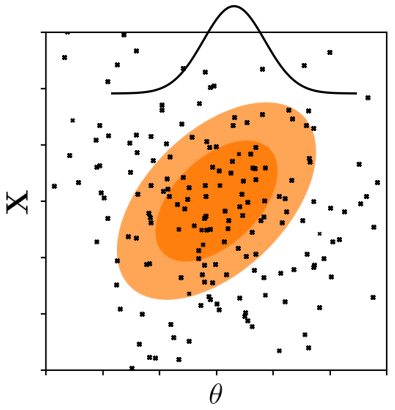
summary of first lecture

SBI is a density estimation problem -
allows to approximate a posterior distribution without an explicit likelihood using a forward model



$p(\theta \mid \mathbf{X})$

deep learning enables density estimation at very high dimension, enabling fast inference

$$(x_0, x_1, ..., x_N) \in X \longrightarrow p_w(X)$$

(evaluation + sampling)

density estimation with neural networks

= neural network

$(x_0, x_1, ..., x_N) \in X \longrightarrow p_w(X) \longrightarrow$ Find w so that: $\max \log p_w(X)$

density estimation with neural networks

= neural network

$(x_0, x_1, ..., x_N) \in X \longrightarrow p_w(X) \longrightarrow$ Find w so that: $\max \log p_w(X)$

MDN

$$p_w(y \mid x) = L(w) = \prod_{i=1}^{N} \sum_{k=1}^{2} \pi_w^{(k)}(x_i) \frac{1}{\sqrt{2\pi}\,\sigma_w^{(k)}(x_i)} \exp\left(-\frac{\left(y_i - f_w^{(k)}(x_i)\right)^2}{2\left[\sigma_w^{(k)}(x_i)\right]^2} \right)$$

a general probability function:

$$p_\theta(x) = \frac{\exp\big(f_\theta(x)\big)}{Z(\theta)},$$

$$Z(\theta) = \int_{\mathcal{X}} \exp\big(f_\theta(x)\big)\, dx,$$

$$\log p_\theta(x) = f_\theta(x) - \log Z(\theta).$$

Neural network

θ =weights

a general probability function:

Neural network
θ =weights

$$p_\theta(x) = \frac{\exp\big(f_\theta(x)\big)}{Z(\theta)},$$

$$Z(\theta) = \int_{\mathcal{X}} \exp\big(f_\theta(x)\big)\, dx,$$

$$\log p_\theta(x) = f_\theta(x) - \log Z(\theta).$$

however, computing the gradient is intractable (norm. constant)

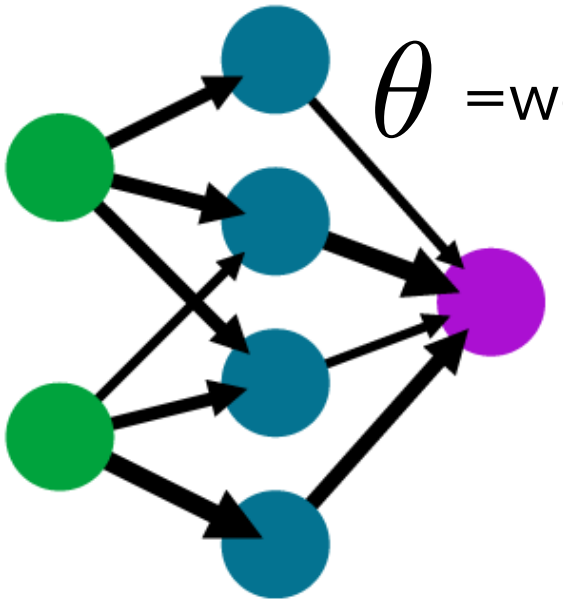
$$\begin{aligned}\nabla_\theta \log Z(\theta) &= \frac{1}{Z(\theta)} \int \exp\big(f_\theta(x)\big)\, \nabla_\theta f_\theta(x)\, dx \\ &= \int \frac{\exp\big(f_\theta(x)\big)}{Z(\theta)}\, \nabla_\theta f_\theta(x)\, dx \\ &= \mathbb{E}_{x\sim p_\theta}[\nabla_\theta f_\theta(x)]\end{aligned}$$

It's an expectation under the **current model (neural network)** p_θ which itself depends on $Z(\theta)$

generative model

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

$$p(z)$$

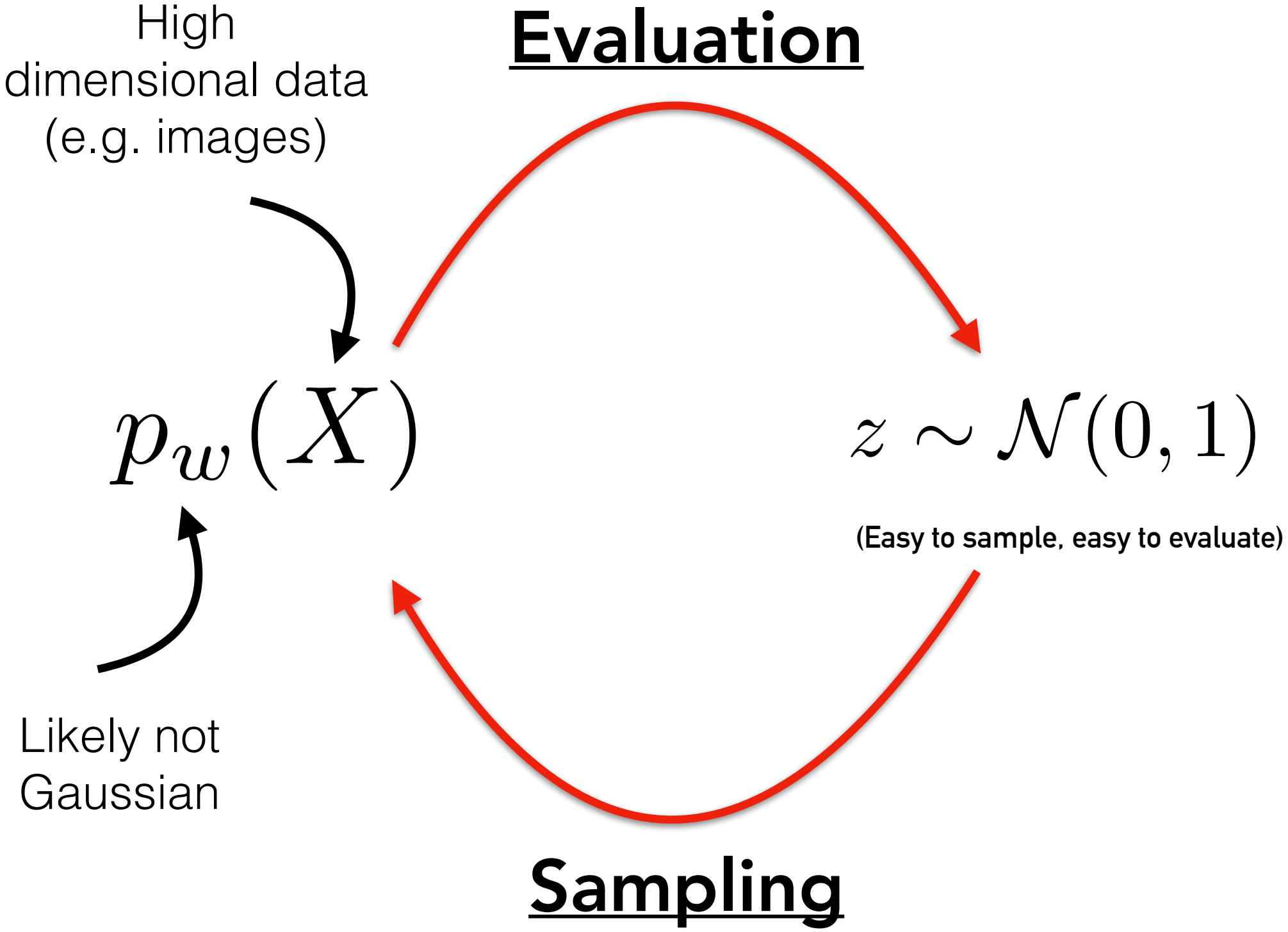


θ =weights

$$p_\theta(x)$$

generative models are proposed solutions to make this optimization problem practical by avoiding to compute the normalization constant $Z(\theta)$.

	Types of generative models		loss
Likelihood based models	directly learn the distribution's probability density function with approximations	VAEs, Normalizing Flows	$\max_{\theta} \sum_{i=1}^{N} \log p_\theta(x_i)$
Implicit generative models	the probability distribution is implicitly represented by a model of its sampling process	GANs	$\mathcal{L}_D = -\frac{1}{m}\sum_{i=1}^{m}\left[\log D(x_i) + \log\left(1 - D(\tilde{x}_i)\right)\right], \quad \mathcal{L}_G^{\mathrm{NS}} = -\frac{1}{m}\sum_{i=1}^{m}\log D(\tilde{x}_i)$
Score based models	model the gradient of the log probability density function	Score based diffusion	$\nabla_x \log p_\theta(x)$

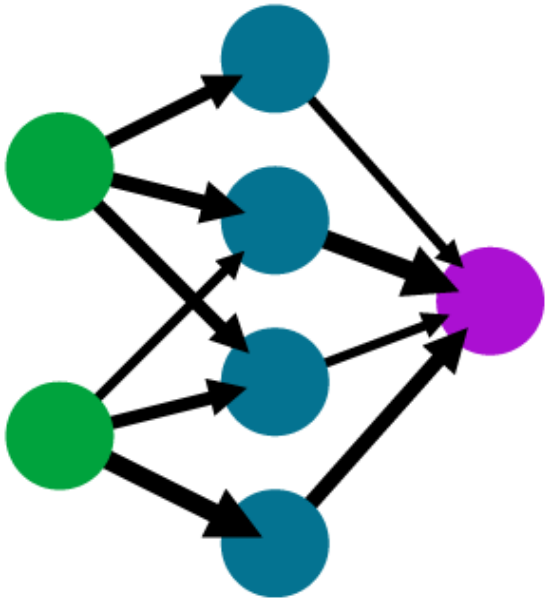

High dimensional data (e.g. images)
Evaluation
$p_w(X)$
$z \sim \mathcal{N}(0, 1)$
(Easy to sample, easy to evaluate)
Likely not Gaussian
Sampling

High dimensional data (e.g. images)

Evaluation

NNs == universal approximators

$p_w(X)$



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

(Easy to sample, easy to evaluate)

(Optimization problem)

Likely not Gaussian

Sampling

Normalizing Flows

Based on change of variables:

$$p_X(x) = p_Z\big(f(x)\big) \left|\det J_f(x)\right|, \qquad J_f(x) = \frac{\partial f(x)}{\partial x^\top}$$

unknown pdf

tractable pdf
(typically Gaussian)

Invertible differentiable
function $z \sim \mathcal{N}(0, 1)$

Determinant of the Jacobian
of f(x)

the neural network parametrizes the function f: $f_\theta(x)$

and is optimized so that:

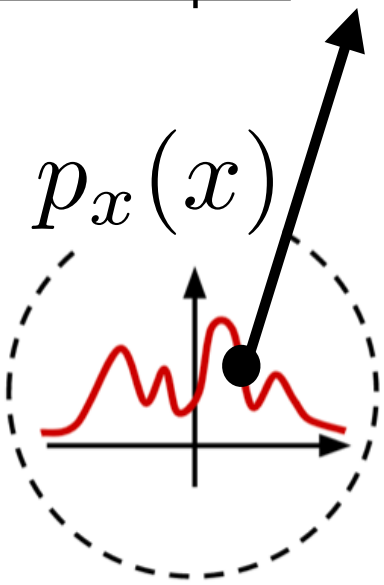
$$p_Z\big(f_\theta(x)\big) \; \big|\det J_f(x)\big| \qquad \text{is maximum}$$

$$x \in X$$

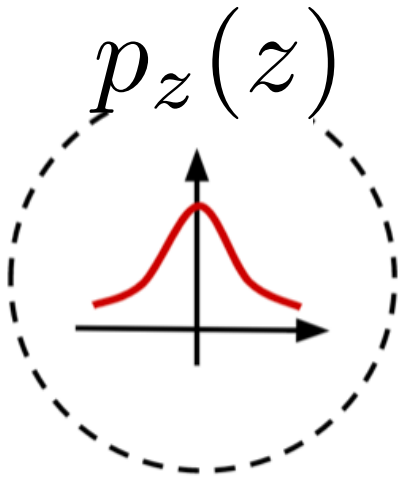
$$\sim p_X(x)$$

training procedure

a sample x_i



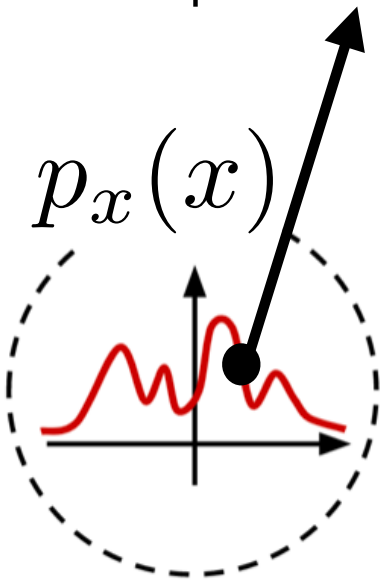
(unknown)



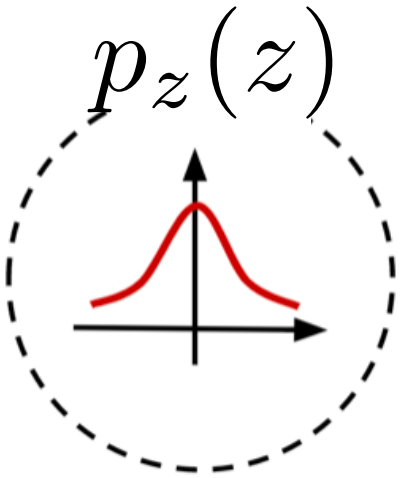
(known and easy)

training procedure

a sample x_i $\longrightarrow$ $f_\theta(x_i)$

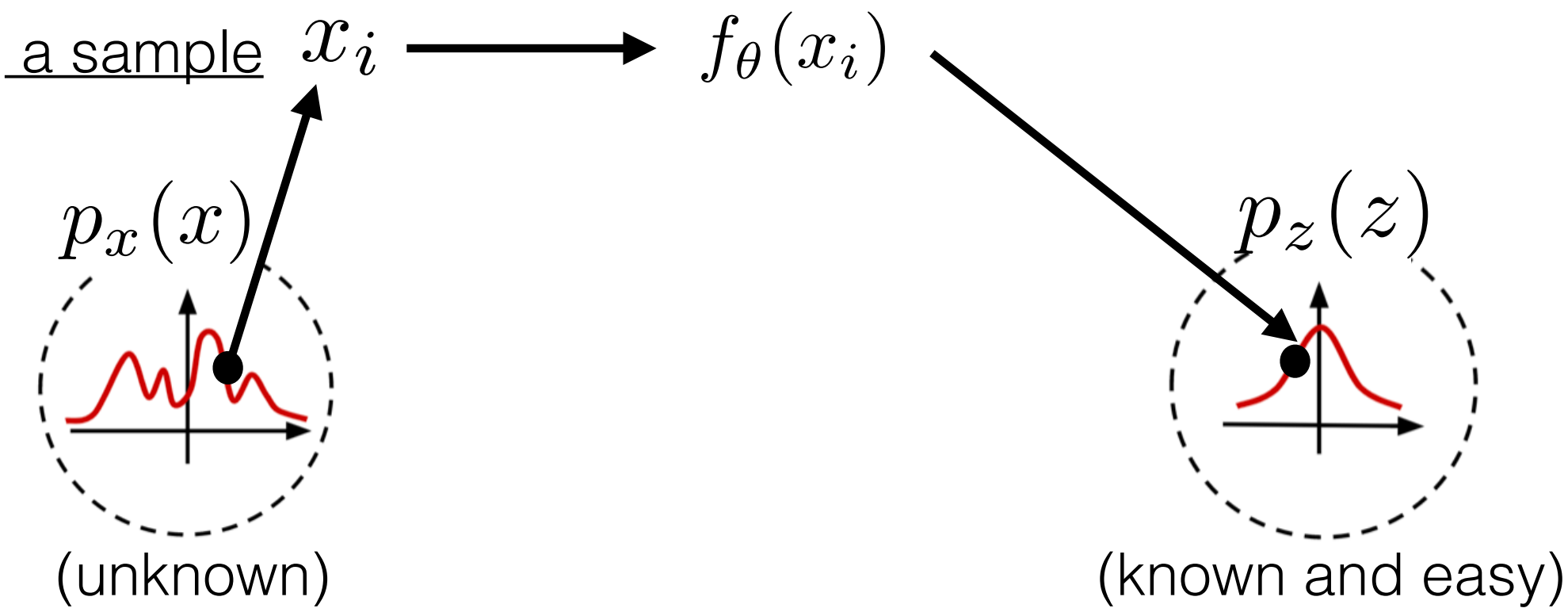


(unknown)

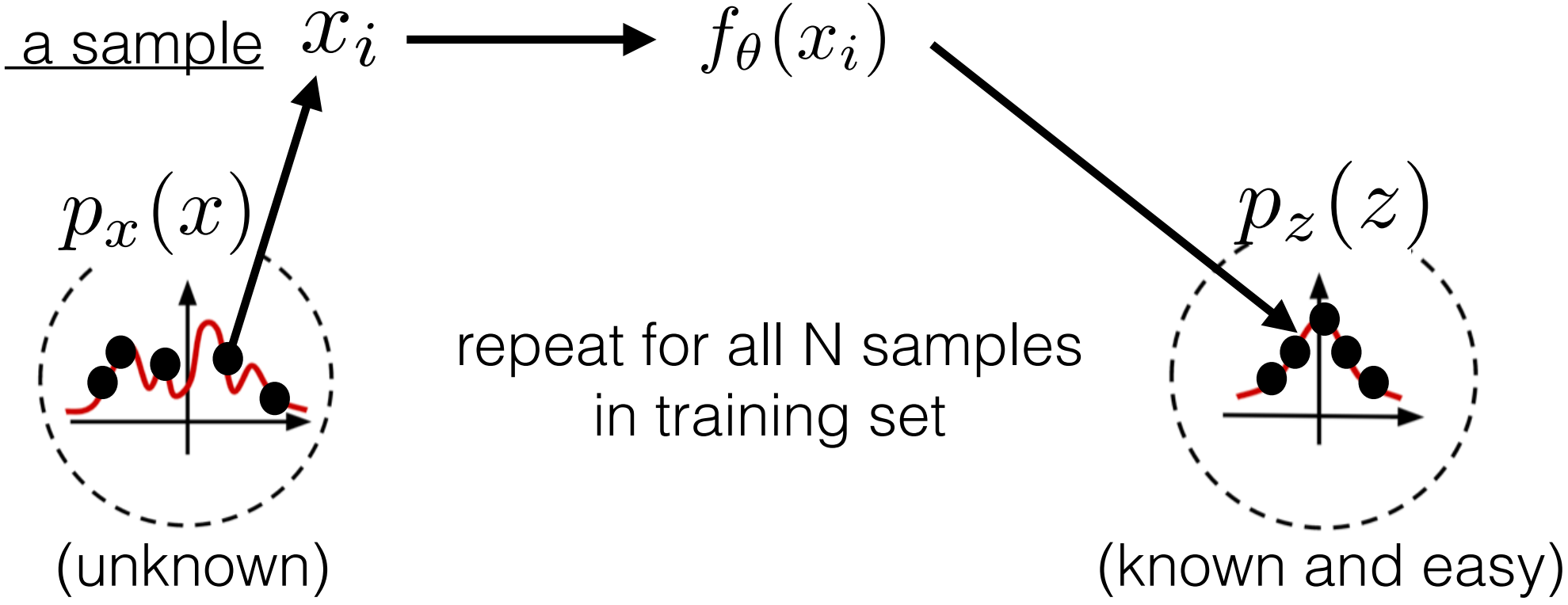


(known and easy)

training procedure



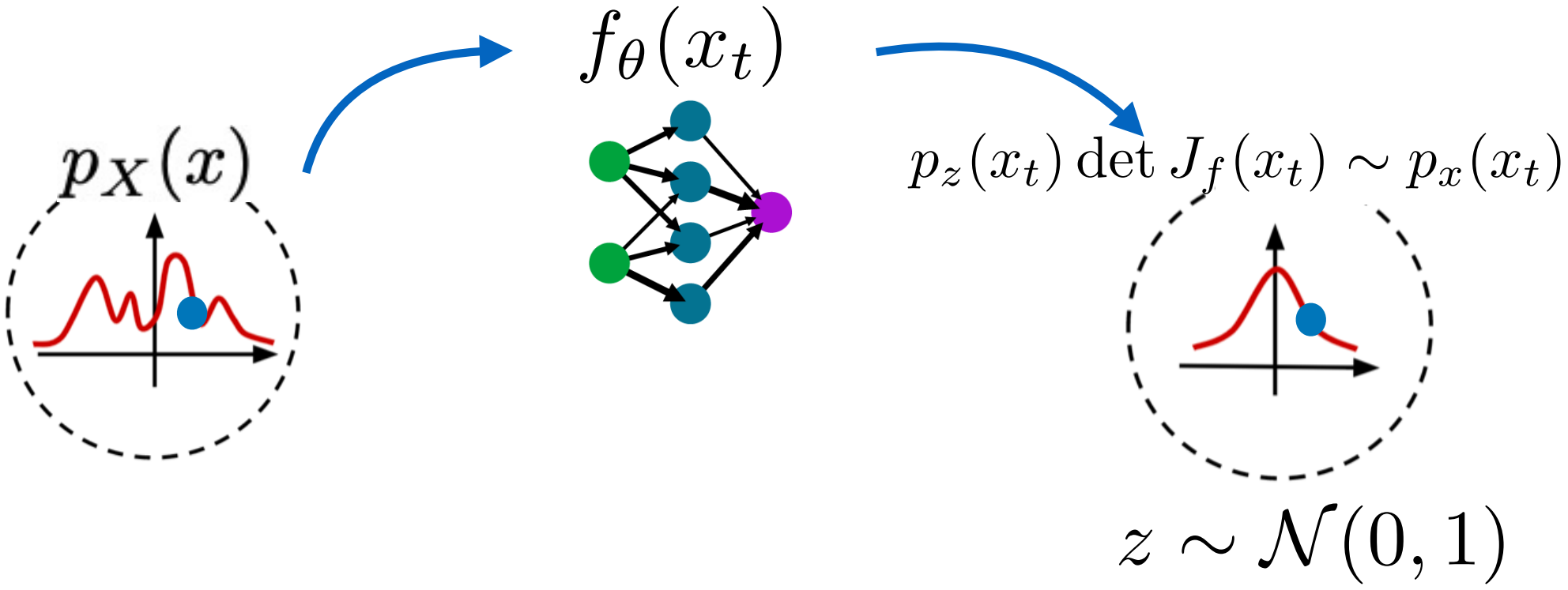
training procedure



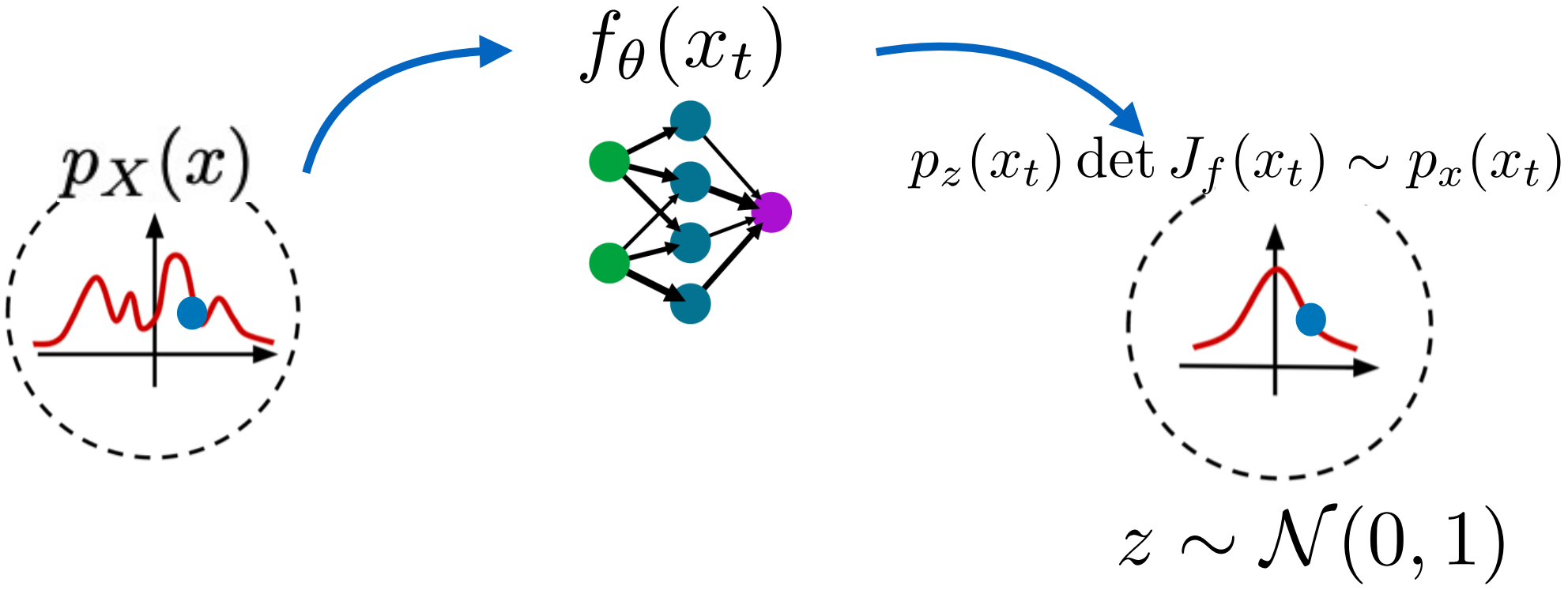
achieved by maximizing this loss function

$$\sum_{i=1}^{N} \log p_z(x_i) + \log \det J_f(x_i)$$

Likelihood evaluation

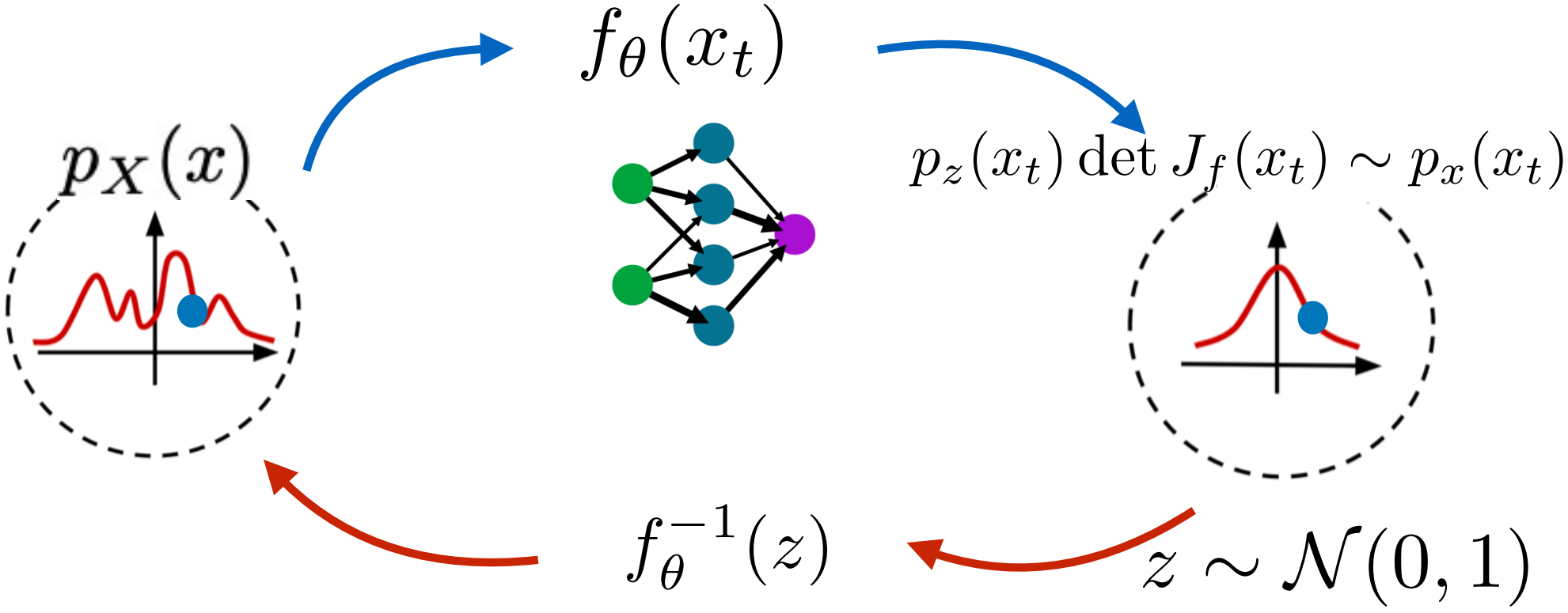


Likelihood evaluation

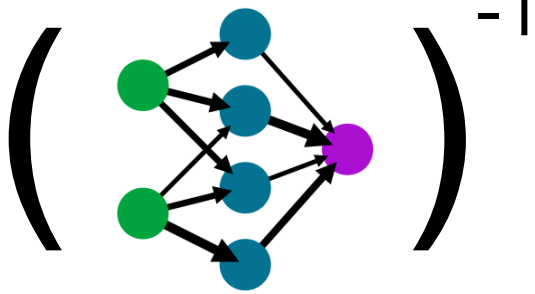


Sampling?

Likelihood evaluation



Sampling



The price to pay is that we need to put some restrictions on the NN architecture

- 1. Easily invertible - to enable sampling
- 2. Jacobian tractable - to optimize the network

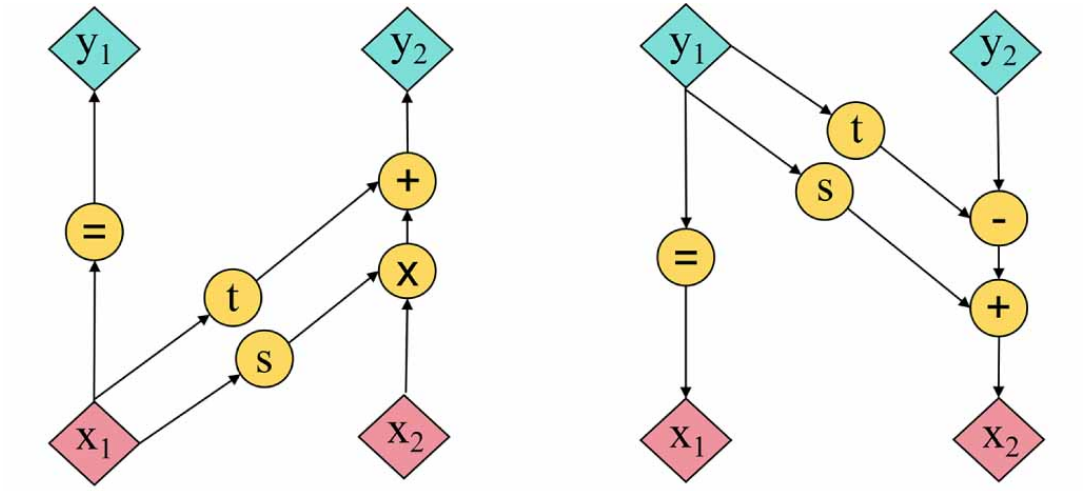
What functions satisfy these conditions?

An example are: RealNVPs (Real-valued Non-Volume Preserving)

$$\mathbf{y}_{1:d} = \mathbf{x}_{1:d}$$

$$\mathbf{y}_{d+1:D} = \mathbf{x}_{d+1:D} \odot \exp(s(\mathbf{x}_{1:d})) + t(\mathbf{x}_{1:d})$$

NN



(a) Forward transformation (b) Inverse transformation

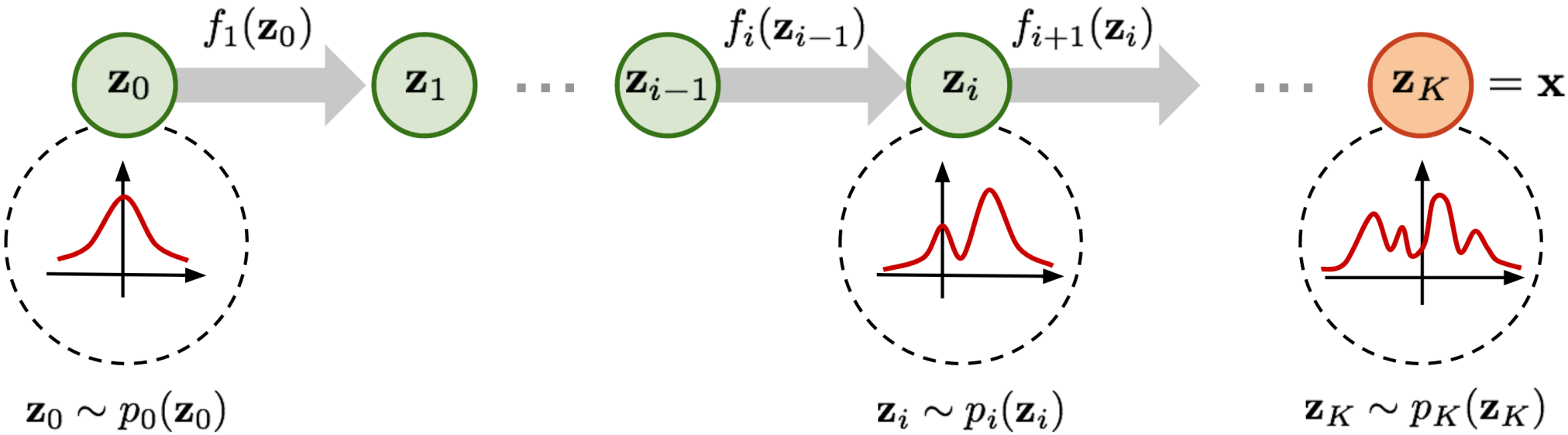
- Easily invertible:

$$\begin{cases} \mathbf{y}_{1:d} & = \mathbf{x}_{1:d} \\ \mathbf{y}_{d+1:D} & = \mathbf{x}_{d+1:D} \odot \exp(s(\mathbf{x}_{1:d})) + t(\mathbf{x}_{1:d}) \end{cases} \Leftrightarrow \begin{cases} \mathbf{x}_{1:d} & = \mathbf{y}_{1:d} \\ \mathbf{x}_{d+1:D} & = (\mathbf{y}_{d+1:D} - t(\mathbf{y}_{1:d})) \odot \exp(-s(\mathbf{y}_{1:d})) \end{cases}$$

- Jacobian:

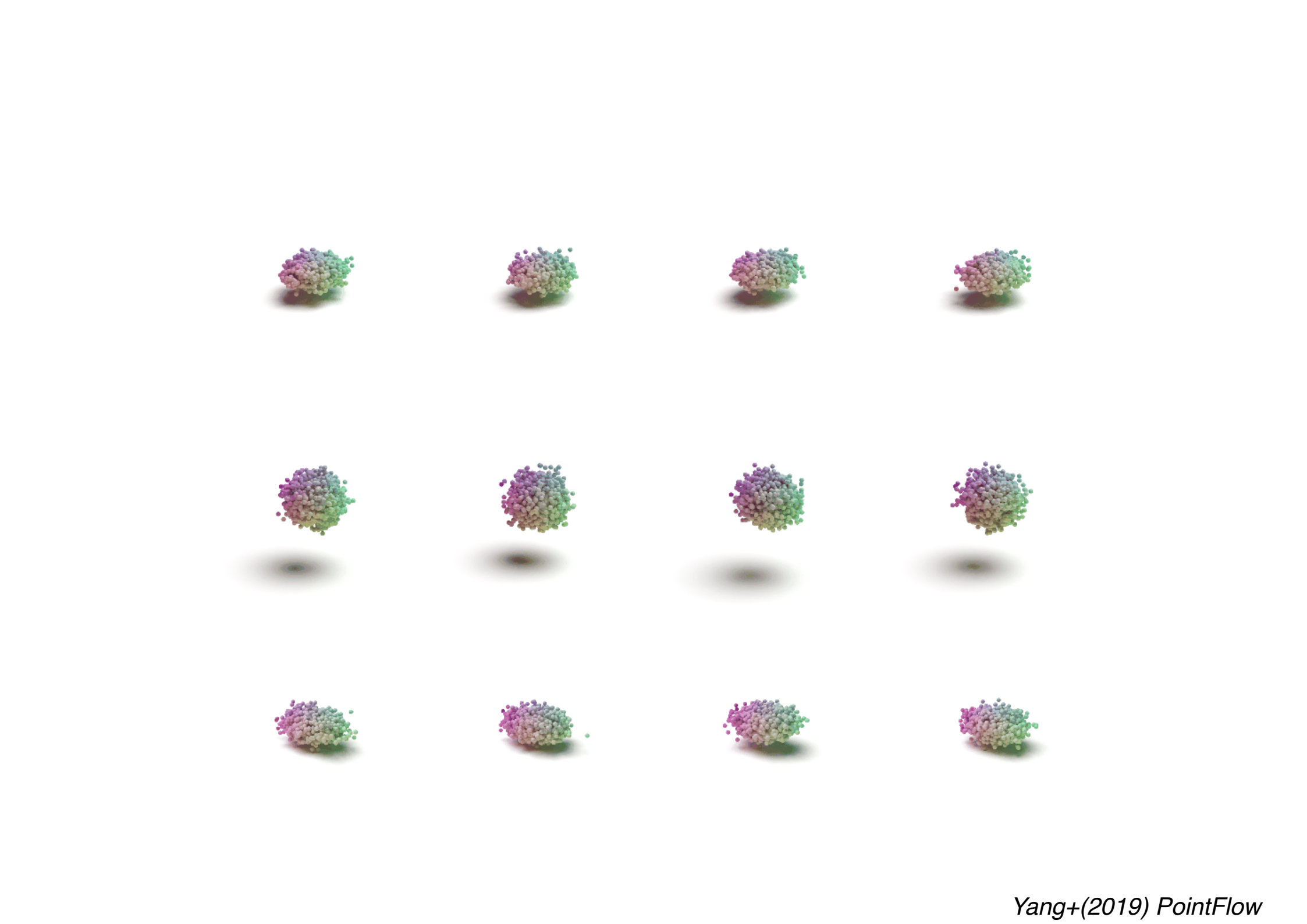
$$\mathbf{J} = \begin{bmatrix} \mathbb{I}_d & \mathbf{0}_{d\times(D-d)} \\ \frac{\partial \mathbf{y}_{d+1:D}}{\partial \mathbf{x}_{1:d}} & \operatorname{diag}(\exp(s(\mathbf{x}_{1:d}))) \end{bmatrix} \qquad \det(\mathbf{J}) = \prod_{j=1}^{D-d} \exp(s(\mathbf{x}_{1:d}))_j = \exp(\sum_{j=1}^{D-d} s(\mathbf{x}_{1:d})_j)$$

Normalizing flow: in practice we use a concatenation of neural networks (called bijectors)



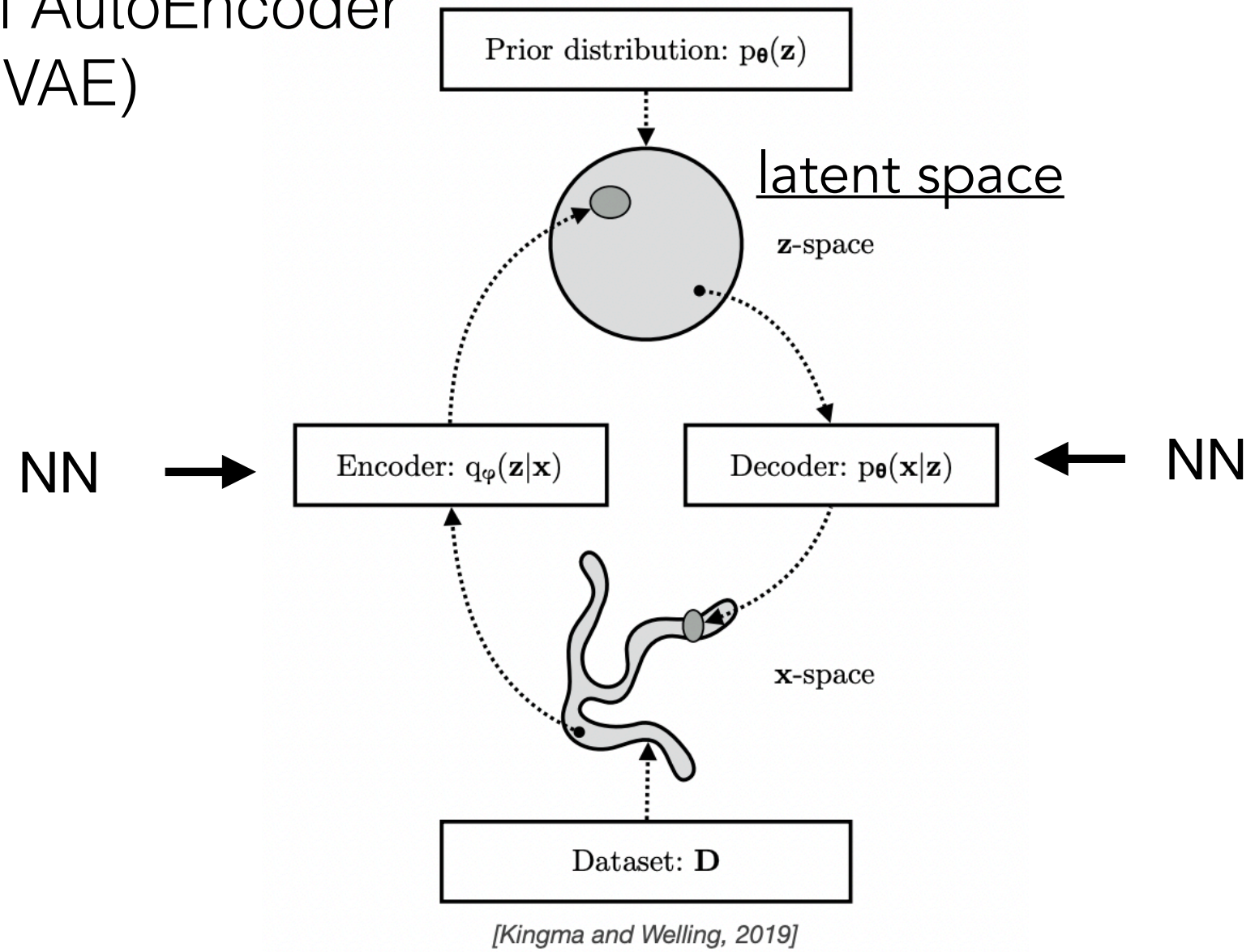
$z_i = f_i(z_{i-1})$ are **invertible** and **differentiable** transformations

$f = f_1 \circ f_2 \ldots \circ f_{k-1} \circ f_k$ is also invertible and differentiable



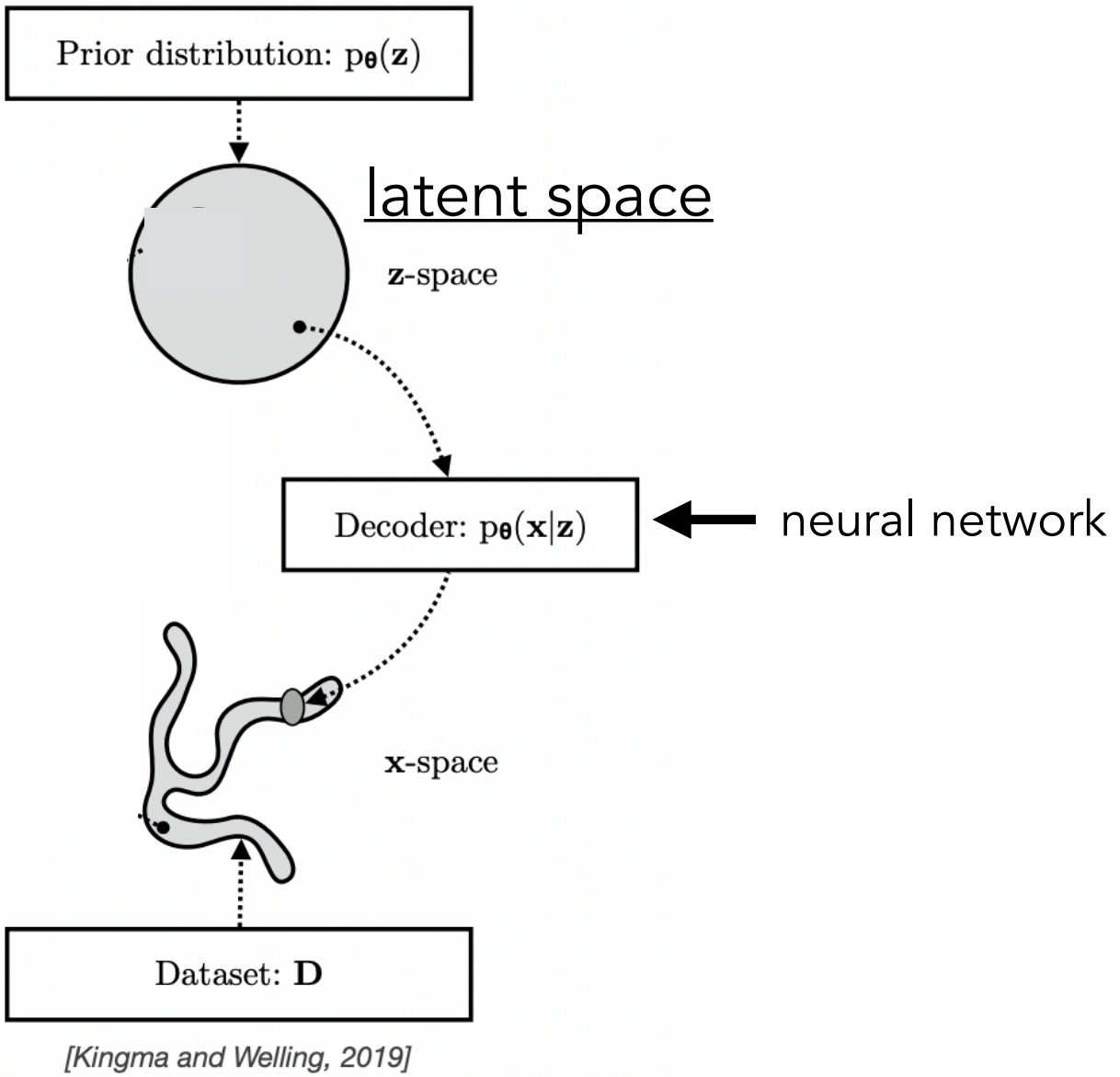
Yang+(2019) PointFlow

Variational AutoEncoder (VAE)



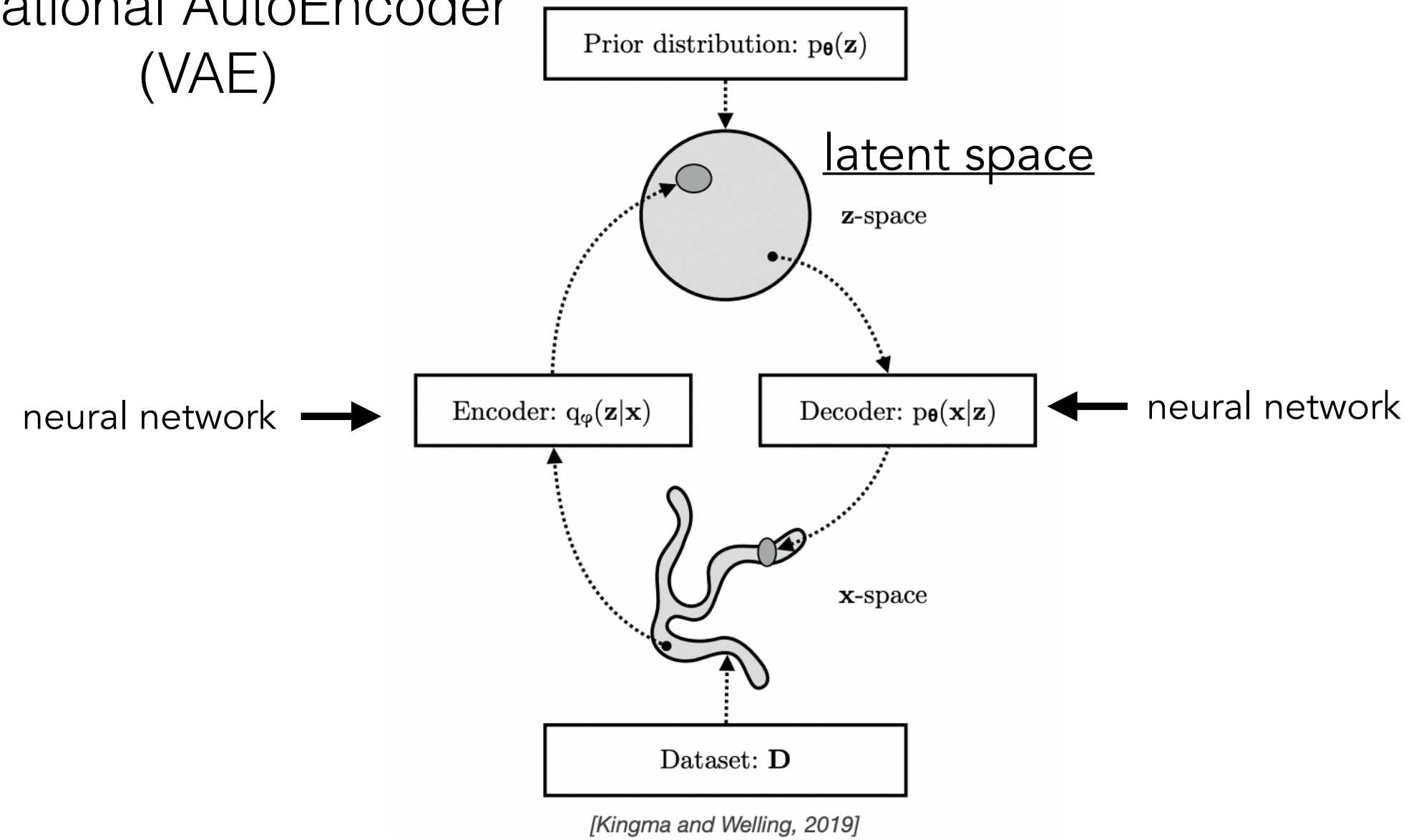
[Kingma and Welling, 2019]

Variational AutoEncoder (VAE)



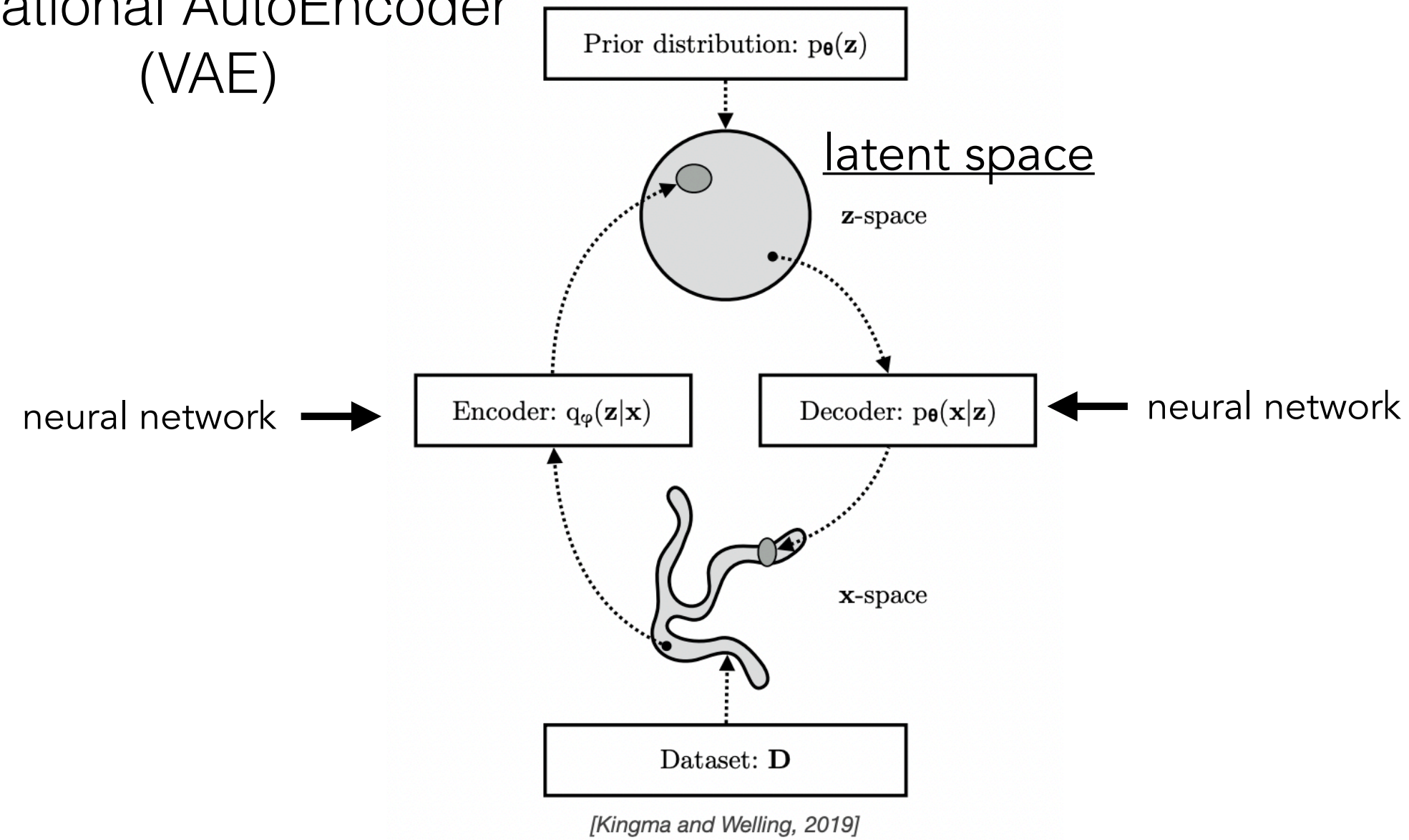
[Kingma and Welling, 2019]

Variational AutoEncoder (VAE)



[Kingma and Welling, 2019]

Variational AutoEncoder (VAE)



[Kingma and Welling, 2019]

Reconstruction term Regularization term

$$\mathcal{L}(\theta, \phi; x) = \mathbb{E}_{q_\phi(z|x)}[\log p_\theta(x \mid z)] - \mathrm{KL}(q_\phi(z \mid x) \,\|\, p_\theta(z))$$

we want to maximize the likelihood of our data:

$$\log p_\theta(x) = \log \int p_\theta(x, z)\, dz = \log \int p_\theta(x \mid z)\, p_\theta(z)\, dz$$

(Bayes Rule)

$$\log p_\theta(x) = \log \int q_\phi(z \mid x)\, \frac{p_\theta(x, z)}{q_\phi(z \mid x)}\, dz = \log \mathbb{E}_{q_\phi(z|x)} \left[\frac{p_\theta(x, z)}{q_\phi(z \mid x)} \right]$$

(Trick to introduce q)

$$\log p_\theta(x) \;\geq\; \mathbb{E}_{q_\phi(z|x)}[\log p_\theta(x, z) - \log q_\phi(z \mid x)] \;\equiv\; \mathcal{L}(\theta, \phi; x)$$

(Jensen's inequality)

ELBO

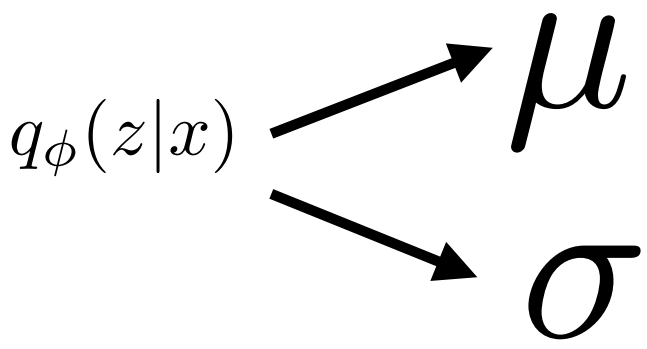
Reconstruction term Regularization term

$$\mathcal{L}(\theta, \phi; x) = \mathbb{E}_{q_\phi(z|x)}[\log p_\theta(x \mid z)] - \mathrm{KL}(q_\phi(z \mid x) \,\|\, p_\theta(z))$$

VAE loss

regularization term (multivariate gaussian and gaussian prior)

$$q_\phi(z \mid x) = \mathcal{N}\big(\mu_\phi(x), \mathrm{diag}(\sigma^2_\phi(x))\big), \quad p_\theta(z) = \mathcal{N}(0, I) \;\Rightarrow\; \mathrm{KL} = \tfrac{1}{2}\sum_{j=1}^{d}\big(\mu_j^2 + \sigma_j^2 - \log\sigma_j^2 - 1\big)$$



latent space

reconstruction term (multivariate gaussian and gaussian prior)

$$p_\theta(x \mid z) = \mathcal{N}\big(f_\theta(z),\, \sigma^2 I\big) \;\Rightarrow\; \mathbb{E}_{q_\phi}[\log p_\theta(x \mid z)] = -\frac{1}{2\sigma^2}\,\mathbb{E}_{q_\phi}\big[\|x - f_\theta(z)\|^2\big] - \frac{D}{2}\log(2\pi\sigma^2)$$

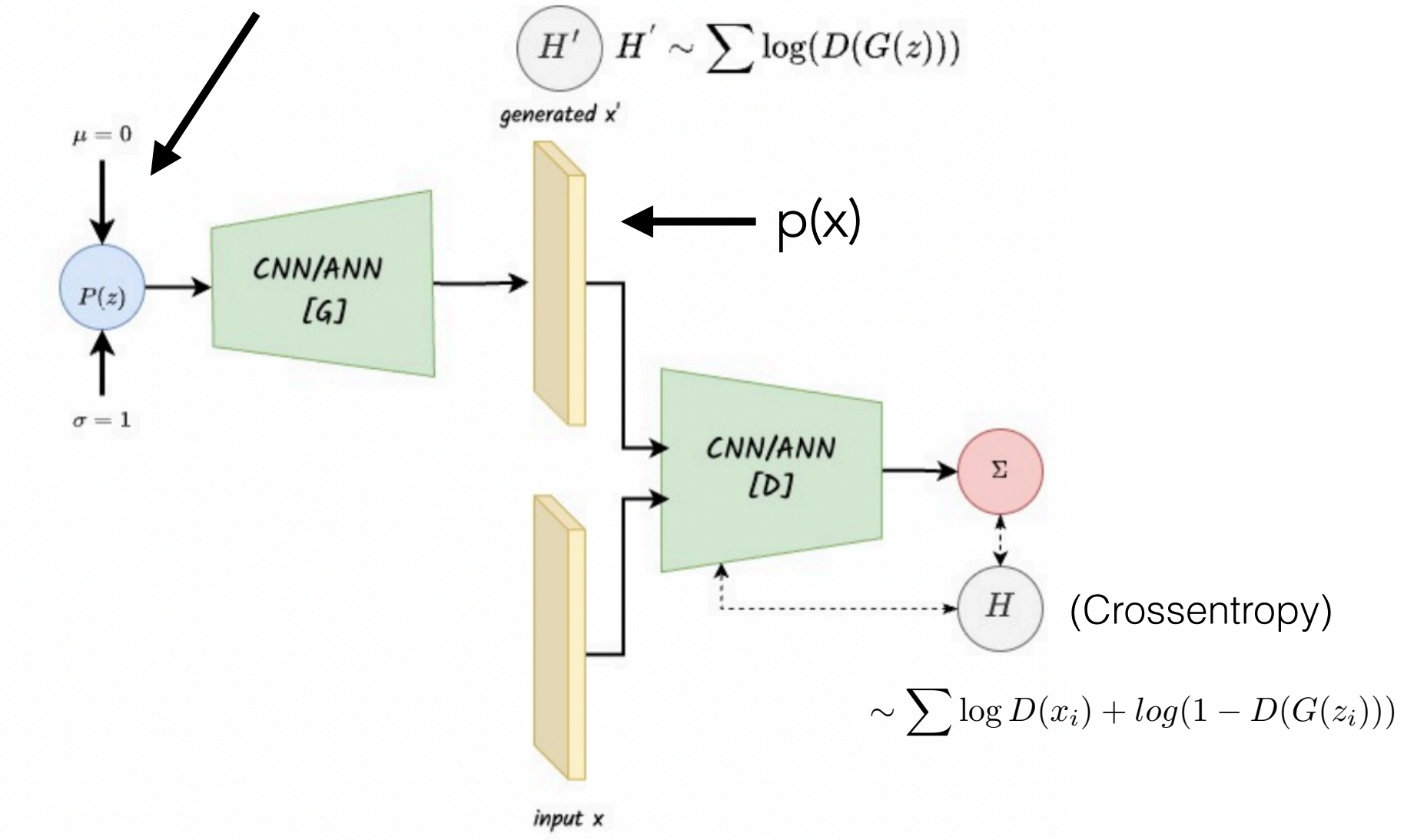
data space

Generative Adversarial Networks

GANs also transform a probability distribution p(z) into p(x)

They convert the problem into a “supervised approach” by using two competing neural networks

The latent variable is here



Generative Adversarial Networks learn the mapping between p(z) and p(x) using an adversarial approach.

discriminator loss

$$\mathcal{L}_D = -\,\mathbb{E}_{x\sim p_{\text{data}}}\big[\log D(x)\big] - \,\mathbb{E}_{z\sim p(z)}\big[\log\big(1 - D(G(z))\big)\big].$$

D maximizes the log-likelihood of the correct label (real/fake).

generator loss

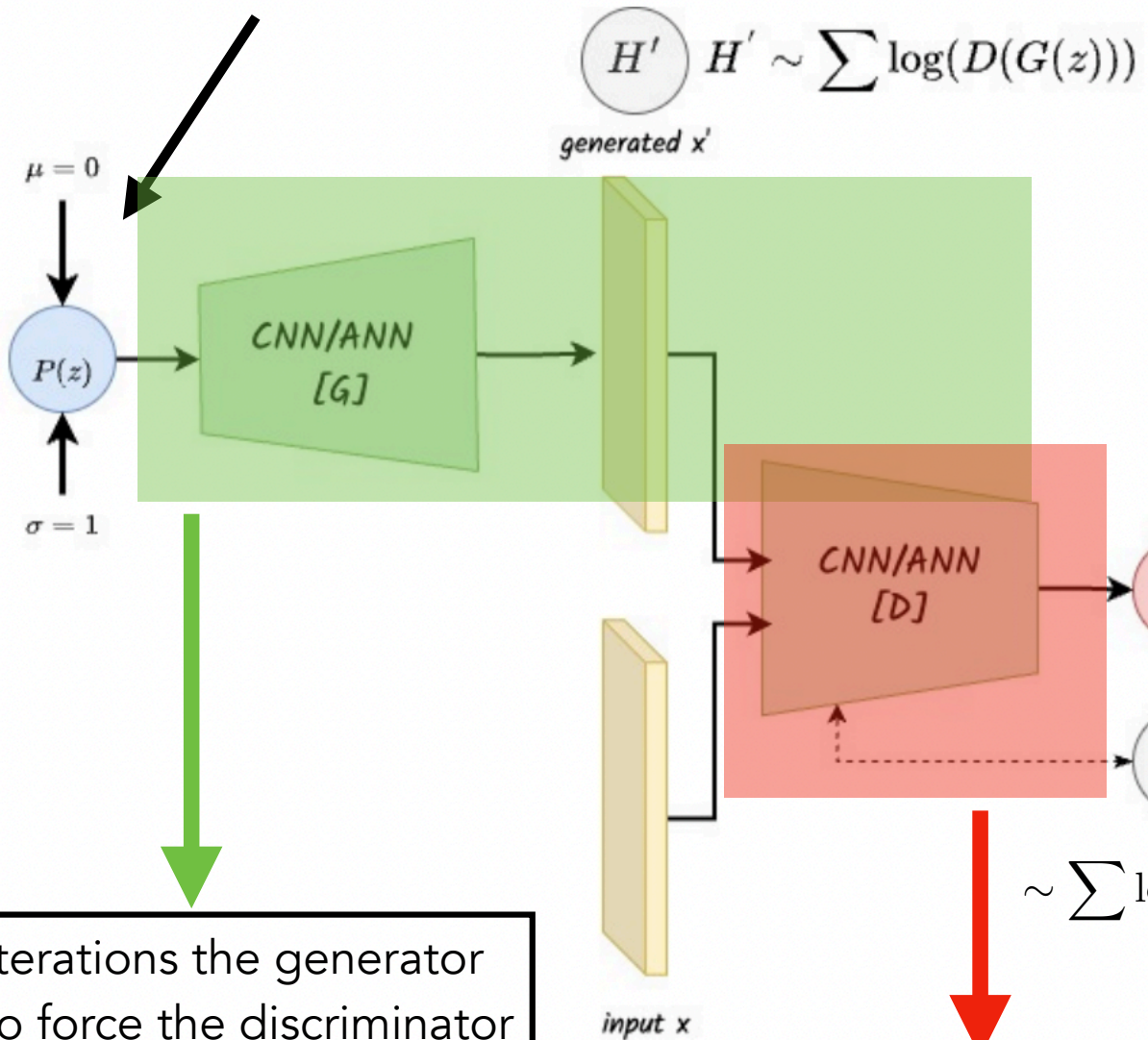
$$\mathcal{L}_G^{\text{NS}} = -\,\mathbb{E}_{z\sim p(z)}\big[\log D(G(z))\big]$$

G tries to make fakes that D labels as real.

$$\mathcal{L}_D = -\frac{1}{m}\sum_{i=1}^{m}\Big[\log D(x_i) + \log\big(1 - D(\tilde{x}_i)\big)\Big], \quad \mathcal{L}_G^{\text{NS}} = -\frac{1}{m}\sum_{i=1}^{m}\log D(\tilde{x}_i)$$

adversarial training

The latent variable is here



(Crossentropy)

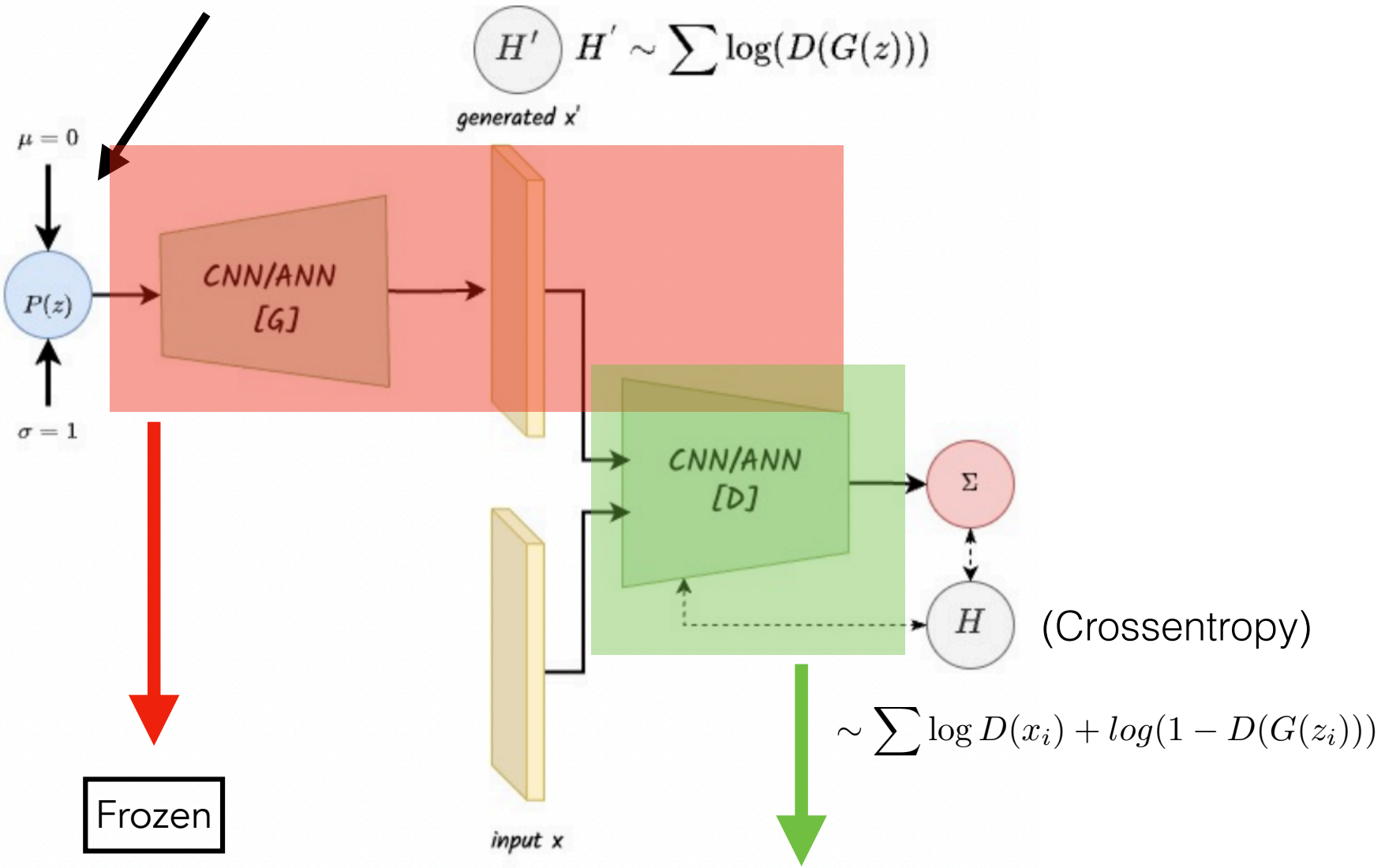
$$\sim \sum \log D(x_i) + log(1 - D(G(z_i)))$$

Every N iterations the generator is trained to force the discriminator to classify as real

Frozen

adversarial training

The latent variable is here

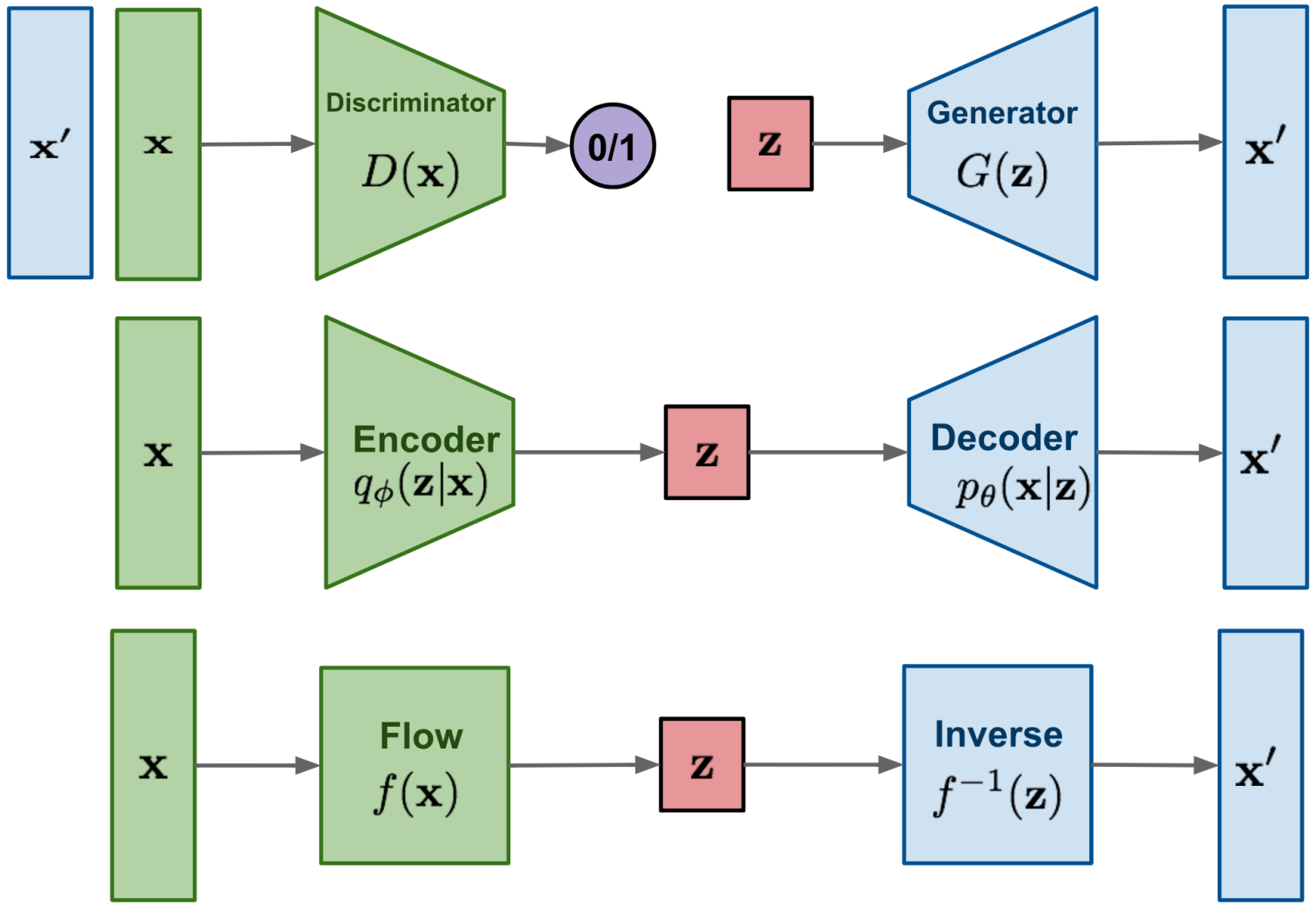


Every N iterations the discriminator is trained to force to distinguish between real and fake

GAN: minimax the classification error loss.

VAE: maximize ELBO.

Flow-based generative models: minimize the negative log-likelihood



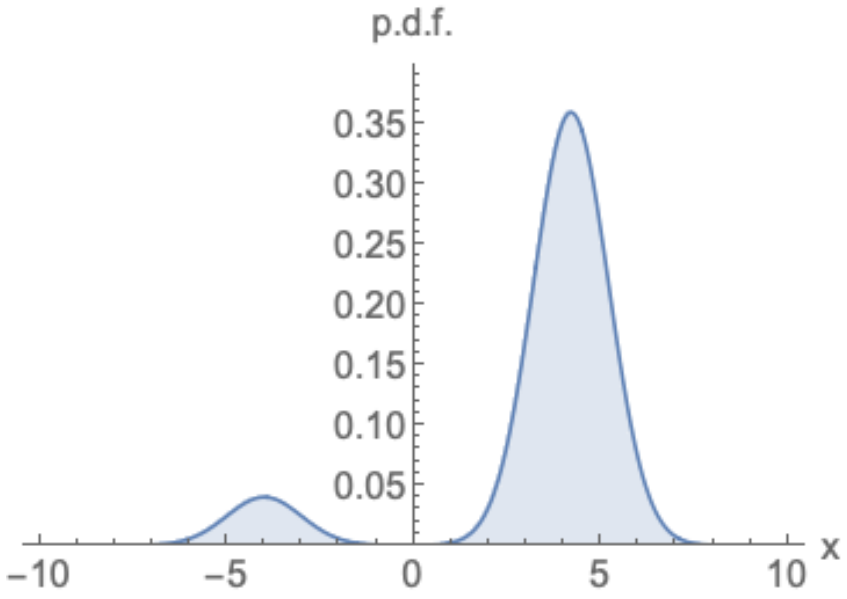
Likelihood-based models estimate either a lower bound (~ELBO) or require restrictions in the NN architectures (~Flows). GANs kind of work around these limitations but adversarial training is unstable.

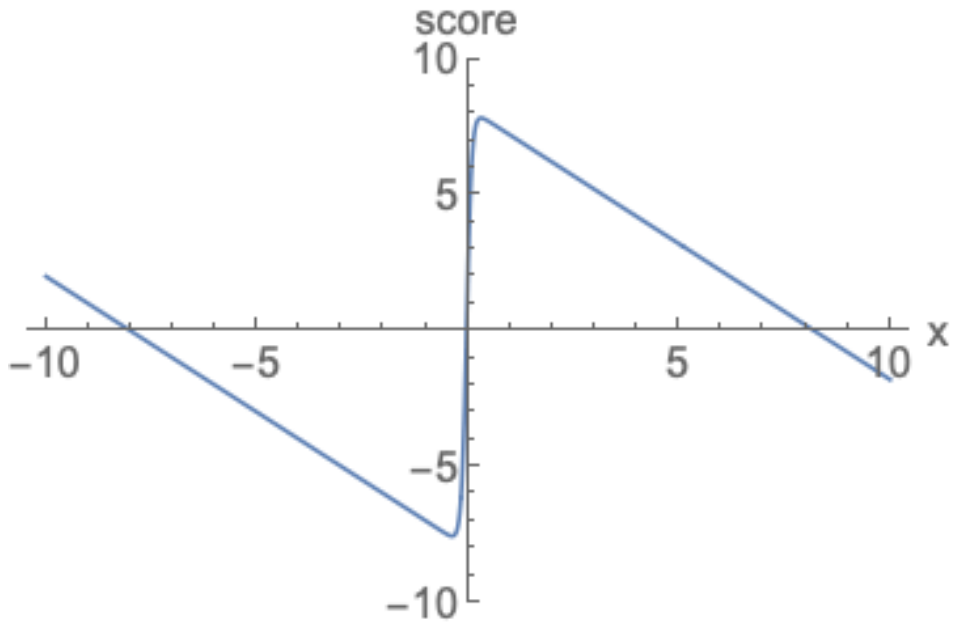
Likelihood-based models estimate either a lower bound (~ELBO) or require restrictions in the NN architectures (~Flows). GANs kind of work around these limitations but adversarial training is unstable.

$$p(x) \longrightarrow \nabla_x \log p(x)$$

s_θ **: Score network, i.e. Neural Network that approximates the gradient of p with respect to the data**

why is the score interesting?





Independent of normalization as compared to p

$$p_\theta(x) = \frac{\exp\big(f_\theta(x)\big)}{Z(\theta)},$$

$$\nabla_x \log p_\theta(x) = \nabla_x f_\theta(x) - \nabla_x Z(\theta)$$

$$\nabla_x \log Z(\theta) = 0$$

One key ingredient is Langevin Dynamics which allows one to sample from a probability distribution using only the score

start with random numbers:

$z_t \sim \mathcal{N}(0,1)$ $\quad$ $\tilde{x_0} \sim \pi(x)$ (arbitrary distribution)

run iteratively for t<T:

$$\tilde{\mathbf{x}}_t = \tilde{\mathbf{x}}_{t-1} + \frac{\epsilon}{2}\nabla_{\mathbf{x}} \log p(\tilde{\mathbf{x}}_{t-1}) + \sqrt{\epsilon}\mathbf{z}_t, \quad \epsilon <<$$

for T large:

$\tilde{x_T}$ is exact sample of $p(x)$

One key ingredient is Langevin Dynamics which allows one to sample from a probability distribution using only the score

start with random numbers:

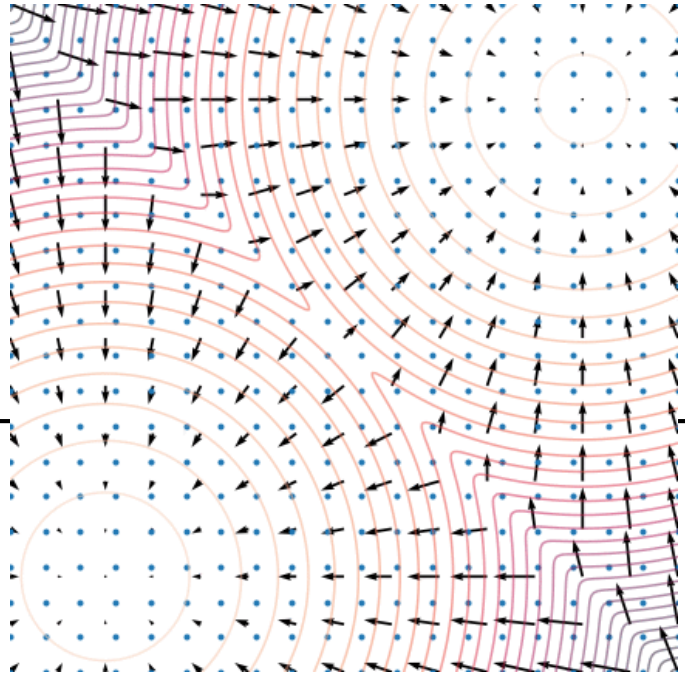
$z_t \sim \mathcal{N}(0,1)$ $\quad$ $\tilde{x_0} \sim \pi(x)$ (arbitrary distribution)

run iteratively for t<T:

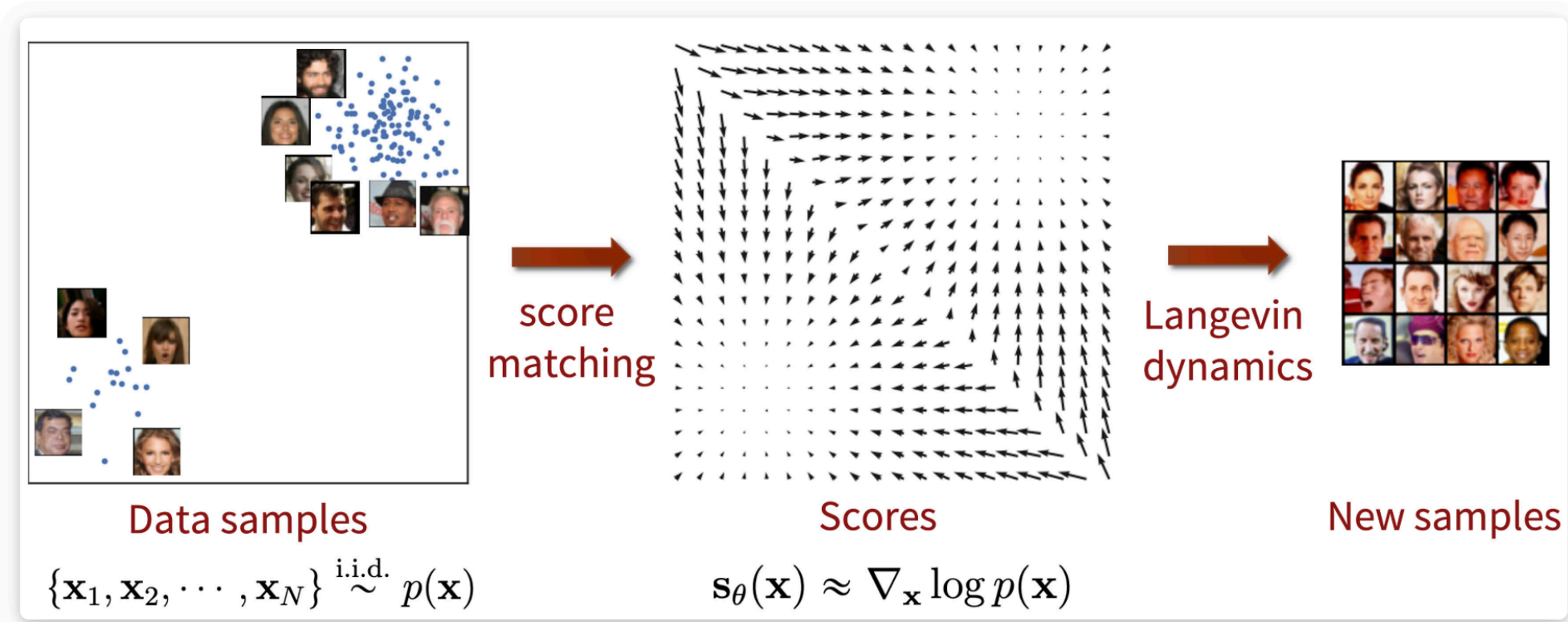
$$\tilde{\mathbf{x}}_t = \tilde{\mathbf{x}}_{t-1} + \frac{\epsilon}{2}\nabla_{\mathbf{x}} \log p(\tilde{\mathbf{x}}_{t-1}) + \sqrt{\epsilon}\mathbf{z}_t, \quad \epsilon <<$$

for T large:

$\tilde{x_T}$ is exact sample of $p(x)$



score based diffusion models



https://yang-song.net/blog/2021/score/

the neural network

We need to estimate the score (with an optimization problem): denoising score matching (DSM)

$$\tilde{x} \mid x \sim \mathcal{N}(x, \sigma^2 I), \qquad \tilde{x} = x + \sigma\varepsilon, \quad \varepsilon \sim \mathcal{N}(0, I), q_\sigma(\tilde{x}) = \int p_{\text{data}}(x)\, \mathcal{N}(\tilde{x}; x, \sigma^2 I)\, dx.$$

$$\frac{1}{2}\mathbb{E}_{q_\sigma(\tilde{\mathbf{x}}|\mathbf{x})p_{\text{data}}(\mathbf{x})}\left[\|\mathbf{s}_\theta(\tilde{\mathbf{x}}) - \nabla_{\tilde{\mathbf{x}}} \log q_\sigma(\tilde{\mathbf{x}}|\mathbf{x})\|_2^2\right].$$

$$s_\theta^*(x) = \nabla_x \log q_\sigma(x) \simeq \nabla_x \log p_{data}(x)$$

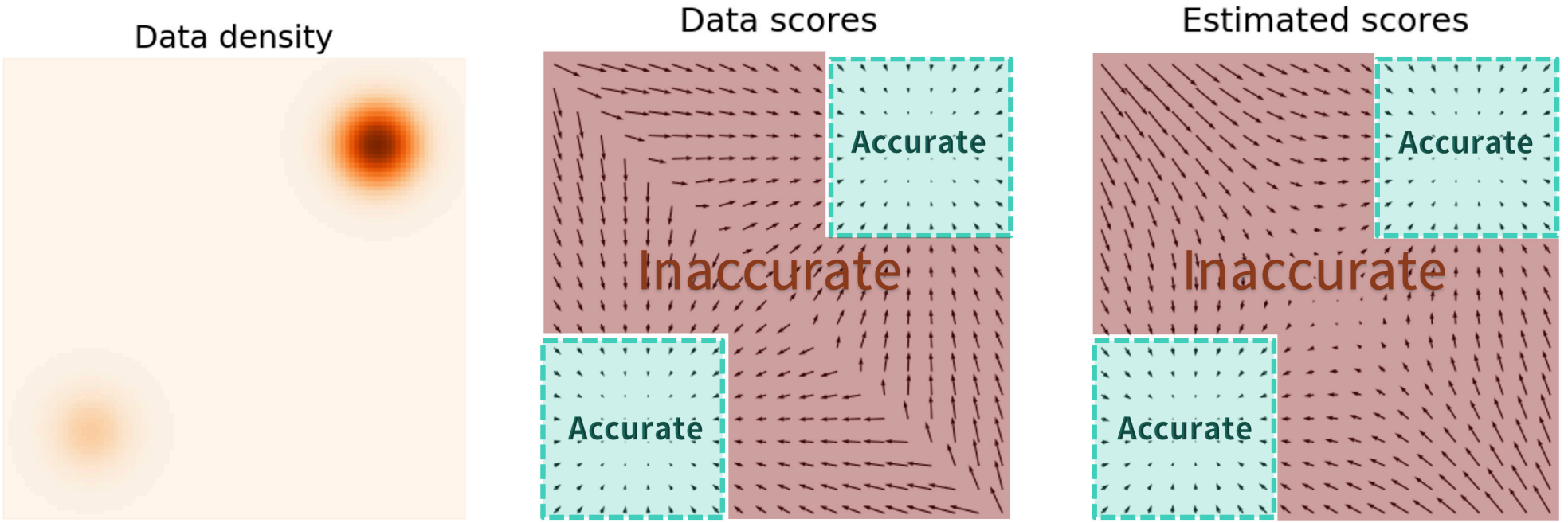
if noise is small

$$\mathcal{J}_{\text{DSM}}(\theta; \sigma) = \mathbb{E}_{x \sim p_{\text{data}}}\, \mathbb{E}_{\tilde{x} \sim \mathcal{N}(x, \sigma^2 I)} \left[\frac{1}{2} \left\| s_\theta(\tilde{x}, \sigma) + \frac{\tilde{x} - x}{\sigma^2} \right\|_2^2\right].$$

$$\mathcal{J}_{\text{DSM}}(\theta; \sigma) = \mathbb{E}_{x \sim p_{\text{data}}}\, \mathbb{E}_{\varepsilon \sim \mathcal{N}(0, I)} \left[\frac{1}{2} \left\| s_\theta(x + \sigma\varepsilon, \sigma) + \frac{\varepsilon}{\sigma} \right\|_2^2\right]$$

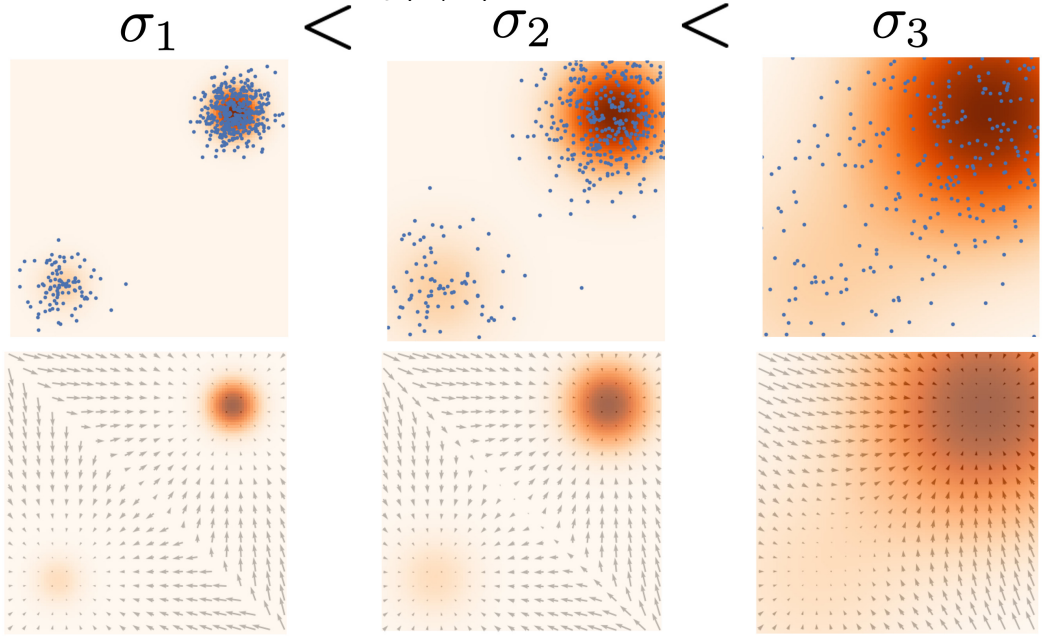
Estimating the score is difficult

In regions of low data density, score matching may not have enough evidence to estimate score functions accurately



noise conditional score network

Let $\{\sigma_i\}_{i=1}^L$ be a positive geometric sequence that satisfies $\frac{\sigma_1}{\sigma_2} = \cdots = \frac{\sigma_{L-1}}{\sigma_L} > 1$. Let $q_\sigma(\mathbf{x}) \triangleq \int p_{\text{data}}(\mathbf{t})\mathcal{N}(\mathbf{x} \mid \mathbf{t}, \sigma^2 I)\mathrm{d}\mathbf{t}$ denote the perturbed data distribution. We choose the noise levels $\{\sigma_i\}_{i=1}^L$ such that σ_1 is large enough to mitigate the difficulties discussed in Section 3, and σ_L is small enough to minimize the effect on data. We aim to train a conditional score network to jointly estimate the scores of all perturbed data distributions, *i.e.*, $\forall \sigma \in \{\sigma_i\}_{i=1}^L : \mathbf{s}_{\boldsymbol{\theta}}(\mathbf{x}, \sigma) \approx \nabla_{\mathbf{x}} \log q_\sigma(\mathbf{x})$. Note that $\mathbf{s}_{\boldsymbol{\theta}}(\mathbf{x}, \sigma) \in \mathbb{R}^D$ when $\mathbf{x} \in \mathbb{R}^D$. We call $\mathbf{s}_{\boldsymbol{\theta}}(\mathbf{x}, \sigma)$ a *Noise Conditional Score Network (NCSN)*.



The loss function for a given sigma is:

$$\ell(\boldsymbol{\theta}; \sigma) \triangleq \frac{1}{2}\mathbb{E}_{p_{\text{data}}(\mathbf{x})}\mathbb{E}_{\tilde{\mathbf{x}} \sim \mathcal{N}(\mathbf{x}, \sigma^2 I)}\left[\left\|\mathbf{s}_{\boldsymbol{\theta}}(\tilde{\mathbf{x}}, \sigma) + \frac{\tilde{\mathbf{x}} - \mathbf{x}}{\sigma^2}\right\|_2^2\right].$$

And then combined:

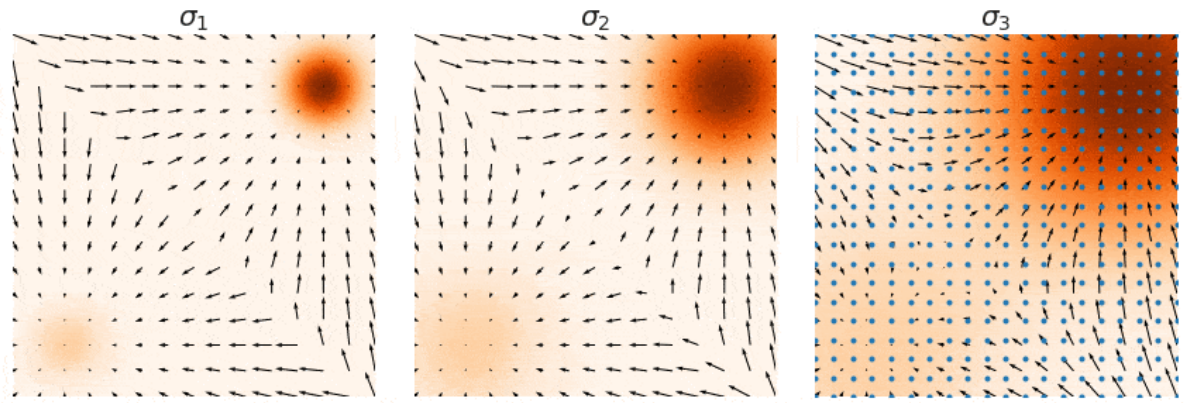
$$\mathcal{L}(\boldsymbol{\theta}; \{\sigma_i\}_{i=1}^L) \triangleq \frac{1}{L}\sum_{i=1}^{L} \lambda(\sigma_i)\ell(\boldsymbol{\theta}; \sigma_i),$$

annealed Langevin dynamics

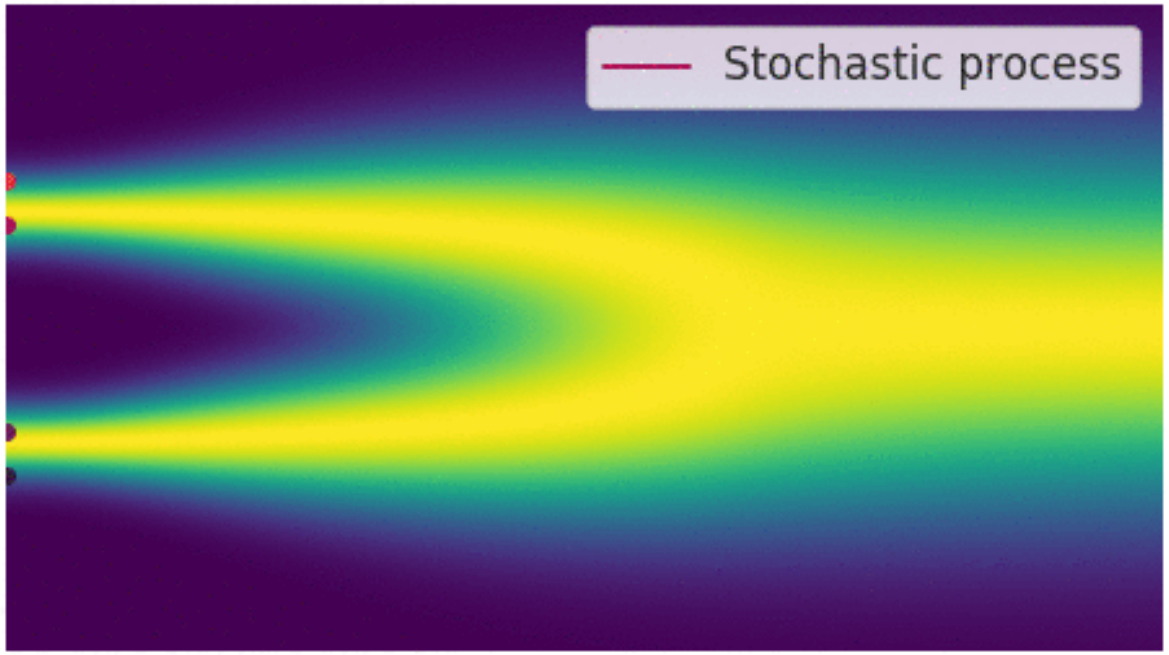
Algorithm 1 Annealed Langevin dynamics.

Require: $\{\sigma_i\}_{i=1}^{L}, \epsilon, T$.

Initialize $\tilde{\mathbf{x}}_0$
for $i \leftarrow 1$ to L **do**
  $\alpha_i \leftarrow \epsilon \cdot \sigma_i^2/\sigma_L^2$ ▷ α_i is the step size.
  for $t \leftarrow 1$ to T **do**
    Draw $\mathbf{z}_t \sim \mathcal{N}(0, I)$
    $\tilde{\mathbf{x}}_t \leftarrow \tilde{\mathbf{x}}_{t-1} + \frac{\alpha_i}{2}\mathbf{s}_{\boldsymbol{\theta}}(\tilde{\mathbf{x}}_{t-1}, \sigma_i) + \sqrt{\alpha_i}\,\mathbf{z}_t$
  end for
  $\tilde{\mathbf{x}}_0 \leftarrow \tilde{\mathbf{x}}_T$
end for
return $\tilde{\mathbf{x}}_T$



Since the distributions $\{q_{\sigma_i}\}_{i=1}^{L}$ are all perturbed by Gaussian noise, their supports span the whole space and their scores are well-defined, avoiding difficulties from the manifold hypothesis. When σ_1 is sufficiently large, the low density regions of $q_{\sigma_1}(\mathbf{x})$ become small and the modes become less isolated. As discussed previously, this can make score estimation more accurate, and the mixing of Langevin dynamics faster. We can therefore assume that Langevin dynamics produce good samples for $q_{\sigma_1}(\mathbf{x})$. These samples are likely to come from high density regions of $q_{\sigma_1}(\mathbf{x})$, which means they are also likely to reside in the high density regions of $q_{\sigma_2}(\mathbf{x})$, given that $q_{\sigma_1}(\mathbf{x})$ and $q_{\sigma_2}(\mathbf{x})$ only slightly differ from each other. As score estimation and Langevin dynamics perform better in high density regions, samples from $q_{\sigma_1}(\mathbf{x})$ will serve as good initial samples for Langevin dynamics of $q_{\sigma_2}(\mathbf{x})$. Similarly, $q_{\sigma_{i-1}}(\mathbf{x})$ provides good initial samples for $q_{\sigma_i}(\mathbf{x})$, and finally we obtain samples of good quality from $q_{\sigma_L}(\mathbf{x})$.



$$dx(t) \;=\; f\big(x(t),t\big)\,dt \;+\; g(t)\,dW_t, \qquad t\in[0,1],\;\; x(0)\sim p_{\text{data}}$$

for example

$$\text{VP (DDPM):}\quad dx \;=\; -\tfrac{1}{2}\,\beta(t)\,x\,dt \;+\; \sqrt{\beta(t)}\,dW_t,$$

- **SDE solution = process:** a continuous family of random variables $\{ x(t) \}_{t \in [0,T]}$ tracing trajectories as time goes from $p_t(x) = p_{x(t)}(x)$.

- **Link to discrete noise scales:** continuous time $t \in [0, T] \leftrightarrow$ discrete index $i = 1, 2, \ldots, L$; likewise densities correspond as $p_t(x) \leftrightarrow p_{\sigma_i}(x)$

- **Start (no noise):** $p_0(x) = p(x)$ (data distribution)

- **End (max noise / prior):** after long enough time, $p_T(x) \approx \pi(x)$ (tractable prior).

- **Finite-step analogy at the end:** $p_T(x) \leftrightarrow p_{\sigma_L}(x)$ with the largest noise level $\sigma_L = \max_{i \in \{1,\ldots,L\}} \sigma_i$.

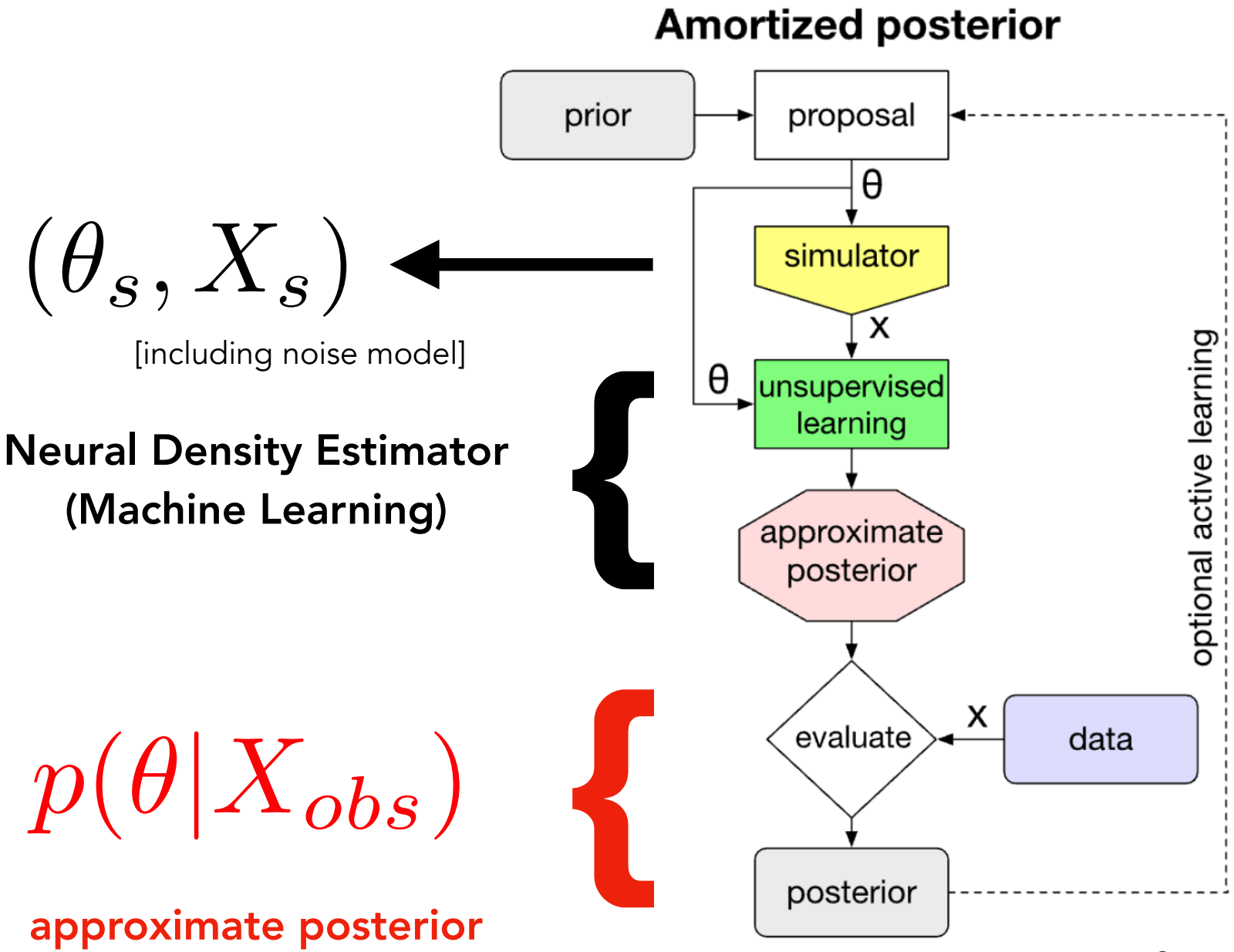
$$s_\theta(x,t) \;\approx\; \nabla_x \log p_t(x), \qquad \min_\theta \; \mathbb{E}_{t\sim\mathcal{U}(0,1)}\Big[\lambda(t)\,\mathbb{E}_{x_0\sim p_{\text{data}},\,\varepsilon\sim\mathcal{N}(0,I)}\Big\|s_\theta(x_t,t) - \nabla_x \log q_t(x_t \mid x_0)\Big\|_2^2\Big].$$

$$\text{VP:} \quad x_t = \sqrt{\bar\alpha(t)}x_0 + \sqrt{1-\bar\alpha(t)}\,\varepsilon \;\Rightarrow\; \nabla_x \log q_t(x_t \mid x_0) = -\frac{\varepsilon}{\sqrt{1-\bar\alpha(t)}}.$$

$$x \leftarrow x + \eta\, s_\theta(x,t) + \sqrt{2\eta}\,\xi, \qquad \xi \sim \mathcal{N}(0,I).$$

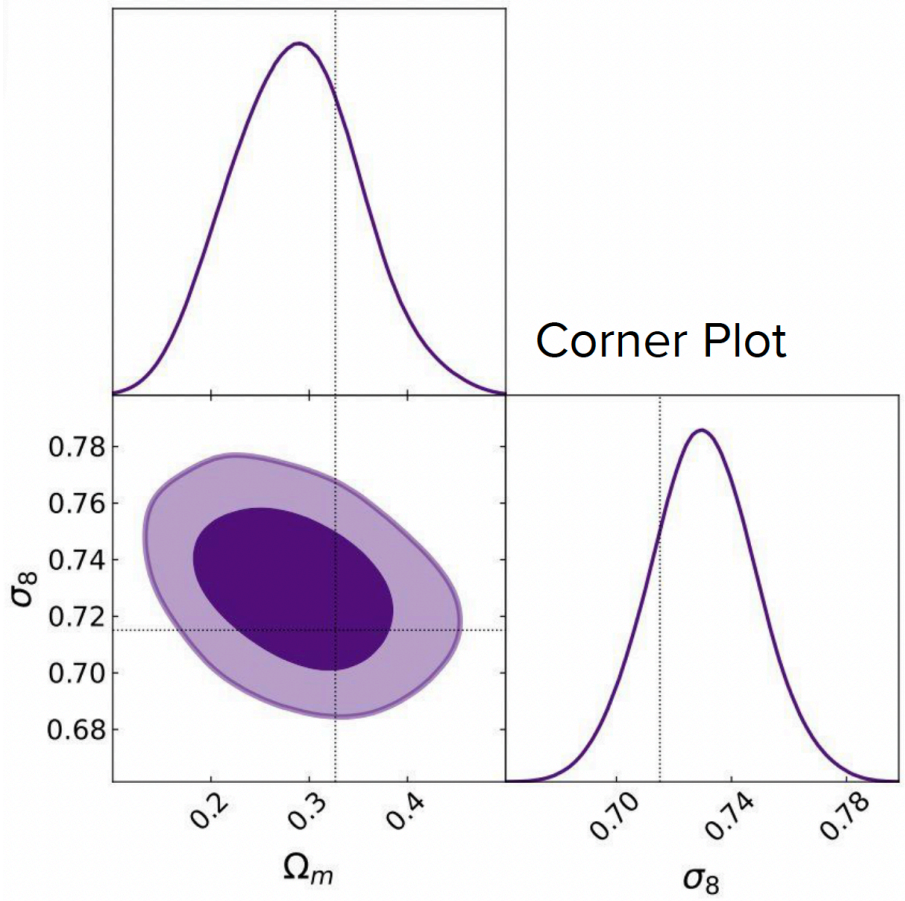
Back to SBI

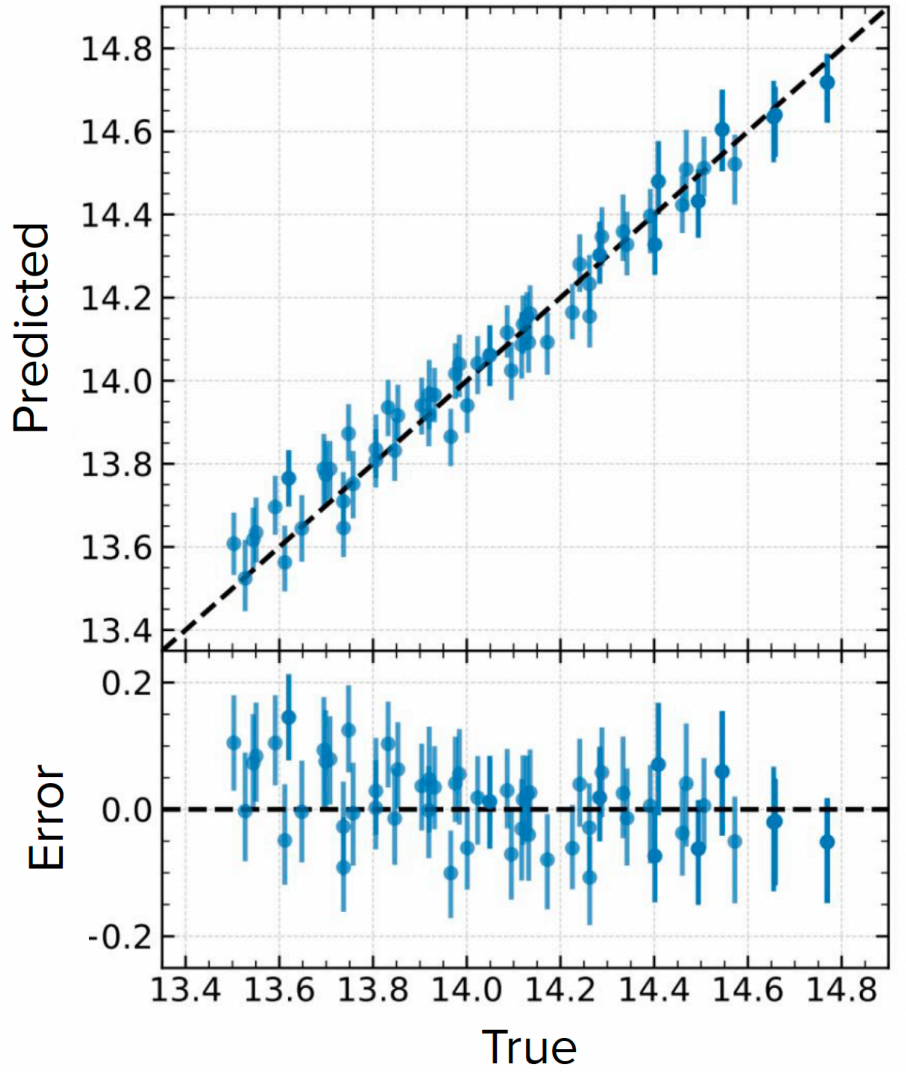
SBI: All you need is a **good** forward model



Cranmer+2020

Validation

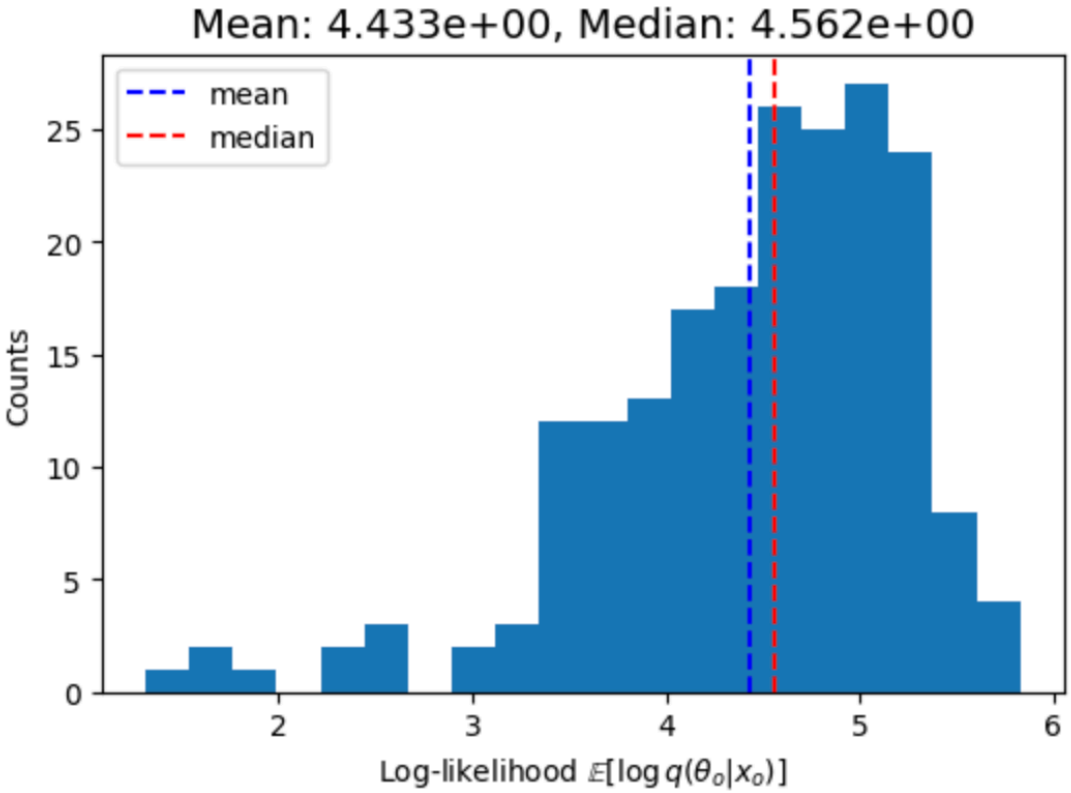




*adapted from M. Ho (SBI conference)

Cumulative Likelihood of the Test Dataset

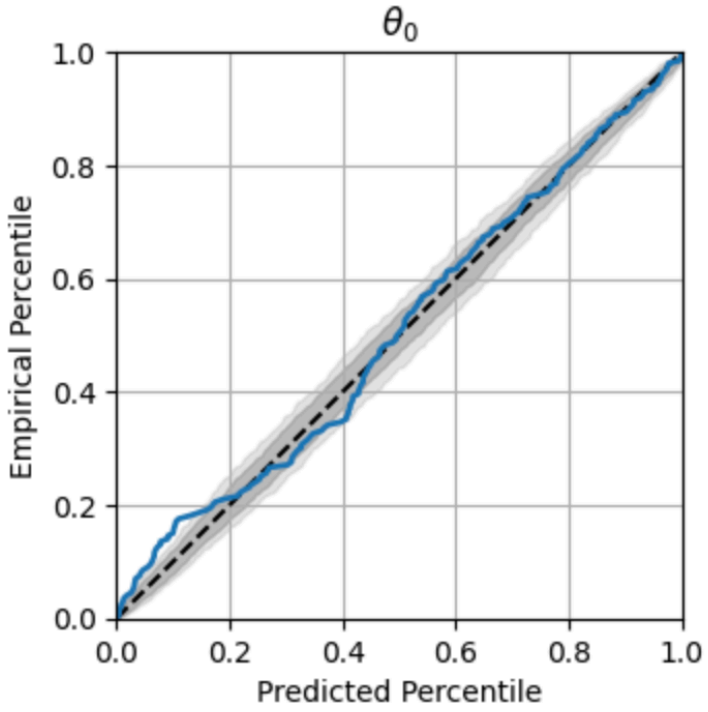
$$\prod_{i=1}^{N_{test}} \hat{P}(\theta_i|x_i)$$



Large values mean good predictive power

P-P plots

$$\mathrm{PIT}(\theta; x_o) = \int_{-\infty}^{\theta} d\theta\, \hat{P}(\theta \mid x_o).$$



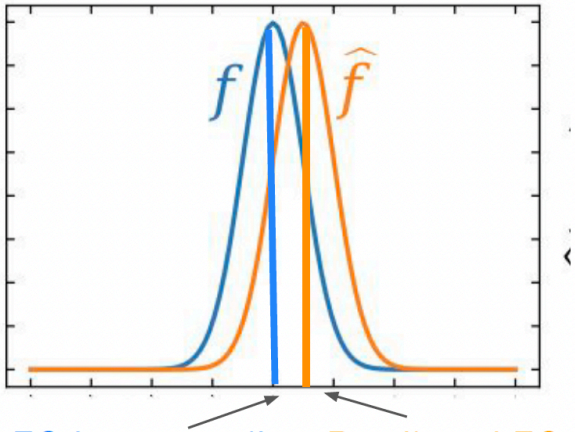
"What percentile level is my posterior model assigning to the true value, and with what frequency does this occur in the test set, e.g. **we should predict the true value below the 50-th percentile 50% of the time**"

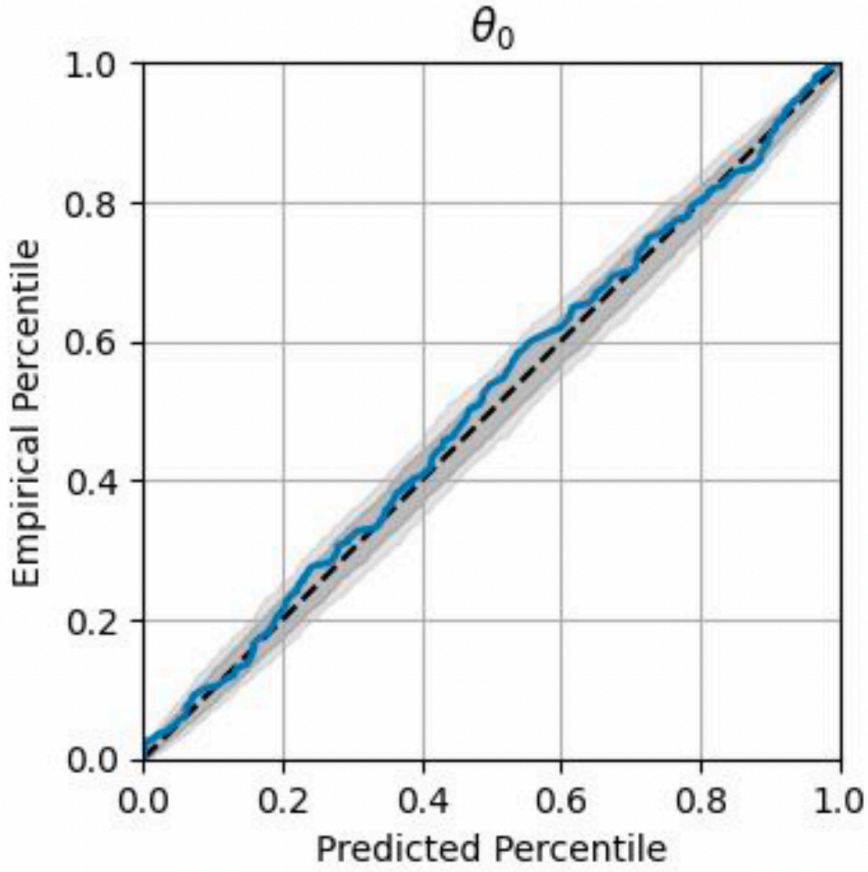
Ho+24

Validation

PIT tests / P-P plots

- Tests of marginal coverage

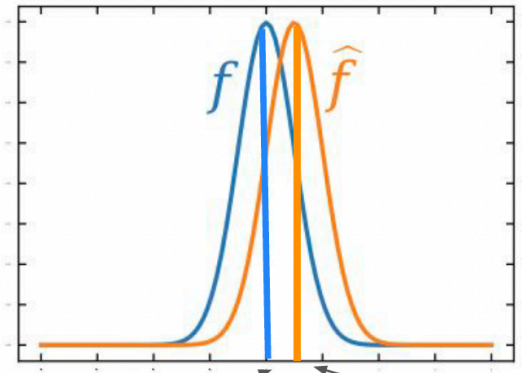


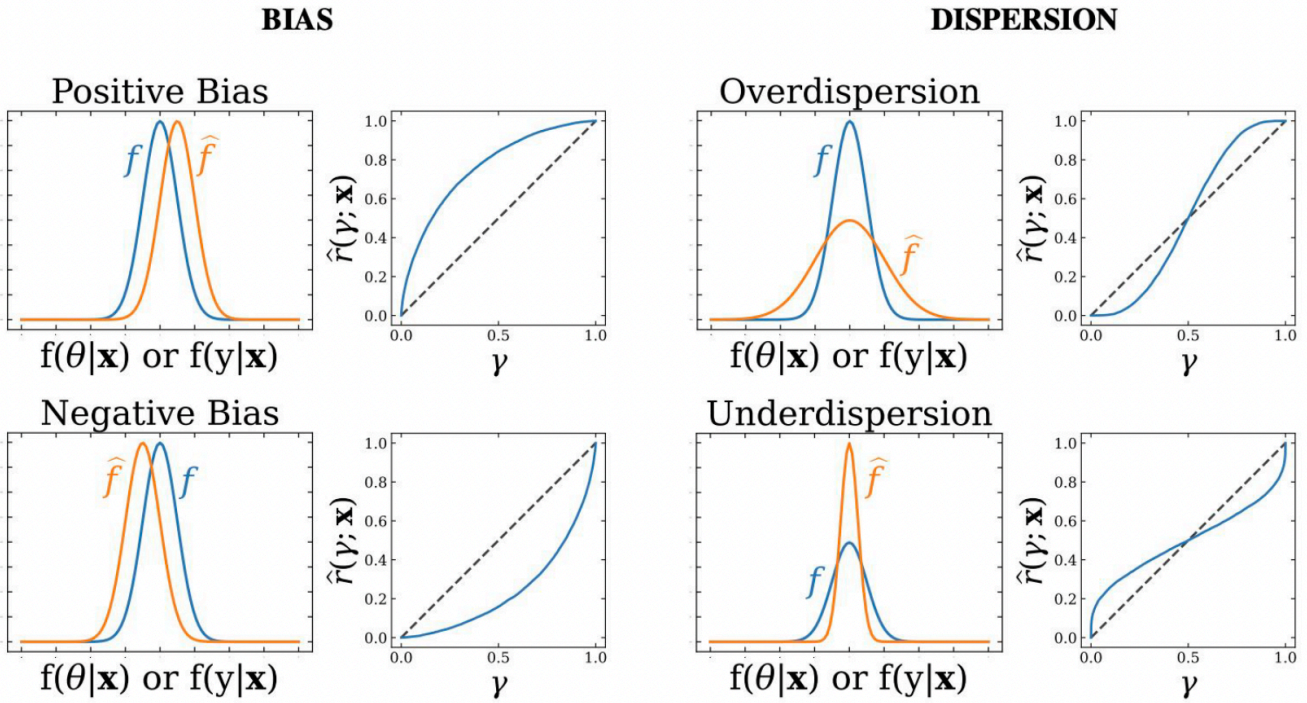


*adapted from M. Ho (SBI conference)

PIT tests / P-P plots

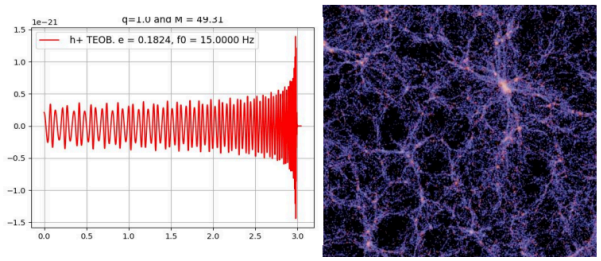
- Tests of marginal coverage

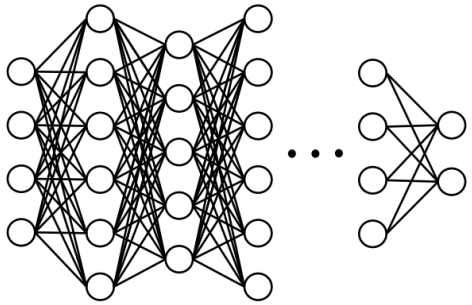


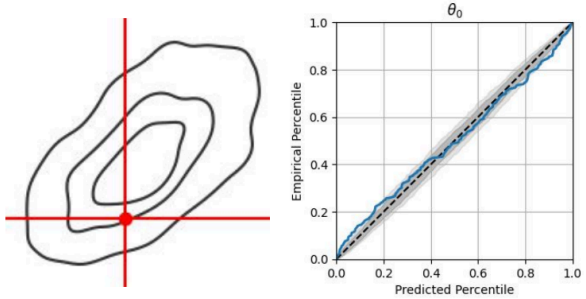


*adapted from M. Ho (SBI conference)

SBI pipeline







I: Data

- ★ Gather data-parameter pairs.
- ★ Load from file or specify a simulator.

II: Inference

- ★ Construct neural architectures.
- ★ Configure training algorithm.
- ★ Minimize losses.

III: Validation

- ★ Measure constraining power.
- ★ Check calibration of posterior error bars.
- ★ Apply to real (unlabeled) data.

Ho+24

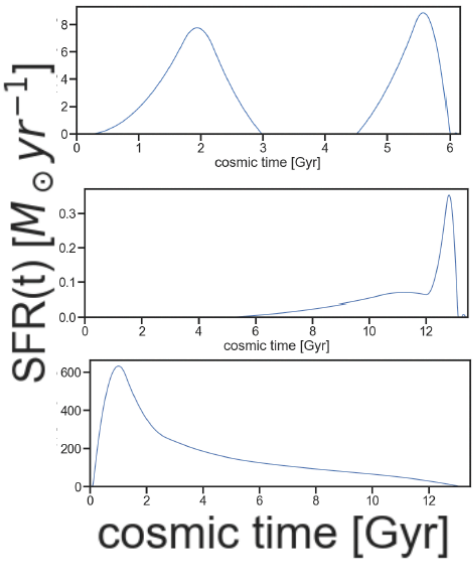
Examples of applications

Forward Model

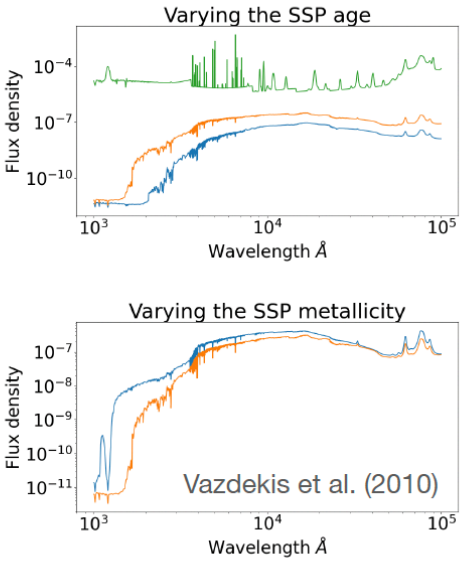
θ [SFR, M*, Av, SFH, Z]

$$F_\lambda(t) = \int_0^t \psi(t-t') \, SSP_\lambda(t', Z(t-t')) \, e^{-\tau(t')} dt'$$

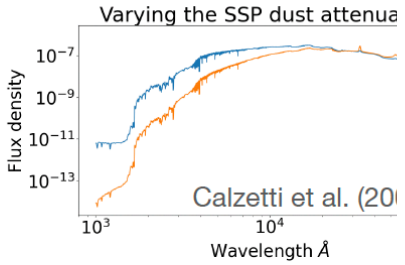
Non-parametric SFHs with a Dirichlet prior



MILES SSP models



Dust attenuation model

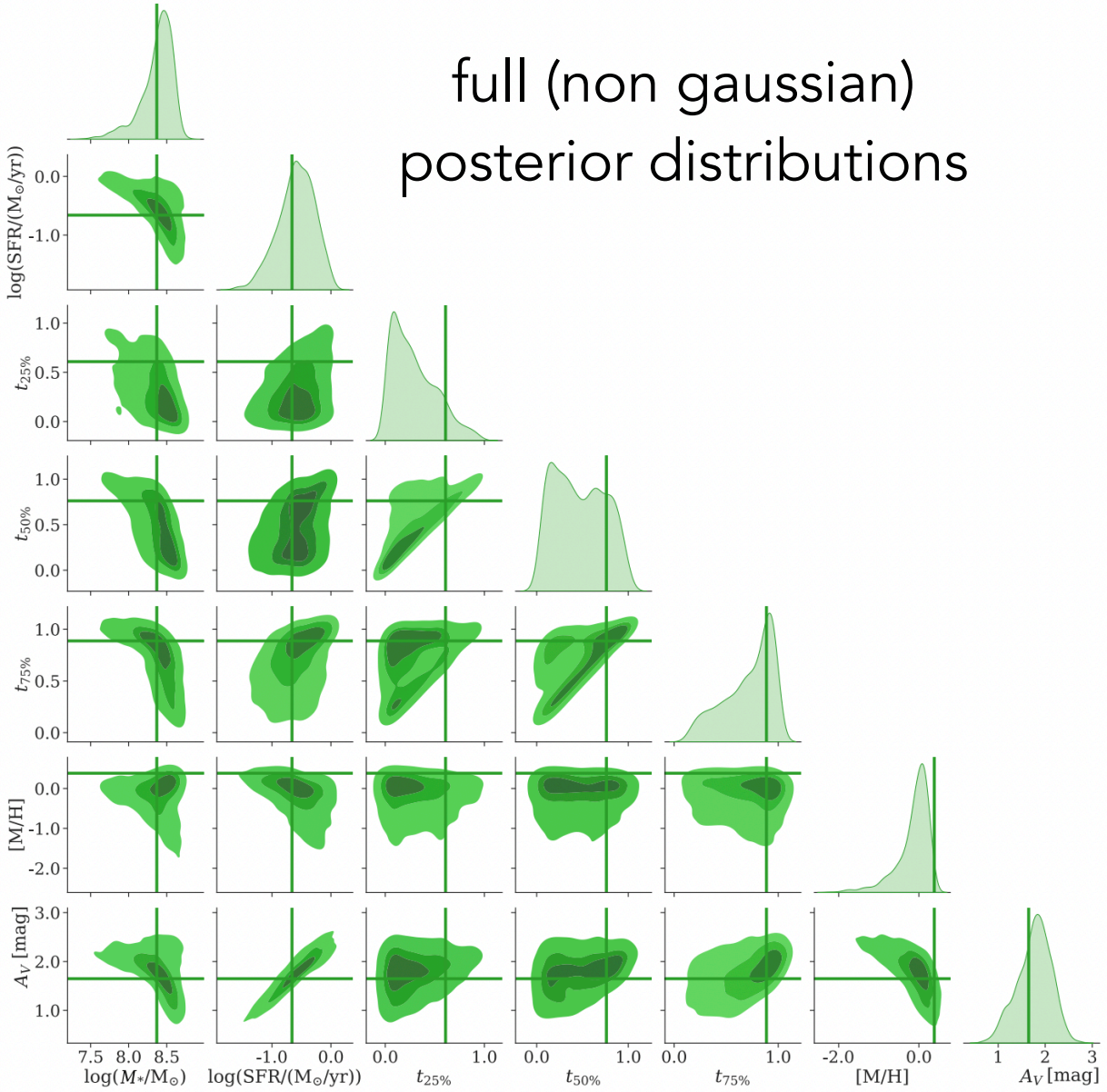


X_s

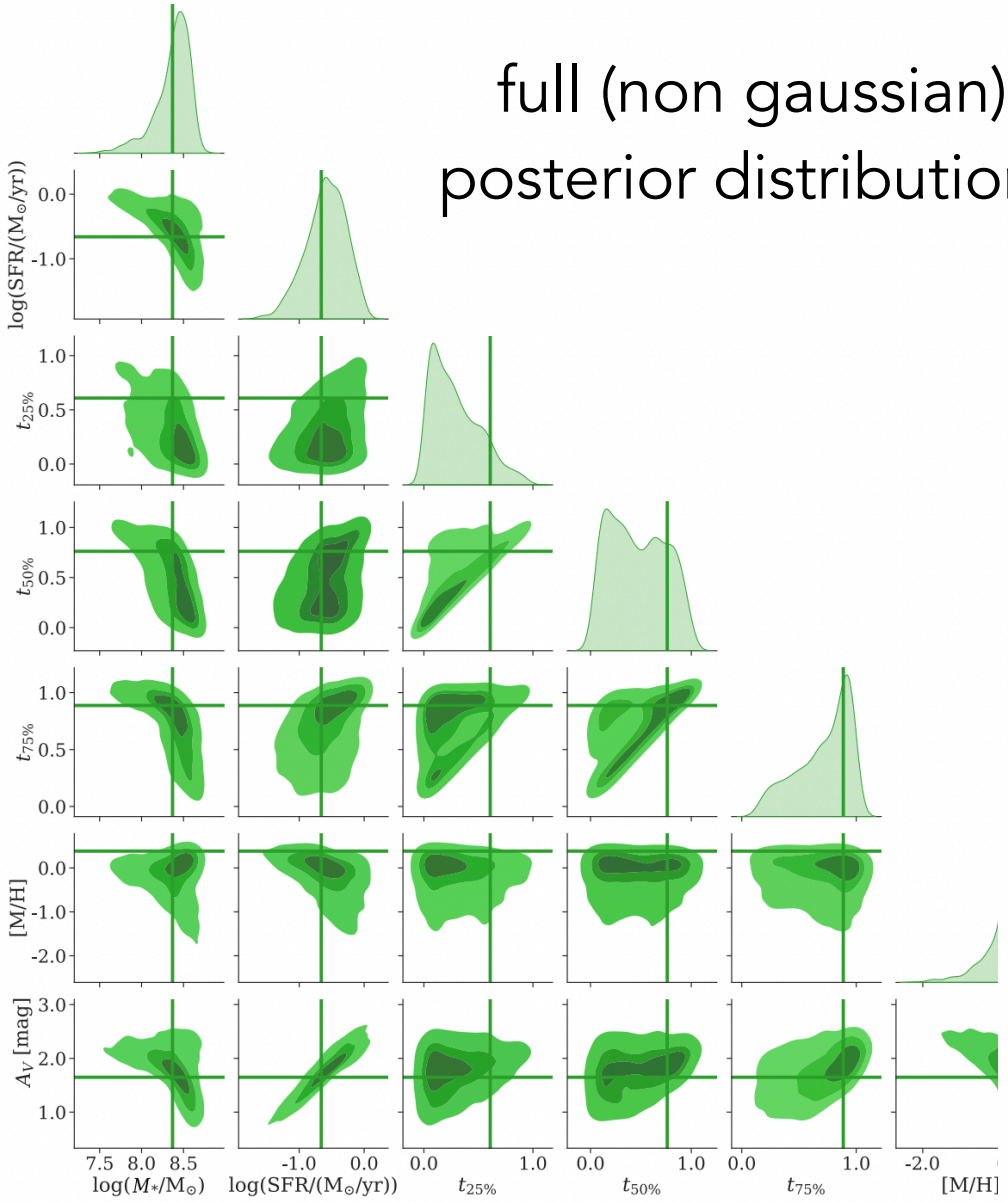
fluxes + noise model

Iglesias-Navarro,MHC+24, 25

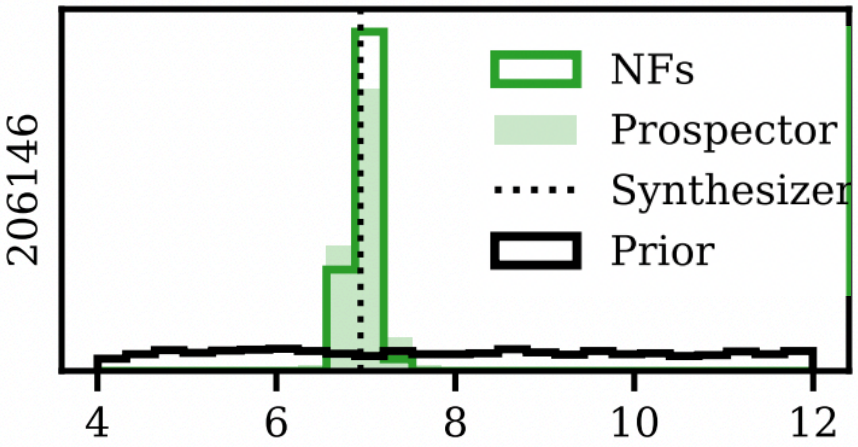
full (non gaussian) posterior distributions



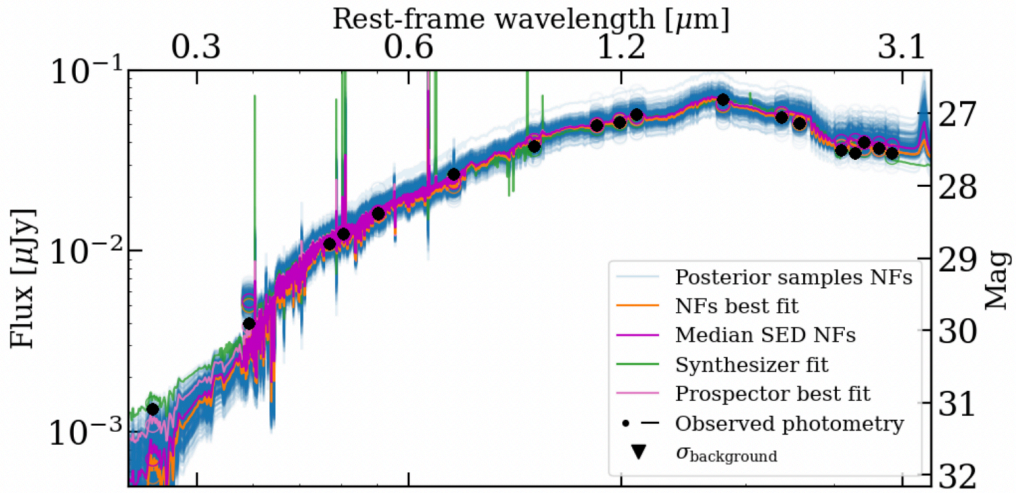
full (non gaussian)
posterior distributions



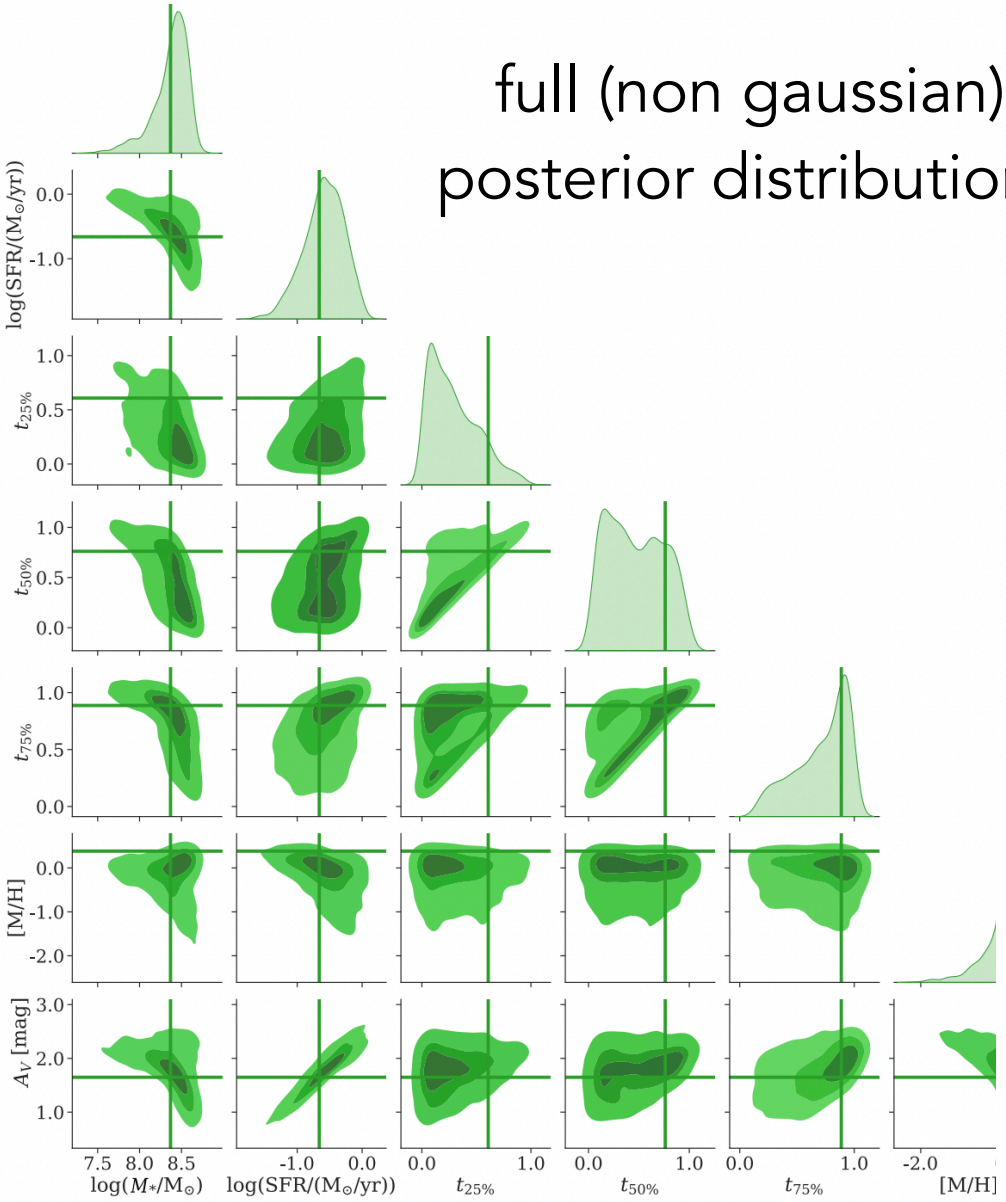
in good agreement with
MCMC



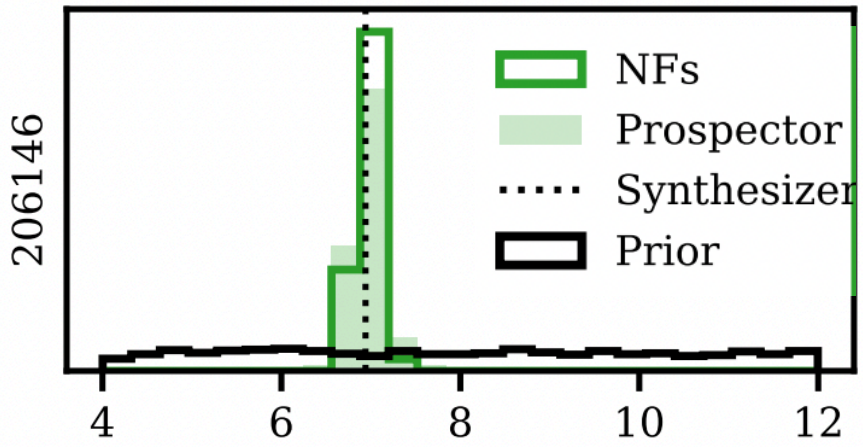
254985

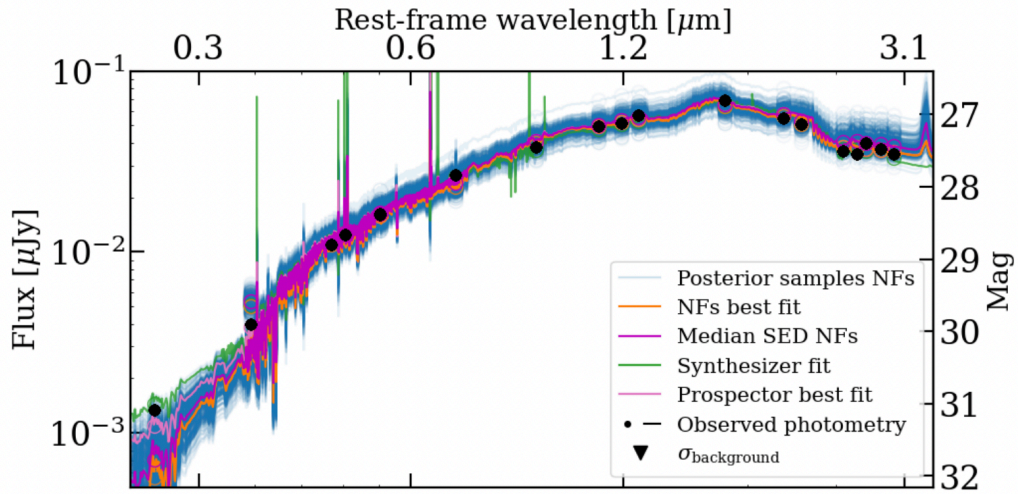


full (non gaussian)
posterior distributions

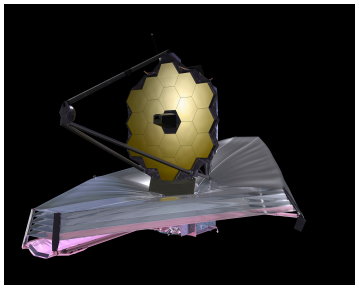


in good agreement with
MCMC





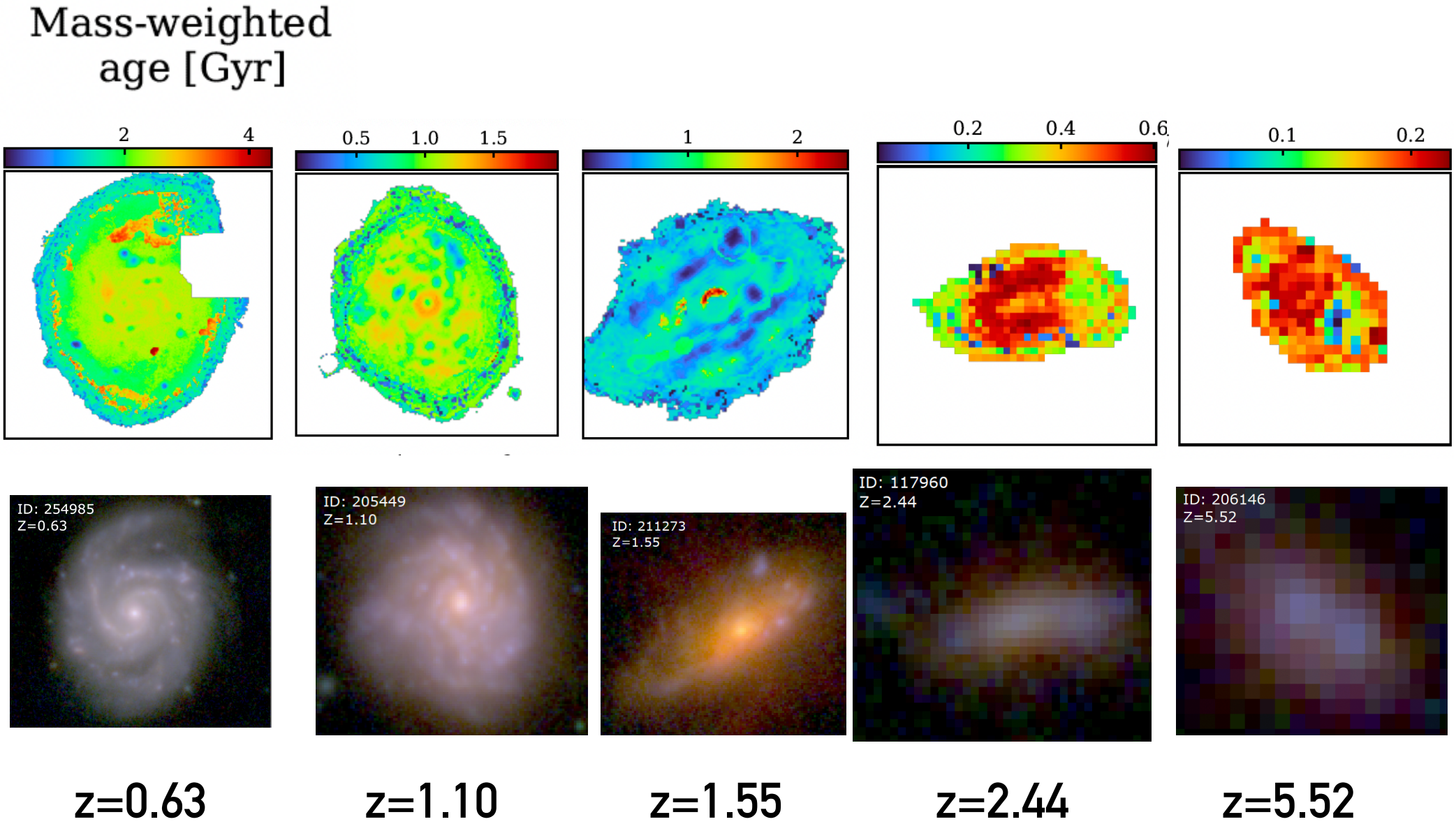
~ 3 orders of magnitude faster than Bayesian SED fitting



JWST + SBI enable
resolved stellar populations
of galaxies up to very high redshift

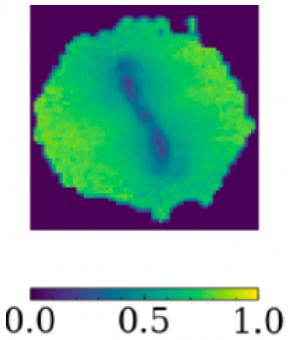
Fit every pixel
(~10k)

ML based SBI of resolved stellar populations

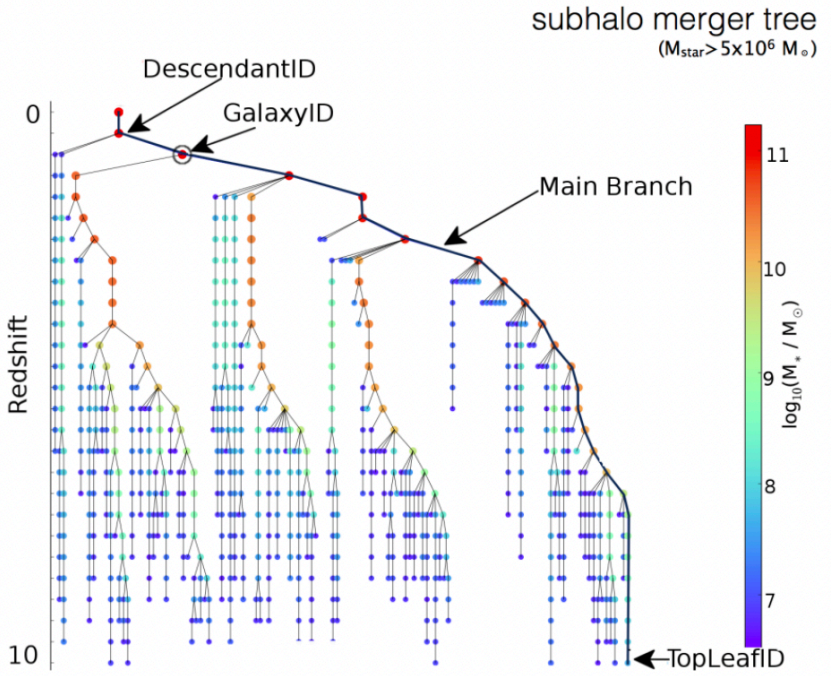


Forward Model: Cosmological simulations

θ

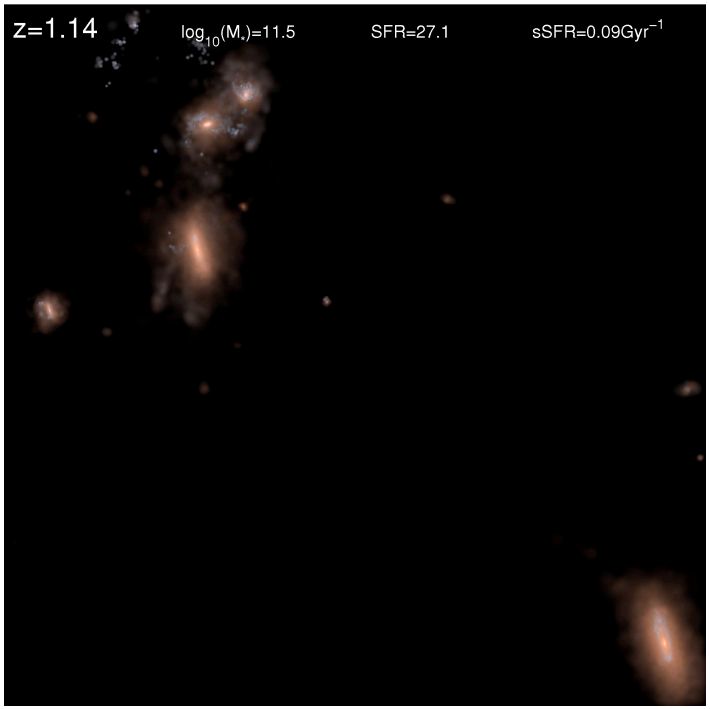


(nD)

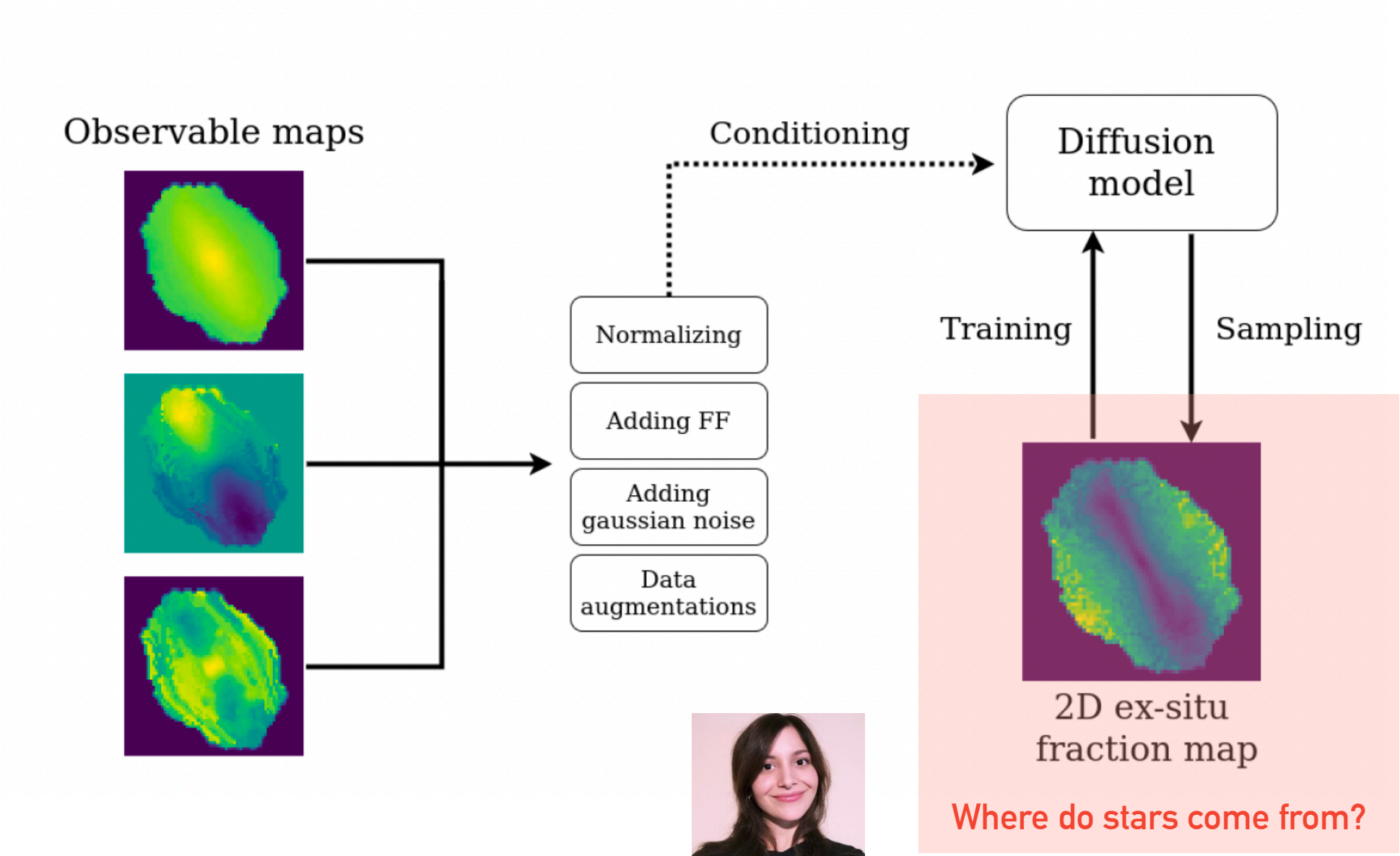


[kinematic and stellar population maps]

(nD)

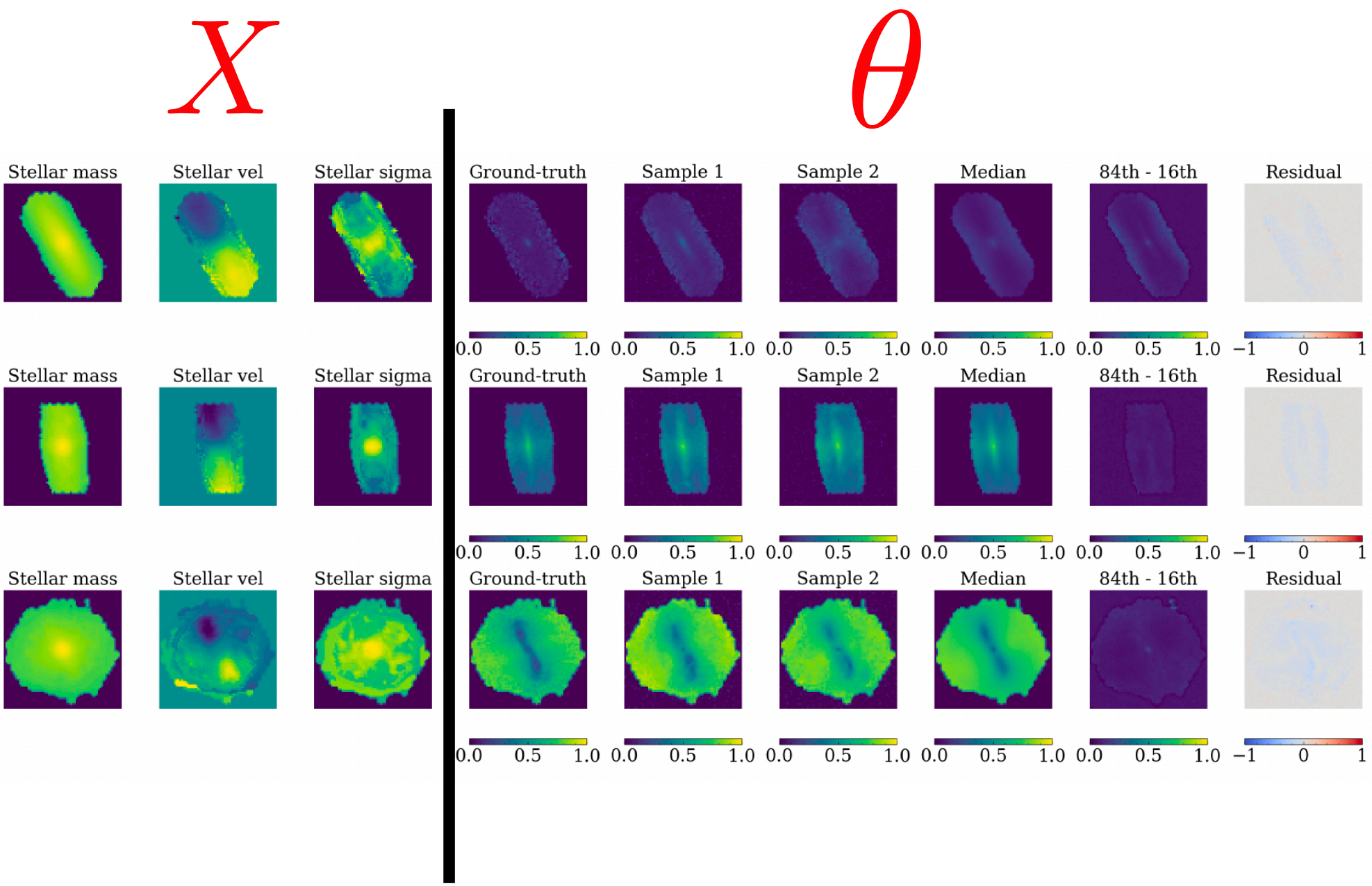


Resolved accretion map of local galaxies via diffusion

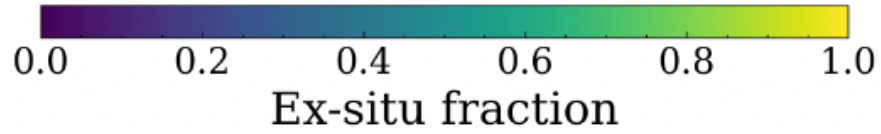


E. Angeloudi

Angeloudi, MHC+25

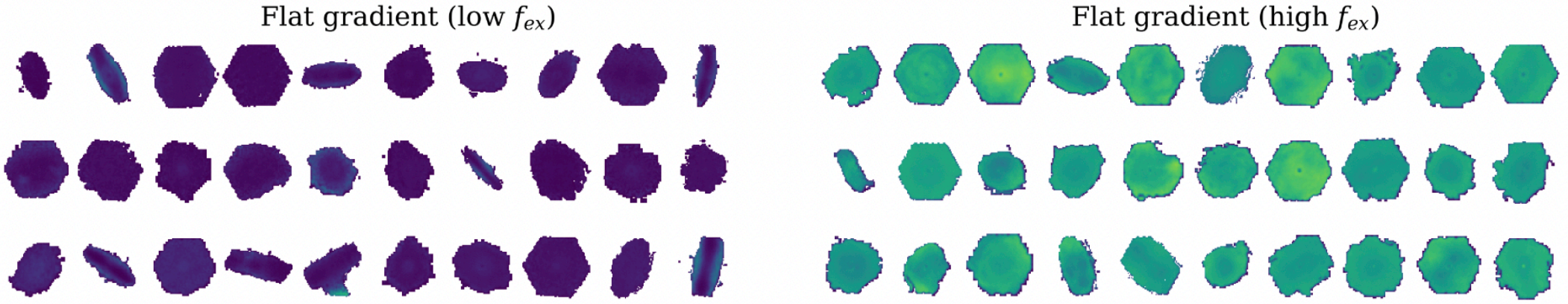


Angeloudi, MHC+25

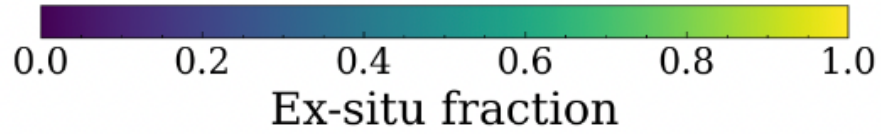


In-situ dominated

merger dominated

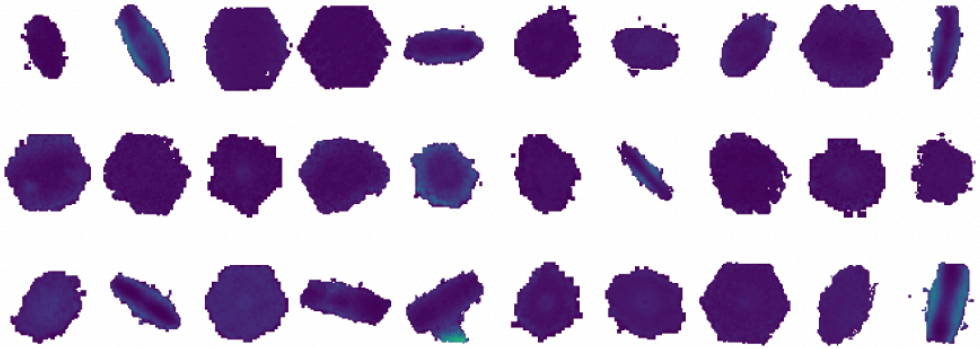


Angeloudi, MHC+25



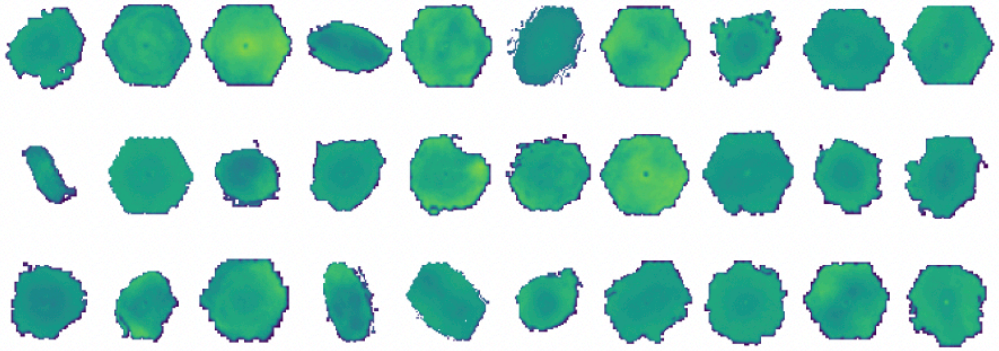
In-situ dominated

Flat gradient (low f_{ex})



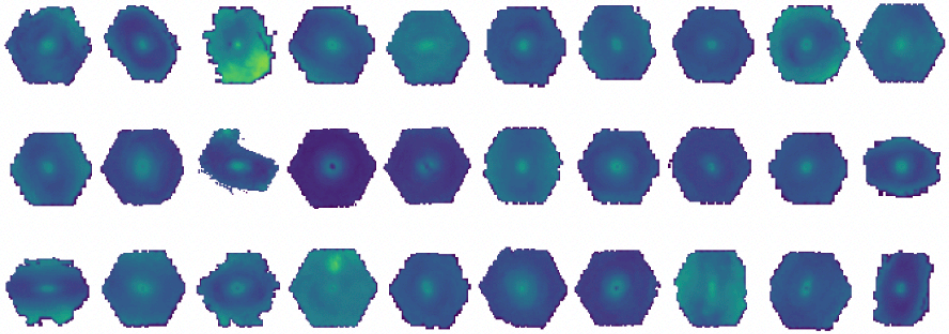
merger dominated

Flat gradient (high f_{ex})



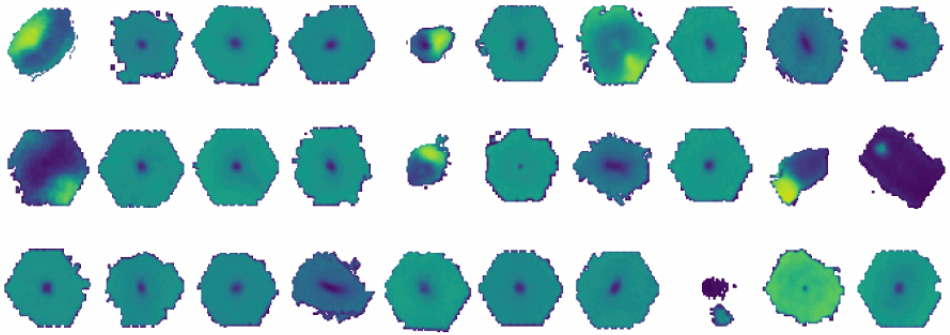
more ex-situ in center

Negative gradient



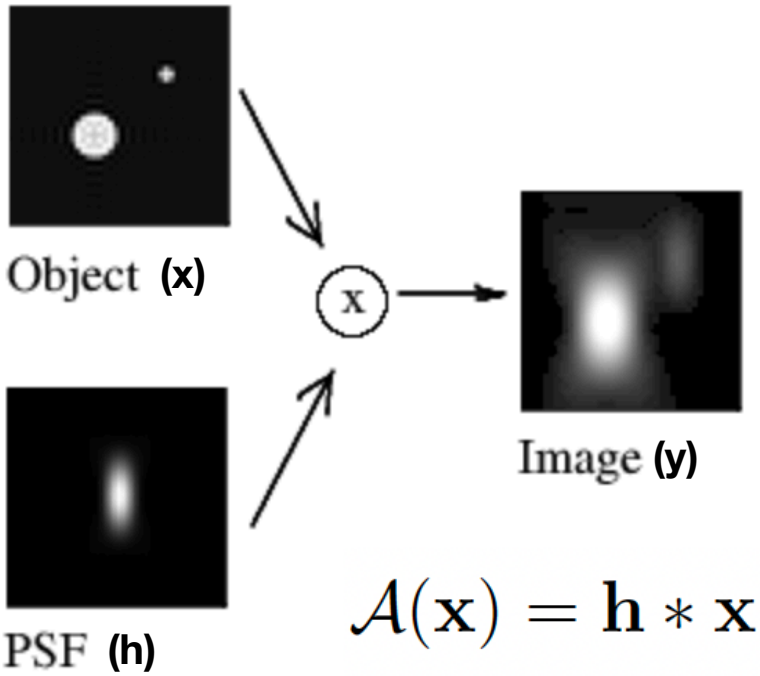
more ex-situ in outskirts

Positive gradient



Angeloudi, MHC+25

Learning the prior: Deconvolution through diffusion

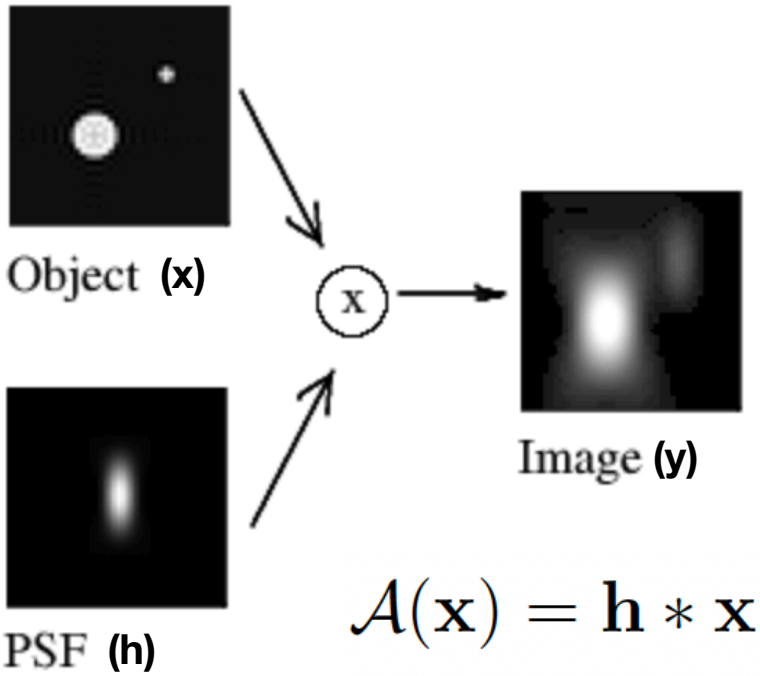


$$\mathcal{A}(\mathbf{x}) = \mathbf{h} * \mathbf{x}$$

$$\mathbf{y} = \mathcal{A}\mathbf{x}_0 + \sigma_y \mathbf{n}$$

Spagnoletti+24

Learning the prior: Deconvolution through diffusion



$$\mathbf{y} = \mathcal{A}\mathbf{x}_0 + \sigma_y \mathbf{n}$$

$$\mathcal{A}(\mathbf{x}) = \mathbf{h} * \mathbf{x}$$

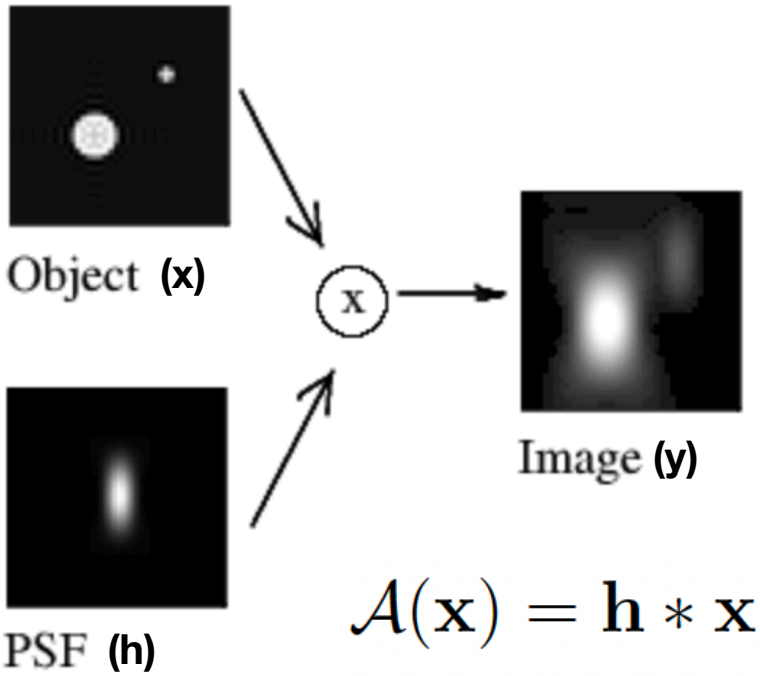
$$\log p(x|y) = \log p(y|x) + \log p(x)$$

Convolution Operator

The space of possible images

Spagnoletti+24

Learning the prior: Deconvolution through diffusion



$$\mathbf{y} = \mathcal{A}\mathbf{x}_0 + \sigma_y \mathbf{n}$$

$$\mathcal{A}(\mathbf{x}) = \mathbf{h} * \mathbf{x}$$

$$\log p(x|y) = \log p(y|x) + \log p(x)$$

Convolution Operator

The space of possible images

neural network

Spagnoletti+24

Learning the prior: Deconvolution through diffusion

$$\mathbf{y} = \mathcal{A}\mathbf{x}_0 + \sigma_y \mathbf{n}, \qquad \mathcal{A}(\mathbf{x}) = \mathbf{h} * \mathbf{x}$$

$$\log p(x|y) = \log p(y|x) + \log p(x)$$

Convolution Operator

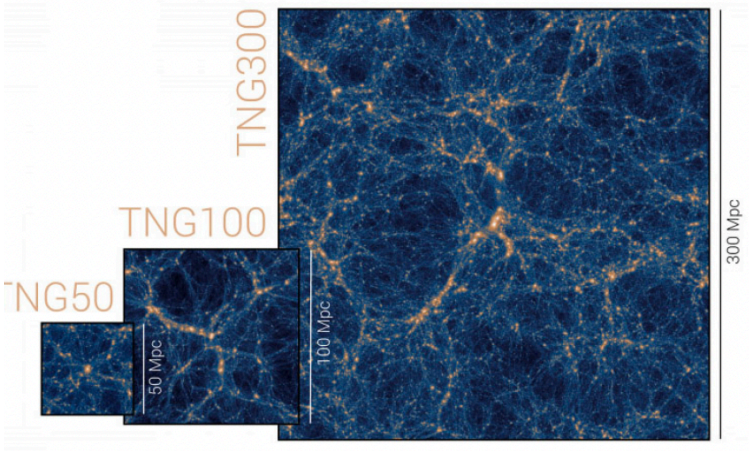
The space of possible images

neural network

Spagnoletti+24

$$p(x)$$

Learned through diffusion models and cosmological simulations



Samples from the prior

Spagnoletti+24

Deconvolved

HSC

HST

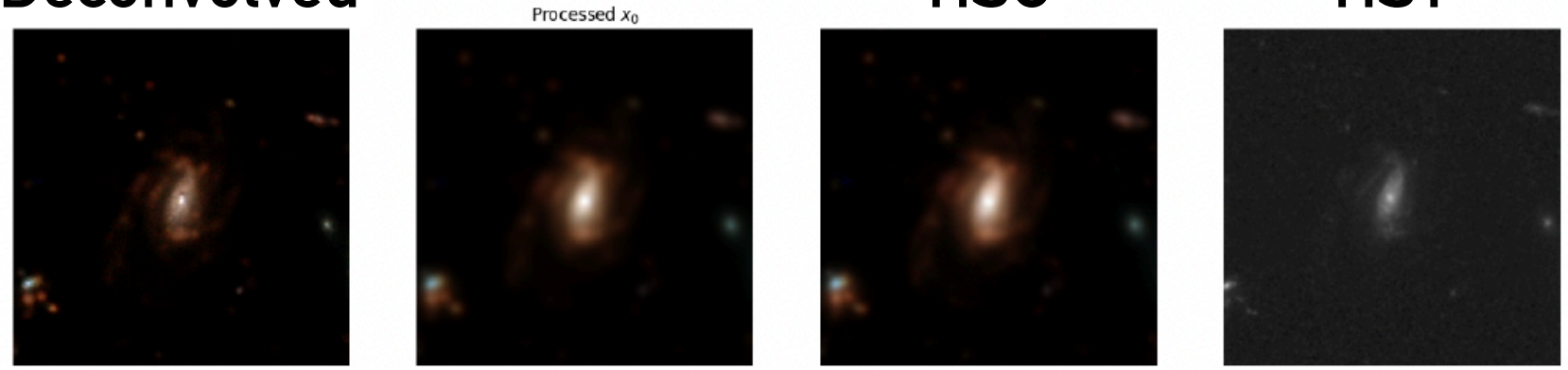


Figure 25: Object 1

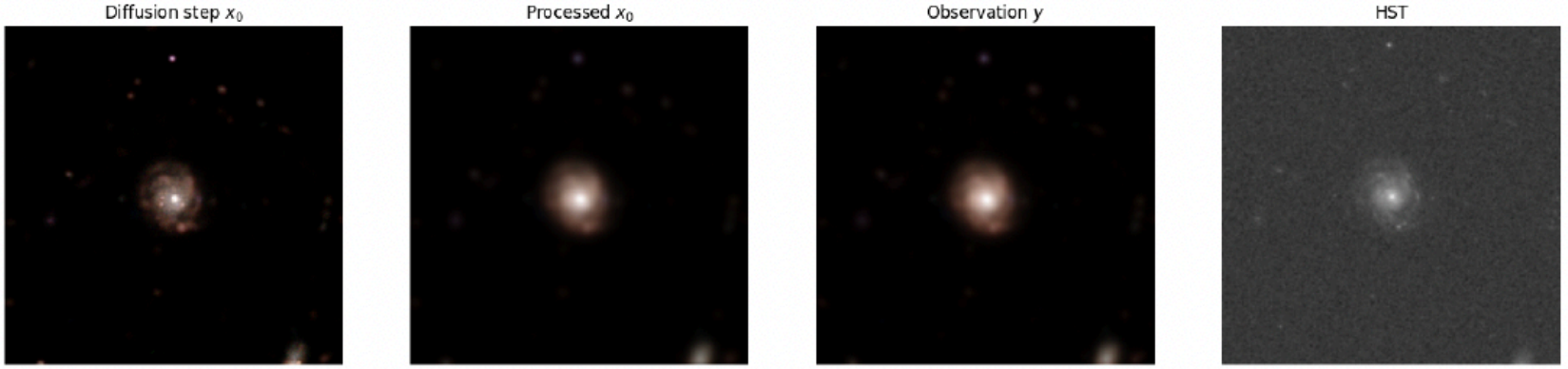
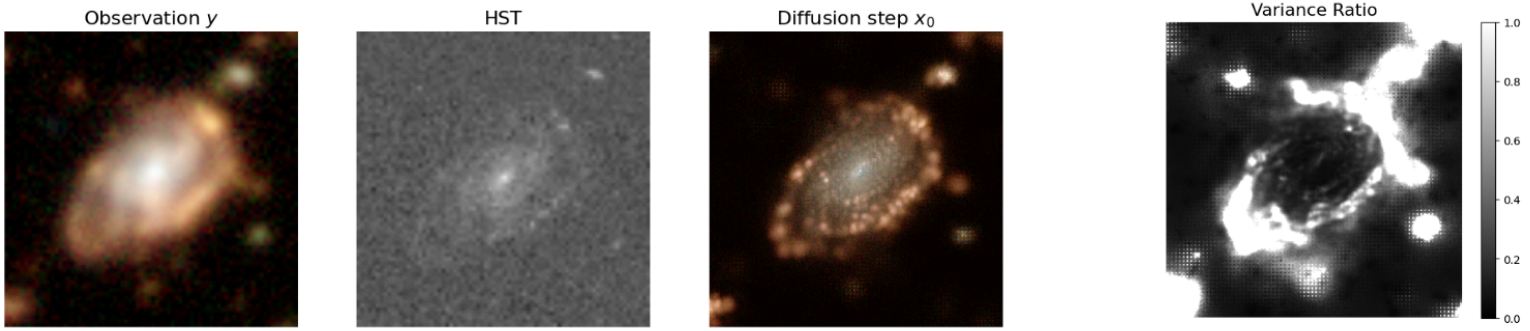
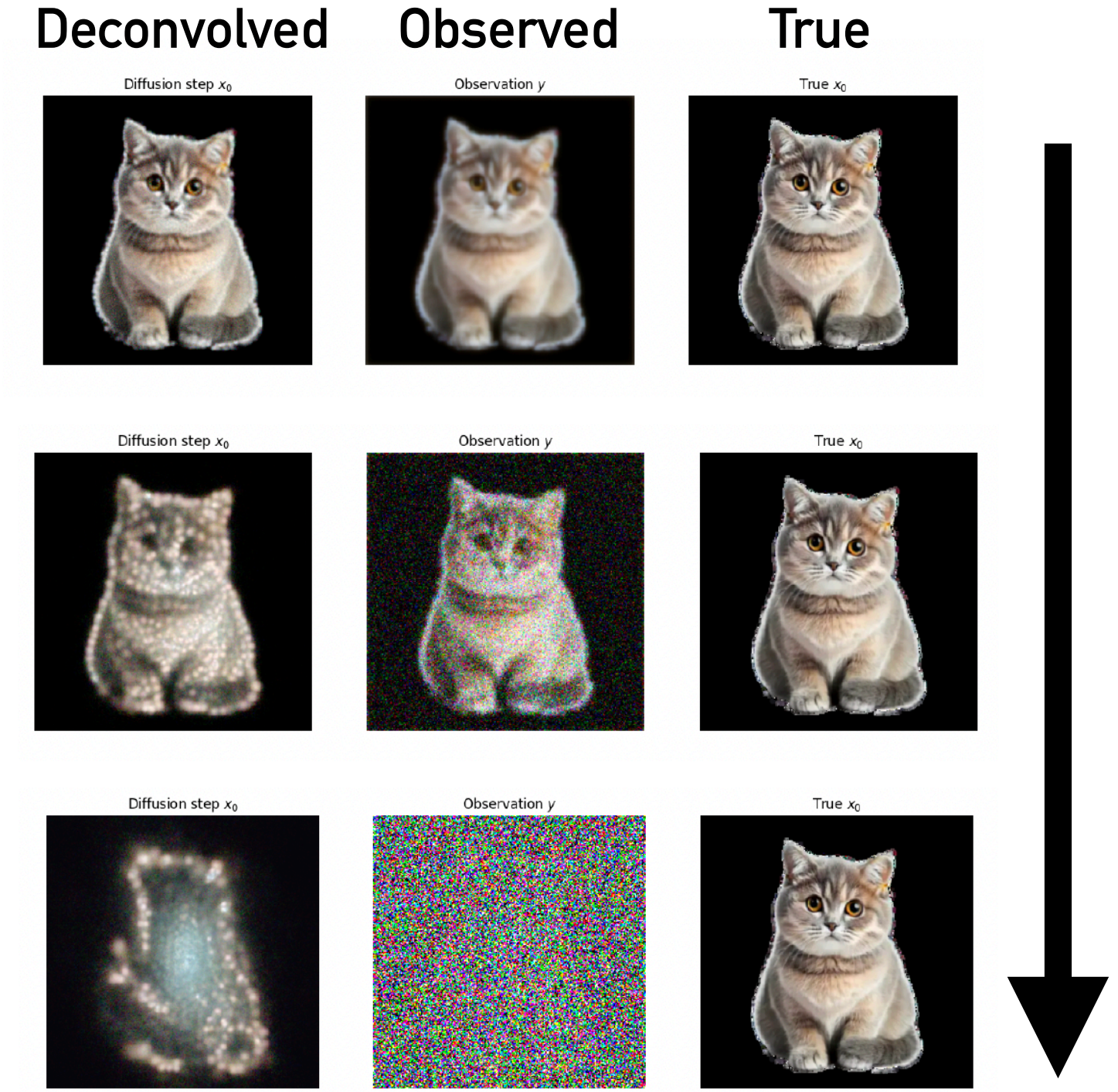


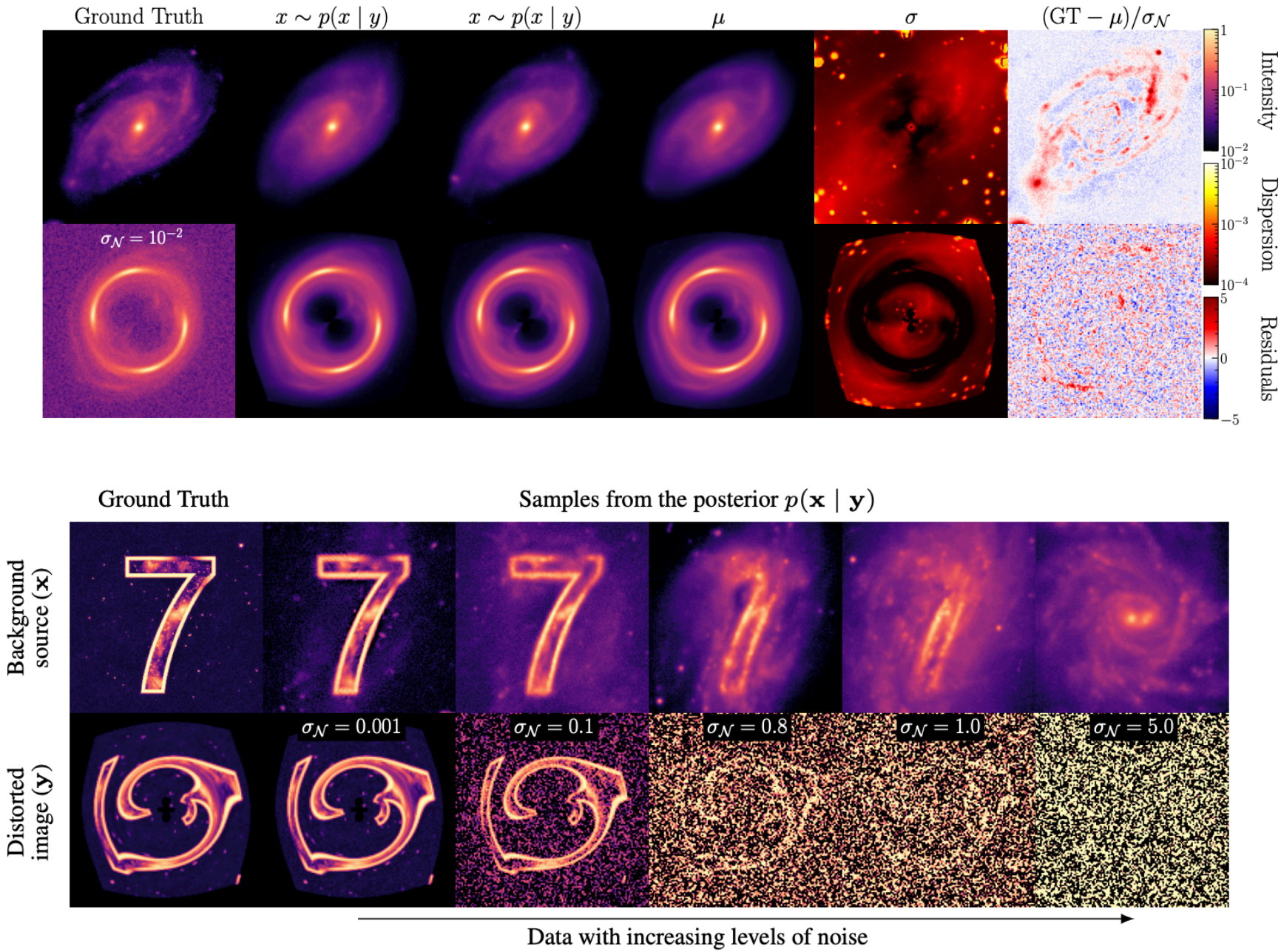
Figure 26: Object 2



Spagnoletti+24

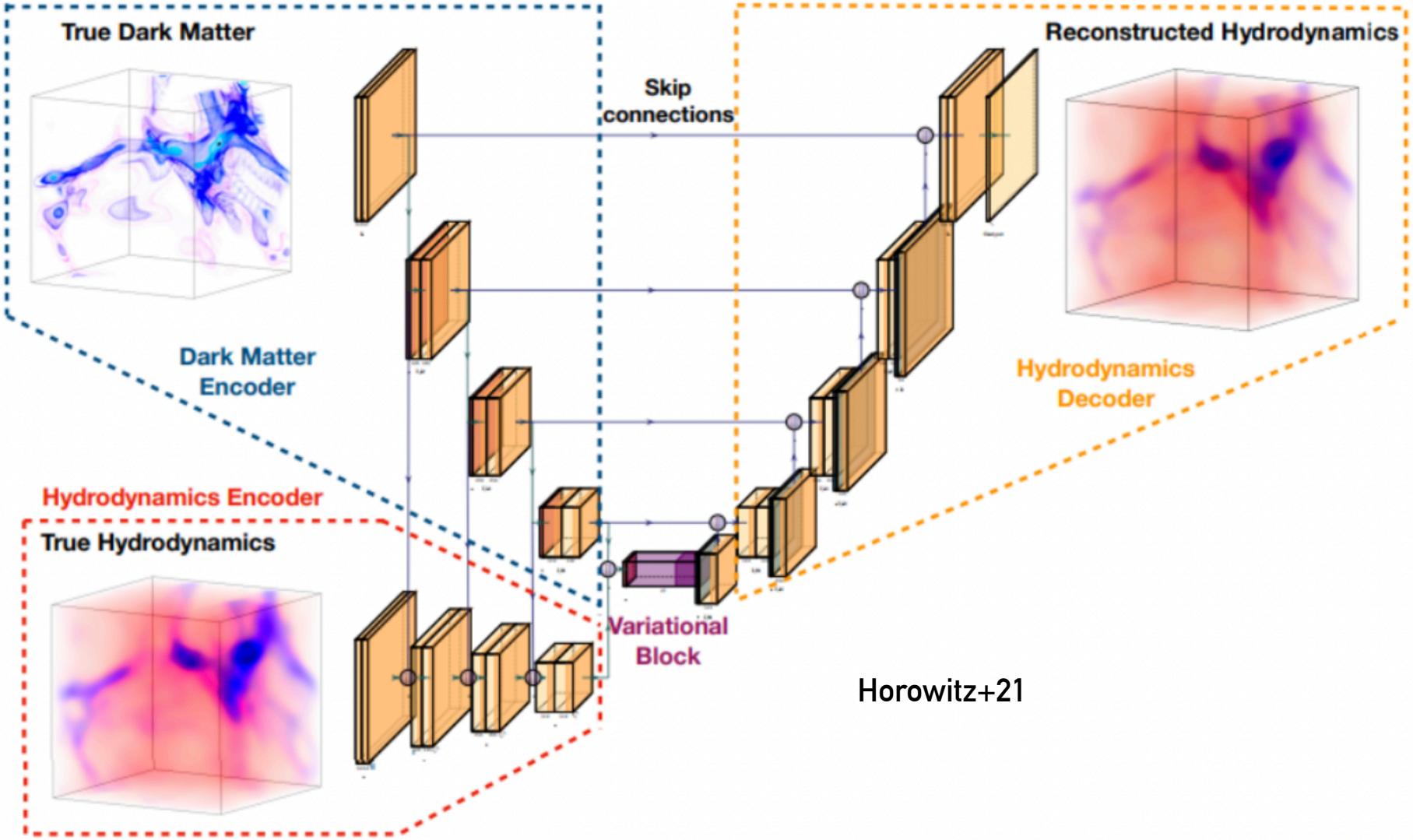


Spagnoletti+24


Ground Truth
$x \sim p(x \mid y)$
$x \sim p(x \mid y)$
μ
σ
$(\mathrm{GT} - \mu)/\sigma_{\mathcal{N}}$
$\sigma_{\mathcal{N}} = 10^{-2}$
Intensity
Dispersion
Residuals
Ground Truth
Samples from the posterior $p(\mathbf{x} \mid \mathbf{y})$
Background source ($\mathbf{x}$)
Distorted image ($\mathbf{y}$)
$\sigma_{\mathcal{N}} = 0.001$
$\sigma_{\mathcal{N}} = 0.1$
$\sigma_{\mathcal{N}} = 0.8$
$\sigma_{\mathcal{N}} = 1.0$
$\sigma_{\mathcal{N}} = 5.0$
Data with increasing levels of noise

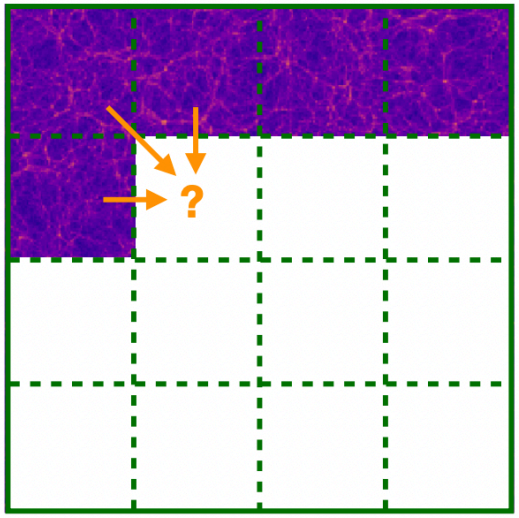
Adam+22

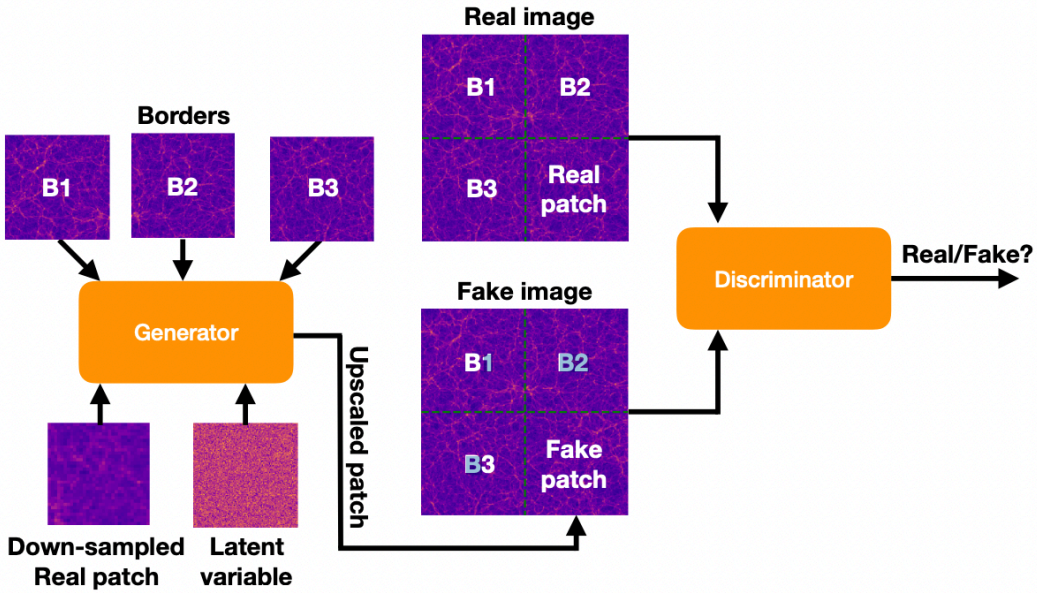
Painting Baryons

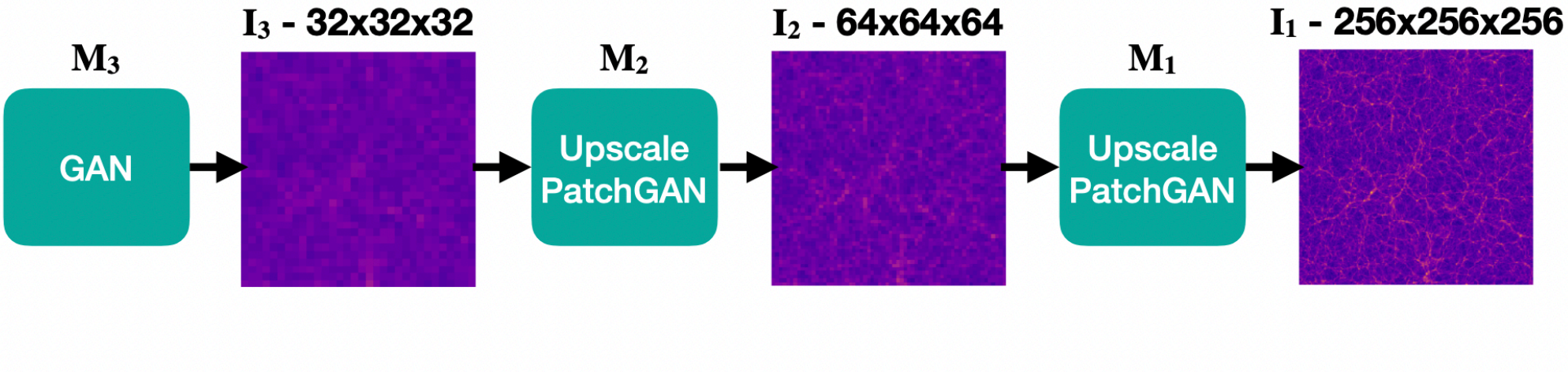


Horowitz+21

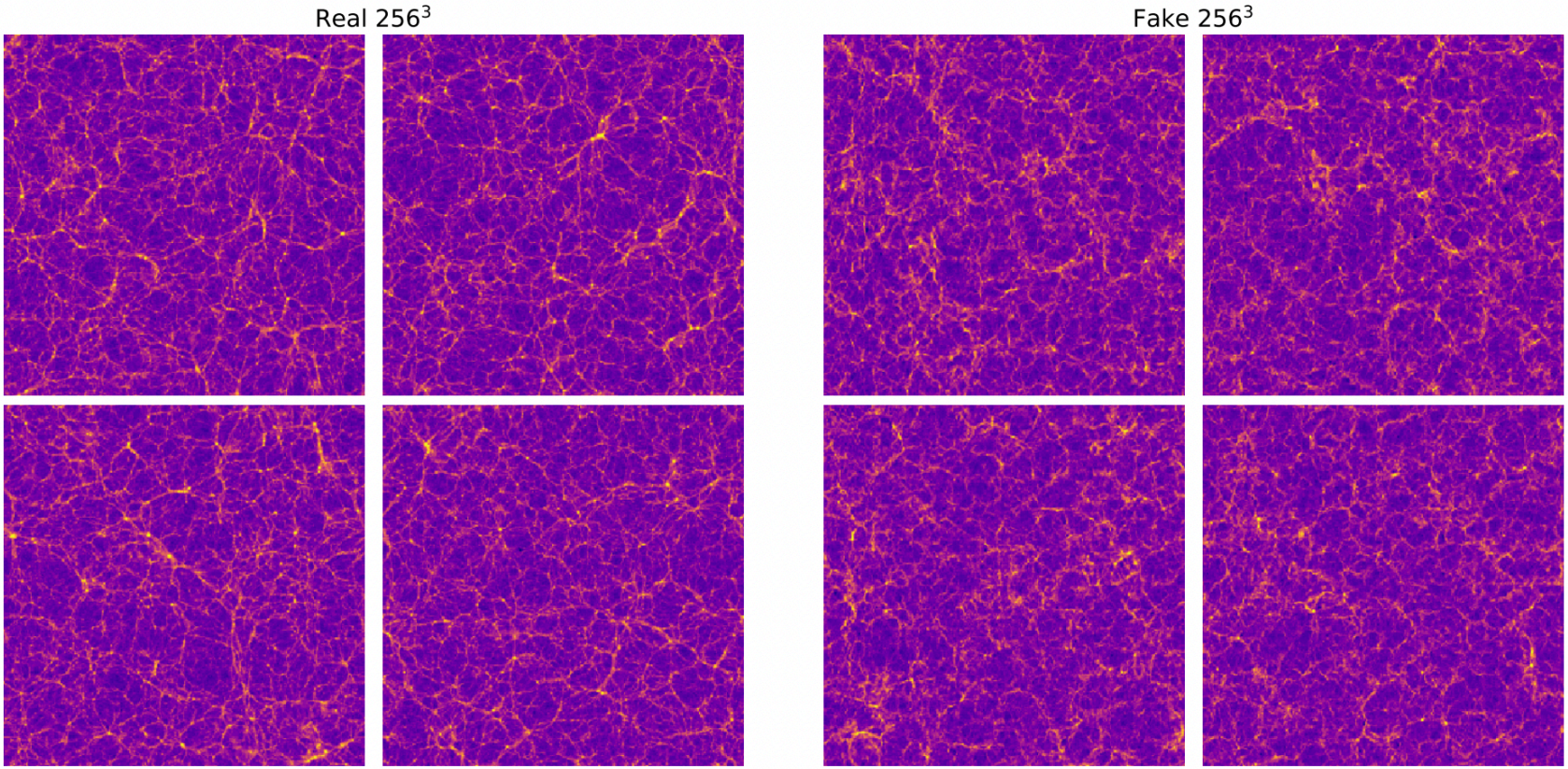
N-body emulation by Deep Generative Modelling

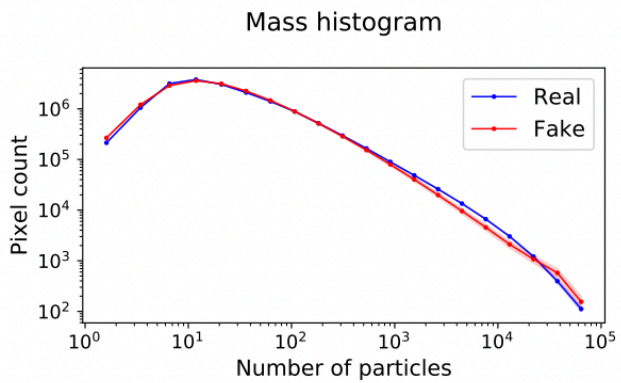


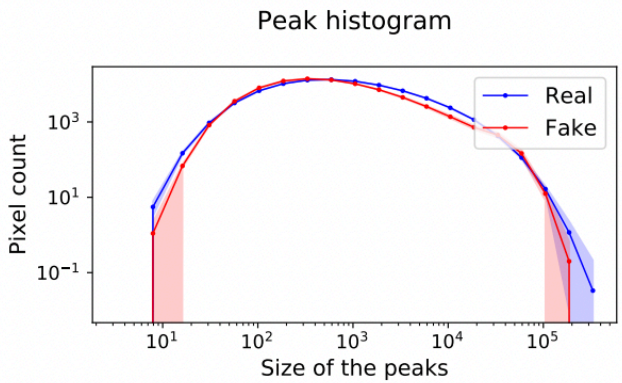


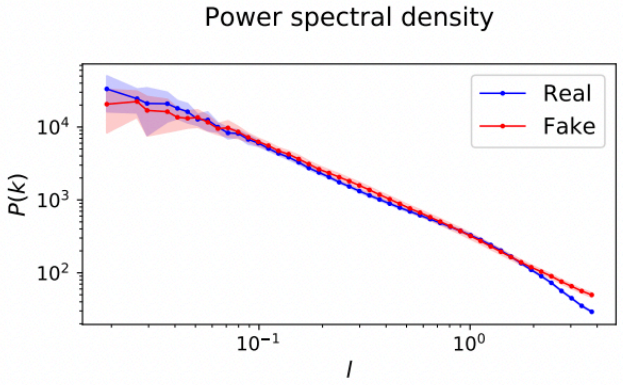


Perraudin+19



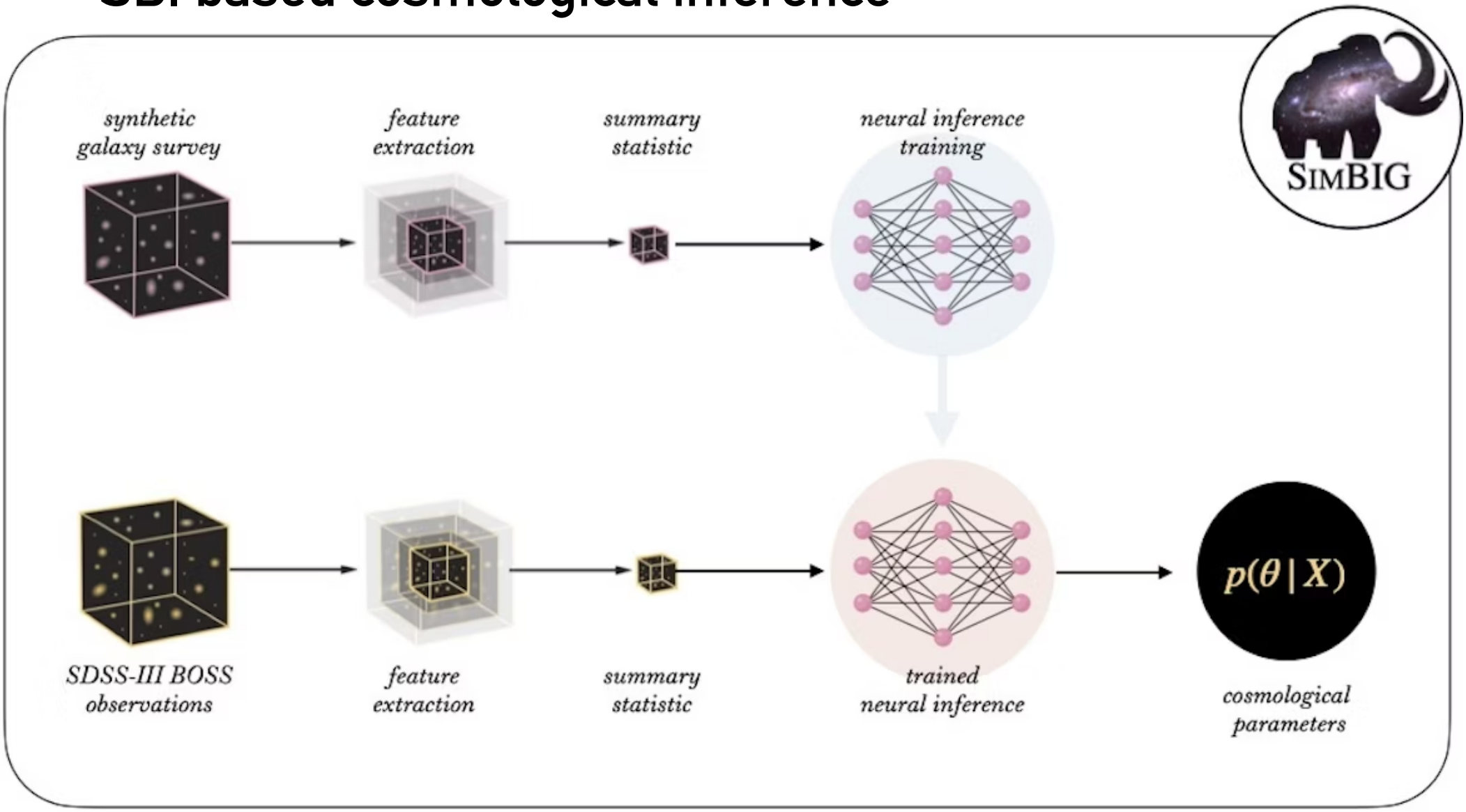






Perraudin+19

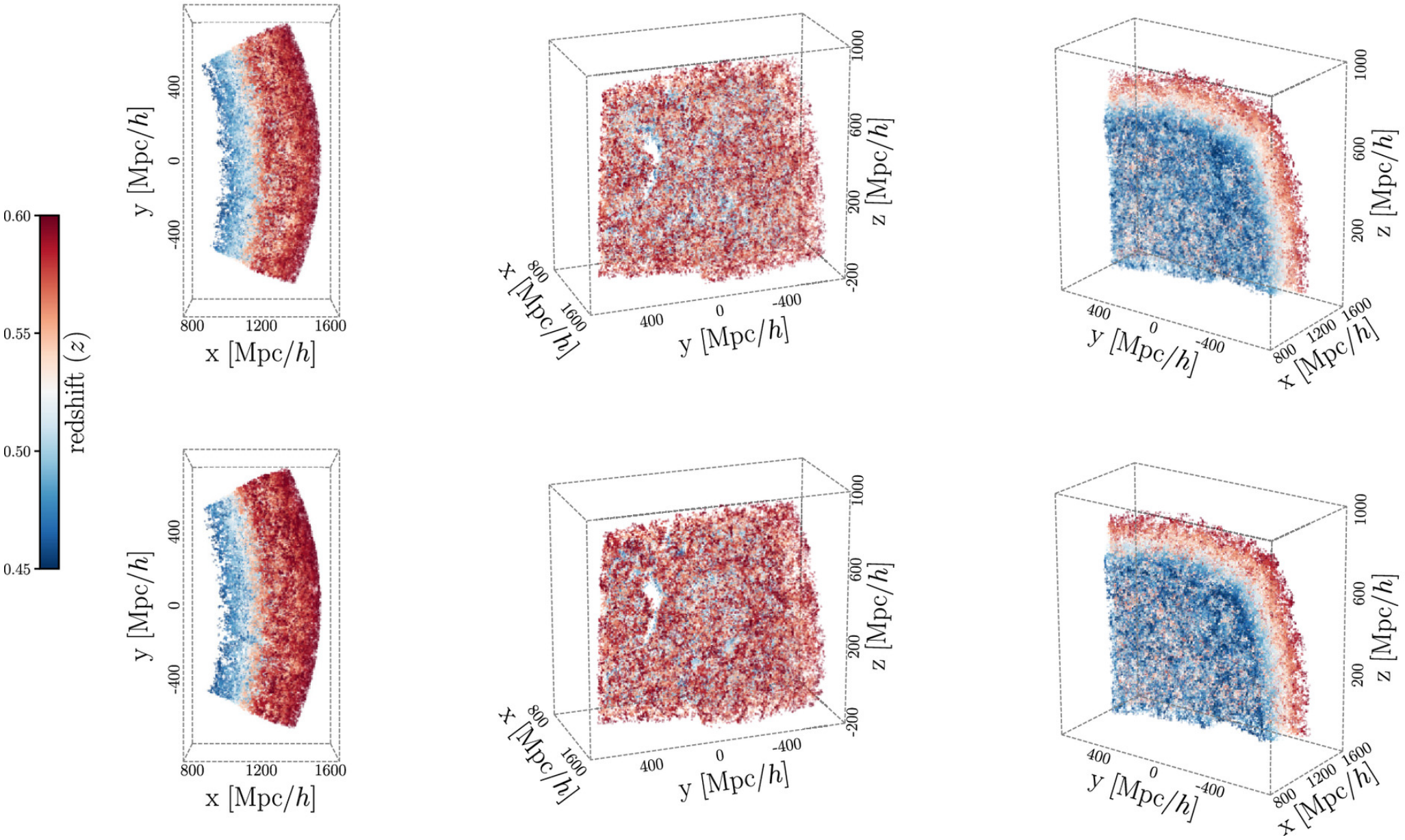
SBI based cosmological inference



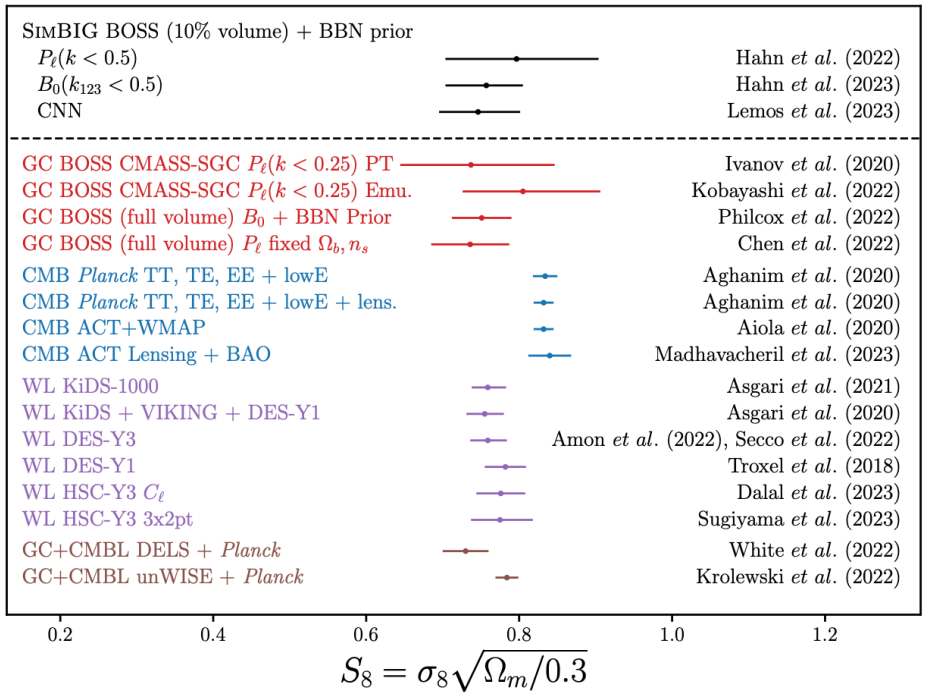
“If you can simulate, you can do inference”

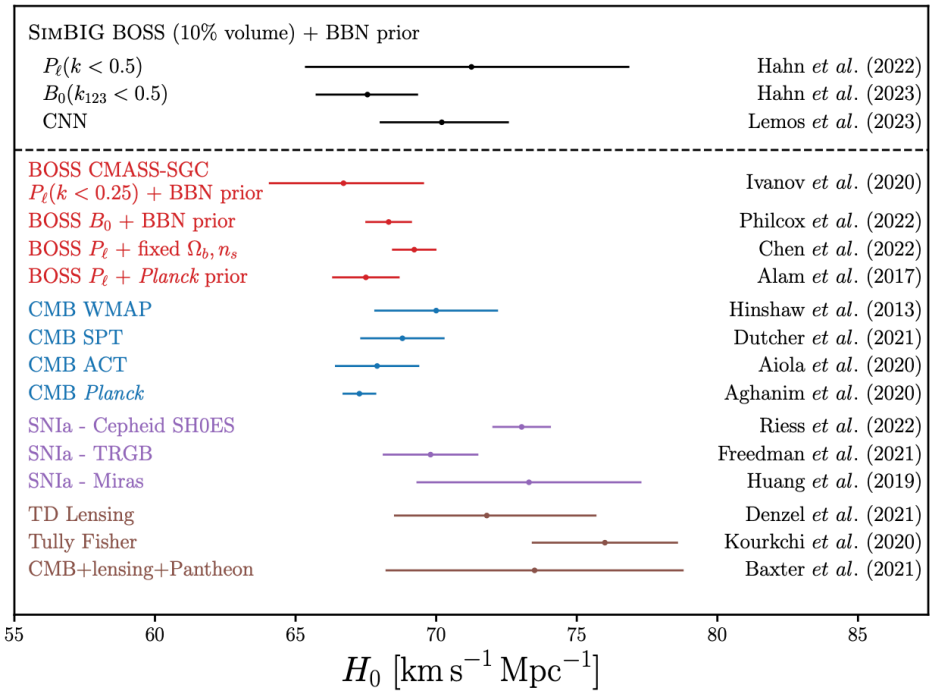
Hahn+23

Towards Field Level Cosmological Inference



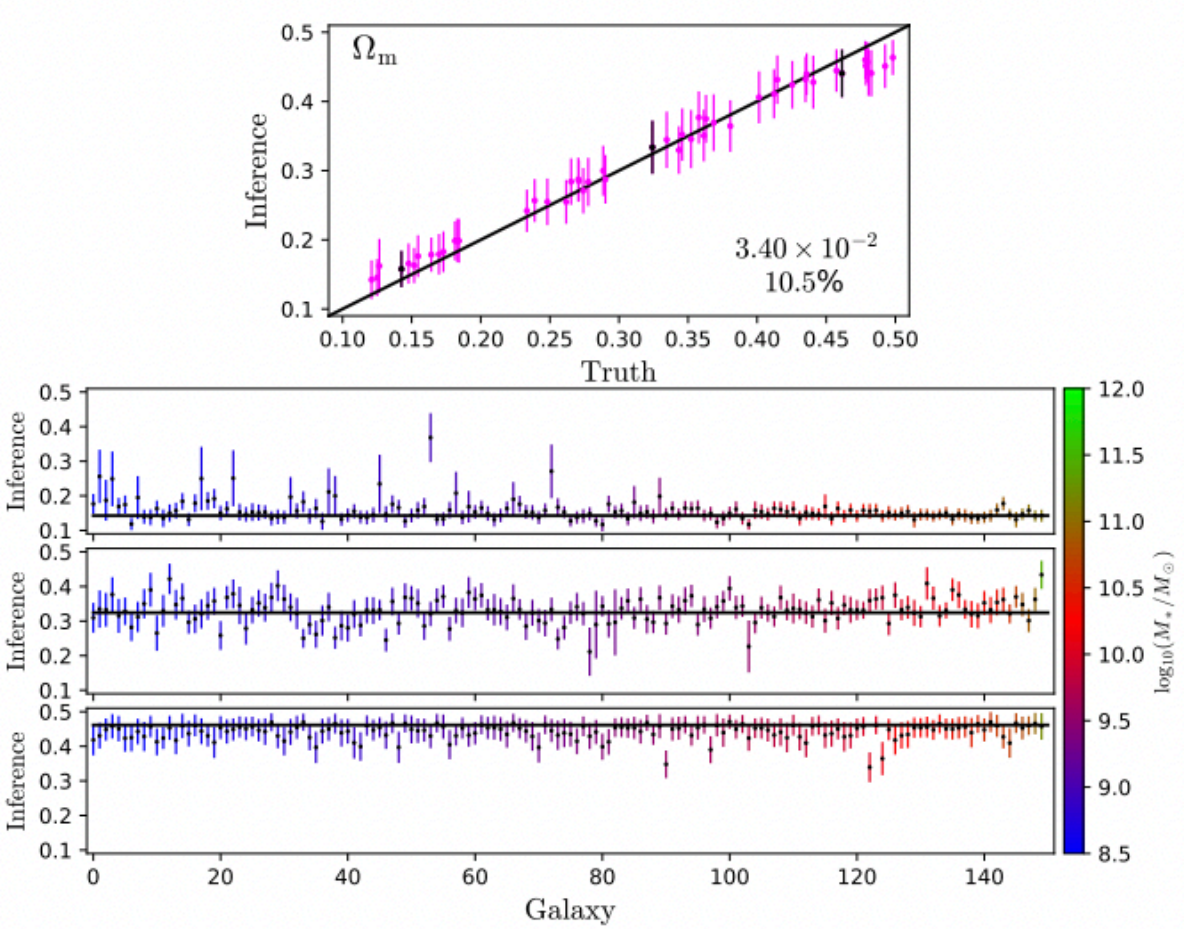
Hahn+23

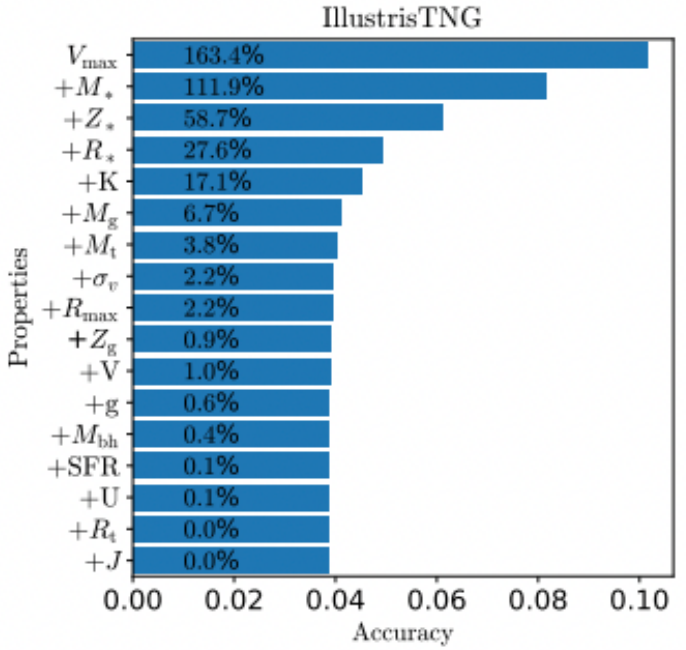

SimBIG BOSS (10% volume) + BBN prior
$P_\ell(k < 0.5)$ Hahn et al. (2022)
$B_0(k_{123} < 0.5)$ Hahn et al. (2023)
CNN Lemos et al. (2023)
GC BOSS CMASS-SGC $P_\ell(k < 0.25)$ PT Ivanov et al. (2020)
GC BOSS CMASS-SGC $P_\ell(k < 0.25)$ Emu. Kobayashi et al. (2022)
GC BOSS (full volume) B_0 + BBN Prior Philcox et al. (2022)
GC BOSS (full volume) P_ℓ fixed Ω_b, n_s Chen et al. (2022)
CMB Planck TT, TE, EE + lowE Aghanim et al. (2020)
CMB Planck TT, TE, EE + lowE + lens. Aghanim et al. (2020)
CMB ACT+WMAP Aiola et al. (2020)
CMB ACT Lensing + BAO Madhavacheril et al. (2023)
WL KiDS-1000 Asgari et al. (2021)
WL KiDS + VIKING + DES-Y1 Asgari et al. (2020)
WL DES-Y3 Amon et al. (2022), Secco et al. (2022)
WL DES-Y1 Troxel et al. (2018)
WL HSC-Y3 C_ℓ Dalal et al. (2023)
WL HSC-Y3 3x2pt Sugiyama et al. (2023)
GC+CMBL DELS + Planck White et al. (2022)
GC+CMBL unWISE + Planck Krolewski et al. (2022)
0.2
0.4
0.6
0.8
1.0
1.2
$S_8 = \sigma_8\sqrt{\Omega_m/0.3}$

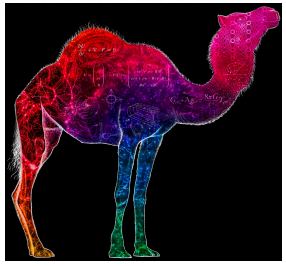

SimBIG BOSS (10% volume) + BBN prior
$P_\ell(k < 0.5)$ Hahn et al. (2022)
$B_0(k_{123} < 0.5)$ Hahn et al. (2023)
CNN Lemos et al. (2023)
BOSS CMASS-SGC $P_\ell(k < 0.25)$ + BBN prior Ivanov et al. (2020)
BOSS B_0 + BBN prior Philcox et al. (2022)
BOSS P_ℓ + fixed Ω_b, n_s Chen et al. (2022)
BOSS P_ℓ + Planck prior Alam et al. (2017)
CMB WMAP Hinshaw et al. (2013)
CMB SPT Dutcher et al. (2021)
CMB ACT Aiola et al. (2020)
CMB Planck Aghanim et al. (2020)
SNIa - Cepheid SH0ES Riess et al. (2022)
SNIa - TRGB Freedman et al. (2021)
SNIa - Miras Huang et al. (2019)
TD Lensing Denzel et al. (2021)
Tully Fisher Kourkchi et al. (2020)
CMB+lensing+Pantheon Baxter et al. (2021)
55
60
65
70
75
80
85
H_0 [km s^{-1} Mpc^{-1}]

Hahn+23

Towards Field Level Cosmological Inference



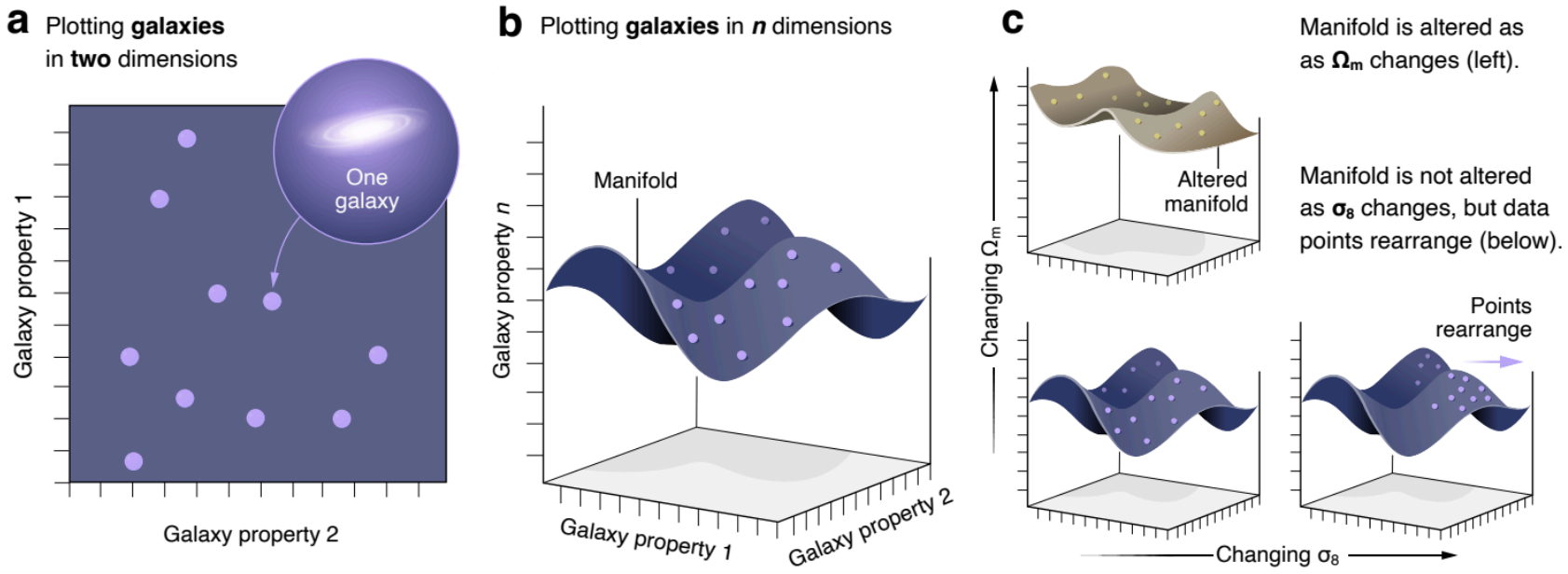




(Cosmology and Astrophysics with Machine Learning Simulations)

CAMELS

Villaescusa-Navarro+22



Lue, Genel, MHC+25