A FASTER INTRODUCTION TO IMAGE PROCESSING

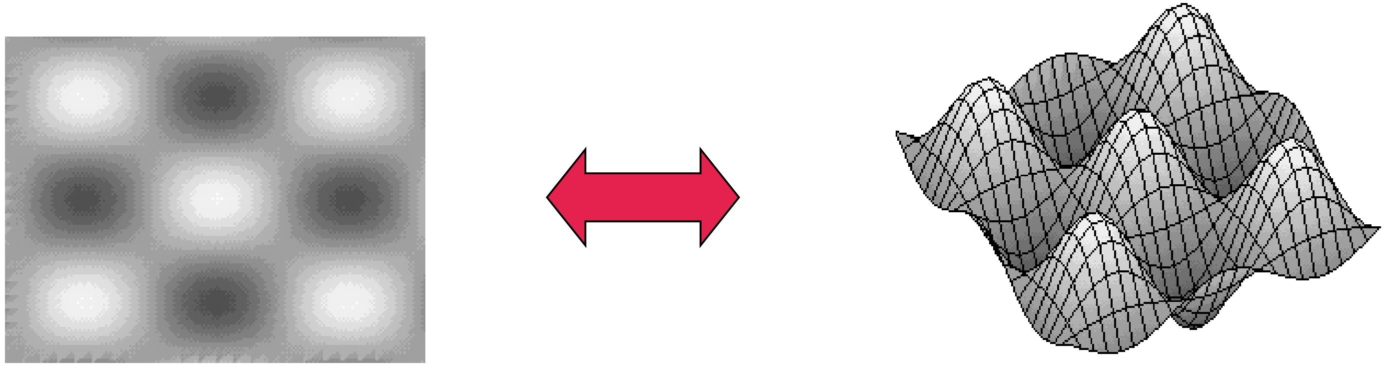
… than early « »

thomas.grenier@insa-lyon.fr

IMAGE PROCESSING

WHAT IS AN *IMAGE*?

- Image definition
  - An image may be defined as a two-dimensional function, $f(x,y)$
    - x and y are spatial (plane) coordinates
    - the amplitude of f at any pair of coordinates (x,y) is called ***intensity*** or ***gray level*** of the image at that point

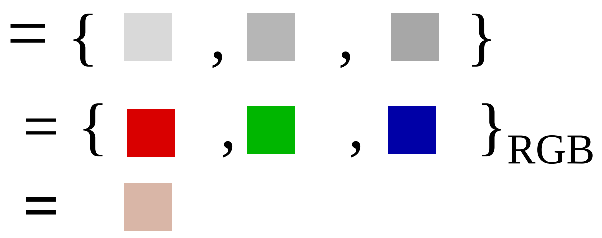


WHAT IS AN *IMAGE*?

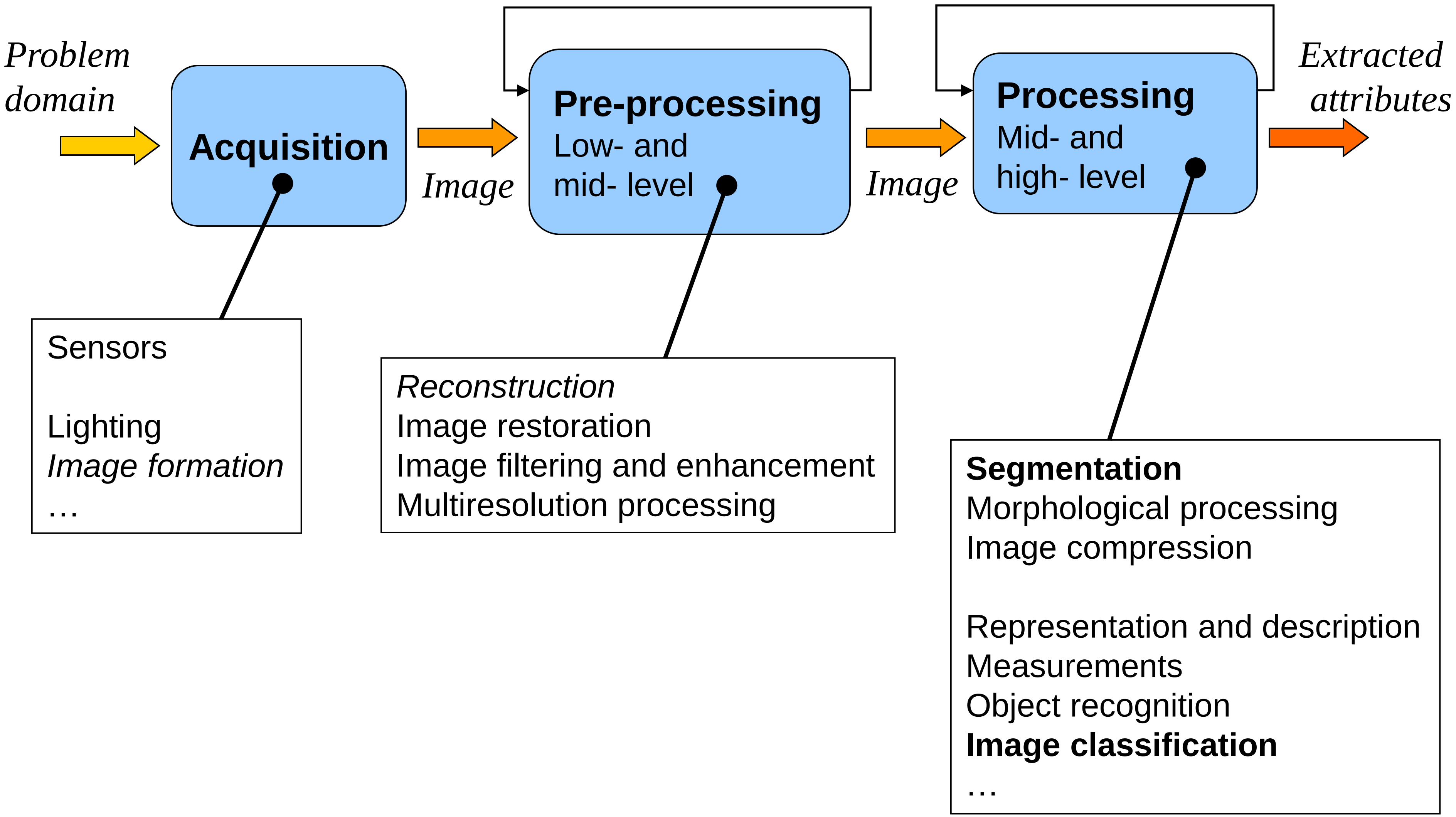
- Digital Image definition
  - The definition of *f* may be extended:
    - as a ***n***-dimensional function,
      - i.e. 3D: $f(x,y,z)$ or image sequence $f(x,y,\mathbf{t})$
    - with amplitudes composed as a vector of data,
      i.e. Color image: 3 components at each point, Complex number

RBG image

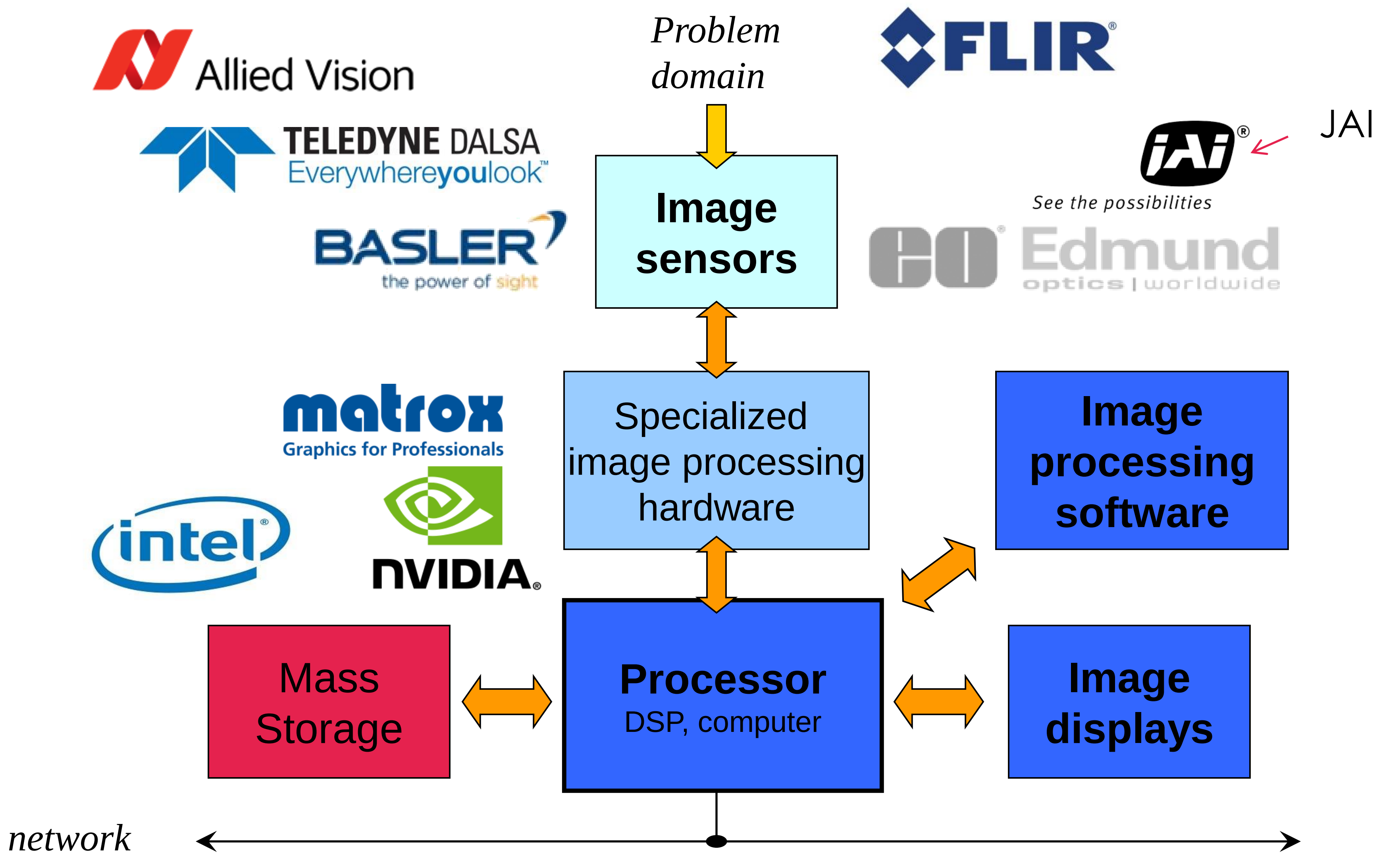
$$f(x_1,y_1) = \{217, 182, 167\}$$



FUNDAMENTAL STEPS IN DIP



COMPONENTS OF AN IMAGE PROCESSING SYSTEM



LINEAR AND NONLINEAR OPERATIONS

- Important classification of an image-processing method
  - Is it a linear or a a nonlinear method ?
- Let H be a general operator

$$H\left[f(x,y)\right]=g(x,y)$$

- H is said to be a ***linear operator*** if

$$H\left[a_i f_i(x,y)+a_j f_j(x,y)\right]=a_i H\left[f_i(x,y)\right]+a_j H\left[f_j(x,y)\right]$$
$$=a_i g_i(x,y)+a_j g_j(x,y)$$

➔ Tools: vector-matrix, PDE, set, *and related operators (+,*, convolution)…*

OPERATIONS ON IMAGES

- Arithmetic Operations $+,-,\times,\div$
  - Application: Corrupted image *g* obtained by adding the noise η to a noiseless image *f*

$$g(x,y) = f(x,y) + \eta(x,y)$$

    - assumptions :
      - at every pair of coordinates (x,y) the noise is uncorrelated
      - the noise has zero average value

    ➔ Noise reduction by summing (averaging) a set of noisy images

$$\sigma_{\bar{g}(x,y)} = \frac{1}{\sqrt{K}} . \sigma_{\eta(x,y)}$$

Application :
Noise Reduction

Original

$\sigma = 3{,}55$

Mean

1 $\sigma = 25{,}45$

2 $\sigma = 18{,}3$

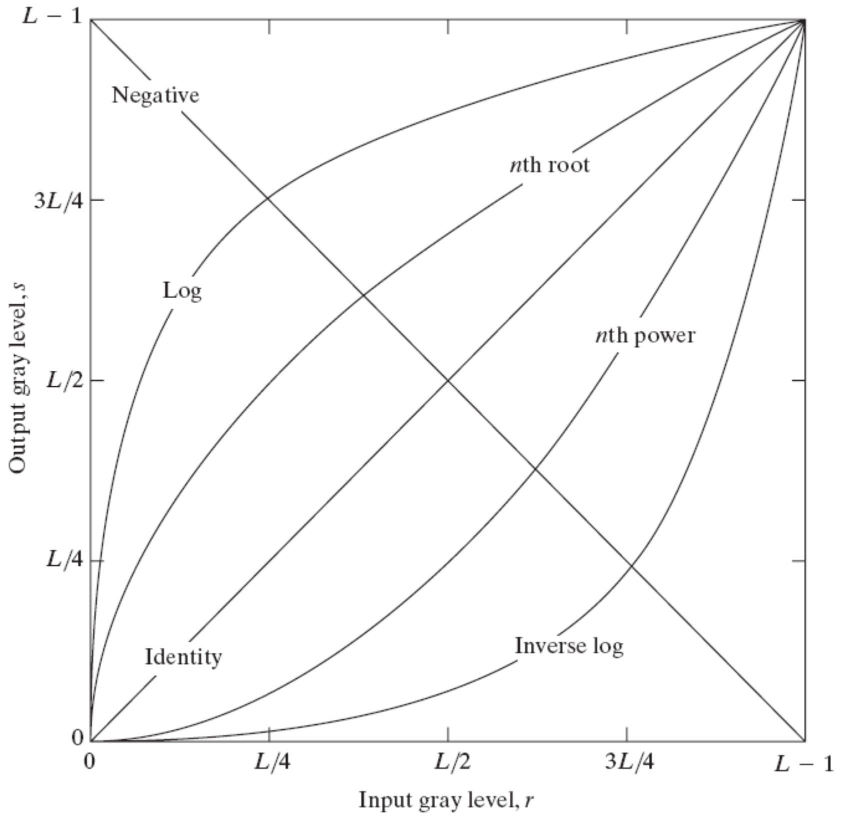
4 $\sigma = 13{,}5$

6 $\sigma = 11{,}6$

OPERATIONS ON IMAGES

- Spatial Operations - Single pixel operations
  - r: original intensity,
  - s: new intensity,
  - T: a transformation function

$$s = T(r)$$

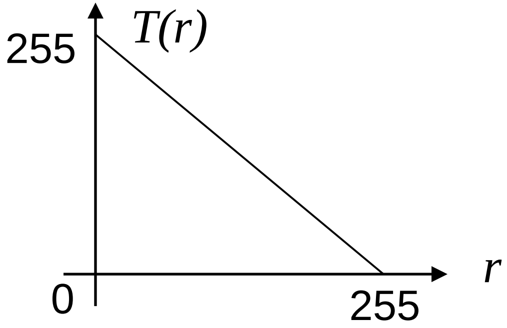


Some basic grey-levels transformation functions used for image enhancement

Range of intensities: *[0,L-1]*

OPERATIONS ON IMAGES

- Spatial Operations - Single pixel operations
  Example

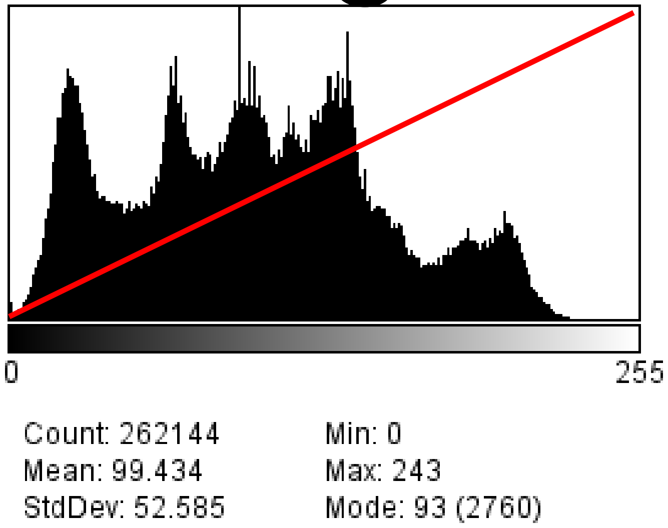


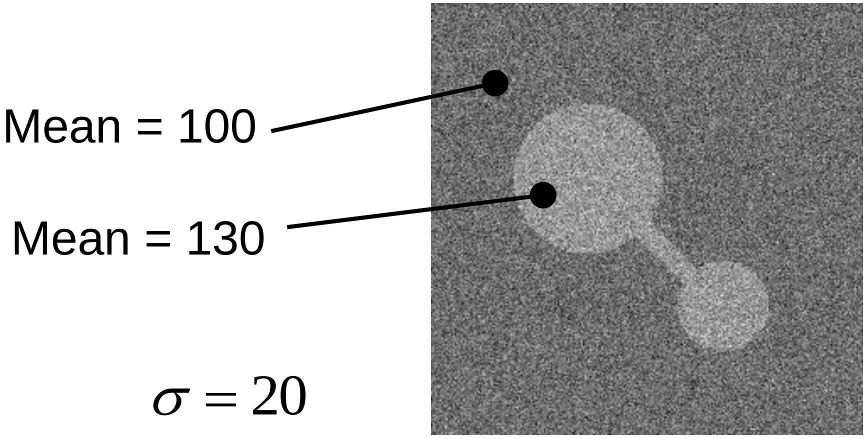
Original

Negative

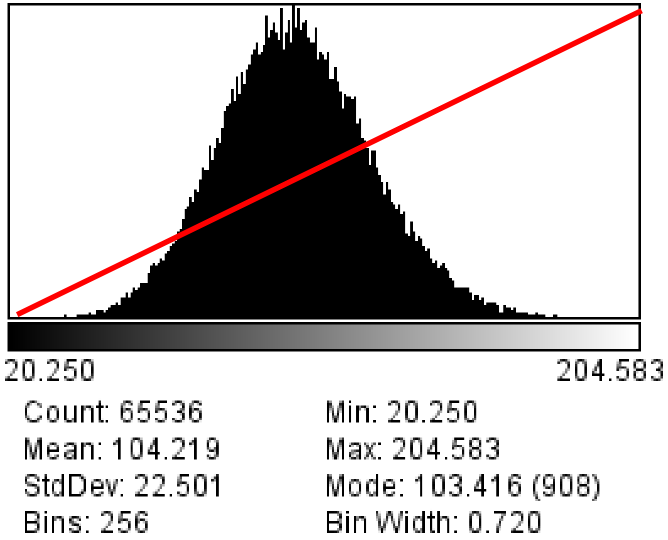
Operations on images : Histogram

- Examples

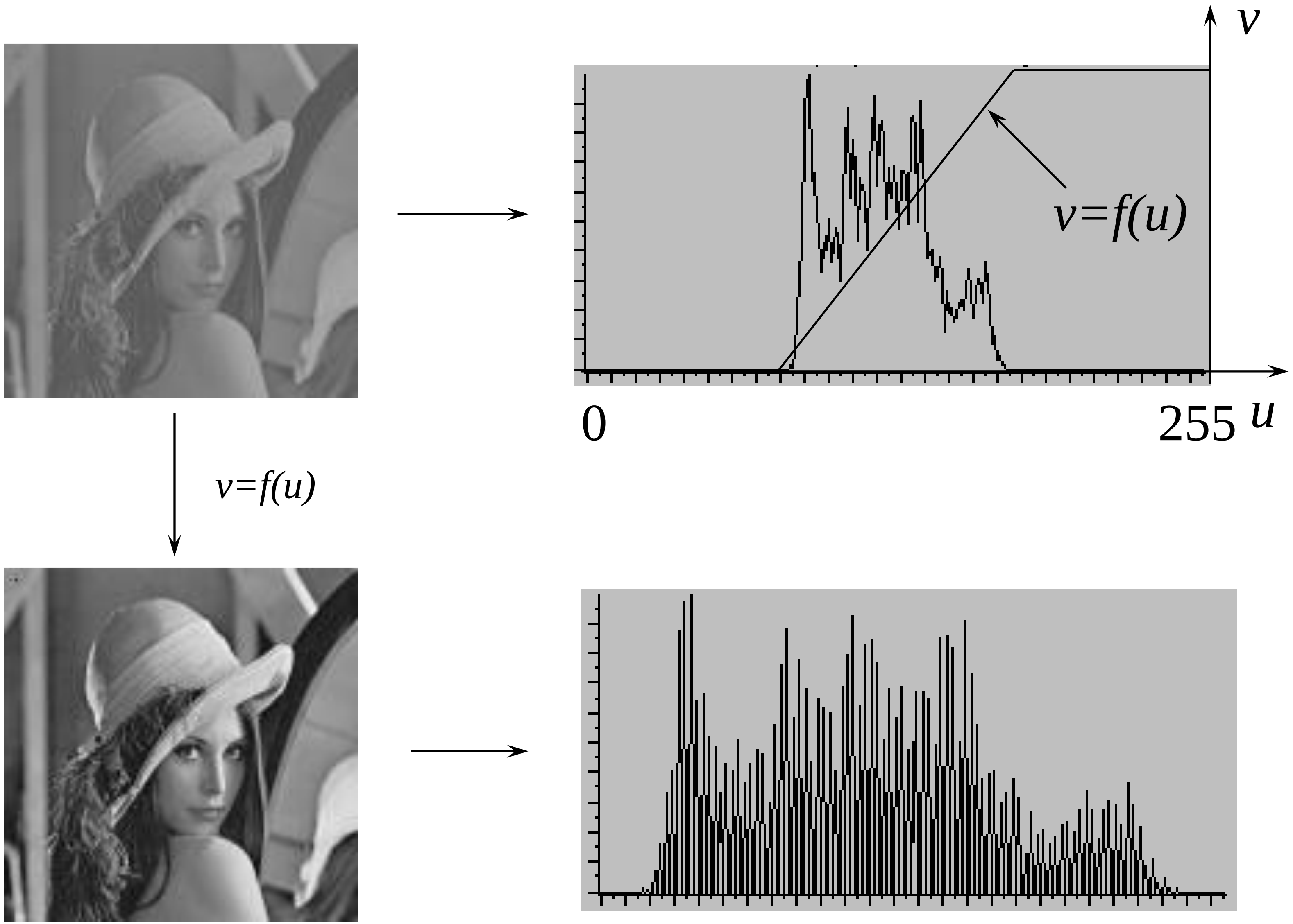




$\sigma = 20$



HISTOGRAM MANIPULATION

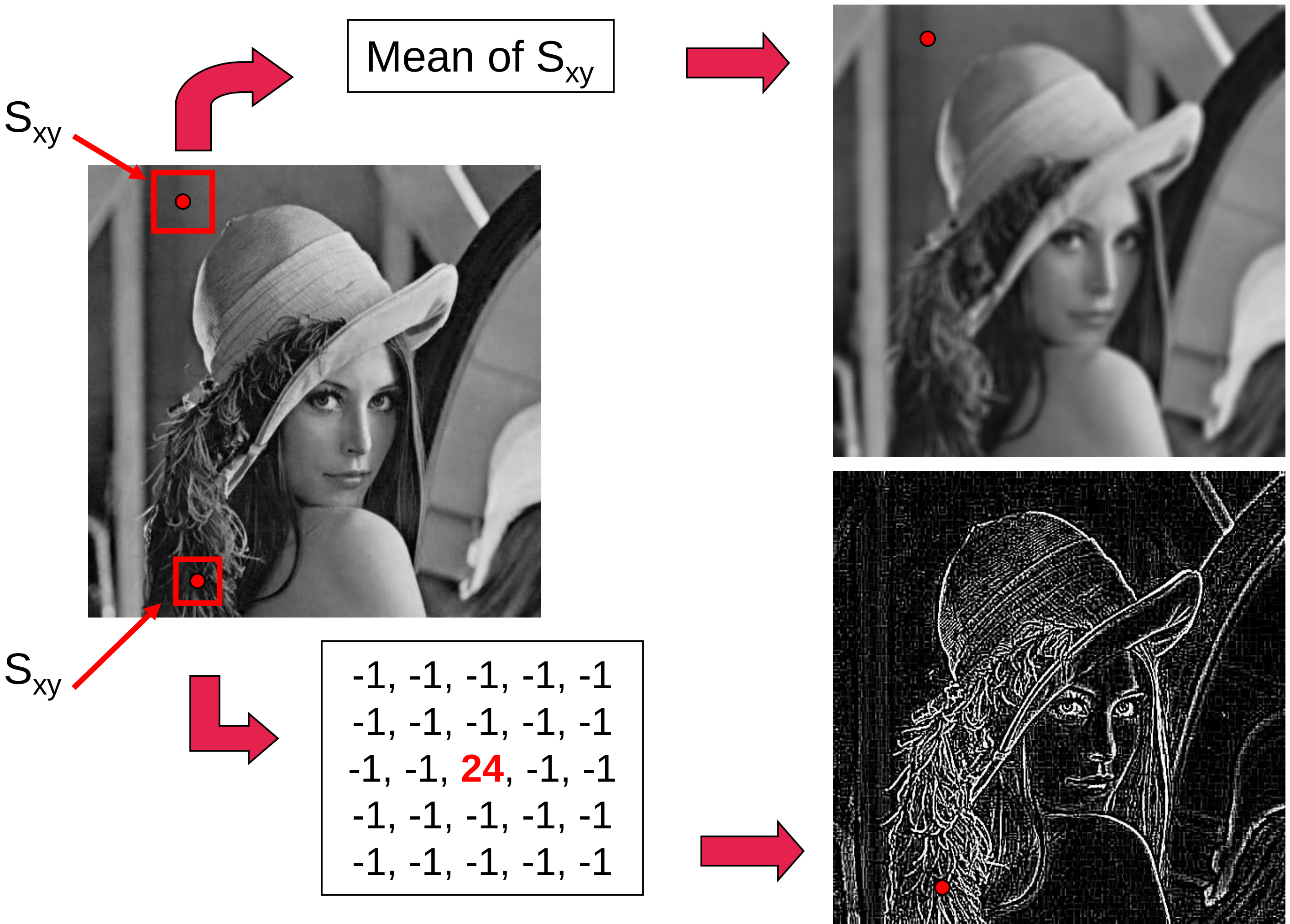


OPERATIONS ON IMAGES

- Spatial Operations - Neighborhood operations
  - S_{xy}: set of coordinates of a neighborhood centered on a point (x,y) in an image *f*.
  - Neighborhood processing generates one corresponding pixel in the output image *g* at the same (x,y) coordinates.
  - The value of that pixel in g is determined by a operation involving the pixels in S_{xy}.

OPERATIONS ON IMAGES

- Spatial Operations - Neighborhood operations



$$g(x,y)=\frac{1}{m.n}\sum_{(r,c)\in S_{xy}} f(r,c)$$

FIR filters…

OPERATIONS ON IMAGES

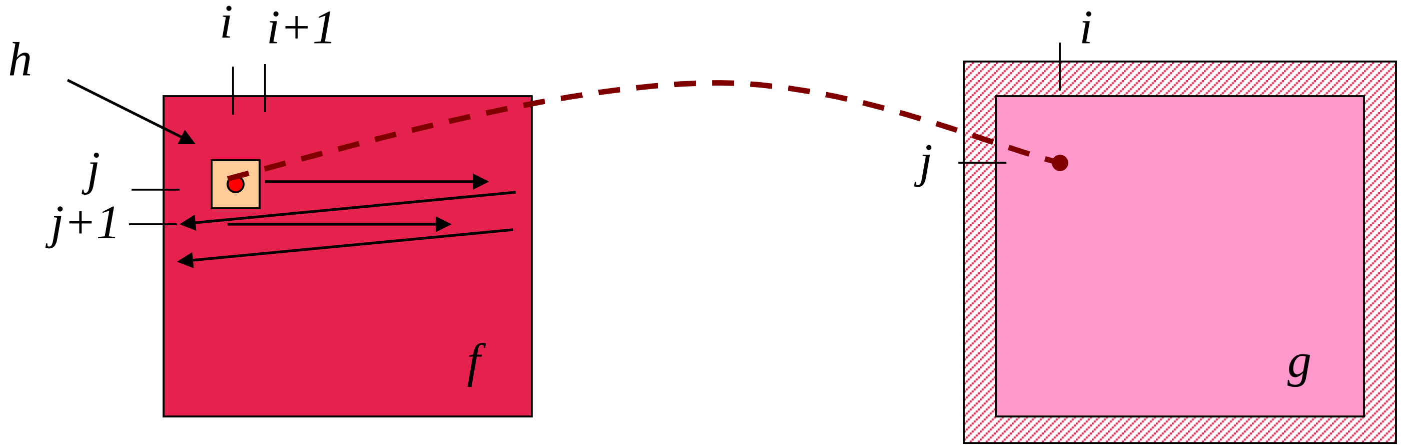
- Convolution $g(x,y) = h(x,y)*f(x,y)$ (two-dimensional convolution)

$$g(i,j) = \sum_{(k,l)\in H}\sum h(k,l)\, f(i-k, j-l)$$

Output image

Convolution mask
Impulse response

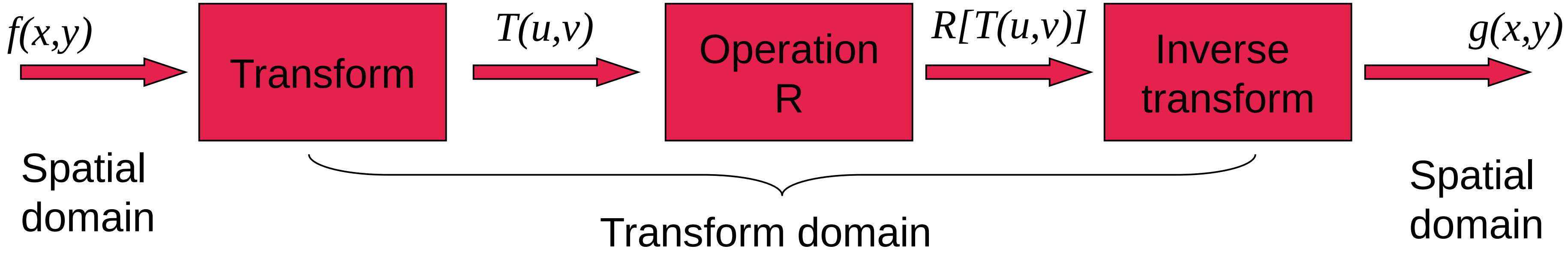
Input image



IMAGES TRANSFORMATION

- Previous methods work in spatial domain
- In some cases, image processing tasks are best formulated in a transform domain.
  - i.e. frequency → Fourier

➔ Many transformations exist



IMAGES TRANSFORMATION

- A particularly important class of 2D linear transforms can be expressed in the general form

$$T(u,v)=\sum_{x=0}^{M-1}\sum_{y=0}^{N-1} f(x,y).r(x,y;u,v)$$

forward transform

Input image

Forward transformation kernel

$$f(x,y)=\sum_{u=0}^{M-1}\sum_{v=0}^{N-1} T(u,v).s(x,y;u,v)$$

Recovered image

Forward transform

inverse transformation kernel

Separable kernel: $r(x,y,u,v)=r_1(x,u).r_2(y,v)$

Symmetric kernel: $r(x,y,u,v)=r_1(x,u).r_1(y,v)$

IMAGES TRANSFORMATION, DFT

- 2-D Discrete Fourier Transform (**DFT**)

Forward kernel $r(x,y,u,v) = e^{-j2\pi(ux/M+vy/N)}$

Inverse kernel $s(x,y,u,v) = \frac{1}{MN} e^{j2\pi(ux/M+vy/N)}$

$$T(u,v) = \sum_{x=0}^{M-1}\sum_{y=0}^{N-1} f(x,y).e^{-2j\pi(ux/M+vy/N)}$$

$$f(x,y) = \frac{1}{MN}\sum_{x=0}^{M-1}\sum_{y=0}^{N-1} T(u,v).e^{2j\pi(ux/M+vy/N)}$$

➔Complex numbers…

➔Modulus and phase (angle)

Sine images → 2 Dirac delta functions

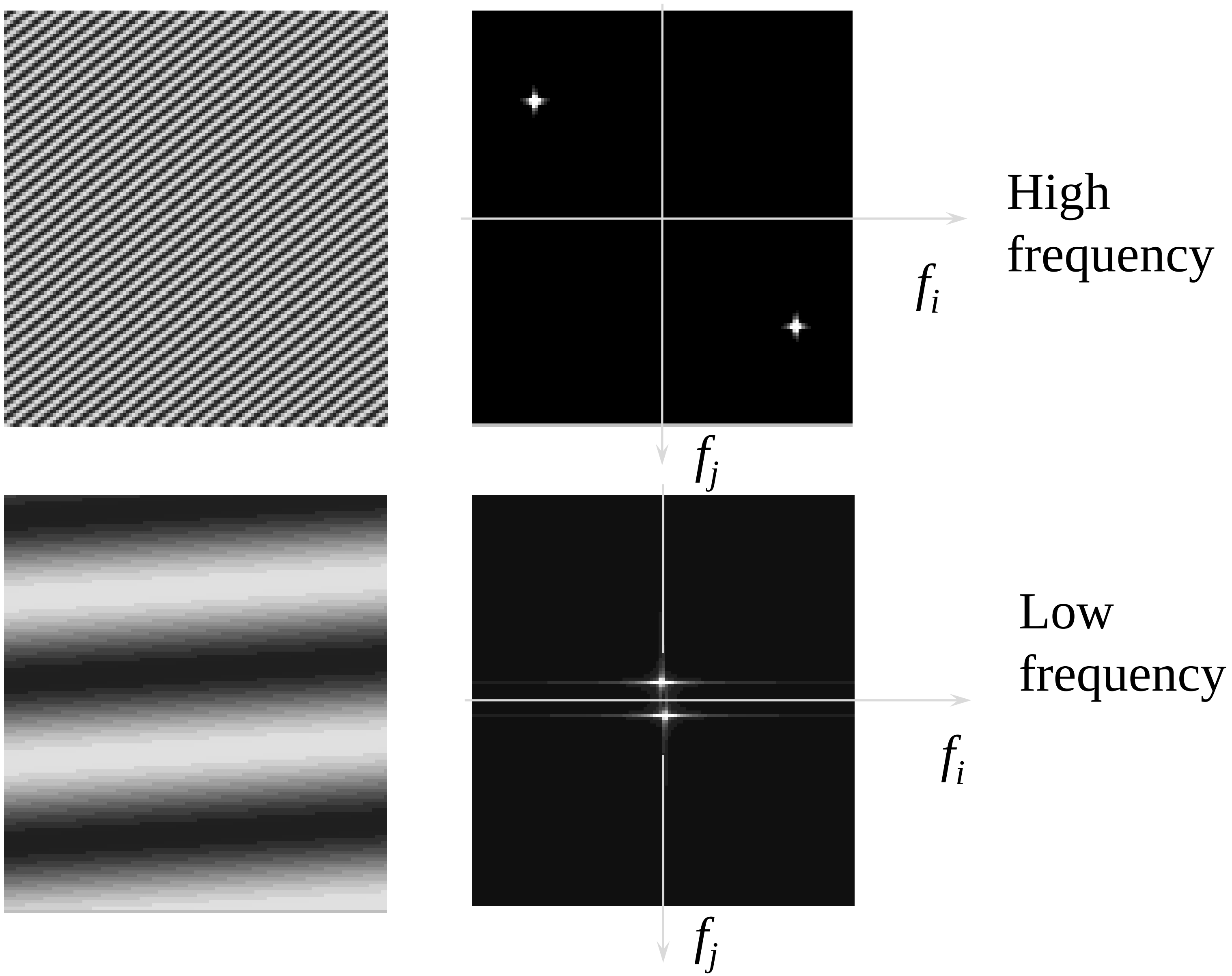
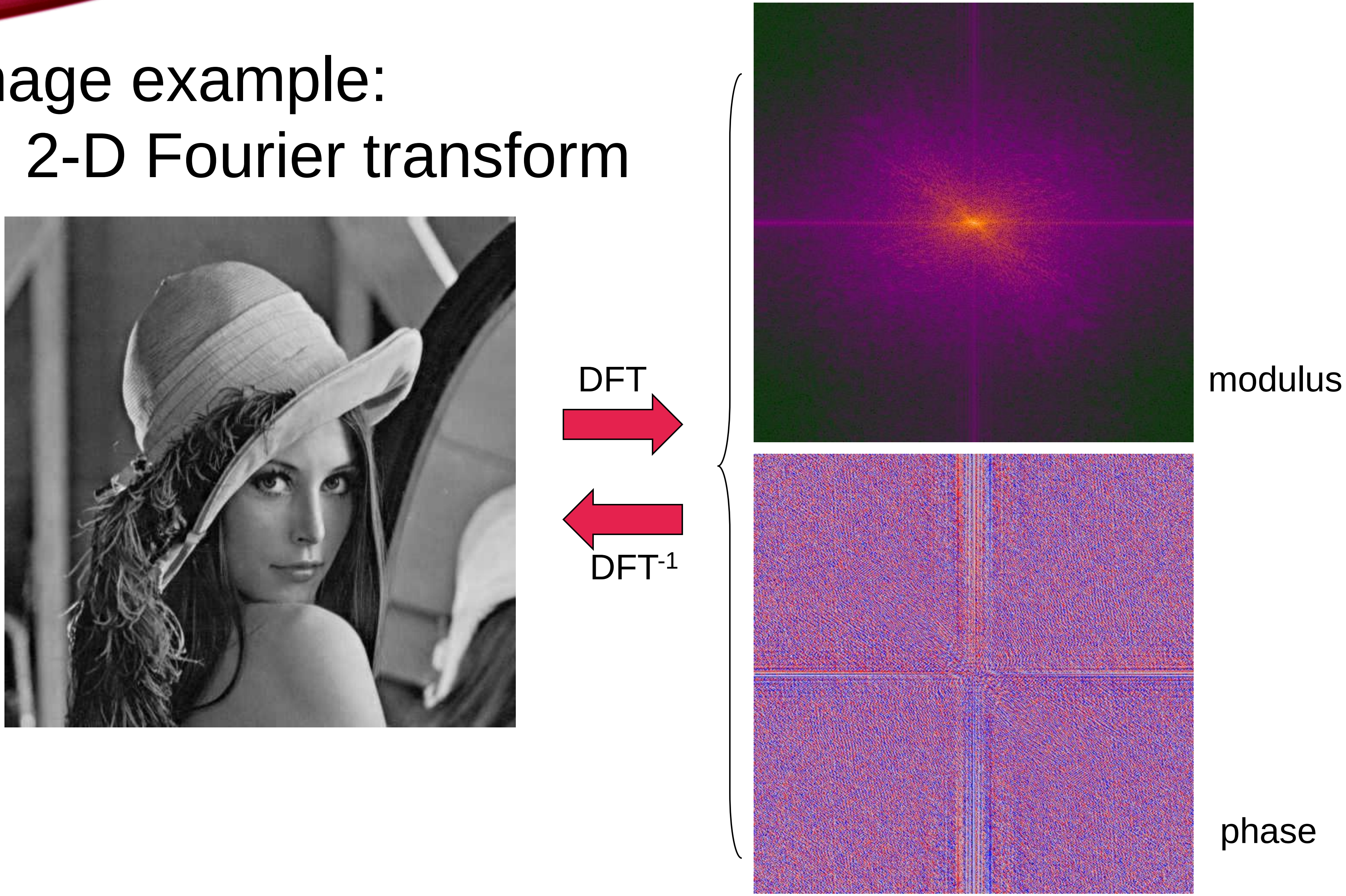
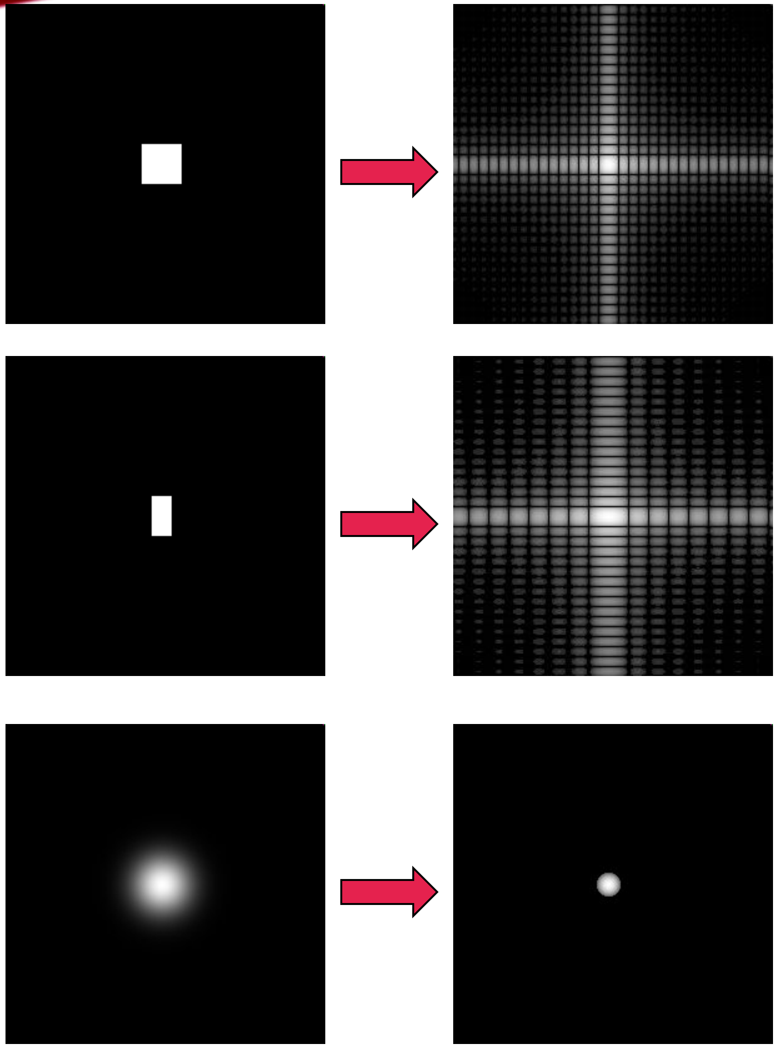


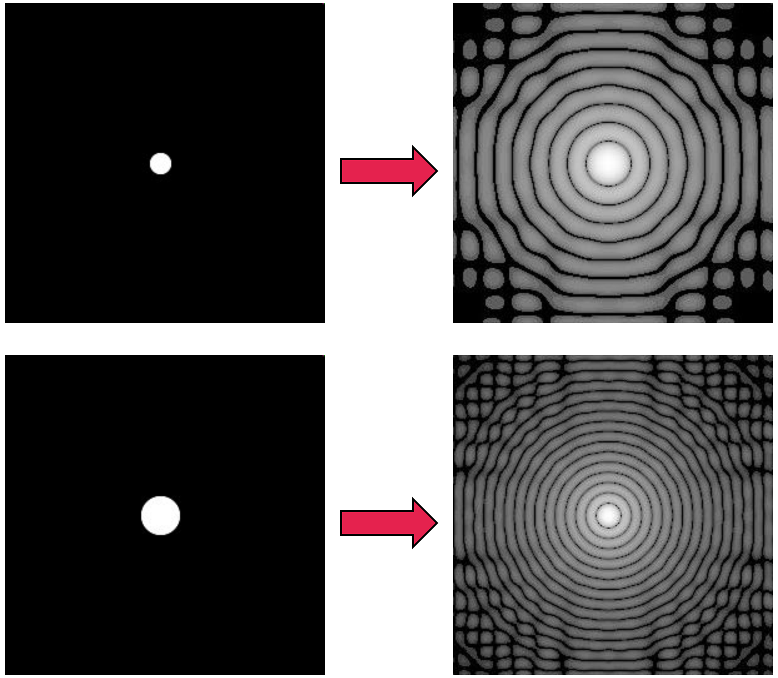
Image example:
2-D Fourier transform

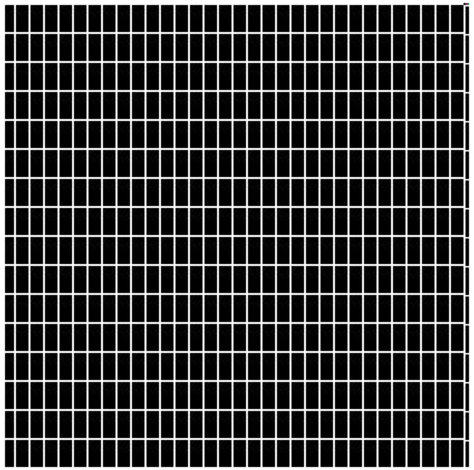


Some basic DFTs

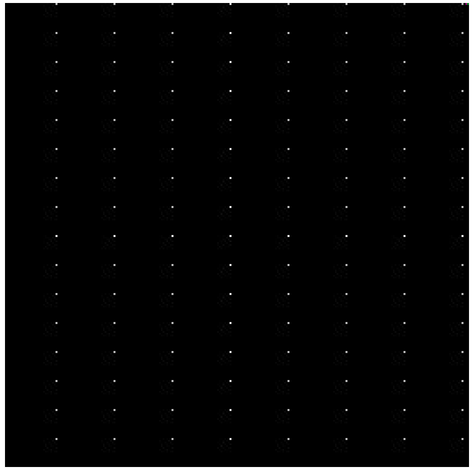


gauss



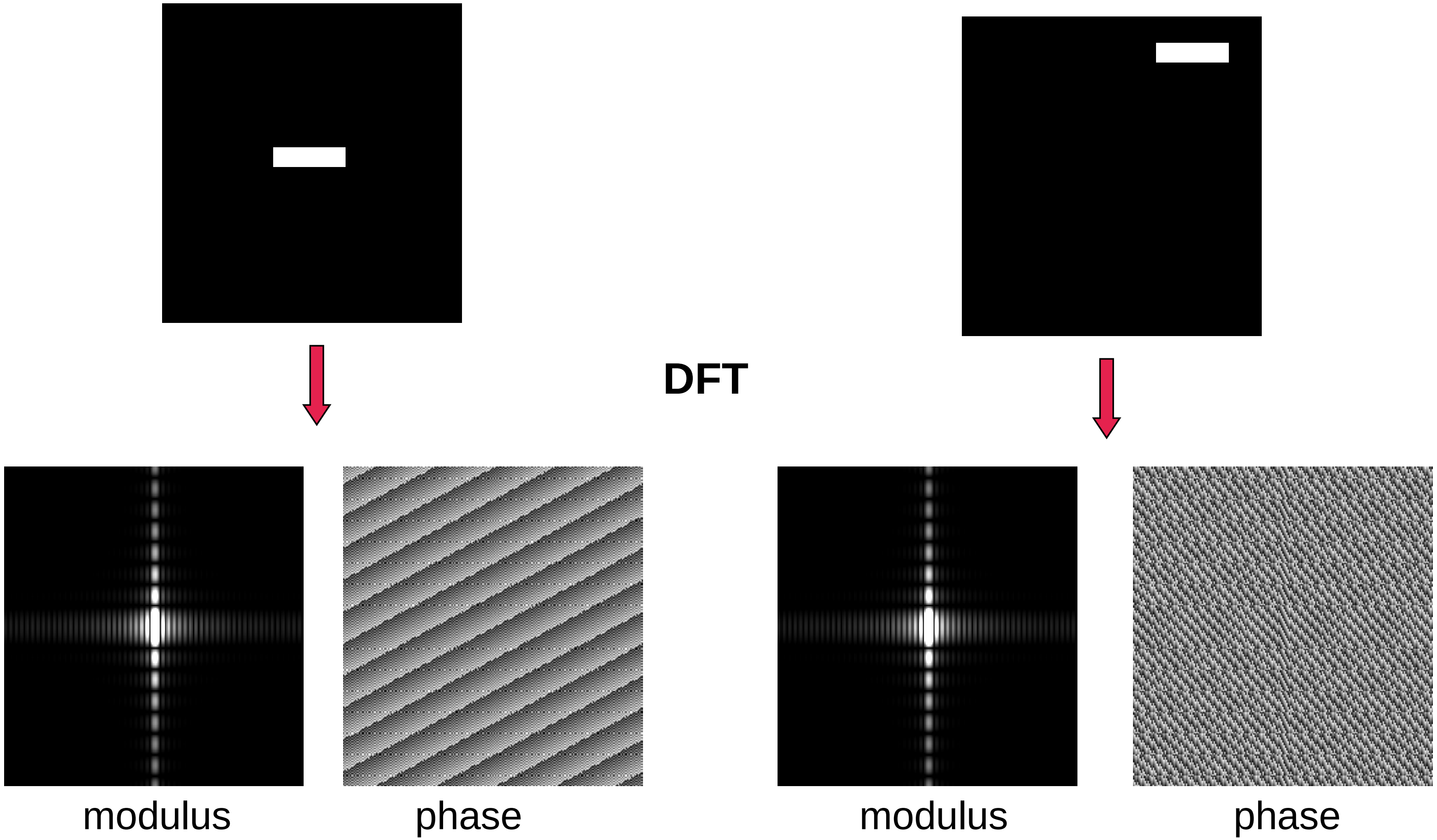


grid



Weighted 2D Dirac comb

Notes on Phase influence



GEOMETRIC AND SPATIAL TRANSFORMATIONS

- Spatial transformations
  - Example
    - Shrink image to half its size

$$(x', y') = T\{(x, y)\} = (x/2, y/2)$$

  - **Affine transform**:

$$\mathbf{x}' = \mathbf{A}.\mathbf{x} + \mathbf{t}$$

$$[x', y', 1] = [x, y, 1].\begin{bmatrix} t_{11} & t_{12} & 0 \\ t_{21} & t_{22} & 0 \\ t_{31} & t_{32} & 1 \end{bmatrix}$$

  - Homographic transform
    - Two cameras looking points on a plane : x,y,…z !
  - Higher order

$$[x', y', 1] = [x, y, x^2, y^2, xy, \ldots, 1].\mathbf{T}$$

Affine transform

identity

$$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \Rightarrow \begin{cases} x' = x \\ y' = y \end{cases}$$

scaling

$$\begin{bmatrix} c_x & 0 & 0 \\ 0 & c_y & 0 \\ 0 & 0 & 1 \end{bmatrix} \Rightarrow \begin{cases} x' = c_x.x \\ y' = c_y.y \end{cases}$$

translation

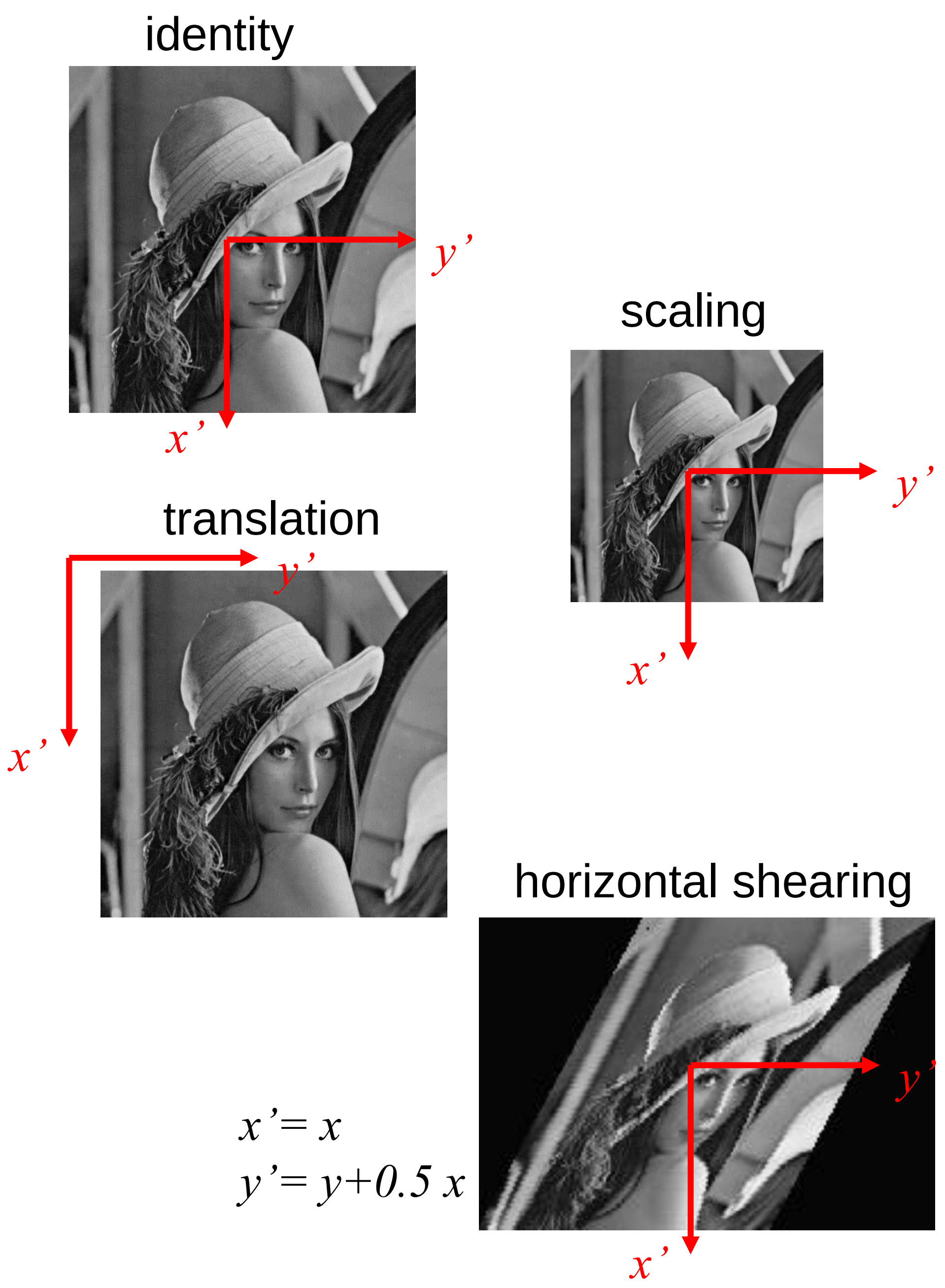
$$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ t_x & t_y & 1 \end{bmatrix} \Rightarrow \begin{cases} x' = x + t_x \\ y' = y + t_y \end{cases}$$

Shear (vertical)

$$\begin{bmatrix} 1 & 0 & 0 \\ s_x & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \Rightarrow \begin{cases} x' = x + s_x.y \\ y' = y \end{cases}$$

Shear (horizontal)

$$\begin{bmatrix} 1 & s_y & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \Rightarrow \begin{cases} x' = x \\ y' = y + s_y.x \end{cases}$$



- Affine transform: rotation

Rotation $$T = \begin{bmatrix} \cos\theta & \sin\theta & 0 \\ -\sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

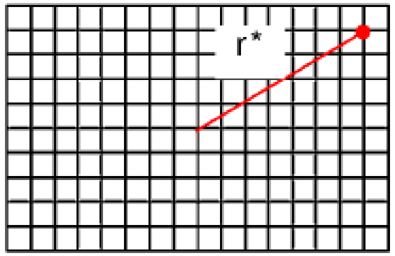
35° degrees rotation (from image center)

Remark : interpolation is necessary to avoid aliasing

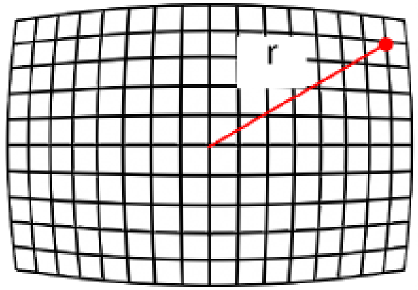
- Higher order transforms

Applications :

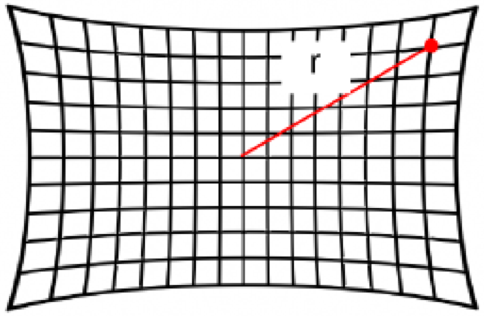
Lens distortion correction, perspective



Orthoscopic projection



Barrel distortion



Pincushion distorsion

APPLICATION : IMAGE FILTRERING

NOISE CLEANING

- Many types of noise…

Gaussian Noise
(sd 25)

Salt and pepper noise

LINEAR FILTERING

- Mean filter

$$g(i,j)=\sum_{(k,l)\in W}\sum h(k,l)\,f(i-k,j-l)$$

W: 25 neighbors

H=(1/25).**I**

Result on Gaussian noise

Result on Salt & Pepper noise

NON LINEAR FILTERING

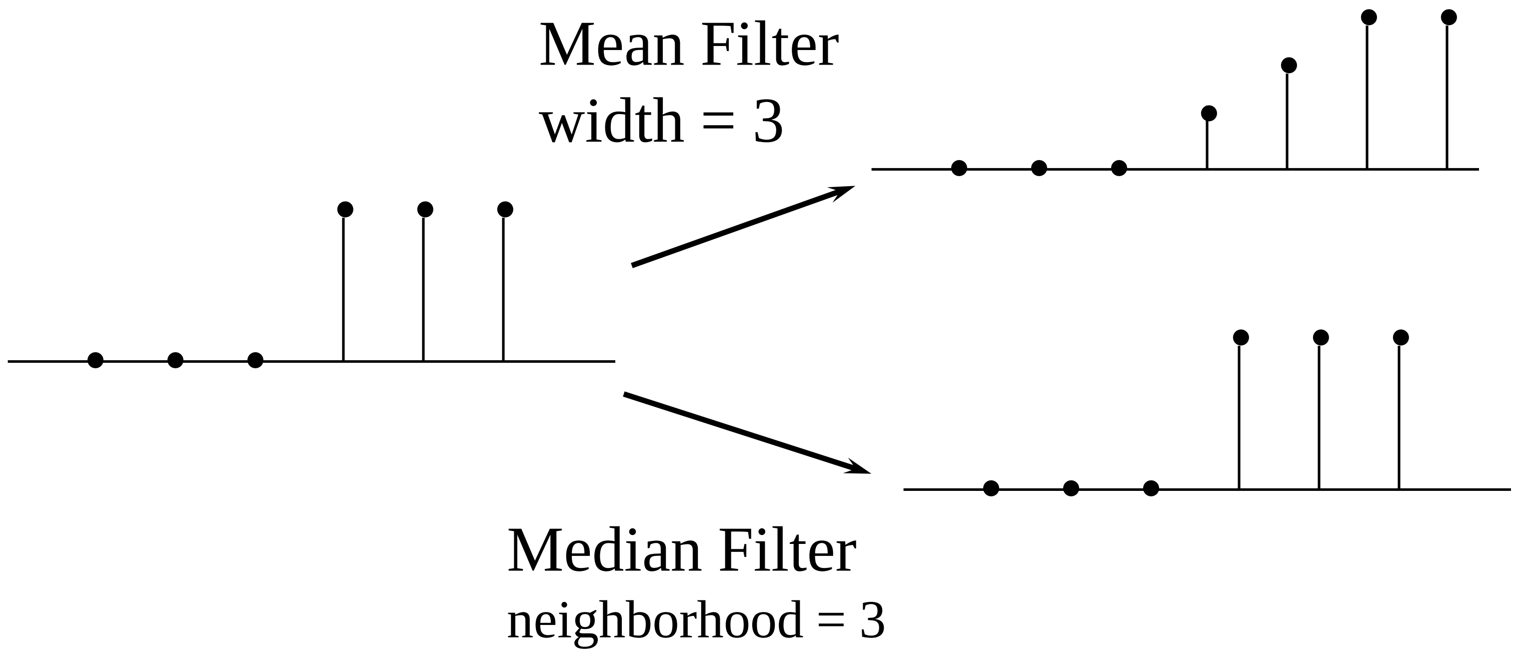
- Median filter

Result on Gaussian noise W=25

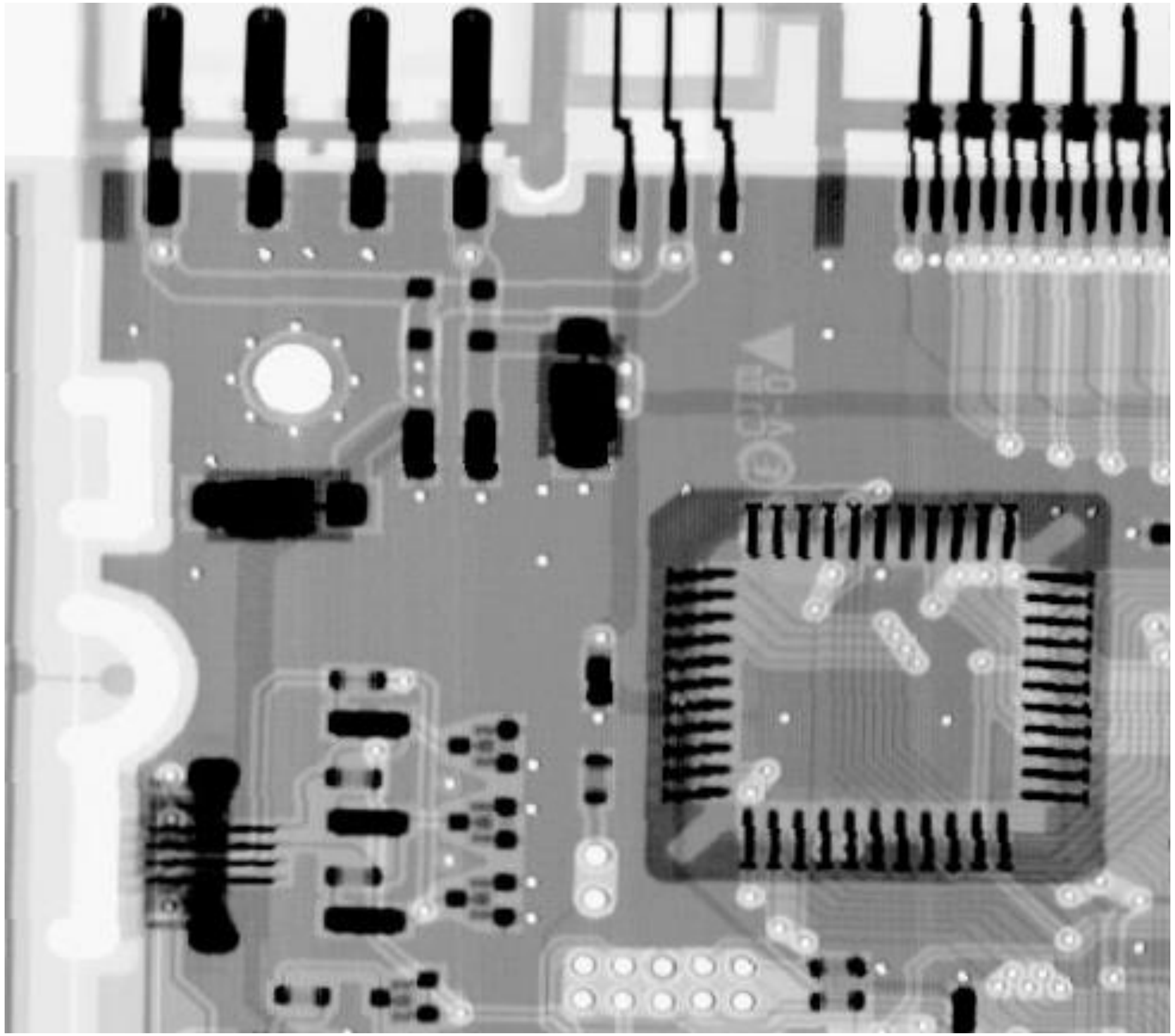
W=9 Result on salt & pepper noise

NON LINEAR FILTERING

- Median filter
  - Advantage of median filtering over linear filtering: edges are preserved

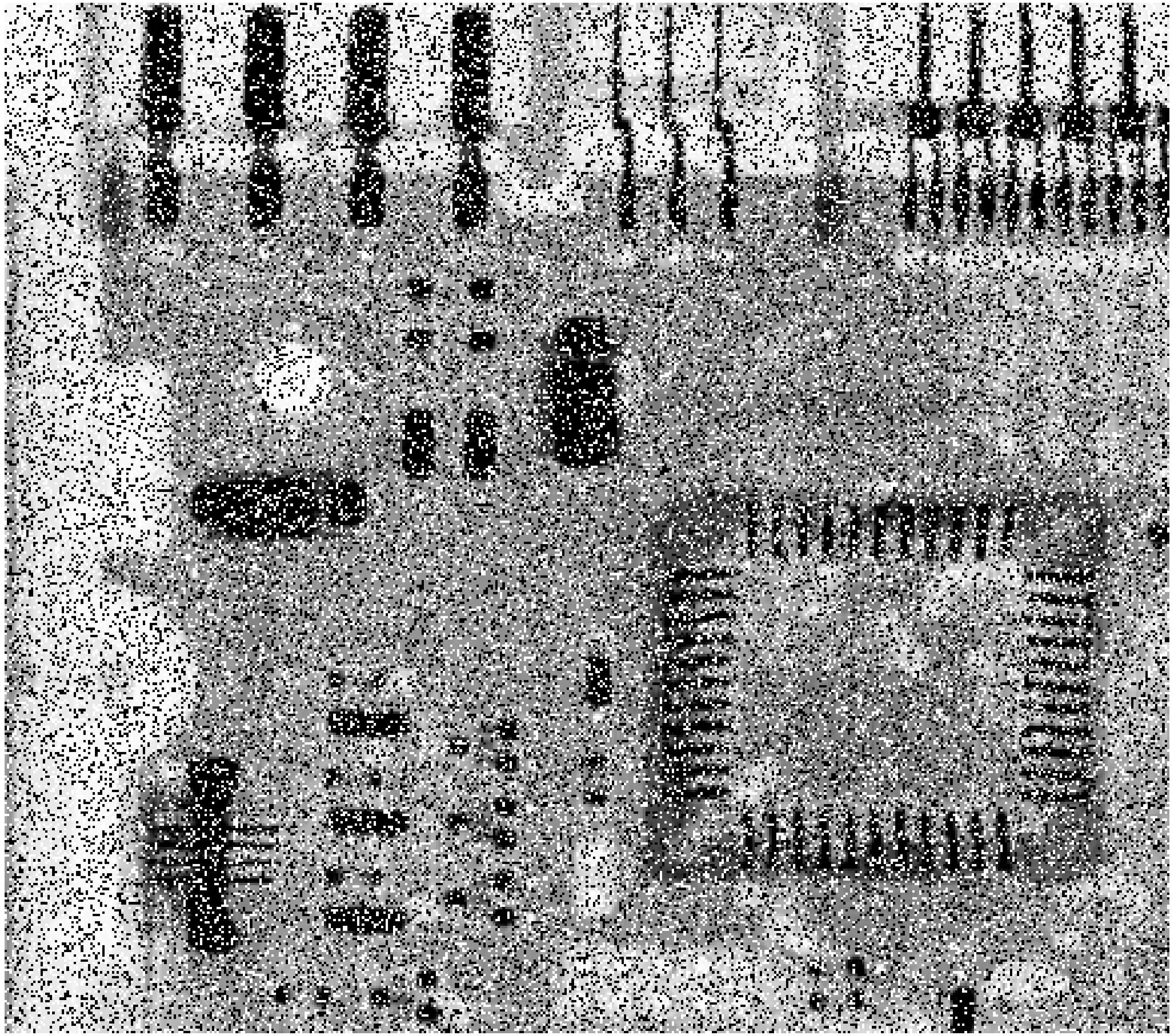


- Adaptive Median filter : Reference

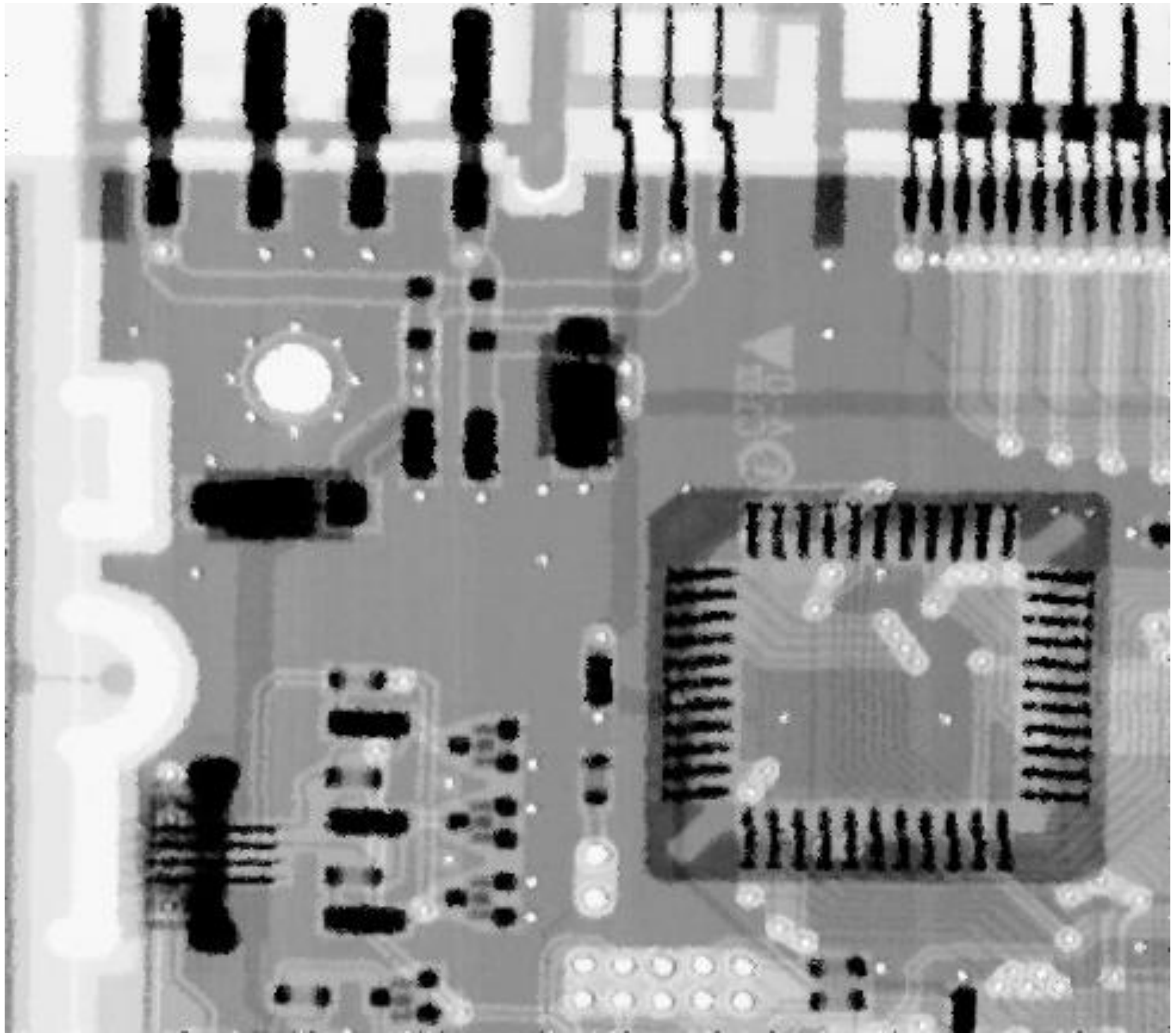


- Adaptive Median filter : Input

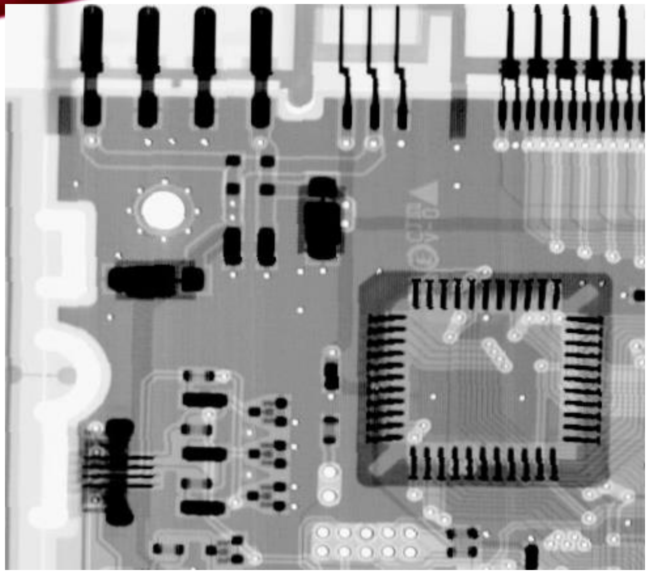
Noisy image (S&P, 0.35)



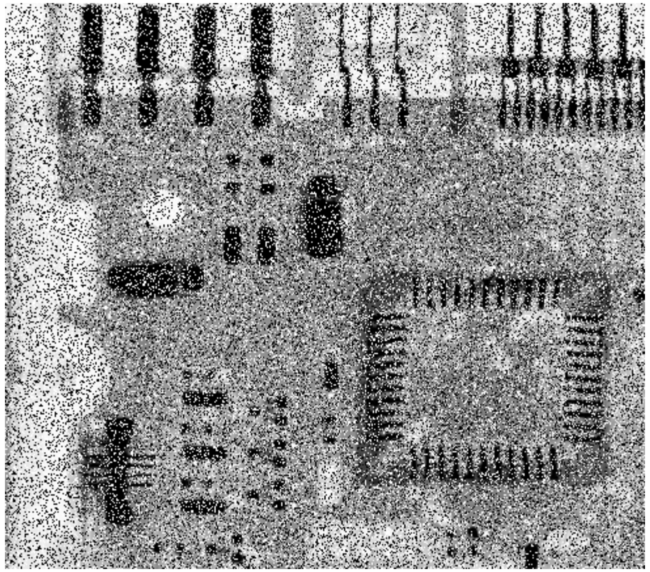
- **Adaptive** Median filter



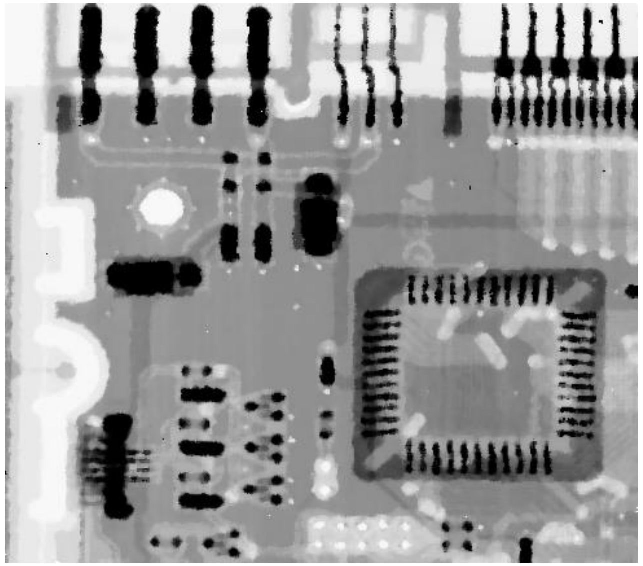
- Result of Adaptive Median filter



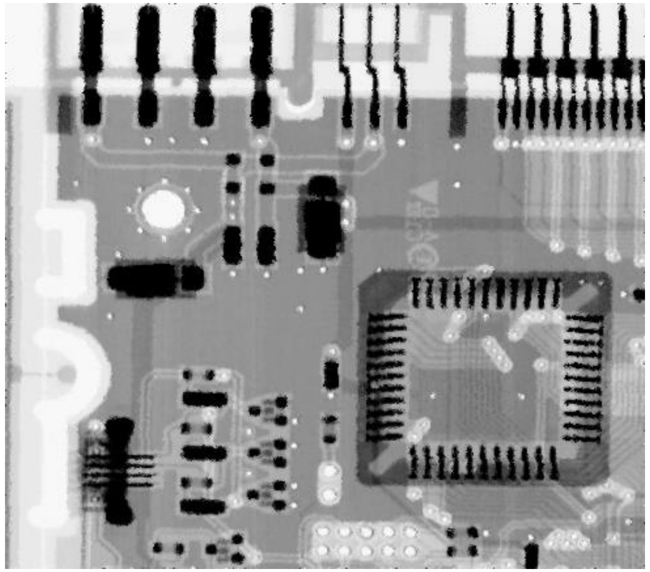
Reference



Noisy image (S&P, 0.35)



Median Filter (5x5)



Adaptive Median (S_{max} 9x9)

RESTORATION: PERIODIC NOISE REDUCTION

$$g(x,y) = f(x,y) + \eta(x,y)$$

- By frequency domain filtering
  - Bandreject filter
  - Notch filter → optimum notch

Build H_{NP} (Notch Pass) by placing a notch pass filter at the location of each spike.
Interference noise pattern is:

$$N(u,v) = \mathrm{H_{NP}}(\mathrm{u,v}).\mathrm{G}(\mathrm{u,v})$$

then $$\eta(\mathrm{x,y}) = \mathrm{FT}^{-1}\left[N(u,v)\right]$$

thus $$\hat{f}(x,y) = g(x,y) - w(x,y).\eta(\mathrm{x,y})$$

Estimate of f(x,y)

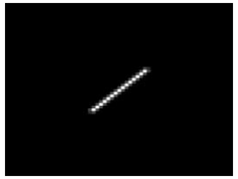
Weighted function (minimizes the effect of components not present in the estimate of η)

→ **How to select w(x,y) ?**

RESTORATION

Reference Image L

$\otimes$



Shift filter

f

$+$

Noise

n

$=$

Corrupted image I

$$I = f \otimes L + n$$

$$(I, f) \xrightarrow{?} L$$

HOW TO 'DECONVOLUTION'?

$$\hat{F}(u,v) = \frac{G(u,v)}{H(u,v)}$$

- Inverse filtering
  - Without noise
  - **With noise**
    - We have to known N!
    - What happen for small values of H(u,v) ?

$$\hat{F}(u,v) = F(u,v) + \frac{N(u,v)}{H(u,v)}$$

➔Solutions

➔Minimum Mean Square Error (Wiener) Filtering

➔Constrained Least Squares Filtering

➔**When H is unknown** : blind deconvolution

Many others approaches exist, ie the 'blind' ones

WIENER (MINIMUM MSE) AND CLS

- Wiener (Fourier), Minimum MSE

$$argmin_{\hat{L}} = E\{(L-\hat{L})^2\} \longrightarrow L(u,v) = \frac{1}{f(u,v)} \frac{|f(u,v)|^2}{|f(u,v)|^2 + \frac{S_\eta(u,v)}{S_L(u,v)}} I(u,v)$$

- Constrained Least Square (Fourier)

$$argmin_{\hat{L}} = ||I - f \otimes \hat{L}||_2^2 + \gamma || \triangle \hat{L}||_2^2 \rightarrow L(u,v) = \frac{1}{f(u,v)} \frac{|f(u,v)|^2}{|f(u,v)|^2 + \gamma |P(u,v)|^2} I(u,v)$$

$f(u,v)$ = degradation function in the Fourier space
$|f(u,v)|^2 = f^*(u,v).f(u,v)$, with $f^*(u,v)$ the complex conjuagte of f
$S_n(u,v) = |N(u,v)|^2$ = noise power spectrum
$S_L(u,v) = |L(u,v)|^2$ = power spectrum of undegraded image
ɣ = parameter to tune

$$p(x,y) = \begin{vmatrix} 0 & -1 & 0 \\ -1 & 4 & -1 \\ 0 & -1 & 0 \end{vmatrix}$$

APPLICATION : IMAGE SEGMENTATION

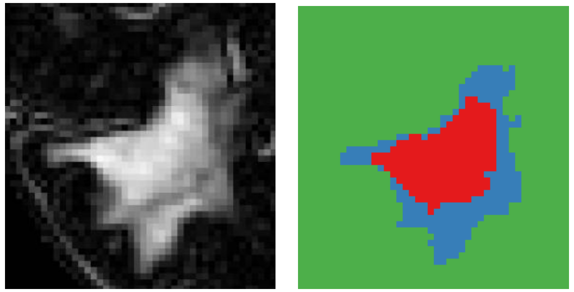
SEGMENTATION

- Visualization
- Counting
- Identification
- Measurements (shape, volumes …)
- Tracking

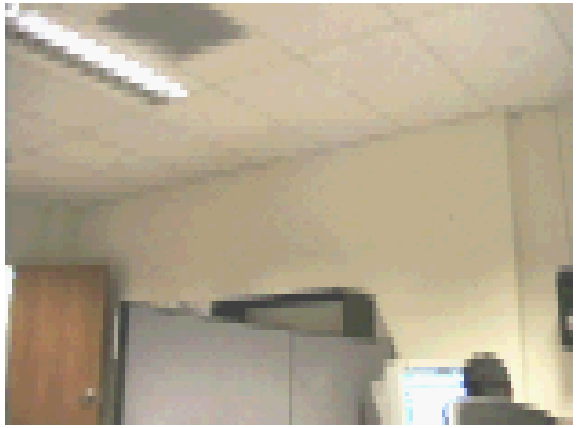
HDR mode

Top-left: iPhone photo #1. Bottom-left: iPhone photo #2. Right: **TrueHDR** result.

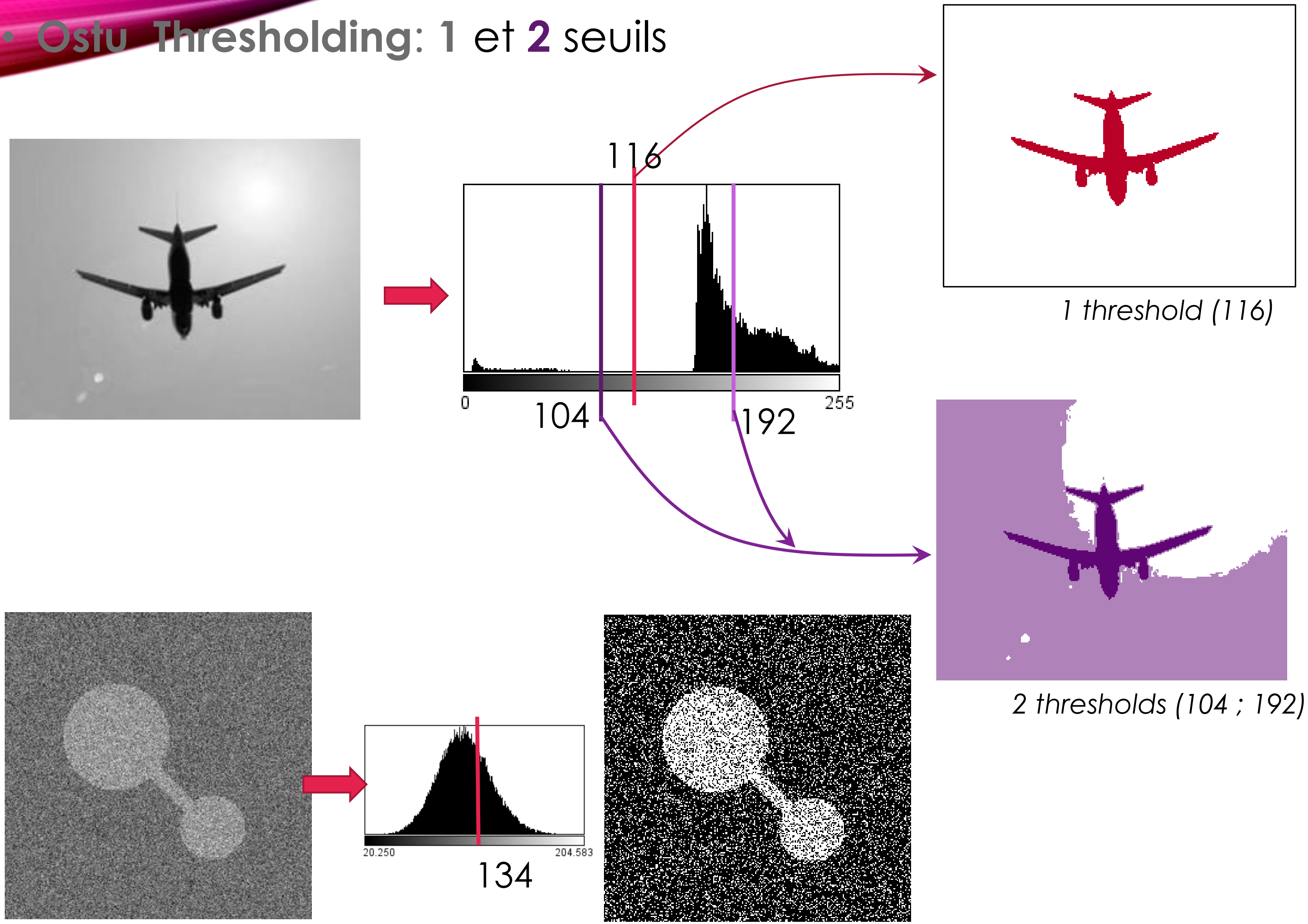
IRM SEP



[Comaniciu 2003]



- **Ostu Thresholding**: **1** et **2** seuils



1 threshold (116)

2 thresholds (104 ; 192)

REGION GROWING

- Algorithm
  - Determine **C *neigbhoors*** of region R
  - Add neigboors ***similar*** to region R
- Iterate while R is growing

Spatial proximity

« Ressemblance »

Exemple

Mean of intensities of R $\boldsymbol{\mu} = E\left[\mathbf{I}(R)\right]$

Ressemblance … $d = \left\|\boldsymbol{\mu} - \mathbf{I}(\mathbf{x})\right\|$

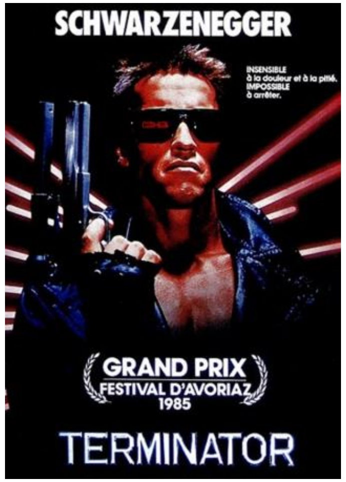
Critère : $\mathbf{x}$ is ***similar*** to R if : $d < seuil$

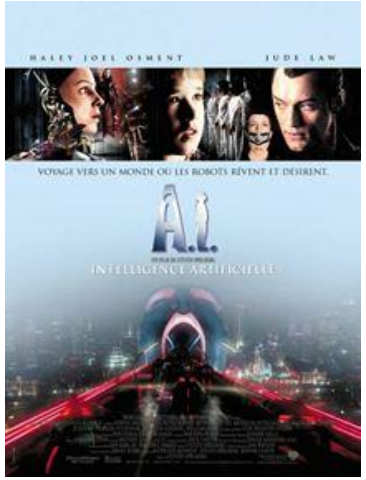
MACHINE LEARNING

As fast as image introduction

ARTIFICIAL INTELLIGENCE AND SOCIETY

Until few years ago only science fiction…





…

DETROIT

BECOME HUMAN

Today : leaders on personnal data use it extensively

amazon Microsoft Baidu 百度 Alibaba.com Tencent 腾讯 Google IBM SAMSUNG

… BUT ALSO

- Why is it so exciting for value creation?
  - Proposes solutions to (all) problems that have been partially solved or not
  - Obtain better performances than conventional approaches (quality and/or use of resources) → 2012
  - Intrinsically allows to have a usable solution after the 'research' and development phase

https://pjreddie.com/darknet/

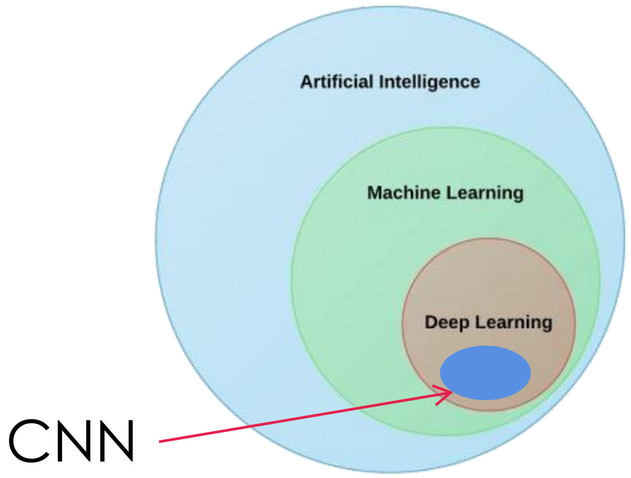
MACHINE LEARNING APPROACHES

- Supervised approaches (*non ANN)*
  - Bayes, kNN, Support Vector Machine, Random Forest, LDA, Regression Logistique, …
- Un-supervised approaches (*non ANN)*
  - *K-means, GMM, Hierarchical approach, DBScan, Laio-Rodriguez, …*

➔ ANN (Artificial Neural Network)

→ « DNN » (**Deep Neural Network**)

MLP/ **CNN** / RNN / BM



NOTATIONS : DATA

- Instance of all features values : **x**
- The corresponding predicted label : *û*
- Examples **x** : *û*
  - Bank statement : solvency
  - Flower picture : identification (rose, dendelion, iris…)
  - Address, surface, year , … : apartment price

$$\mathbf{x} \in \mathcal{X}$$
$$u \in \mathcal{U}$$

➔ a Dataset consists of *N* instances (**x**, *u*) → **X**, **u**

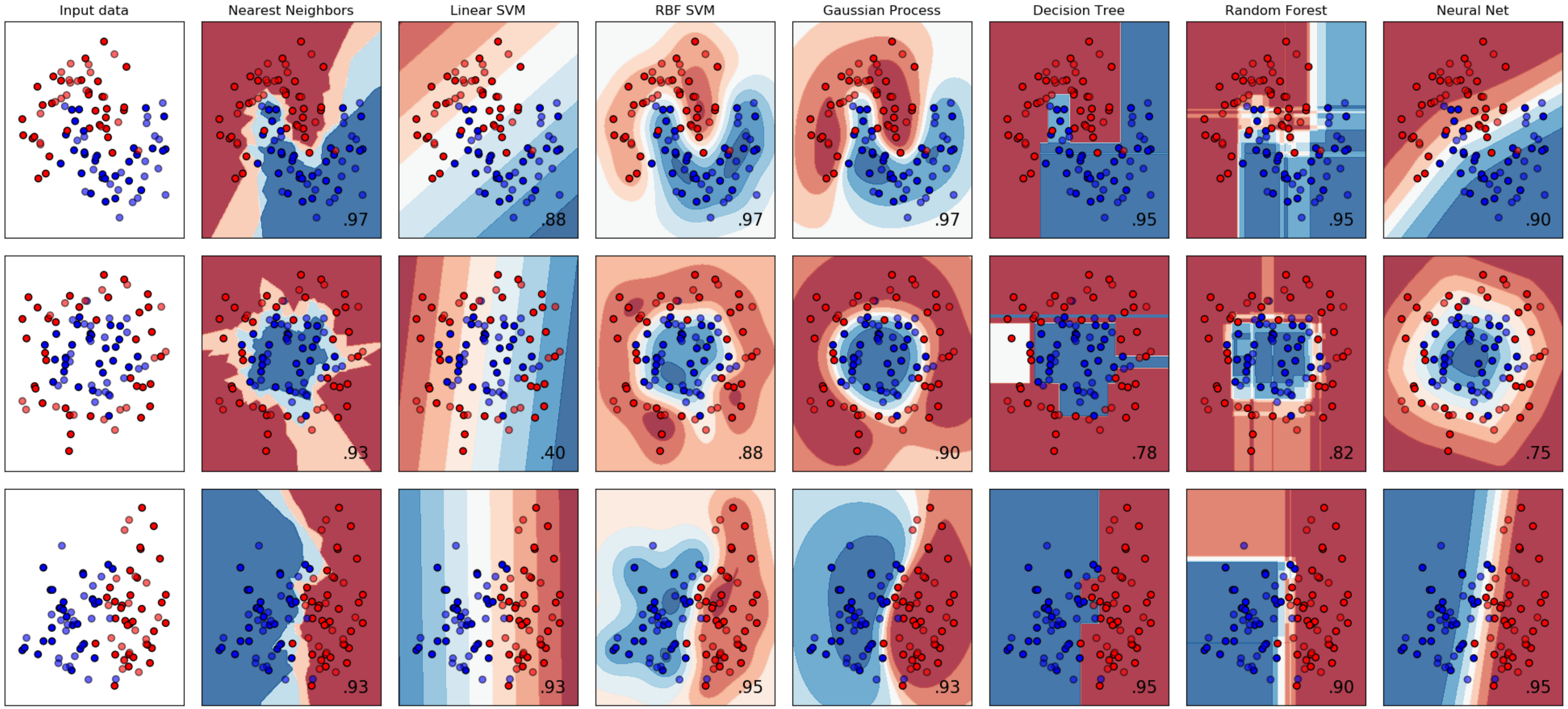
$$\mathcal{D} = \{\mathbf{X}, \mathbf{u}\} \qquad \mathcal{D} = \{(\mathbf{x}_1, u_1), (\mathbf{x}_2, u_2), \ldots, (\mathbf{x}_N, u_N)\}$$

→ **Training set, validation set**
→ **Test set (assessment)**

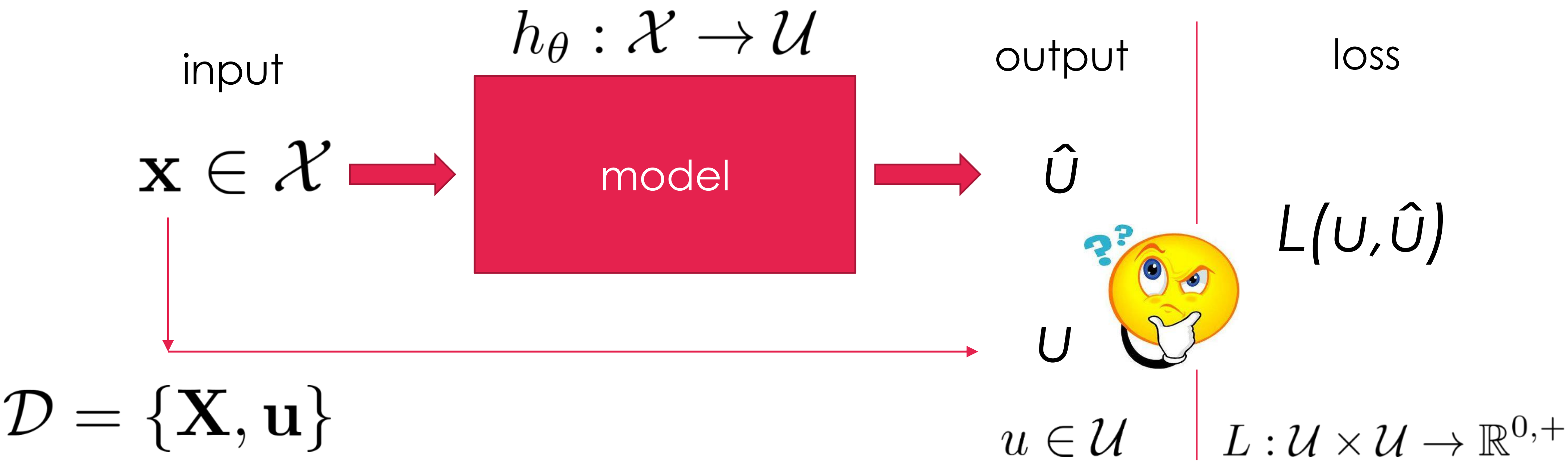
CLASSIFICATION PROBLEM

- Objective : split the space to identify new incomming instance

scikit-learn.org

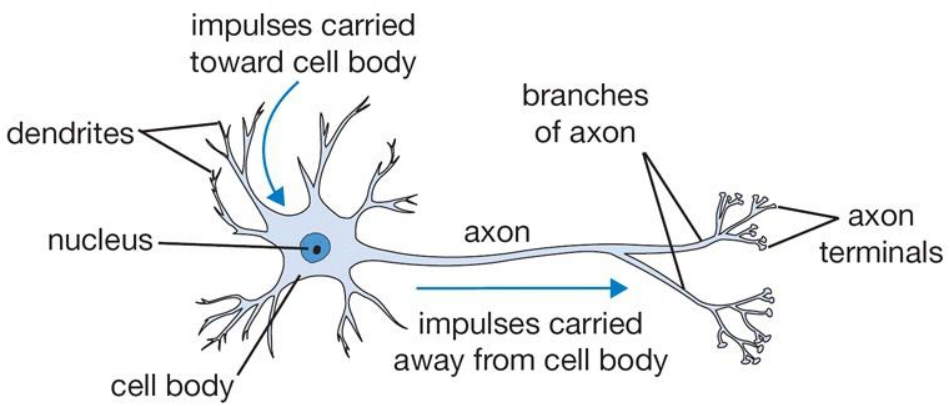


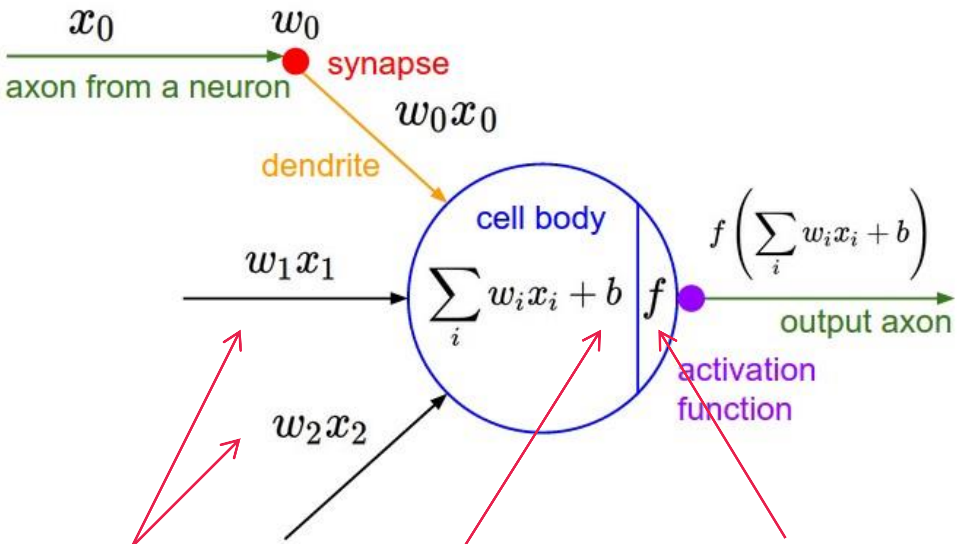
SCHEMA


$h_\theta : \mathcal{X} \rightarrow \mathcal{U}$
input
$\mathbf{x} \in \mathcal{X}$
model
output
û
u
loss
L(u,û)
$\mathcal{D} = \{\mathbf{X}, \mathbf{u}\}$
$u \in \mathcal{U}$
$L : \mathcal{U} \times \mathcal{U} \rightarrow \mathbb{R}^{0,+}$

BRIEFLY, ANN ARE

- Collection of connected **neurons,** structured by layers





weight: w_i

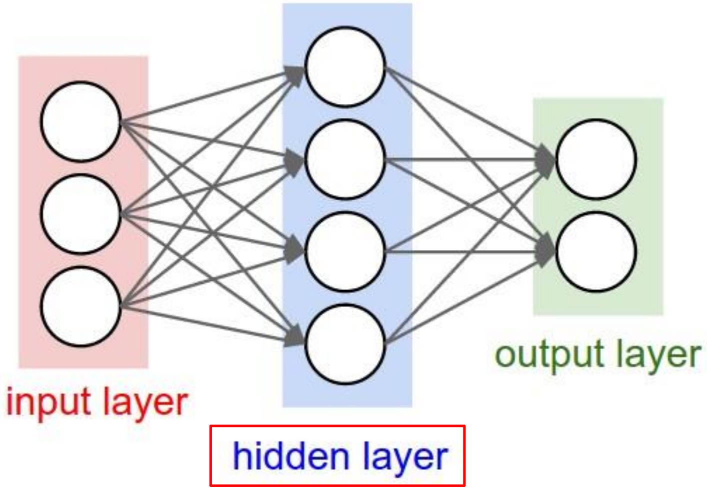
bias : b

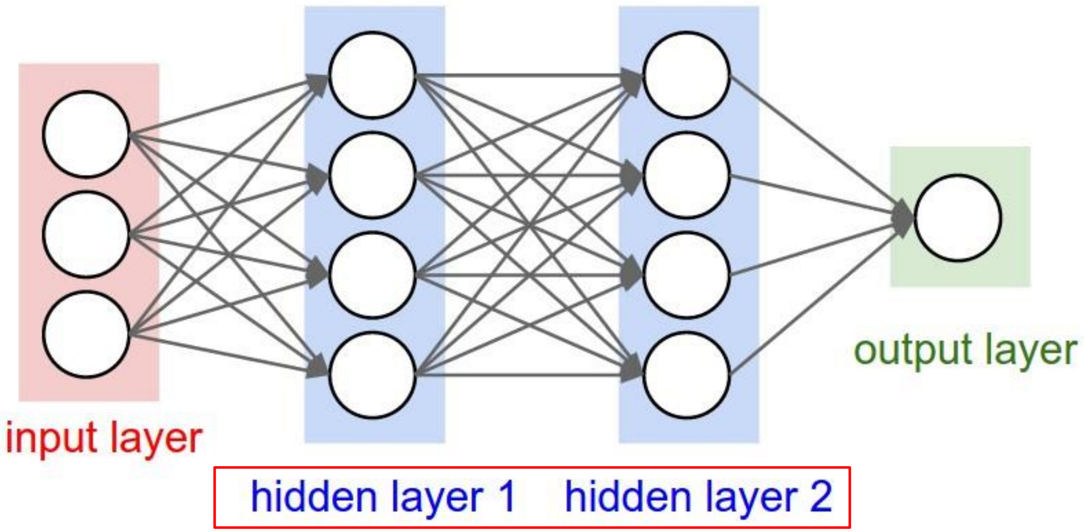
Activation function: $f()$

https://cs231n.github.io/neural-networks-1/

BRIEFLY, ANN ARE

- Collection of connected **neurons**, structured by **layers**





16 + 10 = 26 variables

16 + 20 + 4 = 40 variables

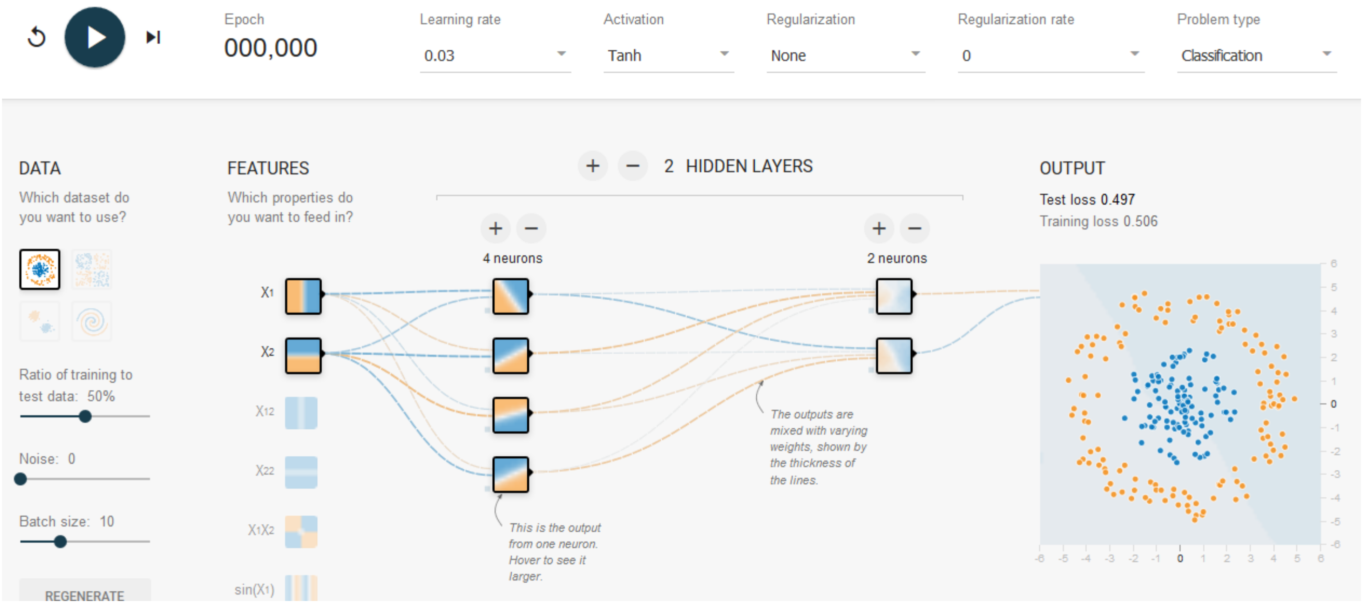
$$\hat{u} = h_\theta(\mathbf{x})$$

https://cs231n.github.io/neural-networks-

PLAYGROUND TENSORFLOW

http://playground.tensorflow.org

- Demo
  - One : simple example
  - Two : XOR case



MODELING ANN TOWARD MATRIX/VECTOR MANIPULATION

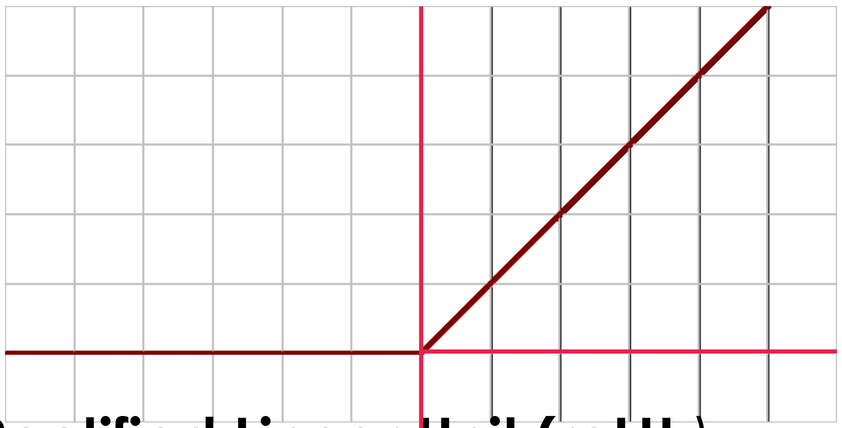
- Neuron with bias : $\mathbf{w}_k$, b_k input : $\mathbf{x}_k$ output : z_k
  - $z_k = \mathbf{w}^T \mathbf{x}_k + b$ $\mathbf{x}$ *of size* K_{l-1}
- A layer *l* (without activation) : $\mathbf{W}_l$, $\mathbf{b}_l$
  - $\mathbf{z}_l = \mathbf{W}_l \mathbf{x} + \mathbf{b}_l$ $\mathbf{W}_l$ *of* K_{l-1} *row by* K_l *lines*

  With $\mathbf{x}^+$ and $\mathbf{W}^+$ are [1,**x**] and [**b**, **W**]
  - $\mathbf{z}_l = \mathbf{W}^+_l \mathbf{x}^+$
- Activation layer
  - $\mathbf{y}_l = f_l(\mathbf{z}_l)$

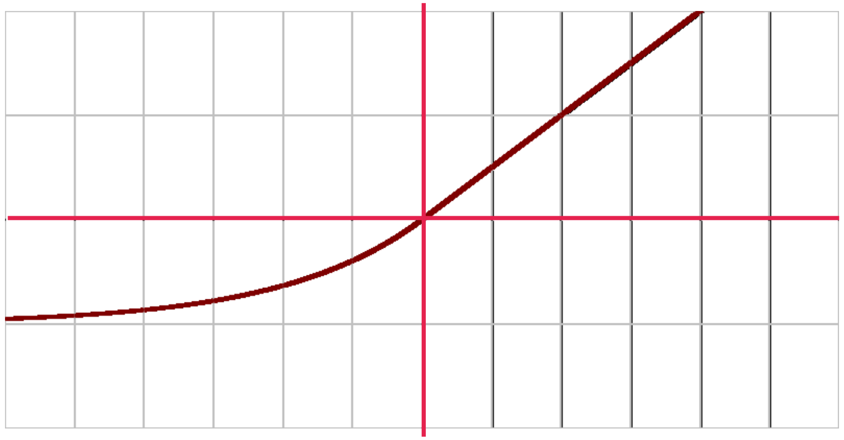
$$z_k^l = {\mathbf{w}_k^l}^T \mathbf{x}_k^l$$

$$y_k^l = f_l(z_k^l)$$

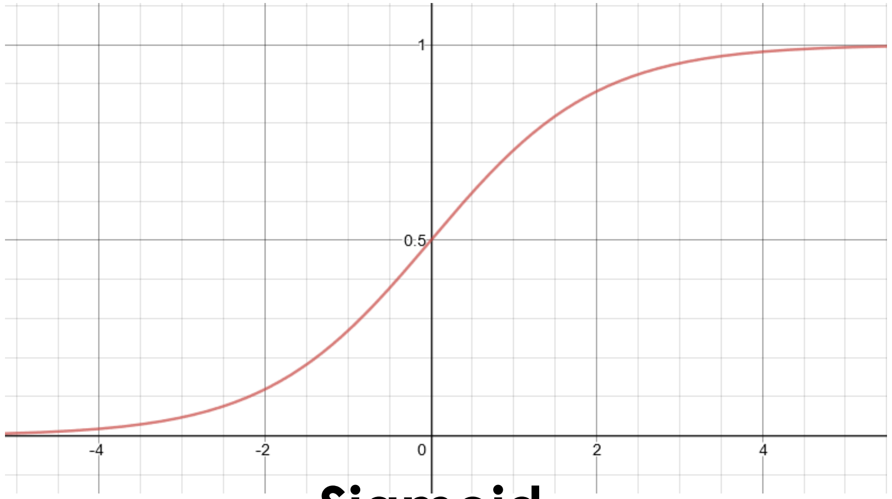
ACTIVATION FUNCTIONS (LOCAL)



Rectified Linear Unit (reLU)
and **Leaky reLU** (one parameter)

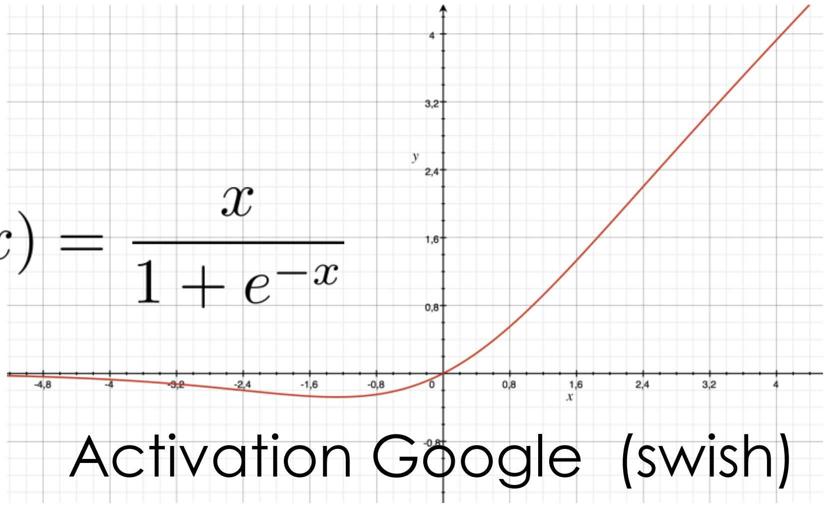


Exponential Linear Unit (eLU)



Sigmoid

$$f(x) = \frac{x}{1 + e^{-x}}$$

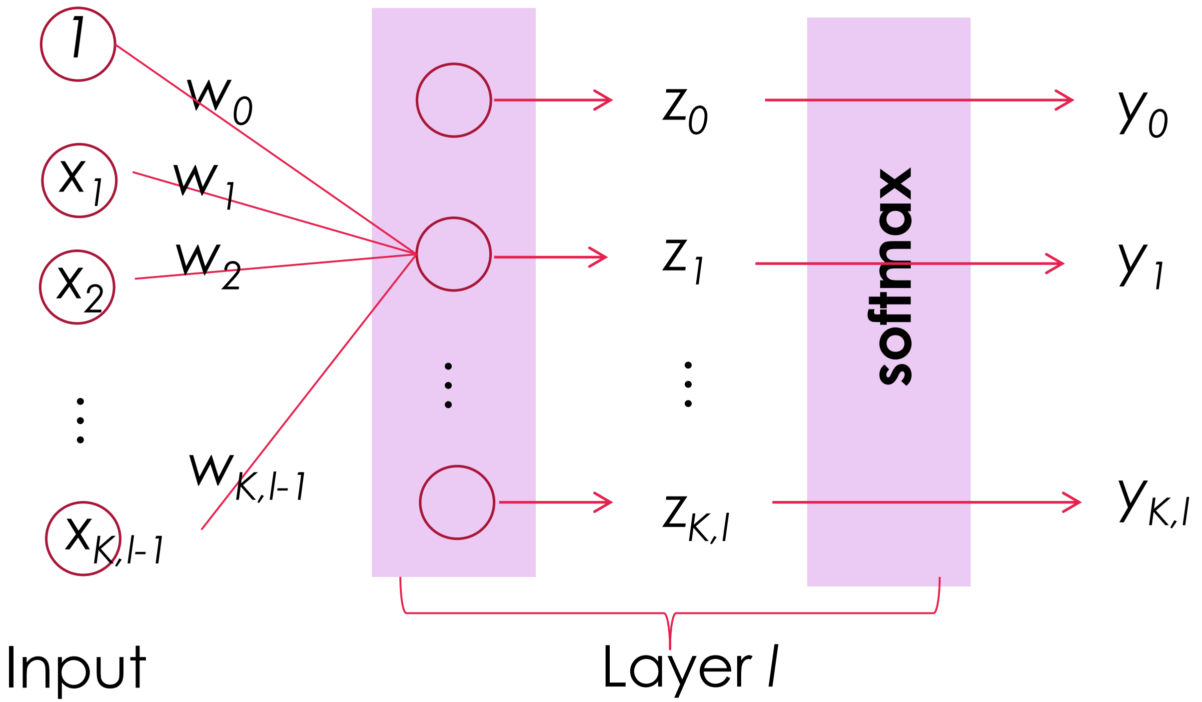


Activation Google (swish)

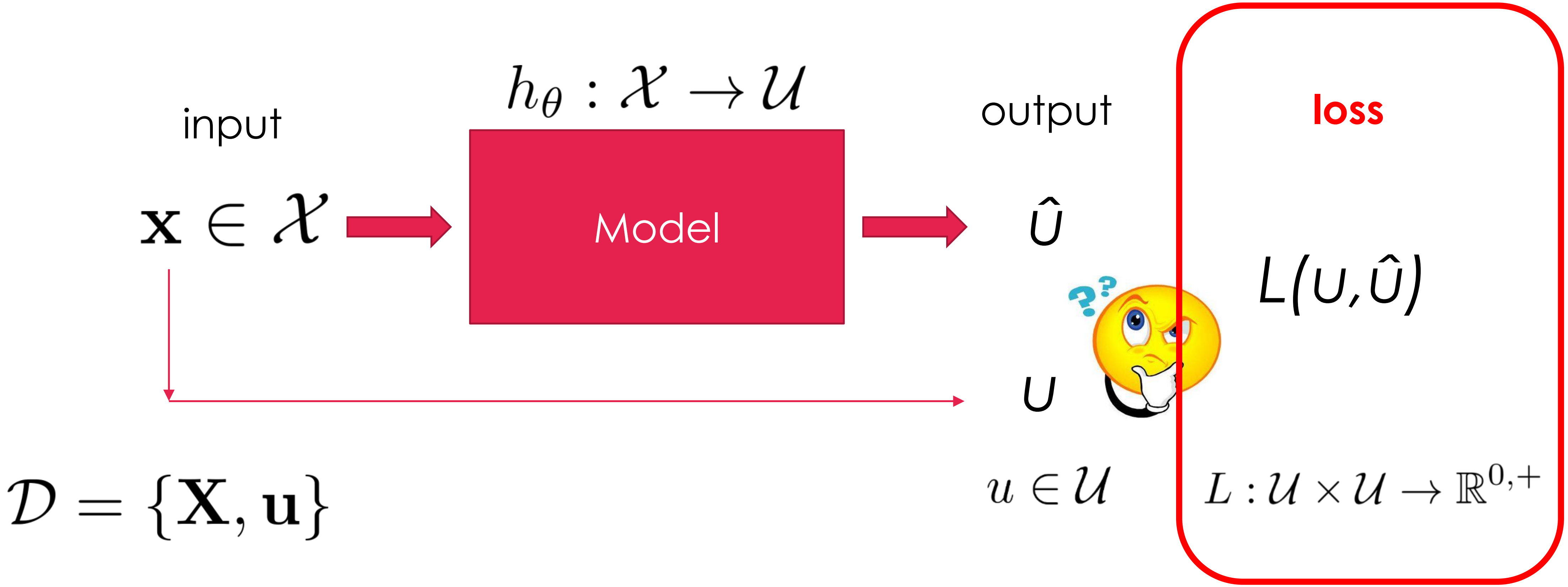
ACTIVATION FUNCTIONS (*OUTPUT* LAYER)

softmax

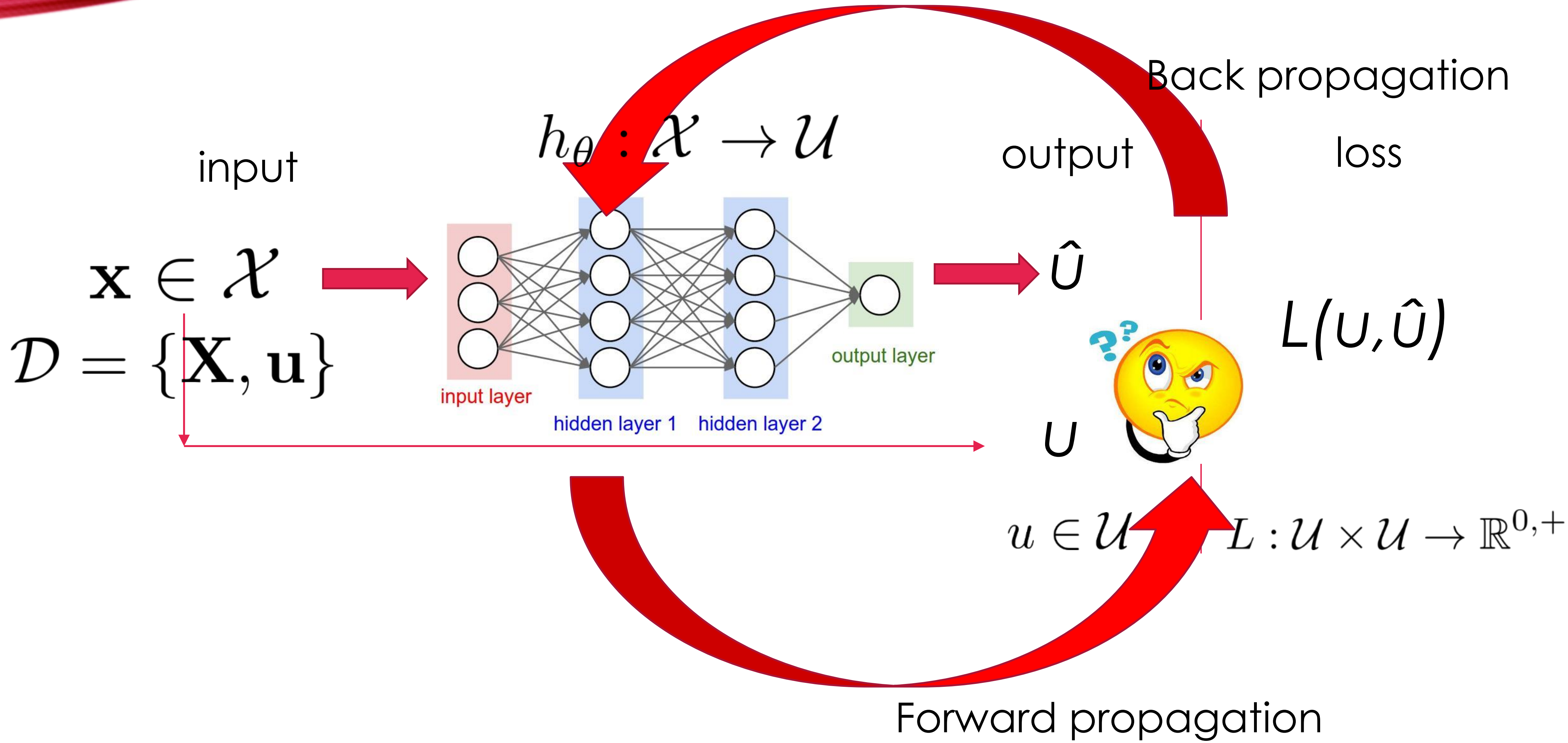
$$f(z_k)_{|l} = \frac{exp(z_k)}{\sum_{i=1}^{K_l} exp(z_i)}$$



SCHEMA


$h_\theta : \mathcal{X} \to \mathcal{U}$
input
output
loss
$\mathbf{x} \in \mathcal{X}$
Model
û
L(u,û)
u
$u \in \mathcal{U}$
$L : \mathcal{U} \times \mathcal{U} \to \mathbb{R}^{0,+}$
$\mathcal{D} = \{\mathbf{X}, \mathbf{u}\}$

SCHEMA, TRAINING



OPTIMIZATION OF W : **TRAINING**

- Loss function must be minimized and will drive the evolution of W: gradient descent

$$\mathbf{W}^{[t+1]} = \mathbf{W}^{[t]} - \eta \nabla_{\mathbf{W}} L(\mathbf{W})$$

Learning rate (η)

- In the perceptron case, Novikoff (1962) prove the convergence in $(D/g)^2$ iterations
  - $||\mathbf{x}^i||_2 < D$
  - $y^i \mathbf{u}^T \mathbf{x}^i \geq g$ with $||\mathbf{u}||_2 = 1$
  - D and g in R^+_*

$$\mathbf{w}_k^{[t+1]} = \mathbf{w}_k^{[t]} - \eta \frac{\partial L(f(\mathbf{w}_k \mathbf{x}^i + b_k), u^i)}{\partial \mathbf{w}_k}$$

OPTIMIZATION W : DERIVATION?

$$\mathbf{W}^{[t+1]} = \mathbf{W}^{[t]} - \eta \nabla_{\mathbf{W}} L(\mathbf{W})$$

- To update the $w^l_{q,k}$ the weight of neuron *q* of layer *l-1* to neuron *k* of layer *l* we need to compute

$$\frac{\partial L(h_{\mathbf{W}}(\mathbf{x}^i), u^i)}{\partial w^l_{q,k}}$$

$$z^l_k = {\mathbf{w}^l_k}^T \mathbf{x}^l_k$$

$$y^l_k = f_l(z^l_k)$$

- Usage of chain rule

$$\frac{\partial L(h_{\mathbf{W}}(\mathbf{x}^i), u^i)}{\partial w^l_{q,k}} = \frac{\partial L(h_{\mathbf{W}}(\mathbf{x}^i), u^i)}{\partial z^l_k} \frac{\partial z^l_k}{\partial w^l_{q,k}}$$

$$= \frac{\partial L(h_{\mathbf{W}}(\mathbf{x}^i), u^i)}{\partial y^l_k} \frac{\partial y^l_k}{\partial z^l_k} \frac{\partial z^l_k}{\partial w^l_{q,k}} = \frac{\partial L(h_{\mathbf{W}}(\mathbf{x}^i), u^i)}{\partial y^l_k} \frac{\partial y^l_k}{\partial z^l_k} \frac{\partial z^l_k}{\partial w^l_{q,k}}$$

OPTIMIZATION W : DERIVATION?

- Matrice view chain rule

$$\mathbf{z}^l = \mathbf{W}^l \mathbf{x}^l + \mathbf{b}^l$$
$$\mathbf{y}^l = f_l(\mathbf{z}^l)$$
$$\mathbf{x}^{l+1} = \mathbf{y}^l$$

$$\frac{\partial L}{\partial \mathbf{W}^l} = \frac{\partial L}{\partial \mathbf{y}^l}\frac{\partial \mathbf{y}^l}{\partial \mathbf{W}^l} = \frac{\partial L}{\partial \mathbf{y}^l}\frac{\partial \mathbf{y}^l}{\partial \mathbf{z}^l}\frac{\partial \mathbf{z}^l}{\partial \mathbf{W}^l}$$

$$= \frac{\partial L}{\partial \mathbf{x}^{l+1}}\frac{\partial \mathbf{y}^l}{\partial \mathbf{z}^l}\frac{\partial \mathbf{W}^l \mathbf{x}^l + \mathbf{b}^l}{\partial \mathbf{W}^l}$$

$$= d'_{l+1} \odot f'_l \frac{\partial \mathbf{W}^l \mathbf{x}^l + \mathbf{b}^l}{\partial \mathbf{W}^l} = (\mathbf{x}^l)^T (d'_{l+1} \odot f'_l)$$

Matrix transpose

Derivative computed for layer l+1
(wrt to its input $\mathbf{x}^{l+1}$)

$$\frac{\partial L}{\partial \mathbf{x}^l} = (\mathbf{W}^l)^T (d'_{l+1} \odot f'_l)$$

Hadamard product
(element wise multiplication)

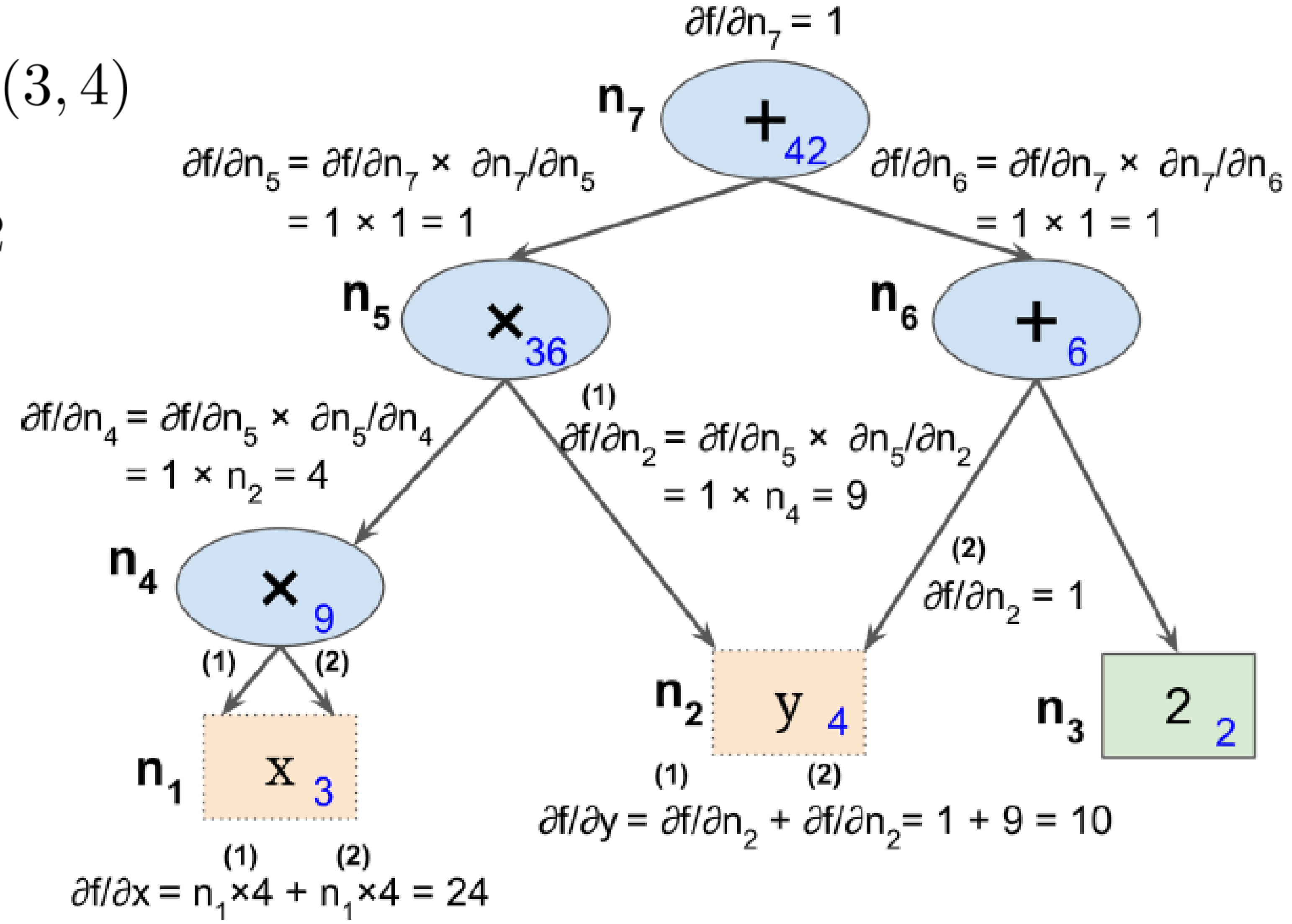
BACKPROPAGATION EXAMPLE

$$\nabla f(3,4)$$

- $f(3,4)$ and

$$f(x,y) = x^2 y + y + 2$$

$$\frac{\partial f}{\partial x} = \frac{\partial f}{\partial n_i} \frac{\partial n_i}{\partial x}$$



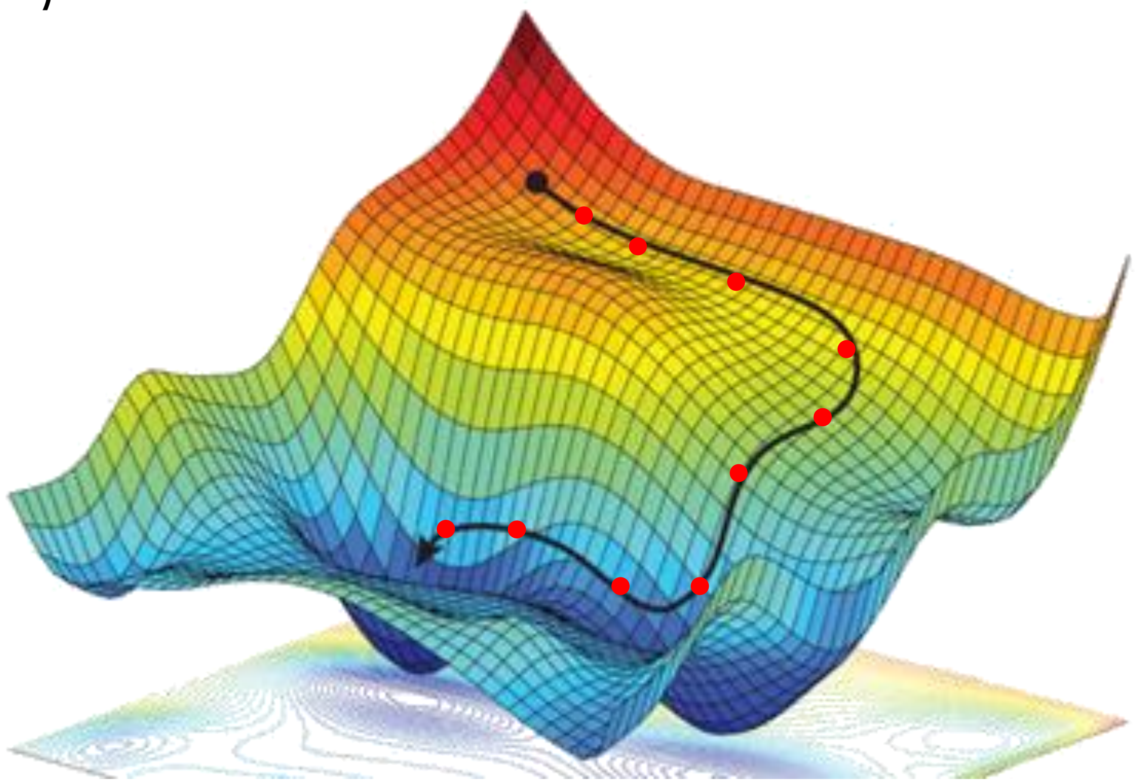
Aurélien Géron:
Hands-on

GRADIENT DESCENT

$$\mathbf{W}^{[t+1]} = \mathbf{W}^{[t]} - \eta \underline{\nabla_{\mathbf{W}} L(\mathbf{W})}$$

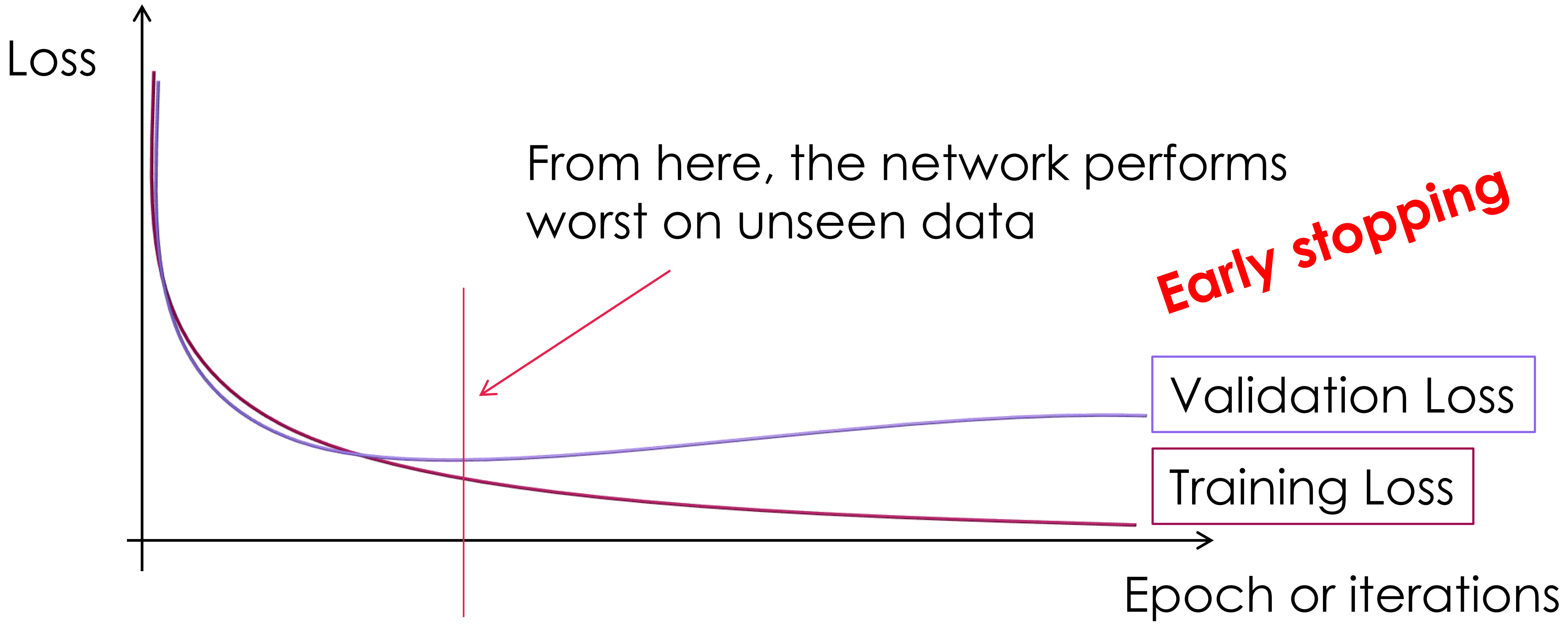
= 0 ? → minimum

- Iterative process
- Can be regularized (**momentum**, **Adam**)
- 3 ways of using data
  - Use the whole dataset
    - Long, small update
  - Use one instance (Stochastic)
    - Fast but noisy
  - Use a couple of instances
    - → mini batch
    - Quite fast, less noisy



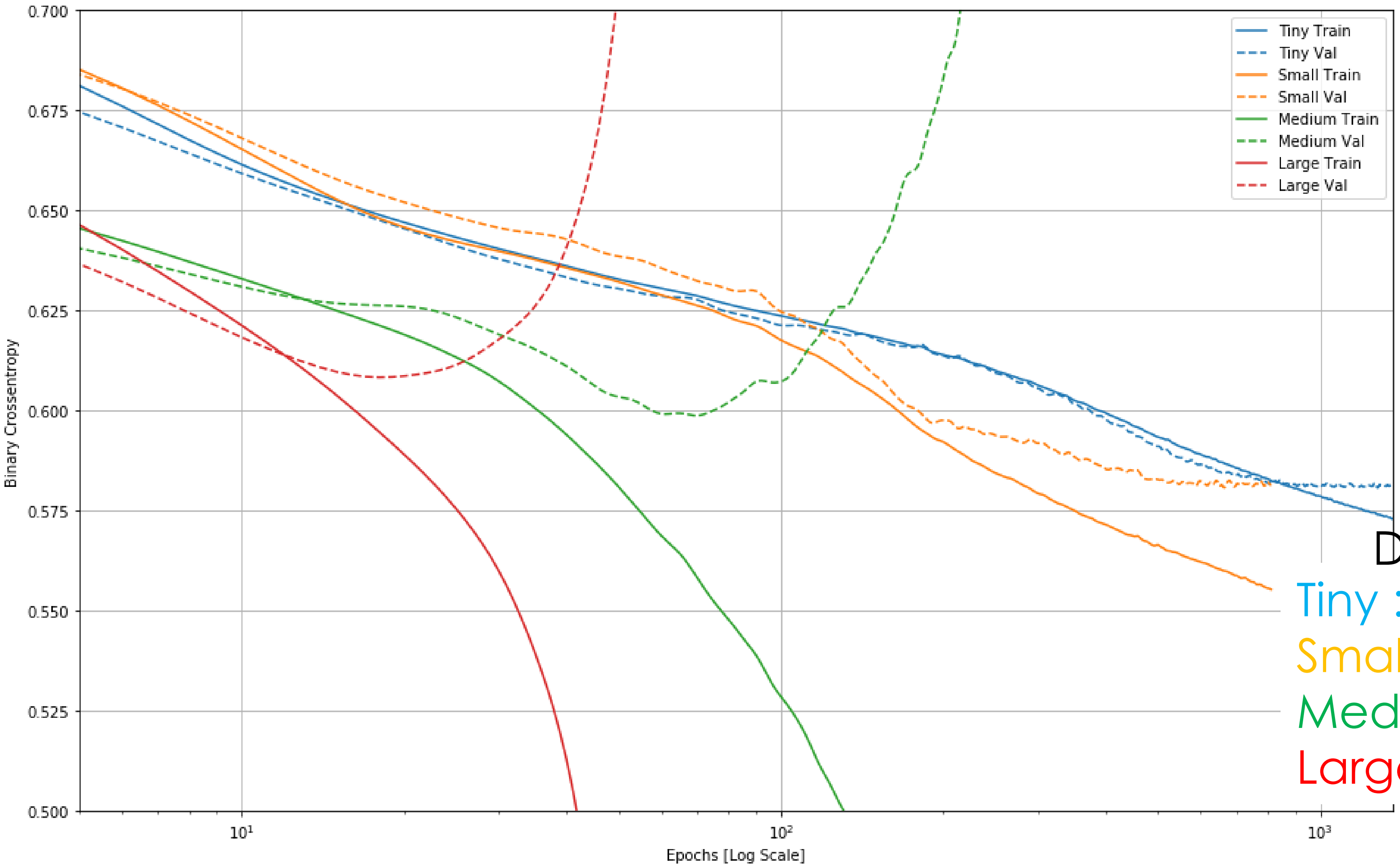
LOSSES EVOLUTION DURING TRAINING

- Loss on training set → learning the problem
- Loss on validation set → generalization ability



REAL EVOLUTION CURVES…

- https://www.tensorflow.org/tutorials/keras/overfit_and_underfit



Dense networks

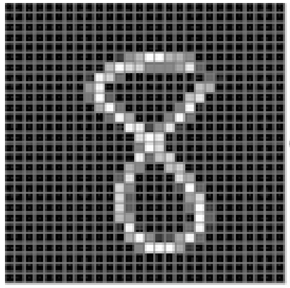
Tiny : 16

Small : 16, 16

Medium:64,64,64

Large: 512, 512, 512, 512

MLP : CLASSIFICATION EXAMPLE



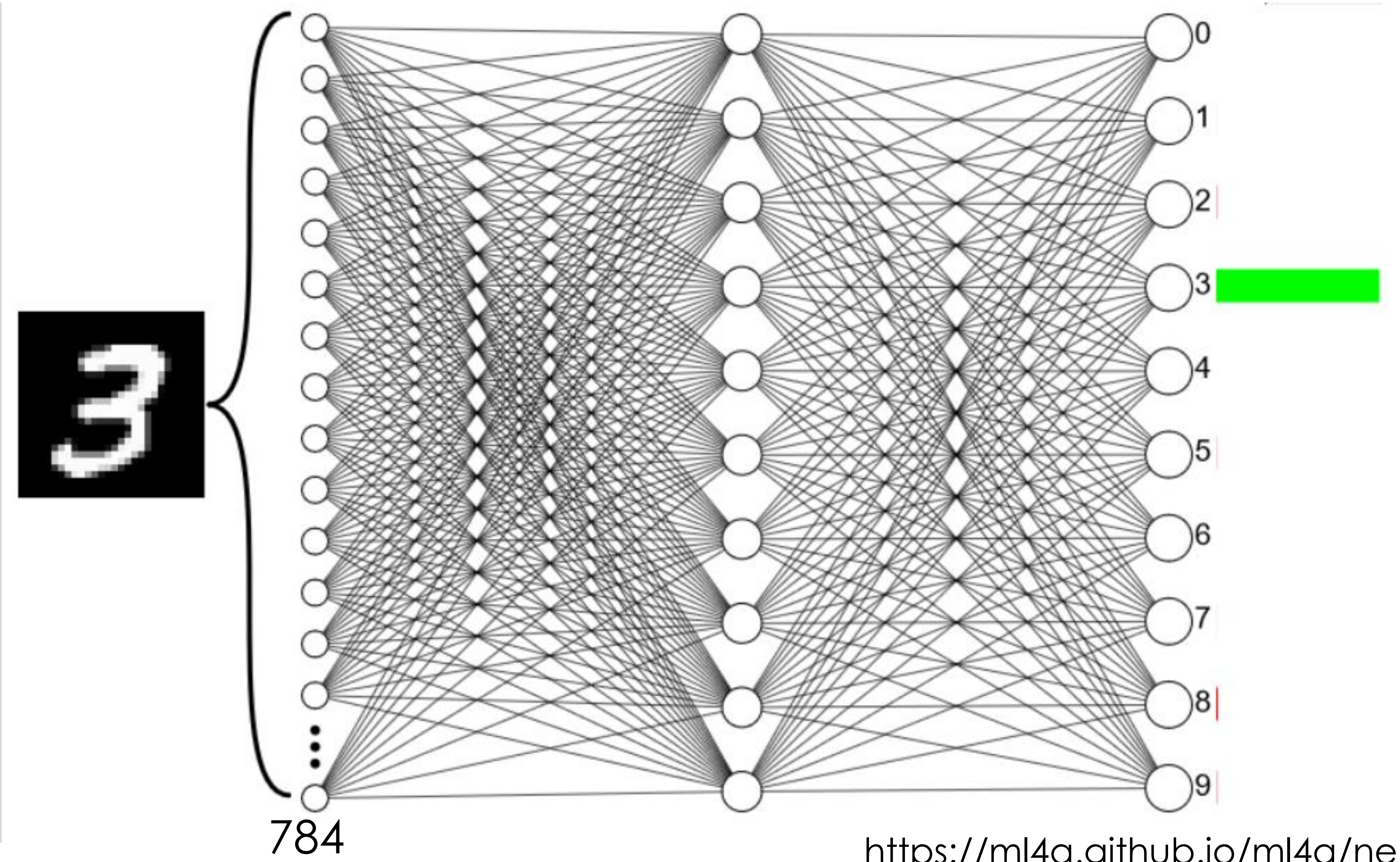
28 x 28
784 pixels

→

0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	4	62	146	182	254	254	181	176	139	15	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	34	186	253	217	208	136	136	136	166	232	99	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	61	242	208	111	3	0	0	0	0	0	18	32	107	43	0	0	0	0	0	0
0	0	0	0	0	0	0	0	156	242	23	0	0	0	0	0	0	0	13	191	181	6	0	0	0	0	0	0
0	0	0	0	0	0	0	0	121	255	98	3	0	0	0	0	0	8	194	225	12	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	169	253	120	3	0	0	0	0	128	247	51	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	3	111	244	169	19	0	14	131	249	117	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	59	241	235	72	142	229	66	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	25	218	254	231	36	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	133	253	221	33	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	19	237	111	196	217	19	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	174	138	0	23	193	204	18	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	96	224	0	0	0	25	218	169	3	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	215	138	0	0	0	0	86	253	99	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	215	97	0	0	0	0	3	162	214	11	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	215	97	0	0	0	0	0	118	253	68	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	185	157	0	0	0	0	0	40	254	98	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	50	244	61	0	0	0	0	112	244	58	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	174	251	142	59	83	167	244	111	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	6	133	253	253	253	169	61	3	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

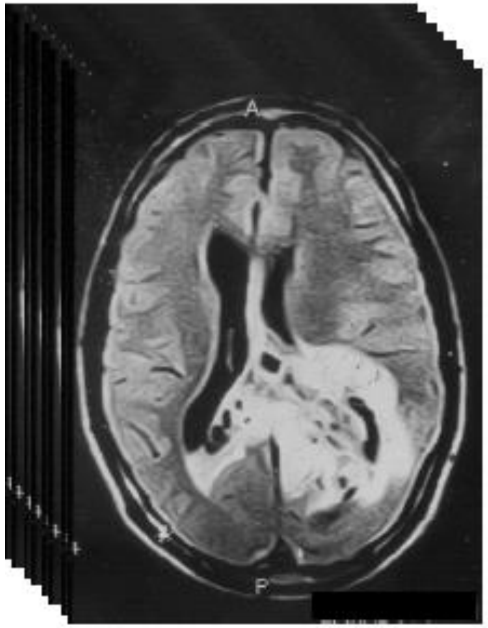
https://ml4a.github.io/ml4a/neural_networks/

MLP : CLASSIFICATION EXAMPLE

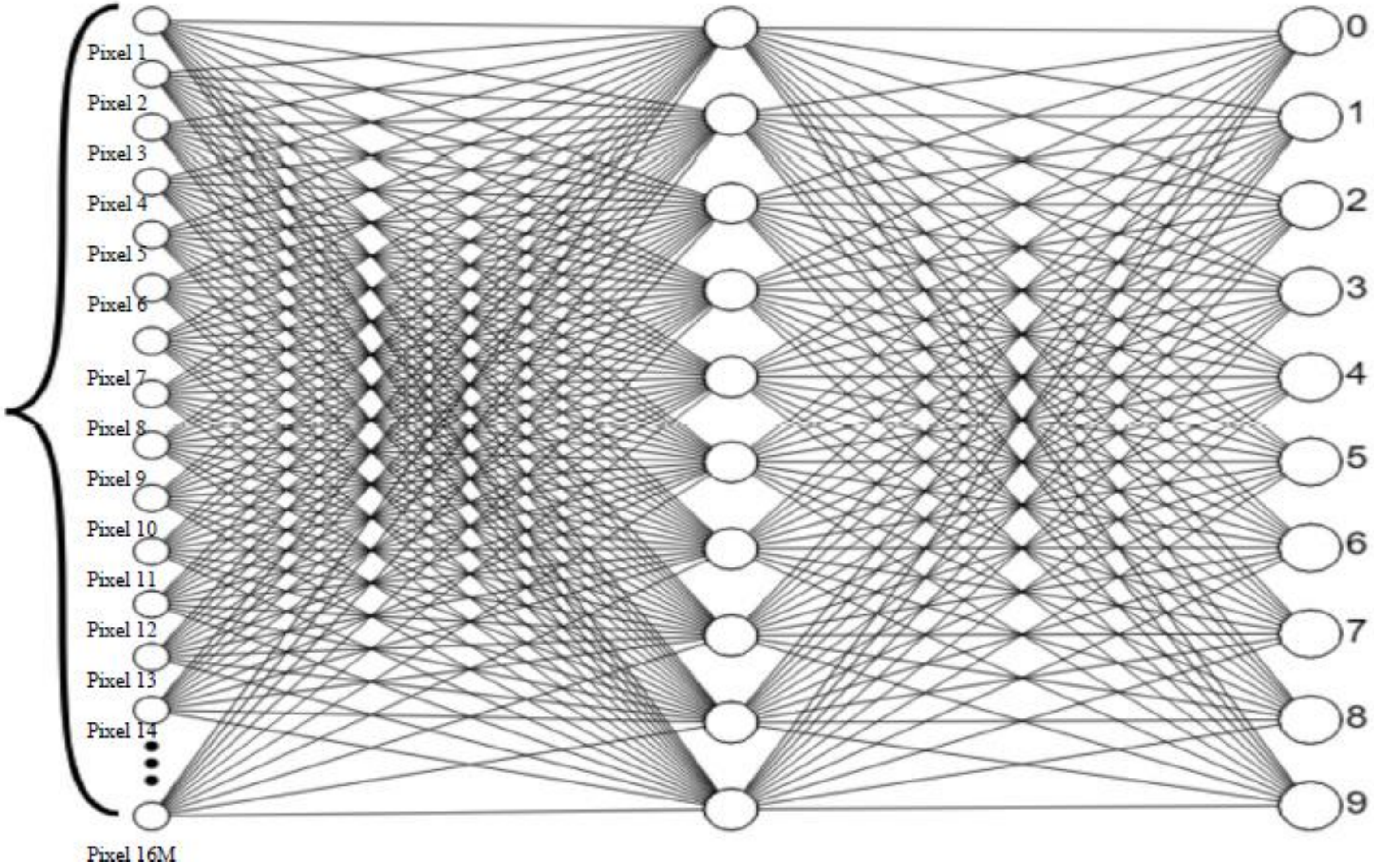


https://ml4a.github.io/ml4a/neural_networks/

MLP: LIMITATIONS



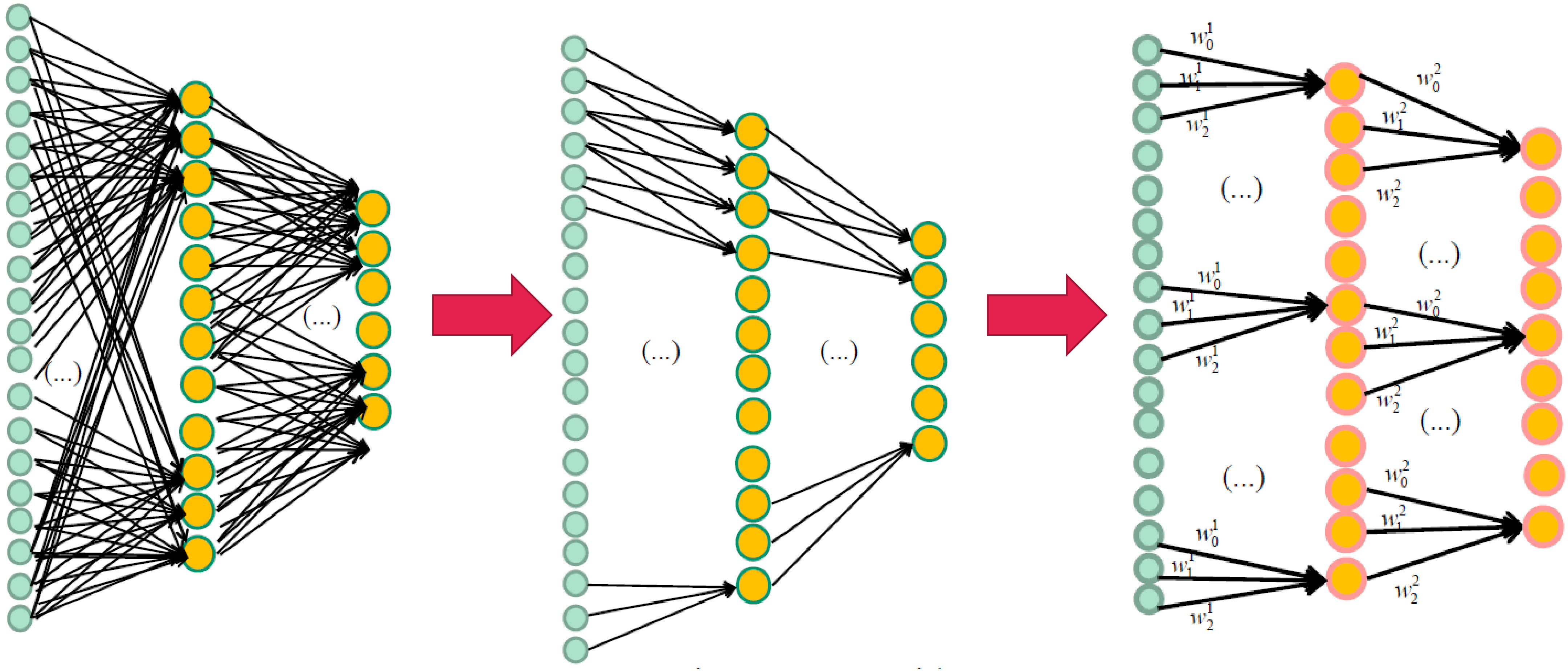
256x256x256



160M parameters for the first layer!

MLP LIMITATIONS AND SOLUTION!

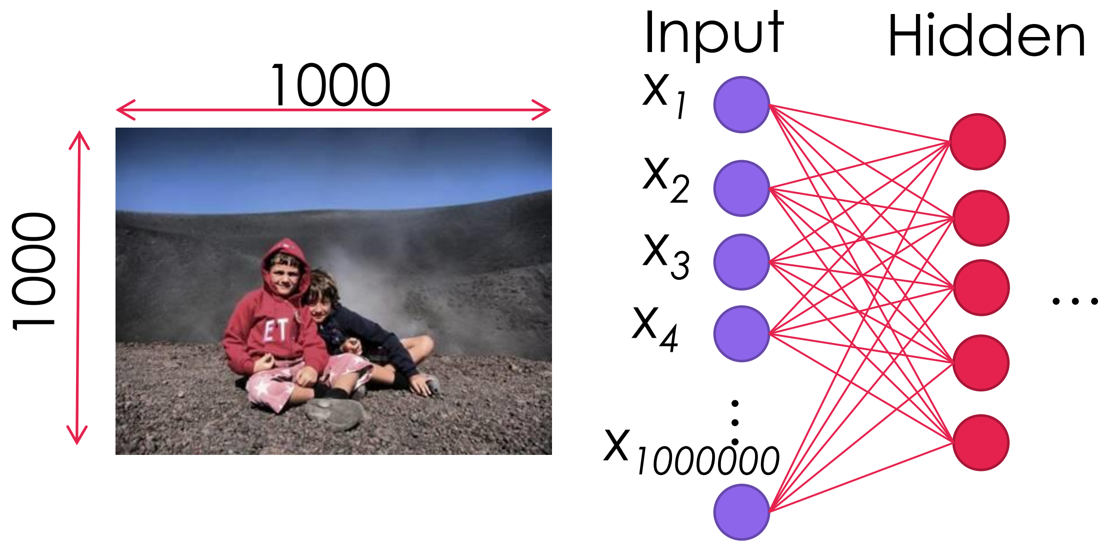
Fully connected | Not fully connected | **Sharing weights**



II – CONVOLUTIONAL NEURAL NETWORK : ELEMENTS

Convolution and related dedicated layers

CNN, YANN LECUN



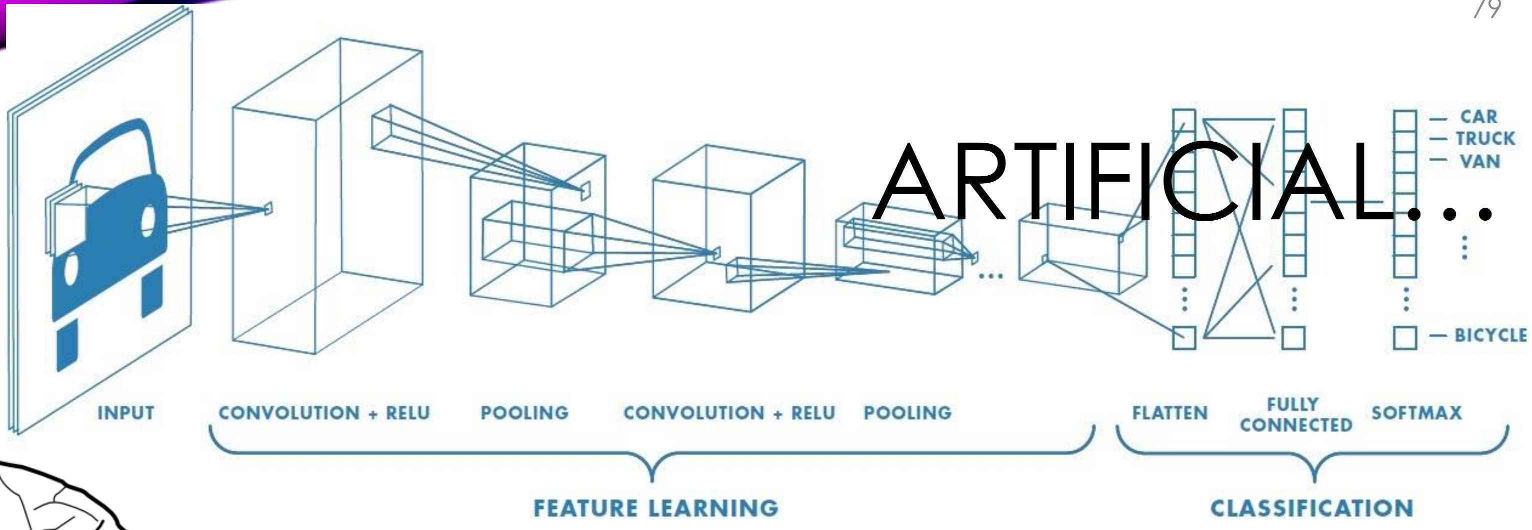
➔ Too much dimnesions!
➔ Use carrefully the *fully connected*

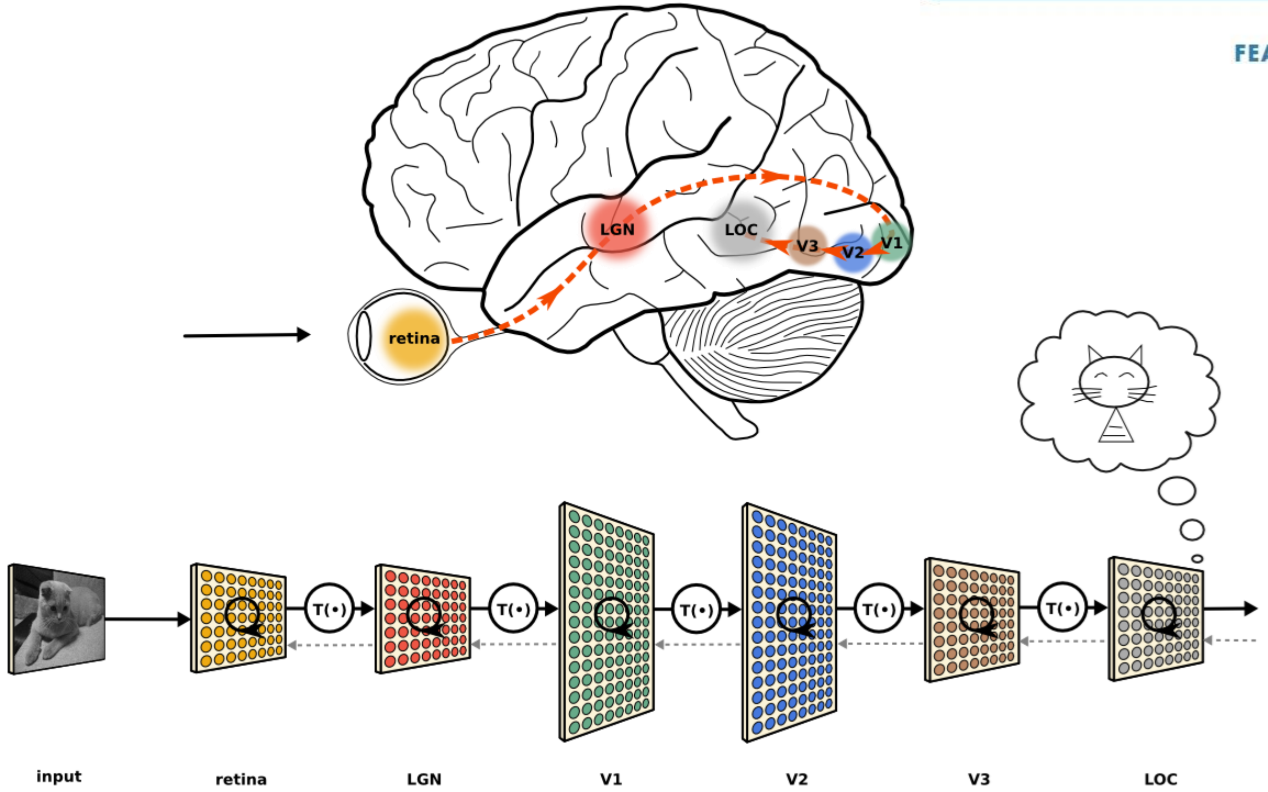
« malédiction de la grande dimension »
Le nombre d'exemples d'apprentissage doit augmenter exponentiellement avec le nombre de dimensions

➔ We have to reduce the number of dimensions if we don't want some influence on result
➔ 2 properties of natural and numeric data (link to human)
  - ➔ Multiscale hierarchic organization
  - ➔ **Symetries**

tinyurl.com/cours-Mallat

ARTIFICIAL…

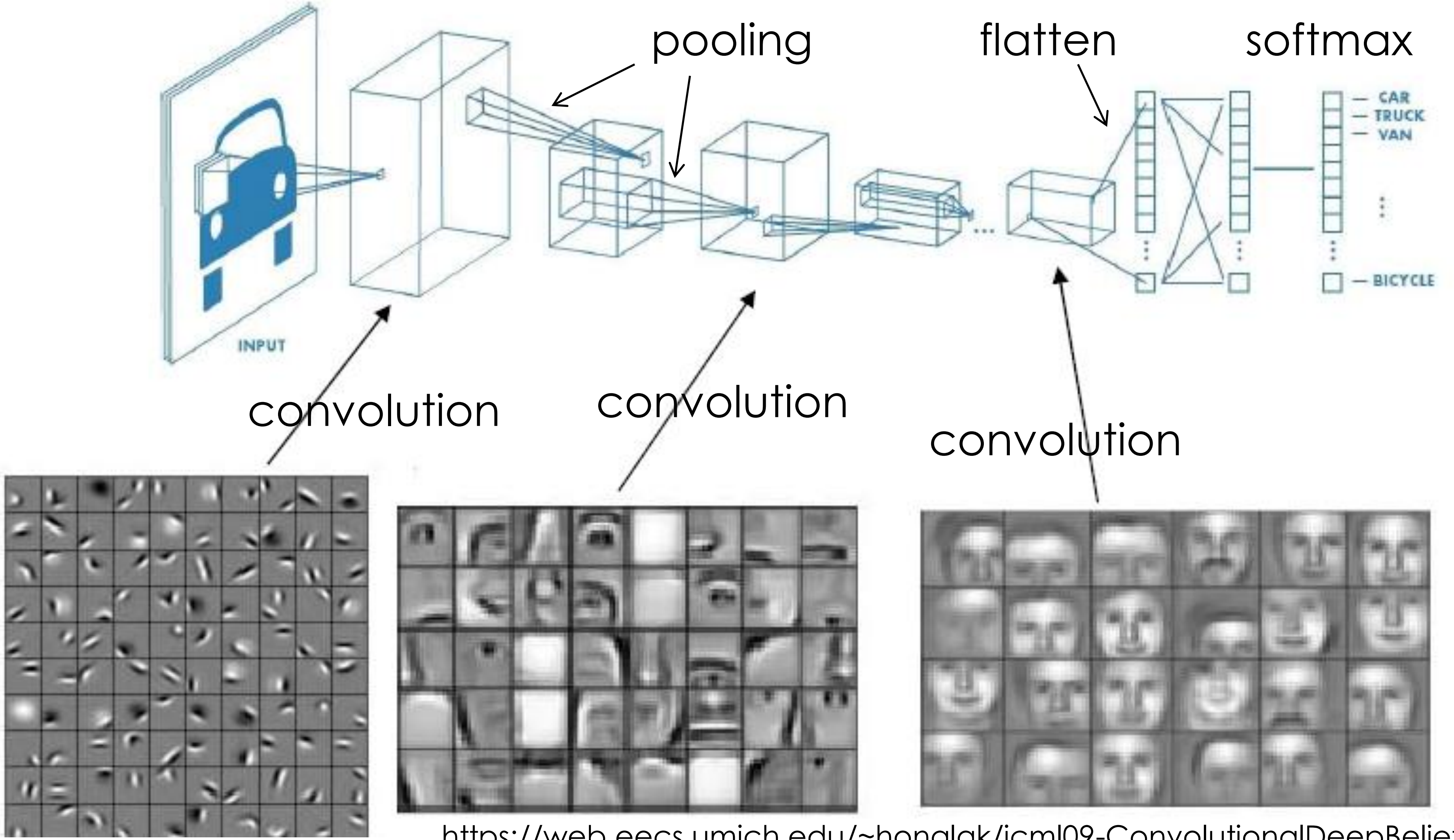




… vs natural

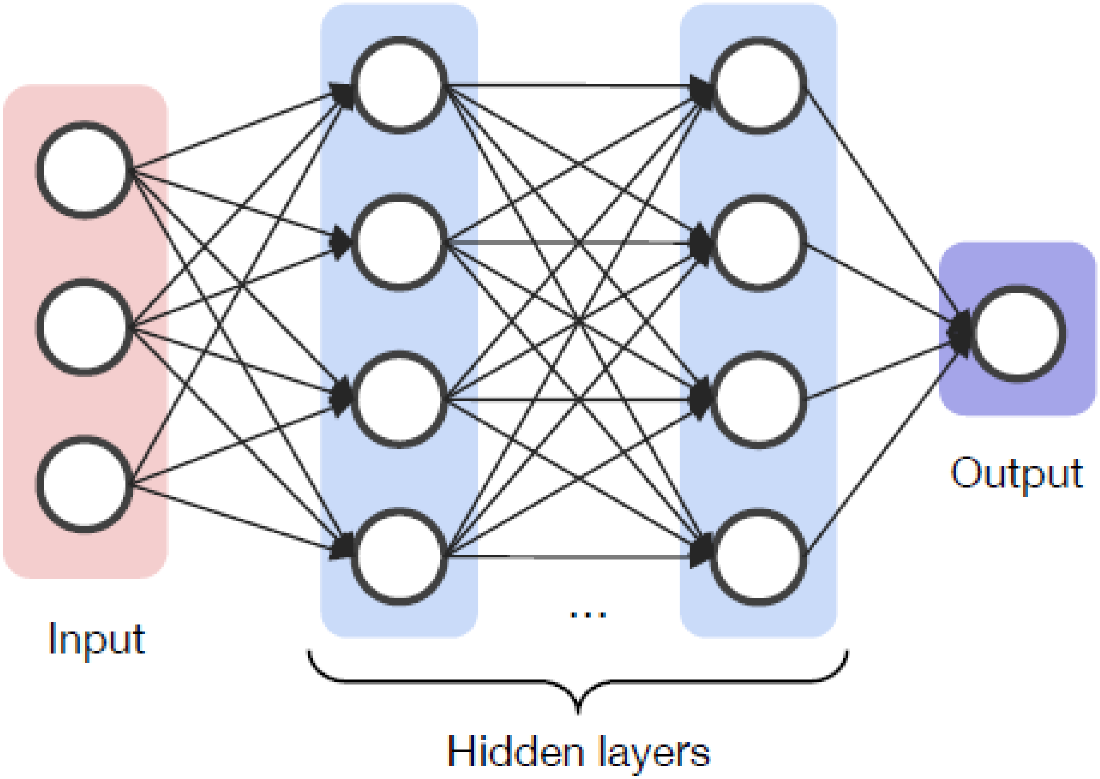
https://neuwritesd.files.wordpress.com/2015/10/visual_stream_small.png
https://fr.mathworks.com/solutions/deep-learning/convolutional-neural-network.htm

TYPICAL CNN (CLASSIFICATION)



https://web.eecs.umich.edu/~honglak/icml09-ConvolutionalDeepBeliefNetworks.pdf

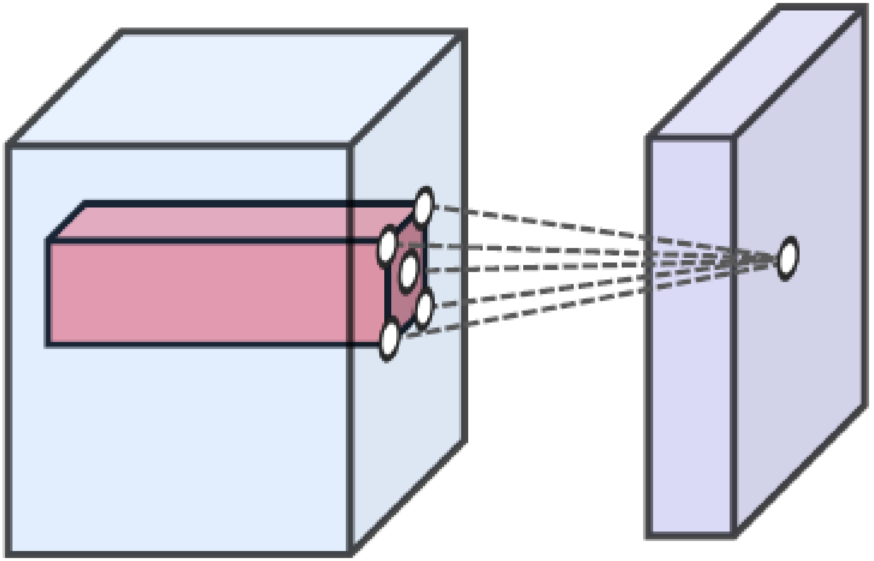
Multi-layer perception (MLP)



Many parameters

Limited depth

CNN



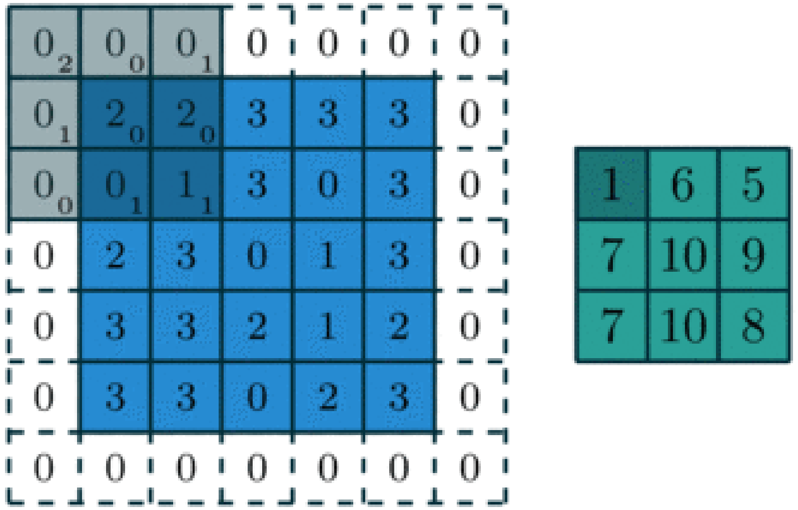
- Shared parameters
- Local connectivity

Less parameters

Deep architectures possible

CONVOLUTION FOR NEURAL NETWORK

- **Convolution**
- Pad
  - border condition
- Stride
  - jump
- Atrous
  - empty pixels in mask

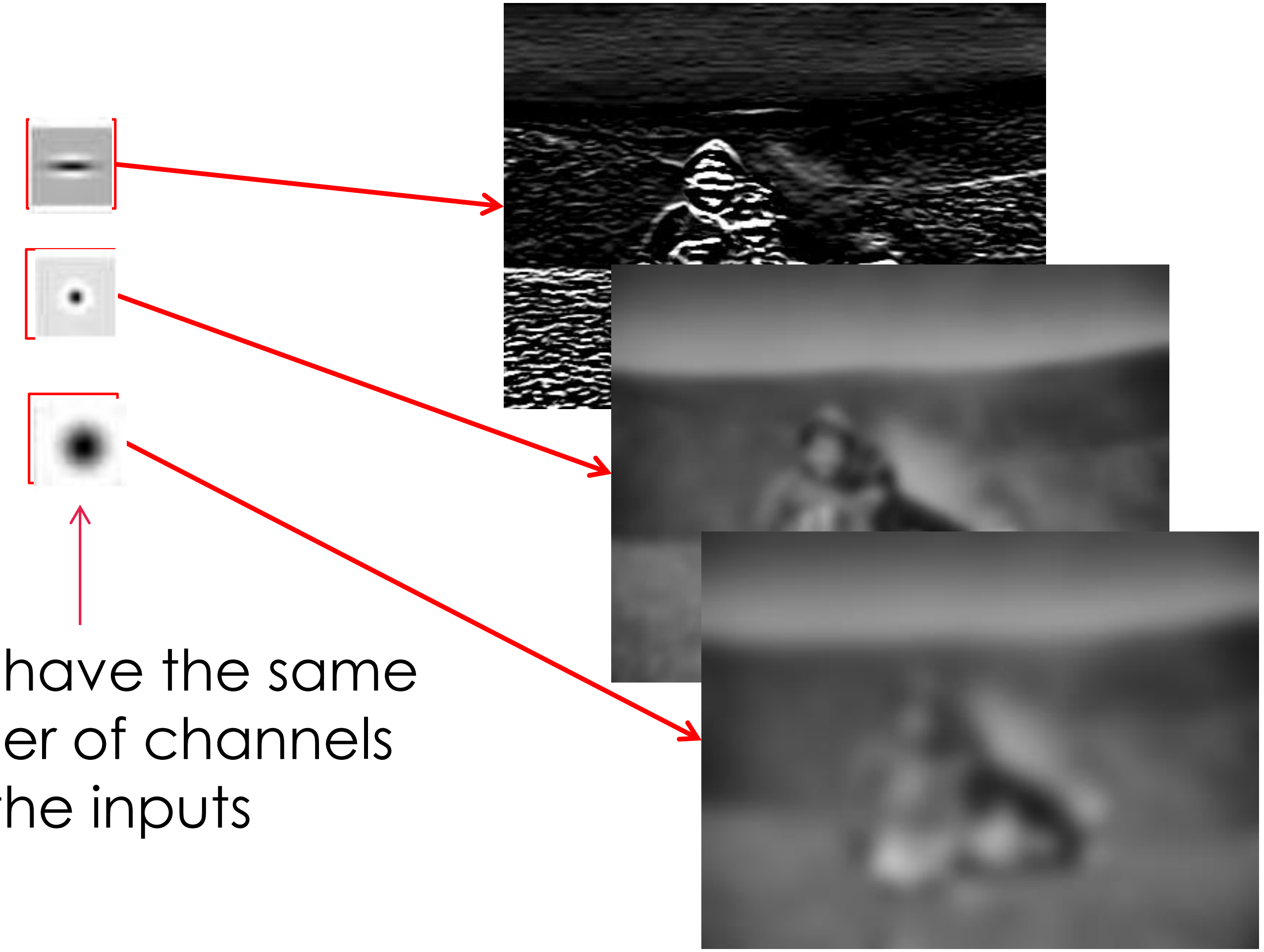


http://deeplearning.net/software/theano/tutorial/conv_arithmetic.html#zero-padding-unit-strides

CONVOLUTION, FILTERS AND CHANNELS

One input channel

convolution



Filters have the same number of channels than the inputs

Three filters →Three output channels

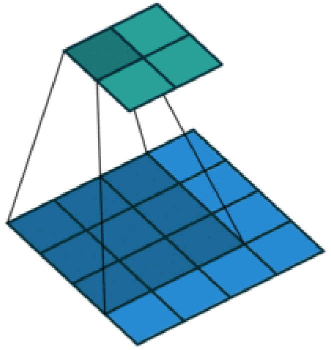
PRE-PROCESSING 'LAYERS'

Intensity values of input data are critical: how to warrant that the dynamic and statistic are similar on all inputs ? → ***pre-process all data!***

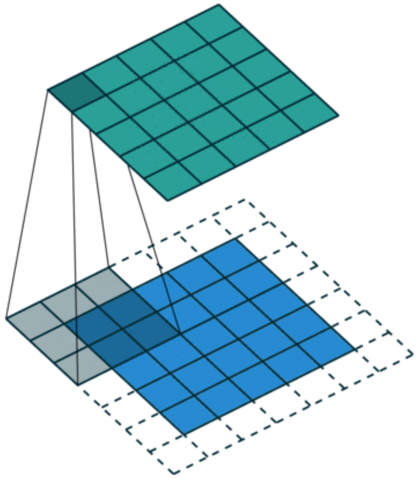
- Mean-substraction
- Normalizatation
- Local contrast normalisation
- (size adaptation and interpolation)

$$\mathbf{x_0} = \mathbf{x} - \hat{\mathbf{x}}$$

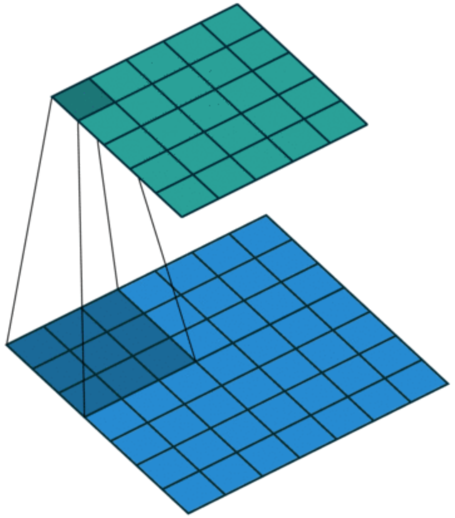
$$\mathbf{x_1} = \frac{\mathbf{x_0}}{\sigma}$$



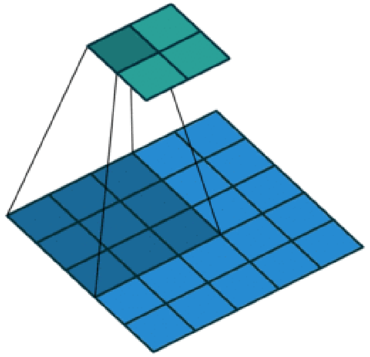
No zero padding,
unit stride



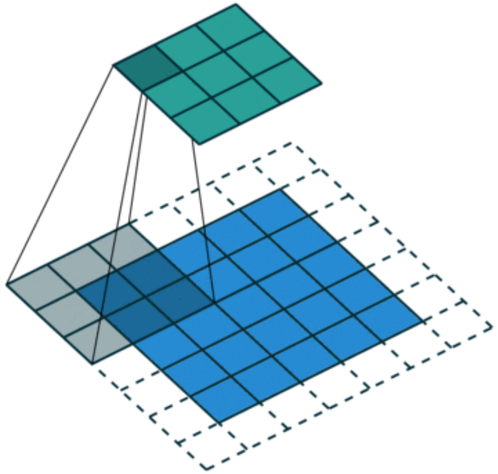
Half size padding ('same' size)
Unit stride



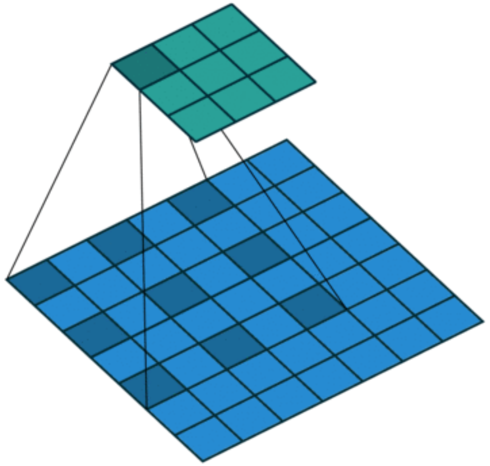
Full padding, Unit stride



No zero padding,
Non-unit stride



Half size padding ('same' size)
Non-unit stride

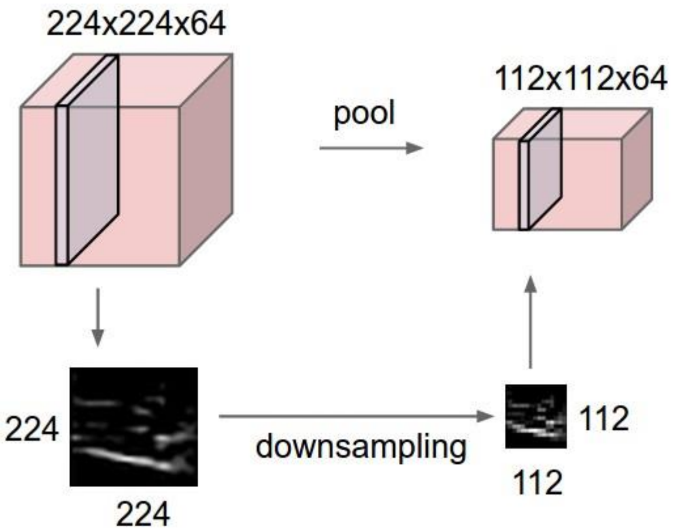


Atrous or dilated filter

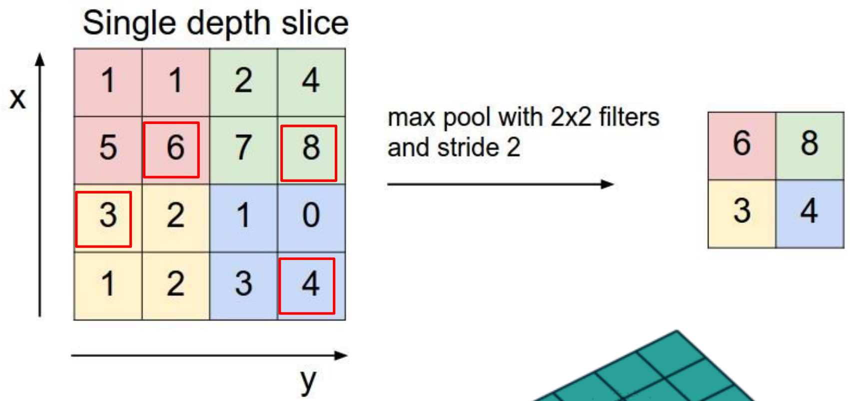
POOLING

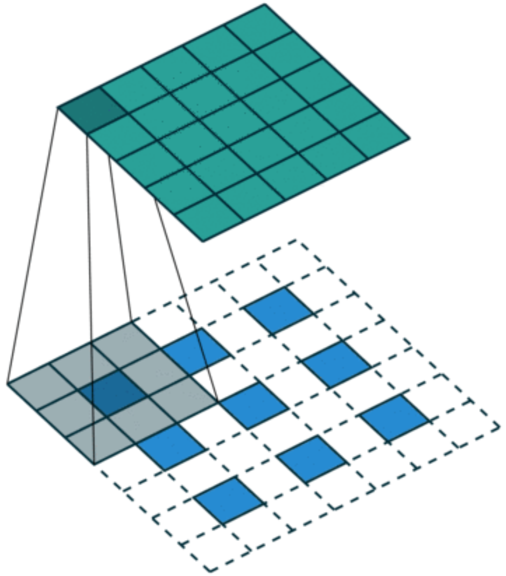
https://cs231n.github.io/convolutional-networks/

- Down sampling
  - **max**,
  - mean
  - …
- Up sampling
  - Nearest Neighbo
  - deconvolution – t



Max pooling

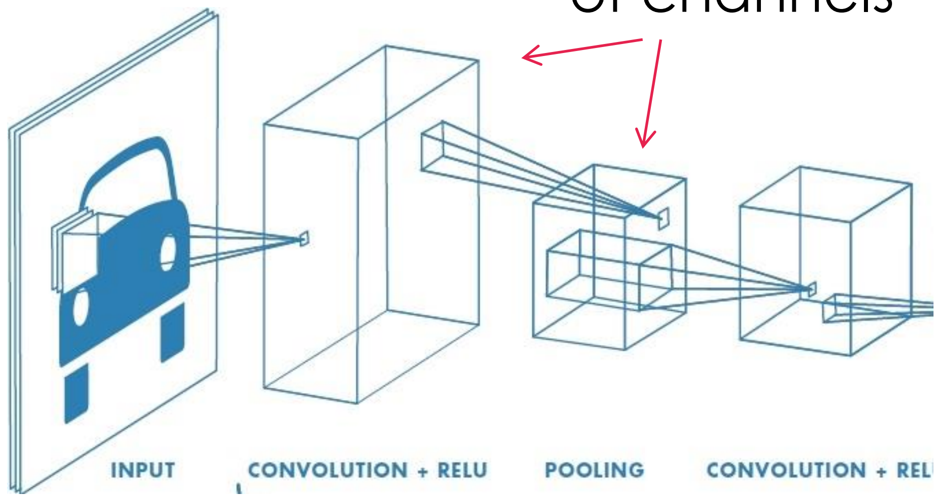




DON'T MISS IT

- Convolution works on all channels
  - Generaly 3 by 3, 5 by 5, 7 by 7 or 1 by 1
  - At a given layer, many channels are obtained, one per convolution
- Pooling works on channel per channel
  - Generaly size is decreased by 4 (2 by 2)
  - (it can be increased)
- Receptive field

Same number of channels

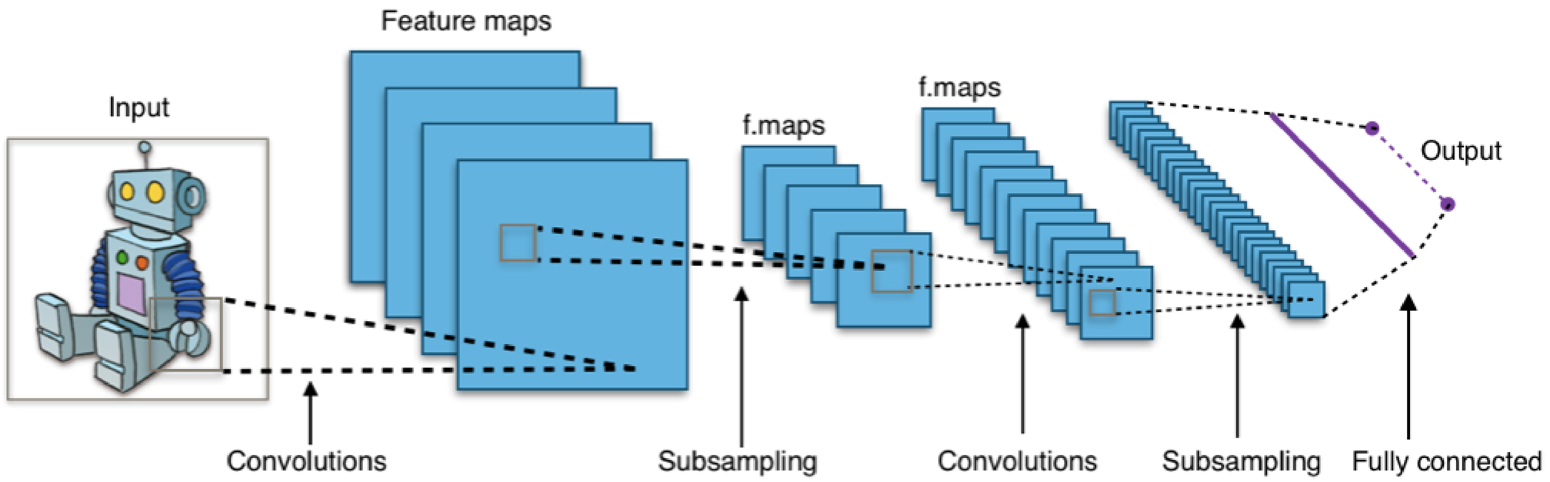


Input channels

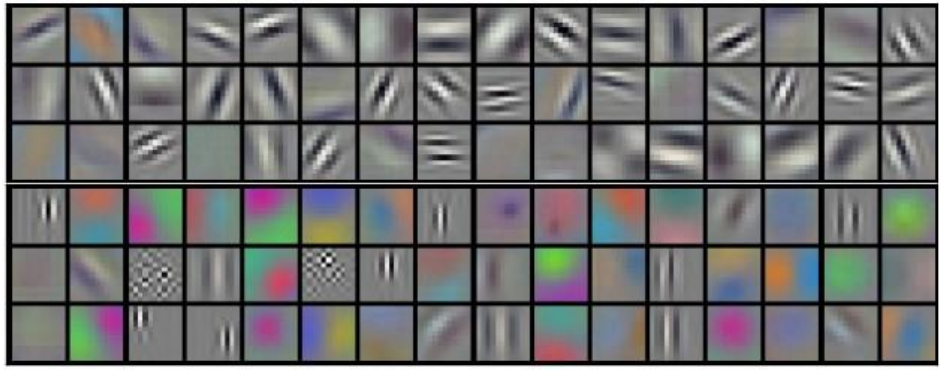
APPLICATIONS: CLASSIFICATION AND SEGMENTATION

1. CNN

- Classification, détection, régression

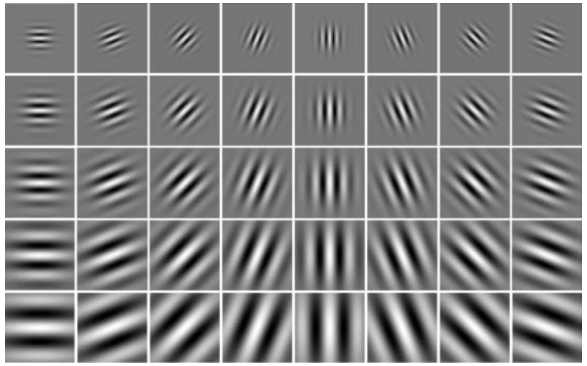


96 filtres (ImageNET)



[Krizhevsky 12]

Filtres Gabor



2. U-NET (2015)

- Segmentation
- Filtering

➔Output

Same support as input