

Plan du cours

1) Introduction au machine learning

2) Régularisation et forêts aléatoires

3) Réseau de neurones

4) Réseau de neurones convolutifs

Pour chaque séance:

1h de cours / support transparent

2h de travaux pratiques (amener un ordinateur portable)

Perceptron et réseau de neurones

Les réseaux dense ou fully-connected on l'inconvénient d'avoir un nombre de poids très grand:

- Si image RGB de 5 x 5 pixels, on a $m = 3 \times 5 \times 5 = 75$ connections et poids par neurone de la première couche.
- Si image RGB de 1000 pixels on a $m = 3 \times 10^3 \times 10^3 = 3 \times 10^6$ poids par neurone à calculer. Inutilisable pour plusieurs neurones et plusieurs couches.

Une solution à ce problème a été apportée par les réseaux de neurone convolutif (CNN, convolutional neural network).

Ces réseaux utilise la notion de convolution en traitement du signal.

Réseaux de neurone convolutif

En traitement du signal une convolution de deux fonctions est défini par :

$$(f * g)(t) = \int_{-\infty}^{\infty} f(t - \tau)g(\tau)d\tau$$

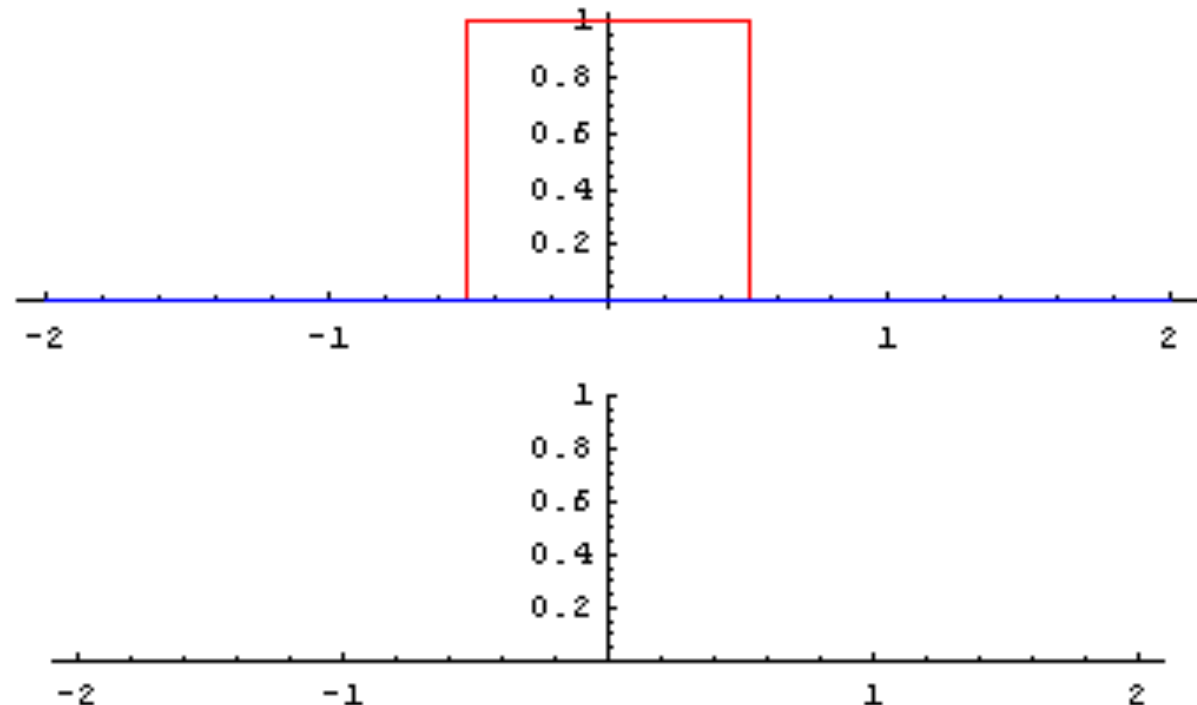
f et g

Fonction porte

=1 si $-1/2 < x < 1/2$

=0 sinon

$f * g$



Filtre 1D

On utilise classiquement en climatologie des filtres pour lisser des séries temporelles.

$$f'(t_i) = \sum_{k=-N}^N f(t_{i-k}) w_{N-k+1}$$

Les paramètres $w_{1 \leq i \leq 2N+1}$
forment un noyau ou kernel

$2N + 1$ est la taille du filtre

Exemple :

$w_i = \frac{1}{2N+1}$ forme la moyenne glissante sur une fenêtre de $2N + 1$ pas de temps.

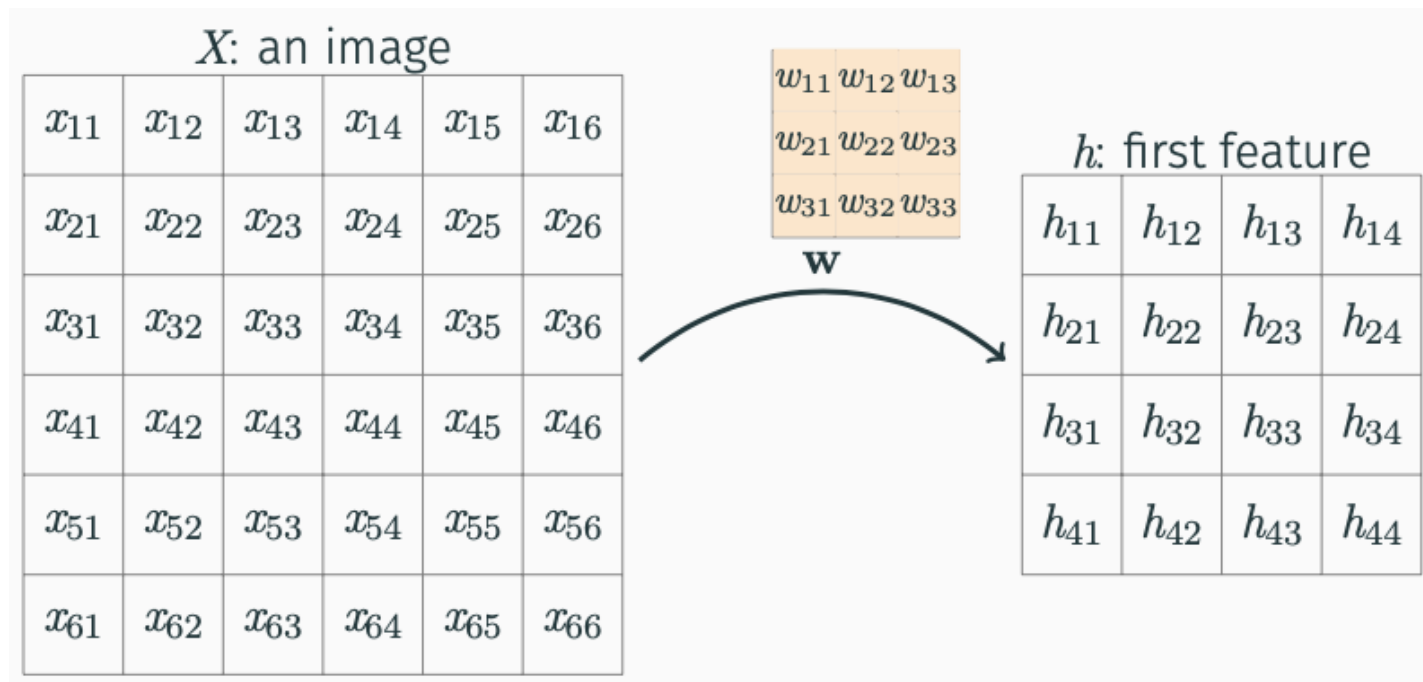
Il existe des filtres avec des meilleures propriétés, comme les filtres de Butterworth ou Lanczos.

Filtre 2D

On utilise classiquement des filtres mettre en avant certaines caractéristiques de l'image.

Exemple : filtre de taille 3 x 3

$$h_{i,j} = \sum_{k=1}^3 \sum_{l=1}^3 x_{i+k-1,j+l-1} w_{k,l}$$



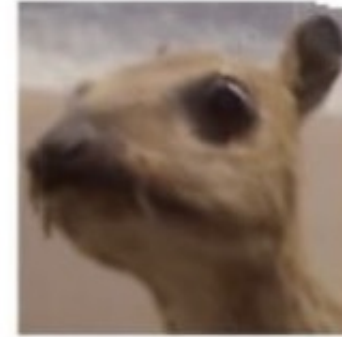
Filtre 2D

Input



Edge detection

$$\begin{bmatrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{bmatrix}$$



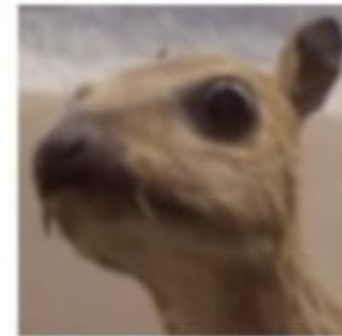
Box mean

$$\frac{1}{9} \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$$



Sharpen

$$\begin{bmatrix} 0 & -1 & 0 \\ -1 & 5 & -1 \\ 0 & -1 & 0 \end{bmatrix}$$

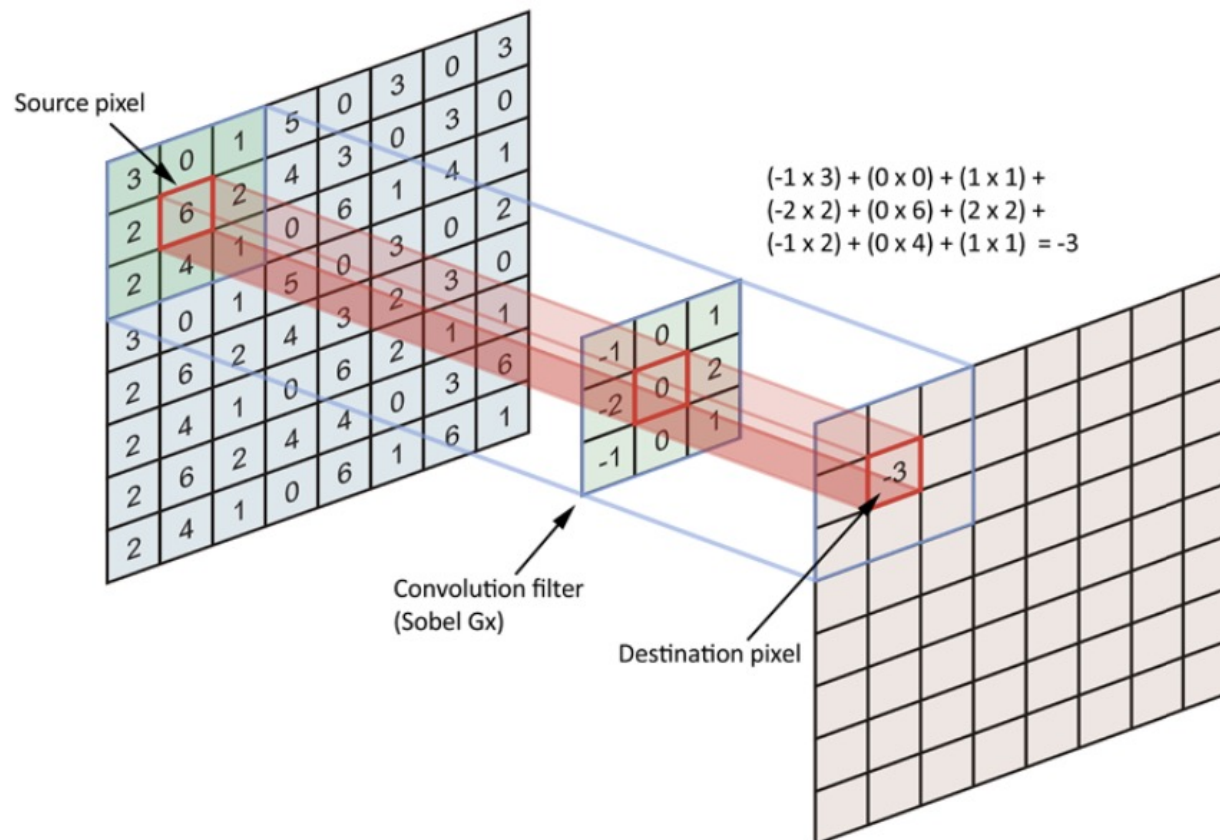


Gaussian blur

$$\frac{1}{16} \begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix}$$

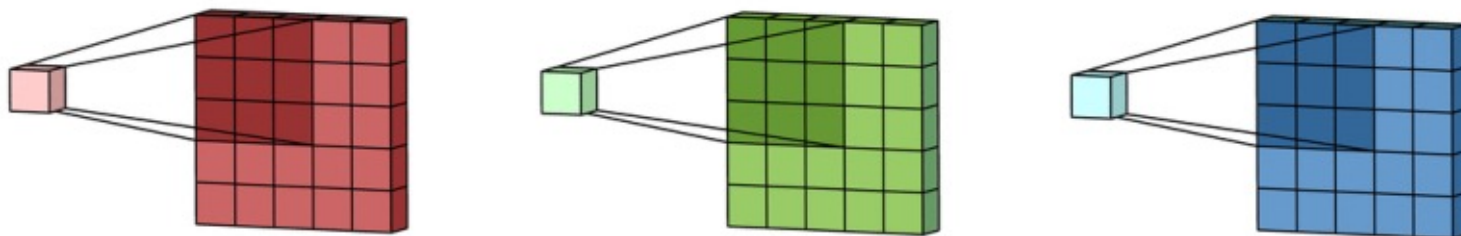
Filtre 2D

Dans un **CNN (Convolutional Neural Network)**, les poids correspondent au kernel. Cela permet d'extraire certaines caractéristiques de l'image.



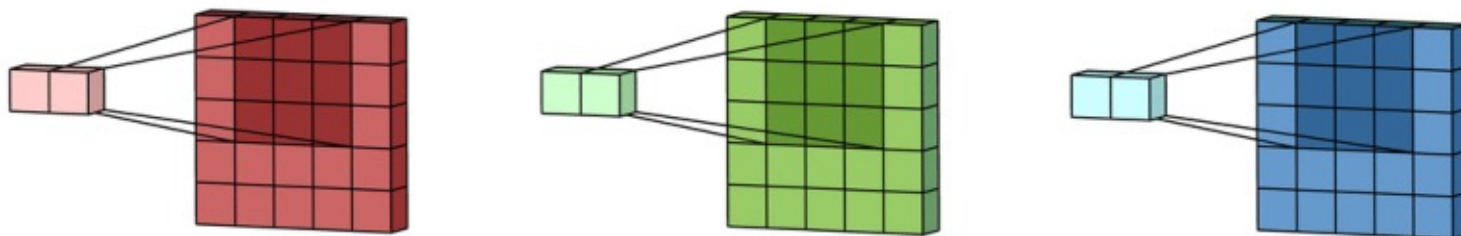
Summary of Convolutional layer steps

1. Convolution



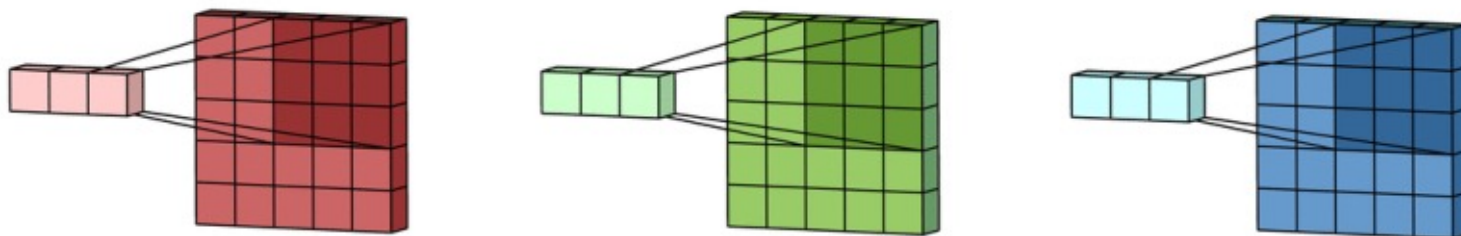
Summary of Convolutional layer steps

1. Convolution



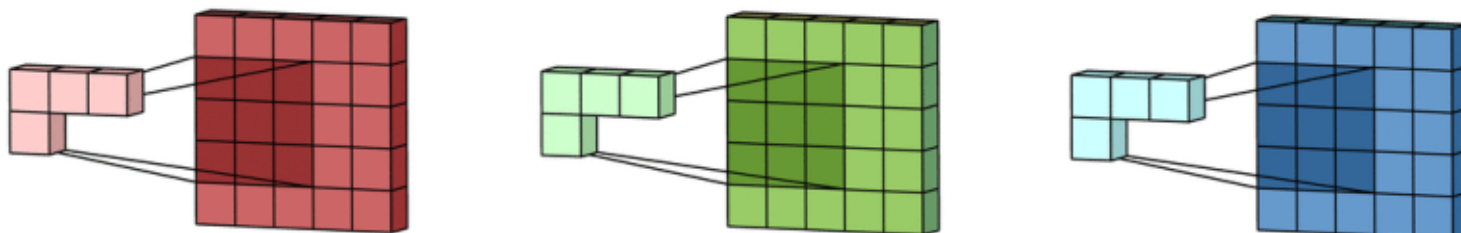
Summary of Convolutional layer steps

1. Convolution



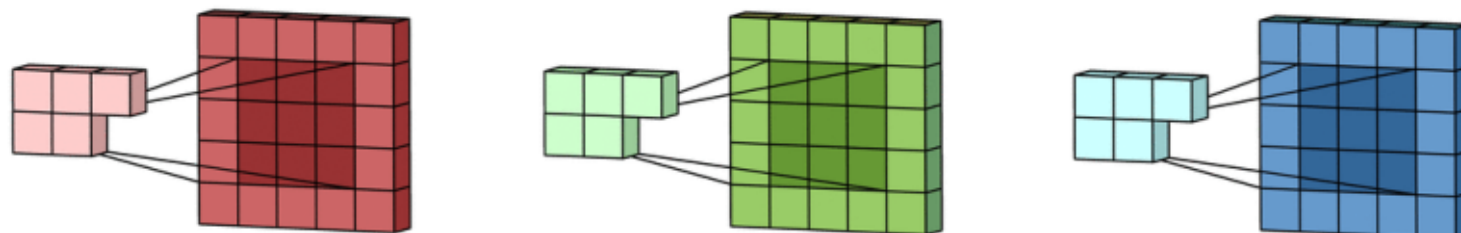
Summary of Convolutional layer steps

1. Convolution



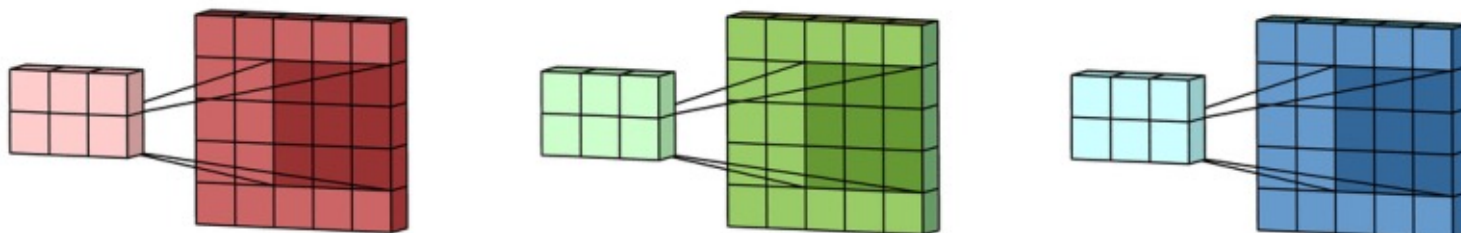
Summary of Convolutional layer steps

1. Convolution



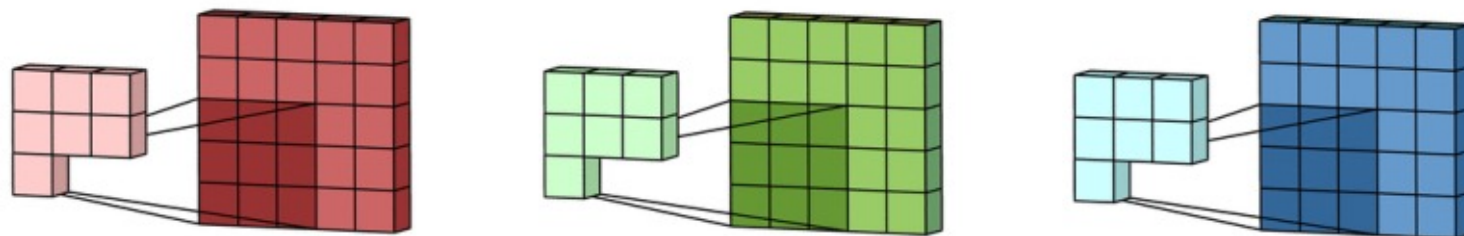
Summary of Convolutional layer steps

1. Convolution



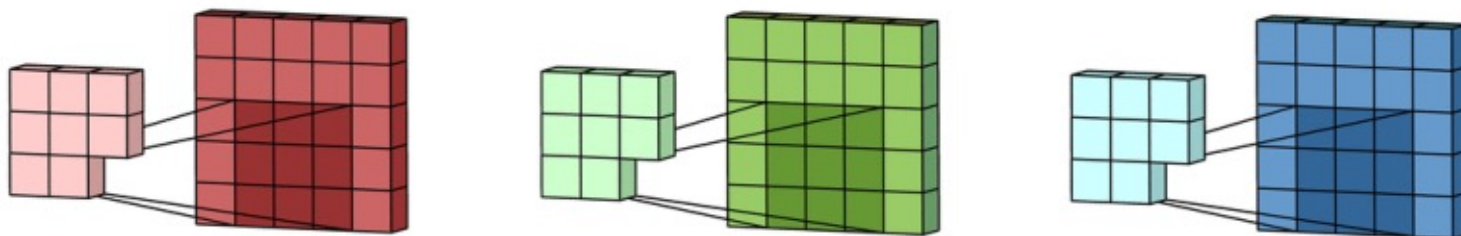
Summary of Convolutional layer steps

1. Convolution



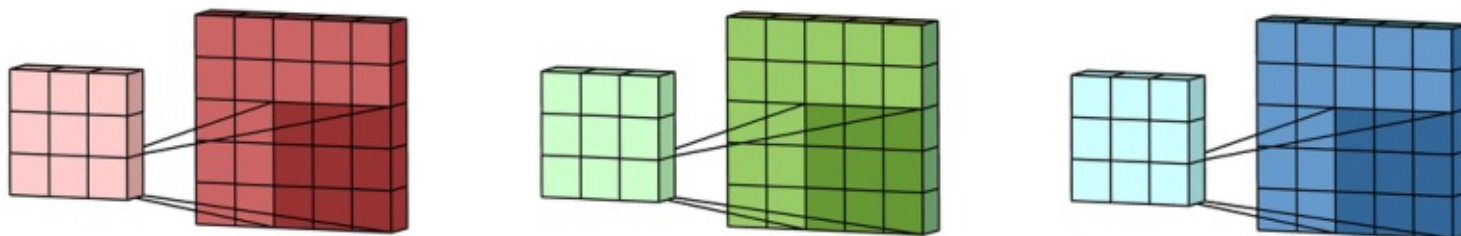
Summary of Convolutional layer steps

1. Convolution



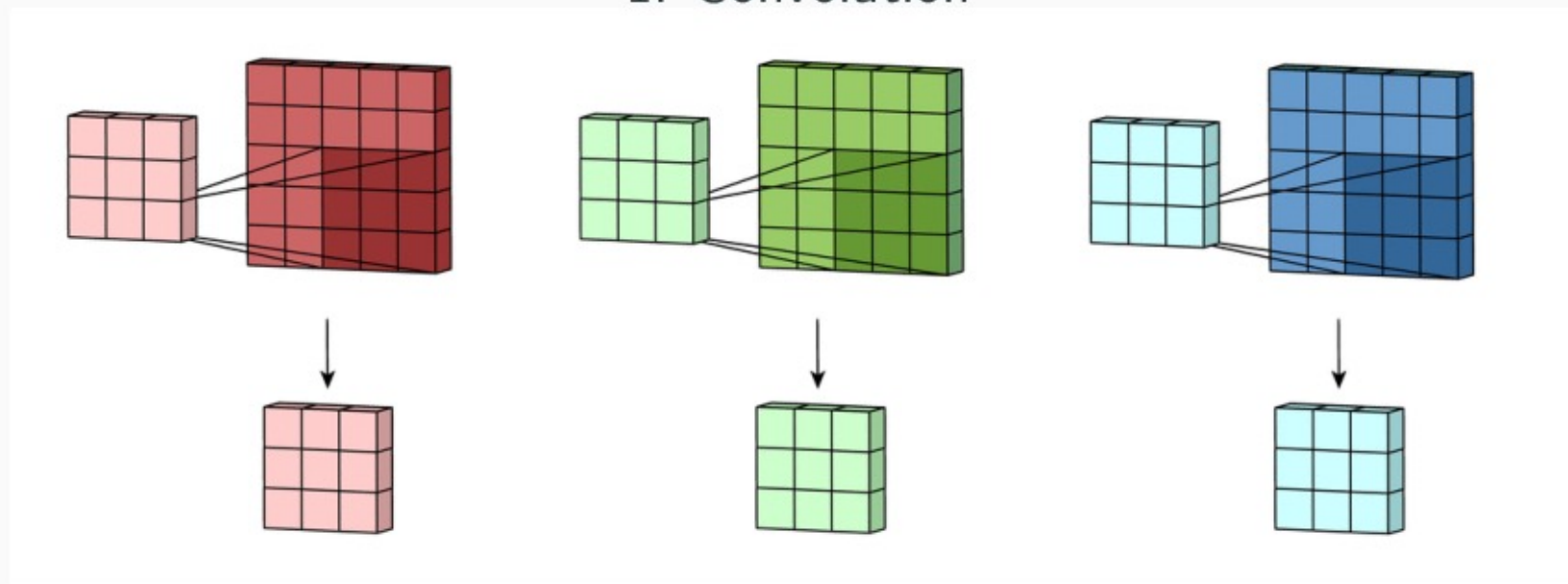
Summary of Convolutional layer steps

1. Convolution



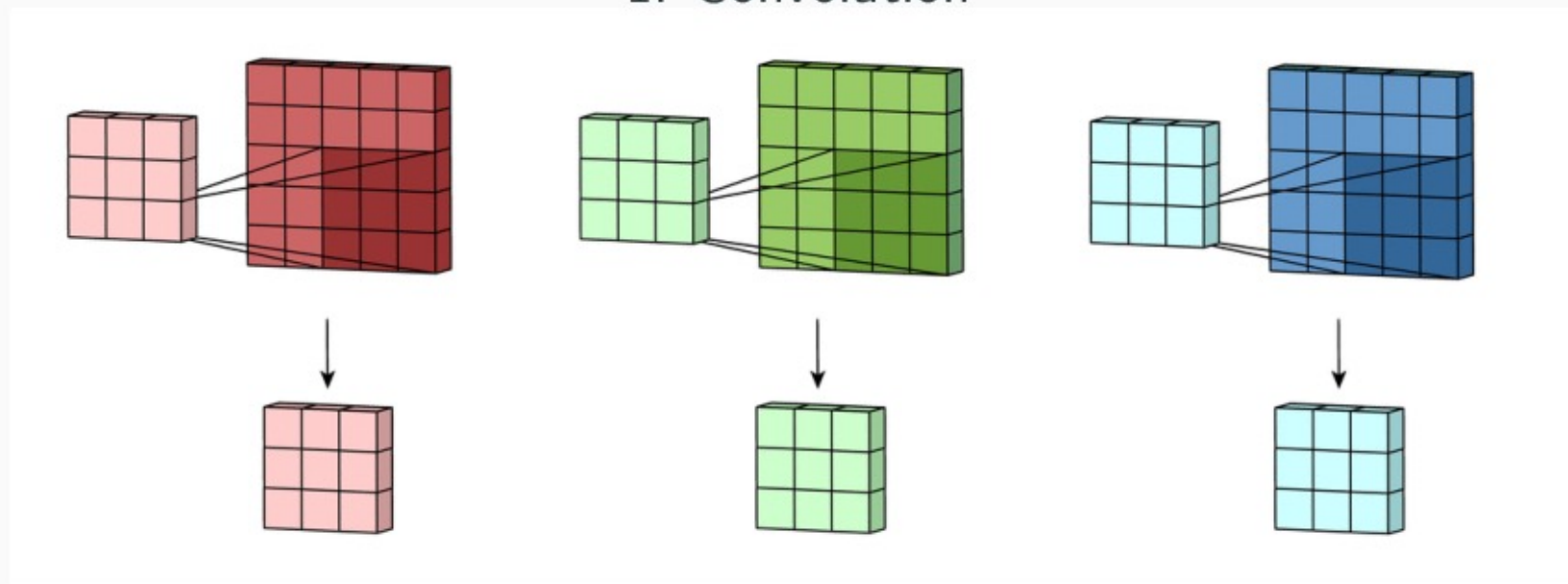
Summary of Convolutional layer steps

1. Convolution



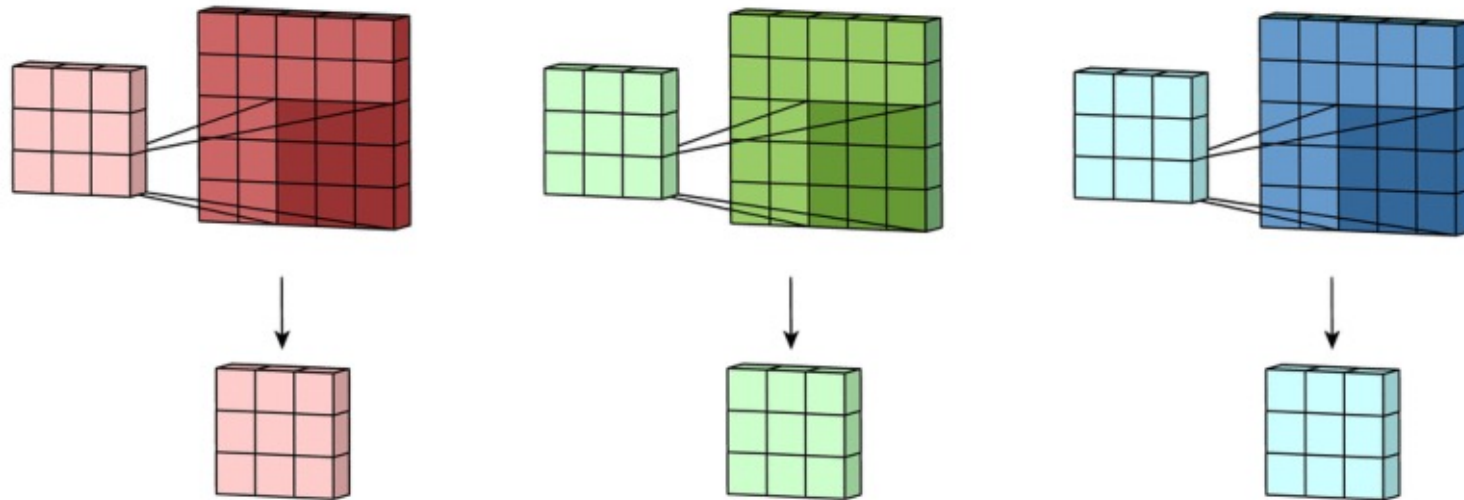
Summary of Convolutional layer steps

1. Convolution



Summary of Convolutional layer steps

1. Convolution

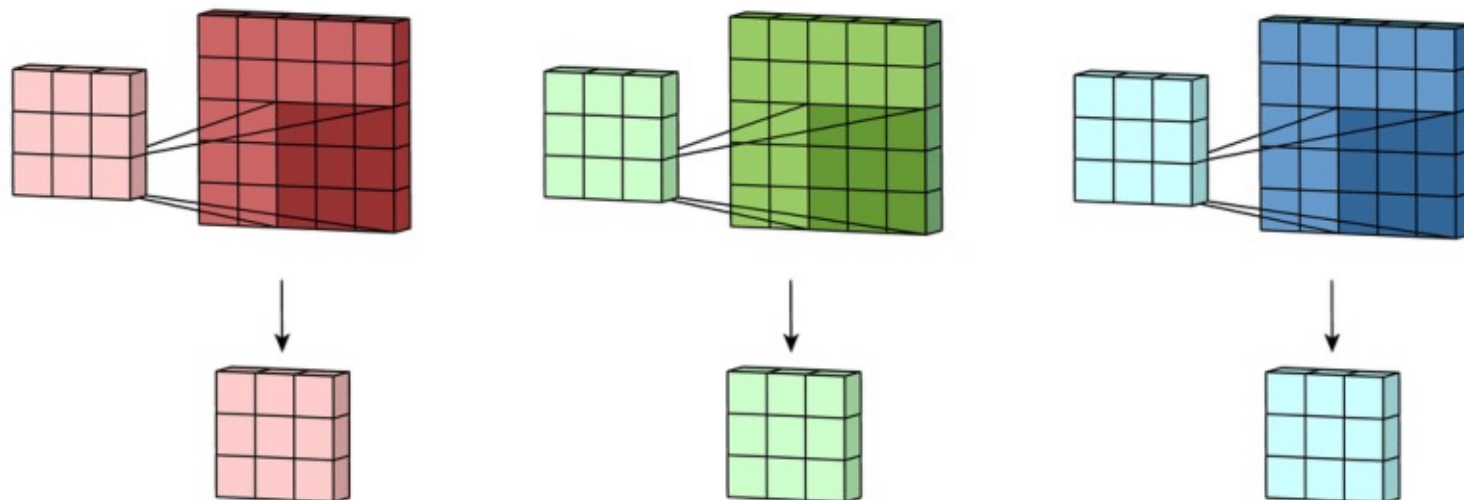


2. Addition

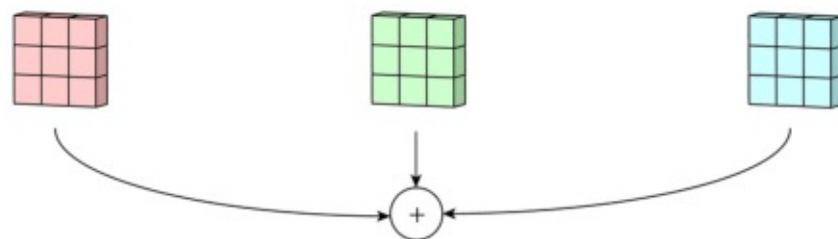


Summary of Convolutional layer steps

1. Convolution

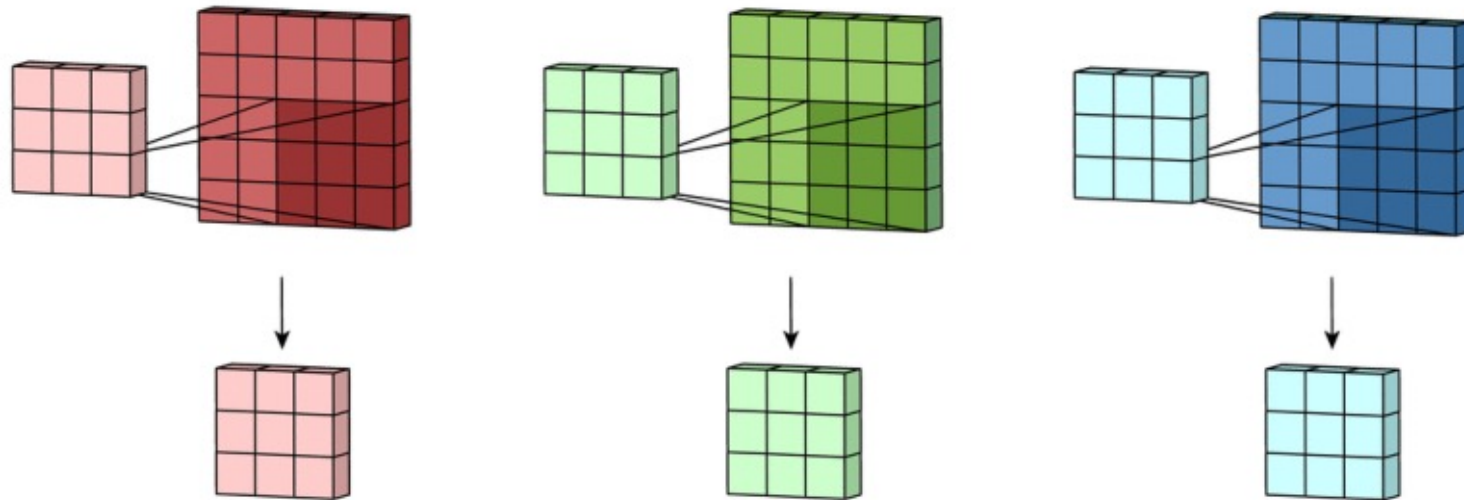


2. Addition

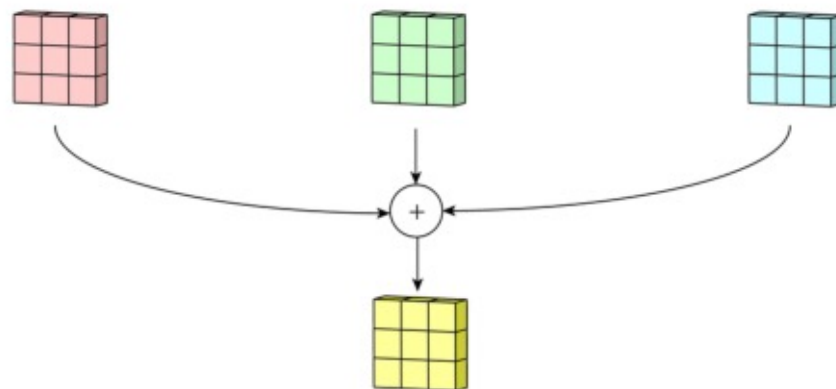


Summary of Convolutional layer steps

1. Convolution

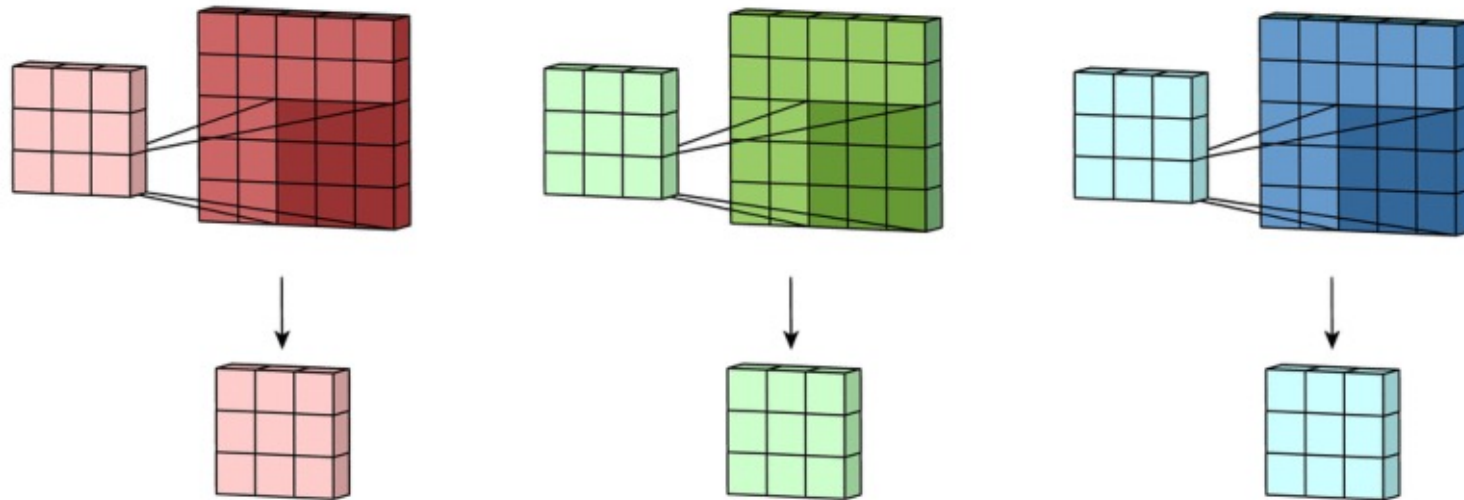


2. Addition

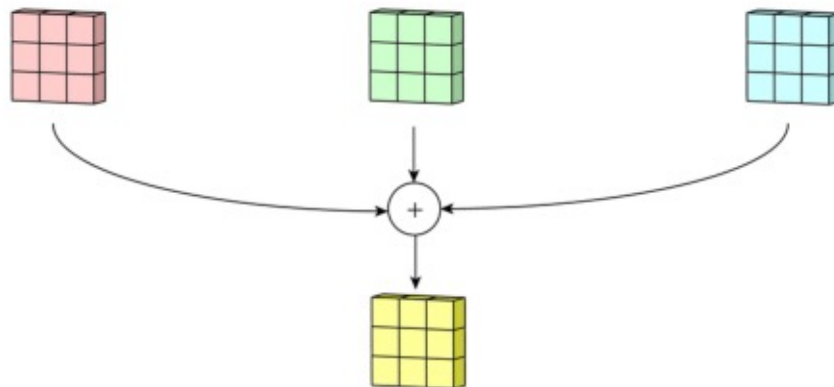


Summary of Convolutional layer steps

1. Convolution

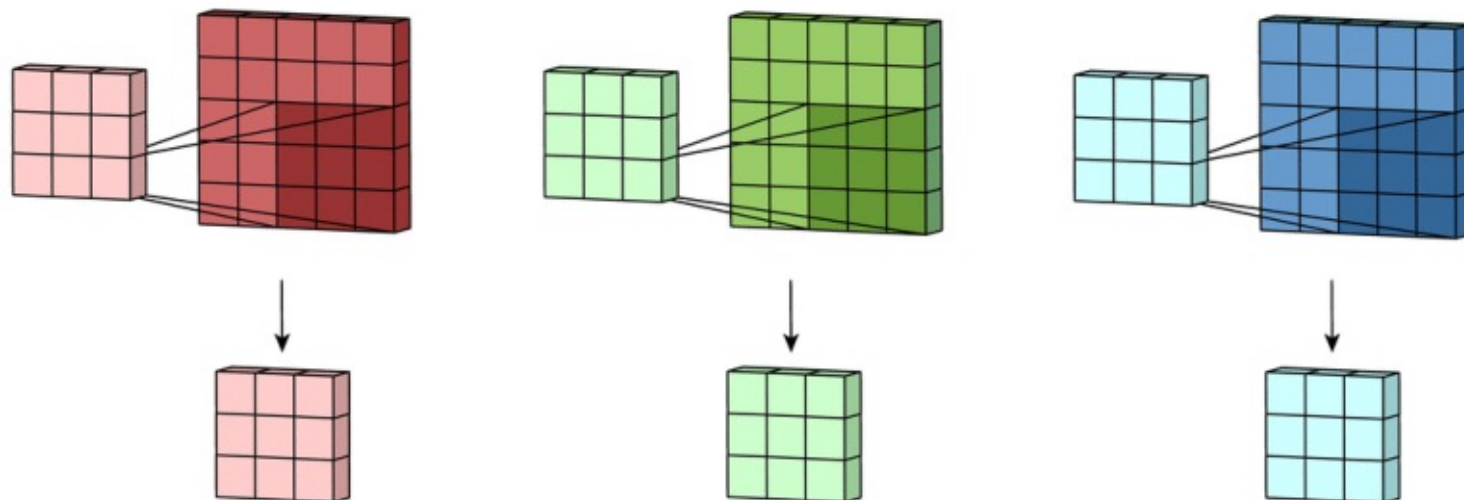


2. Addition

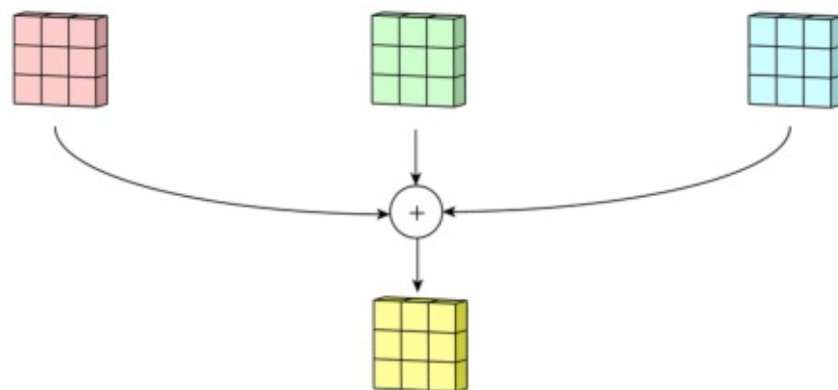


Summary of Convolutional layer steps

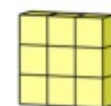
1. Convolution



2. Addition

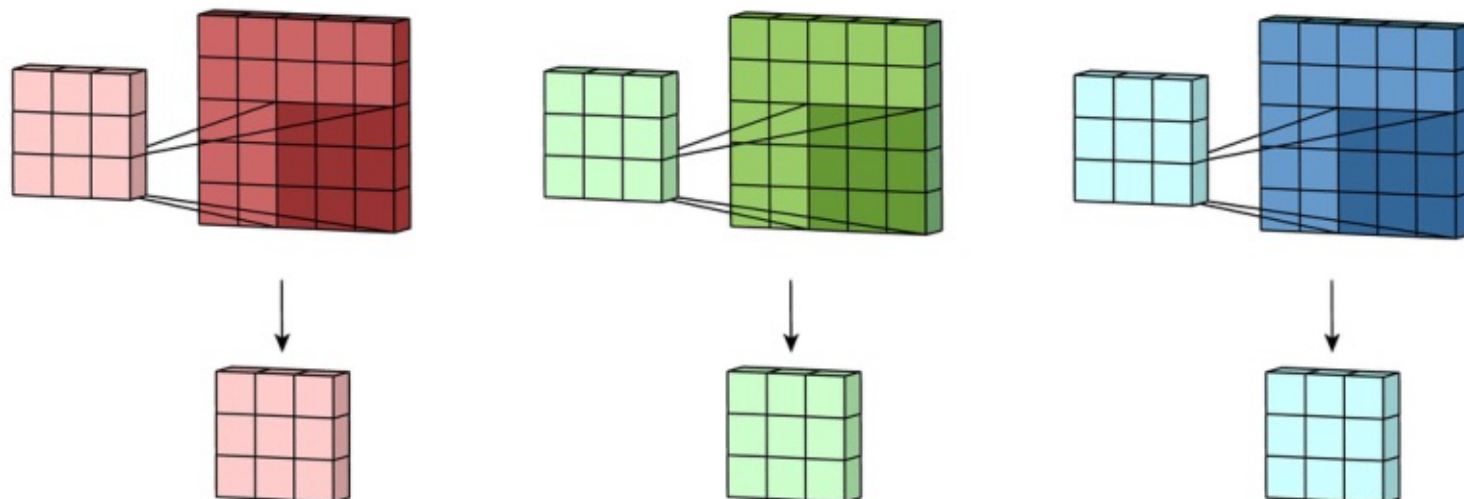


3. Bias

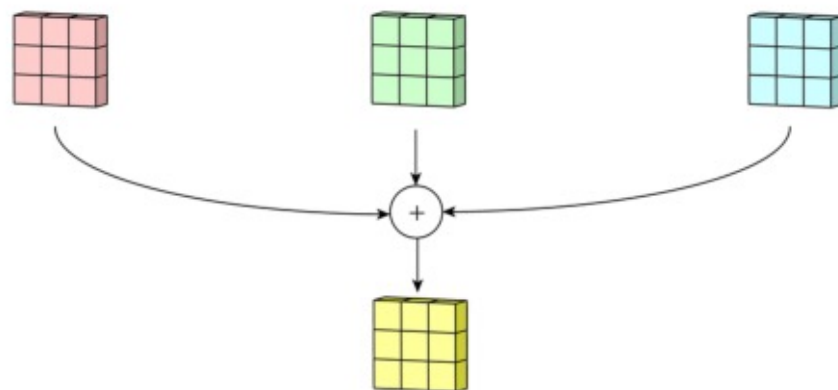


Summary of Convolutional layer steps

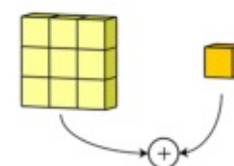
1. Convolution



2. Addition

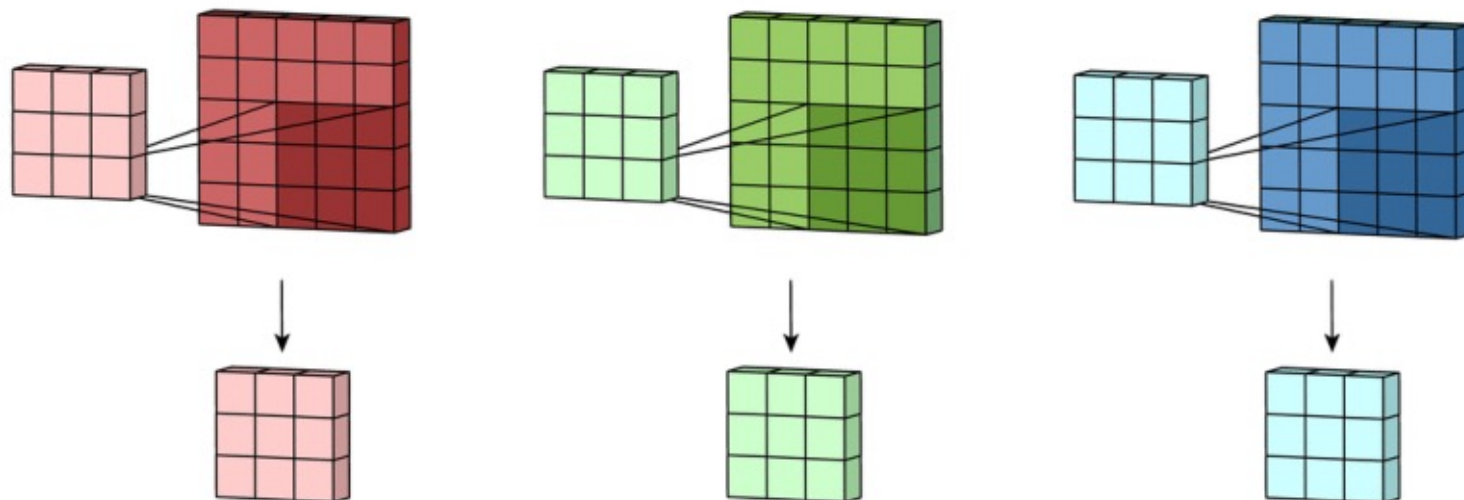


3. Bias

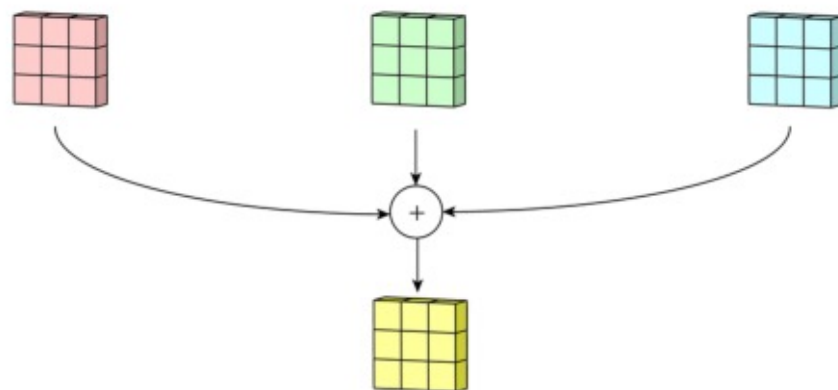


Summary of Convolutional layer steps

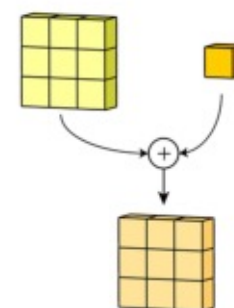
1. Convolution



2. Addition

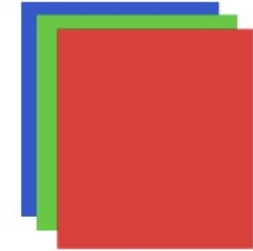


3. Bias



Cas d'un input 3D

RGB



On applique un kernel différent à chaque canaux.

0	0	0	0	0	0	...
0	156	155	156	158	158	...
0	153	154	157	159	159	...
0	149	151	155	158	159	...
0	146	146	149	153	158	...
0	145	143	143	148	158	...
...

Input Channel #1 (Red)

0	0	0	0	0	0	...
0	167	166	167	169	169	...
0	164	165	168	170	170	...
0	160	162	166	169	170	...
0	156	156	159	163	168	...
0	155	153	153	158	168	...
...

Input Channel #2 (Green)

0	0	0	0	0	0	...
0	163	162	163	165	165	...
0	160	161	164	166	166	...
0	156	158	162	165	166	...
0	155	155	158	162	167	...
0	154	152	152	157	167	...
...

Input Channel #3 (Blue)

-1	-1	1
0	1	-1
0	1	1

Kernel Channel #1

1	0	0
1	-1	-1
1	0	-1

Kernel Channel #2

0	1	1
0	1	0
1	-1	1

Kernel Channel #3

Ici kernel de taille 3x3

308

+

-498

+

164

+ 1 = -25

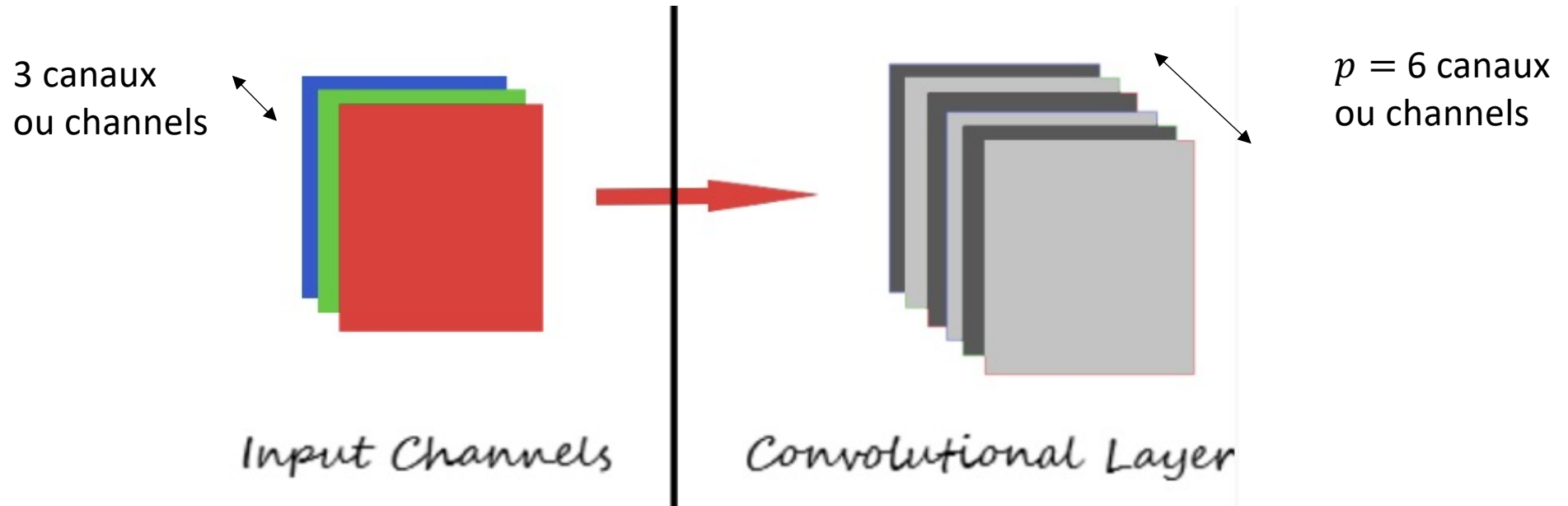
Bias = 1

Output

-25				...
				...
				...
				...
...

Principaux paramètres

- K = taille des kernels
- p = Nombres de canaux de sorties

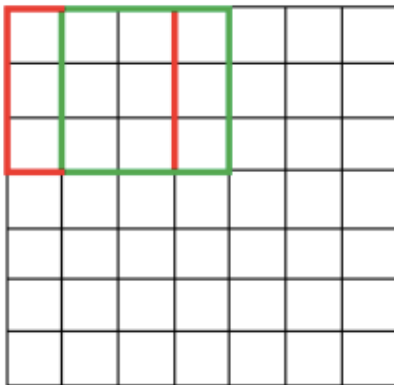


Principaux paramètres

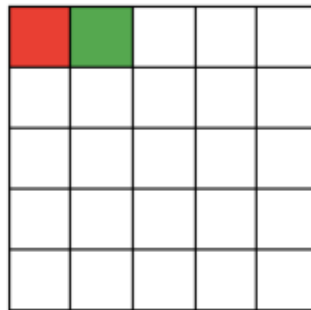
- K = taille des kernels
- p = Nombres de canaux de sorties
- S = Stride

$S = 1$

7 x 7 Input Volume

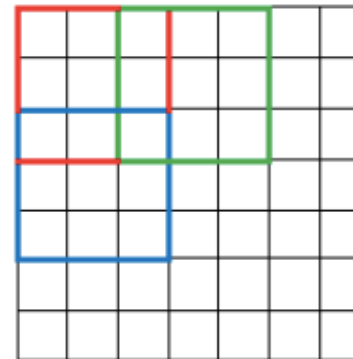


5 x 5 Output Volume



$S = 2$

7 x 7 Input Volume

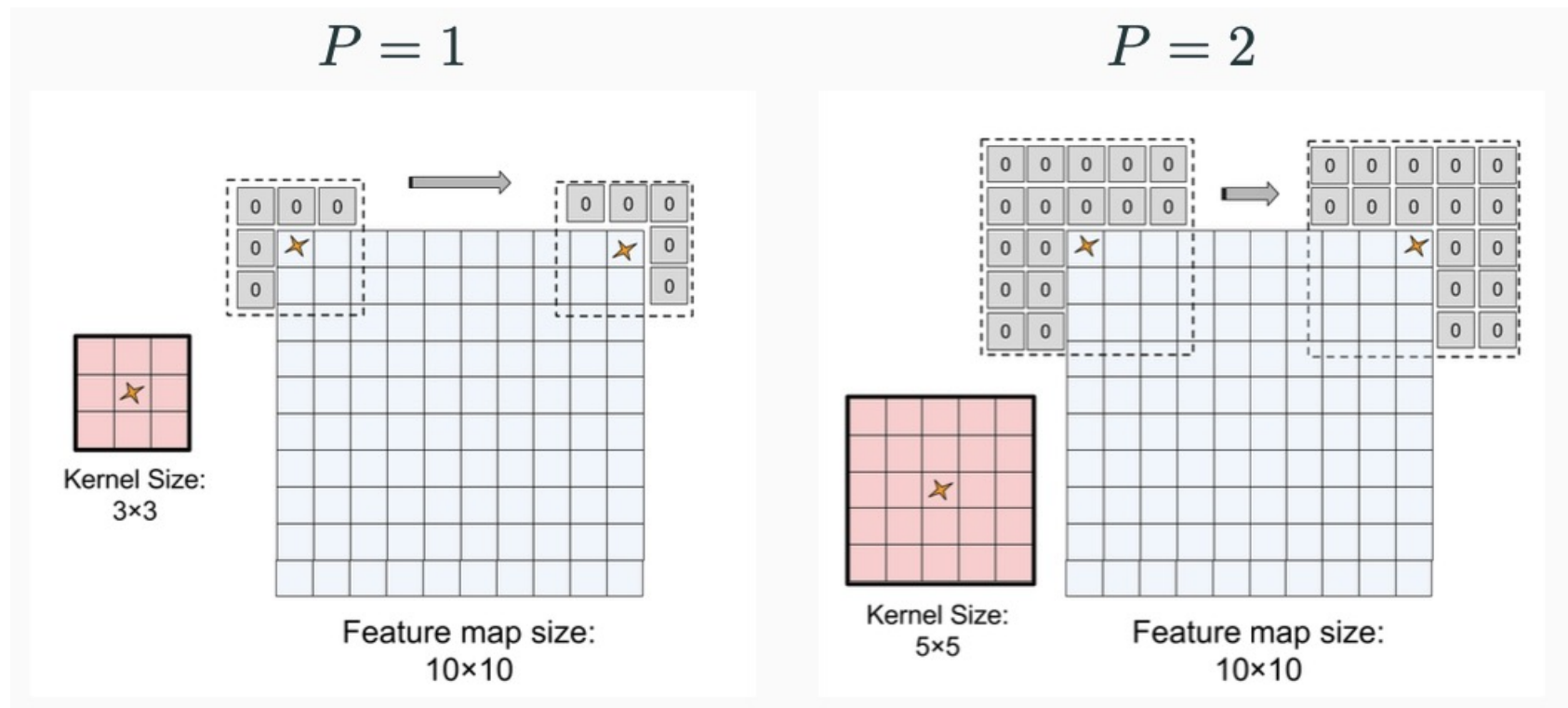


3 x 3 Output Volume



Principaux paramètres

- K = taille des kernels
- p = Nombres de canaux de sorties
- S = Stride
- P = Padding



Principaux paramètres

- K = taille des kernels
- p = Nombres de canaux de sorties
- S = Stride
- P = Padding

Si on a: N_{in} : dimension de l'image d'entrée
 N_{out} : dimension de l'image de sortie

Alors:

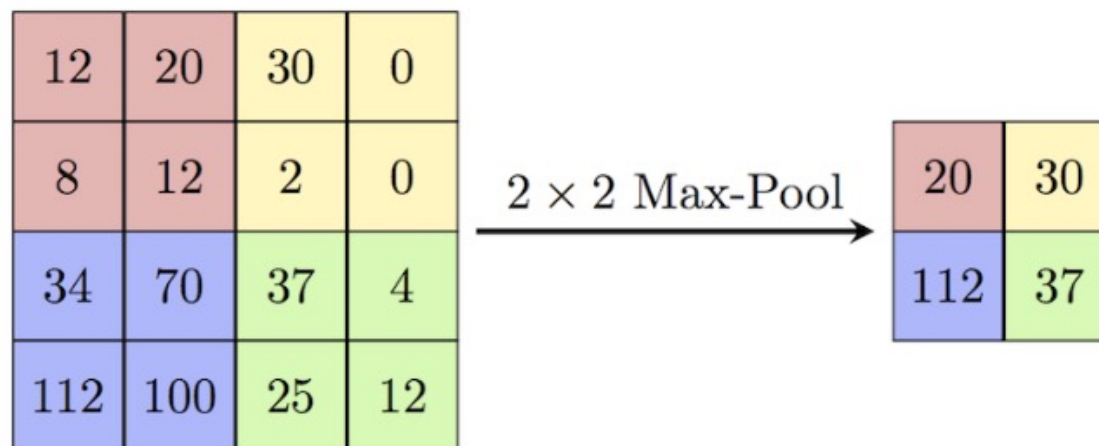
$$N_{out} = \frac{N_{in} - K + 2P}{S} + 1$$

Quelques remarques

- les couches convolutives agissent au niveau local,
- les poids sont invariants par translation (ils ne dépendent pas de la localisation),
- Les CNNs peuvent utiliser des images de taille différente.

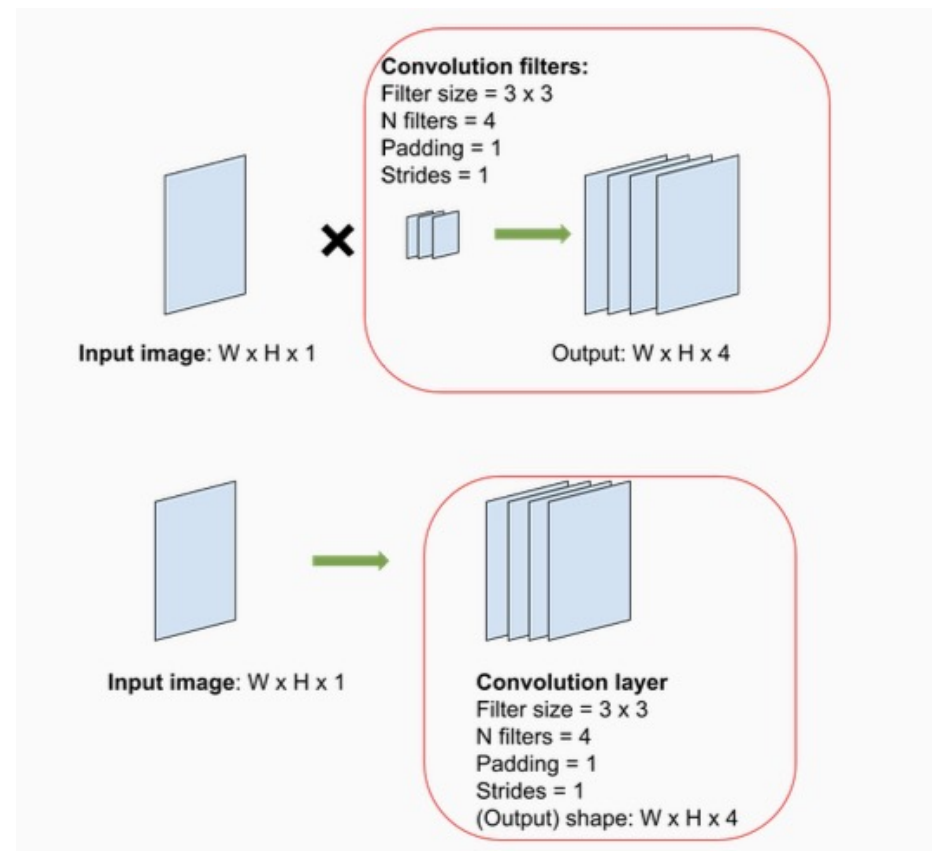
Maxpool

Une transformation **Maxpool** est alors appliqué pour réduire la taille de l'image, tout en conservant les gradients.

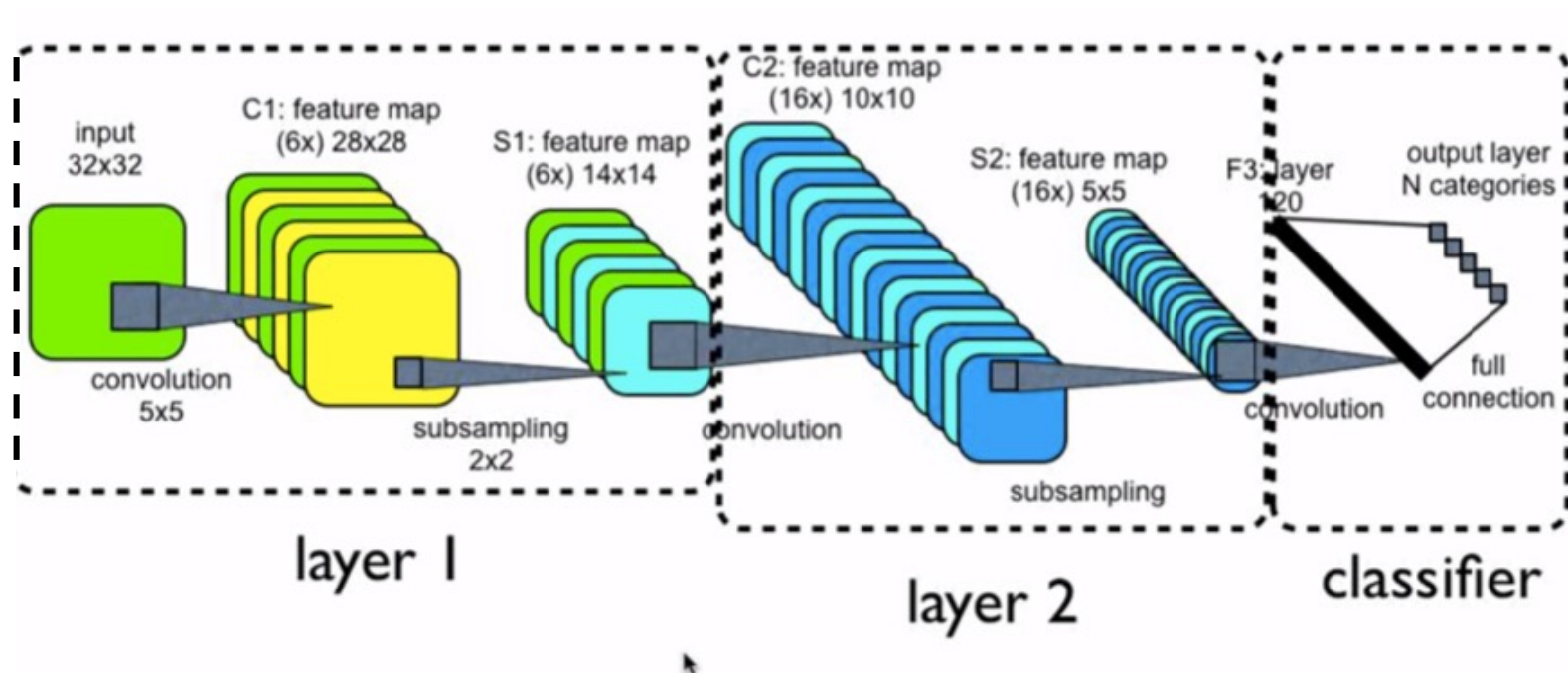


Quelques remarques

- Une opération de convolution est souvent schématisée par une couche convolutive et en faisant apparaître les paramètres de padding, stride, taille des kernels, nombre de canaux.



Exemple de classification



Ce réseau peut répondre à la question : est-ce que l'image appartient à la catégorie k avec $k \in \llbracket 0, N \rrbracket$? Exemple : Chien, chat ou canard?
On peut également faire de la régression. Exemple : à quel point l'image est-elle celle d'un chien?

Segmentation d'image

L'opération de segmentation consiste à produire une image où chaque partie de l'image est classifié.



Person
Bicycle
Background

Segmentation d'image

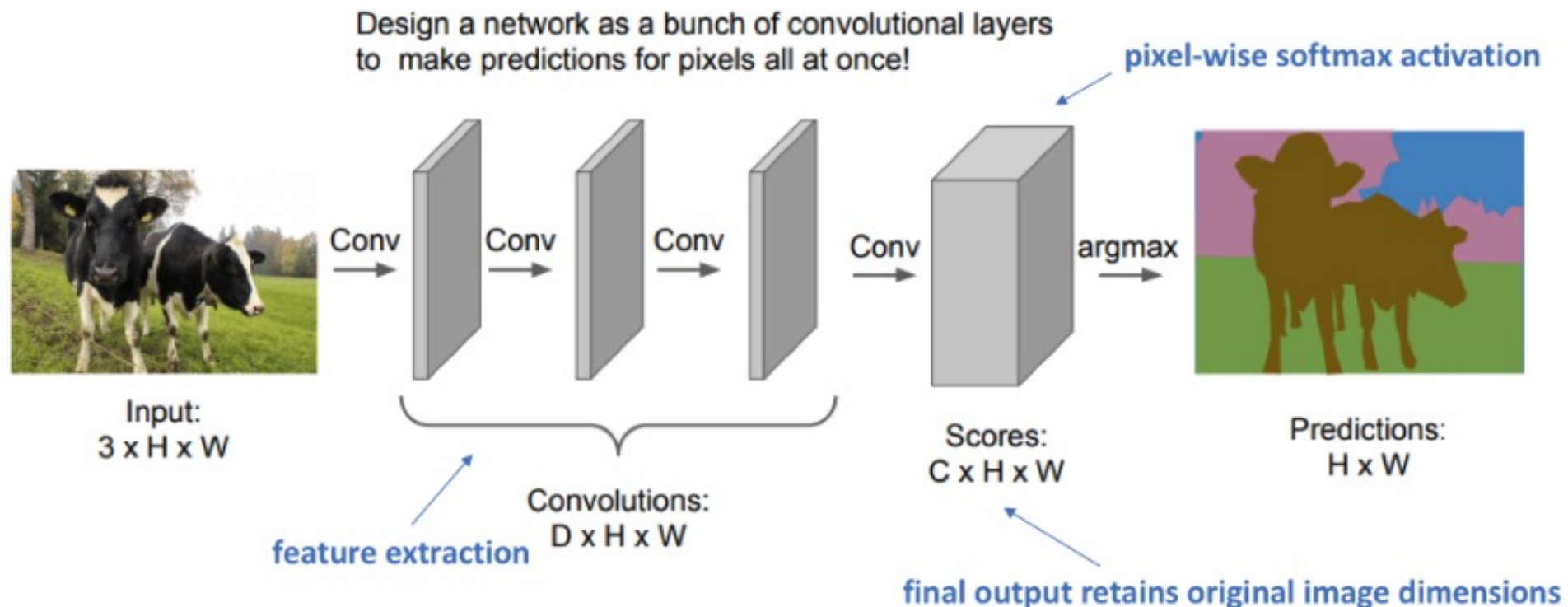
Chaque pixel d'une image est associé à un entier caractérisant une classe.



0: Background/Unknown
1: Person
2: Purse
3: Plants/Grass
4: Sidewalk
5: Building/Structures

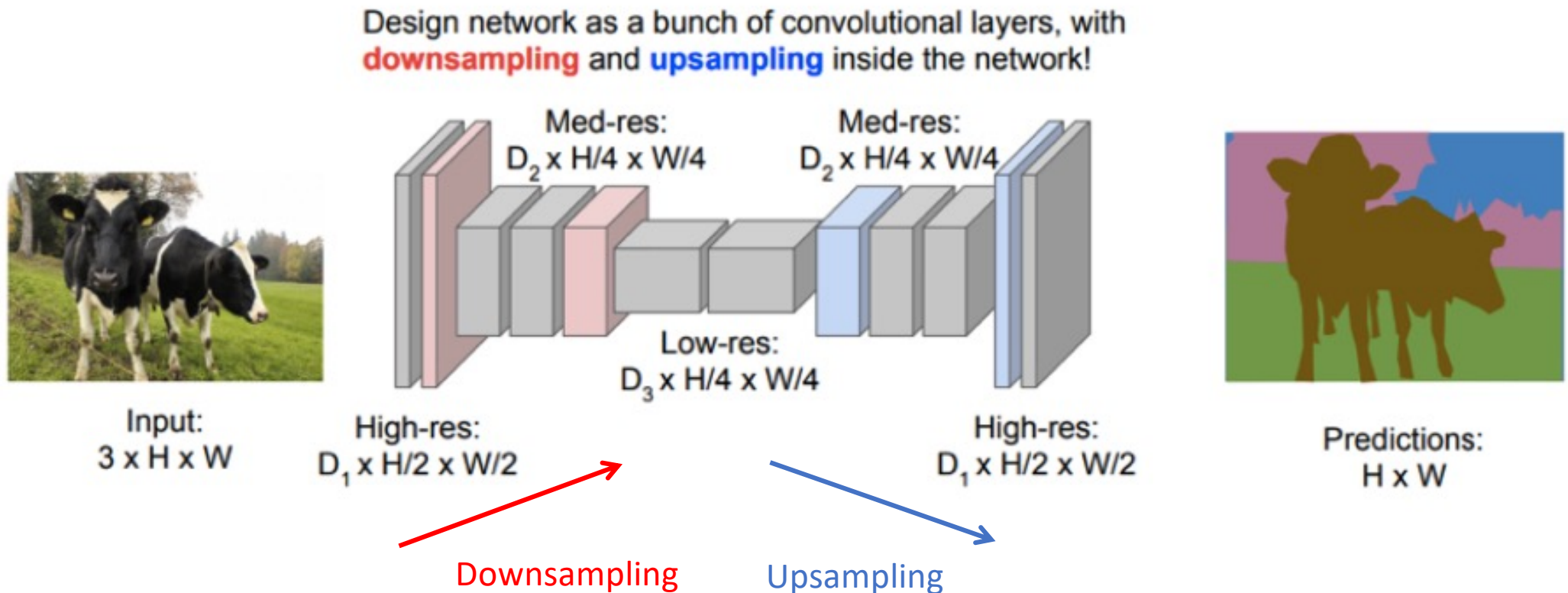
Approche par un CNN simple

Une approche simple consiste à réaliser une série de convolutions en conservant la taille de l'image.



Problème : en conservant la taille de l'image, l'apprentissage peut être long.

Approche plus performante



Avantage : l'apprentissage se fait à plus basse résolution pour les couches centrales.

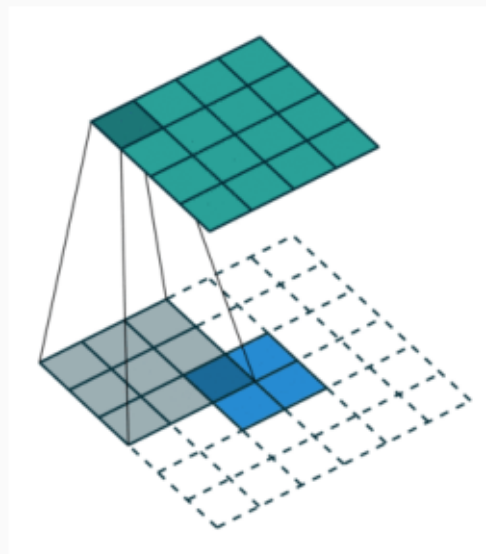
Upsampling-Downsampling

- Pour dégrader la résolution, on utilise du Maxpool (=downsampling)
- Pour retrouver la résolution originale de l'image, on utilise des couches convolutives avec du padding et ou un stride (upsampling).

Upsampling using "Transpose Convolution"

Often "Transpose Convolution" is used for upsampling. A convolution filter is applied to small input image with large padding and/or with strides. The example here shows: Padding = 2, Stride = 0.

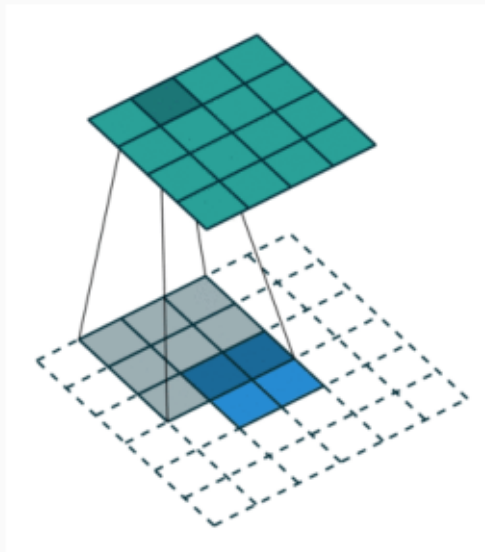
Blue - input image, gray - filter, green - output image.



Upsampling using "Transpose Convolution"

Often "Transpose Convolution" is used for upsampling. A convolution filter is applied to small input image with large padding and/or with strides. The example here shows: Padding = 2, Stride = 0.

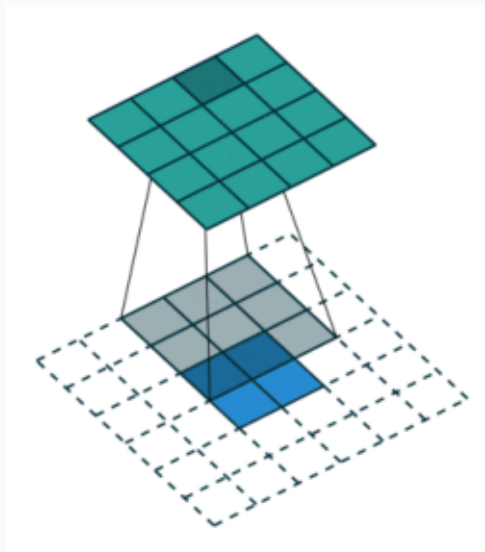
Blue - input image, gray - filter, green - output image.



Upsampling using "Transpose Convolution"

Often "Transpose Convolution" is used for upsampling. A convolution filter is applied to small input image with large padding and/or with strides. The example here shows: Padding = 2, Stride = 0.

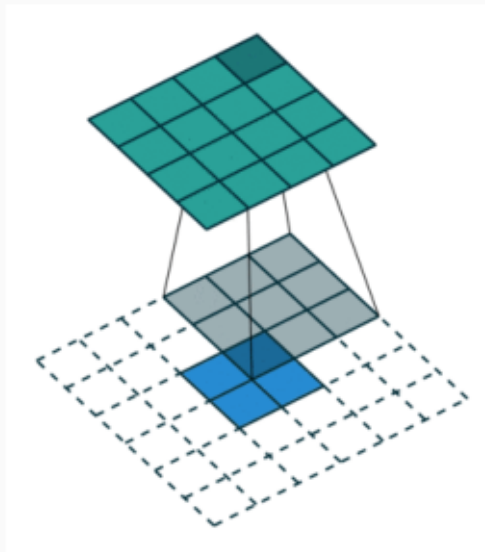
Blue - input image, gray - filter, green - output image.



Upsampling using "Transpose Convolution"

Often "Transpose Convolution" is used for upsampling. A convolution filter is applied to small input image with large padding and/or with strides. The example here shows: Padding = 2, Stride = 0.

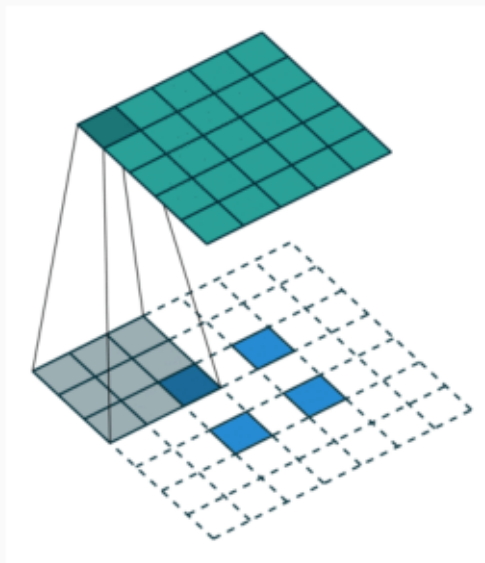
Blue - input image, gray - filter, green - output image.



Upsampling using "Transpose Convolution"

Padding = 2, Stride = 1.

Blue - input image, gray - filter, green - output image.

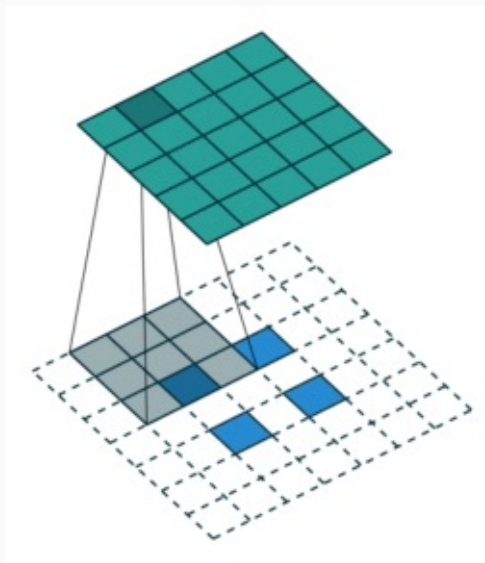


Images from fvisin Francesco

Upsampling using "Transpose Convolution"

Padding = 2, Stride = 1.

Blue - input image, gray - filter, green - output image.

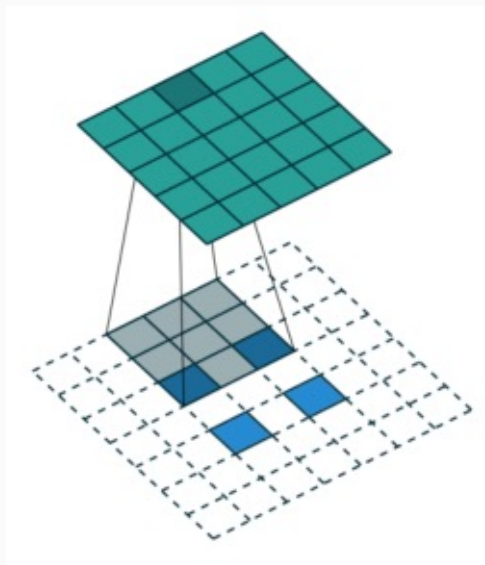


Images from fvisin Francesco

Upsampling using "Transpose Convolution"

Padding = 2, Stride = 1.

Blue - input image, gray - filter, green - output image.

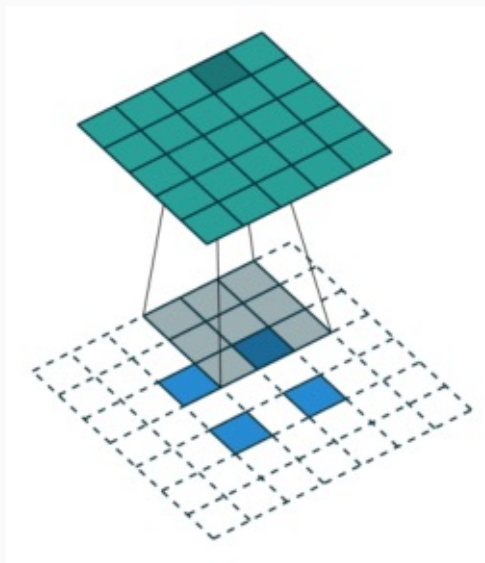


Images from fvisin Francesco

Upsampling using "Transpose Convolution"

Padding = 2, Stride = 1.

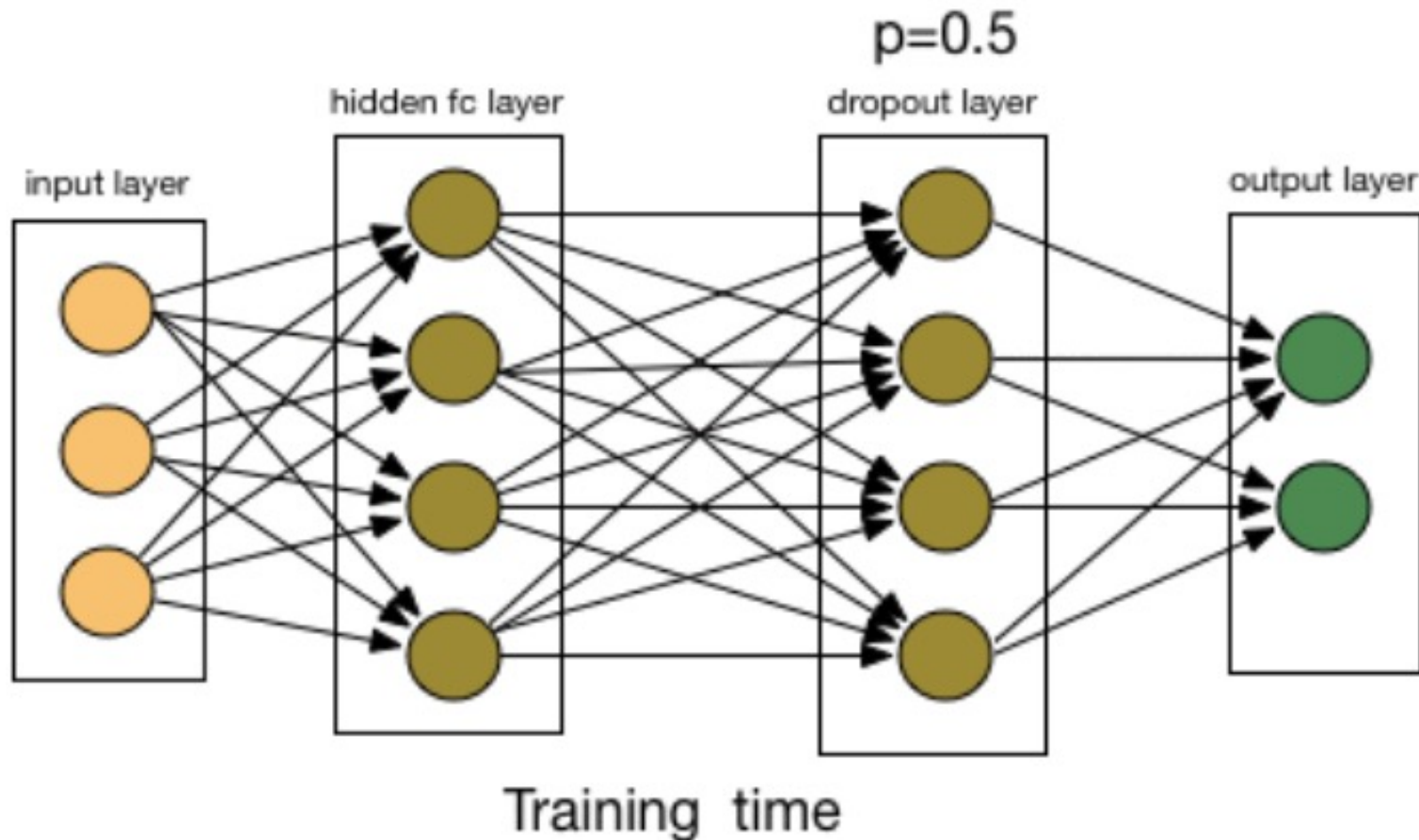
Blue - input image, gray - filter, green - output image.



Images from fvisin Francesco

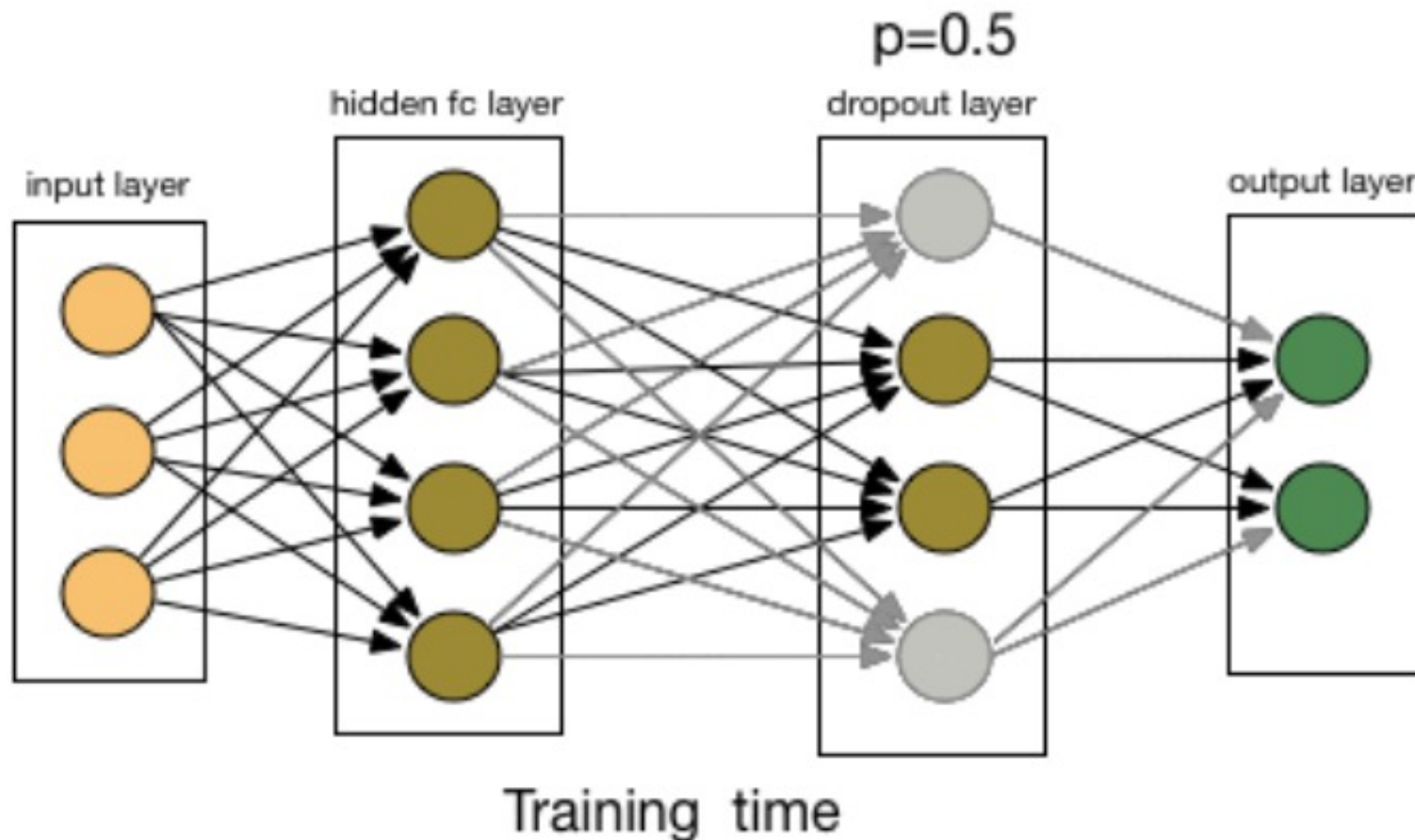
Régularisation : dropout

Utilisation de régularisation **Ridge/Lasso** pour éviter le surapprentissage.
On a aussi la possibilité d'appliquer un taux de **dropout**.



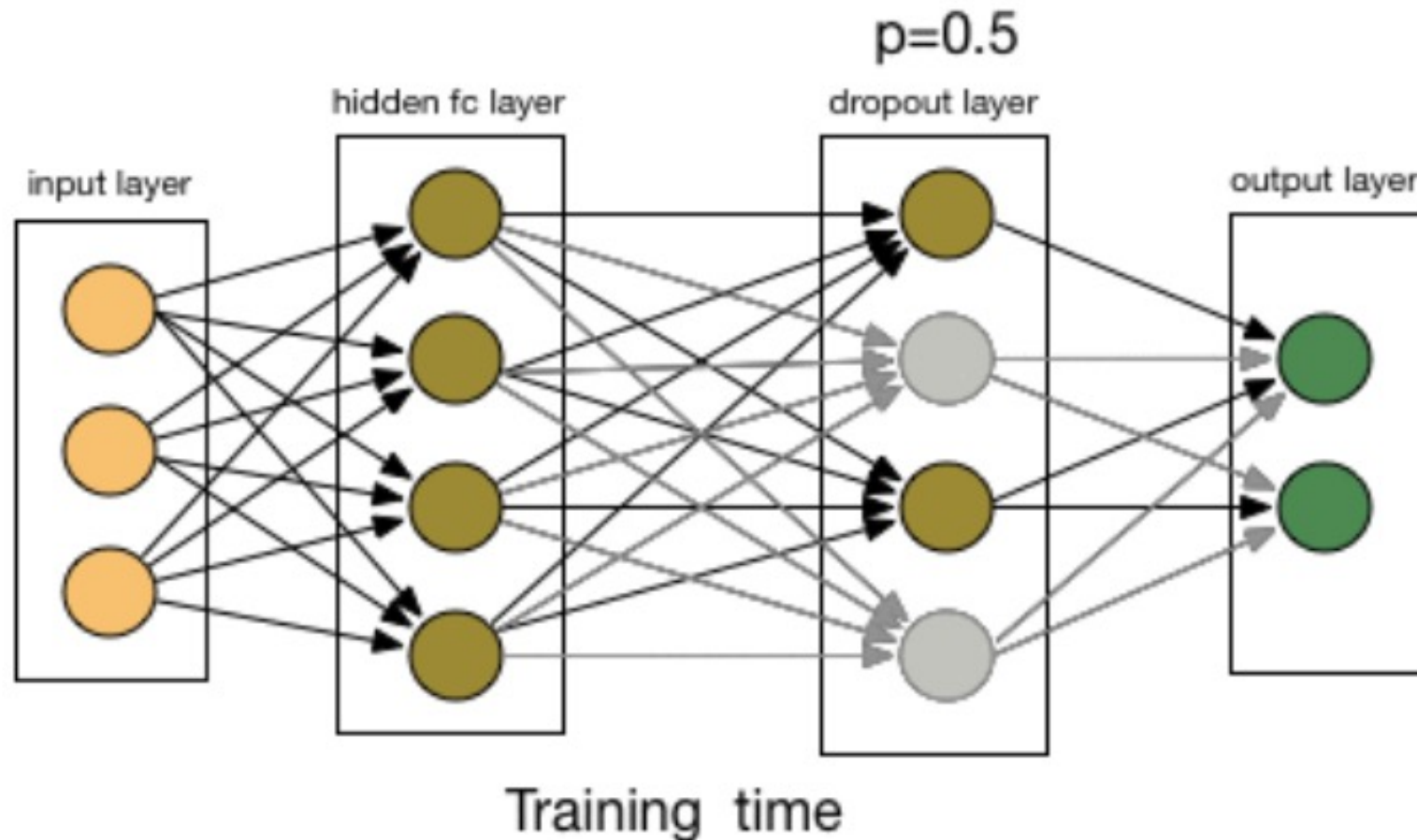
Régularisation : dropout

On désactive une proportion donnée (dropout rate) certains neurones tirés aléatoirement lors de l'apprentissage.

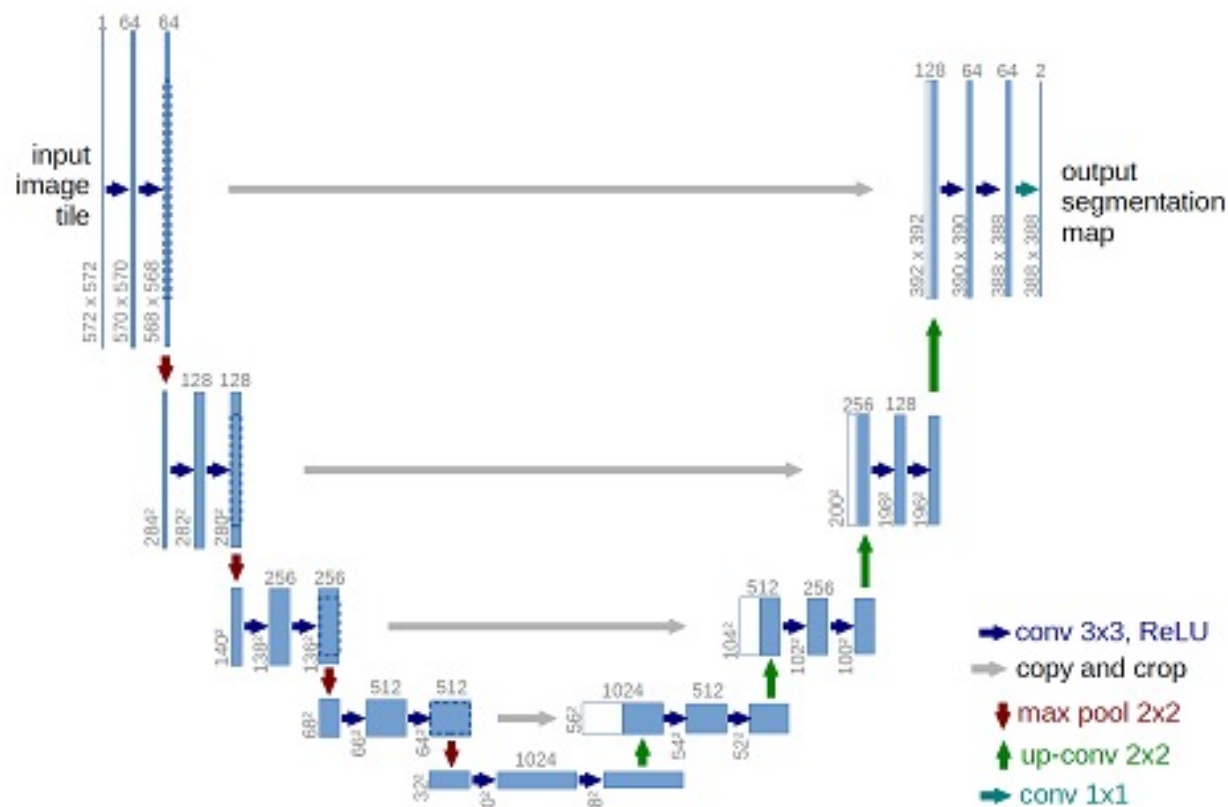


Régularisation : dropout

On désactive une proportion donnée (dropout rate) certains neurones tirés aléatoirement lors de l'apprentissage.



U-Net



Architecture adaptée à l'imagerie médicale très largement adaptée dans d'autre domaines pour la segmentation sémantique.

Ronneberger, O., Fischer, P., & Brox, T. *U-net: Convolutional networks for biomedical image segmentation*. arXiv, 2015. (55K citations)

Travaux pratiques séance 4

Identification des tourbillons océaniques à partir d'une observation de SLA de AVISO.

