Lyon, Avril 21- 25

DLMI2025

Deep Learning for Medical Imaging

Spring school

Generative neural networks

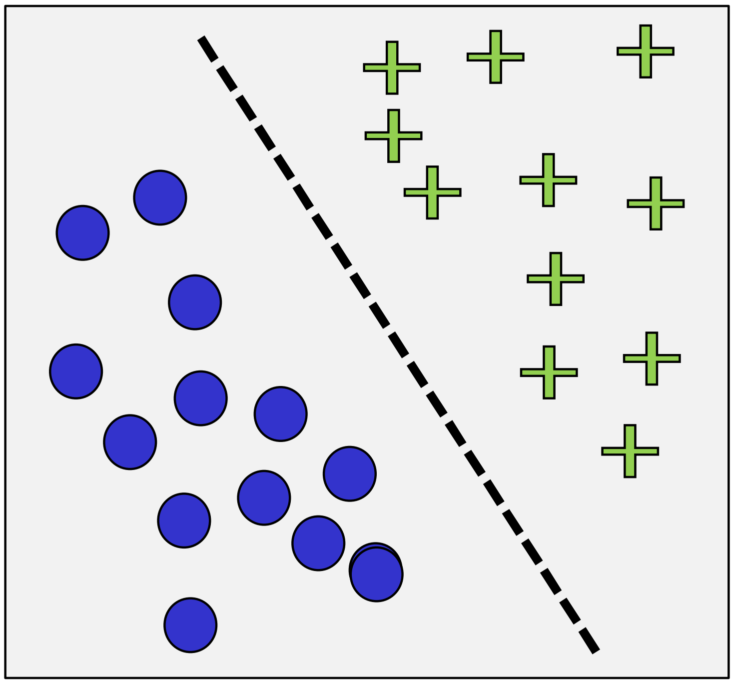
(autoencoders and GANs)

By

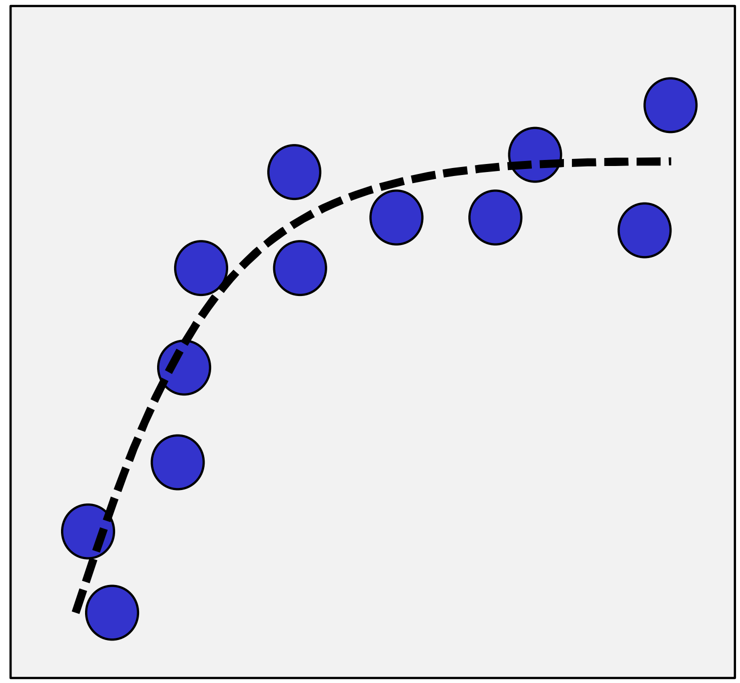
Pierre-Marc Jodoin

So far: supervised learning

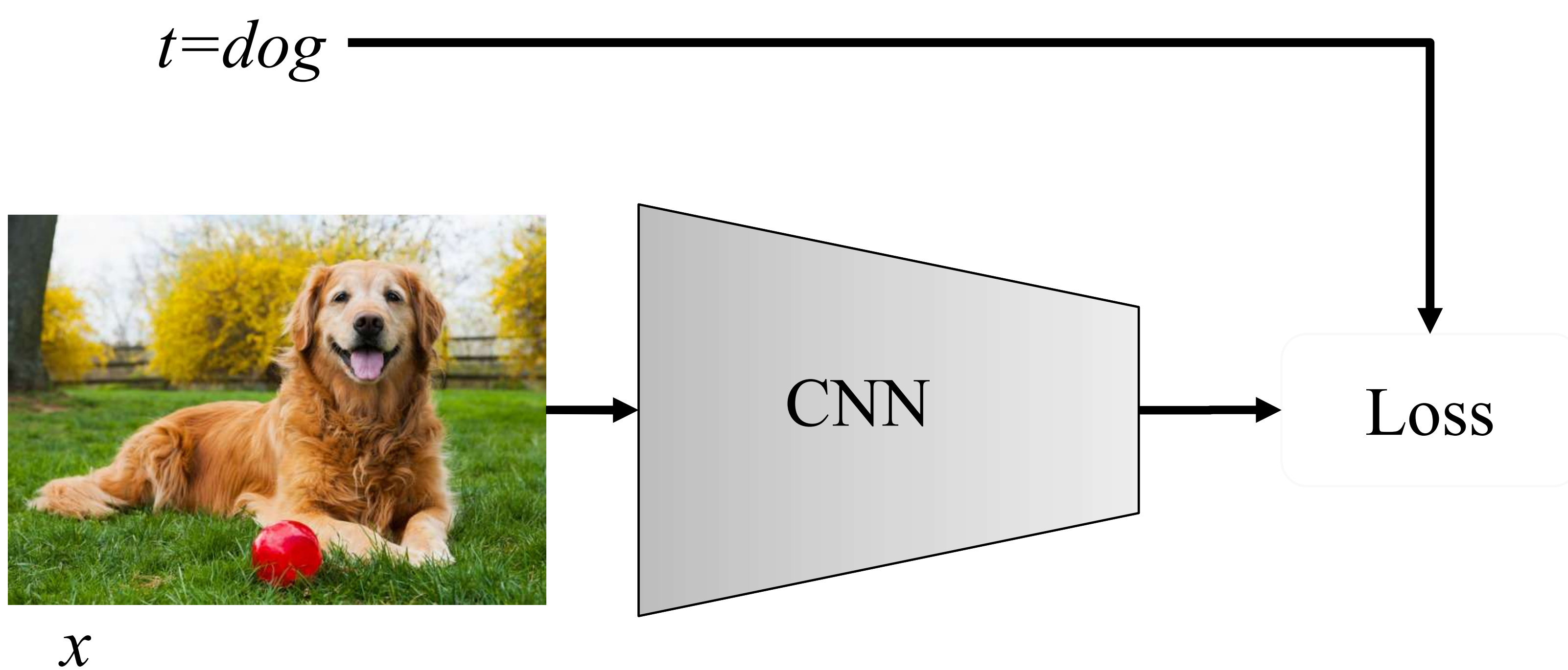
Classification



Régression



Supervised learning with CNN



Supervised vs. Unsupervised

Supervised learning: there is a target

$$D = \left\{ \left(\vec{x}_1, t_1 \right), \left(\vec{x}_2, t_2 \right), \ldots, \left(\vec{x}_N, t_N \right) \right\}$$

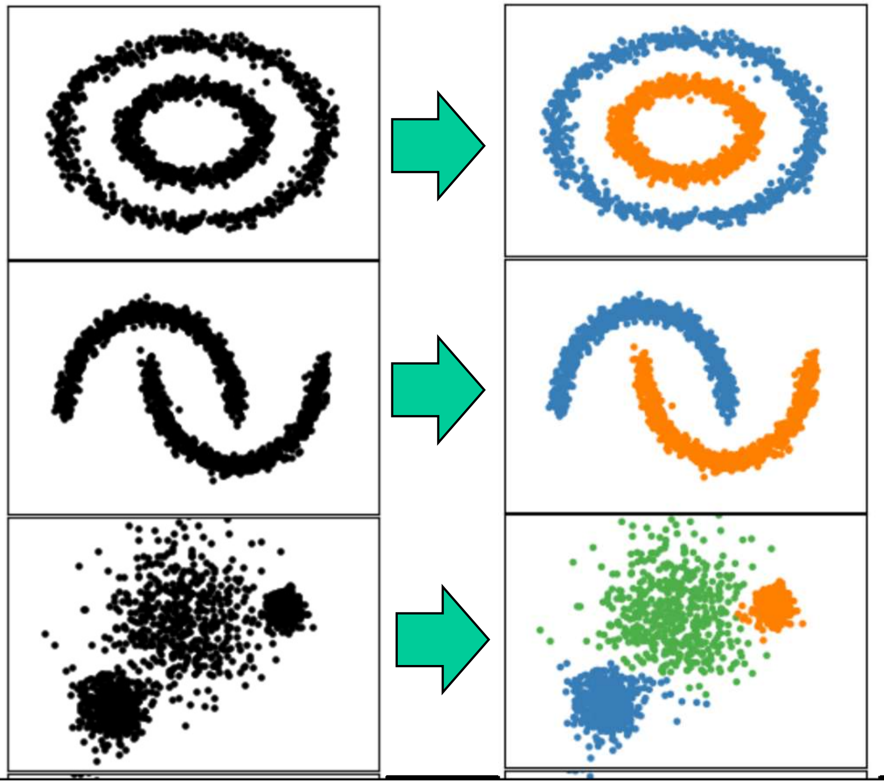
Unsupervised learning: the target is not provided

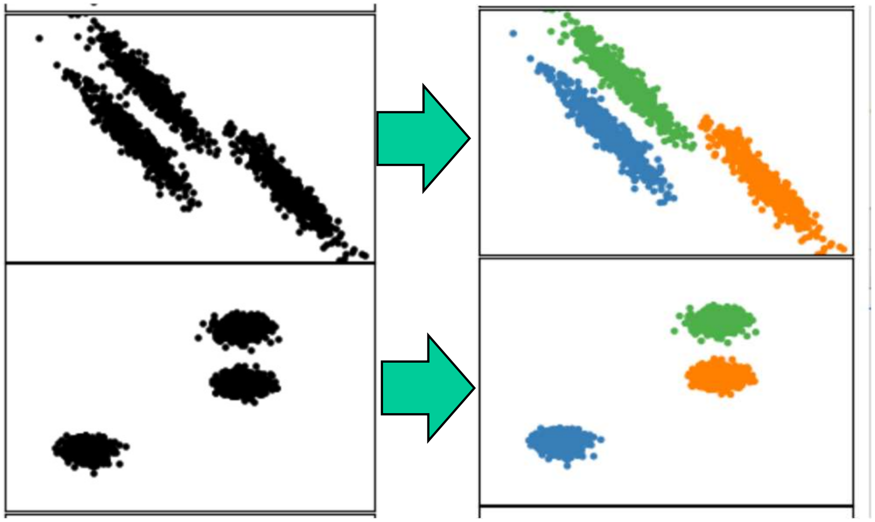
$$D = \left\{ \vec{x}_1, \vec{x}_2, \ldots, \vec{x}_N \right\}$$

Unsupervised learning

Often, unsupervised learning includes one (or more) **latent variables**.

Ex: clustering = find the latent variable "cluster"

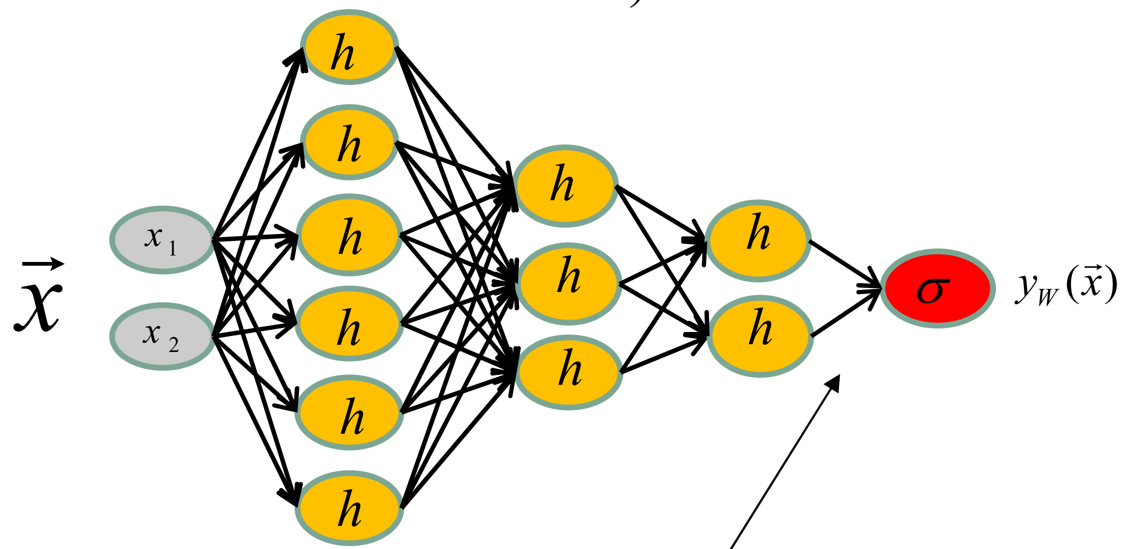


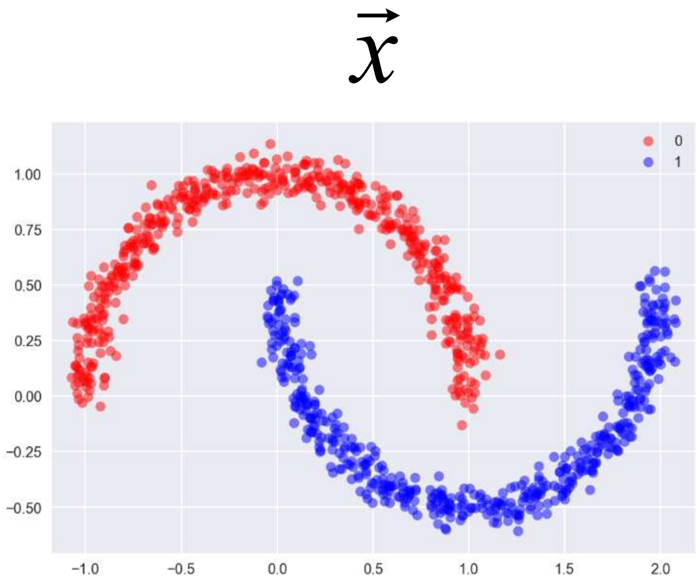


Unsupervised learning by neural networks is based on **2 properties**

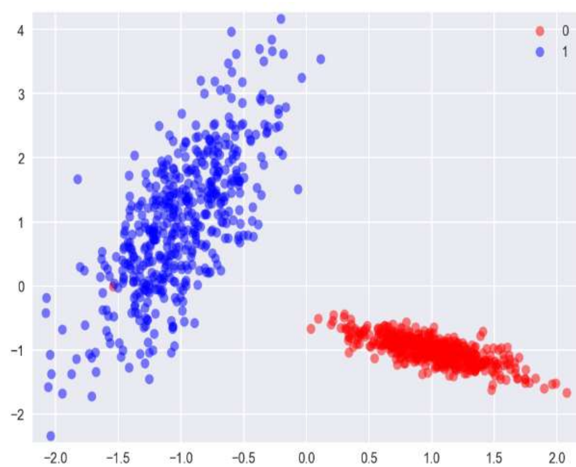
Property 1

Neural networks are excellent machines for projecting raw data to a **lower dimensional space** whose properties depend on the loss (in this case, **linear space** because the output of the network is a linear classifier).

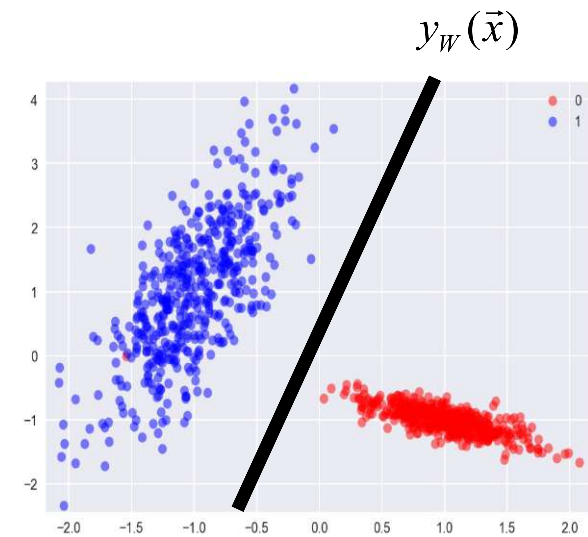




Données en entrée



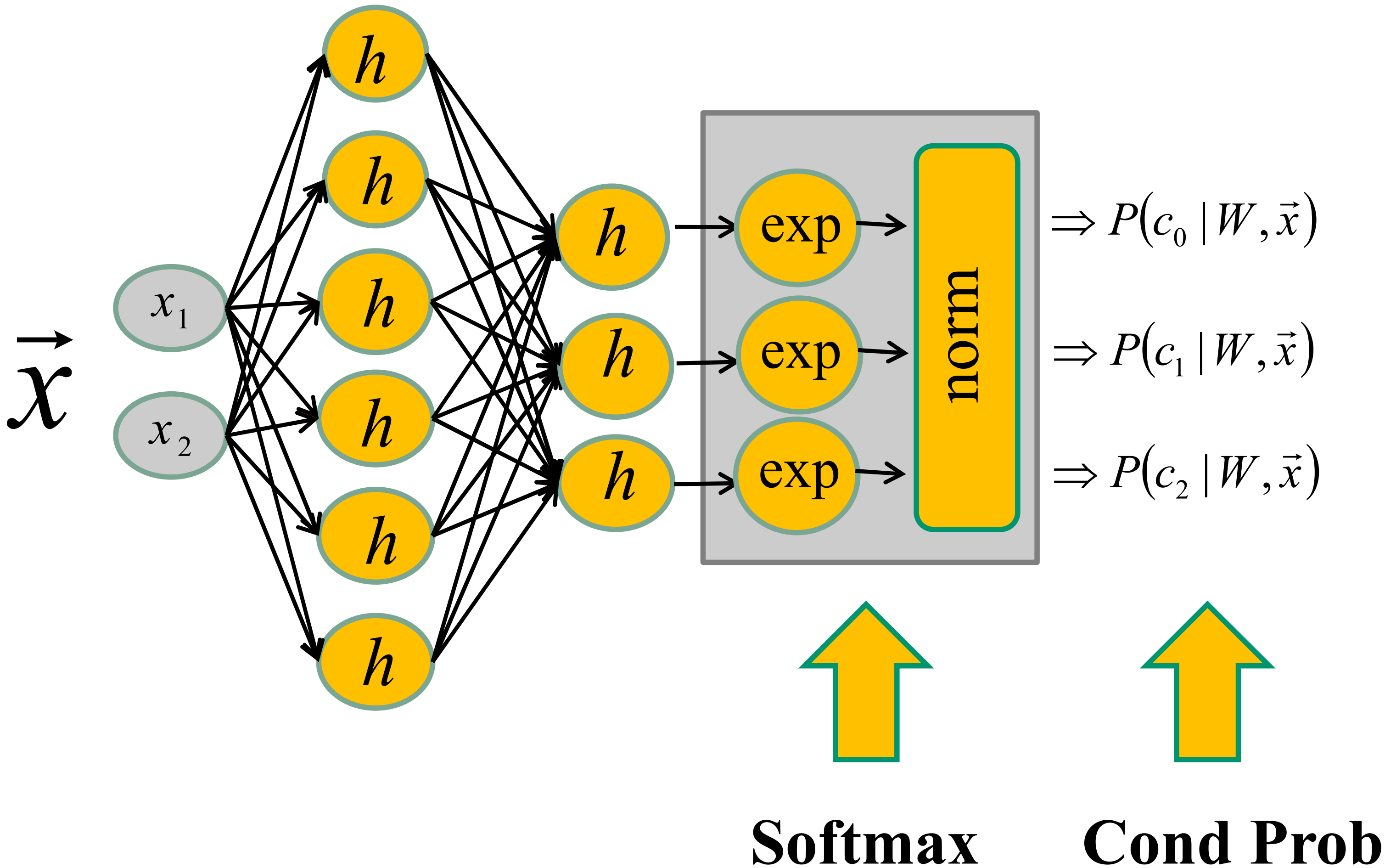
Sortie de la dernière couche



Sortie du réseau

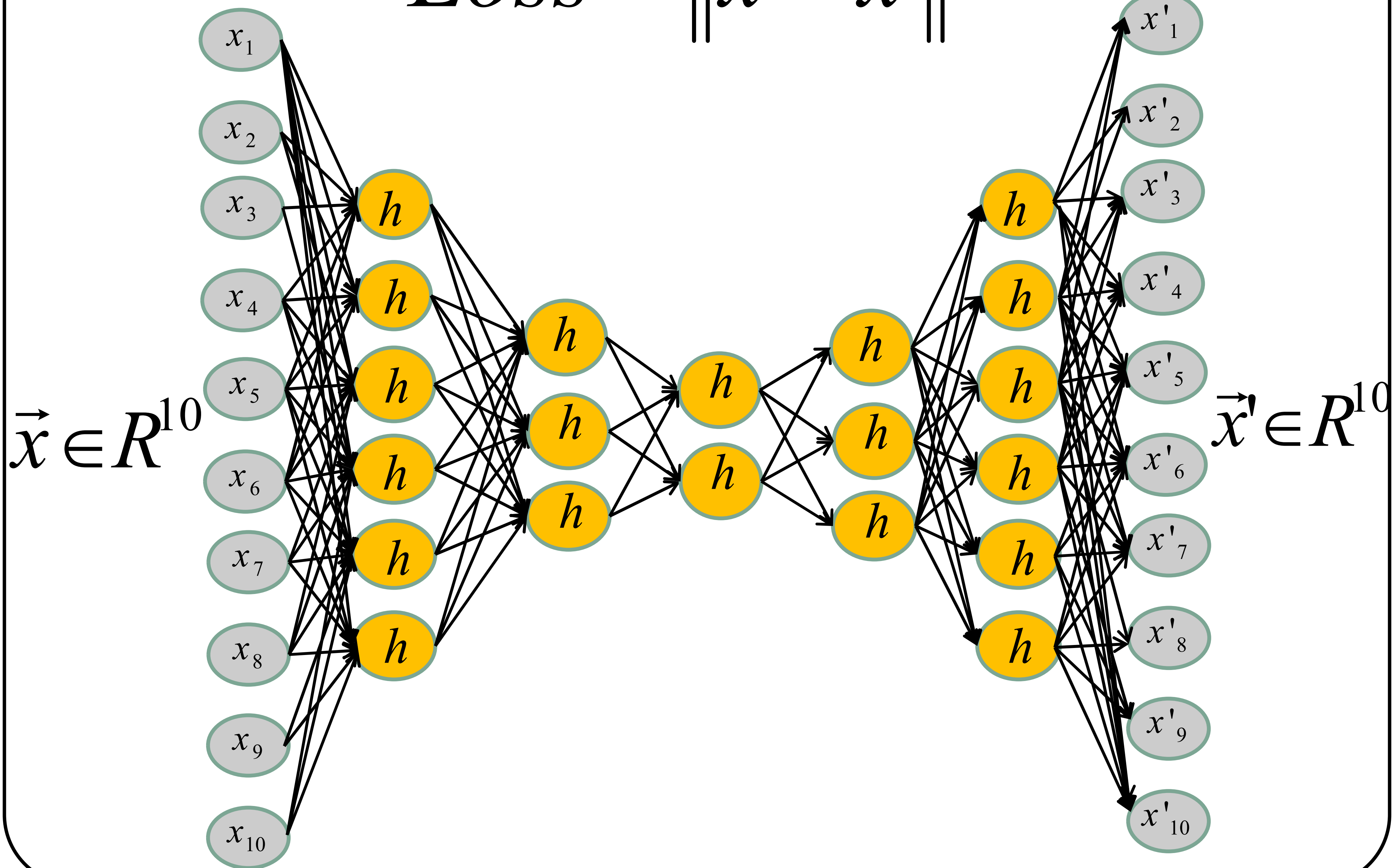
Property 2

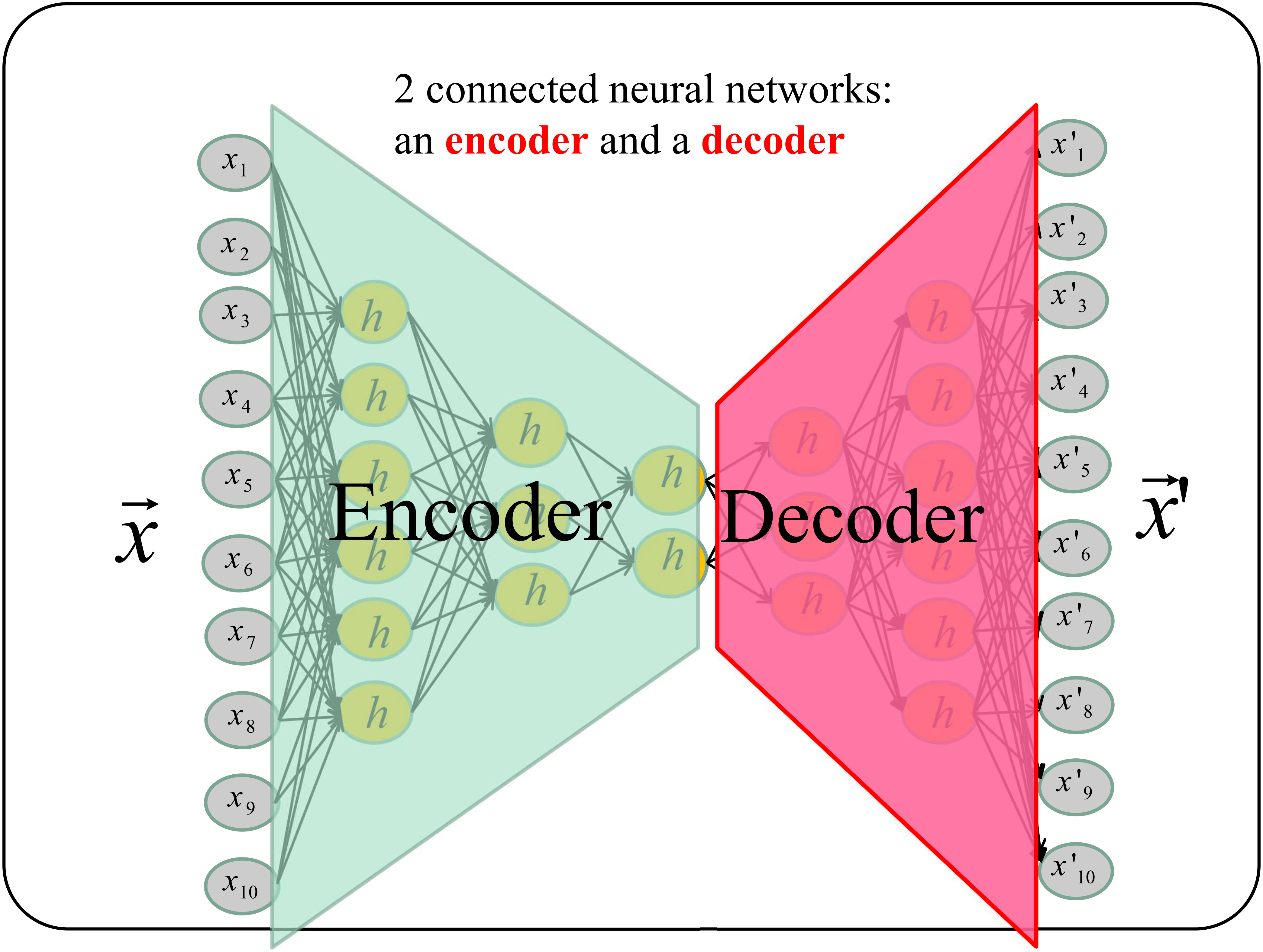
Neural networks are excellent machines for estimating **conditional probabilities**.

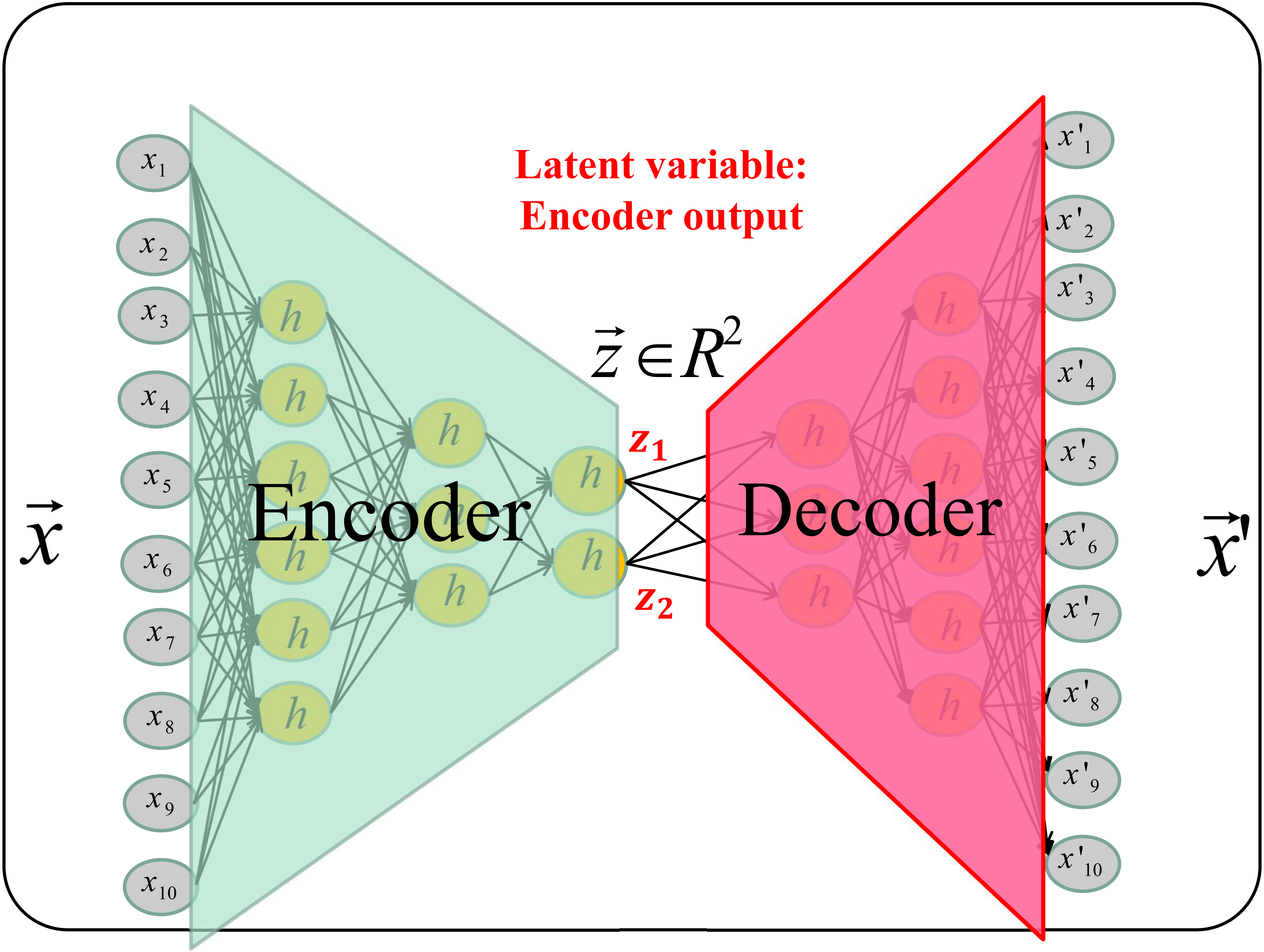


How to use a neural network to learn
the **underlying configuration** of unlabeled data?

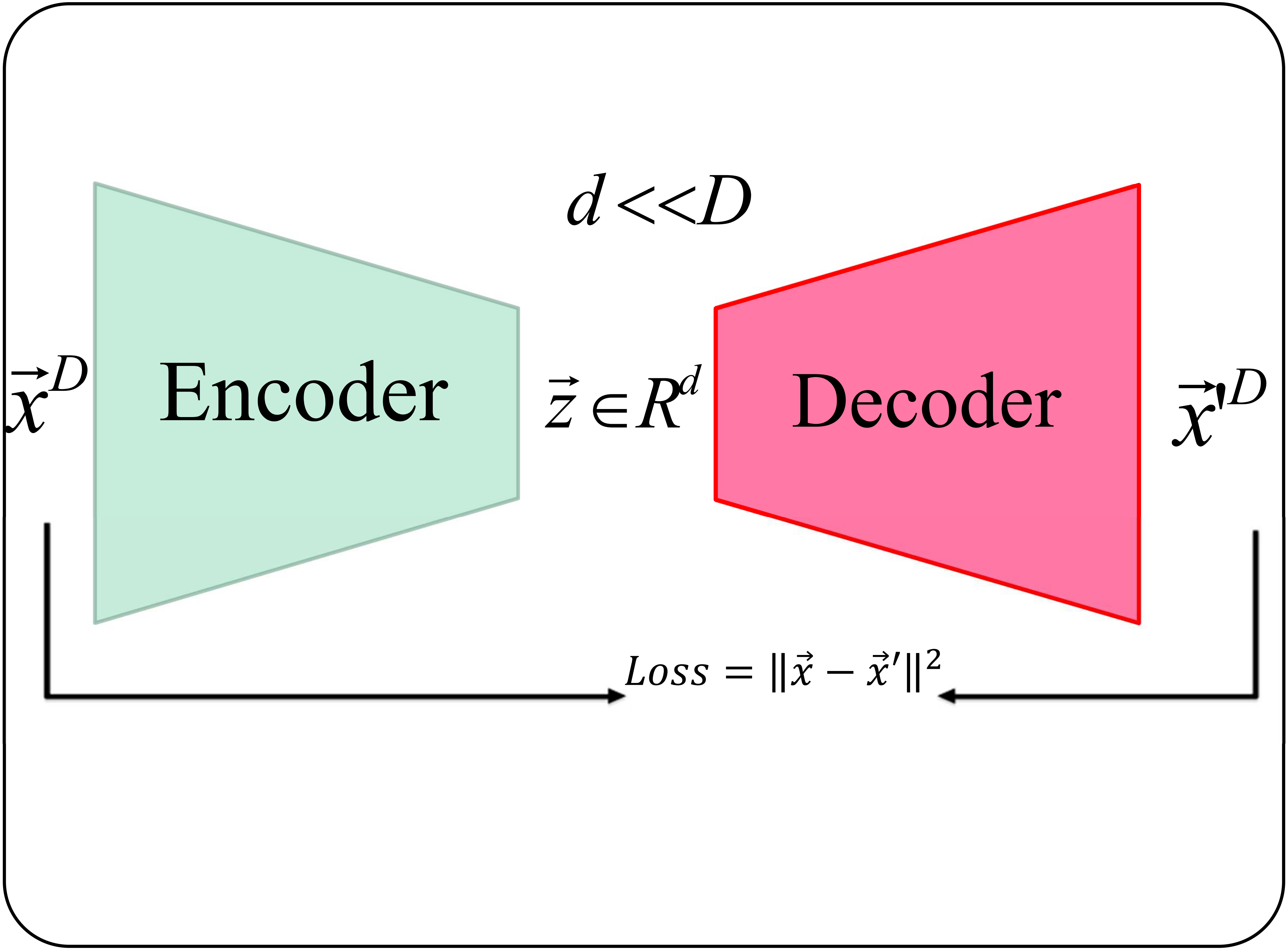
$$Loss = \left\| \vec{x} - \vec{x}' \right\|^2$$



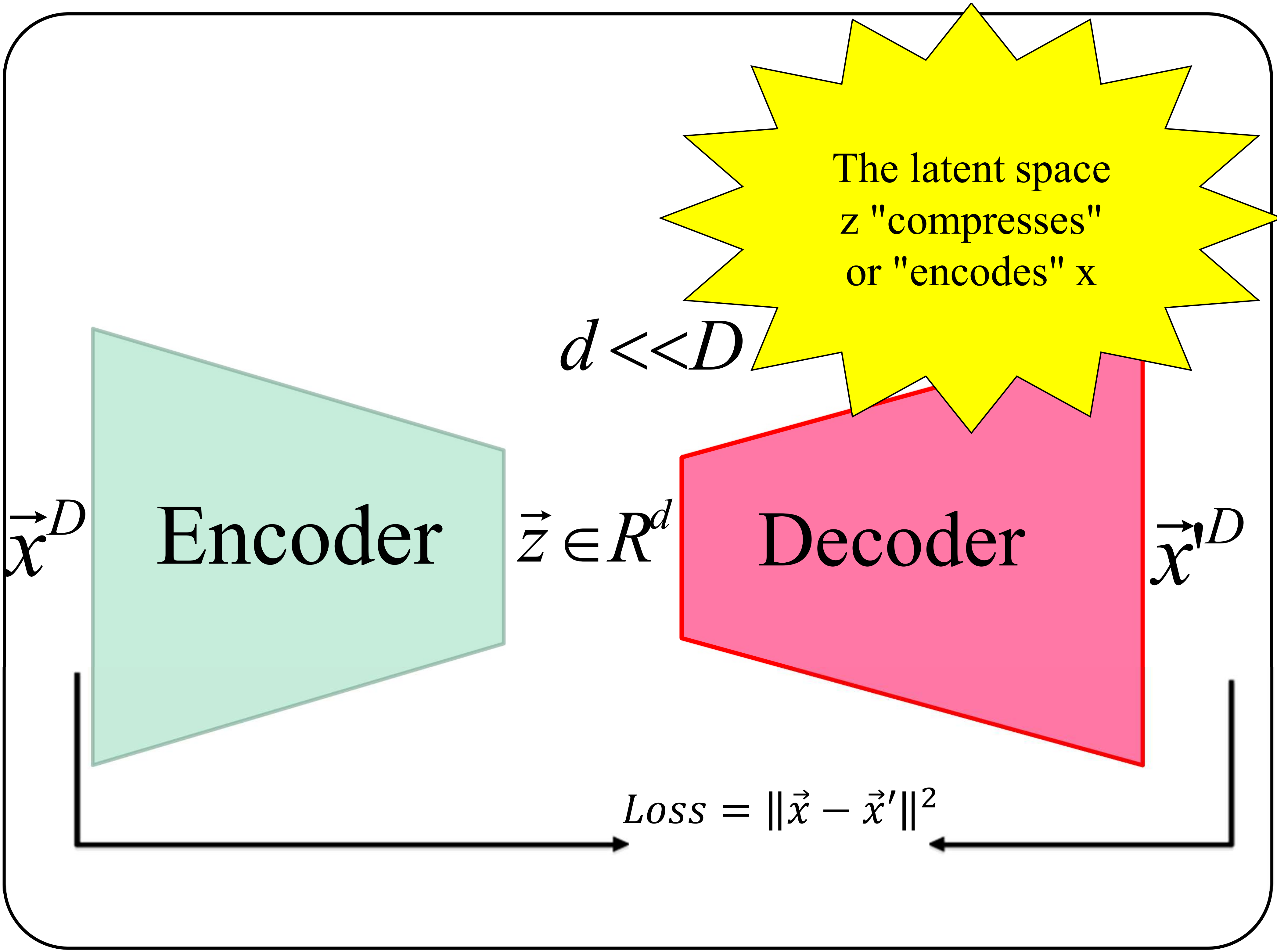

2 connected neural networks:
an **encoder** and a **decoder**
Encoder
Decoder
$\vec{x}$
$\vec{x}'$
x_1
x_2
x_3
x_4
x_5
x_6
x_7
x_8
x_9
x_{10}
h
x'_1
x'_2
x'_3
x'_4
x'_5
x'_6
x'_7
x'_8
x'_9
x'_{10}



Latent variable:
Encoder output
$\vec{z} \in R^2$
z_1
z_2
Encoder
Decoder
$\vec{x}$
$\vec{x}'$
x_1 x_2 x_3 x_4 x_5 x_6 x_7 x_8 x_9 x_{10}
x'_1 x'_2 x'_3 x'_4 x'_5 x'_6 x'_7 x'_8 x'_9 x'_{10}
h

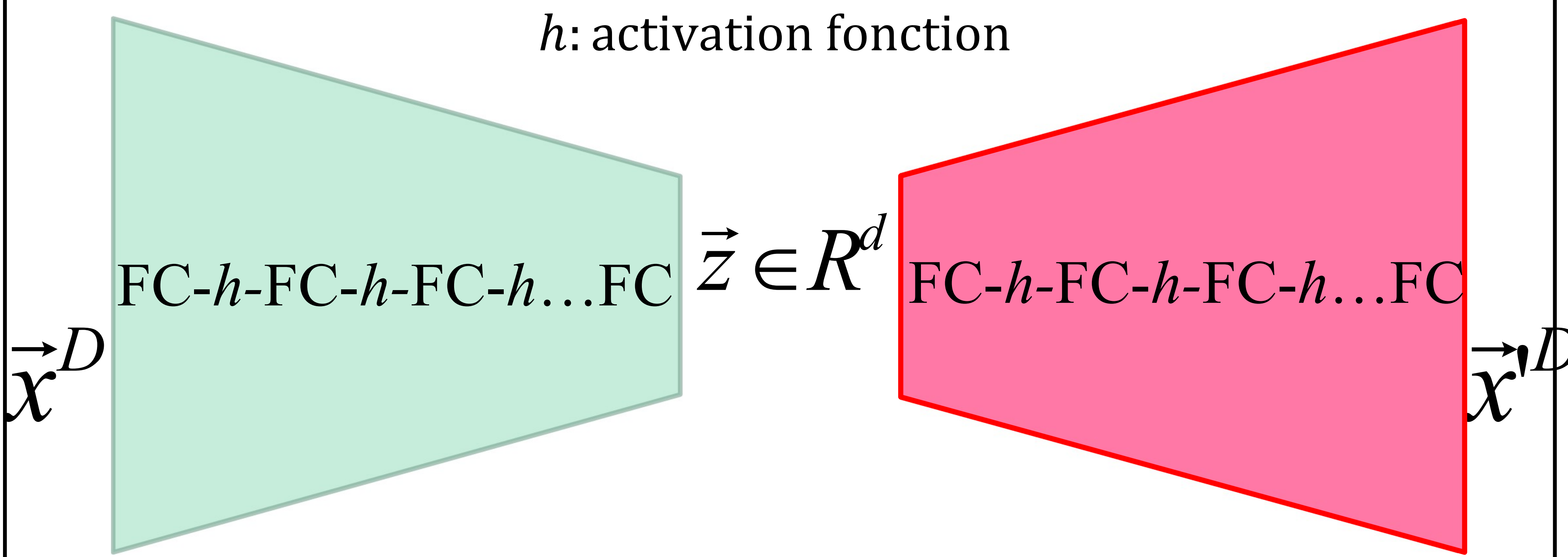


$d << D$
$\vec{x}^D$
Encoder
$\vec{z} \in R^d$
Decoder
$\vec{x}'^D$
$Loss = \|\vec{x} - \vec{x}'\|^2$

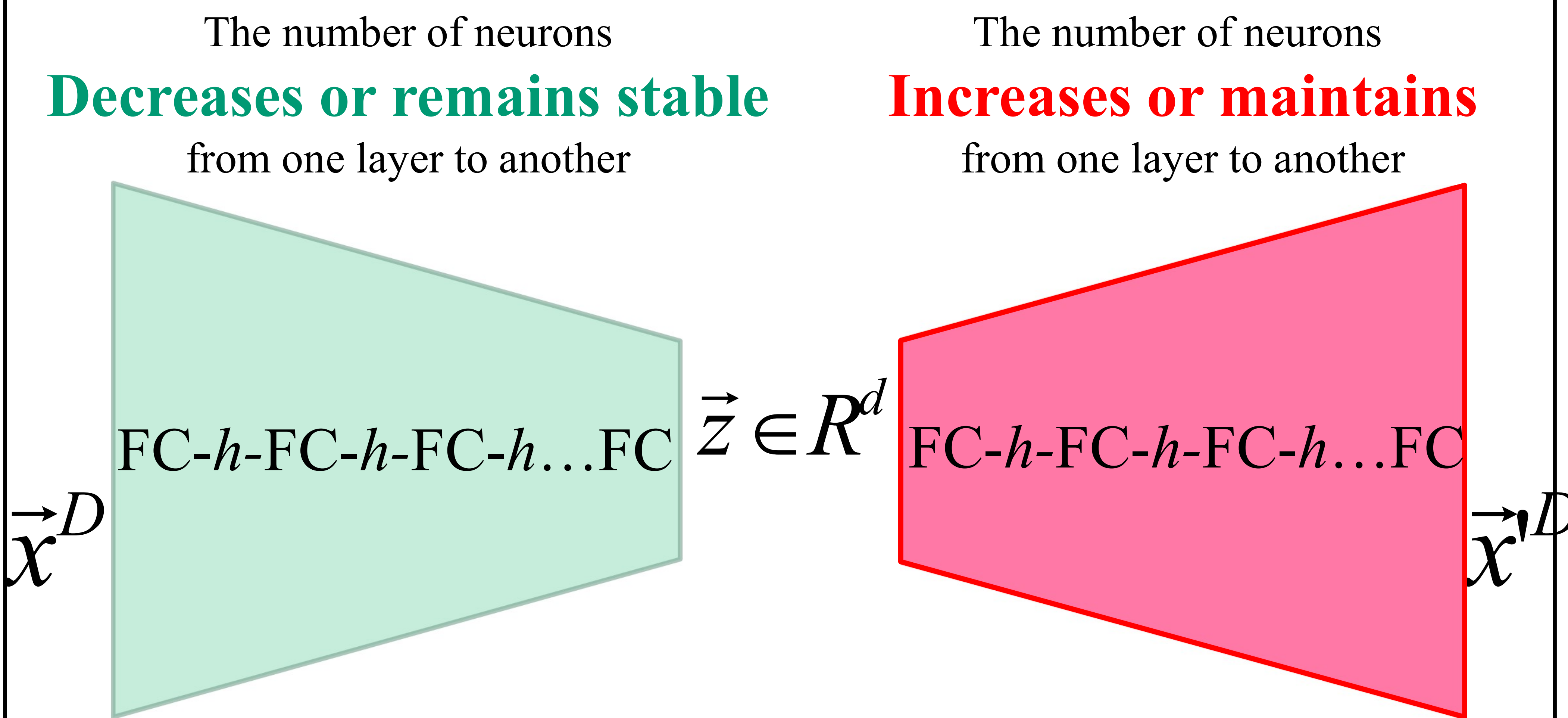


The latent space
z "compresses"
or "encodes" x
$d << D$
$\vec{x}^D$
Encoder
$\vec{z} \in R^d$
Decoder
$\vec{x}'^D$
$Loss = \|\vec{x} - \vec{x}'\|^2$

Fully connected layers



Generally...



The number of neurons
Decreases or remains stable
from one layer to another
The number of neurons
Increases or maintains
from one layer to another
FC-*h*-FC-*h*-FC-*h*…FC
$\vec{z} \in R^d$
FC-*h*-FC-*h*-FC-*h*…FC
$\vec{x}^D$
$\vec{x}'^D$

Toy MNIST autoencoder

```python
class autoencoder(nn.Module):
    def __init__(self):
        super(autoencoder, self).__init__()
        self.encoder = nn.Sequential(
            nn.Linear(28 * 28, 128), nn.ReLU(True),
            nn.Linear(128, 64), nn.ReLU(True),
            nn.Linear(64, 12), nn.ReLU(True),
            nn.Linear(12, 2))
        self.decoder = nn.Sequential(
            nn.Linear(2, 12), nn.ReLU(True),
            nn.Linear(12, 64), nn.ReLU(True),
            nn.Linear(64, 128), nn.ReLU(True),
            nn.Linear(128, 28 * 28))

    def forward(self, x):
        z = self.encoder(x)
        x_prime = self.decoder(z)
        return x_prime
```

2D latent space

Toy MNIST autoencoder

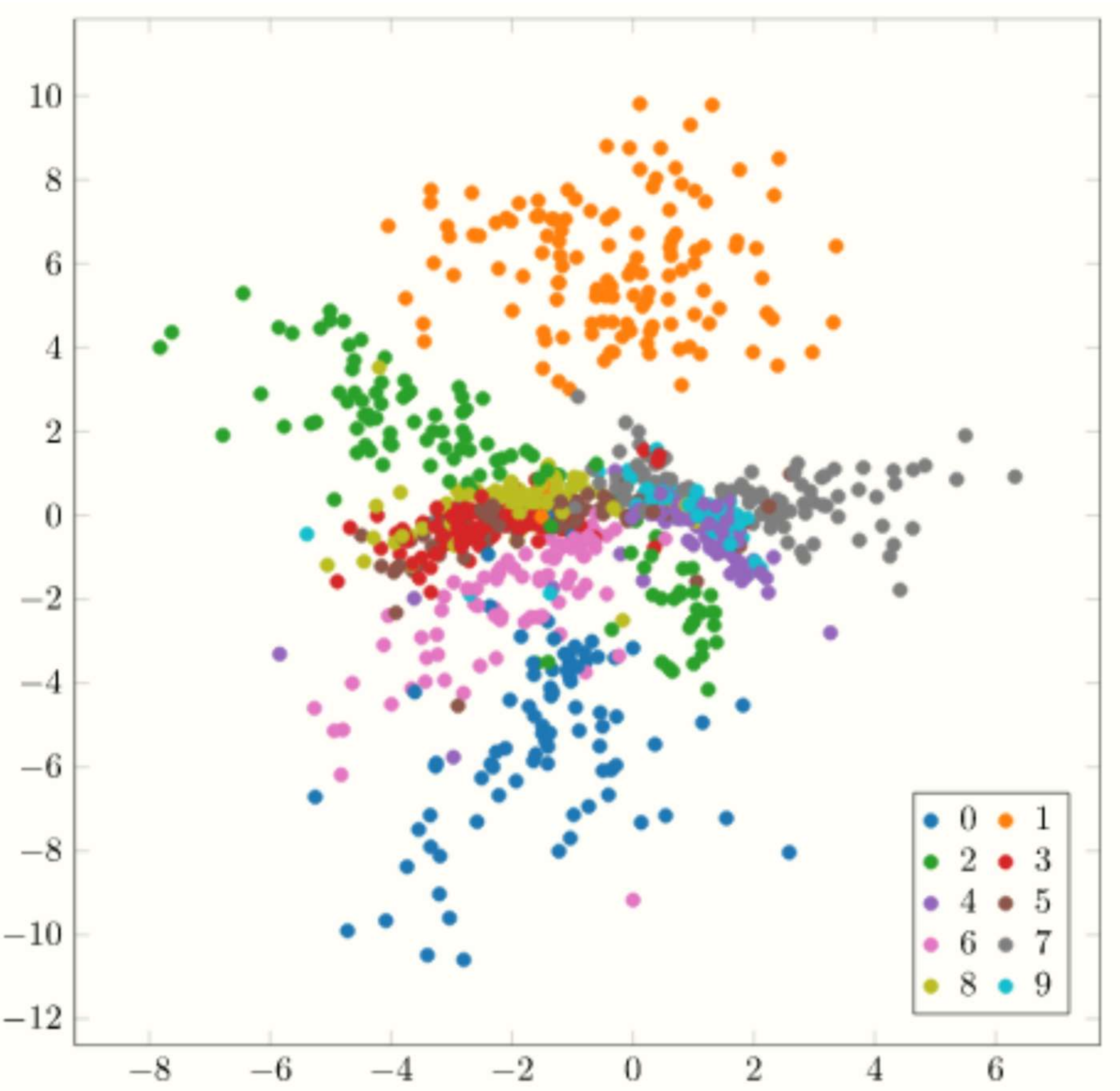
symmetry

```
class autoencoder(nn.Module):
    def __init__(self):
        super(autoencoder, self).__init__()
        self.encoder = nn.Sequential(
            nn.Linear(28 * 28, 128), nn.ReLU(True),
            nn.Linear(128, 64), nn.ReLU(True),
            nn.Linear(64, 12), nn.ReLU(True),
            nn.Linear(12, 2))
        self.decoder = nn.Sequential(
            nn.Linear(2, 12), nn.ReLU(True),
            nn.Linear(12, 64), nn.ReLU(True),
            nn.Linear(64, 128), nn.ReLU(True),
            nn.Linear(128, 28 * 28))

    def forward(self, x):
        z = self.encoder(x)
        x_prime = self.decoder(z)
        return x_prime
```

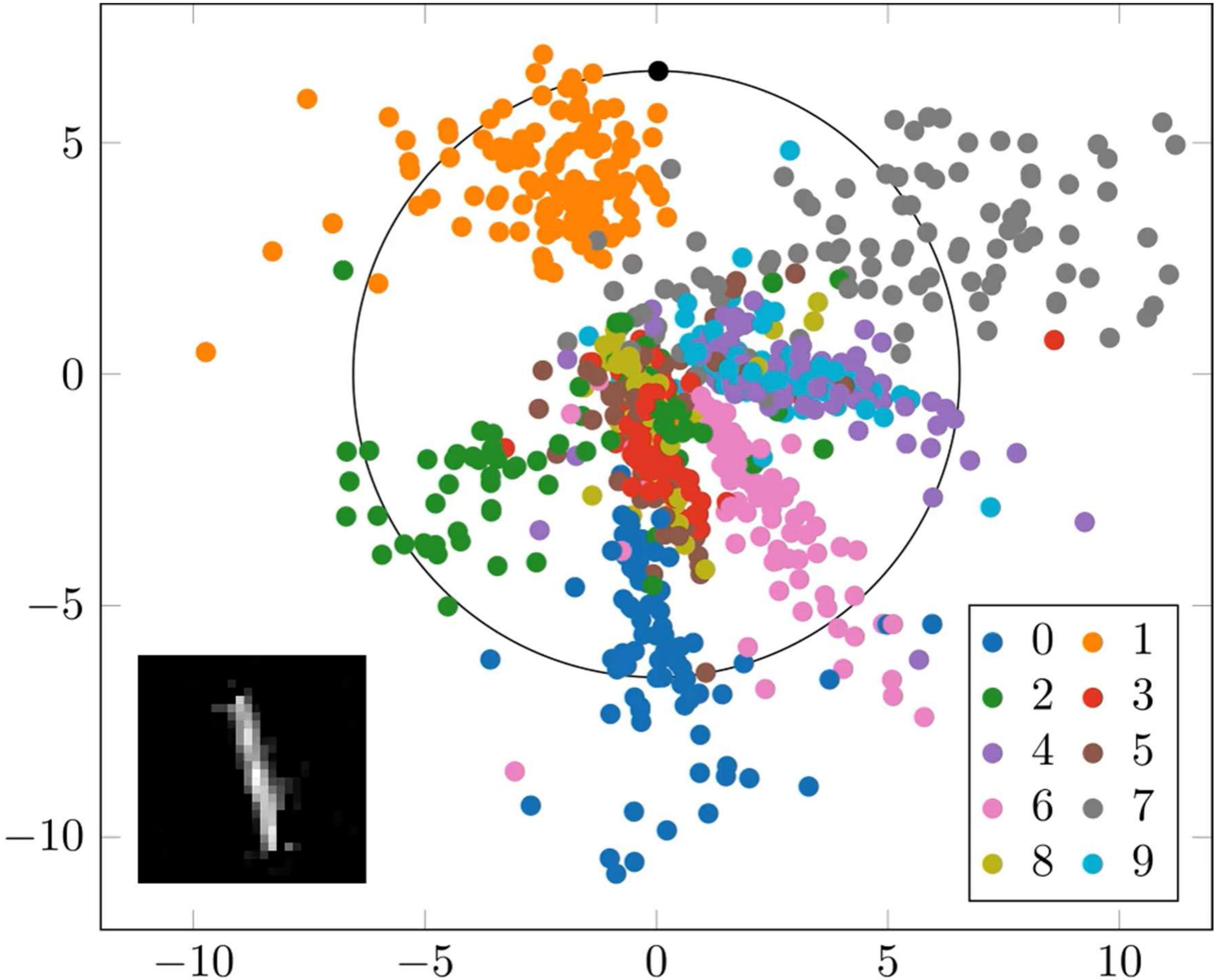
Latent Space Visualization for 1000 MNIST Images

each image corresponds to 1 2D point



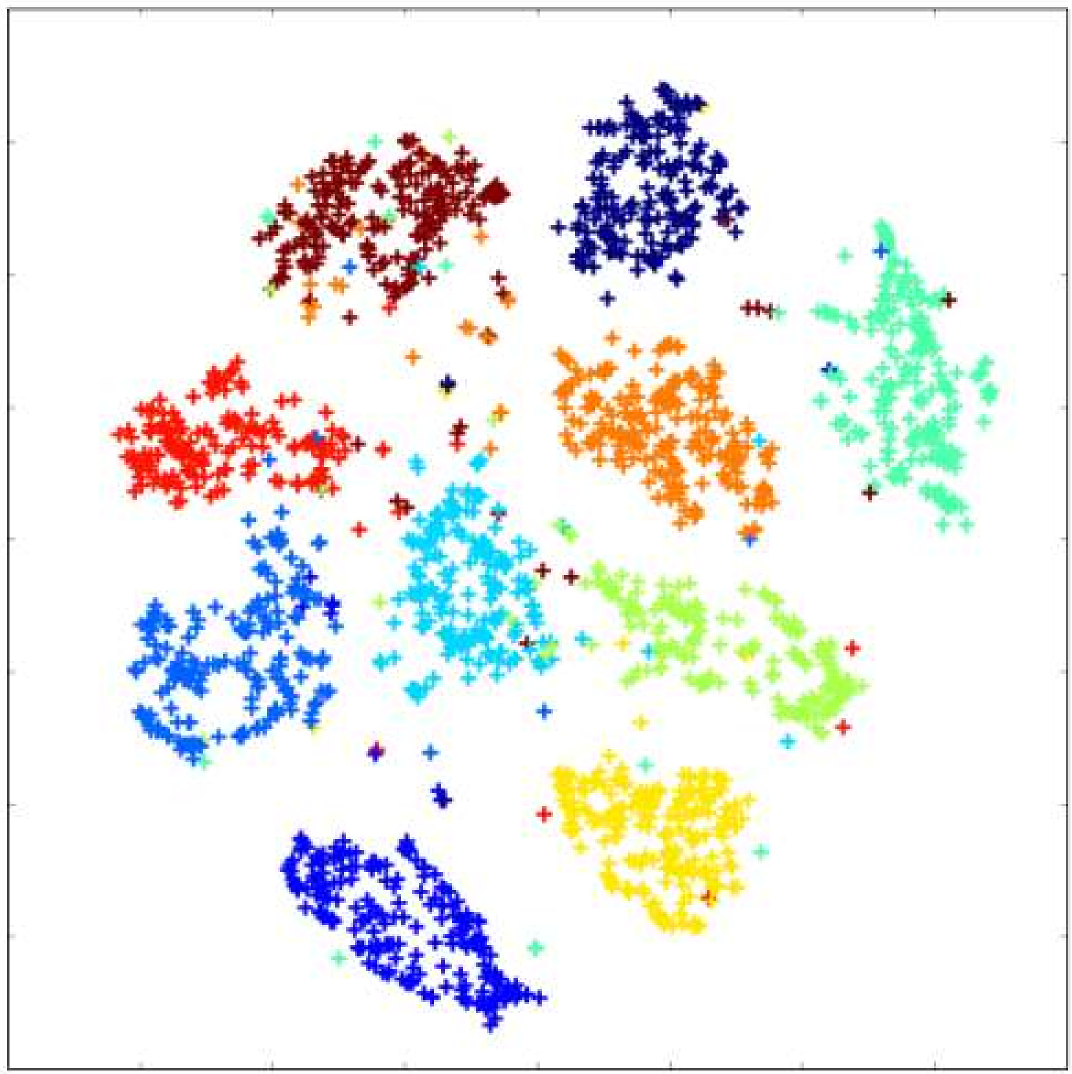
https://gertjanvandenburg.com/blog/autoencoder/

Generate images with the decoder.



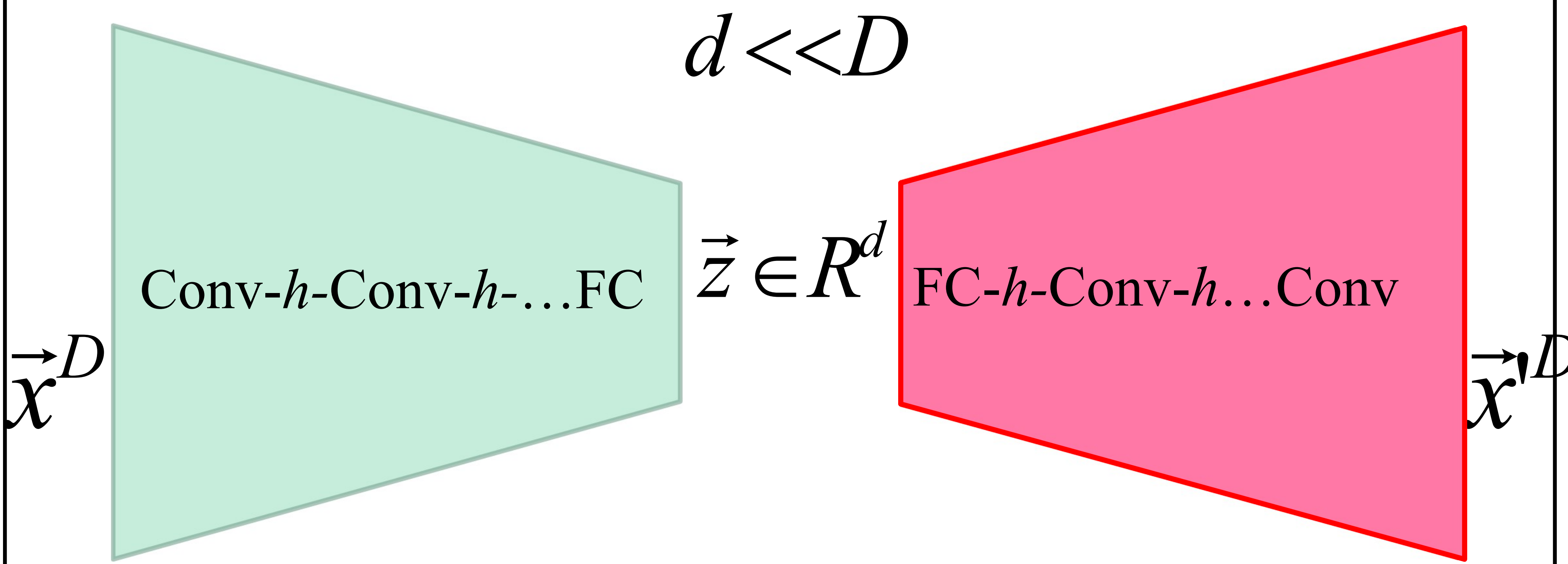
https://gertjanvandenburg.com/blog/autoencoder/

Advanced autoencoders

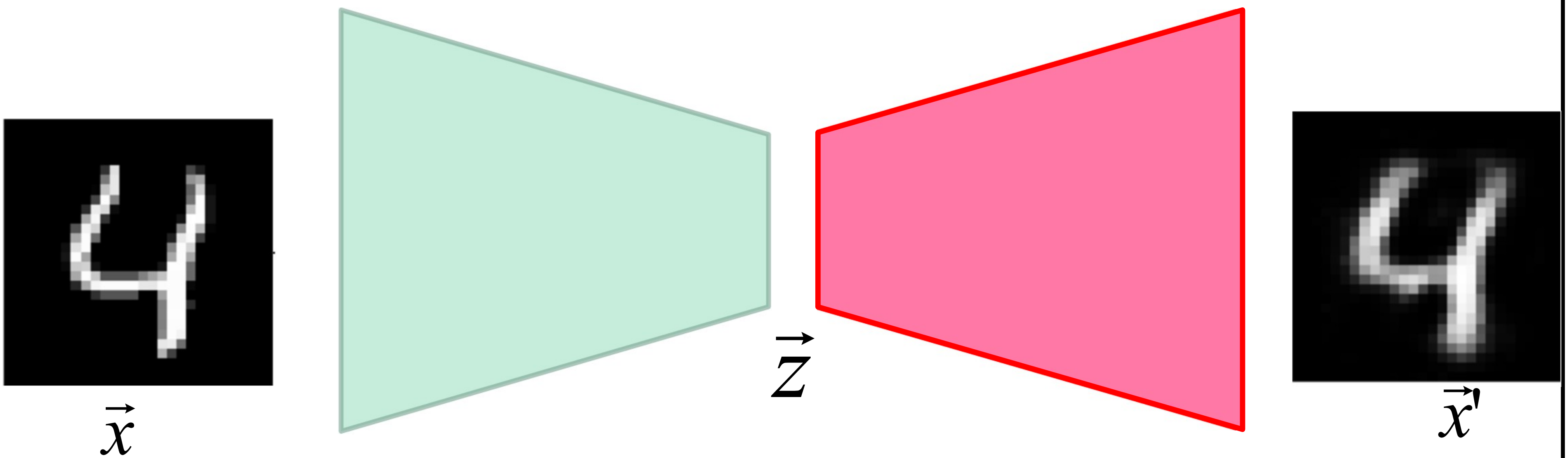


www.ijcai.org/proceedings/2017/0273.pdf

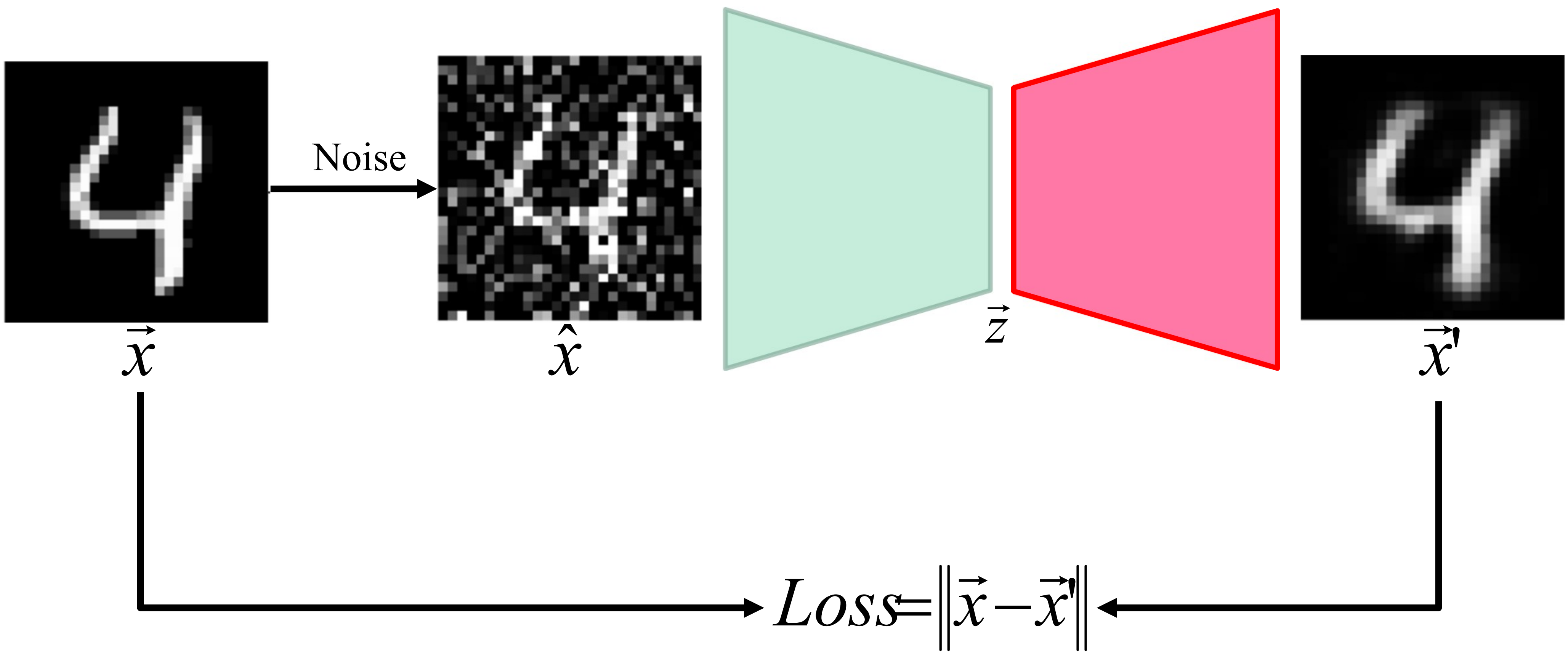
Convolutional layers



Basic autoencoder



Autoencoder for **denoising**

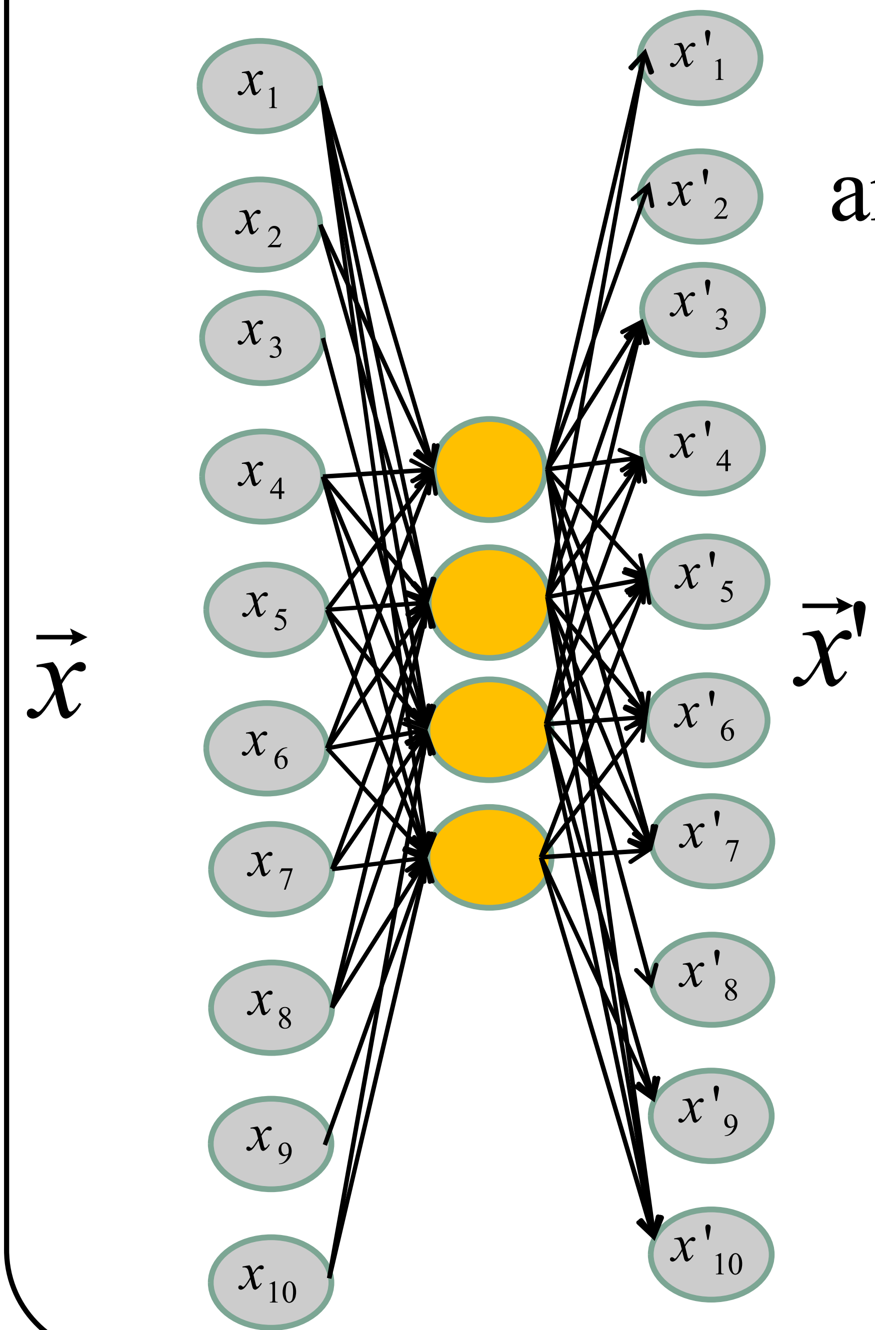


Once trained...

Noisy signal input

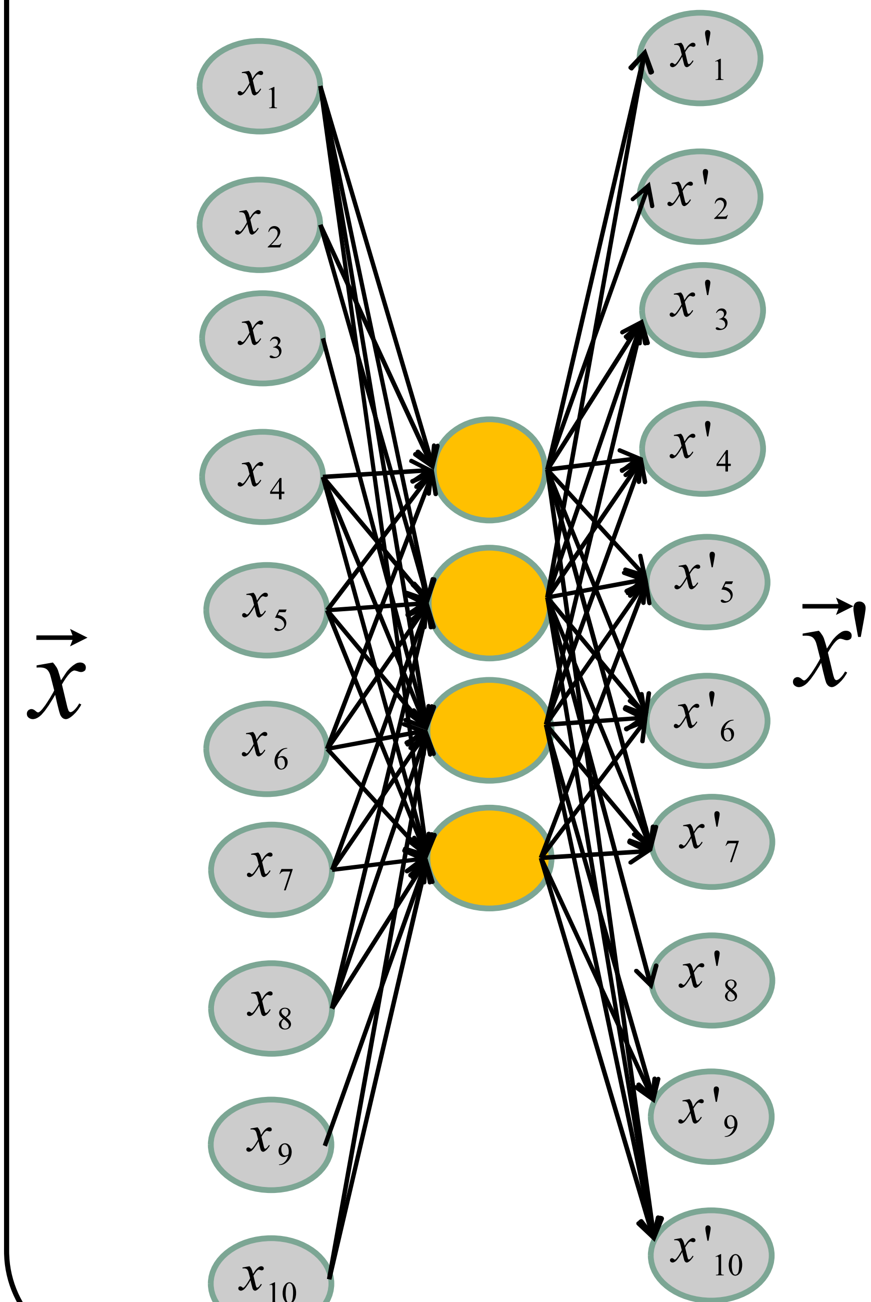
Reconstructed and denoise signal

Single-layer autoencoder
and **without activation function**



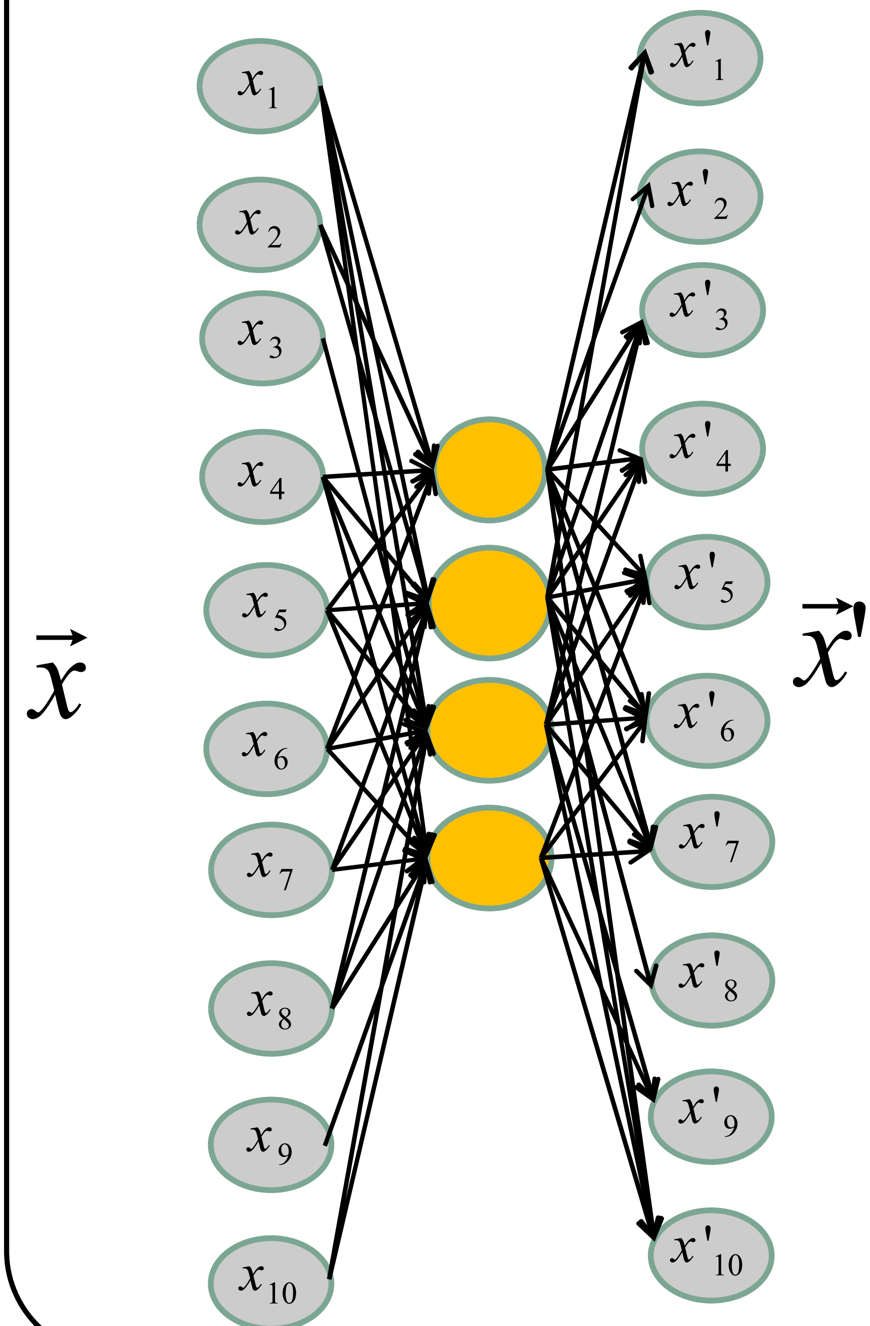
$$W_1 \in R^{4\times 10}$$

$$W_2 \in R^{10\times 4}$$

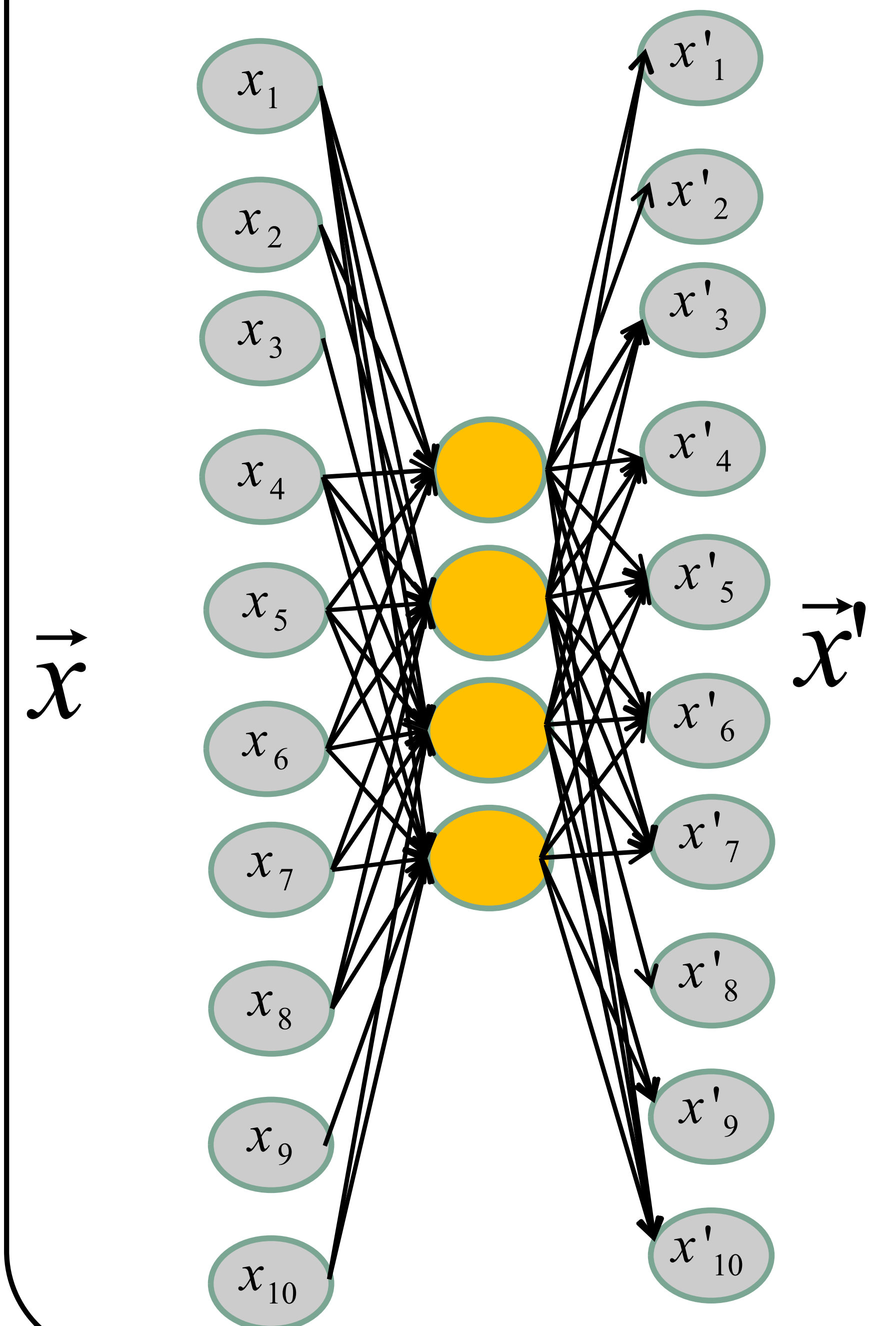


$$\vec{z} = W_1\vec{x}$$

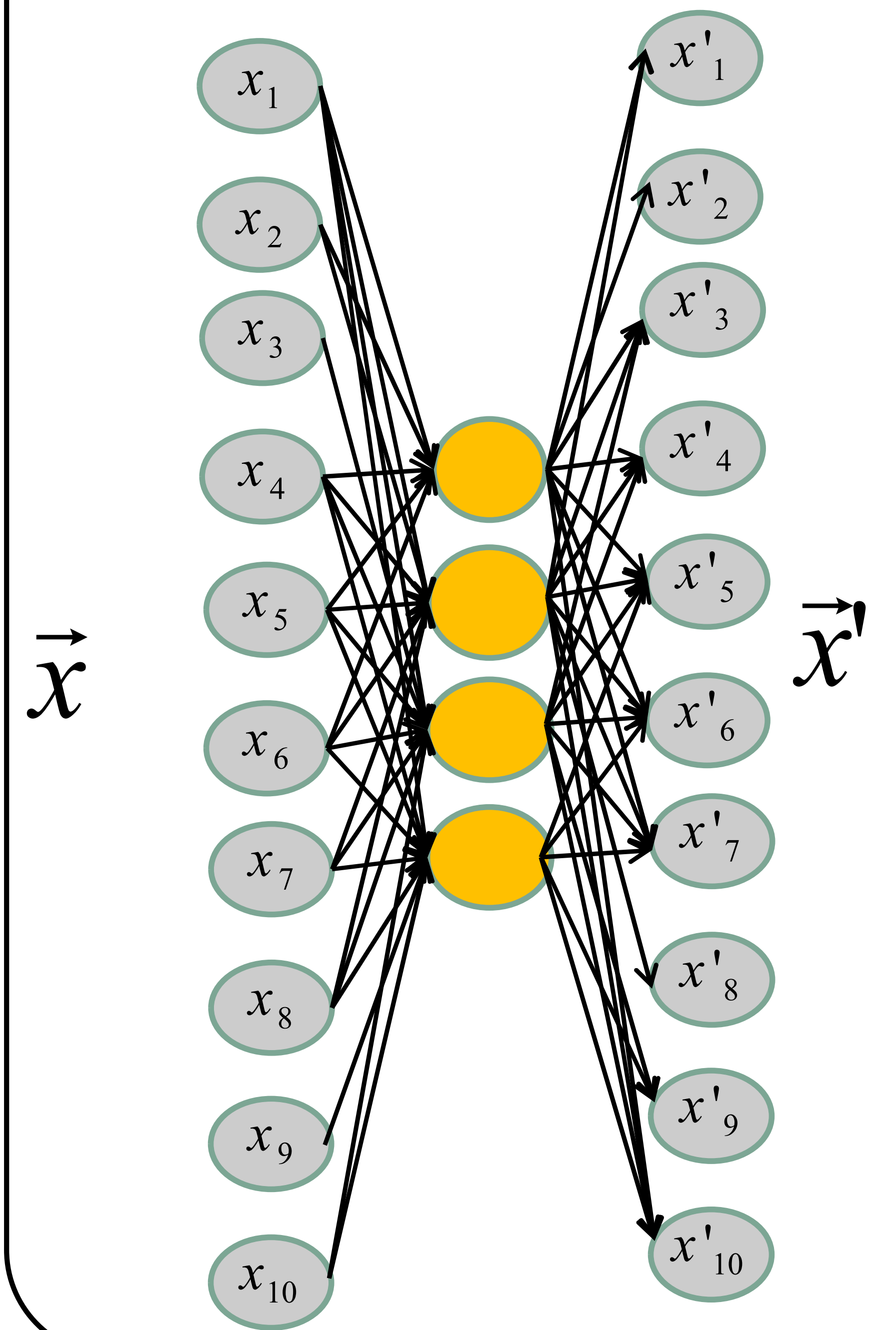
$$\vec{x}' = W_2\vec{z}$$



$$\left.\begin{array}{l} \vec{z} = W_1 \vec{x} \\ \vec{x}' = W_2 \vec{z} \end{array}\right\} \vec{x}' = W_2 W_1 \vec{x}$$

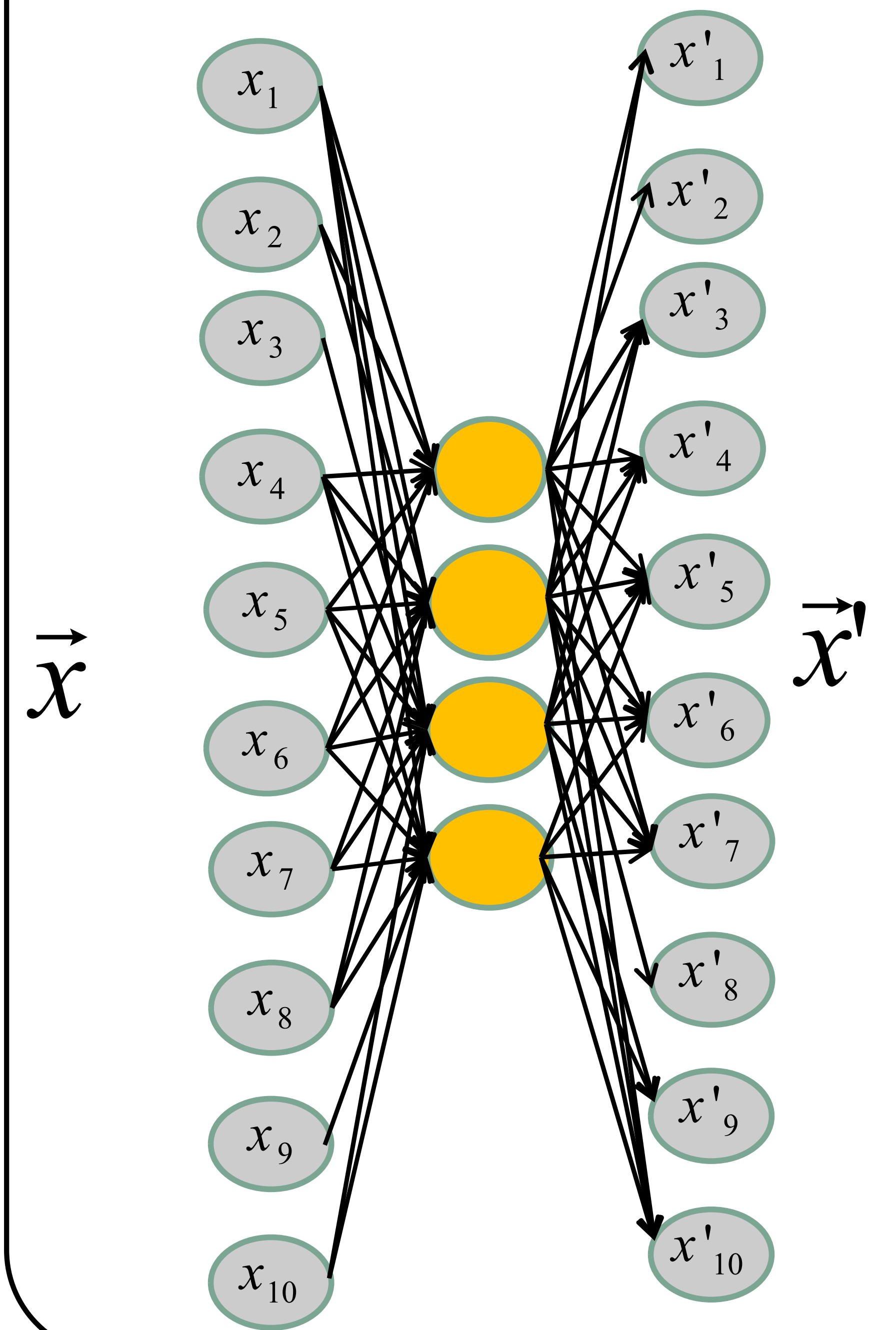


$$Loss = \left\| \vec{x} - W_2 W_1 \vec{x} \right\|^2$$



It can be shown that the solution is

$$W_1 = \left(W_2^T W_2\right)^{-1} W_2^T$$

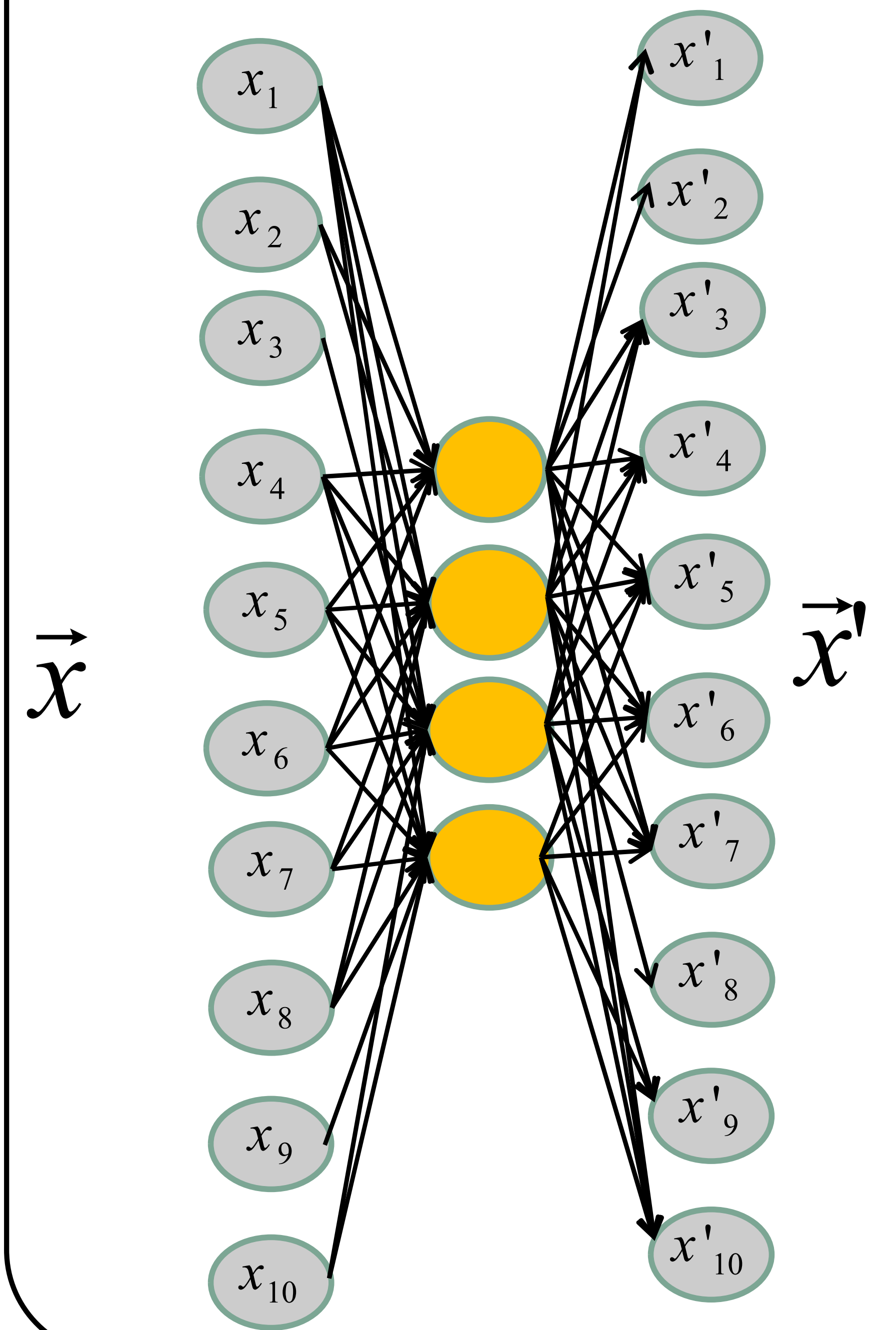


It can be shown that the solution is

$$W_1$$

Matrix consisting of the principals
eigenvectors
of the
variance-covariance matrix
data

SO…



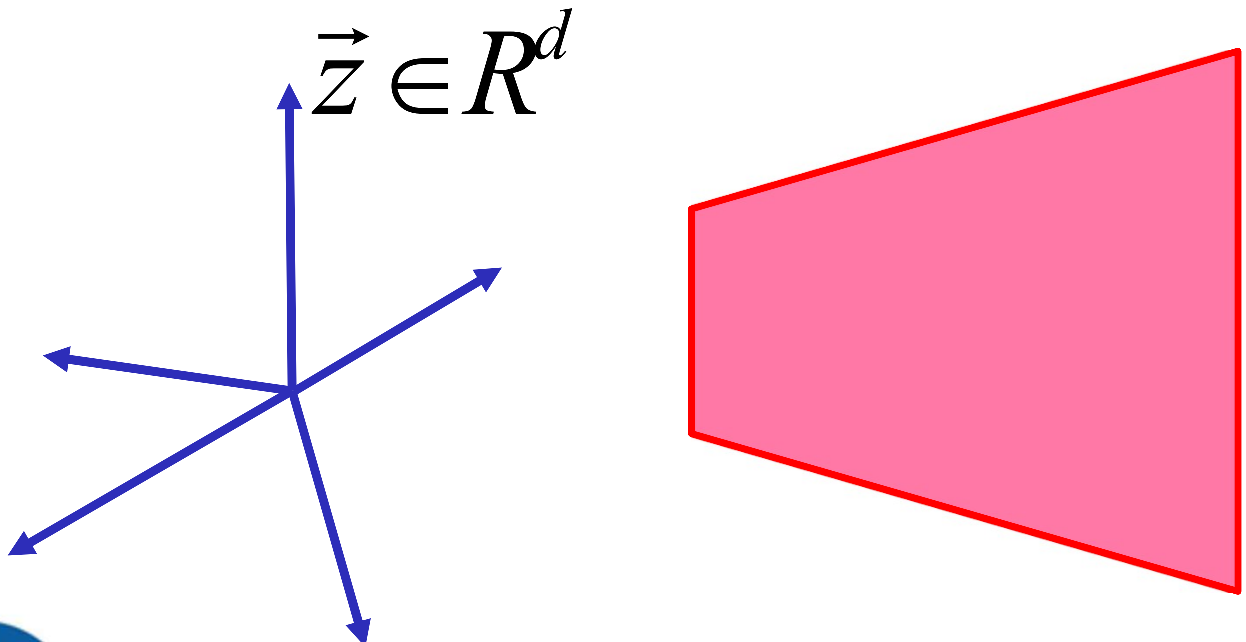
Linear autoencoder

=

Principal Component Analysis (PCA)

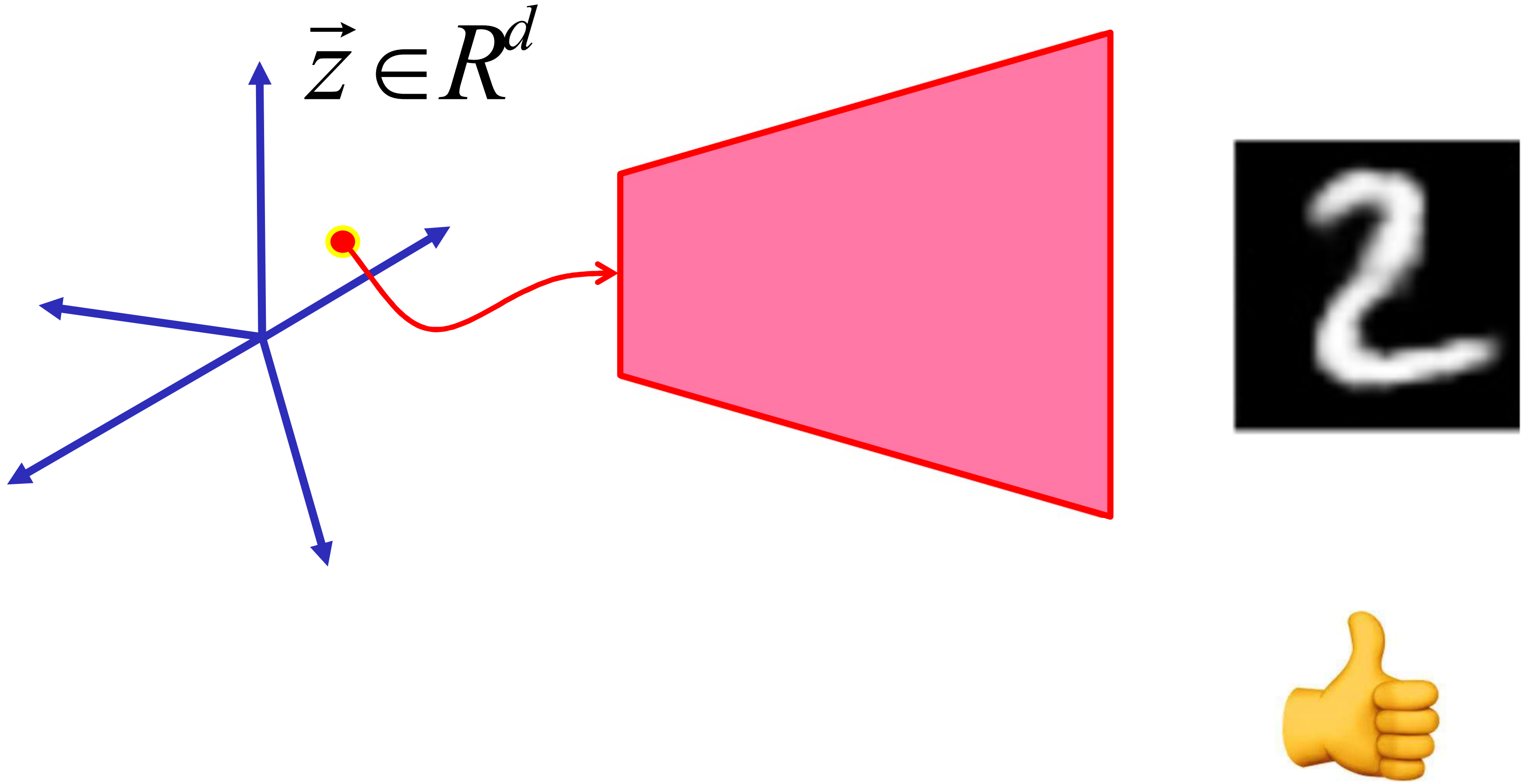
E. Plaut, **From Principal Subspaces to Principal Components with Linear Autoencoders,** arXiv:1804.10253v3, 2018

In general, latent space has between 16 and 128 dimensions.
It can sometimes be more, sometimes less.

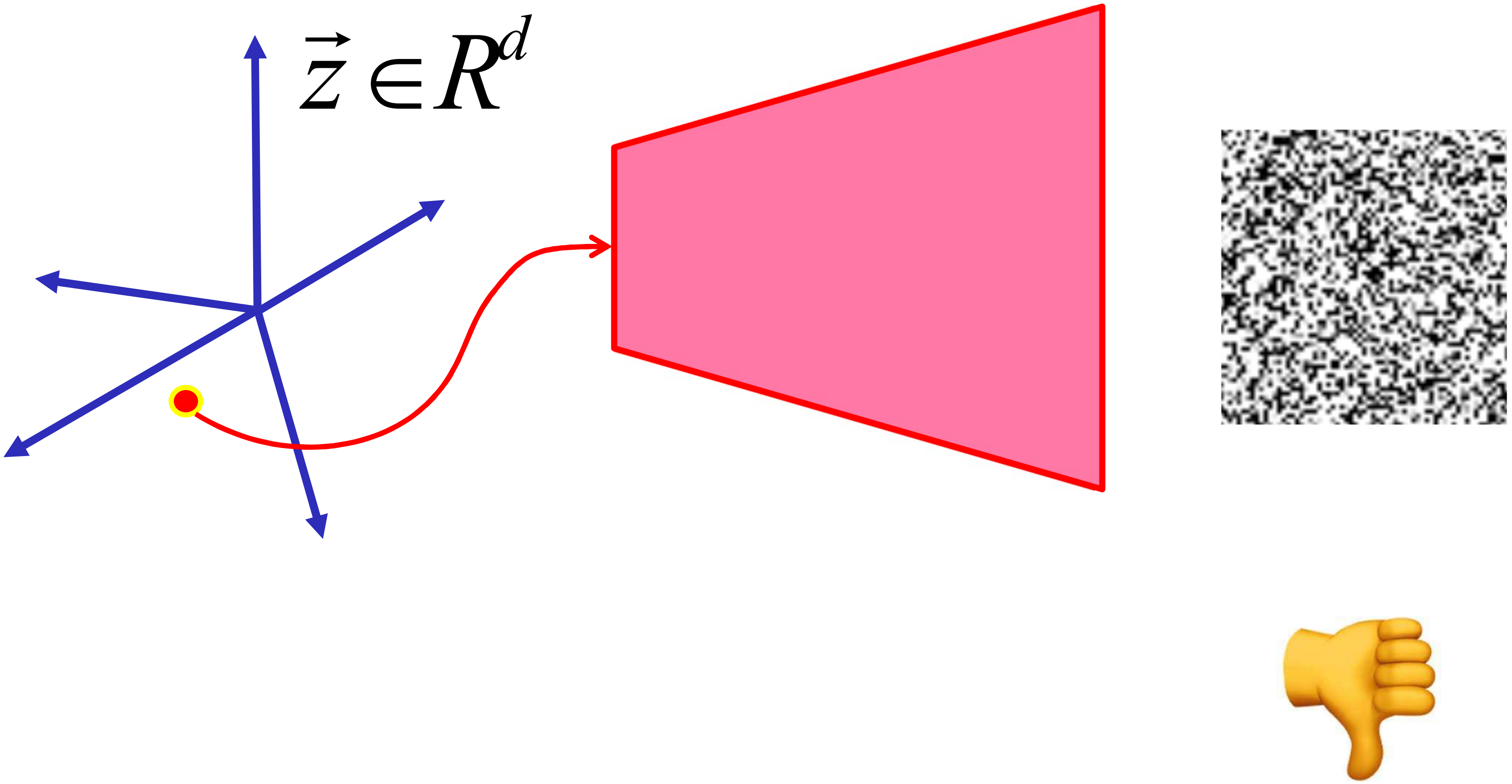


How can one find in this infinite space,
a point $\vec{z}$ that will give a valid image once
Decoded?

With luck, one can select a point at random and reproduce a "good" image (here a "MNIST" image)

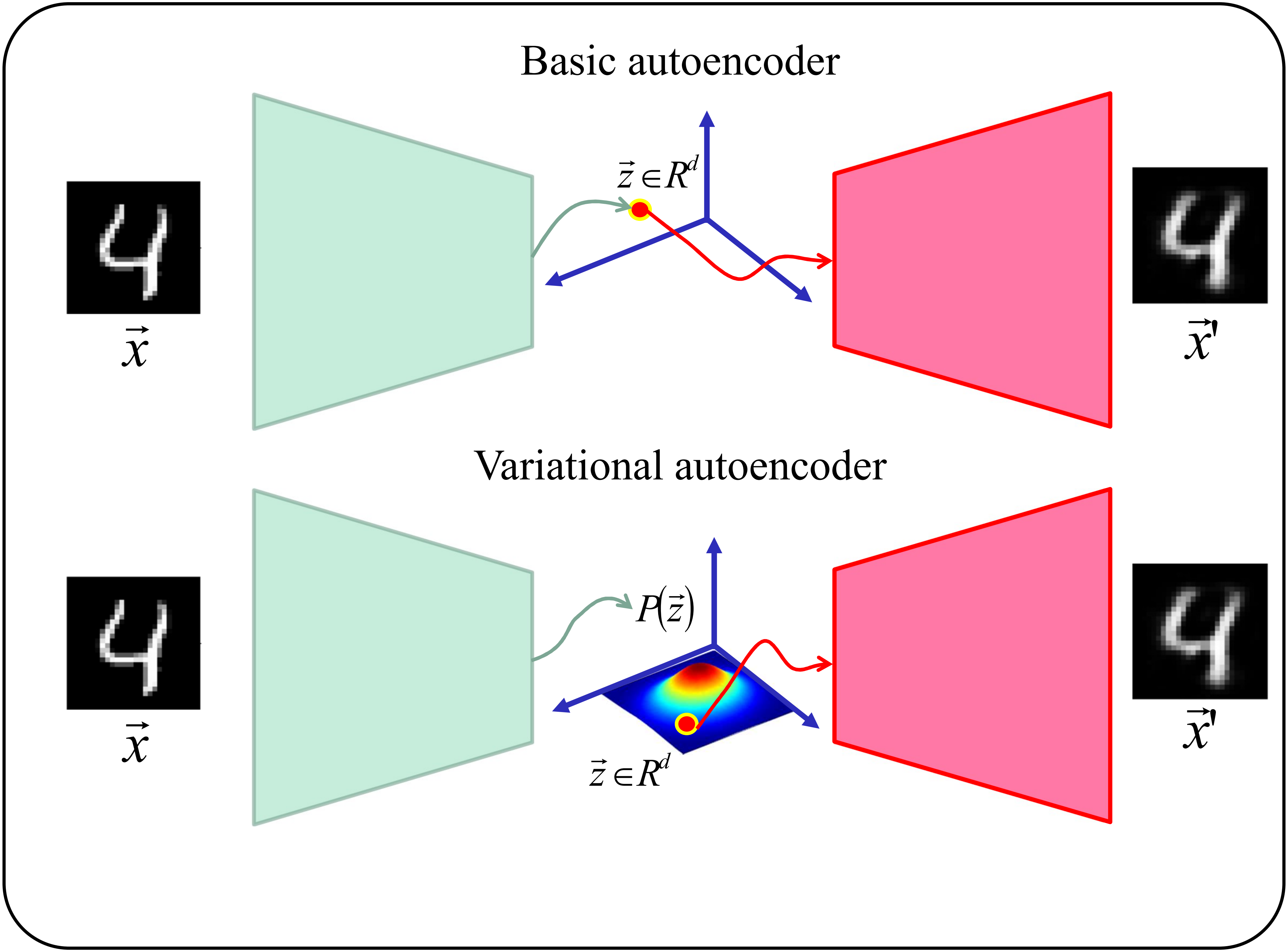


Unfortunately, the vast majority of the time, we will reproduce noise



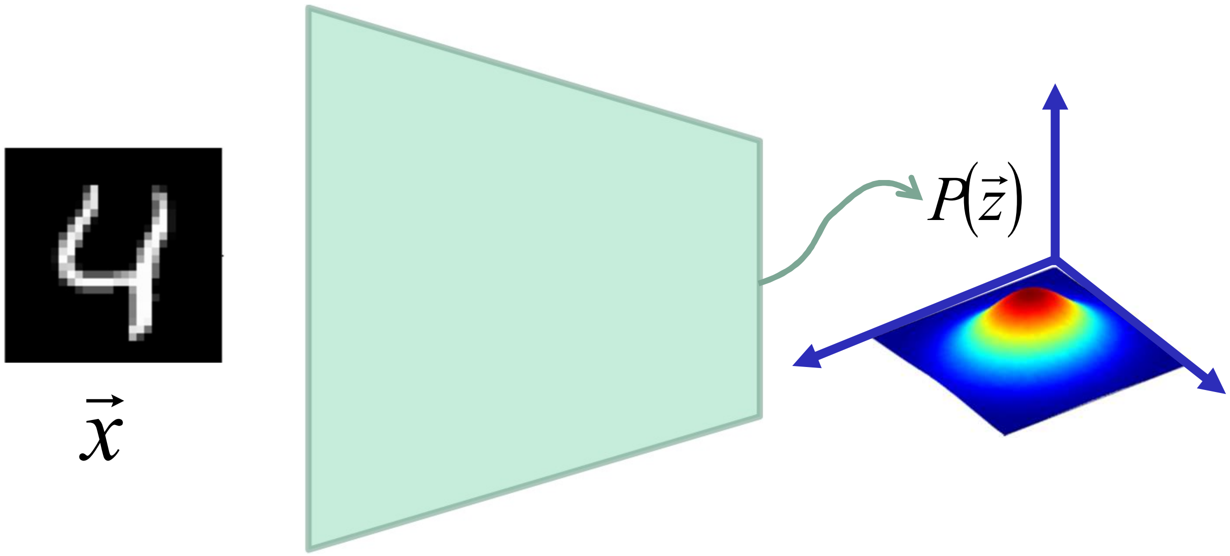
Instead of learning **to reproduce an entry signal**...

Learn how to replicate a **known distribution** $p(\vec{z})$ so that a signal decoded from a **sampled point** is valid



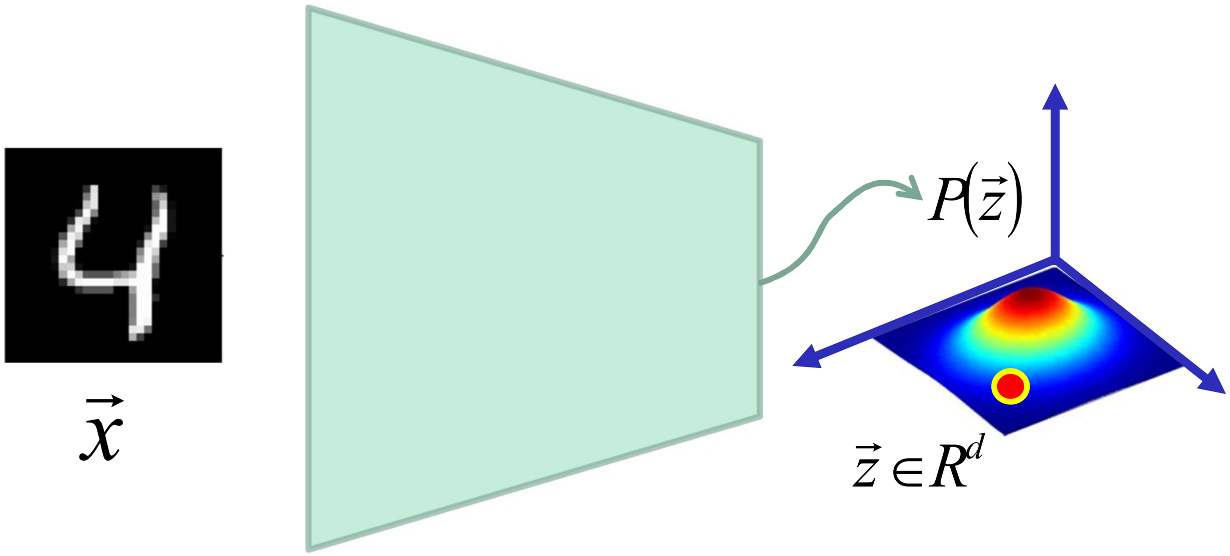
Basic autoencoder
$\vec{z} \in R^d$
$\vec{x}$
$\vec{x}'$
Variational autoencoder
$P(\vec{z})$
$\vec{z} \in R^d$
$\vec{x}$
$\vec{x}'$

Variational autoencoder



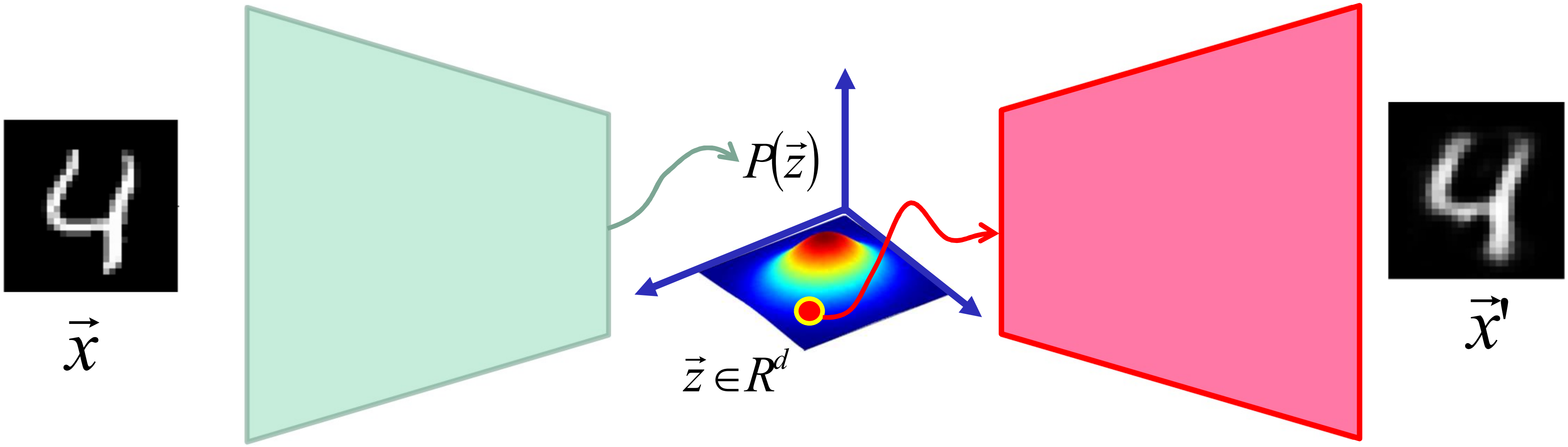
The encoder produces a distribution $P(\vec{z})$ and not just a point $\vec{z}$

Variational autoencoder



We sample one $\vec{z} \sim p(\vec{z})$ at random

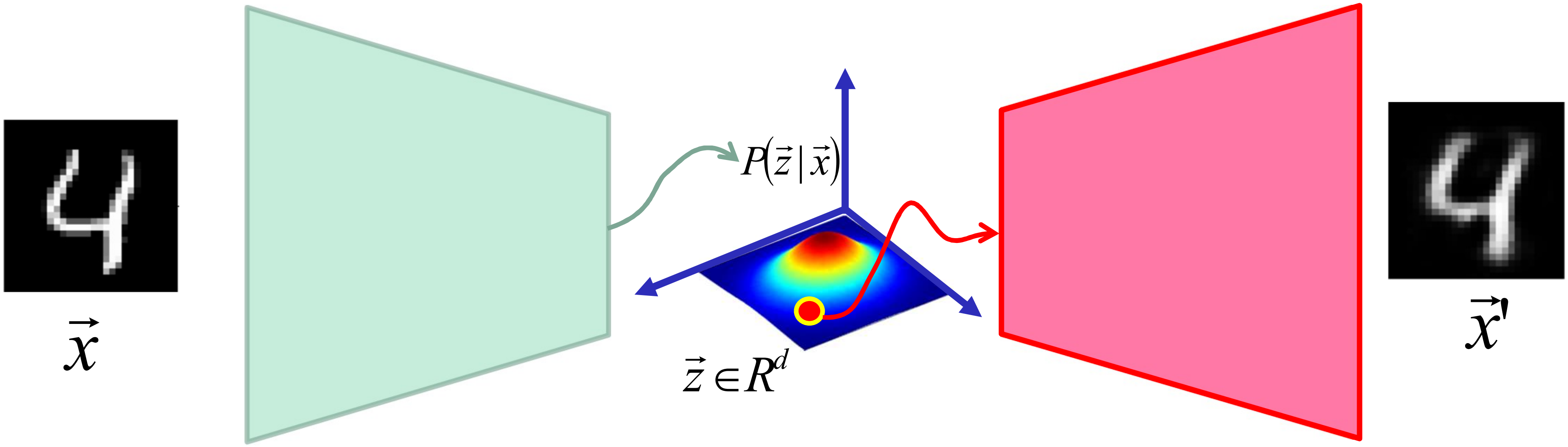
Variational autoencoder



Rebuild $\vec{x}'$

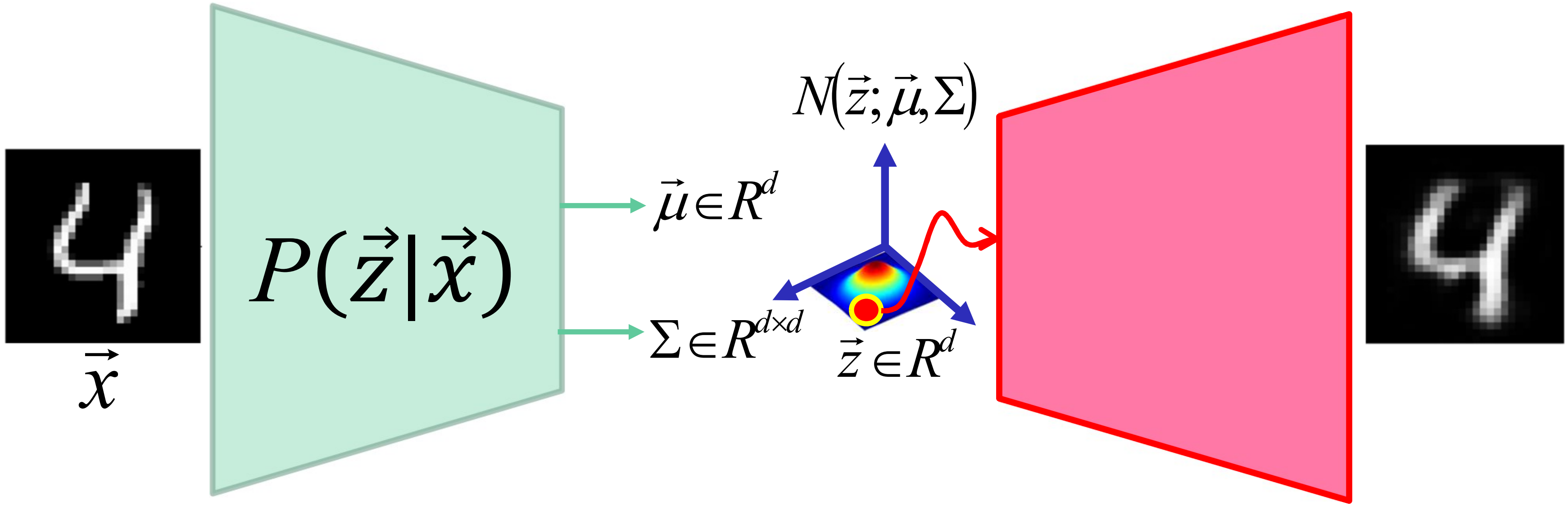
Variational autoencoder

Note 1



Since the distribution of $\vec{z}$ depends on $\vec{x}$ we will say that the learned distribution is

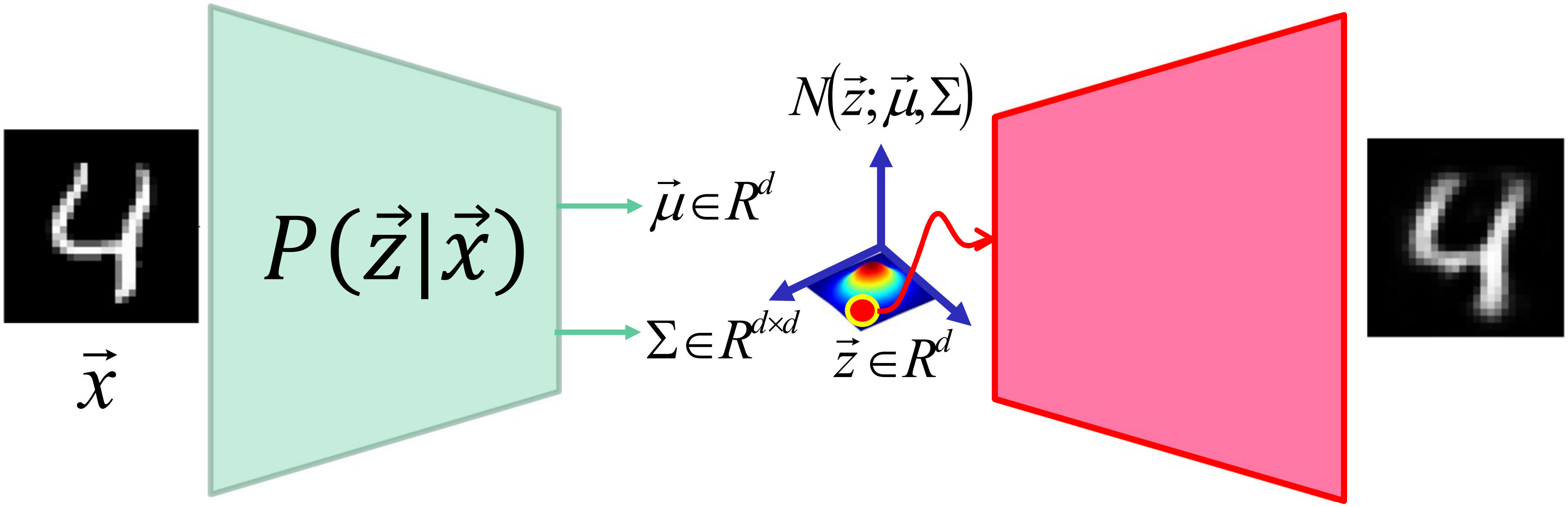
$$P(\vec{z}\,|\,\vec{x})$$



$$P(\vec{z}|\vec{x}) \sim \text{Gaussian}$$

Variational autoencoder

Note 4

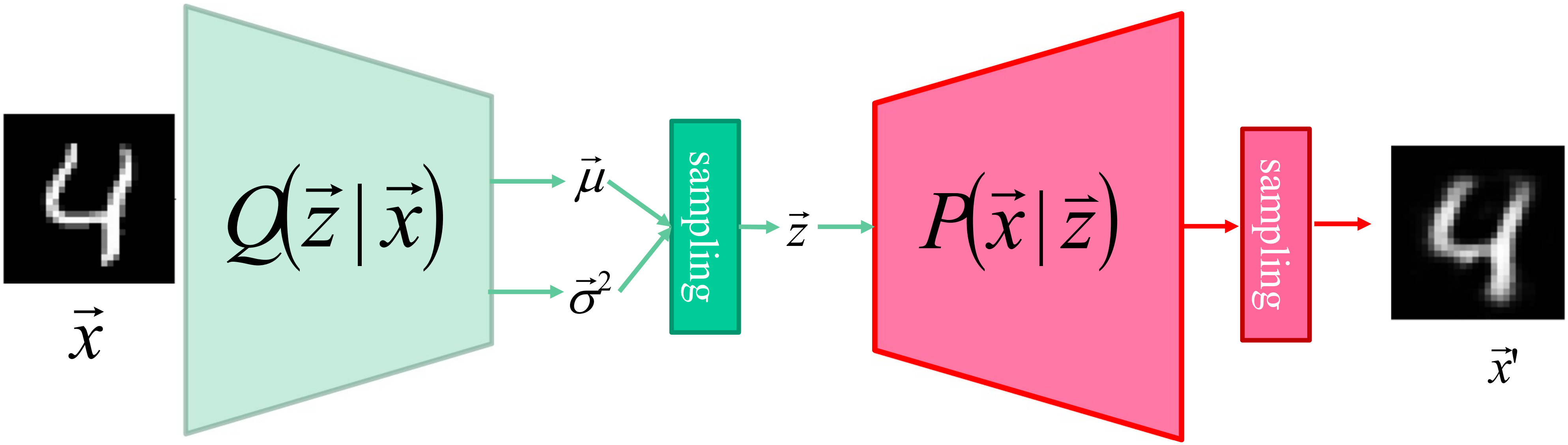


To simplify the calculations, we assume that Σ is diagonal

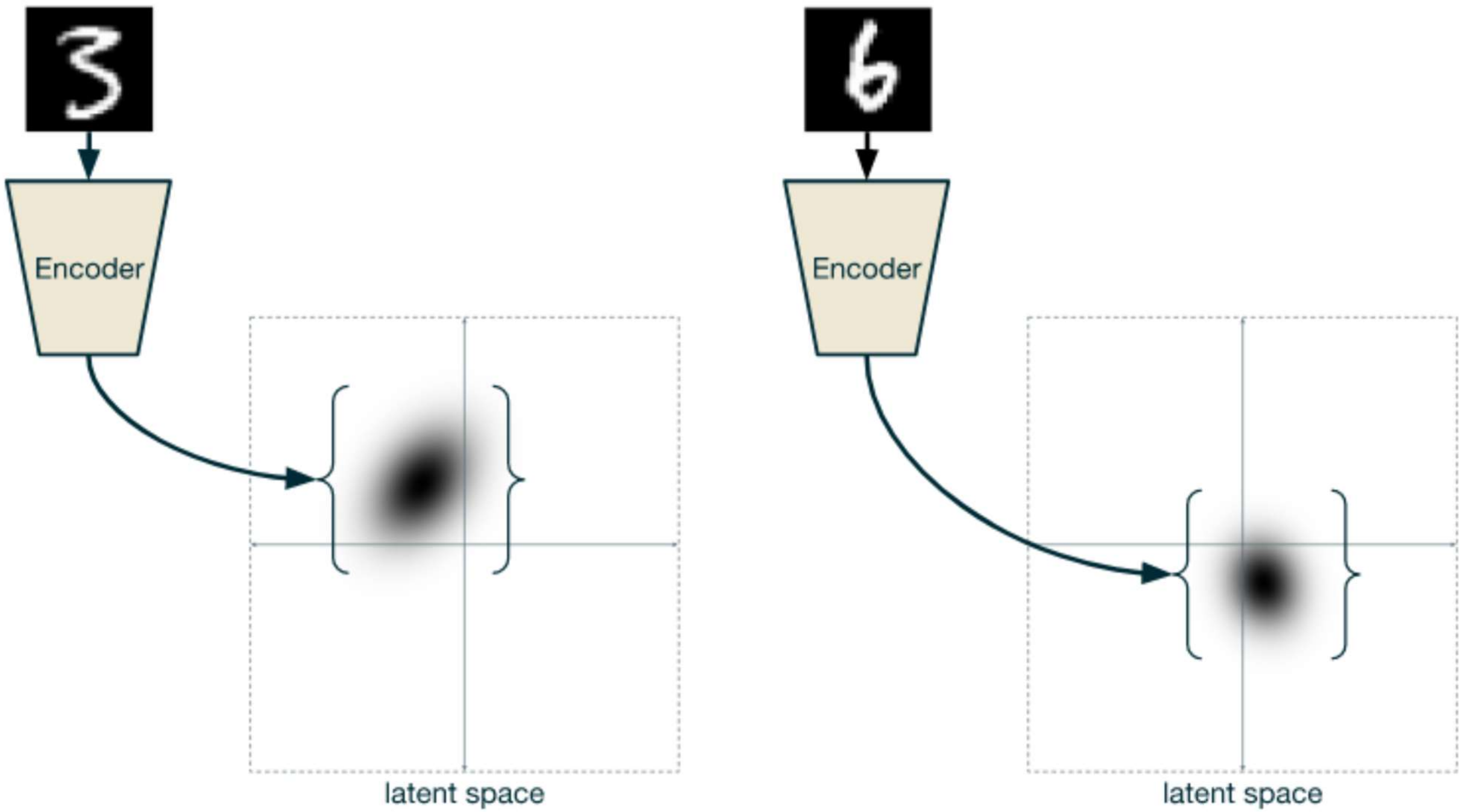
$$\Sigma = \begin{pmatrix} \sigma_1^2 & 0 & 0 & 0 & 0 \\ 0 & \sigma_2^2 & 0 & 0 & 0 \\ 0 & 0 & \sigma_3^2 & 0 & 0 \\ 0 & 0 & 0 & \ddots & 0 \\ 0 & 0 & 0 & 0 & \sigma_d^2 \end{pmatrix}$$

We will therefore predict a vector of variances and not a matrix

Variational autoencoder

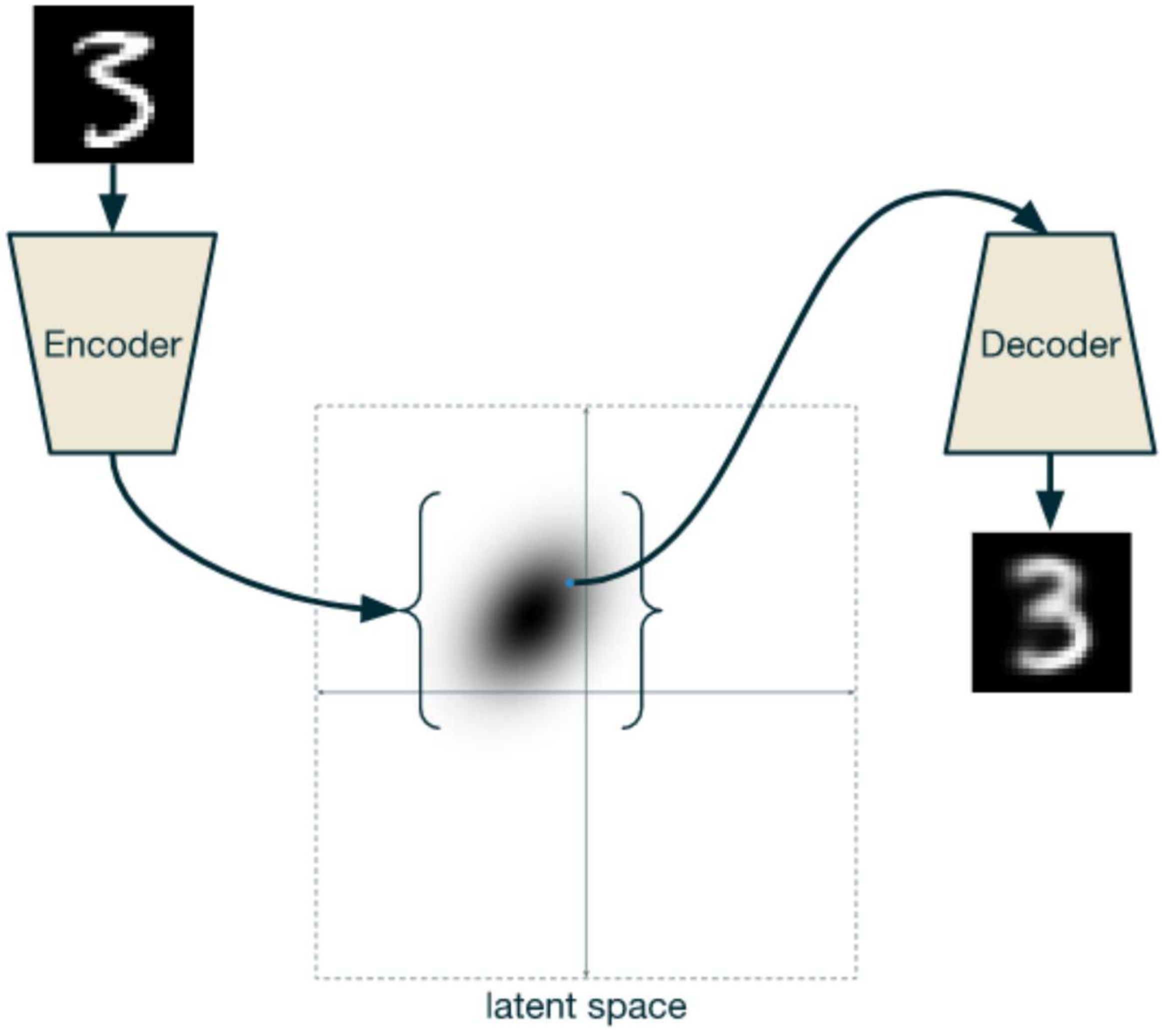


Another way of looking at things...



https://ijdykeman.github.io/ml/2016/12/21/cvae.html

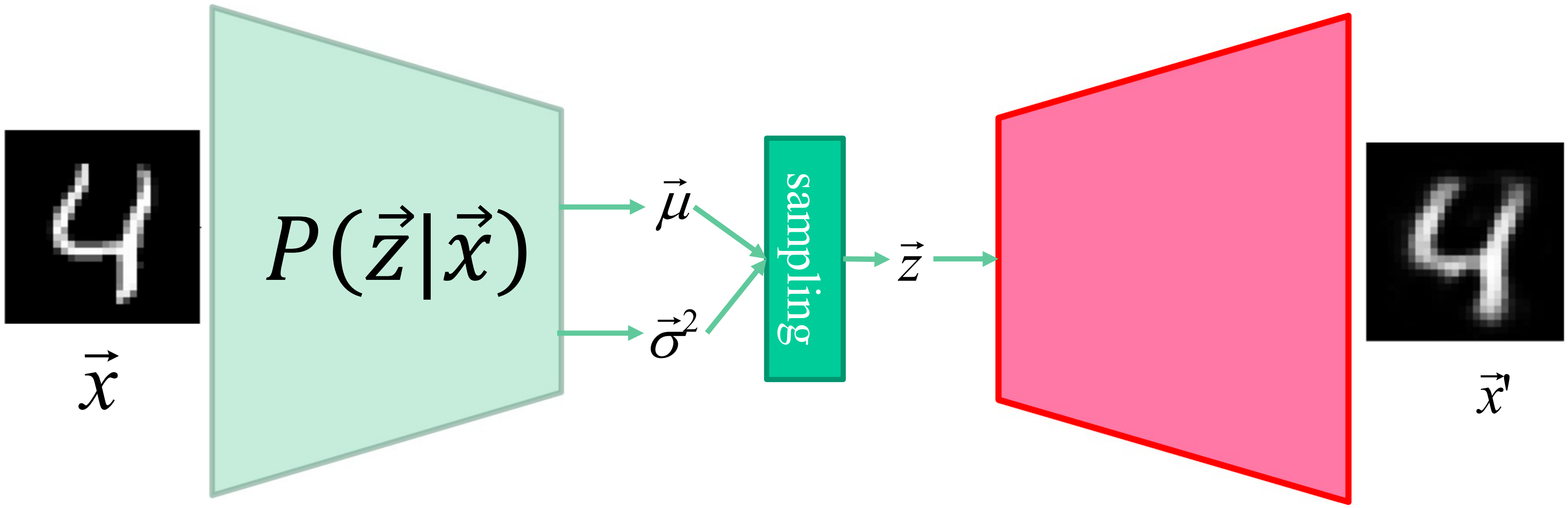
Another way of looking at things...



https://ijdykeman.github.io/ml/2016/12/21/cvae.html

Variational autoencoder

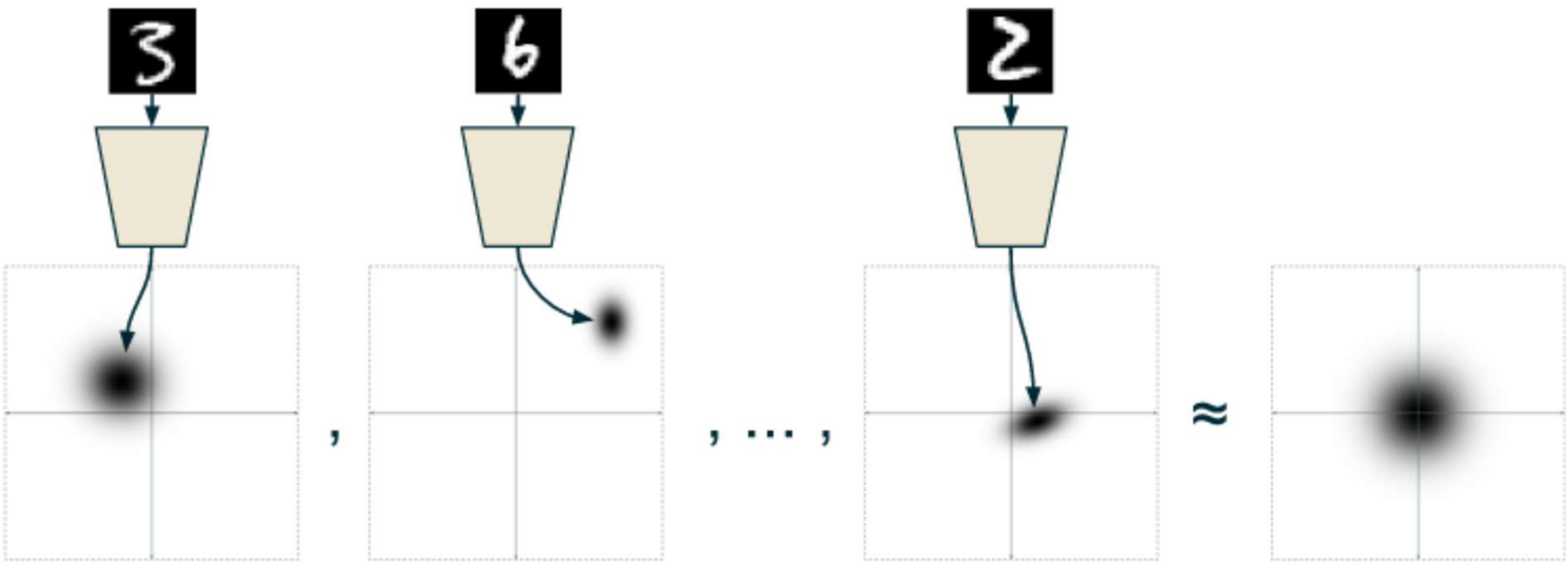
Note 5



A priori distribution of: $\vec{z}$ Gaussian centered at 0 and variance 1

$$P(\vec{z}) = N(\vec{z};0,1)$$

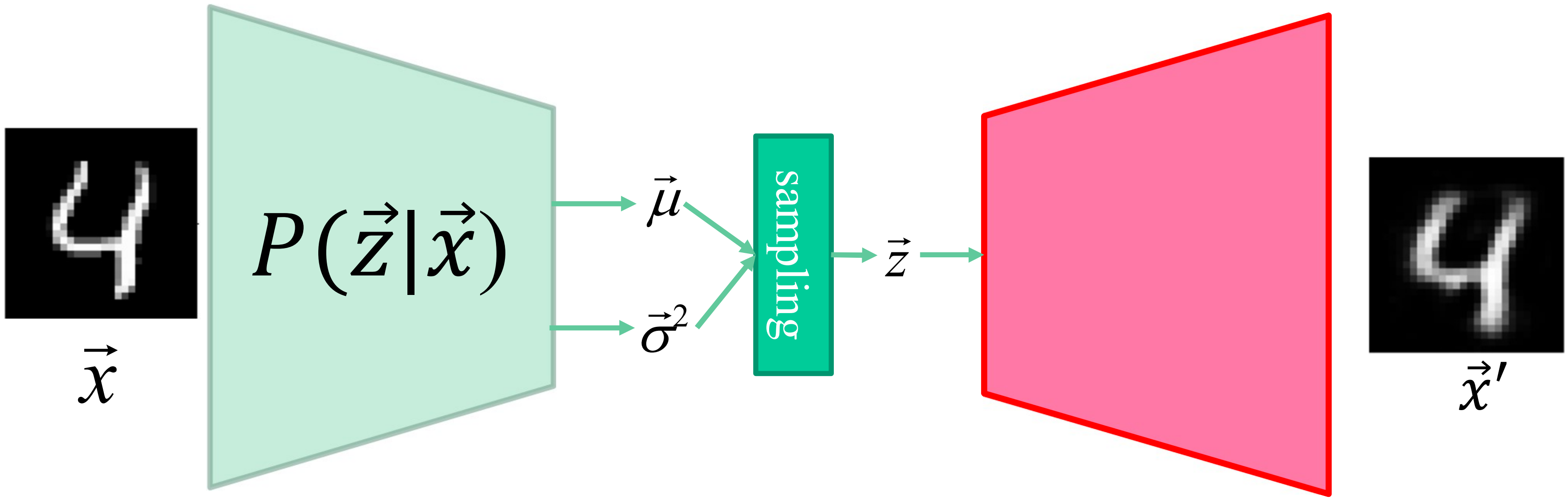
Another way of looking at things...



$$N\left(\vec{z};0,1\right)$$

https://ijdykeman.github.io/ml/2016/12/21/cvae.html

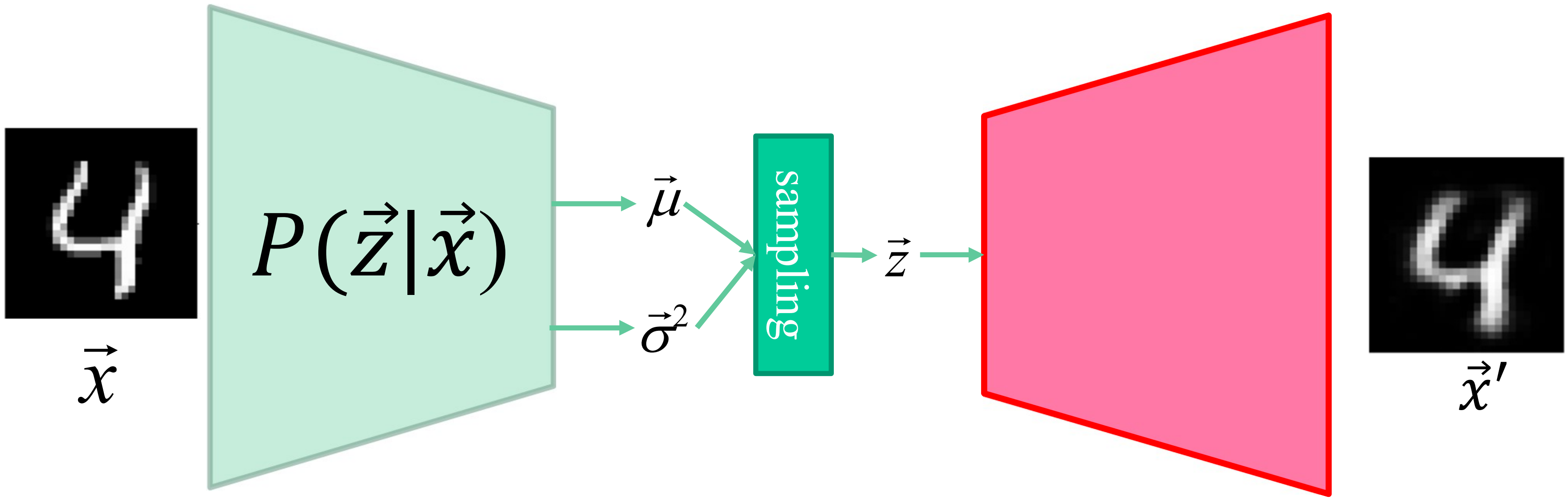
Variational autoencoder



ELBO loss : Evidence Lower Bound

$$Loss = \underbrace{KL\Big(N(\vec{z};0,1), N(\vec{z};\vec{\mu},\Sigma)\Big)}_{\text{Encoder loss}} + \underbrace{\lambda\|\vec{x} - \vec{x}'\|^2}_{\text{Decoder loss}}$$

Variational autoencoder

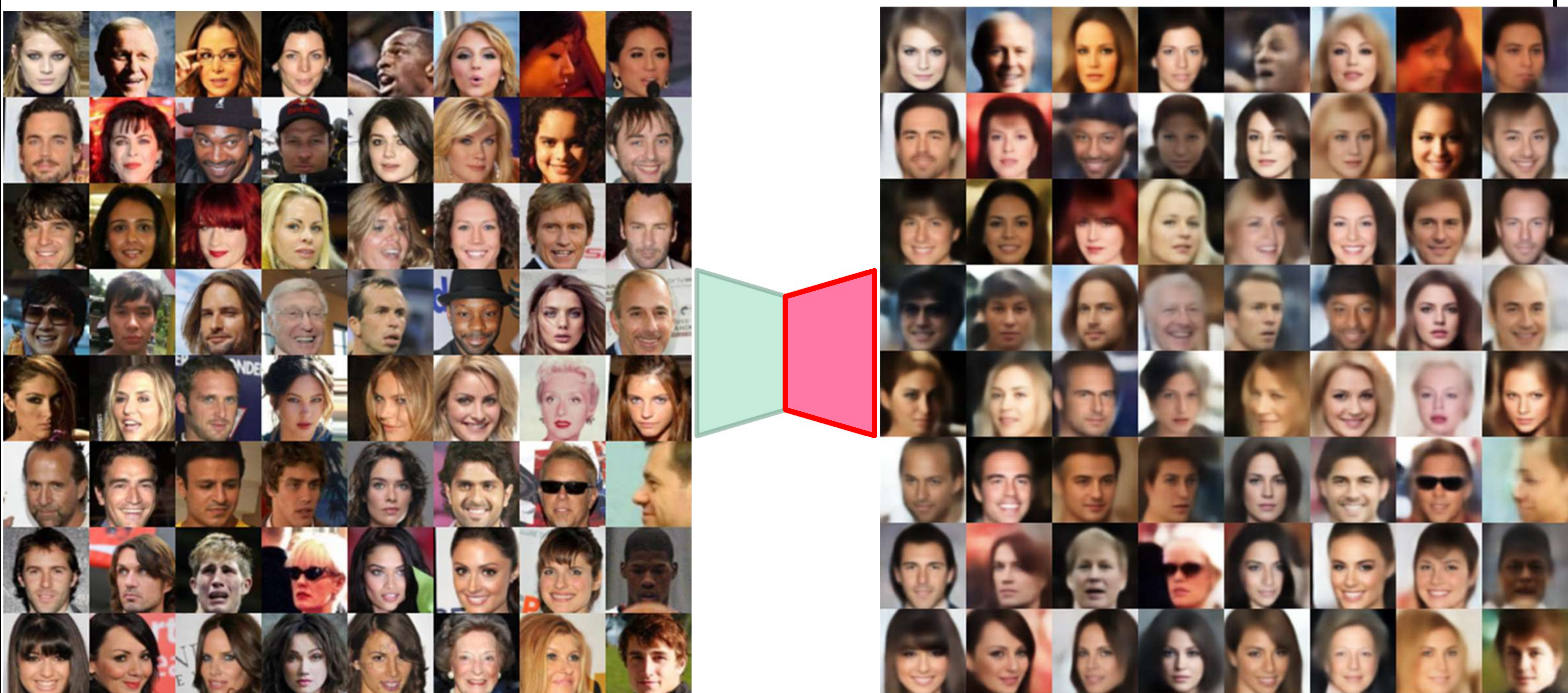


ELBO loss : Evidence Lower Bound

$$Loss = \underbrace{\frac{1}{2}\sum_{i=1}^{d}\left(1 + \log\left(\sigma_i^2\right) - \mu_i^2 - \sigma_i^2\right)}_{\text{Encoder loss}} + \underbrace{\lambda\|\vec{x} - \vec{x}'\|^2}_{\text{Decoder loss}}$$

E.g.: CelebA database

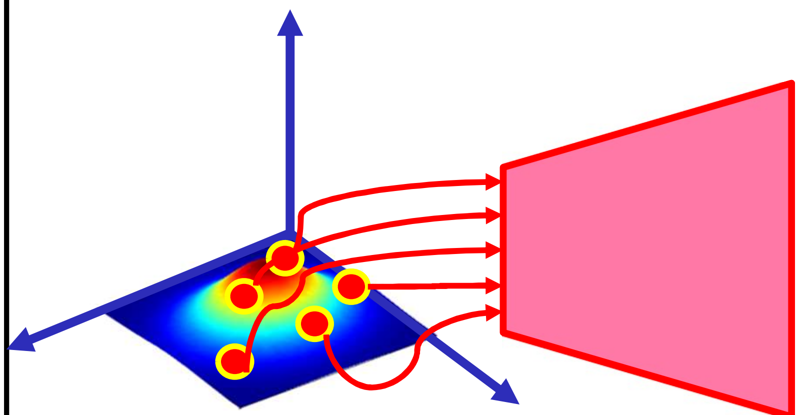
$\vec{x}$ $\vec{x}'$



https://github.com/yzwxx/vae-celebA

Ex.: *CelebA* database

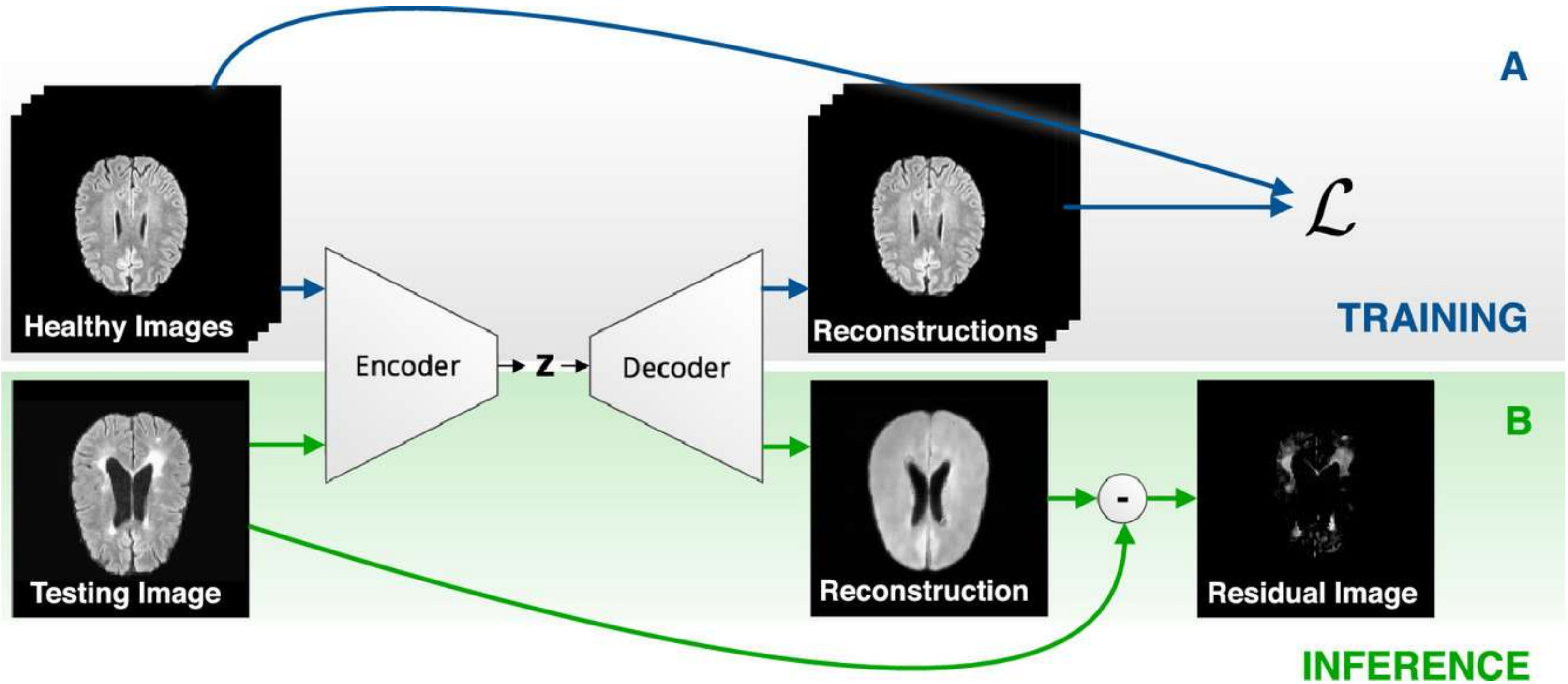
Decoding of random samples $\vec{z}$



https://github.com/yzwxx/vae-celebA

Medical imaging application

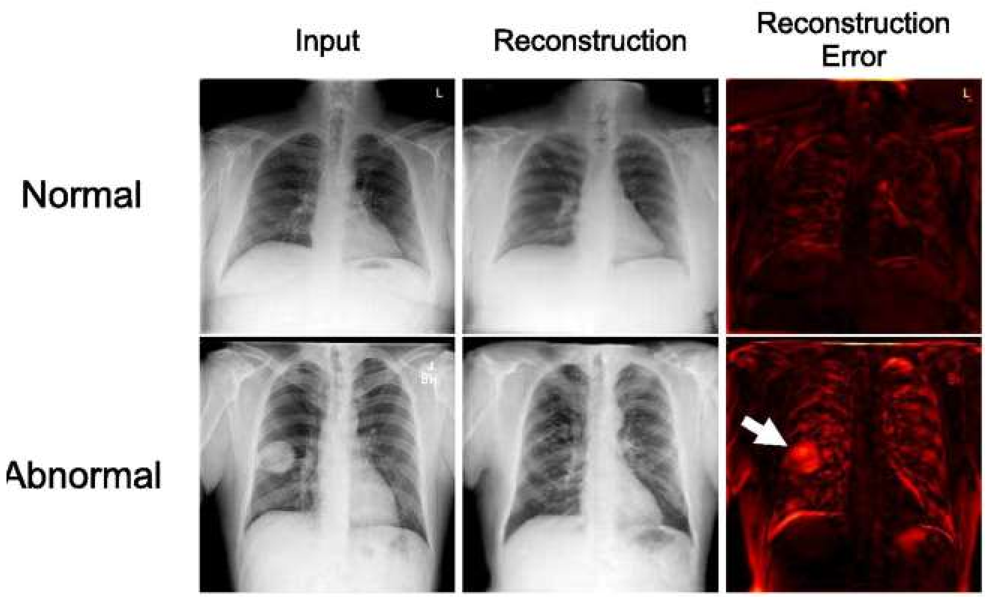
Anomaly detection, *[Baur et al'21]*

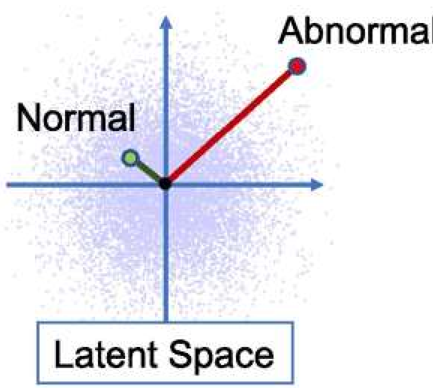


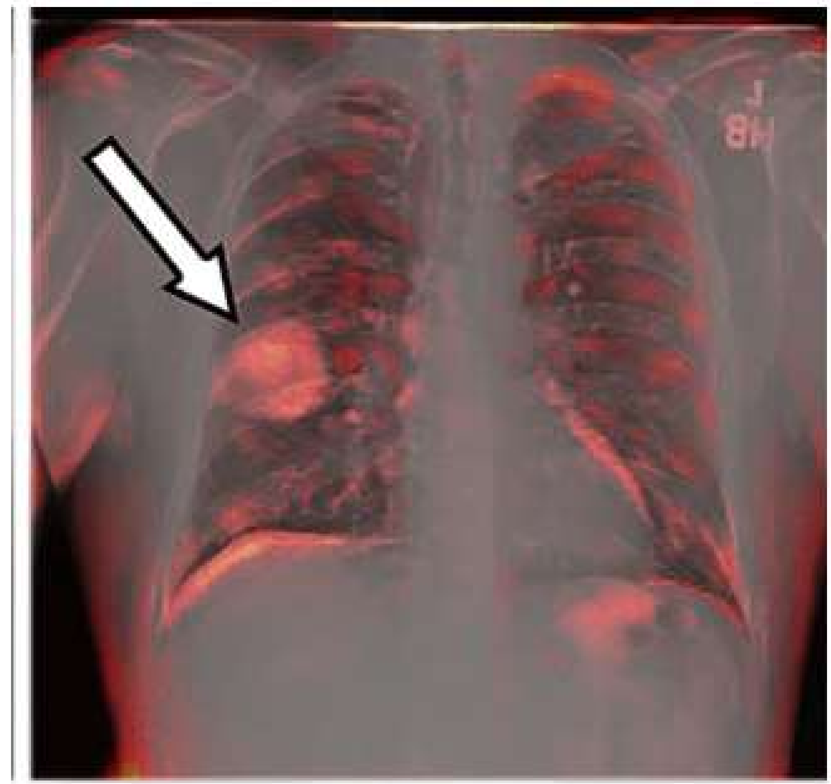
https://www.sciencedirect.com/science/article/pii/S1361841520303169

Medical imaging application

Anomaly detection, *[Nakano et al'21]*



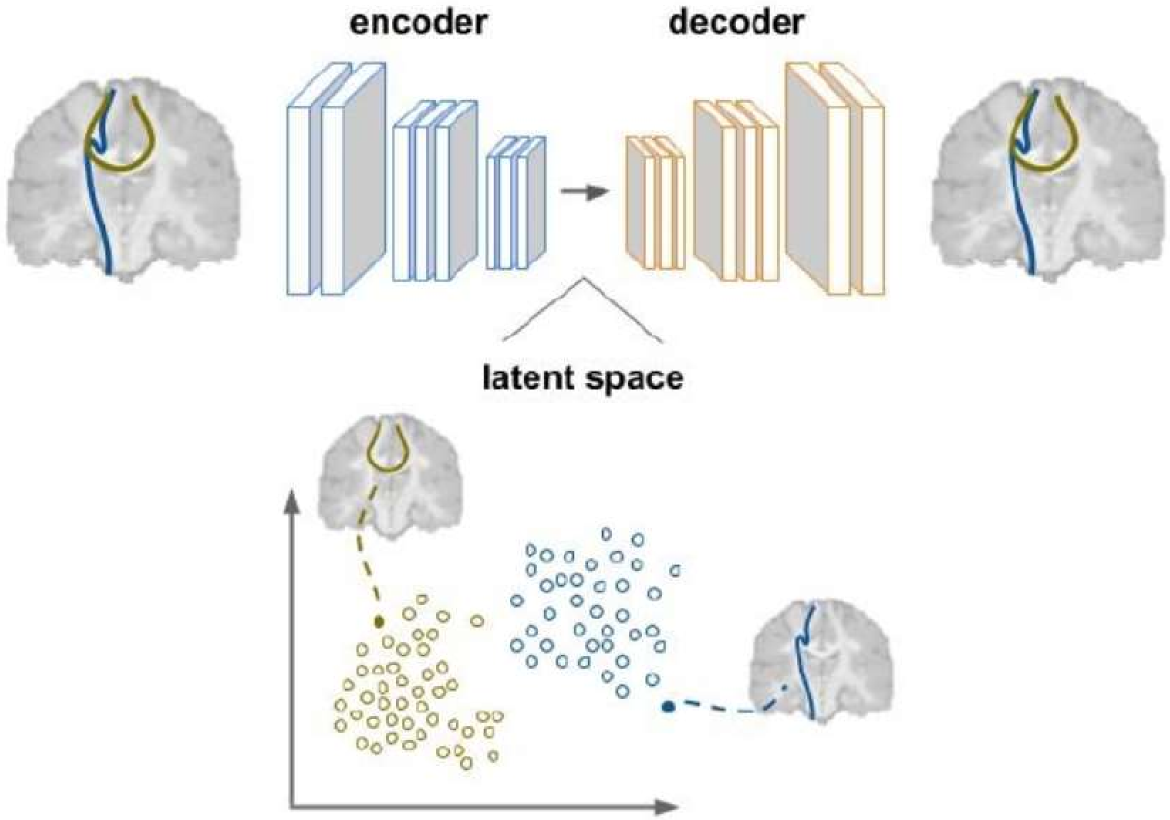




https://link.springer.com/article/10.1007/s10278-020-00413-2

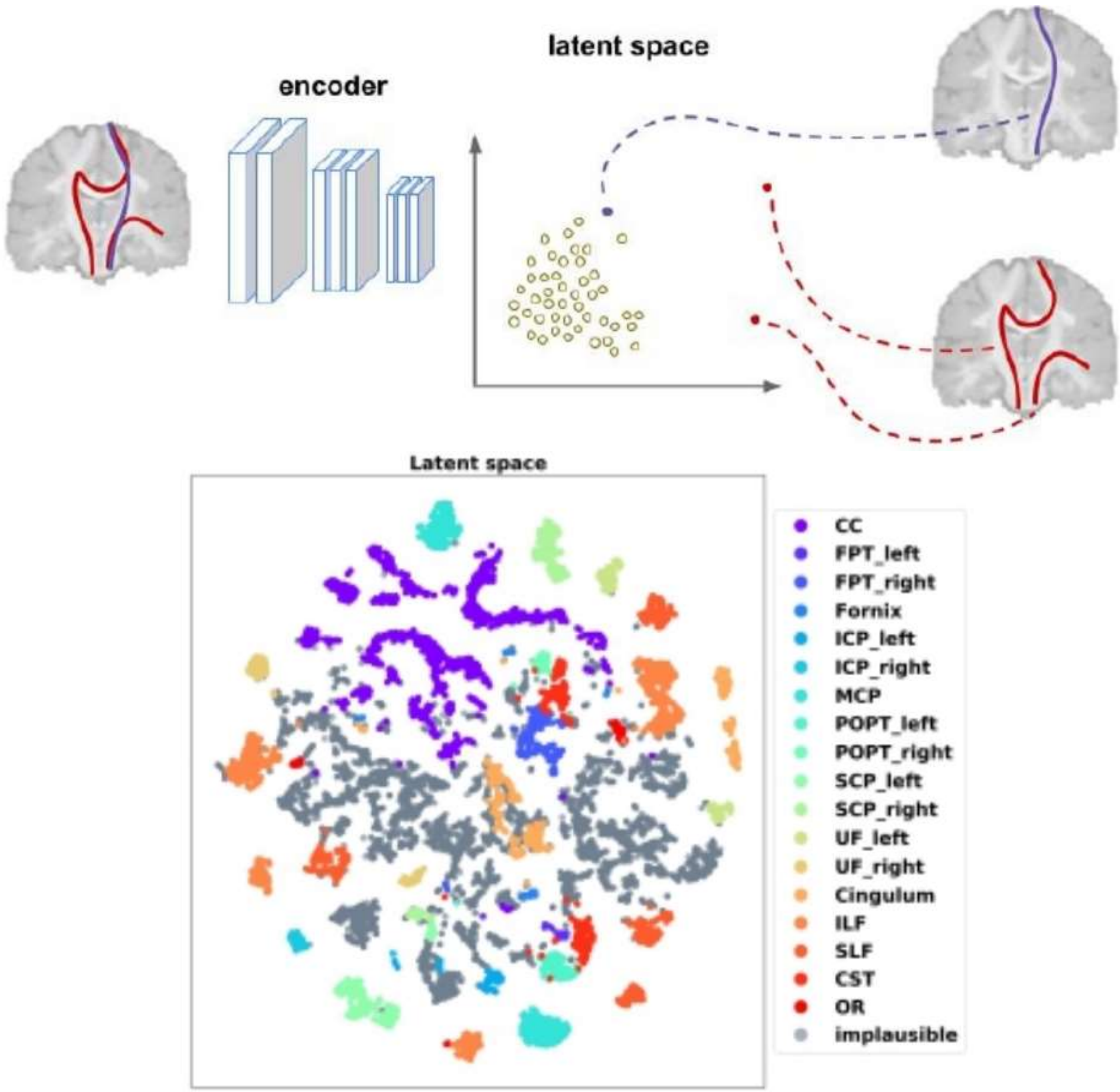
Medical imaging application

Brain fiber filtering, *[Dumais et al'23, Legaretta et al'22]*



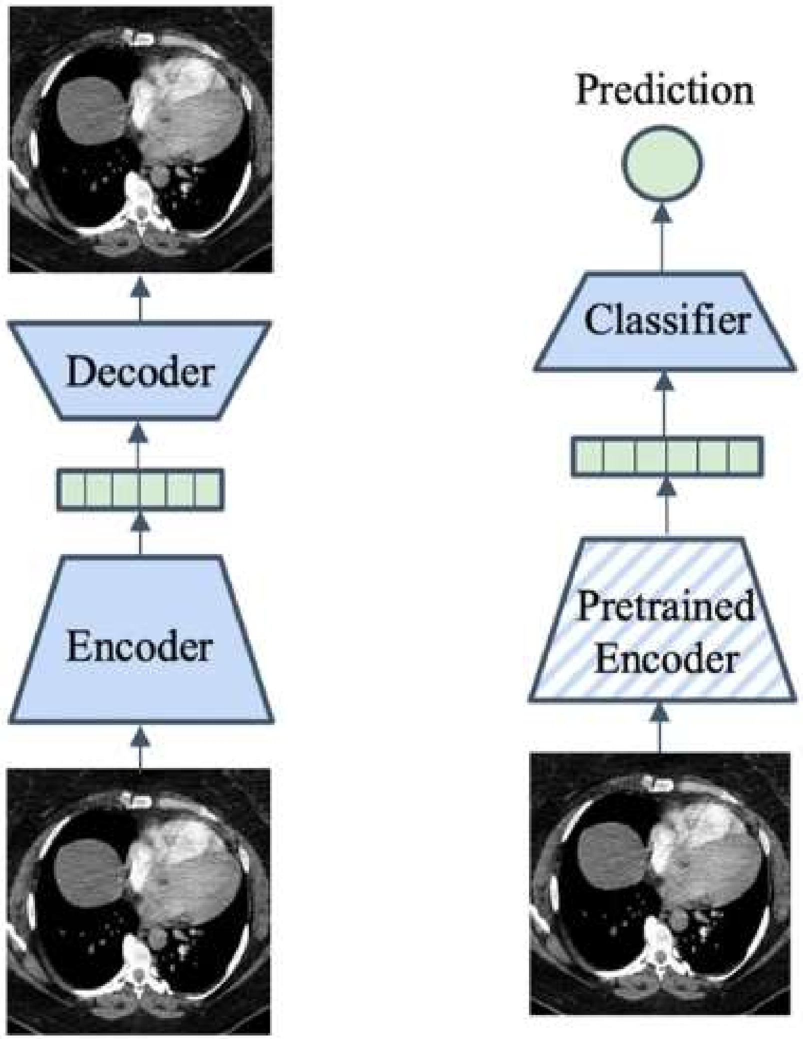
Medical imaging application

Brain fiber filtering, *[Dumais et al'23, Legaretta et al'22]*



Medical imaging application

Pre-training, [*Huang et al. '23*]



https://www.nature.com/articles/s41746-023-00811-0

Several tutorials, VAE

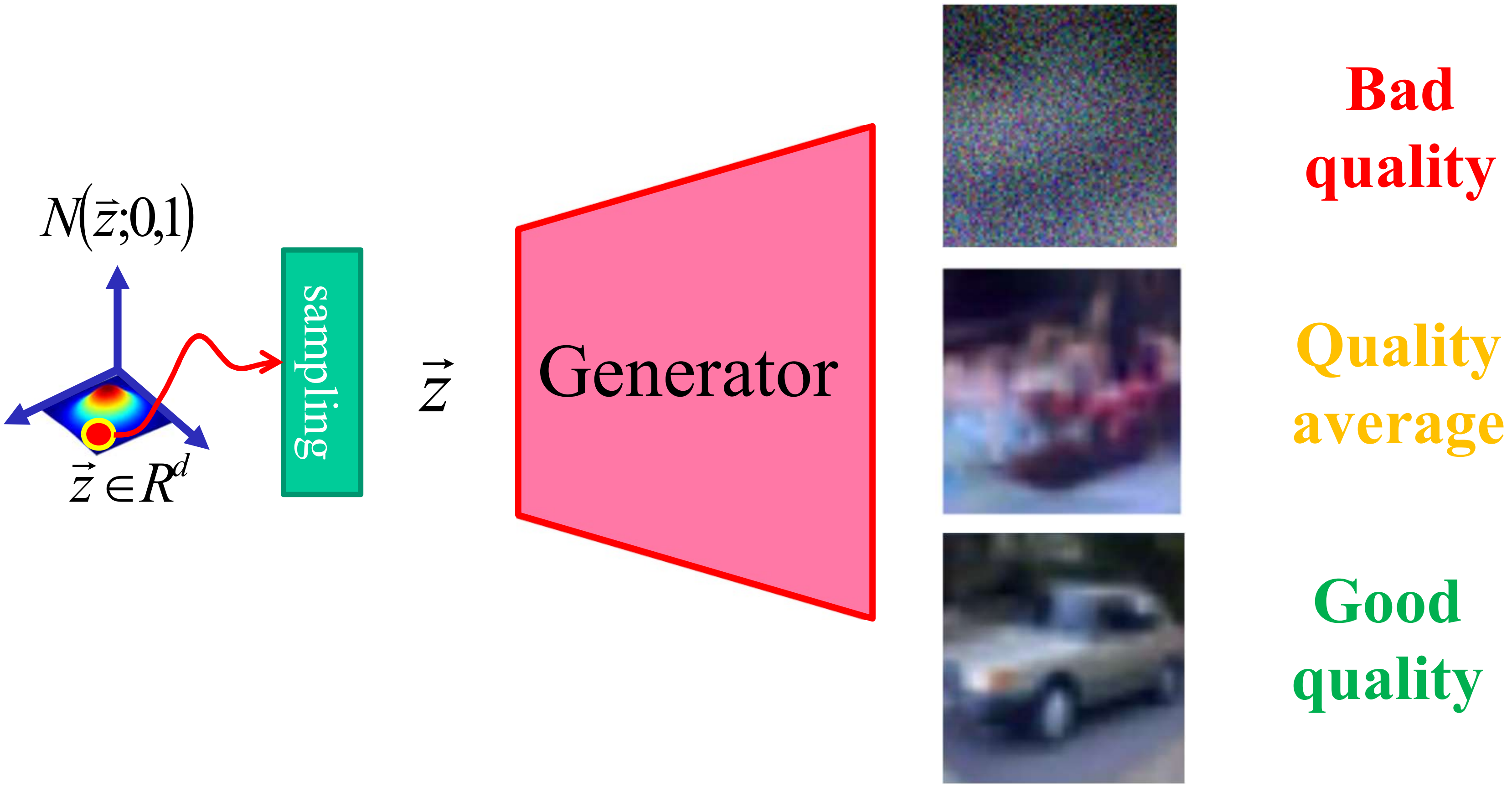
- https://ijdykeman.github.io/ml/2016/12/21/cvae.html
- https://wiseodd.github.io/techblog/2016/12/10/variational-autoencoder/
- https://towardsdatascience.com/deep-latent-variable-models-unravel-hidden-structures-a5df0fd32ae2
- C. Doersch, **Tutorial on Variational Autoencoders,** arXiv:1606.05908

GAN

Generative Adversarial Nets

VAE

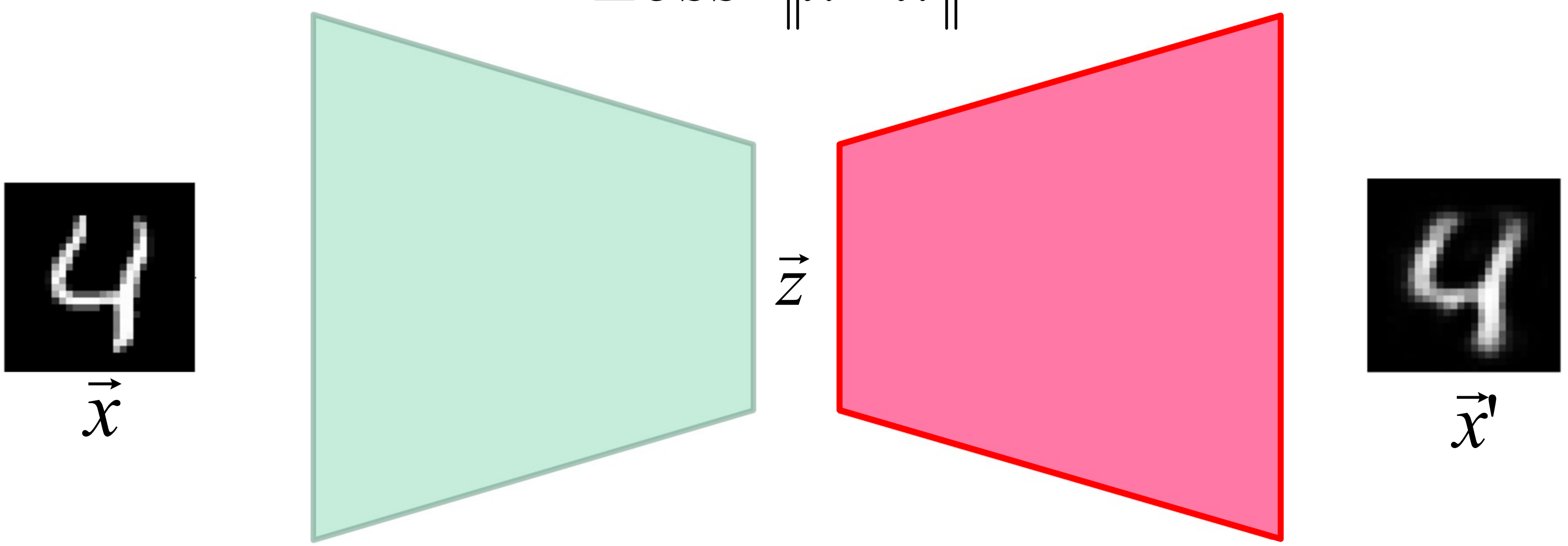
Depending on the performance of the decoder, the images generated will **vary greatly in quality**.



To train a decoder, you need a **loss function** that measures the **degree of realism** of the images produced

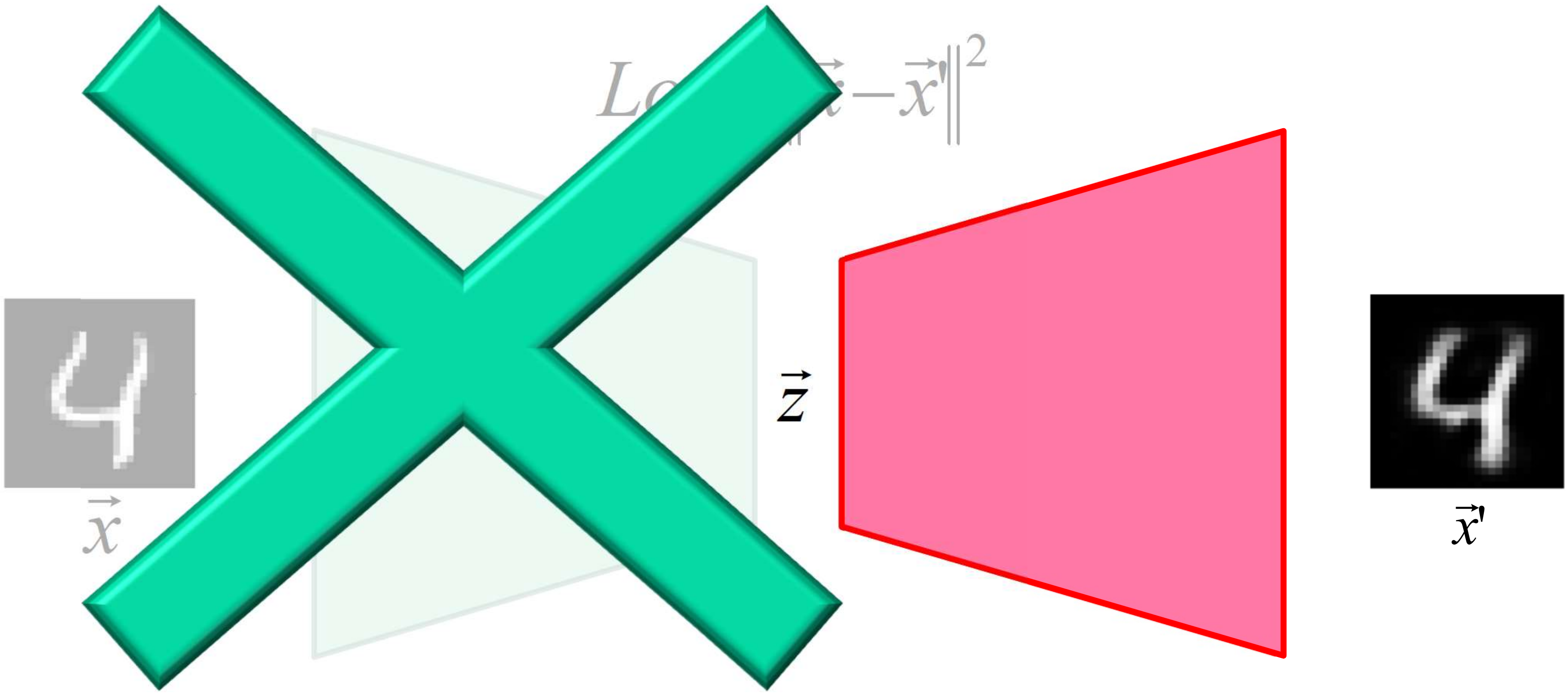
For an autoencoder (variational or not) it's easy!
because we have an **encoder and a reference image**

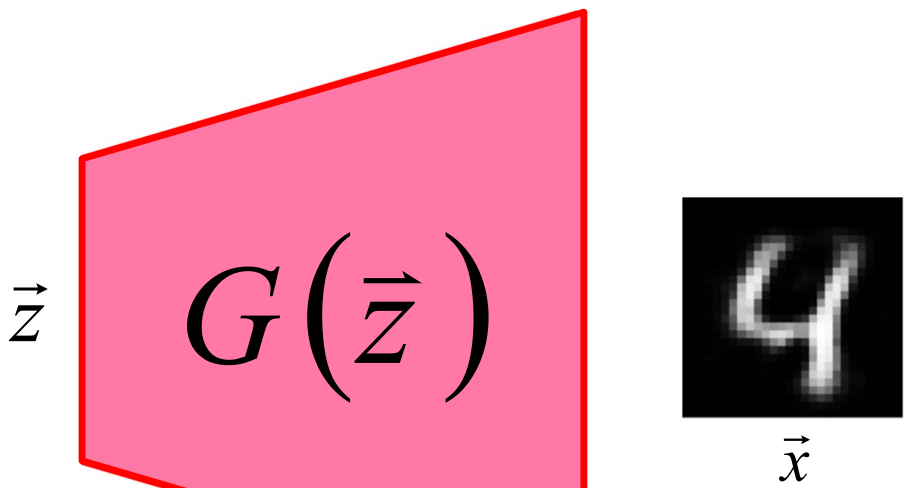
$$Loss = \|\vec{x} - \vec{x}'\|^2$$



To train a decoder, you need a **loss function** that measures the **degree of realism** of the images produced

How do you do an encoderless network?





Loss without a reference target?

Approximate the loss using a

2nd neural network

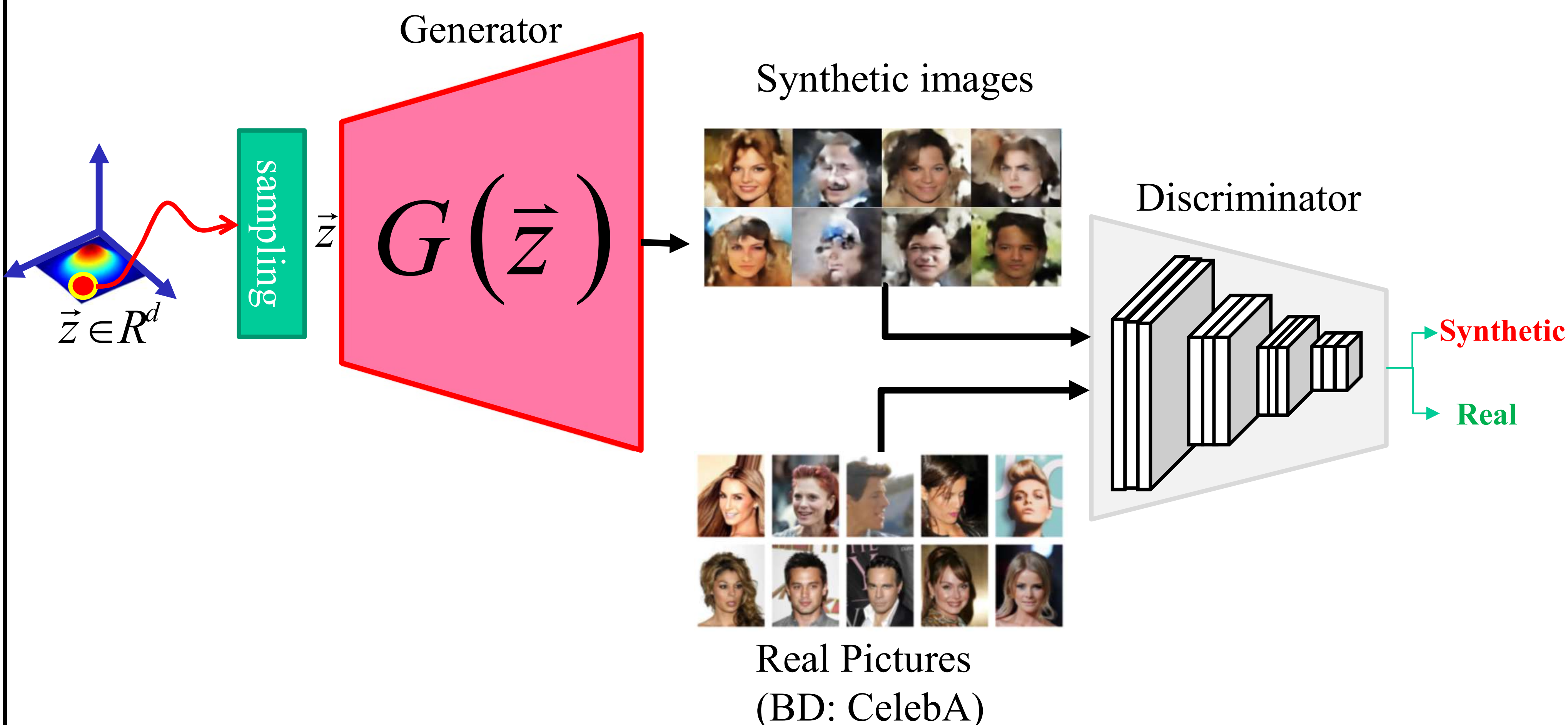
and a

reference database

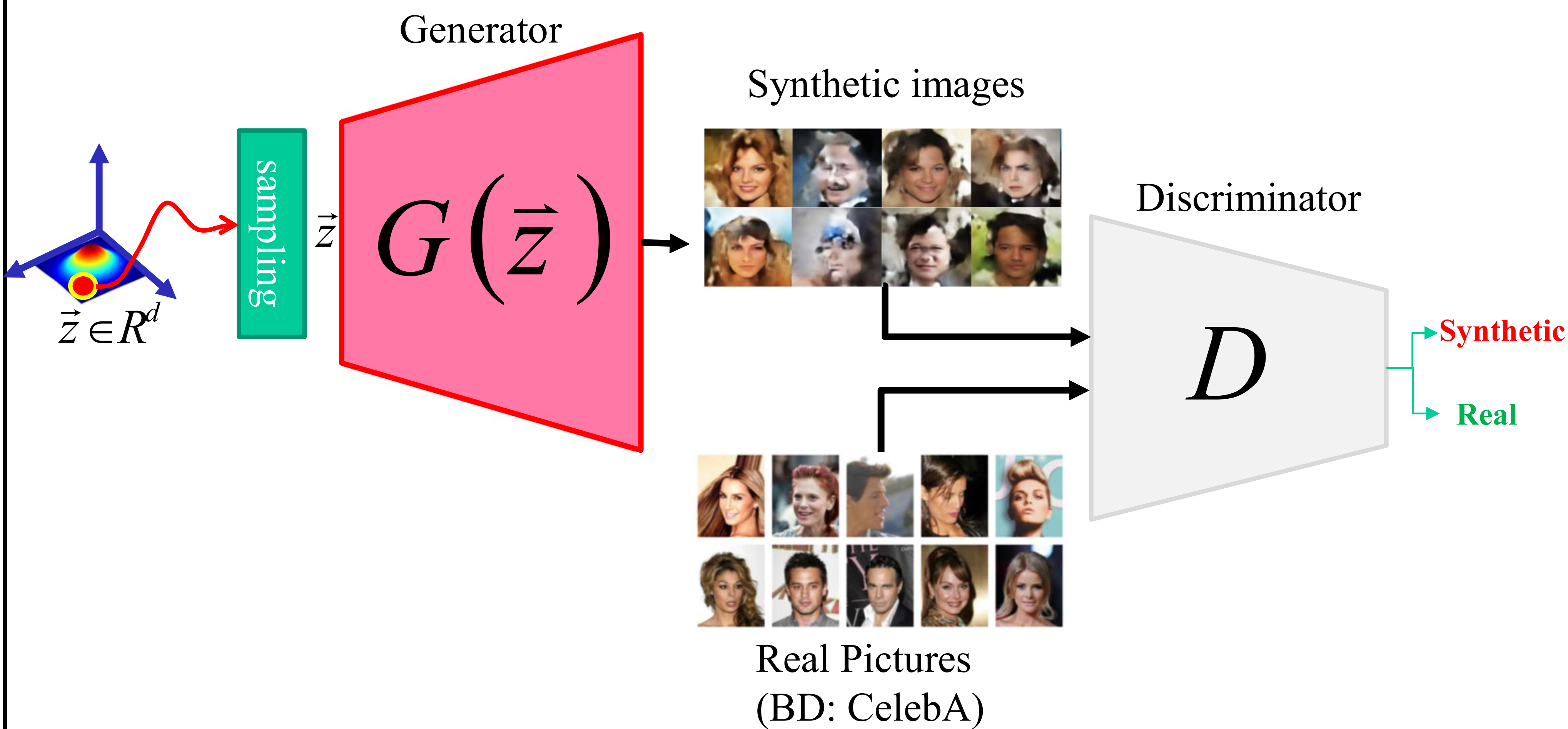
$\vec{z}$

référence?

To drive a decoder, you need a **loss function** that measures the **quality of the images** produced



To drive a decoder, you need a **loss function** that measures the **quality of the images** produced



Labeled data

Produced by the generator

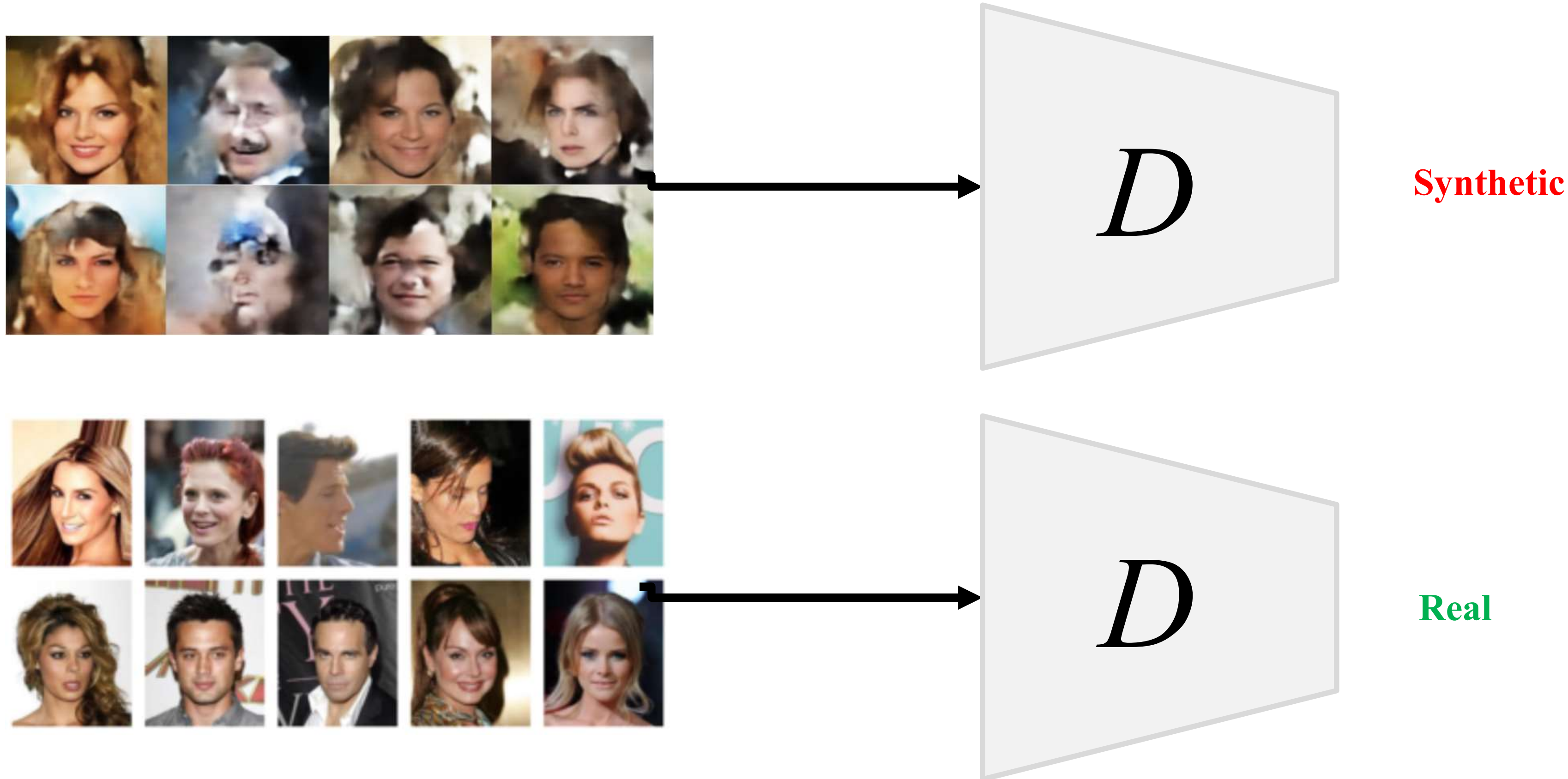
$t = 0 \quad ('synthetic')$

$t = 1 \quad ('real')$

From a real DB

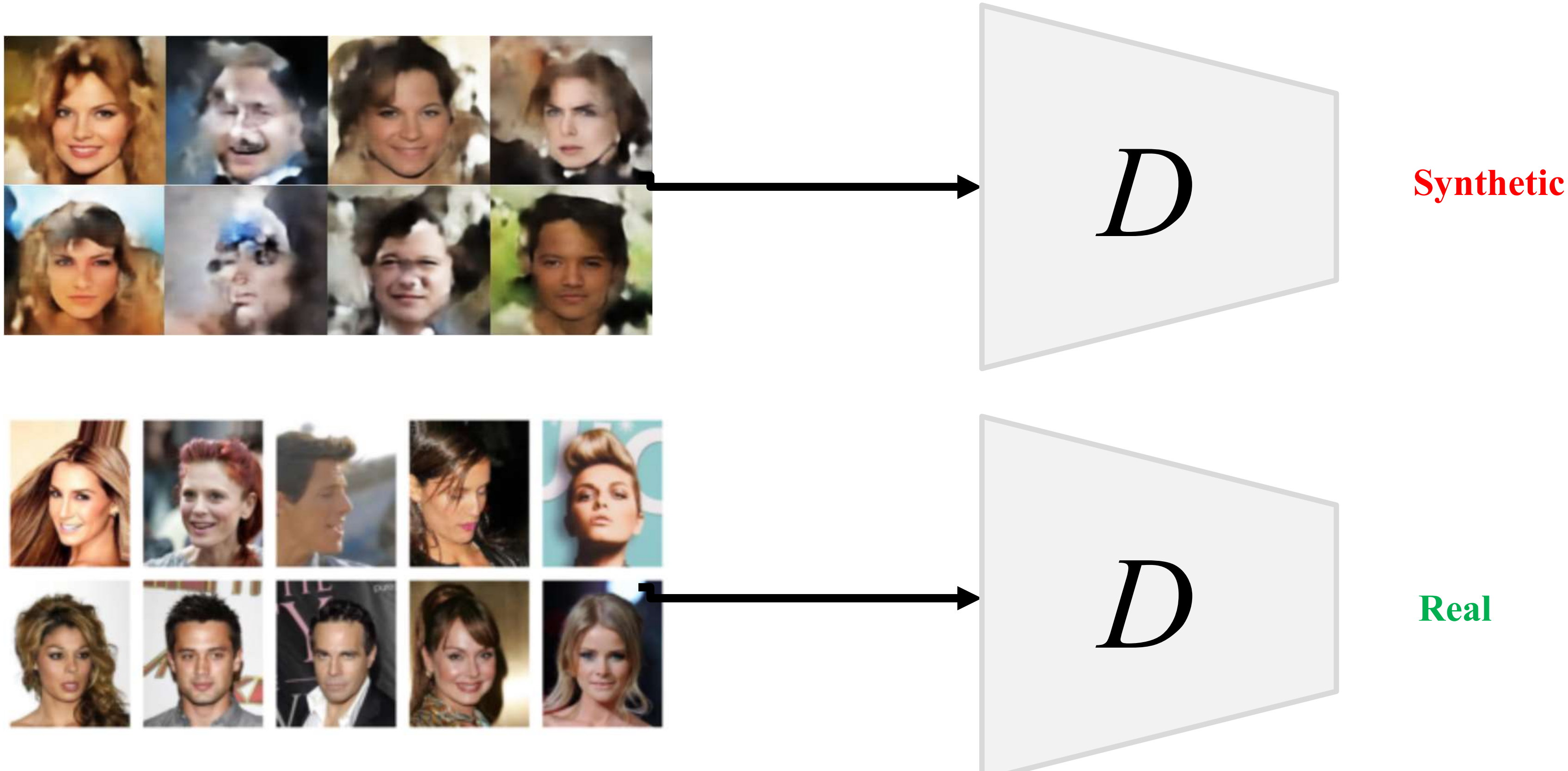
Two networks with **different objectives**:

Discriminator: differentiates synthetic images from real images



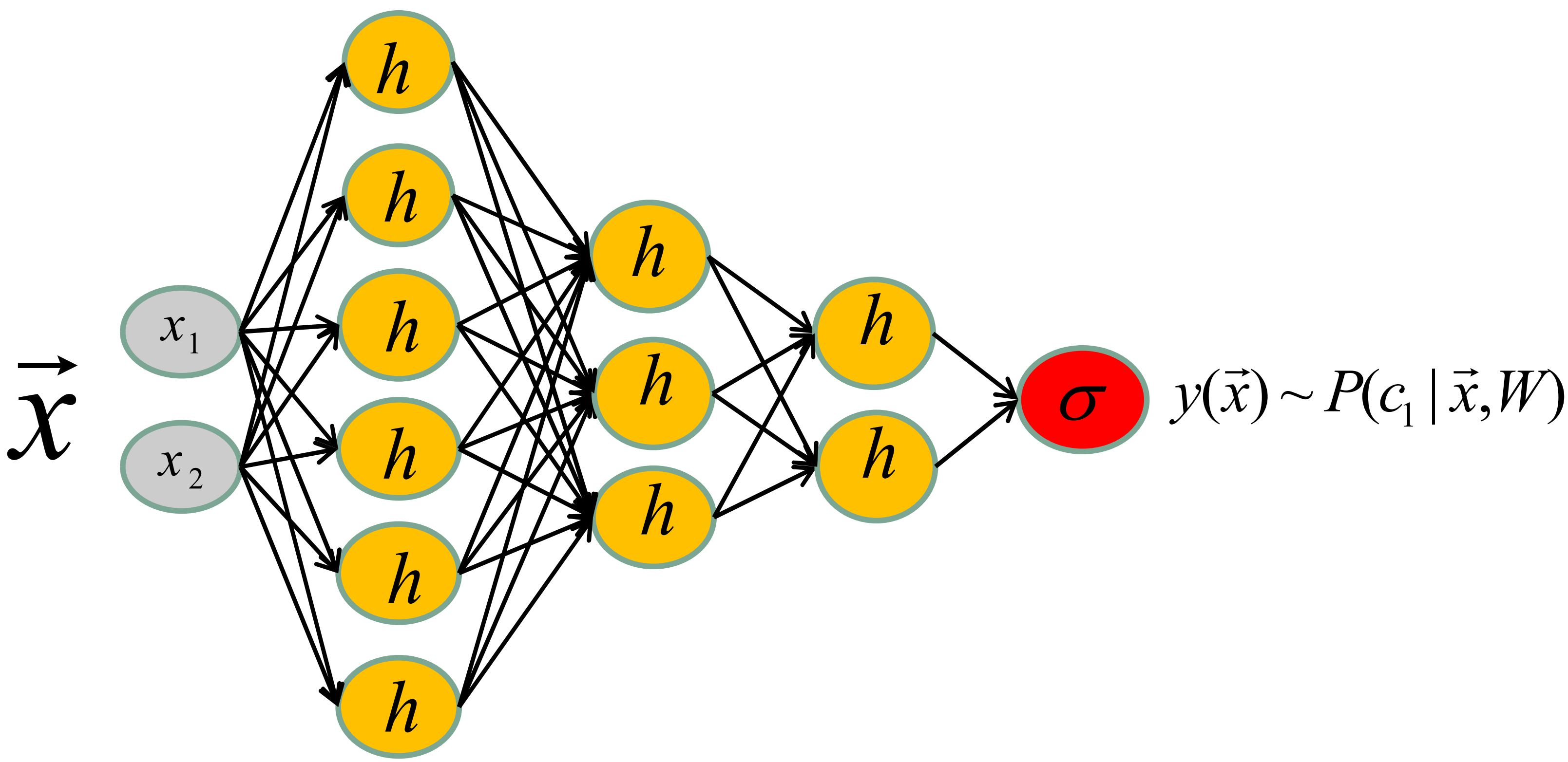
Discriminator: binary classifier (logistic regression)

=> cross-entropy loss



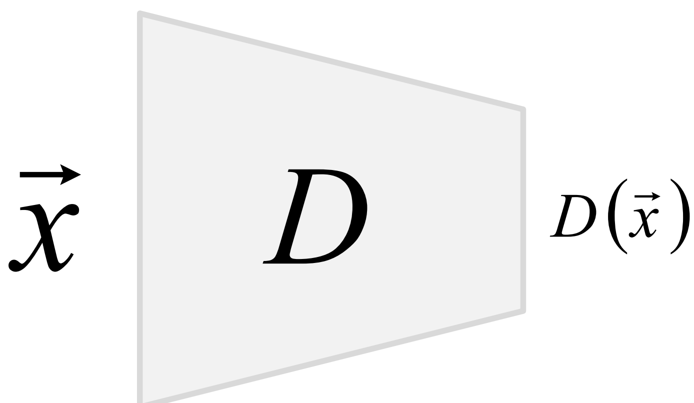
Reminder, cross-entropy for binary logistic classification:

$$L_D = \frac{1}{N}\sum_i -t_i \ln\left(y(\vec{x}_i)\right) - (1-t_i)\ln\left(1-y(\vec{x}_i)\right)$$



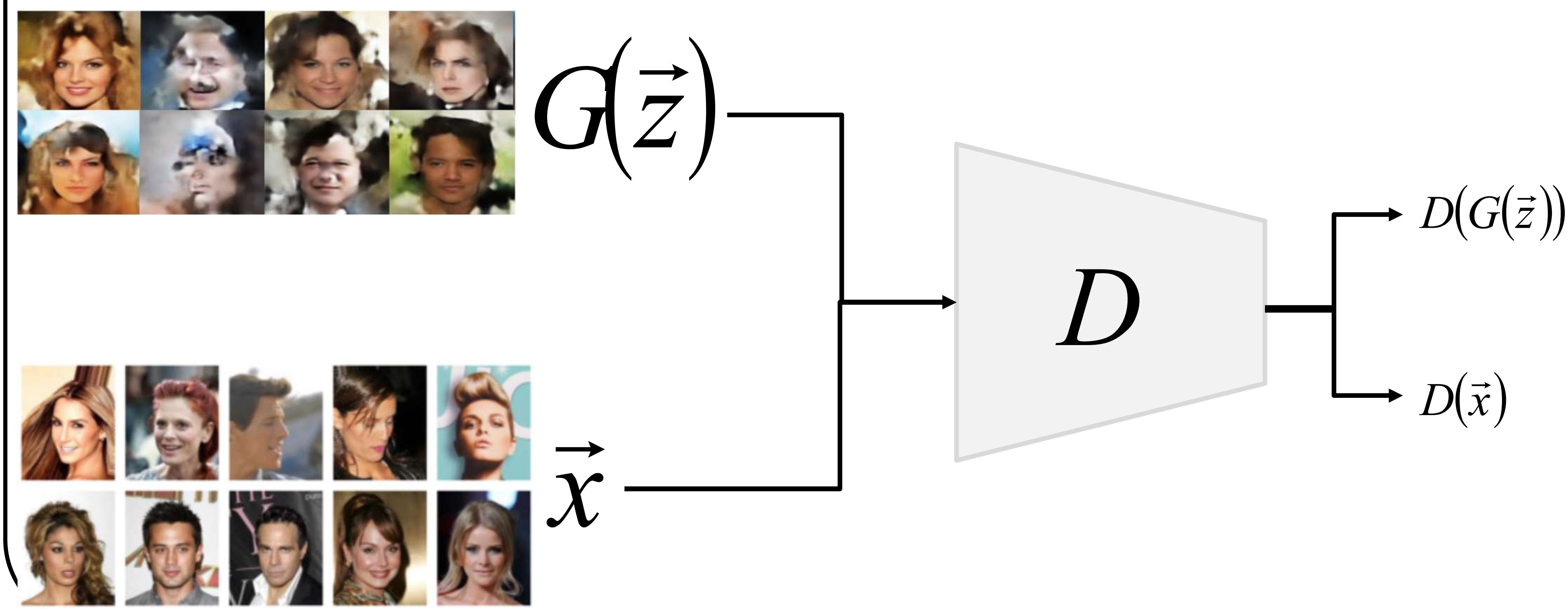
The **discriminator** network is represented by the **letter *D***

$$L_D = \frac{1}{N}\sum_i -t_i \ln\left(D\left(\vec{x}_i\right)\right) - \left(1 - t_i\right)\ln\left(1 - D\left(\vec{x}_i\right)\right)$$



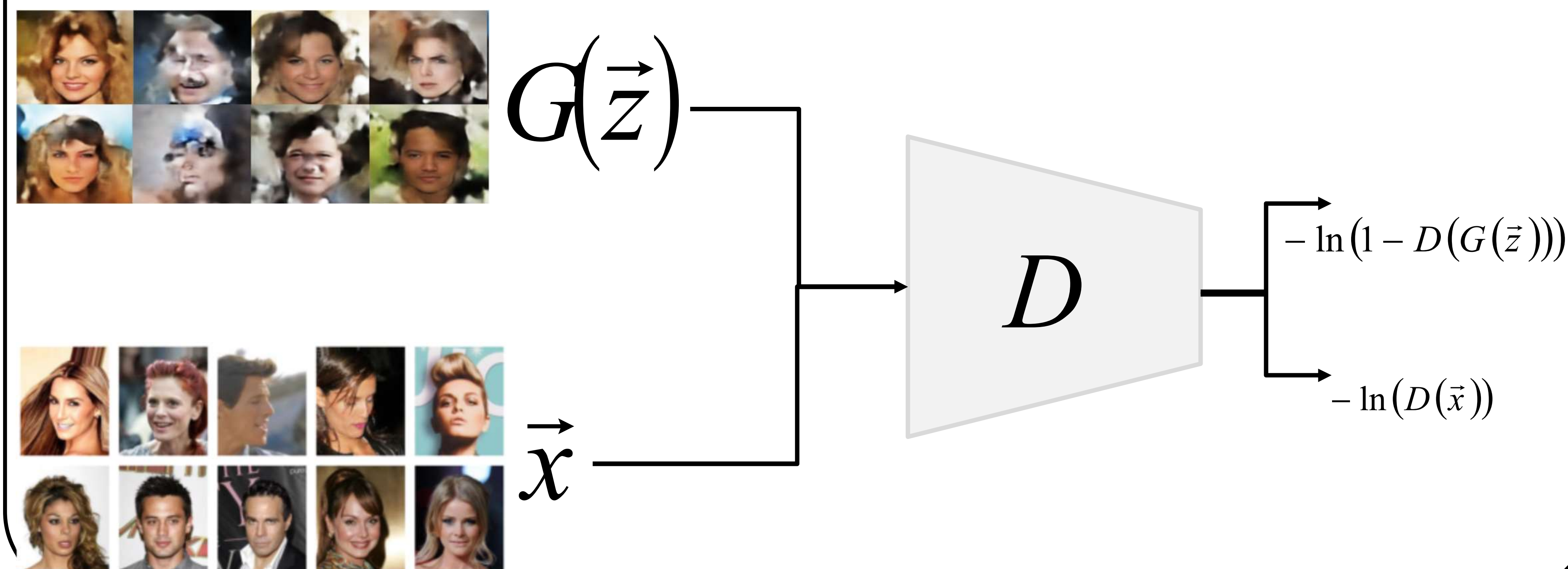
Since the **synthetic images** were generated by the **generator *G***

$$L_D = \frac{1}{N}\sum_i -t_i \ln\left(D\left(\vec{x}_i\right)\right) - \left(1 - t_i\right)\ln\left(1 - D\left(G\left(\vec{z}_i\right)\right)\right)$$



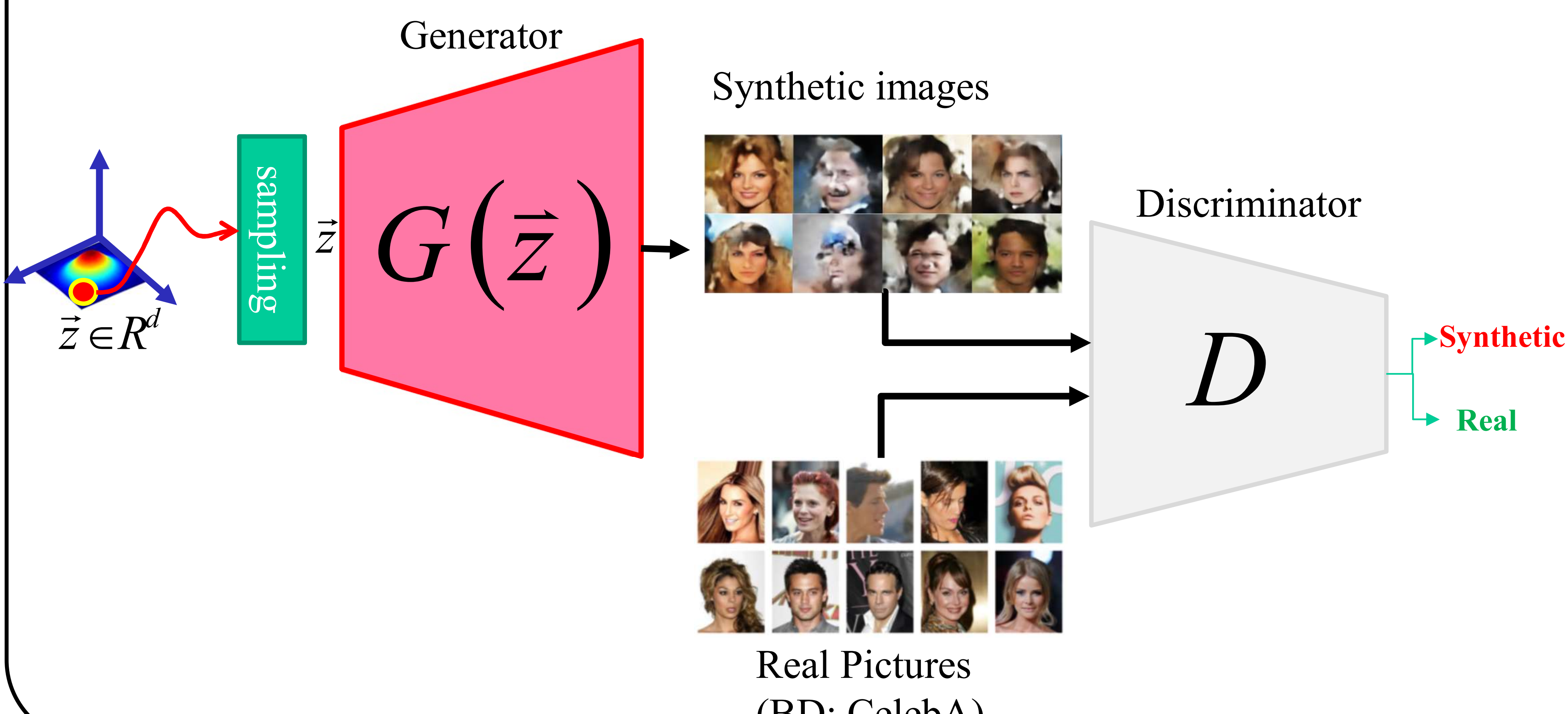
We can separate the loss from real and synthetic images

$$L_D = \underbrace{-\frac{1}{N_{reel}} \sum_i \ln\left(D\left(\vec{x}_i\right)\right)}_{\text{Loss of real frames}} \underbrace{- \frac{1}{N_{syn}} \sum_j \ln\left(1 - D\left(G\left(\vec{z}_j\right)\right)\right)}_{\text{Loss of synthetic images}}$$



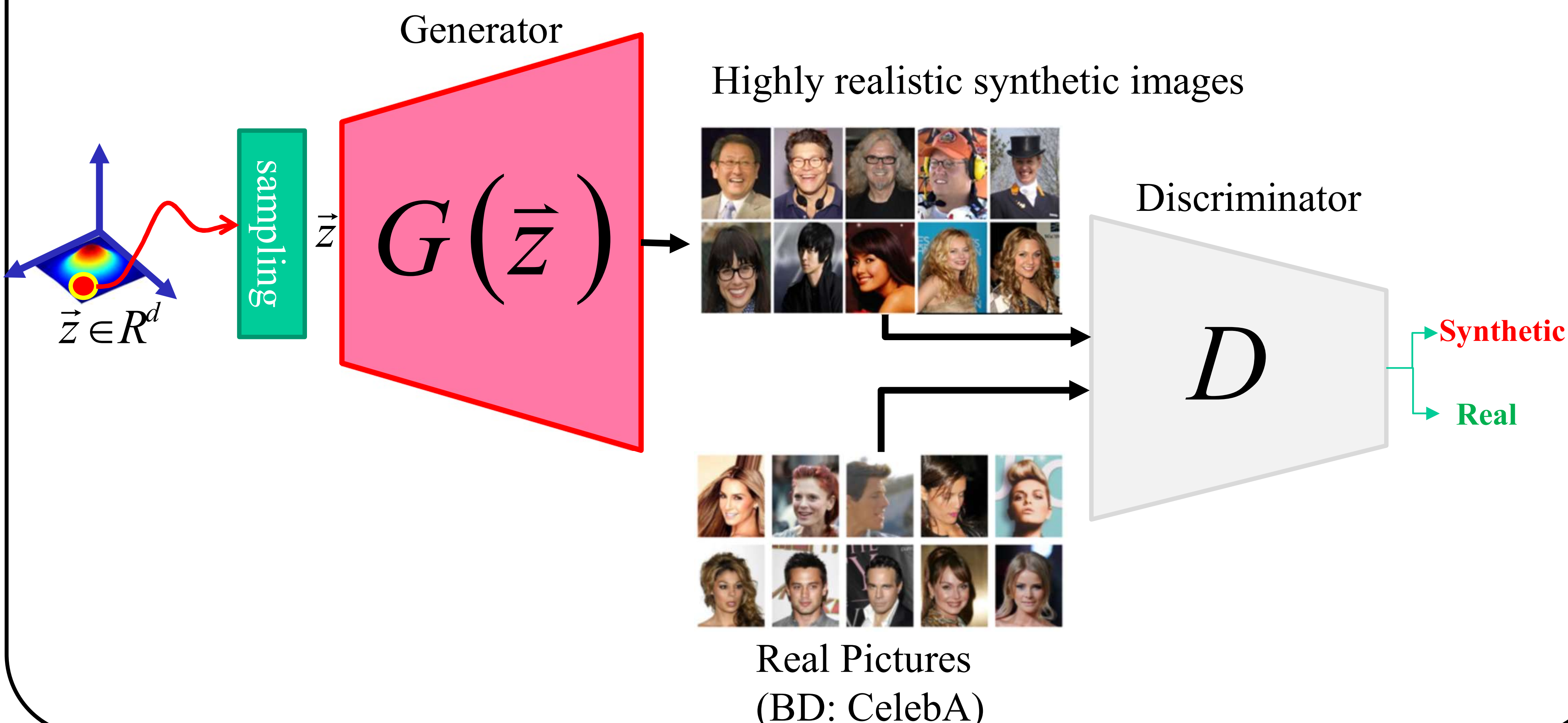
Purpose of the generator

Produce images as realistic as that of the reference comic book



Purpose of the generator

If G succeeds, the discriminator will no longer be able to distinguish between them real images. The discriminator loss will then be high.



« Two player » min-max game

$$\min_G \max_D V(D, G) = \mathbb{E}_{\boldsymbol{x} \sim p_{\text{data}}(\boldsymbol{x})}[\log D(\boldsymbol{x})] + \mathbb{E}_{\boldsymbol{z} \sim p_{\boldsymbol{z}}(\boldsymbol{z})}[\log(1 - D(G(\boldsymbol{z})))].$$

Ian Goodfellow et al., “Generative Adversarial Nets”, NIPS 2014

« Two player » min-max game

$$\min_G \max_D V(D,G) = \mathbb{E}_{\boldsymbol{x}\sim p_{\text{data}}(\boldsymbol{x})}\overbrace{[\log D(\boldsymbol{x})]}^{\substack{\text{Discriminator wants} \\ \text{D(x) = 1 for true data}}} + \mathbb{E}_{\boldsymbol{z}\sim p_{\boldsymbol{z}}(\boldsymbol{z})}[\log(1 - \overbrace{\underbrace{D(G(\boldsymbol{z}))}_{\substack{\text{Generator wants} \\ \text{D(G(x)) = 1 for} \\ \text{synthetic data}}}}^{\substack{\text{Discriminator wants} \\ \text{D(G(x)) = 0 for} \\ \text{synthetic data}}})].$$

Ian Goodfellow et al., “Generative Adversarial Nets”, NIPS 2014

Ian Goodfellow et al., “Generative Adversarial Nets”, NIPS 2014

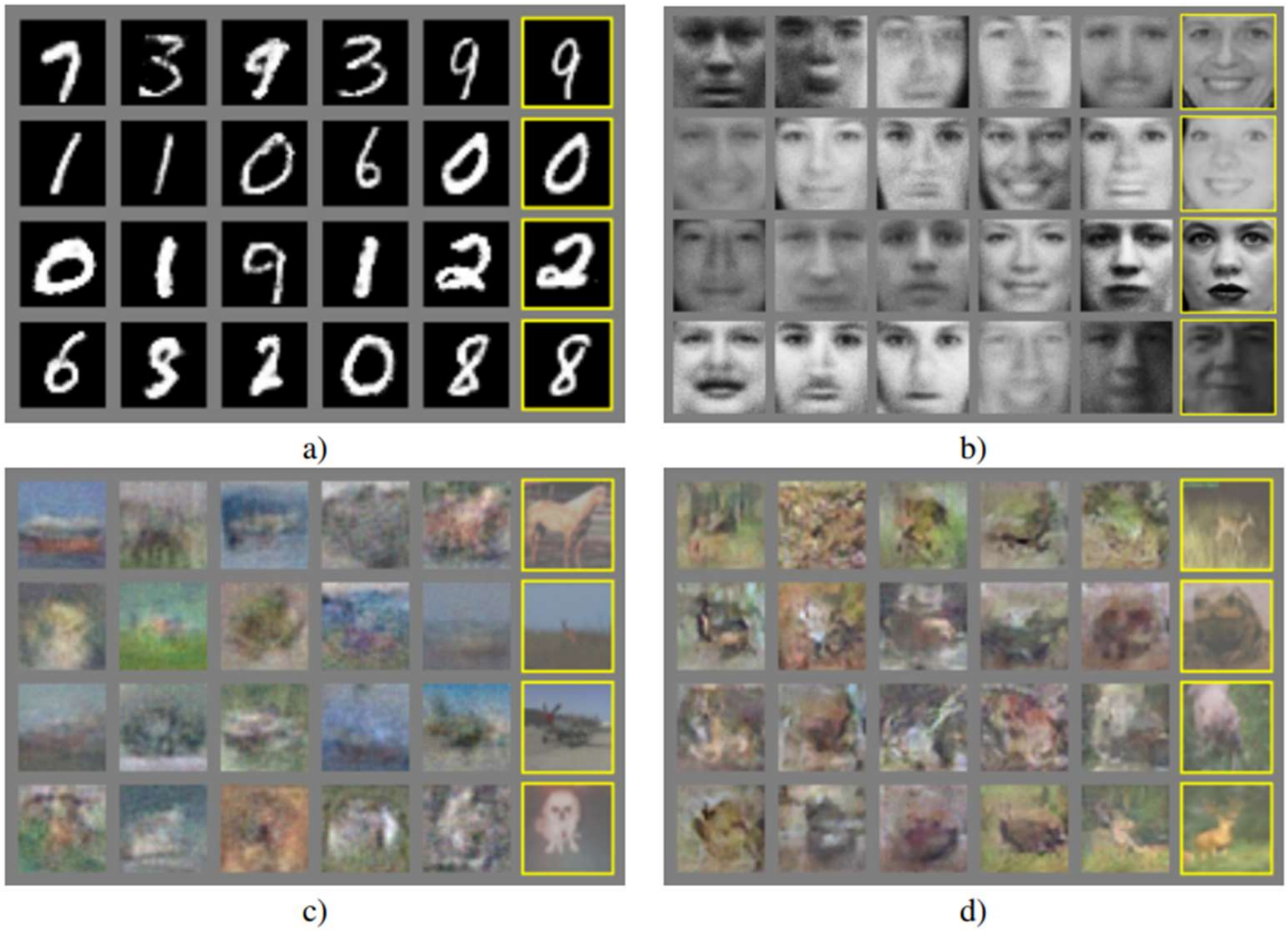
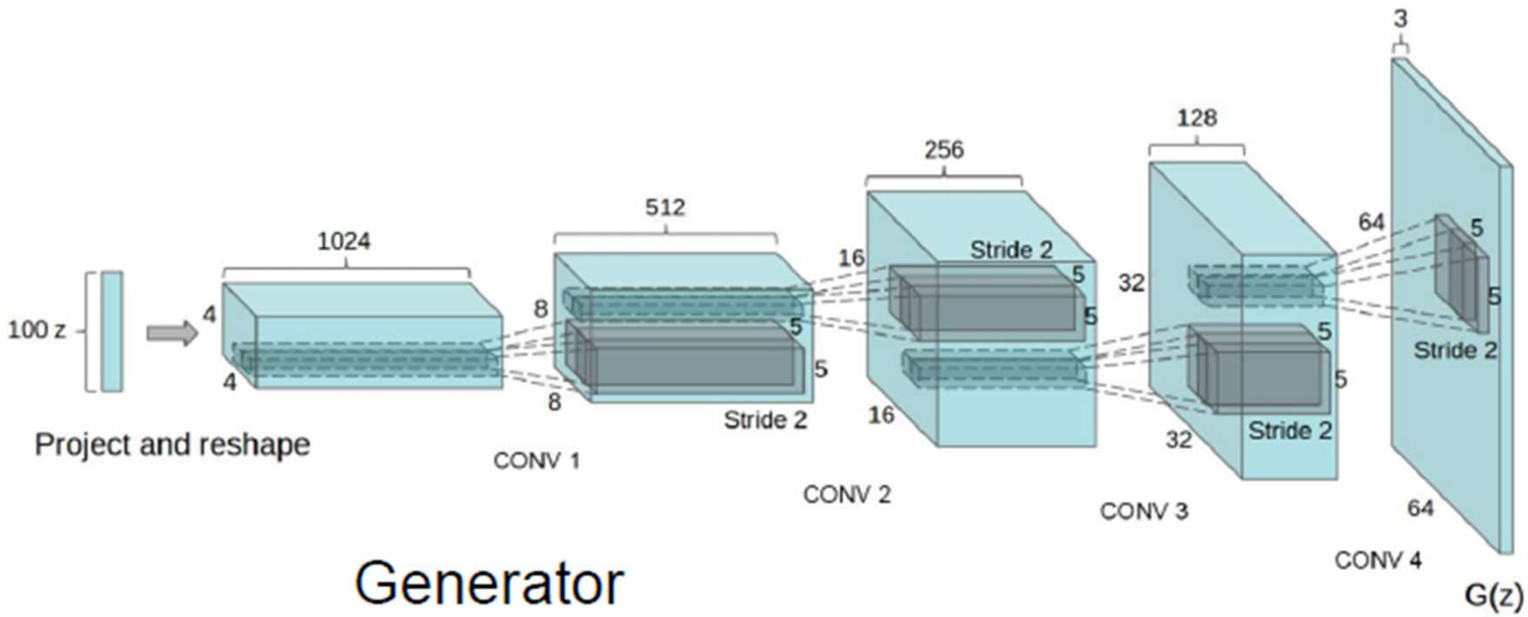


Figure 2: Visualization of samples from the model. Rightmost column shows the nearest training example of the neighboring sample, in order to demonstrate that the model has not memorized the training set. Samples are fair random draws, not cherry-picked. Unlike most other visualizations of deep generative models, these images show actual samples from the model distributions, not conditional means given samples of hidden units. Moreover, these samples are uncorrelated because the sampling process does not depend on Markov chain mixing. a) MNIST b) TFD c) CIFAR-10 (fully connected model) d) CIFAR-10 (convolutional discriminator and “deconvolutional” generator)

Deep Convolution Generative Adversarial Net (DCGAN)



Radford et al, "Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks", ICLR 2016

Deep Convolution Generative Adversarial Net (DCGAN)

Discriminator recommendations

- Conv stride>1 instead of pooling layers
- ReLU everywhere except at the output: tanh

Generator Recommendations

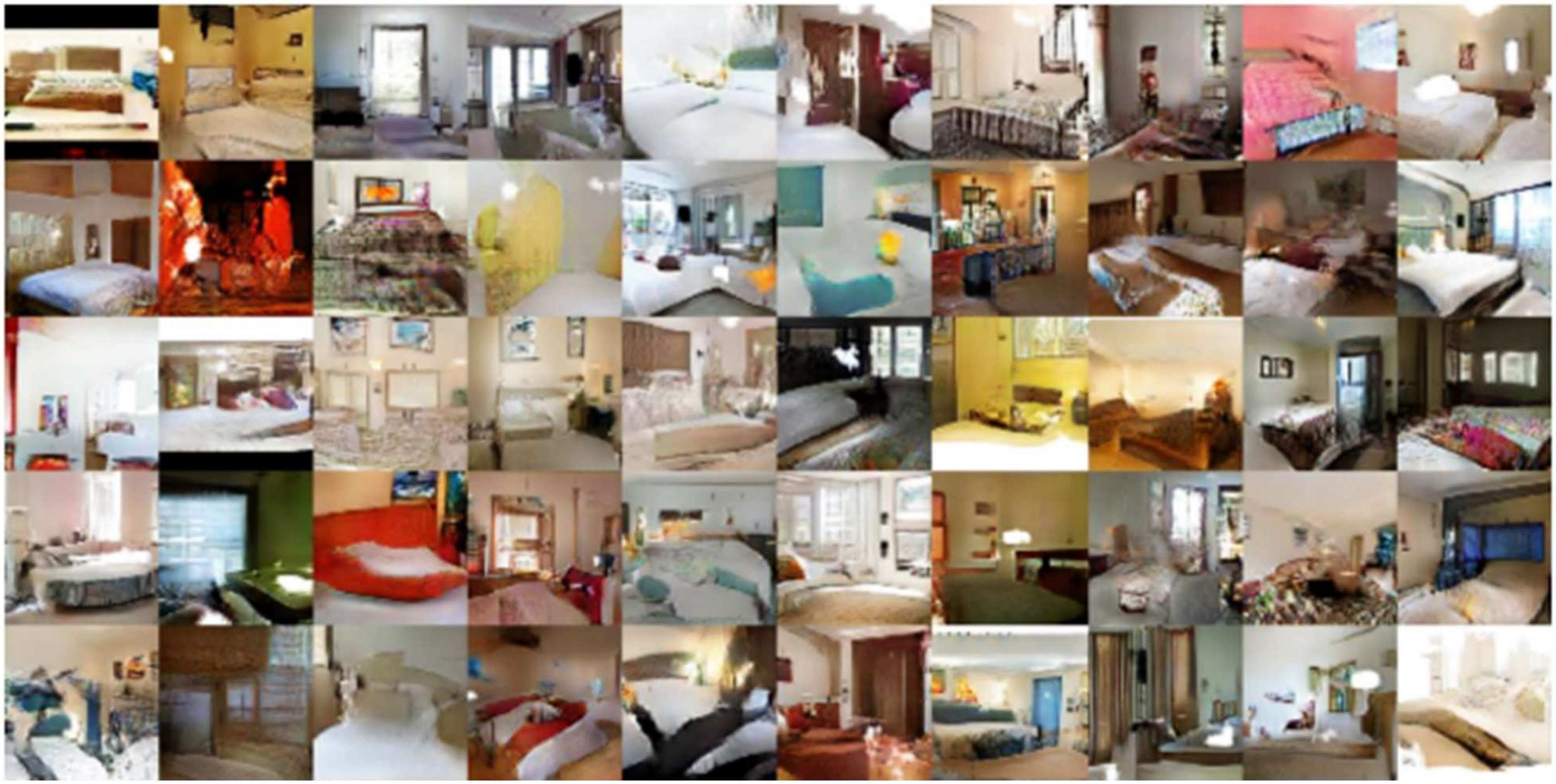
- Conv transpose instead of upsampling
- LeakyReLU everywhere

Other recommendations

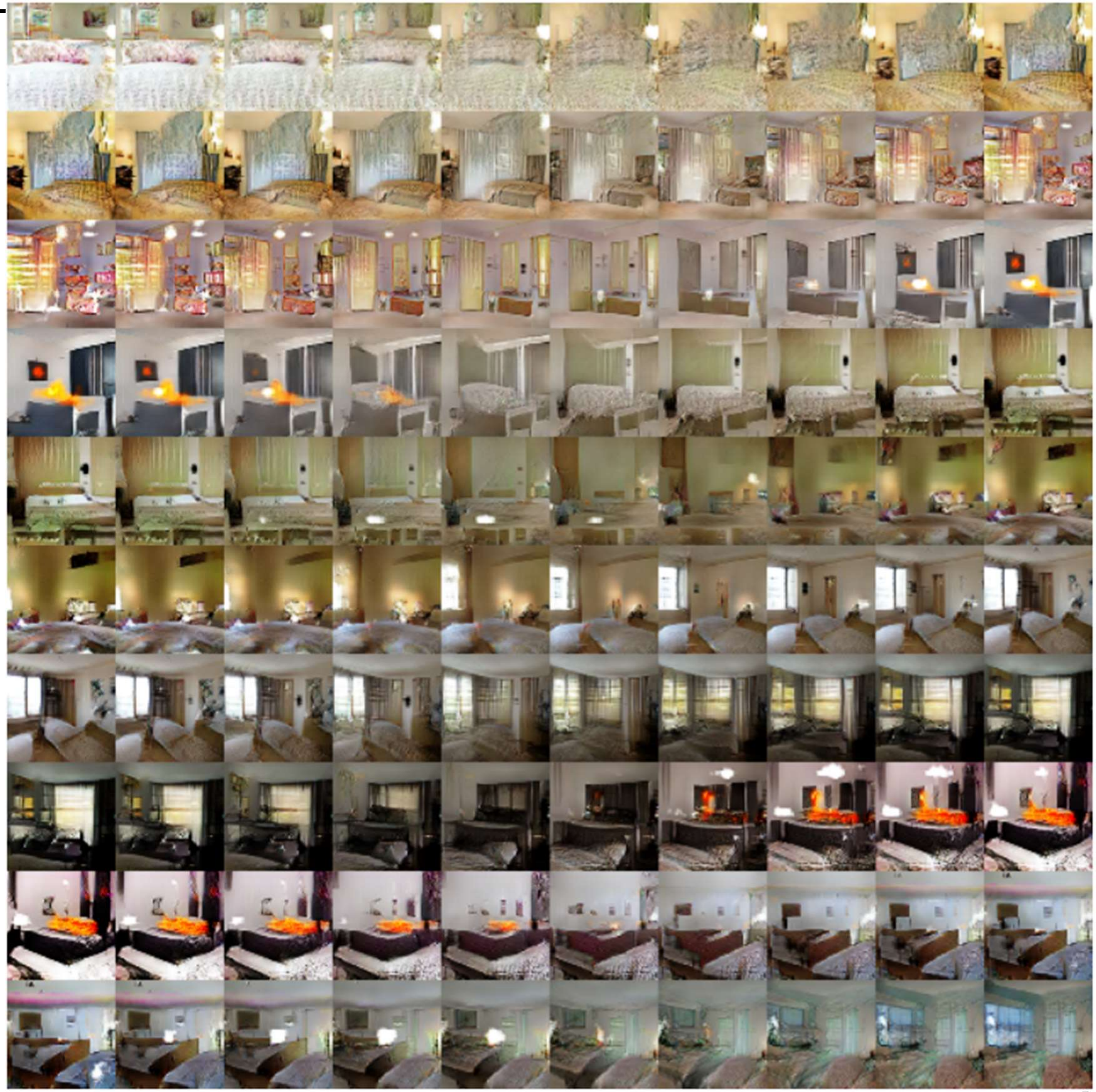
- BatchNorm Everywhere
- No dense layers, just conv

Radford et al, "Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks", ICLR 2016

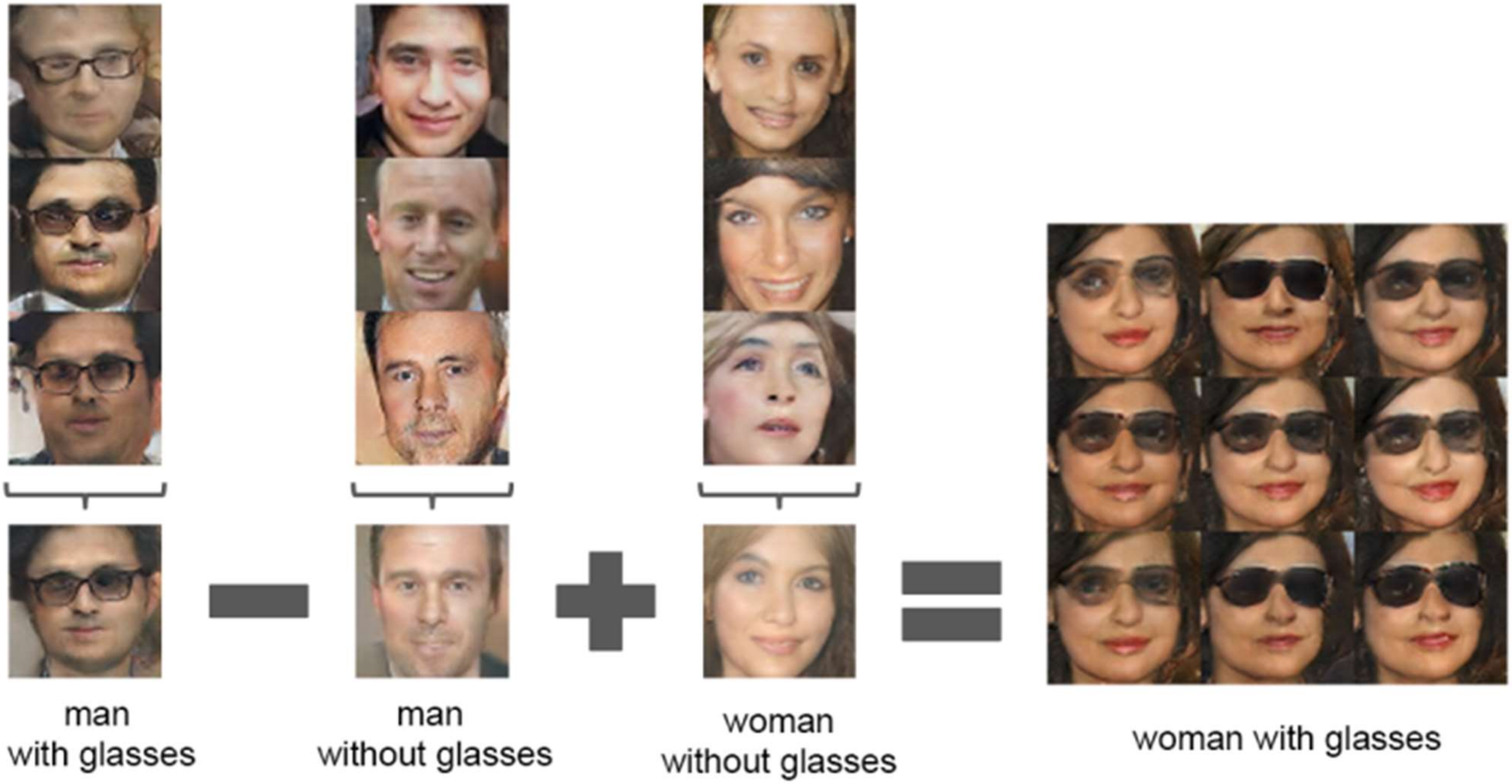
Deep Convolution Generative Adversarial Net (DCGAN)



Interpolation between 9 latent vectors



"Vector arithmetic for visual concepts"



Instability issues

- If discriminator and generator and do not learn together:
  - Vanishing gradients
  - Mode collapse
  - high-resolution images cannot be generated

- Several solutions proposed:
  - *Wasserstein GAN (uses "earth mover distance")*
  - *Least Squares GAN (uses quadratic error distance)*
  - *Progressive GAN….*

Instability issues

- If discriminator and generator and do not learn together:
  - **Vanishing gradients**
  - Mode Collapse
  - Can’t generate high-resolution images
- Plusieurs solutions proposées:

If the discriminator learns too quickly, the generator will be systematically beaten, and will not learn anything

Instability issues

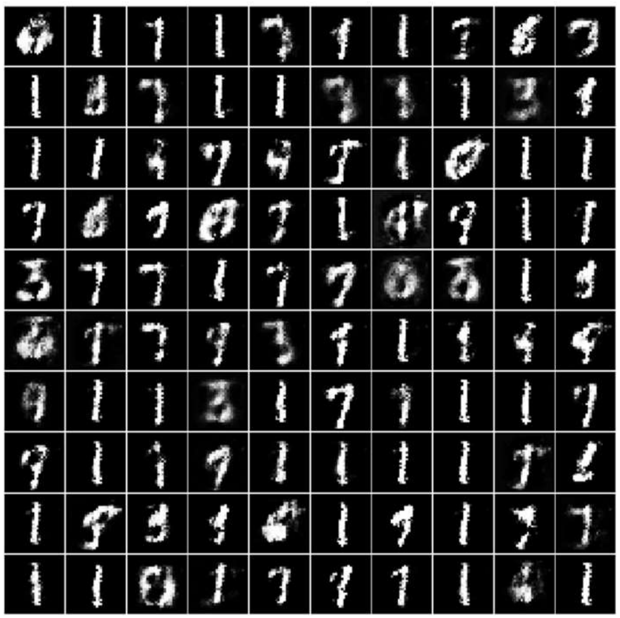
- If discriminator and generator and do not learn together:
  - Vanishing gradients
  - **Mode Collapse**
  - Can’t generate high-resolution images

The generator can learn to generate the same image over and over again and thus beat the discriminator

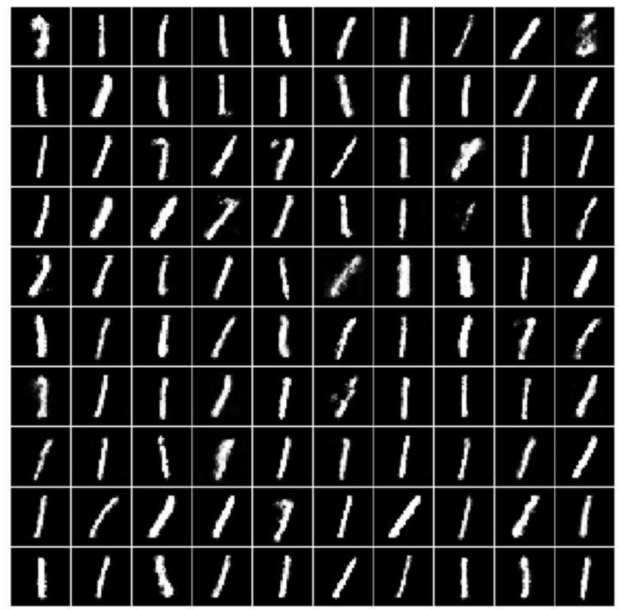
Instability issues

- If discriminator and generator and do not learn together:

Epoch 7



Epoch 21



The ge
thus be

and

https://datascience.stackexchange.com/questions/29485/gan-discriminator-converging-to-one-output

Instability issues

- If discriminator and generator and do not learn together:
  - Vanishing gradients
  - Mode Collapse
  - **Can't generate high-resolution images**

High resolution images are off and blury

GAN zoo

- WGANs (Wasserstein)
- SRGAN (Super resolution)
- cGAN (Conditional)
- CycleGAN
- Pix2Pix
- StackGAN
- ProGAN
- StyleGAN
- VQGAN (Vector Quantization)
- …

GAN zoo

- WGANs (Wasserstein)
- SRGAN (Super resolution)
- cGA
- Cyc
- Pix2
- Stac
- Pro
- StyleGAN
- VQGAN (Vector Quantization)
- …

500 models only here!

github.com/hindupuravinash/the-gan-zoo

Applications

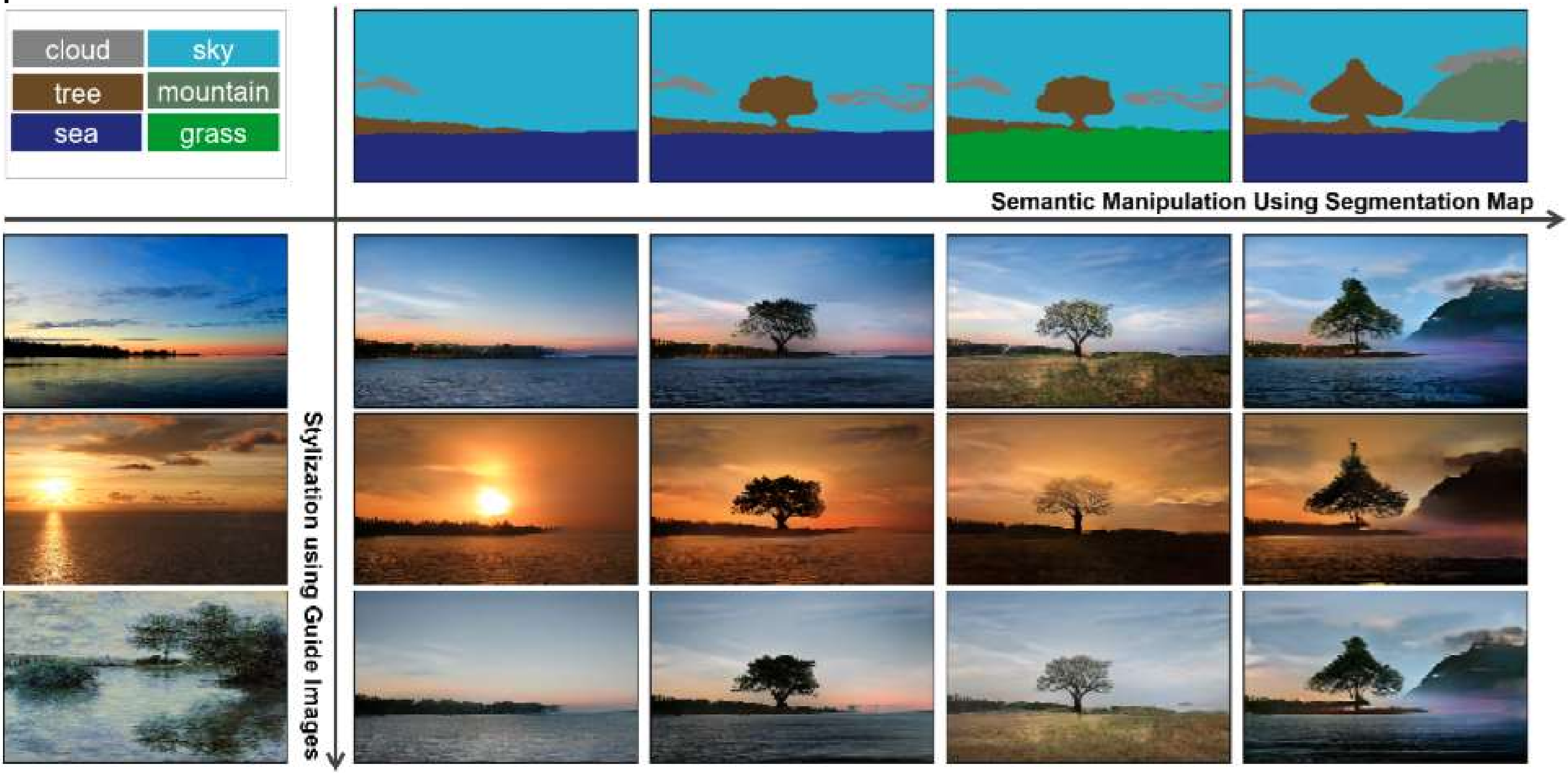
Face generation

StyleGAN

Applications

Plaired image-to-image generation

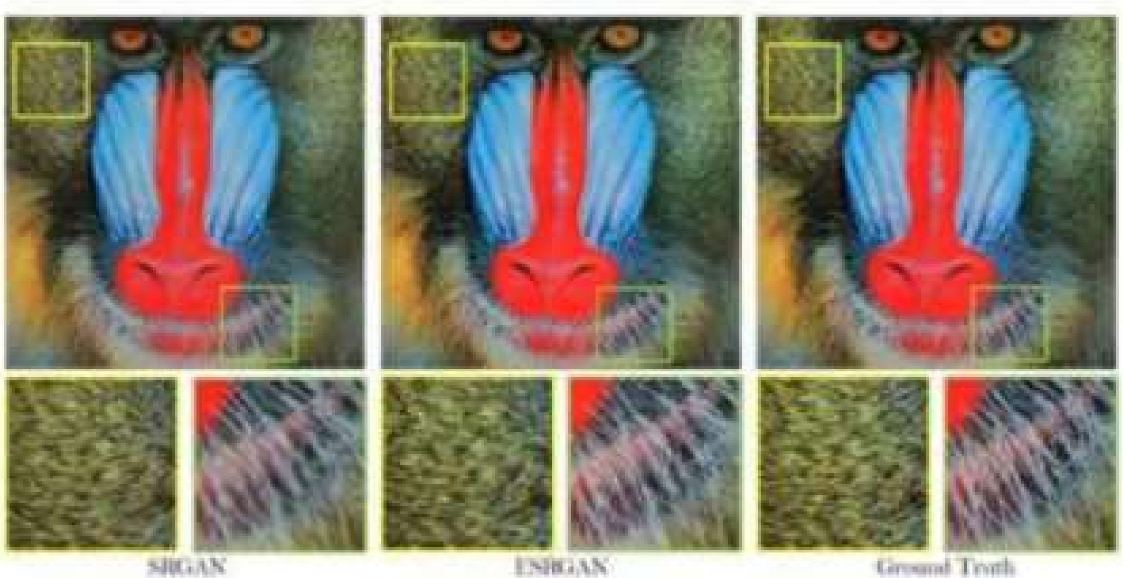
SpadeGAN



Applications

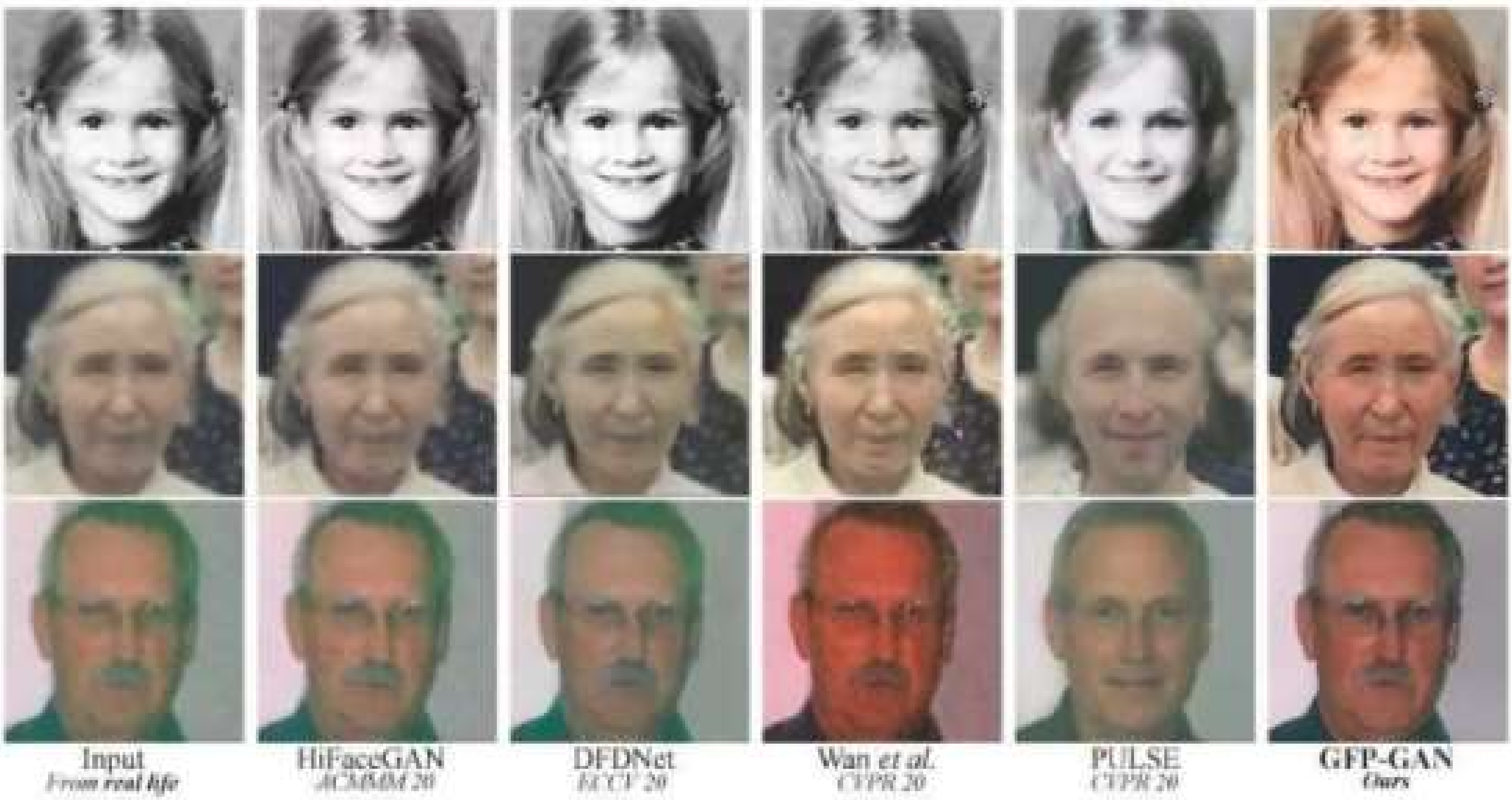
Super Resolution

upsampling a low-resolution image into a higher resolution



Applications

Restoration of old images and noise removal



https://www.slideshare.net/slideshow/generative-adversarial-network-gans/266847545#22

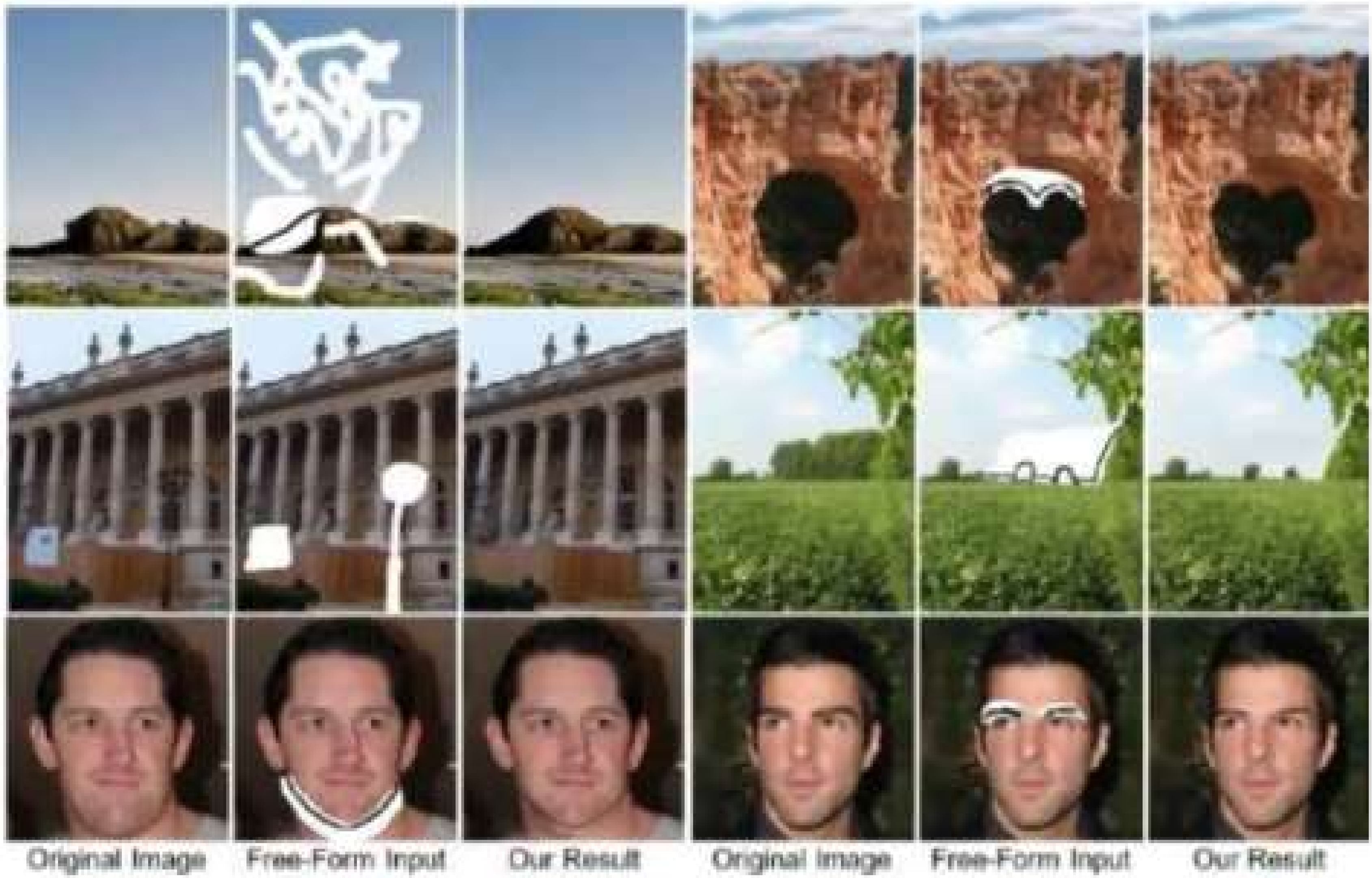
Applications

Text-to-Image Generation (text2image)

Generating Synthetic Images from textual description

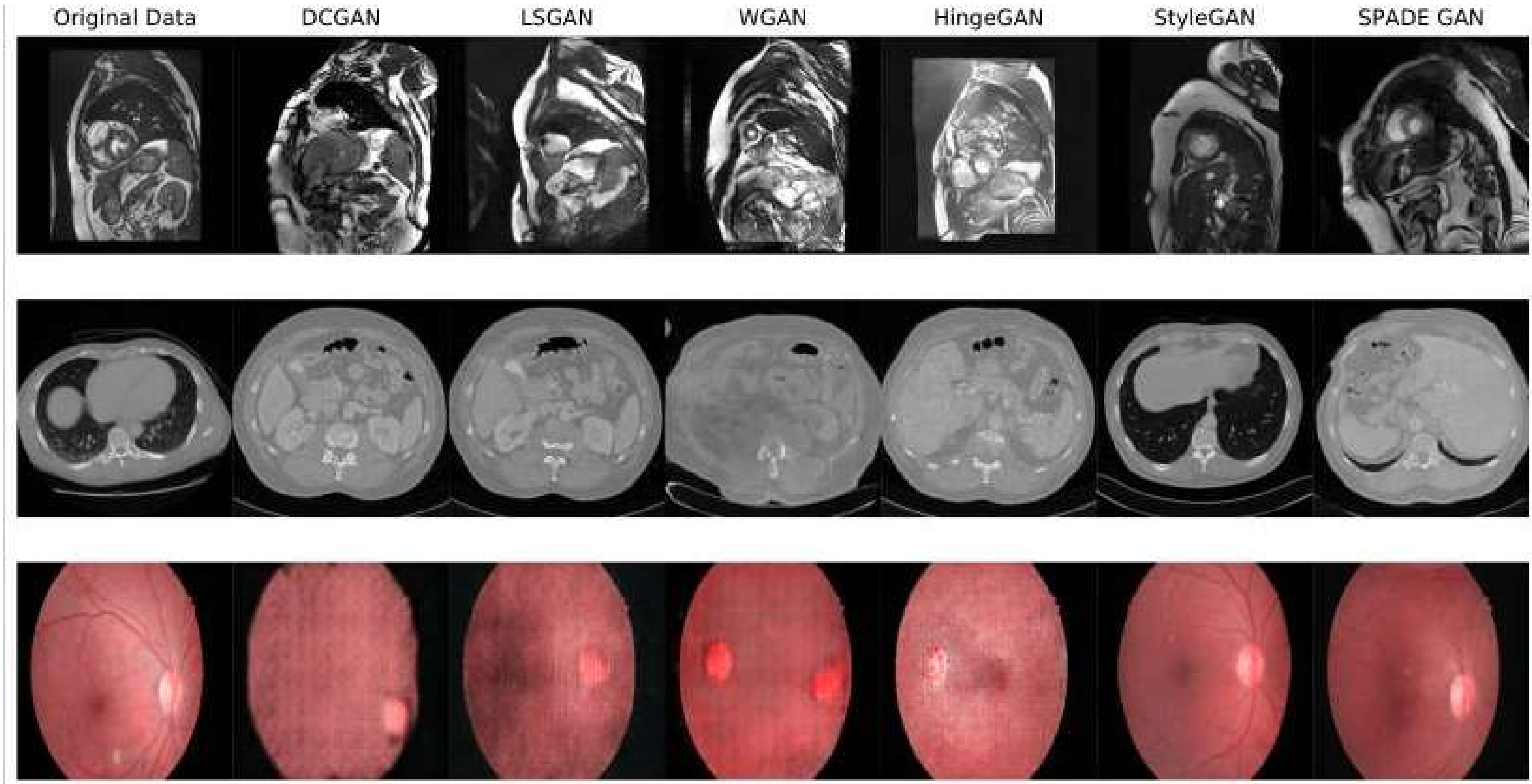
Applications

Fill missing parts of the image (inpainting)



Medical Applications

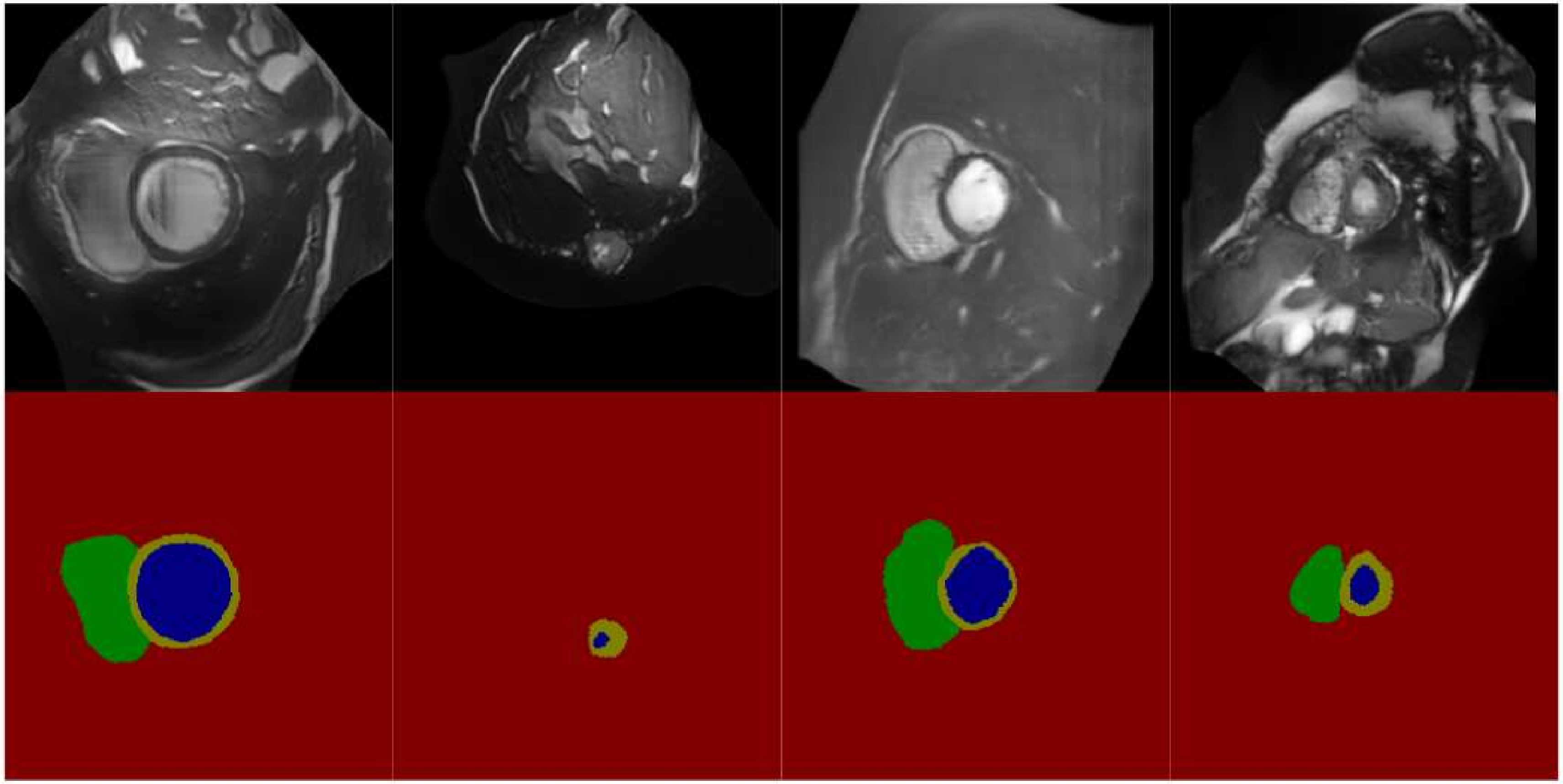
Highly-realistic image generation



Youssef Skandarani, Pierre-Marc Jodoin, Alain Lalande, GANs for Medical Image Synthesis: An Empirical Study, 2023 Mar 16;9(3):69.

Medical Applications

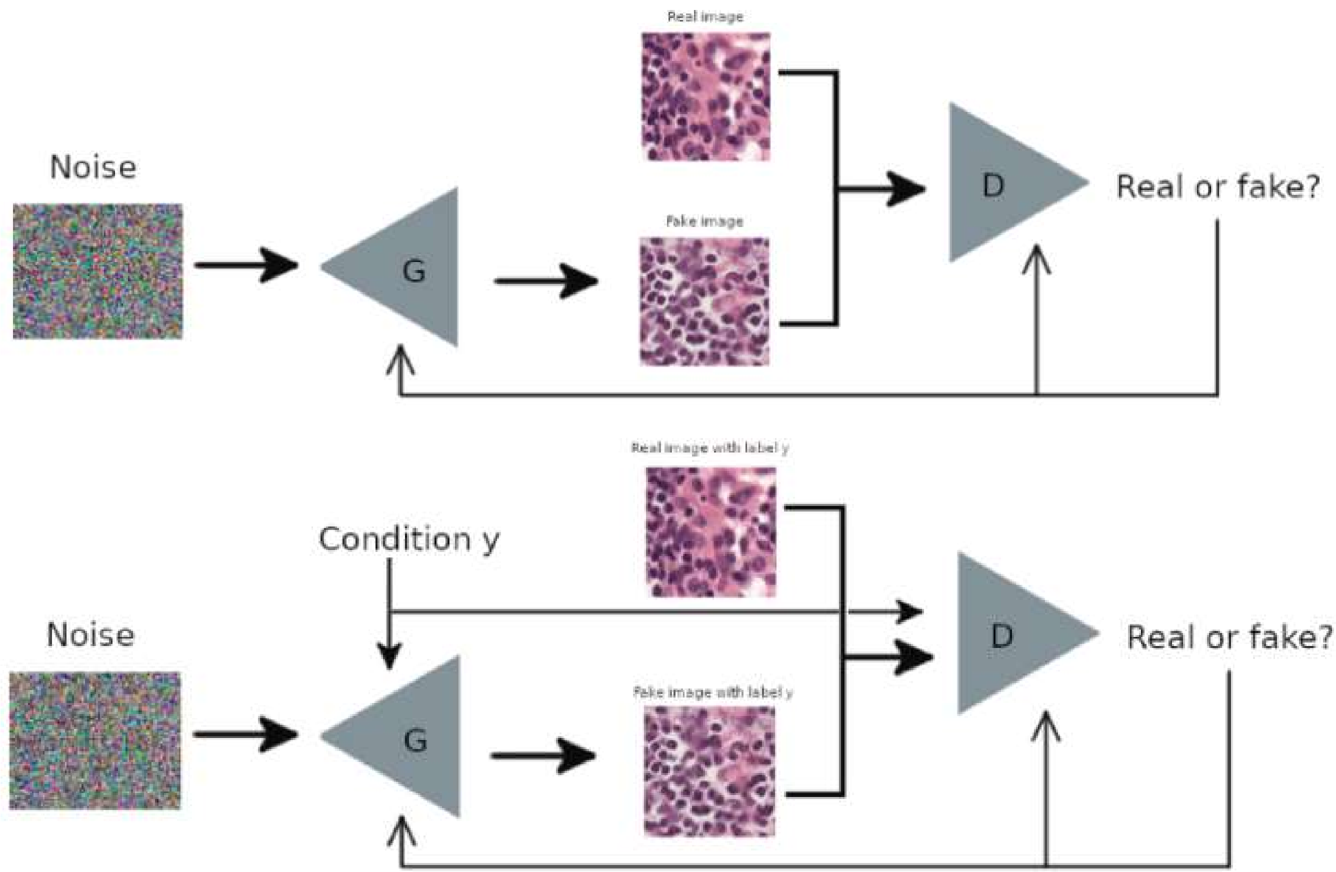
Generate annotated images



Youssef Skandarani, Pierre-Marc Jodoin, Alain Lalande, GANs for Medical Image Synthesis: An Empirical Study, 2023 Mar 16;9(3):69.

Medical applications

condGAN, histopathological images



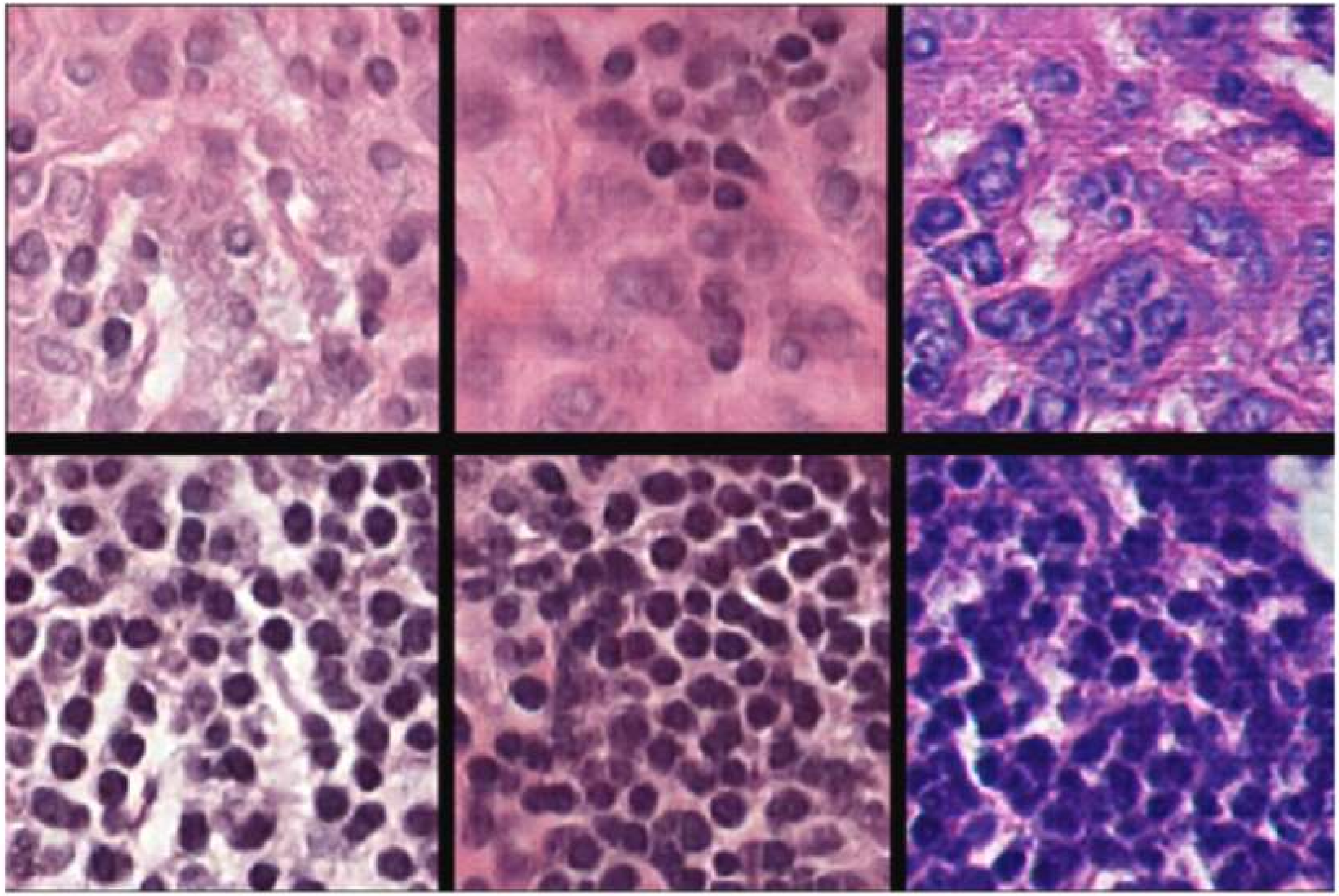
Rubén Fernández Blanco, Pilar Rosado, Esteban Vegas, Ferran Reverter,
Medical image editing in the latent space of Generative Adversarial Networks, Intelligence-Based Medicine, Vol 5, 2021,

Medical applications

Conditional GAN

Cancer

No-Cancer



Rubén Fernández Blanco, Pilar Rosado, Esteban Vegas, Ferran Reverter,
Medical image editing in the latent space of Generative Adversarial Networks, Intelligence-Based Medicine, Vol 5, 2021,

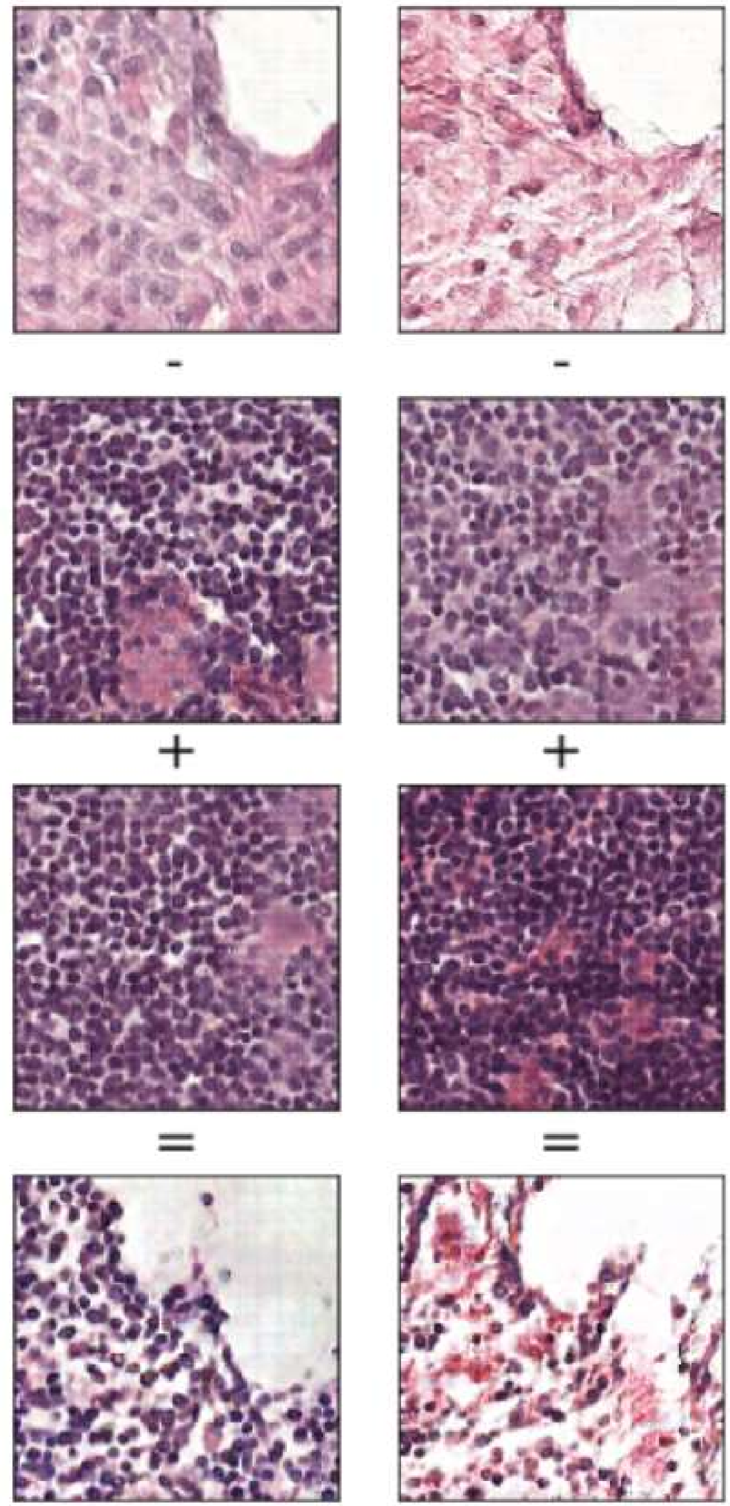
Vector Arithmetics

Cancerous samples
with visible lumen

Cancerous samples
without lumen

Normal samples
without lumen.

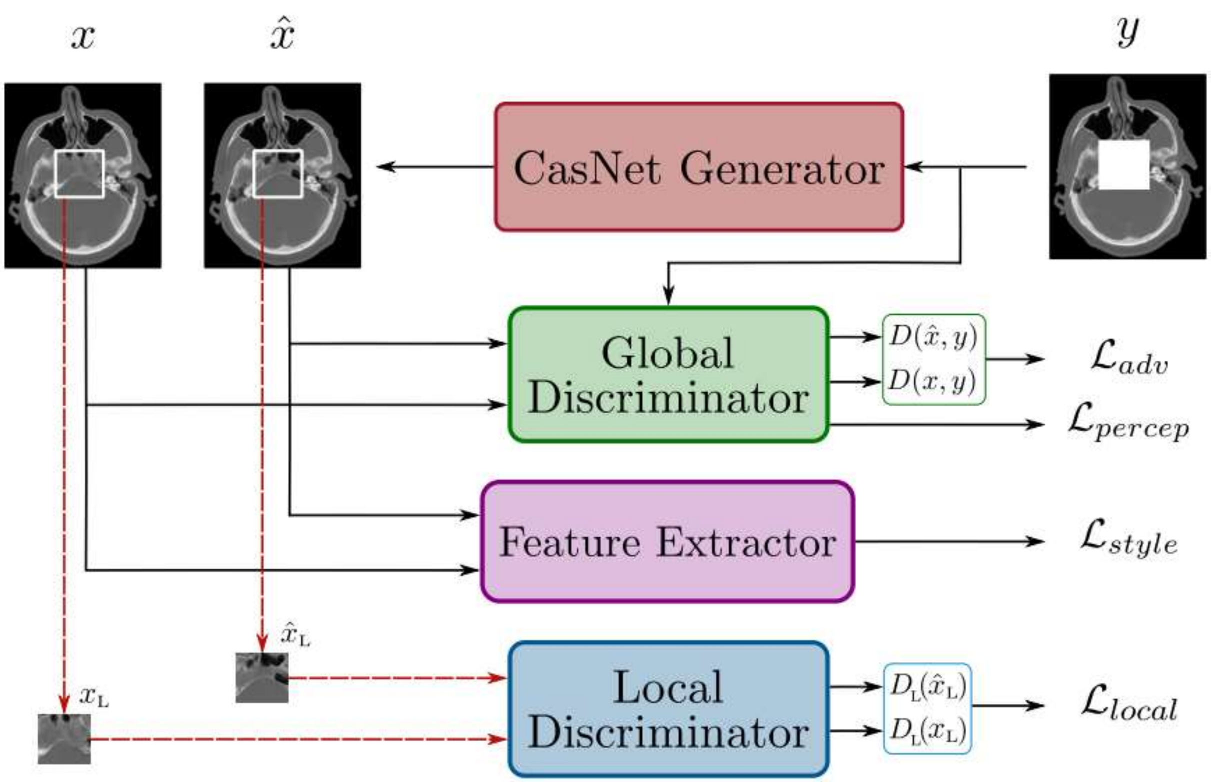
Normal samples
with lumen



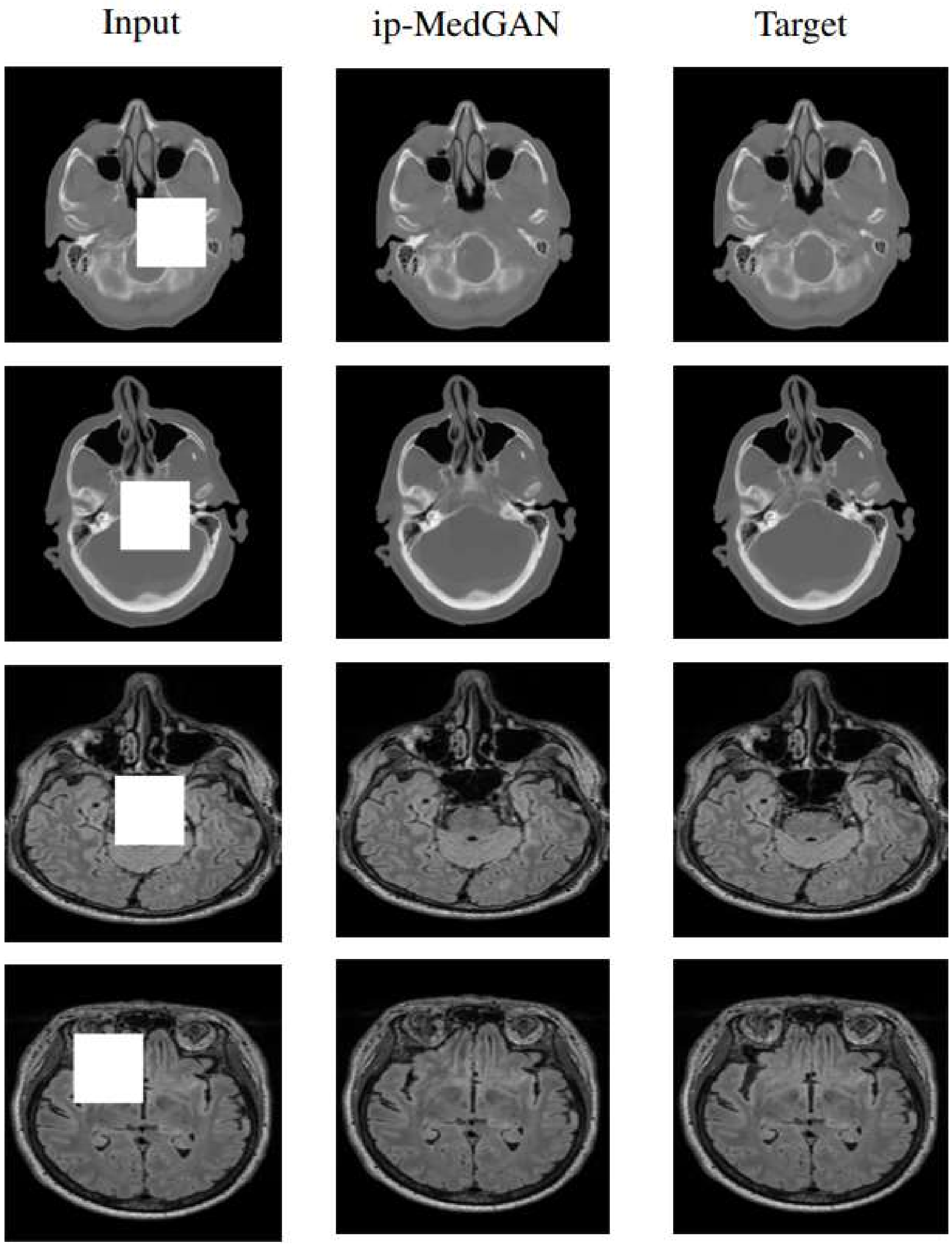
Rubén Fernández Blanco, Pilar Rosado, Esteban Vegas, Ferran Reverter,
Medical image editing in the latent space of Generative Adversarial Networks,

Medical applications

Inpainting (i.e. remove a tumor or a metallic implant)



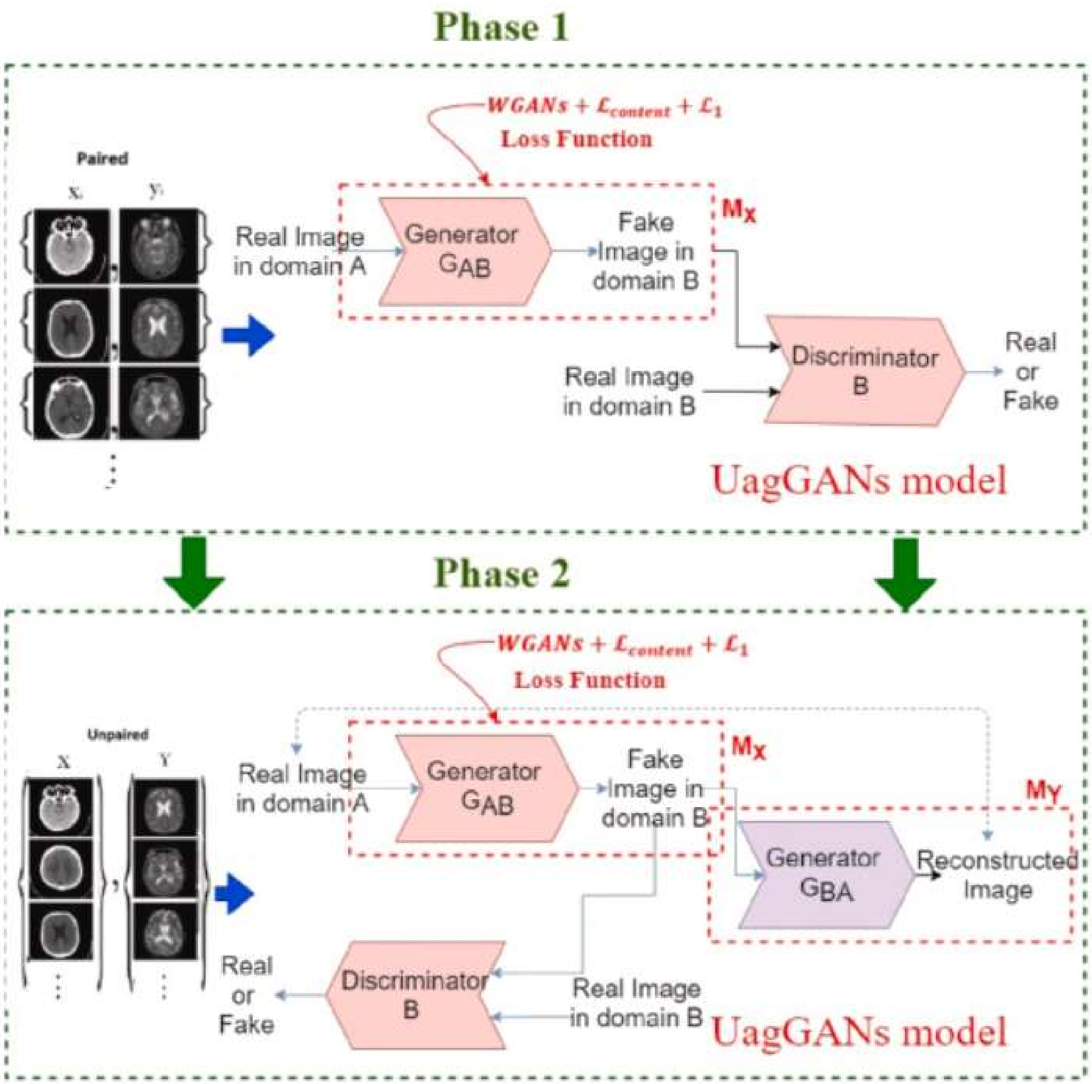
Rubén Fernández Blanco, Pilar Rosado, Esteban Vegas, Ferran Reverter,
Medical image editing in the latent space of Generative Adversarial Networks, Intelligence-Based Medicine, Vol 5, 2021,



Karim Armanious; Youssef Mecky; Sergios Gatidis; Bin Yang Adversarial Inpainting of Medical Image Modalities
IEEE ICASSP, 2019

Medical Applications

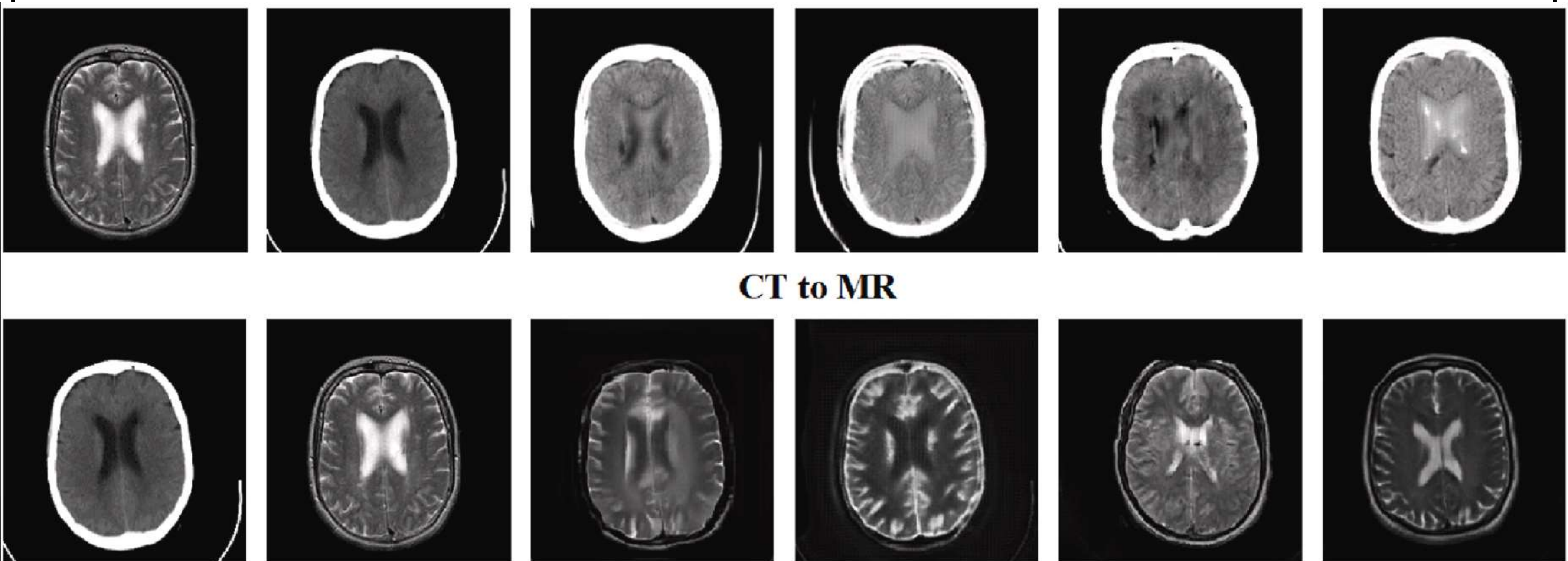
Modality translation



Alaa A. et al. *Paired-Unpaired Unsupervised Attention Guided GAN with Transfer Learning for Bidirectional Brain MR-CT Synthesis*, Computers in Biology and Medicine 136(2):104763

Medical Applications

Modality translation

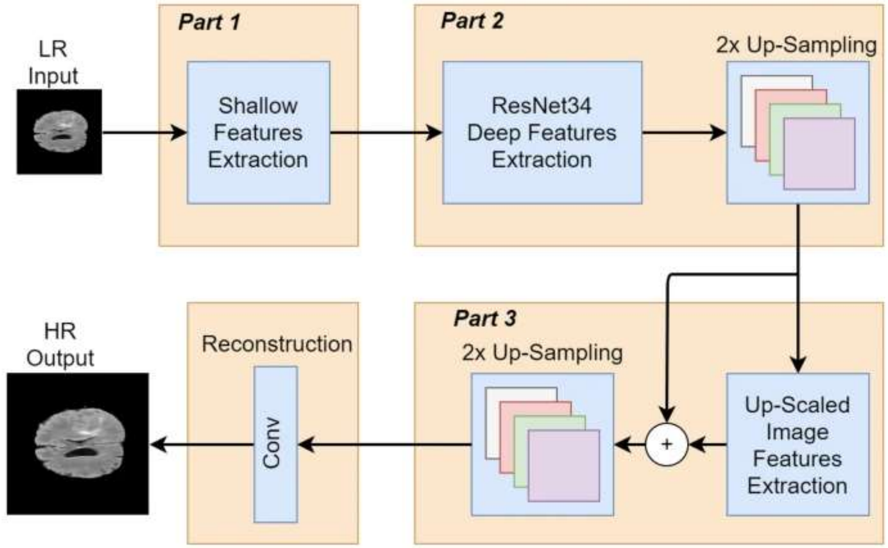


Real ground truth cycleGAN dualGAN discoGAN uagGAN

Alaa A. et al. *Paired-Unpaired Unsupervised Attention Guided GAN with Transfer Learning for Bidirectional Brain MR-CT Synthesis*, Computers in Biology and Medicine 136(2):104763

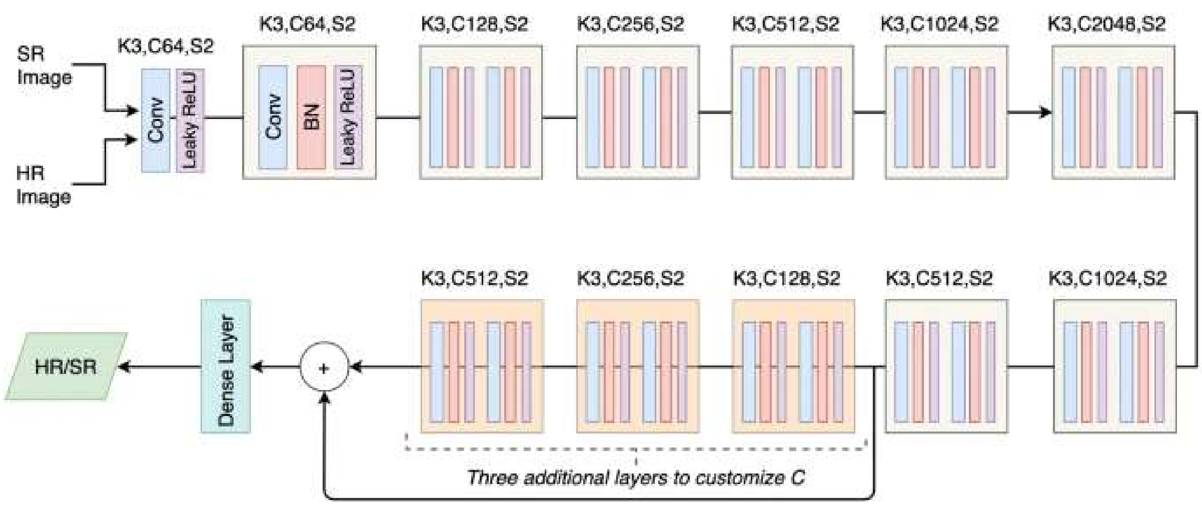
Medical Applications

Super resolution



An overall block diagram to show the two rounds of the features extraction and 2x upscaling.

Generator

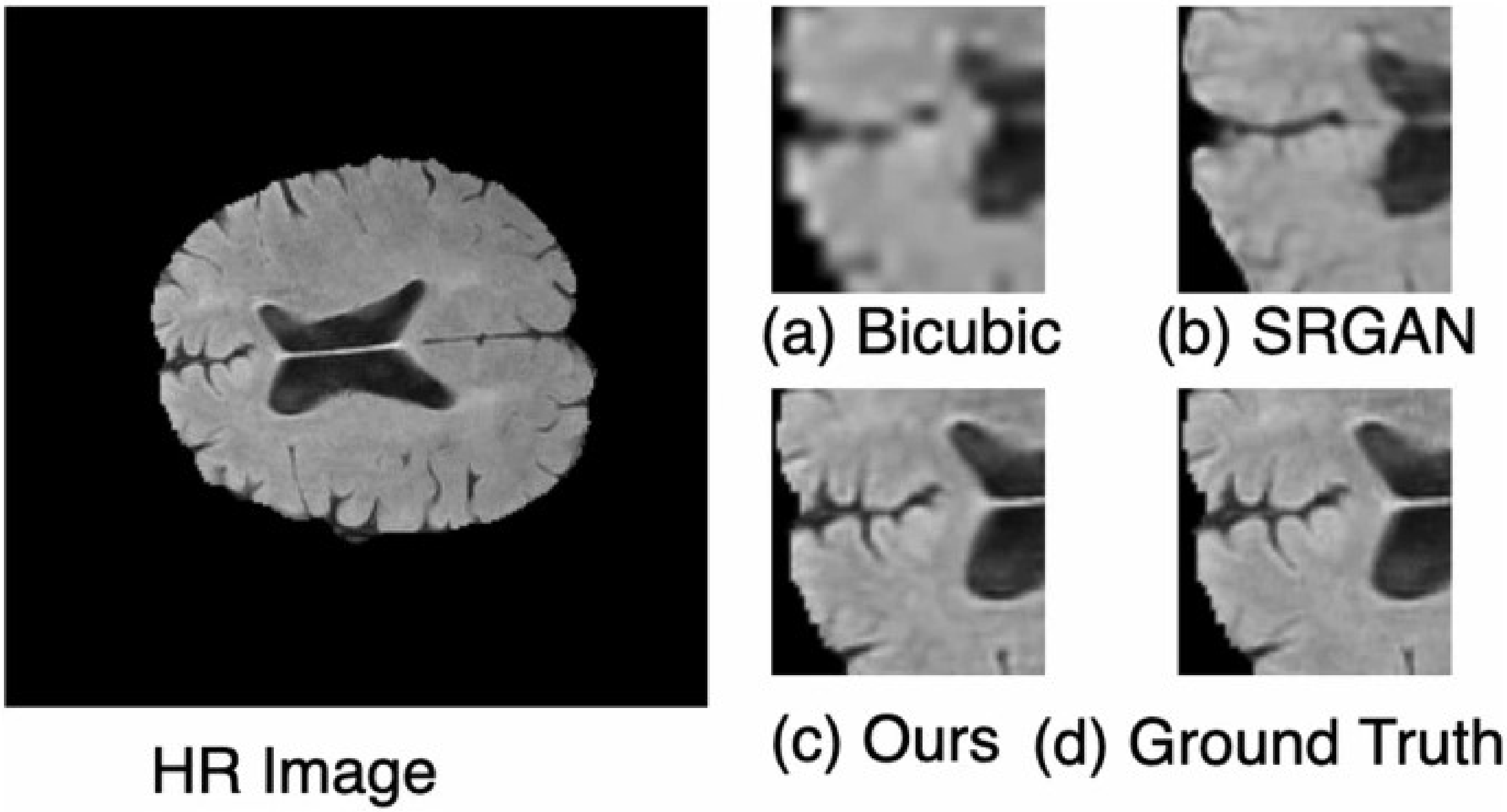


Discriminator

Waqar Ahmad, Hazrat Ali, Zubair Shah & Shoaib Azmat, *A new generative adversarial network for medical images super resolution,* Scientific Reports volume 12, Article number: 9533 (2022)

Medical Applications

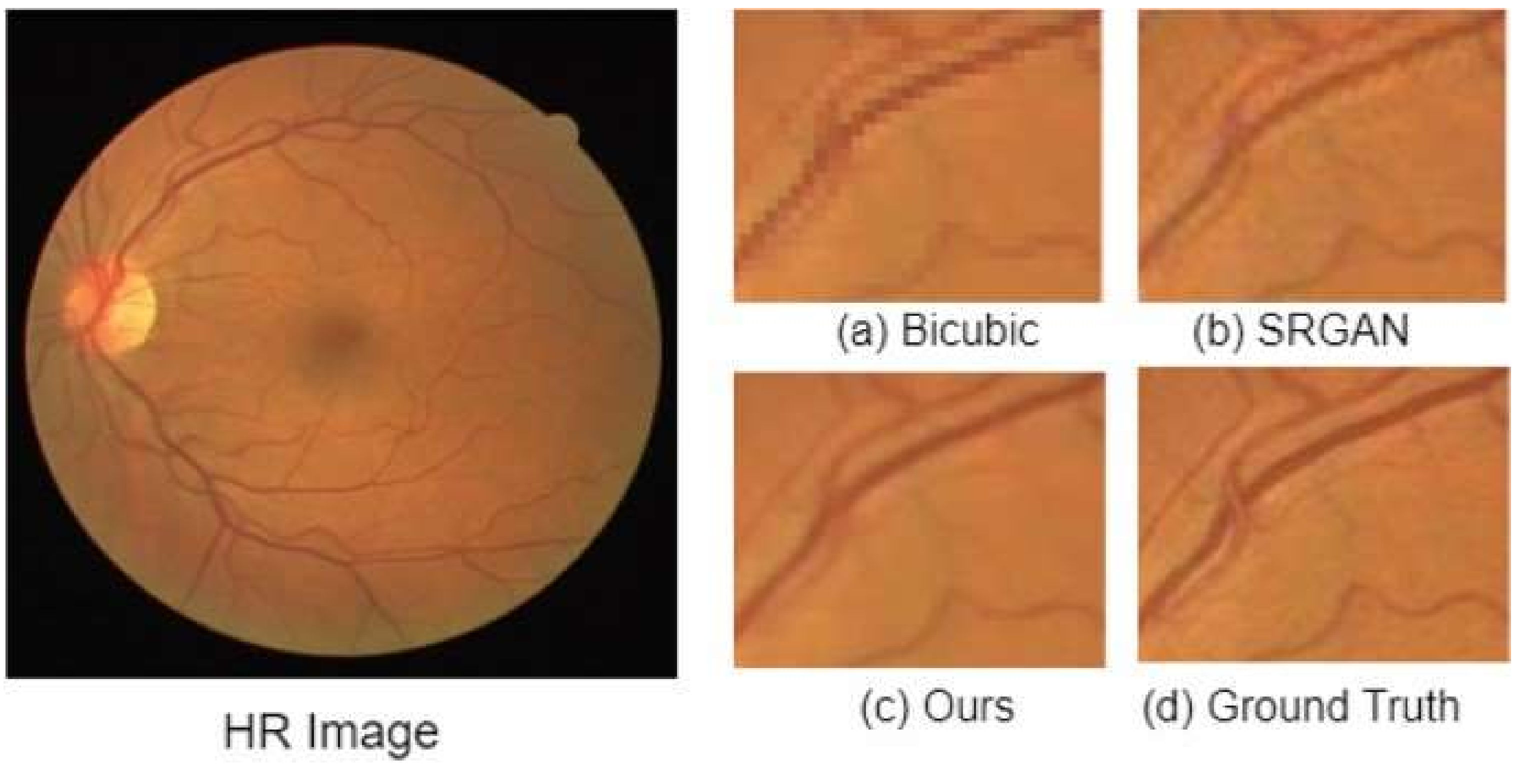
Super resolution



Waqar Ahmad, Hazrat Ali, Zubair Shah & Shoaib Azmat, *A new generative adversarial network for medical images super resolution,* Scientific Reports volume 12, Article number: 9533 (2022)

Medical Applications

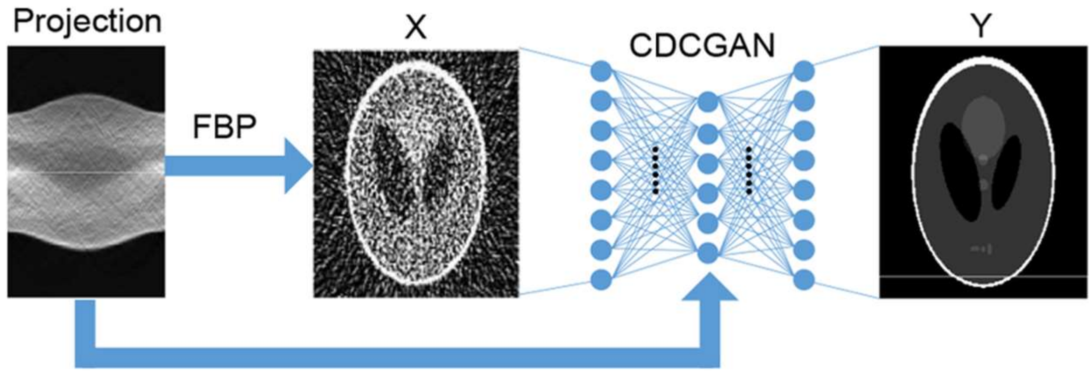
Super resolution

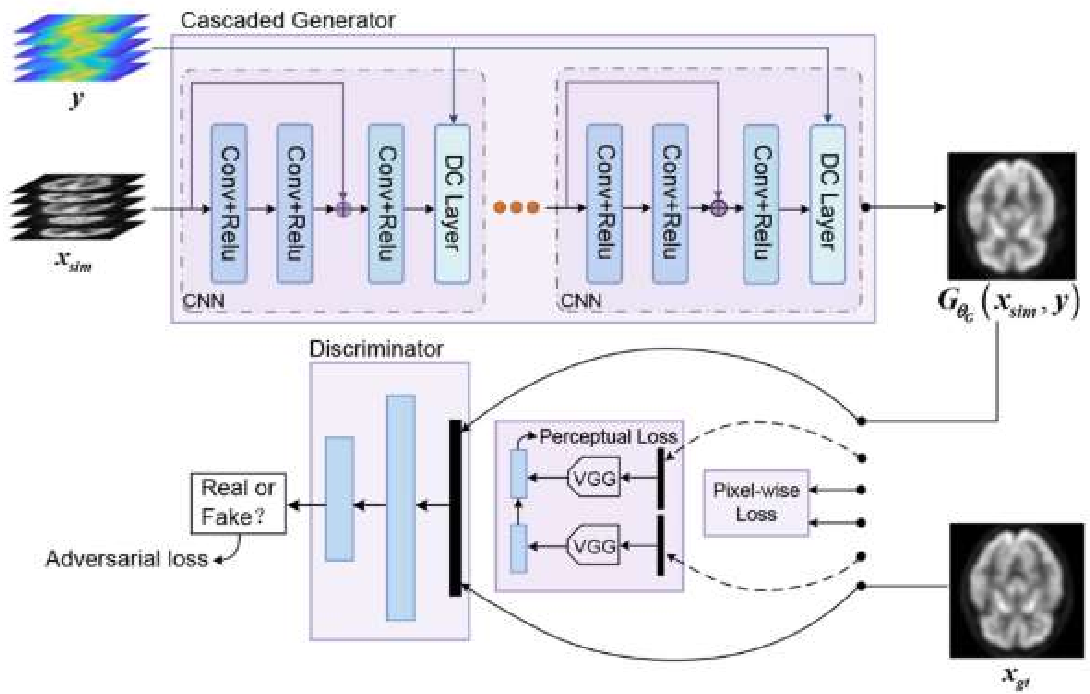


Waqar Ahmad, Hazrat Ali, Zubair Shah & Shoaib Azmat, *A new generative adversarial network for medical images super resolution,* Scientific Reports volume 12, Article number: 9533 (2022)

FBP: filtered back projection

PET scan reconstruction

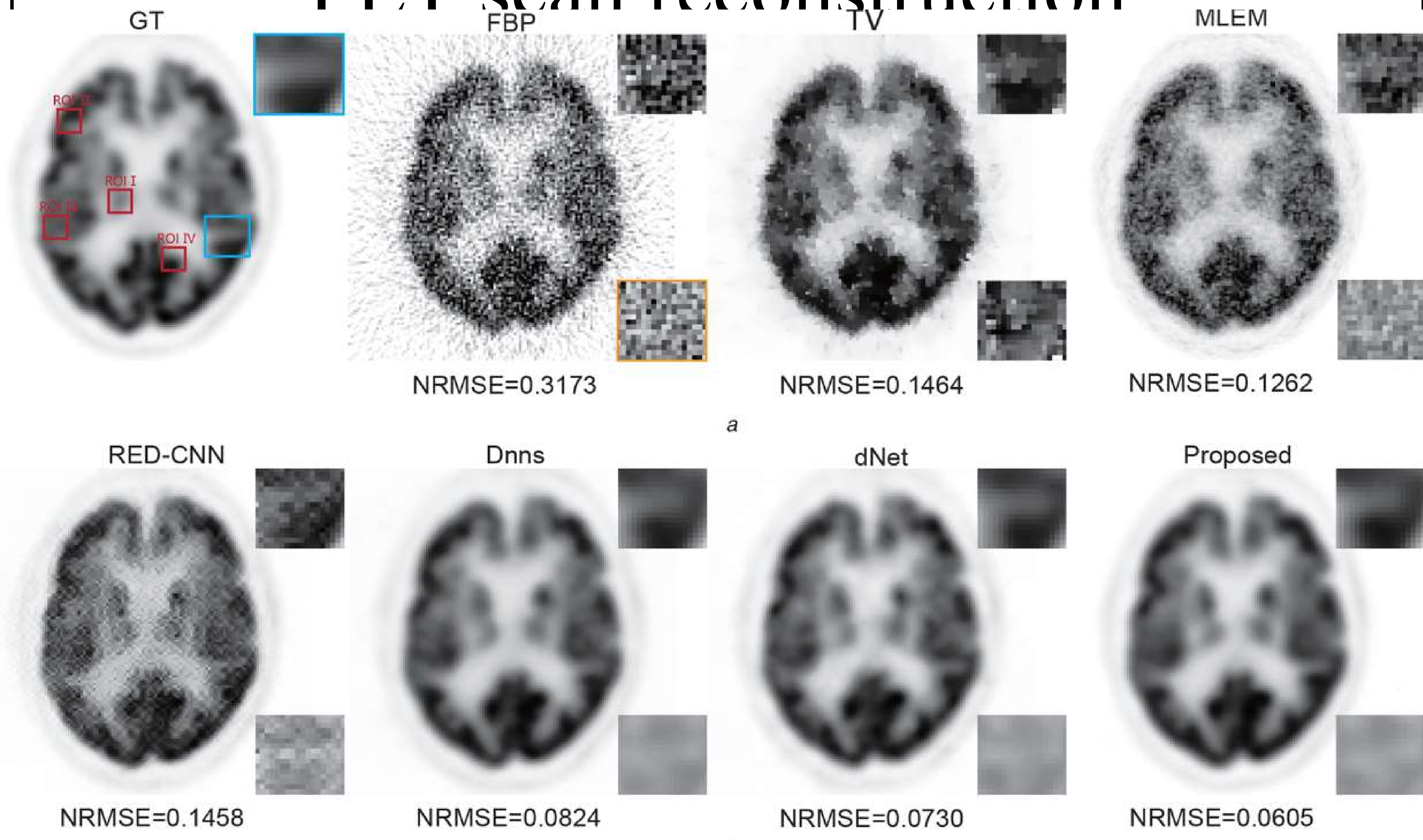




Qianqian D. et al. Iterative PET image reconstruction using cascaded data consistency generative adversarial network, 2021 IET Image Processing

FBP: filtered back projection

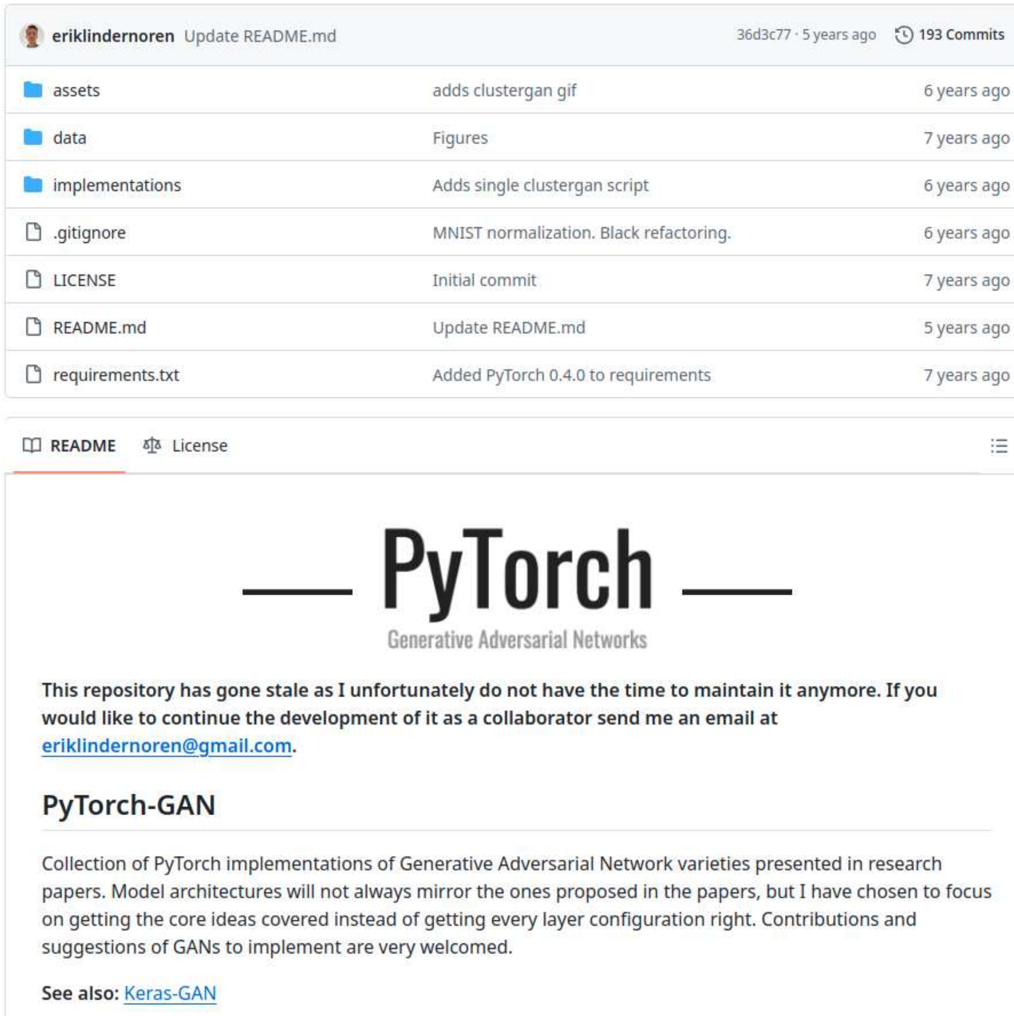
PET scan reconstruction



Qianqian D. et al. Iterative PET image reconstruction using cascaded data consistency generative adversarial network, 2021 IET Image Processing

Lot's of code available

github.com/eriklindernoren/PyTorch-GAN



eriklindernoren Update README.md 36d3c77 · 5 years ago 193 Commits

assets	adds clustergan gif	6 years ago
data	Figures	7 years ago
implementations	Adds single clustergan script	6 years ago
.gitignore	MNIST normalization. Black refactoring.	6 years ago
LICENSE	Initial commit	7 years ago
README.md	Update README.md	5 years ago
requirements.txt	Added PyTorch 0.4.0 to requirements	7 years ago

README License

PyTorch

Generative Adversarial Networks

This repository has gone stale as I unfortunately do not have the time to maintain it anymore. If you would like to continue the development of it as a collaborator send me an email at eriklindernoren@gmail.com.

PyTorch-GAN

Collection of PyTorch implementations of Generative Adversarial Network varieties presented in research papers. Model architectures will not always mirror the ones proposed in the papers, but I have chosen to focus on getting the core ideas covered instead of getting every layer configuration right. Contributions and suggestions of GANs to implement are very welcomed.

See also: Keras-GAN

merci