Convolutional Neural Network

Deep learning par la pratique

I/ Introduction à la vision par ordinateur

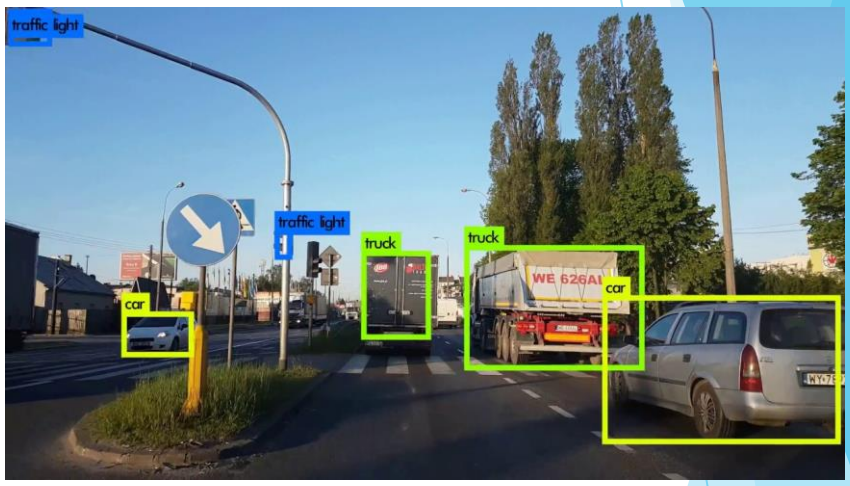
- **I/ Introduction à la vision par ordinateur**
- II/ Convolution & pooling layer
- III/ Exemples d'architectures
- IV/ Transfer learning

Computer vision

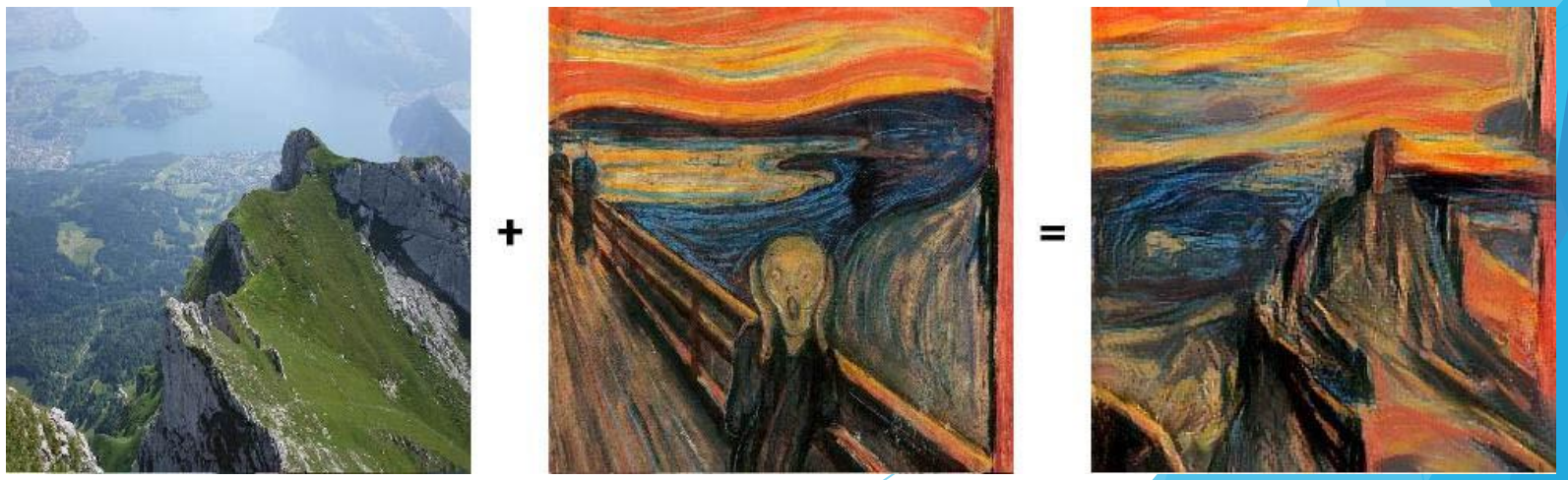
Classification d'images

Cat ? (0/1)

Détection d'objets



Transfert de style



Nombre de caractéristiques

64 x 64

64 x 64 x 3

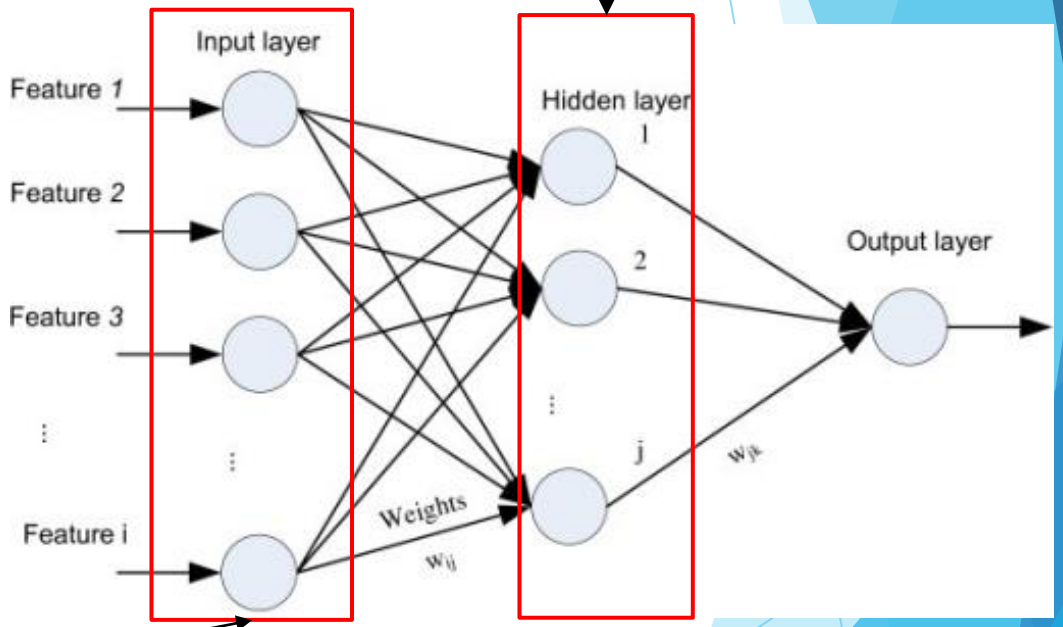
12288 caractéristiques

1000 x 1000

1000 x 1000 x 3

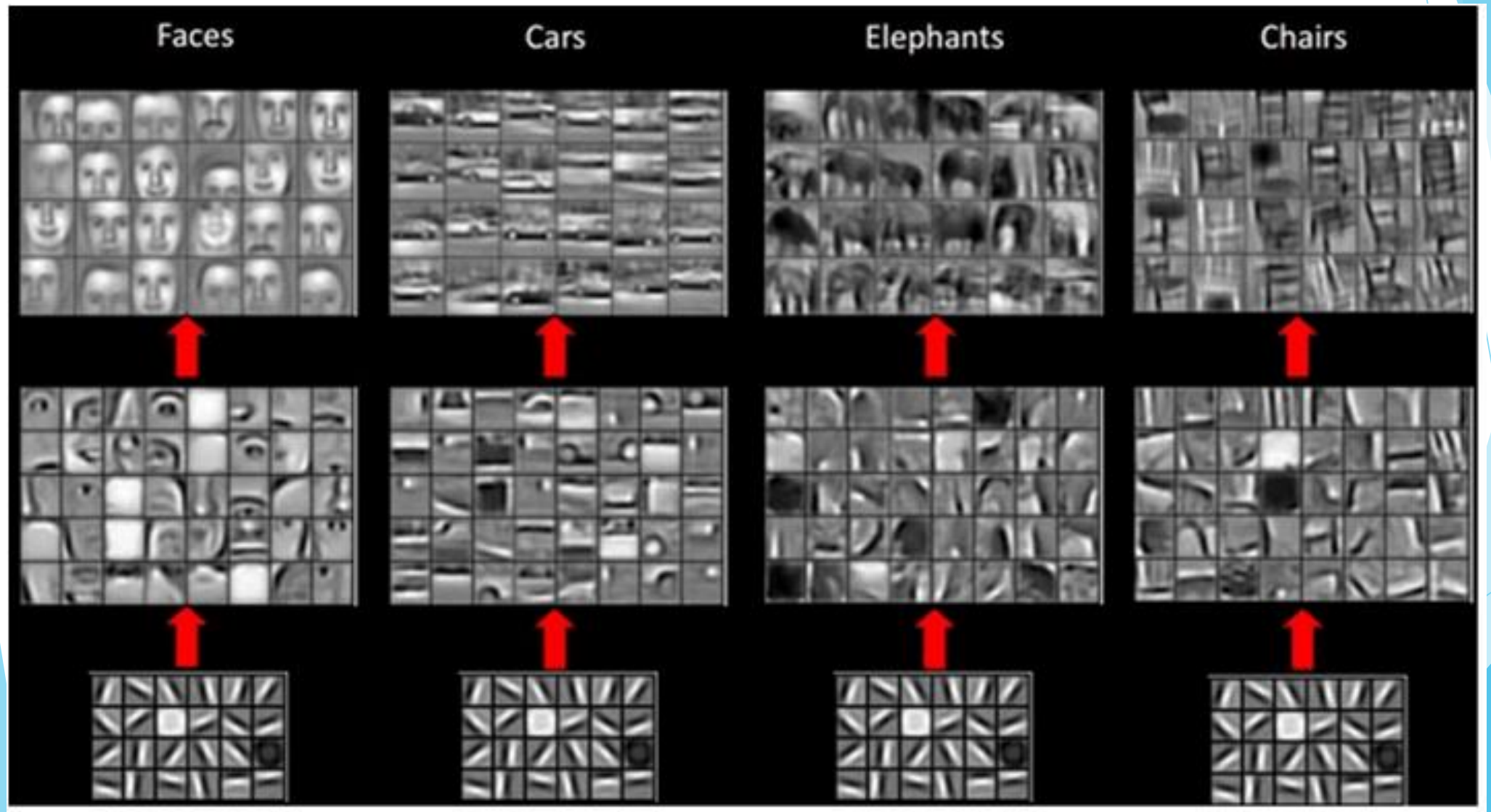
3 millions de caractéristiques

1 000 neurones



$W^{[1]} = 3$ milliards de paramètres

Extraction de caractéristiques



II/ Convolution & pooling layer

- I/ Introduction à la vision par ordinateur
- II/ Convolution & pooling layer
- III/ Exemples d'architectures
- IV/ Transfer learning

Convolution

3	0	1	2	7	4
1	5	8	9	3	1
2	7	2	5	1	3
0	1	3	1	7	8
4	2	1	6	2	8
2	4	5	2	3	9

(6, 6)

Convolution

*

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

(3, 3)

Filtre

w_1	w_2	w_3
w_4	w_5	w_6
w_7	w_8	w_9

=

-5	-4	0	8
-10	-2	2	3
0	-2	-4	-7
-3	-2	-3	-16

(4, 4)

Padding

3	0	1	2	7	4
1	5	8	9	3	1
2	7	2	5	1	3
0	1	3	1	7	8
4	2	1	6	2	8
2	4	5	2	3	9

(6, 6)

(n, n)

*

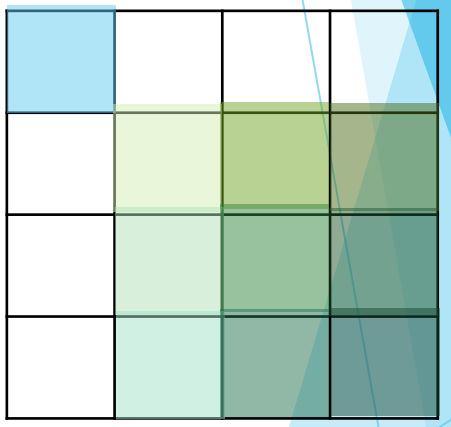
Filtre

w_1	w_2	w_3
w_4	w_5	w_6
w_7	w_8	w_9

(3, 3)

(f, f)

=



(4, 4)

(n - f + 1, n - f + 1)

Padding

	3	0	1	2	7	4	
	1	5	8	9	3	1	
	2	7	2	5	1	3	
	0	1	3	1	7	8	
	4	2	1	6	2	8	
	2	4	5	2	3	9	

(6, 6)

(n, n)

(8, 8)

(n + 2p, n + 2p)

Padding
p = 1

*

Filtre

w_1	w_2	w_3
w_4	w_5	w_6
w_7	w_8	w_9

(3, 3)

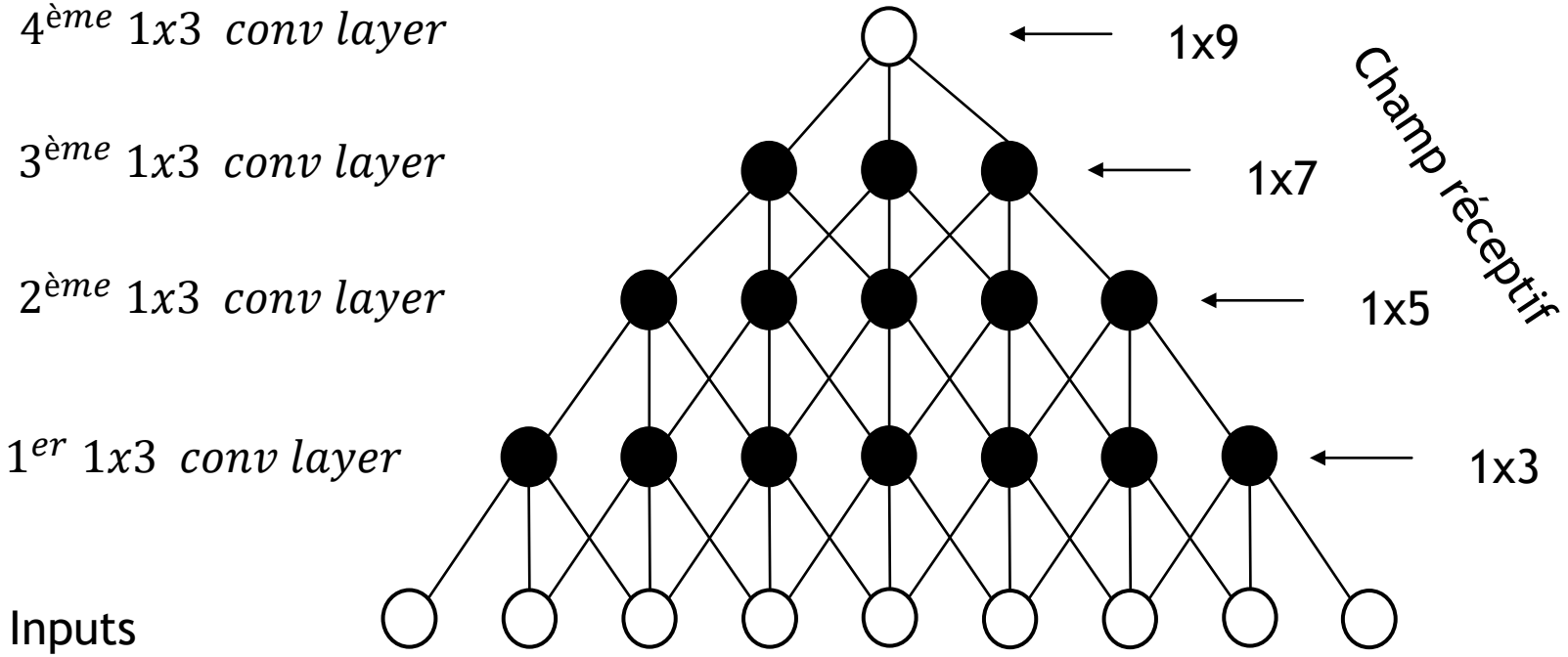
(f, f)

=

(6, 6)

(n + 2p – f + 1, n + 2p – f + 1)

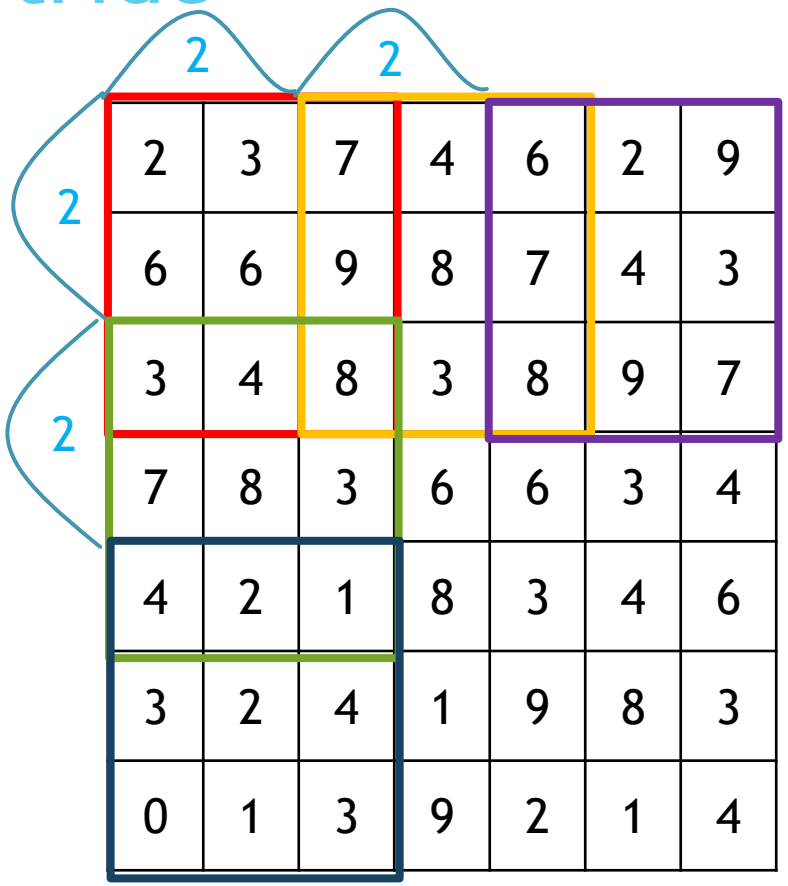
Champ réceptif



Si nous empilons N couches convolutionnelles avec le même filtre de taille 3x3 champ réceptif sur la Nième couche sera $2N + 1\ x\ 2N + 1$

It looks like we need to stack lot of layers!!

Stride



2	3	7	4	6	2	9
6	6	9	8	7	4	3
3	4	8	3	8	9	7
7	8	3	6	6	3	4
4	2	1	8	3	4	6
3	2	4	1	9	8	3
0	1	3	9	2	1	4

(7, 7)

(n + 2p, n + 2p)

Stride
s = 2

filter

w_1	w_2	w_3
w_4	w_5	w_6
w_7	w_8	w_9

(3, 3)

(f, f)

*

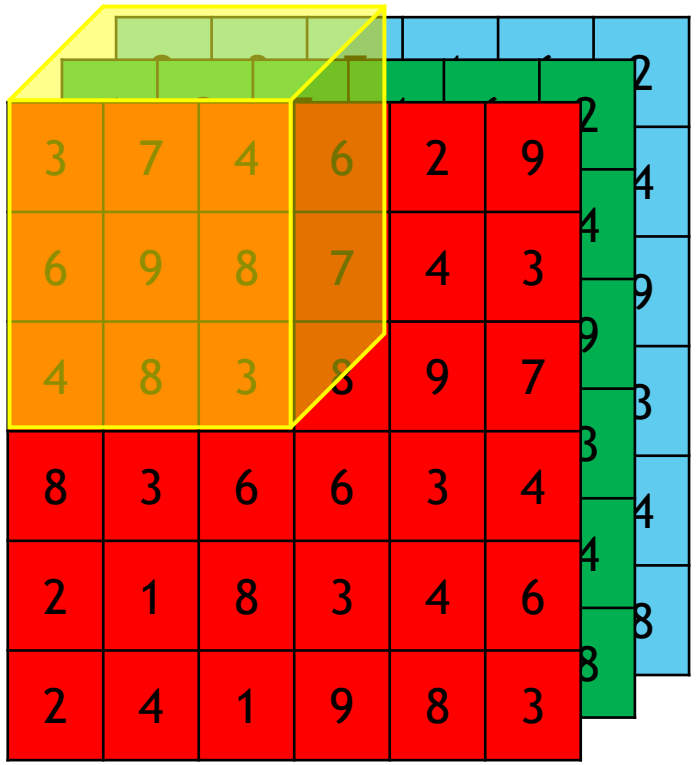
=

91	100	83
69	91	127
44	72	74

(3, 3)

$$\left(\frac{n+2p-f}{s}+1, \frac{n+2p-f}{s}+1\right)$$

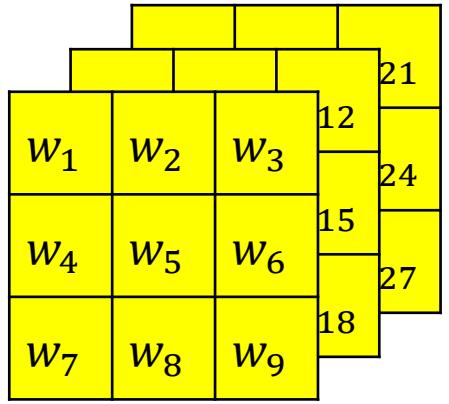
Convolutions pour les images RGB



(6, 6, 3)

Hauteur — Largeur — # channels

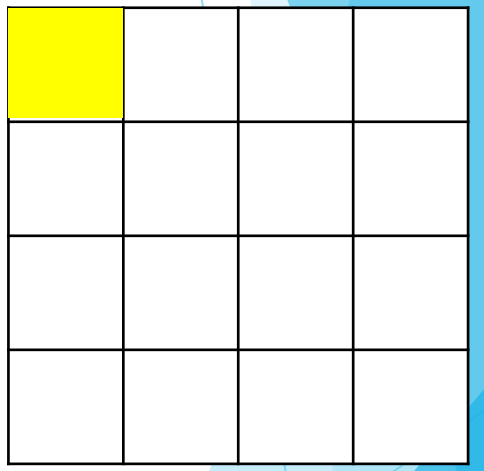
*



(3, 3, 3)

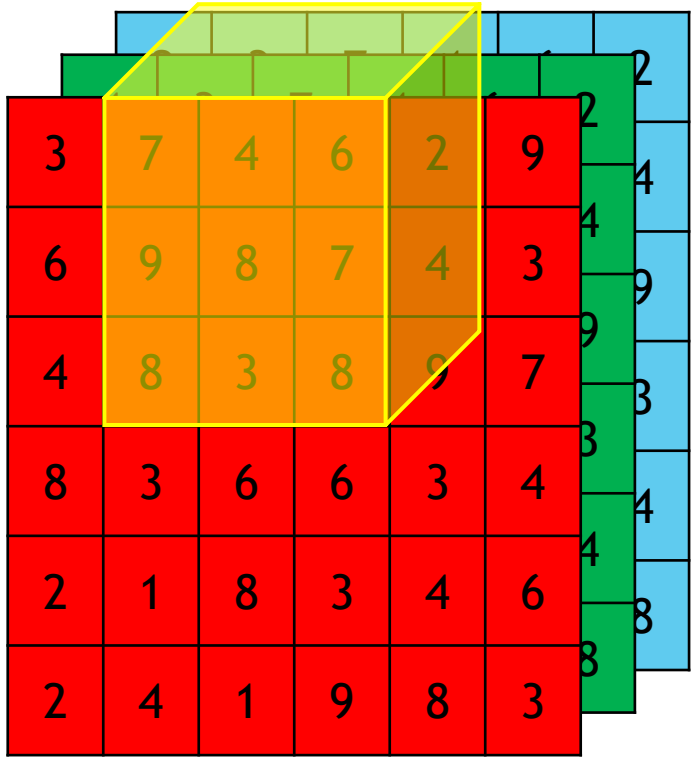
Hauteur — Largeur — # channels

=



(4, 4, 1)

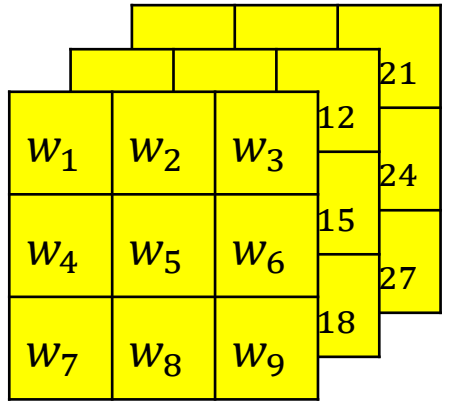
Convolutions pour les images RGB



(6, 6, 3)

Hauteur Largeur # channels

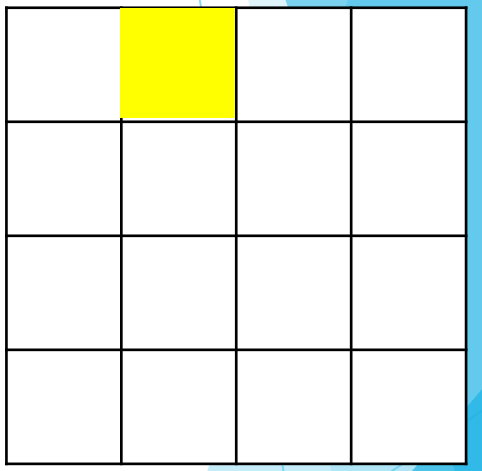
*



(3, 3, 3)

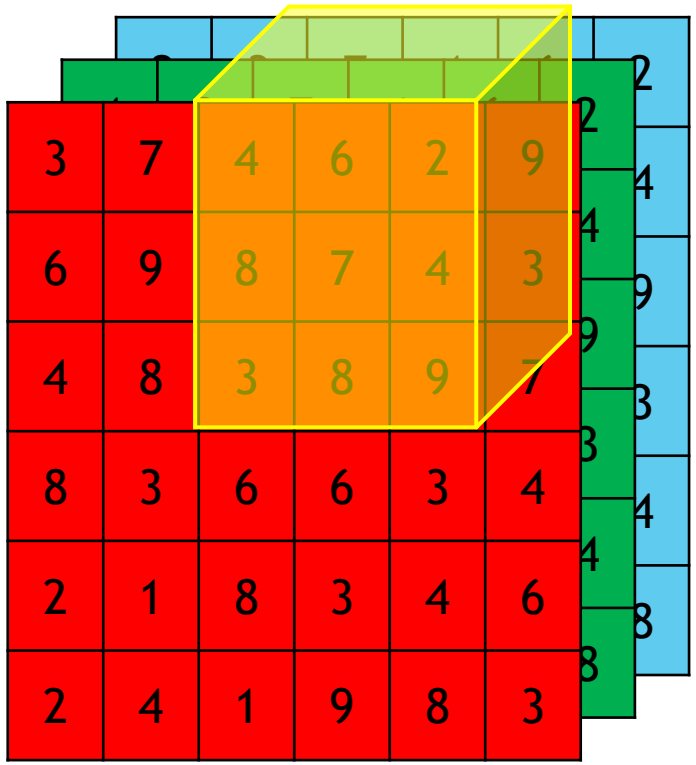
Hauteur Largeur # channels

=



(4, 4, 1)

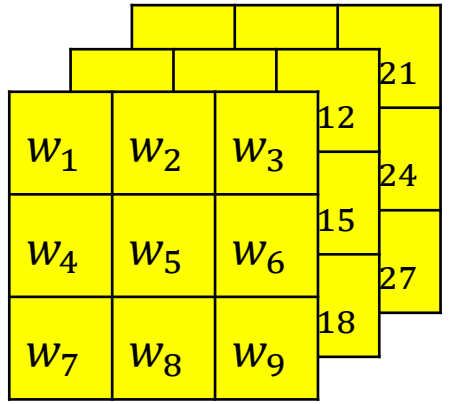
Convolutions pour les images RGB



(6, 6, 3)

Hauteur Largeur # channels

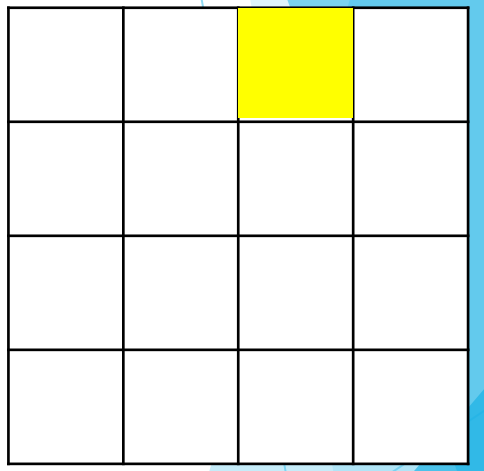
*



(3, 3, 3)

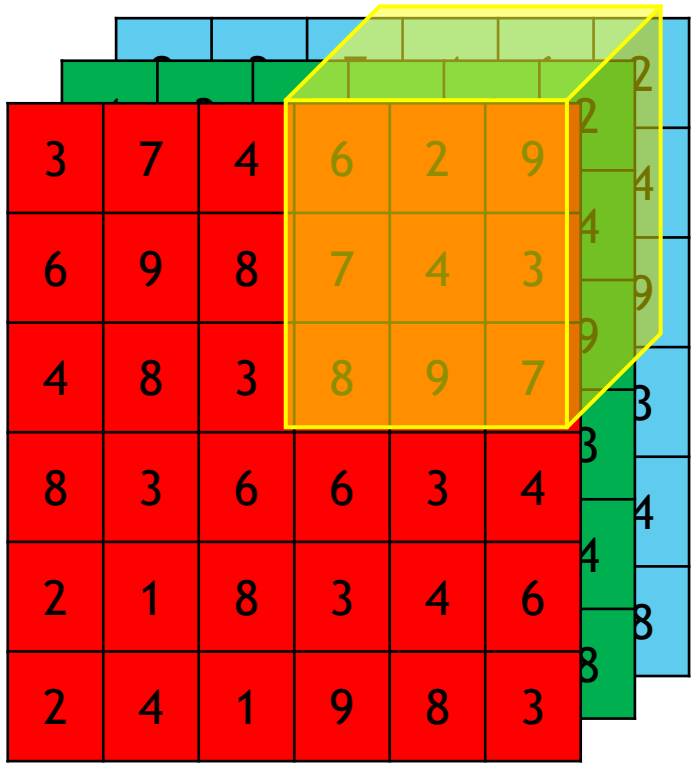
Hauteur Largeur # channels

=



(4, 4, 1)

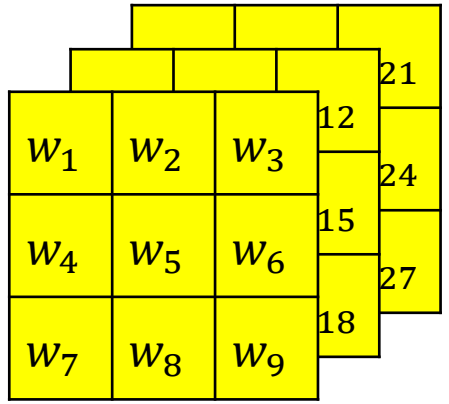
Convolutions pour les images RGB



(6, 6, 3)

Hauteur Largeur # channels

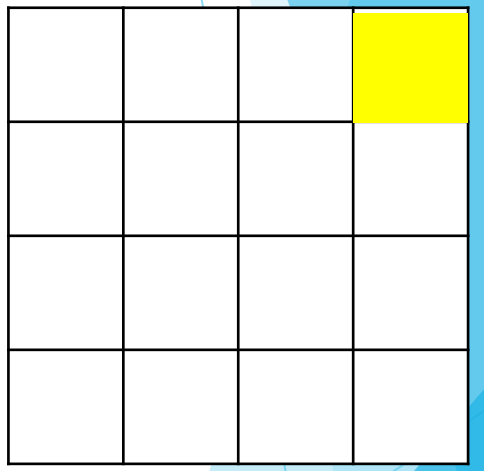
*



(3, 3, 3)

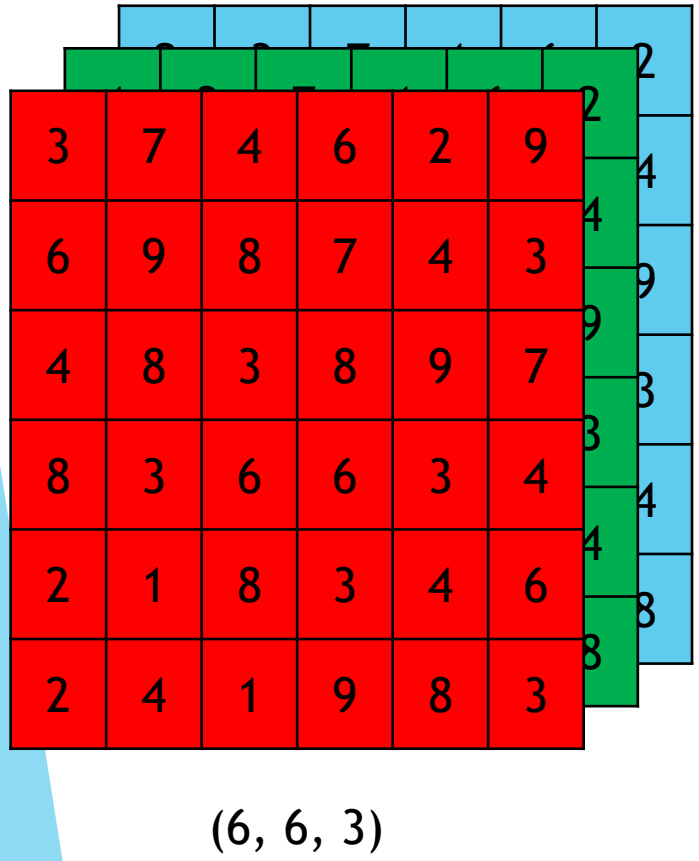
Hauteur Largeur # channels

=



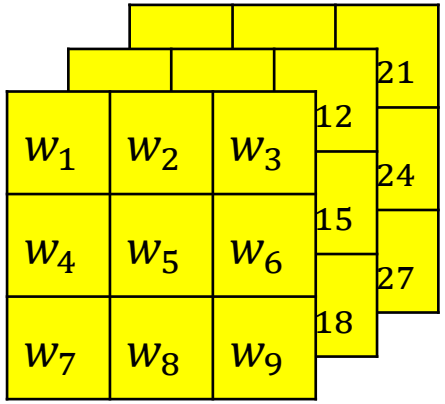
(4, 4, 1)

Plusieurs filtres

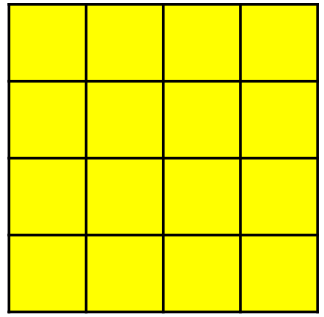


(6, 6, 3)

*

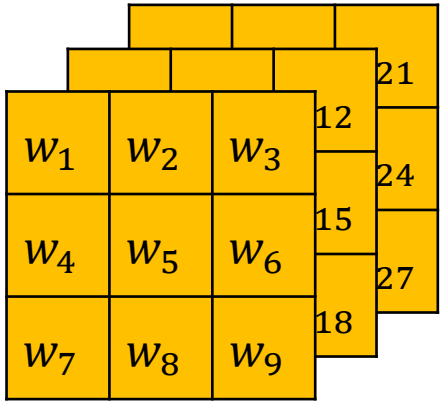


=

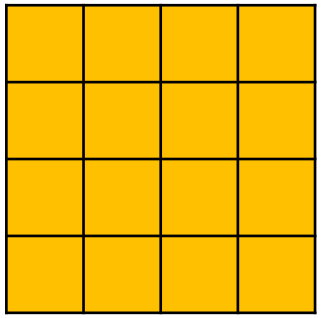


(4, 4, 1)

*

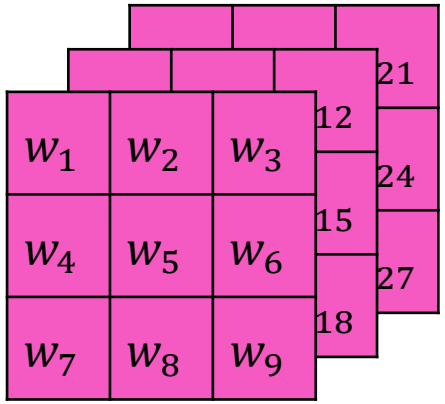


=



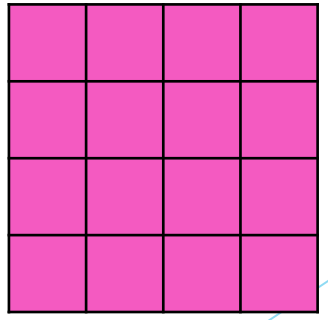
(4, 4, 1)

*



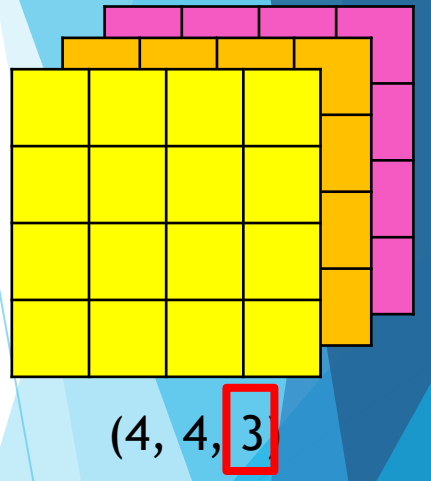
(3, 3, 3)

=



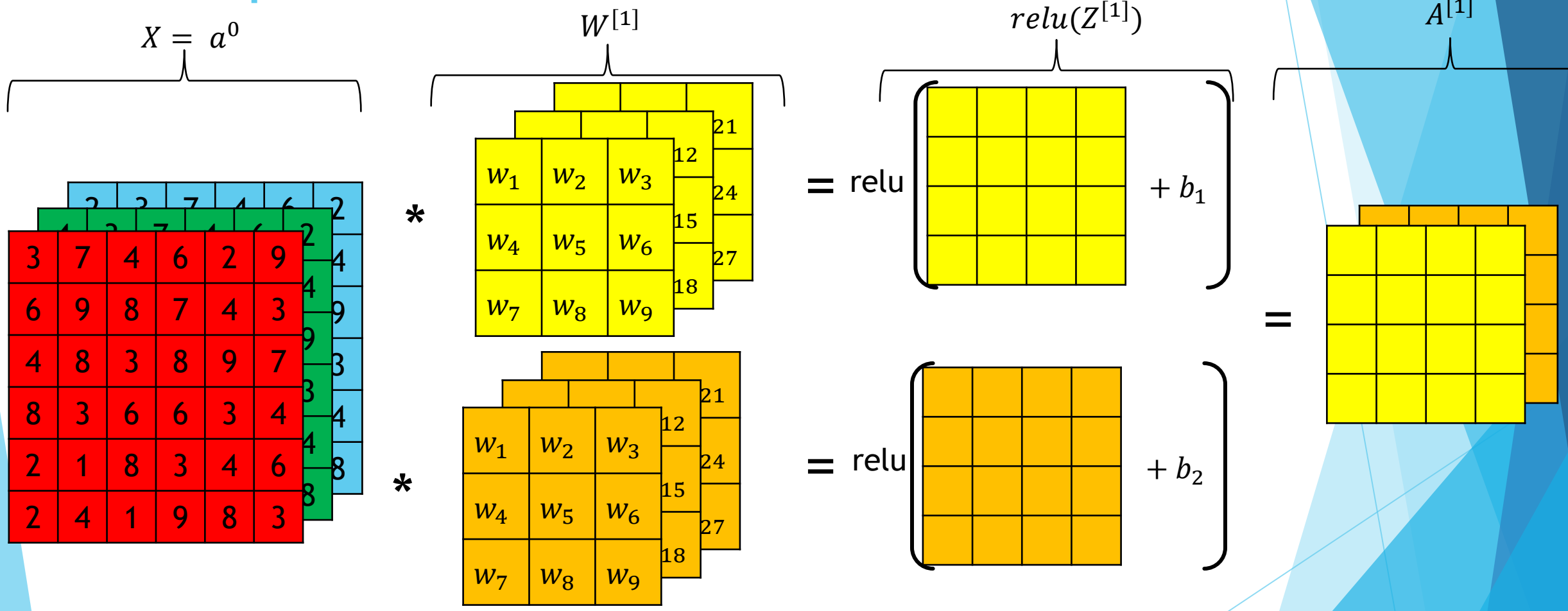
(4, 4, 1)

=



(4, 4, 3)

Exemples de couches



Sommaire des notations

Si une couche *l* est une couche convolutionelle :

$f^{[l]}$= taille du filtre

$p^{[l]}$= padding

$s^{[l]}$= stride

$n_c{}^{[l]}$= nombre de filtres

Dimension d'entrée : $(n_H{}^{[l-1]}, n_w{}^{[l-1]}, n_c{}^{[l-1]})$

Dimension de sortie : $(n_H{}^{[l]}, n_w{}^{[l]}, n_c{}^{[l]})$

$$n_h{}^{[l]} = \left\lfloor \frac{n_h{}^{[l-1]} + 2p^{[l]} - f^{[l]}}{s^{[l]}} + 1 \right\rfloor$$

$$n_w{}^{[l]} = \left\lfloor \frac{n_w{}^{[l-1]} + 2p^{[l]} - f^{[l]}}{s^{[l]}} + 1 \right\rfloor$$

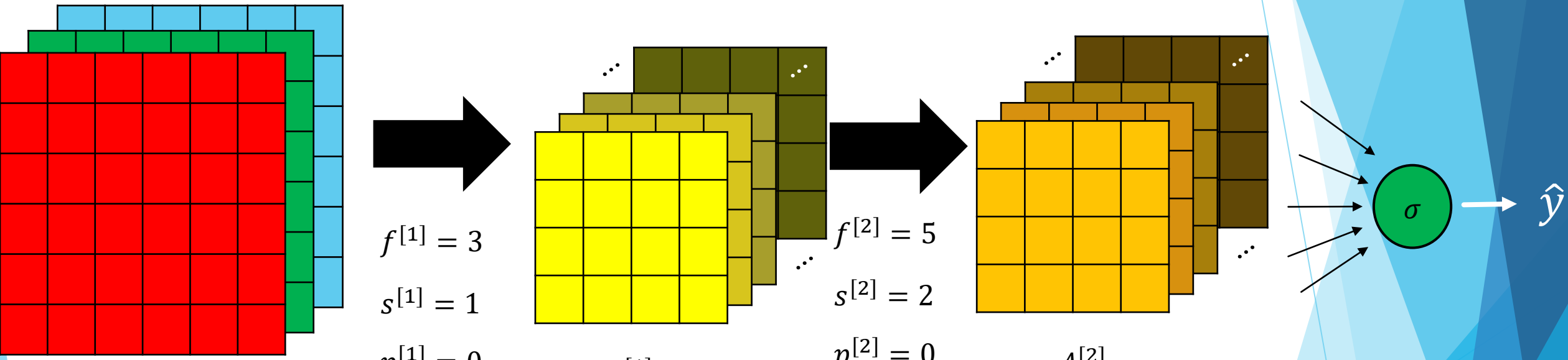
Chaque filtre : $(f^{[l]}, f^{[l]}, n_c{}^{[l-1]})$

L'activation : $a^{[l]} \rightarrow (n_H{}^{[l]}, n_w{}^{[l]}, n_c{}^{[l]})$ $\quad A^{[l]} \rightarrow (m, n_H{}^{[l]}, n_w{}^{[l]}, n_c{}^{[l]})$

Les poids : $W^{[l]} \rightarrow (f^{[l]}, f^{[l]}, n_c{}^{[l-1]}, n_c{}^{[l]})$

Le bias : $b^{[l]} \rightarrow (n_c{}^{[l]})$

Exemple d'architecture simple



X

(39, 39, 3)

$n_H^{[0]} = n_w^{[0]} = 39$

$n_c^{[0]} = 3$

$f^{[1]} = 3$

$s^{[1]} = 1$

$p^{[1]} = 0$

10 filtres

$A^{[1]}$

(37, 37, 10)

$n_H^{[1]} = n_w^{[1]} = 37$

$n_c^{[1]} = 10$

$f^{[2]} = 5$

$s^{[2]} = 2$

$p^{[2]} = 0$

20 filtres

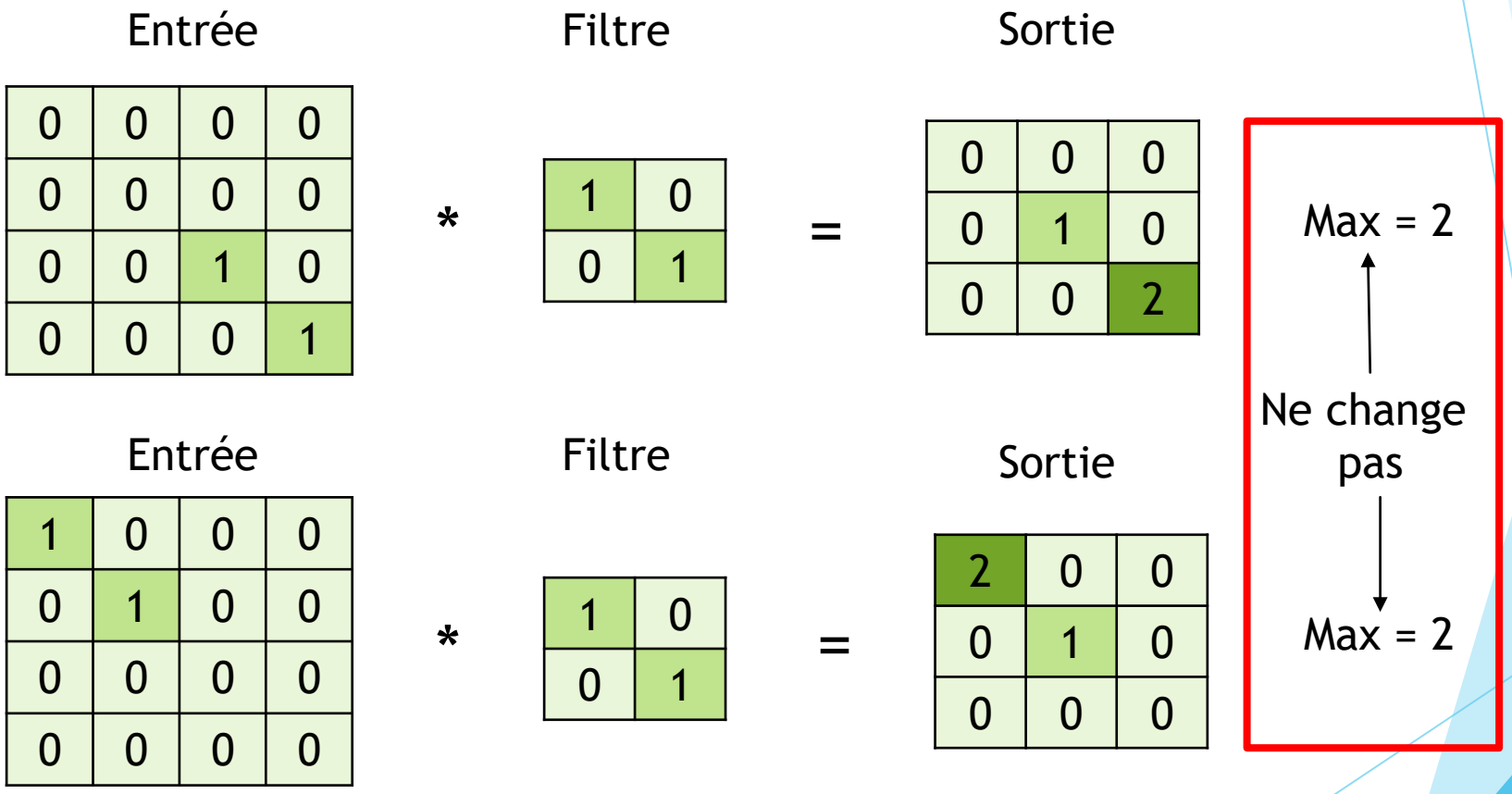
$A^{[2]}$

(17, 17, 20)

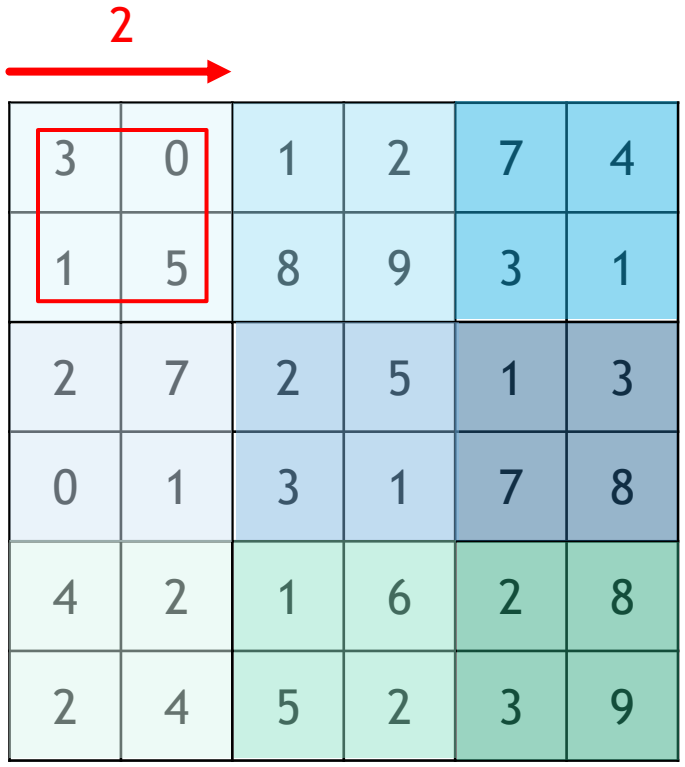
$n_H^{[1]} = n_w^{[1]} = 17$

$n_c^{[1]} = 20$

Comment maintenir l'invariance en translation ?



Pooling layer: Max pooling

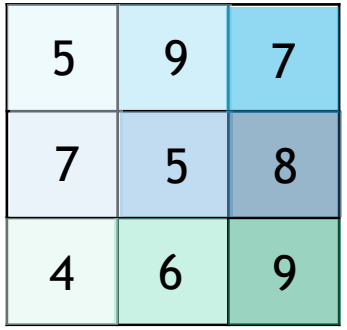


(6, 6)

Hyperparameters:

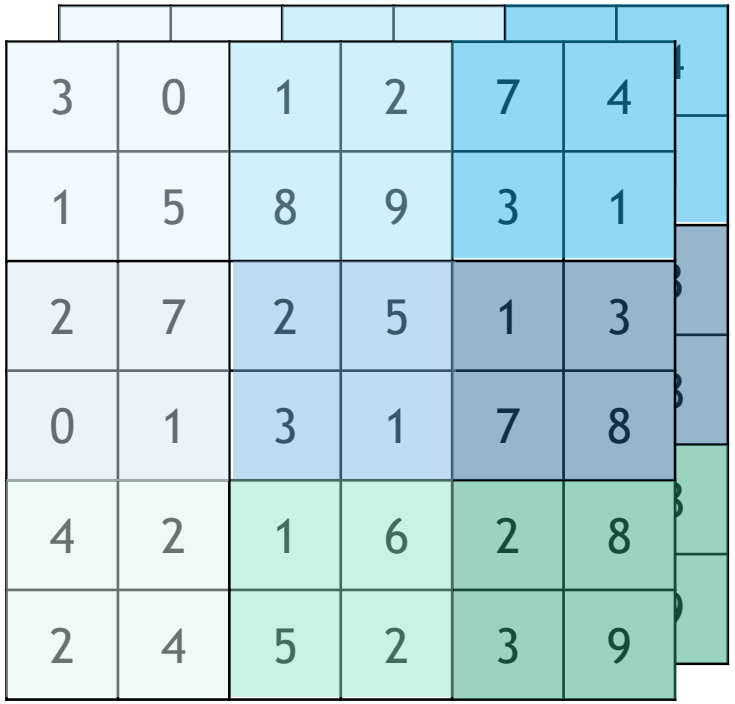
$f = 2$

$s = 2$



(3, 3)

Pooling layer: Max pooling

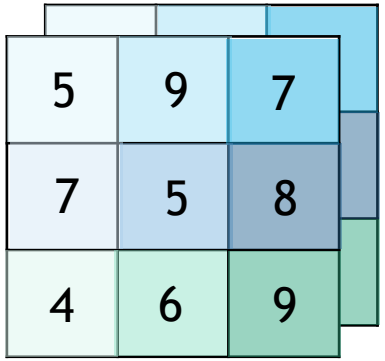


(6, 6, 2)

Hyperparameters:

$f = 2$

$s = 2$

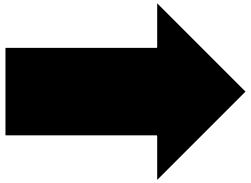


(3, 3, 2)

Pooling layer: Average pooling

3	0	1	2	7	4
1	5	8	9	3	1
2	7	2	5	1	3
0	1	3	1	7	8
4	2	1	6	2	8
2	4	5	2	3	9

(6, 6)



Hyperparameters:

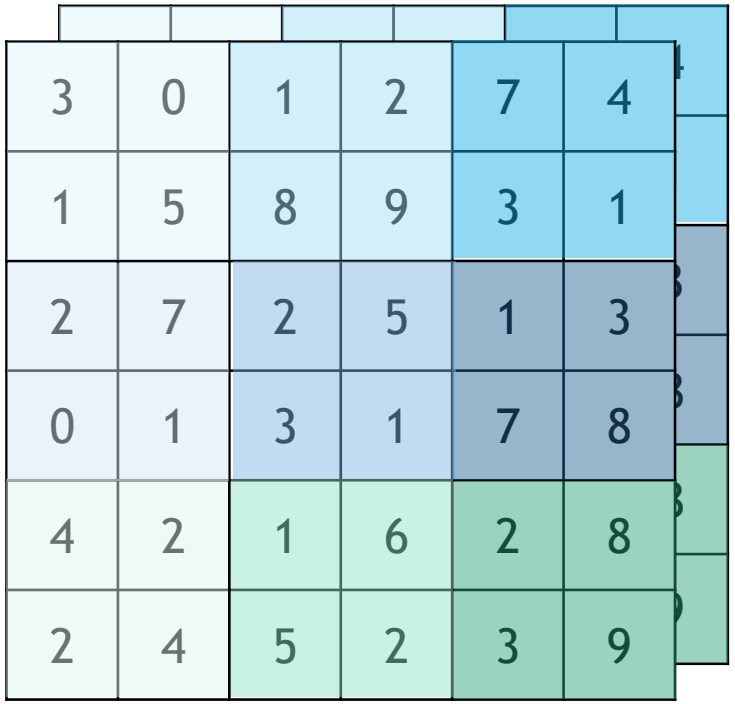
$f = 2$

$s = 2$

2.25	5	3.75
2.5	2.75	4.75
3	3.5	5.5

(3, 3)

Pooling layer: Average pooling

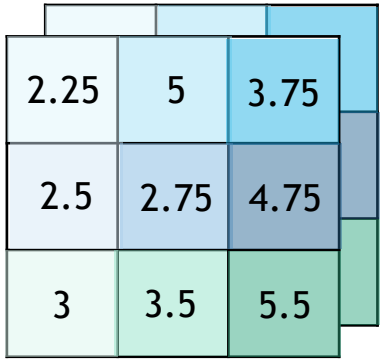


(6, 6, 2)

Hyperparameters:

$f = 2$

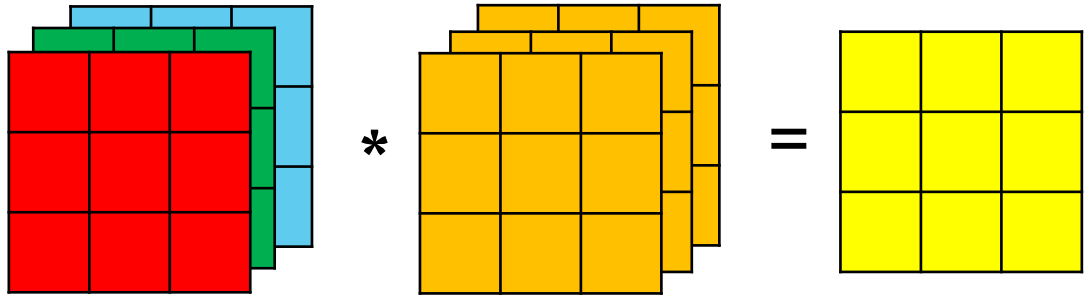
$s = 2$



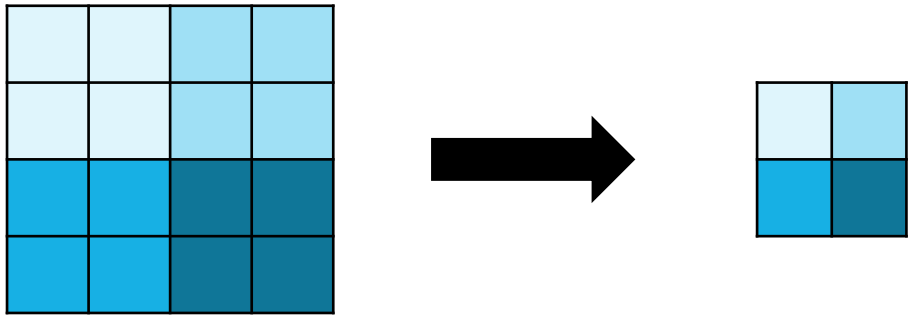
(3, 3, 2)

Trois blocs d'un CNN

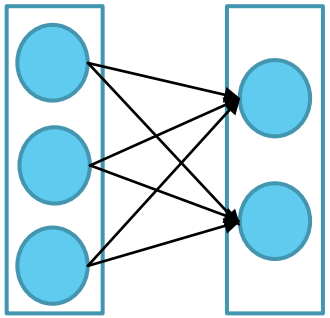
Convolutional bloc



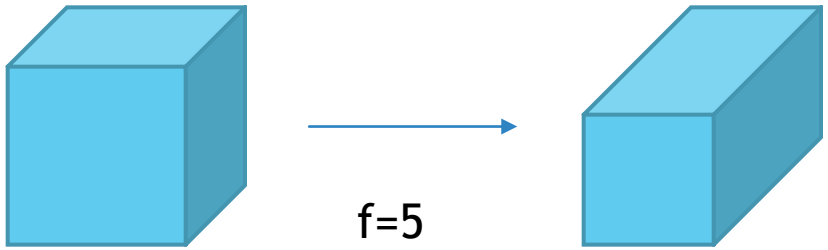
Pooling bloc



Fully connected bloc



Pourquoi des convolutions ?



(32, 32, 3)
3 072

6 filtres

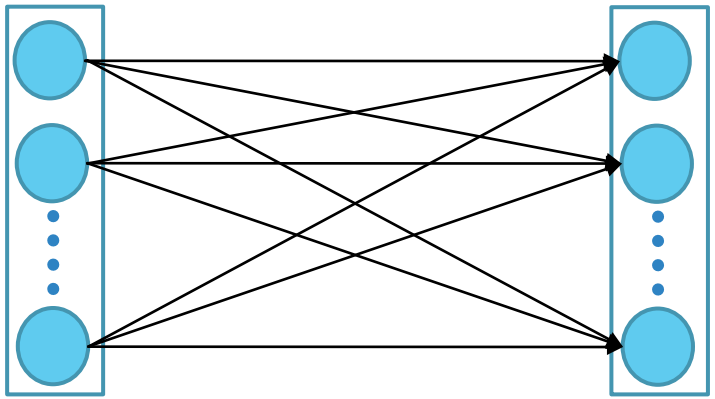
(28, 28, 6)
4 704

f=5 $5 \times 5 = 25$

bias $25 + 1 = 26$

6 filters $26 \times 6 = 156$

156 paramètres



3 072

4 704

$3\,072 \times 4\,704 \approx 14\,000\,000$

14 millions de paramètres

Pourquoi des convolutions ?

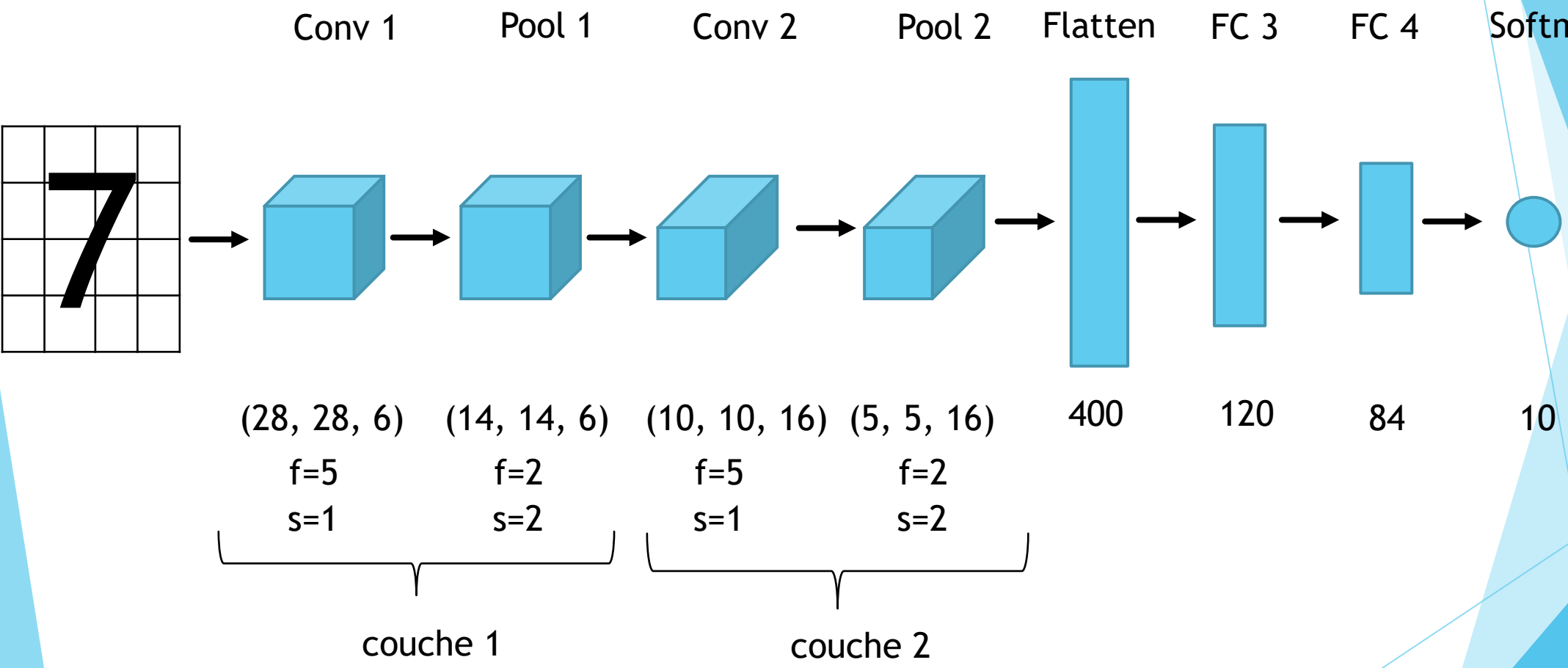
- **Partage des paramètres :** Un détecteur de caractéristiques qui est utile dans une partie de l'image l'est probablement dans une autre partie de l'image.
- **Sparsité des connexions :** Dans chaque couche, chaque valeur de sortie ne dépend que d'un petit nombre d'entrées.
- **Invariance de la traduction :** Si vous translatez l'image, cela ne changera rien pour un réseau neuronal convolutif.

III/ Exemples d'architectures

- I/ Introduction à la vision par ordinateur
- II/ Convolution & pooling layer
- **III/ Exemples d'architectures**
- IV/ Transfer learning

LeNet-5

$\approx 60k$ *paramètres*

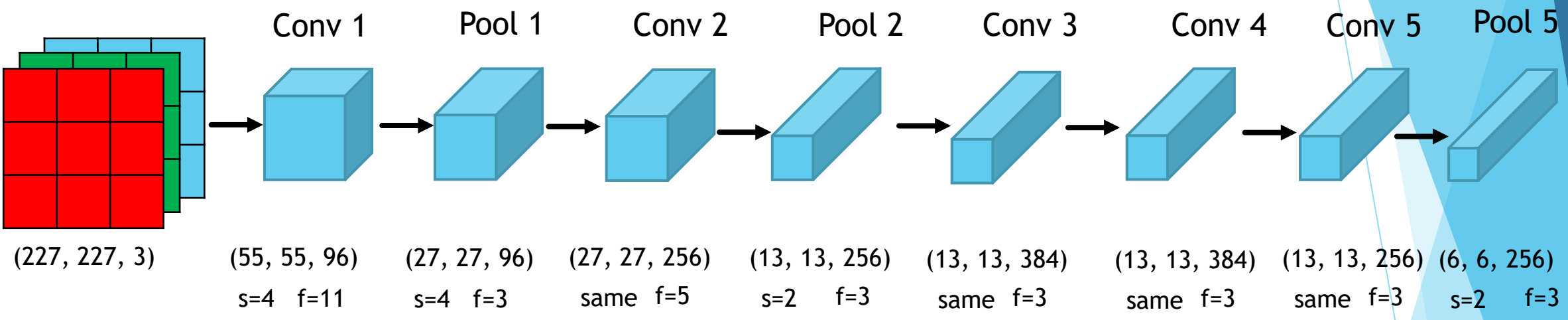


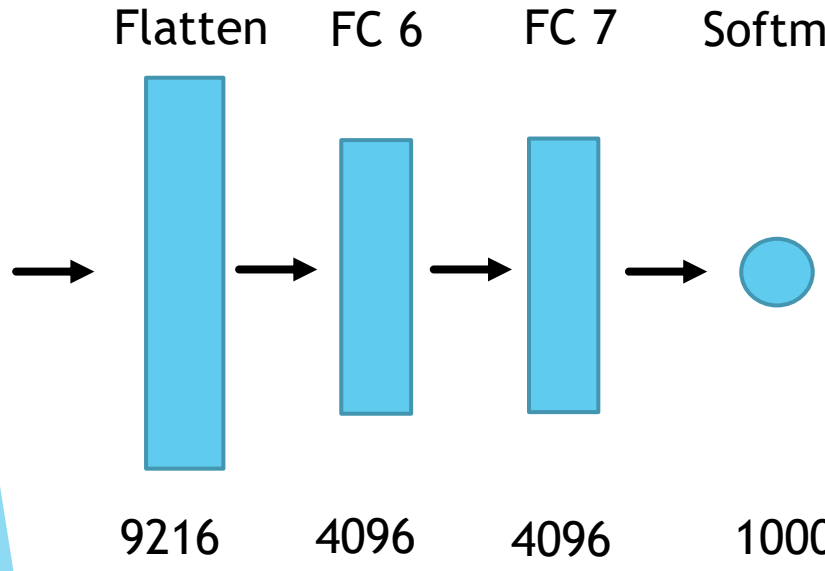
LeNet-5

	Activation shape	Activation size	# parameters
Input:	(32, 32, 3)	3 072	0
CONV1 (f=5, s=1)	(28,28, 8)	6 272	208
POOL1	(14, 14, 8)	1 568	0
CONV2 (f=5, s=1)	(10, 10, 46)	1 600	416
POOL2	(5, 5, 16)	400	0
FC3	(120, 1)	120	48 001
FC4	(84, 1)	84	10 081
Softmax	(10, 1)	10	841

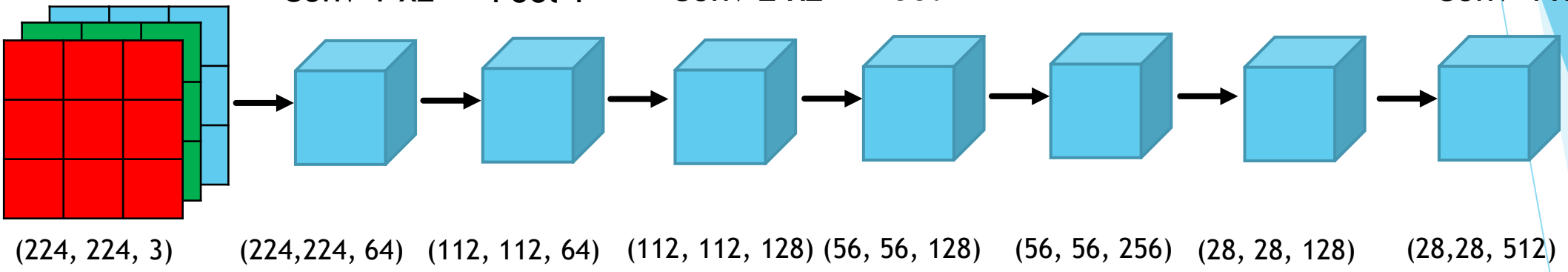
AlexNet

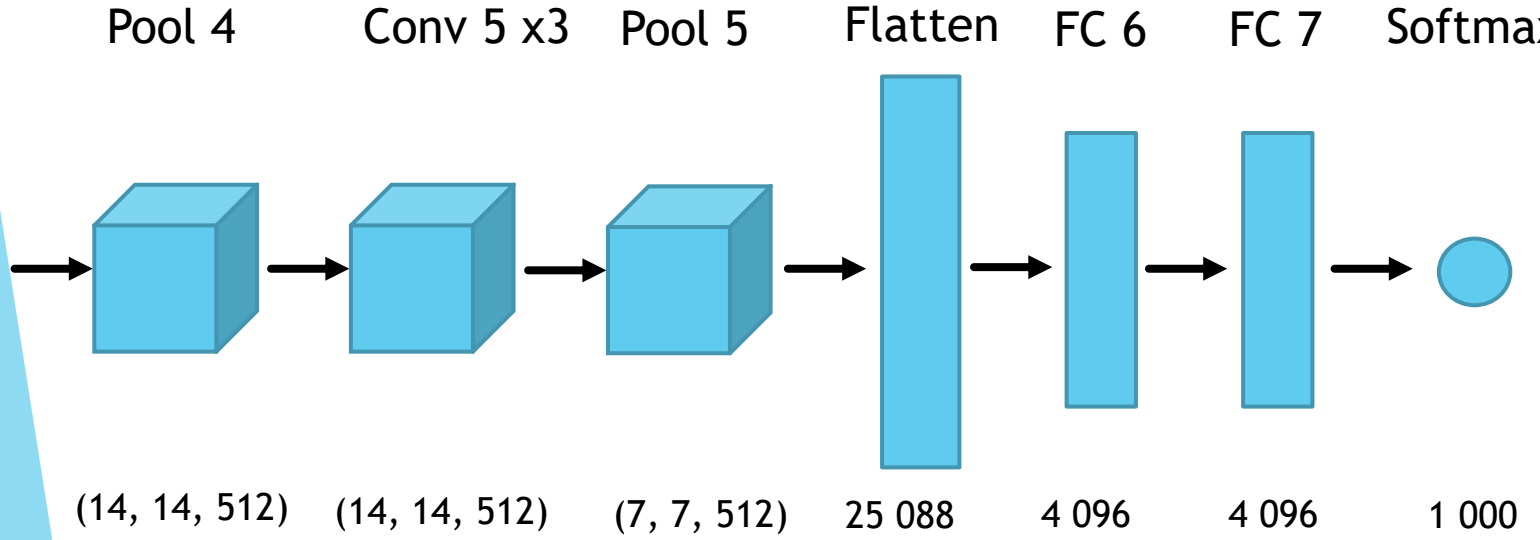
≈ 60 *millions de paramètres*





VGG-16 $\approx 138\ millions$ de paramètres





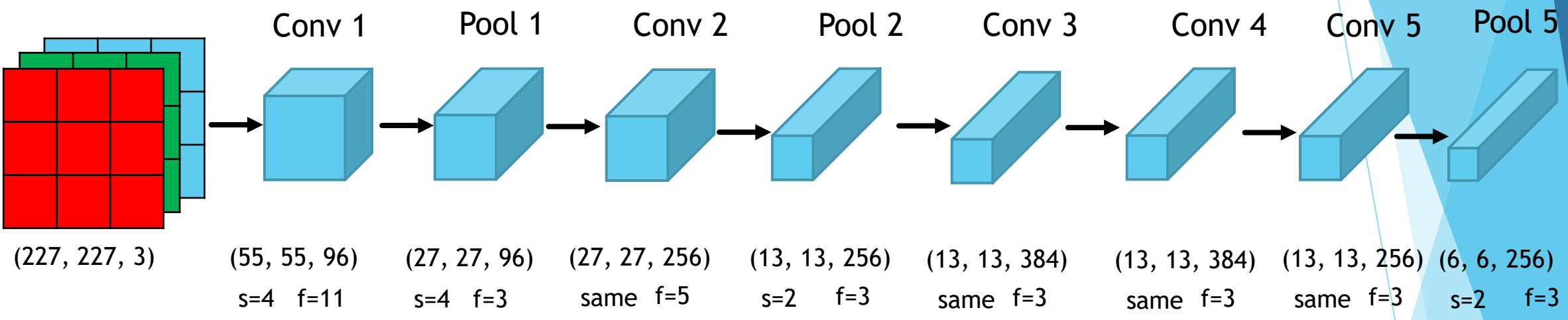
Max-pool
Filter = (2, 2)
Stride = 2

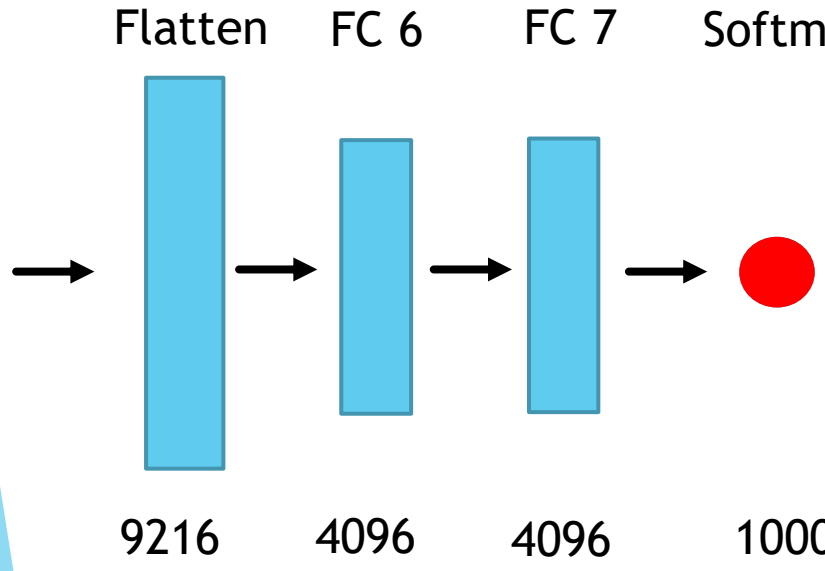
Conv
Padding = same
Filter = (3, 3)
Stride = 1

IV/ Transfer learning

- I/ Introduction à la vision par ordinateur
- II/ Convolution & pooling layer
- III/ Exemples d'architectures
- **IV/ Transfer learning**

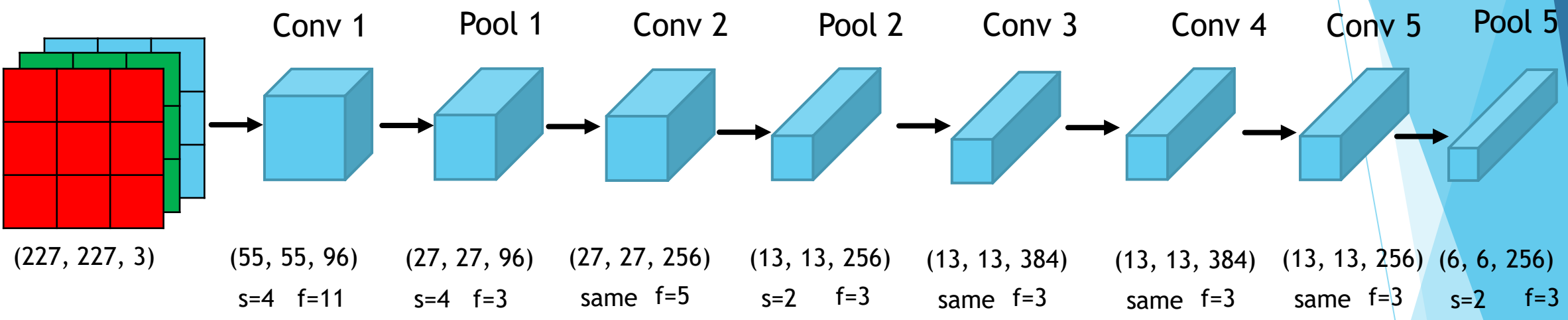
Transfer learning avec peu de données

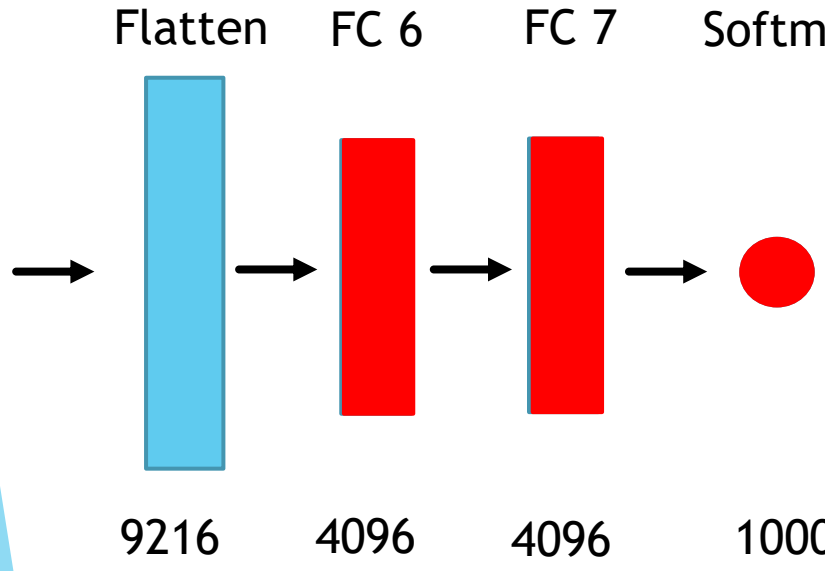




Pour adapter votre modèle,
ne conservez que la dernière couche

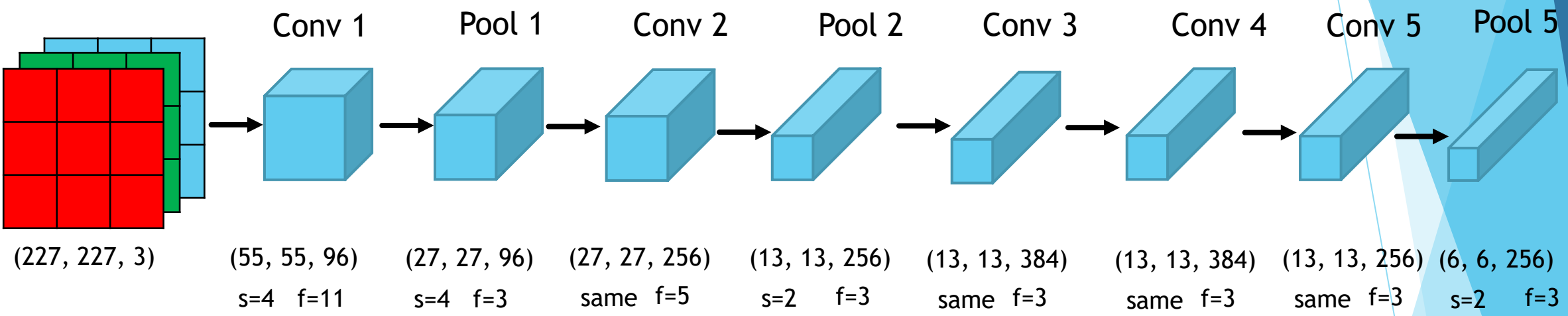
Transfer learning avec plus de données

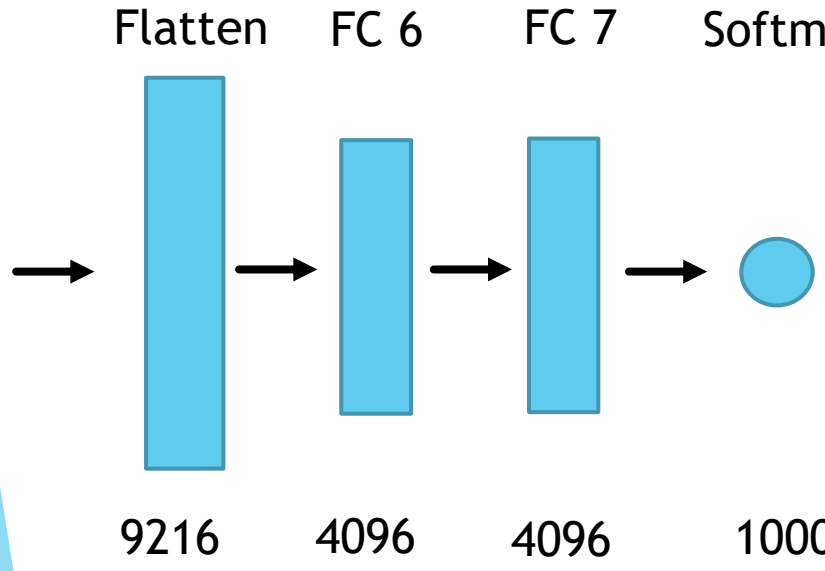




Pour adapter votre modèle,
ne conservez que les dernières couches

Transfer learning avec beaucoup de données





Utiliser les poids du modèle comme initialisation

A retenir

	Domaine d'imagenet	Domaine différent d'imagenet
Petit dataset	Les dernières couches	Collecter plus de données
Grand dataset	Utiliser comme initialisation	Initialisation aléatoire