

An introduction to generative modeling *with applications in physics*

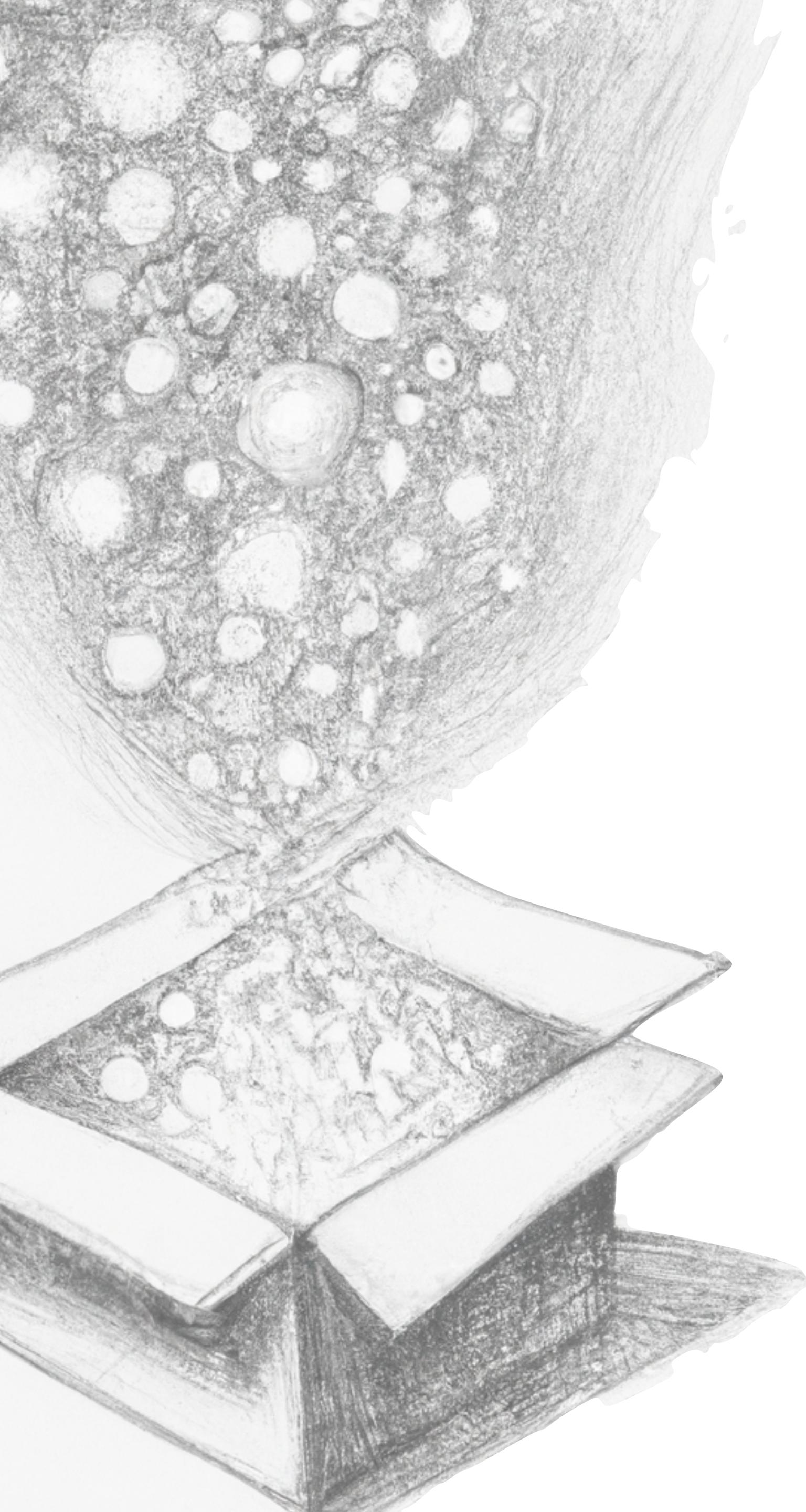
smsharma.io/iaifi-summer-school-2023

Siddharth Mishra-Sharma
smsharma.io/@kdqg1



NSF AI Institute for Artificial Intelligence
and Fundamental Interactions (IAIFI)

IAIFI Summer School
August 8, 2023



An introduction to generative modeling *with applications in physics*

smsharma.io/iaifi-summer-school-2023

Siddharth Mishra-Sharma
smsharma.io/@kdqg1



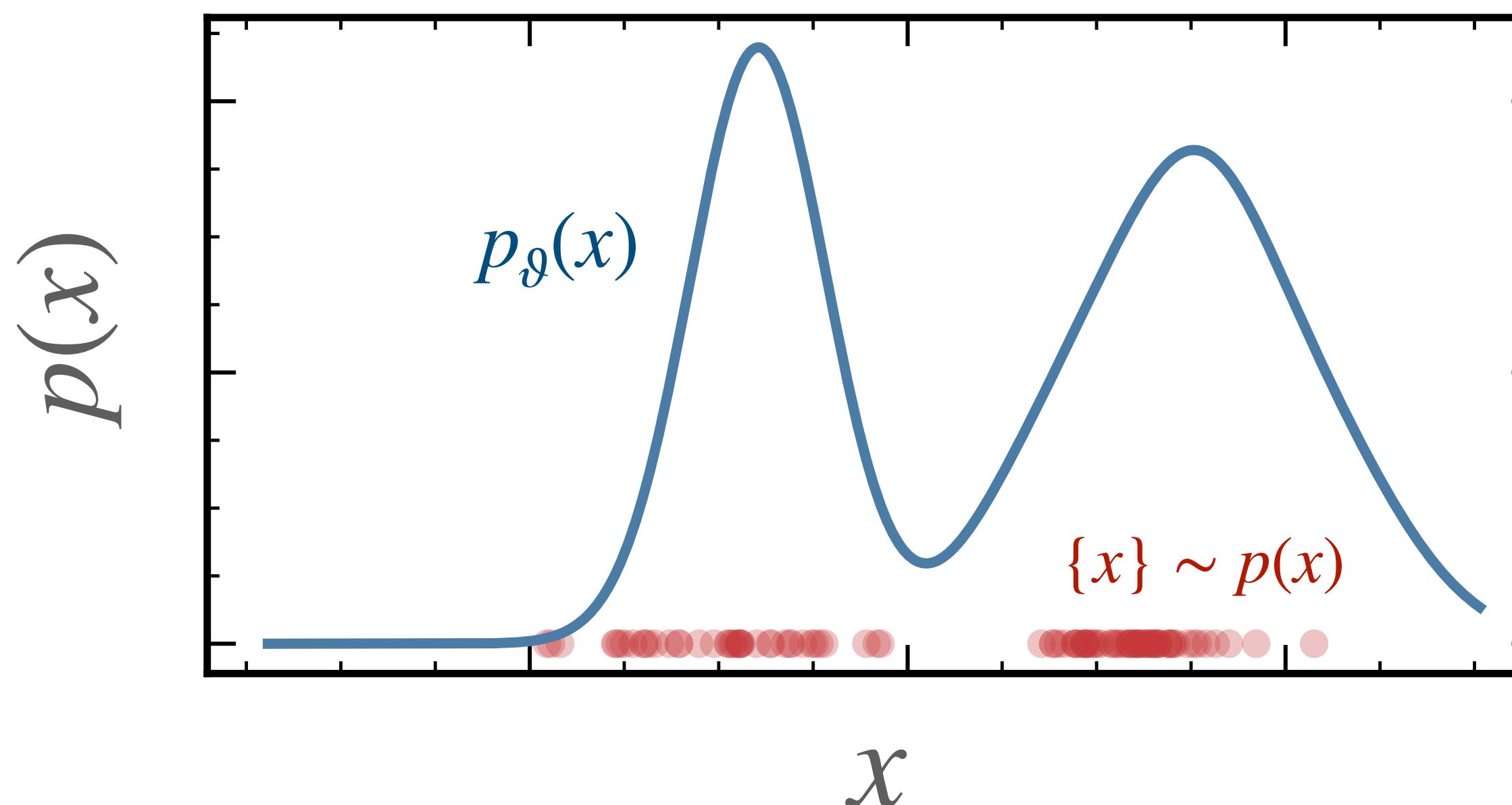
NSF AI Institute for Artificial Intelligence
and Fundamental Interactions (IAIFI)

IAIFI Summer School
August 8, 2023

Generative models

Generative models are simulators of the data

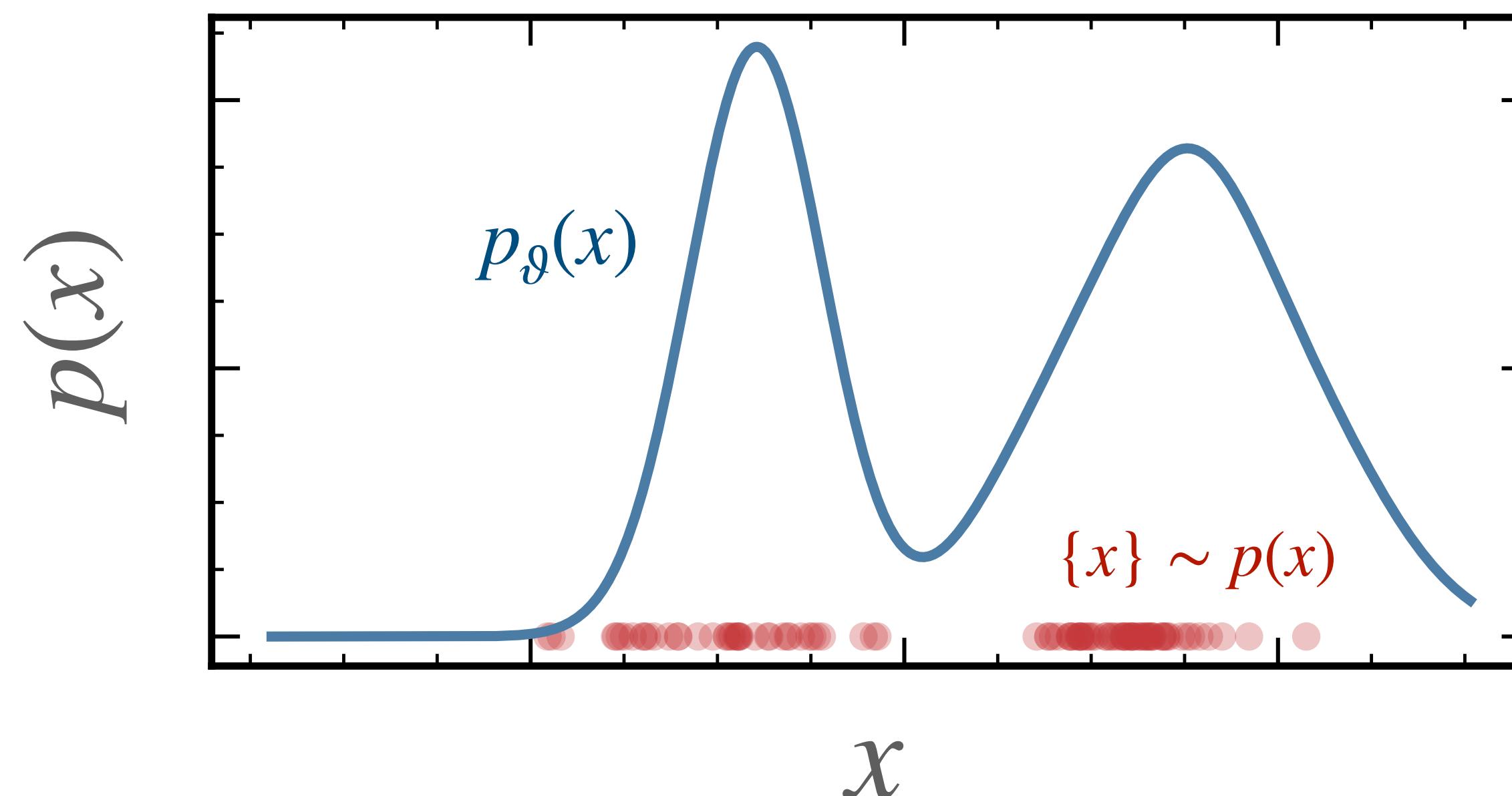
Goal: learn a probability distribution $p_\theta(x)$ that is as close as possible to the true underlying data distribution $p(x)$



Generative models

Generative models are simulators of the data

Goal: learn a probability distribution $p_\vartheta(x)$ that is as close as possible to the true underlying data distribution $p(x)$



1. *Sampling*

$$x \sim p_\vartheta(x)$$

2. *Density estimation*

$$\log p_\vartheta(x)$$

Evolution of deep generative models



Variational autoencoders
(from Kingma et al 2013)

Evolution of deep generative models



Variational autoencoders
(from Kingma et al 2013)

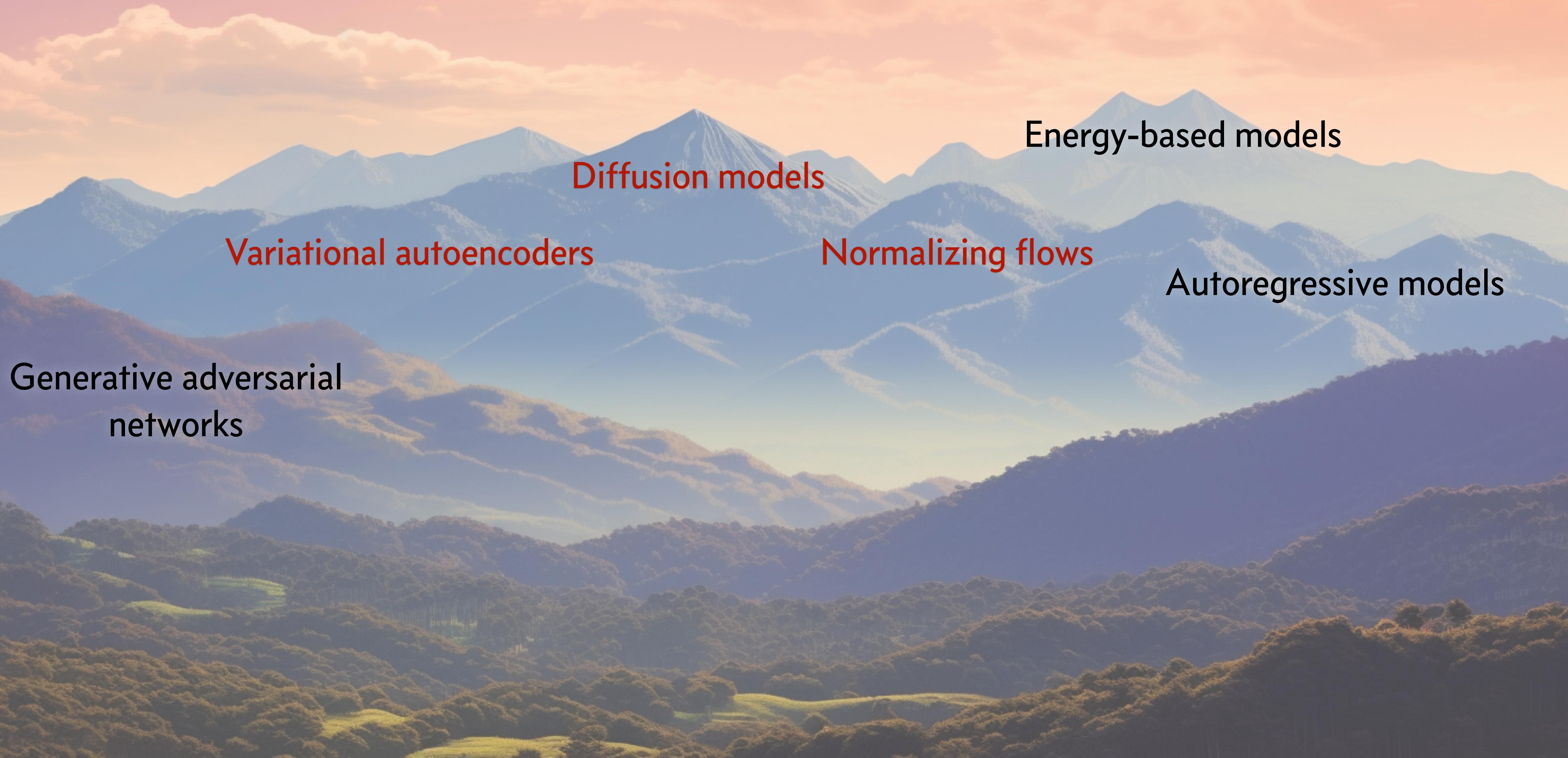


Diffusion models
(Midjourney 2023)



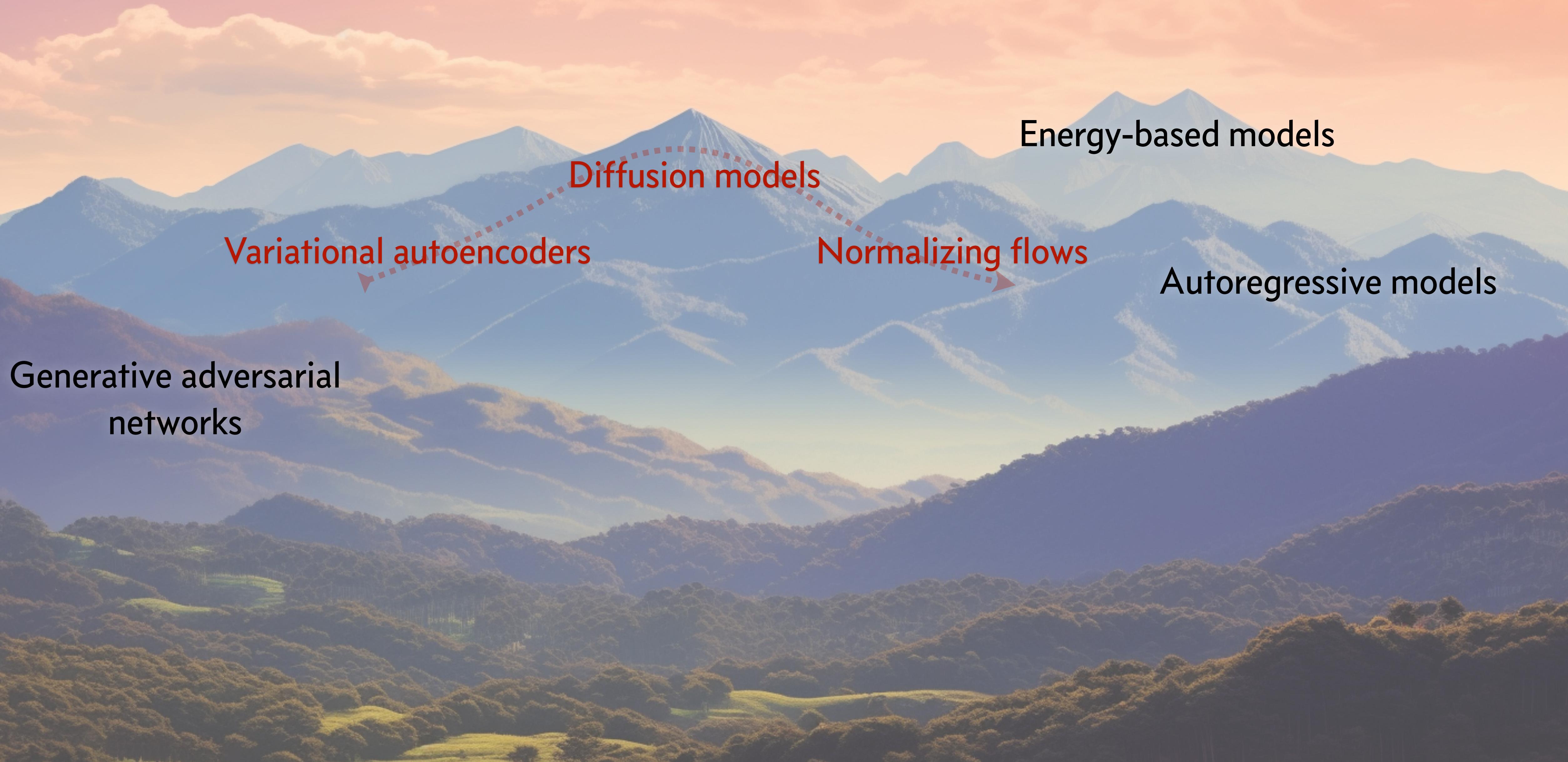
The landscape of deep generative models

[Karsten Kreis; [CVPR 2022 Tutorial](#)]

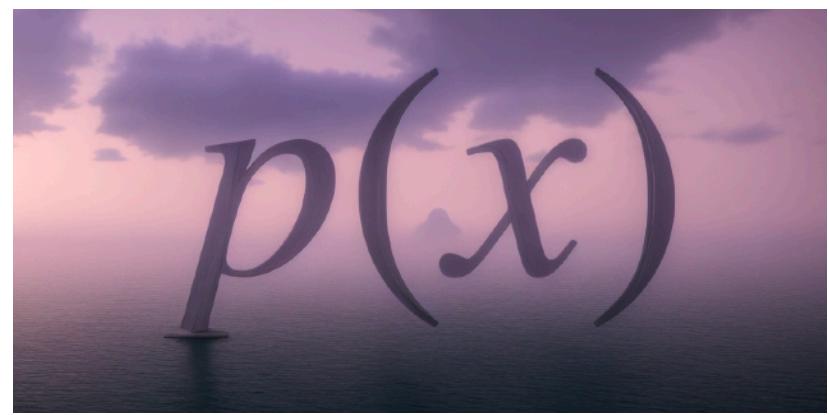


The landscape of deep generative models

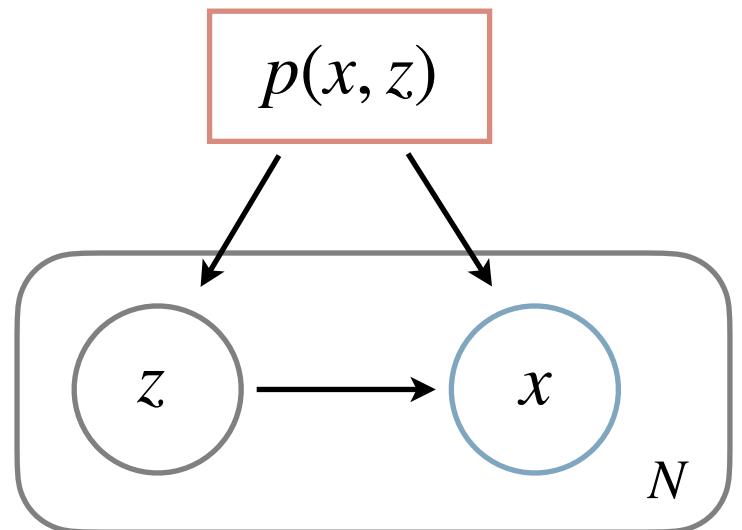
[Karsten Kreis; [CVPR 2022 Tutorial](#)]



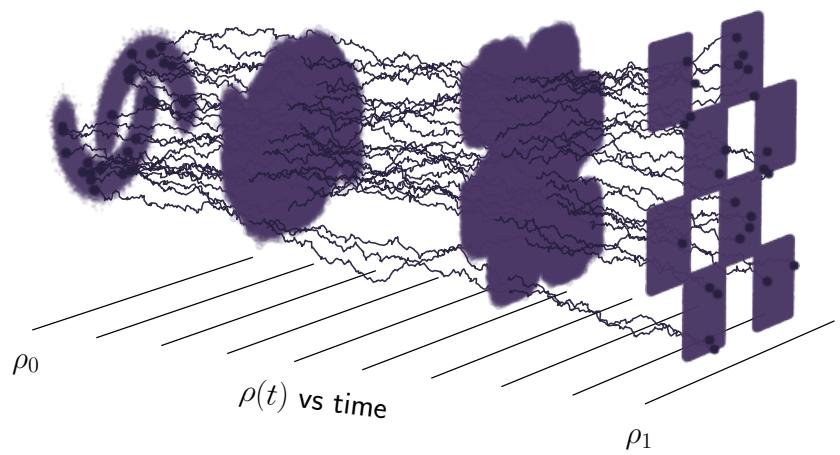
Outline



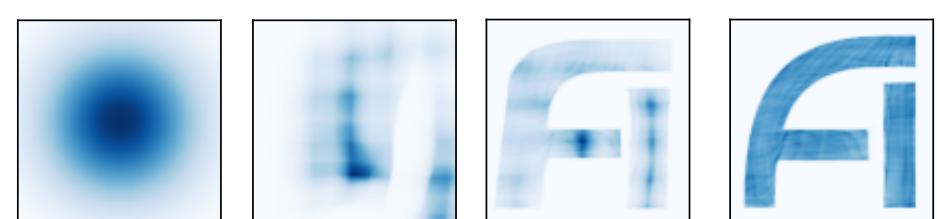
Why (deep) generative modeling?
What is it, and what can it do for you?



Variational auto encoders
Latent-variable modeling, and compression is all you need

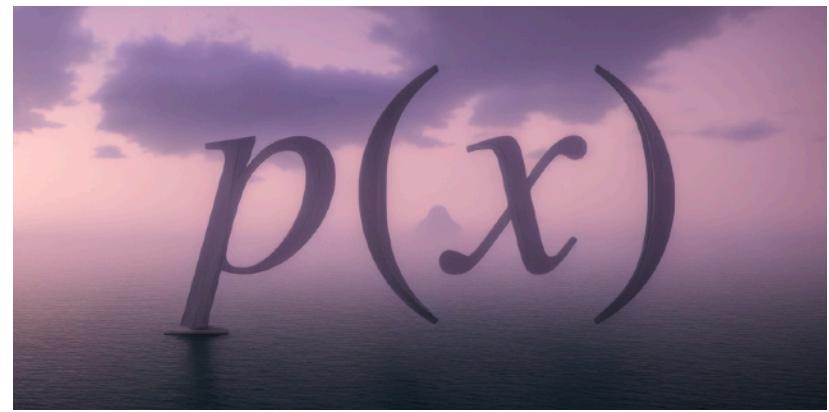


Diffusion models
Models based on iterative refinement

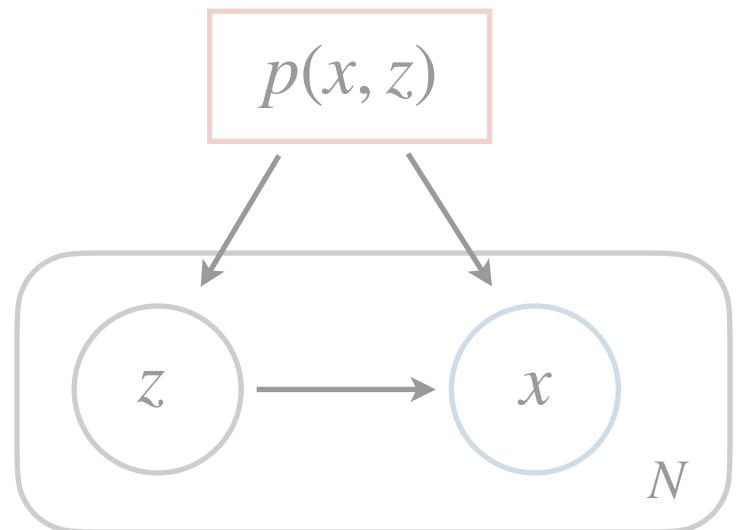


Normalizing flows
Invertible transformations

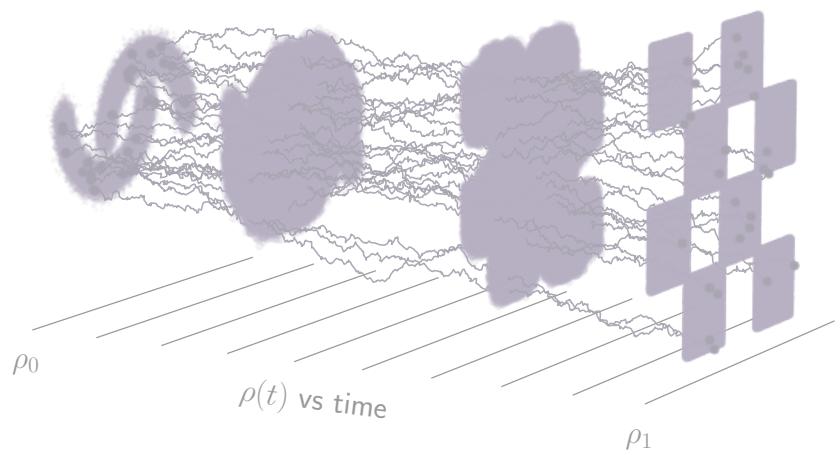
Outline



Why (deep) generative modeling?
What is it, and what can it do for you?



Variational auto encoders
Latent-variable modeling, and compression is all you need



Diffusion models
Models based on iterative refinement



Normalizing flows
Invertible transformations

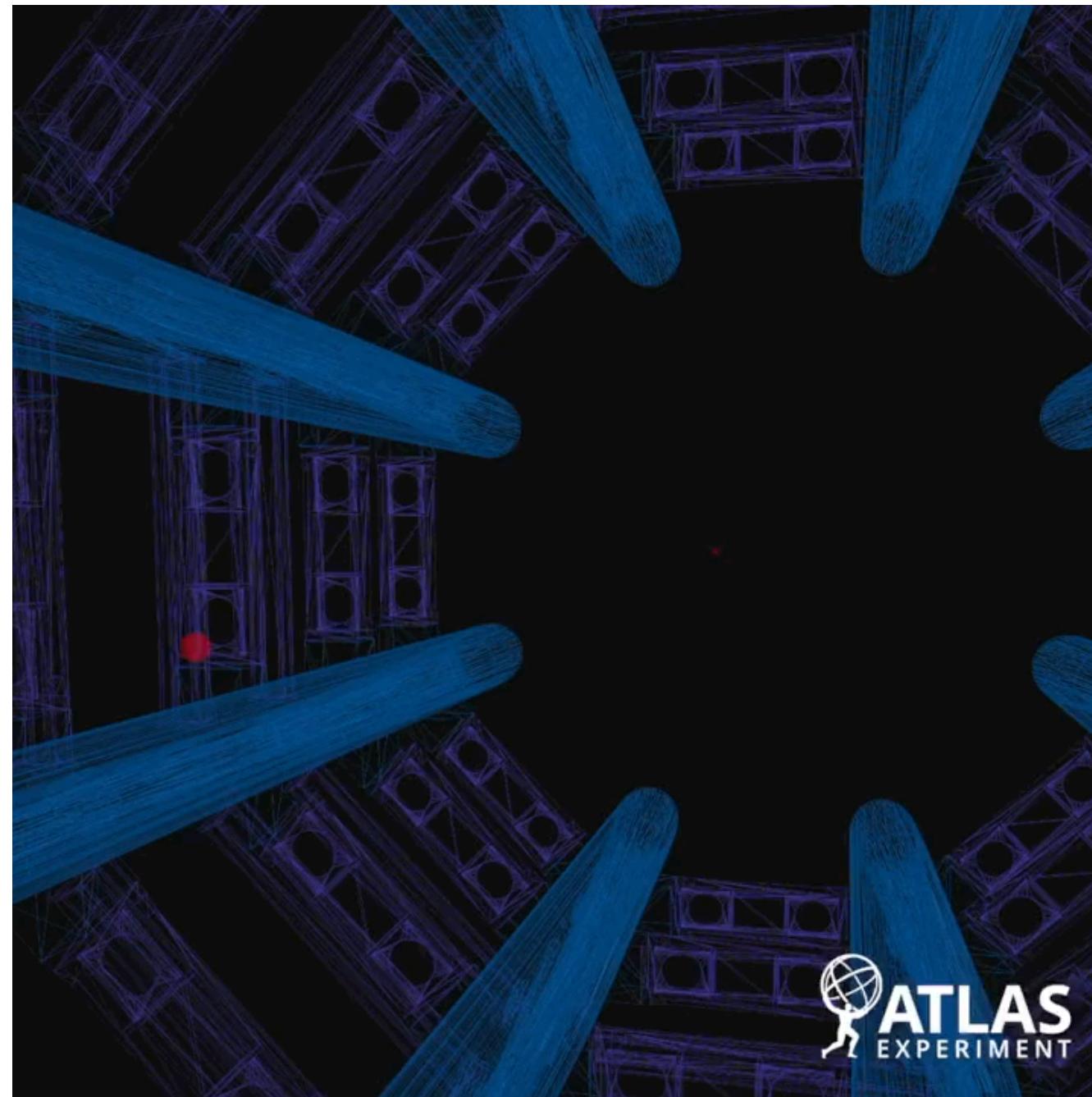
Simulators

$$x \sim p(x)$$

Simulators are ubiquitous: *they prescribe a way to sample from the data distribution*

Collider data

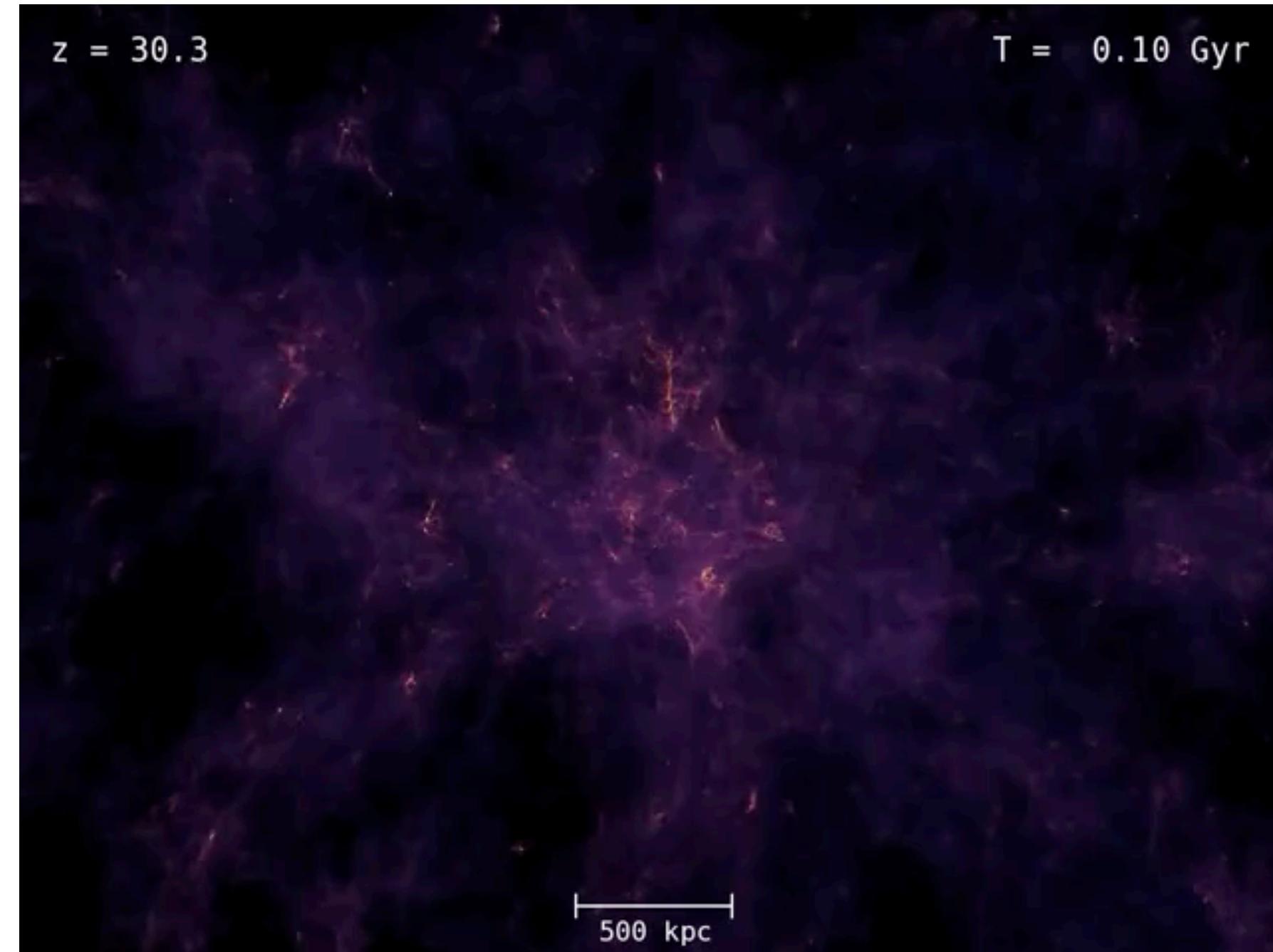
particles $\sim p(\text{particles})$



[C. Cesarotti with ATLAS]

Cosmology data

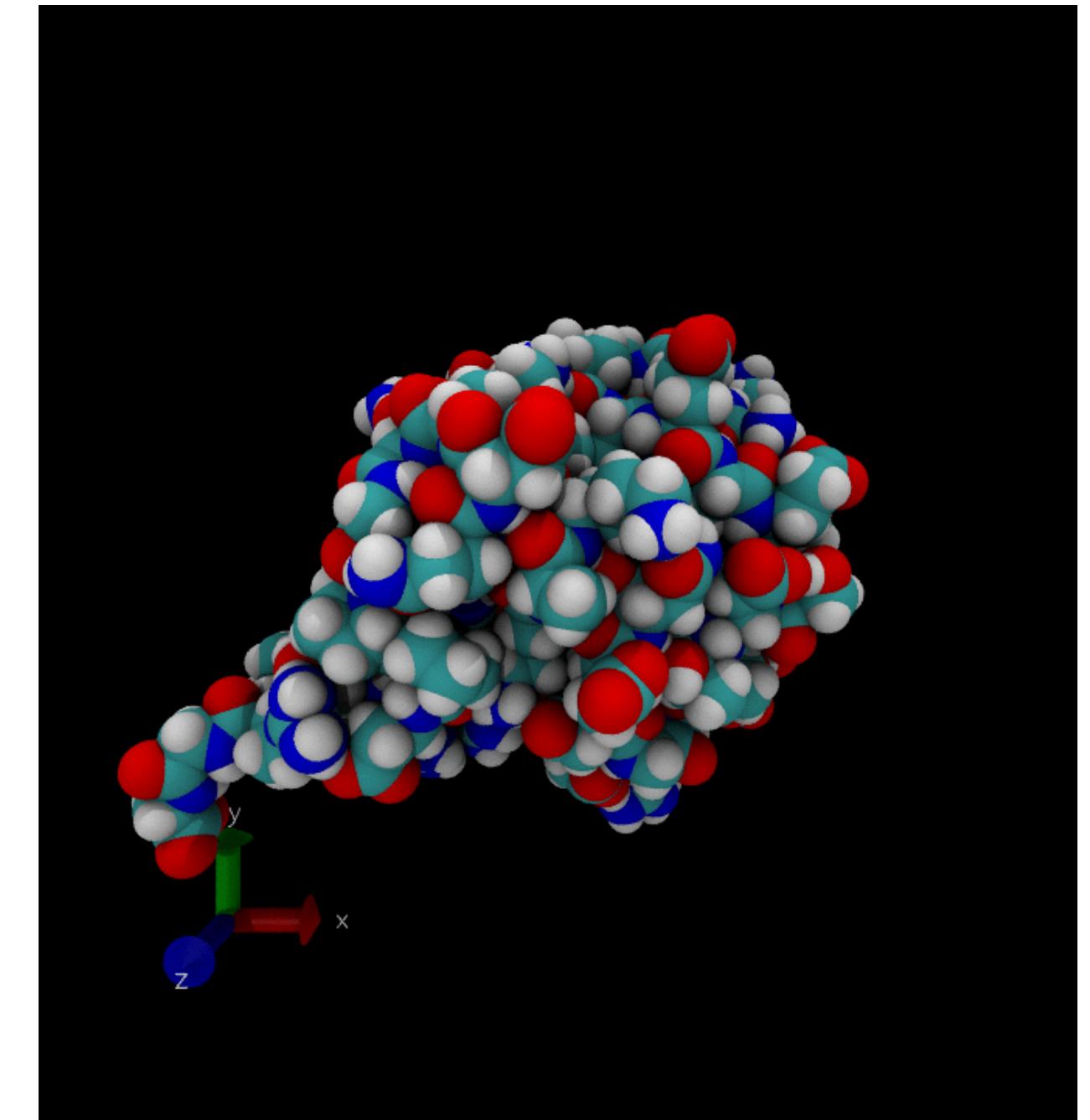
particles $\sim p(\text{particles})$



[Aquarius simulation]

Molecular dynamics

configurations $\sim p(\text{configurations})$



[E. Cancès et al.]

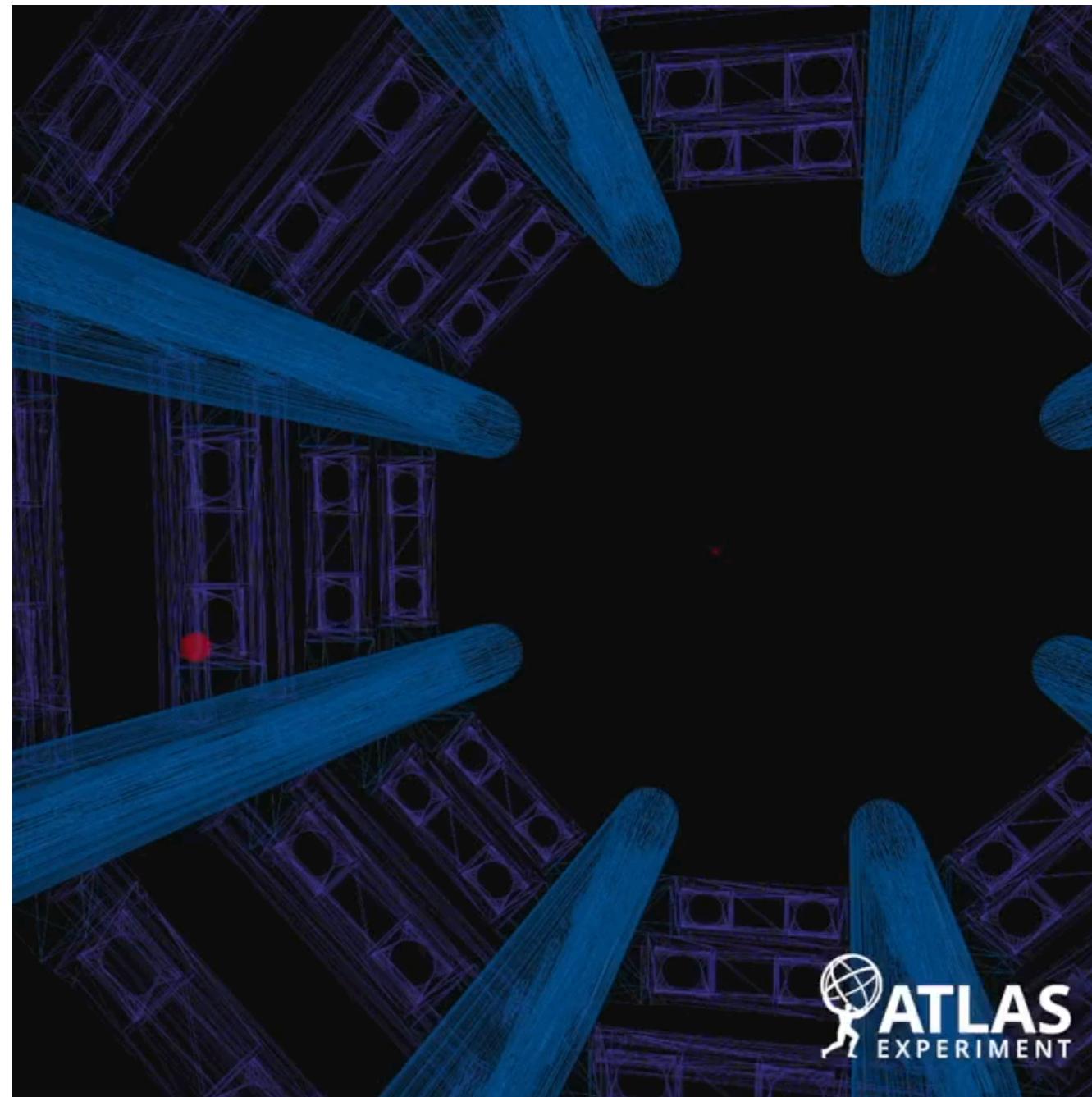
Simulators

$$x \sim p(x)$$

Simulators are ubiquitous: *they prescribe a way to sample from the data distribution*

Collider data

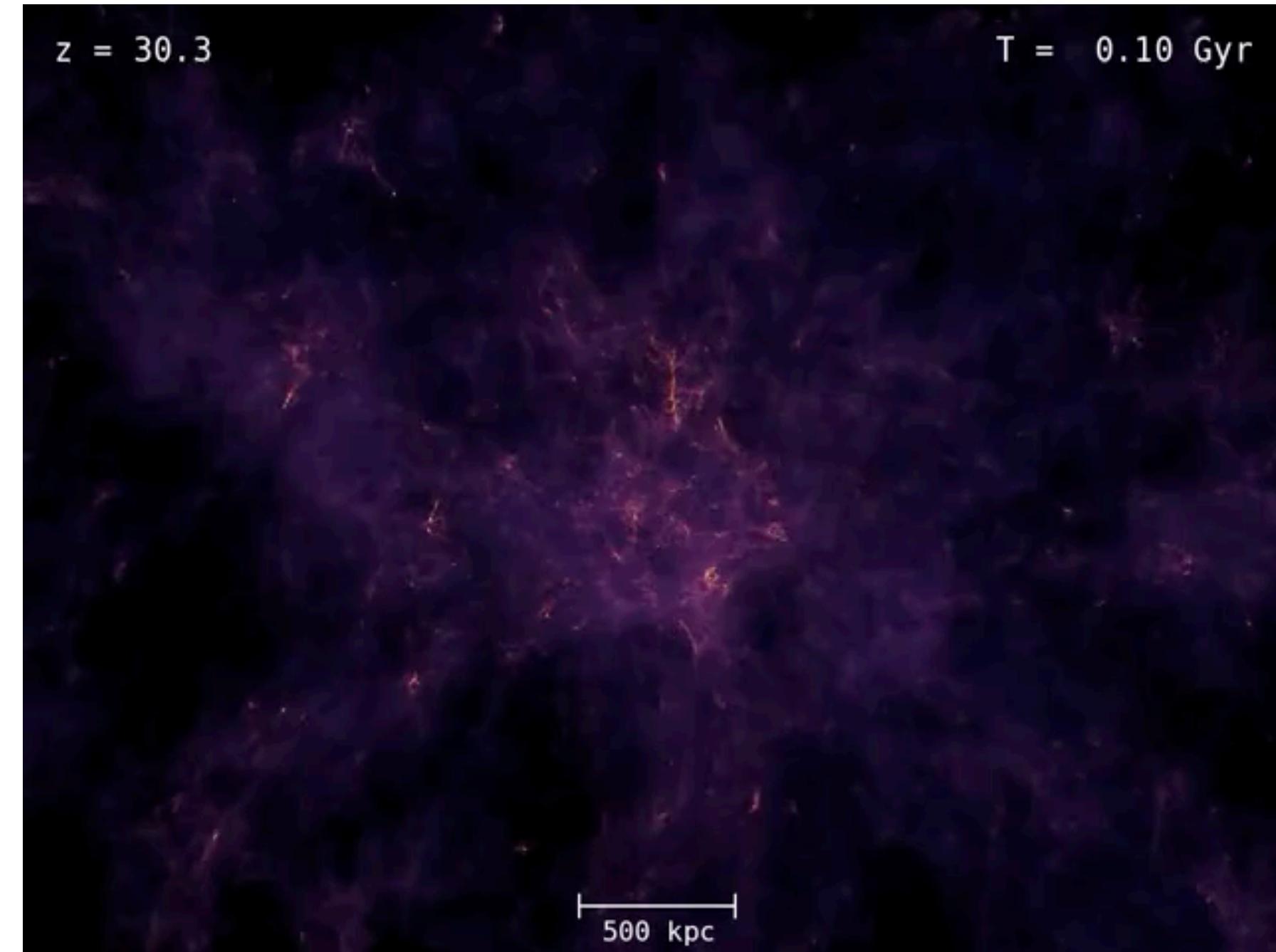
particles $\sim p(\text{particles})$



[C. Cesarotti with ATLAS]

Cosmology data

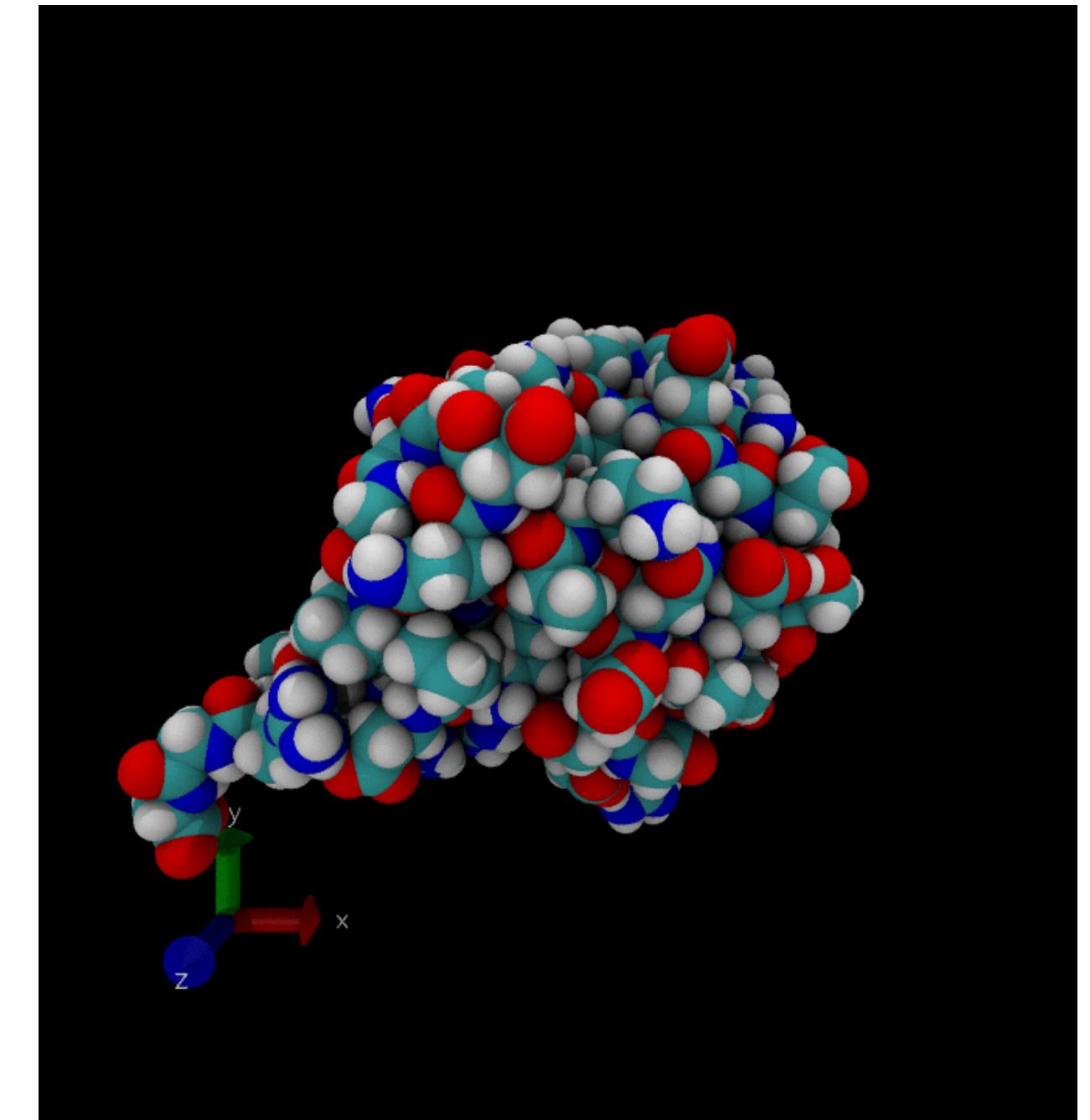
particles $\sim p(\text{particles})$



[Aquarius simulation]

Molecular dynamics

configurations $\sim p(\text{configurations})$



[E. Cancès et al.]

Simulators

$$x \sim p(x)$$

Simulators are ubiquitous: *they prescribe a way to sample from the data distribution*

Collider data

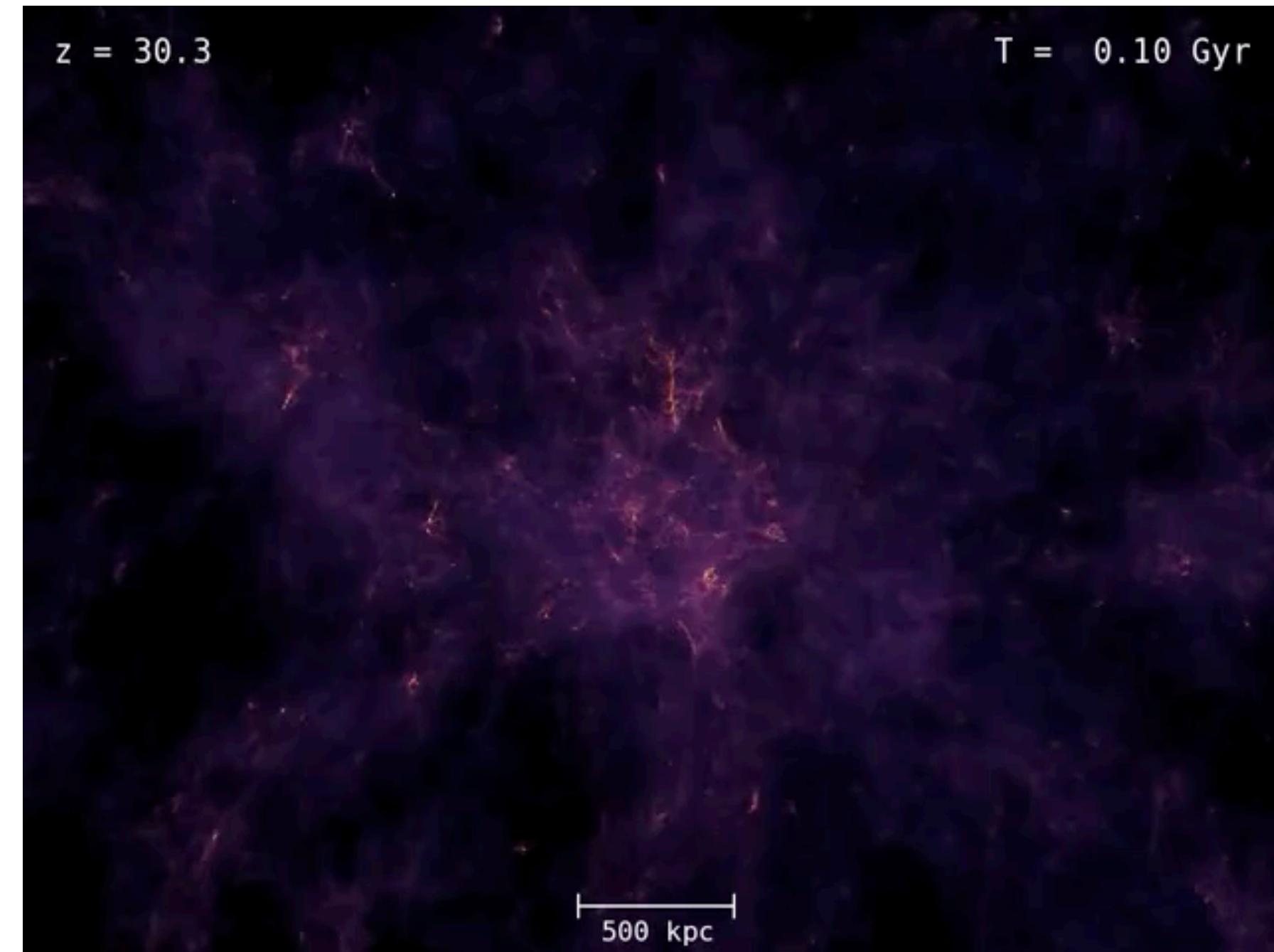
particles $\sim p(\text{particles})$



[C. Cesarotti with ATLAS]

Cosmology data

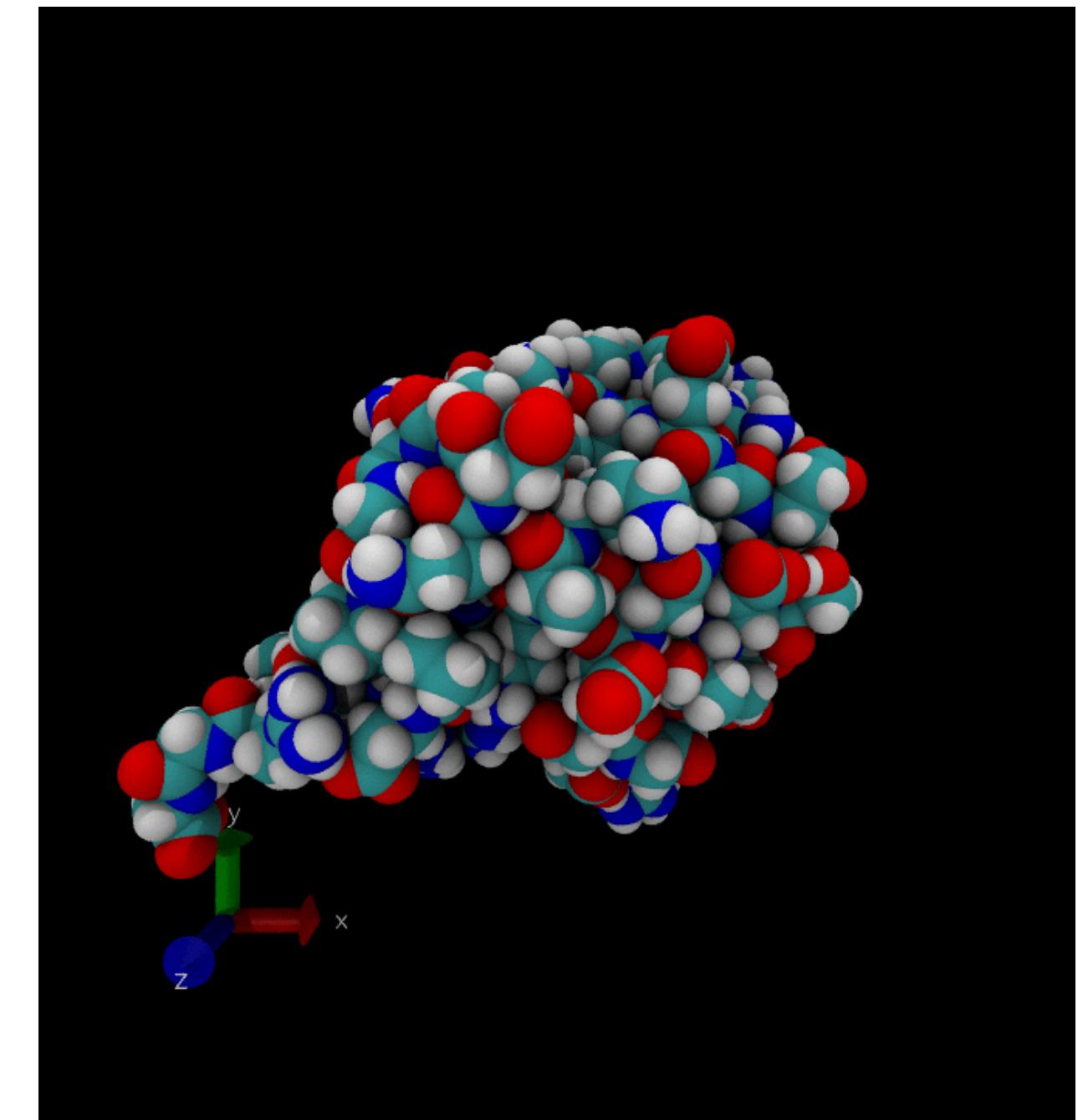
particles $\sim p(\text{particles})$



[Aquarius simulation]

Molecular dynamics

configurations $\sim p(\text{configurations})$



[E. Cancès et al.]

Simulators

$$x \sim p(x)$$

Simulators are ubiquitous: *they prescribe a way to sample from the data distribution*

Collider data

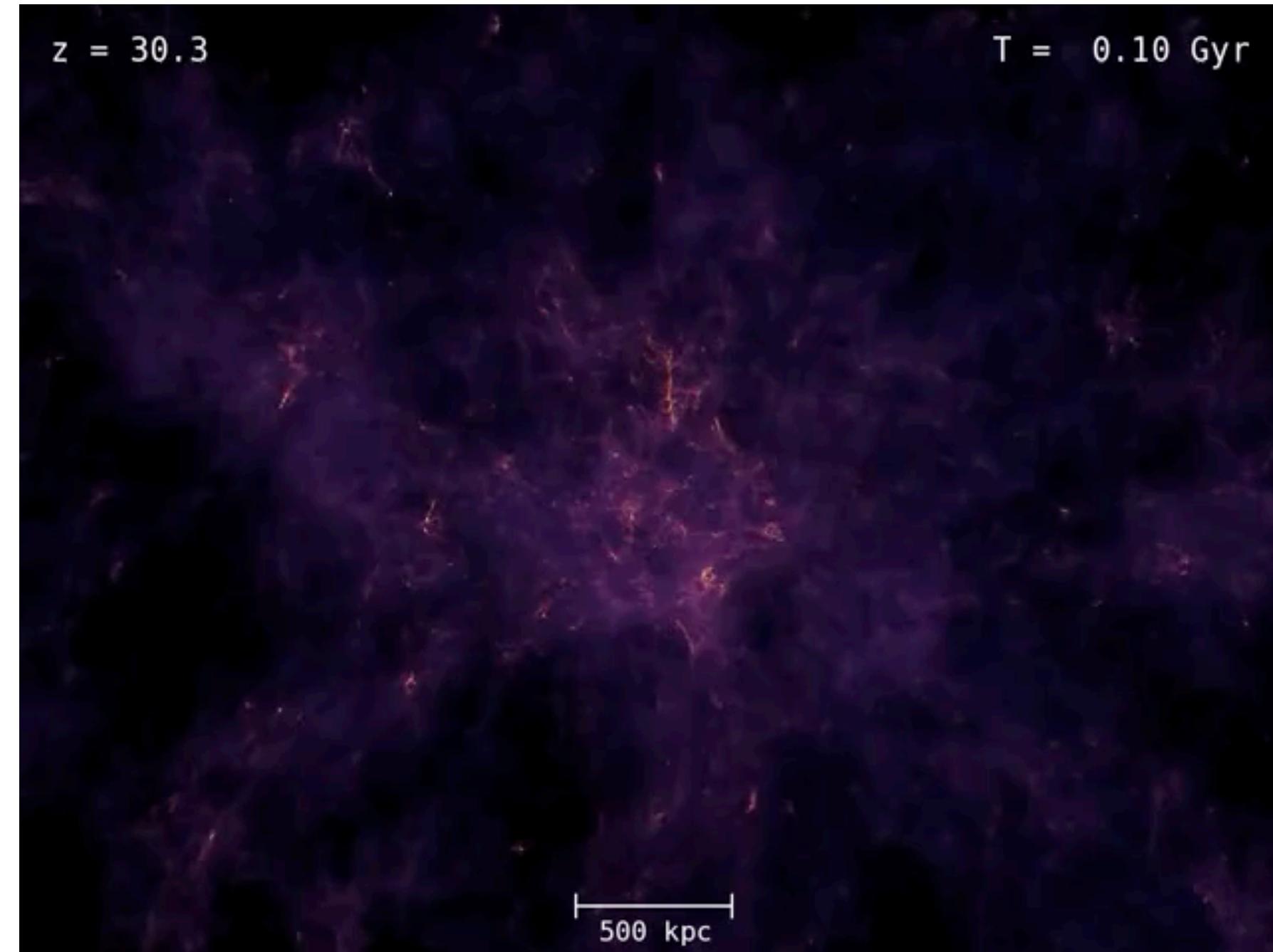
particles $\sim p(\text{particles})$



[C. Cesarotti with ATLAS]

Cosmology data

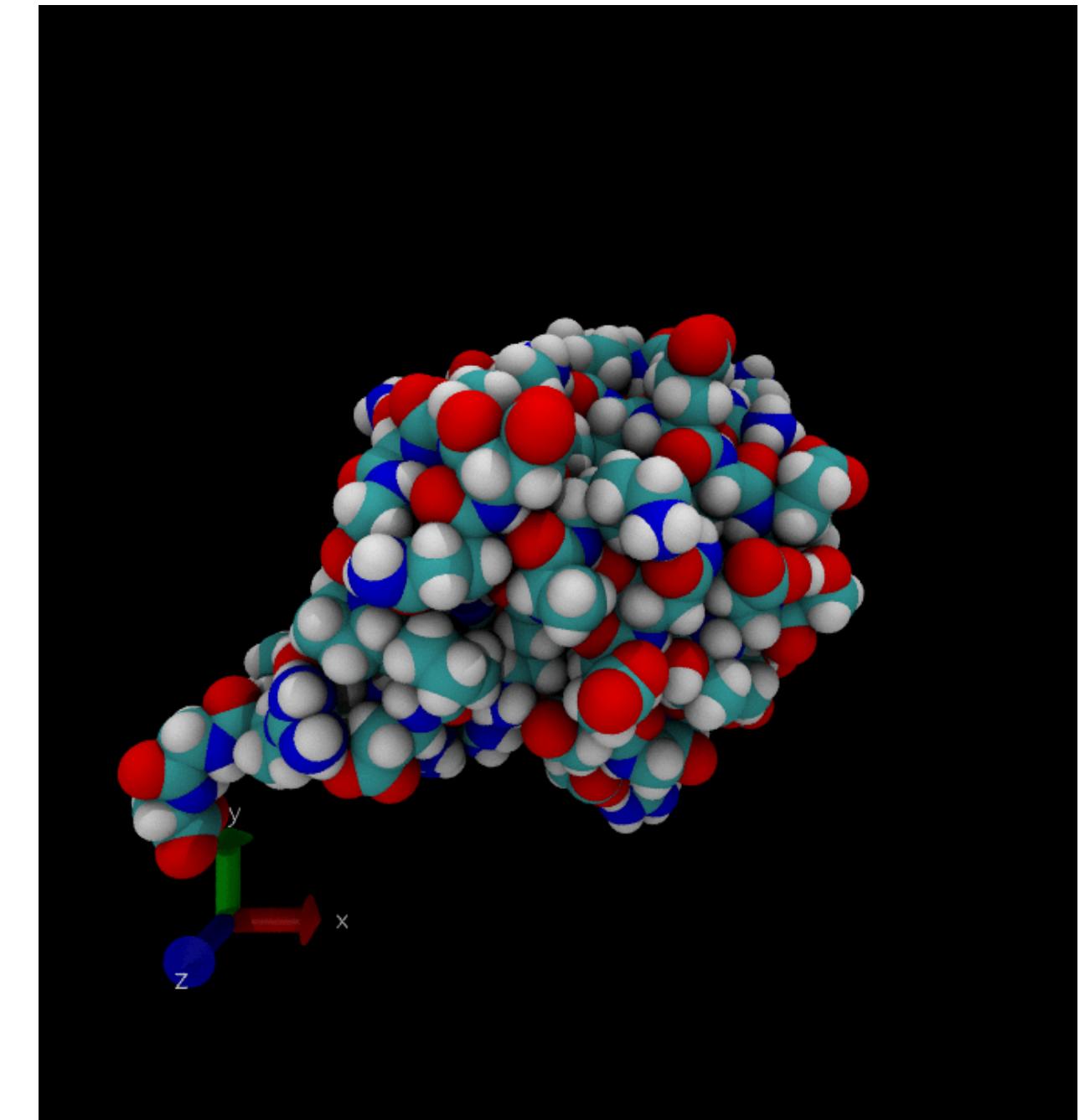
particles $\sim p(\text{particles})$



[Aquarius simulation]

Molecular dynamics

configurations $\sim p(\text{configurations})$



[E. Cancès et al.]

Conditional simulators

Conditional simulations sample from the likelihood $p(x \mid \theta)$

$$x \sim p(x; \mathcal{M})$$

Model

or

$$x \sim p(x \mid \theta)$$

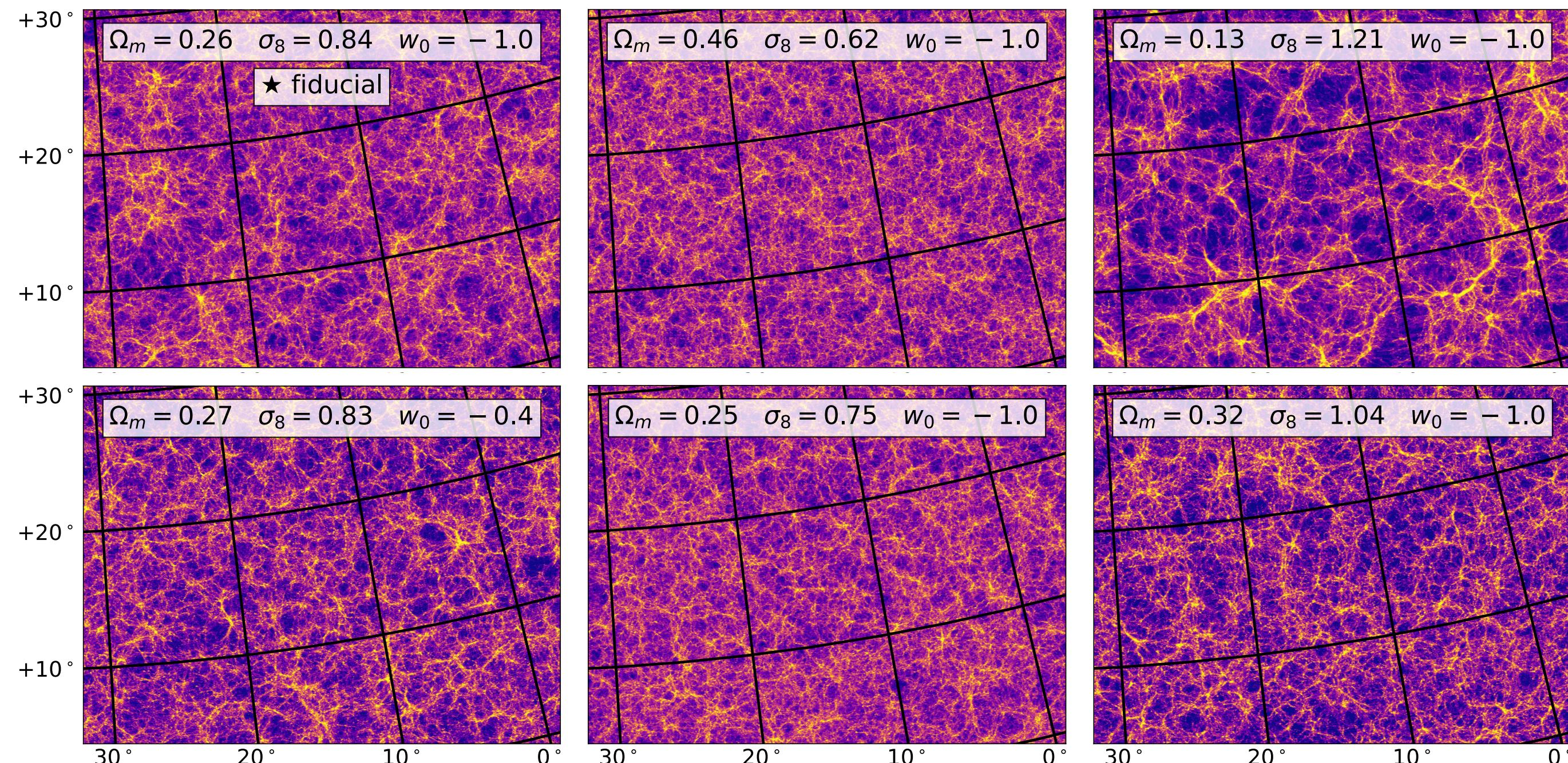
Model
parameters

Conditional simulators

Conditional simulations sample from the likelihood $p(x | \theta)$

Cosmology data

$$\text{map} \sim p(\text{map} | \{\Omega_m, \sigma_8, w_0\})$$



$$x \sim p(x; \mathcal{M})$$

Model

or

$$x \sim p(x | \theta)$$

Model
parameters

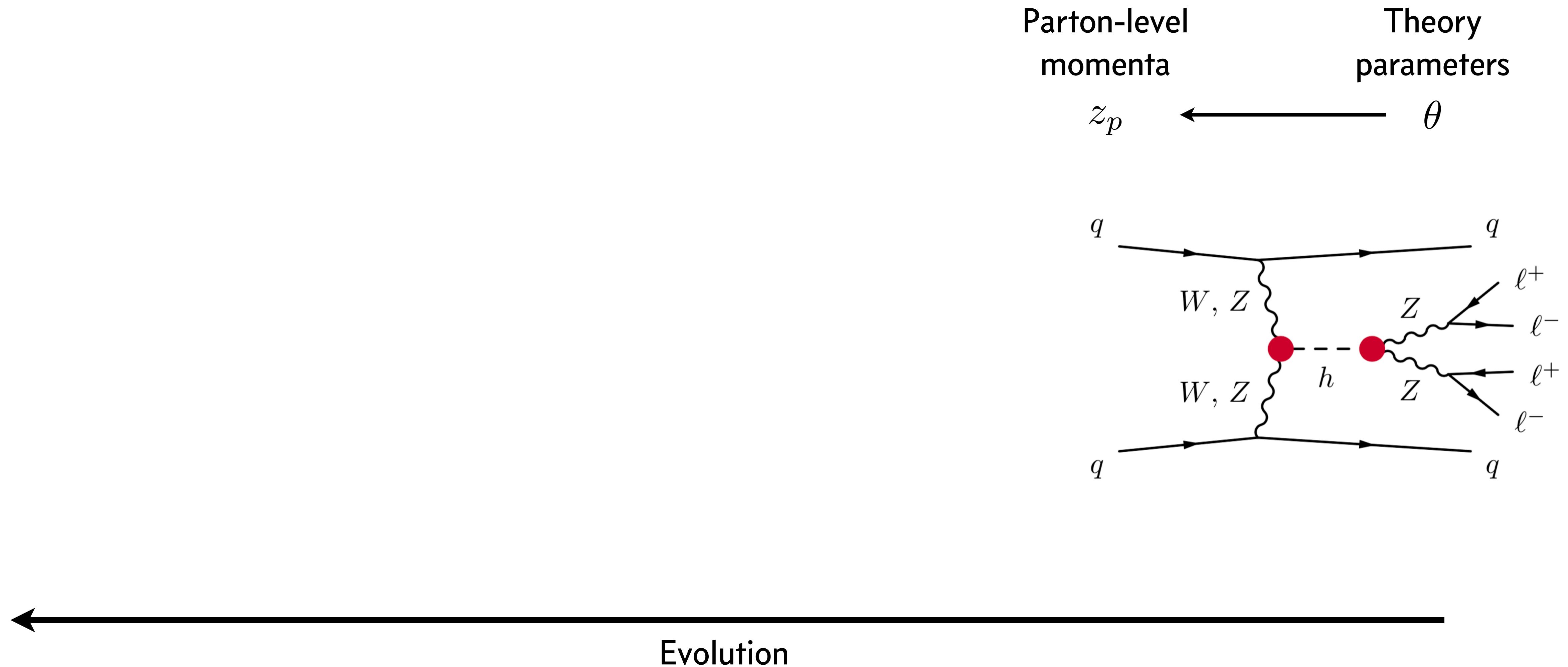
Are simulators all you need?

Theory
parameters
 θ



Evolution

Are simulators all you need?



Are simulators all you need?

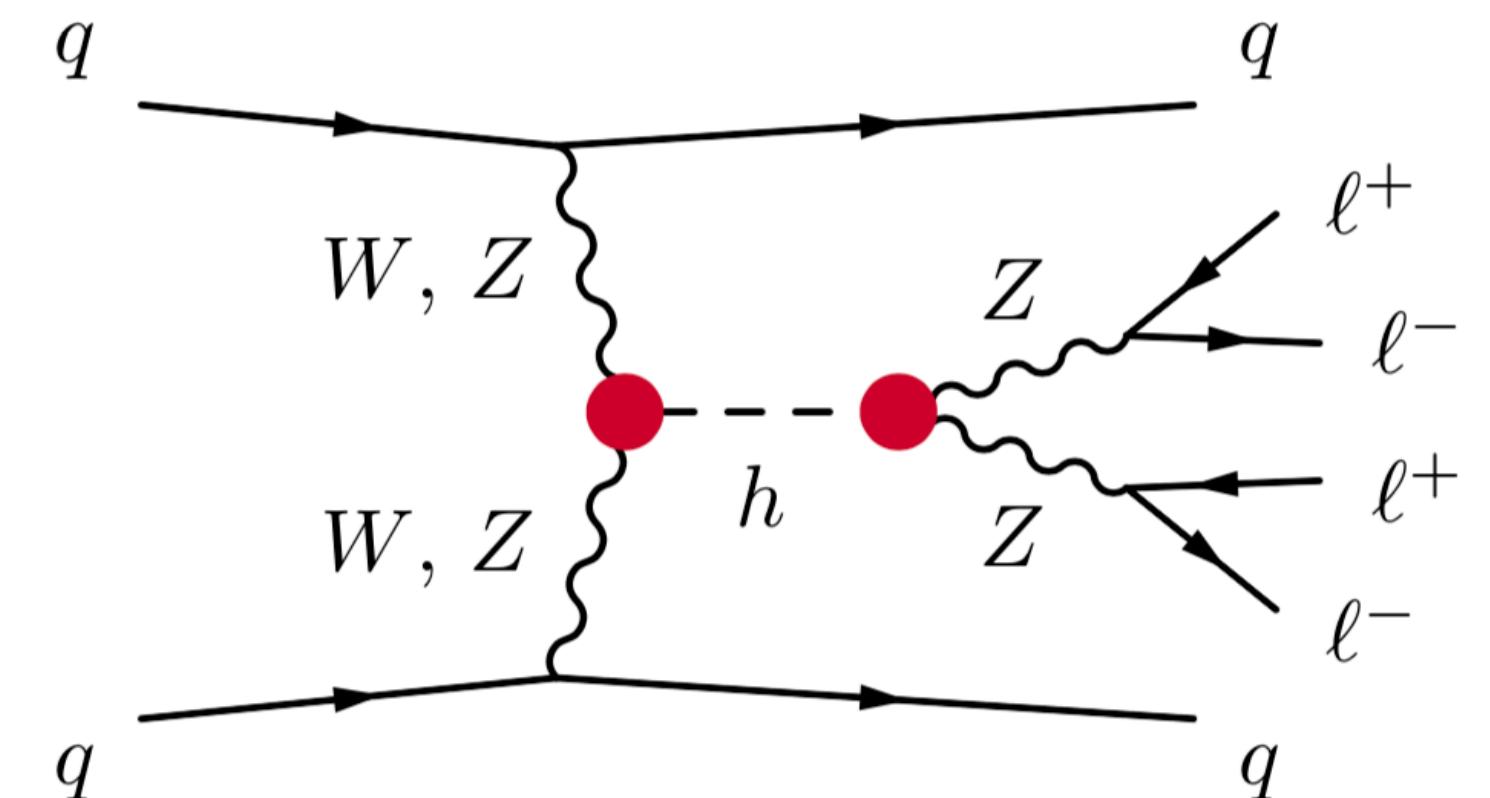
Latent variables

Parton-level
momenta

$$z_p$$

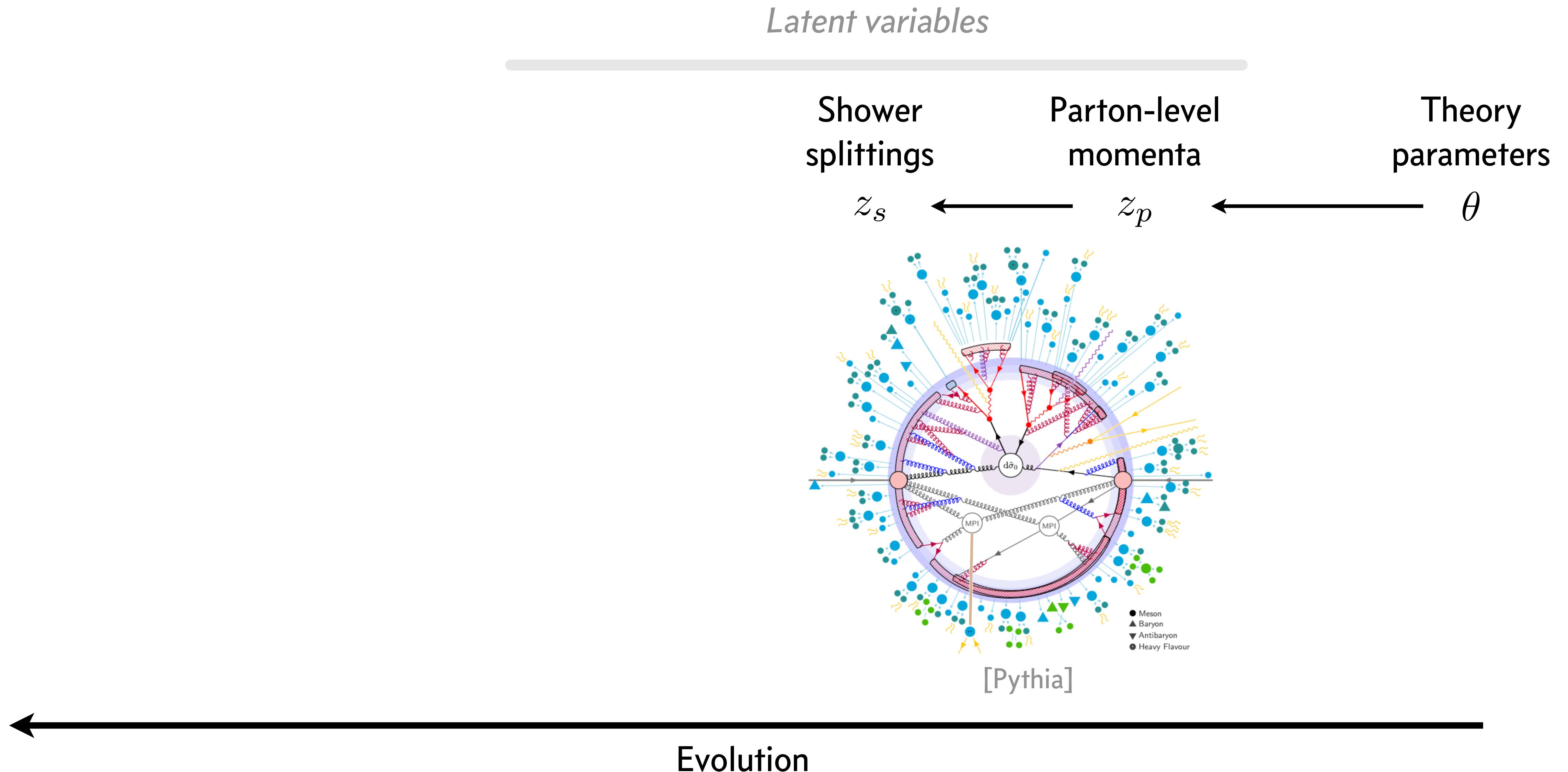
Theory
parameters

$$\theta$$

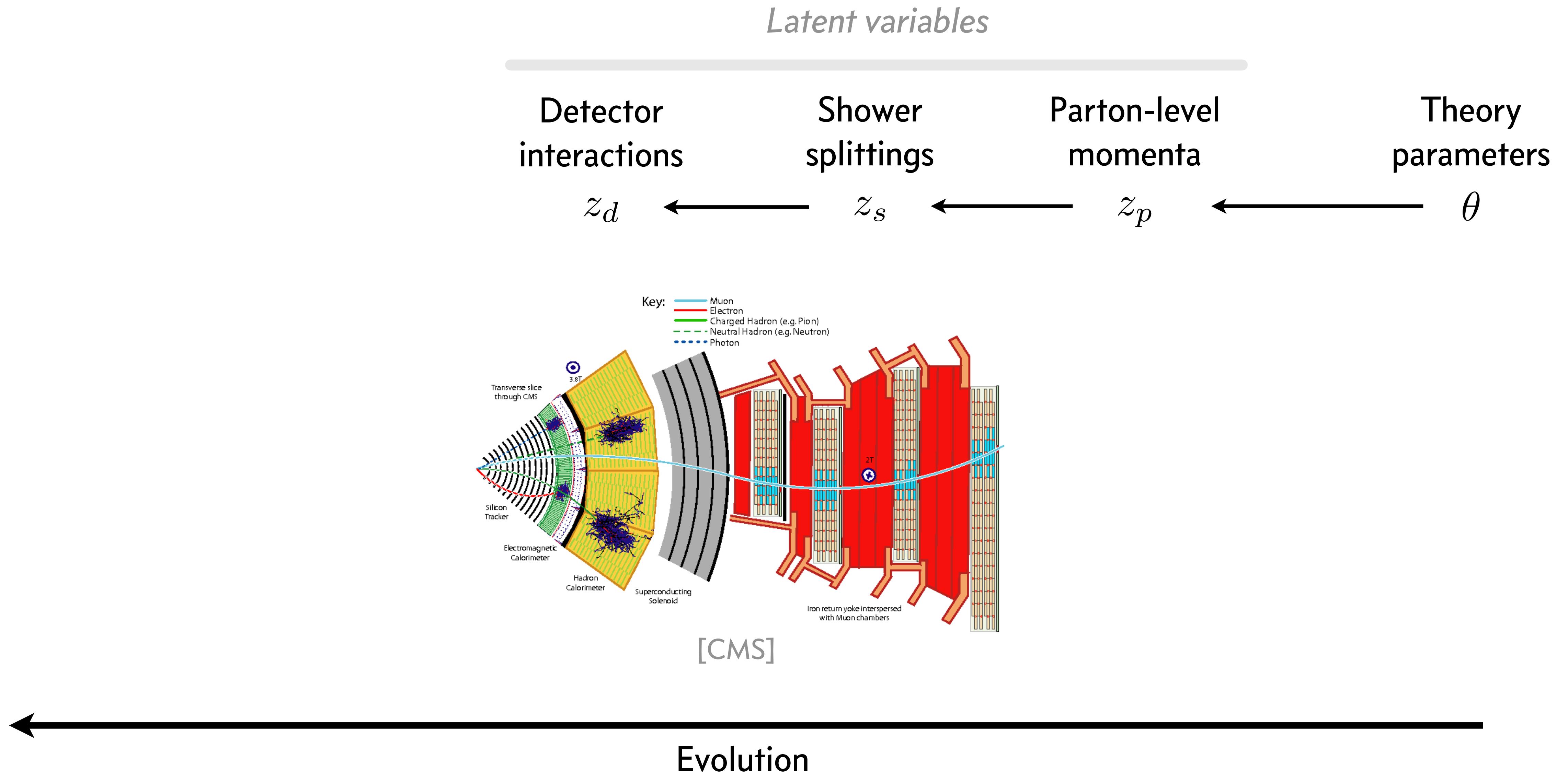


Evolution

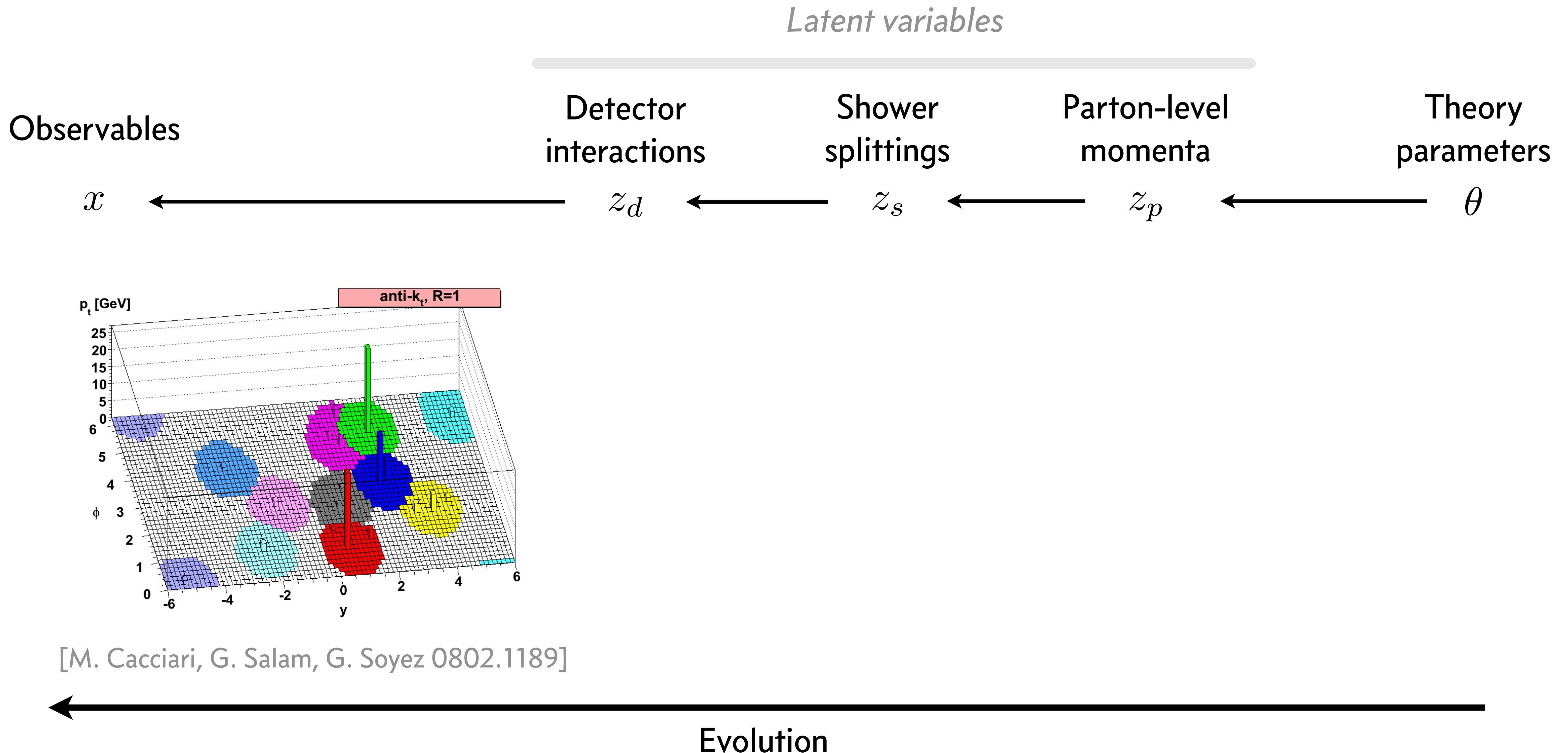
Are simulators all you need?



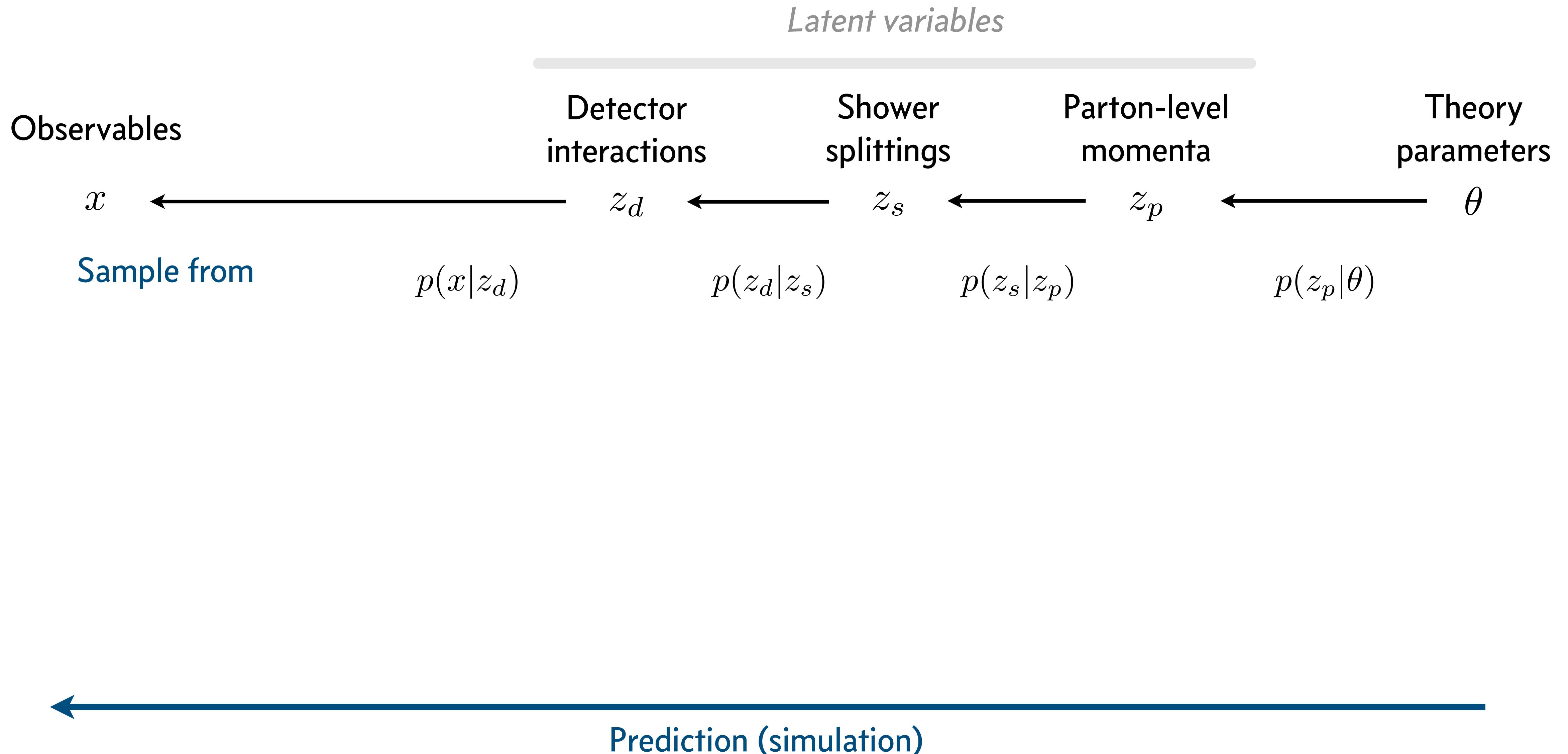
Are simulators all you need?



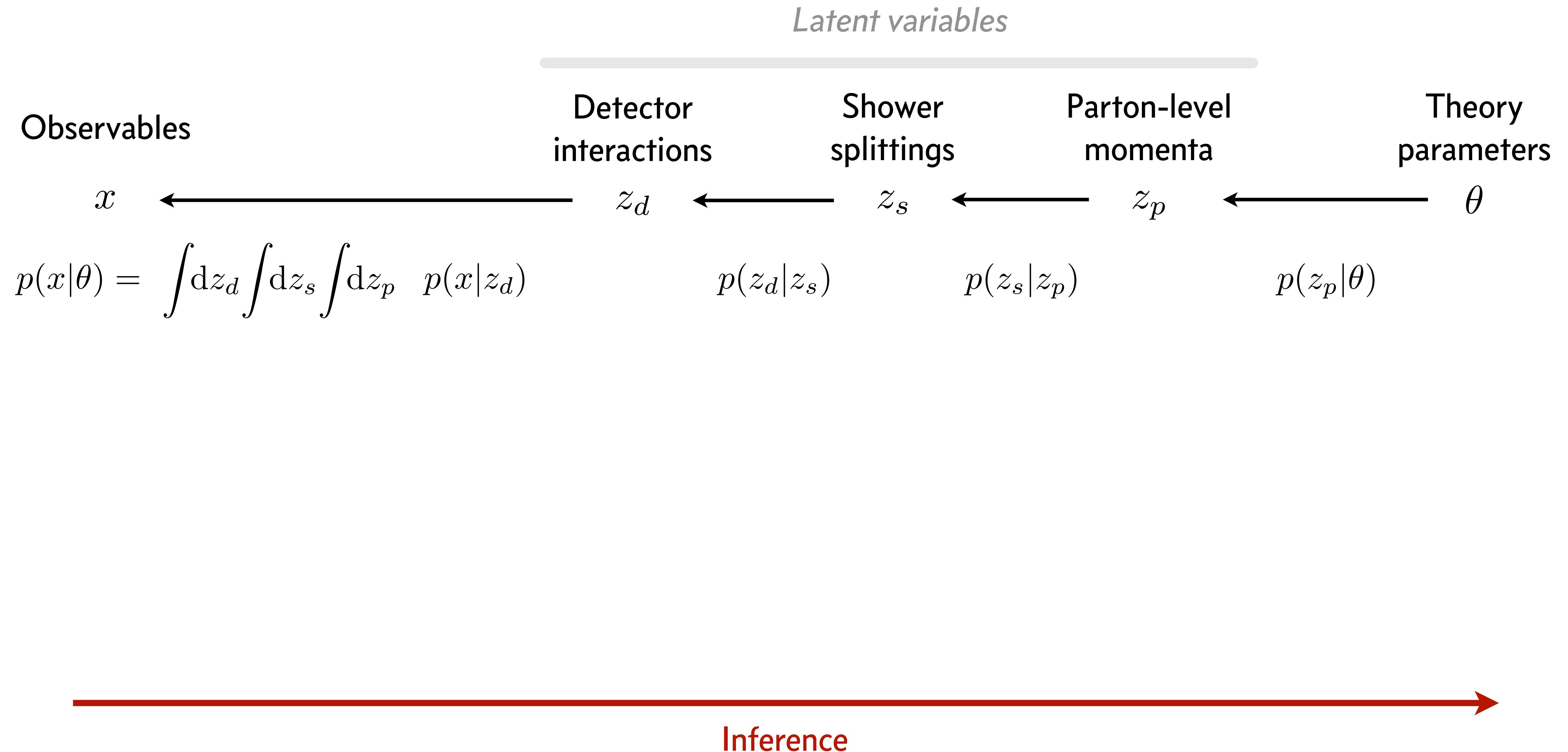
Are simulators all you need?



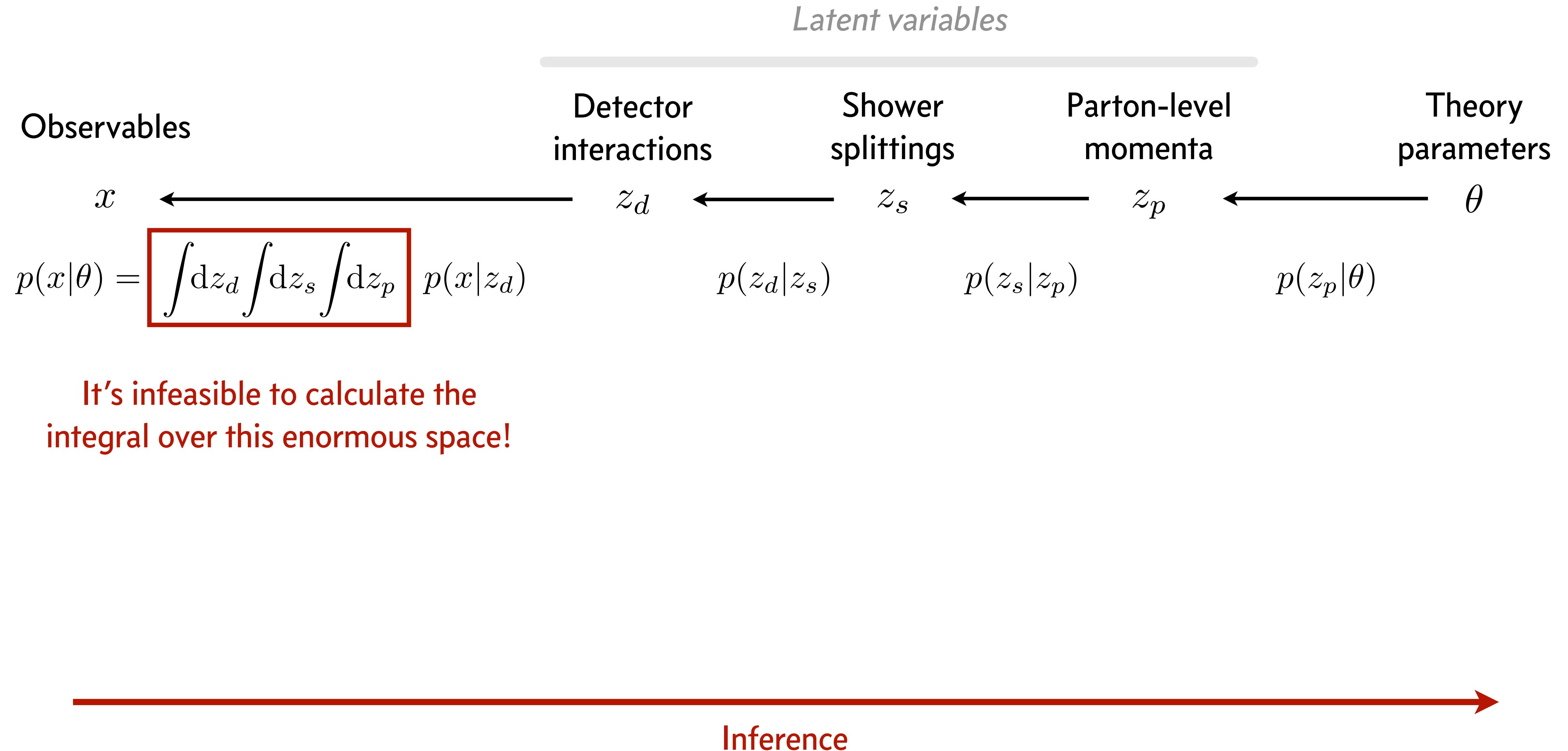
Are simulators all you need?



Are simulators all you need?



Are simulators all you need?



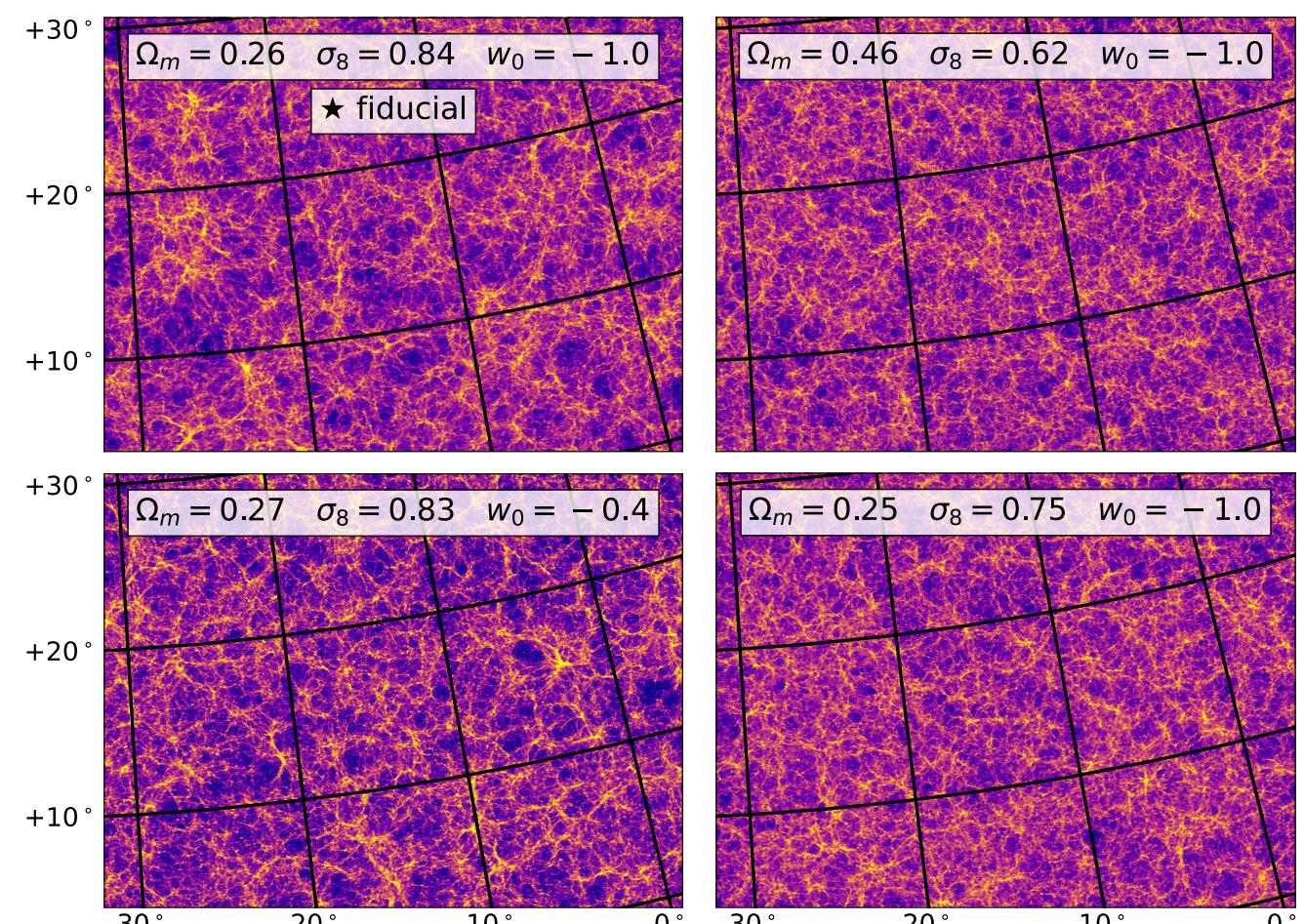
What could we do with $p(x)$?

What could we do with $p(x)$?

Produce samples

*for downstream applications: a
fast simulator/emulator*

$$x \sim p(x \mid \theta)$$



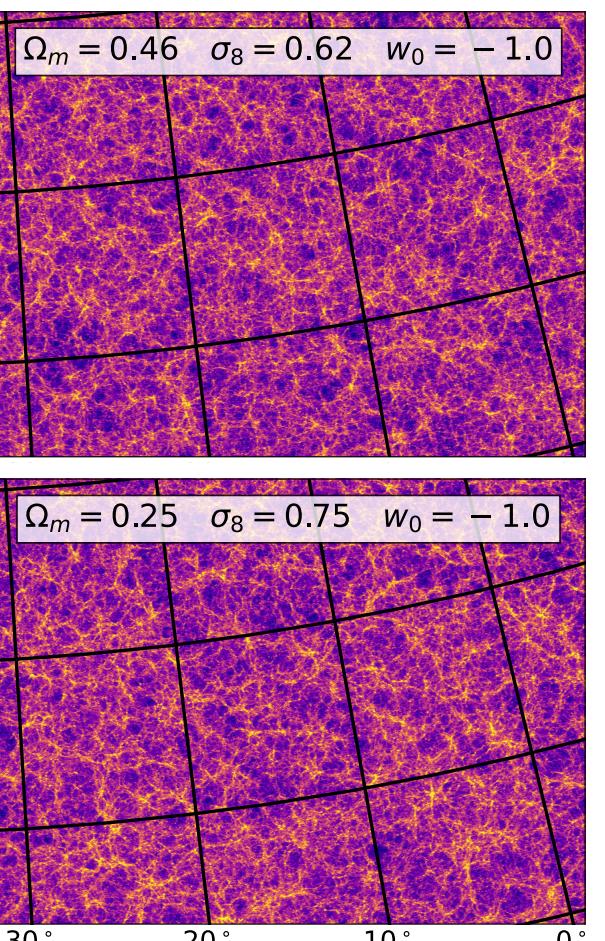
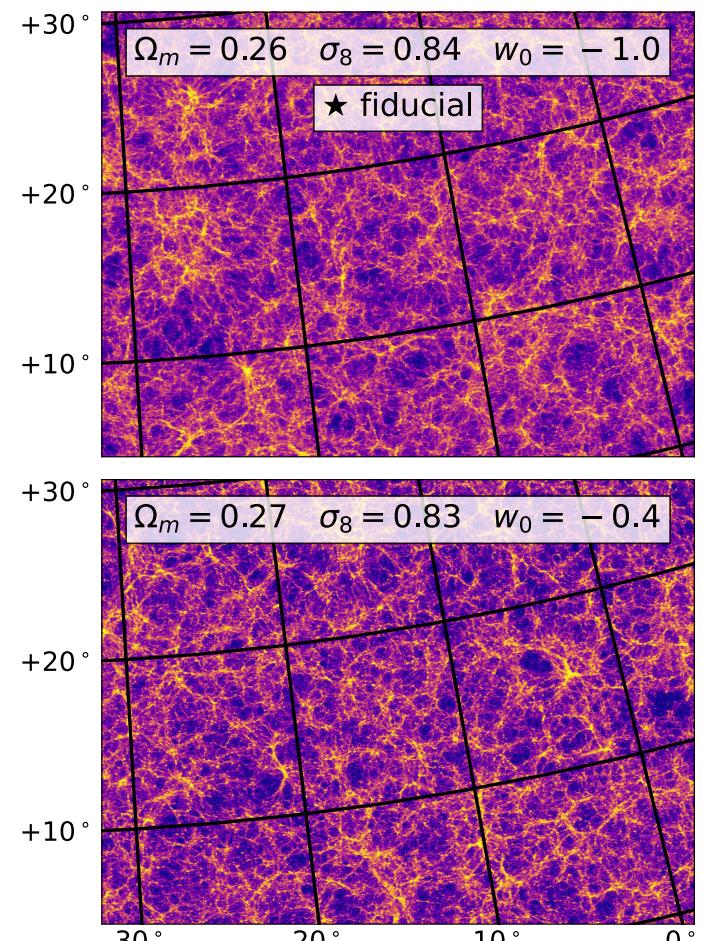
[Kacprzak et al 2022]

What could we do with $p(x)$?

Produce samples

for downstream applications: a fast simulator/emulator

$$x \sim p(x \mid \theta)$$

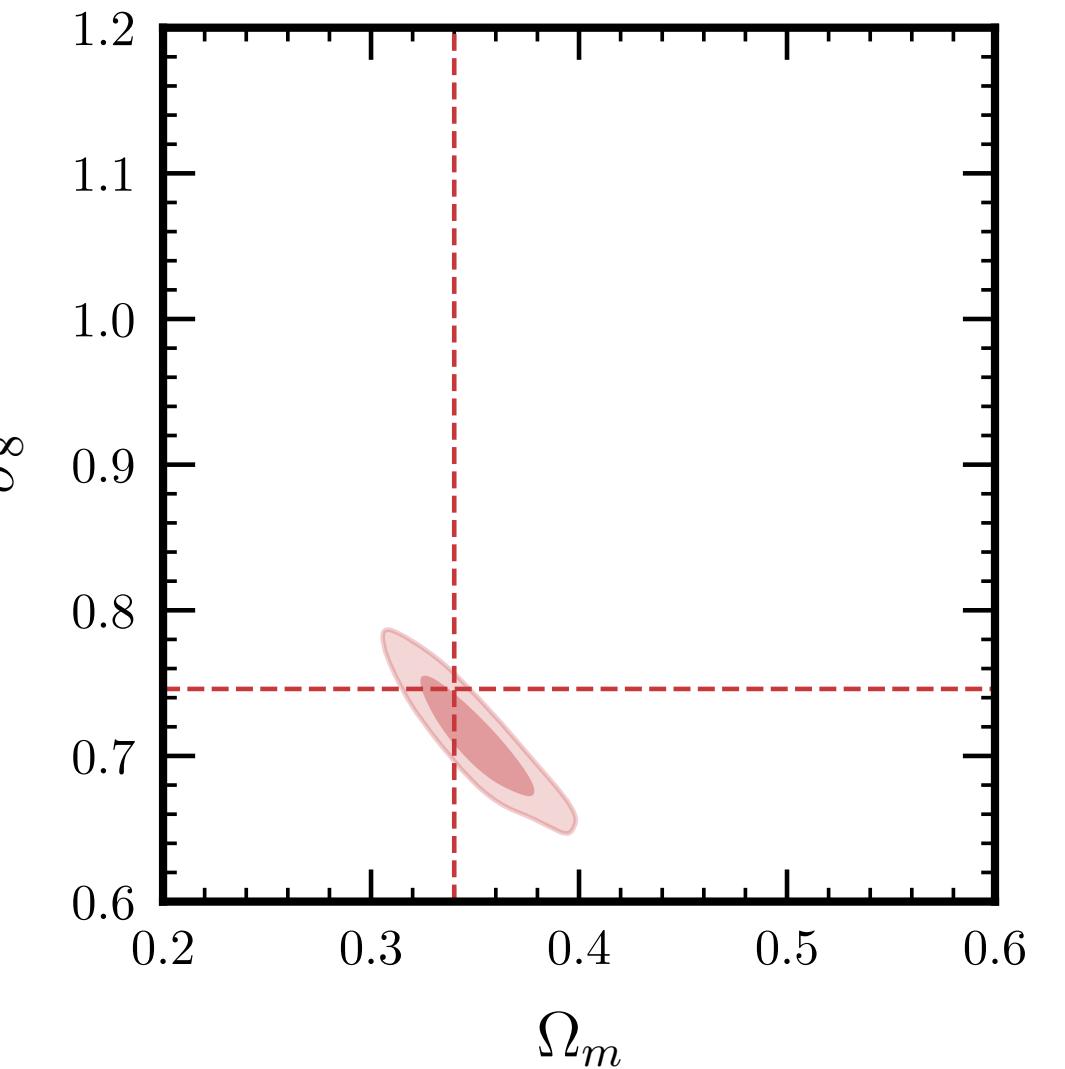


[Kacprzak et al 2022]

Evaluate likelihood

for model selection, parameter inference, outlier detection, ...

$$p(\theta \mid x) = \frac{p(x \mid \theta) \cdot p(\theta)}{p(x)}$$

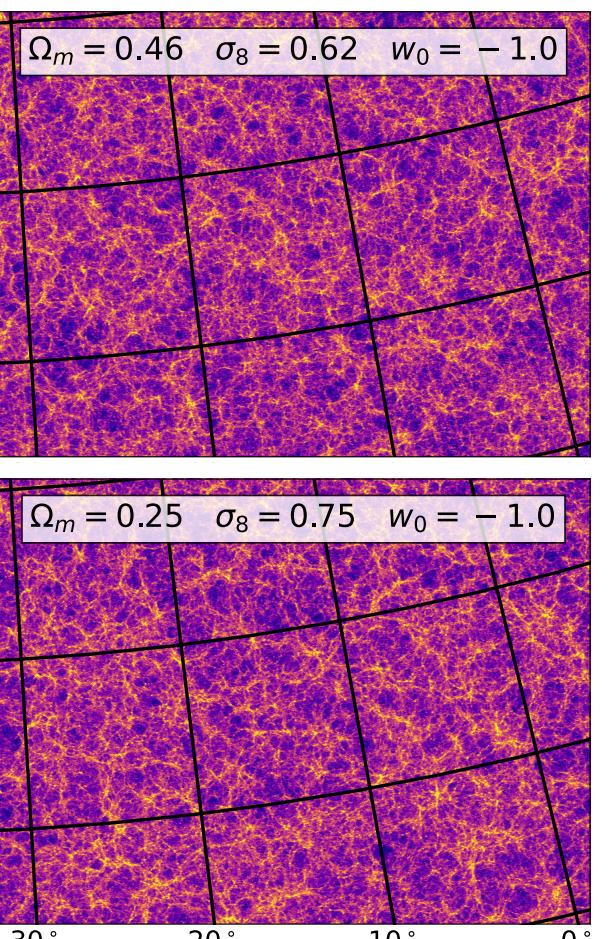
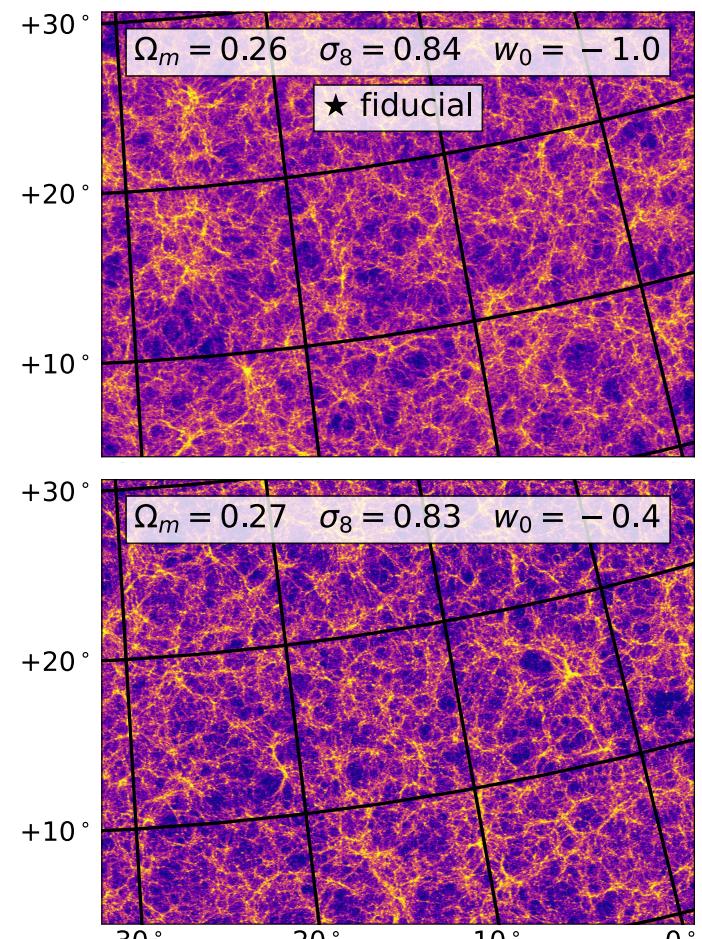


What could we do with $p(x)$?

Produce samples

for downstream applications: a fast simulator/emulator

$$x \sim p(x \mid \theta)$$

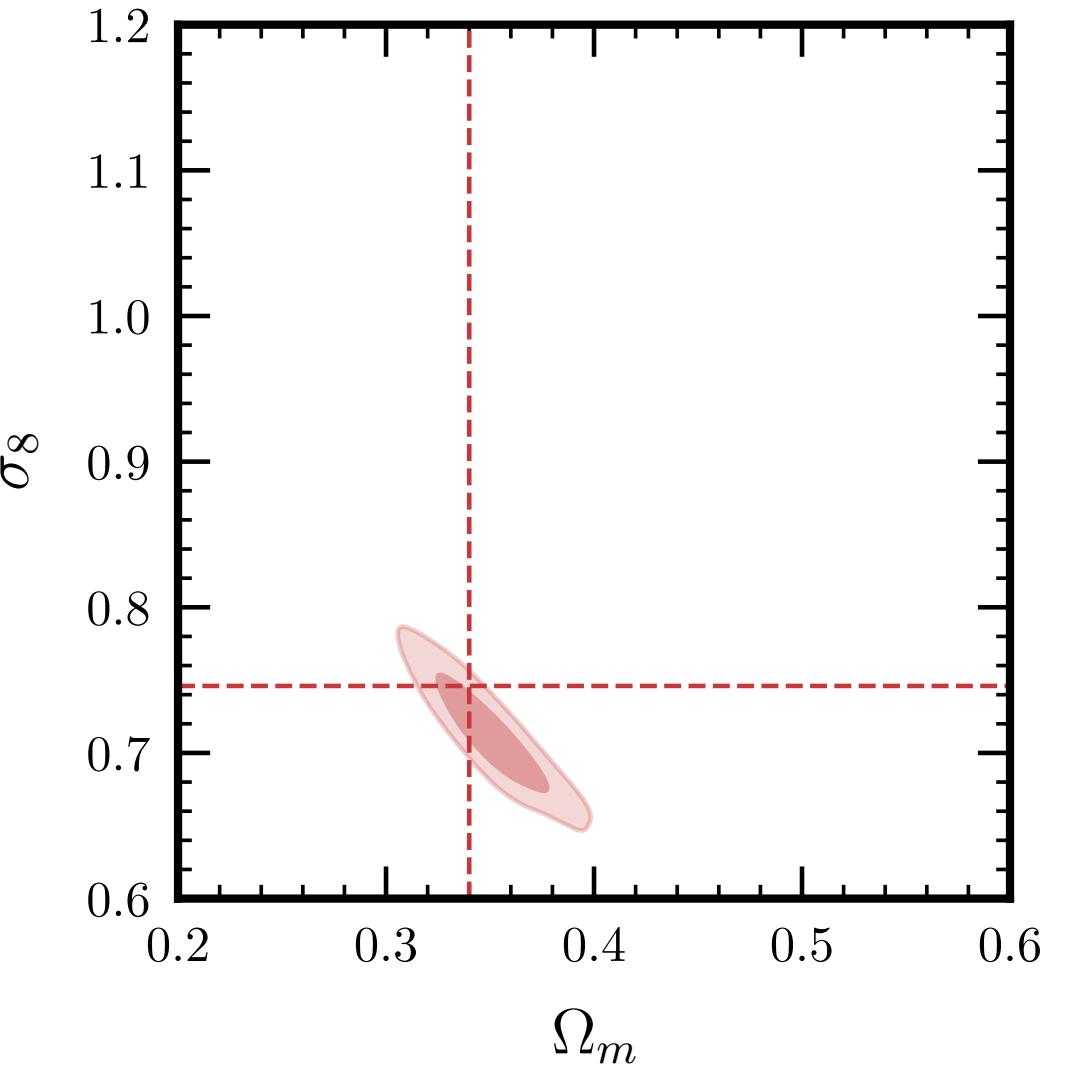


[Kacprzak et al 2022]

Evaluate likelihood

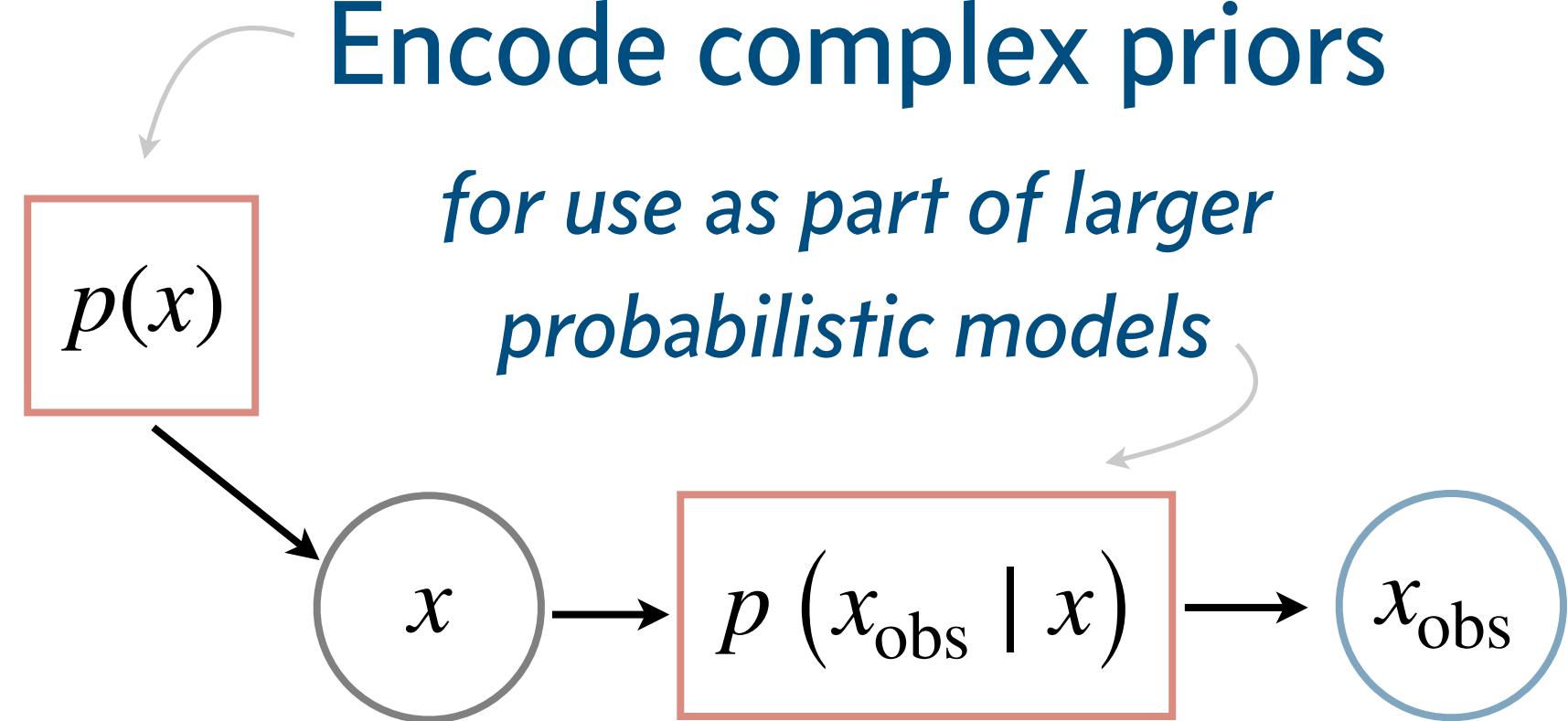
for model selection, parameter inference, outlier detection, ...

$$p(\theta \mid x) = \frac{p(x \mid \theta) \cdot p(\theta)}{p(x)}$$



Encode complex priors

for use as part of larger probabilistic models

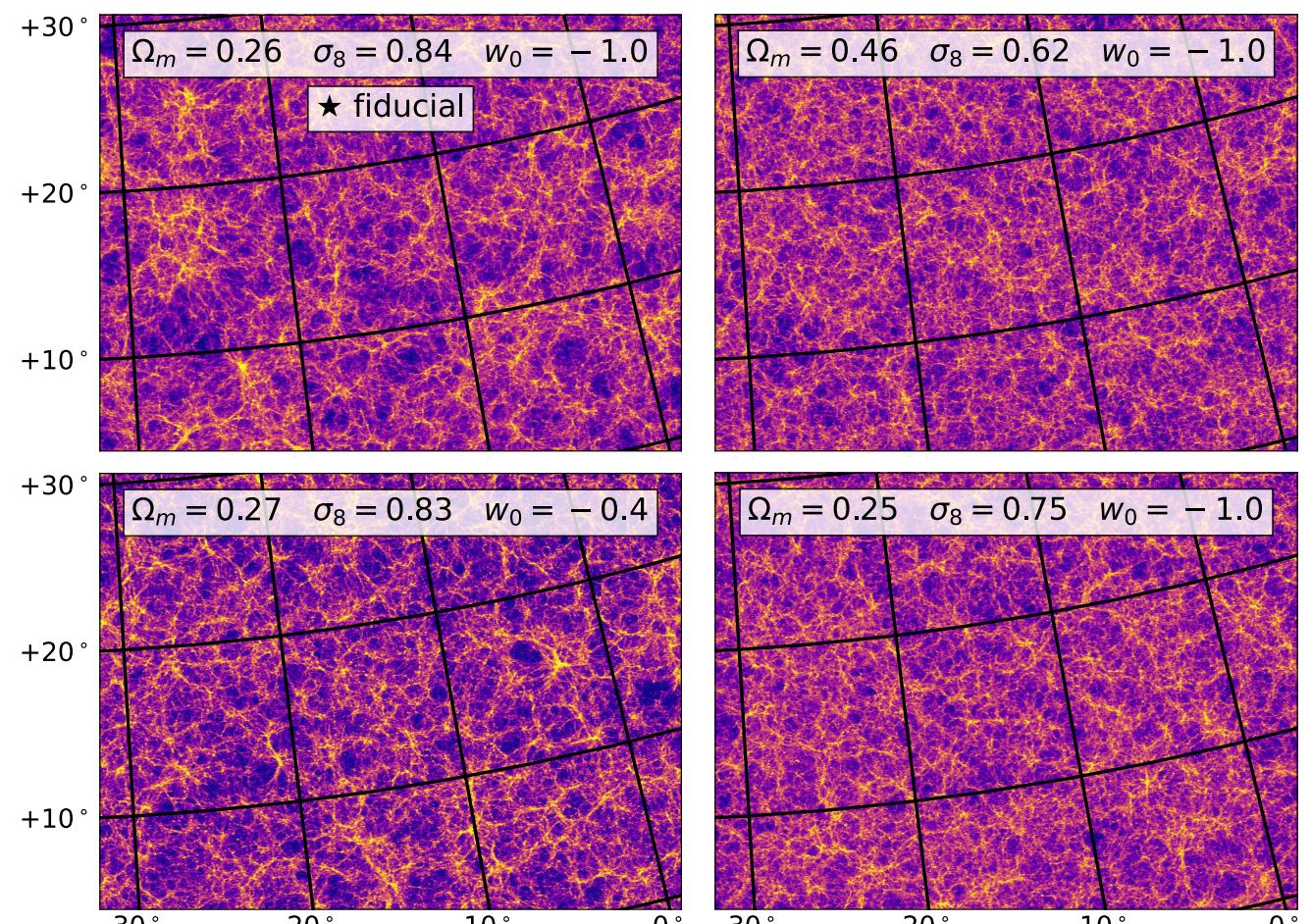


What could we do with $p(x)$?

Produce samples

for downstream applications: a fast simulator/emulator

$$x \sim p(x \mid \theta)$$

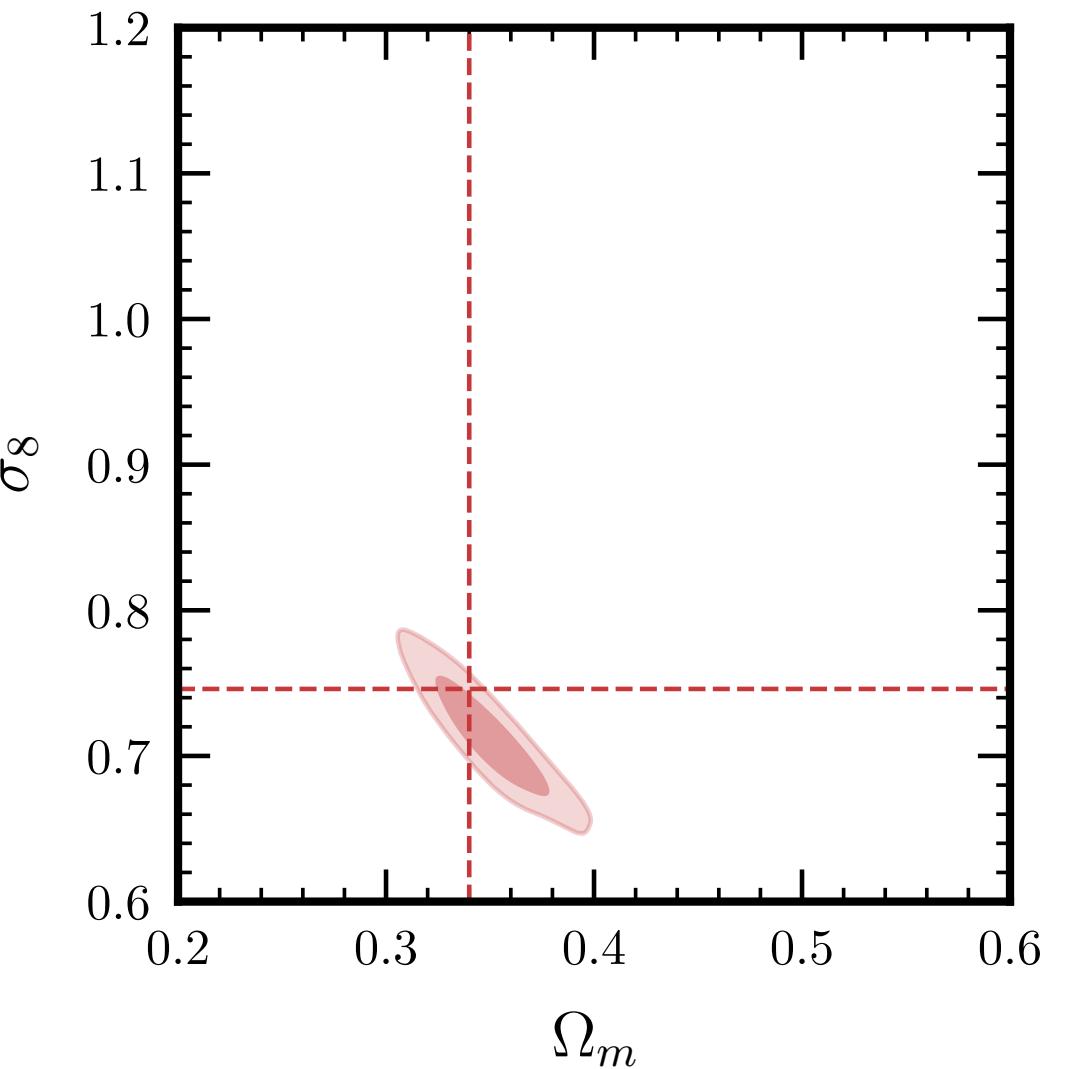


[Kacprzak et al 2022]

Evaluate likelihood

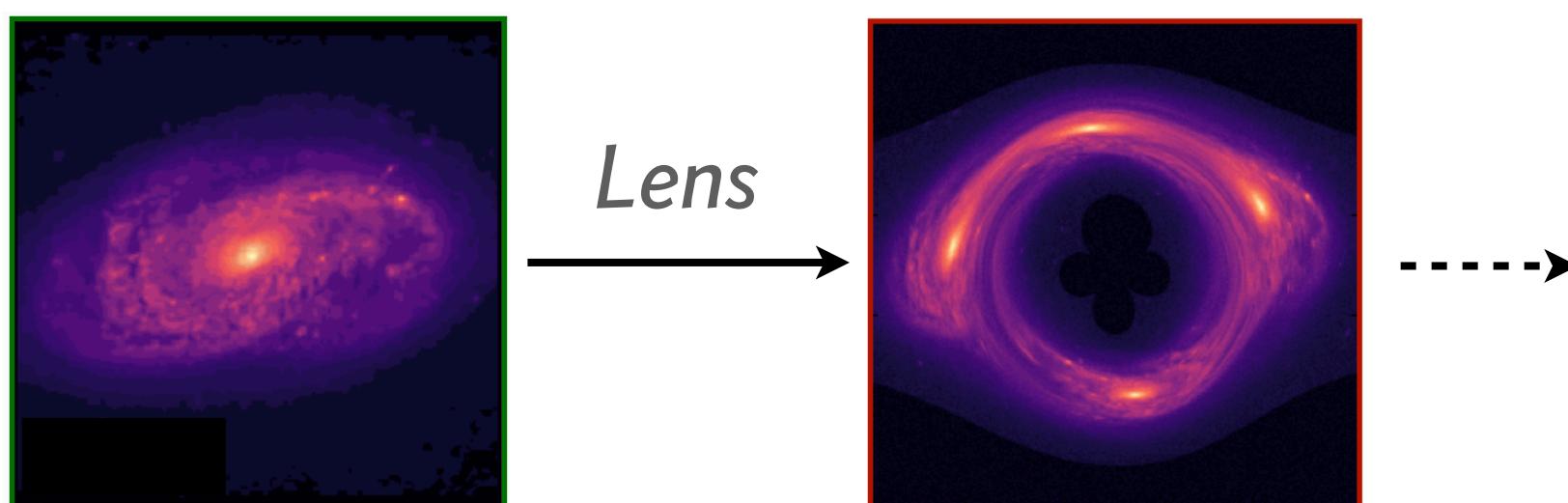
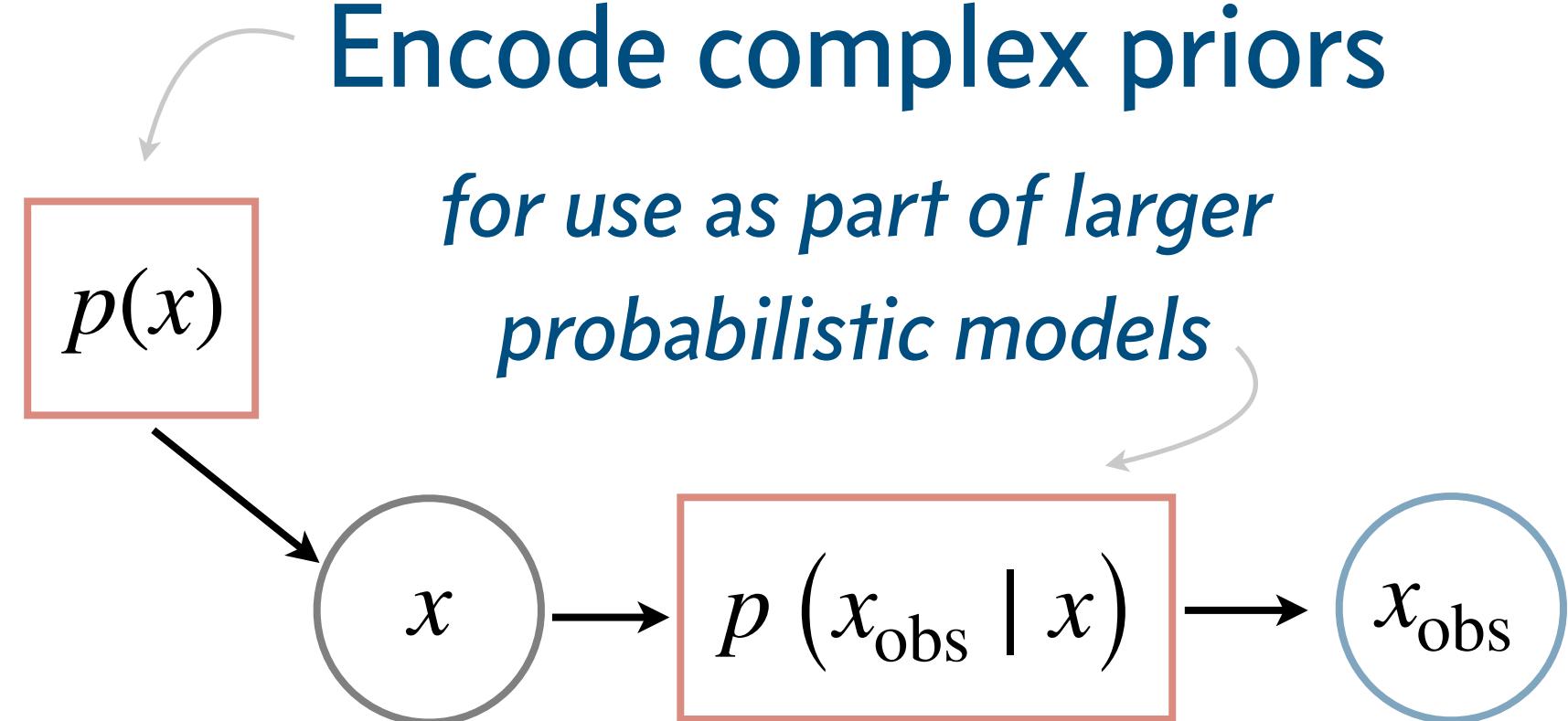
for model selection, parameter inference, outlier detection, ...

$$p(\theta \mid x) = \frac{p(x \mid \theta) \cdot p(\theta)}{p(x)}$$



Encode complex priors

for use as part of larger probabilistic models

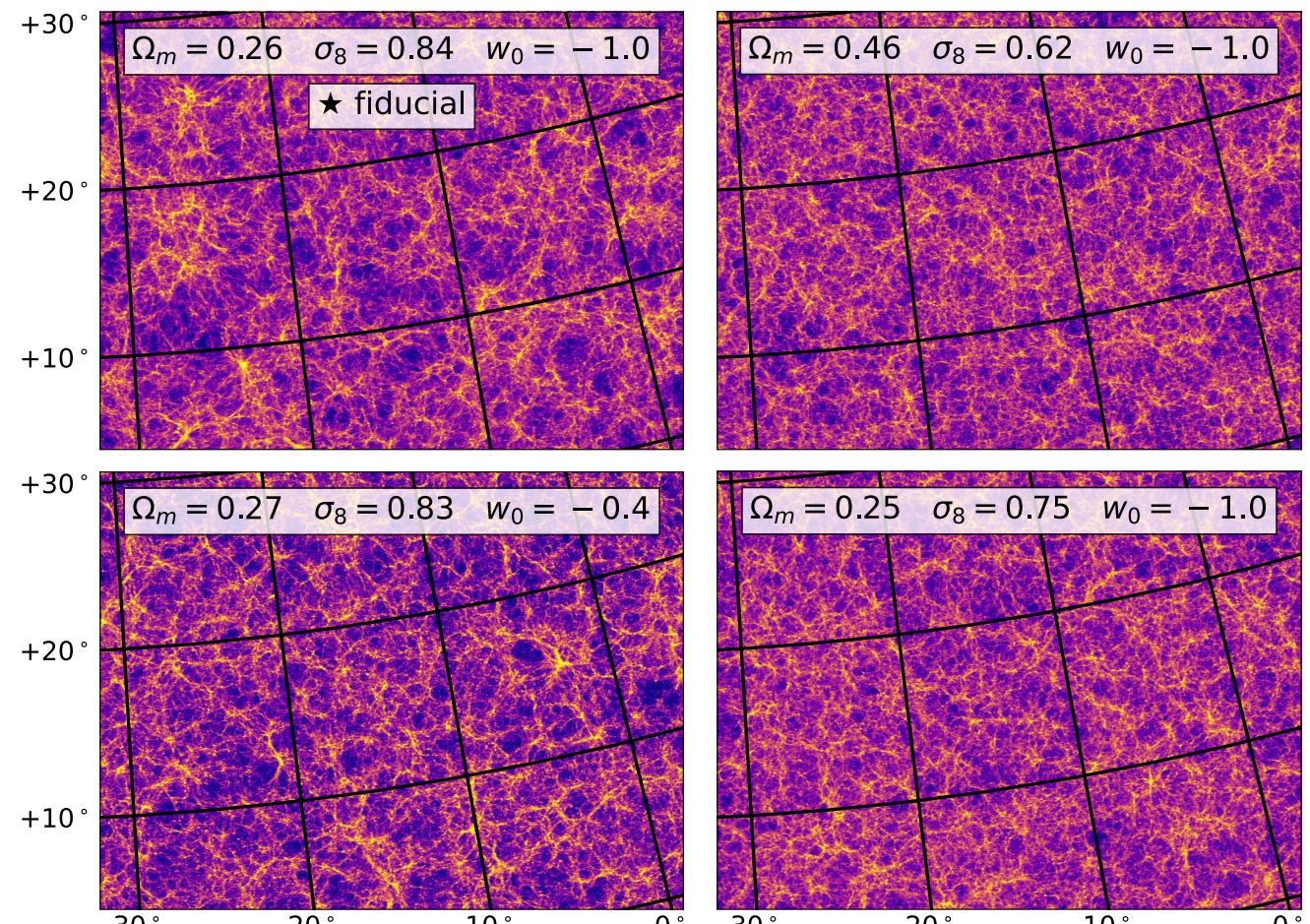


What could we do with $p(x)$?

Produce samples

for downstream applications: a fast simulator/emulator

$$x \sim p(x \mid \theta)$$

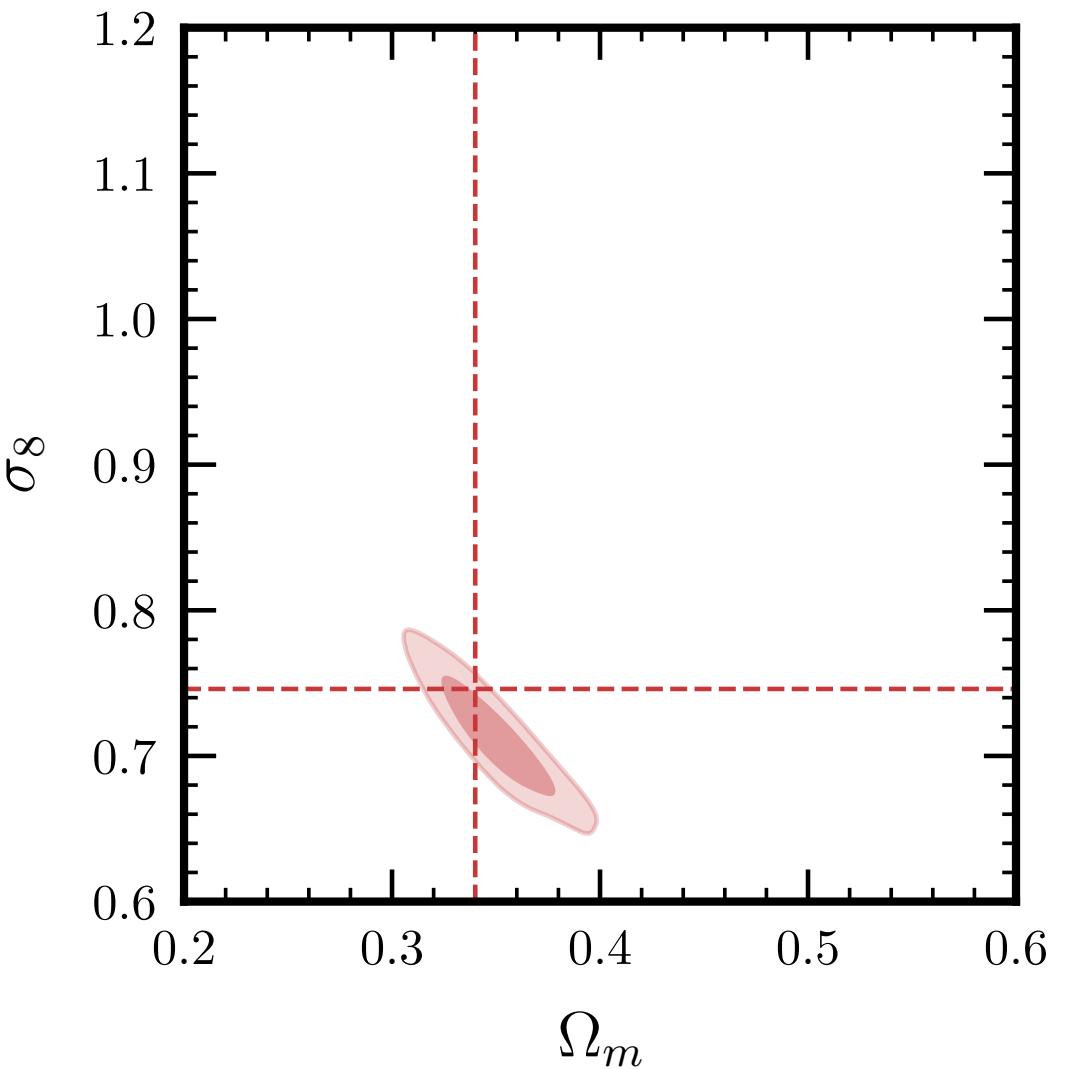


[Kacprzak et al 2022]

Evaluate likelihood

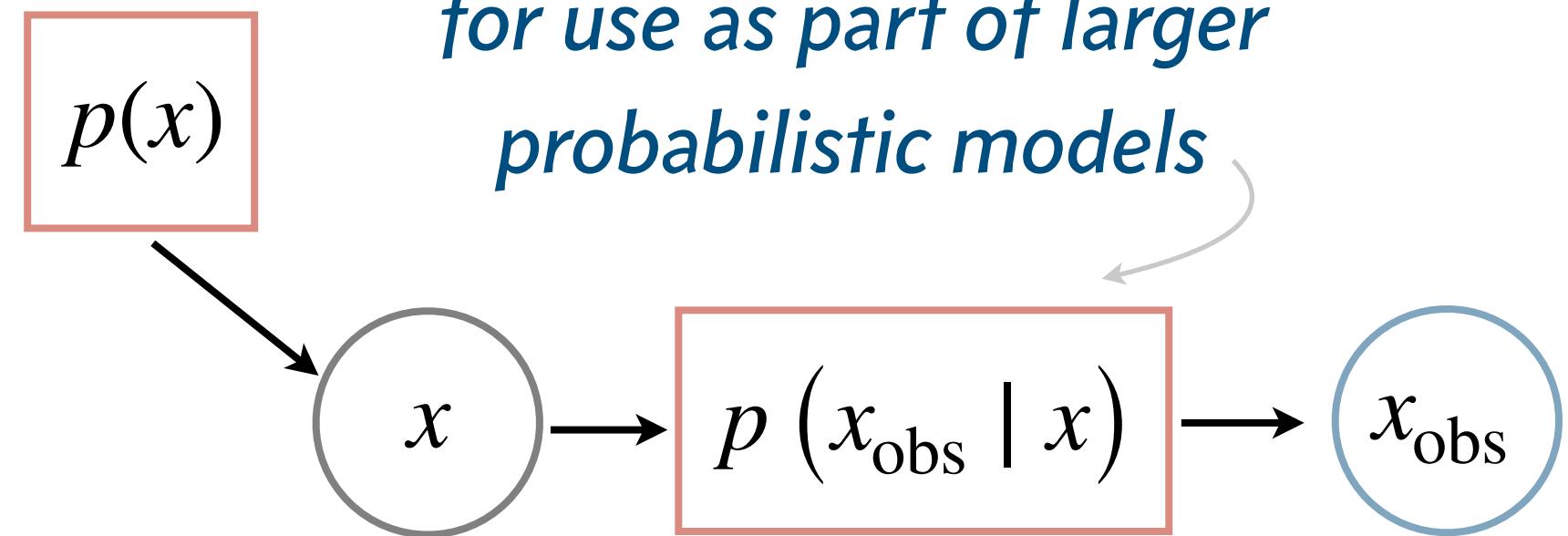
for model selection, parameter inference, outlier detection, ...

$$p(\theta \mid x) = \frac{p(x \mid \theta) \cdot p(\theta)}{p(x)}$$

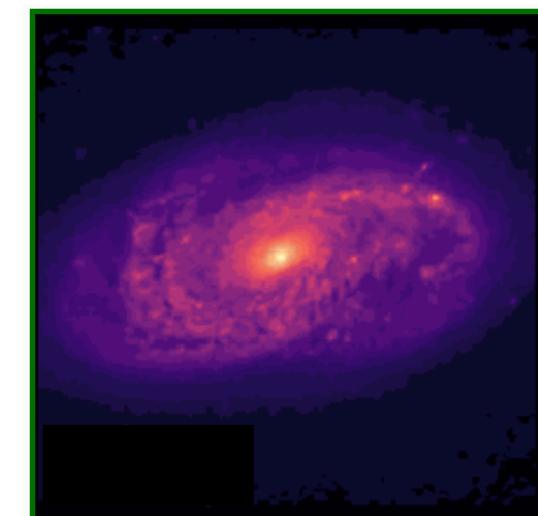


Encode complex priors

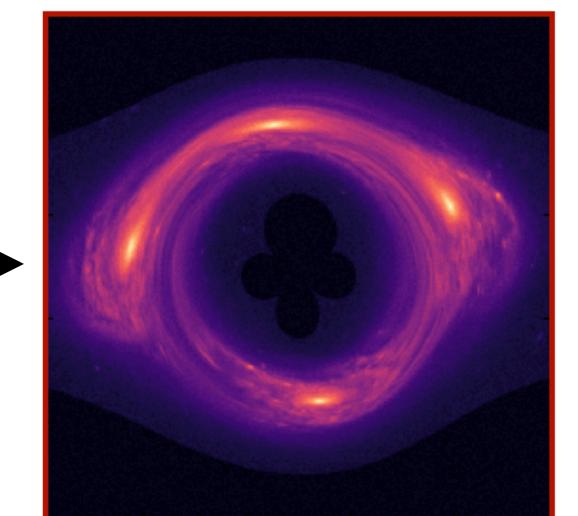
for use as part of larger probabilistic models



$p(\text{galaxies})$



Lens

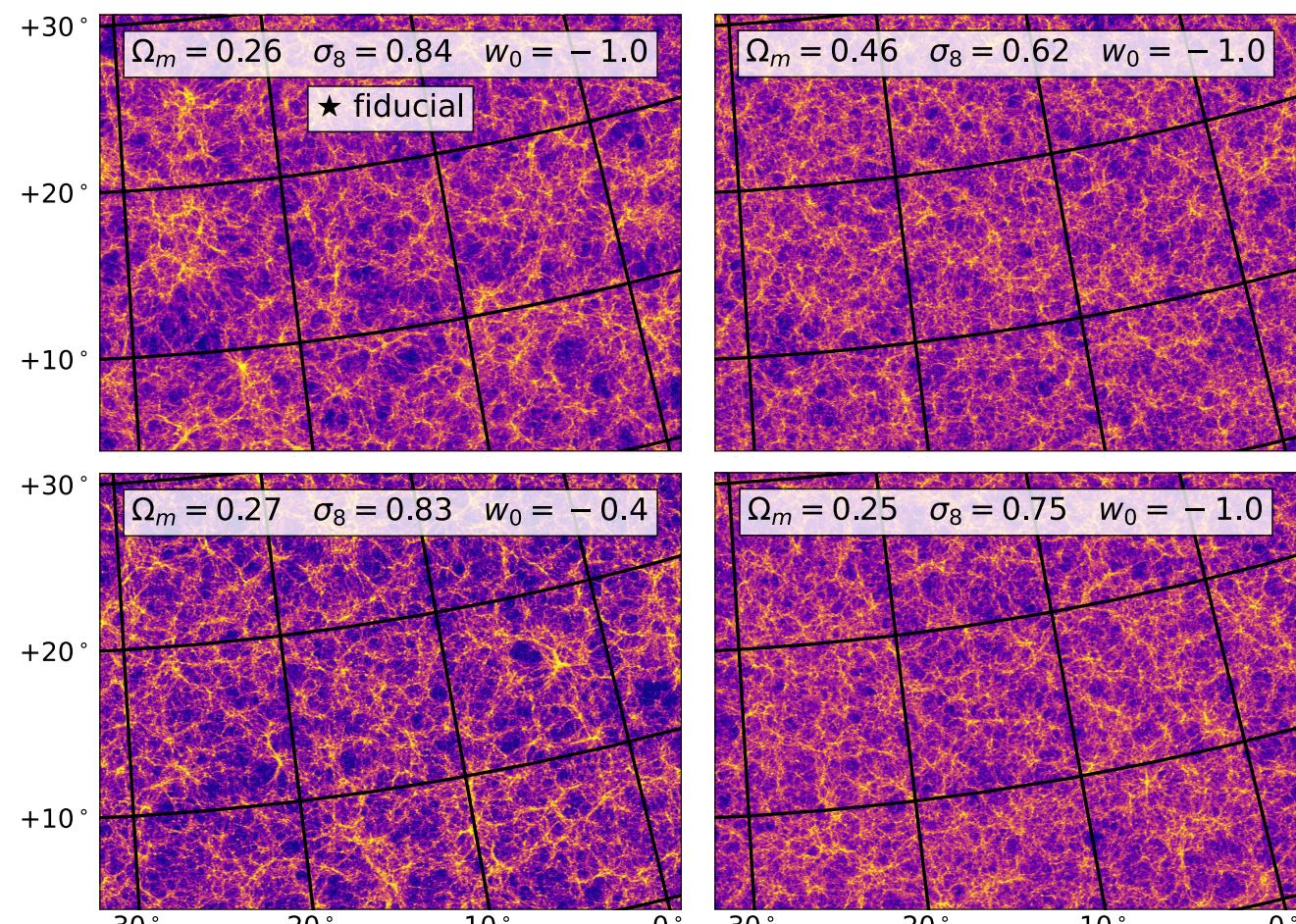


What could we do with $p(x)$?

Produce samples

for downstream applications: a fast simulator/emulator

$$x \sim p(x \mid \theta)$$

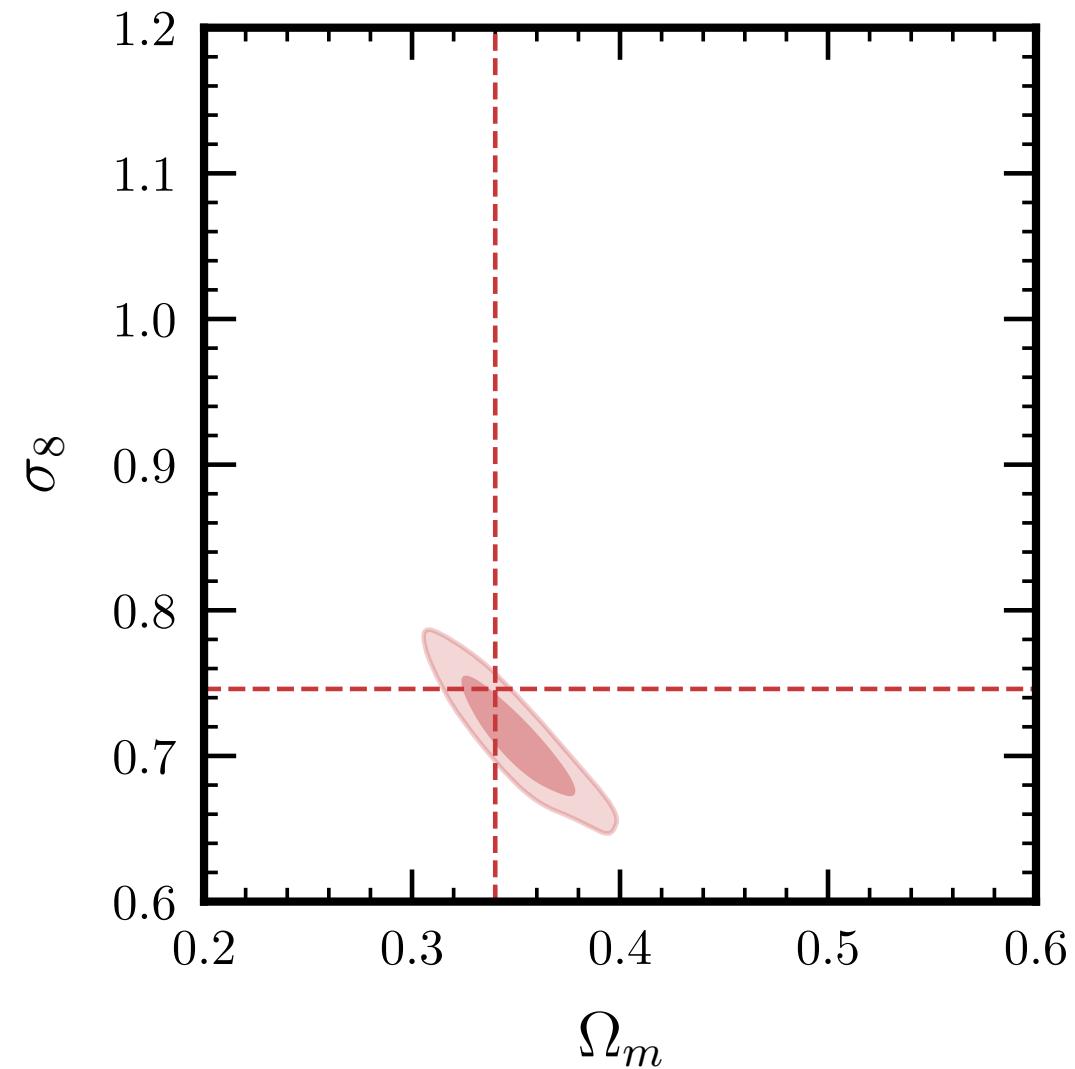


[Kacprzak et al 2022]

Evaluate likelihood

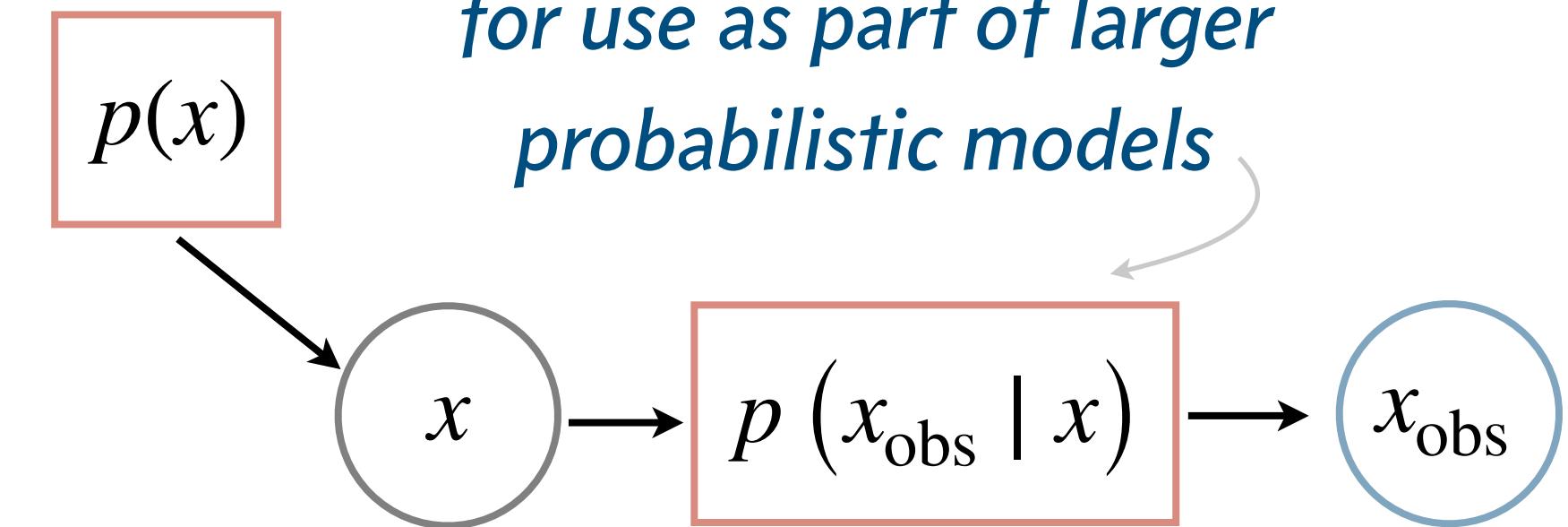
for model selection, parameter inference, outlier detection, ...

$$p(\theta \mid x) = \frac{p(x \mid \theta) \cdot p(\theta)}{p(x)}$$

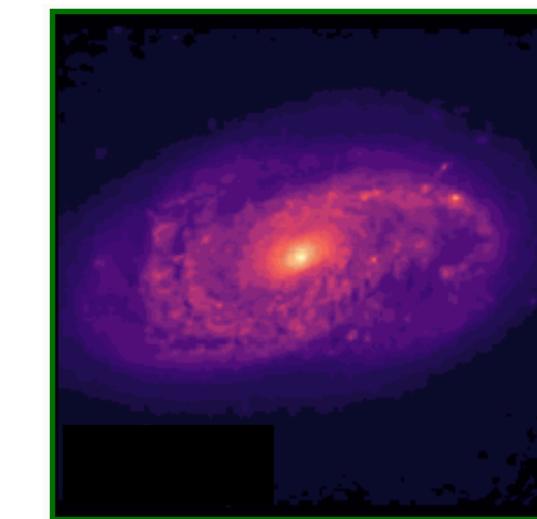


Encode complex priors

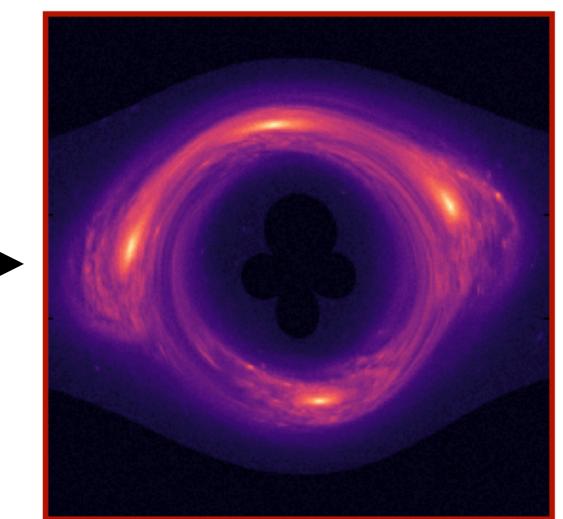
for use as part of larger probabilistic models



$p(\text{galaxies})$



Lens



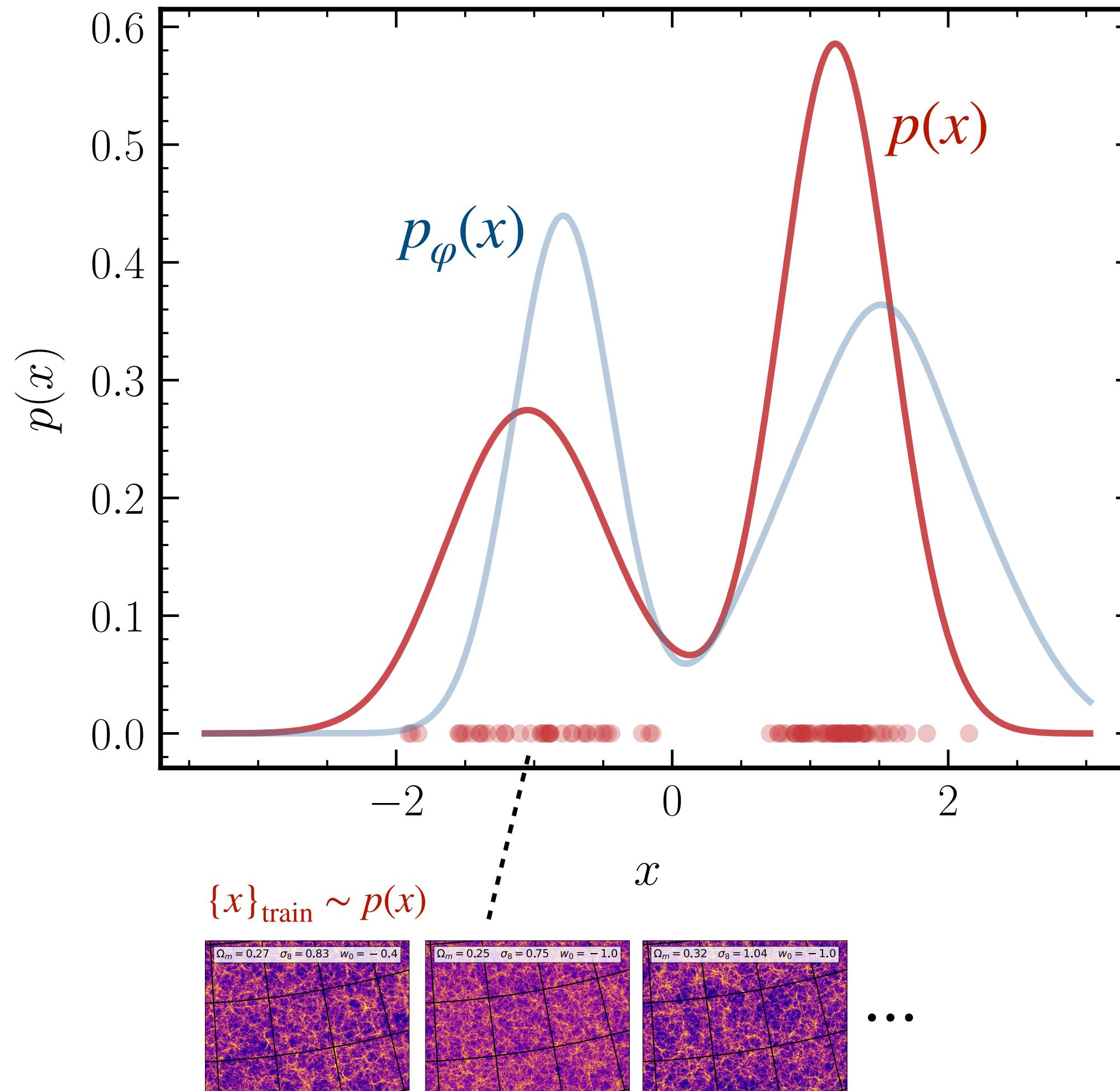
Generative modeling can efficiently enable these for a wide variety of scientific data/models!

Learning the data distribution

I'm sold! *How do I learn a generative model for my data?*

Learning the data distribution

I'm sold! *How do I learn a generative model for my data?*

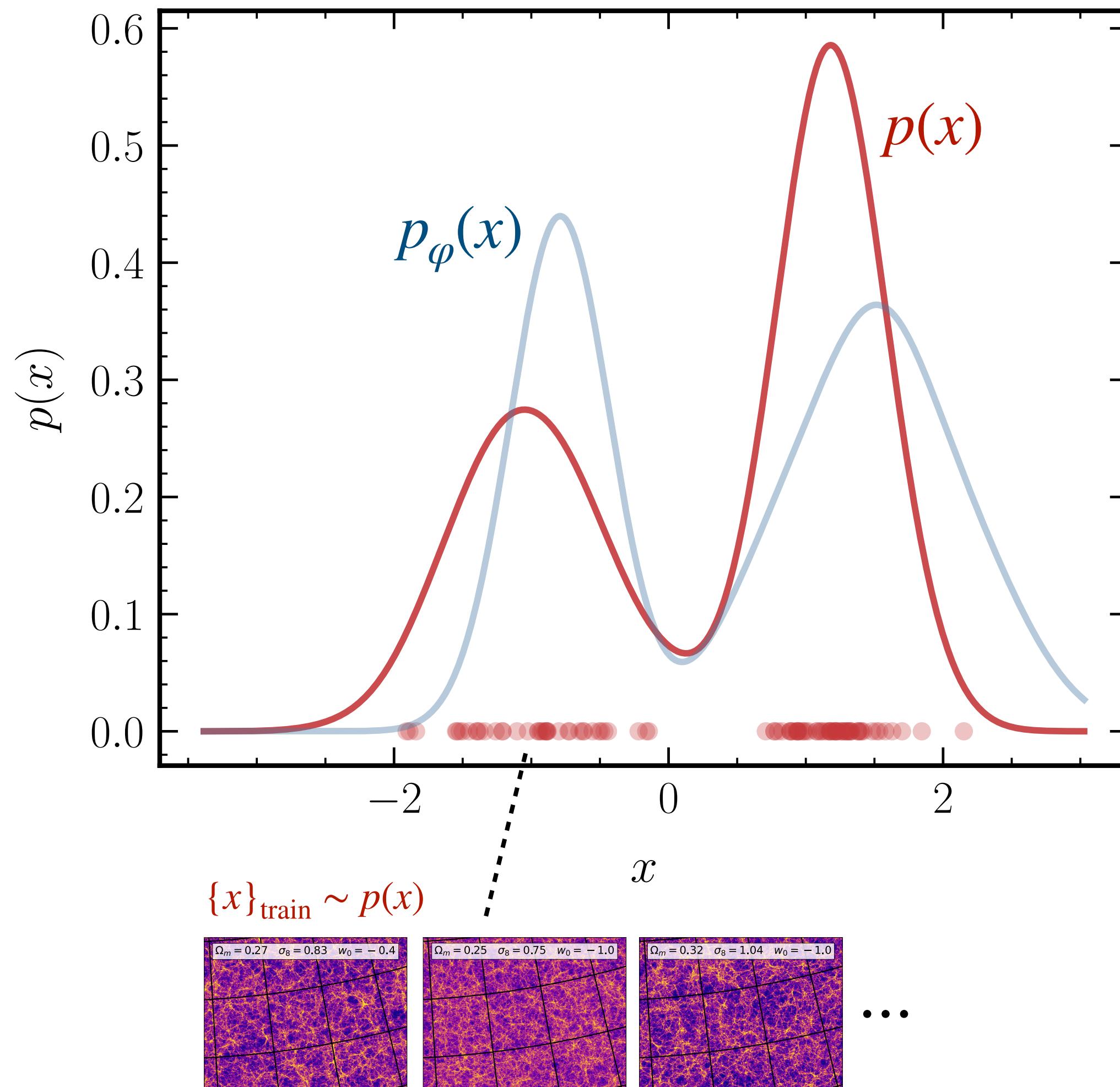


1. Ingredients:

- A parameterized distribution $p_\phi(x)$
- Samples from the data distribution $\{x\}_{\text{train}} \sim p(x)$
(empirical or simulated)

Learning the data distribution

I'm sold! *How do I learn a generative model for my data?*



1. Ingredients:

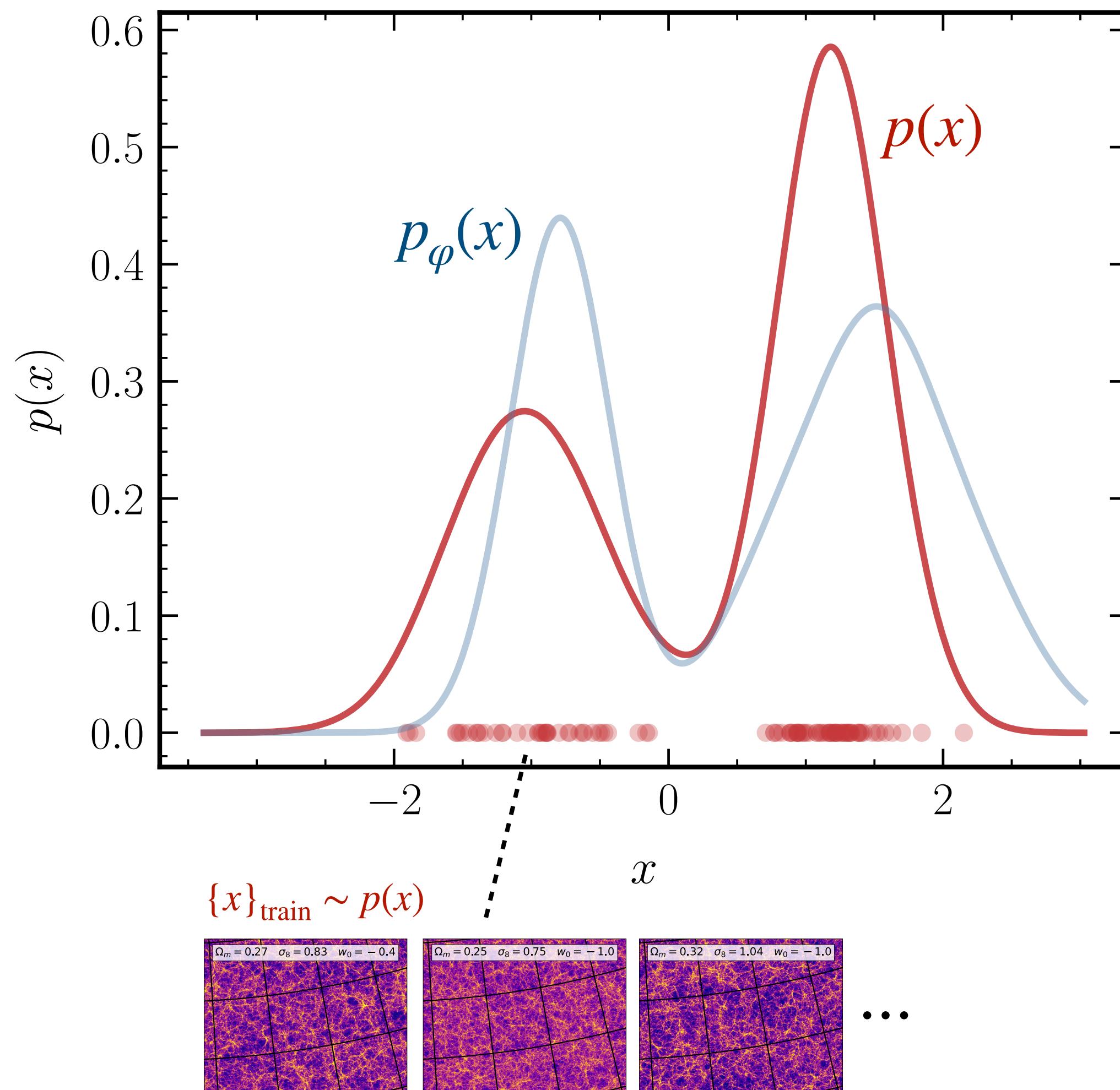
- A parameterized distribution $p_\varphi(x)$
- Samples from the data distribution $\{x\}_{\text{train}} \sim p(x)$
(empirical or simulated)

2. Maximize the likelihood of the model under the training data samples

$$\hat{\varphi} = \arg \max_{\varphi} \left[\log p_\varphi \left(\{x\}_{\text{train}} \right) \right]$$

Learning the data distribution

I'm sold! *How do I learn a generative model for my data?*



1. Ingredients:

- A parameterized distribution $p_\varphi(x)$
- Samples from the data distribution $\{x\}_{\text{train}} \sim p(x)$
(empirical or simulated)

2. Maximize the likelihood of the model under the training data samples

$$\hat{\varphi} = \arg \max_{\varphi} \left[\log p_\varphi \left(\{x\}_{\text{train}} \right) \right]$$

Not so fast...

The curse of dimensionality

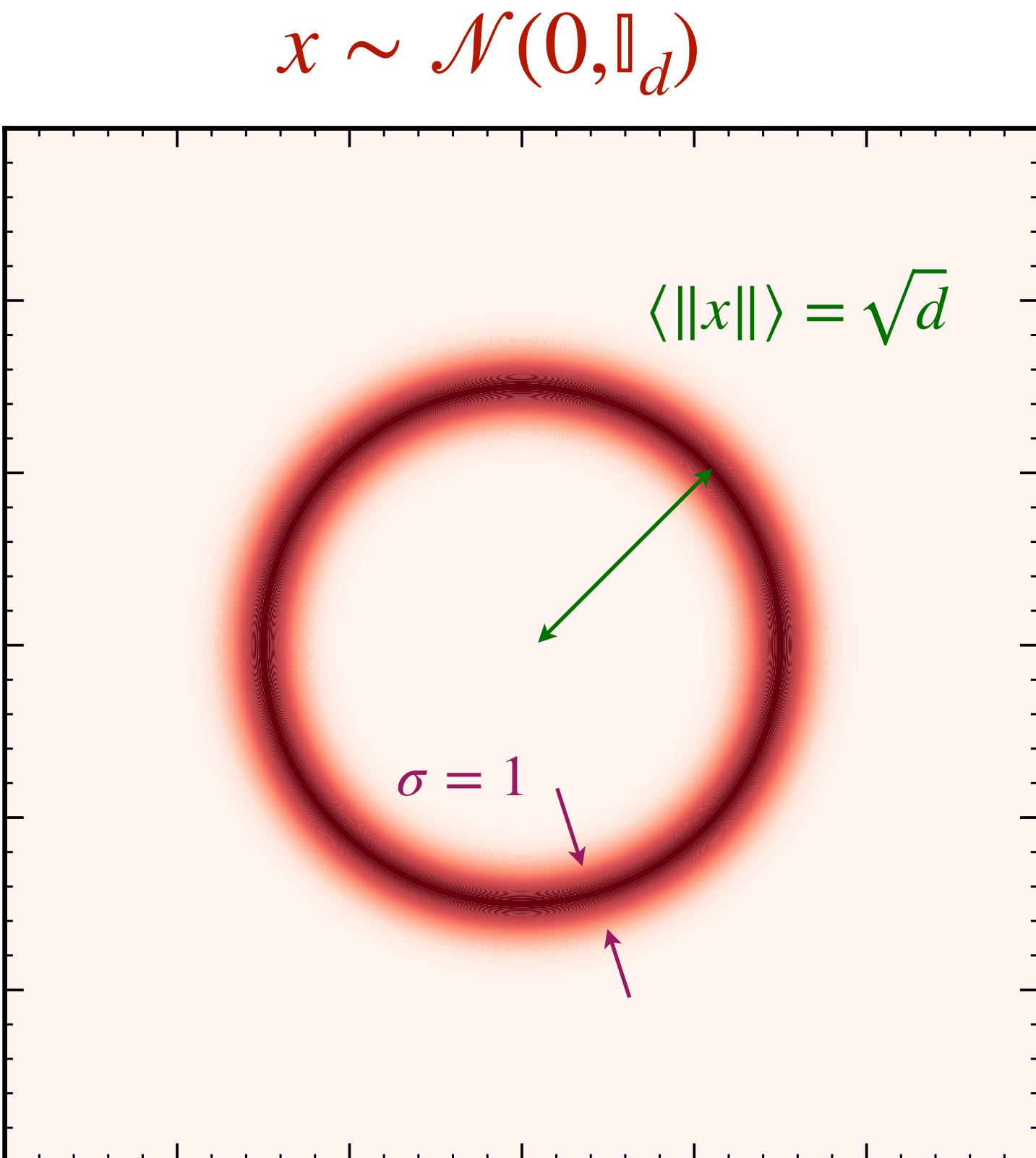
Where is most of the probability mass concentrated in high dimensions?



The curse of dimensionality

Where is most of the probability mass concentrated in high dimensions?

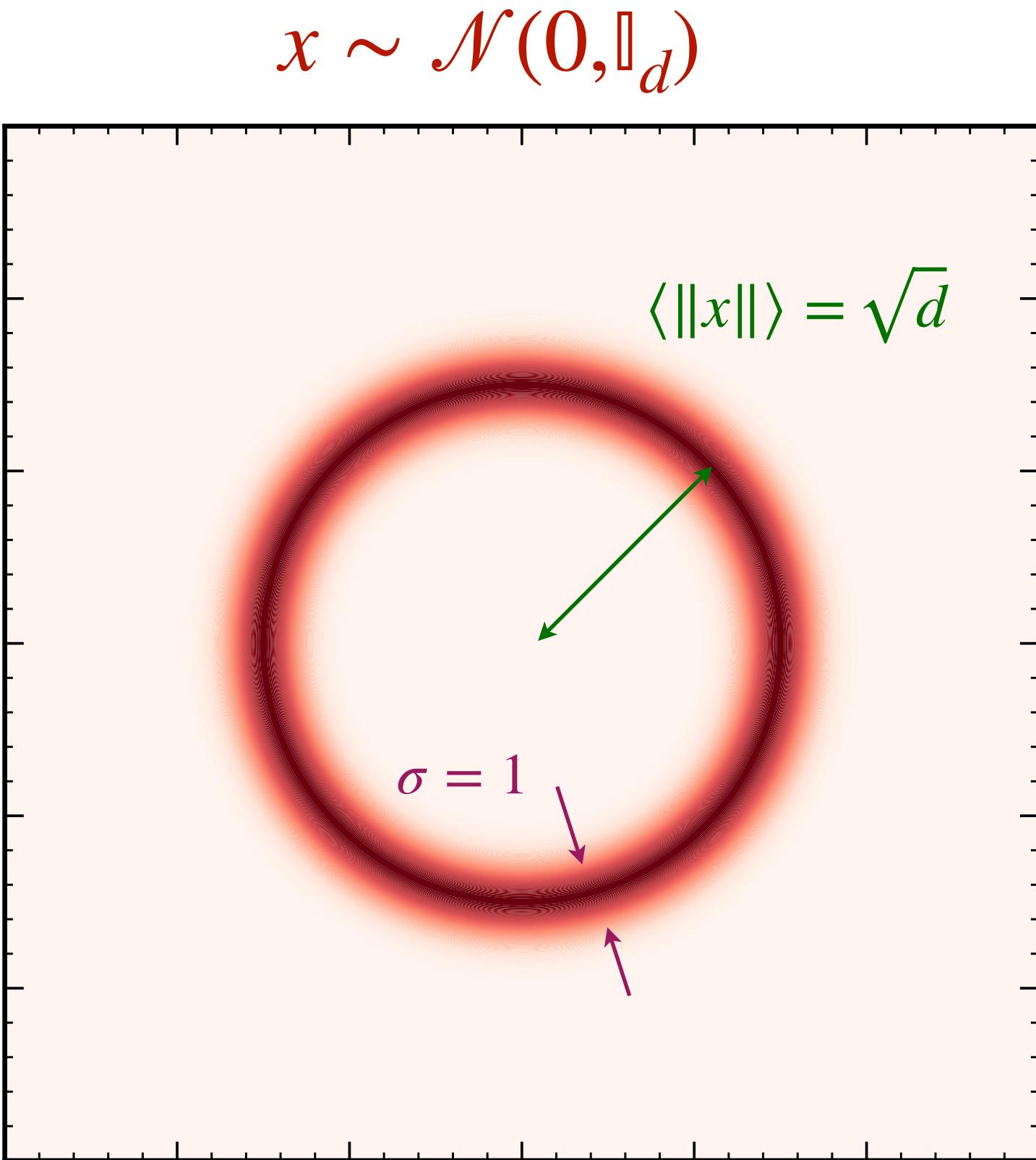
- In high dimensions, most of the probability density of a Gaussian distribution lies in a thin shell at distance \sqrt{d} from the center
- Vanishingly small fraction of distribution support is actually occupied.



The curse of dimensionality

Where is most of the probability mass concentrated in high dimensions?

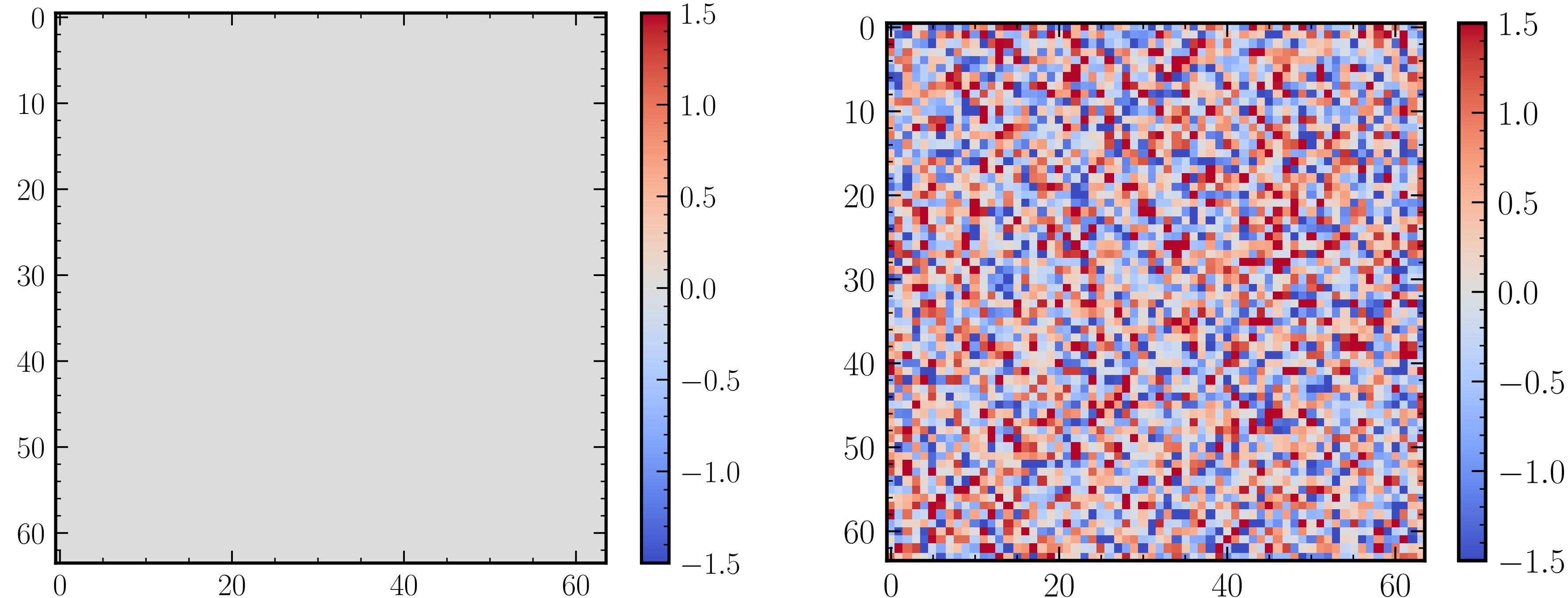
- In high dimensions, most of the probability density of a Gaussian distribution lies in a thin shell at distance \sqrt{d} from the center
- Vanishingly small fraction of distribution support is actually occupied.



Learning high-dimensional distributions is challenging!

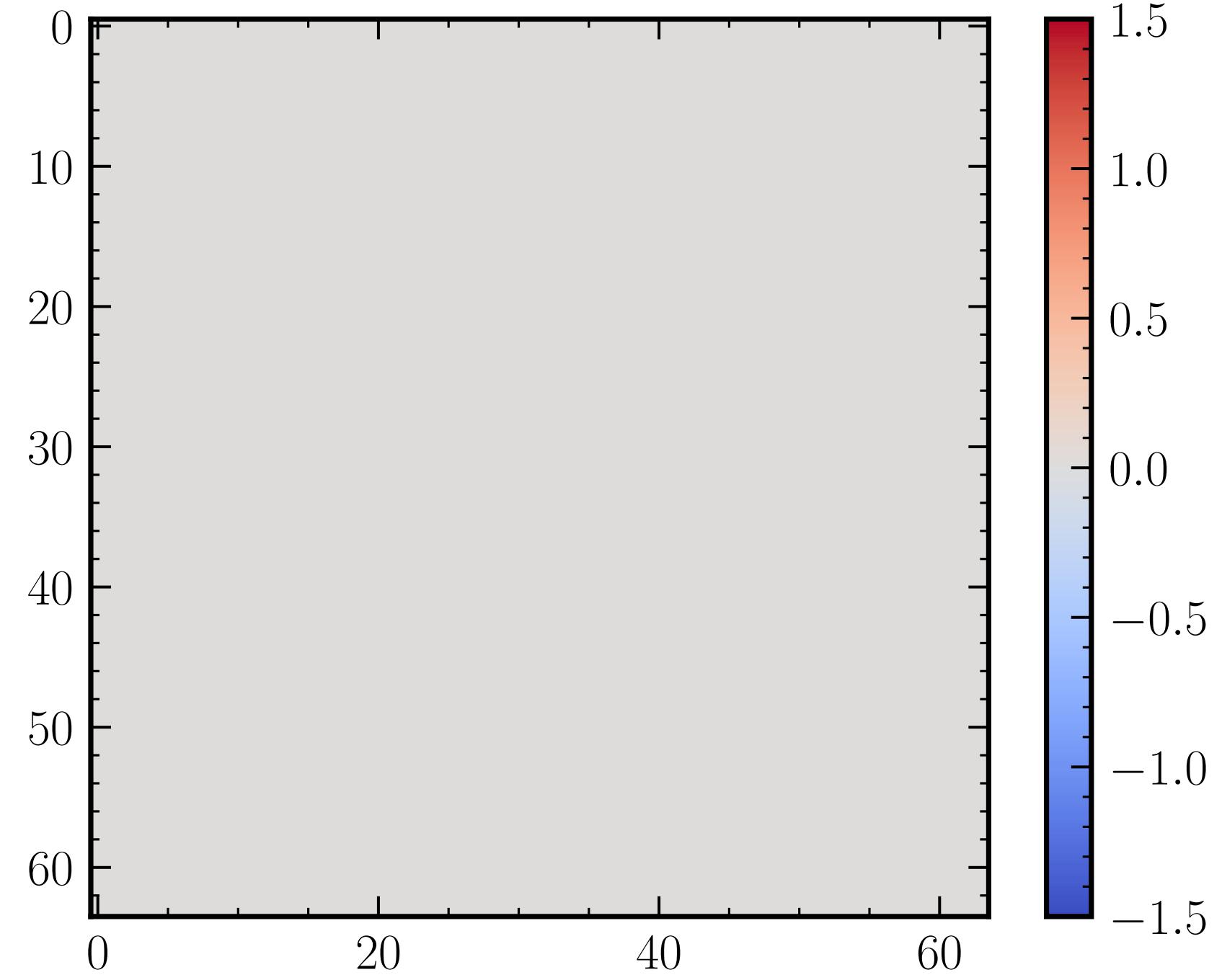
Typicality and likelihood of samples

Which of these samples have a higher likelihood under $\mathcal{L} = \mathcal{N}(0, \mathbb{I}_d)$?

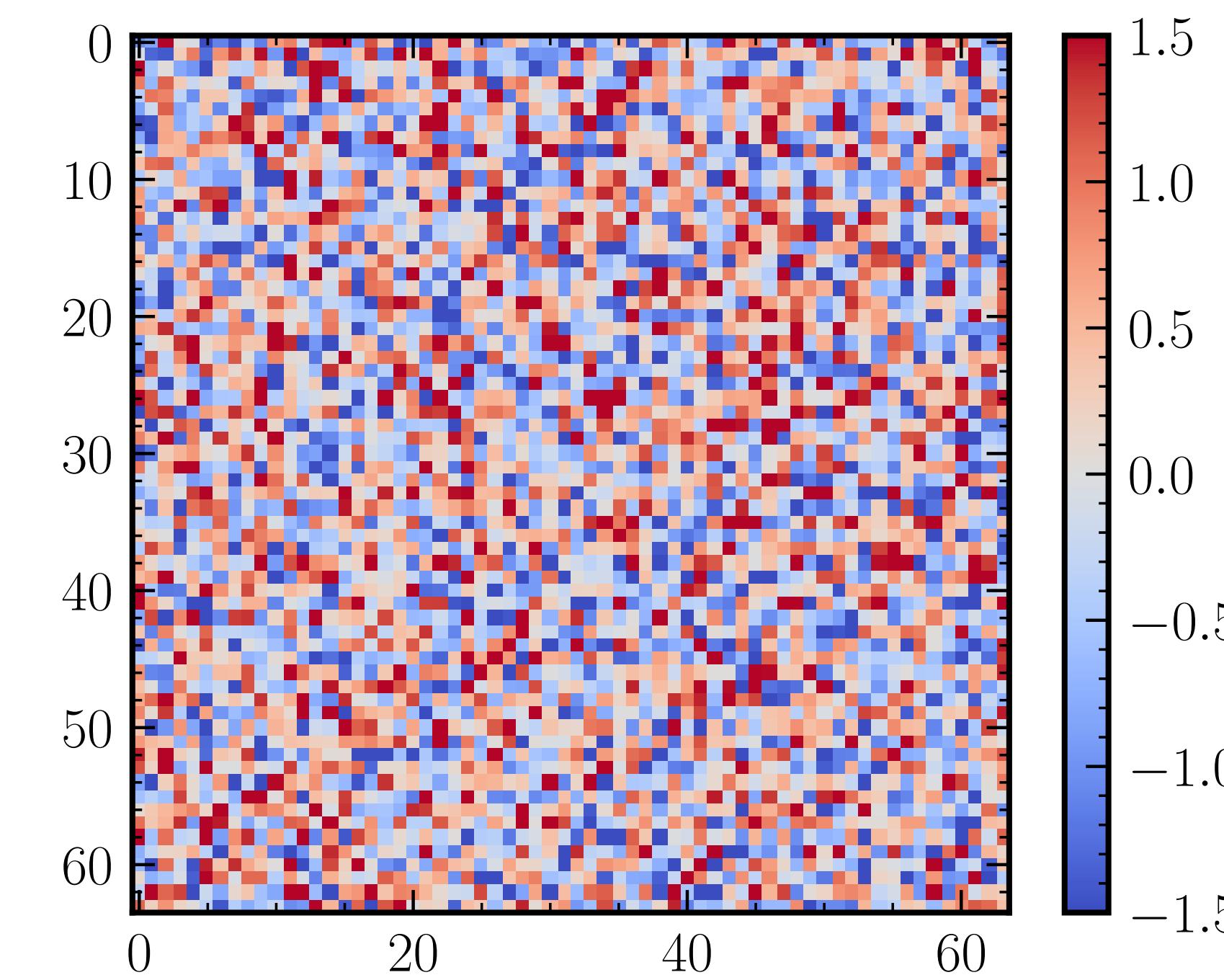


Typicality and likelihood of samples

Which of these samples have a higher likelihood under $\mathcal{L} = \mathcal{N}(0, \mathbb{I}_d)$?



$$\log \mathcal{L} \approx -0.92 \text{ nats/dim}$$

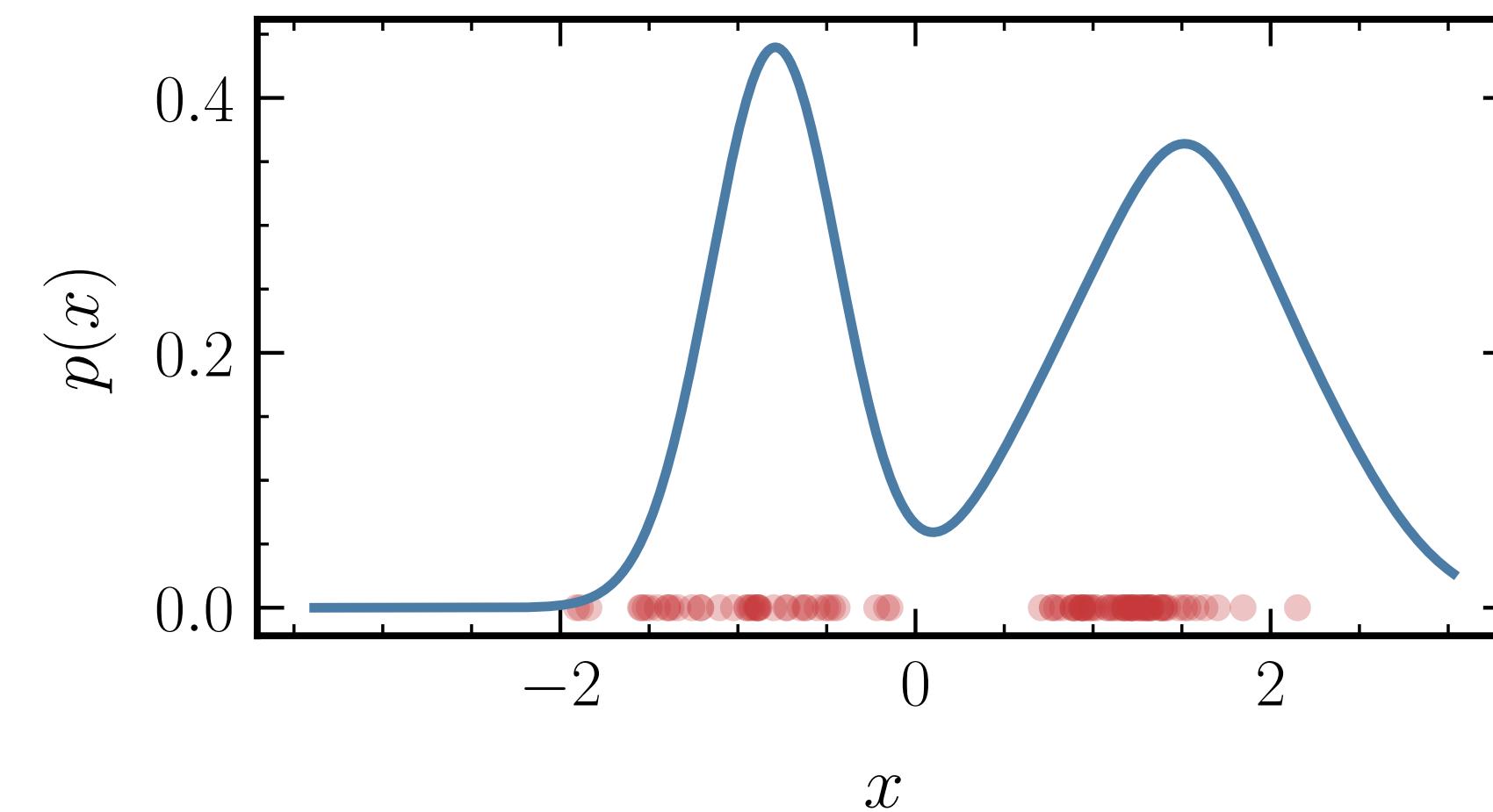


$$\log \mathcal{L} \approx -1.43 \text{ nats/dim}$$

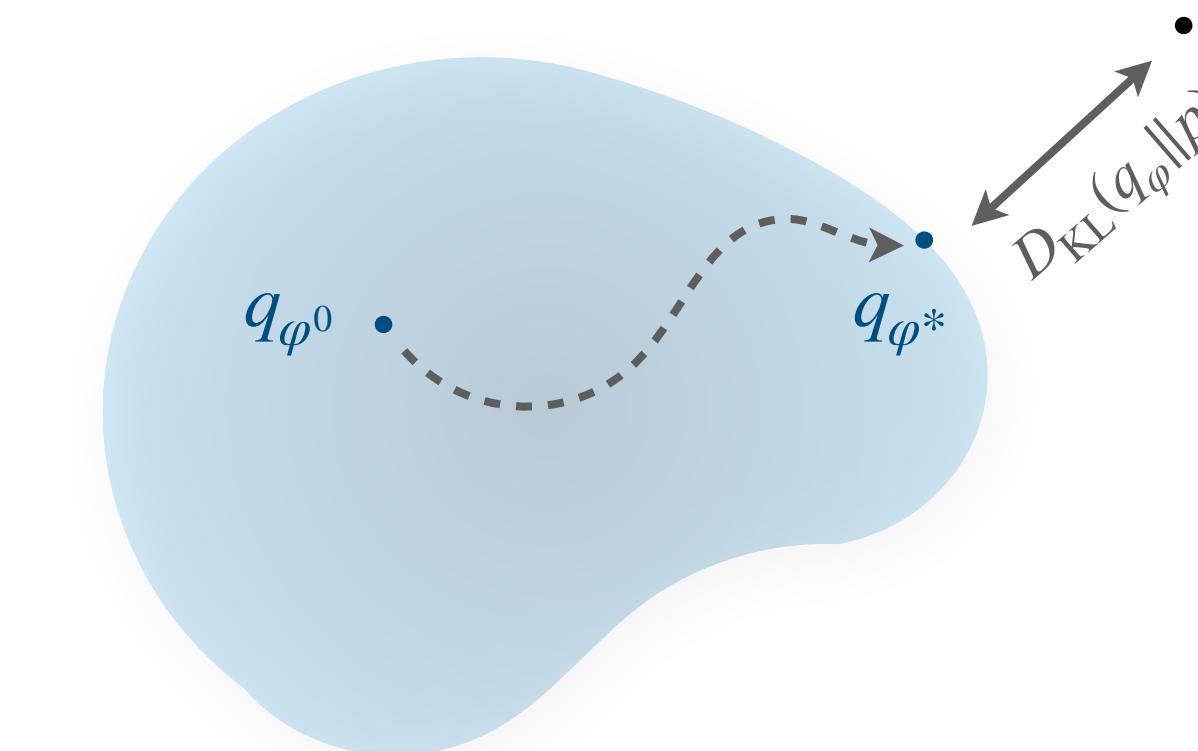
Evaluation of high-dimensional distributions is challenging!

(Some) Ways of training deep generative models

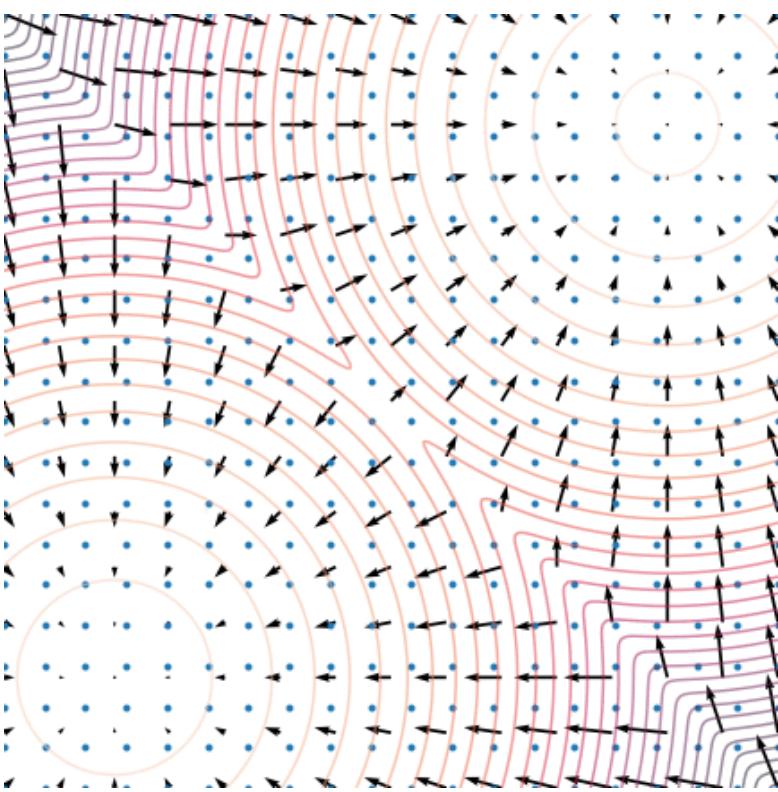
Maximum-likelihood



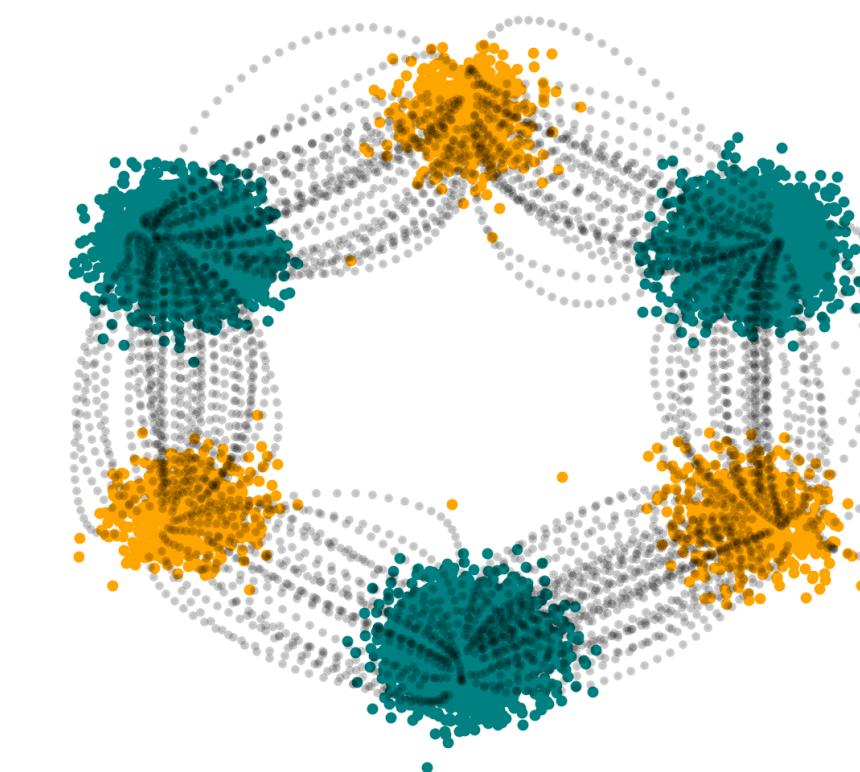
Optimizing a bound on the likelihood



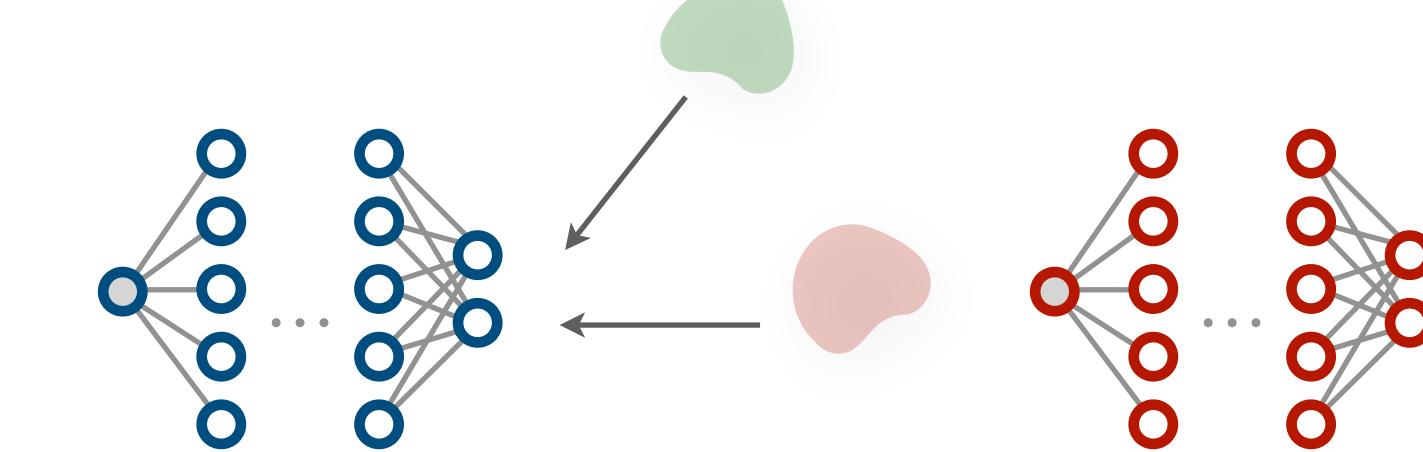
Score-matching



Optimal transport

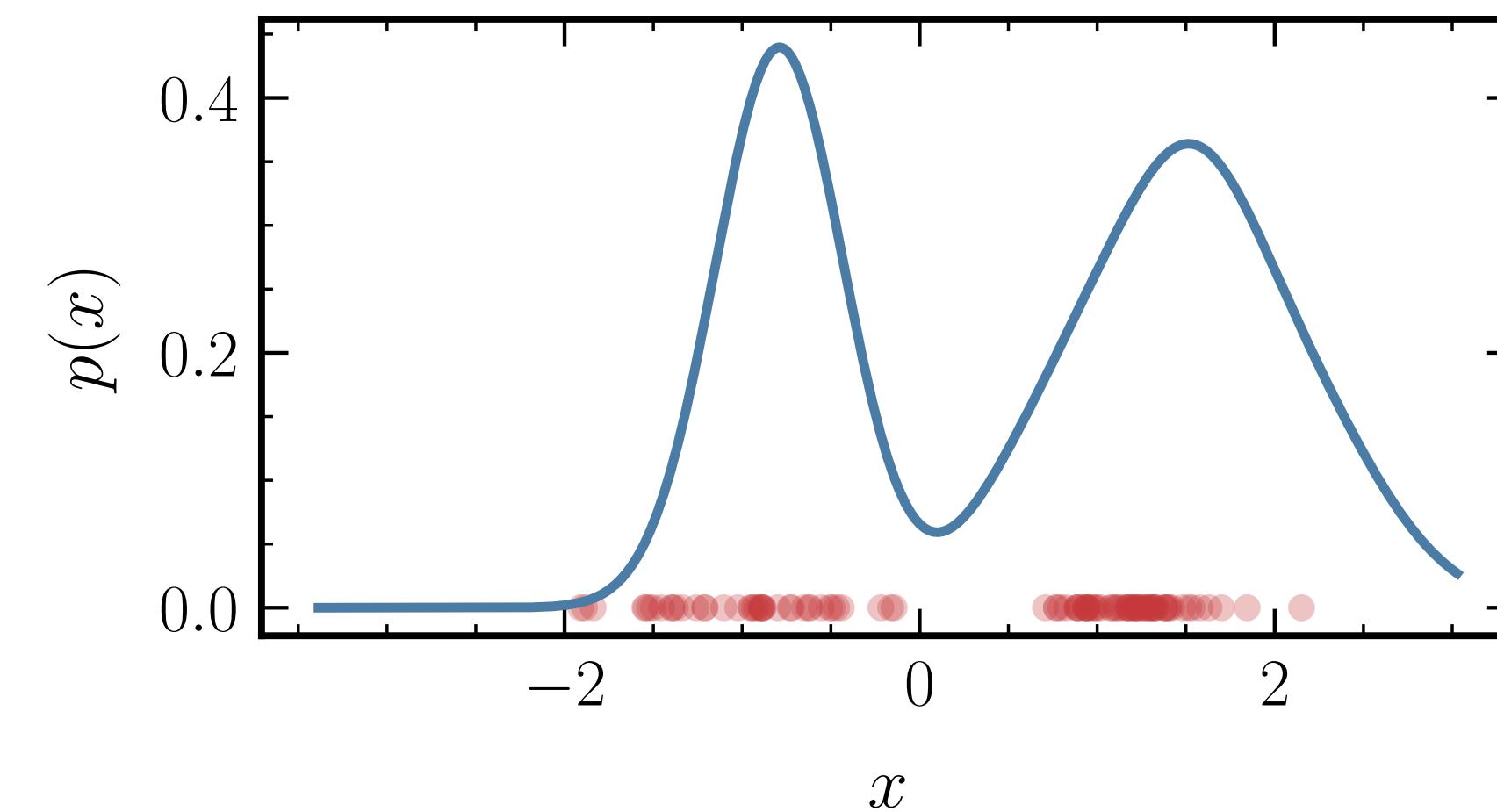


Adversarial training

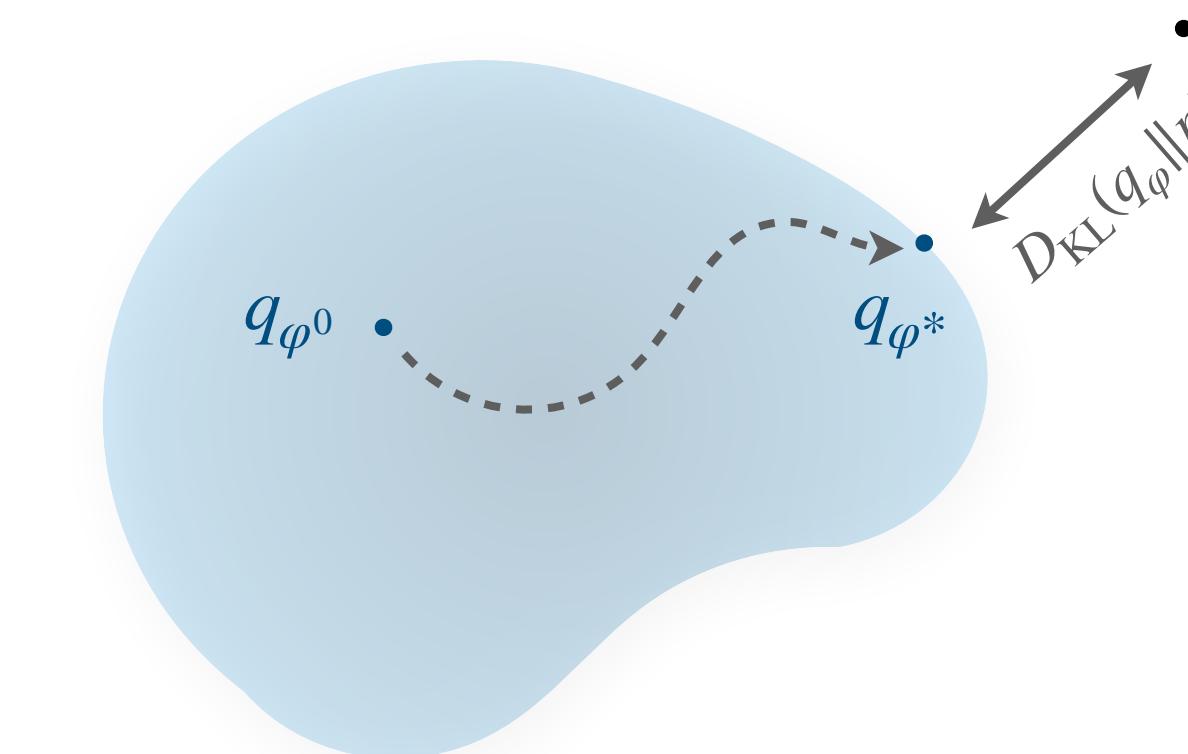


(Some) Ways of training deep generative models

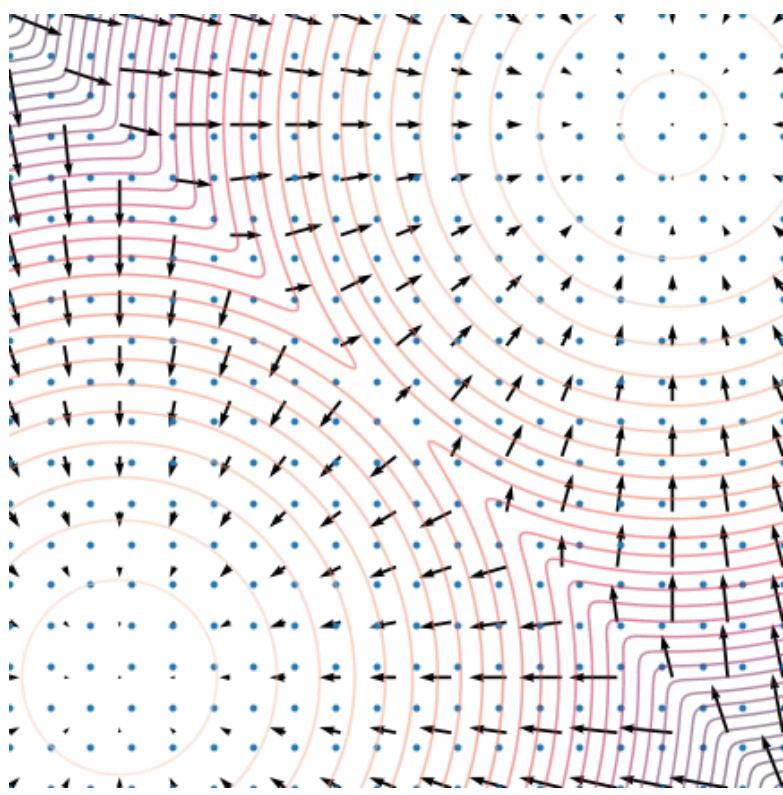
Maximum-likelihood



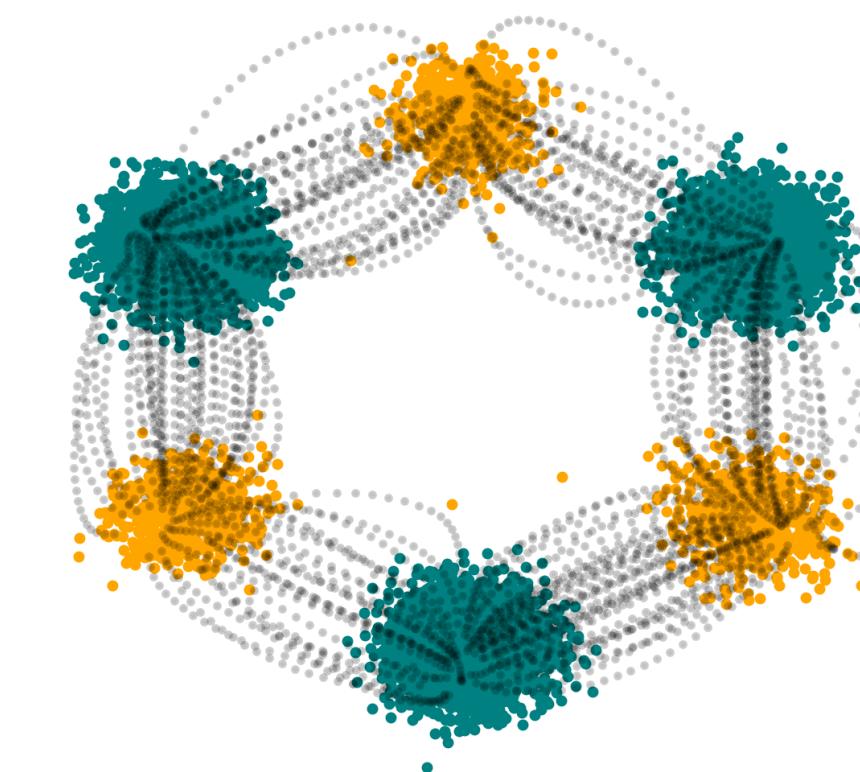
Optimizing a bound on the likelihood



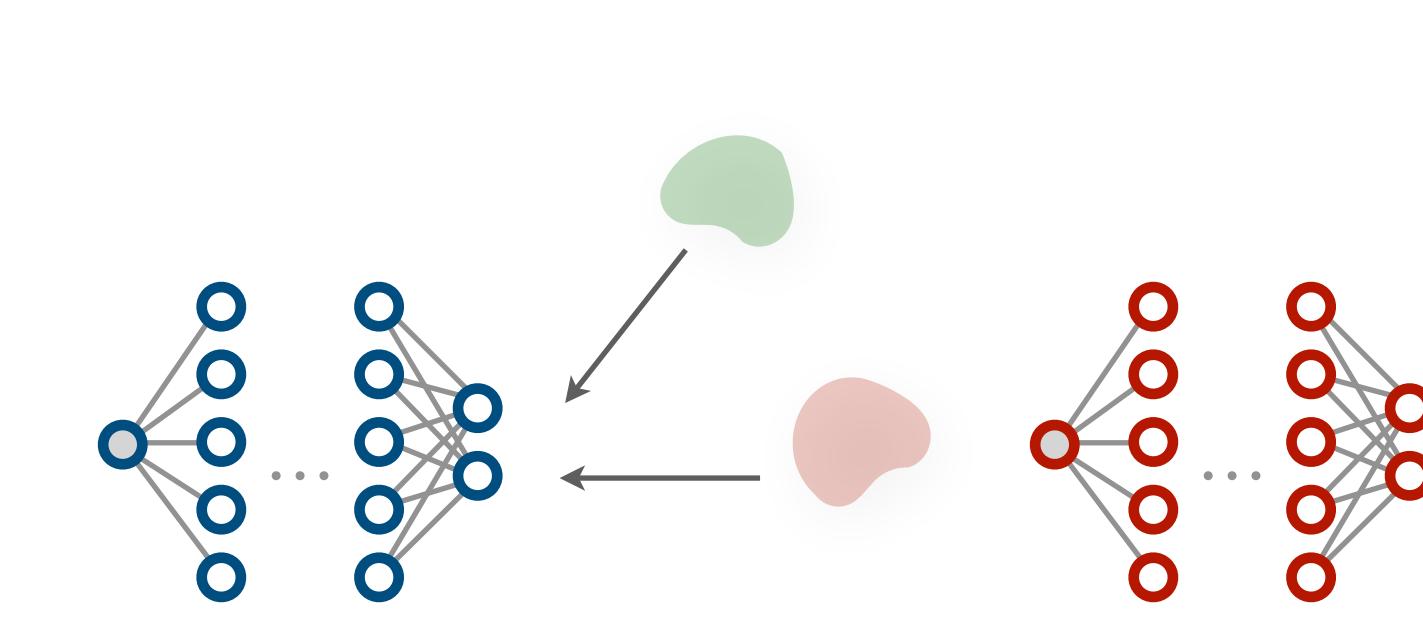
Score-matching



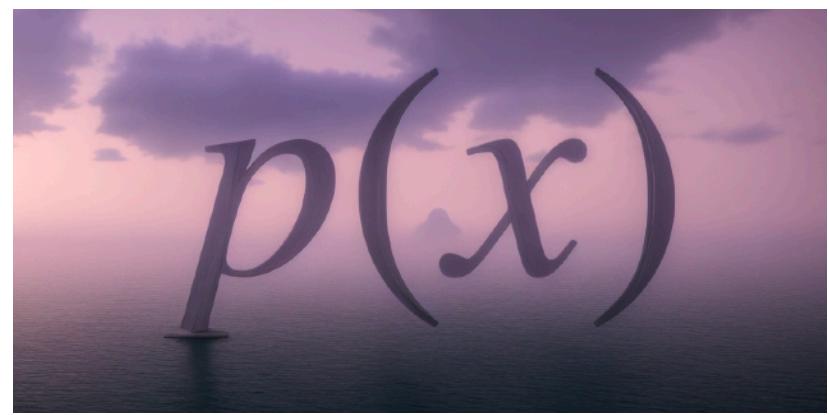
Optimal transport



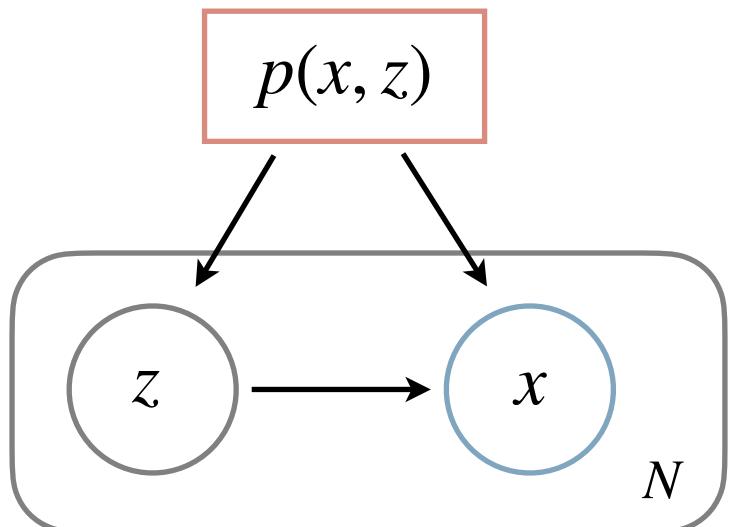
Adversarial training



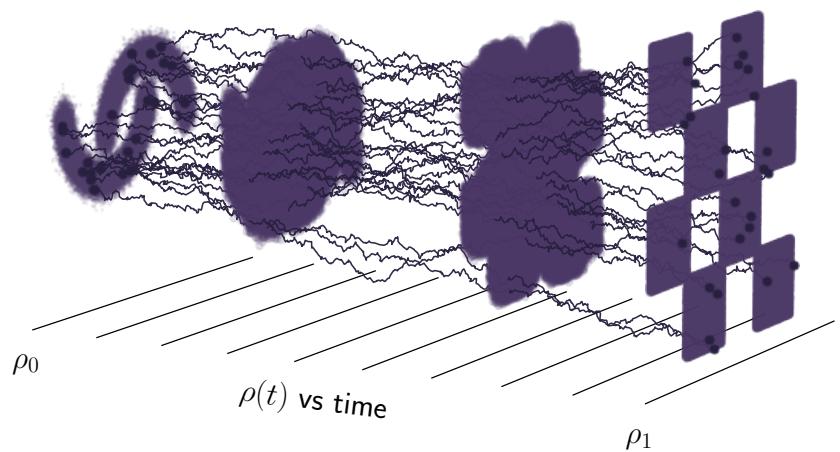
Outline



Why (deep) generative modeling?
What is it, and what can it do for you?



Variational auto encoders
Latent-variable modeling, and compression is all you need

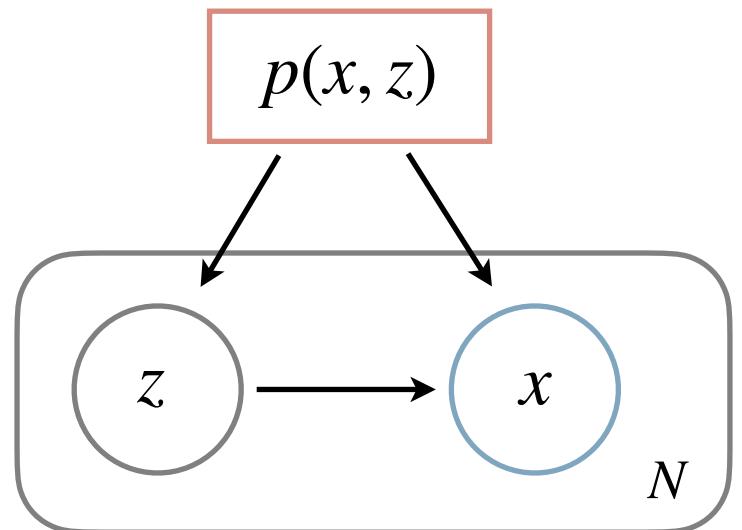


Diffusion models
Models based on iterative refinement



Normalizing flows
Invertible transformations

Outline

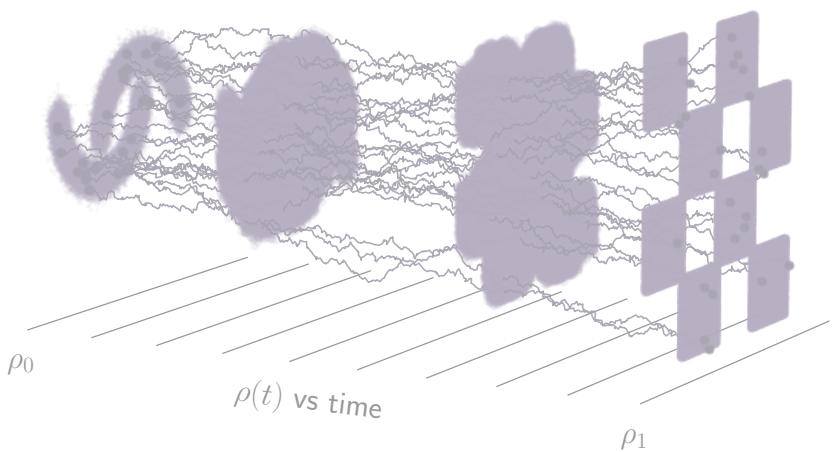


Why (deep) generative modeling?

What is it, and what can it do for you?

Variational auto encoders

Latent-variable modeling, and compression is all you need



Diffusion models

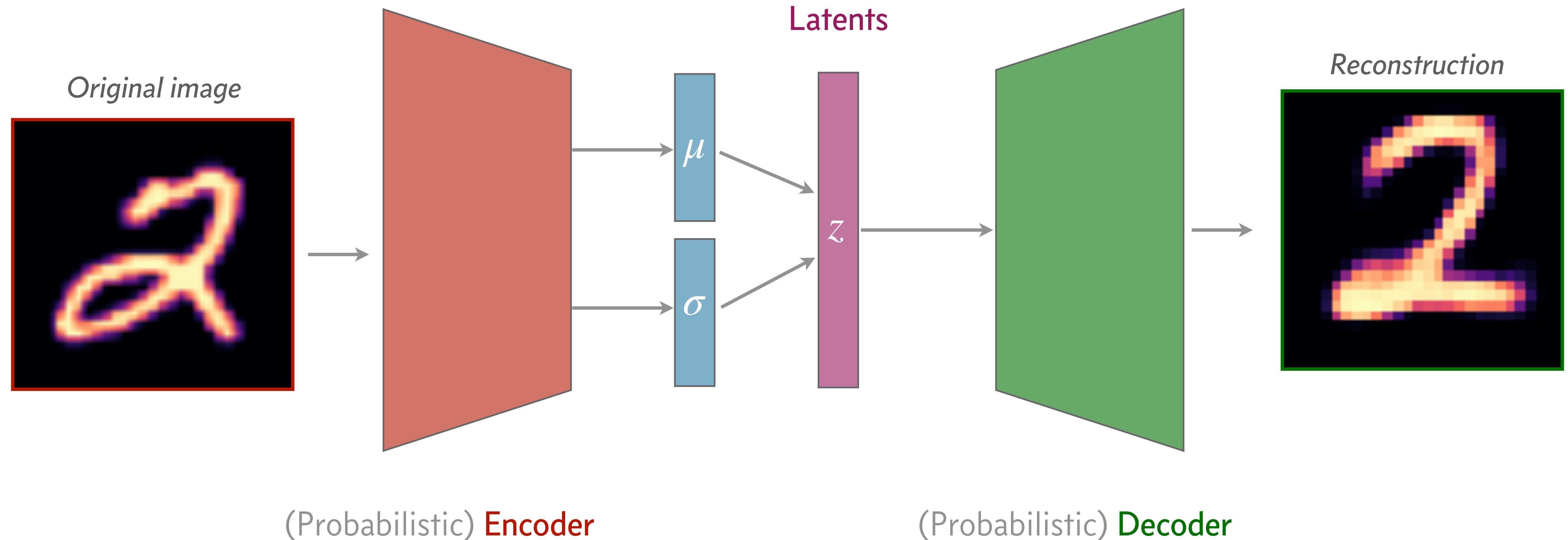
Models based on iterative refinement



Normalizing flows

Invertible transformations

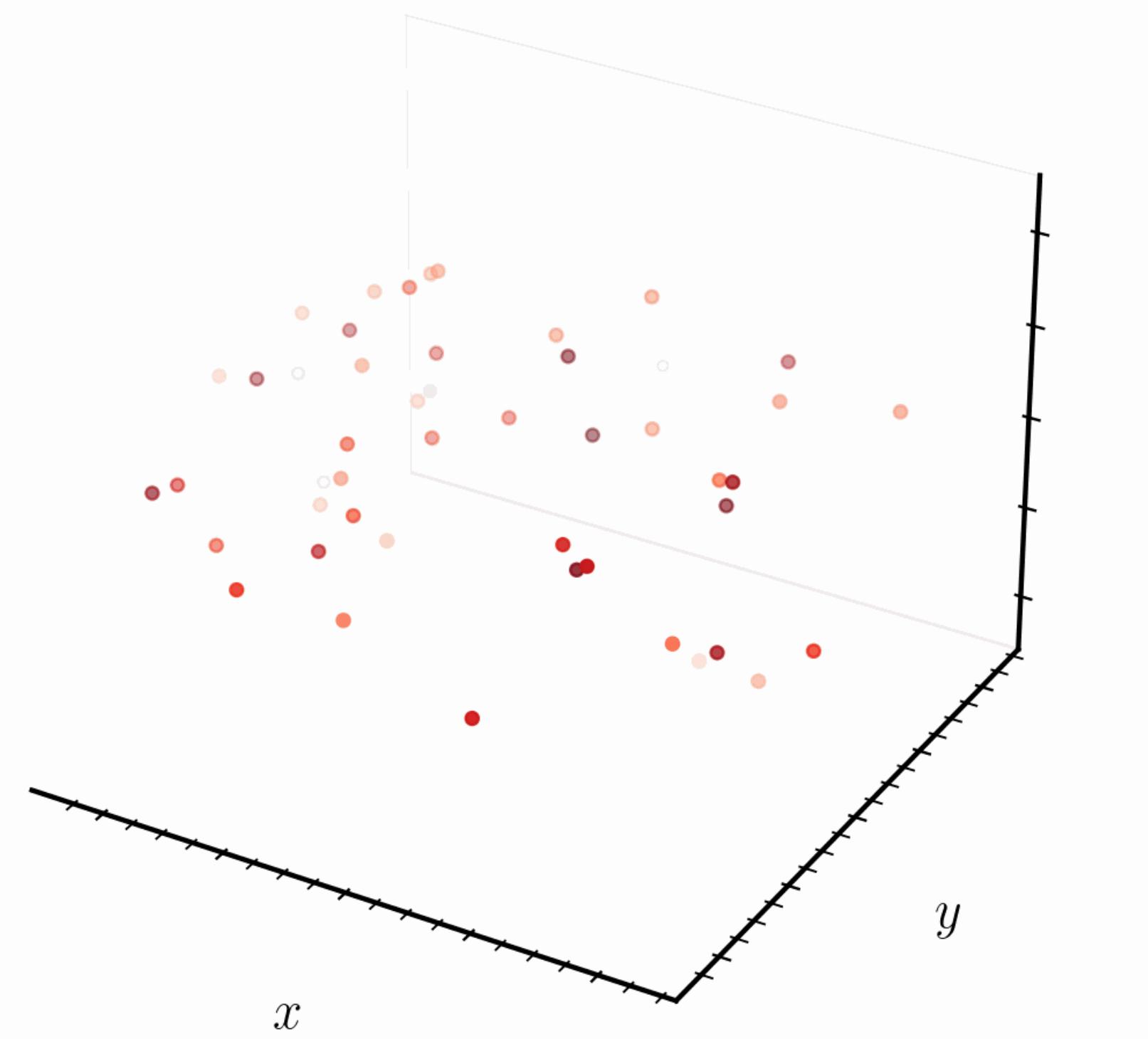
A bird-eye view



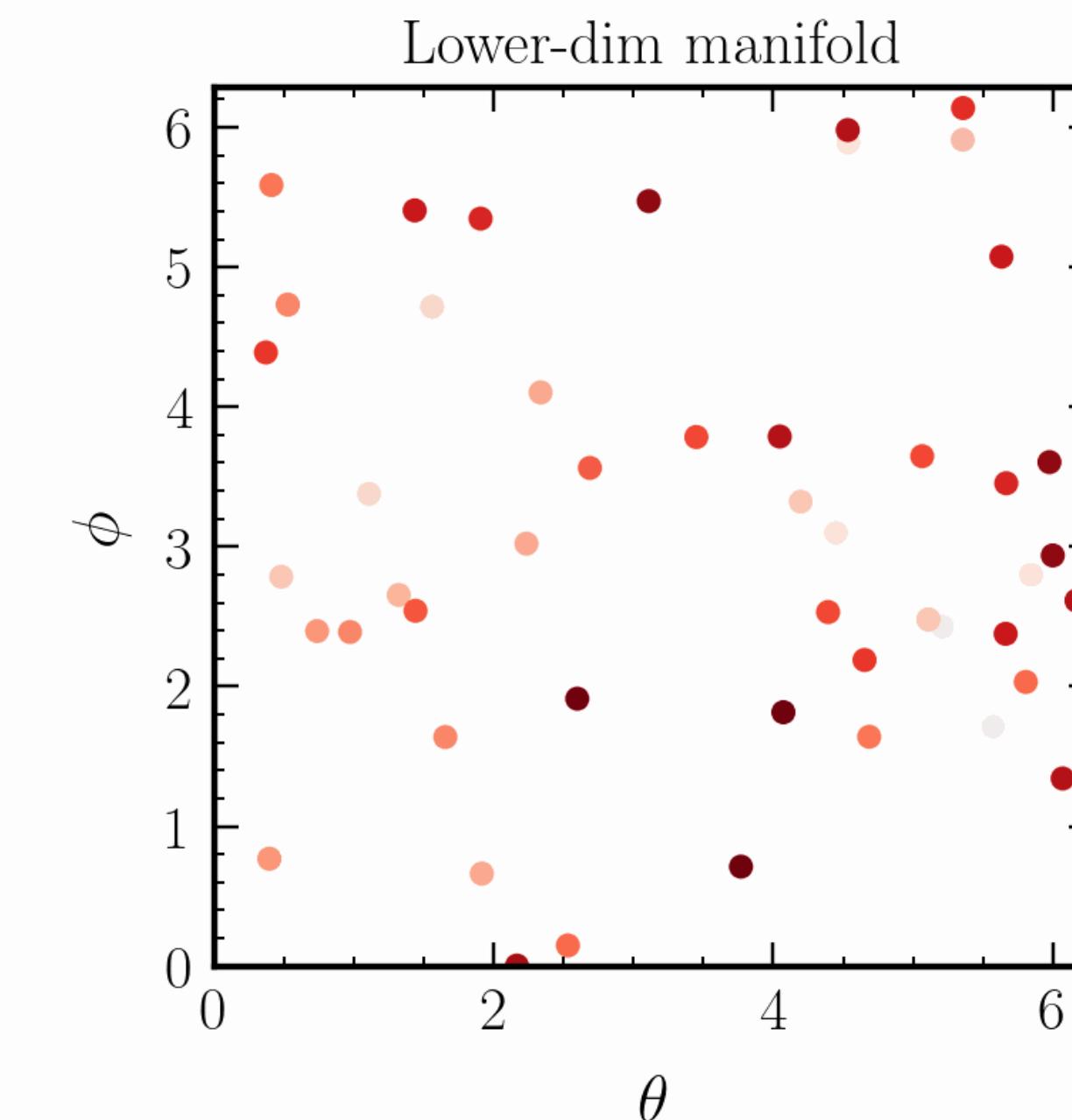
The pursuit of low-dimensional structure

Real-world datasets often live in structured low-dimensional manifolds

“Difficult to model” x



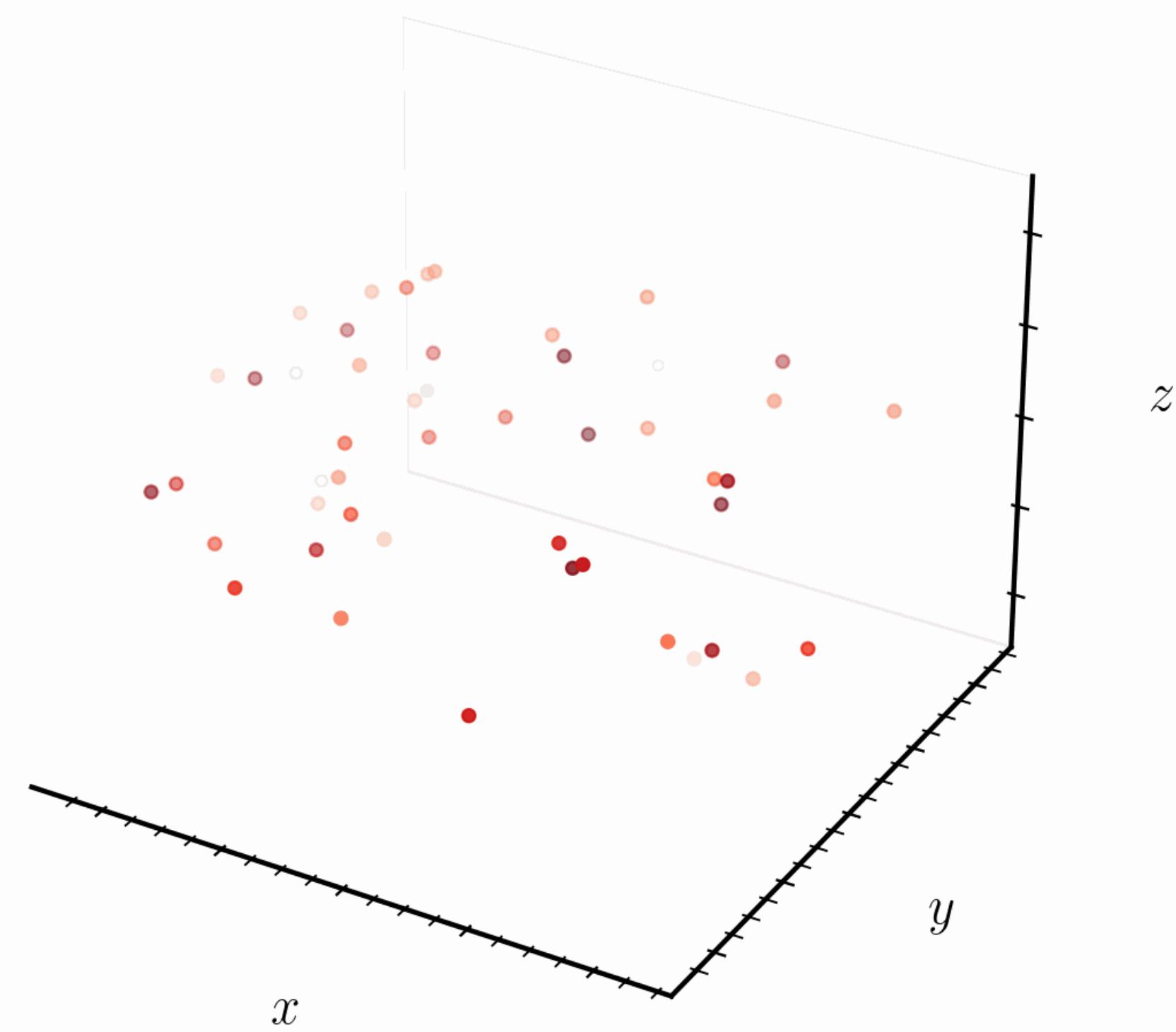
“Easy to model” z



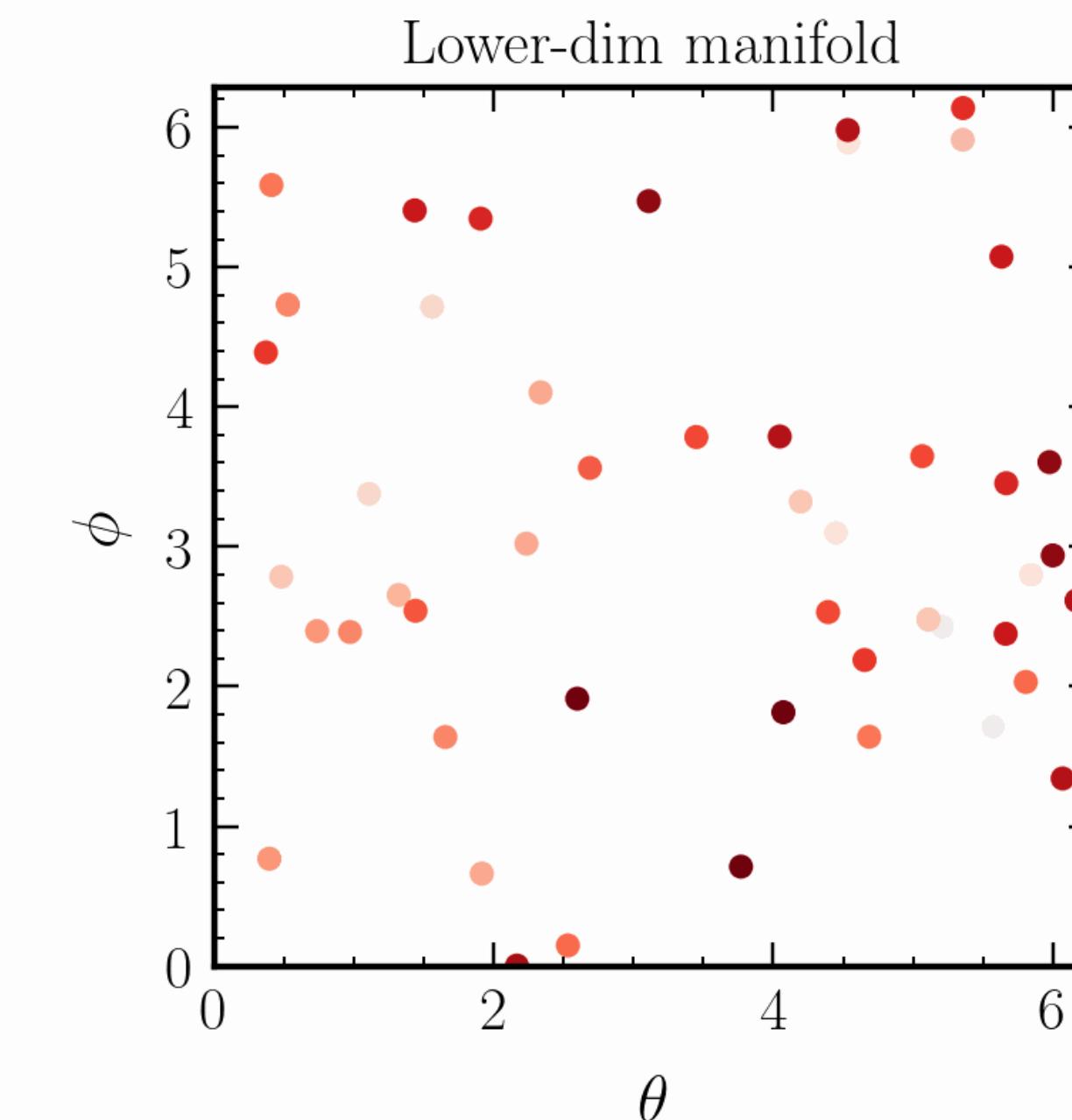
The pursuit of low-dimensional structure

Real-world datasets often live in structured low-dimensional manifolds

“Difficult to model” x



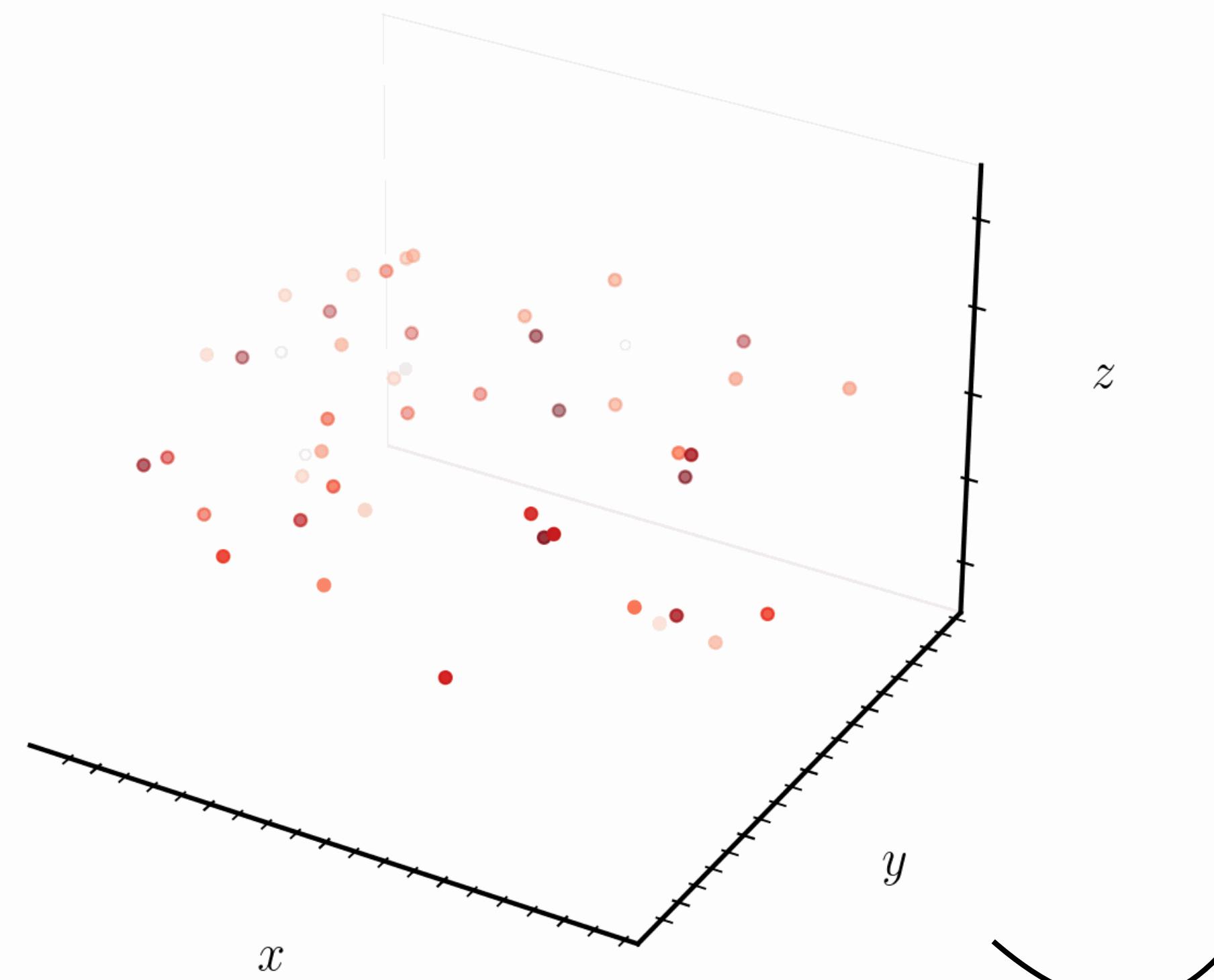
“Easy to model” z



The pursuit of low-dimensional structure

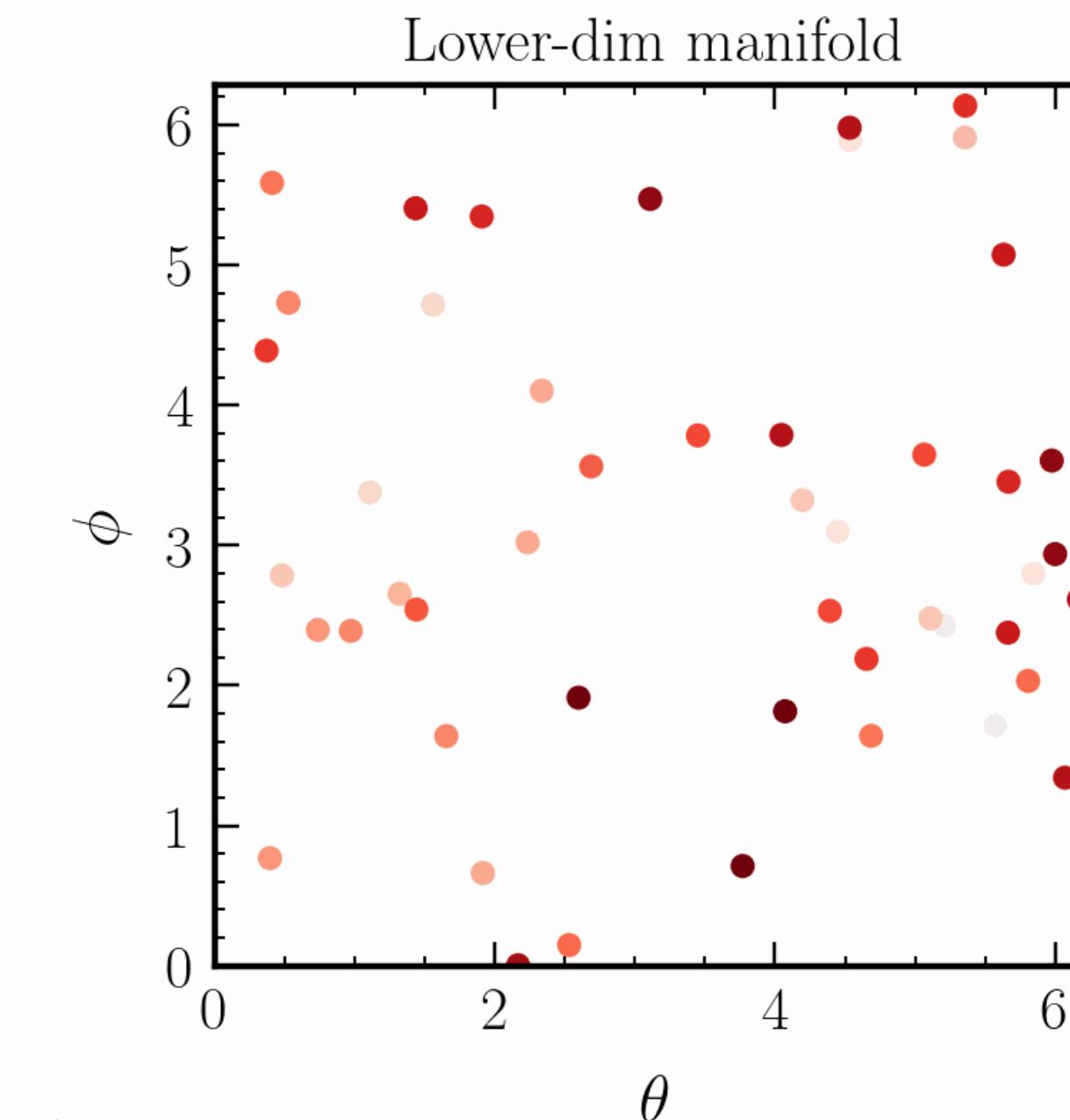
Real-world datasets often live in structured low-dimensional manifolds

“Difficult to model” x



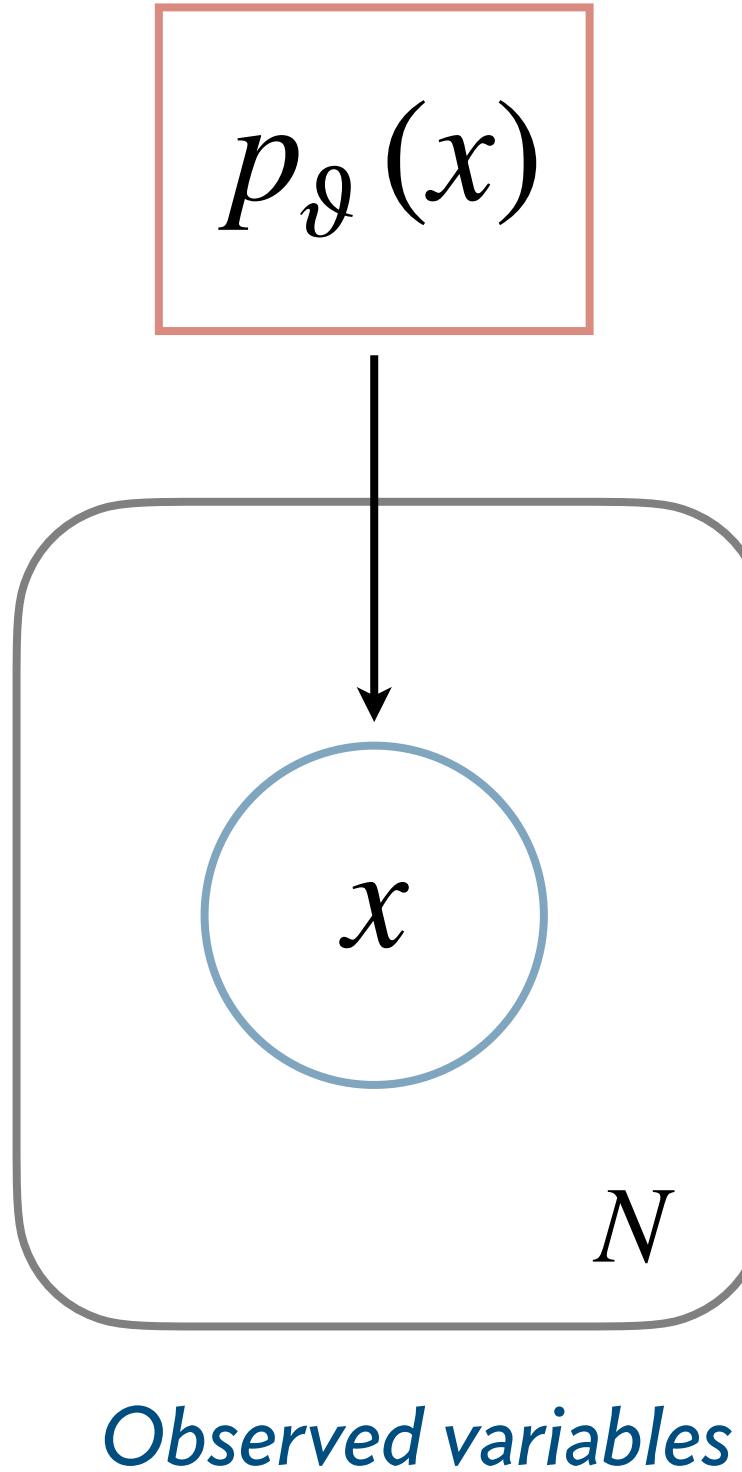
$$p(z \mid x)$$

“Easy to model” z



Latent-variable modeling

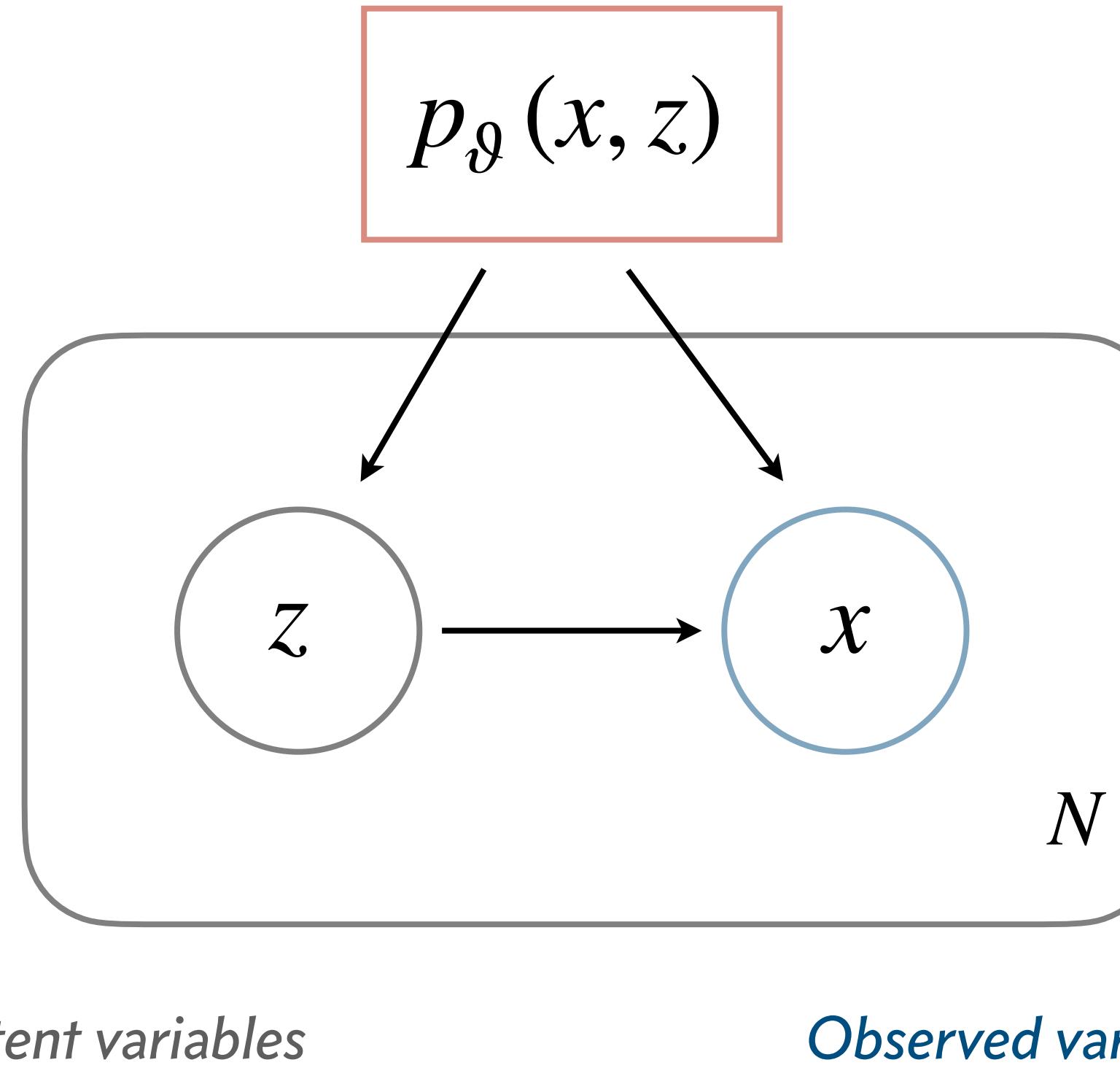
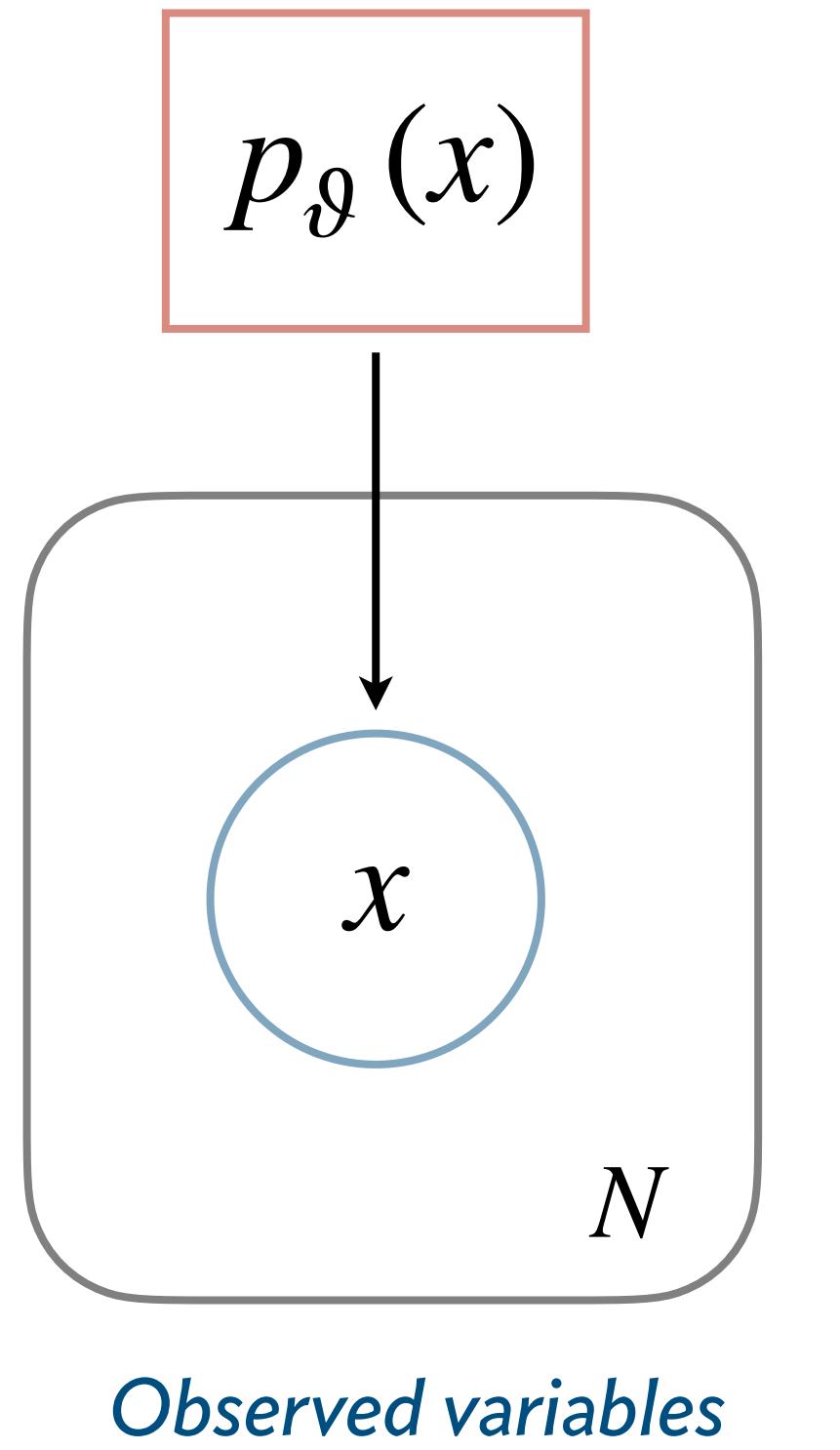
Learn lower-dimensional structure in the data distribution



Latent-variable modeling

Learn lower-dimensional structure in the data distribution

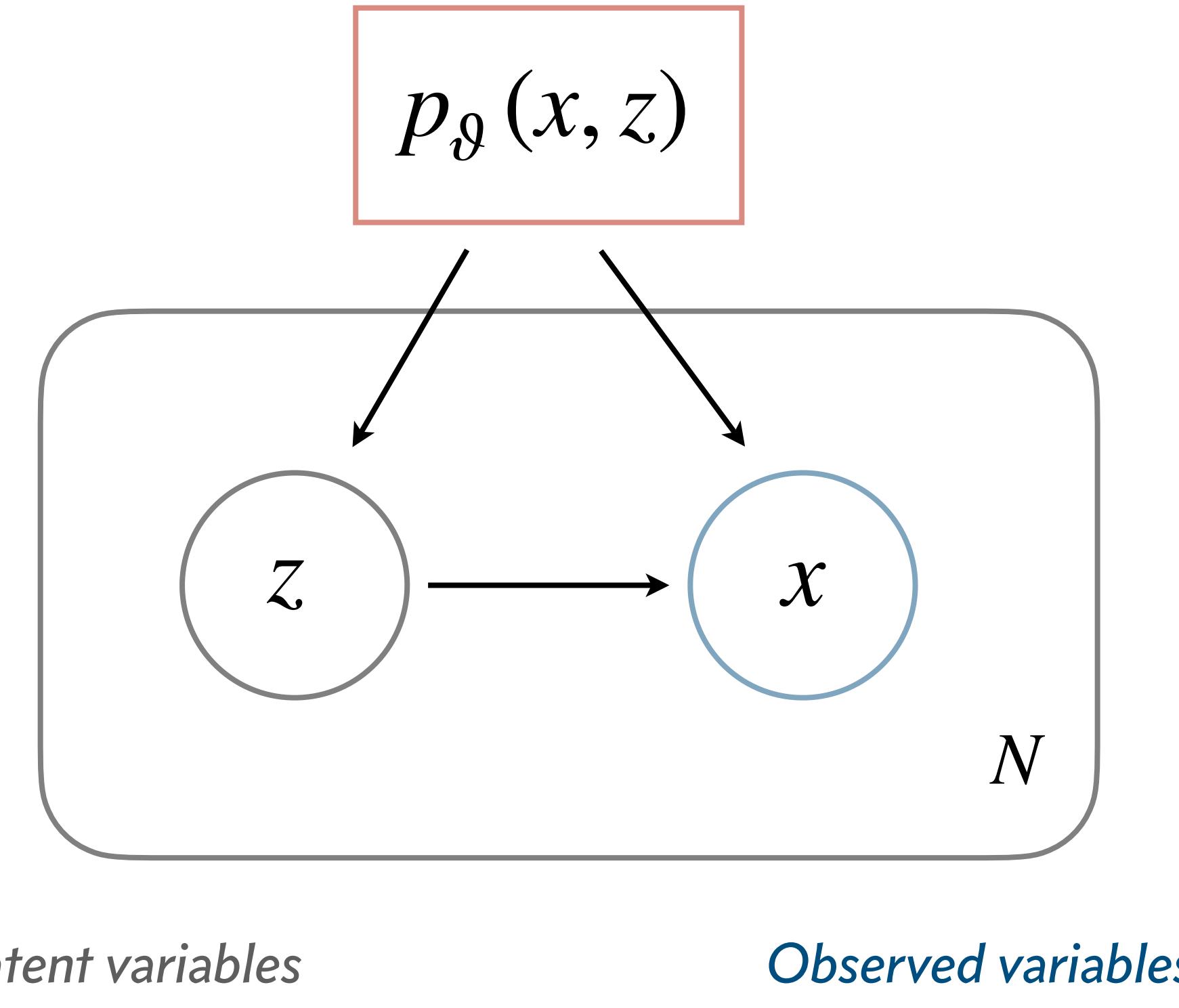
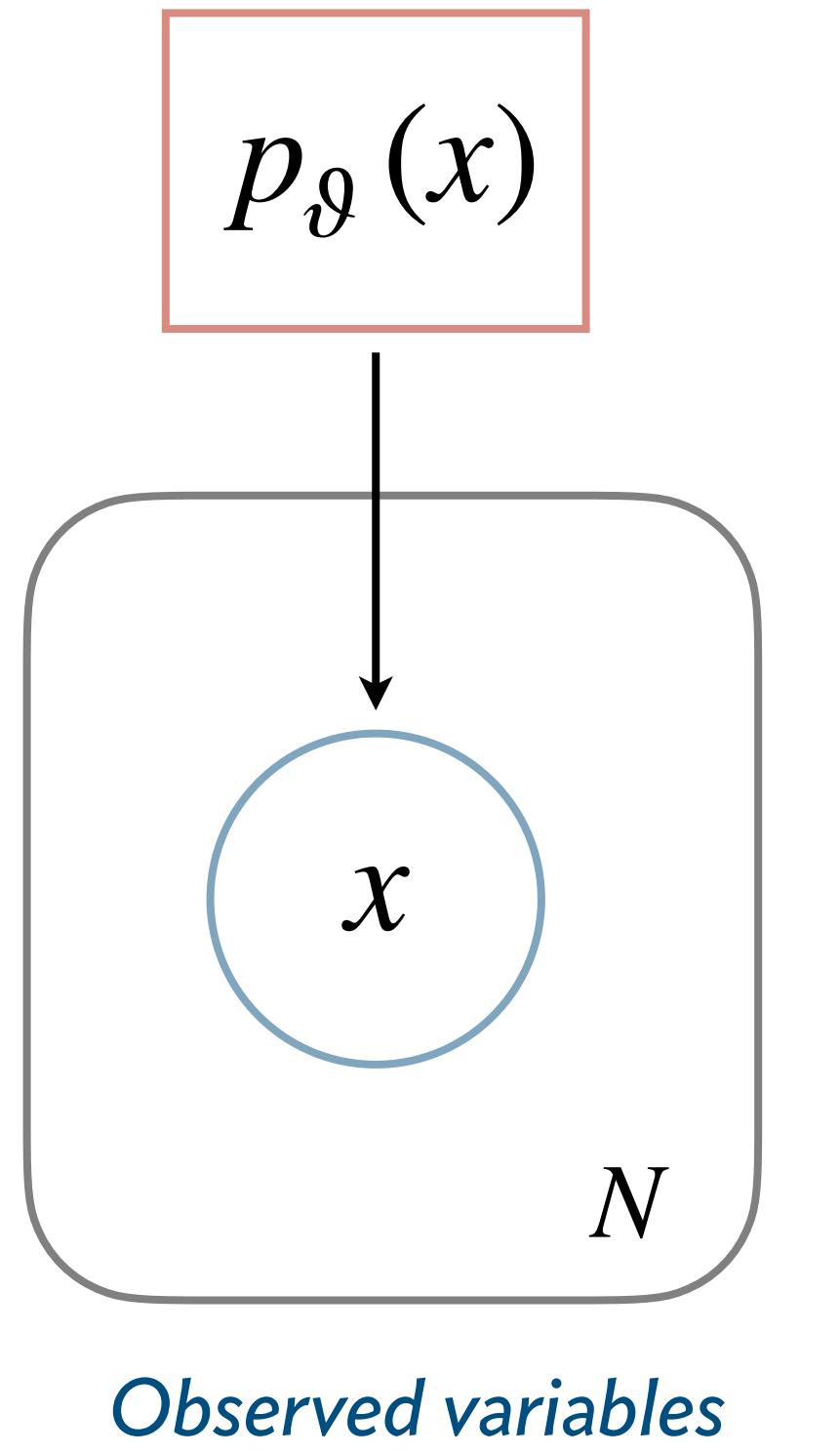
Make the problem easier by making it “harder”: introduce *joint distribution* $p_\theta(x, z)$



Latent-variable modeling

Learn lower-dimensional structure in the data distribution

Make the problem easier by making it “harder”: introduce *joint distribution* $p_\theta(x, z)$



Common factorization:

$$p_\theta(x, z) = p(z) \cdot p_\theta(x | z)$$

Latent-variable modeling

Maximum-likelihood training?

$$\begin{aligned}\vartheta^* &= \arg \max_{\vartheta} p_{\vartheta}(x) \\ &= \arg \max_{\vartheta} \int p_{\vartheta}(x \mid z)p(z) \, dz \\ &= \arg \max_{\vartheta} \left\langle p_{\vartheta}(x \mid z) \right\rangle_{p(z)}\end{aligned}$$

Latent-variable modeling

Maximum-likelihood training?

$$\begin{aligned}\vartheta^* &= \arg \max_{\vartheta} p_{\vartheta}(x) \\ &= \arg \max_{\vartheta} \int p_{\vartheta}(x \mid z)p(z) \, dz \\ &= \arg \max_{\vartheta} \left\langle p_{\vartheta}(x \mid z) \right\rangle_{p(z)}\end{aligned}$$

Difficult to build a good estimator!

Latent-variable modeling

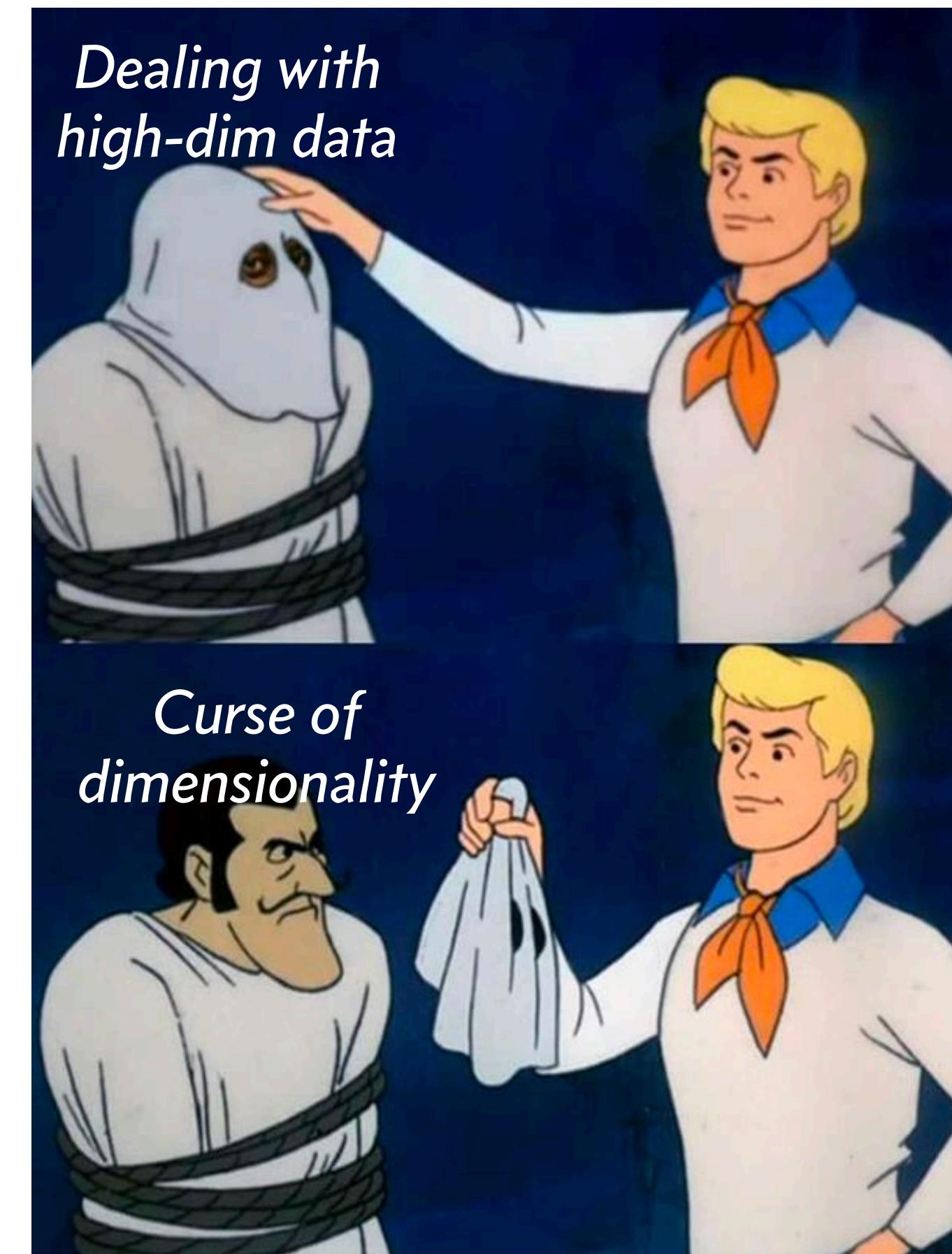
Maximum-likelihood training?

$$\vartheta^* = \arg \max_{\vartheta} p_{\vartheta}(x)$$

$$= \arg \max_{\vartheta} \int p_{\vartheta}(x | z)p(z) dz$$

$$= \arg \max_{\vartheta} \langle p_{\vartheta}(x | z) \rangle_{p(z)}$$

Difficult to build a good estimator!



Latent-variable modeling

Maximum-likelihood training?

$$\begin{aligned}\vartheta^* &= \arg \max_{\vartheta} p_{\vartheta}(x) \\ &= \arg \max_{\vartheta} \int p_{\vartheta}(x \mid z)p(z) dz \\ &= \arg \max_{\vartheta} \left\langle p_{\vartheta}(x \mid z) \right\rangle_{p(z)}\end{aligned}$$

The intractability of $p(x)$ is closely related to the intractability of the *posterior* $p(z \mid x)$

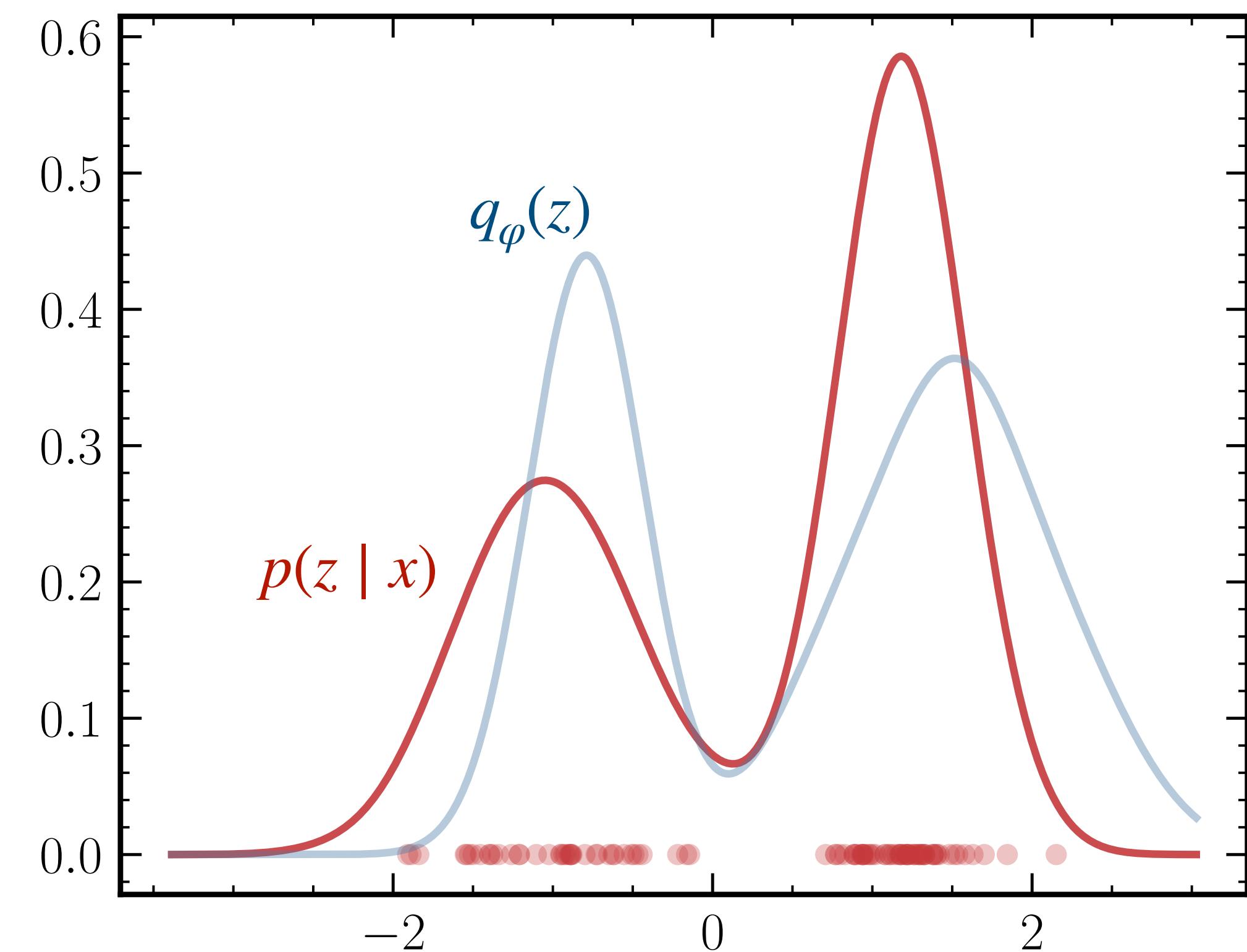
$$p(z \mid x) = \frac{p(x, z)}{p(x)}$$
 (Bayes' theorem)

Difficult to build a good estimator!

Kullback–Leibler (KL) divergence

A measure of similarity between two probability distributions

$$\begin{aligned} D_{\text{KL}}(P\|Q) &= \int_{-\infty}^{\infty} dx p(x) \log \left(\frac{p(x)}{q(x)} \right) \\ &= \left\langle \log \frac{p(x)}{q(x)} \right\rangle_{x \sim p(x)} \end{aligned}$$

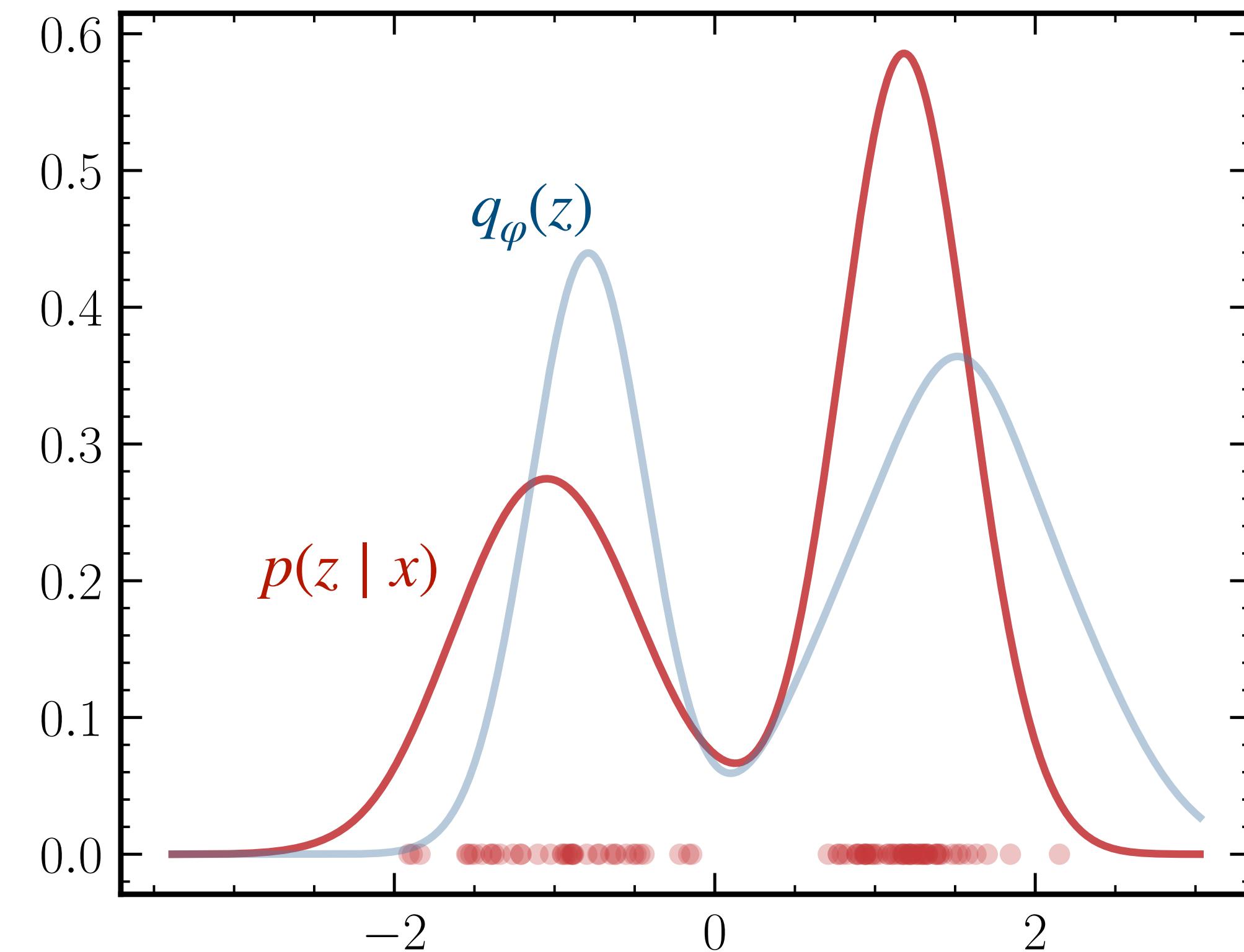


Kullback–Leibler (KL) divergence

A measure of similarity between two probability distributions

$$\begin{aligned} D_{\text{KL}}(P\|Q) &= \int_{-\infty}^{\infty} dx p(x) \log \left(\frac{p(x)}{q(x)} \right) \\ &= \left\langle \log \frac{p(x)}{q(x)} \right\rangle_{x \sim p(x)} \\ &= - \left\langle \log q(x) \right\rangle_{p(x)} + \left\langle \log p(x) \right\rangle_{p(x)} \end{aligned}$$

Cross-entropy $\mathbb{H}(P, Q)$ – (Self-)entropy $\mathbb{H}(P)$



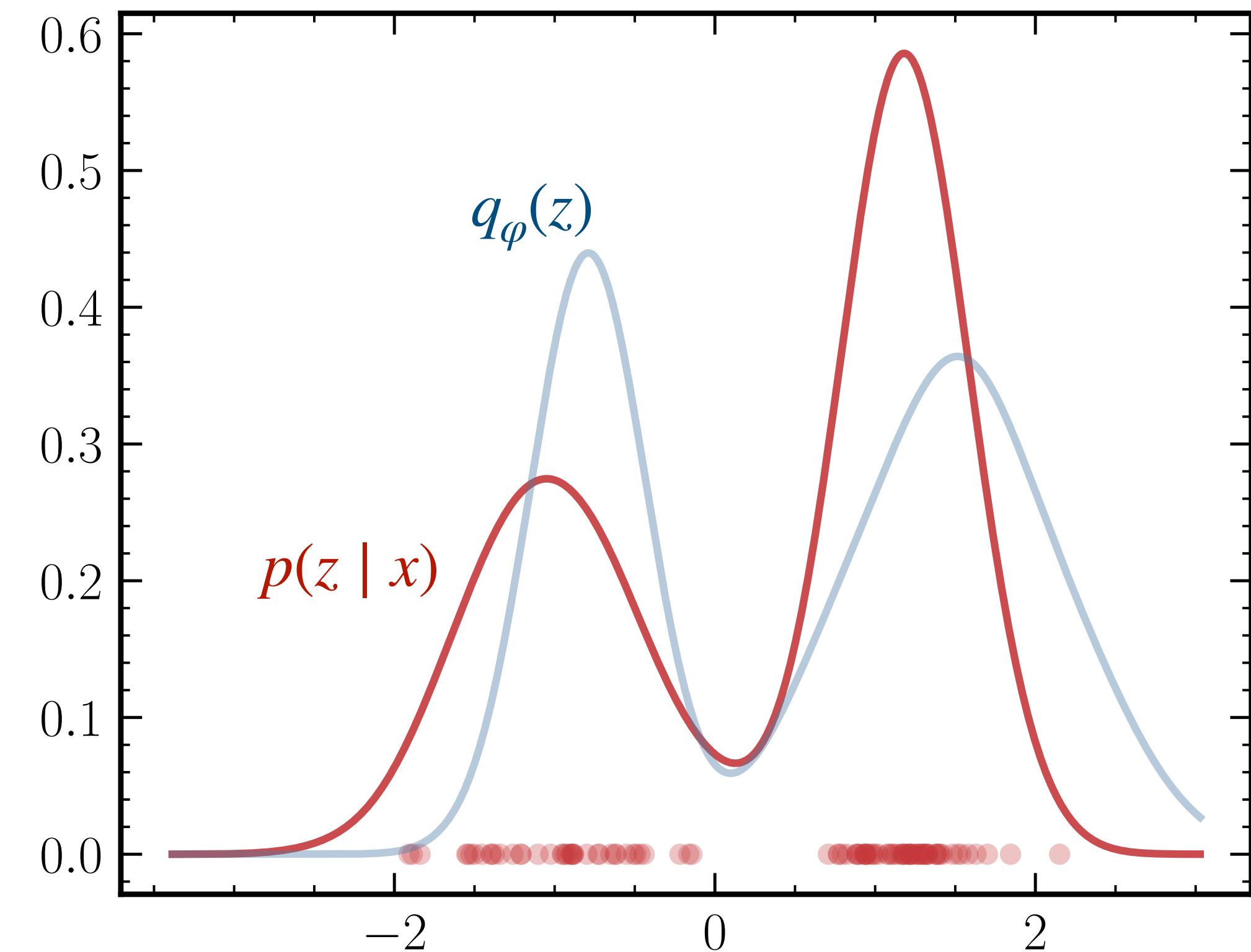
Kullback–Leibler (KL) divergence

A measure of similarity between two probability distributions

$$\begin{aligned} D_{\text{KL}}(P\|Q) &= \int_{-\infty}^{\infty} dx p(x) \log \left(\frac{p(x)}{q(x)} \right) \\ &= \left\langle \log \frac{p(x)}{q(x)} \right\rangle_{x \sim p(x)} \\ &= - \left\langle \log q(x) \right\rangle_{p(x)} + \left\langle \log p(x) \right\rangle_{p(x)} \end{aligned}$$

Cross-entropy $\mathbb{H}(P, Q)$ – (Self-)entropy $\mathbb{H}(P)$

Formally: expected excess “surprise” from
using Q as a model when the actual distribution is P



KL-divergence

A measure of similarity between two probability distributions

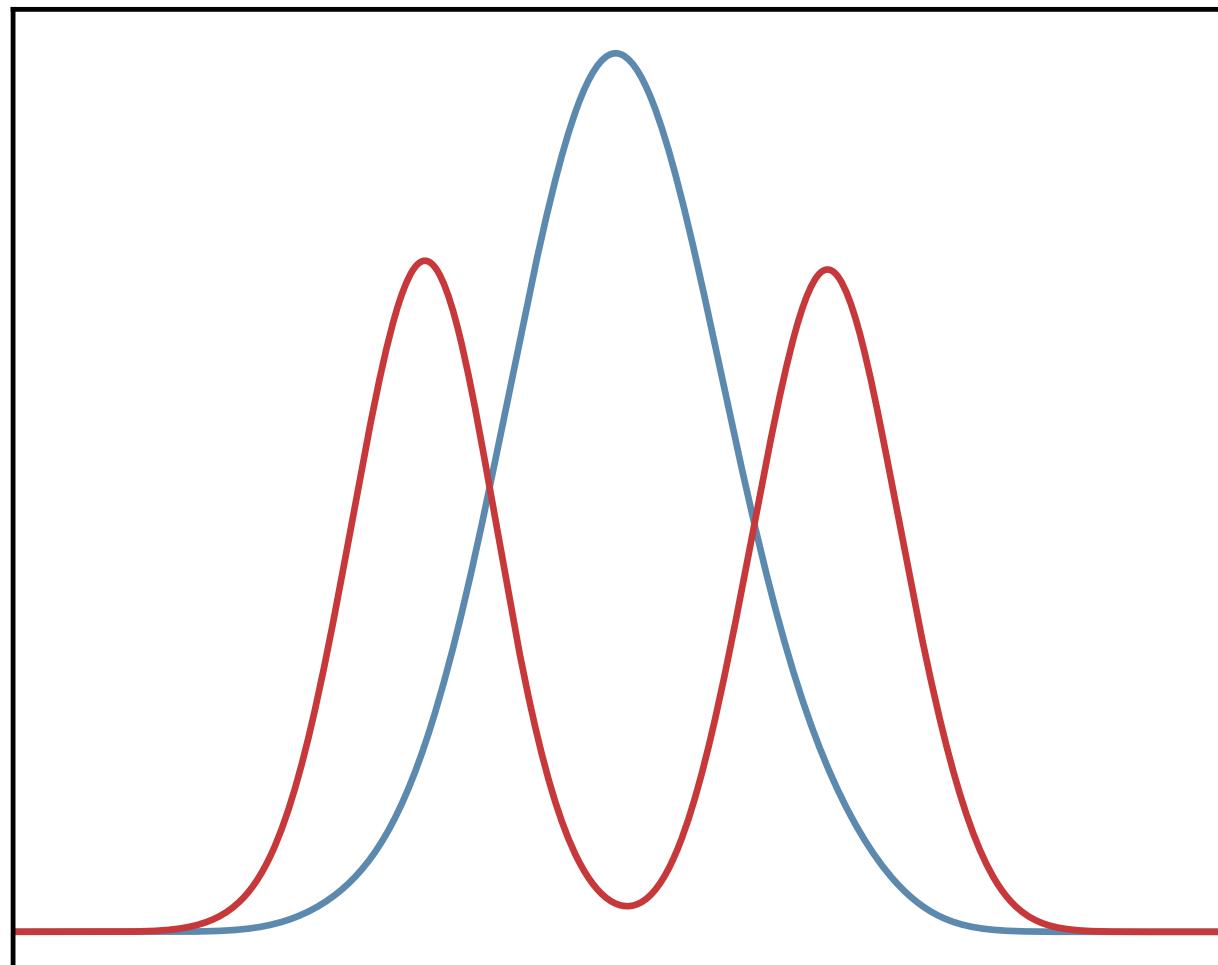
$$D_{\text{KL}}(Q||P) = \int_{-\infty}^{\infty} dx q(x) \log \left(\frac{q(x)}{p(x)} \right)$$

Not symmetric! $D_{\text{KL}}(Q||P) \neq D_{\text{KL}}(P||Q)$

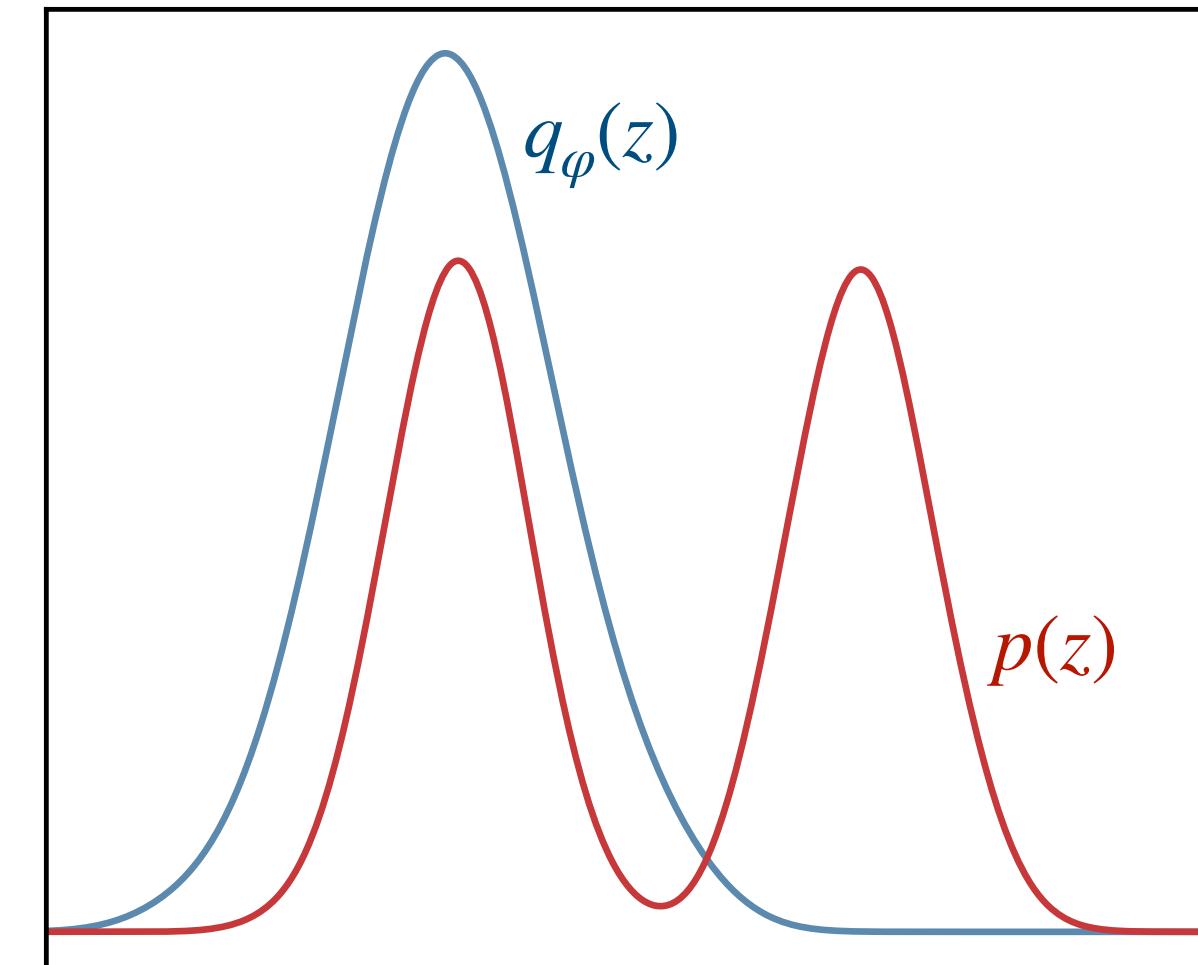
“True” distribution



“Forward” KL $D_{\text{KL}}(P||Q)$



“Reverse” KL $D_{\text{KL}}(Q||P)$



KL-divergence

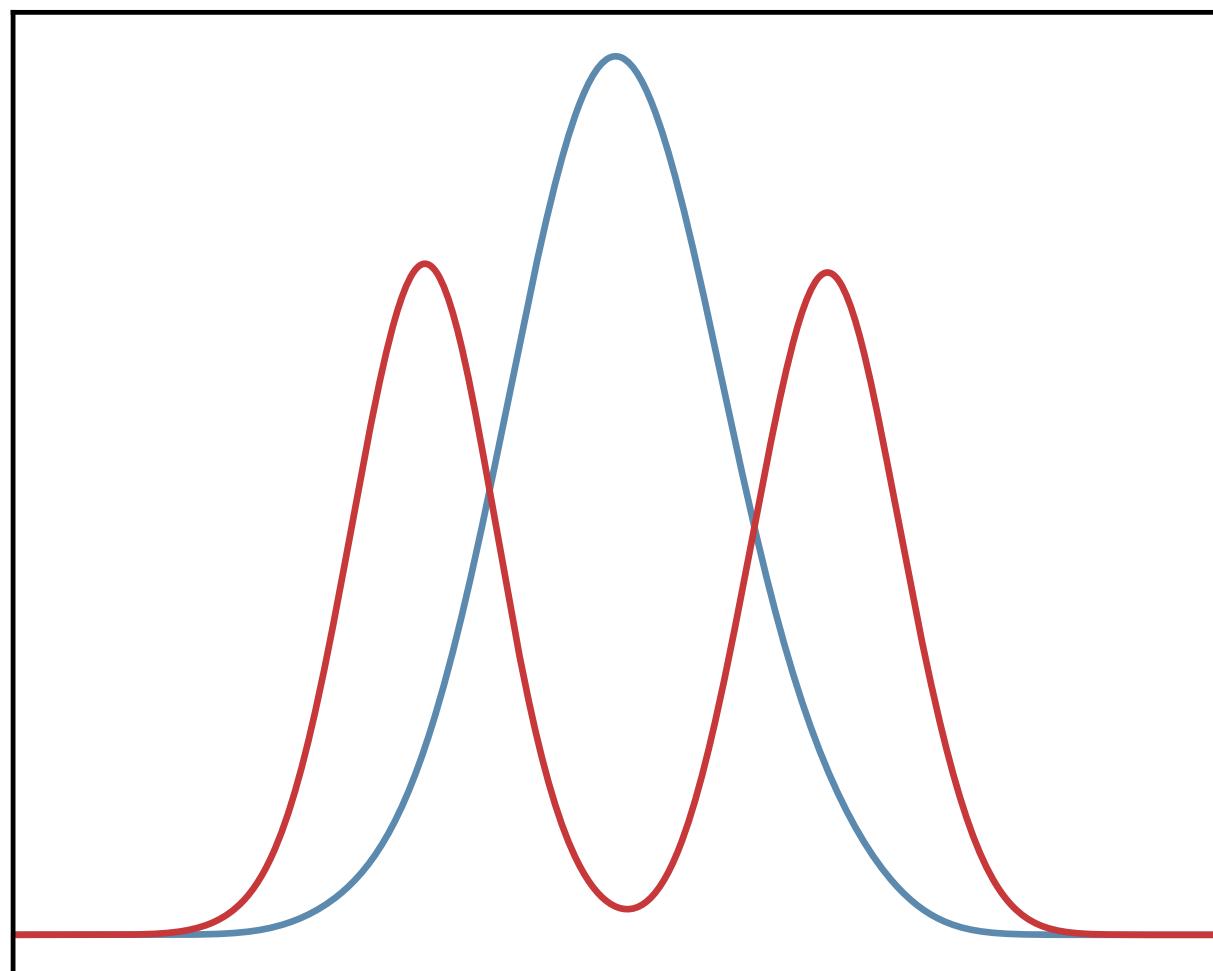
A measure of similarity between two probability distributions

Not symmetric! $D_{\text{KL}}(Q||P) \neq D_{\text{KL}}(P||Q)$

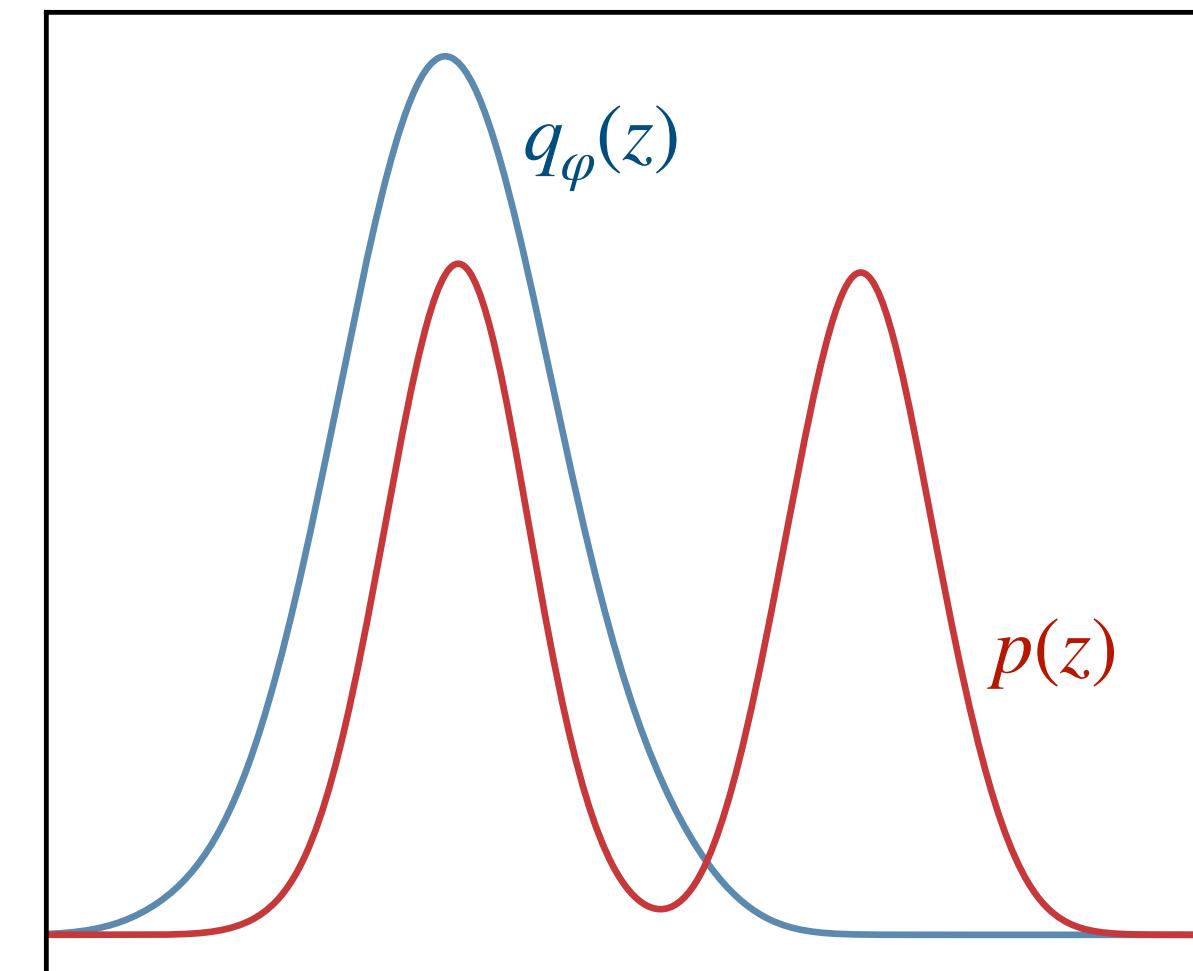
“True” distribution



“Forward” KL $D_{\text{KL}}(P||Q)$



“Reverse” KL $D_{\text{KL}}(Q||P)$



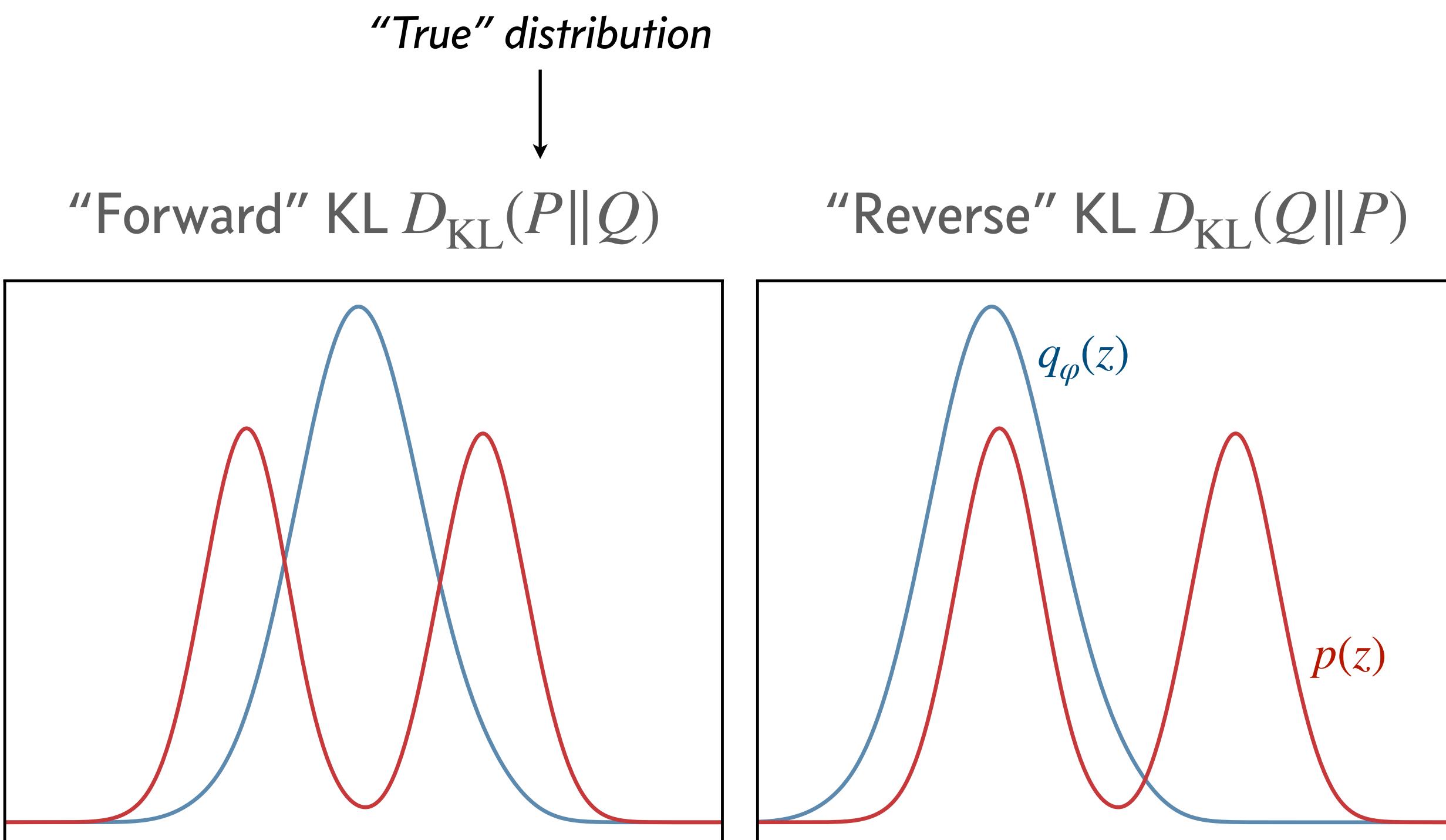
Forward KL

$$D_{\text{KL}}(P_{\mathcal{D}}||Q_{\phi}) = - \left\langle \log q_{\phi}(z) \right\rangle_{z \sim p_{\mathcal{D}}(z)} + \text{const.}$$

KL-divergence

A measure of similarity between two probability distributions

Not symmetric! $D_{\text{KL}}(Q||P) \neq D_{\text{KL}}(P||Q)$



$$D_{\text{KL}}(Q||P) = \int_{-\infty}^{\infty} dx q(x) \log \left(\frac{q(x)}{p(x)} \right)$$

Forward KL

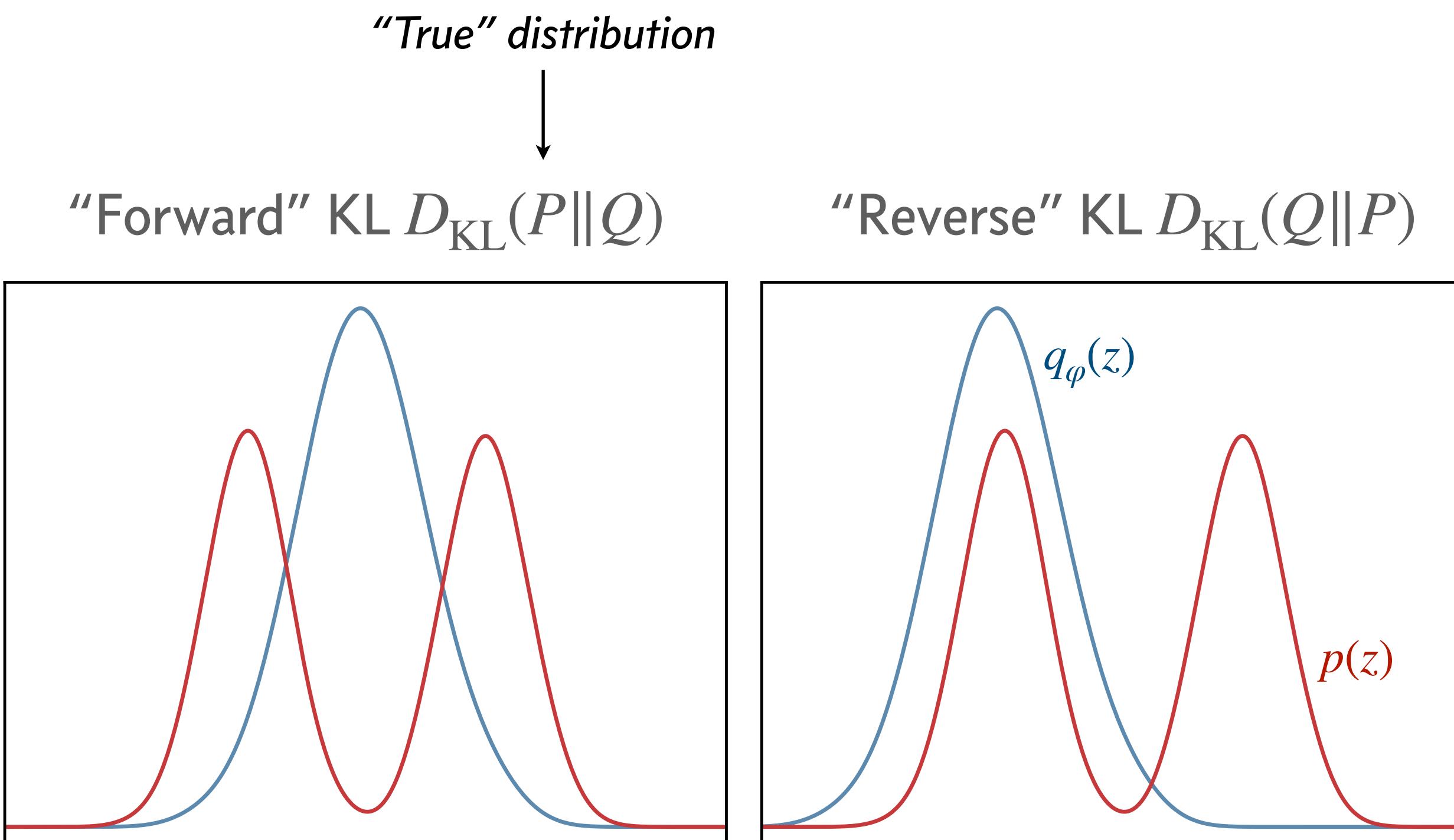
$$D_{\text{KL}}(P_{\mathcal{D}}||Q_{\phi}) = - \left\langle \log q_{\phi}(z) \right\rangle_{z \sim p_{\mathcal{D}}(z)} + \text{const.}$$

Maximum-likelihood inference is equivalent
to minimizing the *forward* KL

KL-divergence

A measure of similarity between two probability distributions

Not symmetric! $D_{\text{KL}}(Q||P) \neq D_{\text{KL}}(P||Q)$



$$D_{\text{KL}}(Q||P) = \int_{-\infty}^{\infty} dx q(x) \log \left(\frac{q(x)}{p(x)} \right)$$

Forward KL

$$D_{\text{KL}}(P_{\mathcal{D}}||Q_{\phi}) = - \left\langle \log q_{\phi}(z) \right\rangle_{z \sim p_{\mathcal{D}}(z)} + \text{const.}$$

Maximum-likelihood inference is equivalent
to minimizing the *forward* KL

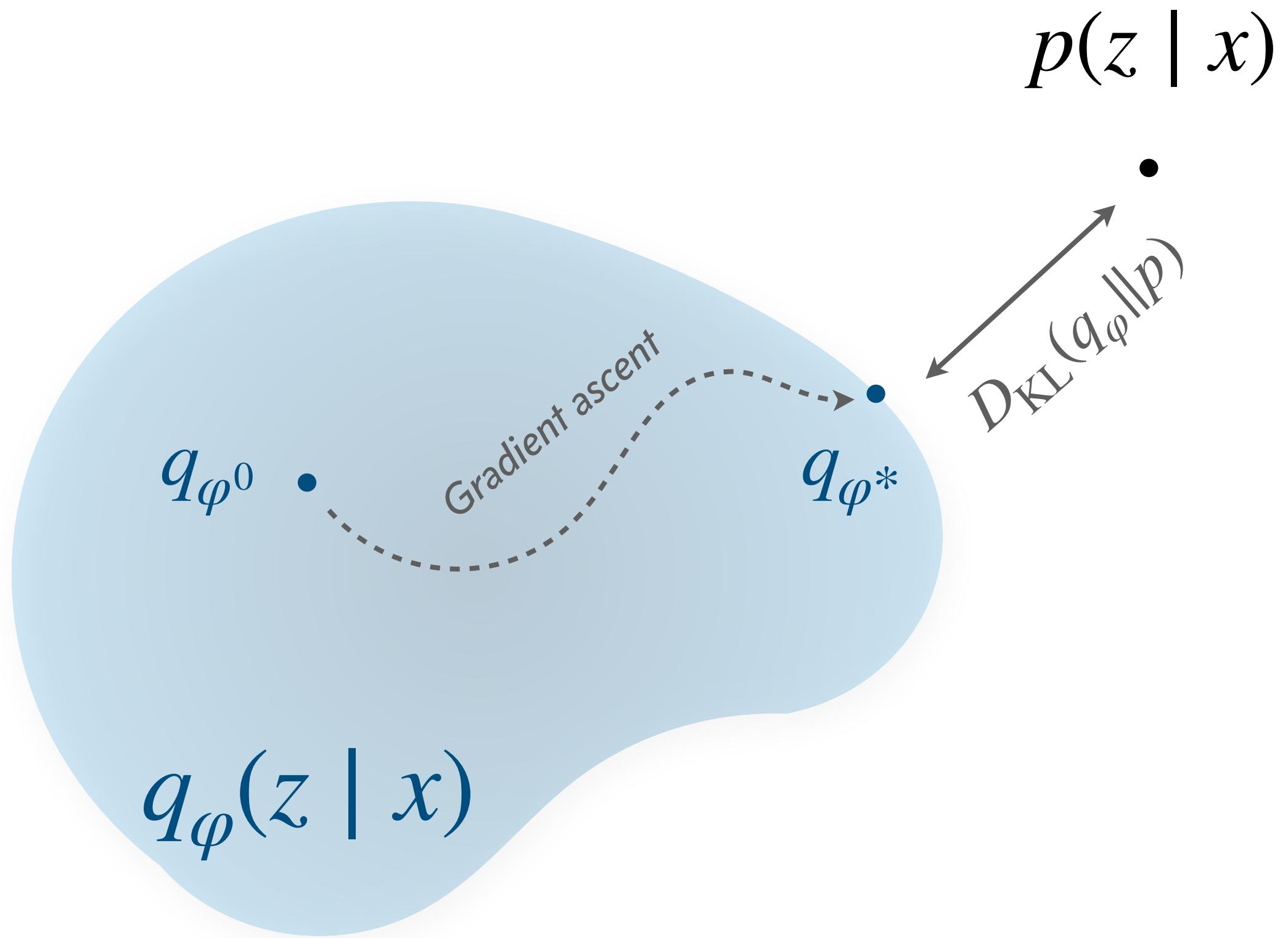
Non-negative! $D_{\text{KL}}(Q||P) \geq 0$

Variational inference

Infer the posterior over the latent parameters

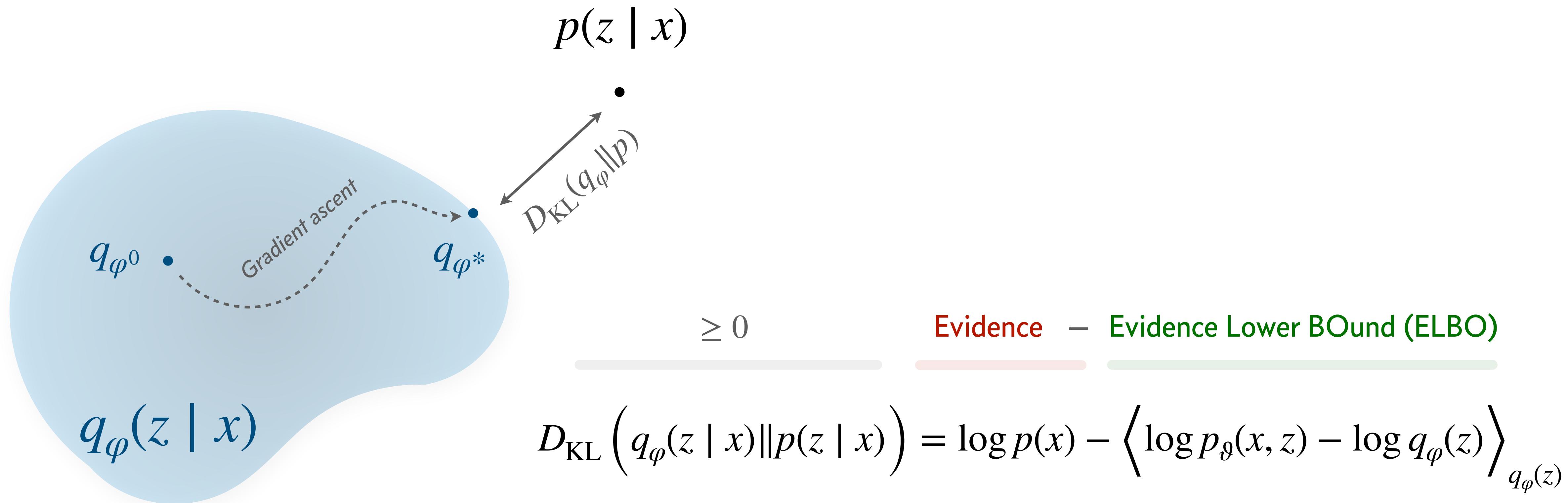
Variational inference

Infer the posterior over the latent parameters



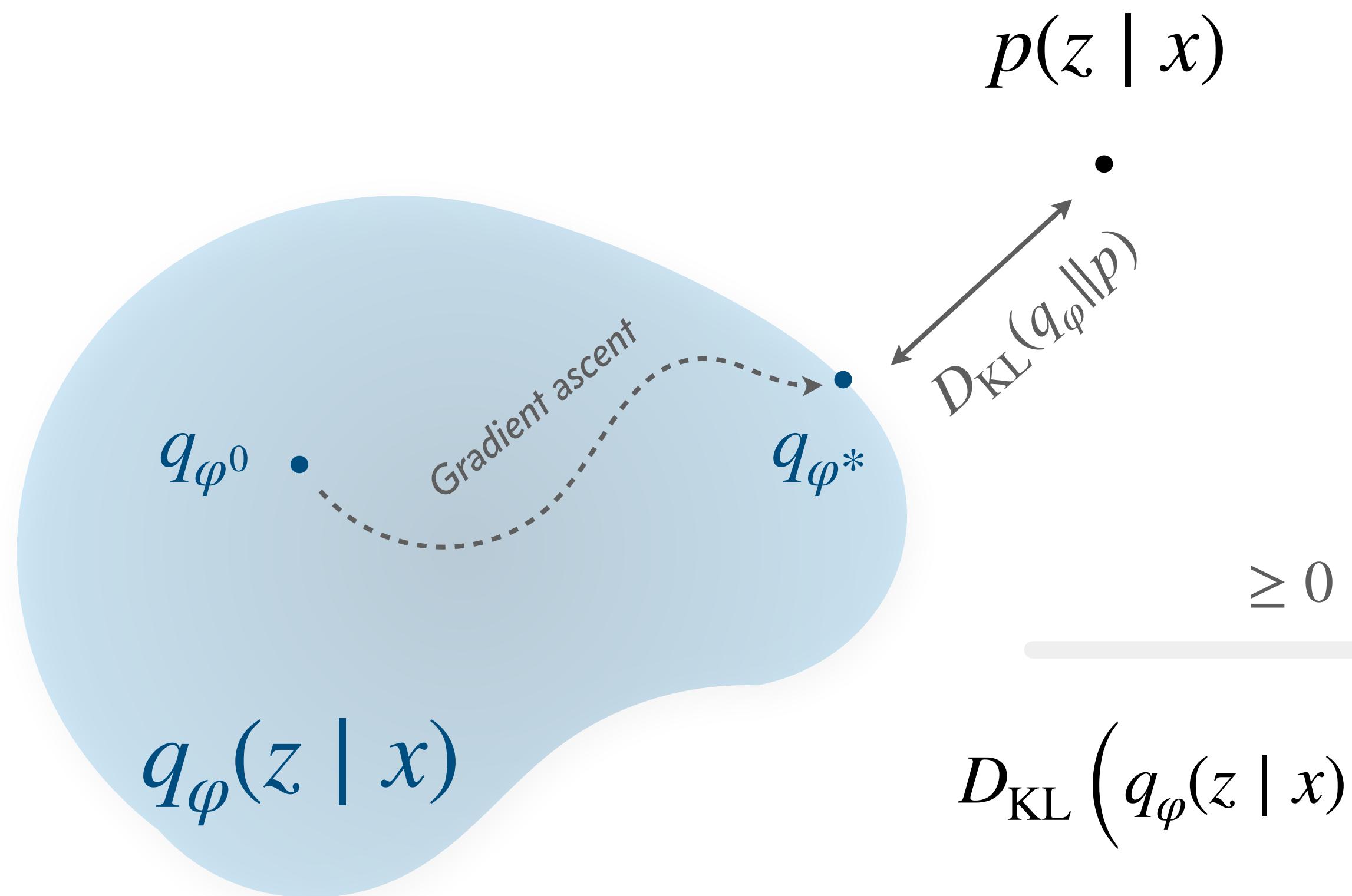
Variational inference

Infer the posterior over the latent parameters



Variational inference

Infer the posterior over the latent parameters



A two-for-one!

- Estimate approximate posterior $q_\phi(z | x) \approx p(z | x)$
- Estimate likelihood/evidence ELBO $\approx p(x)$

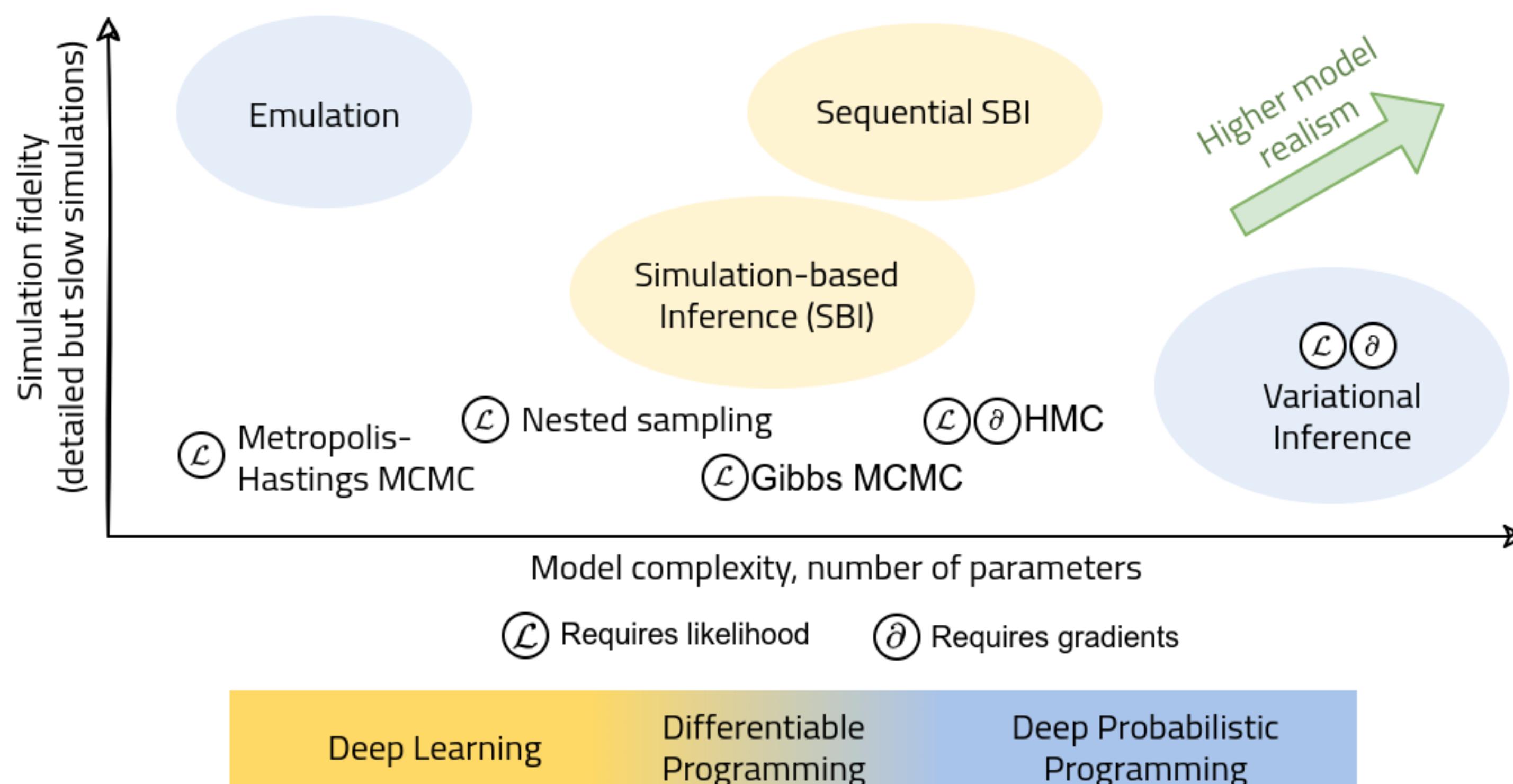
≥ 0

Evidence – Evidence Lower BOund (ELBO)

$$D_{\text{KL}}(q_\phi(z | x) \| p(z | x)) = \log p(x) - \left\langle \log p_\theta(x, z) - \log q_\phi(z) \right\rangle_{q_\phi(z)}$$

Variational inference

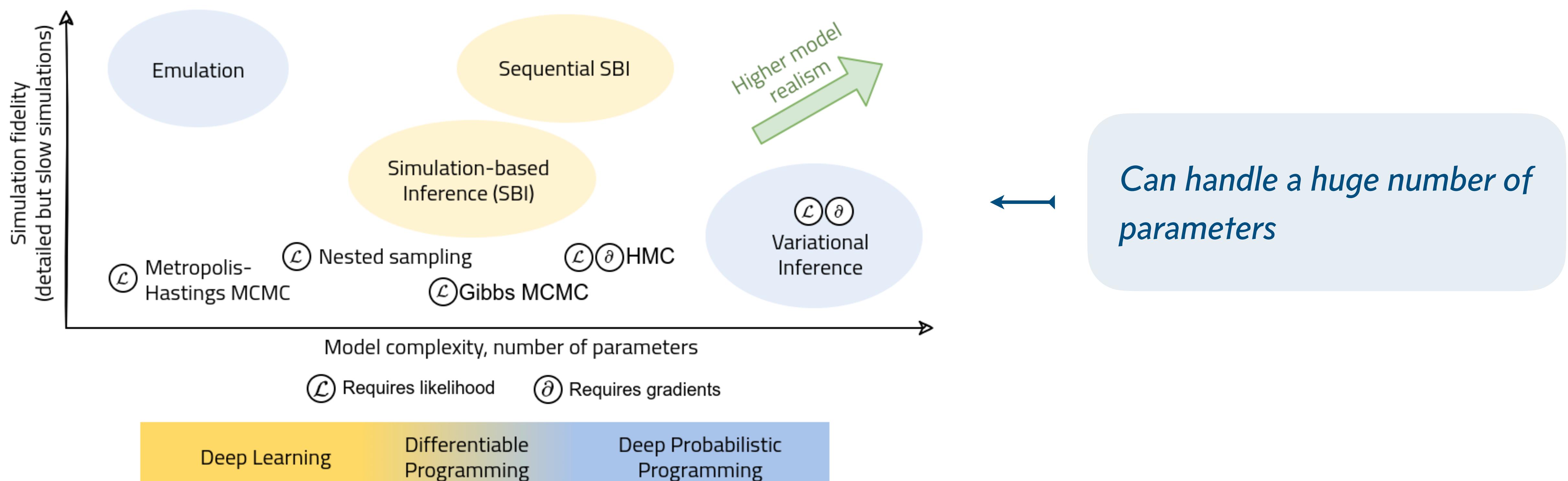
A general-purpose technique for posterior estimation: *optimization instead of sampling*



[EuCAPT White Paper 2021]

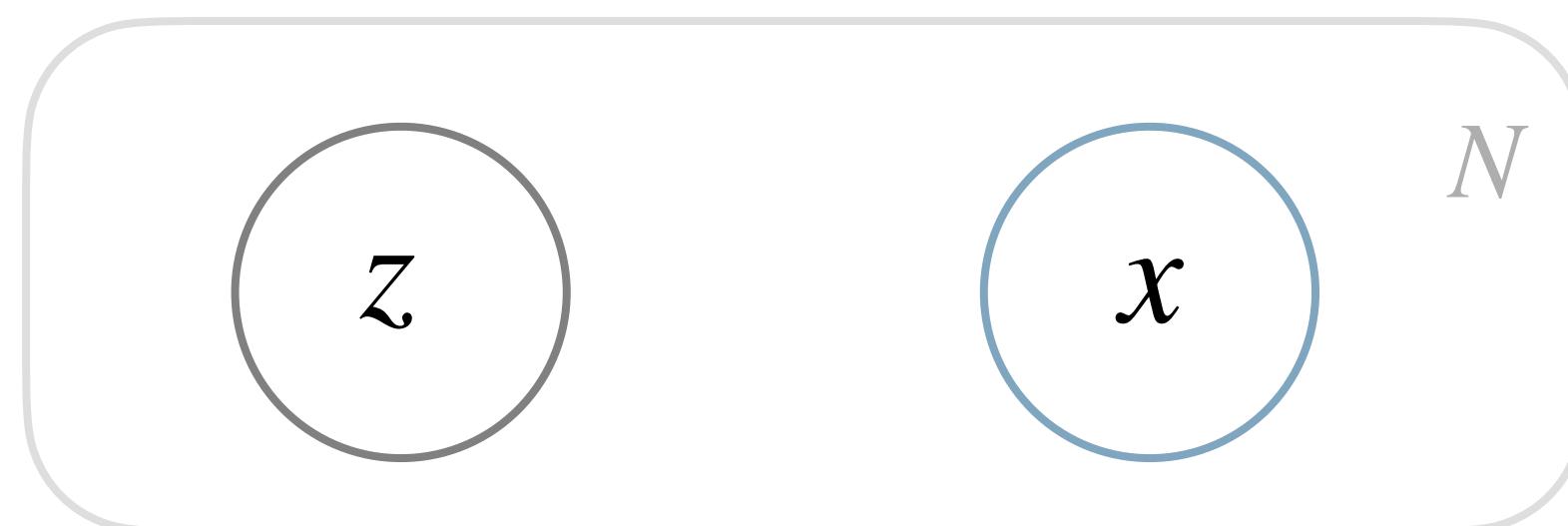
Variational inference

A general-purpose technique for posterior estimation: *optimization instead of sampling*

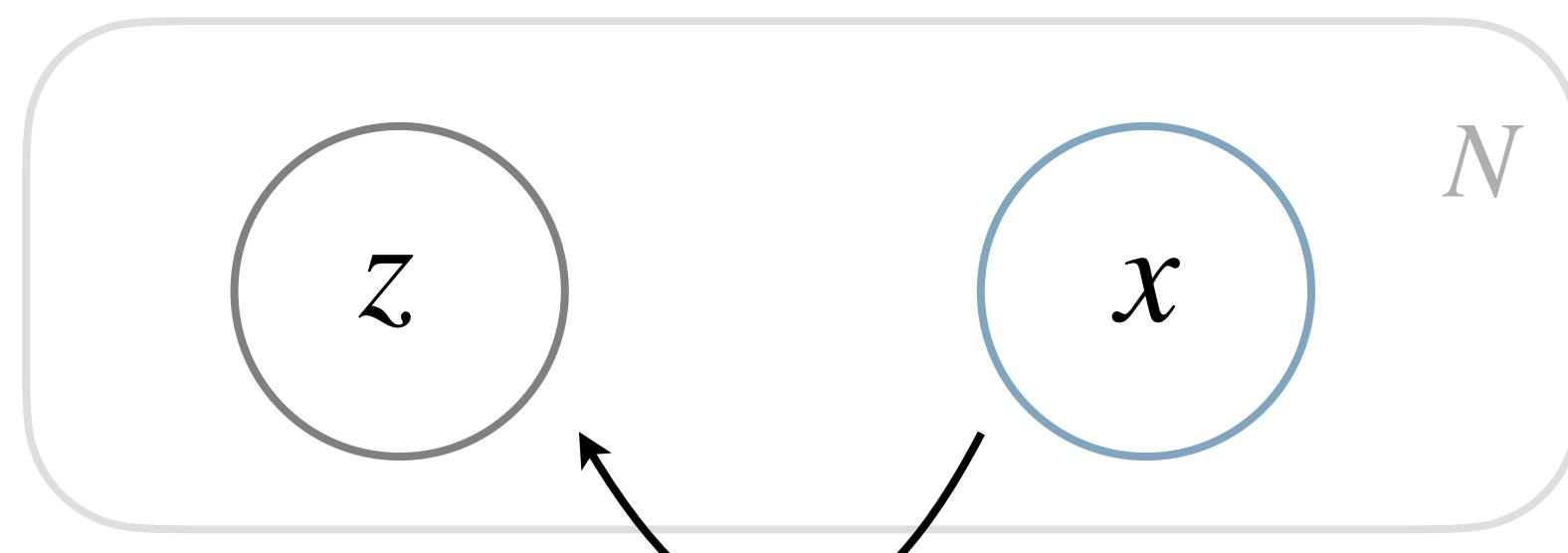


[EuCAPT White Paper 2021]

A Bayesian latent-variable model optimized with variational inference



A Bayesian latent-variable model optimized with variational inference



$$q_\varphi(z | x) \cdot p(x)$$

Forward process

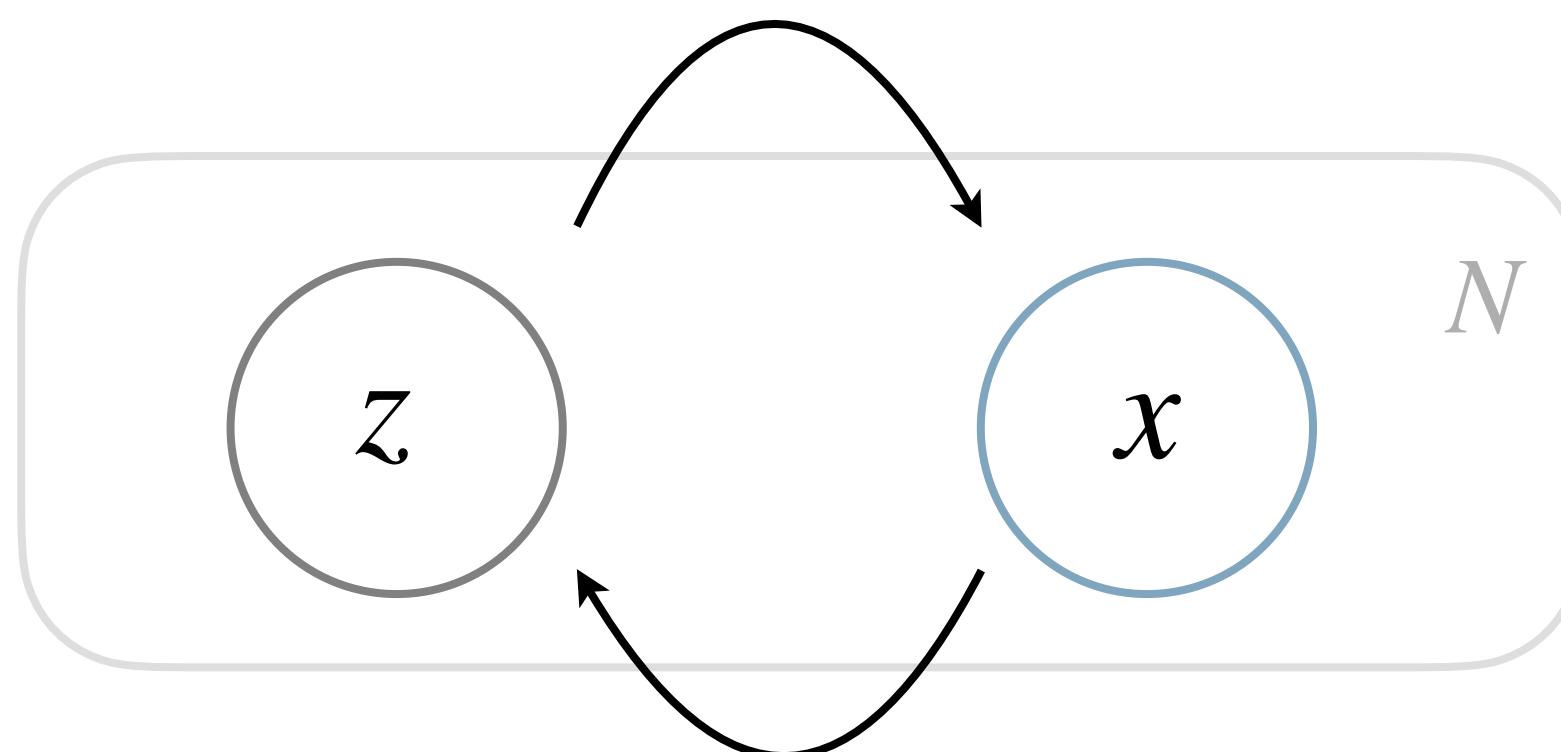
It's so over

A Bayesian latent-variable model optimized with variational inference

We're so back

Reverse process

$$p_{\vartheta}(x | z) \cdot p(z)$$



$$q_{\varphi}(z | x) \cdot p(x)$$

Forward process

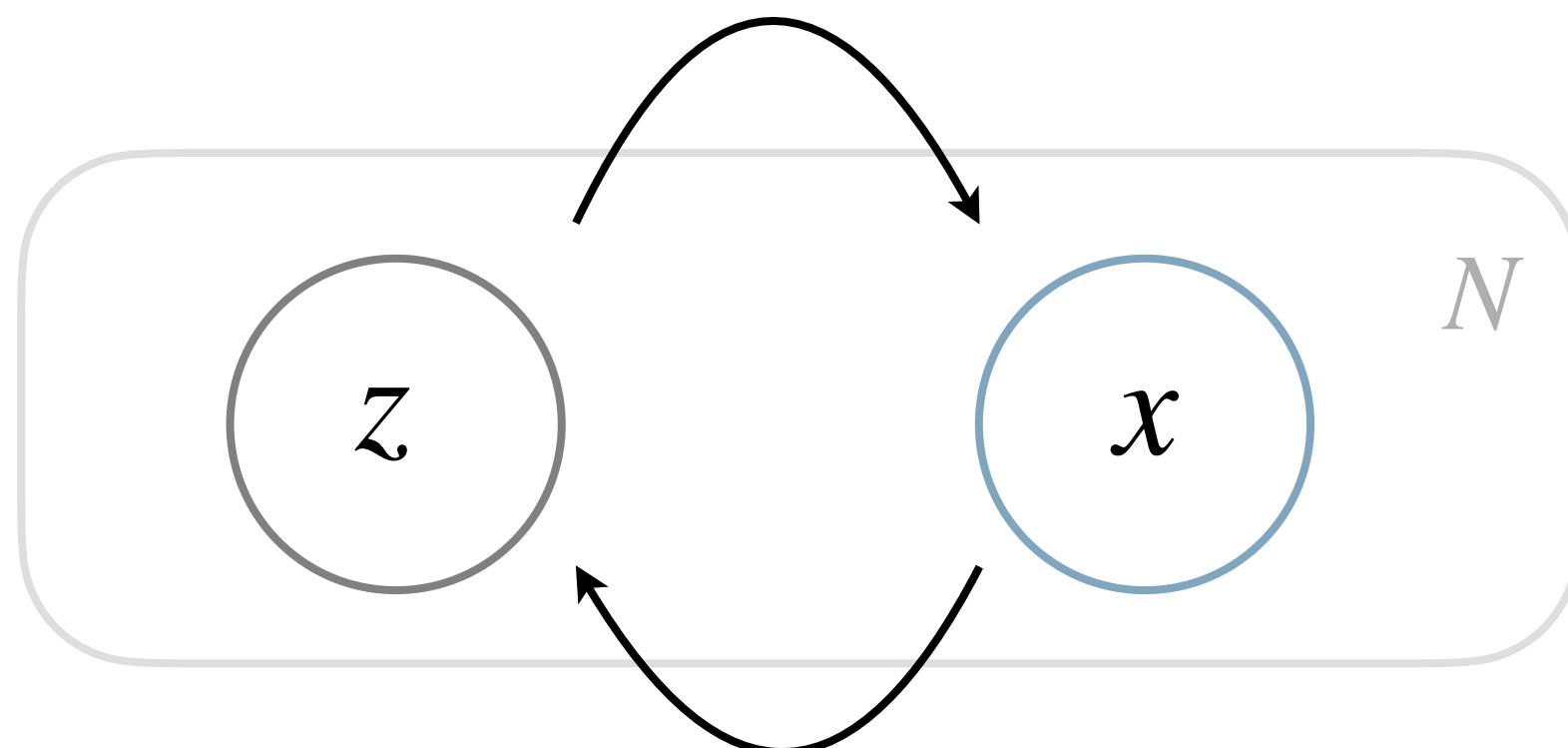
It's so over

A Bayesian latent-variable model optimized with variational inference

We're so back

Reverse process

$$p_{\vartheta}(x | z) \cdot p(z)$$



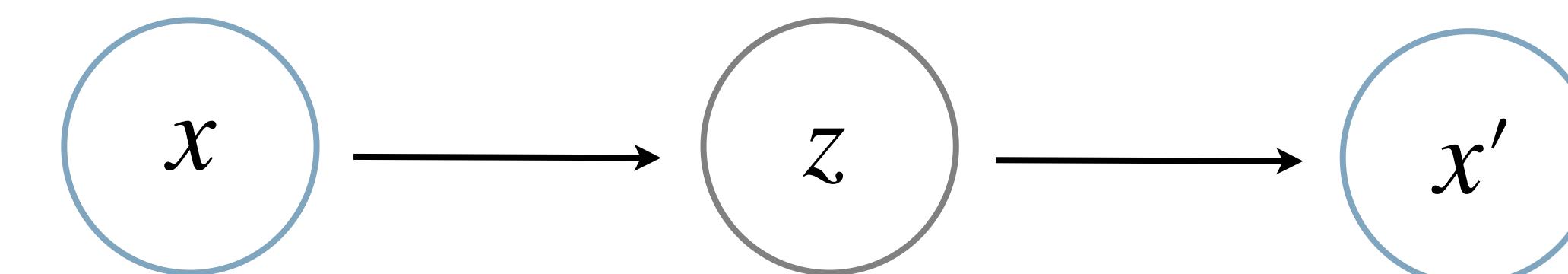
$$q_{\varphi}(z | x) \cdot p(x)$$

Forward process

It's so over

Forward process

Reverse process



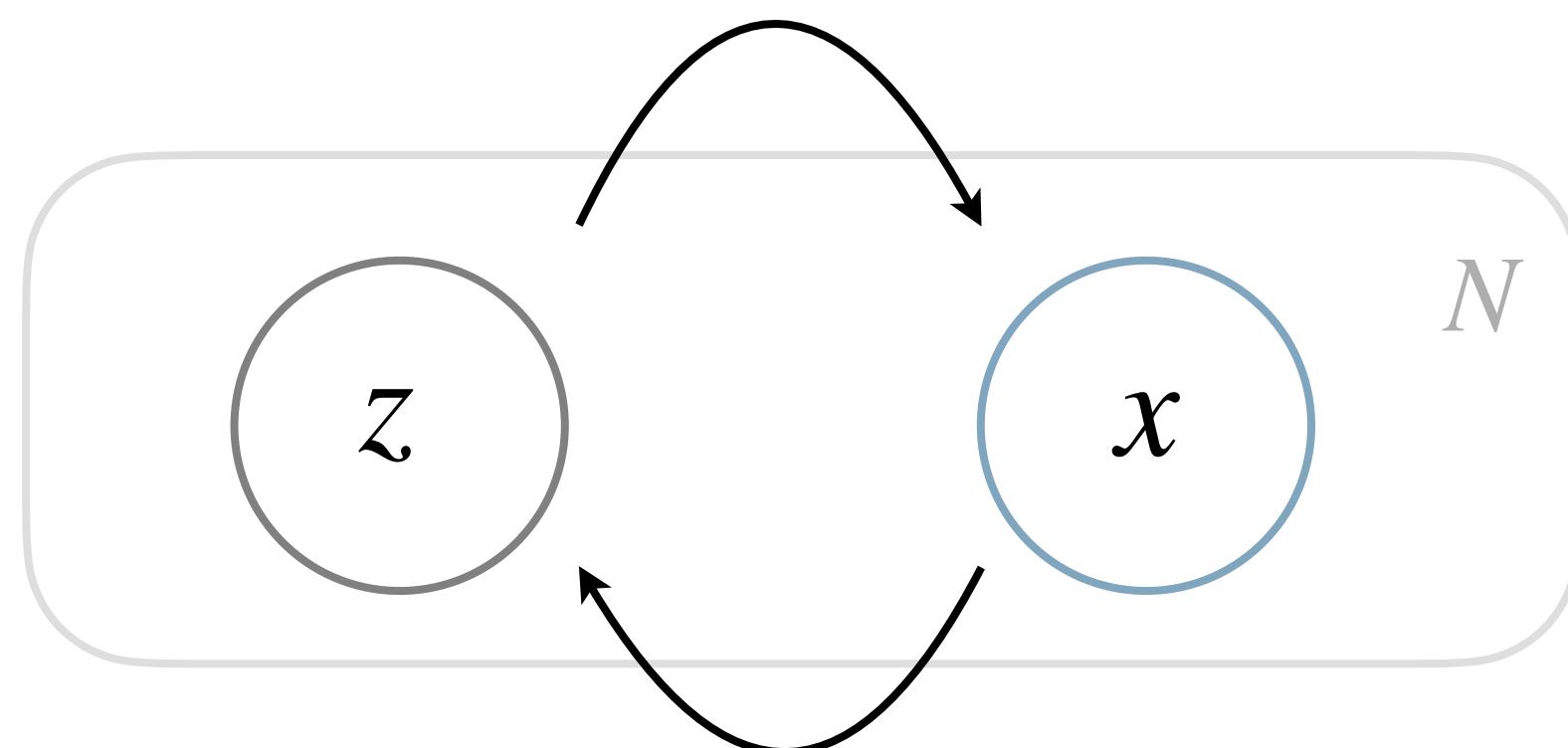
Latents

A Bayesian latent-variable model optimized with variational inference

We're so back

Reverse process

$$p_{\vartheta}(x | z) \cdot p(z)$$



$$q_{\varphi}(z | x) \cdot p(x)$$

Forward process

It's so over

Maximizing ELBO

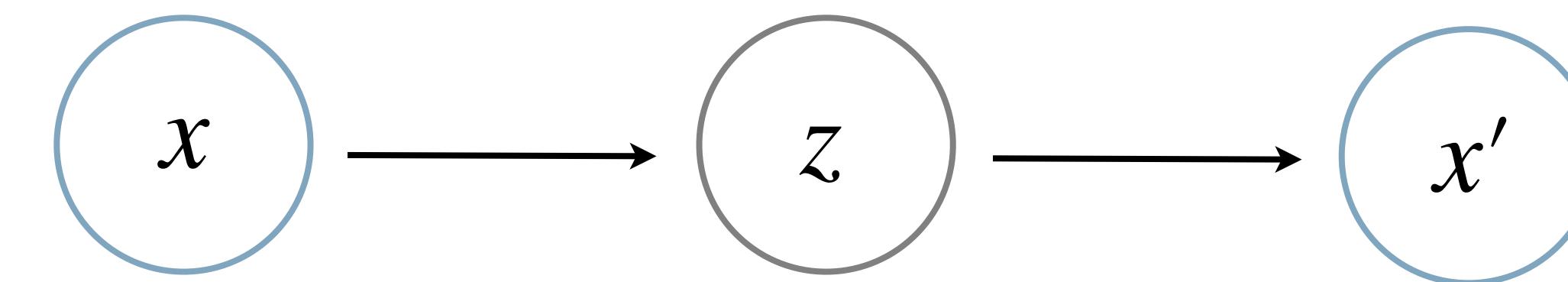
≡ Minimizing reverse KL

≡ "Aligning the forward and reverse processes"

$$\text{Minimize} \left\langle \log \frac{q(x, z)}{p(x, z)} \right\rangle$$

Forward process

Reverse process



Latents



Christopher Yau
@cwcyau

...

People do realise that a variational autoencoder comes from the application of variational inference to a Bayesian latent variable model right? It isn't an arbitrary loss function with a KL term stuck on to it with a tweakable parameter to balance the two?

[https://twitter.com/cwcyau/
status/1440434674556227591](https://twitter.com/cwcyau/status/1440434674556227591)



Christopher Yau
@cwcyyau

...

People do realise that a variational autoencoder comes from the application of variational inference to a Bayesian latent variable model right? It isn't an arbitrary loss function with a KL term stuck on to it with a tweakable parameter to balance the two?



Julian Togelius @togelius · Sep 22, 2021
No. I think of it as an arbitrary loss function and it works well for me. I'm in favor of arbitrary loss functions.

1

1

16



[https://twitter.com/cwcyyau/
status/1440434674556227591](https://twitter.com/cwcyyau/status/1440434674556227591)



Christopher Yau
@cwcyyau

...

People do realise that a variational autoencoder comes from the application of variational inference to a Bayesian latent variable model right? It isn't an arbitrary loss function with a KL term stuck on to it with a tweakable parameter to balance the two?



Julian Togelius @togelius · Sep 22, 2021
No. I think of it as an arbitrary loss function and it works well for me. I'm in favor of arbitrary loss functions.

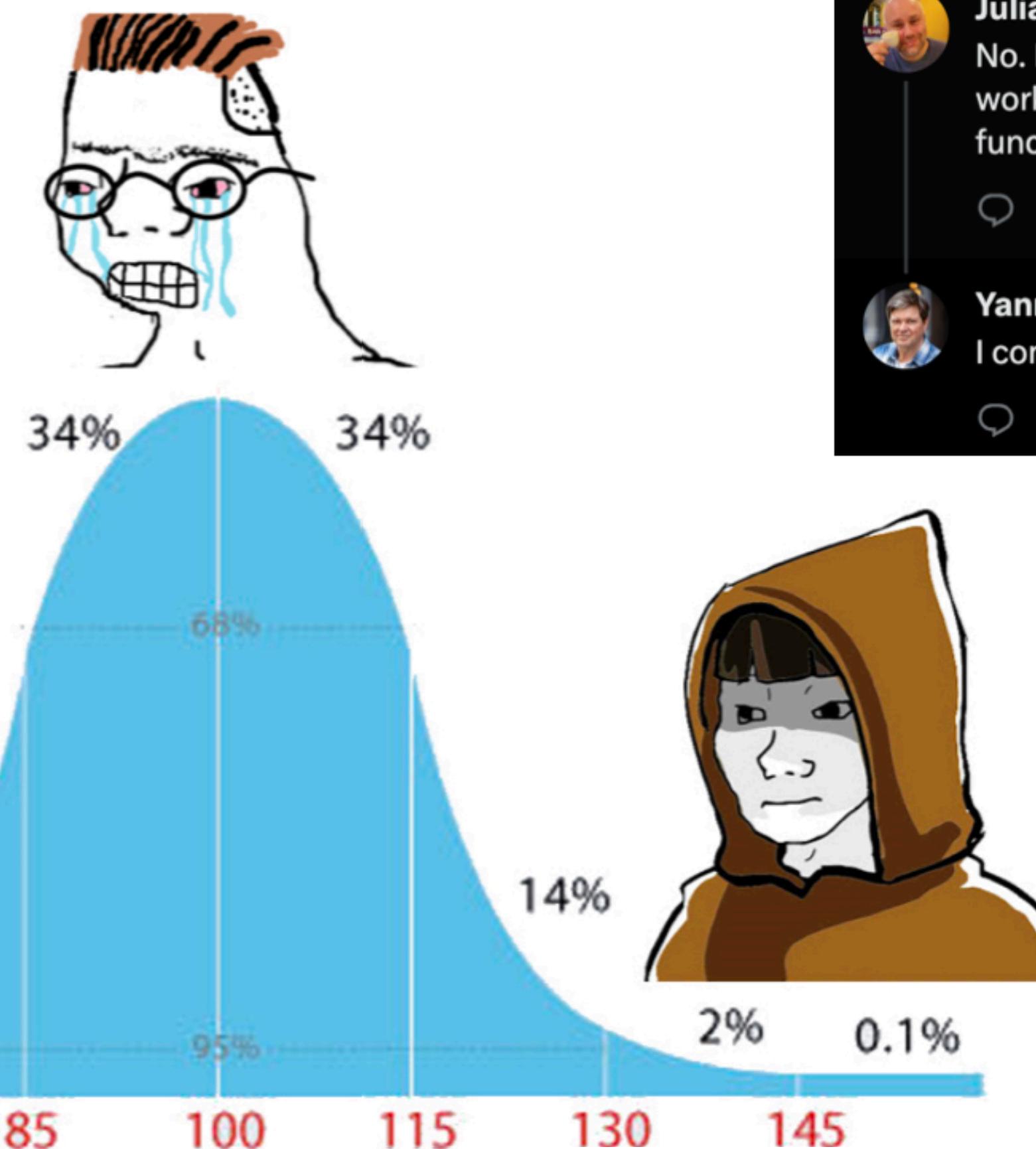
1

1

16

...

...



Julian Togelius @togelius · Sep 22, 2021
No. I think of it as an arbitrary loss function and it works well for me. I'm in favor of arbitrary loss functions.

1

1

16

...



Yann LeCun @ylecun · Sep 22, 2021
I concur.

0

1

6

...

...

...

...

...

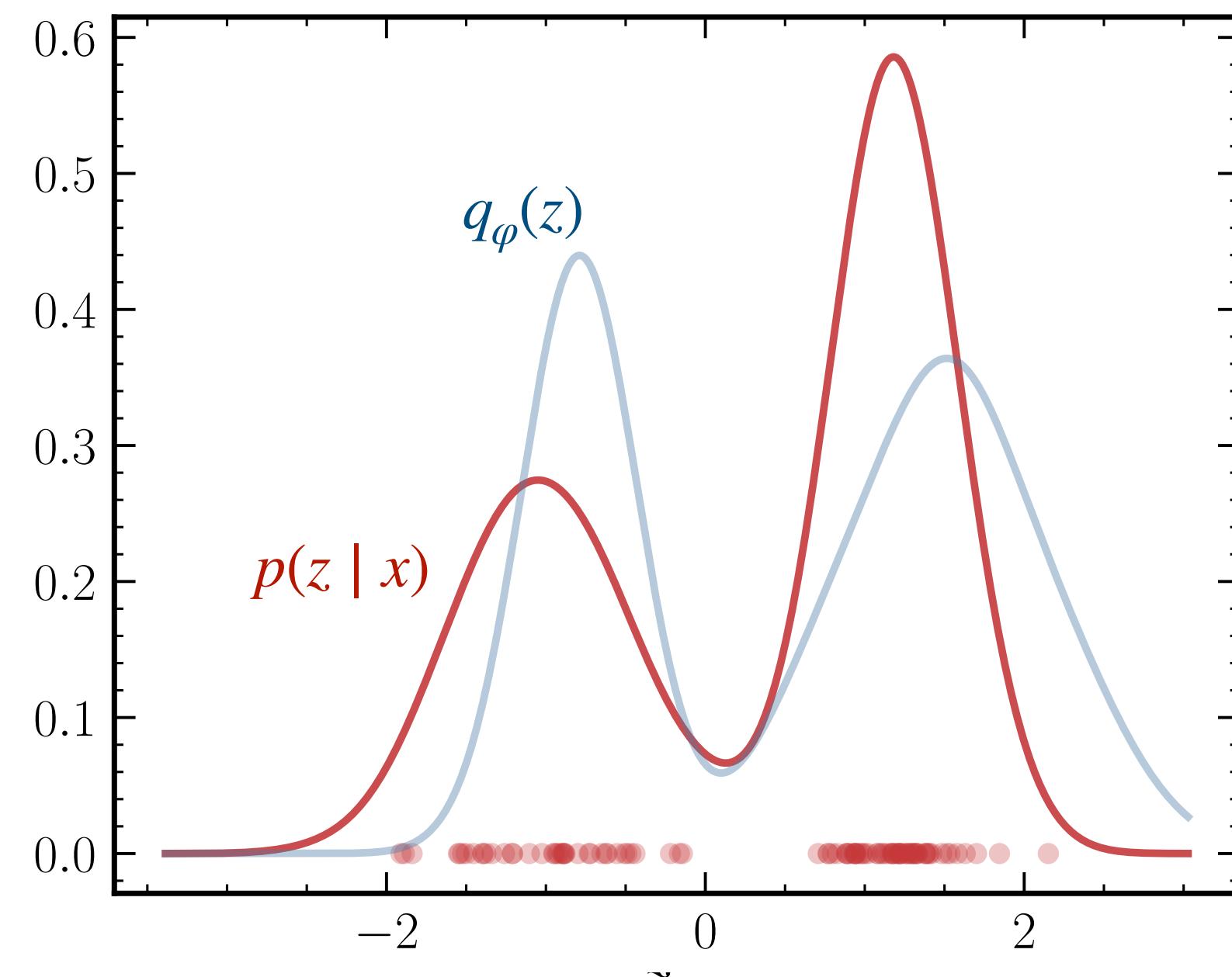
...

...

Variational inference

A general-purpose technique for posterior estimation

$$D_{\text{KL}}(q_{\varphi}(z) \| p(z | x)) = \log p(x) - \left\langle \log p_{\vartheta}(x, z) - \log q_{\varphi}(z) \right\rangle_{q_{\varphi}(z)}$$



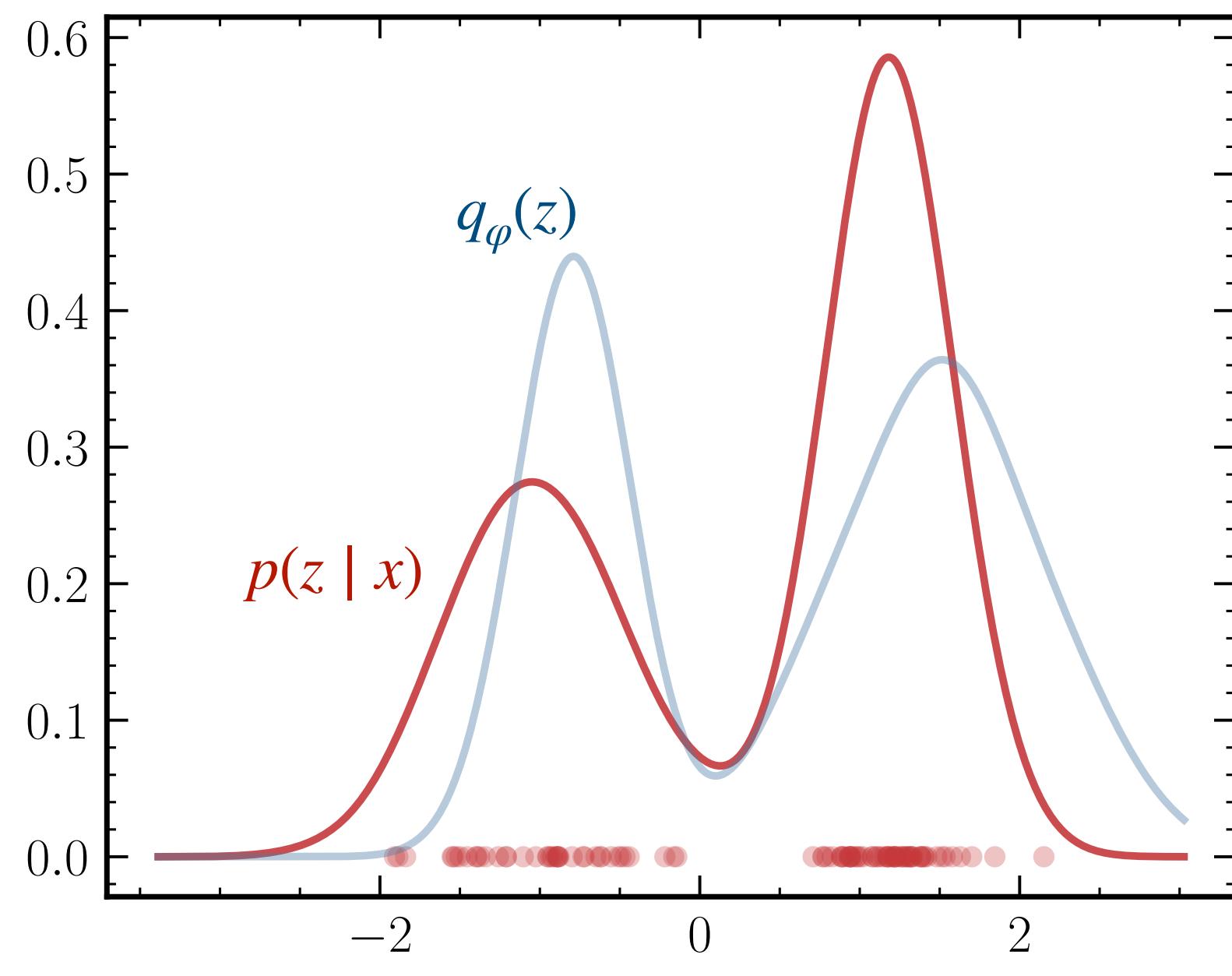
Variational inference

A general-purpose technique for posterior estimation

≥ 0

Evidence – Evidence Lower BOund (ELBO)

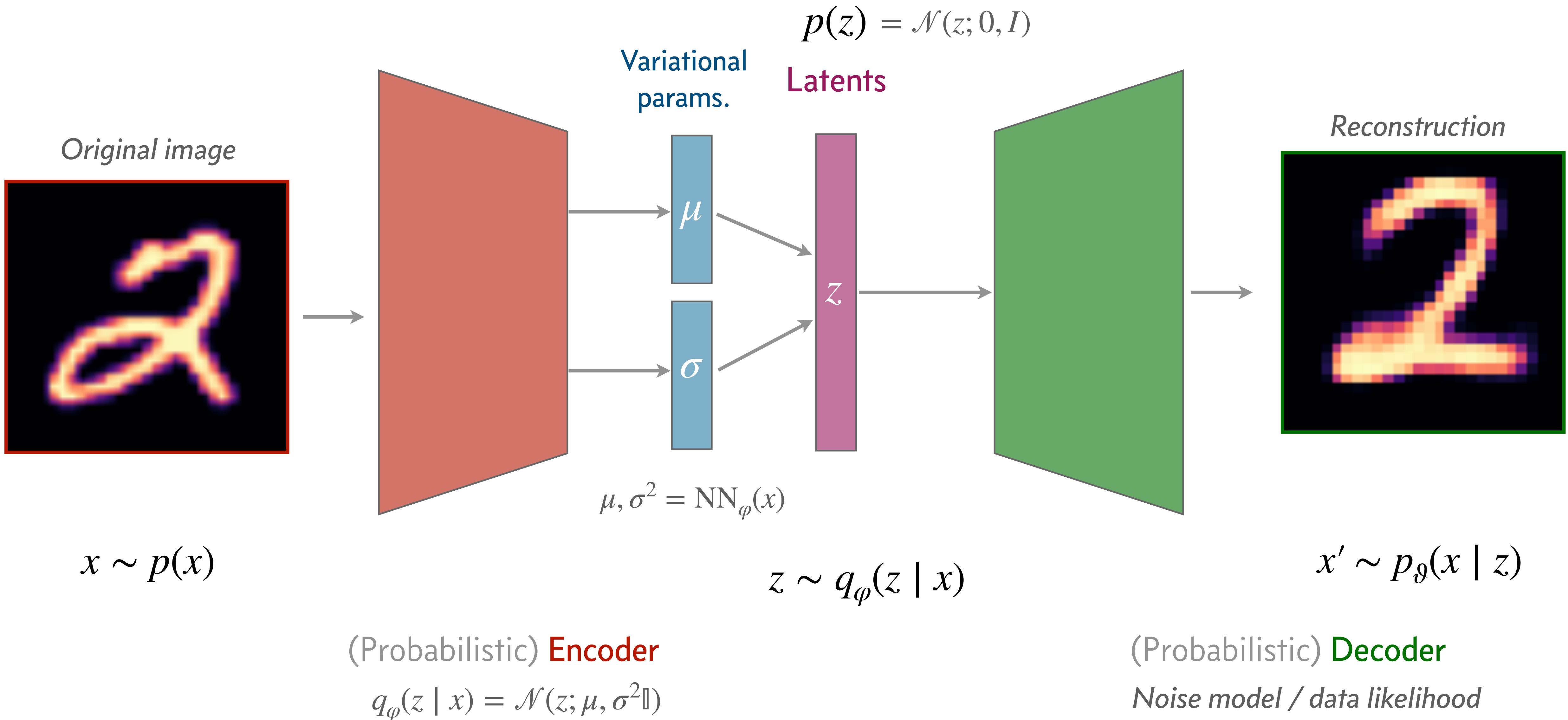
$$D_{\text{KL}}(q_{\varphi}(z) \| p(z | x)) = \log p(x) - \left\langle \log p_{\vartheta}(x, z) - \log q_{\varphi}(z) \right\rangle_{q_{\varphi}(z)}$$



$$\begin{aligned} \text{ELBO} &= \left\langle \log p_{\vartheta}(x, z) - \log q_{\varphi}(z | x) \right\rangle_{q_{\varphi}} \\ &= \left\langle \log p_{\vartheta}(x | z) + \log p(z) - \log q_{\varphi}(z | x) \right\rangle_{q_{\varphi}} \\ &= \left\langle \log p_{\vartheta}(x | z) \right\rangle_{q_{\varphi}} - D_{\text{KL}}(q_{\varphi}(z | x) \| p(z)) \end{aligned}$$

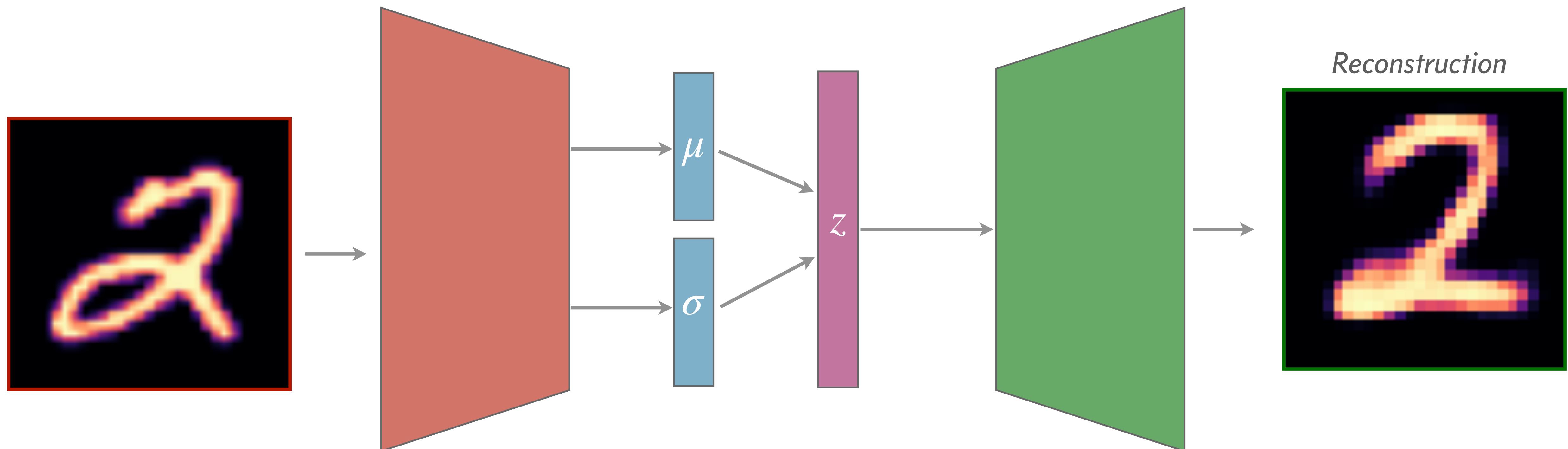
"Reconstruction" "Regularization"

VAEs in practice



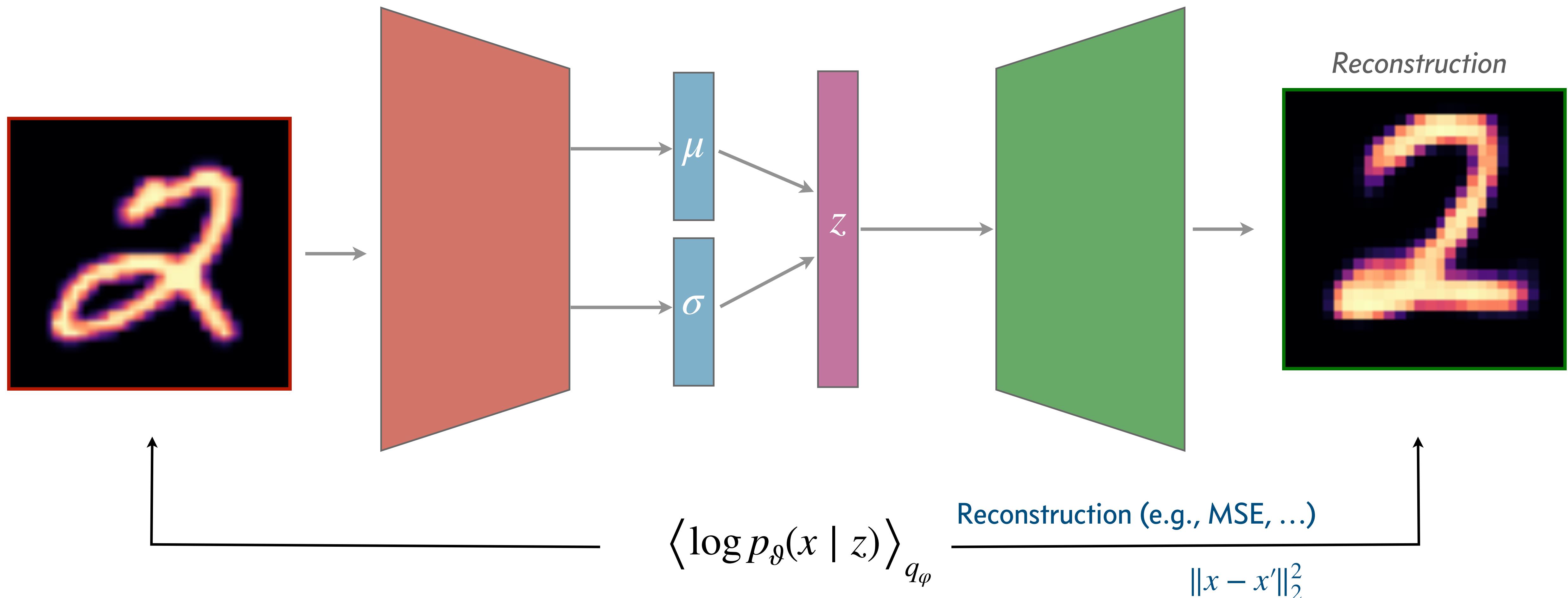
VAEs in practice

$$\text{ELBO} = \left\langle \log p_\vartheta(x | z) \right\rangle_{q_\varphi} - D_{\text{KL}} \left(q_\varphi(z | x) \| p(z) \right)$$

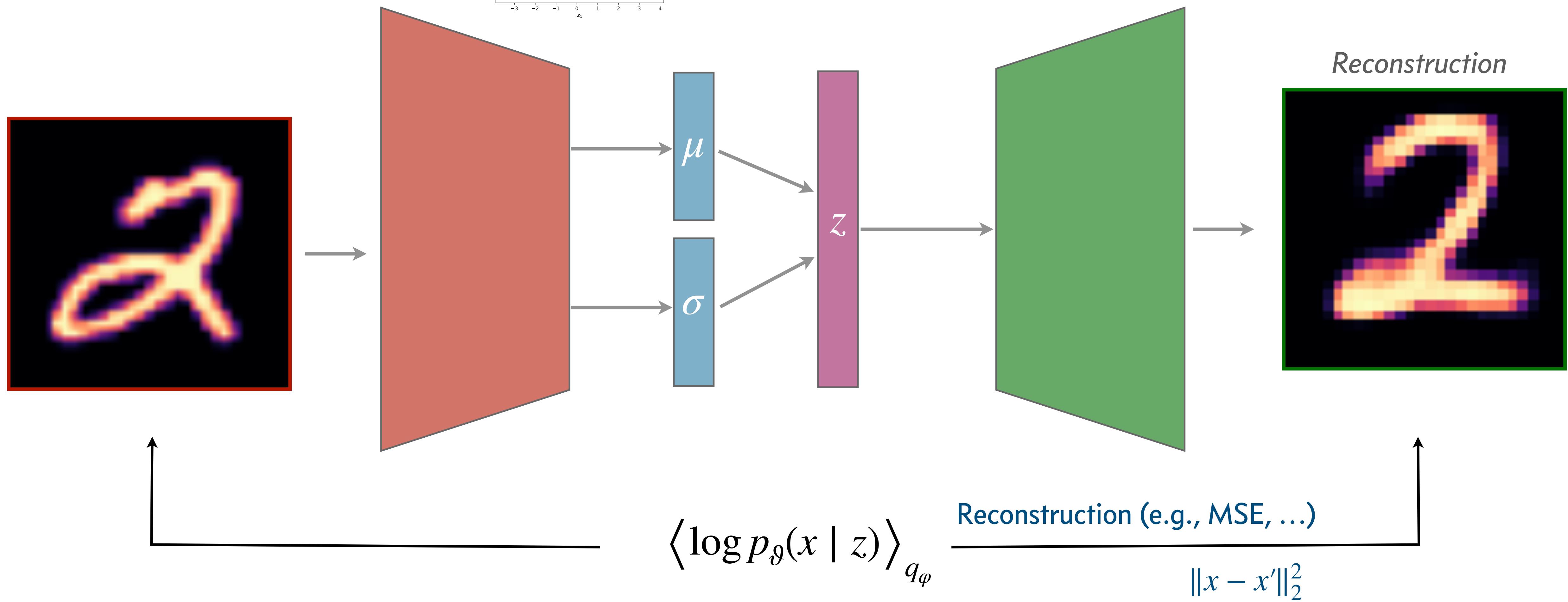


VAEs in practice

$$\text{ELBO} = \left\langle \log p_\vartheta(x | z) \right\rangle_{q_\varphi} - D_{\text{KL}} \left(q_\varphi(z | x) \| p(z) \right)$$



VAEs in practice



A semantically meaningful latent space

The KL-term enforces simplicity in the latent space, encouraging learned semantic structure and *disentanglement*

Pure reconstruction



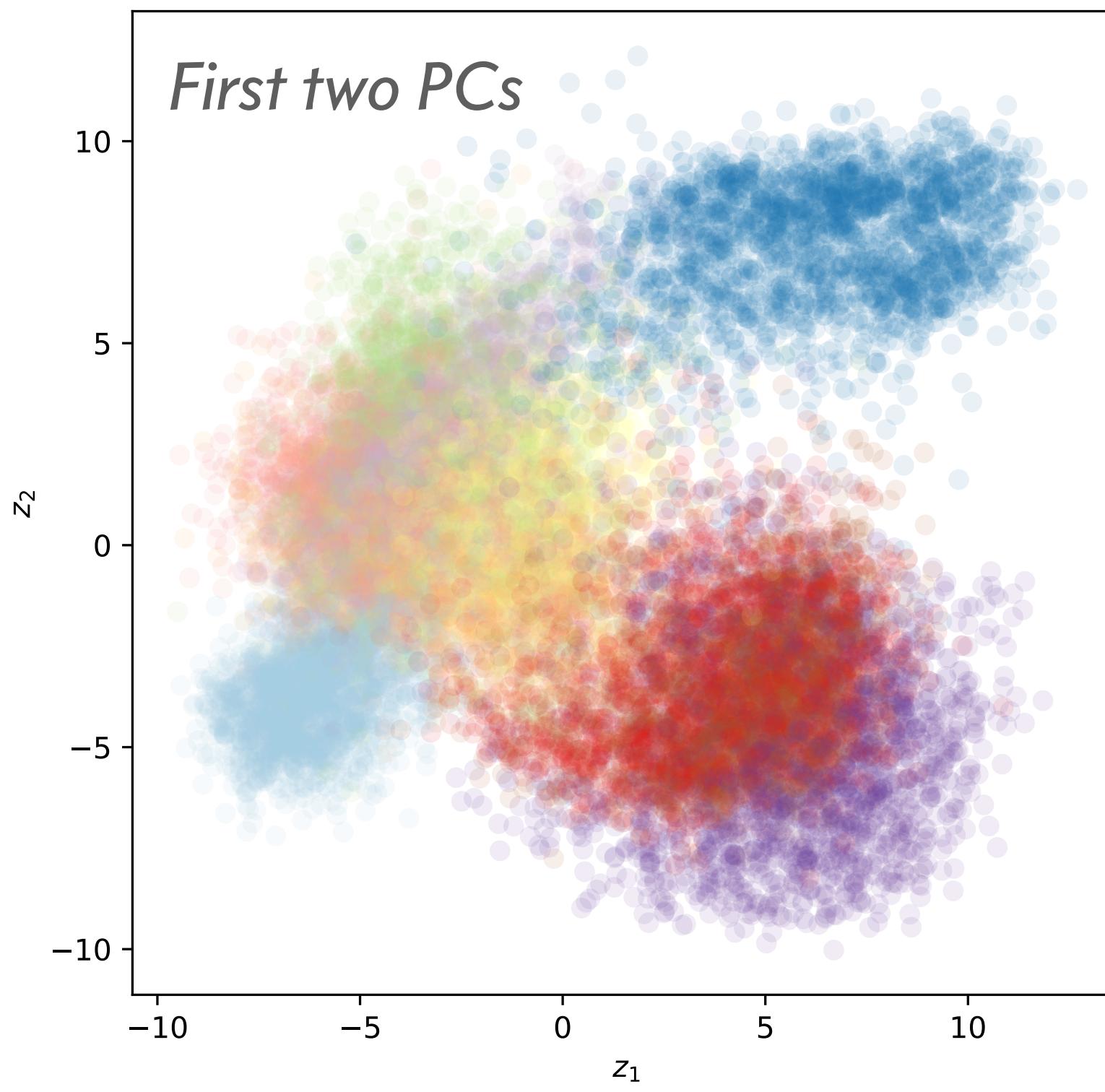
More latent regularization

A semantically meaningful latent space

The KL-term enforces simplicity in the latent space, encouraging learned semantic structure and *disentanglement*

Pure reconstruction

More latent regularization

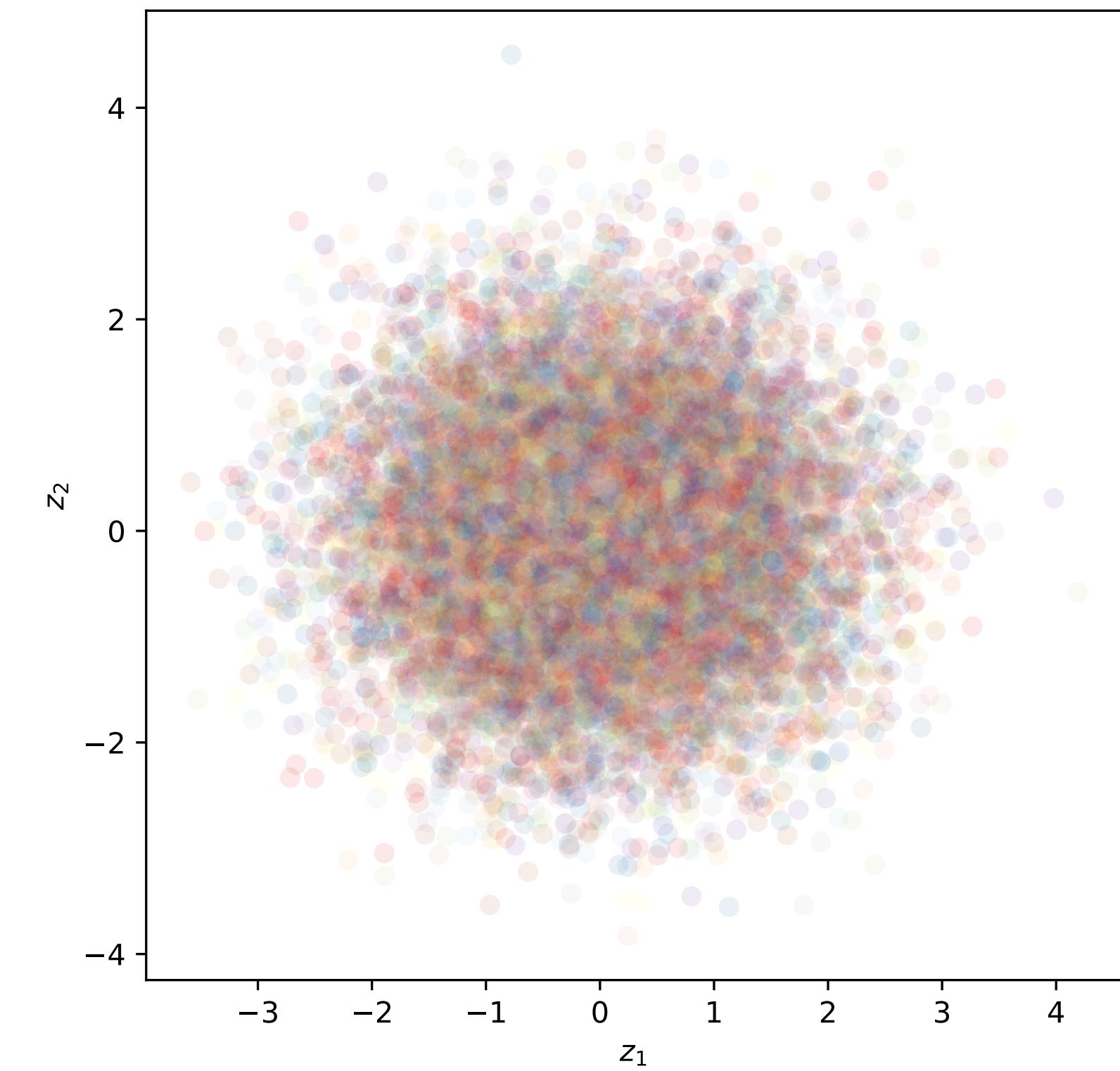
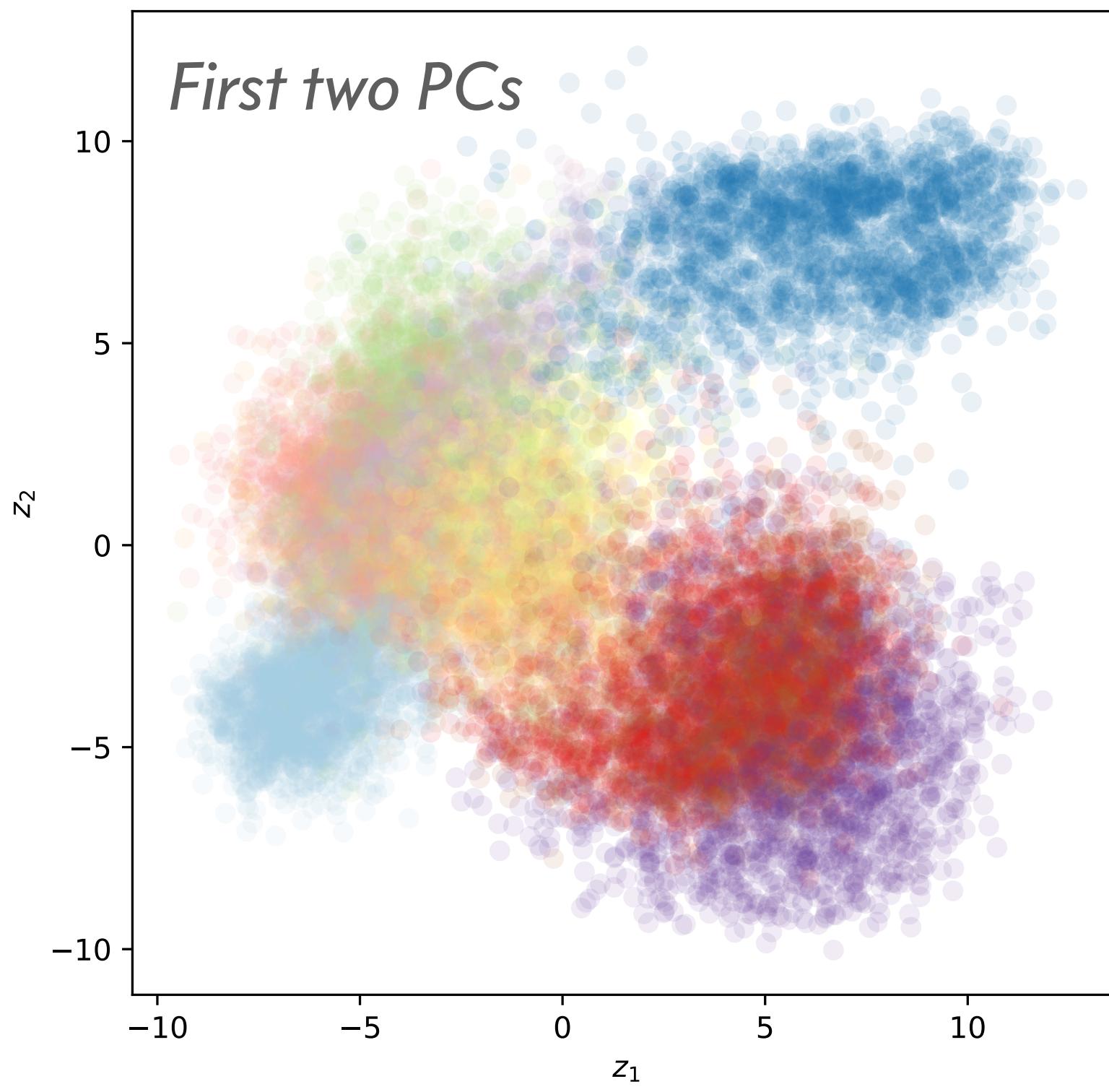


A semantically meaningful latent space

The KL-term enforces simplicity in the latent space, encouraging learned semantic structure and *disentanglement*

Pure reconstruction

More latent regularization

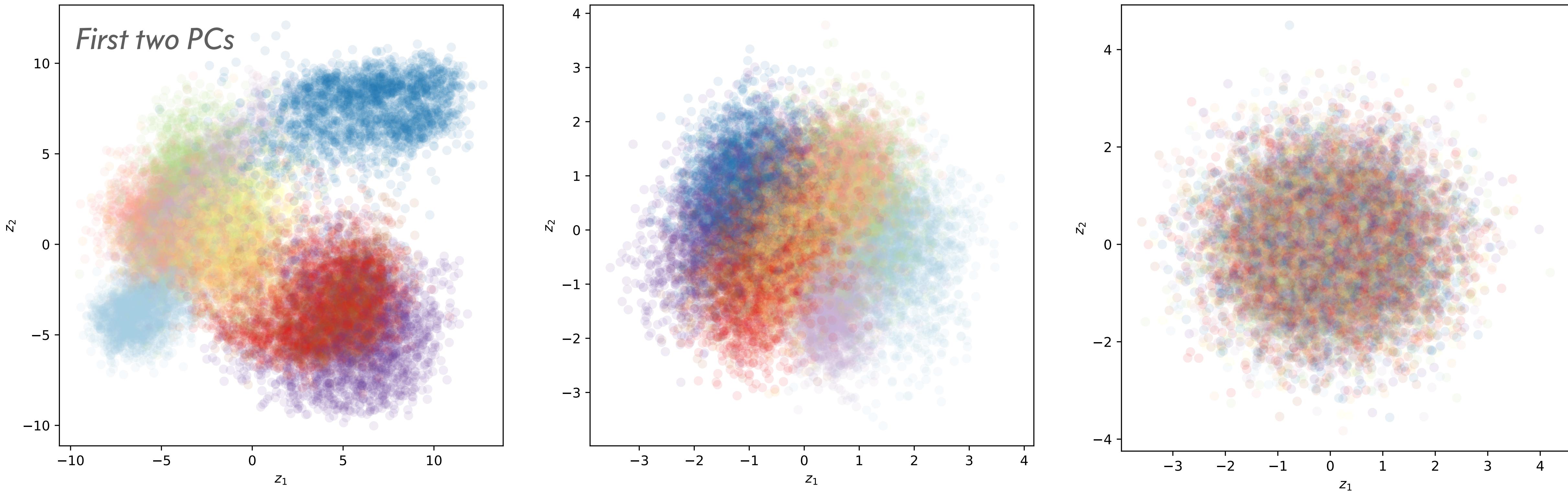


A semantically meaningful latent space

The KL-term enforces simplicity in the latent space, encouraging learned semantic structure and *disentanglement*

Pure reconstruction

More latent regularization



Neural compression: *Rate-distortion theory*

Autoencoding is a form of (neural) compression!

$$-\text{ELBO} = -\left\langle \log p_\theta(x | z) \right\rangle_{q_\varphi} + D_{\text{KL}}\left(q_\varphi(z | x) \| p(z)\right)$$

Neural compression: *Rate-distortion theory*

Autoencoding is a form of (neural) compression!

$$-\text{ELBO} = -\langle \log p_\theta(x | z) \rangle_{q_\phi} + D_{\text{KL}}(q_\phi(z | x) \| p(z))$$

Rate

“Amount of compression”

Neural compression: *Rate-distortion theory*

Autoencoding is a form of (neural) compression!

$$-\text{ELBO} = -\langle \log p_\theta(x | z) \rangle_{q_\varphi} + D_{\text{KL}}(q_\varphi(z | x) \| p(z))$$

Distortion

“Reconstruction loss”

Rate

“Amount of compression”

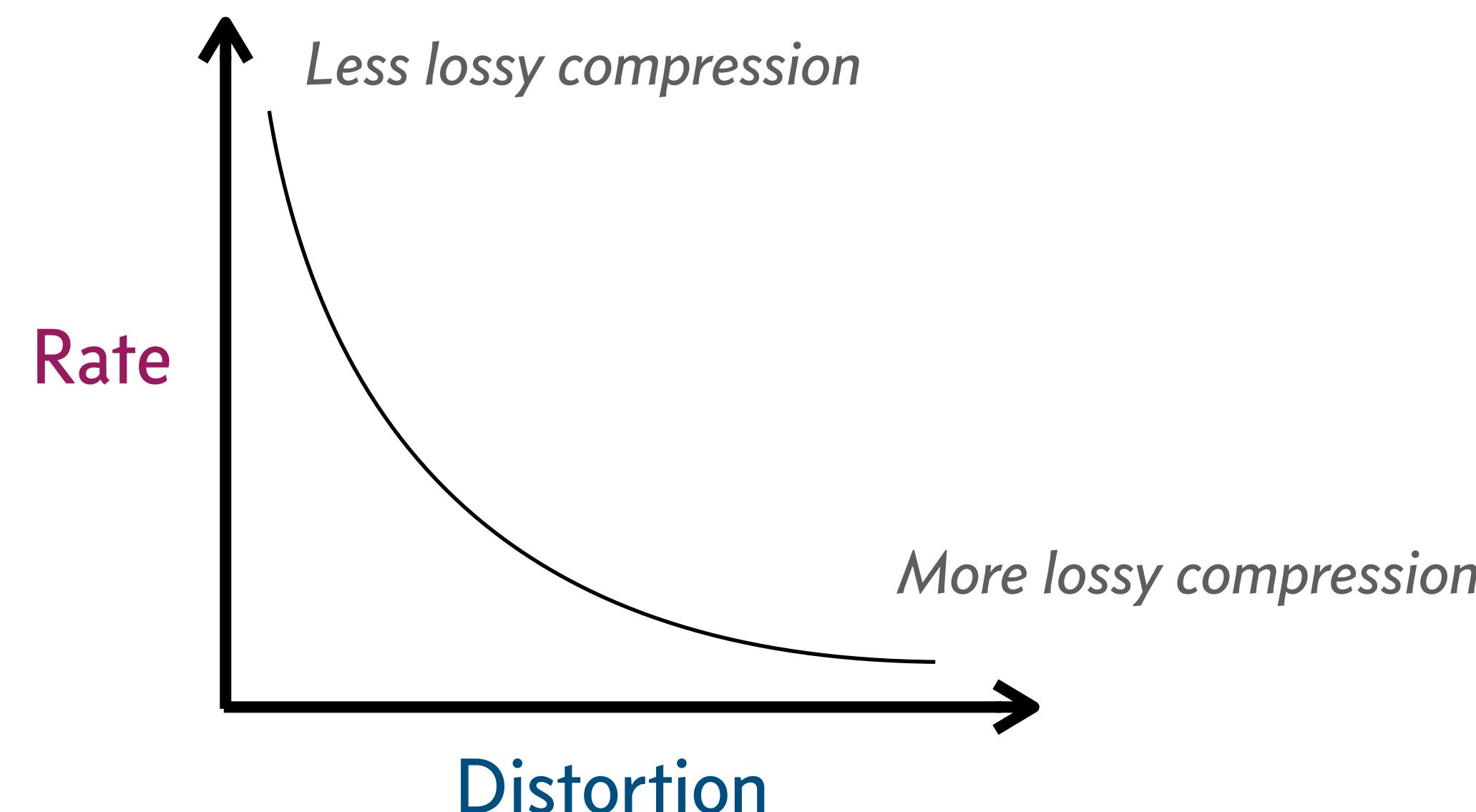
Neural compression: *Rate-distortion theory*

Autoencoding is a form of (neural) compression!

$$-\text{ELBO} = -\langle \log p_\theta(x | z) \rangle_{q_\phi} + D_{\text{KL}}(q_\phi(z | x) \| p(z))$$

— Distortion — Rate
“Reconstruction loss” “Amount of compression”

Rate-distortion curve quantified this tradeoff



Controlling compression and disentanglement: β -VAEs

$$-\text{ELBO} = -\left\langle \log p_{\vartheta}(x | z) \right\rangle_{q_{\phi}} + \beta \cdot D_{\text{KL}} \left(q_{\phi}(z | x) \| p(z) \right)$$

Distortion

Rate

Controlling compression and disentanglement: β -VAEs

$$-\text{ELBO} = -\left\langle \log p_{\vartheta}(x | z) \right\rangle_{q_{\phi}} + \beta \cdot D_{\text{KL}} \left(q_{\phi}(z | x) \| p(z) \right)$$

Distortion

Rate

If the data-generating process is associated with a principled noise model, by using it (the *likelihood*) as the reconstruction loss we are aiming to reconstruct the mean data.

$$\log p(x | z; x') = -\frac{1}{2} \left(\frac{x - x'}{\sigma} \right)^2 + \log \left(\frac{1}{\sigma \sqrt{2\pi}} \right)$$

Controlling compression and disentanglement: β -VAEs

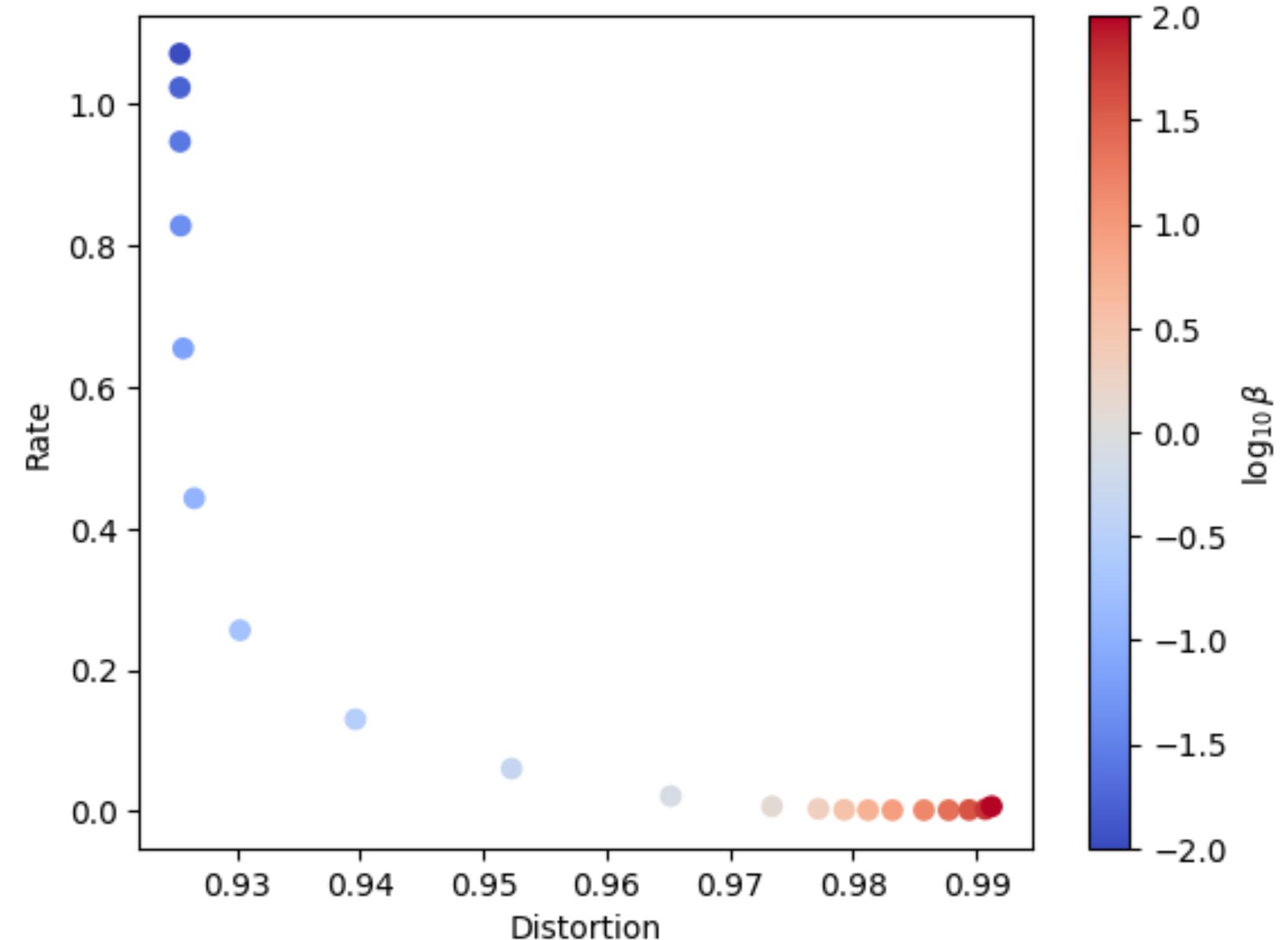
$$-\text{ELBO} = -\left\langle \log p_\vartheta(x | z) \right\rangle_{q_\varphi} + \beta \cdot D_{\text{KL}} \left(q_\varphi(z | x) \| p(z) \right)$$

Distortion

Rate

If the data-generating process is associated with a principled noise model, by using it (the *likelihood*) as the reconstruction loss we are aiming to reconstruct the mean data.

$$\log p(x | z; x') = -\frac{1}{2} \left(\frac{x - x'}{\sigma} \right)^2 + \log \left(\frac{1}{\sigma \sqrt{2\pi}} \right)$$



Controlling compression and disentanglement: β -VAEs

$$-\text{ELBO} = -\left\langle \log p_\vartheta(x | z) \right\rangle_{q_\varphi} + \beta \cdot D_{\text{KL}}(q_\varphi(z | x) \| p(z))$$

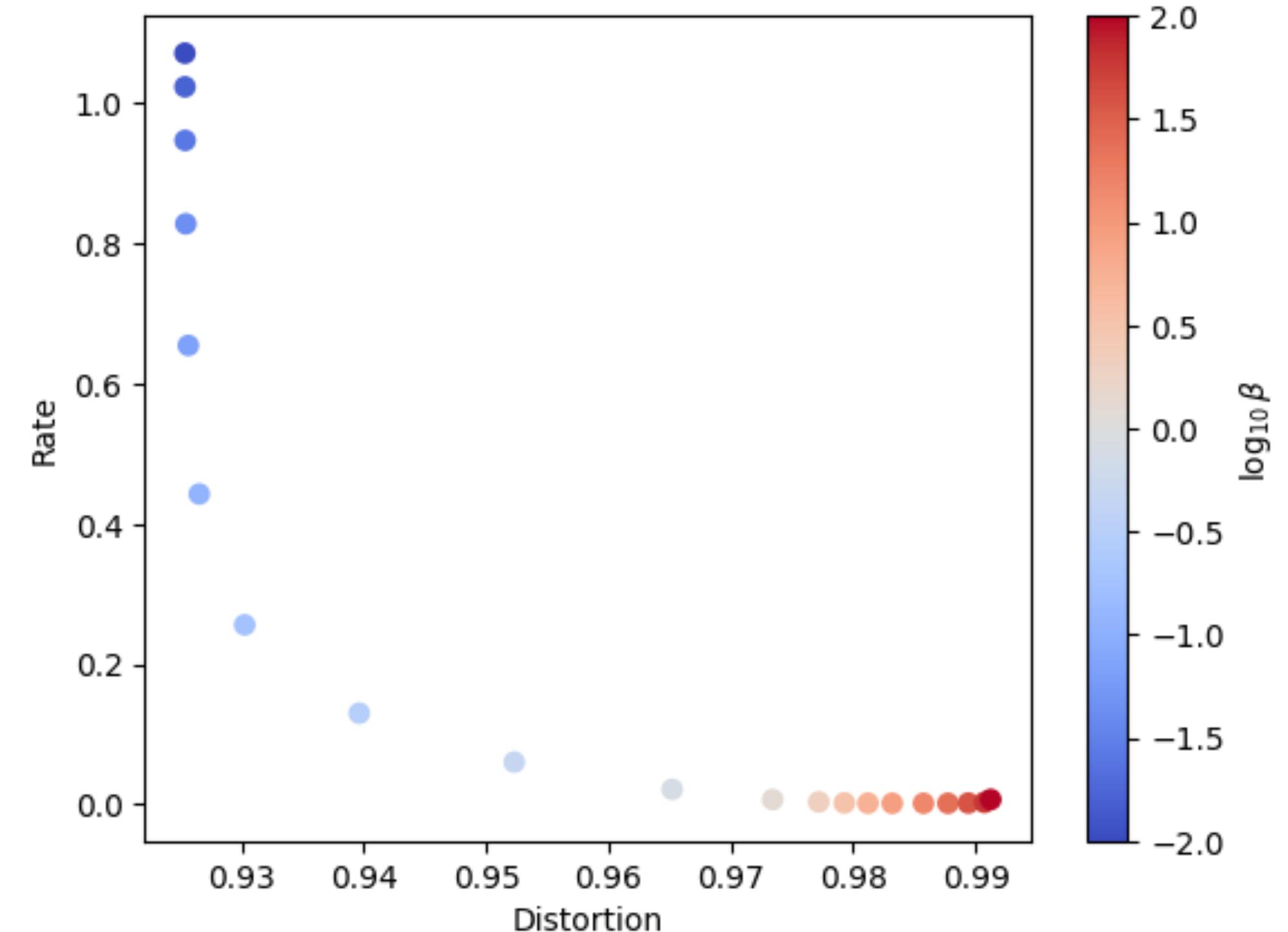
Distortion

Rate

If the data-generating process is associated with a principled noise model, by using it (the *likelihood*) as the reconstruction loss we are aiming to reconstruct the mean data.

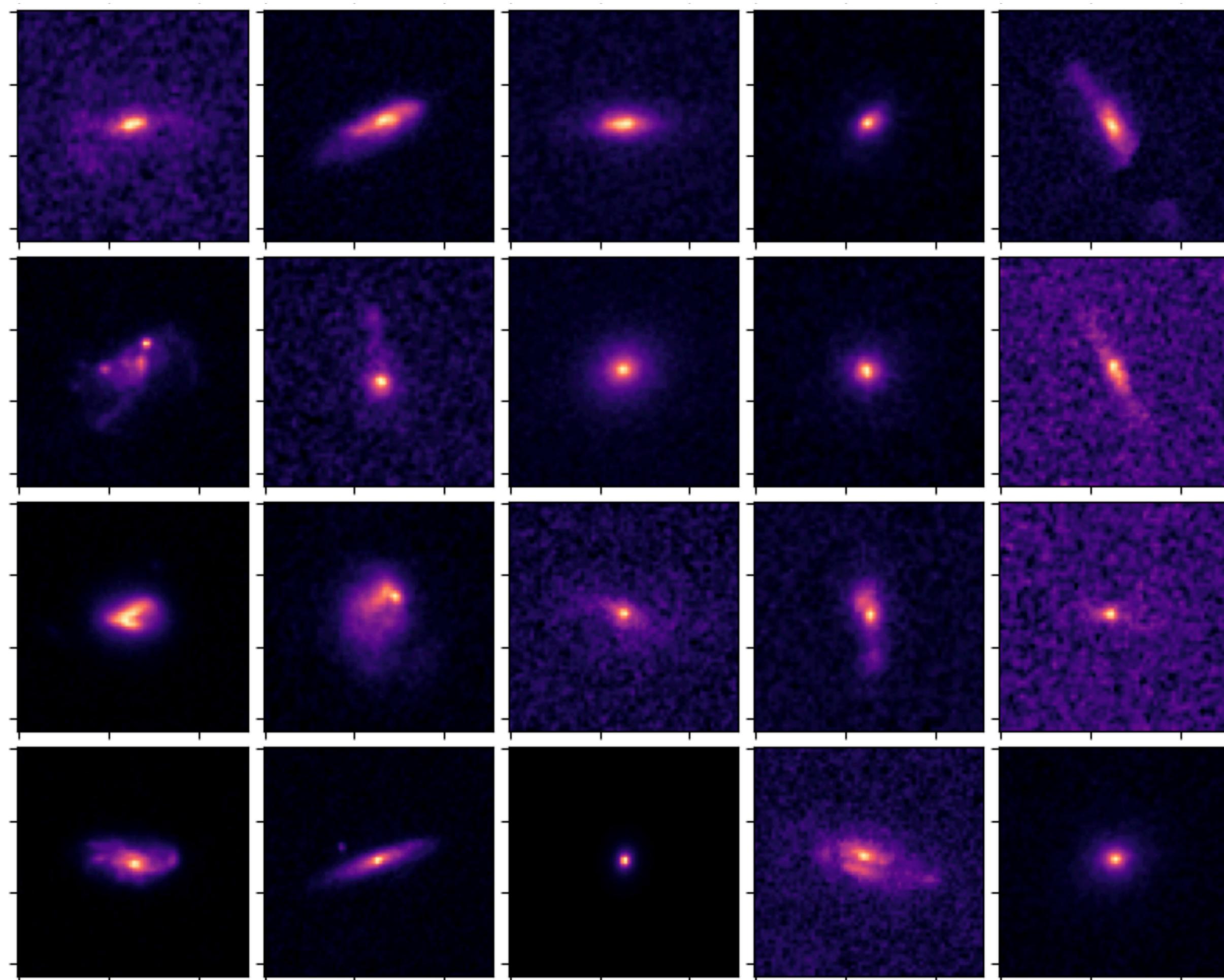
$$\log p(x | z; x') = -\frac{1}{2} \left(\frac{x - x'}{\sigma} \right)^2 + \log \left(\frac{1}{\sigma \sqrt{2\pi}} \right)$$

- Larger σ : More of the data variation is attributed to the likelihood \rightarrow larger " β ", more compression
- Smaller σ : Latents z try to capture more of the variation in the data (e.g. small perceptual features)



Tutorials 1 and 2: variational inference and VAEs

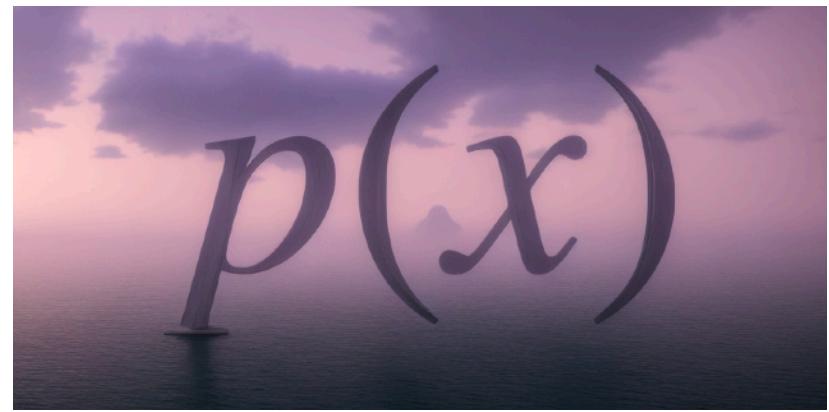
Lead: Carol Cuesta-Lazaro



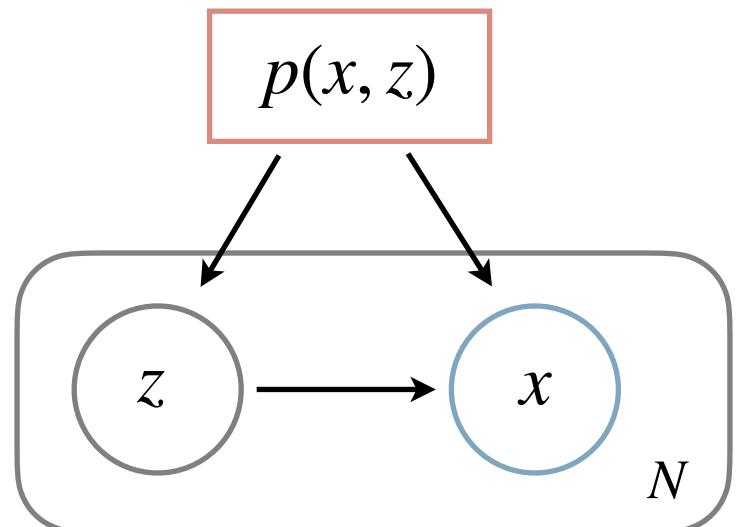
- Implement the ELBO objective for **variational inference**
- Construct a VAE and use it to build a **generative model of galaxy images** using samples from the HST COSMOS dataset
- Boilerplate code for training/reconstruction/ sampling for quick iteration
- Experiment with trade-offs between pure **reconstruction** and a latent space **regularization**

[Mandelbaum et al; <https://zenodo.org/record/3242143>]

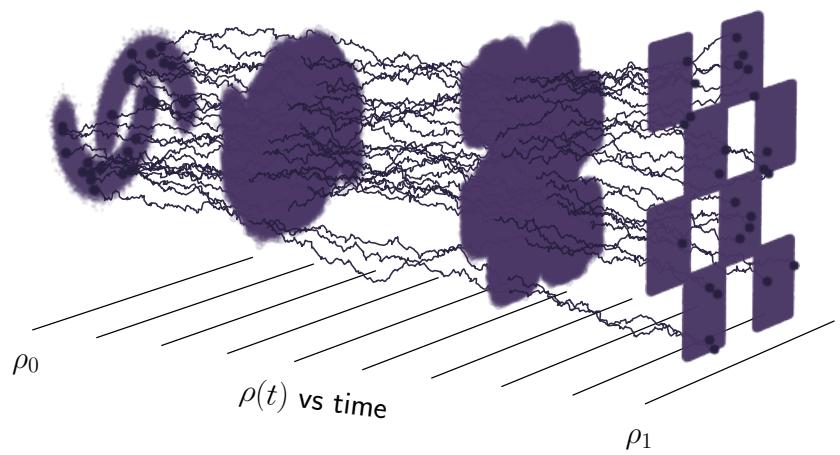
Outline



Why (deep) generative modeling?
What is it, and what can it do for you?



Variational auto encoders
Latent-variable modeling, and compression is all you need



Diffusion models
Models based on iterative refinement

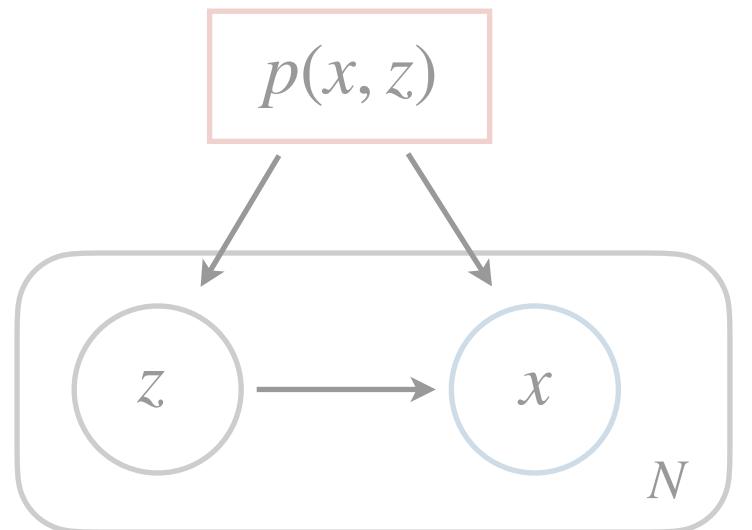


Normalizing flows
Invertible transformations

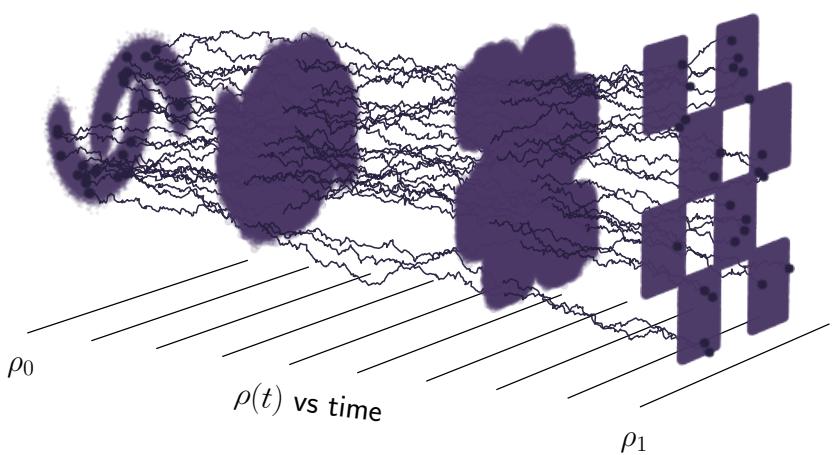
Outline



Why (deep) generative modeling?
What is it, and what can it do for you?



Variational auto encoders
Latent-variable modeling, and compression is all you need



Diffusion models
Models based on iterative refinement



Normalizing flows
Invertible transformations

Diffusion models: overview

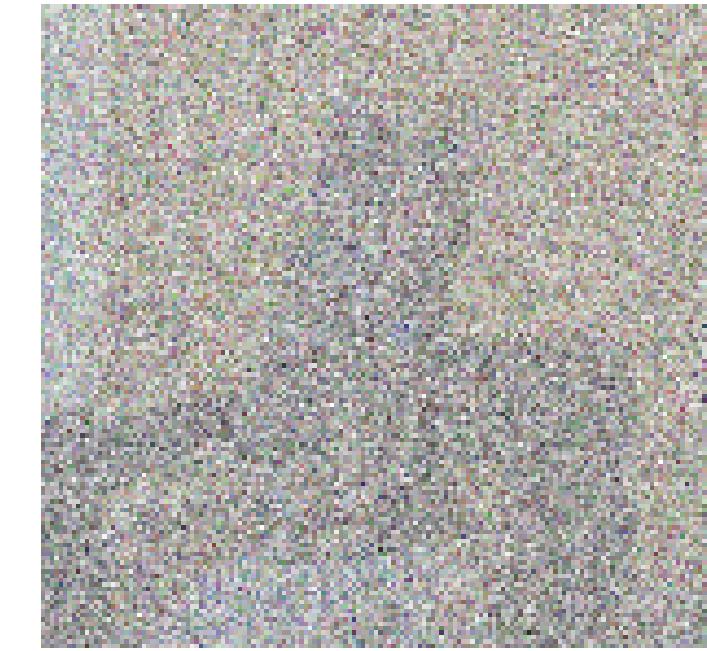
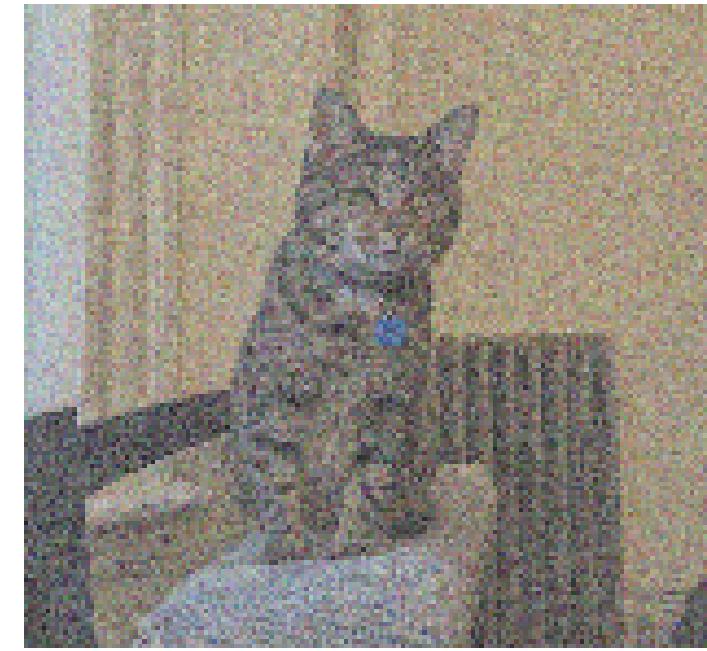
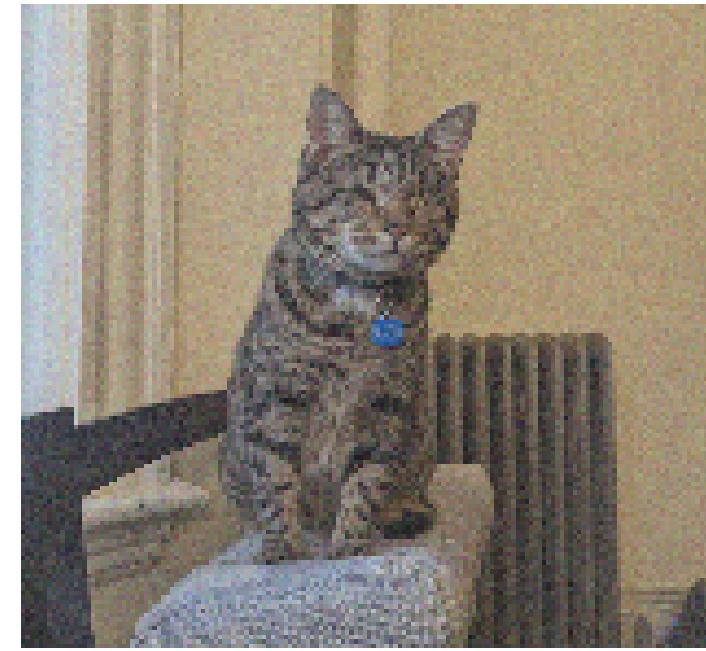
$$x(t=0) \sim p(x)$$



Diffusion models: overview

Forward process (*adding noise*)

$$x(t = 0) \sim p(x)$$

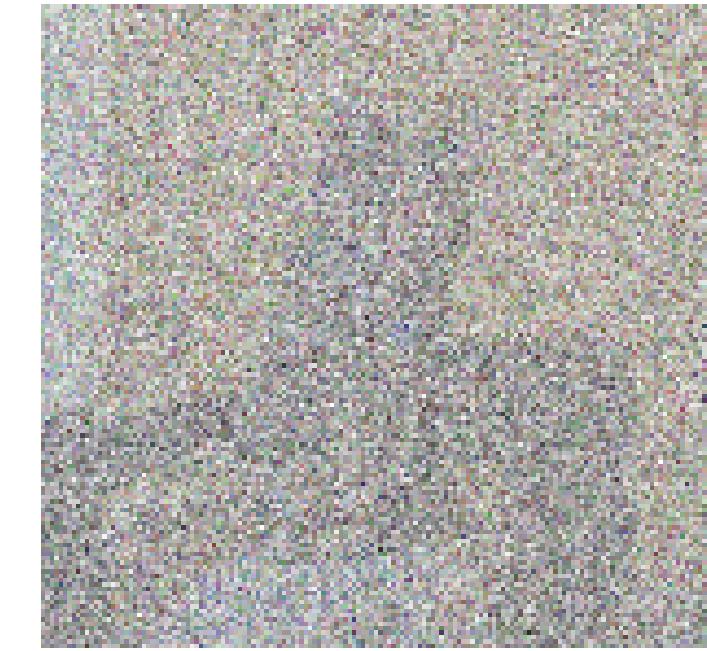
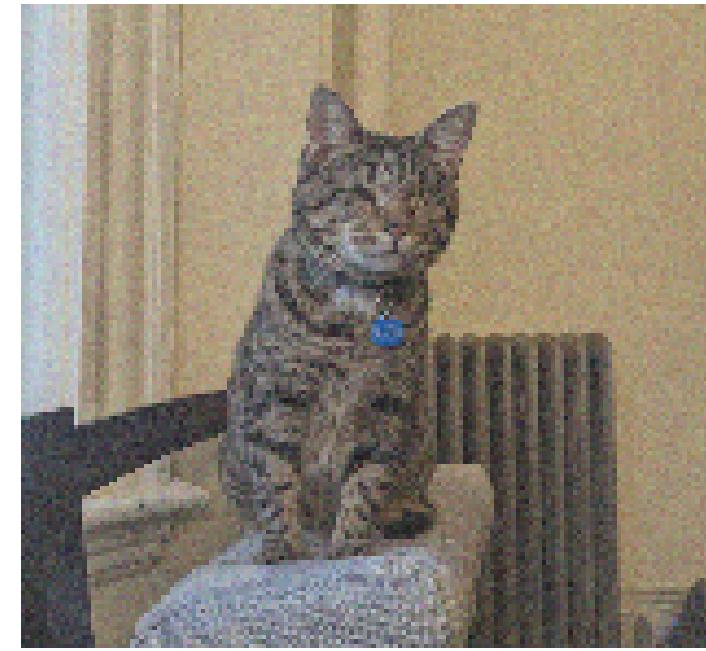
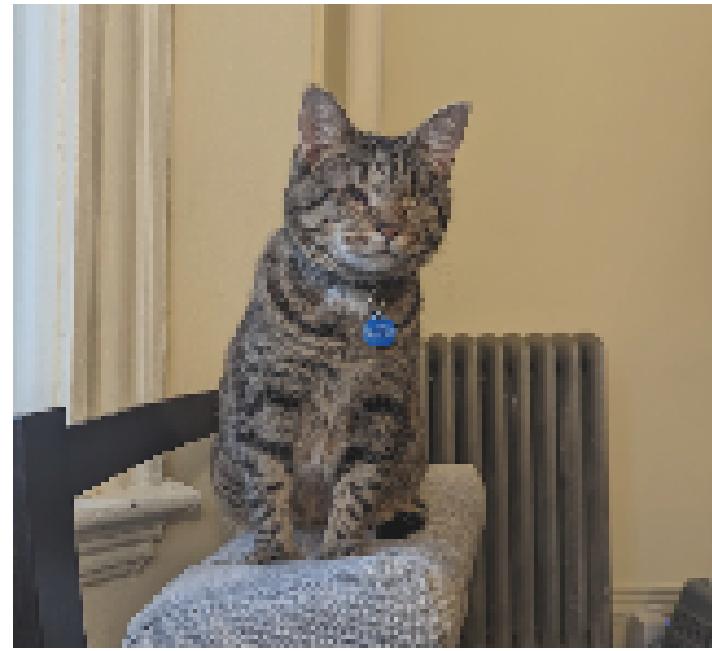


$$x(t = 1) \sim \mathcal{N}(0, 1)$$

Diffusion models: overview

Forward process (*adding noise*)

$$x(t=0) \sim p(x)$$



$$x(t=1) \sim \mathcal{N}(0, 1)$$

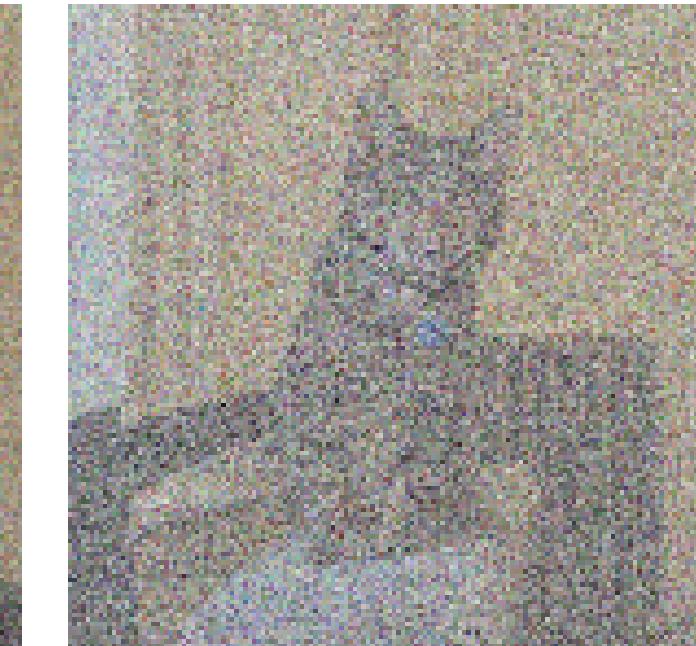
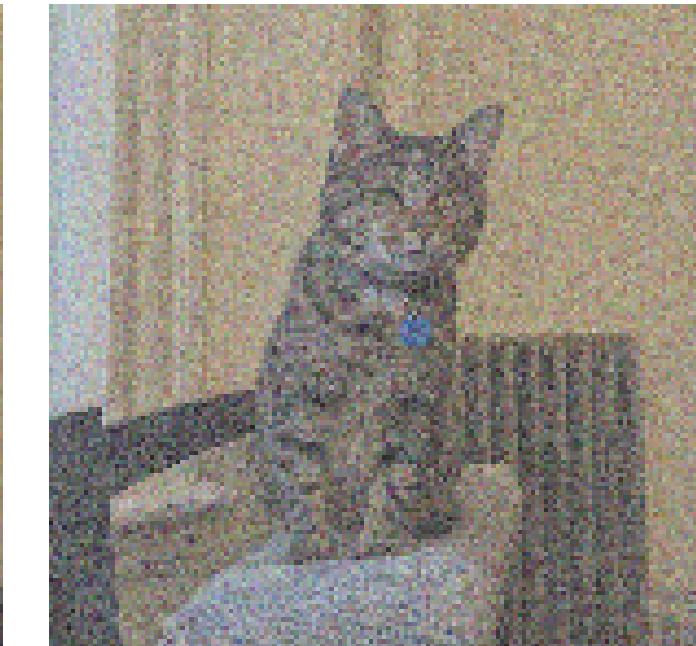
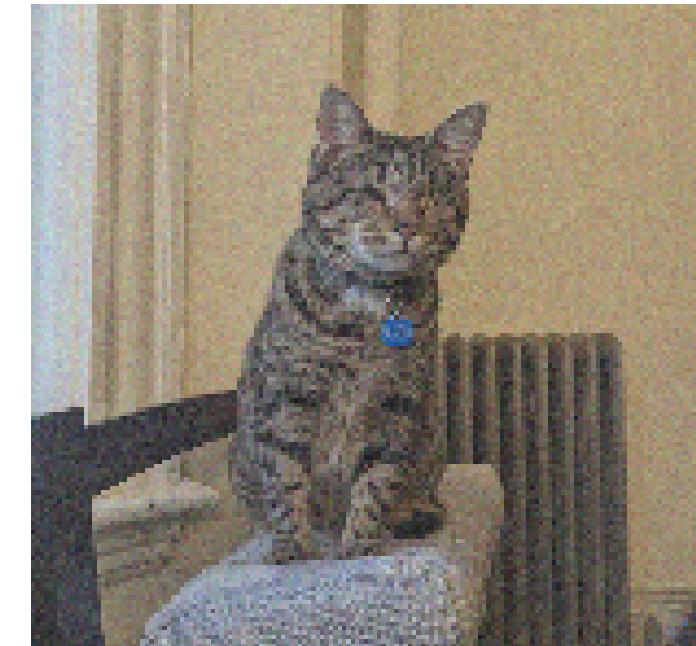
$$x_{t-1} \sim p(x_{t-1} \mid x_t)$$

Reverse process (*denoising*)

Diffusion models: overview

Forward process (adding noise)

$$x(t=0) \sim p(x)$$



$$x(t=1) \sim \mathcal{N}(0, 1)$$

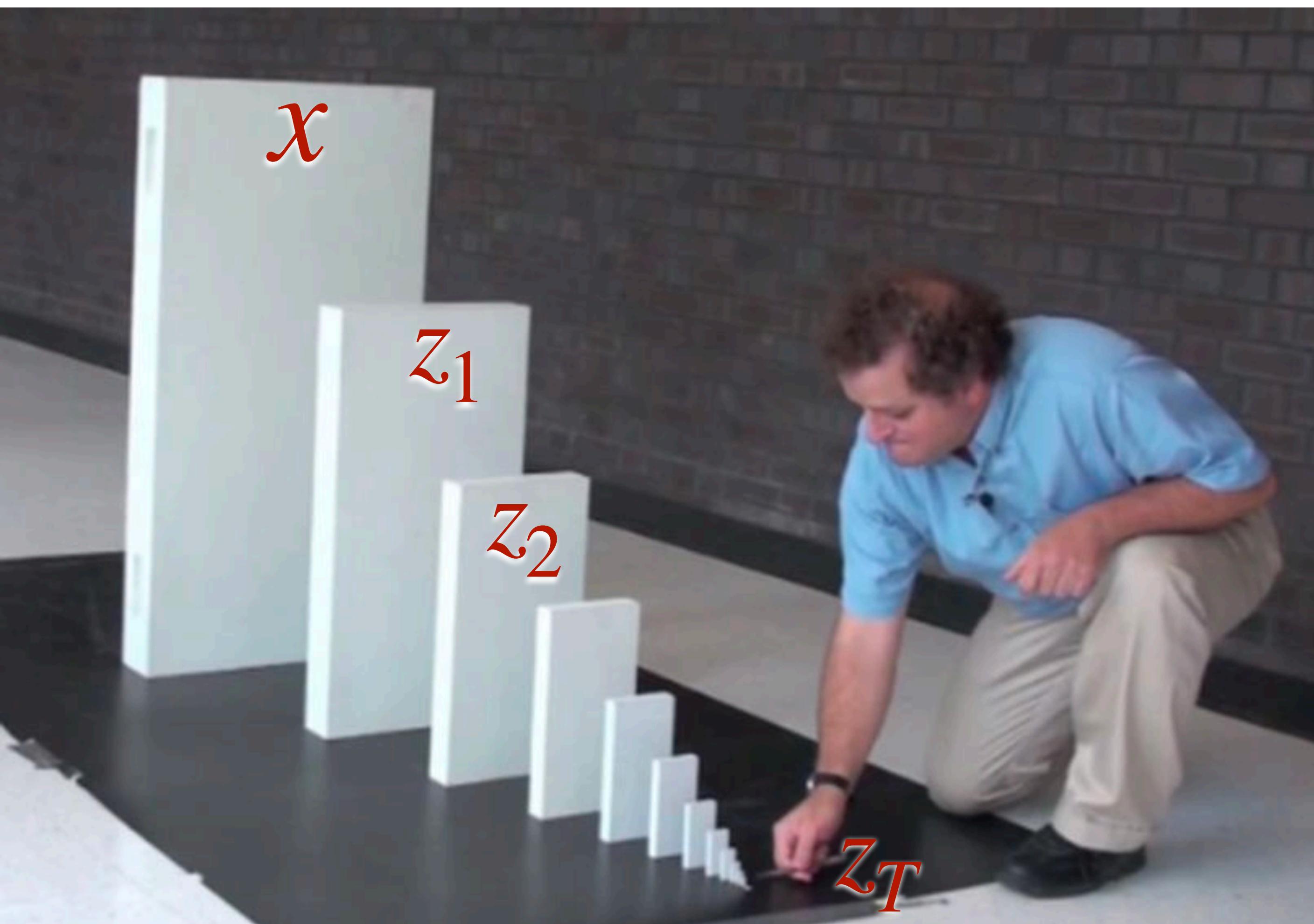
$$x_{t-1} \sim p(x_{t-1} | x_t)$$

Reverse process (denoising)

Prompt: "A cat perched on an Ikea Poang chair"

image $\sim p(\text{image} | \text{text prompt})$

Towards diffusion: hierarchical VAEs



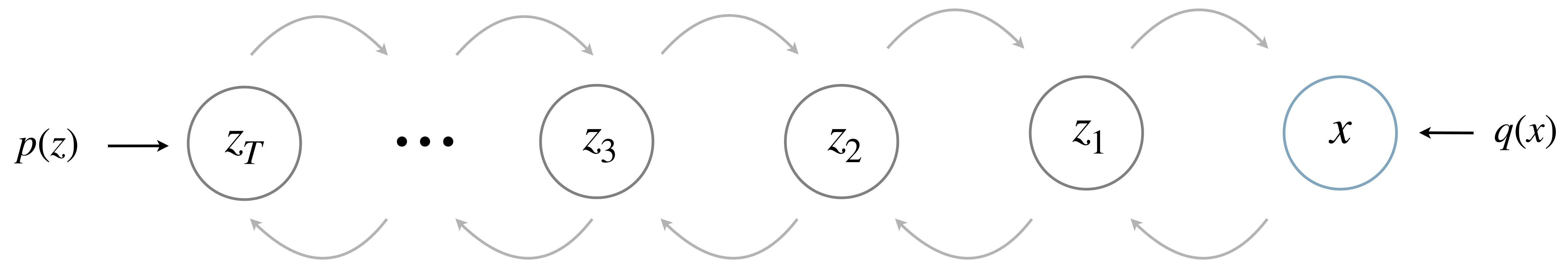
Towards diffusion: (Markovian) hierarchical VAEs

Reverse process $p(x, z_1, z_2, \dots, z_T) = p(z_T) p(z_{T-1} | z_T) \cdots p(z_1 | z_2) p(x | z_1)$

Forward process $q(x, z_1, z_2, \dots, z_T) = q(x) q(z_1 | x) q(z_2 | z_1) \cdots q(z_T | z_{T-1})$

Towards diffusion: (Markovian) hierarchical VAEs

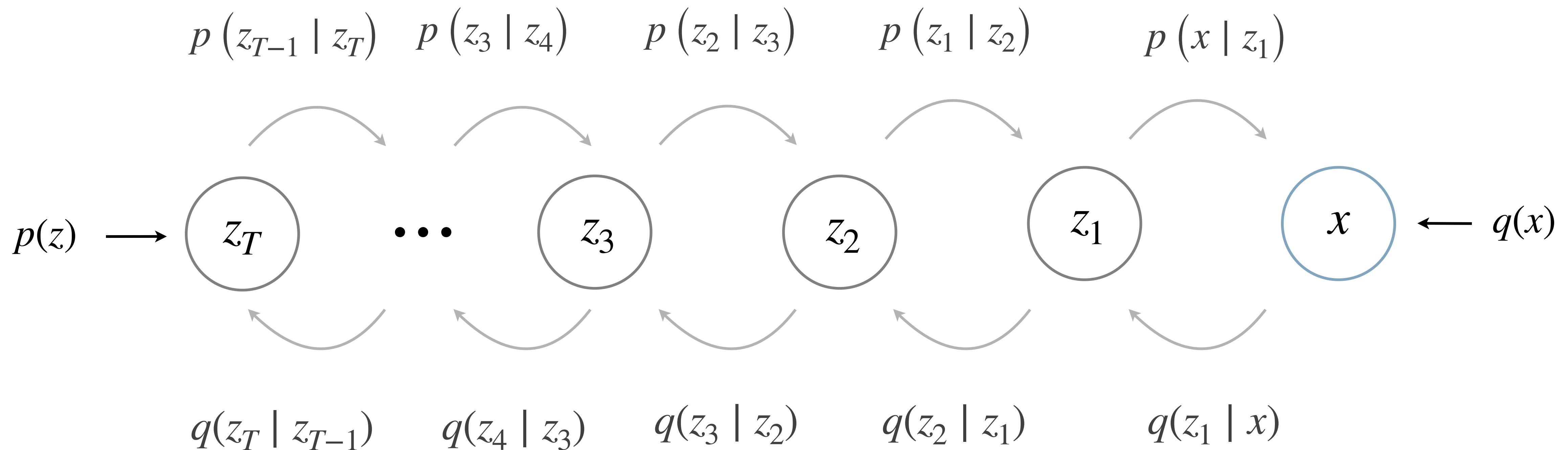
Reverse process $p(x, z_1, z_2, \dots, z_T) = p(z_T) p(z_{T-1} | z_T) \cdots p(z_1 | z_2) p(x | z_1)$



Forward process $q(x, z_1, z_2, \dots, z_T) = q(x) q(z_1 | x) q(z_2 | z_1) \cdots q(z_T | z_{T-1})$

Towards diffusion: (Markovian) hierarchical VAEs

Reverse process $p(x, z_1, z_2, \dots, z_T) = p(z_T) p(z_{T-1} | z_T) \cdots p(z_1 | z_2) p(x | z_1)$



Forward process $q(x, z_1, z_2, \dots, z_T) = q(x) q(z_1 | x) q(z_2 | z_1) \cdots q(z_T | z_{T-1})$

Towards diffusion: (Markovian) hierarchical VAEs

Diffusion models can be seen as hierarchical VAEs with a few restrictions:

- The forward (*encoding*) distribution prescribed as a Markov chain of Gaussians; it is not learned

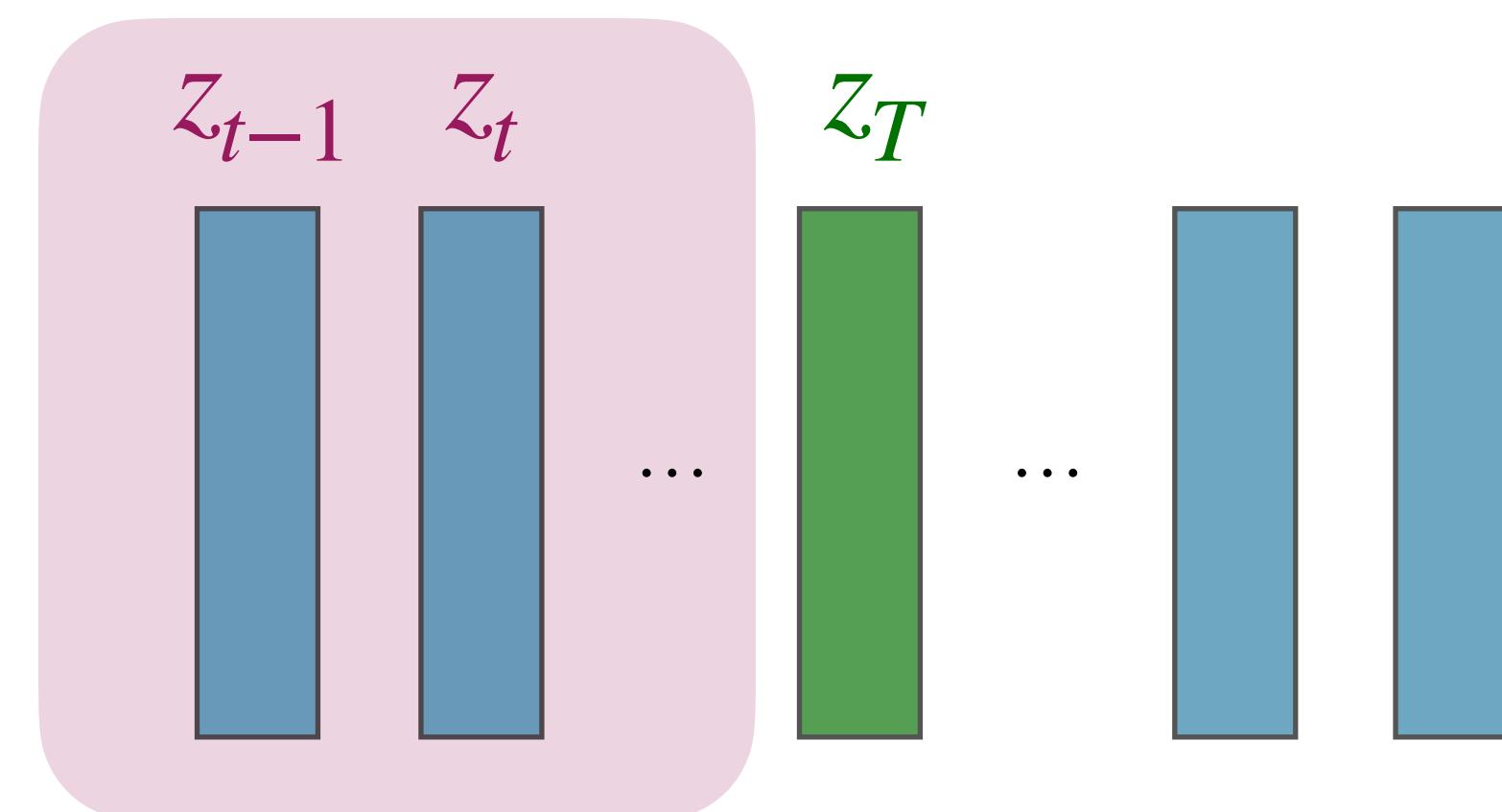
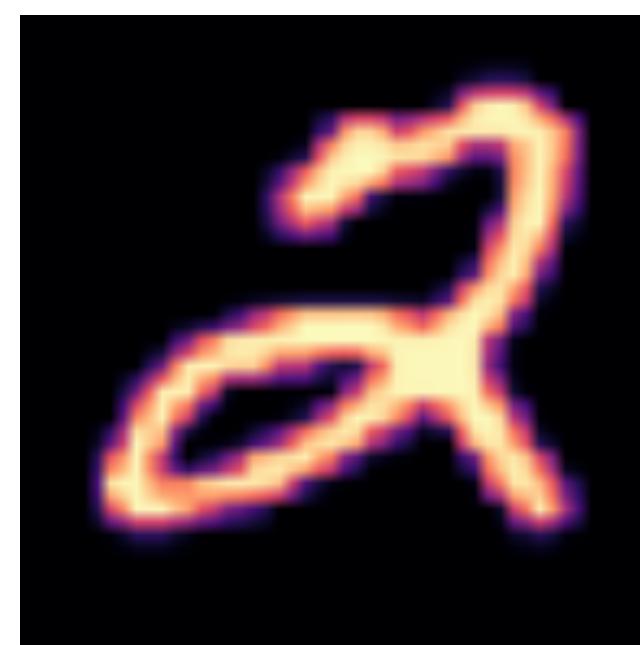
$$q(z_t | z_{t-1}) = \mathcal{N}(z_t; \alpha_t z_{t-1}, \beta_t)$$

- Distributions of latents at the final timestep T is a standard (unit) Gaussian

$$q(z_T | z_{T-1}, \dots, x) = \mathcal{N}(z_T; 0, \mathbb{I})$$

- The dimensionality of latents is the same as the data dimensionality

$$\dim(z_t) = \dim(x)$$



Variational diffusion models

Align the forward and reverse distributions;
variational lower bound (ELBO) as before

[Kingma et al 2021]

[Gory details: Luo 2022; [2208.11970](#)]

$$L = \left\langle \log \frac{q(x, z_1, z_2, \dots, z_T)}{p(x, z_1, z_2, \dots, z_T)} \right\rangle_{q(x)}$$

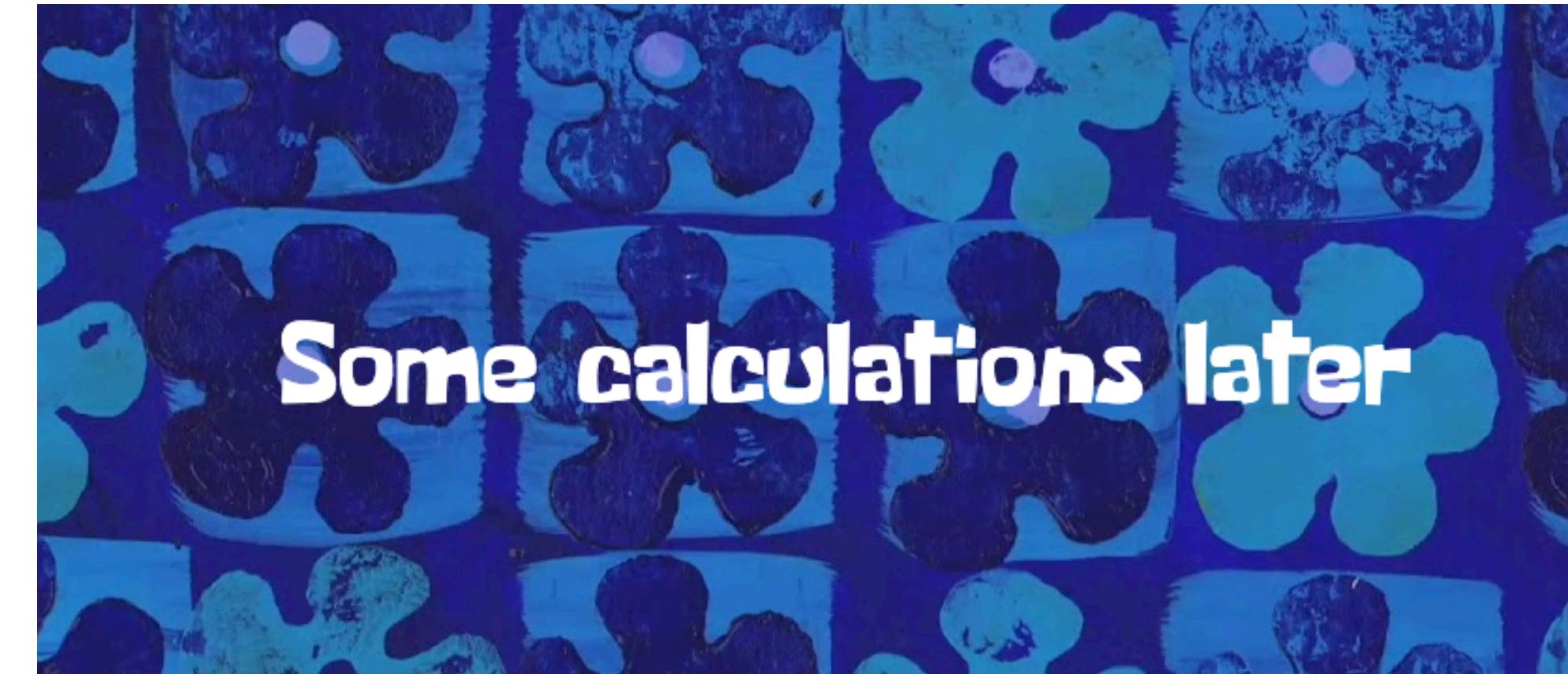
Variational diffusion models

Align the forward and reverse distributions;
variational lower bound (ELBO) as before

$$L = \left\langle \log \frac{q(x, z_1, z_2, \dots, z_T)}{p(x, z_1, z_2, \dots, z_T)} \right\rangle_{q(x)}$$

[Kingma et al 2021]

[Gory details: Luo 2022; [2208.11970](#)]



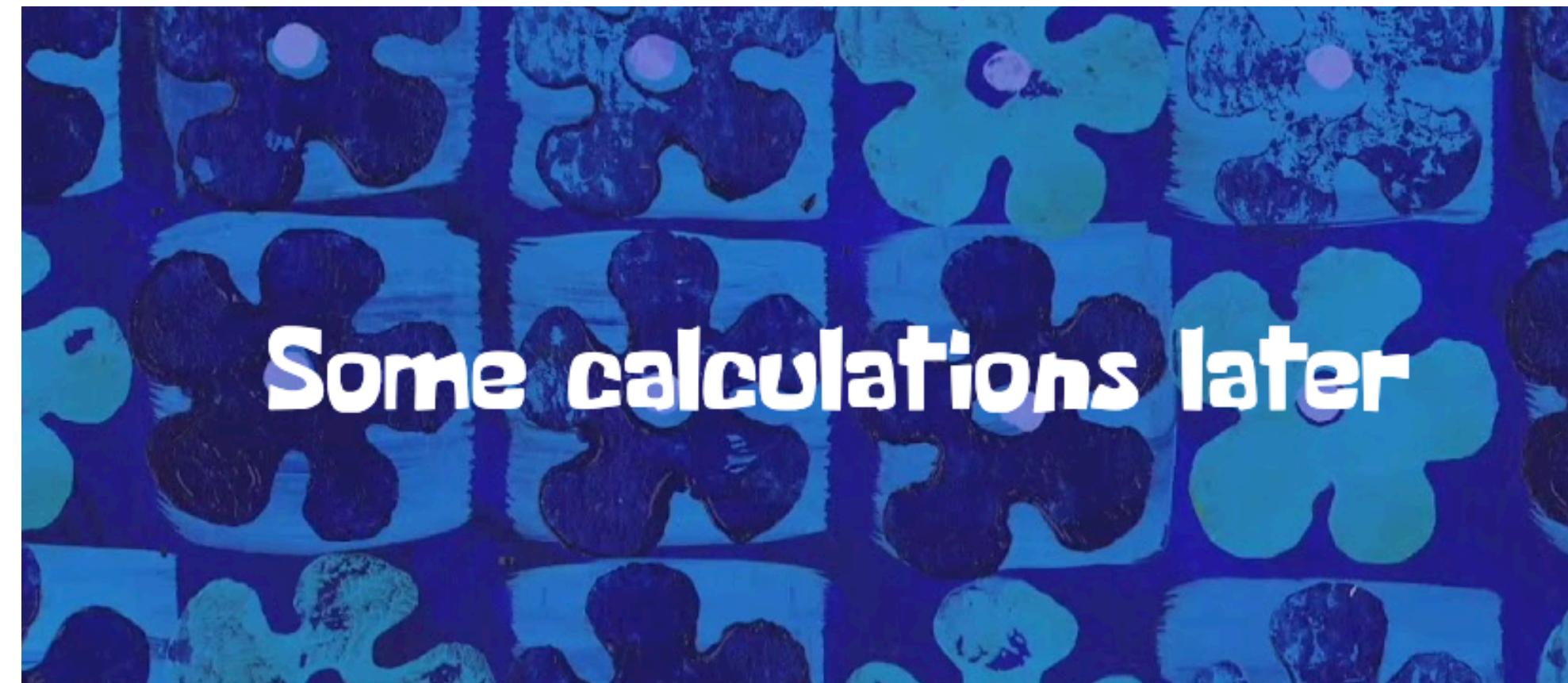
Variational diffusion models

Align the forward and reverse distributions;
variational lower bound (ELBO) as before

$$L = \left\langle \log \frac{q(x, z_1, z_2, \dots, z_T)}{p(x, z_1, z_2, \dots, z_T)} \right\rangle_{q(x)}$$

[Kingma et al 2021]

[Gory details: Luo 2022; [2208.11970](#)]



$$L = \left\langle \log p_\vartheta(x | z_1) \right\rangle_{q(z_1|x)} - D_{\text{KL}}(q(z_T | x) \| p(z_T)) - \sum_{t=2}^T \left\langle D_{\text{KL}}(q(z_{t-1} | z_t, x) \| p_\vartheta(z_{t-1} | z_t)) \right\rangle_{q(z_t|x)}$$

Reconstruction

(Noise model; no trainable parameters)

Prior regularization

(No trainable parameters)

Denoising matching

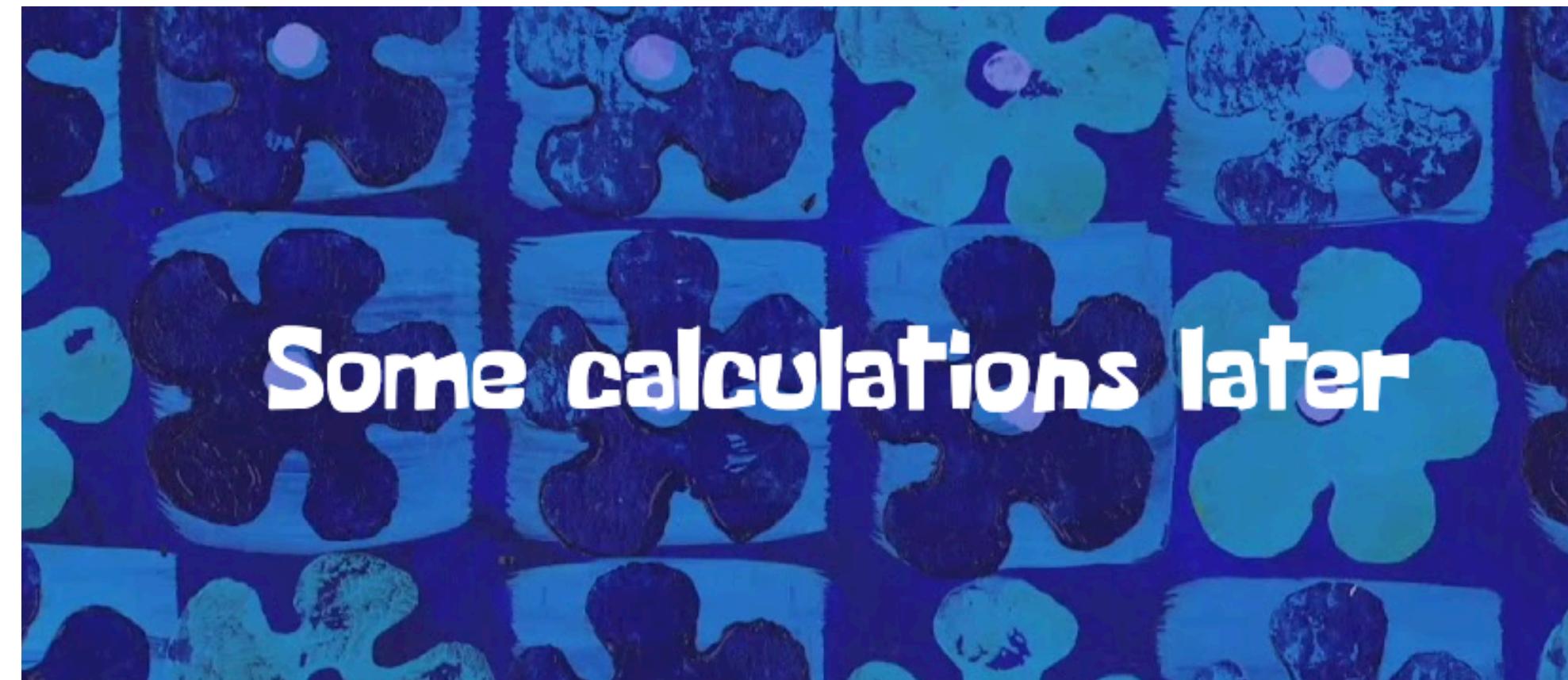
Variational diffusion models

Align the forward and reverse distributions;
variational lower bound (ELBO) as before

$$L = \left\langle \log \frac{q(x, z_1, z_2, \dots, z_T)}{p(x, z_1, z_2, \dots, z_T)} \right\rangle_{q(x)}$$

[Kingma et al 2021]

[Gory details: Luo 2022; [2208.11970](#)]



$$L = \left\langle \log p_\vartheta(x | z_1) \right\rangle_{q(z_1|x)} - D_{\text{KL}}(q(z_T | x) \| p(z_T)) - \sum_{t=2}^T \left\langle D_{\text{KL}}(q(z_{t-1} | z_t, x) \| p_\vartheta(z_{t-1} | z_t)) \right\rangle_{q(z_t|x)}$$

Reconstruction

(Noise model; no trainable parameters)

Prior regularization

(No trainable parameters)

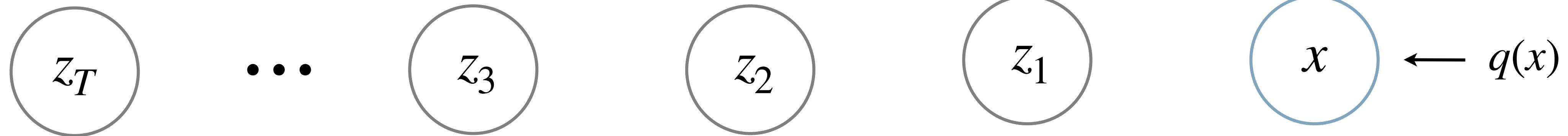
Denoising matching

ELBO! Bound on $p(x)$

The forward process and diffusion kernel

Predict arbitrary timestep without Markovian sampling

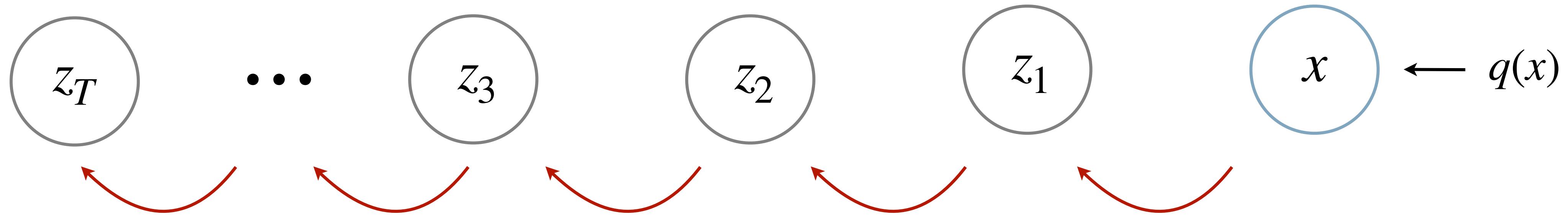
$$\sum_{t=2}^T \left\langle D_{\text{KL}} \left(q(z_{t-1} | z_t, x) \| p_\vartheta(z_{t-1} | z_t) \right) \right\rangle_{q(z_t|x)}$$



The forward process and diffusion kernel

Predict arbitrary timestep without Markovian sampling

$$\sum_{t=2}^T \left\langle D_{\text{KL}} \left(q(z_{t-1} | z_t, x) \| p_\vartheta(z_{t-1} | z_t) \right) \right\rangle_{q(z_t|x)}$$



Variance-preserving noise schedule

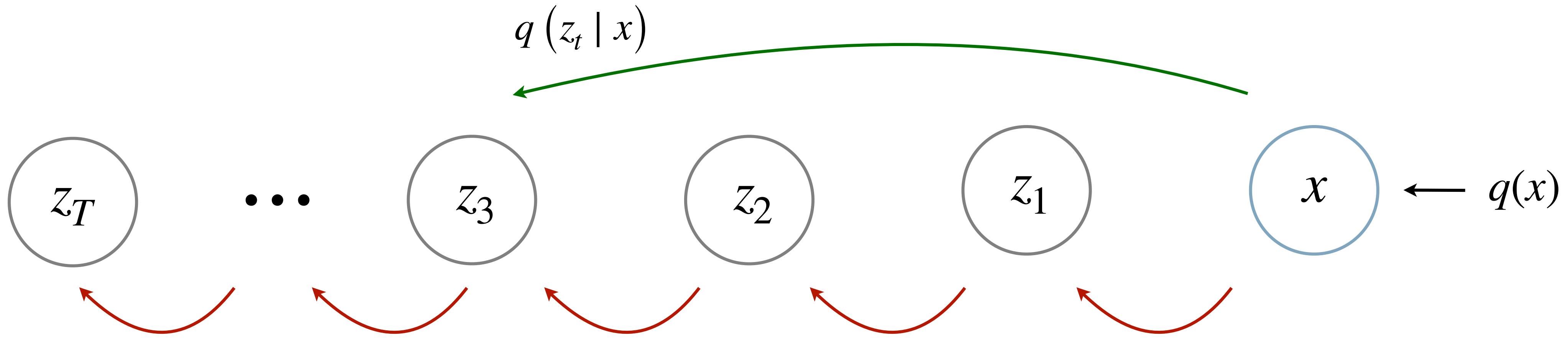
$$q(z_t | z_{t-1}) = \mathcal{N} \left(\sqrt{1 - \beta_t} \cdot z_{t-1}, \beta_t \right)$$

$$z_t = \sqrt{1 - \beta_t} \cdot z_{t-1} + \sqrt{\beta_t} \cdot \varepsilon$$

The forward process and diffusion kernel

Predict arbitrary timestep without Markovian sampling

$$\sum_{t=2}^T \left\langle D_{\text{KL}} \left(q(z_{t-1} | z_t, x) \| p_\vartheta(z_{t-1} | z_t) \right) \right\rangle_{q(z_t | x)}$$



Variance-preserving noise schedule

$$q(z_t | z_{t-1}) = \mathcal{N} \left(\sqrt{1 - \beta_t} \cdot z_{t-1}, \beta_t \right)$$

$$z_t = \sqrt{1 - \beta_t} \cdot z_{t-1} + \sqrt{\beta_t} \cdot \varepsilon$$

Diffusion kernel

$$q(z_t | x) = \mathcal{N} \left(\sqrt{\alpha_t} \cdot x, \sqrt{1 - \alpha_t} \right)$$

$$\bar{\alpha}_t = \prod_{i=1}^t \alpha_i$$

The denoising objective

Given the nature of the forward (noising) process,
 $q(z_{t-1} | z_t, x)$ can be computed analytically

$$q(z_{t-1} | z_t, x) = \mathcal{N}\left(z_{t-1}; \mu_q(x_t, x_0), \sigma_q(t)\mathbb{I}\right)$$

$$\sum_{t=2}^T \left\langle D_{\text{KL}} \left(q(z_{t-1} | z_t, x) \| p_\vartheta(z_{t-1} | z_t) \right) \right\rangle_{q(z_t|x)}$$

The denoising objective

Given the nature of the forward (noising) process,
 $q(z_{t-1} | z_t, x)$ can be computed analytically

$$\sum_{t=2}^T \left\langle D_{\text{KL}} \left(q(z_{t-1} | z_t, x) \| p_{\vartheta}(z_{t-1} | z_t) \right) \right\rangle_{q(z_t|x)}$$

$$q(z_{t-1} | z_t, x) = \mathcal{N}(z_{t-1}; \mu_q(x_t, x_0), \sigma_q(t)\mathbb{I})$$

$$p_{\vartheta}(z_{t-1} | z_t, x) = \mathcal{N}(z_{t-1}; \mu_{\vartheta}(x_t, x_0), \sigma_{\vartheta}(t)\mathbb{I})$$

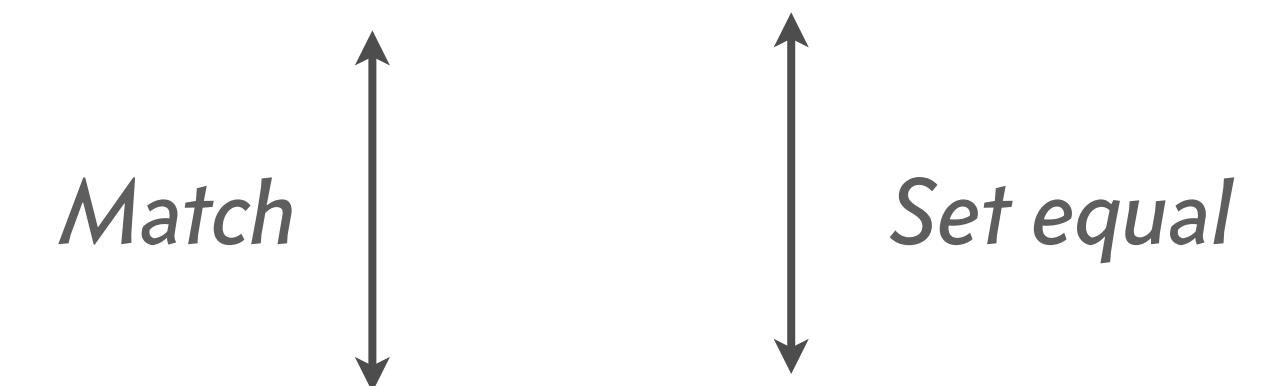
Learnable denoising distribution; assume Gaussian

The denoising objective

Given the nature of the forward (noising) process,
 $q(z_{t-1} | z_t, x)$ can be computed analytically

$$\sum_{t=2}^T \left\langle D_{\text{KL}} \left(q(z_{t-1} | z_t, x) \| p_{\vartheta}(z_{t-1} | z_t) \right) \right\rangle_{q(z_t|x)}$$

$$q(z_{t-1} | z_t, x) = \mathcal{N} \left(z_{t-1}; \mu_q(x_t, x_0), \sigma_q(t) \mathbb{I} \right)$$



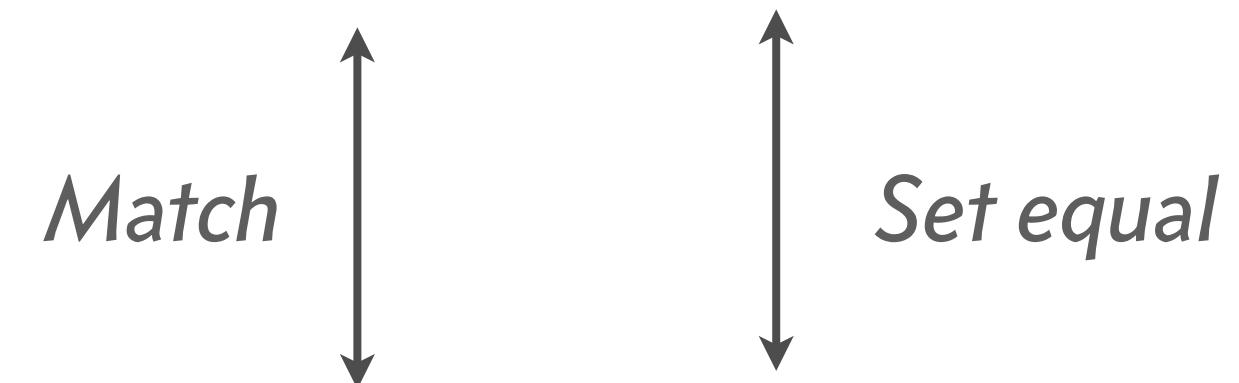
$$p_{\vartheta}(z_{t-1} | z_t, x) = \mathcal{N} \left(z_{t-1}; \mu_{\vartheta}(x_t, x_0), \sigma_{\vartheta}(t) \mathbb{I} \right)$$

Learnable denoising distribution; assume Gaussian

The denoising objective

Given the nature of the forward (noising) process,
 $q(z_{t-1} | z_t, x)$ can be computed analytically

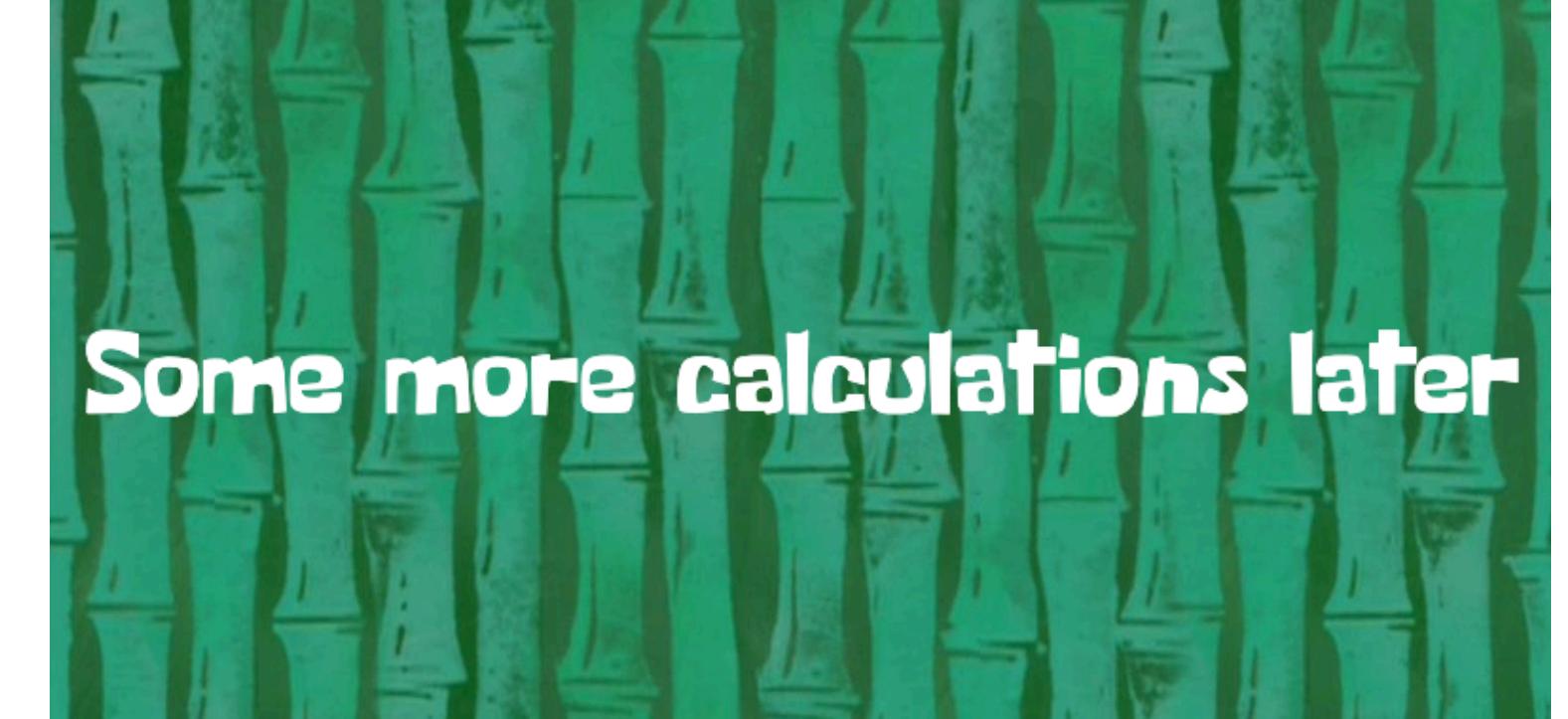
$$q(z_{t-1} | z_t, x) = \mathcal{N}(z_{t-1}; \mu_q(x_t, x_0), \sigma_q(t)\mathbb{I})$$



$$p_\vartheta(z_{t-1} | z_t, x) = \mathcal{N}(z_{t-1}; \mu_\vartheta(x_t, x_0), \sigma_\vartheta(t)\mathbb{I})$$

Learnable denoising distribution; assume Gaussian

$$\sum_{t=2}^T \left\langle D_{\text{KL}} \left(q(z_{t-1} | z_t, x) \| p_\vartheta(z_{t-1} | z_t) \right) \right\rangle_{q(z_t|x)}$$



Denoising loss

$$\frac{1}{2\sigma_q^2(t)} \frac{\bar{\alpha}_{t-1} (1 - \alpha_t)^2}{(1 - \bar{\alpha}_t)^2} \left[\|\hat{x}_\theta(z_t, t) - x\|^2 \right]$$

The denoising objectives

x -prediction; MLE

$$\frac{1}{2\sigma_q^2(t)} \frac{\bar{\alpha}_{t-1} (1 - \alpha_t)^2}{(1 - \bar{\alpha}_t)^2} \left[\| \hat{x}_\theta(z_t, t) - x \|^2 \right]$$

The denoising objectives

x -prediction; MLE

$$\frac{1}{2\sigma_q^2(t)} \frac{\bar{\alpha}_{t-1} (1 - \alpha_t)^2}{(1 - \bar{\alpha}_t)^2} \left[\| \hat{x}_\theta(z_t, t) - x \|^2 \right]$$

ϵ -prediction; MLE

$$\frac{1}{2\sigma_q^2(t)} \frac{(1 - \alpha_t)^2}{(1 - \bar{\alpha}_t) \alpha_t} \left[\| \epsilon - \hat{\epsilon}_\theta(x_t, t) \|^2 \right]$$

The denoising objectives

x -prediction; MLE

$$\frac{1}{2\sigma_q^2(t)} \frac{\bar{\alpha}_{t-1} (1 - \alpha_t)^2}{(1 - \bar{\alpha}_t)^2} \left[\| \hat{x}_\theta(z_t, t) - x \|^2 \right]$$

ϵ -prediction; MLE

$$\frac{1}{2\sigma_q^2(t)} \frac{(1 - \alpha_t)^2}{(1 - \bar{\alpha}_t) \alpha_t} \left[\| \epsilon - \hat{\epsilon}_\theta(x_t, t) \|^2 \right]$$

ϵ -prediction; “simple”

$$\| \epsilon - \hat{\epsilon}_\theta(x_t, t) \|^2$$

Typical objective for training
image diffusion models:
SOTA on many tasks!

The denoising objectives

x -prediction; MLE

$$\frac{1}{2\sigma_q^2(t)} \frac{\bar{\alpha}_{t-1} (1 - \alpha_t)^2}{(1 - \bar{\alpha}_t)^2} \left[\| \hat{x}_\theta(z_t, t) - x \|^2 \right]$$

ϵ -prediction; MLE

$$\frac{1}{2\sigma_q^2(t)} \frac{(1 - \alpha_t)^2}{(1 - \bar{\alpha}_t) \alpha_t} \left[\| \epsilon - \hat{\epsilon}_\theta(x_t, t) \|^2 \right]$$



ϵ -prediction; “simple”

$$\| \epsilon - \hat{\epsilon}_\theta(x_t, t) \|^2$$

Typical objective for training
image diffusion models:
SOTA on many tasks!

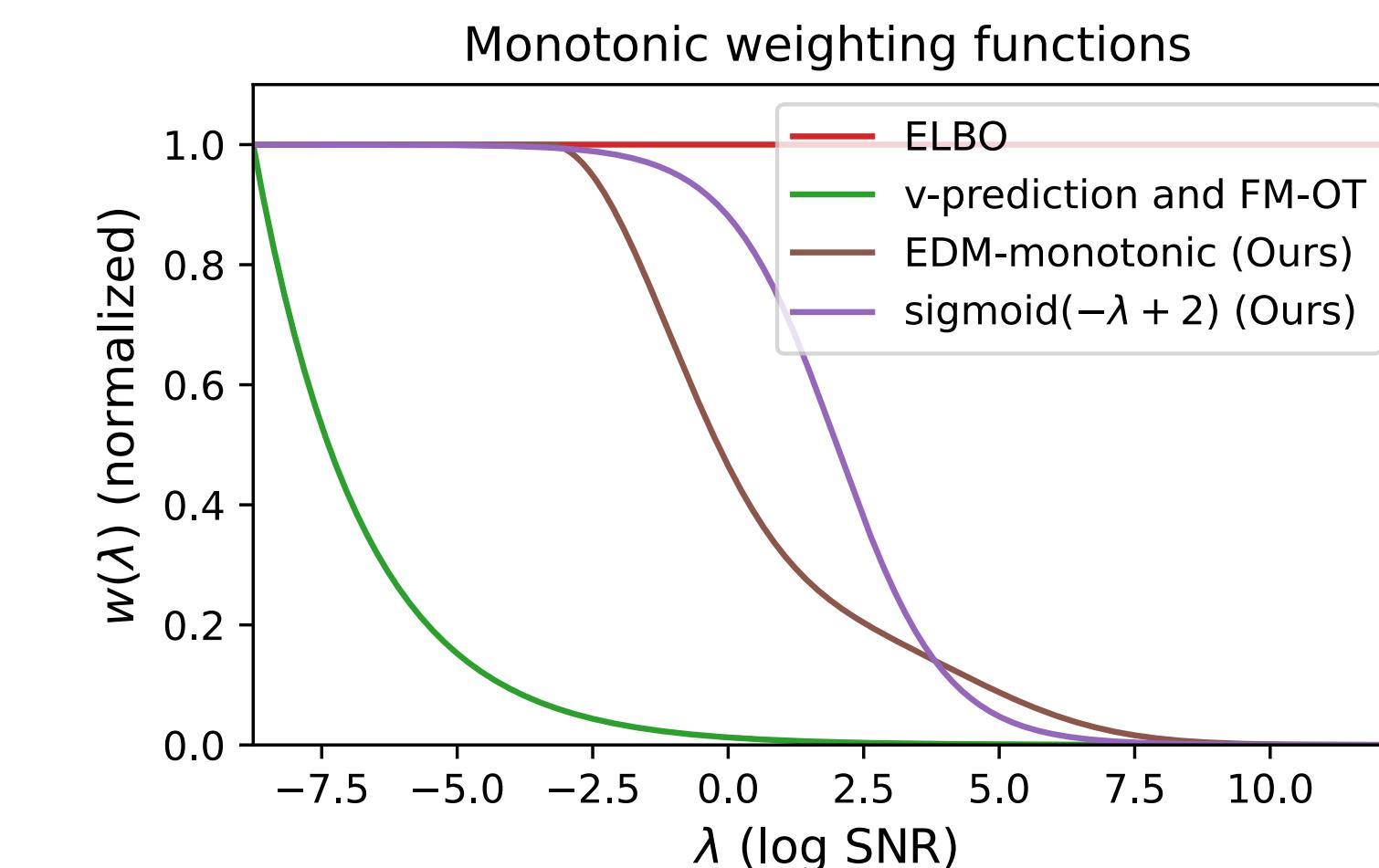
Simple objectives as a weighted sum of ELBOs

Kingma and Gao (2023) showed that common objectives can be written as a weighted sum (across different noise levels) of ELBOs

$$L_w(x) = \left\langle w(t) \cdot w_{\text{ML}}(t) \left\| \epsilon - \hat{\epsilon}_\theta(z_t, t) \right\|^2 \right\rangle_{t, \epsilon}$$

Additional weighting
(w_{ML}^{-1} for ϵ -prediction)

Weighting for ELBO/
ML objective



Simple objectives as a weighted sum of ELBOs

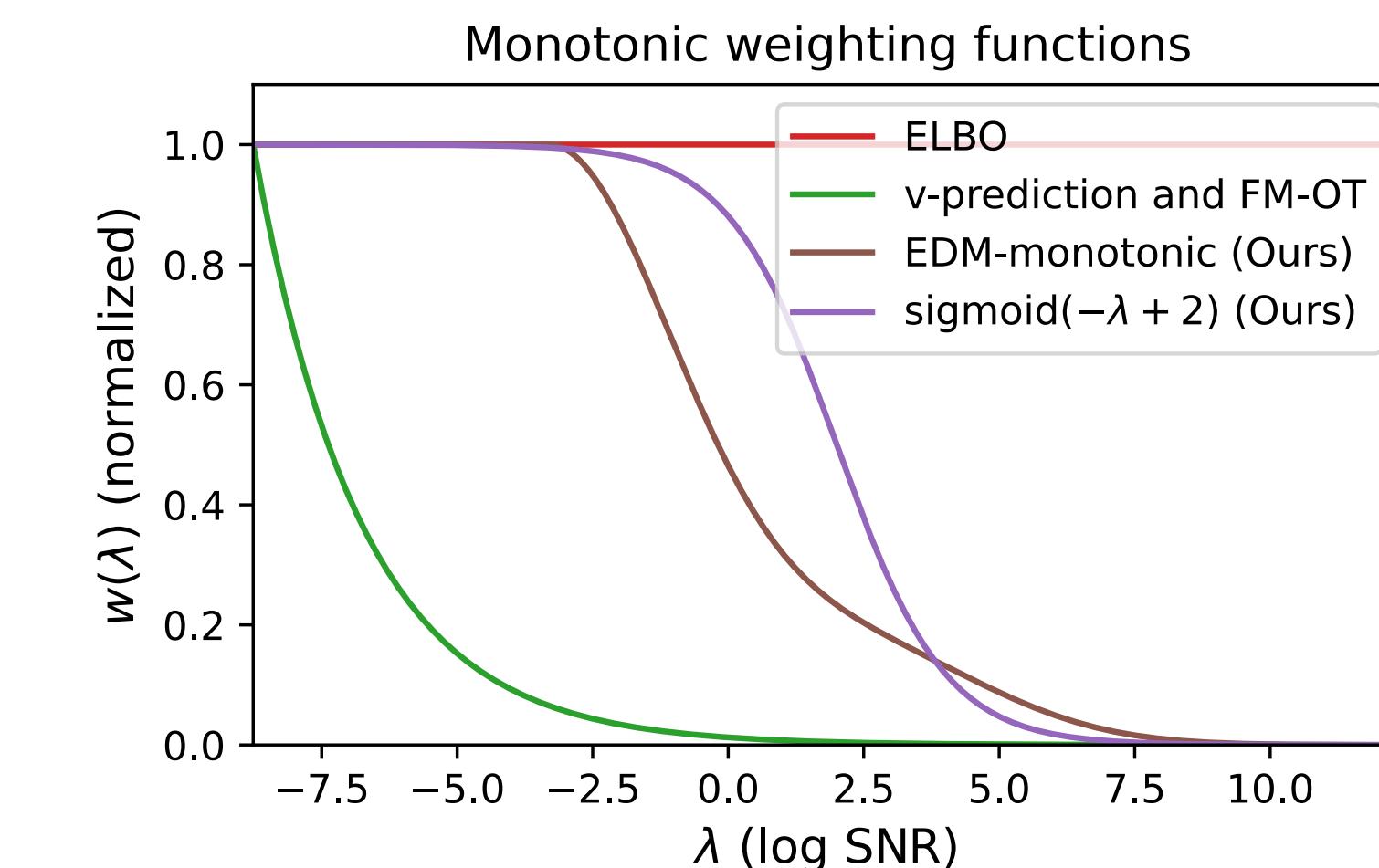
Kingma and Gao (2023) showed that common objectives can be written as a weighted sum (across different noise levels) of ELBOs

$$L_w(x) = \left\langle w(t) \cdot w_{\text{ML}}(t) \left\| \epsilon - \hat{\epsilon}_\theta(z_t, t) \right\|^2 \right\rangle_{t, \epsilon}$$

Additional weighting
(w_{ML}^{-1} for ϵ -prediction)

Weighting for ELBO/
ML objective

$$L(t; x) \equiv D_{\text{KL}} \left(q(z_{t:1} | x) \| p(z_{t:1}) \right)$$



Simple objectives as a weighted sum of ELBOs

Kingma and Gao (2023) showed that common objectives can be written as a weighted sum (across different noise levels) of ELBOs

$$L_w(x) = \left\langle w(t) \cdot w_{\text{ML}}(t) \left\| \epsilon - \hat{\epsilon}_\theta(z_t, t) \right\|^2 \right\rangle_{t, \epsilon}$$

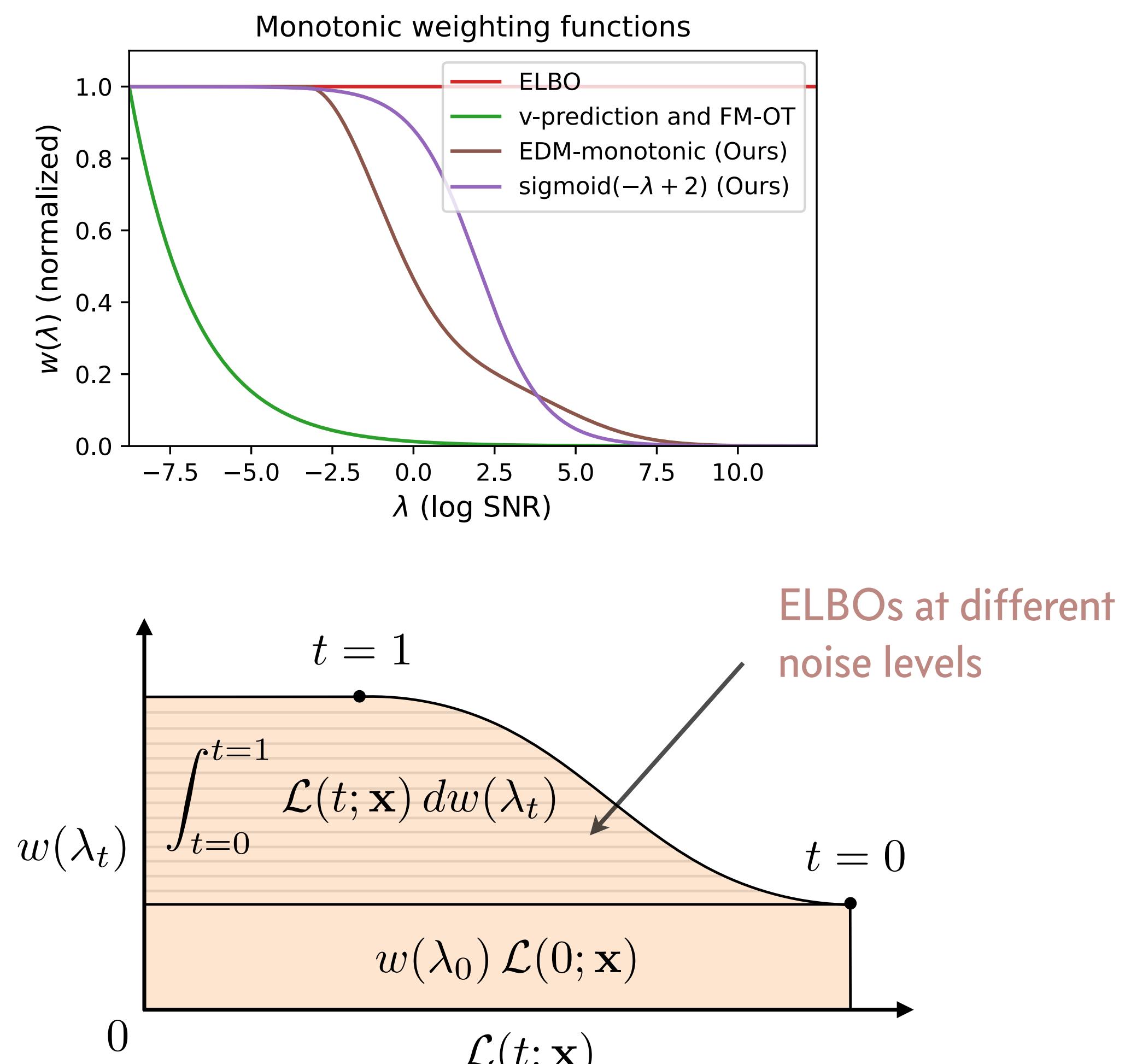
Additional weighting
(w_{ML}^{-1} for ϵ -prediction)

Weighting for ELBO/
ML objective

$$L(t; x) \equiv D_{\text{KL}} \left(q(z_{t:1} | x) \| p(z_{t:1}) \right)$$

$$L_w(x) \propto \int_0^1 \frac{d}{dt} w(t) L(t; x) dt + w(t_0) L(0; x)$$

Interpretation: *data augmentation with additive Gaussian noise / data-distribution smoothing*



Simple objectives as a weighted sum of ELBOs

Kingma and Gao (2023) showed that common objectives can be written as a weighted sum (across different noise levels) of ELBOs

$$L_w(x) = \left\langle w(t) \cdot w_{\text{ML}}(t) \left\| \epsilon - \hat{\epsilon}_\theta(z_t, t) \right\|^2 \right\rangle_{t, \epsilon}$$

Additional weighting
(w_{ML}^{-1} for ϵ -prediction)

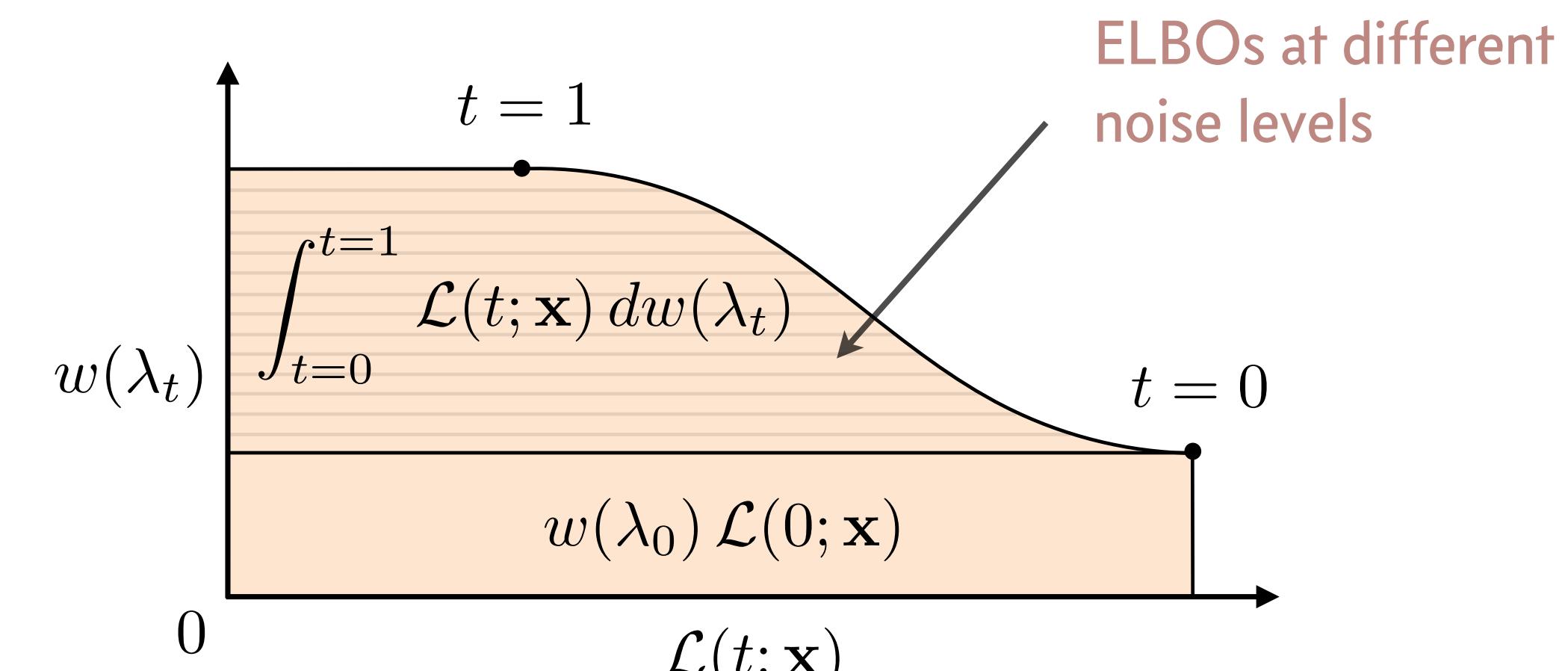
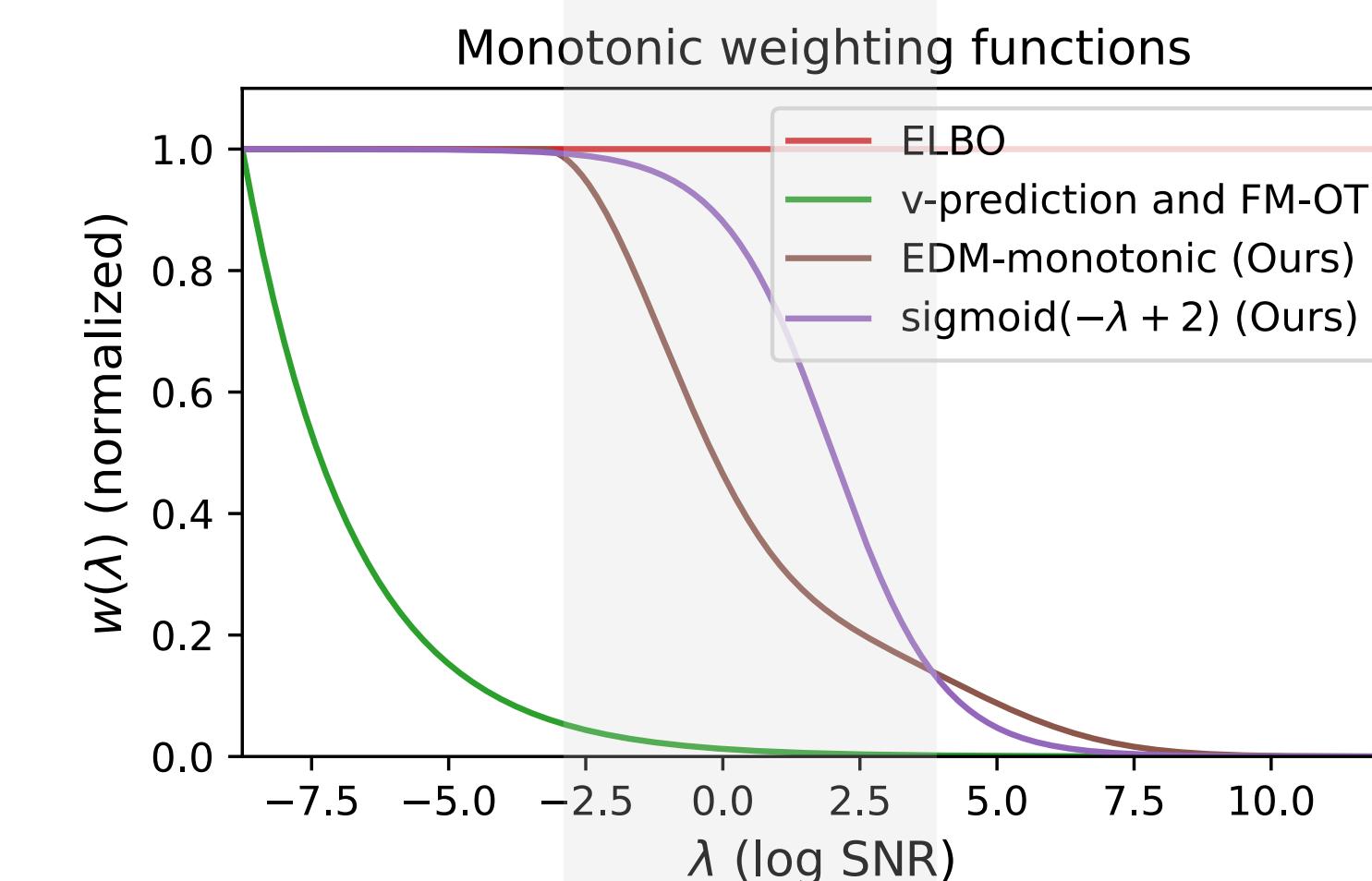
Weighting for ELBO/
ML objective

$$L(t; x) \equiv D_{\text{KL}} \left(q(z_{t:1} | x) \| p(z_{t:1}) \right)$$

$$L_w(x) \propto \int_0^1 \frac{d}{dt} w(t) L(t; x) dt + w(t_0) L(0; x)$$

Interpretation: *data augmentation with additive Gaussian noise / data-distribution smoothing*

More important
for perceptual
quality?



Continuous-time/SDE formulation

$$x_t = \sqrt{1 - \beta_t} \cdot x_{t-1} + \sqrt{\beta_t} \cdot \varepsilon$$

In the limit of infinite time steps, $\Delta_t \rightarrow 0$ and the forward diffusion process can be written as

$$\begin{aligned} x_t &= \sqrt{1 - \beta(t)\Delta_t} x_{t-1} + \sqrt{\beta(t)\Delta_t} \mathcal{N}(0, \mathbb{I}) \\ &\approx x_{t-1} - \frac{\beta(t)\Delta_t}{2} x_{t-1} + \sqrt{\beta(t)\Delta_t} \mathcal{N}(0, \mathbb{I}) \end{aligned}$$

Continuous-time/SDE formulation

$$x_t = \sqrt{1 - \beta_t} \cdot x_{t-1} + \sqrt{\beta_t} \cdot \varepsilon$$

In the limit of infinite time steps, $\Delta_t \rightarrow 0$ and the forward diffusion process can be written as

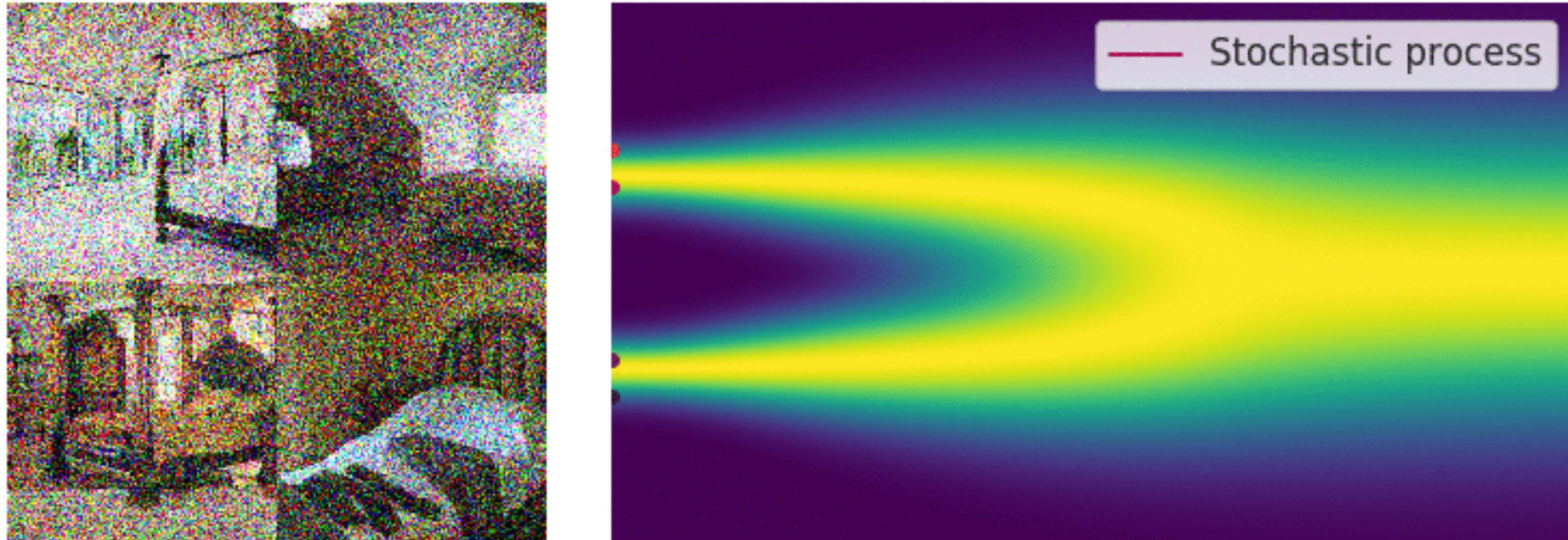
$$\begin{aligned} x_t &= \sqrt{1 - \beta(t)\Delta_t} x_{t-1} + \sqrt{\beta(t)\Delta_t} \mathcal{N}(0, \mathbb{I}) \\ &\approx x_{t-1} - \frac{\beta(t)\Delta_t}{2} x_{t-1} + \sqrt{\beta(t)\Delta_t} \mathcal{N}(0, \mathbb{I}) \end{aligned}$$

Which is an update rule corresponding to the Euler-Murayama discretization of the stochastic differential equation (SDE)

$$dx_t = -\frac{1}{2}\beta(t)x_t dt + \sqrt{\beta(t)}dw_t$$

Continuous-time/SDE formulation

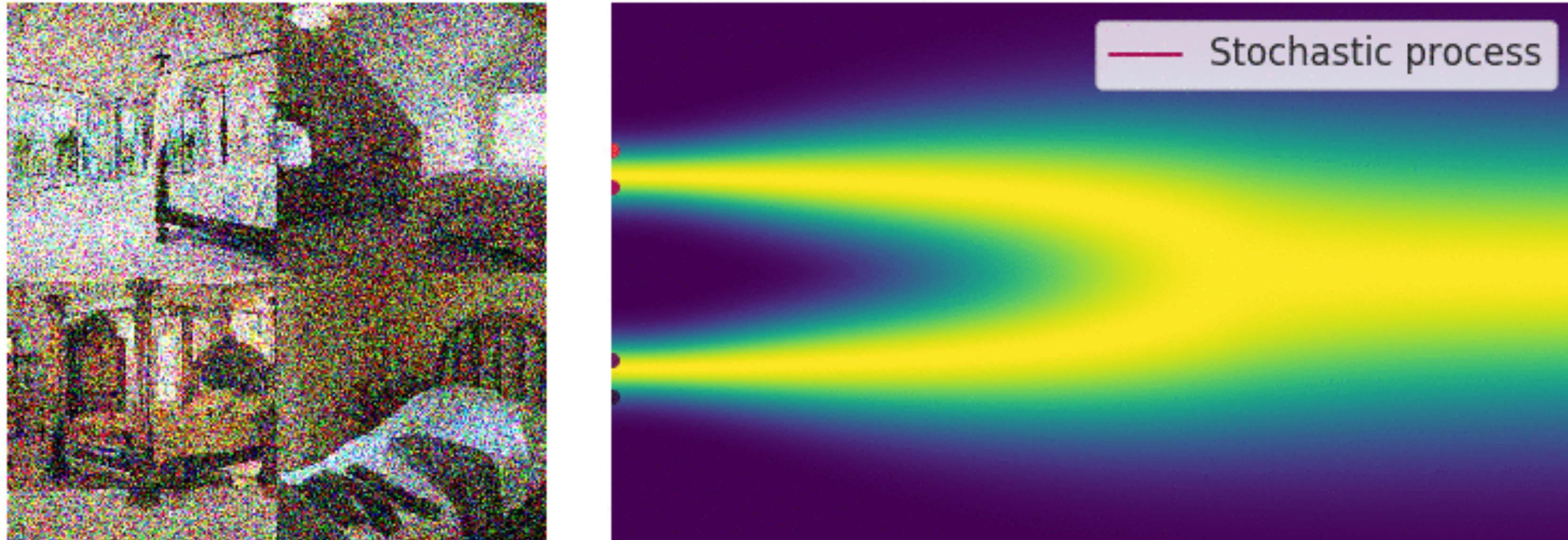
The forward diffusion process defined by an SDE



$$dx_t = -\frac{1}{2}\beta(t)x_t \, dt + \sqrt{\beta(t)}dw_t$$

Continuous-time/SDE formulation

The forward diffusion process defined by an SDE

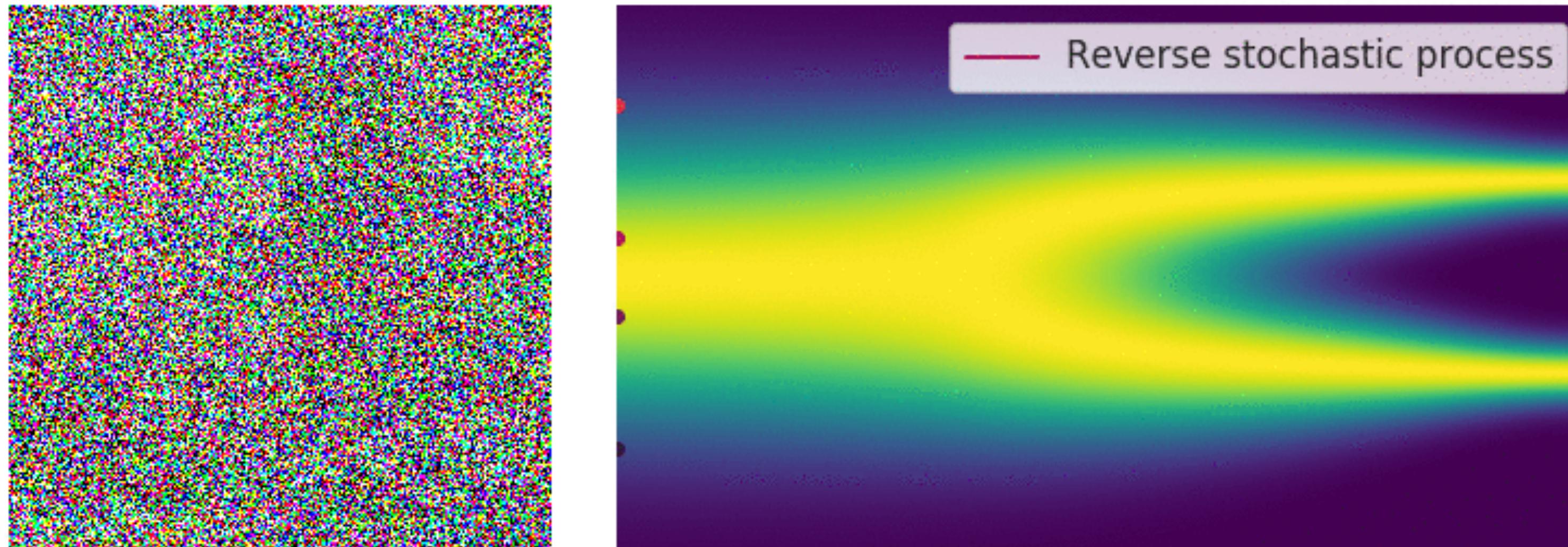


$$dx_t = -\frac{1}{2}\beta(t)x_t \, dt + \sqrt{\beta(t)}dw_t$$

The reverse SDE

The reverse process satisfies a reverse-time SDE that can be derived from the forward SDE and the score of the marginal distribution, $\nabla_{x_t} \log q(x_t)$

[Song et al 2021]

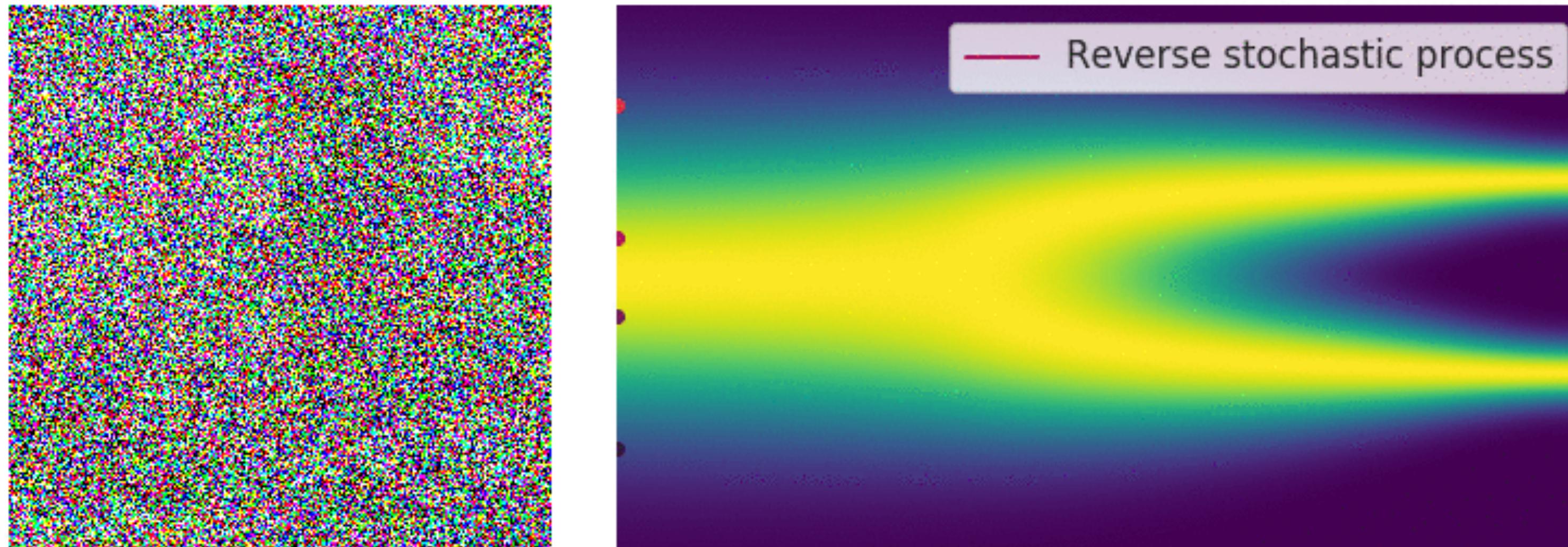


$$dx_t = \left[-\frac{1}{2}\beta(t)x_t - \beta(t) \nabla_{x_t} \log q(x_t) \right] dt + \sqrt{\beta(t)} dw_t$$

The reverse SDE

The reverse process satisfies a reverse-time SDE that can be derived from the forward SDE and the score of the marginal distribution, $\nabla_{x_t} \log q(x_t)$

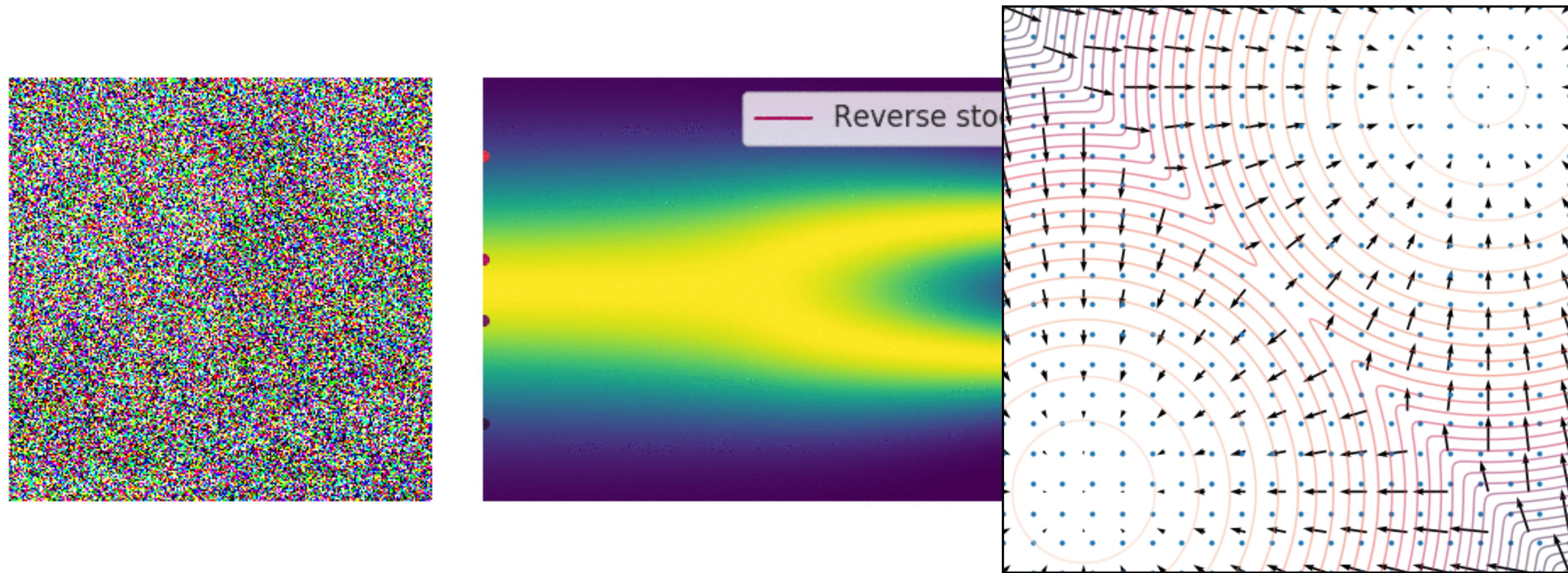
[Song et al 2021]



$$dx_t = \left[-\frac{1}{2}\beta(t)x_t - \beta(t) \nabla_{x_t} \log q(x_t) \right] dt + \sqrt{\beta(t)} dw_t$$

The reverse SDE

The reverse process satisfies a reverse-time SDE that can be derived from the forward SDE and the score of the marginal distribution, $\nabla_{x_t} \log q(x_t)$



$$dx_t = \left[-\frac{1}{2}\beta(t)x_t - \beta(t) \nabla_{x_t} \log q(x_t) \right] dt + \sqrt{\beta(t)} dw_t$$

Denoising score matching

Need to compute the score $\nabla_{x_t} \log q(x_t)$

The *conditional* score $\nabla_{x_t} \log q(x_t \mid x)$ can be computed using the diffusion kernel

$$\nabla_{x_t} \log q(x_t \mid x) = -\frac{(x_t - x)}{\sigma_t^2} = -\frac{\epsilon}{\sigma_t}$$

Denoising score matching

Need to compute the score $\nabla_{x_t} \log q(x_t)$

The *conditional* score $\nabla_{x_t} \log q(x_t | x)$ can be computed using the diffusion kernel

$$\nabla_{x_t} \log q(x_t | x) = - \frac{(x_t - x)}{\sigma_t^2} = - \frac{\epsilon}{\sigma_t}$$

Noise-prediction

$$\frac{1}{2\sigma_q^2(t)} \frac{(1 - \alpha_t)^2}{(1 - \bar{\alpha}_t) \alpha_t} \left[\left\| \epsilon - \hat{\epsilon}_\theta(x_t, t) \right\|^2 \right] \quad \leftrightarrow$$

Score-matching

$$\frac{1}{2\sigma_q^2(t)} \frac{(1 - \alpha_t)^2}{\alpha_t} \left[\left\| s_\theta(x_t, t) - \nabla \log q(x_t) \right\|^2 \right]$$

The noise- and score-prediction networks are equivalent up to a std-scaling

Denoising score matching

* x conditioning disappears when taking expectation wrt x to give marginal score

Need to compute the score $\nabla_{x_t} \log q(x_t)$

The *conditional* score $\nabla_{x_t} \log q(x_t | x)$ can be computed using the diffusion kernel

$$\nabla_{x_t} \log q(x_t | x) = - \frac{(x_t - x)}{\sigma_t^2} = - \frac{\epsilon}{\sigma_t}$$

Noise-prediction

$$\frac{1}{2\sigma_q^2(t)} \frac{(1 - \alpha_t)^2}{(1 - \bar{\alpha}_t) \alpha_t} \left[\left\| \epsilon - \hat{\epsilon}_\theta(x_t, t) \right\|^2 \right]$$



Score-matching

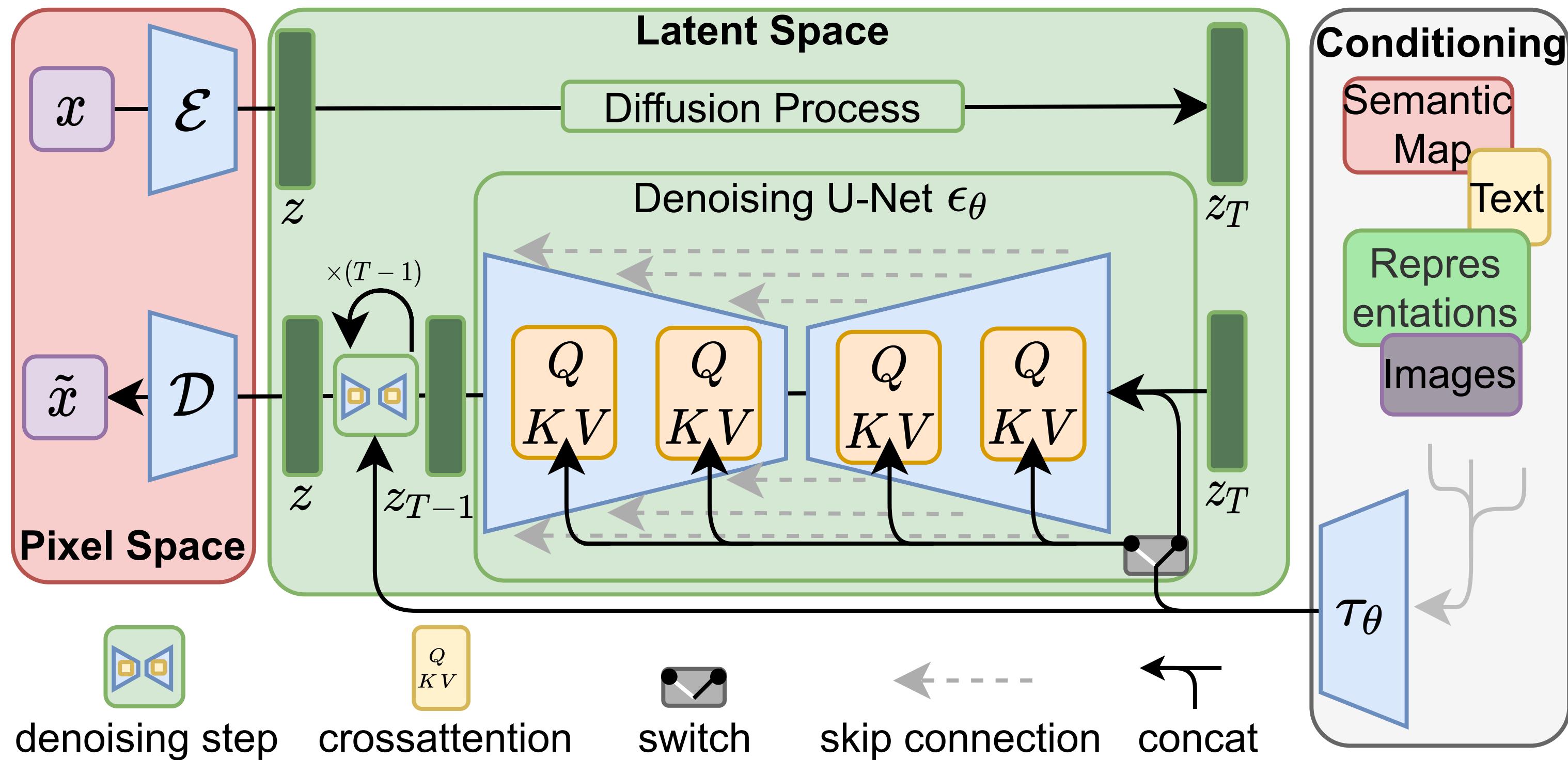
$$\frac{1}{2\sigma_q^2(t)} \frac{(1 - \alpha_t)^2}{\alpha_t} \left[\left\| s_\theta(x_t, t) - \nabla \log q(x_t) \right\|^2 \right]$$

*

The noise- and score-prediction networks are equivalent up to a std-scaling

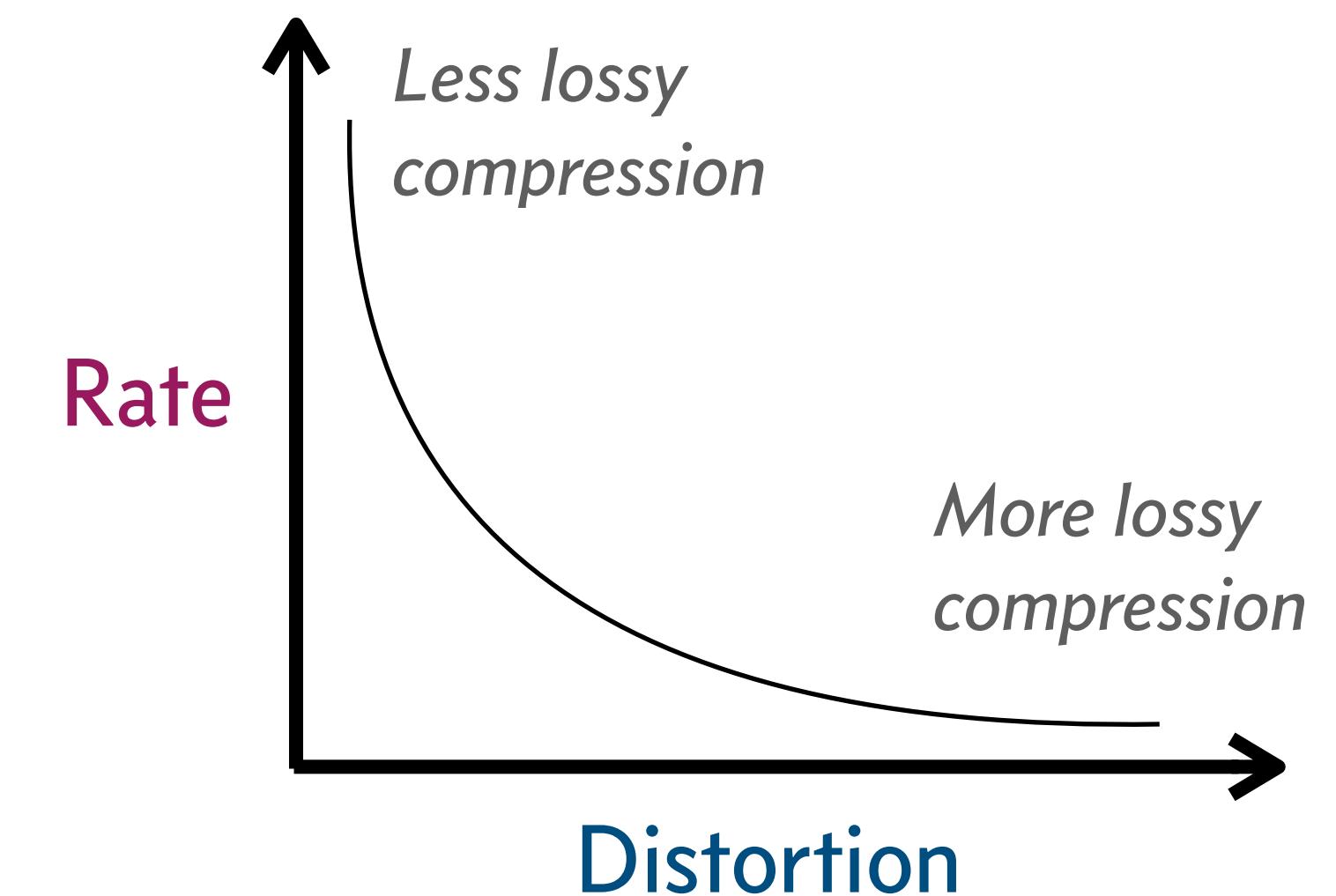
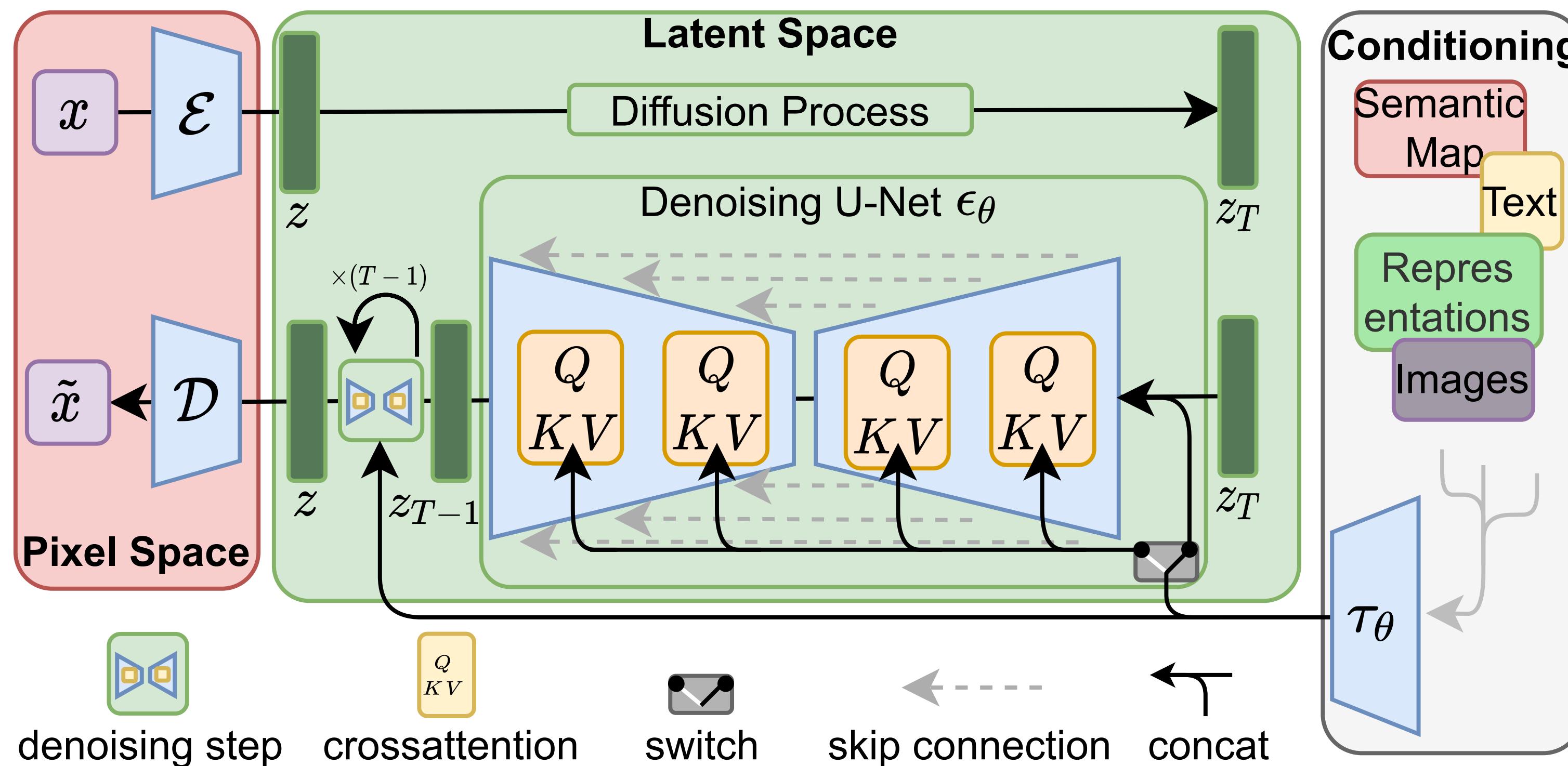
The noise/score-prediction model and *latent diffusion*

[Rombach et al 2021]



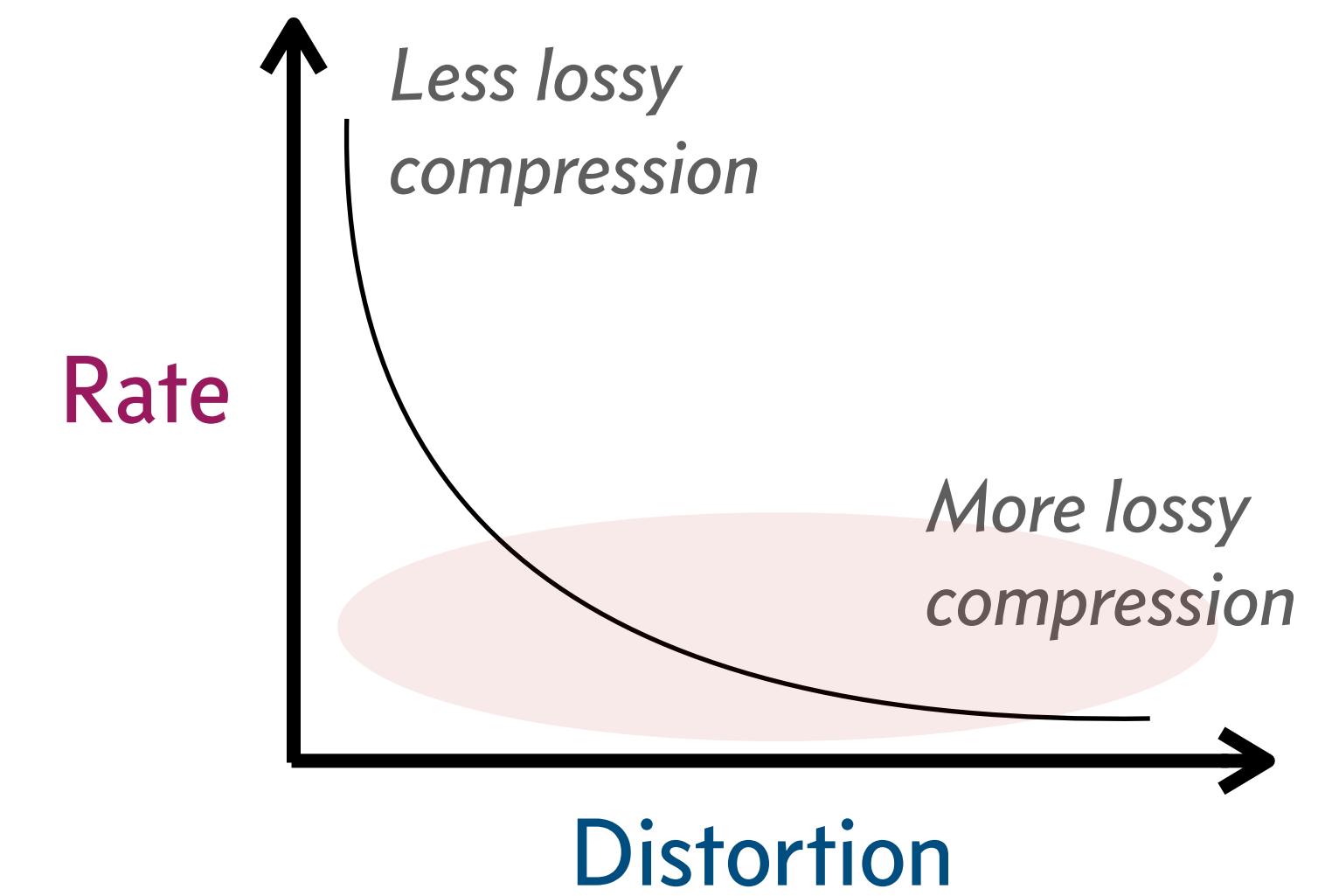
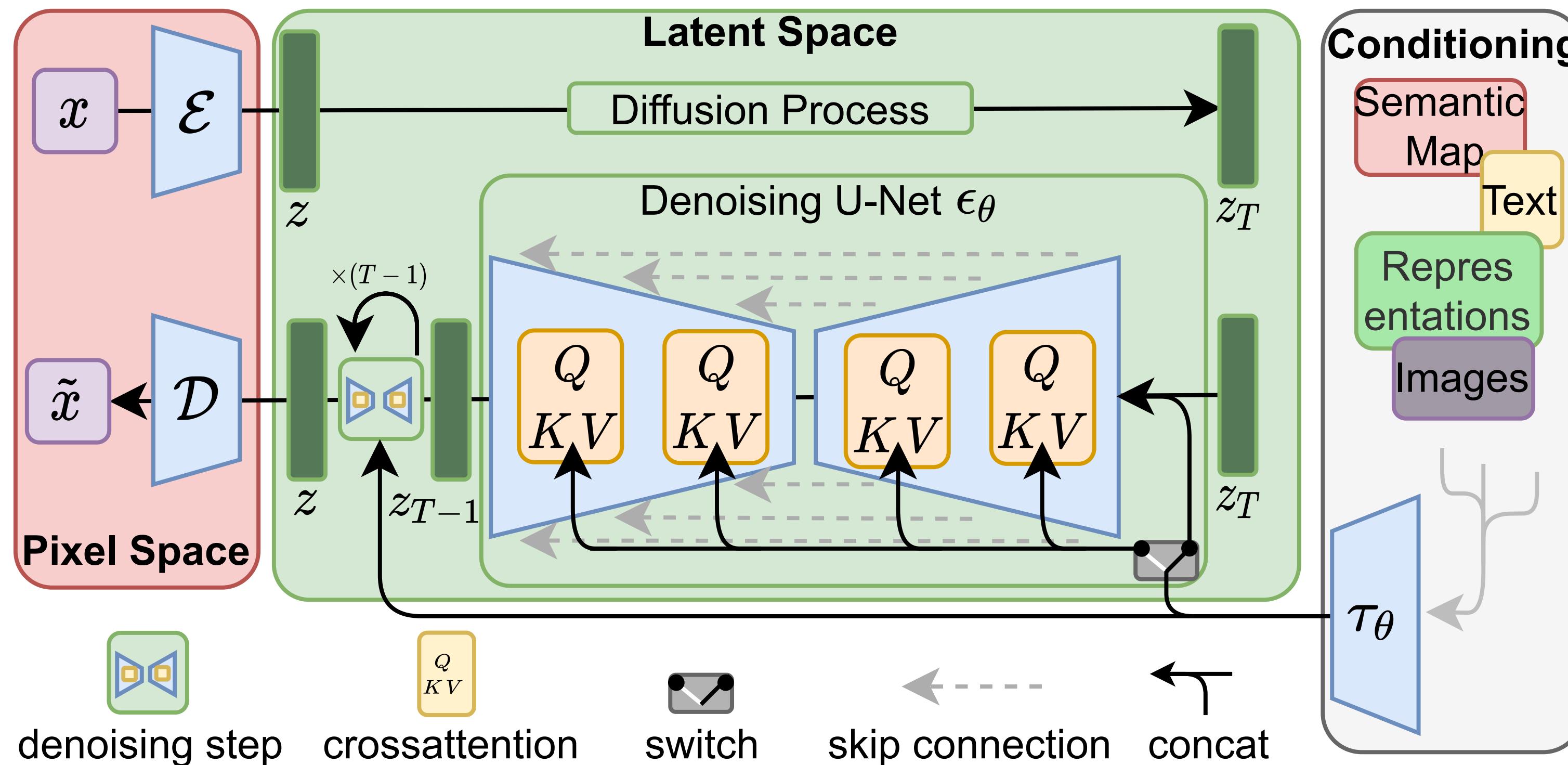
The noise/score-prediction model and *latent diffusion*

[Rombach et al 2021]



The noise/score-prediction model and *latent diffusion*

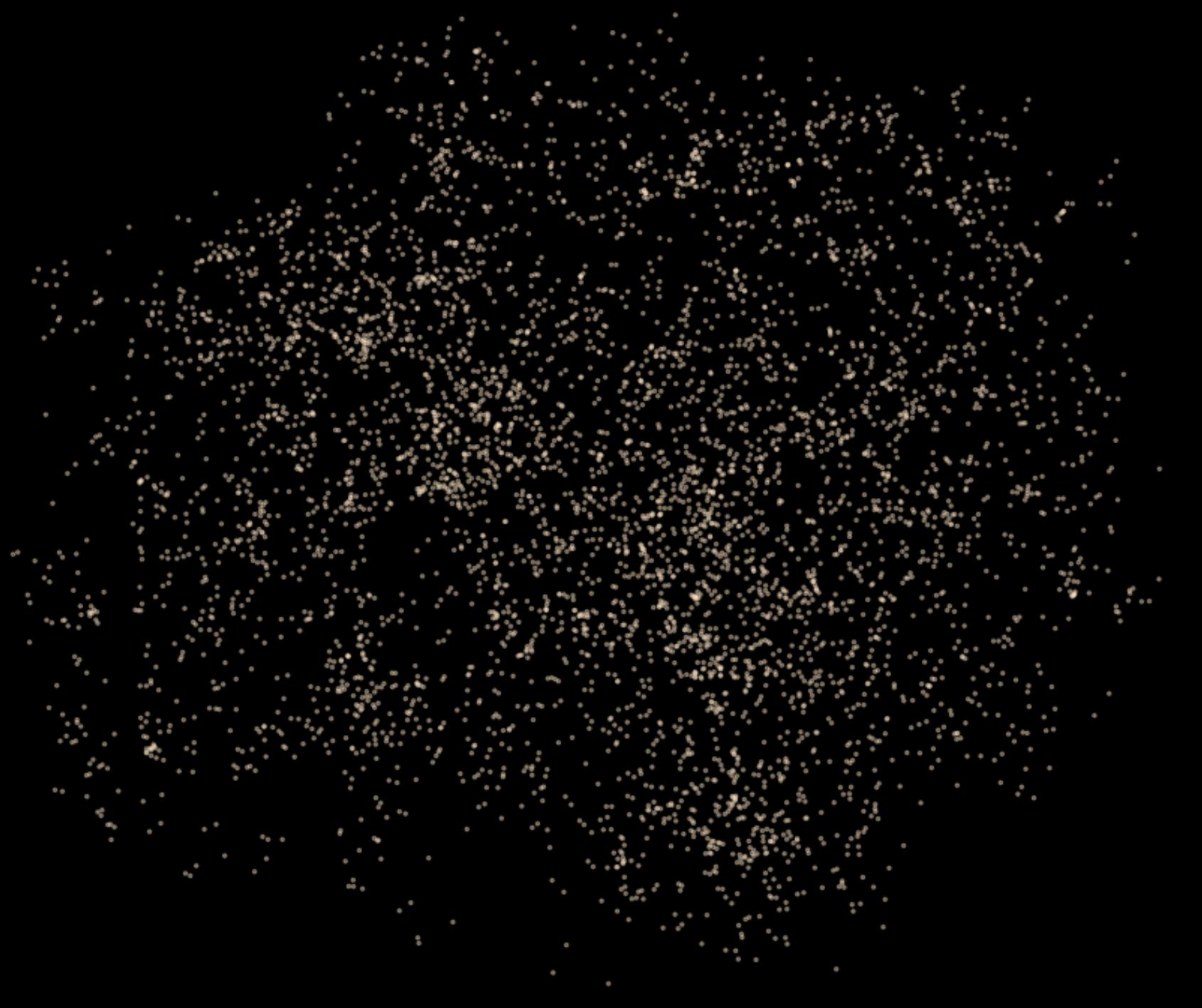
[Rombach et al 2021]



Perceptual compression while retaining semantically meaningful information

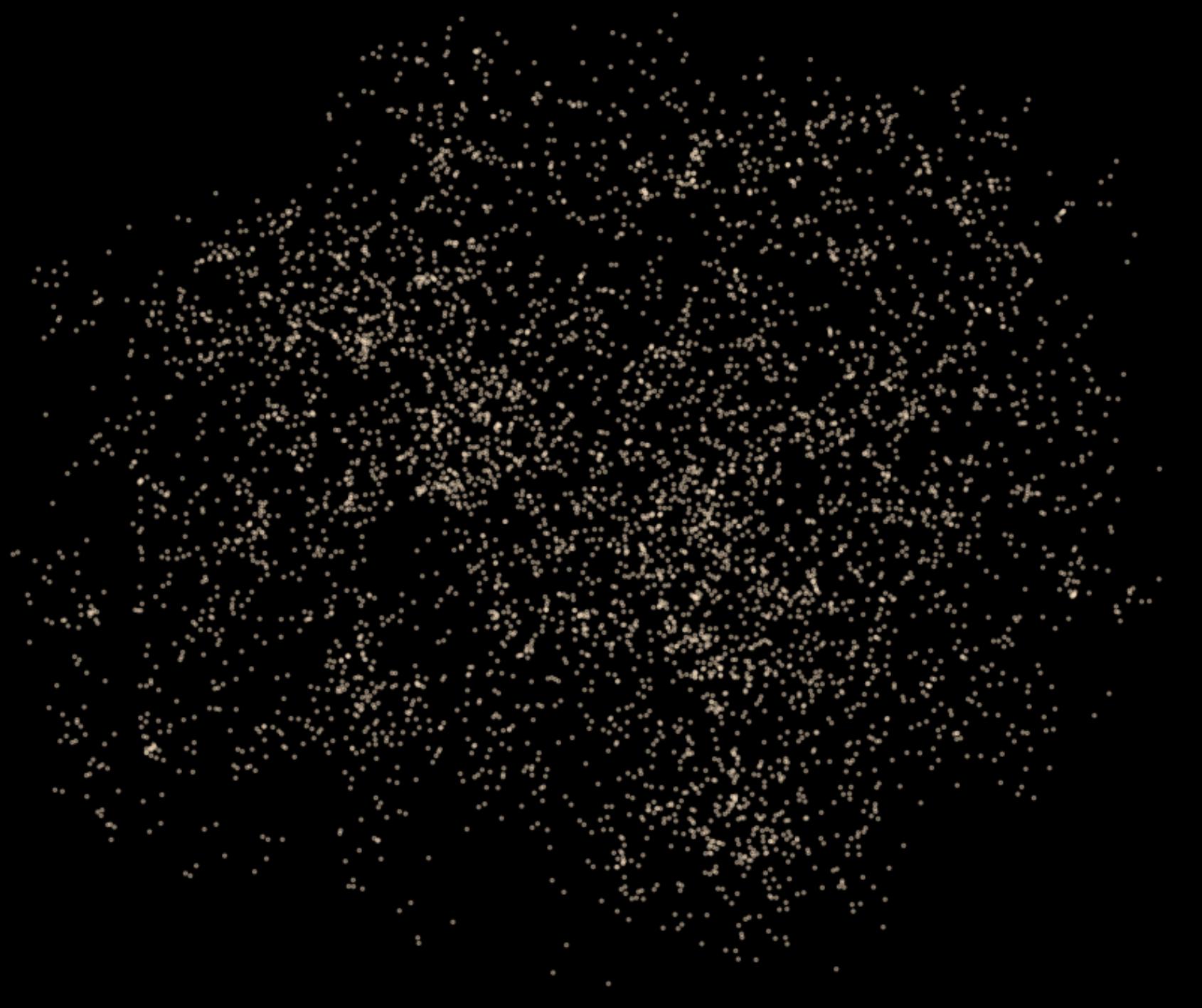
$$\Omega_m = 0.10$$

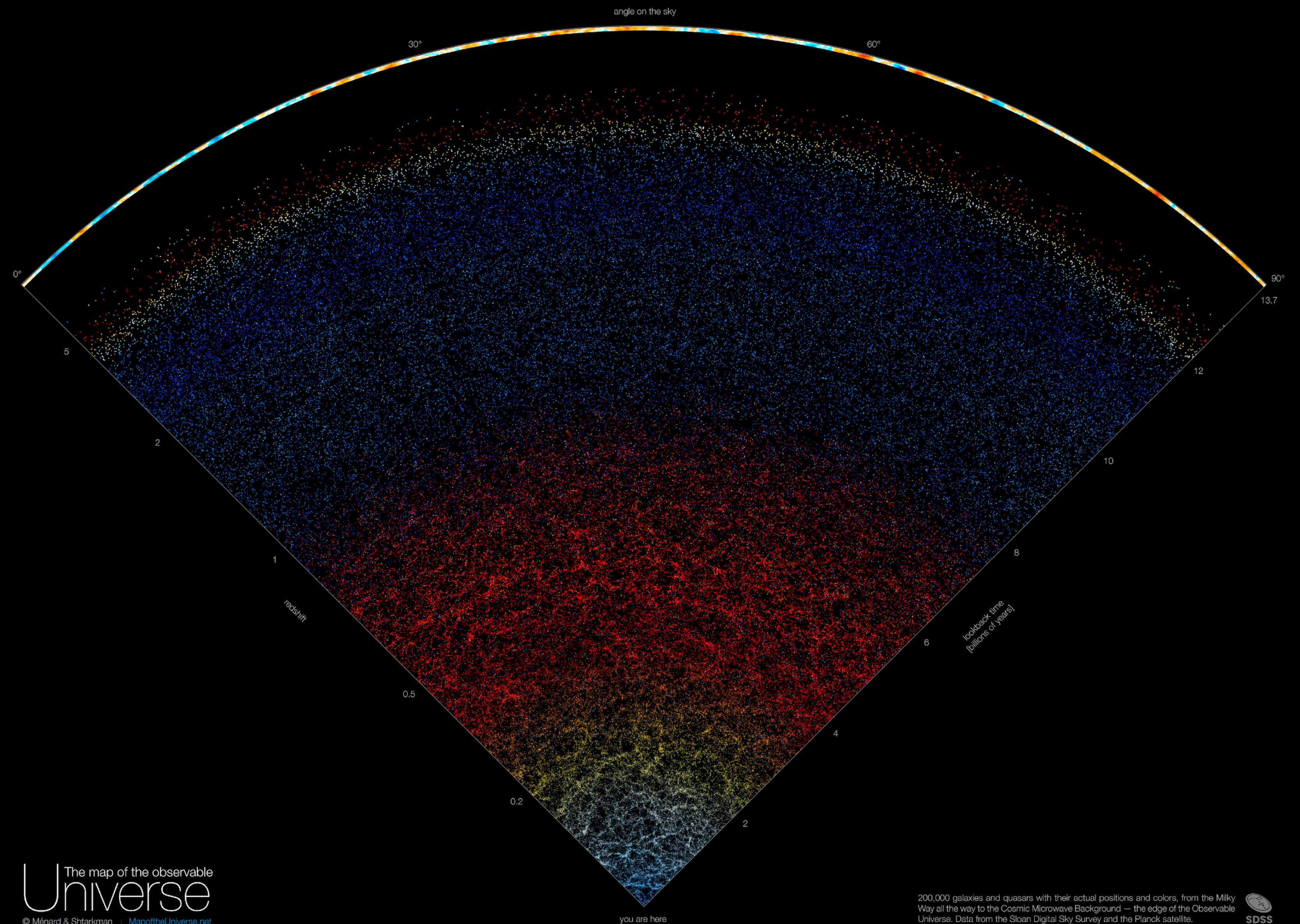
An application to *galaxy clustering*



$$\Omega_m = 0.10$$

An application to *galaxy clustering*

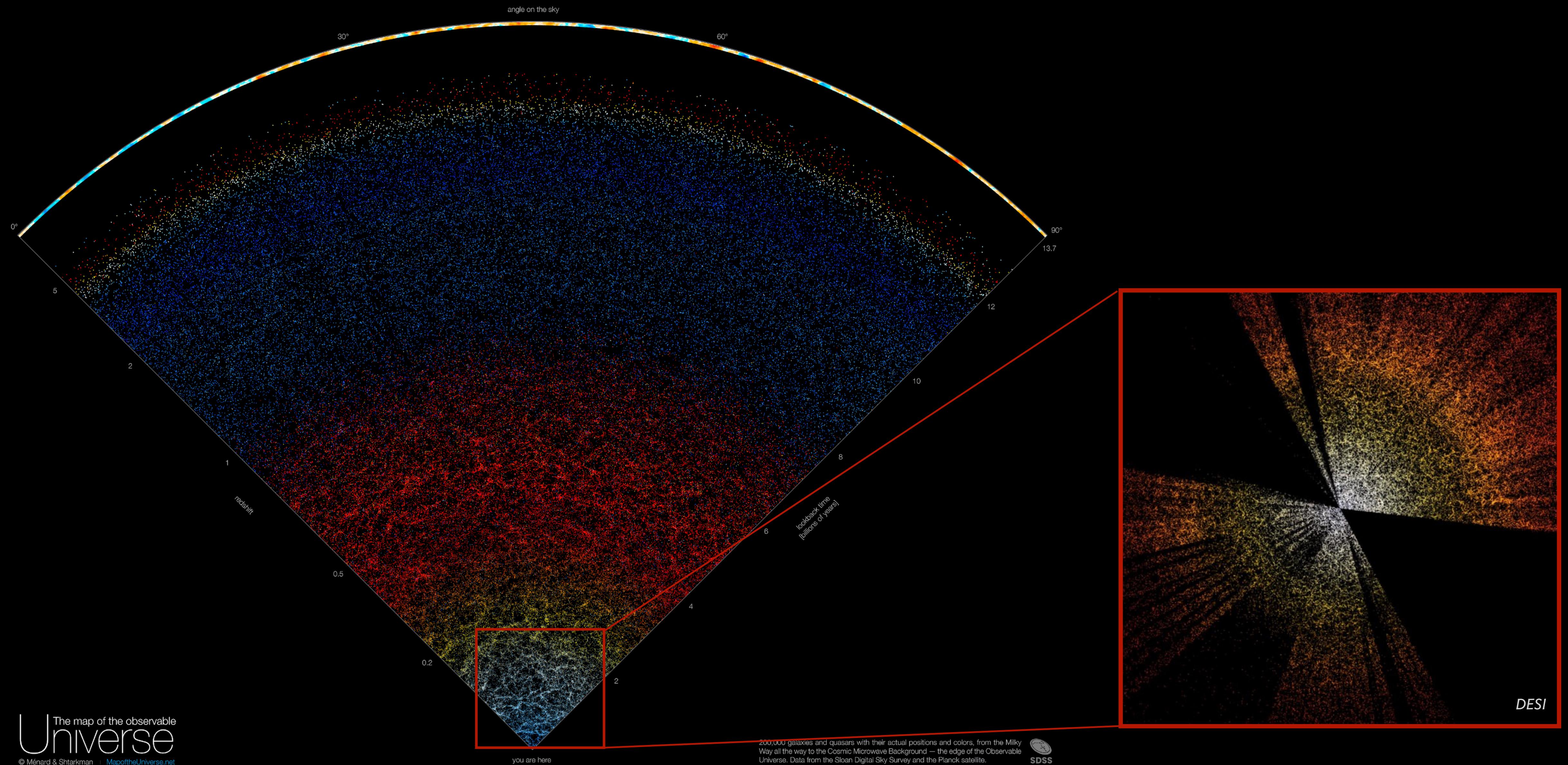




The map of the observable
Universe
© Menard & Shtarkman | MapoftheUniverse.net

200,000 galaxies and quasars with their actual positions and colors, from the Milky Way all the way to the Cosmic Microwave Background — the edge of the Observable Universe. Data from the Sloan Digital Sky Survey and the Planck satellite.

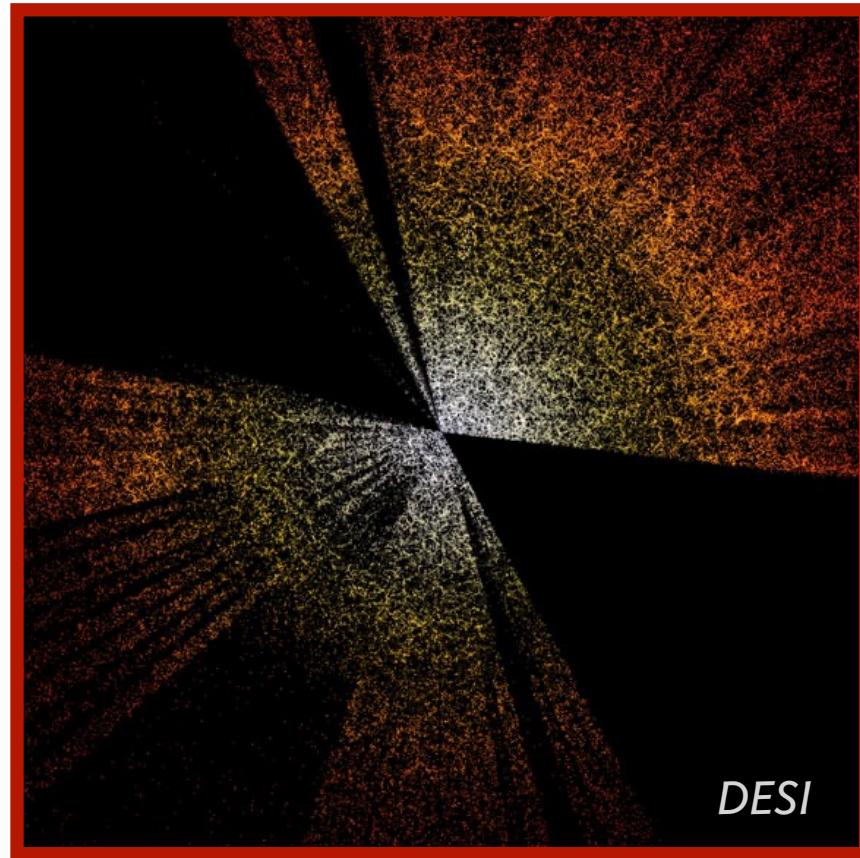




The likelihood of a galaxy field

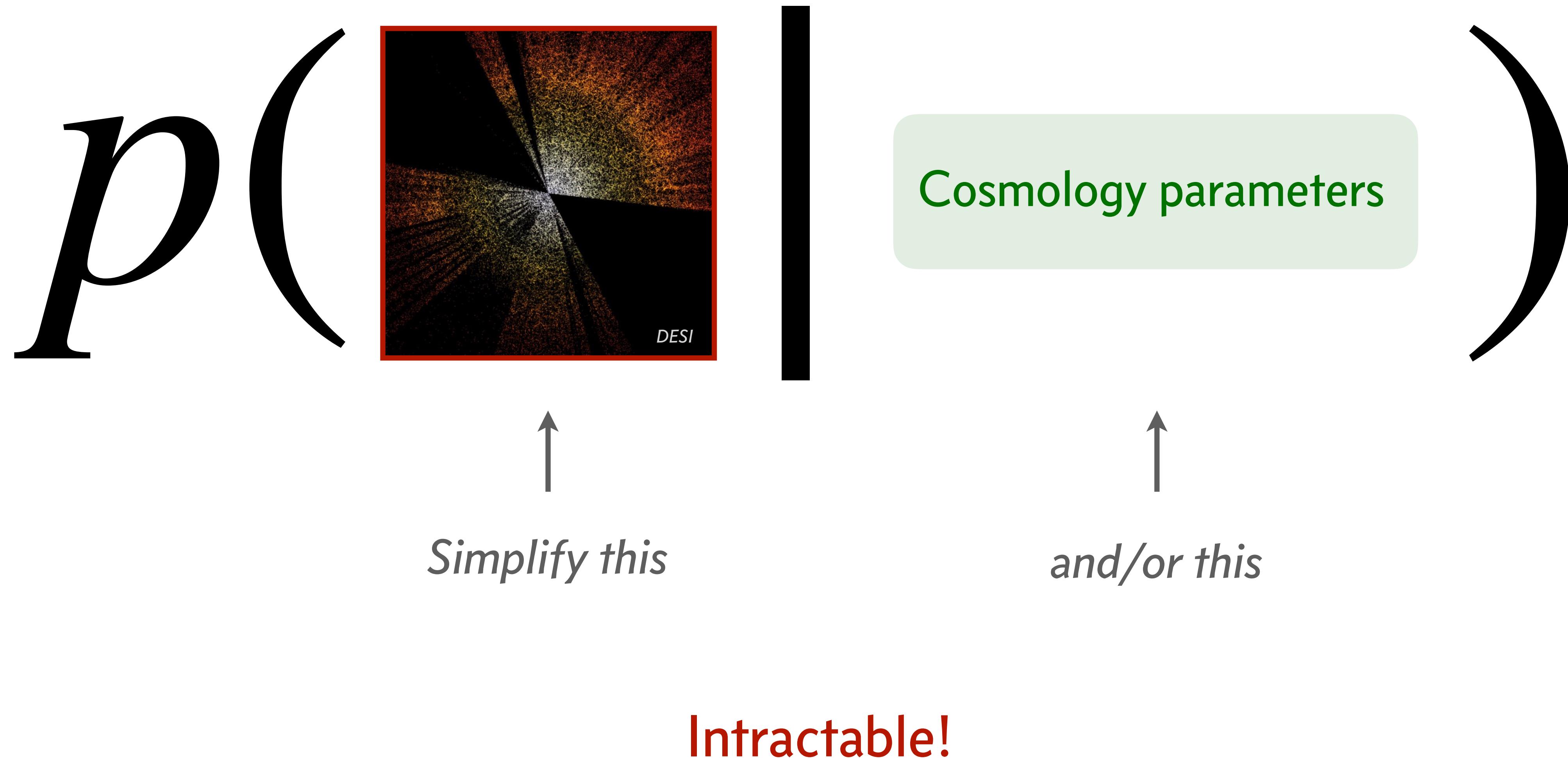
$$p(x | \theta)$$

$$p(\text{Cosmology parameters} | \text{Galaxy field})$$

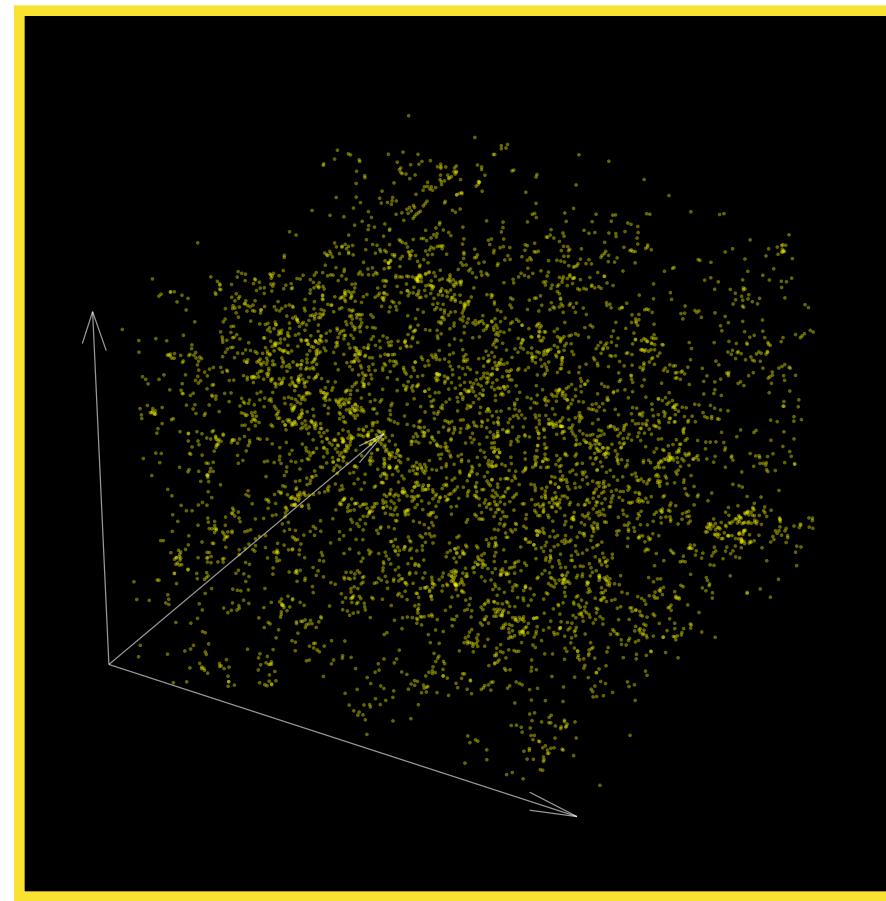


The likelihood of a galaxy field

$$p(x | \theta)$$



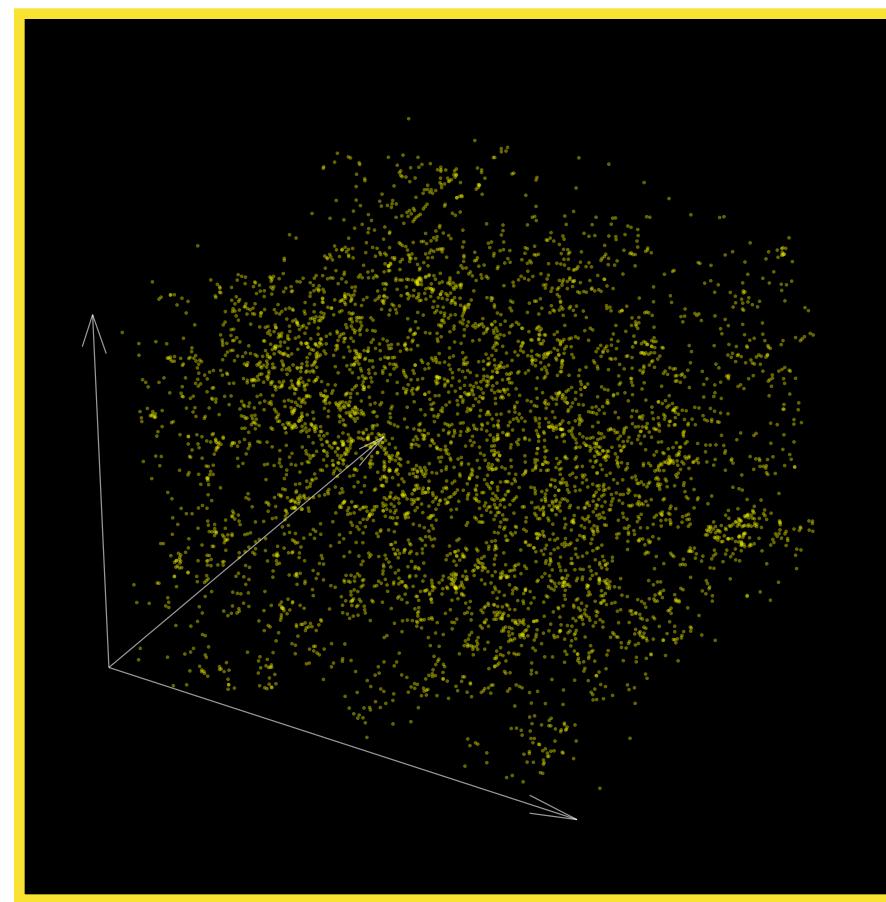
Emulation and inference



$$\sim p(\text{Cosmology} \mid \text{Emulation/sampling})$$

*Emulation/
sampling*

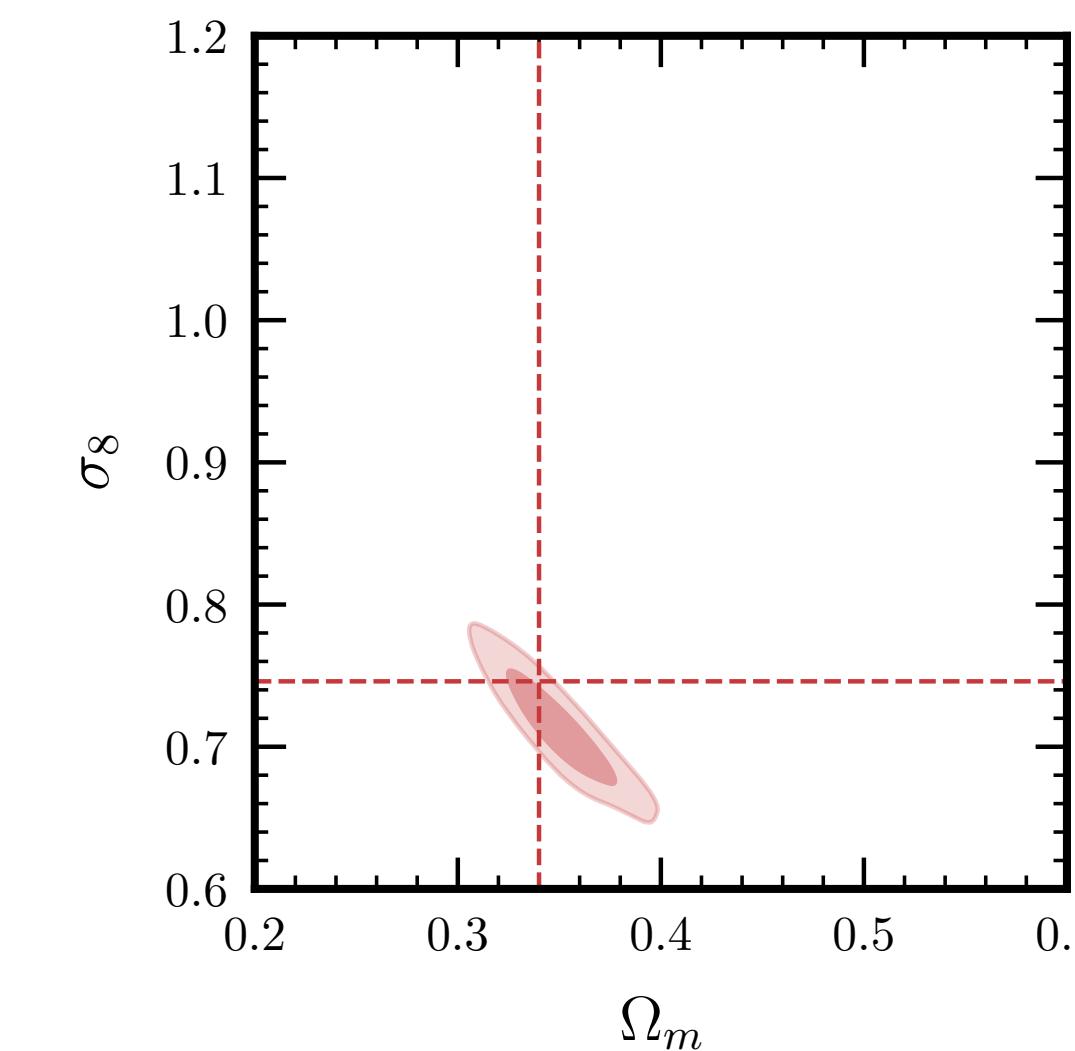
Emulation and inference



$$\sim p(\text{Cosmology} \mid \text{Cosmology})$$

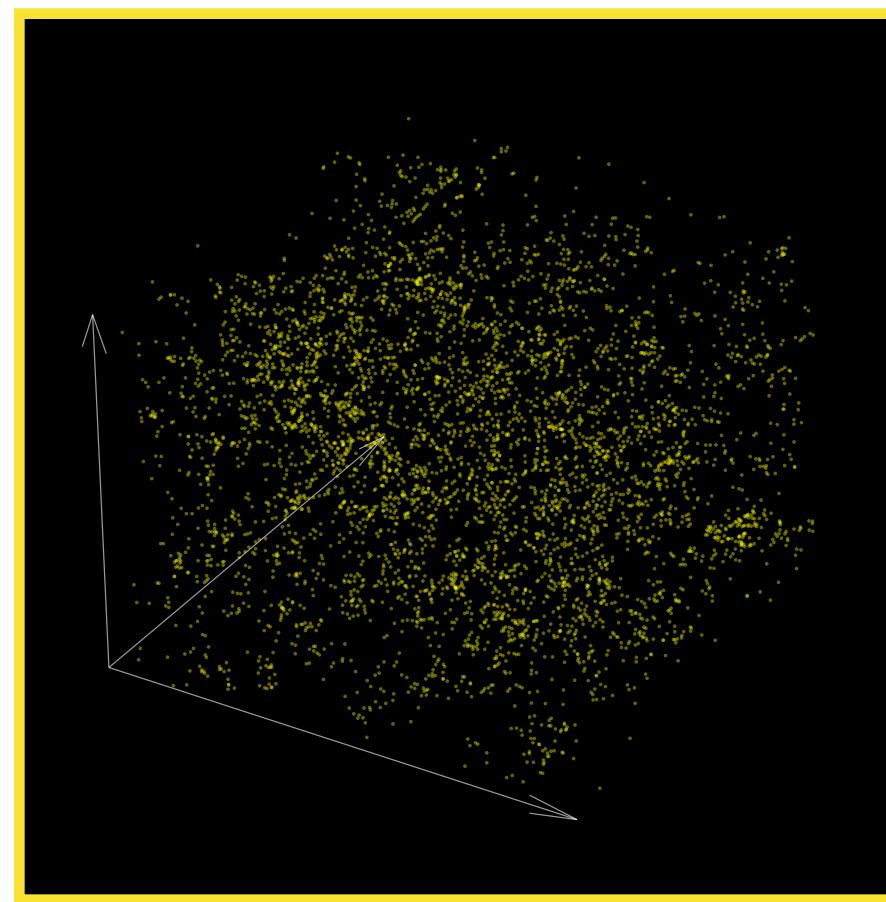
*Emulation/
sampling*

$$p(\text{Cosmology} \mid \Omega_m, \sigma_8)$$



*Parameter
estimation*

Emulation and inference

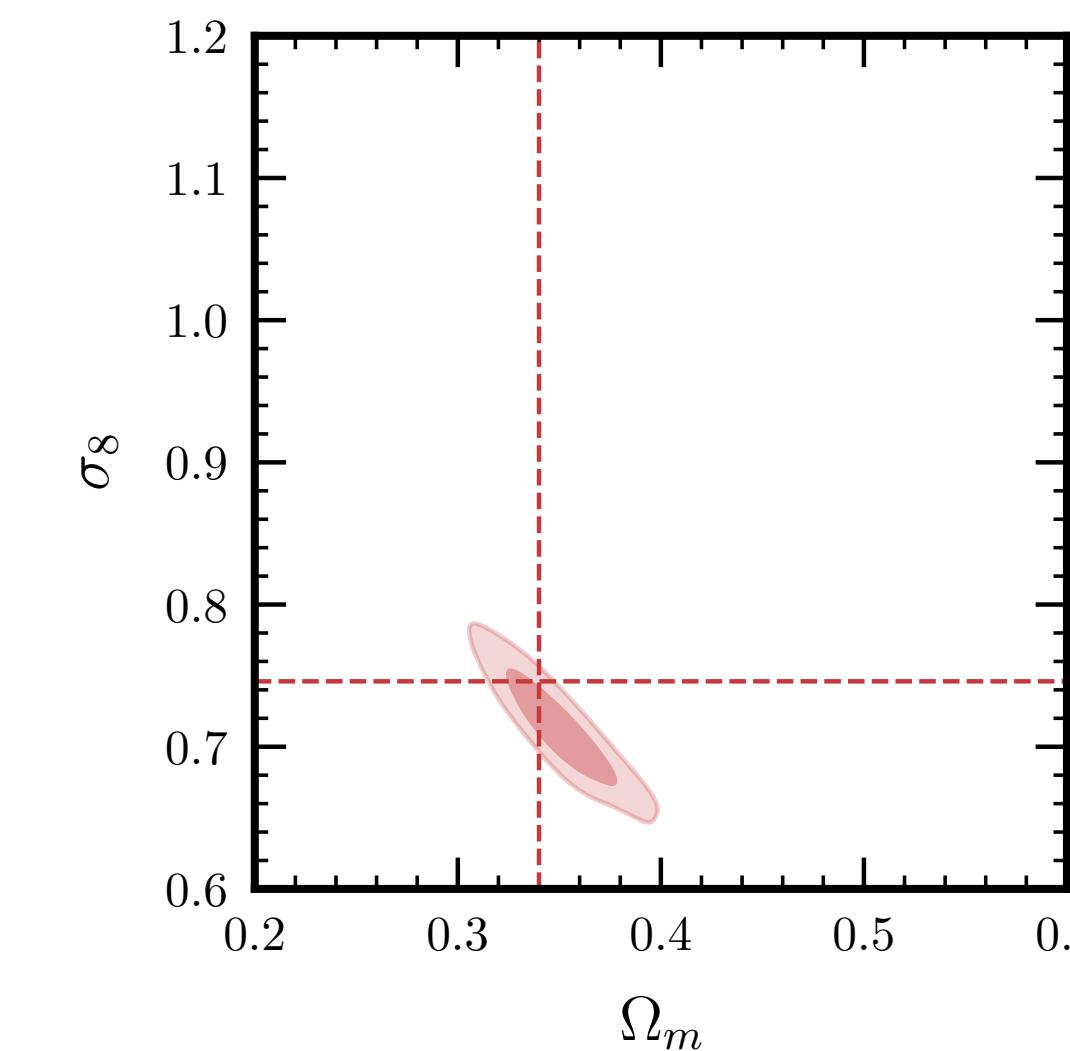


$$\sim p(\text{Cosmology} \mid \text{Cosmology})$$

*Emulation/
sampling*

$$p(\text{Cosmology} \mid \Omega_m, \sigma_8)$$

$\nabla_{\{\Omega_m, \sigma_8\}} p$ *Differentiable likelihood
→ even better!*

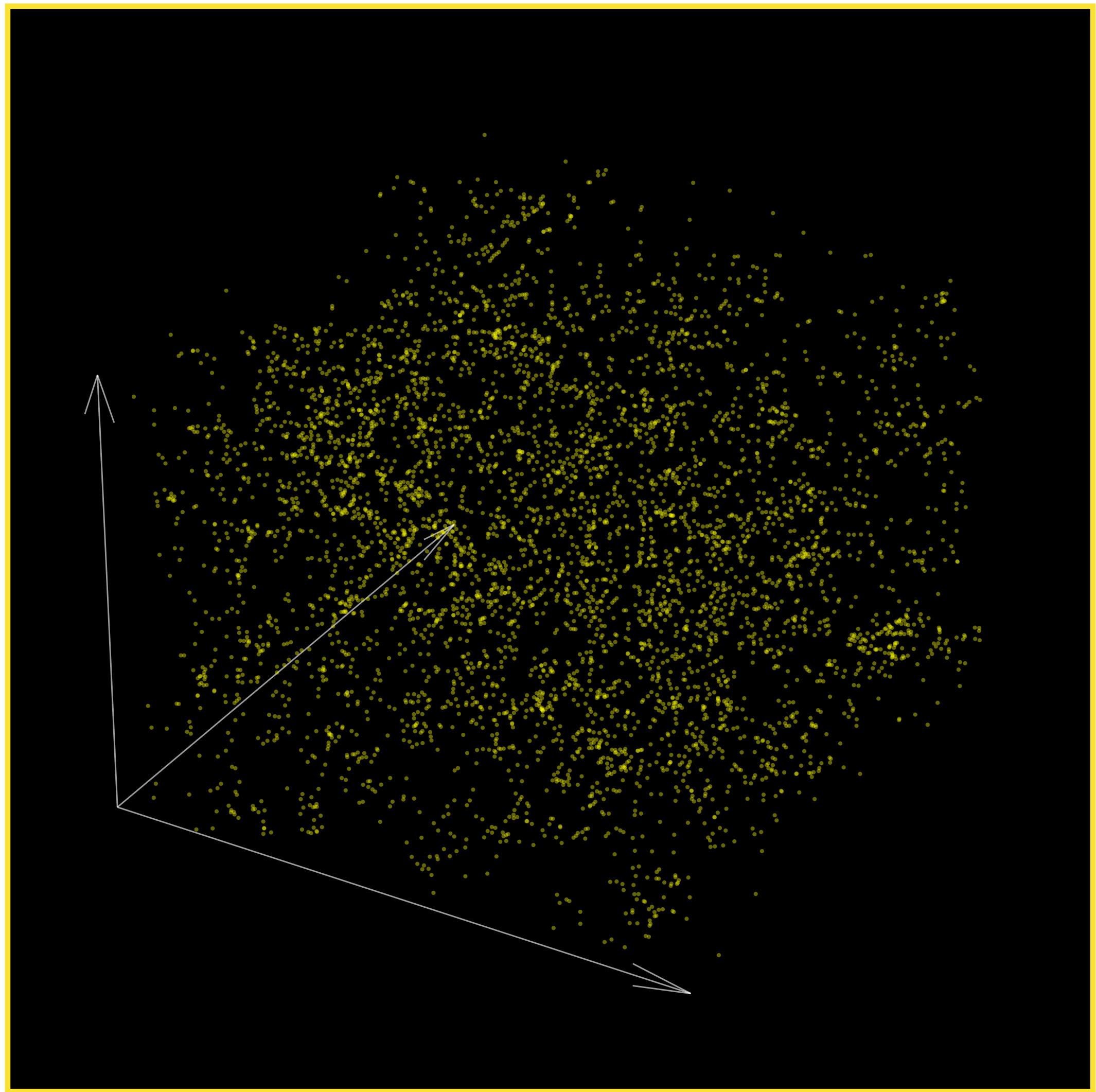


*Parameter
estimation*

The diffusion score model

Want a score model that

- Operates on sets of varying cardinality
- Is permutation equivariant
- Efficiently captures correlation structure of point cloud



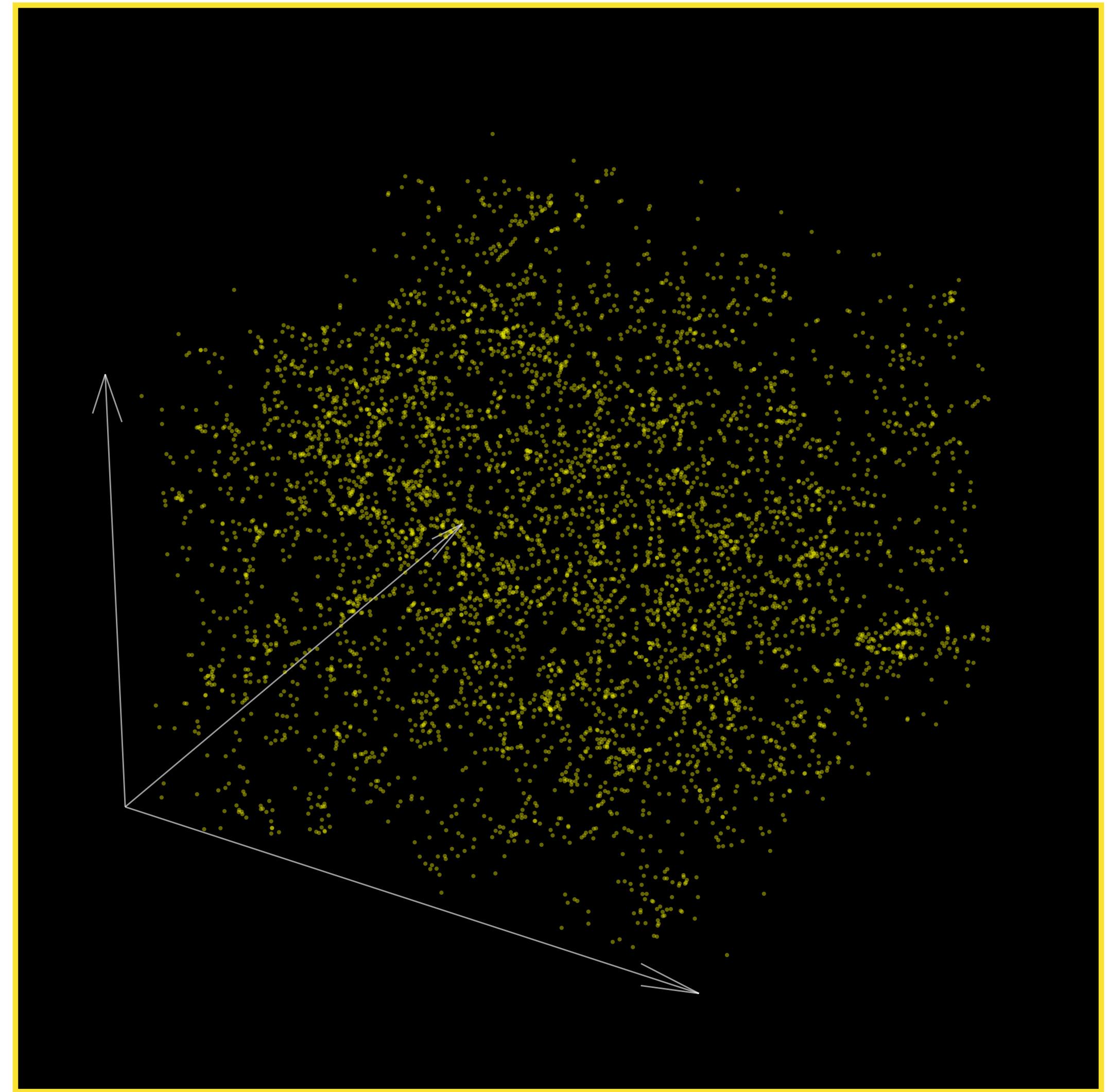
The diffusion score model

Want a score model that

- Operates on sets of varying cardinality
- Is permutation equivariant
- Efficiently captures correlation structure of point cloud

✓ Transformers

✓ Graph neural networks



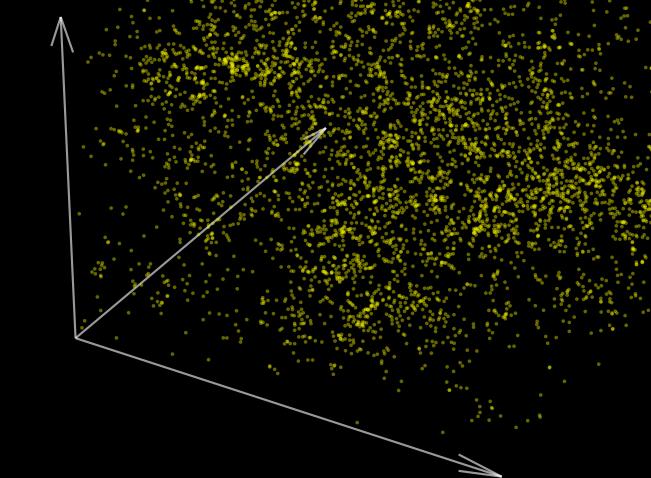
Graph neural network-guided diffusion on galaxies

SM, Cuesta-Lazaro [ICML ML4Astro]

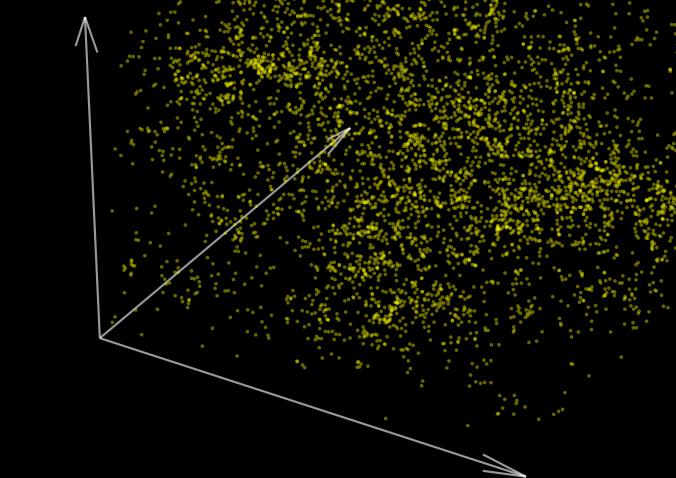
Samples from *Quijote* [Villaescusa-Navarro et al, 2021].

Forward process (*adding noise*)

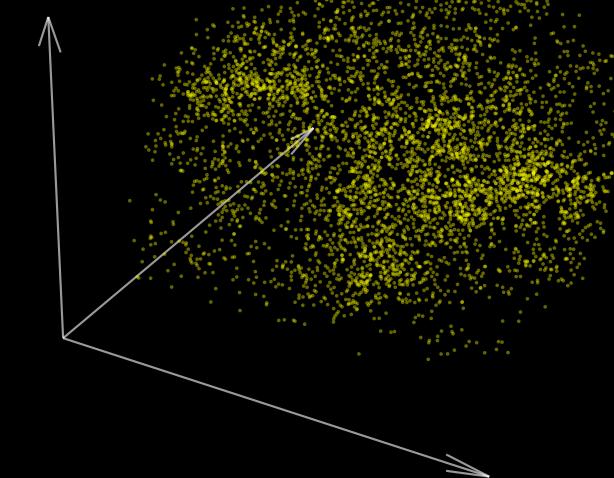
$t = 0.00$



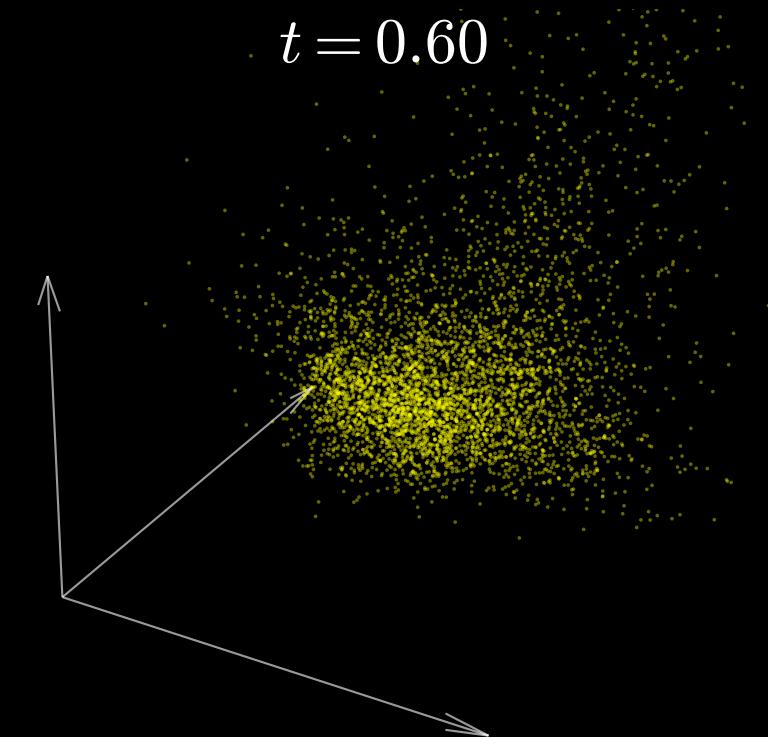
$t = 0.20$



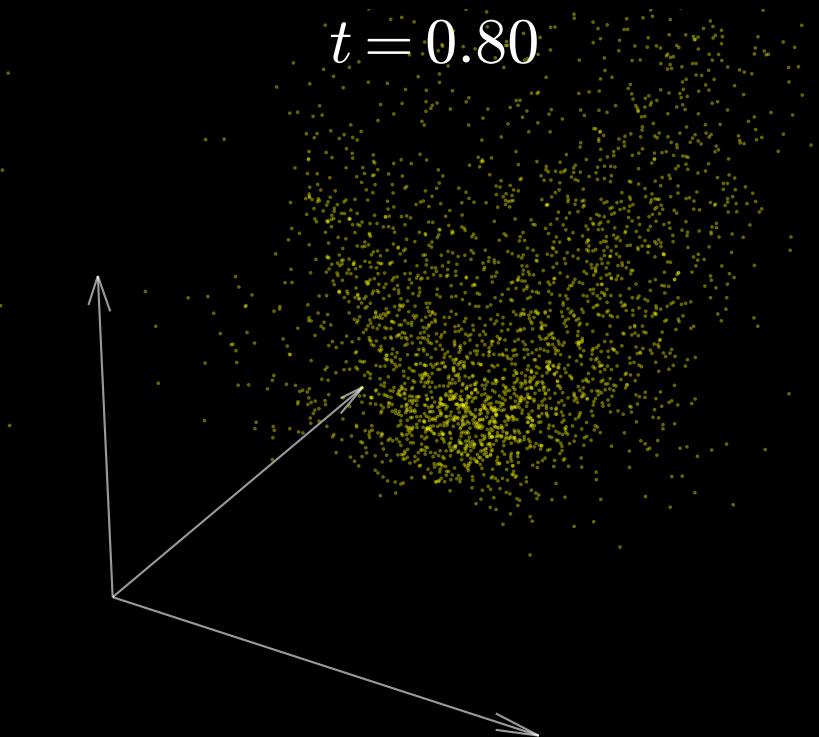
$t = 0.40$



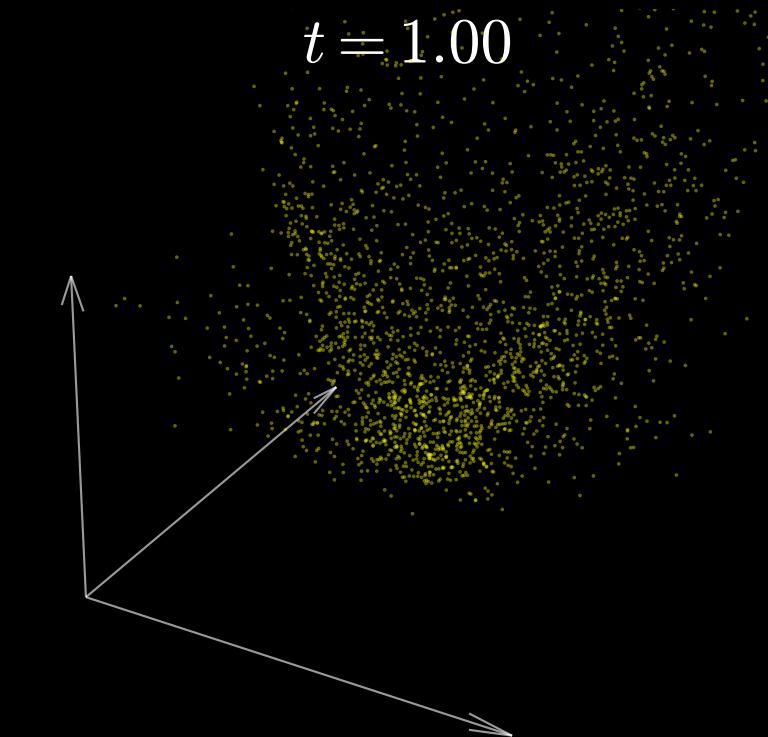
$t = 0.60$



$t = 0.80$



$t = 1.00$



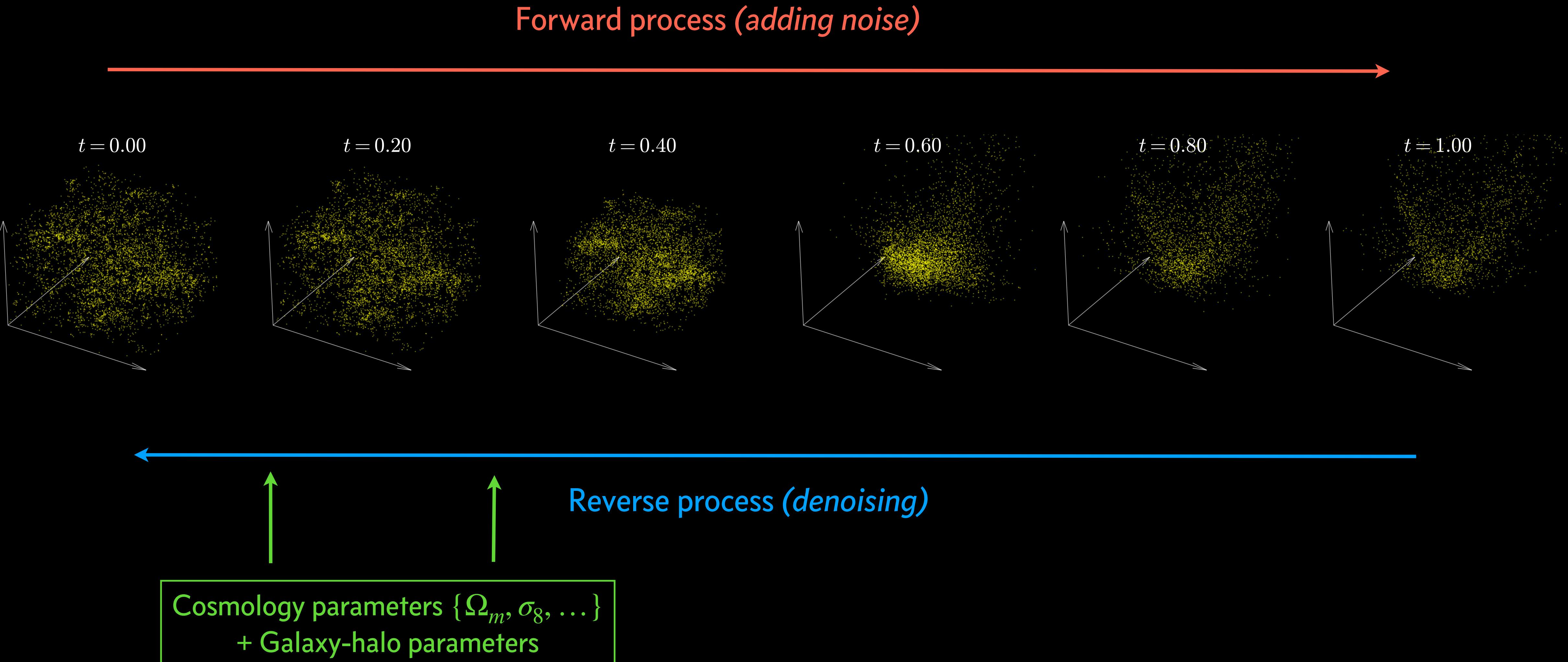
Reverse process (*denoising*)



Graph neural network-guided diffusion on galaxies

SM, Cuesta-Lazaro [ICML ML4Astro]

Samples from *Quijote* [Villaescusa-Navarro et al, 2021].

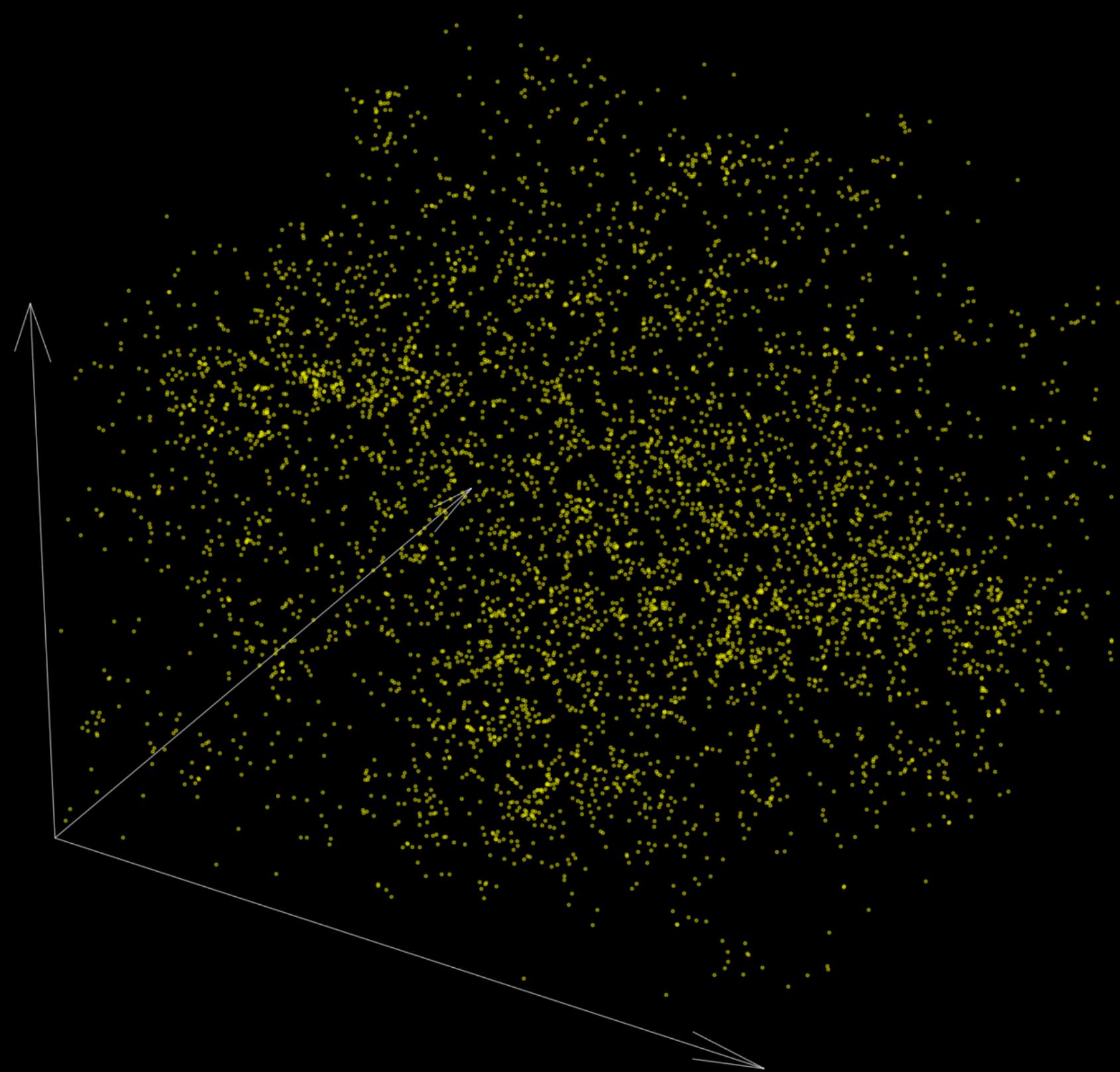


Diffusion on galaxies

SM, Cuesta-Lazaro [ICML ML4Astro]

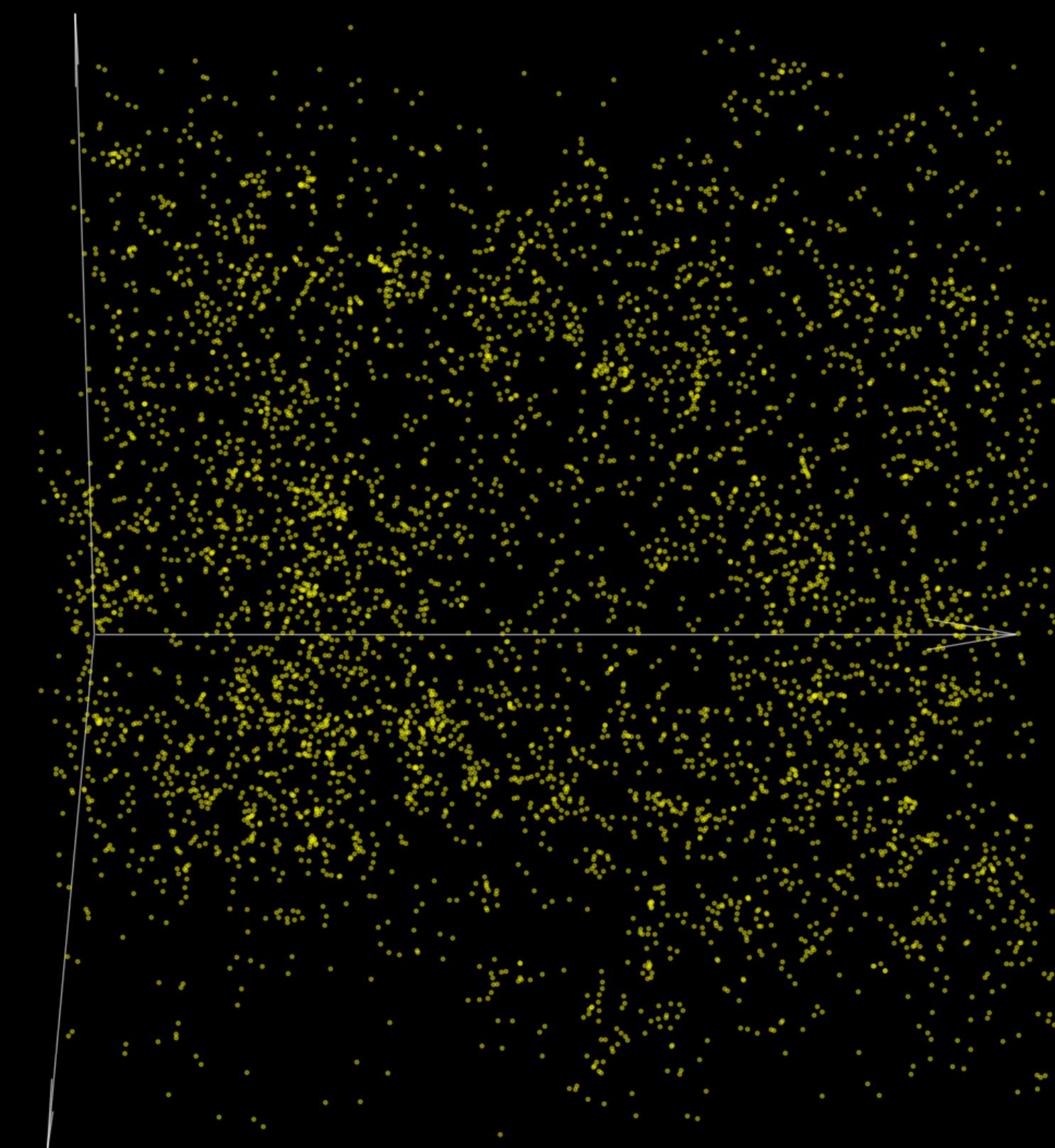
Diffusion process

$t = 0.00$



Conditional generation $x \sim p(x | \Omega_m, \sigma_8)$

$\Omega_m = 0.10, \sigma_8 = 0.60$

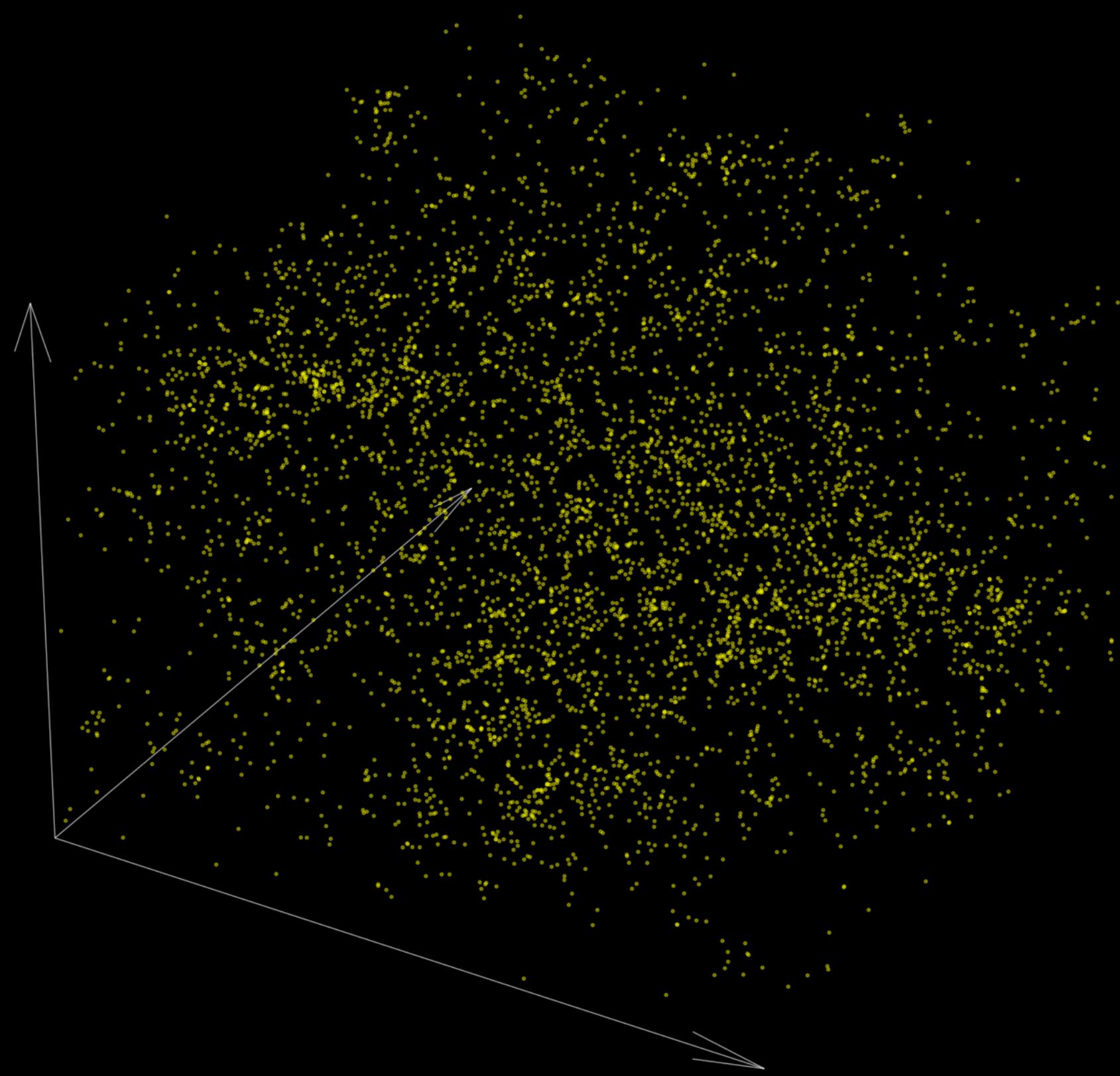


Diffusion on galaxies

SM, Cuesta-Lazaro [ICML ML4Astro]

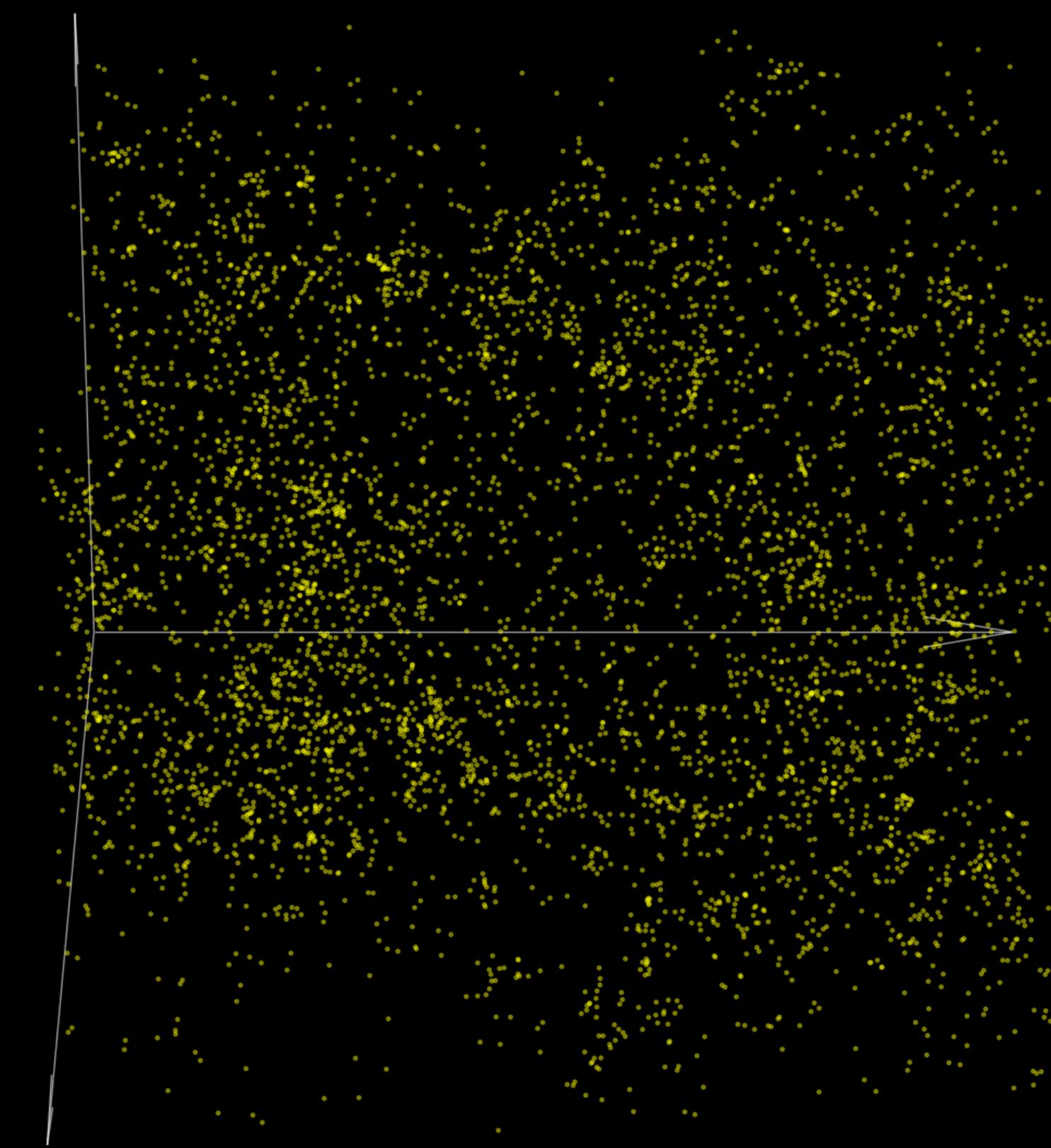
Diffusion process

$t = 0.00$



Conditional generation $x \sim p(x | \Omega_m, \sigma_8)$

$\Omega_m = 0.10, \sigma_8 = 0.60$

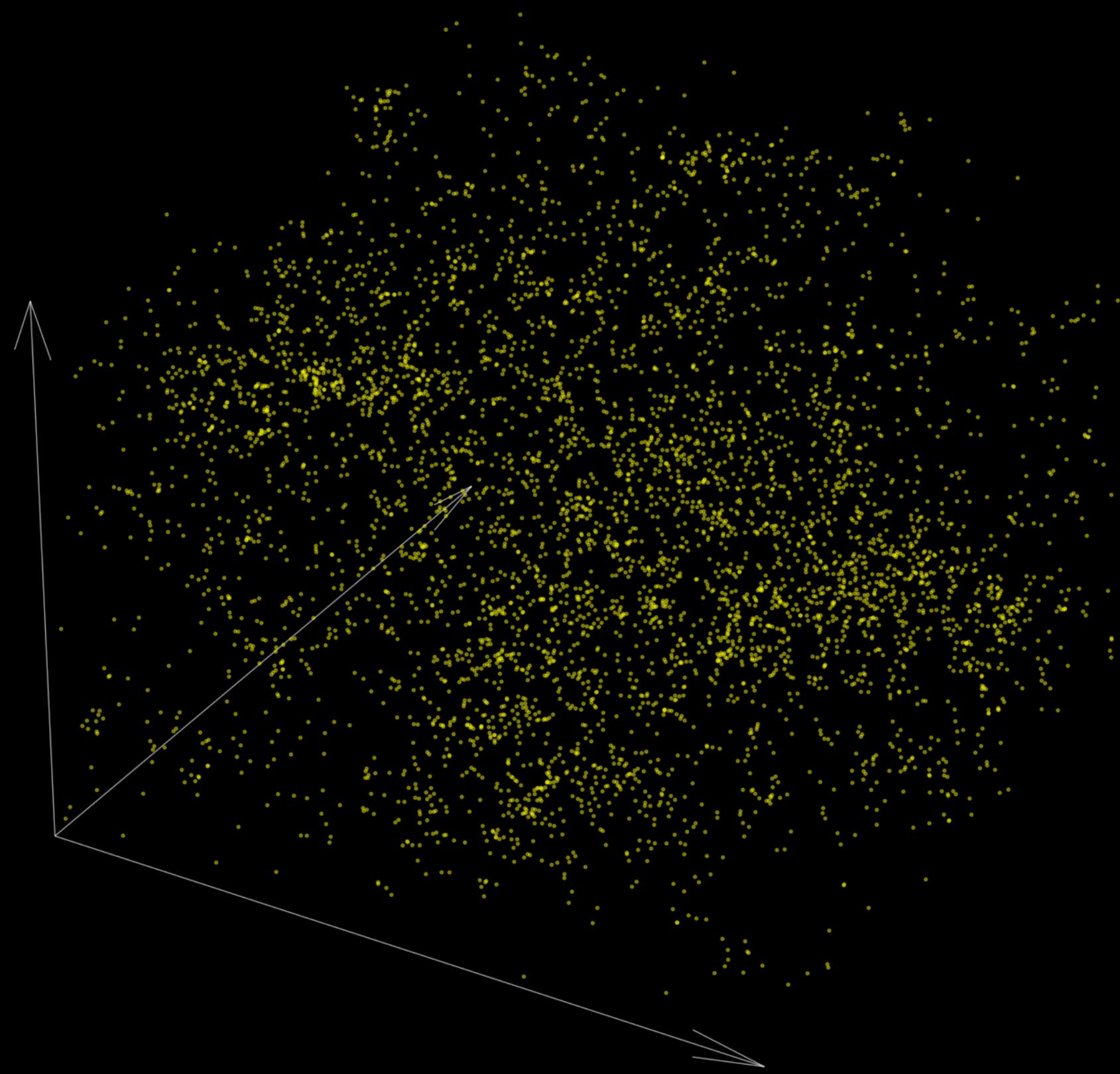


Diffusion on galaxies

SM, Cuesta-Lazaro [ICML ML4Astro]

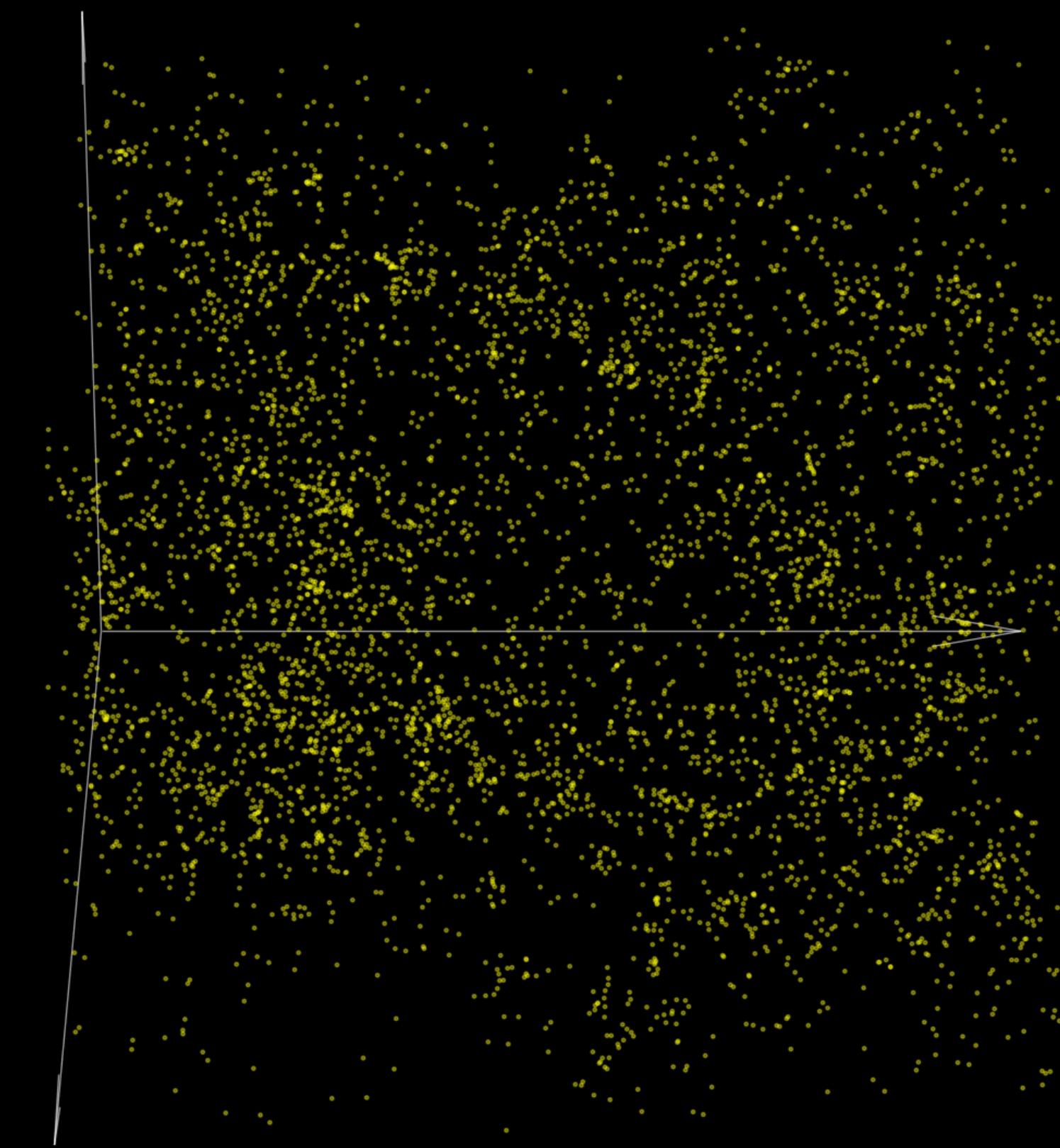
Diffusion process

$t = 0.00$



Conditional generation $x \sim p(x | \Omega_m, \sigma_8)$

$\Omega_m = 0.10, \sigma_8 = 0.60$



Likelihoods and parameter inference

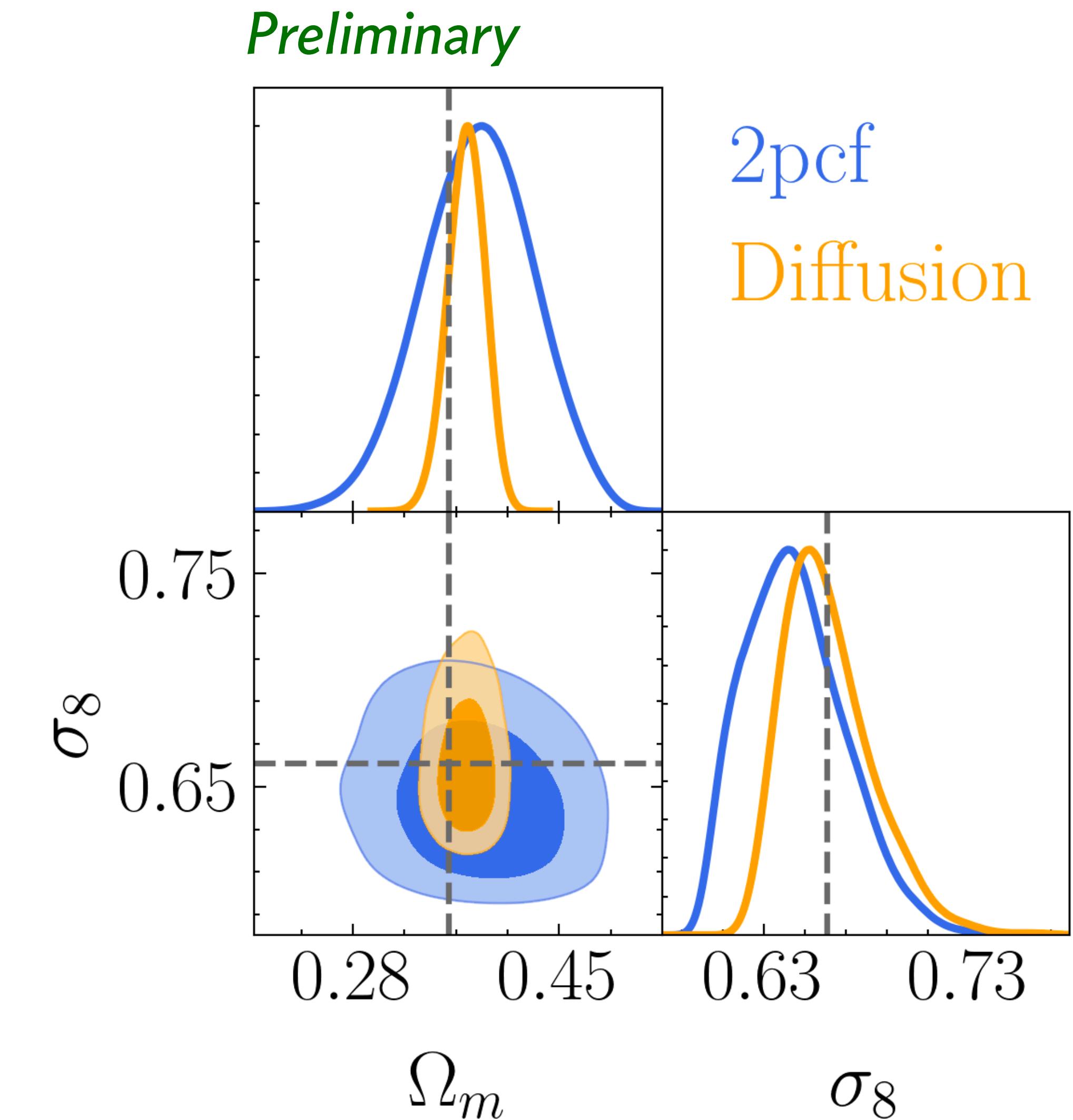
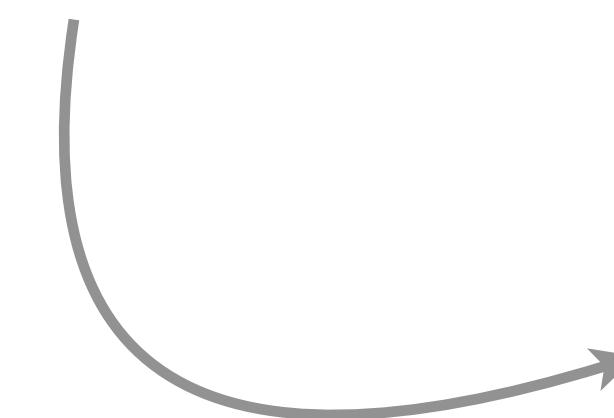
For a given dataset, can use the likelihood $p(x | \theta)$ for posterior parameter inference

- Monte Carlo sampling (MCMC, nested sampling, HMC...)
- Variational inference

Likelihoods and parameter inference

For a given dataset, can use the likelihood $p(x | \theta)$ for posterior parameter inference

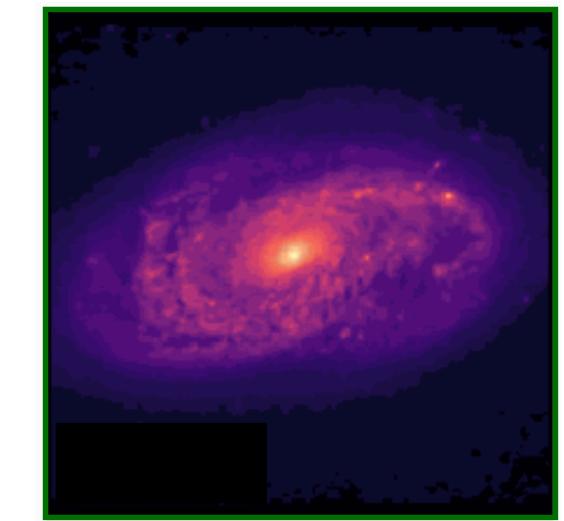
- Monte Carlo sampling (MCMC, nested sampling, HMC...)
- Variational inference



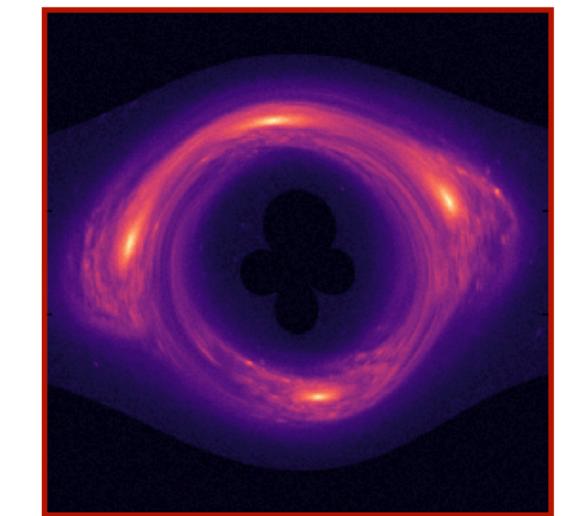
Another application: as a *galaxy prior for gravitational lensing*

[Adam, Coogan, Malkin et al 2022]

$p(\text{galaxies})$



↓ *Gravitational
lensing*



Another application: as a galaxy prior for gravitational lensing

[Adam, Coogan, Malkin et al 2022]

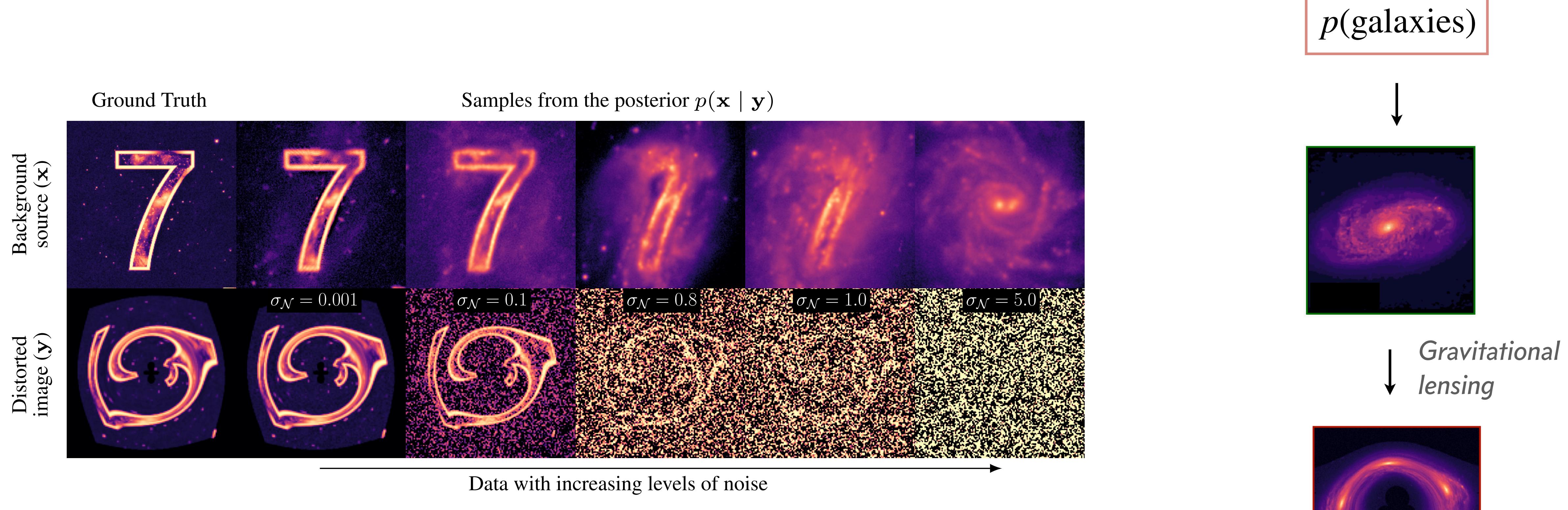
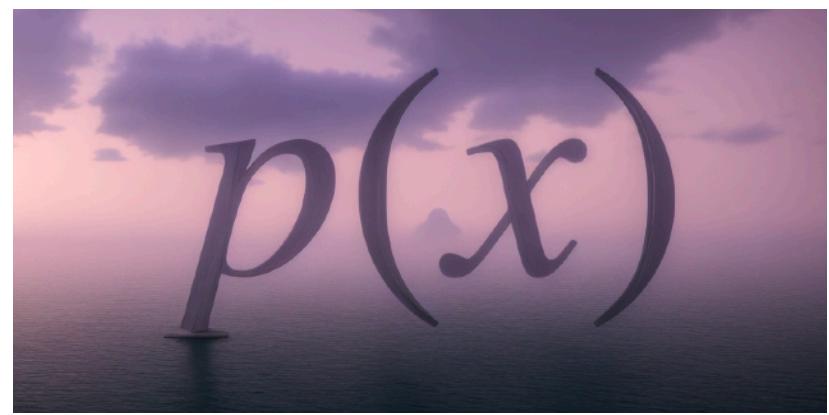
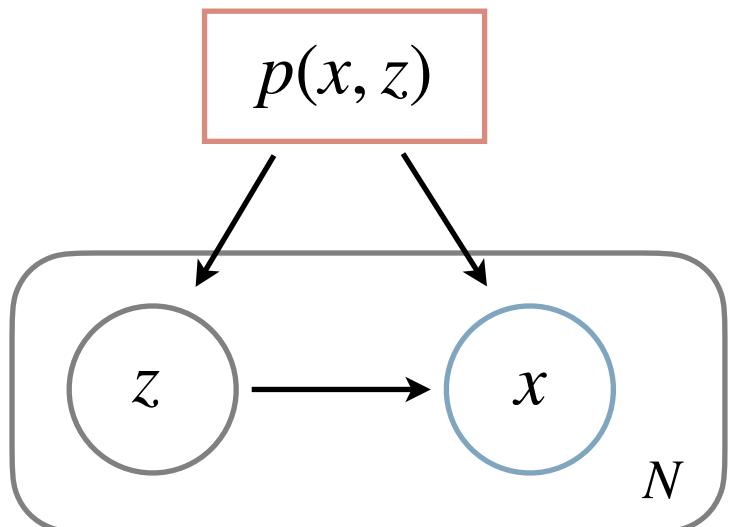


Figure 2: Application of the method to a lensing system with a highly out-of-distribution source. The ground truth is given in the leftmost panel. Other panels show increasingly noisy data (lower row) and a sample from their corresponding source posterior (upper row). As the likelihood becomes less informative, the prior dominates, making the sources increasingly look like galaxies.

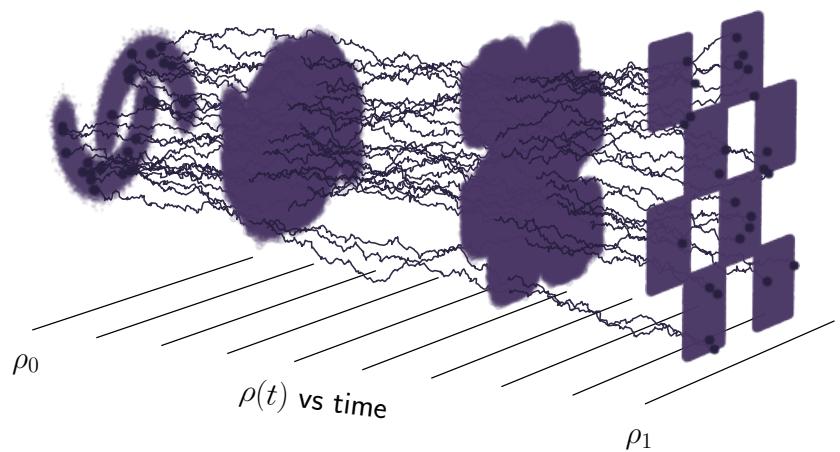
Outline



Why (deep) generative modeling?
What is it, and what can it do for you?



Variational auto encoders
Latent-variable modeling, and compression is all you need



Diffusion models
Models based on iterative refinement

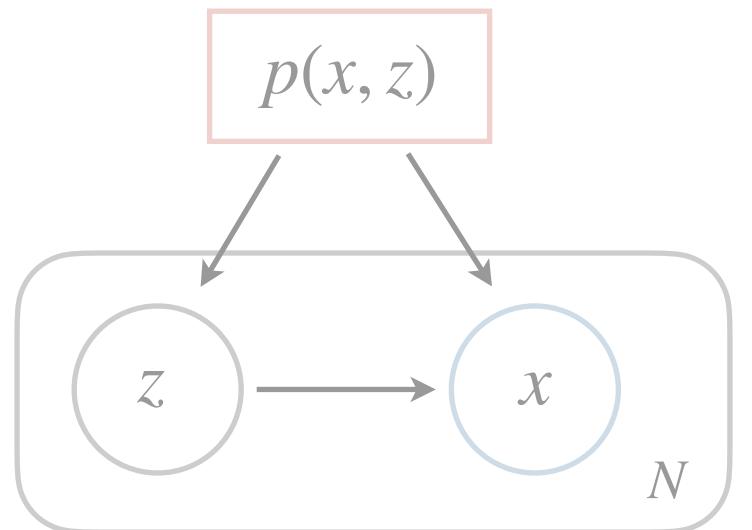


Normalizing flows
Invertible transformations

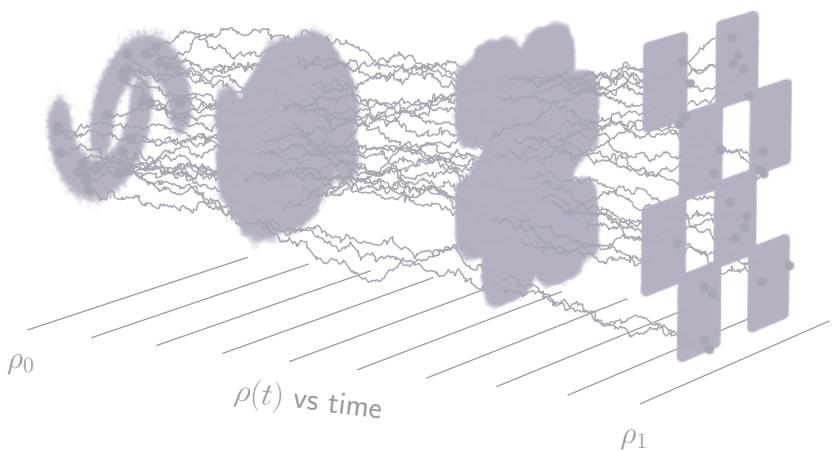
Outline



Why (deep) generative modeling?
What is it, and what can it do for you?



Variational auto encoders
Latent-variable modeling, and compression is all you need



Diffusion models
Models based on iterative refinement



Normalizing flows
Invertible transformations

Normalizing flows

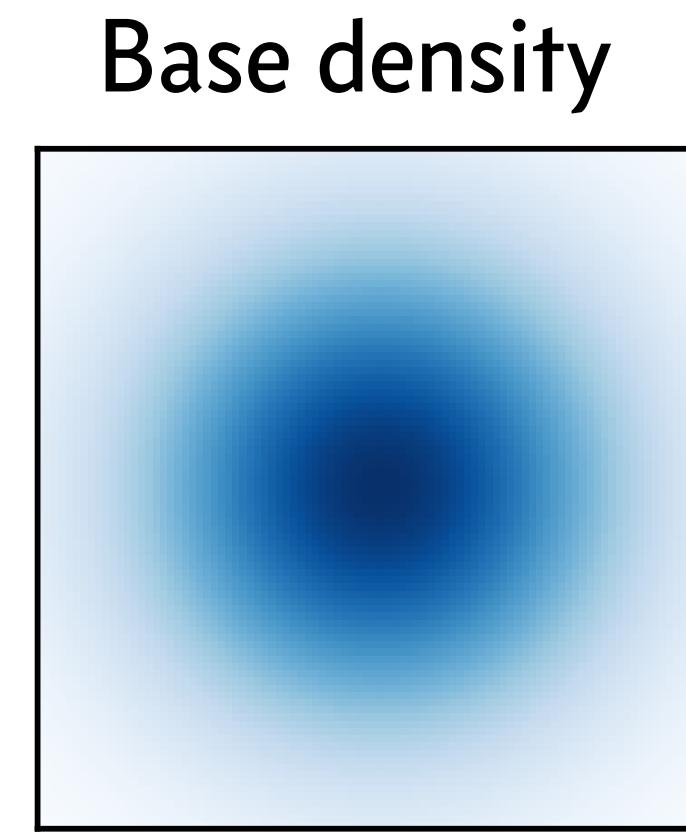
(IAIFI logo)

Target density

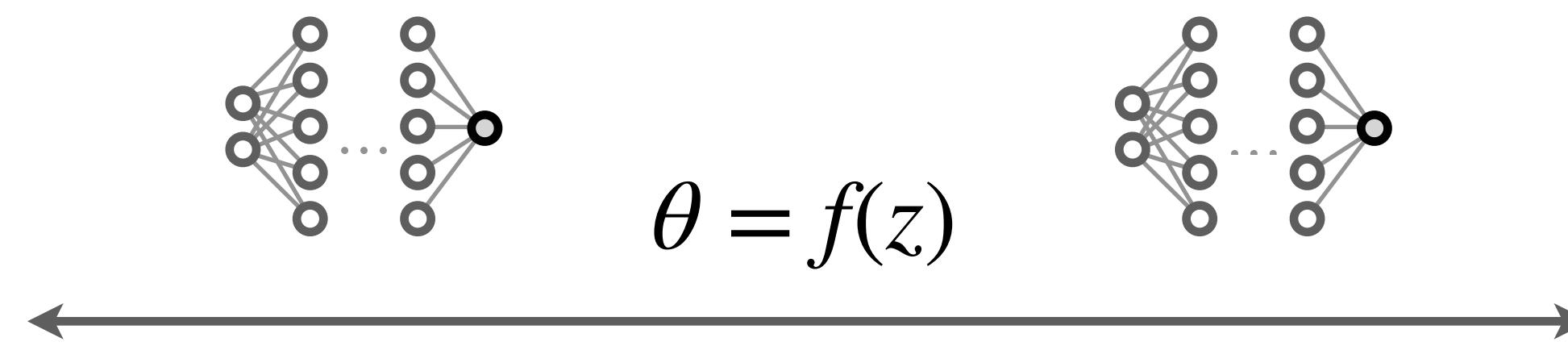


$$p(\theta)$$

Normalizing flows



$$p_z(z)$$



One-to-one transformation

Tractable f^{-1} and $\det \nabla f$

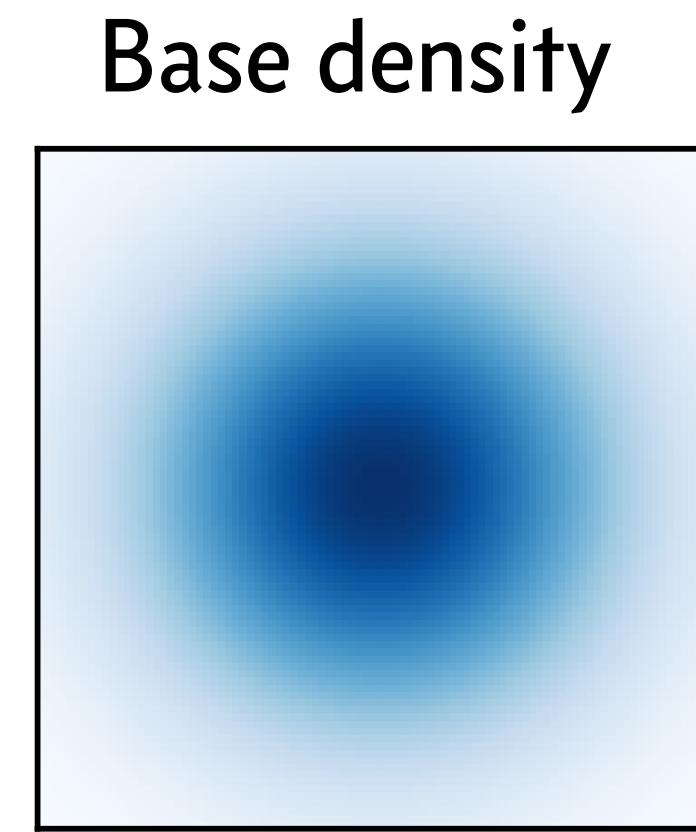
(IAIFI logo)

Target density

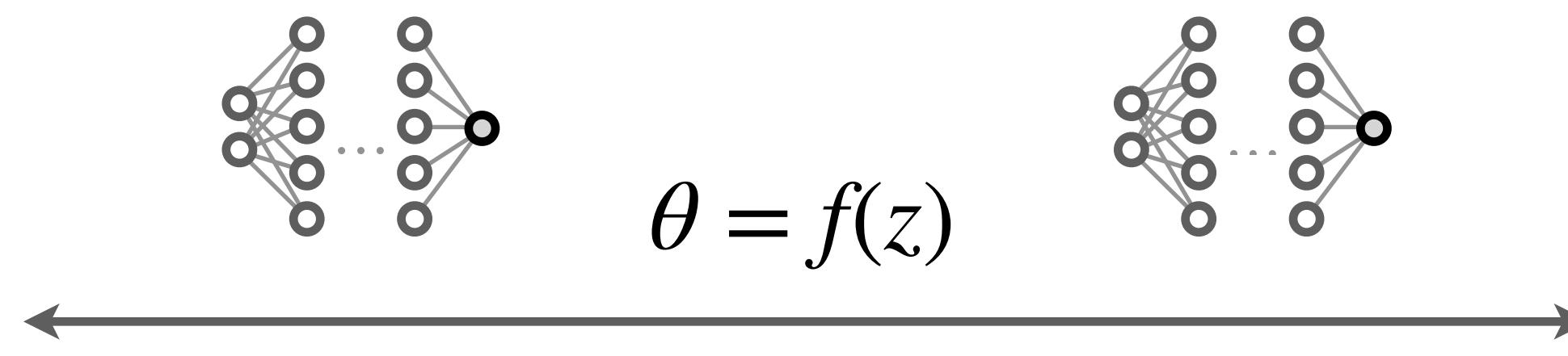


$$p(\theta) = p_z(f^{-1}(\theta)) |\det \nabla f|^{-1}$$

Normalizing flows



$$p_z(z)$$



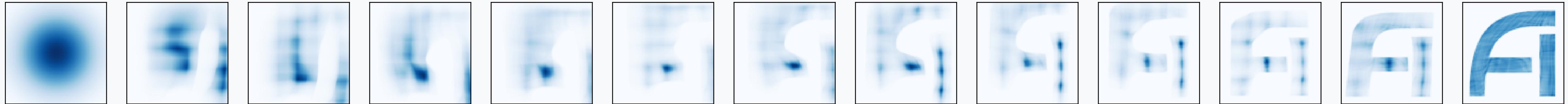
One-to-one transformation

Tractable f^{-1} and $\det \nabla f$

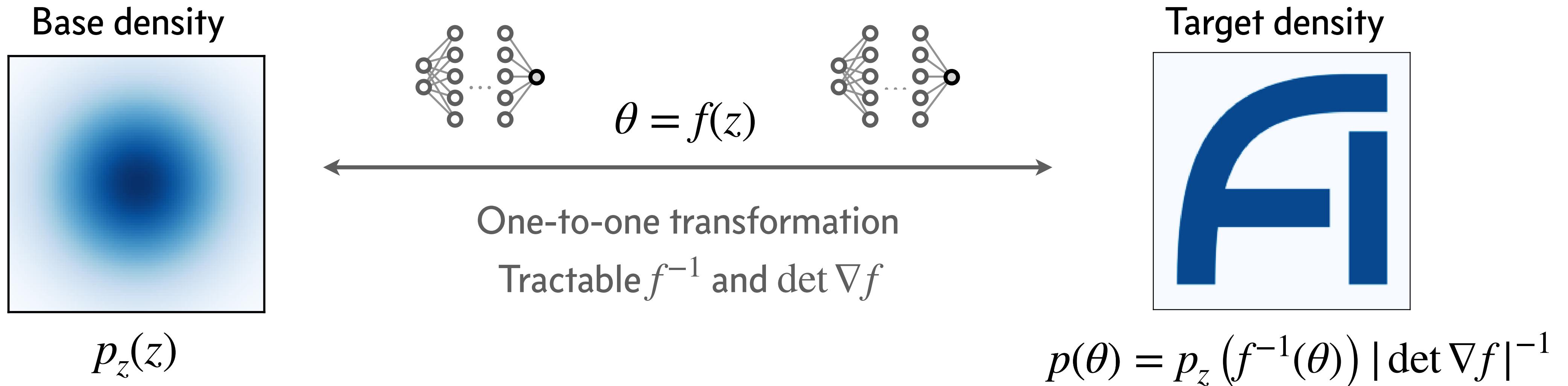


$$p(\theta) = p_z(f^{-1}(\theta)) |\det \nabla f|^{-1}$$

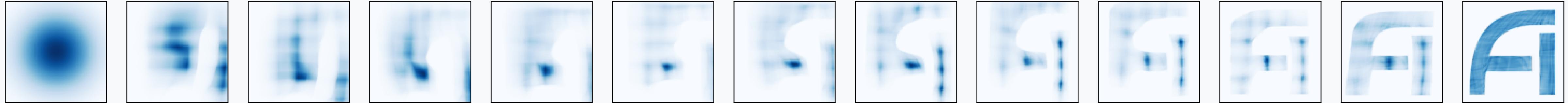
Efficient density estimation: $\log \hat{p}(\theta)$



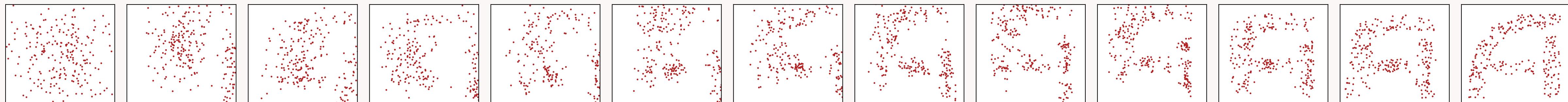
Normalizing flows



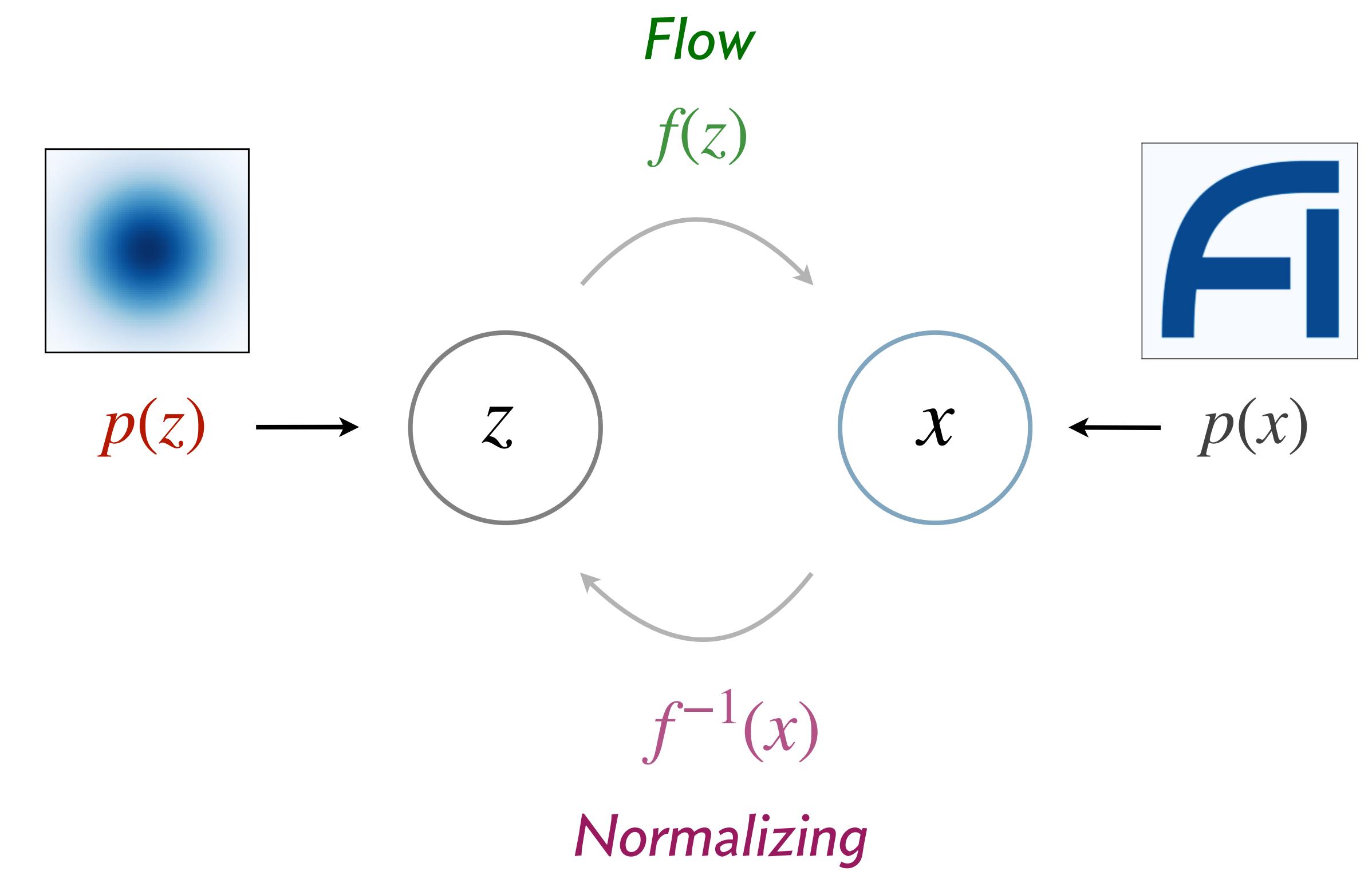
Efficient density estimation: $\log \hat{p}(\theta)$



Efficient sampling: $\theta \sim \hat{p}(\theta)$



Normalizing flows



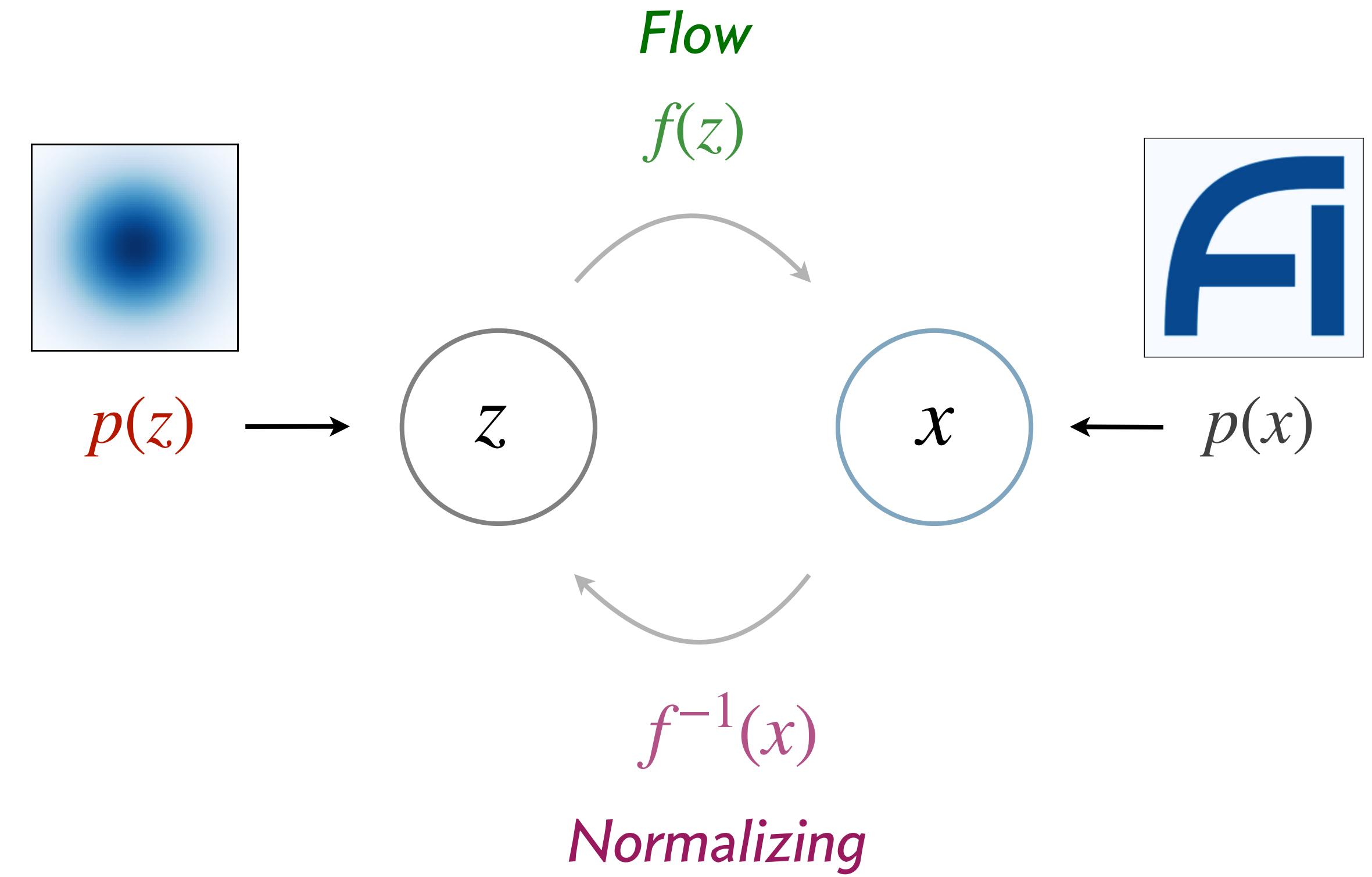
Normalizing flows

The distribution $p(z)$ should

- *Have an easy-to-evaluate density*
- *Be easy to sample from $z \sim p(z)$*

Typically

$$p(z) = \mathcal{N}(0, \mathbb{I})$$



Normalizing flows

The distribution $p(z)$ should

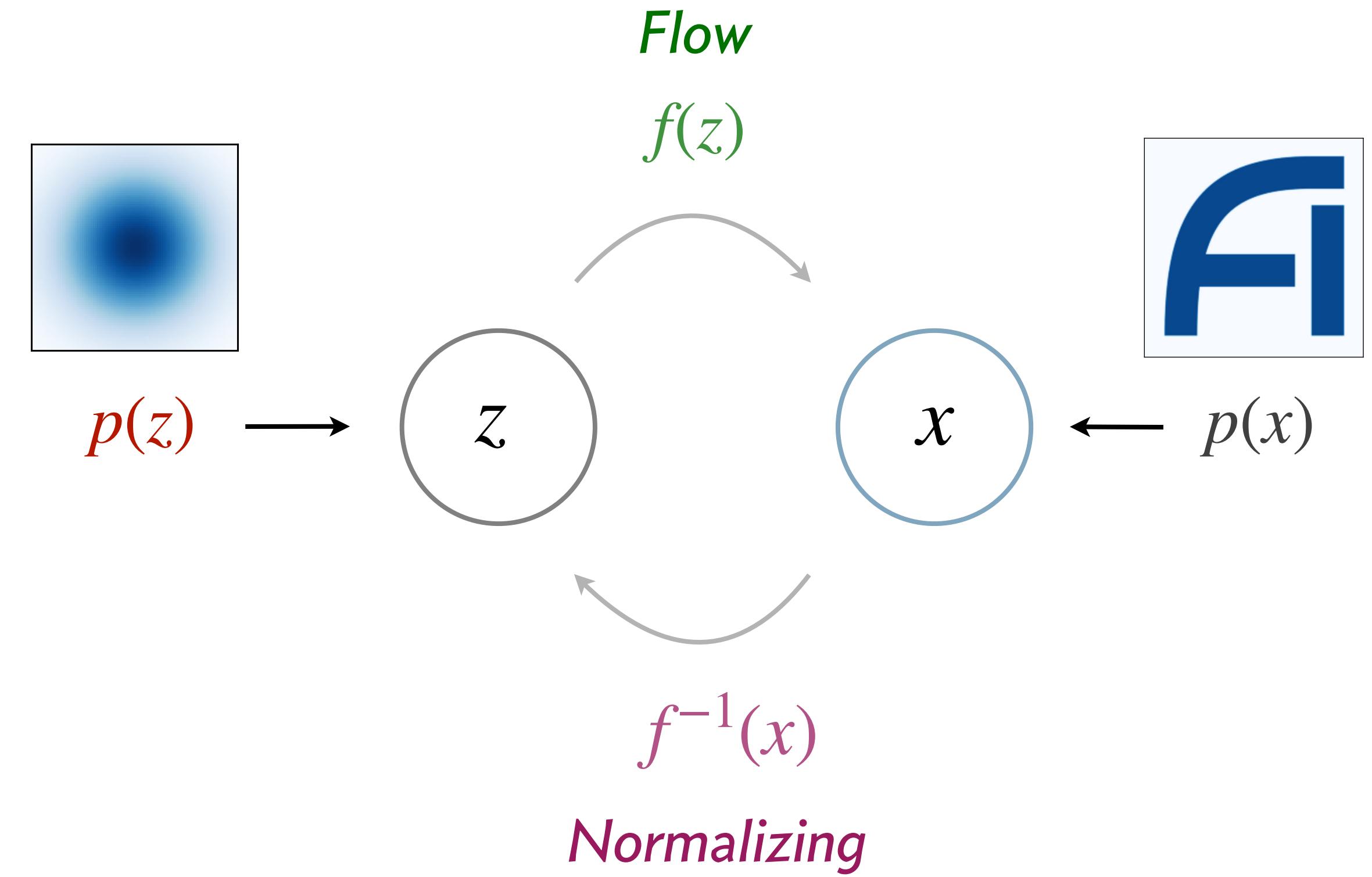
- *Have an easy-to-evaluate density*
- *Be easy to sample from $z \sim p(z)$*

Typically

$$p(z) = \mathcal{N}(0, \mathbb{I})$$

The function f should be

- *One-to-one*
 - *Differentiable*
 - *Invertible*
 - *Tractable f^{-1} and $\det \nabla f$*
- } Diffeomorphism



Normalizing flows

The distribution $p(z)$ should

- Have an easy-to-evaluate density
- Be easy to sample from $z \sim p(z)$

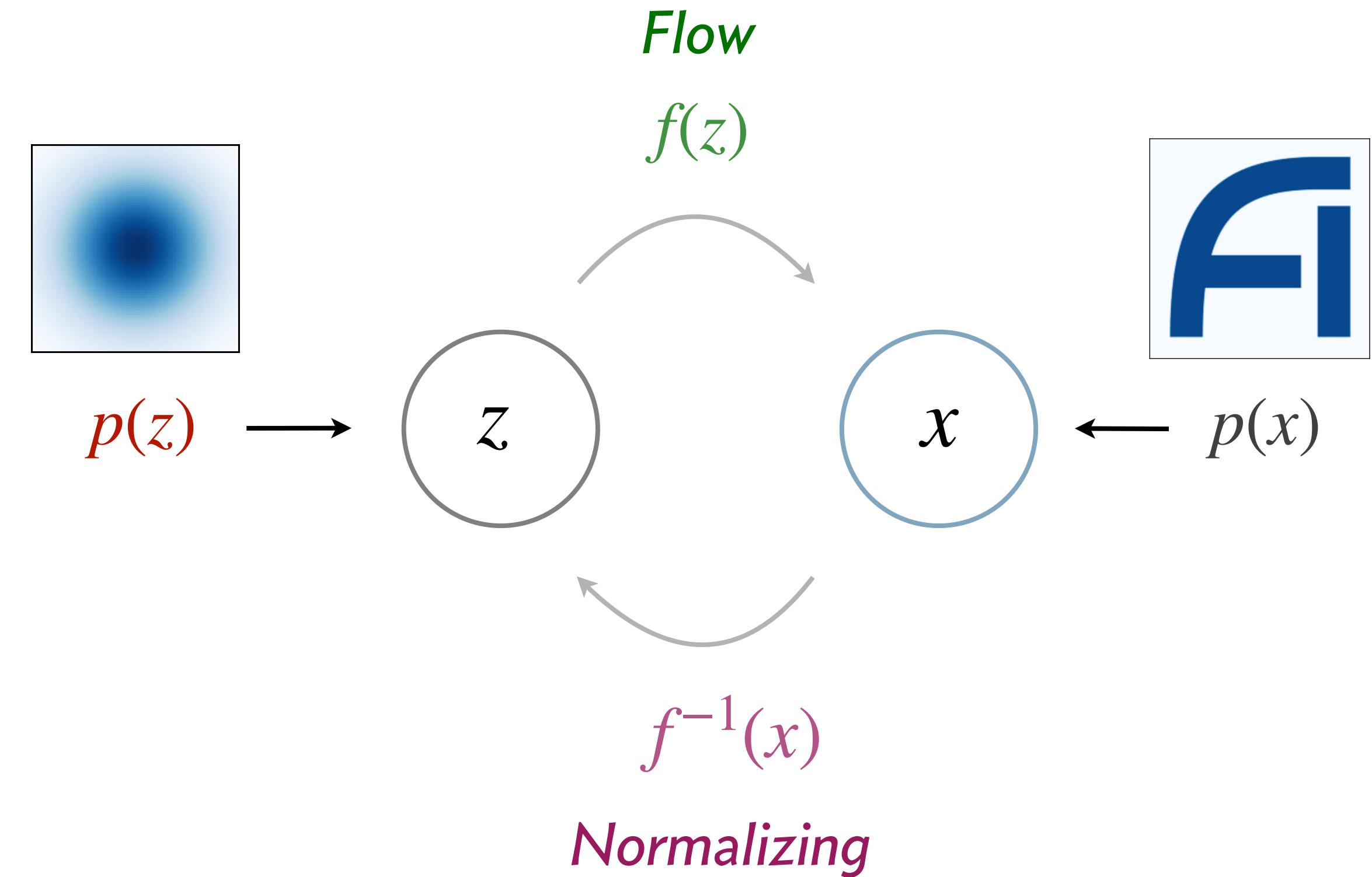
Typically

$$p(z) = \mathcal{N}(0, \mathbb{I})$$

The function f should be

- One-to-one
- Differentiable
- Invertible
- Tractable f^{-1} and $\det \nabla f$

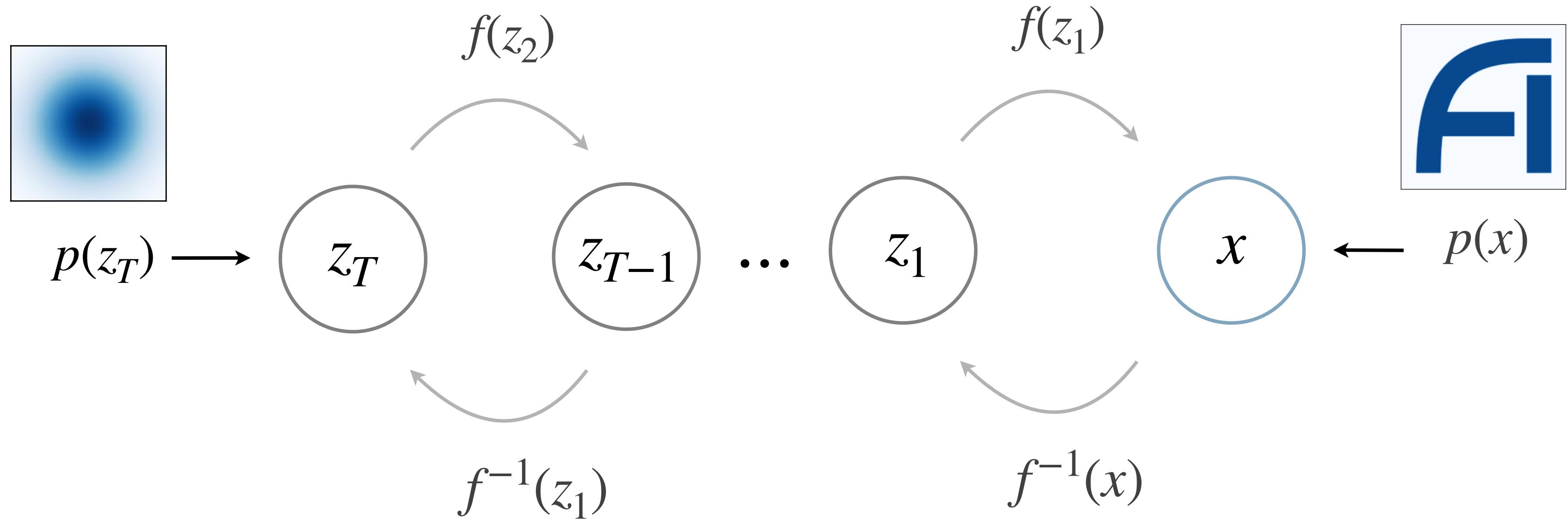
} Diffeomorphism



- Constrained form of the transformation can limit the expressivity of flows compared to e.g. diffusion models.
- However, for certain physics applications the transformation can be restricted in a specific, desired way; *see Miranda Cheng's lectures on Wednesday!*

Normalizing flows

Multiple flow transformation can be easily composed for e.g. expressivity



Computing $p(x)$: *change-of-variables formula*

$$\int p(x)dx = \int p(z)dz = 1$$

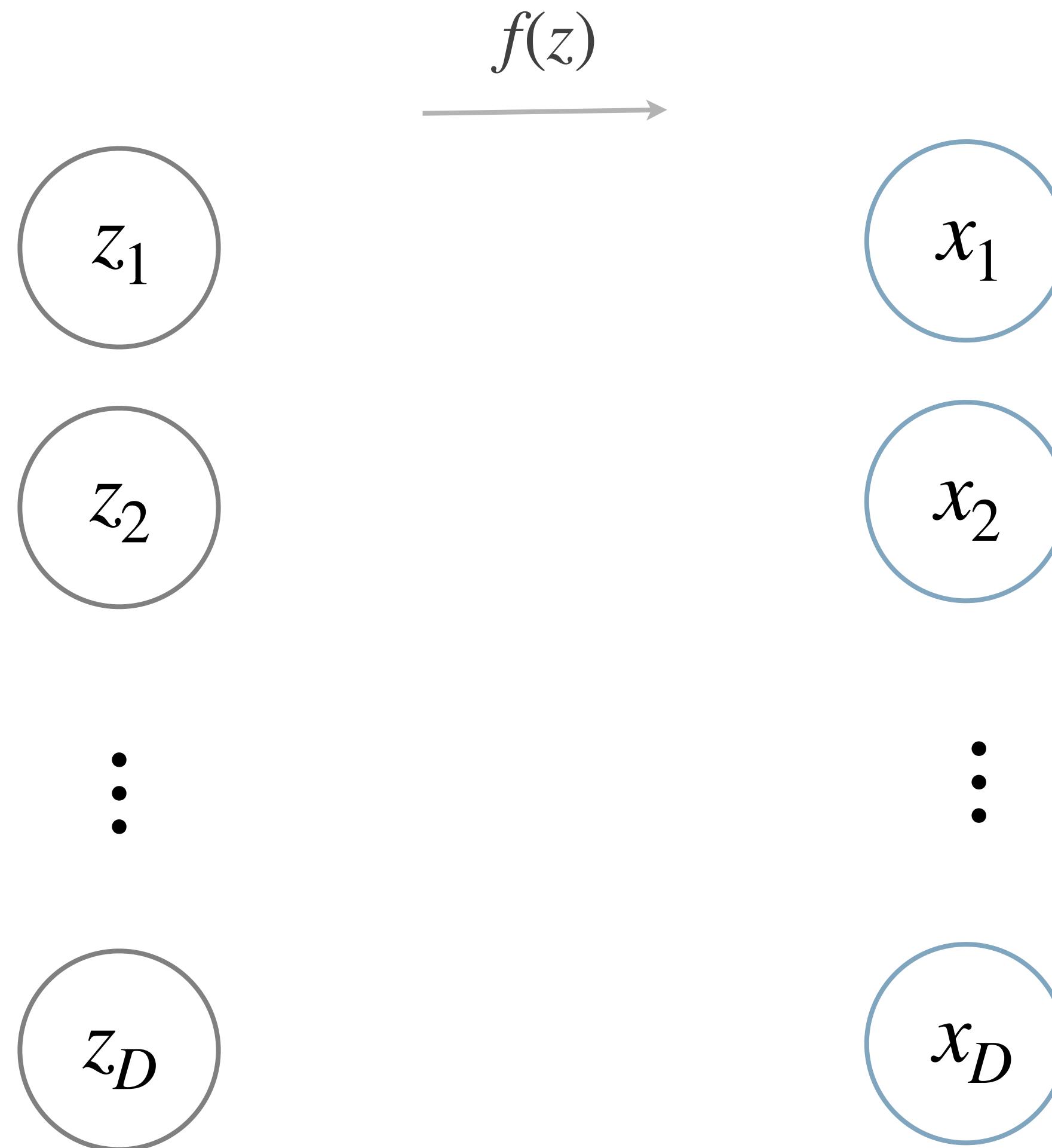
$$p(x) = p(z) \left| \frac{dz}{dx} \right| = p(f^{-1}(x)) \left| \frac{df^{-1}}{dx} \right| = p(f^{-1}(x)) |\det \nabla f|^{-1}$$

Train using maximum-likelihood objective

$$\varphi^* = \left\langle \arg \max_{\varphi} p(f_{\varphi}^{-1}(x)) |\det \nabla f_{\varphi}|^{-1} \right\rangle_{x \sim p(x)}$$

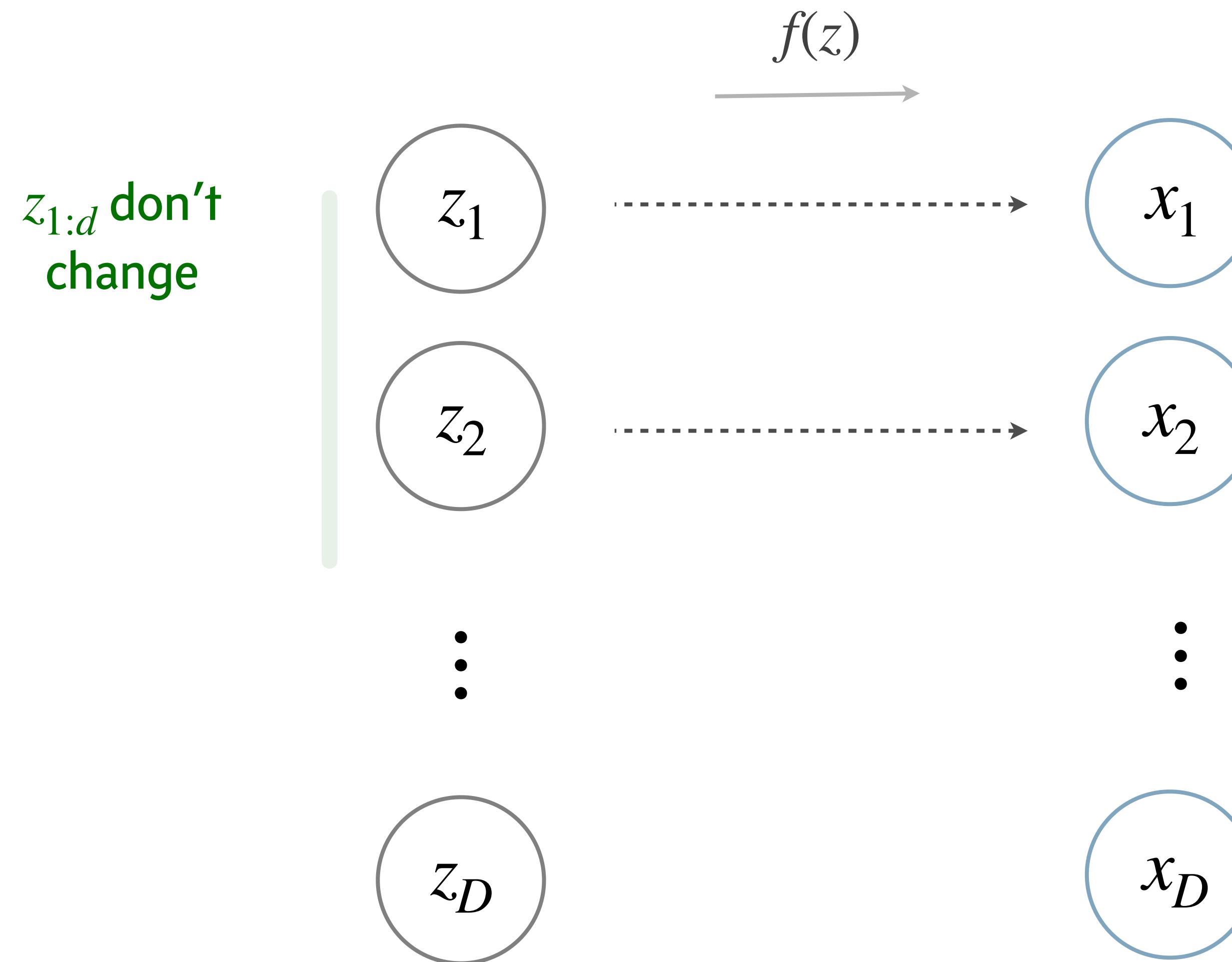
Simple flow transformations

Example: *Affine coupling flow* [RealNVP; Dinh et al 2016]



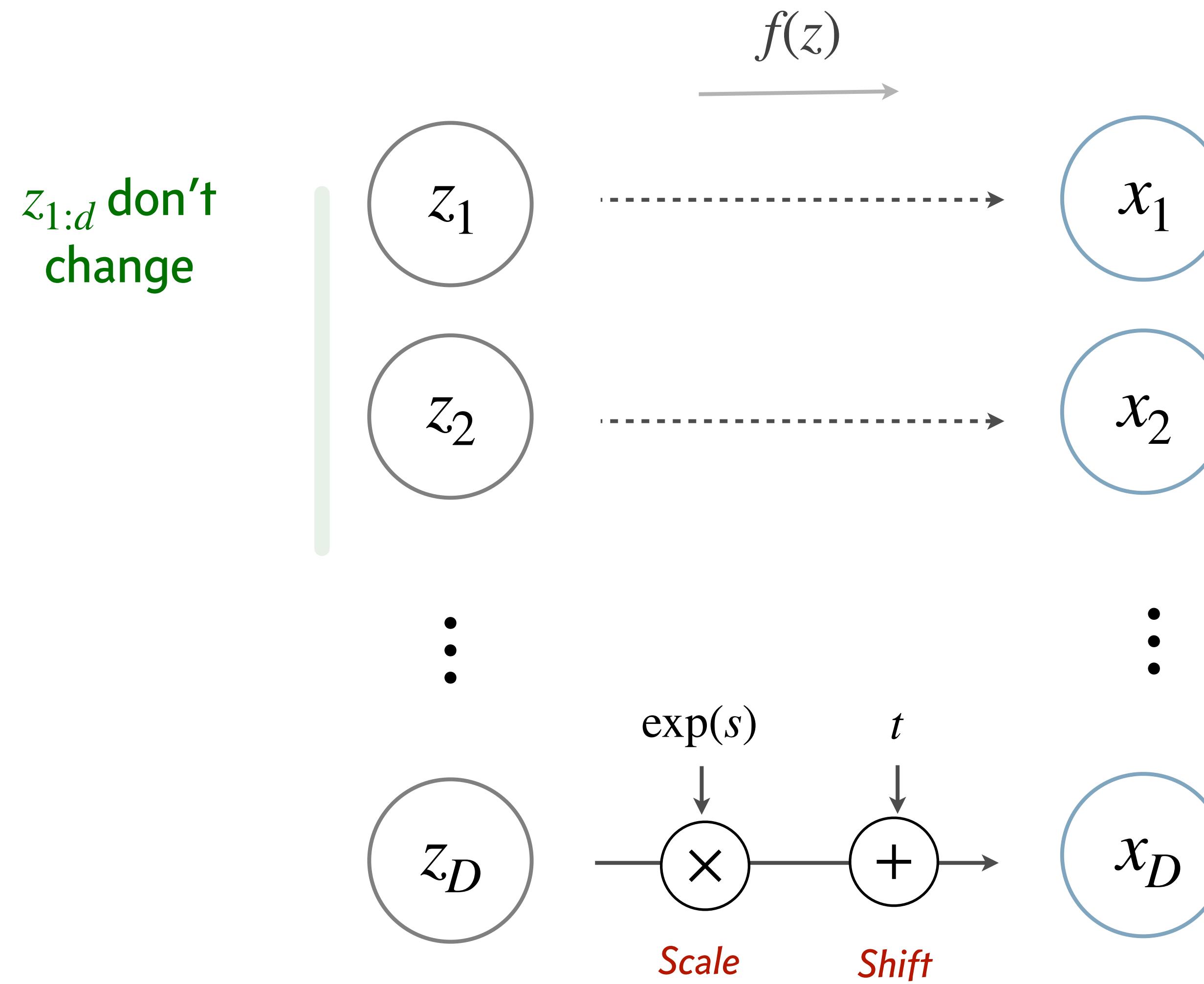
Simple flow transformations

Example: *Affine coupling flow* [RealNVP; Dinh et al 2016]



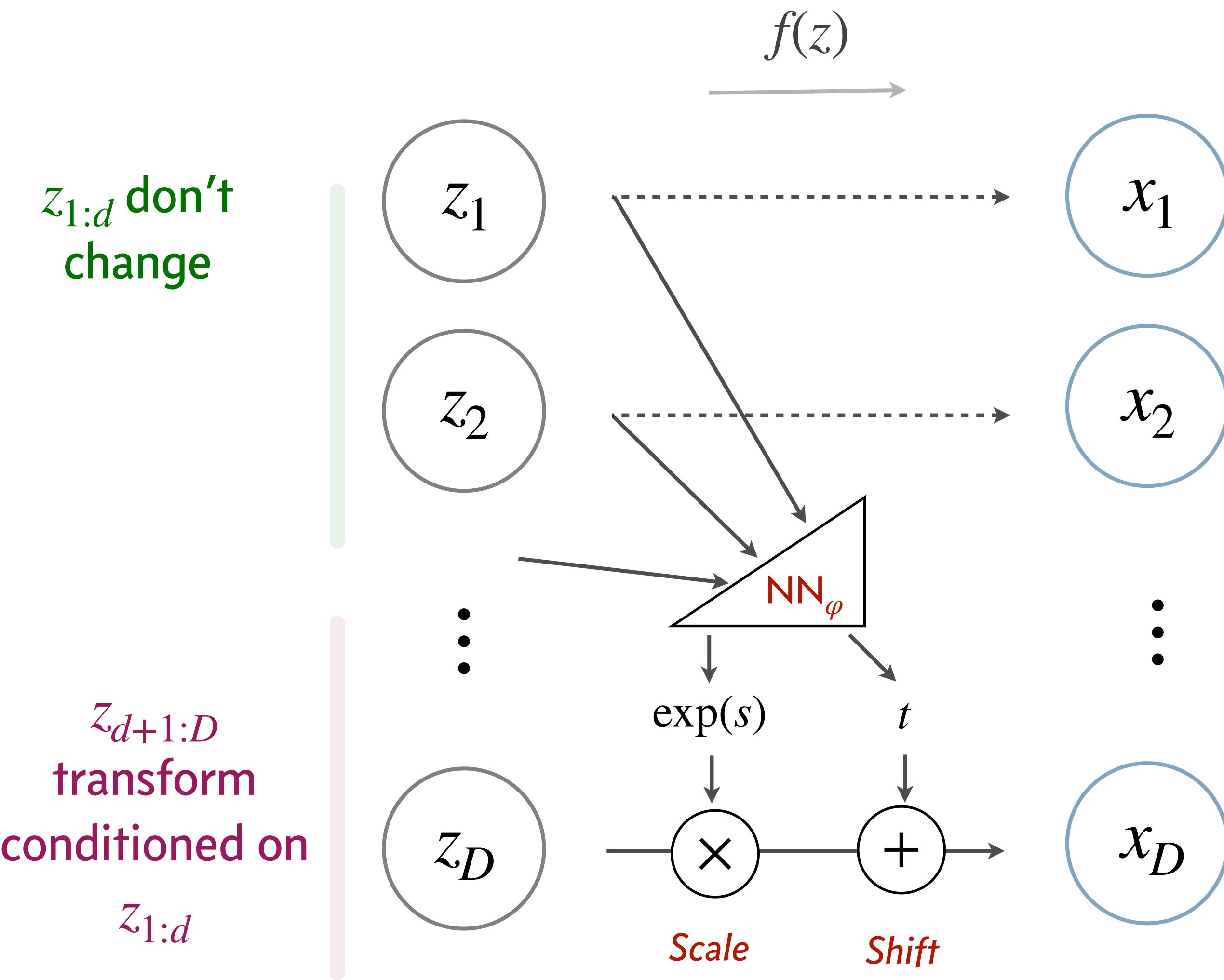
Simple flow transformations

Example: *Affine coupling flow* [RealNVP; Dinh et al 2016]



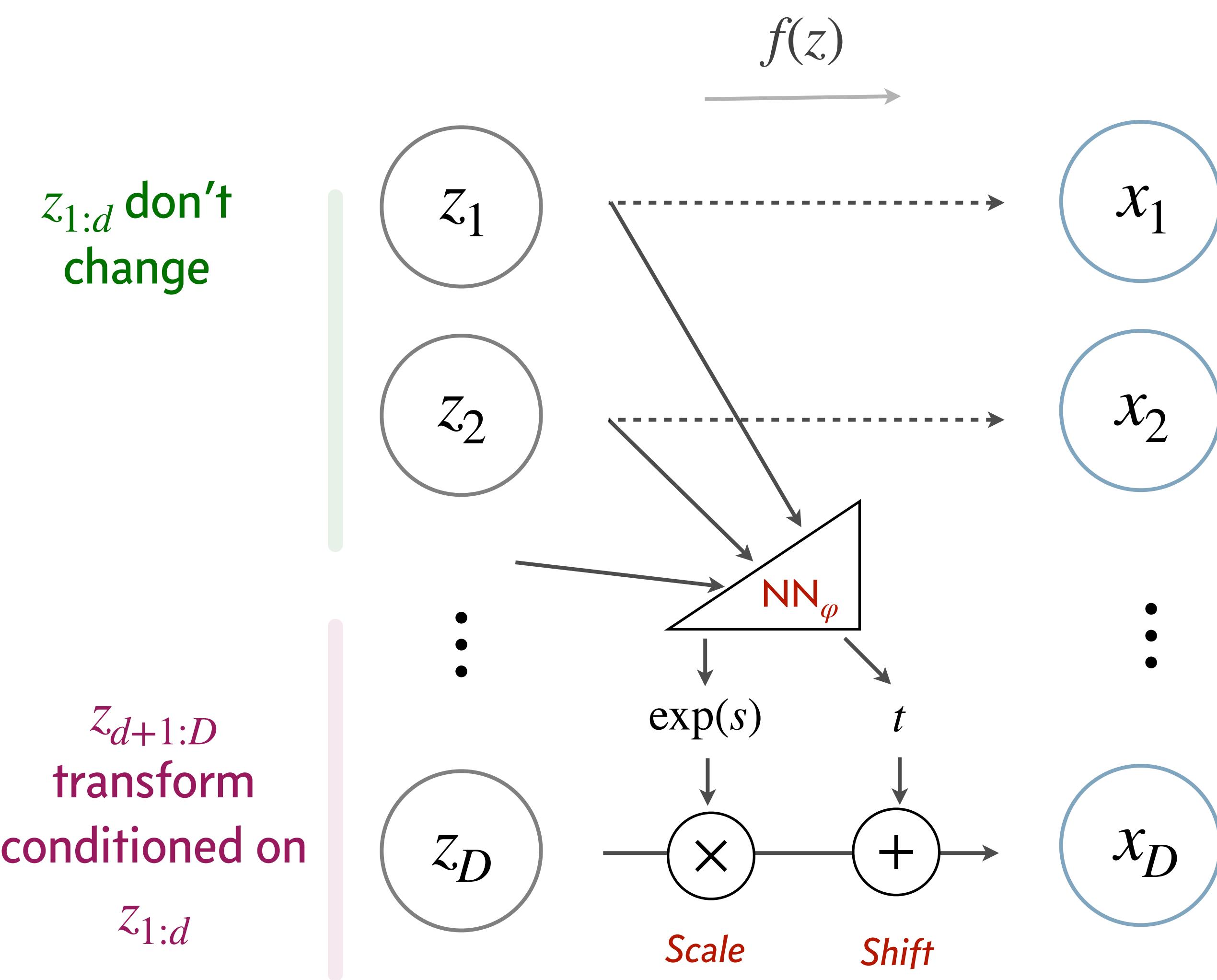
Simple flow transformations

Example: *Affine coupling flow* [RealNVP; Dinh et al 2016]



Simple flow transformations

Example: *Affine coupling flow* [RealNVP; Dinh et al 2016]



Transformation ✓

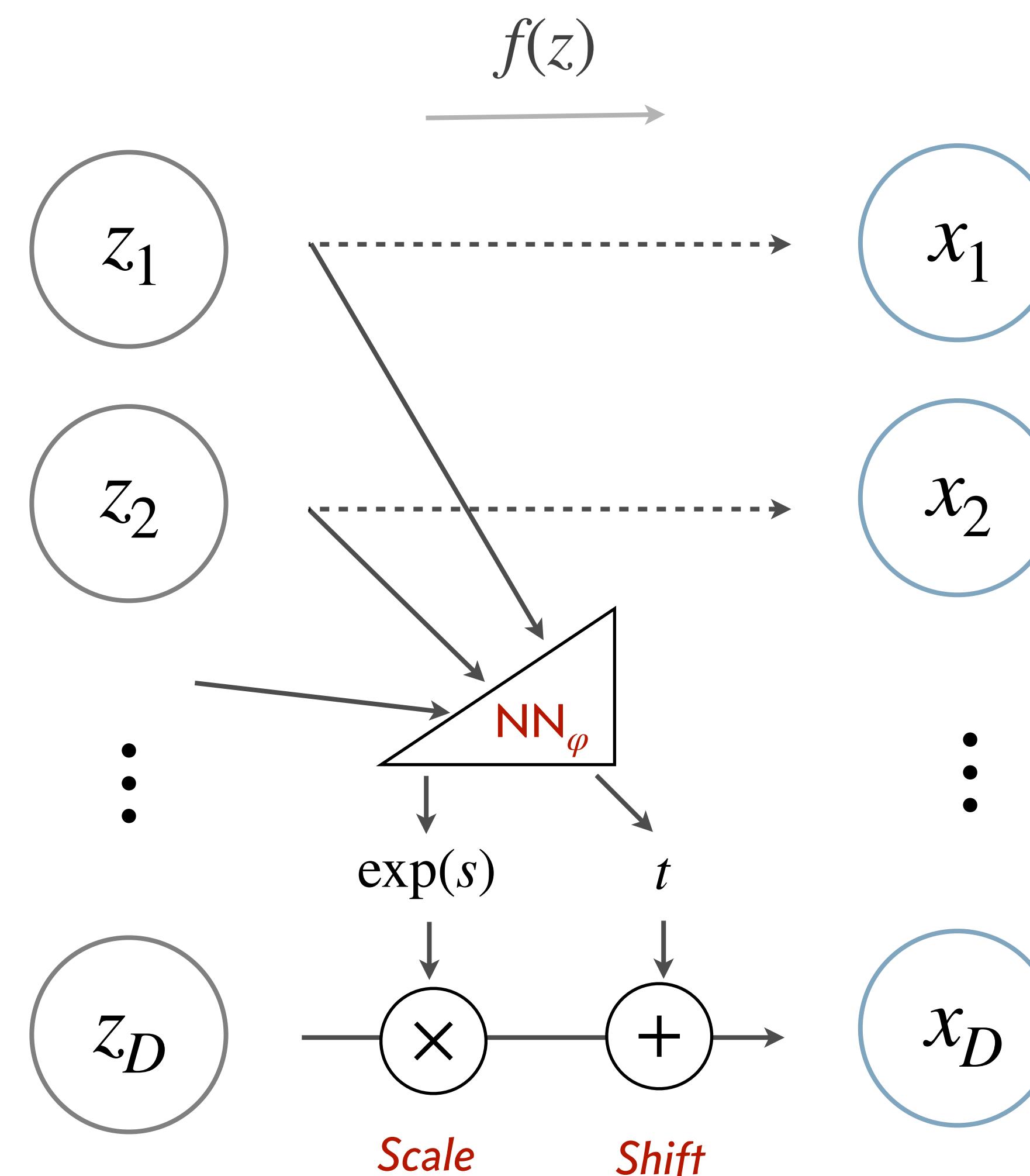
$$x_{d+1:D} = z_{d+1:D} \odot \exp(s(x_{1:d})) + t(x_{1:d})$$

Simple flow transformations

Example: *Affine coupling flow* [RealNVP; Dinh et al 2016]

$z_{1:d}$ don't change

$z_{d+1:D}$ transform conditioned on $z_{1:d}$



Transformation ✓

$$x_{d+1:D} = z_{d+1:D} \odot \exp(s(x_{1:d})) + t(x_{1:d})$$

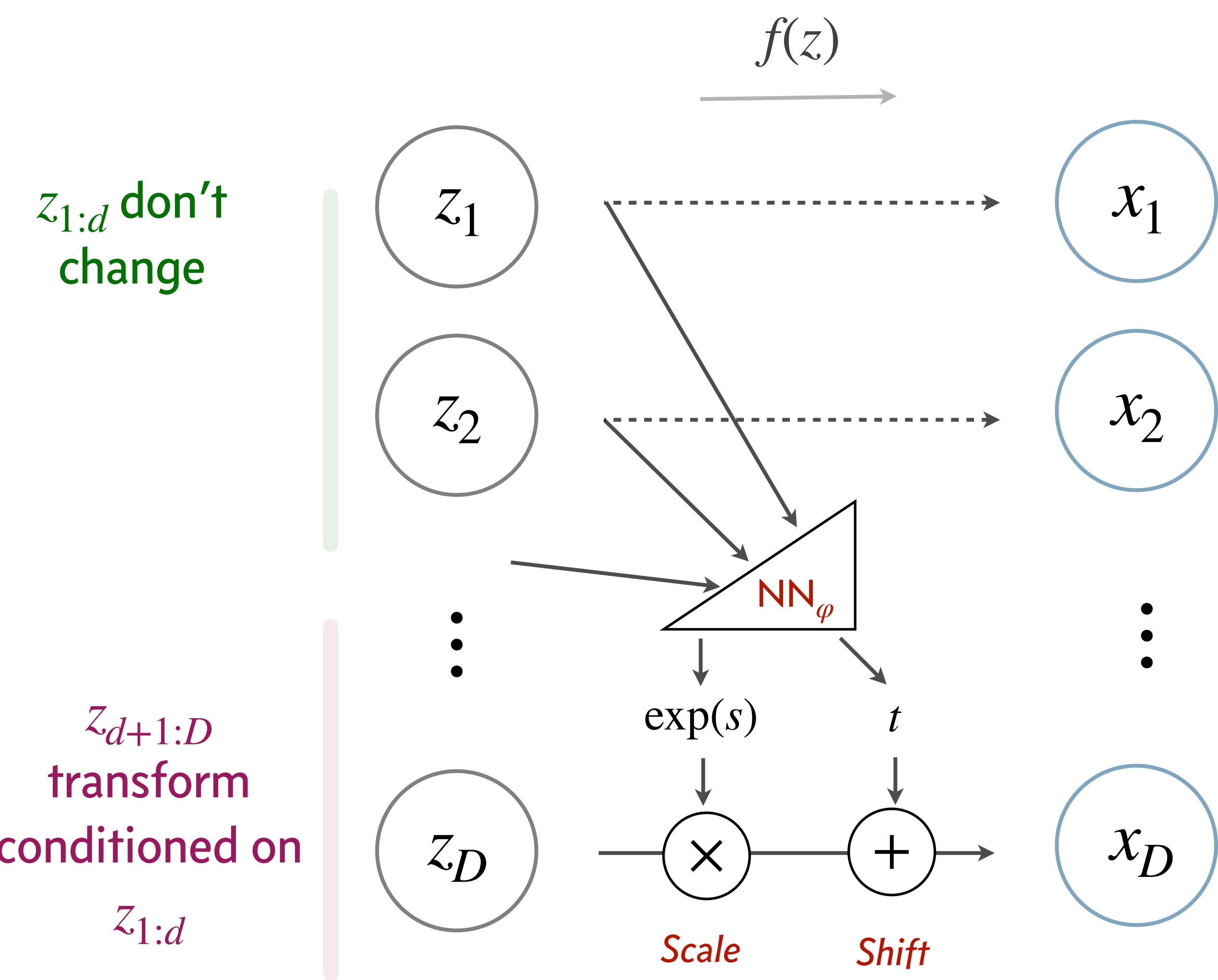


Inverse ✓

$$z_{d+1:D} = (x_{d+1:D} - t(x_{1:d})) \odot \exp(-s(x_{1:d}))$$

Simple flow transformations

Example: *Affine coupling flow* [RealNVP; Dinh et al 2016]



Transformation ✓

$$x_{d+1:D} = z_{d+1:D} \odot \exp\left(s(x_{1:d})\right) + t(x_{1:d})$$



Inverse ✓

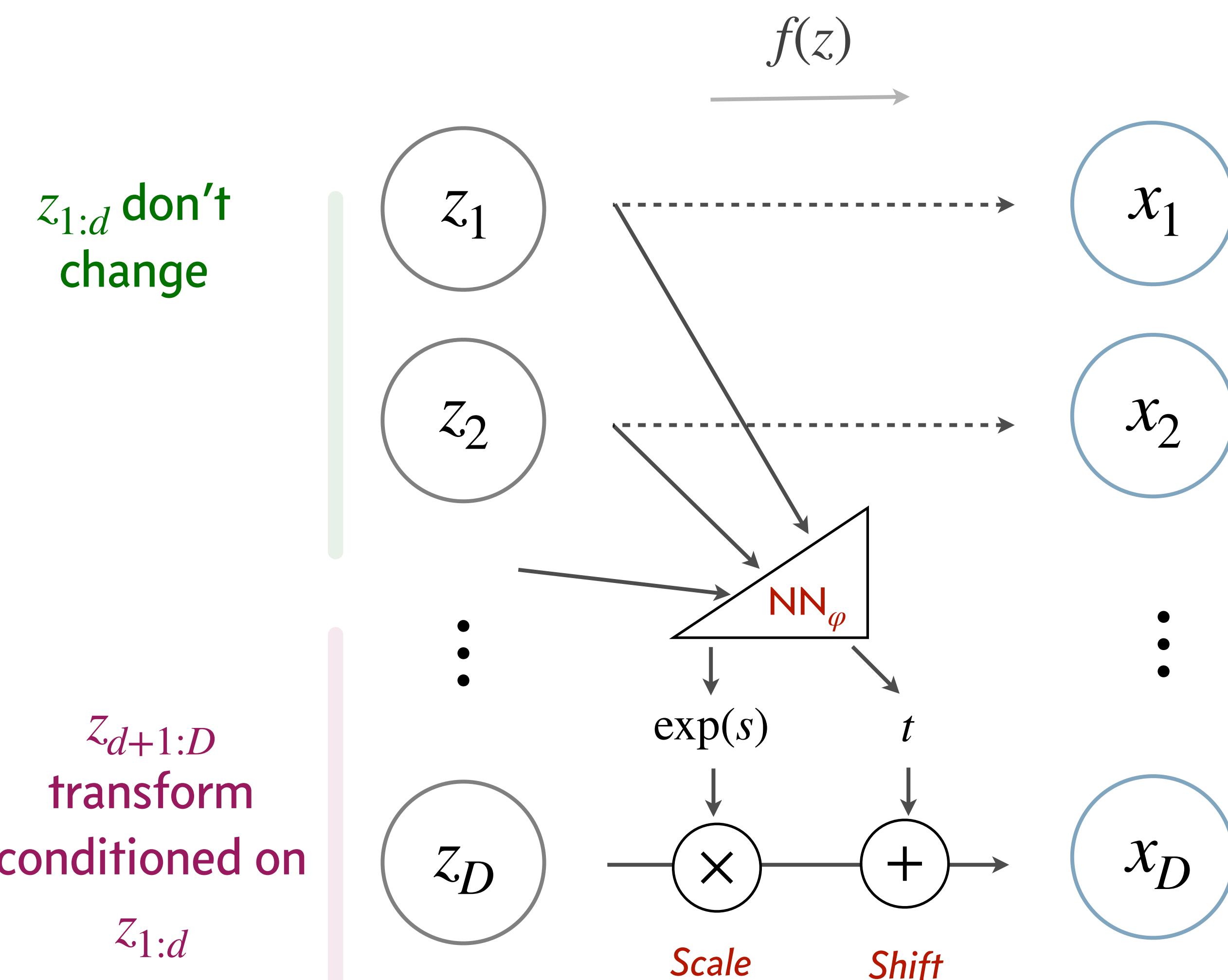
$$z_{d+1:D} = \left(x_{d+1:D} - t(x_{1:d})\right) \odot \exp\left(-s(x_{1:d})\right)$$

Jacobian determinant ✓

$$\det(\nabla f) = \prod_{j=1}^{D-d} \exp\left(s(z_{1:d})\right)_j = \exp\left(\sum_{j=1}^{D-d} s(z_{1:d})_j\right)$$

Simple flow transformations

Example: *Affine coupling flow* [RealNVP; Dinh et al 2016]



Transformation ✓

$$x_{d+1:D} = z_{d+1:D} \odot \exp(s(x_{1:d})) + t(x_{1:d})$$

Inverse ✓

$$z_{d+1:D} = (x_{d+1:D} - t(x_{1:d})) \odot \exp(-s(x_{1:d}))$$

Jacobian determinant ✓

$$\det(\nabla f) = \prod_{j=1}^{D-d} \exp(s(z_{1:d}))_j = \exp\left(\sum_{j=1}^{D-d} s(z_{1:d})_j\right)$$

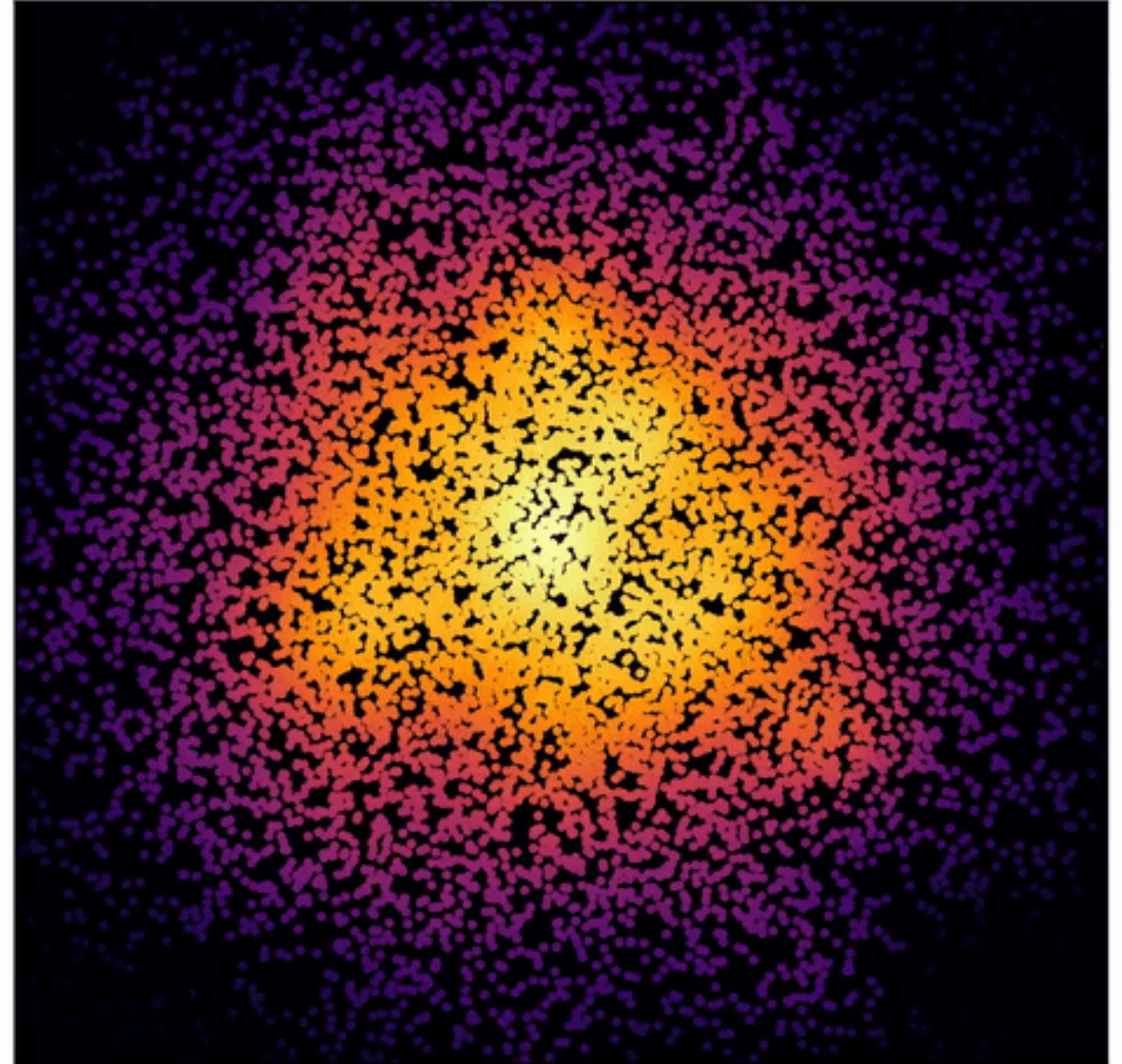
+ Switch up order of transformed variables at every transformation

Continuous-time normalizing flows

Parameterize the transformation by a neural ODE

ODE with reversible dynamics

$$\frac{dx}{dt} = f(x(t))$$

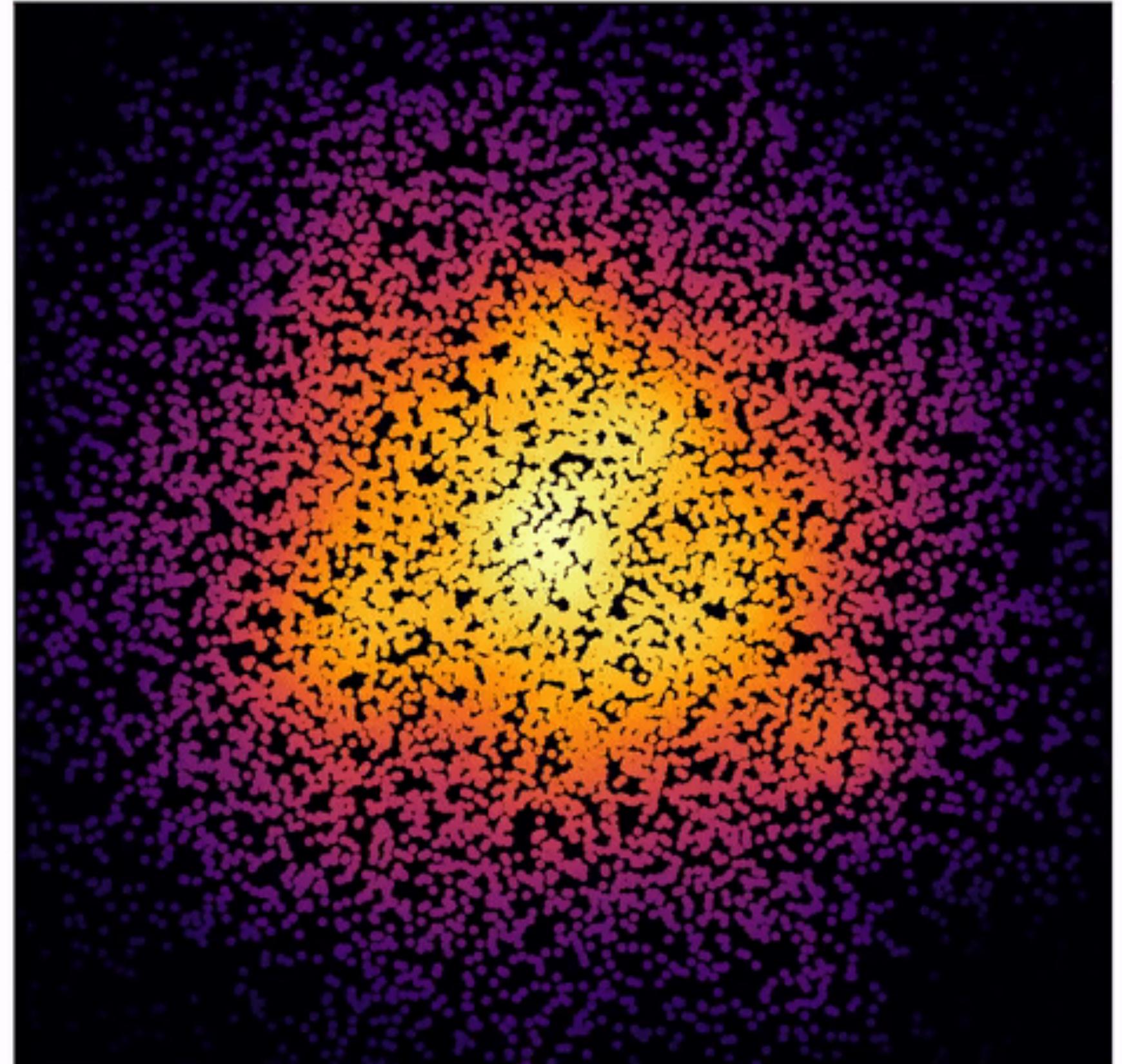


Continuous-time normalizing flows

Parameterize the transformation by a neural ODE

ODE with reversible dynamics

$$\frac{dx}{dt} = f(x(t))$$



Continuous-time normalizing flows

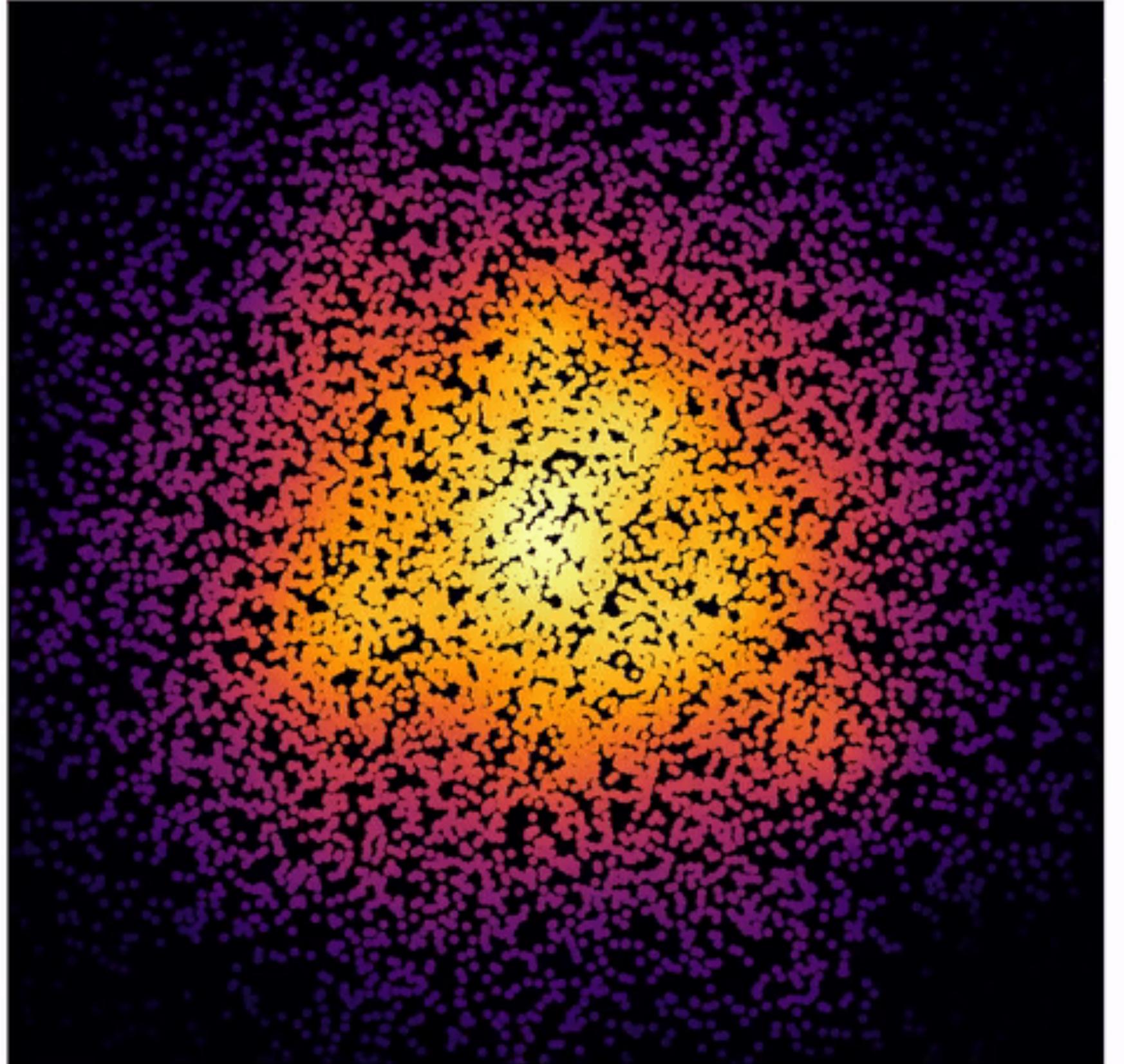
Parameterize the transformation by a neural ODE

ODE with reversible dynamics

$$\frac{dx}{dt} = f(x(t))$$

Instantaneous change-of-variable formula

$$\frac{d \log p(x(t))}{dt} = - \text{Tr} \left(\frac{df}{dx(t)} \right)$$



Continuous-time normalizing flows

Parameterize the transformation by a neural ODE

ODE with reversible dynamics

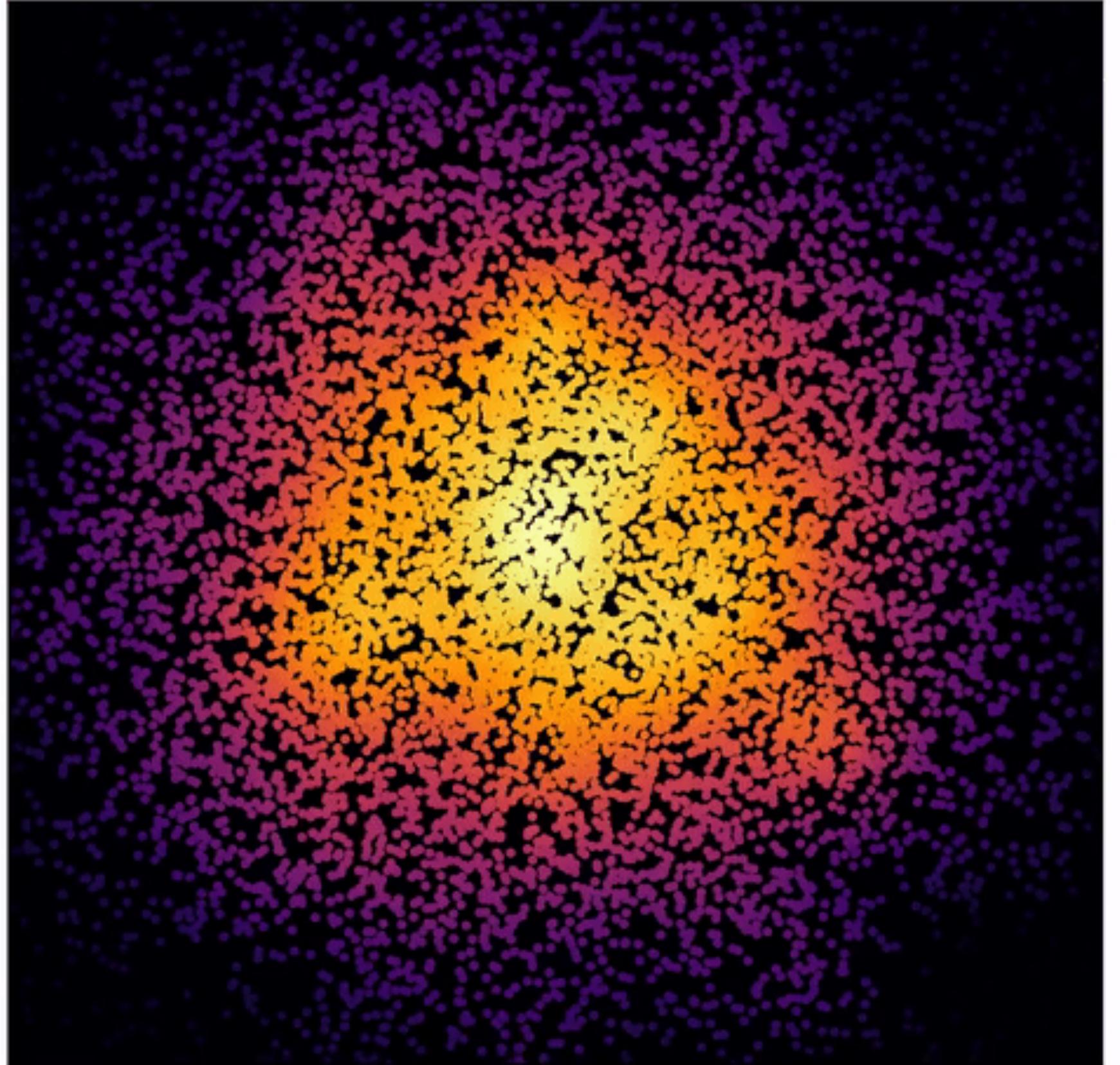
$$\frac{dx}{dt} = f(x(t))$$

Instantaneous change-of-variable formula

$$\frac{d \log p(x(t))}{dt} = - \text{Tr} \left(\frac{df}{dx(t)} \right)$$

Pro ✓

Unrestricted form of transformation $f(x)$!



Continuous-time normalizing flows

Parameterize the transformation by a neural ODE

ODE with reversible dynamics

$$\frac{dx}{dt} = f(x(t))$$

Instantaneous change-of-variable formula

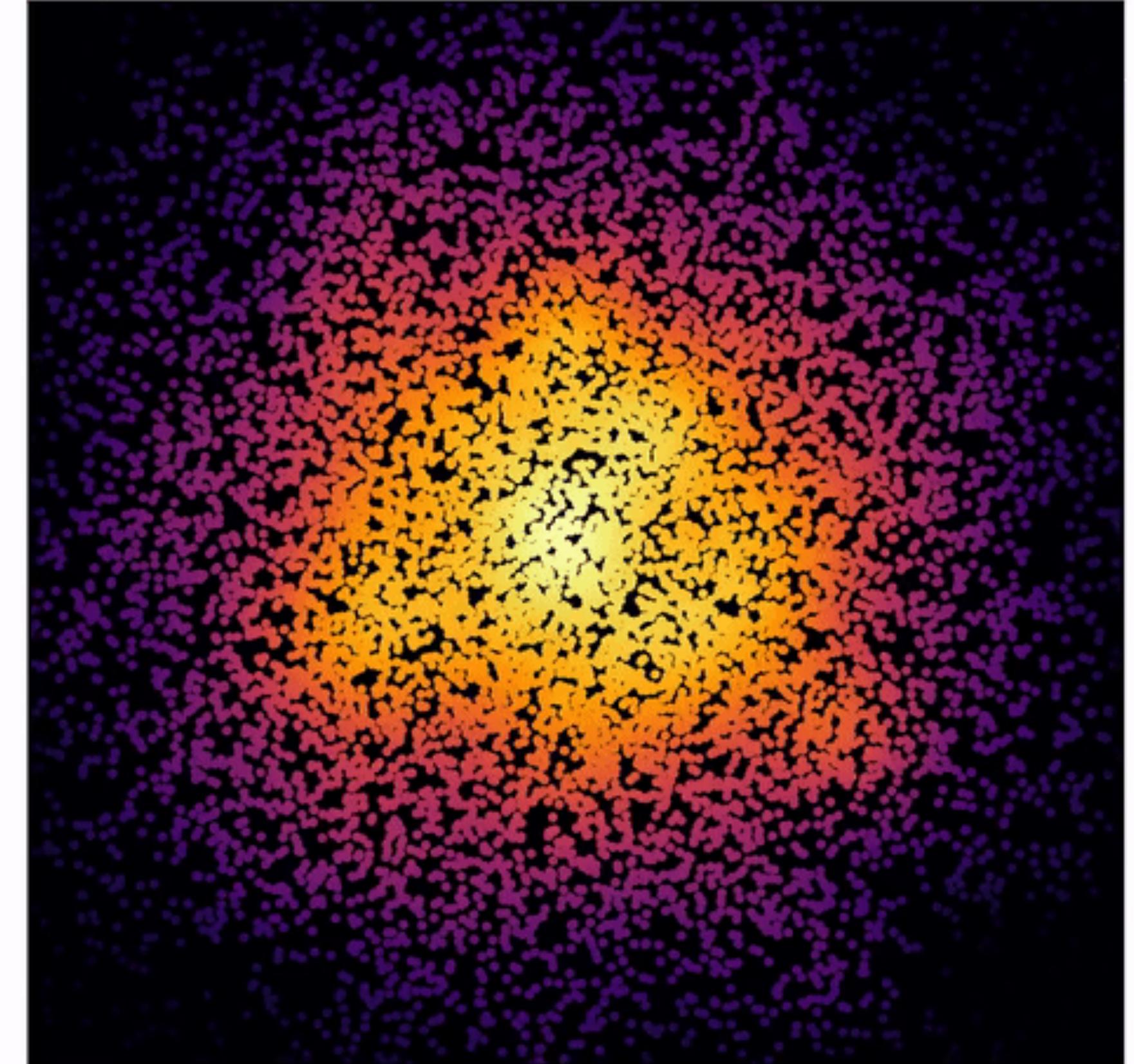
$$\frac{d \log p(x(t))}{dt} = - \text{Tr} \left(\frac{df}{dx(t)} \right)$$

Pro ✓

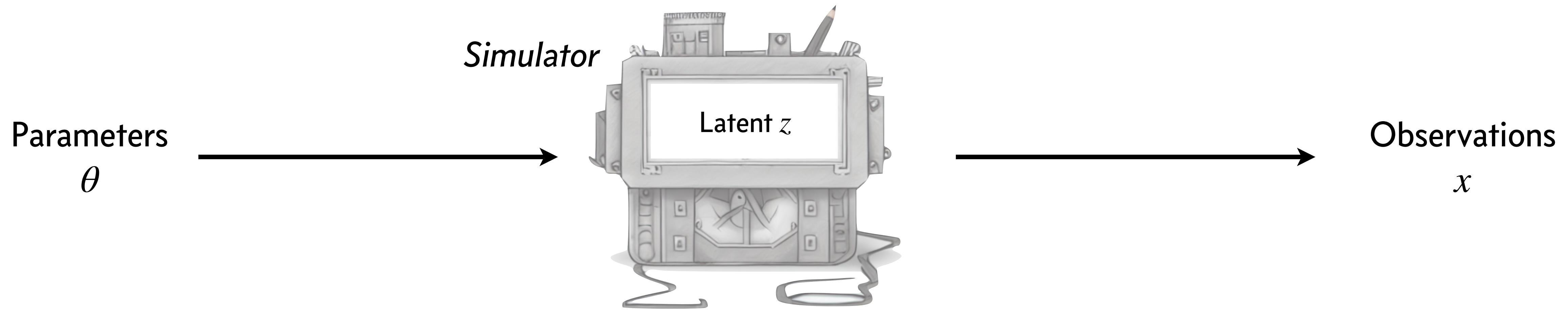
Unrestricted form of transformation $f(x)$!

Cons ☹

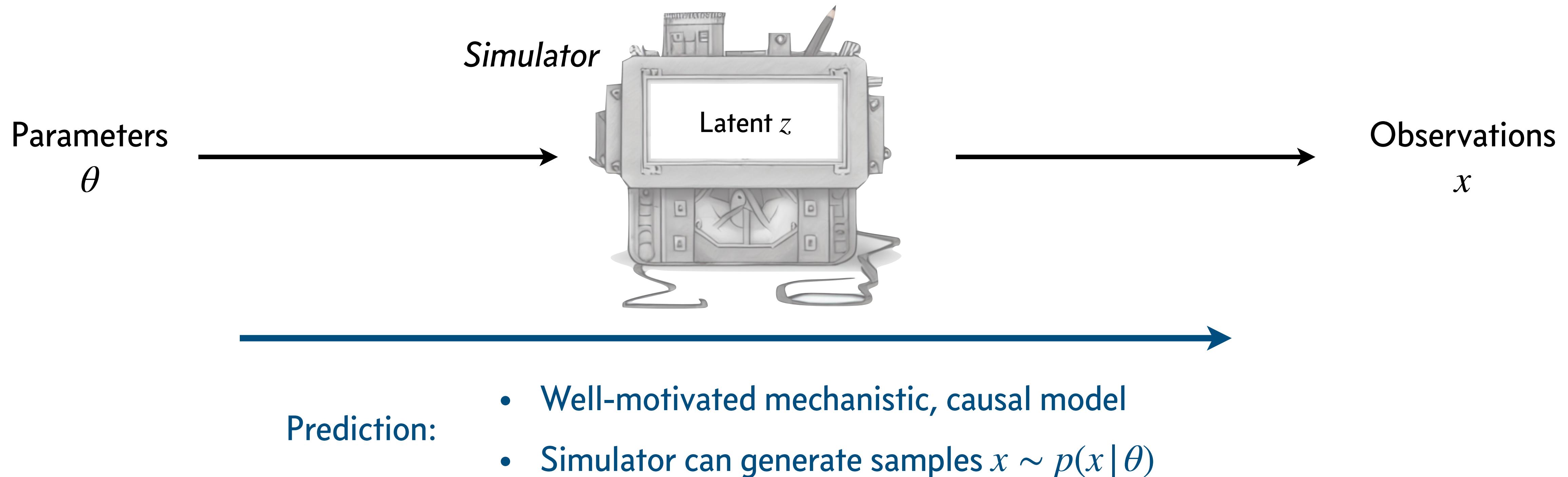
- Need for efficient trace calculation
- Solving an ODE and backpropping through the solution can make for cumbersome training



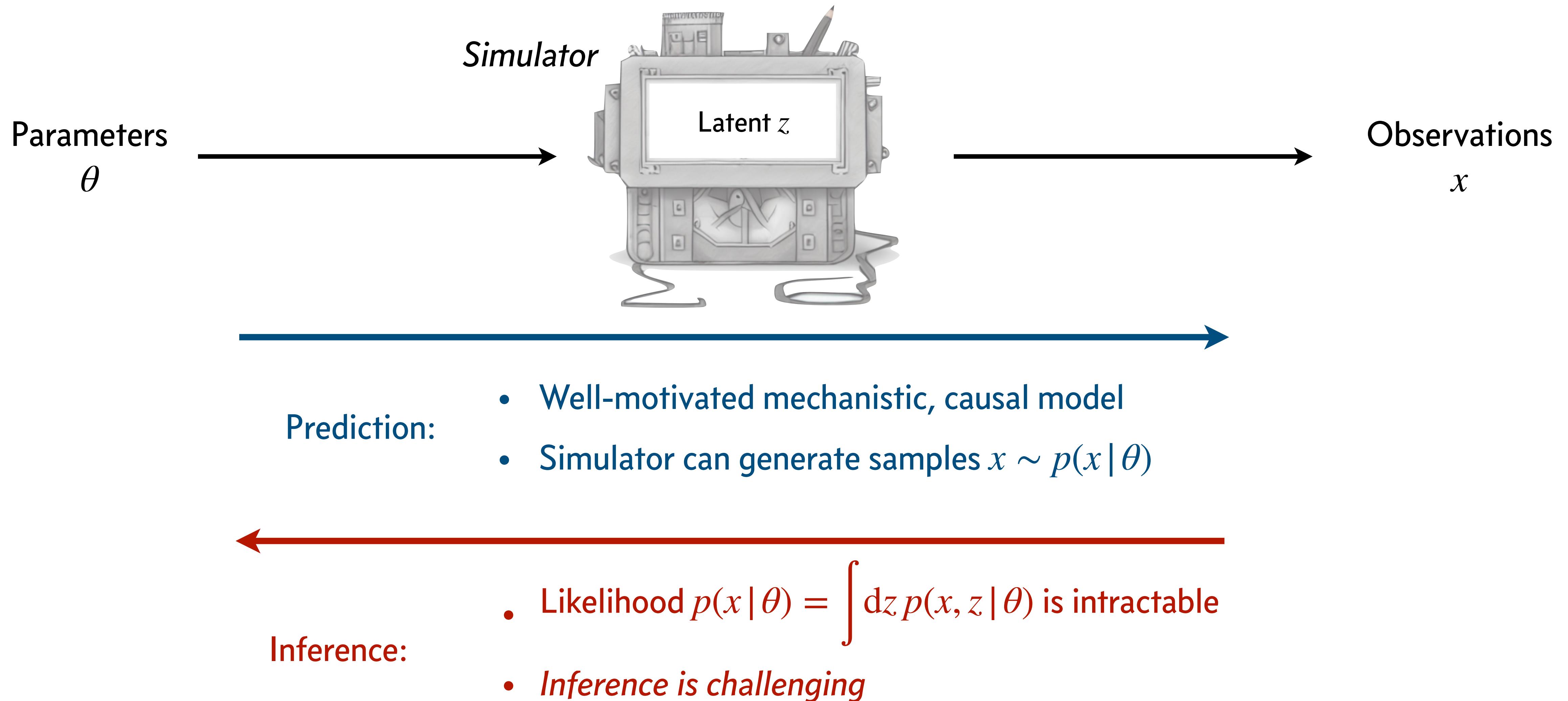
Simulation-based inference (SBI)



Simulation-based inference (SBI)

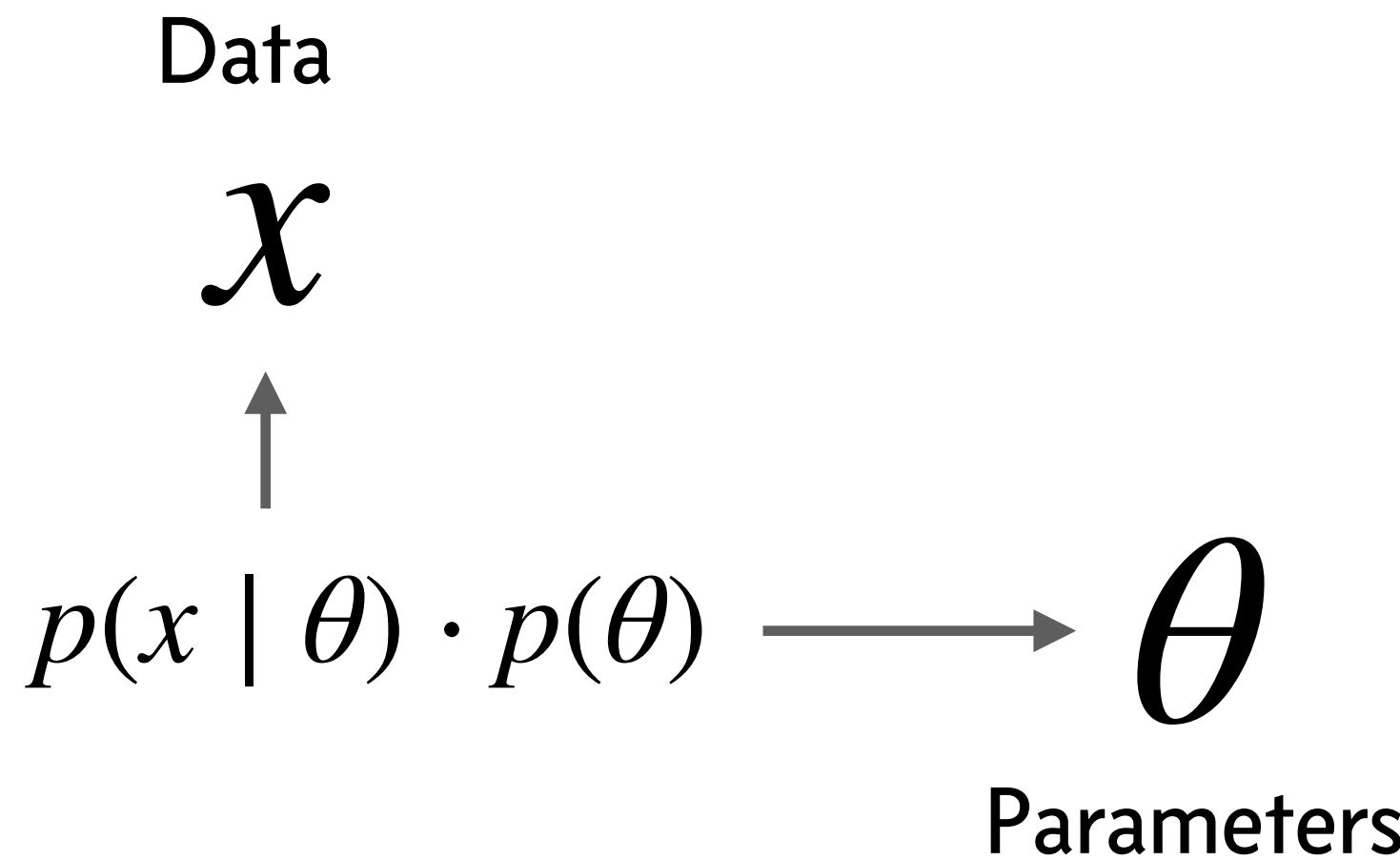


Simulation-based inference (SBI)



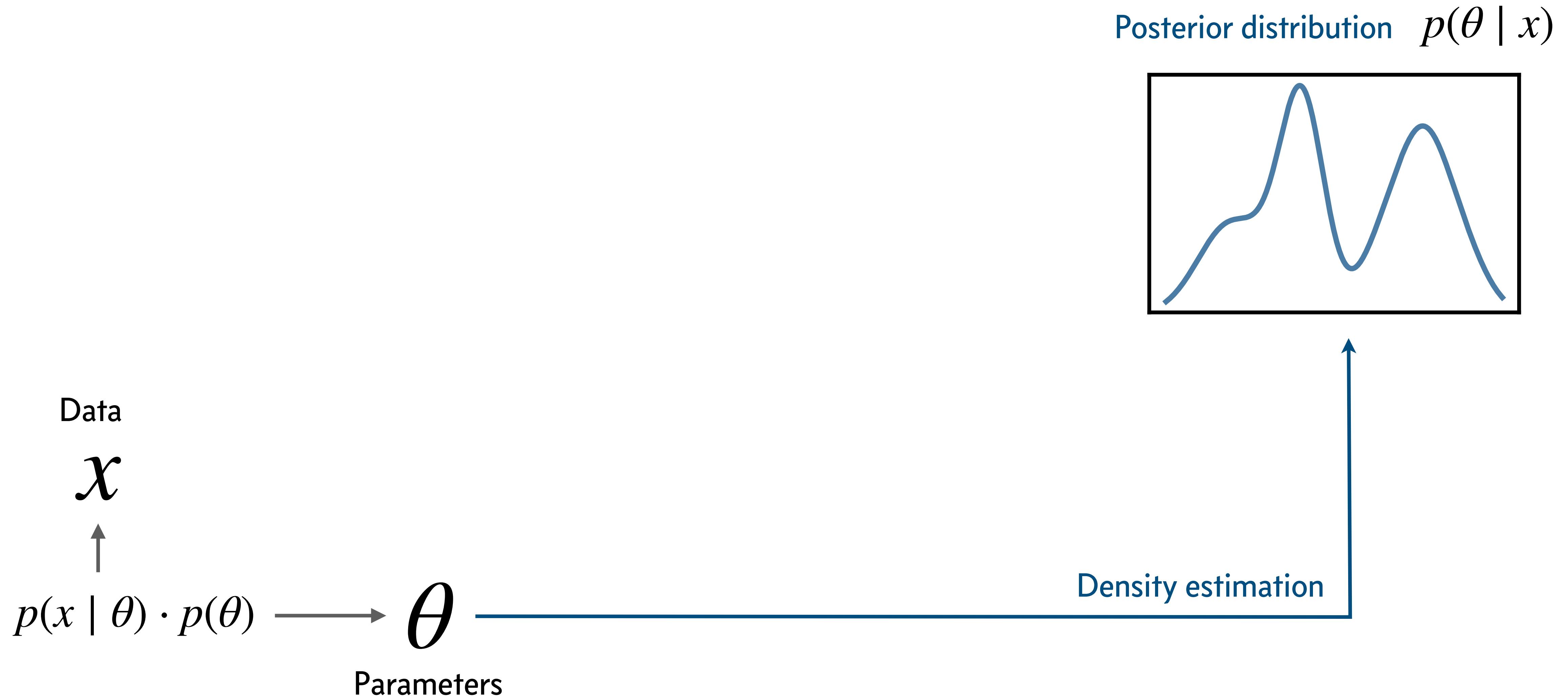
Flows in *simulation-based inference*

Flows are commonly employed as *conditional posterior density estimators* in simulation-based inference



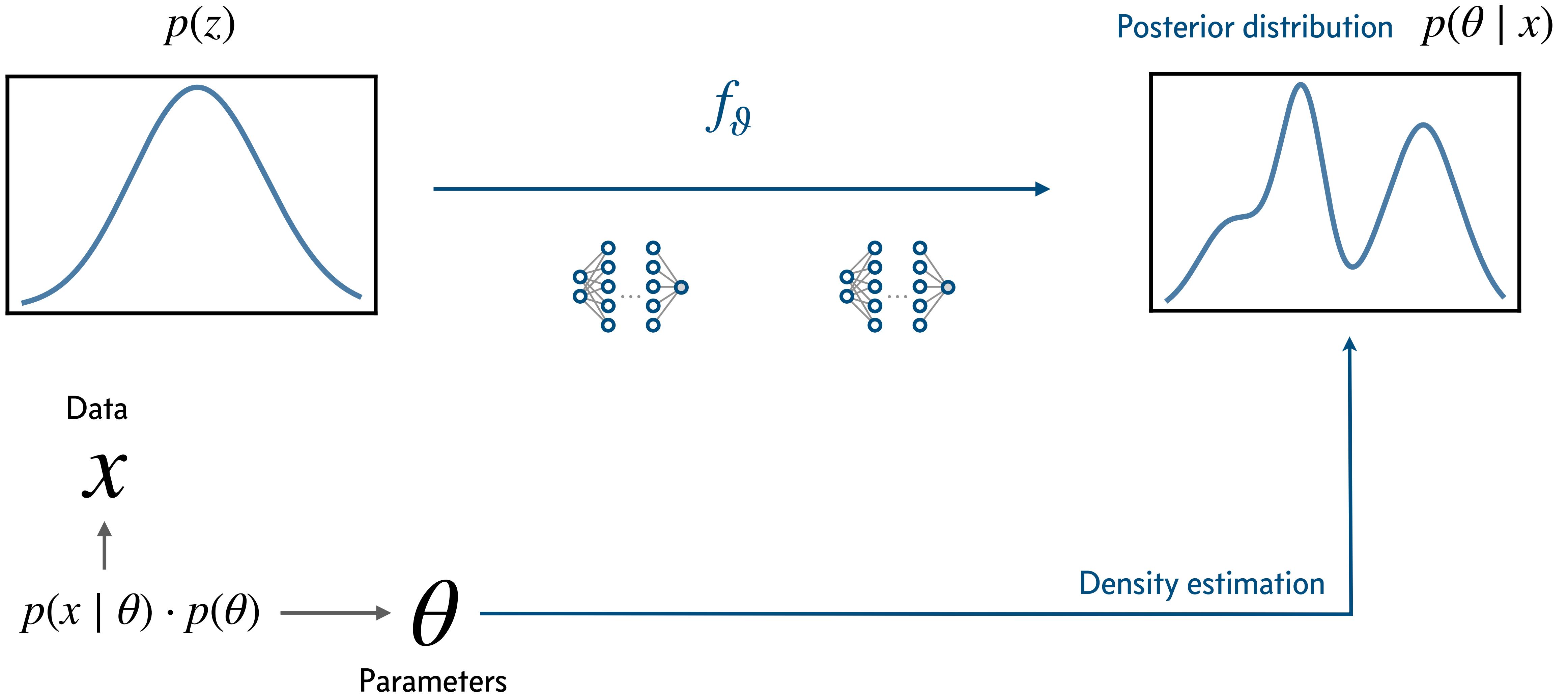
Flows in simulation-based inference

Flows are commonly employed as *conditional posterior density estimators* in simulation-based inference



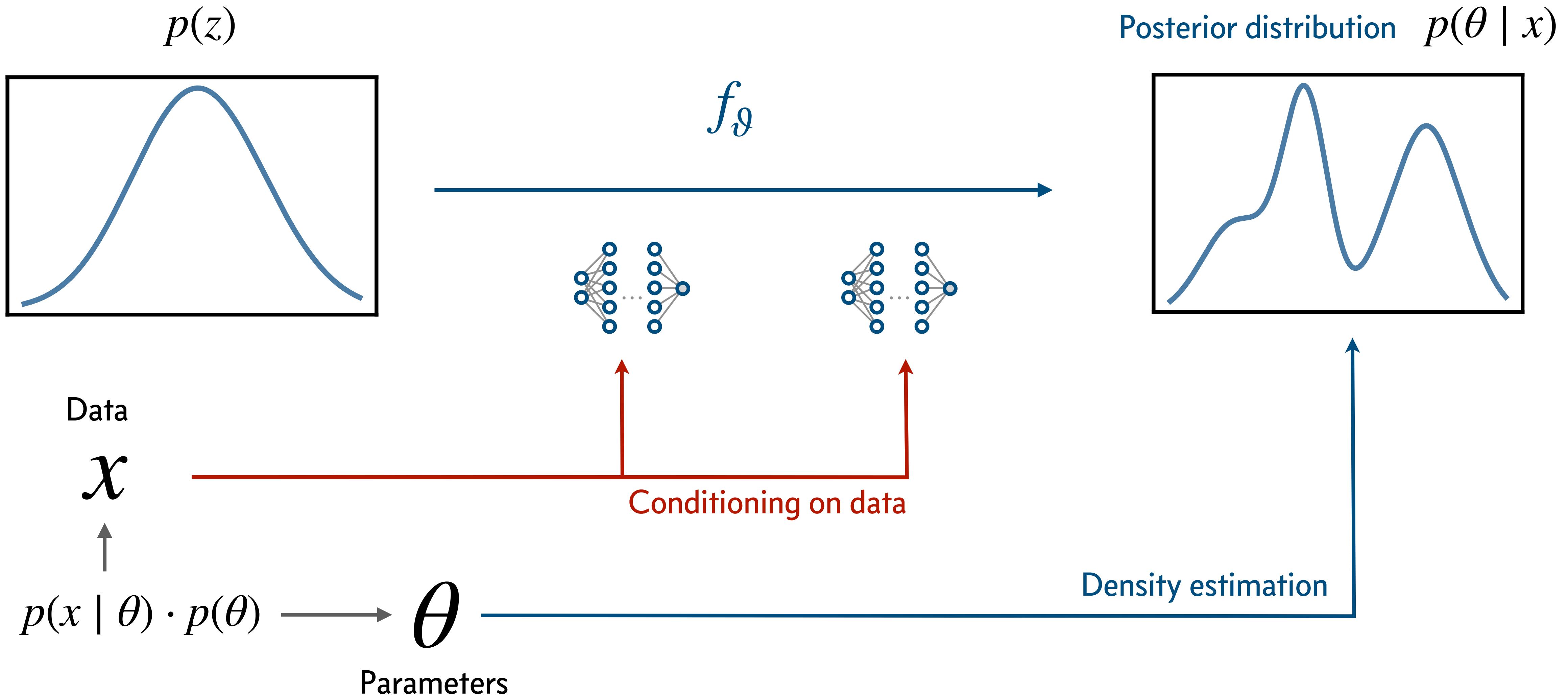
Flows in simulation-based inference

Flows are commonly employed as *conditional posterior density estimators* in simulation-based inference



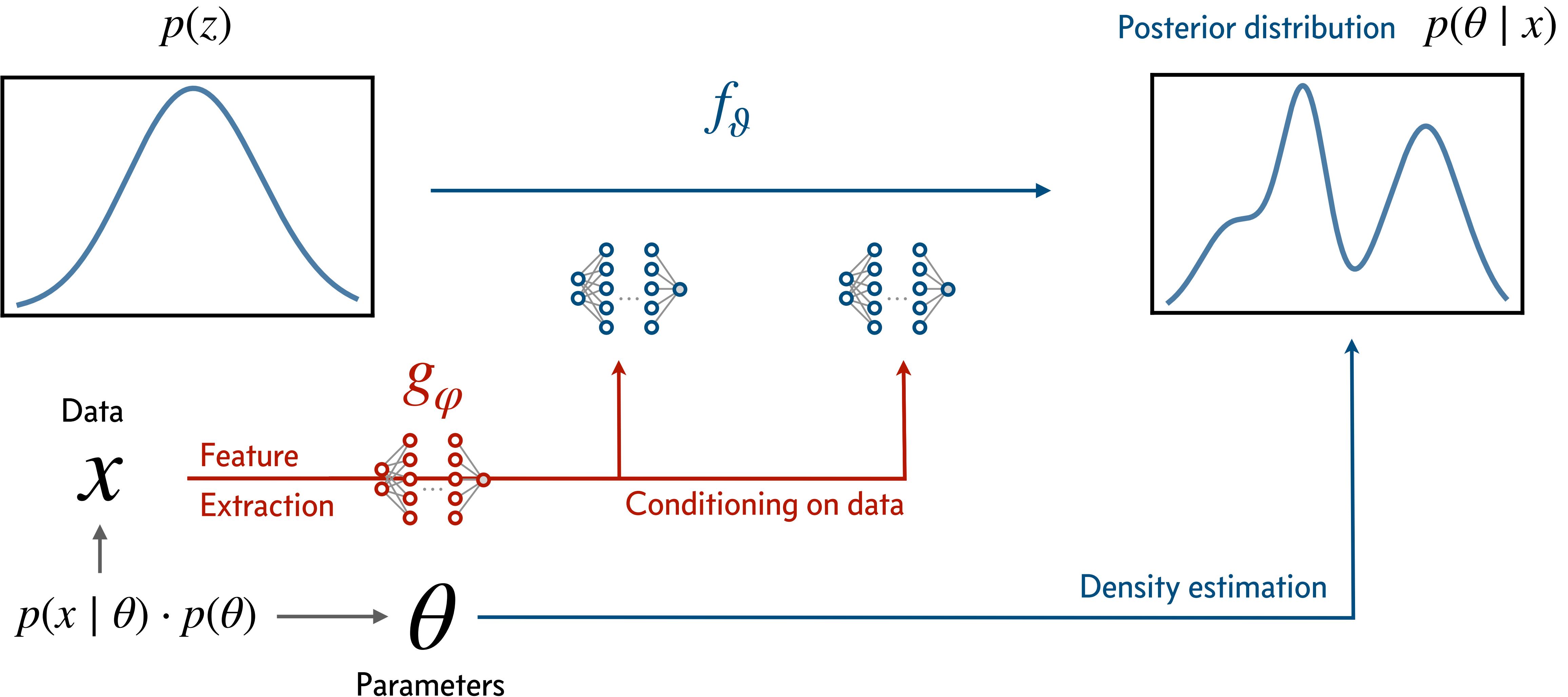
Flows in simulation-based inference

Flows are commonly employed as *conditional posterior density estimators* in simulation-based inference



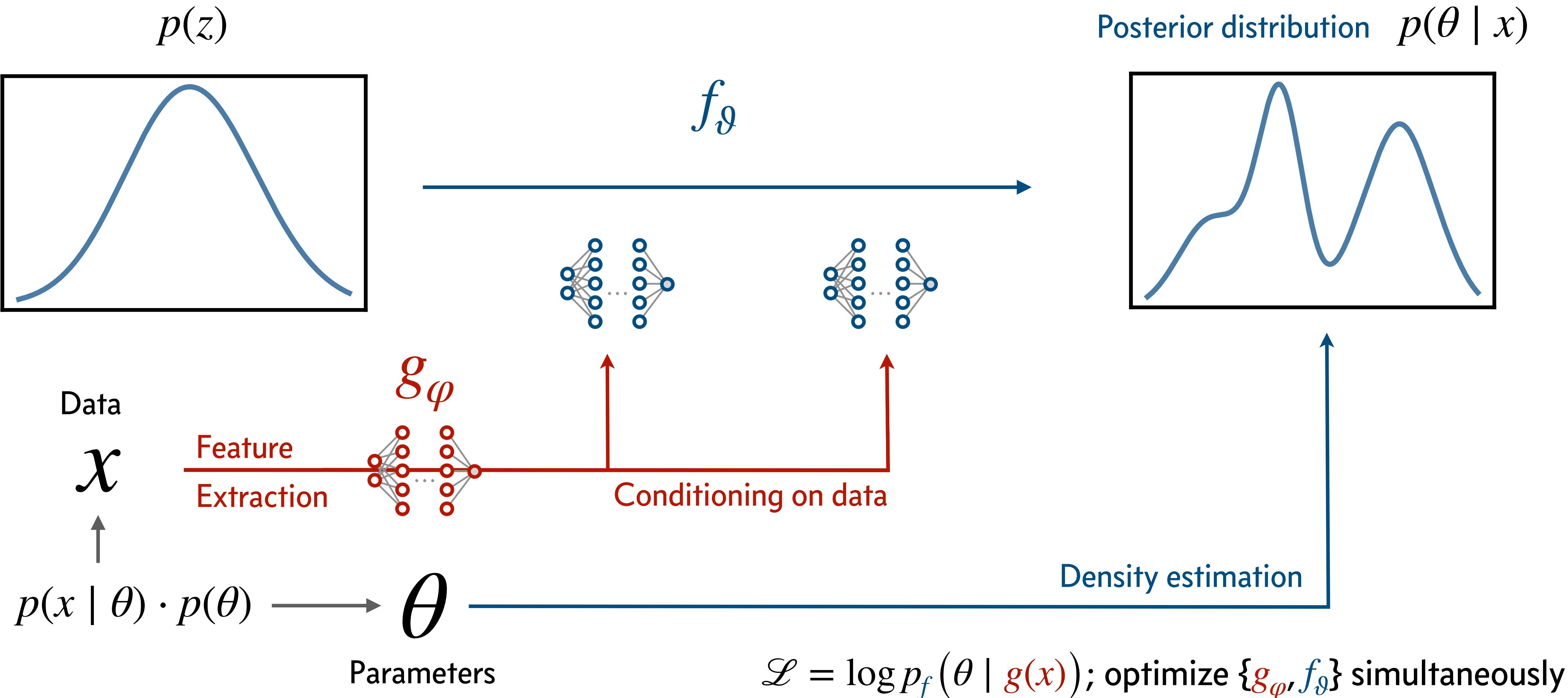
Flows in simulation-based inference

Flows are commonly employed as *conditional posterior density estimators* in simulation-based inference

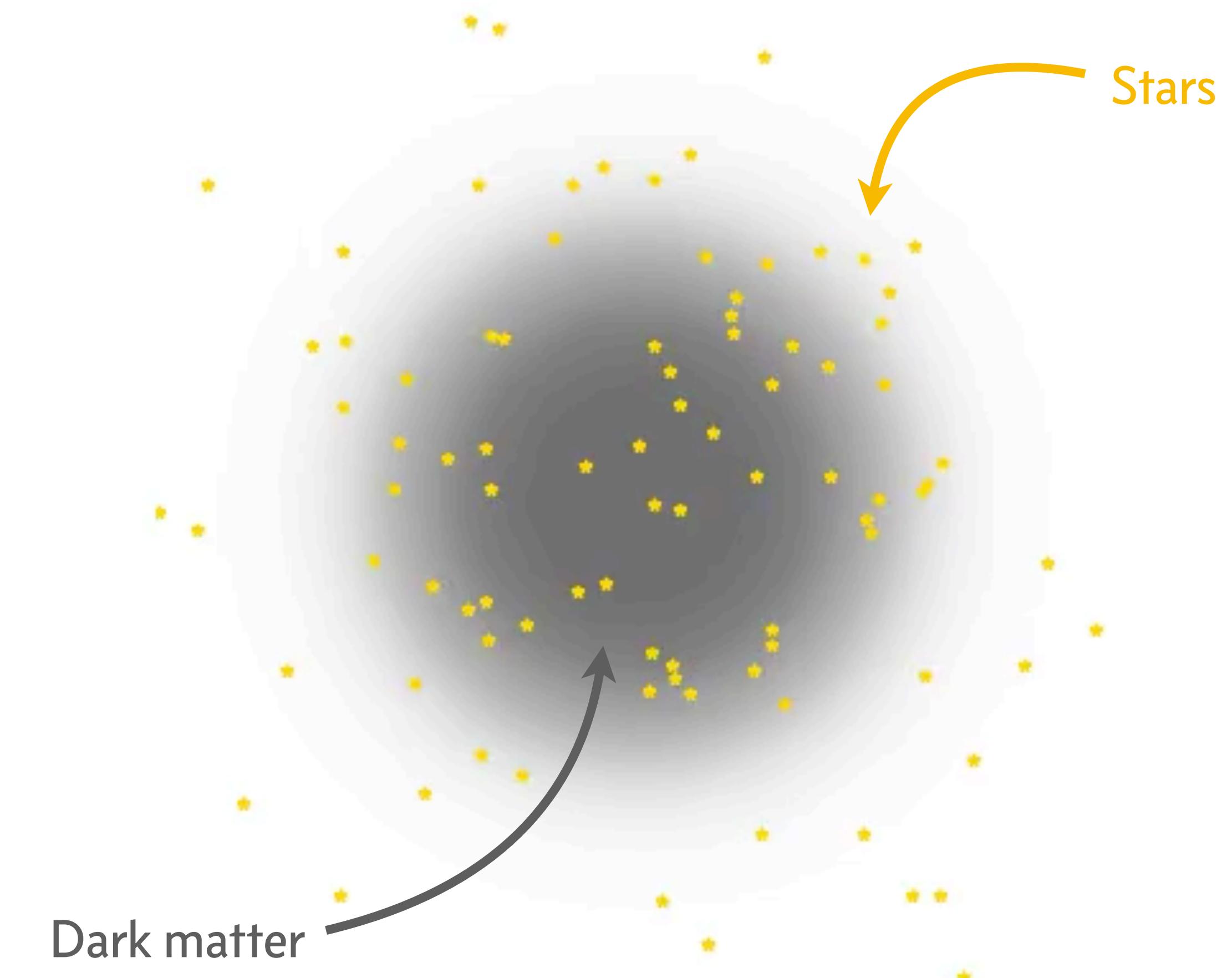


Flows in simulation-based inference

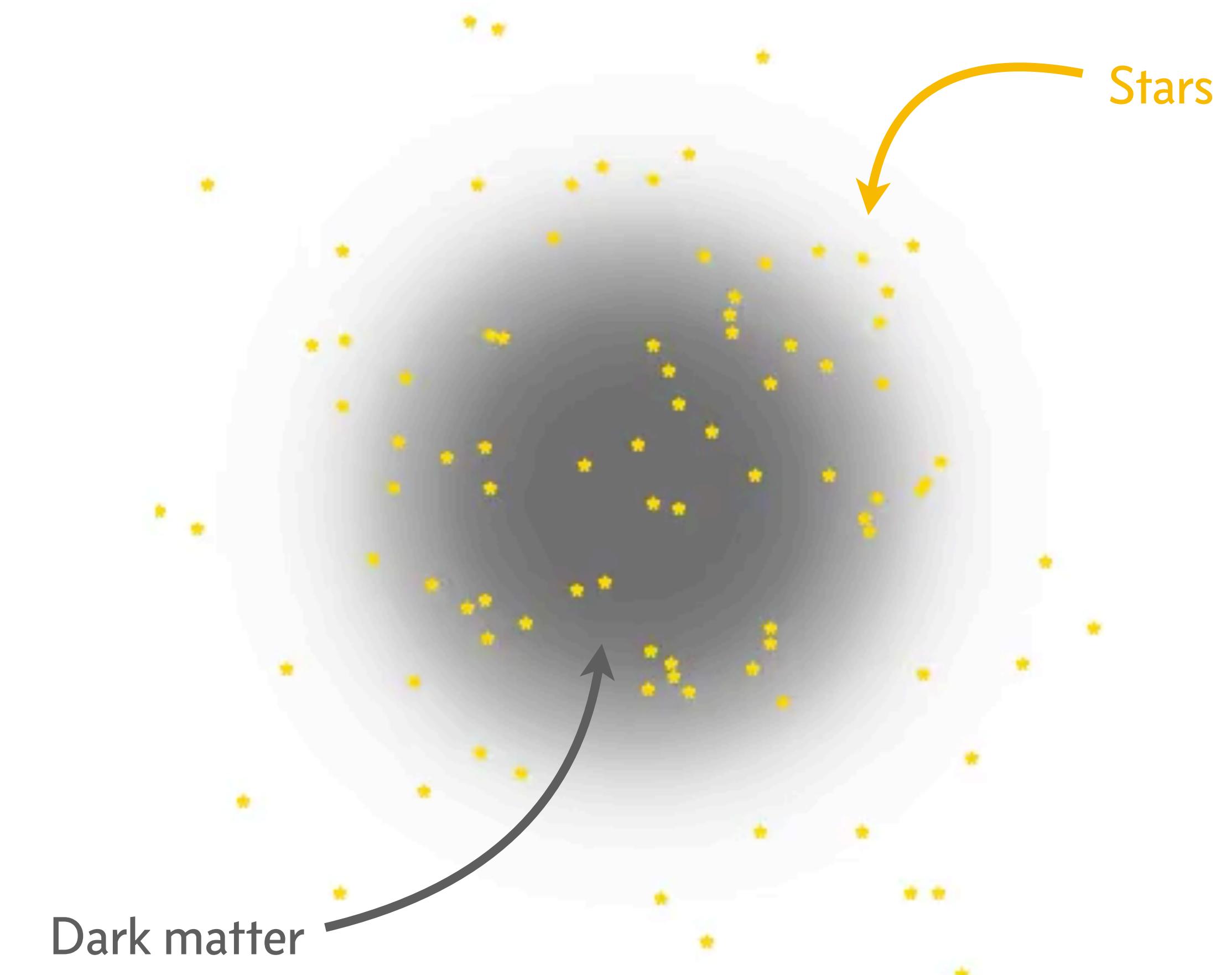
Flows are commonly employed as *conditional posterior density estimators* in simulation-based inference



An application to *dwarf galaxies*



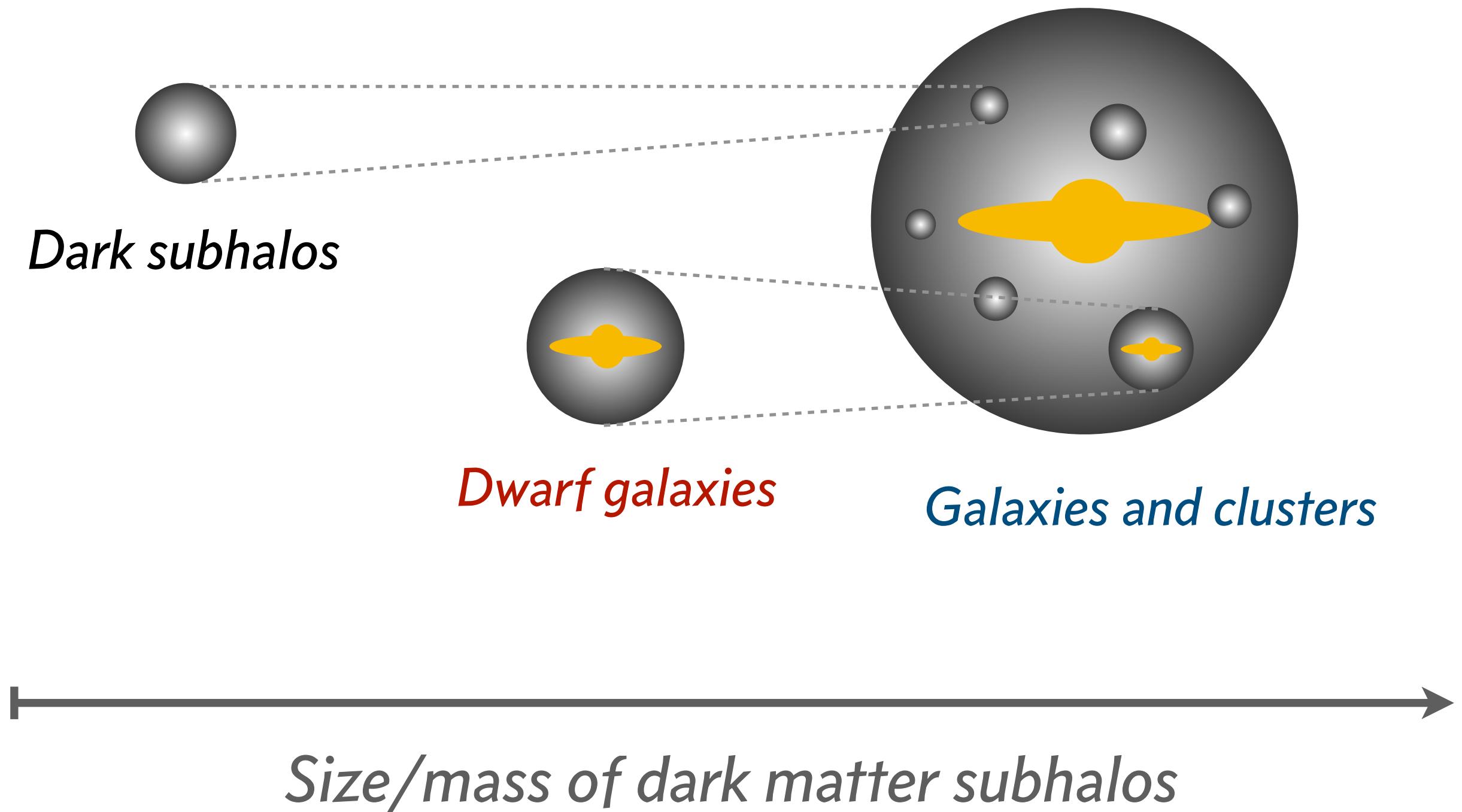
An application to *dwarf galaxies*



Another application: extracting the dark matter distribution from dwarf galaxies

Dwarf galaxies are intermediate-sized galaxies well-suited for studying dark matter

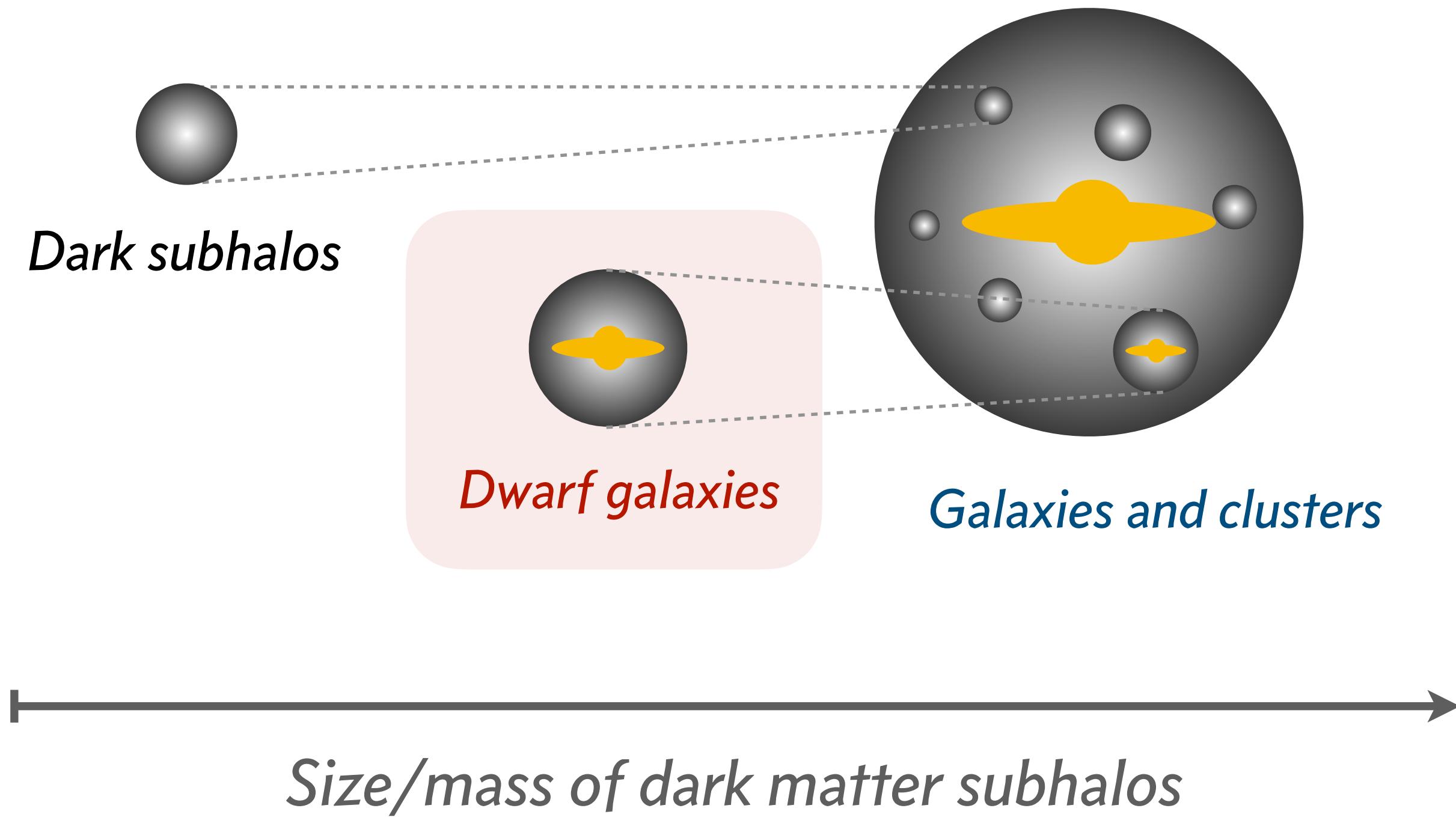
[Nguyen, SM et al 2022]



Another application: extracting the dark matter distribution from dwarf galaxies

Dwarf galaxies are intermediate-sized galaxies well-suited for studying dark matter

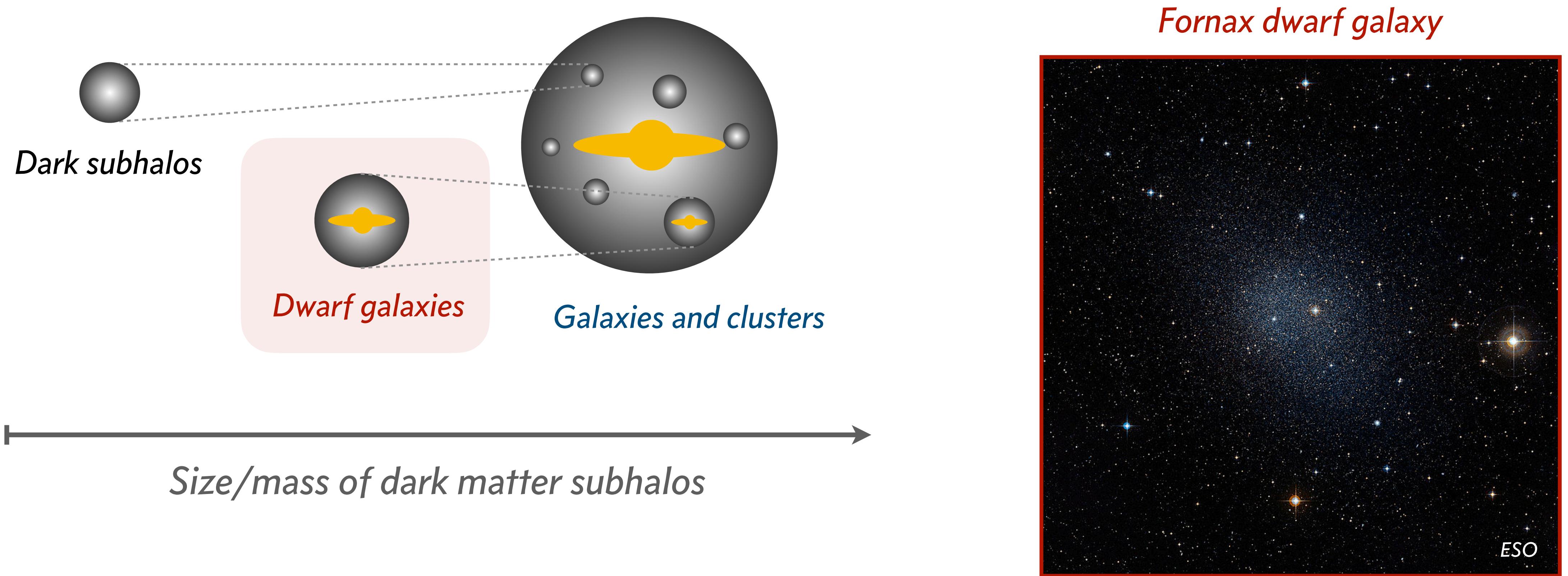
[Nguyen, SM et al 2022]



Another application: extracting the dark matter distribution from dwarf galaxies

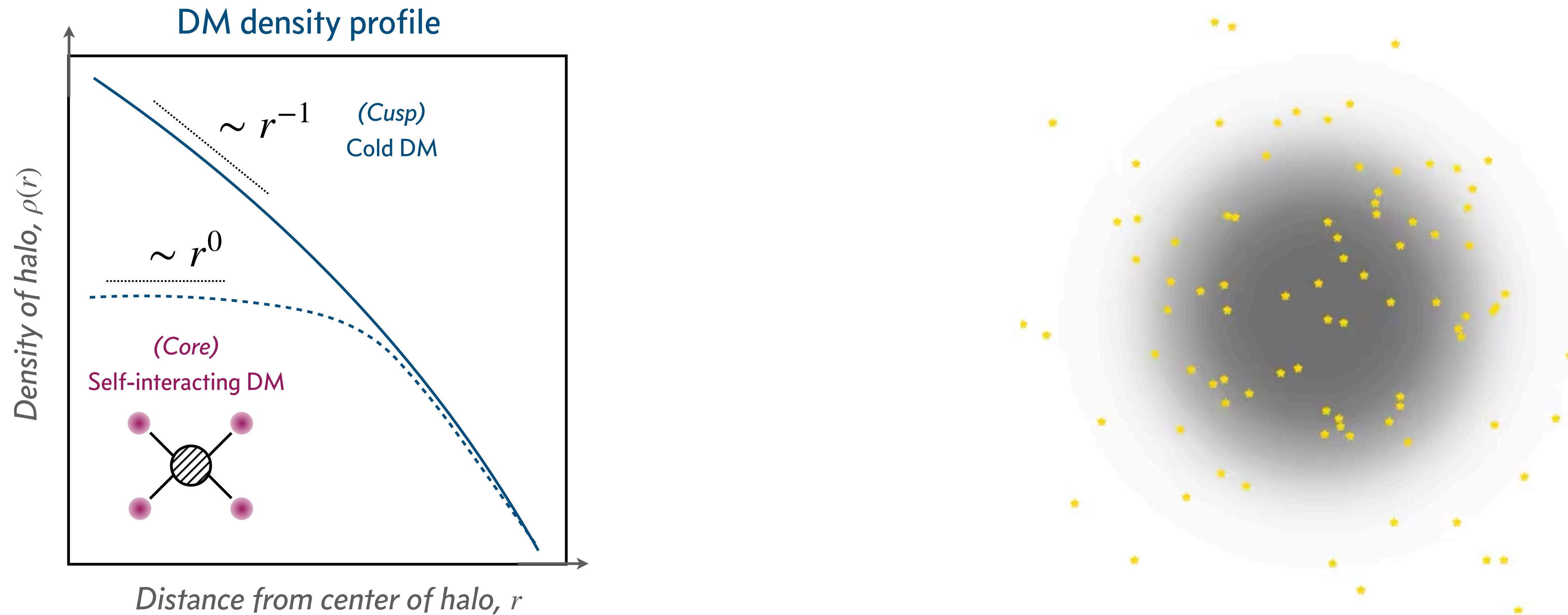
Dwarf galaxies are intermediate-sized galaxies well-suited for studying dark matter

[Nguyen, SM et al 2022]



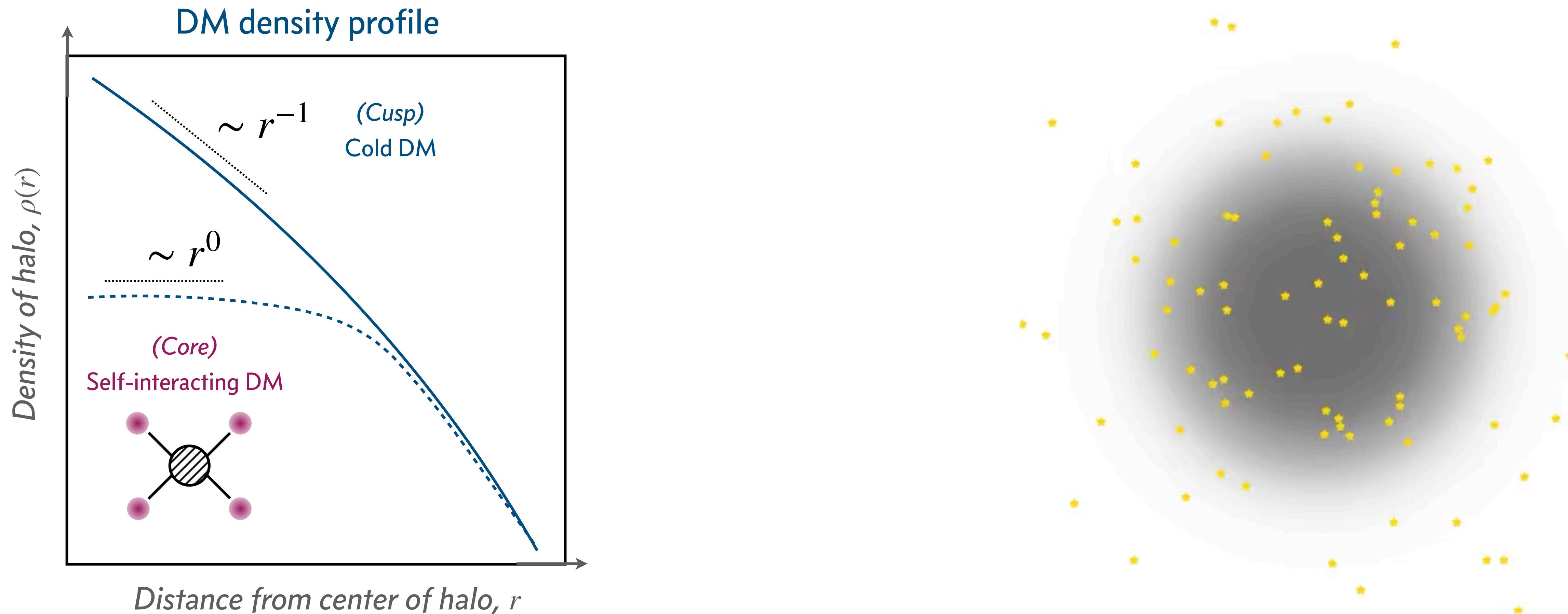
Extracting the dark matter distribution from dwarf galaxies

[Nguyen, SM et al 2022]



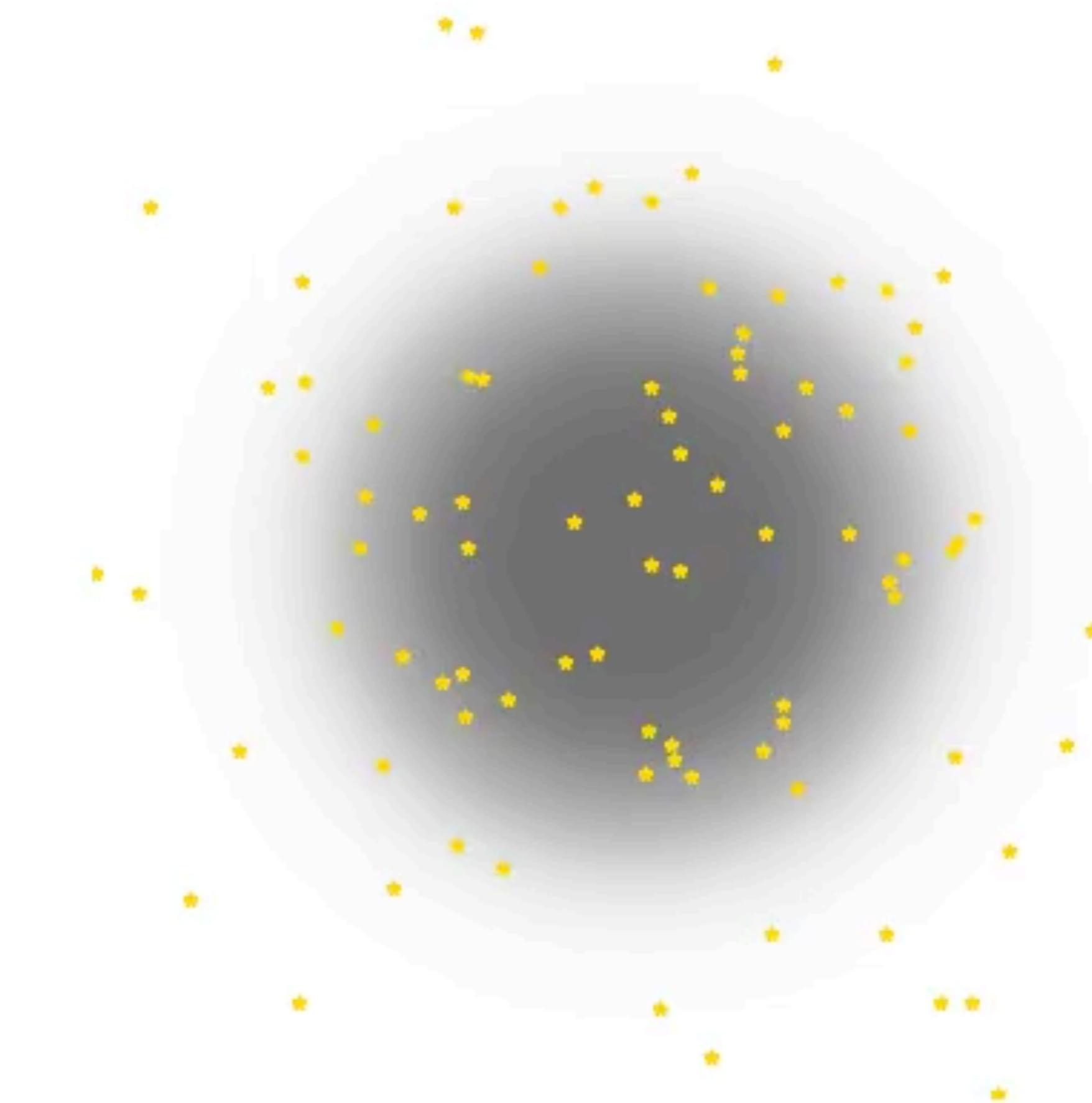
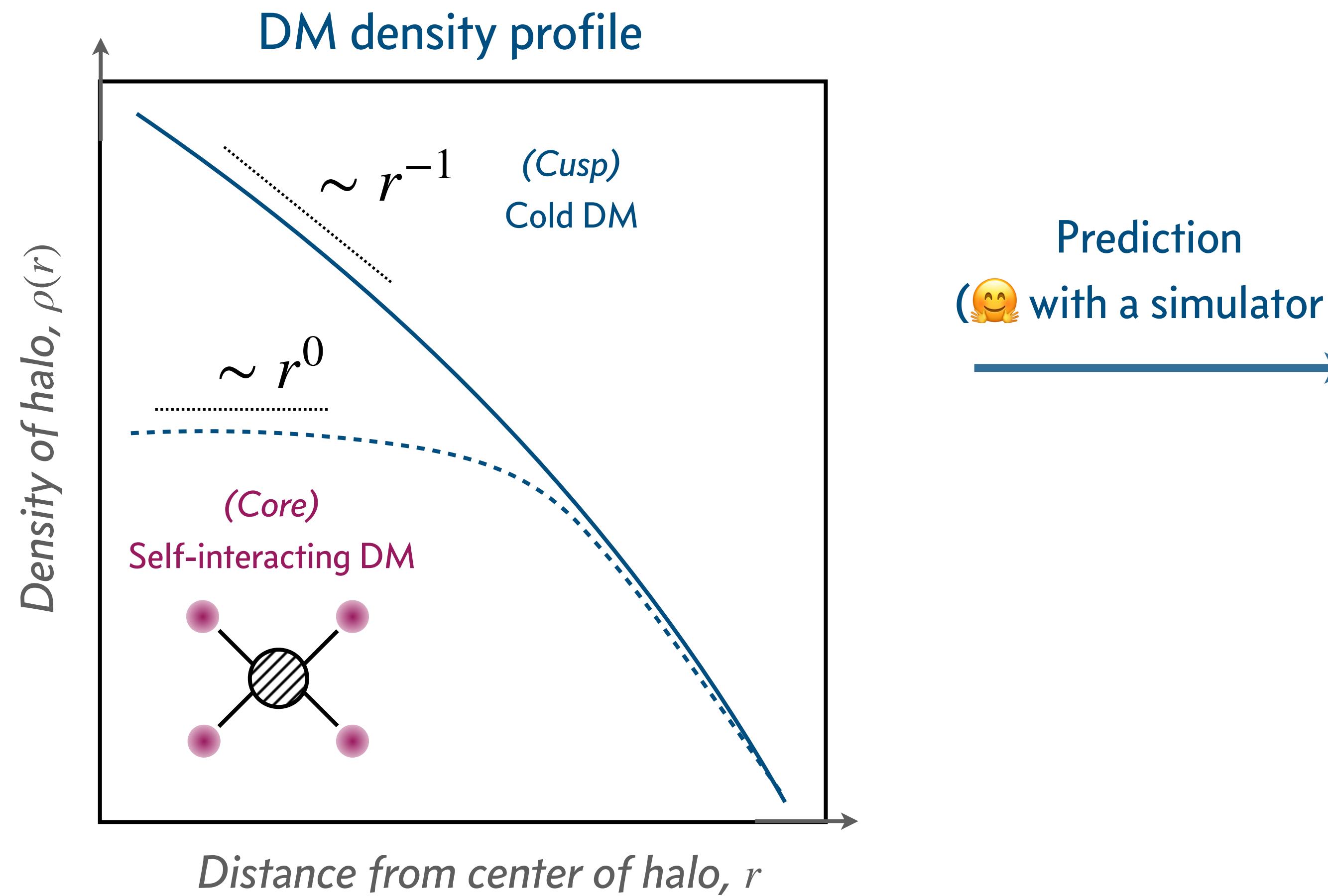
Extracting the dark matter distribution from dwarf galaxies

[Nguyen, SM et al 2022]



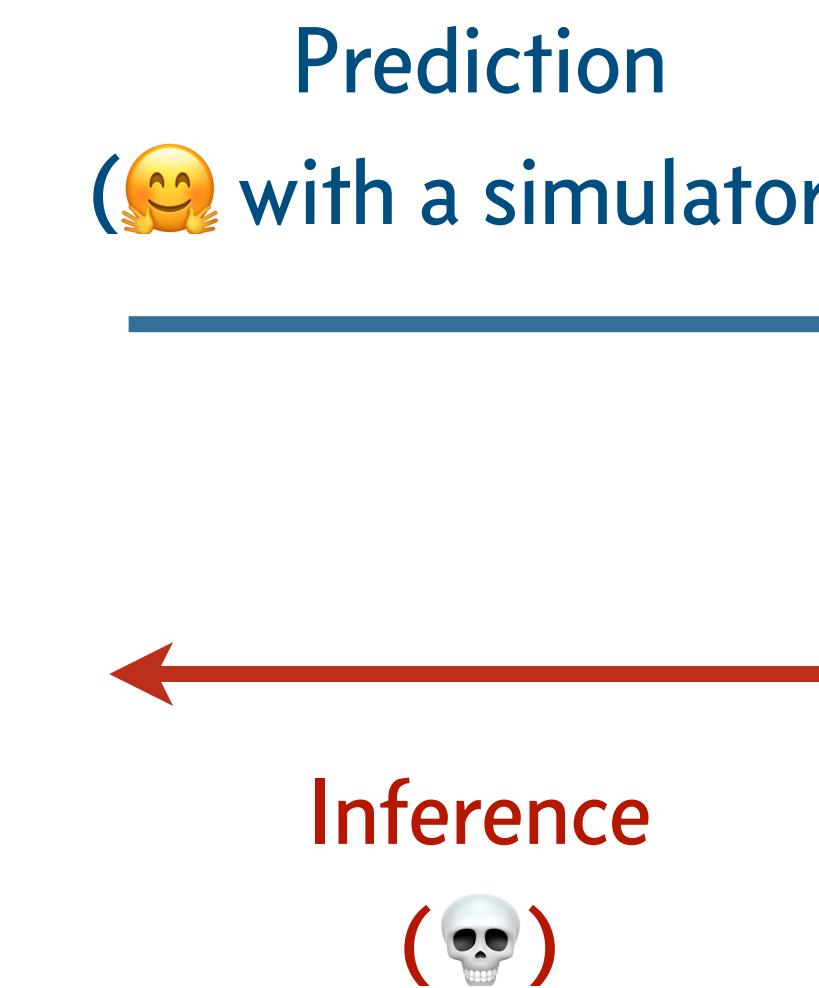
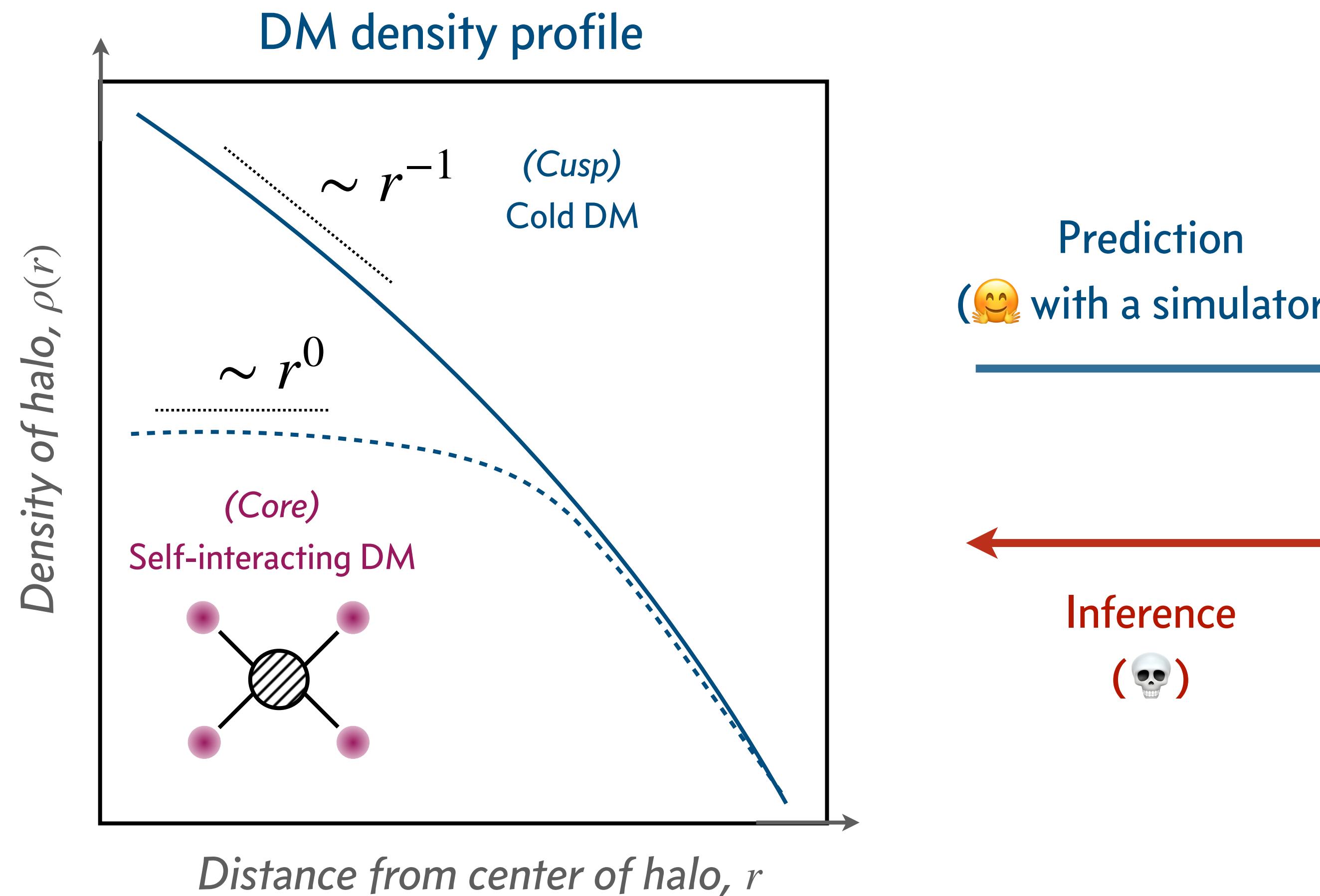
Extracting the dark matter distribution from dwarf galaxies

[Nguyen, SM et al 2022]

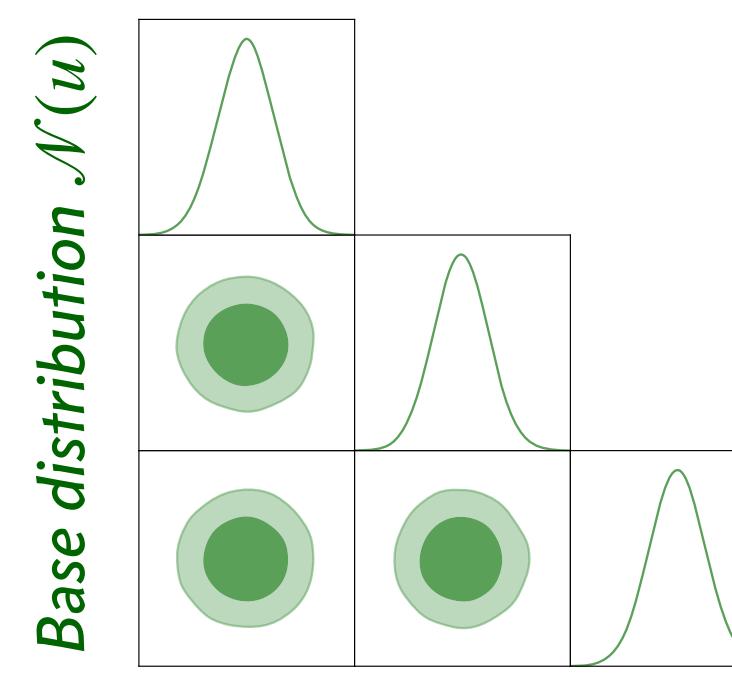


Extracting the dark matter distribution from dwarf galaxies

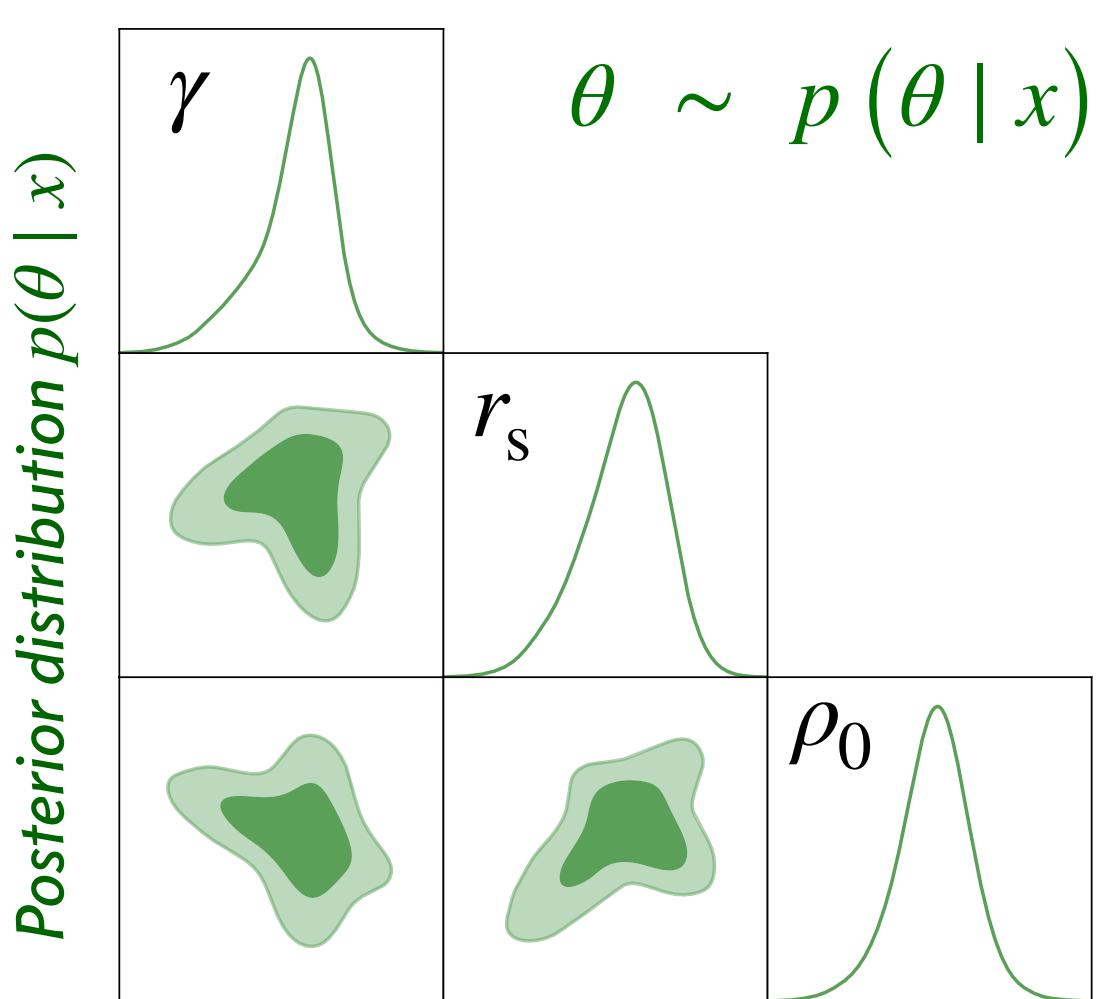
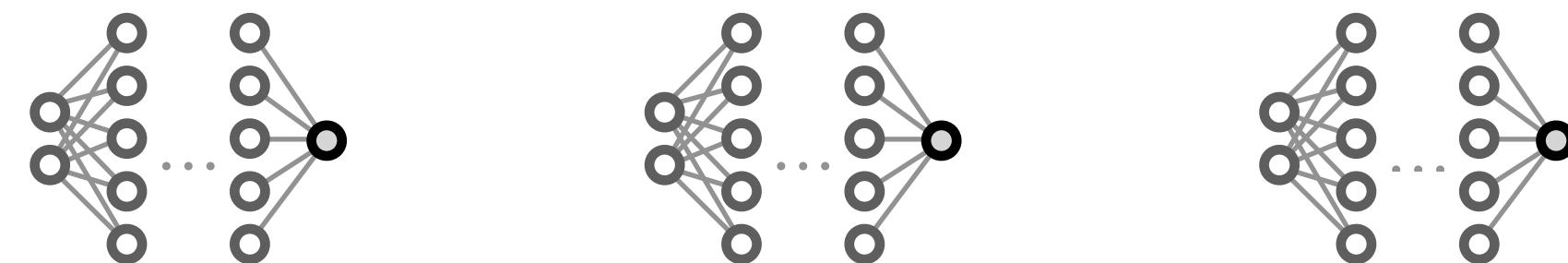
[Nguyen, SM et al 2022]



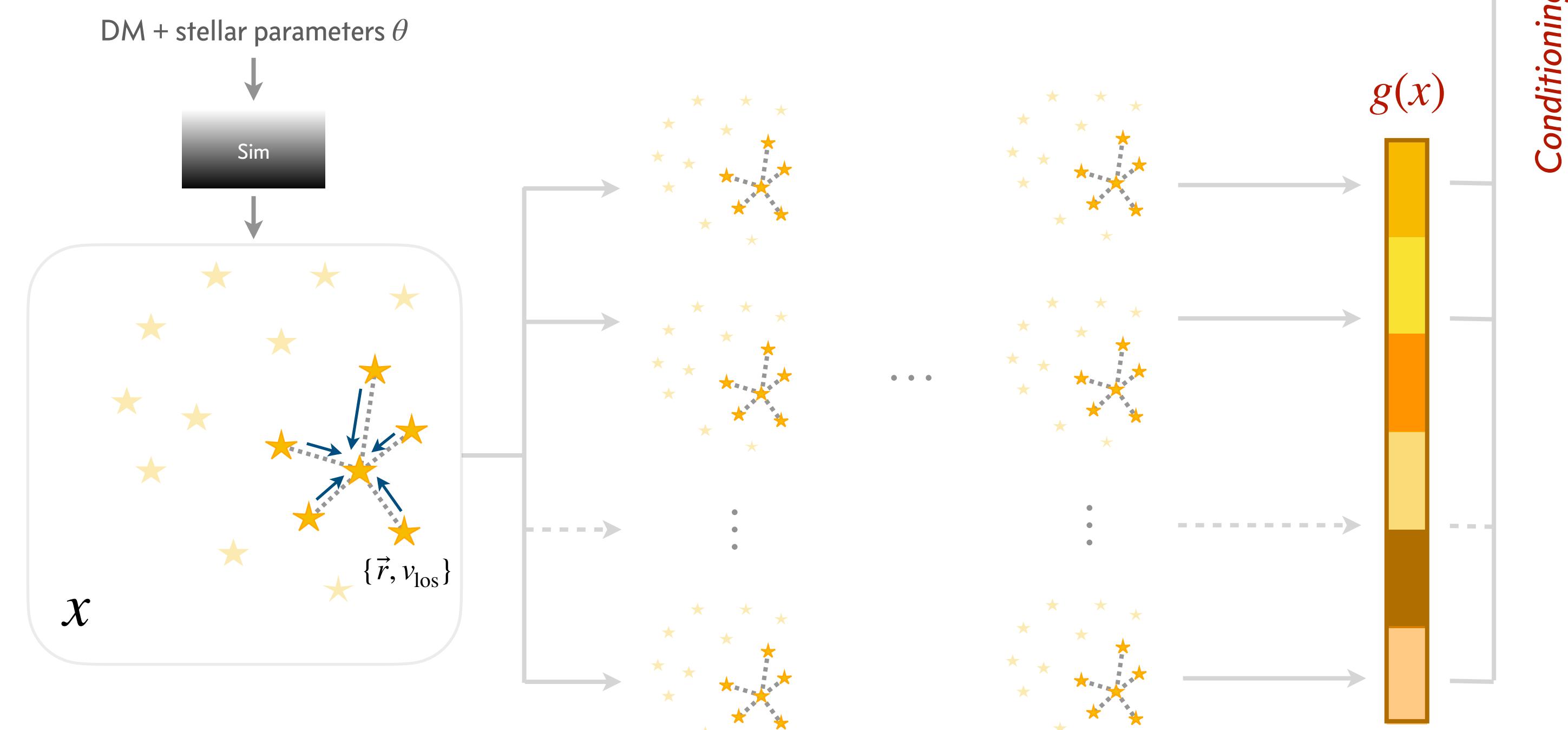
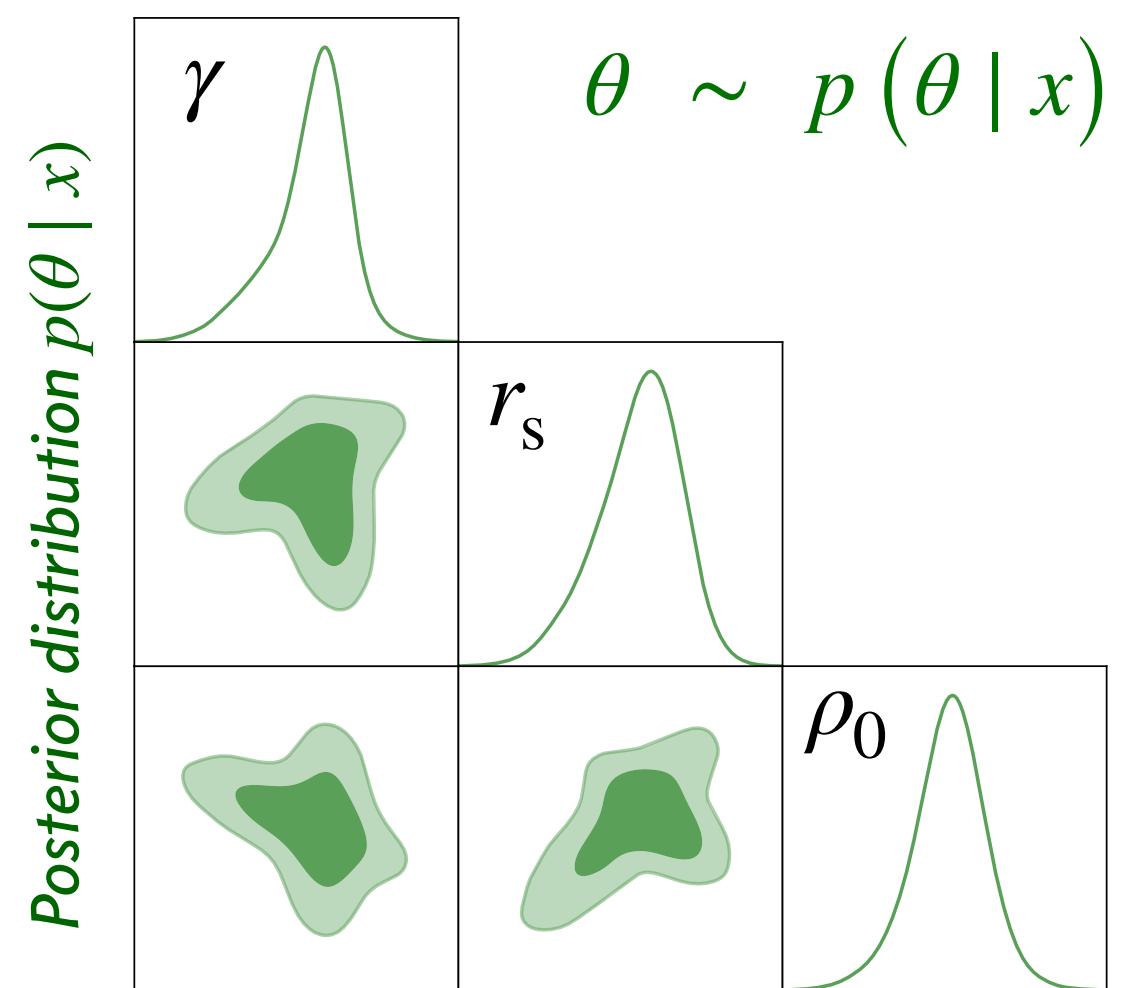
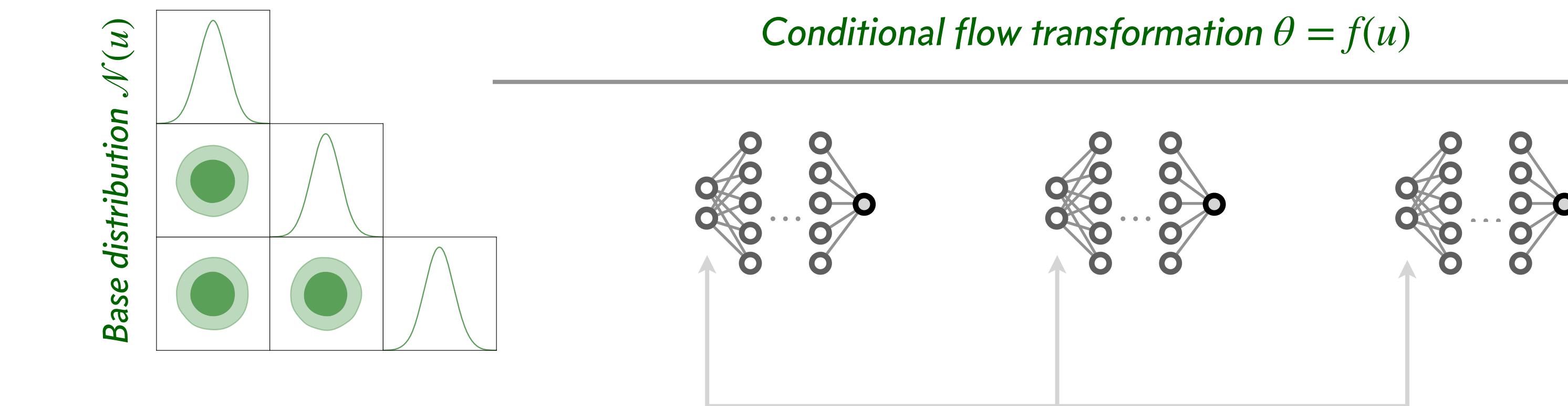
Extracting the dark matter distribution [Nguyen, SM et al 2022]



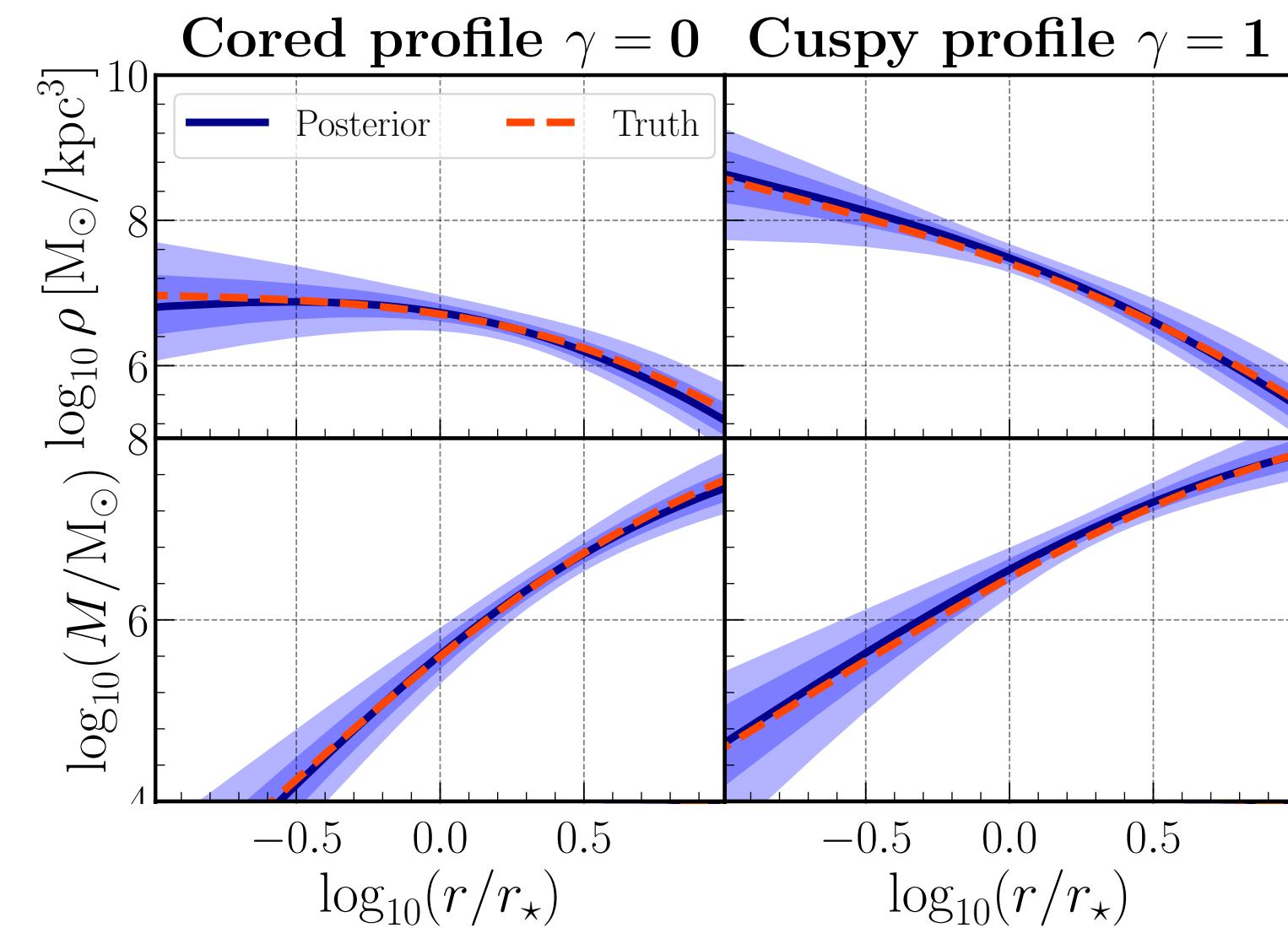
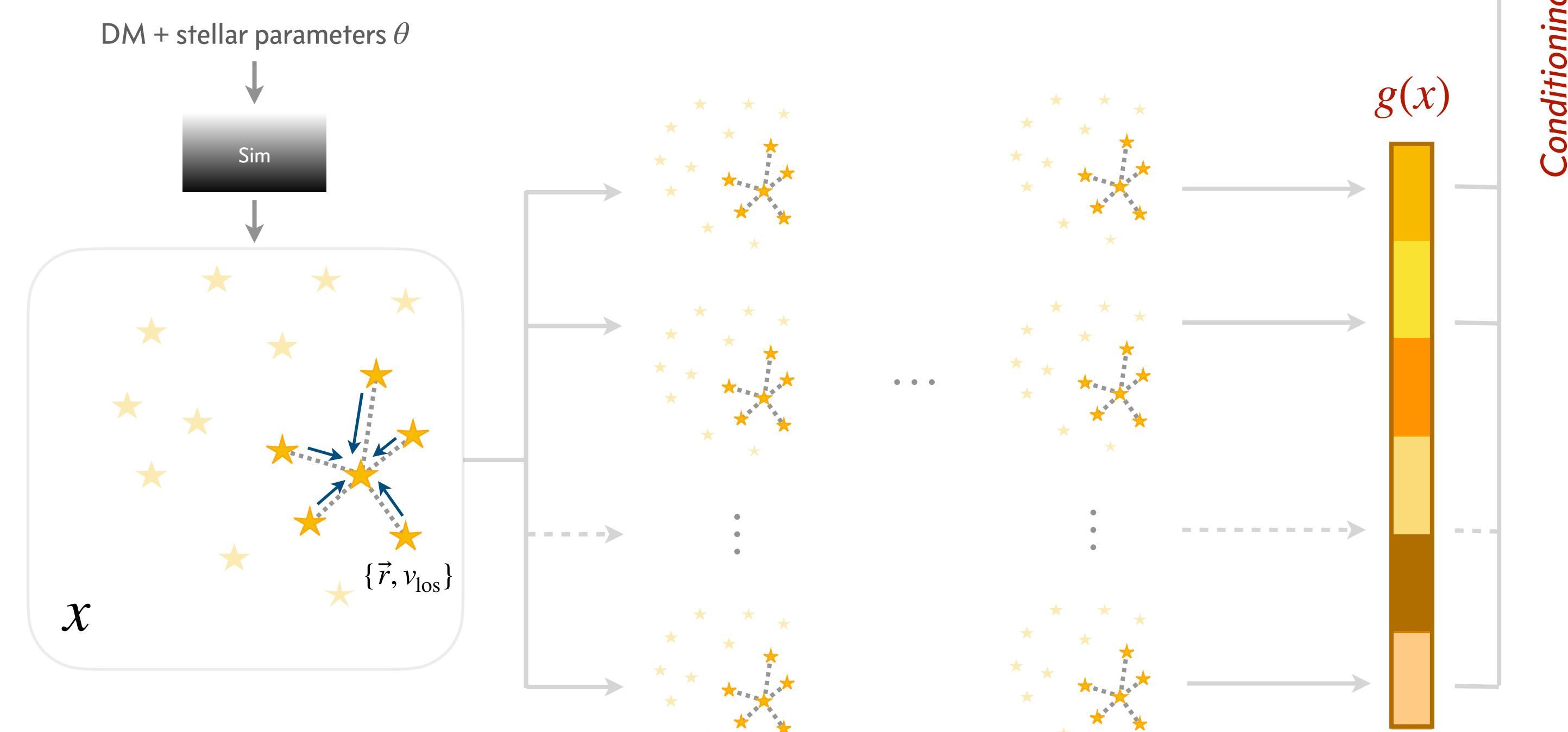
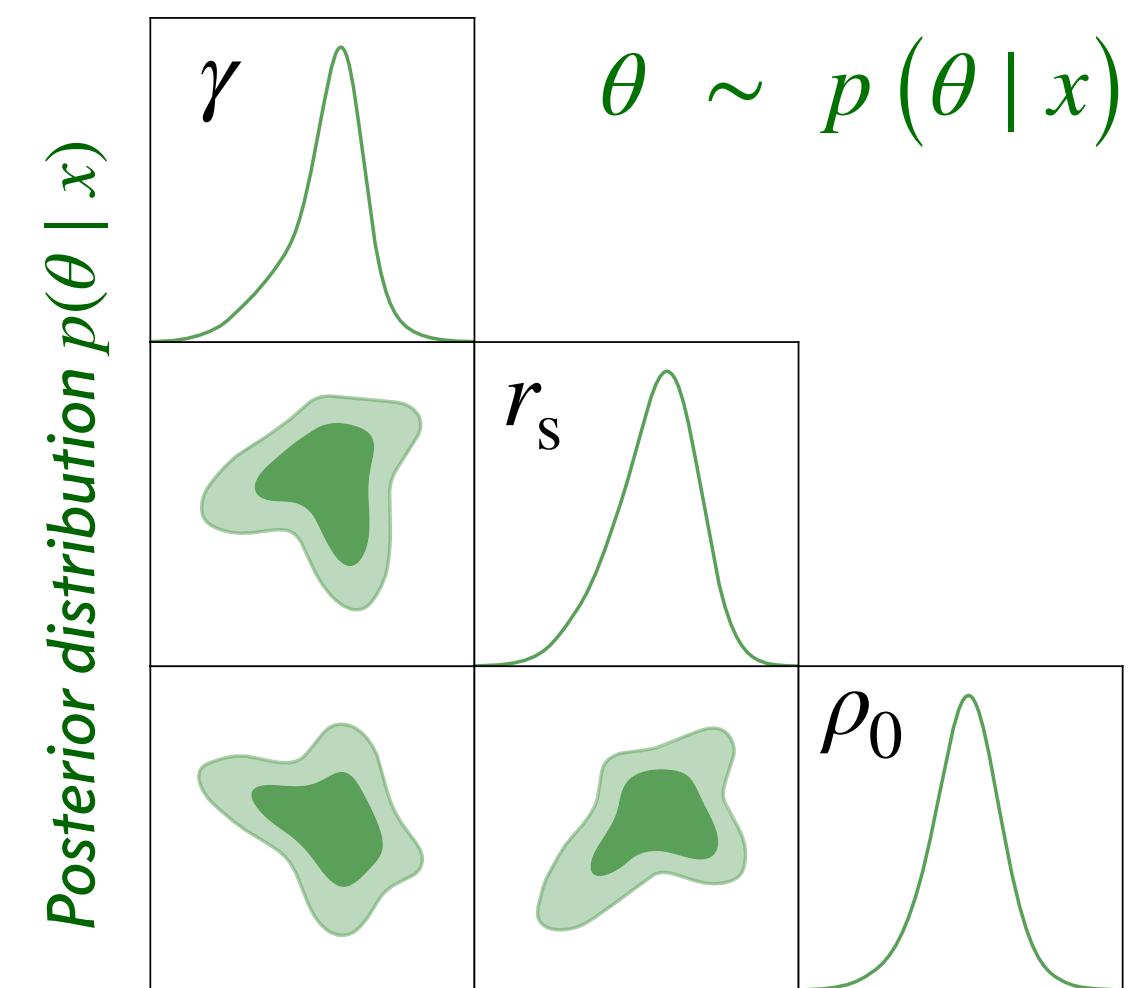
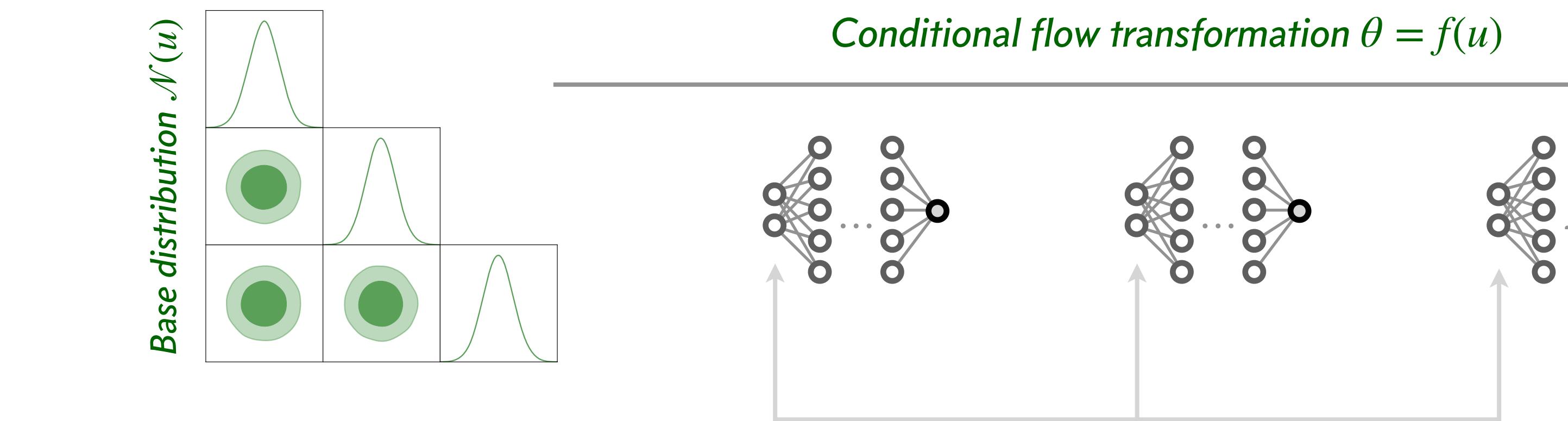
Conditional flow transformation $\theta = f(u)$



Extracting the dark matter distribution [Nguyen, SM et al 2022]

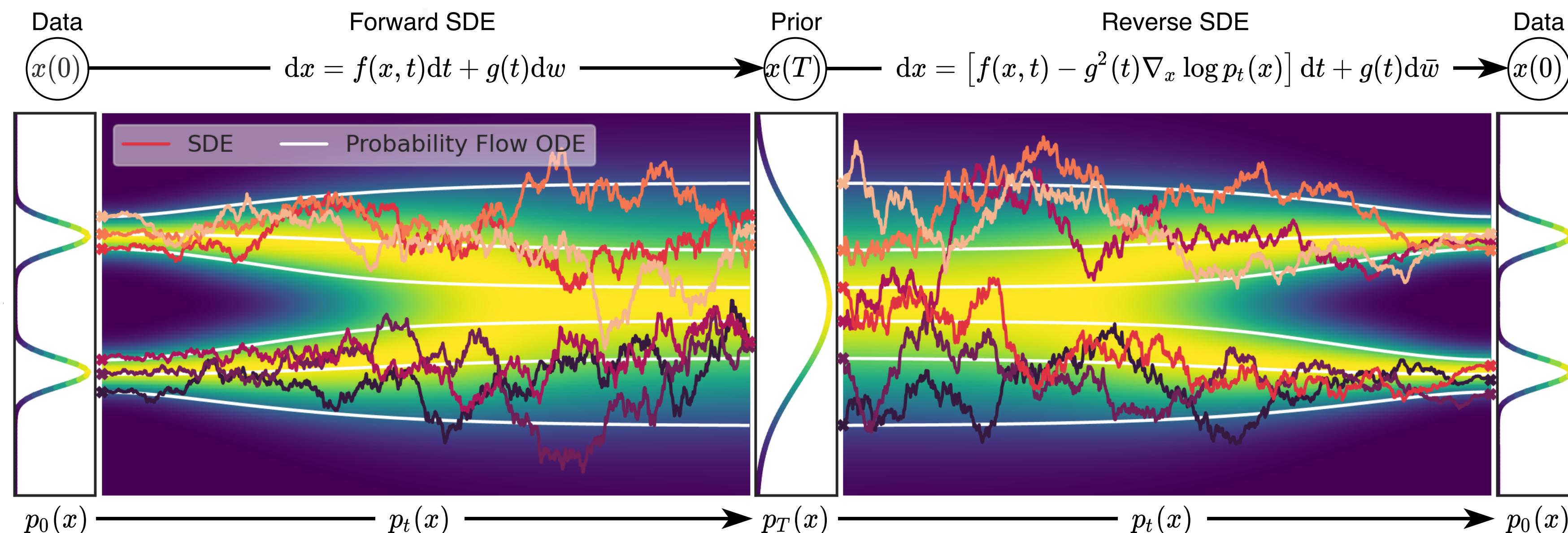


Extracting the dark matter distribution [Nguyen, SM et al 2022]



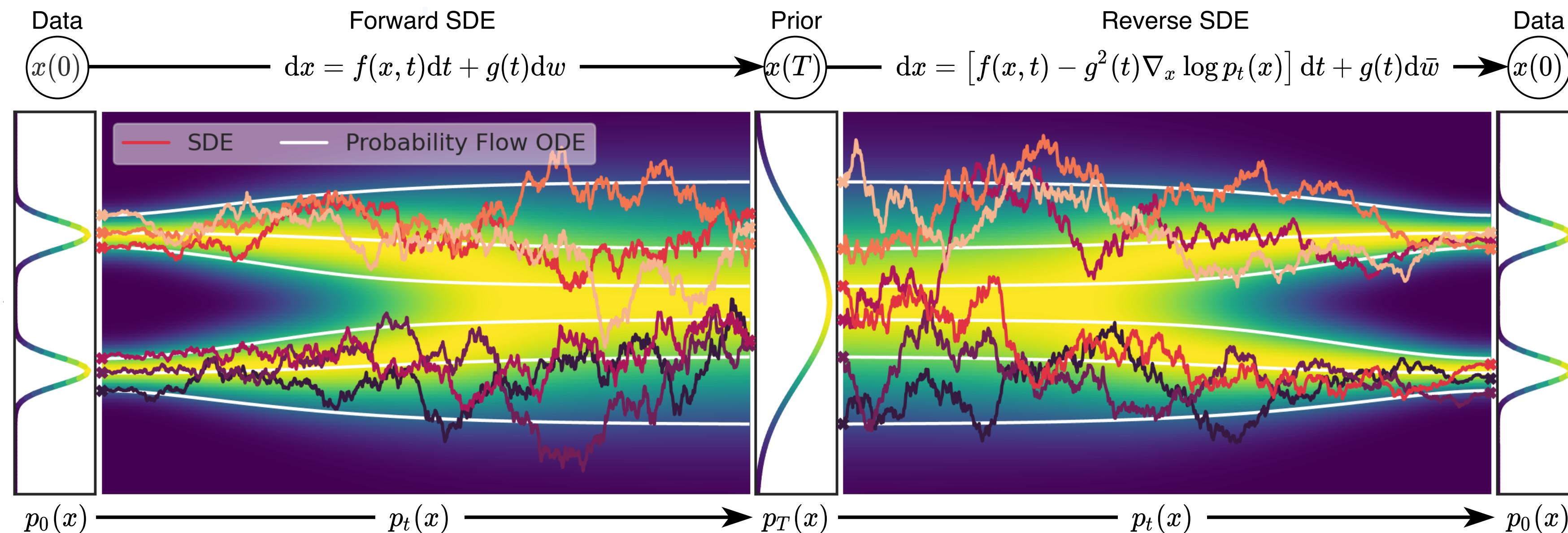
Back to diffusion: the *probability flow ODE*

For any diffusion process, there exists a corresponding deterministic process whose trajectories share the same marginal probability densities $p(x_t)$ as the SDE [Song et al 2021]



Back to diffusion: the *probability flow ODE*

For any diffusion process, there exists a corresponding deterministic process whose trajectories share the same marginal probability densities $p(x_t)$ as the SDE [Song et al 2021]

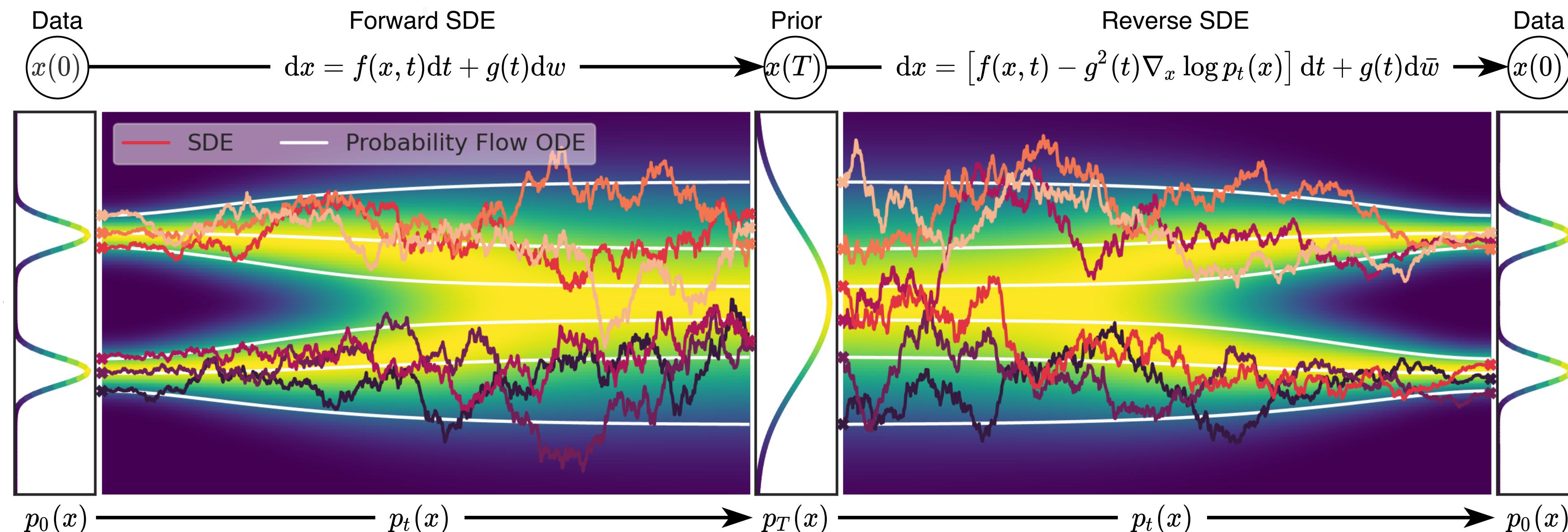


Probability flow ODE

$$dx = \left[f(x, t) - \frac{1}{2}g^2(t) \nabla_x \log p_t(x) \right] dt$$

Back to diffusion: the *probability flow ODE*

For any diffusion process, there exists a corresponding deterministic process whose trajectories share the same marginal probability densities $p(x_t)$ as the SDE [Song et al 2021]



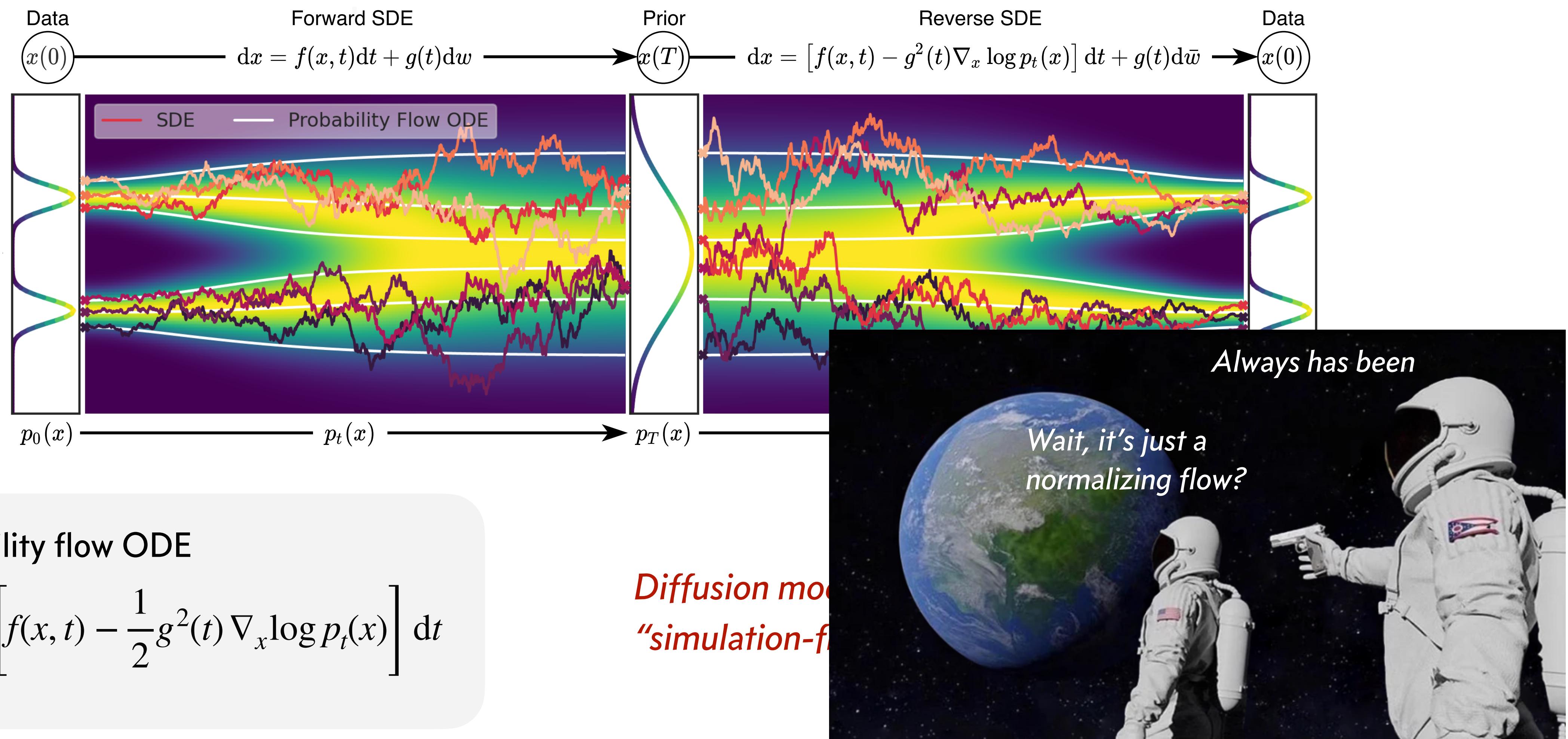
Probability flow ODE

$$dx = \left[f(x, t) - \frac{1}{2}g^2(t)\nabla_x \log p_t(x) \right] dt$$

Diffusion models are just normalizing flows trained in a “simulation-free” way by score/noise-matching!

Back to diffusion: the *probability flow ODE*

For any diffusion process, there exists a corresponding deterministic process whose trajectories share the same marginal probability densities $p(x_t)$ as the SDE [Song et al 2021]

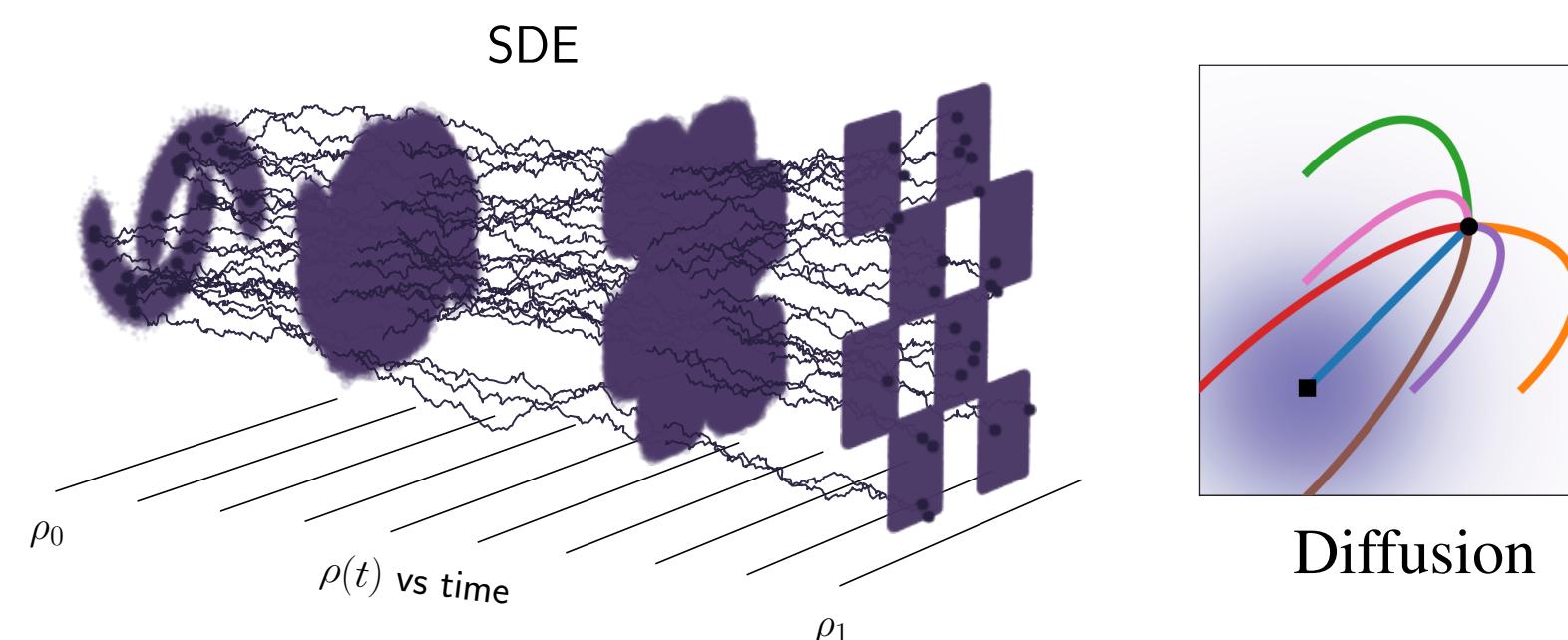


Iterative refinement, interpolants, consistency, ...

Empirical success of diffusion models has led to many new formulations and methods of training

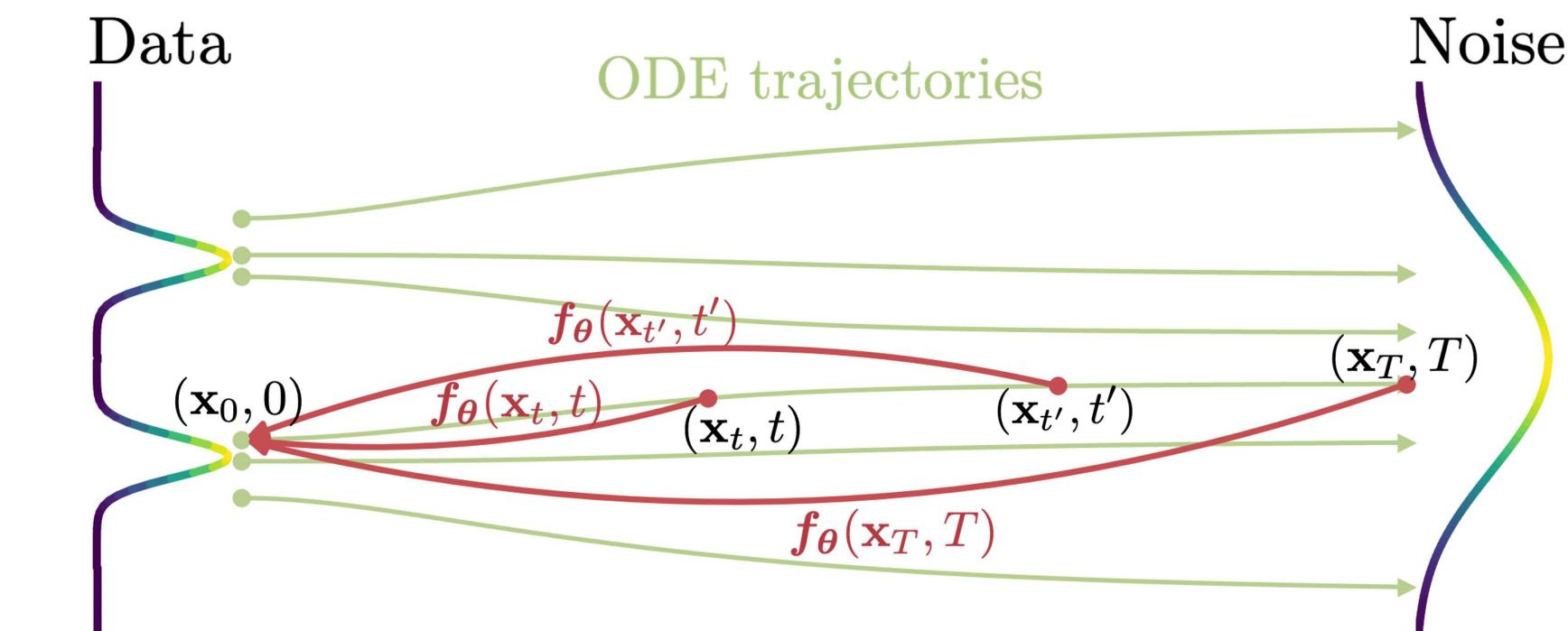
Stochastic interpolants and flow matching

[Albergo et al 2023, Lipman et al 2022, ...]



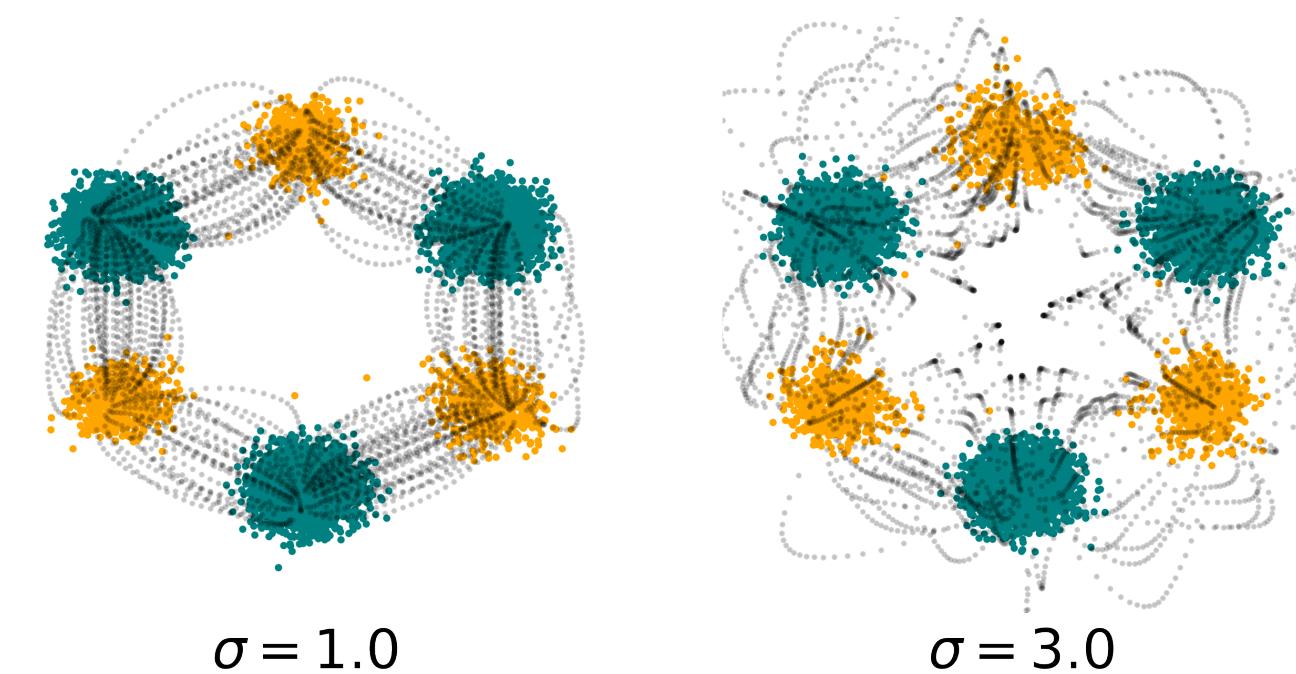
ODE trajectory consistency, Fokker-Planck regularization, ...

[Song et al 2023, Lai et al 2023, ...]



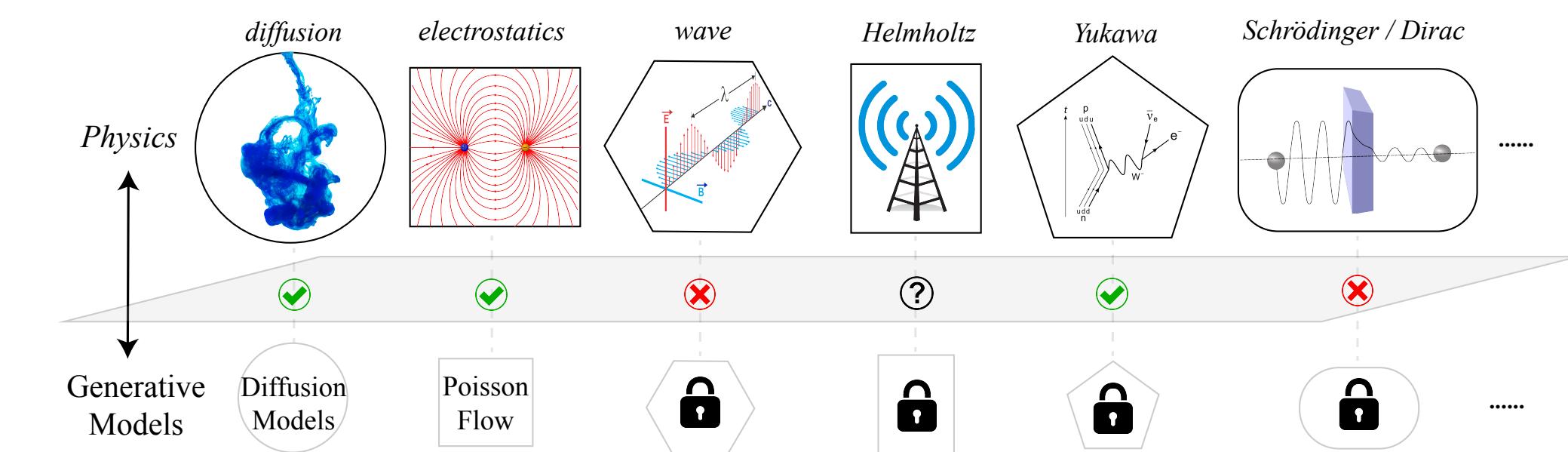
Diffusion Schrödinger's bridges

[Shi et al 2023, ...]



Generative models inspired by other physical processes

[Xu et al 2022, Liu et al 2023, ...]





End.