Autoencoders & GANs

Nicoletta Noceti

Deep Learning

Slides by Nicoletta Noceti, Francesca Odone, Giorgio Cantarini

Introduction

Let's change the framework

- In supervised settings both input and output are available in our training set

- We now work with datasets for which the output is not known, i.e. in **unsupervised scenarios**

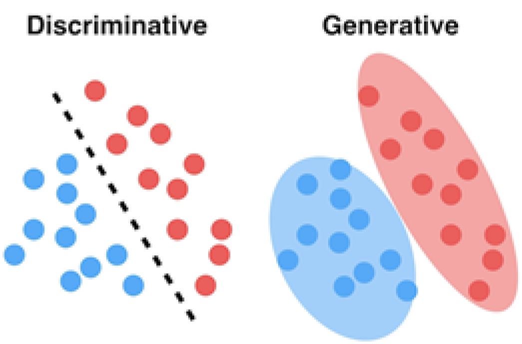
- Examples of applications: clustering, extracting hidden structures in data, retrieving similar data, generating new examples

Generative modeling

Given a Training set X with the associated labels Y:

Discriminative models $p(Y|X)$

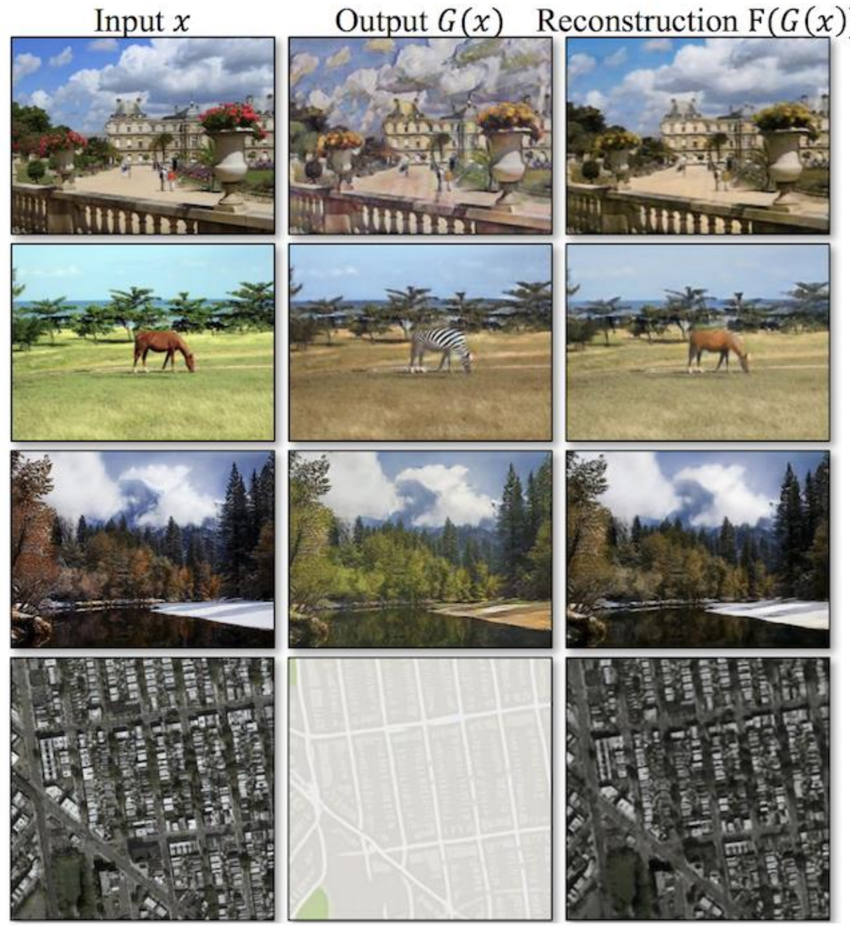
Generative models $p(X)$



Generative modeling

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

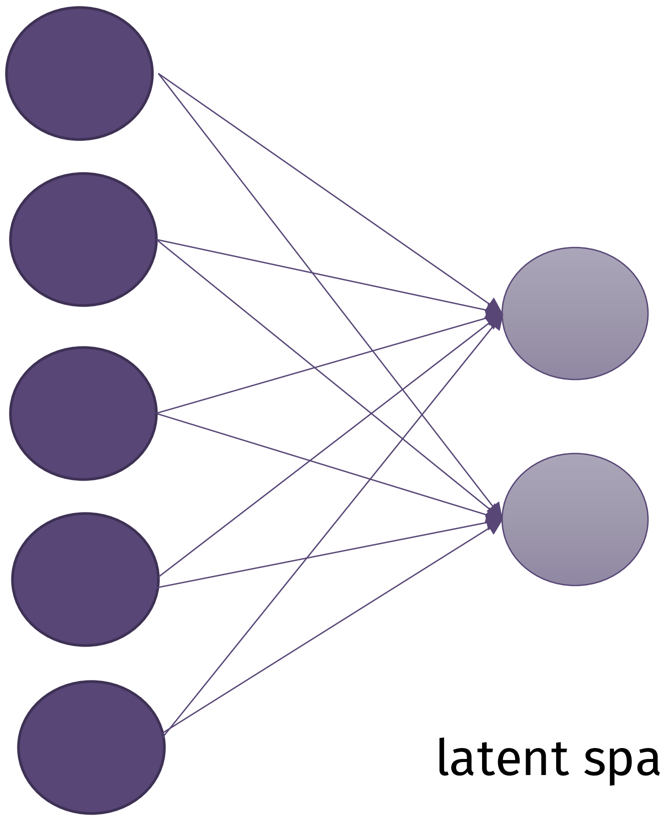
Autoencoders

- 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

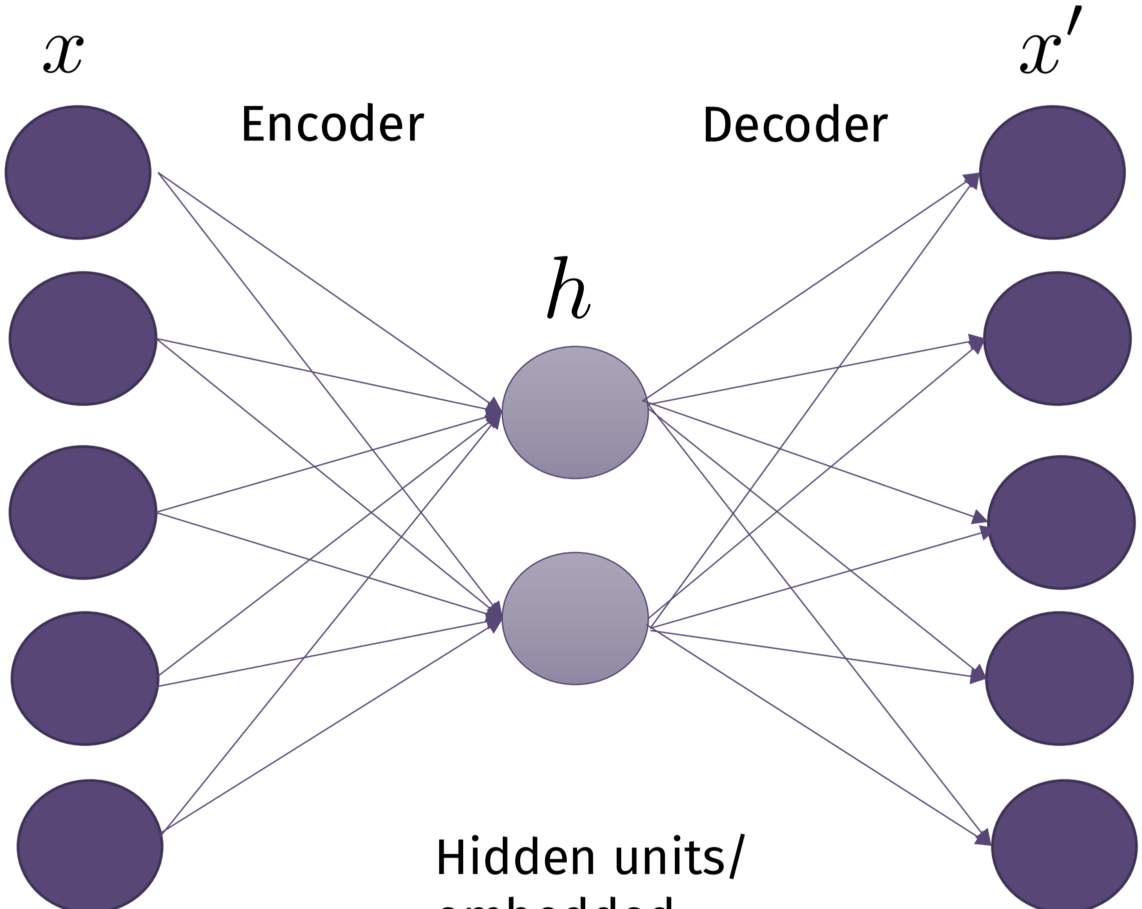
Autoencoders are an **unsupervised** approach for learning a **lower-dimensional** feature representation

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



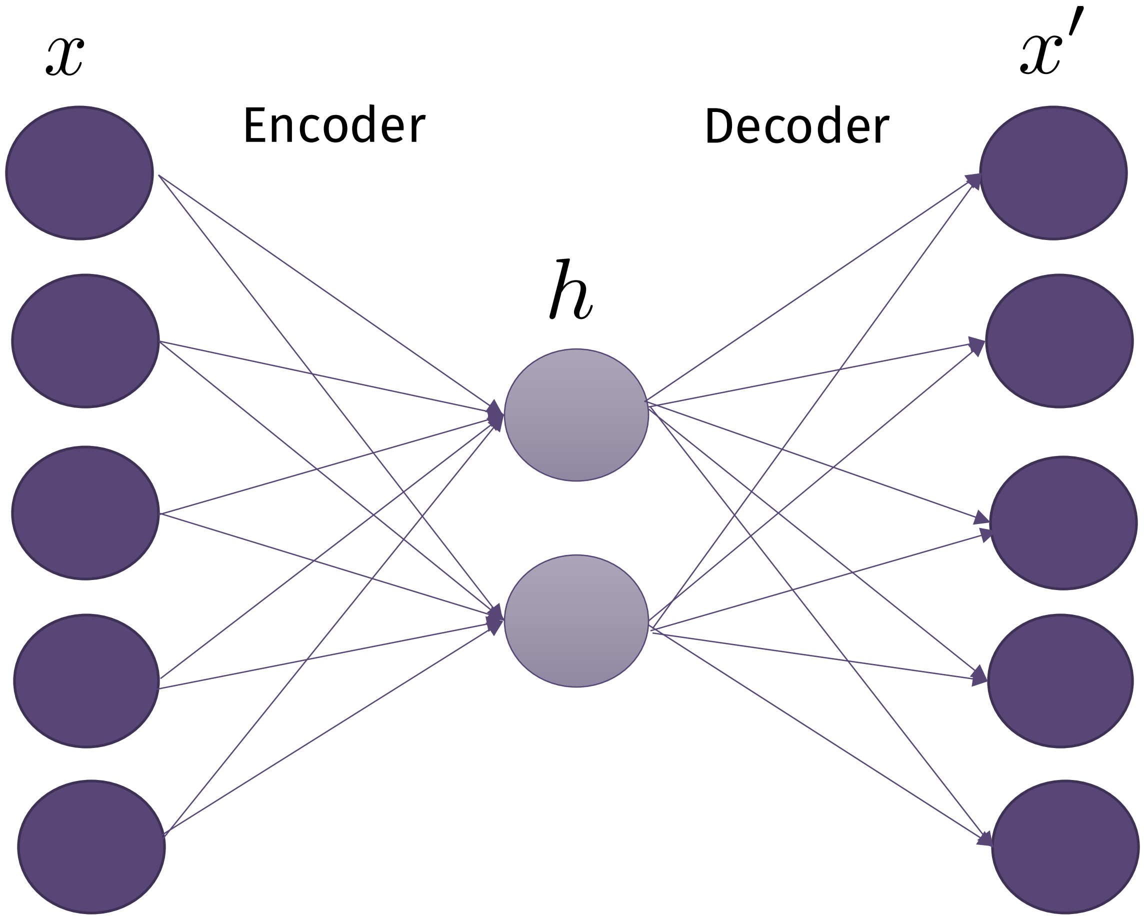
Input

Hidden units/ embedded space/latent space

Output = reconstructed input

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!)

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

Autoencoders and PCA

If we do not use non-linear activations and use a loss function based on MSE

$$L(x, x') = ||x - x'||_2^2 = \sum_i (x_i - x_i')^2$$

we obtain something very similar to PCA

A difference is that the latent dimensions will not be necessarily orthogonal and will have (more or less) the same variance

Undercomplete autoencoder

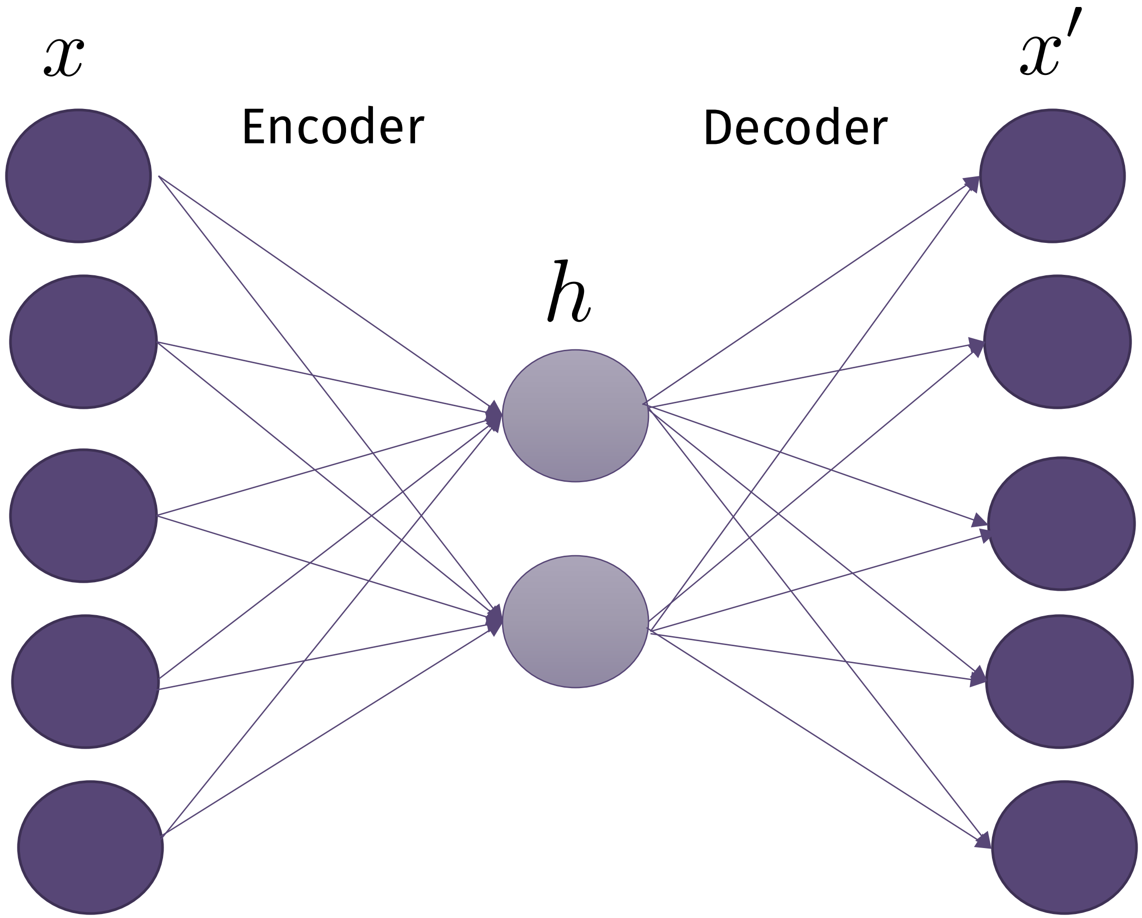
- A further constraint connecting this approach to PCA is to force h to have a smaller dimension than x

- It is commonly known as **undercomplete autoencoder:** learning an undercomplete representation forces the autoencoder **to capture the most salient features**

- Giving too much capacity to the model, <u>it fails to learn anything useful</u> (simple copy...)

Basic autoencoder

Beyond deterministic functions

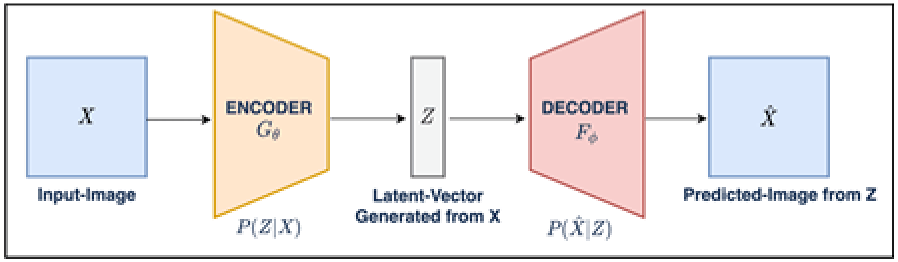


$$p_{encoder}(h|x)$$
$$p_{decoder}(x|h)$$

UniGe | MaLGa

Autoencoders

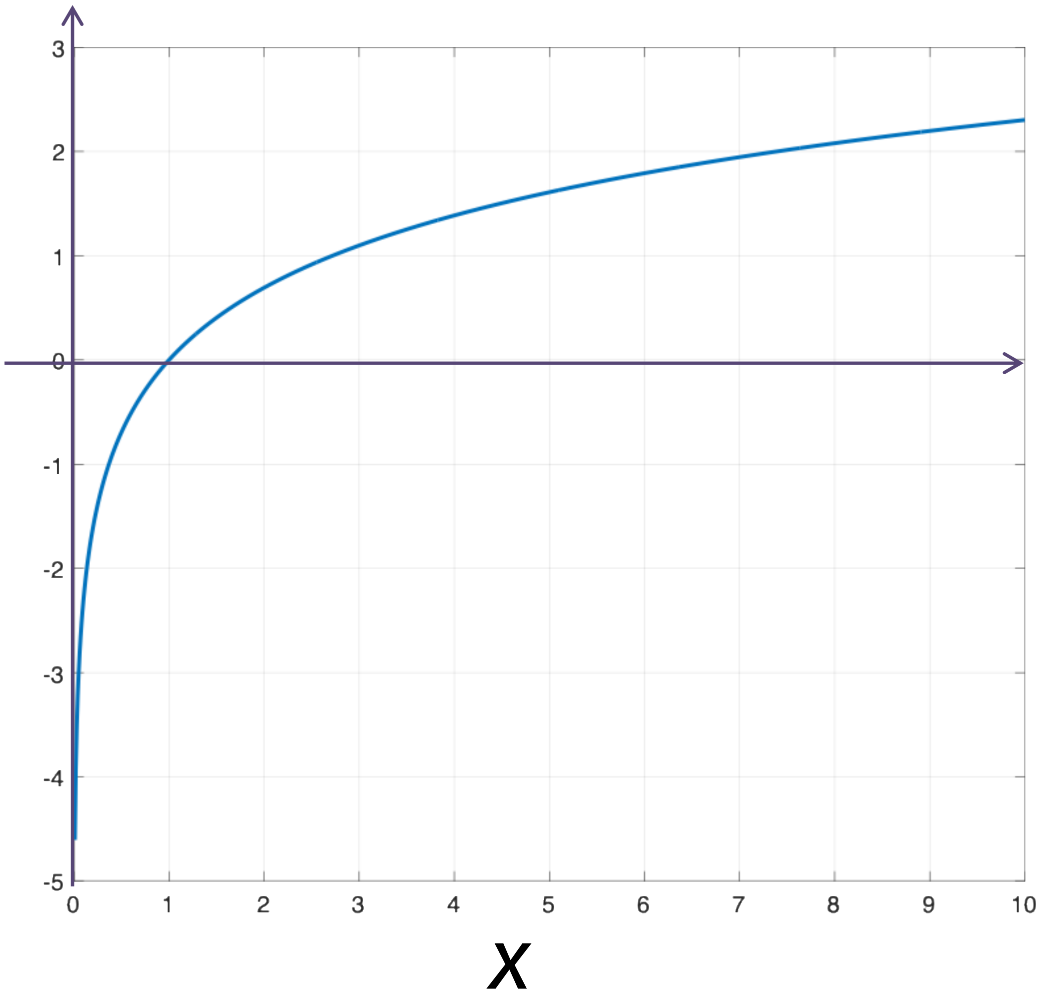
More in general



$$loss : \underbrace{\mathbb{E}_{P_\phi(Z|X)}[\log P_\theta(\hat{X}|Z)]}_{\text{reconstruction error}}$$

- Each sample is modeled as a point in the latent space
- **No** Regularizer:
  - Close points not necessarily similar once decoded
  - Exist points of the latent space not meaningful

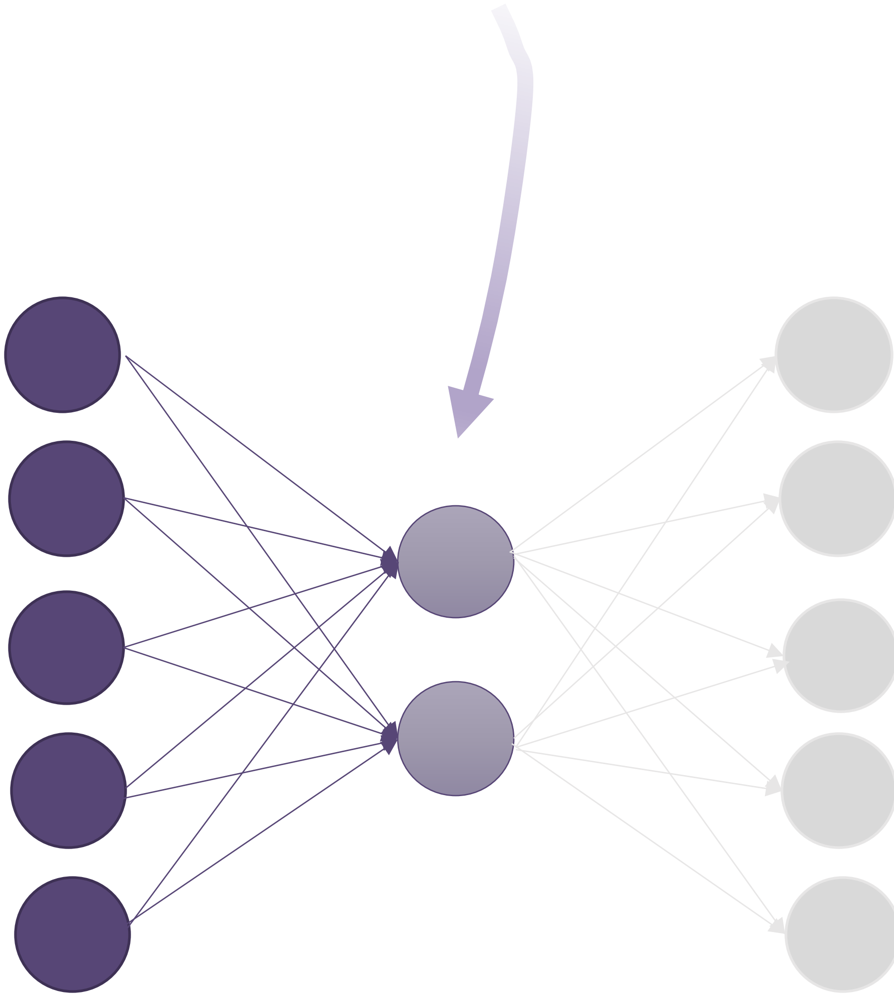
Interlude: logarithm



$y = log(x)$

UniGe | MaLGa

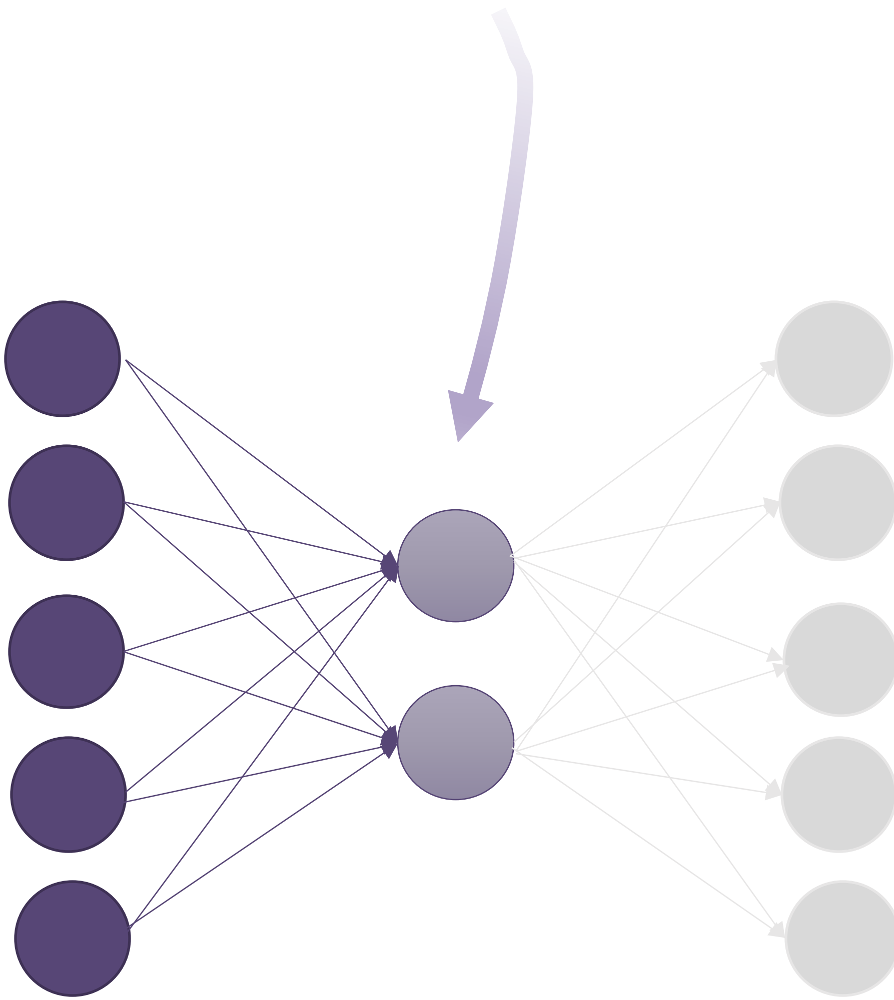
How can we use the latent space?



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

- Why do we need it?

Embedding or latent variables



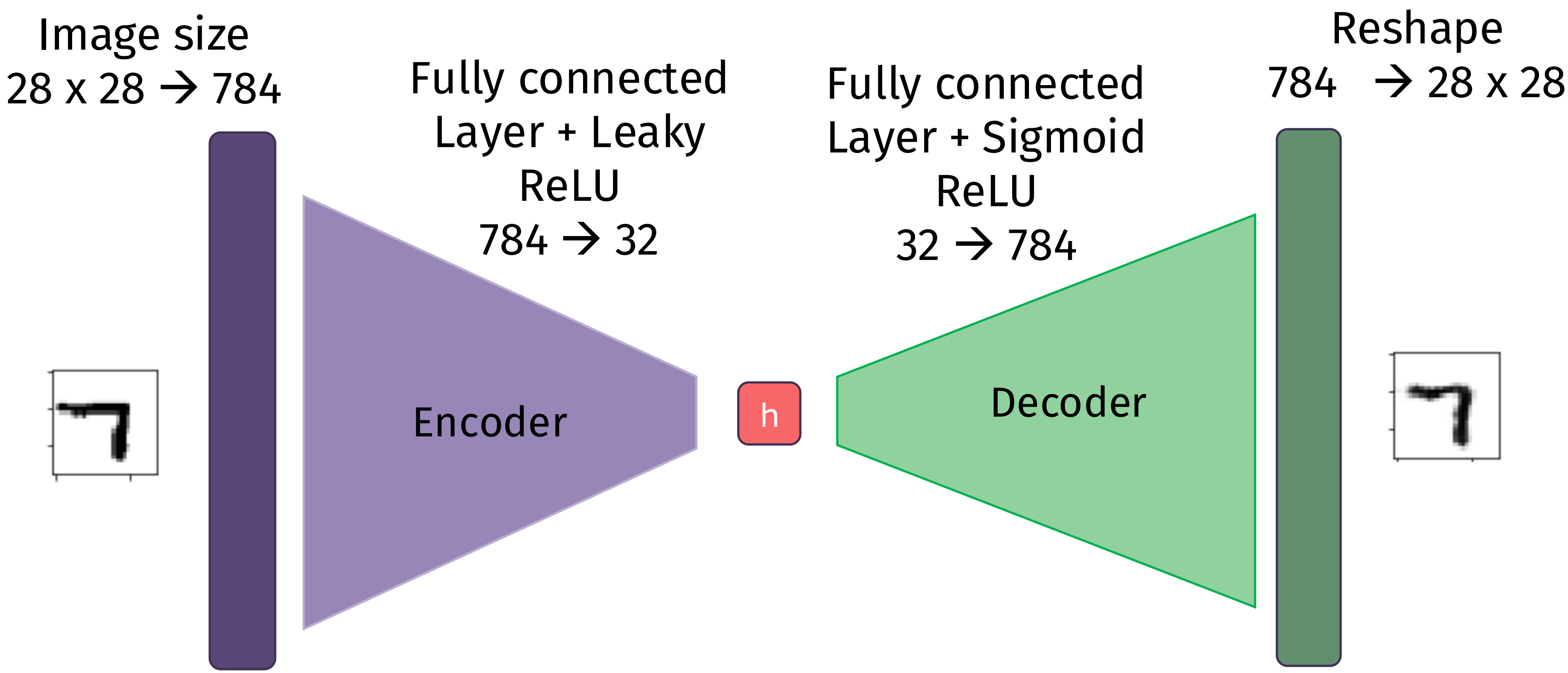
- Autoencoders can be seen as an **unsupervised** approach for learning a **lower-dimensional** feature representation

- After training, you can disregard the output and **use embedding as inputs** to classic machine learning methods

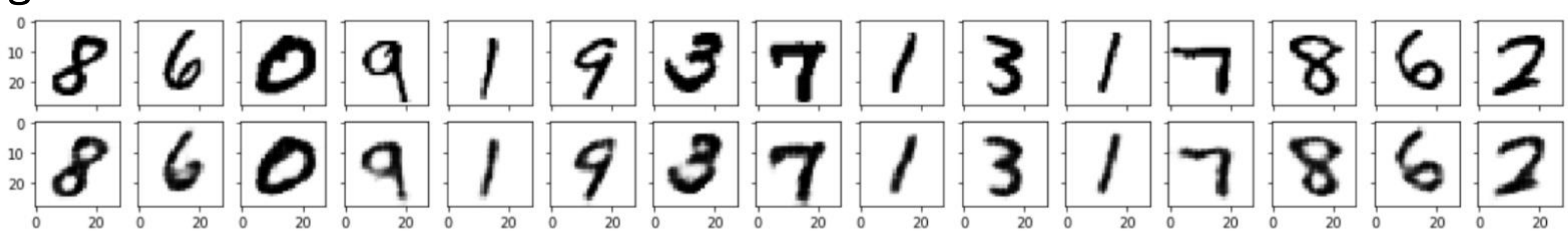
- **Transfer learning**: train autoencoders on large datasets and fine tune on your (smaller) dataset

- **Visualization** (projecting the embeddings in lower a dimensional space)

Autoencoders: an example



Original



Reconstructed

Dimensionality of the latent space

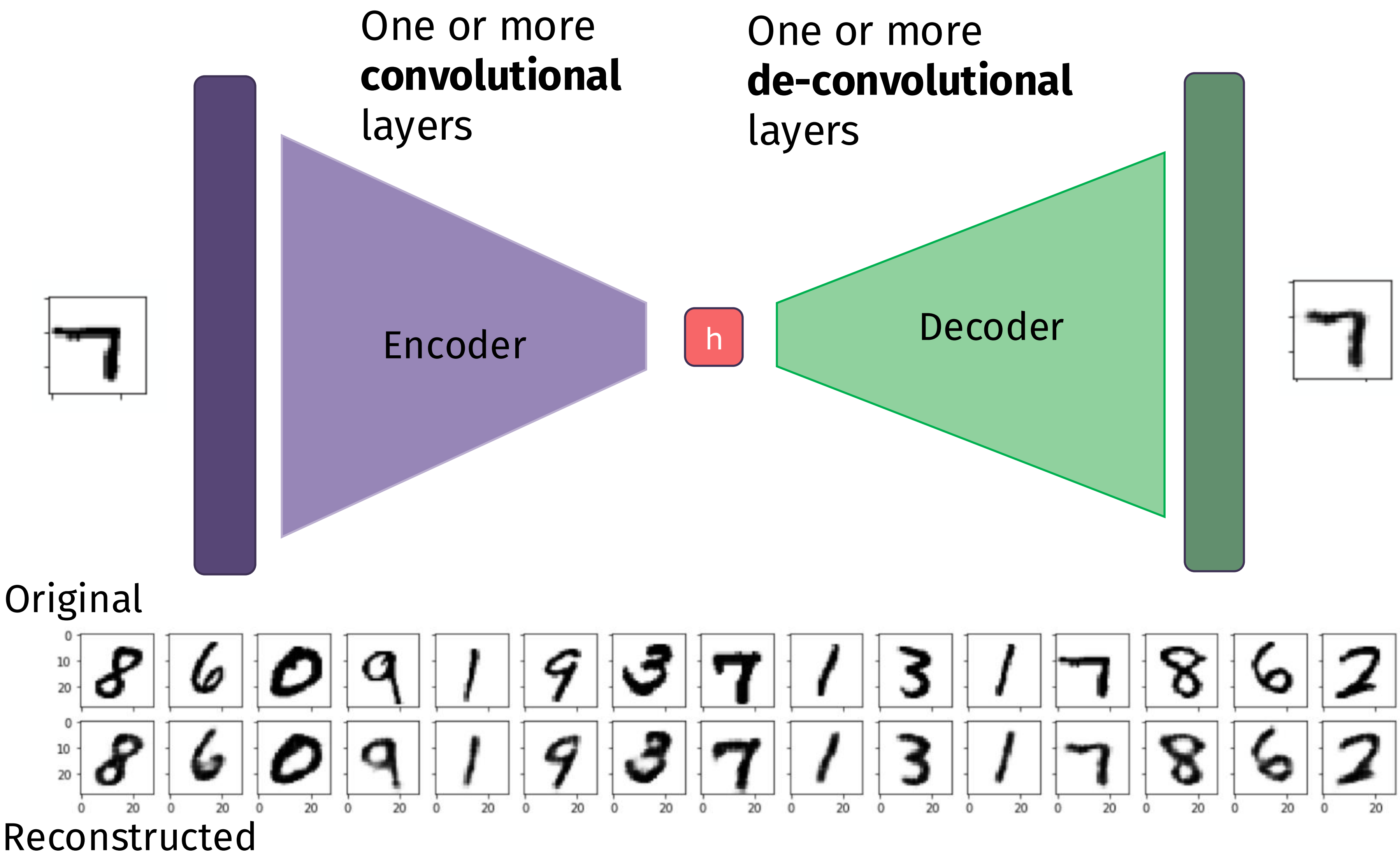
reconstruction quality

2D Latent Space

5D Latent Space

Autoencoding compresses!

Convolutional autoencoders: the concept



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

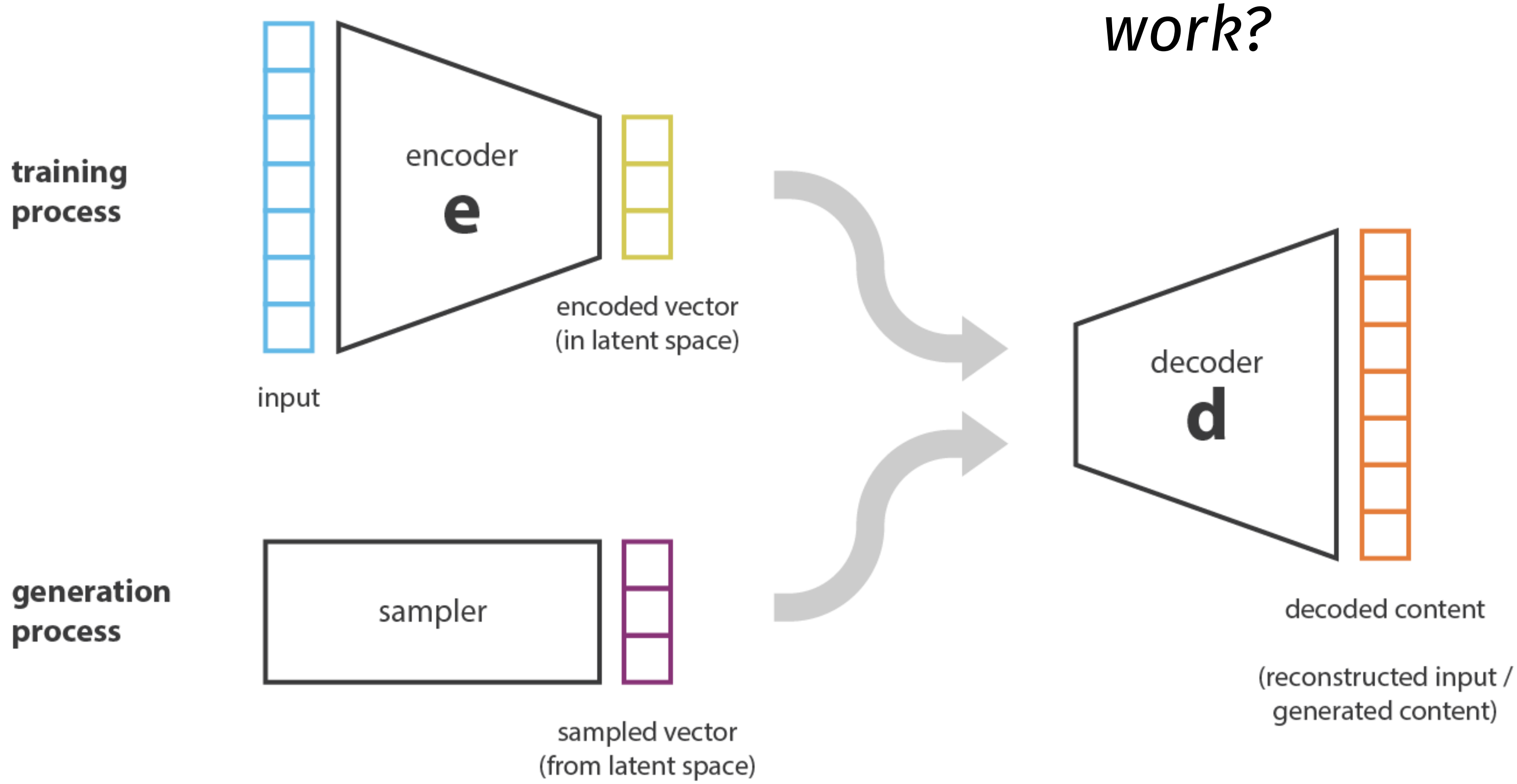
$$L(x, g(f(x)) + \lambda ||h||_1$$

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

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

Autoencoder as data generator

Does it work?



Picture from https://towardsdatascience.com/understanding-variational-autoencoders-vaes-f70510919f73

A latent space with no structure

- Difficult to ensure a priori an organization of the latent space that can allow for a generative process
- To produce latent space with a structure we may resort to the use of variational auto-encoders

Variational autoencoders

- A variational autoencoder (VAE) is an autoencoder in which some good properties in the latent space are ensured
- Instead of encoding an input into a point in the latent space, a VAE encodes it as a distribution over the latent space.

Autoencoders vs VAE

- <u>Simple autoencoders</u>

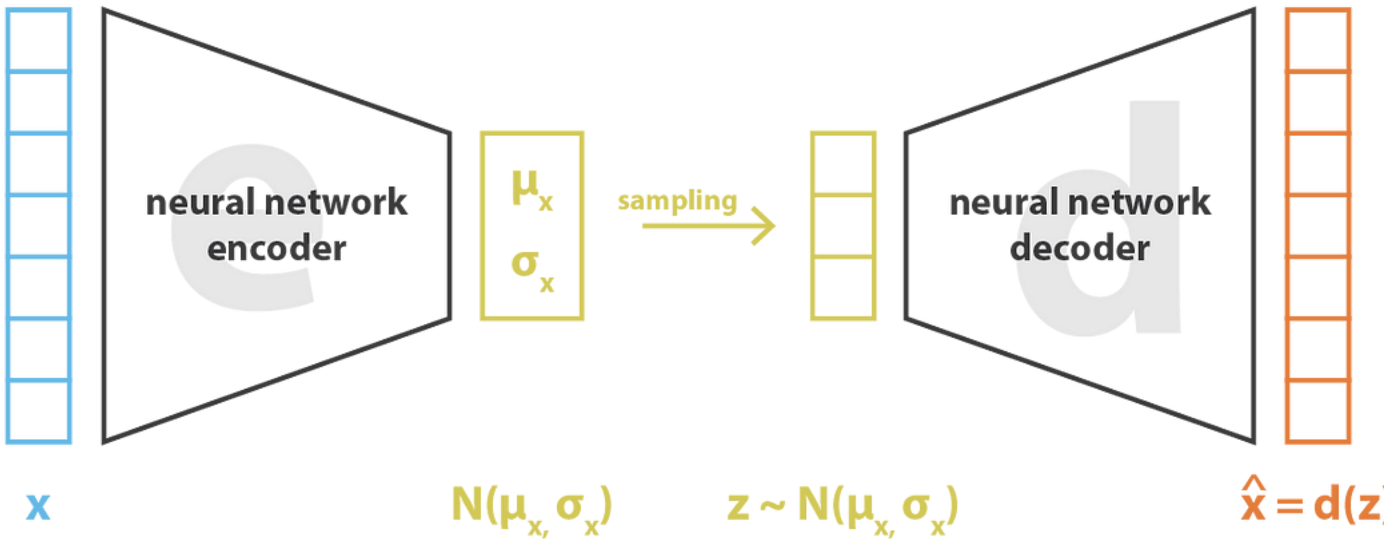
  Input x → encoding → latent representation z = e(x) → decoding → reconstruction of input d(z)

- <u>VAE</u>

  Input x → encoding → latent distribution p(z|x) → sampling → sampled latent representation z ~ p(z|x) → decoding → reconstruction of input d(z)

- The p distributions are chosen to be normal: the encoder can learn mean and the covariance matrix

Variational Autoencoders



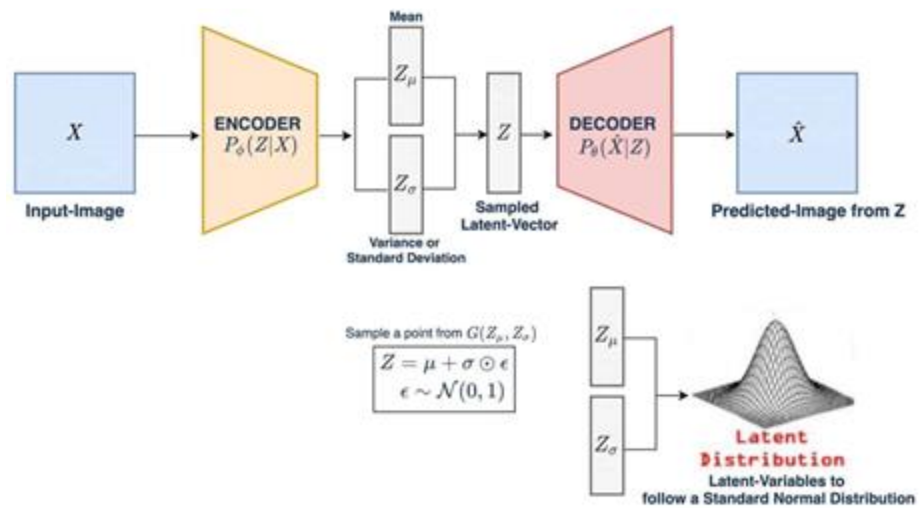
$$L(x, x') = ||x - x'||^2 + KL(N(\mu_x, \sigma_x), N(0, 1))$$

Picture from https://towardsdatascience.com/understanding-variational-autoencoders-vaes-f70510919f73

Variational Autoencoder (VAE)

More in general

$$\underbrace{\mathbb{E}_{P_\phi(Z|X)}[\log P_\theta(\hat{X}|Z)]}_{\text{reconstruction error}} - \underbrace{\beta KL(P_\phi(Z|X)||\mathcal{N}(0,1))}_{\text{regularizer}}$$

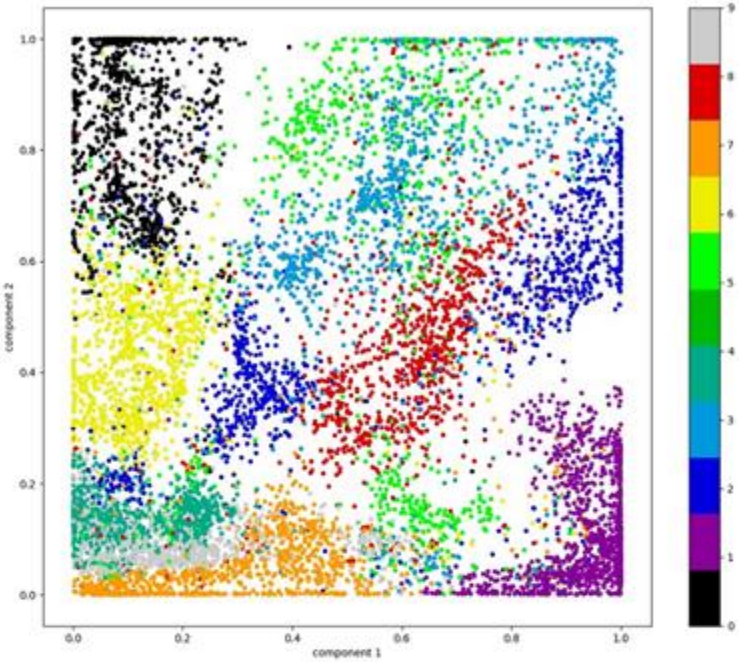


- Each sample is modelled as $\mathcal{N}(0,1)$
- Regularizer add properties:
  - *Continuity*: close points give similar content once decoded
  - *Completeness*: every point of the latent space should be meaningful

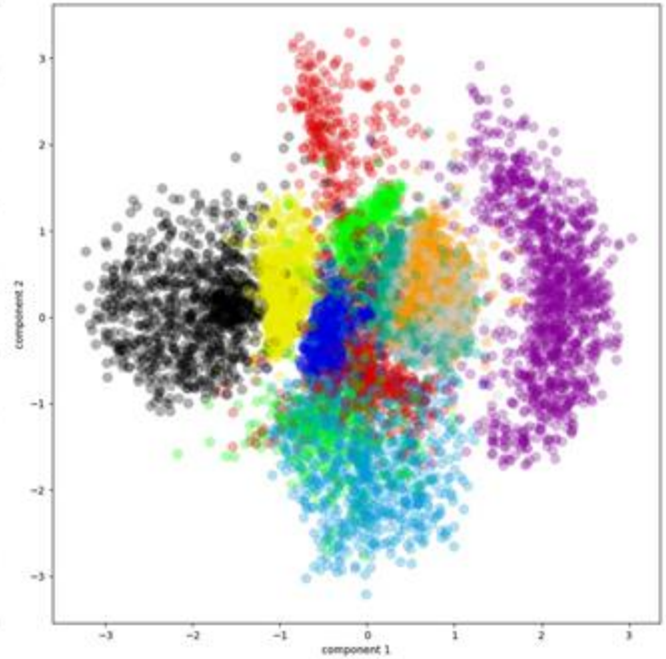
https://openreview.net/pdf?id=Sy2fzU9gl

Latent representation

Autoencoder



Variational autoencoder

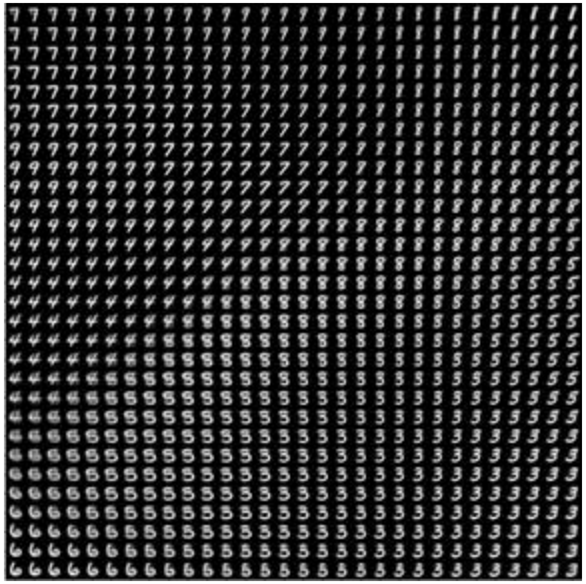


UniGe | MaLGa

Generate new samples from latent space

Autoencoder

Variational autoencoder



An example of use:
disentanglement learning

Disentangled representations

Intuition

Learn a representation h that separates the distinct and informative Factors of Variations (FoVs) in the data, so that

"A change in a single underlying factor of variation leads to a change in a single factor in the learned representation"

Disentangled representations

Intuition

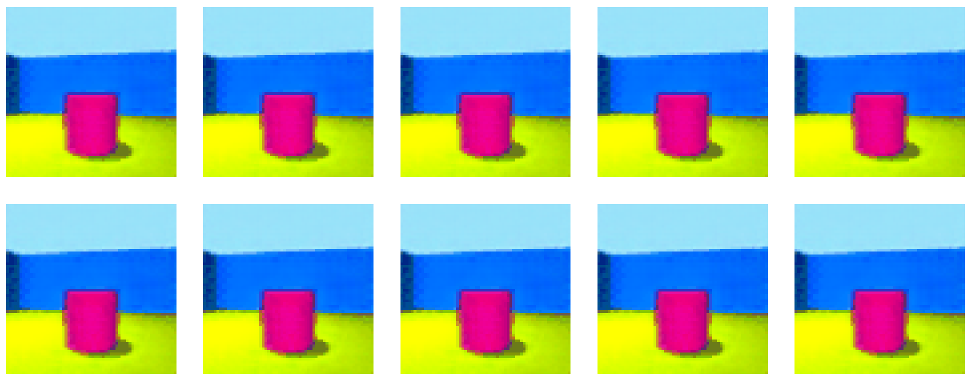
Floor color

Wall color

Object color

Object scale

Object shape



Camera position

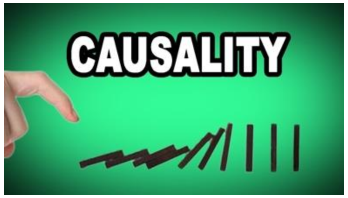
Disentangled representations

Properties

- **Modularity:** *a factor influences only a portion of the representation* ← achievable if the FoVs are independent
- **Compactness:** *the portion of the representation affected by a FoV should be as small as possible (ideally, only one dimension)*
- **Completeness:** *all FoVs are encoded in the representation*

→ Disentanglement favours interpretability

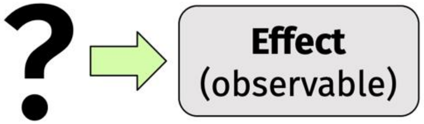
Motivations

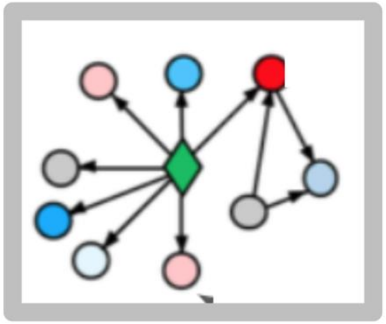


A parenthesis on causality

Causality refers to a framework to model and learn causal relationships between variables, events, or tasks

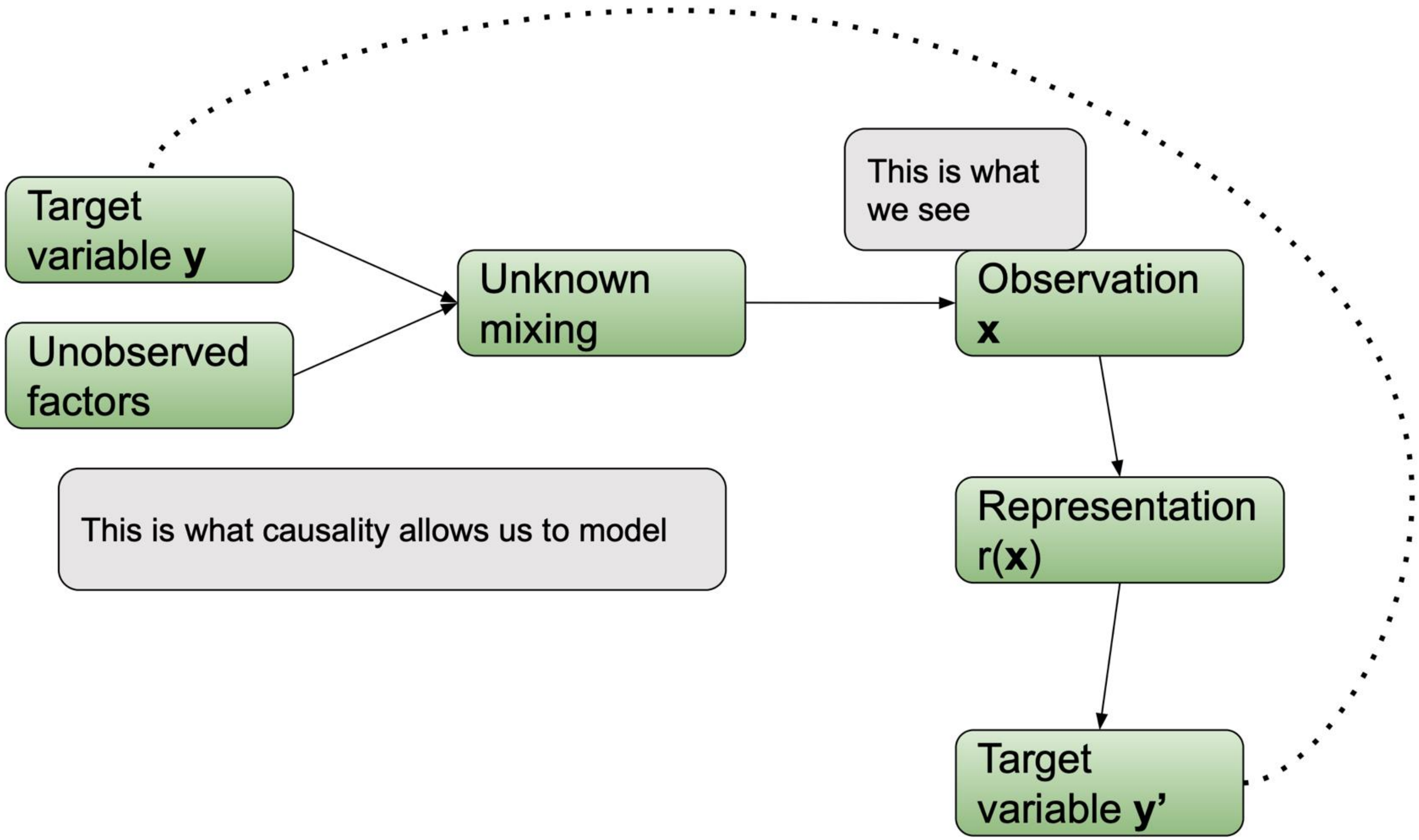
C*ausal models* contain the mechanisms giving rise to the observed statistical dependences between variables and data and allows to model distribution shifts through the notion of interventions





Motivations

A parenthesis on causality



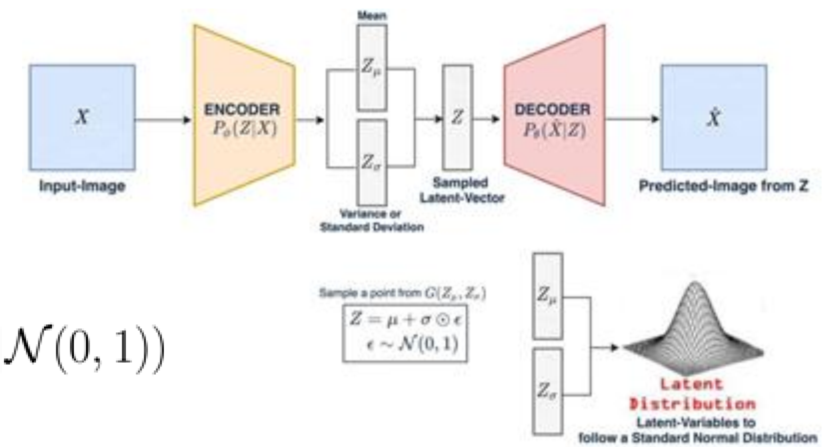
Motivations

A parenthesis on causality

- In a modular representation of the world (where the modules correspond to physical causal mechanisms), many modules are expected to behave similarly across different tasks and environments.
- When learning a causal model, fewer examples may be required to adapt to a new task/environment, as most knowledge (the modules) can be reused without further training

→ **Beneficial for problem-to-problem generalization**

Beta-VAE



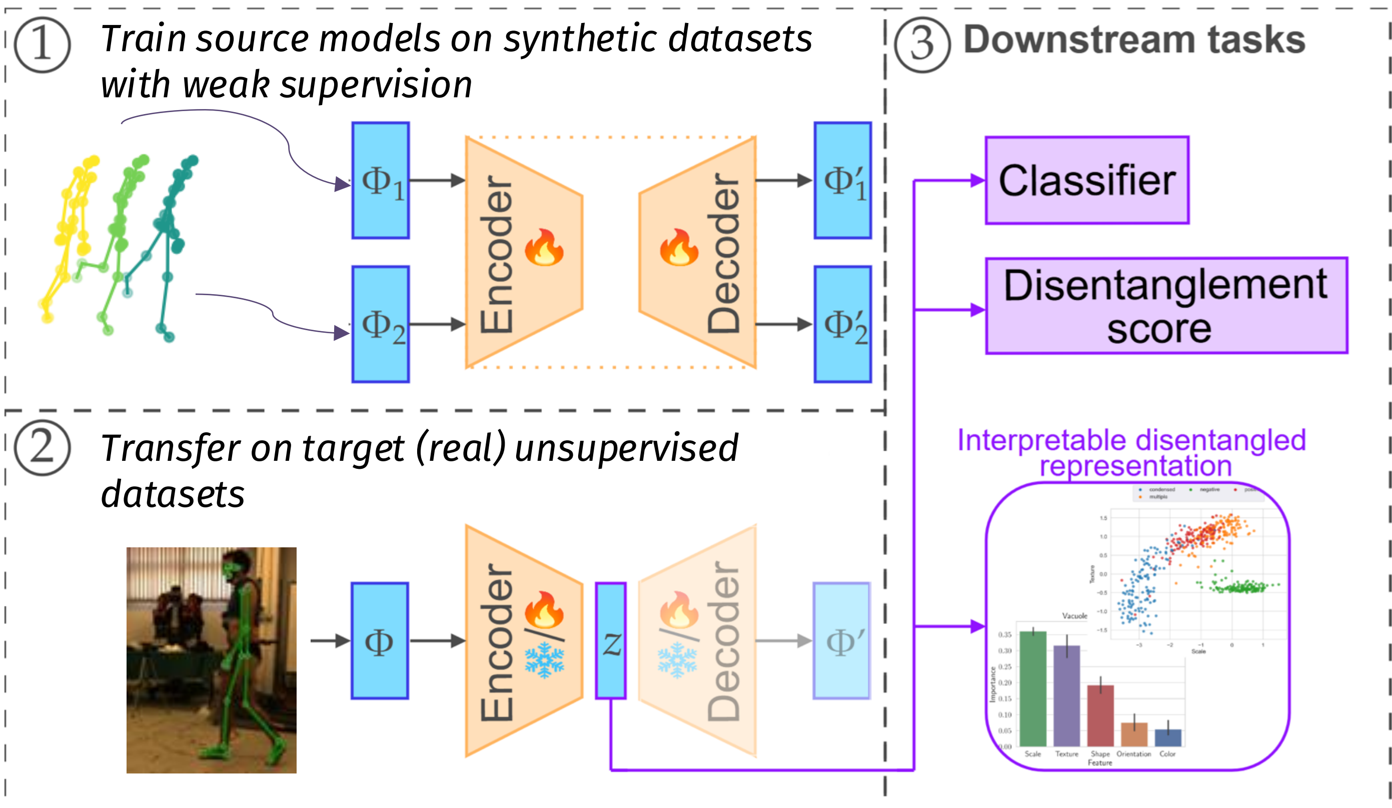
$$loss : \mathbb{E}_{P_\phi(Z|X)}[\log P_\theta(\hat{X}|Z)] - \beta KL(P_\phi(Z|X)||\mathcal{N}(0,1))$$

- With $\beta > 1$ the model is pushed to learn a more compact latent representation of the data.
- It does not allow correlations among the factors or hierarchies over them.
- Reconstruction quality must be sacrificed.
- Purely unsupervised

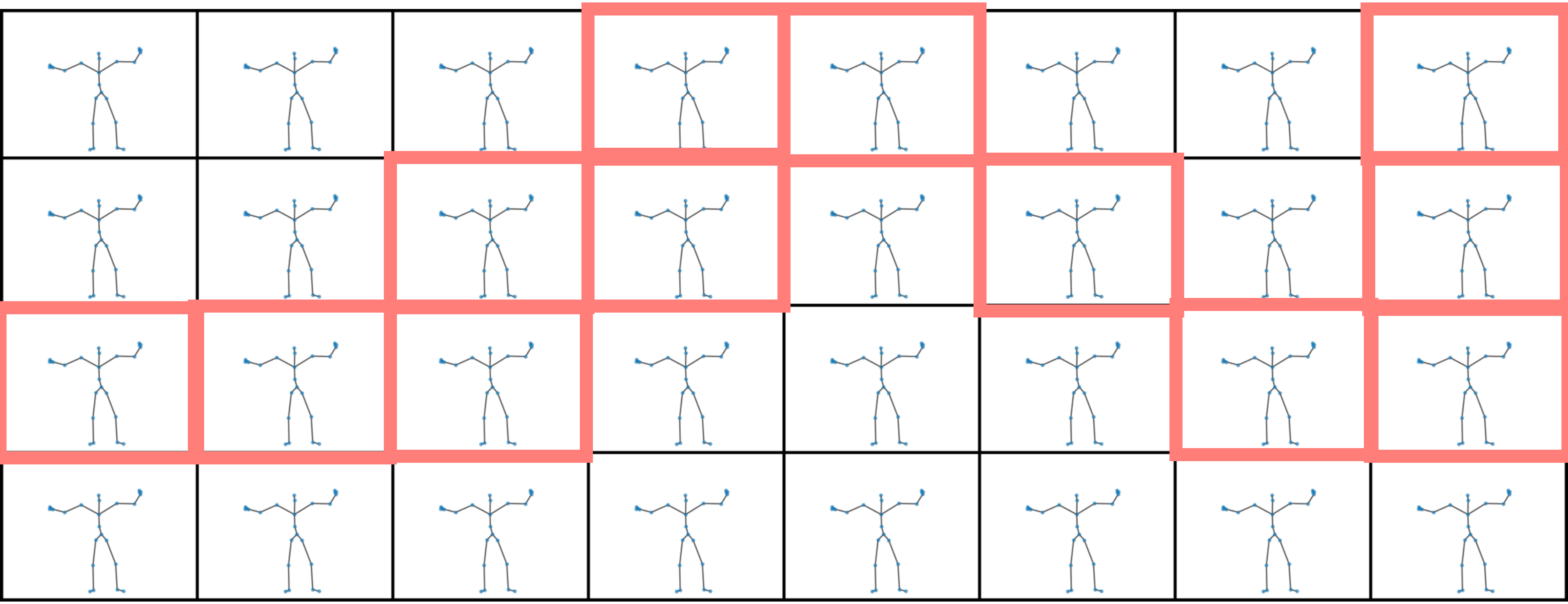
UniGe | MaLGa

https://openreview.net/pdf?id=Sy2fzU9gl

An example of disentangled representations



An example of disentangled representations



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...

A turing test

A

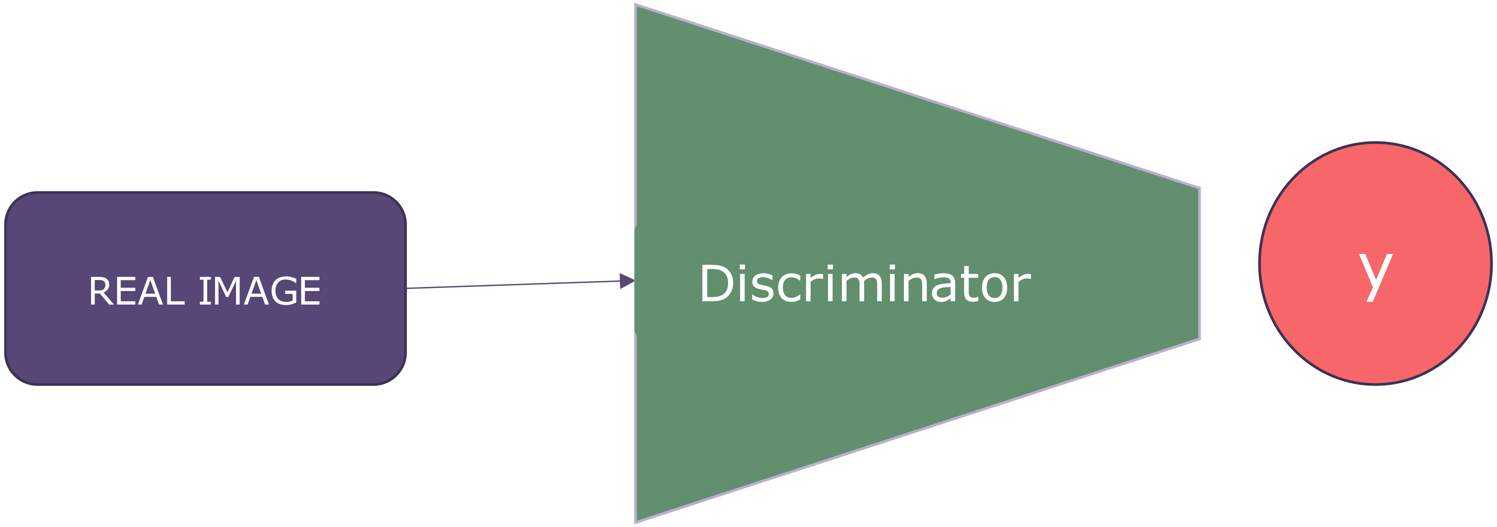
B

Which one is real?

GANs

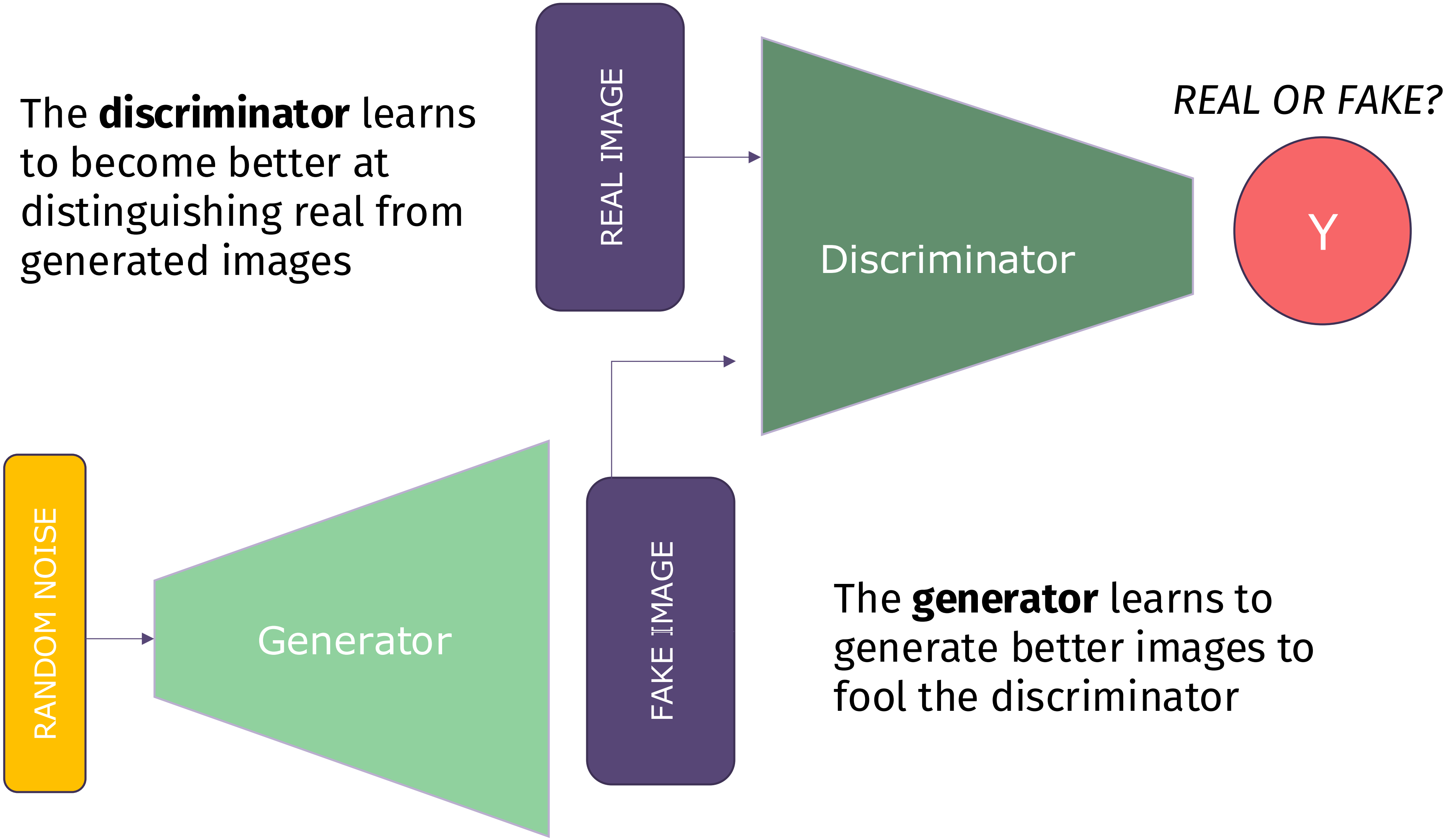
- A **generator network** directly produces samples
- Its adversary, the **discriminator network**, attempts to distinguish between samples drawn from the training data and samples drawn from the generator
- The discriminator estimates a probability values evaluating how likely is that the sample is 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

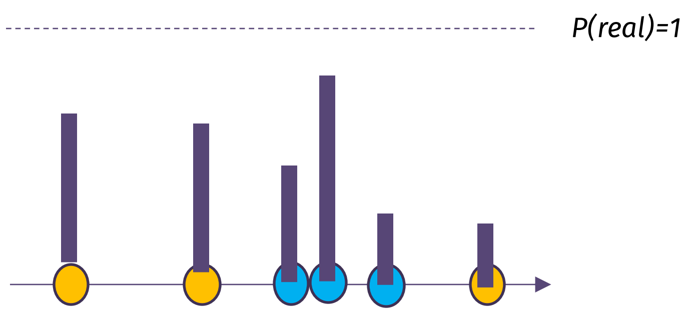
Intuition

GENERATOR

Intuition

GENERATOR

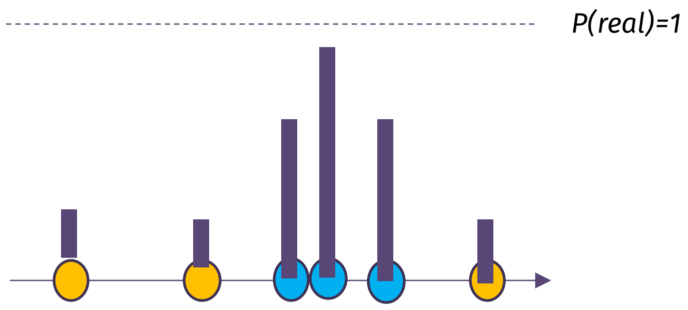
DISCRIMINATOR



Intuition

GENERATOR

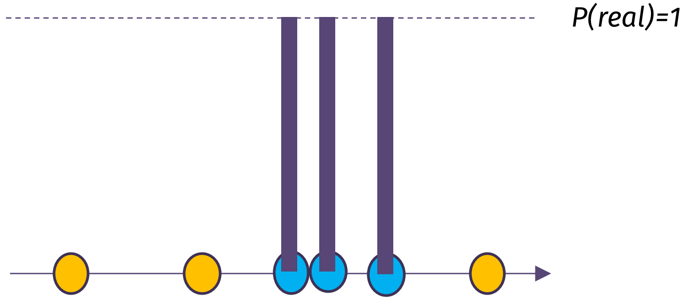
DISCRIMINATOR



Intuition

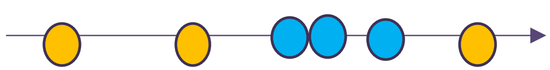
GENERATOR

DISCRIMINATOR

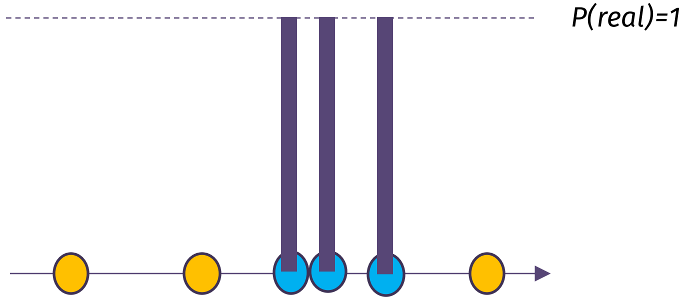


Intuition

GENERATOR

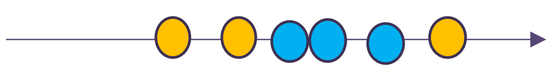


DISCRIMINATOR

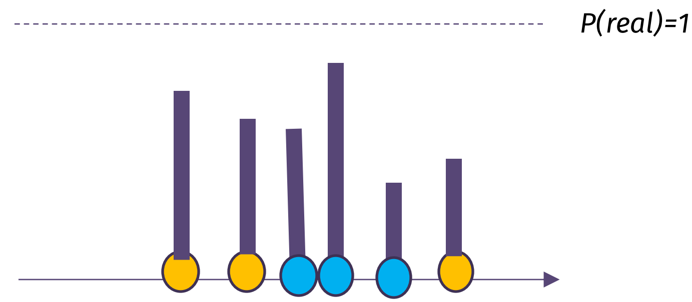


Intuition

GENERATOR

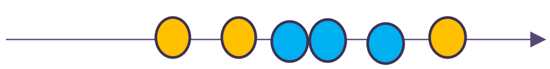


DISCRIMINATOR

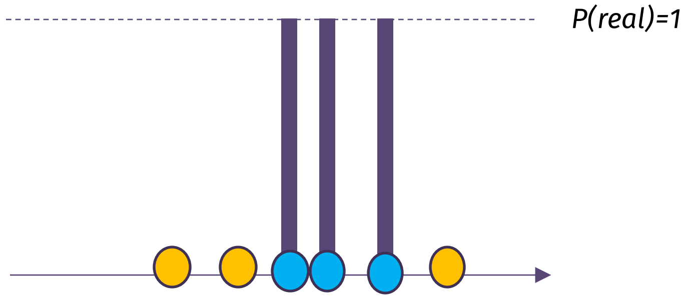


Intuition

GENERATOR

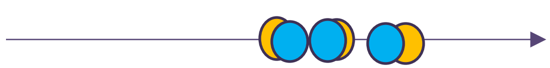


DISCRIMINATOR

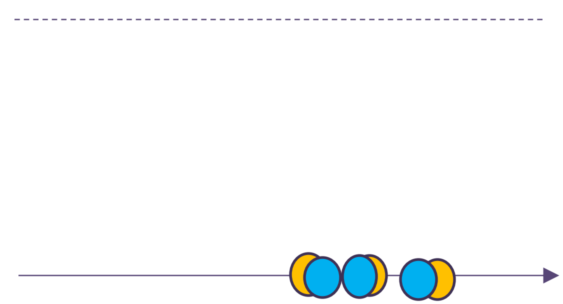


Intuition

GENERATOR



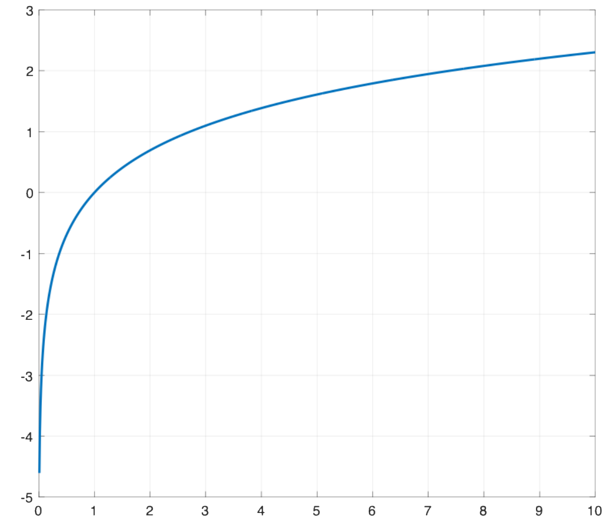
DISCRIMINATOR



P(real)=1

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

GAN model and training



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

Two-player Minmax game with value function:

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

The **Discriminator** wants D(x) = 1 and D(G(z))=0 → tries to maximize V

The **Generator** wants D(G(z))=1 → tries to minimize V

Minmax is solved through alternating gradient descent → the parameters θ_g and θ_d 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

UniGe

MaLGa