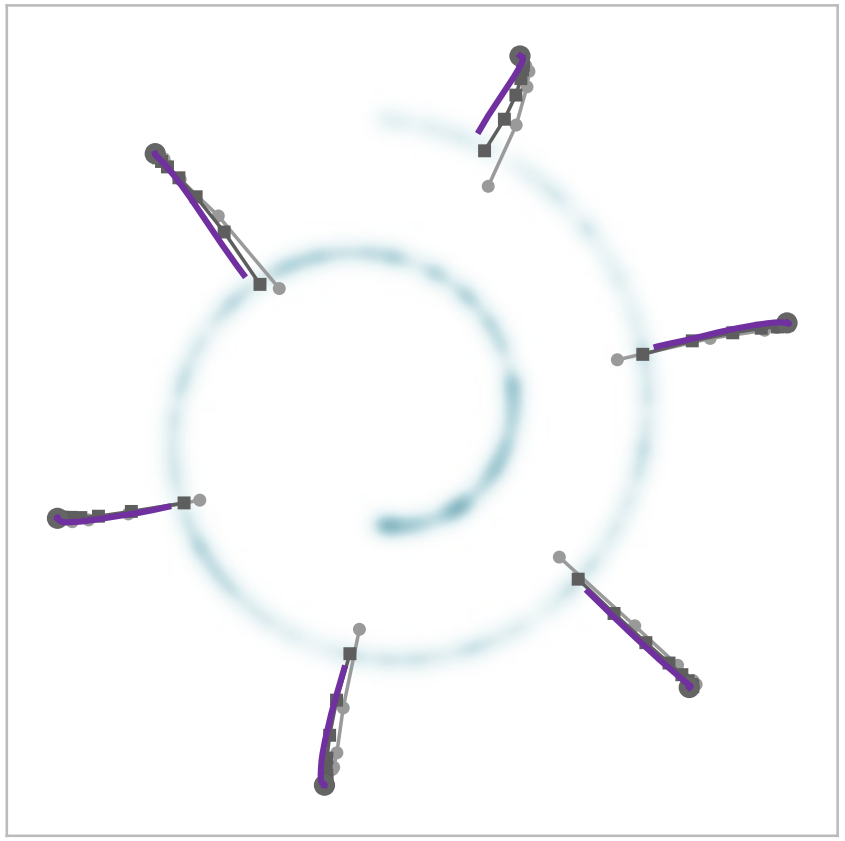
DDIM: 50 deterministic steps

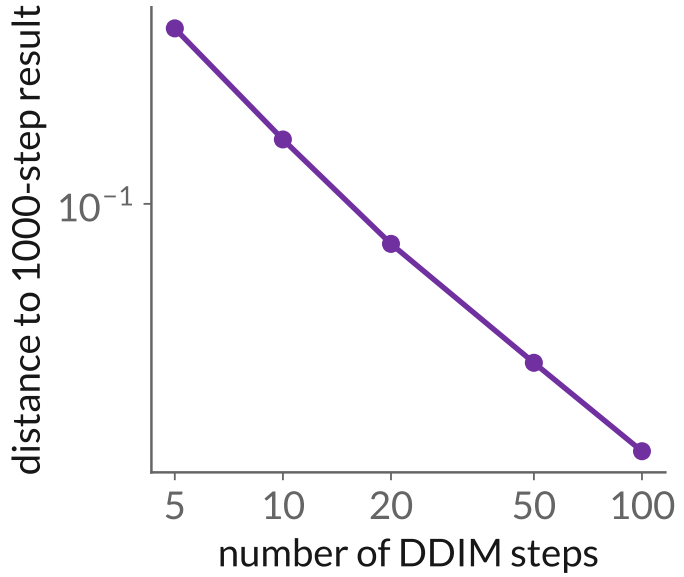


Same starting points (gray), same trained model

DDPM: a random walk that drifts onto the data. DDIM: smooth, nearly straight paths

DDIM as an ODE solver





With $\bar{x} = x/\sqrt{\bar{\alpha}}$ and $\sigma = \sqrt{1-\bar{\alpha}}/\sqrt{\bar{\alpha}}$:

$$\bar{x}_s = \bar{x}_t + (\sigma_s - \sigma_t)\,\epsilon_\theta$$

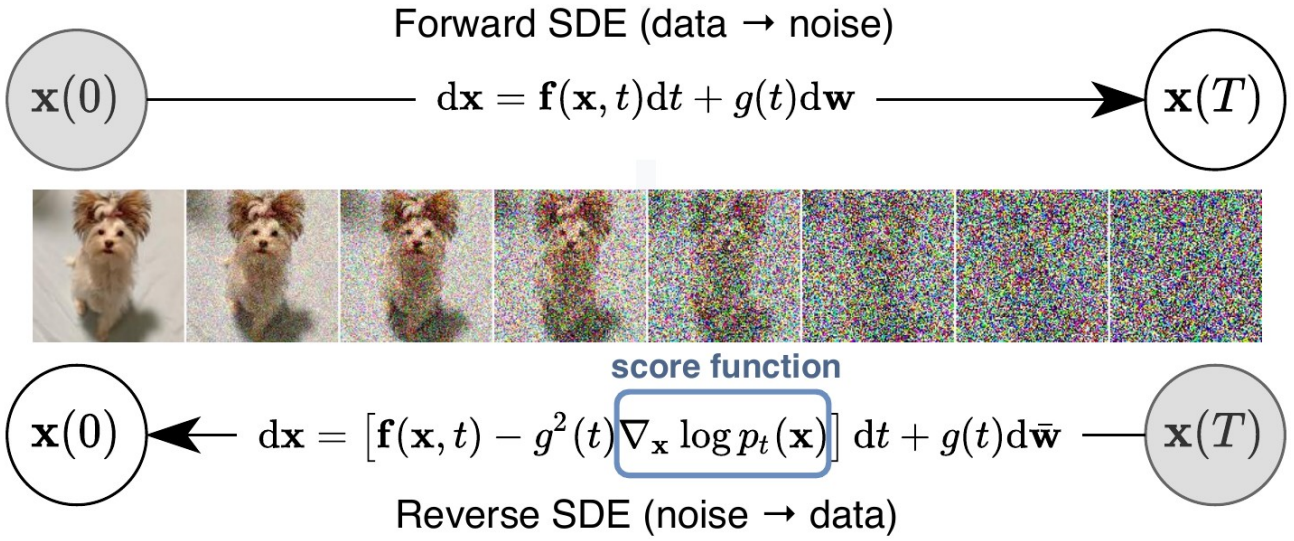
One Euler step of $\frac{d\bar{x}}{d\sigma} = \epsilon_\theta$

Gray: 5 and 10 steps overshoot.
Purple: 1000

Error halves when steps double

Better ODE solvers (e.g. DPM-Solver) reach good samples in about 10–20 calls

Continuous time



Infinitely many tiny steps: noise is added continuously

Reverse in time with the score: continuous DDPM

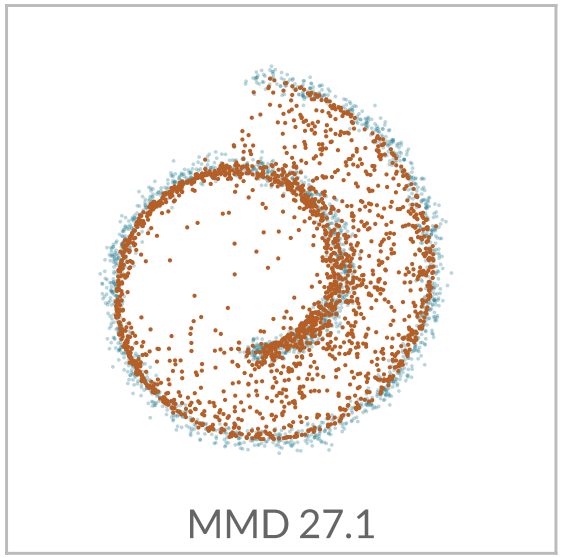
A deterministic ODE with the same q_t: continuous DDIM

L18: flow matching learns such a path directly

Source: Score-based SDEs (2021)

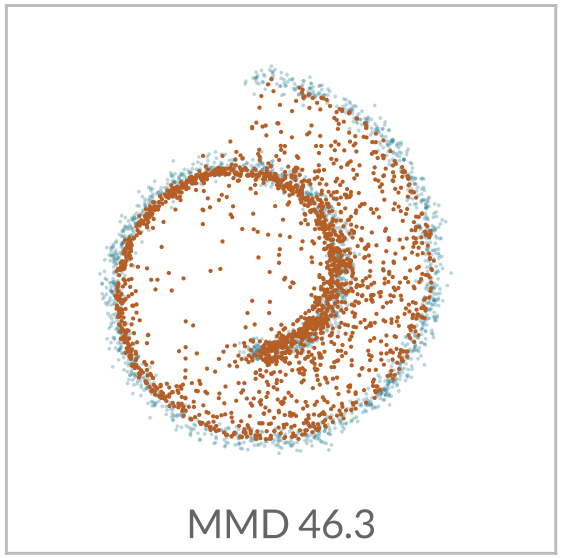
Stochastic or deterministic?

$\eta = 0$, 10 steps



MMD 27.1

$\eta = 1$, 10 steps



MMD 46.3

$\eta = 0$, 50 steps



MMD 6.7

$\eta = 1$, 50 steps



MMD 7.8

MMD to the data ($\times 10^{-3}$, lower is better; data vs. data: 6.2)

10 steps: deterministic is closer. 50 steps: both close

Fresh noise can correct earlier errors, but needs more steps to average out

Lecture outline

01 Why generate in small steps

02 Turning data into noise

03 Learning to remove noise

04 Where the loss comes from

05 The denoising network

06 Scores and faster sampling

07 Results and comparison

DDPM results (2020)

Model	IS	FID	NLL Test (Train)
Conditional			
EBM [11]	8.30	37.9	
JEM [17]	8.76	38.4	
BigGAN [3]	9.22	14.73	
StyleGAN2 + ADA (v1) [29]	**10.06**	**2.67**	
Unconditional			
Diffusion (original) [53]			≤ 5.40
Gated PixelCNN [59]	4.60	65.93	3.03 (2.90)
Sparse Transformer [7]			**2.80**
PixelIQN [43]	5.29	49.46	
EBM [11]	6.78	38.2	
NCSNv2 [56]		31.75	
NCSN [55]	8.87 ± 0.12	25.32	
SNGAN [39]	8.22 ± 0.05	21.7	
SNGAN-DDLS [4]	9.09 ± 0.10	15.42	
StyleGAN2 + ADA (v1) [29]	$\mathbf{9.74} \pm 0.05$	3.26	
Ours (L, fixed isotropic $\mathbf{\Sigma}$)	7.67 ± 0.13	13.51	≤ 3.70 (3.69)
Ours (L_{simple})	9.46 ± 0.11	**3.17**	≤ 3.75 (3.72)



CIFAR-10: FID 3.17, the best unconditional result in the table (lower is better, L16)

LSUN churches and bedrooms, 256×256 : close to ProgressiveGAN

Left table: FID and likelihood (NLL, bits per dimension)

Source: DDPM (2020)

Quality and coverage against GANs

Model	FID	sFID	Prec	Rec
LSUN Bedrooms 256×256				
DCTransformer† [42]	6.40	6.66	0.44	**0.56**
DDPM [25]	4.89	9.07	0.60	0.45
IDDPM [43]	4.24	8.21	0.62	0.46
StyleGAN [27]	2.35	6.62	0.59	0.48
ADM (dropout)	**1.90**	**5.59**	**0.66**	0.51
LSUN Horses 256×256				
StyleGAN2 [28]	3.84	6.46	0.63	0.48
ADM	2.95	**5.94**	0.69	**0.55**
ADM (dropout)	**2.57**	6.81	**0.71**	**0.55**
LSUN Cats 256×256				
DDPM [25]	17.1	12.4	0.53	0.48
StyleGAN2 [28]	7.25	**6.33**	0.58	0.43
ADM (dropout)	**5.57**	6.69	**0.63**	**0.52**
ImageNet 64×64				
BigGAN-deep* [5]	4.06	3.96	**0.79**	0.48
IDDPM [43]	2.92	**3.79**	0.74	0.62
ADM	2.61	**3.77**	0.73	0.63
ADM (dropout)	**2.07**	4.29	0.74	**0.63**

Precision ≈ quality, recall ≈ coverage (L16)

LSUN bedrooms: ADM FID 1.90 vs StyleGAN 2.35

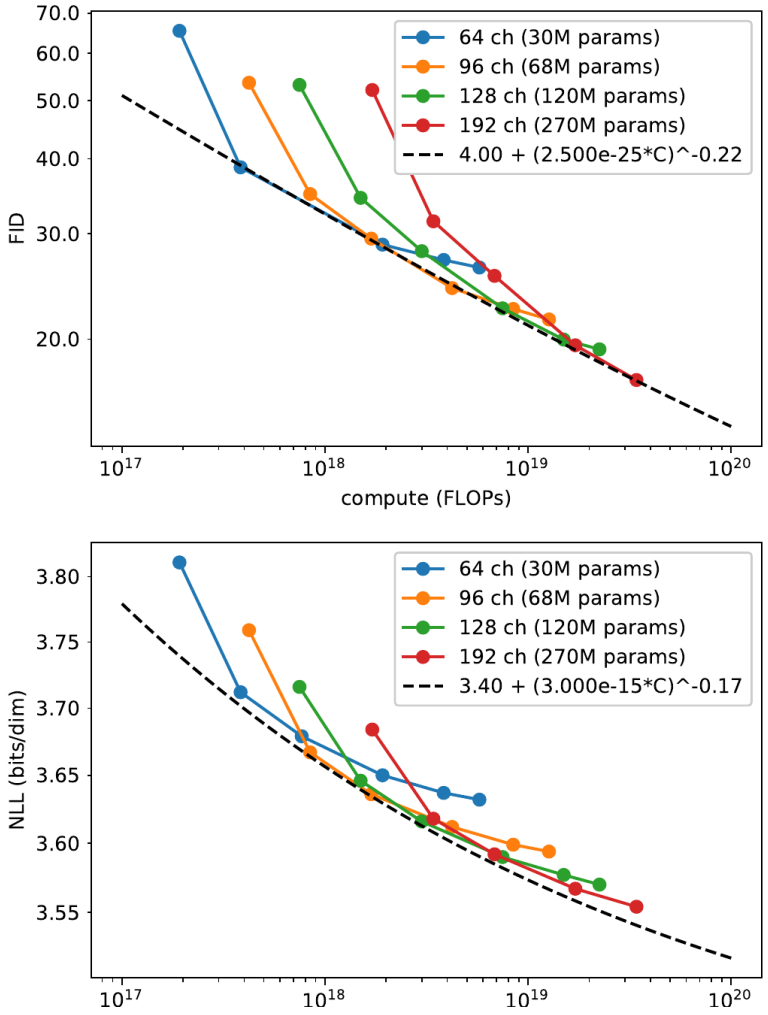
ADM's recall beats the GAN's in all four blocks

Diffusion covers more of the data at similar or better quality

No guidance or labels in these rows (L18 adds them)

Source: Diffusion Models Beat GANs (2021)

Larger models keep improving



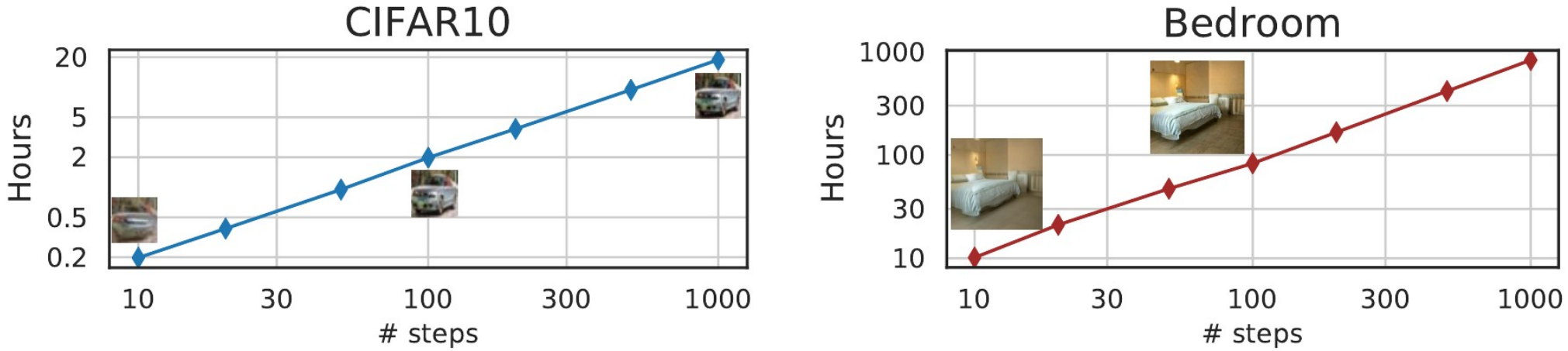
ImageNet 64×64 ; four model widths

FID (top) and likelihood (bottom) against training compute

Quality improves steadily with model size and compute

Source: Improved DDPM (2021)

Quality versus sampling cost



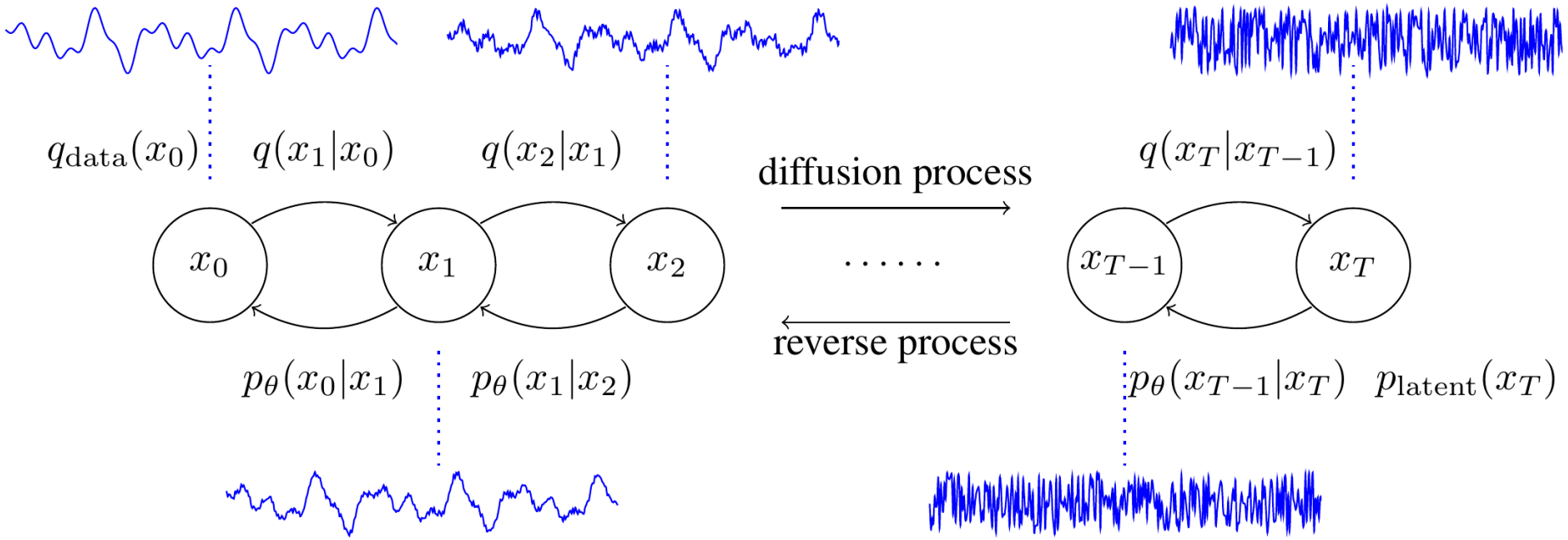
Hours to generate 50,000 images on one GPU: time grows linearly with the number of steps

CIFAR-10: about 20 hours at 1000 steps, 0.2 hours at 10

A GAN needs one call per image; diffusion needs tens to thousands

Source: DDIM (2021)

Beyond images: audio



x_0 : a speech waveform, 22,050 numbers per second

Same noising and denoising; the network is a stack of 1-D dilated convolutions

DiffWave: 2.64M parameters; speech quality on par with WaveNet; a fast version uses 6 steps

Source: DiffWave (2021)

VAE, GAN and diffusion

	VAE (L15)	GAN (L16)	Diffusion
Training	likelihood bound	a two-player game	regression on noise
Samples	often blurry	sharp	sharp
Coverage	good	can collapse	good
Calls per sample	1	1	10–1000

Typical behavior of the standard models; each line has many exceptions

Summary

- Why: one big Gaussian step is too simple; small steps back are nearly Gaussian
- Forward: a fixed chain adds Gaussian noise; any level is reachable in one jump
- Training: predict the added noise with a squared error, a reweighted likelihood bound
- Sampling: many small learned Gaussian steps, with fresh noise
- Network: any same-shape network that is told t, usually a U-Net
- Noise prediction is a scaled score; DDIM reuses it for large deterministic steps
- Results: GAN-level quality, better coverage, slower sampling

Next time: conditional generation and flow matching

Control

Generate what a label or a sentence asks for: conditional noise prediction and guidance

Straighter paths

Learn the velocity of a path from noise to data directly: flow matching

Builds on today: noise prediction, DDIM sampling and the ODE view