Generative Modelling, Density Estimation and Simulation Based Inference

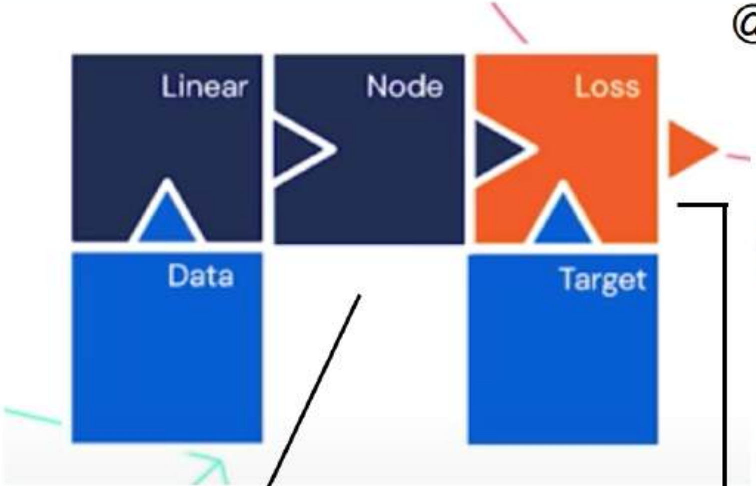
A. Boucaud, M. Huertas-Company

APC, CNRS, FRANCE

Ecole d'été Rodolphe Clédassou - Session 2024 - Cycle3

RECAP Cycle 1: NNs as universal approximators

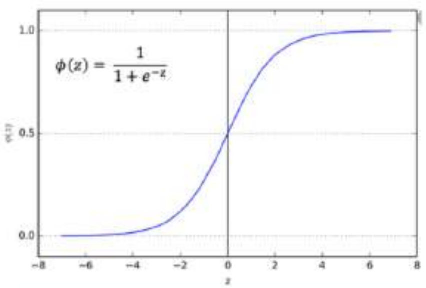
(deepmind @Czarnecki)



$$f_w(X) = \theta$$

$$\boxed{g_3(W_3 g_2(W_2 g_1(W_1 \vec{x_0})))}$$

activation function

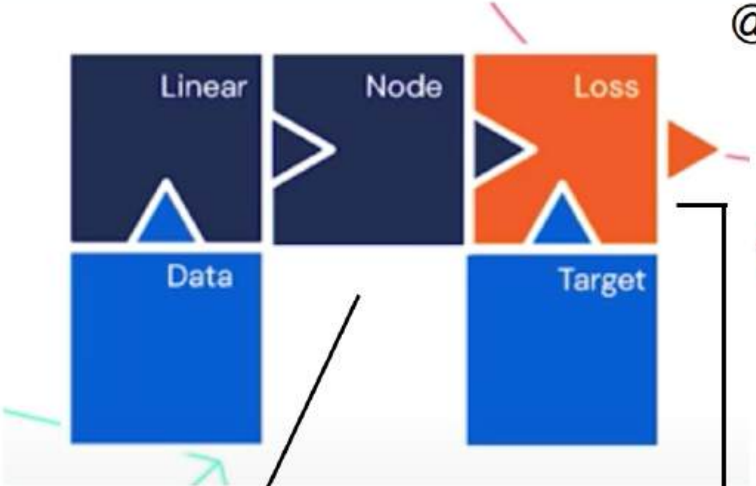


$$\frac{1}{N}\sum_{i=1}^{N}(y_i - f_w(x))^2$$

Minimized through gradient descent (backpropagation)

RECAP Cycle 1: NNs as universal approximators

(deepmind @Czarnecki)



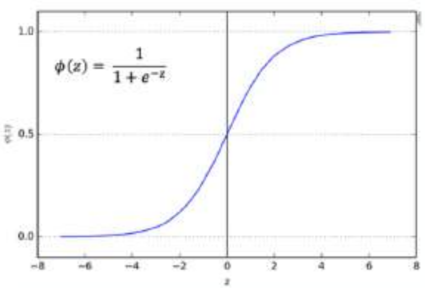
$$p_w(\theta|X)$$

$$\cancel{f_w(X) = \theta}$$

$$\vdots$$

$$\boxed{g_3(W_3 g_2(W_2 g_1(W_1 \vec{x_0})))}$$

activation function



$$\frac{1}{N}\sum_{i=1}^{N}(y_i - f_w(x))^2$$

Minimized through gradient descent (backpropagation)

RECAP Cycle 2: supervised deep learning

Spatially correlated data in euclidean grid

CNN

$$p_w(\theta|X)$$

GNN

Non-euclidean data with complex locality

TFs

Time series, sequences

Estimate a probability density function of an arbitrarily high dimensional data

High
dimensional data
(e.g. images)

$$p_w(X)$$

Likely not
Gaussian

Estimate a probability density function of an arbitrarily high dimensional data

High dimensional data (e.g. images)

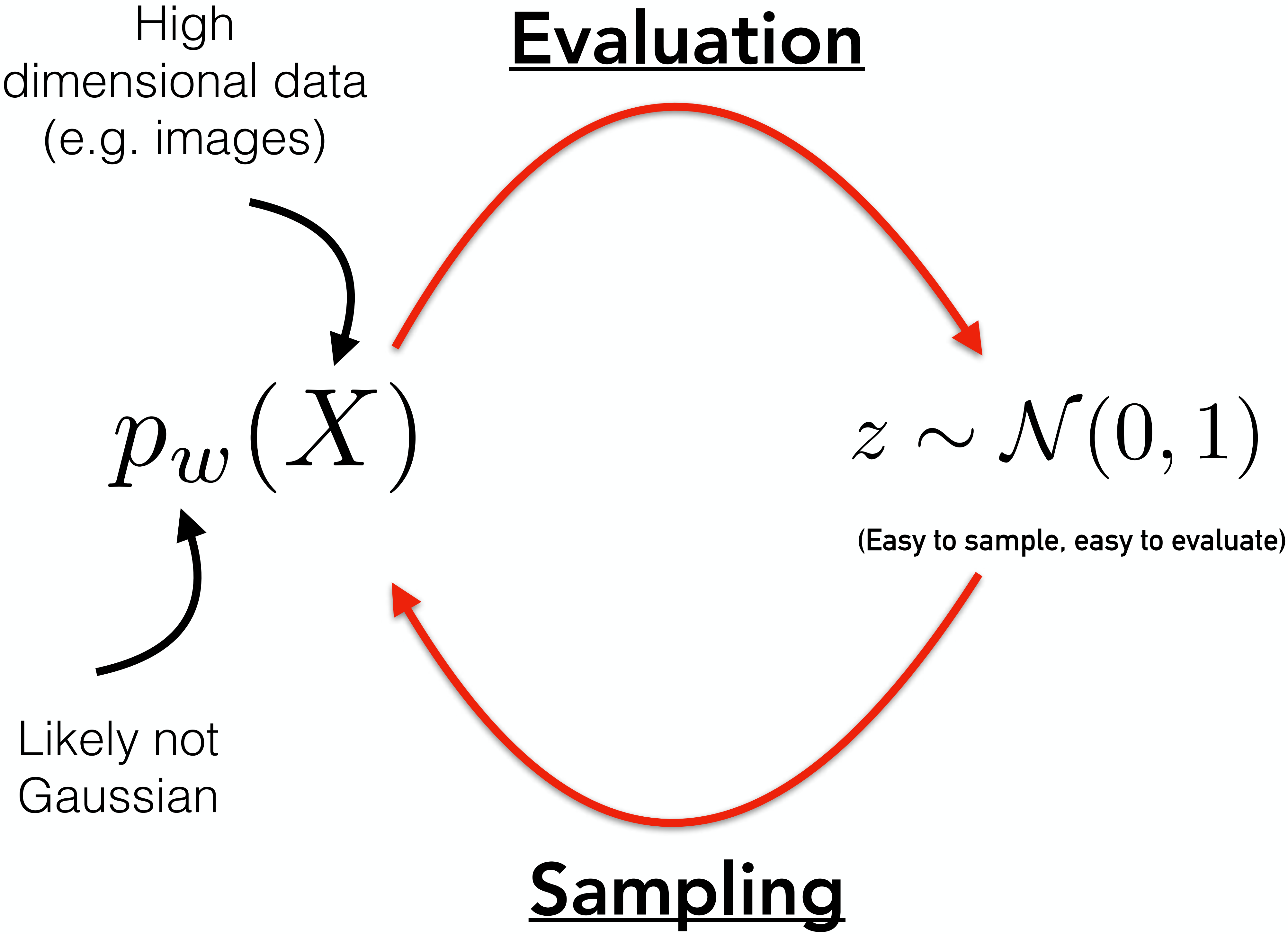
$$p_w(X)$$

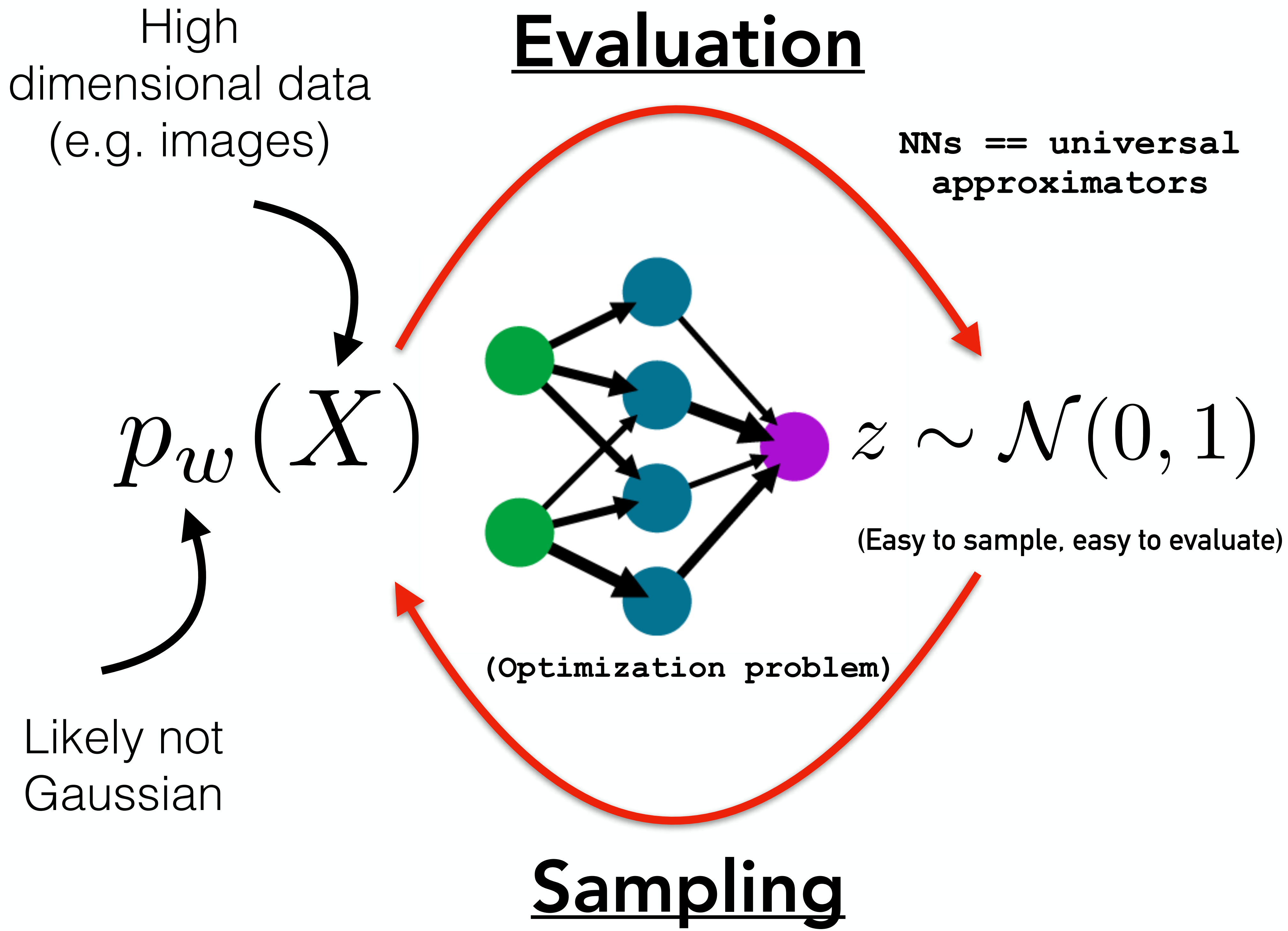
Likely not Gaussian

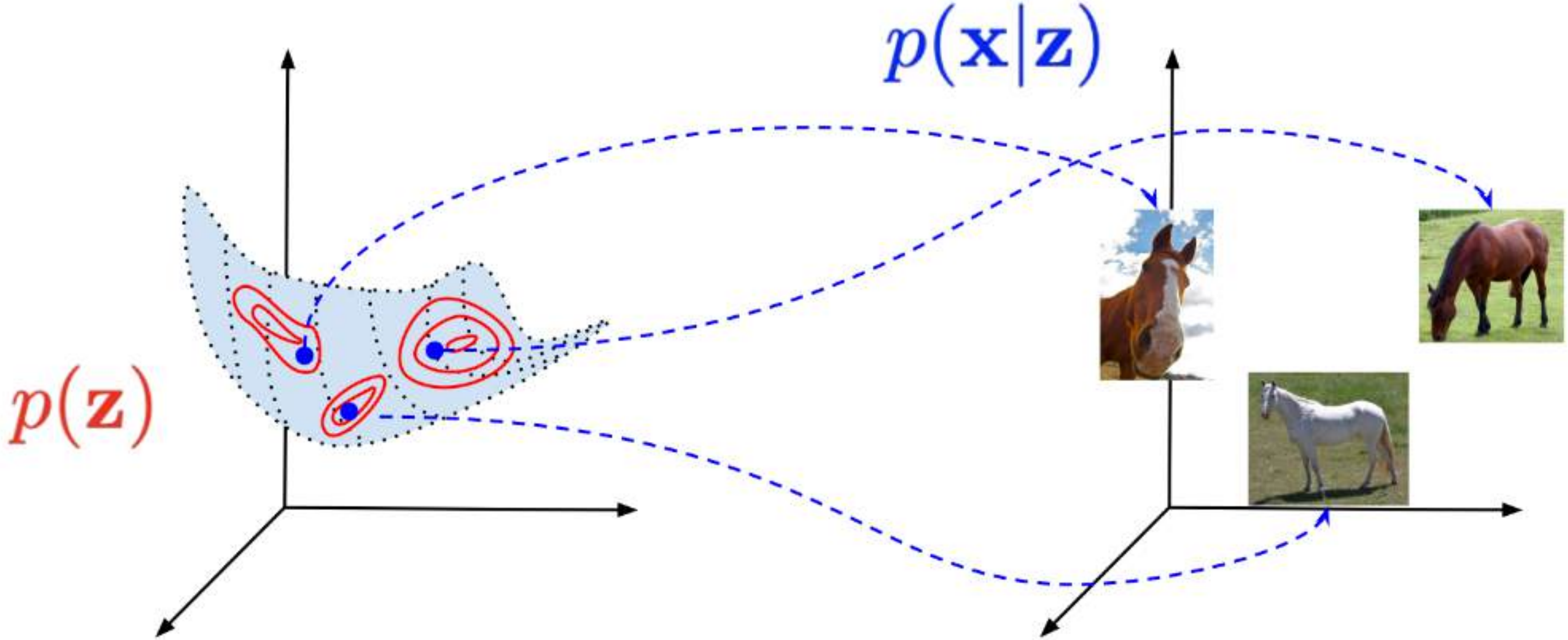
Why?

1. Sampling. how can I interpolate / extrapolate a (small, sparse) dataset to generate new data sampled from the same (unknown) distribution?

2. Likelihood evaluation. what is the probability that a new observation is drawn from the same (unknown) distribution as some reference (small, sparse) dataset?


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


$p(\mathbf{x}|\mathbf{z})$
$p(\mathbf{z})$

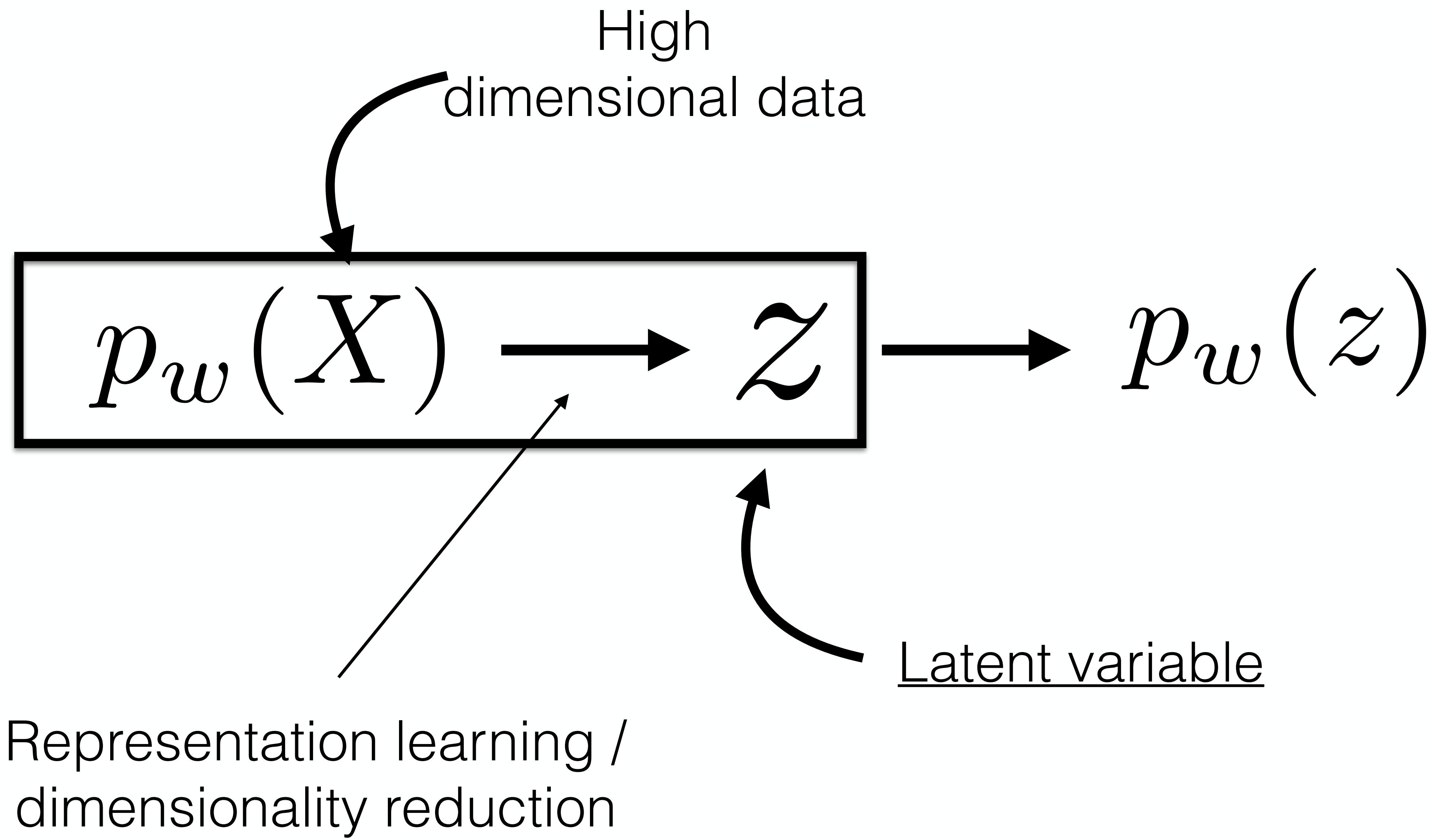
High
dimensional data

$$p_w(X) \longleftrightarrow z \longleftrightarrow p_w(z)$$

Latent variable

Representation learning /
dimensionality reduction

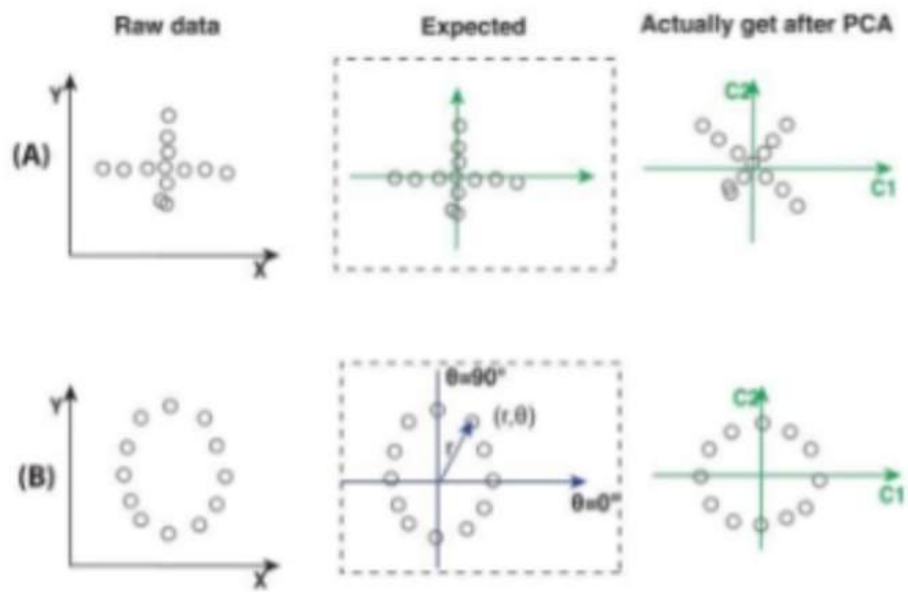
1. Representation learning


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

LIMITATIONS OF PCA

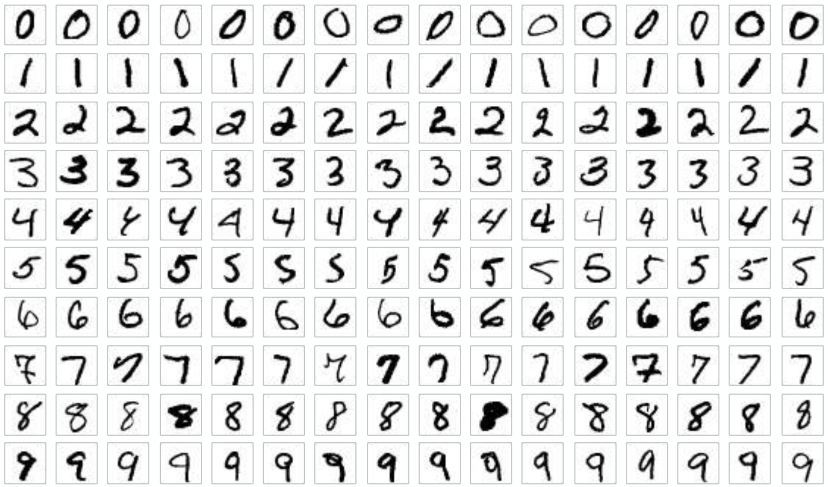
PCA APPLY LINEAR TRANSFORMATIONS

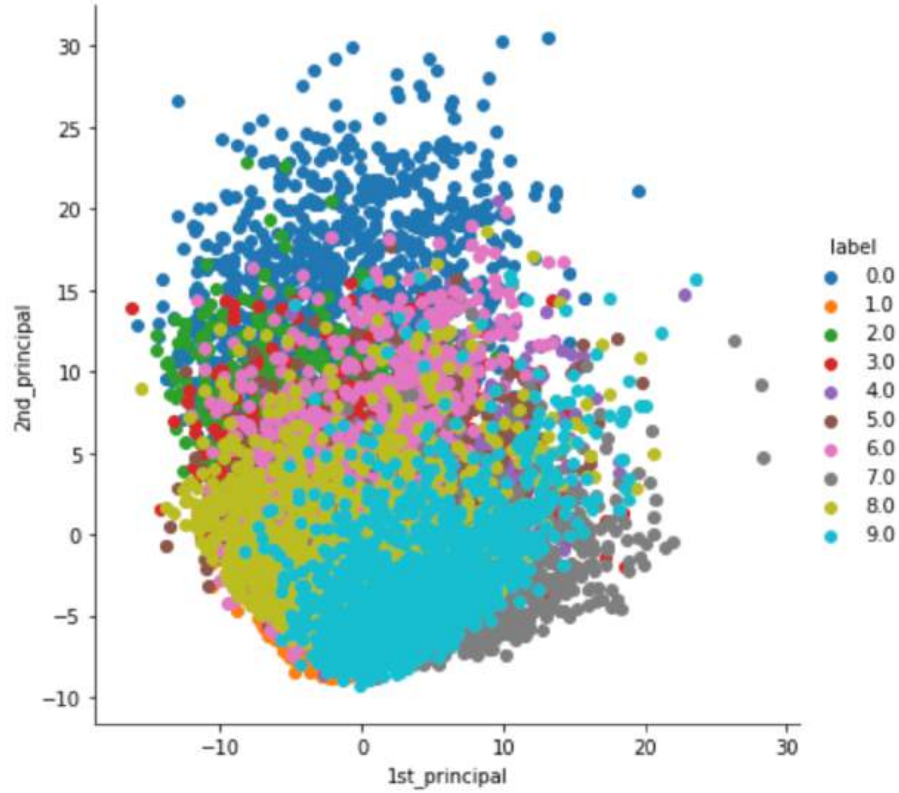
SINCE WE USE THE COVARIANCE MATRIX, IT ASSUMES THAT THE DATA FOLLOWS A **MULTIDIMENSIONAL NORMAL DISTRIBUTION**



LIMITATIONS OF PCA

AND DATA IS NOT ALWAYS GAUSSIAN….

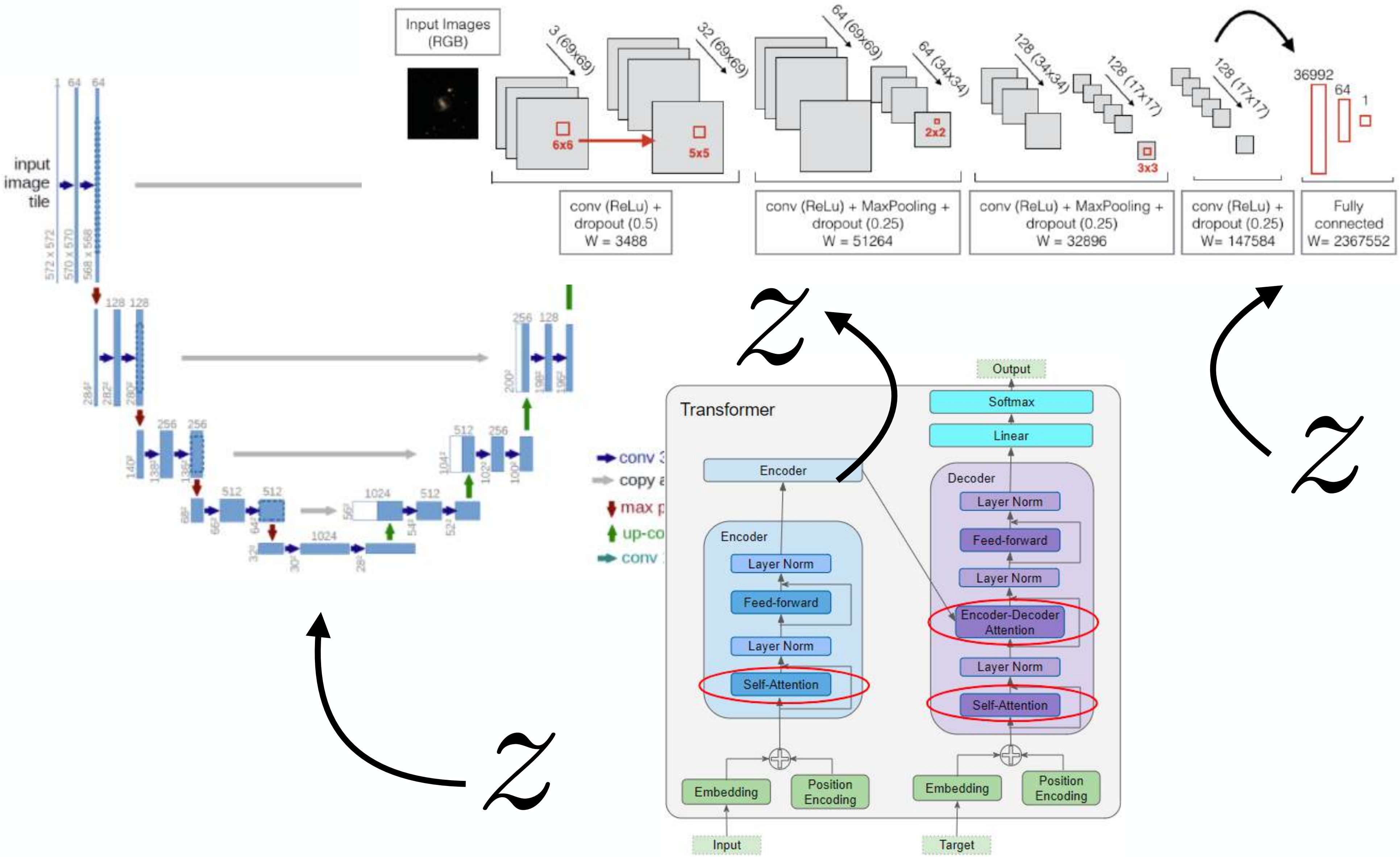




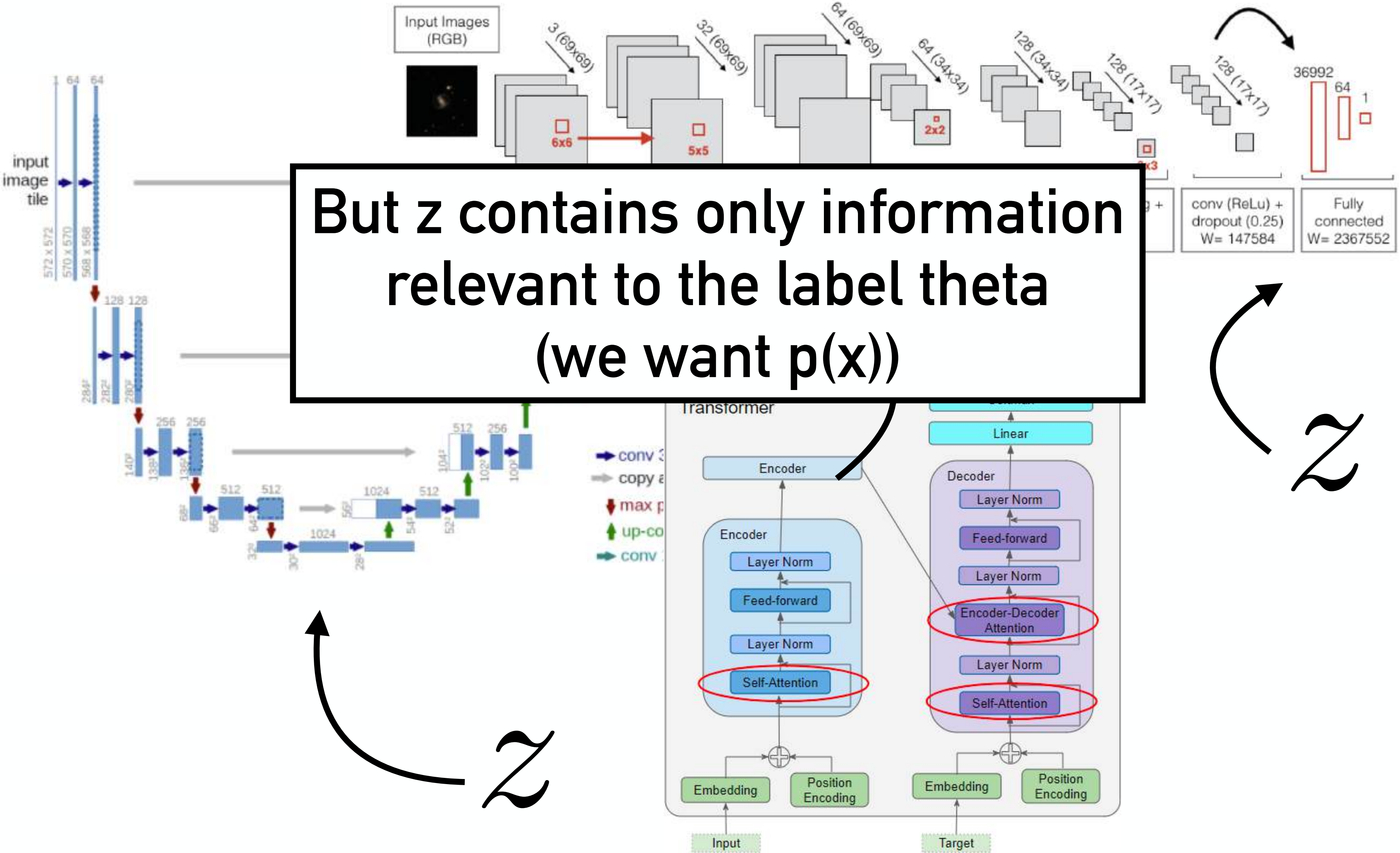
https://medium.com/analytics-vidhya/principal-component-analysis-pca-with-code-on-mnist-dataset-da7de0d07c22

CAN WE GENERALIZE THAT?

Any supervised NN obtains a non linear representation “z”

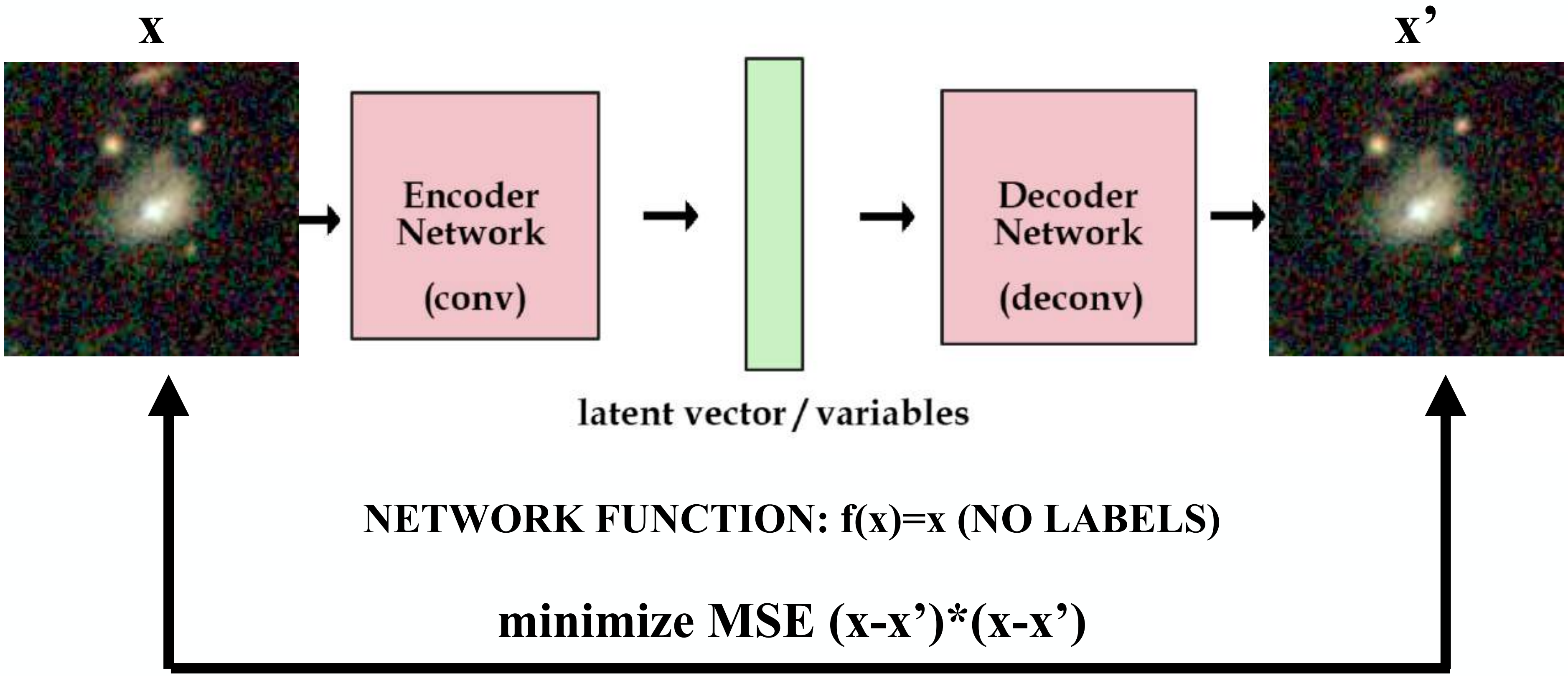


Any supervised NN obtains a non linear representation “z”



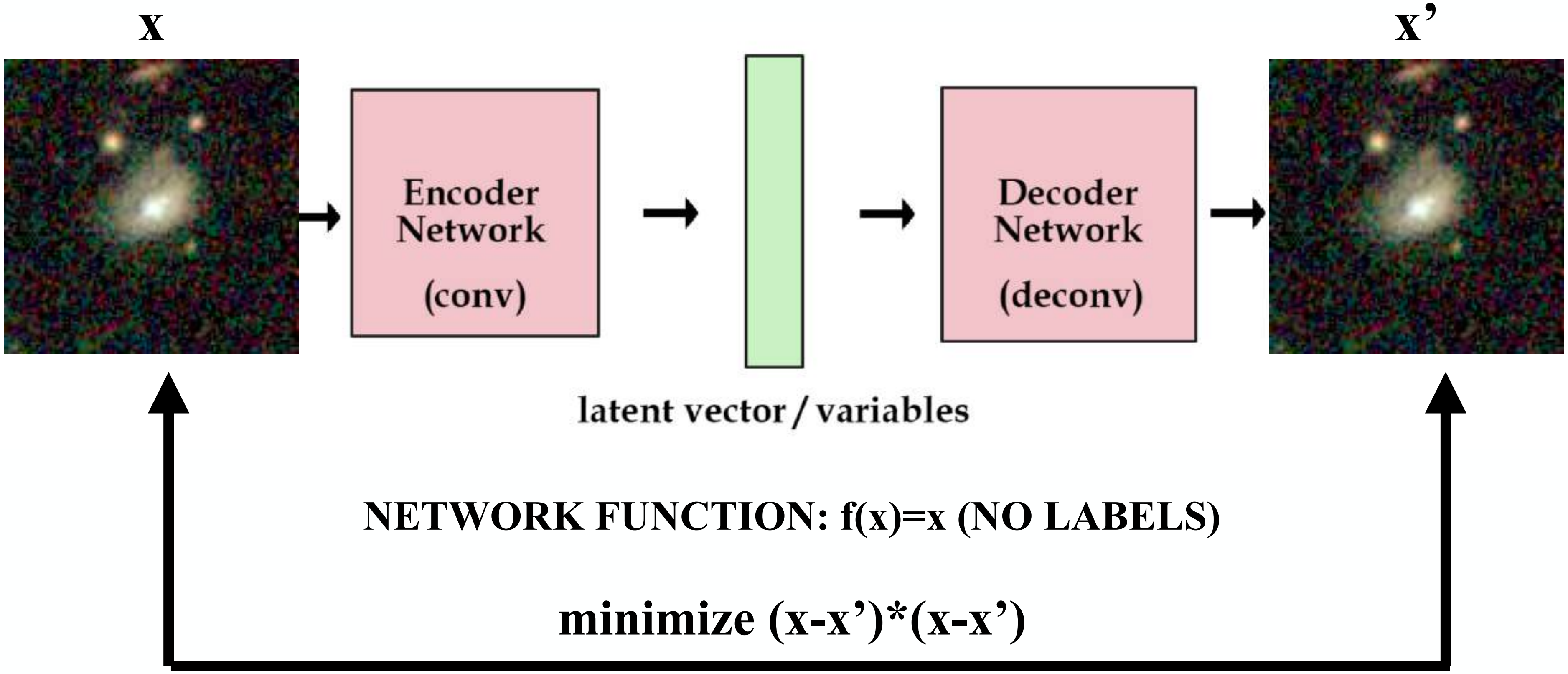
But z contains only information relevant to the label theta (we want p(x))

(CONVOLUTIONAL) AUTO-ENCODER



<u>AN AUTO-ENCODER IS SIMPLY ANY NETWORK WITH IDENTICAL INPUT AND OUTPUT</u>

CONVOLUTIONAL AUTO-ENCODER

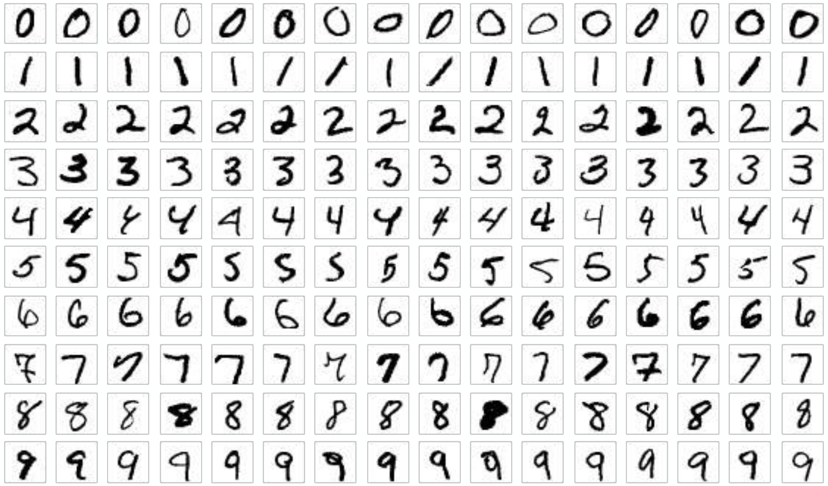


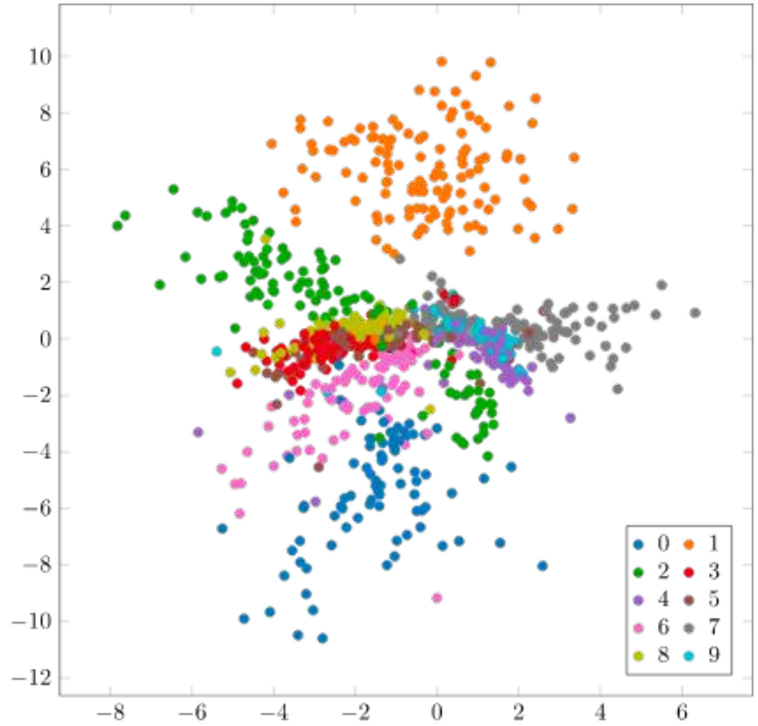
BY REDUCING THE DIMENSIONALITY IN THE LATENT SPACE WE FORCE THE NETWORK TO LEARN A REPRESENTATION OF THE INPUT DATA IN A LOWER DIMENSIONALITY SPACE

* NO NEED TO BE CONVOLUTIONAL - ANY NEURAL NETWORK WITH A BOTTLENECK WILL DO THE JOB

* **QUESTION**: WHAT WOULD HAPPEN IF WE SET AN AUTOENCODER WITH NO ACTIVATION FUNCTIONS?

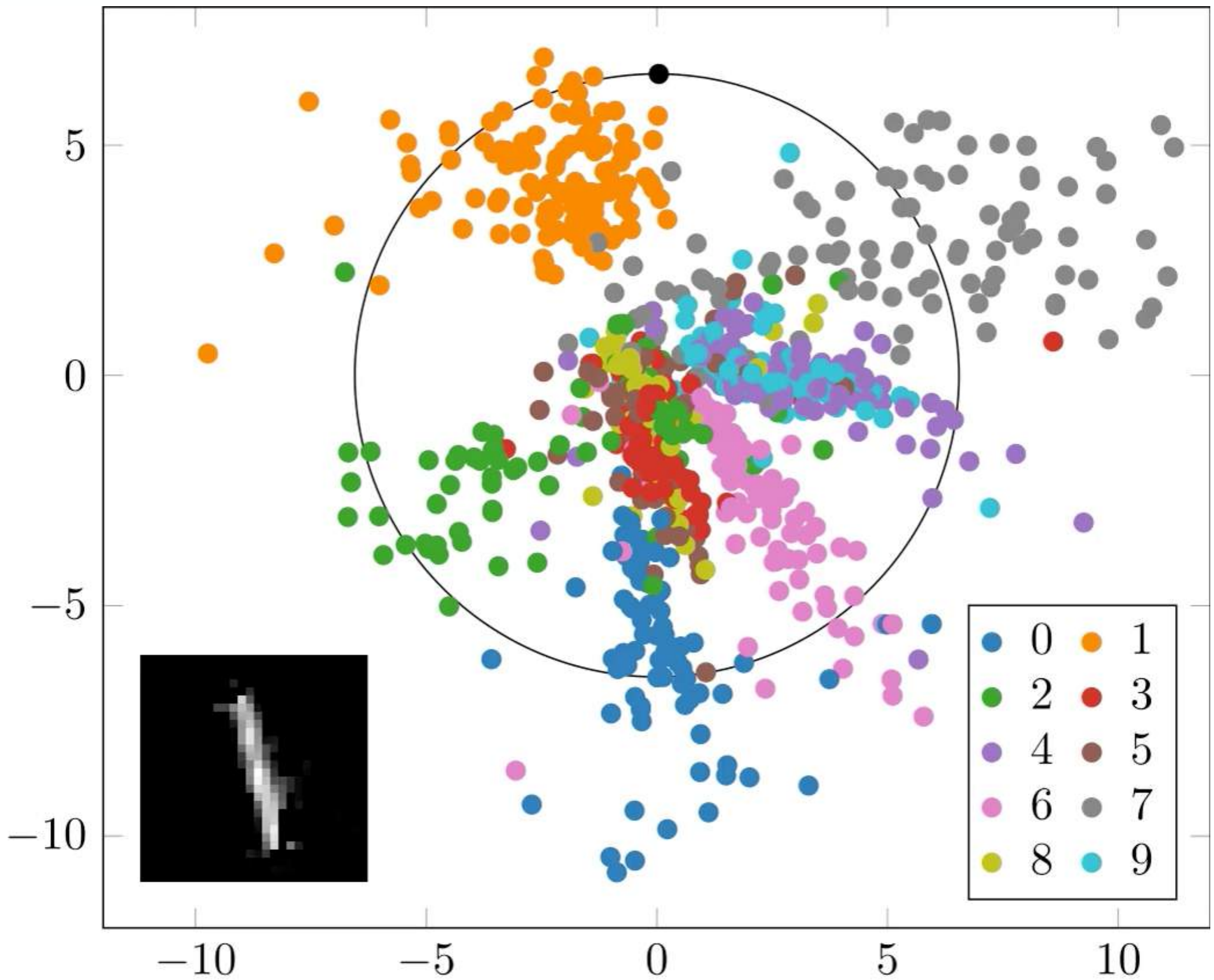
AUTOENCODER REPRESENTATION OF MNIST





https://gertjanvandenburg.com/blog/autoencoder/

AUTOENCODER REPRESENTATION OF MNIST



https://gertjanvandenburg.com/blog/autoencoder/

SELF-SUPERVISED LEARNING

Another alternative is to invent a target to obtain z

$$p_w(\hat{\theta}|X)$$

(Still using a discriminative model, with a general target, and use the latent z as a representation of the data)

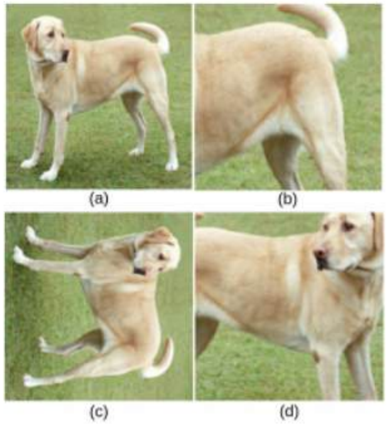
SELF-SUPERVISED LEARNING

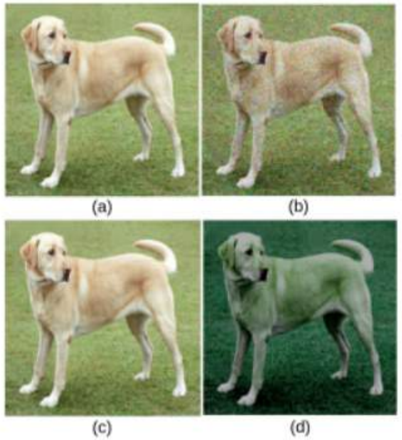
Another alternative is to invent a target to obtain z
(Pretext tasks)

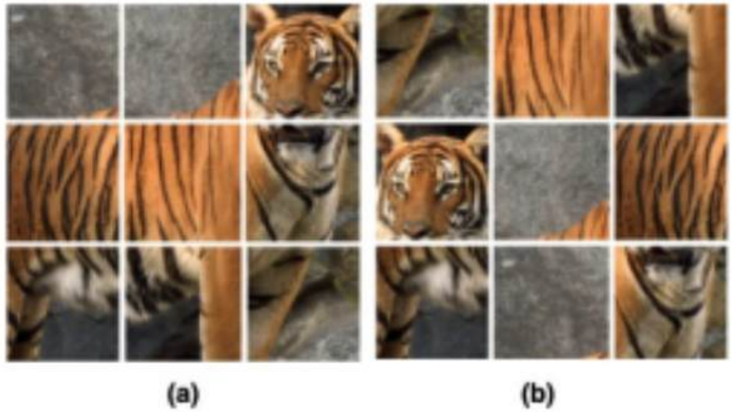
$$p_w(\hat{\theta}|X)$$

Rotation, zoom etc..

(Still using a discriminative model, with a general target, and use the latent z as a representation of the data) e.g. contrastive learning uses the similarity between augmented version as a pretext task







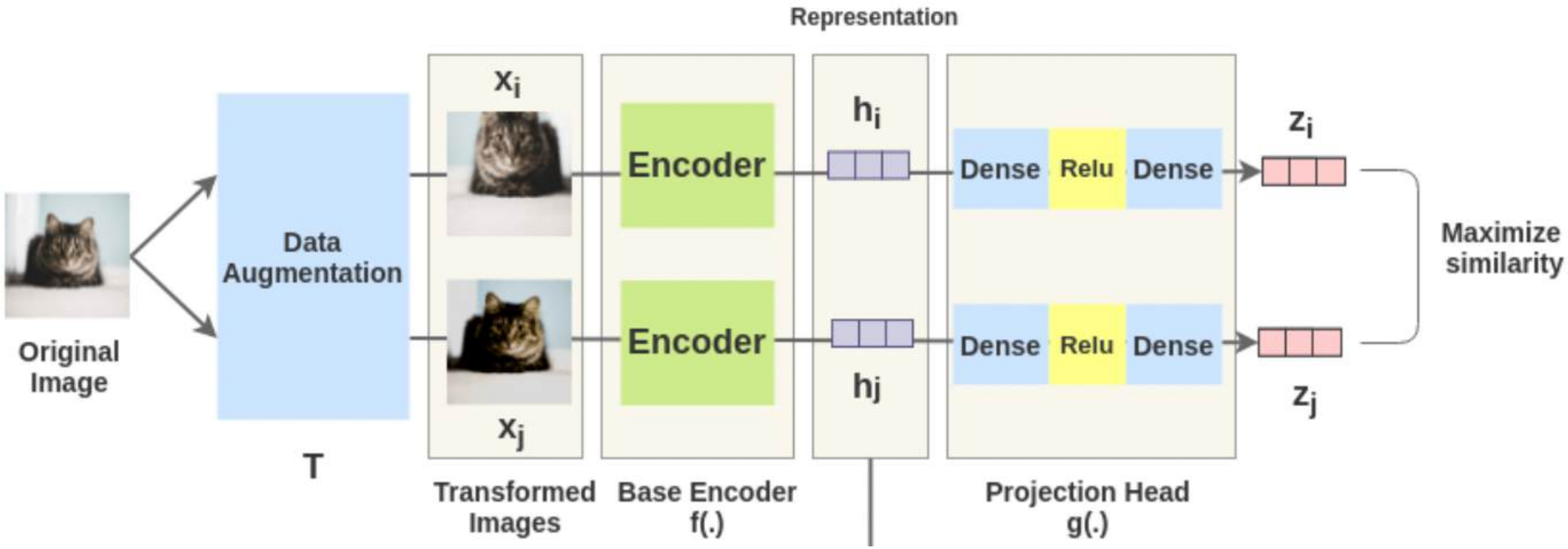
Color Augmentation

Image Rotation

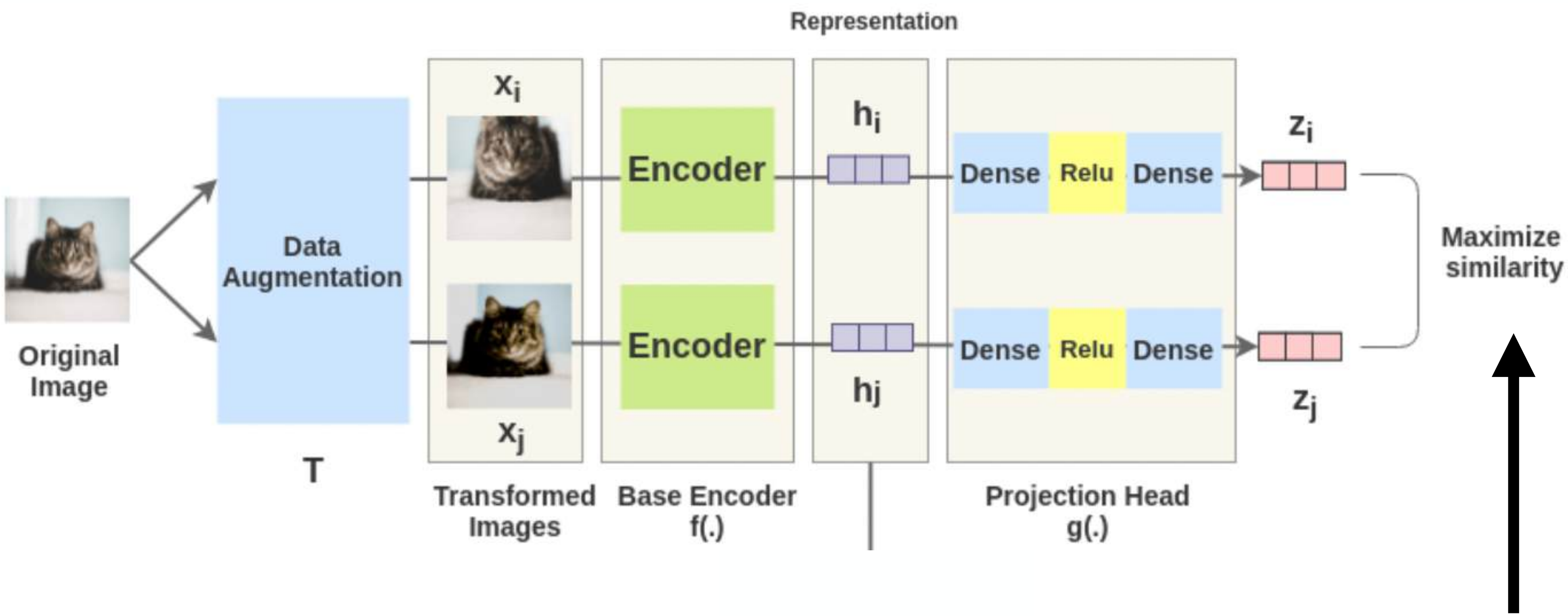
Image Cropping

Any geometrical transformation …

The augmented versions of the images are passed through siamese networks and projected into a latent variable z



The augmented versions of the images are passed through siamese networks and projected into a latent variable z



The key is the loss function

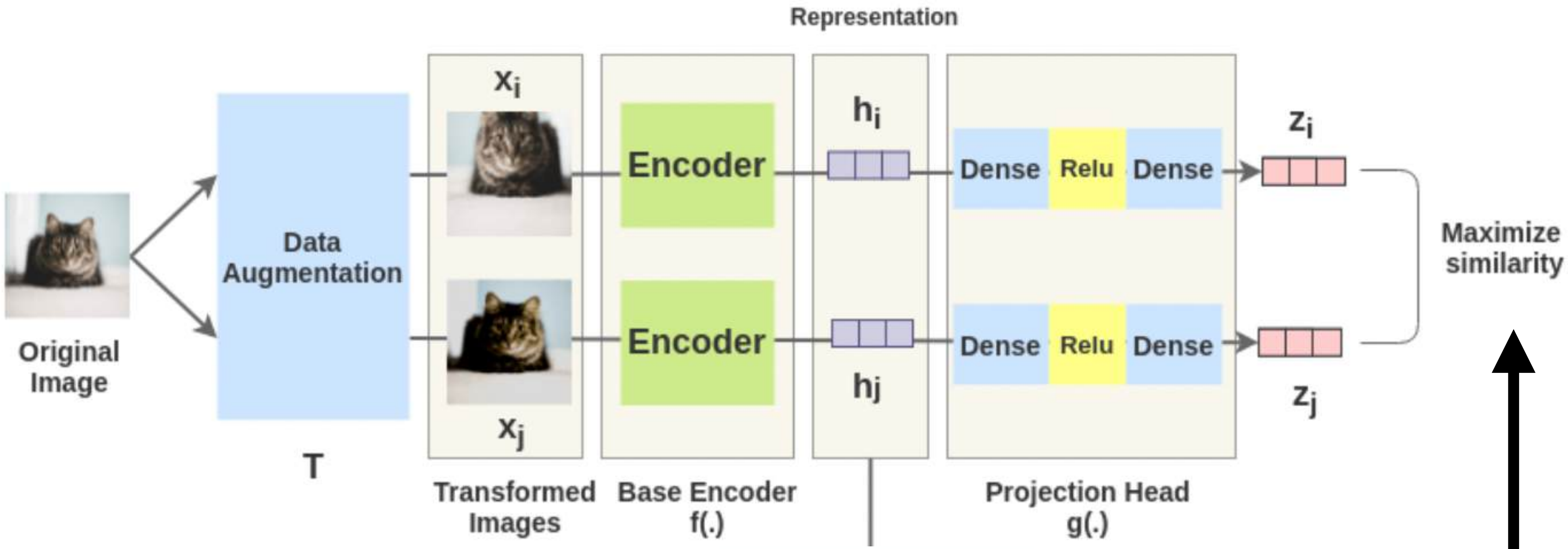
Similarly between
two representations of positive pairs

<u>The contrastive loss:</u>

$$l_{i,j} = -\log \frac{\exp(\langle z_i, z_j \rangle / h)}{\Sigma_{k=1,k\neq i}^{2N} \exp(\langle z_i, z_k \rangle / h)},$$

Sum of all similarities between
negative pairs

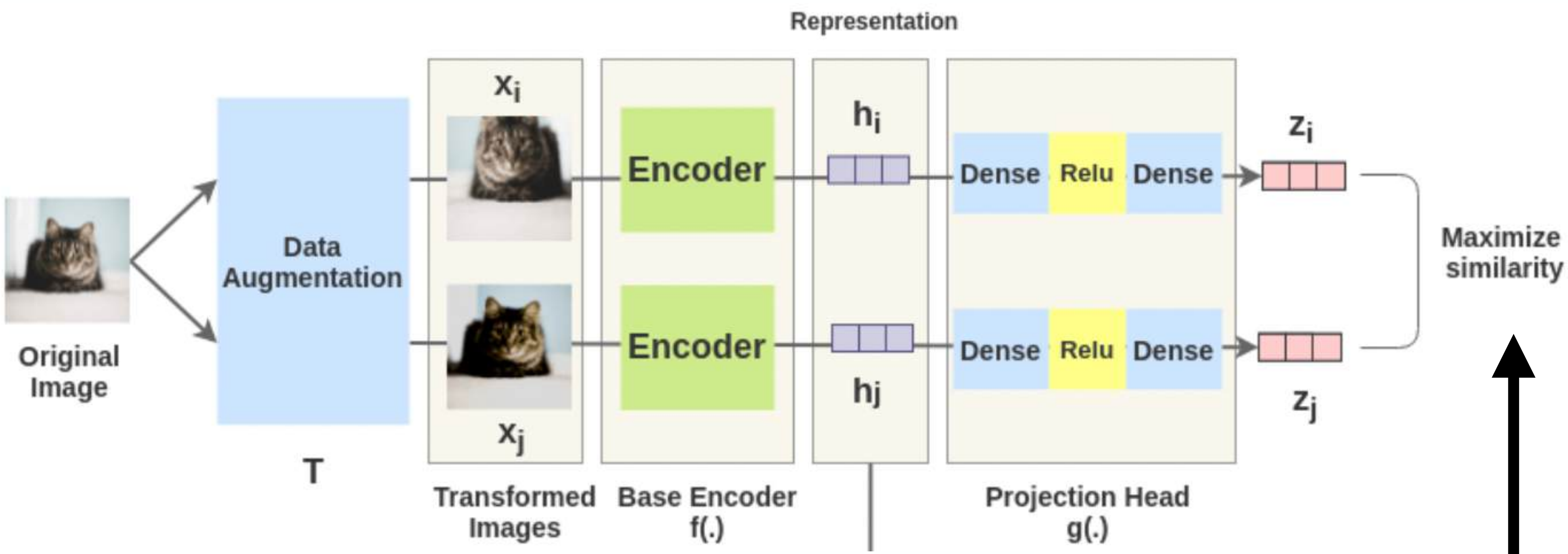
The augmented versions of the images are passed through siamese networks and projected into a latent variable z



z

The key is the loss function

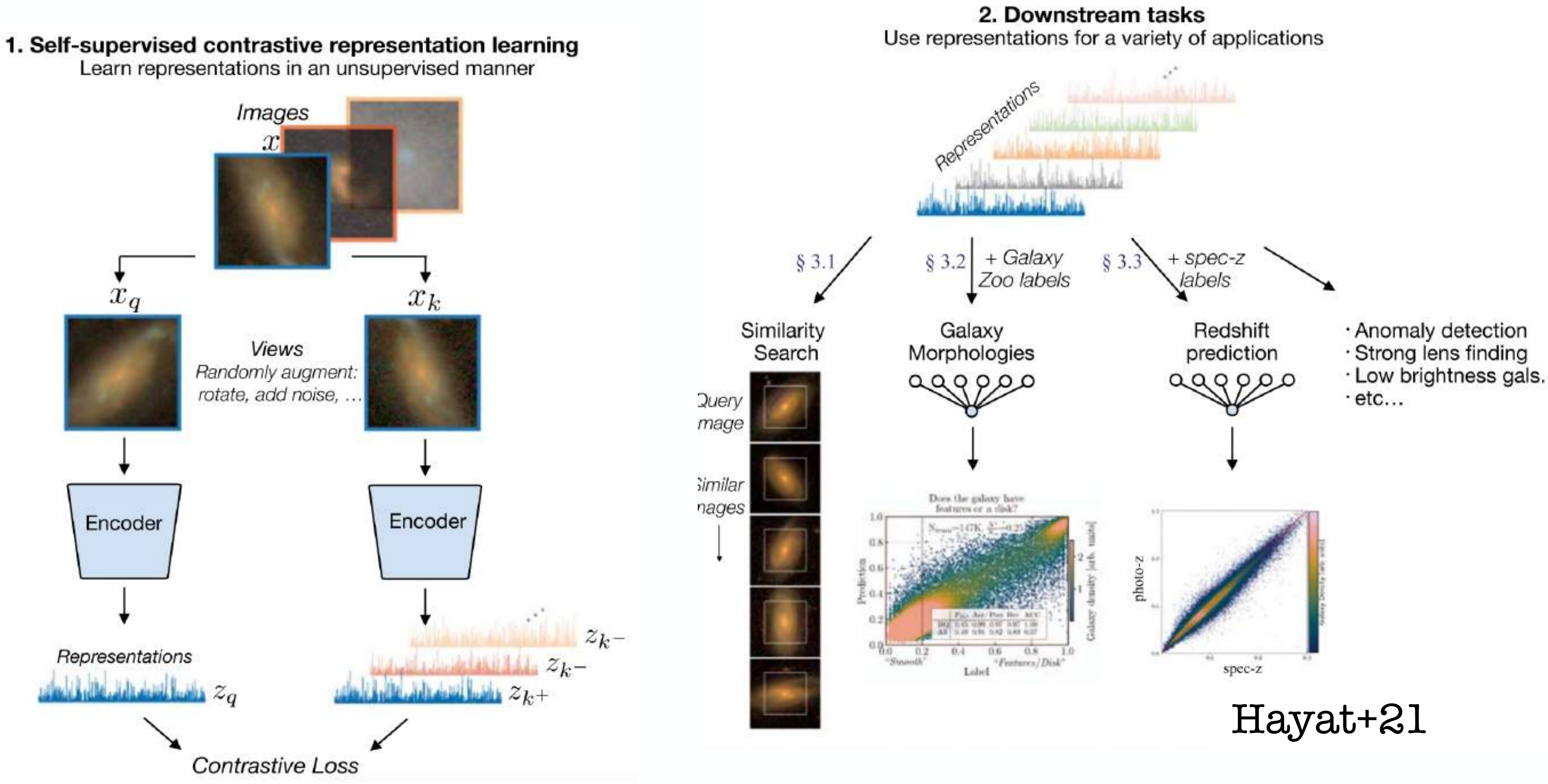
The augmented versions of the images are passed through siamese networks and projected into a latent variable z



$$\mathcal{Z}$$

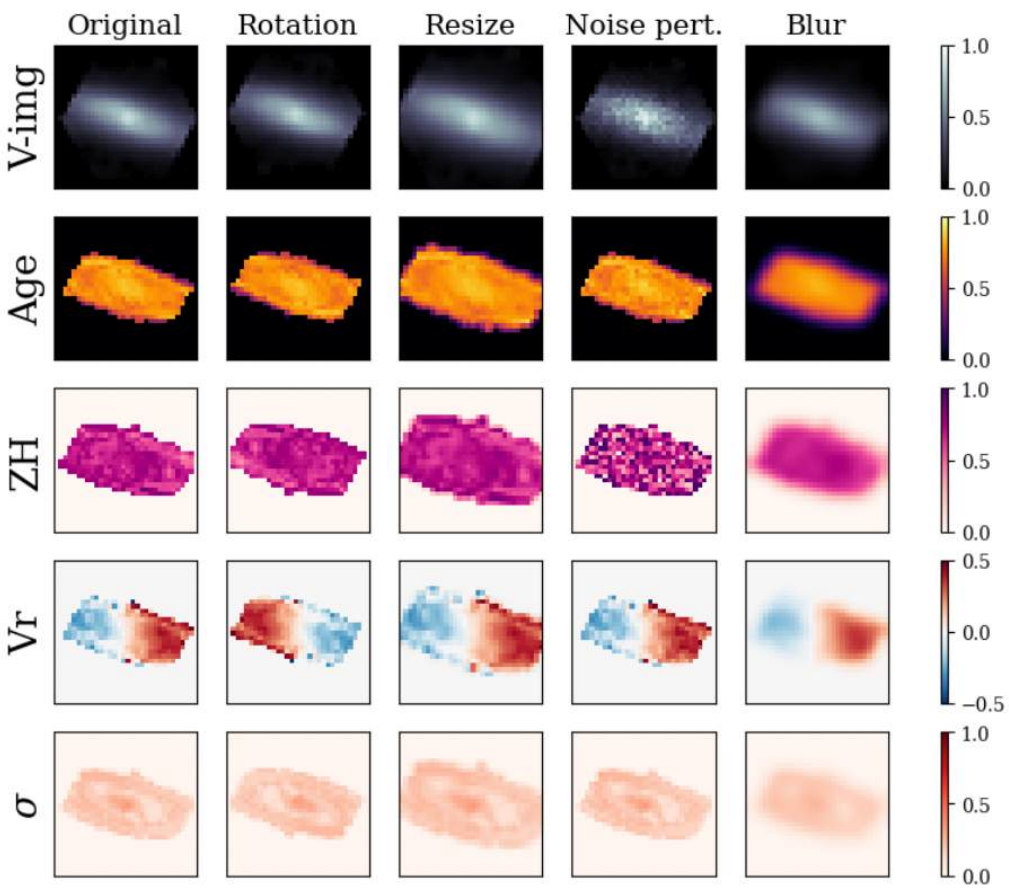
Downstream tasks

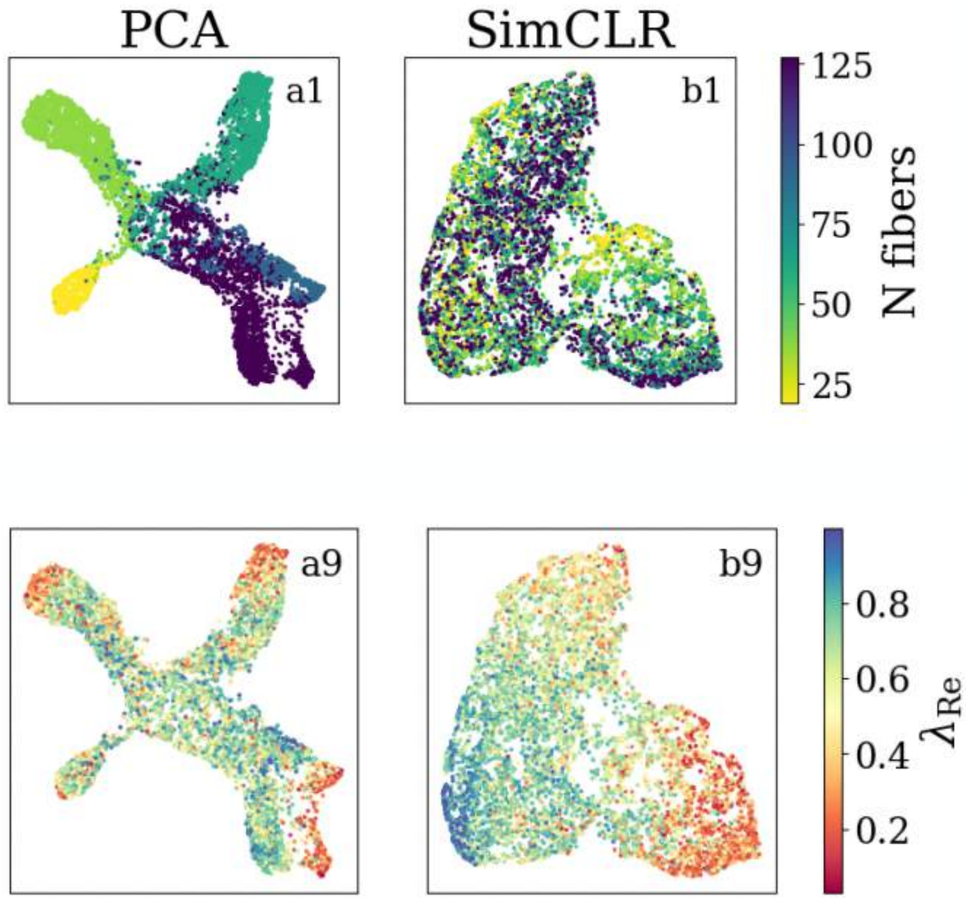
The key is the loss function

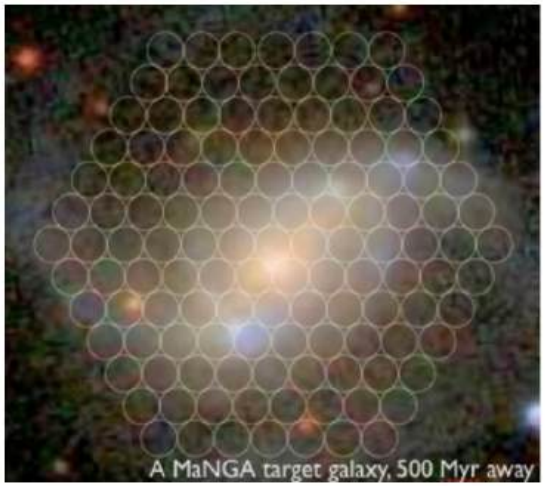


It turns out that the representations learned are very general and can be used for a variety of "downstream tasks"

(Foundation Models) = Model not trained for a specific task, which can be easily fine tuned for a variety of downstream tasks



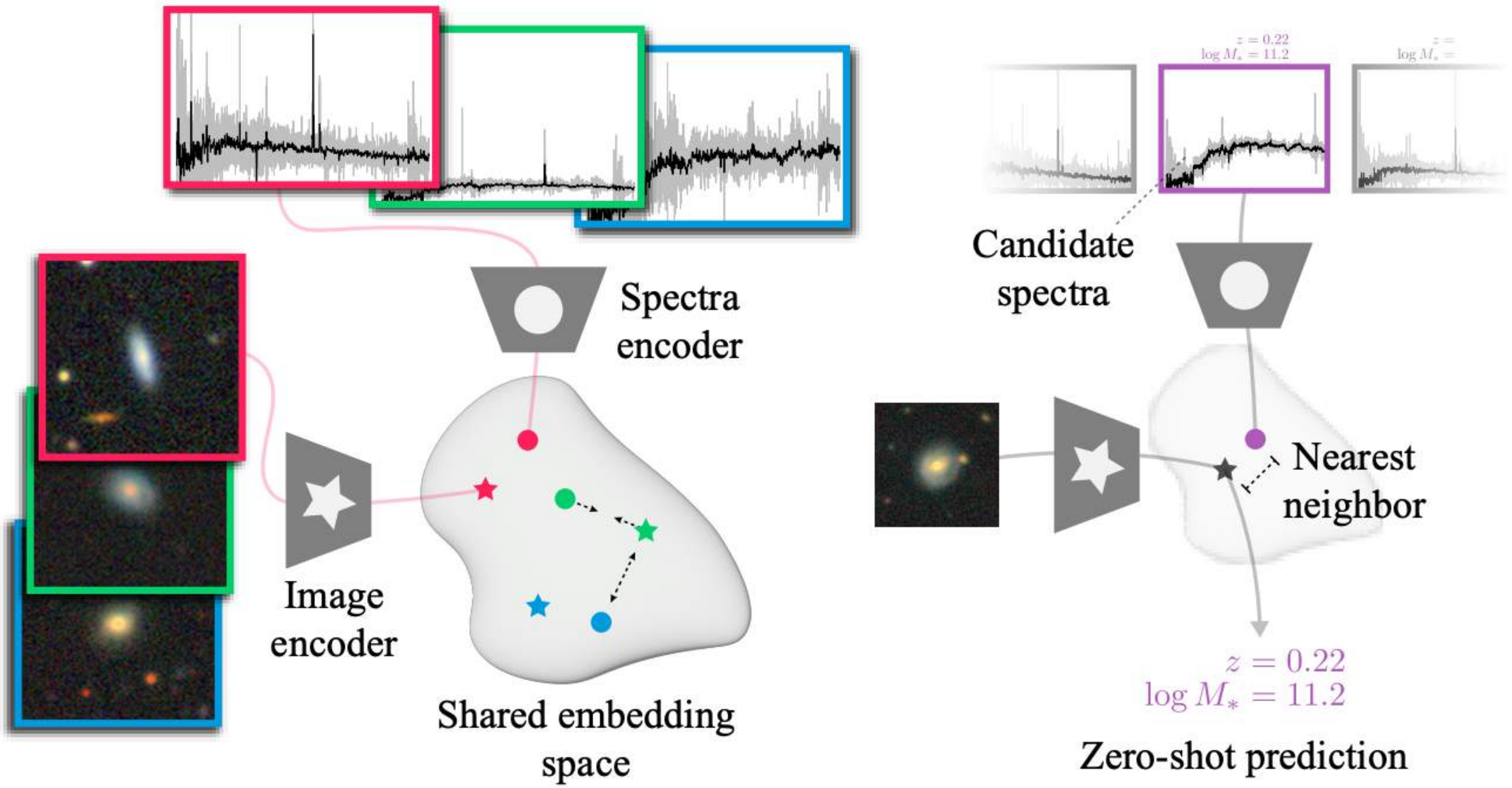




Contrastive learning representation of Manga galaxies

Sarmiento+21

Allows for “straightforward” multimodal representation learning

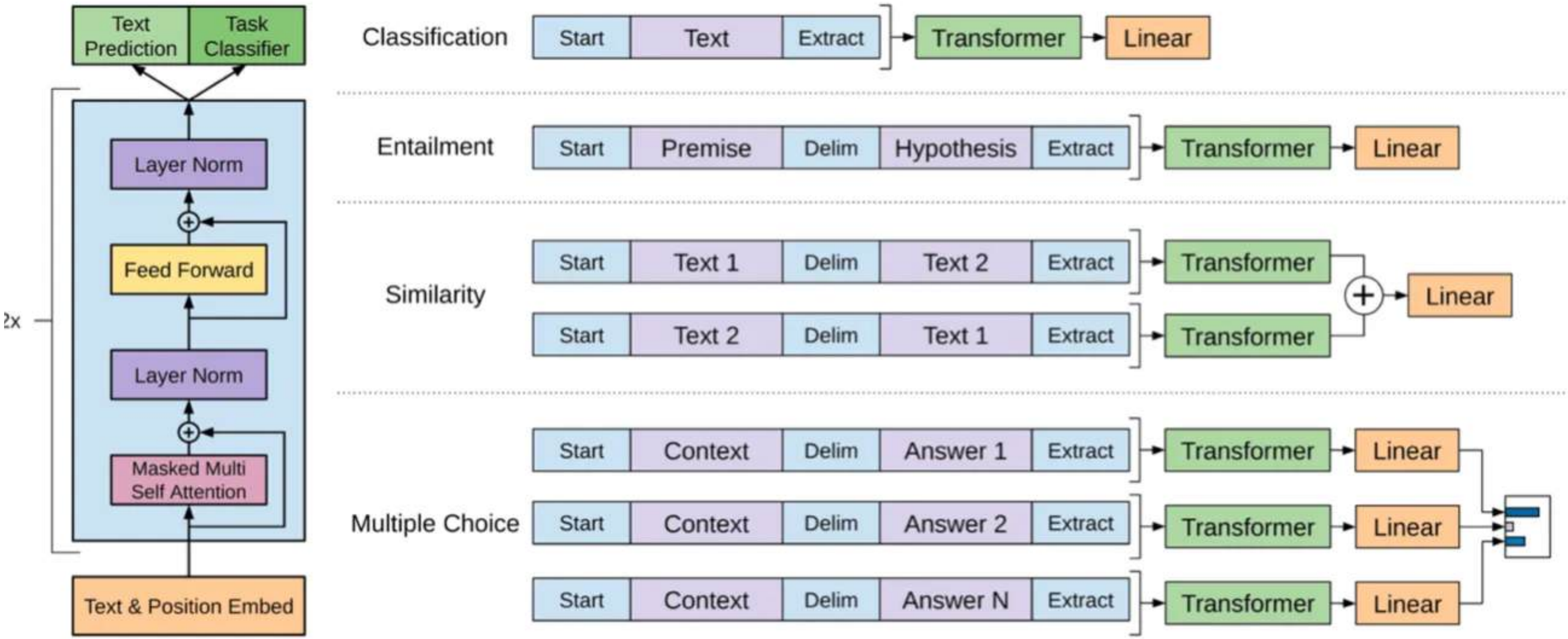


Parker+24

$$l_{i,j} = -\log \frac{\exp(\langle z_i, z_j \rangle / h)}{\Sigma_{k=1,k\neq i}^{2N} \exp(\langle z_i, z_k \rangle / h)},$$

The loss function lives in the latent space

LLMs are Foundation Models



2. Sampling

RECAP5b:

$$p_w(X)$$

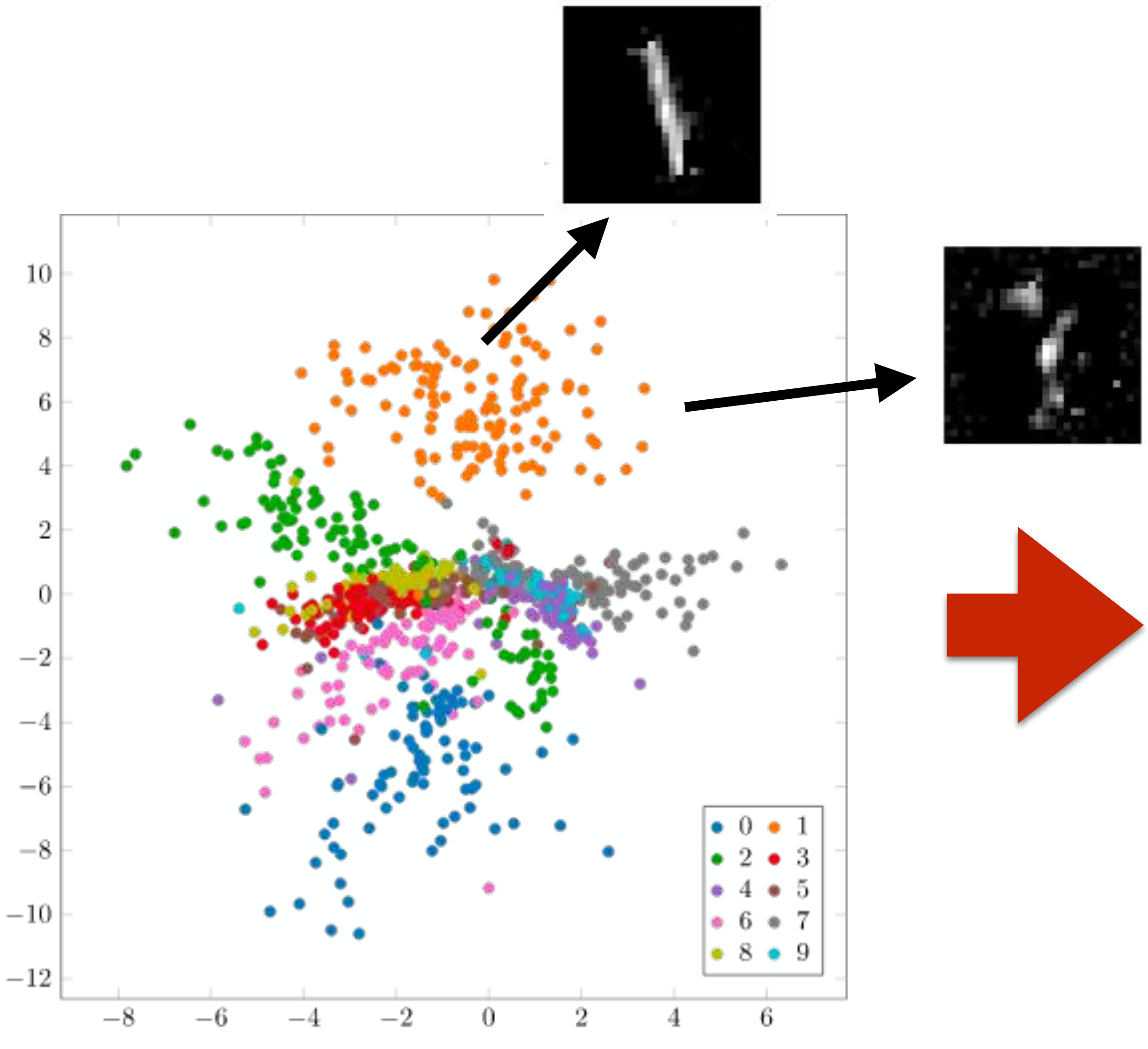
High
dimensional data

Density Estimation

$$X \longrightarrow \boxed{z \longrightarrow p_w(z)}$$

Representation learning /
dimensionality reduction

<u>Latent variable</u>



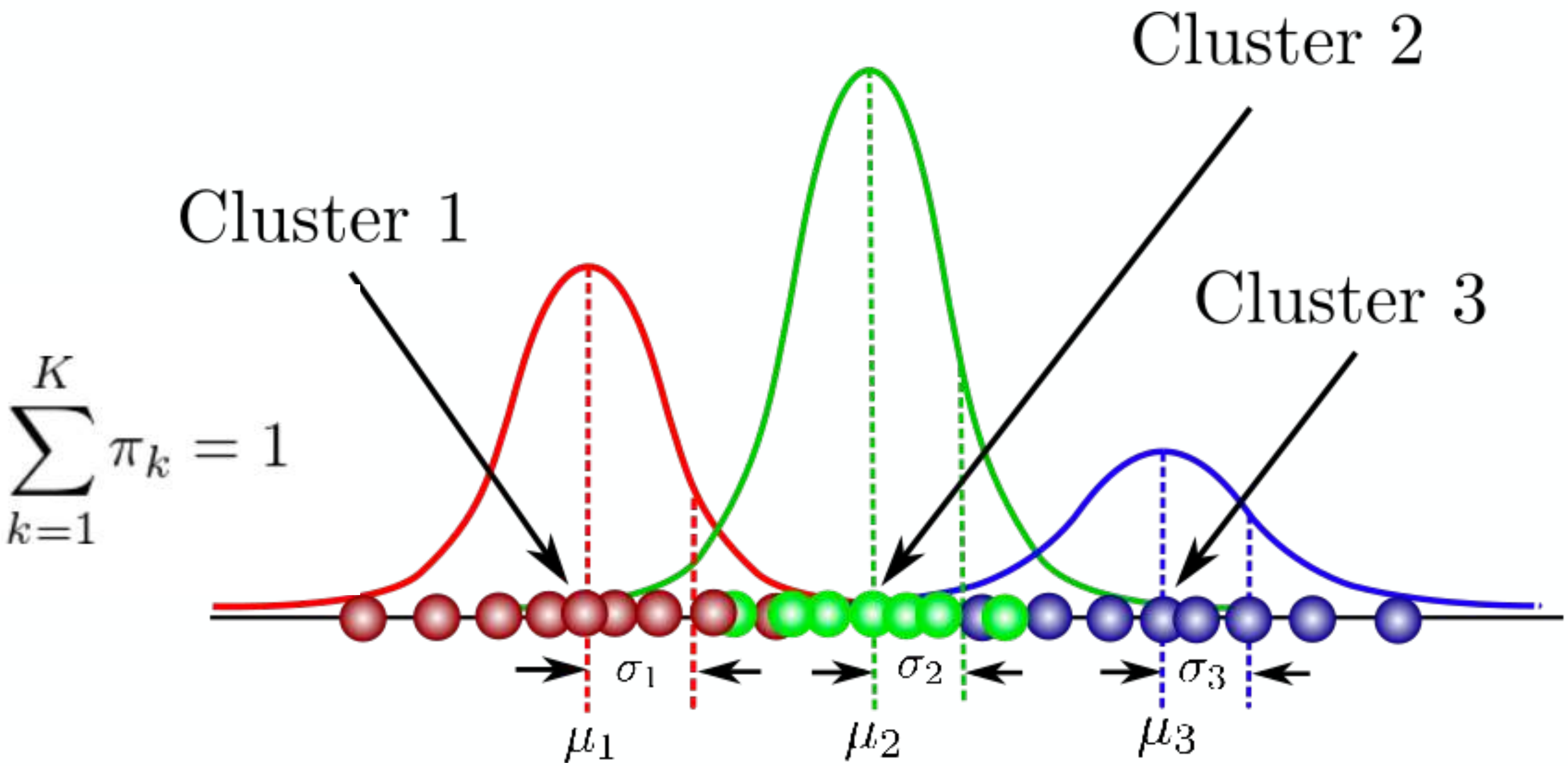
z space (latent space)

HOW CAN I
ESTIMATE P(X)?

(P(z|x))

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

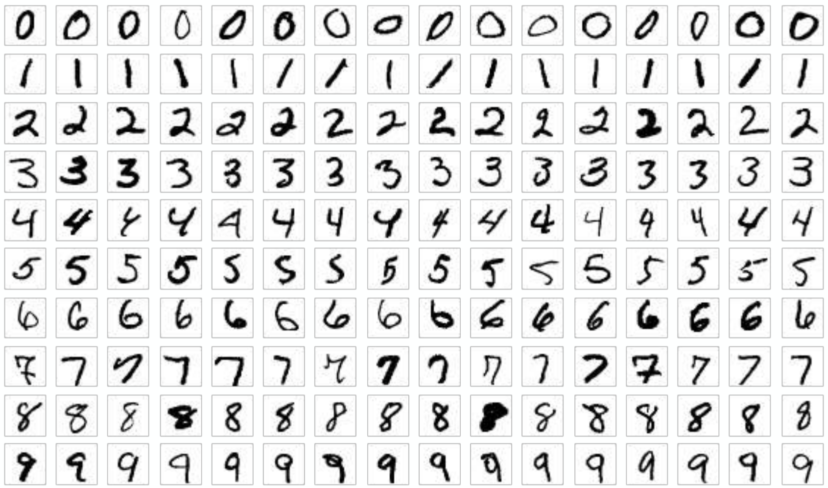
GAUSSIAN MIXTURE MODELS (GMMs) ARE DENSITY ESTIMATOR METHODS THAT FIT MULTIPLE GAUSSIANS TO THE REPRESENTATION

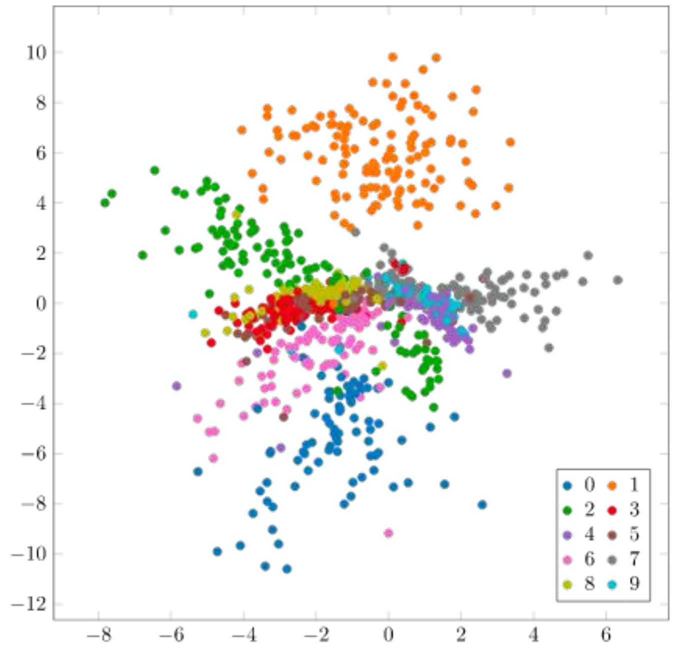


<u>means, sigmas and scale factors of each gaussian are free parameters</u>

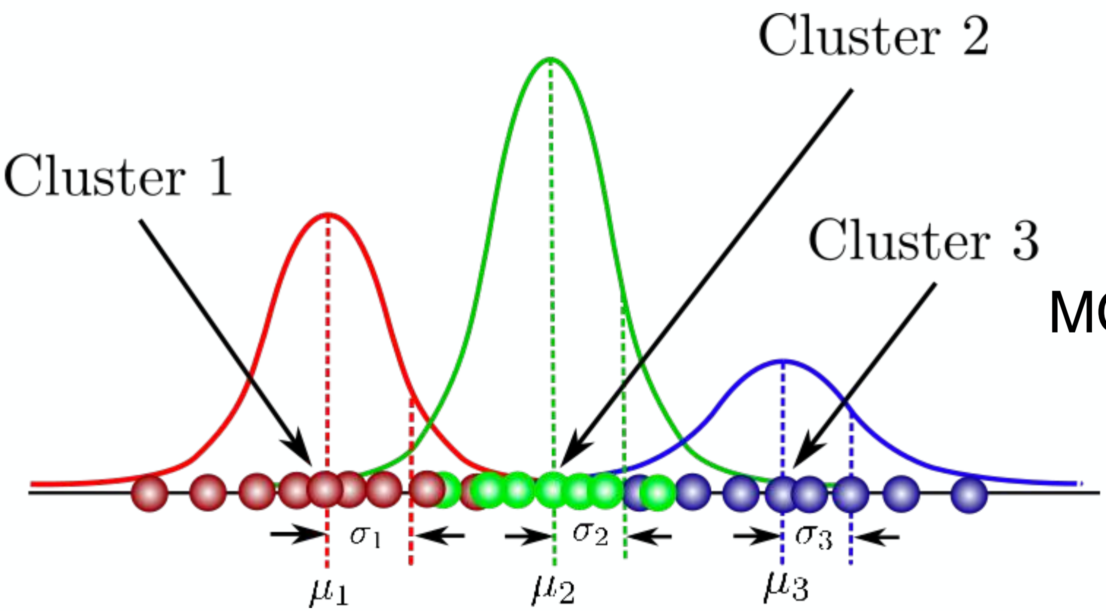
https://towardsdatascience.com/gaussian-mixture-models-explained-6986aaf5a95

DATA





REPRESENTATION
(AUTOENCODER - z)

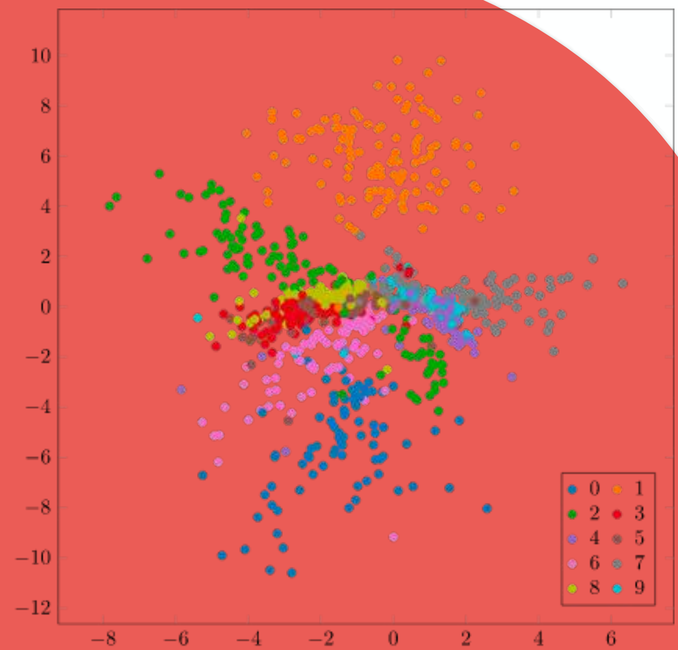


MODELLING OF P(z|x) WITH GMMs

GMMs

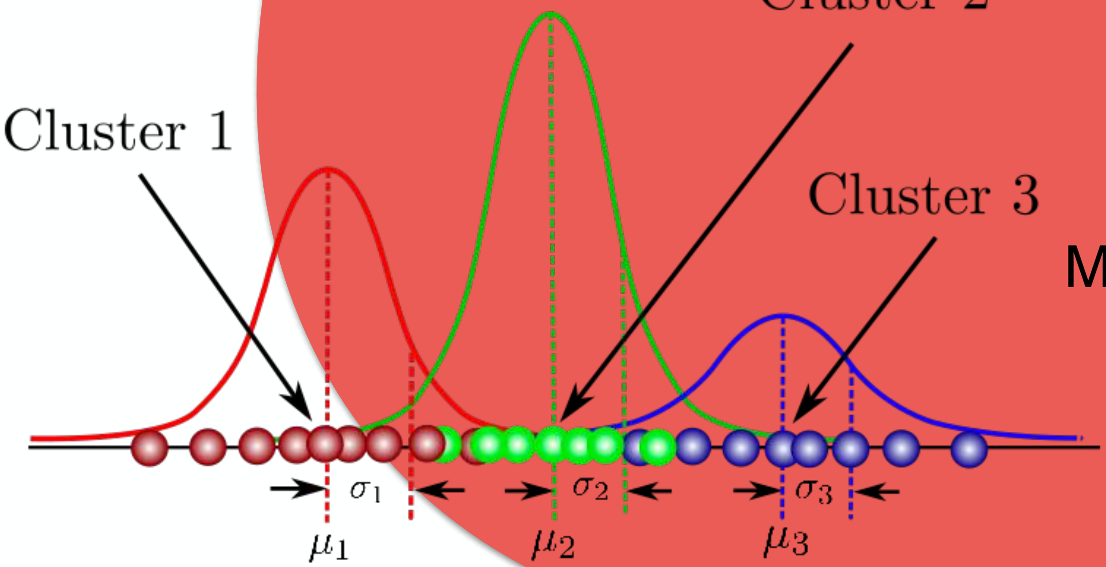
- ++ Both sampling and evaluating are straightforward with GMMs
- — However, performance scales poorly with increasing dimensionality
- — Depends on initial choices…
- And nothing guarantees that the latent space is well behaved as a mixture of gaussians

DATA



REPRESENTATION
(AUTOENCODER)

CAN WE COMBINE THESE 2 STEPS?



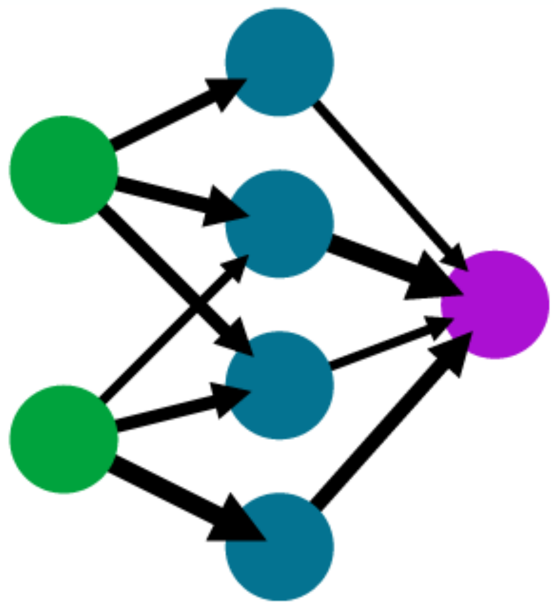
MODELING OF P(z|x) WITH GMMs

Evaluation

High
dimensional data
(e.g. images)

NNs == universal approximators

$$p_w(X)$$



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

(Easy to sample, easy to evaluate)

(Optimization problem)

Likely not
Gaussian

Sampling

<u>* Not addressing score based models (diffusion) in this lecture</u>

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)

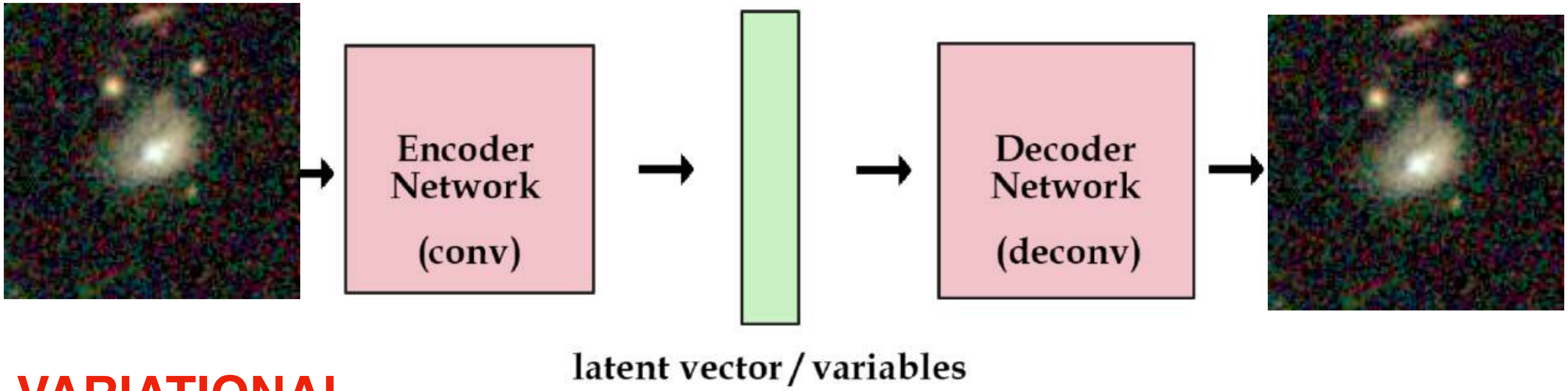
GAN

(GENERATIVE ADVERSARIAL NETOWRK)

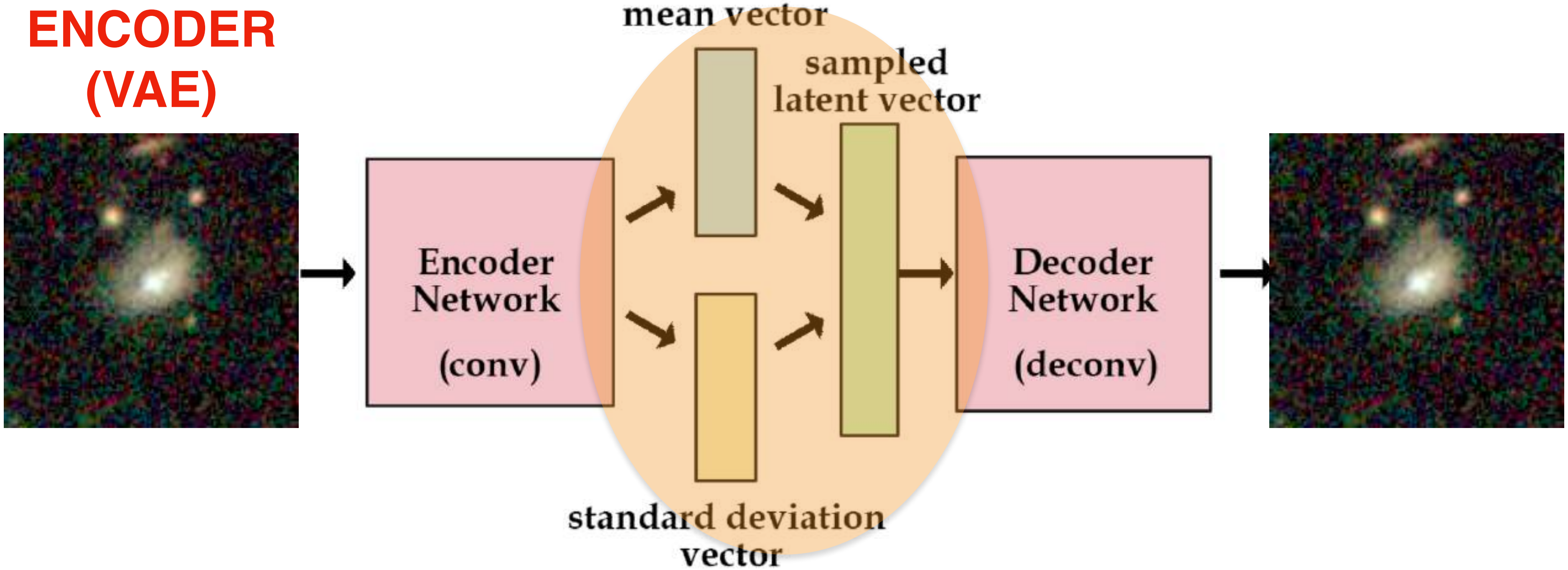
NF-ARF

(NORMALIZING FLOW, AUTOREGRESSIVE FLOW)

AUTO-ENCODER

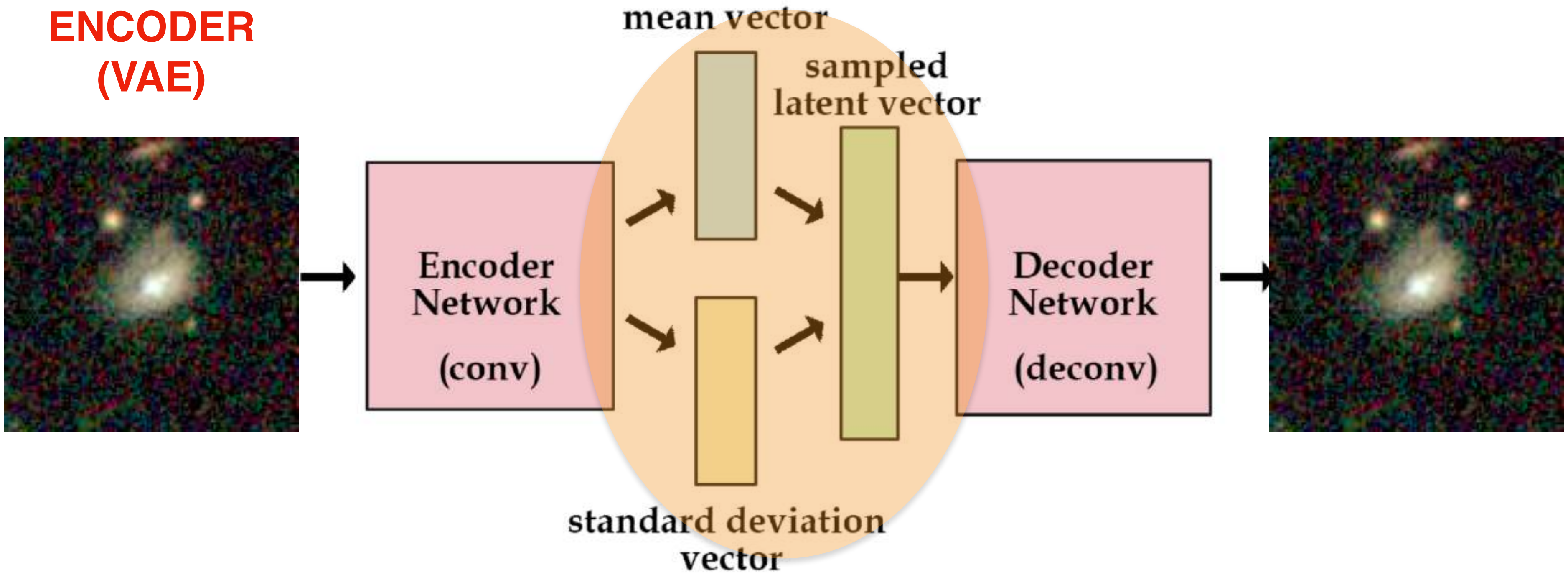


VARIATIONAL AUTO-ENCODER (VAE)



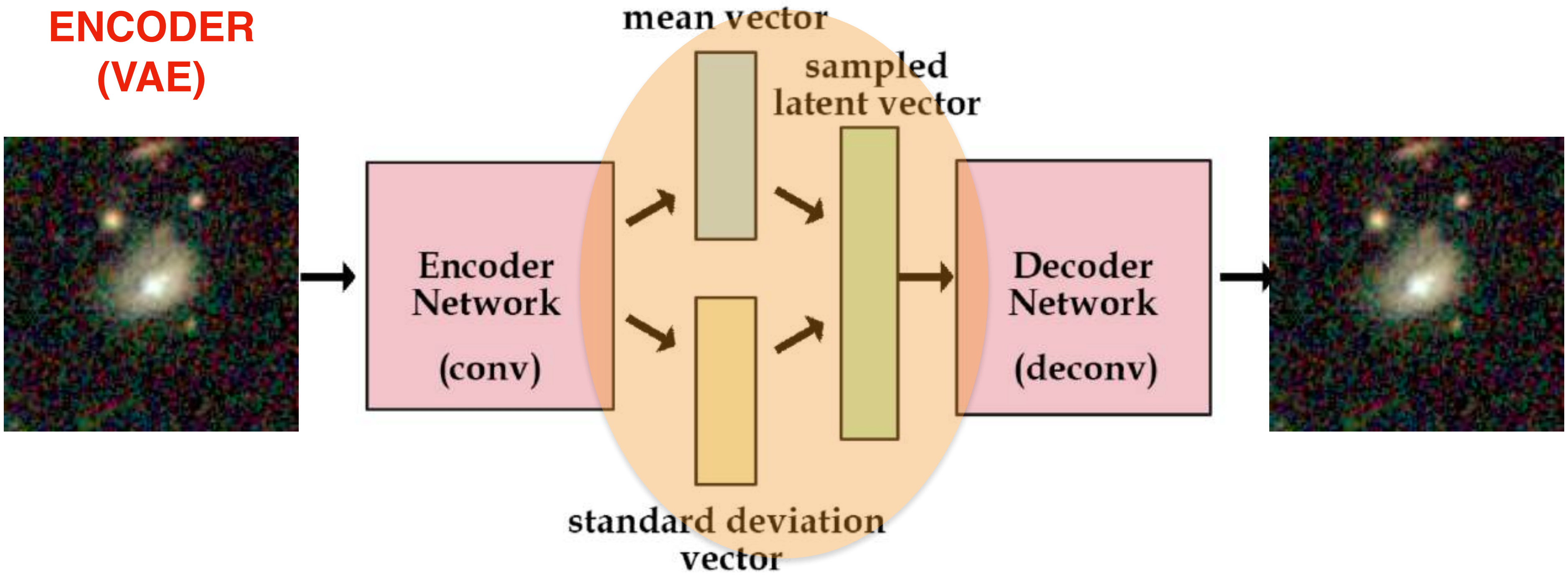
LET'S MODEL THE LATENT SPACE WITH A MIXTURE OF GAUSSIANS

VARIATIONAL AUTO-ENCODER (VAE)



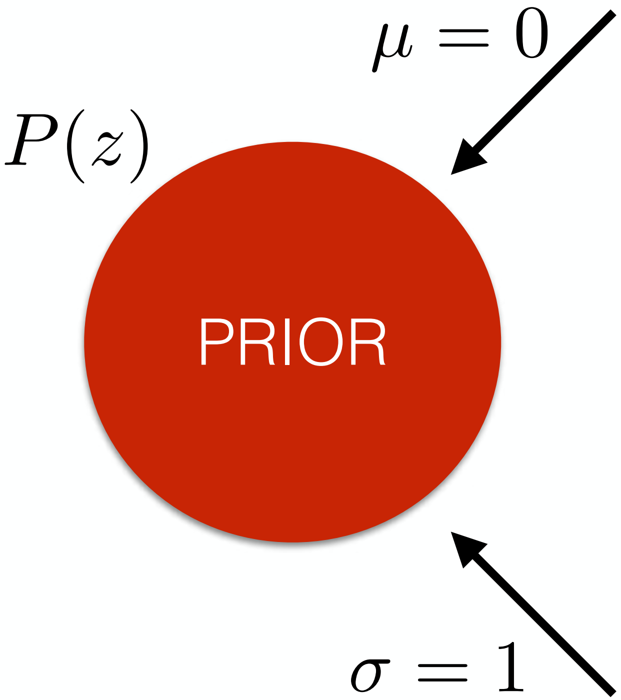
HOWEVER, NOTHING GUARANTEES US THAT P(z) CAN BE ACCURATELY MODELLED BY A MIXTURE OF GAUSSIANS….

VARIATIONAL AUTO-ENCODER (VAE)

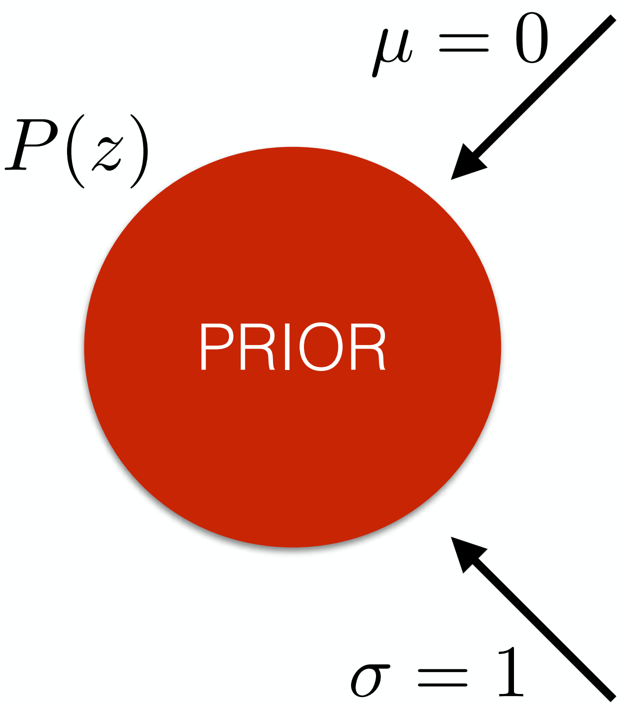


HOWEVER, NOTHING GUARANTEES US THAT THE LATENT SPACE CAN BE MODELLED BY A MIXTURE OF GAUSSIANS….

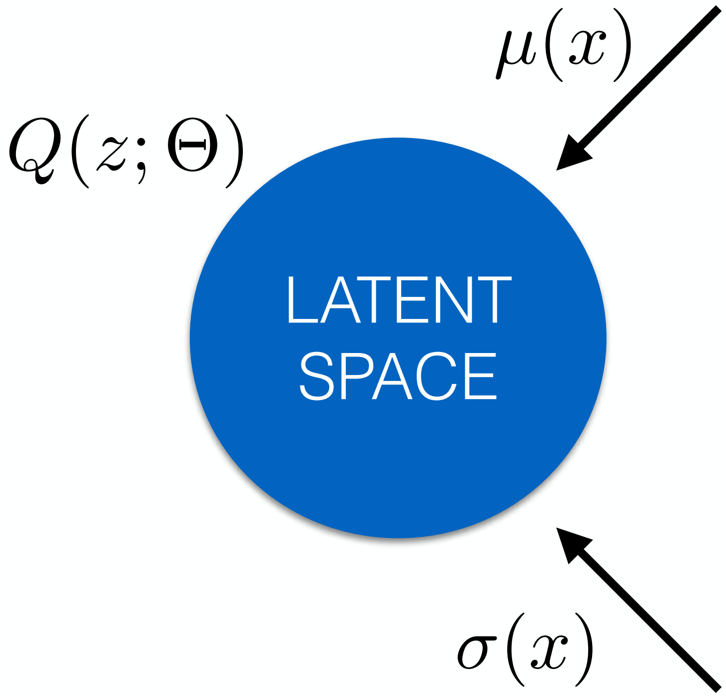
… LET'S FORCE IT TO BE GAUSSIAN LIKE!



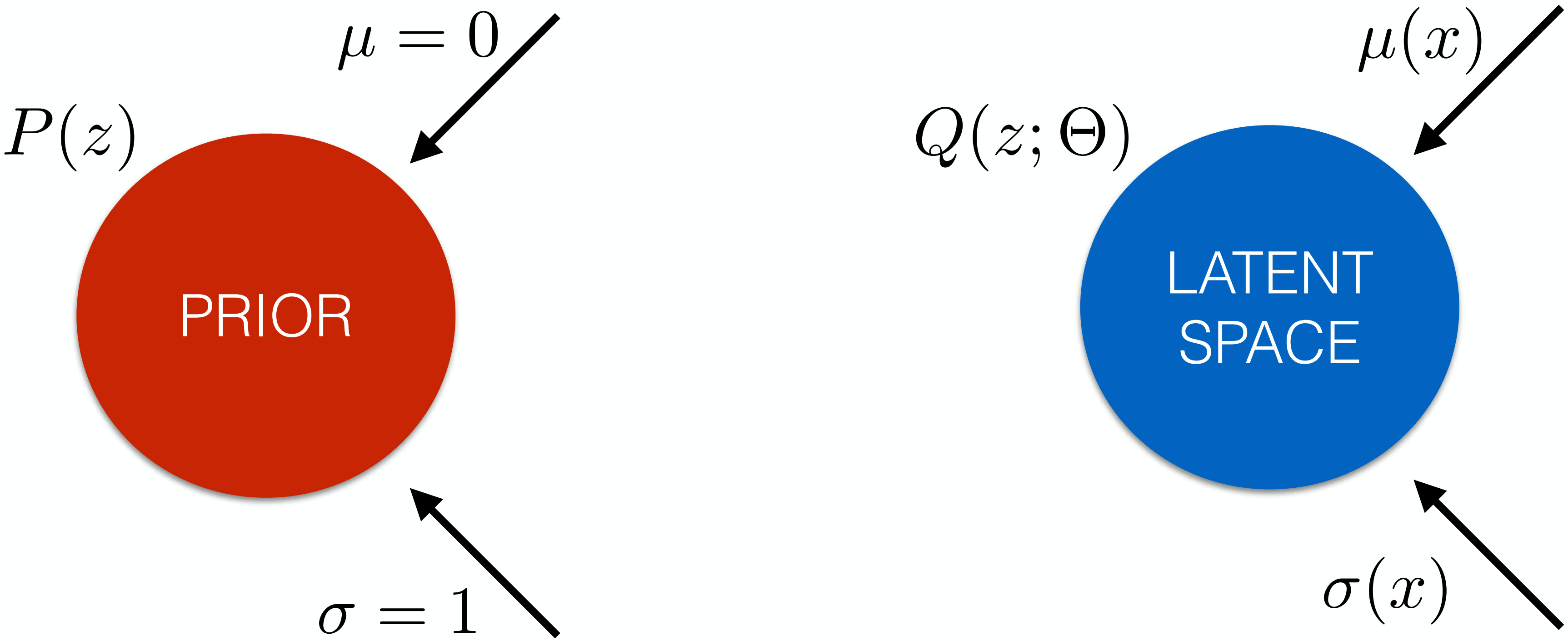
WE ASSUME A SIMPLE PRIOR



WE ASSUME A SIMPLE PRIOR

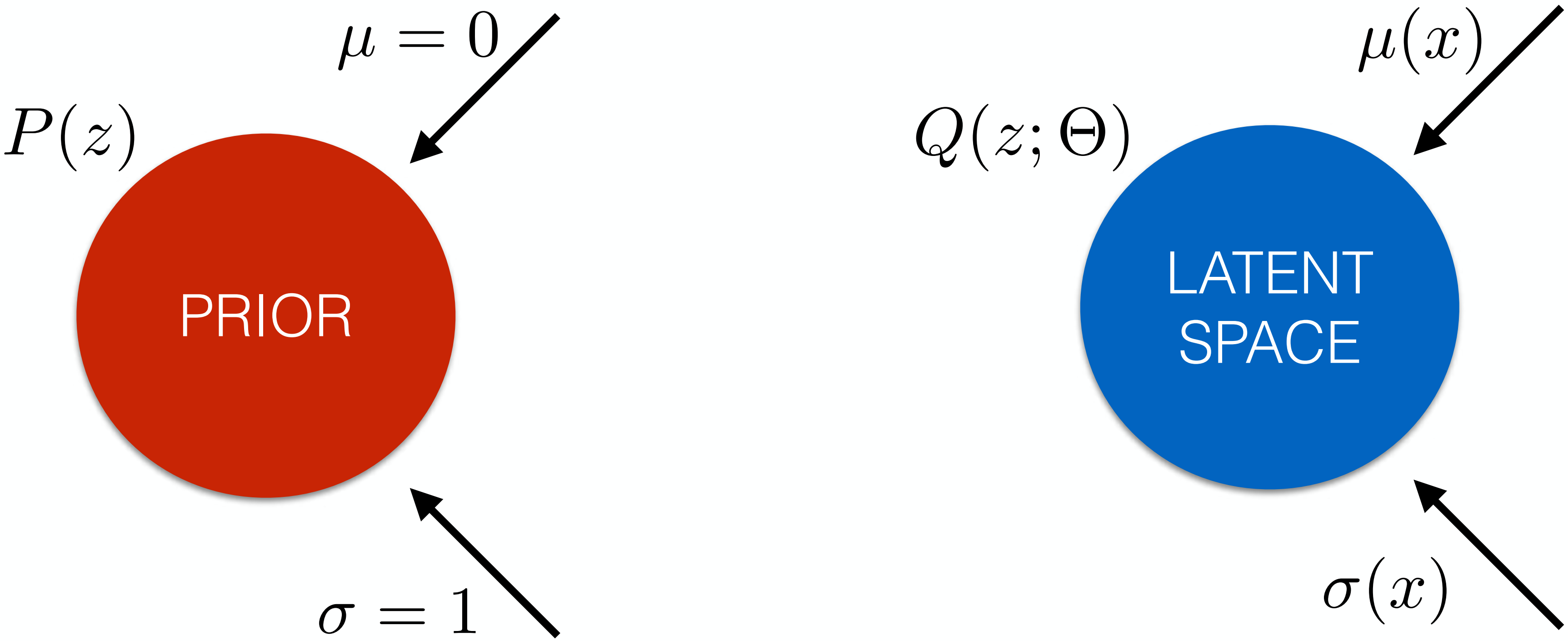


LATENT SPACE MODELING



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...

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

~MSE

REGULARIZATION TERM

~Gaussian

WHAT WOULD BE THEN THE LOSS FUNCTION OF A VAE?

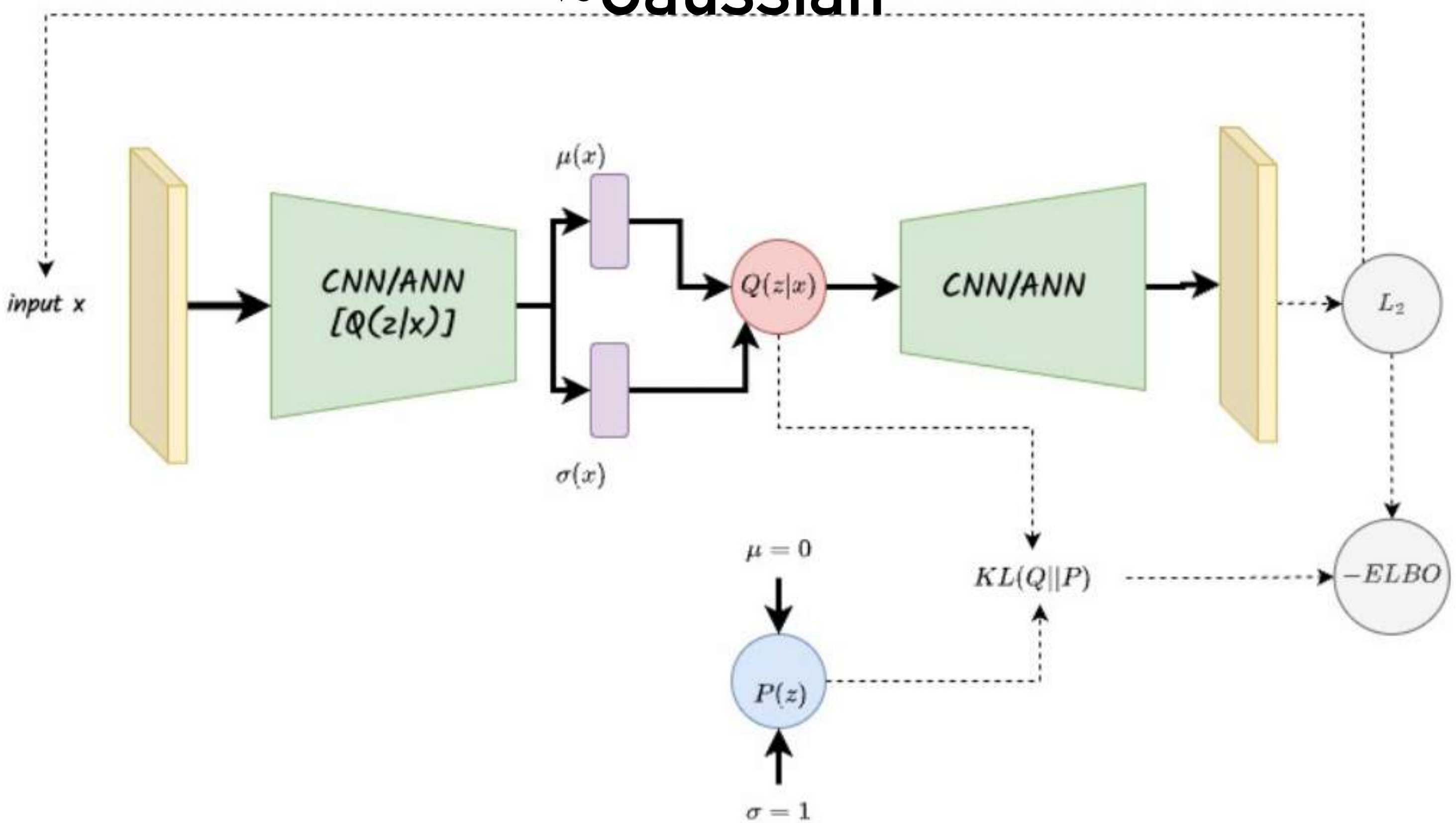
The key insight of VAE is that we are actually performing **variational inference** here, which then tells us what the loss function should be...

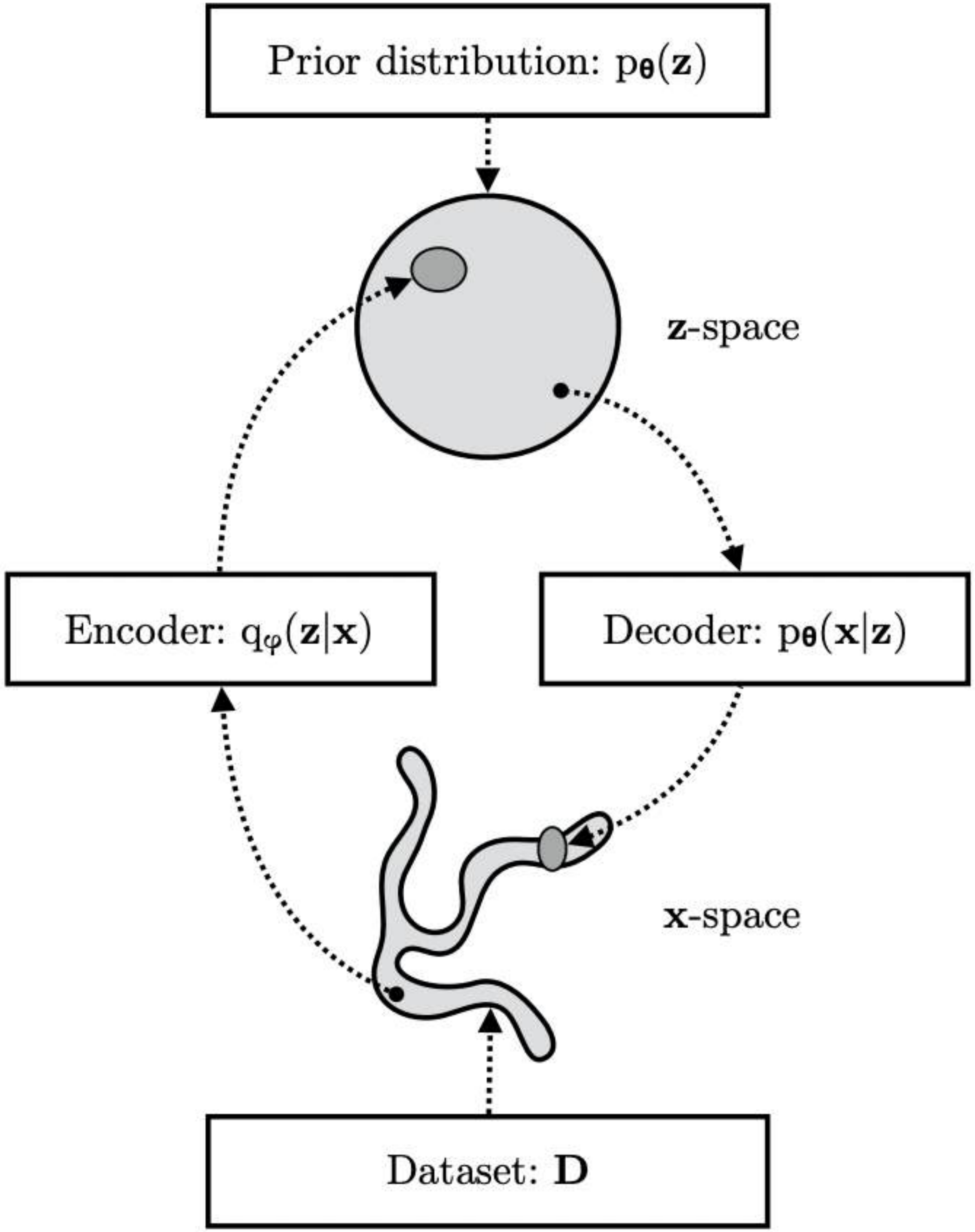
$$-\mathrm{ELBO} = \langle \log P(\mathbf{x} \mid \mathbf{z}) \rangle_{z \sim Q} + \mathrm{KL}\left(Q(\mathbf{z}; \Theta) \parallel P(\mathbf{z})\right) \; ,$$

L2 LOSS

REGULARIZATION TERM

~Gaussian





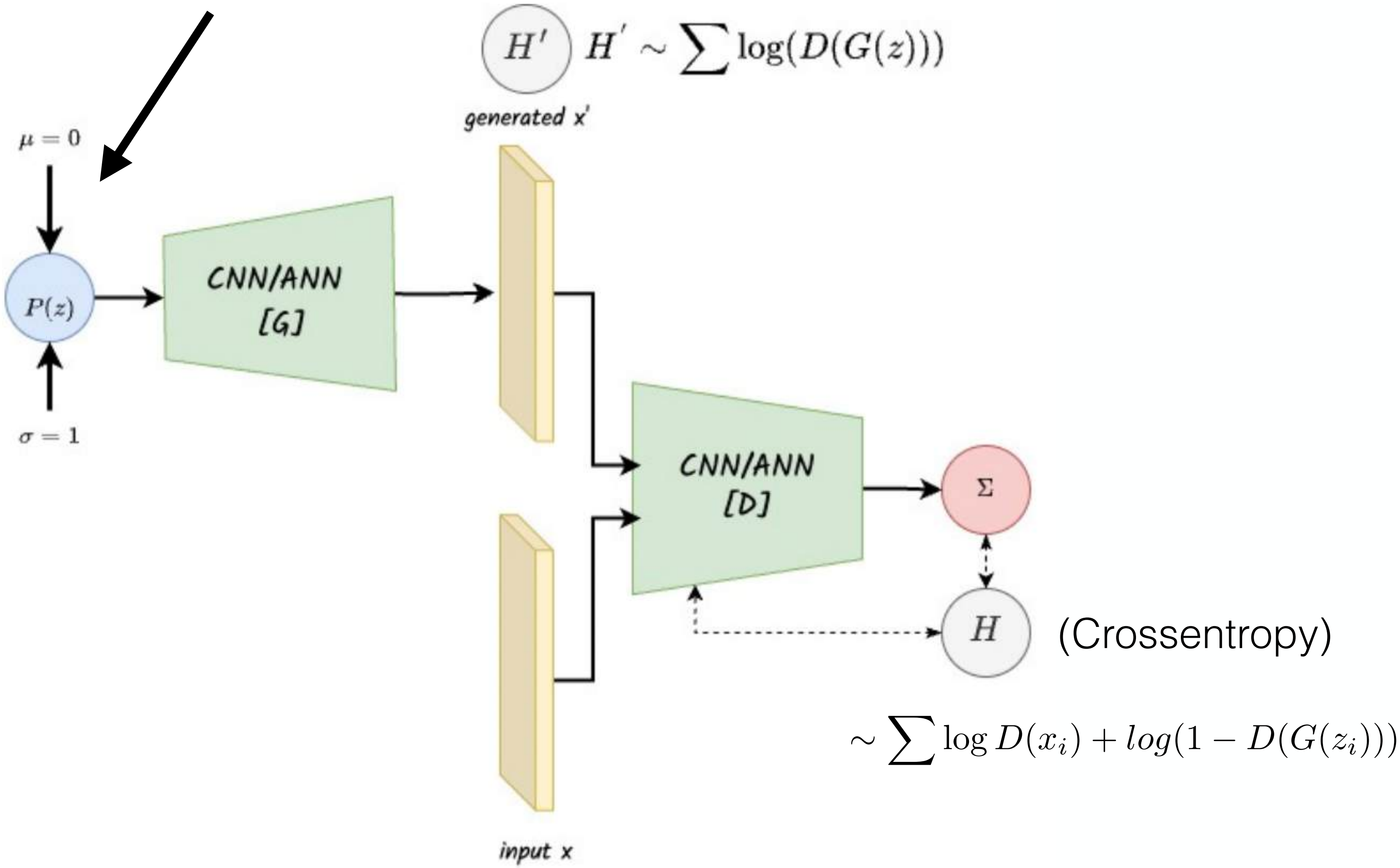
[Kingma and Welling, 2019]

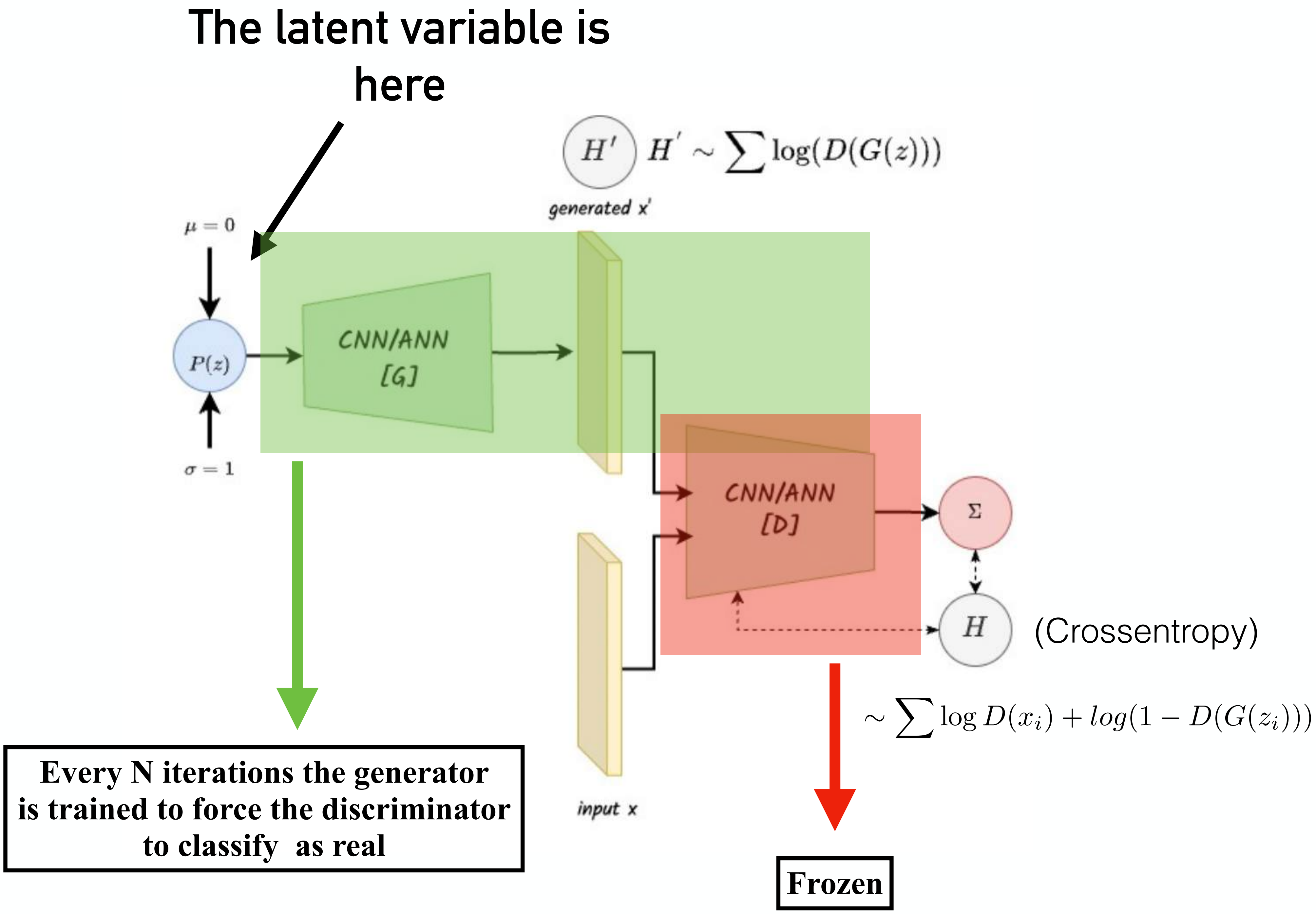
Generative Adversarial Networks

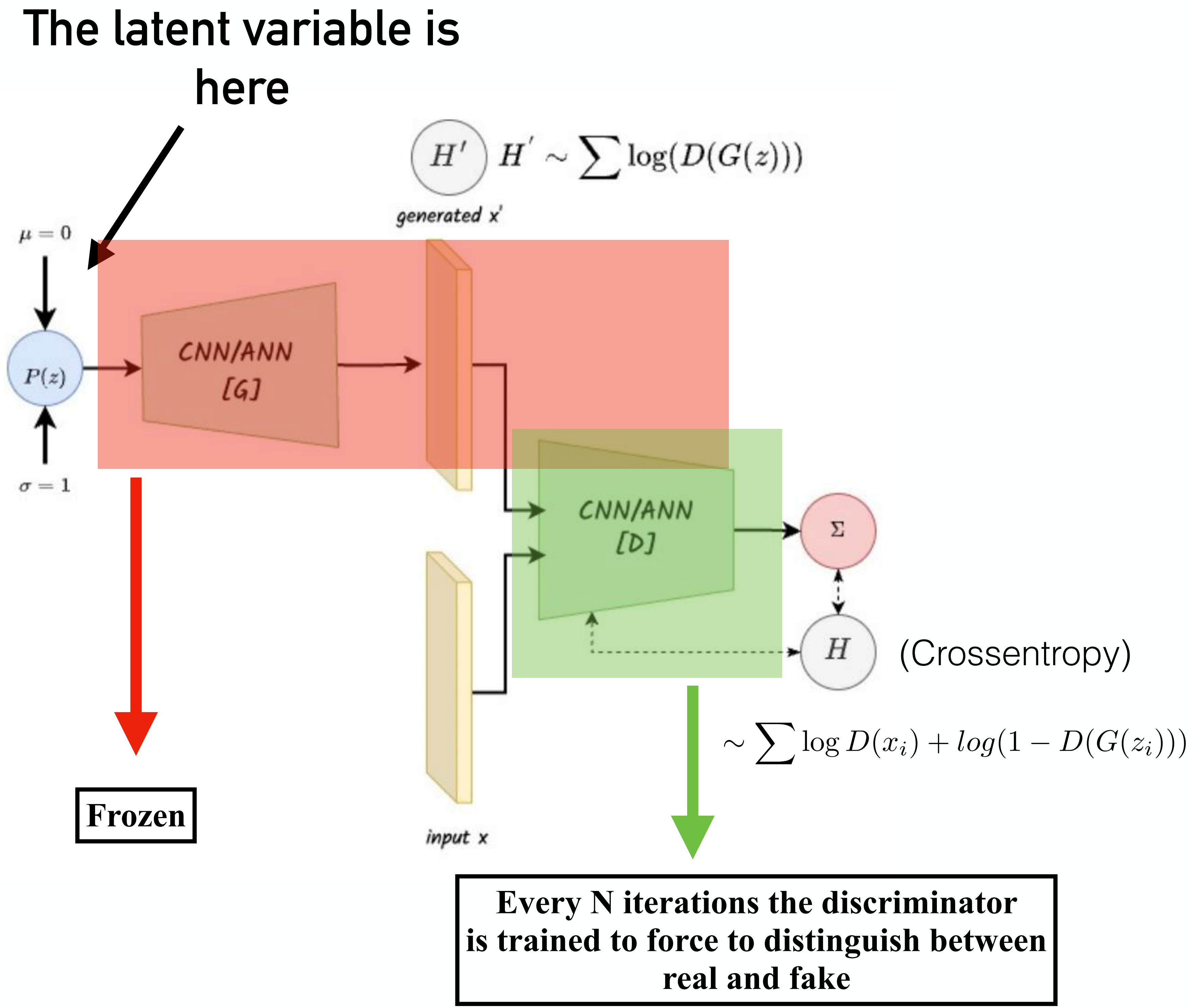
As for VAEs,
the goal of generative Adversarial Networks (GANs) is to estimate
$p(z|x)$

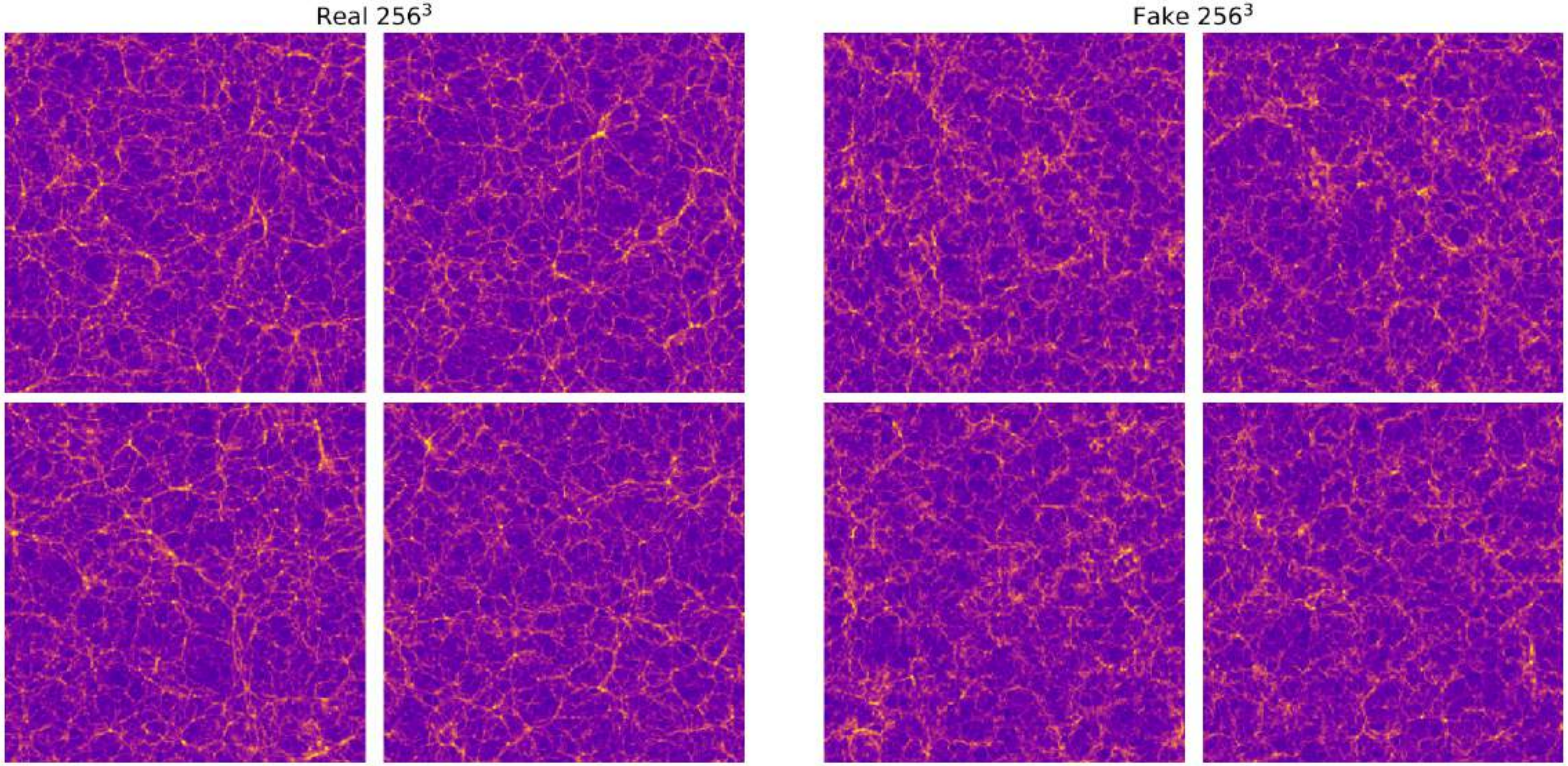
They convert the problem into a "supervised approach" by using two competing neural networks

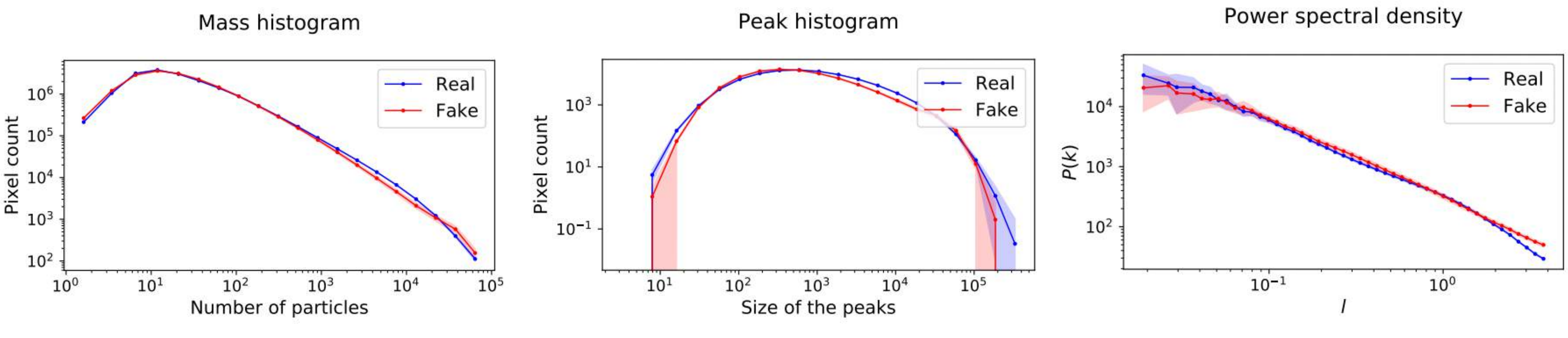
The latent variable is here



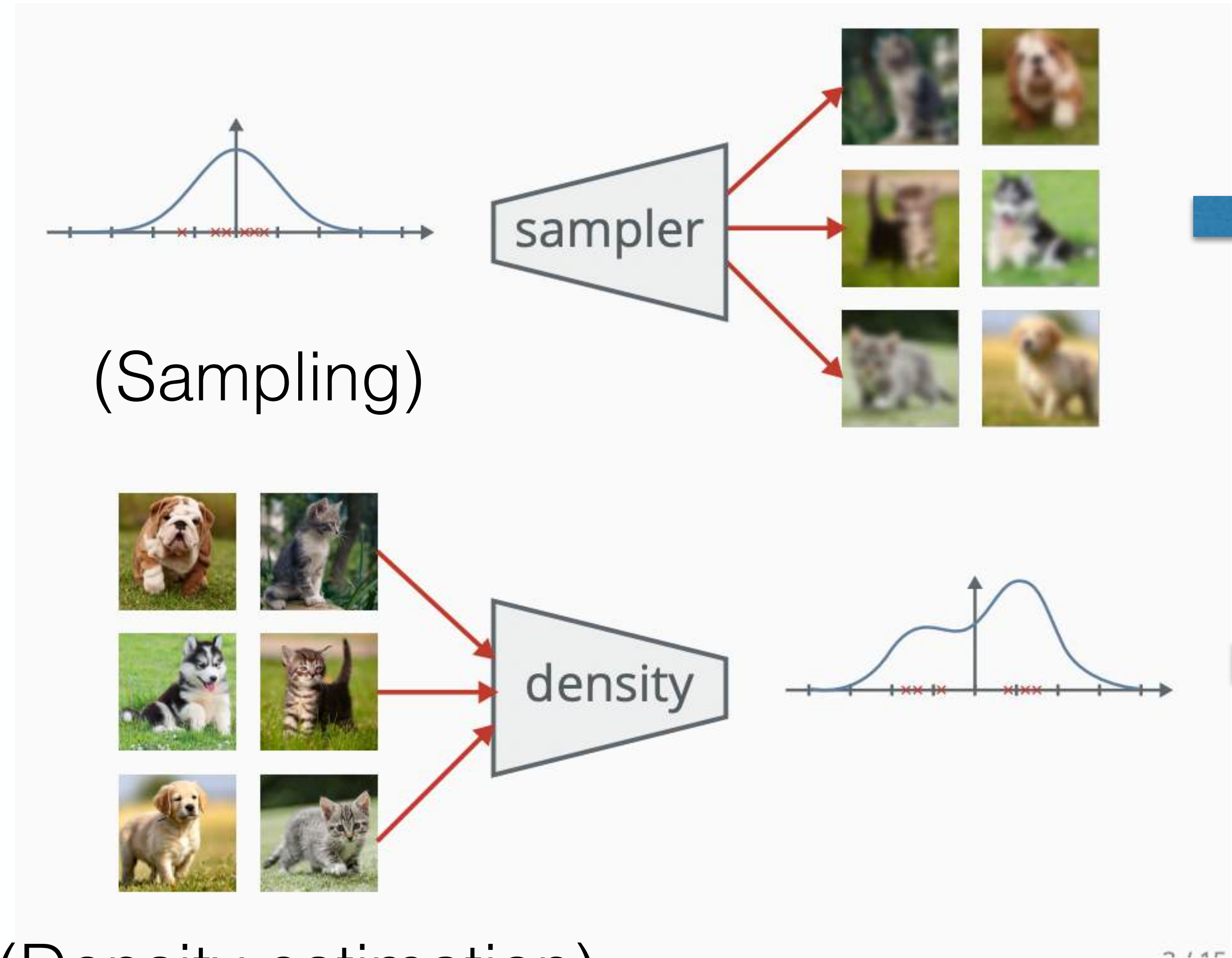

The latent variable is here
H'
$H' \sim \sum \log(D(G(z)))$
generated x'
$\mu = 0$
$P(z)$
$\sigma = 1$
CNN/ANN [G]
CNN/ANN [D]
Σ
H
(Crossentropy)
$\sim \sum \log D(x_i) + log(1 - D(G(z_i)))$
input x
Every N iterations the generator is trained to force the discriminator to classify as real
Frozen


The latent variable is here
H'
$H' \sim \sum \log(D(G(z)))$
generated x'
$\mu = 0$
$P(z)$
$\sigma = 1$
CNN/ANN [G]
CNN/ANN [D]
Σ
H
(Crossentropy)
$\sim \sum \log D(x_i) + log(1 - D(G(z_i)))$
input x
Frozen
Every N iterations the discriminator is trained to force to distinguish between real and fake





Perraudin+19



(Sampling)

(Density estimation)

easy with
VAEs and GANs

~ VAEs
difficult with
GANs

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

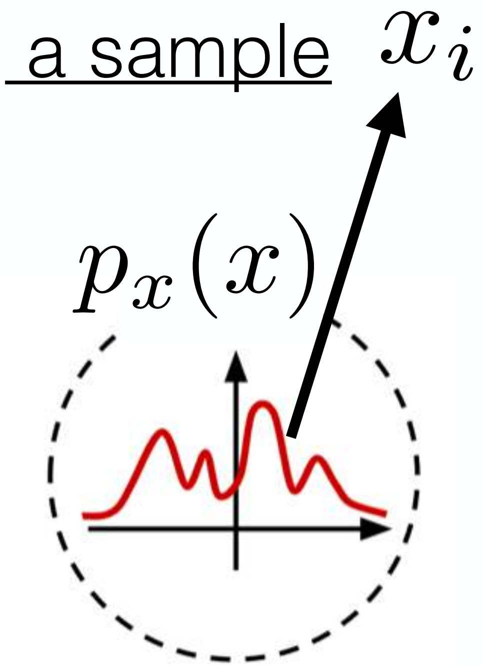
tractable pdf
(typically Gaussian)

Invertible differentiable
function

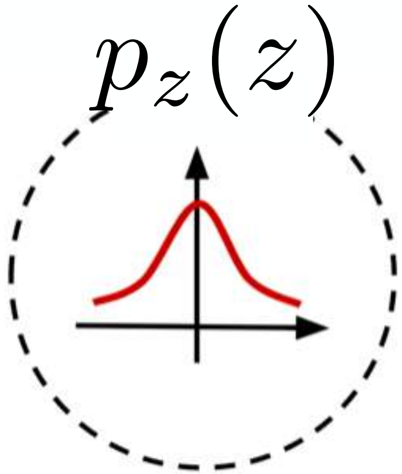
Determinant of the Jacobian
of f(x)

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

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



(unknown)



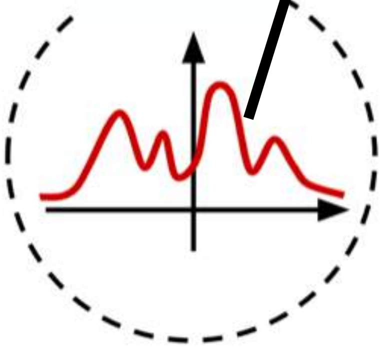
(known and easy)

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

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

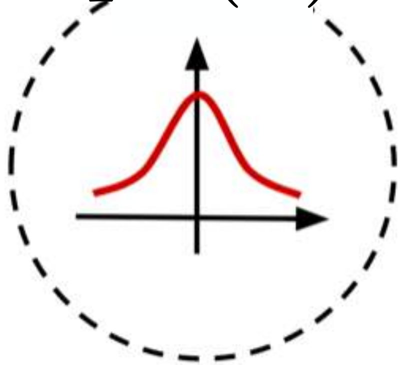
a sample $x_i \longrightarrow f_w(x_i)$

$p_x(x)$



(unknown)

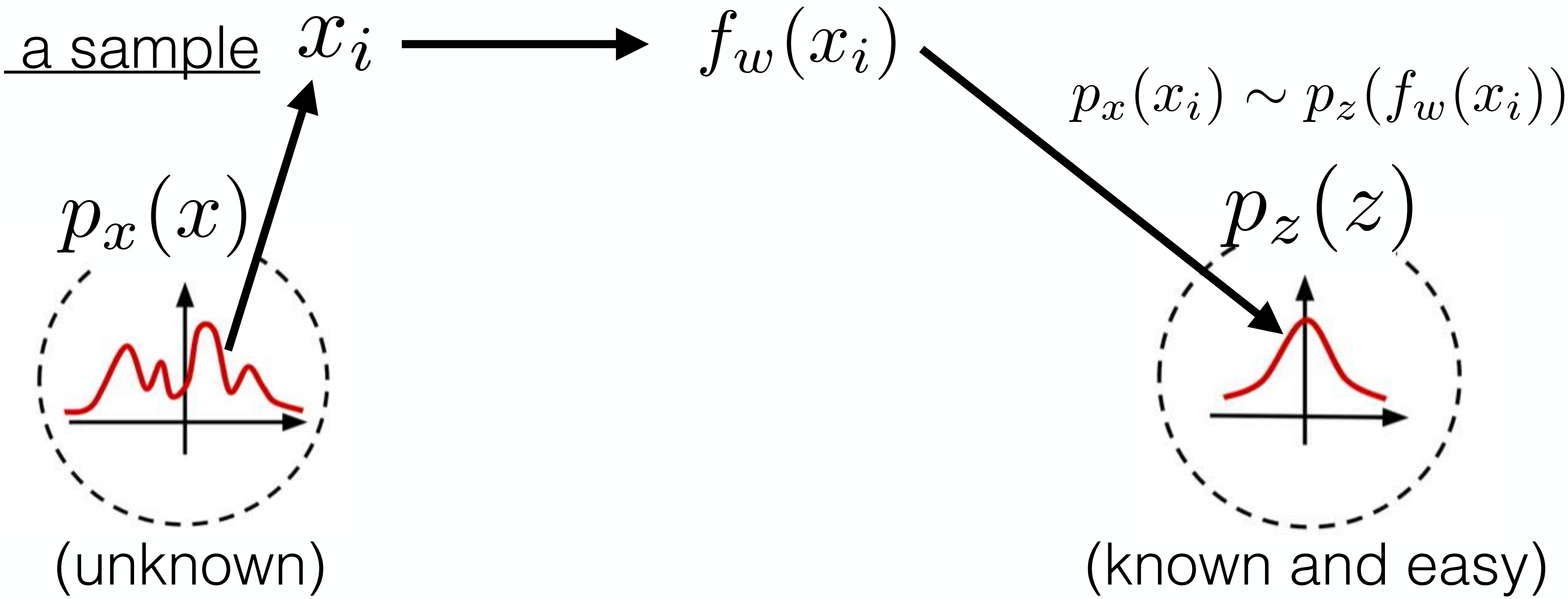
$p_z(z)$



(known and easy)

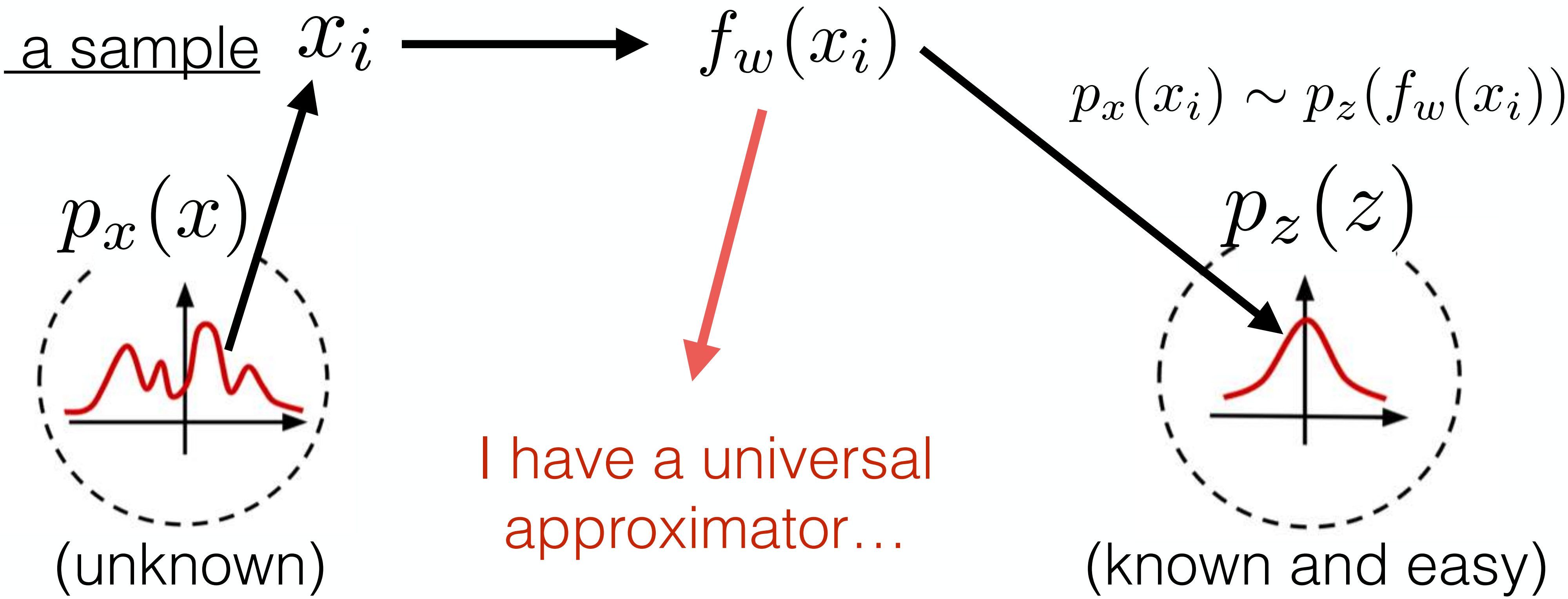
$$p_X(x) = p_Z(f(x))|det J_{f(x)}|$$

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)



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

NN learnable weights

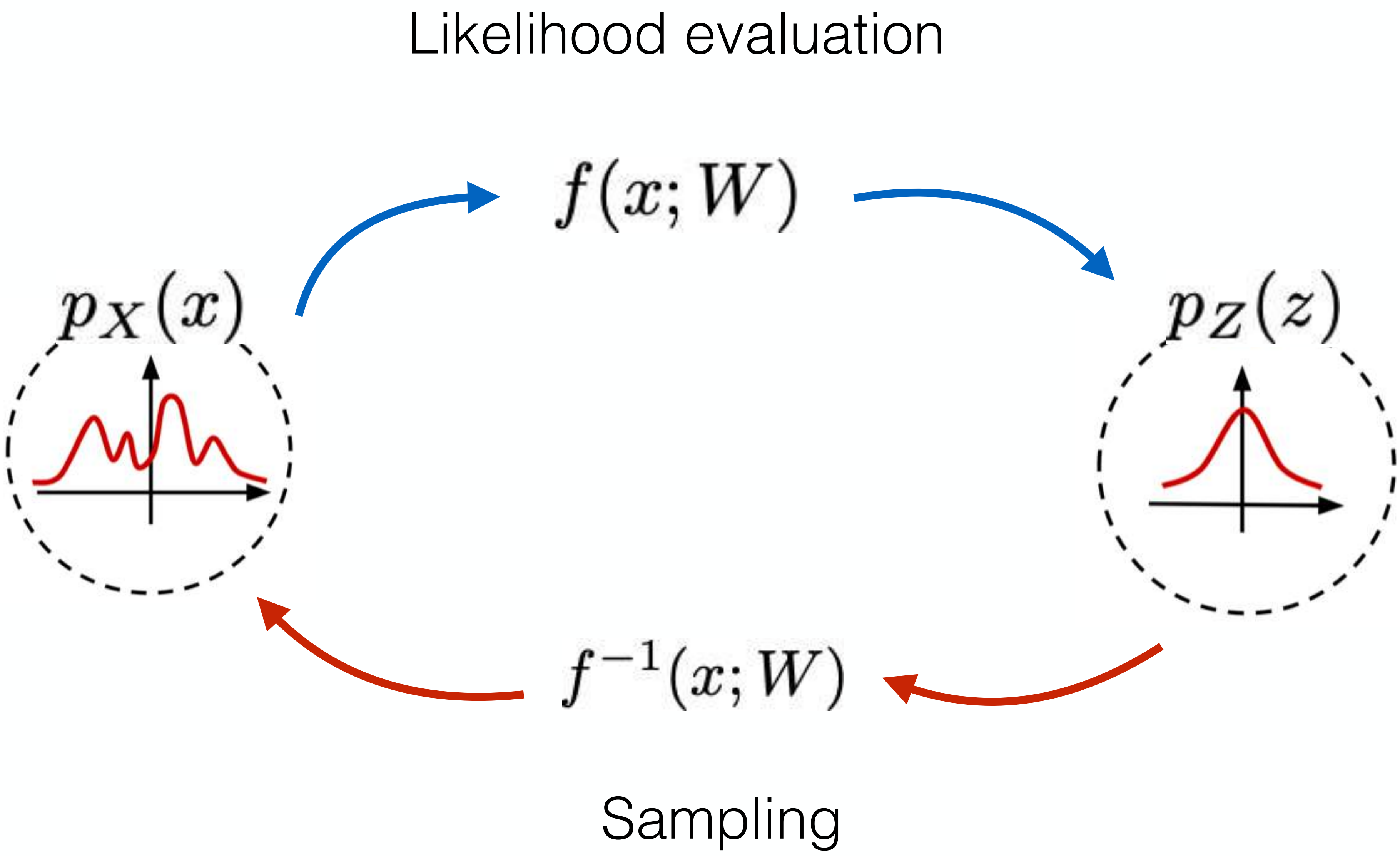
Then f(x) becomes: $f(x; W)$

NN learnable weights

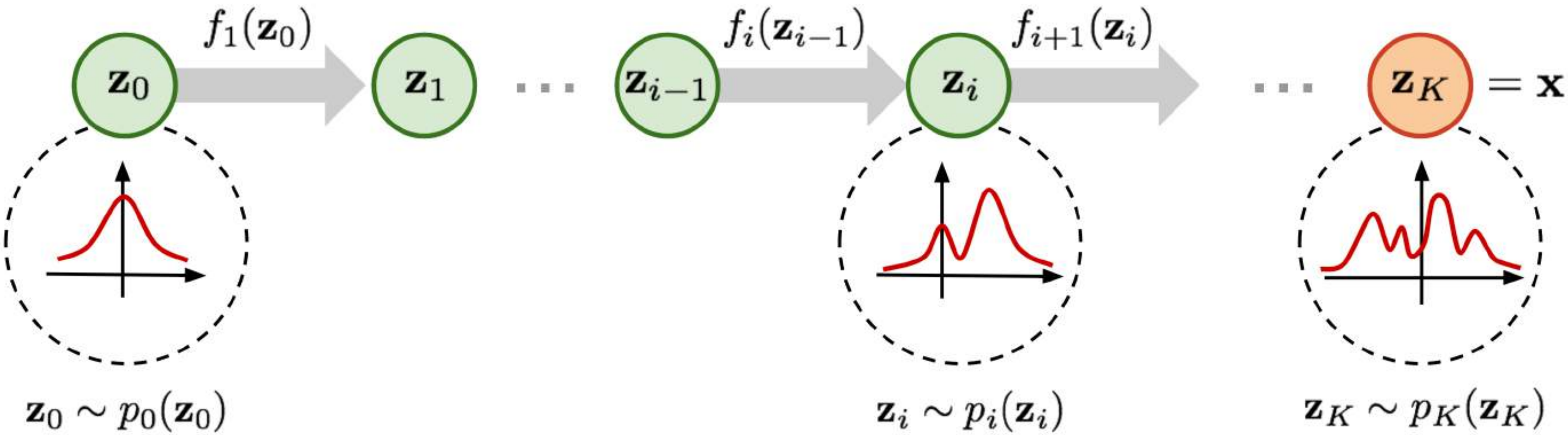
And the loss function: log likelihood

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

This is a gaussian


Likelihood evaluation
$f(x; W)$
$p_X(x)$
$p_Z(z)$
$f^{-1}(x; W)$
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)

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

NN

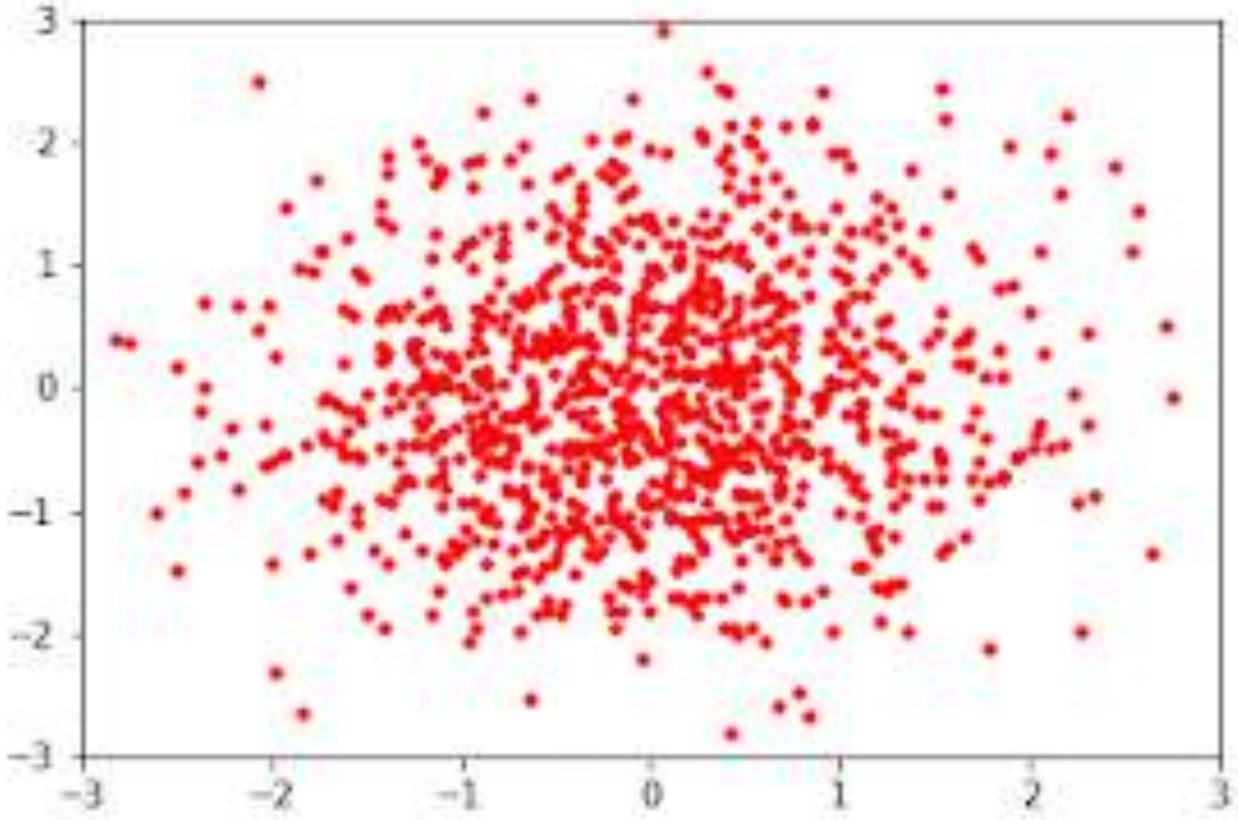
- Easily invertible:

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

- Jacobian:

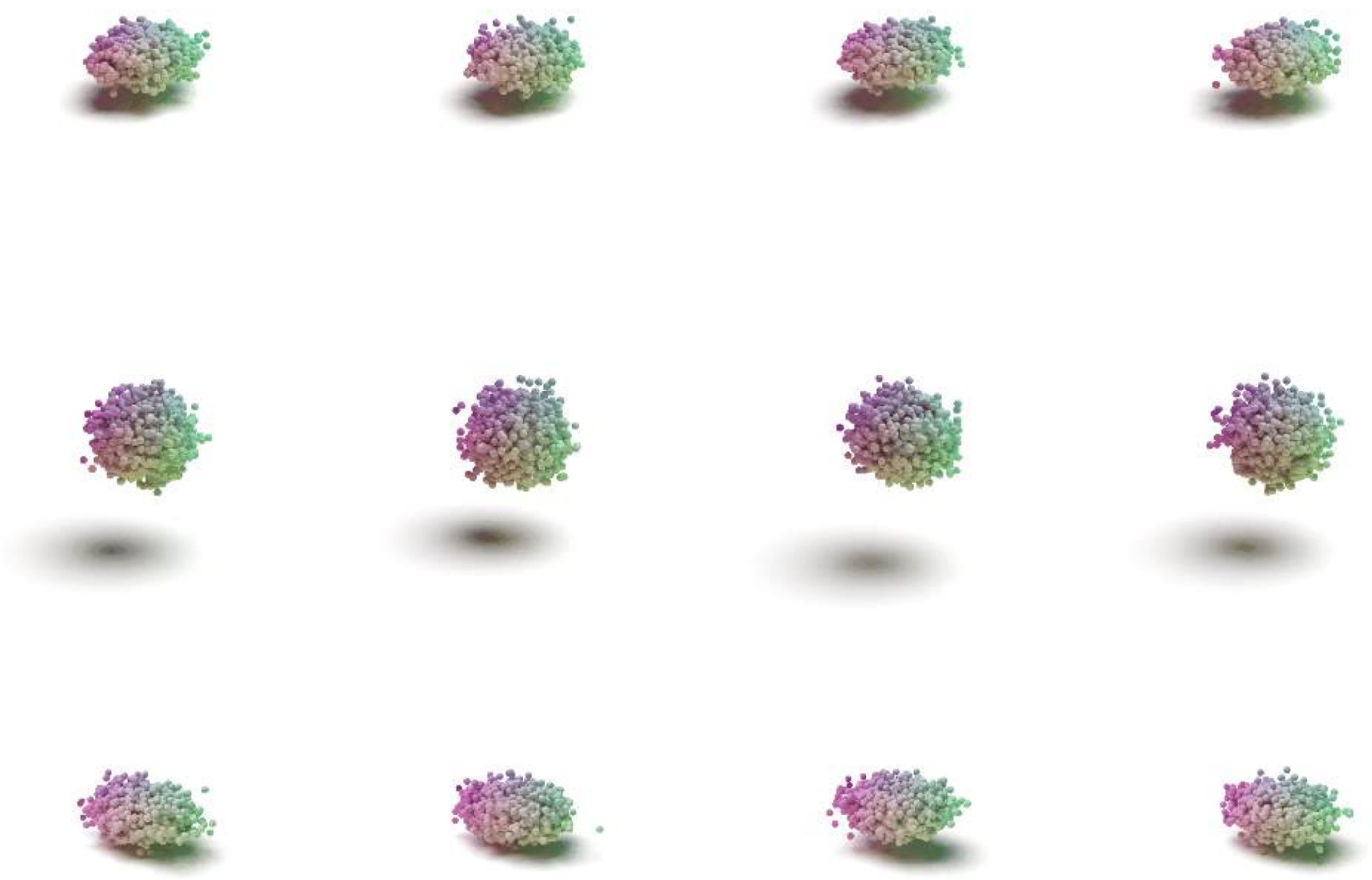
$$\mathbf{J} = \begin{bmatrix} \mathbb{I}_d & \mathbf{0}_{d\times(D-d)} \\ \frac{\partial \mathbf{y}_{d+1:D}}{\partial \mathbf{x}_{1:d}} & \operatorname{diag}(\exp(s(\mathbf{x}_{1:d}))) \end{bmatrix} \qquad \det(\mathbf{J}) = \prod_{j=1}^{D-d} \exp(s(\mathbf{x}_{1:d}))_j = \exp(\sum_{j=1}^{D-d} s(\mathbf{x}_{1:d})_j)$$

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



credit: Eric Jang

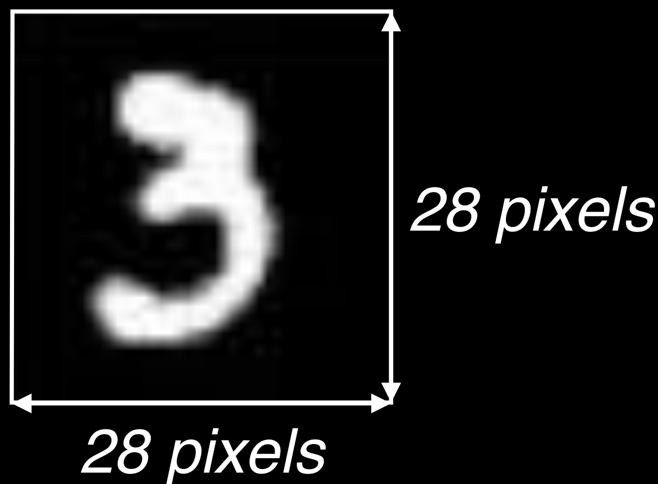
normalizing flows are generative models that are easy to evaluate and flexibly expressive



Yang+(2019) PointFlow

MNIST data set

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

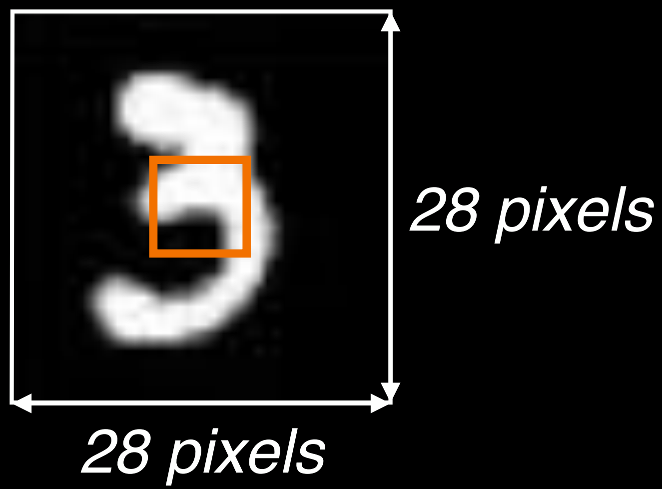


MNIST data set

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

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

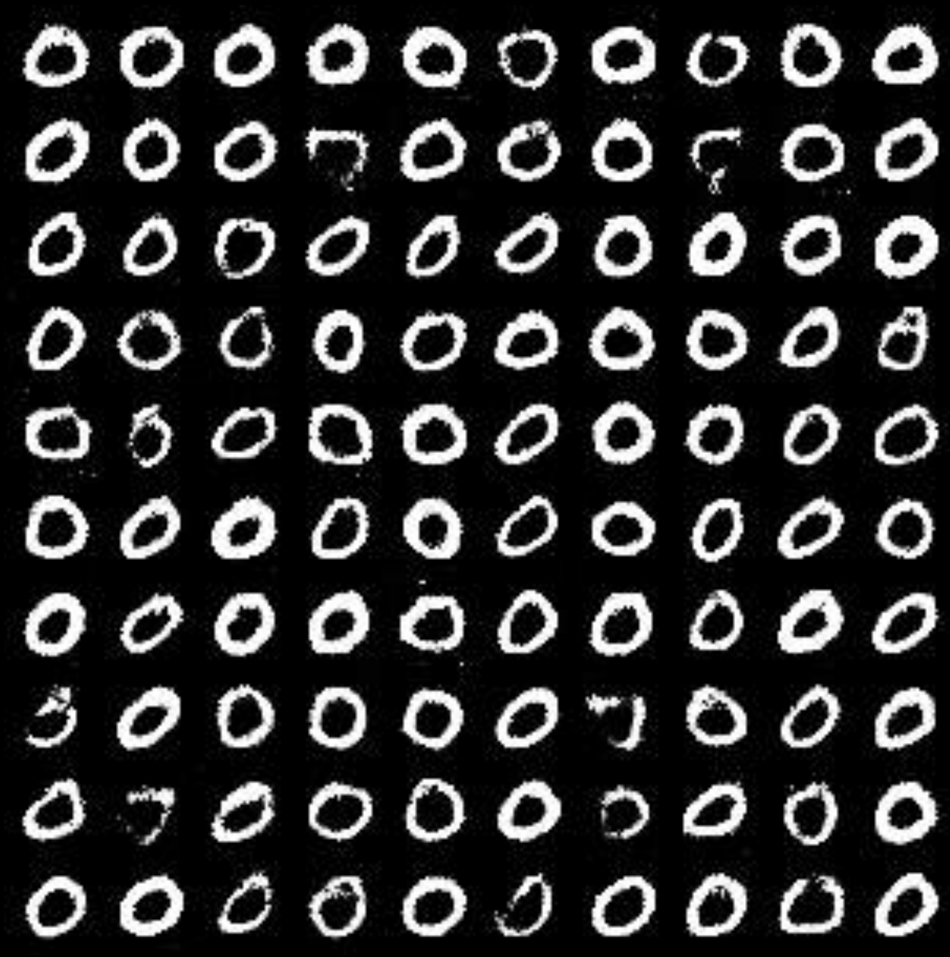
train a generative model to estimate **conditional distributions**



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

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



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

$$p(\text{pi}\overset{\theta}{\cancel{\text{xels}\mid}}\,\text{c}\underset{X}{\cancel{\text{entral pixe}}}\text{ls}) \approx q_{\phi}$$

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	✓	✗	✓	✗

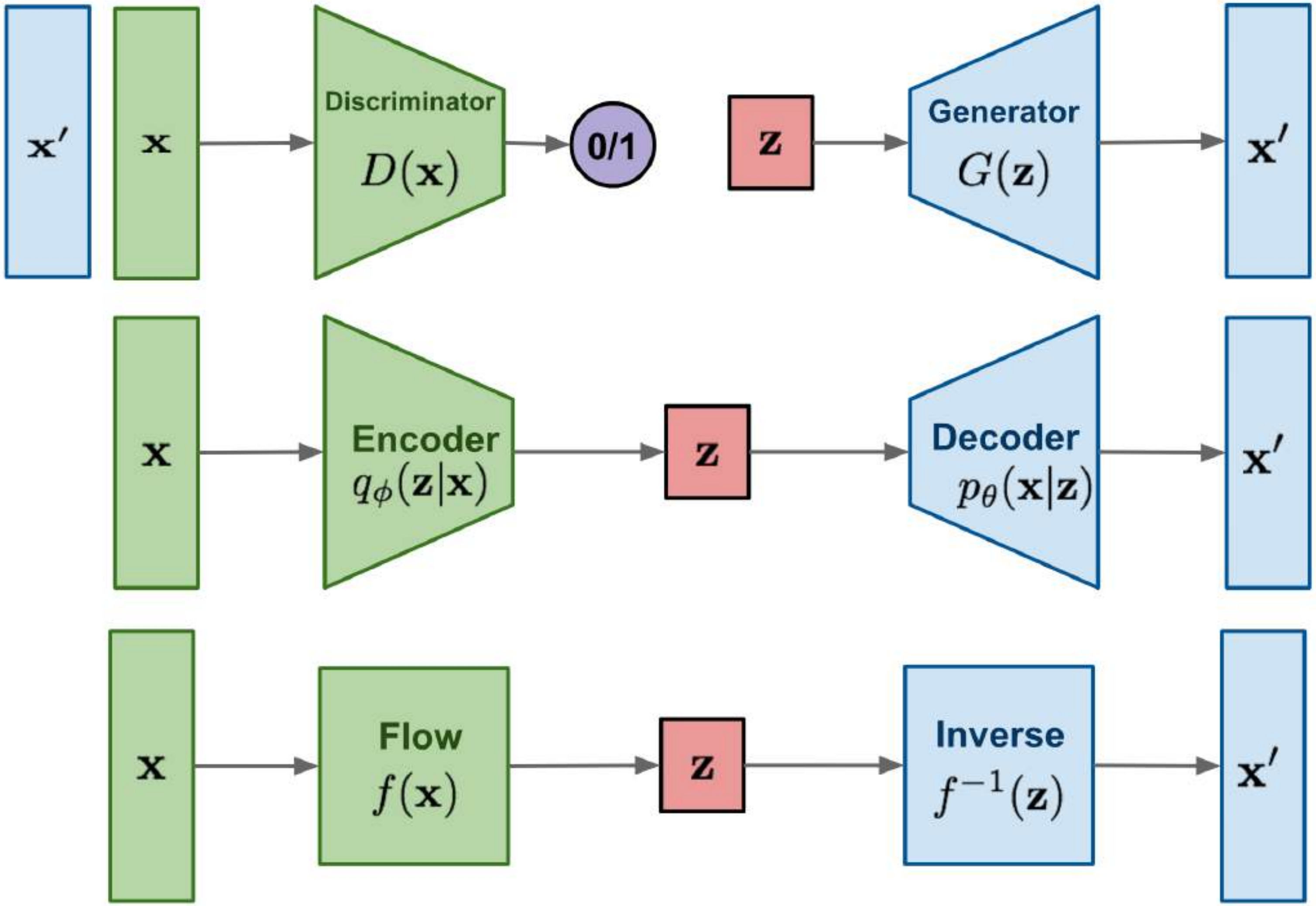
Grathwohl+18

* there is a third member of the family now: score based models which try to estimate the gradient of the likelihood (score) instead of the likelihood

GAN: minimax the classification error loss.

VAE: maximize ELBO.

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



(+ score based diffusion models)

4. Simulation Based Inference (SBI)

“standard” bayesian inference

goal of inference

posterior **likelihood** **prior**

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

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

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)**

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

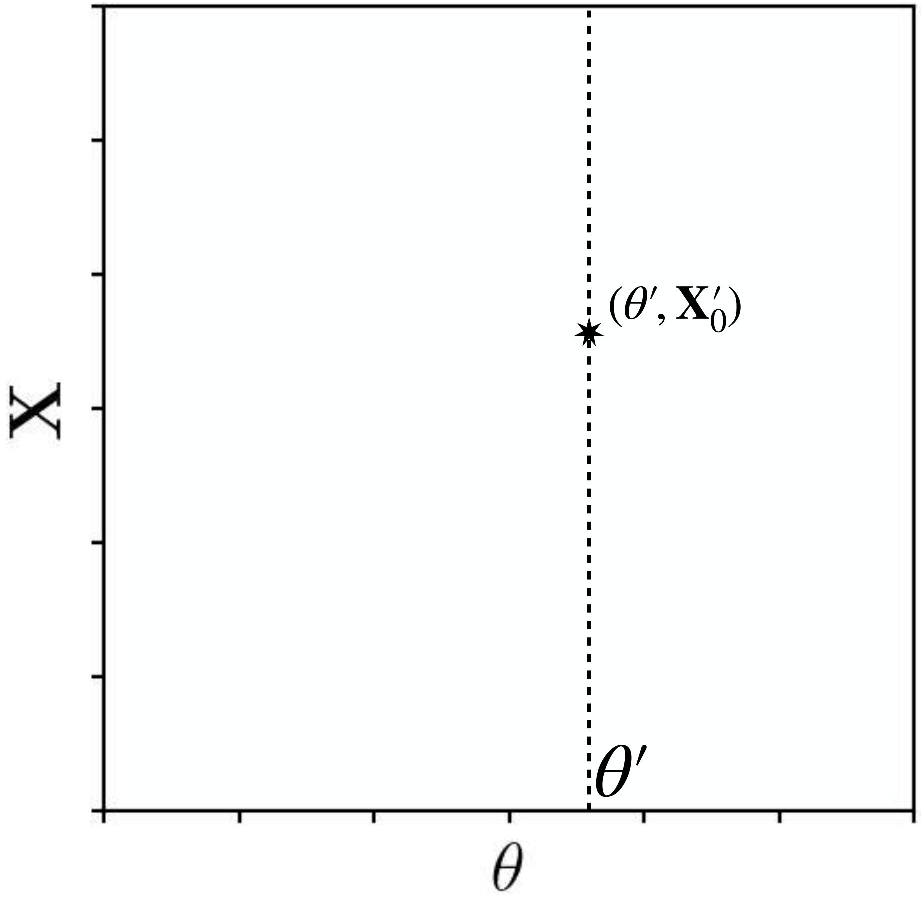
Sometimes the likelihood is intractable. e.g. Baryonic properties of DM haloes?

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

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

forward model F that can simulate observations:

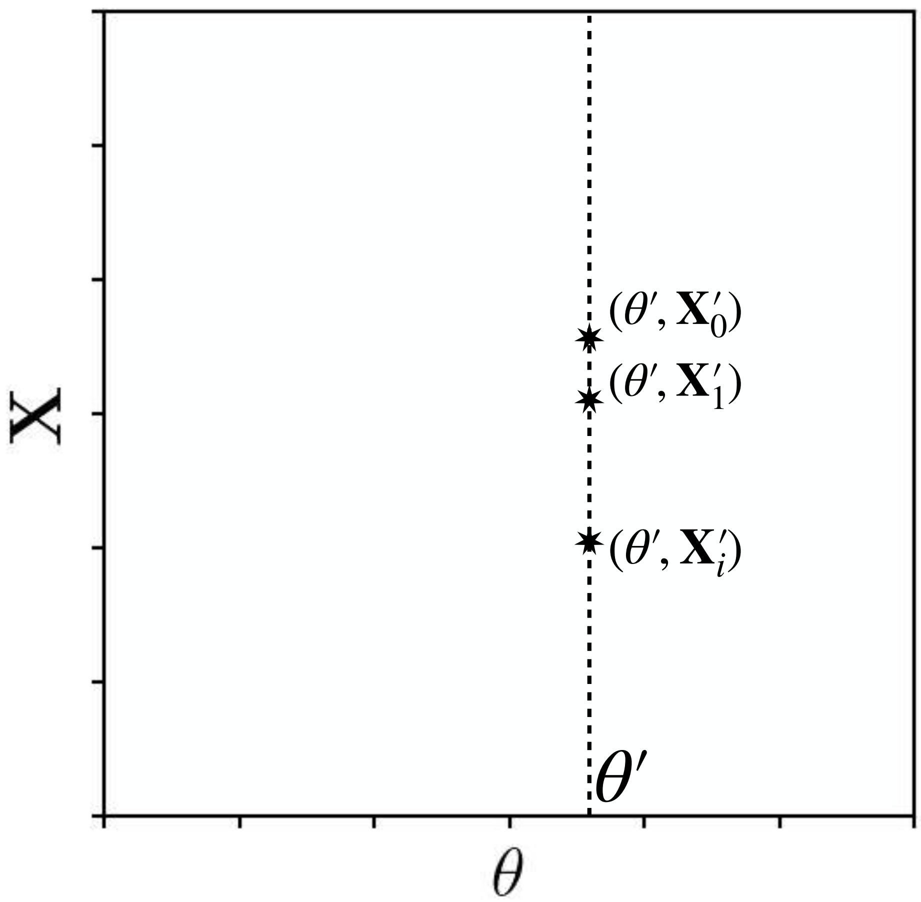
$$\mathbf{X}'_0, \mathbf{X}'_1 \ldots \sim F(\theta')$$



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

forward model F that can simulate observations:

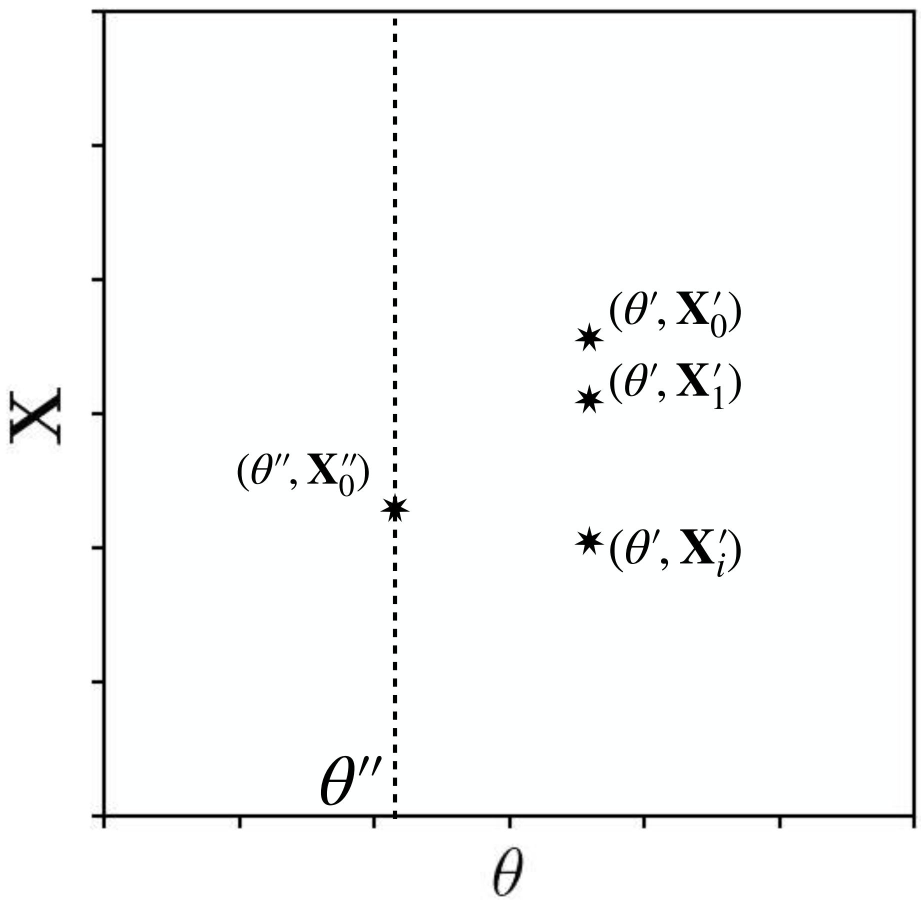
$$\mathbf{X}'_0, \mathbf{X}'_1 \ldots \sim F(\theta')$$



F must **include noise model**

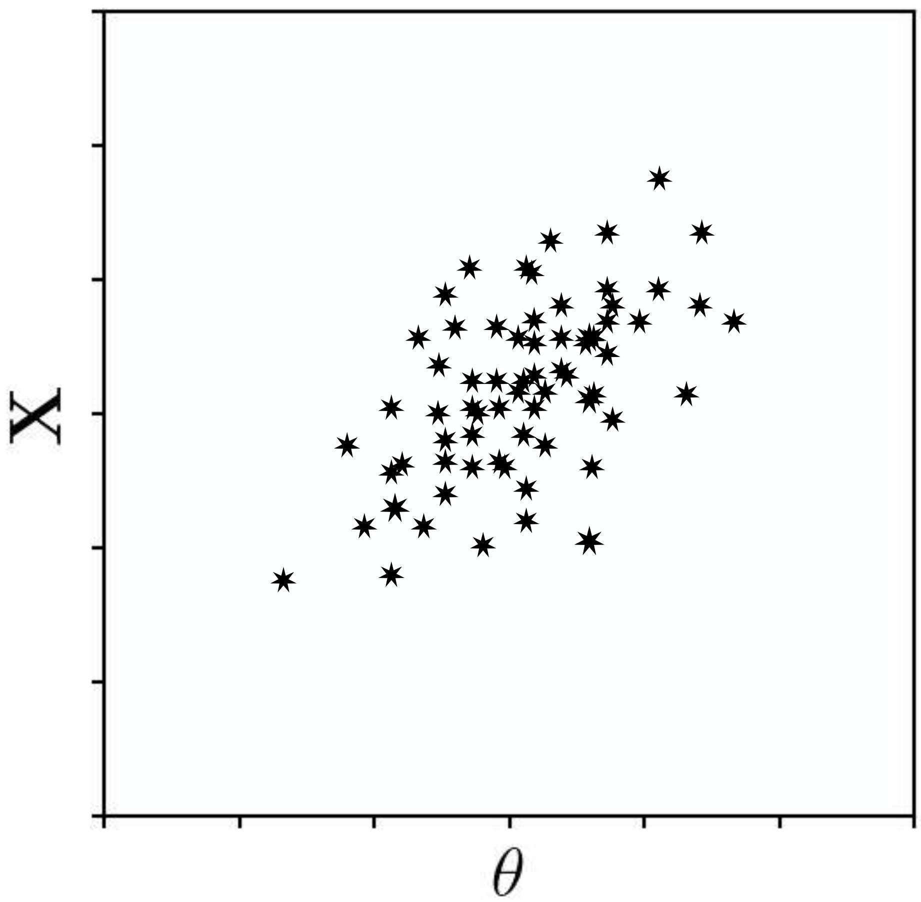
forward model F that can simulate observations:

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



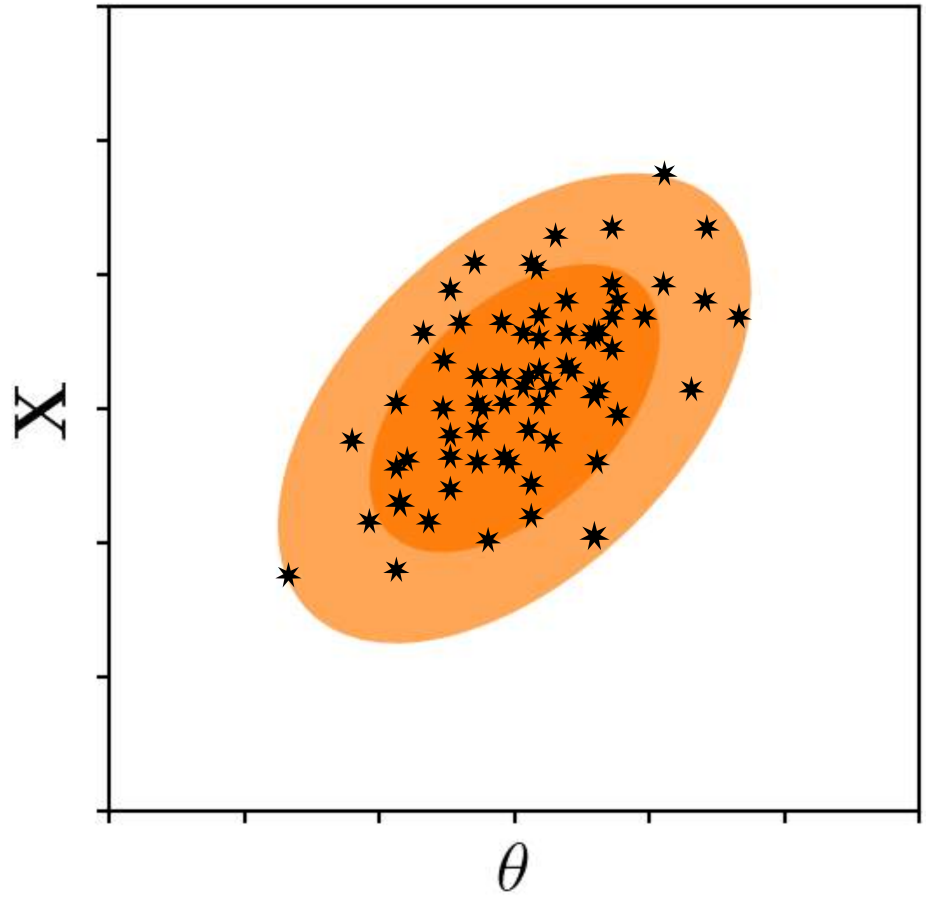
forward model F that can simulate observations:

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



forward model F that can simulate observations:

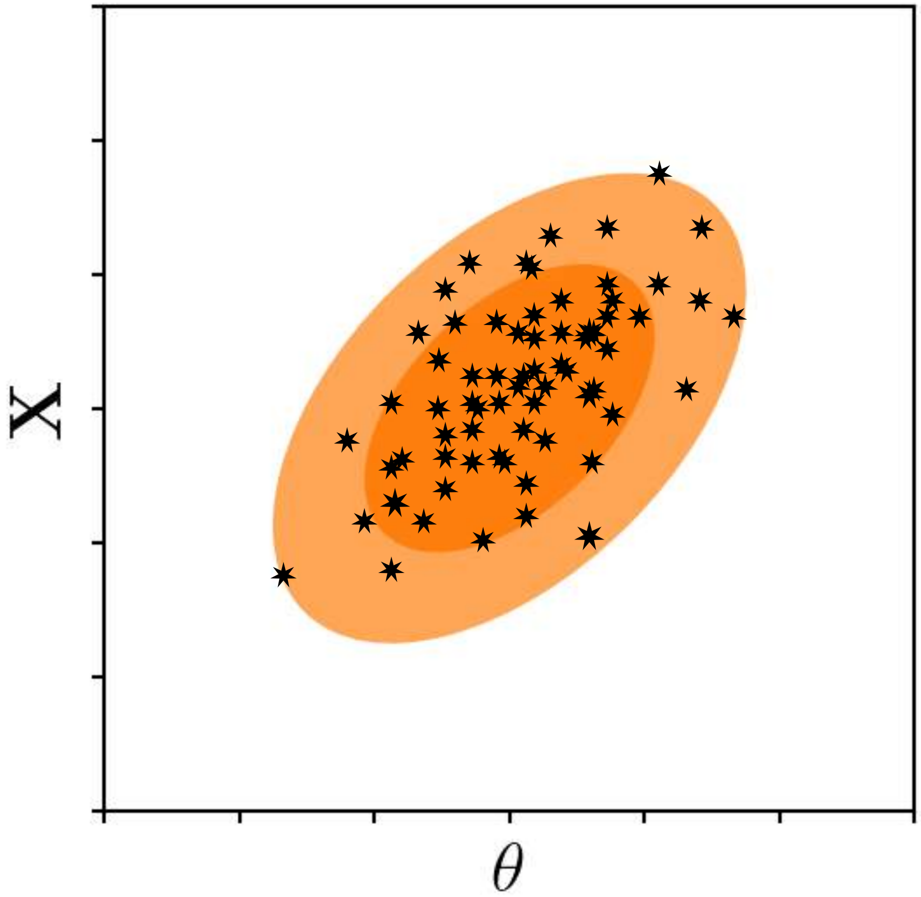
$$\mathbf{X}_0', \mathbf{X}_1' \ldots \sim F(\theta')$$



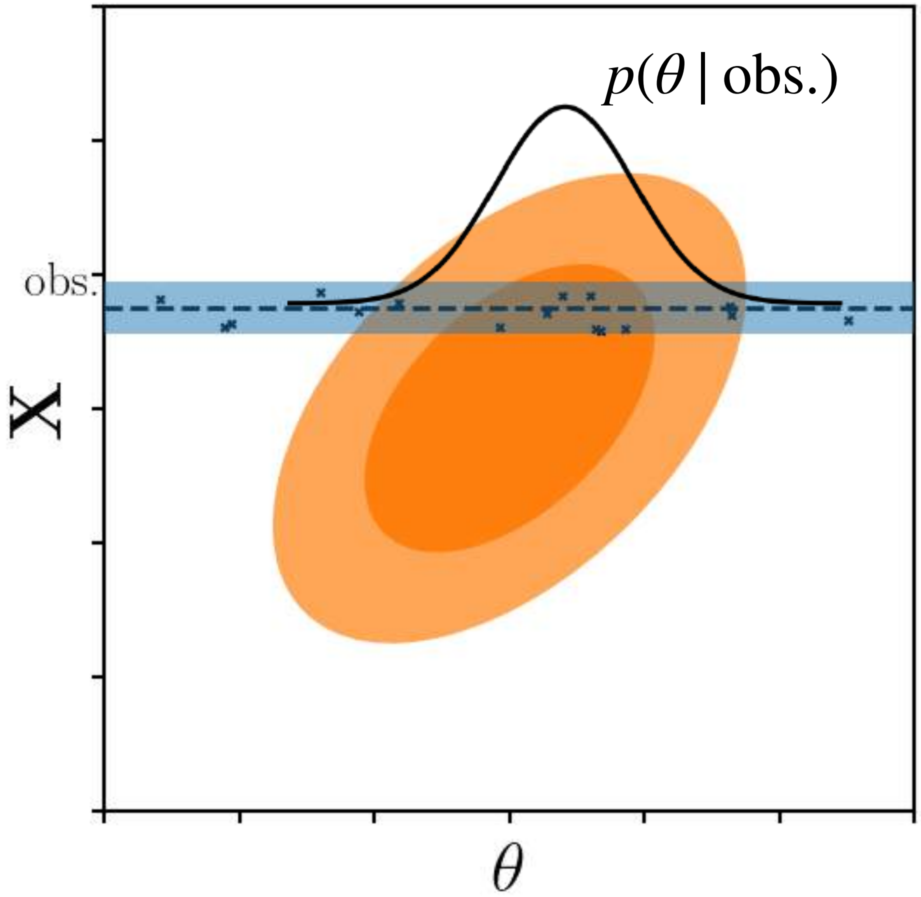
F allows us to sample $(\theta', X') \sim p(\theta, X)$ ← Unknown

forward model F that can simulate observations:

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



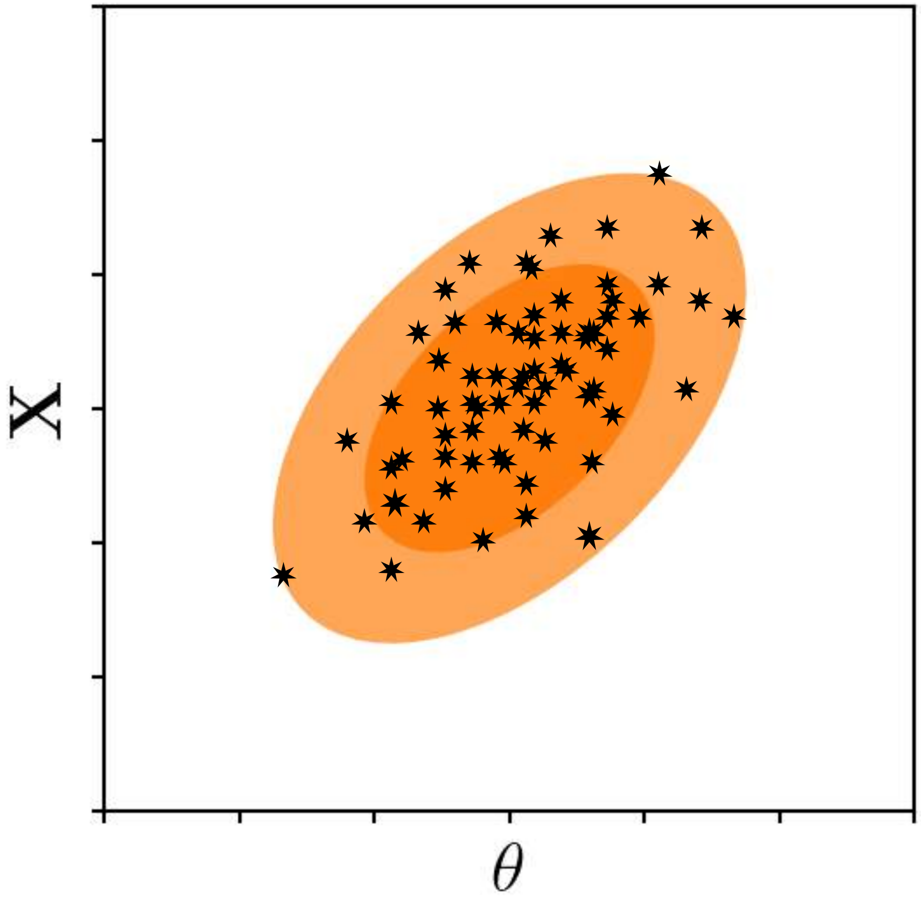
F allows us to sample $(\theta', X') \sim p(\theta, X)$ ← Unknown with F one way to infer the posterior is through *brute force* — **approximate bayesian computation** (ABC)



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!**



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

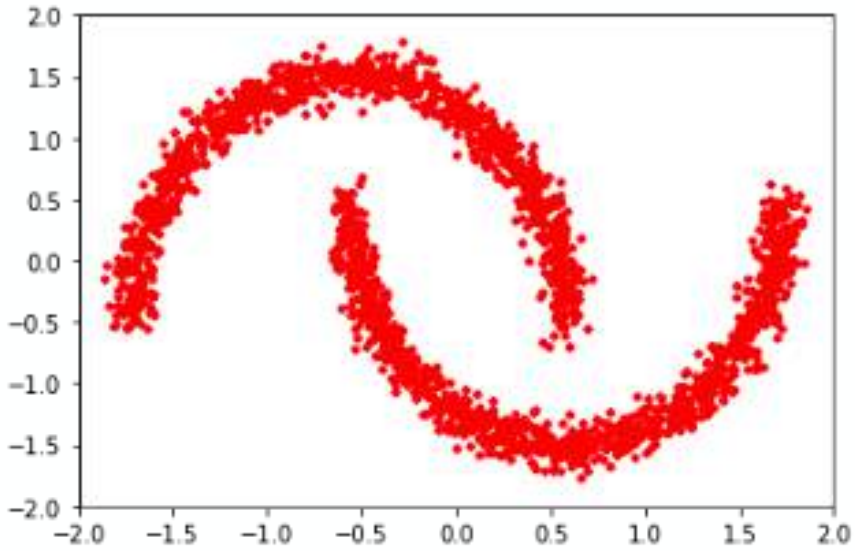
Pixel space!

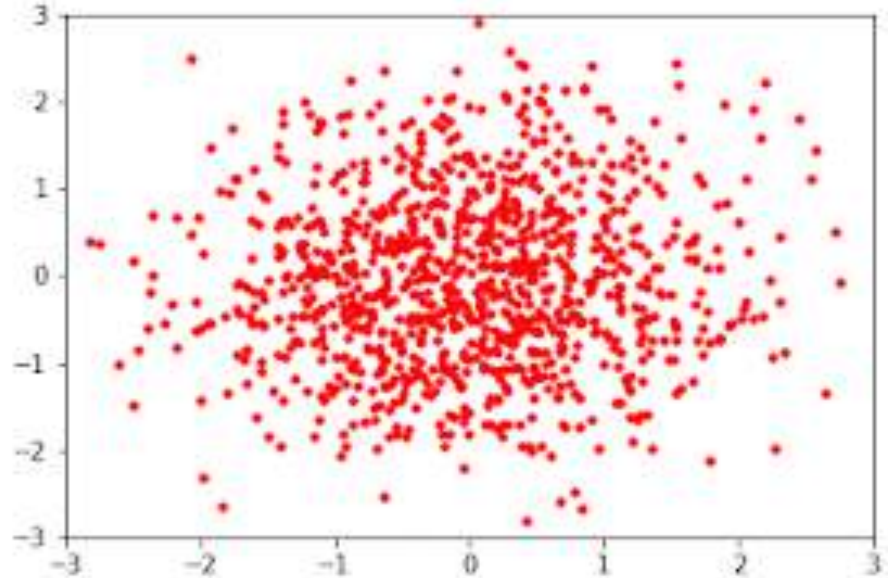
Pixel space

The deep learning revolution has enabled fast density estimation at very high dimension

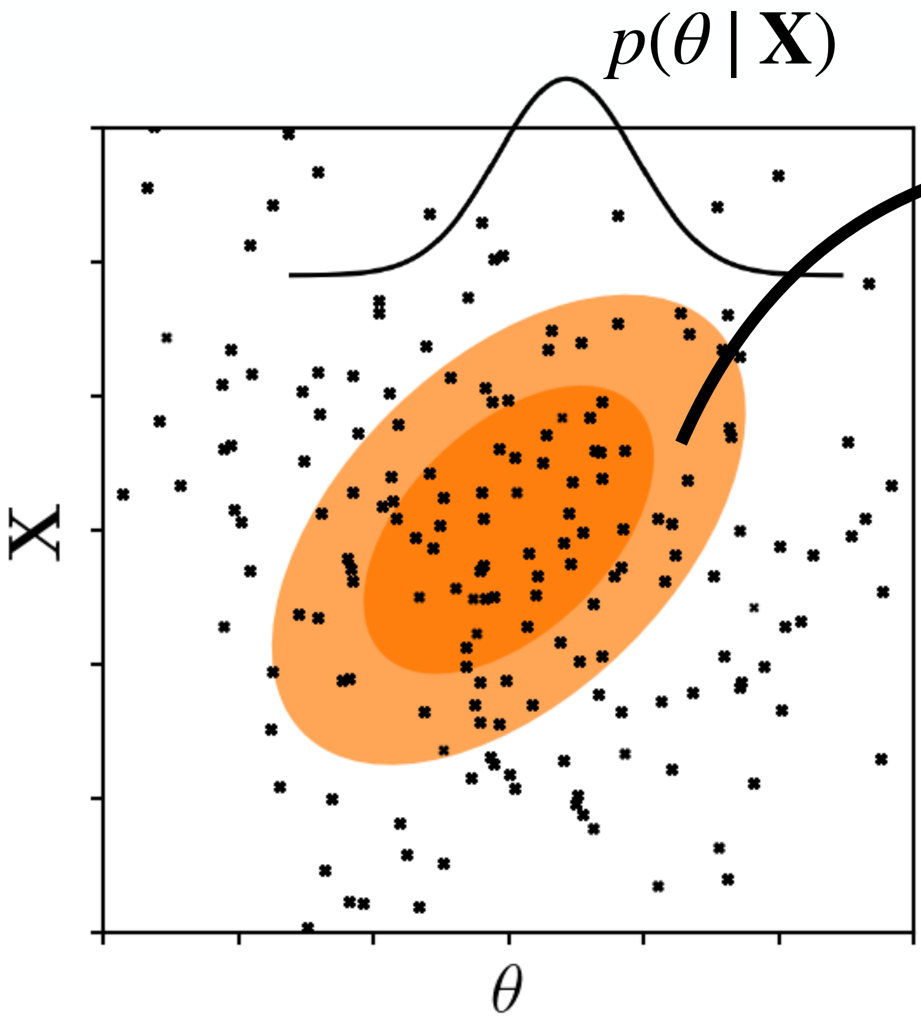
$$p(\theta|x)$$

Not Gaussian





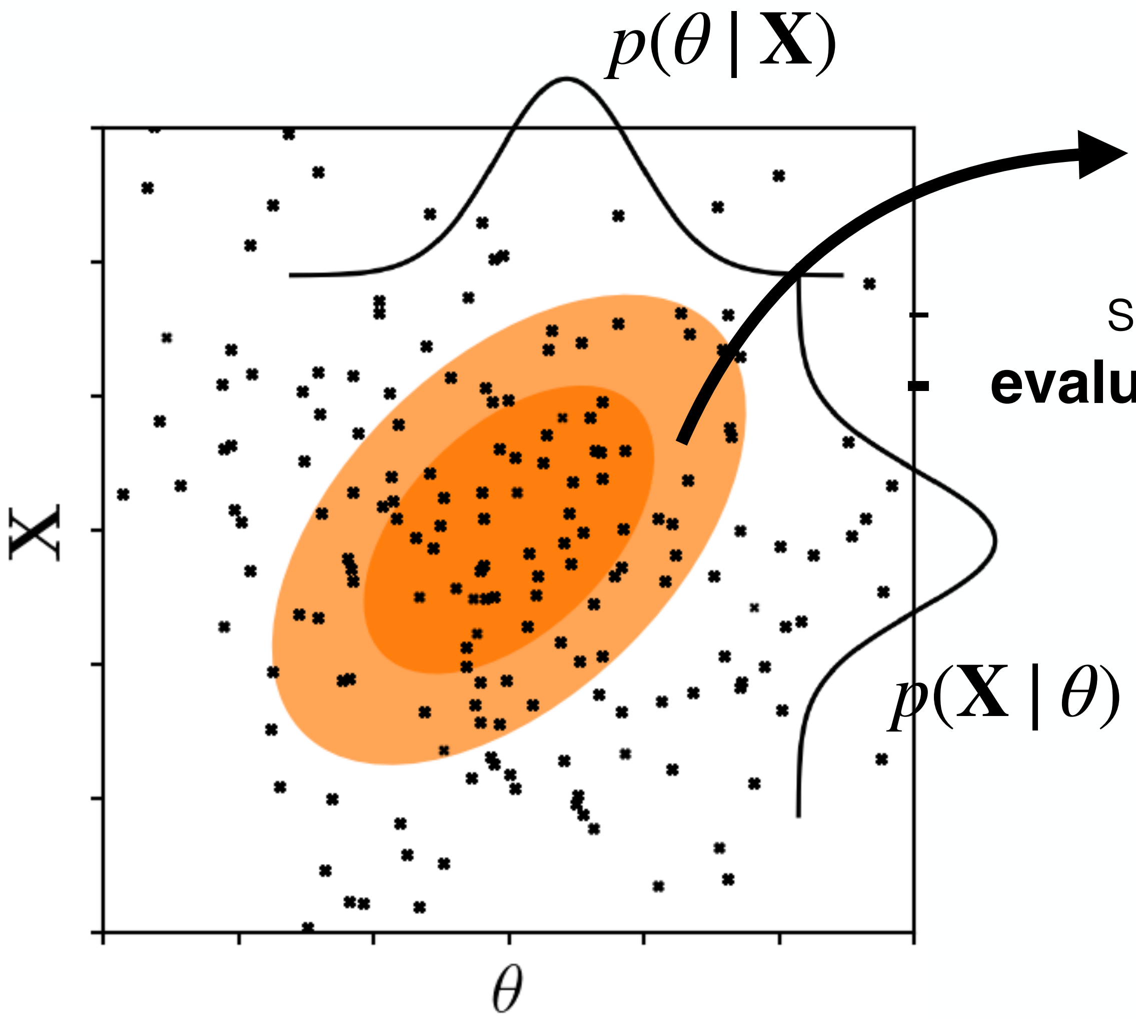
simulation-based inference is a **density estimation problem!**



<u>Generative model:</u>

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

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



<u>Generative model:</u>

sample from the posterior

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

- Assuming we have a perfect simulator, we have (data, parameter) pairs. How do we do inference?

$$\mathbf{x} \qquad \theta$$

$$\underbrace{P(\boldsymbol{\theta}|\mathbf{x})}_{\text{“Posterior”}} \propto \underbrace{P(\mathbf{x}|\boldsymbol{\theta})}_{\text{“Likelihood”}} \underbrace{P(\boldsymbol{\theta})}_{\text{“Prior”}}$$

Neural **P**osterior **E**stimation

Neural **L**ikelihood **E**stimation

$\mathbf{x}$ θ

- Assuming we have a <u>perfect simulator</u>, we have (data, parameter) pairs. How do we do inference?

$$\underbrace{P(\boldsymbol{\theta}|\mathbf{x})}_{\text{“Posterior”}} \propto \underbrace{P(\mathbf{x}|\boldsymbol{\theta})}_{\text{“Likelihood”}} \underbrace{P(\boldsymbol{\theta})}_{\text{“Prior”}}$$

Neural Posterior Estimation	Neural Likelihood Estimation
Amortized	Requires sampling (e.g. MCMC)
(Need to sample many posteriors for many X’s)	**(Need to build model into hierarchical sampling scheme)**

simulation-based inference flowchart for amortized inference

generate training data using forward model F

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

train generative model

$$p(\theta \,|\, \mathbf{X}) \approx q_\phi(\theta \,|\, \mathbf{X})$$

infer posterior estimate

$$\theta' \sim q_\phi(\theta \,|\, \mathbf{X}_{\text{obs}})$$

simulation-based inference flowchart for amortized inference

assumes
the forward model
perfectly matches reality

generate training data using forward model F

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

train generative model

$$p(\theta \,|\, \mathbf{X}) \approx q_\phi(\theta \,|\, \mathbf{X})$$

infer posterior estimate

$$\theta' \sim q_\phi(\theta \,|\, \mathbf{X}_{\mathrm{obs}})$$

simulation-based inference flowchart for amortized inference

assumes
the forward model
perfectly matches reality

generate training data using forward model F

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

train generative model

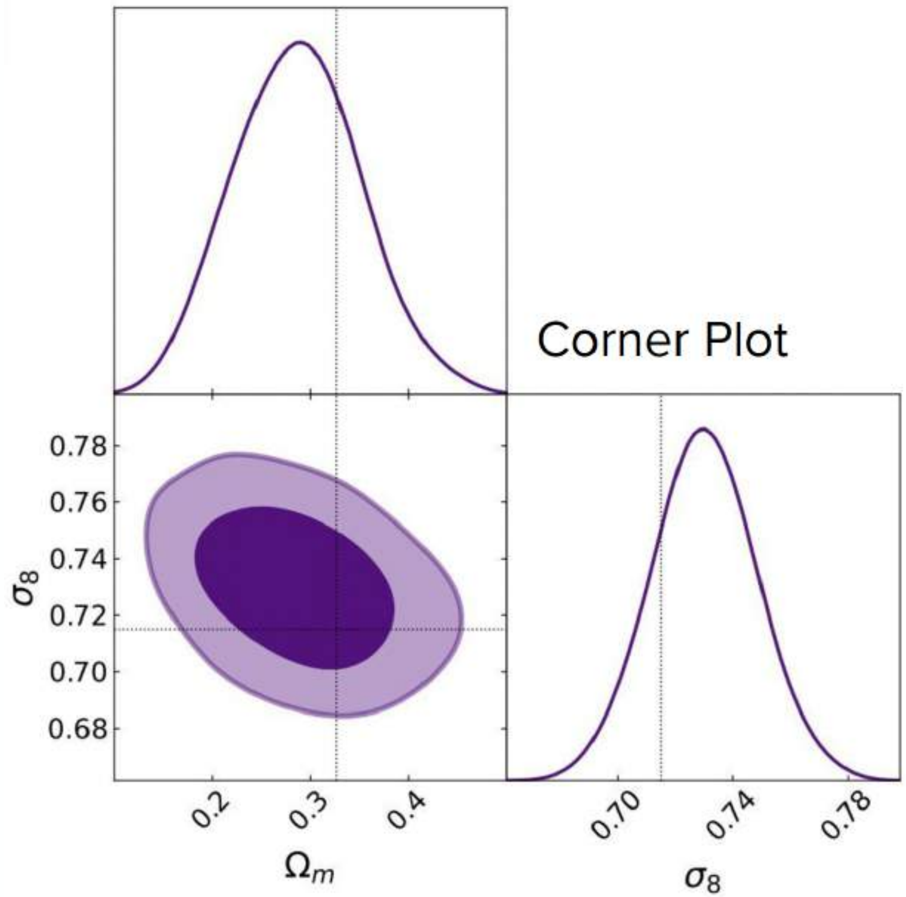
$$p(\theta \,|\, \mathbf{X}) \approx q_\phi(\theta \,|\, \mathbf{X})$$

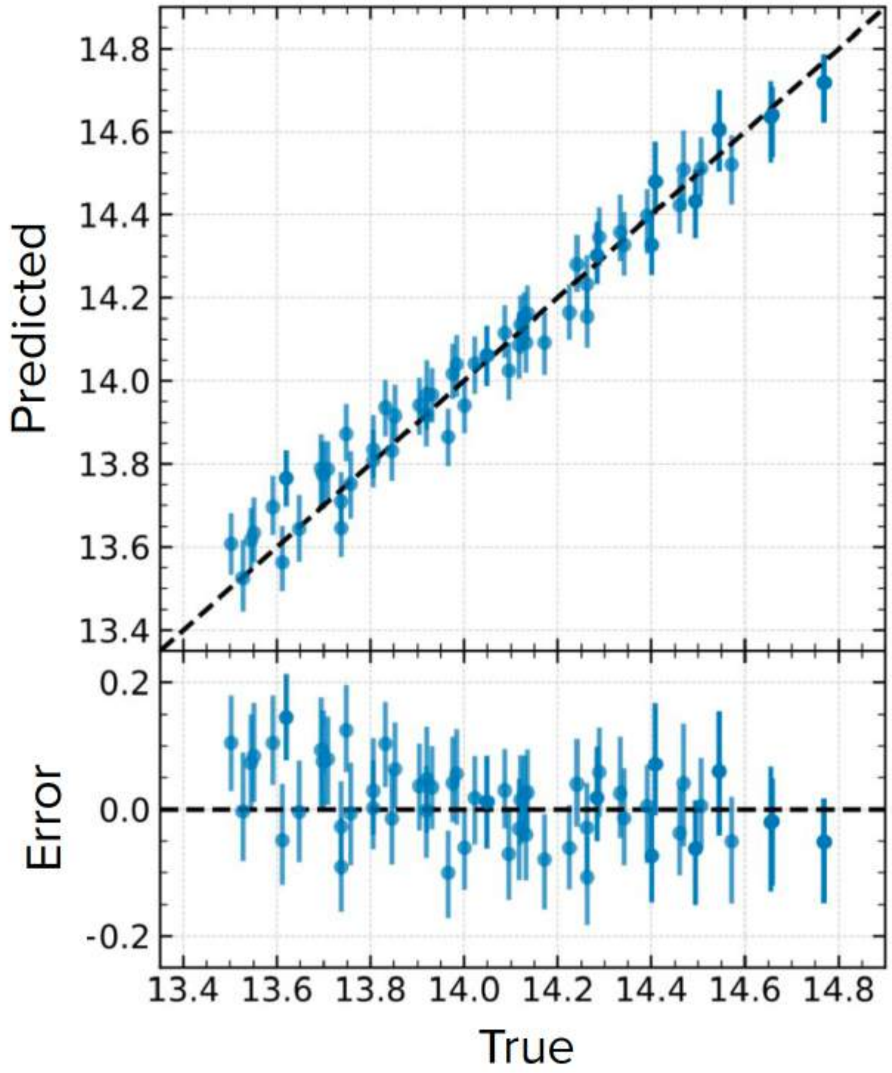
infer posterior estimate

$$\theta' \sim q_\phi(\theta \,|\, \mathbf{X}_{\text{obs}})$$

no guarantee it converges to the true posterior

Validation

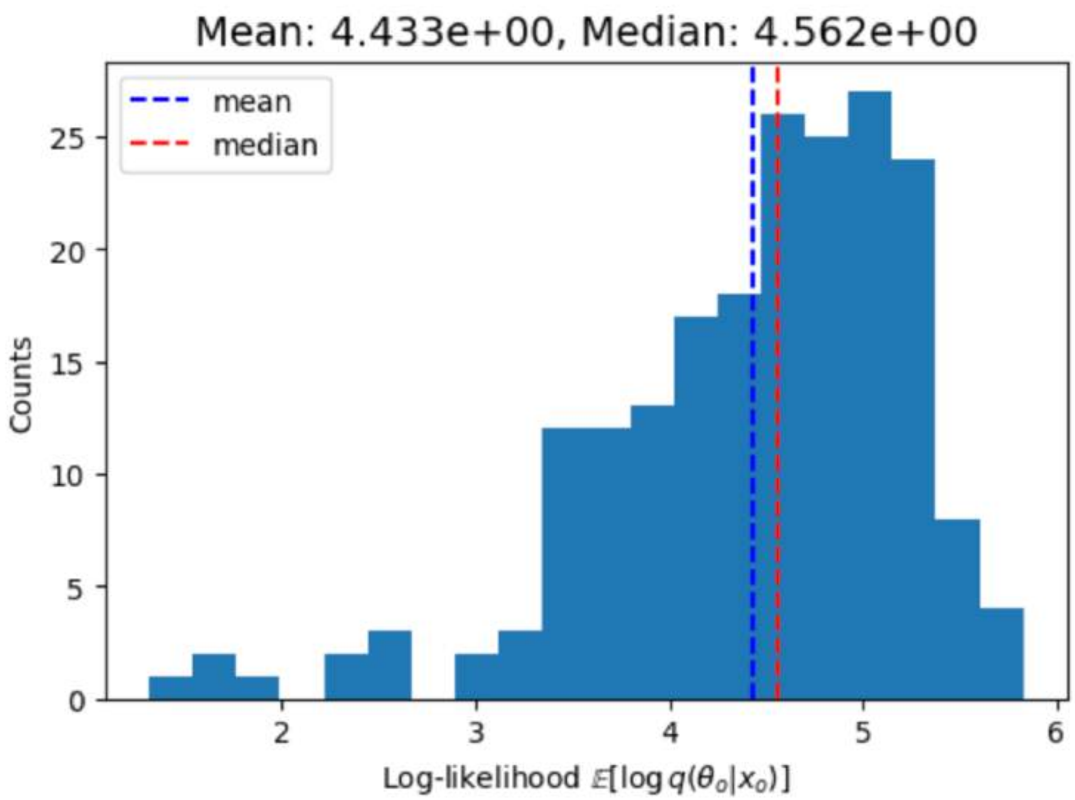




***adapted from M. Ho (SBI conference)**

Cumulative Likelihood of the Test Dataset

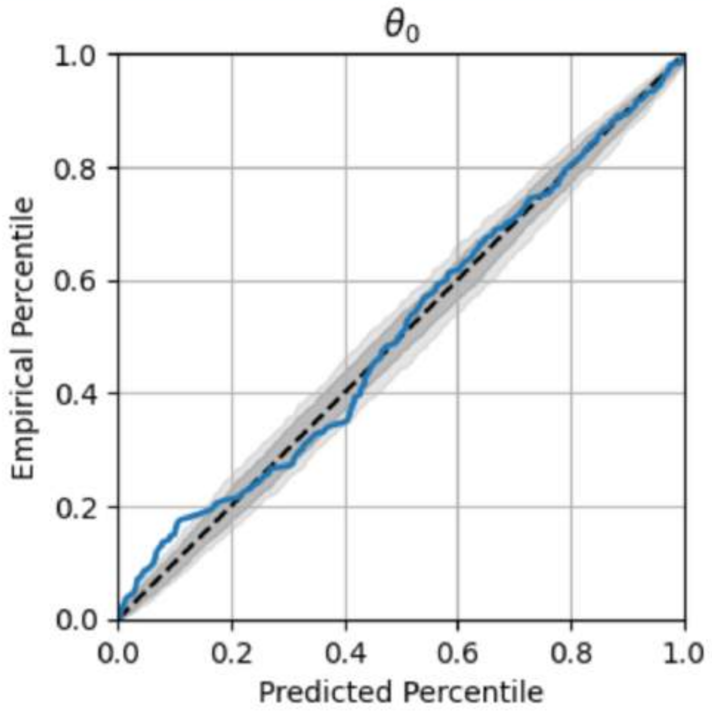
$$\prod_{i=1}^{N_{test}} \hat{P}(\theta_i | x_i)$$



Large values mean good predictive power

P-P plots

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



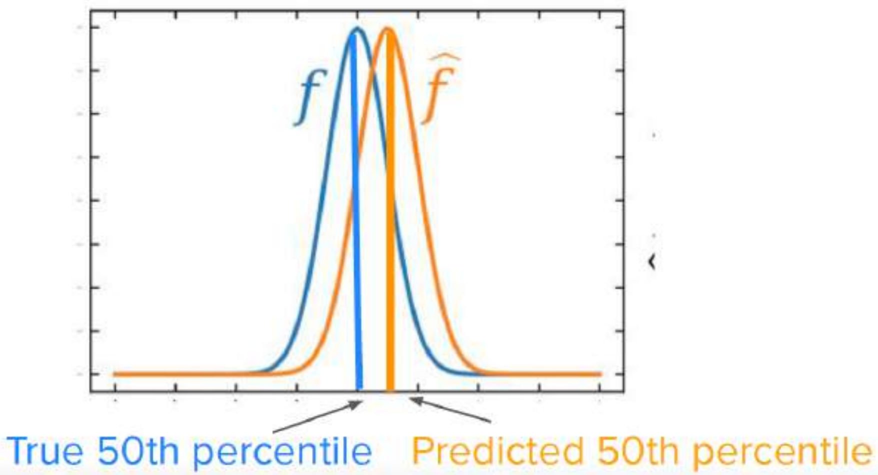
"What percentile level is my posterior model assigning to the true value, and with what frequency does this occur in the test set, e.g. **we should predict the true value below the 50-th percentile 50% of the time**"

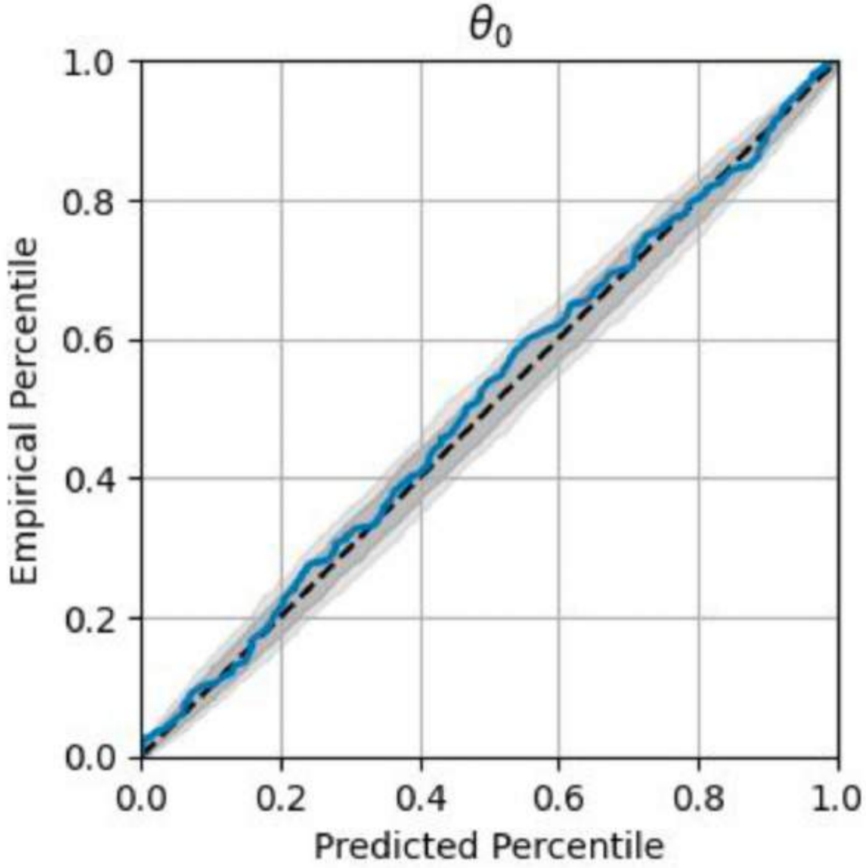
Ho+24

Validation

PIT tests / P-P plots

- Tests of marginal coverage

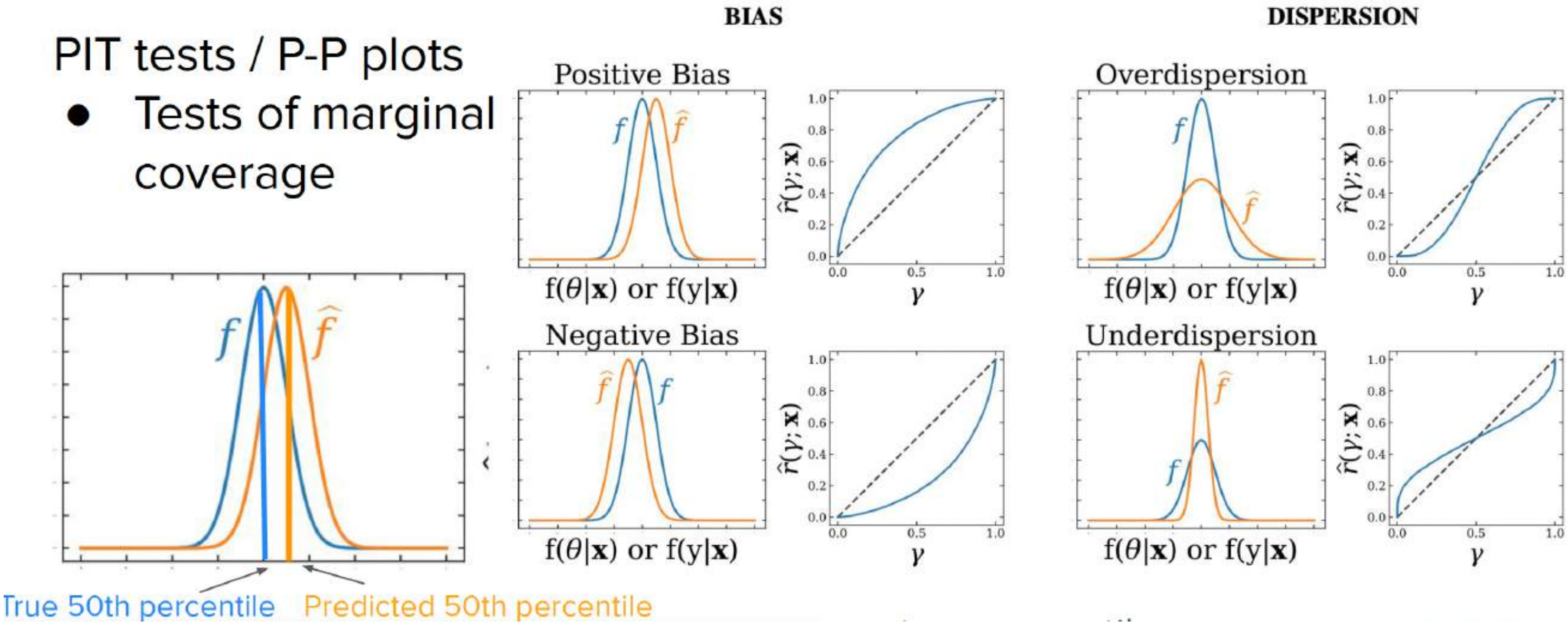




*adapted from M. Ho (SBI conference)

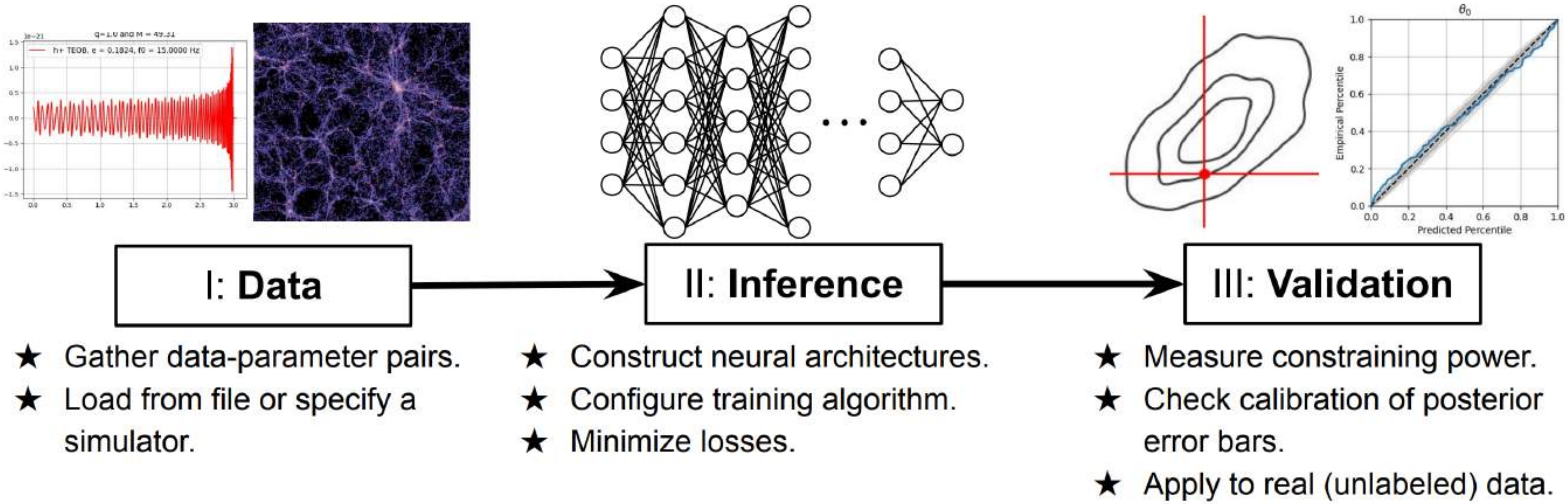
PIT tests / P-P plots

- Tests of marginal coverage



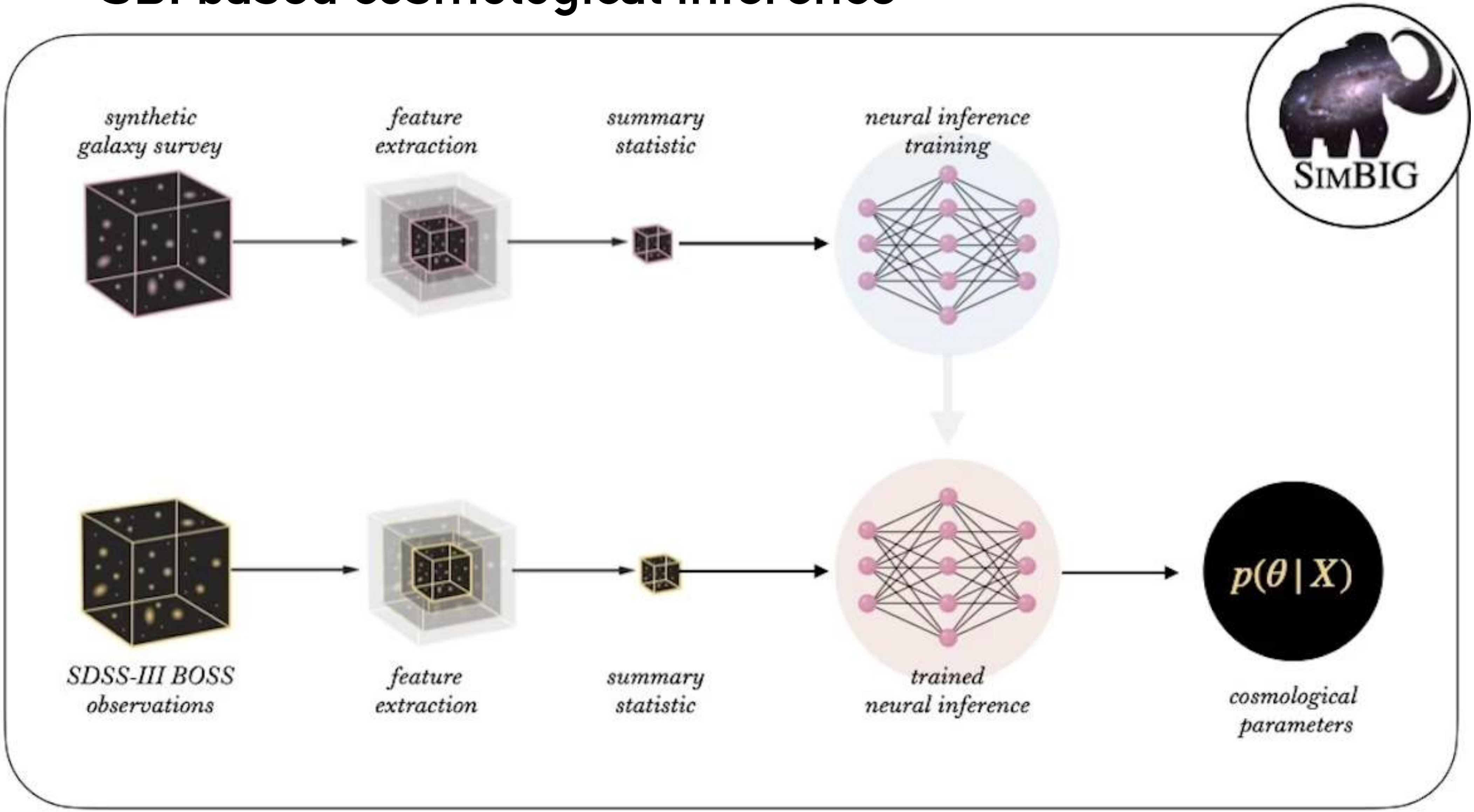
*adapted from M. Ho (SBI conference)

SBI pipeline



Ho+24

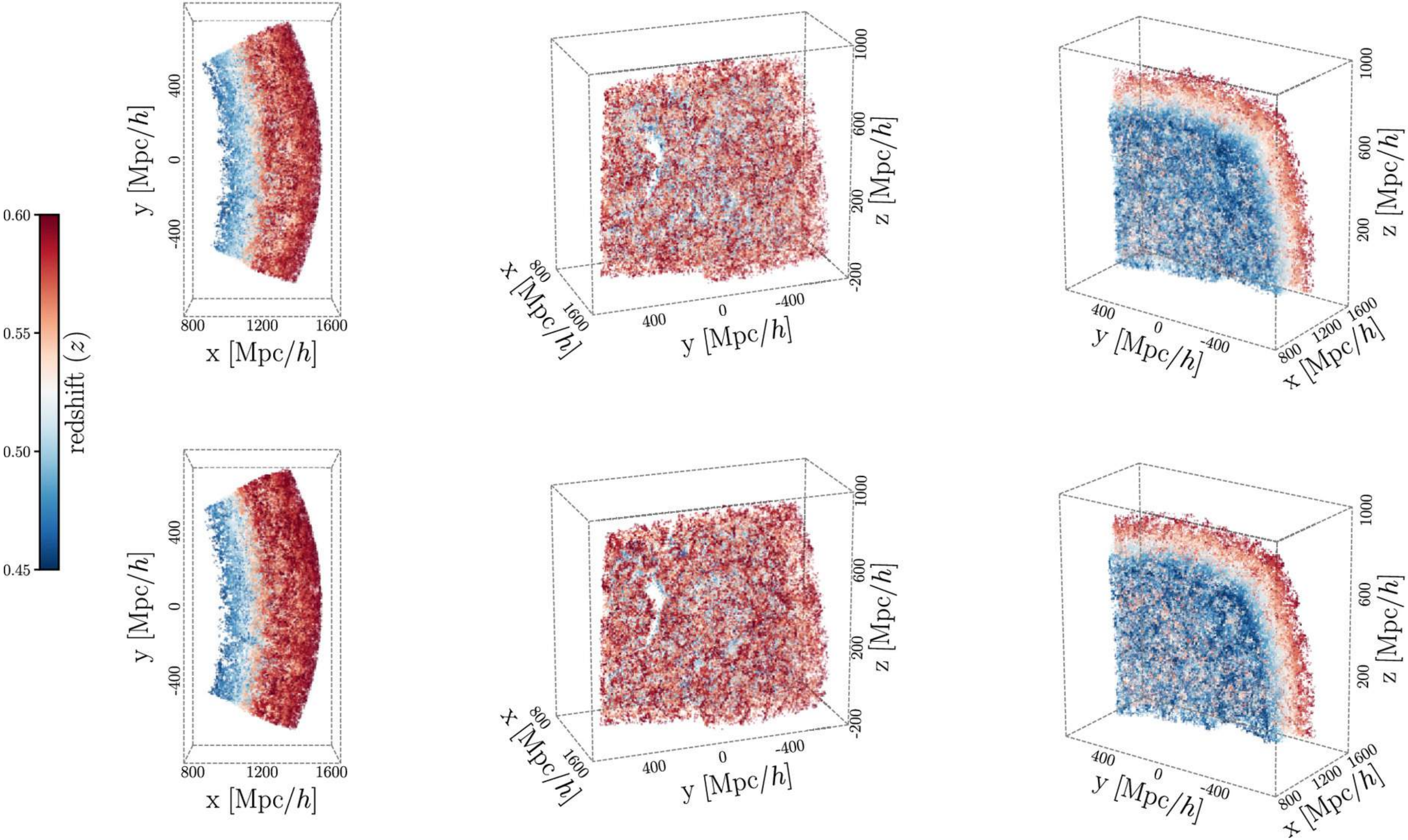
SBI based cosmological inference



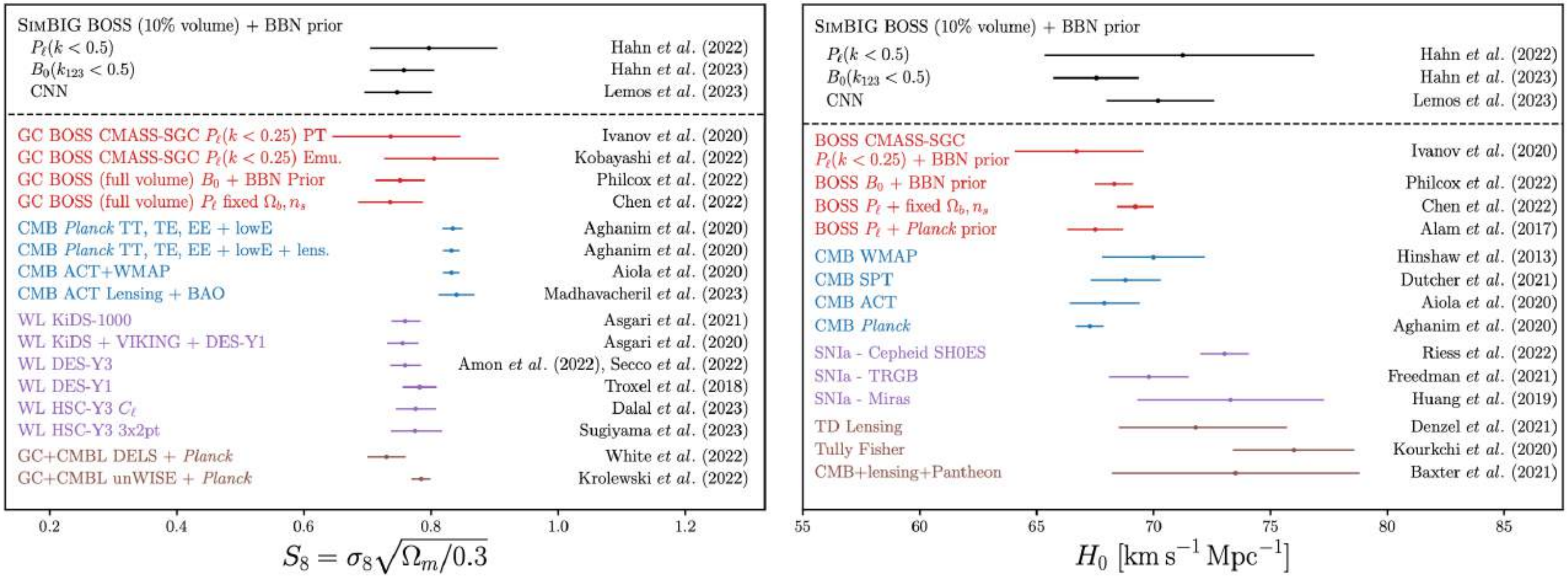
"If you can simulate, you can do inference"

Hahn+23

Towards Field Level Cosmological Inference

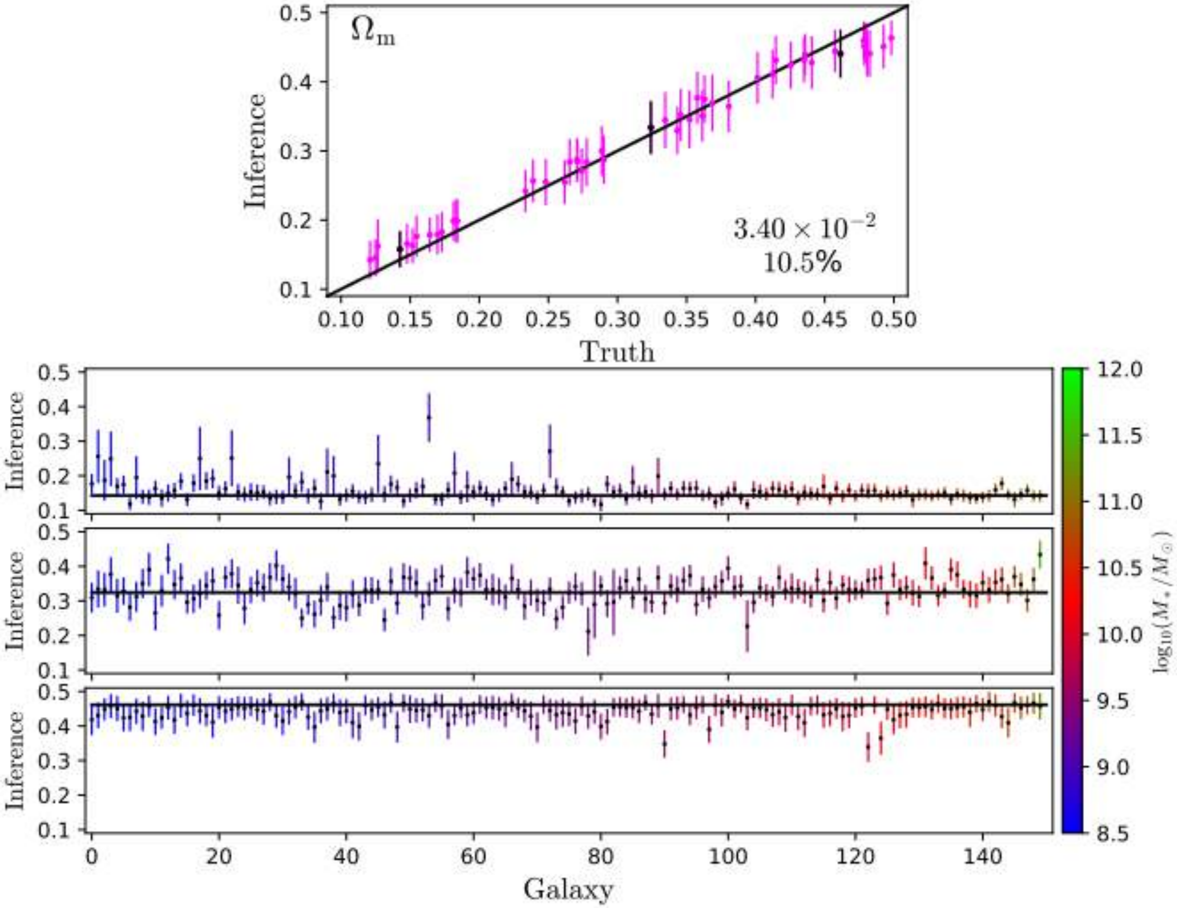


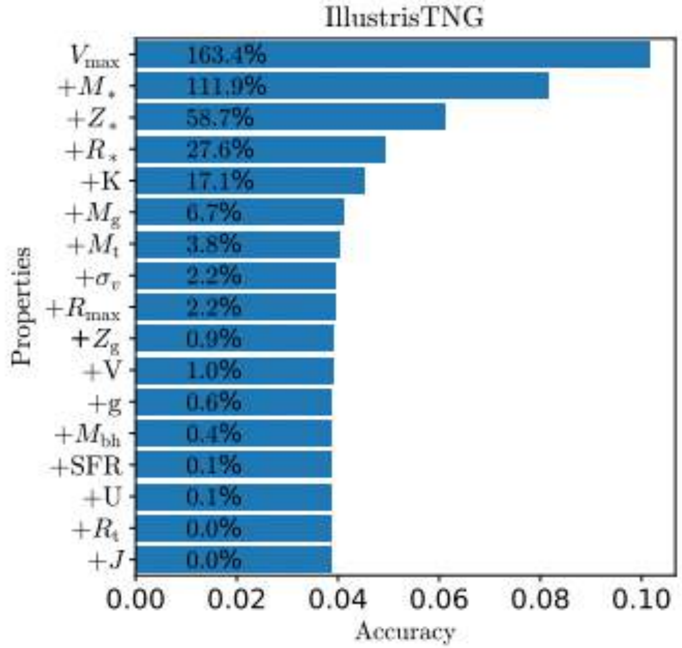
Hahn+23


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

Extra Material

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: score based models which try to estimate the gradient of the likelihood (score) instead of the 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.

***refer to the original paper by Song and Ermon 2019 (Much clearer than any blogpost)**

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**

One key ingredient is Langevin Dynamics:

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

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

$$\tilde{x_0} \sim \pi(x)$$

For epsilon small, T large:

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

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

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

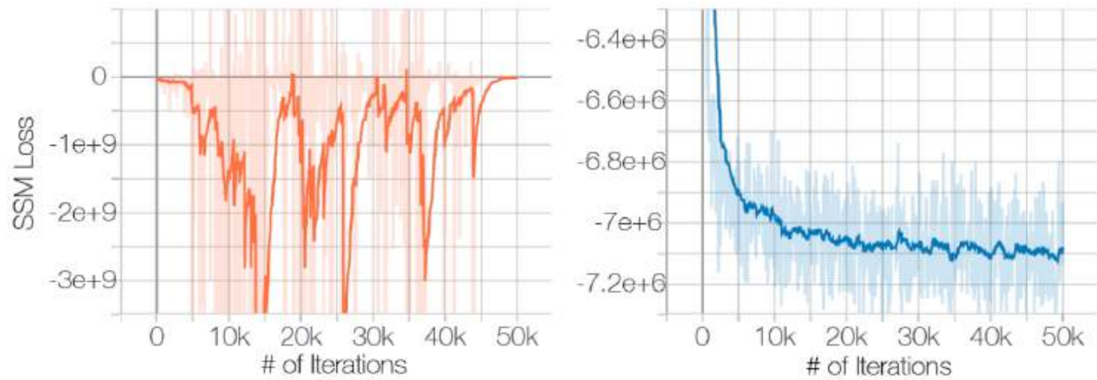
Perturbation with gaussian noise

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

If noise is small

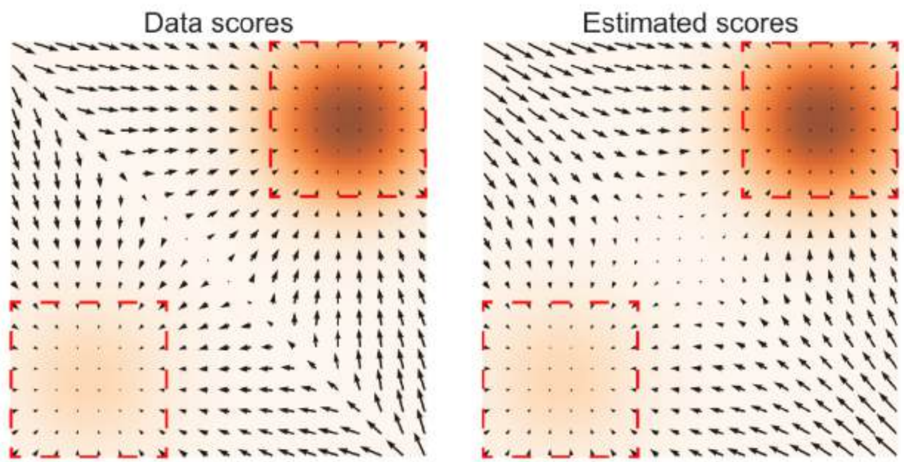
Estimating the score is difficult

1. Manifold Hypothesis



If the support of x is not the whole space, score estimation techniques fail

2. Low Density Regions



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

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:

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

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

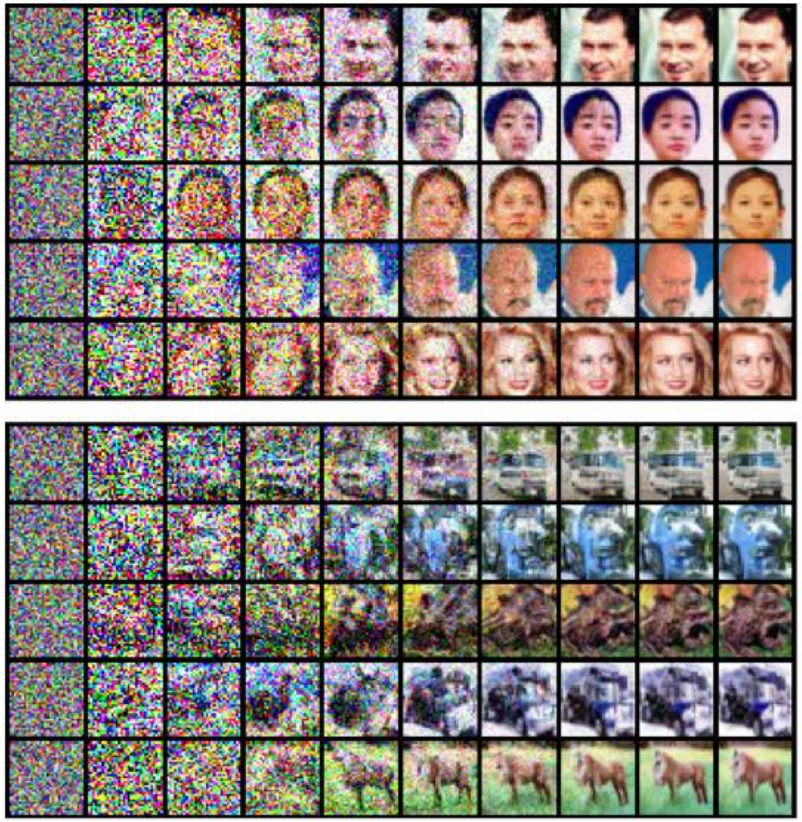
    $\tilde{\mathbf{x}}_t \leftarrow \tilde{\mathbf{x}}_{t-1} + \frac{\alpha_i}{2} \mathbf{s}_{\boldsymbol{\theta}}(\tilde{\mathbf{x}}_{t-1}, \sigma_i) + \sqrt{\alpha_i}\, \mathbf{z}_t$

  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 $q_{\sigma_L}(\mathbf{x})$.