IFT 615 – Intelligence Artificielle

Application – vision artificielle

Professeur: Froduald Kabanza

Assistants: D'Jeff Nkashama et Léo Chartrand

UDS Université de Sherbrooke

Sujets couverts

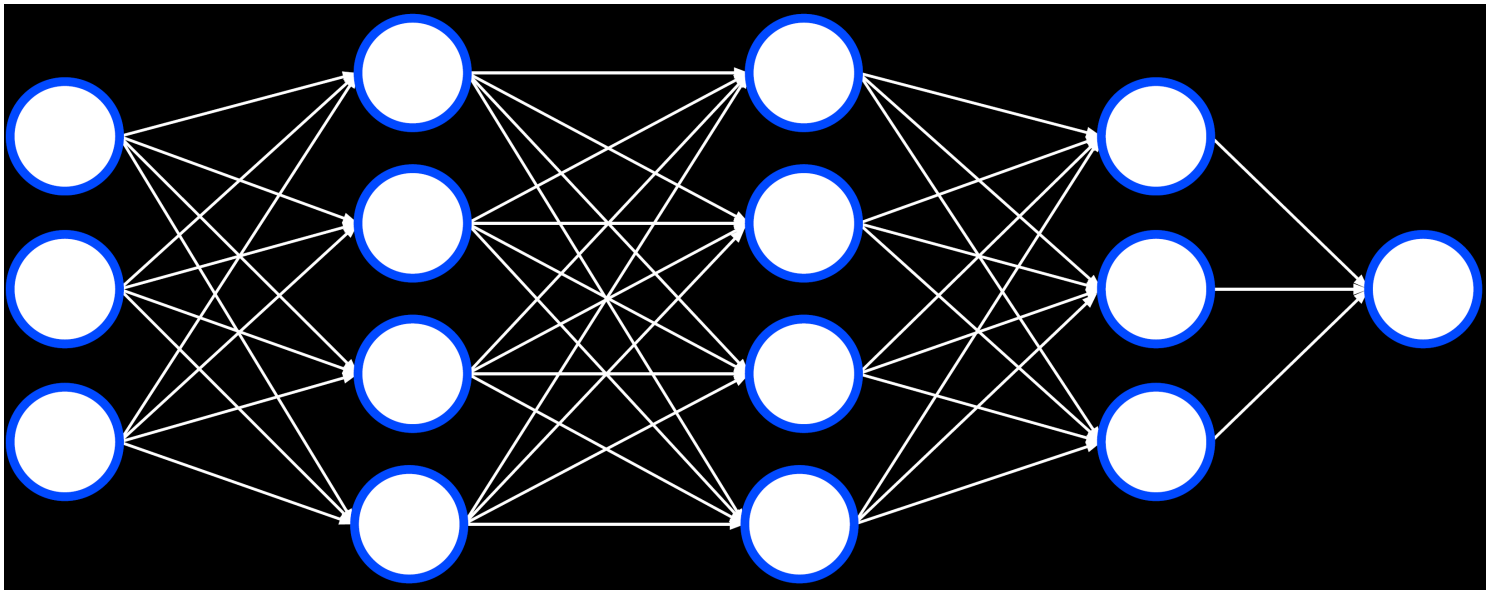
- Convolution d'une image
- Max-pooling d'une image
- Réseau de neurones à convolution (CNN)

- (Optionnel) Opérations bas niveau sur les images
  - détection de contour
  - calcul de gradients d'image
  - corrélation 2D
  - convolution 2D

RAPPELS

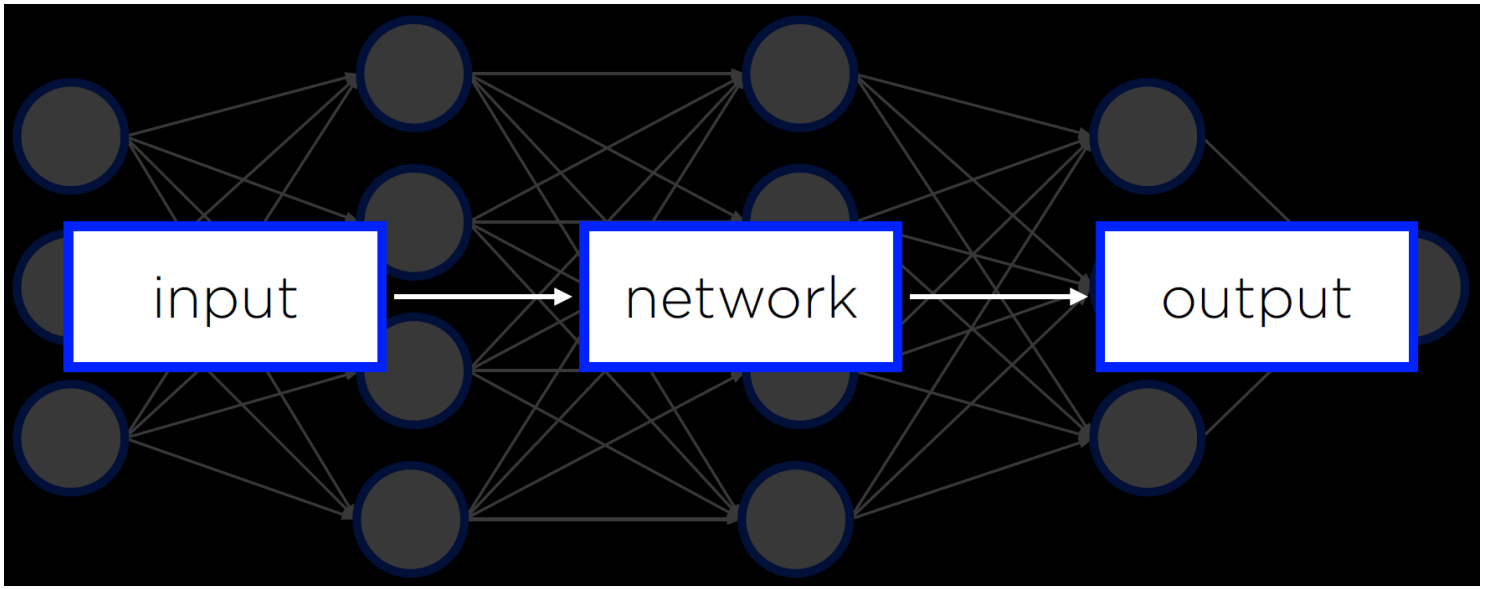
Réseau de neurone artificiel

Réseau de neuronne *feedfoward* (multi-perceptron)



Réseau de neurone artificiel

Réseau de neuronne *feedfoward* (multi-perceptron)



Concepts

- poids, fonction d'activation, modèle, inférence
- entrainement, perte, optimisation (minimisation de la perte), rétropropagation du gradient, sous-apprentissage, sur-apprentissage
- paramètres, hyperparamètres

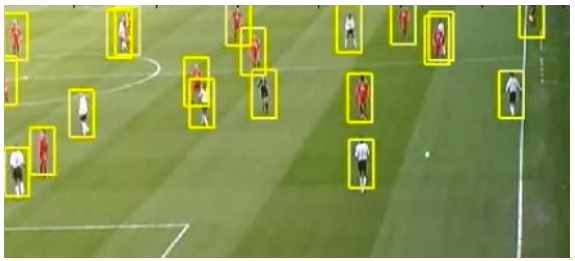
RÉSEAU DE NEURONES À CONVOLUTION

Motivation

- La vue est un sens très utile à la survie d'un organisme
  - apporte beaucoup d'information sur son environnement (nourriture, prédateur, etc.)
- Presque toutes les créatures intelligentes sont dotées de vision
- Chez l'humain ≈25% du cerveau sert à la vision
  - pour l'ouïe, c'est ≈8%
  - pour le touché, c'est ≈3%
- Ça donne une idée de la complexité de la tâche à résoudre...

Applications de la vision artificielle

- Reconnaissance de caractères (*OCR*)
- Reconnaissance d'objets (visages, plaques d'immatriculation)
- Recherche d'images
- Analyse de documents
- Détection et suivi de joueurs
- Voitures autonomes, drones, robots
- Domotique et sécurité



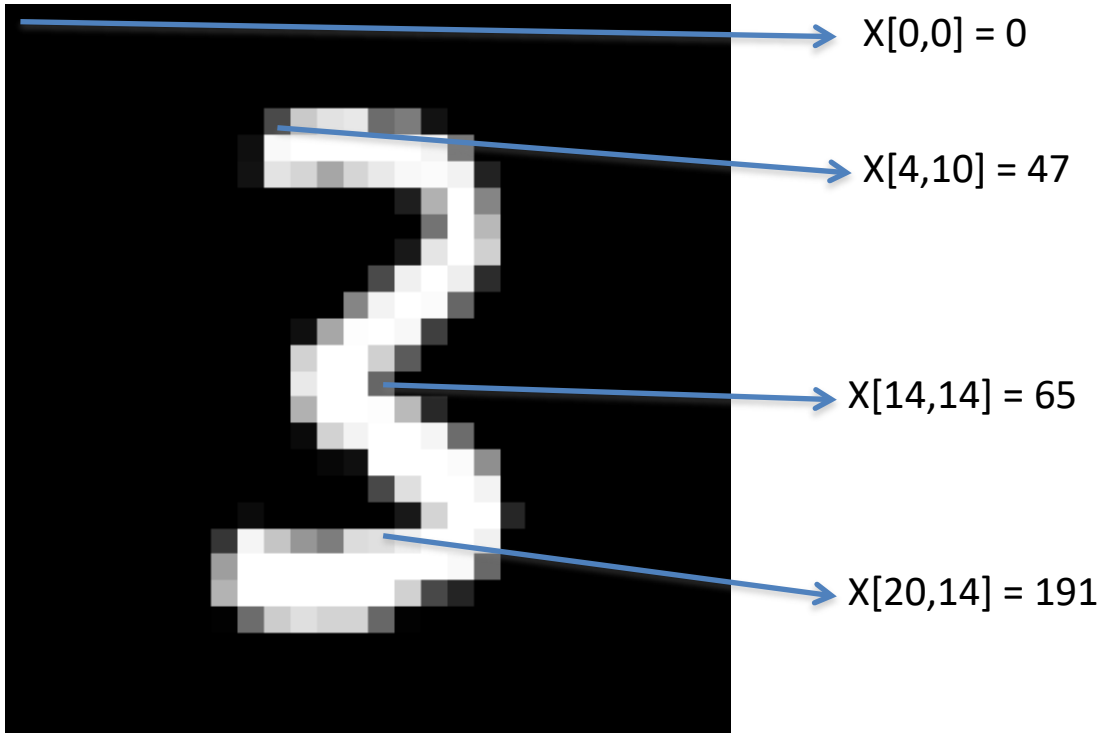
Dans cette leçon...

- On va voir deux opérations sur des images
  - convolution
  - max-pooling
- L'objectif est d'introduire le réseau de neurones à convolution
- Optionnellement, pour ceux qui souhaitent approfondir, des diapos additionnelles expliquent
  - Le gradient d'image
  - La détection de contours avec le gradient d'image
  - La correction 2D et sa relation avec la convolution 2D

Représentation brute d'une image

- Image en niveau de gris: tableau 2D de pixels, entiers positifs de 8 bits

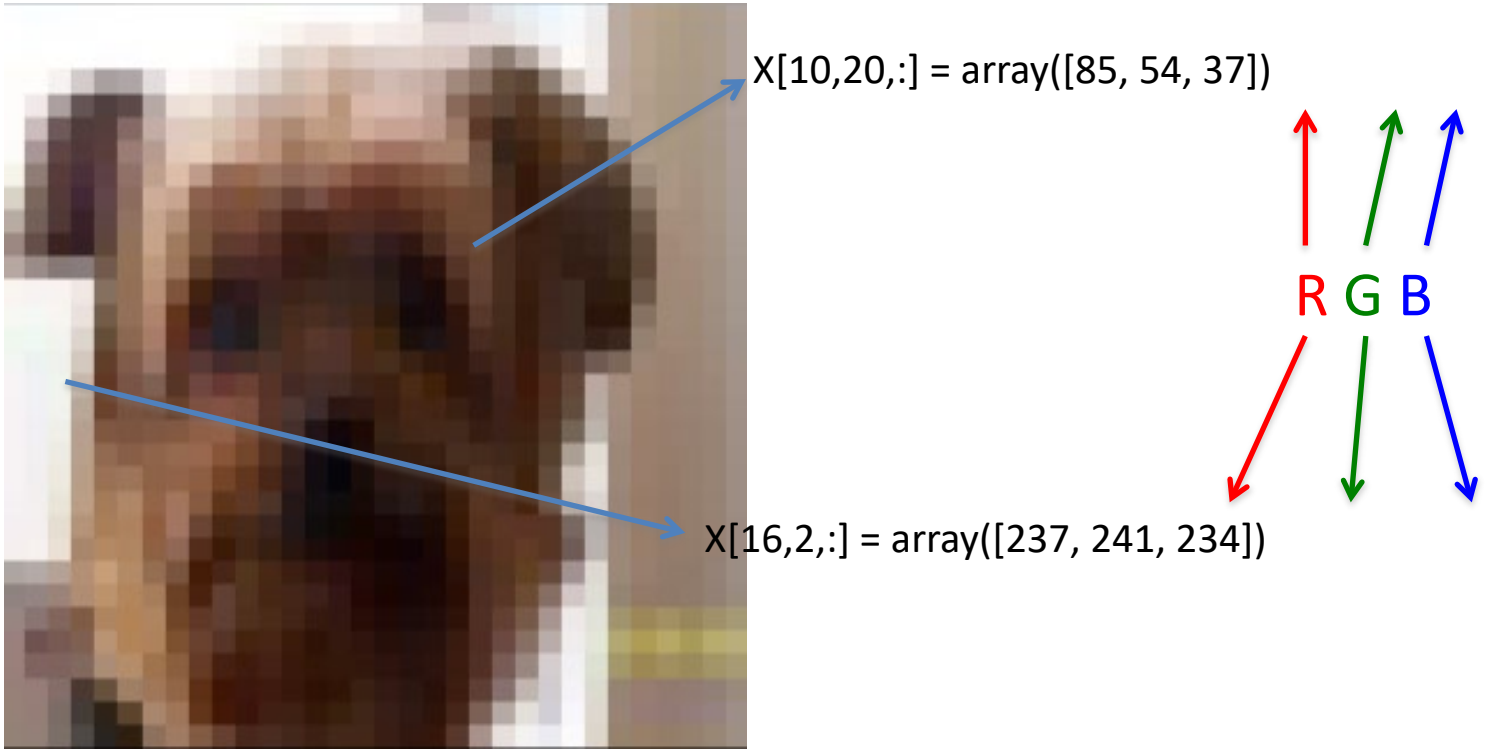
X =



Représentation brute d'une image

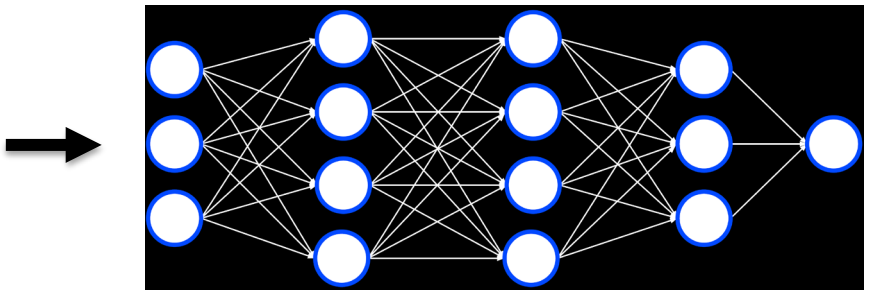
- Image en couleur: tableau 3D de pixels RGB, entiers positifs de 8 bits

X =

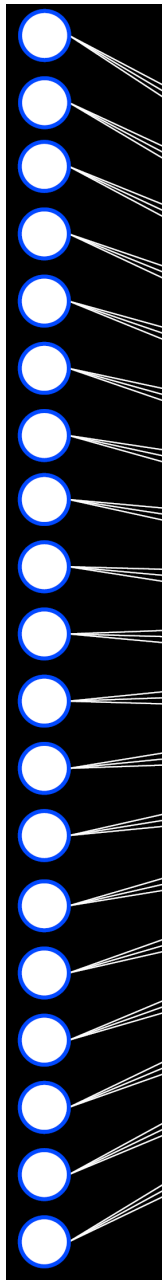


Opérations bas niveau sur les images

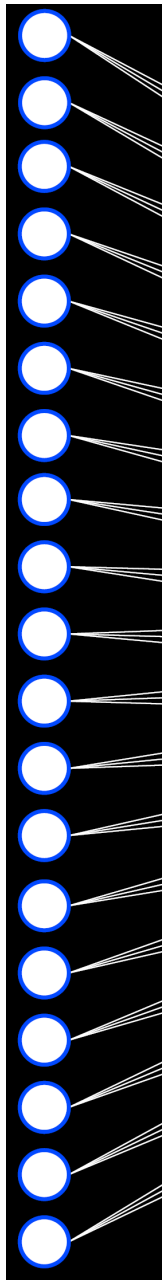
- La représentation sous forme de pixels a des désavantages
  - elle est lourde, c.-à-d. coûteuse en mémoire
    - » 1024x1024 pixels de 8 bits (en niveau de gris) = 1 MB / image
    - » 1024x1024 pixels de 24bits (canaux RGB) = 3 MB / image
  - elle contient plus d'information qu'on en a besoin
    - » pour détecter une voiture dans une image, la couleur n'est pas utile
    - » la scène (arrière plan) dans laquelle se trouve un objet à détecter peut être ignorée
- On aimerait appliquer des **opérations bas niveau simples** (**prétraitement**) sur les images, afin d'y **extraire l'information pertinente (c.-à-d., caractéristiques** ou ***features*** en anglais**)** pour la tâche à résoudre



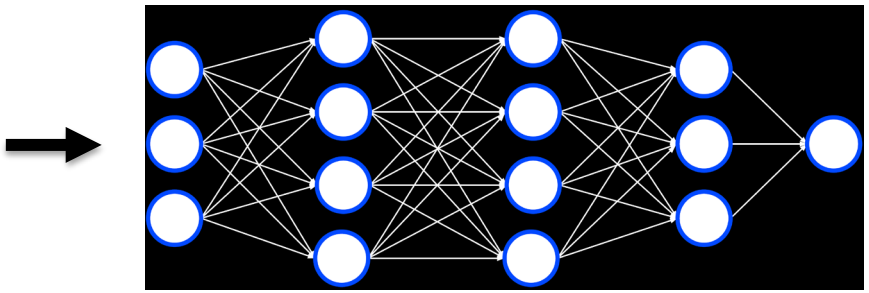
Credit: Harvard CS 50 AI course



Credit: Harvard CS 50 AI course



Credit: Harvard CS 50 AI course

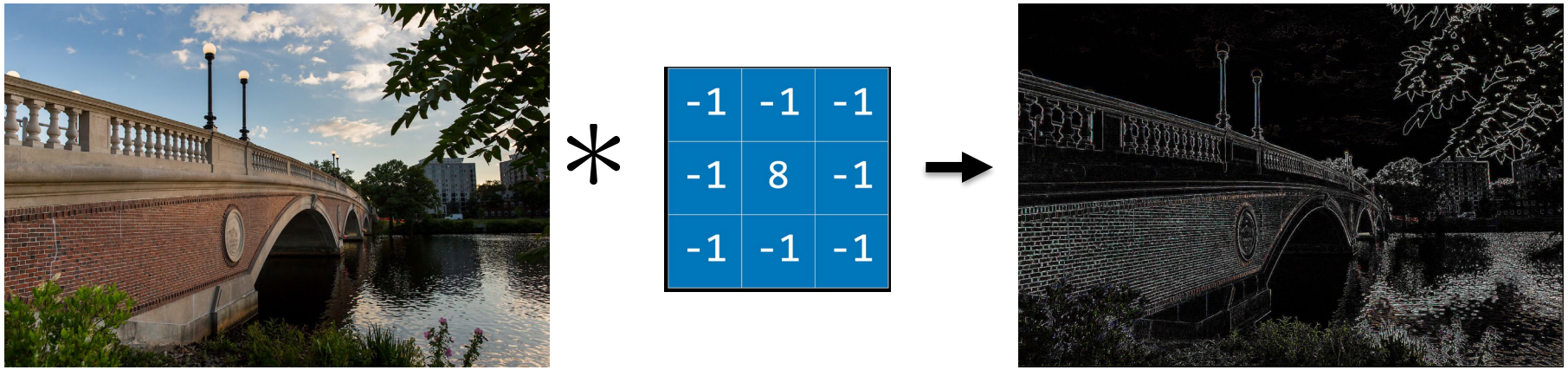


Credit: Harvard CS 50 AI course

Convolution d'une image

Opération d'appliquer un **filtre** à une image. Elle consiste à additioner chaque pixel de l'image avec les pixels avoisinants pondérés par les valeurs du filtre.

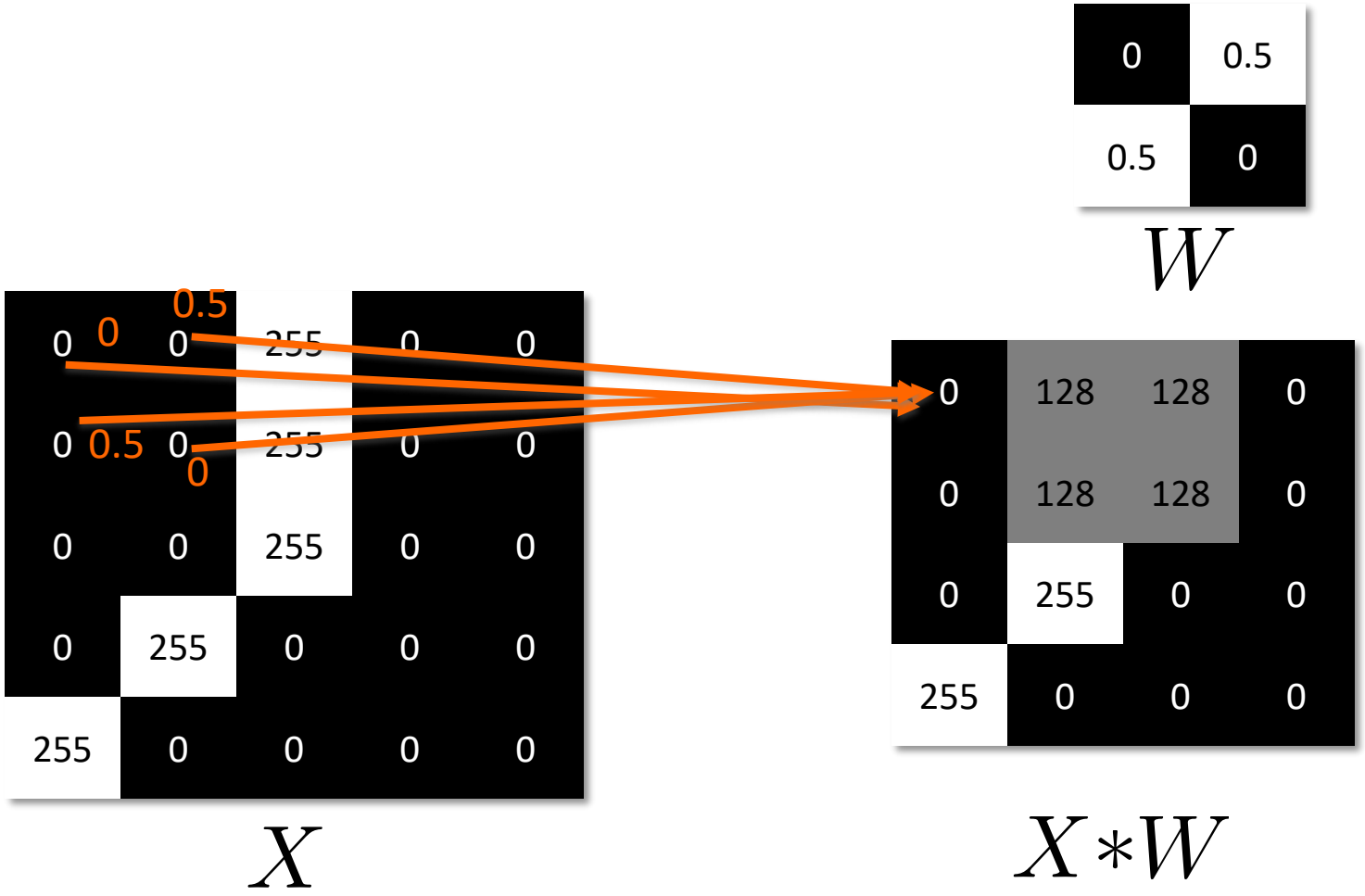
Illustration



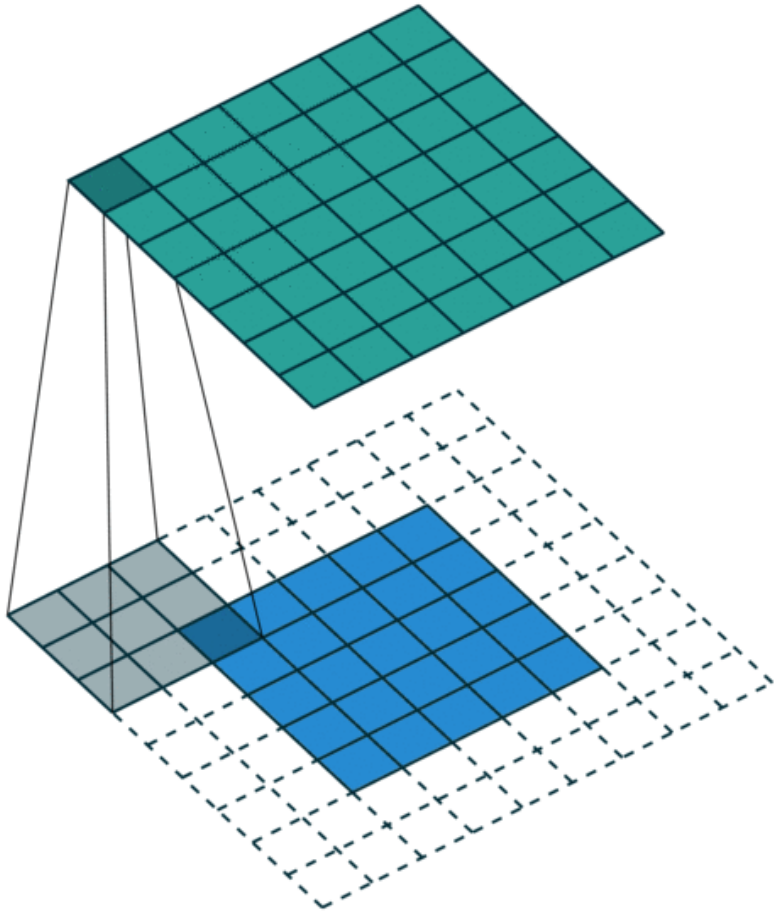
Credit: Harvard CS 50 AI course

Le filtre est aussi appelé un **noyau**.

Exemple 1

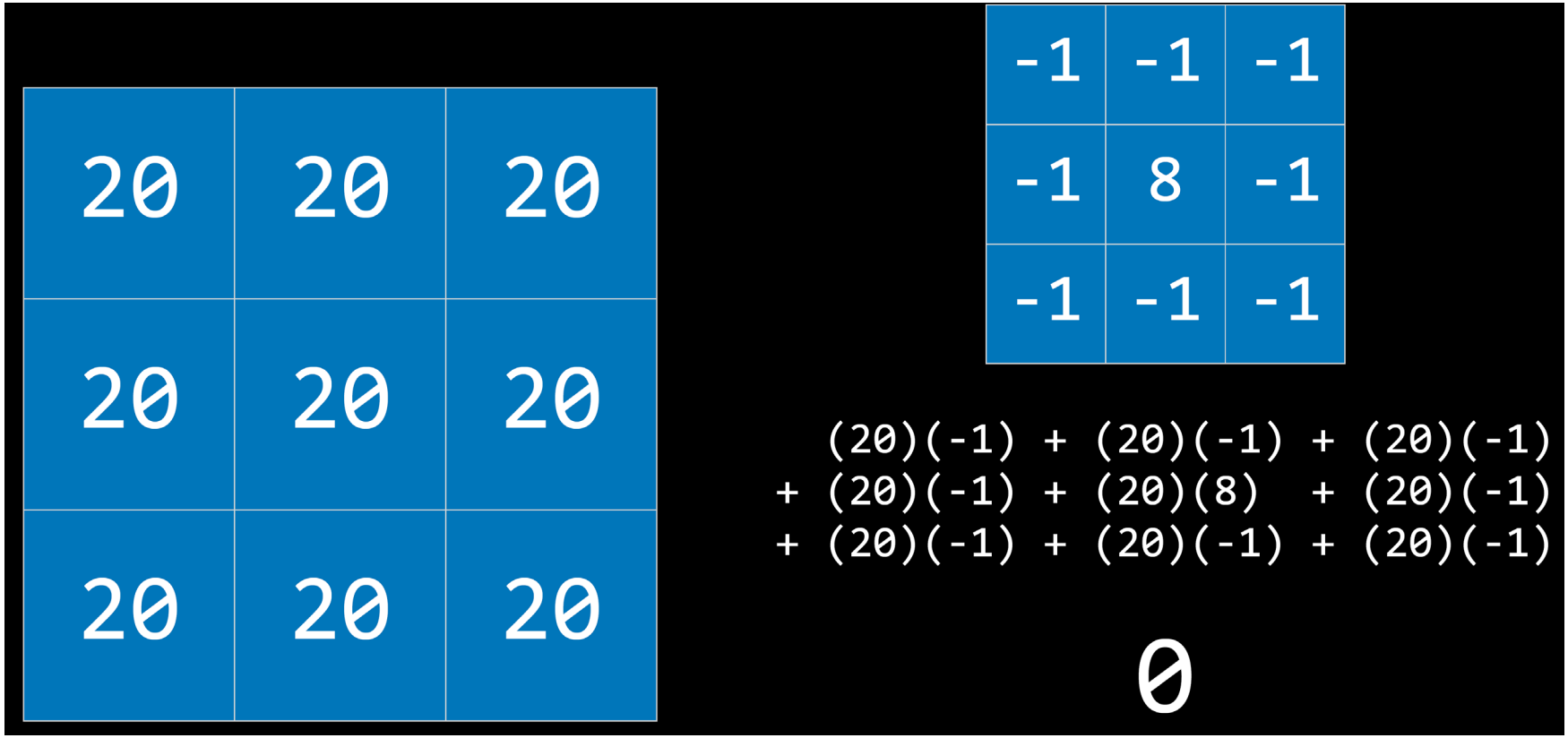

0
0.5
0.5
0
W
0.5
0
0
0.5
0
0
0
255
0
0
0
0
255
0
0
0
0
255
0
0
0
255
0
0
0
255
0
0
0
X
0
128
128
0
0
128
128
0
0
255
0
0
255
0
0
0
$X*W$

Illustration

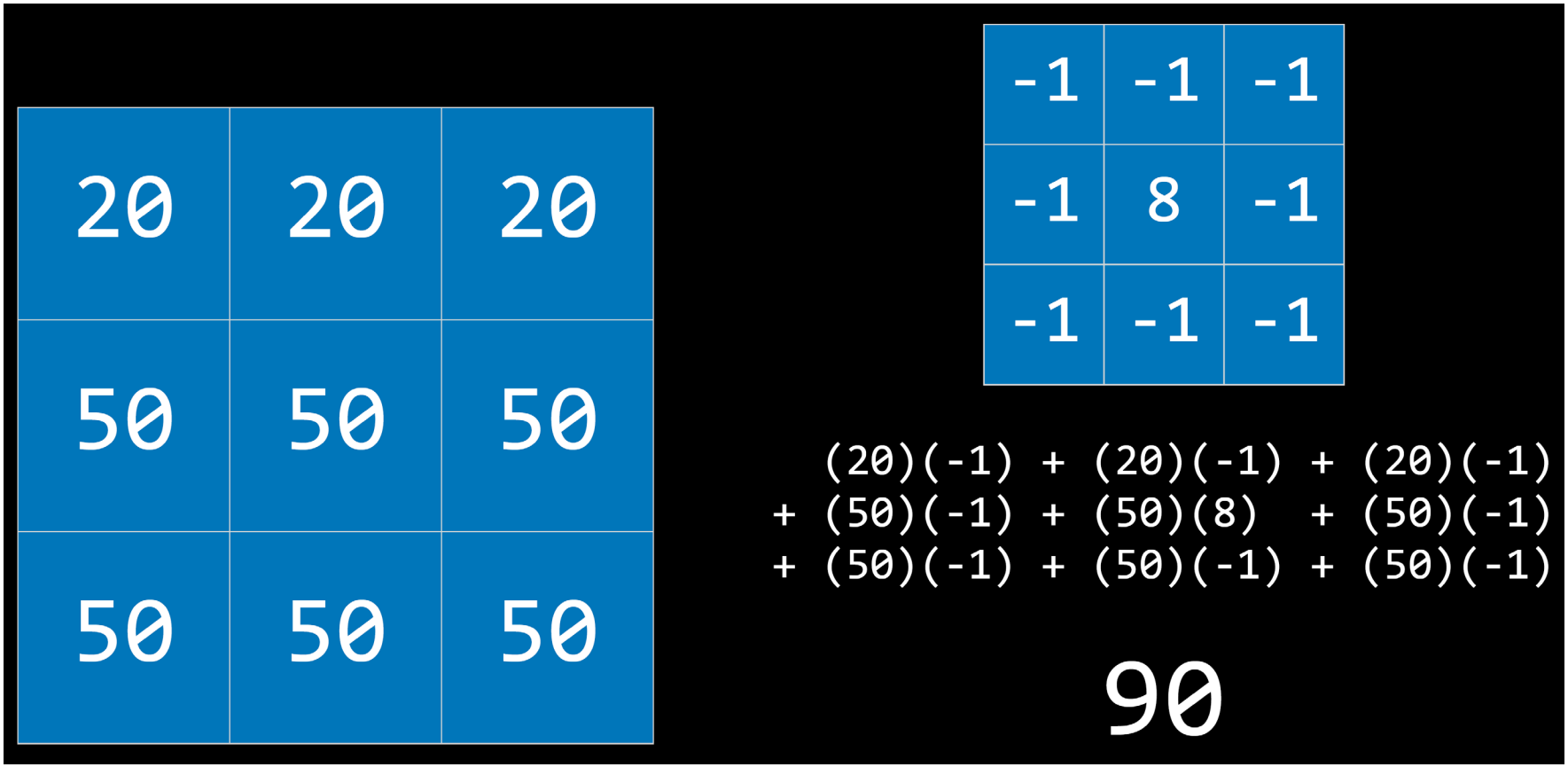


Crdeit: https://towardsdatascience.com/convolution-vs-correlation-af868b6b4fb5

Exemple 2


20 20 20
20 20 20
20 20 20
-1 -1 -1
-1 8 -1
-1 -1 -1
(20)(-1) + (20)(-1) + (20)(-1)
+ (20)(-1) + (20)(8) + (20)(-1)
+ (20)(-1) + (20)(-1) + (20)(-1)
0

Exemple 2


20 20 20
50 50 50
50 50 50
-1 -1 -1
-1 8 -1
-1 -1 -1
(20)(-1) + (20)(-1) + (20)(-1)
+ (50)(-1) + (50)(8) + (50)(-1)
+ (50)(-1) + (50)(-1) + (50)(-1)
90

$*$

-1	-1	-1
-1	8	-1
-1	-1	-1

Credit: Harvard CS 50 AI course

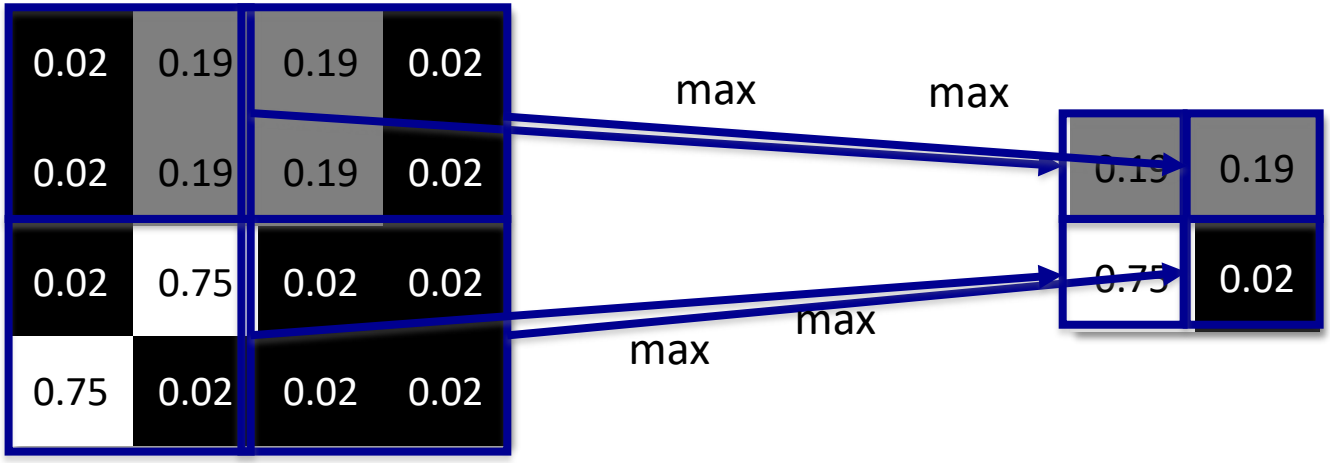
Pooling d'une image

Opération de **réduire la taille de l'image** remplaçant des régions de l'image par leurs échantillons

Max-pooling d'une image

Opération de **réduire la taille de l'image** remplaçant chaque région de l'image par le maximum de la région

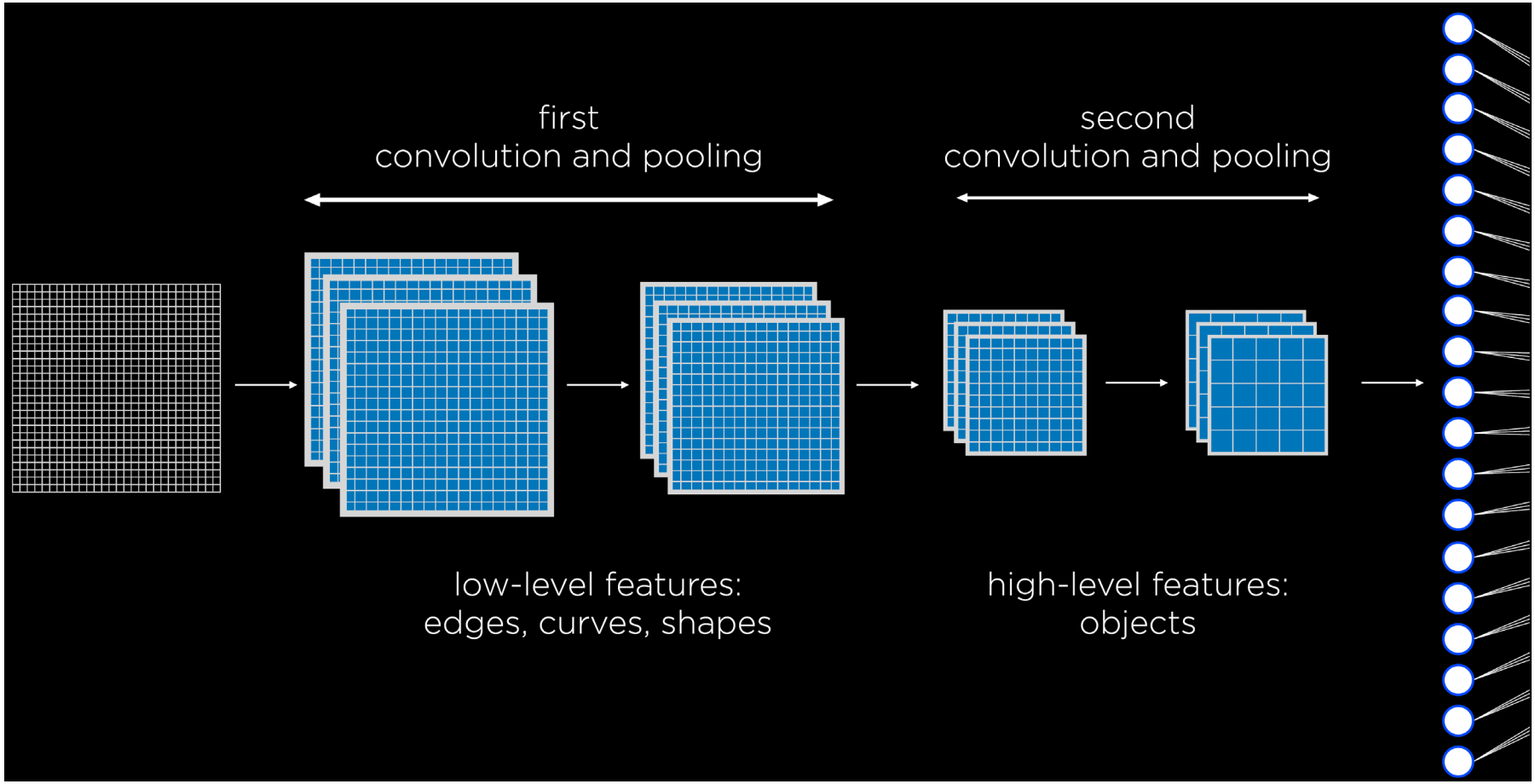
Exemple


0.02
0.19
0.19
0.02
0.02
0.19
0.19
0.02
0.02
0.75
0.02
0.02
0.75
0.02
0.02
0.02
max
max
max
max
0.19
0.19
0.75
0.02

Réseau de neurones à convolution

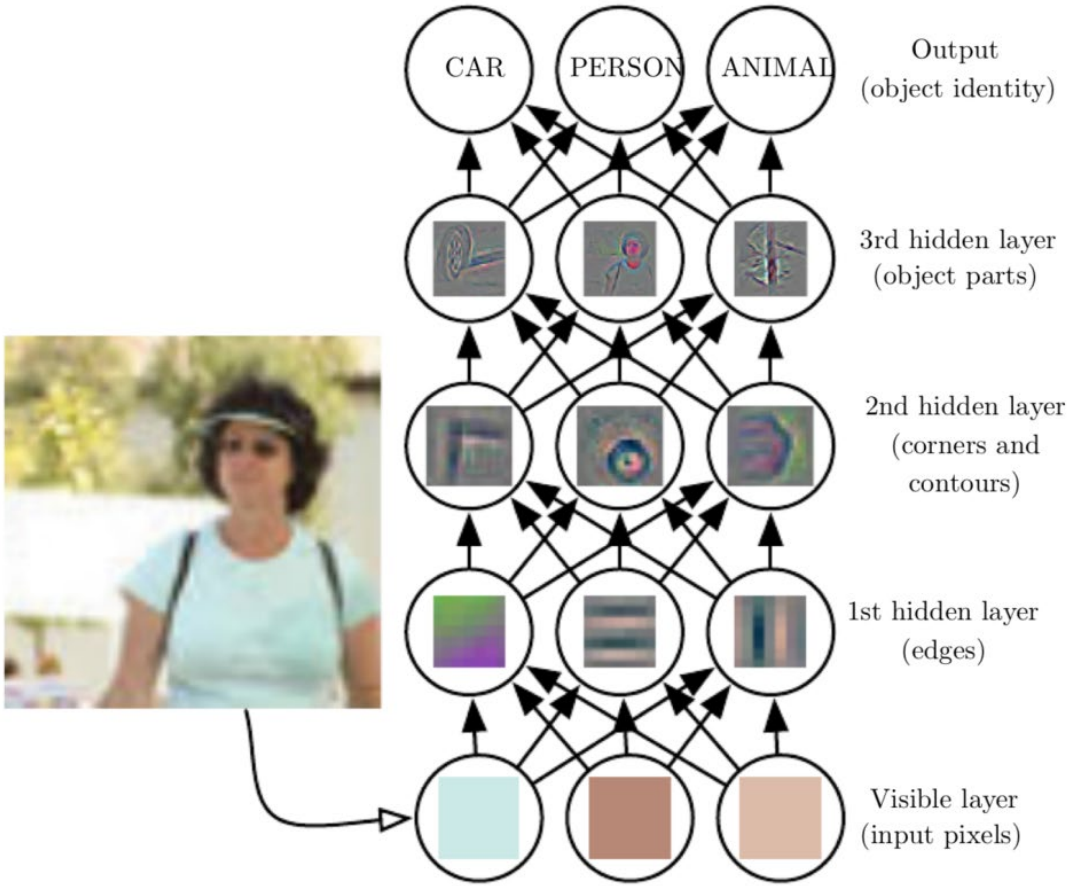
Réseau de neurones qui utilise la convolution – le plus souvent appliqué pour l'analyse d'images

Réseau de neurone à convolution



Credit: Harvard CS 50 AI course

Chaque couche apprend une abstraction



https://bdtechtalks.com/2020/01/06/convolutional-neural-networks-cnn-convnets/

Conclusion

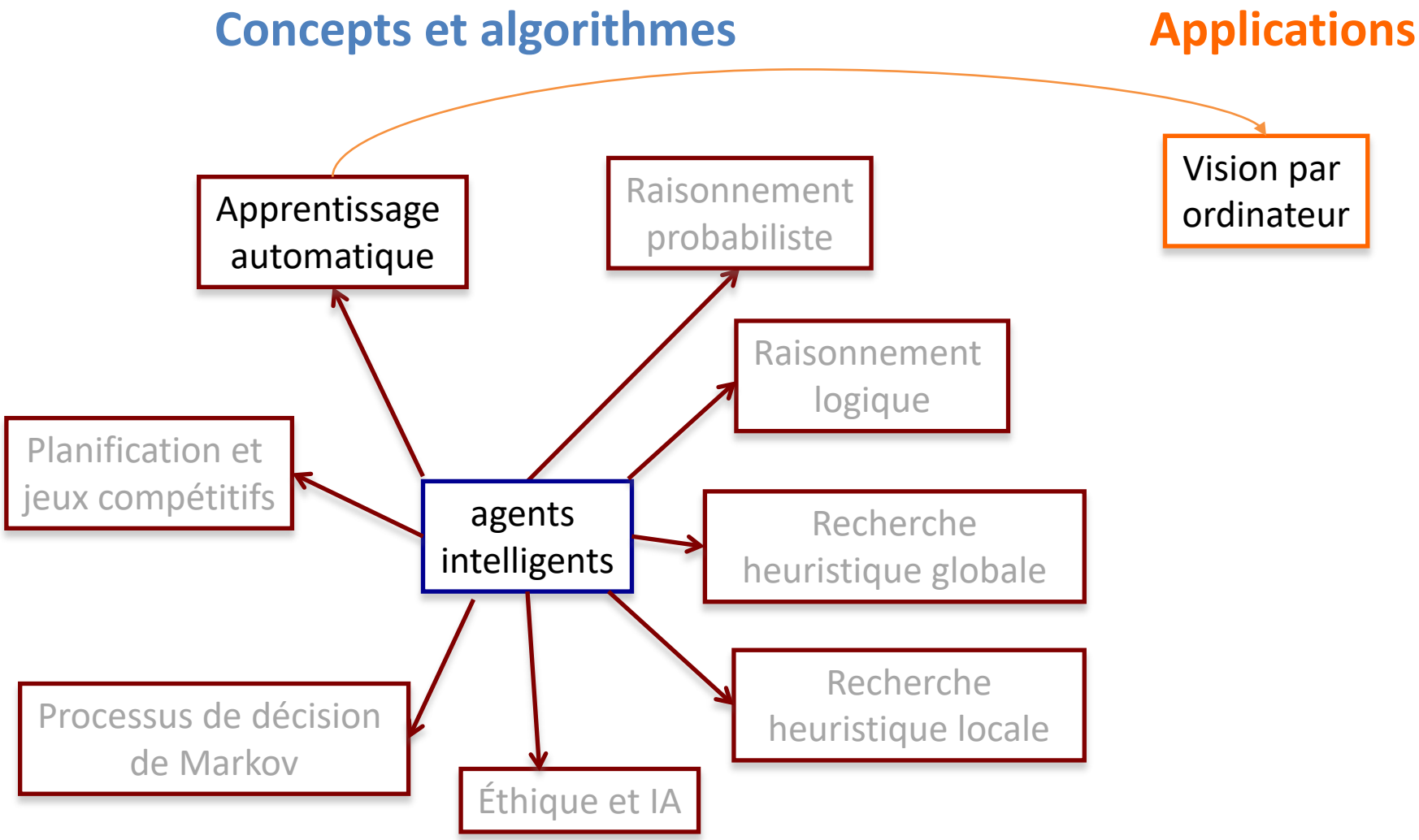
- La vision artificielle est un des problèmes fondamentaux de l'IA
- L'architecture CNN est une des architectures fondamentales en apprentissage automatique, avec plusieurs applications, pas seulement en vision
- La plus part des plateformes d'apprentissage automatique supportent les CNN

Conclusion

- Ce cours a introduit réseau de neurone de façon simple et pratique

- Cours plus avancés:
  - Le bacc en imagerie offre plusieurs cours sur le sujet (ex.: **IMN 559 - Vision par ordinateur**)
    - » ces cours peuvent être suivis à la maîtrise...
  - **IFT 603 – Techniques d'apprentissage**
  - **IFT 725 – Réseaux neuronaux** : couvre les réseaux à convolution avec plus de détails (cours de maîtrise)

Sujets couverts



Vous devriez être capable de...

- Décrire ce qu'une convolution
- Décrire ce qu'un max-pooling
- Expliquer ce qui distingue un réseau de neurones à convolution d'un réseau de neurones standard (perceptron multi-couches)

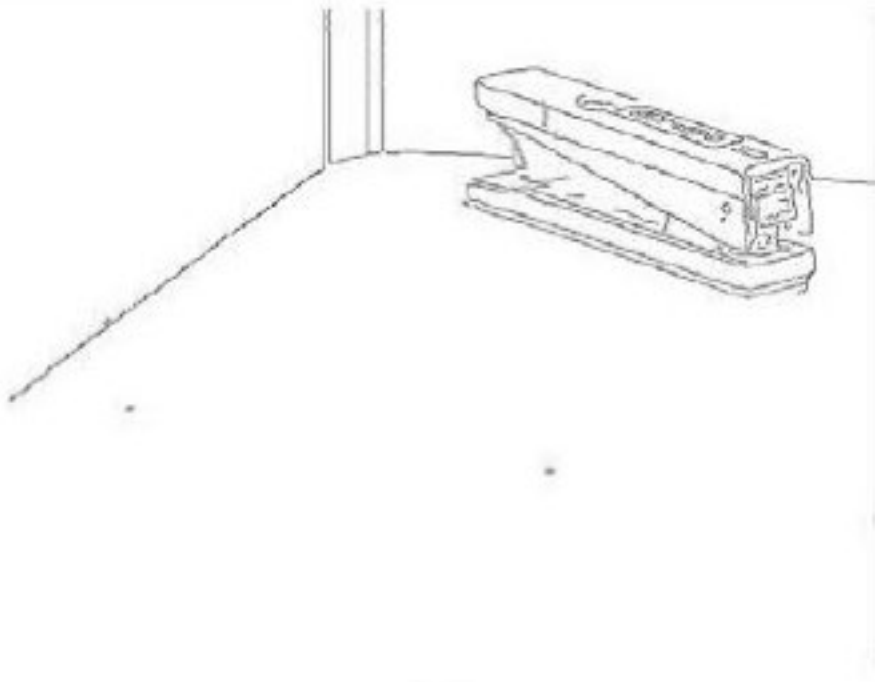
Optionnel (pas couvert pour l'examen)

APPROFONDIR LA CONVOLUTION

Contour

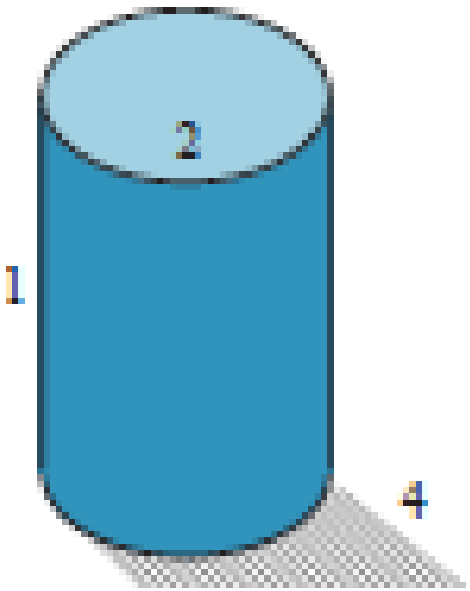
- Un contour est un changement soudain dans l'intensité/couleur de pixels adjacents

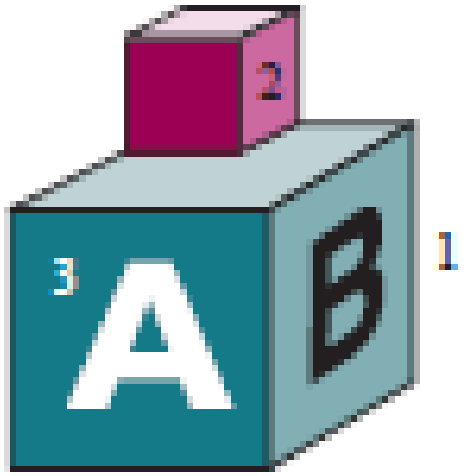
image originale

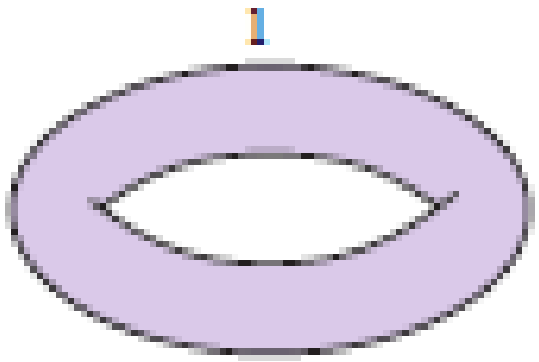


extraction des contours

Types de contours d'images





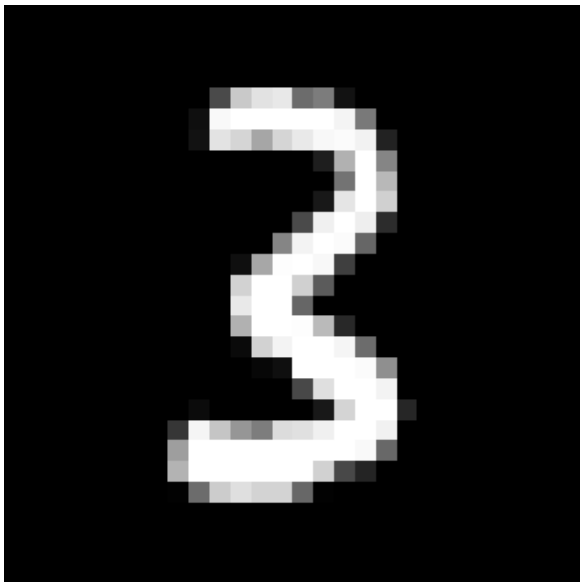


1- changement de profondeur
2- changement d'orientation de surface
3- changement de couleur (réflexion)
4- changement d'illumination

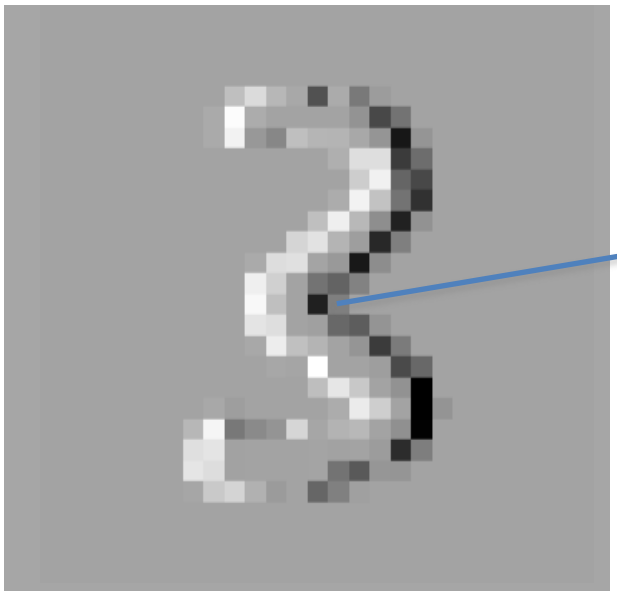
Gradient d'image

- Pour détecter si un pixel est sur la frontière d'un contour, on peut regarder la valeur relative des pixels autour de ce pixel

- Exemple: variation **horizontale** H[i,j] = X[i,j+1] – X[i,j]



X



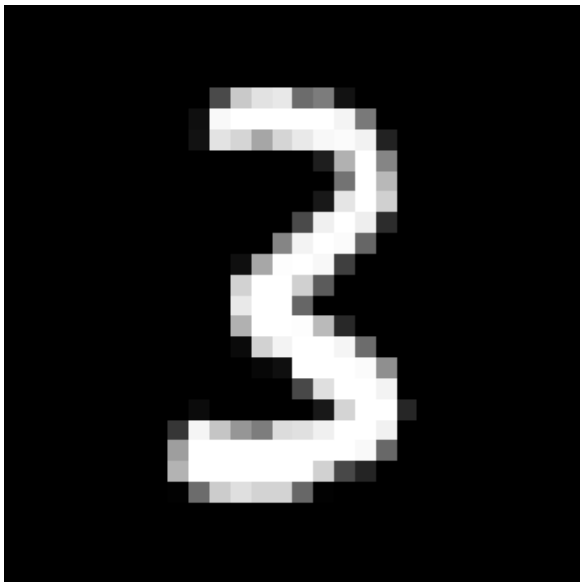
H

H[14,14] = X[14,15] – X[14,14]

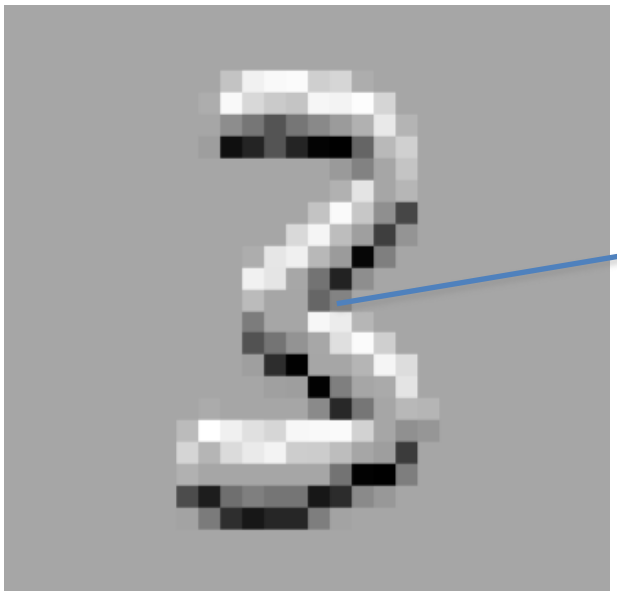
Gradient d'image

- Pour détecter si un pixel est sur la frontière d'un contour, on peut regarder la valeur relative des pixels autour de ce pixel

- Exemple: variation **verticale** V[i,j] = X[i+1,j] – X[i,j]



X



V

V[14,14] = X[15,14] – X[14,14]

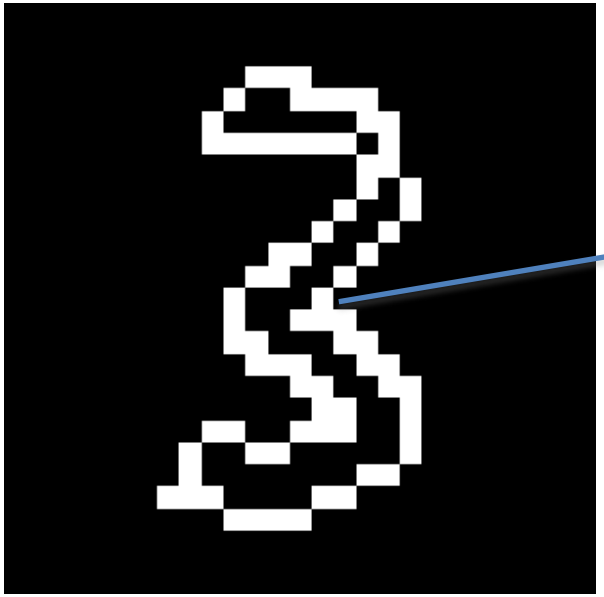
Détecter un contours à partir des gradients d'image

- Un pixel ferait partie d'un contour si la somme des variations (positive ou négative) horizontale et verticale est élevée

$$E[i,j] = \text{sqrt}(V[i,j]^{**}2 + H[i,j]^{**}2)$$

- On applique un seuil pour déterminer si contour ou pas

X



E > 128

Gradient d'image

- On peut voir le calcul des variations comme des dérivées partielles
- La « fonction » $f(a,b)$ serait la valeur de l'image à la position (a,b)

$$\frac{\partial f(a,b)}{\partial b} = \lim_{\Delta \to 0} \frac{f(a,b+\Delta) - f(a,b)}{\Delta} \approx \underbrace{X[i,j+1] - X[i,j] = H[i,j]}_{\Delta = 1}$$

$$\frac{\partial f(a,b)}{\partial a} = \lim_{\Delta \to 0} \frac{f(a+\Delta,b) - f(a,b)}{\Delta} \approx \overbrace{X[i+1,j] - X[i,j] = V[i,j]}^{\Delta = 1}$$

Gradient d'image

- Si H[i,j] et V[i,j] sont les dérivées partielles de l'image, alors

$$G[i,j,:] = [\, H[i,j], V[i,j] \,]$$

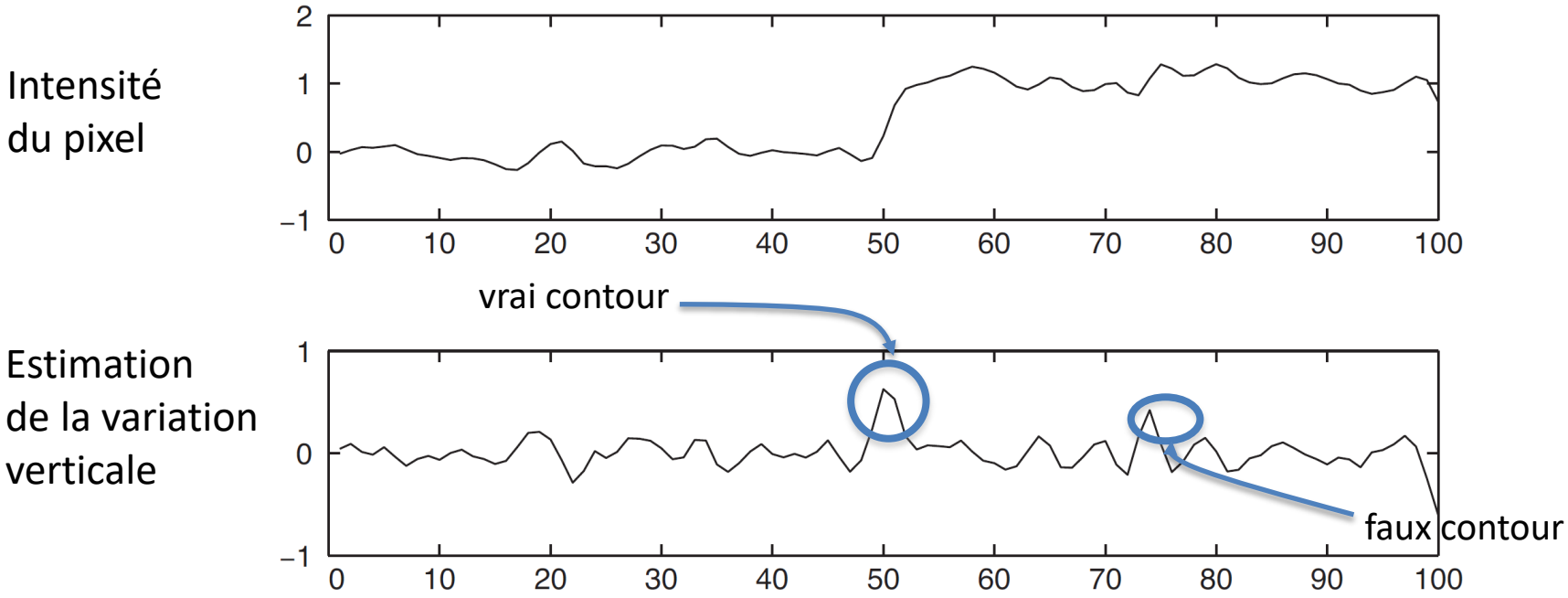
est le **gradient de l'image**, à la position (i,j)

- Pour détecter contours, l'idée serait de calculer donc la norme euclidienne de ces gradients et voir où les gradients changent significativement

$$E[i,j] = \mathrm{sqrt}(V[i,j]^2 + H[i,j]^2) = \underbrace{\mathrm{sqrt}(\mathrm{sum}(G[i,j,:]^2))}_{\text{norme du vecteur } G[i,j,:]}$$

Exemple de calcul du gradient bruité

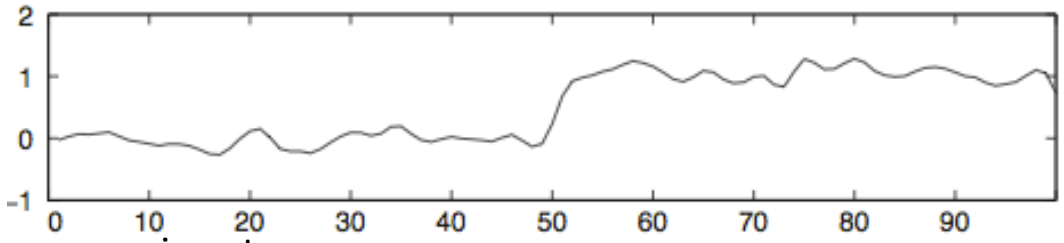
- Des vraies images sont bruitées et donc les variations des gradients vont l'être aussi



- Pour éliminer la détection de faux contours, on peut lisser l'image

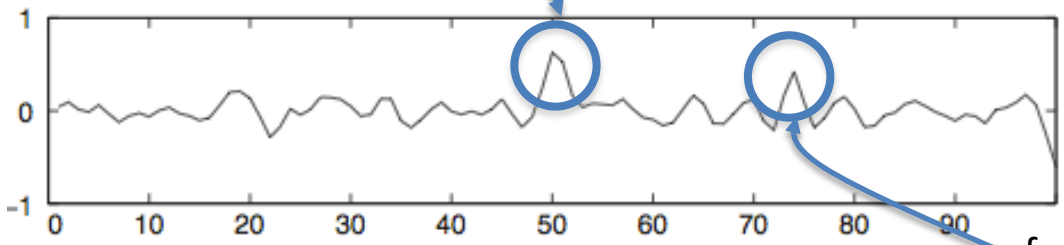
Calcul gradient d'image lissé

Intensité du pixel



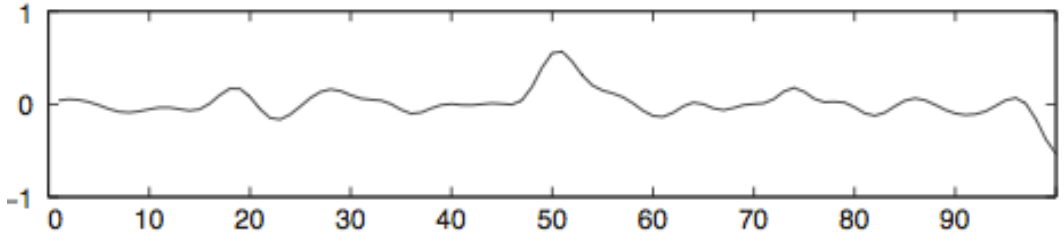
vrai contour

Estimation de la variation verticale



faux contour

Estimation de la variation Verticale d'une version lissée de l'intensité



Lissage Gaussien d'une image

- Pour éliminer la détection de ces faux contours, on lisse l'image en appliquant un filtre Gaussien à l'image.

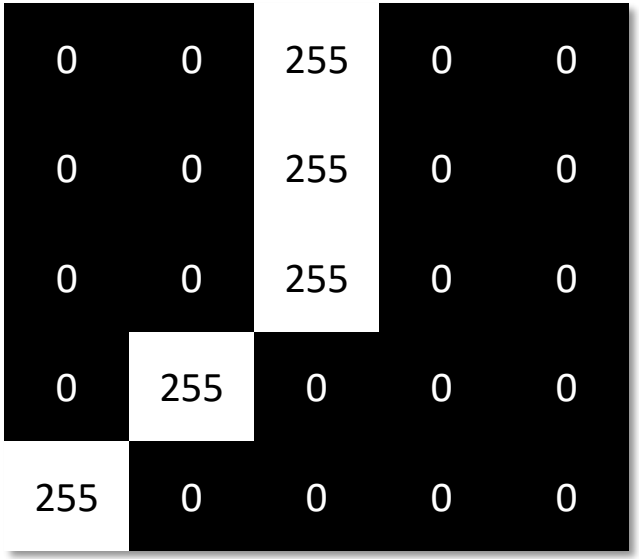
- Soit $G_\sigma(x) = \frac{1}{\sqrt{2\pi}\sigma} e^{-x^2/2\sigma^2}$ Formule générale du filtre gaussien

- Remplacer l'intensité $X[i_0,j_0]$ du pixel (i_0,j_0) par $U(i_0,j_0)$ défini comme suit:

$C(i_0,j_0)$ = somme, sur tous les (i,j), de $X[i,j]G_\sigma(d)$,
d étant la distance de (i_0,j_0) à (i,j)
σ est un hyper-paramètre

$C(i_0, j_0) = \Sigma_i \Sigma_j X[i,j]G_\sigma(d)$

Lissage Gaussien d'une image



X

U(0,0)	U(0,1)	U(0,2)	U(0,3)	U(0,4)
U(1,0)	U(1,1)	U(1,2)	U(1,3)	U(1,4)
U(2,0)	U(2,1)	U(2,2)	U(2,3)	U(2,4)
U(3,0)	U(3,1)	U(3,2)	U(3,3)	U(3,4)
U(4,0)	U(4,1)	U(2,2)	U(4,3)	U(4,4)

U

$U(i_0, j_0) = \Sigma_i \Sigma_j X[i,j] G_\sigma(d)$,

d étant la distance de (i_0, j_0) à (i,j).

$$G_\sigma(x) = \frac{1}{\sqrt{2\pi}\sigma} e^{-x^2/2\sigma^2}$$

Lissage Gaussien d'une image

$U(i_0, j_0) = \Sigma_i \Sigma_j X[i,j] G_\sigma(d)$,

d étant la distance de (i_0, j_0) à (i,j).

$$G_\sigma(x) = \frac{1}{\sqrt{2\pi}\sigma} e^{-x^2/2\sigma^2}$$

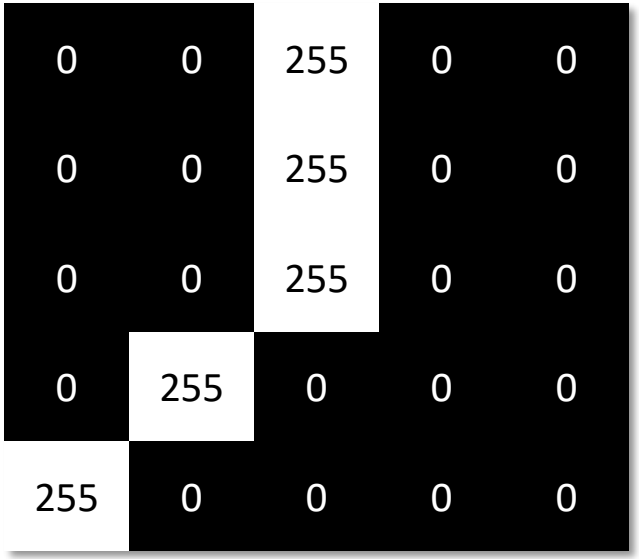
- On dit que U est la **convolution** de G_σ *et* X

- De façon générale u est la **convolution** de deux functions f et g ($u = f \bigstar g$), si

$$C(x,y) = \sum_{u=-\infty}^{+\infty} \sum_{v=-\infty}^{+\infty} f(u,v)\, g(x-u, y-v)$$

- Vu que l'influence d'une gaussienne s'atténue rapidement avec la distance, en général on remplace +- ∞ par +- 3σ

Lissage Gaussien d'une image



X

U(0,0)	U(0,1)	U(0,2)	U(0,3)	U(0,4)
U(1,0)	U(1,1)	U(1,2)	U(1,3)	U(1,4)
U(2,0)	U(2,1)	U(2,2)	U(2,3)	U(2,4)
U(3,0)	U(3,1)	U(3,2)	U(3,3)	U(3,4)
U(4,0)	U(4,1)	U(2,2)	U(4,3)	U(4,4)

C

$C(i_0, j_0) = \Sigma_i \Sigma_j X[i,j] G_\sigma(d)$,

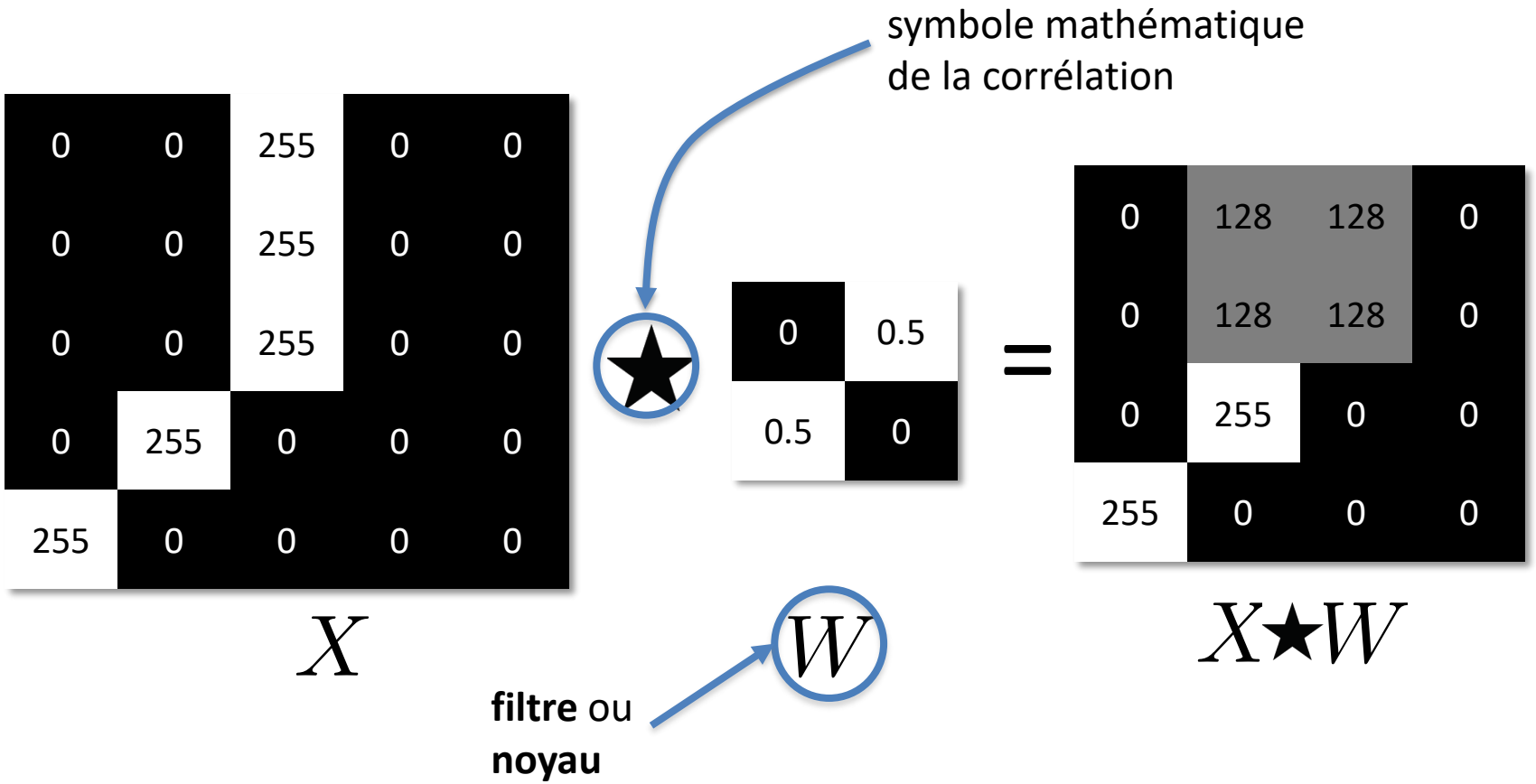
d étant la distance de (i_0, j_0) à (i,j).

On somme seulement sur (i,j) dans un voisinage inférieur à 3σ

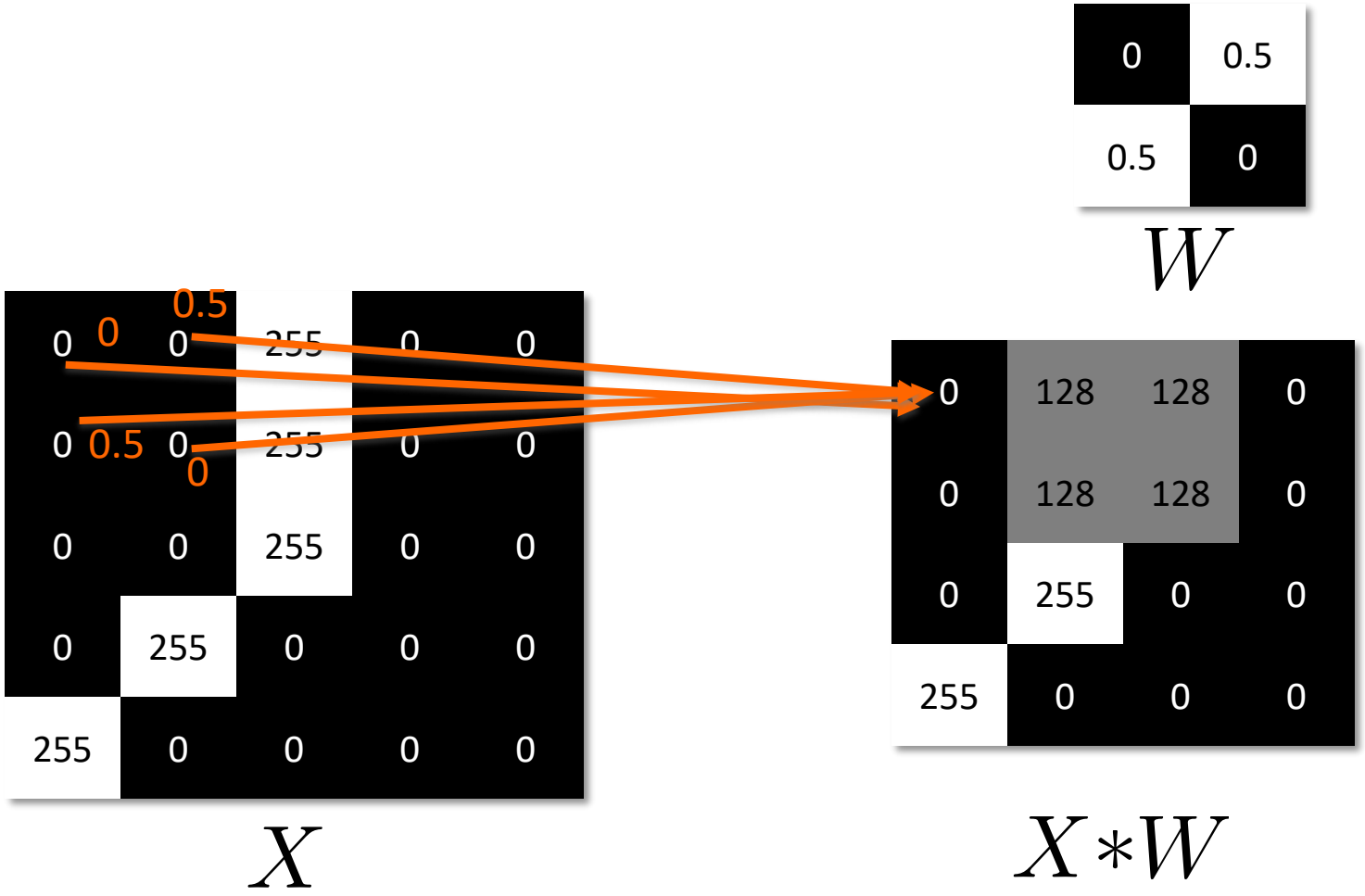
$$G_\sigma(x) = \frac{1}{\sqrt{2\pi}\sigma} e^{-x^2/2\sigma^2}$$

Corrélation 2D

Appliquer un filtre gaussien dans un voisinage limité peut être vu comme l'application d'une **corrélation 2D**



Correlation 2D

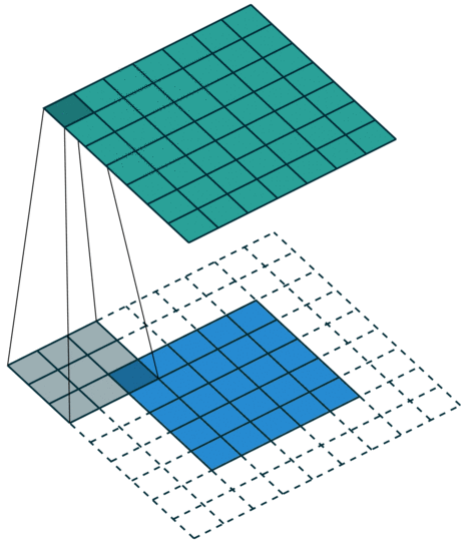


Corrélation 2D

- Le calcul des tableaux H et V peut être vu comme l'application d'une **corrélation 2D**

- On calcule le résultat C d'une convolution d'un **filtre** ou **noyau** W de taille h par w sur une image X comme suit

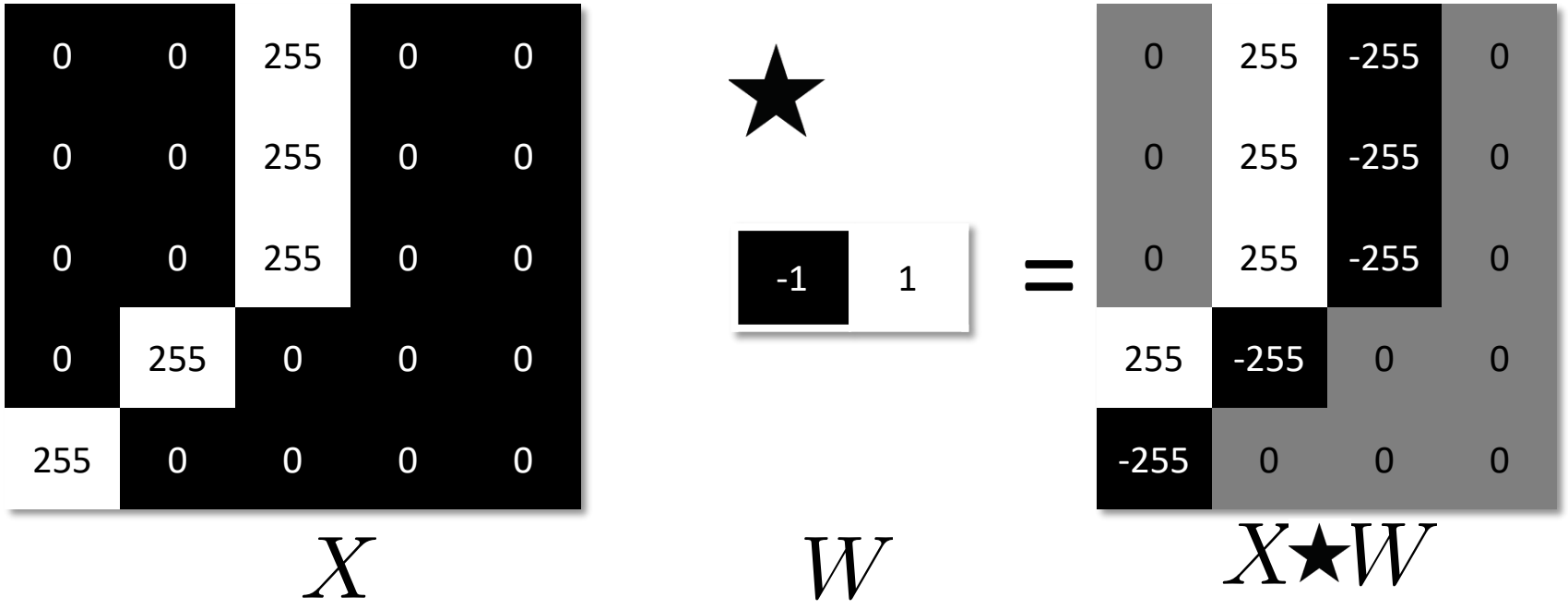
```
def correlation (X,W):
  h,w = W.shape
  C = zeros((X.shape[0]-h+1,X.shape[1]-w+1))
  for i in range(X.shape[0]-h+1):
    for j in range(X.shape[1]-w+1):
      C[i,j] = sum(X[i:i+h,j:j+w] * W)
  return C
```



https://towardsdatascience.com/convolution-vs-correlation-af868b6b4fb5

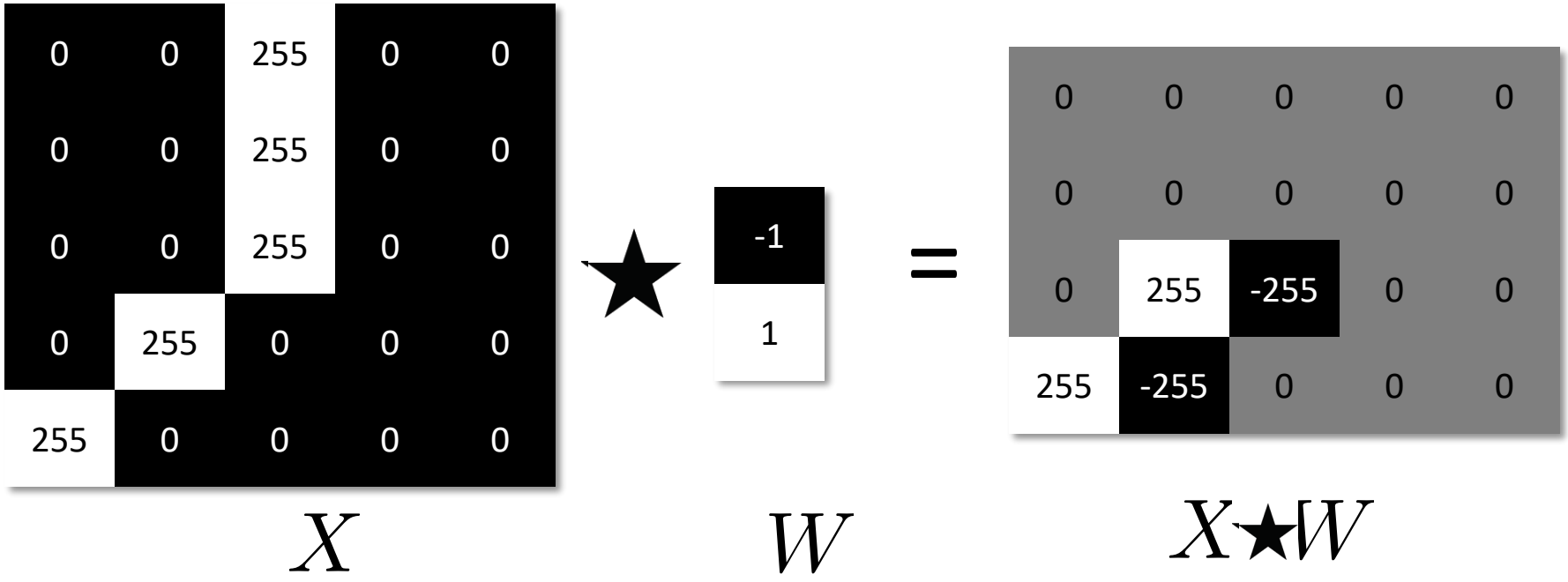
Corrélation 2D

- Calculer H est l'équivalent de faire une corrélation avec le filtre
  W = array([[-1,1]])



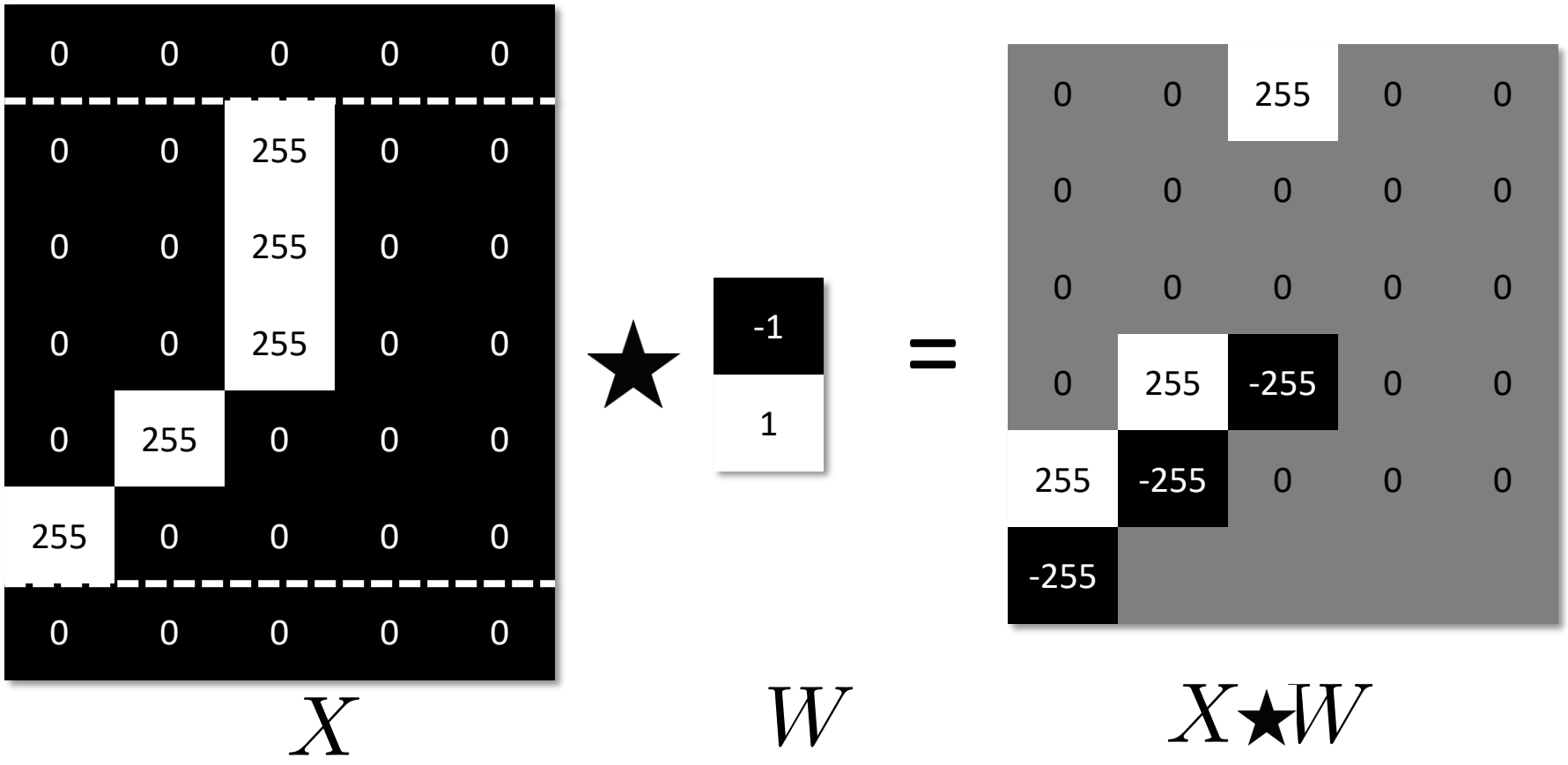
Corrélation 2D

- Calculer V est l'équivalent de faire une corrélation avec le filtre
  W = array([[-1],[1]])

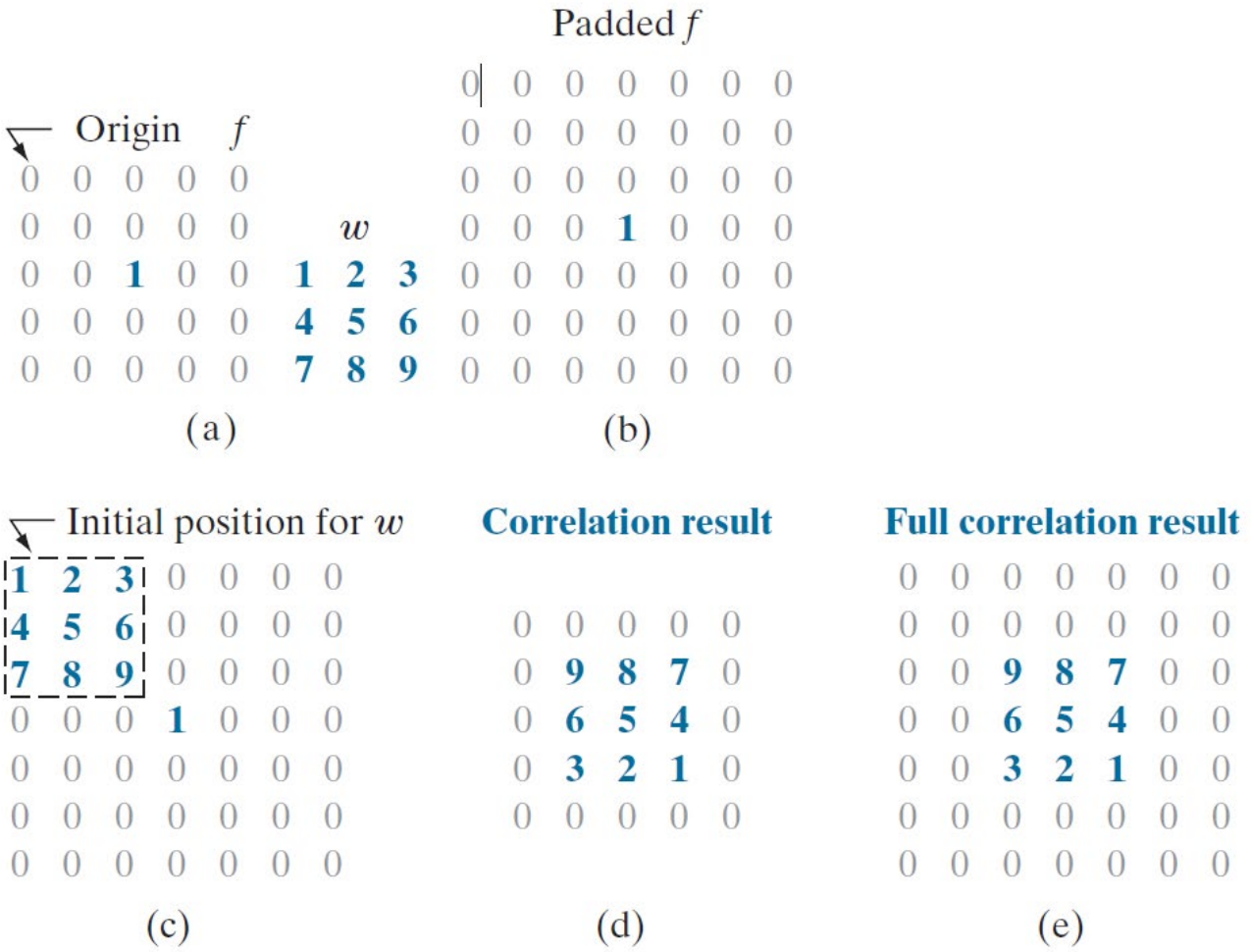


Corrélation 2D

Afin d'appliquer le filtre à toutes les positions dans l'image, on ajoute parfois les zéros nécessaires autour de l'image (*zero padding*)



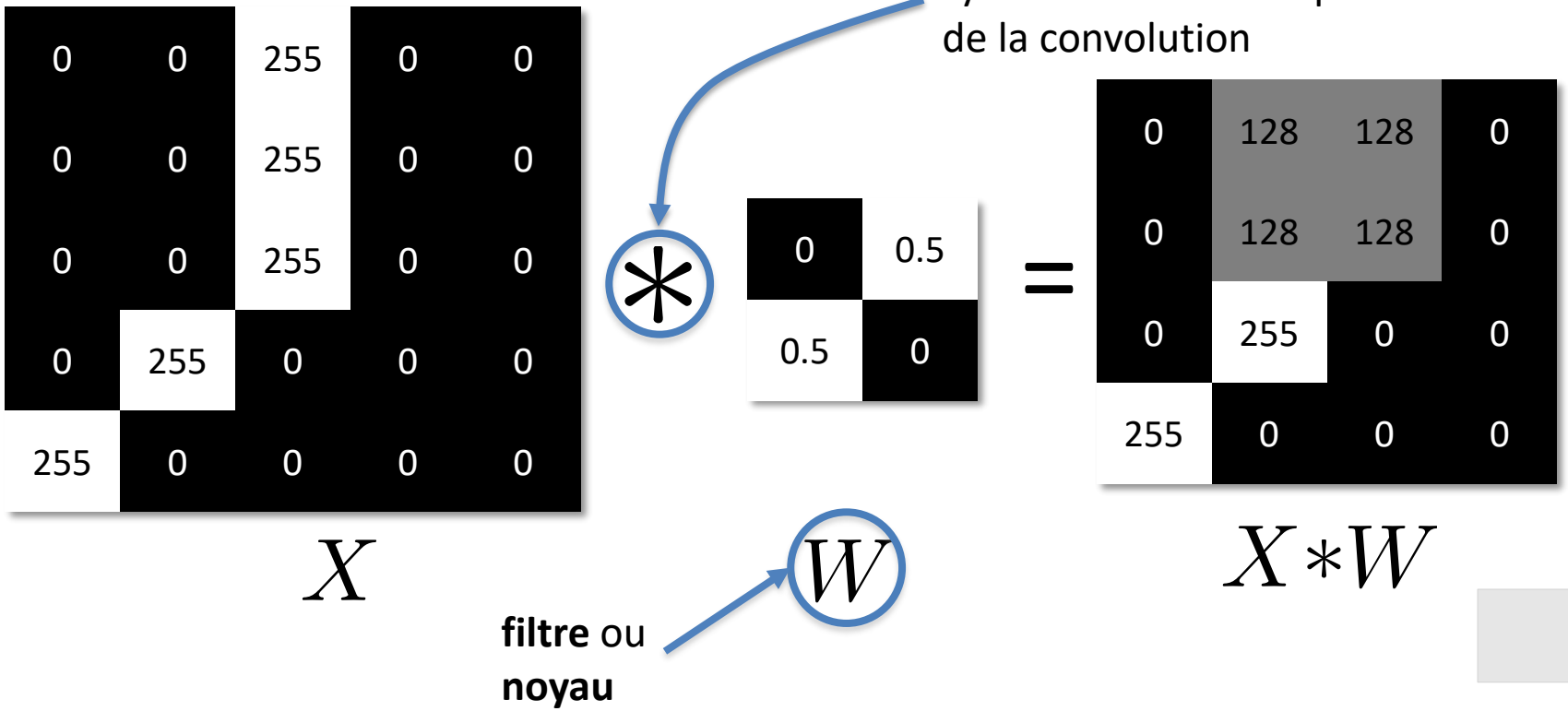
Corrélation 2D sur un signal unitaire



https://towardsdatascience.com/convolution-vs-correlation-af868b6b4fb5

Convolution 2D

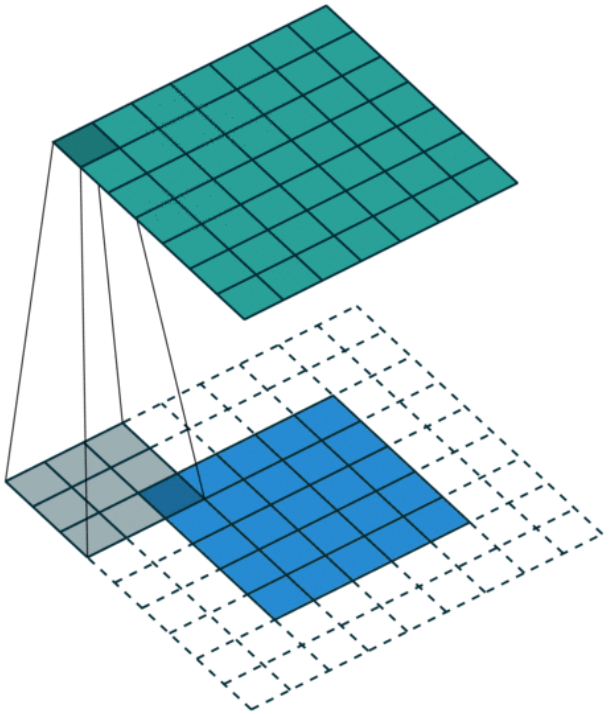
- La **convolution 2D** revient à une corrélation qu'on appliquerait en prenant comme point de référence des indexes du filtre la dernière rangée et la dernière colonne



Convolution 2D

- La **convolution 2D** est une opération liée à la corrélation 2D, au cœur des architectures de réseaux de neurones pour la vision entre autres.
- Équivalent à faire une corrélation après avoir inversé l'ordre des rangées et des colonnes.

```
def convolution (X,W):
  return correlation(X,W::-1, ::-1)
```



https://towardsdatascience.com/convolution-vs-correlation-af868b6b4fb5

- Le résultat est parfois le même
  - Par exemple si le filtre est symétrique horizontalement et verticalement

Si on va plus loin...

- L'estimation des gradients tel que présentée ($X[i,j+1] - X[i,j]$) peut être améliorée
  - voir les filtres de Sobel (*Sobel operator*) http://en.wikipedia.org/wiki/Sobel_operator
- La détection des contours à l'aide d'un simple seuil peut être améliorée
  - voir le filtre de Canny (*Canny edge detector*) http://en.wikipedia.org/wiki/Canny_edge_detector
- On peut extraire à partir des contours l'information sur la présence de lignes droites ou de cercles (ex.: un robot qui veut détecter les limites d'une pièce)
  - http://en.wikipedia.org/wiki/Hough_transform

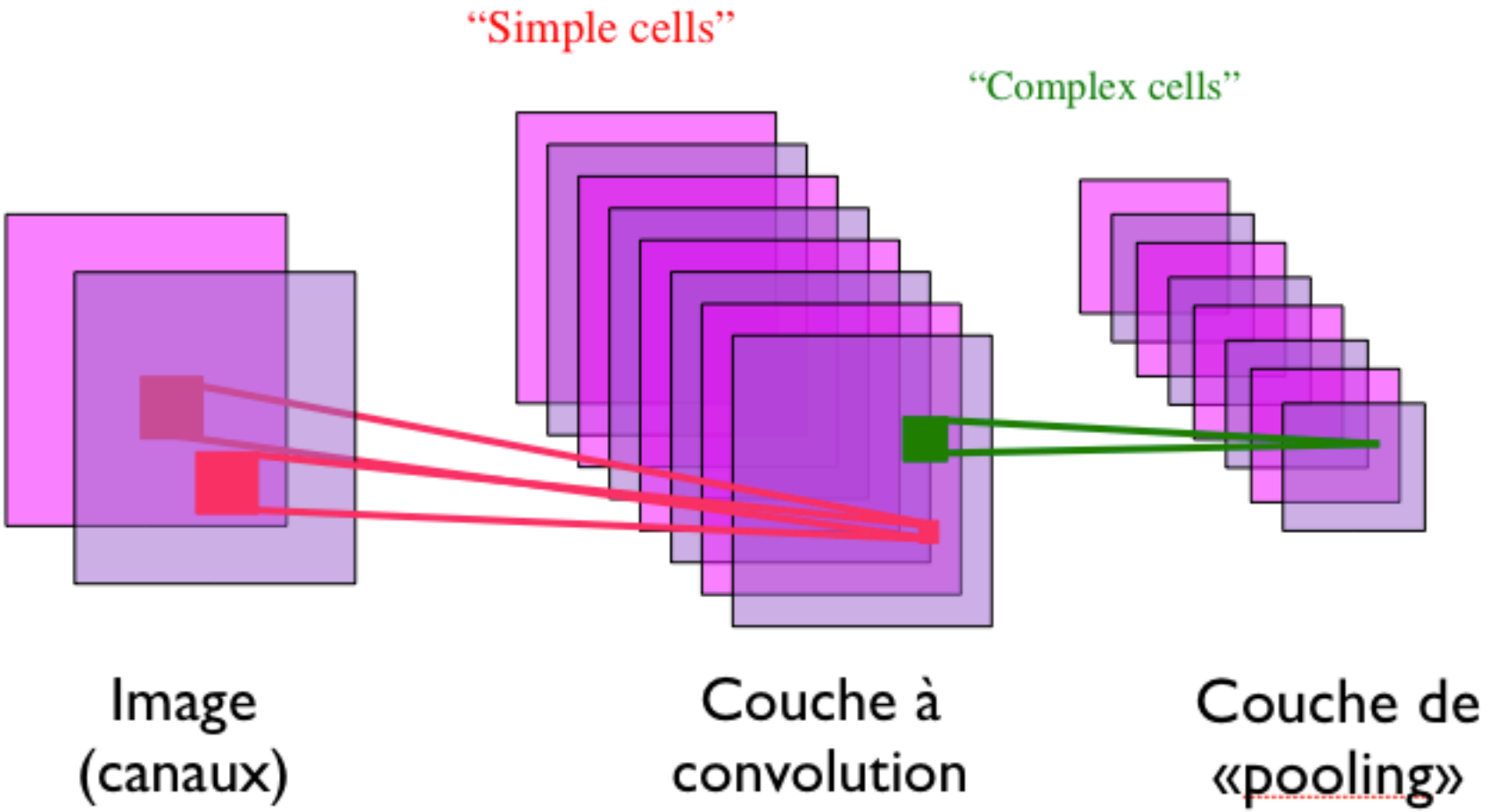
Réseau de neurones à convolution

- Un **réseau de neurones à convolution** est un cas spécial de réseau de neurones
  - Neocognition (Fukushima, 1980)
  - LeNet (LeCun, 1989)

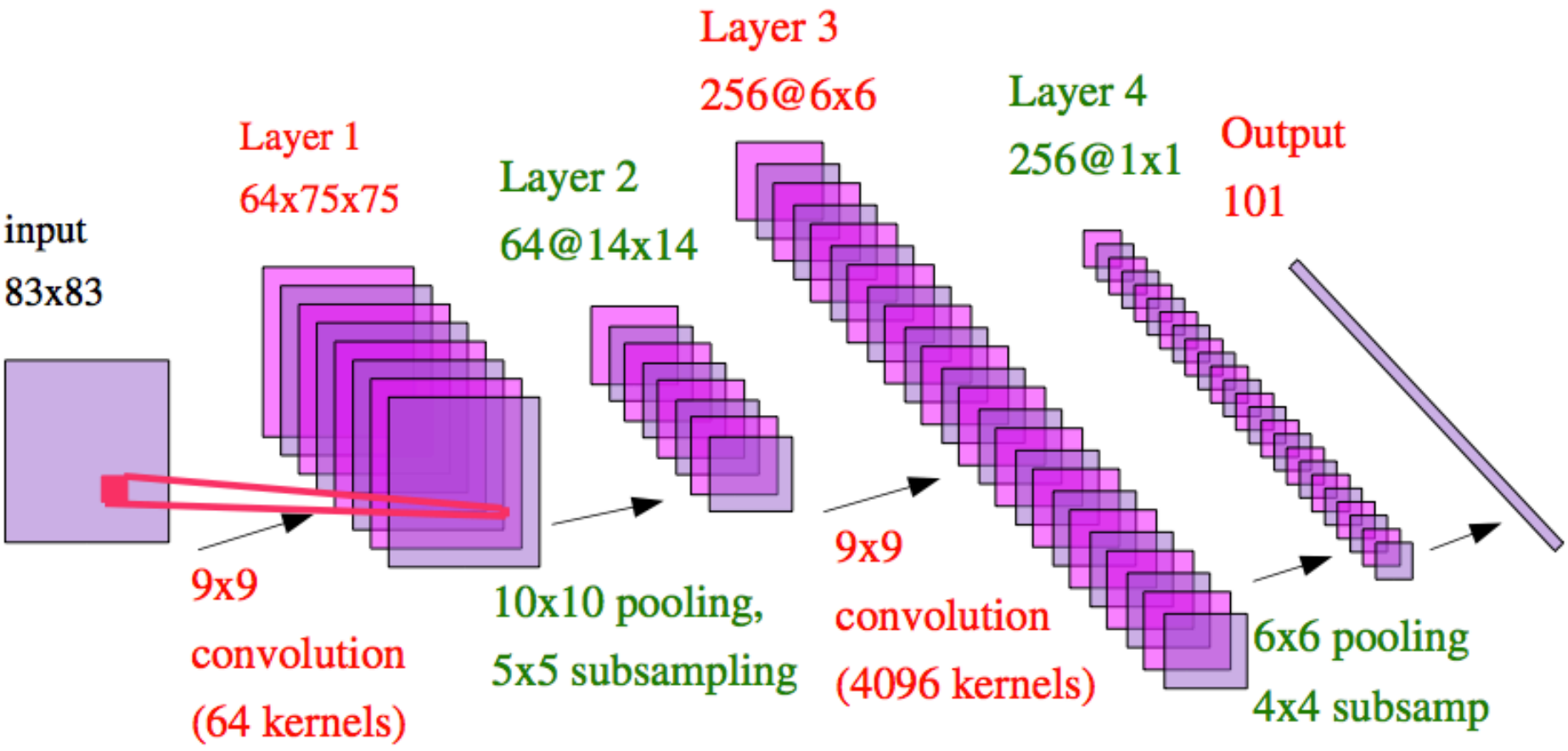
- Comme un réseau de neurones standard, on l'entraîne par descente de gradient stochastique à l'aide de la rétropropagation des gradients

- Spécificité: ils implémentent **3 idées**:
  - connectivité parcimonieuse («sparse»)
  - connectivité locale
  - partage de paramètres

Réseau de neurones à convolution: structure des couches cachées



Réseau de neurones à convolution: réseau complet



Histoire du CNN

- **1959 & 1962**: David Hubel & Torsten Wiesel : Fonctionnement de la vision chez les animaux https://www.youtube.com/watch?v=y_l4kQ5wjiw

- **1980s** : Kunihiko Fukushima – Architecture Neocognitron inspire des travaux de Hubel et Wiesel
  - Neocognitron: A Self-organizing Neural Network Model for a Mechanism of Pattern Recognition Unaffected by Shift in Position"

- **1990s** : LeCun – CNN, inspiré par les travaux de Fukushima
  - Gradient-Based Learning Applied to Document recognition

- **2012** : Hinton et al. – AlexNet performe mieux sur ImageNet mieux que les approches traditionelles
  - ImageNet Classification with Deep Convolutional Neural Networks