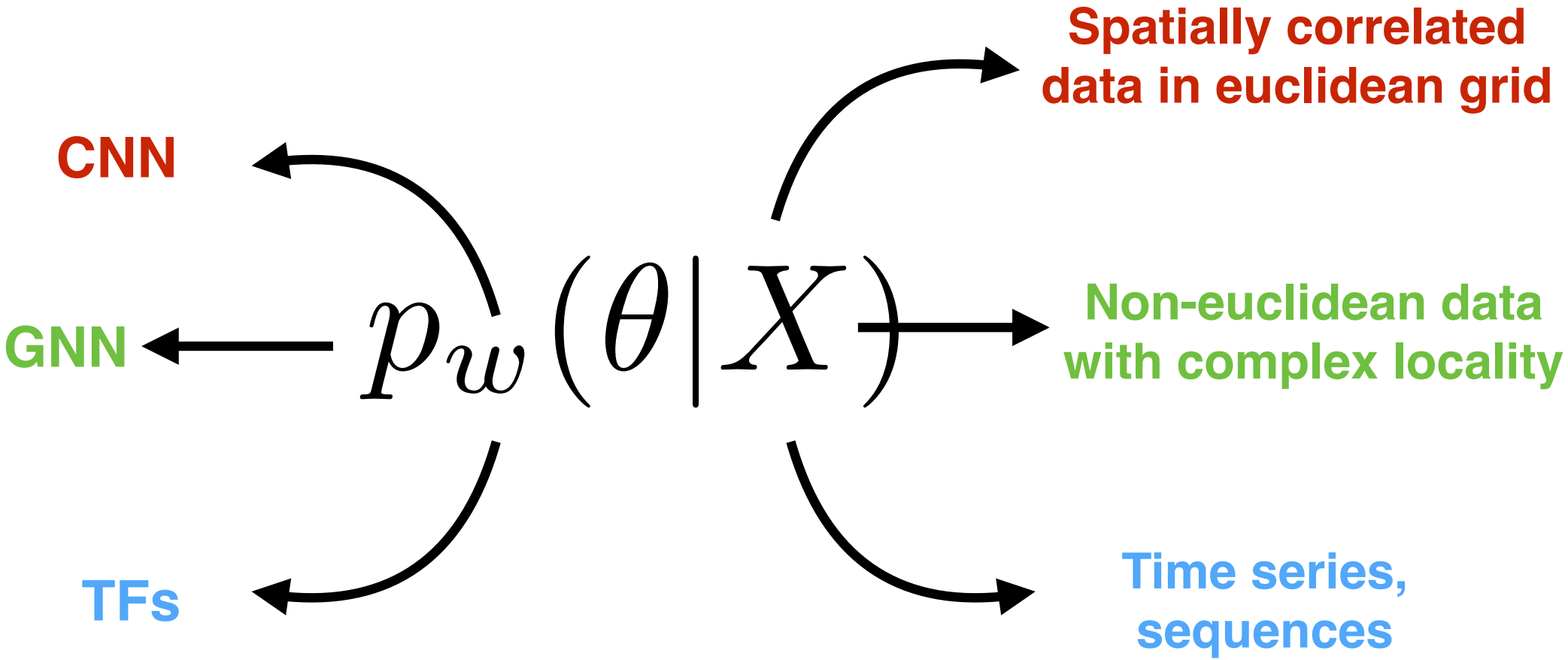
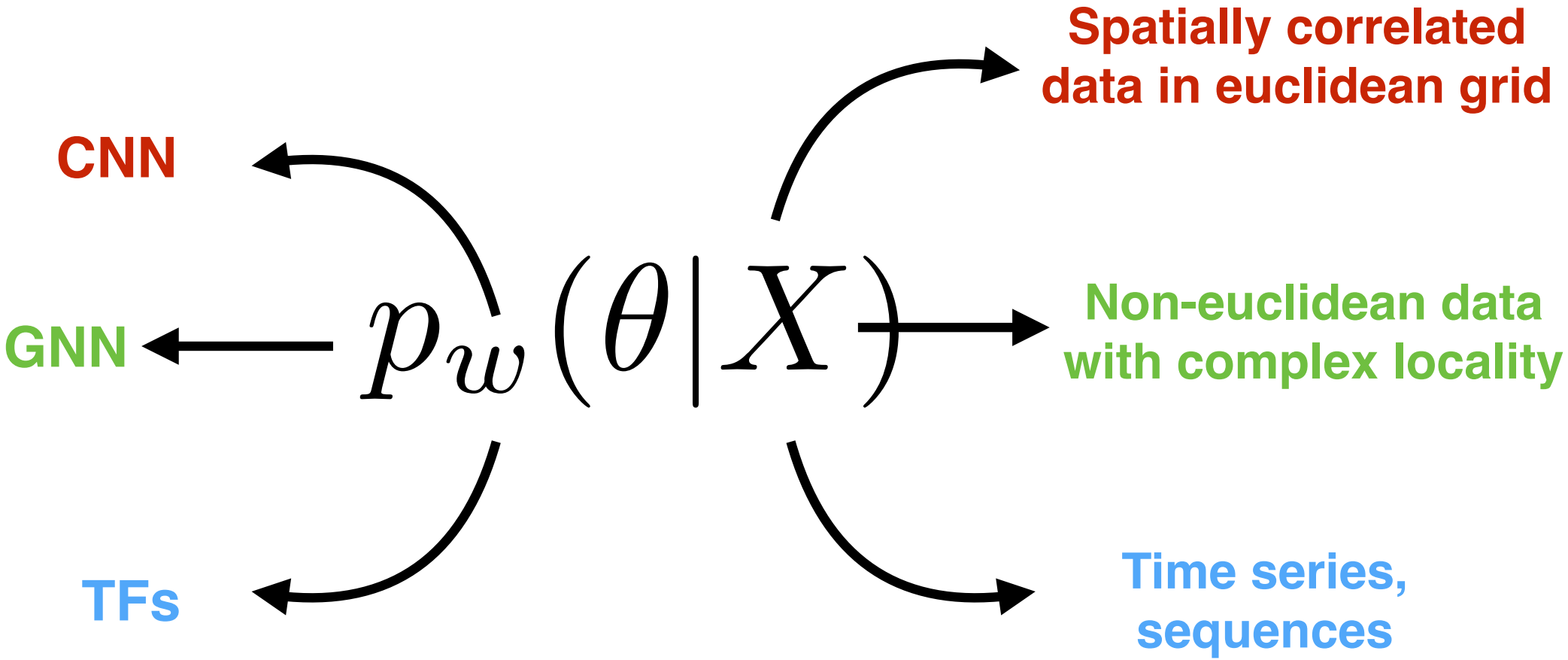
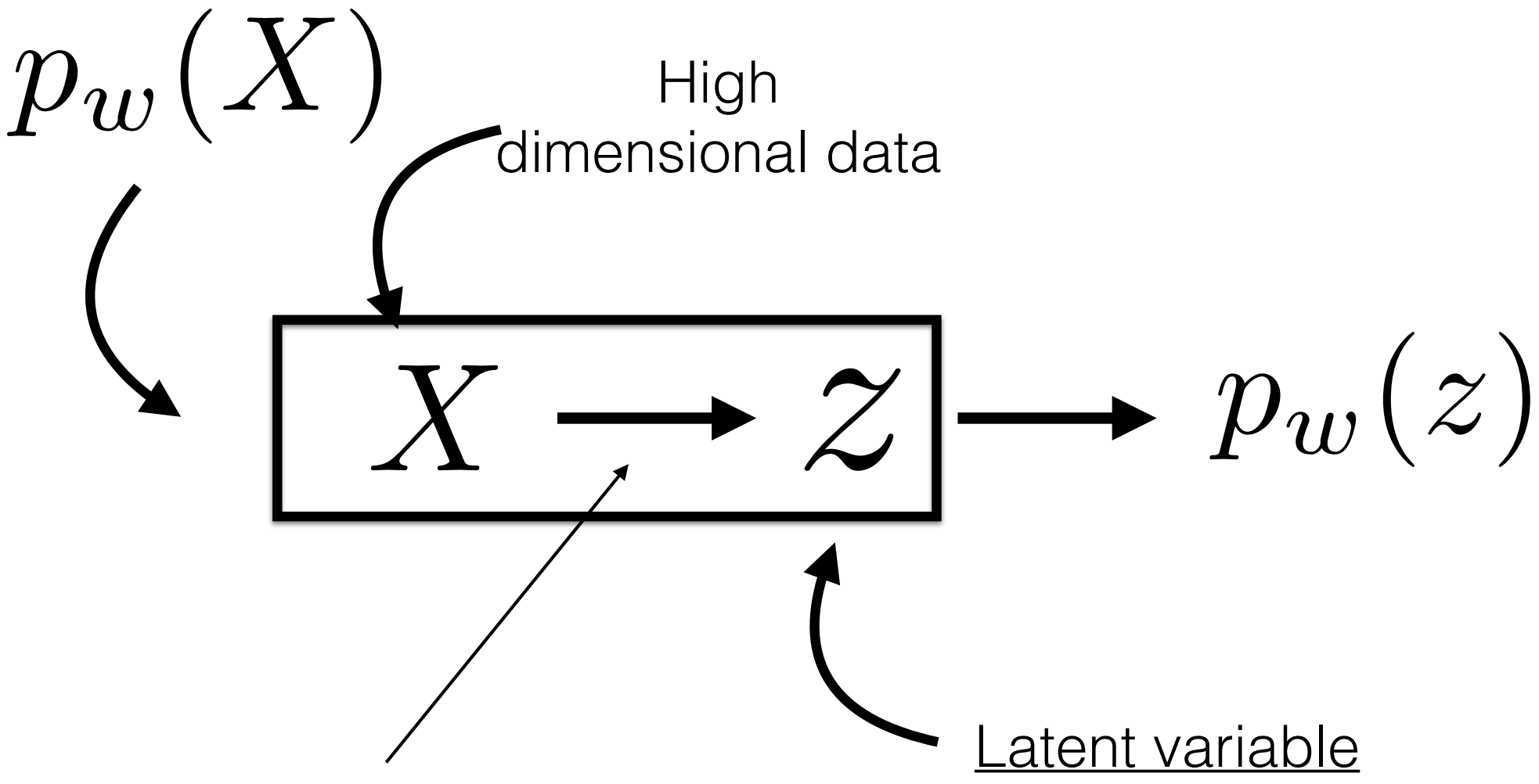
RECAP5a:


Spatially correlated
data in euclidean grid
CNN
GNN
$p_w(\theta|X)$
Non-euclidean data
with complex locality
TFs
Time series,
sequences

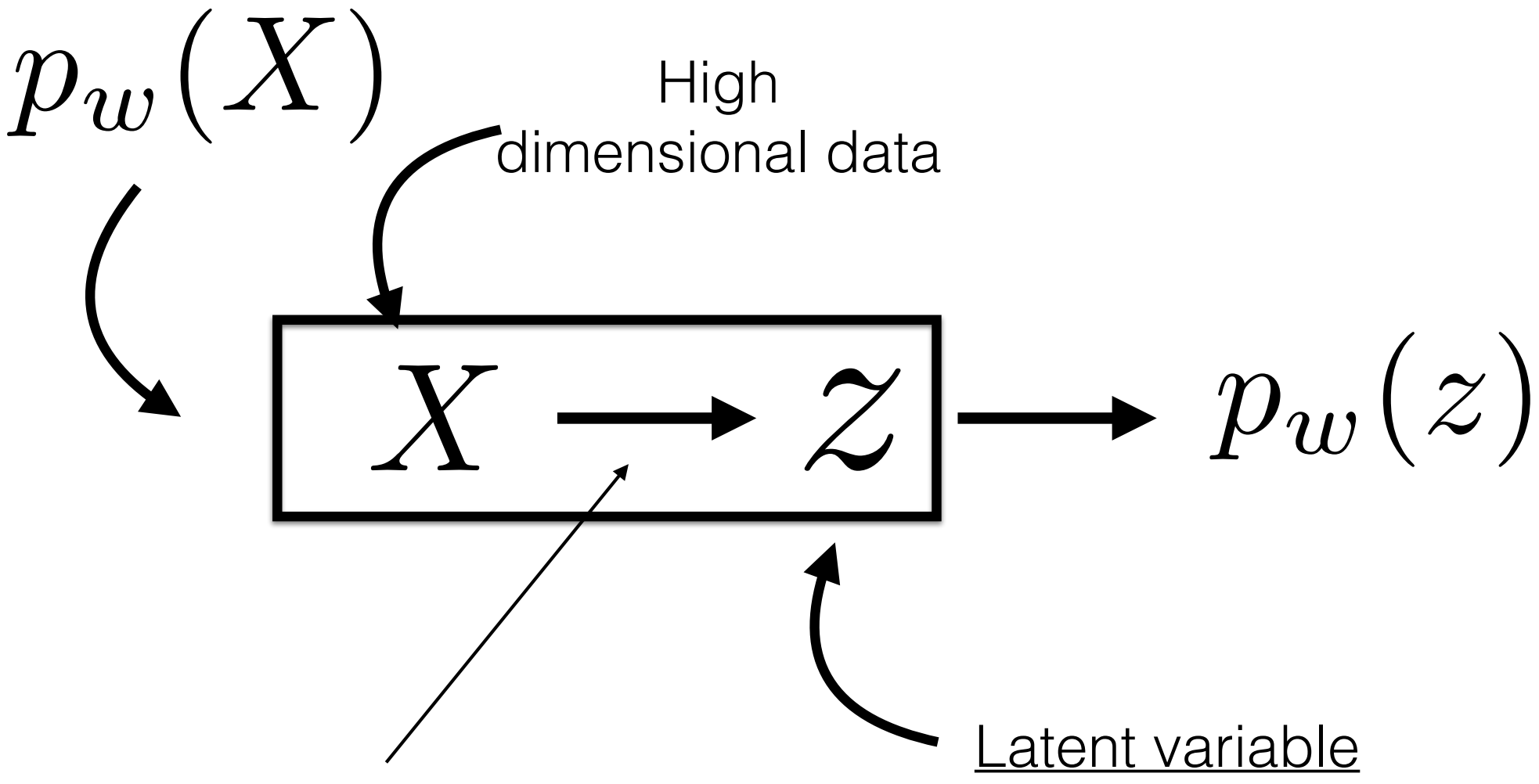
RECAP5b:


Spatially correlated
data in euclidean grid
CNN
GNN
$p_w(\theta|X)$
Non-euclidean data
with complex locality
TFs
Time series,
sequences

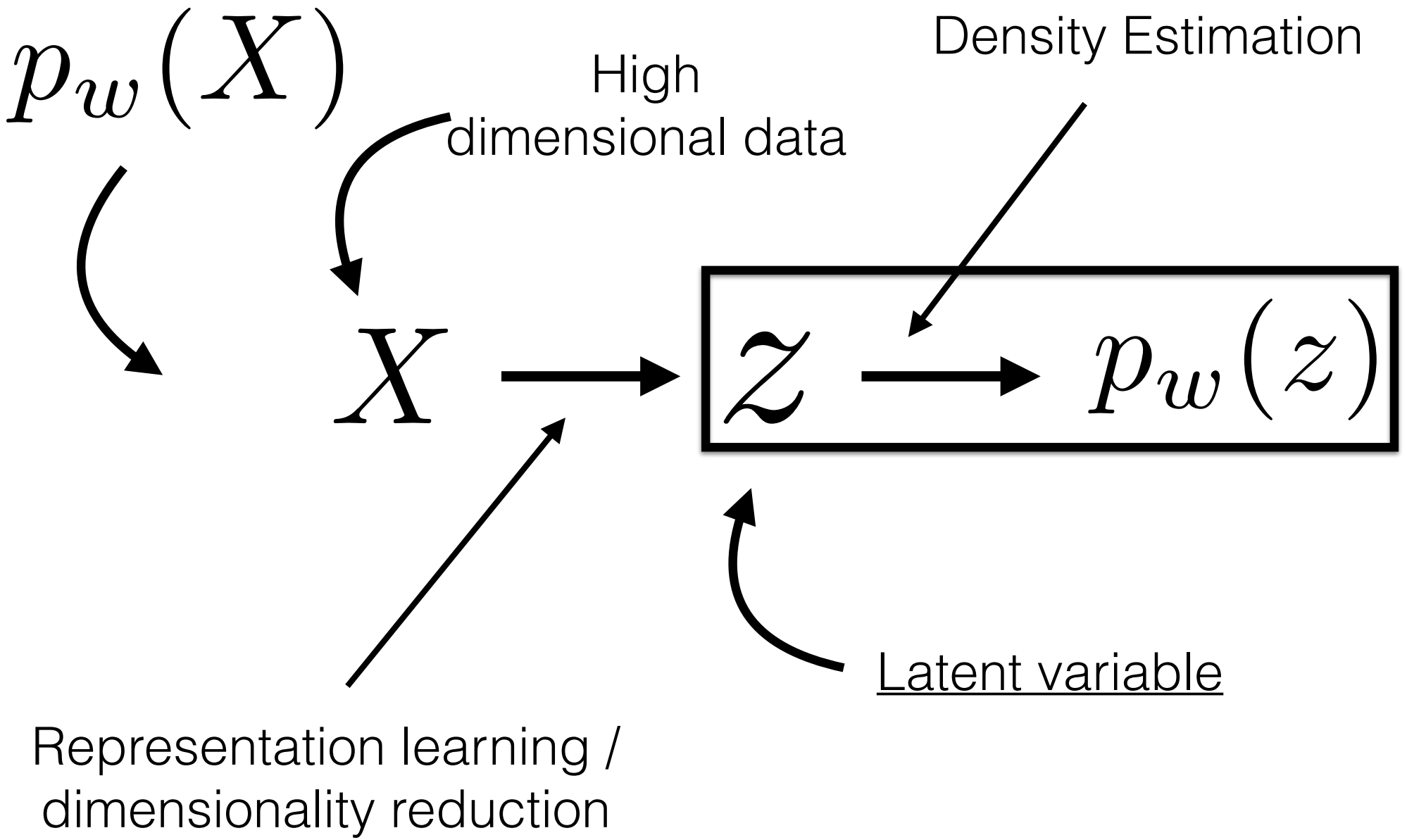
RECAP5b:


$p_w(X)$
High
dimensional data
X
z
$p_w(z)$
Latent variable
Representation learning /
dimensionality reduction

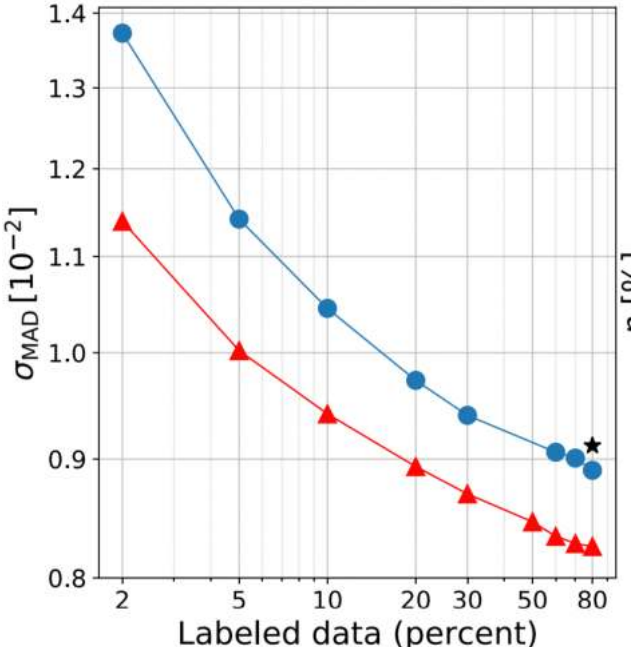
RECAP5b:

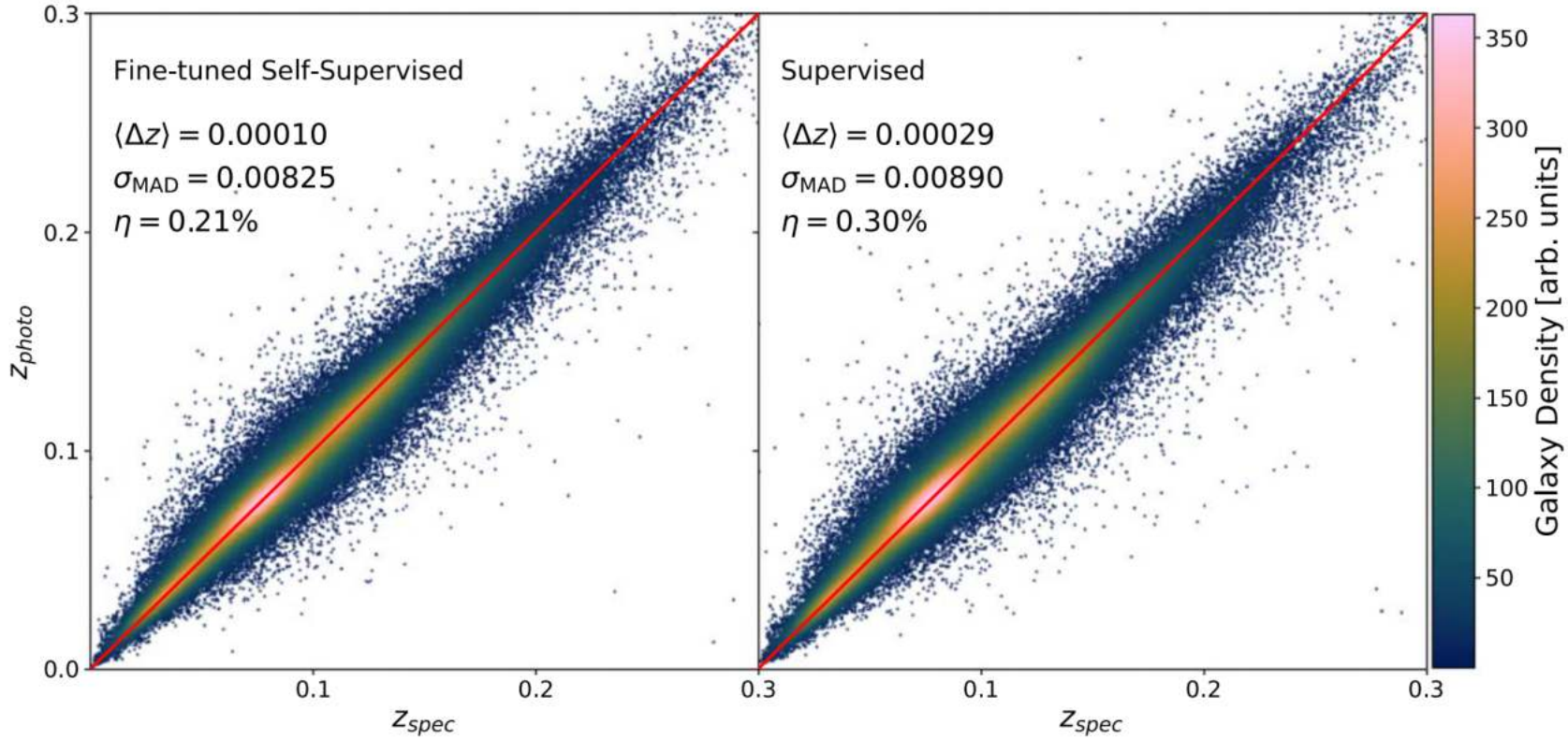

$p_w(X)$
High
dimensional data
X
z
$p_w(z)$
Latent variable
Representation learning /
dimensionality reduction

RECAP5b:

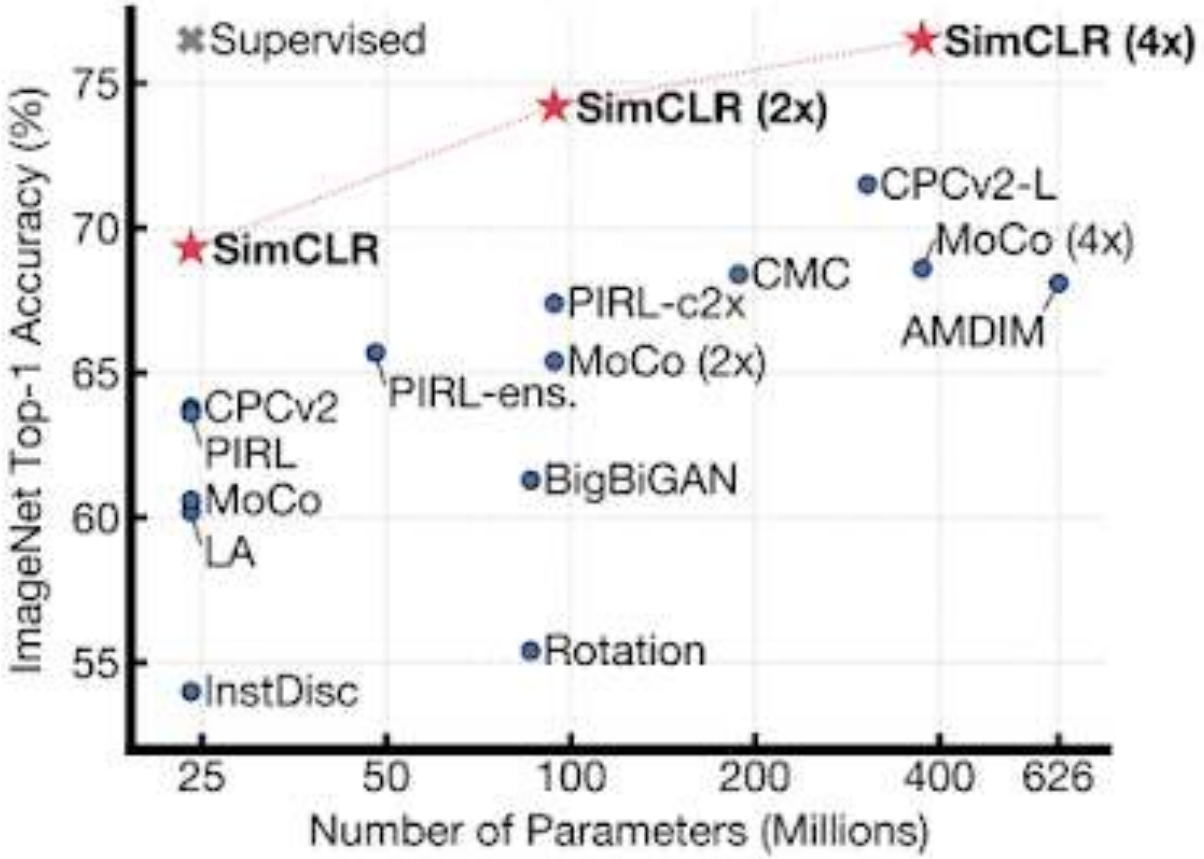

$p_w(X)$
High dimensional data
Density Estimation
X
z
$p_w(z)$
Latent variable
Representation learning / dimensionality reduction

Hayat+21


1.4
1.3
1.2
1.1
1.0
0.9
0.8
$\sigma_{\rm MAD}\,[10^{-2}]$
2
5
10
20
30
50
80
Labeled data (percent)

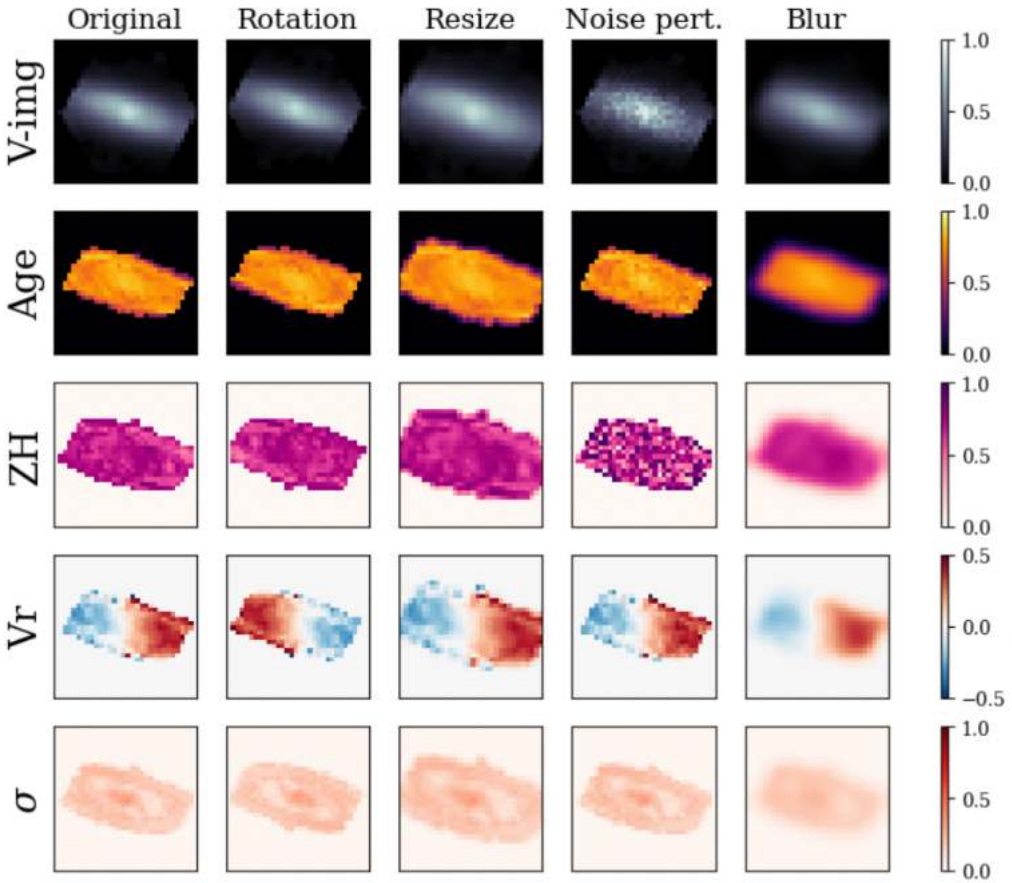

Fine-tuned Self-Supervised
$\langle \Delta z \rangle = 0.00010$
$\sigma_{\rm MAD} = 0.00825$
$\eta = 0.21\%$
Supervised
$\langle \Delta z \rangle = 0.00029$
$\sigma_{\rm MAD} = 0.00890$
$\eta = 0.30\%$
z_{photo}
z_{spec}
z_{spec}
Galaxy Density [arb. units]
0.0
0.1
0.2
0.3
50
100
150
200
250
300
350

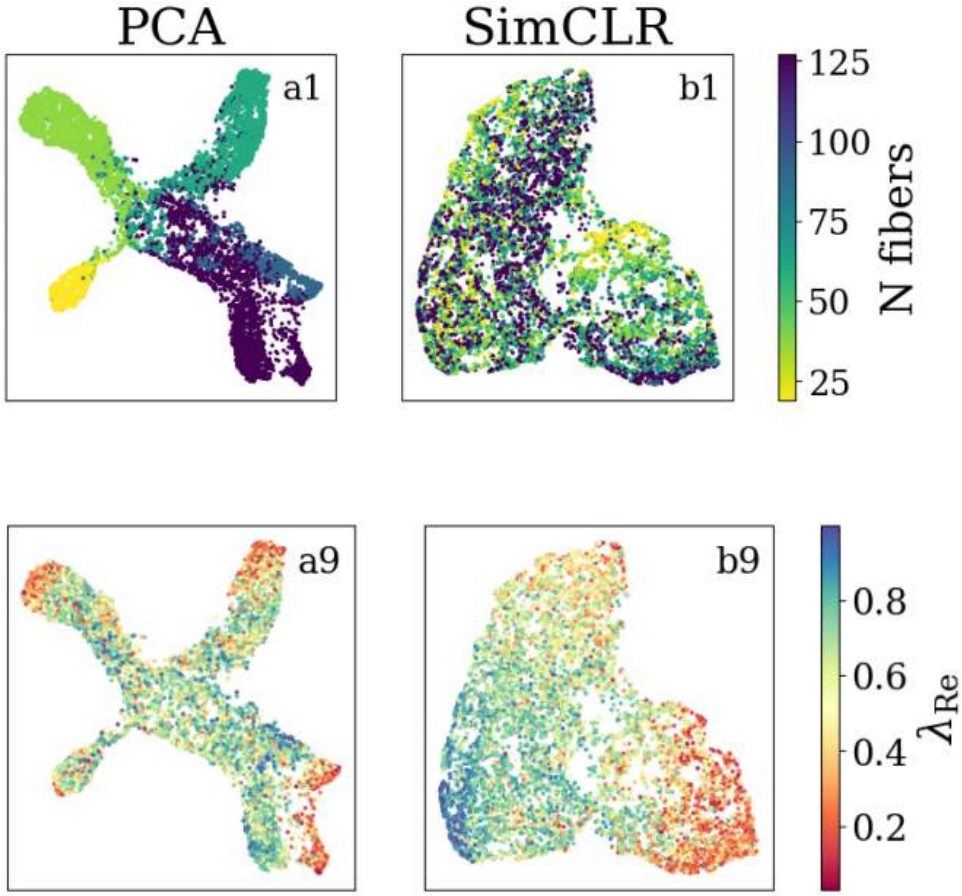
SELF-SUPERVISED LEARNING REACHES COMPARABLE ACCURACY TO FULLY SUPERVISED APPROACHES...

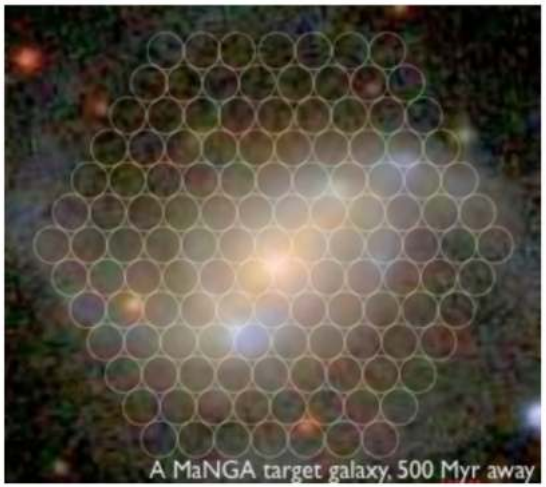


Chen+20

2. Sampling







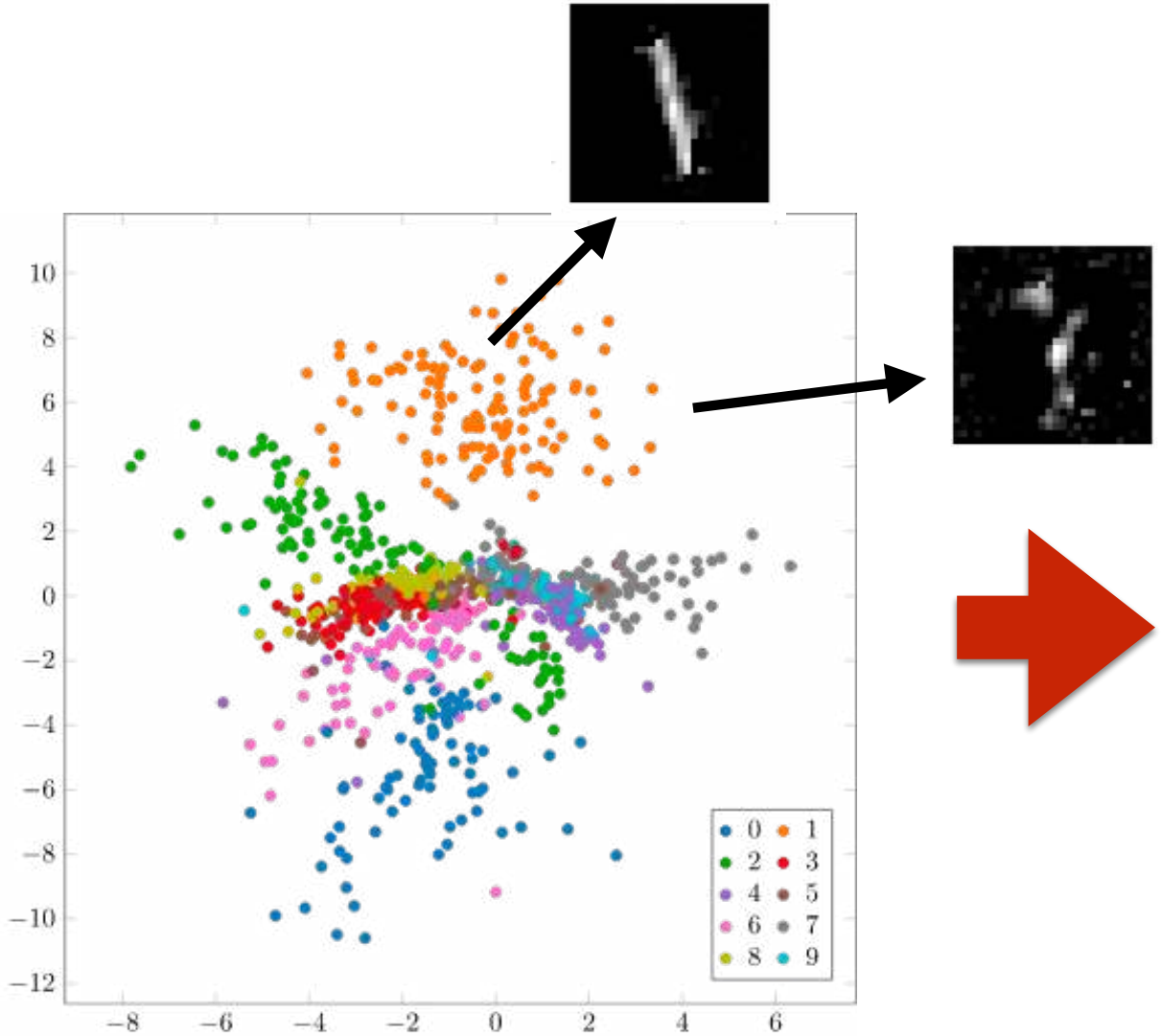
Contrastive learning representation of Manga galaxies

Sarmiento+21

HOW MIGHT YOU SOLVE THESE RELATED PROBLEMS ONCE YOU HAVE THE LATENT SPACE COORDINATES FOR YOUR TRAINING SAMPLE?

GENERATE A RANDOM SAMPLE DRAWN FROM THE INPUT DISTRIBUTION ("**SAMPLING**")

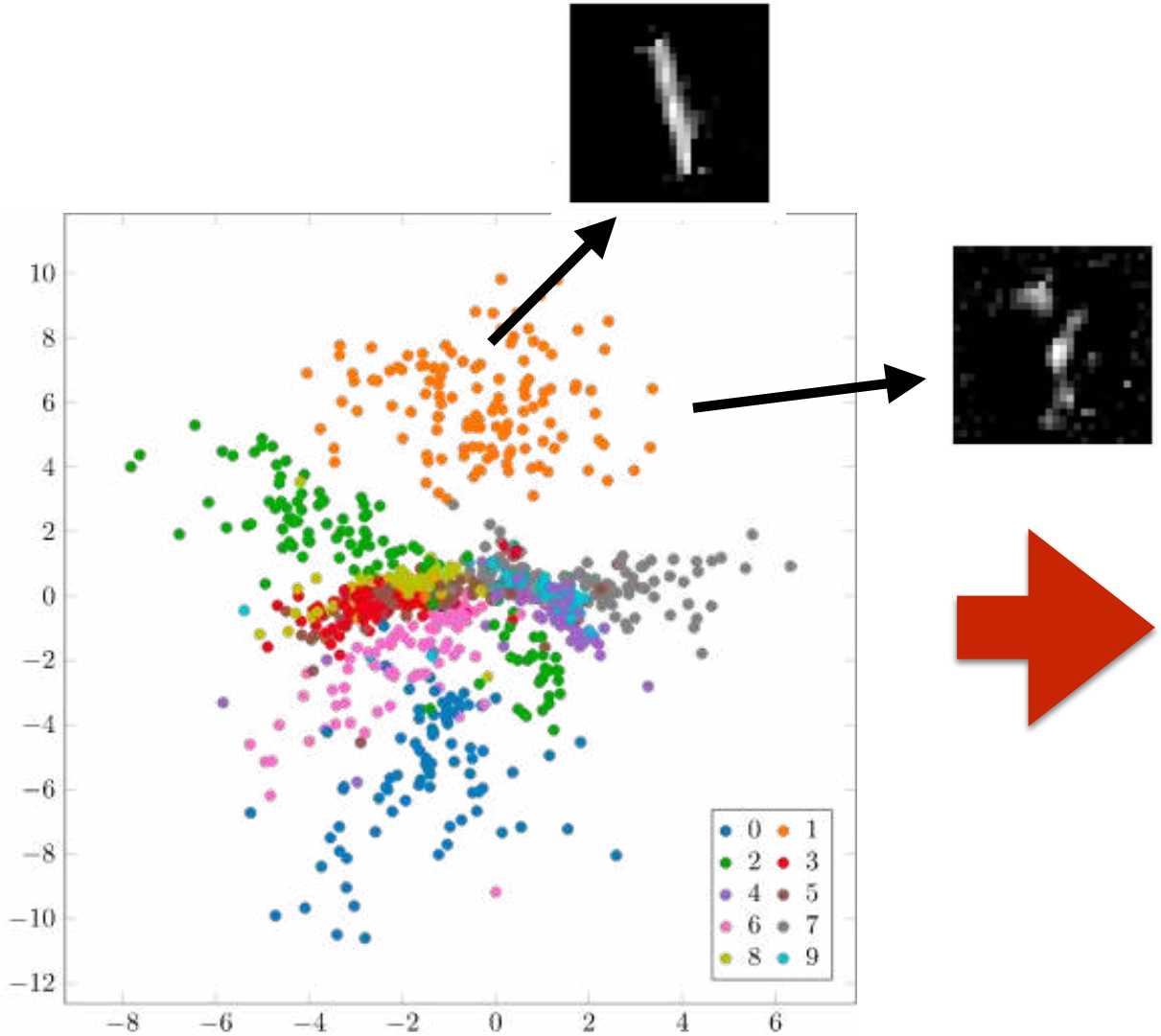
ESTIMATE THE PROBABILITY DENSITY OF AN ARBITRARY INPUT, RELATIVE TO THE INPUT DISTRIBUTION ("**DENSITY ESTIMATION**")



z space (latent space)

HOW CAN I ESTIMATE P(X)?

(P(z|x))



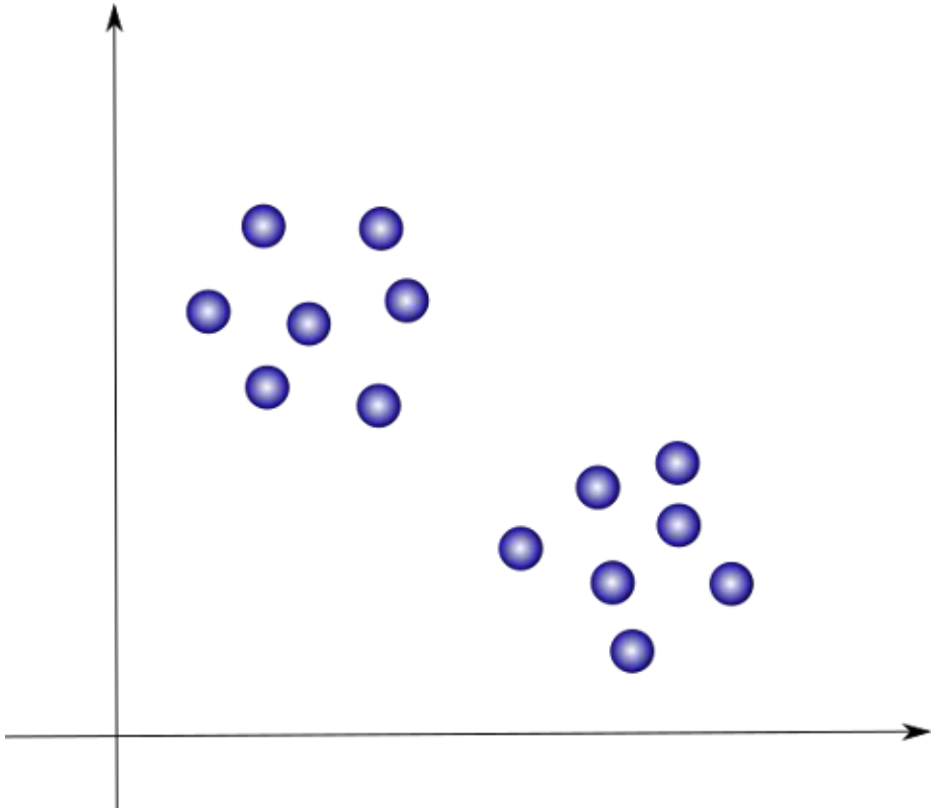
z space (latent space)

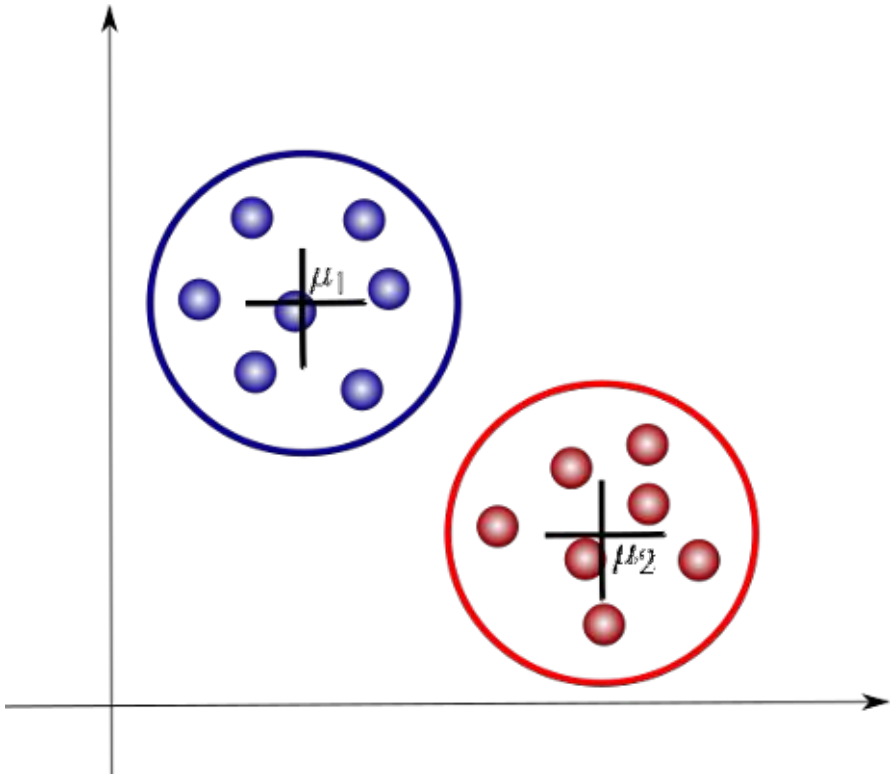
HOW CAN I ESTIMATE P(X)?

(P(z|x))

When you do not know, assume it is close to Gaussian…

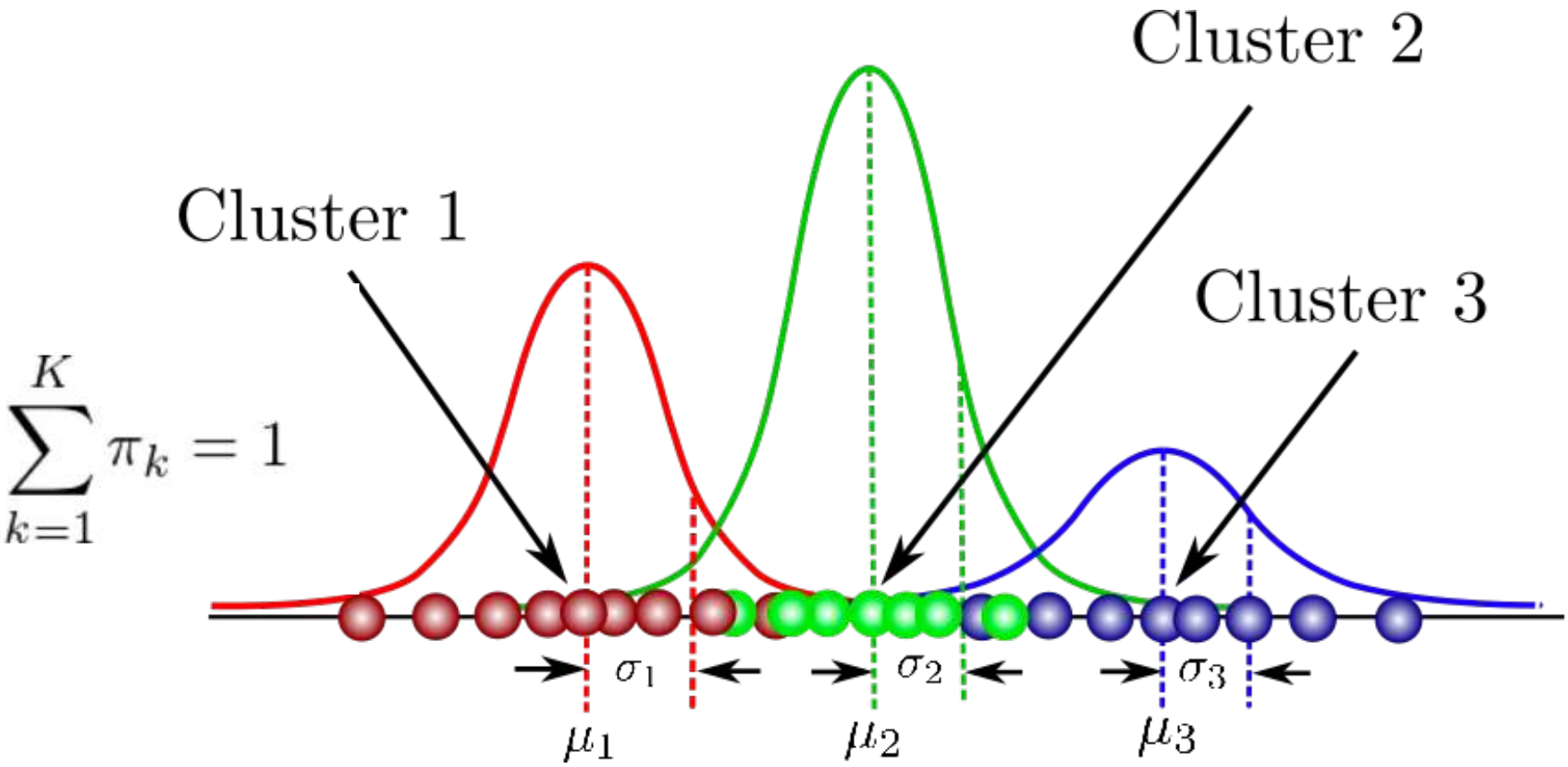
GAUSSIAN MIXTURE MODELS (GMMs) ARE DENSITY ESTIMATOR METHODS THAT FIT MULTIPLE GAUSSIANS TO THE REPRESENTATION





https://towardsdatascience.com/gaussian-mixture-models-explained-6986aaf5a95

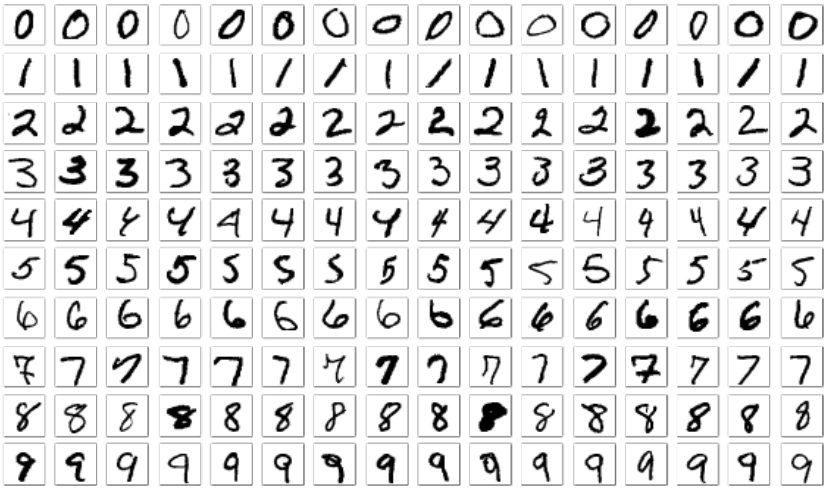
GAUSSIAN MIXTURE MODELS (GMMs) ARE DENSITY ESTIMATOR METHODS THAT FIT MULTIPLE GAUSSIANS TO THE REPRESENTATION

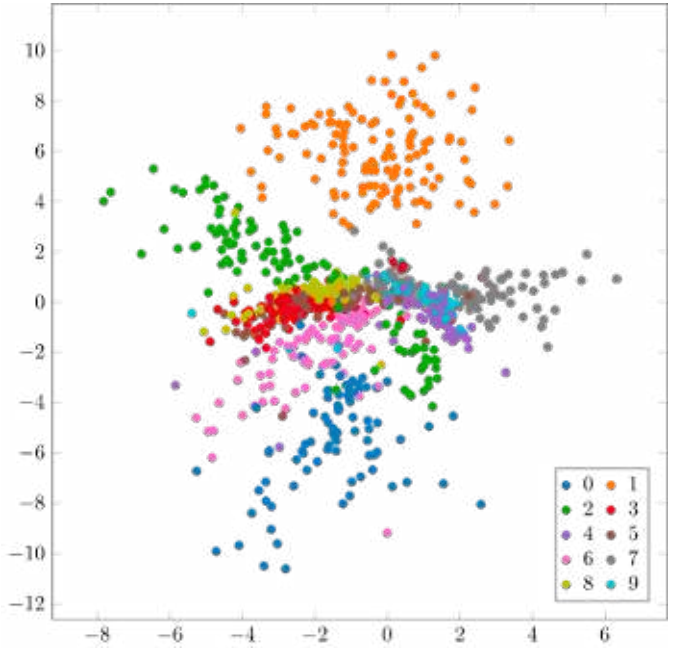


means, sigmas and scale factors of each gaussian are free parameters

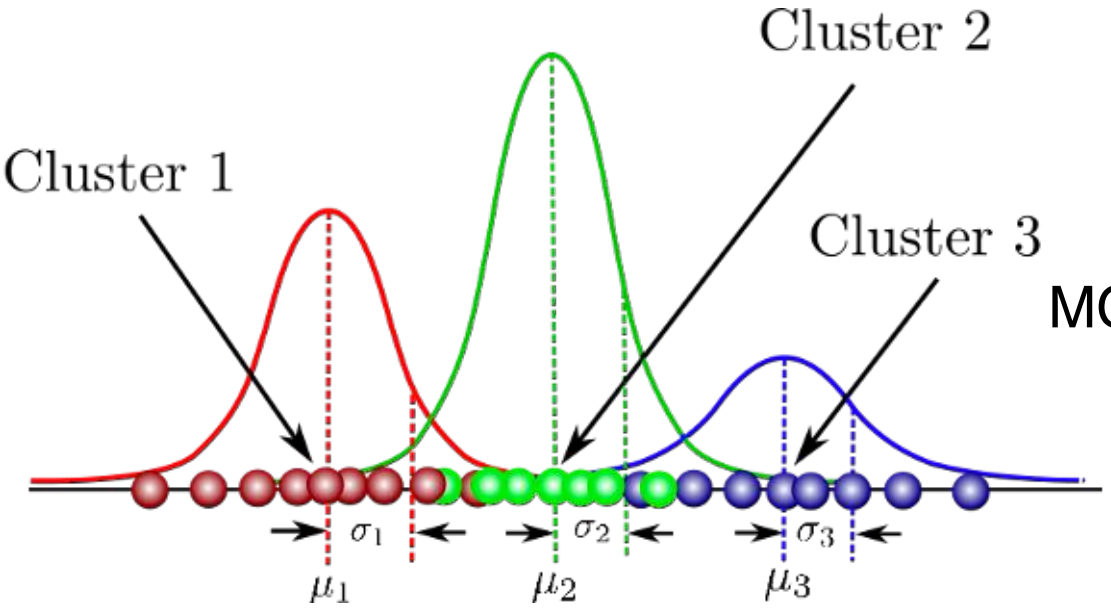
https://towardsdatascience.com/gaussian-mixture-models-explained-6986aaf5a95

DATA





REPRESENTATION
(AUTOENCODER - z)

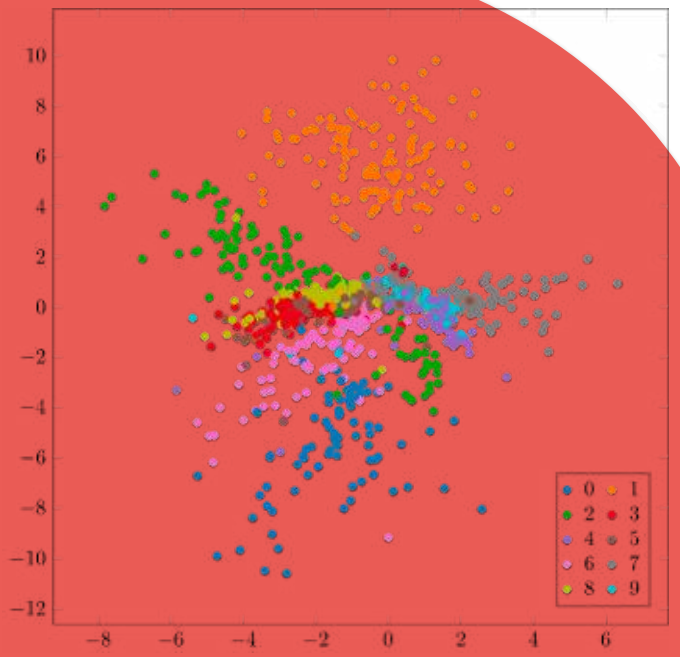


MODELLING OF P(z|x) WITH GMMs

GMMs

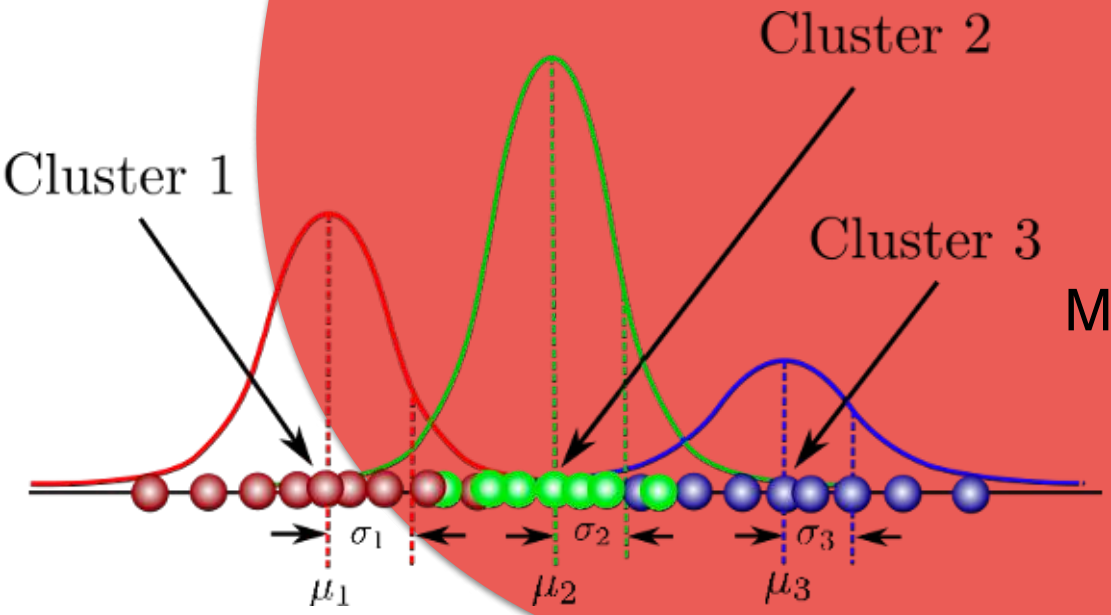
- ++ Both sampling and evaluating are straightforward with GMMs
- — However, performance scales poorly with increasing dimensionality
- — Depends on initial choices...
- And nothing guarantees that the latent space is well behaved as a mixture of gaussians

DATA



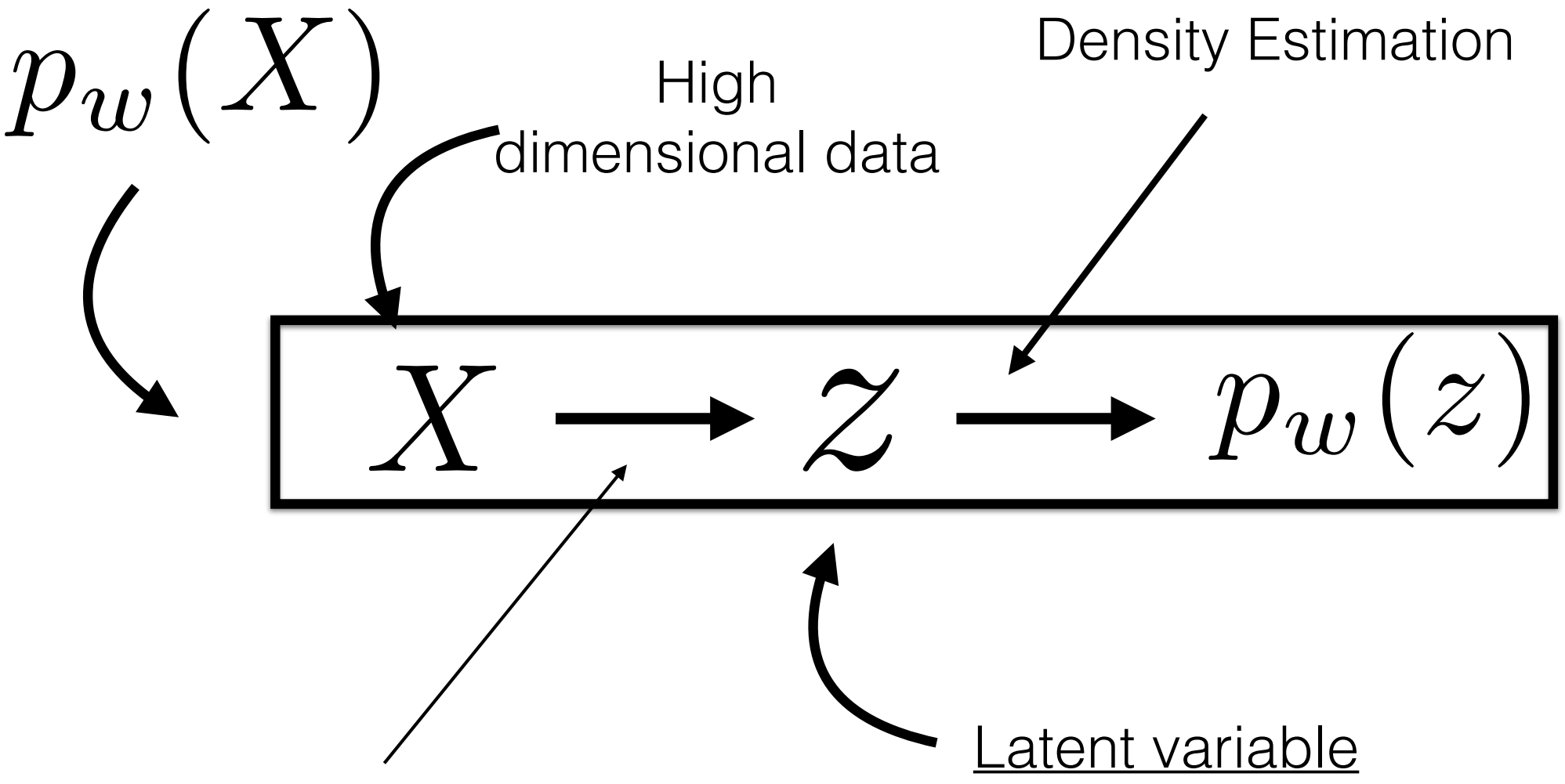
REPRESENTATION
(AUTOENCODER)

CAN WE COMBINE THESE 2 STEPS?



MODELING OF P(z|x) WITH GMMs

RECAP5b:


$p_w(X)$
High dimensional data
Density Estimation
X
z
$p_w(z)$
Latent variable
Representation learning / dimensionality reduction

Much of the recent progress in unsupervised deep learning has been to invent network architectures that are capable of solving either or both of these related problems directly, without resorting to any auxiliary methods

VAE

(VARIATIONAL AUTOENCODER)

GAN

(GENERATIVE ADVERSARIAL NETOWRK)

NF-ARF

(NORMALIZING FLOW, AUTOREGRESSIVE FLOW)

*** Not addressing score based models (diffusion) in this lecture**

Much of the recent progress in unsupervised deep learning has been to invent network architectures that are capable of solving either or both of these related problems directly, without resorting to any auxiliary methods

VAE

(VARIATIONAL AUTOENCODER)

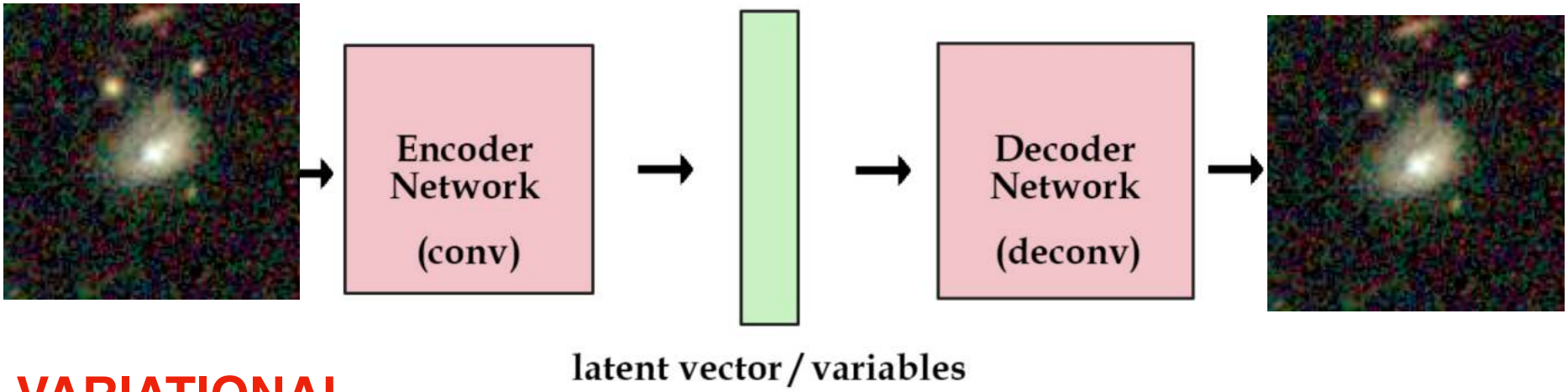
GAN

(GENERATIVE ADVERSARIAL NETOWRK)

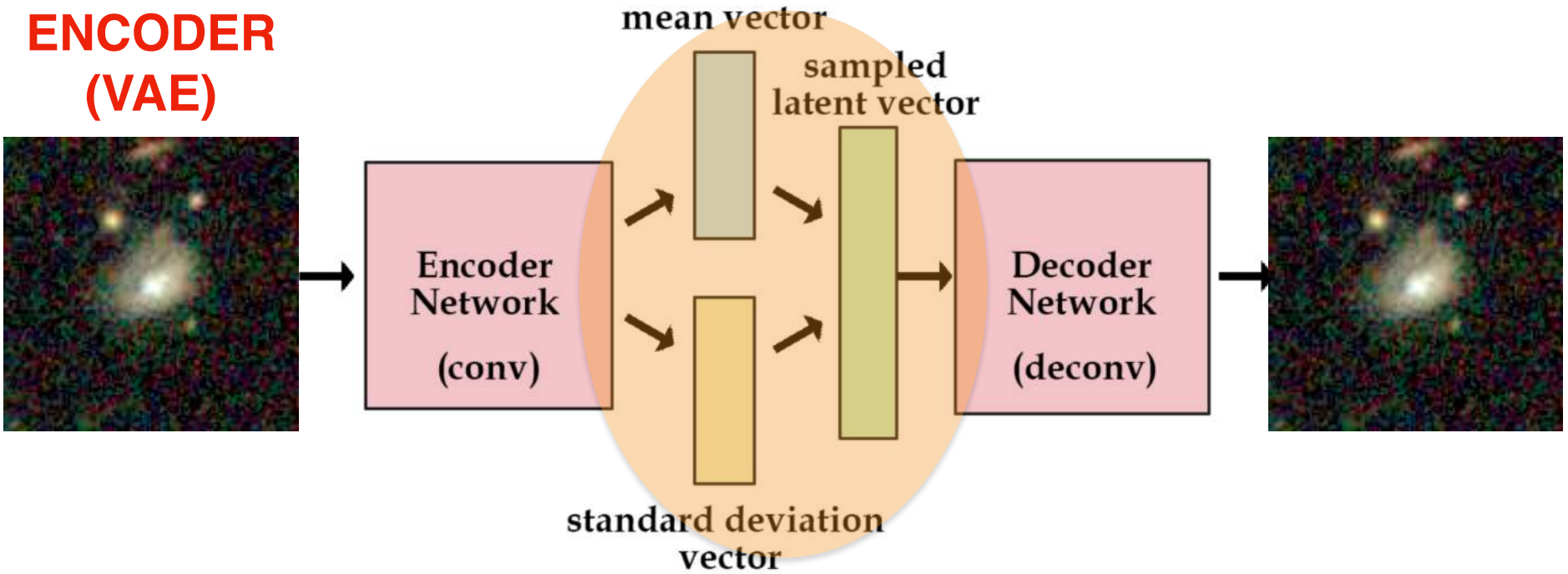
NF-ARF

(NORMALIZING FLOW, AUTOREGRESSIVE FLOW)

AUTO-ENCODER

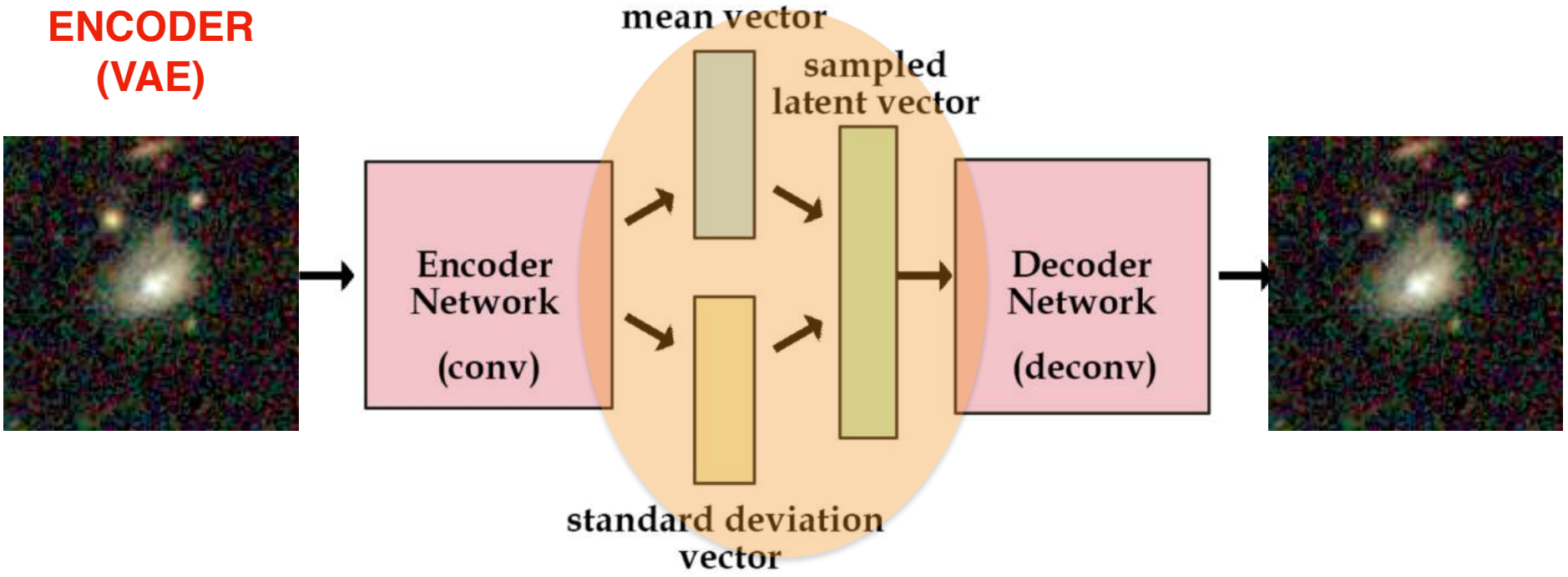


VARIATIONAL AUTO-ENCODER (VAE)



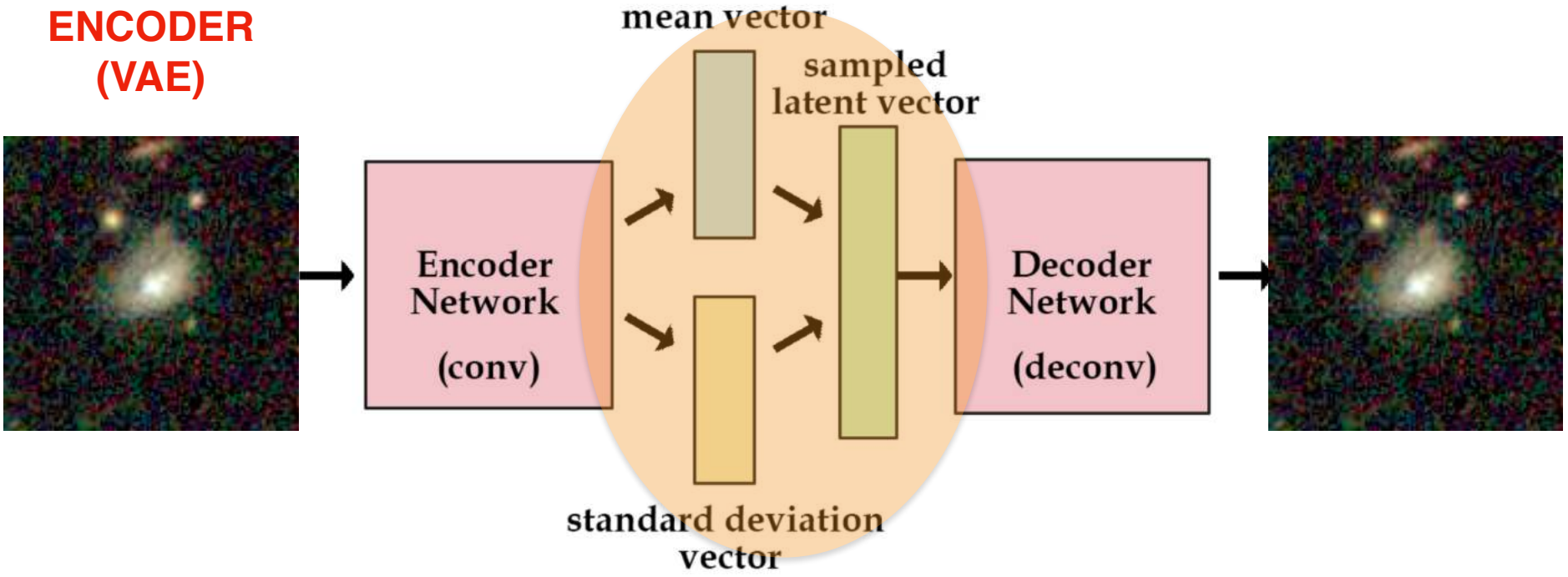
LET'S MODEL THE LATENT SPACE WITH A MIXTURE OF GAUSSIANS (reparametrization trick from the second lecture)

VARIATIONAL AUTO-ENCODER (VAE)



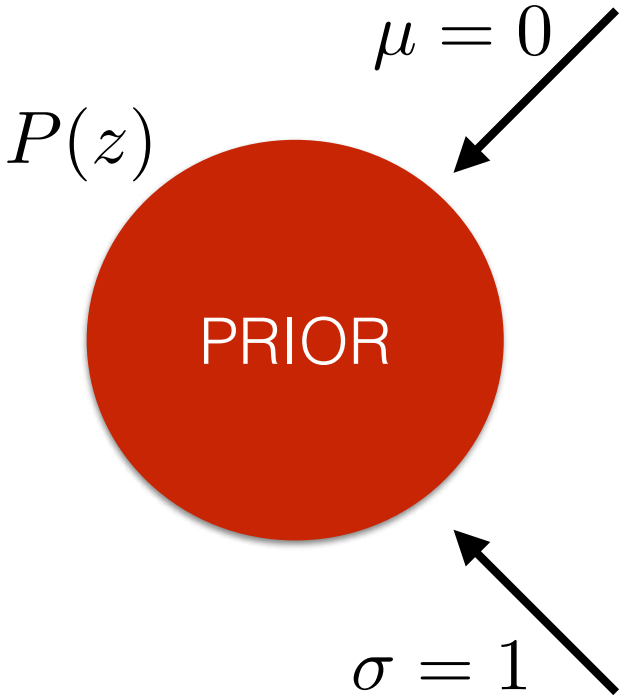
HOWEVER, NOTHING GUARANTEES US THAT P(z) CAN BE ACCURATELY MODELLED BY A MIXTURE OF GAUSSIANS….

VARIATIONAL AUTO-ENCODER (VAE)

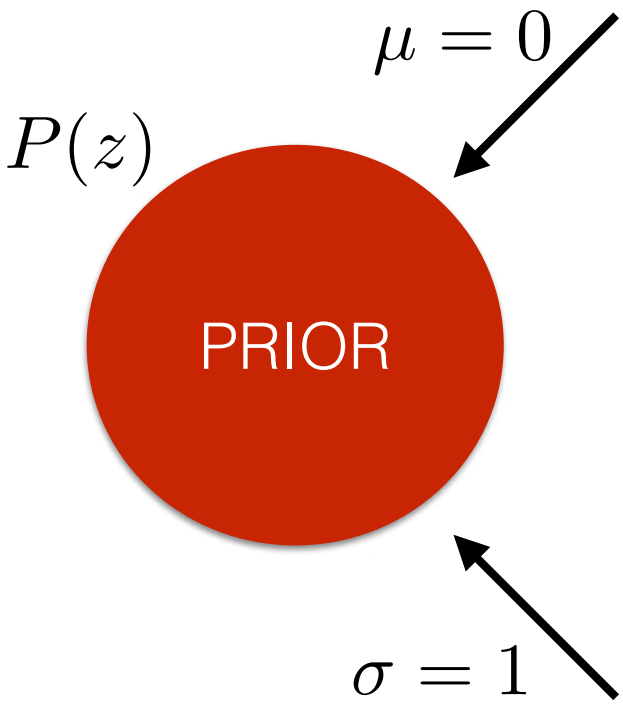


HOWEVER, NOTHING GUARANTEES US THAT THE LATENT SPACE CAN BE MODELLED BY A MIXTURE OF GAUSSIANS….

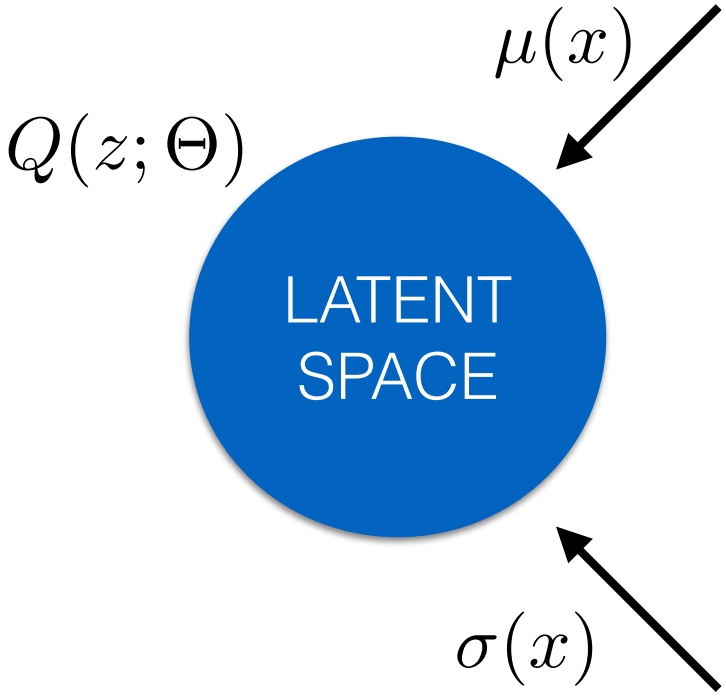
… LET'S FORCE IT TO BE GAUSSIAN LIKE!



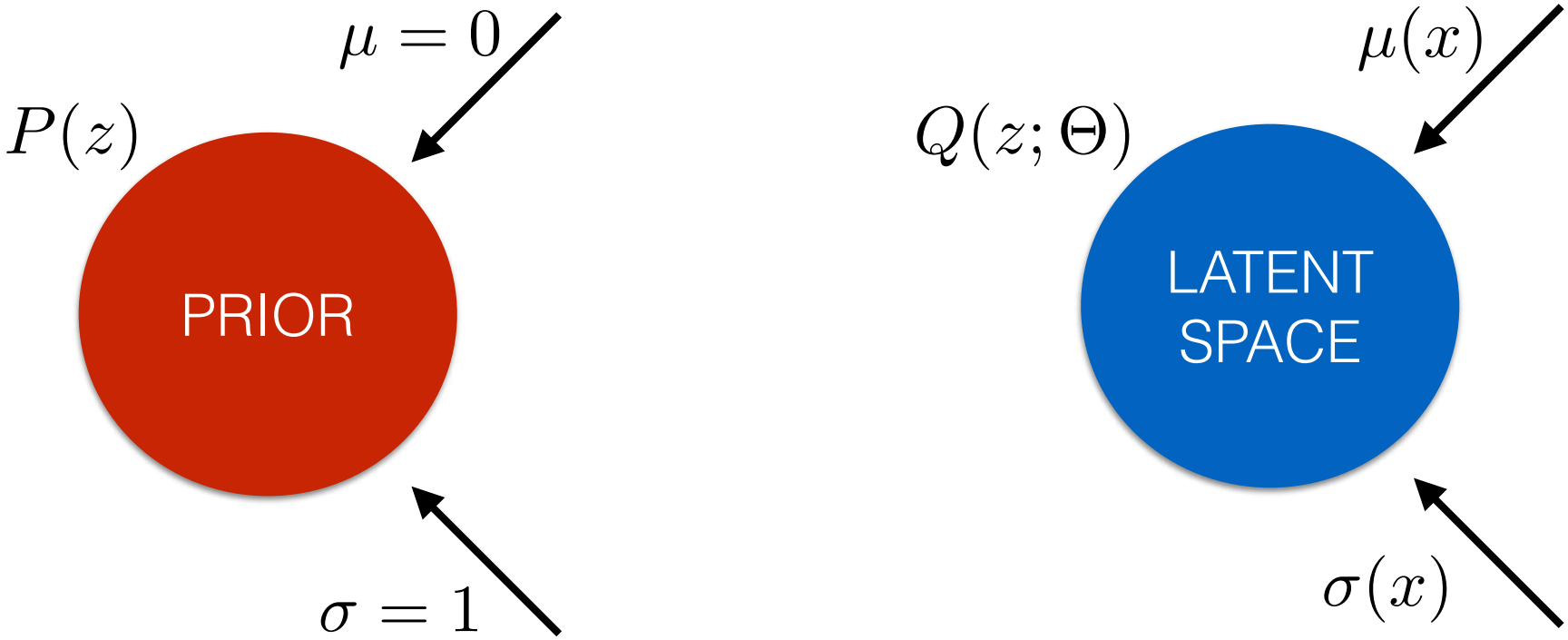
WE ASSUME A SIMPLE PRIOR



WE ASSUME A SIMPLE PRIOR

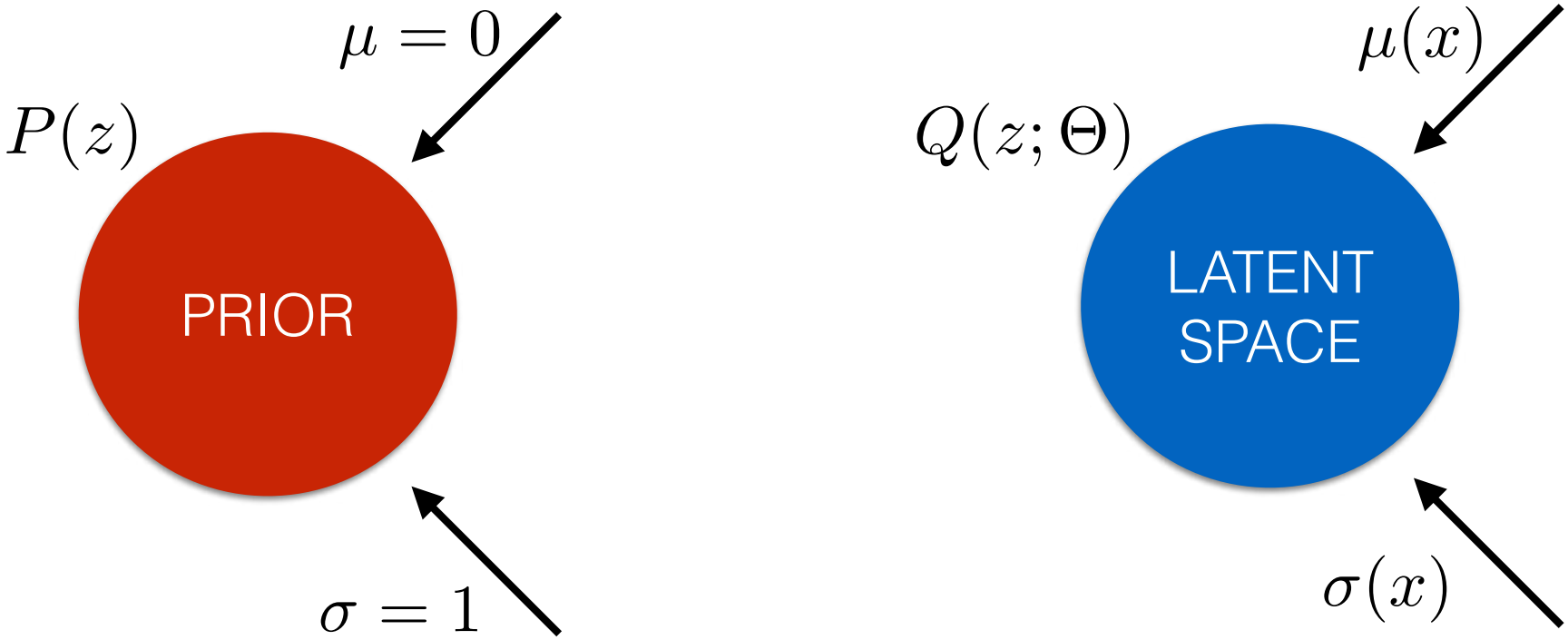


LATENT SPACE MODELING



WE ASSUME A SIMPLE PRIOR ⟺ LATENT SPACE MODELLING

WE WANT Q TO BE CLOSE TO THE PRIOR…



WE ASSUME A SIMPLE PRIOR ⟺ LATENT SPACE MODELLING

WE WANT Q TO BE CLOSE TO THE PRIOR…

$$D_{\mathrm{KL}}(P \parallel Q) = \sum_{x \in \mathcal{X}} P(x) \log\left(\frac{P(x)}{Q(x)}\right). \qquad D_{\mathrm{KL}}(P \parallel Q) = \int_{\mathcal{X}} \log\left(\frac{dP}{dQ}\right) \frac{dP}{dQ}\, dQ,$$

WE MINIMIZE THE K-L DIVERGENCE BETWEEN P AND Q

WHAT WOULD BE THEN THE LOSS FUNCTION OF A VAE?

The key insight of VAE is that we are actually performing **variational inference** here, which then tells us what the loss function should be...

$$-\text{ELBO} = \langle \log P(\mathbf{x} \mid \mathbf{z}) \rangle_{z \sim Q} + \text{KL}\left(Q(\mathbf{z}; \Theta) \parallel P(\mathbf{z})\right) \ ,$$

~MSE

~Gaussian

REGULARIZATION TERM

WHAT WOULD BE THEN THE LOSS FUNCTION OF A VAE?

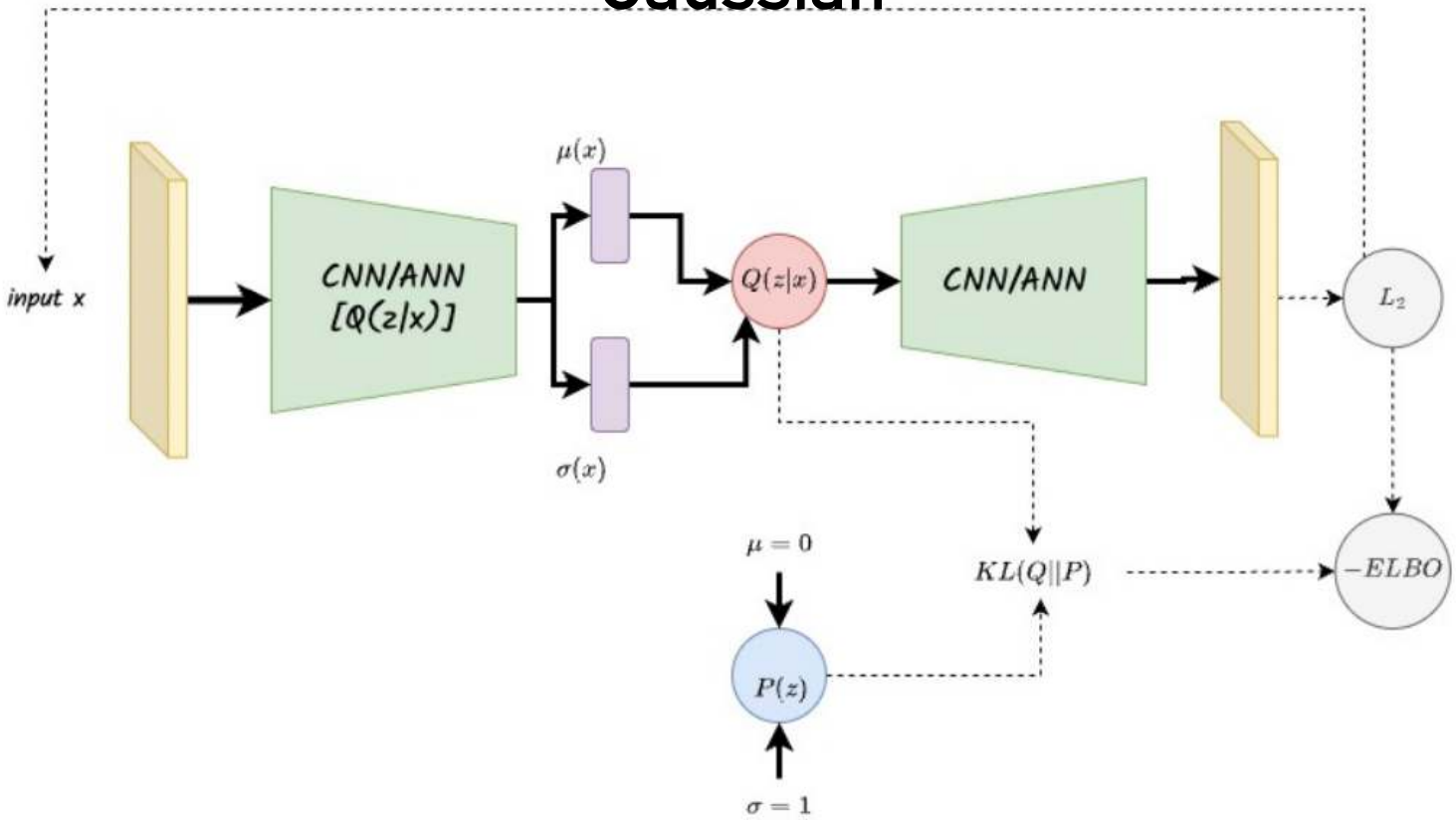
The key insight of VAE is that we are actually performing **variational inference** here, which then tells us what the loss function should be...

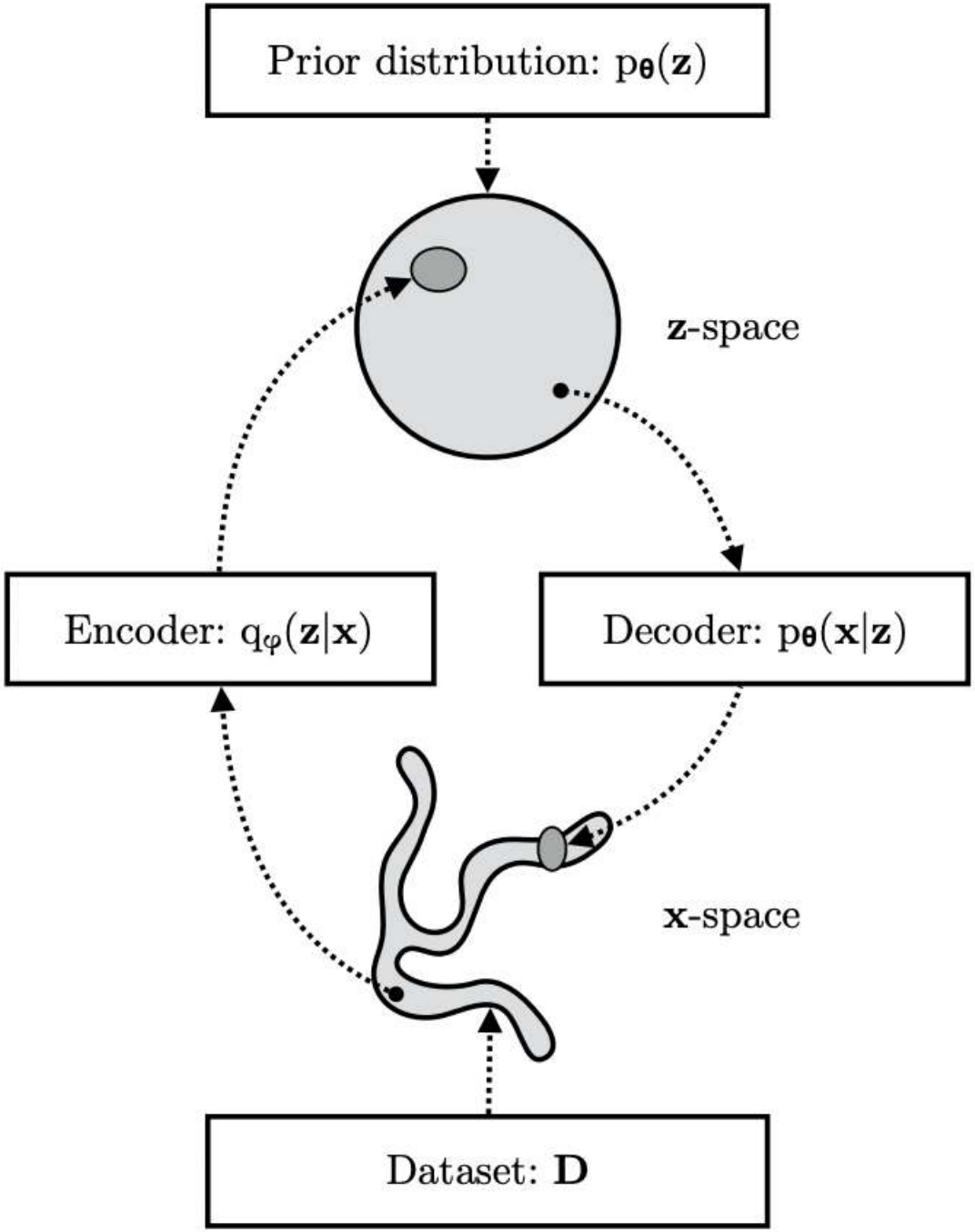
$$-\text{ELBO} = \langle \log P(\mathbf{x} \mid \mathbf{z}) \rangle_{z \sim Q} + \text{KL}\left(Q(\mathbf{z}; \Theta) \parallel P(\mathbf{z})\right) \ ,$$

L2 LOSS

REGULARIZATION TERM

~Gaussian

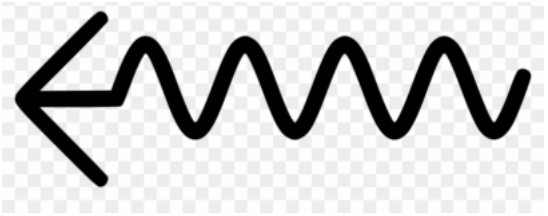


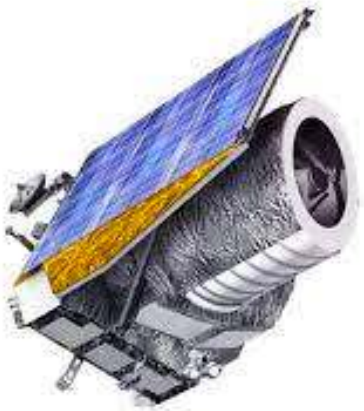


[Kingma and Welling, 2019]

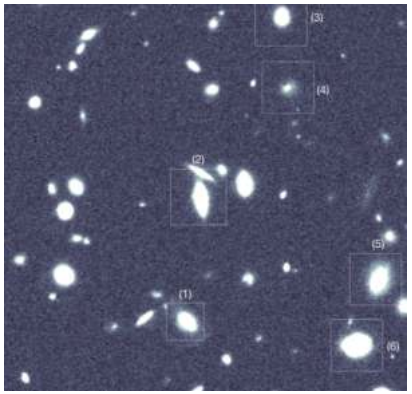
Preparing a space mission requires very detailed simulations

From photons

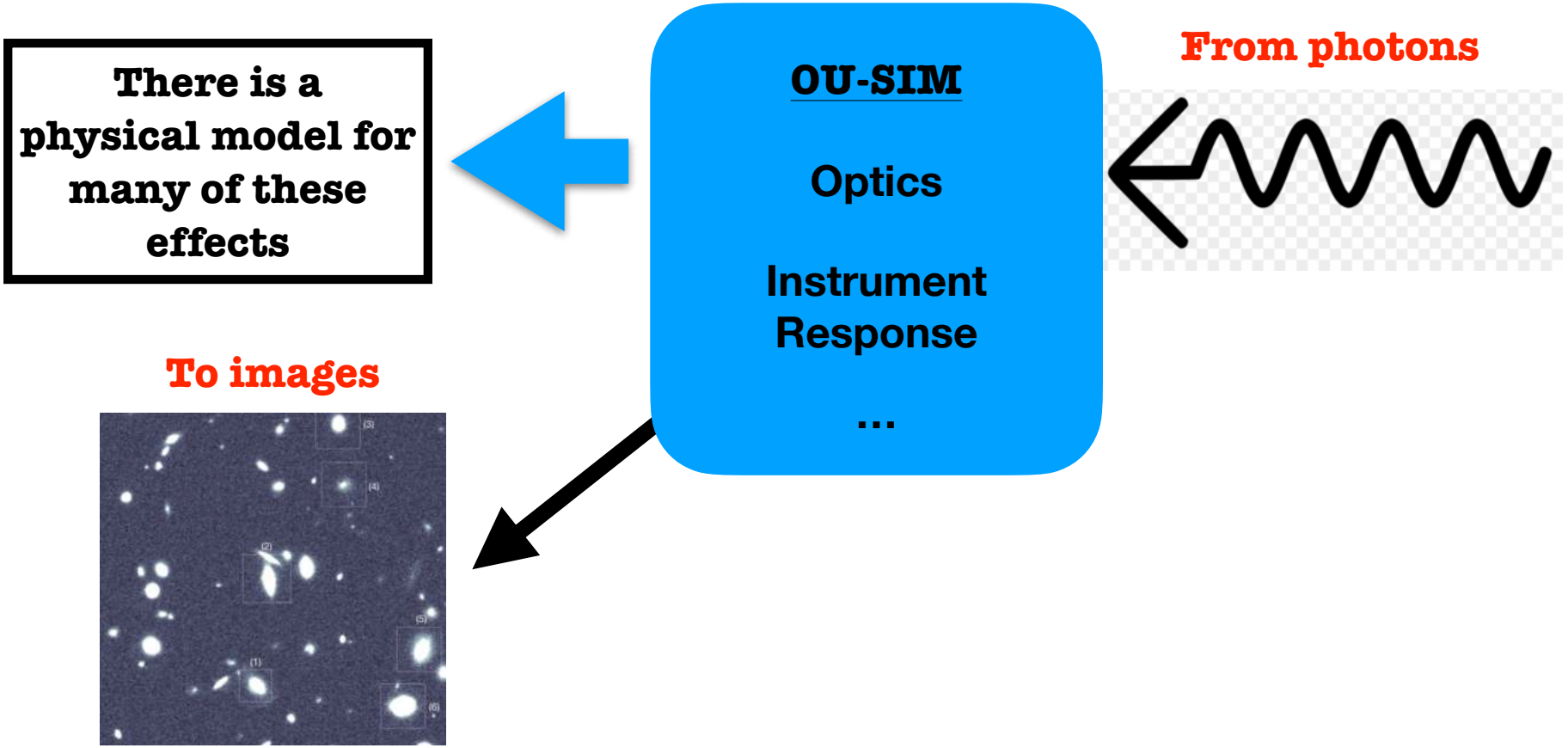




To images



Preparing a space mission requires very detailed simulations



Preparing a space mission requires very detailed simulations

There is a physical model for many of these effects

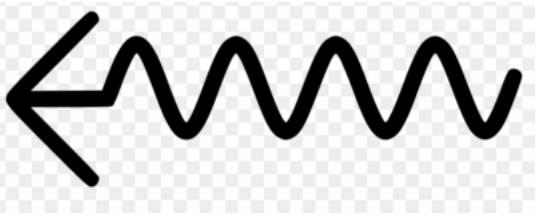
OU-SIM

Optics

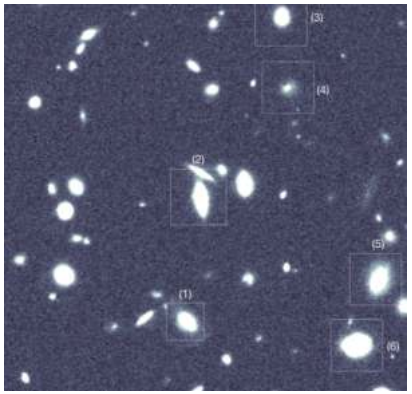
Instrument Response

...

From photons



To images



Others are more difficult to model

Preparing a space mission requires very detailed simulations

There is a physical model for many of these effects

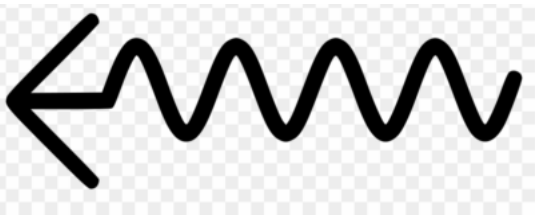
OU-SIM

Optics

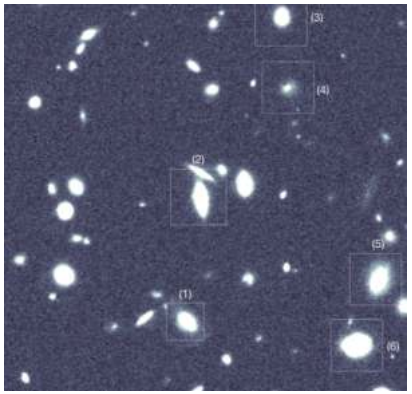
Instrument Response

...

From photons



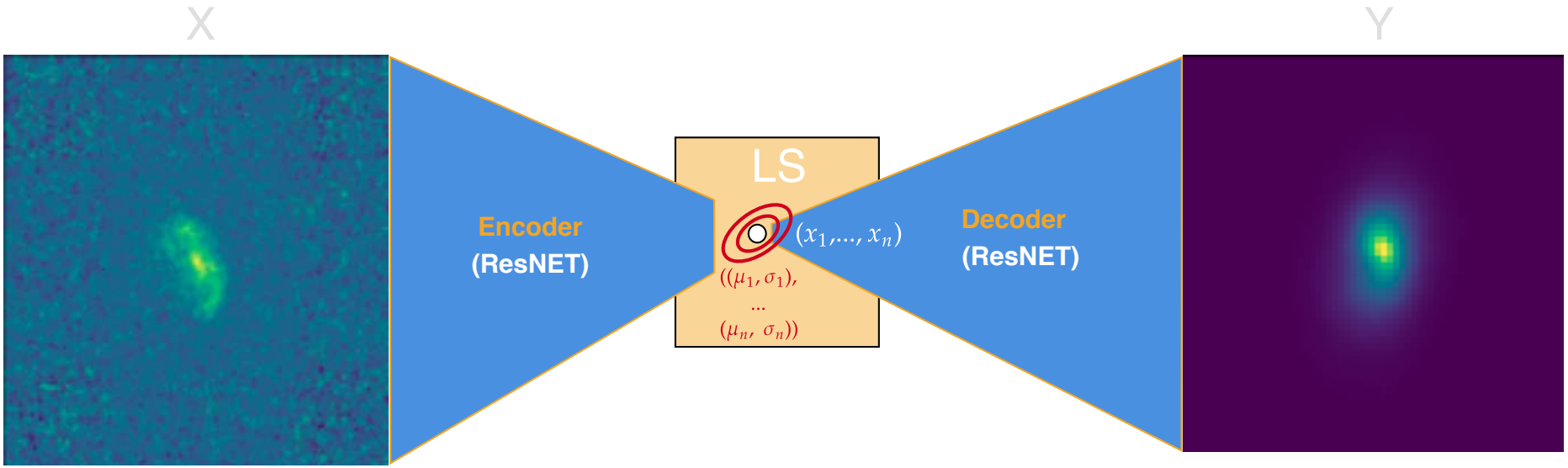
To images



Neural Network

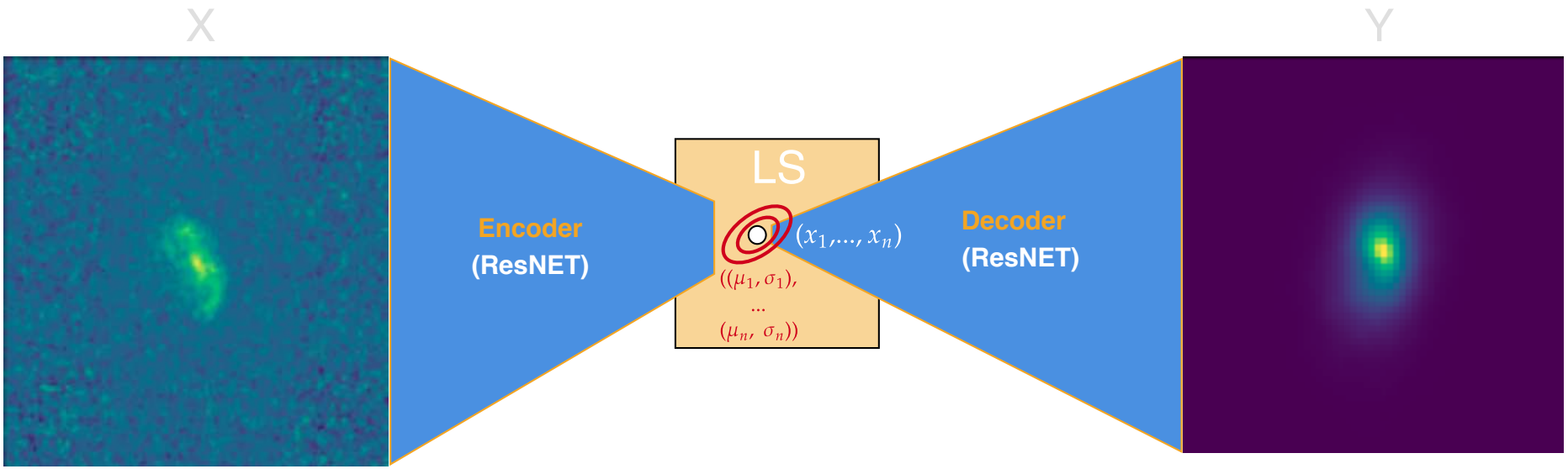
Others are more difficult to model

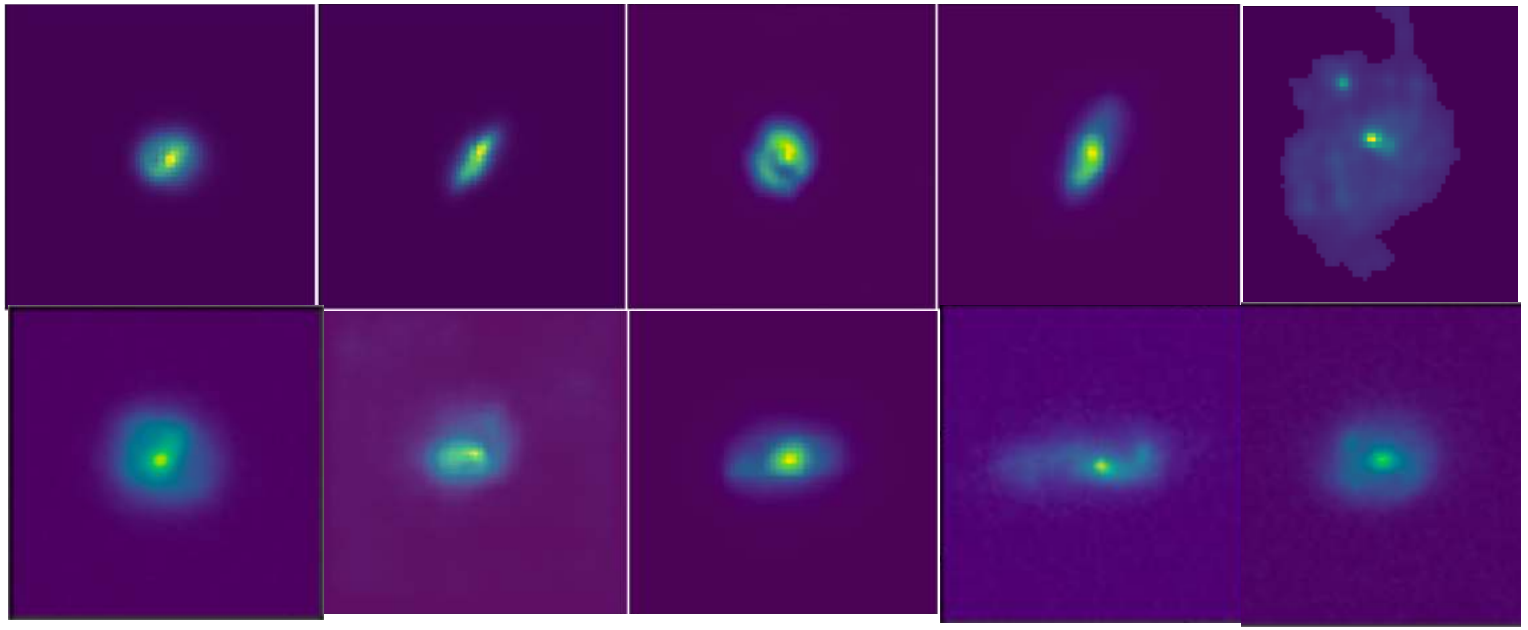
Include a Generative Model for galaxy generation in the Euclid Simulation Pipeline to model process for which we do not have a physical model



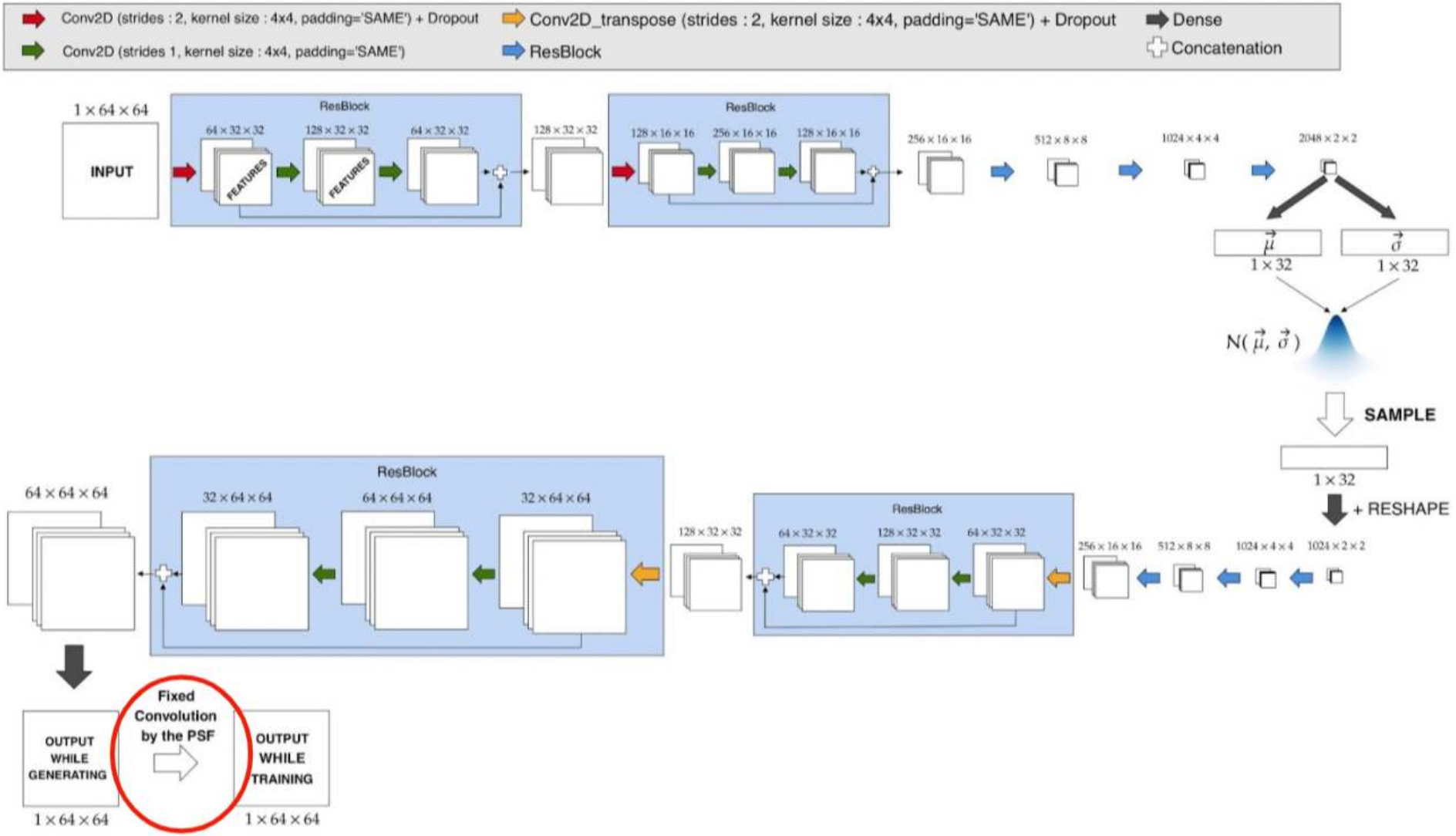
Bretonnière+21

Include a Generative Model for galaxy generation in the Euclid Simulation Pipeline to model process for which we do not have a physical model



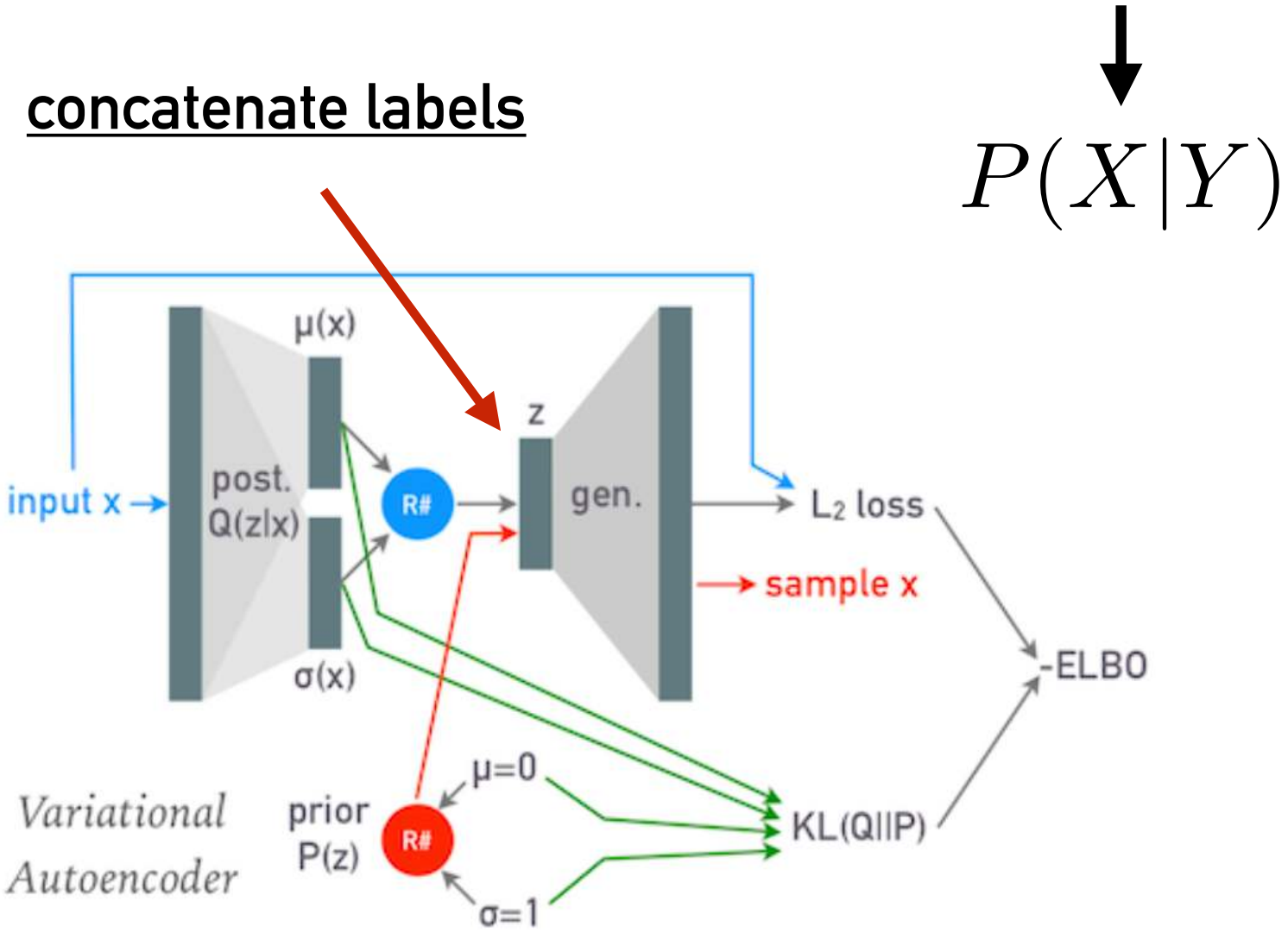


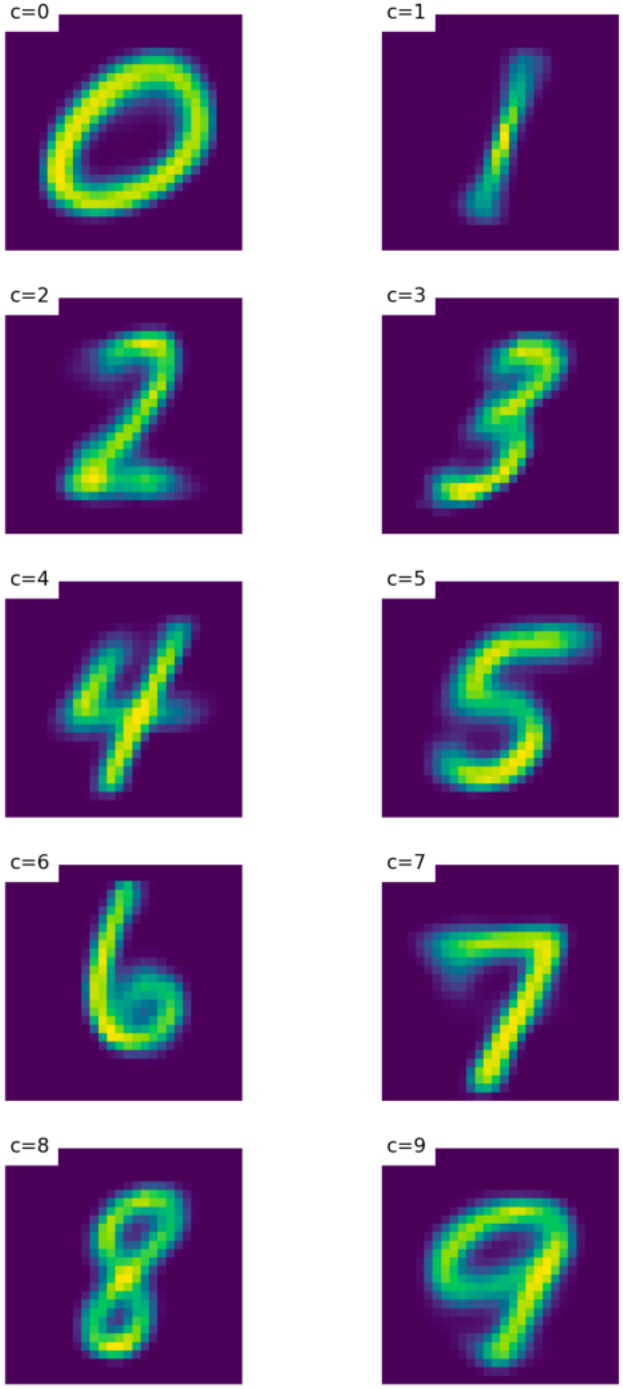
Bretonnière+21



Physical information is included as well within the architecture

VAE's can also be conditioned, to generate a given class



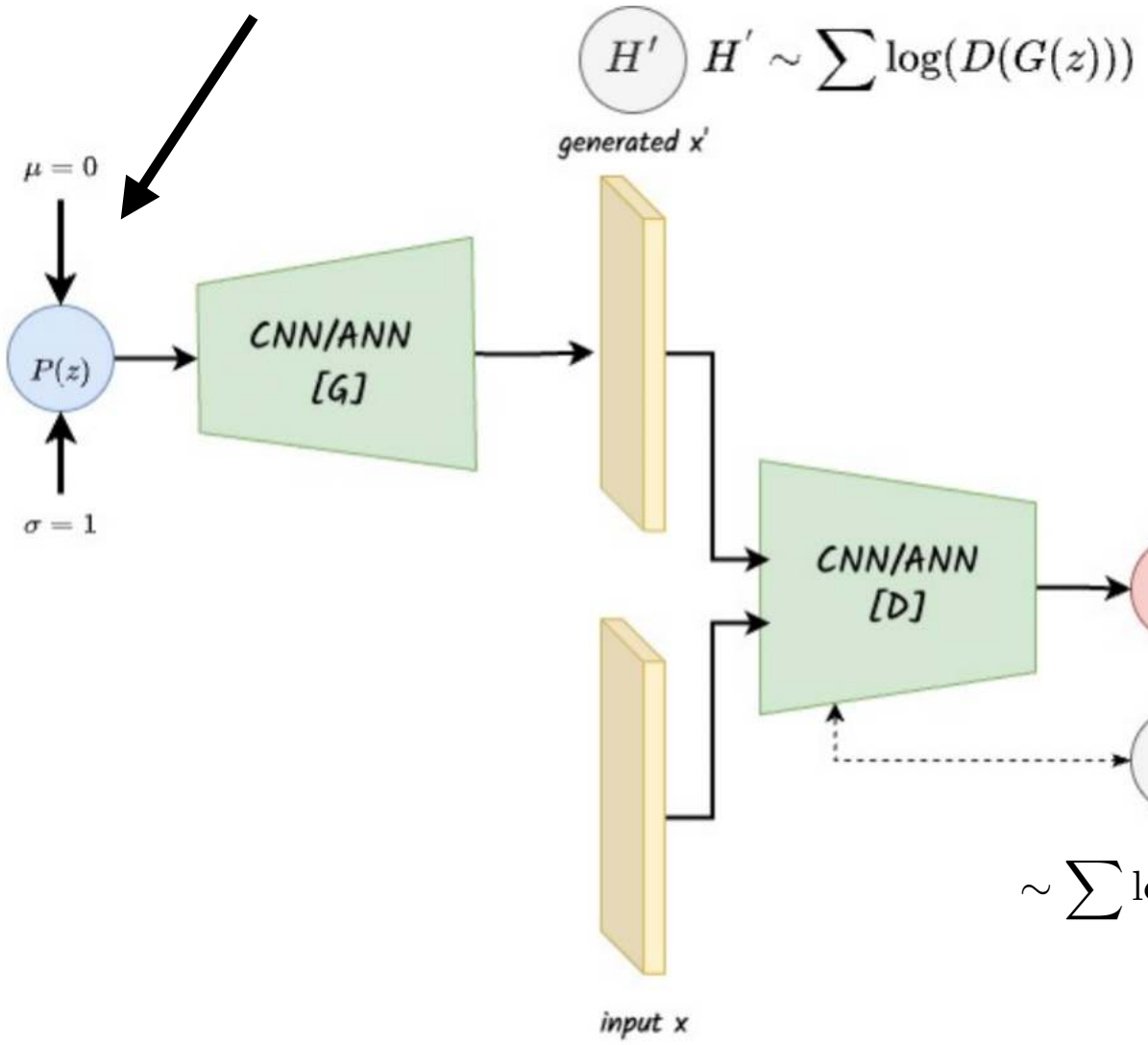

c=0
c=1
c=2
c=3
c=4
c=5
c=6
c=7
c=8
c=9

Generative Adversarial Networks

As for VAEs,
the goal of generative Adversarial Networks (GANs) is to estimate
$p(z|x)$

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

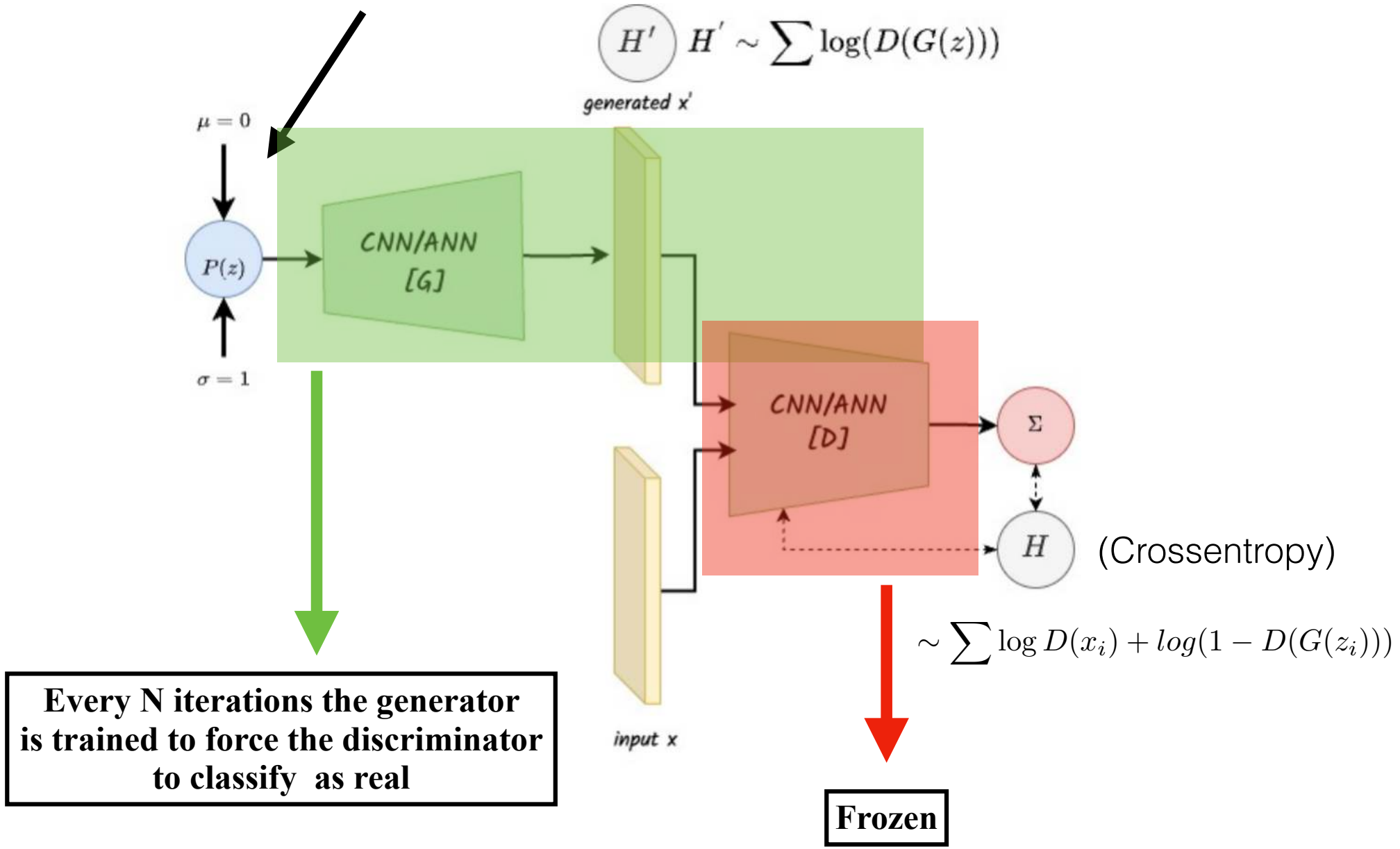
The latent variable is here



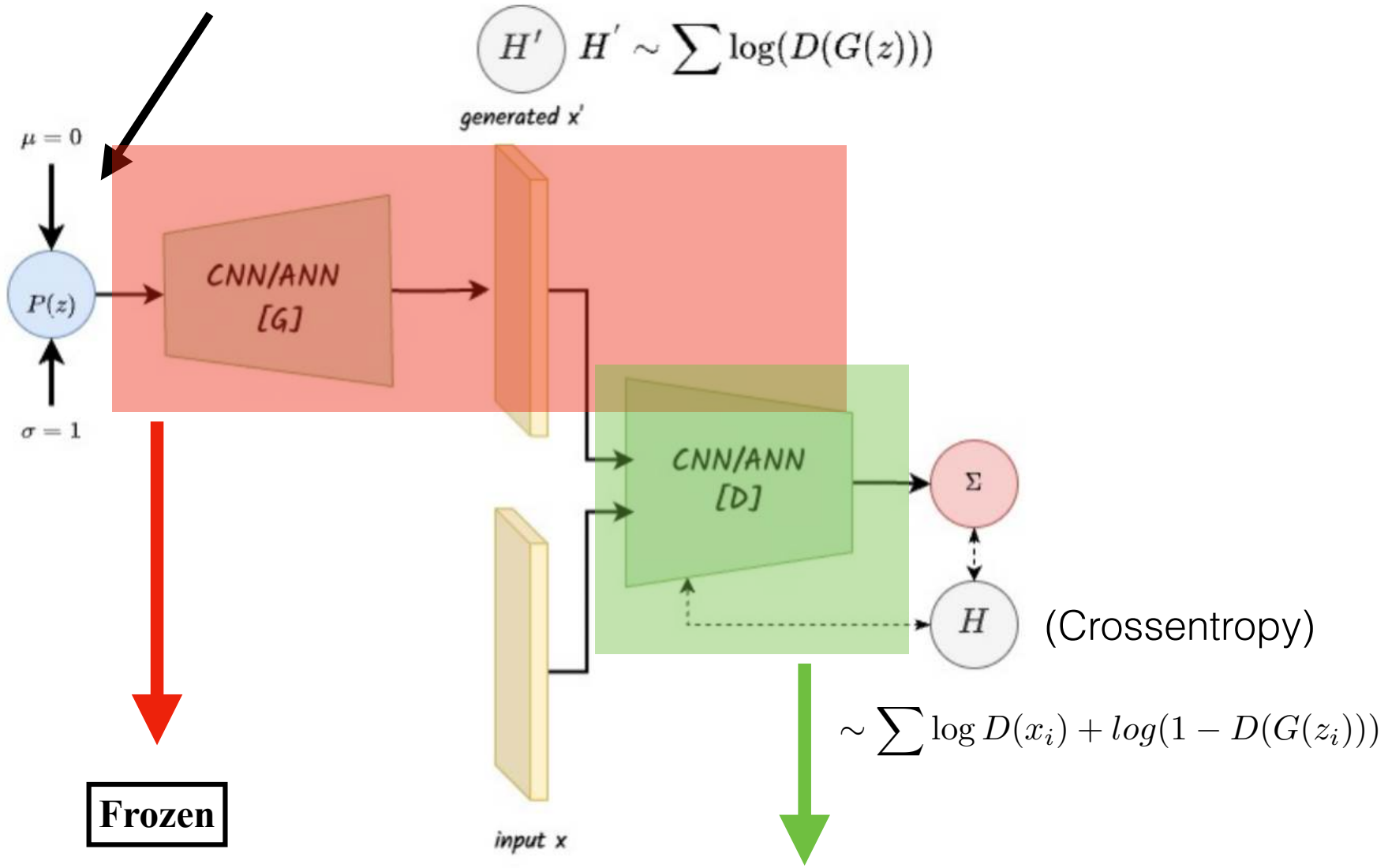
(Crossentropy)

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

The latent variable is here



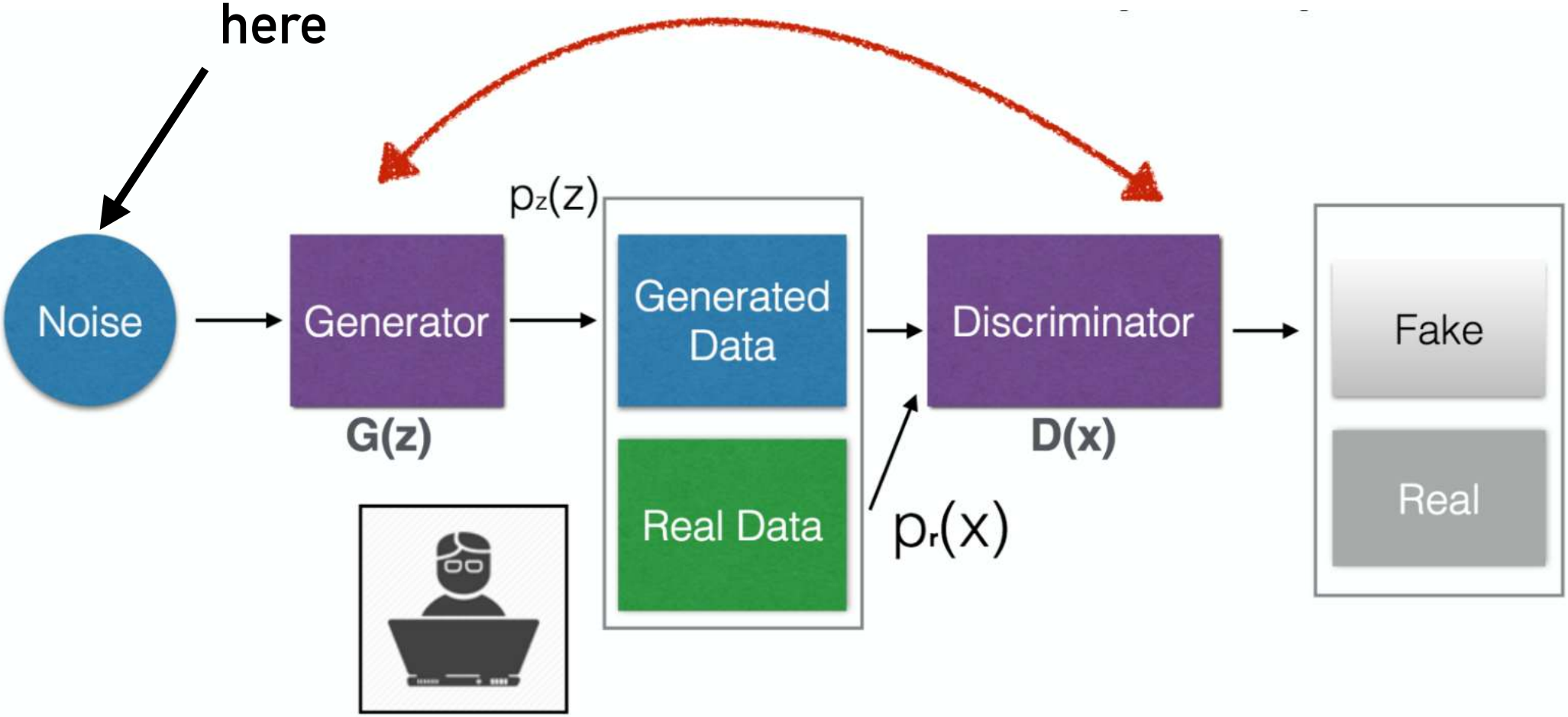
The latent variable is here



GENERATIVE ADVERSARIAL NETWORKS

(Goodfellow+14)

The latent variable is here

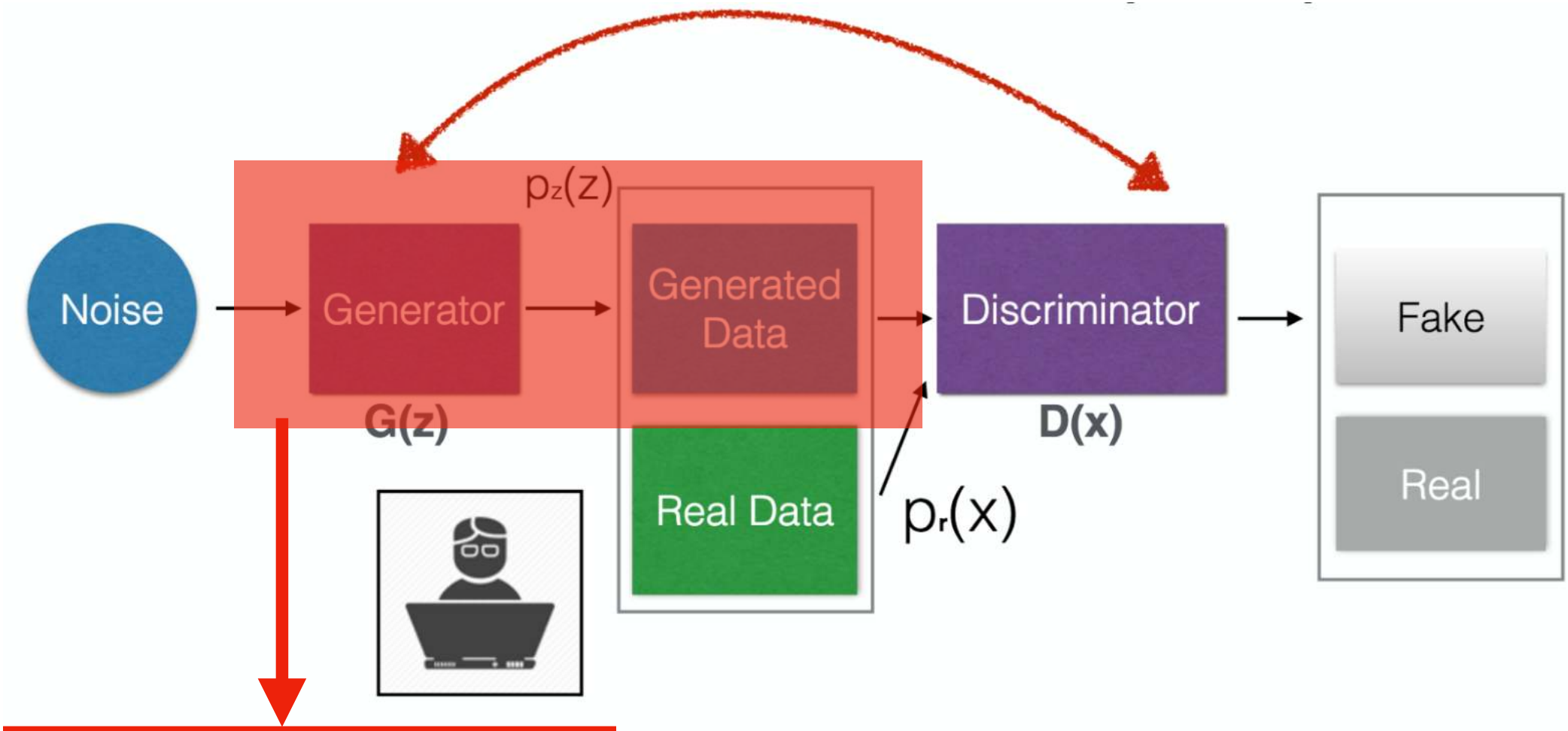


TWO COMPETING NETWORKS

GENERATIVE ADVERSARIAL NETWORKS

(Goodfellow+)

TWO COMPETING NETWORKS

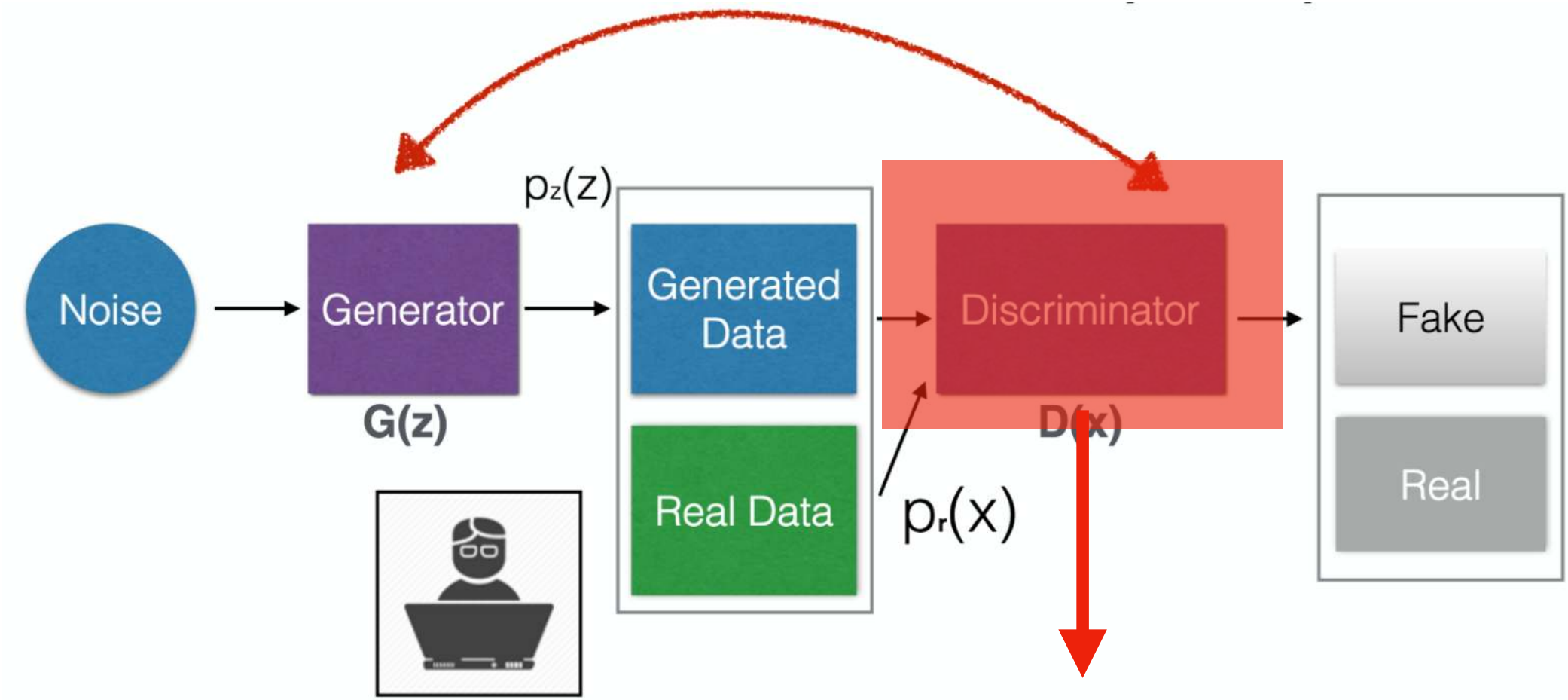


Every N iterations the generator is trained to force the discriminator to classify as real

GENERATIVE ADVERSARIAL NETWORKS

(Goodfellow+)

TWO COMPETING NETWORKS

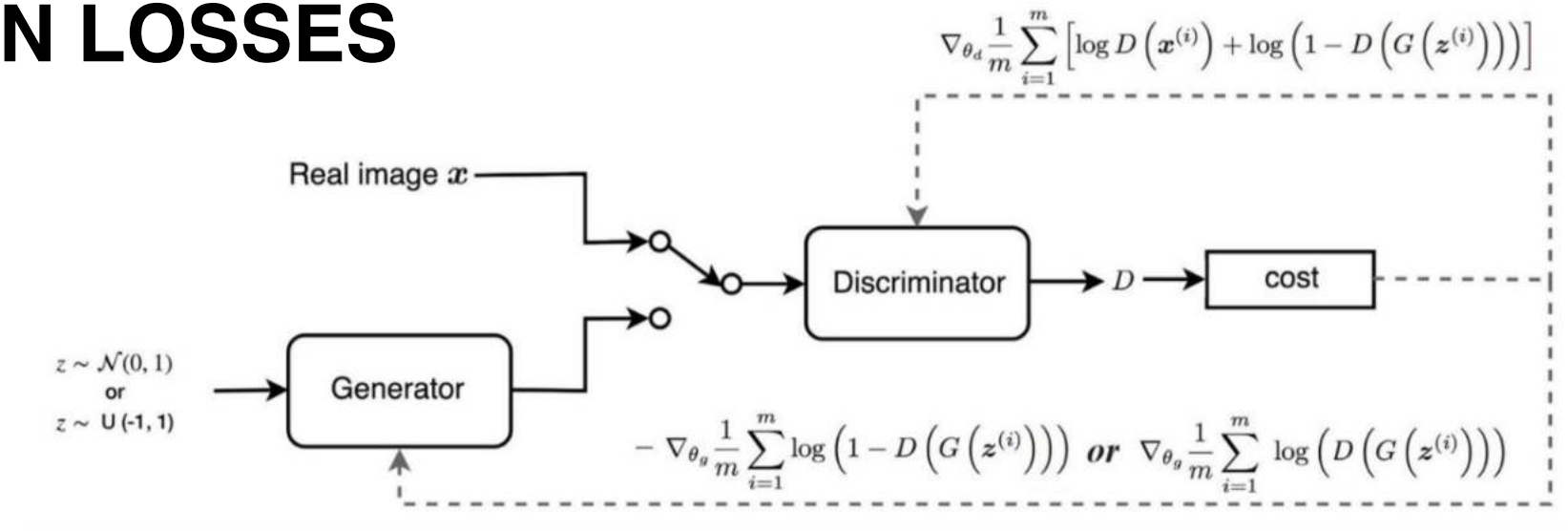


Every N iterations the discriminator is trained to force to distinguish between real and fake

IN PRACTICE

GAN LOSSES

DISCRIMINATOR LOSS (CROSS-ENTROPY)

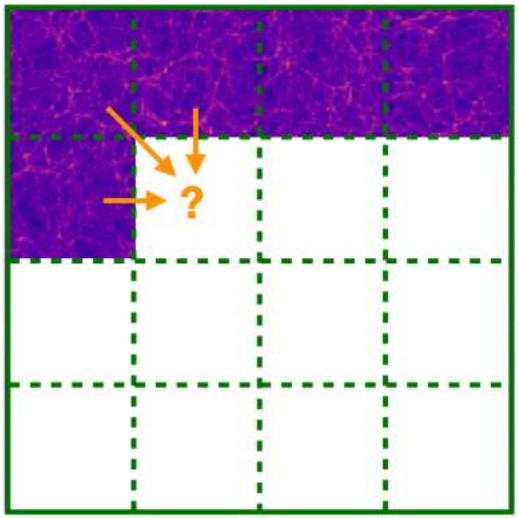


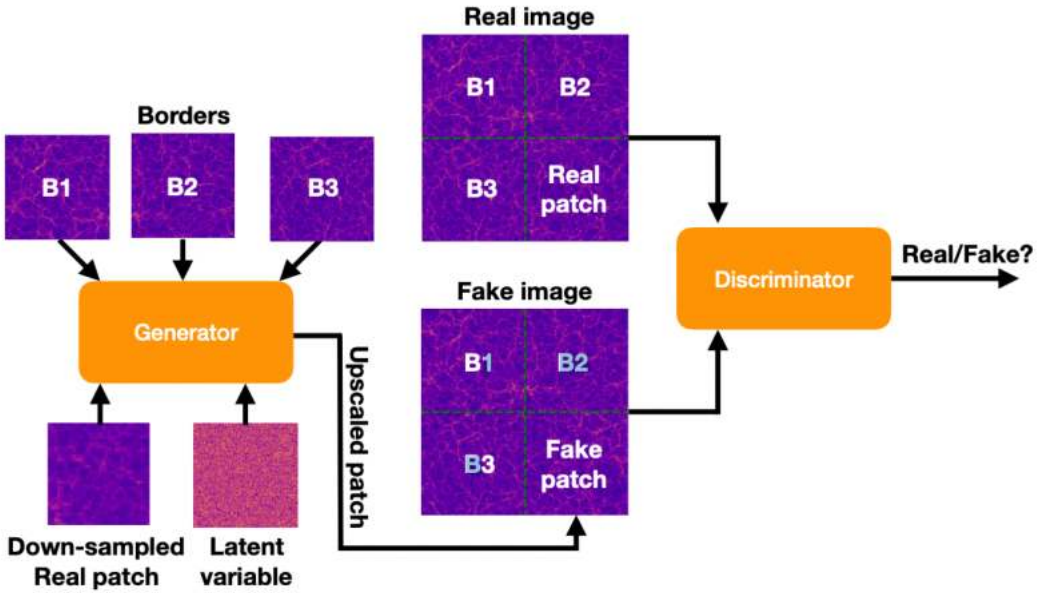
GENERATOR LOSS

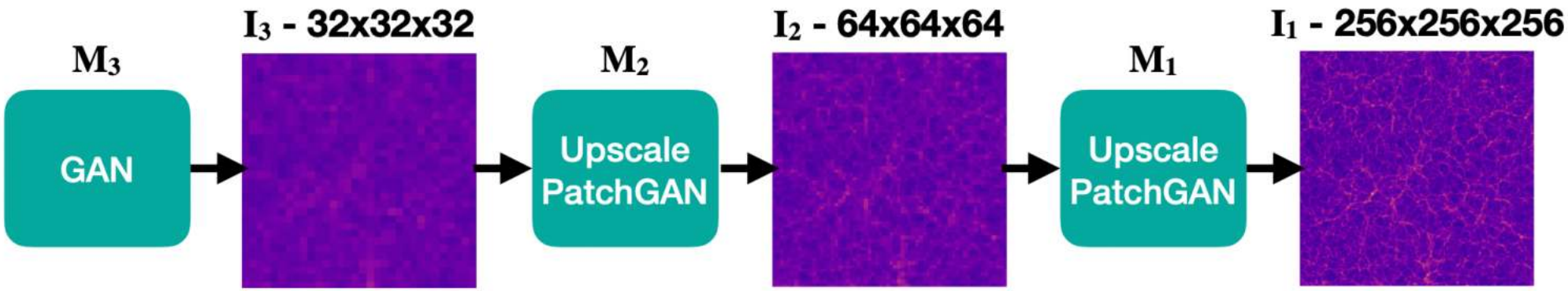
GANs have remained state-of-the-art for data generation (until diffusion…)

Karras+19

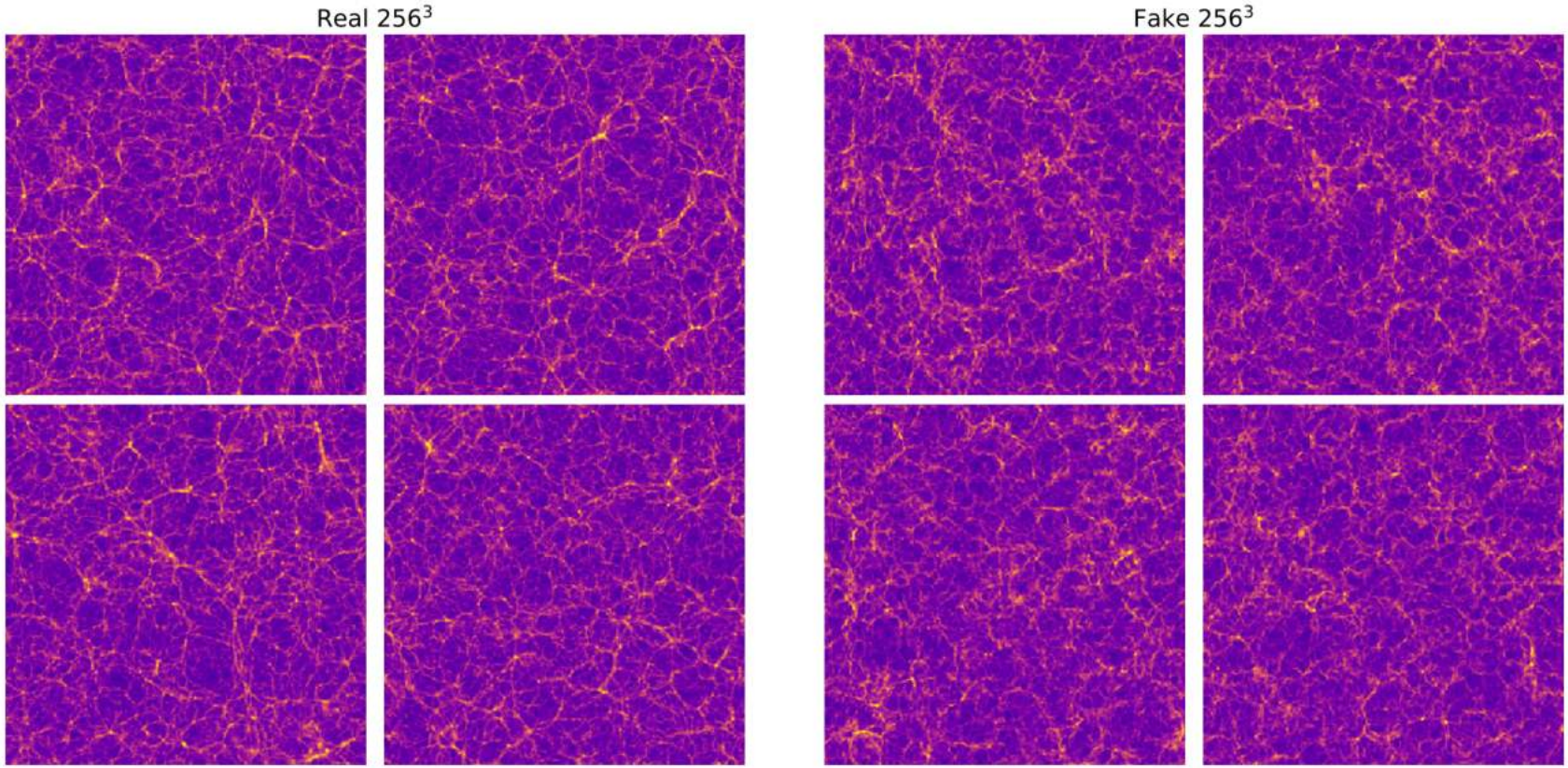
N-body emulation by Deep Generative Modelling

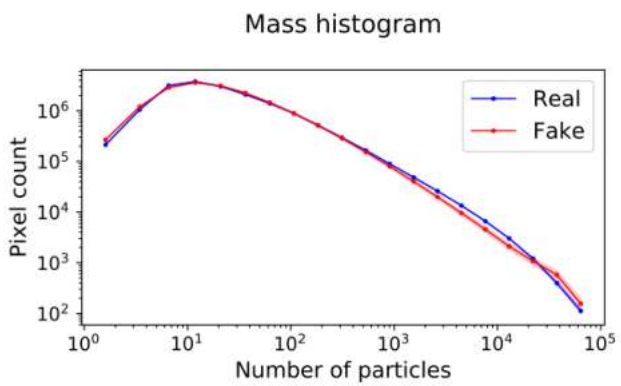


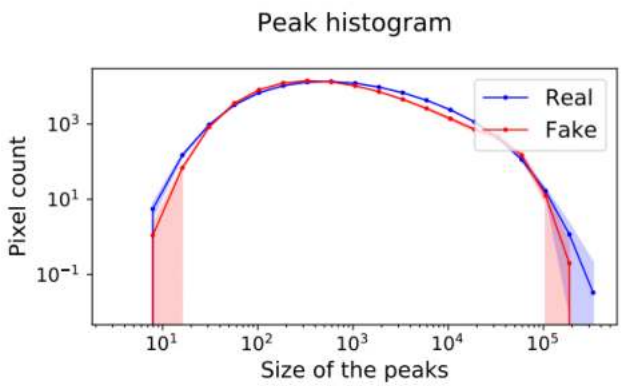


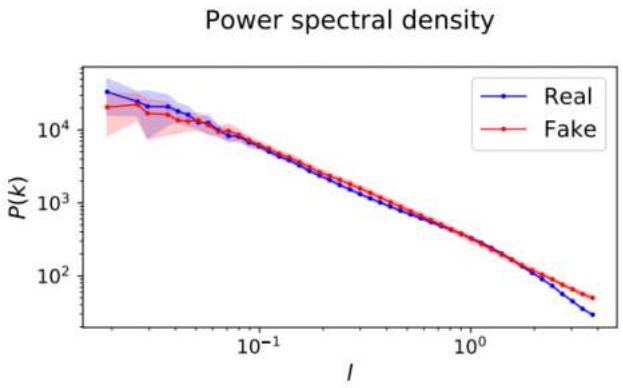


Perraudin+19

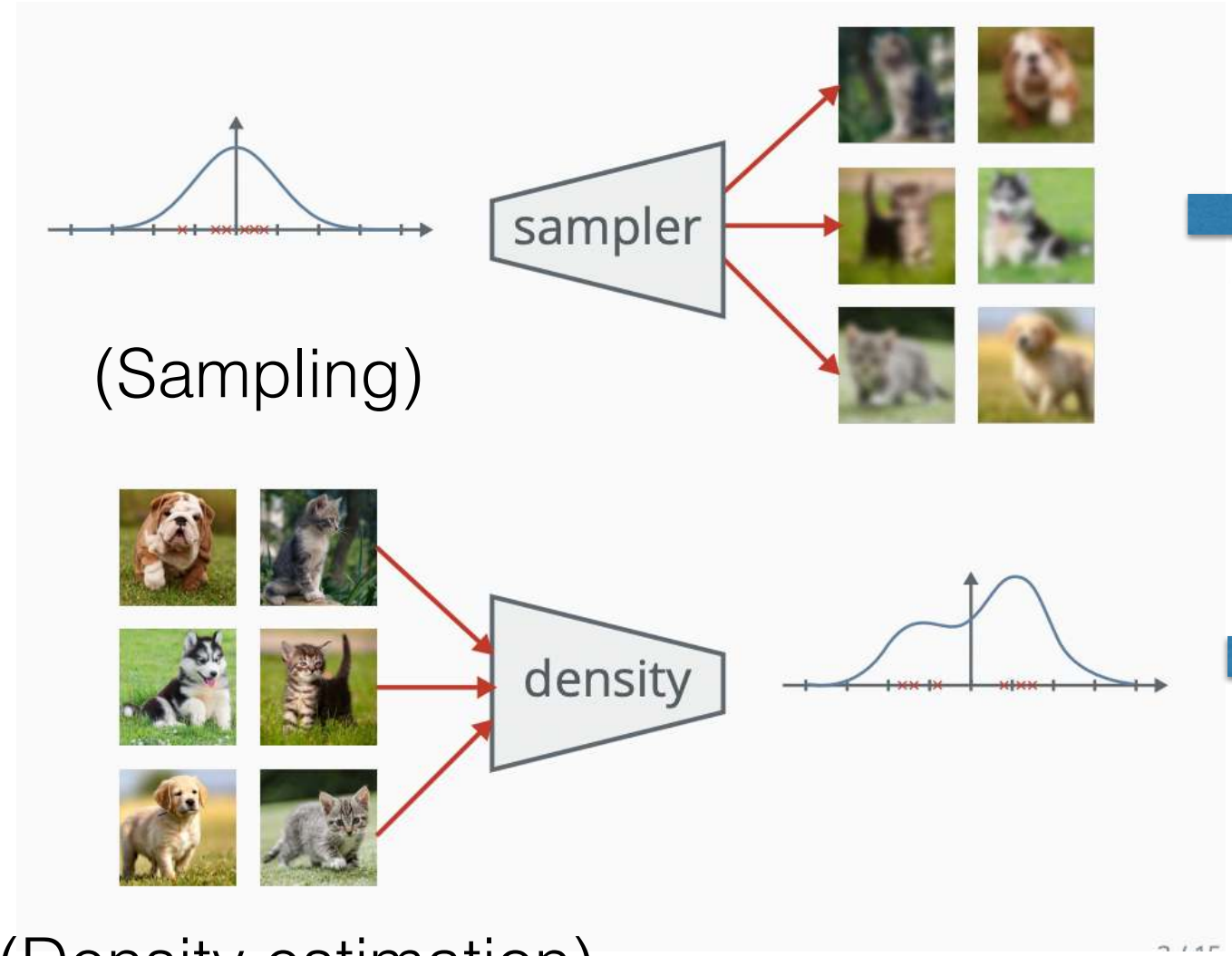








Perraudin+19



(Sampling)

(Density estimation)

easy with
VAEs and GANs

~ VAEs
difficult with
GANs

WHAT IS AN ANOMALY OR OUTLIER?

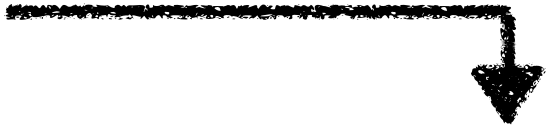
$$p(X)$$

you data follows some probability distribution p

WHAT IS AN ANOMALY OR OUTLIER?

$$p(X)$$

you data follows some probability distribution p



Then an object will be anomalous if:

$$p(x_i) < \epsilon$$

WHAT IS AN ANOMALY OR OUTLIER?

$$p(X)$$

you data follows some probability distribution p

Then an object will be anomalous if:

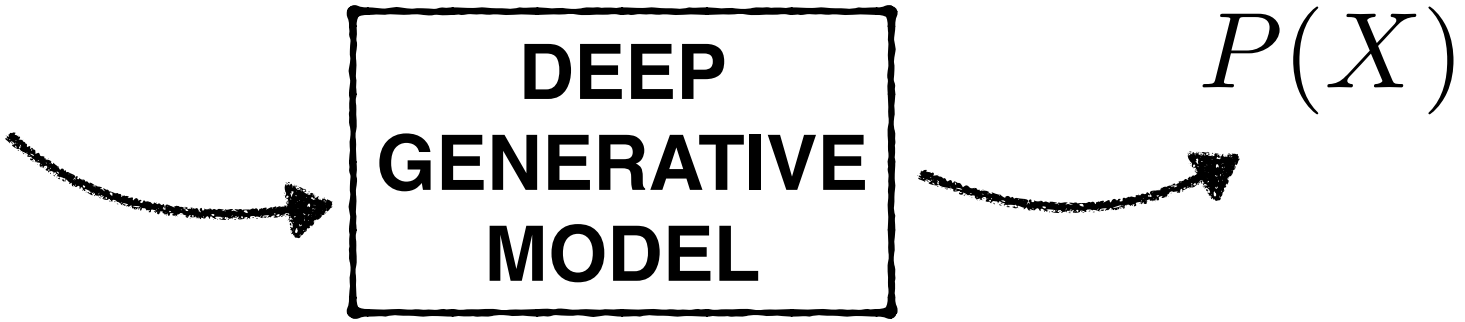
$$p(x_i) < \epsilon$$

HOW DO WE COMPUTE THE PROBABILITY DISTRIBUTION p?

GENERATIVE MODELS DO PRECISELY THAT:

DATA:

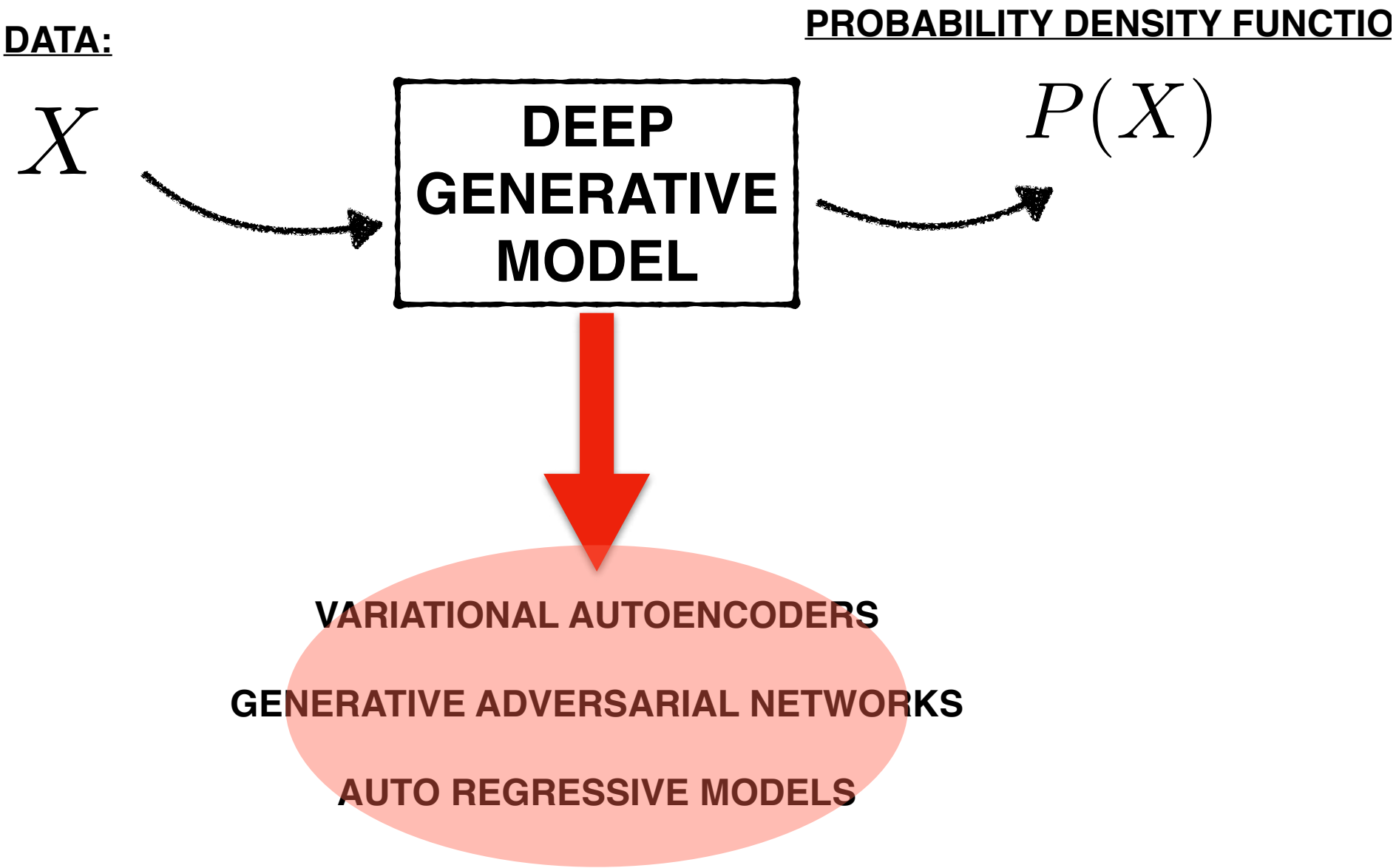
X



PROBABILITY DENSITY FUNCTIO

$P(X)$

GENERATIVE MODELS DO PRECISELY THAT:



Razavi+19

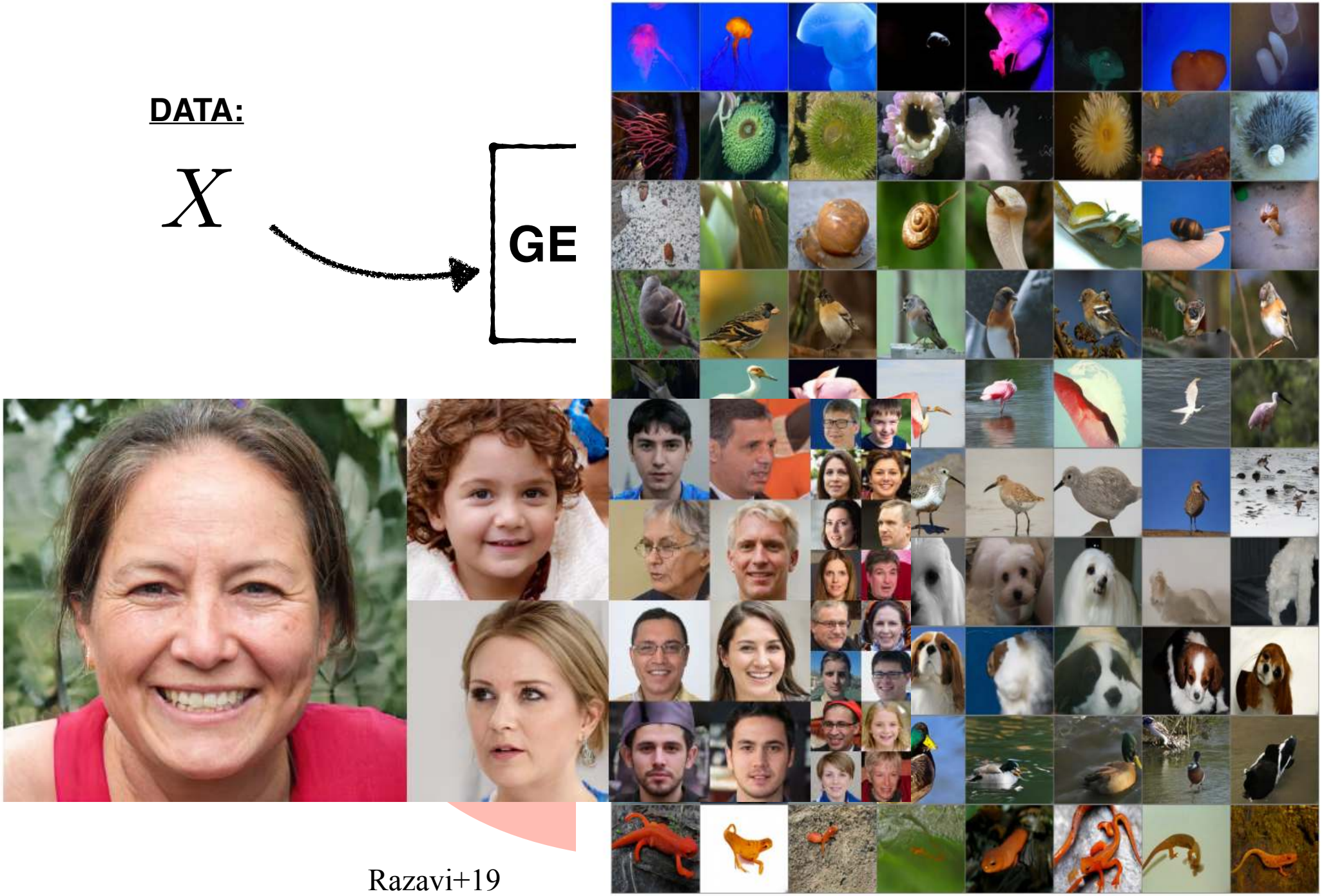
GENERATIVE MODELS DO PRECISELY THAT:

DATA:

X

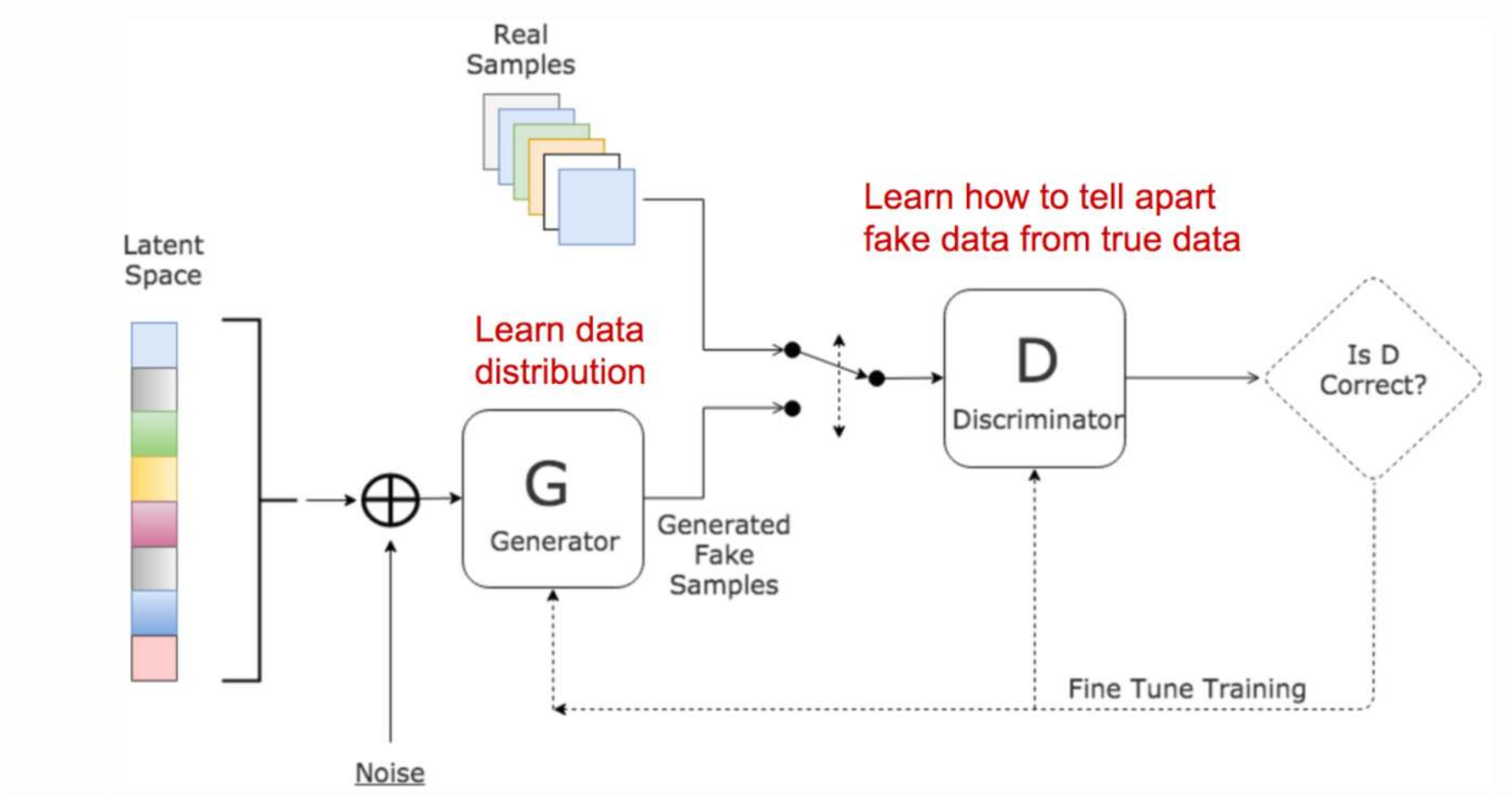
GE

O



Razavi+19

USE GENERATIVE MODELING TO LEARN P(X) [NORMAL DATA]



(Image source: www.kdnuggets.com/2017/01/generative-…-learning.html)

Hyper Suprime Cam Survey (HSC) (Subaru telesocpe 1400 sq. deg, r~26, 436 million

WGAN $\longrightarrow$ $P_{HSC}(X)$

(20 < i < 20.5; 1 million)

Storey-Fisher, MHC, Leauthaud+ (in prep)

Hyper Suprime Cam Survey (HSC)
(Subaru telesocpe 1400 sq. deg, r~26)

WGAN

$P_{HSC}(X)$

(20 < i < 20.5; 1 million)

x_i

(individual detection)

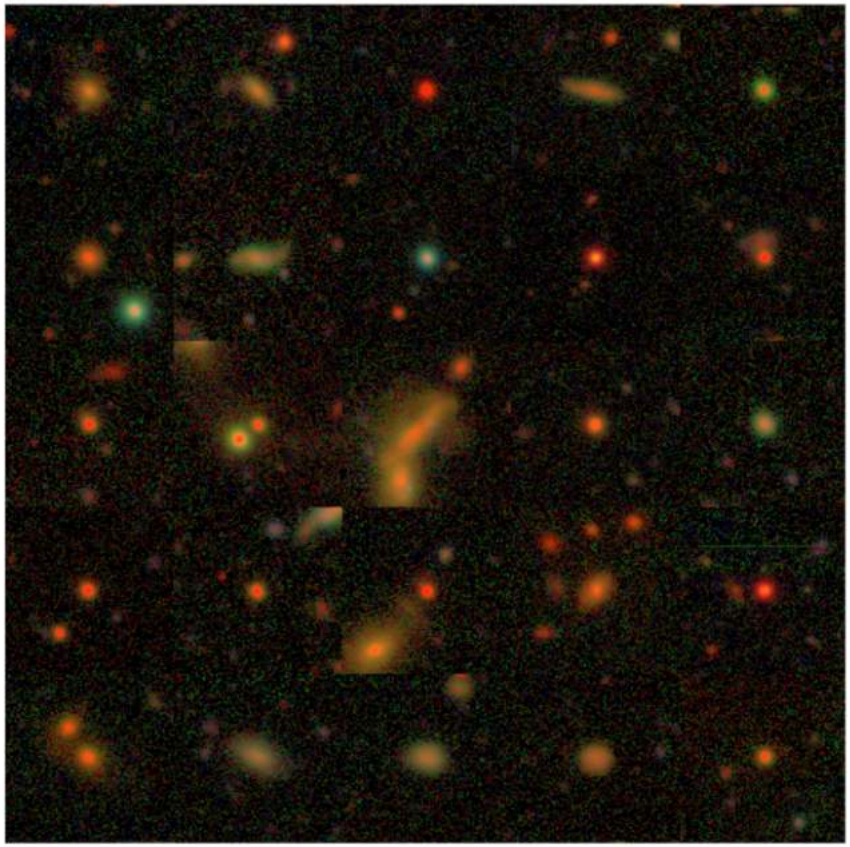
$P_{HSC}(x_i)$

identify most "unlikely" objects

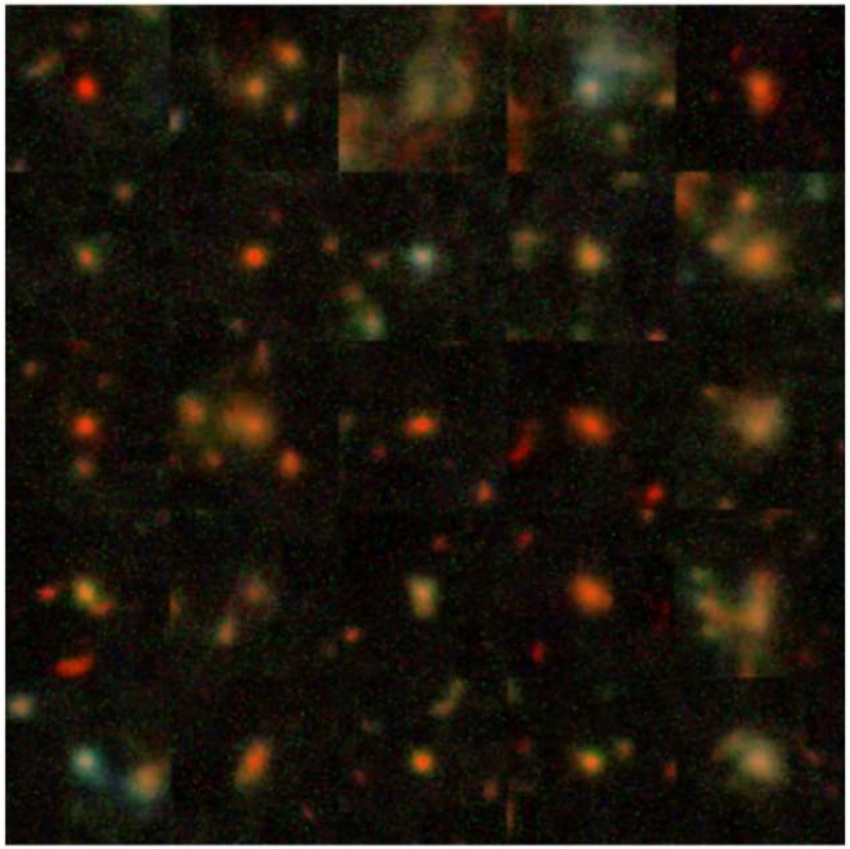
Storey-Fisher, MHC+21

Margalef-Bentabol, MHC+20

REAL



WGAN GENERATED



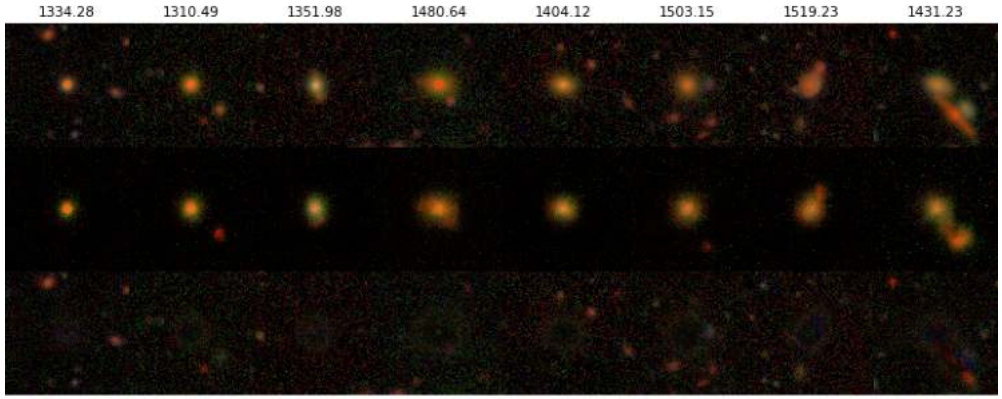
COMPUTE ANOMALY SCORE BASED ON WGAN RECONSTRUCTION ERROR

ANOMALY SCORE: $AS = \lambda G + (1 - \lambda)C$

Hyperparameter

RMS btw input and generated

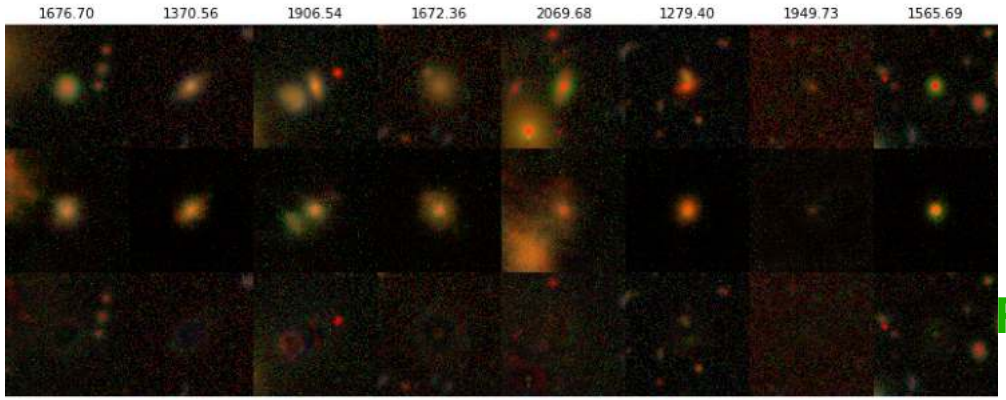
RMS btw critic features from input and generated



REAL

BEST RECONSTRUCTION

RESIDUAL



REAL

BEST RECONSTRUCTION

RESIDUAL

Schlegl+17

Storey-Fisher+21

Margalef-Bentabol, MHC+20

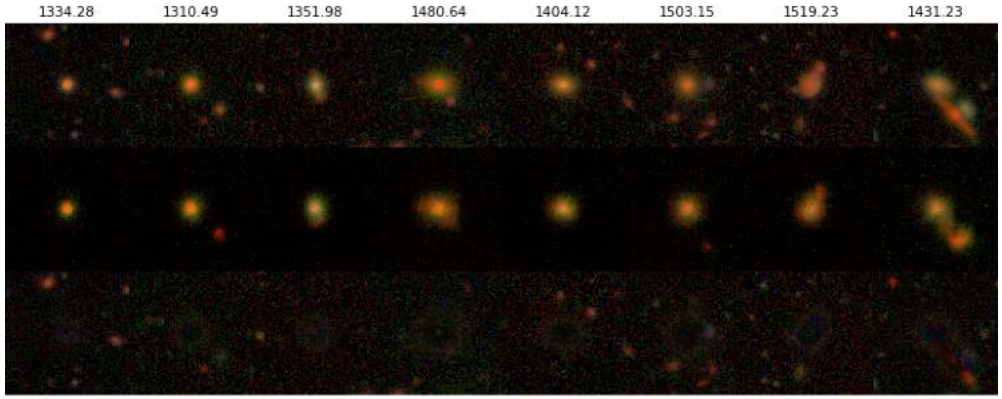
COMPUTE ANOMALY SCORE BASED ON WGAN RECONSTRUCTION ERROR

ANOMALY SCORE: $AS = \lambda G + (1 - \lambda)C$

Hyperparameter

RMS btw input and generated

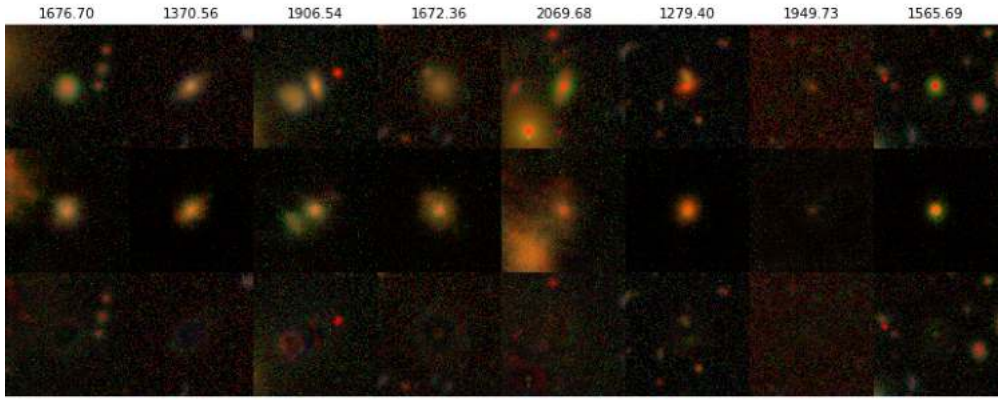
RMS btw critic features from input and generated



REAL

BEST RECONSTRUCTION

RESIDUAL



REAL

BEST RECONSTRUCTION

RESIDUAL

Schlegl+17

Storey-Fisher+20 (in prep)

Margalef-Bentabol, MHC+20

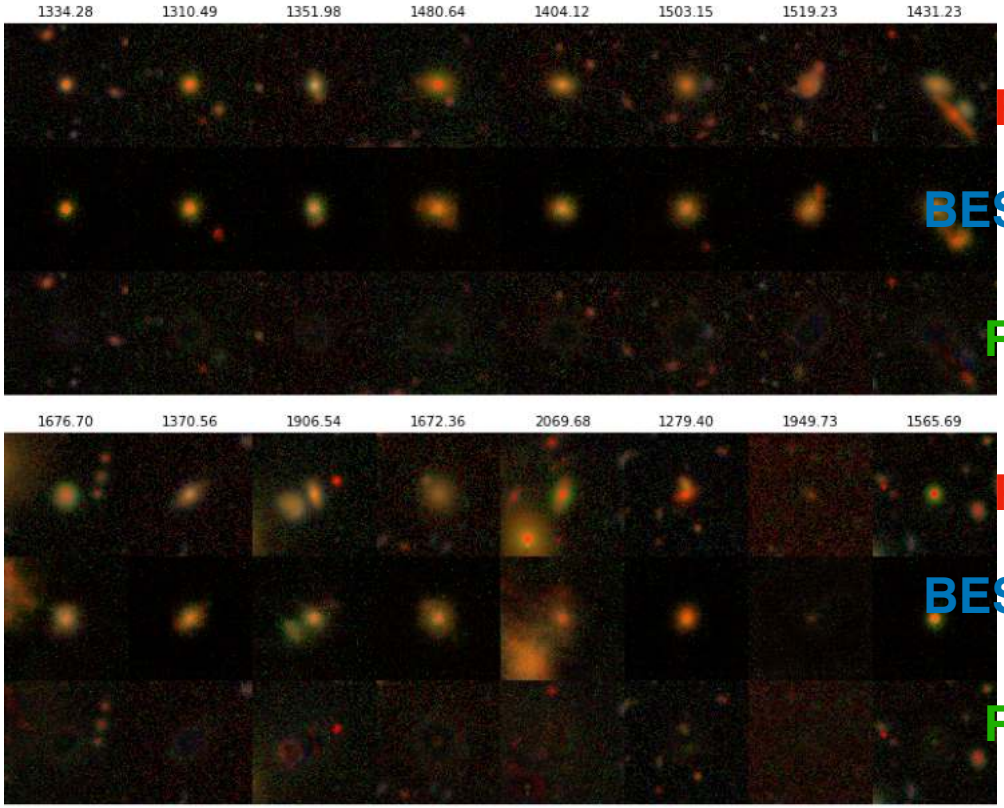
COMPUTE ANOMALY SCORE BASED ON WGAN RECONSTRUCTION ERROR

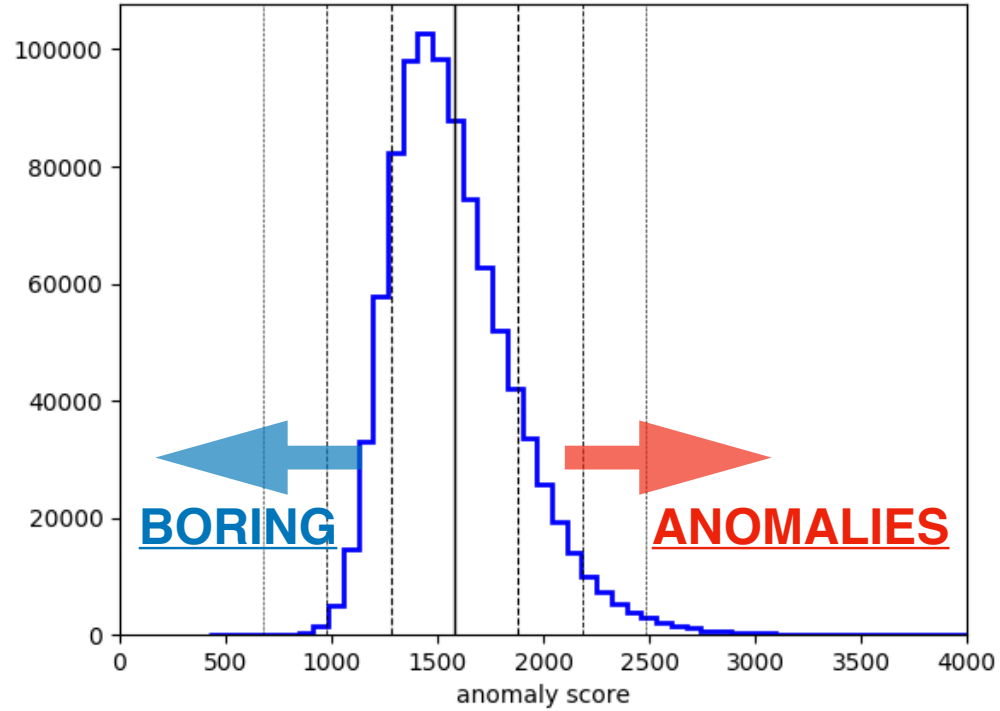
ANOMALY SCORE: $AS = \lambda G + (1 - \lambda)C$

Hyperparameter

RMS
and

RMS btw critic features

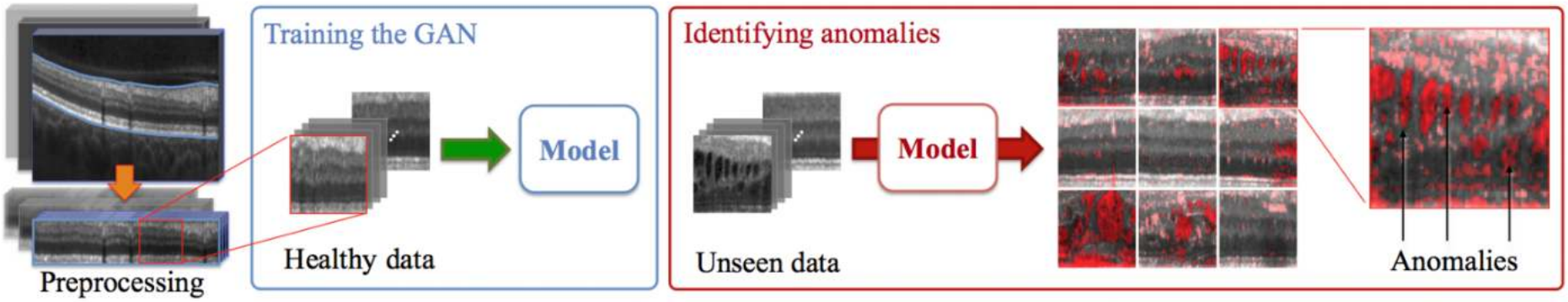


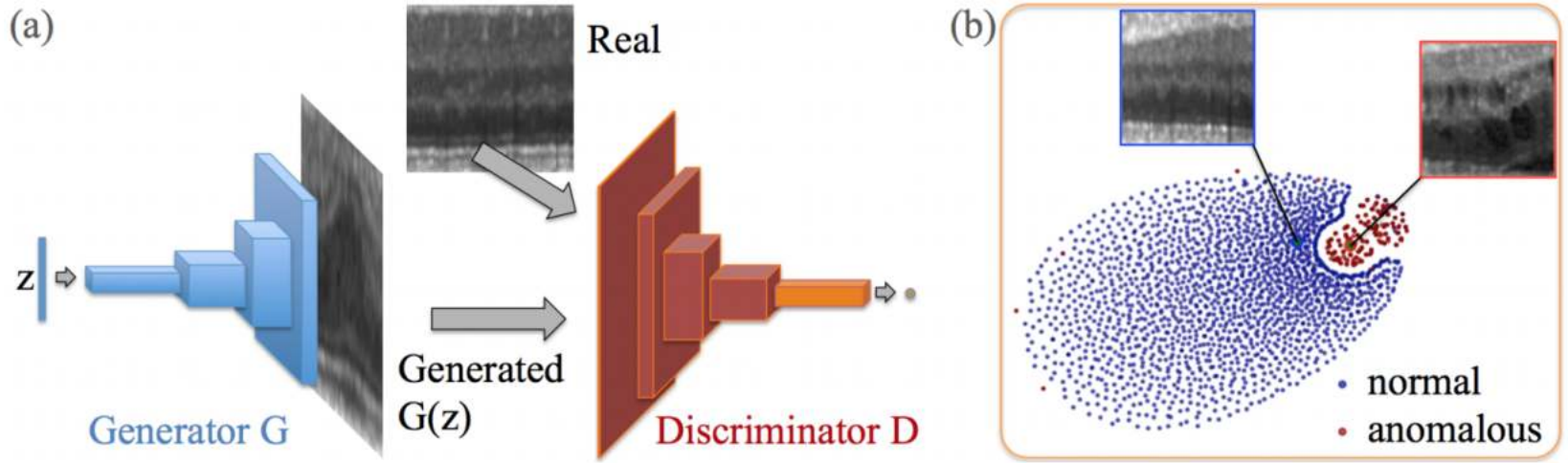


Storey-Fisher+21

Margalef-Bentabol, MHC+20

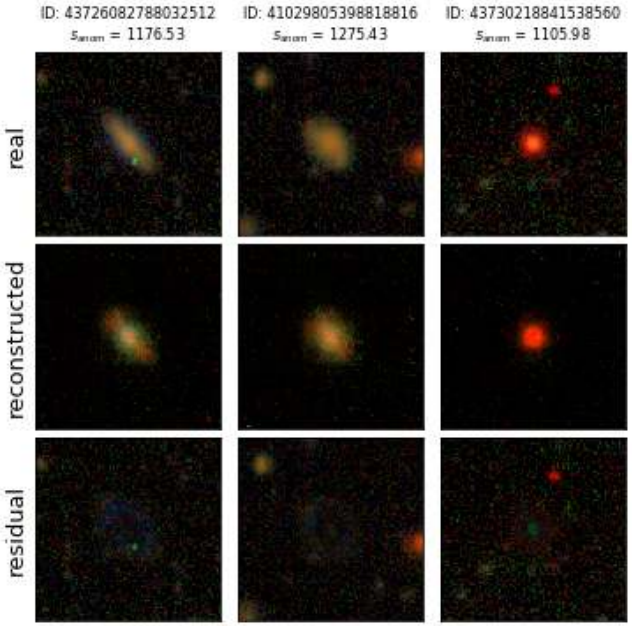
This is again inspired by biomedical applications



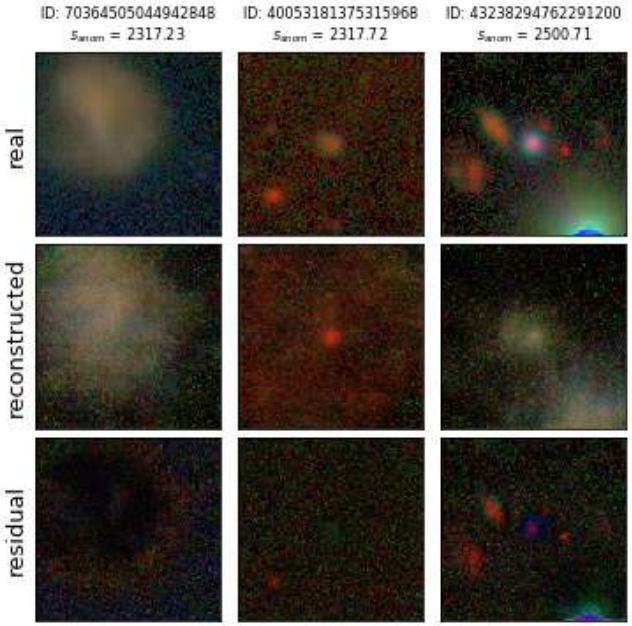


Schlegl+17

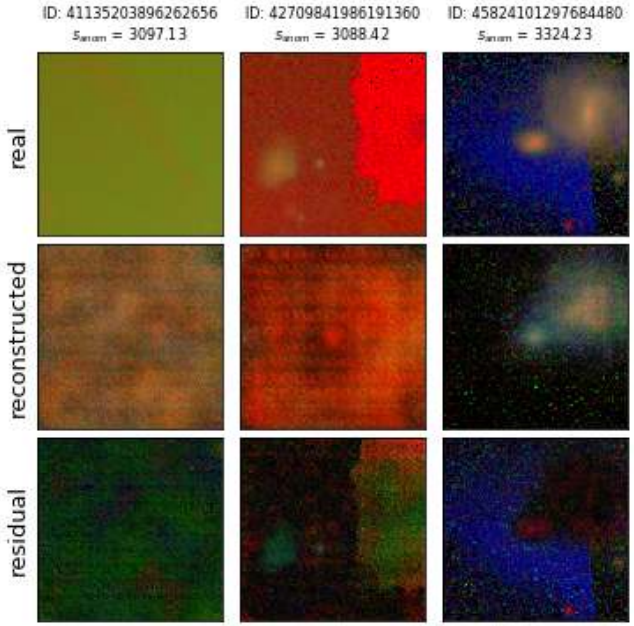
AVERAGE



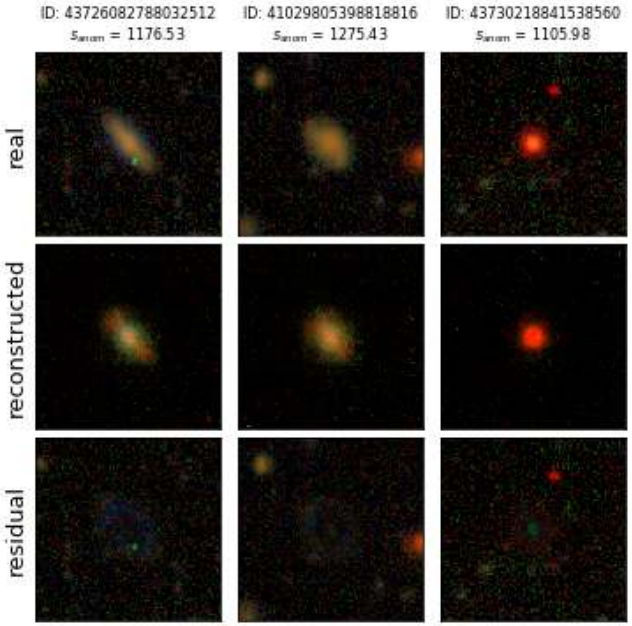
1-SIGMA



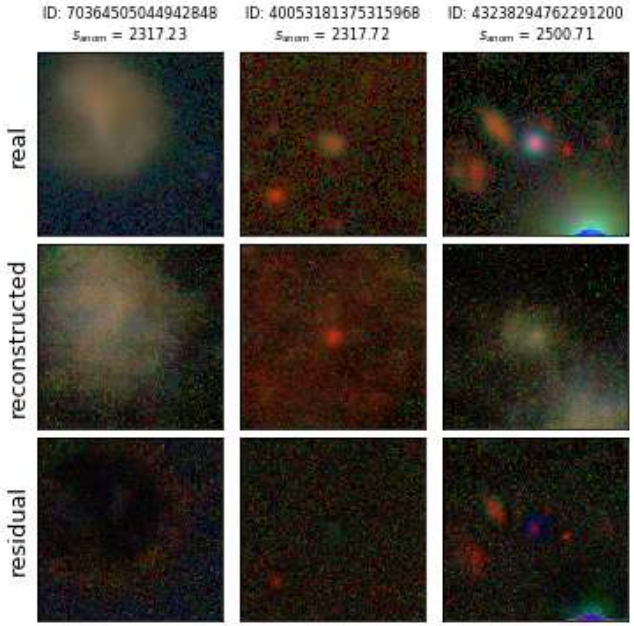
>3-SIGMA



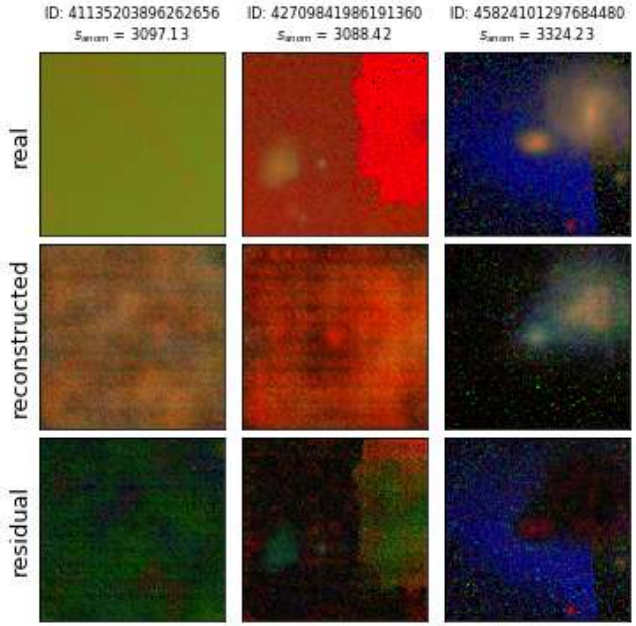
AVERAGE



1-SIGMA

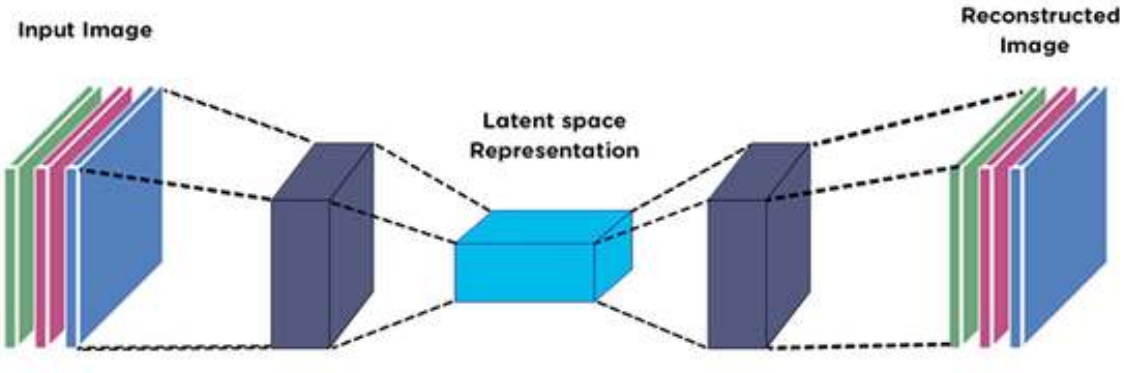


>3-SIGMA

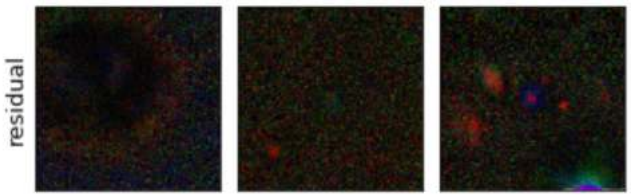


HOW DO WE FILTER OUT "INTERESTING" ANOMALIES?

HUNTING "INTERESTING" OUTLIERS



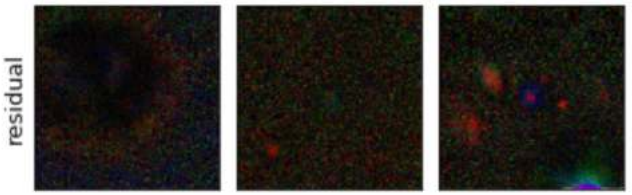
D~32



residual images

D~27,000

32 dimension space

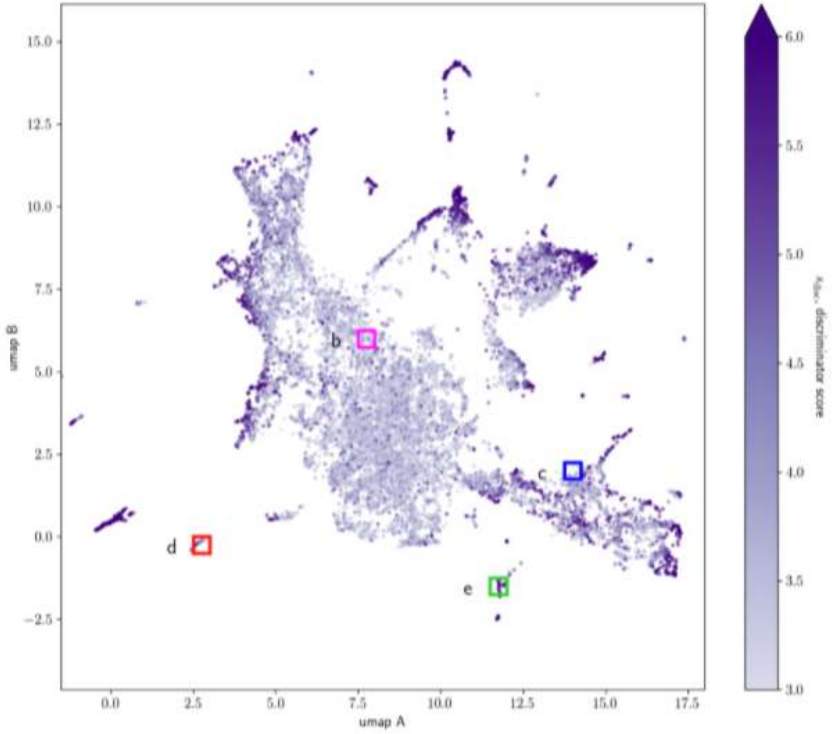


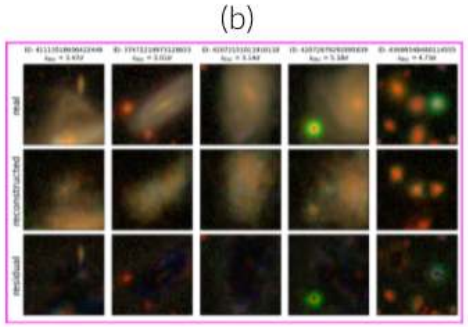
residual images

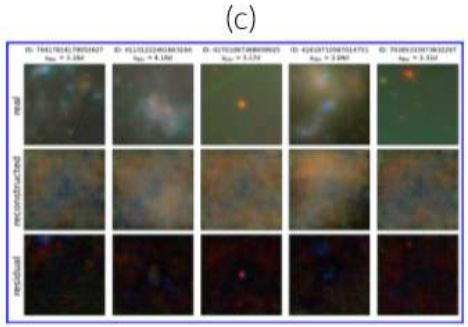
D~27,000

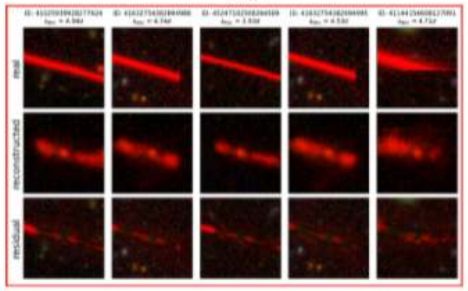
AUTO-ENCODERS TO REDUCE THE DIMENSIONALITY OF RESIDUAL IMAGES

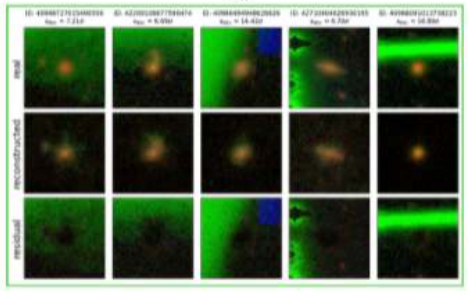
UMAP of high-anomaly sample ($s_{disc} > 3\sigma$)



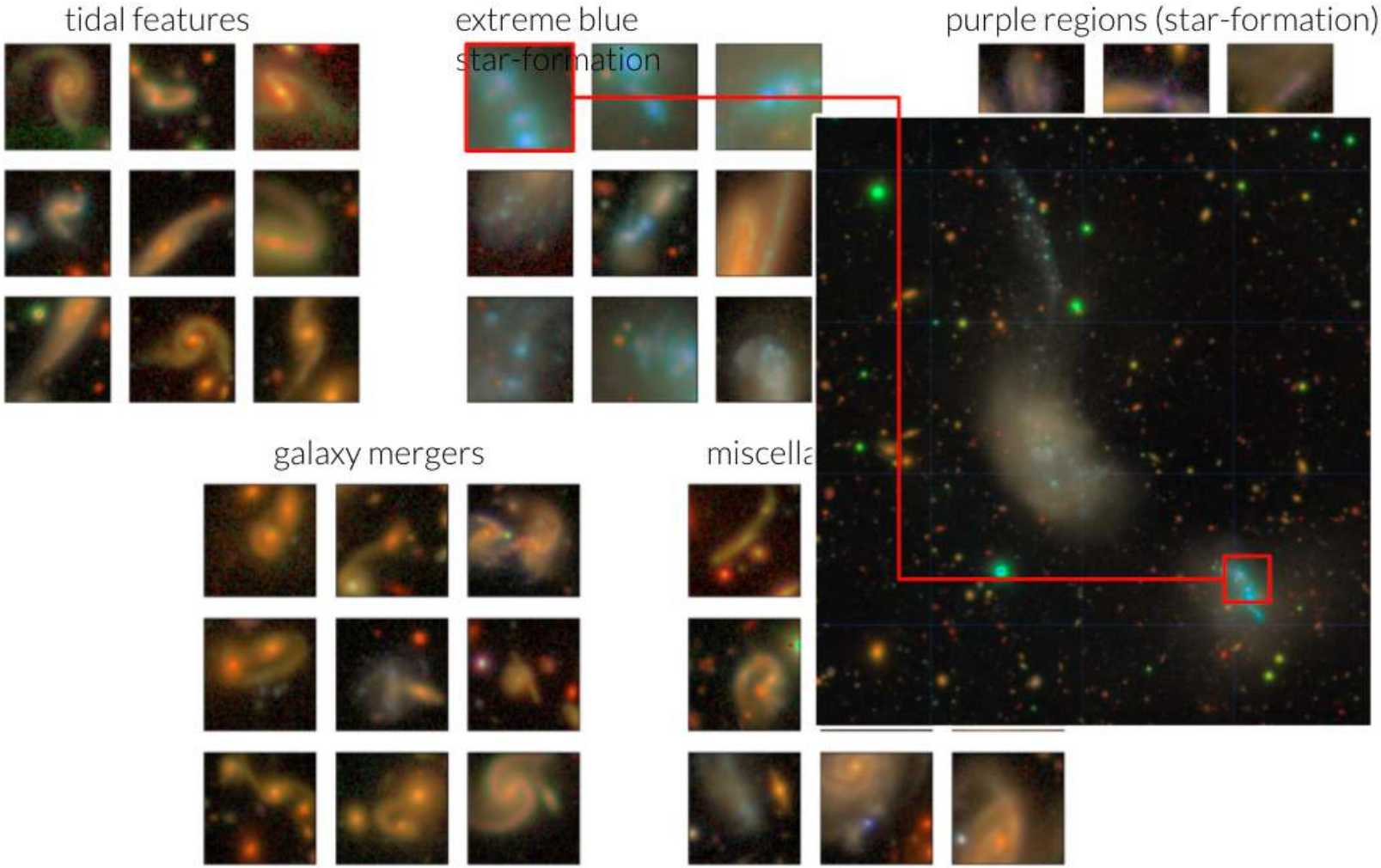








Detected interesting anomalies



3. Density Estimation for likelihood evaluation

Normalizing Flows

Based on change of variables:

$$p_X(x) = p_Z(f(x))|det J_{f(x)}|$$

unknown pdf

tractable pdf
(typically Gaussian)

Invertible differentiable
function

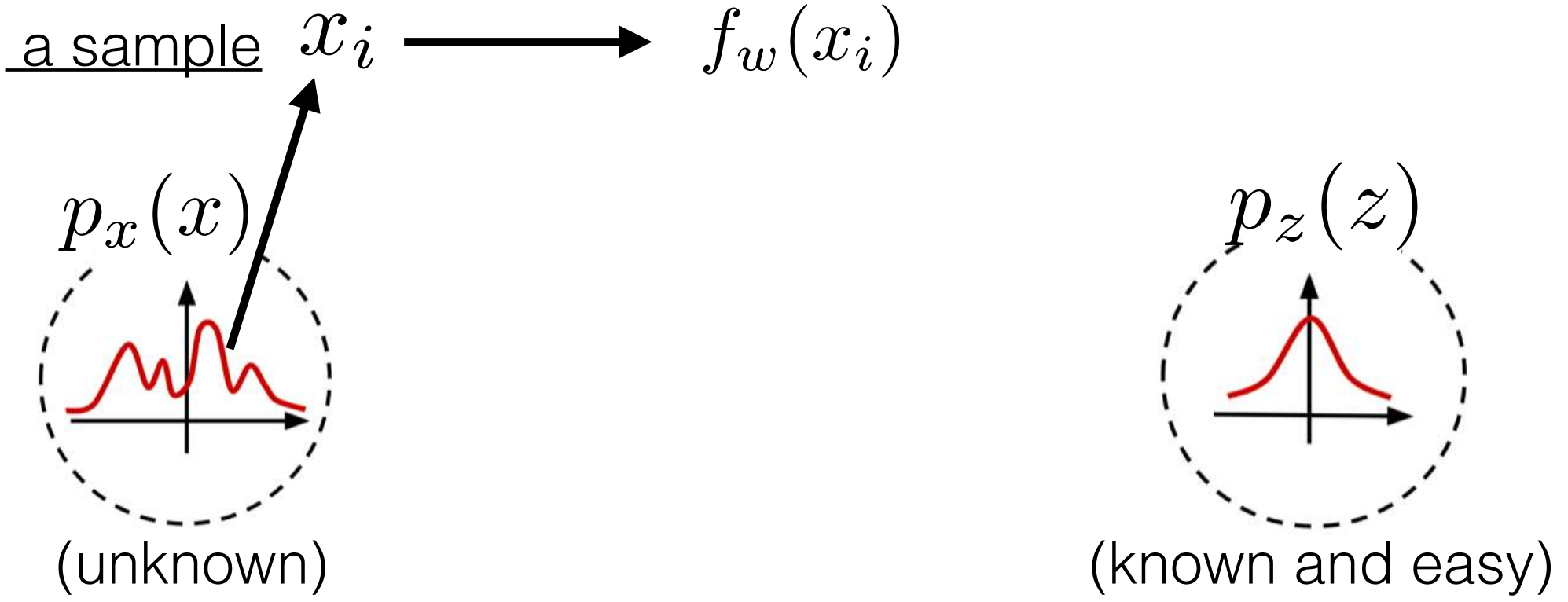
Determinant of the Jacobian
of f(x)

$$p_X(x) = p_Z(f(x))|detJ_{f(x)}|$$

Can represent px(x) in terms of f(x) and pz(x)
(notice there is no dimensionality reduction as opposed to VAEs)

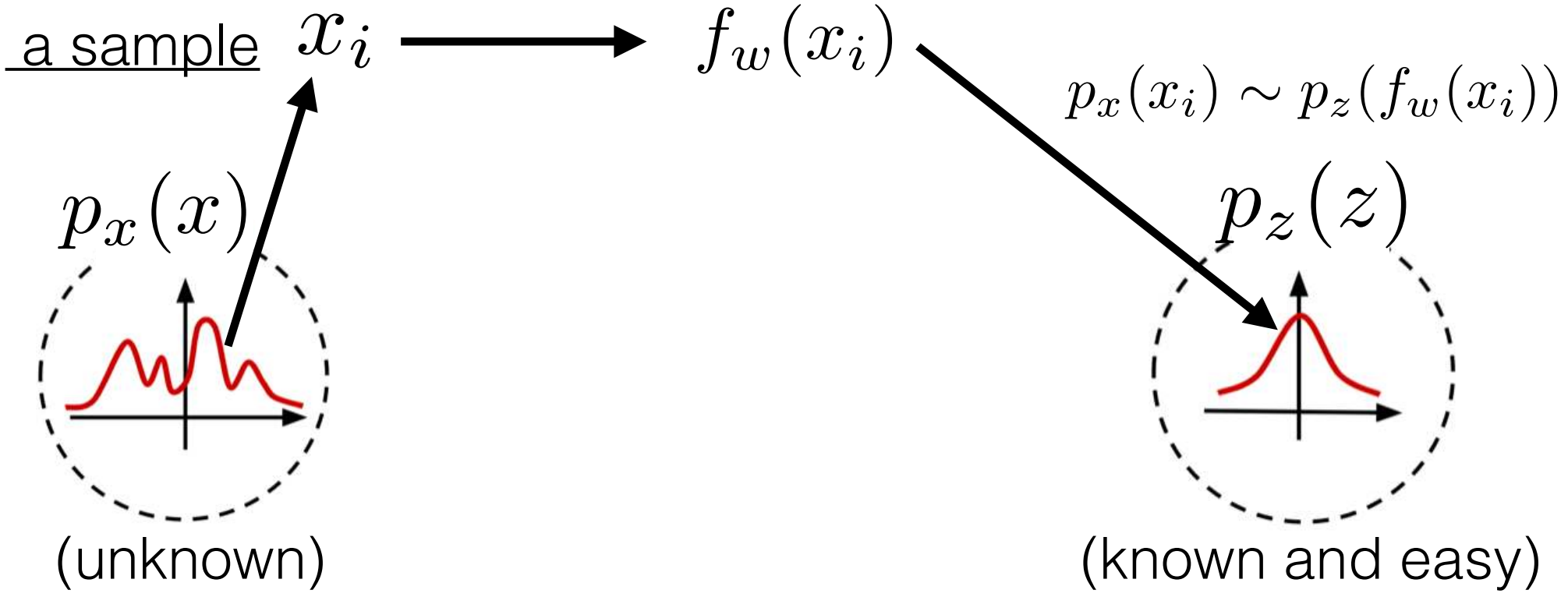
$$p_X(x) = p_Z(f(x))|detJ_{f(x)}|$$

Can represent px(x) in terms of f(x) and pz(x)
(notice there is no dimensionality reduction as opposed to VAEs)



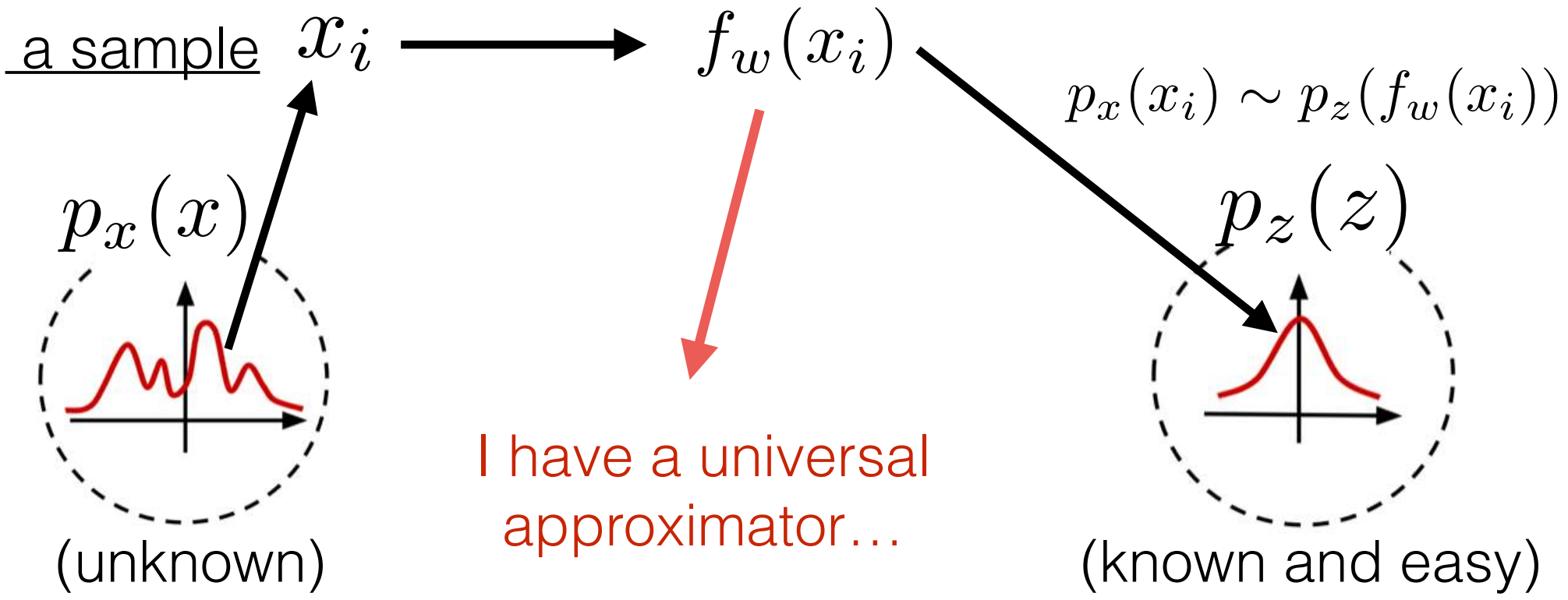
$$p_X(x) = p_Z(f(x))|detJ_{f(x)}|$$

Can represent px(x) in terms of f(x) and pz(x)
(notice there is no dimensionality reduction as opposed to VAEs)



$$p_X(x) = p_Z(f(x))|detJ_{f(x)}|$$

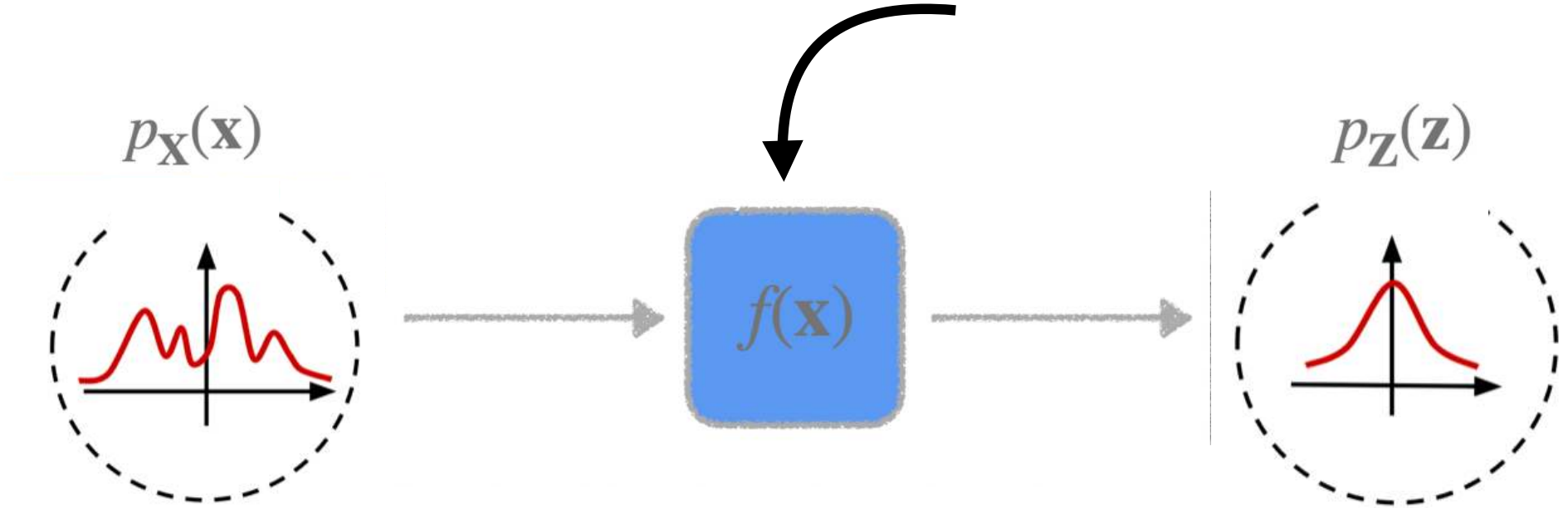
Can represent px(x) in terms of f(x) and pz(x)
(notice there is no dimensionality reduction as opposed to VAEs)



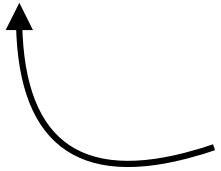
$$p_X(x) = p_Z(f(x))|detJ_{f(x)}|$$

Can represent px(x) in terms of f(x) and pz(x)
(notice there is no dimensionality reduction as opposed to VAEs)

Learn this transformation with a NN
(remember it's differentiable)



Then f(x) becomes: $f(x; W)$



NN learnable weights

Then f(x) becomes: $f(x; W)$

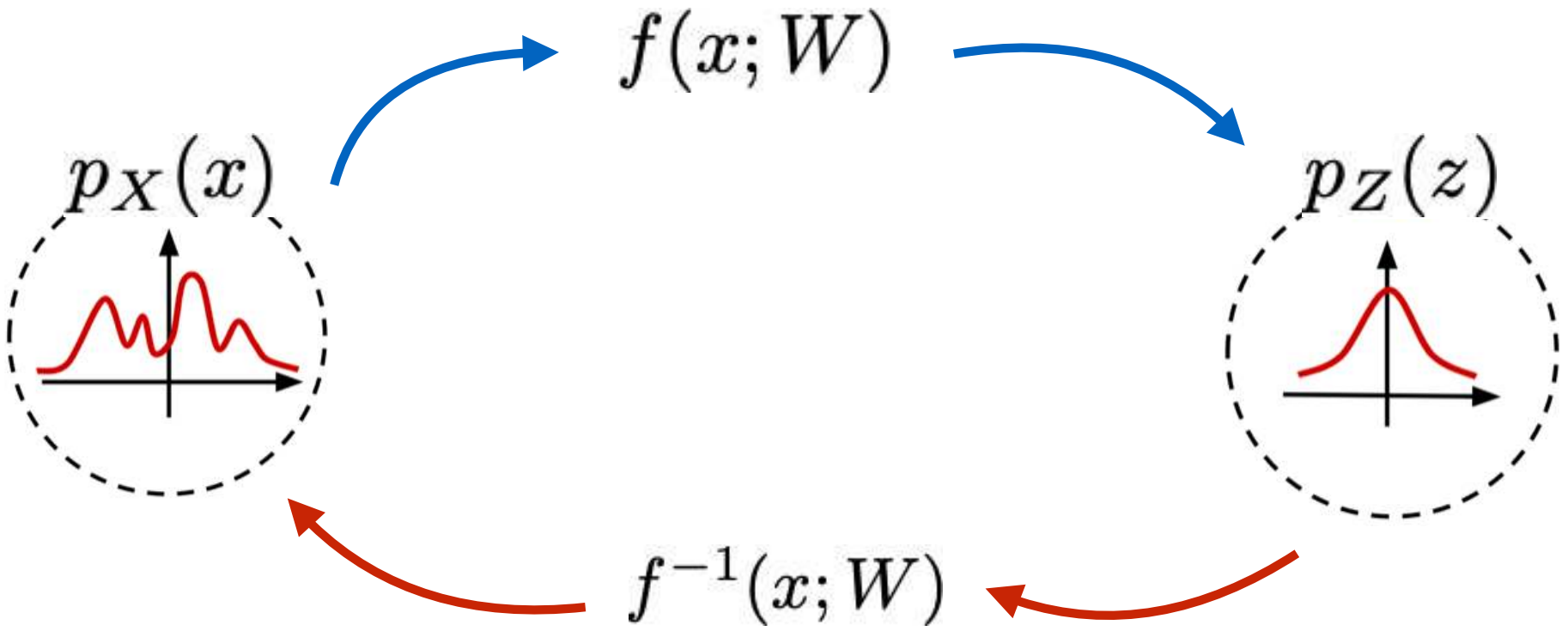
NN learnable weights

And the loss function: log likelihood

$$\sum_{i=1}^{N} log(p_Z(f(x_i; W)) + log(|detJ_{f(x_i;W)}|))$$

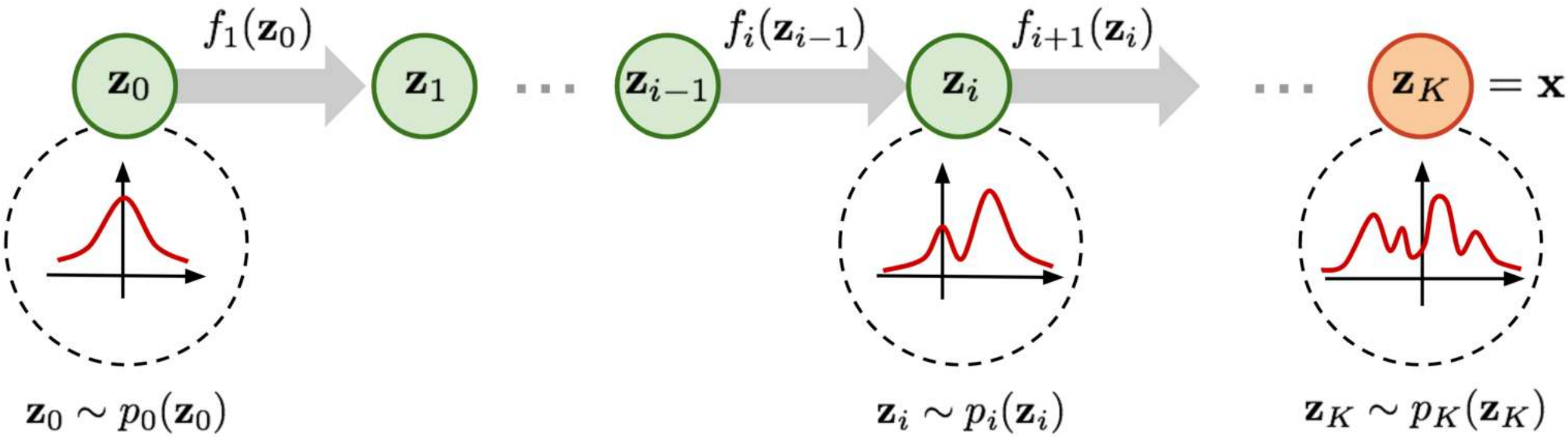
This is a gaussian

Likelihood evaluation



Sampling

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

What functions satisfy these conditions?

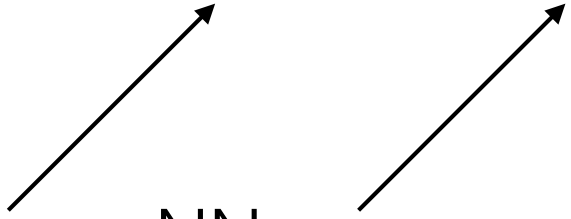
- 1. Easily invertible
- 2. Jacobian easy to compute

What functions satisfy these conditions?

- 1. Easily invertible
- 2. Jacobian easy to compute

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

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



NN

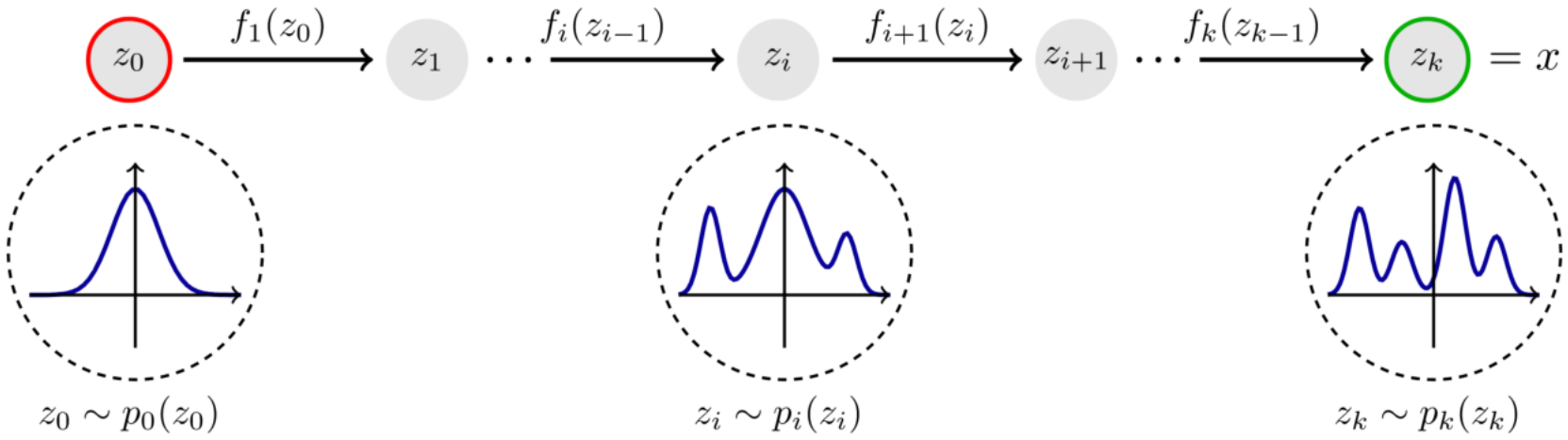
- Easily invertible:

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

- Jacobian:

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

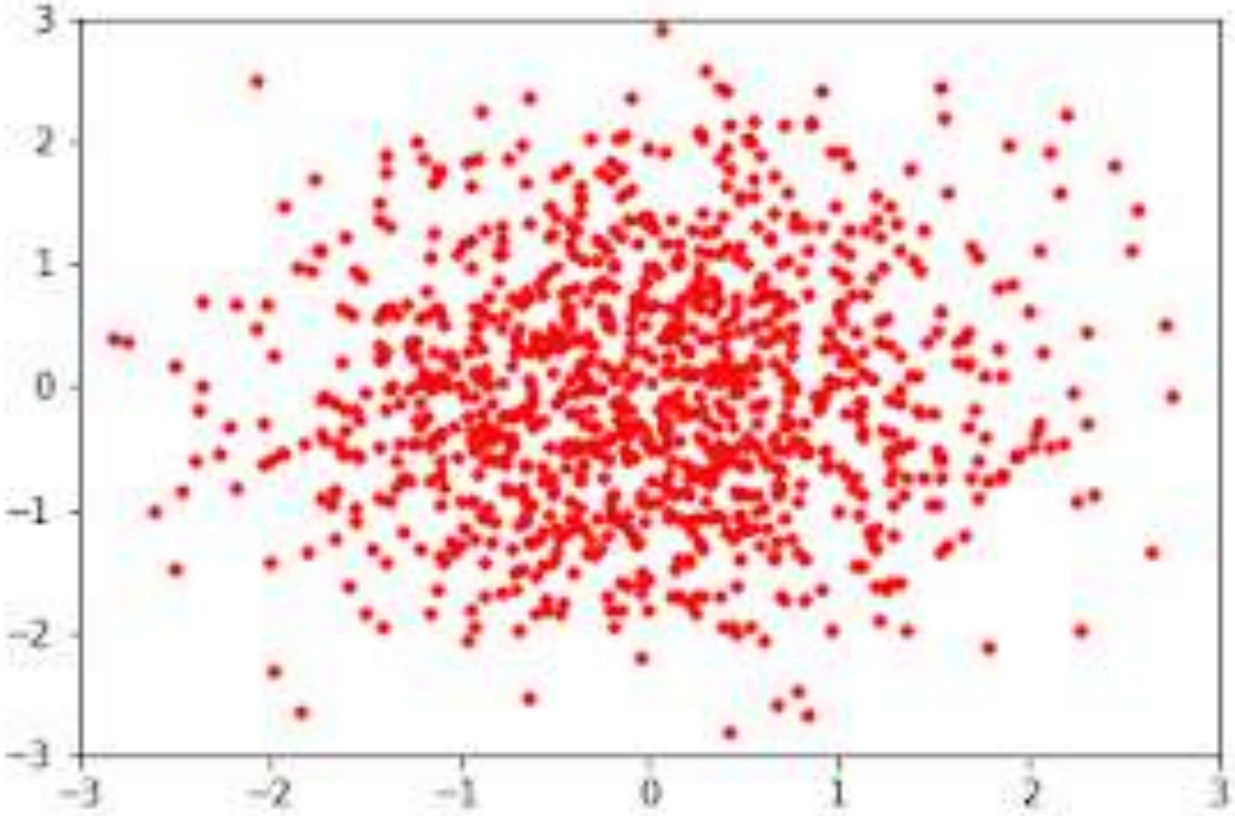
normalizing flows are generative models that are easy to evaluate and flexibly expressive



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

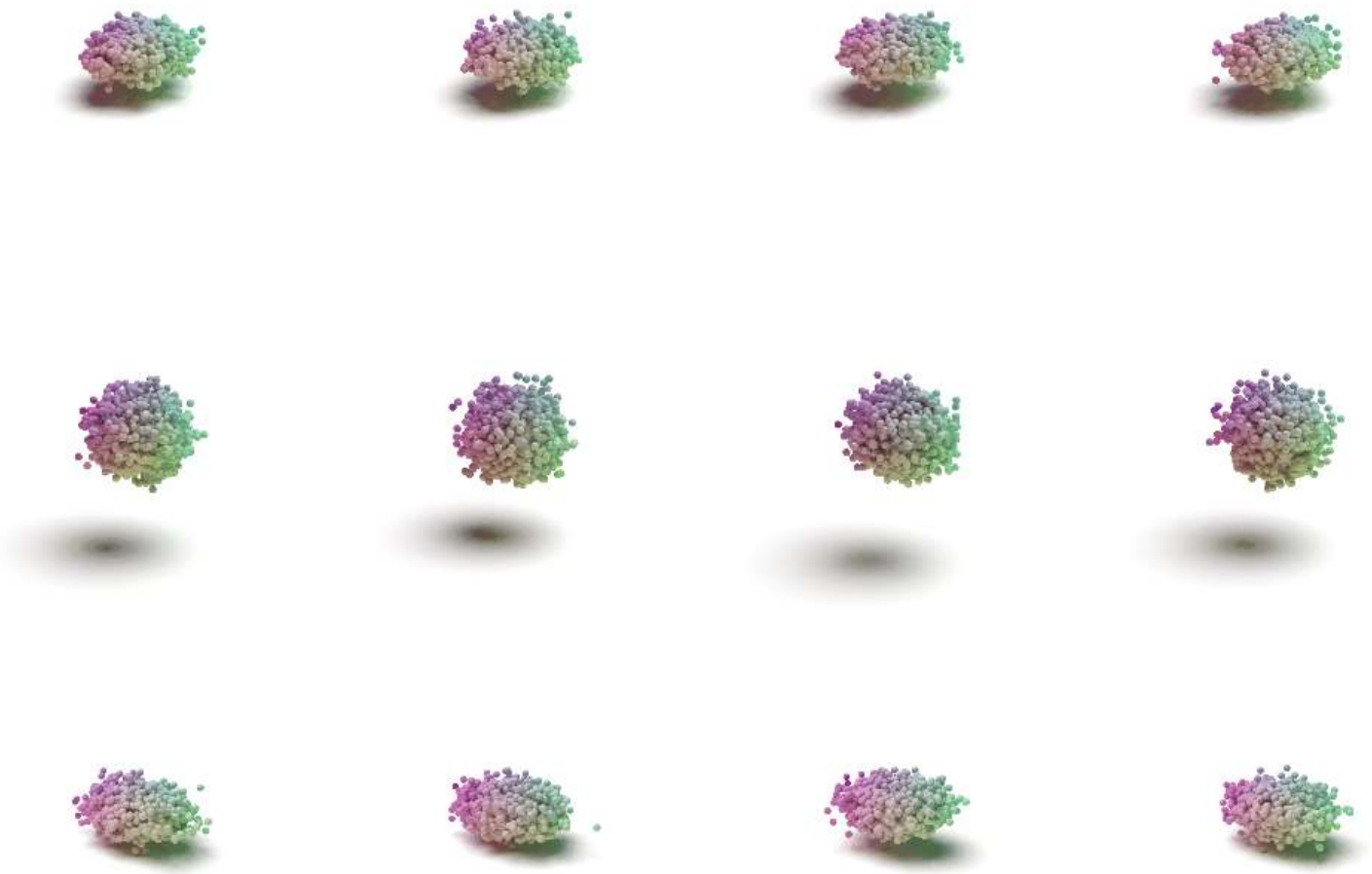
$f = f_1 \circ f_2 \ldots \circ f_{k-1} \circ f_k$ is also invertible and differentiable

normalizing flows are generative models that are easy to evaluate and flexibly expressive

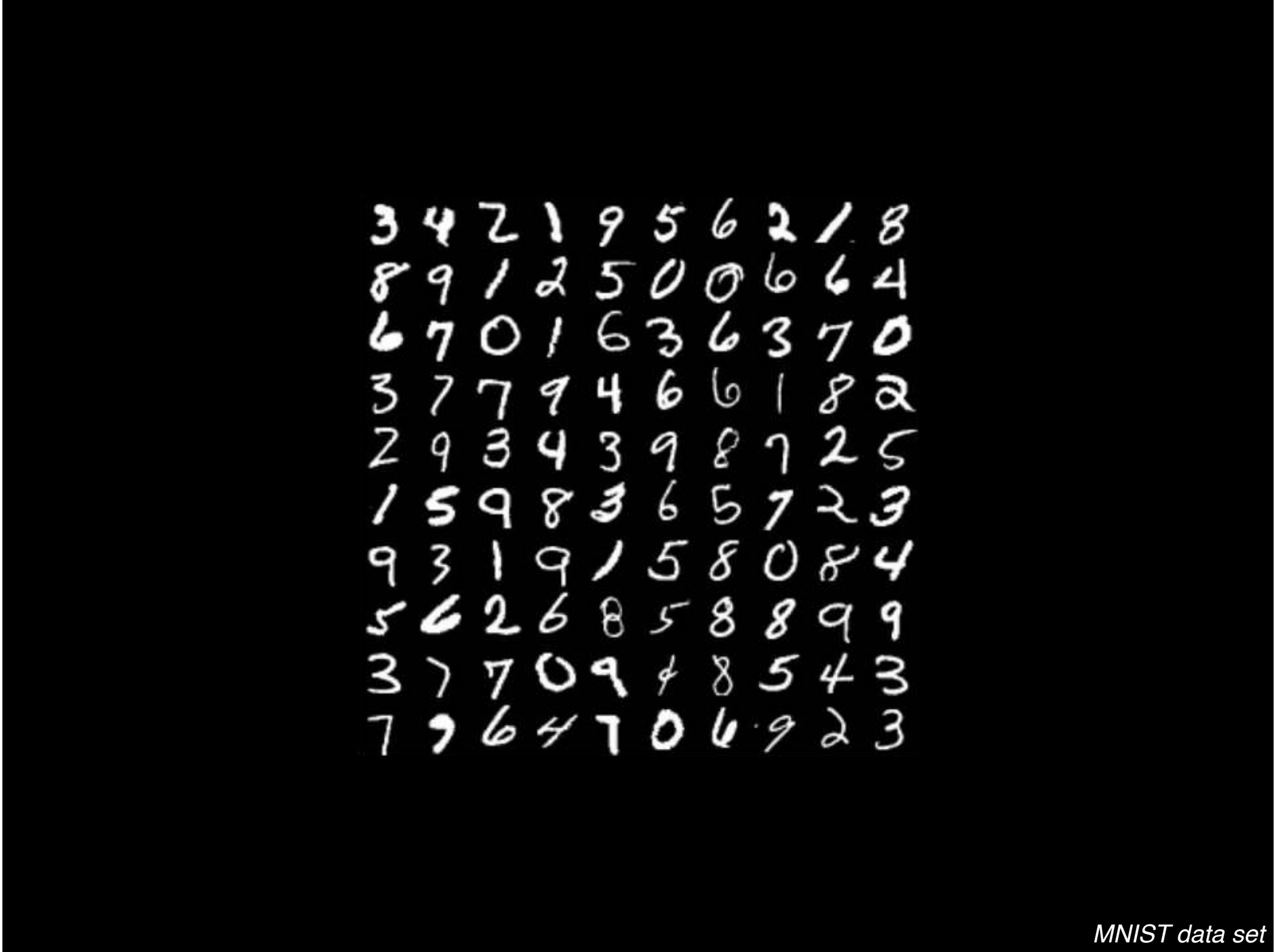


credit: Eric Jang

normalizing flows are generative models that are easy to evaluate and flexibly expressive

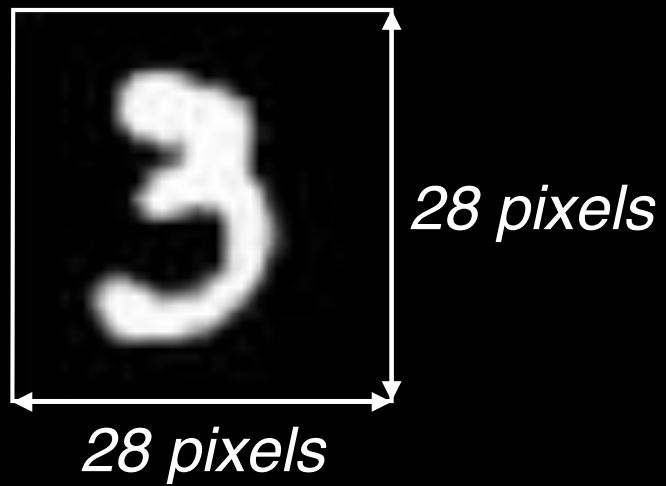


Yang+(2019) PointFlow



MNIST data set

train a generative model on MNIST — *estimate*
$p(\text{pixels}) \approx q_\phi(\text{pixels})$

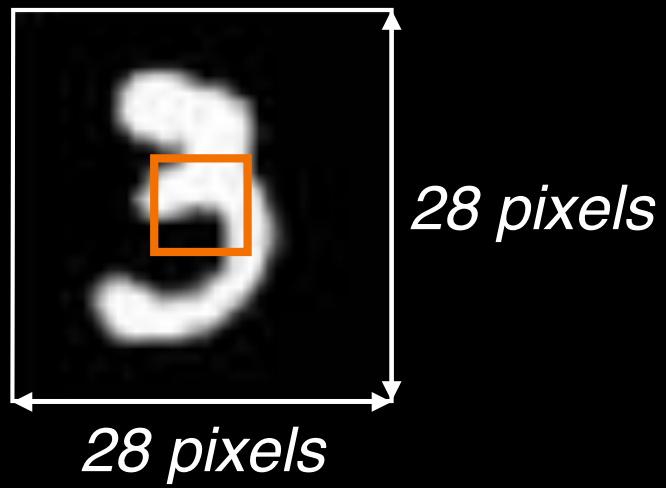


MNIST data set

sample $q_\phi(\text{pixels})$

sample q_ϕ(pixels)

train a generative model to estimate **conditional distributions**

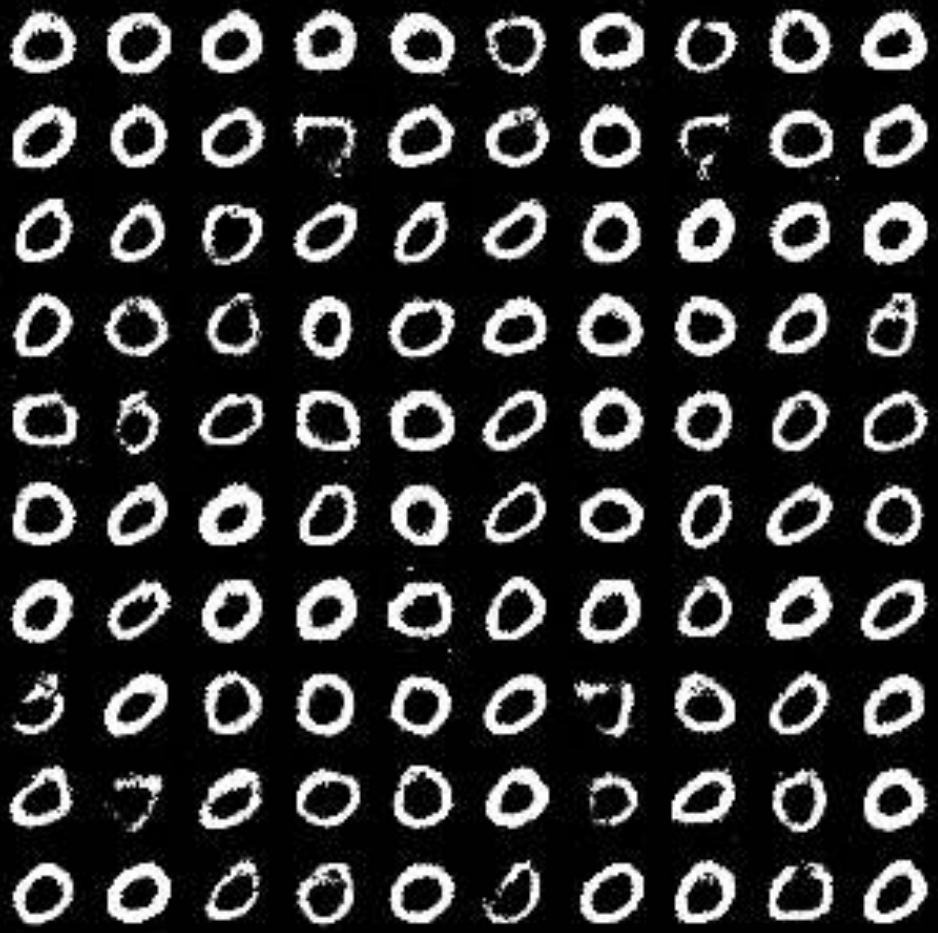


$$p(\text{pixels} \mid \text{central pixels}) \approx q_\phi(\text{pixels} \mid \text{central pixels})$$

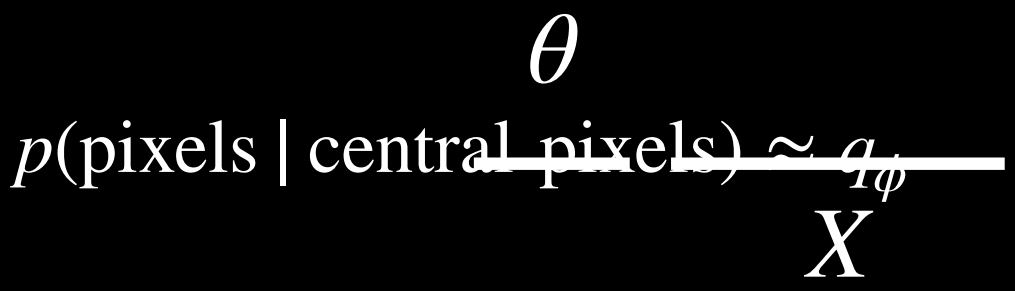
train a generative model to estimate **conditional distributions**

samples from $p(\text{pixels} \mid \text{central pixels} = 0)$ *should* just be 0s

samples drawn from $q_\phi(\text{pixels} \mid \text{central pixels} = 0)$



$$p(\text{pixels} \mid \text{central pixels}) \approx q_\phi$$


θ
p(pixels | central pixels) ≃ q_φ
X

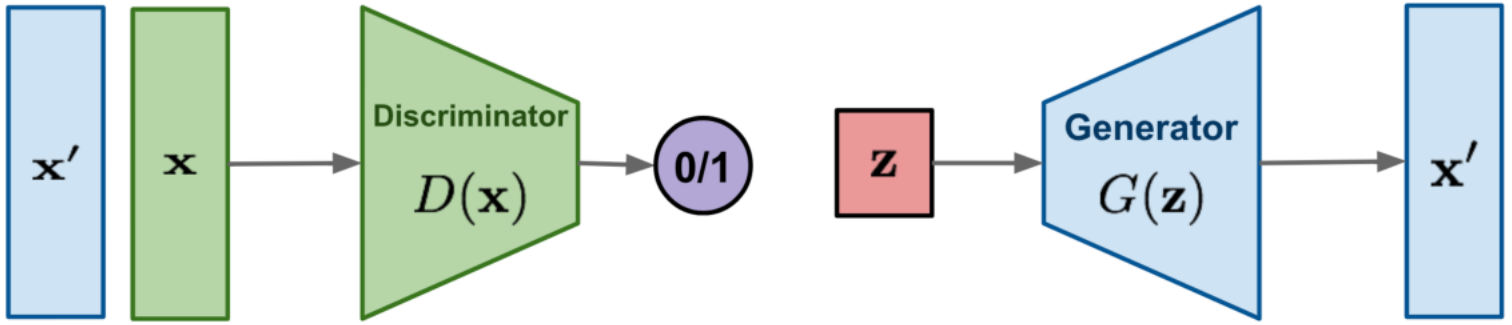
GANs AND VAEs ARE VERY POWERFUL BUT DO NOT PROVIDE AN EXPLICIT LIKELIHOOD

Method	Train on data	One-pass Sampling	Exact log-likelihood	Free-form Jacobian
Variational Autoencoders	✓	✓	✗	✓
Generative Adversarial Nets	✓	✓	✗	✓
Likelihood-based Autoregressive	✓	✗	✓	✗

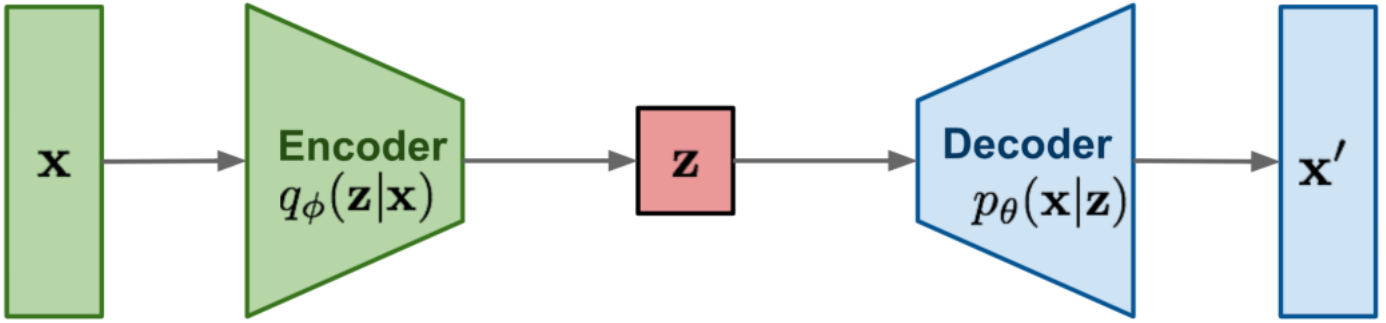
Grathwohl+18

* there is a third member of the family now: diffusion models which are as accurate as GANs and provide exact likelihood. (gradient)

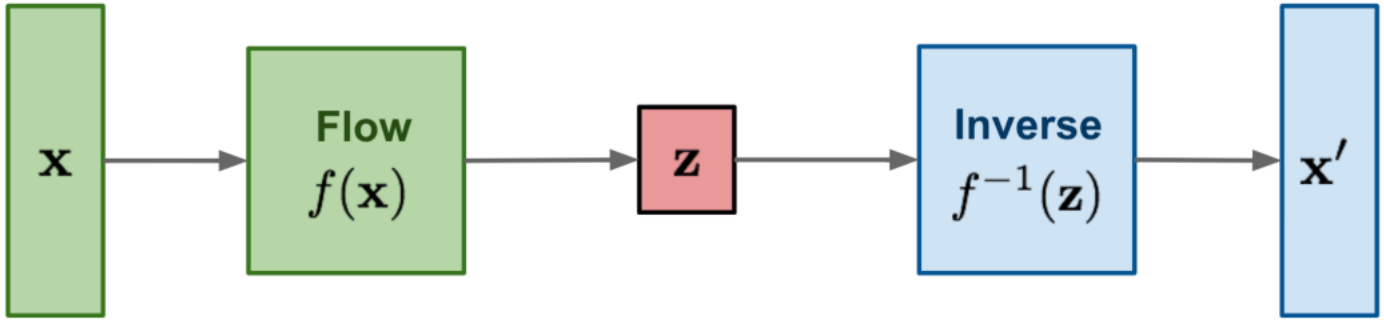
GAN: minimax the classification error loss.



VAE: maximize ELBO.



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



(+ score based diffusion models)

https://huggingface.co/spaces/lambdalabs/image-mixer-demo

“standard” bayesian inference

goal of inference

posterior *likelihood* *prior*

$$p(\theta \mid X) = \frac{p(X \mid \theta)\, p(\theta)}{p(X)}$$

$$\ln p(X \mid \theta) = \ln \mathscr{L} = \left(X - m(\theta)\right)^{T} \mathbf{C}^{-1} \left(X - m(\theta)\right)$$

covariance matrix

model

“standard” bayesian inference

$$\ln p(\theta \mid X) = \ln p(X \mid \theta) + \ln p(\theta) + \text{const}.$$

sample $p(\theta \mid X)$ using montecarlo sampling to estimate posterior
(*e.g. MCMC, HMC, nested sampling)*

Gaussian likelihood is often **an incorrect assumption (not enough data)**

references: Hogg+(2010), Hogg (2012)

Gaussian likelihood is often **an incorrect assumption (not enough data)**

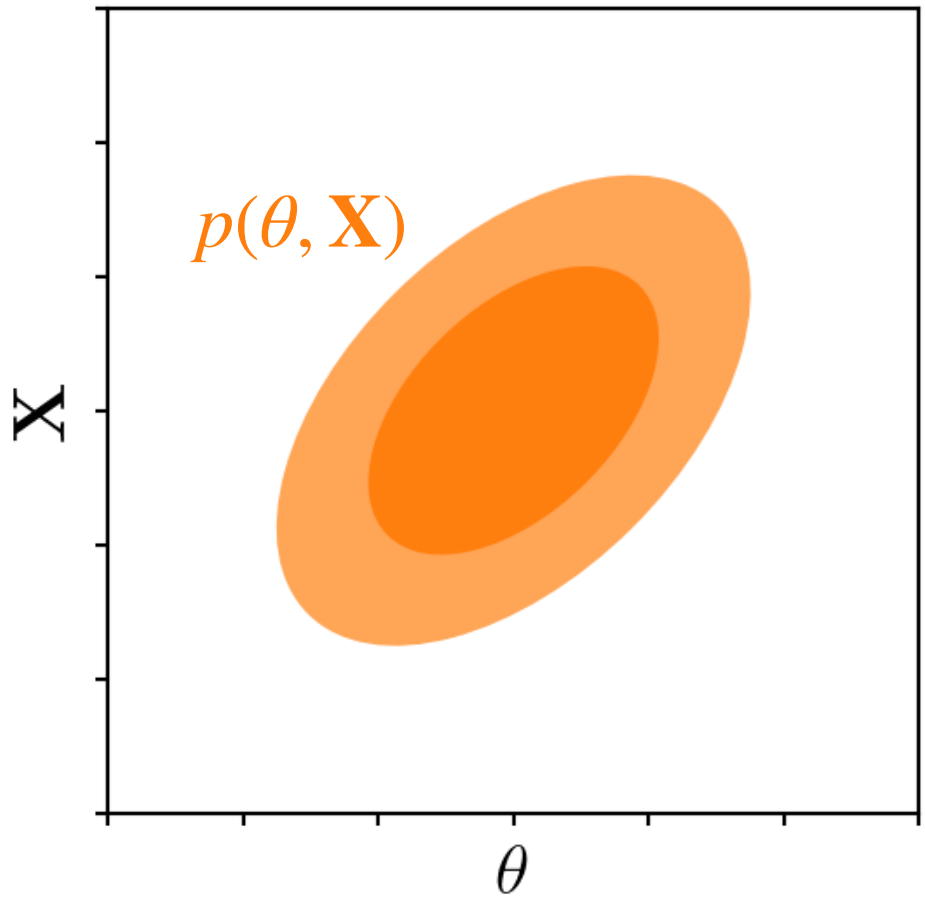
$$\ln p(X \mid \theta) = \ln \mathscr{L} \neq \left(X - m(\theta)\right)^T \mathbf{C}^{-1} \left(X - m(\theta)\right)$$

Sometimes the likelihood is intractable. e.g. Baryonic properties of DM haloes?

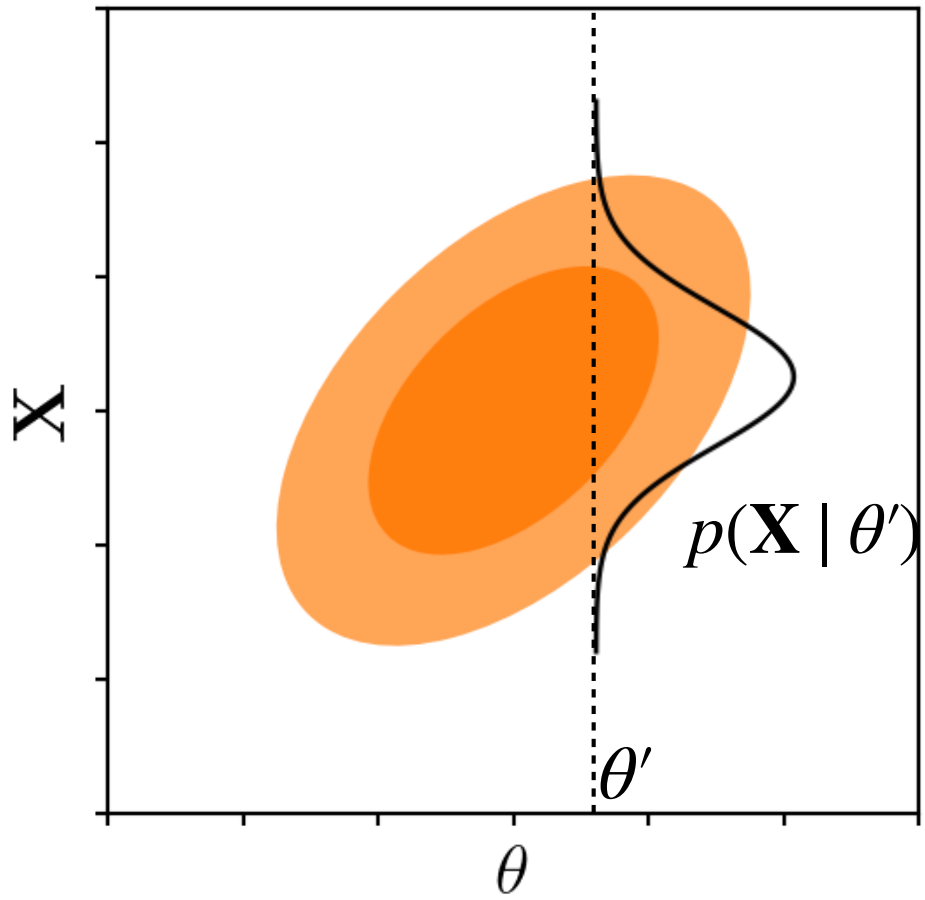
MCMC is slow! amortized inference to infer *billions* of observations

references: Hogg+(2010), Hogg (2012)

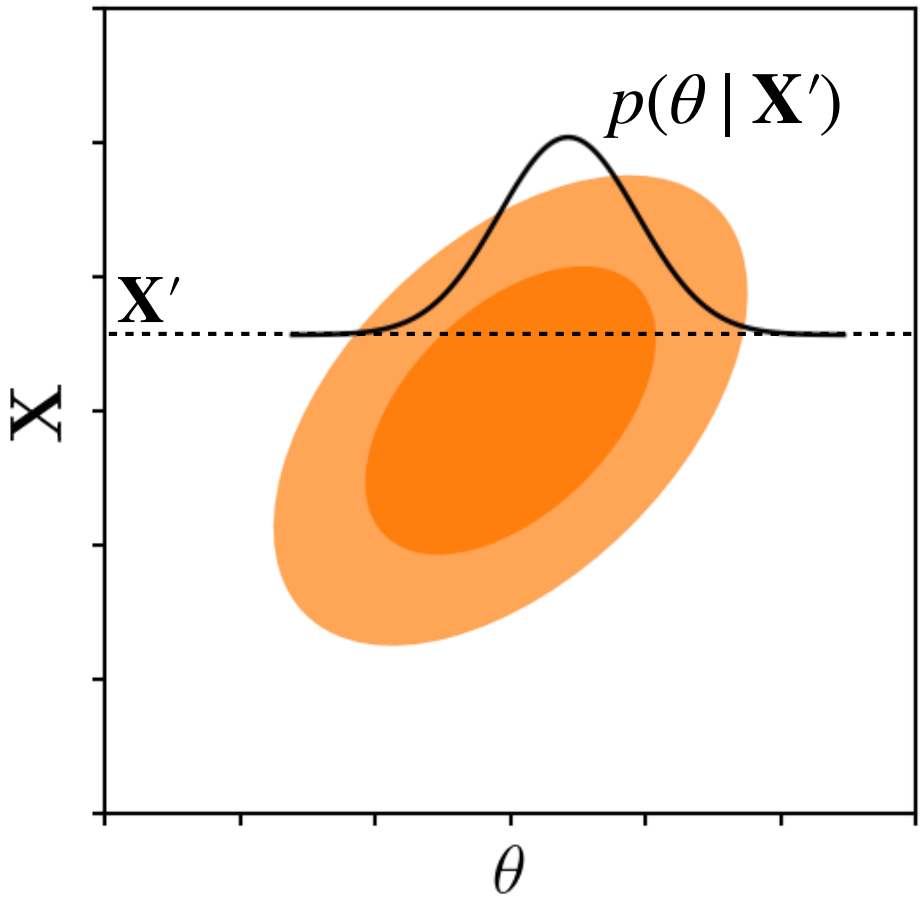
consider an experiment with *1D observation* $\mathbf{X}$ and *1D parameter* θ



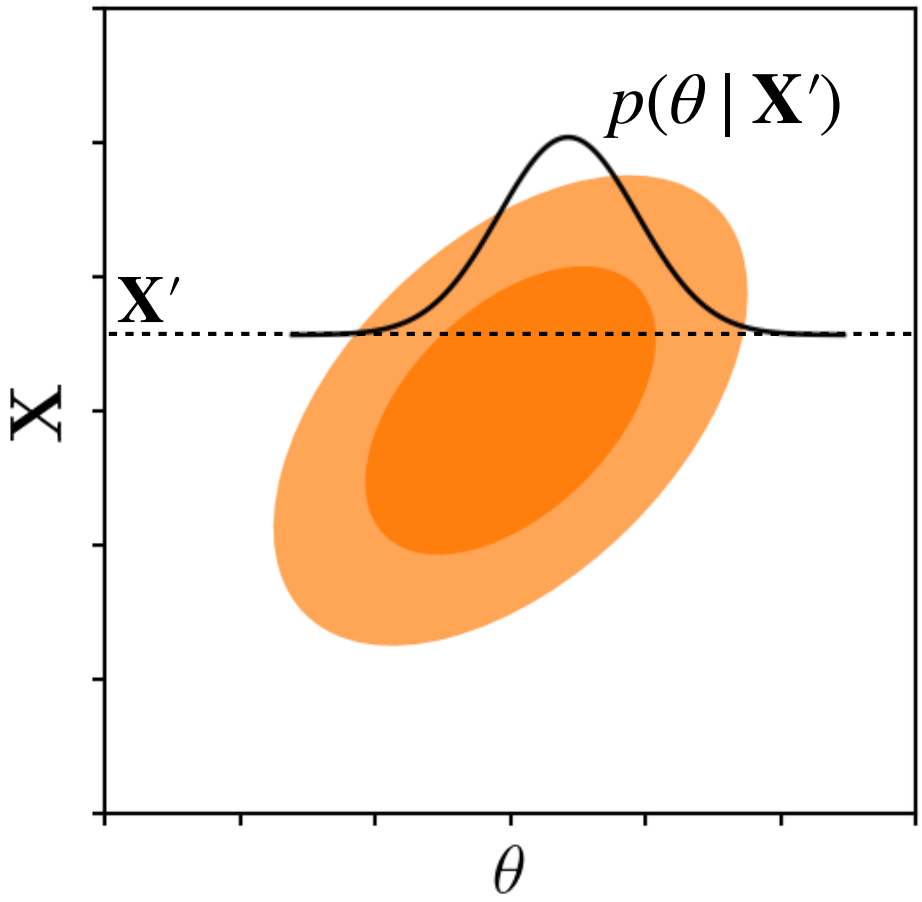
$p(\theta, \mathbf{X})$ includes both the **likelihood** $p(\mathbf{X} \mid \theta')$ and posterior $p(\theta \mid \mathbf{X}')$



$p(\theta, \mathbf{X})$ includes both the likelihood $p(\mathbf{X} \mid \theta')$ and **posterior** $p(\theta \mid \mathbf{X}')$



$p(\theta, \mathbf{X})$ includes both the likelihood $p(\mathbf{X} \mid \theta')$ and **posterior** $p(\theta \mid \mathbf{X}')$

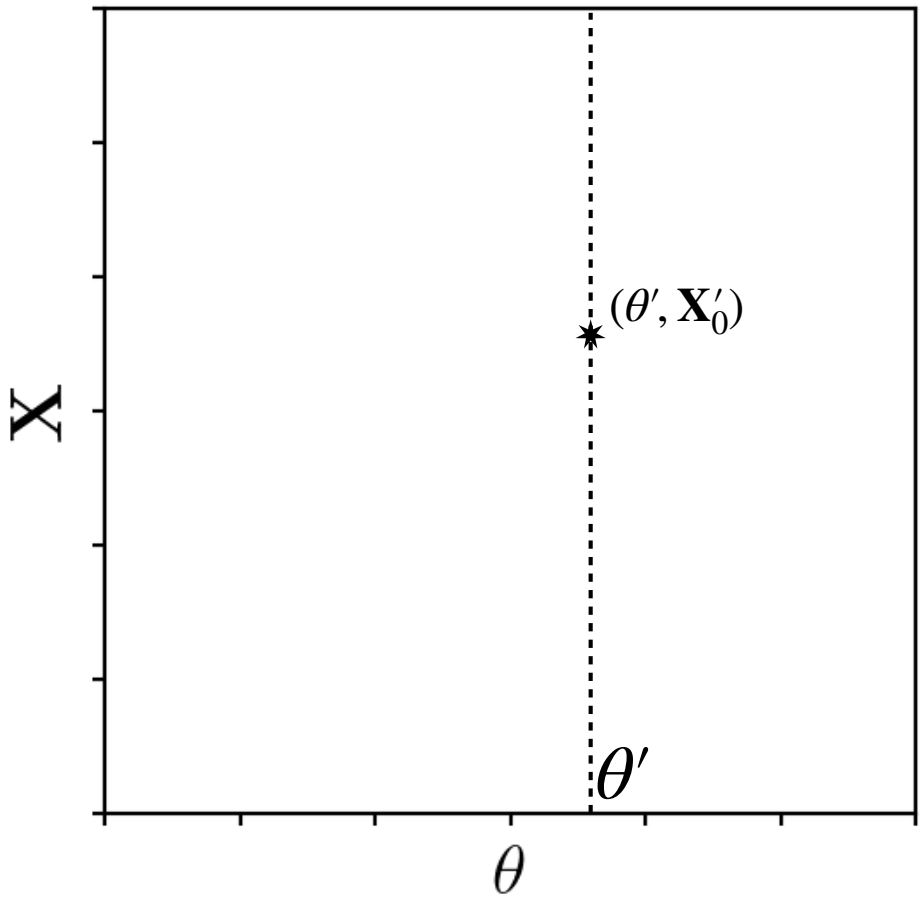


$$p(\theta \mid X) = \frac{p(X \mid \theta)\, p(\theta)}{p(X)}$$

$$p(\theta, X) = p(X \mid \theta)\, p(\theta) = p(\theta \mid X)\, p(X) \longrightarrow \text{Bayes rule}$$

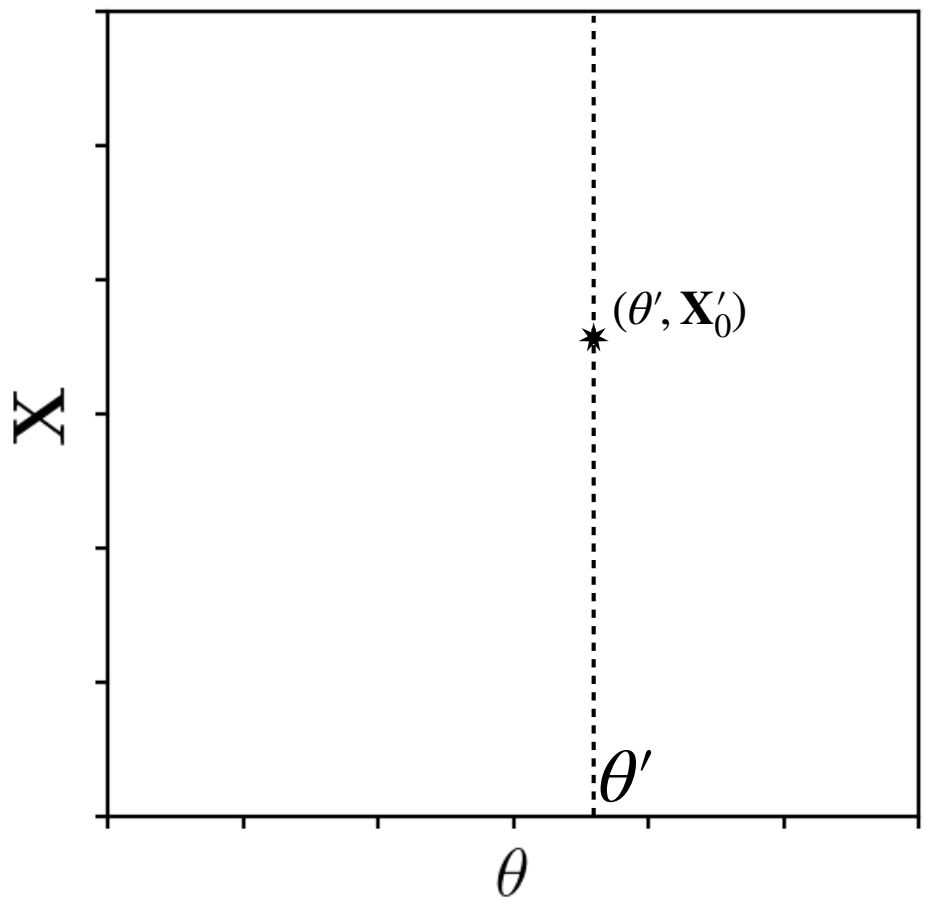
forward model F that can simulate observations:

$$\mathbf{X}'_0, \mathbf{X}'_1 \ldots \sim F(\theta')$$



forward model F that can simulate observations:

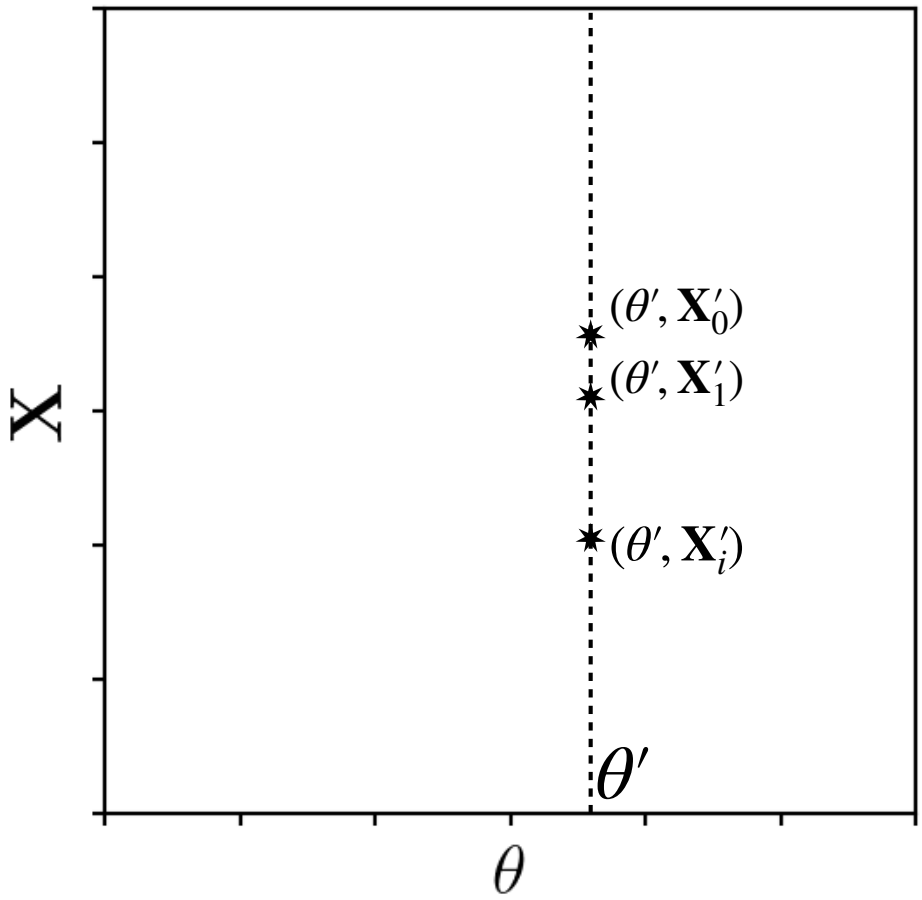
$$\mathbf{X}'_0, \mathbf{X}'_1 \ldots \sim F(\theta')$$



F must **include noise model** and **observational systematics**

forward model F that can simulate observations:

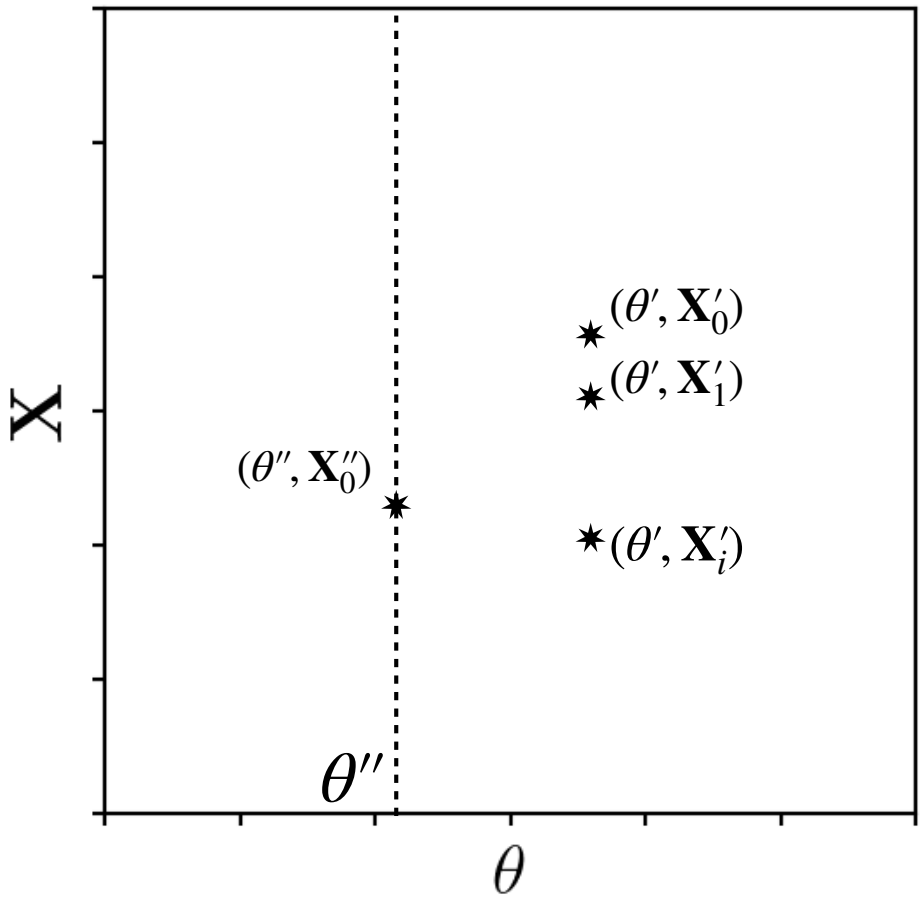
$$\mathbf{X}'_0, \mathbf{X}'_1 \ldots \sim F(\theta')$$



F must **include noise model**

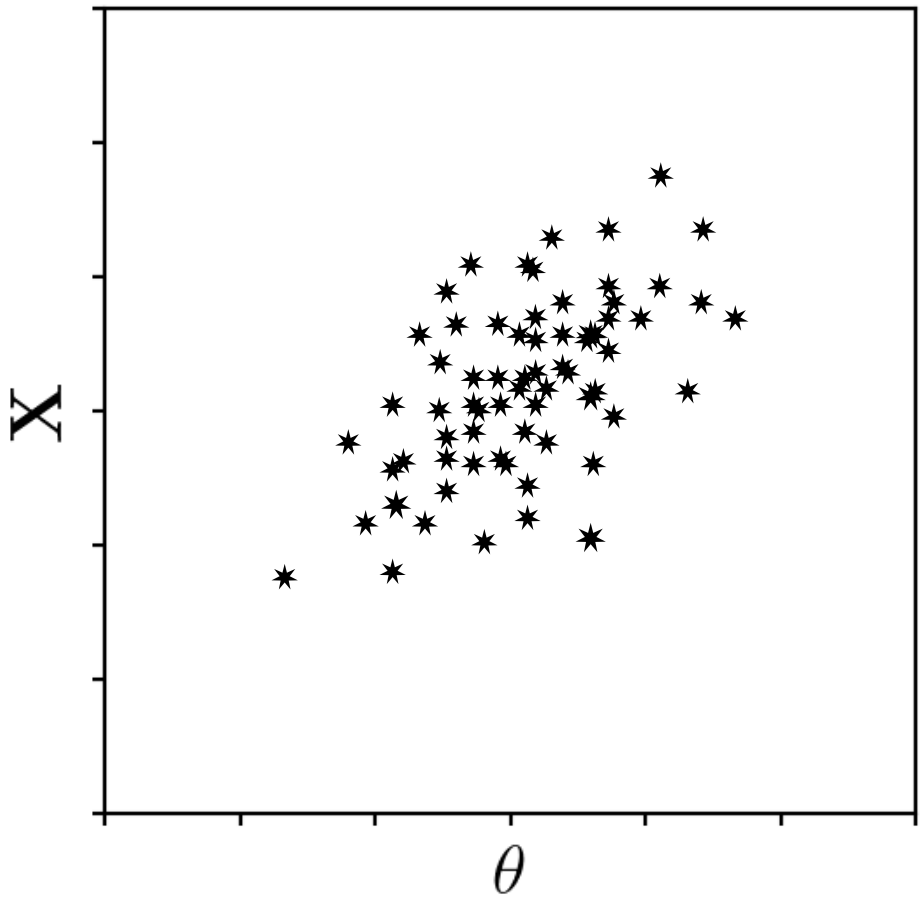
forward model F that can simulate observations:

$$\mathbf{X}'_0, \mathbf{X}'_1 \ldots \sim F(\theta')$$



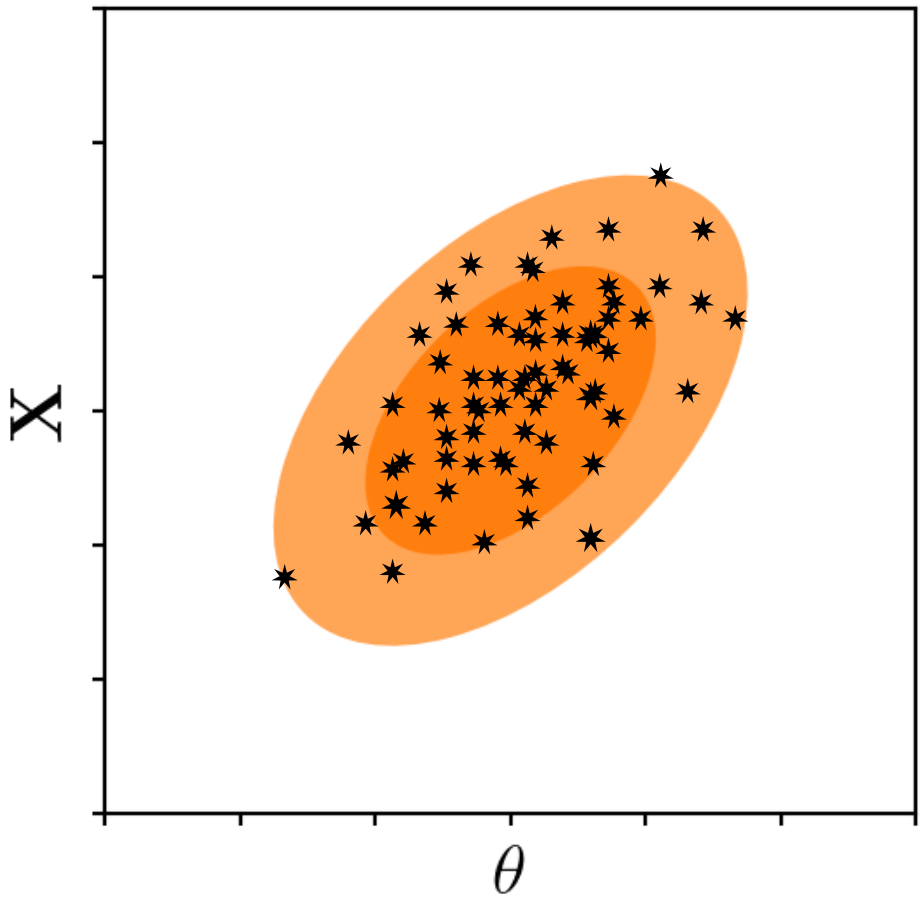
forward model F that can simulate observations:

$$\mathbf{X}'_0, \mathbf{X}'_1 \ldots \sim F(\theta')$$



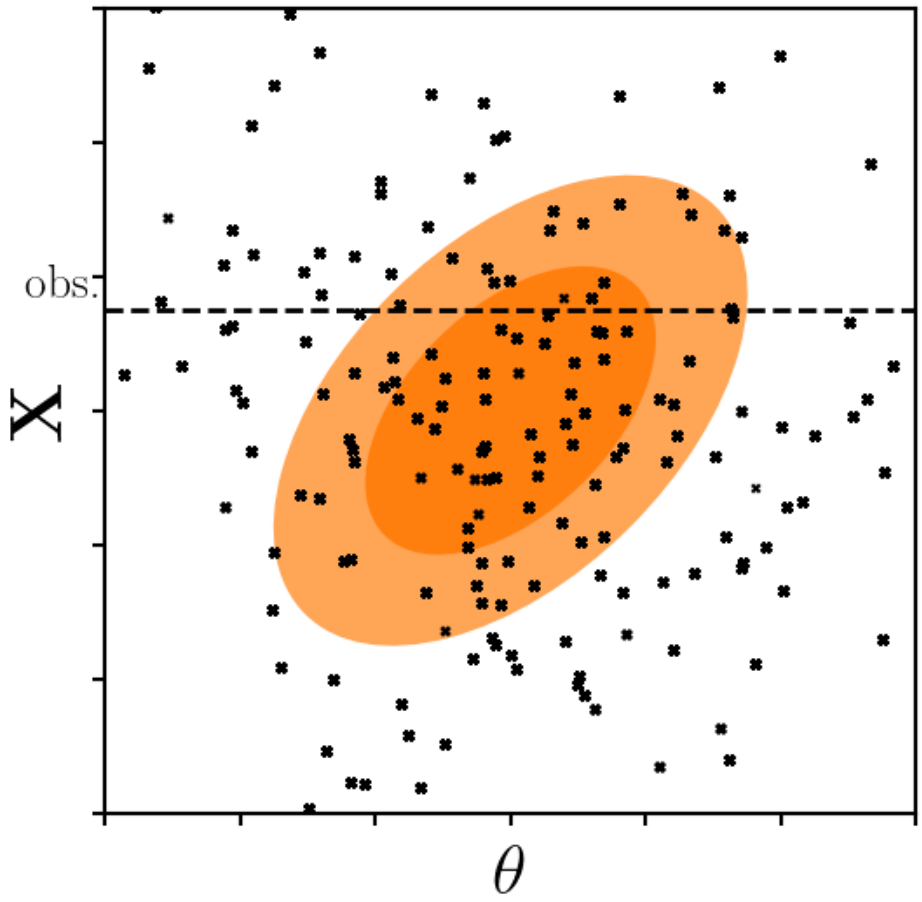
forward model F that can simulate observations:

$$\mathbf{X}'_0, \mathbf{X}'_1 \ldots \sim F(\theta')$$

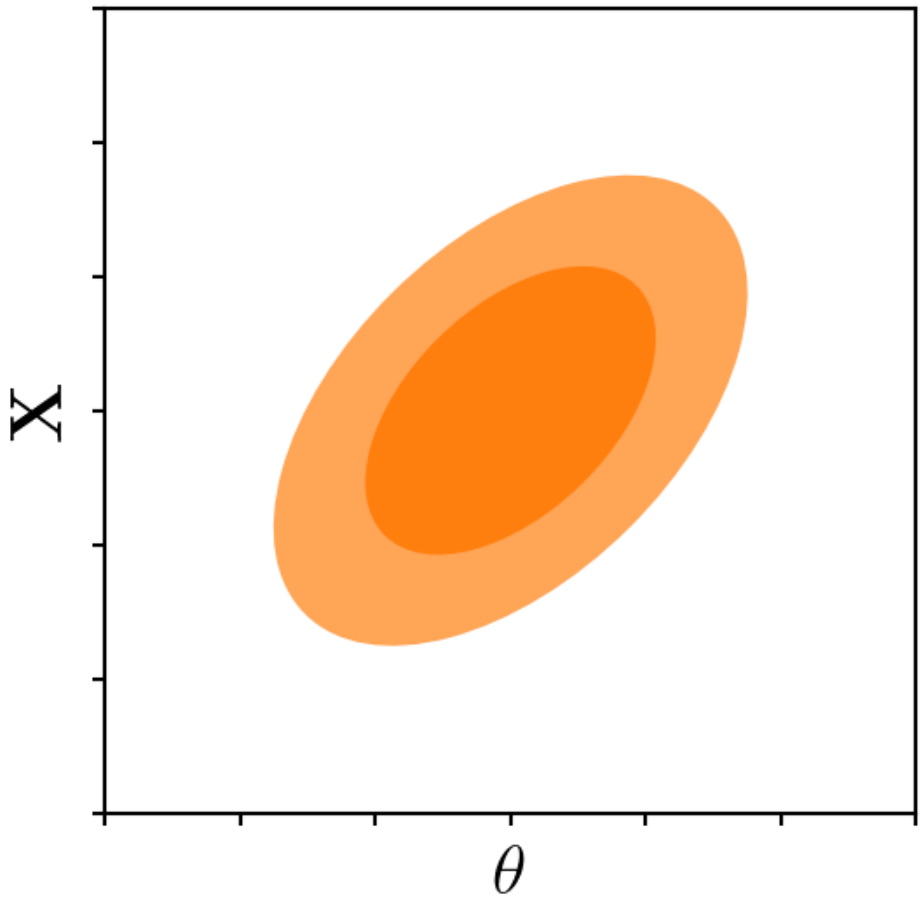


F allows us to sample $(\theta', X') \sim p(\theta, X)$ ← Unknown

with F one way to infer the posterior is through *brute force* — **approximate bayesian computation** (ABC)

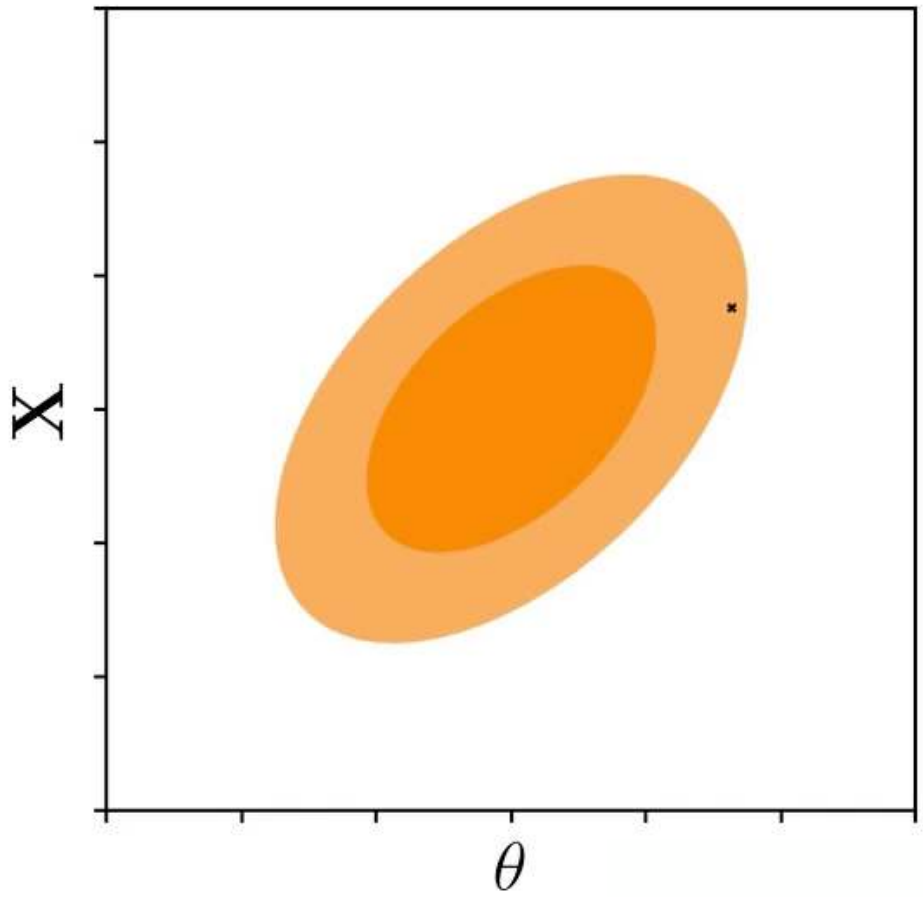


with F one way to infer the posterior is through *brute force* — **approximate bayesian computation** (ABC)



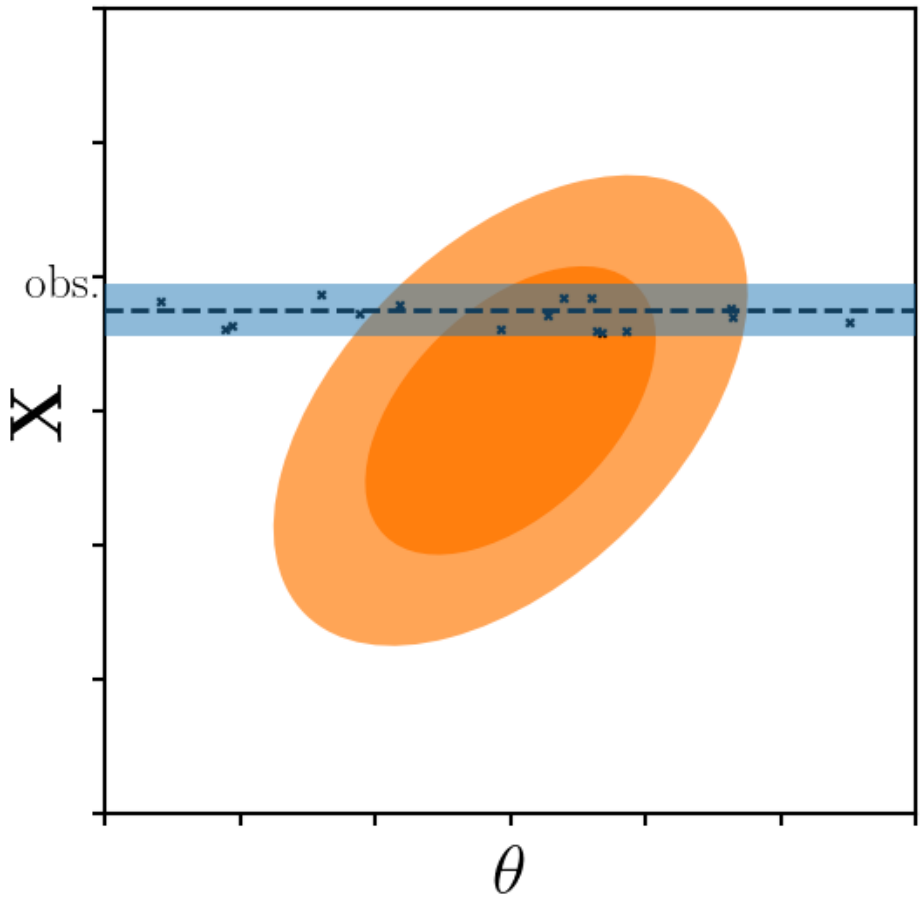
$$\theta' \sim p(\theta)$$
$$\mathbf{X}' \sim F(\theta')$$

with F one way to infer the posterior is through *brute force* — **approximate bayesian computation** (ABC)



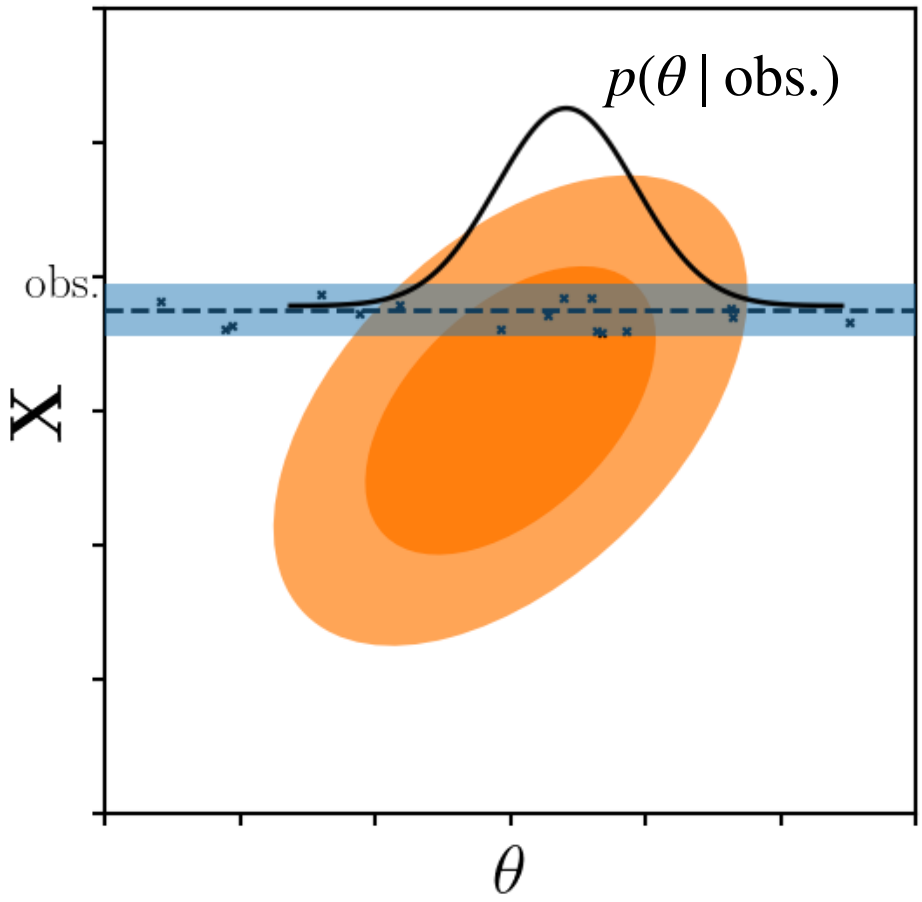
$$\theta' \sim p(\theta)$$
$$\mathbf{X}' \sim F(\theta')$$

with F one way to infer the posterior is through *brute force* — **approximate bayesian computation** (ABC)



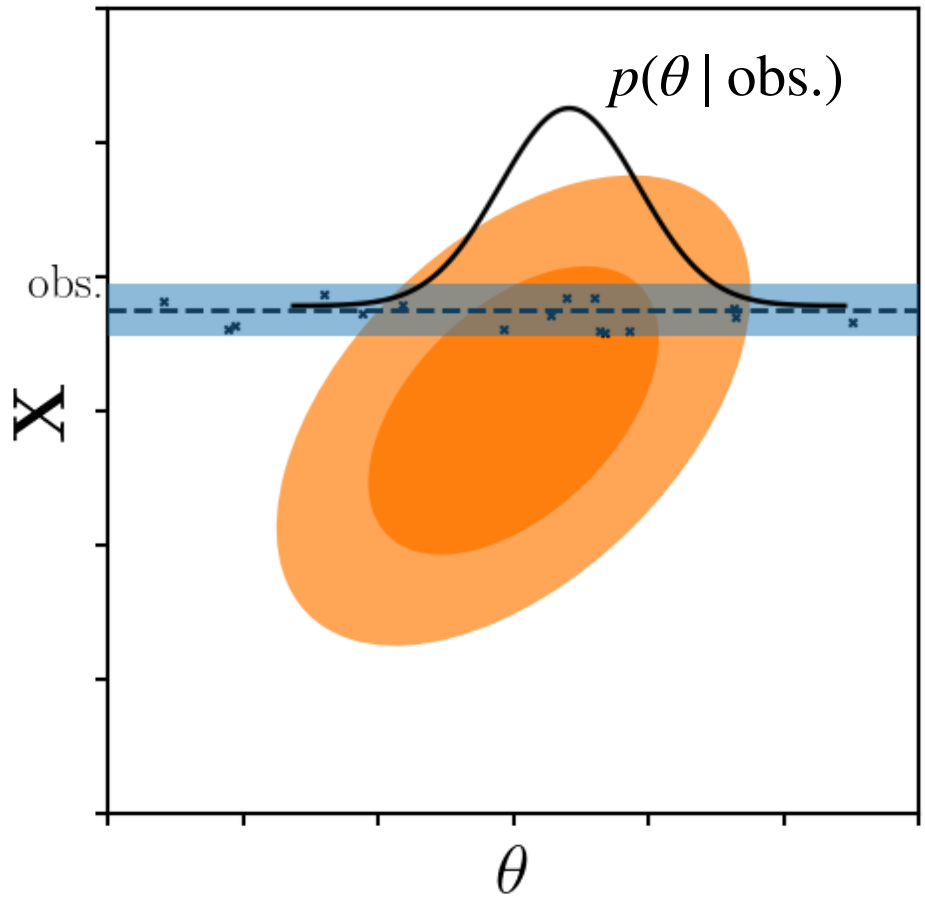
only keep the simulations “close” to the observation

with F one way to infer the posterior is through *brute force* — **approximate bayesian computation** (ABC)



only keep the simulations “close” to the observation

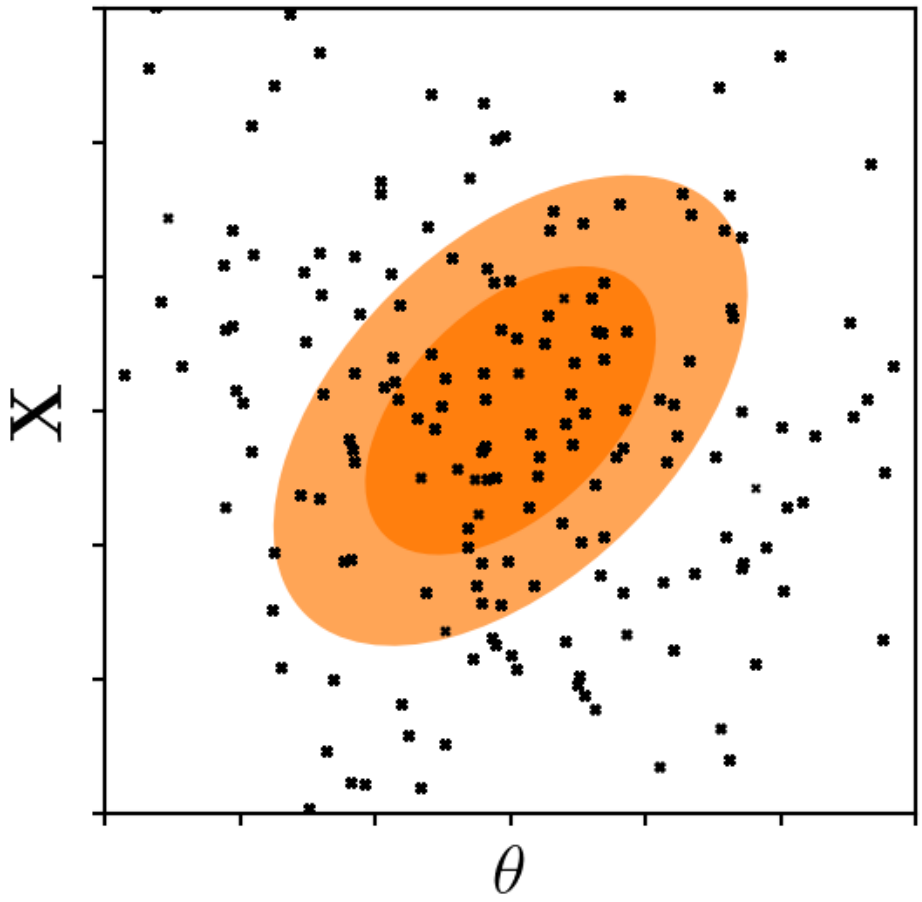
with F one way to infer the posterior is through *brute force* — **approximate bayesian computation** (ABC)



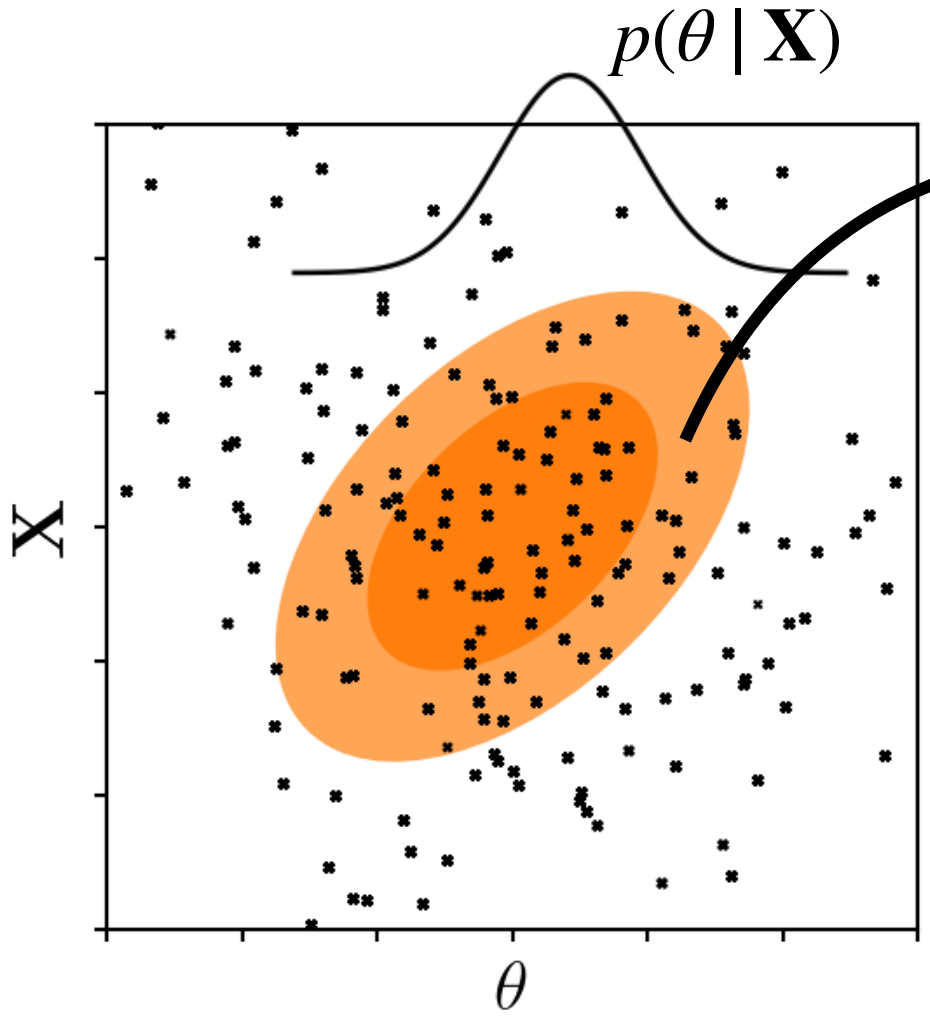
only keep the simulations “close” to the observation

(This is called simulation-based inference, likelihood-free inference or implicit inference)

simulation-based inference is a **density estimation problem!**



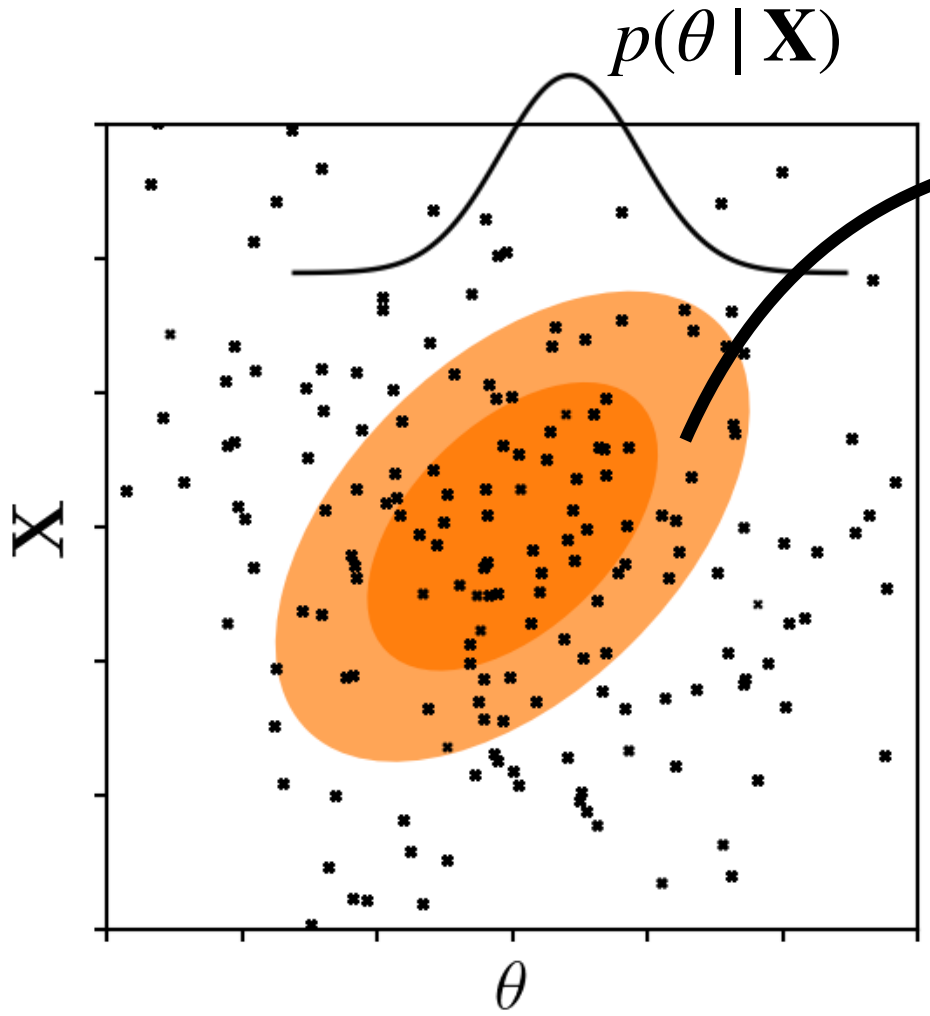
simulation-based inference is a **density estimation problem!**



Generative model:

- **sample from the posterior (amortized)**

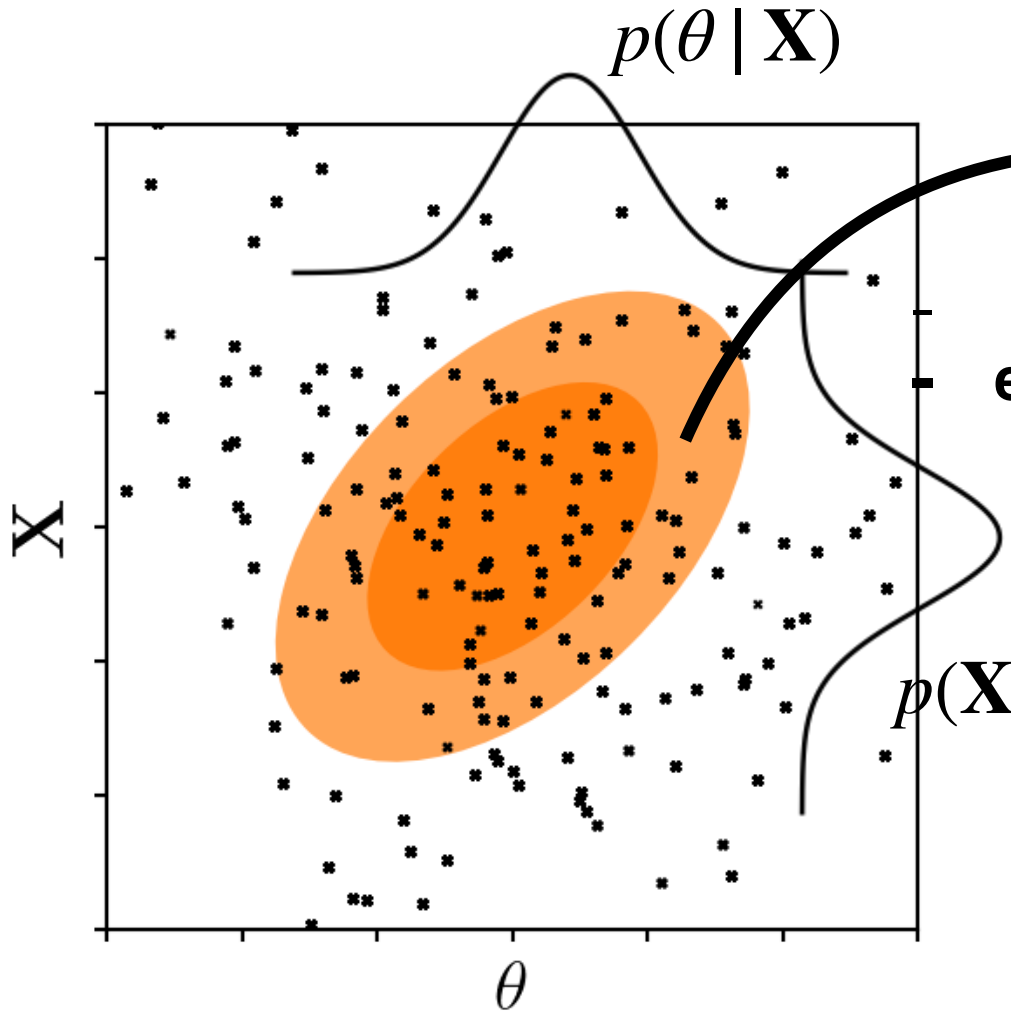
simulation-based inference is a **density estimation problem!**



Generative model:

- **sample from the posterior**
- evaluate likelihood (e.g. NF) and then MCMC

simulation-based inference is a **density estimation problem!**

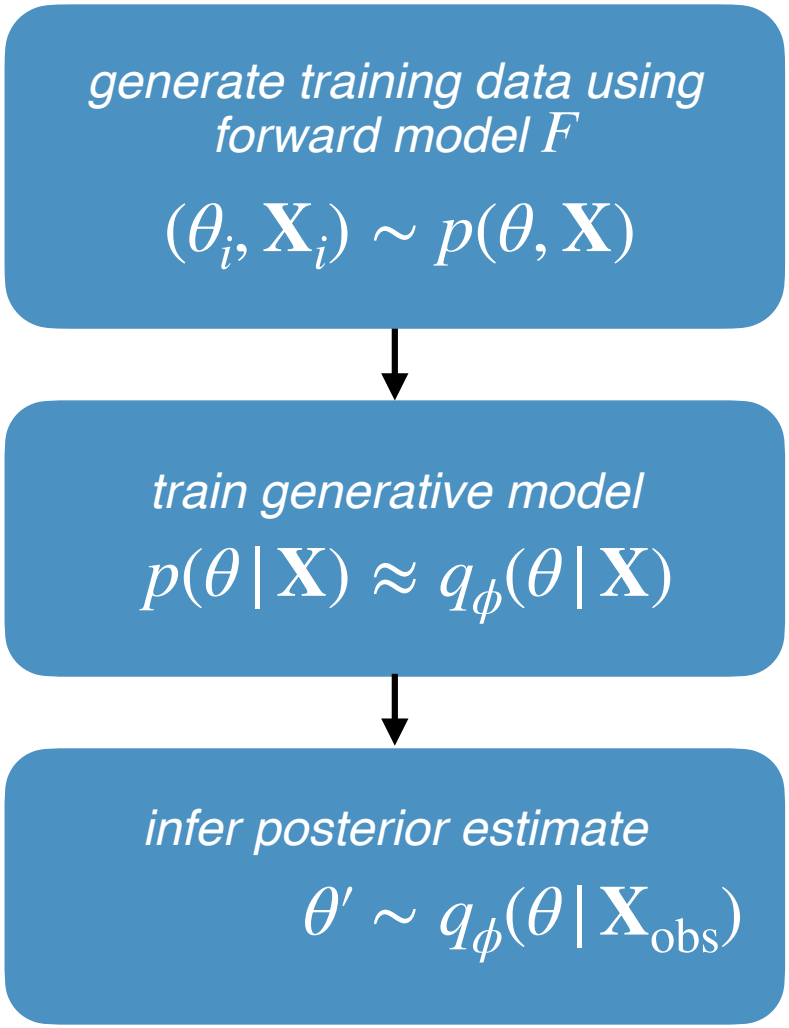


Generative model:

sample from the posterior

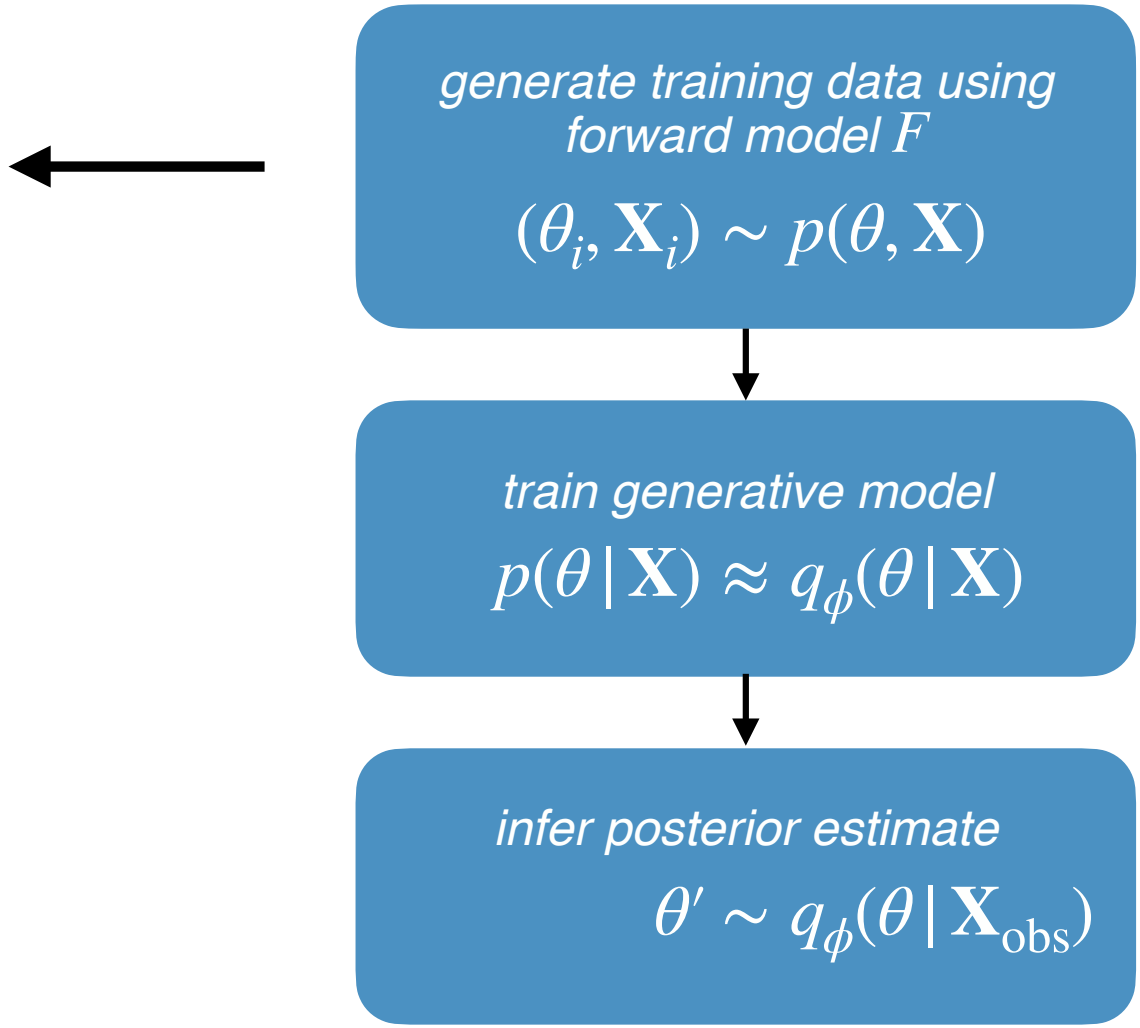
evaluate likelihood (e.g. NF) and then MCMC

simulation-based inference flowchart for amortized inference

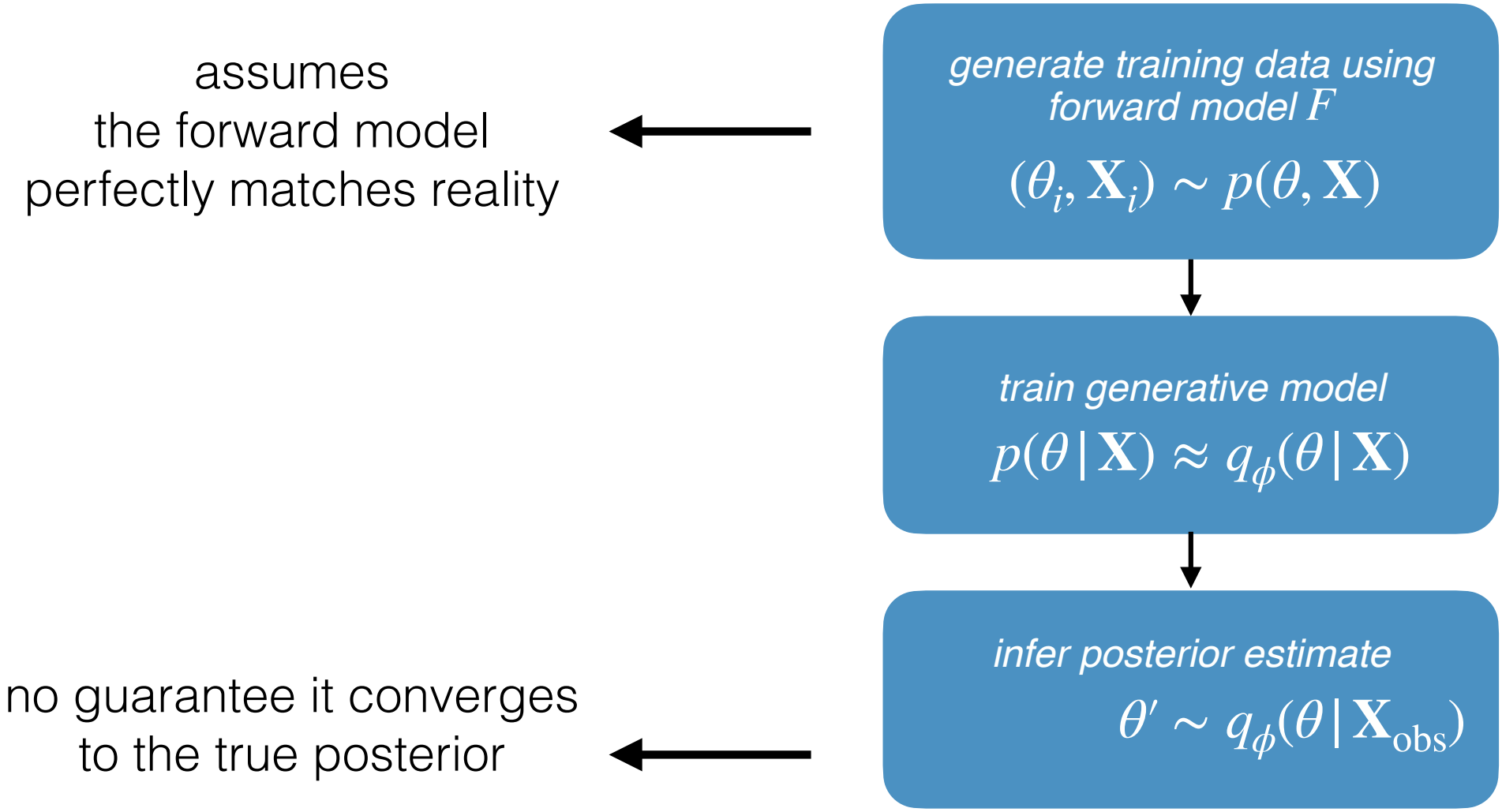


simulation-based inference flowchart for amortized inference

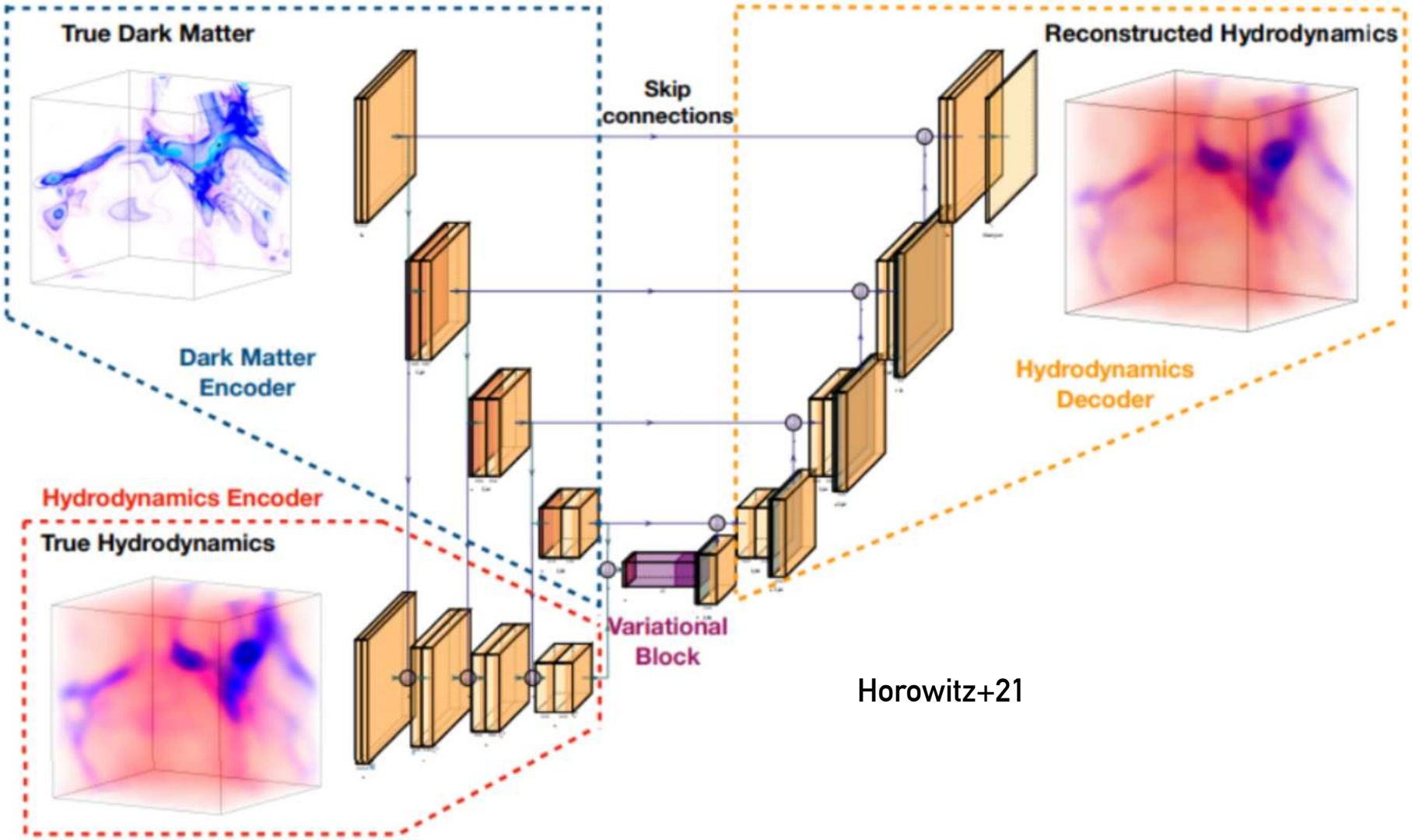
assumes
the forward model
perfectly matches reality



simulation-based inference flowchart for amortized inference



Painting Baryons



Horowitz+21

simulation-based inference for Bayesian **galaxy SED modeling**

Hahn & Melchior (2022)
https://changhoonhahn.github.io/SEDflow

galaxy SED as a **composite stellar population**: star formation history, metallicity history, stellar population synthesis, dust

$$f_{\rm CSP}(\lambda, t) = \int_{t'=0}^{t'=t} {\rm SFR}(t')\ f_{\rm SSP}(t',\ Z(t')\ )\ e^{-\tau_{\rm dust}(t')}\ {\rm d}t'$$

for a review Conroy+(2013)

many galaxy SED models available: *e.g.* ***PROVABGS***

$\log M_*$

$\tau_{\rm BC}$ *birth-cloud dust optical depth*

$\tau_{\rm ISM}$ *diffuse cloud dust optical depth*

δ *index of Calzetti+(2000) attenuation curve*

$\beta_1^{\rm SFH}, \beta_2^{\rm SFH}, \beta_3^{\rm SFH}, \beta_4^{\rm SFH}, f_{\rm burst}, t_{\rm burst}$

$\gamma_1^{\rm ZH}, \gamma_2^{\rm ZH}$

PROVA BGS

Hahn+(2022b), Kwon,***Hahn***,Alsing(2022)

many galaxy SED models available: *e.g.* ***PROVABGS***

$\log M_*$

$\tau_{\rm BC}$

$\tau_{\rm ISM}$

δ

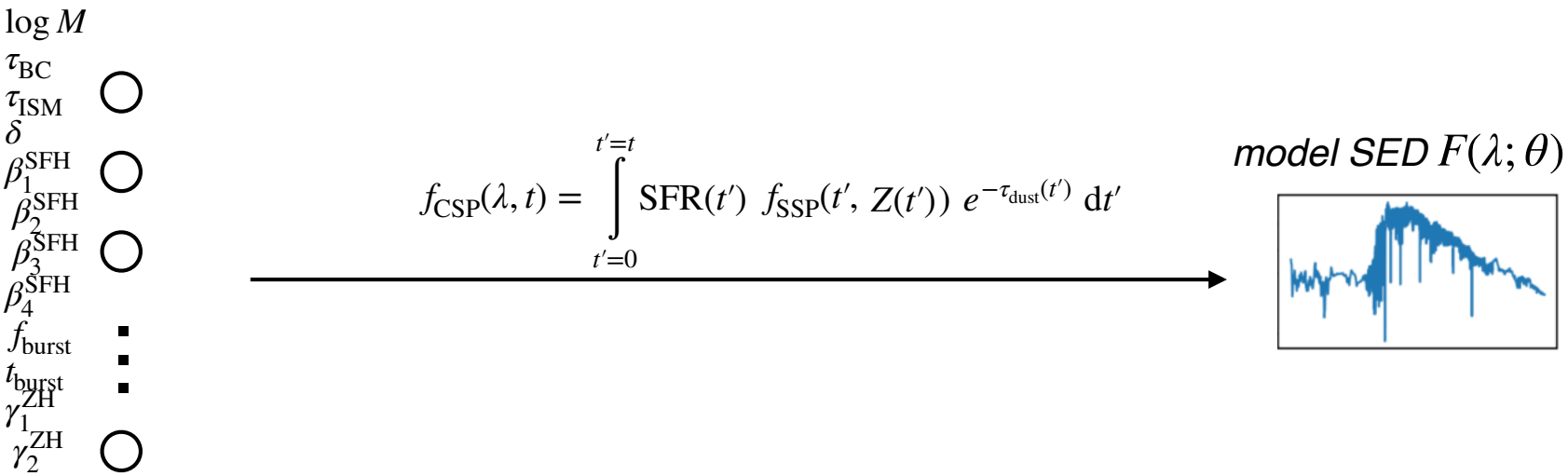
$\beta_1^{\rm SFH}, \beta_2^{\rm SFH}, \beta_3^{\rm SFH}, \beta_4^{\rm SFH}, f_{\rm burst}, t_{\rm burst}$ *star formation history*

$\gamma_1^{\rm ZH}, \gamma_2^{\rm ZH}$ *metallicity history*

PROVA BGS

Hahn+(2022b), Kwon,***Hahn***,Alsing(2022)

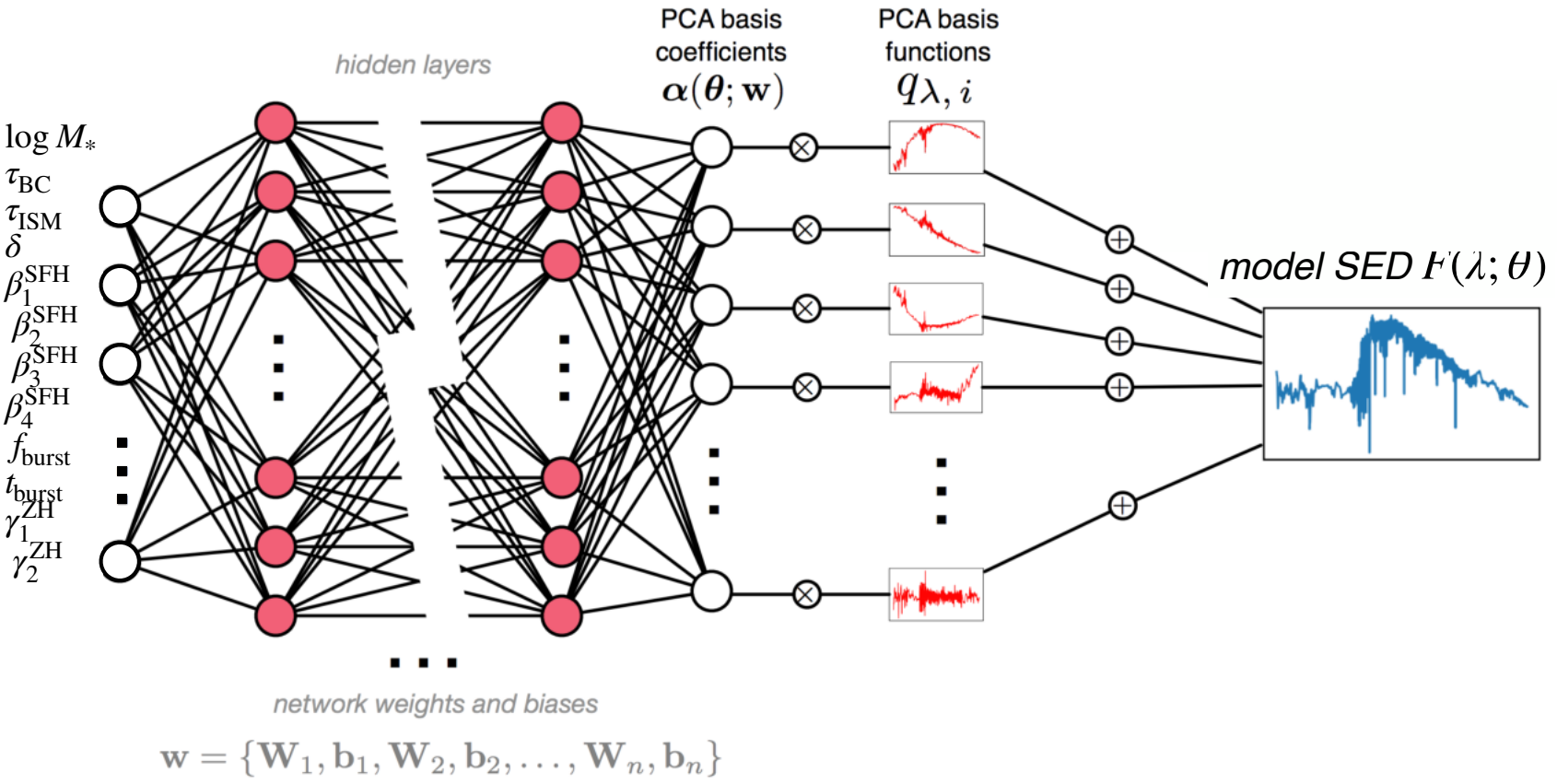
many galaxy SED models available: *e.g. PROVABGS* comes with **neural emulator**



PROVA BGS

*Alsing+**Hahn**+(2019), **Hahn**+(2022b), Kwon,**Hahn**,Alsing(2022)*

many galaxy SED models available: *e.g. PROVABGS* comes with **neural emulator** — *1000x faster*



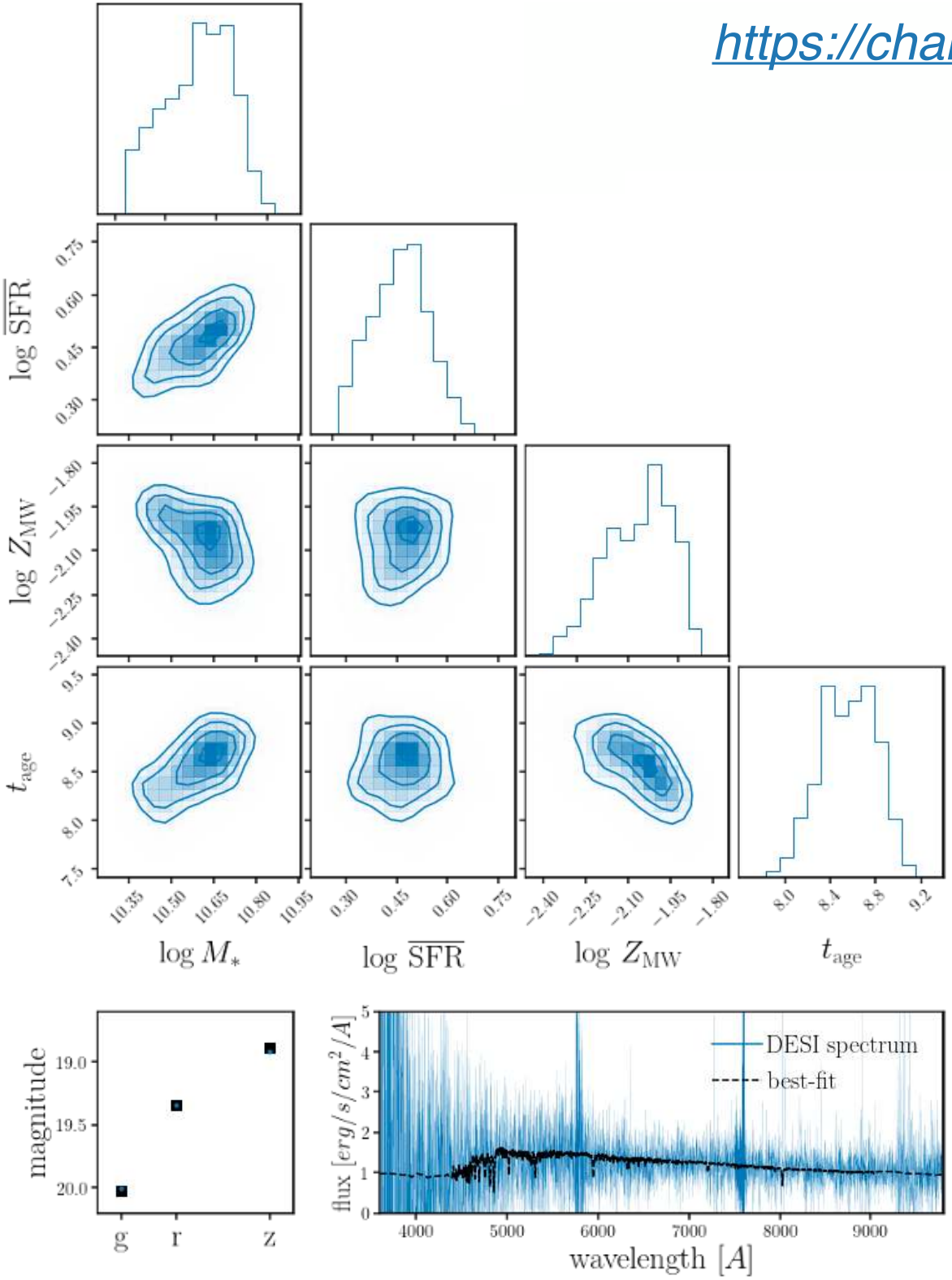
James Kyubin Kwon
UC Santa Barbara

PROVA BGS

*Alsing+**Hahn**+(2019), **Hahn**+(2022b), Kwon,**Hahn**,Alsing(2022)*

https://changhoonhahn.github.io/provabgs

~10 CPU min/galaxy



PROVA BGS

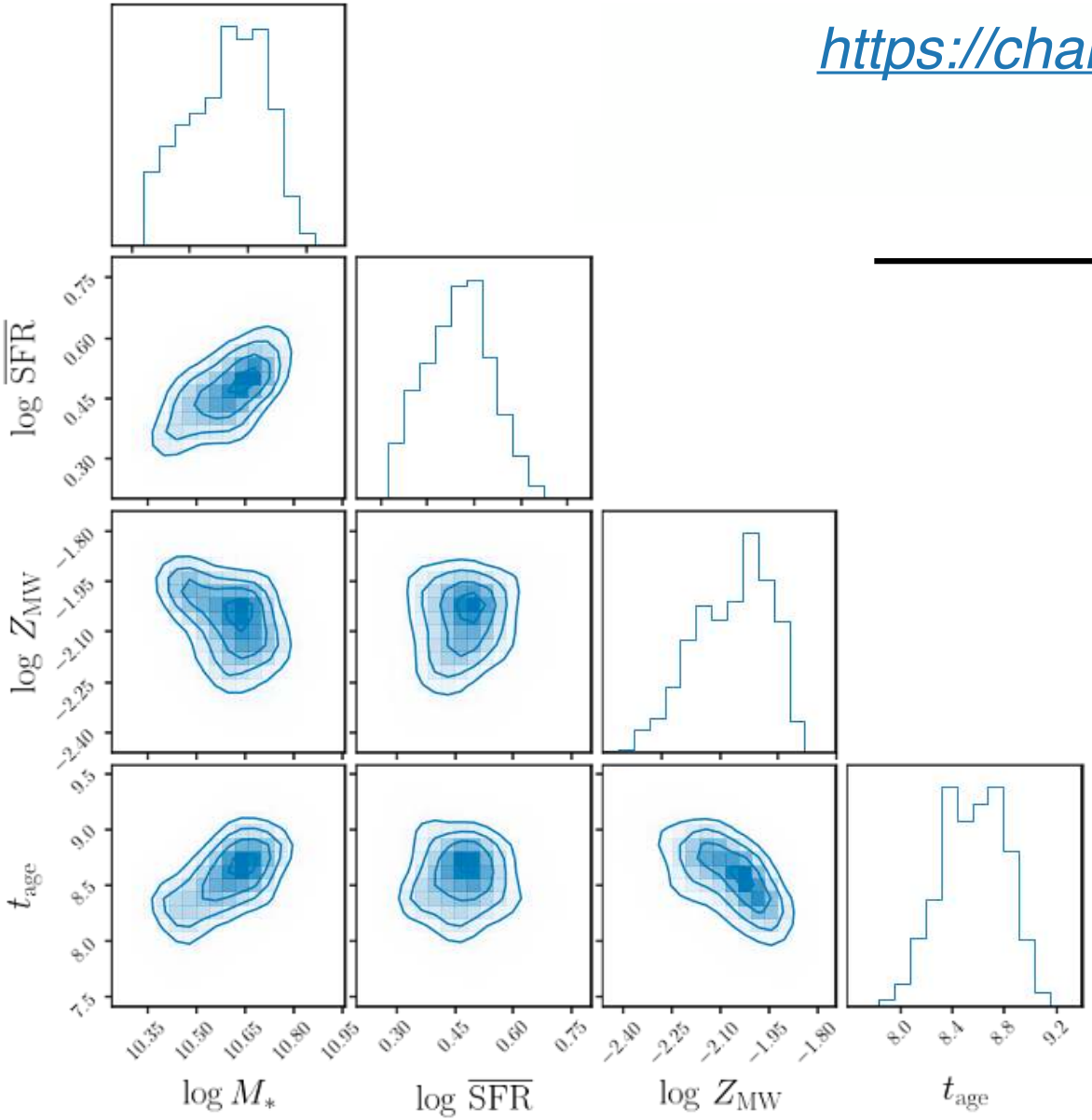
Hahn+(2022b), Kwon,***Hahn***,Alsing(2022)

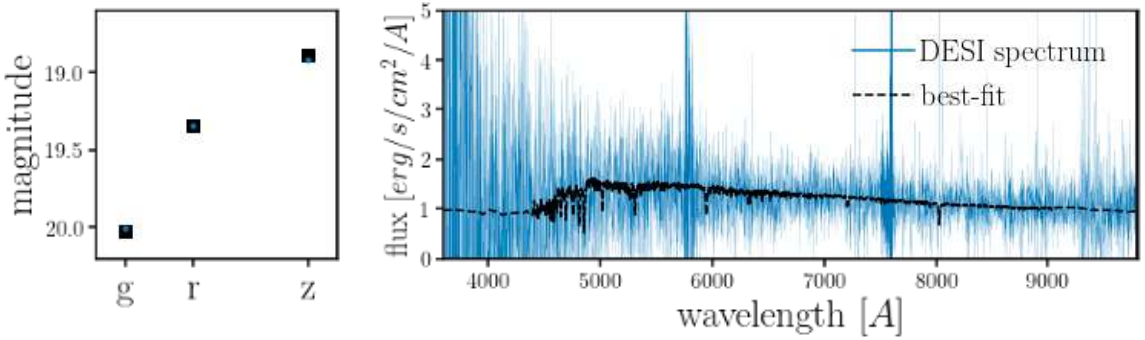
https://changhoonhahn.github.io/provabgs

~10 CPU min/galaxy

x (≫10 million galaxies)

=≫2 million CPU hours





PROVA BGS

Hahn+(2022b), Kwon,***Hahn***,Alsing(2022)

why **simulation-based inference**?

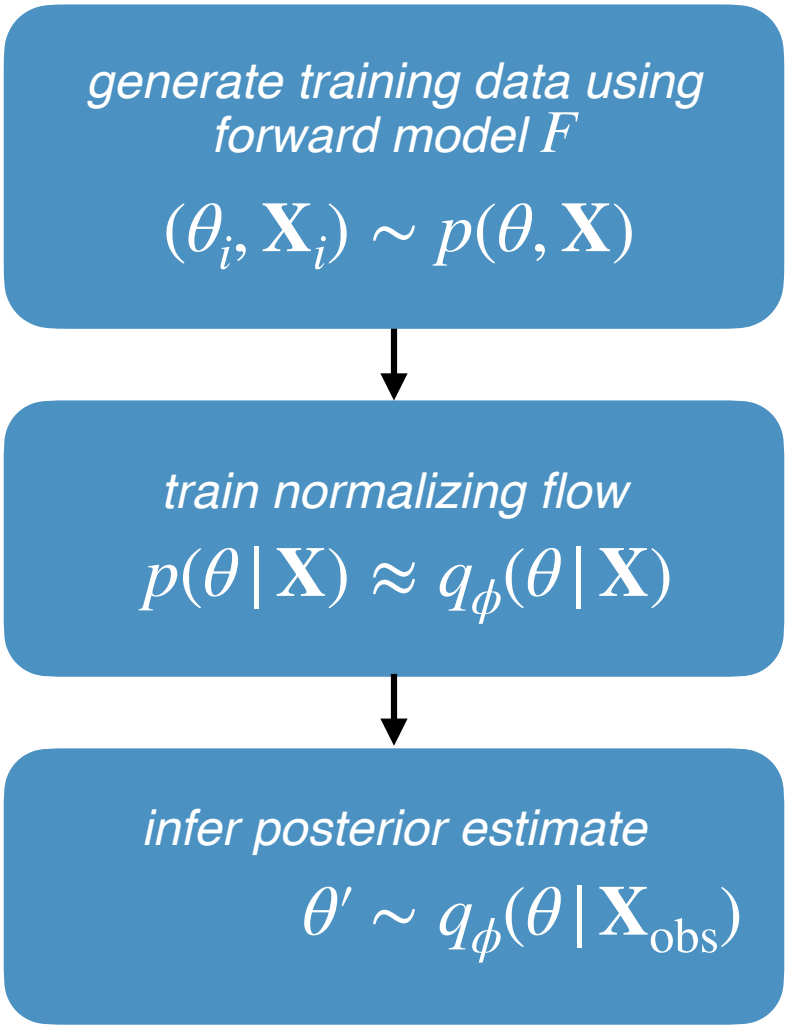
Gaussian likelihood is often **an incorrect assumption**

simulations have **advantages over analytic models**

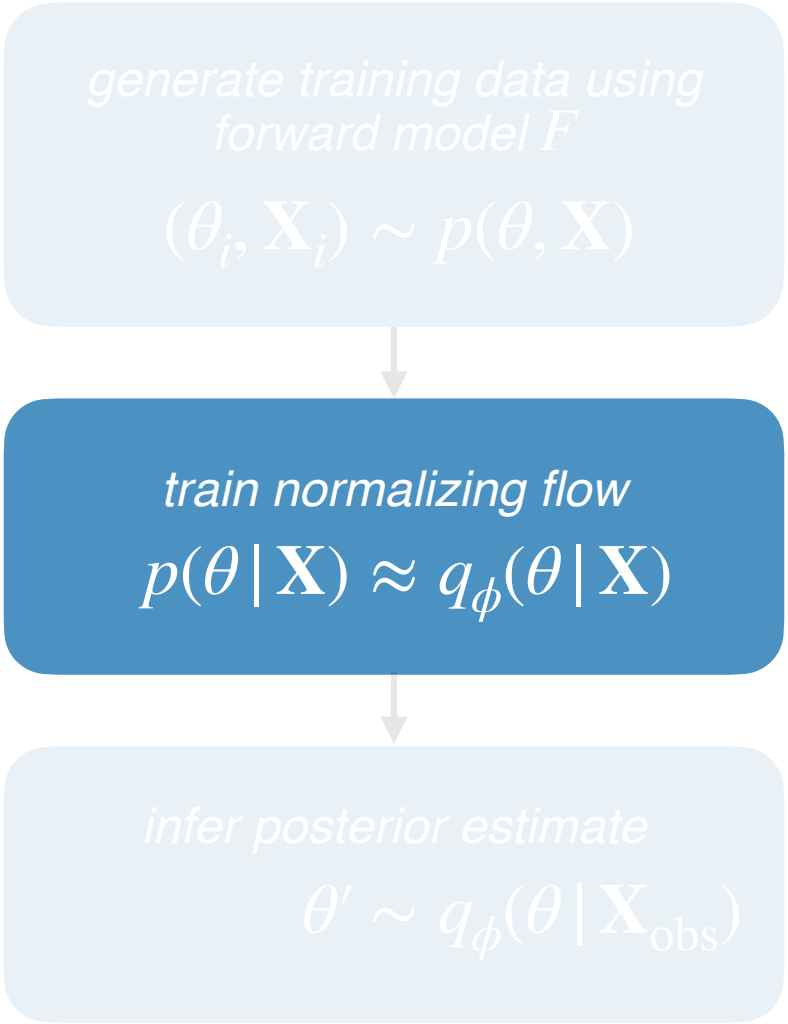
amortized inference to infer *billions* of observations

references: Hogg+(2010), Hogg (2012)

simulation-based inference using **normalizing flows** flowchart

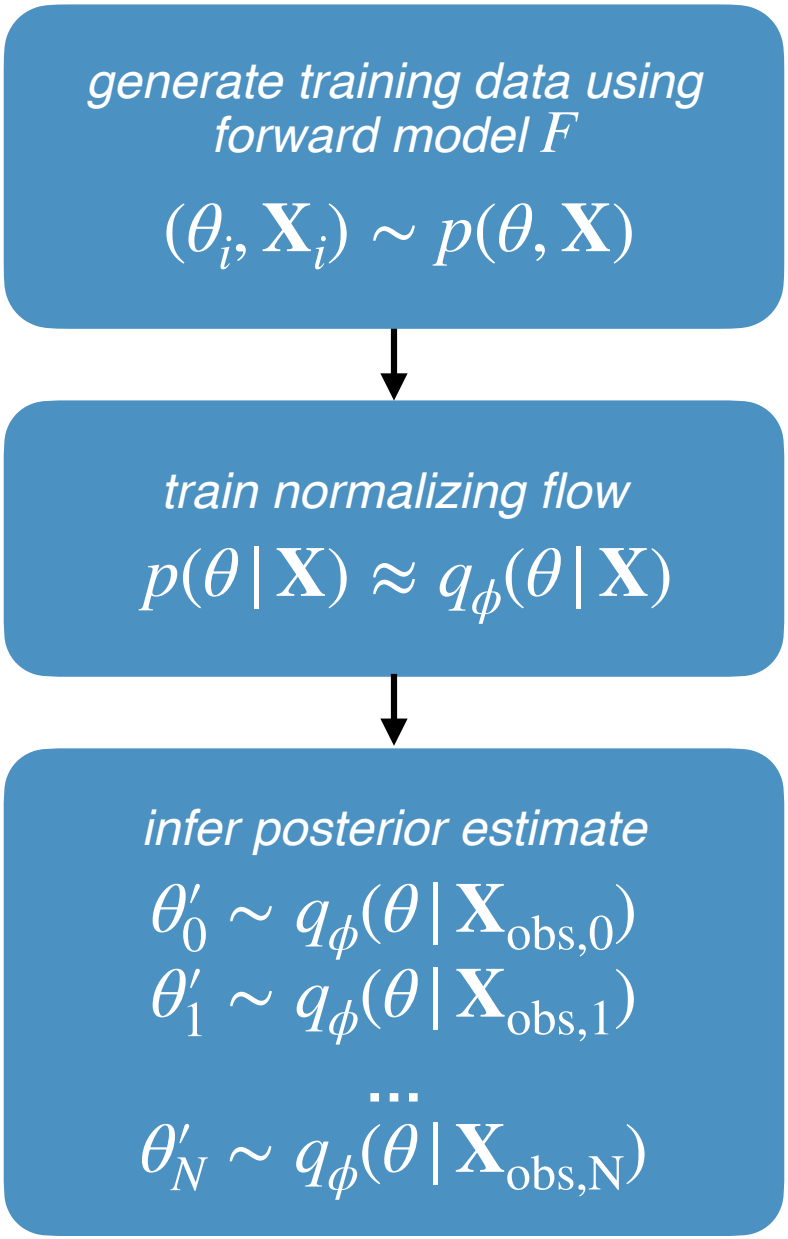


simulation-based inference using **normalizing flows** flowchart



we can train $p(\theta \mid \mathbf{X}) \approx q_\phi(\theta \mid \mathbf{X})$ over the **entire** $\mathbf{X}$-space

simulation-based inference using **normalizing flows** flowchart



SPS parameters

$$p(\theta_{\mathrm{SPS}} \mid D, \sigma, z)$$

SPS parameters

$$p(\theta_{\mathrm{SPS}} \mid D, \sigma, z)$$

SDSS photometry

SPS parameters

$$p(\theta_{\mathrm{SPS}} \mid D, \sigma, z)$$

SDSS photometry

measured uncertainties

SPS parameters

$$p(\theta_{\mathrm{SPS}} \mid D, \sigma, z)$$

SDSS photometry

measured uncertainties

measured redshift

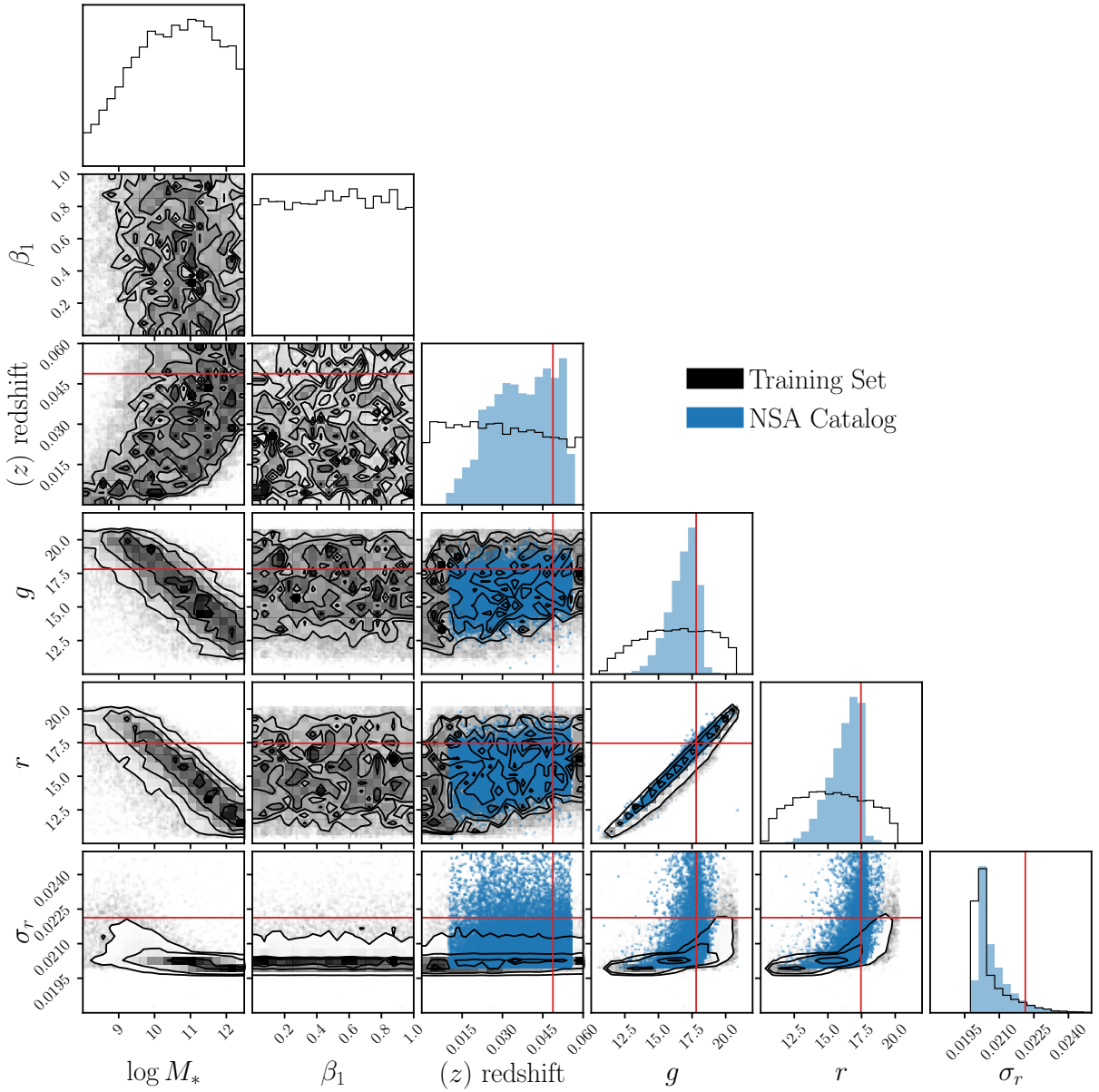
generate training data using forward model F

$$(\theta_i, \mathbf{X}_i) \sim p(\theta, \mathbf{X})$$

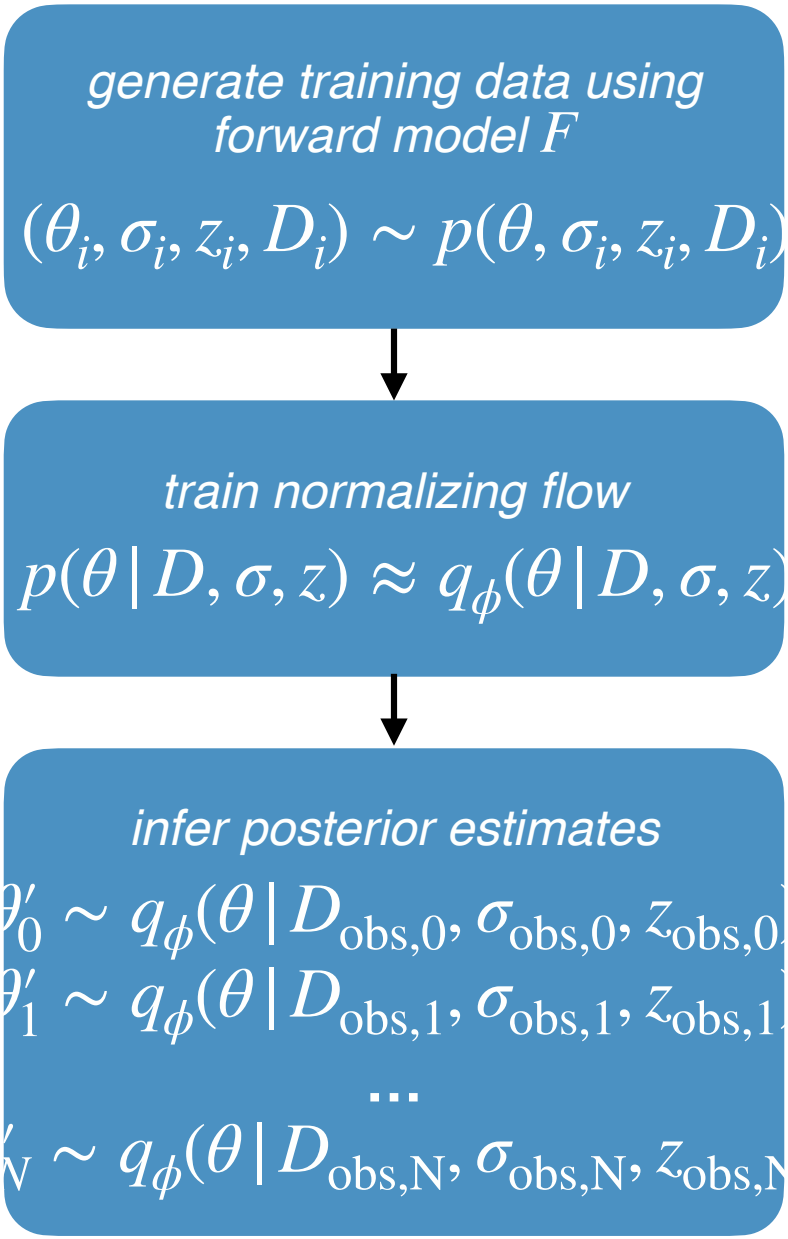
sample $\theta' \sim p(\theta_{\rm SPS}, \sigma, z)$

forward model observations

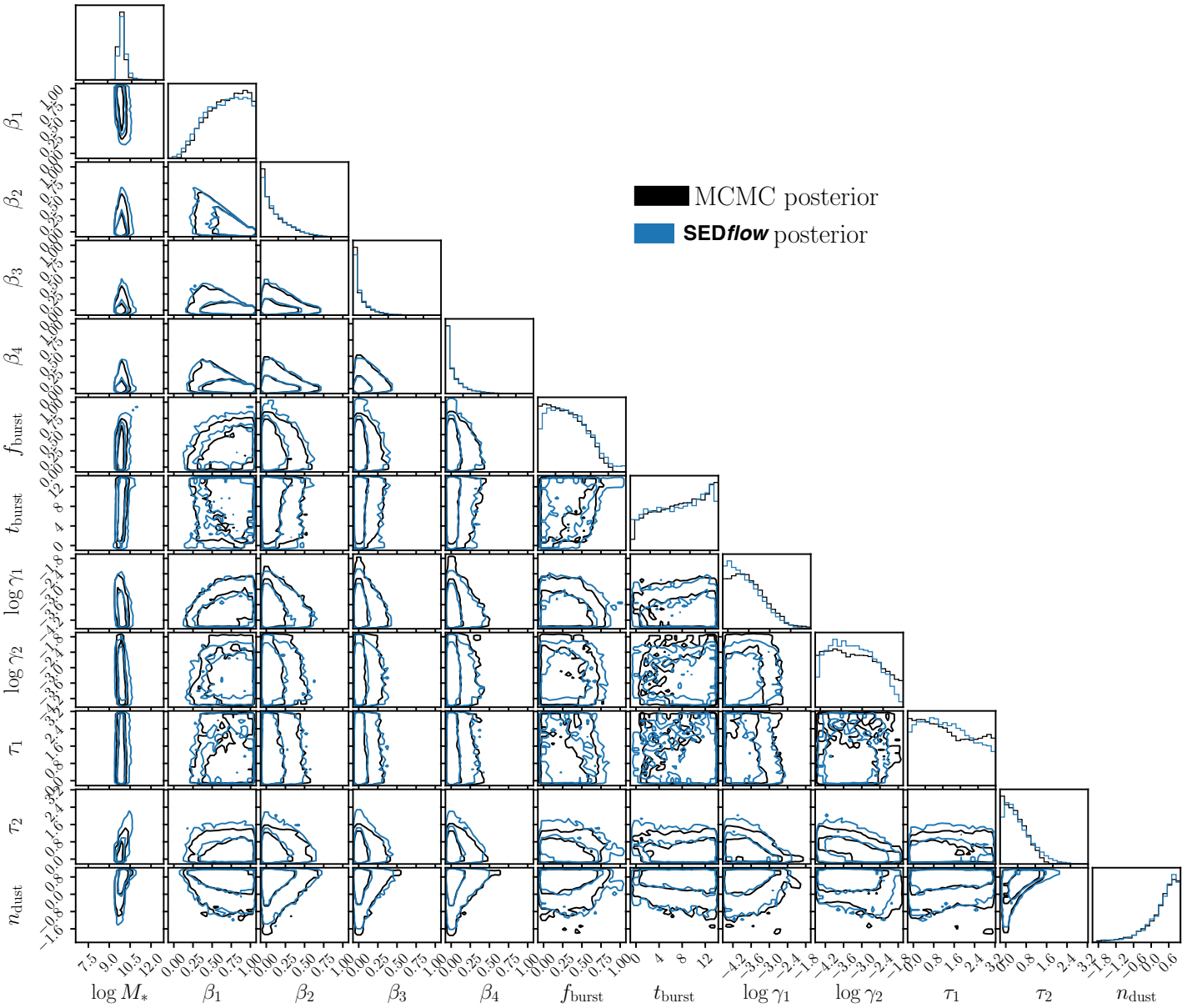
$$D' = \text{SED model}(\theta) + \text{noise}(\theta)$$



Hahn *& Melchior (2022)*

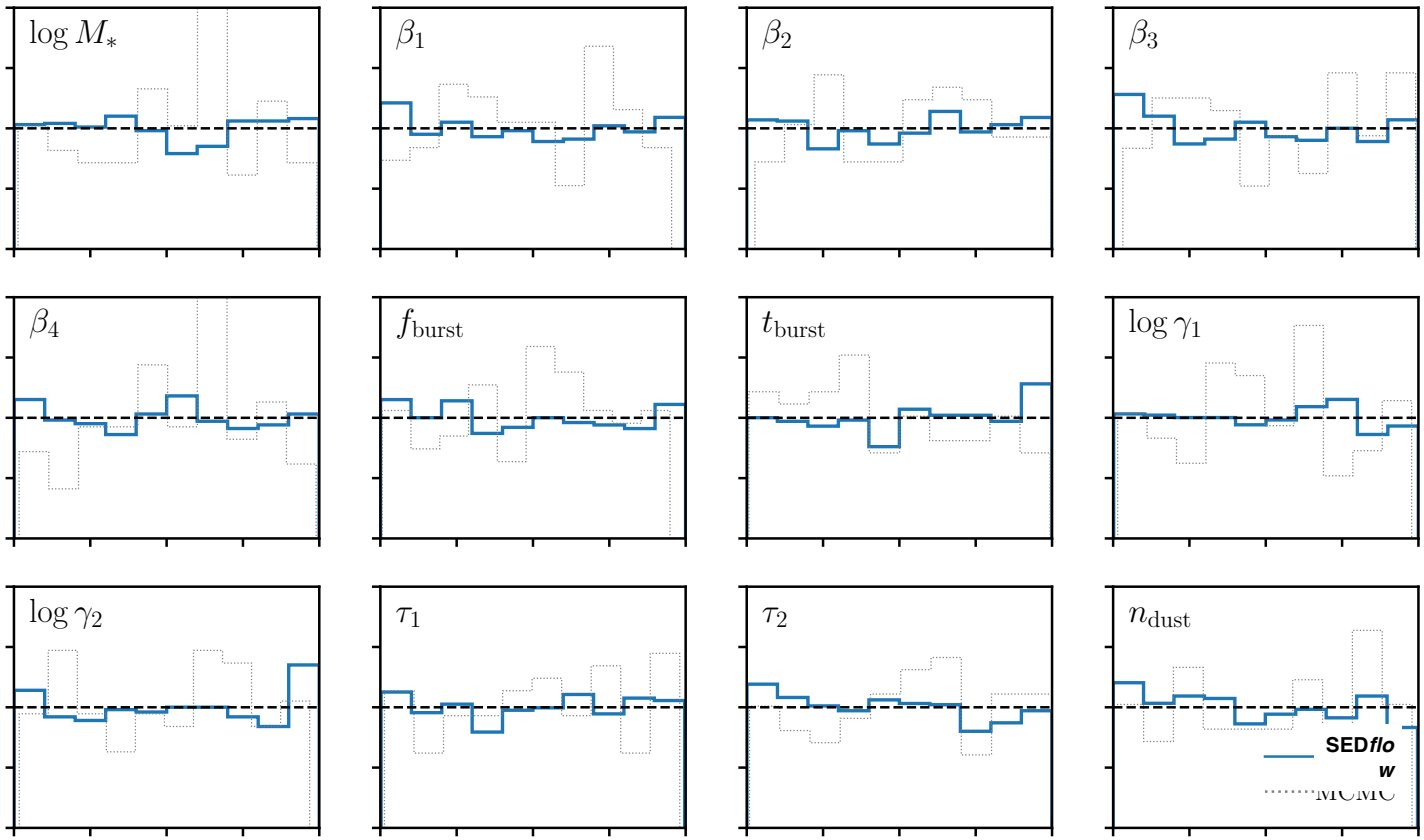

generate training data using forward model F
$(\theta_i, \sigma_i, z_i, D_i) \sim p(\theta, \sigma_i, z_i, D_i)$
train normalizing flow
$p(\theta \mid D, \sigma, z) \approx q_\phi(\theta \mid D, \sigma, z)$
infer posterior estimates
$\theta'_0 \sim q_\phi(\theta \mid D_{\text{obs},0}, \sigma_{\text{obs},0}, z_{\text{obs},0}$
$\theta'_1 \sim q_\phi(\theta \mid D_{\text{obs},1}, \sigma_{\text{obs},1}, z_{\text{obs},1}$
...
$\theta'_N \sim q_\phi(\theta \mid D_{\text{obs,N}}, \sigma_{\text{obs,N}}, z_{\text{obs,N}}$

SED*flow* posteriors are *indistinguishable* from MCMC posteriors



Hahn *& Melchior (2022)*

SED*flow* posteriors accurately estimate the true posteriors — *more accurate than MCMC*



*simulation-based calibration — **Hahn** & Melchior (2022)*

SED*flow* posteriors accurately estimate the true posteriors —
takes ~1 second per galaxy

SED*flow* posteriors accurately estimate the true posteriors —
takes ~1 second per galaxy

(*~1 second per galaxy*)
x (10 million galaxies)

< 3000 CPU hours

Thank you!

* mhuertas@iac.es