

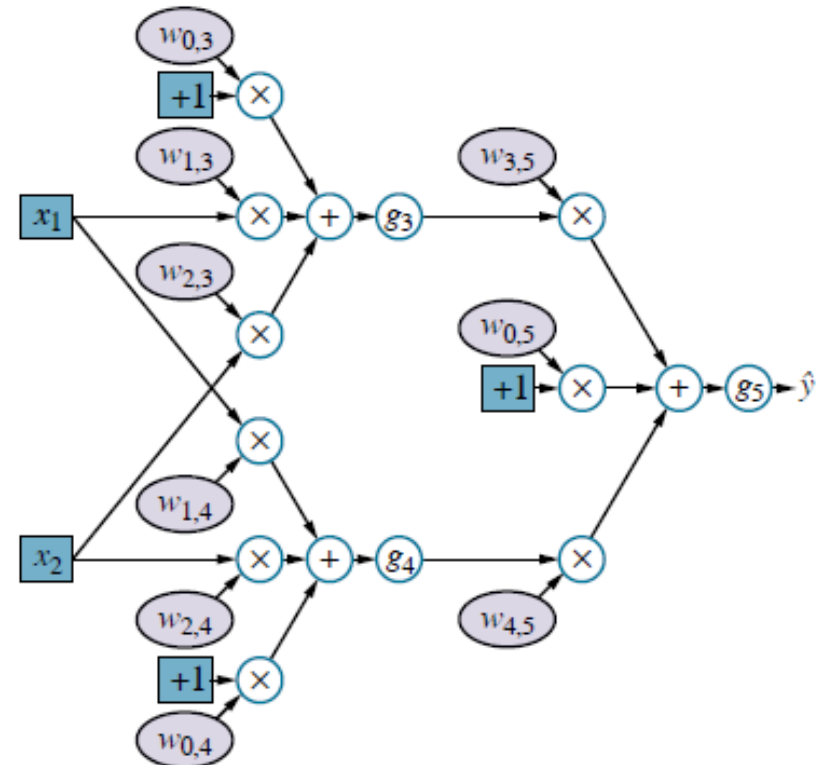
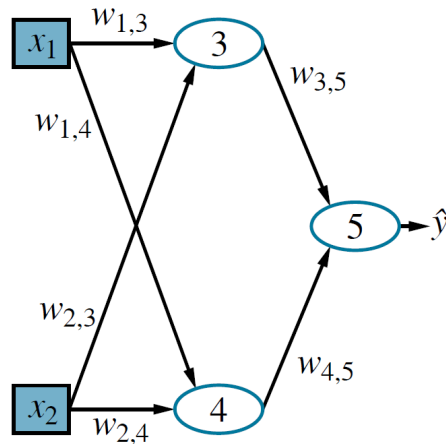
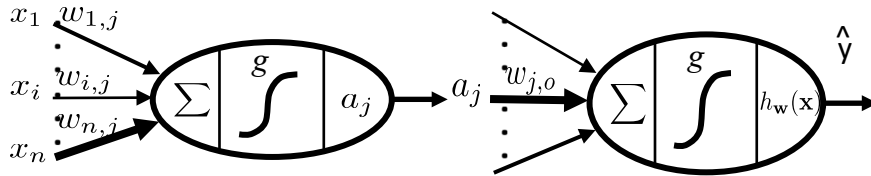
IFT 615 – Intelligence Artificielle

Application – vision artificielle

Professeur: Froduald Kabanza

Assistants: D'Jeff Nkashama

Rappel – Perceptron multicouche



On note:

- g la fonction d'activation
- in_j la somme pondérée par les poids des entrées du neurone j
- a_j la sortie (activité) du neurone

On a donc $a_j = g(in_j) = g(\sum_i w_{ij} a_i)$

Sujets couverts

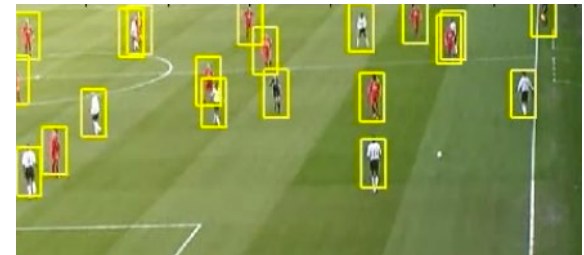
- Opérations bas niveau sur les images
 - ◆ détection de contour
 - ◆ calcul de gradients d'image
 - ◆ Corrélation 2D
 - ◆ Convolution 2D
- Aperçu de l'architecture d'un réseau de neurones à convolution (CNN)

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 $\approx 25\%$ du cerveau sert à la vision
 - ◆ pour l'ouïe, c'est $\approx 8\%$
 - ◆ pour le touché, c'est $\approx 3\%$
- 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

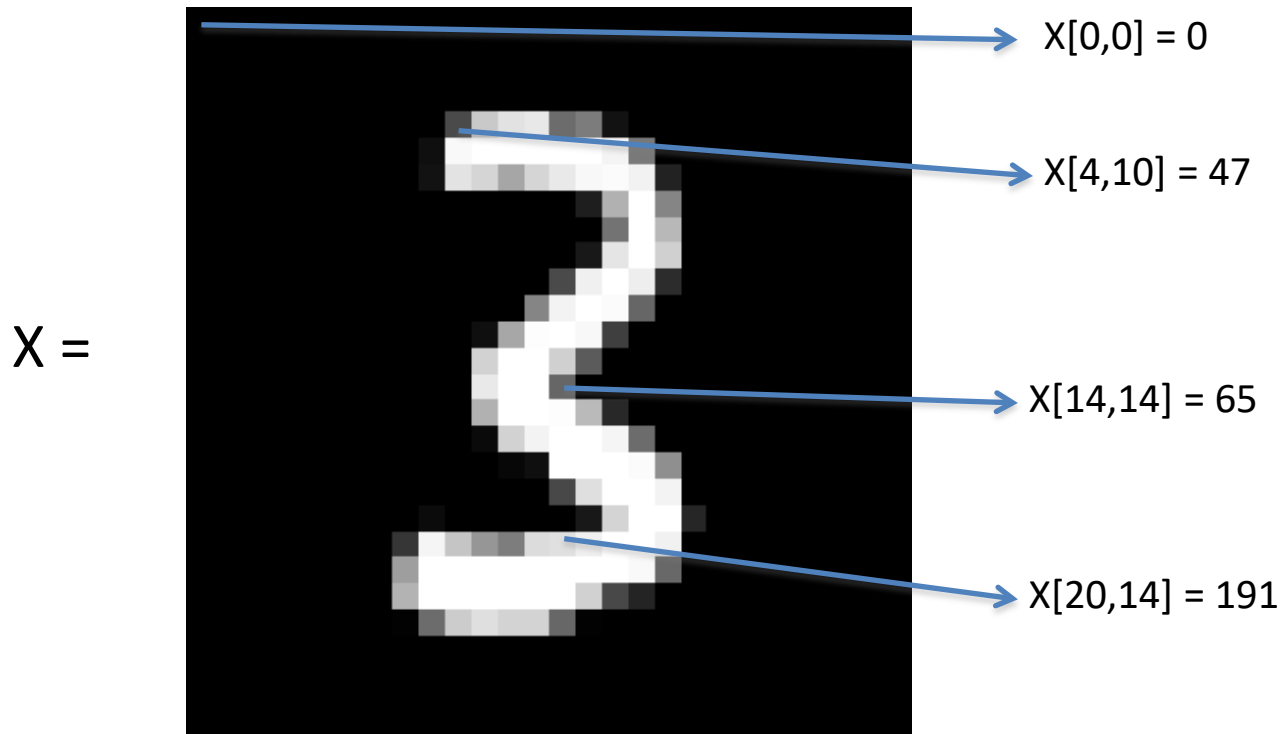


Dans cette leçon...

- On va voir comment on manipule des images
 - ◆ quelle représentation de base utiliser
 - ◆ quel genre de prétraitements sont utiles
- L'objectif est d'introduire certains deux des concepts fondamentaux fréquemment utilisés en vision artificielle
 - ◆ gradients d'image
 - ◆ Convolution
- On voit aussi grossièrement l'architecture d'un réseau de neurones à convolution (CNN)

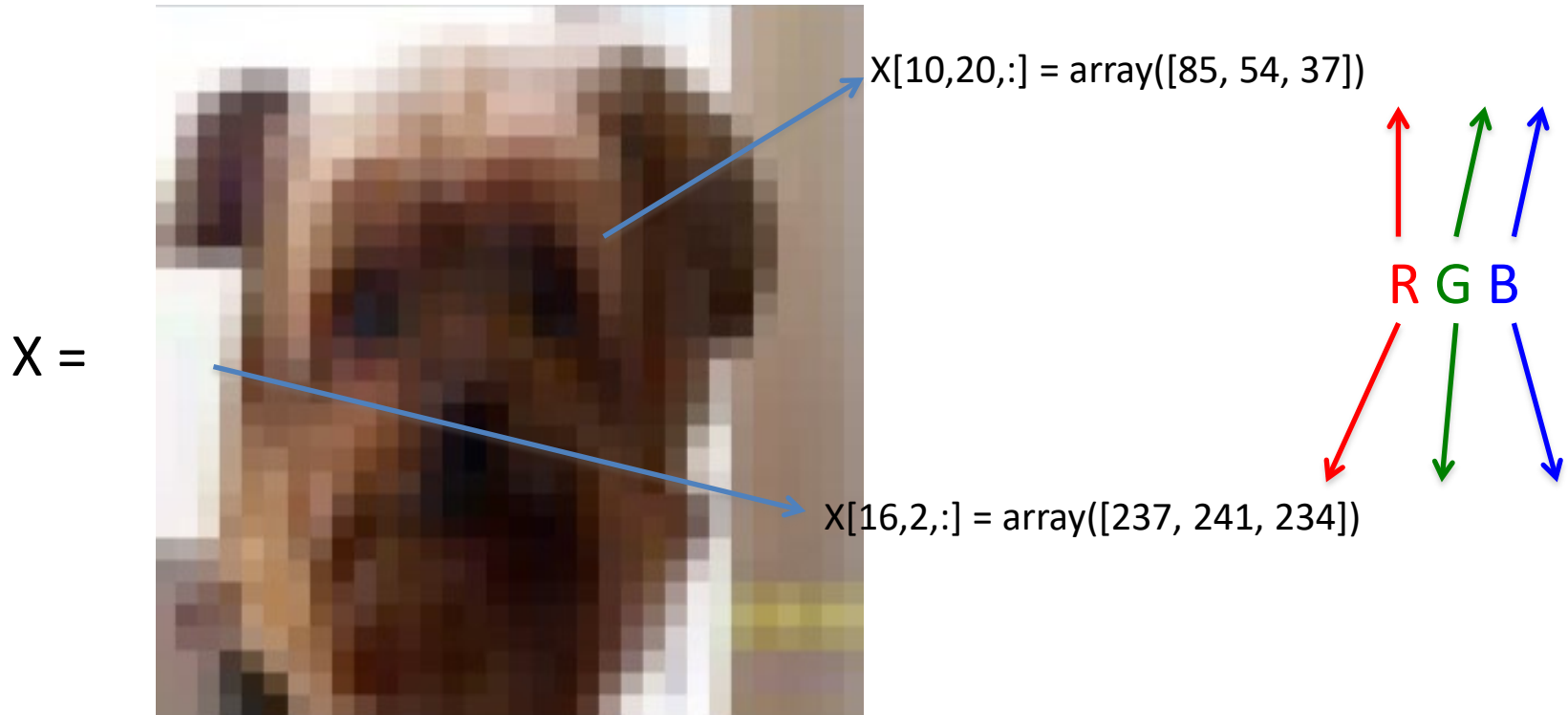
Représentation brute d'une image

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



Représentation brute d'une image

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



Opérations bas niveau sur les images

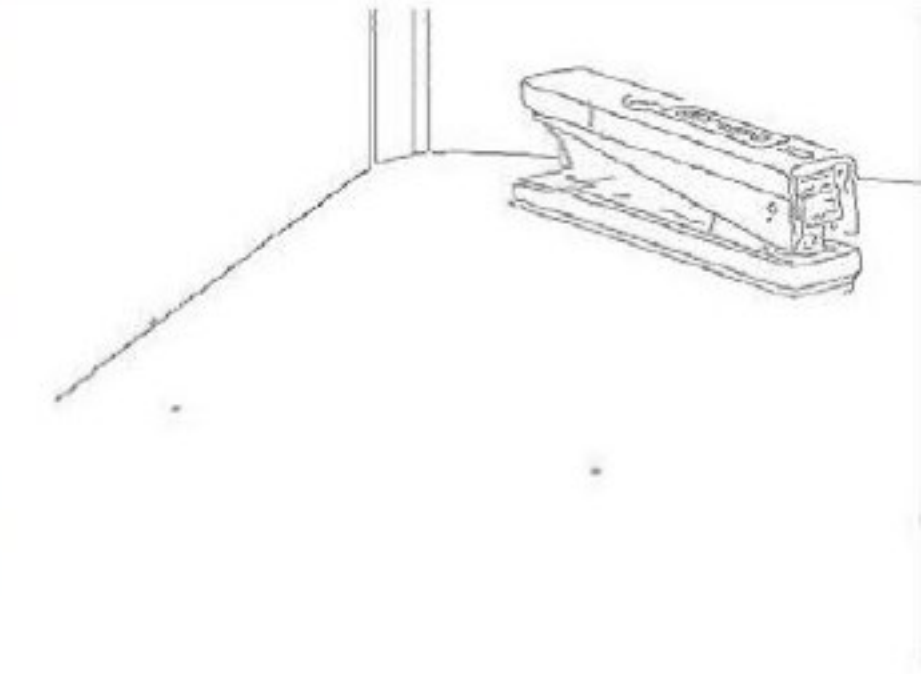
- La représentation sous forme de pixels a des désavantages
 - ◆ elle est lourde, c.-à-d. coûteuse en mémoire
 - » 1024x1024 pixels de 8 bits (en niveau de gris) = 1 MB / image
 - » 1024x1024 pixels de 24bits (canaux RGB) = 3 MB / image
 - ◆ elle contient plus d'information qu'on en a besoin
 - » pour 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

Contour

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



image originale



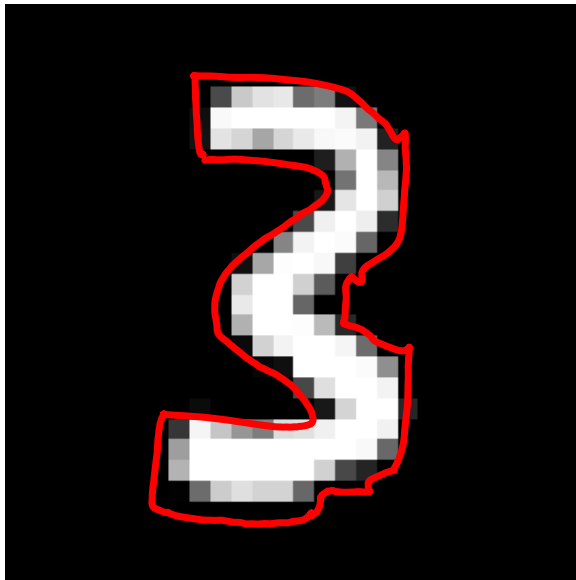
extraction des contours

Vision animale

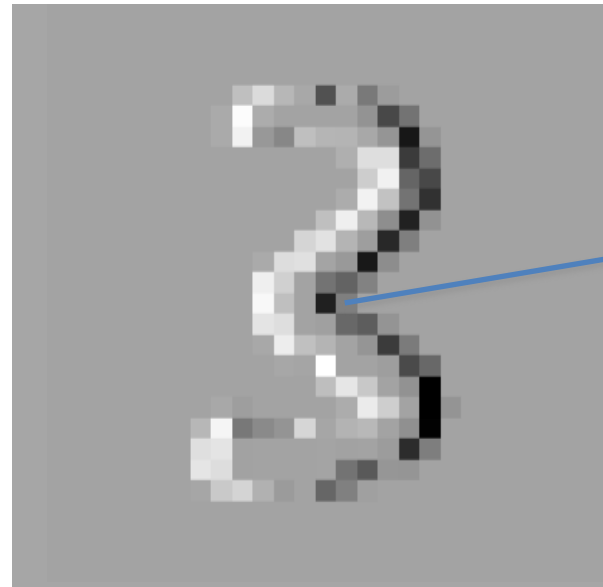
- **1959 & 1962:** [David Hubel & Torsten Wiesel](https://www.youtube.com/watch?v=y_l4kQ5wjiw) : Fonctionnement de la vision chez les animaux https://www.youtube.com/watch?v=y_l4kQ5wjiw

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

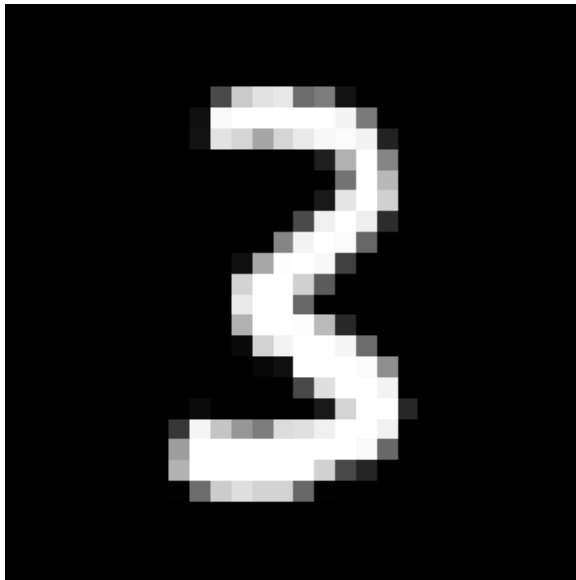


H

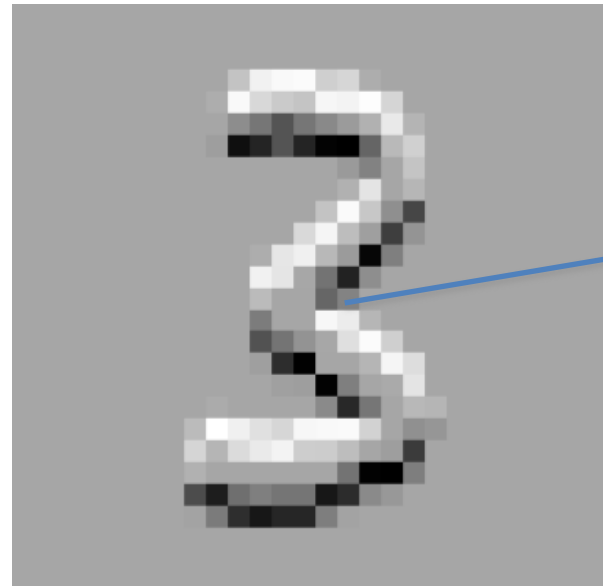
$$H[14,14] = X[14,15] - X[14,14]$$

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

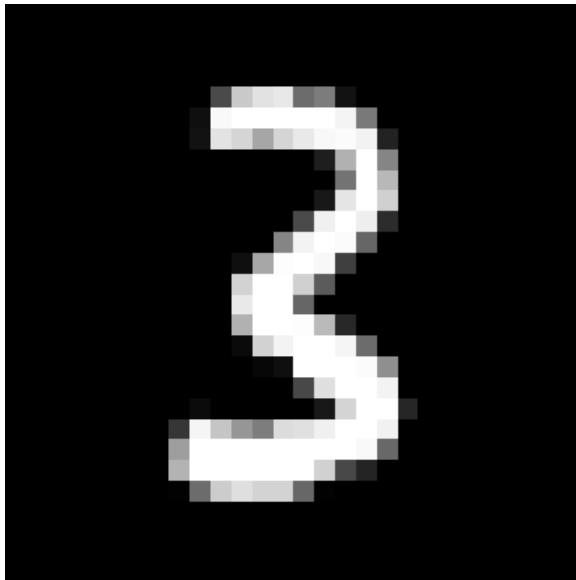
$$V[14,14] = X[15,14] - X[14,14]$$

Détecter un contours à partir des gradients d'image

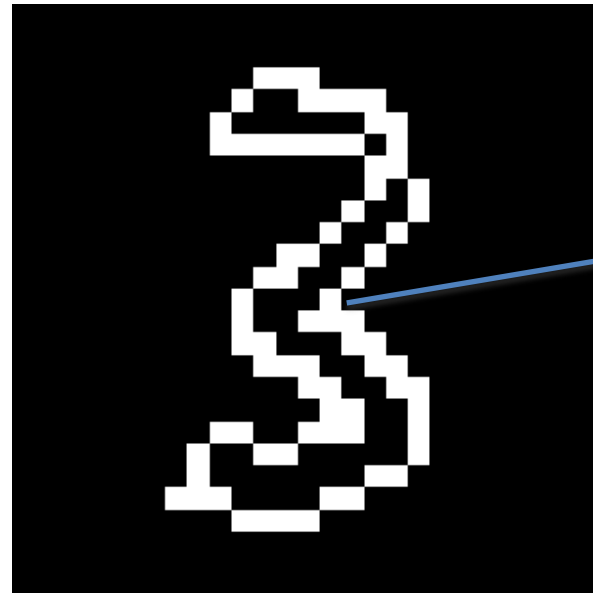
- Un pixel ferait partie d'un contour si la somme des variations (positive ou négative) horizontale et verticale est élevée

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

- On applique un seuil pour déterminer si contour ou pas



X



$E > 128$

$E[14,14] > 128$

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)

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

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

Gradient d'image

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

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

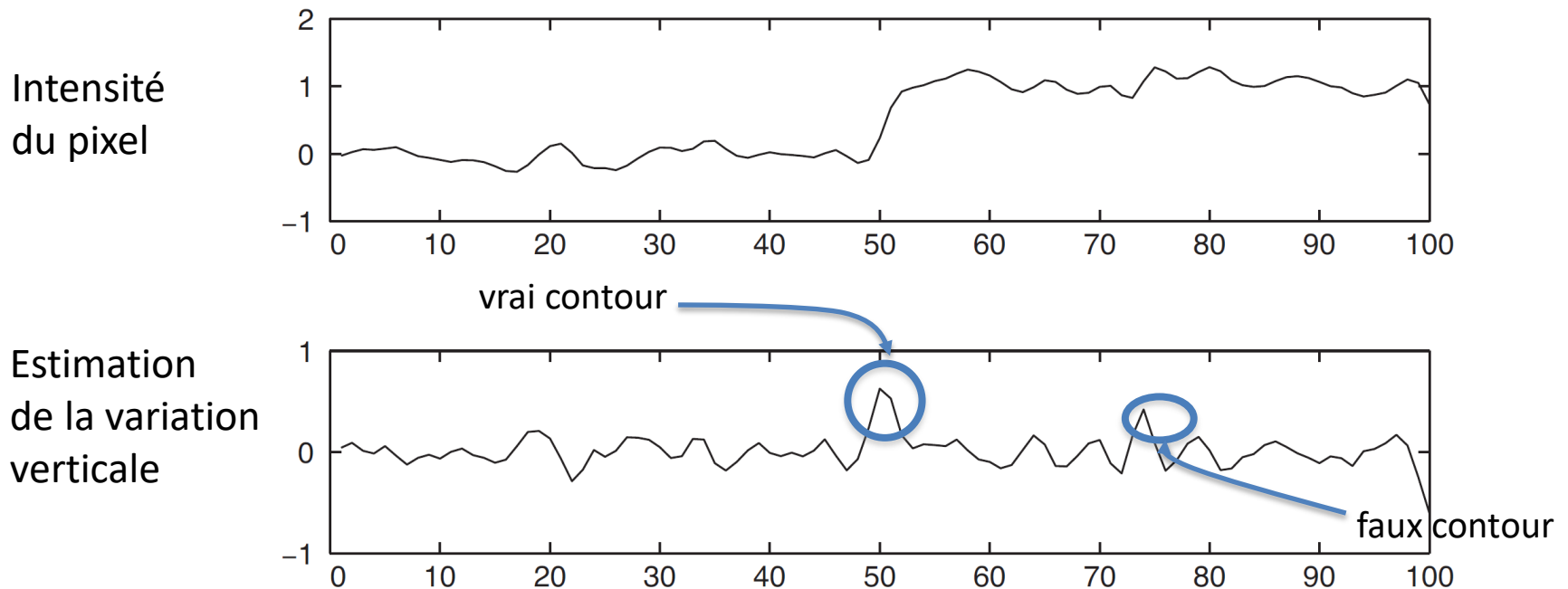
est le **gradient de l'image**, à la position (i,j)

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

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

Exemple de calcul du gradient bruité

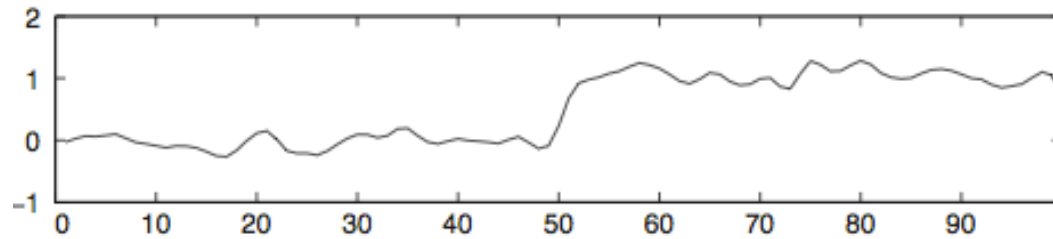
- Des vraies images sont bruitées et donc les variations des gradients vont l'être aussi



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

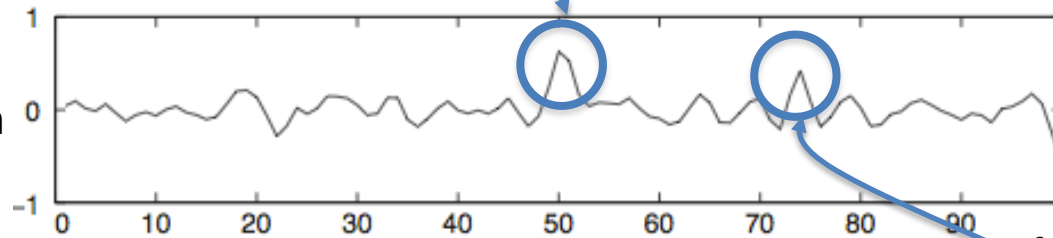
Calcul gradient d'image lissé

Intensité
du pixel



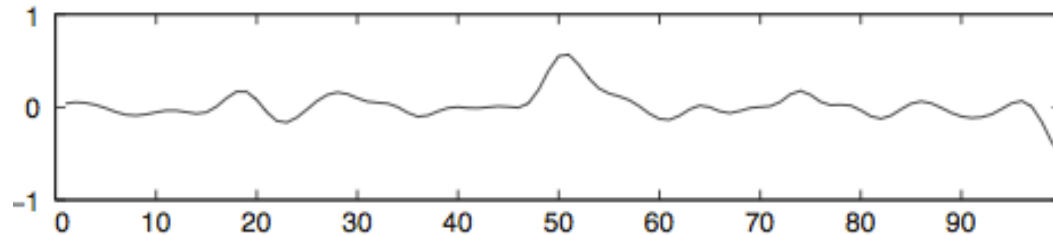
vrai contour

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

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

Lissage Gaussien d'une image

0	0	255	0	0
0	0	255	0	0
0	0	255	0	0
0	255	0	0	0
255	0	0	0	0

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

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

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

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

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 fonctions f et g ($u = f \star g$), 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 $\pm \infty$ par $\pm 3\sigma$

Lissage Gaussien d'une image

0	0	255	0	0
0	0	255	0	0
0	0	255	0	0
0	255	0	0	0
255	0	0	0	0

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

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

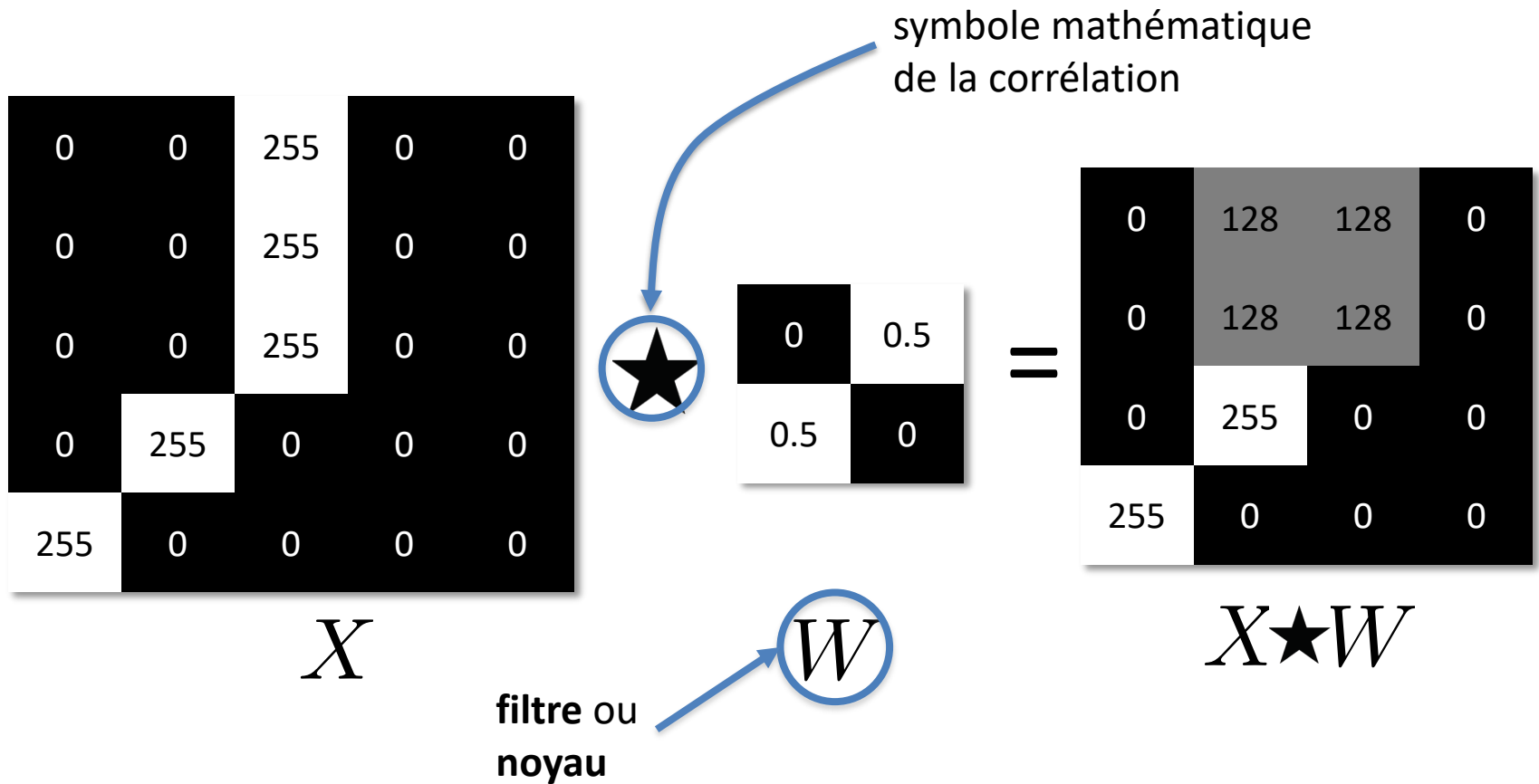
$$C(i_0, j_0) = \sum_i \sum_j X[i,j] G_\sigma(d),$$

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

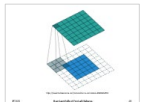
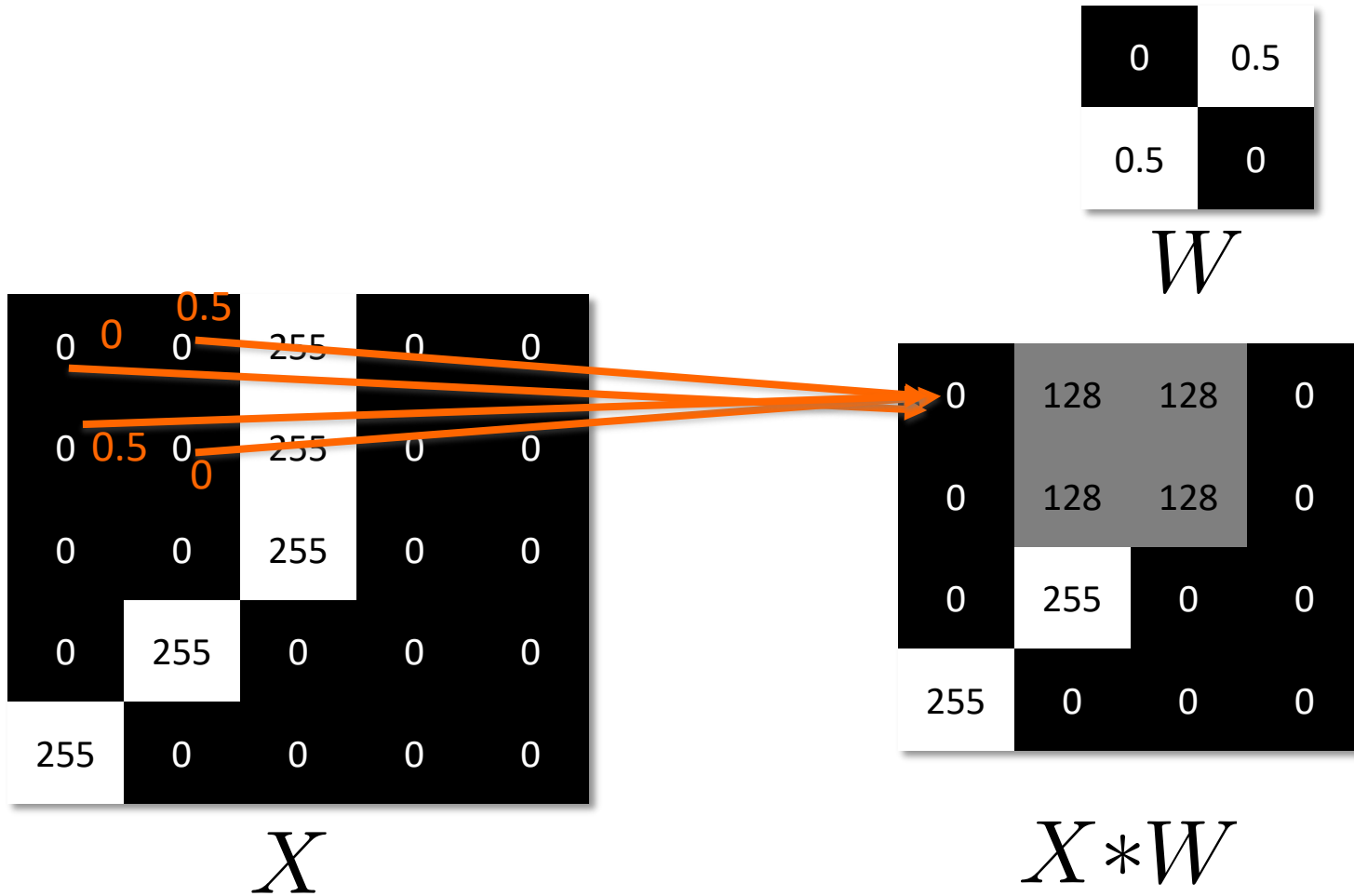
$$G_\sigma(x) = \frac{1}{\sqrt{2\pi}\sigma} e^{-x^2/2\sigma^2}$$

Corrélation 2D

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



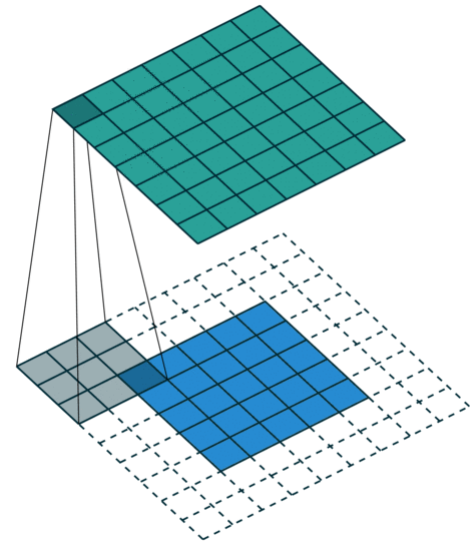
Correlation 2D



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 $W = \text{array}([[-1,1]])$

The diagram illustrates the 2D correlation operation. It shows the input matrix X , the kernel W , and the resulting output matrix $X \star W$.

Input Matrix X (5x5):

0	0	255	0	0
0	0	255	0	0
0	0	255	0	0
0	255	0	0	0
255	0	0	0	0

Kernel W (1x2):

-1	1
----	---

Output Matrix $X \star W$ (5x4):

0	255	-255	0
0	255	-255	0
0	255	-255	0
255	-255	0	0
-255	0	0	0

Corrélation 2D

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

The diagram illustrates the 2D correlation operation. It shows the input matrix X , the kernel W , and the resulting output matrix $X \star W$.

Input Matrix X :

0	0	255	0	0
0	0	255	0	0
0	0	255	0	0
0	255	0	0	0
255	0	0	0	0

Kernel W :

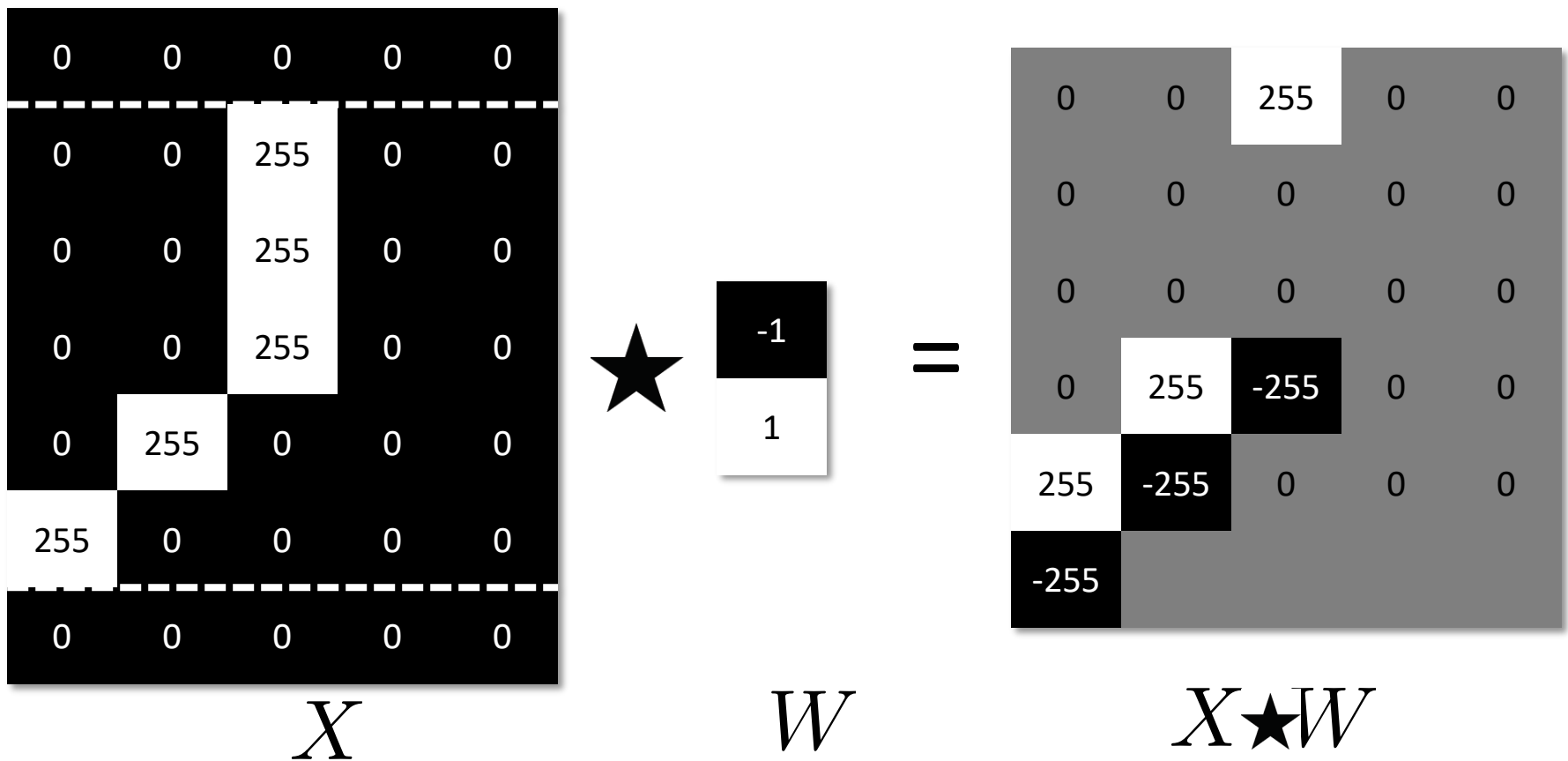
-1
1

Output Matrix $X \star W$:

0	0	0	0	0
0	0	0	0	0
0	255	-255	0	0
255	-255	0	0	0
0	0	0	0	0

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

						Padded f							
↙	Origin f												
	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	w	0	0	0	1	0	0	0
	0	0	1	0	0	1 2 3	0	0	0	0	0	0	0
	0	0	0	0	0	4 5 6	0	0	0	0	0	0	0
	0	0	0	0	0	7 8 9	0	0	0	0	0	0	0

(a) (b)

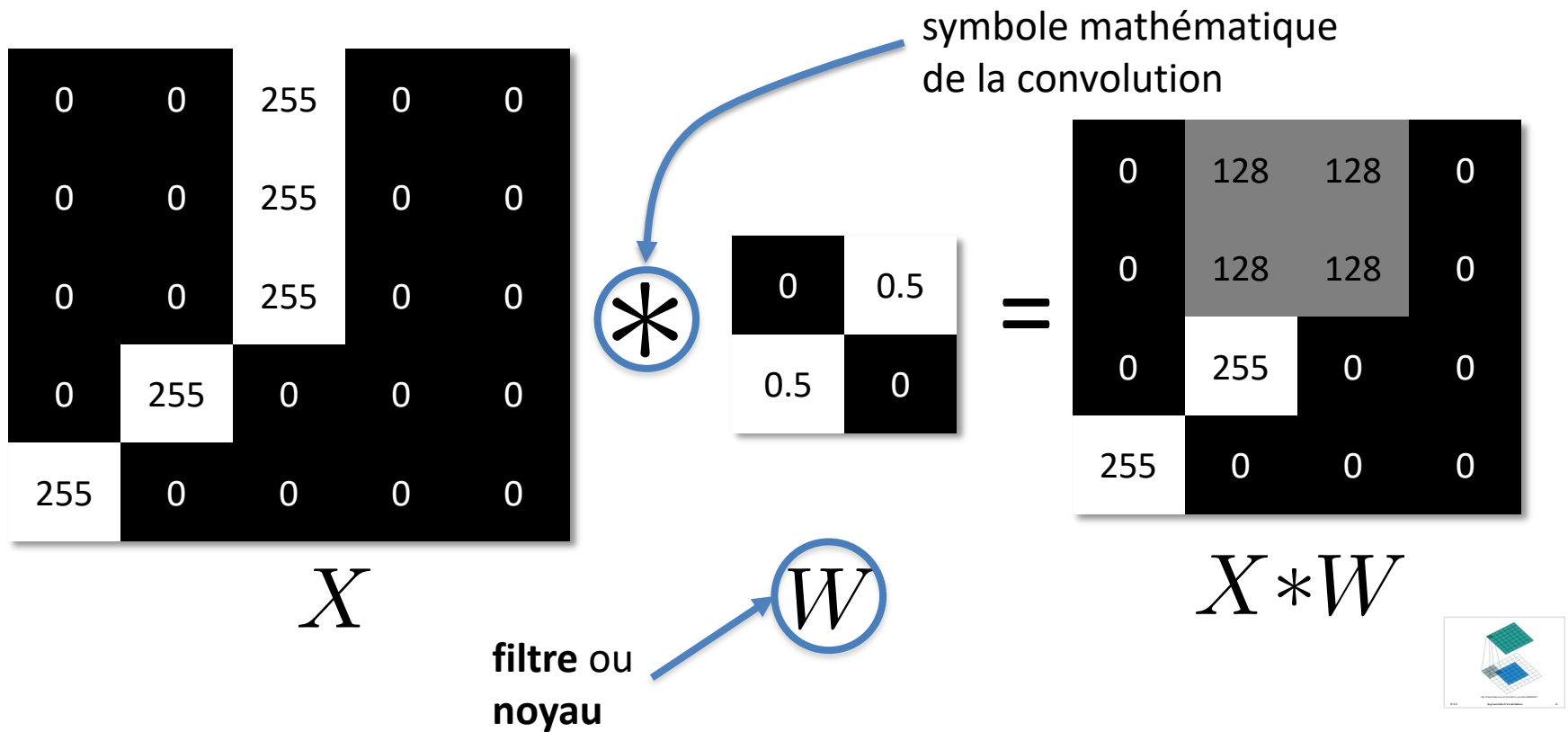
↙ Initial position for w						Correlation result					Full correlation result				
1 2 3	0	0	0	0	0						0	0	0	0	0
4 5 6	0	0	0	0	0		0	0	0	0	0	0	0	0	0
7 8 9	0	0	0	0	0		0	9 8 7	0		0	0	9 8 7	0	0
0	0	0	1	0	0	0	0	6 5 4	0		0	0	6 5 4	0	0
0	0	0	0	0	0	0	0	3 2 1	0		0	0	3 2 1	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

(c) (d) (e)

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

Convolution 2D

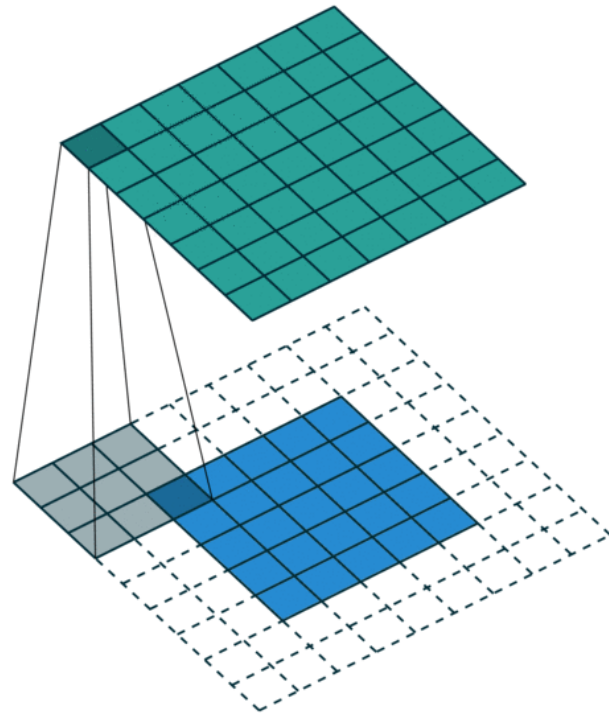
- La **convolution 2D** revient à une corrélation qu'on appliquerait en prenant comme point de référence des indexes du filtre la dernière rangée et la dernière colonne



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)
```



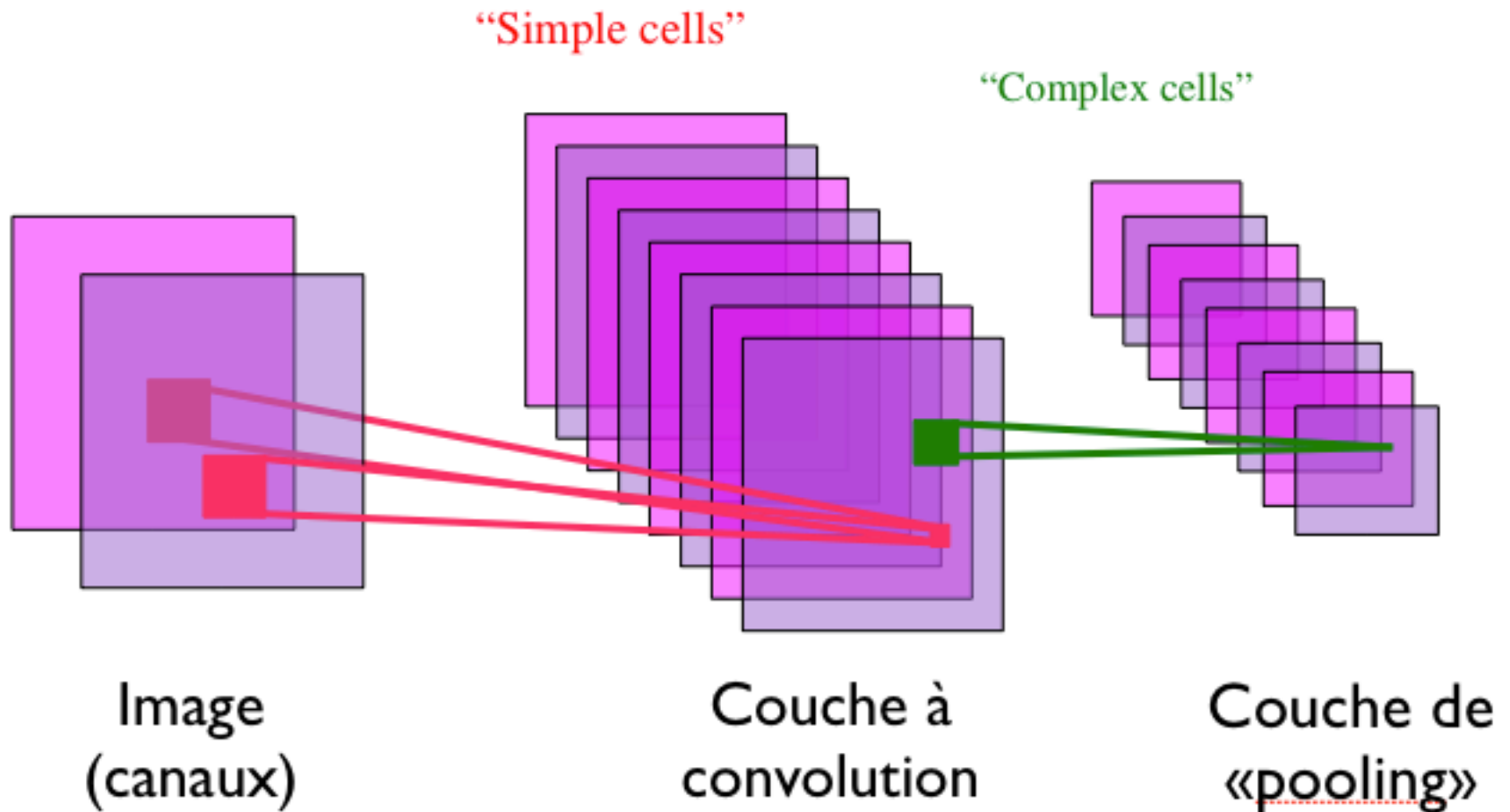
<https://towardsdatascience.com/convolution-vs-correlation-af868b6b4fb5>

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

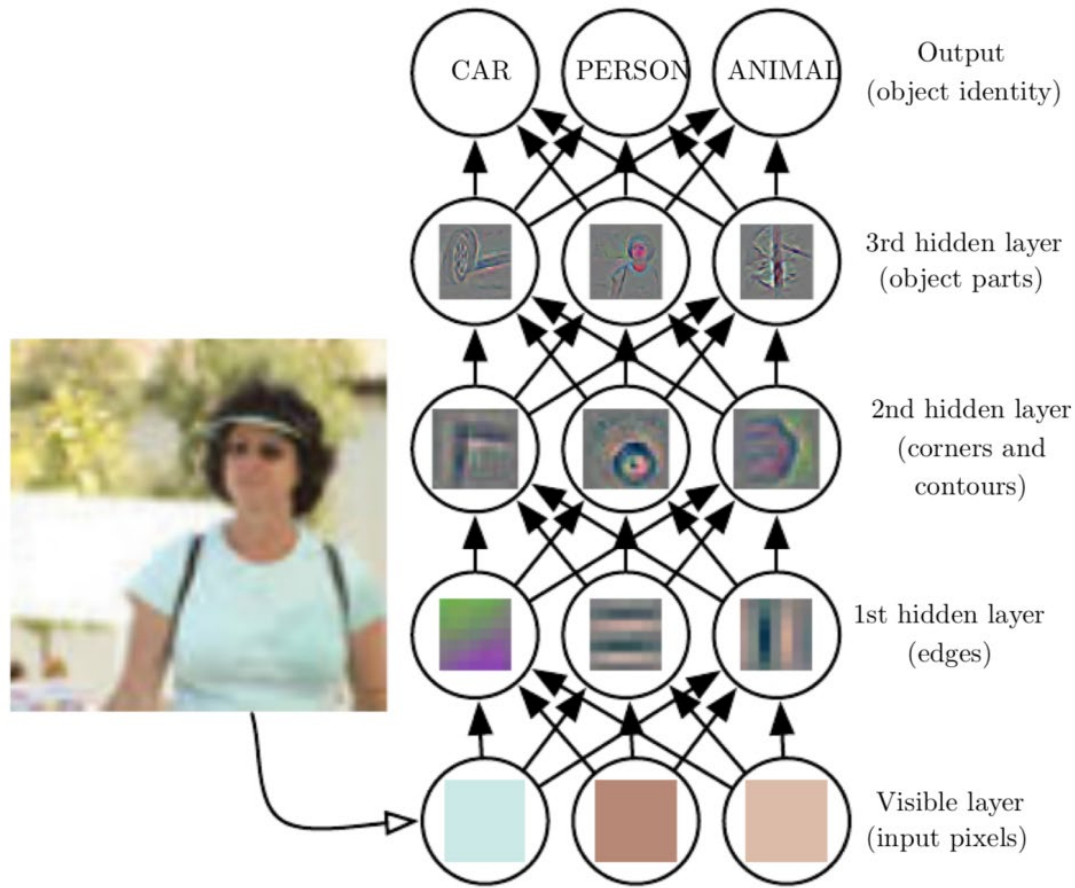
Réseau de neurones à convolution

- 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



Chaque couche apprend une abstraction



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

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** : Kunihiro 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 traditionnelles
 - ◆ [ImageNet Classification with Deep Convolutional Neural Networks](#)

Conclusion

- La vision artificielle est un des problèmes fondamentaux de l'IA
- Plusieurs algorithmes de vision artificielle sont basés sur des CNN : *ResNet*, *YOLO* et plus
- Plusieurs plateformes supporte ces algorithmes: *Anaconda*, *Scikit Learn*, *Amazon SageMaker*, *Microsoft Azure ML*, *Google Vertex AI* et d'autres

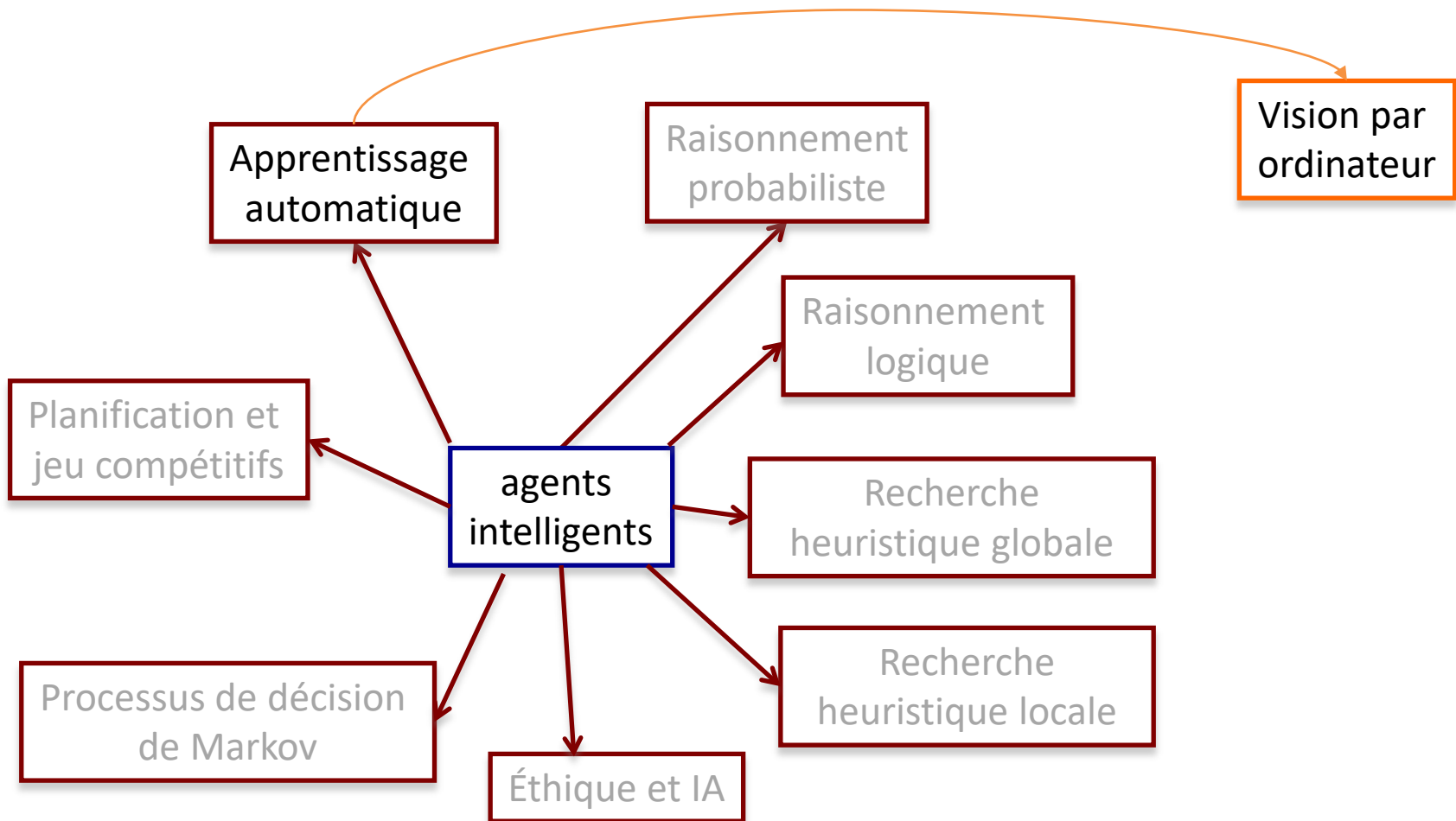
Conclusion

- Ce cours a introduit les concepts de détection de contours et de convolution
- 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

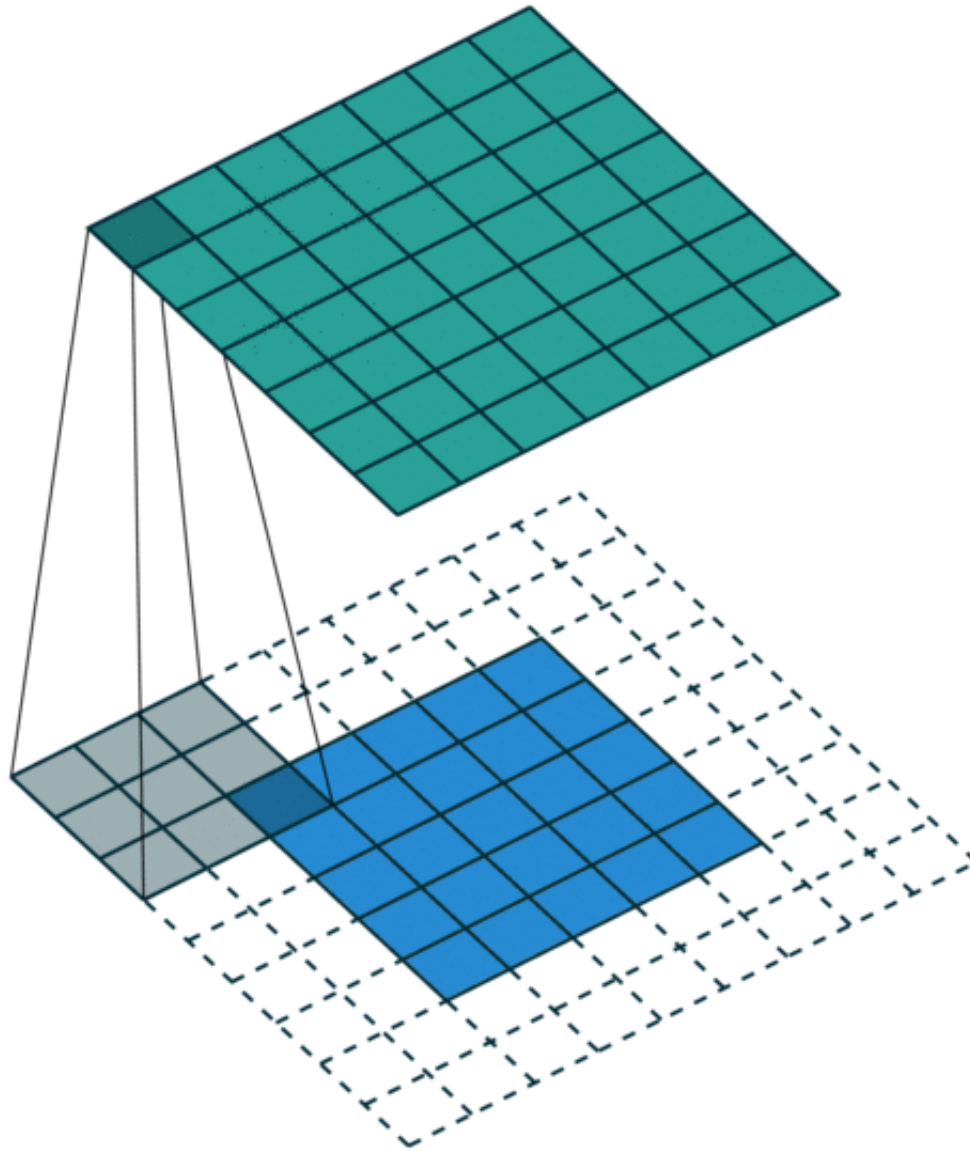
Concepts et algorithmes

Applications



Vous devriez être capable de...

- Décrire globalement ce qu'est un contour et comment on peut le détecter
- Décrire ce qu'est un gradient d'image et connaître ses propriétés
- Décrire ce qu'est une convolution et savoir la calculer
- Expliquer la différence entre une corrélation et une convolution.
- Expliquer ce qui distingue un réseau de neurones à convolution d'un réseau de neurones standard (perceptron multi-couches)



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