PART V: DIMENSIONALITY REDUCTION, GENERATIVE MODELLING AND DENSITY ESTIMATION

TYPES OF MACHINE LEARNING

SUPERVISED

LEARNS A MAPPING FROM X [FEATURES] TO Y [LABELS]

$$P(Y|X)$$

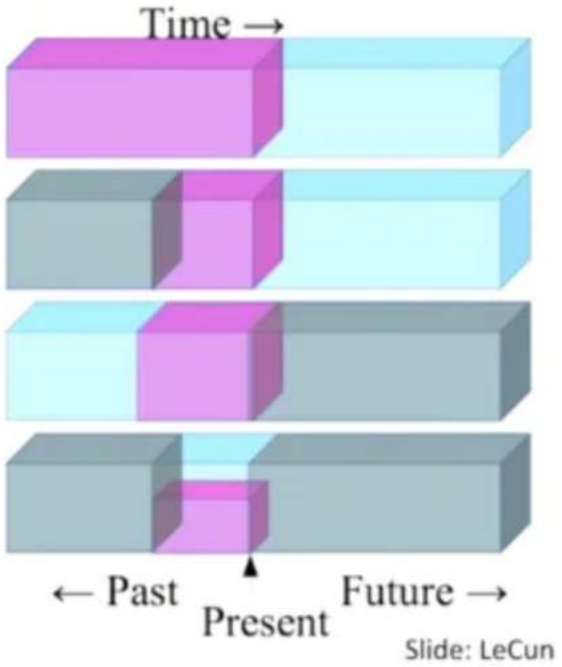
UNSUPERVISED AND SELF-SUPERVISED

NO LABELS - DISCOVER PATTERNS

$$P(X) \qquad P(\hat{Y}|X)$$

Self-Supervised Learning vs. Unsupervised

- Predict any part of the input from any other part.
- Predict the future from the past.
- Predict the future from the recent past.
- Predict the past from the present.
- Predict the top from the bottom.
- Predict the occluded from the visible
- Pretend there is a part of the input you don't know and predict that.



a self-supervised learning system creates a pretext task , which transforms the problem into a "supervised" problem: predict parts of its inputs based on the other parts of its inputs

Types of Machine Learning

Approach	**Discriminative**		**Generative**	
Data type	Target Function	Method	Target Function	Method
Labels	$p(y\|x)$	All supervised networks with bottleneck	$p(x\|y)$	Conditional generative models
No Labels	$p(\hat{y}\|x)$	Self-supervised learning	$p(x)$	Generative models, Autoencoders

There are 3 major applications in astronomy:

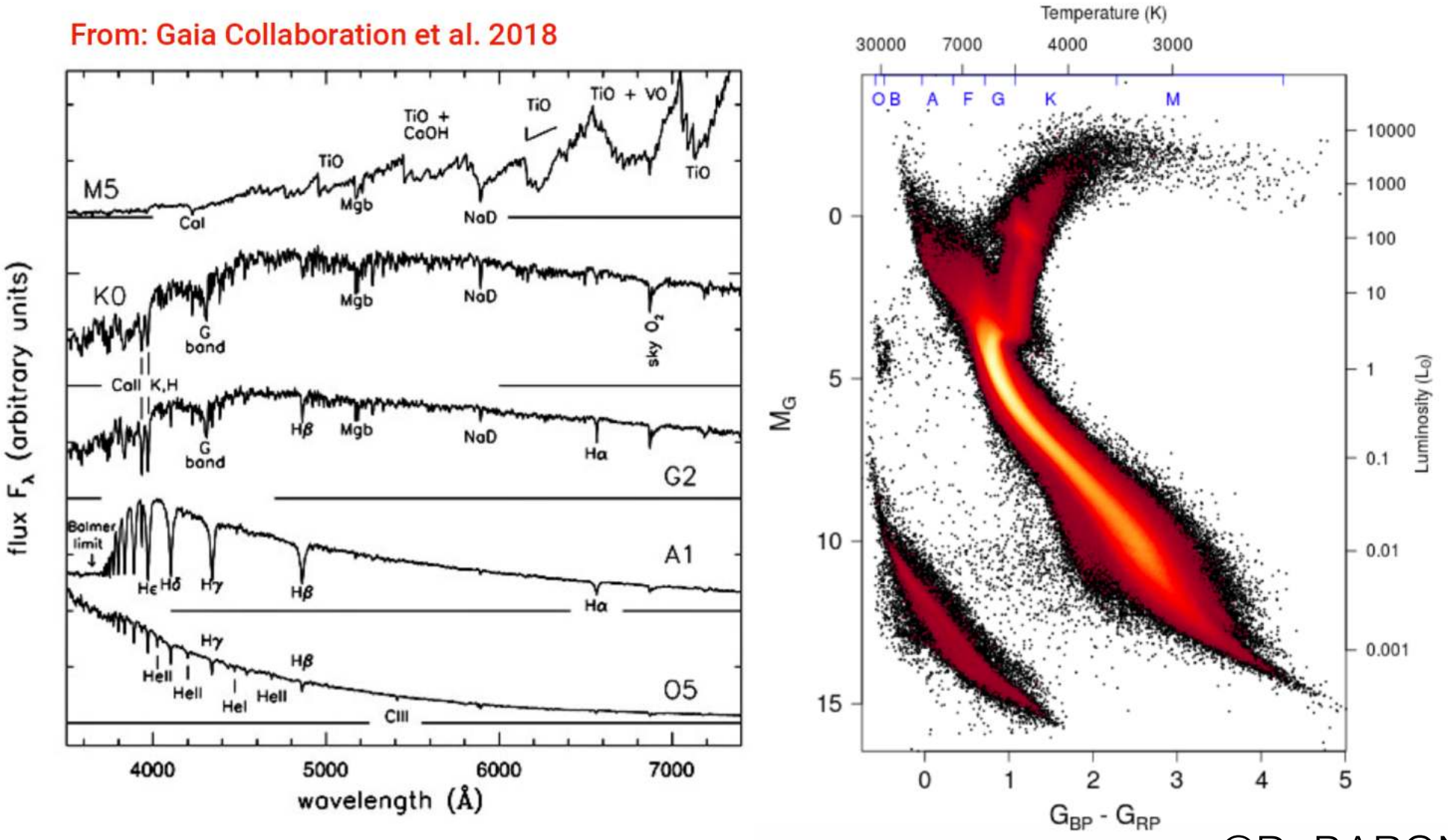
- **DIMENSIONALITY REDUCTION:** HOW CAN I REPRESENT MY COMPLEX DATA MORE EFFICIENTLY TO GET NEW INSIGHTS INTO ITS STRUCTURE?

- **SAMPLING:** HOW CAN I INTERPOLATE / EXTRAPOLATE A (SMALL, SPARSE) DATASET TO GENERATE NEW DATA SAMPLED FROM THE SAME (UNKNOWN) DISTRIBUTION?

- **LIKELIHOOD EVALUATION:** WHAT IS THE PROBABILITY THAT A NEW OBSERVATION IS DRAWN FROM THE SAME (UNKNOWN) DISTRIBUTION AS SOME REFERENCE (SMALL, SPARSE) DATASET?

1. Dimensionality Reduction

From: Gaia Collaboration et al. 2018



@D. BARON

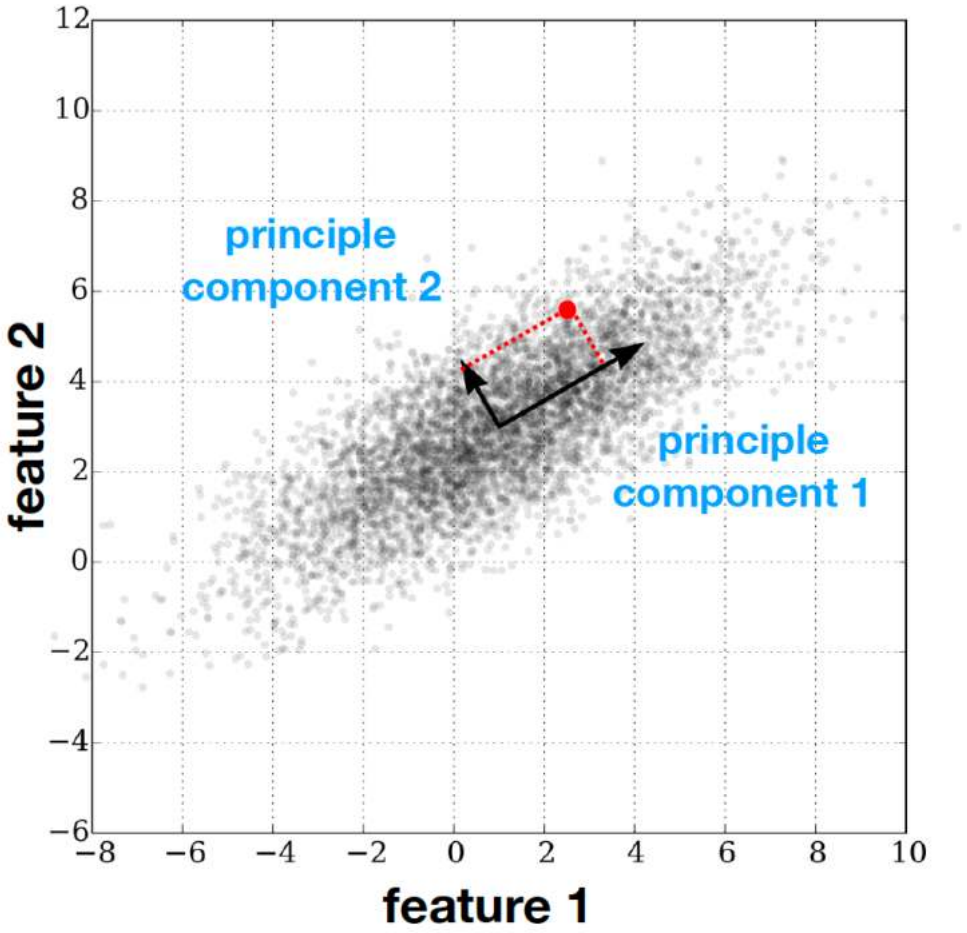
THE STELLAR MAIN SEQUENCE IS A DIMENSIONALITY REDUCTION

PRINCIPAL COMPONENT ANALYSIS

PCA CONVERT A SET OF (CORRELATED) VARIABLES INTO A SET OF VALUES LINEARLY UNCORRELATED

1. THE FIRST PRINCIPLE COMPONENT ("PROTOTYPE"), HAS THE LARGEST POSSIBLE VARIANCE

2. THE FOLLOWING COMPONENTS HAVE THE LARGEST VARIABLES WITH THE ADDITIONAL CONSTRAIN THAT THEY ARE ORTHOGONAL TO THE PRECEDING COMPONENTS



PRINCIPAL COMPONENT ANALYSIS

1. COMPUTE THE **COVARIANCE MATRIX** OF YOUR DATA

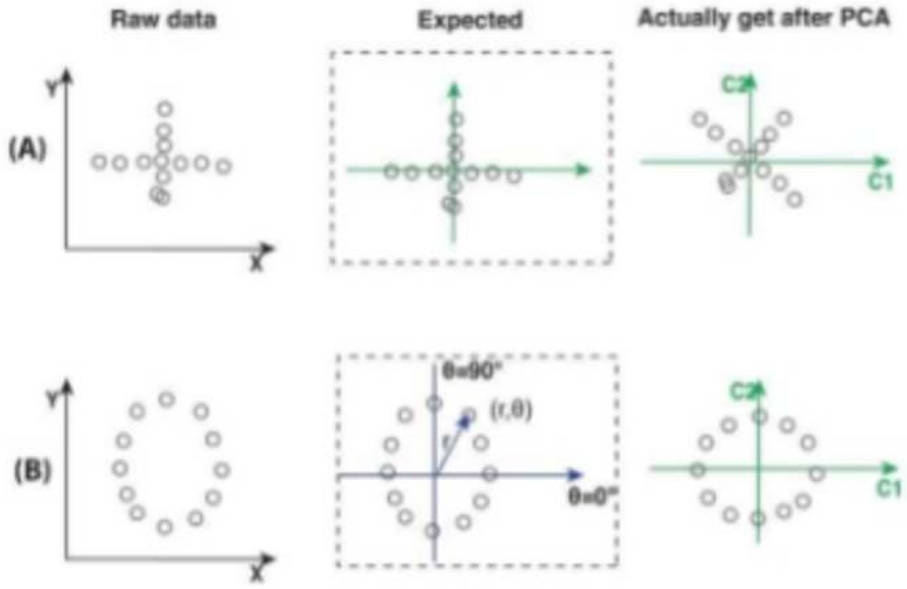
2. COMPUTE THE **EIGENVALUES AND EIGENVECTORS** OF THE **COVARAINCE MATRIX**

3.THE **EIGENVECTORS** PROVIDE THE DIRECTION OF MAXIMUM VARIANCE AND THE **EIGENVALUES** THE ‘IMPORTANCE’ OF THAT PARTICULAR FEATURE

LIMITATIONS OF PCA

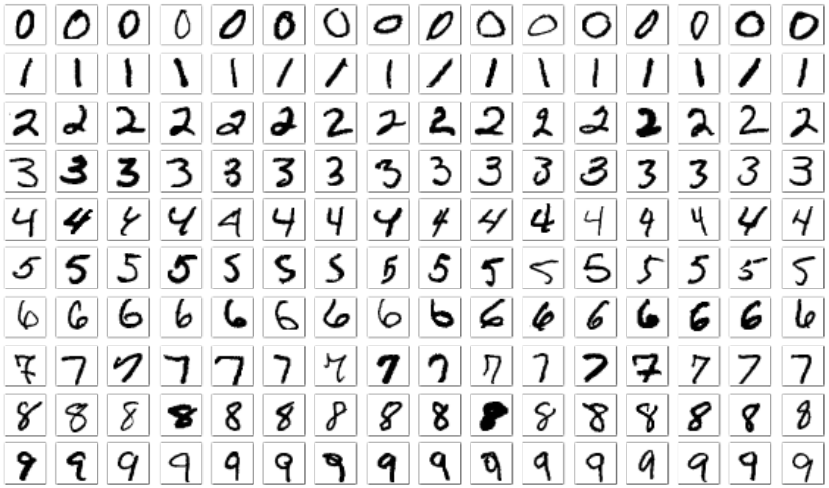
PCA APPLY LINEAR TRANSFORMATIONS

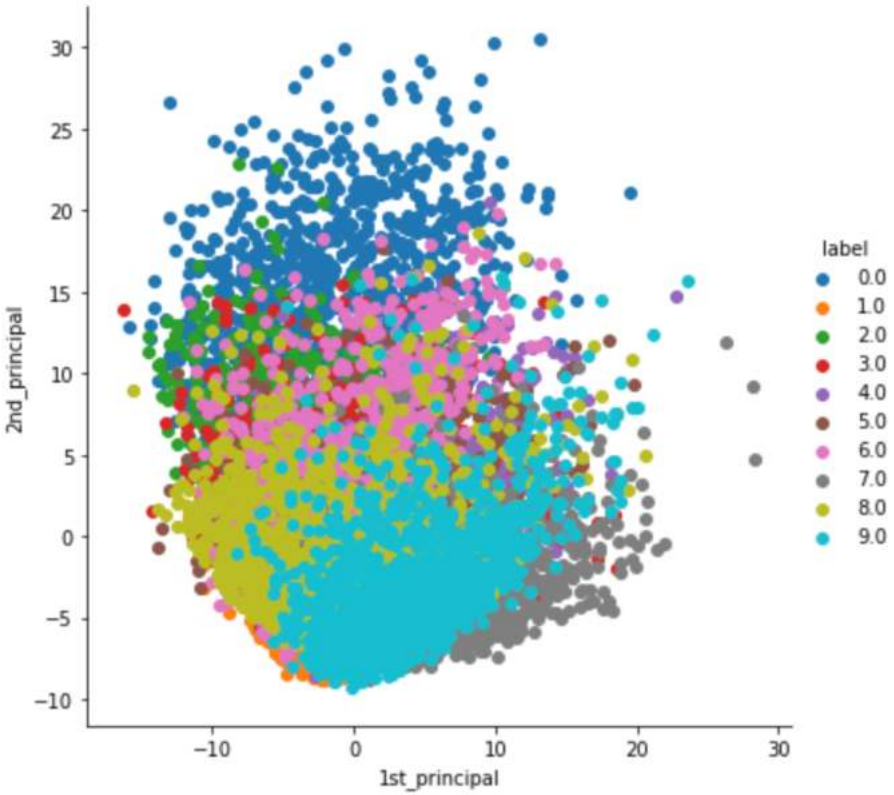
SINCE WE USE THE COVARIANCE MATRIX, IT ASSUMES THAT THE DATA FOLLOWS A **MULTIDIMENSIONAL NORMAL DISTRIBUTION**



LIMITATIONS OF PCA

AND DATA IS NOT ALWAYS GAUSSIAN….

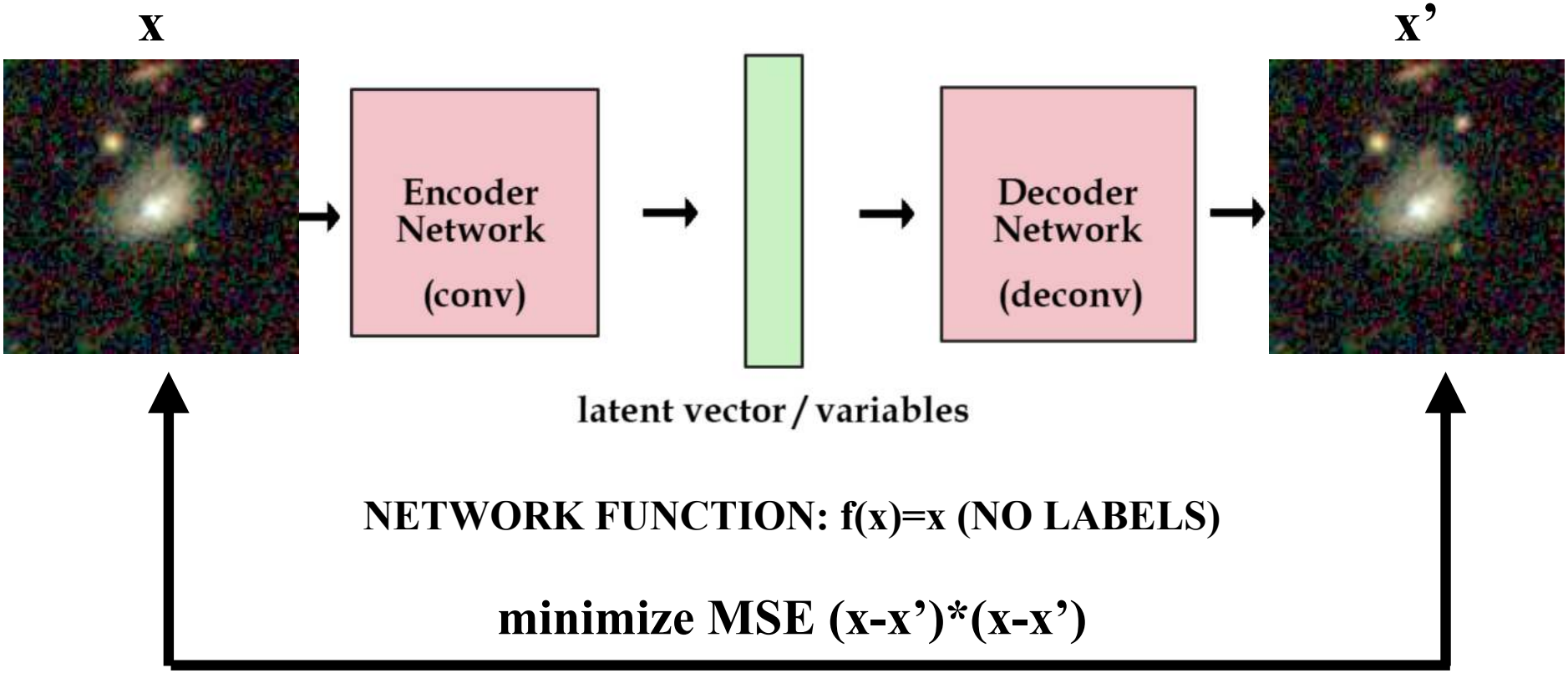




https://medium.com/analytics-vidhya/principal-component-analysis-pca-with-code-on-mnist-dataset-da7de0d07c22

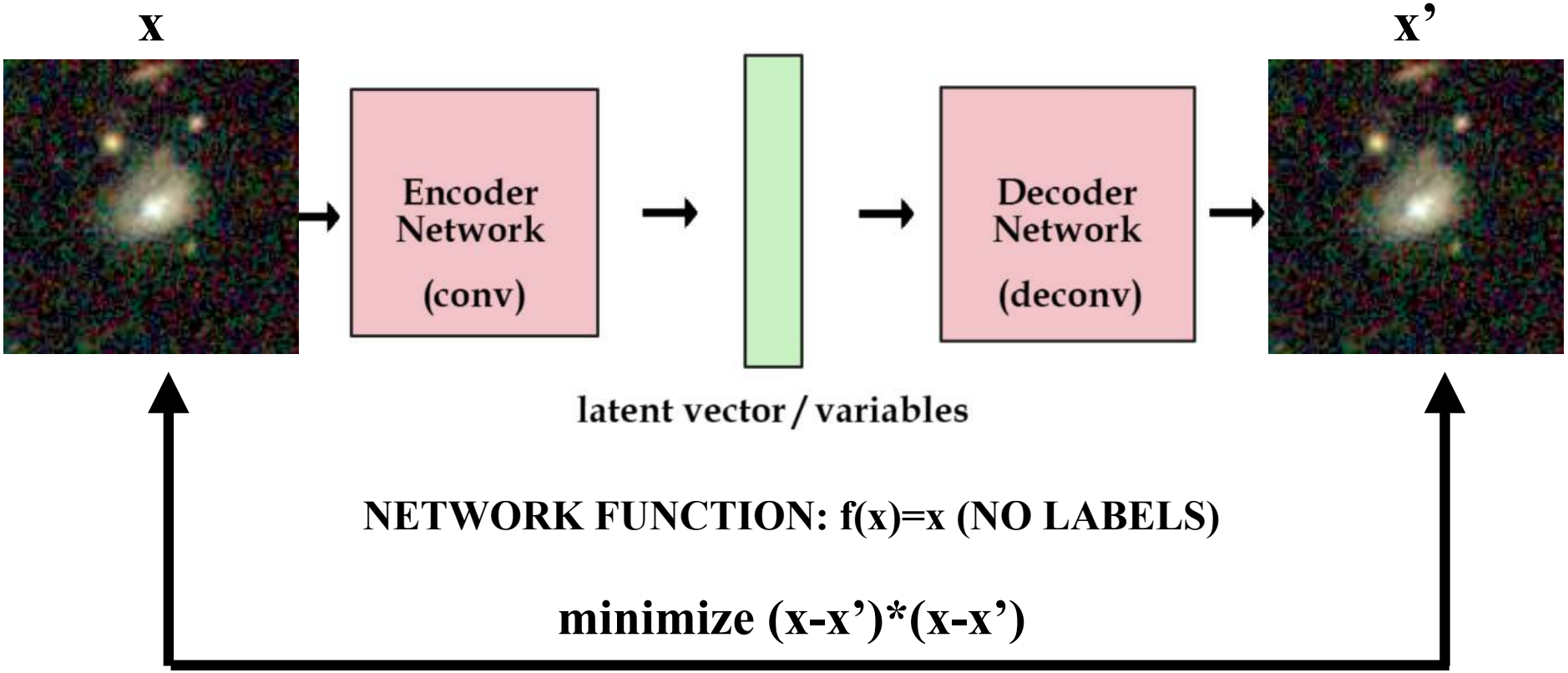
CAN WE GENERALIZE THAT?

CONVOLUTIONAL AUTO-ENCODER



AN AUTO-ENCODER IS ANY NETWORK WITH IDENTICAL INPUT AND OUTPUT

CONVOLUTIONAL AUTO-ENCODER

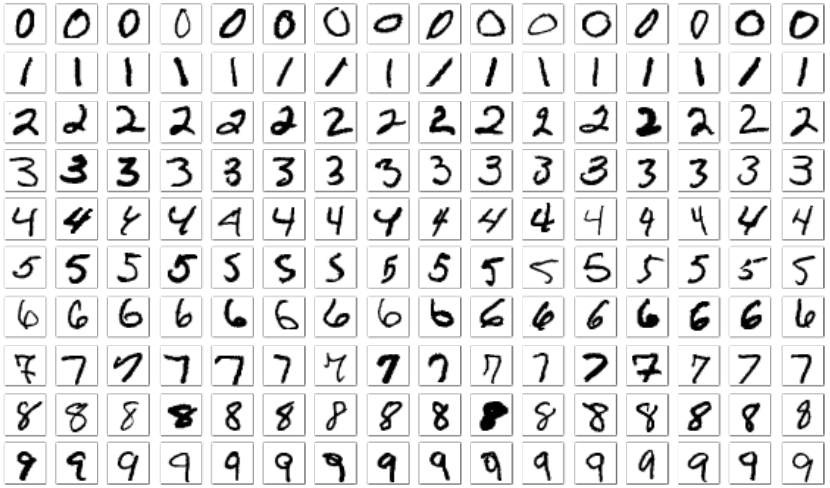


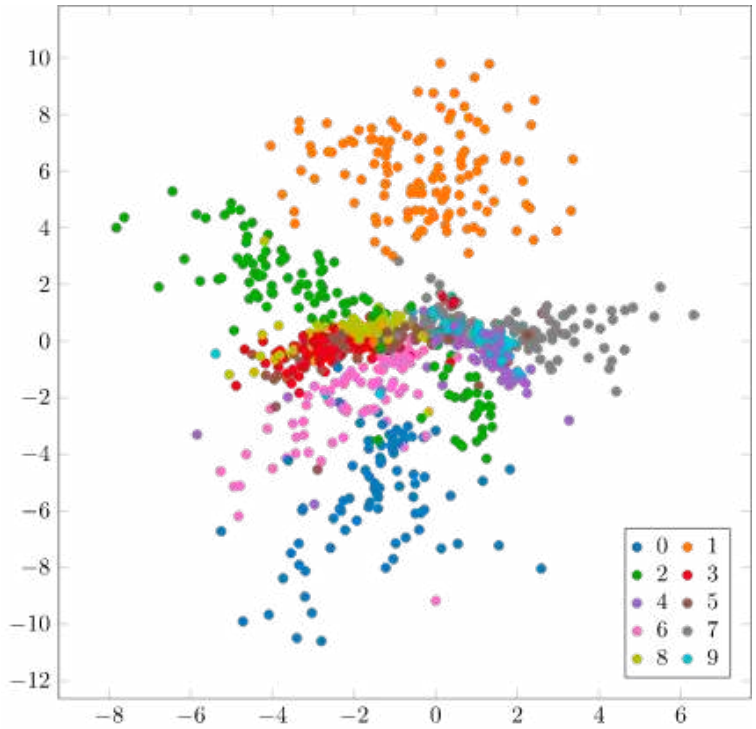
BY REDUCING THE DIMENSIONALITY IN THE LATENT SPACE WE FORCE THE NETWORK TO LEARN A REPRESENTATION OF THE INPUT DATA IN A LOWER DIMENSIONALITY SPACE

* 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?

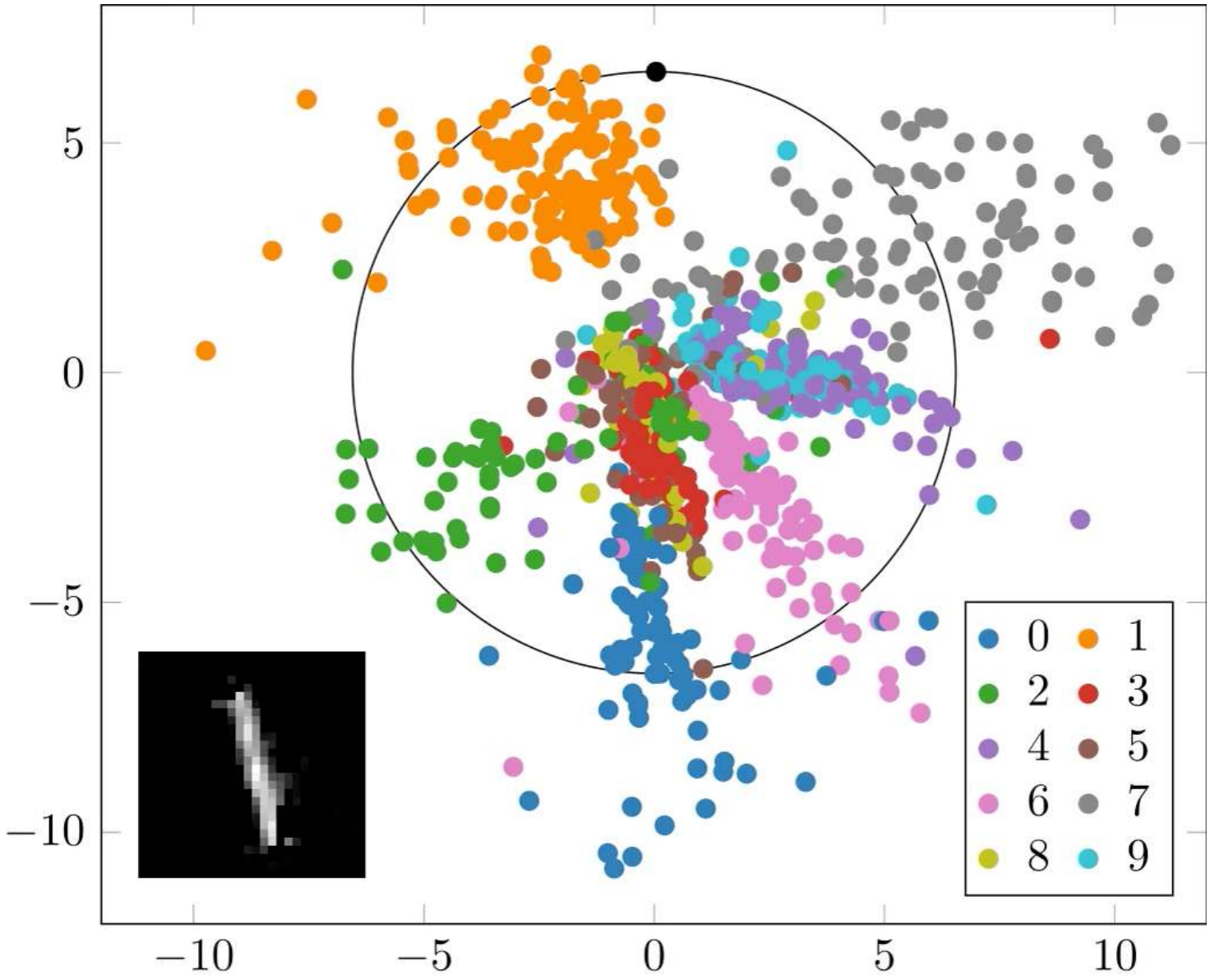
AUTOENCODER REPRESENTATION OF MNIST



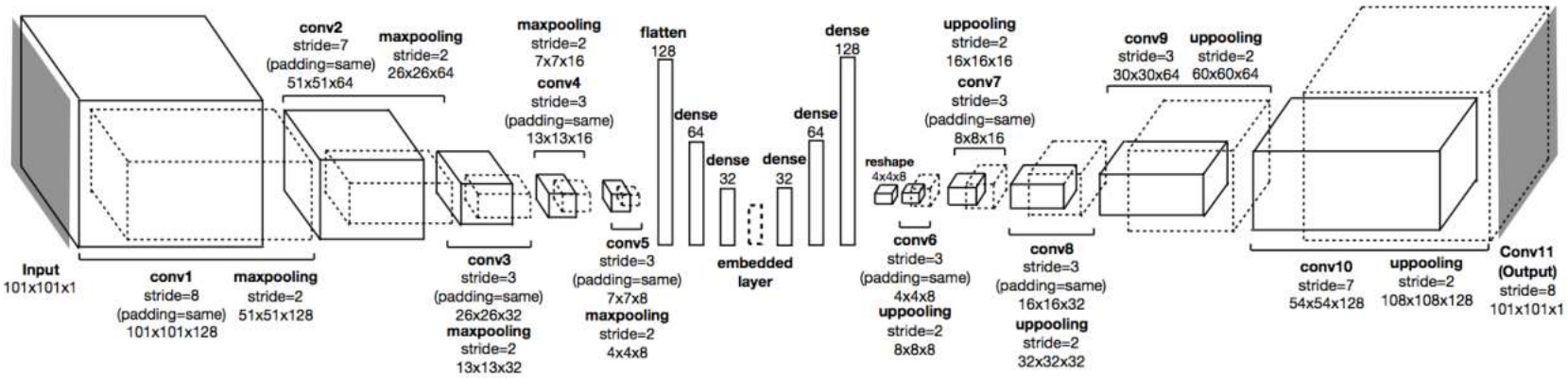


https://gertjanvandenburg.com/blog/autoencoder/

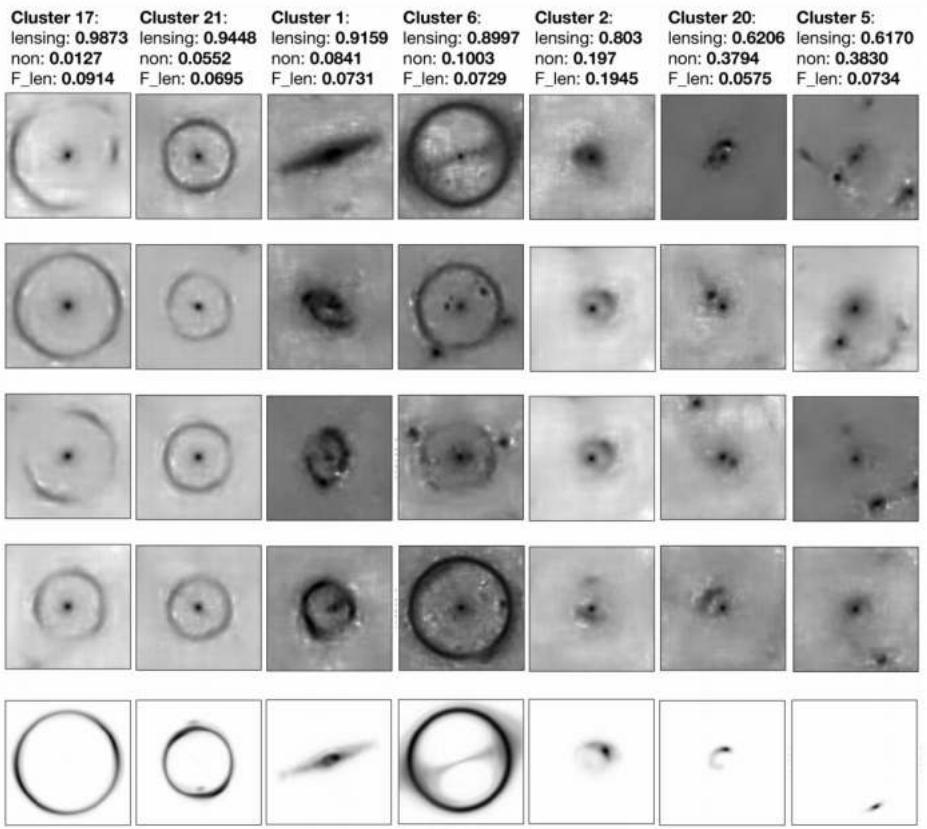
AUTOENCODER REPRESENTATION OF MNIST

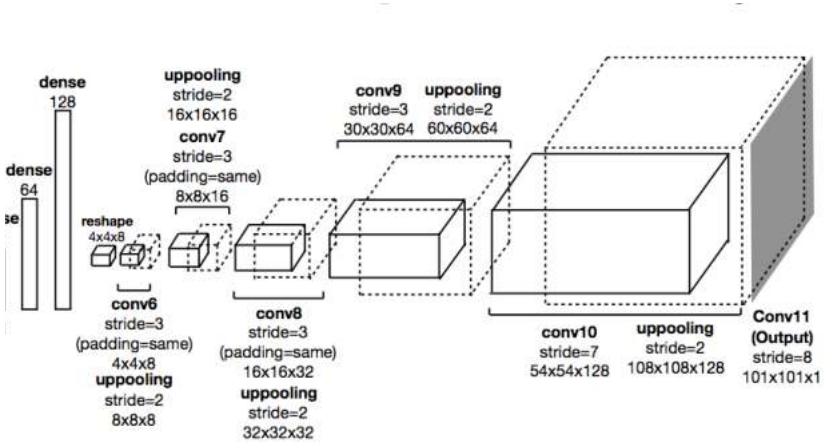


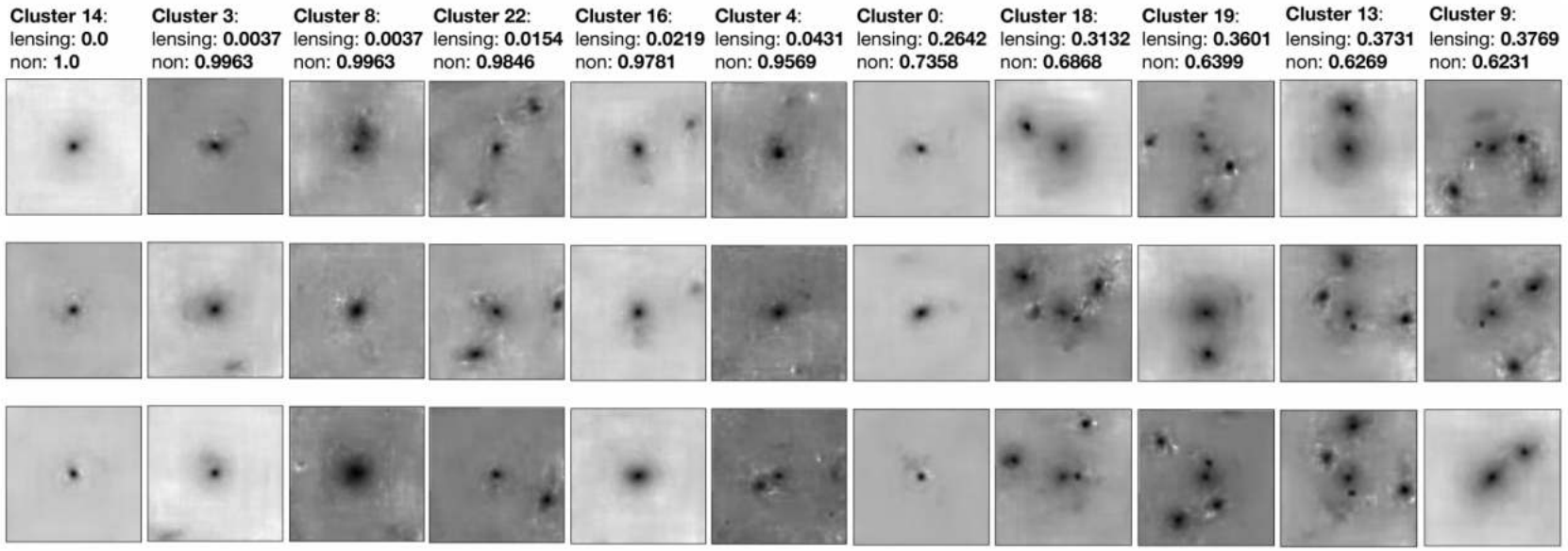
https://gertjanvandenburg.com/blog/autoencoder/


Input
101x101x1
conv1
stride=8
(padding=same)
101x101x128
maxpooling
stride=2
51x51x128
conv2
stride=7
(padding=same)
51x51x64
maxpooling
stride=2
26x26x64
conv3
stride=3
(padding=same)
26x26x32
maxpooling
stride=2
13x13x32
maxpooling
stride=2
7x7x16
conv4
stride=3
(padding=same)
13x13x16
conv5
stride=3
(padding=same)
7x7x8
maxpooling
stride=2
4x4x8
flatten
128
dense
64
dense
32
embedded
layer
dense
32
dense
64
dense
128
reshape
4x4x8
conv6
stride=3
(padding=same)
4x4x8
uppooling
stride=2
8x8x8
uppooling
stride=2
16x16x16
conv7
stride=3
(padding=same)
8x8x16
conv8
stride=3
(padding=same)
16x16x32
uppooling
stride=2
32x32x32
conv9
stride=3
30x30x64
uppooling
stride=2
60x60x64
conv10
stride=7
54x54x128
uppooling
stride=2
108x108x128
Conv11
(Output)
stride=8
101x101x1

Cheng+20


Cluster 17: lensing: 0.9873 non: 0.0127 F_len: 0.0914
Cluster 21: lensing: 0.9448 non: 0.0552 F_len: 0.0695
Cluster 1: lensing: 0.9159 non: 0.0841 F_len: 0.0731
Cluster 6: lensing: 0.8997 non: 0.1003 F_len: 0.0729
Cluster 2: lensing: 0.803 non: 0.197 F_len: 0.1945
Cluster 20: lensing: 0.6206 non: 0.3794 F_len: 0.0575
Cluster 5: lensing: 0.6170 non: 0.3830 F_len: 0.0734


dense
128
dense
64
reshape
4x4x8
conv6
stride=3
(padding=same)
4x4x8
uppooling
stride=2
8x8x8
uppooling
stride=2
16x16x16
conv7
stride=3
(padding=same)
8x8x16
conv8
stride=3
(padding=same)
16x16x32
uppooling
stride=2
32x32x32
conv9
stride=3
30x30x64
uppooling
stride=2
60x60x64
conv10
stride=7
54x54x128
uppooling
stride=2
108x108x128
Conv11
(Output)
stride=8
101x101x1


Cluster 14: lensing: 0.0 non: 1.0
Cluster 3: lensing: 0.0037 non: 0.9963
Cluster 8: lensing: 0.0037 non: 0.9963
Cluster 22: lensing: 0.0154 non: 0.9846
Cluster 16: lensing: 0.0219 non: 0.9781
Cluster 4: lensing: 0.0431 non: 0.9569
Cluster 0: lensing: 0.2642 non: 0.7358
Cluster 18: lensing: 0.3132 non: 0.6868
Cluster 19: lensing: 0.3601 non: 0.6399
Cluster 13: lensing: 0.3731 non: 0.6269
Cluster 9: lensing: 0.3769 non: 0.6231

Cheng+20

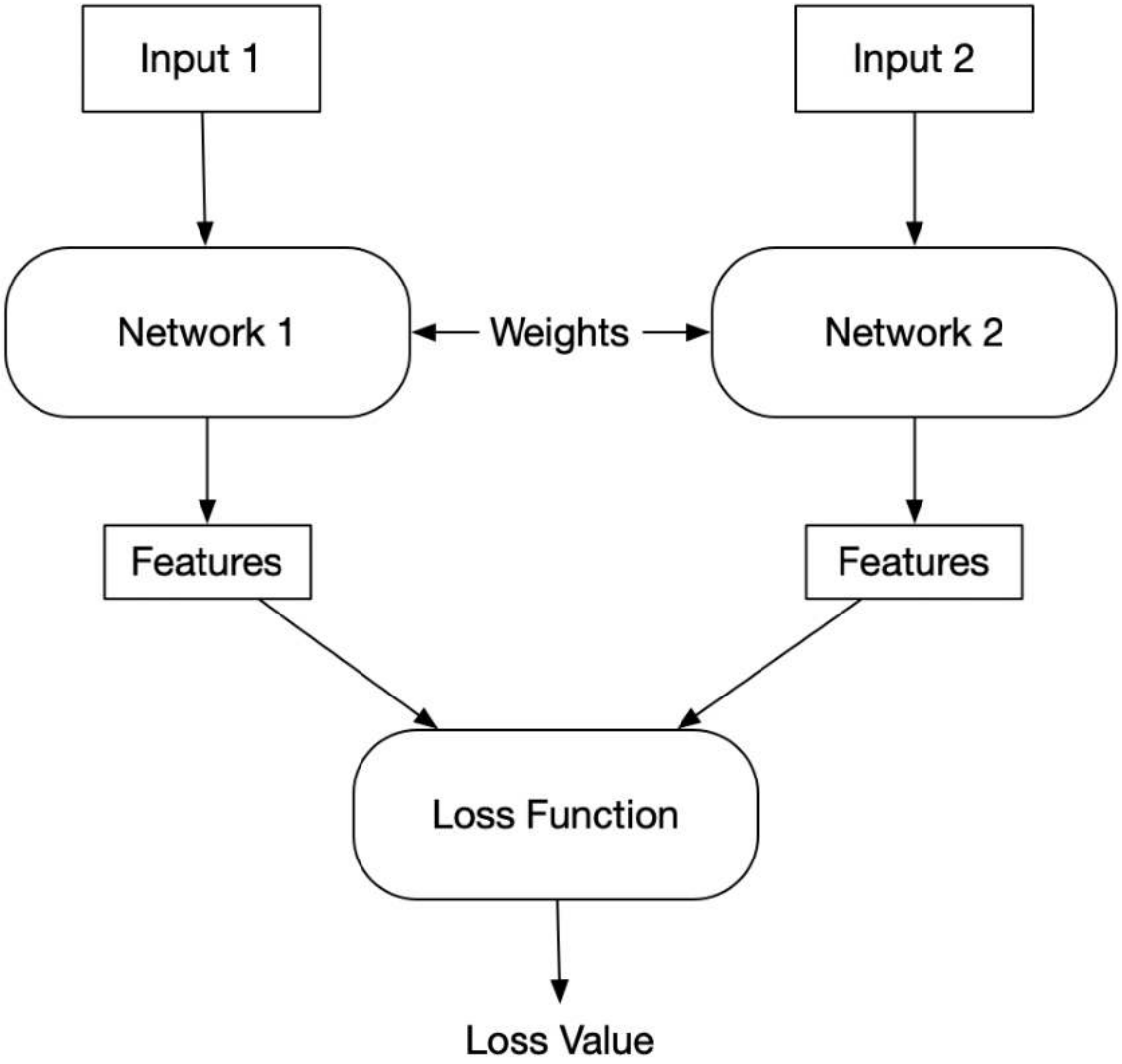
SELF-SUPERVISED CONTRASTIVE LEARNING

Humans tend to identify objects without remembering all the details, by creating some abstract representations which are used to identify new objects of similar time.

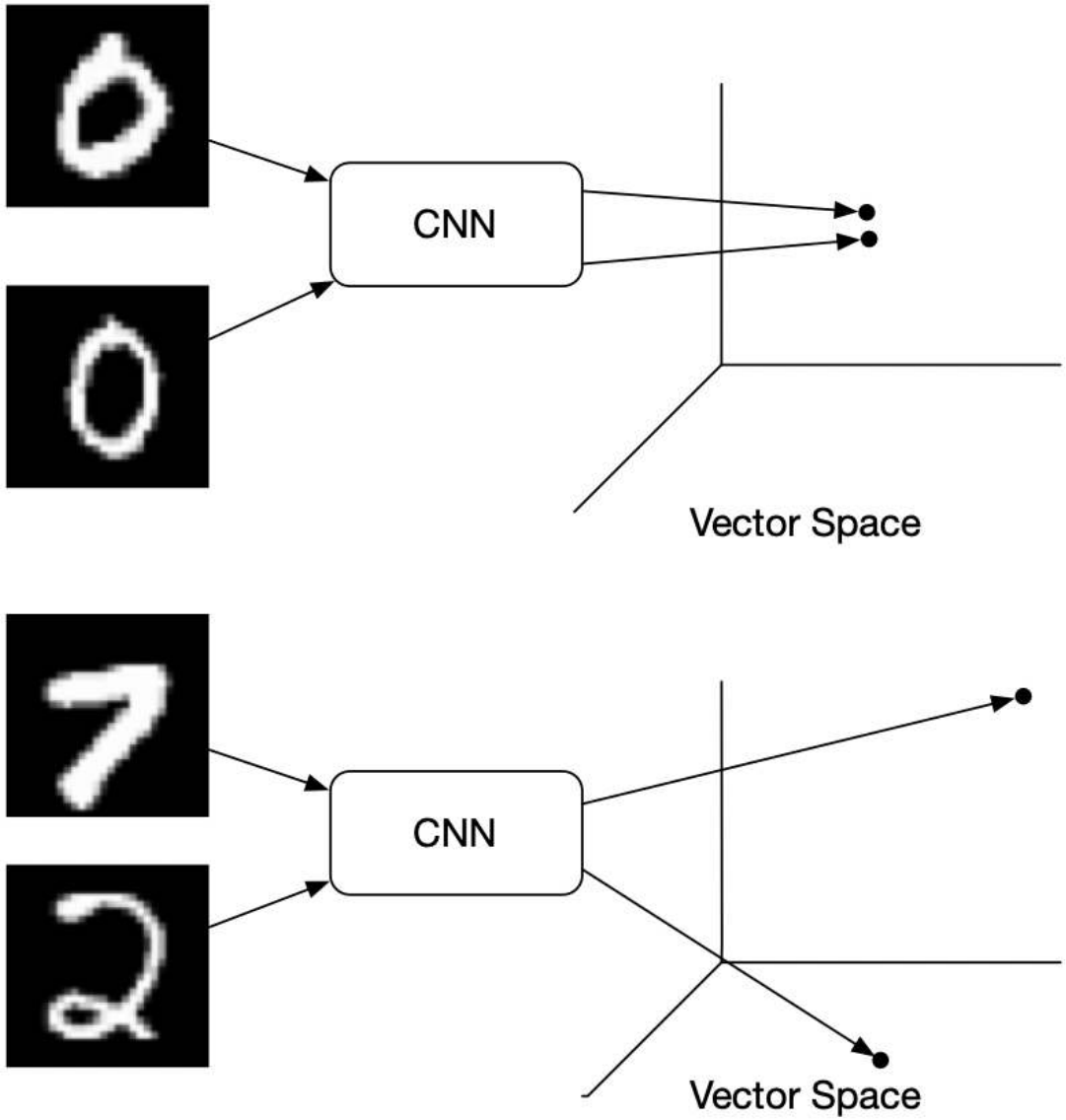
The main goal of self-supervised contrastive learning is to create and generalize these representations.

Therefore, it aims at building some general representations that can be used for other “downstream tasks”

Siamese networks are two networks sharing weights and common loss function



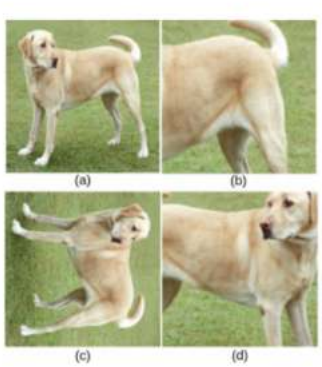
In a supervised setting, similar objects are used to be together by simply using a crossentropy loss

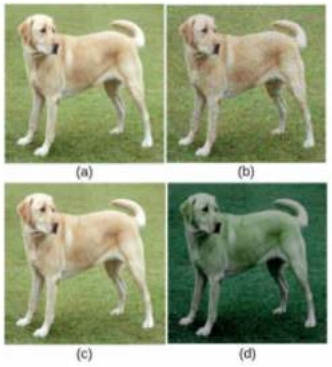


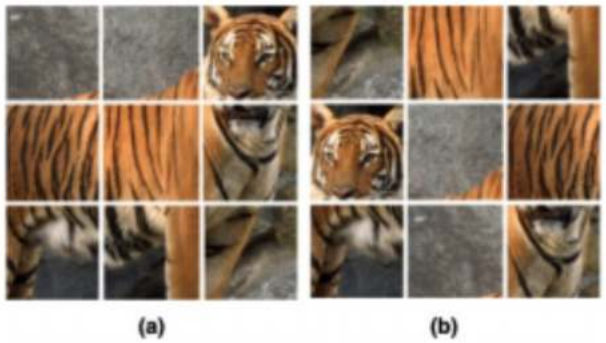
In most cases, labels are not available.

How do you recognise similar objects?

What would be the loss function?





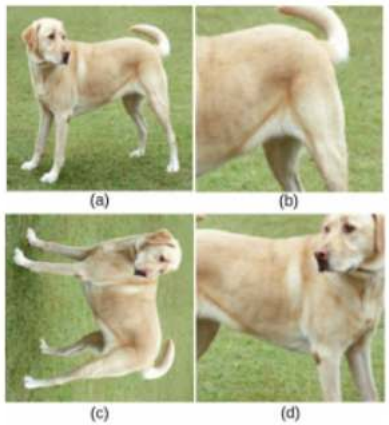


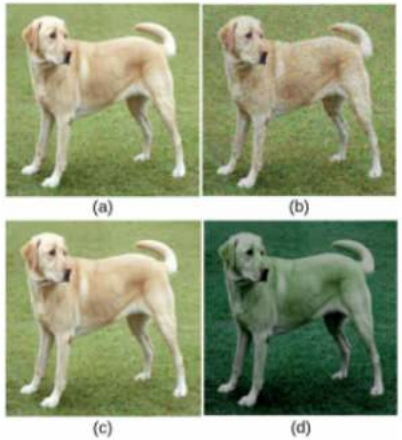
In most cases, labels are not available.

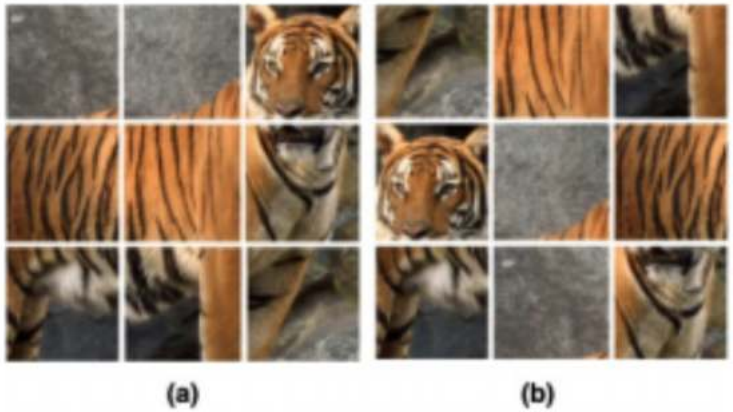
How do you recognise similar objects?

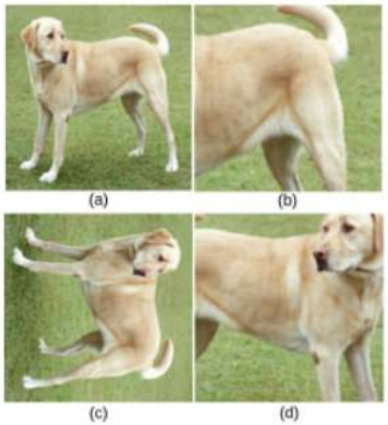
What would be the loss function?

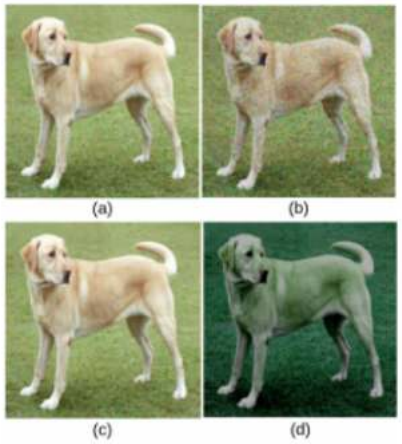
We create“pretext tasks” from a unique image:

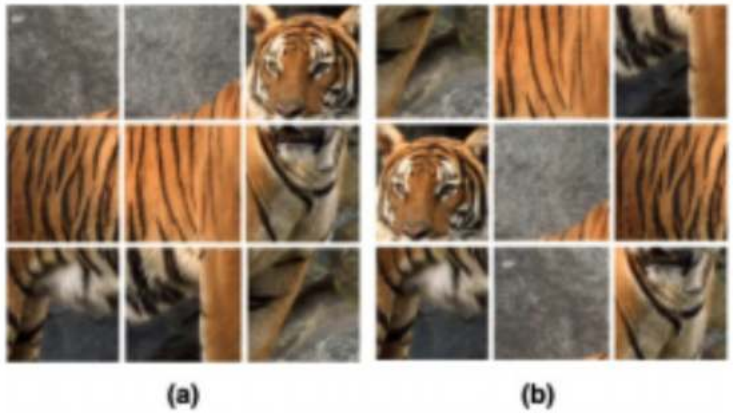












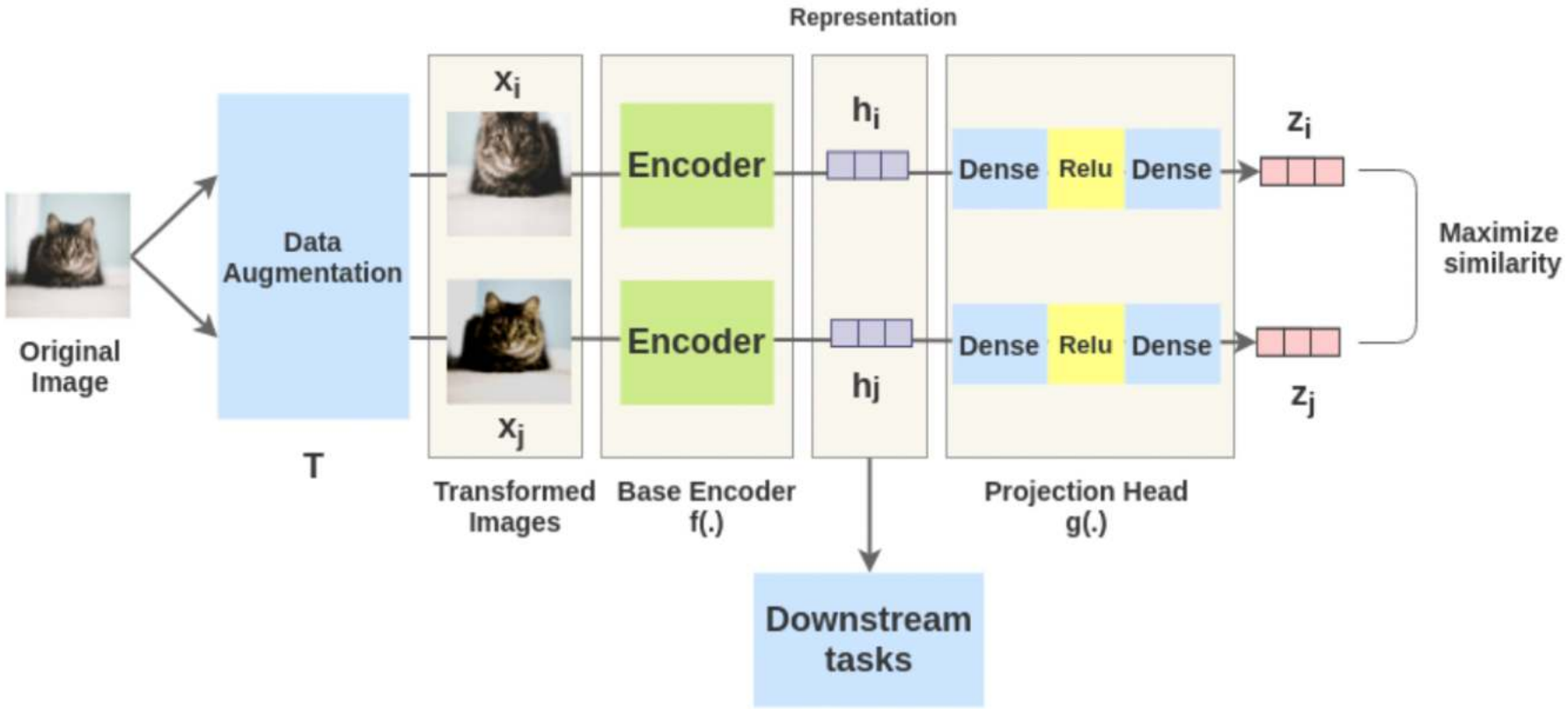
Color Augmentation

Image Rotation

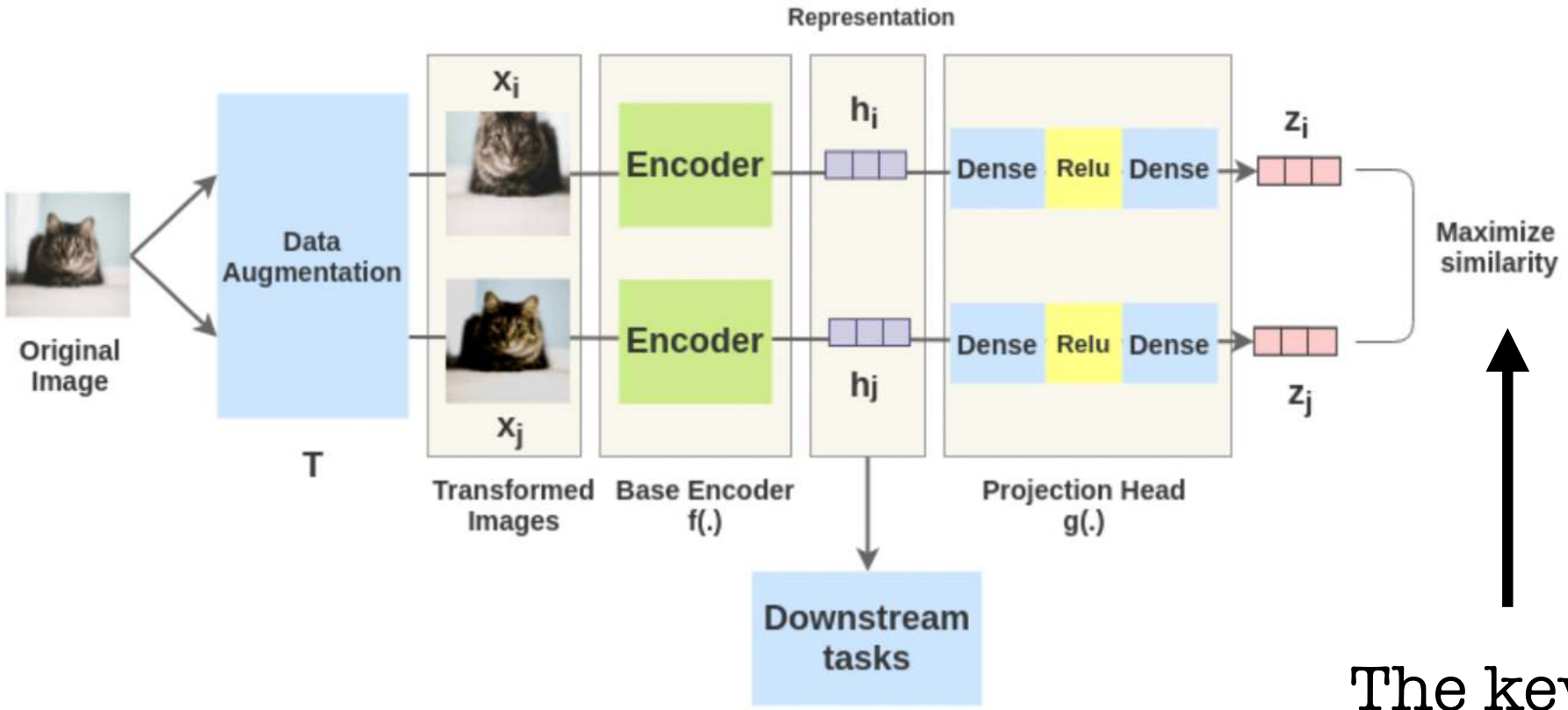
Image Cropping

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

The contrastive loss:

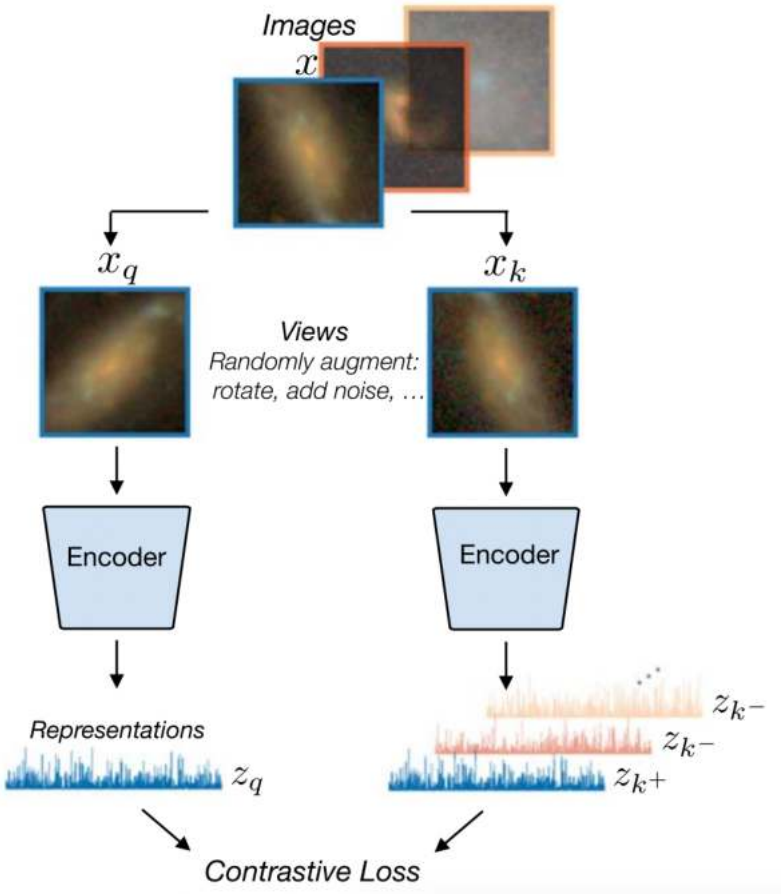
Similarly between
two representations of positive pairs

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

Sum of all similarities between
negative pairs

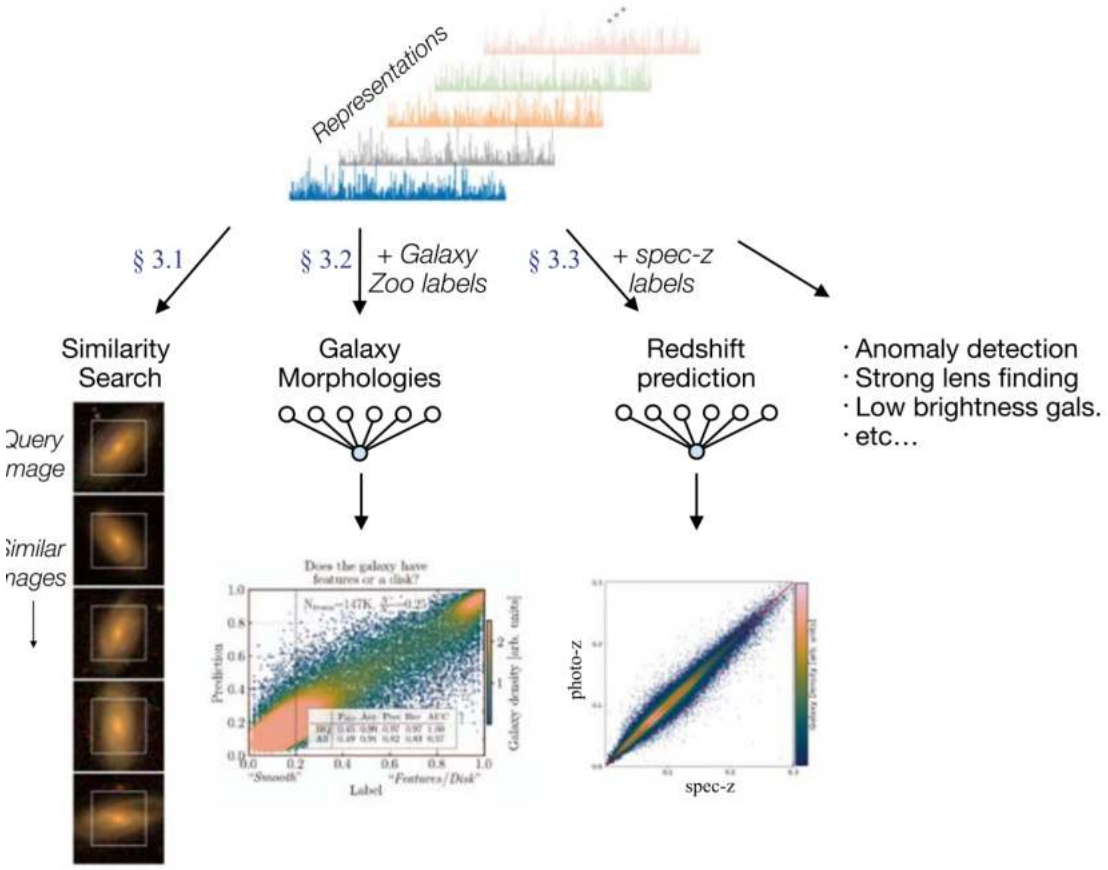
1. Self-supervised contrastive representation learning

Learn representations in an unsupervised manner



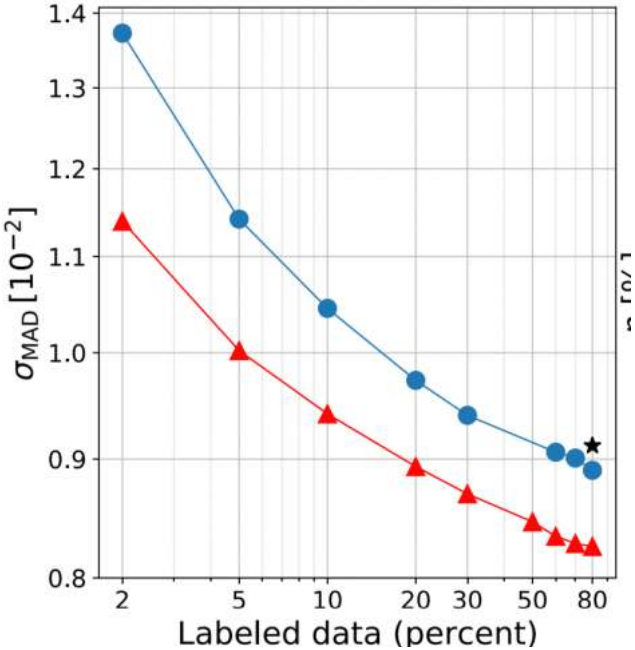
2. Downstream tasks

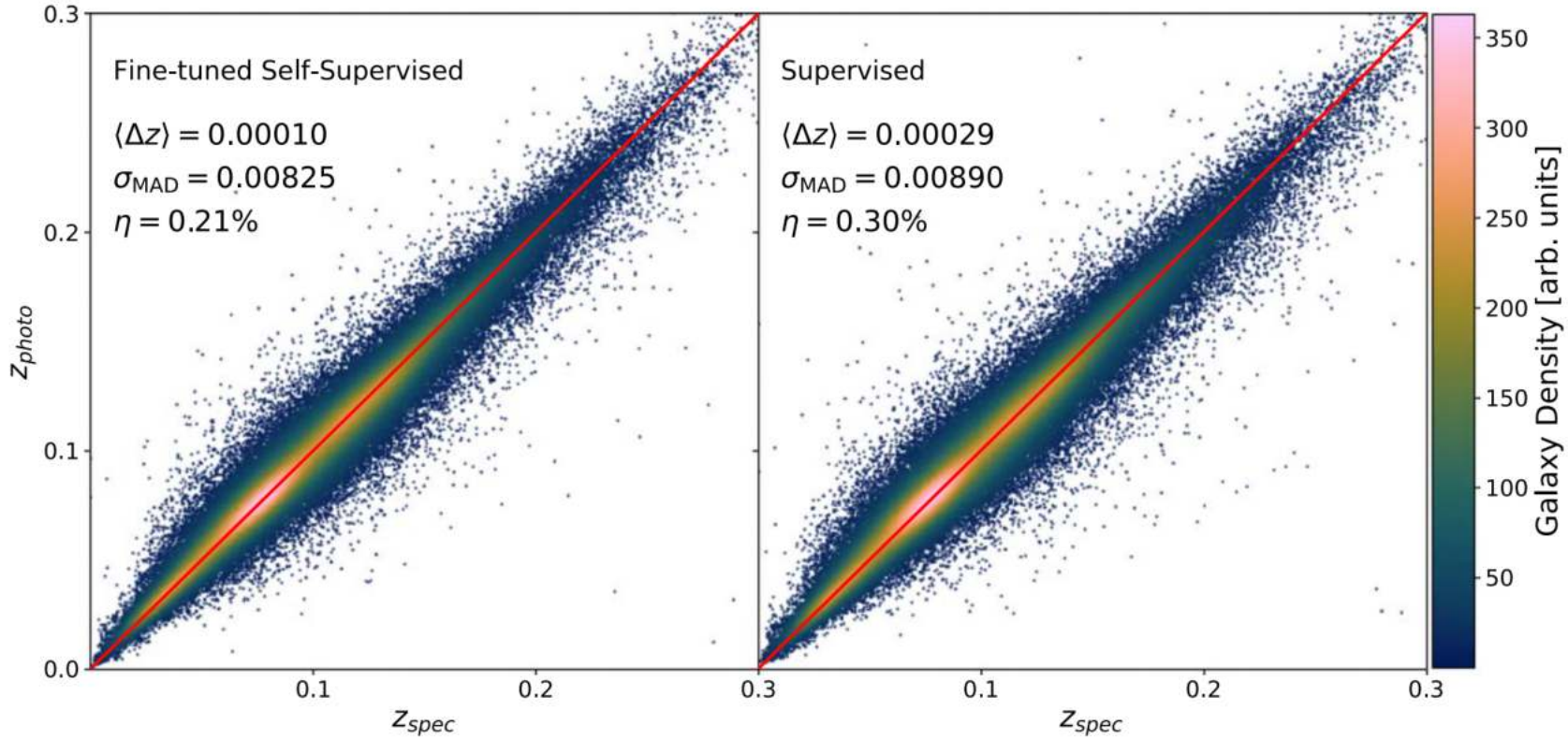
Use representations for a variety of applications



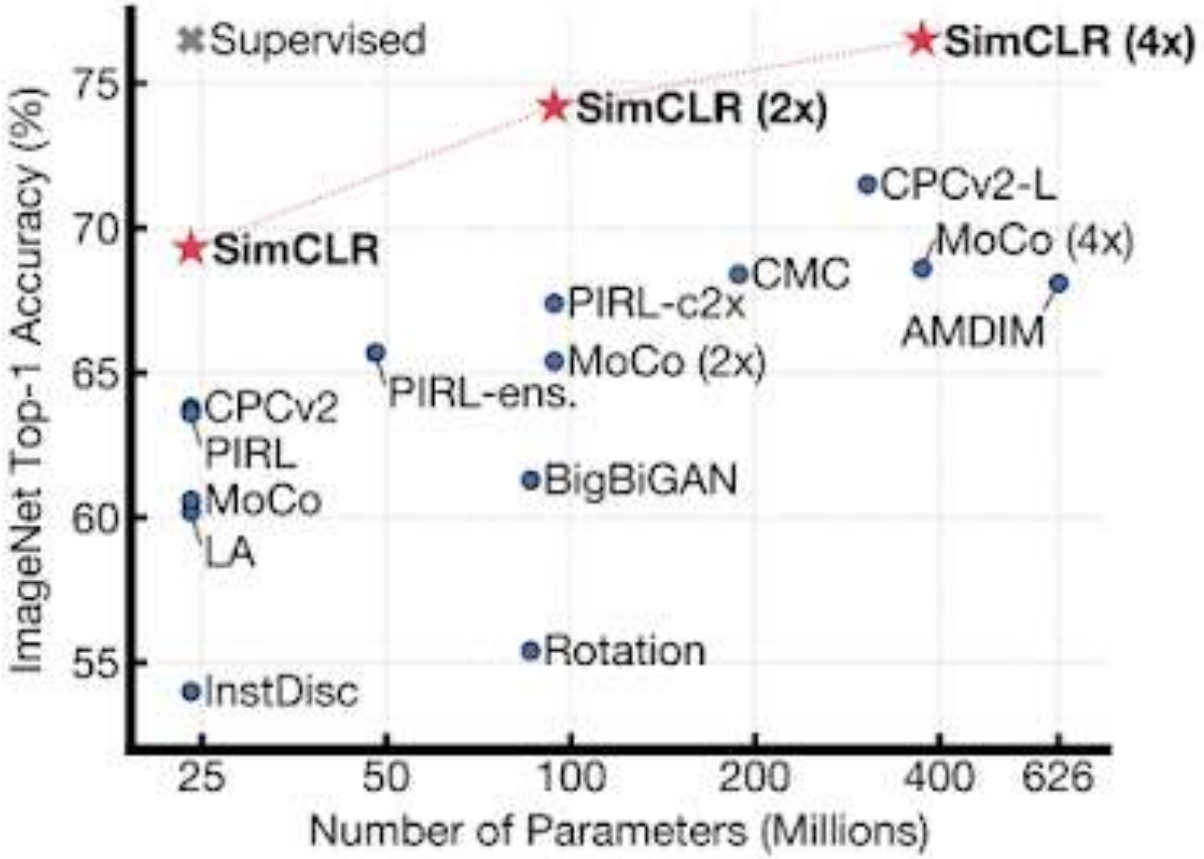
Hayat+21

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)

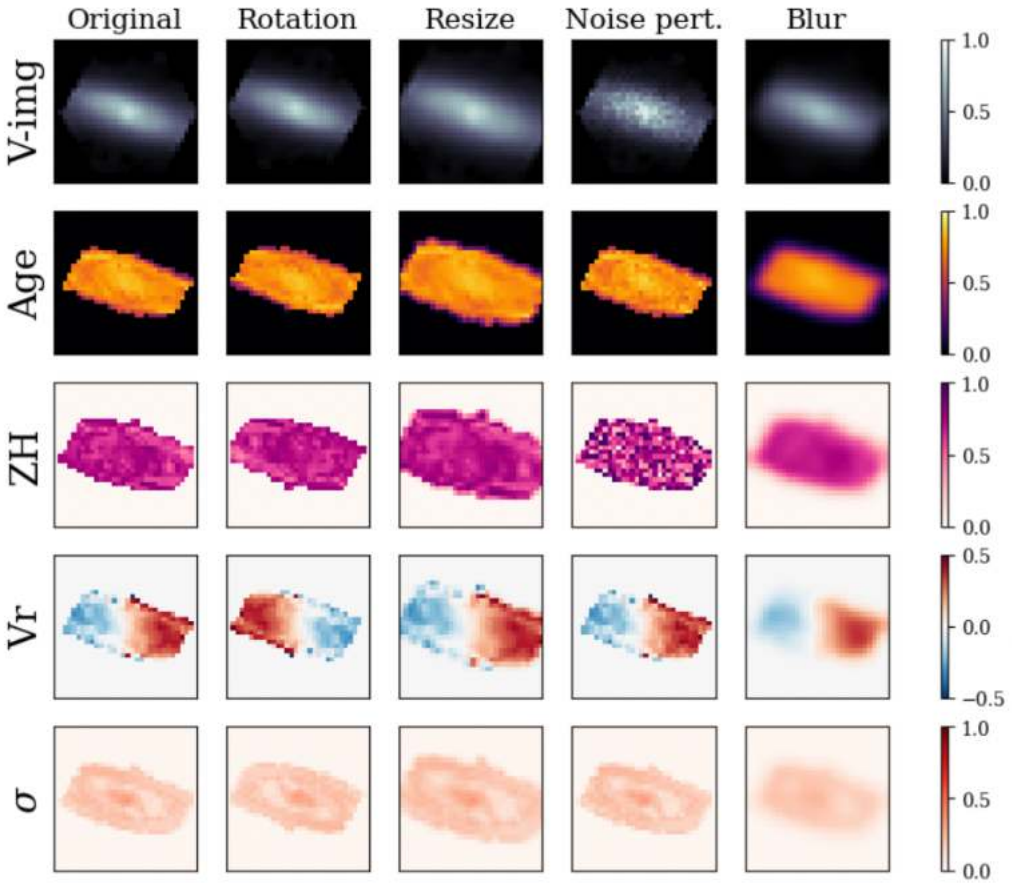

Fine-tuned Self-Supervised
$\langle \Delta z \rangle = 0.00010$
$\sigma_{\rm MAD} = 0.00825$
$\eta = 0.21\%$
Supervised
$\langle \Delta z \rangle = 0.00029$
$\sigma_{\rm MAD} = 0.00890$
$\eta = 0.30\%$
z_{photo}
z_{spec}
z_{spec}
Galaxy Density [arb. units]
0.0
0.1
0.2
0.3
50
100
150
200
250
300
350

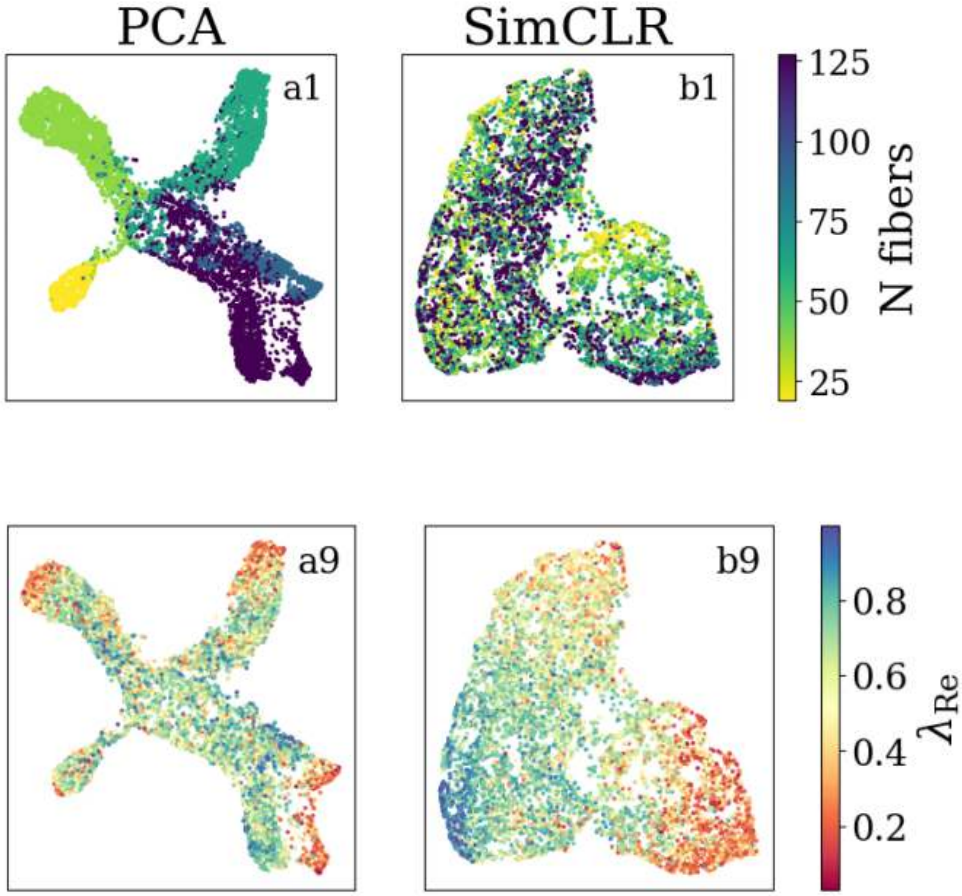
SELF-SUPERVISED LEARNING REACHES COMPARABLE ACCURACY TO FULLY SUPERVISED APPROACHES...

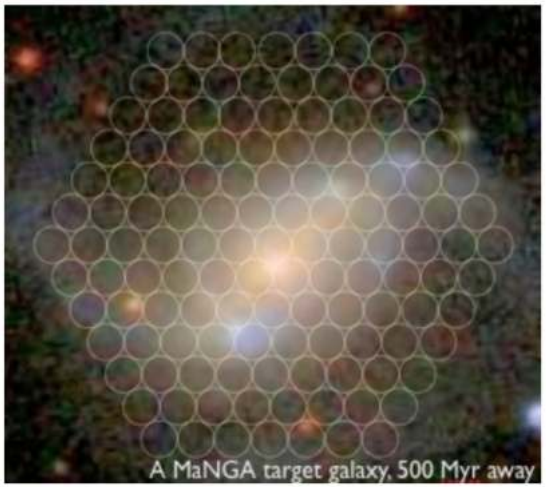


Chen+20

2. Sampling







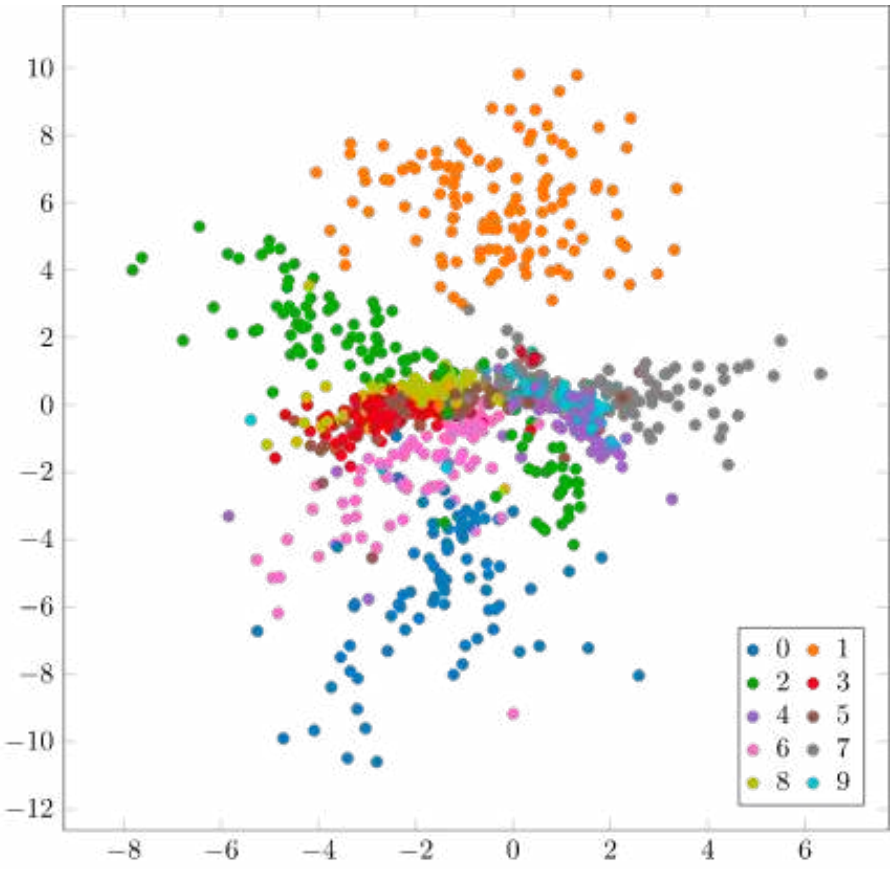
Contrastive learning representation of Manga galaxies

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**")

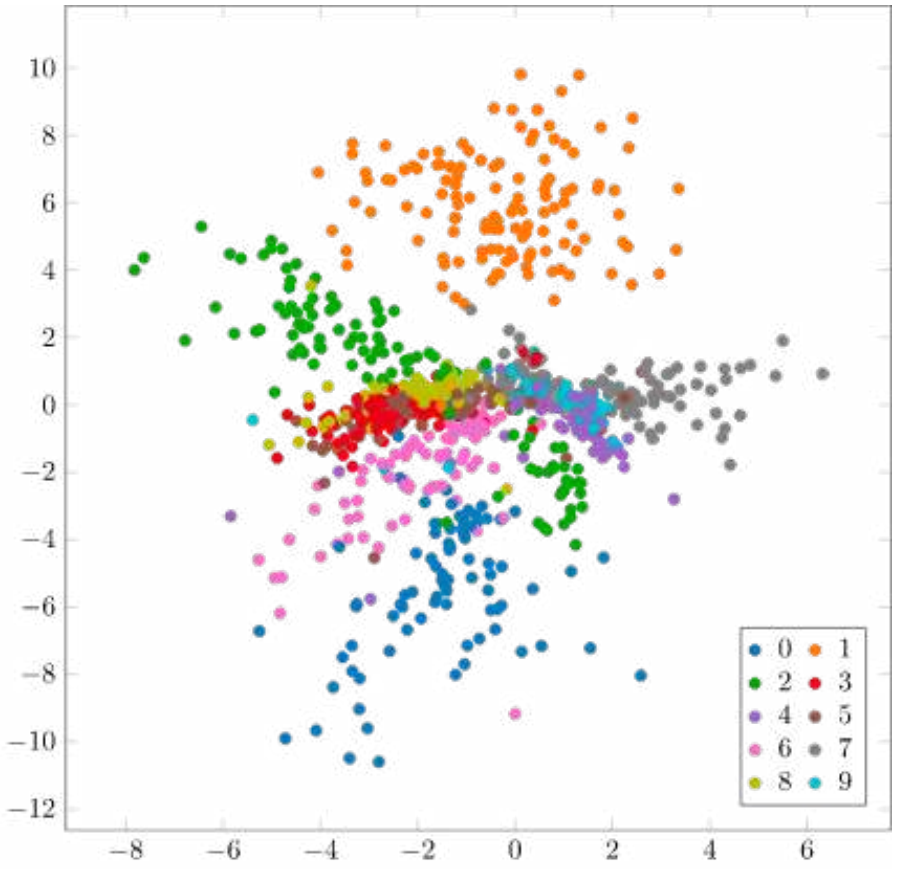
ESTIMATE THE PROBABILITY DENSITY OF AN ARBITRARY INPUT, RELATIVE TO THE INPUT DISTRIBUTION ("**DENSITY ESTIMATION**")



z space (latent space)

HOW CAN I ESTIMATE P(X)?

(P(z|x))

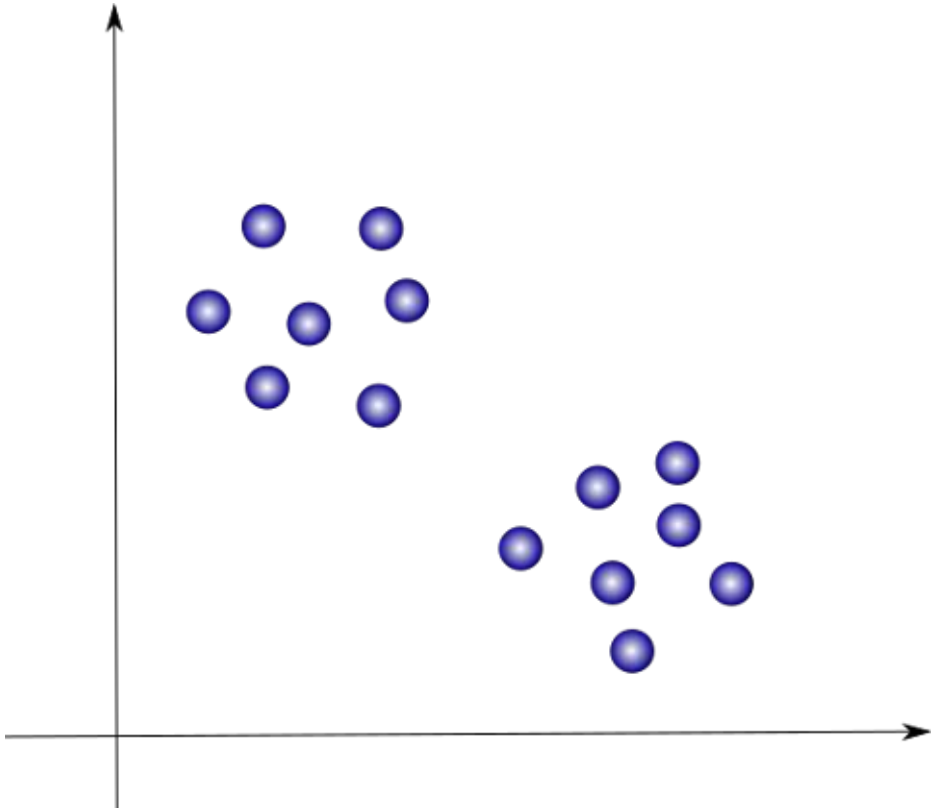


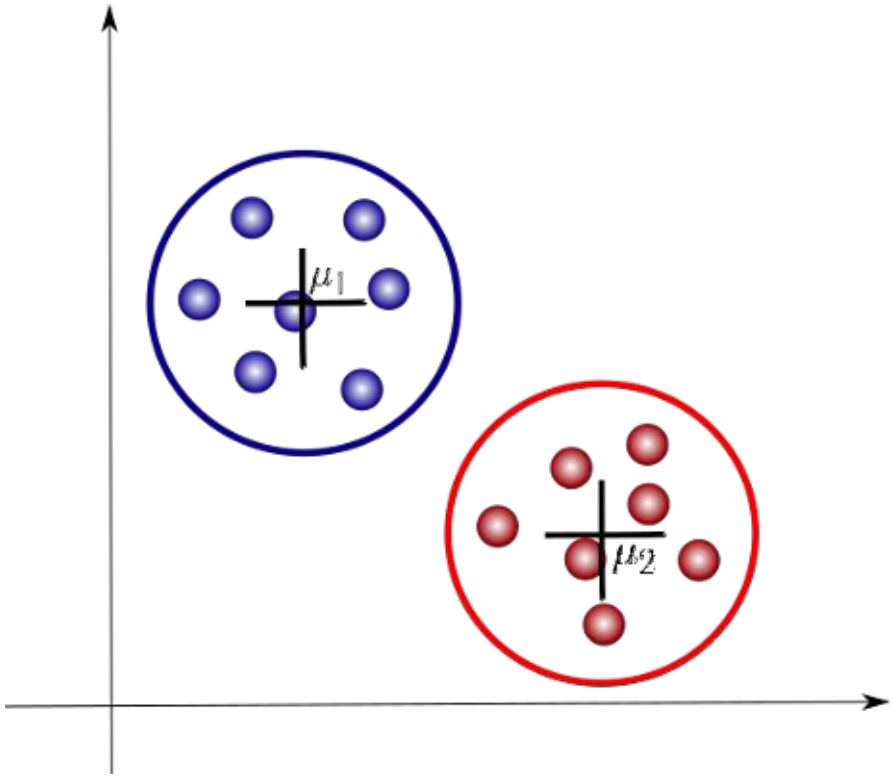
HOW CAN I ESTIMATE P(X)?

(P(z|x))

WHEN YOU DON'T KNOW, ASSUME IT IS GAUSSIAN….

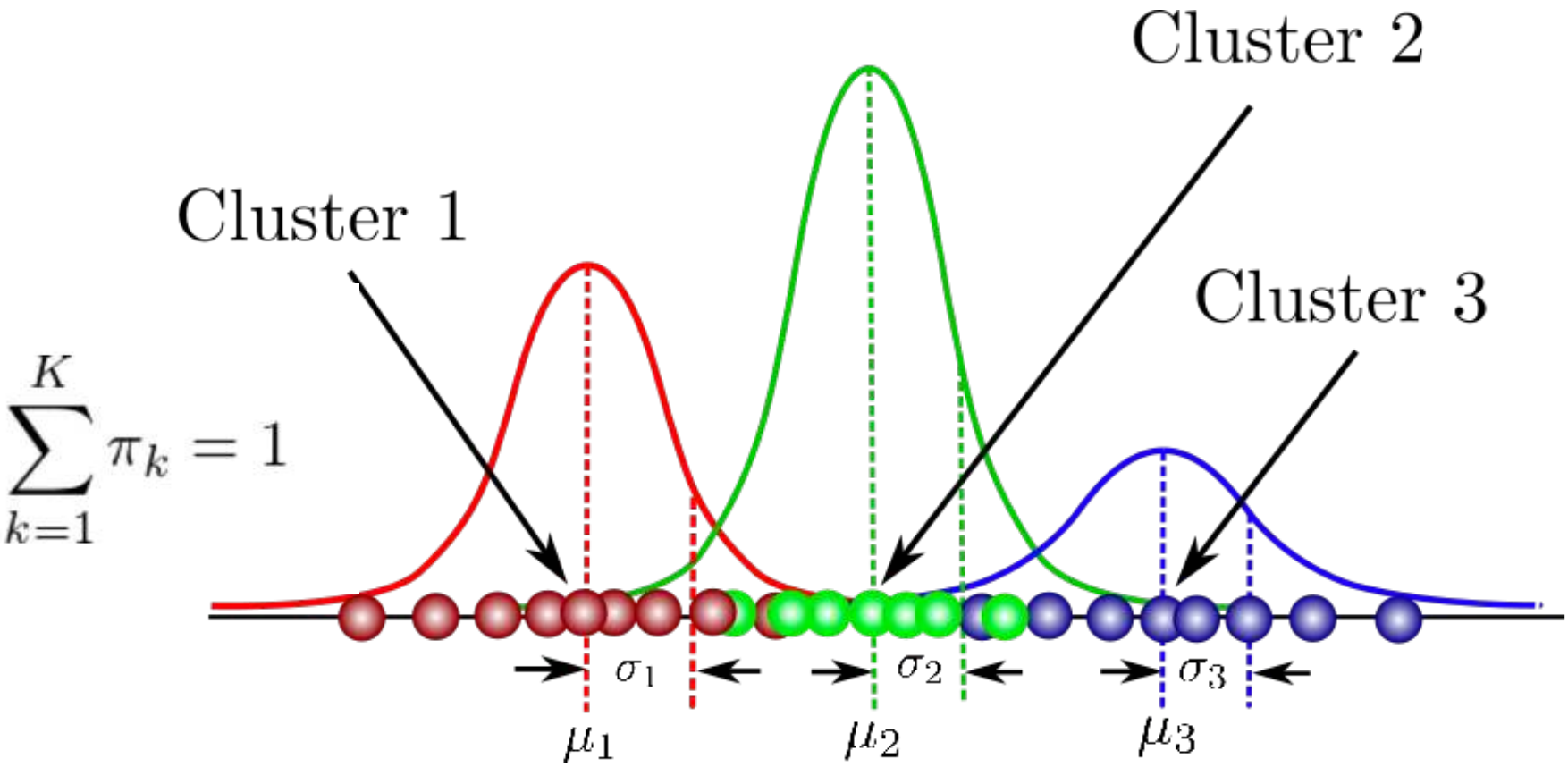
GAUSSIAN MIXTURE MODELS (GMMs) ARE DENSITY ESTIMATOR METHODS THAT FIT MULTIPLE GAUSSIANS TO THE REPRESENTATION





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

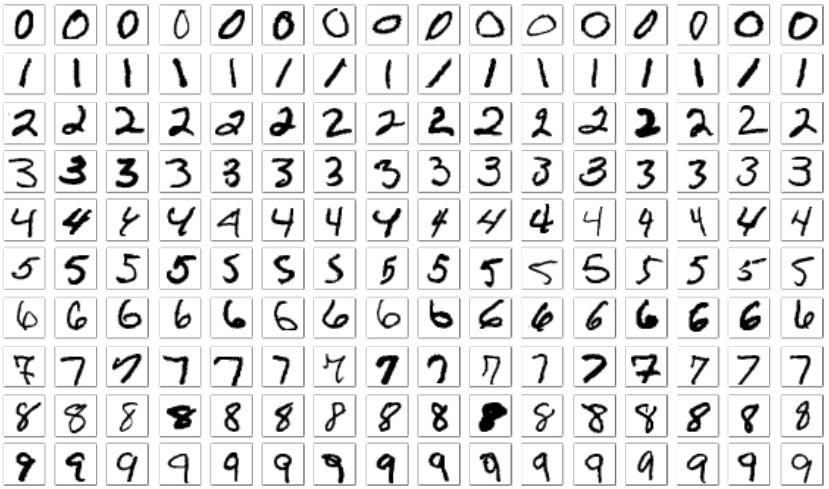
GAUSSIAN MIXTURE MODELS (GMMs) ARE DENSITY ESTIMATOR METHODS THAT FIT MULTIPLE GAUSSIANS TO THE REPRESENTATION

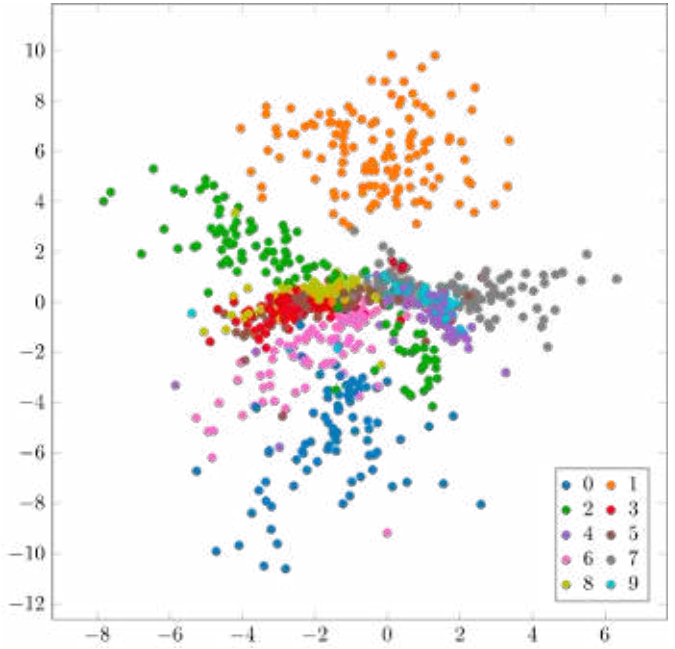


means, sigmas and scale factors of each gaussian are free parameters

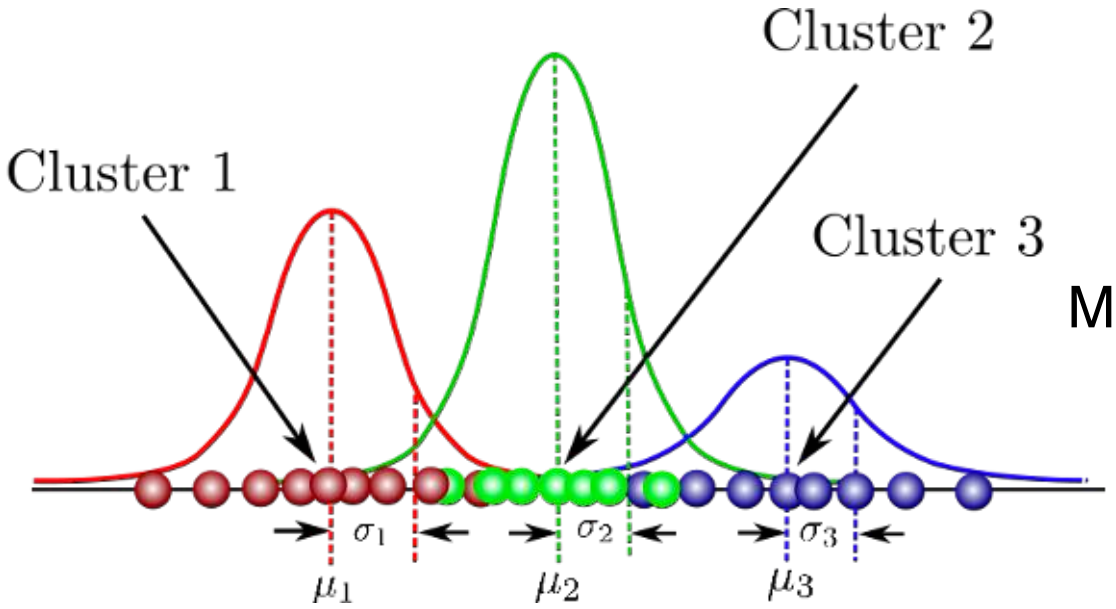
https://towardsdatascience.com/gaussian-mixture-models-explained-6986aaf5a95

DATA





REPRESENTATION
(AUTOENCODER - z

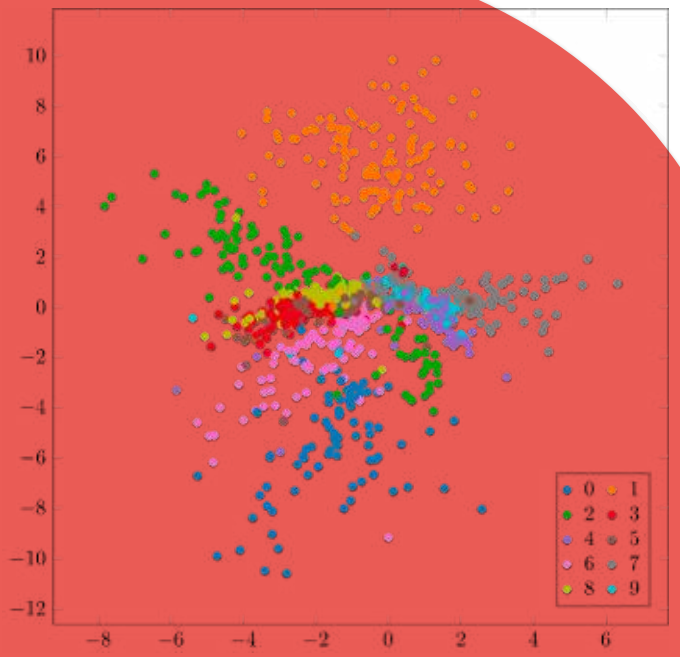


MODELING OF P(z|x) WITH GMMs

GMMs

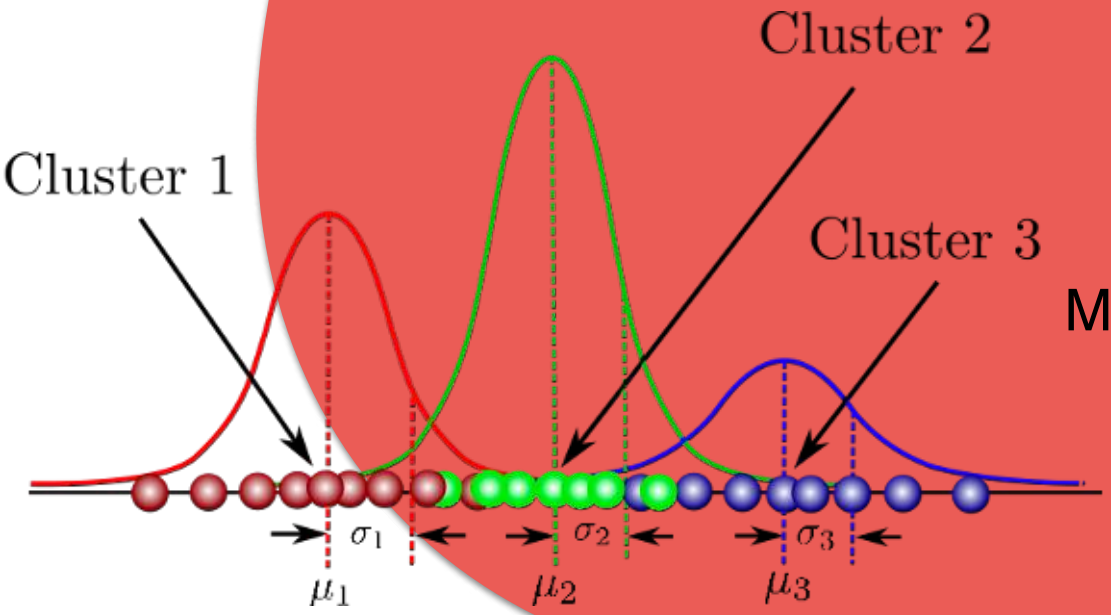
- Both sampling and evaluating are straightforward with GMMs
- However, performance scales poorly with increasing dimensionality
- Depends on initial choices...

DATA



REPRESENTATION
(AUTOENCODER)

CAN WE COMBINE THESE 2 STEPS?



MODELING OF P(z|x) WITH GMMs

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 FLOWS, AUTOREGRESSIVE FLOWS)

*** 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)

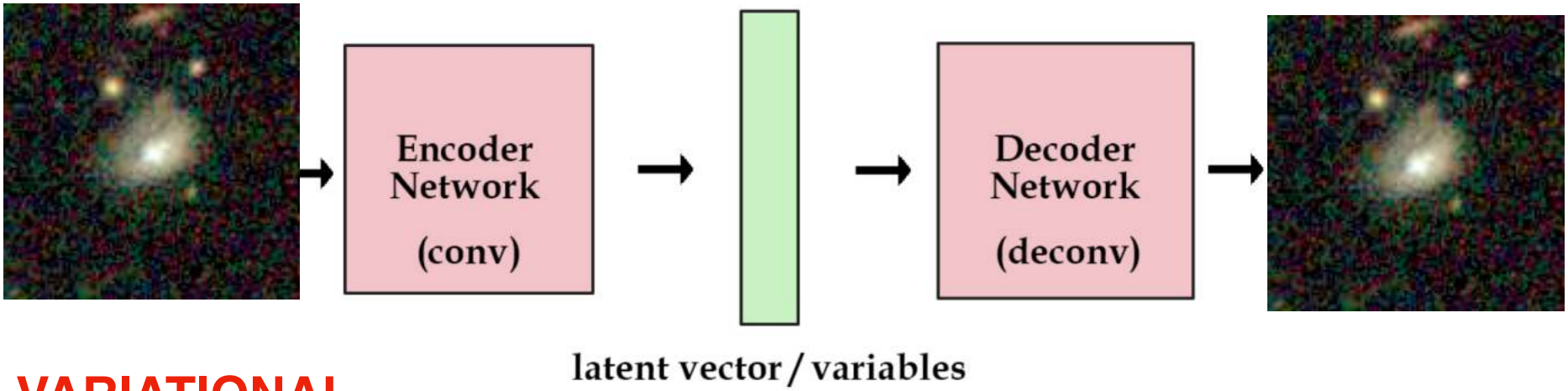
GAN

(GENERATIVE ADVERSARIAL NETOWRK)

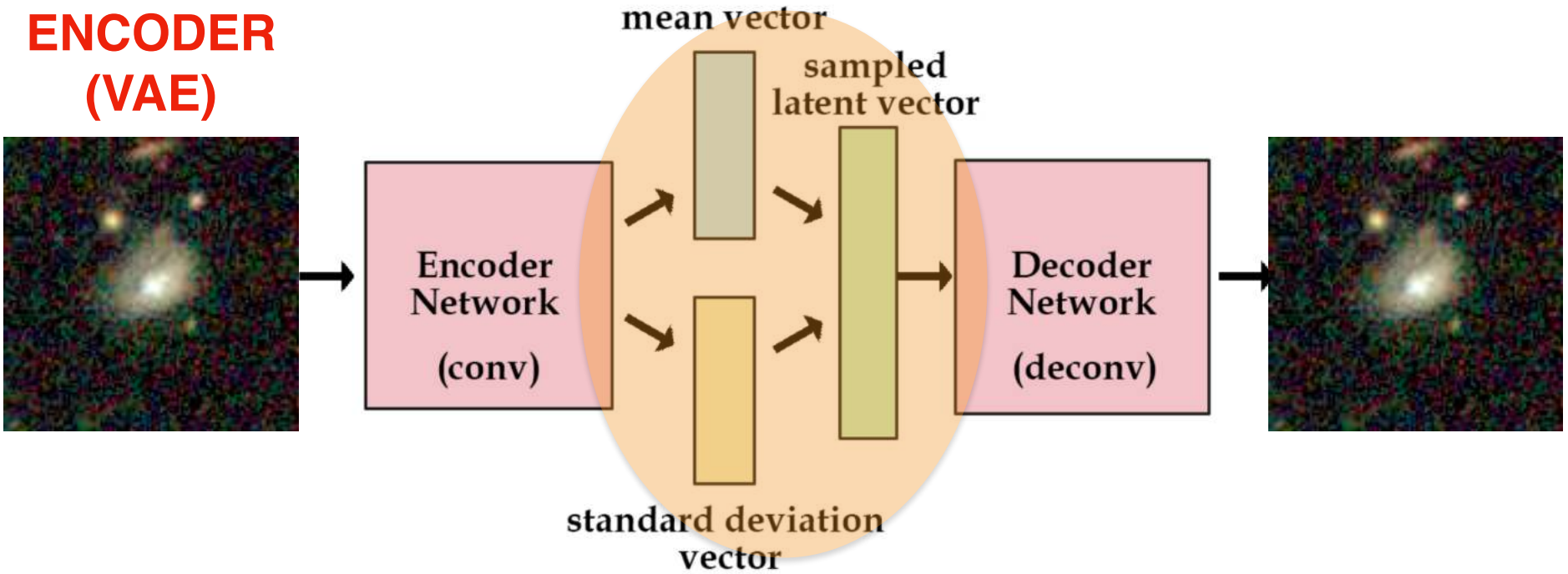
NF-ARF

(NORMALIZING FLOWS, AUTOREGRESSIVE FLOWS)

AUTO-ENCODER

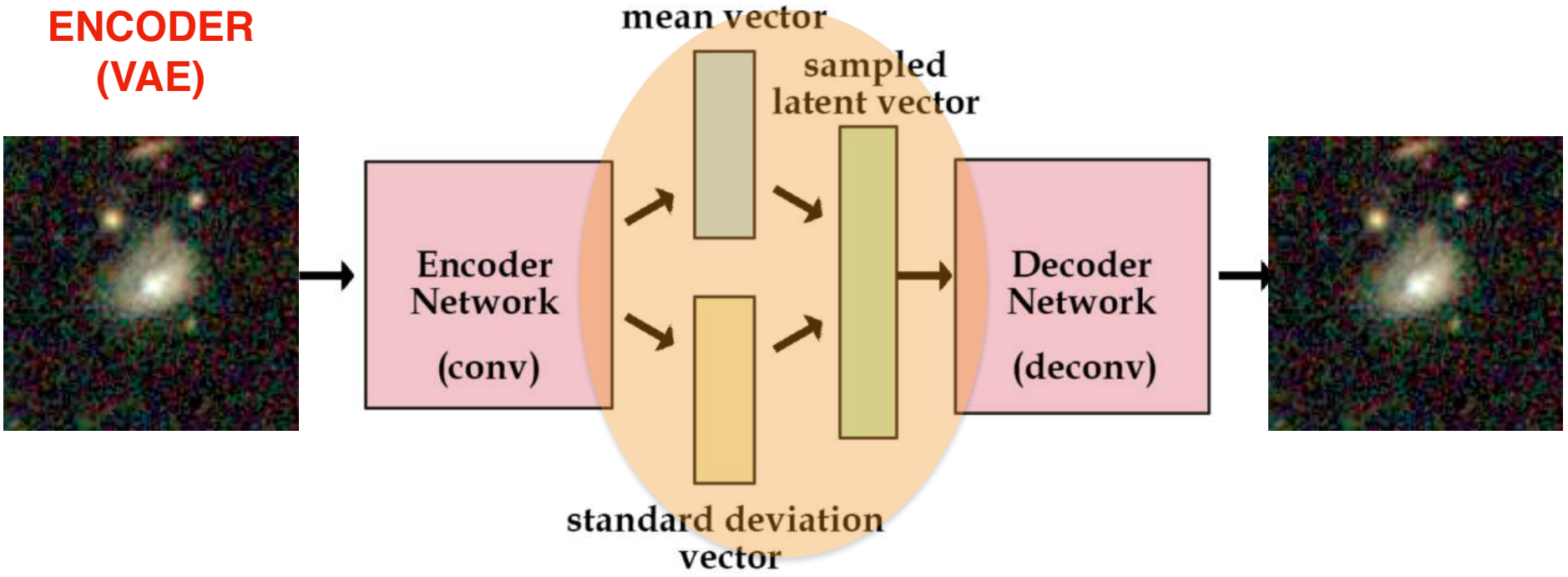


VARIATIONAL AUTO-ENCODER (VAE)



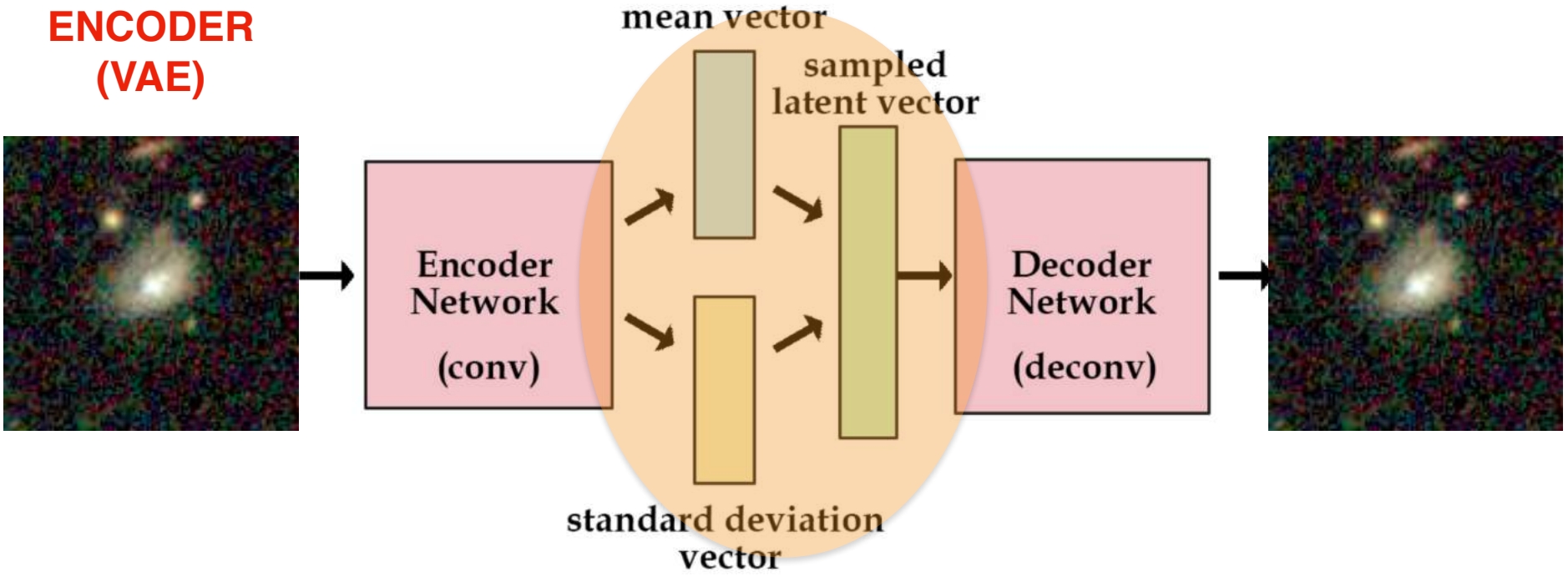
LET'S MODEL THE LATENT SPACE WITH A MIXUTRE OF GAUSSIANS

VARIATIONAL AUTO-ENCODER (VAE)



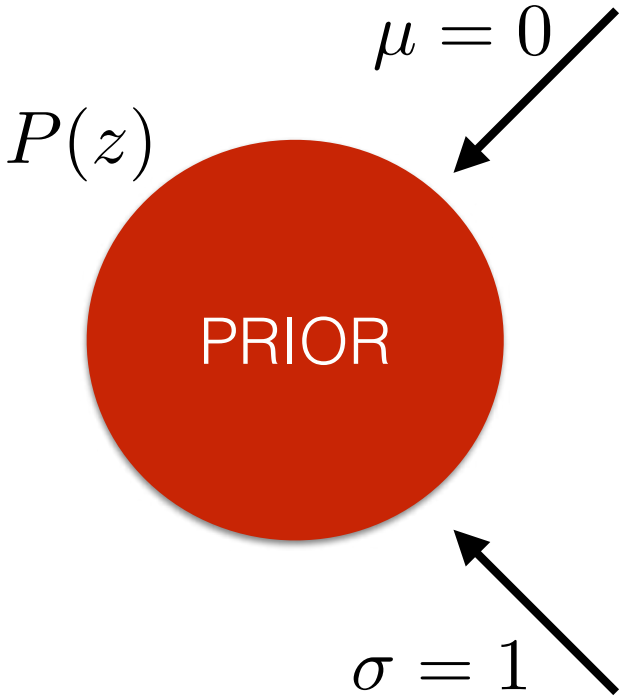
HOWEVER, NOTHING GUARANTEES US THAT P(z|x) CAN BE ACCURATELY MODELLED BY A MIXTURE OF GAUSSIANS….

VARIATIONAL AUTO-ENCODER (VAE)

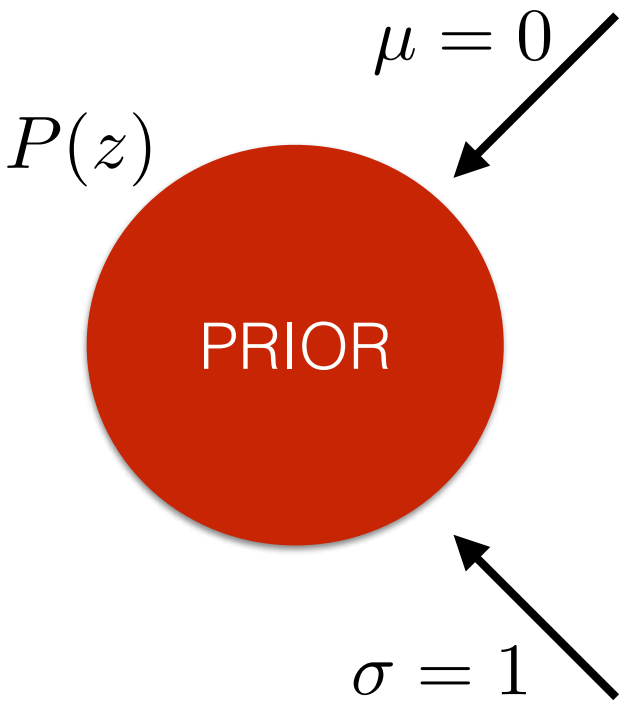


HOWEVER, NOTHING GUARANTEES US THAT THE LATENT SPACE CAN BE MODELLED BY A MIXTURE OF GAUSSIANS….

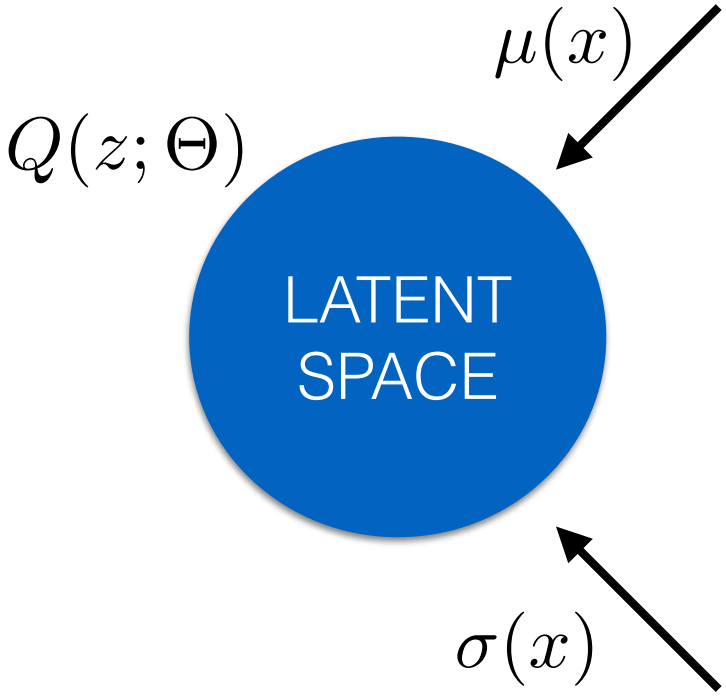
… LET'S FORCE IT TO BE GAUSSIAN LIKE!



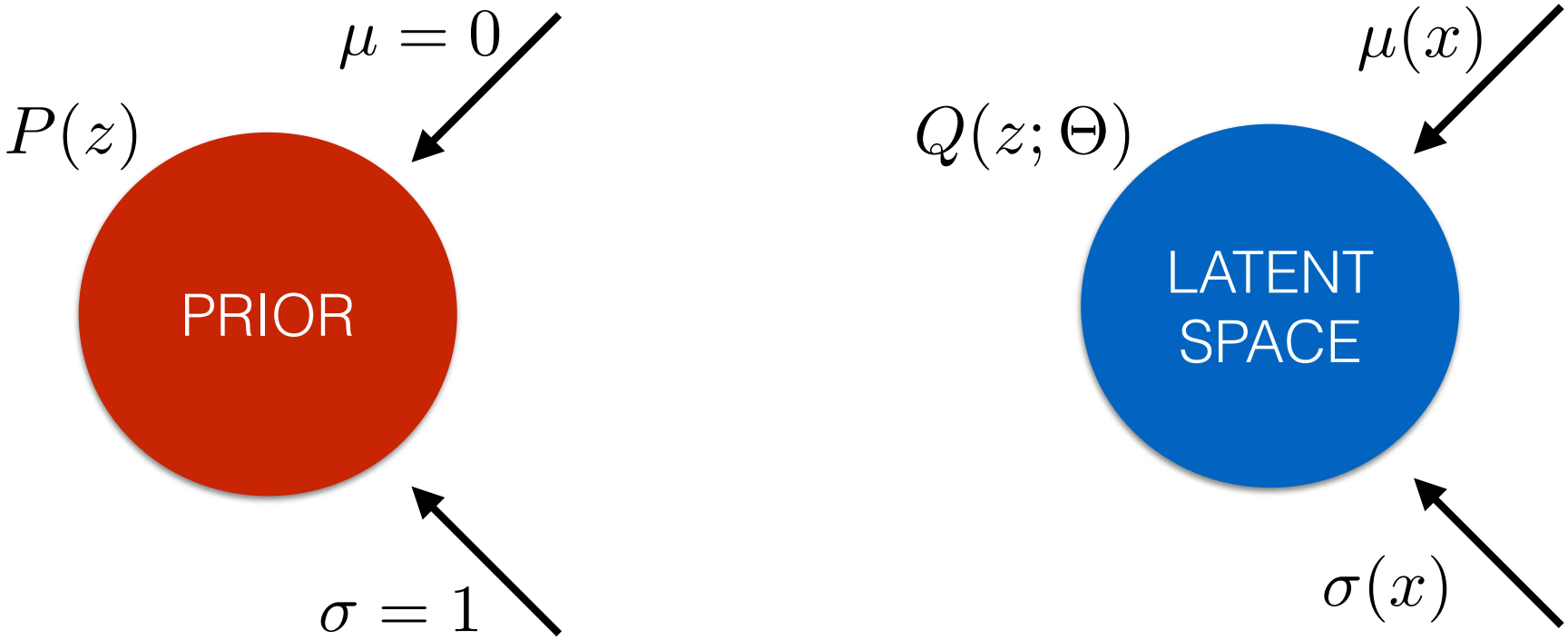
WE ASSUME A SIMPLE PRIOR



WE ASSUME A SIMPLE PRIOR

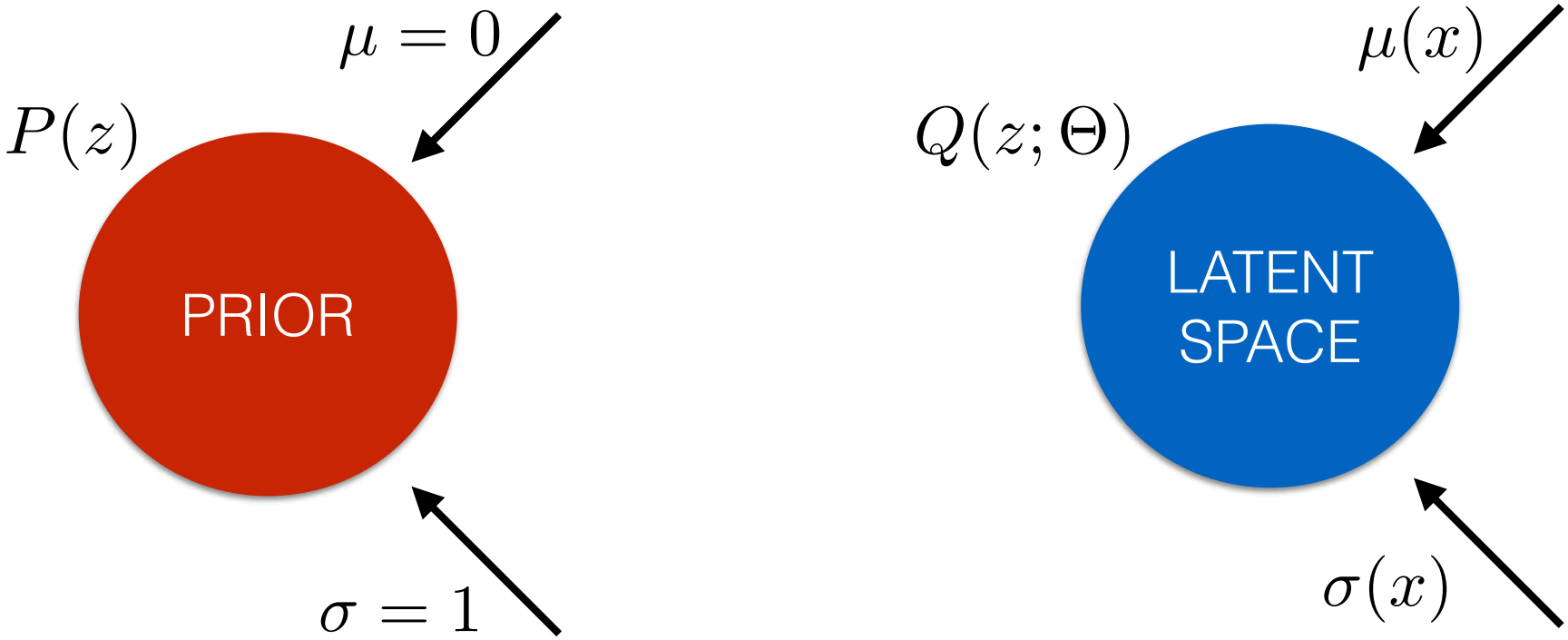


LATENT SPACE MODELING



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...

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

L2 LOSS

~Gaussian

REGULARIZATION TERM

WHAT WOULD BE THEN THE LOSS FUNCTION OF A VAE?

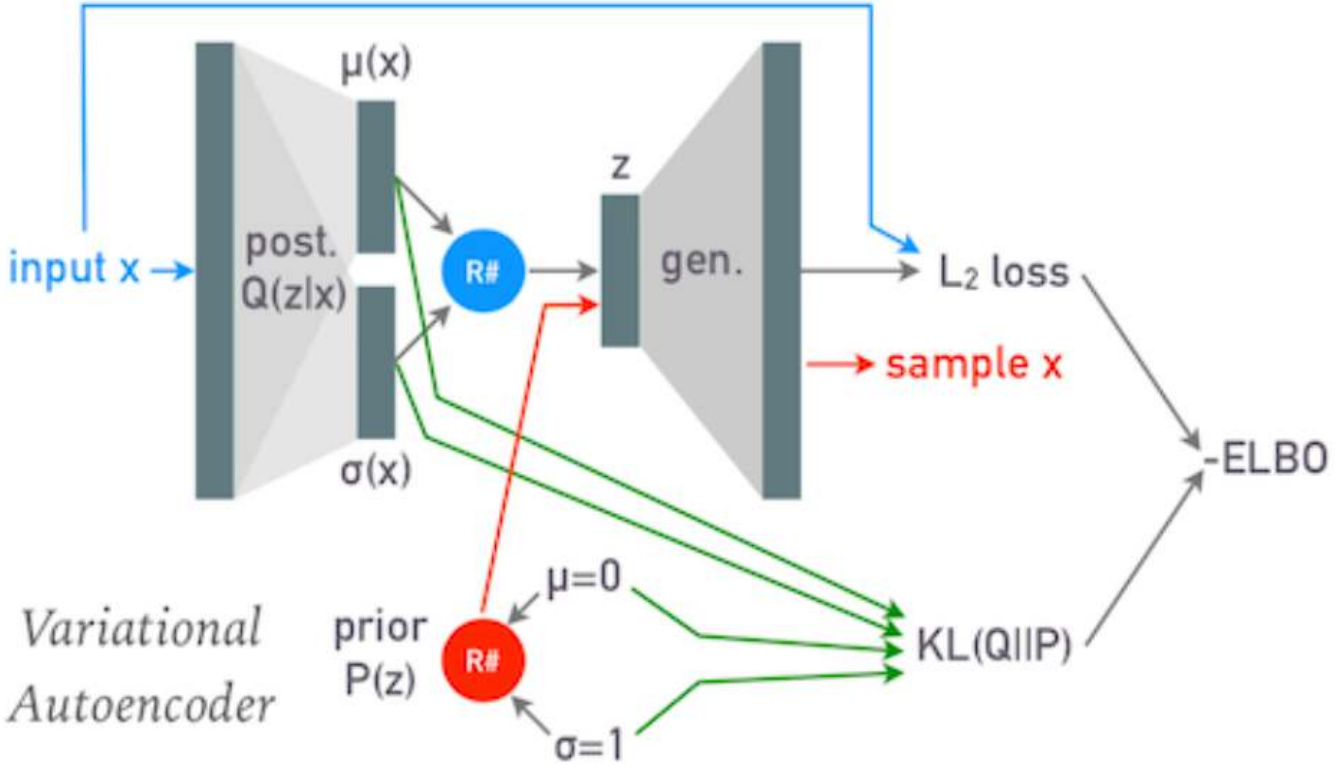
The key insight of VAE is that we are actually performing **variational inference** here, which then tells us what the loss function should be...

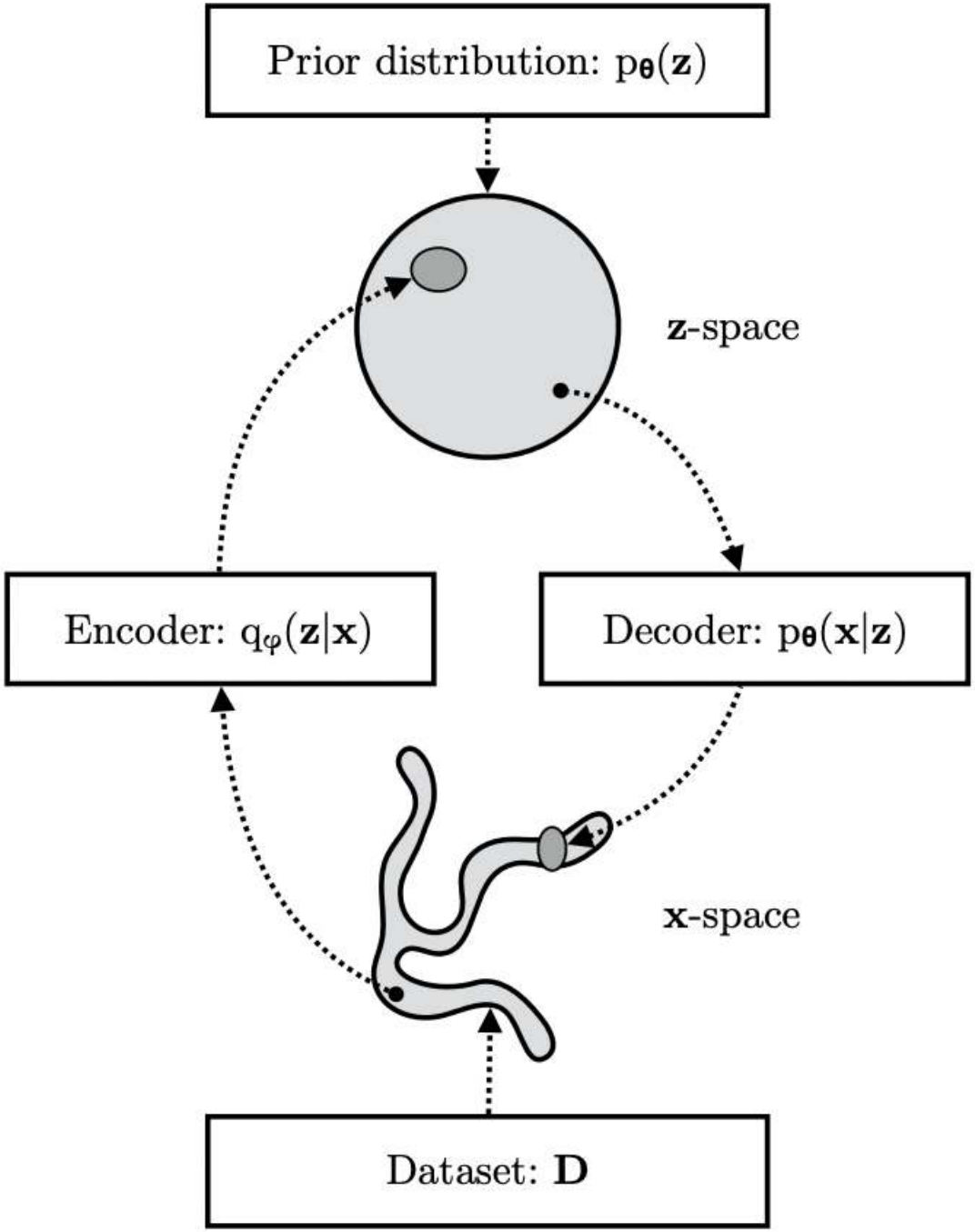
$$-\text{ELBO} = \langle \log P(\mathbf{x} \mid \mathbf{z}) \rangle_{z \sim Q} + \text{KL}\left(Q(\mathbf{z}; \Theta) \parallel P(\mathbf{z})\right) \ ,$$

L2 LOSS

REGULARIZATION TERM

~Gaussian

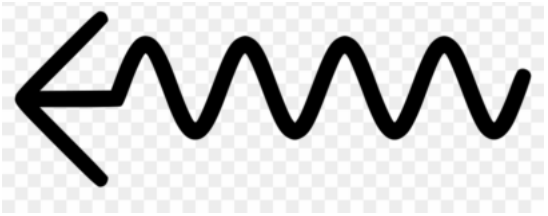


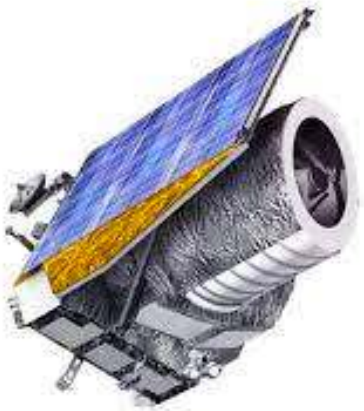


[Kingma and Welling, 2019]

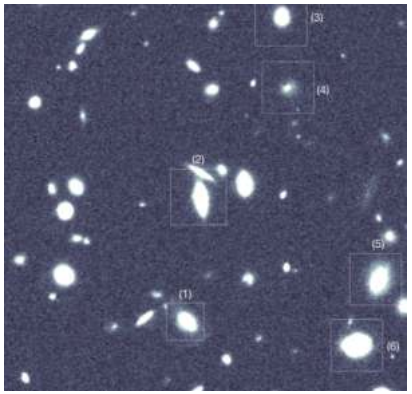
Preparing a space mission requires very detailed simulations

From photons

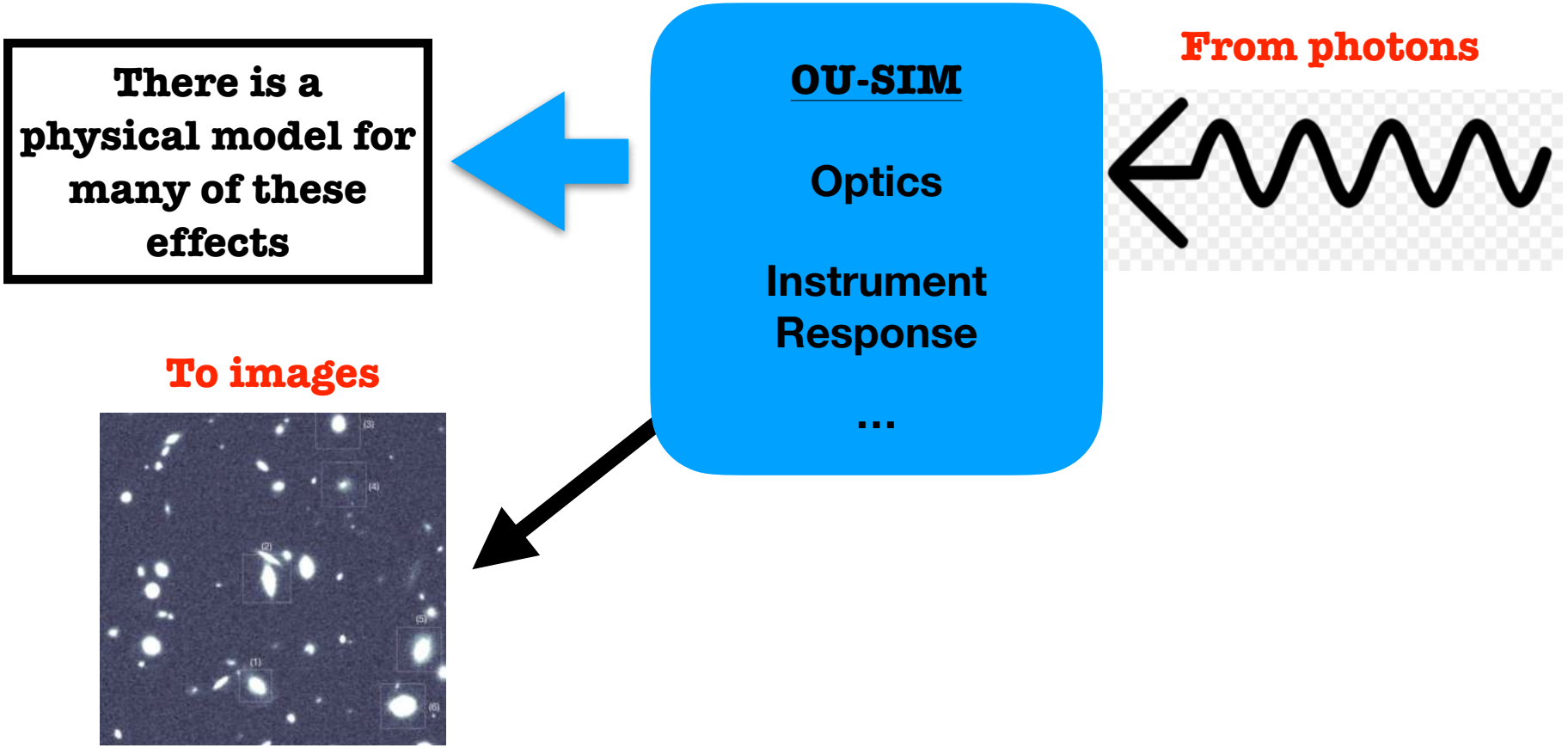




To images



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

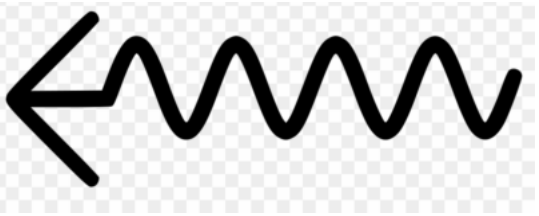
OU-SIM

Optics

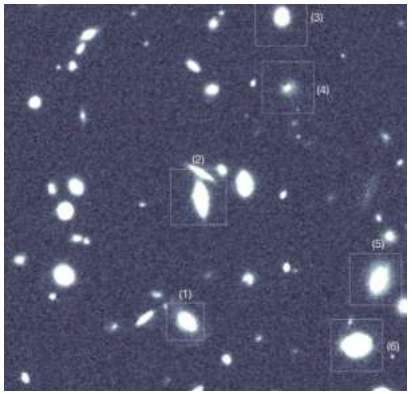
Instrument Response

...

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

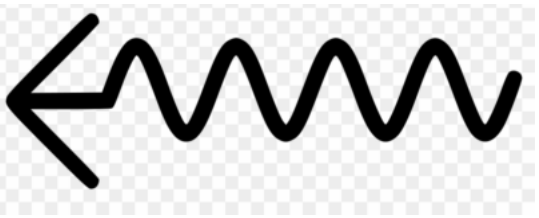
OU-SIM

Optics

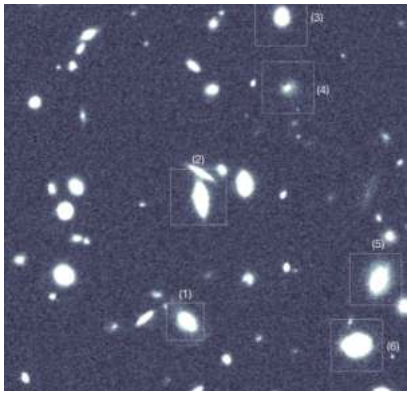
Instrument Response

...

From photons



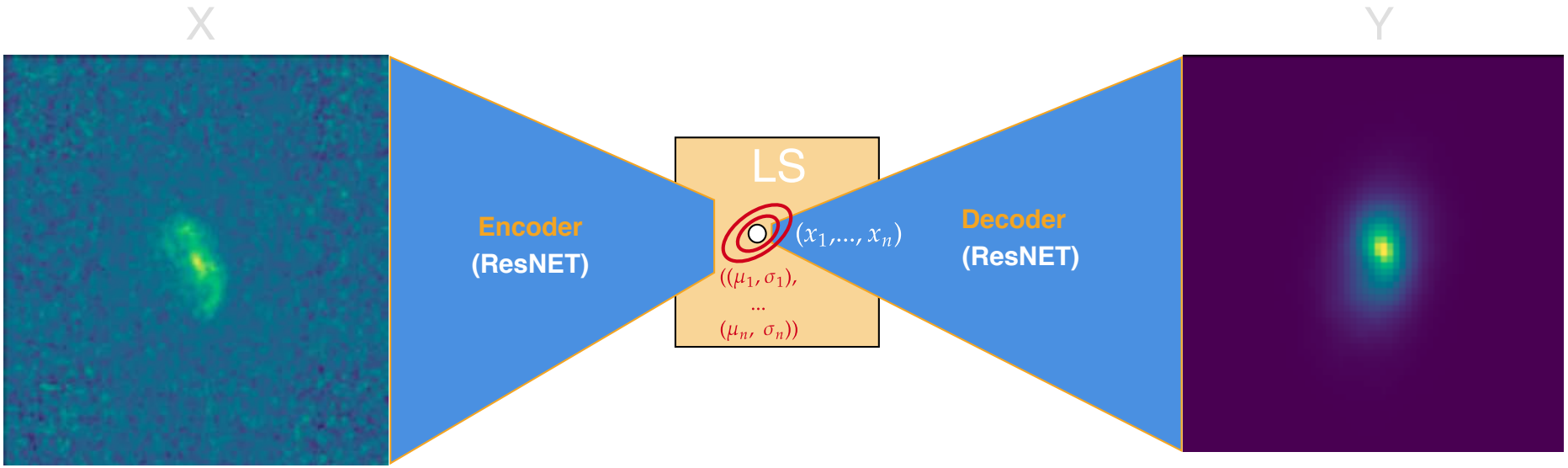
To images



Neural Network

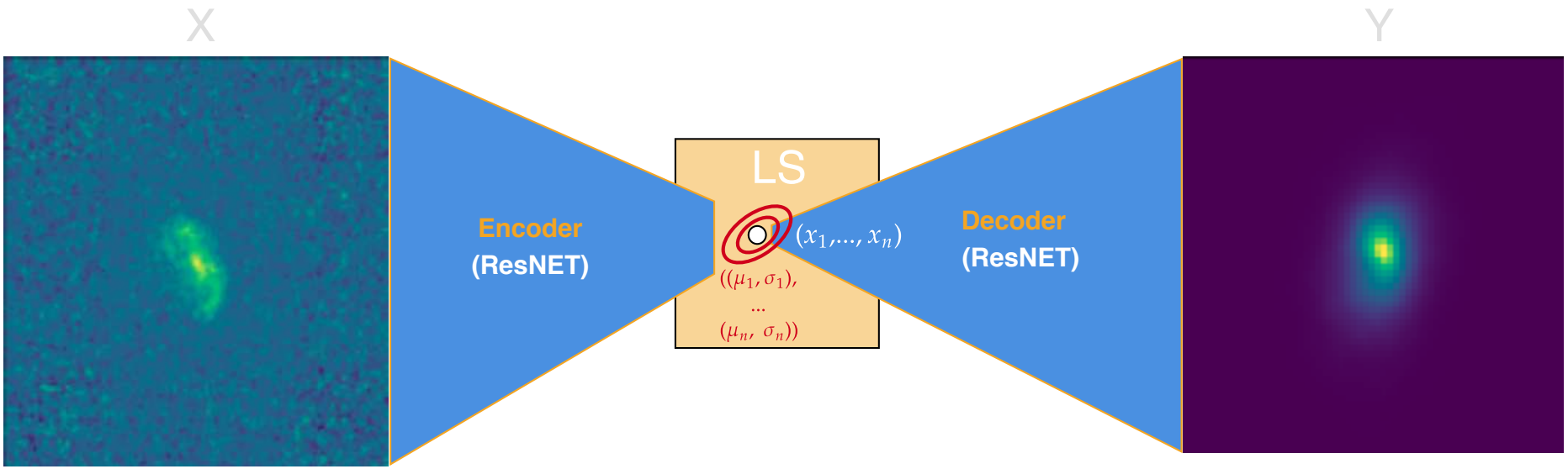
Others are more difficult to model

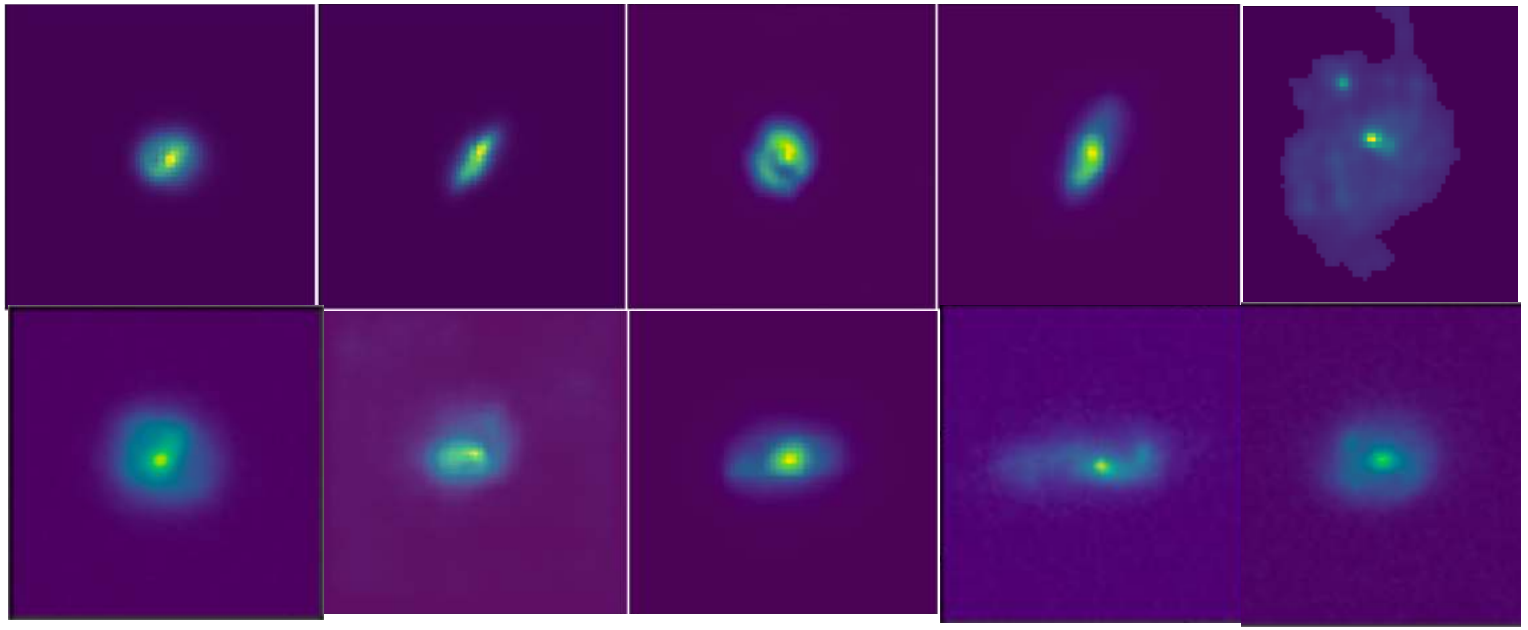
Include a Generative Model for galaxy generation in the Euclid Simulation Pipeline to model process for which we do not have a physical model



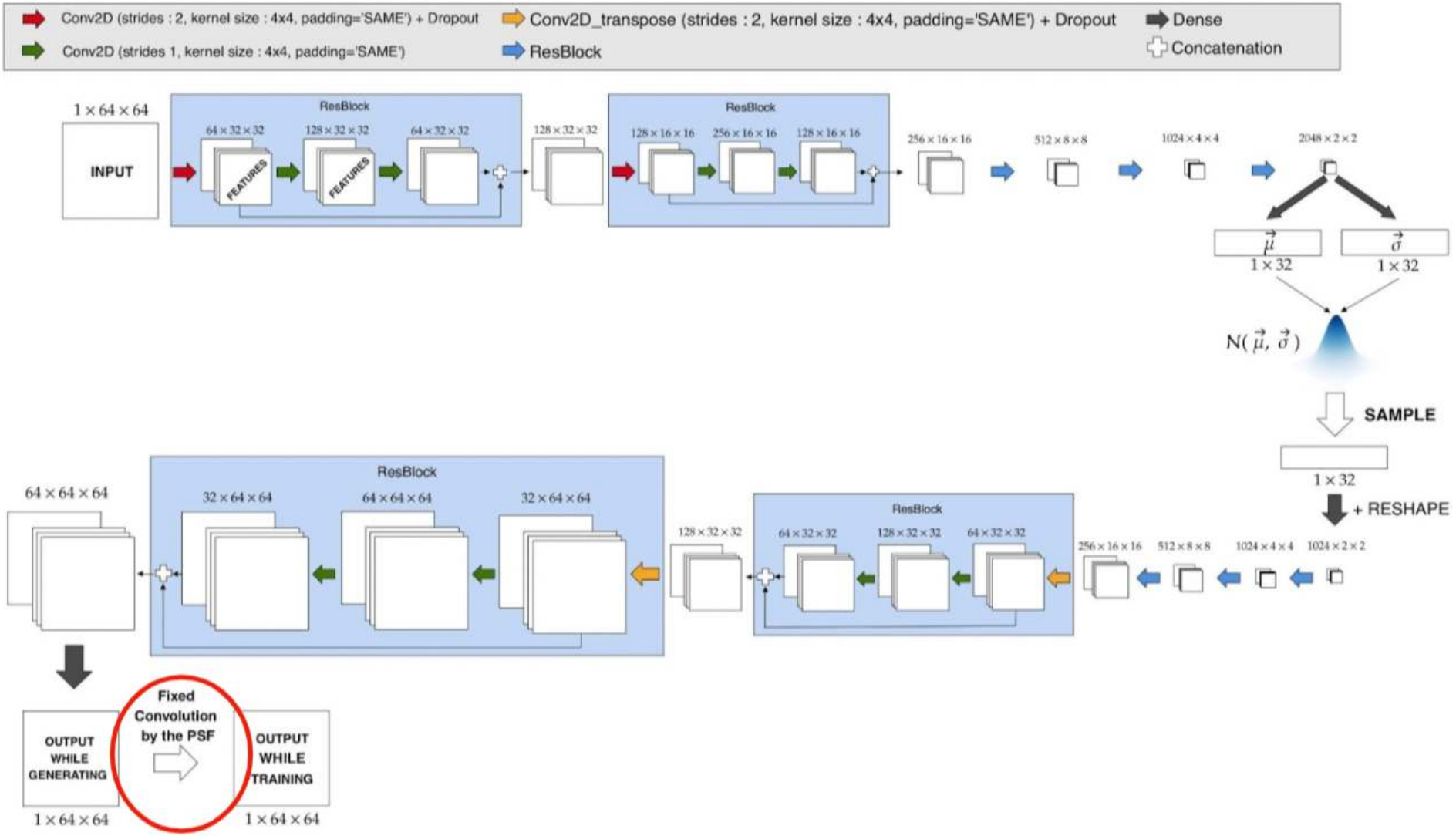
Bretonnière+21

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



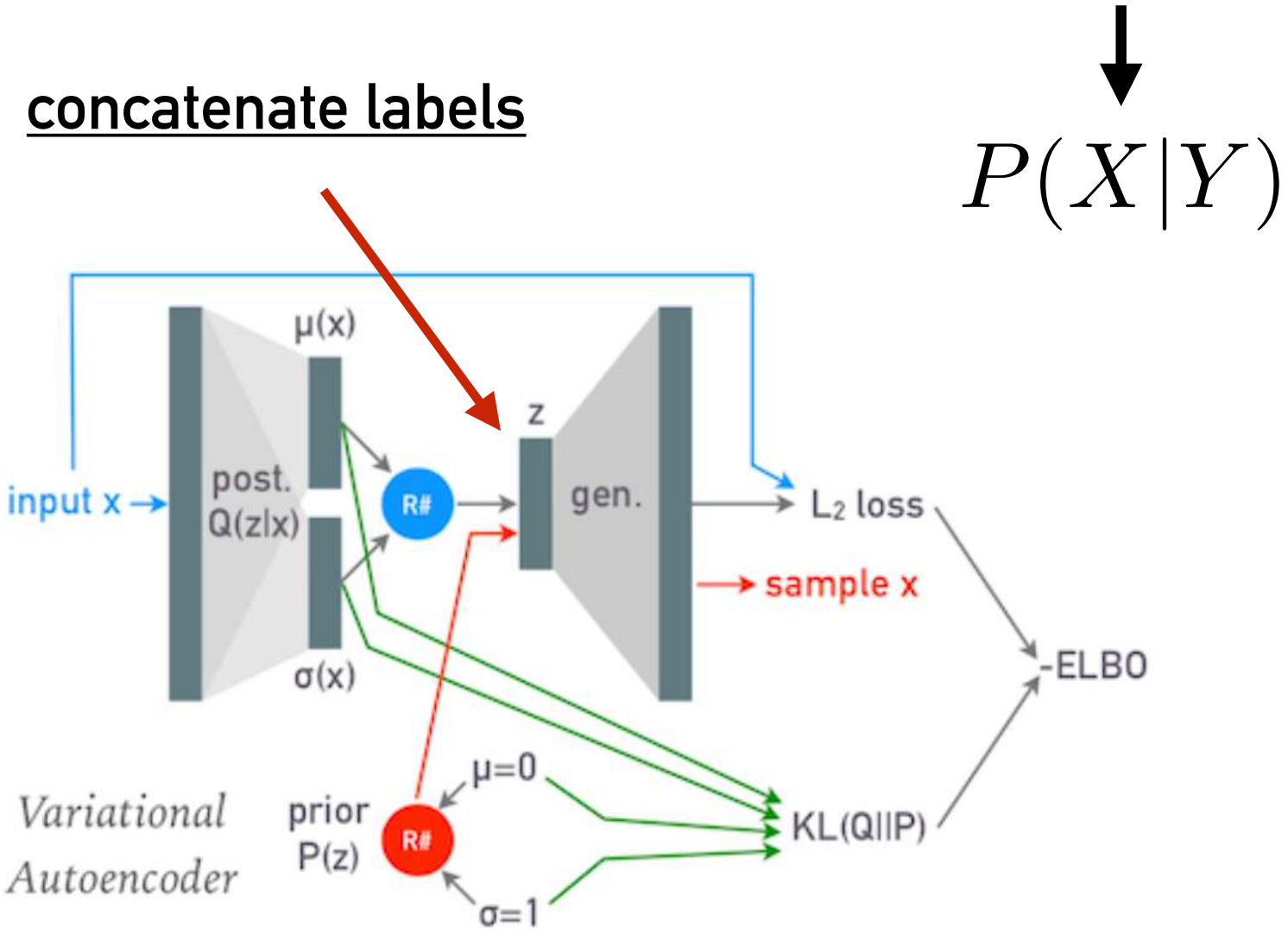


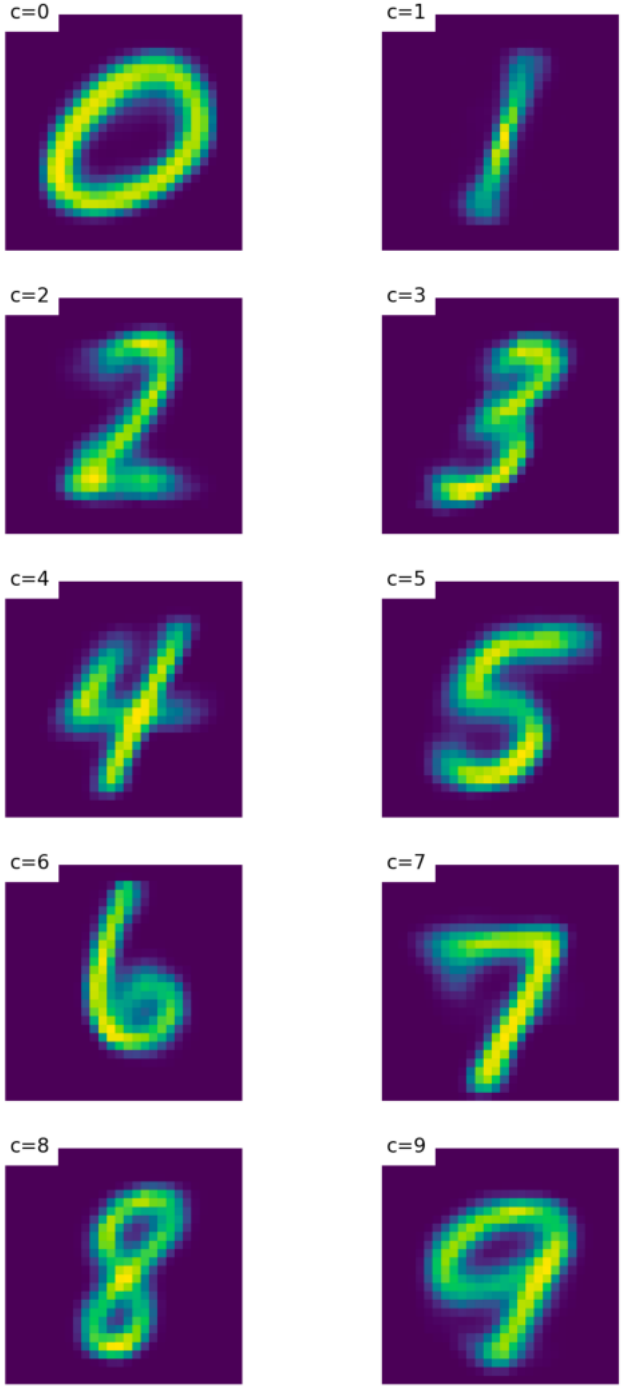
Bretonnière+21



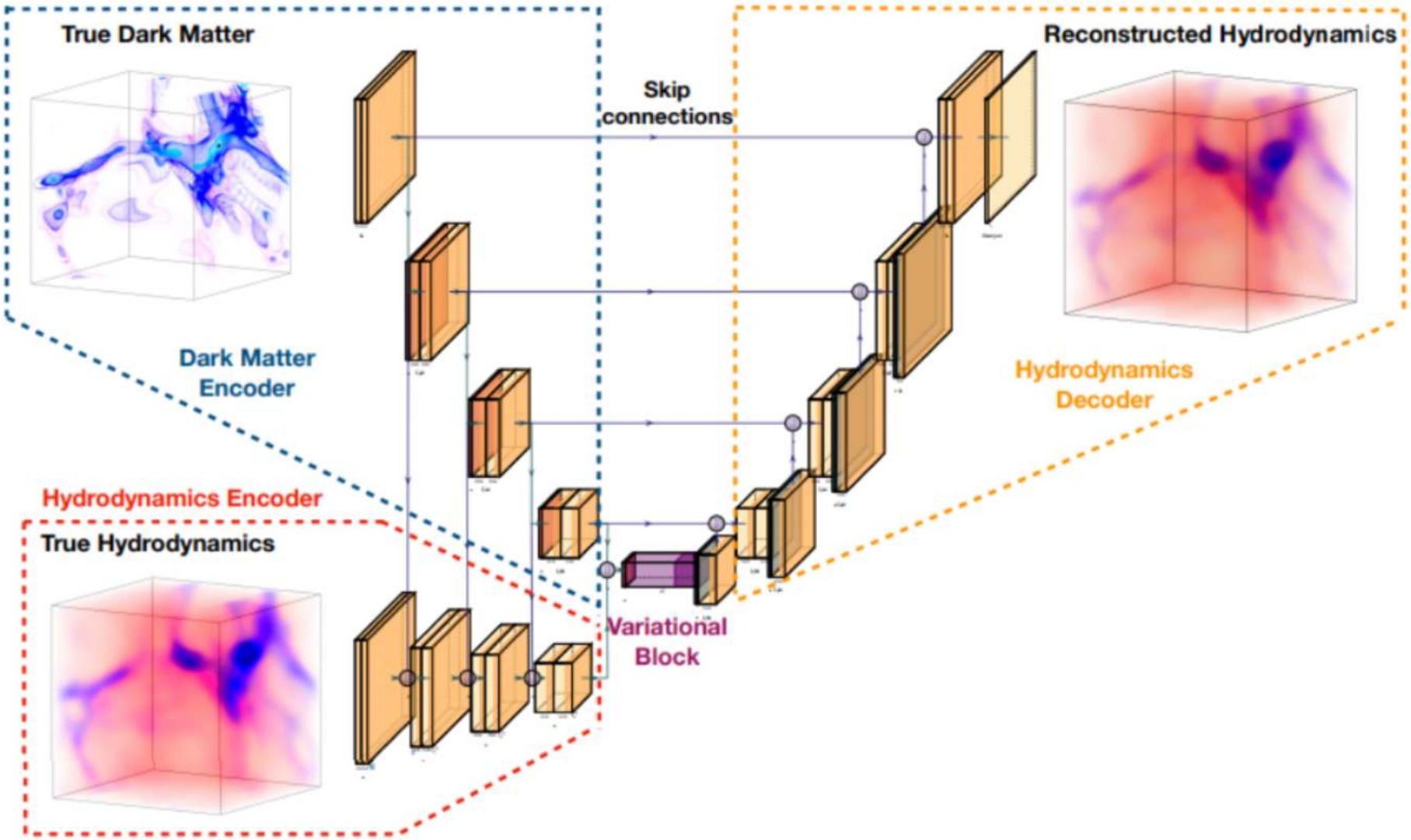
Physical information is included as well within the architecture

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

Painting Baryons



Rodriguez+19, Modi+18, Berger+18, He+18, Zhang+19, Troster+19, Zamudio-Fernandez+19, Perraudin+19, Charnock+19, List+19, Giusarma+19, Bernardini+19, Chardin+19, Mustafa+19, Ramanah+20, Tamosiunas+20, Feder+20, Moster+20, Thiele+20, Wadekar+20, Dai+20, Li+20, Lucie-Smith+20, Kasmanoff+20, Ni+21, Rouhiainen+21, Harrington+21, Horowitz+21, Horowitz+21, Bernardini+21, Schaurecker+21, Etezad-Razavi+21, Curtis+21

Horowitz+21

Generative Adversarial Networks

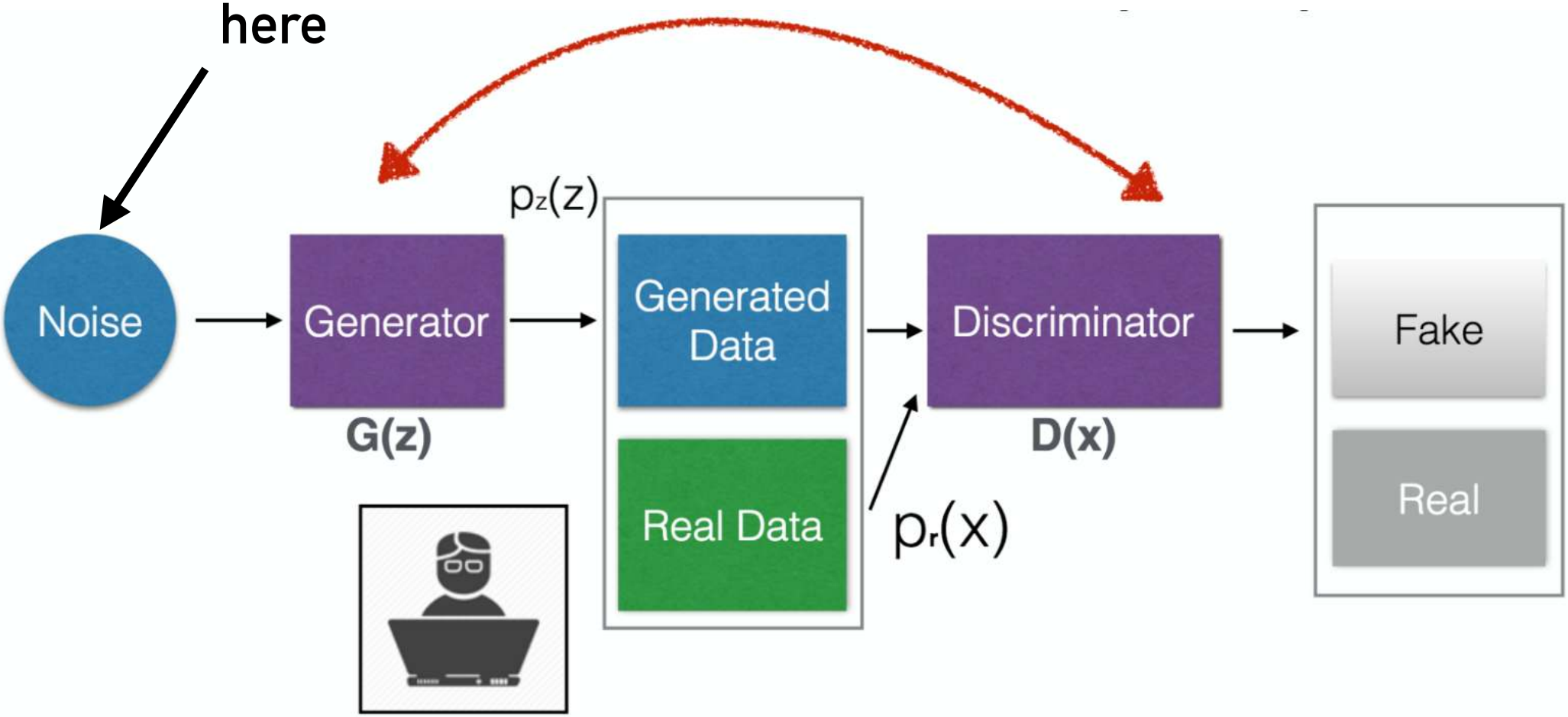
As for VAEs,
the goal of generative Adversarial Networks (GANs) is to estimate
$p(z|x)$

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

GENERATIVE ADVERSARIAL NETWORKS

(Goodfellow+14)

The latent variable is here

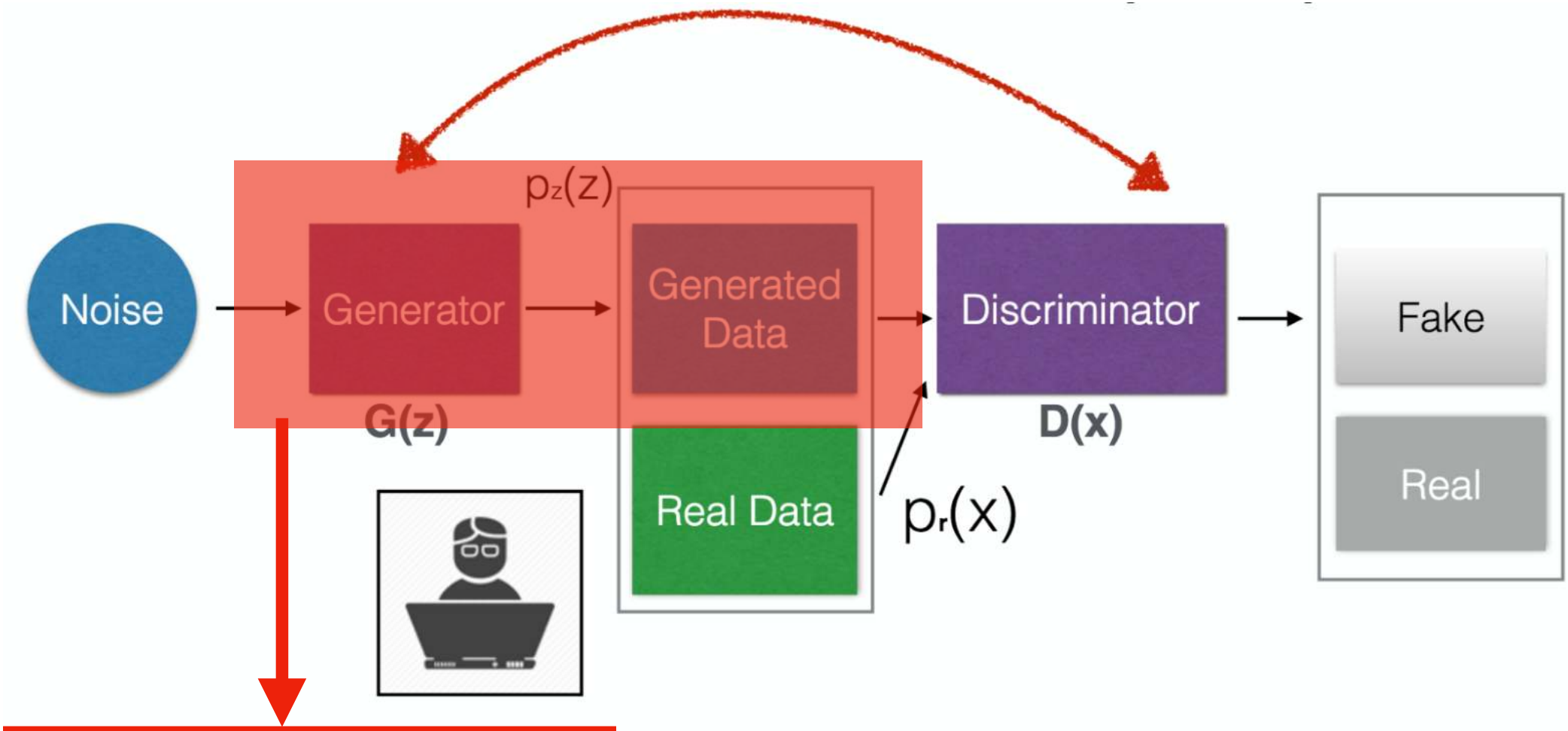


TWO COMPETING NETWORKS

GENERATIVE ADVERSARIAL NETWORKS

(Goodfellow+)

TWO COMPETING NETWORKS

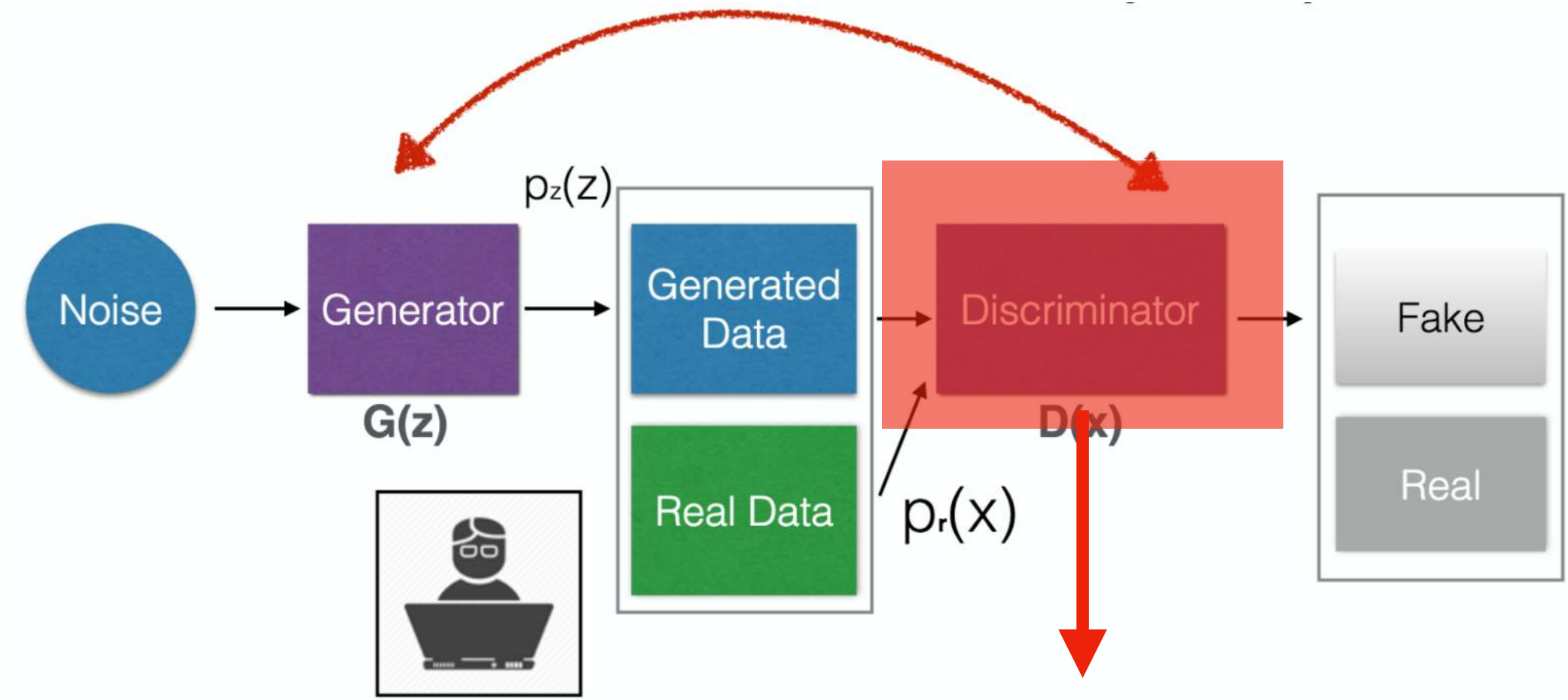


Every N iterations the generator is trained to force the discriminator to classify as real

GENERATIVE ADVERSARIAL NETWORKS

(Goodfellow+)

TWO COMPETING NETWORKS

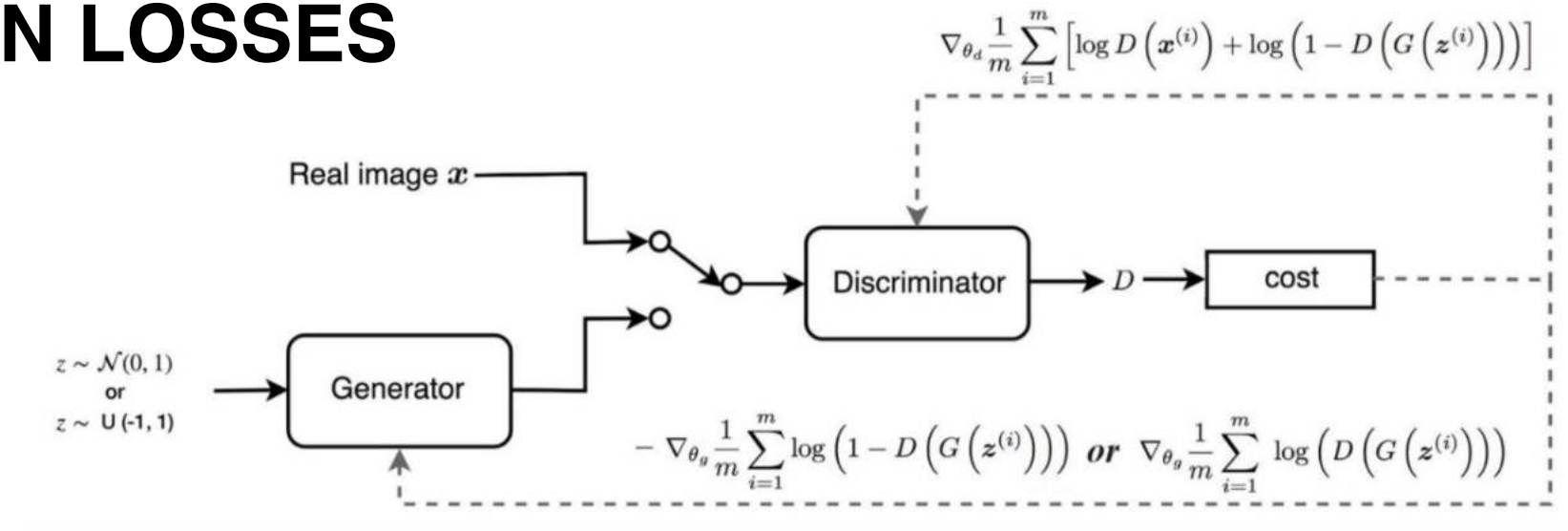


Every N iterations the discriminator is trained to force to distinguish between real and fake

IN PRACTICE

GAN LOSSES

DISCRIMINATOR LOSS (CROSS-ENTROPY)

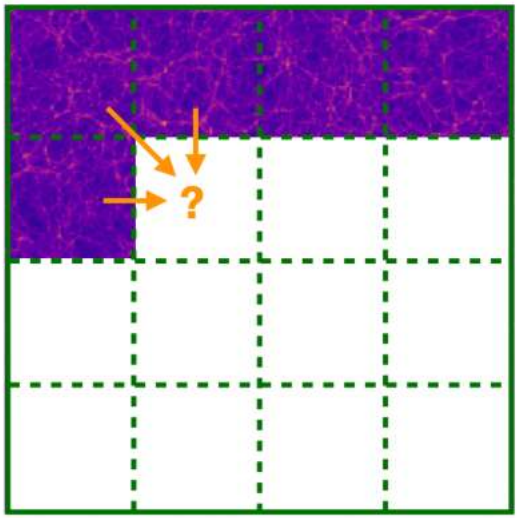


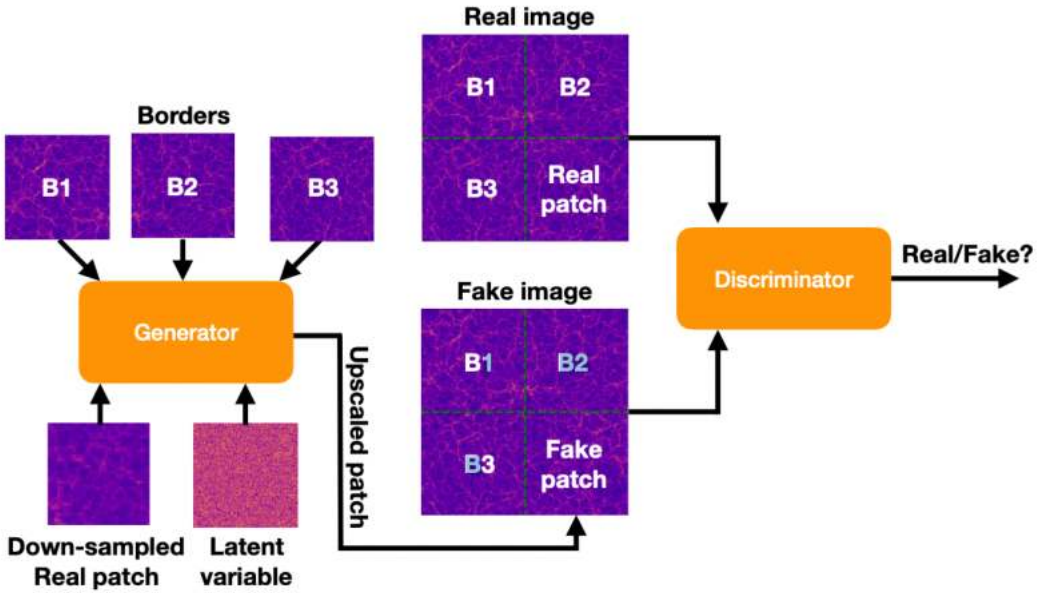
GENERATOR LOSS

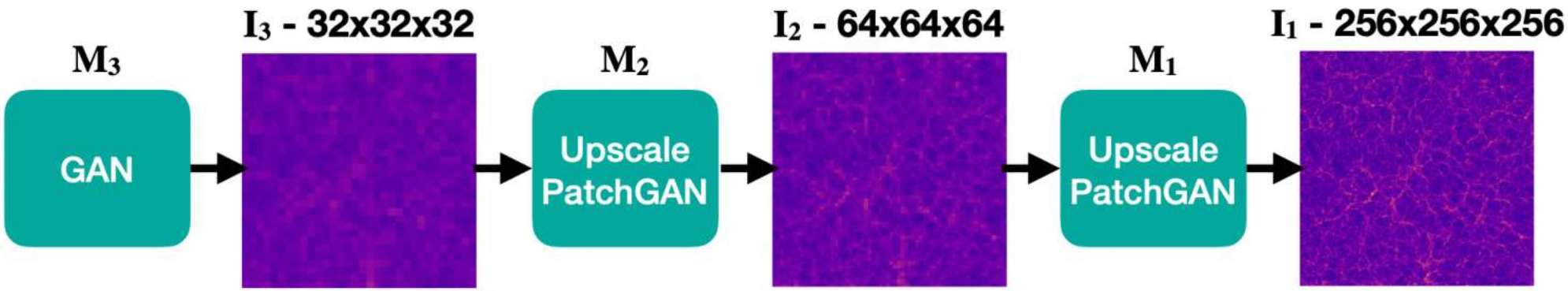
GANs have remained state-of-the-art for data generation (until diffusion…)

Karras+19

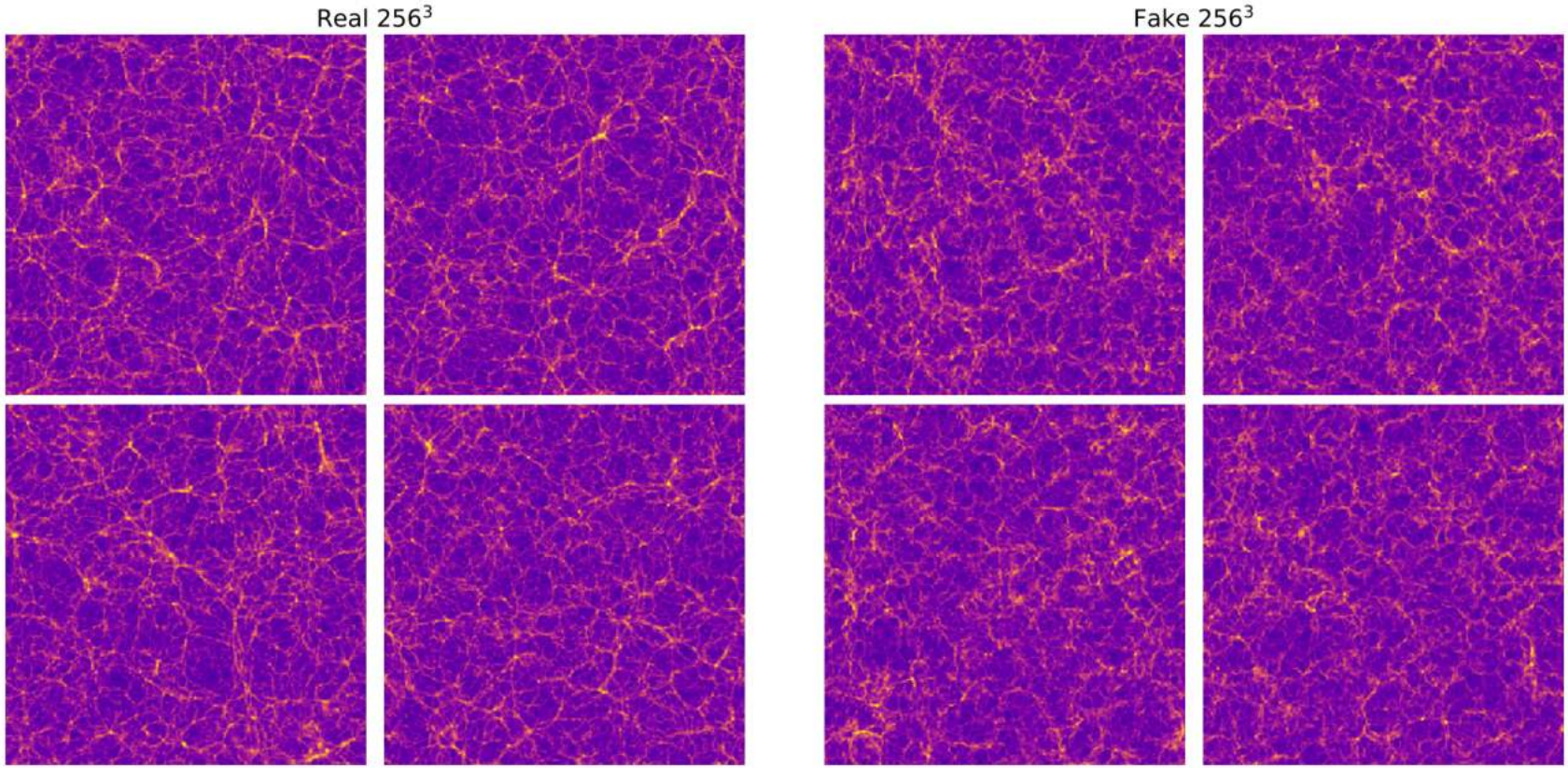
N-body emulation by Deep Generative Modelling

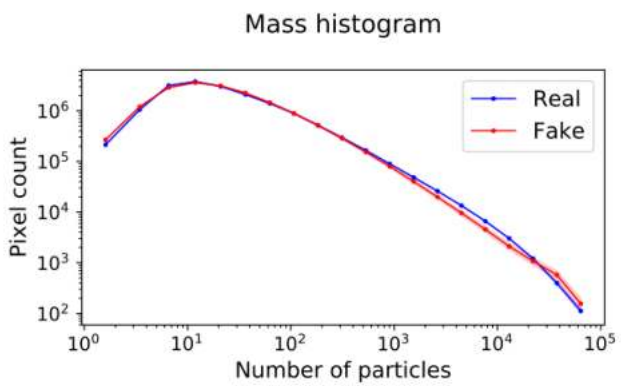


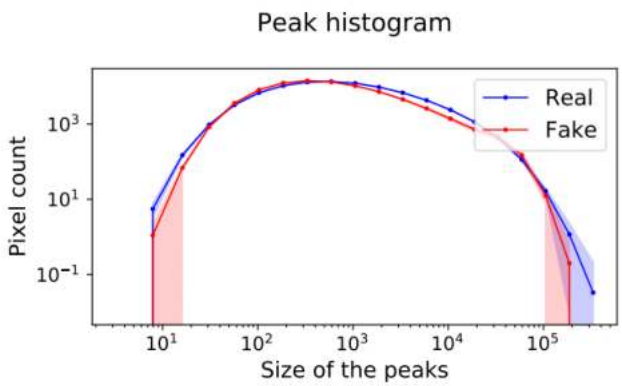


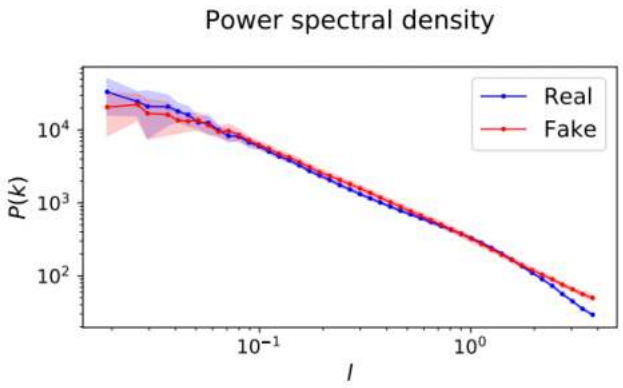


Perraudin+19

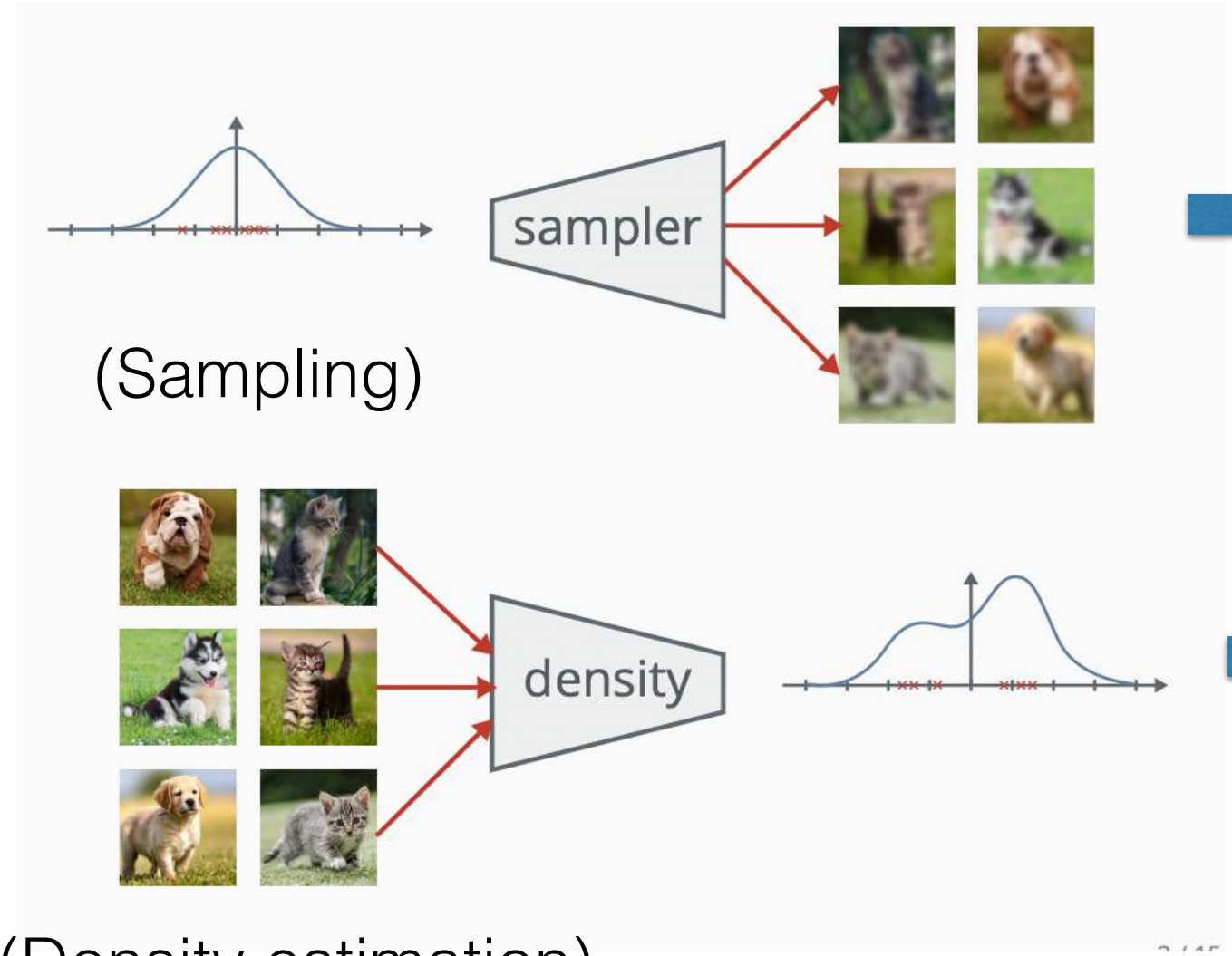








Perraudin+19



(Sampling)

(Density estimation)

easy with
VAEs and GANs

~ VAEs
difficult with
GANs

WHAT IS AN ANOMALY OR OUTLIER?

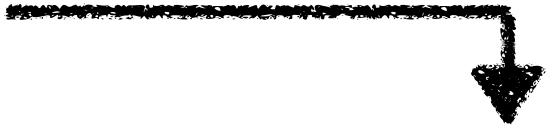
$$p(X)$$

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:

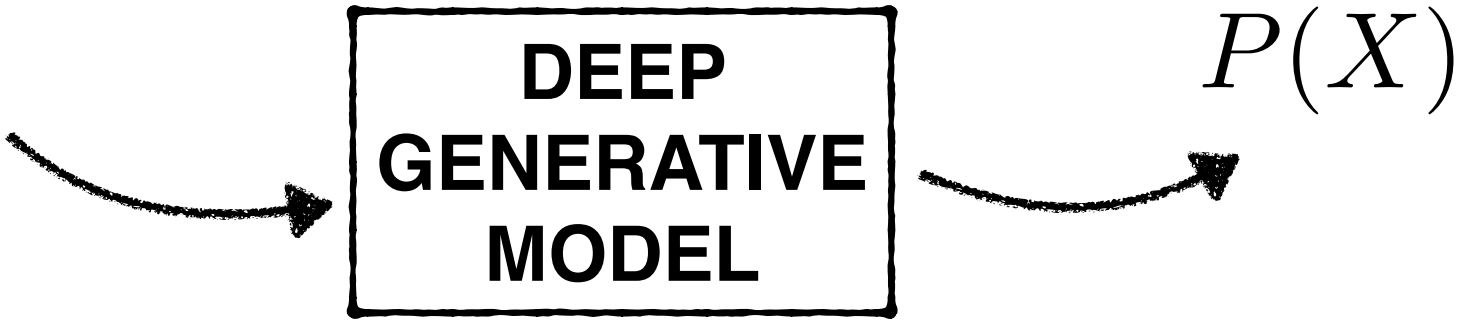
$$p(x_i) < \epsilon$$

HOW DO WE COMPUTE THE PROBABILITY DISTRIBUTION p?

GENERATIVE MODELS DO PRECISELY THAT:

DATA:

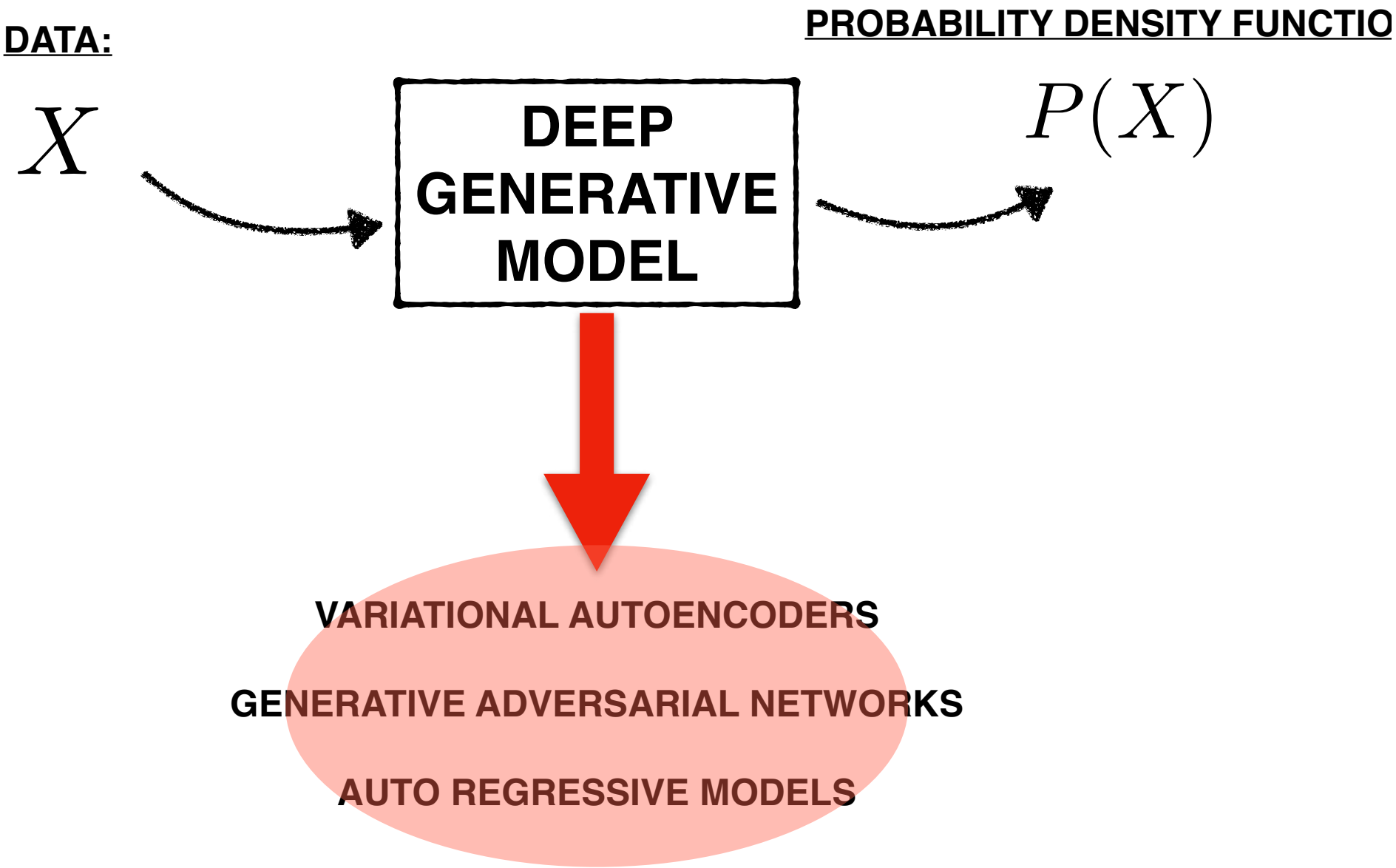
X



PROBABILITY DENSITY FUNCTIO

$P(X)$

GENERATIVE MODELS DO PRECISELY THAT:



Razavi+19

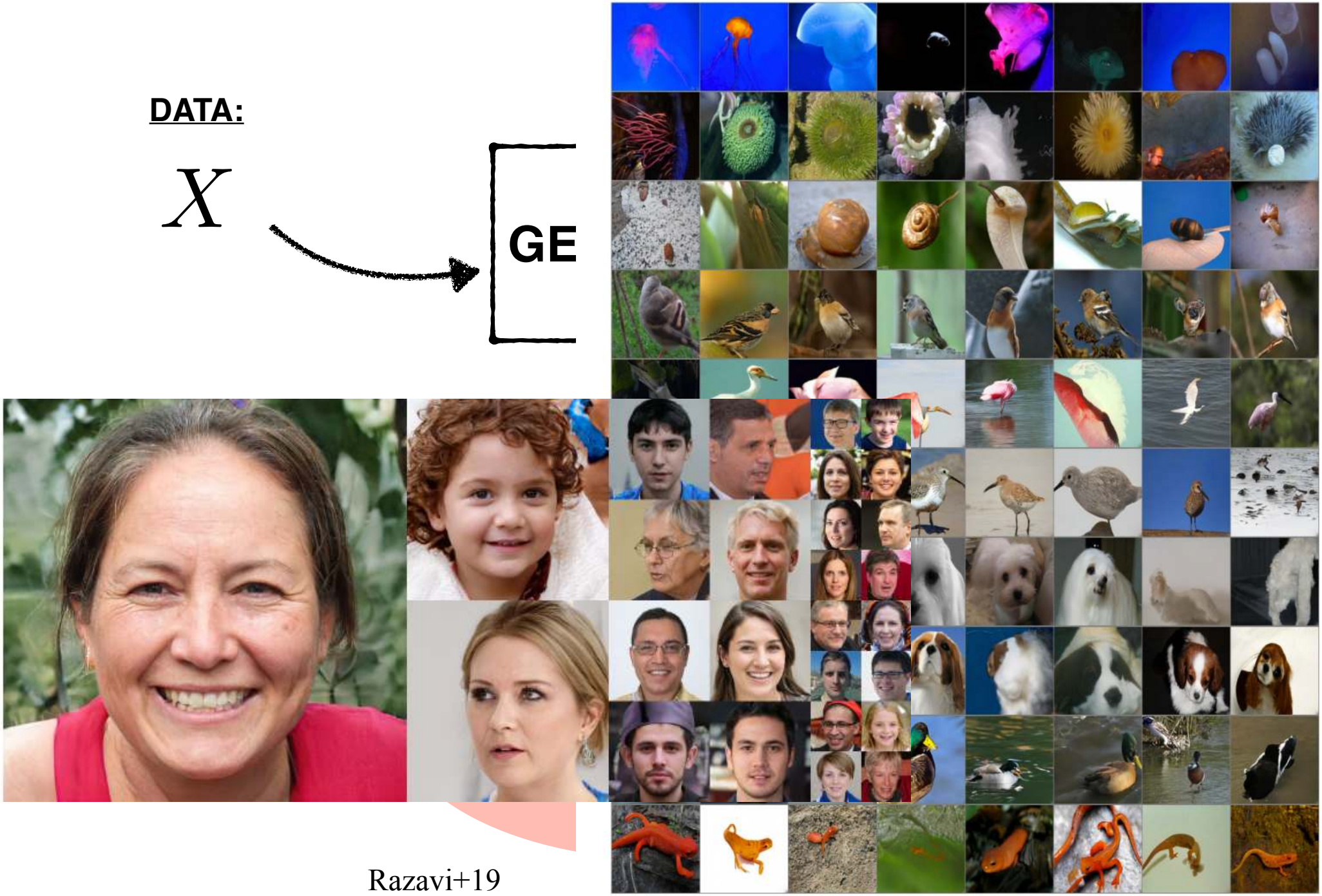
GENERATIVE MODELS DO PRECISELY THAT:

DATA:

X

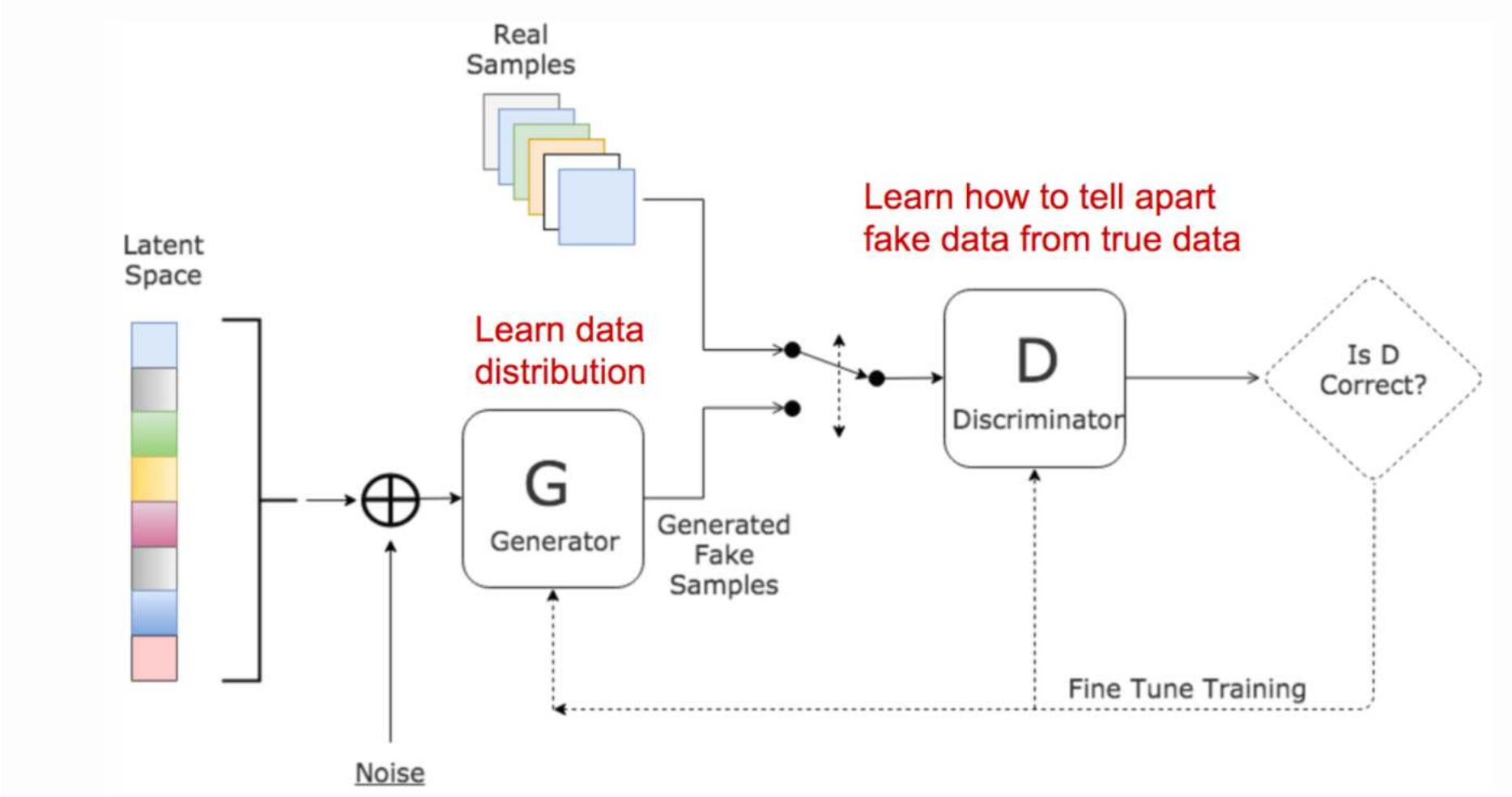
GE

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)

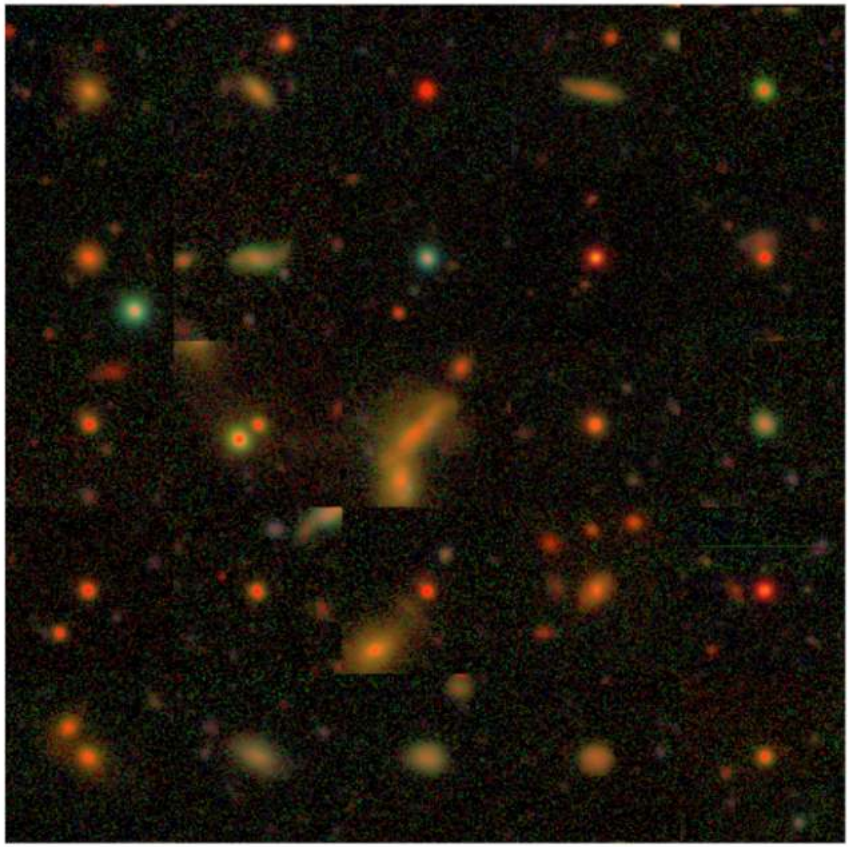
$P_{HSC}(x_i)$

identify most "unlikely" objects

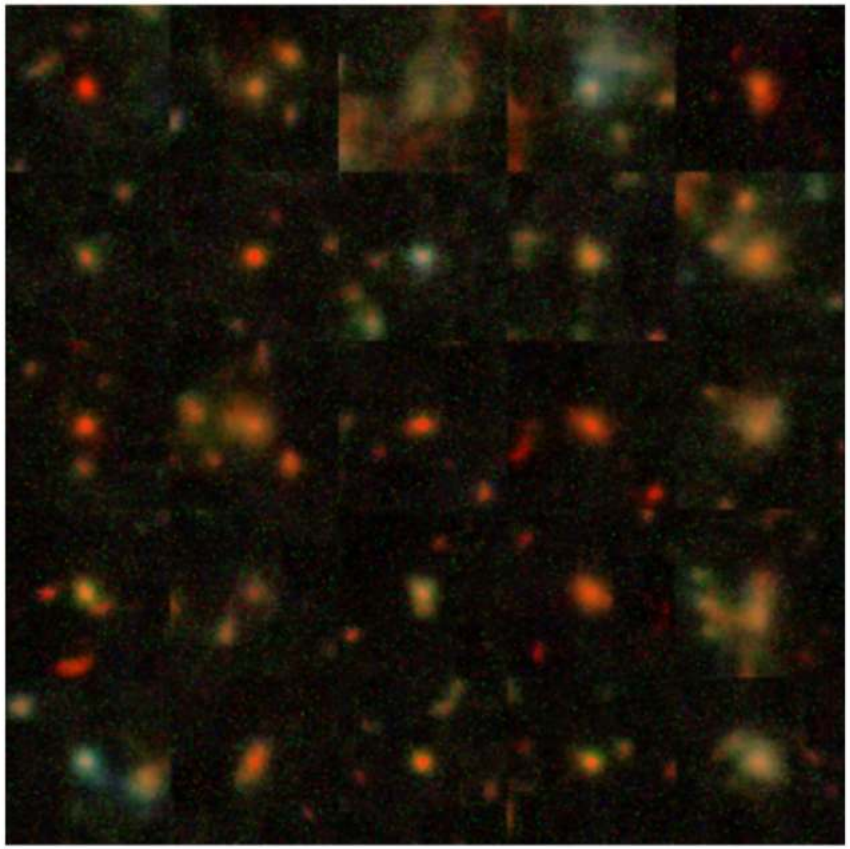
Storey-Fisher, MHC+21

Margalef-Bentabol, MHC+20

REAL



WGAN GENERATED



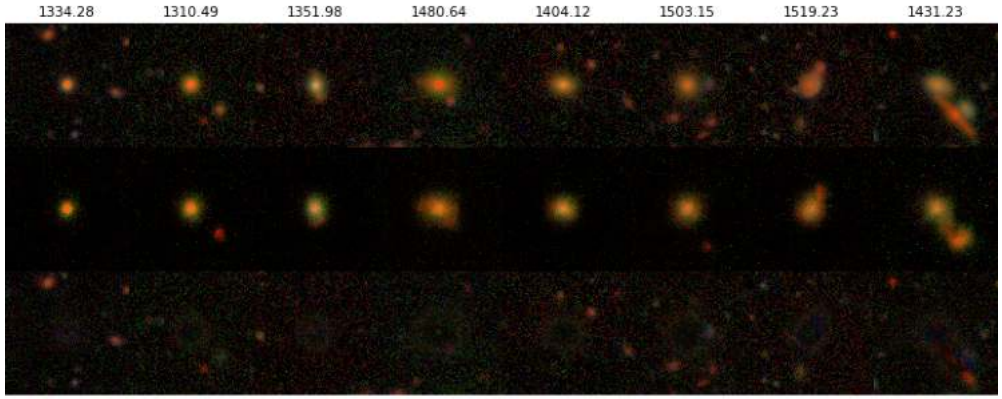
COMPUTE ANOMALY SCORE BASED ON WGAN RECONSTRUCTION ERROR

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

Hyperparameter

RMS btw input and generated

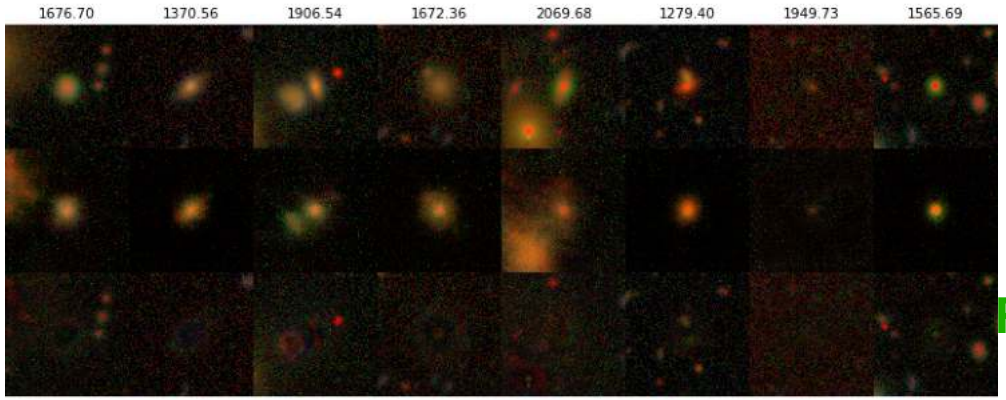
RMS btw critic features from input and generated



REAL

BEST RECONSTRUCTION

RESIDUAL



REAL

BEST RECONSTRUCTION

RESIDUAL

Schlegl+17

Storey-Fisher+21

Margalef-Bentabol, MHC+20

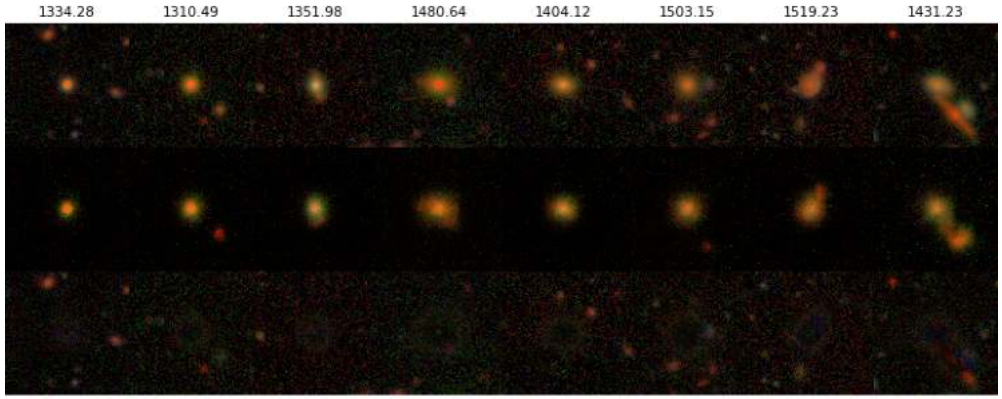
COMPUTE ANOMALY SCORE BASED ON WGAN RECONSTRUCTION ERROR

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

Hyperparameter

RMS btw input and generated

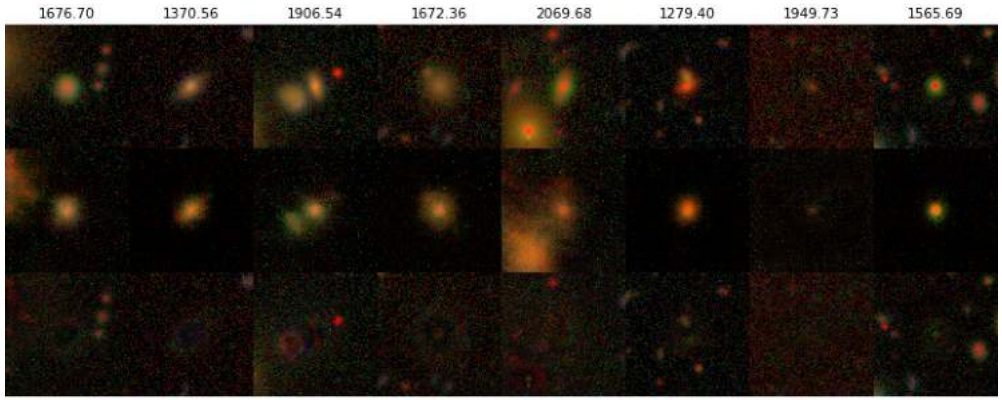
RMS btw critic features from input and generated



REAL

BEST RECONSTRUCTION

RESIDUAL



REAL

BEST RECONSTRUCTION

RESIDUAL

Schlegl+17

Storey-Fisher+20 (in prep)

Margalef-Bentabol, MHC+20

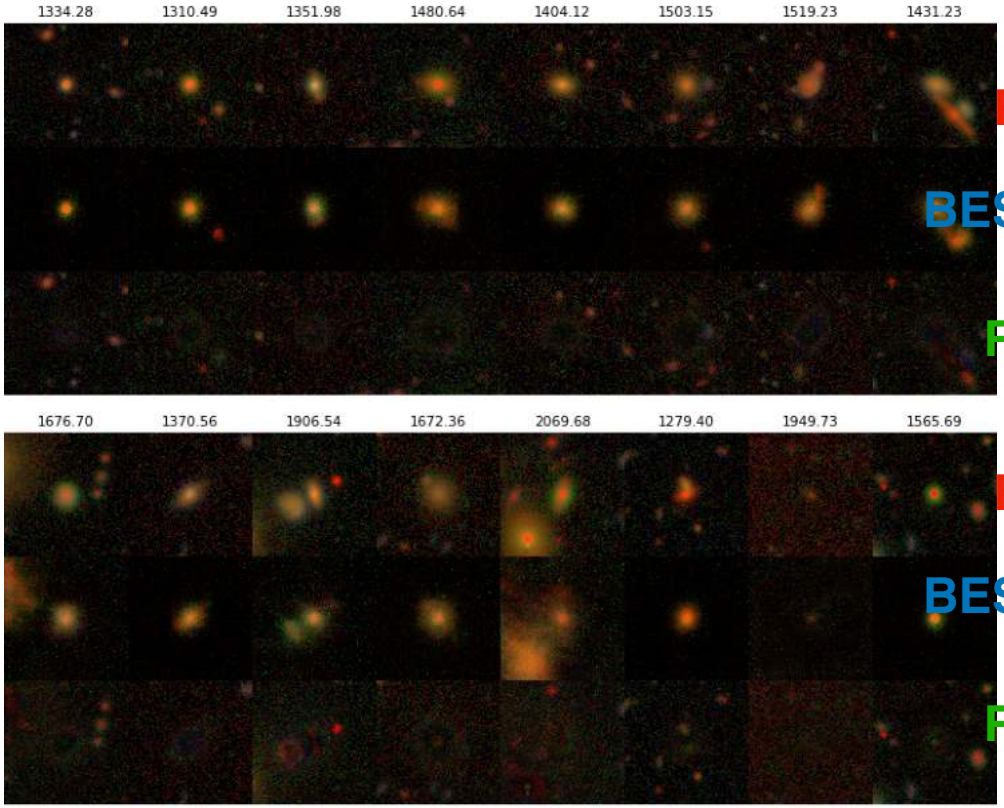
COMPUTE ANOMALY SCORE BASED ON WGAN RECONSTRUCTION ERROR

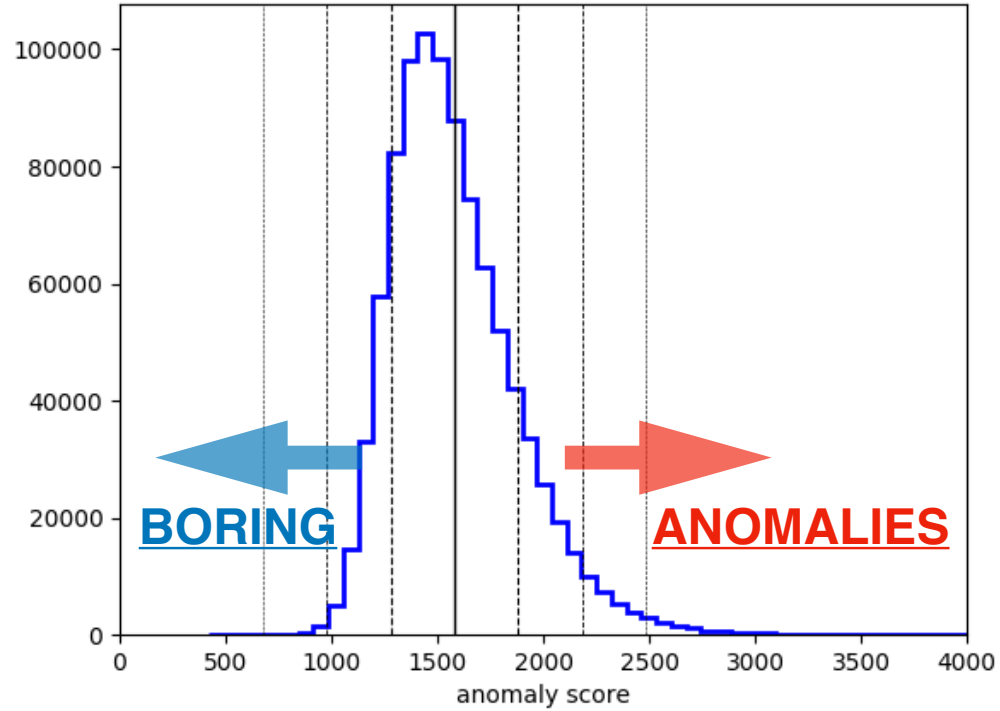
ANOMALY SCORE: $AS = \lambda G + (1 - \lambda)C$

Hyperparameter

RMS
and

RMS btw critic features

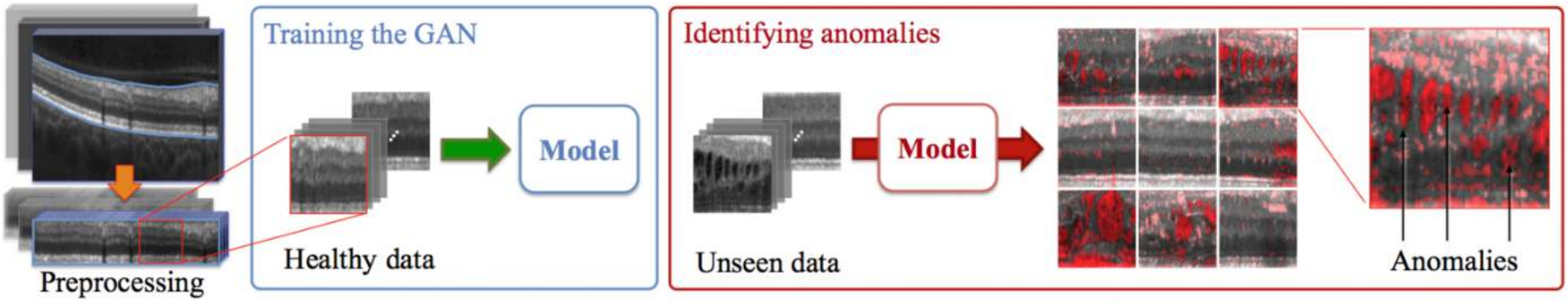


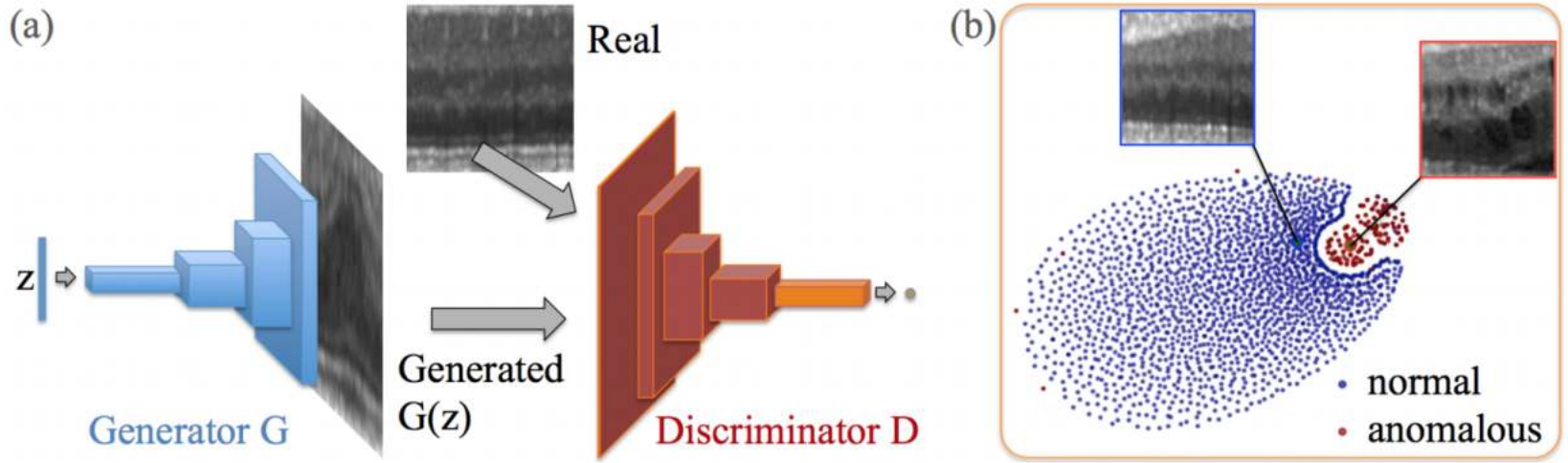


Storey-Fisher+21

Margalef-Bentabol, MHC+20

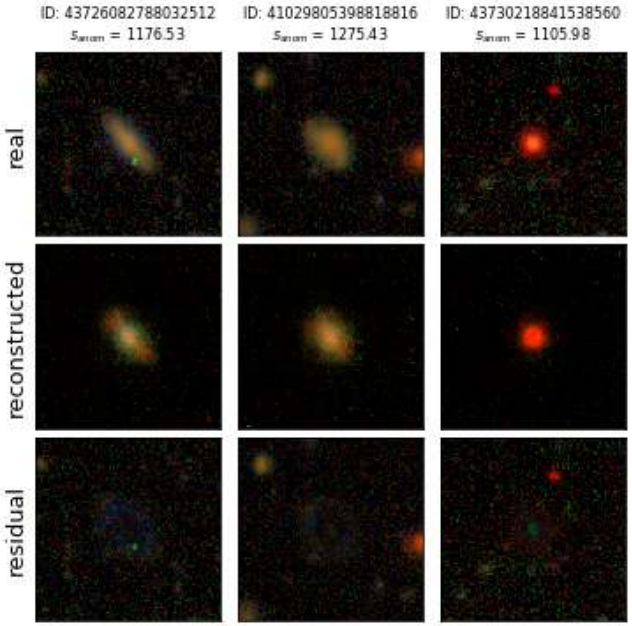
This is again inspired by biomedical applications



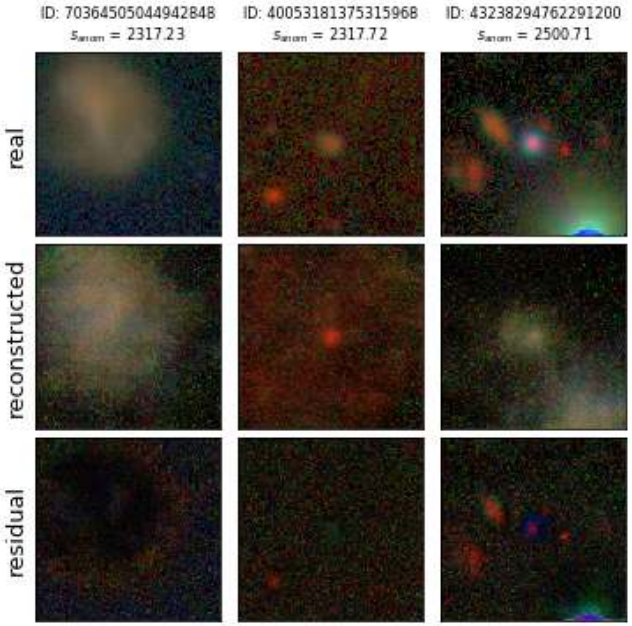


Schlegl+17

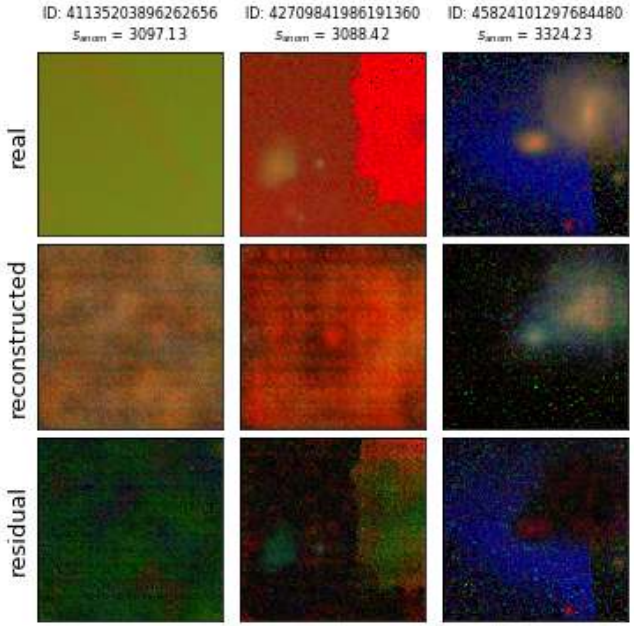
AVERAGE



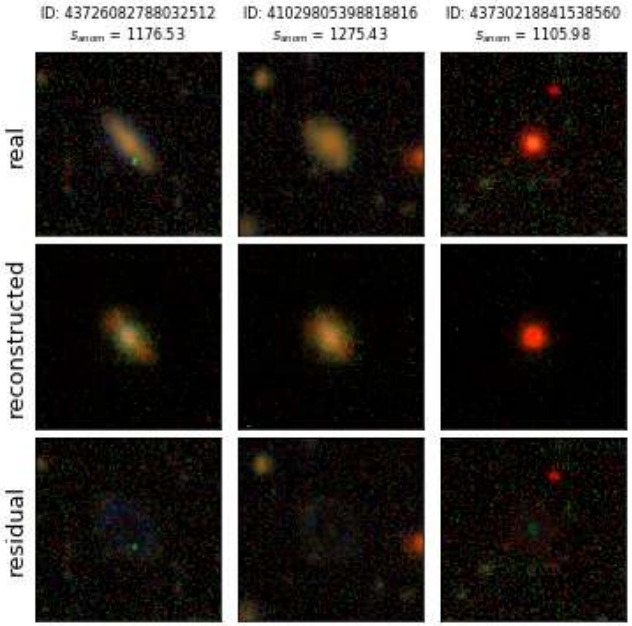
1-SIGMA



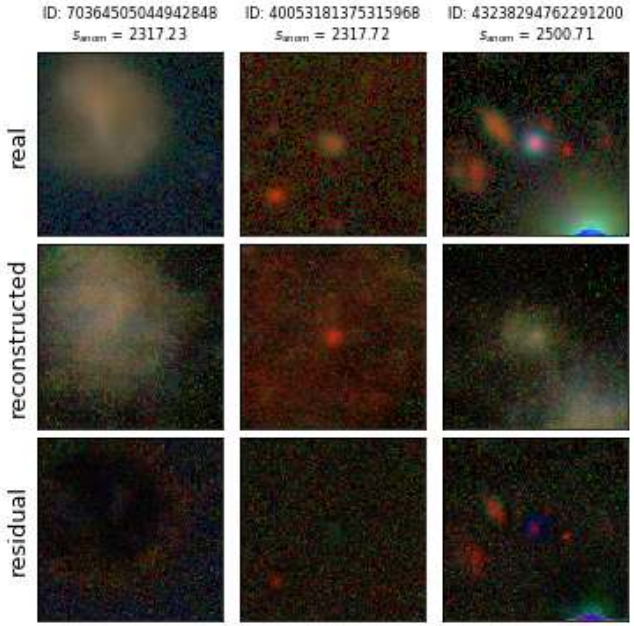
>3-SIGMA



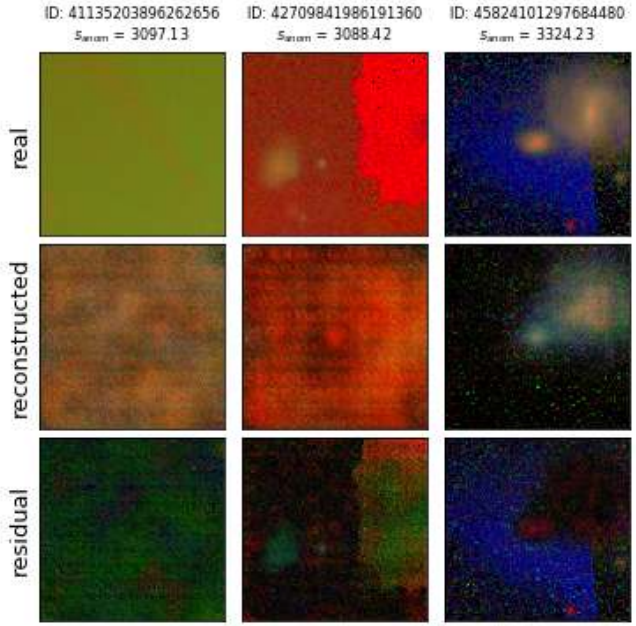
AVERAGE



1-SIGMA

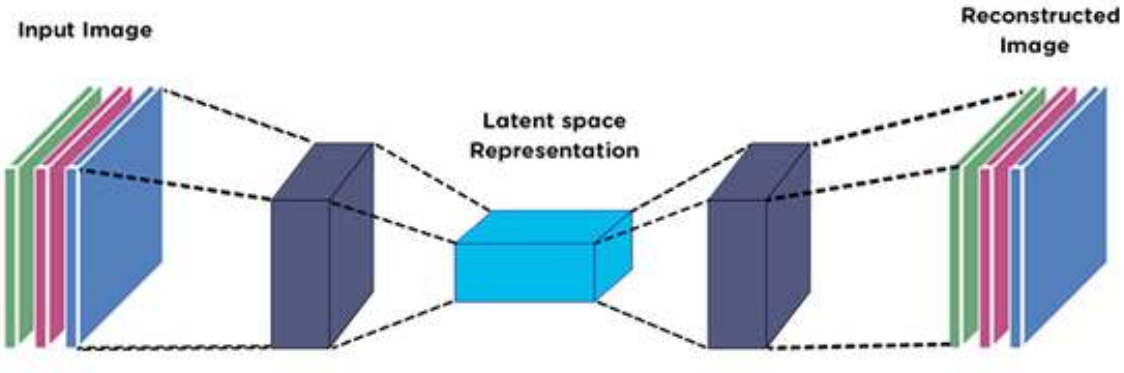


>3-SIGMA

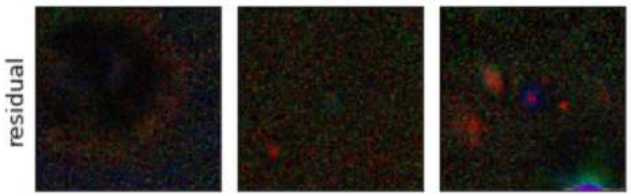


HOW DO WE FILTER OUT "INTERESTING" ANOMALIES?

HUNTING "INTERESTING" OUTLIERS



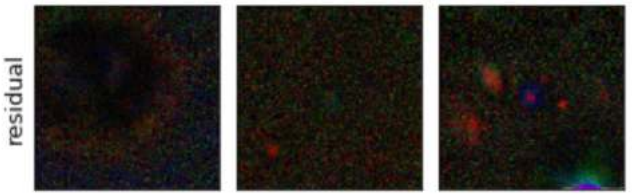
D~32



residual images

D~27,000

32 dimension space

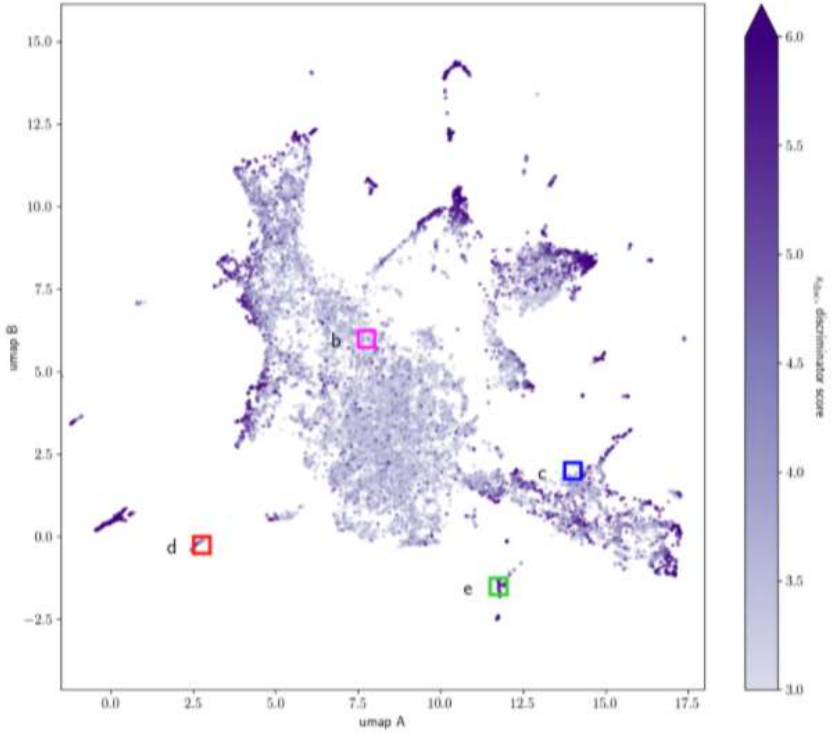


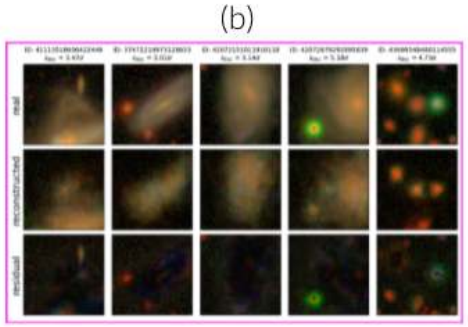
residual images

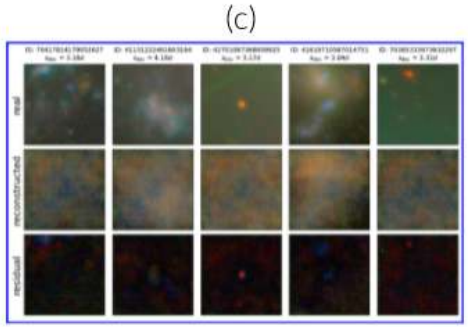
D~27,000

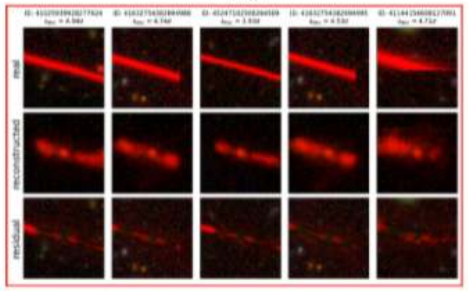
AUTO-ENCODERS TO REDUCE THE DIMENSIONALITY OF RESIDUAL IMAGES

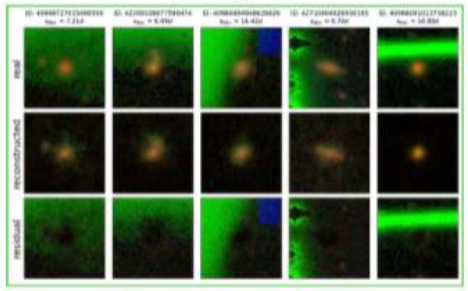
UMAP of high-anomaly sample ($s_{disc} > 3\sigma$)



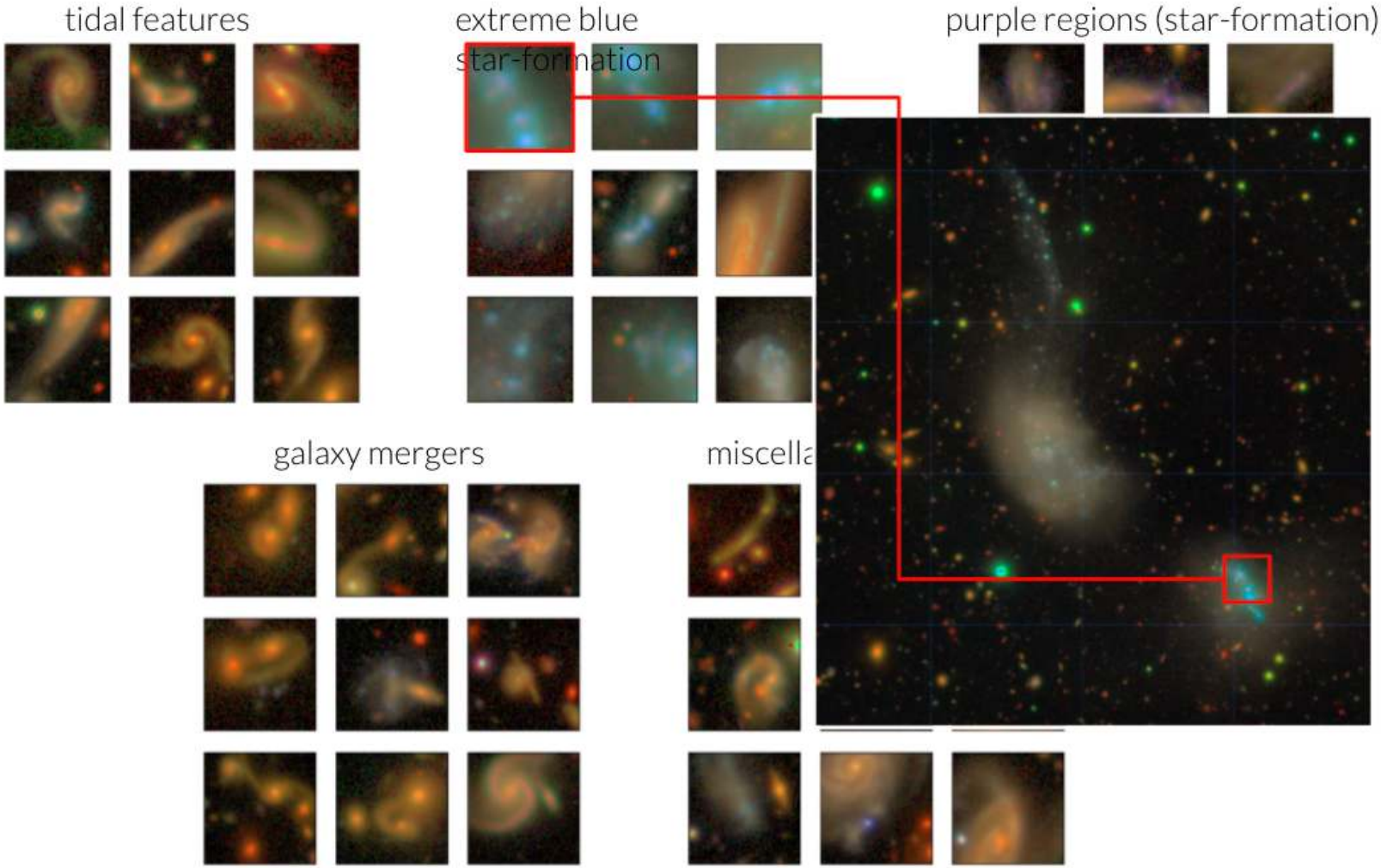








Detected interesting anomalies



3. Density Estimation for likelihood evaluation: simulation based inference

Normalizing Flows

Based on change of variables:

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

unknown pdf

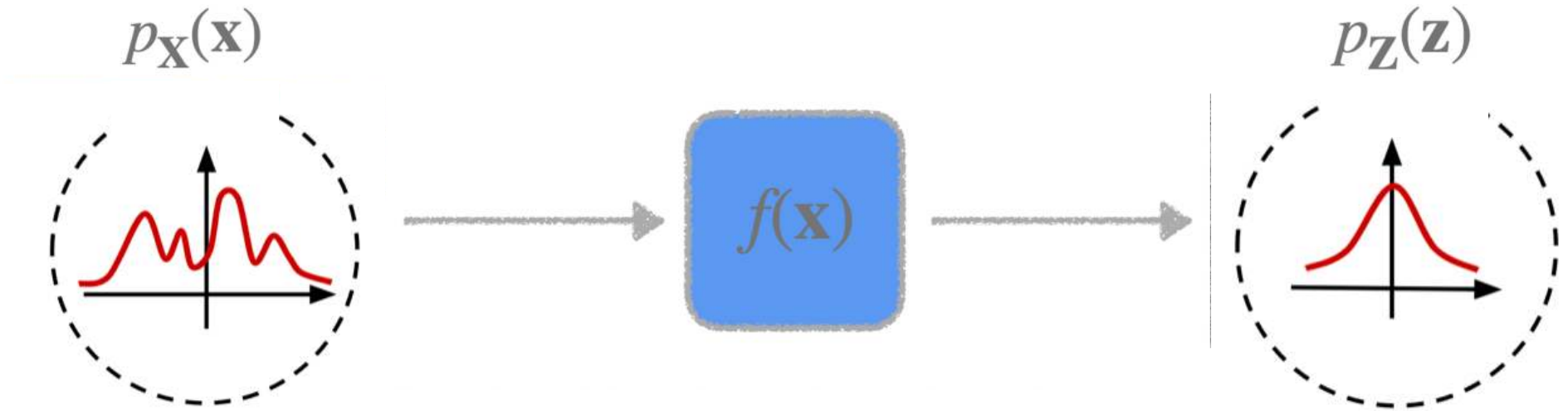
tractable pdf
(typically Gaussian)

Invertible differentiable
function

Determinant of the Jacobian
of f(x)

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

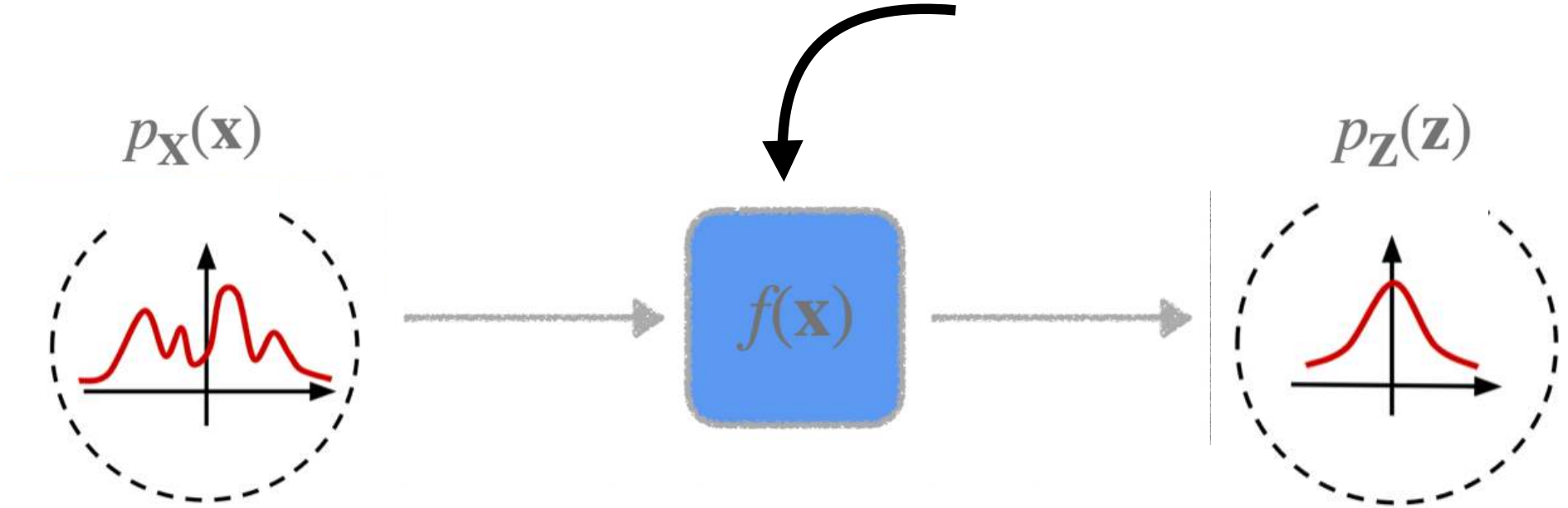
Can represent px(x) in terms of f(x) and pz(x)
(notice there is no dimensionality reduction as opposed to VAEs)



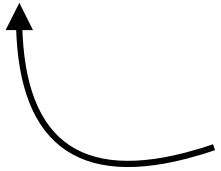
$$p_X(x) = p_Z(f(x))|detJ_{f(x)}|$$

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

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



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



NN learnable weights

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

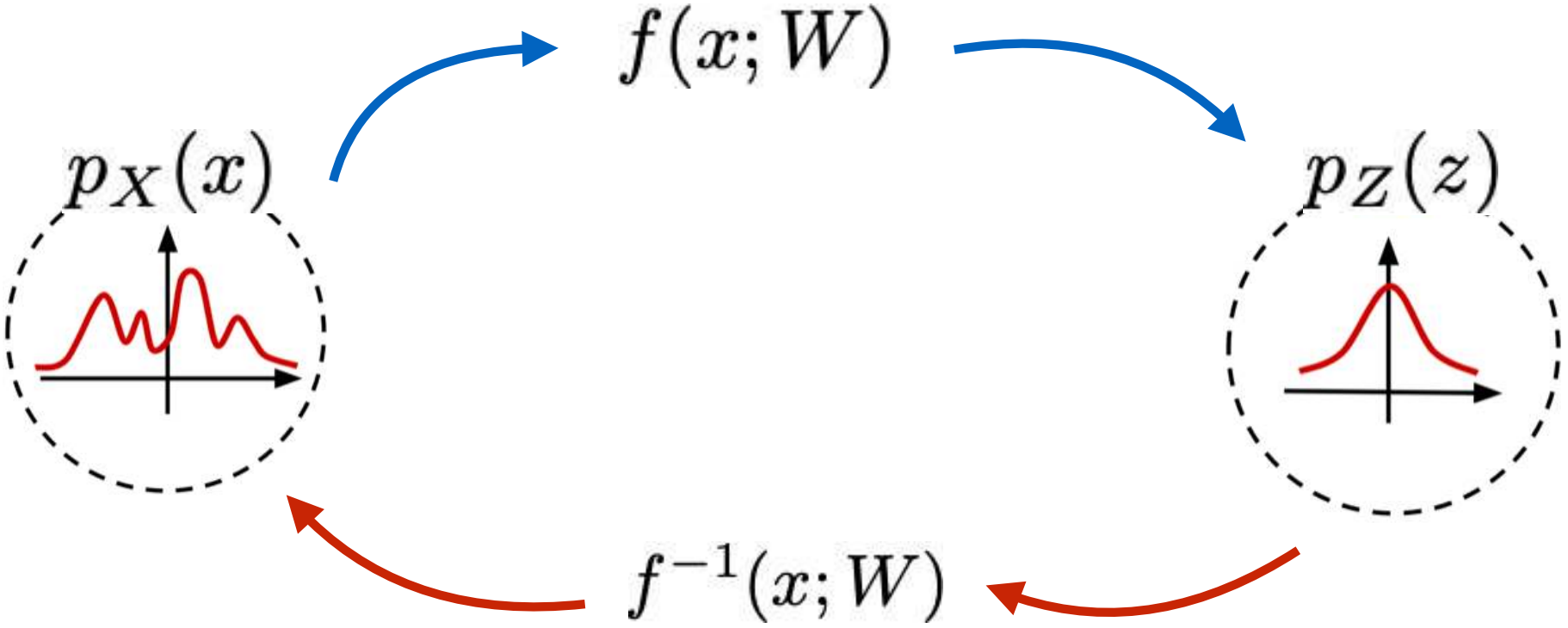
NN learnable weights

And the loss function: log likelihood

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

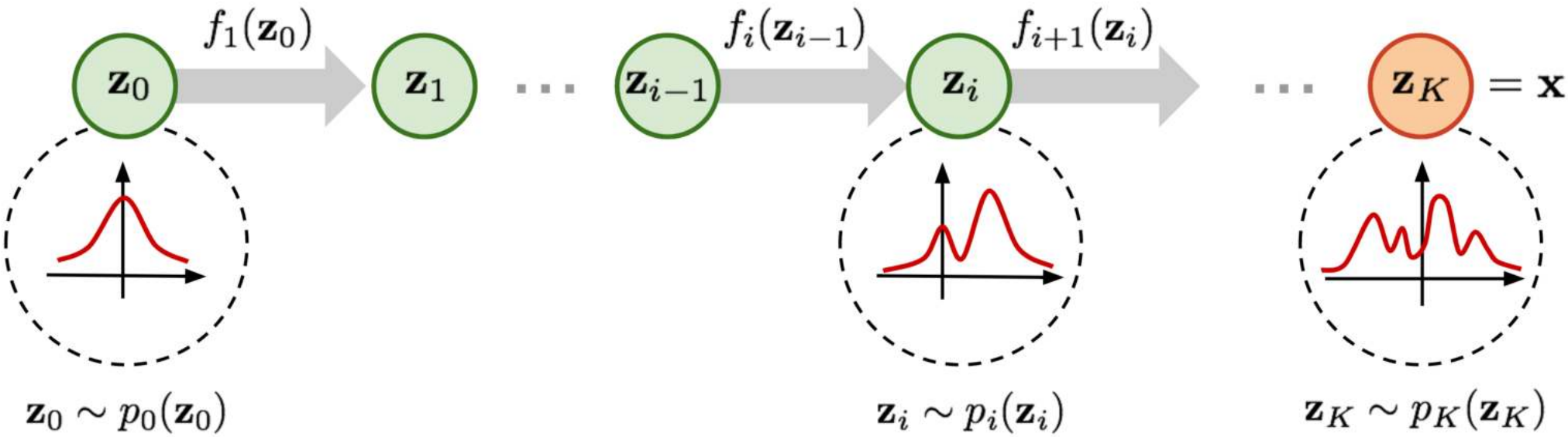
This is a gaussian

Density 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?

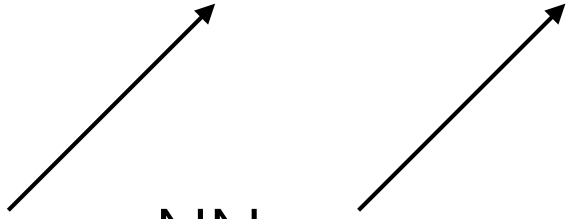
- 1. Easily invertible
- 2. Jacobian easy to compute

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

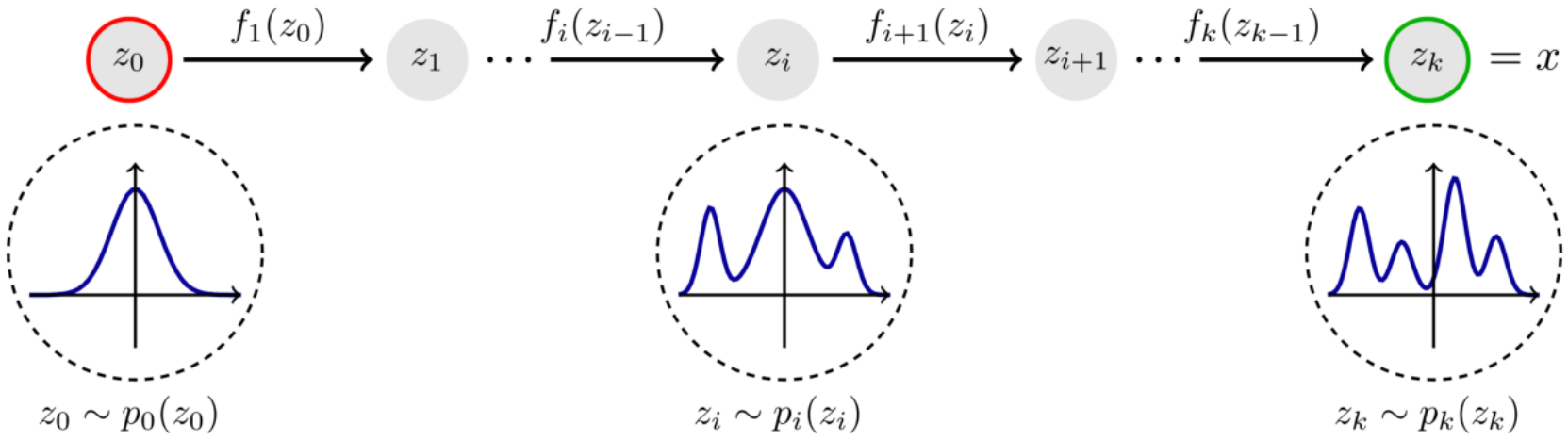
- Easily invertible:

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

- Jacobian:

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

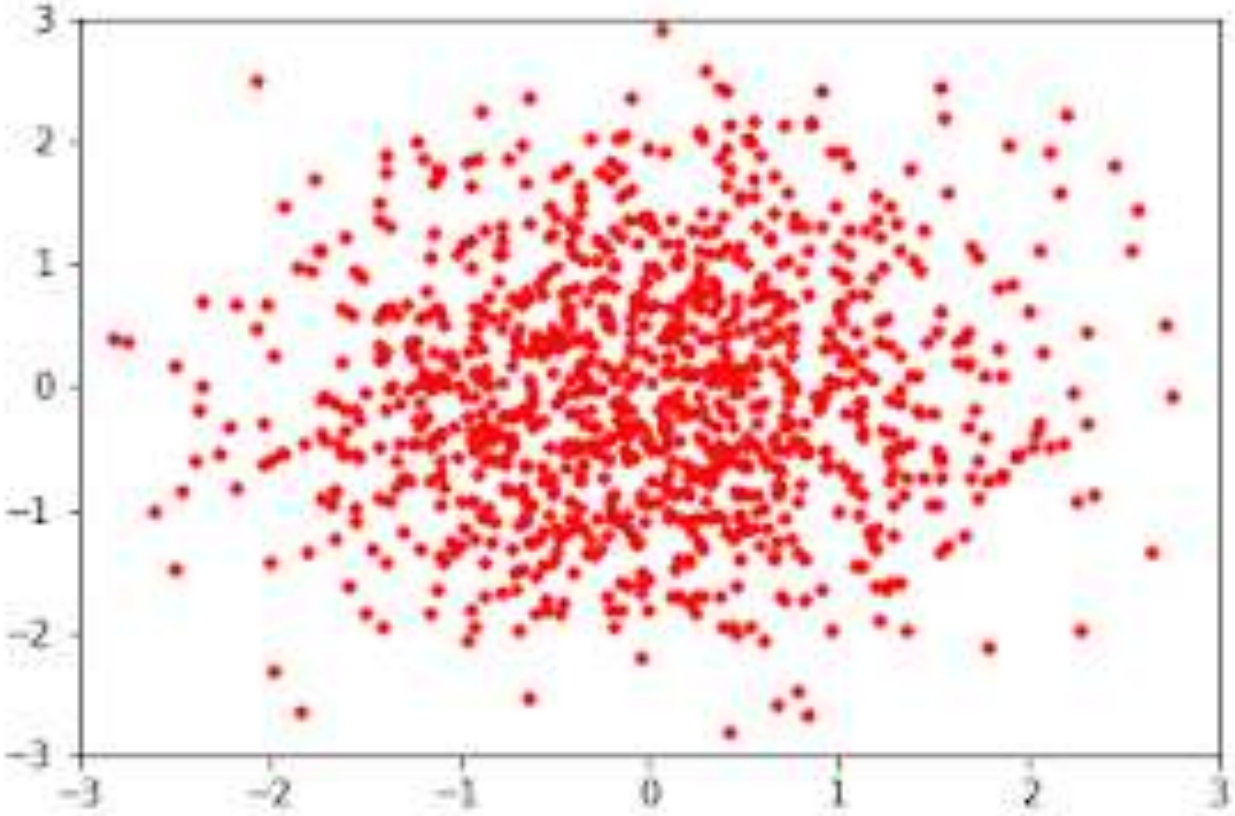
normalizing flows are generative models that are easy to evaluate and flexibly expressive



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

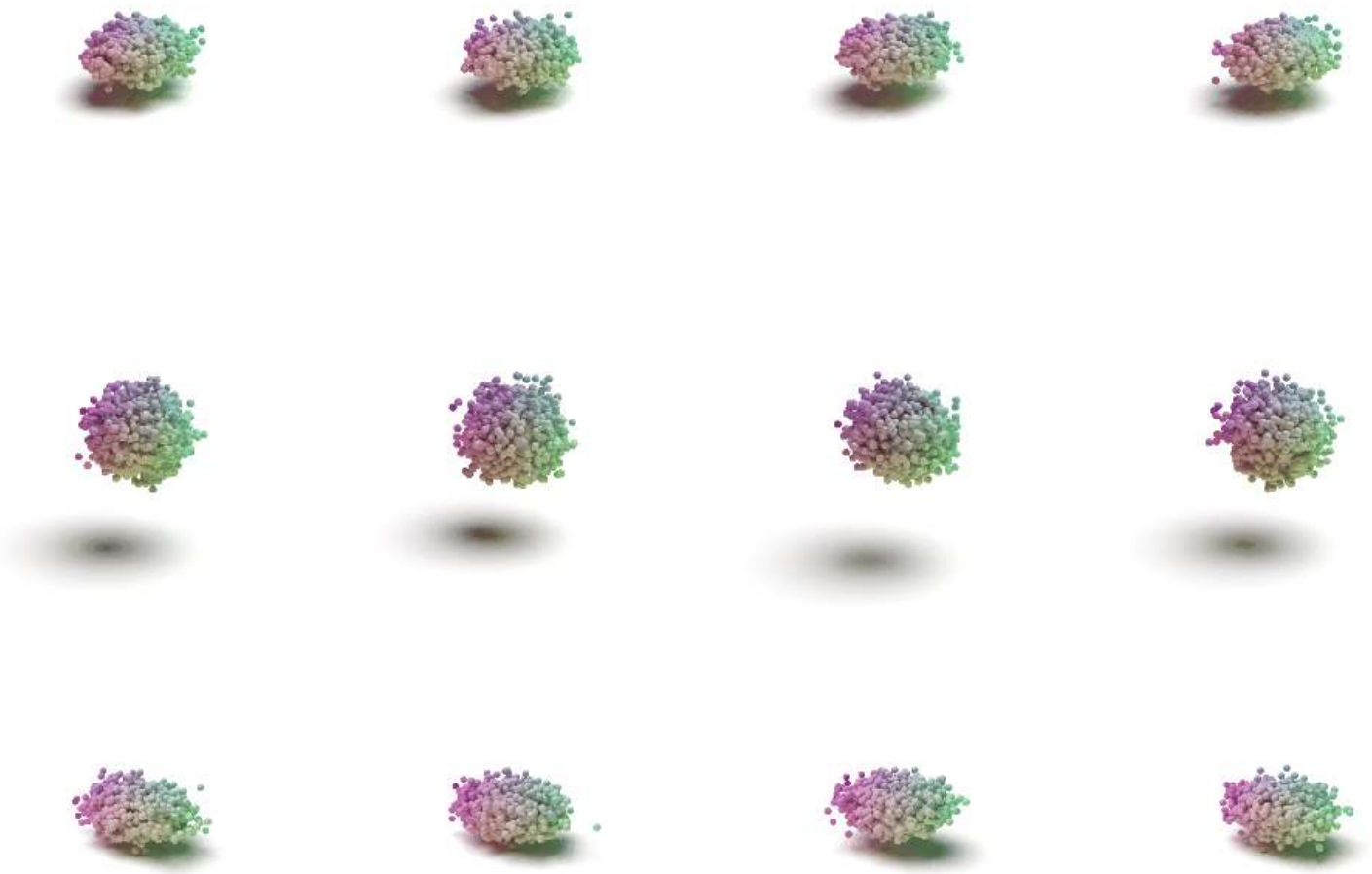
$f = f_1 \circ f_2 \ldots \circ f_{k-1} \circ f_k$ is also invertible and differentiable

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

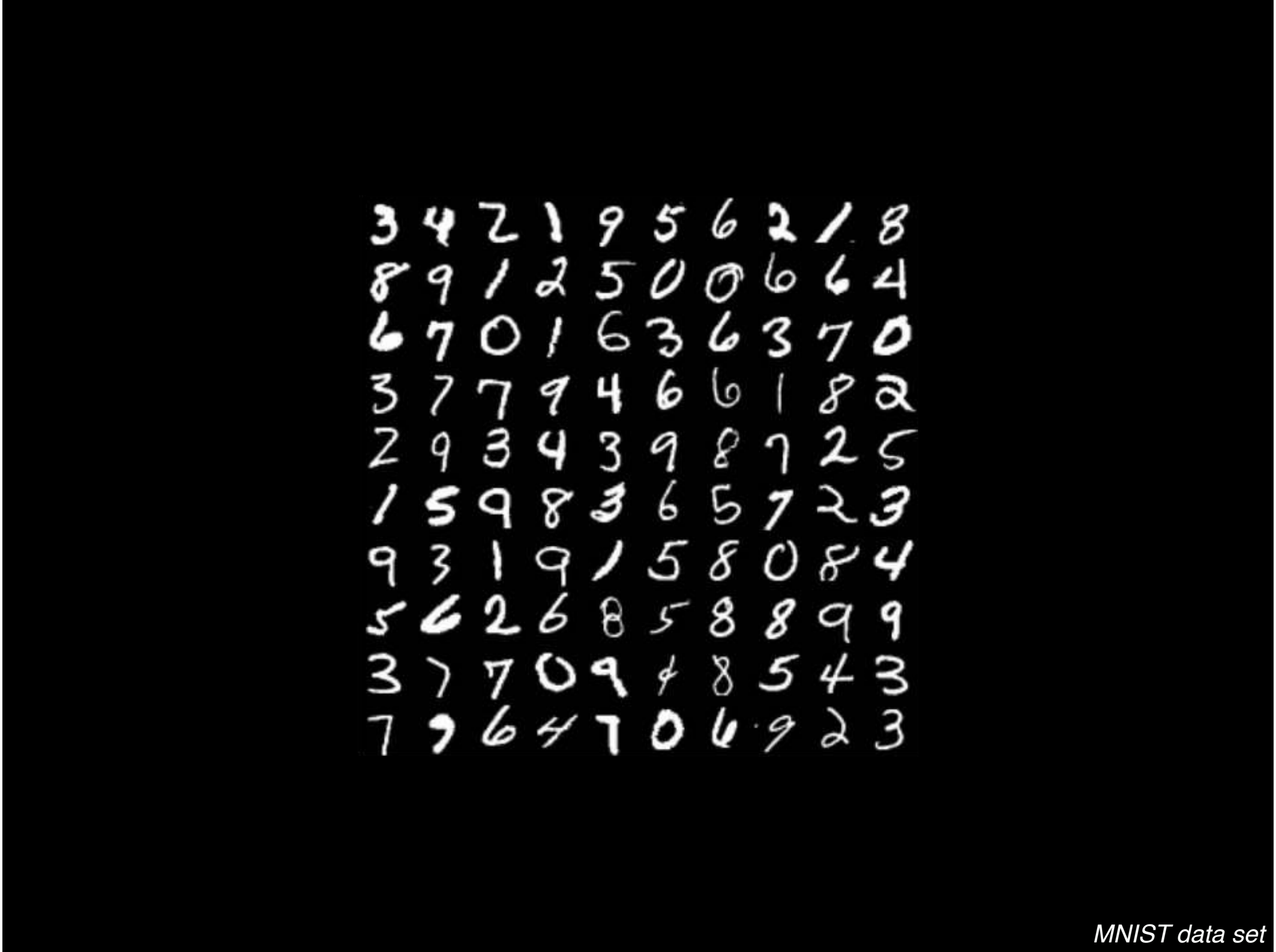


credit: Eric Jang

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

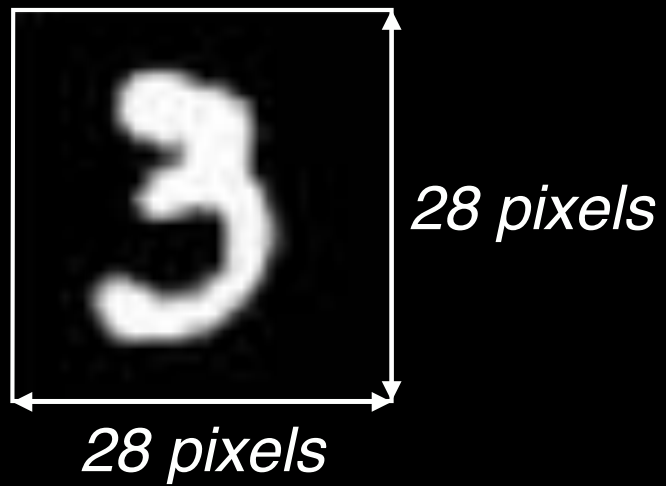


Yang+(2019) PointFlow



MNIST data set

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

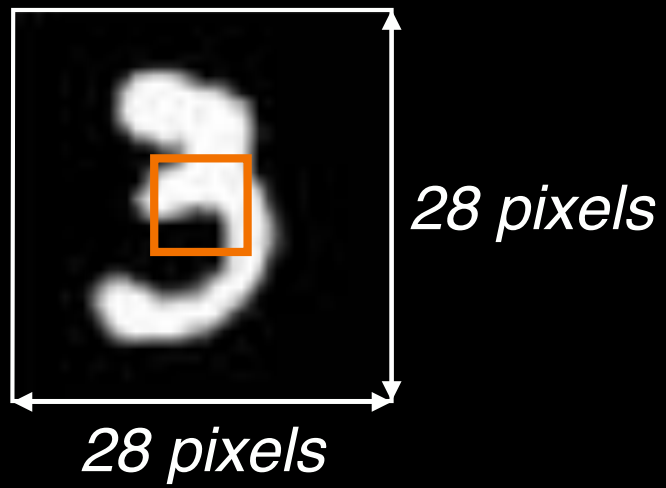


MNIST data set

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

sample q_ϕ(pixels)

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

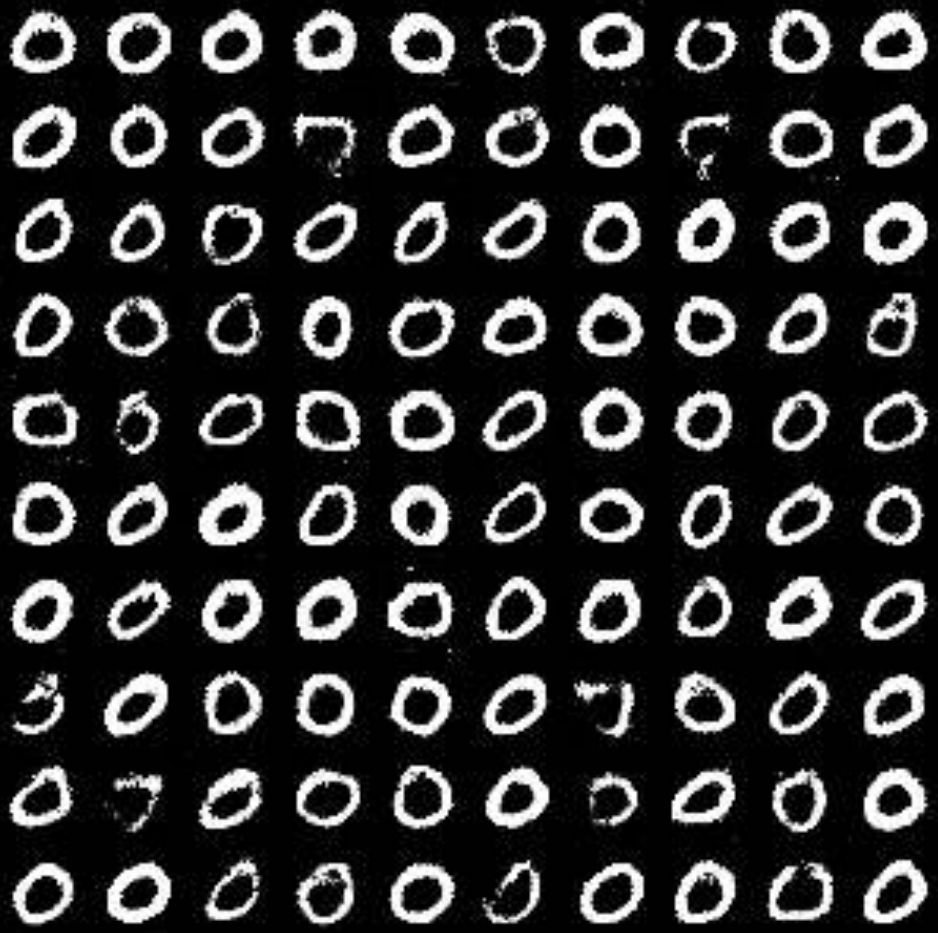


$$p(\text{pixels} \mid \text{central pixels}) \approx q_\phi(\text{pixels} \mid \text{central pixels})$$

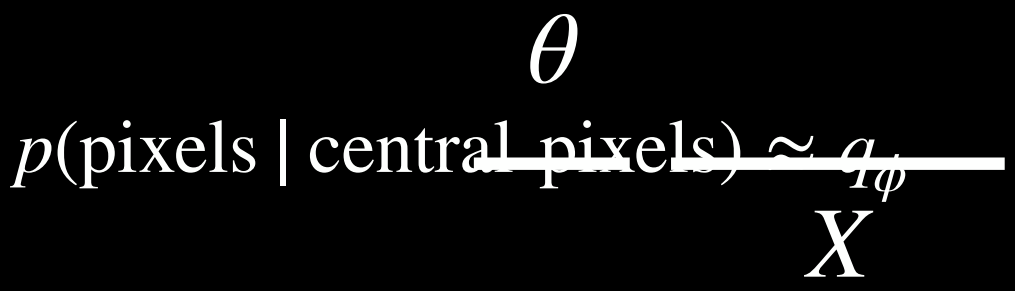
train a generative model to estimate **conditional distributions**

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

GAN: minimax the classification error loss.

VAE: maximize ELBO.

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