UniGe | MaLGa

Autoencoders & GANs

Deep Learning: A hands-on introduction

A course offered in the PhD program in Computer Science and Systems Engineering

Nicoletta Noceti

Machine Learning Genoa Center – University of Genoa

Slides by Francesca Odone, Nicoletta Noceti, Giorgio Cantarini

UniGe | MaLGa

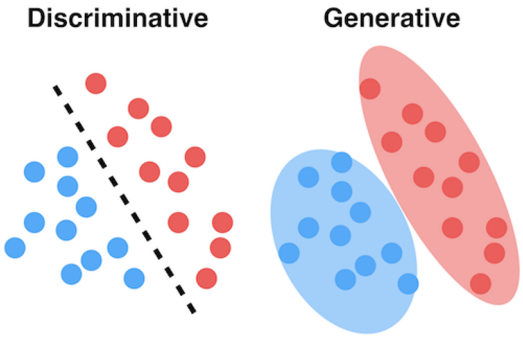
Introduction

Generative modeling

Given a Training set X with the associated labels Y:

Discriminative models $p(Y|X)$

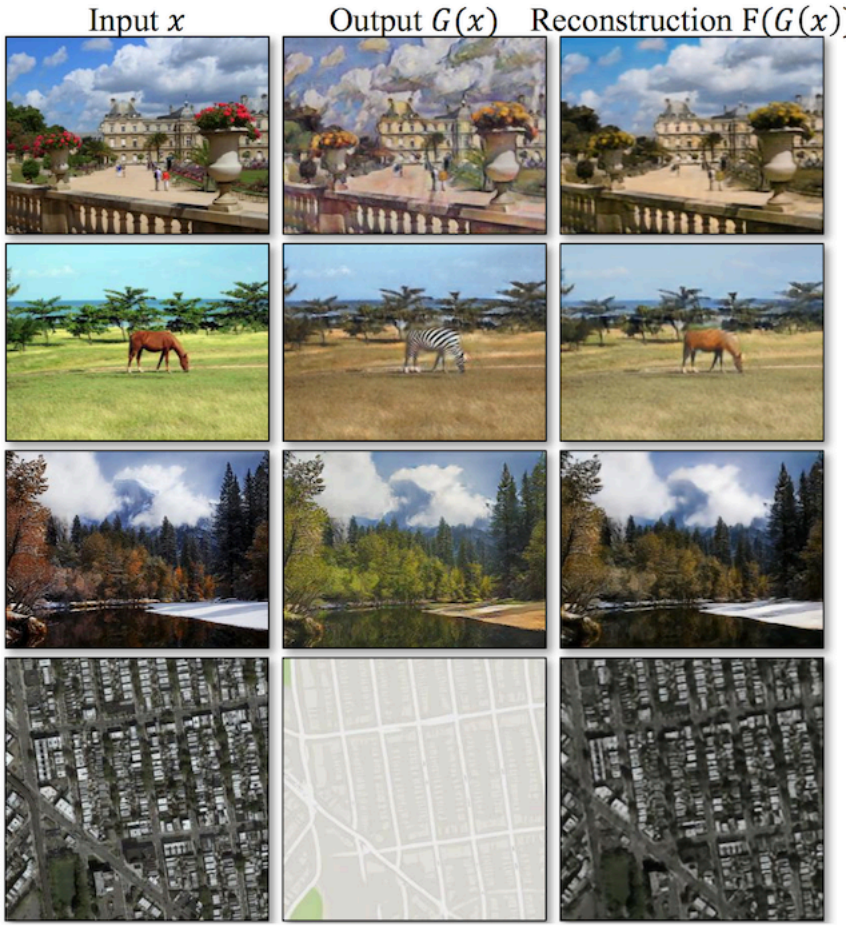
Generative models $p(X)$



Generative modeling

Goal: Take as input unlabeled training samples from some distribution and learn a model that represent that distribution

- density estimation
- learn appropriate representations or embeddings
- generate new data


Input x
Output $G(x)$
Reconstruction F$(G(x)$

UniGe | MaLGa

Autoencoders

Autoencoders *(automatic encoders)*

- **Unsupervised** approach for learning **a lower dimensional feature representation** of an input from unlabelled training data

- The model is usually **forced to give priority to some specific aspects** in the data

- It is composed by two parts:

  - An **encoder** function, h = f(x): it describes the lower dimensional code to represent the input

  - A **decoder** function, r = g(h), that produces the approximate reconstruction

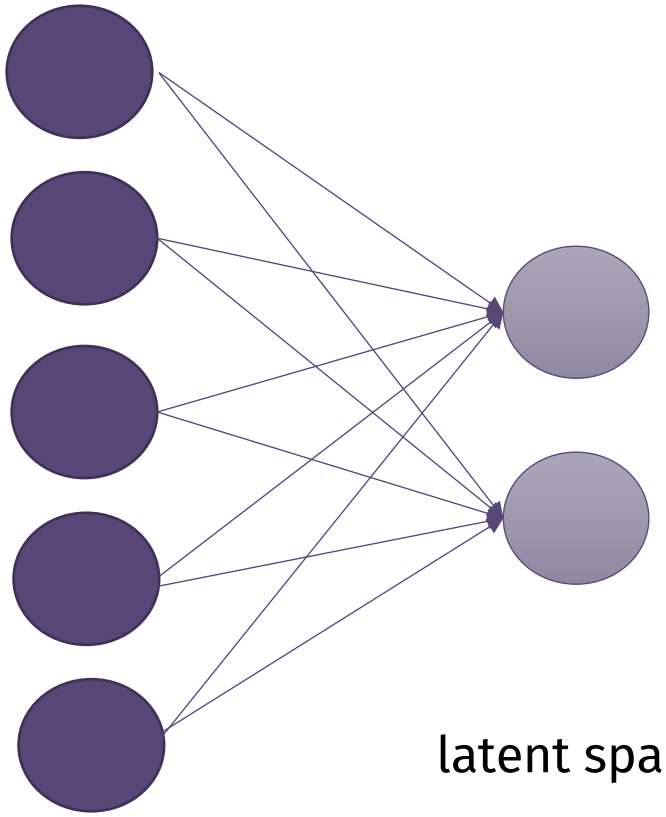
Autoencoders

- Traditionally (1987... 1994), they were used for **dimensionality reduction** and **feature learning**

- More recently, they have been applied to **generative models**

- They may be thought of as a special case of feedforward networks, and they can be trained using the very same strategies

Basic autoencoder



latent space h

Input x

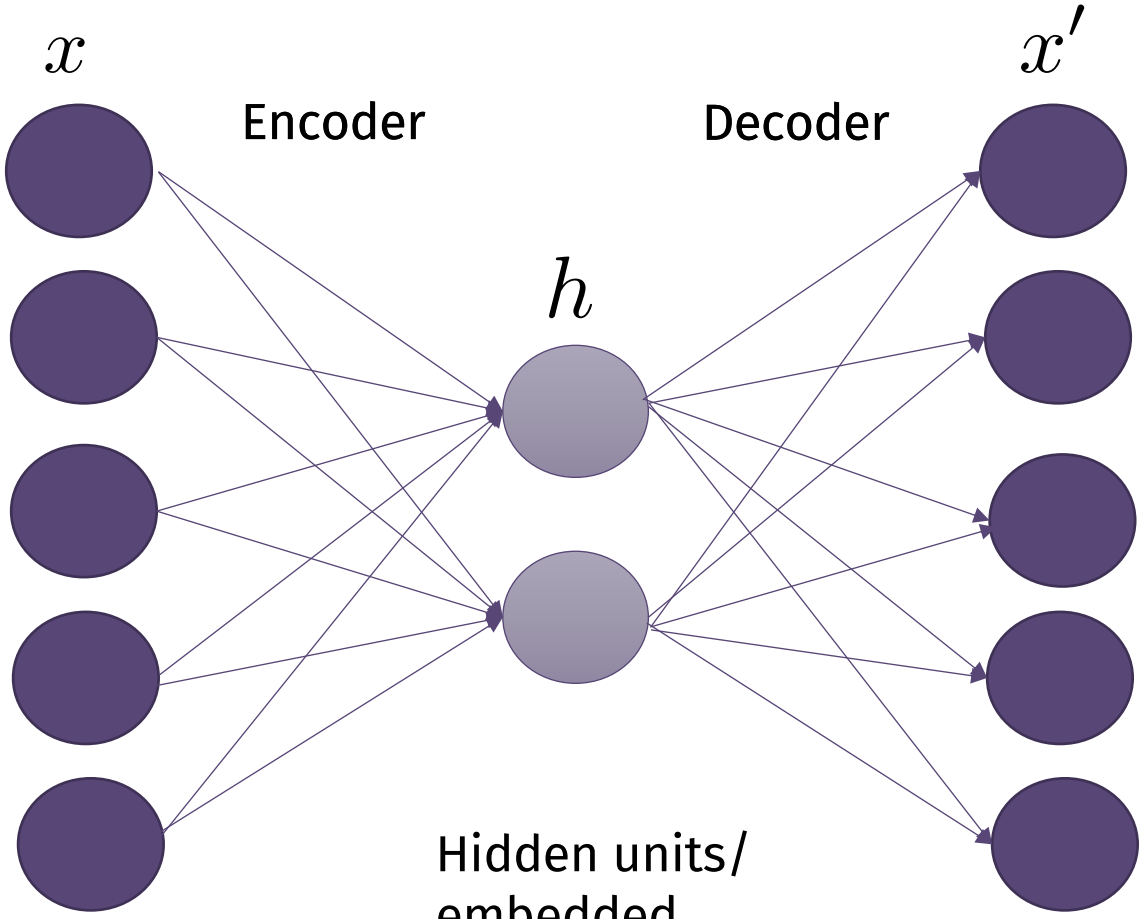
Autoencoders are an **unsupervised** approach for learning a **lower-dimensional** feature representation

The **encoder** learns a mapping from data *x* to a low-dimensional latent space, *h*

How can we learn the latent space?

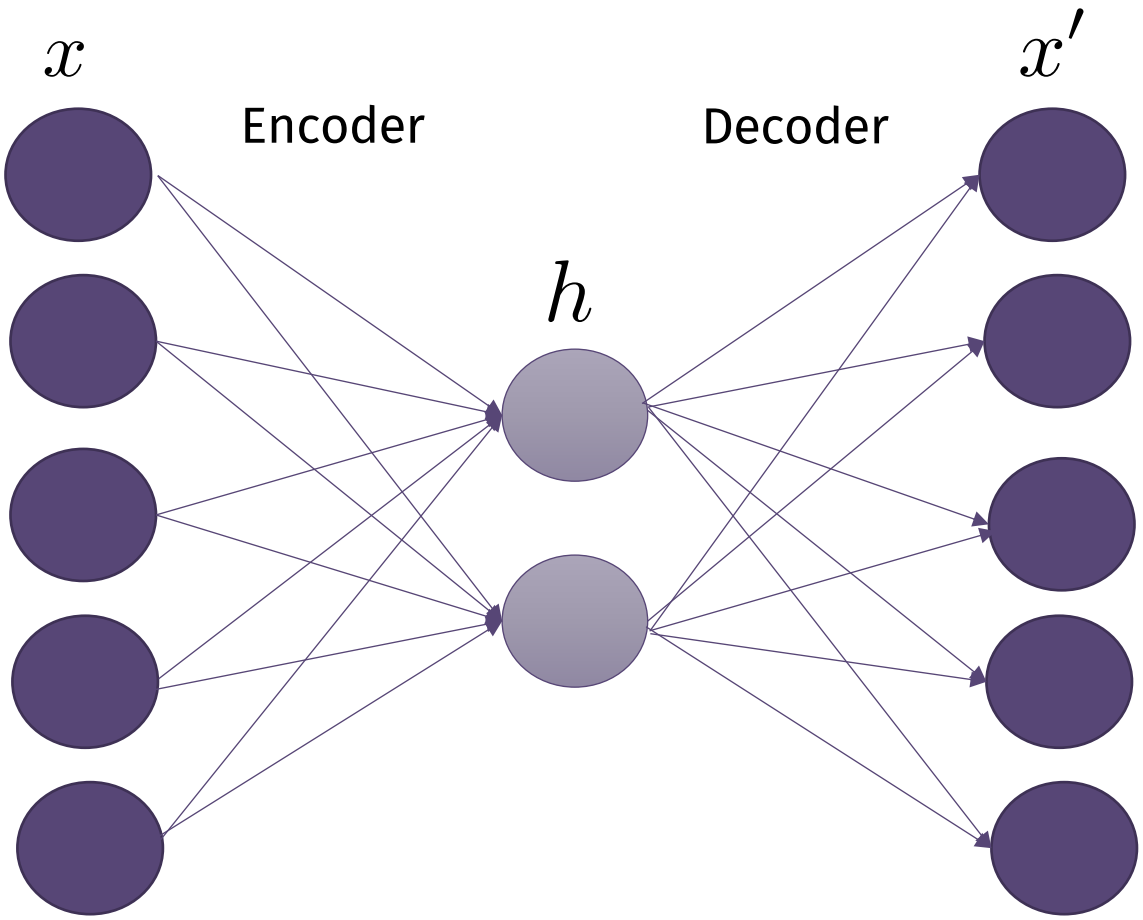
Basic autoencoder

How can we learn the latent space?
Train the model to use it to reconstruct the original data



The **decoder** learns mapping back from latent *h* to a reconstructed observation *x'*

Basic autoencoder



Encoder

$$h = f(x)$$

Decoder

$$x' = g(h)$$

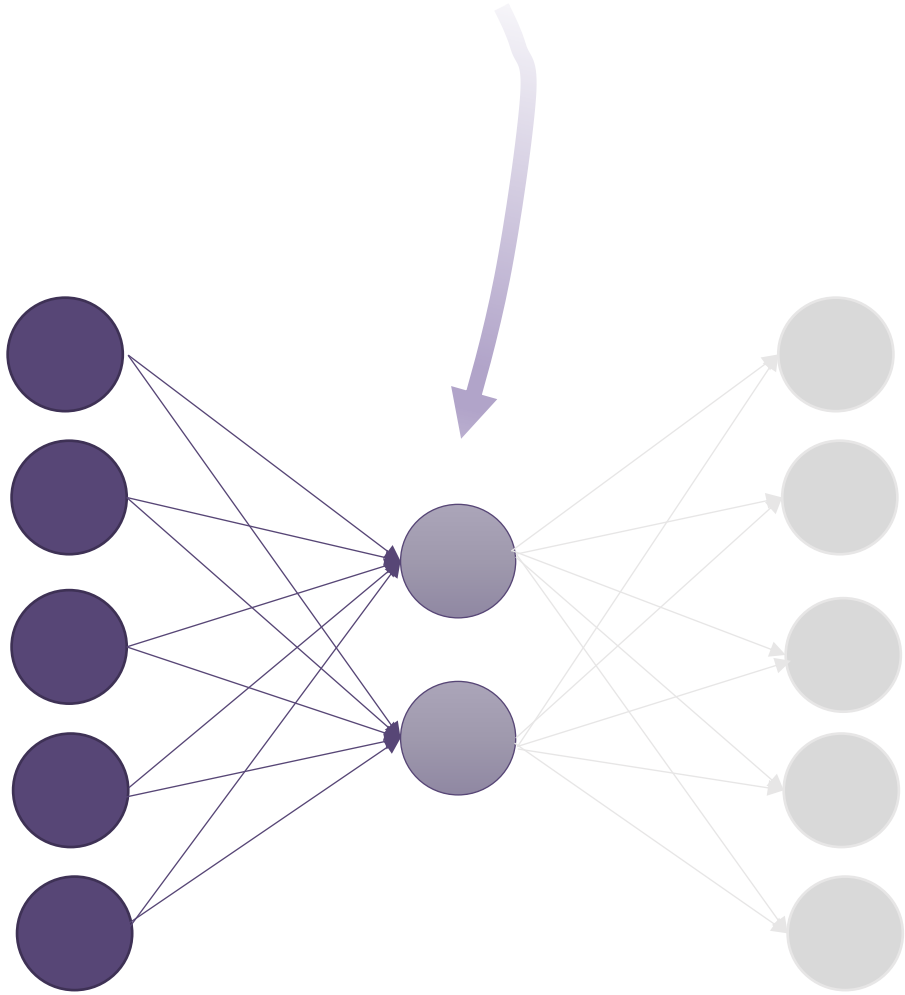
Loss

$$L(x, g(f(x)))$$

Example of a reconstruction loss (notice, no labels!)

$$L(x, x') = ||x - x'||^2$$

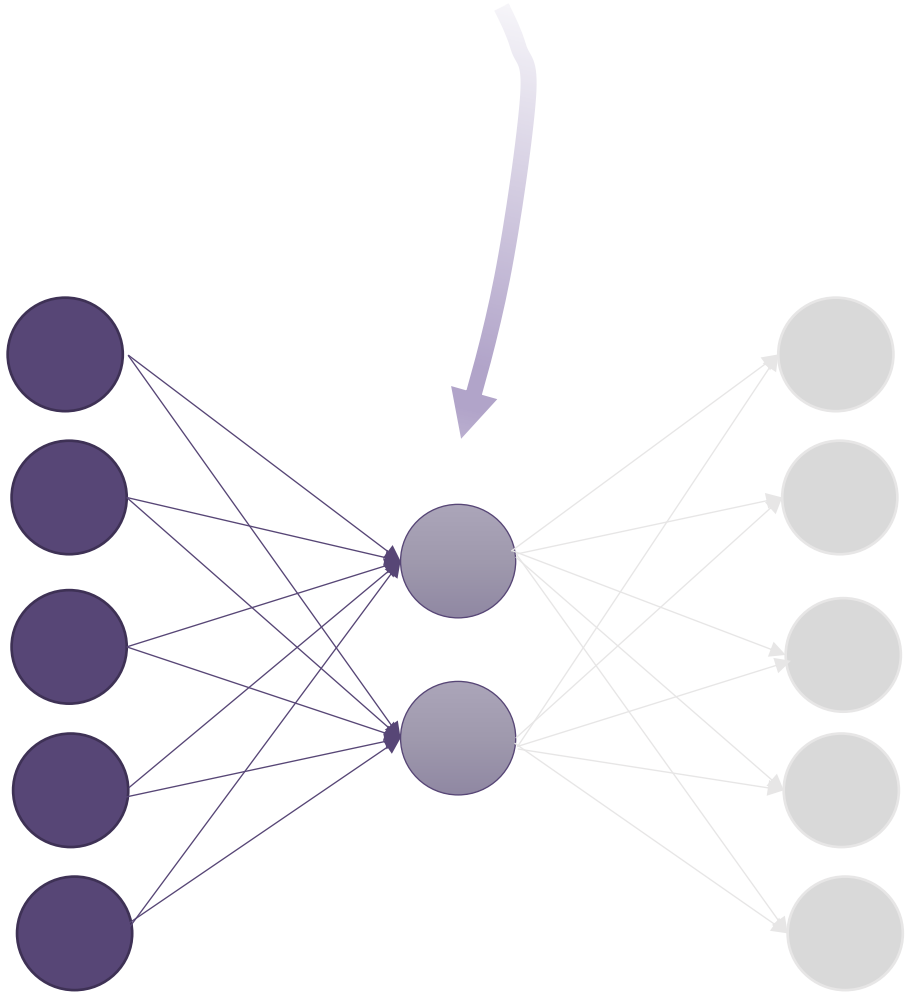
How can we use the latent space?



- Autoencoders can be seen as an **unsupervised** approach for learning a **lower-dimensional** feature representation

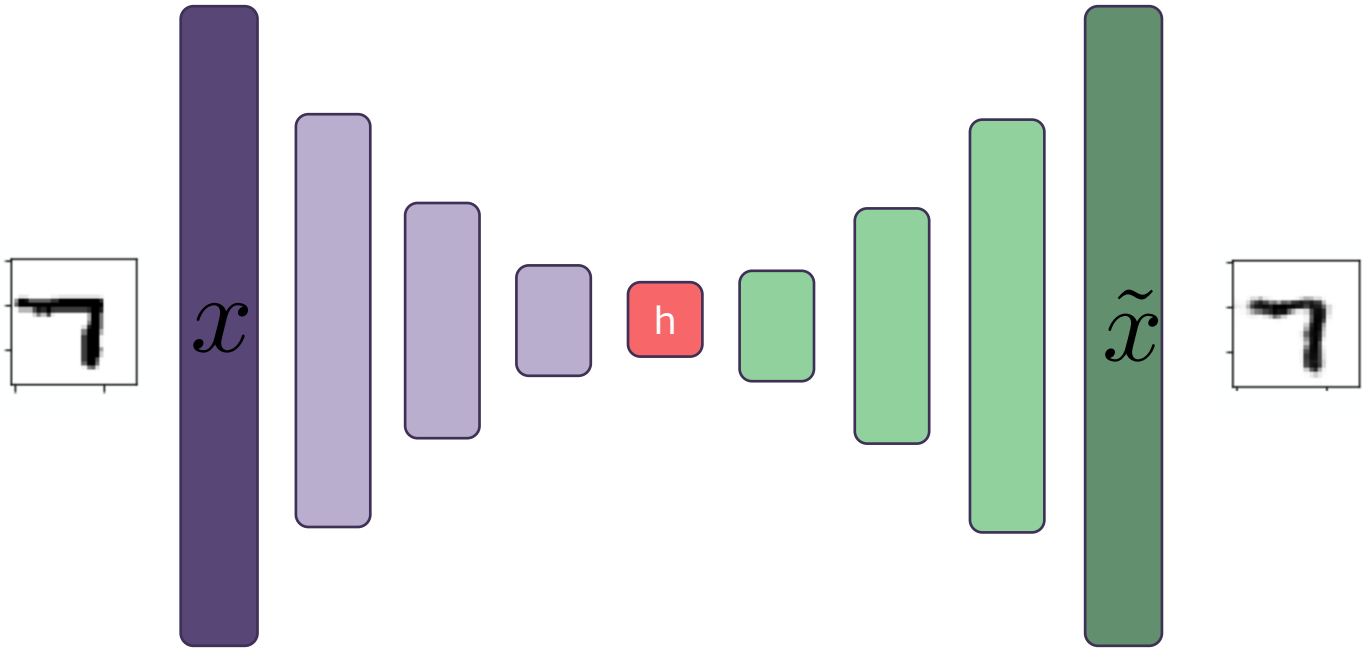
- Why do we need it?

Embedding or latent variables



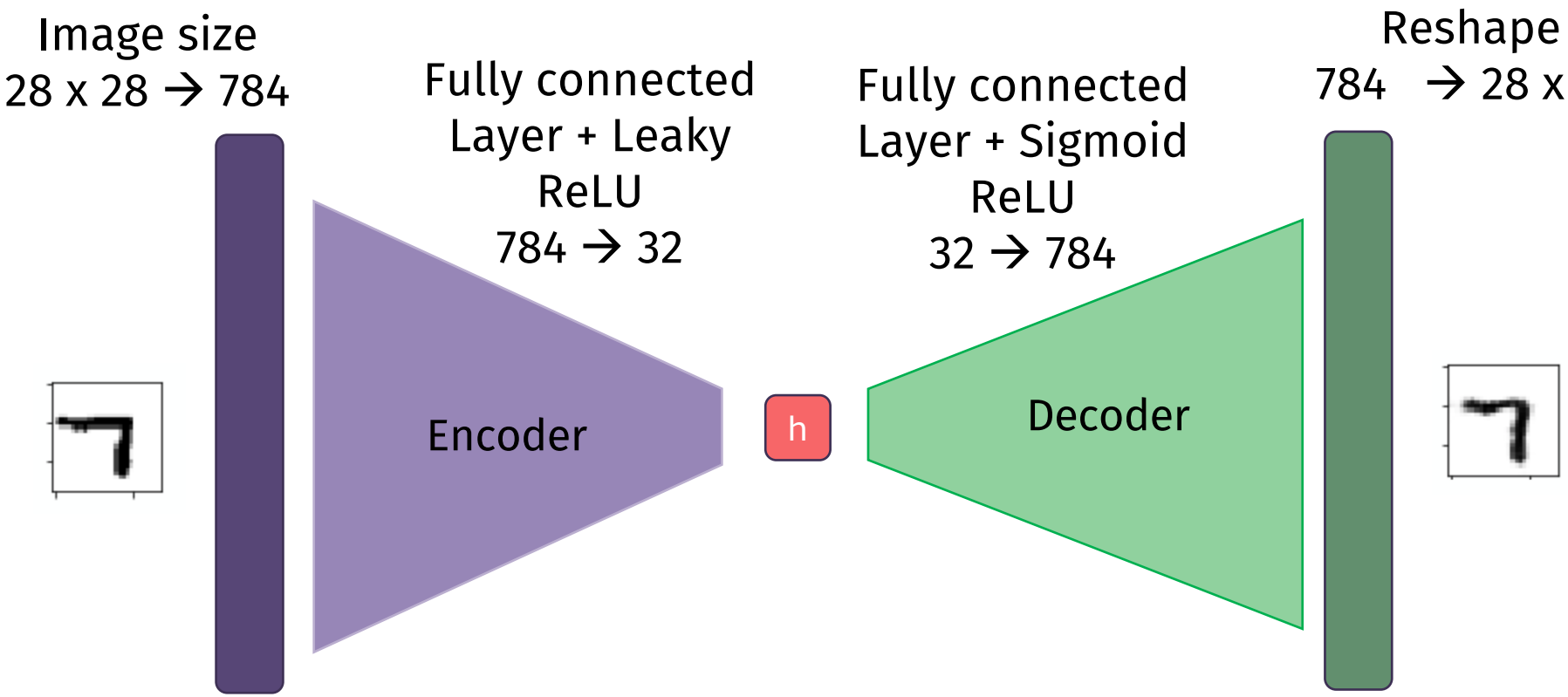
- Autoencoders can be seen as an **unsupervised** approach for learning a **lower-dimensional** feature representation
- After training, you can disregard the output and **use embedding as inputs** to classic machine learning methods
- **Transfer learning**: train autoencoders on large datasets and fine tune on your (smaller) dataset
- **Visualization** (projecting the embeddings in lower a dimensional space)

Autoencorders

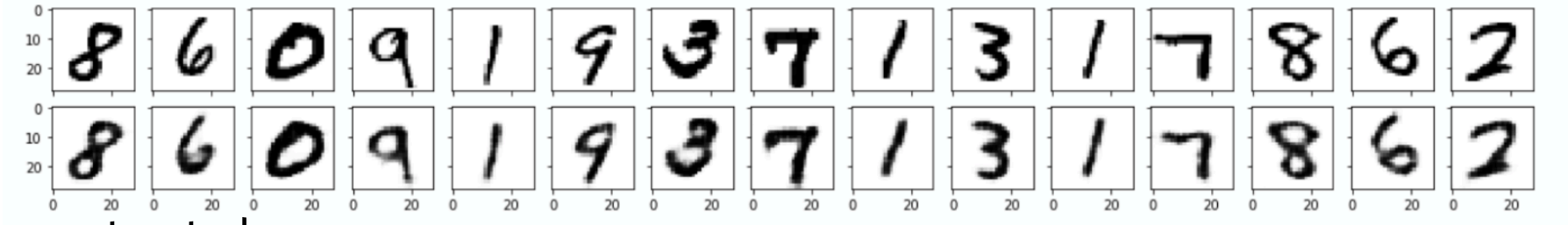


Arbitrary complex encoder

Autoencoders: an example



Original



Reconstructed

Dimensionality of the latent space

reconstruction quality

2D Latent Space

5D Latent Space

Autoencoding compresses!

Autoencoders for representation learning

Hidden layer forces the network to lean a compressed latent representation

Reconstruction loss forces the latent representation to capture as much information on the data as possible

The latent variables will be

- Good for the training distribution
- Bad for other types of input

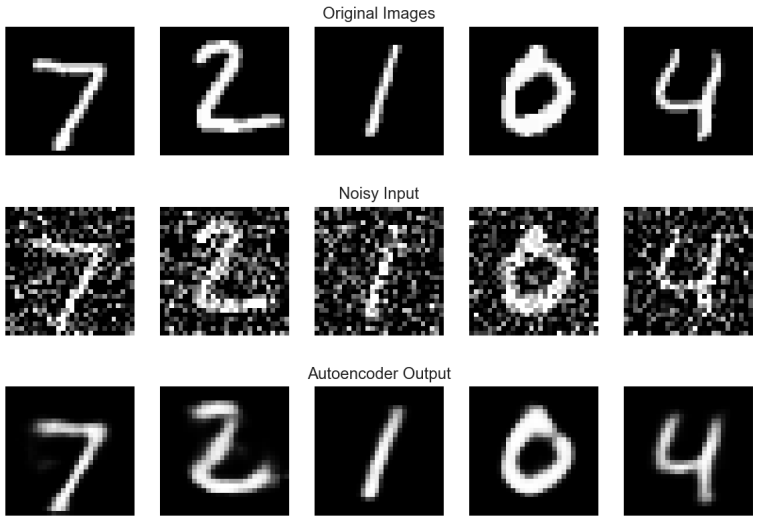
Application example: denoising autoencoder (DAE)

It minimizes the loss

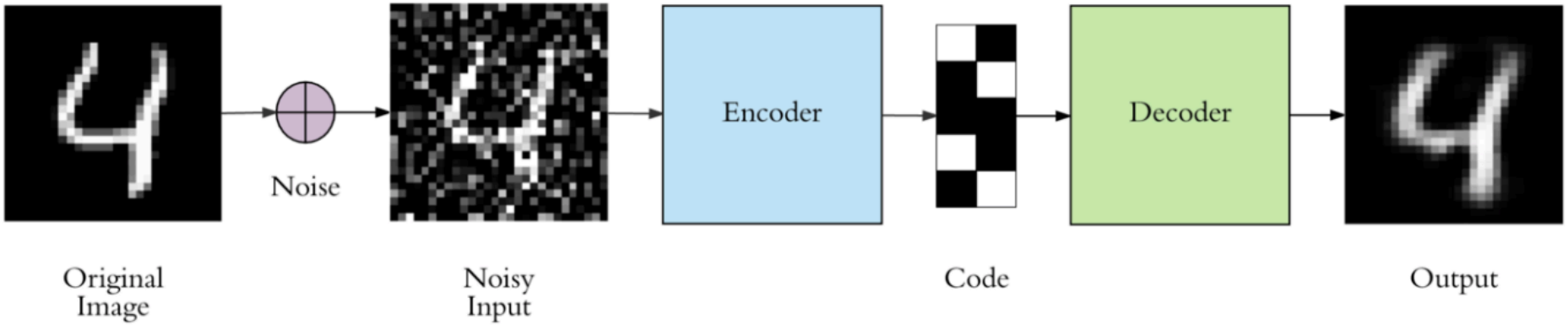
$$L(x, g(f(\tilde{x})))$$

where $\tilde{x}$ is a copy of x corrupted by some form of noise

Thus, the encoder learns how to undo this corruption on the input rather than simply copying it



Denoising autoencoders



Regularized autoencoders

You might have high capacity when the model has equal or higher dimension than the input. In the latter case it is called **overcomplete autoencoder**

Regularized autoencoder provides the ability to train an architecture choosing the code dimension and the capacity of encoder and decoder based on the complexity of the distribution

Idea: regularized autoencoders use a loss function that encourages the model to have certain properties (as sparsity for instance)

Regularized autoencoders

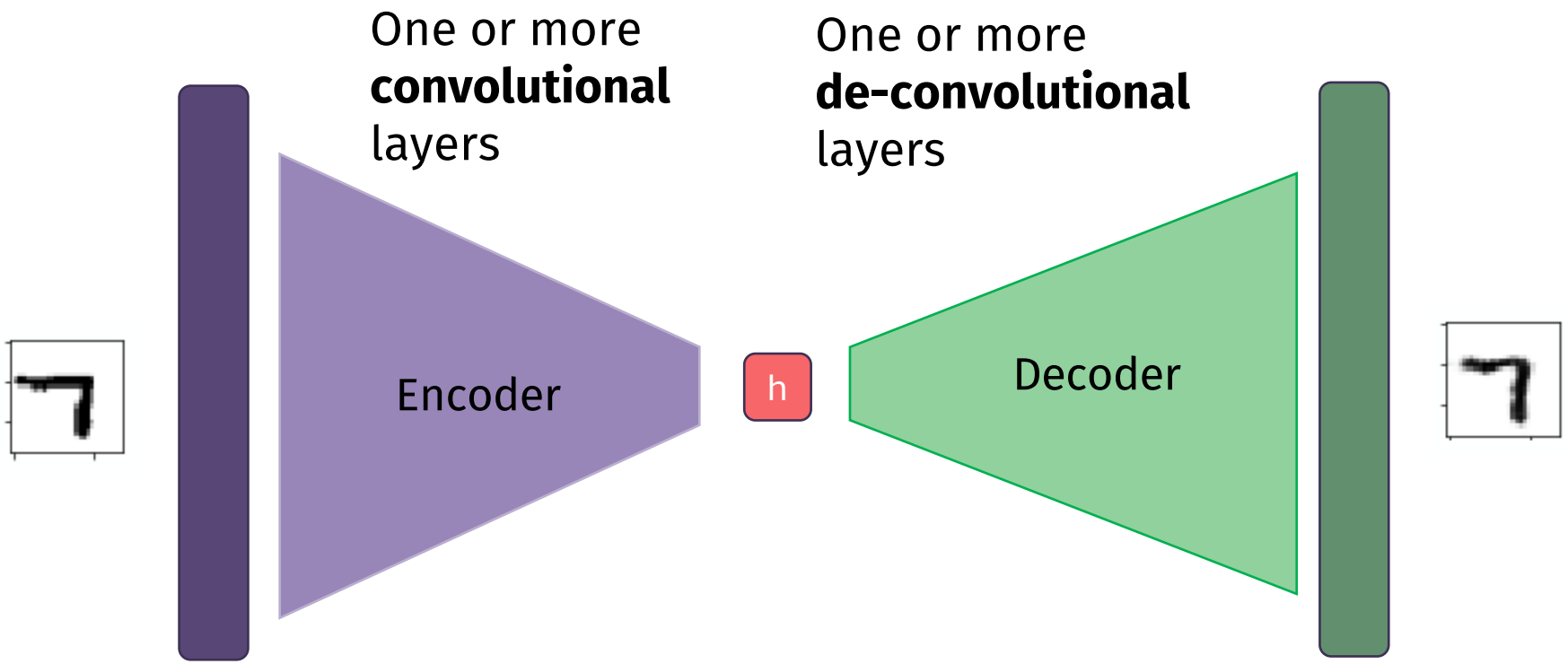
An example of regularized autoencoder is the **sparse autoencoder**, when you apply a sparsity penalty on the code layer, so that the loss becomes

$$L(x, g(f(x)) + \lambda ||h||_1$$

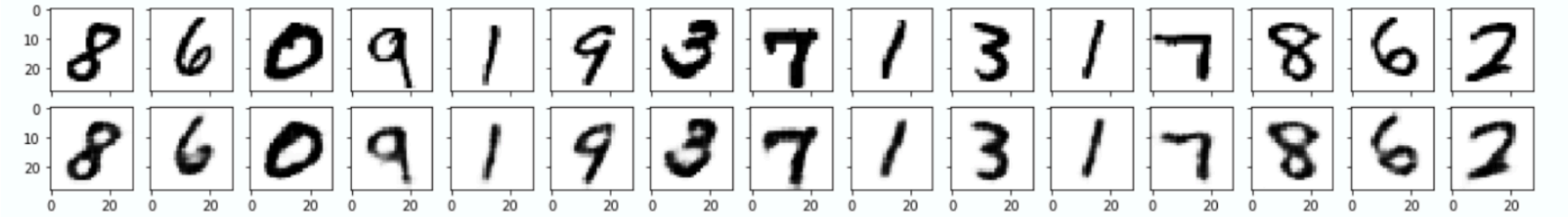
An alternative is to penalize the derivatives, forcing to learn a function that does not change much when the input (x) slightly changes

$$L(x, g(f(x)) + \lambda \sum_i ||\nabla_x (h_i)||^2$$

Convolutional autoencoders: the concept



Original



Reconstructed

De-convolution

When managing image data, encoder and decoder are made of, respectively, convolutional and de-convolutional layers

De-convolution (often referred to as transposed convolution because, mathematically, deconvolution is in fact a different operation) allows to go from a lower resolution image to a higher resolution image

Regular convolution (in the encoder)

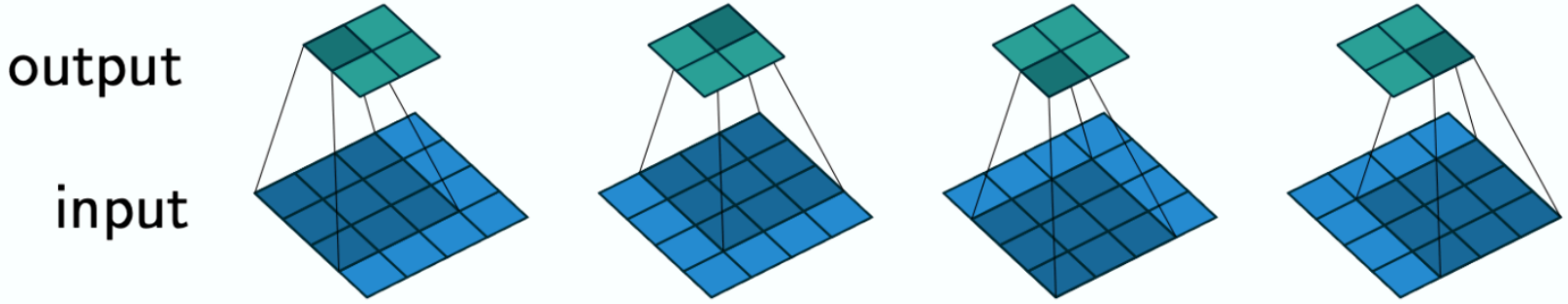
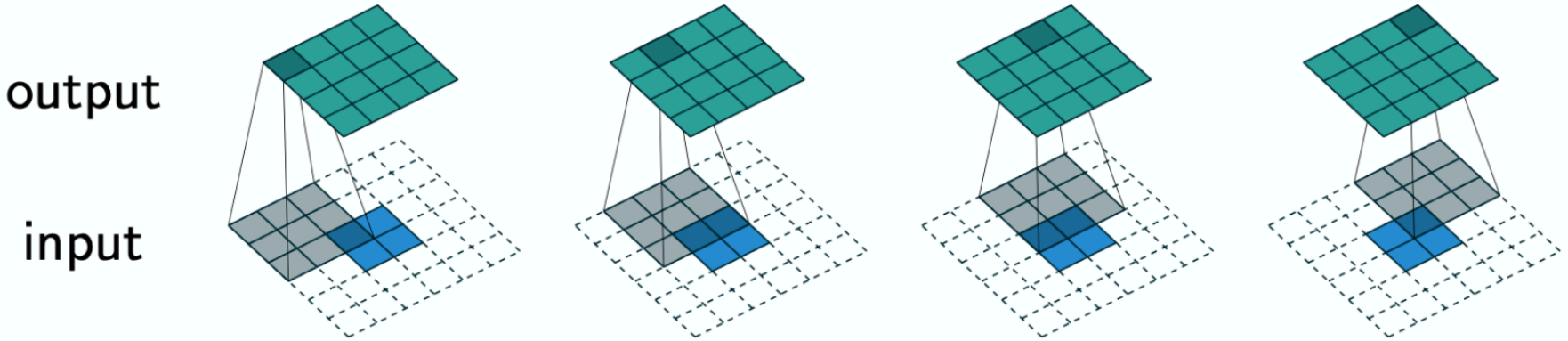


Figure 2.1: (No padding, unit strides) Convolving a 3×3 kernel over a 4×4 input using unit strides (i.e., $i = 4$, $k = 3$, $s = 1$ and $p = 0$).

Image from https://github.com/rasbt/stat479-deep-learning-ss19/blob/master/L15_autoencoder/L15_autoencoder_slides.pdf

De-convolution (in the decoder)



Dumoulin Visin https://github.com/rasbt/stat479-deep-learning-ss19/blob/master/L15_autoencoder/L15_autoencoder_slides.pdf

UniGe | MaLGa

Generative Adversarial Networks
GANs

Generative Adversarial Networks (GANs)

- Their purpose is to **generate new data instances**

- They learn the distribution of the training set and can generate new data never seen before

- They are based on a game theoretic scenario in which a generator network must compete against an adversary

GANs in the last few years...

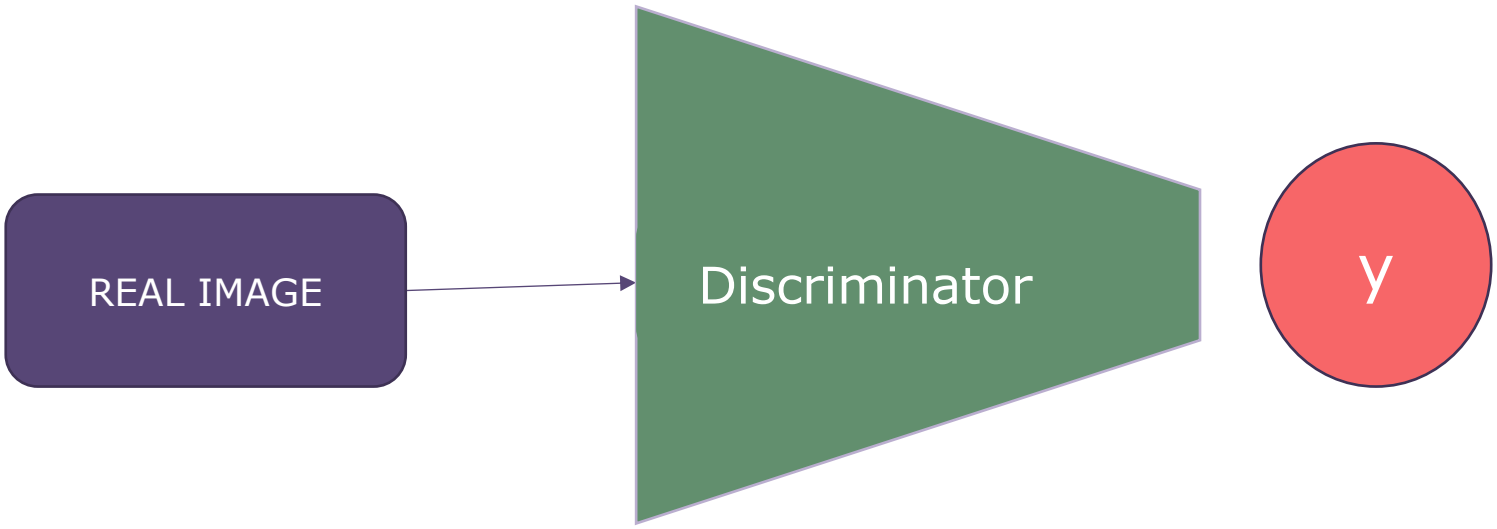
GANs

- A **generator network** directly produces samples

- Its adversary, the **discriminator network**, attempts to distinguish between samples drawn from the training data and samples drawn from the generator

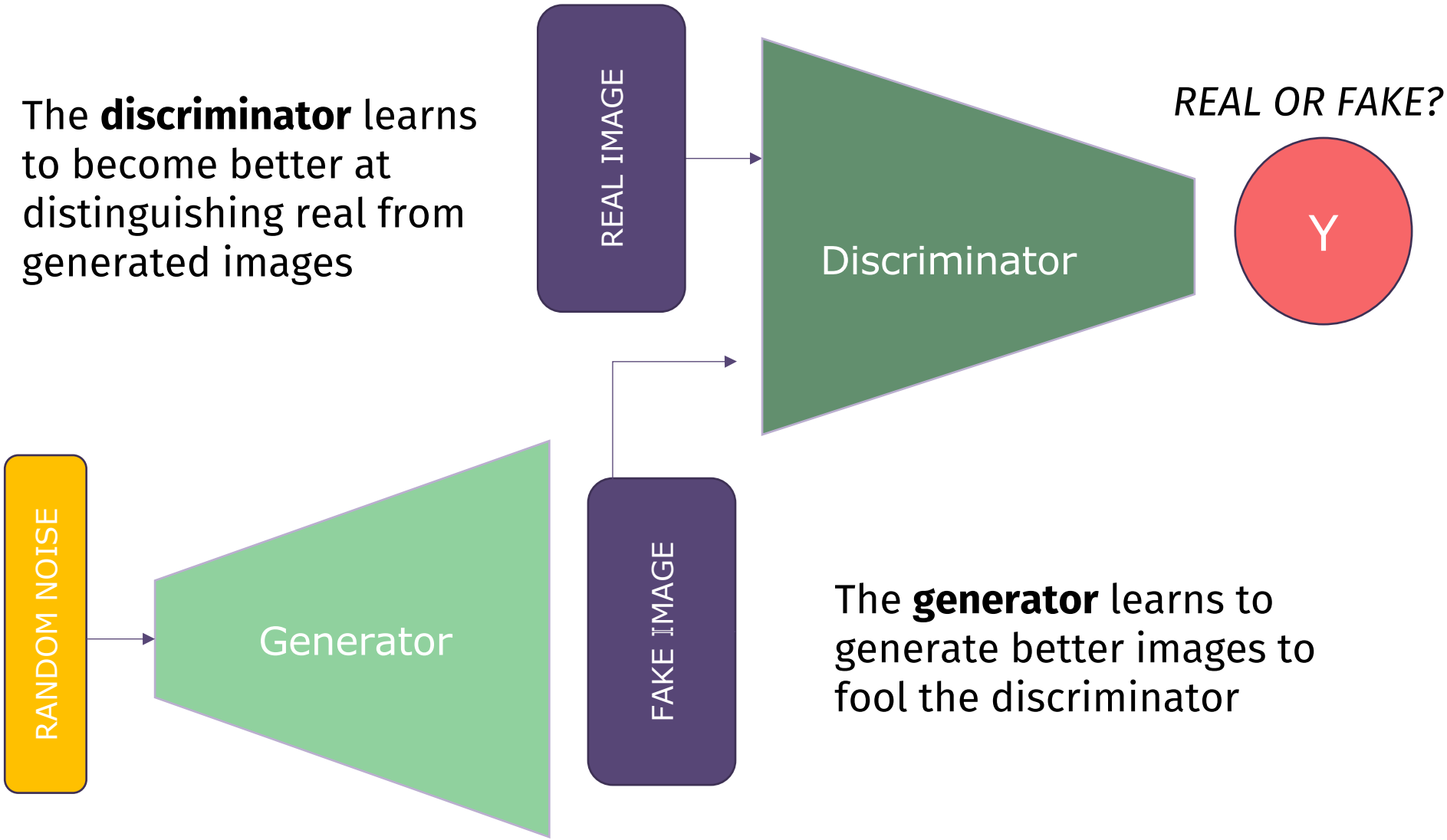
- The discriminator estimates a probability values evaluating how likely is that the sample is a training example rather than a fake sample drawn from the model

Discriminator



GAN

The **discriminator** learns to become better at distinguishing real from generated images



The **generator** learns to generate better images to fool the discriminator

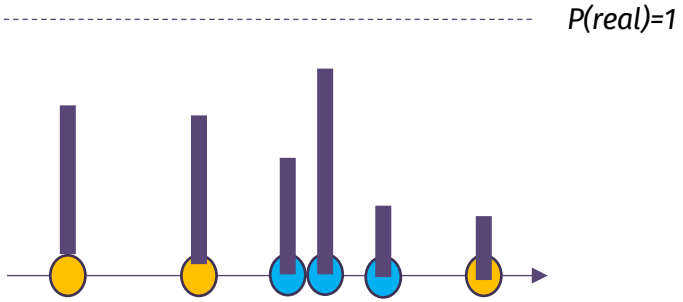
Intuition

GENERATOR

Intuition

GENERATOR

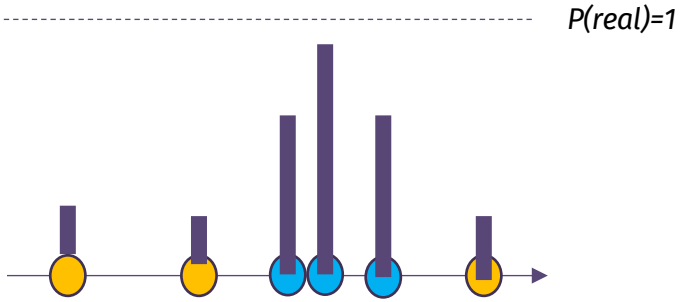
DISCRIMINATOR



Intuition

GENERATOR

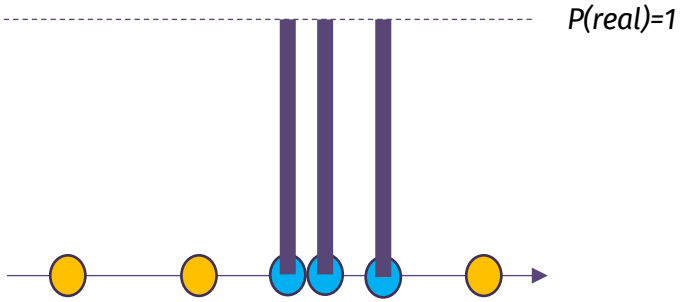
DISCRIMINATOR



Intuition

GENERATOR

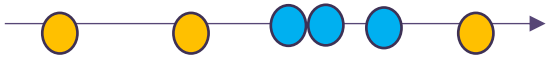
DISCRIMINATOR

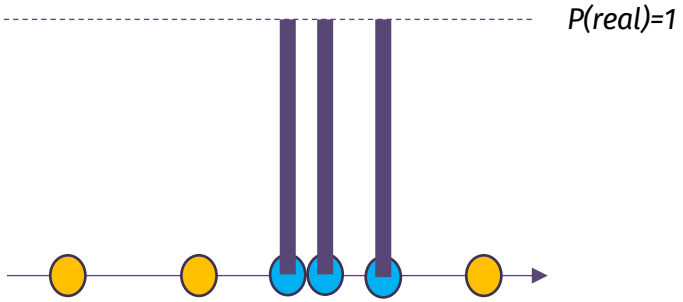


Intuition

GENERATOR

DISCRIMINATOR

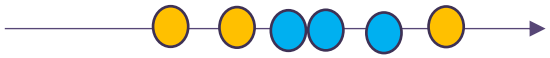


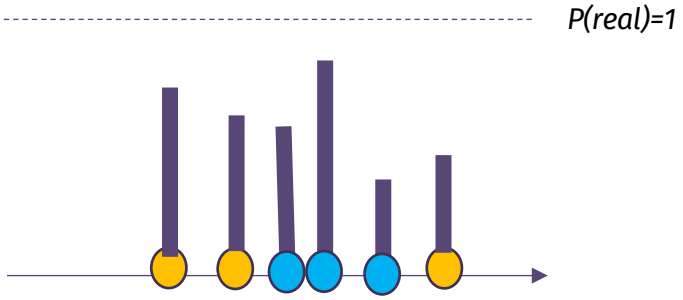


Intuition

GENERATOR

DISCRIMINATOR

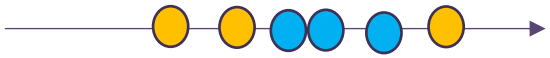


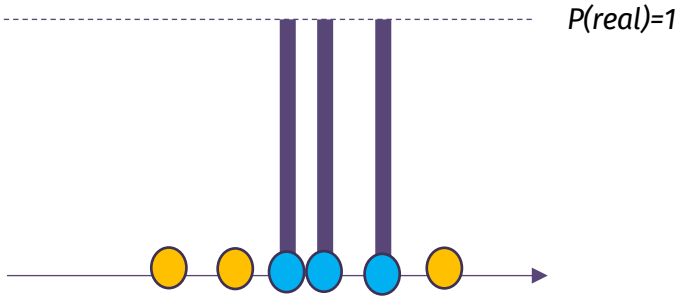


Intuition

GENERATOR

DISCRIMINATOR





Intuition

GENERATOR

DISCRIMINATOR

...the process continues until the discriminator is unable to learn how to distinguish between real and fake

GAN model and training

The two players $G(z, \theta_g)$ and $D(x, \theta_d)$ are two differentiable functions implemented by Deep Neural Networks.

Two-player Minmax game with value function:

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

Minmax is solved through alternating gradient descent: the parameters θ_d and θ_g are updated iteratively.

GAN model and training

$$\min_G \max_D V(D, G)$$

- At convergence, the generator's samples are indistinguishable from real data, and the discriminator outputs 0.5 everywhere

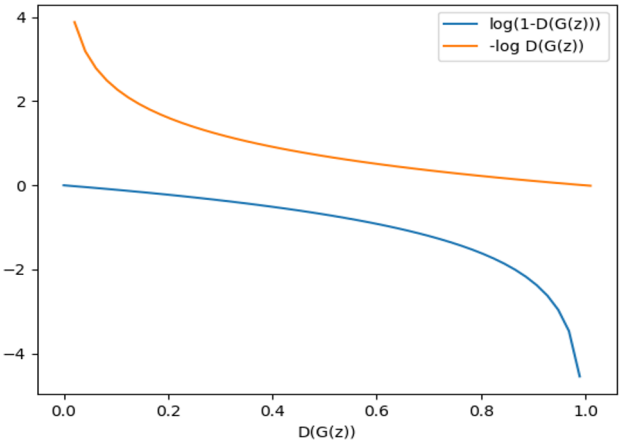
- In other words the convergence is reached when the actions of one of the players do not change depending on the actions of the other players

- As you can imagine, training can be very slow

GAN model and training

In practice the Generator is not trained to minimize the previously seen formulation but to maximize the following:

$$\min_G \mathbb{E}_{z \sim p_z(z)} \left[\log \left(1 - D \left(G \left(z \right) \right) \right) \right] \rightarrow \max_G \mathbb{E}_{z \sim p_z(z)} \left[\log \left(D \left(G \left(z \right) \right) \right) \right]$$



GAN model and training

The value function can be re-written as:

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

This change does not affect the dynamics of training and it helps especially in the first phase of learning.

UniGe | MaLGa

GAN examples

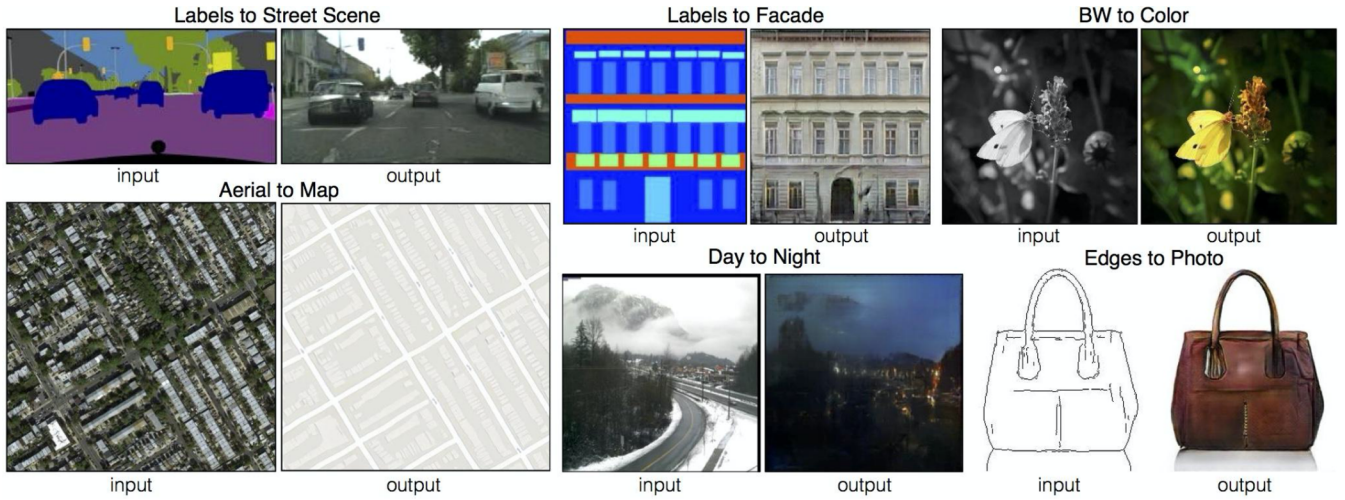
State of the art

“Large Scale GAN Training for High Fidelity Natural Image Synthesis” (2019)

https://arxiv.org/pdf/1809.11096.pdf

State of the art

“Image-to-Image Translation with Conditional Adversarial Nets” (2017)



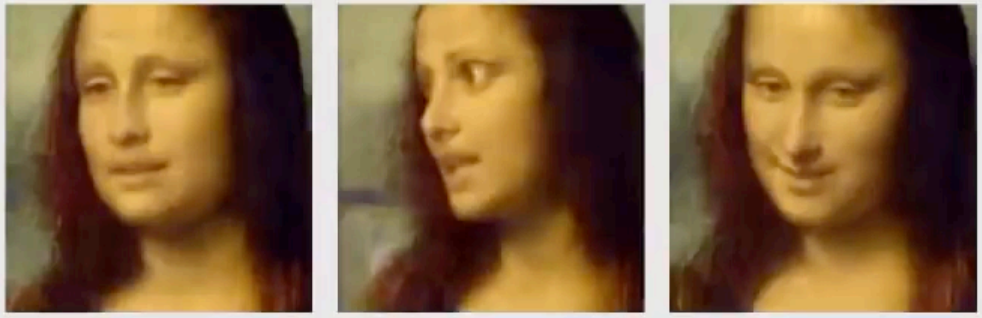
https://arxiv.org/pdf/1611.07004.pdf

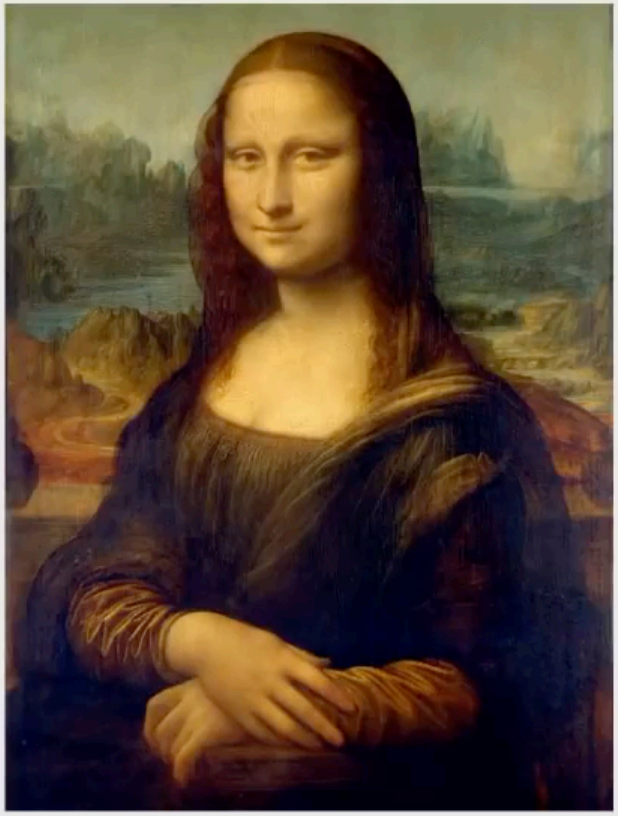
State of the art

“Unsupervised Image to Image Translation Networks” (2018)

https://www.youtube.com/watch?v=9VC0c3pndbI https://www.youtube.com/watch?v=Z_Rxf0TfBJE

Living portraits

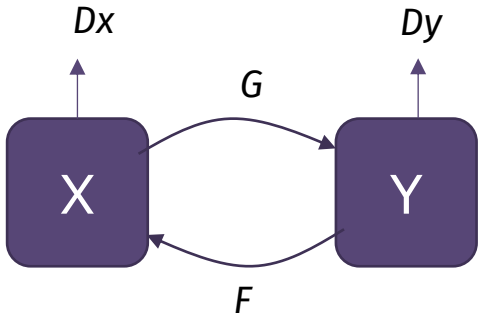


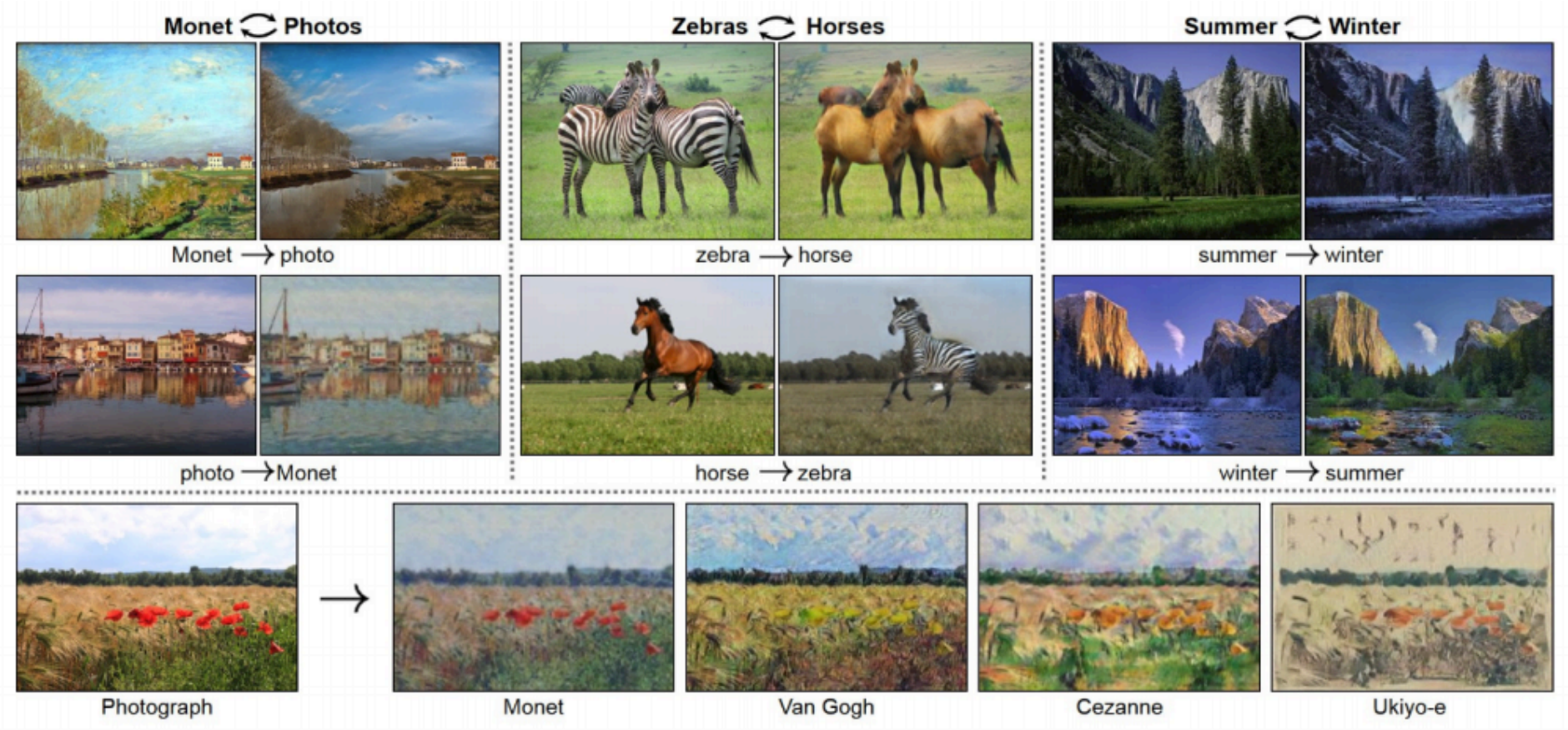


"Few-Shot Adversarial Learning of Realistic Neural Talking Head Models" (2019)

https://www.youtube.com/watch?v=KHKVTDbR5Ig

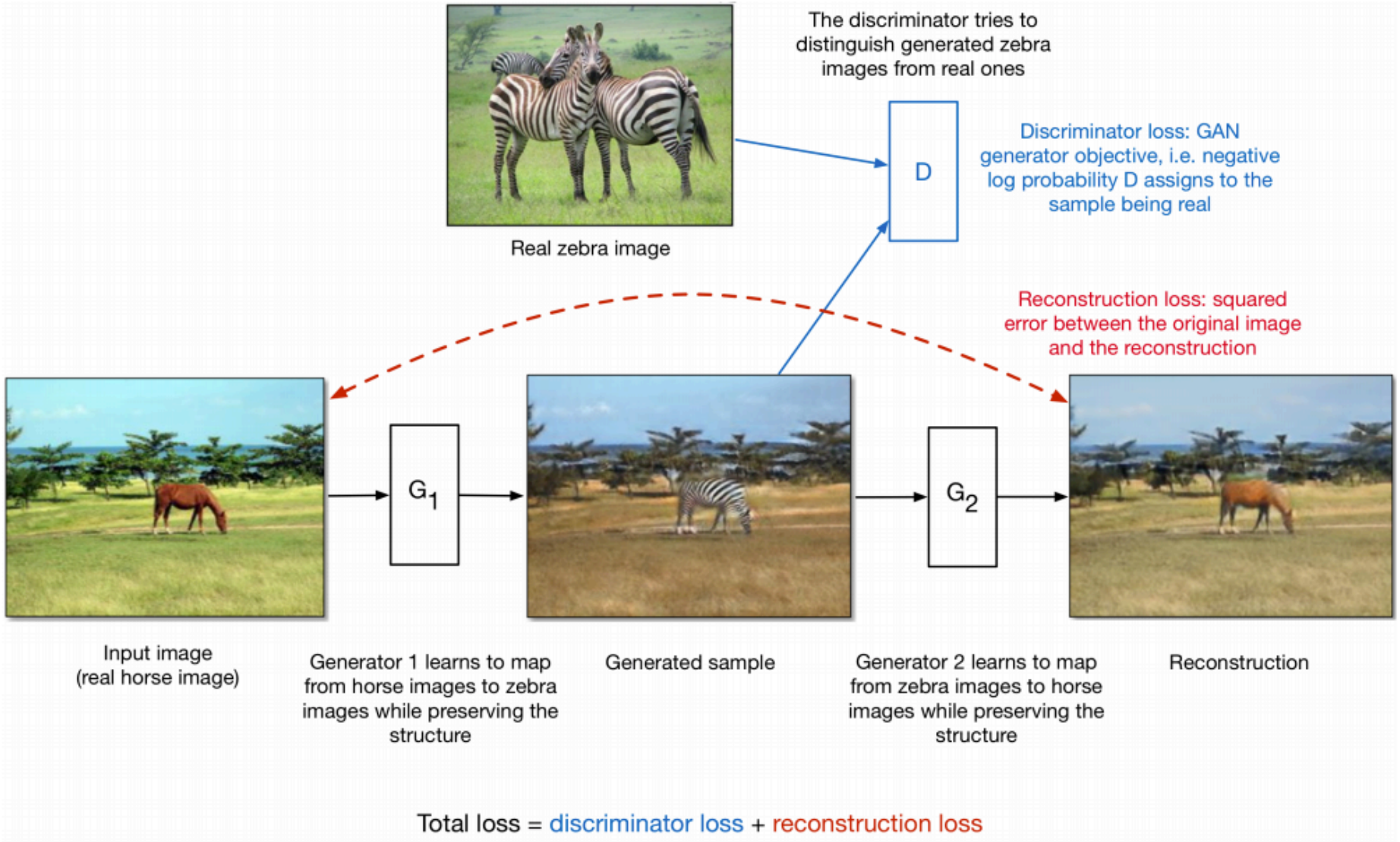
CycleGAN





https://junyanz.github.io/CycleGAN/

CycleGAN


The discriminator tries to distinguish generated zebra images from real ones
D
Discriminator loss: GAN generator objective, i.e. negative log probability D assigns to the sample being real
Real zebra image
Reconstruction loss: squared error between the original image and the reconstruction
G1
G2
Input image (real horse image)
Generator 1 learns to map from horse images to zebra images while preserving the structure
Generated sample
Generator 2 learns to map from zebra images to horse images while preserving the structure
Reconstruction
Total loss = discriminator loss + reconstruction loss

More exploration...

https://machinelearningmastery.com/impressive-applications-of-generative-adversarial-networks/

UniGe