Lecture 5

Diffusion Models

6.S978 Deep Generative Models

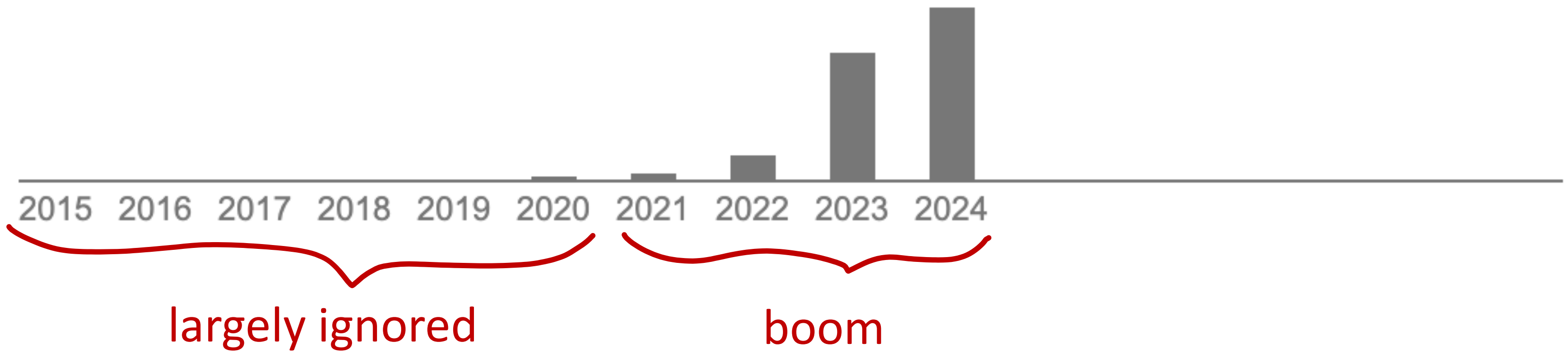
Kaiming He
Fall 2024, EECS, MIT

Overview

- Diffusion Models
- Energy-based Models and Score Matching

Deep unsupervised learning using nonequilibrium thermodynamics

Authors	Jascha Sohl-Dickstein, Eric A Weiss, Niru Maheswaranathan, Surya Ganguli
Publication date	2015/3/12
Journal	International Conference on Machine Learning
Total citations	Cited by 5630



Diffusion Models

Diffusion Models

Forward process

- add noise to data

Reverse process

- learn to denoise

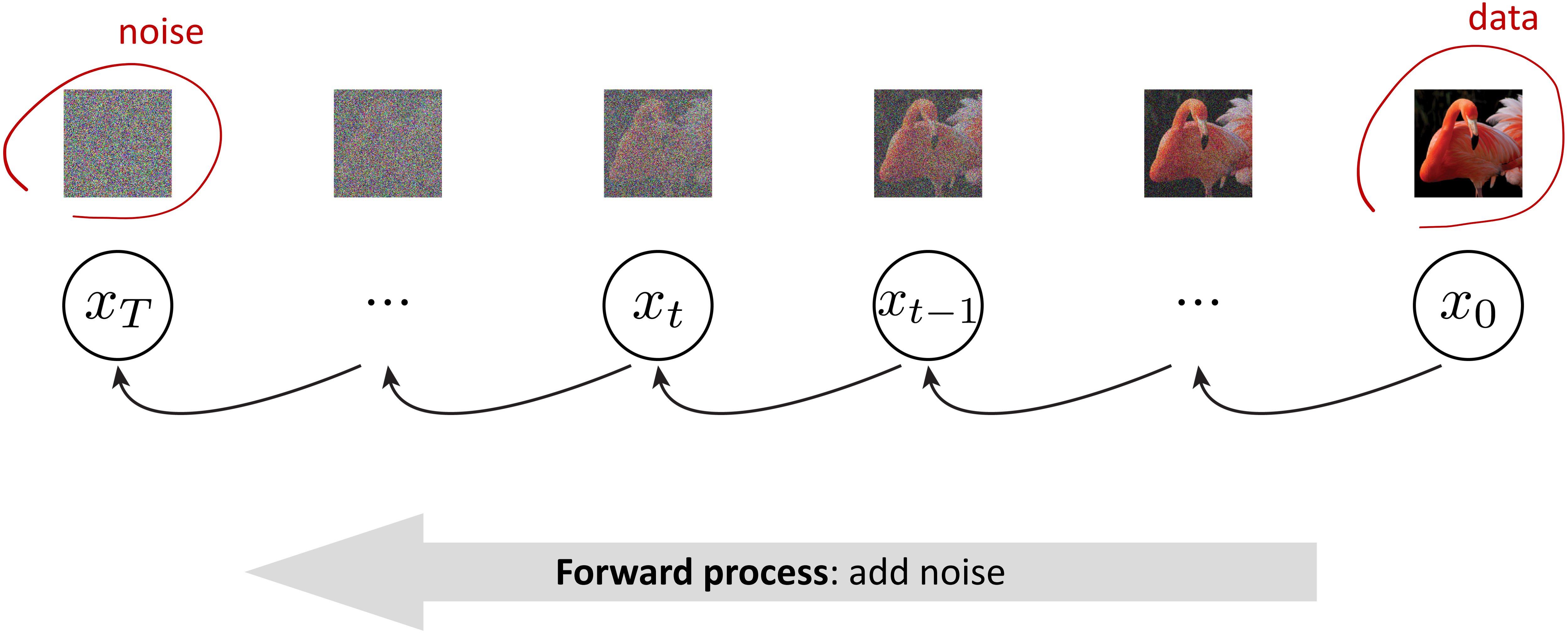
Training objective

- from Hierarchical VAE to L2 loss

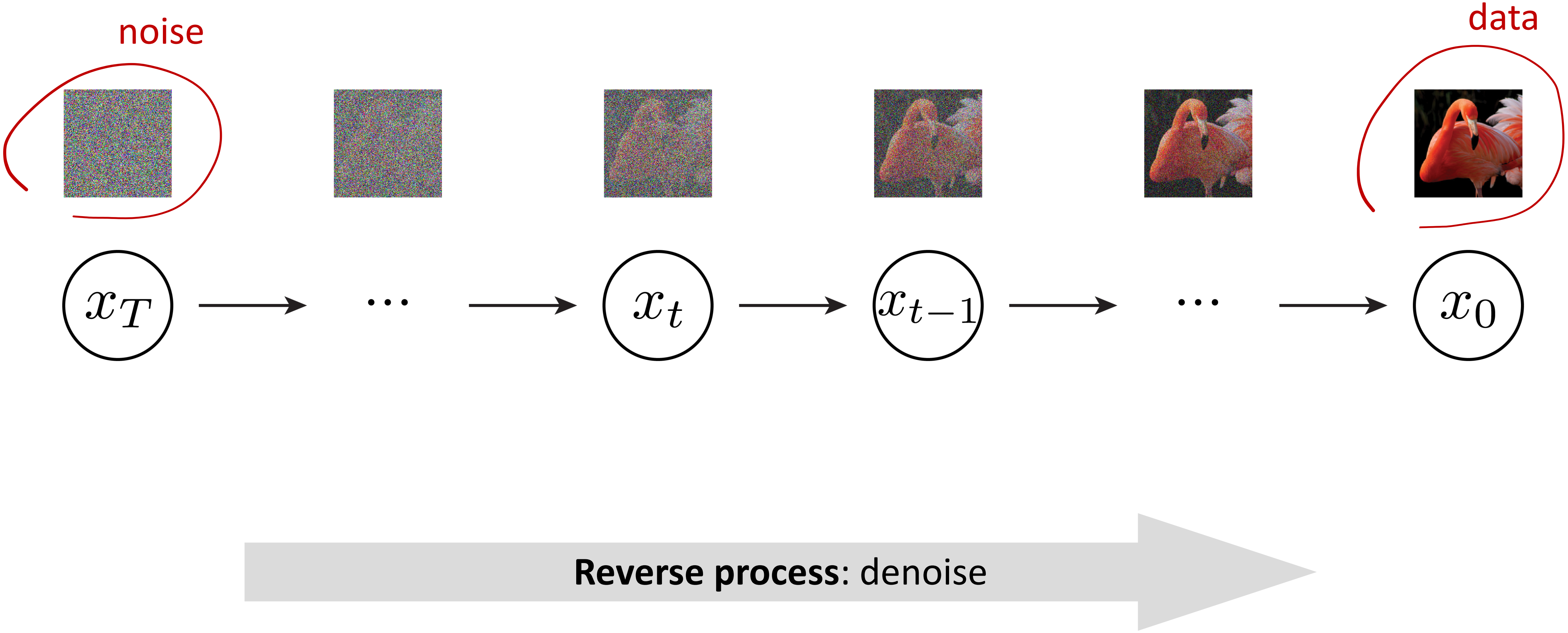
Noise Conditional Network

- represent distributions

… in a nutshell

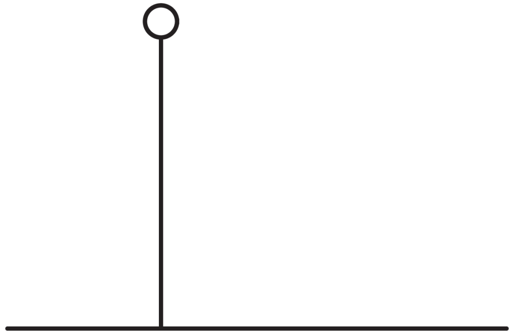


... in a nutshell

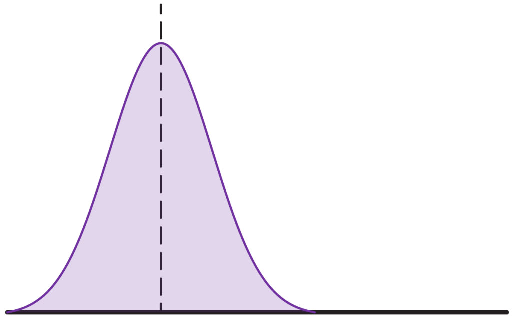


What is noise?

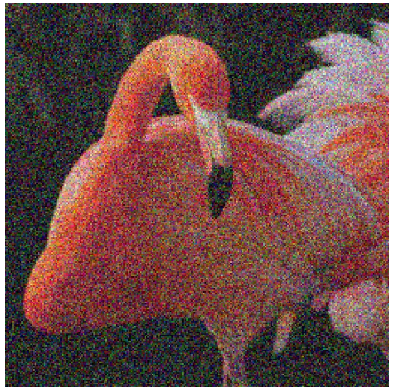
- Adding Gaussian noise ⇔ sampling $x \sim \mathcal{N}(x \mid x_0, \sigma)$



$$p(x) = \delta(x - x_0)$$

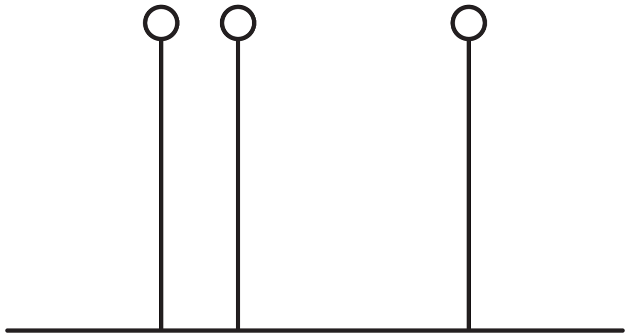


$$p(x) = \mathcal{N}(x \mid x_0, \sigma)$$

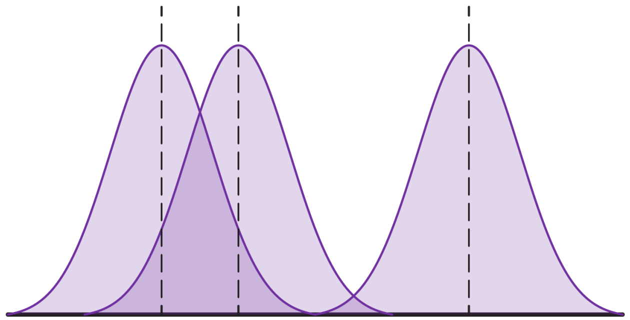


What is noise?

- Adding Gaussian noise ⇔ sampling $x \sim \mathcal{N}(x \mid x_0, \sigma)$



$p_{\text{data}}(x)$

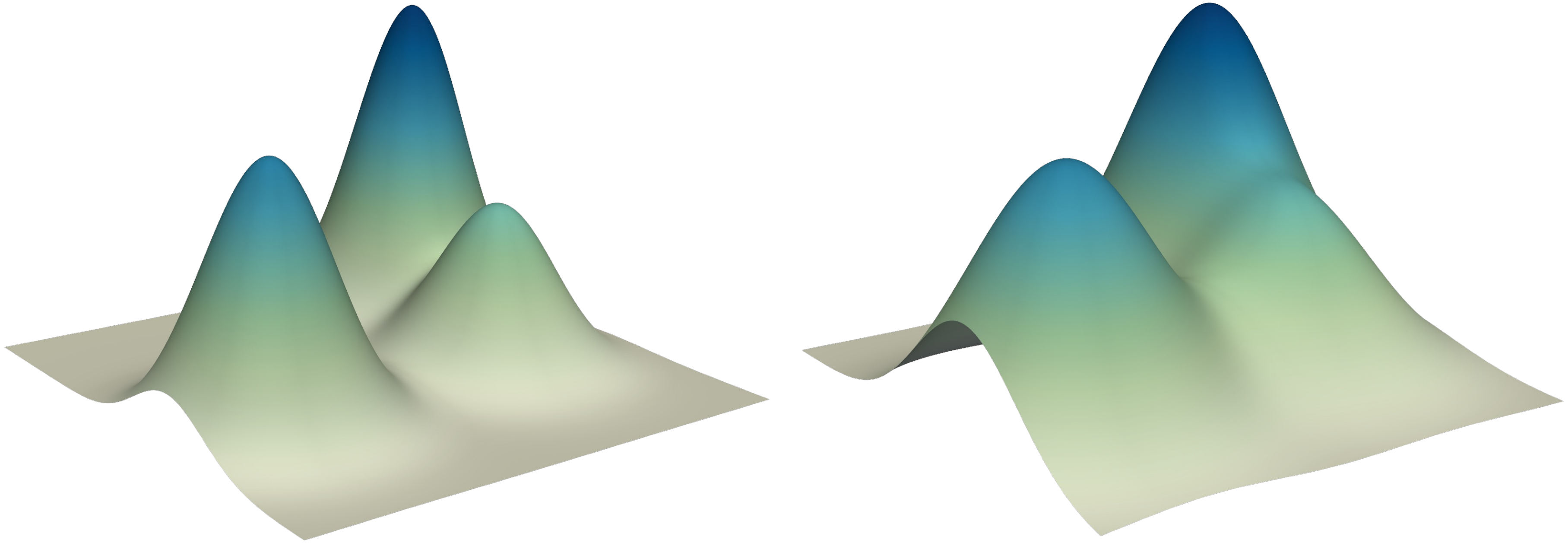


$p_{\text{data}}(x) * \mathcal{N}(x \mid 0, \sigma)$

convolution
(of pdf)

What is noise?

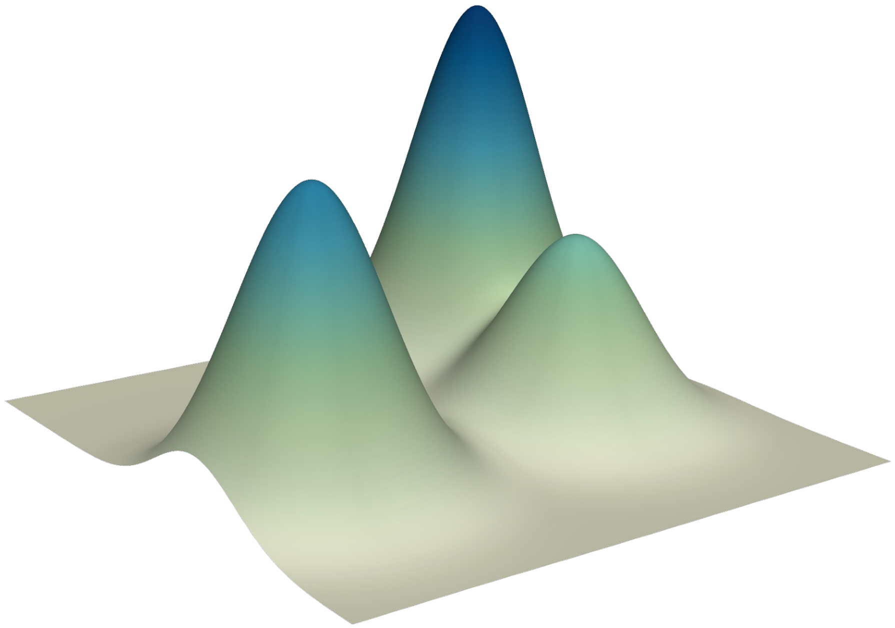
- Adding Gaussian noise ⇔ sampling $x \sim \mathcal{N}(x \mid x_0, \sigma)$



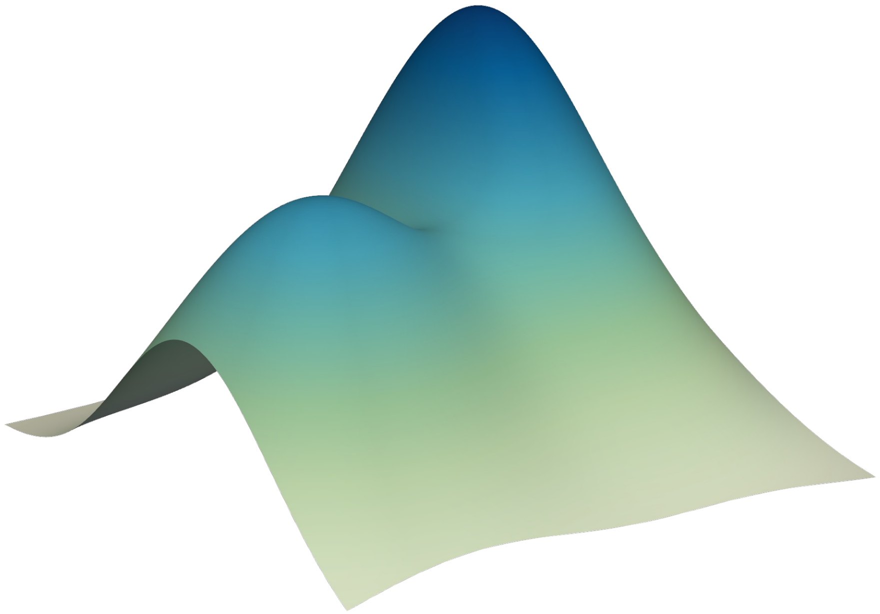
$p_{\text{data}}(x)$ $\quad$ $p_{\text{data}}(x) * \mathcal{N}(x \mid 0, \sigma)$

What is noise?

- Adding Gaussian noise ⇔ sampling $x \sim \mathcal{N}(x \mid x_0, \sigma)$



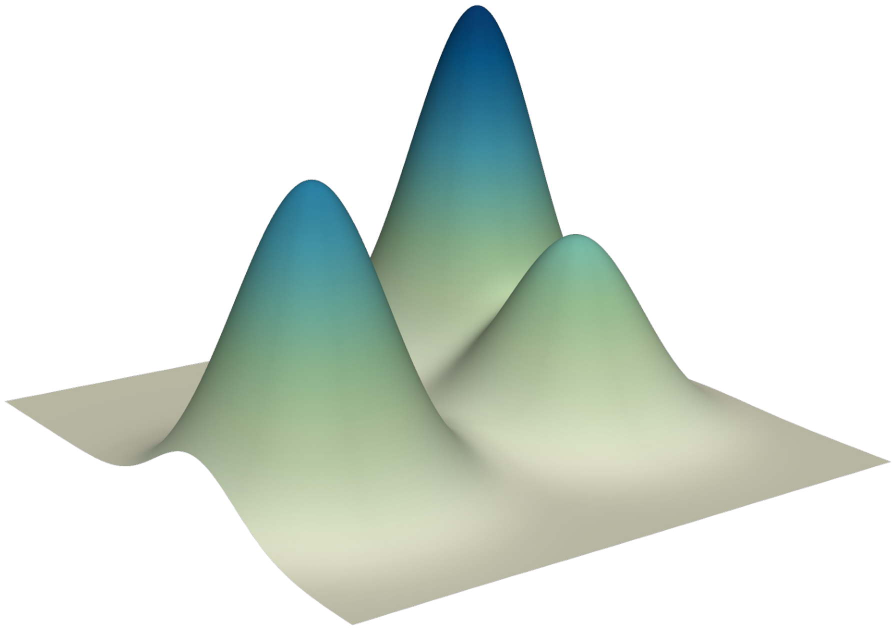
$p_{\text{data}}(x)$



$p_{\text{data}}(x) * \mathcal{N}(x \mid 0, \sigma)$

What is noise?

- Adding Gaussian noise ⇔ sampling $x \sim \mathcal{N}(x \mid x_0, \sigma)$

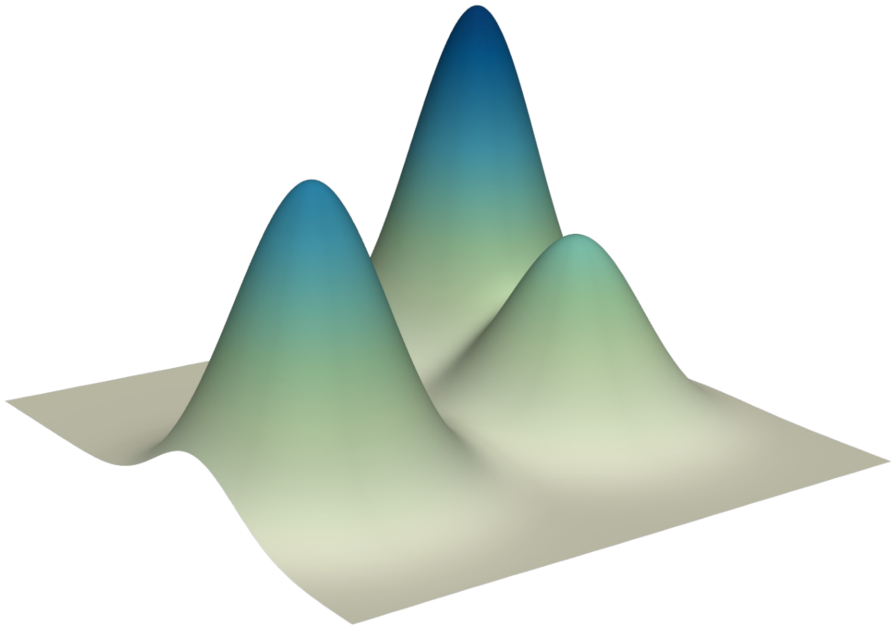


$p_{\text{data}}(x)$

$p_{\text{data}}(x) * \mathcal{N}(x \mid 0, \sigma)$

What is noise?

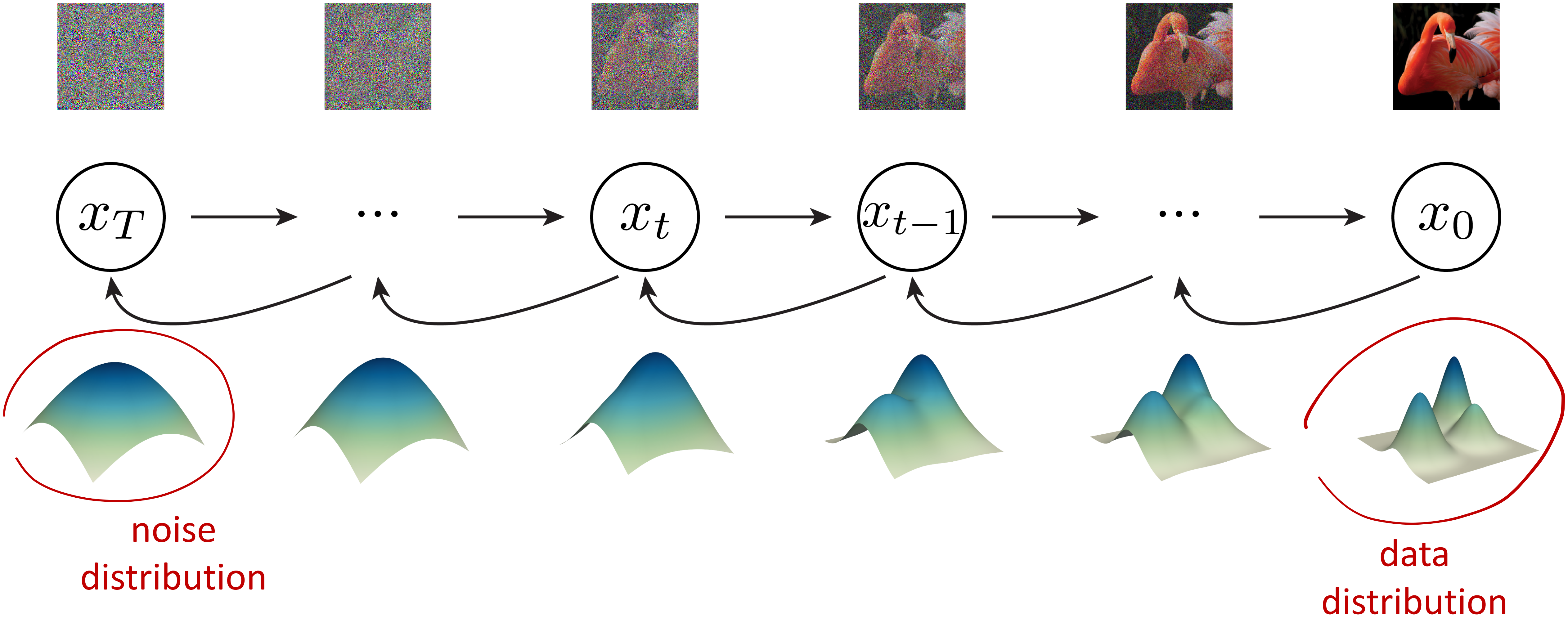
- Adding Gaussian noise ⇔ sampling $x \sim \mathcal{N}(x \mid x_0, \sigma)$



$p_{\text{data}}(x)$

$p_{\text{data}}(x) * \mathcal{N}(x \mid 0, \sigma)$

What is noise?



Diffusion Models

Forward process

- add noise to data

Reverse process

- learn to denoise

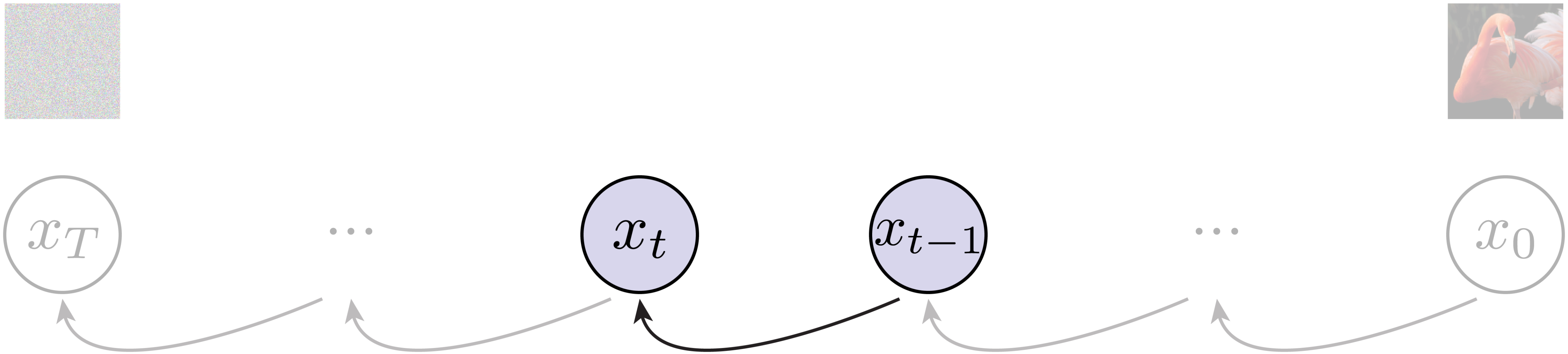
Training objective

- from Hierarchical VAE to L2 loss

Noise Conditional Network

- represent distributions

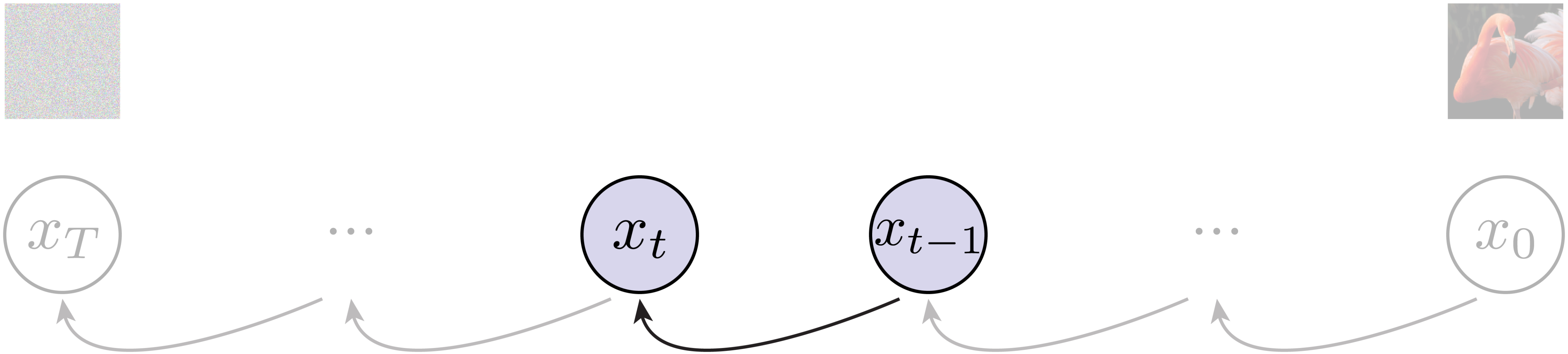
Forward Process



$$x_t = \underline{\sqrt{1-\beta_t}}\, x_{t-1} + \underline{\sqrt{\beta_t}}\,\epsilon, \quad \epsilon \sim \mathcal{N}(0, \mathbf{I})$$

coefficients:
variance preserving

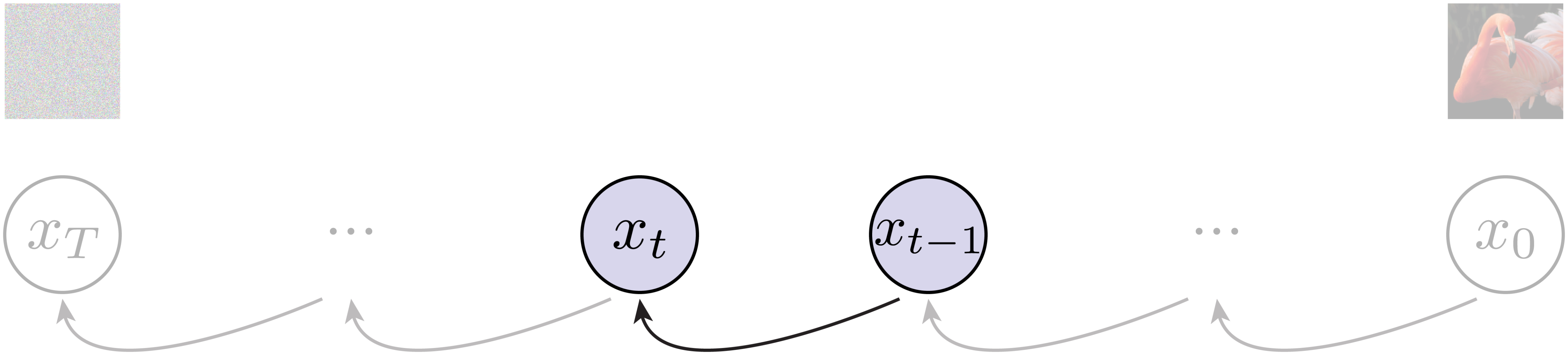
Forward Process



$$x_t = \sqrt{1-\beta_t}\,x_{t-1} + \sqrt{\beta_t}\,\epsilon, \quad \epsilon \sim \mathcal{N}(0, \mathbf{I})$$

t: “schedule”,
key to Diffusion Models’ success

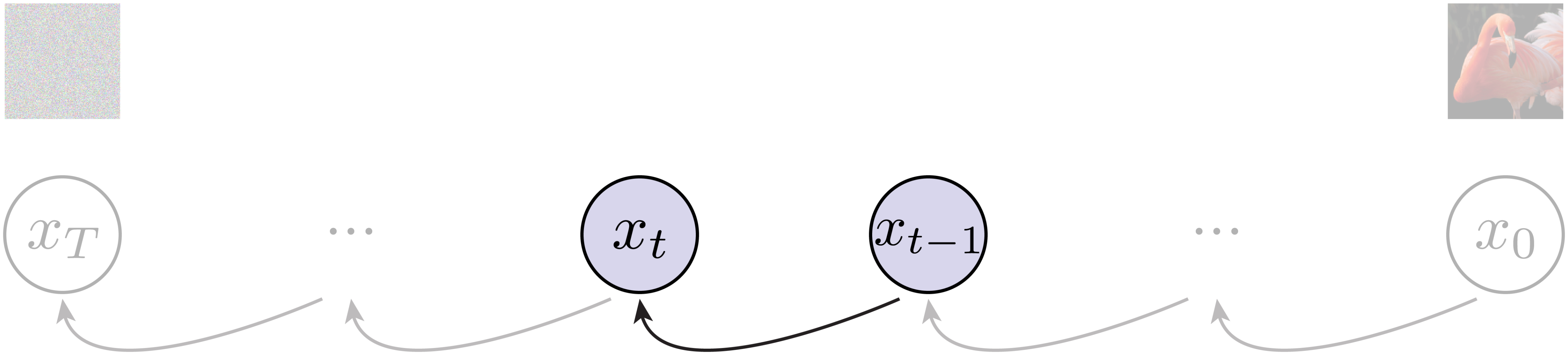
Forward Process



$$x_t = \sqrt{1-\beta_t}x_{t-1} + \sqrt{\beta_t}\epsilon, \quad \epsilon \sim \mathcal{N}(0, \mathbf{I})$$

mean of x_t 　 std of x_t

Forward Process

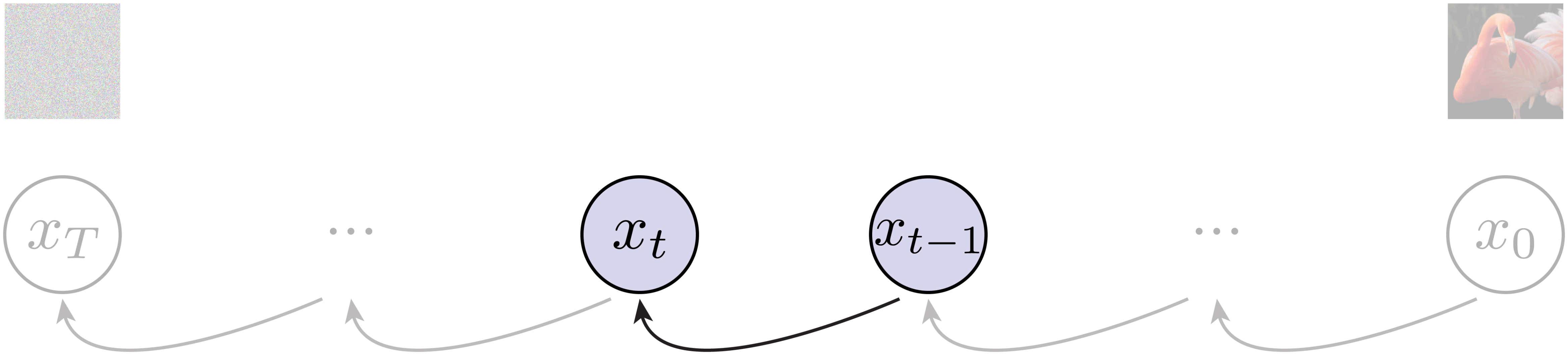


$$q(x_t \mid x_{t-1})$$

$$= \mathcal{N}(x_t \mid \underline{\sqrt{1-\beta_t}x_{t-1}}, \beta_t \mathbf{I})$$

mean of x_t var of x_t

Forward Process

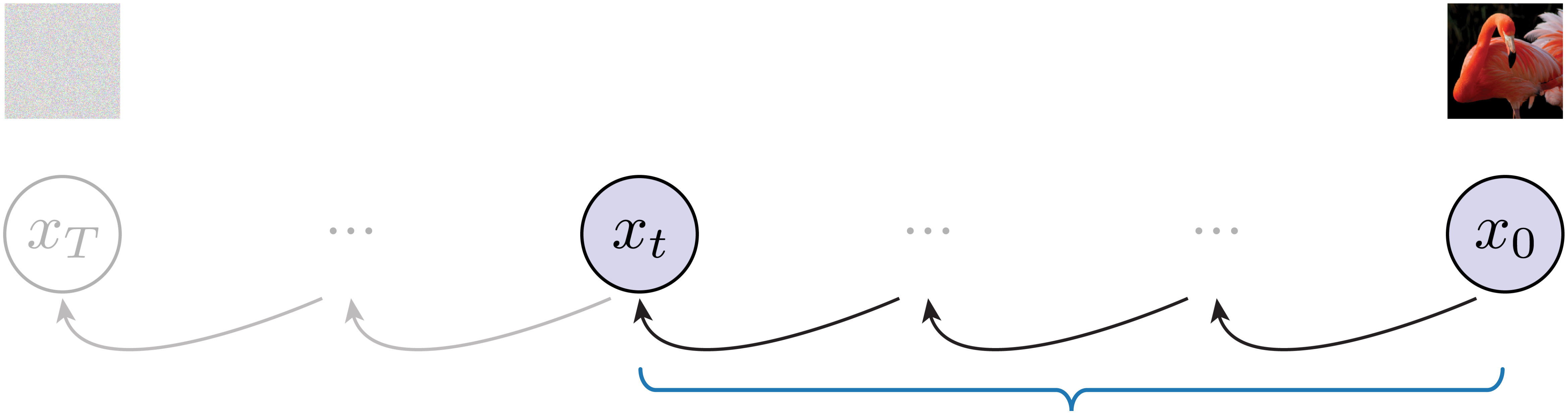


$$q(x_t \mid x_{t-1})$$
$$= \mathcal{N}(x_t \mid \sqrt{1-\beta_t}\,x_{t-1}, \beta_t \mathbf{I})$$

identity matrix

- sampling is i.i.d.
- dim = dim of data

Forward Process



- sampling without simulation
- x_t from x_0 in closed form

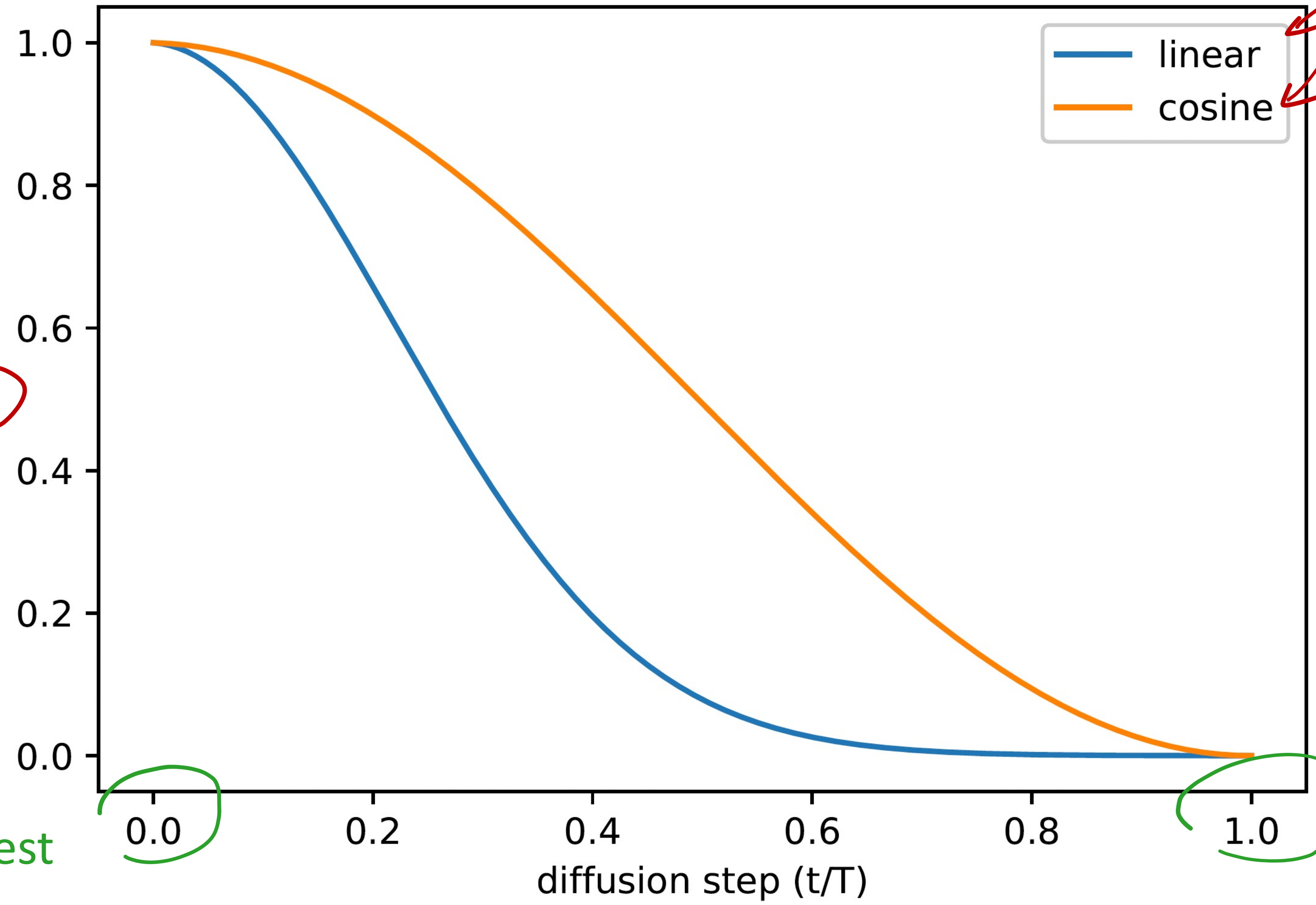
$$q(x_t \mid x_0)$$
$$= \mathcal{N}(x_t \mid \sqrt{\bar{\alpha}_t}x_0, (1-\bar{\alpha}_t)\mathbf{I})$$

coefficients
given by β

$$\alpha_t := 1 - \beta_t$$
$$\bar{\alpha}_t := \prod_{s=1}^{t} \alpha_s$$

Forward Process: Noise Schedule

schedule of β

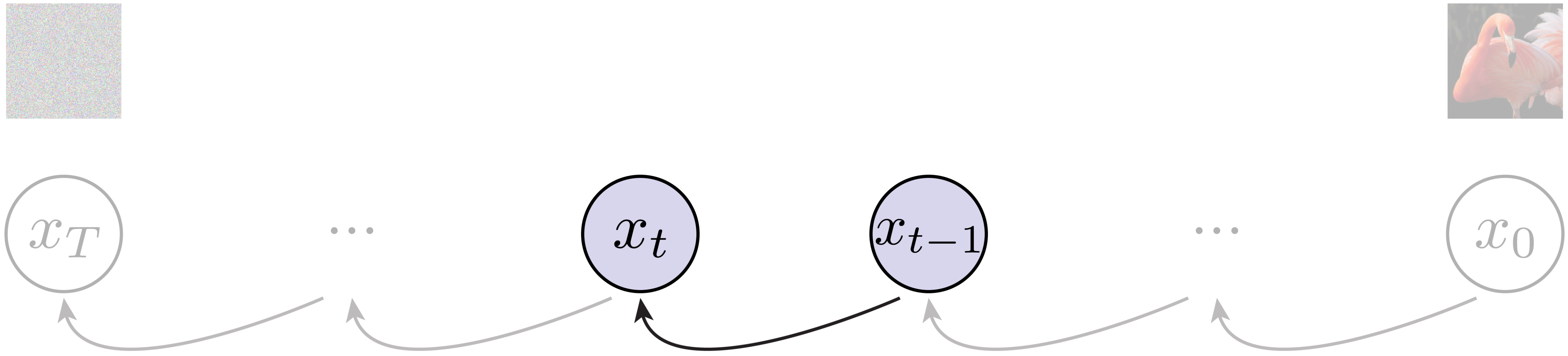


schedule of $\overline{\alpha}$

cleanest

noisiest

Forward Process



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

tl; dr:

- pre-defined conditional distributions
- Gaussian w/ controllable mean/std
- <u>divide</u> and conquer

Diffusion Models

Forward process

- add noise to data

Reverse process

- learn to denoise

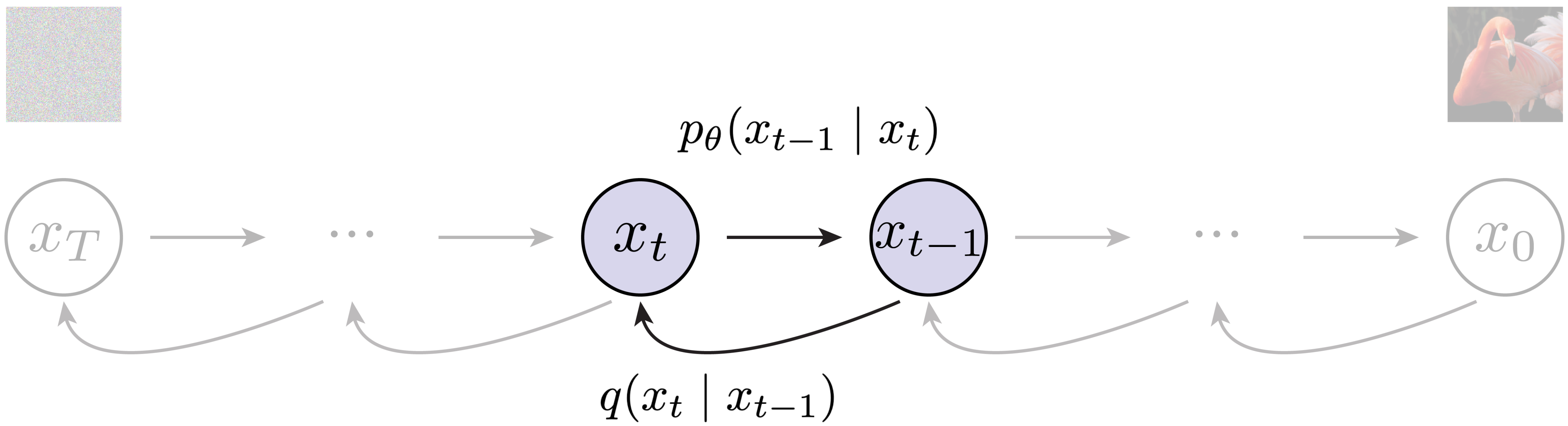
Training objective

- from Hierarchical VAE to L2 loss

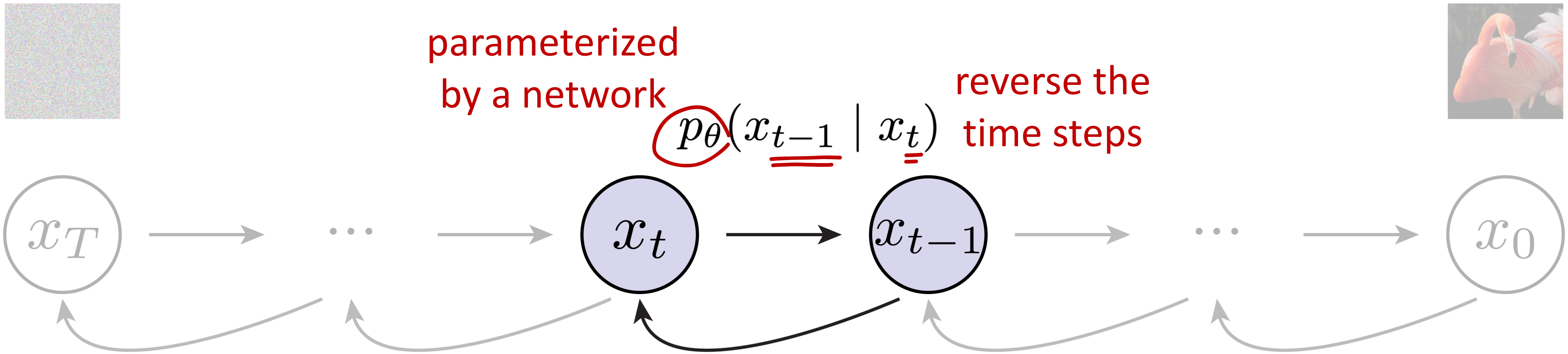
Noise Conditional Network

- represent distributions

Reverse Process



Reverse Process



parameterized by a network

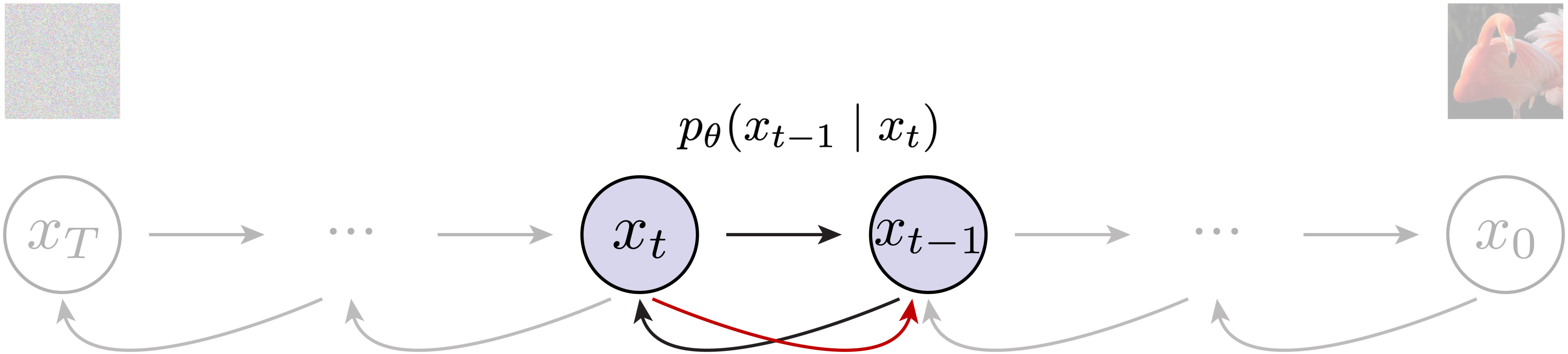
$p_\theta(x_{t-1} \mid x_t)$

reverse the time steps

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

- known
- but not our target

Reverse Process

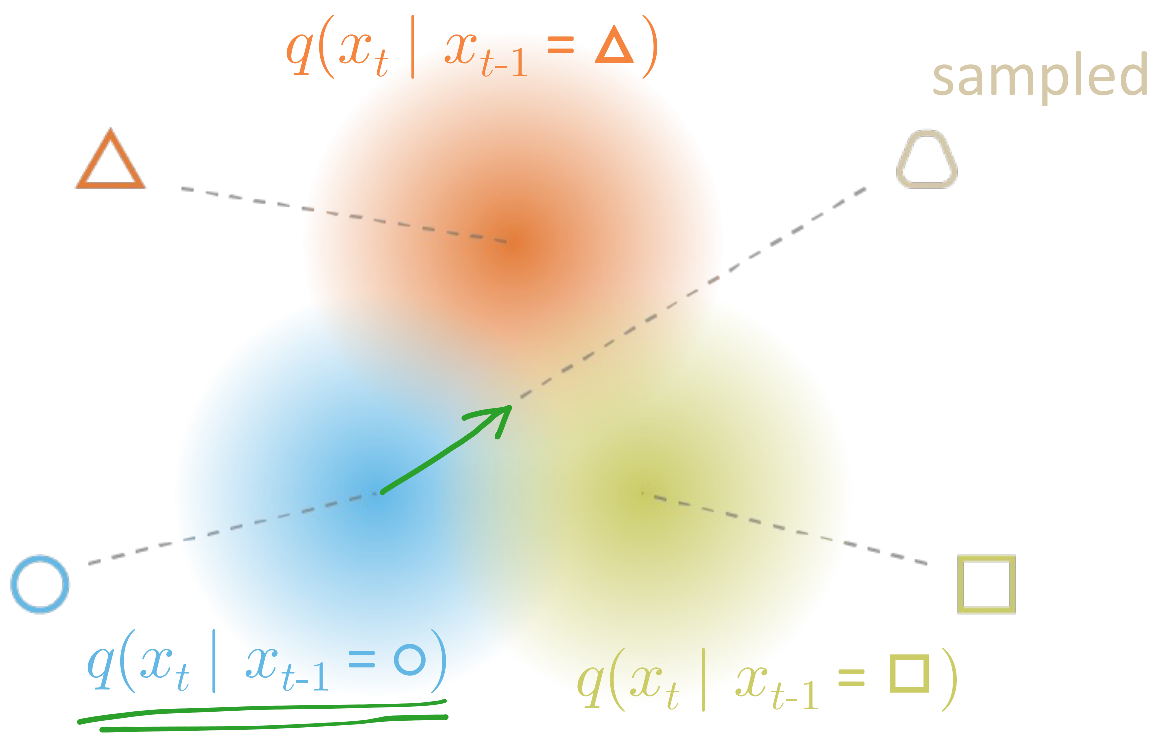


$$\cancel{q(x_t \mid x_{t-1})}$$

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

- our target
- but unknown

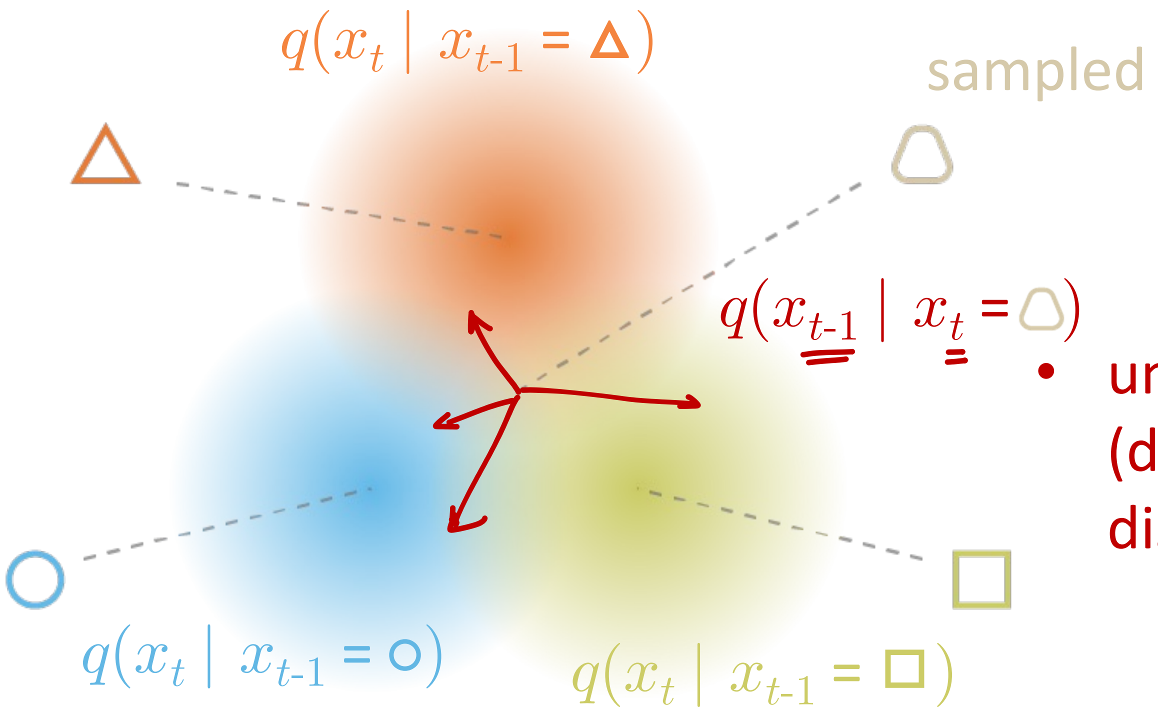
Why are the reverse conditionals unknown?



- known (Gaussian)

Figure adapted from: Joseph Rocca "Understanding Variational Autoencoders (VAEs)"
https://towardsdatascience.com/understanding-variational-autoencoders-vaes-f70510919f73

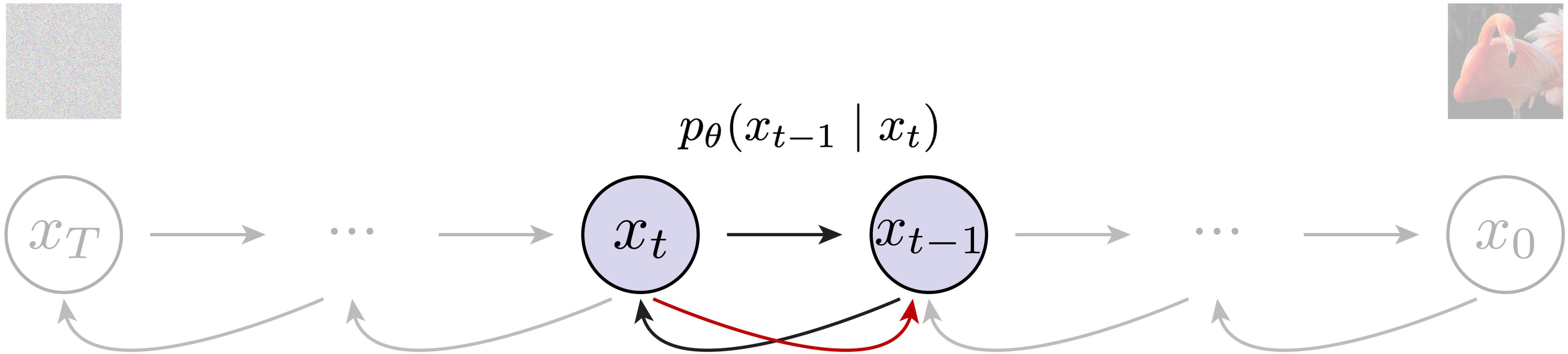
Why are the reverse conditionals unknown?



- unknown (depends on data distribution)

Figure adapted from: Joseph Rocca "Understanding Variational Autoencoders (VAEs)"
https://towardsdatascience.com/understanding-variational-autoencoders-vaes-f70510919f73

Reverse Process



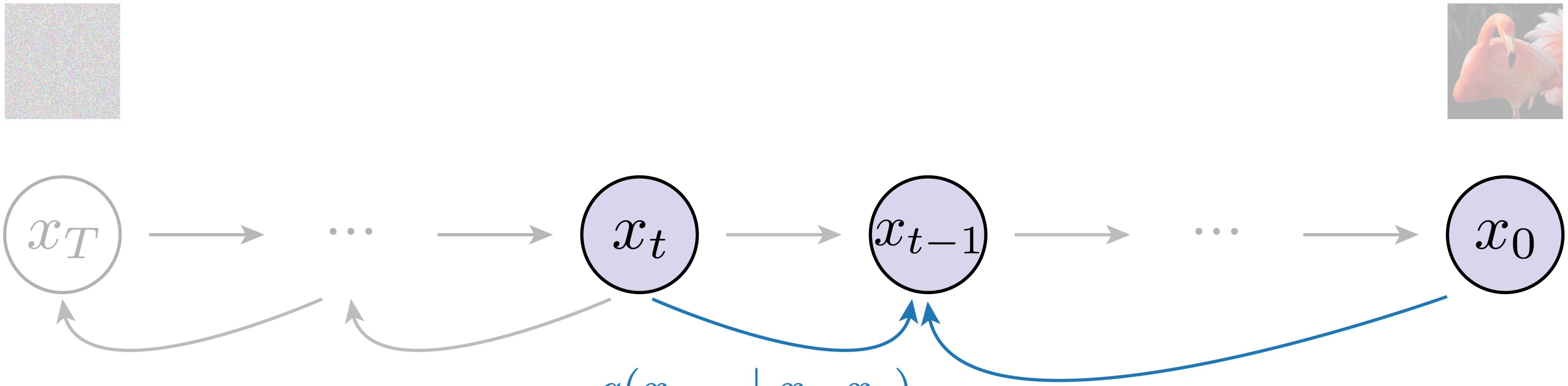
$$\cancel{q(x_t \mid x_{t-1})}$$

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

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

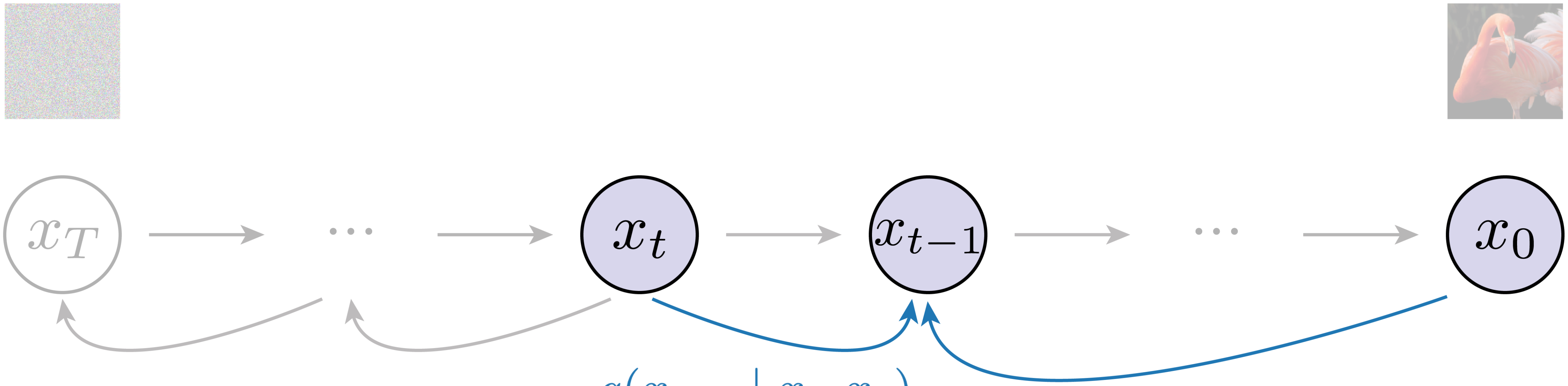
- known
- Gaussian

Reverse Process



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

Reverse Process



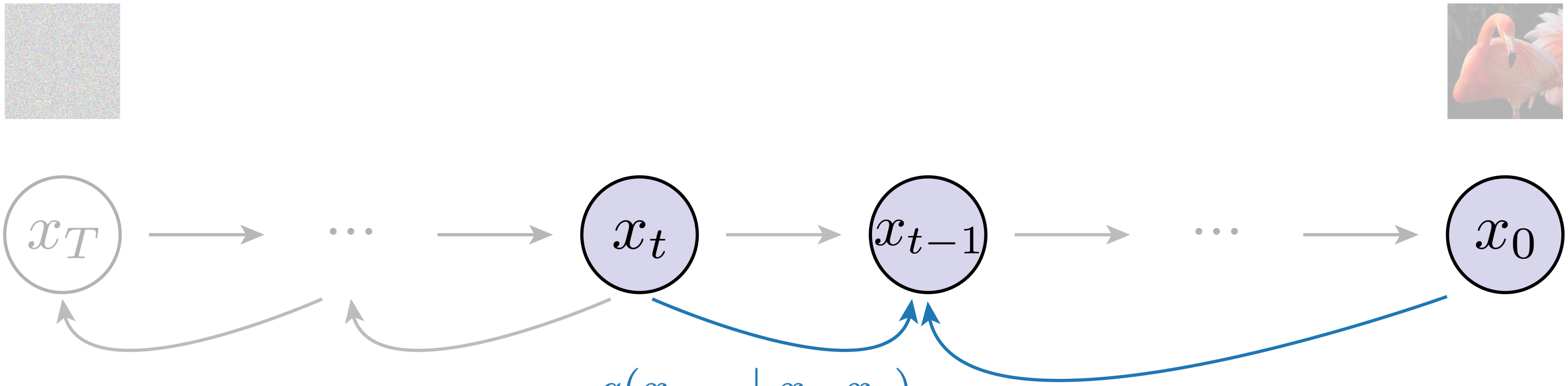
$$q(x_{t-1} \mid x_t, x_0)$$

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

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

var

Reverse Process



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

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

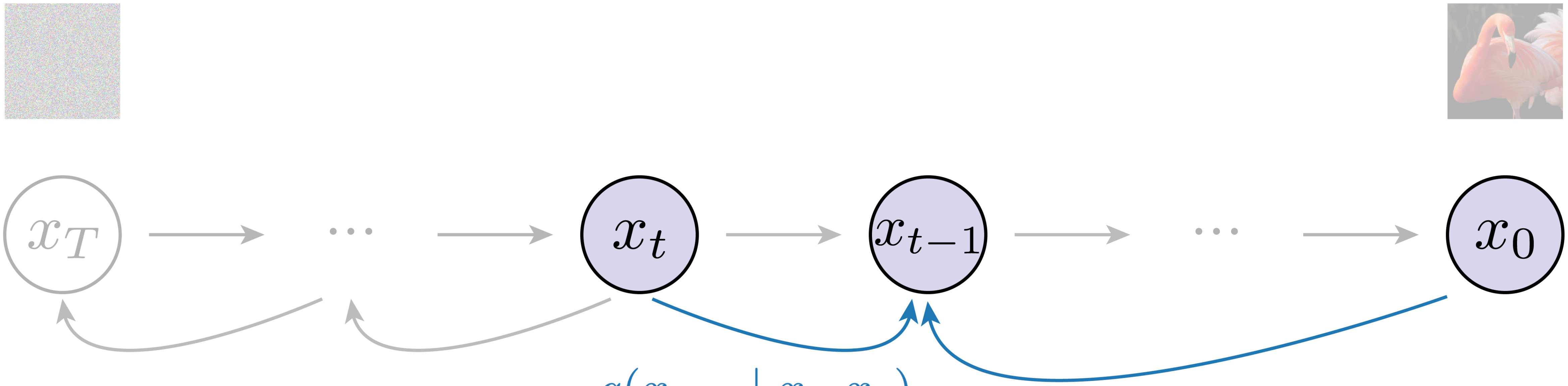
$$\tilde{\mu}_t(x_t, x_0) := \frac{\sqrt{\bar{\alpha}_{t-1}}\beta_t}{1-\bar{\alpha}_t} x_0 + \frac{\sqrt{\alpha_t}(1-\bar{\alpha}_{t-1})}{1-\bar{\alpha}_t} x_t$$

mean

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

var

Reverse Process



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

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

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

mean

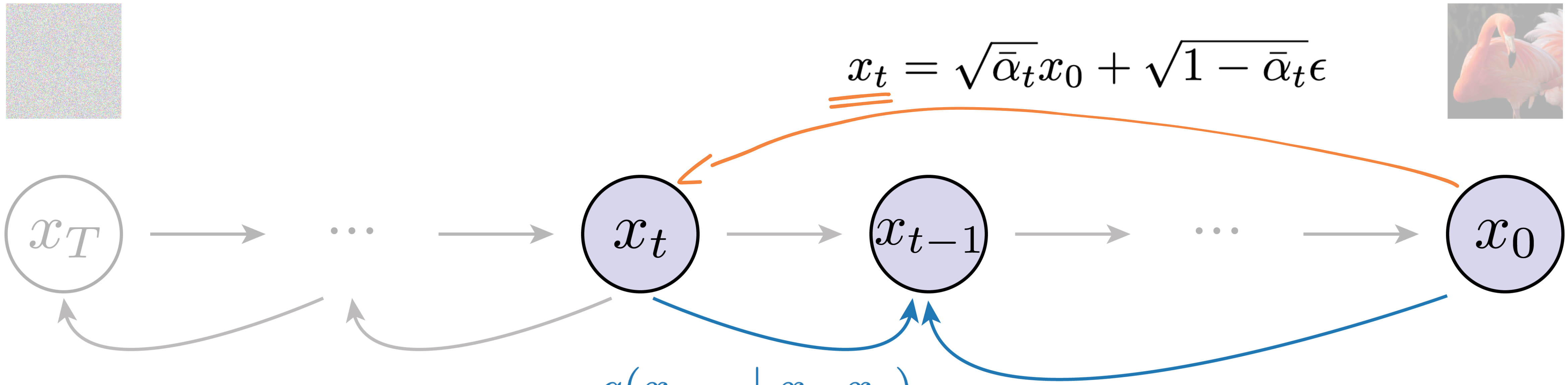
linear combination

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

var

Reverse Process

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



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

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

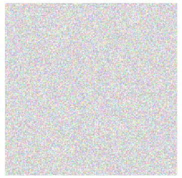
$$\tilde{\mu}_t(x_t, x_0) := \cancel{\frac{\sqrt{\bar{\alpha}_{t-1}}\beta_t}{1-\bar{\alpha}_t}x_0 + \frac{\sqrt{\alpha_t}(1-\bar{\alpha}_{t-1})}{1-\bar{\alpha}_t}x_t}$$

mean

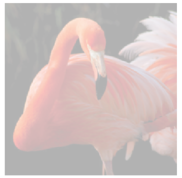
$$\tilde{\beta}_t := \frac{1-\bar{\alpha}_{t-1}}{1-\bar{\alpha}_t}\beta_t$$

var

Reverse Process



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



$$x_T \rightarrow \cdots \rightarrow x_t \rightarrow x_{t-1} \rightarrow \cdots \rightarrow x_0$$

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

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

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

mean

$$= \frac{1}{\sqrt{\bar{\alpha}_t}}\left(x_t - \frac{1-\alpha_t}{\sqrt{1-\bar{\alpha}_t}}\epsilon\right)$$

"noise"

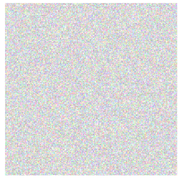
$$\tilde{\beta}_t := \frac{1-\bar{\alpha}_{t-1}}{1-\bar{\alpha}_t}\beta_t$$

var

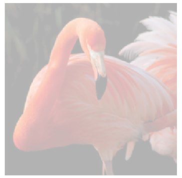
Reverse Process

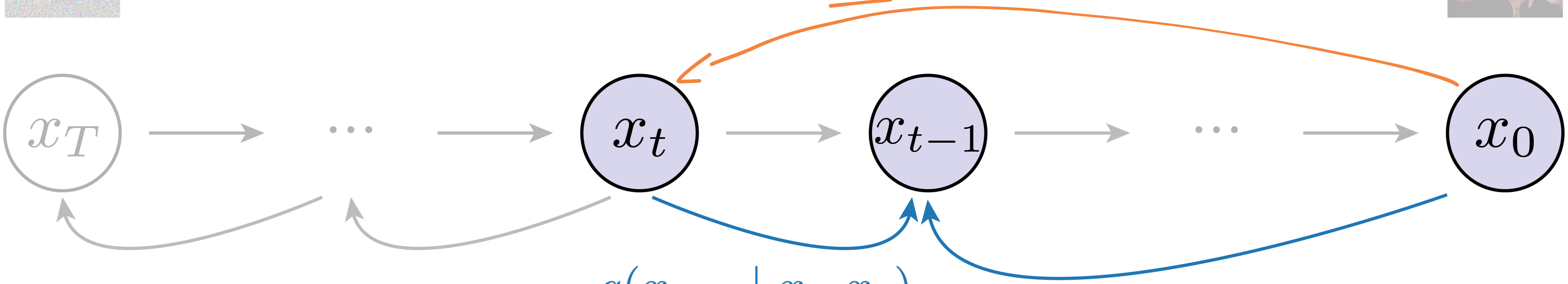
tl; dr:

- outcome of the dependency graph
- some linear combinations



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





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

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

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

mean

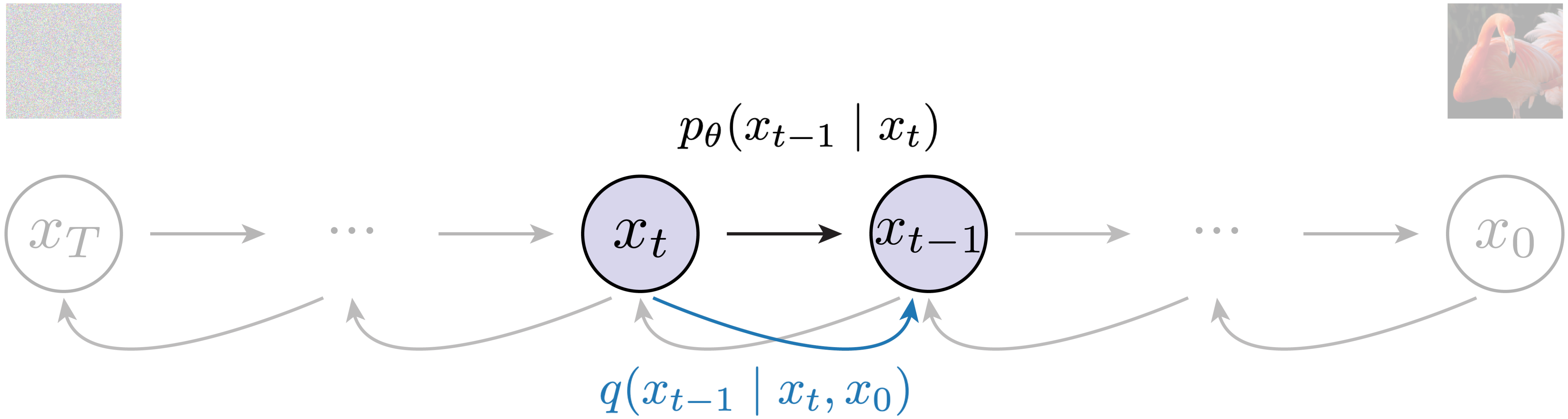
$$= \frac{1}{\sqrt{\bar{\alpha}_t}}\left(x_t - \frac{1-\alpha_t}{\sqrt{1-\bar{\alpha}_t}}\epsilon\right)$$

"noise"

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

var

Reverse Process

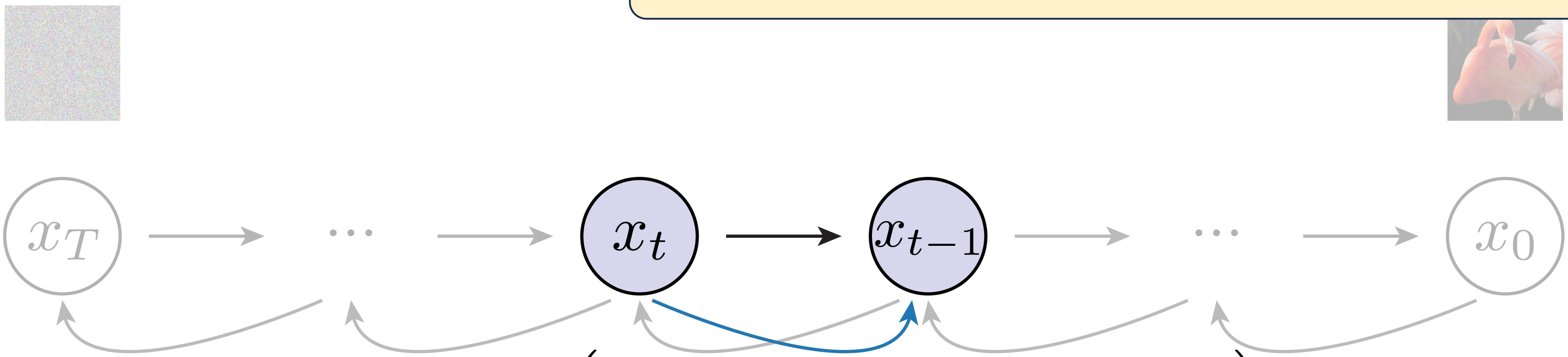


- **tl; dr**: a known Gaussian
- we want to learn it by p_θ
- we can represent p_θ by a Gaussian
- minimize KL divergence

Reverse Process

D_{KL} of two Gaussians is like L2 loss: (pset 1)

$$D_{KL}(\mathcal{N}_1 \parallel \mathcal{N}_2) = \log\left(\frac{\sigma_2}{\sigma_1}\right) + \frac{\sigma_1^2 + \|\mu_1 - \mu_2\|^2}{2\sigma_2^2} - \frac{1}{2}$$

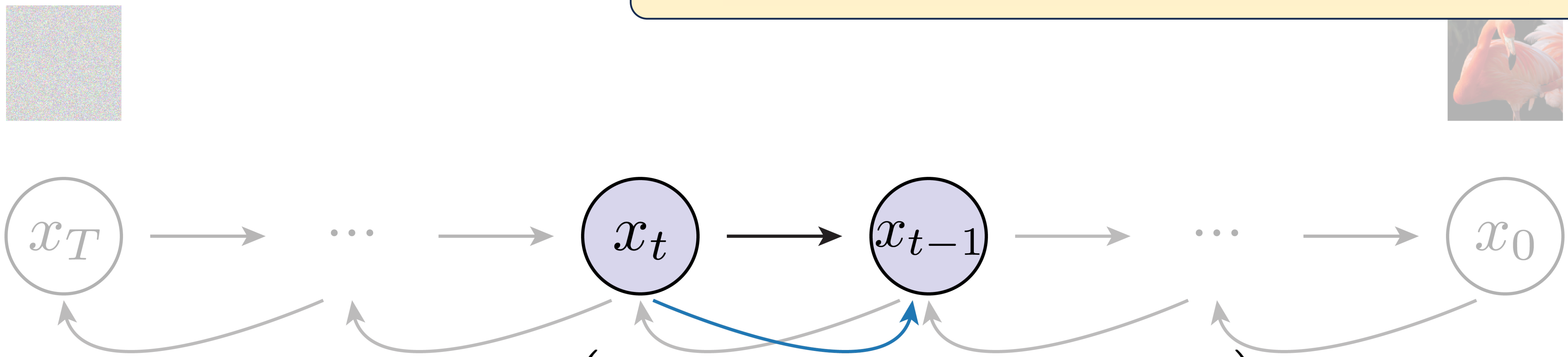


$$D_{\text{KL}}\Big(q(x_{t-1} \mid x_t, x_0) \;||\; p_\theta(x_{t-1} \mid x_t)\Big)$$

Reverse Process

D_{KL} of two Gaussians is like L2 loss: (pset 1)

$$D_{KL}(\mathcal{N}_1 \parallel \mathcal{N}_2) = \log\left(\frac{\sigma_2}{\sigma_1}\right) + \frac{\sigma_1^2 + \|\mu_1 - \mu_2\|^2}{2\sigma_2^2} - \frac{1}{2}$$



$$D_{\mathrm{KL}}\Big(q(x_{t-1} \mid x_t, x_0) \;||\; \underline{p_\theta(x_{t-1} \mid x_t)}\Big)$$

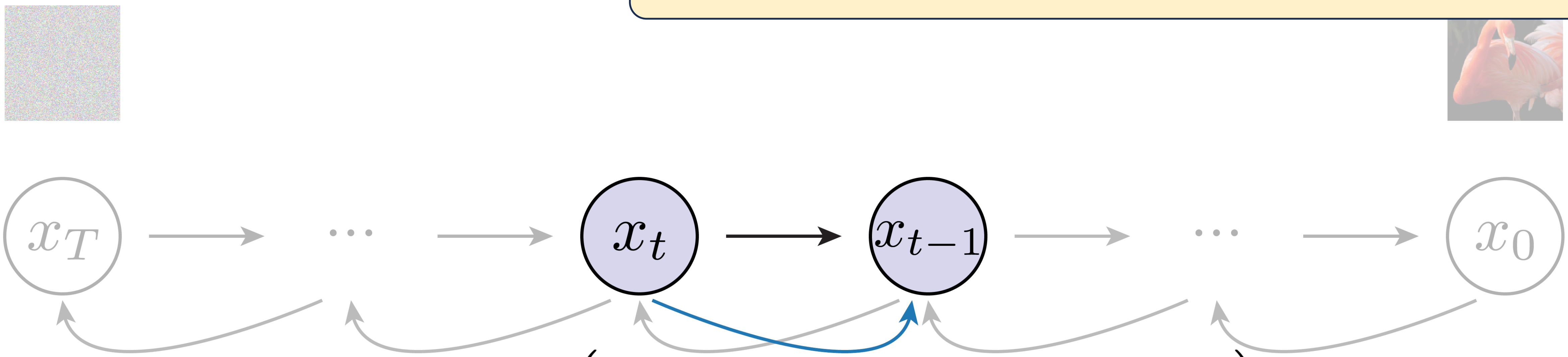
$$\mathcal{N}(x_{t-1} \mid \mu_\theta(x_t), \sigma_t^2 \mathbf{I})$$

learn preset

Reverse Process

D_{KL} of two Gaussians is like L2 loss: (pset 1)

$$D_{KL}(\mathcal{N}_1 \parallel \mathcal{N}_2) = \log\left(\frac{\sigma_2}{\sigma_1}\right) + \frac{\sigma_1^2 + \|\mu_1 - \mu_2\|^2}{2\sigma_2^2} - \frac{1}{2}$$



$$D_{\mathrm{KL}}\left(q(x_{t-1} \mid x_t, x_0) \parallel p_\theta(x_{t-1} \mid x_t)\right)$$

$$\mathcal{N}(x_{t-1} \mid \tilde{\mu}(x_t, x_0), \tilde{\beta}_t \mathbf{I})$$

$$\frac{1}{\sqrt{\bar{\alpha}_t}}\left(x_t - \frac{1-\alpha_t}{\sqrt{1-\bar{\alpha}_t}}\epsilon\right)$$

"noise"

$$\mathcal{N}(x_{t-1} \mid \mu_\theta(x_t), \sigma_t^2 \mathbf{I})$$

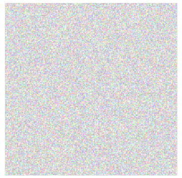
learn

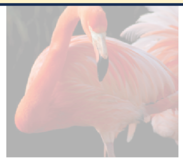
preset

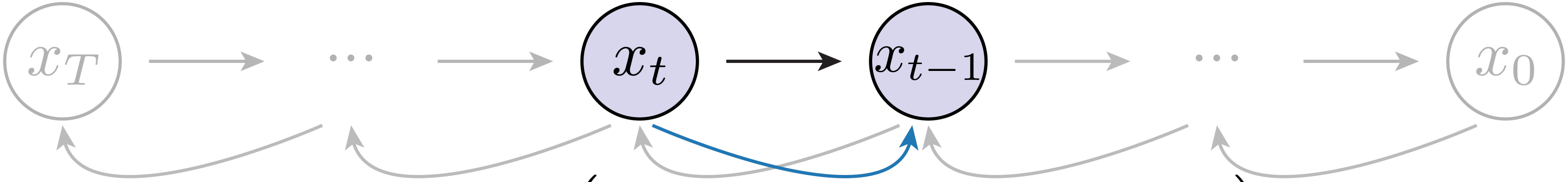
Reverse Process

D_{KL} of two Gaussians is like L2 loss: (pset 1)

$$D_{KL}(\mathcal{N}_1 \parallel \mathcal{N}_2) = \log\left(\frac{\sigma_2}{\sigma_1}\right) + \frac{\sigma_1^2 + \|\mu_1 - \mu_2\|^2}{2\sigma_2^2} - \frac{1}{2}$$







$$D_{\mathrm{KL}}\left(\underline{q(x_{t-1} \mid x_t, x_0)} \;\|\; \underline{p_\theta(x_{t-1} \mid x_t)}\right)$$

$$\mathcal{N}(x_{t-1} \mid \tilde{\mu}(x_t, x_0), \tilde{\beta}_t \mathbf{I}) \qquad \mathcal{N}(x_{t-1} \mid \mu_\theta(x_t), \sigma_t^2 \mathbf{I})$$

learn preset

$$\frac{1}{\sqrt{\bar{\alpha}_t}}\left(x_t - \frac{1-\alpha_t}{\sqrt{1-\bar{\alpha}_t}}\epsilon\right)$$

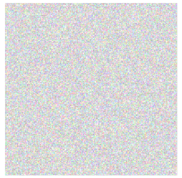
“noise”

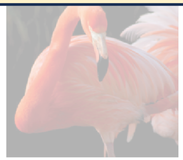
estimate noise

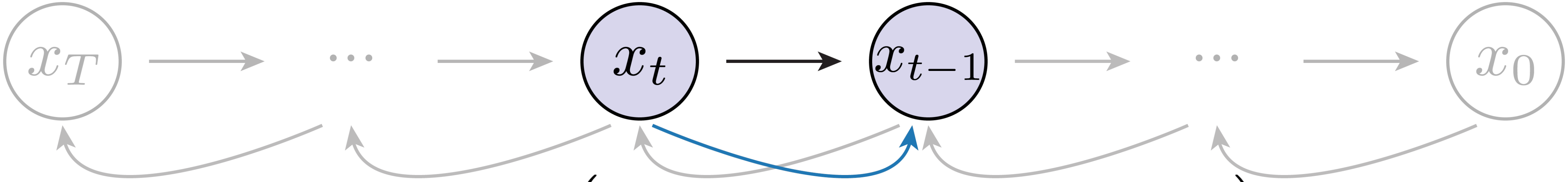
Reverse Process

D_{KL} of two Gaussians is like L2 loss: (pset 1)

$$D_{KL}(\mathcal{N}_1 \parallel \mathcal{N}_2) = \log\left(\frac{\sigma_2}{\sigma_1}\right) + \frac{\sigma_1^2 + \|\mu_1 - \mu_2\|^2}{2\sigma_2^2} - \frac{1}{2}$$







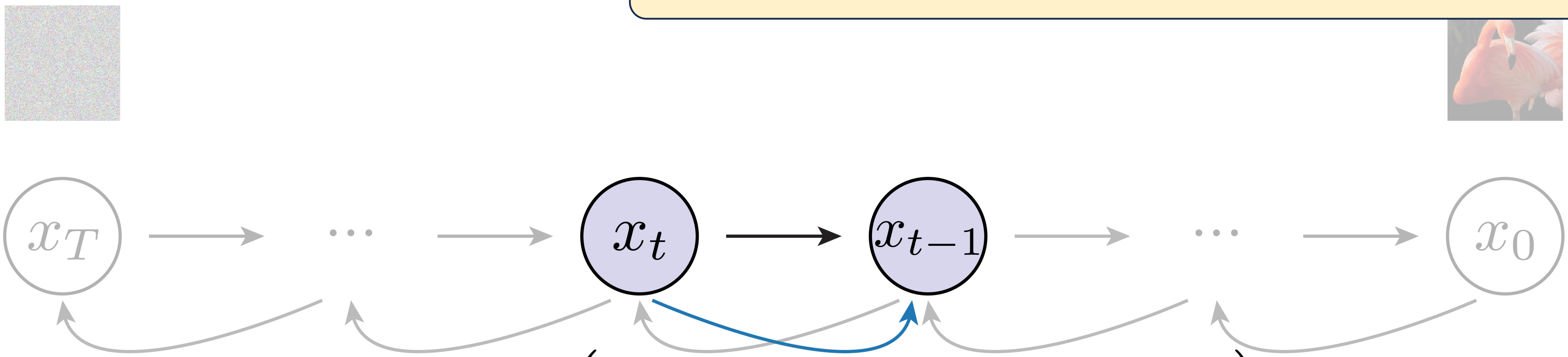
$$\underbrace{D_{\mathrm{KL}}\Big(q(x_{t-1} \mid x_t, x_0) \;||\; p_\theta(x_{t-1} \mid x_t)\Big)}$$

$$w_t \|\epsilon - \epsilon_\theta(x_t)\|^2$$

Reverse Process

D_{KL} of two Gaussians is like L2 loss: (pset 1)

$$D_{KL}(\mathcal{N}_1 \parallel \mathcal{N}_2) = \log\left(\frac{\sigma_2}{\sigma_1}\right) + \frac{\sigma_1^2 + \|\mu_1 - \mu_2\|^2}{2\sigma_2^2} - \frac{1}{2}$$



$$D_{\mathrm{KL}}\left(q(x_{t-1} \mid x_t, x_0) \,||\, p_\theta(x_{t-1} \mid x_t)\right)$$

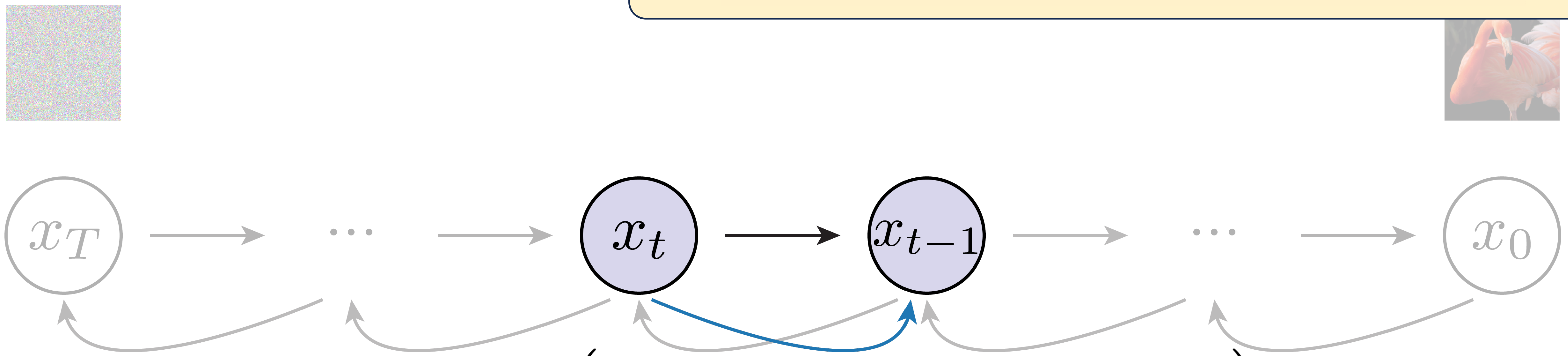
$$w_t \|\epsilon - \epsilon_\theta(x_t)\|^2$$

- a network to predict noise
- input: noisy image

Reverse Process

D_{KL} of two Gaussians is like L2 loss: (pset 1)

$$D_{KL}(\mathcal{N}_1 \parallel \mathcal{N}_2) = \log\left(\frac{\sigma_2}{\sigma_1}\right) + \frac{\sigma_1^2 + \|\mu_1 - \mu_2\|^2}{2\sigma_2^2} - \frac{1}{2}$$



$$D_{\mathrm{KL}}\Big(q(x_{t-1} \mid x_t, x_0) \;||\; p_\theta(x_{t-1} \mid x_t)\Big)$$

$$w_t \|\epsilon - \epsilon_\theta(x_t)\|^2$$

- weights due to α_t , β_t
- but set as 1 **(critical)**

- a network to predict noise
- input: noisy image

tl; dr

- some dependency graphs
- some linear combinations
- D_{KL}
- L2 loss of noise

Diffusion Models

Forward process

- add noise to data

Reverse process

- learn to denoise

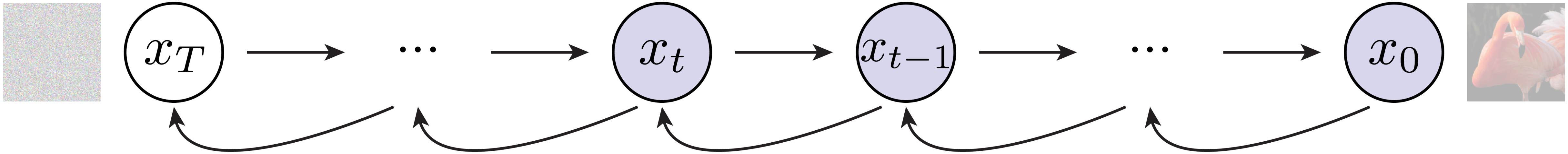
Training objective

- from Hierarchical VAE to L2 loss

Noise Conditional Network

- represent a distribution

Training Objective



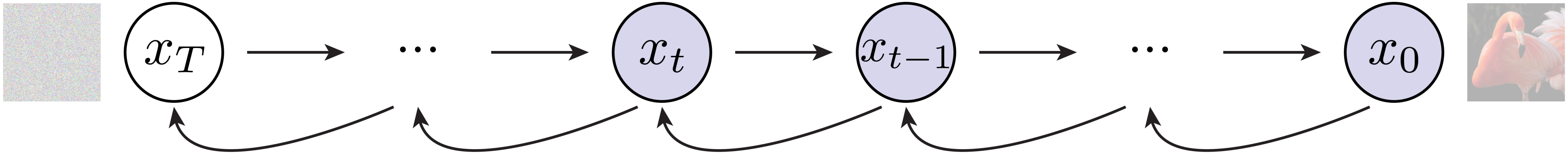
$$\mathcal{L}_{\text{VLB}} := \mathcal{L}_T + \mathcal{L}_{T-1} + \ldots + \mathcal{L}_0$$

$$\mathcal{L}_T := D_{\text{KL}}\Big(q(x_T \mid x_0) \,||\, p_\theta(x_T)\Big)$$

$$\mathcal{L}_{t-1} := D_{\text{KL}}\Big(q(x_{t-1} \mid x_t, x_0) \,||\, p_\theta(x_{t-1} \mid x_t)\Big)$$

$$\mathcal{L}_0 := -\log p_\theta(x_0 \mid x_1)$$

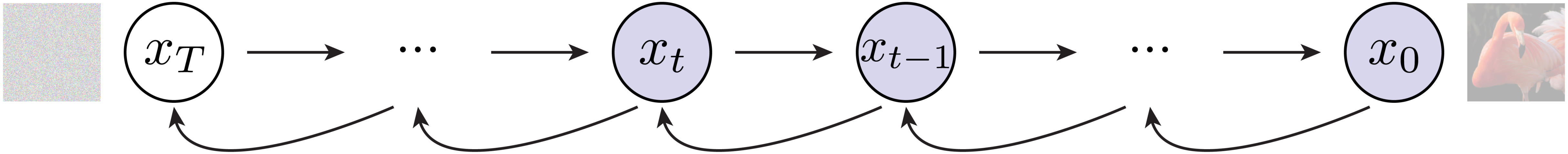
Training Objective



- variational lower bound
- like ELBO

$$
\begin{aligned}
\mathcal{L}_{\mathrm{VLB}} &:= \mathcal{L}_T + \mathcal{L}_{T-1} + \ldots + \mathcal{L}_0 \\
\mathcal{L}_T &:= D_{\mathrm{KL}}\Big(q(x_T \mid x_0) \,||\, p_\theta(x_T)\Big) \\
\mathcal{L}_{t-1} &:= D_{\mathrm{KL}}\Big(q(x_{t-1} \mid x_t, x_0) \,||\, p_\theta(x_{t-1} \mid x_t)\Big) \\
\mathcal{L}_0 &:= -\log p_\theta(x_0 \mid x_1)
\end{aligned}
$$

Training Objective



- variational lower bound
- like ELBO

$$\mathcal{L}_{\text{VLB}} := \mathcal{L}_T + \mathcal{L}_{T-1} + \ldots + \mathcal{L}_0$$

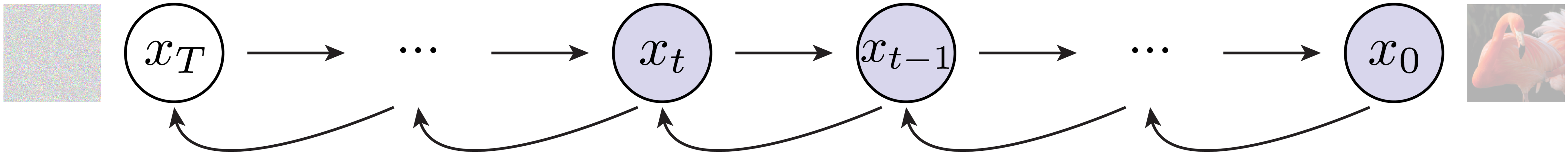
$$\mathcal{L}_T := D_{\mathrm{KL}}\Big(q(\overset{z}{\cancel{x_T}} \mid x_0) \,||\, p_\theta(\overset{z}{\cancel{x_T}})\Big)$$

it's ELBO if one step

$$\cancel{\mathcal{L}_{t-1} := D_{\mathrm{KL}}\Big(q(x_{t-1} \mid x_t, x_0) \,||\, p_\theta(x_{t-1} \mid x_t)\Big)}$$

$$\mathcal{L}_0 := -\log p_\theta(x_0 \mid \overset{z}{\cancel{x_1}})$$

Training Objective



$$\mathcal{L}_{\text{VLB}} := \mathcal{L}_T + \mathcal{L}_{T-1} + \ldots + \mathcal{L}_0$$

no parameter, unlike VAE's q_ϕ

$$\mathcal{L}_T := D_{\text{KL}}\Big(\underline{q(x_T \mid x_0)} \,\|\, \underline{p_{\cancel{\theta}}(x_T)}\Big)$$

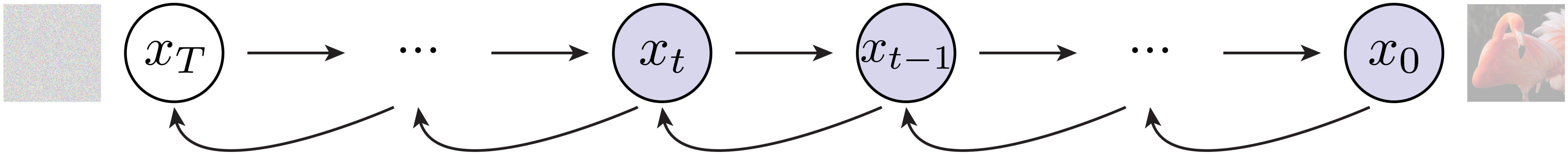
constant

Gaussian

$$\mathcal{L}_{t-1} := D_{\text{KL}}\Big(q(x_{t-1} \mid x_t, x_0) \,\|\, p_\theta(x_{t-1} \mid x_t)\Big)$$

$$\mathcal{L}_0 := -\log p_\theta(x_0 \mid x_1)$$

Training Objective



$$\mathcal{L}_{\text{VLB}} := \mathcal{L}_T + \mathcal{L}_{T-1} + \ldots + \mathcal{L}_0$$

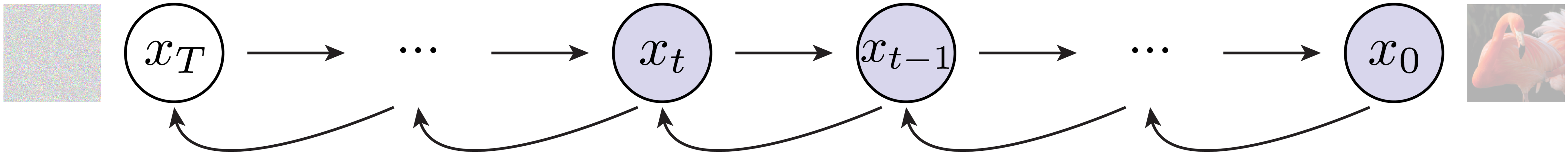
$$\mathcal{L}_T := D_{\text{KL}}\Big(q(x_T \mid x_0) \,||\, p_\theta(x_T)\Big)$$

$$\mathcal{L}_{t-1} := D_{\text{KL}}\Big(q(x_{t-1} \mid x_t, x_0) \,||\, p_\theta(x_{t-1} \mid x_t)\Big)$$

reconstruction loss,
like VAE

$$\mathcal{L}_0 := -\log p_\theta(x_0 \mid x_1)$$

Training Objective



$$\mathcal{L}_{\text{VLB}} := \mathcal{L}_T + \mathcal{L}_{T-1} + \ldots + \mathcal{L}_0$$

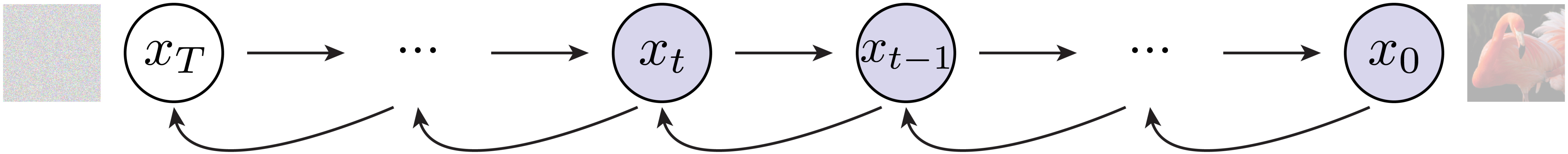
$$\mathcal{L}_T := D_{\text{KL}}\Big(q(x_T \mid x_0) \,||\, p_\theta(x_T)\Big)$$

L2 loss on noise

$$\mathcal{L}_{t-1} := D_{\text{KL}}\Big(q(x_{t-1} \mid x_t, x_0) \,||\, p_\theta(x_{t-1} \mid x_t)\Big)$$

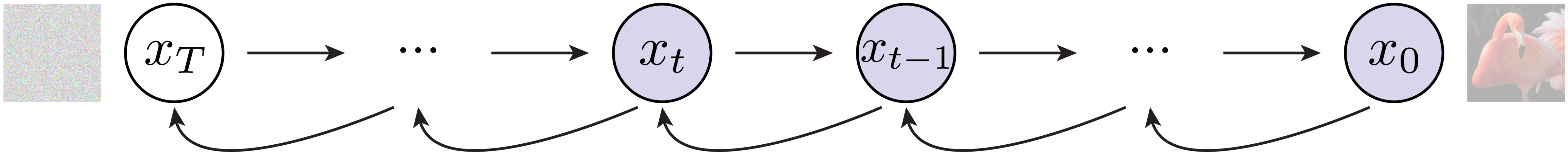
$$\mathcal{L}_0 := -\log p_\theta(x_0 \mid x_1)$$

Training Objective



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

Training Objective



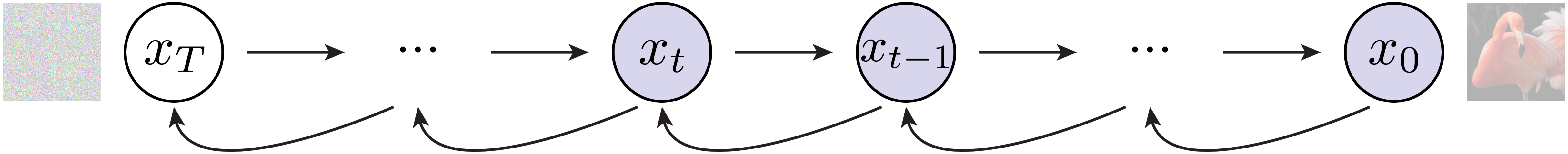
$$\mathcal{L} = \mathbb{E}_{x_0, t, \epsilon}\left[w_t \|\epsilon - \epsilon_\theta(x_t, t)\|^2 \right]$$

over p_{data}

over $[1, T]$

over $\mathcal{N}(0, \mathbf{I})$

Training Objective



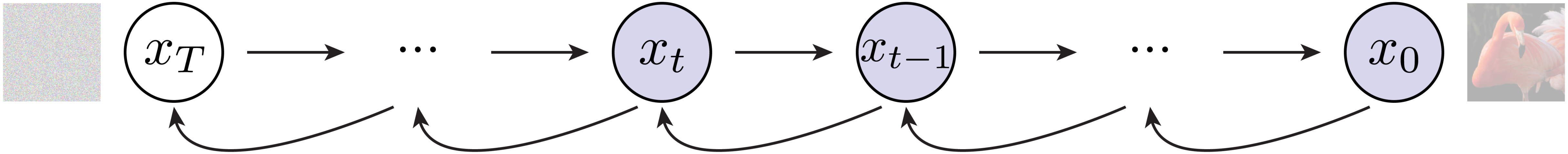
$$\mathcal{L} = \mathbb{E}_{x_0,t,\epsilon}\left[w_t \|\epsilon - \epsilon_\theta(x_t, t)\|^2 \right]$$

set as 1 **(critical)**

Objective	IS	FID
L, learned diagonal $\mathbf{\Sigma}$	–	–
L, fixed isotropic $\mathbf{\Sigma}$	7.67 ± 0.13	13.51
$\|\tilde{\boldsymbol{\epsilon}} - \boldsymbol{\epsilon}_\theta\|^2$ (L_{simple})	$\mathbf{9.46 \pm 0.11}$	$\mathbf{3.17}$

[Ho et al. 2020]; see more in [Salimans & Ho, 2022]

Training Objective



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

network to predict noise

conditioned on noise level **(critical)**

Diffusion Models

Forward process

- add noise to data

Reverse process

- learn to denoise

Training objective

- from Hierarchical VAE to L2 loss

Noise Conditional Network

- represent a distribution

Noise Conditional Network

- Diffusion models decompose a distribution into **many** simpler ones.
- We need the same # networks to fit **all** of them.
- We can **combine** all into one "powerful" network.
- This network is conditioned on noise level t.

- **Noise Conditional Network** [Song & Ermon 2019]: things made work

*It is called Noise Conditional Score Network (NCSN) in [Song & Ermon 2019] in the context of score matching.

Noise Conditional Network

How to represent $p_\theta(x_{t-1} \mid x_t)$

- network input: x_t
- network output: μ and σ of a distribution

- parametrize μ by: $\epsilon_\theta(x_t, t)$

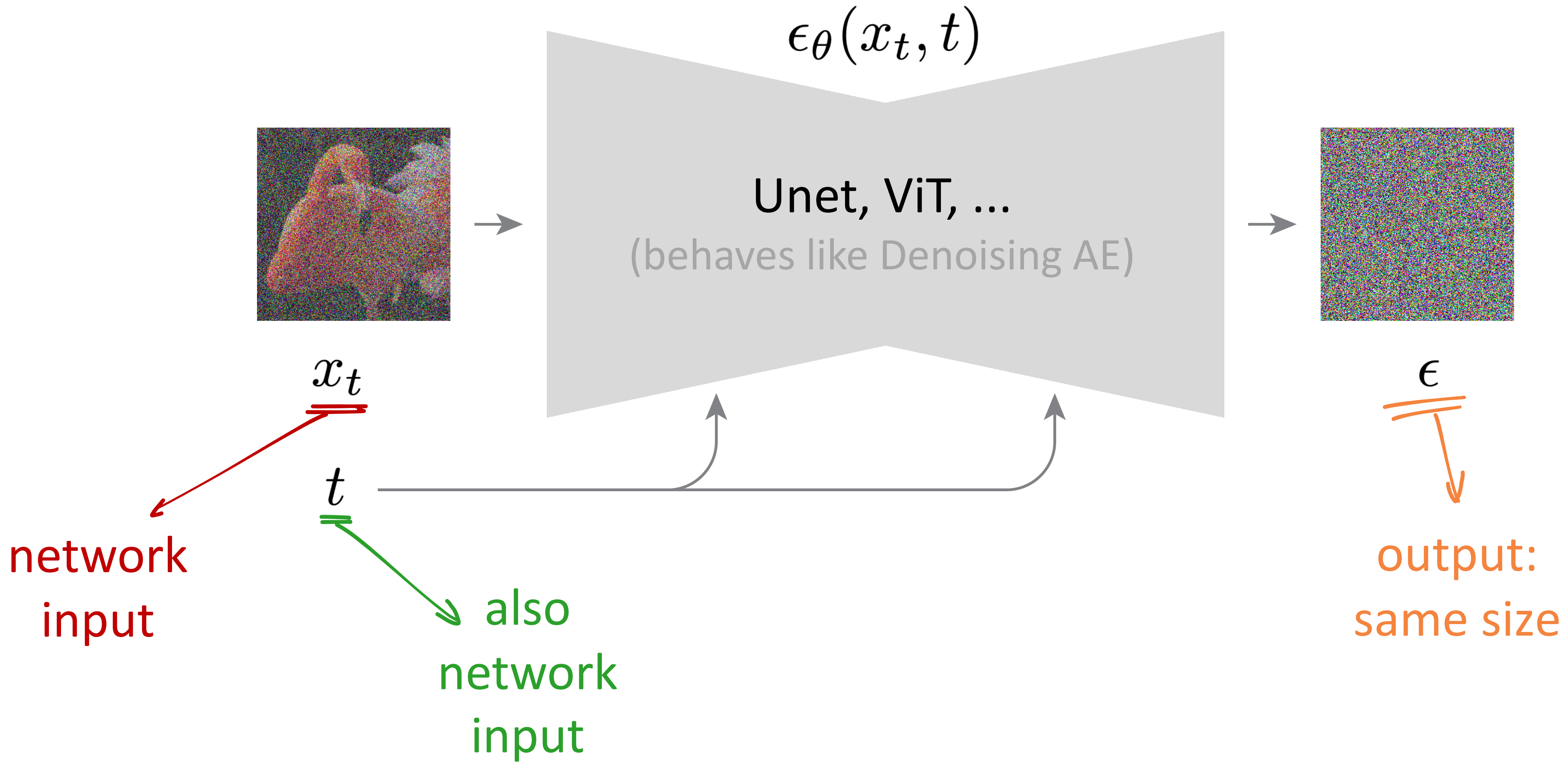
noisy image:

- condition
- network input

noise level:

- condition
- network input

Noise Conditional Network



Diffusion algorithm annotated:

Algorithm 1 Training

1: **repeat**
2: $\quad \mathbf{x}_0 \sim q(\mathbf{x}_0)$
3: $\quad t \sim \mathrm{Uniform}(\{1, \ldots, T\})$
4: $\quad \boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$
5: $\quad$ Take gradient descent step on

$$\nabla_\theta \left\| \boldsymbol{\epsilon} - \boldsymbol{\epsilon}_\theta\big(\overbrace{\sqrt{\bar\alpha_t}\mathbf{x}_0 + \sqrt{1-\bar\alpha_t}\boldsymbol{\epsilon}}^{x_t}, t\big) \right\|^2$$

6: **until** converged

Algorithm 2 Sampling

1: $\mathbf{x}_T \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$
2: **for** $t = T, \ldots, 1$ **do**
3: $\quad \mathbf{z} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ if $t > 1$, else $\mathbf{z} = \mathbf{0}$
4: $\quad \mathbf{x}_{t-1} = \overbrace{\frac{1}{\sqrt{\alpha_t}}\left(\mathbf{x}_t - \frac{1-\alpha_t}{\sqrt{1-\bar\alpha_t}}\boldsymbol{\epsilon}_\theta(\mathbf{x}_t, t)\right)}^{\text{estimated } \mu} + \underbrace{\sigma_t \mathbf{z}}_{\text{sampling from estimated distribution}}$
5: **end for**
6: **return** $\mathbf{x}_0$

Diffusion algorithm annotated:

Algorithm 1 Training

1: **repeat**
2: $\mathbf{x}_0 \sim q(\mathbf{x}_0)$
3: $t \sim \mathrm{Uniform}(\{1, \ldots, T\})$
4: $\boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$
5: Take gradient descent step on
$$\nabla_\theta \left\| \boldsymbol{\epsilon} - \boldsymbol{\epsilon}_\theta(\sqrt{\bar{\alpha}_t}\mathbf{x}_0 + \sqrt{1-\bar{\alpha}_t}\boldsymbol{\epsilon}, t) \right\|^2$$
6: **until** converged

Algorithm 2 Sampling

1: $\mathbf{x}_T \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$
2: **for** $t = T, \ldots, 1$ **do**
3: $\mathbf{z} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ if $t > 1$, else $\mathbf{z} = \mathbf{0}$
4: $\mathbf{x}_{t-1} = \frac{1}{\sqrt{\alpha_t}}\left(\mathbf{x}_t - \frac{1-\alpha_t}{\sqrt{1-\bar{\alpha}_t}}\boldsymbol{\epsilon}_\theta(\mathbf{x}_t, t)\right) + \sigma_t \mathbf{z}$
5: **end for**
6: **return** $\mathbf{x}_0$

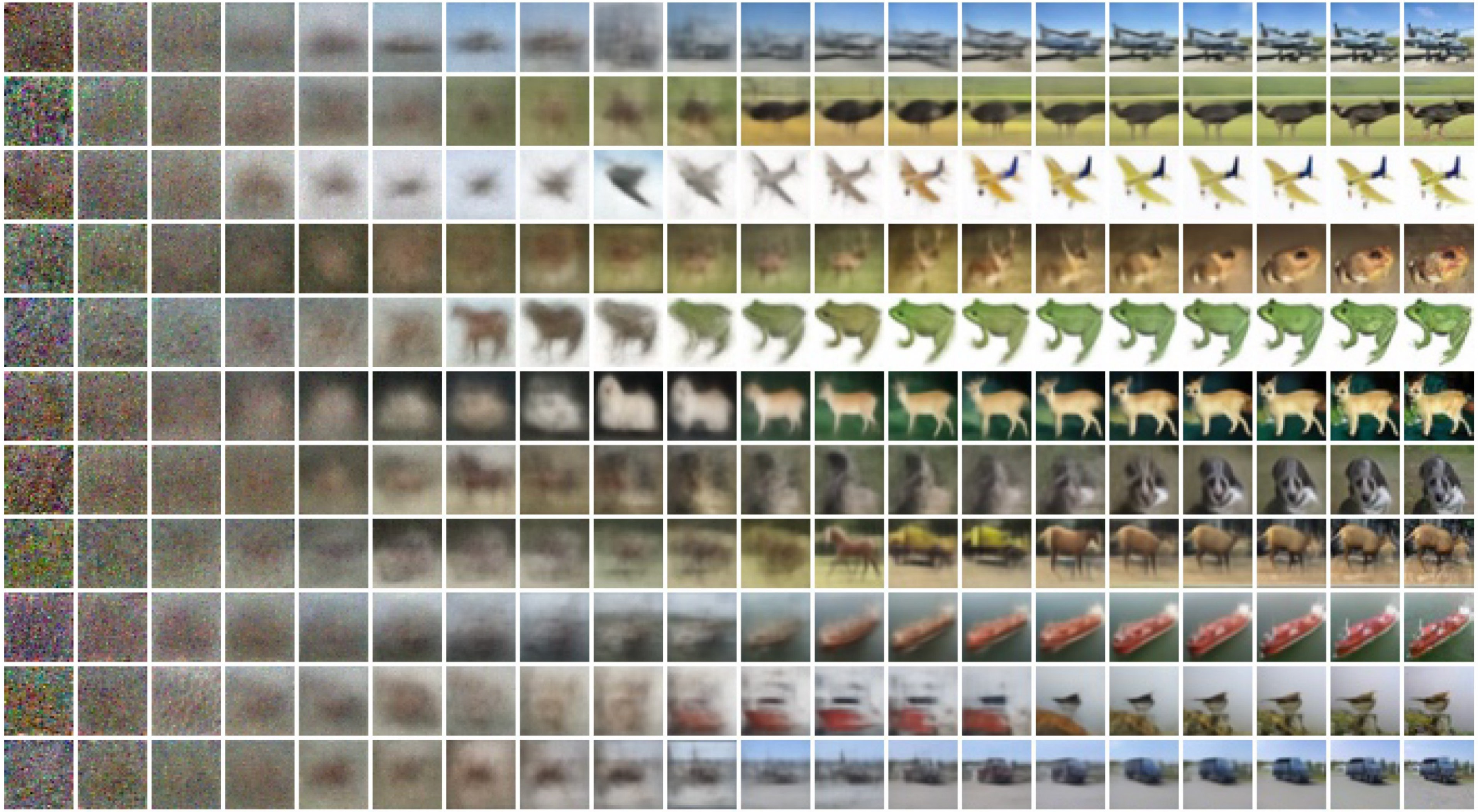
tl; dr: noising and denoising

- Turns out to be extremely simple
- Being “simple and effective” moves the needle

Example: Unconditional Generation on CIFAR-10

noise

x_T



generated

x_0

[Ho et al, 2020]

Example: shared intermediate latents

[Ho et al, 2020]

Summary

Forward process

- add noise to data

Reverse process

- learn to denoise

Training objective

- from Hierarchical VAE to L2 loss

Noise Conditional Network

- represent distributions

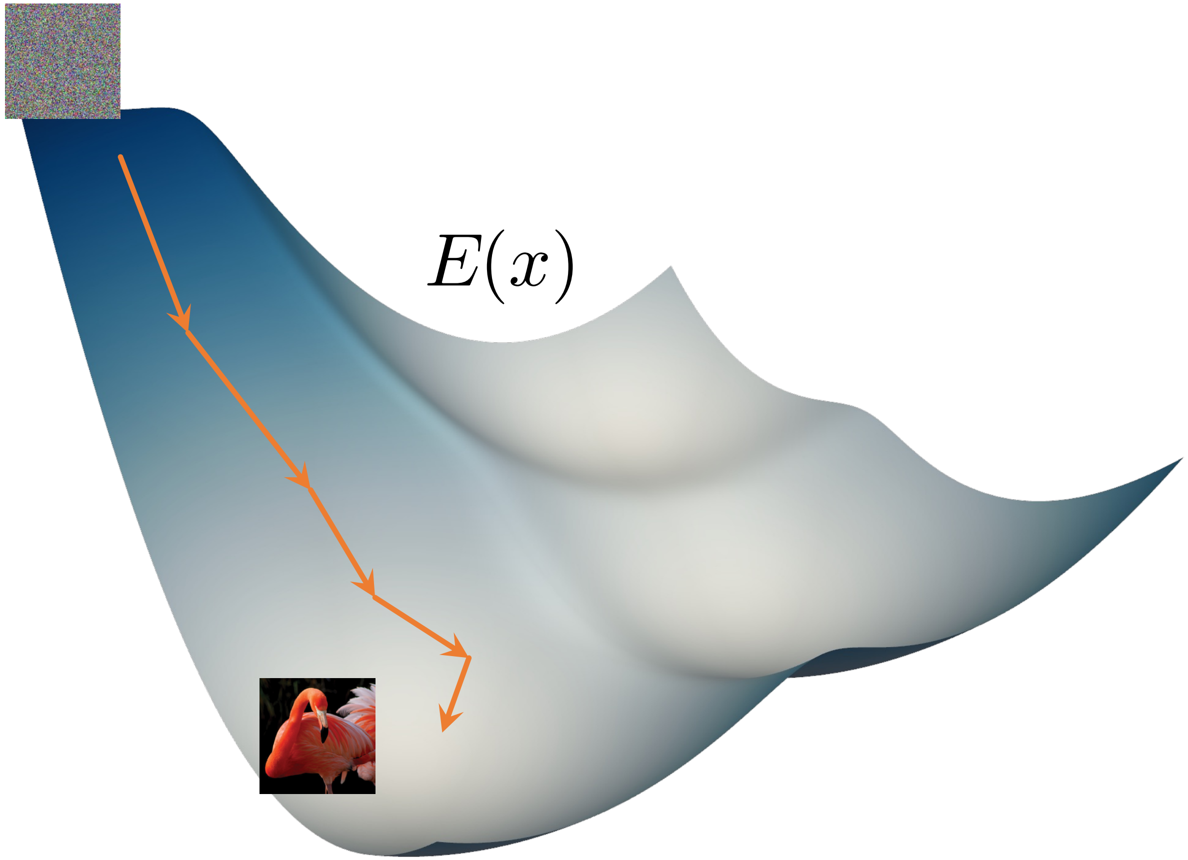
Energy-based Models and Score Matching

Diffusion and Score Matching

- Diffusion Models are closely related to **Score Matching**.
- Score Matching is one solution to **Energy-based Models**.
- Energy-based Models:
  - can be probabilistic or non-probabilistic
  - can be generative or discriminative
- Many useful concepts in diffusion co-evolved w/ score matching
  - Annealed importance sampling [Neal 1998]
  - Denoising score matching [Vincent 2011]
  - Noise Conditional Score Network [Song & Ermon 2019]

Energy-based Models

- Define a <u>scalar</u> function, called “energy”.
- At <u>inference</u> time, find x that minimizes energy

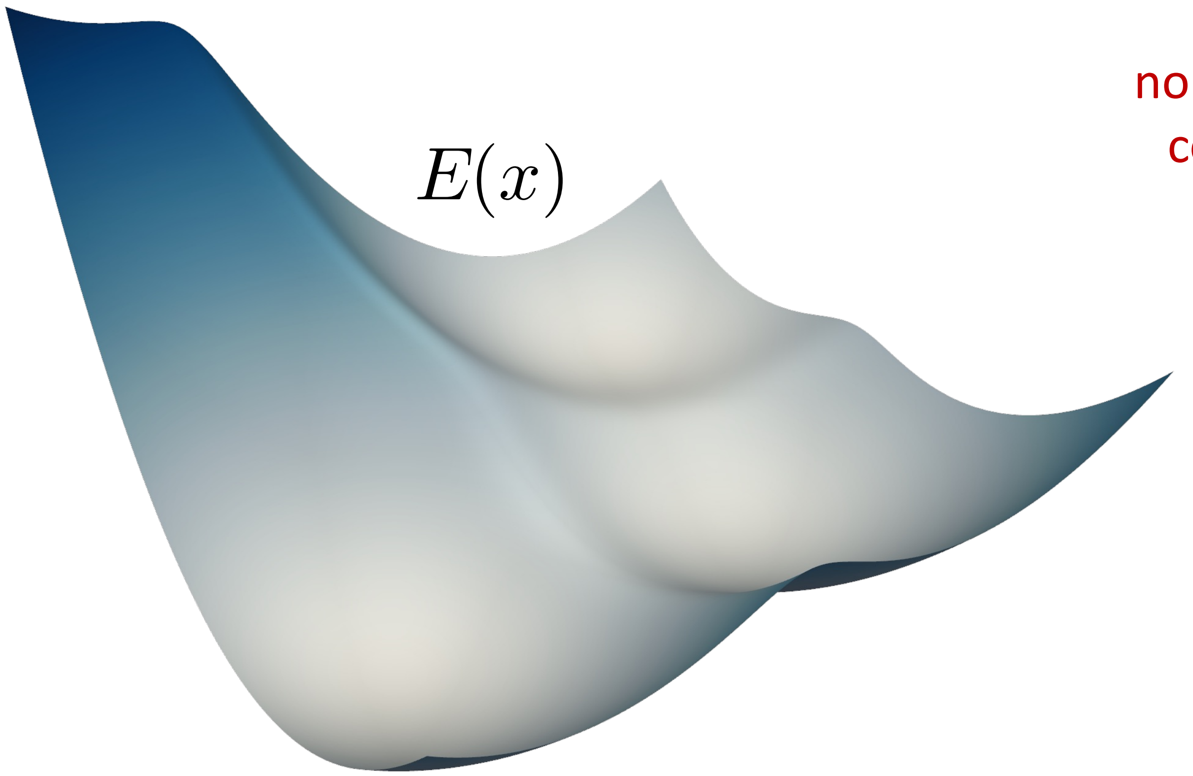


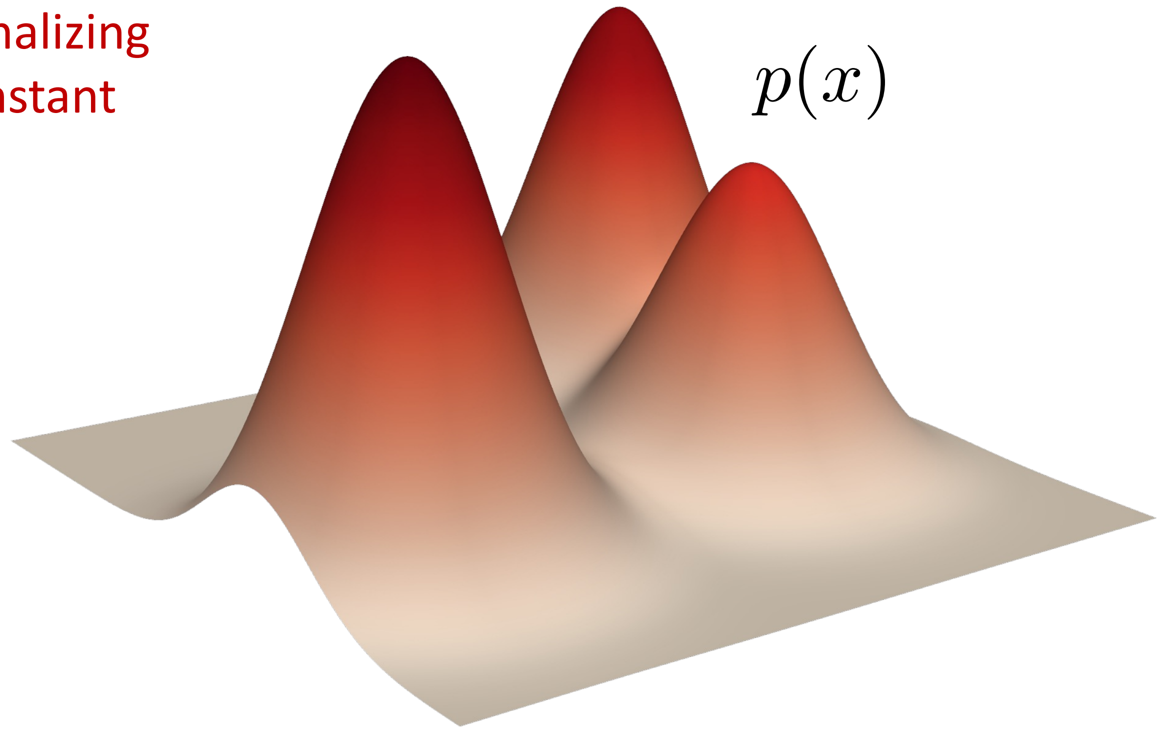
Energy-based Models

- We can use an energy to model a probability distribution

$$p(x) = \frac{\exp(-E(x))}{Z}$$

normalizing constant





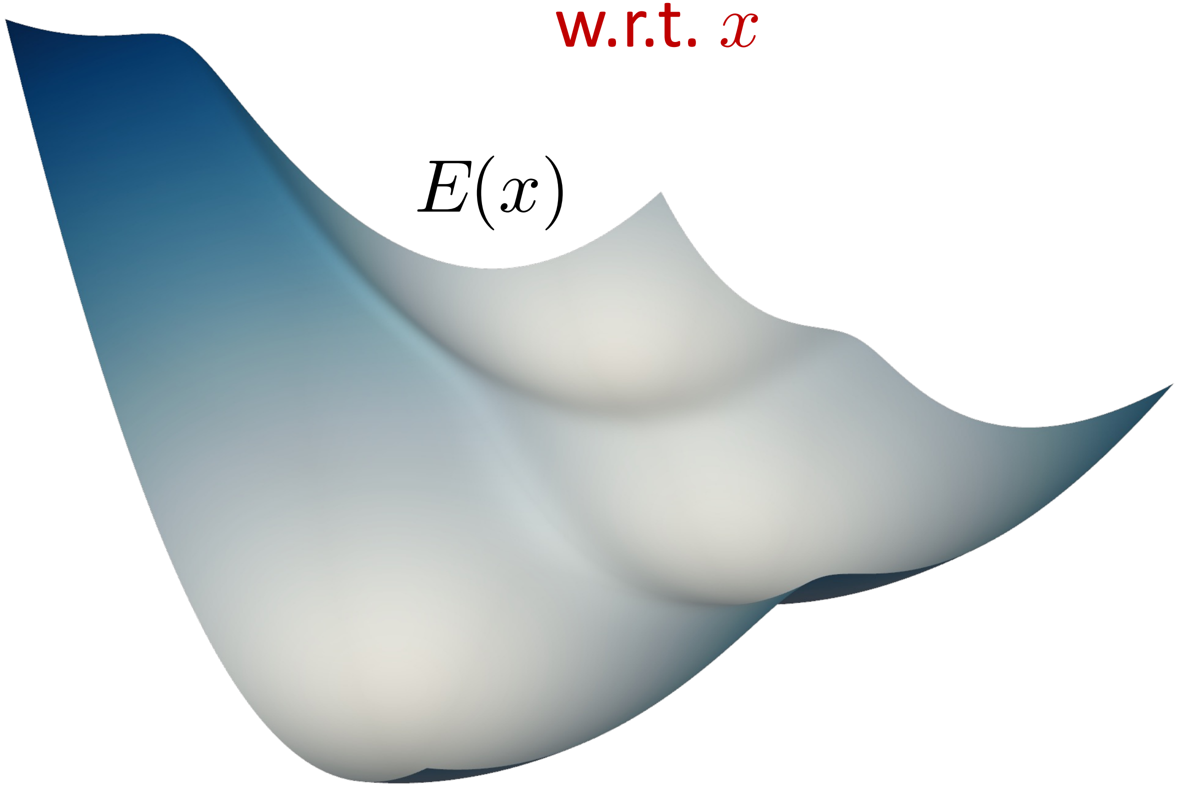
Energy-based Models

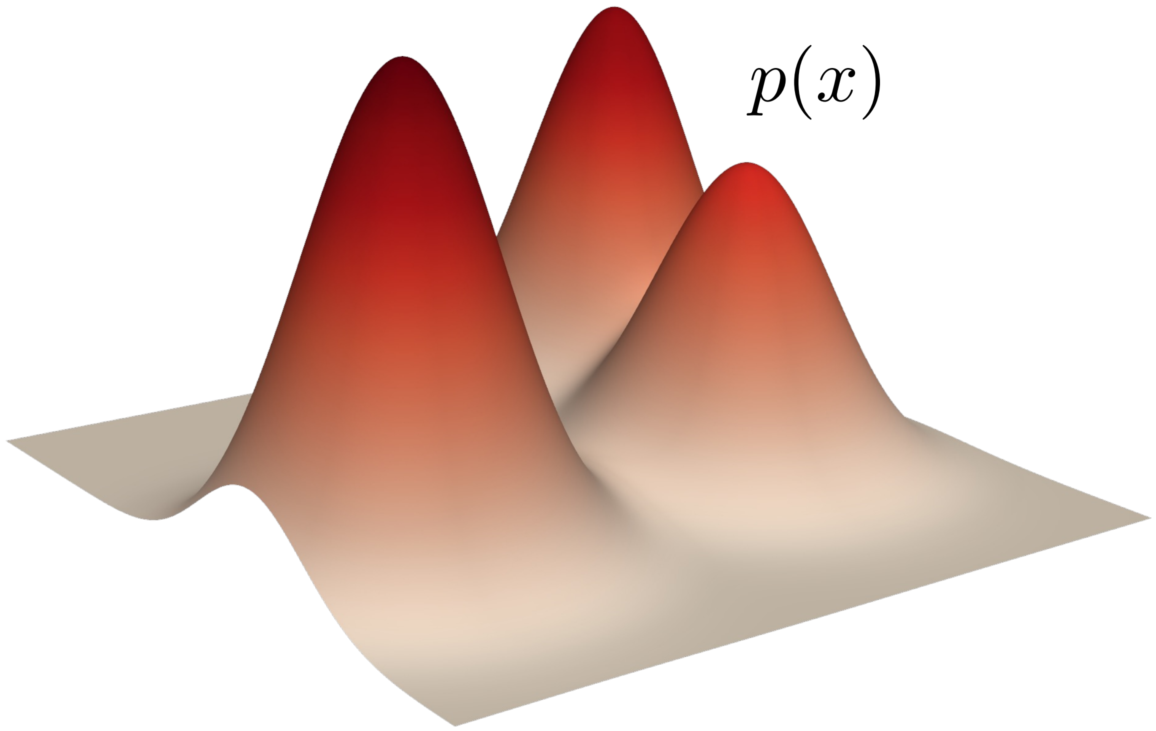
- **“Score function”**: gradient of log-probability

$$\nabla_x \log p(x) = -\nabla_x E(x) - \nabla_x \log Z$$

w.r.t. x

$=0$



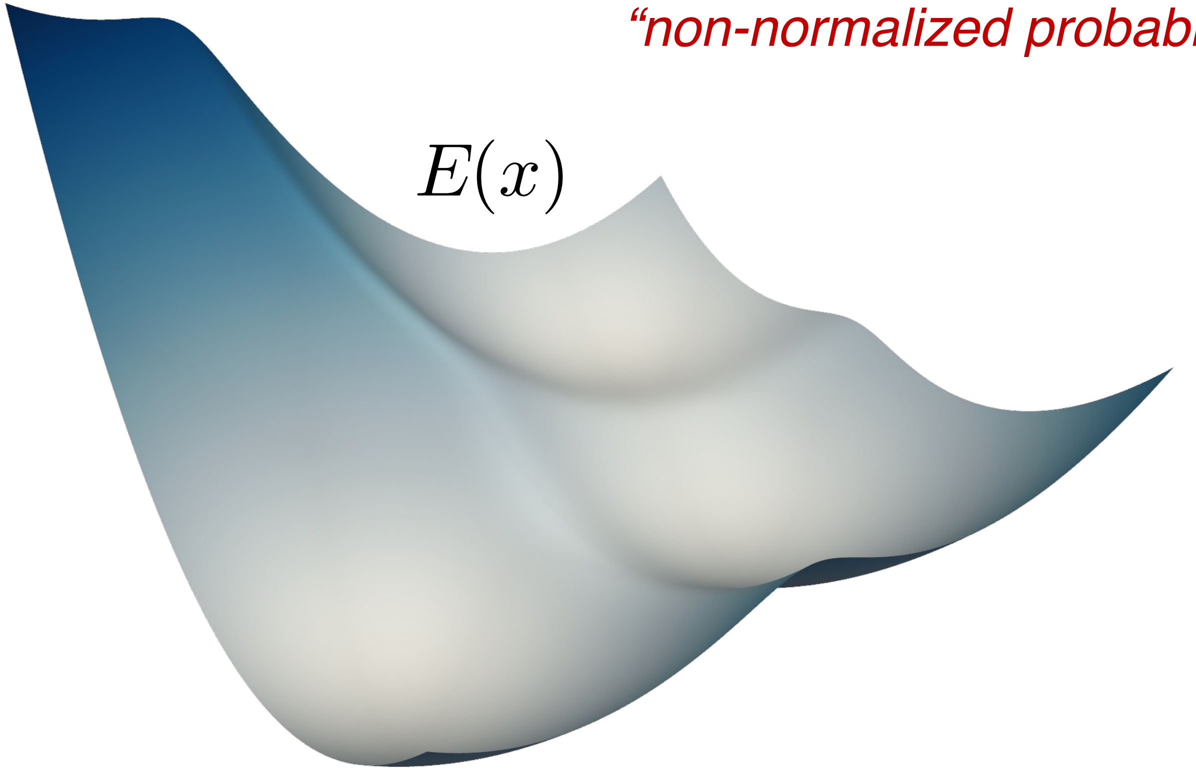


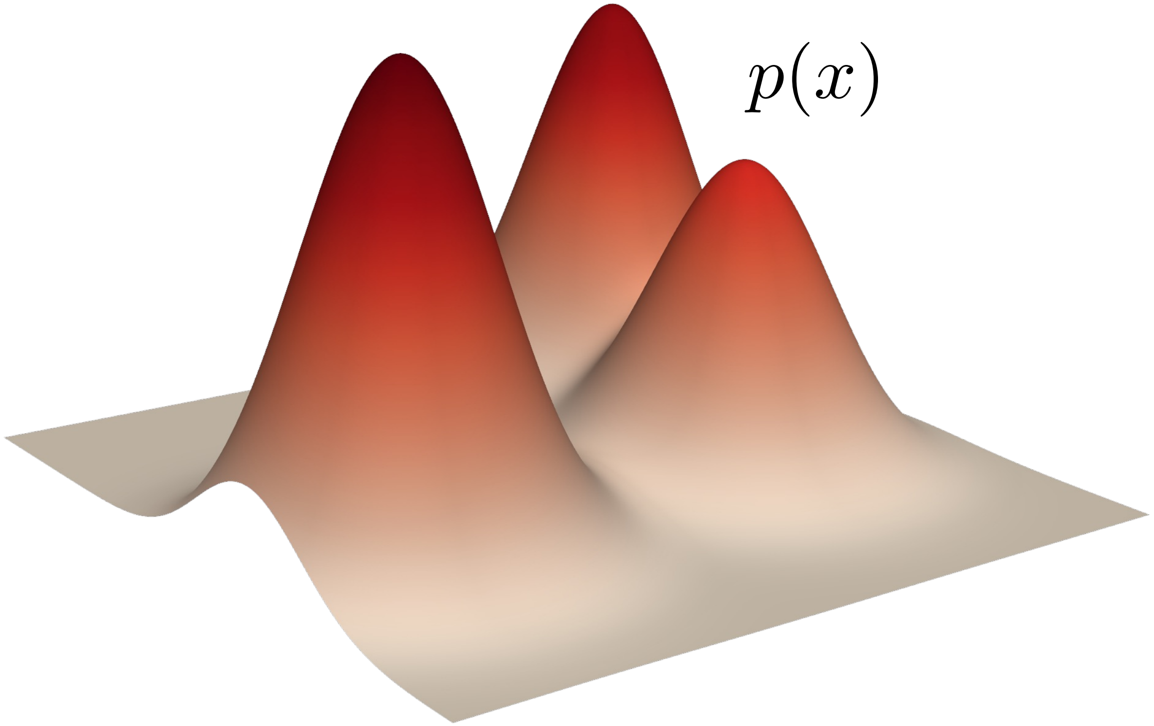
Energy-based Models

- **“Score function”**: gradient of log-probability

$$\nabla_x \log p(x) = -\nabla_x E(x)$$

“non-normalized probabilistic models”



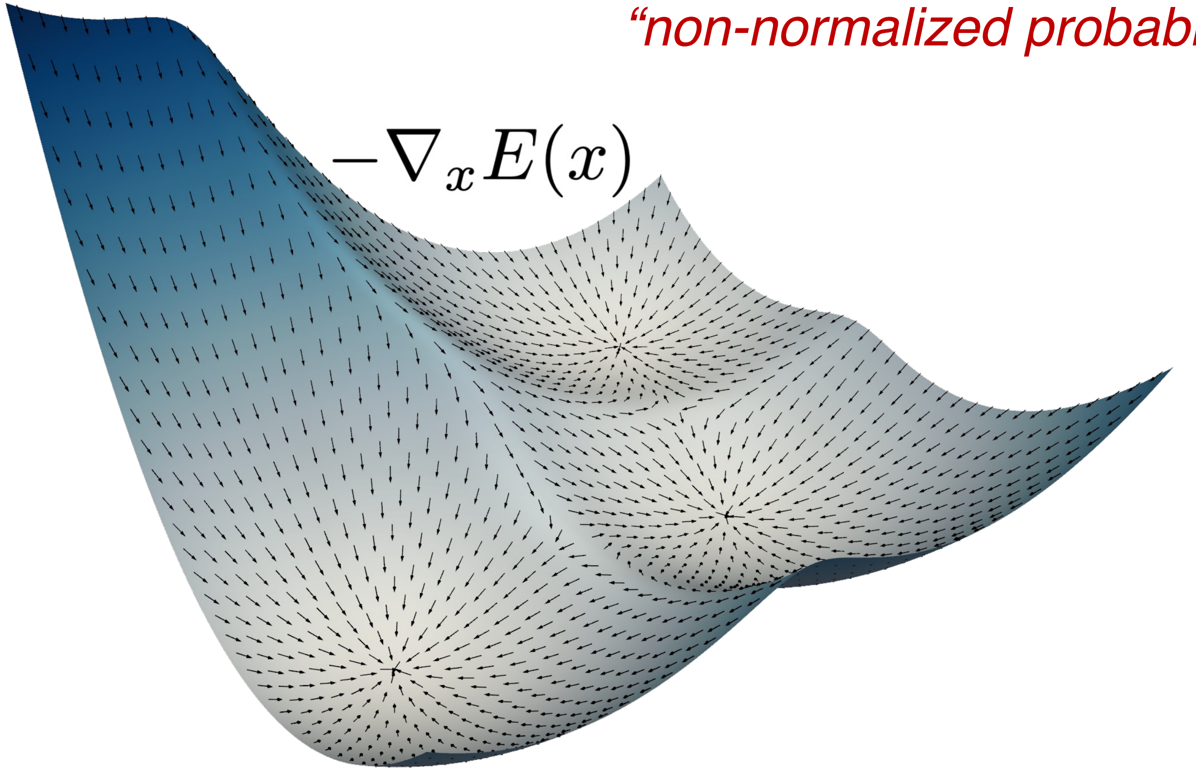


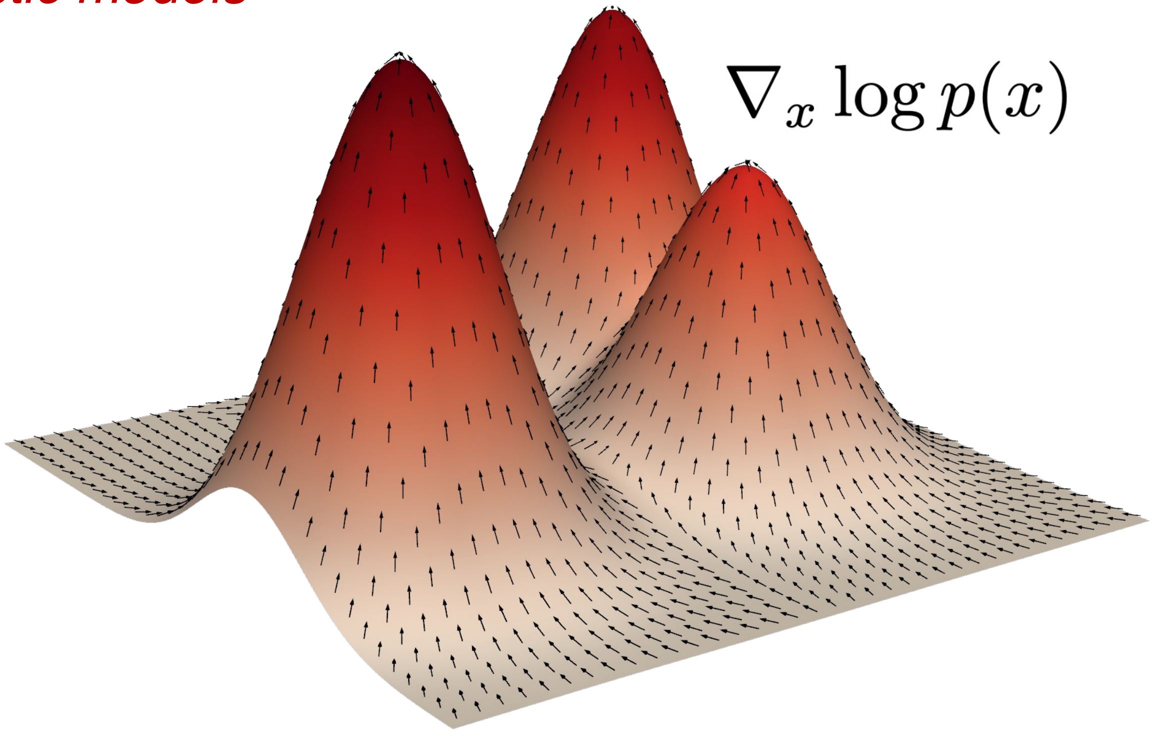
Energy-based Models

- **"Score function"**: gradient of log-probability

$$\nabla_x \log p(x) = -\nabla_x E(x)$$

"non-normalized probabilistic models"



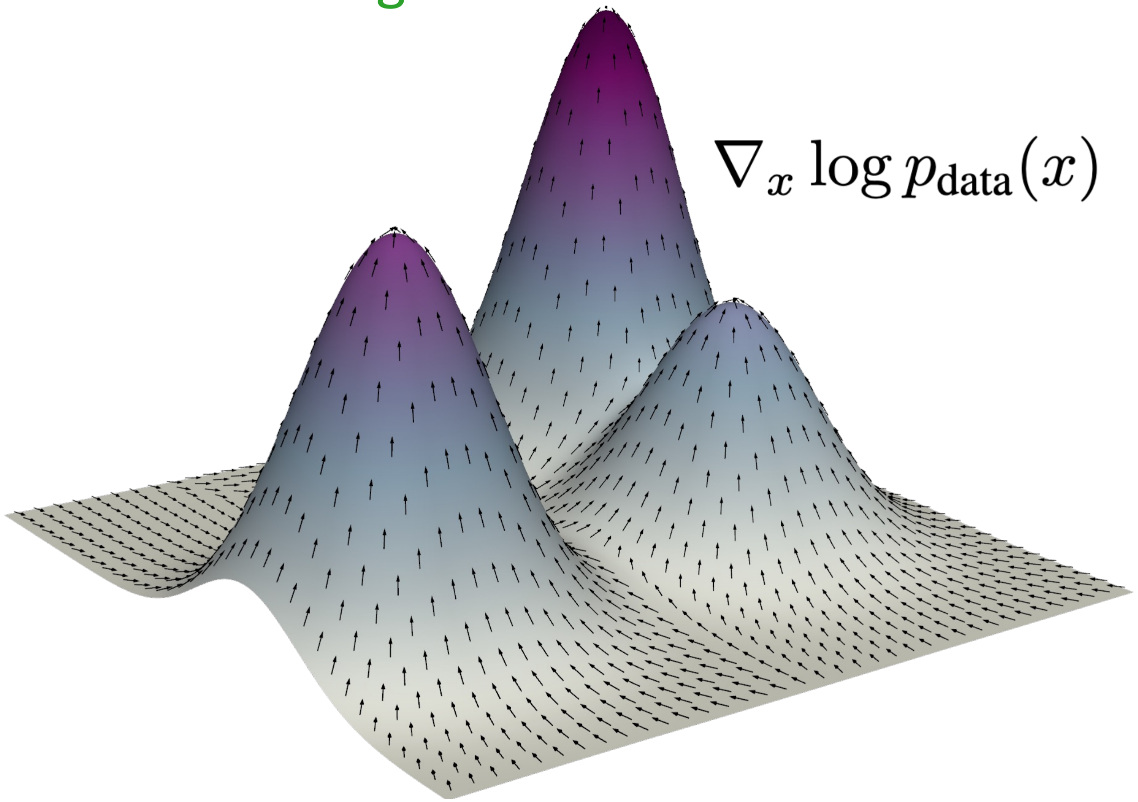


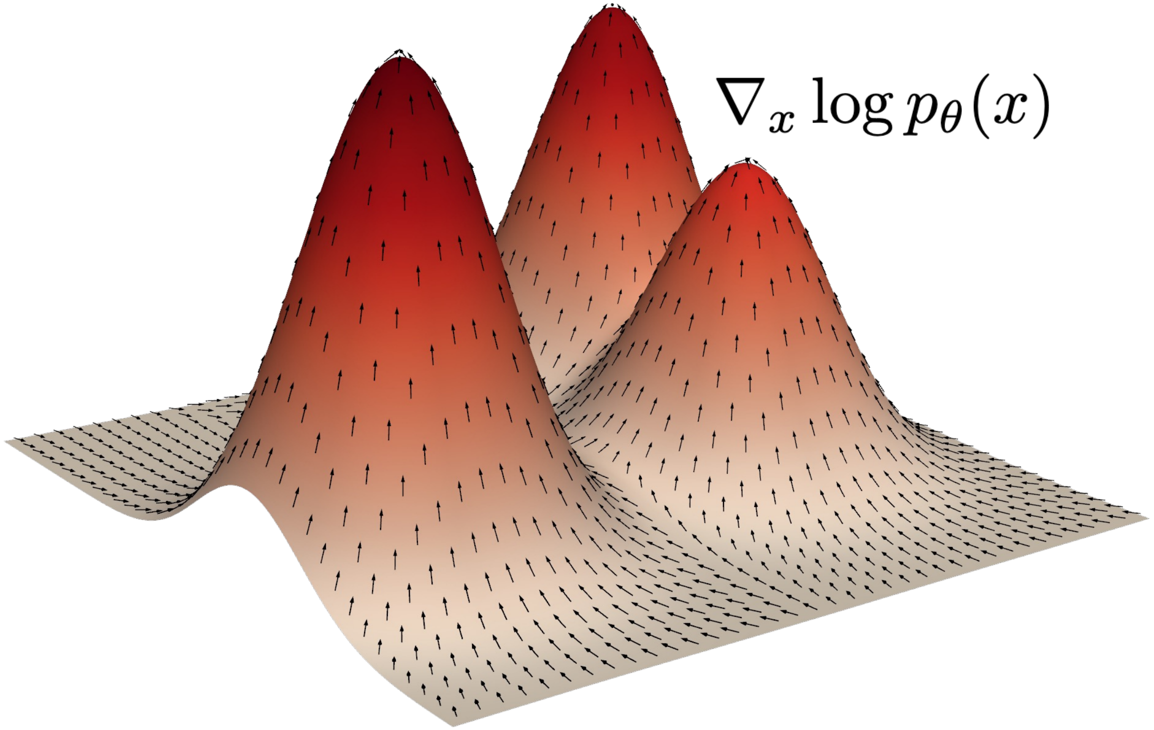
*only visualize directions

Score Matching

- Instead of parametrizing p, we can parametrize the score

$$\underbrace{D_F(p_{\text{data}}(x) \,\|\, p_\theta(x))}_{\text{Fisher divergence}} = \mathbb{E}_{p_{\text{data}}(x)}\left[\frac{1}{2}\|\underbrace{\nabla_x \log p_{\text{data}}(x)}_{\text{score of data}} - \underbrace{\nabla_x \log p_\theta(x)}_{\text{parameterized score}}\|^2\right]$$



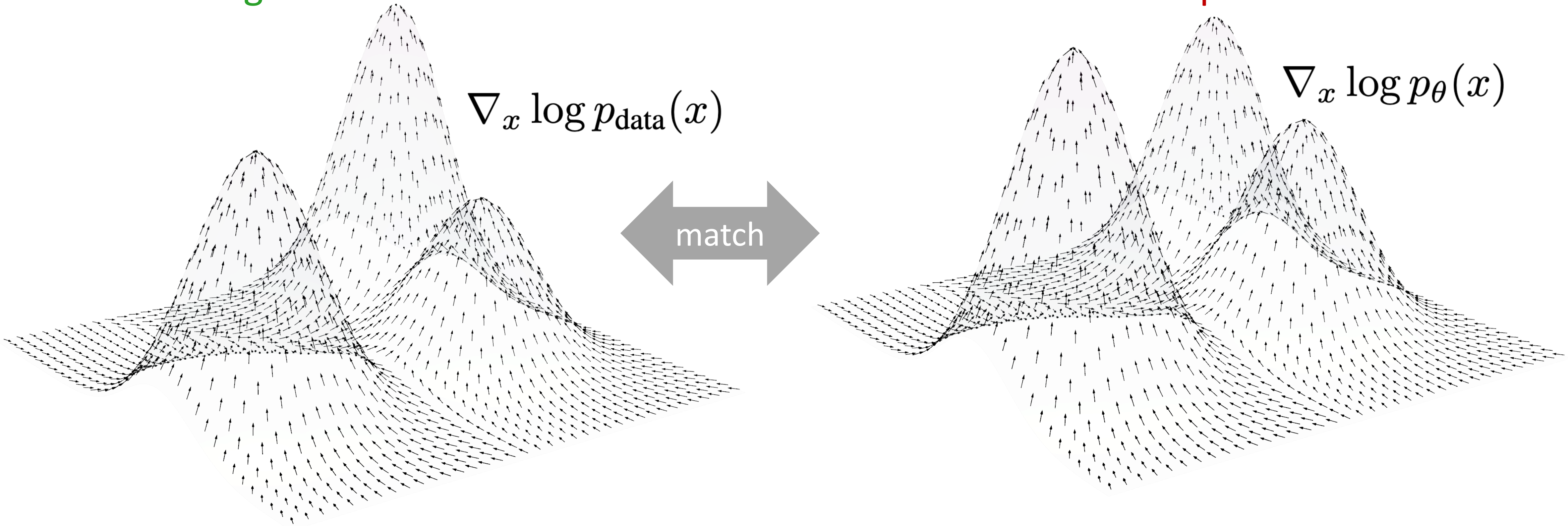


*only visualize directions

Score Matching

- Instead of parametrizing p, we can parametrize the score

$$\underbrace{D_F(p_{\text{data}}(x) \,\|\, p_\theta(x))}_{\text{Fisher divergence}} = \mathbb{E}_{p_{\text{data}}(x)}\left[\frac{1}{2}\left\|\underbrace{\nabla_x \log p_{\text{data}}(x)}_{\text{score of data}} - \underbrace{\nabla_x \log p_\theta(x)}_{\text{parameterized score}}\right\|^2\right]$$



*only visualize directions

Denoising Score Matching

- with noised data $\tilde{x} := x + \epsilon$, it can be proven: [Vincent, 2011]

$$\underbrace{D_F(q(\tilde{x}) \parallel p_\theta(\tilde{x}))}_{\text{Fisher divergence of noised data}} = \mathbb{E}_{\underbrace{q(x,\tilde{x})}_{\text{joint distribution}}} \left[\frac{1}{2} \left\| \underbrace{\nabla_{\tilde{x}} \log q(\tilde{x} \mid x)}_{\text{score of conditional}} - \underbrace{\nabla_{\tilde{x}} \log p_\theta(\tilde{x})}_{\text{parameterized score}} \right\|^2 \right] + \text{constant}$$

See: Vincent, "A Connection Between Score Matching and Denoising Autoencoders", Neural Computation, 2011

Denoising Score Matching

- with noised data $\tilde{x} := x + \epsilon$, it can be proven: [Vincent, 2011]

$$D_F(q(\tilde{x}) \parallel p_\theta(\tilde{x})) = \mathbb{E}_{q(x,\tilde{x})}\left[\frac{1}{2}\left\|\nabla_{\tilde{x}} \log q(\tilde{x} \mid x) - \nabla_{\tilde{x}} \log p_\theta(\tilde{x})\right\|^2\right] + \text{constant}$$

Gaussian noise:

$$= \frac{1}{\sigma^2}\underbrace{(x - \tilde{x})}_{-\epsilon}$$

a network to predict (negative) noise

Langevin Dynamics

- Given a score function, we can sample x from p by iterating:

$$x_t \leftarrow x_{t-1} + \underbrace{\frac{\sigma^2}{2}}_{\text{step size}} \underbrace{\nabla_x \log p_\theta(x_{t-1})}_{\text{score function}} + \sigma z_t$$

step size

score function

(don't need to know p)

$\mathcal{N}(0, \mathbf{I})$

(neg) gradient of energy

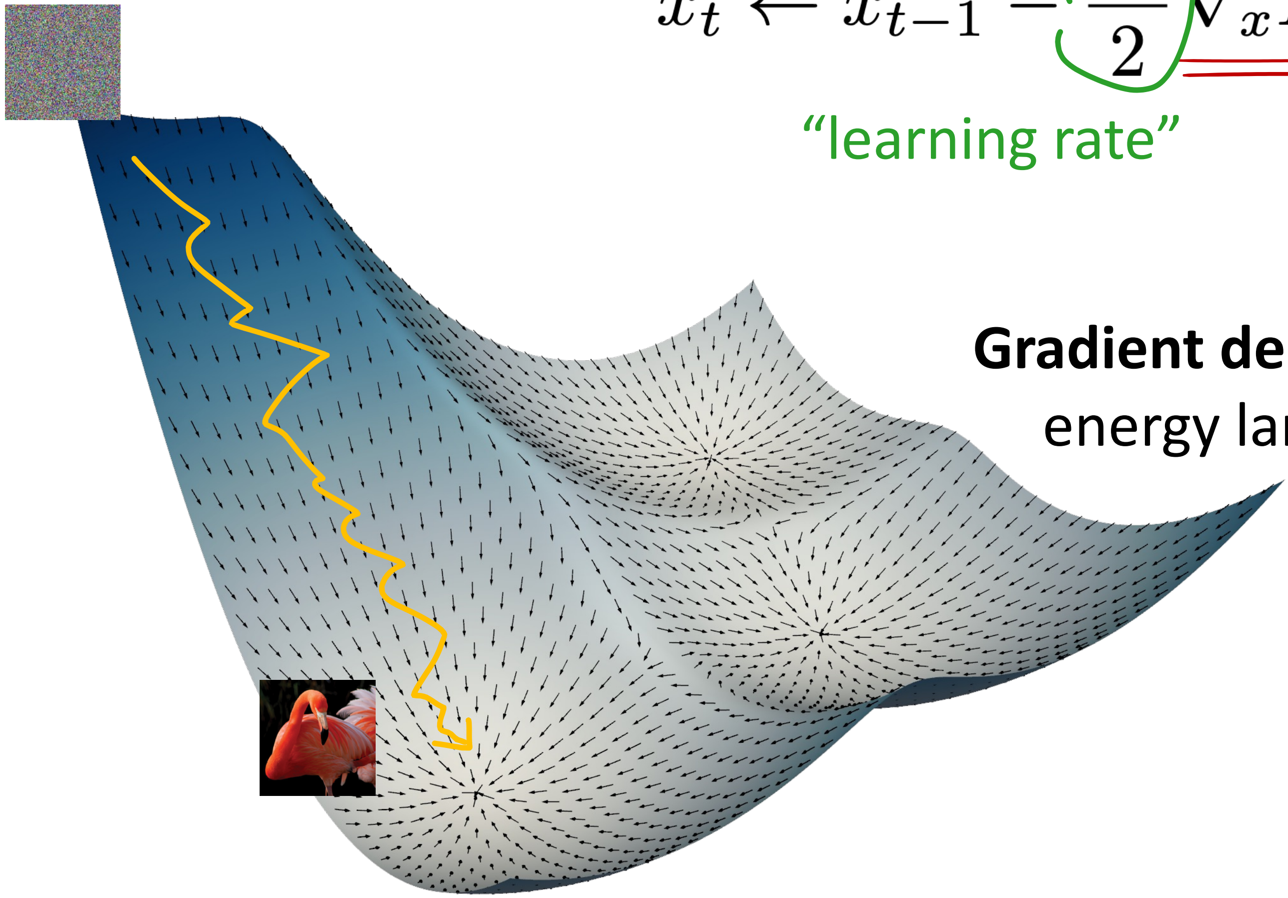
$$-\nabla_x E_\theta(x_{t-1})$$

Langevin Dynamics

- Given a score function, we can sample x from p by iterating:

$$x_t \leftarrow x_{t-1} - \frac{\sigma^2}{2}\nabla_x E_\theta(x_{t-1}) + \sigma z_t$$

"learning rate" gradient perturbation



Gradient decent in the energy landscape

(Recap) Diffusion algorithm

Algorithm 1 Training

repeat

  $\mathbf{x}_0 \sim q(\mathbf{x}_0)$

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

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

  Take gradient descent step on

$$\nabla_\theta \left\| \boldsymbol{\epsilon} - \boldsymbol{\epsilon}_\theta(\sqrt{\bar{\alpha}_t}\mathbf{x}_0 + \sqrt{1-\bar{\alpha}_t}\boldsymbol{\epsilon}, t) \right\|^2$$

until converged

score function

Algorithm 2 Sampling

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

for $t = T, \ldots, 1$ **do**

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

$$\mathbf{x}_{t-1} = \frac{1}{\sqrt{\alpha_t}}\left(\mathbf{x}_t - \frac{1-\alpha_t}{\sqrt{1-\bar{\alpha}_t}}\boldsymbol{\epsilon}_\theta(\mathbf{x}_t, t)\right) + \sigma_t \mathbf{z}$$

end for

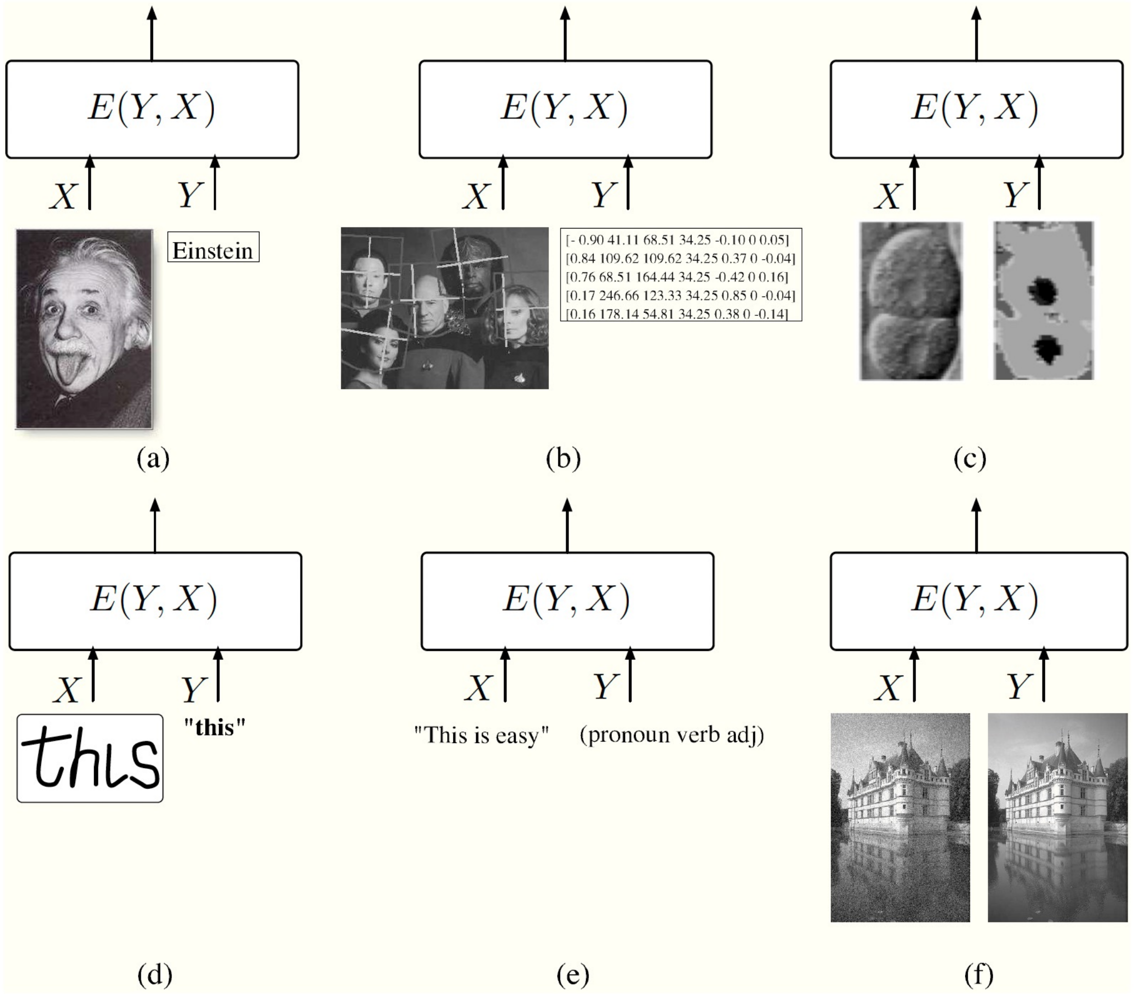
return $\mathbf{x}_0$

score function

Langevin Dynamics

More about Energy-based Models ...

- At <u>inference</u> time, find a solution that minimizes energy



LeCun et al., "A Tutorial on Energy-Based Learning", 2006

Various Perspectives on Diffusion Models ...

- Hierarchical VAE
- Energy-based Models and Score Matching
- Autoregressive models

> be a fully expressive conditional distribution. With these choices, $D_{\mathrm{KL}}(q(\mathbf{x}_T) \parallel p(\mathbf{x}_T)) = 0$, and minimizing $D_{\mathrm{KL}}(q(\mathbf{x}_{t-1}|\mathbf{x}_t) \parallel p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t))$ trains p_θ to copy coordinates $t+1, \ldots, T$ unchanged and to predict the t^{th} coordinate given $t+1, \ldots, T$. Thus, training p_θ with this particular diffusion is training an autoregressive model.

[Ho et al, 2020]

- SDE and ODE
- Normalizing Flows
- Recurrent Neural Networks

See also: Dieleman, "Perspectives on diffusion". https://sander.ai/2023/07/20/perspectives.html

This Lecture

- Diffusion Models

- Energy-based Models and Score Matching

Main References

- Sohl-Dickstein et al. “Deep Unsupervised Learning using Nonequilibrium Thermodynamics”, ICML 2015
- Ho et al. “Denoising Diffusion Probabilistic Models”, NeurIPS 2019

- Hyvärinen. “Estimation of non-normalized statistical models by score matching”, JMLR 2005.
- Song & Ermon. “Generative Modeling by Estimating Gradients of the Data Distribution”, NeurIPS 2019
- Song & Kingma. “How to Train Your Energy-Based Models”, arXiv 2021