

RÉSEAU DE NEURONES

PRESENTED BY

Soltane sadok

SOMMAIRE

01

**INTRODUCTION AUX NEURONES
BIOLOGIQUES**

02

**HISTORIQUE DES RÉSEAUX DE NEURONES
ARTIFICIELS**

03

**FONDEMENTS DES RÉSEAUX DE NEURONES
ARTIFICIELS**

04

**ARCHITECTURES DE RÉSEAU
POPULAIRES(CNN)**

05

**QU'EST CE QU'UN RÉSEAUX DE
NEURONES CONVOLUTIVE?**

06

LES COUCHES CNN

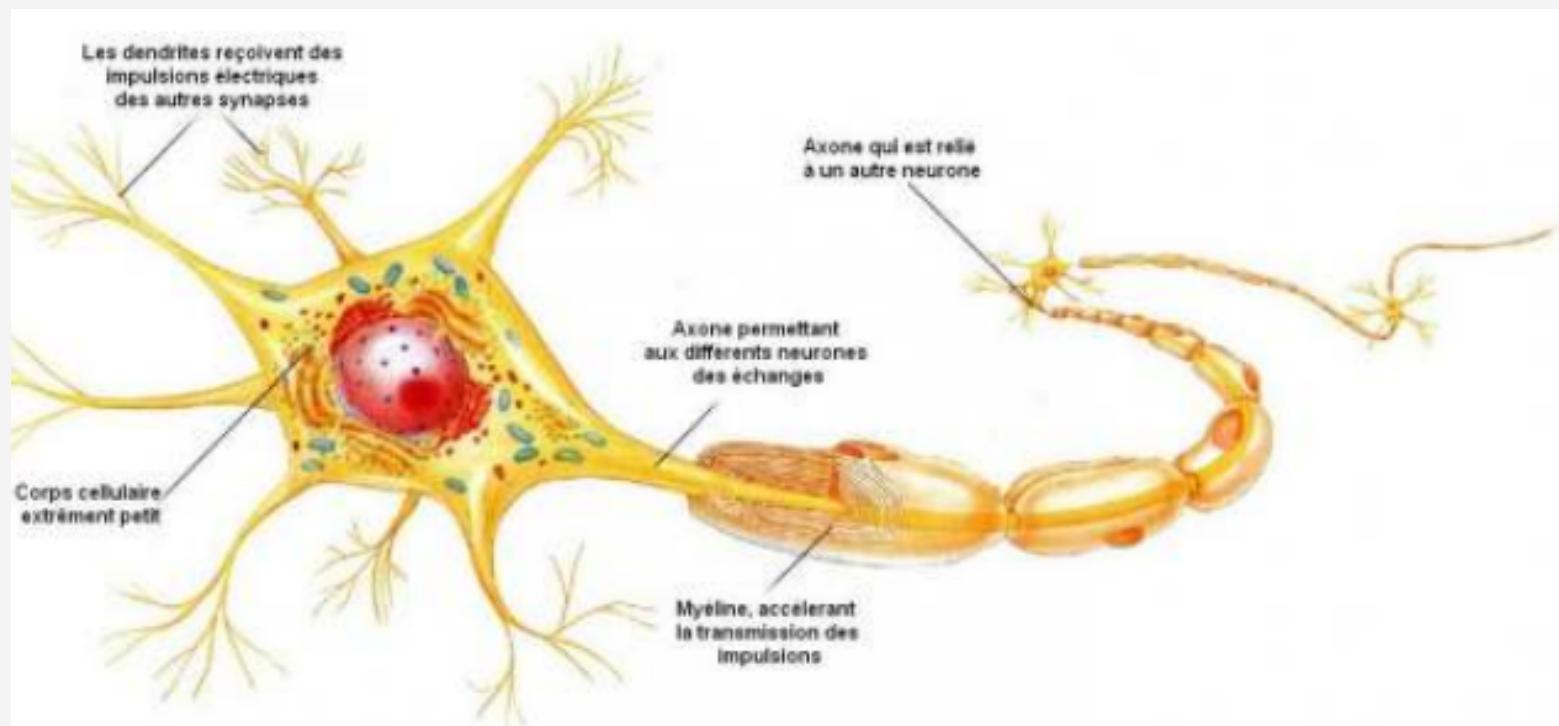
07

ARCHITECTURE D'UN CNN

08

IMPLÉMENTATION

Introduction aux neurones biologiques



Les neurones sont les cellules de base du système nerveux et jouent un rôle crucial dans le traitement de l'information dans le cerveau humain. Chaque neurone est composé de trois parties principales : le corps cellulaire, les dendrites et l'axone.

- Le corps cellulaire contient le noyau et les organites cellulaires nécessaires au fonctionnement du neurone.
- Les dendrites sont de minces prolongements qui reçoivent les signaux électriques (potentiel d'action) des autres neurones ou des cellules sensorielles.
- L'axone est un prolongement unique et long du neurone qui transmet les signaux électriques (potentiel d'action) aux autres neurones ou aux cellules musculaires ou glandulaires.

Lorsqu'un signal électrique est suffisamment fort pour dépasser un seuil au niveau du corps cellulaire, il déclenche la génération d'un potentiel d'action qui se propage le long de l'axone. Ce signal est ensuite transmis aux neurones voisins via des synapses, des points de connexion entre les axones et les dendrites.

Le rôle des neurones dans le traitement de l'information consiste à recevoir, intégrer et transmettre des signaux électriques et chimiques, permettant ainsi la transmission rapide et efficace des informations à travers le système nerveux.

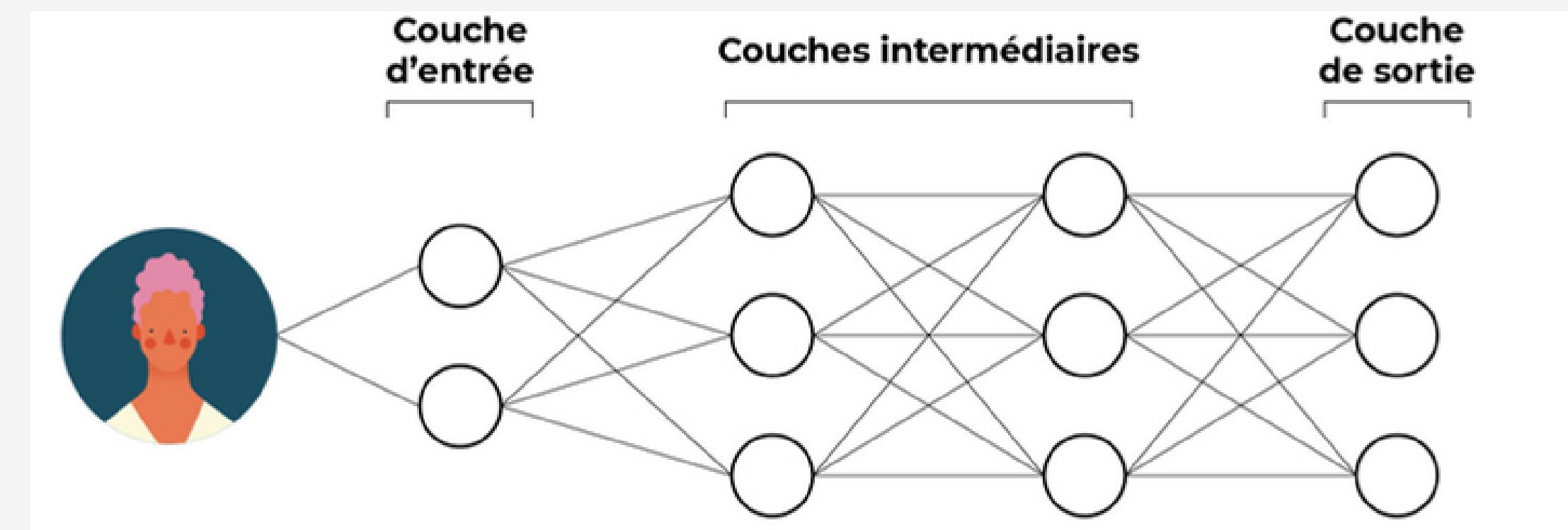


HISTORIQUE DES RÉSEAUX DE NEURONES ARTIFICIELS

Les premières idées sur les réseaux de neurones artificiels remontent aux travaux de Warren McCulloch et Walter Pitts en 1943, qui ont proposé un modèle de neurone formel basé sur des seuils binaires et des fonctions booléennes. Ils ont démontré que ces réseaux de neurones formels étaient capables de réaliser des calculs logiques élémentaires.

En 1958, Frank Rosenblatt a développé le perceptron, un type de réseau de neurones artificiels capable d'apprendre à classer des entrées en deux catégories. Le perceptron était constitué d'une seule couche de neurones connectés aux entrées et produisant une sortie binaire.

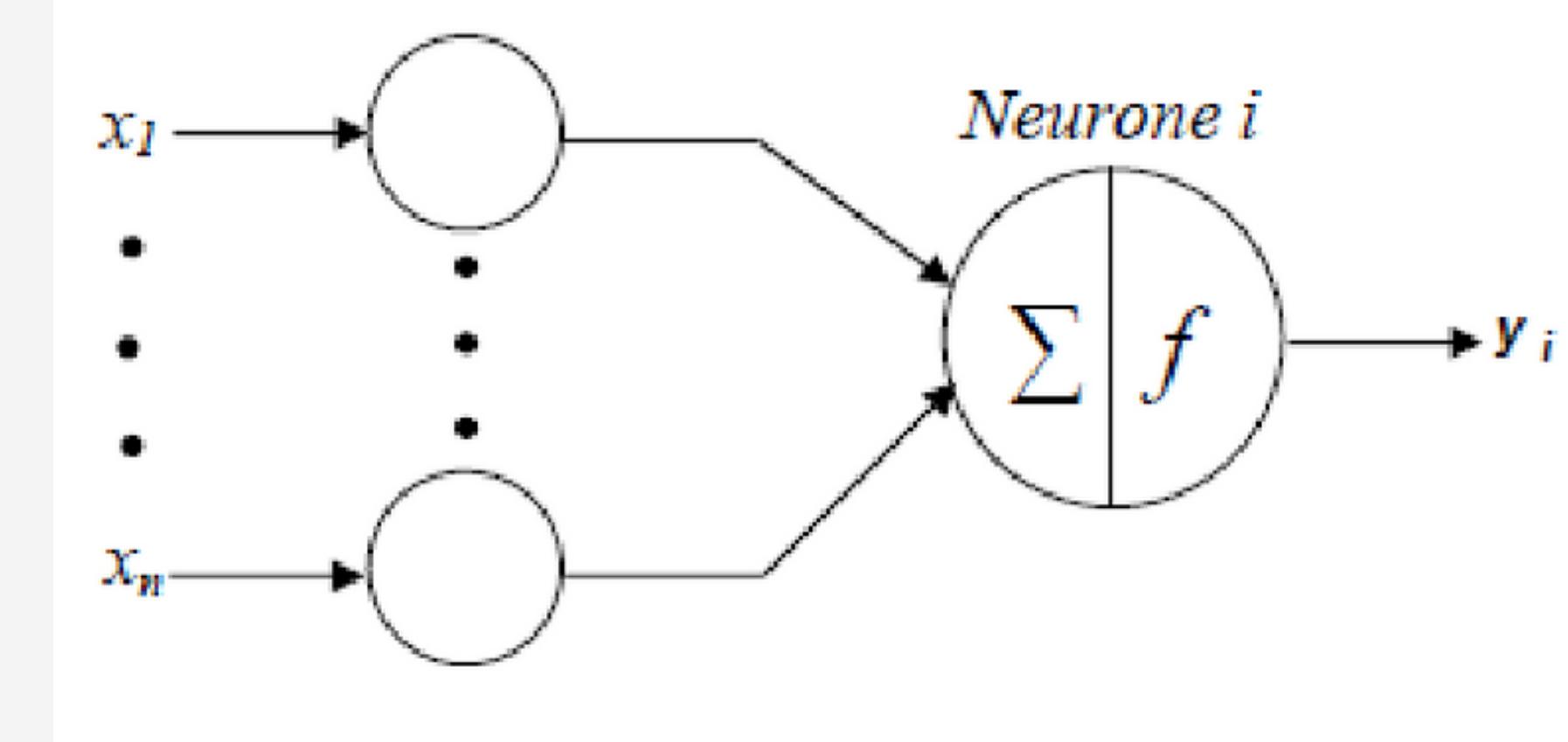
Le perceptron et les travaux de Rosenblatt ont jeté les bases du développement ultérieur des réseaux de neurones artificiels, mais leur utilisation était limitée en raison de leur incapacité à résoudre des problèmes non linéaires ou à apprendre des représentations hiérarchiques de données complexes.



FONDEMENTS DES RÉSEAUX DE NEURONES ARTIFICIELS

Neurones formels

Les neurones formels, ou perceptrons, sont les unités de base des réseaux de neurones artificiels. Inspirés des neurones biologiques, les neurones formels reçoivent des entrées, appliquent des poids à ces entrées, effectuent une somme pondérée des entrées et des poids, puis appliquent une fonction d'activation pour produire une sortie. La sortie du neurone formel est généralement utilisée comme entrée pour d'autres neurones dans le réseau.



Connexions pondérées

Les connexions entre les neurones dans un réseau de neurones artificiels sont pondérées, ce qui signifie que chaque connexion a un poids associé. Ces poids déterminent l'importance relative des entrées pour chaque neurone. Pendant l'apprentissage, les poids sont ajustés afin d'optimiser les performances du réseau sur une tâche spécifique.

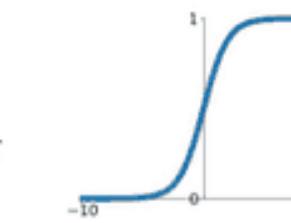
Fonctions d'activation

Les fonctions d'activation sont des fonctions mathématiques appliquées à la sortie pondérée d'un neurone formel pour introduire une non-linéarité dans le réseau. Ils permettent aux réseaux de neurones d'apprendre et de représenter des relations non linéaires entre les entrées et les sorties. Quelques exemples de fonctions d'activation couramment utilisées incluent la fonction sigmoïde, la fonction tangente hyperbolique (tanh), la fonction ReLU (Rectified Linear Unit), etc.

Activation Functions

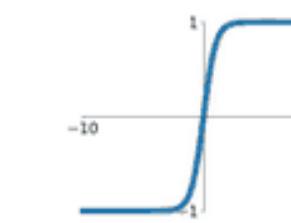
Sigmoid

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



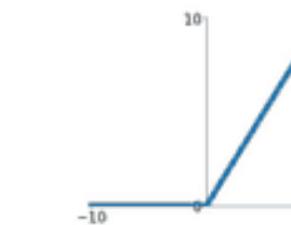
tanh

$$\tanh(x)$$



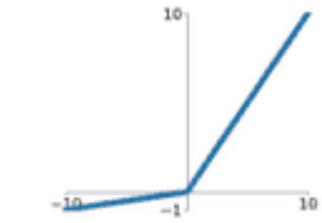
ReLU

$$\max(0, x)$$



Leaky ReLU

$$\max(0.1x, x)$$

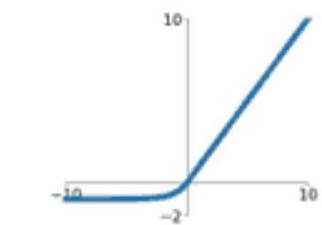


Maxout

$$\max(w_1^T x + b_1, w_2^T x + b_2)$$

ELU

$$\begin{cases} x & x \geq 0 \\ \alpha(e^x - 1) & x < 0 \end{cases}$$



Types d'apprentissage

Il existe plusieurs types d'apprentissage dans les réseaux de neurones artificiels :

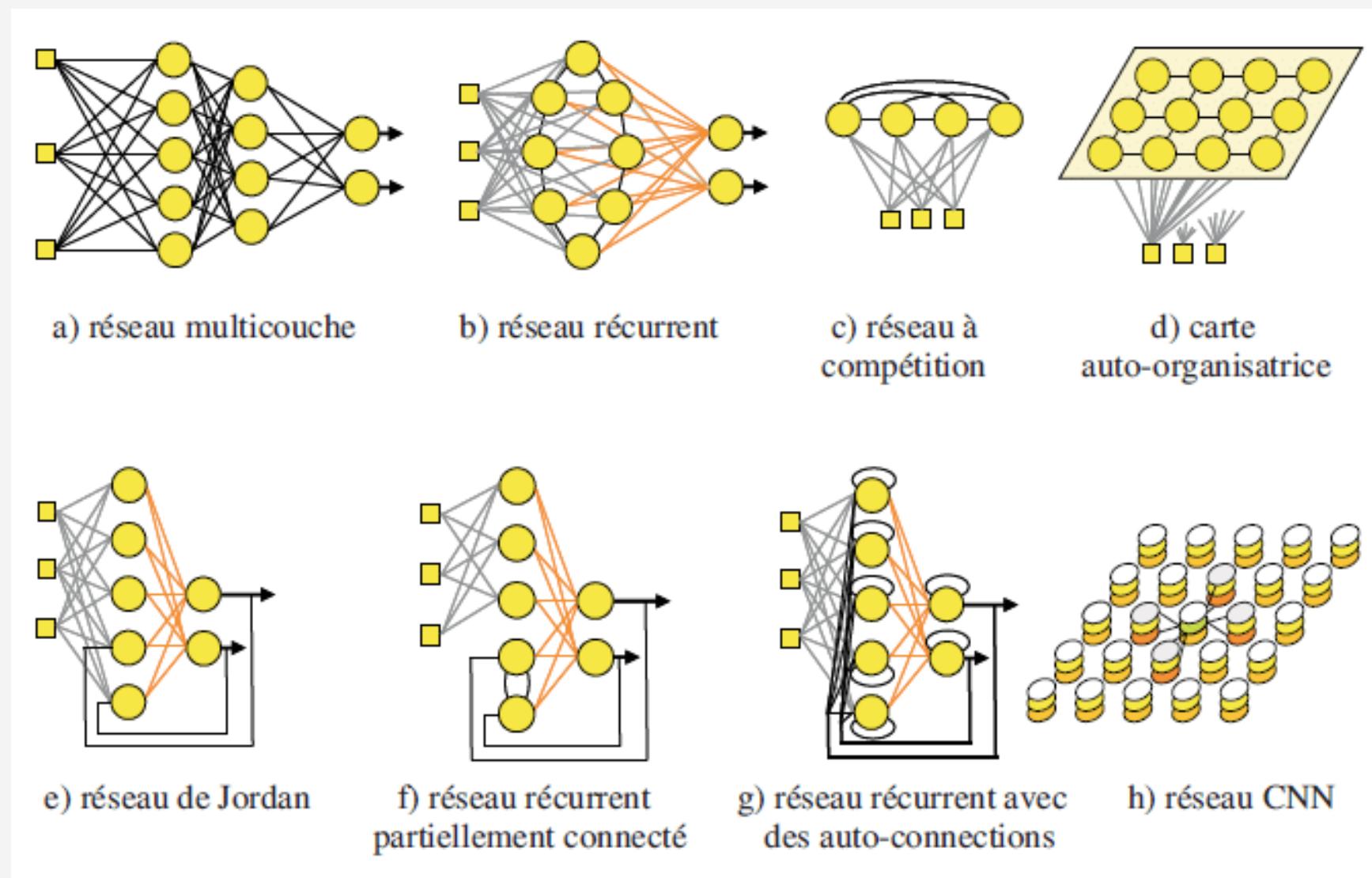
Apprentissage supervisé : Dans l'apprentissage supervisé, le réseau est entraîné sur un ensemble de données étiquetées, où chaque exemple d'entraînement est associé à une étiquette de sortie connue. Le réseau ajuste ses poids pour minimiser la différence entre ses prédictions et les étiquettes réelles.

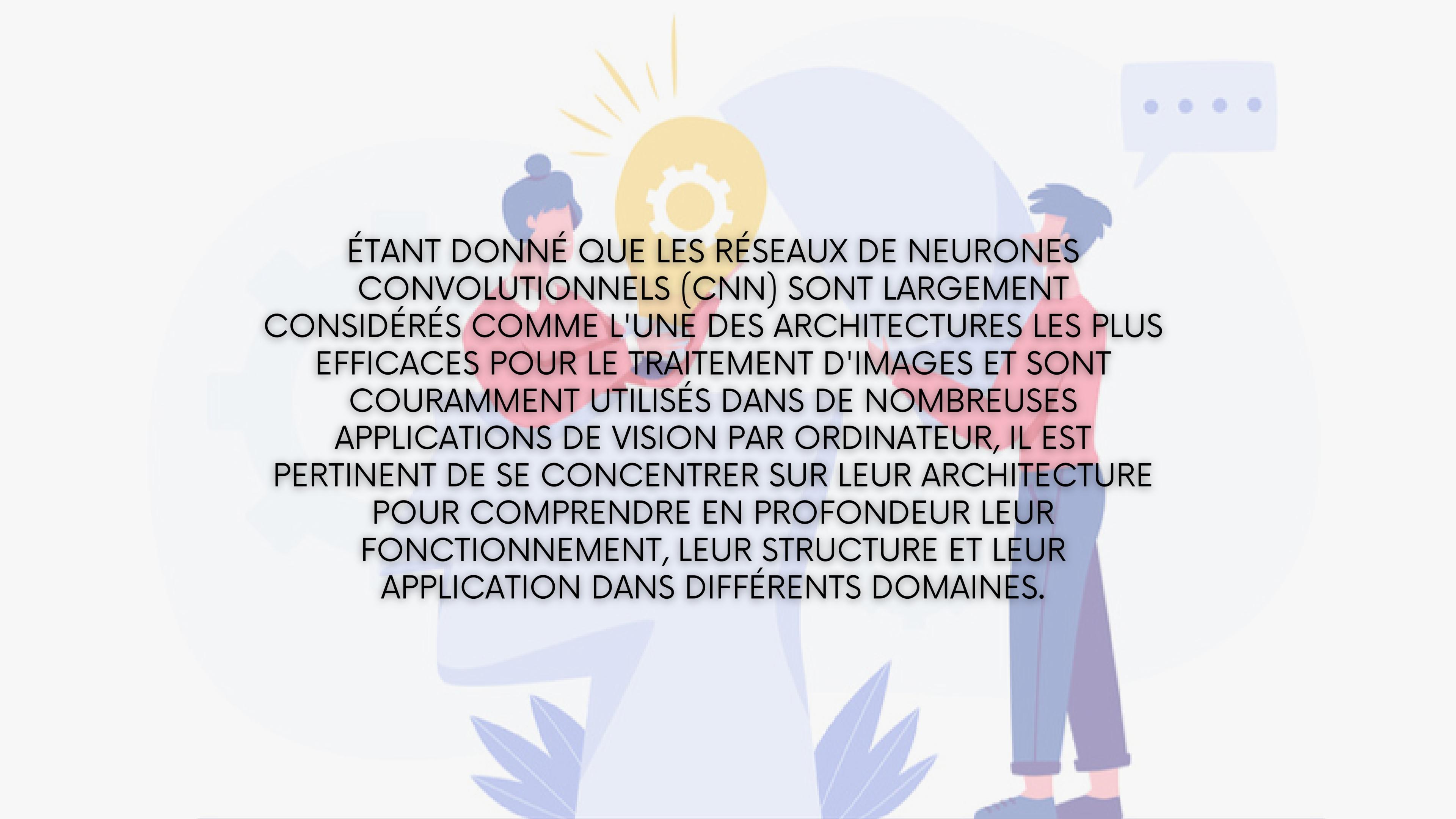
Apprentissage non supervisé : Contrairement à l'apprentissage supervisé, l'apprentissage non supervisé n'utilise pas d'étiquettes de sortie. Le réseau est entraîné sur un ensemble de données non étiquetées pour découvrir des structures ou des patterns intrinsèques dans les données, comme la segmentation, la réduction de dimensionnalité ou le clustering.

Apprentissage par renforcement : Dans l'apprentissage par renforcement, le réseau apprend à prendre des décisions séquentielles en interagissant avec un environnement dynamique. Il reçoit des récompenses ou des pénalités en fonction des actions qu'il prend, ce qui lui permet d'apprendre à maximiser sa récompense cumulative au fil du temps en utilisant des techniques telles que la méthode Q-learning ou les algorithmes de politique de gradient.

Architectures de réseau

Les architectures de réseau décrivent la disposition des neurones et des connexions dans un réseau de neurones. Il existe plusieurs architectures de réseau de neurones, notamment les réseaux de neurones multicouches (MLP), les réseaux de neurones récurrents (RNN), les autoencodeurs, les réseaux de neurones adversaires génératifs (GAN), les réseaux de neurones convolutionnels (CNN) qui est le plus utilisé, etc. Chaque architecture est adaptée à des types spécifiques de données et de tâches.



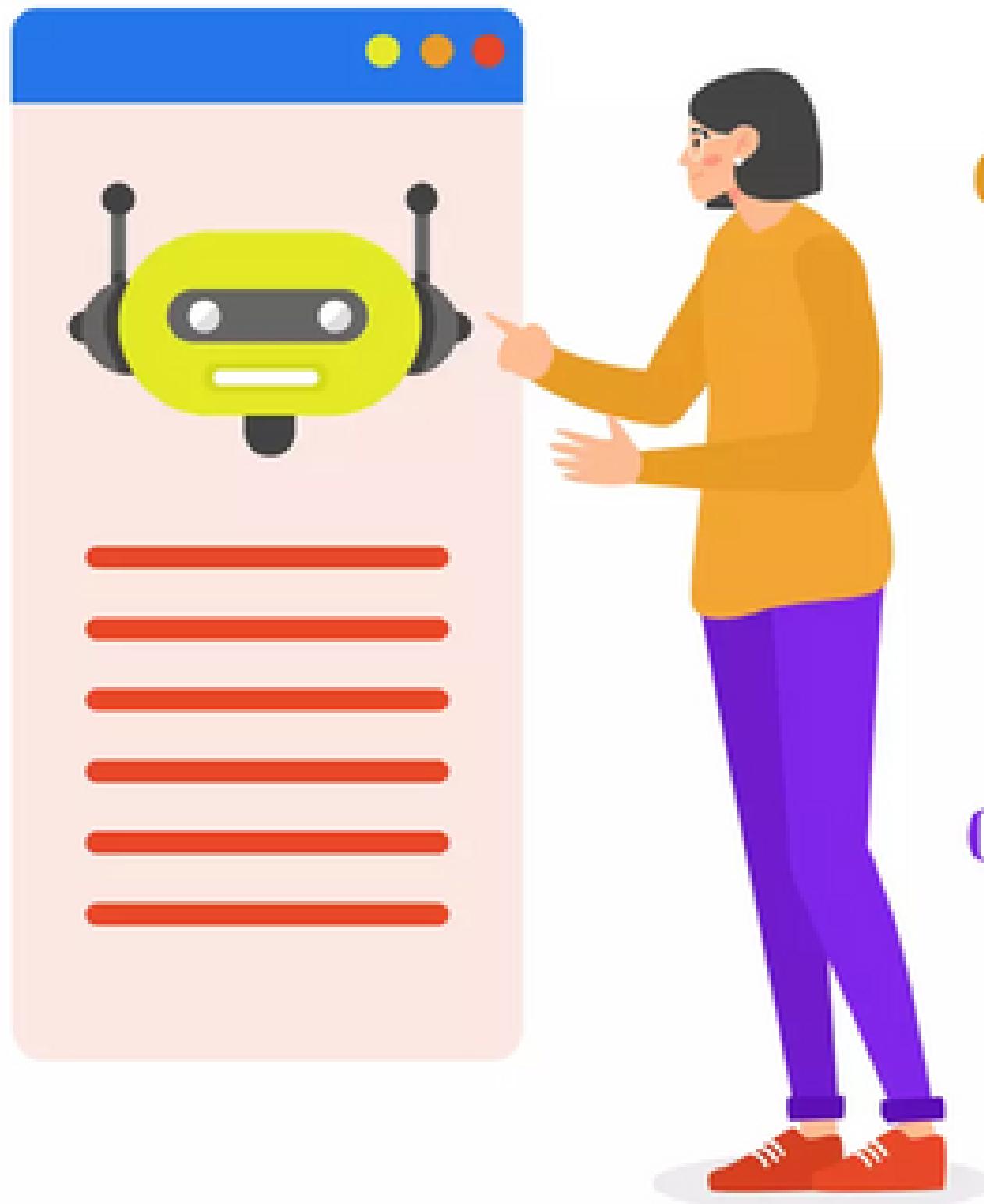


ÉTANT DONNÉ QUE LES RÉSEAUX DE NEURONES CONVOLUTIONNELS (CNN) SONT LARGEMENT CONSIDÉRÉS COMME L'UNE DES ARCHITECTURES LES PLUS EFFICACES POUR LE TRAITEMENT D'IMAGES ET SONT COURAMMENT UTILISÉS DANS DE NOMBREUSES APPLICATIONS DE VISION PAR ORDINATEUR, IL EST PERTINENT DE SE CONCENTRER SUR LEUR ARCHITECTURE POUR COMPRENDRE EN PROFONDEUR LEUR FONCTIONNEMENT, LEUR STRUCTURE ET LEUR APPLICATION DANS DIFFÉRENTS DOMAINES.

Qu'est ce qu'un réseau de neurones convolutive ?

01 Il s'agit d'une classe de Deep Learning.

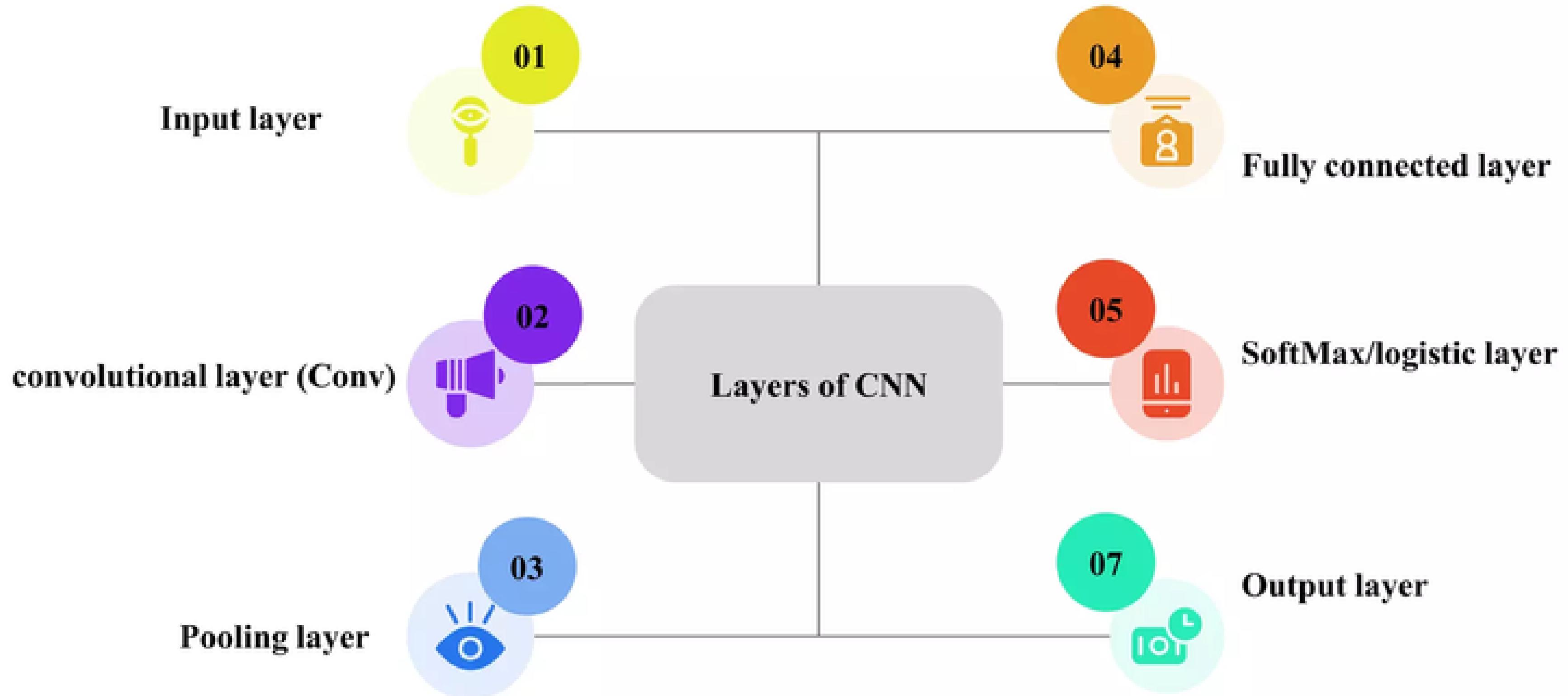
03 Les réseaux de neurones convolutifs sont une des catégories principales pour faire de la reconnaissance d'images, des classifications d'images, des détections d'objets, reconnaissance de visages, etc.



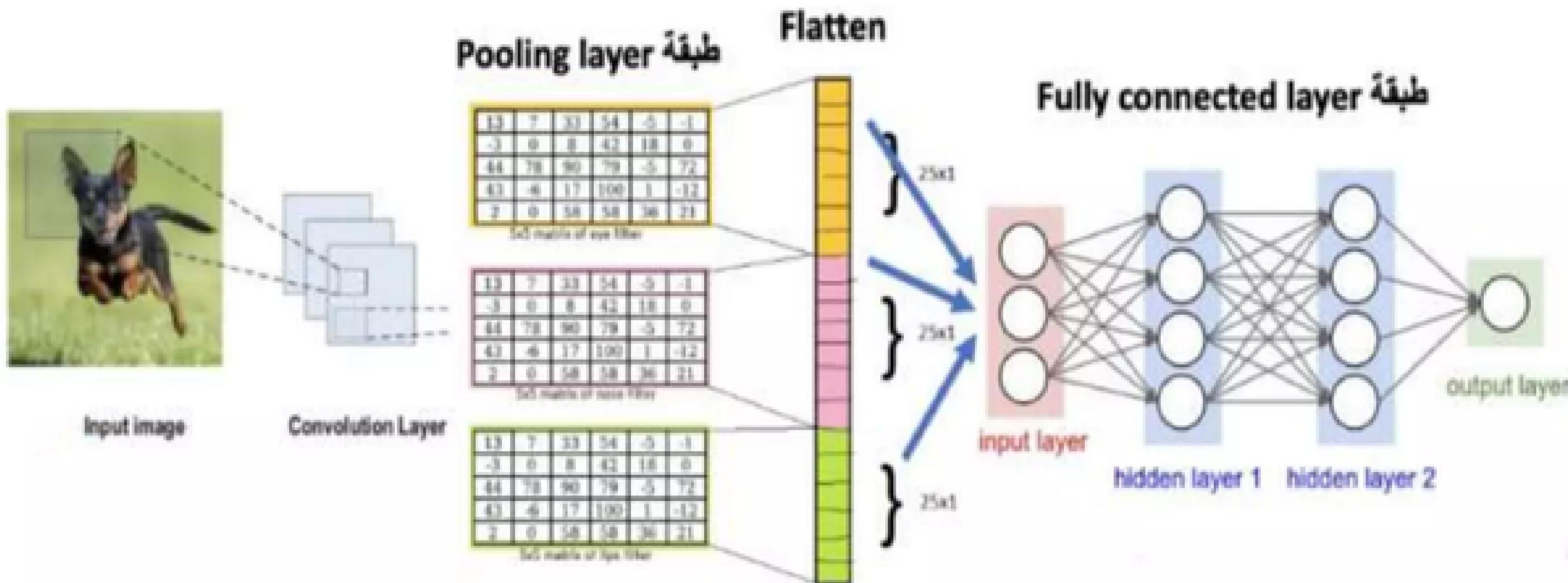
02 Il est similaire au réseau neuronal de base, le CNN possède également des paramètres d'apprentissage comme le réseau neuronal, c'est-à-dire des poids, des biais, etc.

04 Les 3 composants de base définissent le CNN :
-La couche de convolution
-La couche de mise en commun
-La couche de sortie (ou couche entièrement connectée)

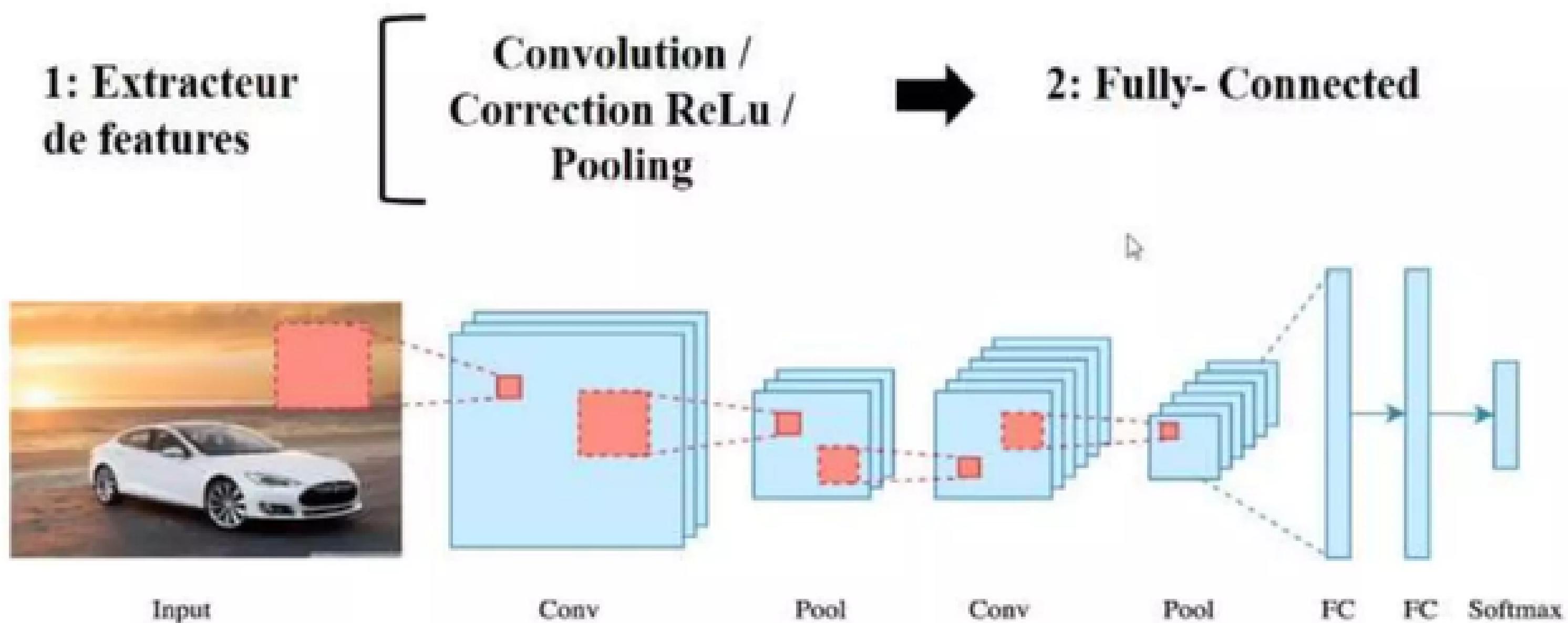
Les composants de base de couches du CNN :



Les couches de CNN :

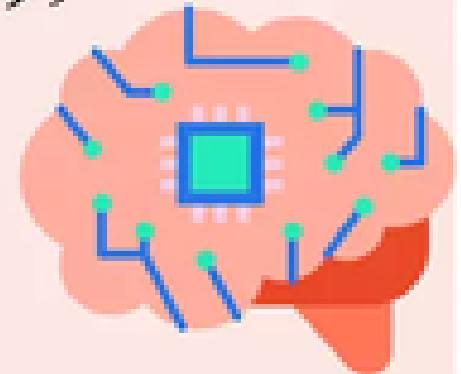


Les CNN sont spécialement conçus pour traiter des images en entrée



Couche d'entrée (Input Layer)

Couche d'entrée (Input Layer) : Cette couche reçoit les données d'entrée, généralement représentées sous forme d'images. Les images sont généralement représentées sous la forme d'un tenseur multidimensionnel, par exemple, pour des images en niveaux de gris de taille 28x28, le tenseur serait de dimension (28, 28, 1), où 1 représente le canal de couleur.



La couche convulsive

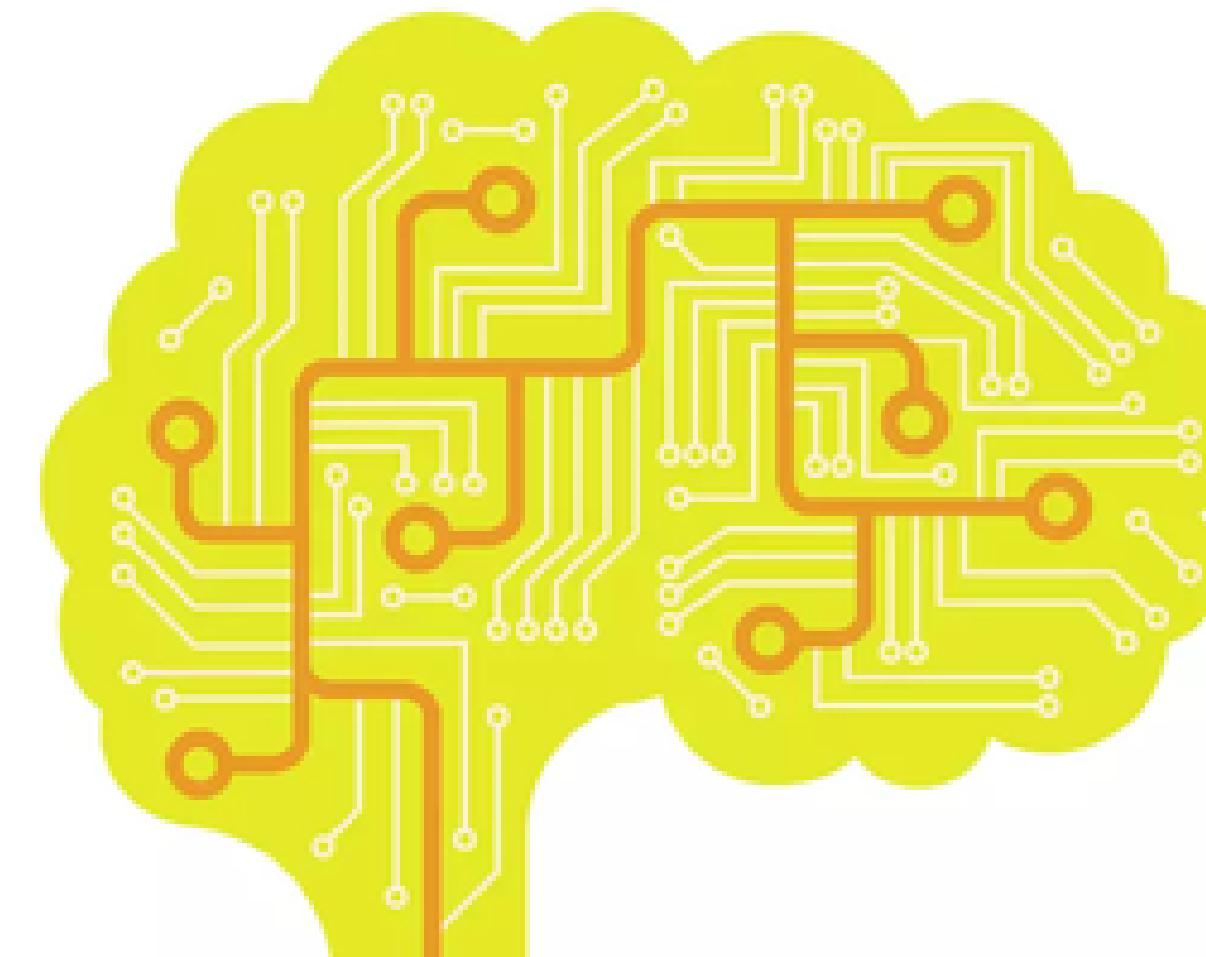
La convolution, d'un point de vue simpliste, est le fait d'appliquer un filtre mathématique à une image. D'un point de vue plus technique, il s'agit de faire glisser une matrice par-dessus une image, et pour chaque pixel, utiliser la somme de la multiplication de ce pixel par la valeur de la matrice. Cette technique nous permet de trouver des parties de l'image qui pourraient nous être intéressantes. Prenons la Figure ci-dessous à gauche comme exemple d'image et la Figure à droite comme exemple de

5	2	1	3	5
4	3	2	3	4
0	2	1	2	2
3	3	2	3	2
M	2	1	3	2

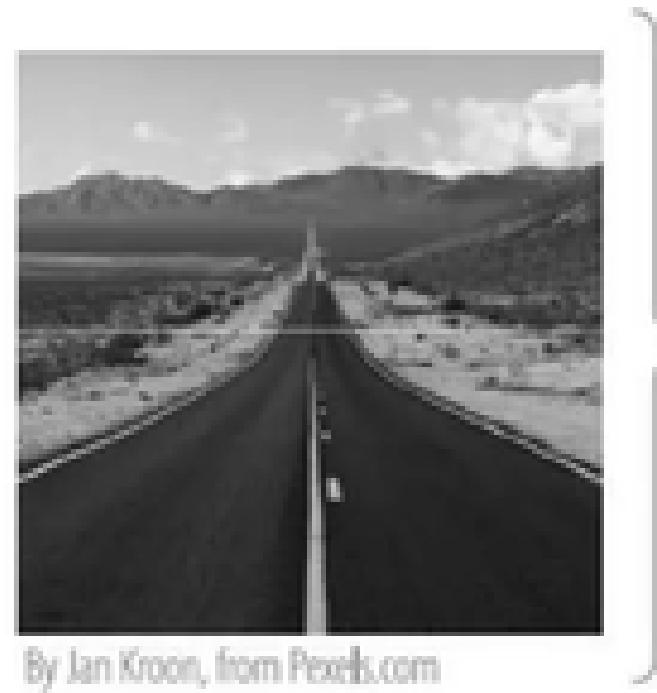
Exemple simpliste des valeurs des pixels d'une image 5x5

1	0	1
0	1	0
1	0	1

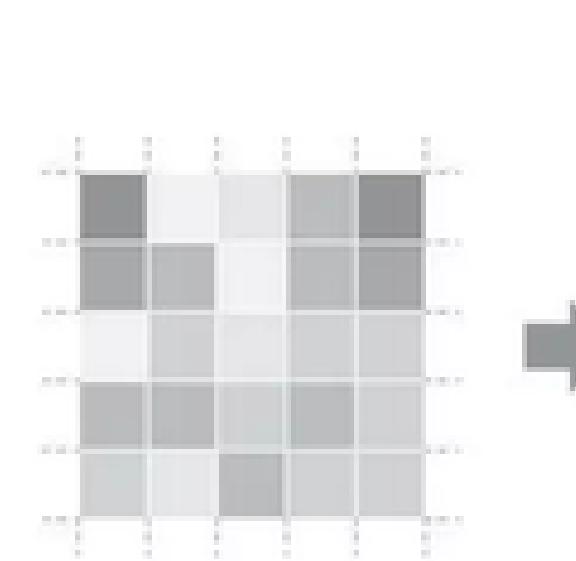
Exemple de valeurs d'une matrice utilisée comme filtre



Appliquer le filtre sur l'image :



2D convolution



5	2	1	3	5
4	3	2	3	4
0	2	1	2	2
3	3	2	3	2
2	1	3	2	2

Image piece

5	2	1
4	3	2
0	2	1

x

Kernel 3x3

1	0	1
0	1	0
1	0	1

$$= \boxed{10}$$

y

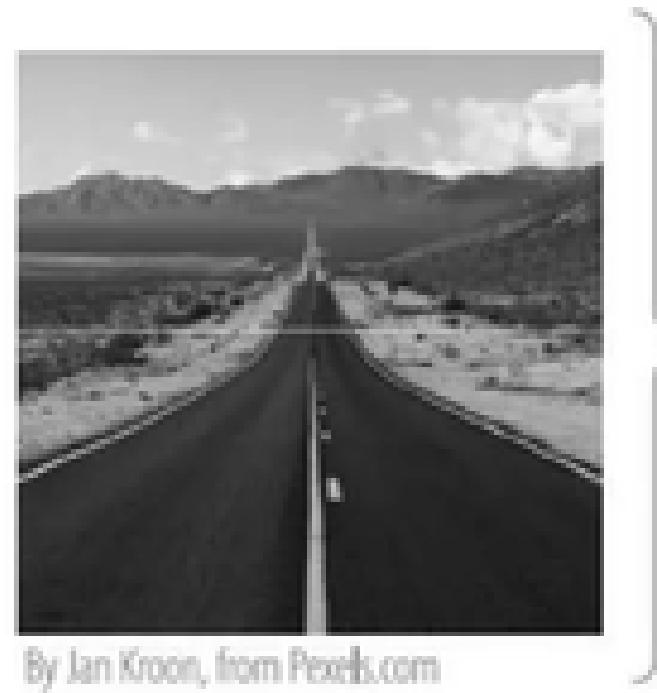
$$\begin{aligned} y &= 5 \times 1 + 2 \times 0 + 1 \times 1 \\ &+ 4 \times 0 + 3 \times 1 + 2 \times 0 \\ &+ 0 \times 1 + 2 \times 0 + 1 \times 1 = 10 \end{aligned}$$

1	0	1
0	1	0
1	0	1

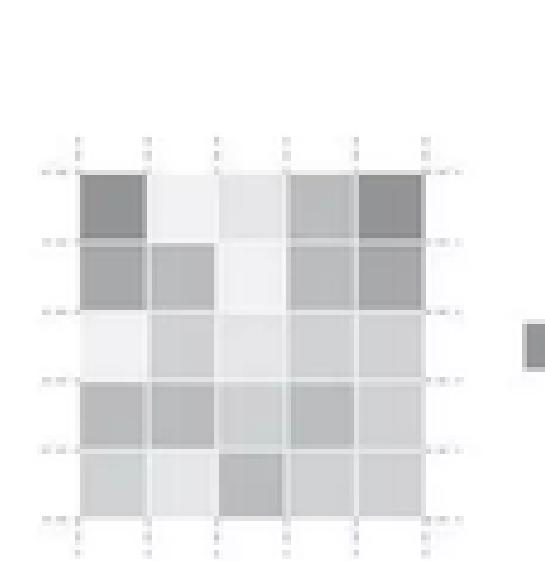
$$y = \sum_{i=1}^n \sum_{j=1}^m x_{i,j} \cdot w_{i,j} \quad \text{with} \quad \begin{cases} n & \text{kernel width} \\ m & \text{kernel height} \end{cases}$$

\otimes Hadamard product

Appliquer le filtre sur l'image :



2D convolution



5	2	1	3	5
4	3	2	3	4
0	2	1	2	2
3	3	2	3	2
2	1	3	2	2

Image piece

5	2	1
4	3	2
0	2	1

x

Kernel 3x3

1	0	1
0	1	0
1	0	1

w

$$= \boxed{10}$$

y

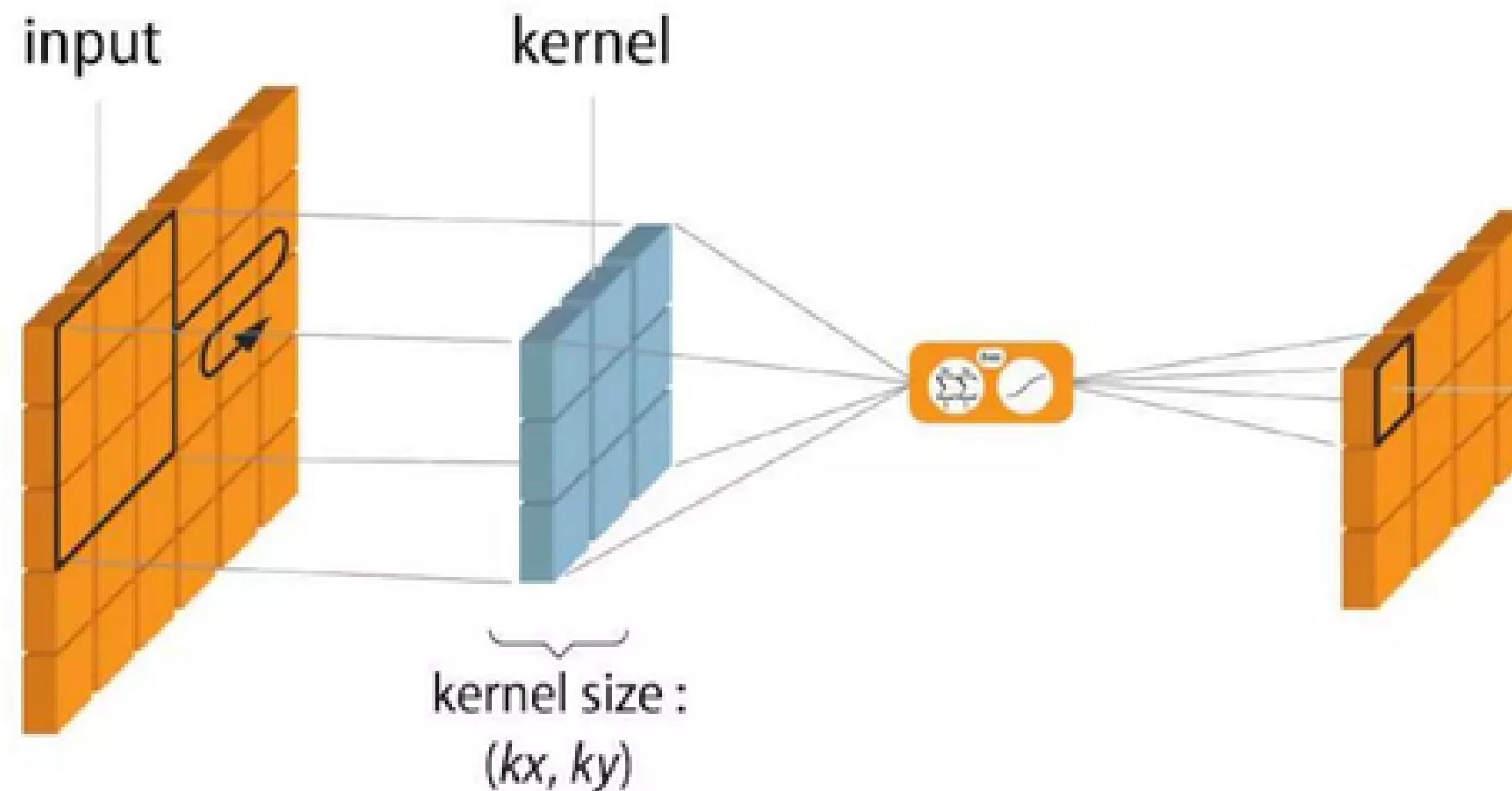
$$\begin{aligned} y &= 5 \times 1 + 2 \times 0 + 1 \times 1 \\ &+ 4 \times 0 + 3 \times 1 + 2 \times 0 \\ &+ 0 \times 1 + 2 \times 0 + 1 \times 1 = 10 \end{aligned}$$

1	0	1
0	1	0
1	0	1

$$y = \sum_{i=1}^n \sum_{j=1}^m x_{i,j} \cdot w_{i,j} \quad \text{with} \quad \begin{cases} n & \text{kernel width} \\ m & \text{kernel height} \end{cases}$$

\otimes Hadamard product

couches de convolution

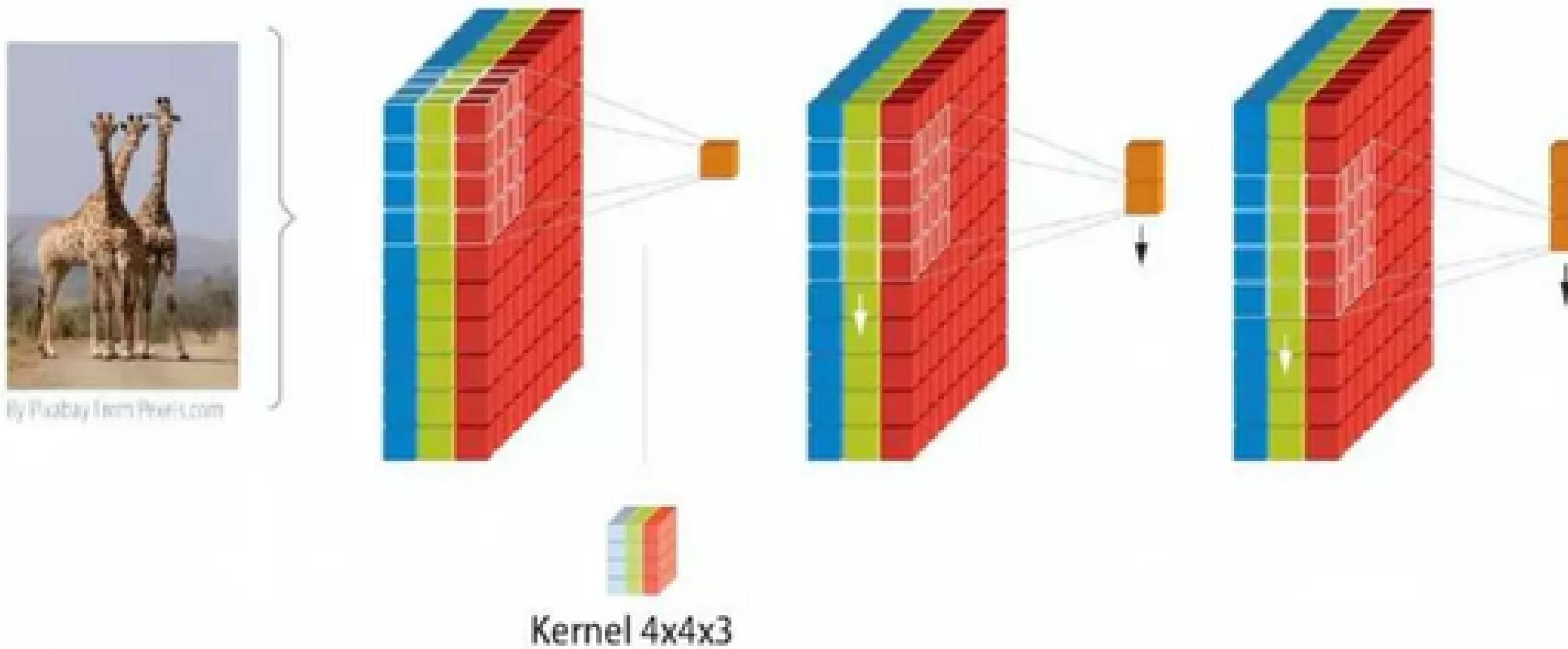


$$y = \sigma(\sum \text{input} \otimes \text{kernel} + b)$$

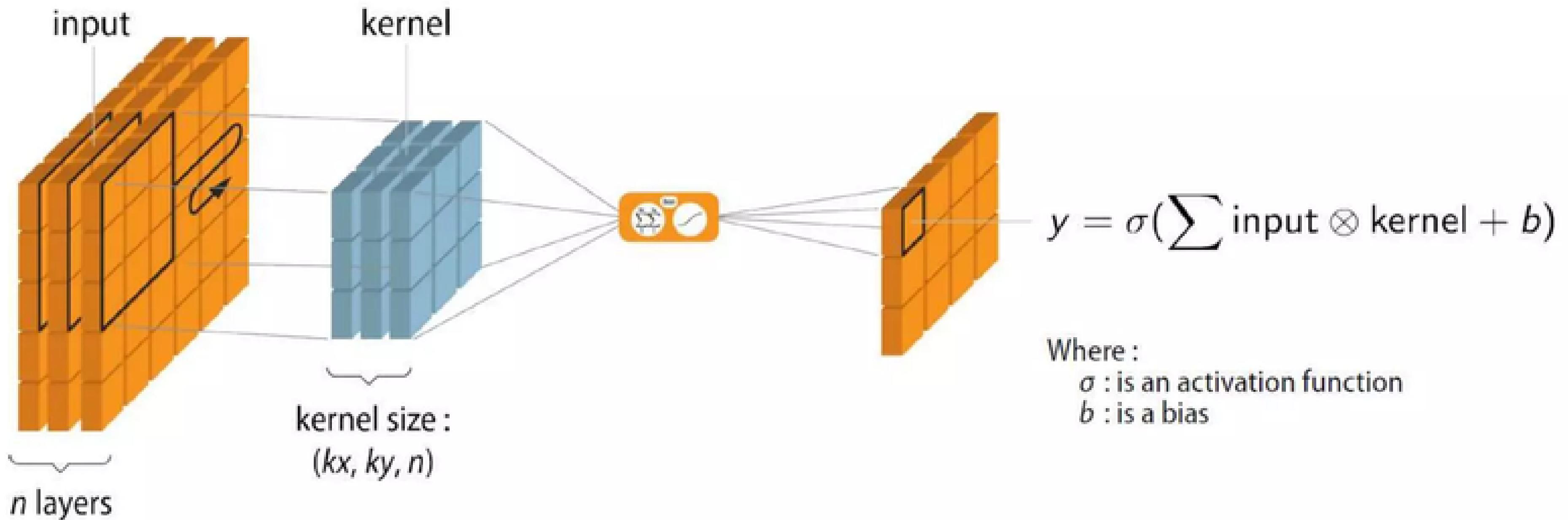
Where :

σ : is an activation function
 b : is a bias

Number of parameters for a convolutional layer : $k_x \cdot k_y \cdot n_z + 1$



3D convolution



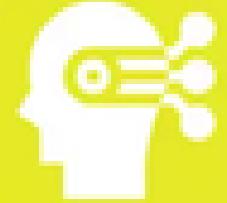
If we want to generate m convolutional layers, we will need m convolutional neurons

Padding et Stride



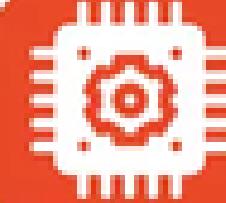
01 **Stride**

02 **Padding**



Stride (pas) :

Le stride fait référence à la taille du pas utilisé lorsque le filtre de convolution se déplace à travers les données d'entrée ou les cartes de caractéristiques. Il détermine la quantité de déplacement spatial entre les applications successives du filtre. Un stride de 1 signifie que le filtre se déplace d'un pixel à la fois. Un stride de 2 signifie qu'il se déplace de deux pixels à la fois, et ainsi de suite. Un stride plus élevé réduit la dimension spatiale de la sortie.

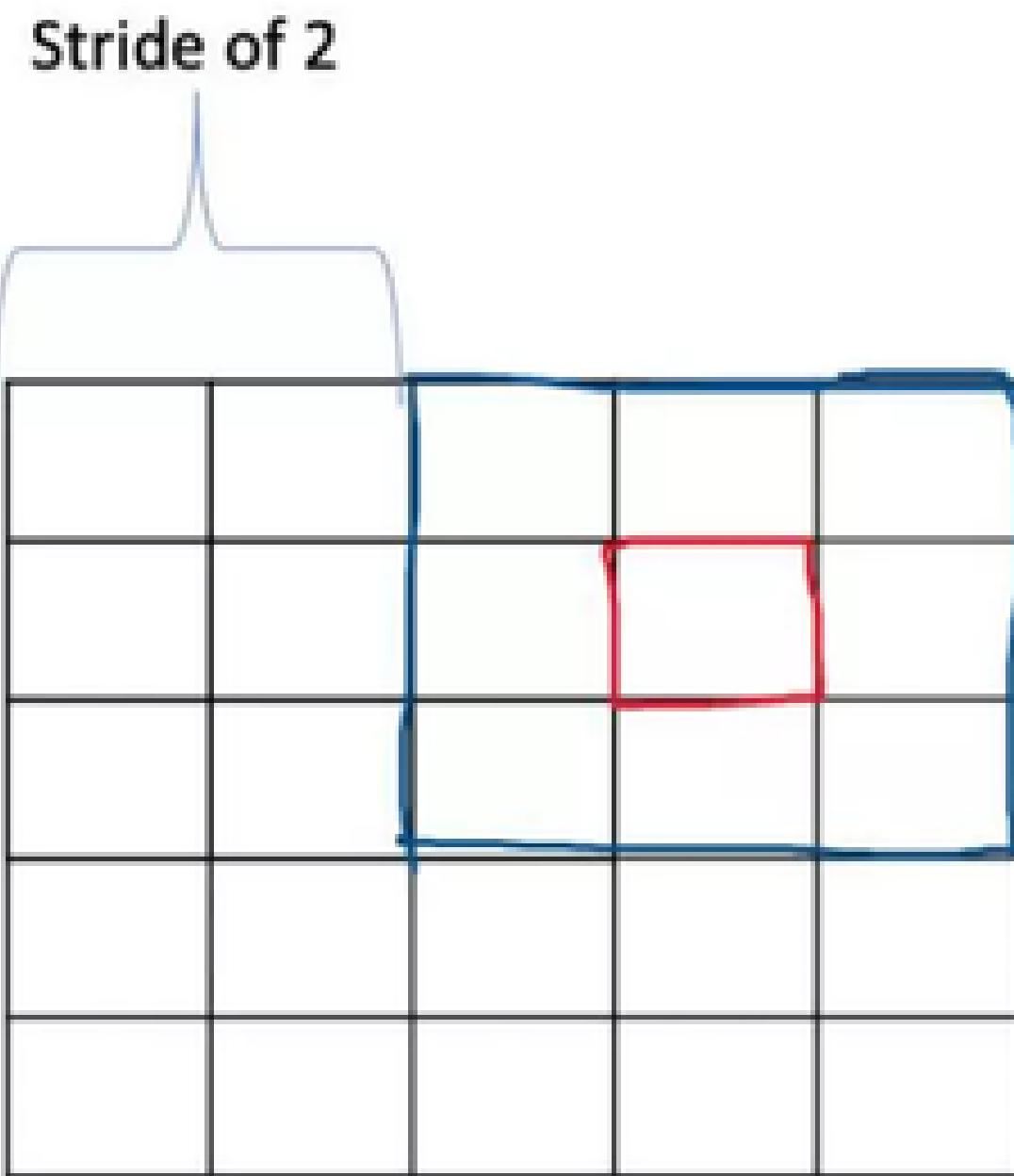


Padding (Remplissage) :

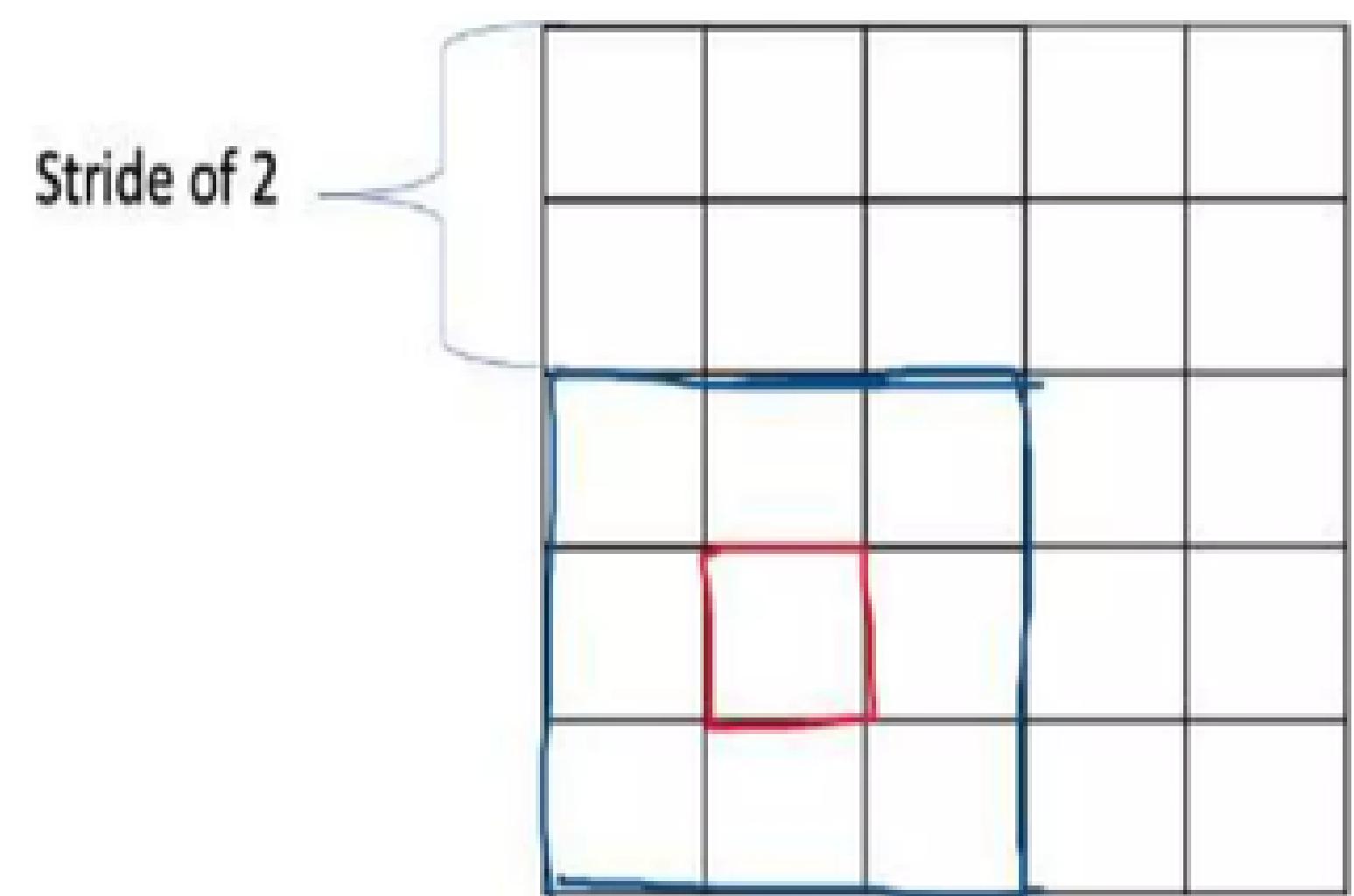
Le padding fait référence à l'ajout de zéros autour des bords de l'image ou des cartes de caractéristiques avant d'appliquer la convolution. Il est utilisé pour préserver la taille spatiale de la sortie et éviter une réduction excessive de la dimensionnalité. Le padding permet de conserver les informations sur les bords de l'image et peut être utile pour maintenir la résolution spatiale ou pour améliorer la performance du modèle. Deux types courants de padding sont le "valid padding" (sans remplissage) où aucun zéro n'est ajouté et le "same padding" où suffisamment de zéros sont ajoutés pour que la taille de l'entrée et de la sortie soit la même.

&

Exemple de Stride



stride = (2,2) à droite

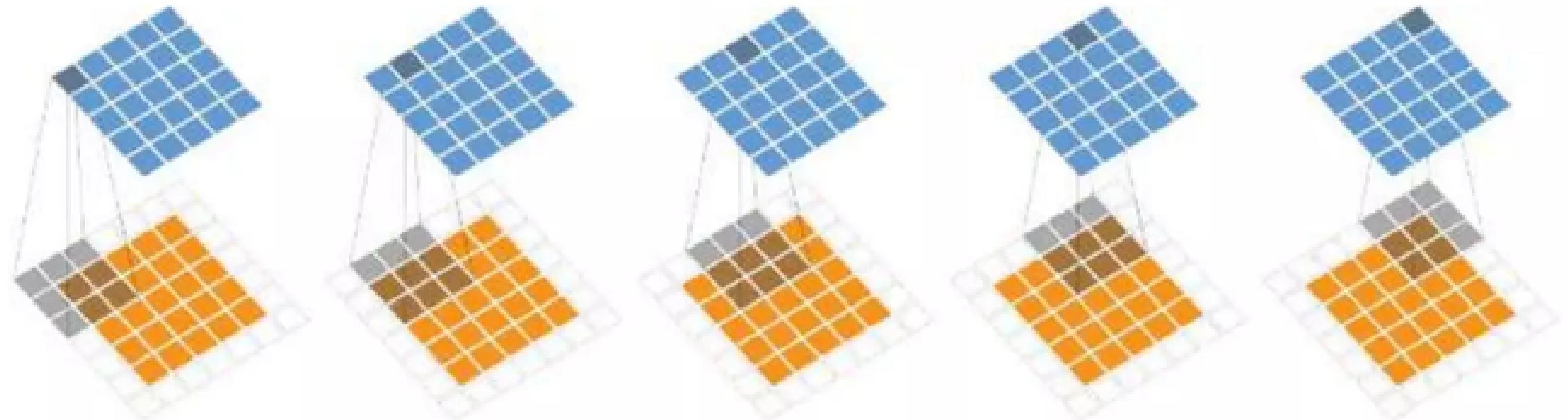


stride = (2,2) en bas

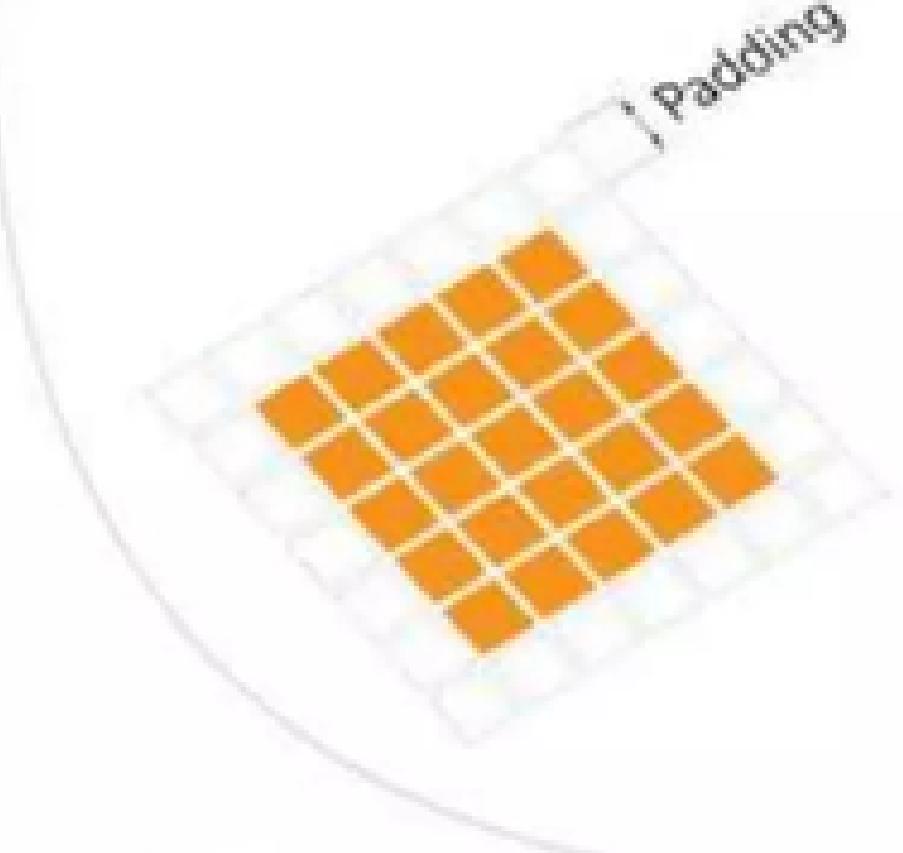
Exemple de Padding

0	0	0	0	0	0
					0
				0	
					0
					0
					0

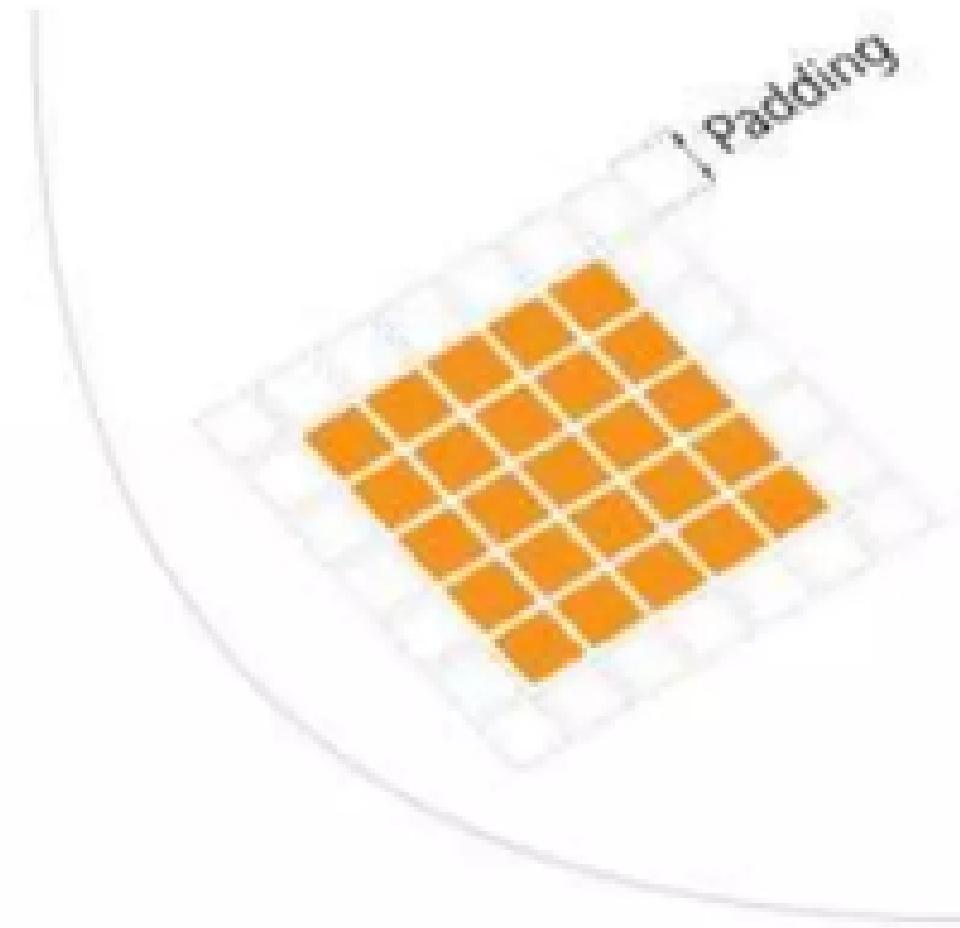
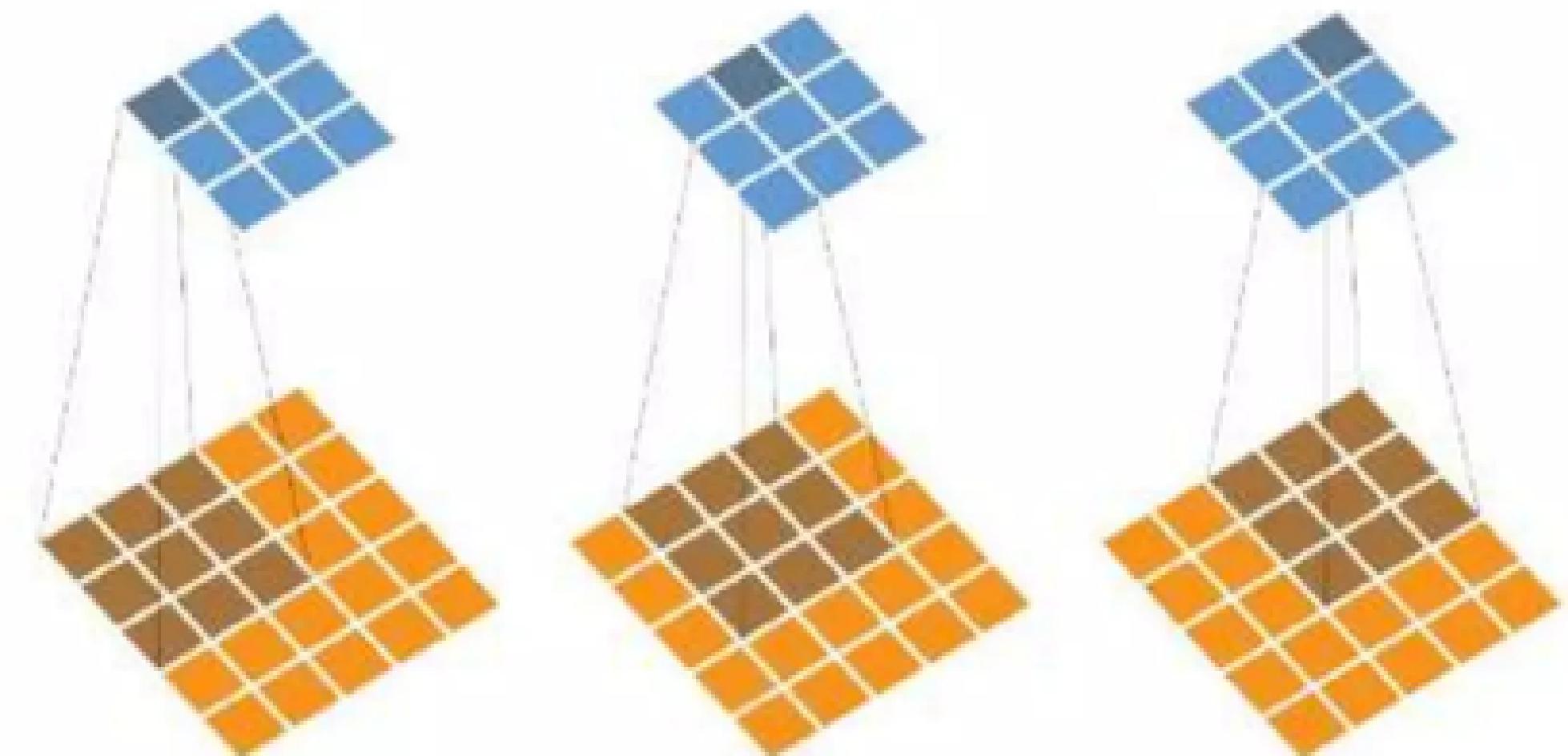
Parameters of a convolutional layer : padding



padding = 'same' \Rightarrow size is preserved



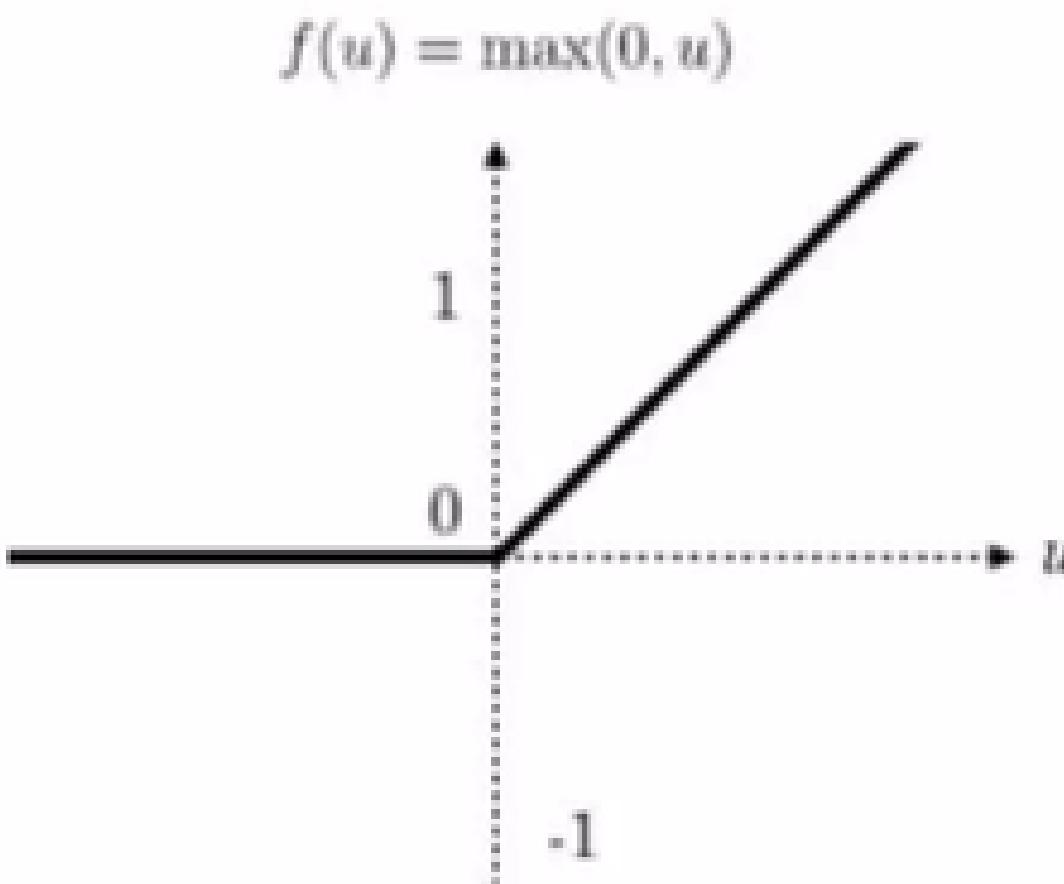
Parameters of a convolutional layer : padding



padding = 'valid' \Rightarrow Size is not preserved (no padding)

ReLU – Unité linéaire rectifiée

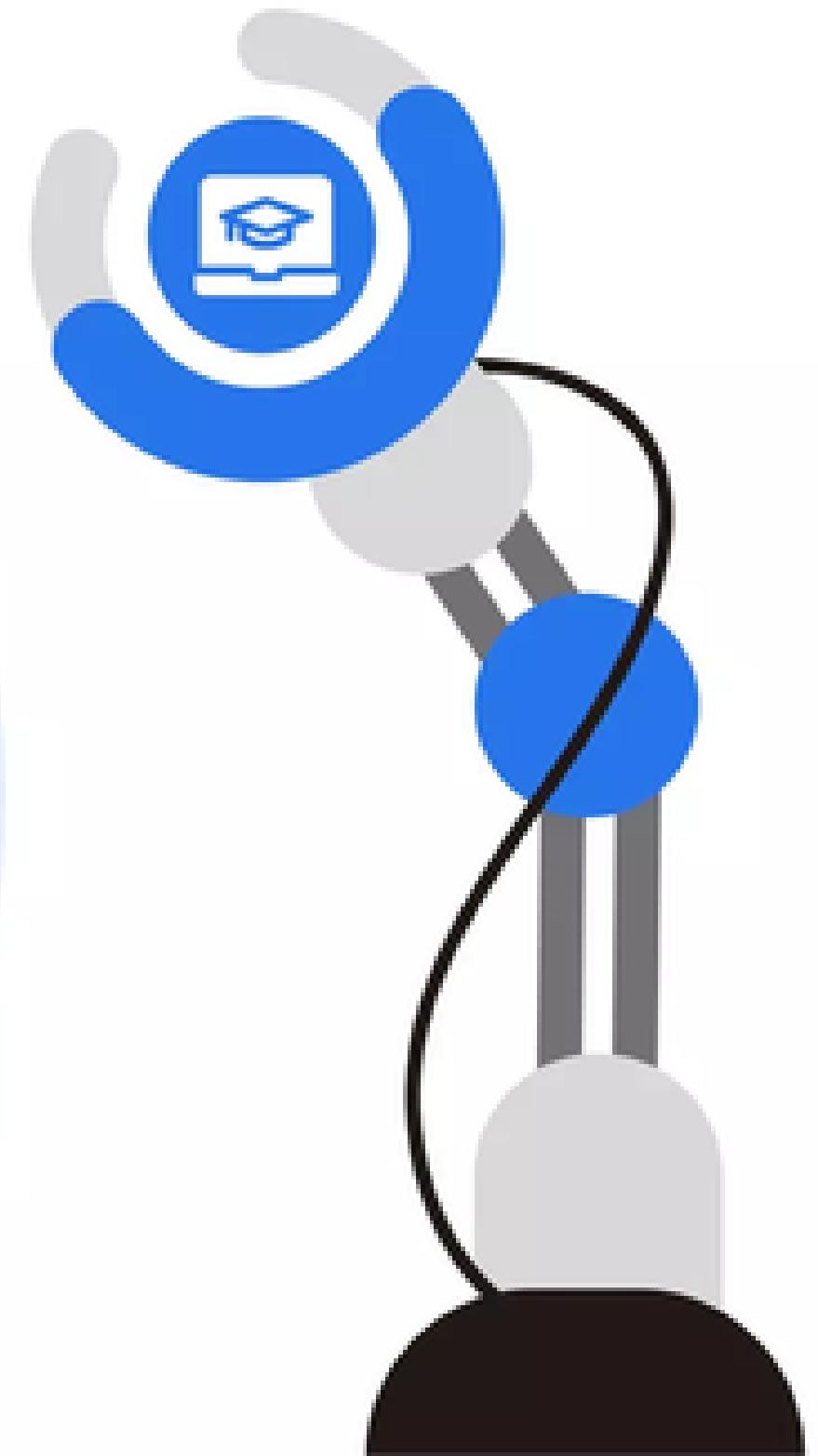
ReLU est une fonction qui doit être appliquée à chaque pixel d'une image après convolution, et remplace chaque valeur négative par un 0. Si cette fonction n'est pas appliquée, la fonction créée sera linéaire et le problème XOR persiste puisque dans la couche de convolution, aucune fonction d'activation n'est appliquée.



Comment choisir les features ?

Contrairement aux méthodes traditionnelles, les features ne sont pas pré-définies selon un formalisme particulier mais apprises par le réseau lors la phase d'entraînement ! Les noyaux des filtres désignent les poids de la couche de convolution. Ils sont initialisés puis mis à jour par rétropropagation du gradient.

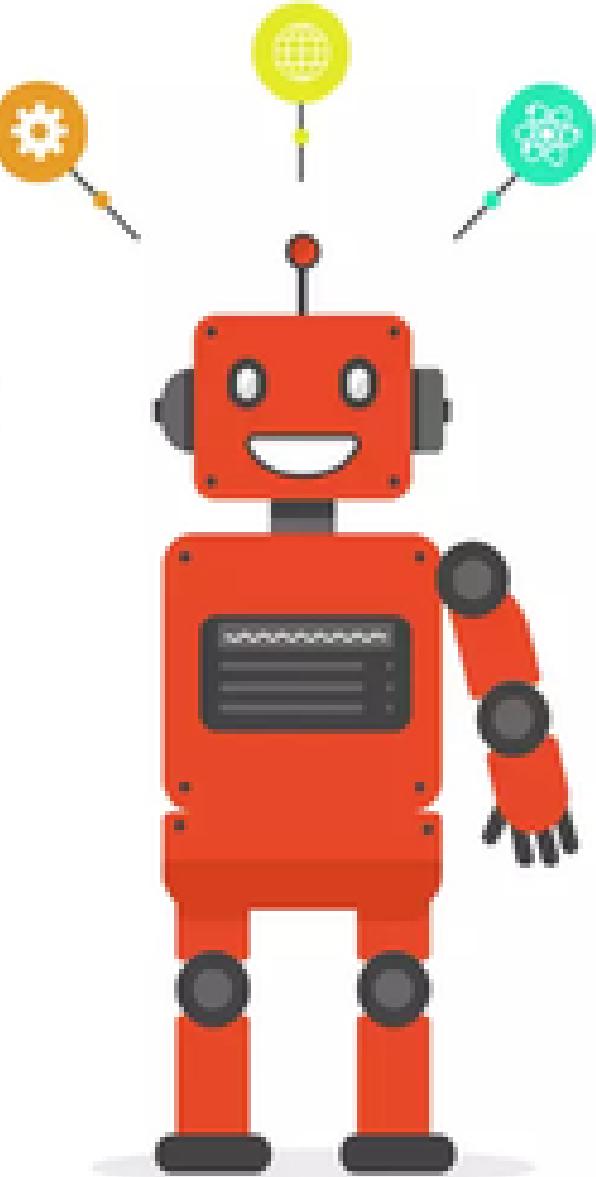
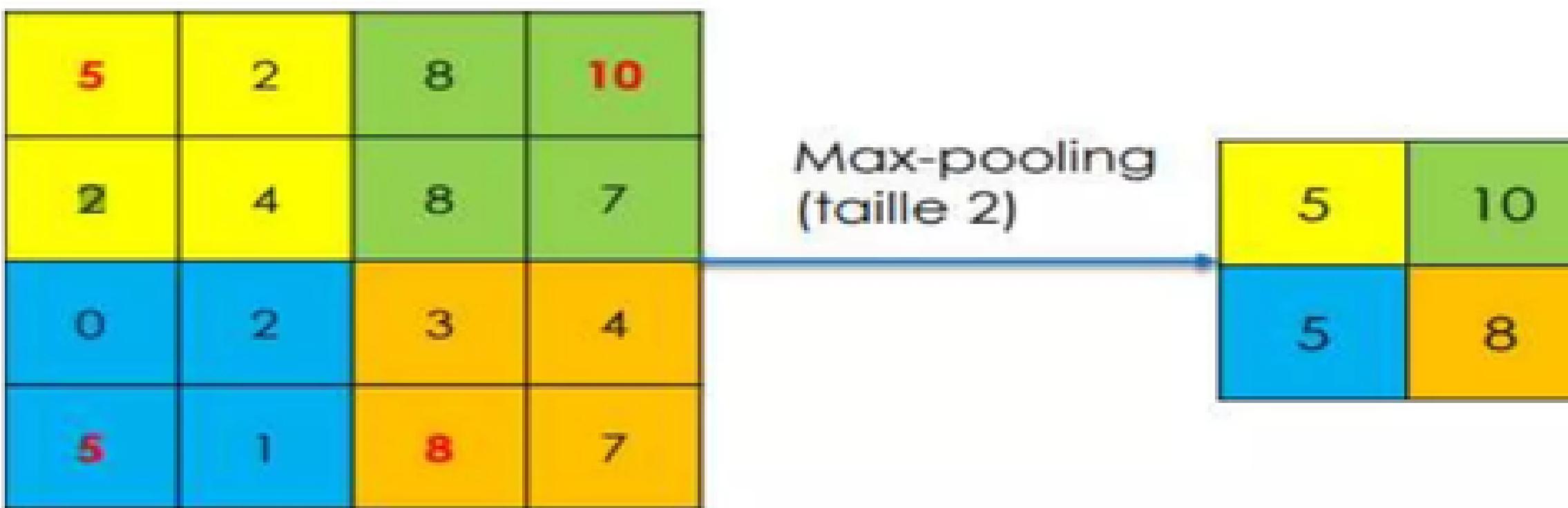
C'est là toute la force des réseaux de neurones convolutifs : ceux-ci sont capables de déterminer tout seul les éléments discriminants d'une image, en s'adaptant au problème posé. Par exemple, si la question est de distinguer les chats des chiens, les features automatiquement définies peuvent décrire la forme des oreilles ou des pattes.



CNN: Pooling

Le Pooling : opération utilisée pour réduire la dimension

- ❖ Recherche de détails plus « grossiers », de plus grandes « structures » dans l'image
- ❖ Max-pooling de taille l : on prend l'élément maximal de chaque sous-tableau de taille l
- ❖ Sum-pooling de taille l : on fait la somme de tous les éléments de chaque sous tableau de taille l



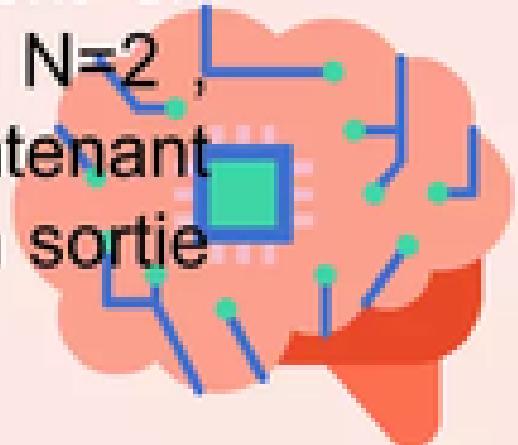
Couche fully-connected

La couche fully-connected constitue toujours la dernière couche d'un réseau de neurones. Ce type de couche reçoit un vecteur en entrée et produit un nouveau vecteur en sortie. Pour cela, elle applique une combinaison linéaire puis éventuellement une fonction d'activation aux valeurs reçues en entrée.

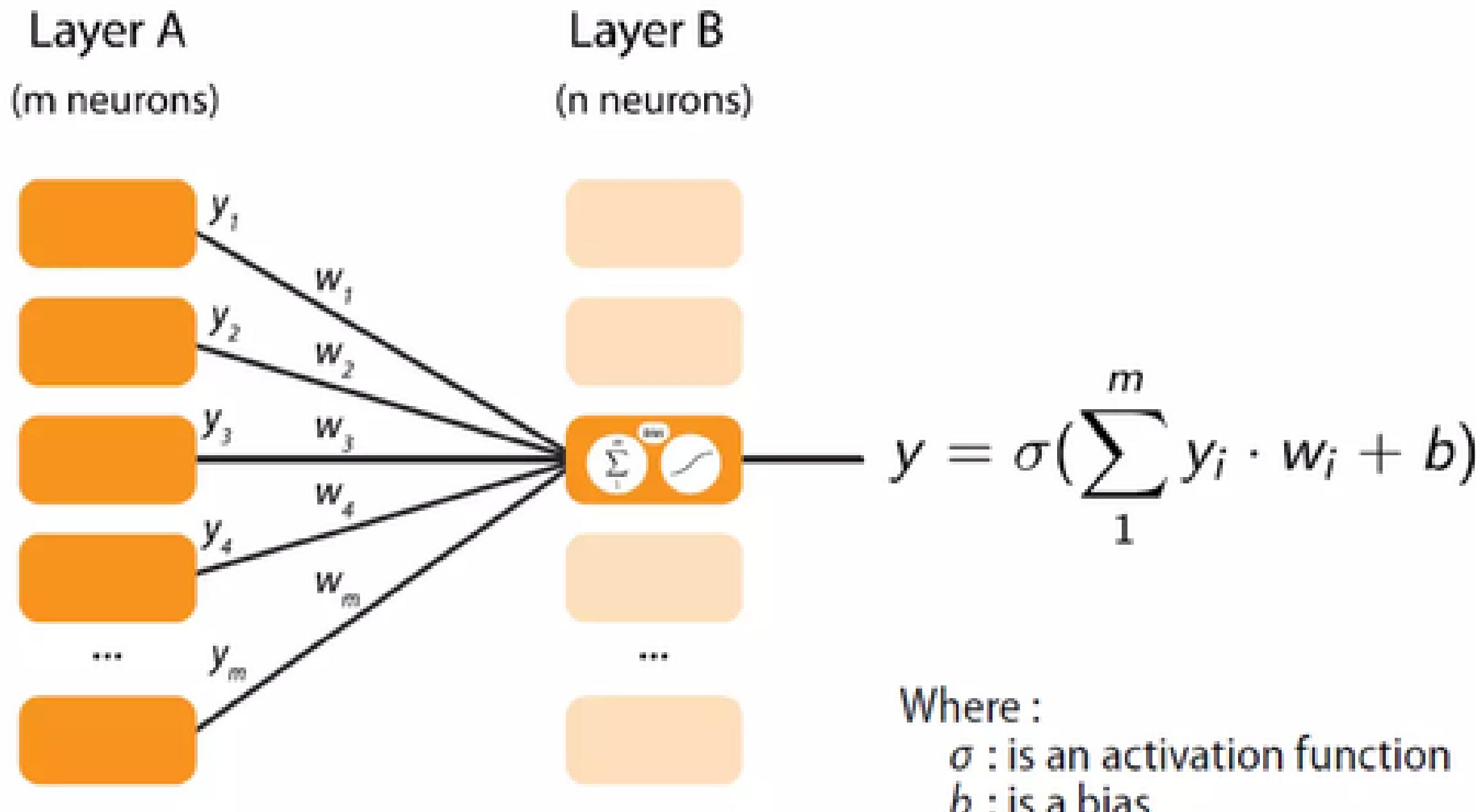
La dernière couche fully-connected permet de classifier l'image en entrée du réseau : elle renvoie un vecteur de taille N , où N est le nombre de classes dans notre problème de classification d'images. Chaque élément du vecteur indique la probabilité pour l'image en entrée d'appartenir à une classe.

Par exemple, si le problème consiste à distinguer les chats des chiens, le vecteur final sera de taille 2 : le premier élément (respectivement, le deuxième) donne la probabilité d'appartenir à la classe "chat" (respectivement "chien"). Ainsi, le vecteur [0.9 0.1] signifie que l'image a 90% de chances de représenter un chat.

Pour calculer les probabilités, la couche fully-connected multiplie donc chaque élément en entrée par un poids, fait la somme, puis applique une fonction d'activation (logistique si $N=2$, softmax si $N>2$) : Ce traitement revient à multiplier le vecteur en entrée par la matrice contenant les poids. Le fait que chaque valeur en entrée soit connectée avec toutes les valeurs en sortie explique le terme fully-connected.

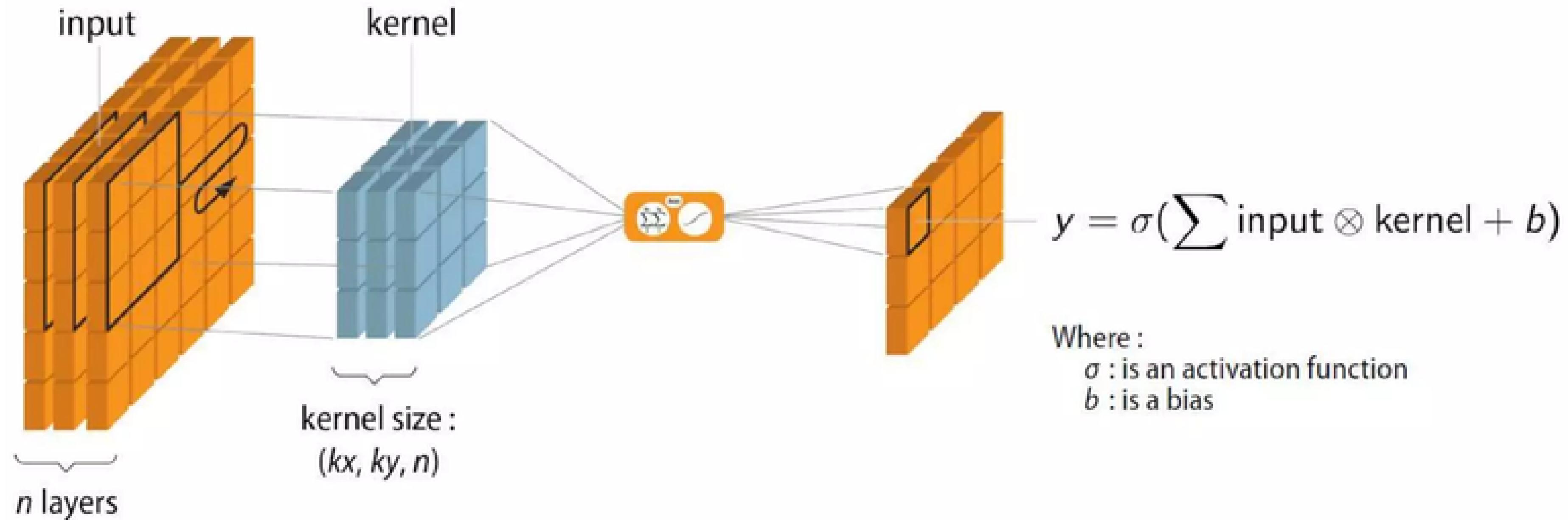


For a fully connected layer :



Number of parameters for a DNN layer : $n(m + 1)$

For a convolutional layer :



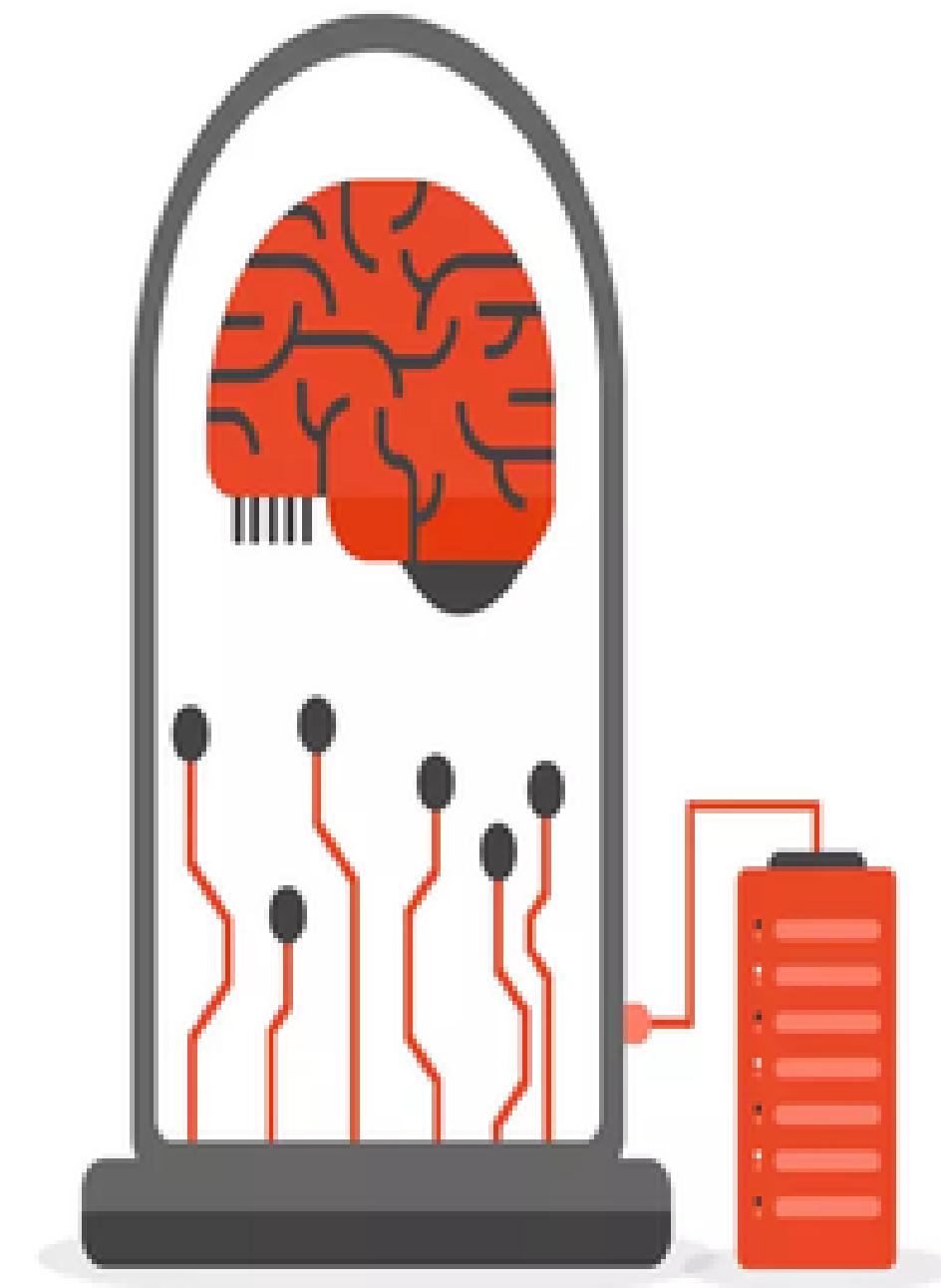
Number of parameters for a convolutional layer : $n.kx.Ky + 1$

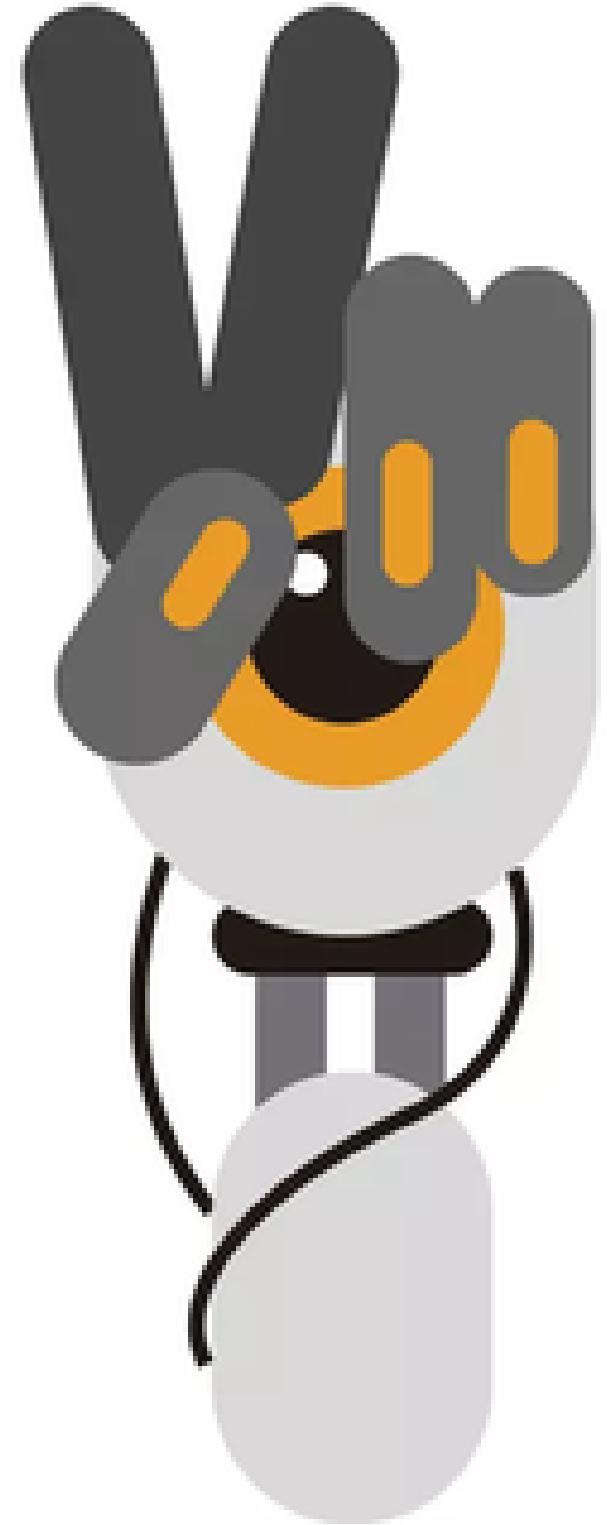
If we want to generate m convolutional layers, we will need m convolutional neurons, so, number of parameters is : $m.(n.kx.ky + 1)$

Architecture d'un CNN

Un réseau de neurones à convolution peut avoir plusieurs étapes de convolution, ReLu et Pooling. Une règle à respecter est que la fonction de ReLu doit obligatoirement être appliquée après une étape de convolution afin d'avoir une réponse non-linéaire, mais le Pooling n'est pas obligatoire.

Après être passé par toutes les étapes de convolution, ReLu et Pooling, nous pouvons passer à la classification des images. La dernière phase consiste à envoyer tous les pixels dans un réseau de neurones multicouches. Étant donné que nous avons pu récupérer les parties les plus importantes d'une image que nous avons condensée, la phase de classification sera beaucoup plus performante qu'en utilisant un réseau de neurones artificiels sans convolution.

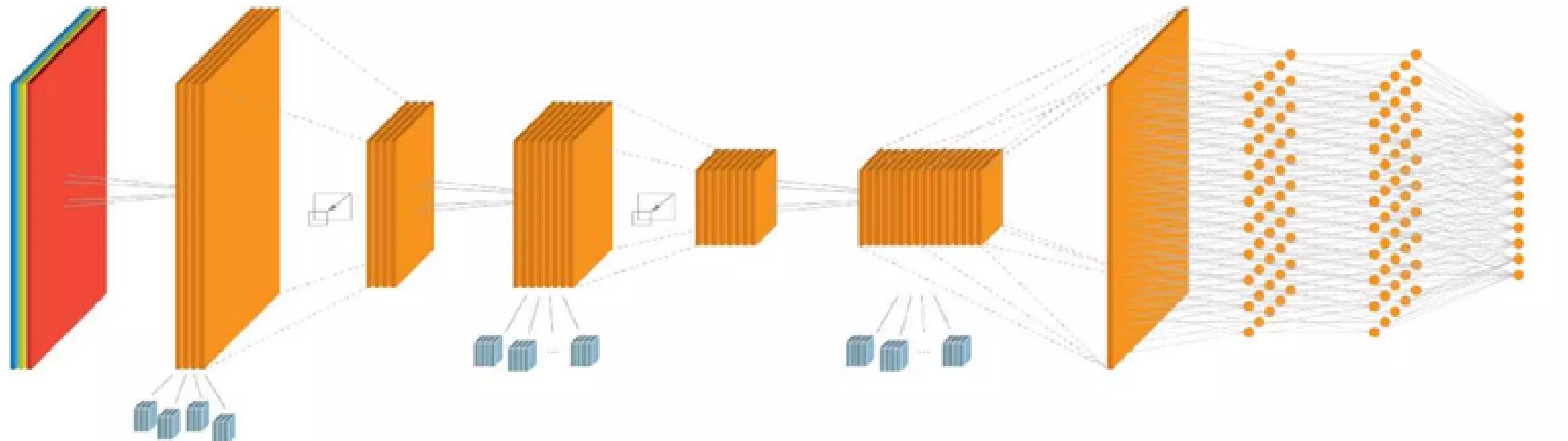




Un CNN est simplement un empilement de plusieurs couches de convolution, pooling, correction ReLU et fully-connected. Chaque image reçue en entrée va donc être filtrée, réduite et corrigée plusieurs fois, pour finalement former un vecteur. Dans le problème de classification, ce vecteur contient les probabilités d'appartenance aux classes.

Tous les réseaux de neurones convolutifs doivent commencer par une couche de convolution et finir par une couche fully-connected. Les couches intermédiaires peuvent s'empiler de différentes manières, à condition que la sortie d'une couche ait la même structure que l'entrée de la suivante. Par exemple, une couche fully-connected, qui renvoie toujours un vecteur, ne peut pas être placée avant une couche de pooling, puisque cette dernière doit recevoir une matrice 3D.

En général, un réseau de neurones empile plusieurs couches de convolution et de correction ReLU, ajoute ensuite une couche de pooling (facultative), et répète ce motif plusieurs fois ; puis, il empile des couches fully-connected. Plus il y a de couches, plus le réseau de neurones est "profond" : on est en plein dans le **Deep Learning** !



Input
layer

Convolutional
layers

Pooling

Convolutional
layers

Pooling

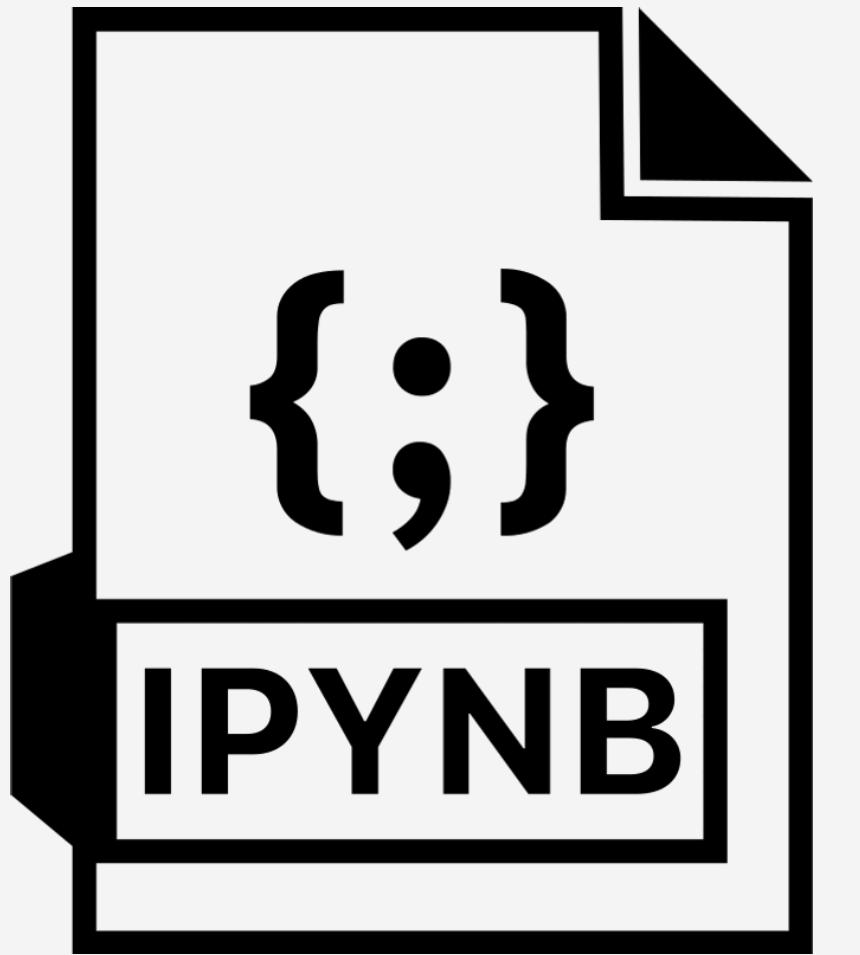
Convolutional
layers

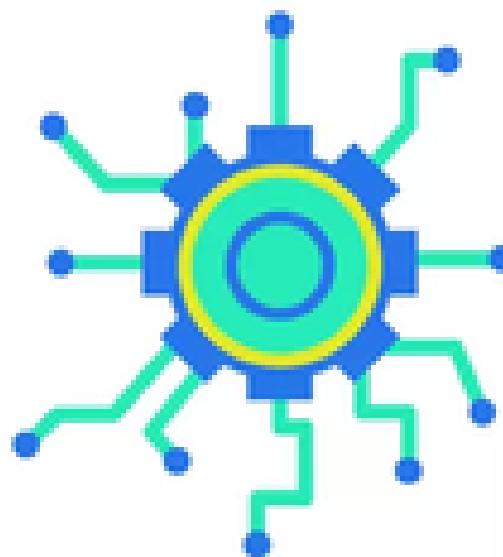
Flat layer

Dense layers

Output
layer

Implementation



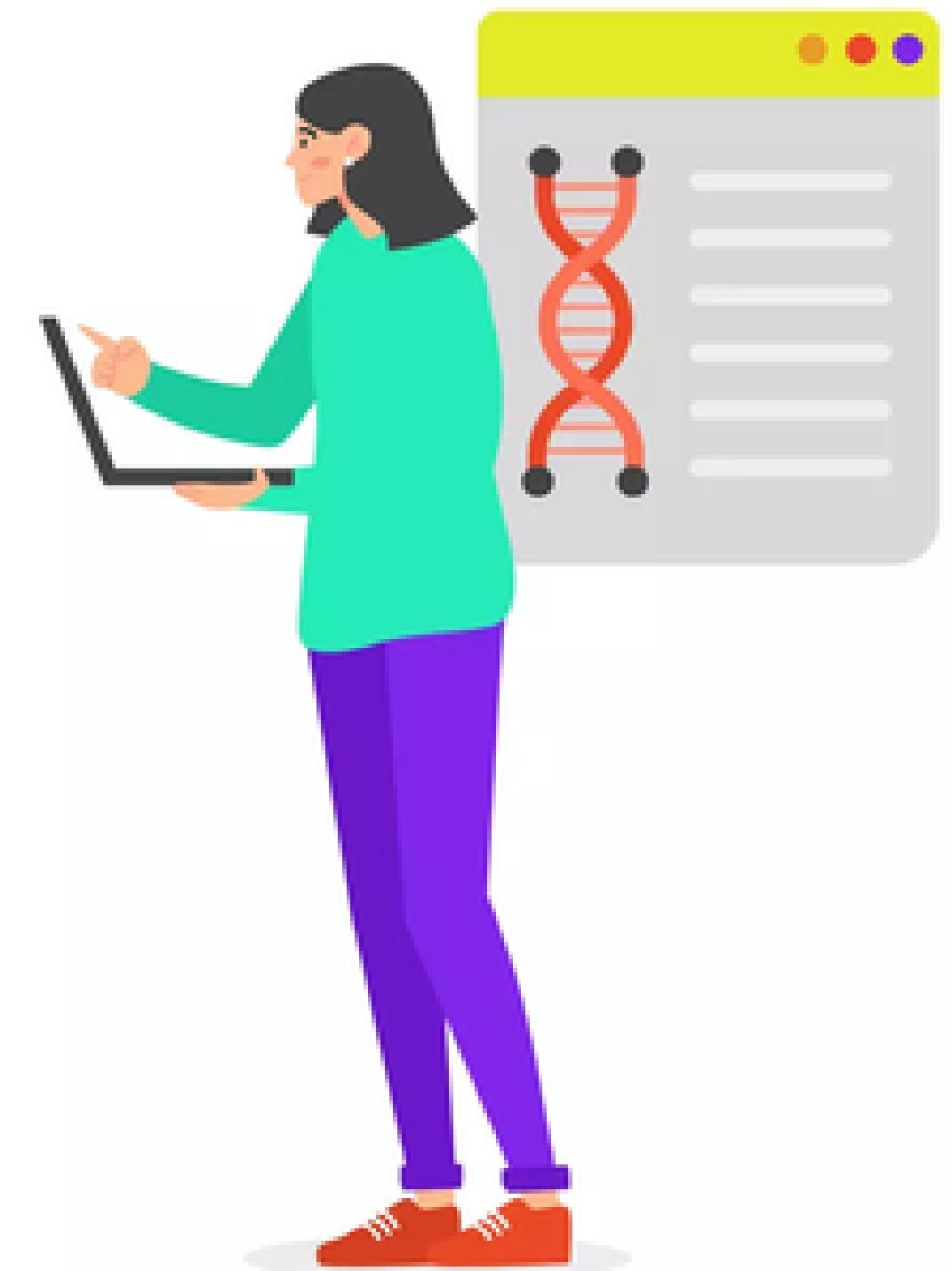


Pour résumer :

- Réseaux de neurones convolutifs :
Permet d'extraire les caractéristiques pertinentes dans les inputs : caractéristiques qui sont invariantes par translation (forme dans les images), corrélation locale.
- La partie de la couche convulsive d'un CNN consiste à appliquer des filtres de convolution sur les données d'entrée pour extraire des caractéristiques pertinentes, suivi d'une fonction d'activation non linéaire pour introduire de la non-linéarité dans le modèle. Cette opération est répétée avec plusieurs filtres pour capturer des informations à différentes échelles et niveaux d'abstraction, ce qui permet au CNN d'apprendre des représentations hiérarchiques des données.
- le stride contrôle le déplacement spatial lors de la convolution, tandis que le padding permet de contrôler la taille de la sortie en ajoutant des zéros autour des bords de l'entrée.
- la couche fully-connected d'un CNN est responsable de la classification ou de la régression en utilisant les caractéristiques extraites par les couches convolutives.

Conclusion

Finalement, le principe de fonctionnement d'un CNN est assez facile à comprendre, mais paradoxalement, la mise en place d'un tel procédé pour classifier des images demeure très complexe étant donné le nombre considérable de paramètres à définir : nombre, taille, déplacement des filtres, choix de la méthode de pooling, choix du nombre de couches de neurones, nombre de neurones par couches, etc.



T₁ H₄ A₁ N₁ K₅ S₁

