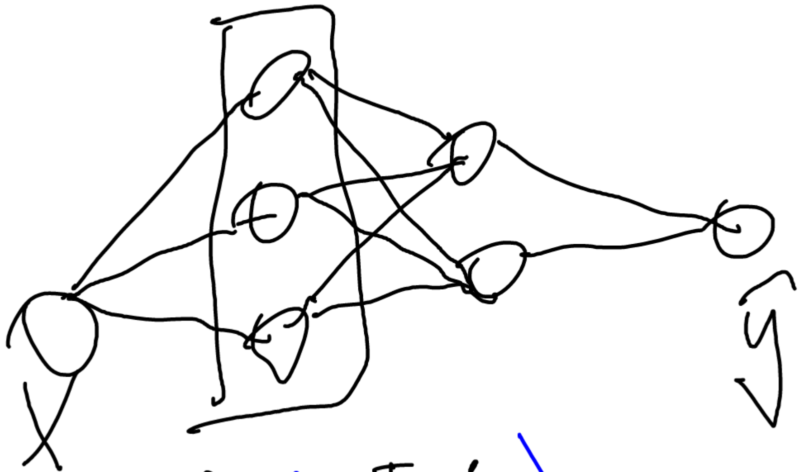
Convolutional Neural Networks (CNN)

- Feed-forward Neural Networks (Réseaux Denses) (FFNN)



$h = g(W^T + b)$

- CNN
- Recurrent Neural Networks (RNN)

Modèles Génératifs → apprentissage non supervisé

- VAE
- GAN

$\boxed{\text{CNN}}$: special kind of NN for processing data with grid-like topology.

→ processing $\boxed{\text{Image}}$

→ objects 2D.

Relies on a mathematical operation called convolution

I / The convolution operation

example: tracking the location of a space ship in a $\boxed{\text{laser sensor}}$

⊗ $x(t)$ at instant t.

measurement of $x(t)$

⇓

moyenne sur plusieurs pas de temps.

x^t

average of the measurements over several timesteps with weighting

$f^{\circ}\ \omega(a)$

$$\boxed{s(t) = \int x(a)\, w(t-a)\, da}$$

"smoothed" estimate of the position of the spaceship at time t.

$$\boxed{s(t) = (x * w)(t)}$$

opération de convolution 1D

cas continu

output
[feature map

input

kernel function

$$s(t) = \sum_{-\infty}^{+\infty} x(a)\, w(t-a) = (x * w)(t)$$

cas discret 1D

2D convolution case (discret)

$$\underbrace{S(i,j)}_{\text{feature map}} = (\underset{\substack{\uparrow\\ \text{Input 2D}}}{I} * \underset{\substack{\uparrow\\ \text{kernel}}}{k})(i,j)$$

$$\boxed{= \sum_m \sum_n I(m,n) \times K(i-m, j-n)}$$

Commutative

$$\boxed{= \sum_m \sum_n I(i-m, j-n) \times K(i,j)}$$

B/ Properties of a convolution operation

1. "sparse" interactions
2. parameter sharing
3. "equivariante" representation.

① Sparse Interactions

Kernel operator 'smaller' (in terms of dimensionality) than the Input

↳ store fewer parameters

↳ reduce memory requirements.

② Parameter Sharing

Kernel function used more than once in the model

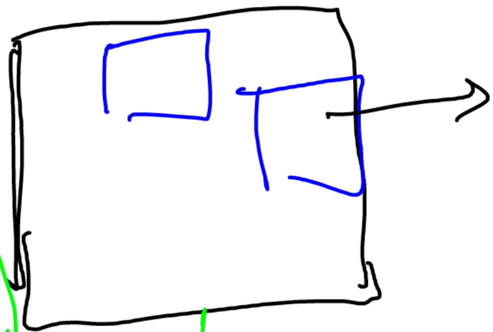
↳ At each position (2D) of the Input data.

3 Equivariance to translation

ex: a pattern in a Image can be recognized in the picture (Input data)

C/ Application of the Convolution f° to 2D object (Images)

INPUT FEATURE MAP



extracts patches from its Input feature map

(H,W,C) → colors

and applies the same transformation to all the patches

OUTPUT FEATURE MAP

3D (H,W,F) → number of filters.

↳ encode a specific aspect of the Input data.

ex: MNIST dataset

noir et blanc

Image en couleur

Input data (28,28,1)

1 → $\boxed{3}$ RGB

Opération de convolution

→ f° de kernel with one parameter, the number of filters. (= 32)

OUTPUT FEATURE MAP

(28,28,32)

[:, :, n] → 2D spatial map of the response of this filter over the input.

* size of the extracted patches = kernel size

⊗ Depth of the output feature map = # of filters sur l'Image.

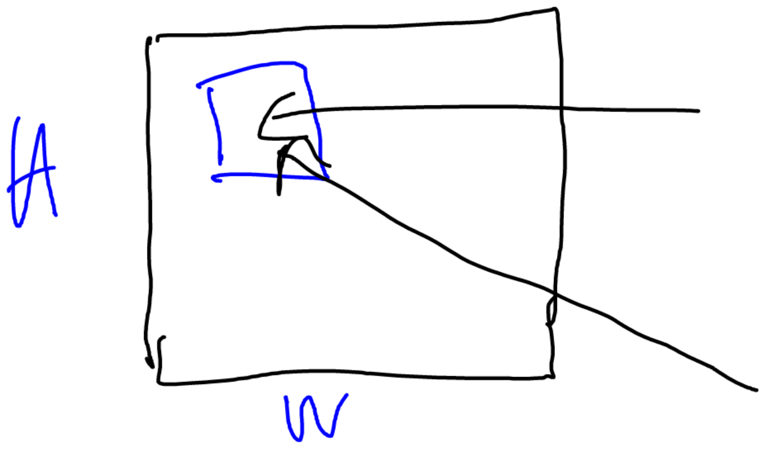
II / Pooling Operation

typical layer of a CNN consists of 3 stages.

→ Convolution : several convolution performed in // on the Input data to produce output feature map (with a number of filter)

→ detector stage : non-linear activation function (ex : ReLU)

→ Pooling Operation : summary statistics of the nearby output.



Average Pooling

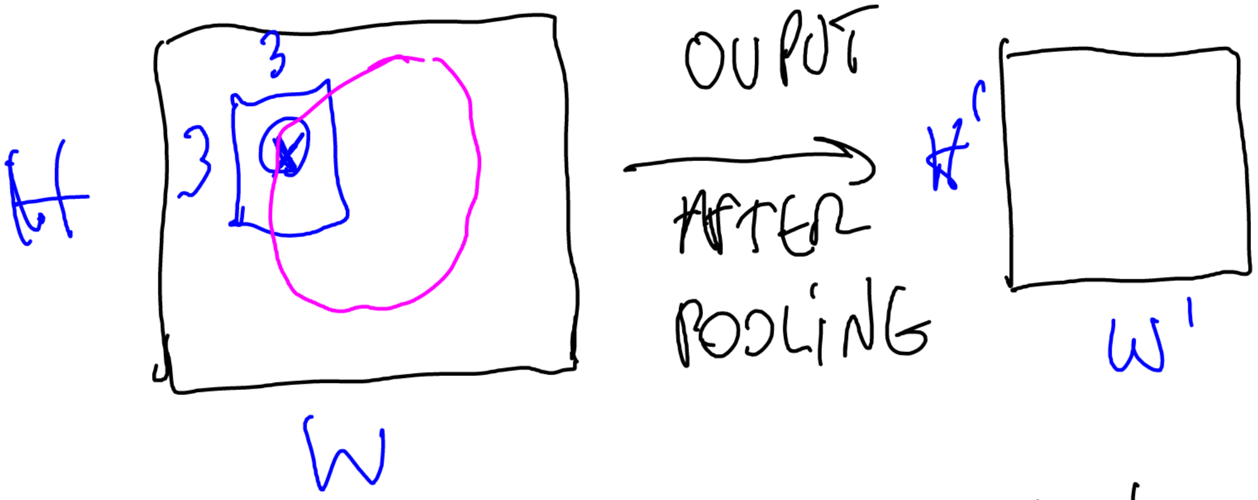
mean of the patch

Max Pooling

maximum over the patch

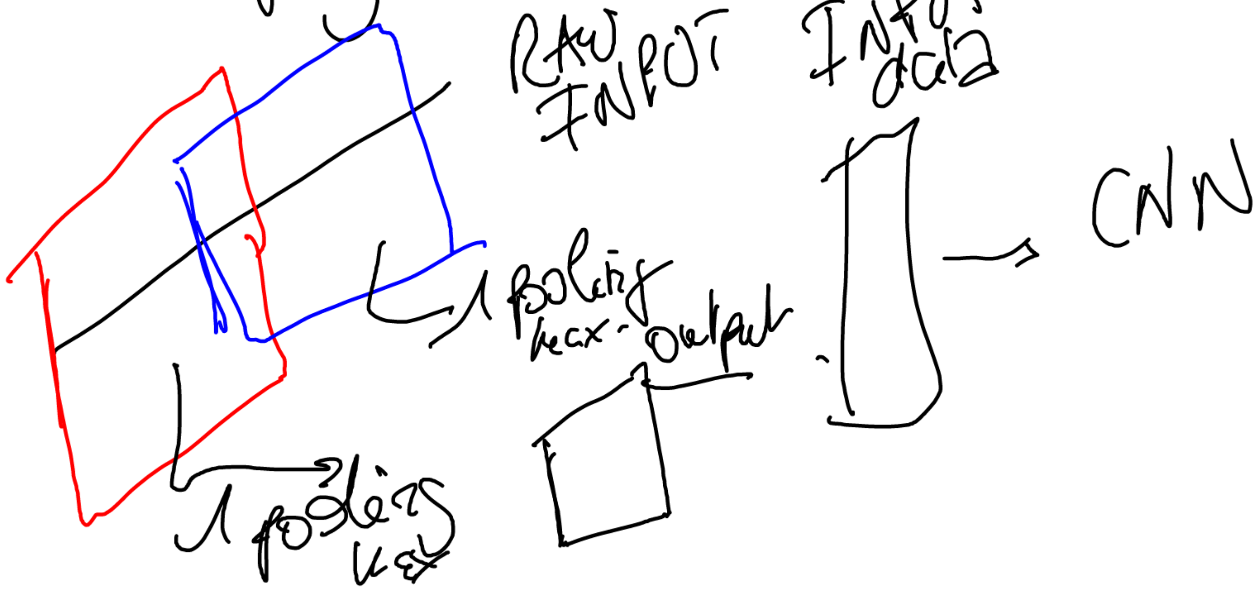
purpose of the pooling operation

↳ "downsampling" sampling:



↳ ~~become~~ invariant to the small translations in the Input data.

↳ CNN can process images of (input) varying size.



segmented Img

bounding box + text

pooling operation as a "transform" operation over feature map

2D feature map

(201, 2048) → 2048

1D-map

↑ Image feature per bounding box

↑ average pooling sur les 201 boxes.

MAX POOLING works better than AVERAGE POOLING.

Consists at extracting windows (= patches) from the input feature map and outputting the max value of each channel ↳ C

↳ 2 hyperparameters

- window size
- stride → used for downsampling the img.