

Réseaux de neurones

IFT 780

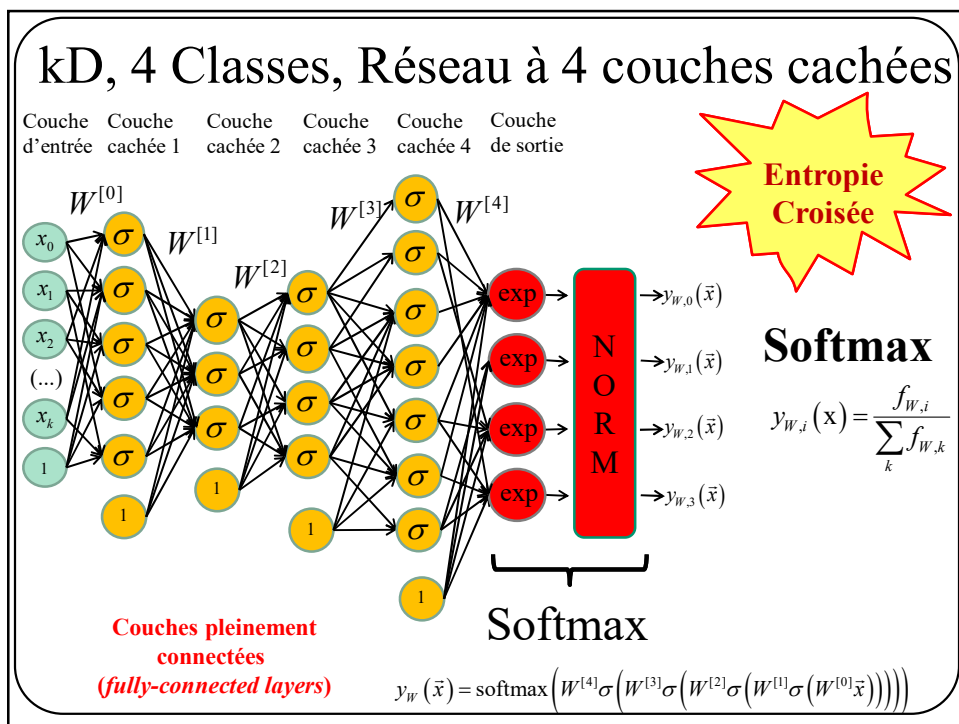
Réseaux à convolution

Par  
Pierre-Marc Jodoin

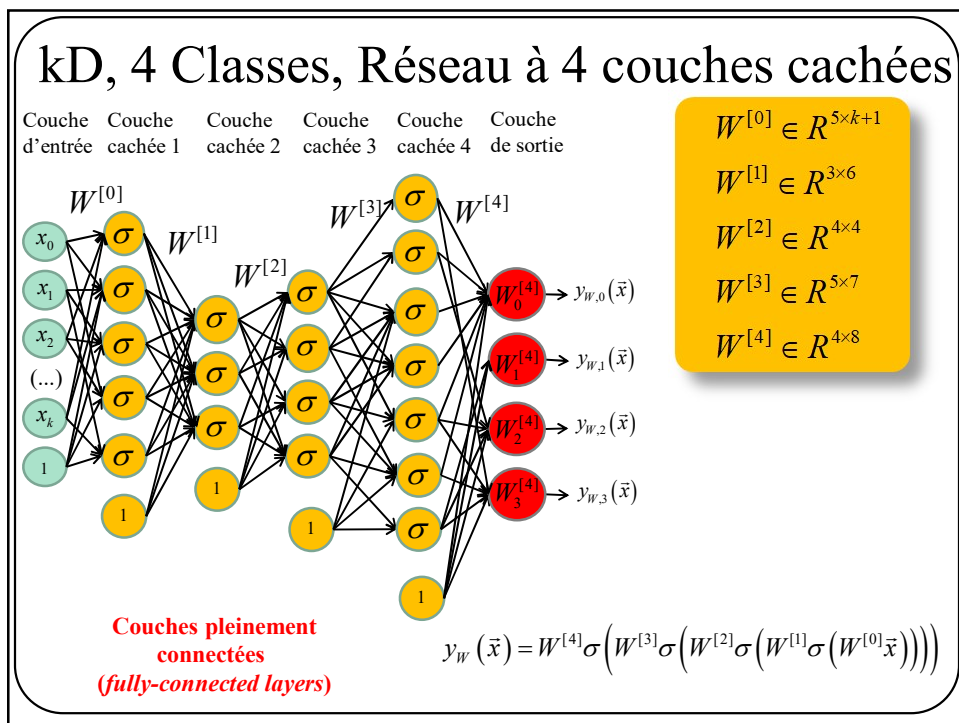
Par

1



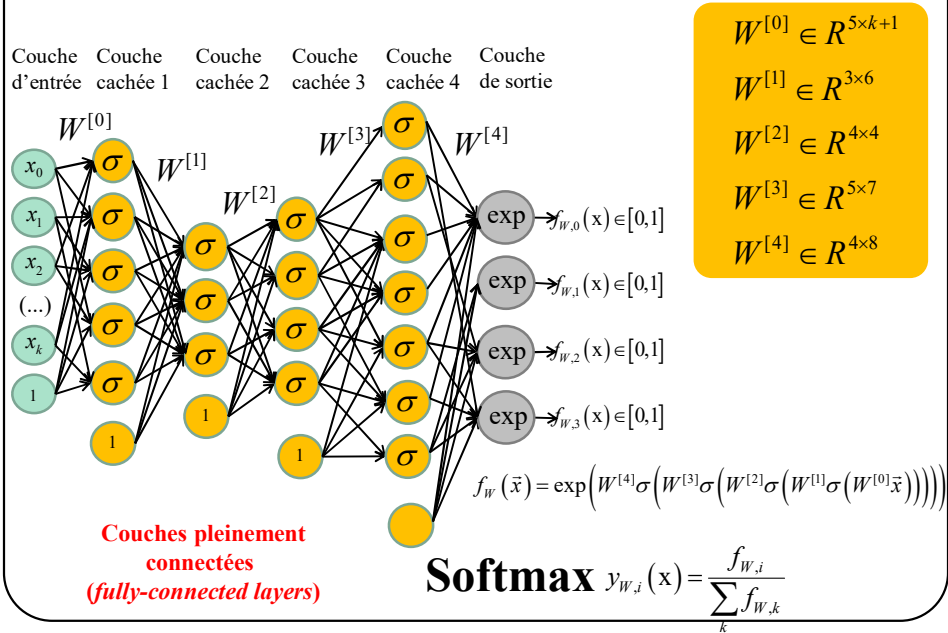


3



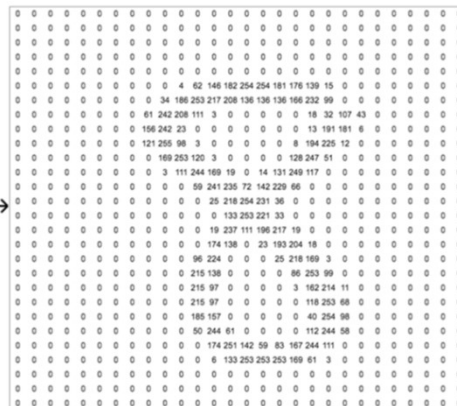
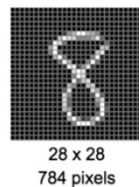
4

## kD, 4 Classes, Réseau à 4 couches cachées



5

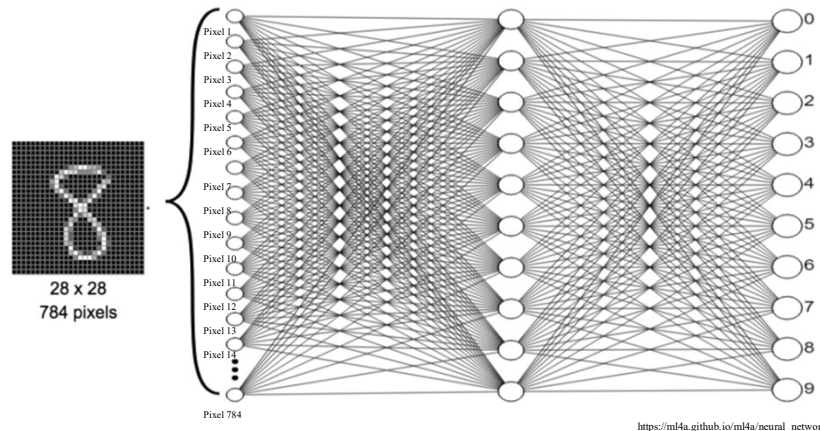
## Comment classifier des images?



[https://ml4a.github.io/ml4a/neural\\_networks/](https://ml4a.github.io/ml4a/neural_networks/)

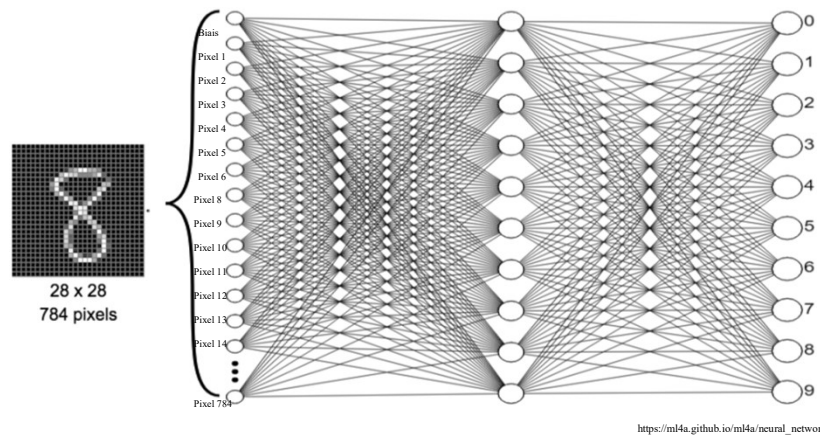
6

## Comment classifier des images?



7

## Beaucoup de paramètres (7850 dans la couche 1)



8

Beaucoup trop de paramètres  
(655,370 dans la couche 1)

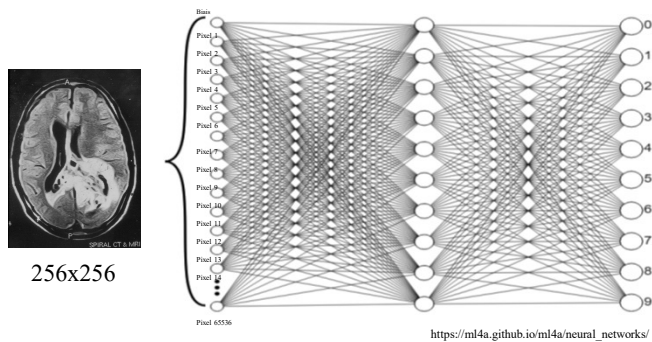


Image médicale (IRM de cerveau)

9

Beaucoup **TROP** de paramètres  
(160M dans la couche 1)

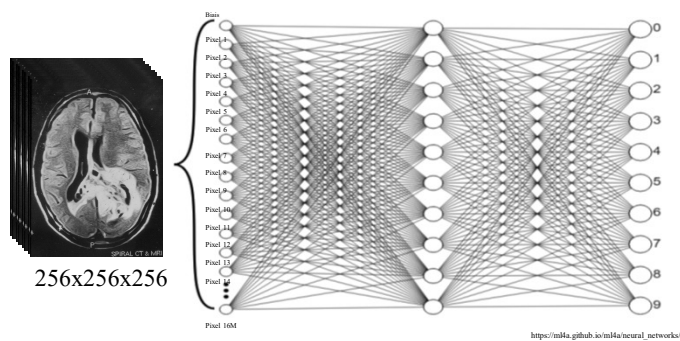


Image médicale 3D (IRM de cerveau)

10

## Comment réduire le nombre de connections?

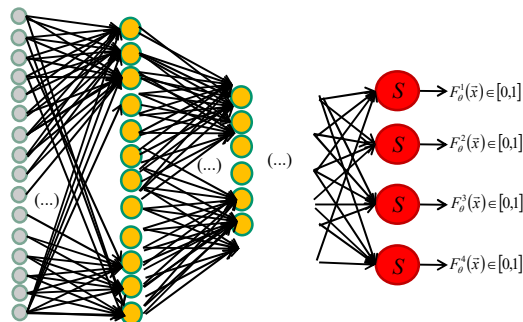


11

11

## Comment réduire le nombre de connections?

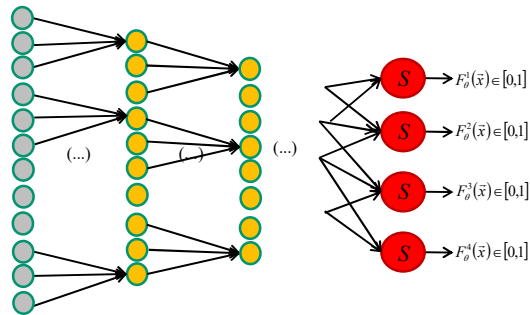
Les **couches pleinement connectées (fully-connected layers)** sont problématiques lorsque le **nombre de neurones est élevé**.



**150-D en entrée** avec **150 neurones** dans la 1ère couche => **22,200 paramètres** dans la couche d'entrée!!

12

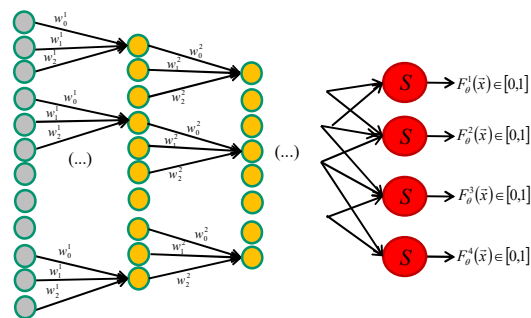
## Solution : connexions partielles



**150-D en entrée avec 148 neurones dans la 1ère couche => 444 paramètres dans la première couche!!**

13

## Paramètres partagés : les neurones de la couche 1 partagent les mêmes poids



Convolution

150-D en entrée avec 148 neurones dans la 1ère couche => **3 paramètres dans la couche d'entrée!!**

**Faible nombre de paramètres = on peut augmenter la profondeur!**

14

# Convolution et couche convolutionnelle **1D**

15

**Exemple 1D de la convolution**

$$(f * W)(v) = \sum_{u=-\infty}^{\infty} f(u)W(v-u)$$

(signal d'entrée)  
 $f(u)$   
10 20 -30 40 -50

(filtre)  
 $W(u)$   
.1 .2 .3

(filtre)  
 $W(-u)$   
.3 .2 .1

$(f*W)(1)$

10	20	-30	40	-50
x	x	x		
.3	.2	.1		

3+4+-3  
↓  
4      

$(f*W)(2)$

10	20	-30	40	-50
x	x	x		
.3	.2	.1		

6-6+4  
↓  
4    4   

$(f*W)(3)$

10	20	-30	40	-50
x	x	x		
.3	.2	.1		

-9+8+-5  
↓  
4    4    -6

16

16



# En gros

**convolution** = **produit scalaire** + **translation**

17

La convolution des réseaux de neurones = **corrélation**

$$(f * W)(v) = \sum_{u=-\infty}^{\infty} f(u)W(v+u)$$



(signal d'entrée)

$f(u)$

10 20 -30 40 -50

(filtre)

$W(u)$

.1 .2 .3

(filtre)

$W(+u)$

.1 .2 .3

$(f * W)(1)$

10 20 -30 40 -50

x x x

.1 .2 .3

1+4-9

-4

$(f * W)(2)$

10 20 -30 40 -50

x x x

.1 .2 .3

2-6+12

-4

8

$(f * W)(3)$

10 20 -30 40 -50

x x x

.1 .2 .3

-3+8-15

-4

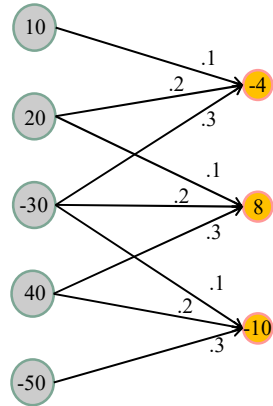
8

-10

18

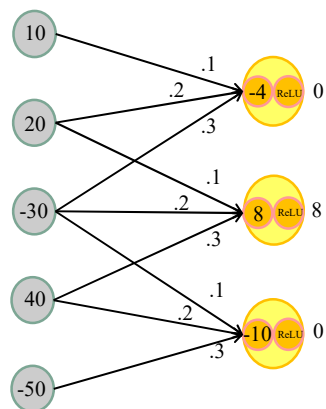
18

L'opération de la page précédente est équivalente à



19

L'opération de la page précédente est équivalente à

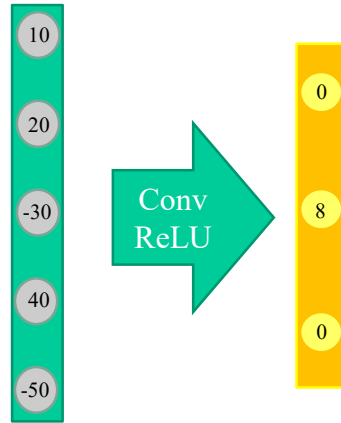


Fonction  
d'activation  
(ex. ReLU)

20

20

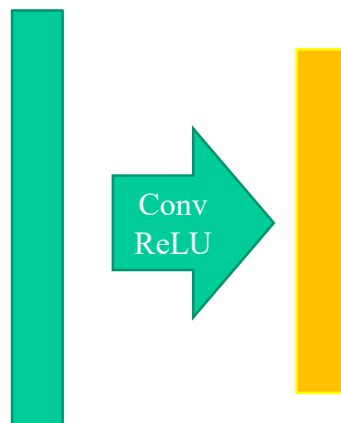
### Représentation graphique courante (simple)



21

21

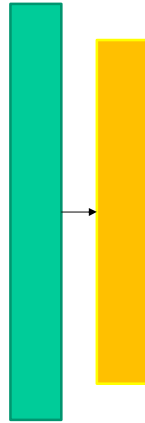
### Représentation graphique courante (encore plus simple)



22

22

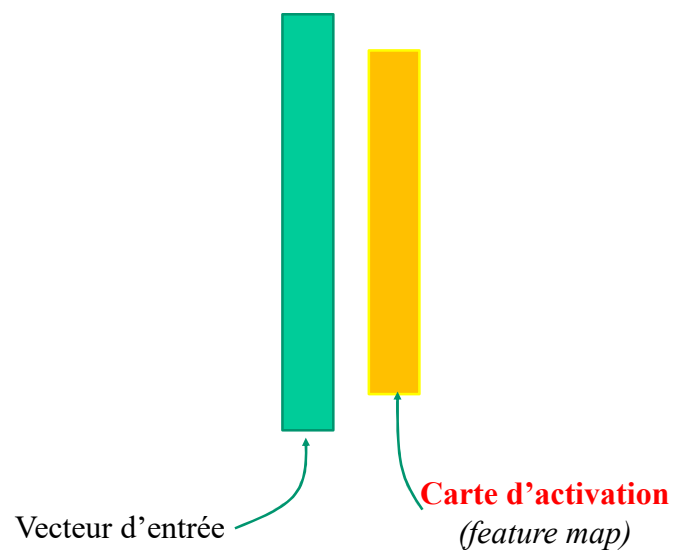
### Représentation graphique courante (vraiment ultra simple)



23

23

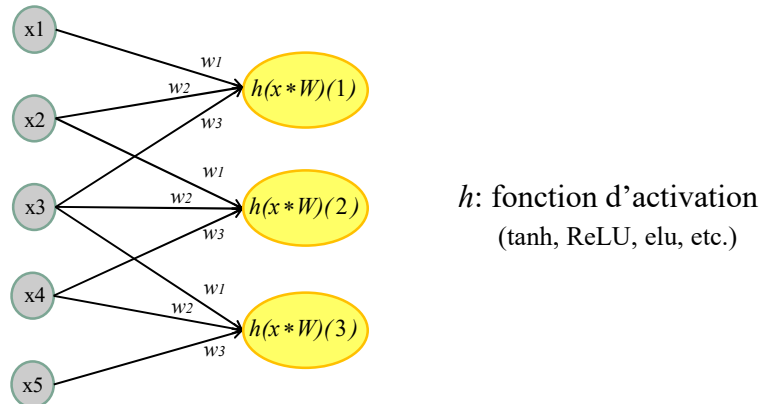
### Représentation graphique courante (eehhh...)



24

24

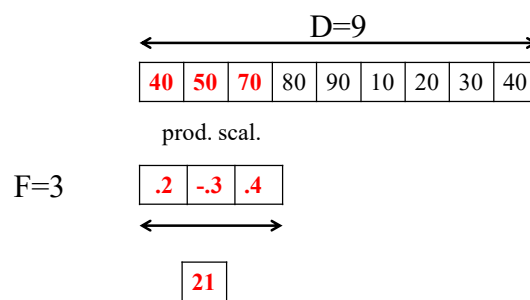
Apprentissage = apprendre les **poids**  $w_i$  **des filtres convolutifs**



25

25

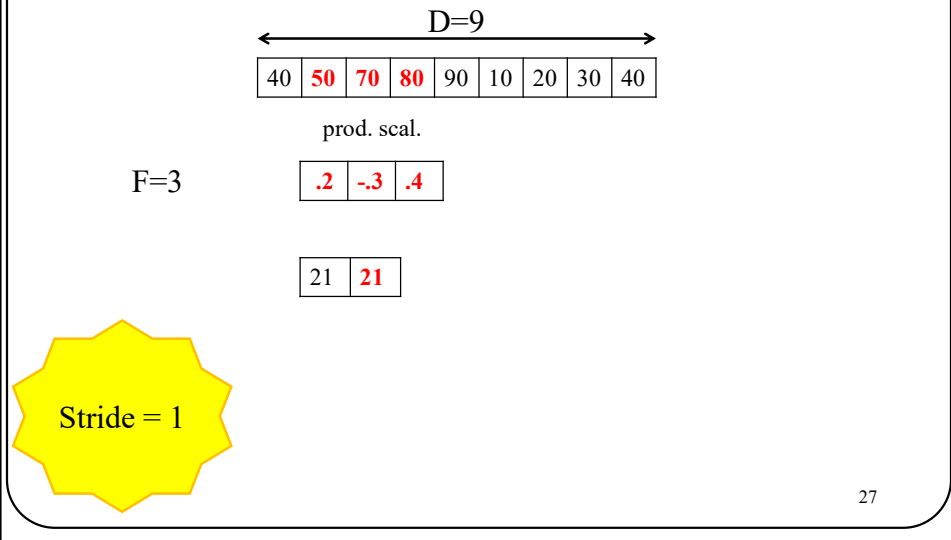
*Stride* et calcul de la taille de la carte d'activation



26

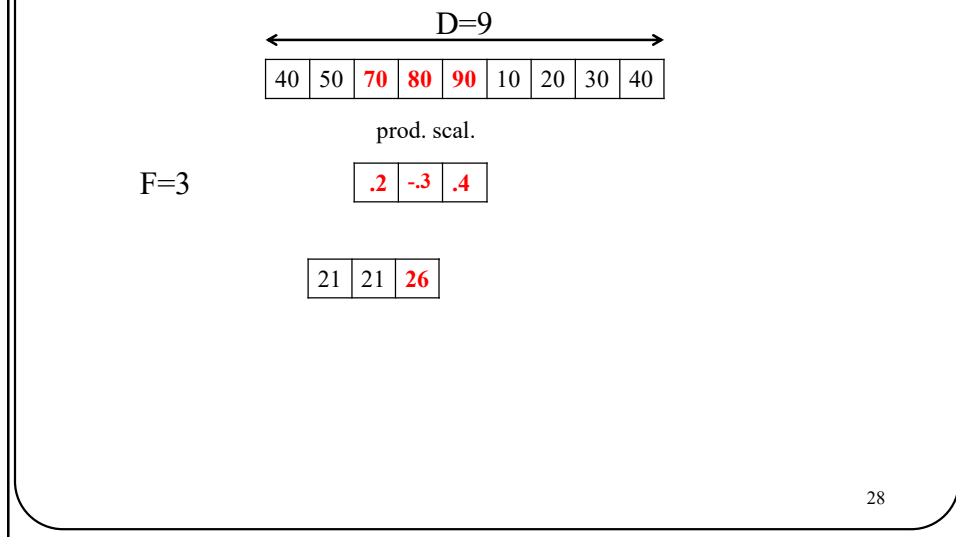
26

## *Stride* et calcul de la taille de la carte d'activation



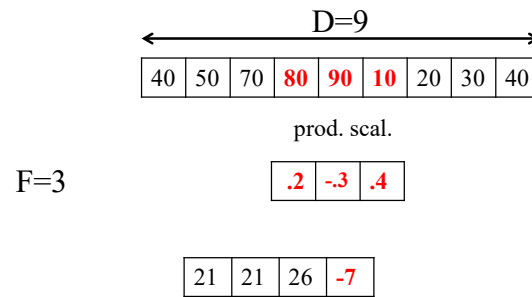
27

## *Stride* et calcul de la taille de la carte d'activation



28

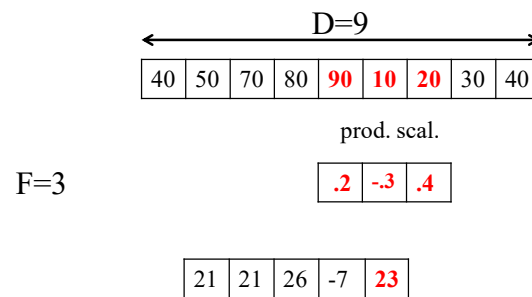
## *Stride* et calcul de la taille de la carte d'activation



29

29

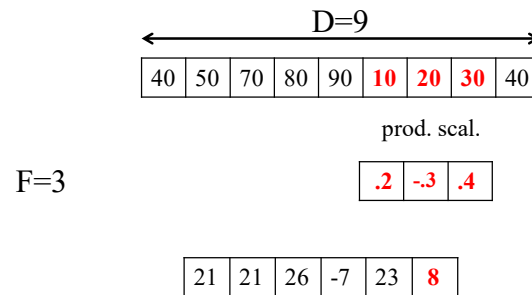
## *Stride* et calcul de la taille de la carte d'activation



30

30

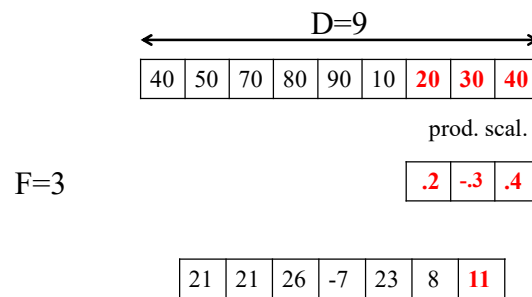
## *Stride* et calcul de la taille de la carte d'activation



31

31

## *Stride* et calcul de la taille de la carte d'activation



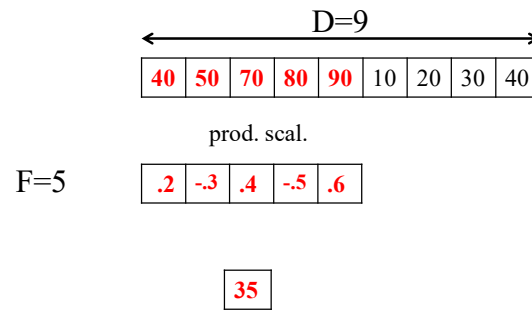
Taille de la carte d'activation = **7**

32

32



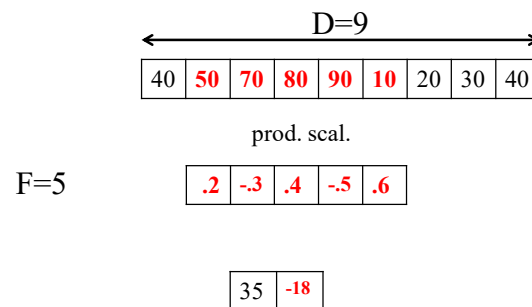
## *Stride* et calcul de la taille de la carte d'activation



33

33

## *Stride* et calcul de la taille de la carte d'activation

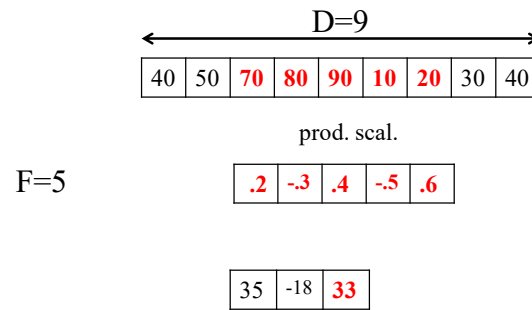


Stride = 1

34

34

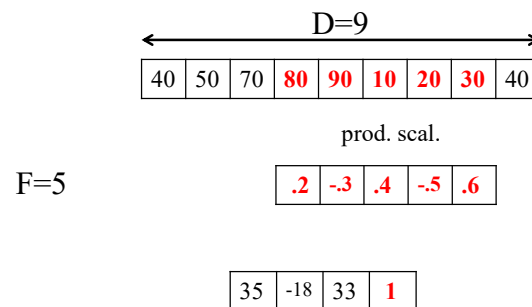
## *Stride* et calcul de la taille de la carte d'activation



35

35

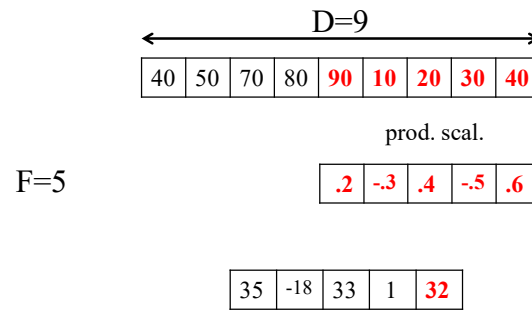
## *Stride* et calcul de la taille de la carte d'activation



36

36

## *Stride* et calcul de la taille de la carte d'activation

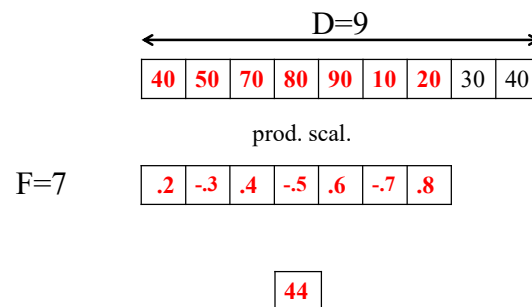


Taille de la carte d'activation = **5**

37

37

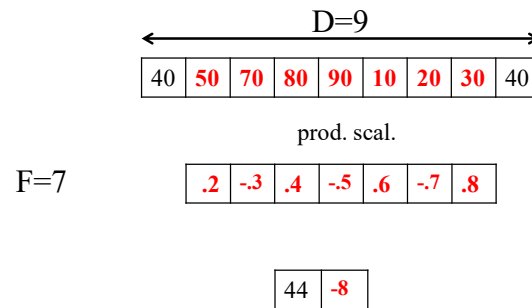
## *Stride* et calcul de la taille de la carte d'activation



38

38

## *Stride* et calcul de la taille de la carte d'activation

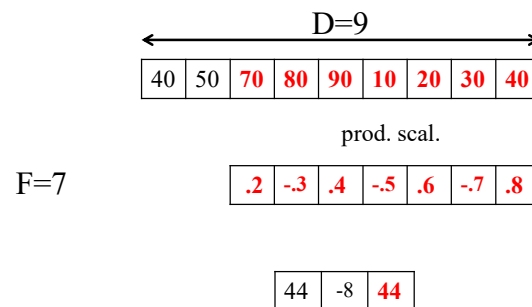


Stride = 1

39

39

## *Stride* et calcul de la taille de la carte d'activation

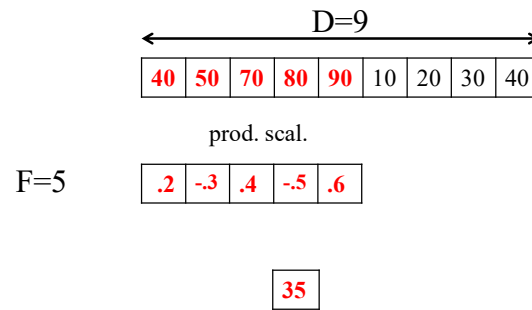


Taille de la carte d'activation = **3**

40

40

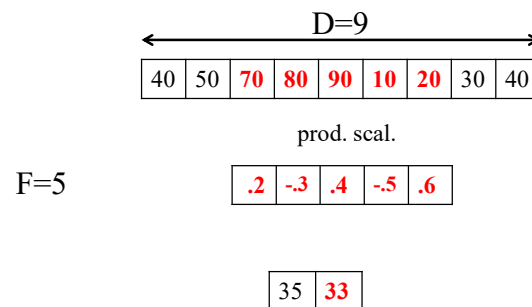
## *Stride* et calcul de la taille de la carte d'activation



41

41

## *Stride* et calcul de la taille de la carte d'activation

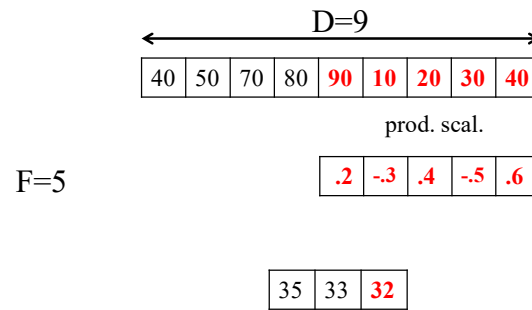


Stride = 2

42

42

## *Stride* et calcul de la taille de la carte d'activation

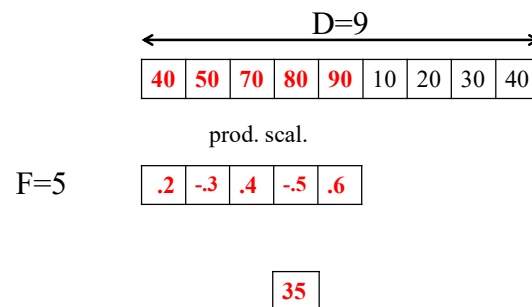


Taille de la carte d'activation = **3**

43

43

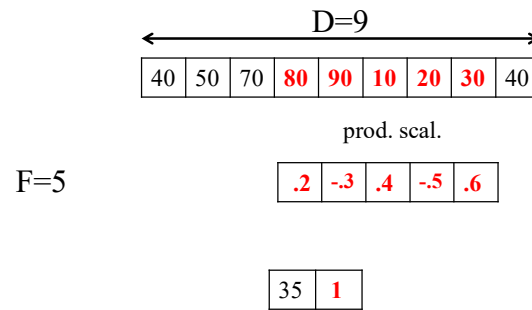
## *Stride* et calcul de la taille de la carte d'activation



44

44

## *Stride* et calcul de la taille de la carte d'activation

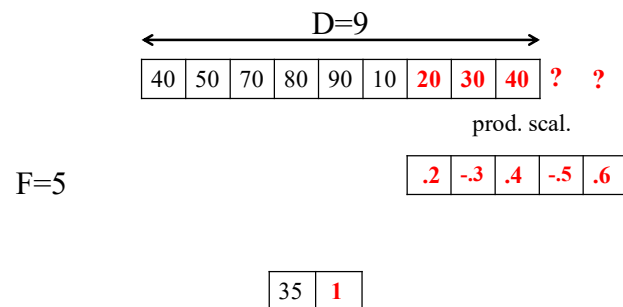


Stride = 3

45

45

## *Stride* et calcul de la taille de la carte d'activation



**ERREUR! Combinaison D-F-S invalide**

46

46

## Stride et calcul de la taille de la carte d'activation

$$\text{Taille de la carte d'activation} = (\mathbf{D-F})/\mathbf{S+1}$$

Doit être un entier

47

47

Parfois on souhaite que le **nombre de neurones** dans la carte d'activation soit **le même** que la couche précédente

$$\begin{array}{cccccc} ? & 10 & 20 & 30 & 40 & 50 \\ \times & \times & \times & & & \\ \hline .1 & .2 & .3 & & & \end{array}$$

Comment gérer les bords?

**Option 1 :** Ajout de zéros (« *zero padding* » remplacer ? par 0)

$$\begin{array}{cccccc} f(u) & & (f*W)(u) \\ 0 & 10 & 20 & 30 & 40 & 50 & 0 \\ & & & & 8 & -4 & 8 & 10 & -6 \end{array}$$

**Option 2 :** Réflexion (« *reflexion padding* »)

$$\begin{array}{cccccc} f(u) & & (f*W)(u) \\ 20 & 10 & 20 & 30 & 40 & 50 & 40 \\ & & & & 10 & -4 & 8 & 10 & 2 \end{array}$$

**Option 3 :** Étirement (« *stretching padding* »)

$$\begin{array}{cccccc} f(u) & & (f*W)(u) \\ 10 & 10 & 20 & 30 & 40 & 50 & 50 \\ & & & & 9 & -4 & 8 & 10 & 21 \end{array}$$

48

48



Parfois on souhaite que le **nombre de neurones** dans la carte d'activation soit **le même** que la couche précédente

Comment gérer les bords?

$$\begin{array}{c} ? \begin{bmatrix} 10 & 20 & 30 & 40 & 50 \end{bmatrix} \\ \times \quad \times \quad \times \\ \begin{bmatrix} 1 & 2 & 3 \end{bmatrix} \end{array}$$

**Option 1 :** Ajout de zéros (« *zero padding* » remplacer ? par 0)

$$f(u) = \begin{bmatrix} 0 & 10 & 20 & 30 & 40 & 50 & 0 \end{bmatrix}$$

$$(f^*W)(u) = \begin{bmatrix} 8 & -4 & 8 & 10 & -6 \end{bmatrix}$$

**Option 2 :** Réflexion (« *reflexion padding* »)

De loin l'option la plus utilisée

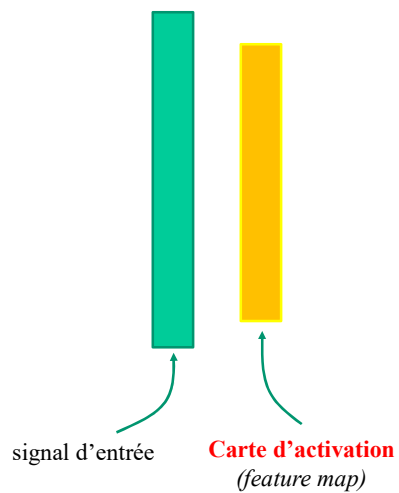
$$f(u) = \begin{bmatrix} 10 & 10 & 20 & 30 & 40 & 50 & 50 \end{bmatrix}$$

$$(f^*W)(u) = \begin{bmatrix} 9 & -4 & 8 & 10 & -21 \end{bmatrix}$$

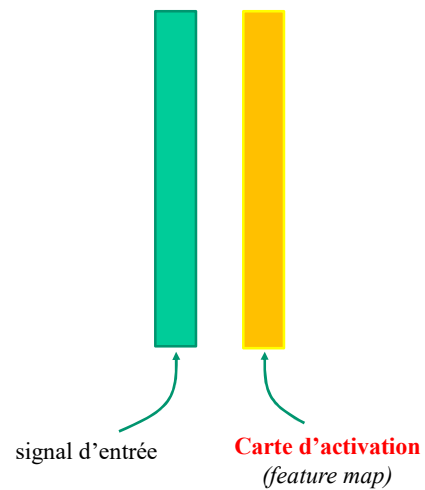
49

49

**Couche convolutionnelle  
sans « padding »**

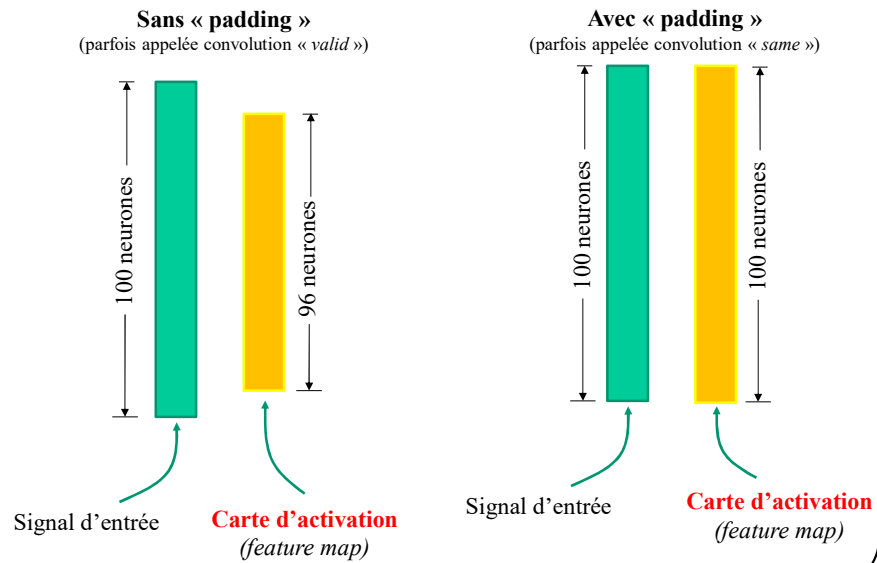


**Couche convolutionnelle  
avec « padding »**



50

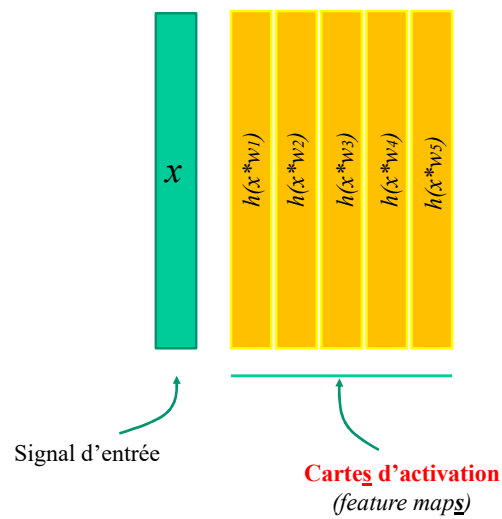
Exemple : taille de filtre = 5, stride=1



51

Il est possible d'apprendre **plusieurs filtres par couche**

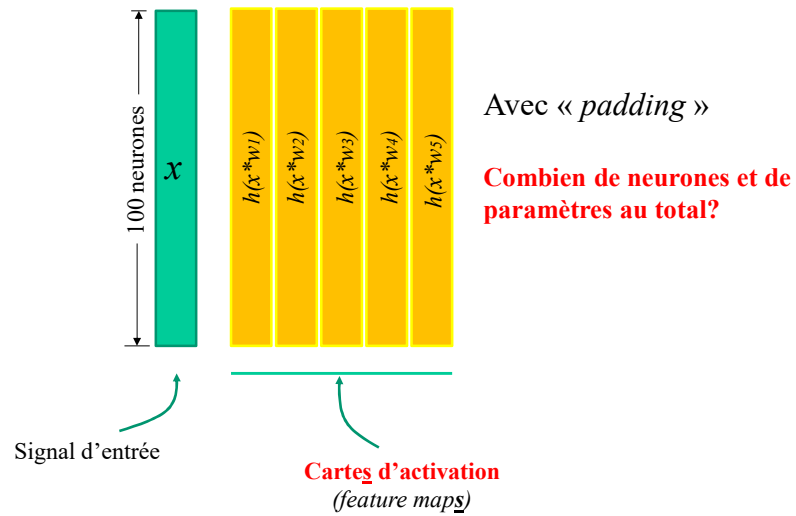
(ex. 5 filtres donnant 5 cartes d'activation)



52

Taille de filtre = 5

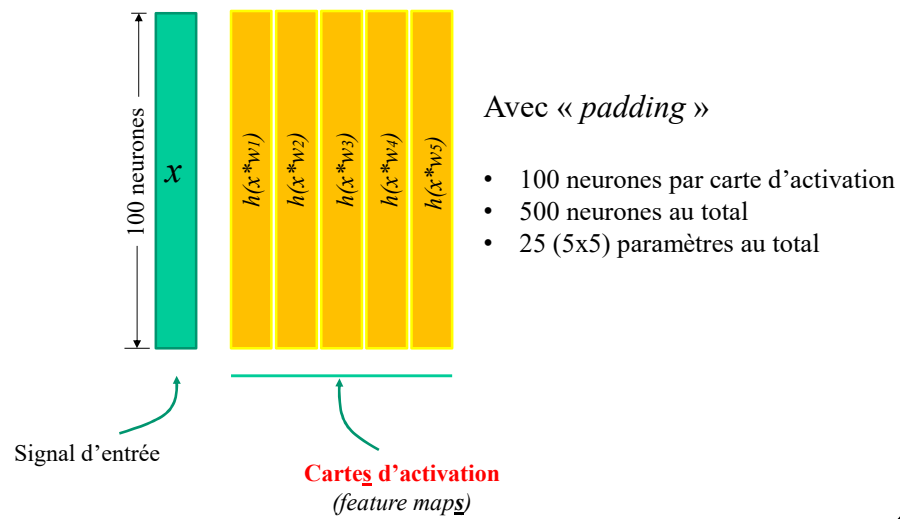
(ex. 5 filtres donnant 5 cartes d'activation)



53

Taille de filtre = 5

(ex. 5 filtres donnant 5 cartes d'activation)



54

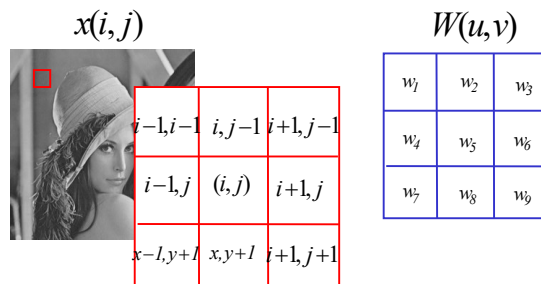
# Convolution et couche convolutionnelle **2D**

55

## Filtage 2D

(sans flip de filtre)

$$(x * W)(i, j) = \sum_u \sum_v f(i + u, j + v) W(u, v)$$

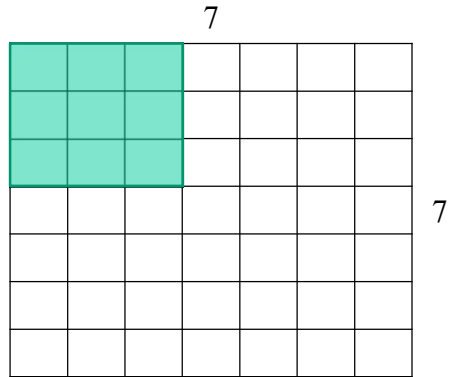


$$\begin{aligned} (x * W)(i, j) = & w_1 x(i-1, j-1) + w_2 x(i, j-1) + w_3 x(i+1, j-1) \\ & + w_4 x(i-1, j) + w_5 x(i, j) + w_6 x(i+1, j) \\ & + w_7 x(i-1, j+1) + w_8 x(i, j+1) + w_9 x(i+1, j+1) \end{aligned}$$

56

## Convolution 2D

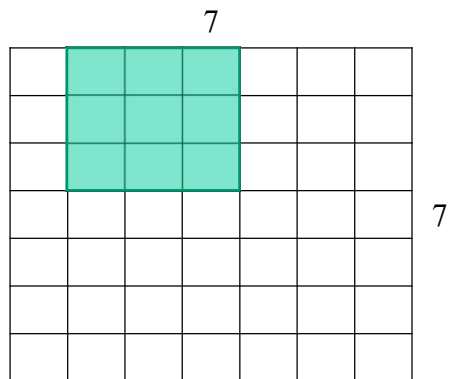
Filtre = 3x3  
Stride = 1



57

## Convolution 2D

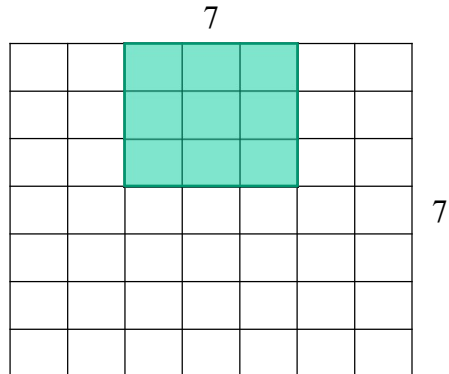
Filtre = 3x3  
Stride = 1



58

## Convolution 2D

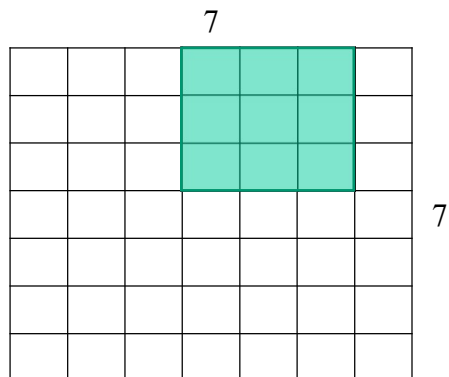
Filtre = 3x3  
Stride = 1



59

## Convolution 2D

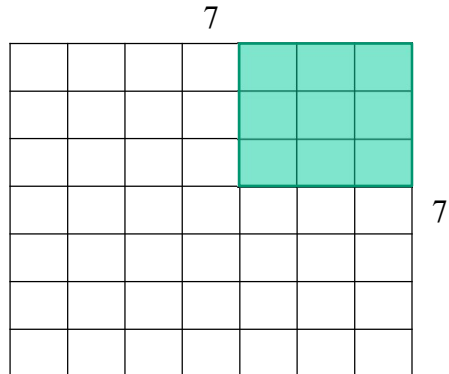
Filtre = 3x3  
Stride = 1



60

## Convolution 2D

Filtre = 3x3  
Stride = 1

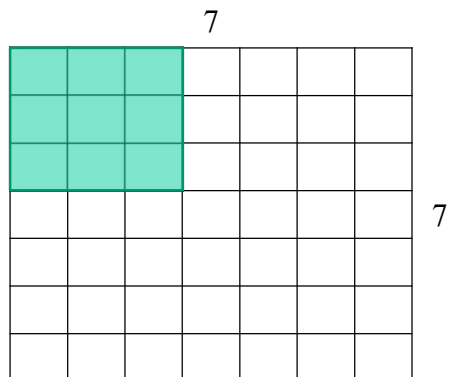


Taille de la carte d'activation (pour stride 1) = **5x5**

61

## Convolution 2D

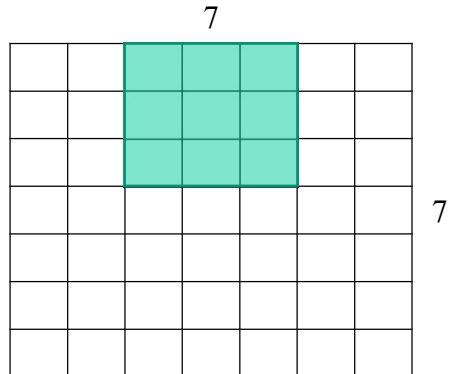
Filtre = 3x3  
Stride = 2



62

## Convolution 2D

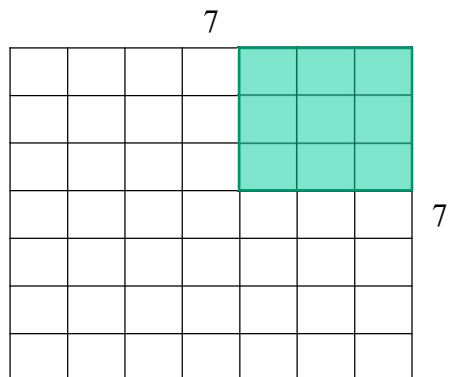
Filtre = 3x3  
Stride = 2



63

## Convolution 2D

Filtre = 3x3  
Stride = 2



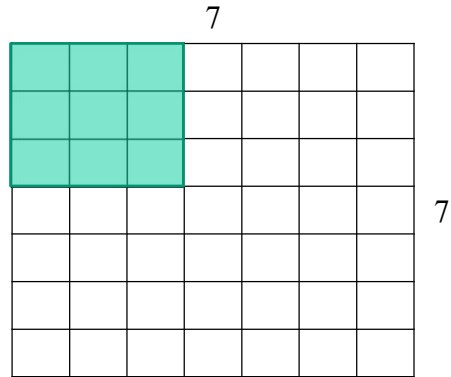
Taille de la carte d'activation (pour stride 2) = **3x3**

64



## Convolution 2D

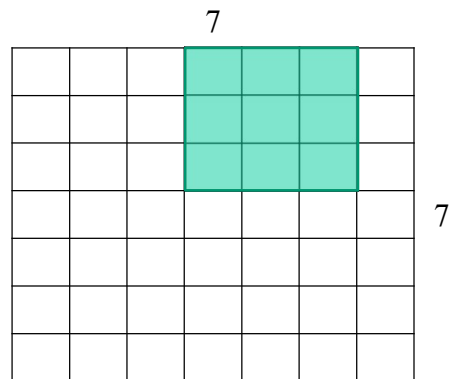
Filtre = 3x3  
Stride = 3



65

## Convolution 2D

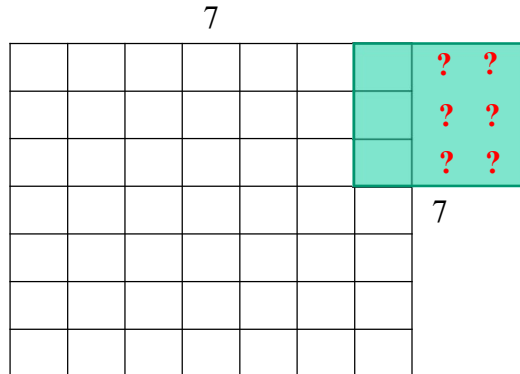
Filtre = 3x3  
Stride = 3



66

## Convolution 2D

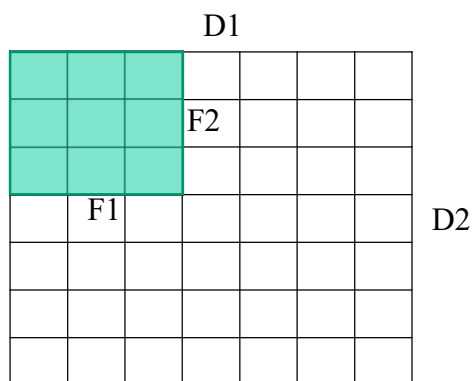
Filtre = 3x3  
Stride = 2



**Combinaison D-F-S invalide!**

67

## Convolution 2D

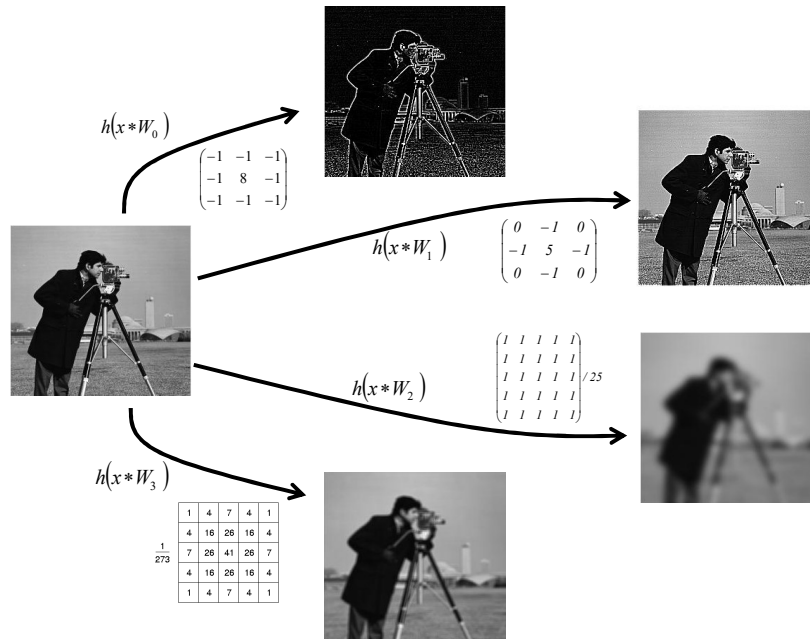


Taille de la carte d'activation :

$$(D1-F1)/S+1 \times (D2-F2)/S+1$$

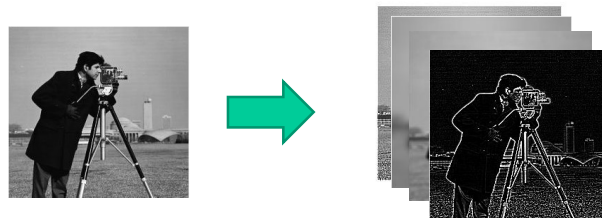
68

## Différents filtres = différentes cartes d'activation



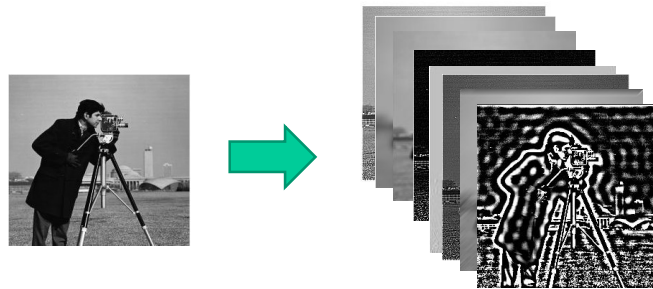
69

## 4 filtres = Couche convolutive avec 4 cartes d'activation



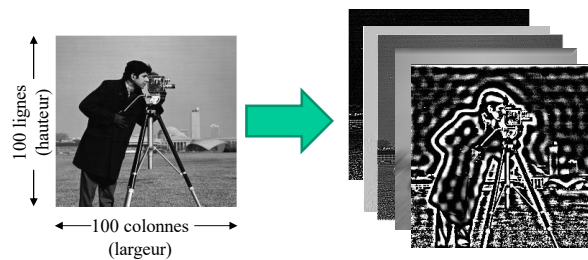
70

**K filtres = Couche convolutive avec K cartes d'activation**



71

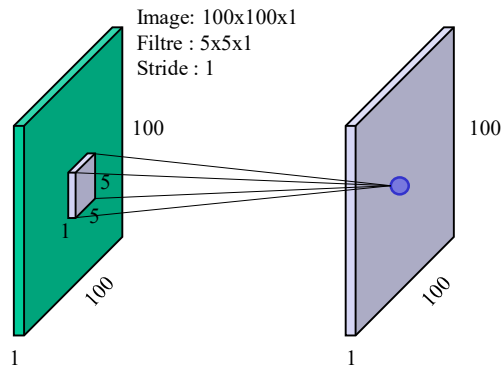
Ex.: taille de filtre : 5x5, 5 cartes d'activation, convolution « same »



- 10,000 neurones par carte d'activation
- 50,000 neurones au total
- $5 \times 5 \times 5 = 125$  paramètres au total

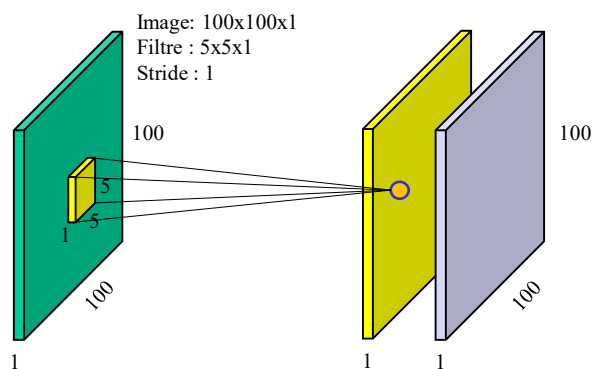
72

Représentation schématique  
(1 filtre et 1 carte d'activation, convolution « same »)



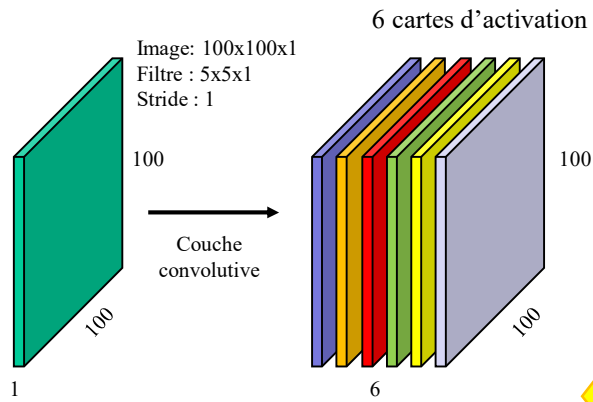
73

Représentation schématique  
(2 filtres et 2 cartes d'activation, convolution « same »)



74

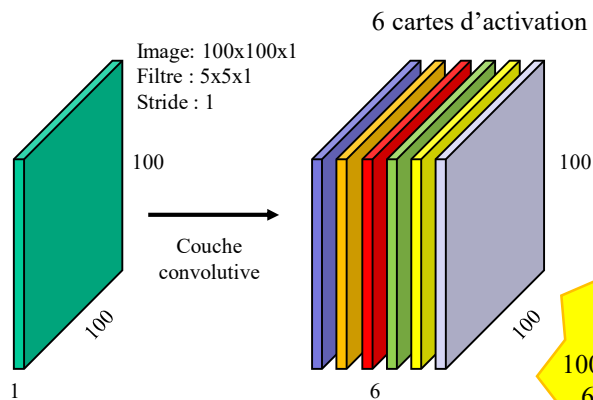
Représentation schématique  
(6 filtres et 6 cartes d'activation, convolution « same »)



Combien de neurones  
et de paramètres  
au total?

75

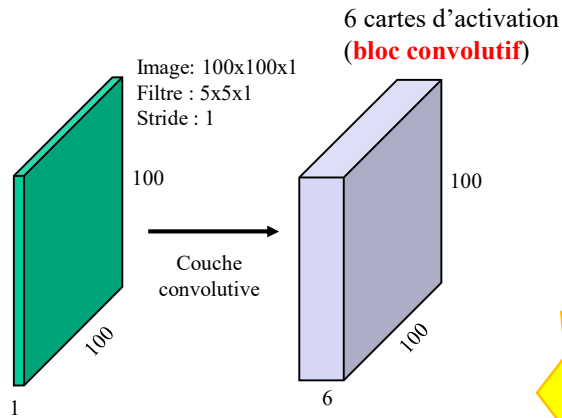
Représentation schématique  
(6 filtres et 6 cartes d'activation, convolution « same »)



$100 \times 100 \times 6 = 60,000$  neurones  
 $6 \times 5 \times 5 \times 1 = 150$  paramètres

76

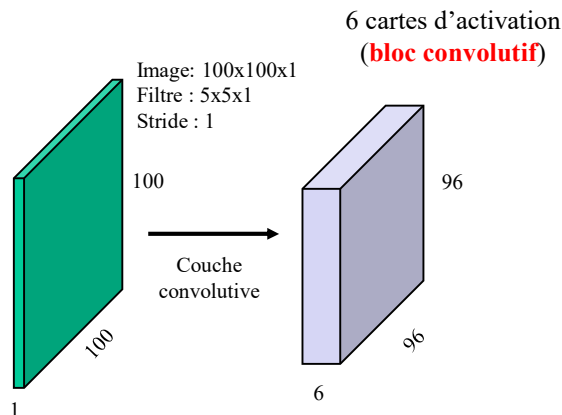
Représentation schématique simplifiée  
(6 filtres et 6 cartes d'activation, convolution « *same* »)



$100 \times 100 \times 6 = 60,000$  neurones  
 $6 \times 5 \times 5 \times 1 = 150$  paramètres

77

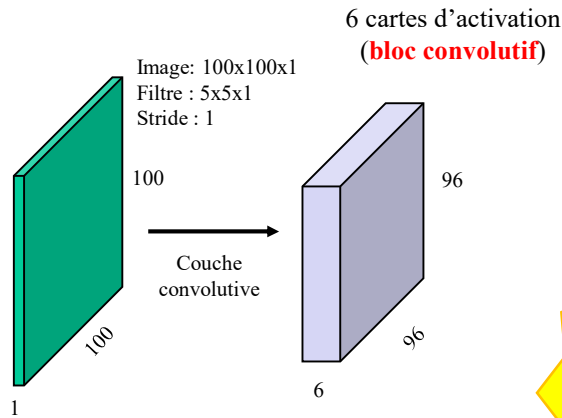
Représentation schématique simplifiée  
(6 filtres et 6 cartes d'activation, convolution « *valid* »)



Combien de neurones  
et de paramètres  
au total?

78

Représentation schématique simplifiée  
(6 filtres et 6 cartes d'activation, convolution « valid »)



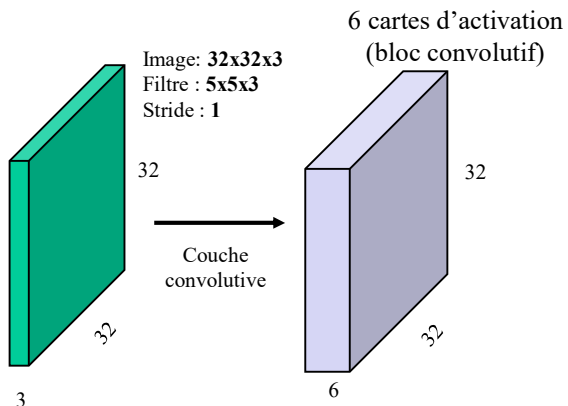
96x96x6=55,296 neurones  
6x5x5x1=150 paramètres

79

Représentation schématique images couleur  
(ex.: images RGB de CIFAR10  
convolution « *same* »)



Exemples cifar10

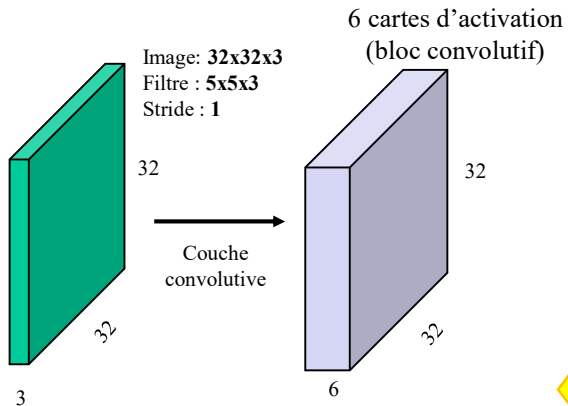


Combien de neurones  
et de paramètres  
au total?

80



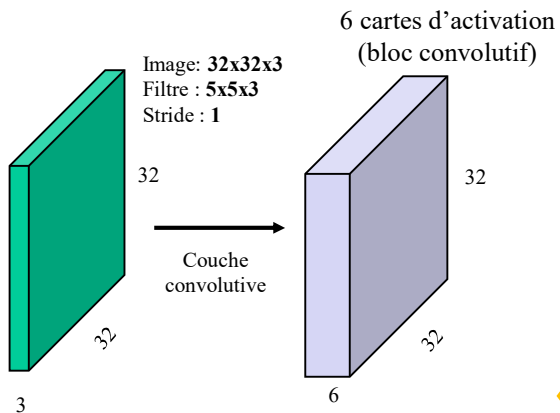
Représentation schématique images couleur  
(ex.: images RGB de CIFAR10  
convolution « *same* »)



$32 \times 32 \times 6 = 6,144$  neurones  
 $6 \times 5 \times 5 \times 3 = \underline{450}$  paramètres

81

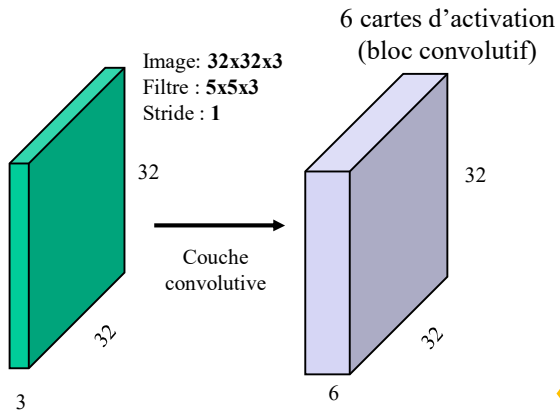
Représentation schématique images couleur  
(ex.: images RGB de CIFAR10  
convolution « *same* »)



Qu'arrivera-t-il si on  
utilise une stride de 3?

82

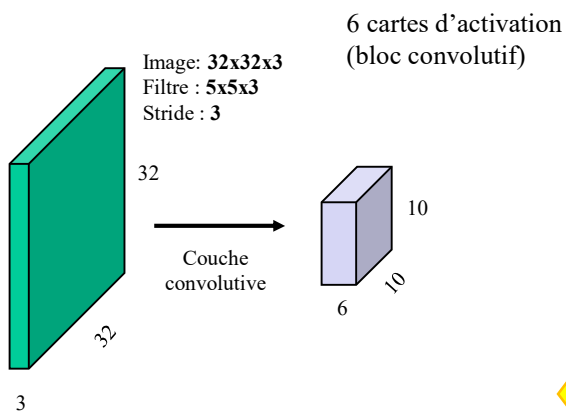
Représentation schématique images couleur  
(ex.: images RGB de CIFAR10  
convolution « same »)



$$(D-F)/S+1 = (32-5)/3+1=10$$

83

Représentation schématique images couleur  
(ex.: images RGB de CIFAR10  
convolution « valid »)



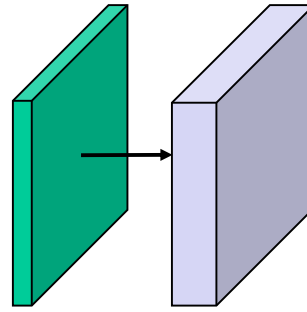
$$10 \times 10 \times 6 = 600 \text{ neurones}$$

$$6 \times 5 \times 5 \times 3 = 450 \text{ paramètres}$$

84

## Exemple

Volume en entrée :  $32 \times 32 \times 3$   
**10 filtres  $5 \times 5$**  avec **stride = 1**  
et convolution « *same* »

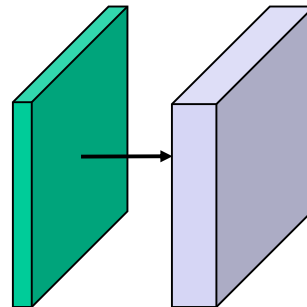


**Combien y a-t-il de paramètres dans cette couche?**

85

## Exemple

Volume en entrée :  $32 \times 32 \times 3$   
**10 filtres  $5 \times 5$**  avec **stride = 1**  
et convolution « *same* »



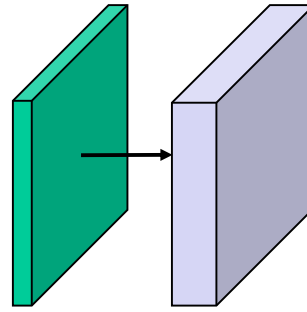
Combien y a-t-il de paramètres dans cette couche?

Chaque filtre a  $5 \times 5 \times 3 = 75$  paramètres  
Comme il y a **10 filtres** : **750** paramètres

86

## Exemple

Volume en entrée :  $32 \times 32 \times 3$   
**10 filtres  $5 \times 5$**  avec **stride = 1**  
et convolution « **same** ».



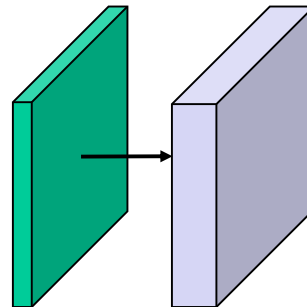
Combien y a-t-il de paramètres dans cette couche **si on ajoute un biais**?

Chaque filtre a  $5 \times 5 \times 3 + 1 = 76$  paramètres (+1 pour le biais)  
Comme il y a **10 filtres** : **760** paramètres

87

## Exemple

Volume en entrée :  $32 \times 32 \times 3$   
**10 filtres  $5 \times 5$**  avec **stride = 1**  
et convolution « **valid** »

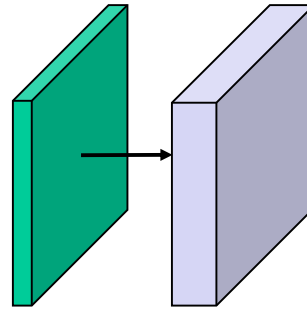


Combien de paramètres dans cette couche?

88

## Exemple

Volume en entrée : **32 x 32 x 3**  
**10 filtres 5x5** avec **stride = 1**  
et convolution « **valid** »



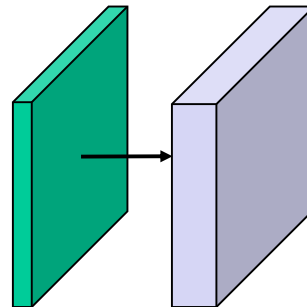
Combien de paramètres dans cette couche?

**Même chose**, cela ne change pas la conformité des filtres

89

## Exemple

Volume en entrée : **32 x 32 x 3**  
**10 filtres 5x5** avec **stride = 1**  
et convolution « **valid** »

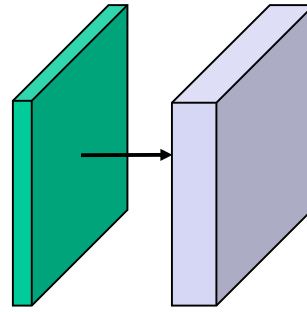


Combien de **neurones** dans les cartes d'activations?

90

## Exemple

Volume en entrée : **32 x 32 x 3**  
**10 filtres 5x5** avec **stride = 1**  
et convolution « **valid** »

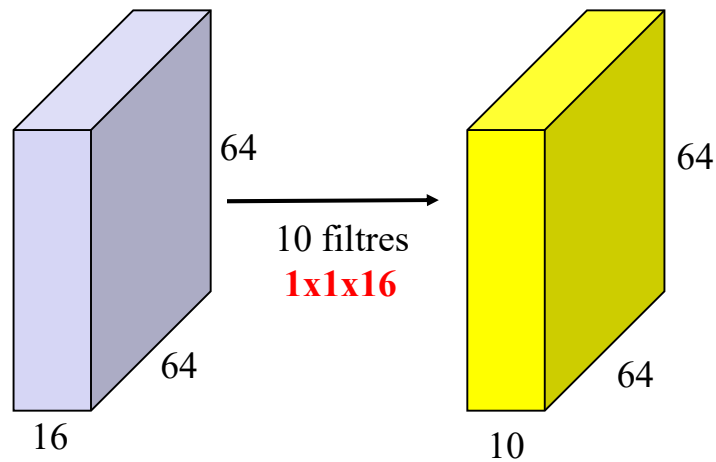


Combien de **neurones** dans les cartes d'activations?

$$(32-5+1) \times (32-5+1) \times 10 = 7,840$$

91

## Des filtres 1x1? Oui ça marche



92

## Exemple simple d'un filtre 1x1



$$\begin{bmatrix} 1 & 1 & 1 \\ \frac{1}{3} & \frac{1}{3} & \frac{1}{3} \end{bmatrix}$$

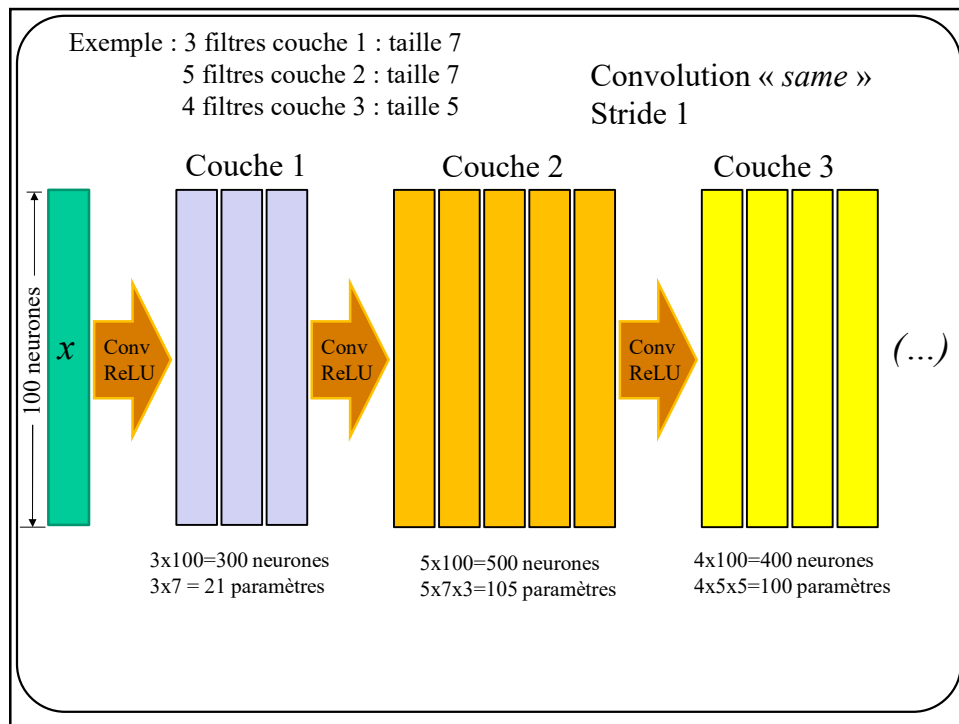


Filtre moyennant les canaux **rouge**, **vert**, **bleu** d'une image couleur.  
Résultat, une image en **niveau de gris**.

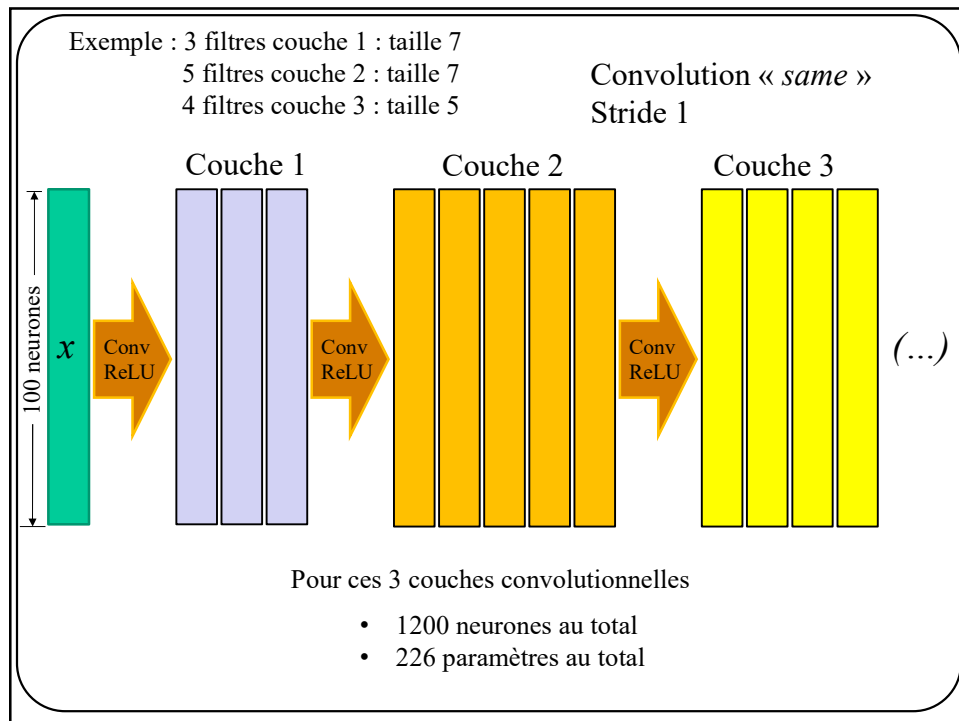
93

Tout comme un Perceptron  
multi-couches, un réseau à  
convolution contient **plusieurs  
couches consécutives**

94

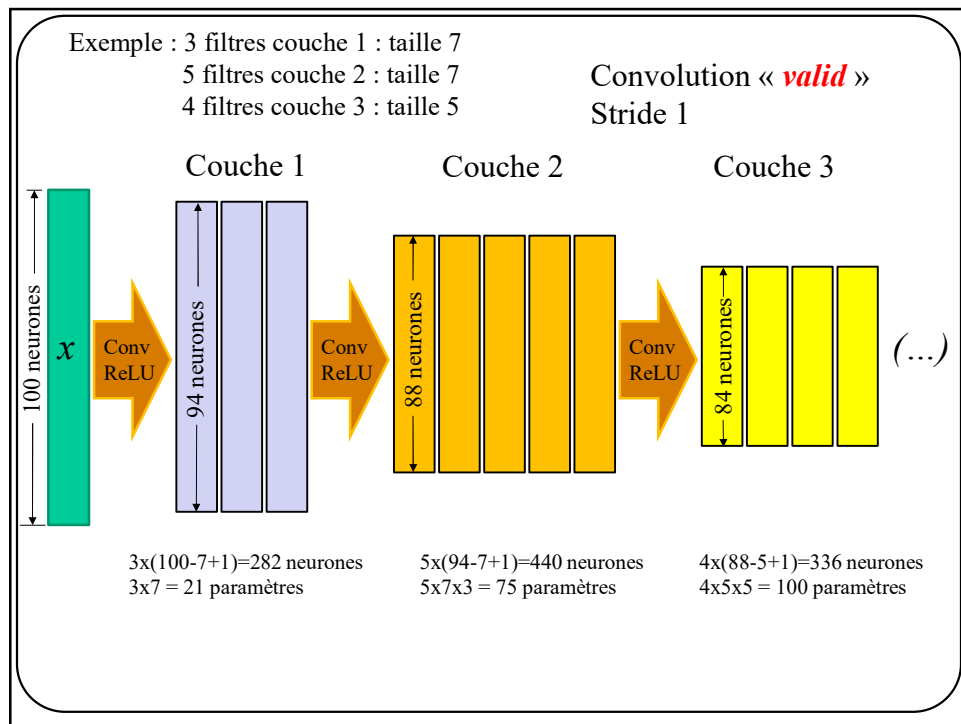


95

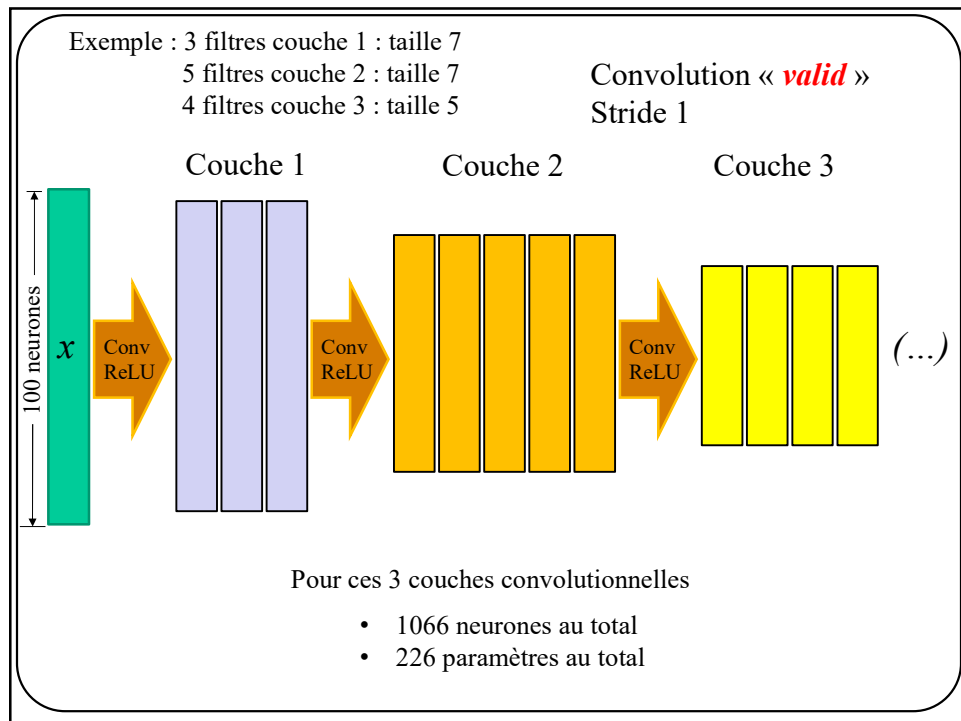


96





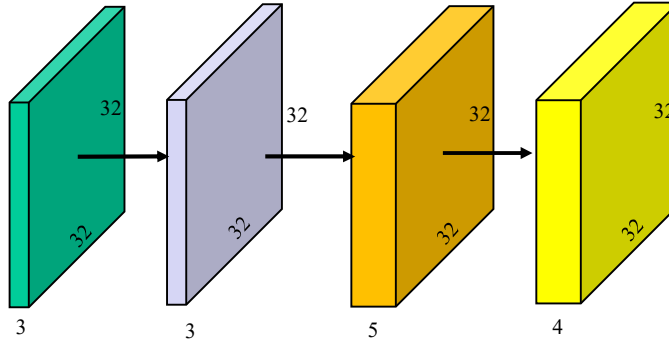
97



98

Image RGB : couche 1 : 3 filtres de taille 7x7  
 couche 2 : 5 filtres de taille 9x9  
 couche 3 : 4 filtres de taille 11x11  
 convolution « *same* »

Image: 32x32x3  
 Stride : 1

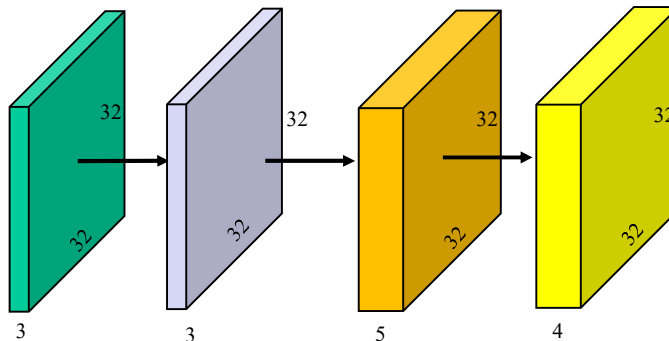


$3 \times (32 \times 32) = 3,072$  neurones     $5 \times (32 \times 32) = 5,120$  neurones     $4 \times (32 \times 32) = 4,096$  neurones  
 $3 \times 7 \times 7 \times 3 = 441$  paramètres     $5 \times 9 \times 9 \times 3 = 1,215$  paramètres     $4 \times 11 \times 11 \times 5 = 2,420$  paramètres

99

Image RGB : couche 1 : 3 filtres de taille 7x7  
 couche 2 : 5 filtres de taille 9x9  
 couche 3 : 4 filtres de taille 11x11  
 convolution « *same* »

Image: 32x32x3  
 Stride : 1

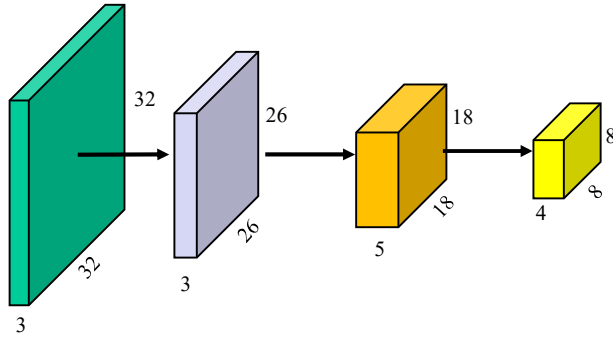


12,288 neurones au total  
 4,076 paramètres au total

100

Image RGB : couche 1 : 3 filtres de taille 7x7  
 couche 2 : 5 filtres de taille 9x9  
 couche 3 : 4 filtres de taille 11x11  
 convolution « valid »

Image: 32x32x3  
 Stride : 1

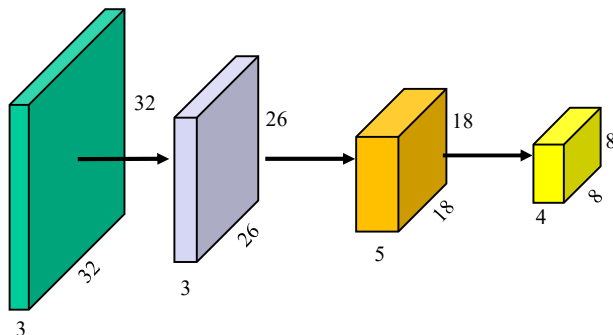


$3 \times (26 \times 26) = 2,028$  neurones     $5 \times (18 \times 18) = 1,620$  neurones     $4 \times (8 \times 8) = 256$  neurones  
 $3 \times 7 \times 7 \times 3 = 441$  paramètres     $5 \times 9 \times 9 \times 3 = 1,215$  paramètres     $4 \times 11 \times 11 \times 5 = 2,420$  paramètres

101

Image RGB : couche 1 : 3 filtres de taille 7x7  
 couche 2 : 5 filtres de taille 9x9  
 couche 3 : 4 filtres de taille 11x11  
 convolution « valid »

Image: 32x32x3  
 Stride : 1

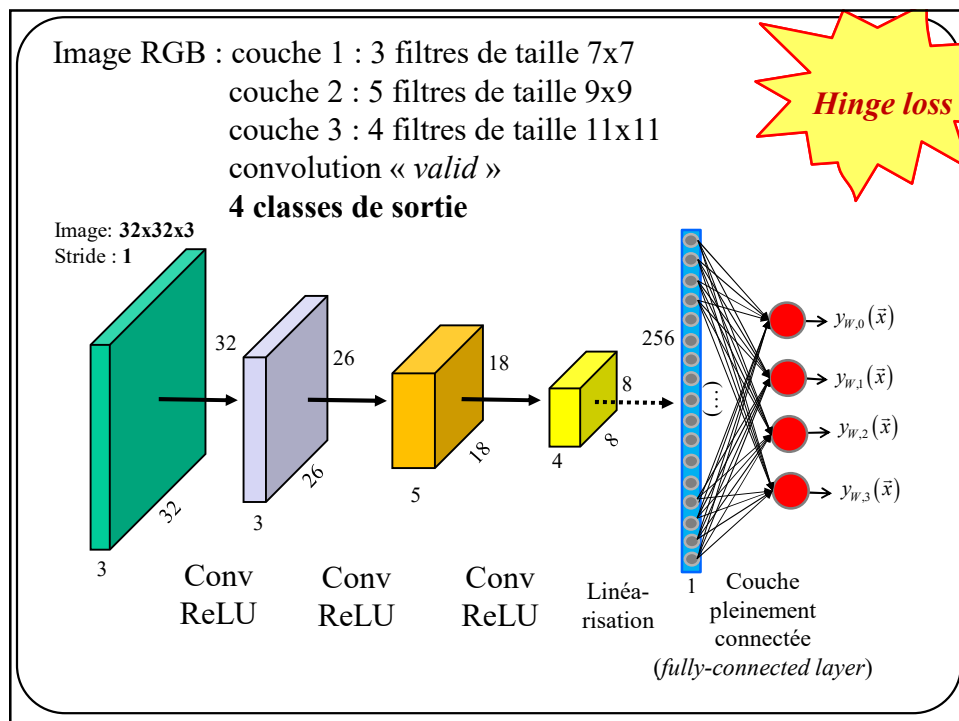


3,904 neurones au total  
 4,076 paramètres au total

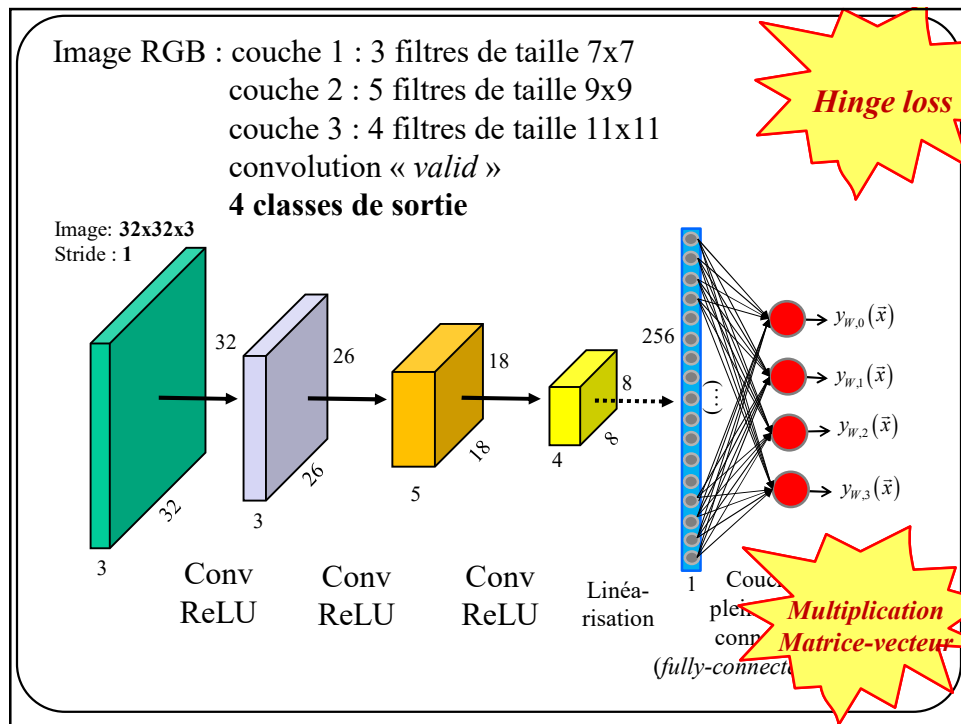
102

Tout comme un perceptron multi-couches, un réseau à convolution se termine par une **couche de sortie** avec **1 neurone par variable prédite**

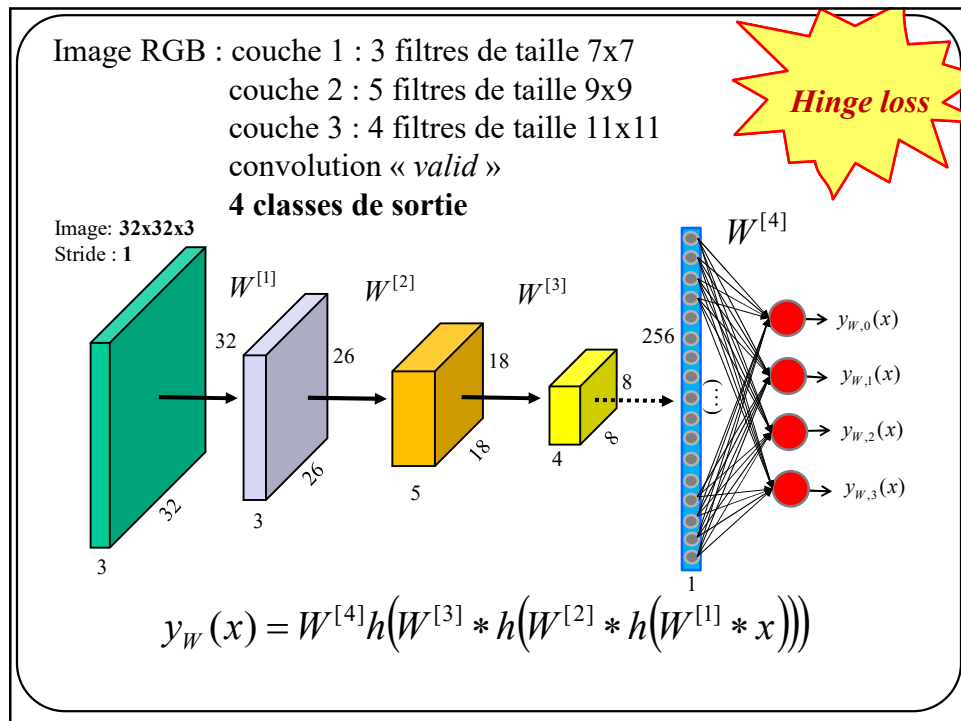
103



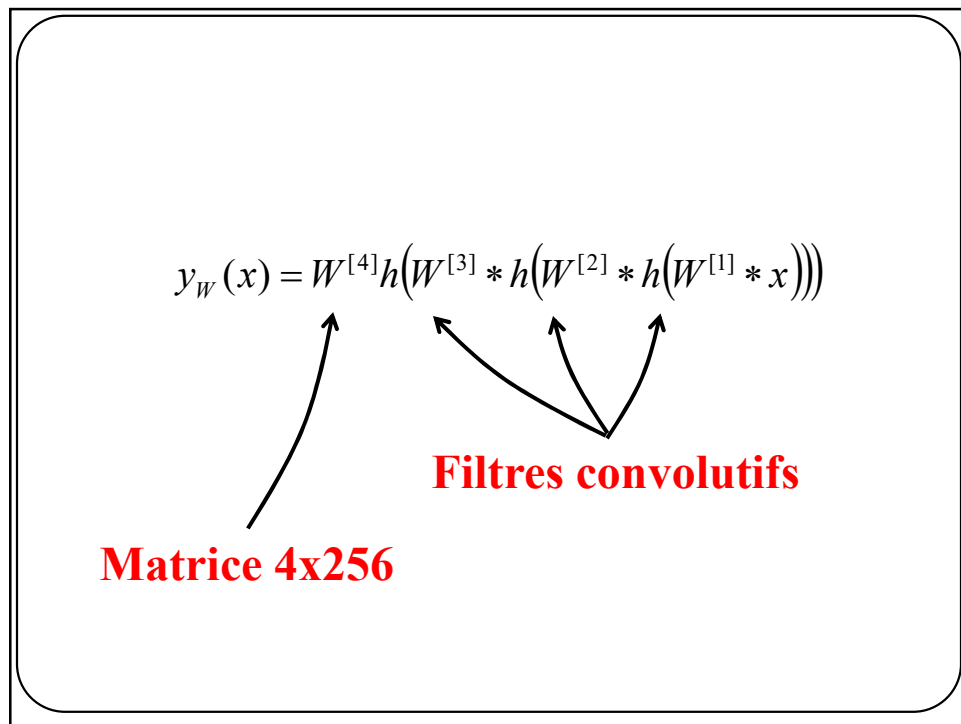
104



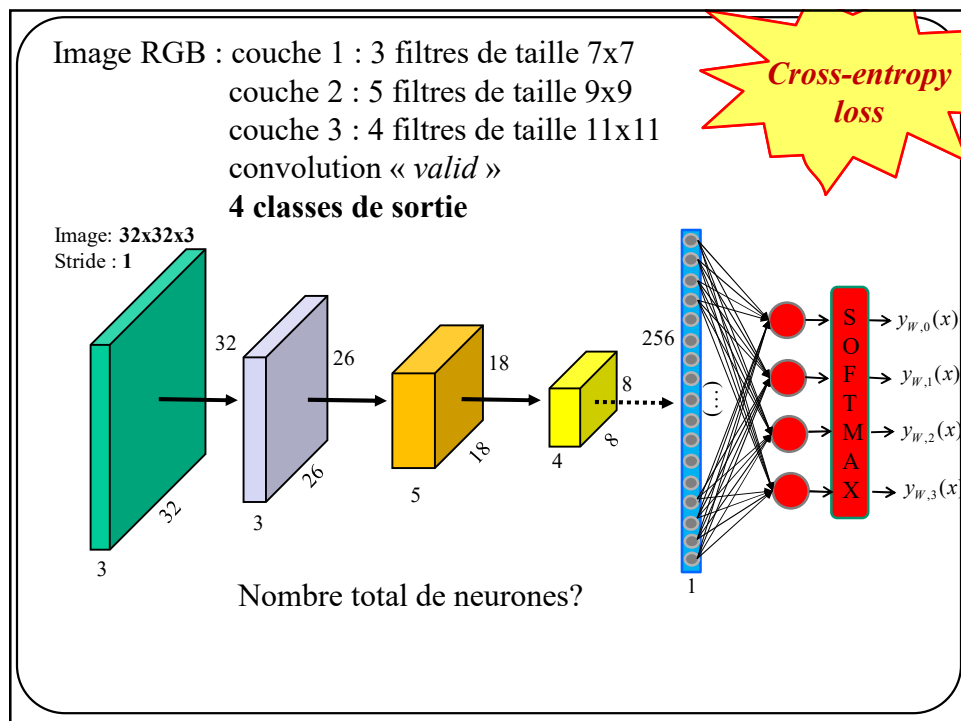
105



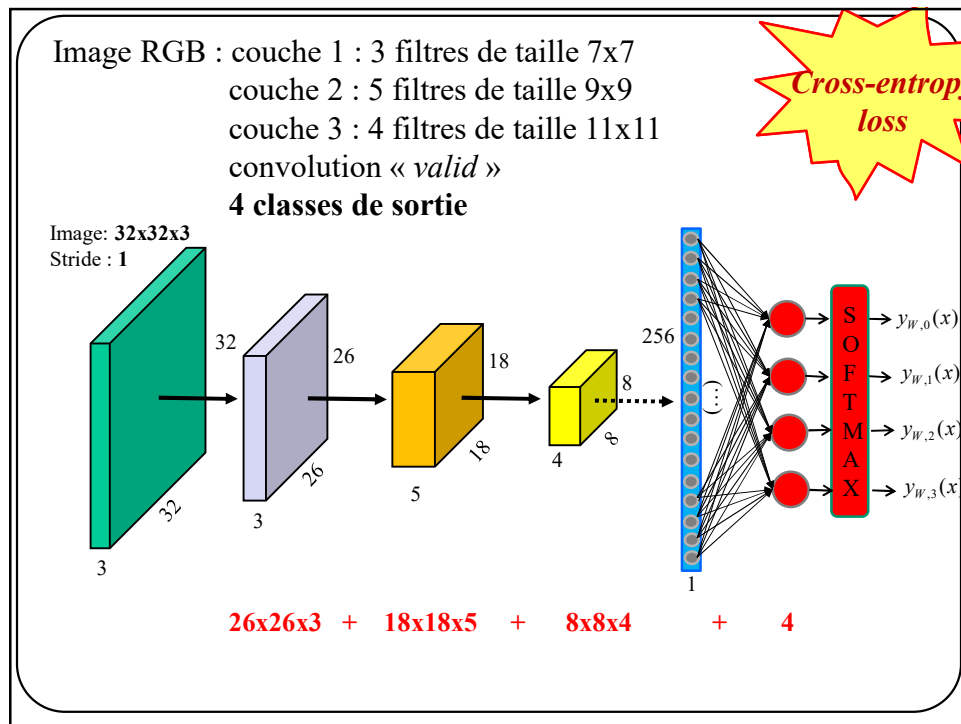
106



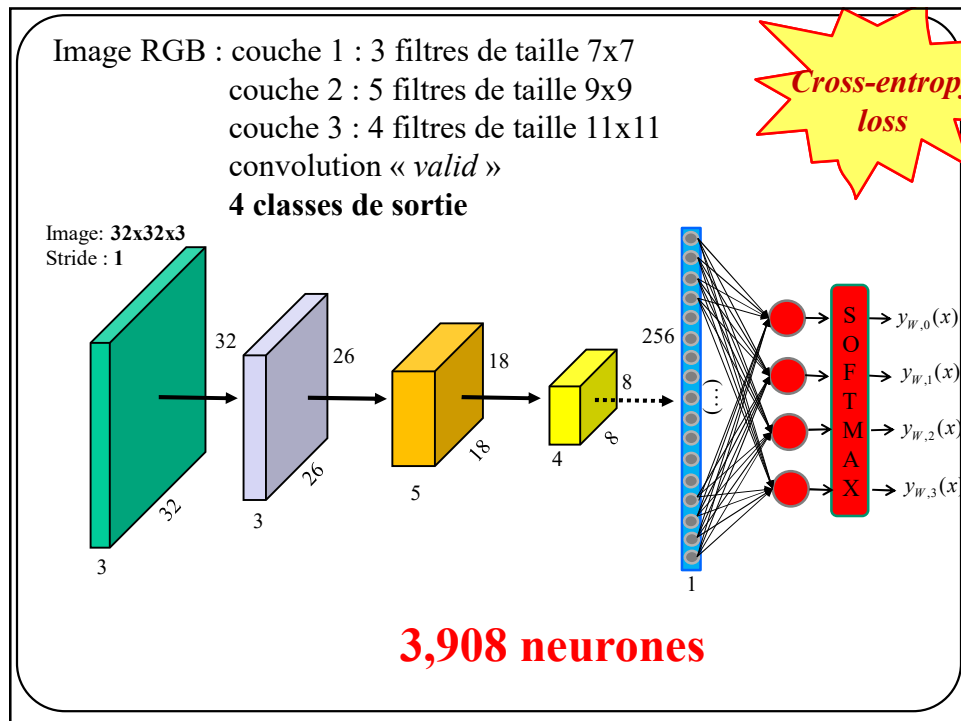
107



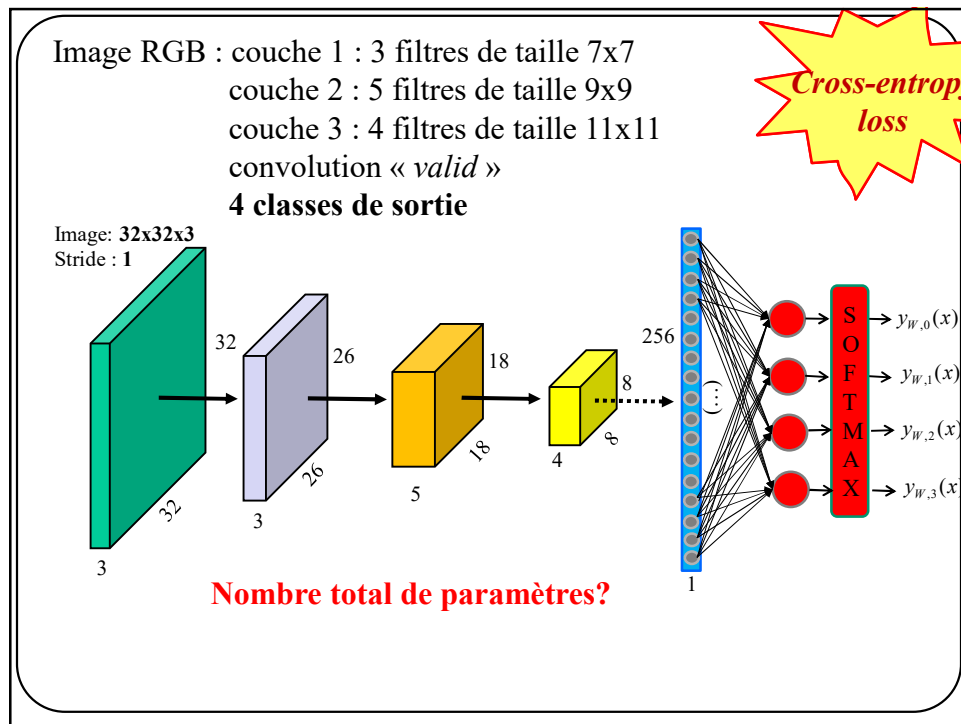
108



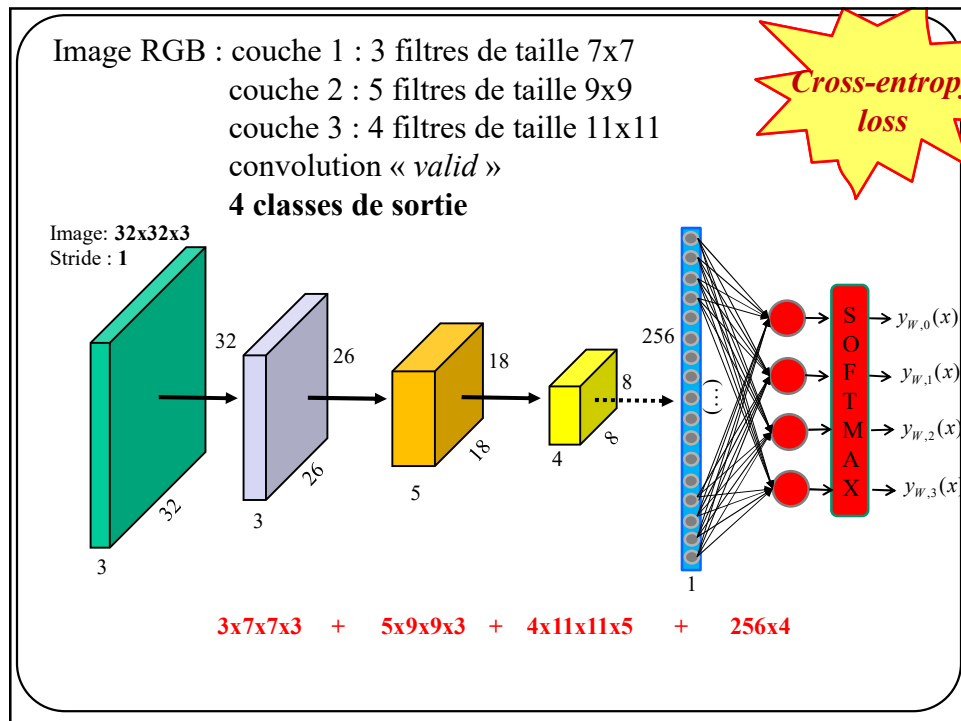
109



110

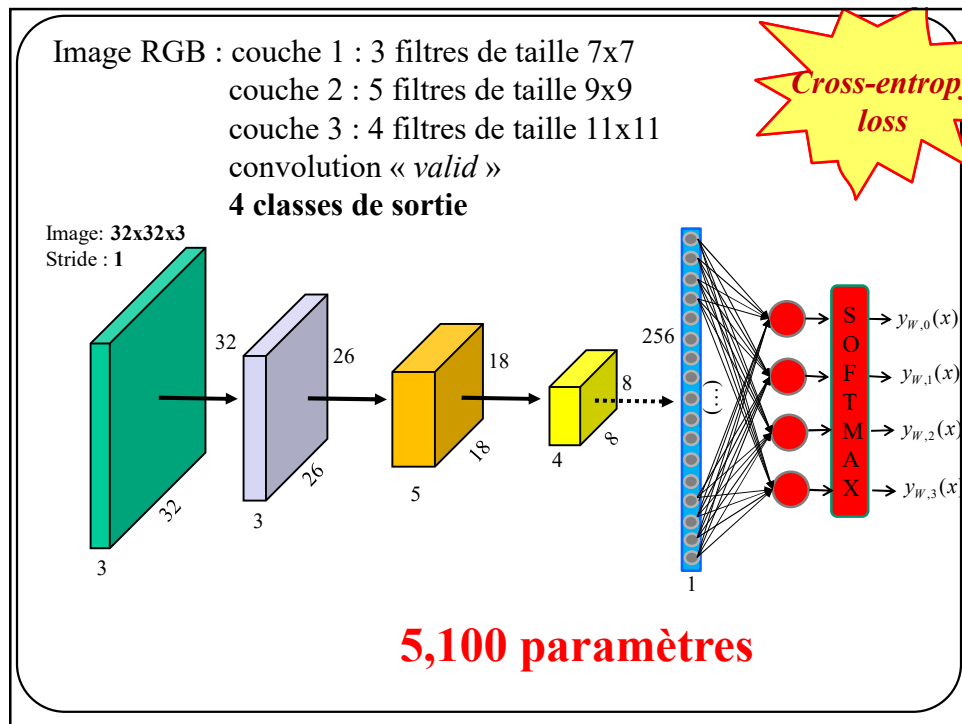


111

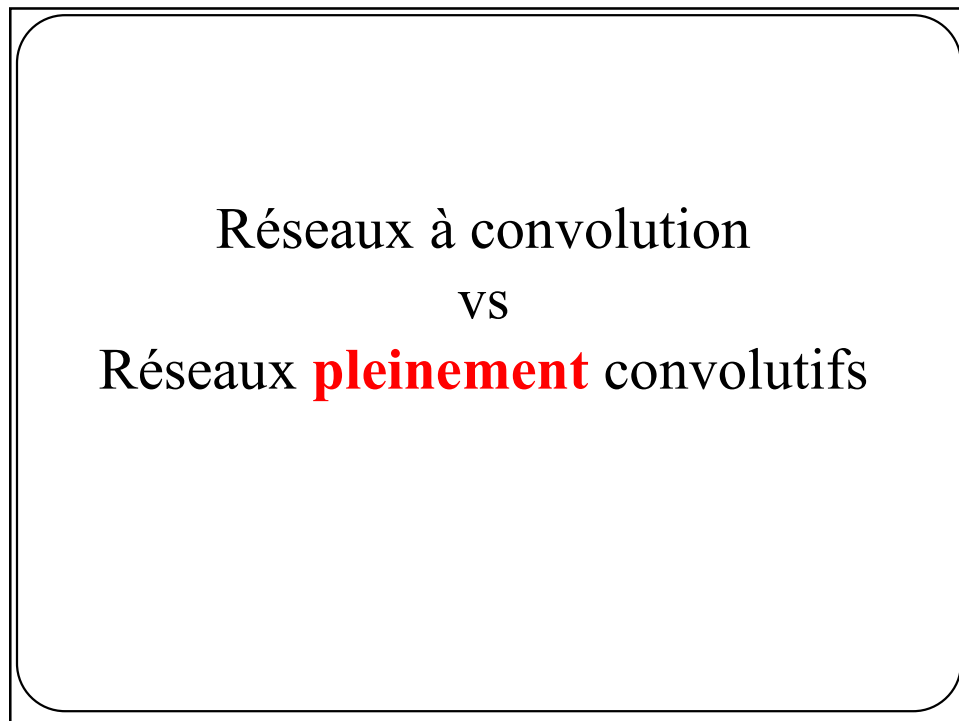


112

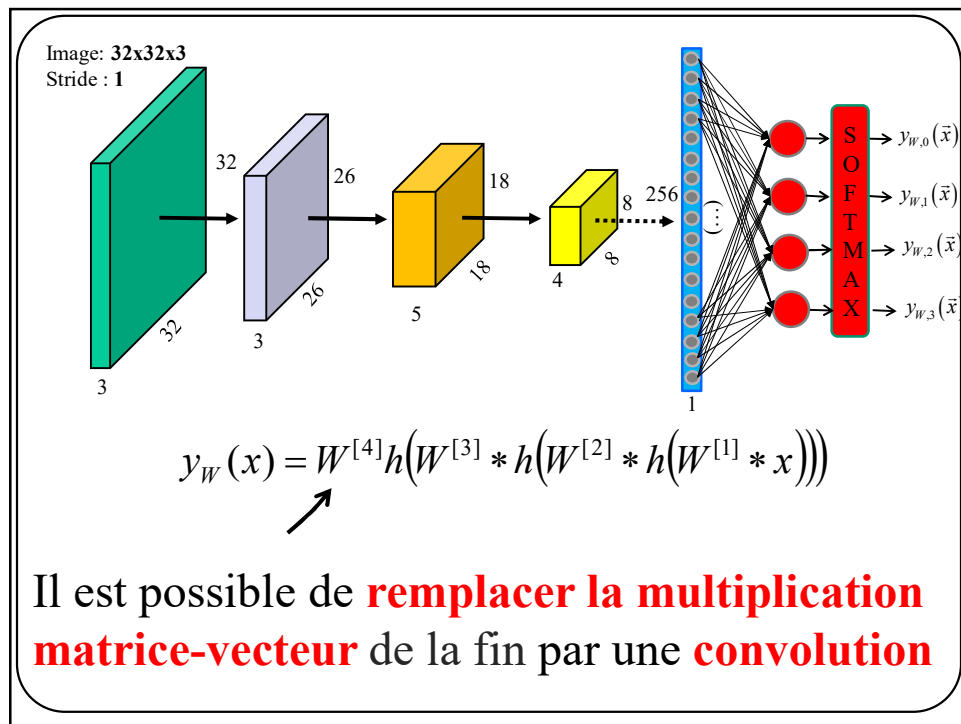




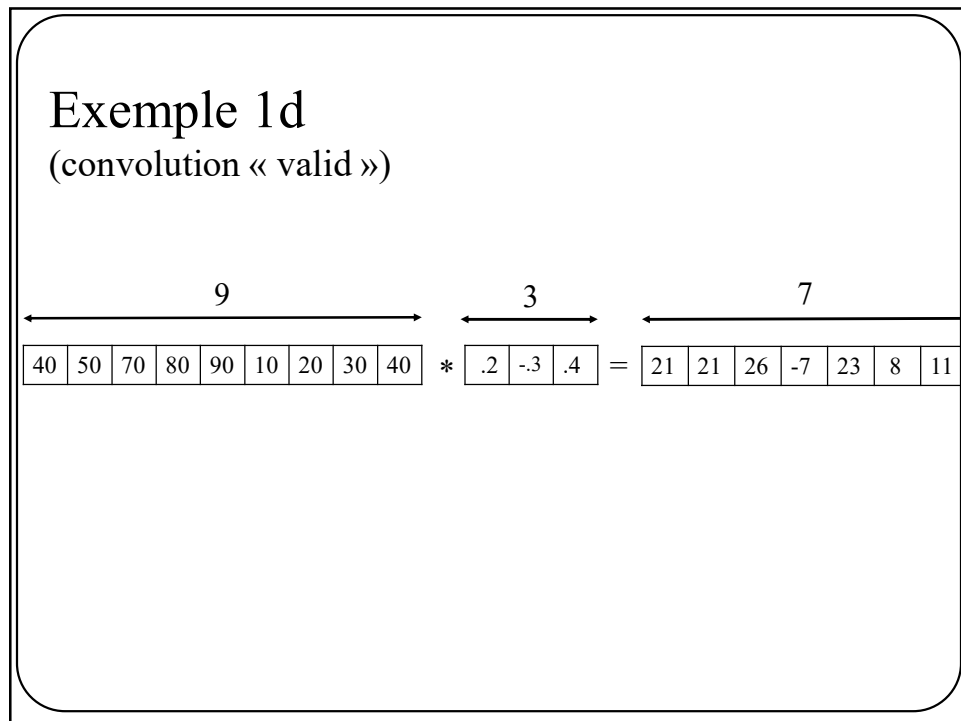
113



114



115



116

## Exemple 1d

(convolution « valid »)

$$\begin{array}{c}
 \xleftarrow{9} \quad \quad \quad \xleftarrow{5} \quad \quad \quad \xleftarrow{5} \\
 \boxed{40} \boxed{50} \boxed{70} \boxed{80} \boxed{90} \boxed{10} \boxed{20} \boxed{30} \boxed{40} * \boxed{.2} \boxed{-.3} \boxed{.4} \boxed{-.5} \boxed{.6} = \boxed{35} \boxed{-18} \boxed{33} \boxed{1} \boxed{32}
 \end{array}$$

117

## Exemple 1d

(convolution « valid »)

$$\begin{array}{c}
 \xleftarrow{9} \quad \quad \quad \xleftarrow{7} \quad \quad \quad \xleftarrow{3} \\
 \boxed{40} \boxed{50} \boxed{70} \boxed{80} \boxed{90} \boxed{10} \boxed{20} \boxed{30} \boxed{40} * \boxed{.2} \boxed{-.3} \boxed{.4} \boxed{-.5} \boxed{.6} \boxed{-.7} \boxed{.8} = \boxed{44} \boxed{-8} \boxed{44}
 \end{array}$$

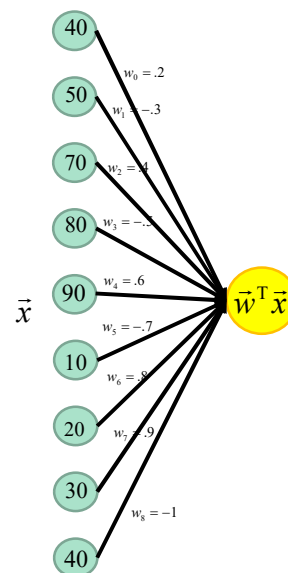
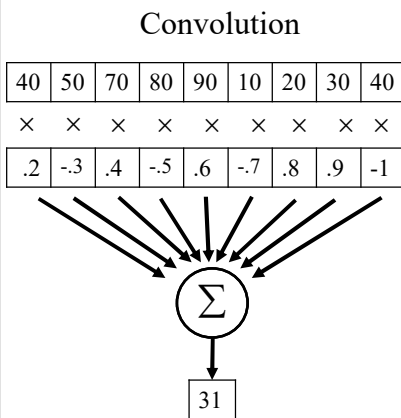
118

Taille filtre = nb de neurones couche précédente

$$\begin{array}{cccccccccc}
 \xleftarrow{9} & & & & & & & & & \xleftarrow{9} & & & \xleftarrow{1} \\
 \hline
 40 & 50 & 70 & 80 & 90 & 10 & 20 & 30 & 40 & * & .2 & -.3 & .4 & -.5 & .6 & -.7 & .8 & .9 & -1 & = & 31
 \end{array}$$

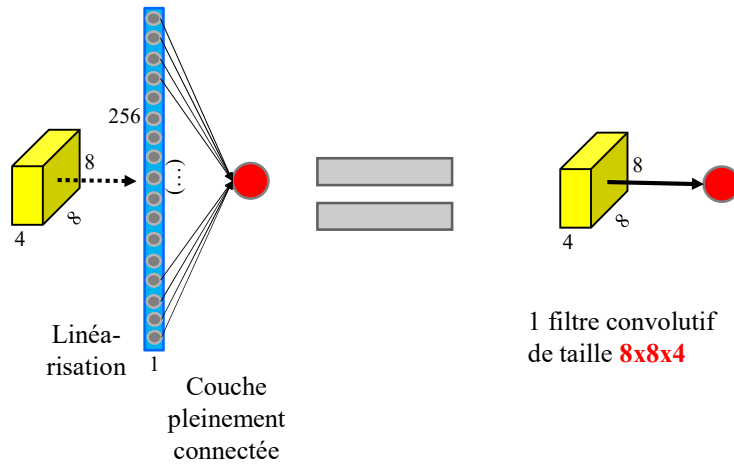
119

Signal d'entrée de **taille 9** convolué avec un filtre « same » de **taille 9** correspond à une **couche pleinement connectée**



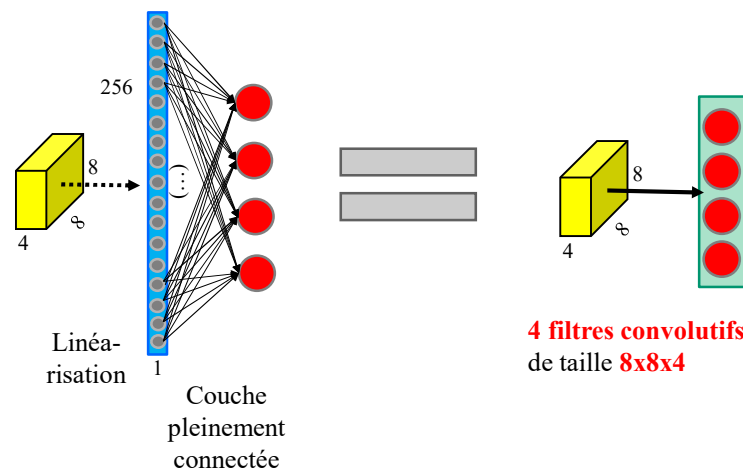
120

Même chose pour une **convolution 2D**

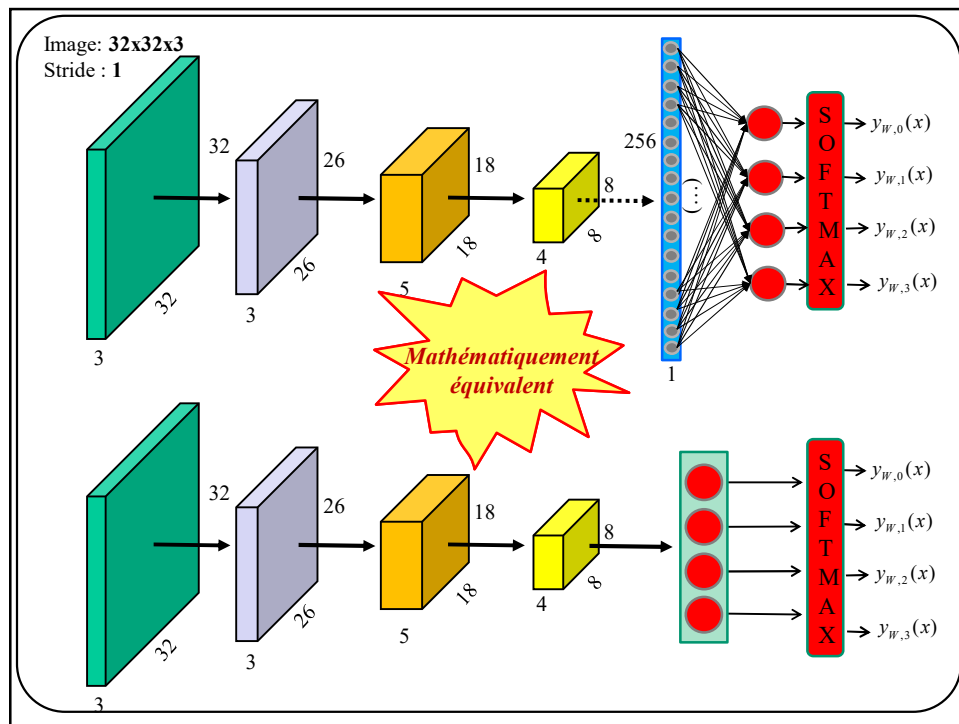


121

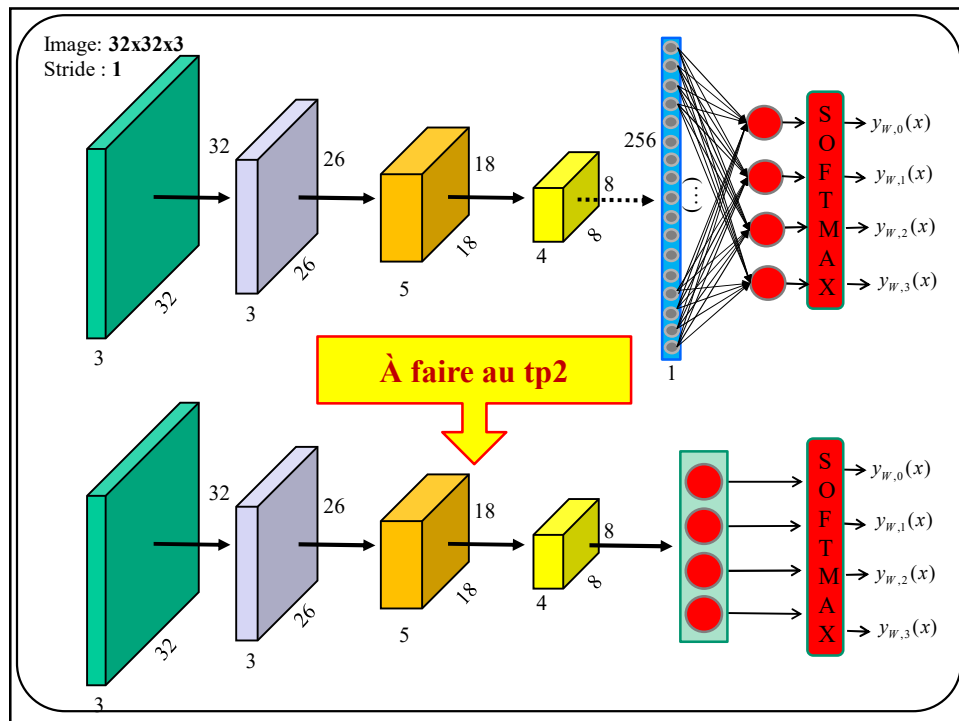
Même chose pour une **convolution 2D**



122



123



124

## Configurations équivalentes

couche 1 : 3 filtres de taille 7x7  
couche 2 : 5 filtres de taille 9x9  
couche 3 : 4 filtres de taille 11x11  
**couche 4 pleinement connectée 256x4**  
Softmax

couche 1 : 3 filtres de taille 7x7  
couche 2 : 5 filtres de taille 9x9  
couche 3 : 4 filtres de taille 11x11  
**couche 4 : 4 filtres de taille 8x8**  
Softmax

En fait, presque équivalent ...

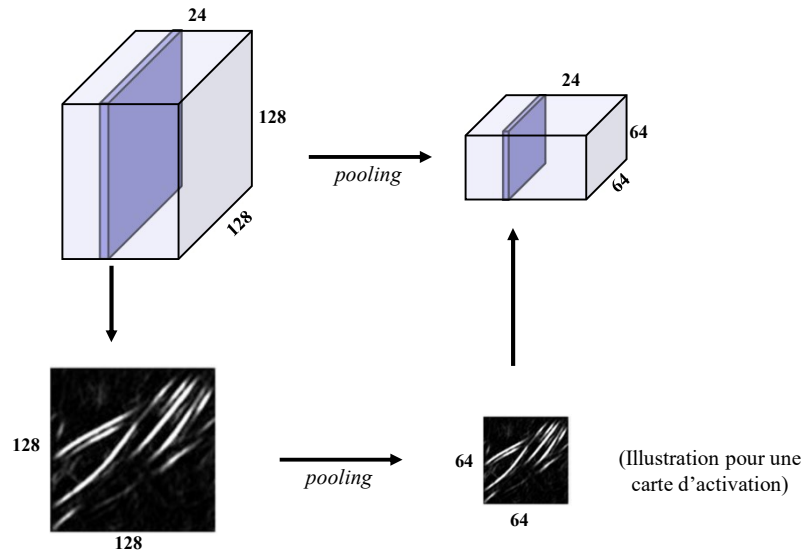
**Question : qu'arrive-t-il si on remplace l'image 32x32x3 par une image 64x64x3?**

125

# *Pooling*

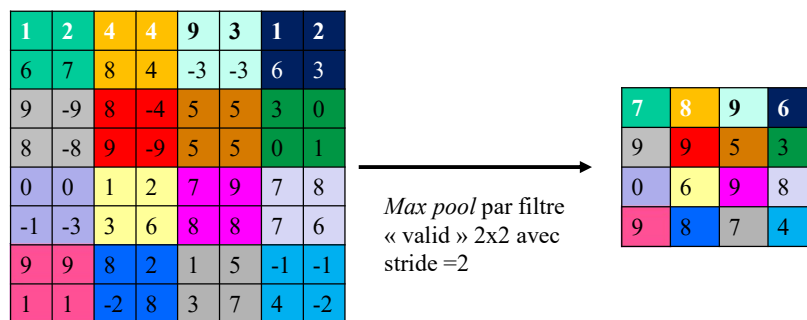
126

## Réduction de la taille des cartes d'activation



127

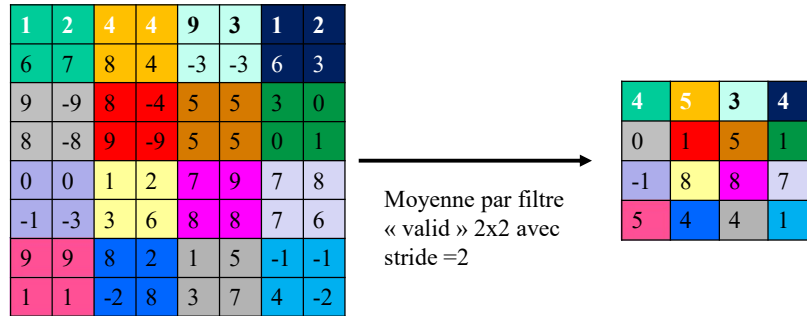
## Max pooling



128

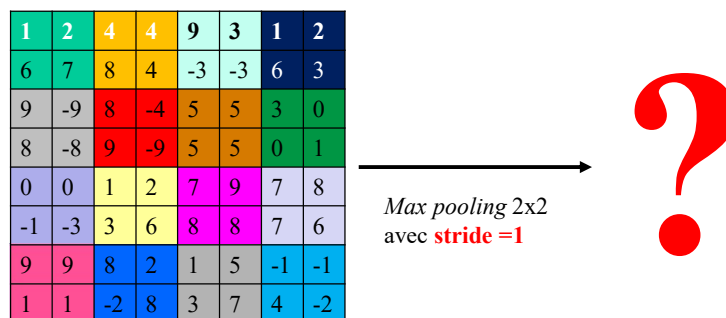


## Mean pooling



129

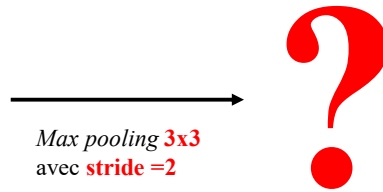
## Max pooling



130

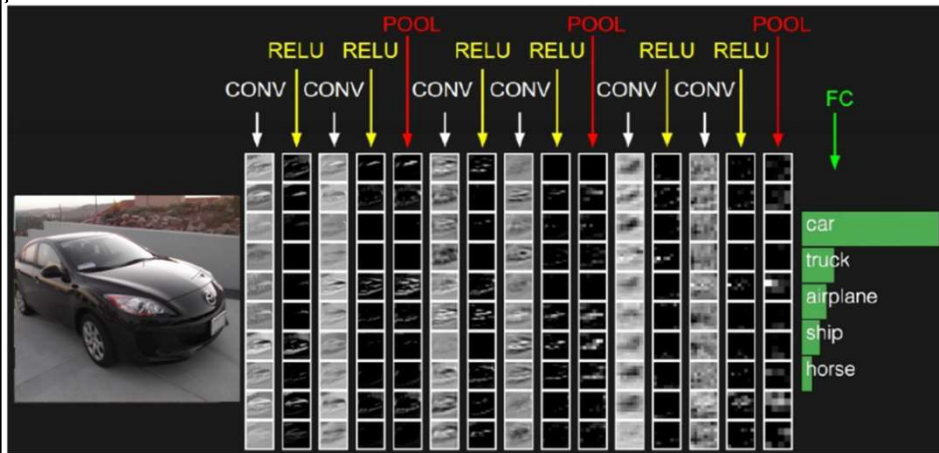
## Max pooling

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



131

## Illustration d'un CNN complet



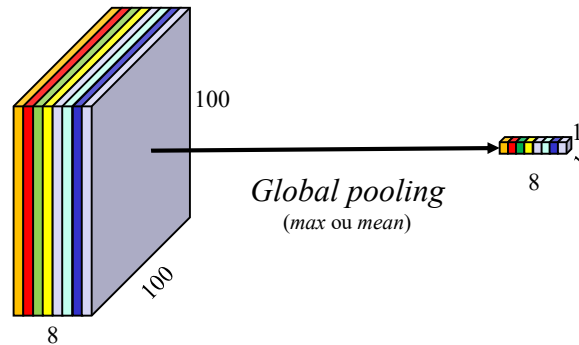
Crédit : cs231 Stanford

132

## *Global pooling*

*Max ou Mean pooling « valid » avec un filtre de la taille des canaux*

**Résultat** : un **vecteur** de la taille du nombre de canaux



133

## **Multiplication matricielle parcimonieuse**

<https://towardsdatascience.com/a-comprehensive-introduction-to-different-types-of-convolutions-in-deep-learning-669281e58215>

134

Il est **plus rapide** de multiplier des matrices que de les convoluer.

Ex.: convolution « valid », un canal d'entrée et une carte d'activation, filtre 3x3

Entrée		Filtre		
X0	X1	X2	X3	
X4	X5	X6	X7	
X8	X9	X10	X11	
X12	X13	X14	X15	

\*

W0	W1	W2
W3	W4	W5
W6	W7	W8

=

Y1	Y2
Y3	Y4

135

Il est **plus rapide** de multiplier des matrices que de les convoluer.

Ex.: convolution « valid », un canal d'entrée et une carte d'activation, filtre 3x3

Entrée		Filtre		
X0	X1	X2	X3	
X4	X5	X6	X7	
X8	X9	X10	X11	
X12	X13	X14	X15	

\*

W0	W1	W2
W3	W4	W5
W6	W7	W8

=

Y0	Y1
Y2	Y3

On peut **remplacer** une **convolution** par une **multiplication matrice-matrice** ou **matrice-vecteur**.  
De façons :

- 1- en **linéarisant** l'entrée et en « **matriciant** » le filtre
- 2- en **linéarisant** le filtre et en « **matriciant** » l'entrée

136

## Rappel

Ex.: convolution « *valid* », un canal d'entrée et une carte d'activation, filtre 3x3

W0	W1	W2	X3
W3	W4	W5	X7
W6	W7	W8	X11
X12	X13	X14	X15

<b>Y0</b>	Y1
Y2	Y3

$$\mathbf{Y0} = W0.X0 + W1.X1 + W2.X2 + W3.X4 + W4.X5 + W5.X6 + W6.X8 + W7.X9 + W8.X10$$

137

## Rappel

Ex.: convolution « *valid* », un canal d'entrée et une carte d'activation, filtre 3x3

X0	W0	W1	W2
X4	W3	W4	W5
X8	W6	W7	W8
X12	X13	X14	X15

Y0	<b>Y1</b>
Y2	Y3

$$\mathbf{Y1} = W0.X1 + W1.X2 + W2.X3 + W3.X5 + W4.X6 + W5.X7 + W6.X9 + W7.X10 + W8.X11$$

138

## Rappel

Ex.: convolution « *valid* », un canal d'entrée et une carte d'activation, filtre 3x3

X0	X1	X2	X3
W0	W1	W2	X7
W3	W4	W5	X11
W6	W7	W8	X15

Y0	Y1
<b>Y2</b>	Y3

$$\mathbf{Y2} = W0.X4 + W1.X5 + W2.X6 + W3.X8 + W4.X9 + W5.X10 + W6.X12 + W7.X13 + W8.X14$$

139

## Rappel

Ex.: convolution « *valid* », un canal d'entrée et une carte d'activation, filtre 3x3

X0	X1	X2	X3
X4	W0	W1	W2
X8	W3	W4	W5
X12	W6	W7	W8

Y0	Y1
Y2	<b>Y3</b>

$$\mathbf{Y3} = W0.X5 + W1.X6 + W2.X7 + W3.X9 + W4.X10 + W5.X11 + W6.X13 + W7.X14 + W8.X15$$

140

## Linéarisation de l'entrée et « matriçage » du filtre

Ex.: convolution « *valid* », un canal d'entrée et une carte d'activation, filtre 3x3

W0	W1	W2	0	W3	W4	W5	0	W6	W7	W8	0	0	0	0	0
0	W0	W1	W2	0	W3	W4	W5	0	W6	W7	W8	0	0	0	0
0	0	0	0	W0	W1	W2	0	W3	W4	W5	0	W6	W7	W8	0
0	0	0	0	0	W0	W1	W2	0	W3	W4	W5	0	W6	W7	W8

 $\times$ 

X0
X1
X2
X3
X4
X5
X6
X7
X8
X9
X10
X11
X12
X13
X14
X15

 $=$ 

Y0
Y1
Y2
Y3

141

## Linéarisation de l'entrée et « matriçage » du filtre

Ex.: convolution « *valid* », un canal d'entrée et une carte d'activation, **filtre 2x2**

Entrée

X0	X1	X2	X3
X4	X5	X6	X7
X8	X9	X10	X11
X12	X13	X14	X15

Filtre

W0	W1
W2	W3

\*

=

Y0	Y1	Y2
Y3	Y4	Y5
Y6	Y7	Y8

142

## Linéarisation de l'entrée et « matriçage » du filtre

Ex.: convolution « valid », un canal d'entrée et une carte d'activation, **filtre 2x2**

W0	W1	0	0	W2	W3	0	0	0	0	0	0	0	0	0	0	0	X0	
0	W0	W1	0	0	W2	W3	0	0	0	0	0	0	0	0	0	0	X1	
0	0	W0	W1	0	0	W2	W3	0	0	0	0	0	0	0	0	0	X2	
0	0	0	0	W0	W1	0	0	W2	W3	0	0	0	0	0	0	0	X3	
0	0	0	0	0	W0	W1	0	0	W2	W3	0	0	0	0	0	0	X4	Y0
0	0	0	0	0	0	W0	W1	0	0	W2	W3	0	0	0	0	0	X5	Y1
0	0	0	0	0	0	0	W0	W1	0	0	W2	W3	0	0	0	0	X6	Y2
0	0	0	0	0	0	0	0	W0	W1	0	0	W2	W3	0	0	0	X7	Y3
0	0	0	0	0	0	0	0	0	W0	W1	0	0	W2	W3	0	0	X8	Y4
0	0	0	0	0	0	0	0	0	0	W0	W1	0	0	W2	W3	0	X9	Y5
0	0	0	0	0	0	0	0	0	0	0	W0	W1	0	0	W2	W3	X10	Y6
0	0	0	0	0	0	0	0	0	0	0	0	W0	W1	0	0	W2	X11	Y7
0	0	0	0	0	0	0	0	0	0	0	0	0	W0	W1	0	0	X12	Y8
																	X13	
																	X14	
																	X15	

143

## Linéarisation de l'entrée et « matriçage » du filtre

Ex.: convolution « valid », un canal d'entrée et **deux cartes d'activation**, filtre 2x2

Entrée					Filtre					
X0	X1	X2	X3	*	W0	W1	=	Y0	Y1	Y2
X4	X5	X6	X7		W2	W3		Y3	Y4	Y5
X8	X9	X10	X11		W4	W5		Y6	Y7	Y8
X12	X13	X14	X15		W6	W7		Y9	Y10	Y11
								Y12	Y13	Y14
								Y15	Y16	Y17

144



## Linéarisation de l'entrée et « matriçage » du filtre

Ex.: convolution « valid », un canal d'entrée et  
**deux cartes d'activation**, filtre 2x2

W4	W5	0	0	W6	W7	0	0	0	0	0	0	0	0	0	0	0	0	X0
0	W0	W1	0	0	W2	W3	0	0	0	0	0	0	0	0	0	0	0	X1
0	0	W0	W1	0	0	W2	W3	0	0	0	0	0	0	0	0	0	0	X2
0	0	0	W0	W1	0	0	W2	W3	0	0	0	0	0	0	0	0	0	X3
0	0	0	0	0	W0	W1	0	0	W2	W3	0	0	0	0	0	0	0	X4
0	0	0	0	0	0	W0	W1	0	0	W2	W3	0	0	0	0	0	0	X5
0	0	0	0	0	0	0	W0	W1	0	0	W2	W3	0	0	0	0	0	X6
0	0	0	0	0	0	0	0	W0	W1	0	0	W2	W3	0	0	0	0	X7
0	0	0	0	0	0	0	0	0	W0	W1	0	0	W2	W3	0	0	0	X8
0	0	0	0	0	0	0	0	0	0	W0	W1	0	0	W2	W3	0	0	X9
0	0	0	0	0	0	0	0	0	0	0	W0	W1	0	0	W2	W3	0	X10
0	0	0	0	0	0	0	0	0	0	0	0	W0	W1	0	0	W2	W3	X11
0	0	0	0	0	0	0	0	0	0	0	0	0	W0	W1	0	0	W2	X12
0	0	0	0	0	0	0	0	0	0	0	0	0	0	W0	W1	0	0	X13
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	W0	W1	0	X14
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	W0	W1	X15

145

## Linéarisation de l'entrée et « matriçage » du filtre

Ex.: convolution « valid », un canal d'entrée, une carte d'activation, filtre 2x2  
**mini-batch de 2 entrées.**

Entrée					Filtre					
X0	X1	X2	X3	*	W0	W1	=	Y0	Y1	Y2
X4	X5	X6	X7		W2	W3		Y3	Y4	Y5
X8	X9	X10	X11					Y6	Y7	Y8
X12	X13	X14	X15					Y9	Y10	Y11
X16	X17	X18	X19					Y12	Y13	Y14
X20	X21	X22	X23					Y15	Y16	Y17
X24	X25	X26	X27							
X28	X29	X30	X31							

146

## Linéarisation de l'entrée et « matriçage » du filtre

Ex.: convolution « *valid* », un canal d'entrée, une carte d'activation, filtre 2x2 **mini-batch de 2 entrées**.

W0	W1	0	0	W2	W3	0	0	0	0	0	0	0	0	0	0	0	0
0	W0	W1	0	0	W2	W3	0	0	0	0	0	0	0	0	0	0	0
0	0	W0	W1	0	0	W2	W3	0	0	0	0	0	0	0	0	0	0
0	0	0	0	W0	W1	0	0	W2	W3	0	0	0	0	0	0	0	0
0	0	0	0	0	W0	W1	0	0	W2	W3	0	0	0	0	0	0	0
0	0	0	0	0	0	W0	W1	0	0	W2	W3	0	0	0	0	0	0
0	0	0	0	0	0	0	0	W0	W1	0	0	W2	W3	0	0	0	0
0	0	0	0	0	0	0	0	0	W0	W1	0	0	W2	W3	0	0	0
0	0	0	0	0	0	0	0	0	0	W0	W1	0	0	W2	W3	0	0

 $\times$ 

X0	X16
X1	X17
X2	X18
X3	X19
X4	X20
X5	X21
X6	X22
X7	X23
X8	X24
X9	X25
X10	X26
X11	X27
X12	X28
X13	X29
X14	X30
X15	X31

 $=$ 

Y0	Y9
Y1	Y10
Y2	Y11
Y3	Y12
Y4	Y13
Y5	Y14
Y6	Y15
Y7	Y16
Y8	Y17

147

## Linéarisation de l'entrée et « matriçage » du filtre

Ex.: convolution « *valid* », **2 canaux d'entrée**, une carte d'activation, filtre 2x2 **mini-batch de 1 entrée**.

Entrée

X0	X1	X2
X3	X4	X5
X6	X7	X8
X9	X10	X11
X12	X13	X14
X15	X16	X17

\*

Filtre

W0	W1
W2	W3
W4	W5
W6	W7

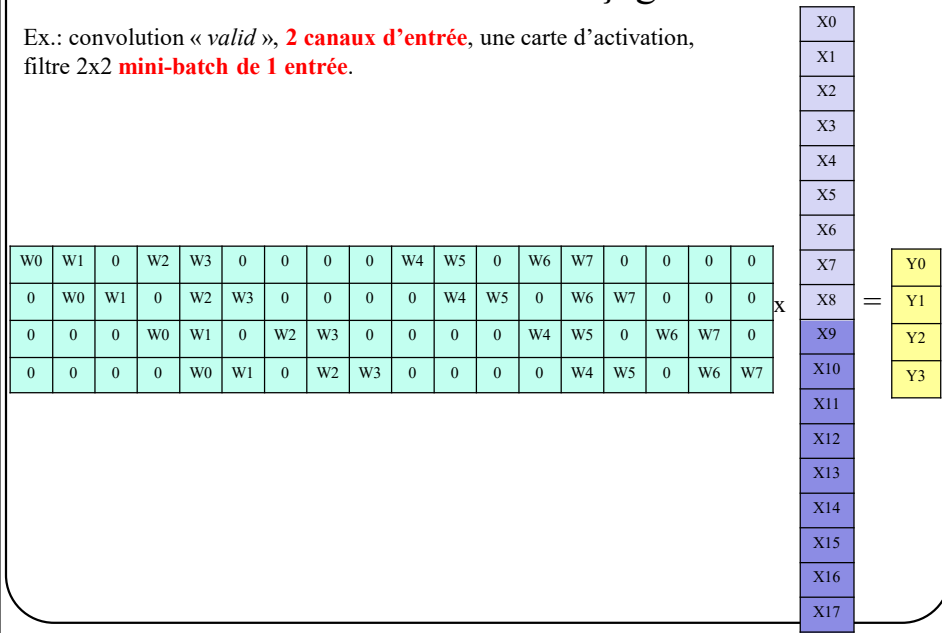
=

Y0	Y1
Y2	Y3

148

## Linéarisation de l'entrée et « matriçage » du filtre

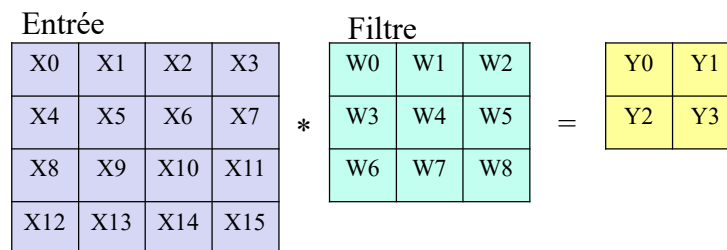
Ex.: convolution « *valid* », **2 canaux d'entrée**, une carte d'activation, filtre 2x2 **mini-batch de 1 entrée**.



149

## On peut faire la même chose mais en **linéarisant le filtre** et en « **matriçant** » l'entrée

Ex.: convolution « *valid* »



150

On peut faire la même chose mais en **linéarisant le filtre** et en « **matriçant** » l'entrée

Ex.: convolution « *valid* »

Entrée

X0	X1	X2	X3
X4	X5	X6	X7
X8	X9	X10	X11
X12	X13	X14	X15

X0
X1
X2
X4
X5
X6
X8
X9
X10

151

On peut faire la même chose mais en **linéarisant le filtre** et en « **matriçant** » l'entrée

Ex.: convolution « *valid* »

Entrée

X0	X1	X2	X3
X4	X5	X6	X7
X8	X9	X10	X11
X12	X13	X14	X15

X0	X1
X1	X2
X2	X3
X4	X5
X5	X6
X6	X7
X8	X9
X9	X10
X10	X11

152

On peut faire la même chose mais en **linéarisant le filtre** et en « **matriçant** » l'entrée

Ex.: convolution « *valid* »

Entrée

X0	X1	X2	X3
X4	X5	X6	X7
X8	X9	X10	X11
X12	X13	X14	X15

X0	X1	X4
X1	X2	X5
X2	X3	X6
X4	X5	X8
X5	X6	X9
X6	X7	X10
X8	X9	X11
X9	X10	X12
X10	X11	X13

153

On peut faire la même chose mais en **linéarisant le filtre** et en « **matriçant** » l'entrée

Ex.: convolution « *valid* »

Entrée

X0	X1	X2	X3
X4	X5	X6	X7
X8	X9	X10	X11
X12	X13	X14	X15

X0	X1	X4	X5
X1	X2	X5	X6
X2	X3	X6	X7
X4	X5	X8	X9
X5	X6	X9	X10
X6	X7	X10	X11
X8	X9	X11	X13
X9	X10	X12	X14
X10	X11	X13	X15

154

On peut faire la même chose mais en **linéarisant le filtre** et en « **matriçant** » l'entrée

Ex.: convolution « *valid* »

W0	W1	W2	W3	W4	W5	W6	W7	W8
----	----	----	----	----	----	----	----	----

 $\times$ 

X0	X1	X4	X5
X1	X2	X5	X6
X2	X3	X6	X7
X4	X5	X8	X9
X5	X6	X9	X10
X6	X7	X10	X11
X8	X9	X11	X13
X9	X10	X12	X14
X10	X11	X13	X15

 $=$ 

Y1	Y2	Y3	Y4
----	----	----	----

155

**Ou encore...**

Ex.: convolution « *valid* »

X0	X1	X2	X4	X5	X6	X8	X9	X10
X1	X2	X3	X5	X6	X7	X9	X10	X11
X4	X5	X6	X8	X9	X10	X11	X12	X13
X5	X6	X7	X9	X10	X11	X13	X14	X15

 $\times$ 

W0
W1
W2
W3
W4
W5
W6
W7
W8

 $=$ 

Y0
Y1
Y2
Y3

156

On peut faire la même chose mais en **linéarisant le filtre** et en « **matriçant** » l'entrée

Exercice à la maison, voir comment cette 2<sup>e</sup> approche s'applique au cas à

- Plusieurs canaux en entrée
- Plusieurs cartes d'activation
- Plusieurs entrées (mini-batch)

Sinon, voir **im2col** du **travail pratique 2**.

157

Comment calculer la  
rétropropagation dans un CNN?

À faire au TP2

158