**Assignment 2**

# Learning latent space representations

**and application to image generation**

Benjamin Negrevergne, Alexandre Vérine

PSL University – Paris Dauphine – Équipes *MILES*

MILES
Machine Intelligence and Learning Systems

# Outline

1. What is a good representation
2. Learning representations with Deep Learning
3. Making sense of learned representations
4. Synthetic data generation
   GANS

# Reptile classification challenge

**Task:** classify pictures of crocodiles and alligators

- **Representation 1: features**

$$\begin{bmatrix} beast\_color & = & Light \\ beast\_size & = & Large \end{bmatrix} \longrightarrow f_1 \longrightarrow \text{crocodile}$$

# Reptile classification challenge

**Task:** classify pictures of crocodiles and alligators

- **Representation 1: features**

$$\begin{bmatrix} beast\_color & = & Dark \\ beast\_size & = & Small \end{bmatrix} \longrightarrow f_1 \longrightarrow \text{alligator}$$

# Reptile classification challenge

**Task:** classify pictures of crocodiles and alligators

- **Representation 1: features**

$$\begin{bmatrix} beast\_color & = & Dark \\ beast\_size & = & Small \end{bmatrix} \rightarrow f_1 \rightarrow \text{alligator}$$

- **Representation 2: Use raw pixel data**

$$\rightarrow f_2 \rightarrow \text{crocodile}$$

# Reptile classification challenge

**Task:** classify pictures of crocodiles and alligators

- **Representation 1: features**

$$\begin{bmatrix} beast\_color & = & Dark \\ beast\_size & = & Small \end{bmatrix} \rightarrow f_1 \rightarrow \text{alligator}$$

- **Representation 2: Use raw pixel data**

$$\rightarrow f_2 \rightarrow \text{alligator}$$

# Reptile classification challenge

**Task:** classify pictures of crocodiles and alligators

- **Representation 1: features**

$$\begin{bmatrix} beast\_color & = & Dark \\ beast\_size & = & Small \end{bmatrix} \rightarrow f_1 \rightarrow \text{alligator}$$

- **Representation 2: Use raw pixel data**

$$\rightarrow f_2 \rightarrow \text{alligator}$$

Which **representation** of the input is easier to work with ? Why ?

# What's the difference?

$f_1 : \mathbb{R}^2 \rightarrow \mathbb{R}$

Input space 1: $\mathbb{R}^2$

$f_2 : \mathbb{R}^{w \times h \times 3} \rightarrow \mathbb{R}$

Input space 2: $\mathbb{R}^{w \times h \times 3}$

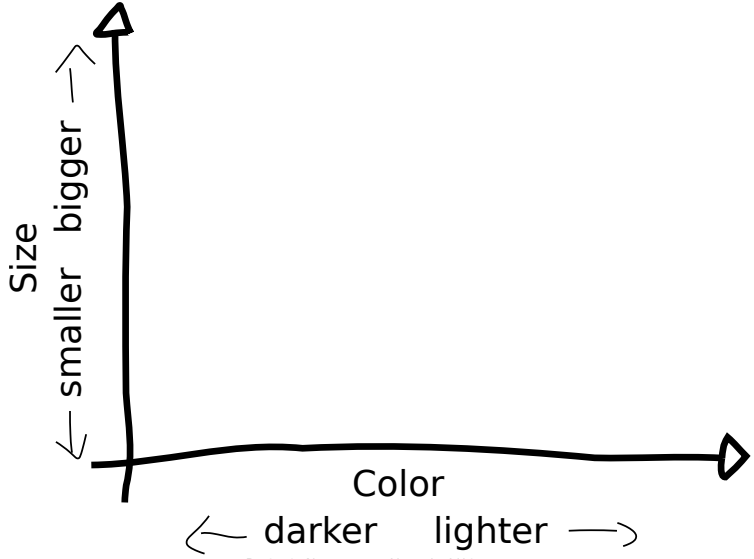
# What's the difference?

$f_1 : \mathbb{R}^2 \rightarrow \mathbb{R}$

Input space 1: $\mathbb{R}^2$

$f_2 : \mathbb{R}^{w \times h \times 3} \rightarrow \mathbb{R}$

Input space 2: $\mathbb{R}^{w \times h \times 3}$



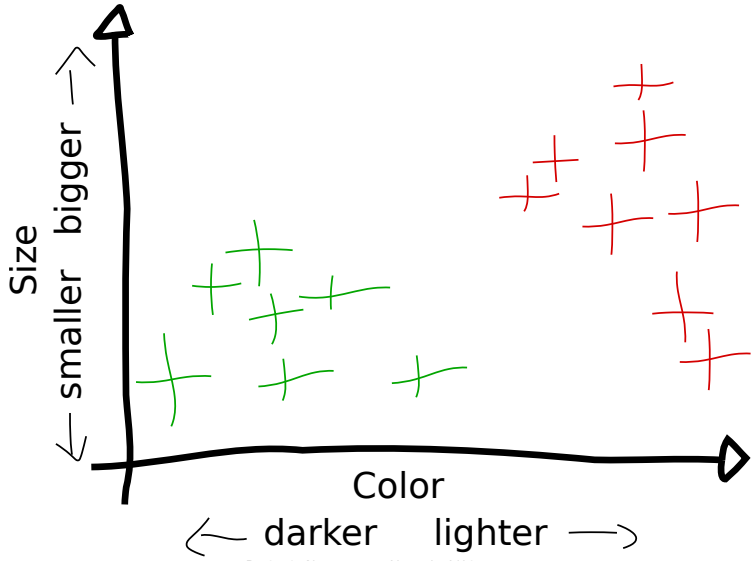
# What's the difference?

$f_1 : \mathbb{R}^2 \rightarrow \mathbb{R}$

Input space 1: $\mathbb{R}^2$

$f_2 : \mathbb{R}^{w \times h \times 3} \rightarrow \mathbb{R}$

Input space 2: $\mathbb{R}^{w \times h \times 3}$



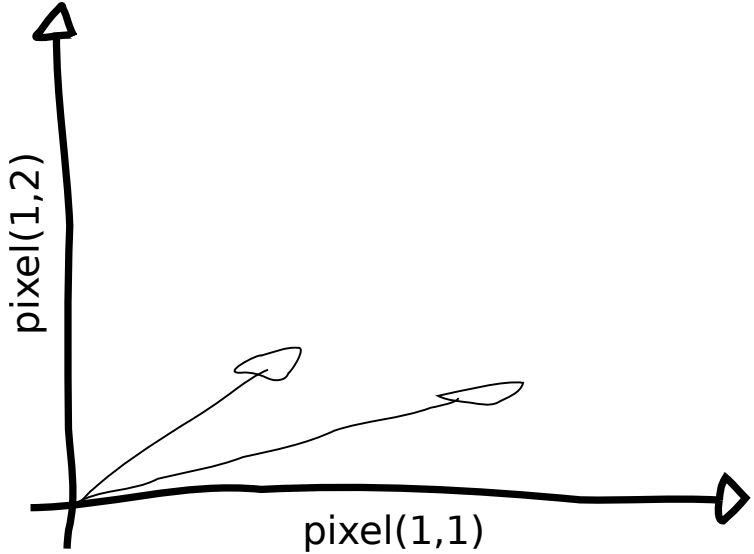
# What's the difference?

$f_1 : \mathbb{R}^2 \rightarrow \mathbb{R}$

Input space 1: $\mathbb{R}^2$

$f_2 : \mathbb{R}^{w \times h \times 3} \rightarrow \mathbb{R}$

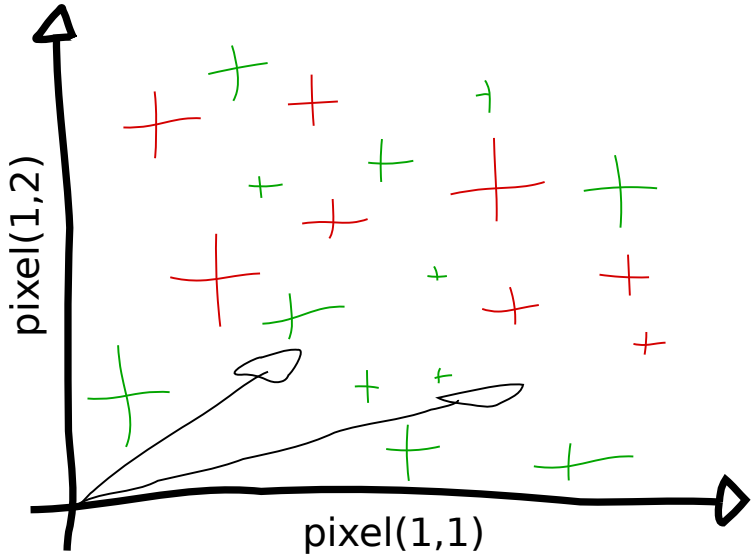
Input space 2: $\mathbb{R}^{w \times h \times 3}$



# What's the difference?

$f_1 : \mathbb{R}^2 \rightarrow \mathbb{R}$
Input space 1: $\mathbb{R}^2$

$f_2 : \mathbb{R}^{w \times h \times 3} \rightarrow \mathbb{R}$
Input space 2: $\mathbb{R}^{w \times h \times 3}$



# Pro/cons

- **Representation 2: raw pixel data**
  - \+ Contains all the information available
  - − Input data points are not (nearly) linearly separable
    Features are individually non-discriminant
  - − Difficult to process with simple models

- **Representation 1: high-level features**
  - \+ Input data points are (almost) linearly separable
    Individual features are highly discriminant
  - \+ Can be processed with simple (linear) models
  - − Requires expertise and manual labor to be built

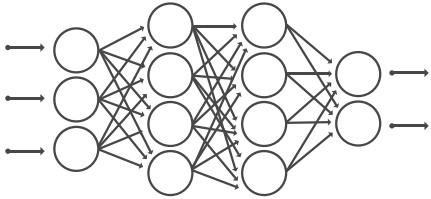
# Pro/cons

- **Representation 2: raw pixel data**
  - \+ Contains all the information available
  - − Input data points are not (nearly) linearly separable
    Features are individually non-discriminant
  - − Difficult to process with simple models

- **Representation 1: high-level features**
  - \+ Input data points are (almost) linearly separable
    Individual features are highly discriminant
  - \+ Can be processed with simple (linear) models
  - − Requires expertise and manual labor to be built

Can we build high level features automatically ?

# Deep learning



# Deep learning

$$f = \underbrace{f_1 \circ f_2 \circ \ldots \circ f_{n-1}}_{g} \circ \underbrace{f_n}_{h}$$

**Remarks**

- $h$ is a linear classifier: $\mathcal{Z}_{n-1} \to \mathcal{Y}$
  - Data points **must be** linearly separable in $\mathcal{Z}_{n-1}$
- Data points $x$ **are not** linearly separable in the input space $\mathcal{X}$

# Deep learning

$$f = \underbrace{f_1 \circ f_2 \circ \ldots \circ f_{n-1}}_{g} \circ \underbrace{f_n}_{h}$$

**Remarks**

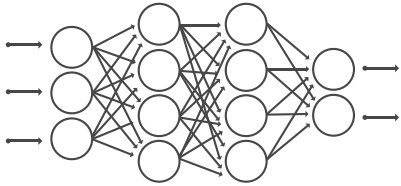
- $h$ is a linear classifier: $\mathcal{Z}_{n-1} \to \mathcal{Y}$
  - ▶ Data points **must be** linearly separable in $\mathcal{Z}_{n-1}$
- Data points $x$ **are not** linearly separable in the input space $\mathcal{X}$

**What is $g$?**

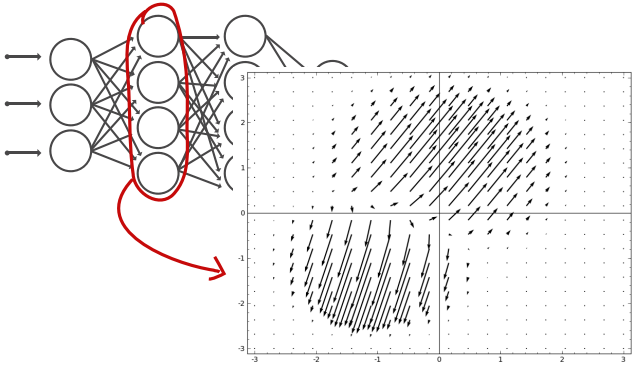
- a function $g : \mathcal{X} \to \mathcal{Z}_{n-1}$
- such that data points are linearly separable in $\mathcal{Z}_{n-1}$

▶ a representation of the point in $\mathcal{X}$ that is adequate for the task at hand

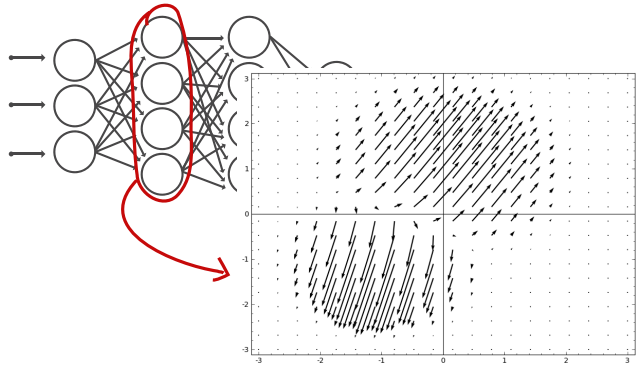
# Learning deep representations

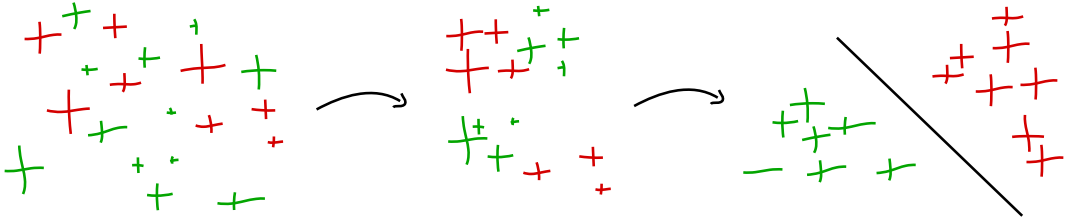


# Learning deep representations

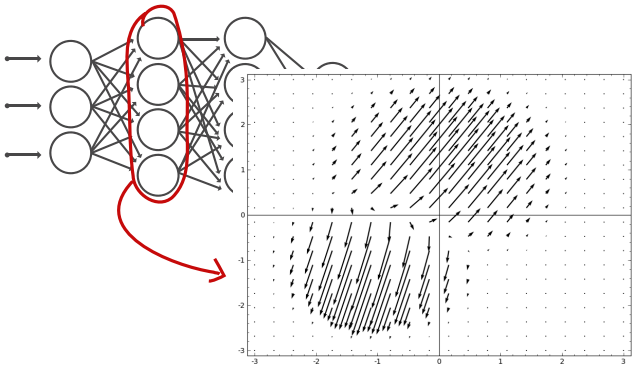


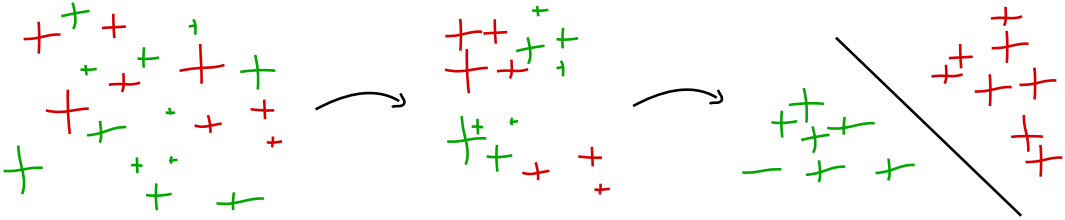
# Learning deep representations





# Learning deep representations





Note: the decision boundary is a complex high dimensional hypersurface in $\mathcal{X}$

# Deep representations: first lessons

- DNN learn how to projet inputs into a latent space
- The structure of the latent space is useful for the task at hand.
- Given an input $x \in \mathcal{X}$ we say that $g(x)$ is an **embedding** of $x$, i.e. continuous vector representations of input data (image, text, graph ...)

# Outline

1. What is a good representation
2. Learning representations with Deep Learning
3. Making sense of learned representations
4. Synthetic data generation
   GANS

# Outline

1. What is a good representation
2. Learning representations with Deep Learning
3. Making sense of learned representations
4. Synthetic data generation
   GANS

# Image embeddings

- $D$: a database with 80 000 pictures.
- $f = g \circ h$: a classifier trained on object recognition
  $g$ non-linear function, $h$ linear classifier
- $x$ a random picture from the internet

$$x_1 = \underset{x' \in D}{\arg\min} \, ||g(x) - g(x')||$$

$$x_2 = \underset{x' \in D \setminus \{x_1\}}{\arg\min} \, ||g(x) - g(x')||$$

# Image embeddings

- $D$: a database with 80 000 pictures.
- $f = g \circ h$: a classifier trained on object recognition
  $g$ non-linear function, $h$ linear classifier
- $x$ a random picture from the internet

$$x_1 = \arg\min_{x' \in D} ||g(x) - g(x')||$$

$$x_2 = \arg\min_{x' \in D \setminus \{x_1\}} ||g(x) - g(x')||$$

$x$

# Image embeddings

- $D$: a database with 80 000 pictures.
- $f = g \circ h$: a classifier trained on object recognition
  $g$ non-linear function, $h$ linear classifier
- $x$ a random picture from the internet

$$x_1 = \arg\min_{x' \in D} ||g(x) - g(x')||$$

$$x_2 = \arg\min_{x' \in D \setminus \{x_1\}} ||g(x) - g(x')||$$

$x$

$x_1$

$x_2$

# Query by image (2)

$x$

# Query by image (2)

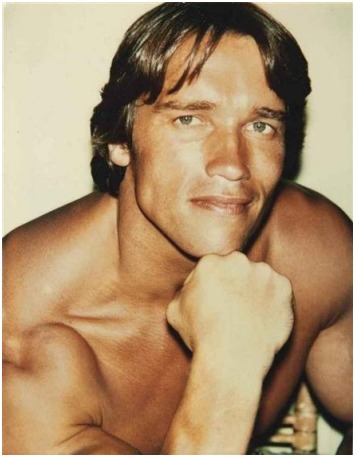
$x$

$x_1$

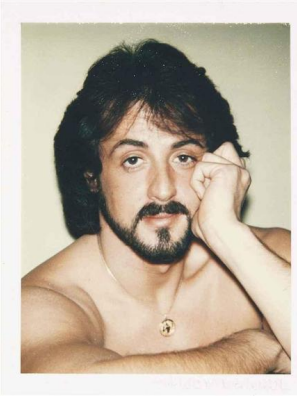
$x_2$

# Query by image (2)

$x$



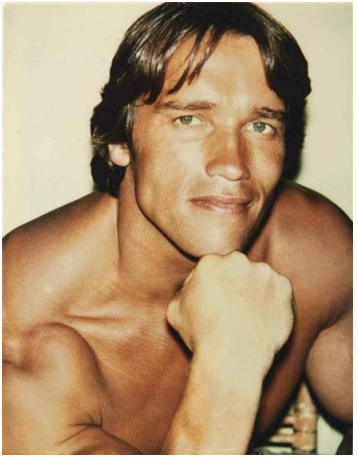
$x_1$



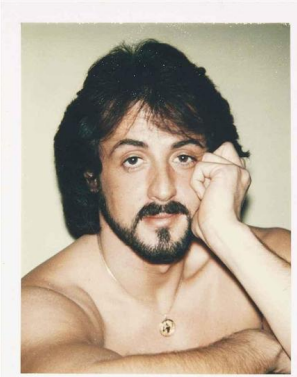
$x_2$

# Query by image (2)

$x$



$x_1$



$x_2$

**Result:** The distance in the latent space seems to be meaningful!

# Word embeddings

We can train (monolingual) *word embeddings*,
i.e. representations trained to predict well words that appear in its context (ref here)

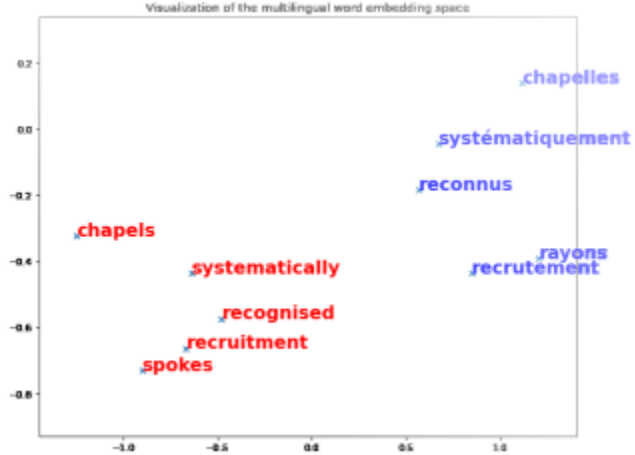



Image generated using pretrained embeddings available here. *En avant guiGuan team*, 2021.

# Word embeddings

We can train (monolingual) *word embeddings*,
i.e. representations trained to predict well words that appear in its context (ref here)

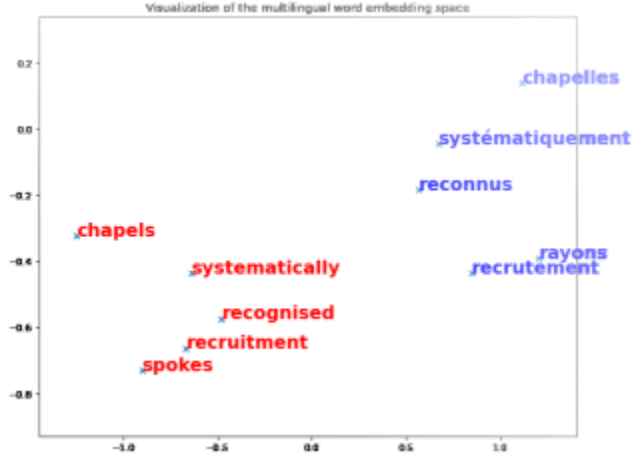
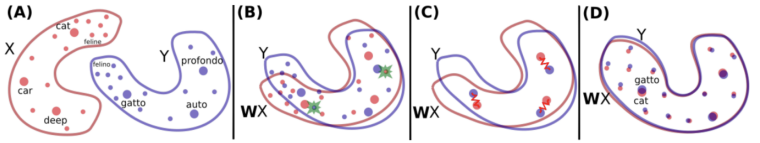



Image generated using pretrained embeddings available here. *En avant guiGuan team*, 2021.

▶ The structure between word embeddings is preserved across languages

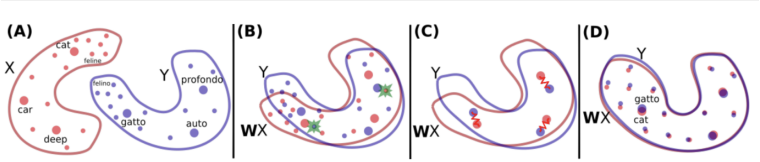
# Word to word translation using word embeddings




Exploit linear transformations and rotations to translate a word.

$$Y = \mathbf{W}X$$

# Word to word translation using word embeddings




Exploit linear transformations and rotations to translate a word.

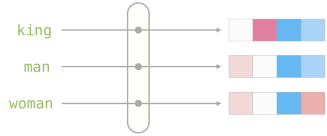
$$Y = \mathbf{W}X$$

Learn **W**

- with a parallel corpus (e.g. supervised dataset FR-EN)
- without a parallel corpus using a GAN

# Text manipulation with word embeddings

The embedding space is geometric!

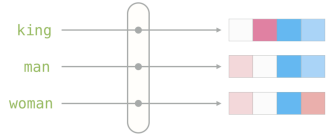



The embeddings space is geometric:

$$v(king) - v(man) + v(woman) = \quad ?$$

# Text manipulation with word embeddings

The embedding space is geometric!




The embeddings space is geometric:

$$v(king) - v(man) + v(woman) = \quad \text{queen}$$

# Meaningful features

*Dimensionality reduction by learning an invariant mapping.*
Hadsell, R., Chopra, S., LeCun, Y. CVPR (2006)

Learns a mapping $g$ that maps inputs with **few controlled variations** to a **low dimensional space**

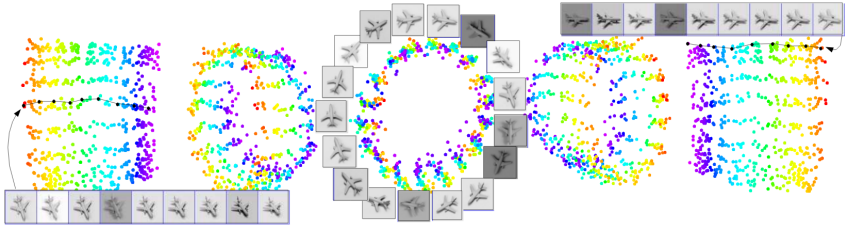
- all pictures are pictures of planes with different poses
- 9 different elevations and 18 different azimuth (orientation).
- input pictures are projected into a low 3-dimensional space

# Meaningful features

*Dimensionality reduction by learning an invariant mapping.*
Hadsell, R., Chopra, S., LeCun, Y. CVPR (2006)

Learns a mapping $g$ that maps inputs with **few controlled variations** to a **low dimensional space**

- all pictures are pictures of planes with different poses
- 9 different elevations and 18 different azimuth (orientation).
- input pictures are projected into a low 3-dimensional space

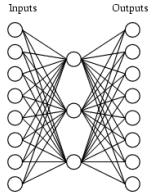


**Result:** There is a clear relation between learned features (spacial coordinates) and desired features (elevation, azimuth)

# Outline

1. What is a good representation
2. Learning representations with Deep Learning
3. Making sense of learned representations
4. Synthetic data generation
   GANS

# Autoencoders



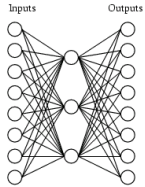

A target function:

| Input | | Output |
|---|---|---|
| 10000000 | $\rightarrow$ | 10000000 |
| 01000000 | $\rightarrow$ | 01000000 |
| 00100000 | $\rightarrow$ | 00100000 |
| 00010000 | $\rightarrow$ | 00010000 |
| 00001000 | $\rightarrow$ | 00001000 |
| 00000100 | $\rightarrow$ | 00000100 |
| 00000010 | $\rightarrow$ | 00000010 |
| 00000001 | $\rightarrow$ | 00000001 |

Can this be learned??

# Autoencoders

A network:




Learned hidden layer representation:

| Input | | Hidden Values | | | | Output |
|---|---|---|---|---|---|---|
| 10000000 | → | .89 | .04 | .08 | → | 10000000 |
| 01000000 | → | .01 | .11 | .88 | → | 01000000 |
| 00100000 | → | .01 | .97 | .27 | → | 00100000 |
| 00010000 | → | .99 | .97 | .71 | → | 00010000 |
| 00001000 | → | .03 | .05 | .02 | → | 00001000 |
| 00000100 | → | .22 | .99 | .99 | → | 00000100 |
| 00000010 | → | .80 | .01 | .98 | → | 00000010 |
| 00000001 | → | .60 | .94 | .01 | → | 00000001 |

slides Tom Mitchell

# Data generation with autoencoders

How to generate new synthetic data?

- **First Idea:**
  - sample a vector $z$ in the latent space $\mathcal{Z}$
  - decode $z$ into an image $x = d(z)$

- **Problem?**

# Data generation with autoencoders

How to generate new synthetic data?

**First Idea:**

- sample a vector $z$ in the latent space $\mathcal{Z}$
- decode $z$ into an image $x = d(z)$

**Problem**

How can we sample $z$ so that $d(z)$ is mapped to a point in the *data manifold*?

# Data generation with autoencoders

How to generate new synthetic data?

## First Idea:

- sample a vector $z$ in the latent space $\mathcal{Z}$
- decode $z$ into an image $x = d(z)$

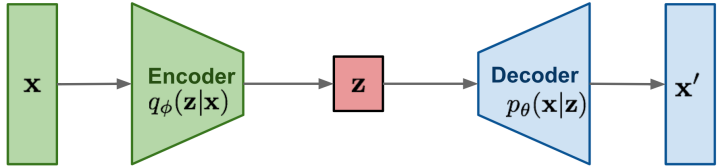
## Problem

How can we sample $z$ so that $d(z)$ is mapped to a point in the *data manifold*?

## Solution

- regularize the latent space so that the encoded training data in normally distributed in $\mathcal{Z}$
- sample $z$ from a normal distribution
- decode $z$ into an image $x = d(z)$

# Data generation with Autoencoders

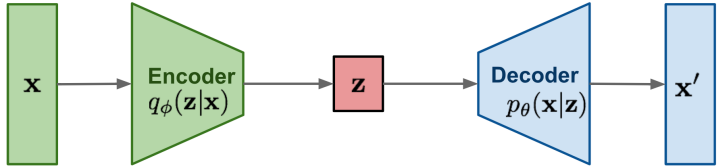



VAE, image from https://lilianweng.github.io

## Main differences with AE

- Use a probabilistic encoder
- Regularize the latent space during training

# Data generation with Autoencoders




VAE, image from https://lilianweng.github.io

## Main differences with AE

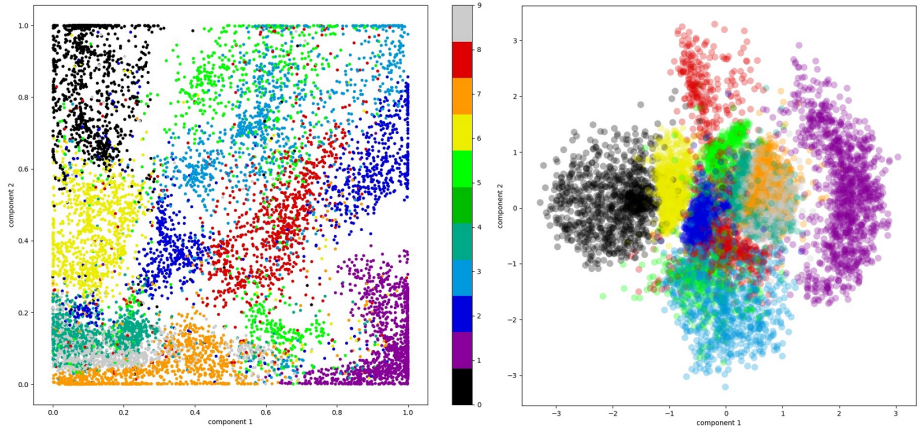
- Use a probabilistic encoder
- Regularize the latent space during training

## Training procedure

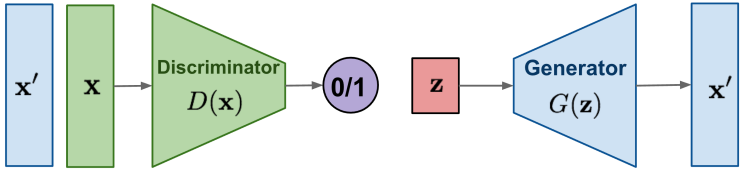
1. take a training data point $x$, obtain $\mu_x$ and $\sigma_x$ from the encoder
2. sample $z \sim \mathcal{N}(\mu_x, \sigma_x)$
3. decode $z$ into $\tilde{x}$
4. compute the loss and update parameters

$$loss(x, \tilde{x}) = \|x - \tilde{x}\| + KL(\mathcal{N}(\mu_x, \sigma_x), \mathcal{N}(0, I))$$

# AE vs. VAE



# Generative Adversarial Networks




GAN, image from https://lilianweng.github.io

## Main difference with VAE

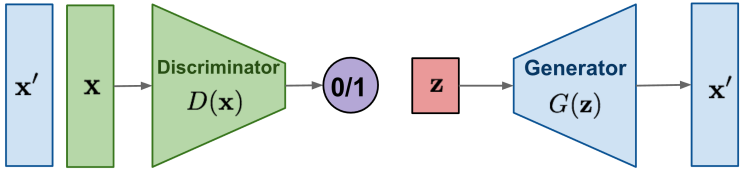
- Minimize an estimate of the divergence $\mathcal{D}(P\|\hat{P}_G)$, instead of the reconstruction error
  - $P$ is the true data distribution
  - $\hat{P}_G$ is the model distribution induced by a generator function $G : \mathcal{Z} \to \mathcal{X}$ (i.e. a decoder)
  - Since $P$ is unavailable, we use a discriminator $D : \mathcal{X} \to [0, 1]$ to estimate $\mathcal{D}(P\|\hat{P}_G)$
  - $D$ is trained to distinguish samples from $P$ and samples from $\hat{P}_G$

## Training procedure

$G$ and $D$ are trained simultaneously to solve the following min-max problem:

$$\min_G \max_D \mathbb{E}_{x_r \sim P}[\log D(x)] + \mathbb{E}_{x_g \sim \hat{P}_G}[\log 1 - D(x)]$$

# Generative Adversarial Networks



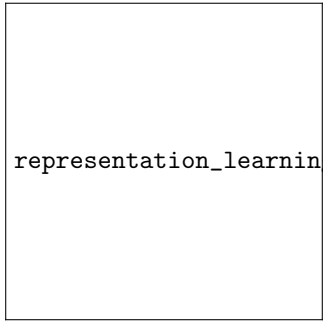

GAN, image from https://lilianweng.github.io

References:

- Generative Adversarial Nets
  https://arxiv.org/pdf/1406.2661
- Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks https://arxiv.org/abs/1511.06434

# Goal of Assignment 2

1. Train a GAN on MNIST.
2. The structure of the Generator is fixed.
3. Use different one or two possible improvements to improve the data generation.



# Possible improvements

- f-GANs
  1. f-GAN: Training Generative Neural Samplers using Variational Divergence Minimization
- WGAN
  1. Wasserstein GAN
- Rejection Sampling
  1. Discriminator Rejection Sampling
  2. Metropolis-Hastings Generative Adversarial Networks
- Latent Rejection sampling
  1. Latent reweighting, an almost free improvement for GANs
- Gradient ascent
  1. Discriminator optimal transport
  2. Refining Deep Generative Models via Discriminator Gradient Flow
  3. Your GAN is Secretly an Energy-based Model and You Should use Discriminator Driven Latent Sampling
- Classifier guidance generation
  1. MMGAN: Generative Adversarial Networks for Multi-Modal Distributions
  2. Gaussian Mixture Generative Adversarial Networks for Diverse Datasets, and the Unsupervised Clustering of Images

# Requirements Assignment 2

1. Train a vanilla GAN
2. Write a script generate.py that generate 10000 samples in the folder samples (use mine).
3. Based on these 10k samples, you will be evaluated on FID, Precision and Recall.
   Precision/Recall