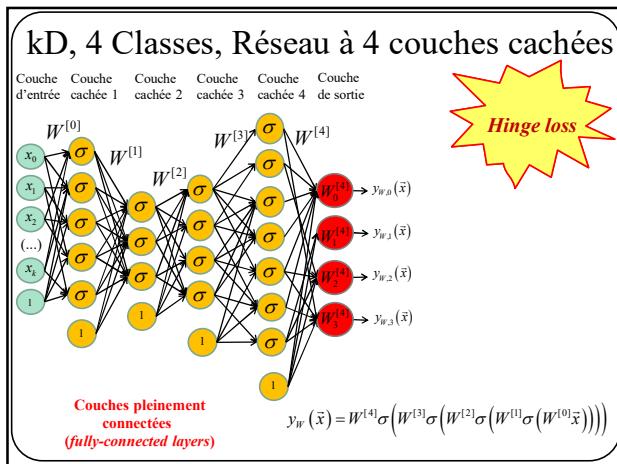
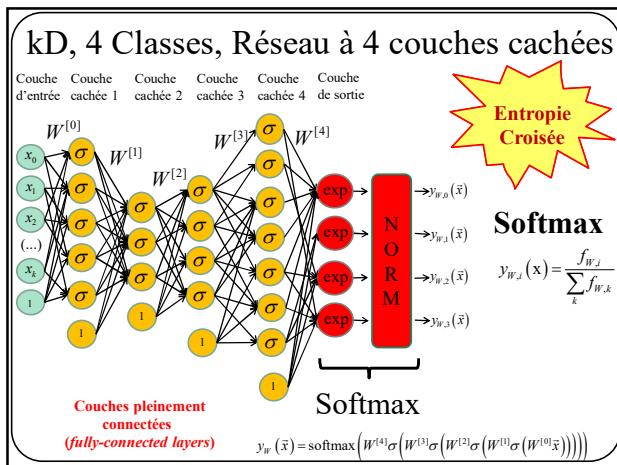


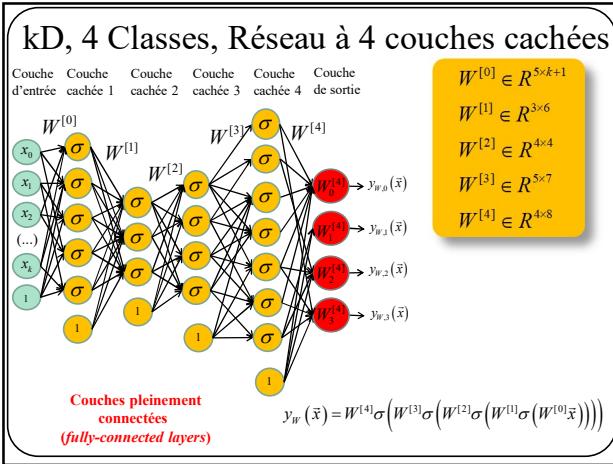
1



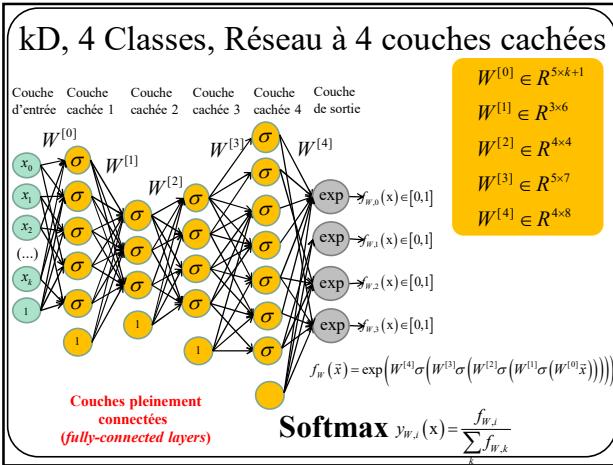
2



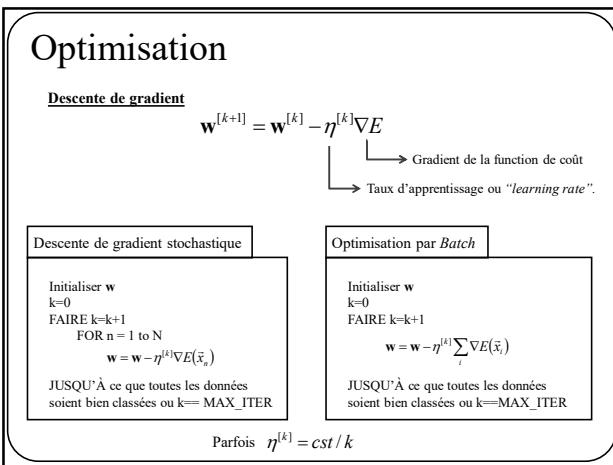
3



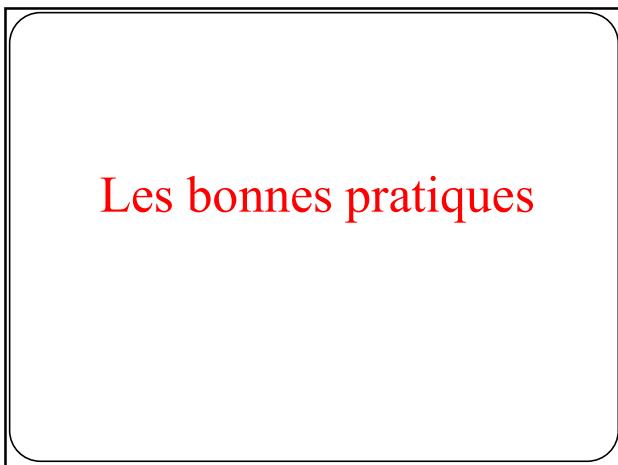
4



5



6



7

---



---



---



---



---



---



---



---

## Optimisation

**Descente de gradient**

$$\mathbf{w}^{[k+1]} = \mathbf{w}^{[k]} - \eta^{[k]} \nabla E$$

Essentiel au TP4

Gradient de la fonction de coût

Taux d'apprentissage ou "learning rate".

Optimisation par **mini-batch**

```

Initialiser w
k=0
FAIRE k=k+1
FAIRE n=0 à N par sauts de MBS /*Mini-batch size*/
    w = w - η[k] ∑i=aa+MBS ∇E(̄xi)
} Itération
JUSQU'À ce que toutes les données soient bien classées ou
k==MAX_ITER
  
```

8

---



---



---



---



---



---



---



---

## Optimisation

**Descente de gradient**

$$\mathbf{w}^{[k+1]} = \mathbf{w}^{[k]} - \eta^{[k]} \nabla E$$

Gradient de la fonction de coût

Taux d'apprentissage ou "learning rate".

Optimisation par **mini-batch**

```

Initialiser w
k=0
FAIRE k=k+1
FAIRE n=0 à N par sauts de MBS /*Mini-batch size*/
    w = w - η[k] ∑i=aa+MBS ∇E(̄xi)
} Epoch
JUSQU'À ce que toutes les données soient bien classées ou
k==MAX_ITER
  
```

9

---



---



---



---



---



---



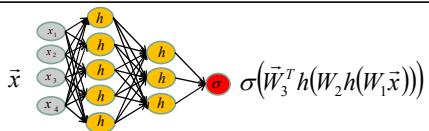
---



---

Mini-batch = **vectorisation** de la propagation avant et de la rétro-propagation

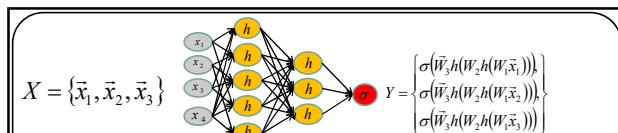
10



Propagation avant pour un réseau à 2 couches cachées (7 étapes)

Étape 2 →	$\vec{x}$	$\in IR^4$
	$W_1 \vec{x}$	$\in IR^5$
	$h(W_1 \vec{x})$	$\in IR^5$
	$W_2 h(W_1 \vec{x})$	$\in IR^3$
	$h(W_2 h(W_1 \vec{x}))$	$\in IR^3$
Étape 6 →	$\vec{W}_3^T (h(W_2 h(W_1 \vec{x})))$	$\in IR$
	$\sigma(\vec{W}_3^T (h(W_2 h(W_1 \vec{x}))))$	$\in IR$

11



Propagation avant pour un réseau à 2 couches cachées (7 étapes)

POUR i allant de 0 à 2	$\vec{x} = X[i]$	$\in IR^4$
	$W_i \vec{x}$	$\in IR^5$
	$h(W_i \vec{x})$	$\in IR^5$
	$W_2 h(W_i \vec{x})$	$\in IR^3$
	$h(W_2 h(W_i \vec{x}))$	$\in IR^3$
	$\vec{W}_3^T (h(W_2 h(W_i \vec{x})))$	$\in IR$
	$Y[i] = \sigma(\vec{W}_3^T (h(W_2 h(W_i \vec{x}))))$	$\in IR$

Solution naïve et peu efficace

TP3

12

## Solution

Il est plus efficace d'effectuer UNE multiplication matricielle que PLUSIEURS multiplications matrice-vecteur (exemple de la 2<sup>e</sup> étape, batch de 3)

$$W_1 \vec{x}_1 = \begin{pmatrix} w_{11} & w_{12} & w_{13} & w_{14} \\ w_{21} & w_{22} & w_{23} & w_{24} \\ w_{31} & w_{32} & w_{33} & w_{34} \\ w_{41} & w_{42} & w_{43} & w_{44} \\ w_{51} & w_{52} & w_{53} & w_{54} \end{pmatrix} \begin{bmatrix} a \\ b \\ c \\ d \end{bmatrix} = \begin{bmatrix} u_1 \\ u_2 \\ u_3 \\ u_4 \\ u_5 \end{bmatrix}$$
  

$$W_1 \vec{x}_2 = \begin{pmatrix} w_{11} & w_{12} & w_{13} & w_{14} \\ w_{21} & w_{22} & w_{23} & w_{24} \\ w_{31} & w_{32} & w_{33} & w_{34} \\ w_{41} & w_{42} & w_{43} & w_{44} \\ w_{51} & w_{52} & w_{53} & w_{54} \end{pmatrix} \begin{bmatrix} d \\ e \\ f \\ g \end{bmatrix} = \begin{bmatrix} v_1 \\ v_2 \\ v_3 \\ v_4 \\ v_5 \end{bmatrix}$$
  

$$W_1 \vec{x}_3 = \begin{pmatrix} w_{11} & w_{12} & w_{13} & w_{14} \\ w_{21} & w_{22} & w_{23} & w_{24} \\ w_{31} & w_{32} & w_{33} & w_{34} \\ w_{41} & w_{42} & w_{43} & w_{44} \\ w_{51} & w_{52} & w_{53} & w_{54} \end{pmatrix} \begin{bmatrix} h \\ i \\ j \\ k \end{bmatrix} = \begin{bmatrix} z_1 \\ z_2 \\ z_3 \\ z_4 \\ z_5 \end{bmatrix}$$

13

## Solution

Il est plus efficace d'effectuer UNE multiplication matricielle que PLUSIEURS matrice-vecteur (exemple de la 2<sup>e</sup> étape, batch de 3)

$$W_1 X = \begin{pmatrix} w_{11} & w_{12} & w_{13} & w_{14} \\ w_{21} & w_{22} & w_{23} & w_{24} \\ w_{31} & w_{32} & w_{33} & w_{34} \\ w_{41} & w_{42} & w_{43} & w_{44} \\ w_{51} & w_{52} & w_{53} & w_{54} \end{pmatrix} \begin{bmatrix} a & d & h \\ b & e & i \\ c & f & j \\ d & g & k \end{bmatrix} = \begin{bmatrix} u_1 & v_1 & z_1 \\ u_2 & v_2 & z_2 \\ u_3 & v_3 & z_3 \\ u_4 & v_4 & z_4 \\ u_5 & v_5 & z_5 \end{bmatrix}$$

14

## Solution

Il est plus efficace d'effectuer UNE multiplication matricielle que PLUSIEURS produits scalaires (exemple de la 6<sup>e</sup> étape, batch de 3)

$$(w_1 \quad w_2 \quad w_3) \begin{bmatrix} a \\ b \\ c \end{bmatrix} = (w_1 a + w_2 b + w_3 c)$$
  

$$(w_1 \quad w_2 \quad w_3) \begin{bmatrix} d \\ e \\ f \end{bmatrix} = (w_1 d + w_2 e + w_3 f)$$
  

$$(w_1 \quad w_2 \quad w_3) \begin{bmatrix} g \\ h \\ i \end{bmatrix} = (w_1 g + w_2 h + w_3 i)$$

15

## Solution

Il est plus efficace d'effectuer **UNE multiplication matricielle** que **PLUSIEURS** produits scalaires (**exemple de la 6<sup>e</sup> étape, batch de 3**)

$$\begin{bmatrix} w_1 & w_2 & w_3 \end{bmatrix} \begin{pmatrix} a & d & g \\ b & e & h \\ c & f & i \end{pmatrix} = \begin{pmatrix} w_1a + w_2b + w_3c \\ w_1d + w_2e + w_3f \\ w_1g + w_2h + w_3i \end{pmatrix} = Y$$

UNE  
multiplication  
matricielle

16

## Conclusion

100% du temps, on combine ensemble les données dans des **mini-batch** de 2 à 32 données.

17

## Les réseaux à convolution

18

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

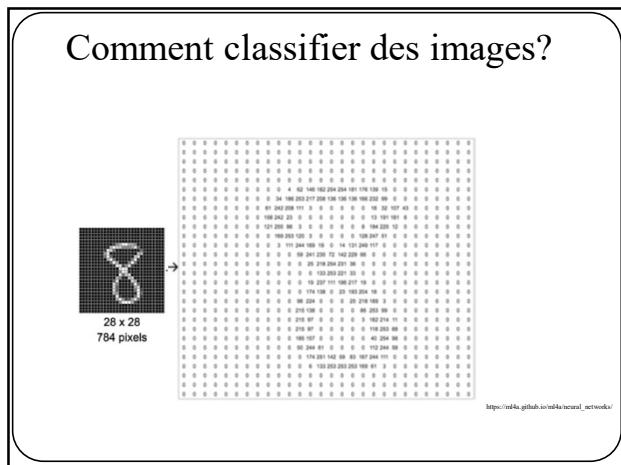
---

---

---

---

---



19

---

---

---

---

---

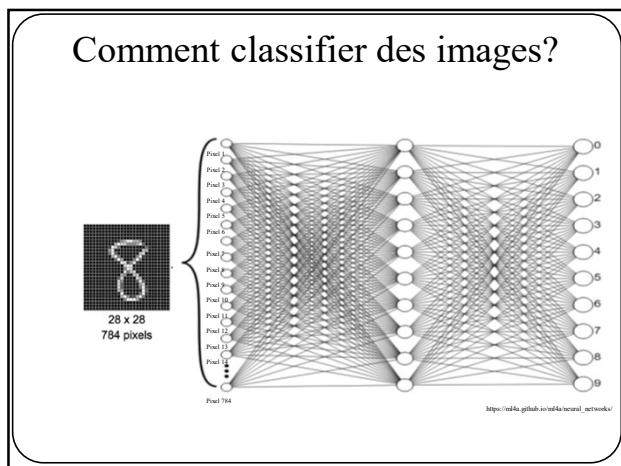
---

---

---

---

---



20

---

---

---

---

---

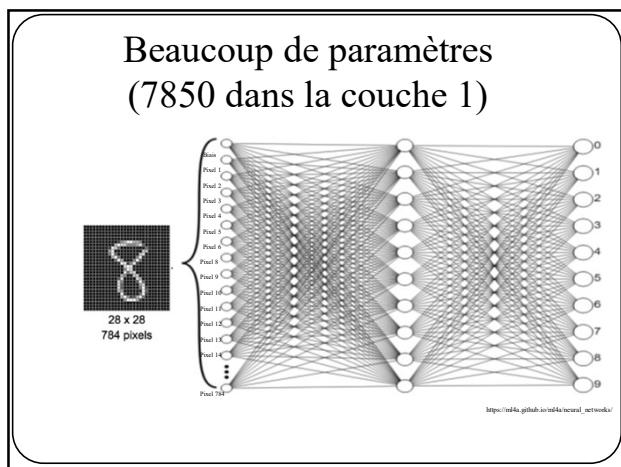
---

---

---

---

---



21

---

---

---

---

---

---

---

---

---

---

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

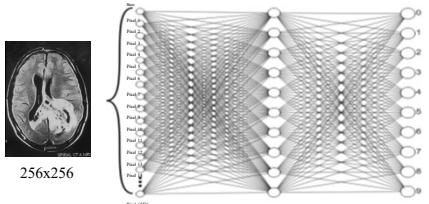


Image médicale (IRM de cerveau)

22

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

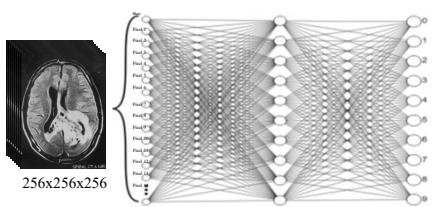


Image médicale 3D (IRM de cerveau)

23

Comment réduire le nombre de connections?



24

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

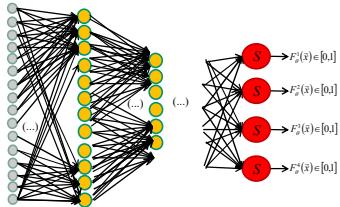
---

---

24

Comment réduire le nombre de connections?

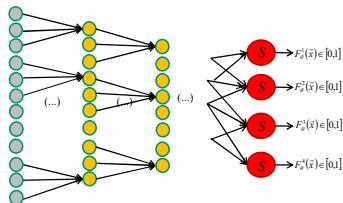
Les **couches pleinement connectées** (*fully-connected layers*) sont problématiques lorsque le **nombre de neurones** est élevé.



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

25

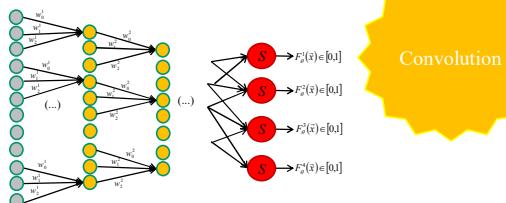
Solution : connexions partielles



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

26

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



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!

27

# Convolution et couche convolutionnelle 1D

28

---



---



---



---



---



---



---



---

## Exemple 1D de la convolution

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

(signal d'entrée)  
 $f(u)$   
 $\boxed{10|20|30|40|50}$

(filtre)  
 $W(u)$   
 $\boxed{.1|.2|.3}$

(filtre)  
 $W(-u)$   
 $\boxed{.3|.2|.1}$

$(f * W)(1)$

$\boxed{10|20|30|40|50}$   
 $x\ x\ x$   
 $\boxed{.3|.2|.1}$

$3+4+1=8$

$\downarrow$

$\boxed{4|4|4}$

$(f * W)(2)$

$\boxed{10|20|30|40|50}$   
 $x\ x\ x$   
 $\boxed{.3|.2|.1}$

$6+8+4=18$

$\downarrow$

$\boxed{4|4|4}$

$(f * W)(3)$

$\boxed{10|20|30|40|50}$   
 $x\ x\ x$   
 $\boxed{.3|.2|.1}$

$-9+8+5=4$

$\downarrow$

$\boxed{4|4|4}$

29

---



---



---



---



---



---



---



---

## En gros

**convolution = produit scalaire + translation**

---



---



---



---



---



---

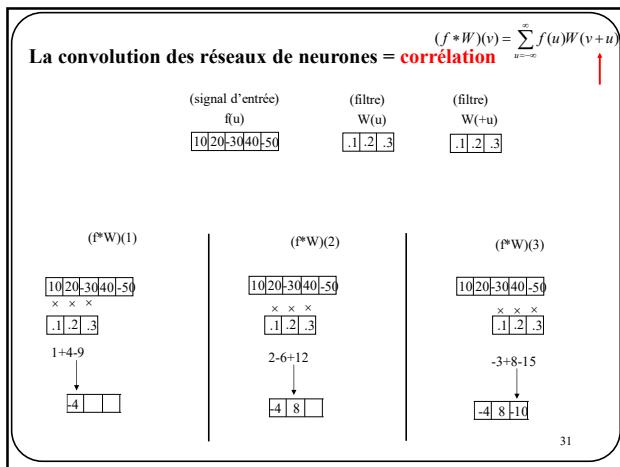


---

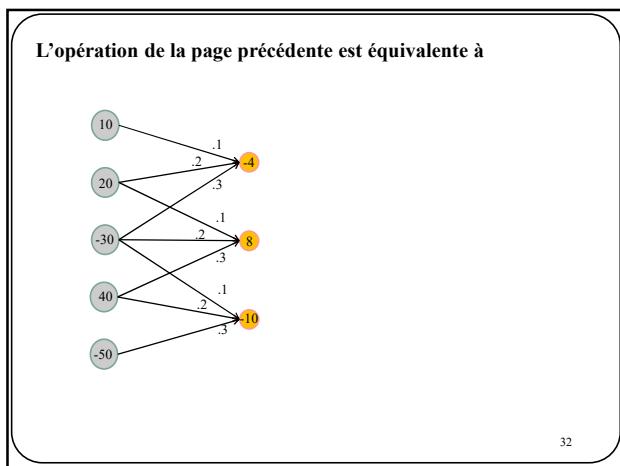


---

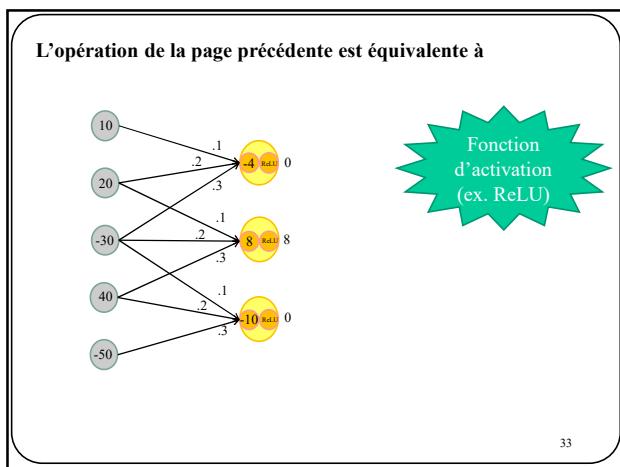
30



31

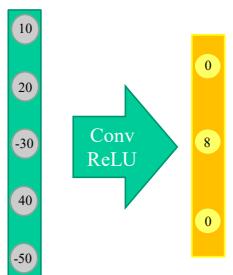


32



33

**Représentation graphique courante (simple)**



34

---

---

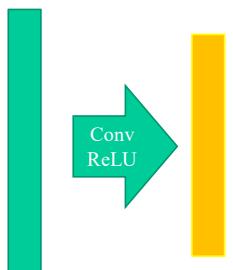
---

---

---

---

**Représentation graphique courante (encore plus simple)**



35

---

---

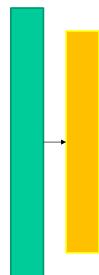
---

---

---

---

**Représentation graphique courante (vraiment ultra simple)**



36

---

---

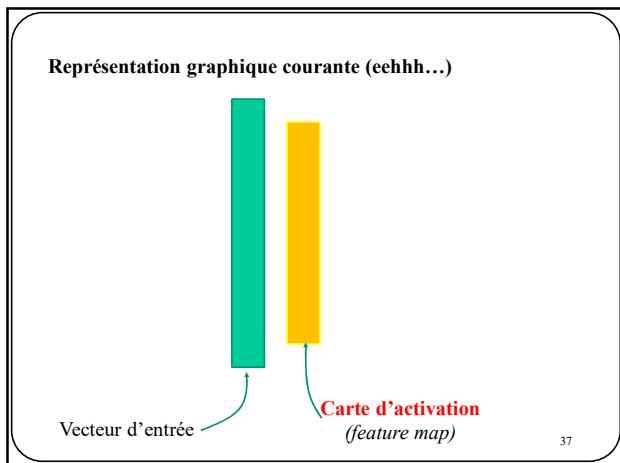
---

---

---

---

36



37

---



---



---



---



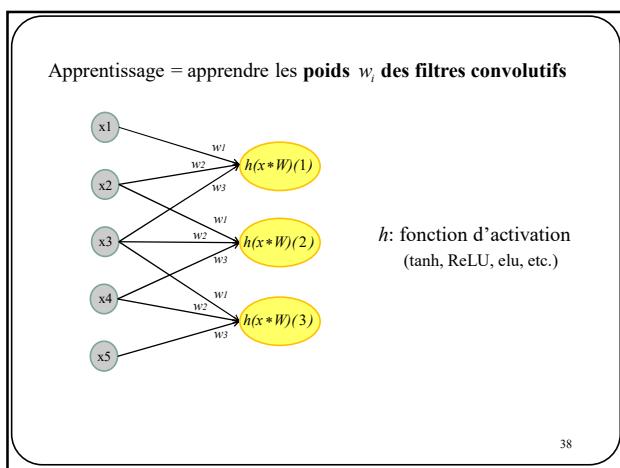
---



---



---



38

---



---



---



---



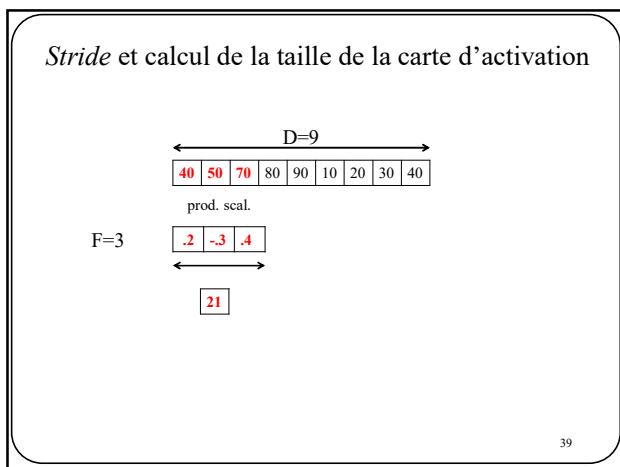
---



---



---



39

---



---



---



---



---

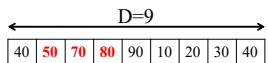


---



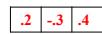
---

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



prod. scal.

F=3



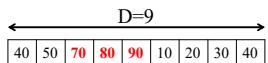
21

Stride = 1

40

40

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



prod. scal.

F=3

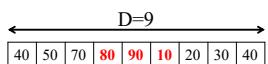


21 21 26

41

41

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



prod. scal.

F=3



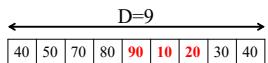
21 21 26 -7

42

42

14

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



prod. scal.

F=3

.2 .-3 .4

[21 | 21 | 26 | -7 | **23**]

43

43

---

---

---

---

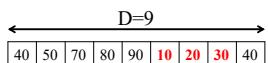
---

---

---

---

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



prod. scal.

F=3

.2 .-3 .4

[21 | 21 | 26 | -7 | 23 | **8**]

44

44

---

---

---

---

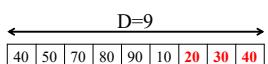
---

---

---

---

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



prod. scal.

F=3

.2 .-3 .4

[21 | 21 | 26 | -7 | 23 | 8 | **11**]

45

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

---

---

---

---

---

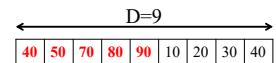
---

---

---

45

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



prod. scal.

F=5

.2	-3	.4	-5	.6
----	----	----	----	----

35

46

46

---

---

---

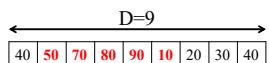
---

---

---

---

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



prod. scal.

F=5

.2	-3	.4	-5	.6
----	----	----	----	----

35 -18

Stride = 1

47

---

---

---

---

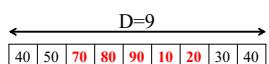
---

---

---

47

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



prod. scal.

F=5

.2	-3	.4	-5	.6
----	----	----	----	----

35 -18 33

48

---

---

---

---

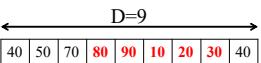
---

---

---

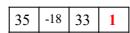
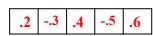
48

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



prod. scal.

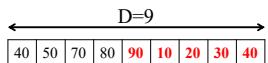
F=5



49

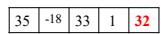
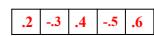
49

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



prod. scal.

F=5

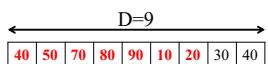


Taille de la carte d'activation = **5**

50

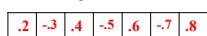
50

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



prod. scal.

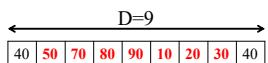
F=7



51

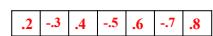
51

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



prod. scal.

F=7



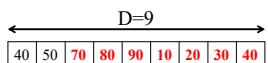
44 -8

Stride = 1

52

52

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



prod. scal.

F=7



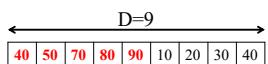
44	-8	44
----	----	----

Taille de la carte d'activation = 3

53

53

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



prod scal

F=5



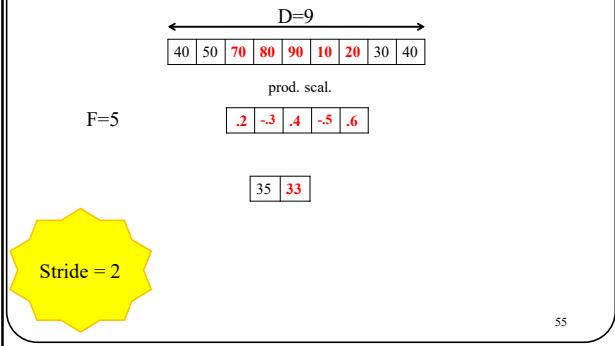
35

54

54

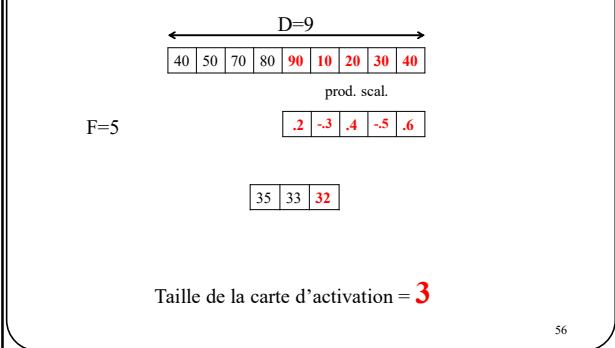
18

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



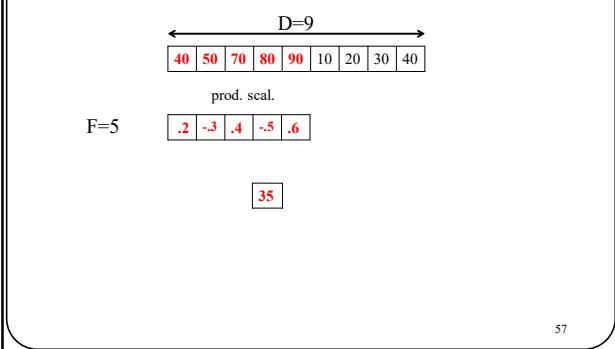
55

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



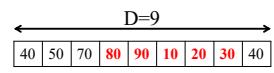
56

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



57

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



prod. scal.

F=5

.2	-.3	.4	-.5	.6
----	-----	----	-----	----

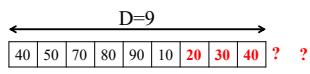
35	1
----	---

Stride = 3

58

58

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



prod. scal.

F=5

.2	-.3	.4	-.5	.6
----	-----	----	-----	----

35	1
----	---

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

59

59

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

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



60

60

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

?	10	20	30	40	50
$\times$	$\times$	$\times$			
1	2	3			

**Comment gérer les bords?**

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



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



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

61

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

Comment gérer les bords?

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

$f(u)$	$(F^*W)(u)$
$[0 \mid 10 \mid 20 \mid 30 \mid 40 \mid 50 \mid 0]$	$[8 \mid -4 \mid 8 \mid 10 \mid -6]$

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

De loin l'option la plus utilisée

$f(u)$	$(F^*W)(u)$
$[10 \mid 10 \mid 20 \mid 30 \mid 40 \mid 50 \mid 50]$	$[9 \mid -4 \mid 8 \mid 10 \mid -2]$

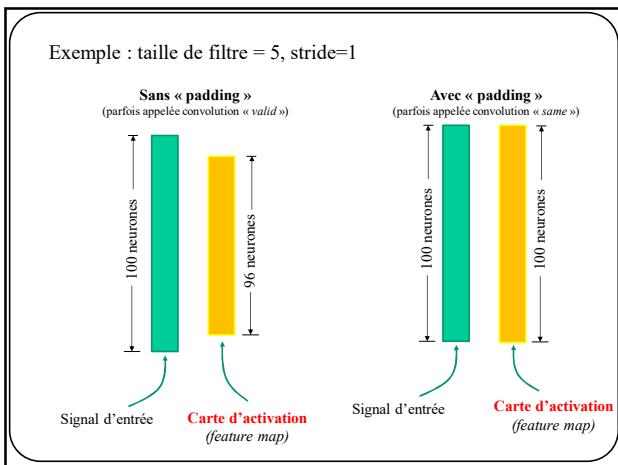
62

The diagram illustrates two types of convolutional layers:

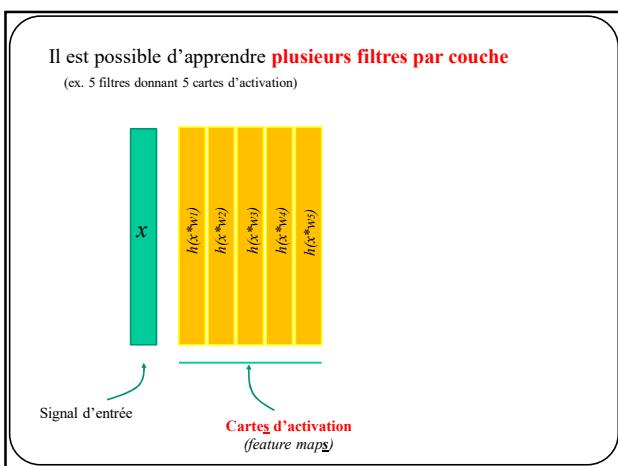
- Couche convolutionnelle sans « padding »**: This layer takes a single green input signal and produces a single yellow feature map.
- Couche convolutionnelle avec « padding »**: This layer also takes a single green input signal and produces a single yellow feature map, but it includes a red border around the yellow feature map, representing zero-padded input.

Both feature maps are labeled **Carte d'activation (feature map)**.

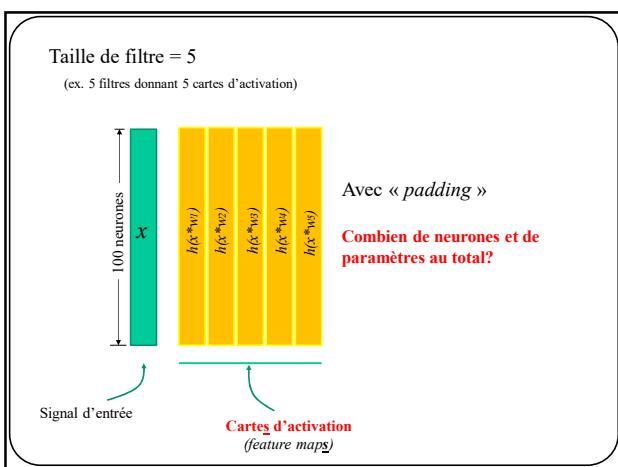
63



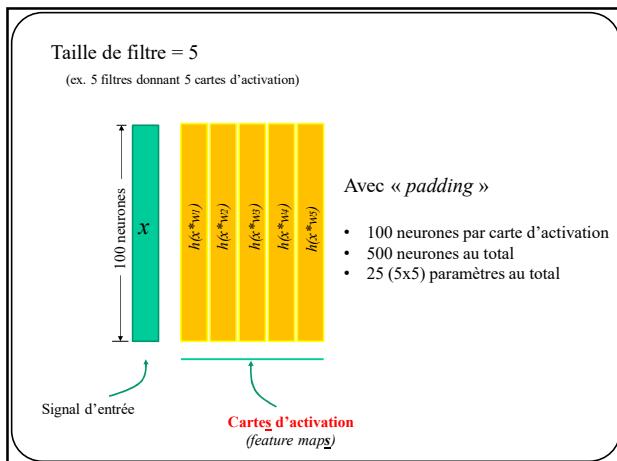
64



65



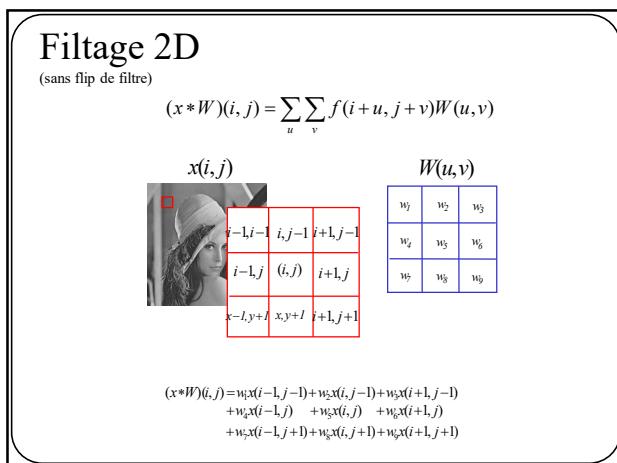
66



67

# Convolution et couche convolutionnelle **2D**

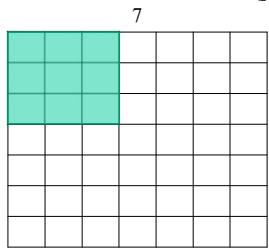
68



69

## Convolution 2D

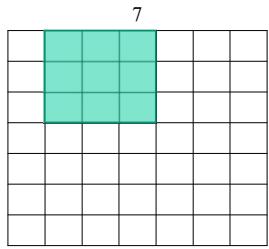
Filtre = 3x3  
Stride = 1



70

## Convolution 2D

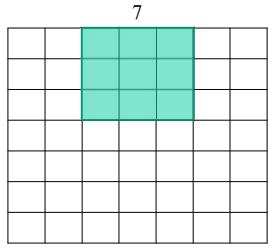
Filtre = 3x3  
Stride = 1



71

## Convolution 2D

Filtre = 3x3  
Stride = 1



72

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

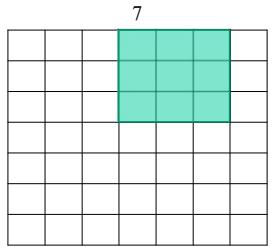
---

---

---

## Convolution 2D

Filtre = 3x3  
Stride = 1



73

---

---

---

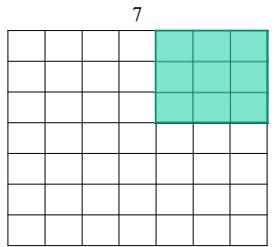
---

---

---

## Convolution 2D

Filtre = 3x3  
Stride = 1



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

74

---

---

---

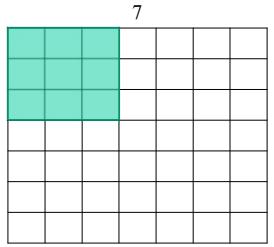
---

---

---

## Convolution 2D

Filtre = 3x3  
Stride = 2



75

---

---

---

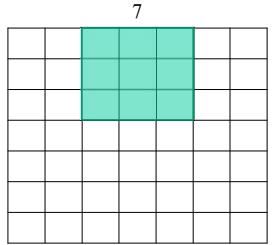
---

---

---

## Convolution 2D

Filtre = 3x3  
Stride = 2



76

---

---

---

---

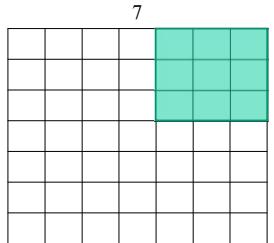
---

---

---

## Convolution 2D

Filtre = 3x3  
Stride = 2



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

77

---

---

---

---

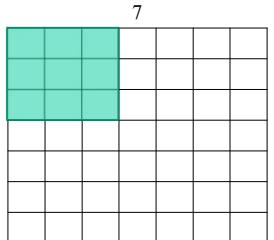
---

---

---

## Convolution 2D

Filtre = 3x3  
Stride = 3



78

---

---

---

---

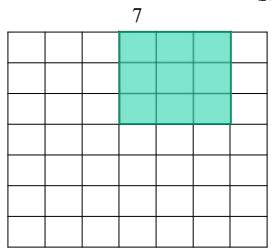
---

---

---

## Convolution 2D

Filtre = 3x3  
Stride = 3



79

---

---

---

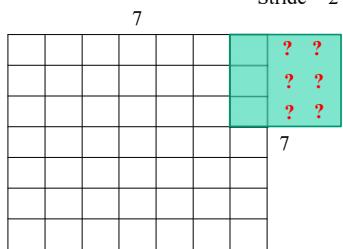
---

---

---

## Convolution 2D

Filtre = 3x3  
Stride = 2



Combinaison D-F-S invalide!

80

---

---

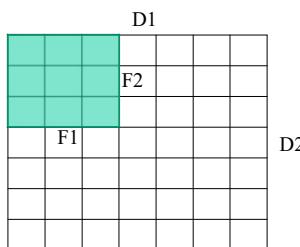
---

---

---

---

## Convolution 2D



Taille de la carte d'activation :

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

81

---

---

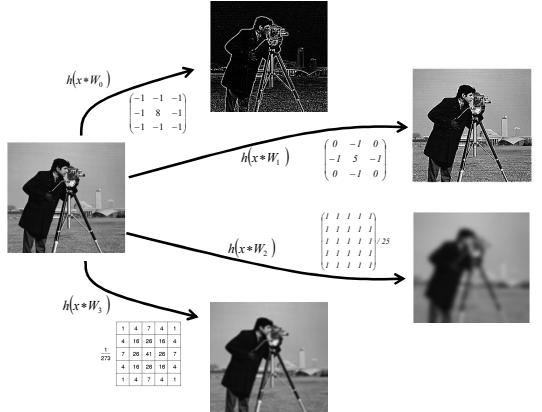
---

---

---

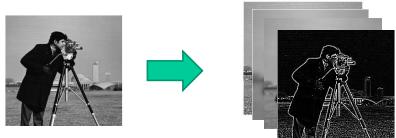
---

Différents filtres = différentes cartes d'activation



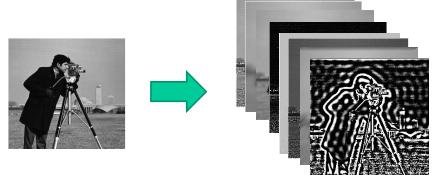
82

4 filtres = Couche convulsive avec 4 cartes d'activation

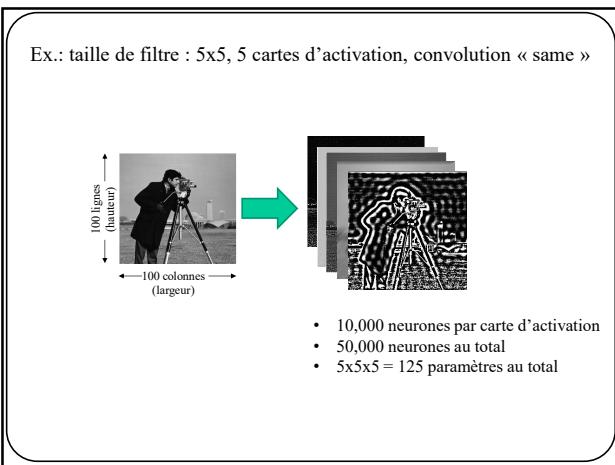


83

K filtres = Couche convulsive avec K cartes d'activation



84



85

---

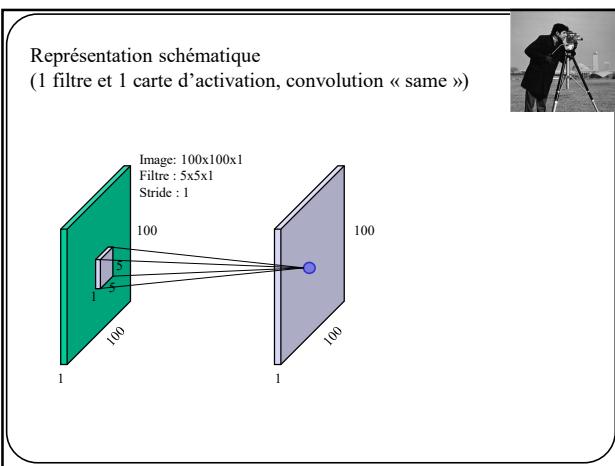
---

---

---

---

---



86

---

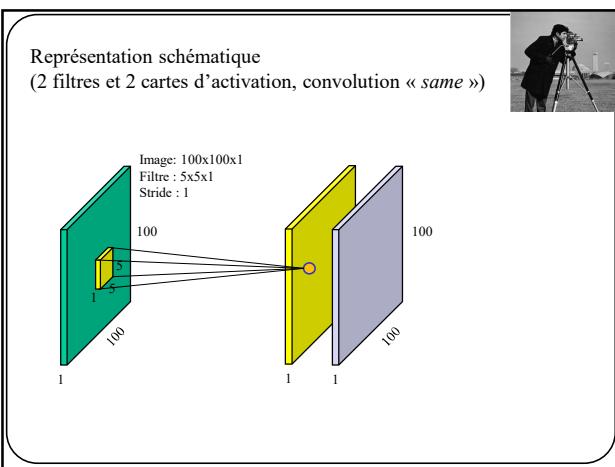
---

---

---

---

---



87

---

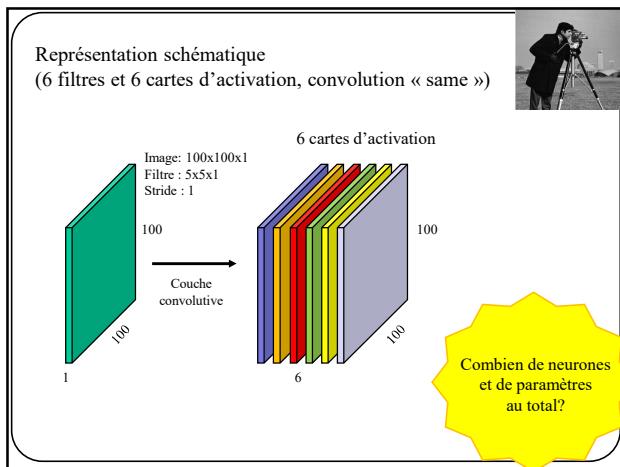
---

---

---

---

---



88

---



---



---



---



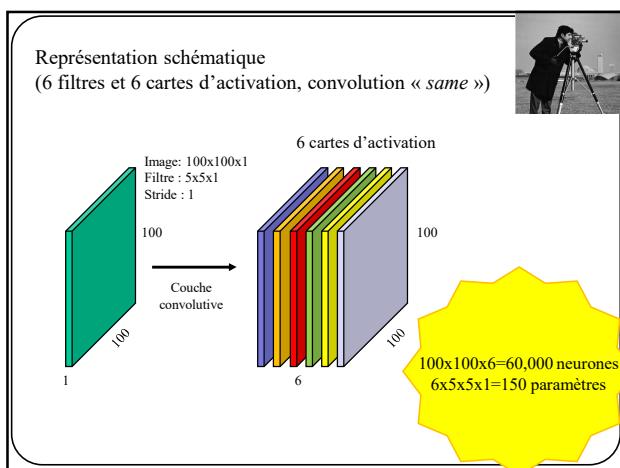
---



---



---



89

---



---



---



---



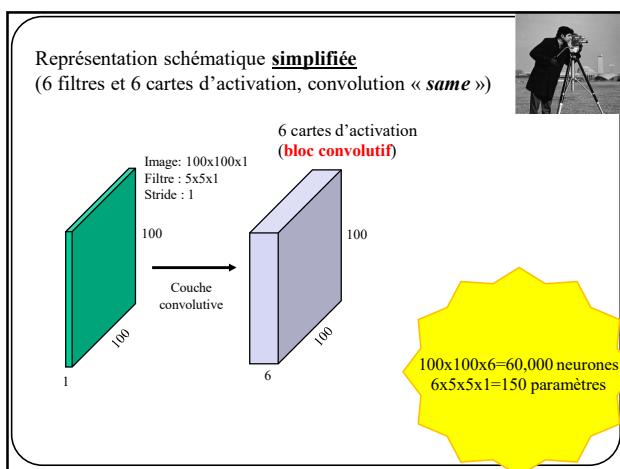
---



---



---



90

---



---



---



---



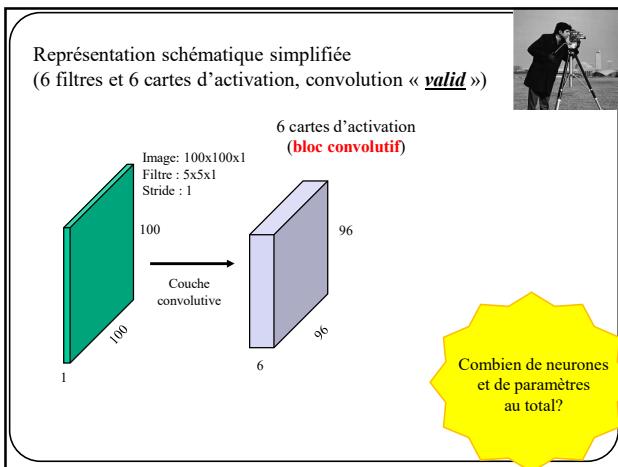
---



---



---



91

---



---



---



---



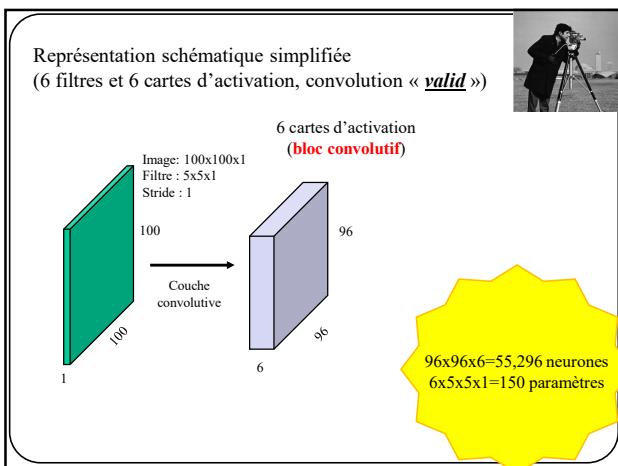
---



---



---



92

---



---



---



---



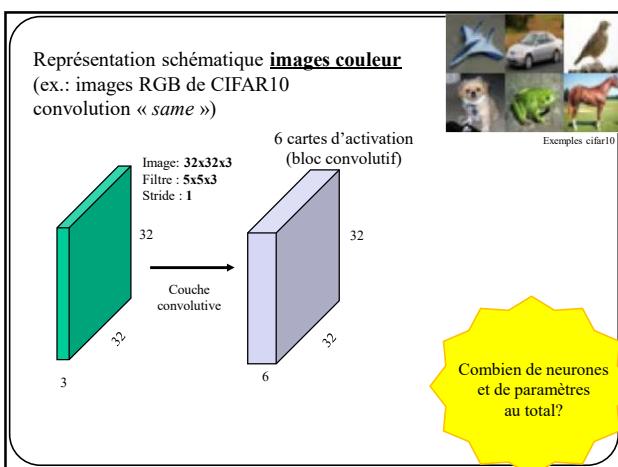
---



---



---



93

---



---



---



---



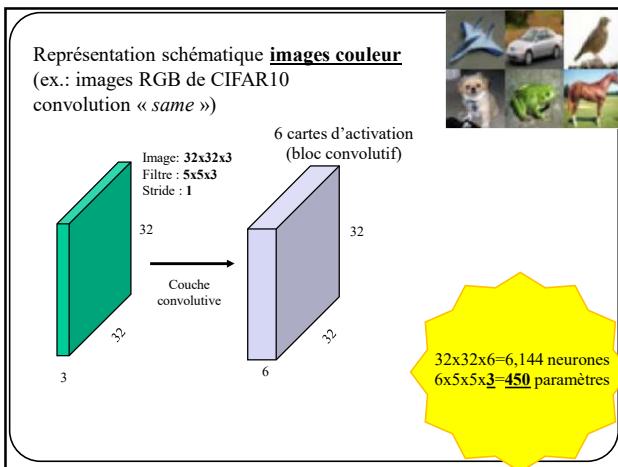
---



---



---



94

---



---



---



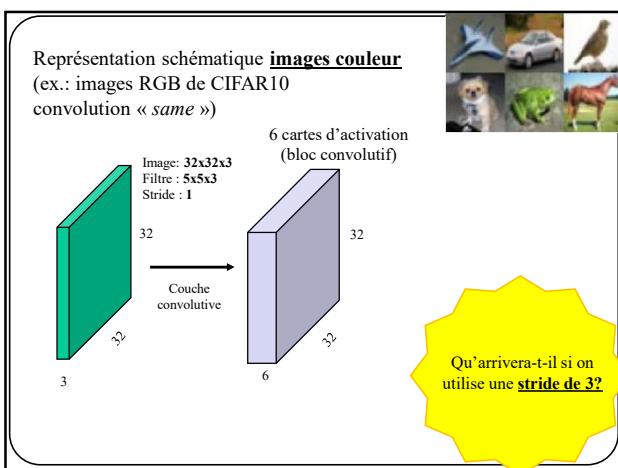
---



---



---



95

---



---



---



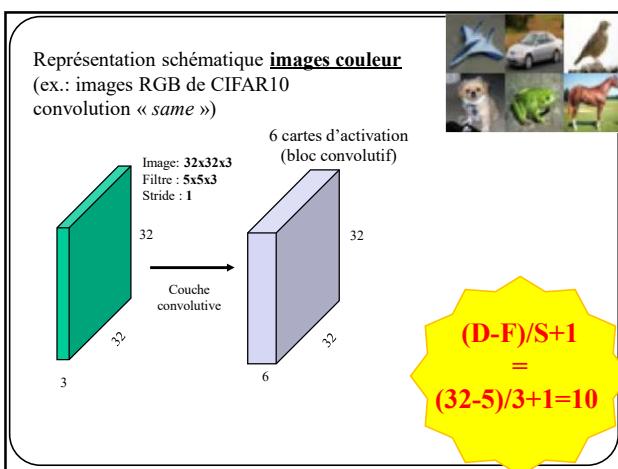
---



---



---



96

---



---



---



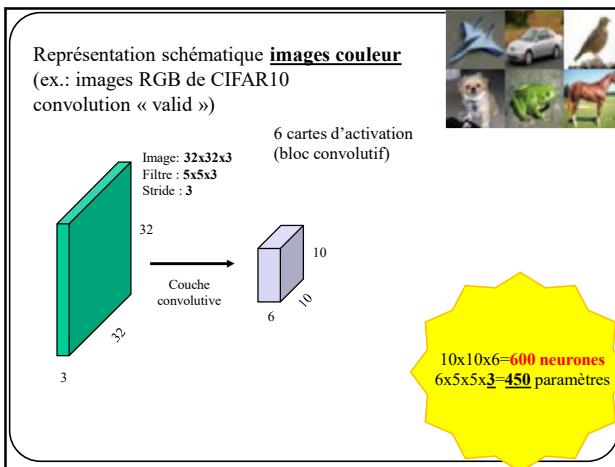
---



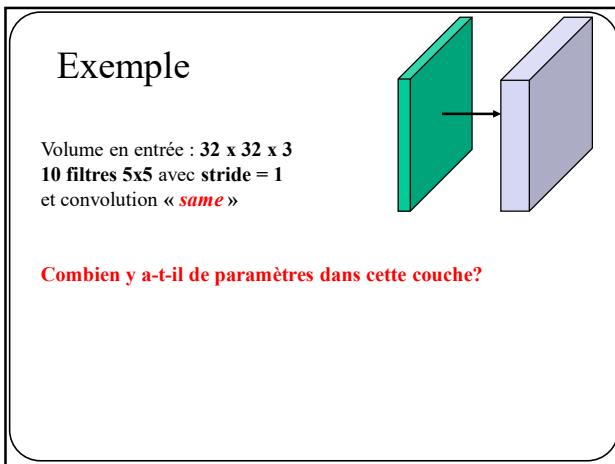
---



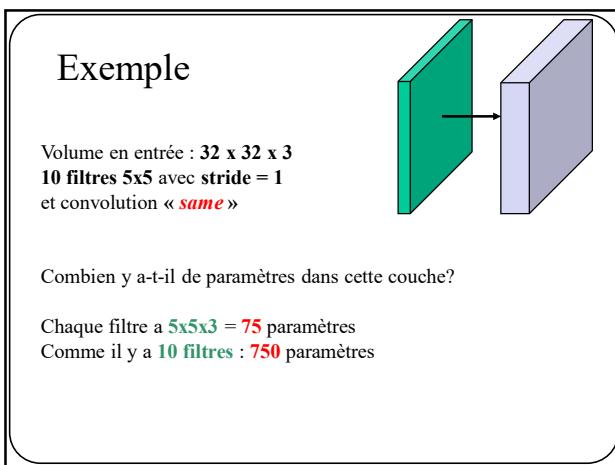
---



97



98



99

---



---



---



---



---



---



---



---



---



---



---



---



---



---



---



---



---



---



---



---



---



---



---



---



---



---



---



---



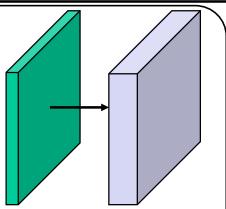
---



---

## Exemple

Volume en entrée :  $32 \times 32 \times 3$   
10 filtres 5x5 avec stride = 1  
et convolution « **same** ».



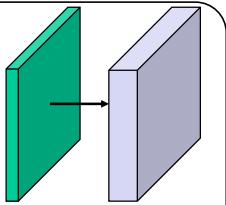
Combien y a-t-il de paramètres dans cette couche **si on ajoute un biais?**

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

100

## Exemple

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

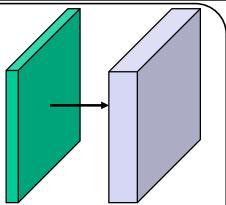


Combien de paramètres dans cette couche?

101

## Exemple

Volume en entrée :  $32 \times 32 \times 3$   
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

102

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

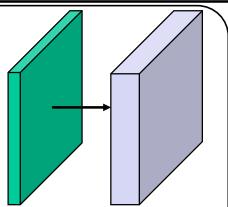
---

---

---

### Exemple

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



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

---

---

---

---

---

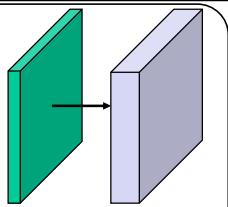
---

---

103

### Exemple

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



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

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

---

---

---

---

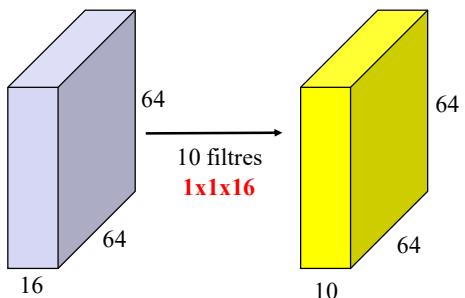
---

---

---

104

### Des filtres 1x1? Oui ça marche



---

---

---

---

---

---

---

105

## Exemple simple d'un filtre 1x1



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

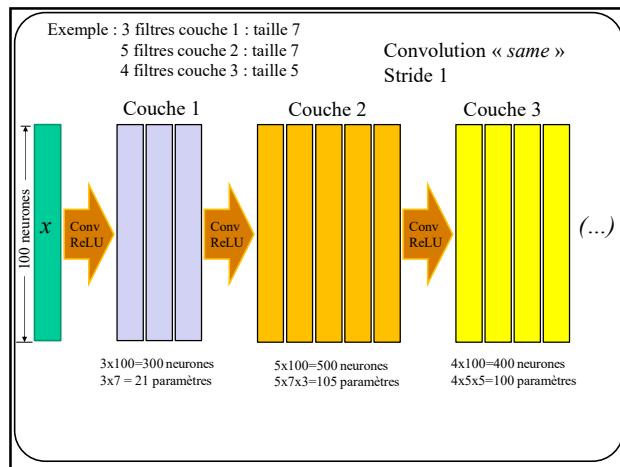


Filtre moyennant les canaux **rouge, vert, bleu** d'une image couleur.  
Résultat, une image en **niveau de gris**.

106

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

107



108

---

---

---

---

---

---

---

---

---

---

---

---

---

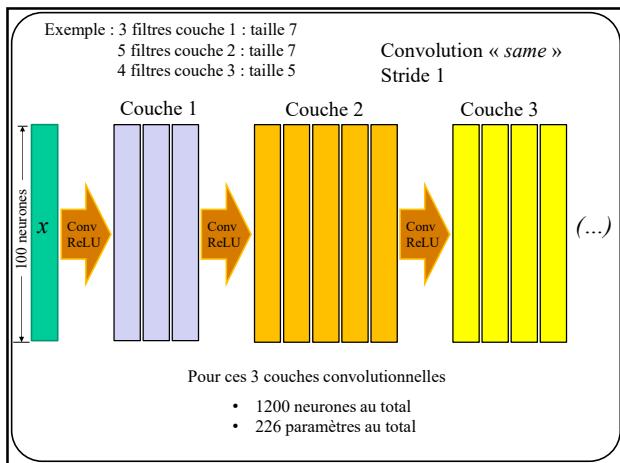
---

---

---

---

---



109

---

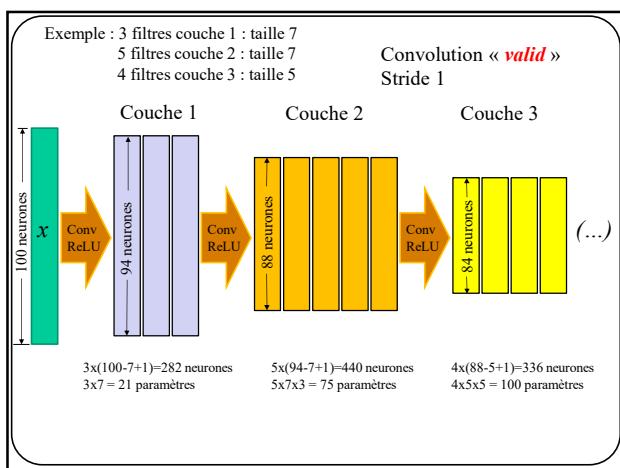
---

---

---

---

---



110

---

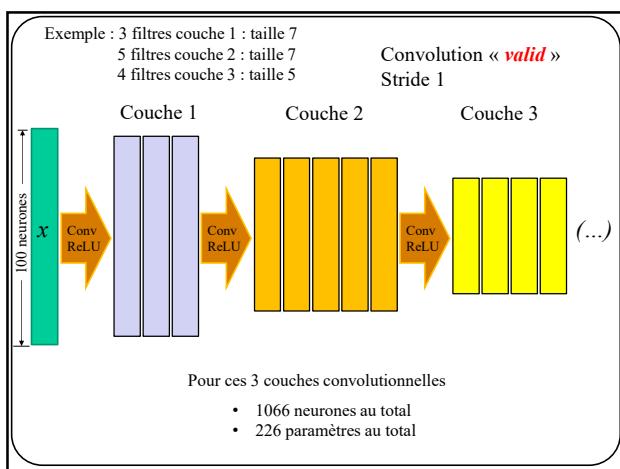
---

---

---

---

---



111

---

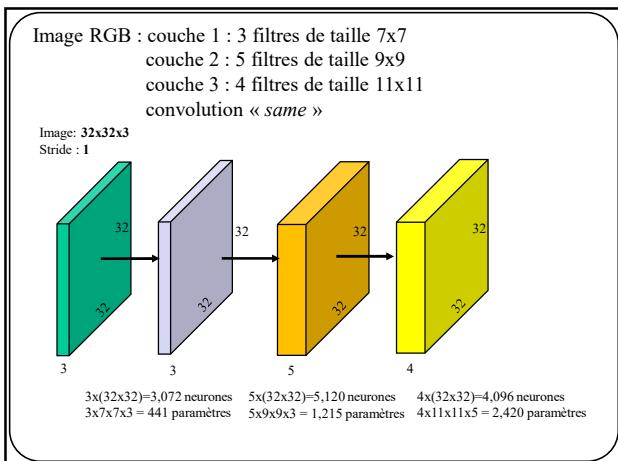
---

---

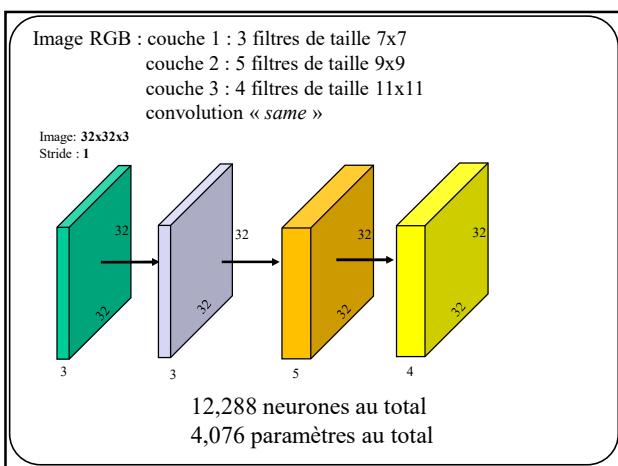
---

---

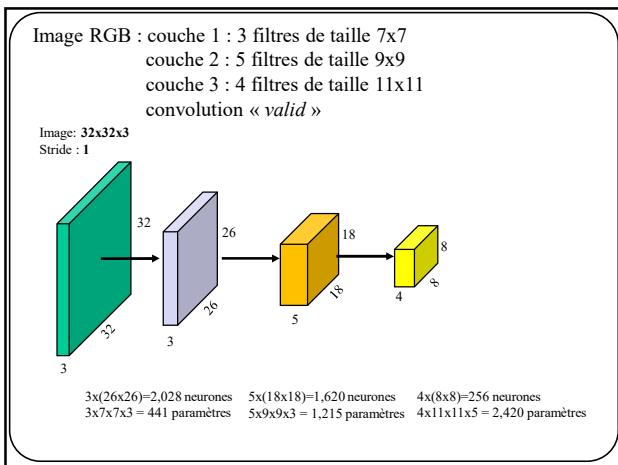
---



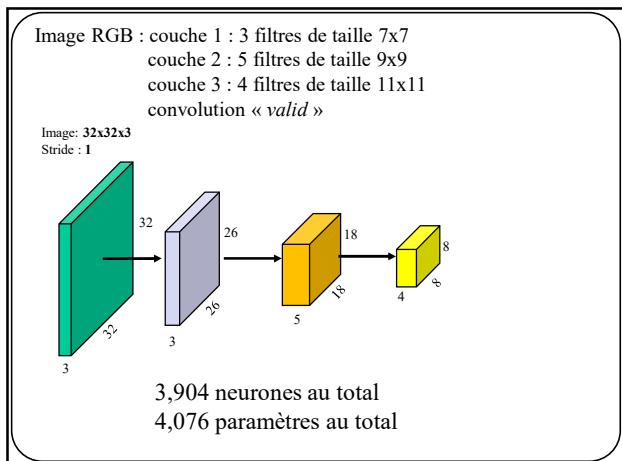
112



113



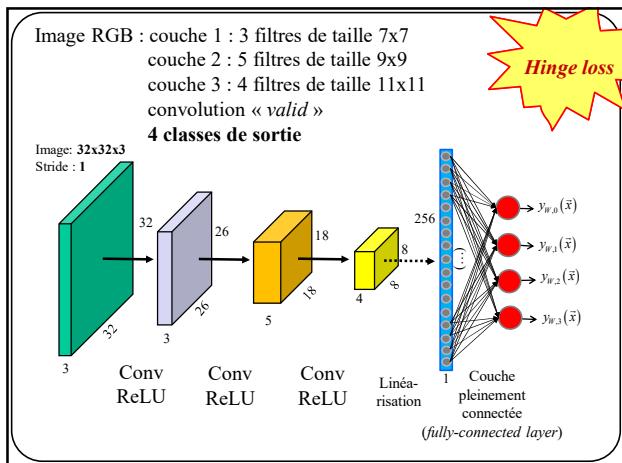
114



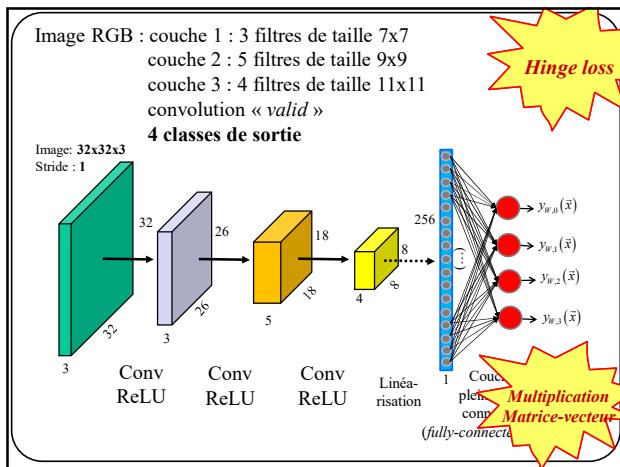
115

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

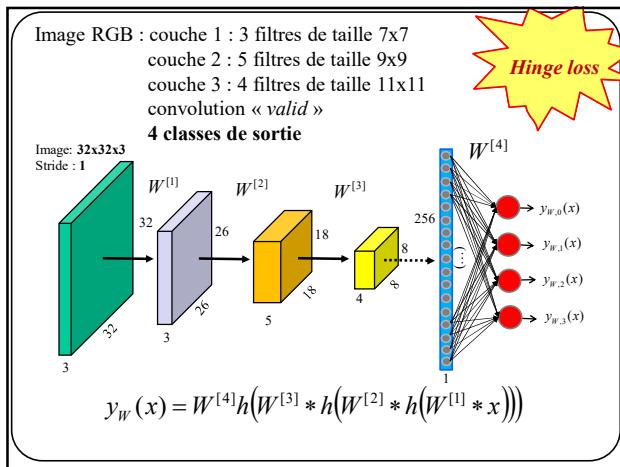
116



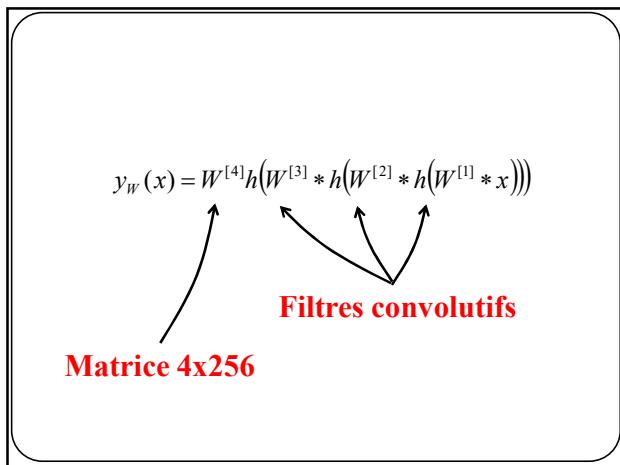
117



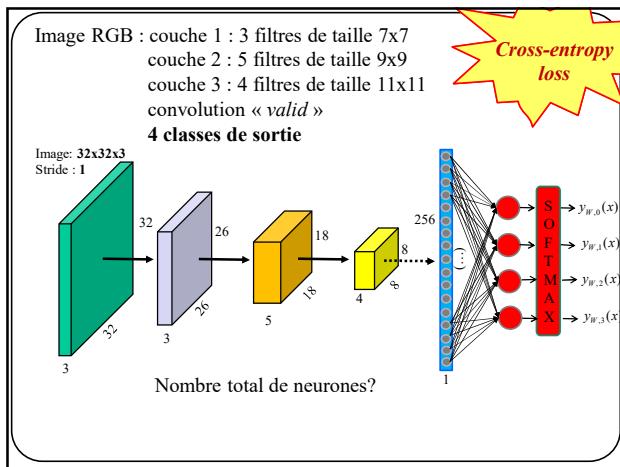
118



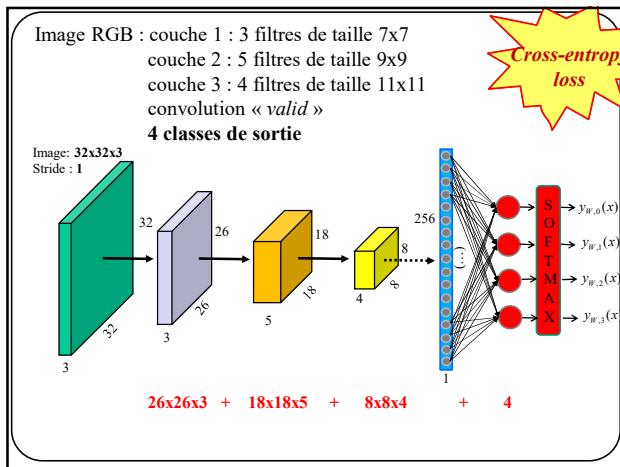
119



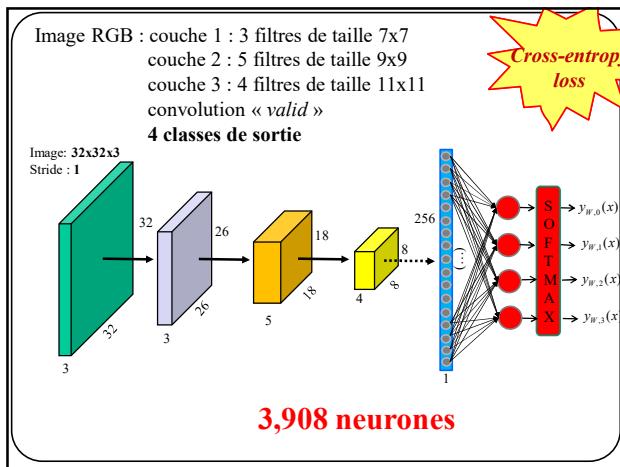
120



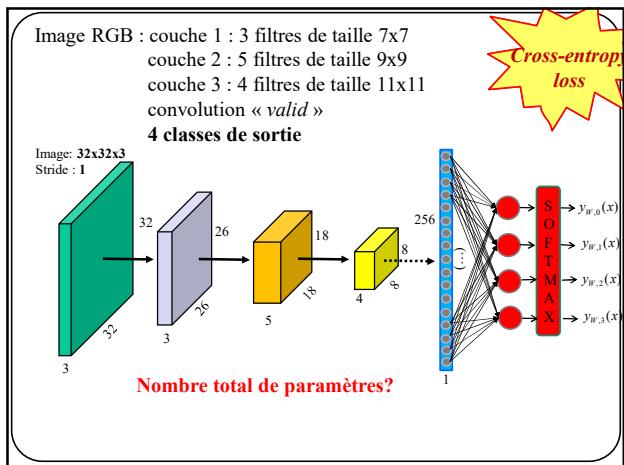
121



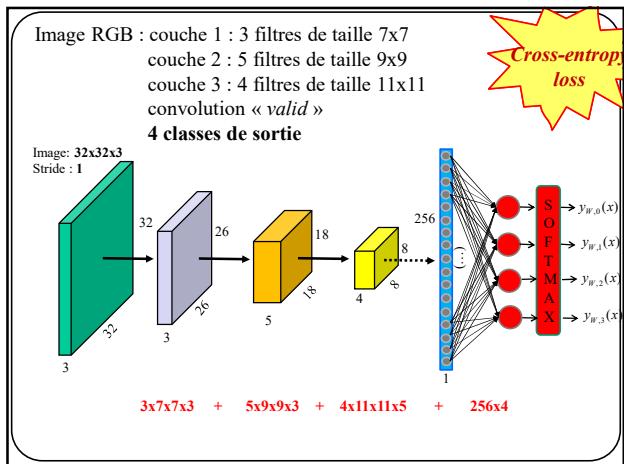
122



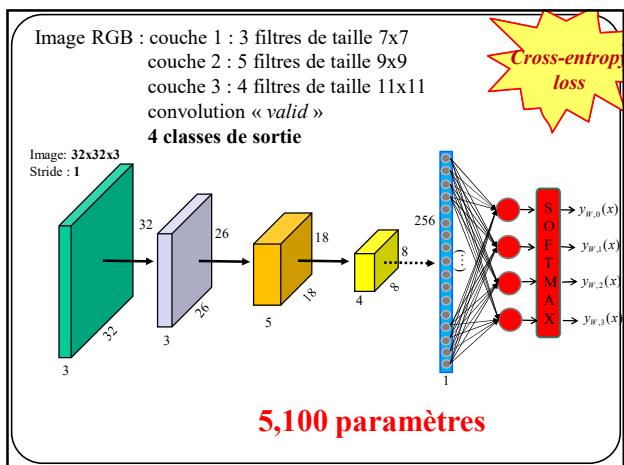
123



124



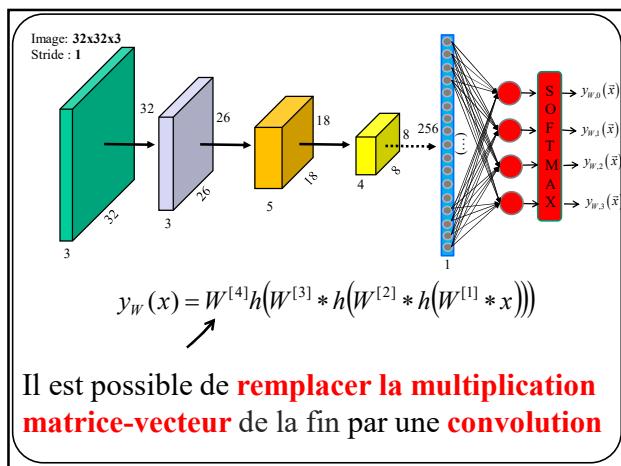
125



126

## Réseaux à convolution vs Réseaux **pleinement** convolutifs

127



128

## Exemple 1d

$$\begin{array}{ccccccc} & 9 & & 3 & & 7 & \\ \hline - & & & - & & & \\ | & & & | & & & | \\ 40 & 50 & 70 & 80 & 90 & 10 & 20 & 30 & 40 & * & .2 & -.3 & .4 & = & 21 & 21 & 26 & -7 & 23 & 8 & 11 \end{array}$$

129

**Exemple 1d**  
(convolution « valid »)

$$\begin{array}{ccccccc} & 9 & & 5 & & 5 & \\ \xleftarrow{\hspace{1cm}} & & \xleftarrow{\hspace{1cm}} & & \xleftarrow{\hspace{1cm}} & & \\ \boxed{40} & \boxed{50} & \boxed{70} & \boxed{80} & \boxed{90} & \boxed{10} & \boxed{20} \end{array} * \begin{bmatrix} .2 & -.3 & .4 & -.5 & .6 \end{bmatrix} = \begin{bmatrix} 35 & -18 & 33 & 1 & 32 \end{bmatrix}$$

130

**Exemple 1d**  
(convolution « valid »)

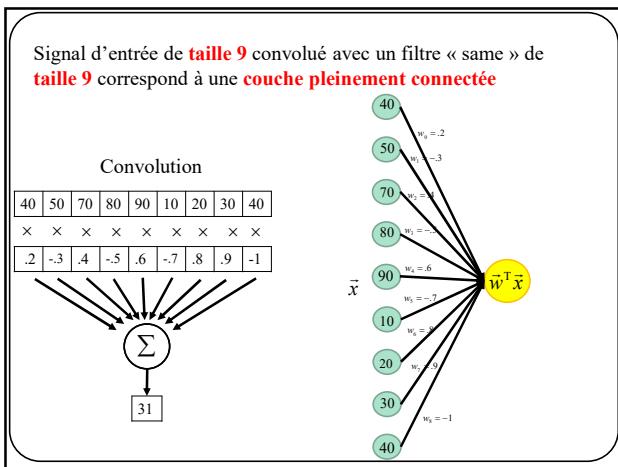
$$\begin{array}{ccccccc} & 9 & & 7 & & 3 & \\ \xleftarrow{\hspace{1cm}} & & \xleftarrow{\hspace{1cm}} & & \xleftarrow{\hspace{1cm}} & & \\ \boxed{40} & \boxed{50} & \boxed{70} & \boxed{80} & \boxed{90} & \boxed{10} & \boxed{20} \end{array} * \begin{bmatrix} .2 & -.3 & .4 & -.5 & .6 & -.7 & .8 \end{bmatrix} = \begin{bmatrix} 44 & -8 & 44 \end{bmatrix}$$

131

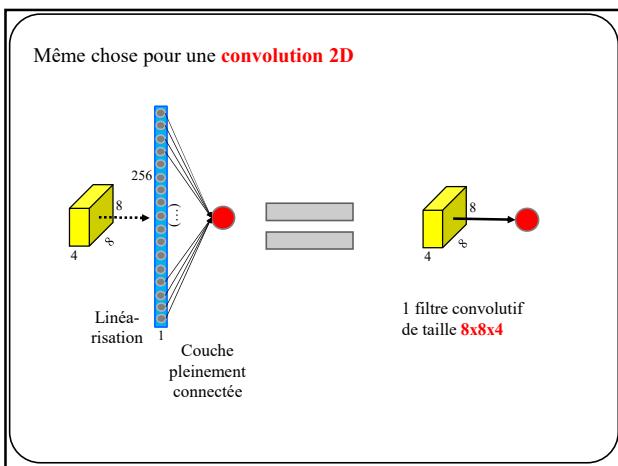
Taille filtre = nb de neurones couche précédente

$$\begin{array}{ccccc} & 9 & & 9 & & 1 \\ \xleftarrow{\hspace{1cm}} & & \xleftarrow{\hspace{1cm}} & & \xleftarrow{\hspace{1cm}} & \\ \boxed{40} & \boxed{50} & \boxed{70} & \boxed{80} & \boxed{90} & \boxed{10} \end{array} * \begin{bmatrix} .2 & -.3 & .4 & -.5 & .6 & -.7 & .8 & .9 & -.1 \end{bmatrix} = \begin{bmatrix} 31 \end{bmatrix}$$

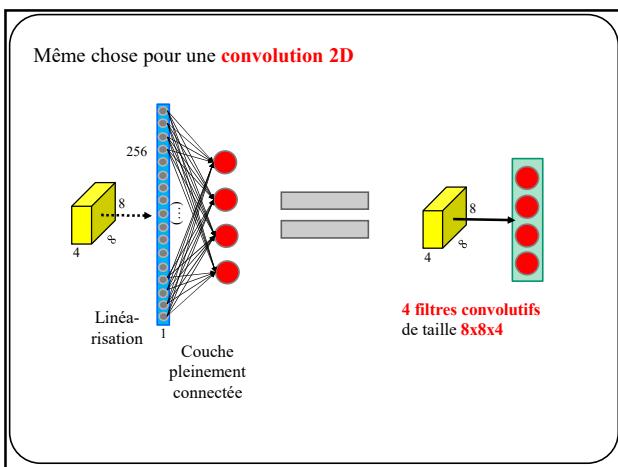
132



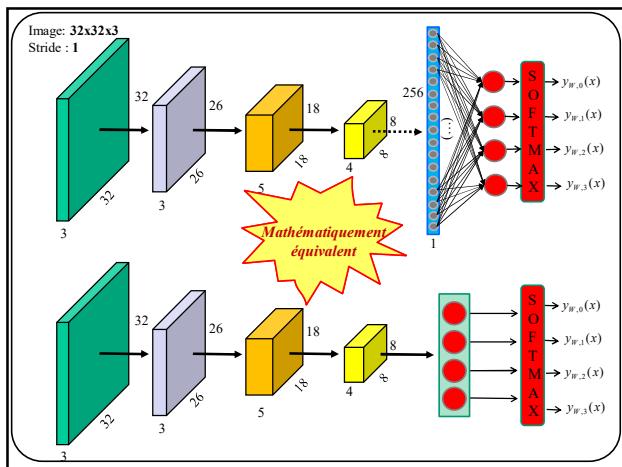
133



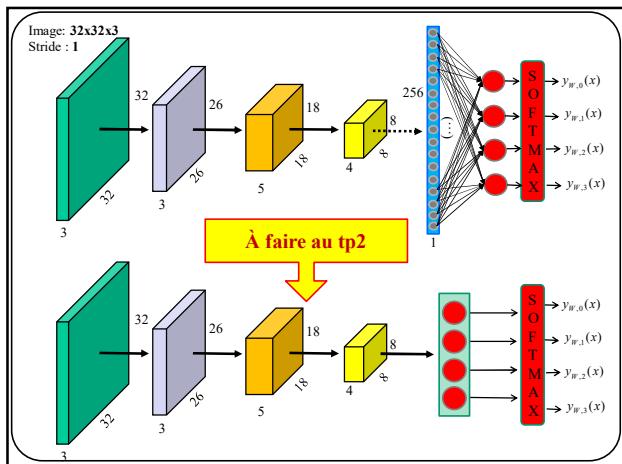
134



135



136



137

## Configurations équivalentes

couche 1 : 3 filtres de taille 7x7

couche 1 : 5 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 1 : 5 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?**

138

# Pooling

139

---

---

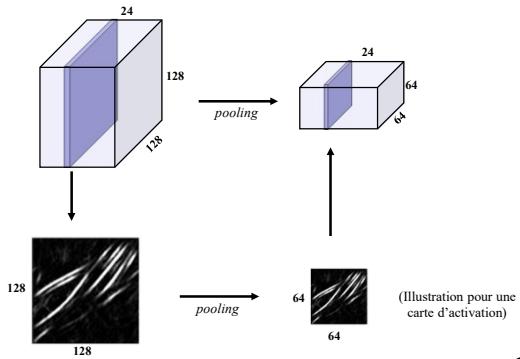
---

---

---

---

Réduction de la taille des cartes d'activation



140

---

---

---

---

---

---

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

Max pool par filtre « valid » 2x2 avec stride =2

141

---

---

---

---

---

---

## Mean 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

Moyenne par filtre  
« valid » 2x2 avec  
stride =2

4	5	3	4
0	1	5	1
-1	8	8	7
5	4	4	1

142

## 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

Max pooling 2x2  
avec stride =1



143

## Max pooling

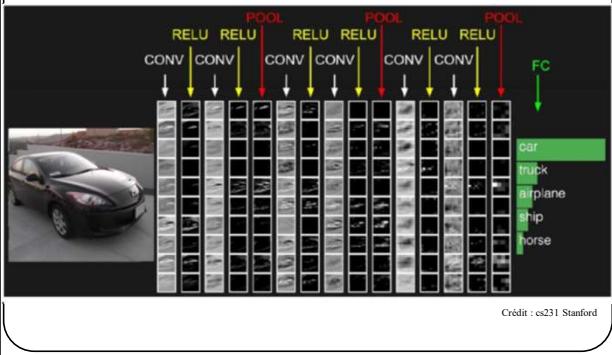
1	2	4	4	9	3	1	2
6	7	8	4	-3	-3	6	3
9	-9	8	-4	5	5	3	0
8	-8	9	-9	5	5	0	1
0	0	1	2	7	9	7	8
-1	-3	3	6	8	8	7	6
9	9	8	2	1	5	-1	-1
1	1	-2	8	3	7	4	-2

Max pooling 3x3  
avec stride =2



144

## Illustration d'un CNN complet



145

## Multiplication matricielle parcimonieuse

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

147

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

$$\begin{array}{c}
 \text{Entrée} \\
 \begin{array}{|c|c|c|c|} \hline
 X_0 & X_1 & X_2 & X_3 \\ \hline
 X_4 & X_5 & X_6 & X_7 \\ \hline
 X_8 & X_9 & X_{10} & X_{11} \\ \hline
 X_{12} & X_{13} & X_{14} & X_{15} \\ \hline
 \end{array}
 \end{array}
 *
 \begin{array}{c}
 \text{Filtre} \\
 \begin{array}{|c|c|c|} \hline
 W_0 & W_1 & W_2 \\ \hline
 W_3 & W_4 & W_5 \\ \hline
 W_6 & W_7 & W_8 \\ \hline
 \end{array}
 \end{array}
 =
 \begin{array}{|c|c|} \hline
 Y_1 & Y_2 \\ \hline
 Y_3 & Y_4 \\ \hline
 \end{array}$$

148

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

\*

Filtre		
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** en **linéarisant** le filtre et en « **matriçant** » l'entrée

149

## 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

\*

Y0	Y1
Y2	Y3

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

150

## 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	Y1
Y2	Y3

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

151

## 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
Y2	Y3

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

152

## 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	Y3

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

153

## Autrement dit...

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

Y0
----

X0
X1
X2
X4
X5
X6
X8
X9
X10

154

Autrement dit...

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

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

Y0	Y1
----	----

155

Autrement dit...

X0	X1	X2	X3
W0	W1	W2	X7
W3	W4	W5	X11
W6	W7	W8	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	X12
X9	X10	X12	X13
X10	X11	X13	X15

Y0	Y1
Y2	

156

Autrement dit...

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

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

Y0	Y1
Y2	Y3

157

Convolution « valid » en linéarisant le filtre et en « matriçant » l'entrée

$$\begin{array}{ccccccccc|c} & & & & & & & & & \\ \text{W0} & \text{W1} & \text{W2} & \text{W3} & \text{W4} & \text{W5} & \text{W6} & \text{W7} & \text{W8} & \times \end{array} \quad = \quad \begin{array}{cccc} \text{X0} & \text{X1} & \text{X4} & \text{X5} \\ \text{X1} & \text{X2} & \text{X5} & \text{X6} \\ \text{X2} & \text{X3} & \text{X6} & \text{X7} \\ \text{X4} & \text{X5} & \text{X8} & \text{X9} \\ \text{X5} & \text{X6} & \text{X9} & \text{X10} \\ \text{X6} & \text{X7} & \text{X10} & \text{X11} \\ \text{X8} & \text{X9} & \text{X11} & \text{X13} \\ \text{X9} & \text{X10} & \text{X12} & \text{X14} \\ \text{X10} & \text{X11} & \text{X13} & \text{X15} \end{array}$$

158

Autre exemple  
conv « valid », mini-batch de 2 entrées

$$\begin{array}{|c|c|c|c|} \hline
 X_0 & X_1 & X_2 & X_3 \\ \hline
 X_4 & X_5 & X_6 & X_7 \\ \hline
 X_8 & X_9 & X_{10} & X_{11} \\ \hline
 X_{12} & X_{13} & X_{14} & X_{15} \\ \hline
 \end{array}
 *
 \begin{array}{|c|c|c|} \hline
 \text{Filtre} & & \\ \hline
 W_0 & W_1 & W_2 \\ \hline
 W_3 & W_4 & W_5 \\ \hline
 W_6 & W_7 & W_8 \\ \hline
 \end{array}
 =
 \begin{array}{|c|c|} \hline
 Y_0 & Y_1 \\ \hline
 Y_2 & Y_3 \\ \hline
 Y_4 & Y_5 \\ \hline
 Y_6 & Y_7 \\ \hline
 \end{array}$$

159

Autre exemple  
conv « valid », mini-batch de 2 entrées

	X16	X17	X20	X21	
w0	X0	X1	X4	X5	
w1	X1	X2	X5	X6	
w2	X2	X3	X6	X7	
w3	X4	X5	X8	X9	
w4	X5	X6	X9	X10	
w5	X6	X7	X10	X11	
w6	X8	X9	X11	X13	
w7	X9	X10	X12	X14	
w8	X10	X11	X13	X15	
x					
	Y0	Y1	Y2	Y3	
	Y4	Y5	Y6	Y7	

160

### Autre exemple

conv « valid », une entrée, deux filtres

Entrée

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

Filtre

W0	W1	W2
W3	W4	W5
W6	W7	W8
W9	W10	W11
W12	W13	W14
W15	W16	W17

Y0	Y1
Y2	Y3
Y4	Y5
Y6	Y7

161

### Autre exemple

conv « valid », une entrée, deux filtres

$$\begin{array}{|c|c|c|c|} \hline X0 & X1 & X4 & X5 \\ \hline X1 & X2 & X5 & X6 \\ \hline X2 & X3 & X6 & X7 \\ \hline X4 & X5 & X8 & X9 \\ \hline X5 & X6 & X9 & X10 \\ \hline X6 & X7 & X10 & X11 \\ \hline X8 & X9 & X11 & X13 \\ \hline X9 & X10 & X12 & X14 \\ \hline X10 & X11 & X13 & X15 \\ \hline \end{array} \times \begin{array}{|c|c|c|c|} \hline W0 & W1 & W2 & W3 \\ \hline W3 & W4 & W5 & W6 \\ \hline W6 & W7 & W8 & W9 \\ \hline W9 & W10 & W11 & W12 \\ \hline W12 & W13 & W14 & W15 \\ \hline W15 & W16 & W17 & W18 \\ \hline \end{array} = \begin{array}{|c|c|c|c|} \hline Y0 & Y1 & Y2 & Y3 \\ \hline Y4 & Y5 & Y6 & Y7 \\ \hline \end{array}$$

162

### Autre exemple

conv « valid », une entrée avec deux canaux, un filtre

Entrée

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

Filtre

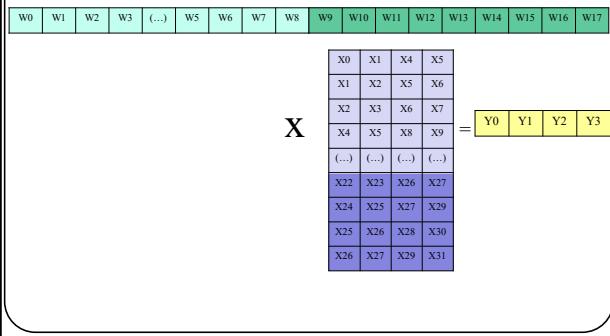
W0	W1	W2
W3	W4	W5
W6	W7	W8
W9	W10	W11
W12	W13	W14
W15	W16	W17

Y0	Y1
Y2	Y3

163

### Autre exemple

conv « valid », une entrée avec deux canaux, un filtre

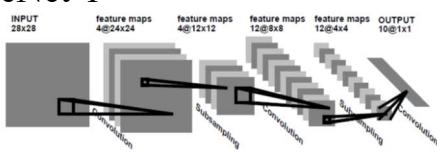


164

### Exemples d'architectures connues

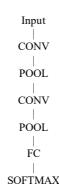
165

### LeNet-1



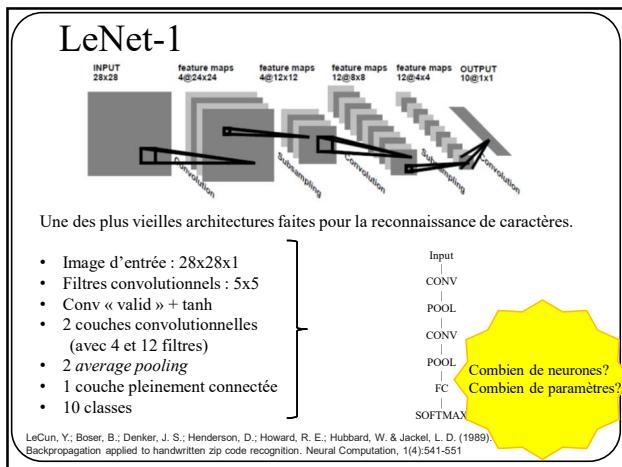
Une des plus vieilles architectures faites pour la reconnaissance de caractères.

- Image d'entrée : 28x28x1
- Filtres convolutionnels : 5x5
- Conv « valid » + tanh
- 2 couches convolutionnelles (avec 4 et 12 filtres)
- 2 *average pooling*
- 1 couche pleinement connectée
- 10 classes

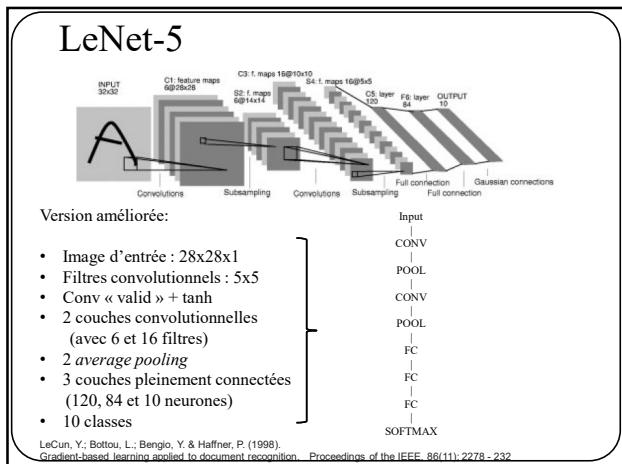


LeCun, Y.; Boser, B.; Denker, J. S.; Henderson, D.; Howard, R. E.; Hubbard, W. & Jackel, L. D. (1989). Backpropagation applied to handwritten zip code recognition. *Neural Computation*, 1(4):541-551

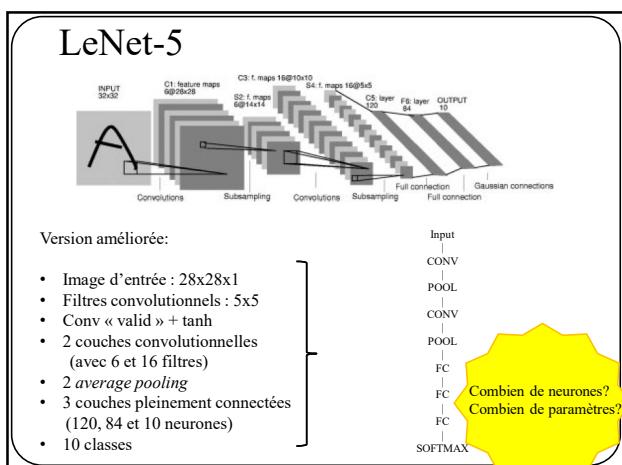
166



167



168



169

## Classification d'images

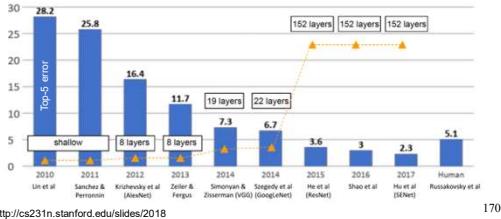
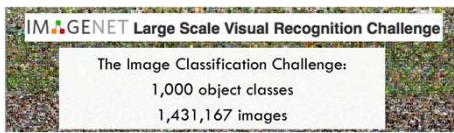


Image: <http://cs231n.stanford.edu/slides/2018>

170

## Classification d'images

AlexNet [Krizhevsky et al. 2012]

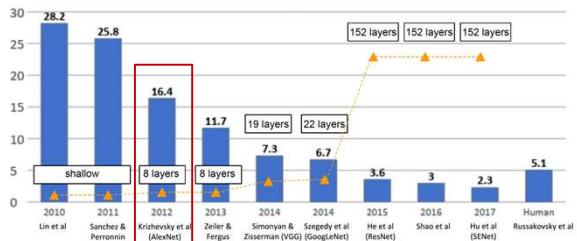
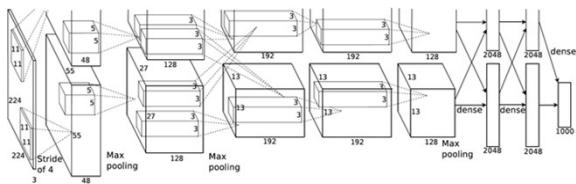


Image: <http://cs231n.stanford.edu/slides/2018>

171

## AlexNet



- Premier CNN à bien performer sur ImageNet (amélioration de 10% par rapport aux autres)
- Utilisation de techniques aujourd'hui fréquemment utilisées: **ReLU**, **data augmentation** and **dropout**
- Utilisation de **GPUs** (2 dans leur cas)
- Point de départ de la **révolution du “deep learning”** en vision par ordinateur

Image: Krizhevsky et al. "Imagenet classification with deep convolutional neural networks," NIPS 2012.

172

## AlexNet

Architecture:

CONV1  
MAX POOL1

NORM1

CONV2  
MAX POOL2

NORM2

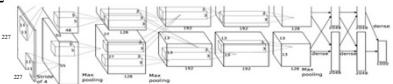
CONV3  
CONV4  
CONV5

Max POOL3

FC6

FC7

FC8



Entrée : image RGB: 227x227x3

Couche 1 (CONV1): 96 filtres de taille 11x11 avec stride de 4 et conv "valid"

Quelle est la taille des cartes d'activation?

Réponse:  $(227-11)/4+1 = 55$

173

## AlexNet

Architecture:

CONV1  
MAX POOL1

NORM1

CONV2  
MAX POOL2

NORM2

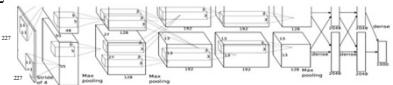
CONV3  
CONV4  
CONV5

Max POOL3

FC6

FC7

FC8



Entrée : image RGB: 227x227x3

Couche 1 (CONV1): 96 filtres de taille 11x11 avec stride de 4 et conv "valid"

Cartes d'activation : 96 x 55 x 55

Q: Quel est le nombre de paramètres?

174

## AlexNet

Architecture:

CONV1  
MAX POOL1

NORM1

CONV2  
MAX POOL2

NORM2

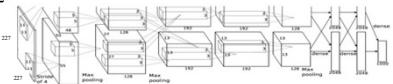
CONV3  
CONV4  
CONV5

Max POOL3

FC6

FC7

FC8



Entrée : image RGB: 227x227x3

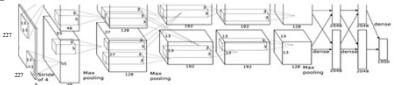
Couche 1 (CONV1): 96 filtres de taille 11x11 avec stride de 4 et conv "valid"

Cartes d'activation : 96 x 55 x 55

Paramètres :  $11 \times 11 \times 96 \times 3 = 34,848$

175

## AlexNet



ENTRÉE : 227x227x3

CONV1: 96 x 55 x 55

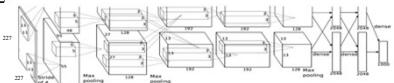
Couche 2 MaxPool : 3x3 stride stride 2

Quelle est la taille des cartes d'activation?

Réponse:  $(55-3)/2+1 = 27$

176

## AlexNet



ENTRÉE : 227x227x3

CONV1: 96 x 55 x 55

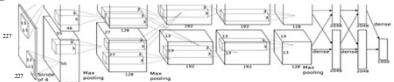
Couche 2 MaxPool : 3x3 stride stride 2  
27 x 27 x 96

Combien y a-t-il de paramètres?

Réponse: 0!

177

## AlexNet



ENTRÉE : 227x227x3

CONV1: 55 x 55 x 96

MAX POOL1: 27 x 27 x 96

...

178

---

---

---

---

---

---

---

---

---

---

---

---

---

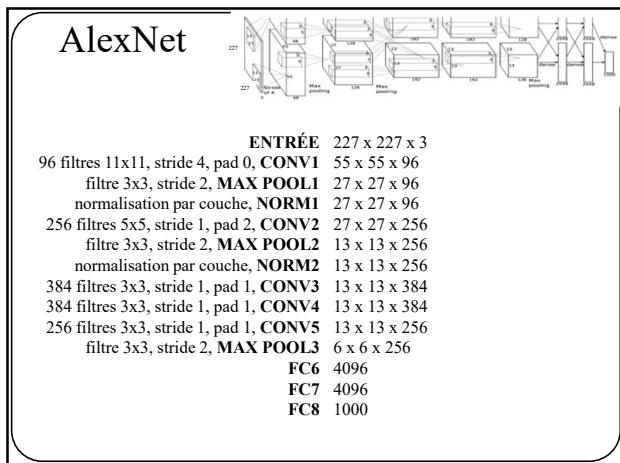
---

---

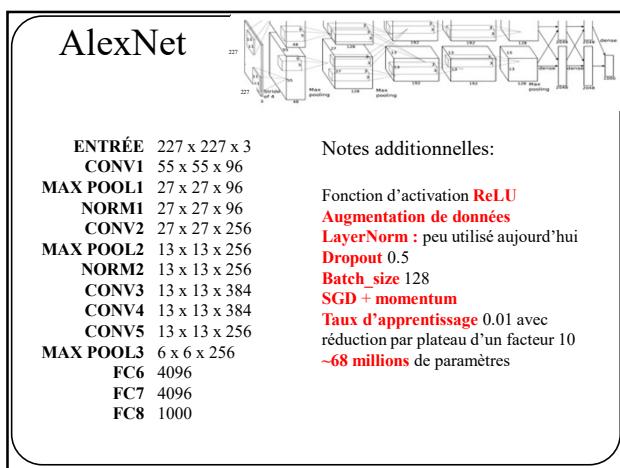
---

---

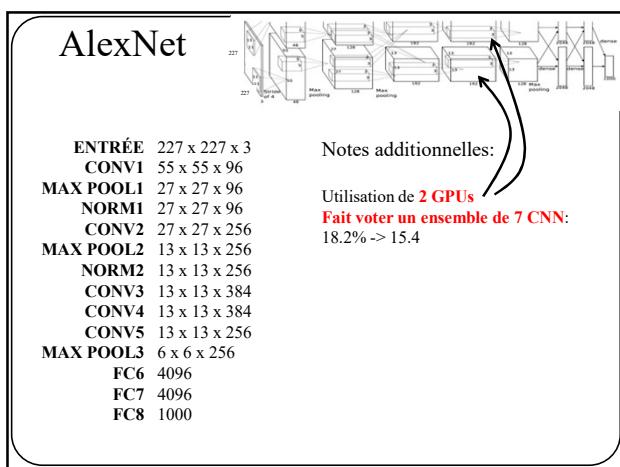
---



179



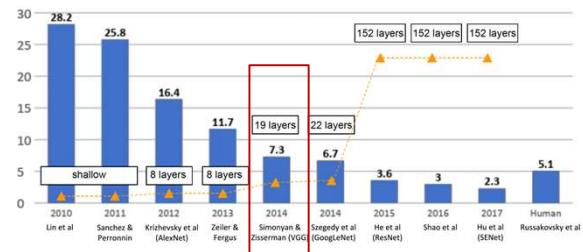
180



181

## Classification d'images

VGGNet [Simonyan and Zisserman, 2014]

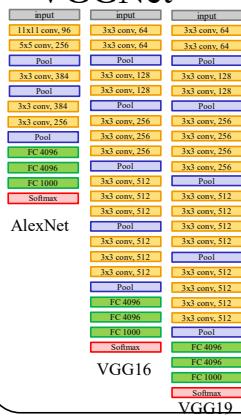


Karen Simonyan, Andrew Zisserman "Very Deep Convolutional Networks for Large-Scale Image Recognition", ICLR 2015

Image: <http://cs231n.stanford.edu/slides/2018>

182

## VGGNet



Ce qui caractérise VGGNet par rapport à ses prédecesseurs:

- Uniquement des **filtres 3x3, stride 1, pad 1**
- **Plus profond**

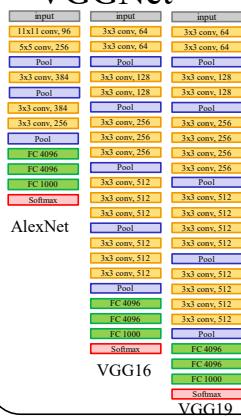
AlexNet : 8 couches

VGGNet : 16 ou 19 couches

7.3% d'erreur contre 11.7% pour ZFNet

183

## VGGNet



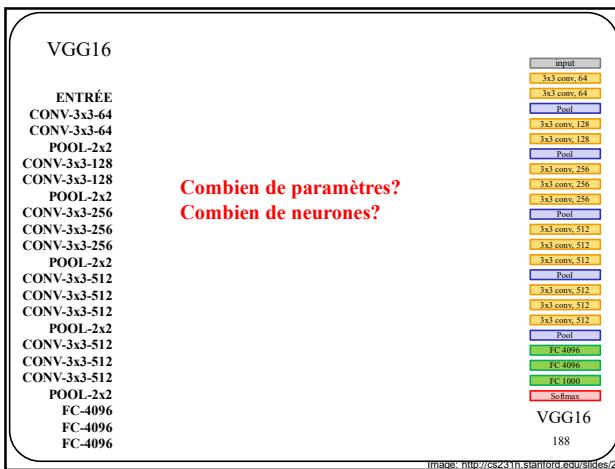
Pourquoi utiliser des filtres 3x3?

**Indice1** : quel est le champ récepteur (*receptive field*) d'une série de 3 couches 3x3?

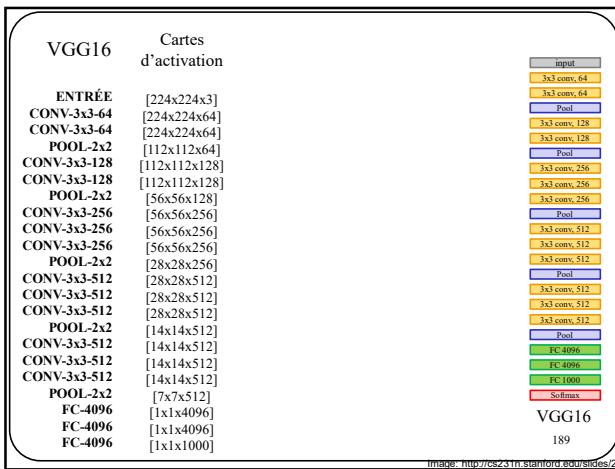
**Indice2** : combien de paramètres pour ces 3 couches?

184

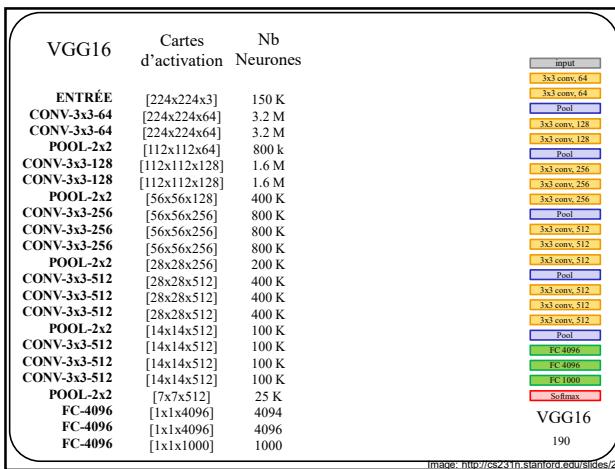




188



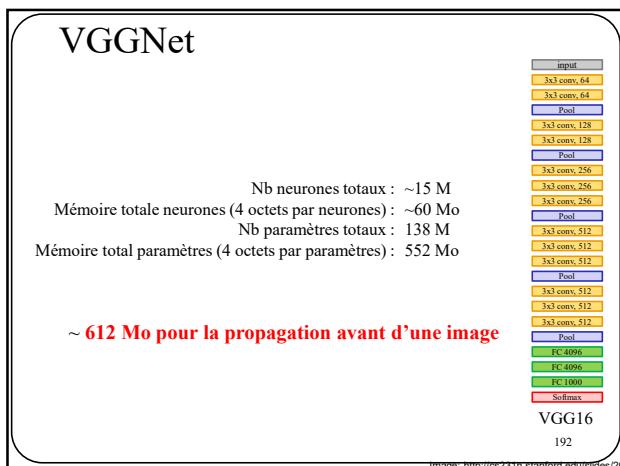
189



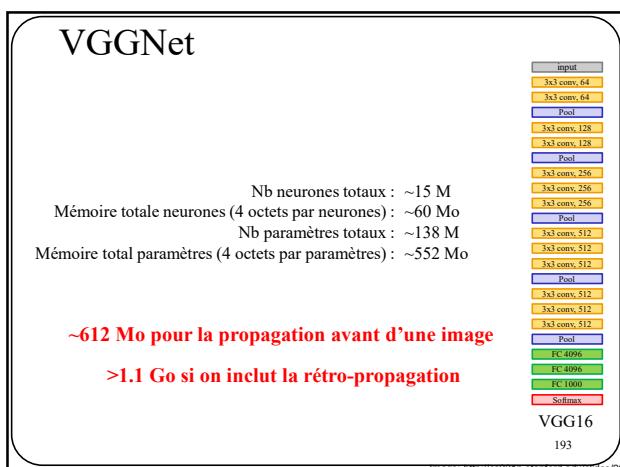
190

VGG16	Cartes d'activation	Nb Neurones	Nb Paramètres	
<b>ENTRÉE</b>	[224x224x3]	150 K	0	
CONV-3x3-64	[224x224x64]	3.2 M	$(3*3*3)*64 = 1,728$	input 3x3 conv, 64 3x3 conv, 64 Pool
CONV-3x3-64	[224x224x64]	3.2 M	$(3*3*64)*64 = 36,864$	3x3 conv, 128 3x3 conv, 128 Pool
POOL-2x2	[112x112x64]	800 k	0	3x3 conv, 256 3x3 conv, 256 3x3 conv, 256 3x3 conv, 256
CONV-3x3-128	[112x112x128]	1.6 M	$(3*3*64)*128 = 73,728$	3x3 conv, 512 3x3 conv, 512 3x3 conv, 512 3x3 conv, 512
CONV-3x3-128	[112x112x128]	1.6 M	$(3*3*128)*128 = 147,456$	3x3 conv, 512 3x3 conv, 512 3x3 conv, 512 3x3 conv, 512
POOL-2x2	[56x56x128]	400 K	0	3x3 conv, 512 3x3 conv, 512 3x3 conv, 512 3x3 conv, 512
CONV-3x3-256	[56x56x256]	800 K	$(3*3*128)*256 = 294,912$	3x3 conv, 512 3x3 conv, 512 3x3 conv, 512 3x3 conv, 512
CONV-3x3-256	[56x56x256]	800 K	$(3*3*256)*256 = 589,824$	3x3 conv, 512 3x3 conv, 512 3x3 conv, 512 3x3 conv, 512
CONV-3x3-256	[56x56x256]	800 K	$(3*3*256)*256 = 589,824$	3x3 conv, 512 3x3 conv, 512 3x3 conv, 512 3x3 conv, 512
POOL-2x2	[28x28x256]	200 K	0	3x3 conv, 512 3x3 conv, 512 3x3 conv, 512 3x3 conv, 512
CONV-3x3-512	[28x28x512]	400 K	$(3*3*256)*512 = 1,179,648$	3x3 conv, 512 3x3 conv, 512 3x3 conv, 512 3x3 conv, 512
CONV-3x3-512	[28x28x512]	400 K	$(3*3*512)*512 = 2,359,296$	3x3 conv, 512 3x3 conv, 512 3x3 conv, 512 3x3 conv, 512
CONV-3x3-512	[28x28x512]	400 K	$(3*3*512)*512 = 2,359,296$	3x3 conv, 512 3x3 conv, 512 3x3 conv, 512 3x3 conv, 512
POOL-2x2	[14x14x512]	100 K	0	3x3 conv, 512 3x3 conv, 512 3x3 conv, 512 3x3 conv, 512
CONV-3x3-512	[14x14x512]	100 K	$(3*3*512)*512 = 2,359,296$	3x3 conv, 512 3x3 conv, 512 3x3 conv, 512 3x3 conv, 512
CONV-3x3-512	[14x14x512]	100 K	$(3*3*512)*512 = 2,359,296$	3x3 conv, 512 3x3 conv, 512 3x3 conv, 512 3x3 conv, 512
POOL-2x2	[7x7x512]	25 K	0	3x3 conv, 512 3x3 conv, 512 3x3 conv, 512 3x3 conv, 512
FC-4096	[1x1x4096]	4094	$7*7*512*4096 = 102,760,448$	FC 4096 FC 4096 FC 4096 FC 4096 Softmax
FC-4096	[1x1x4096]	4096	$4096*4096 = 16,777,216$	FC 4096 FC 4096 FC 4096 FC 4096
FC-4096	[1x1x1000]	1000	$4096*1000 = 4,096,000$	FC 1000 Softmax
				VGG16 191

191



192



193

## Réseaux récurrents (si le temps le permet)

194

---



---



---



---



---



---



---

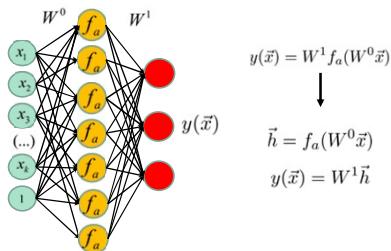


---



---

### Réseau de neurones de base (régression)



$$y(\vec{x}) = W^1 f_a(W^0 \vec{x})$$

$$\vec{h} = f_a(W^0 \vec{x})$$

$$y(\vec{x}) = W^1 \vec{h}$$

$f_a$  : fonction d'activation

195

---



---



---



---



---



---



---

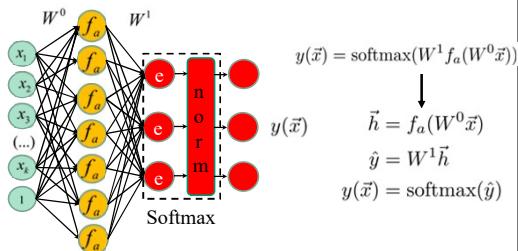


---



---

### Réseau de neurones de base (classification)



$$y(\vec{x}) = \text{softmax}(W^1 f_a(W^0 \vec{x}))$$

$$\vec{h} = f_a(W^0 \vec{x})$$

$$\hat{y} = W^1 \vec{h}$$

$$y(\vec{x}) = \text{softmax}(\hat{y})$$

196

---



---



---



---



---



---



---

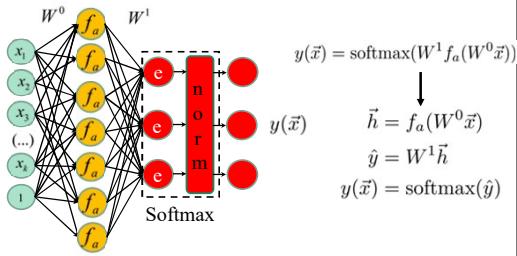


---



---

## Réseau de neurones de base (classification)

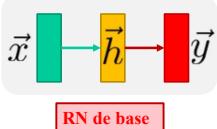


Ne permet que des tâches “1 pour 1”

- Classification (1 image = 1 étiquette)
  - Régression (1 donnée = 1 vecteur)
  - Localisation (1 boîte = 1 classification + 1 régression)

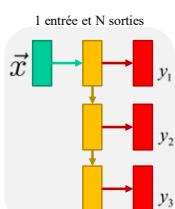
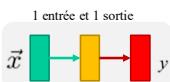
197

## Illustration simplifiée



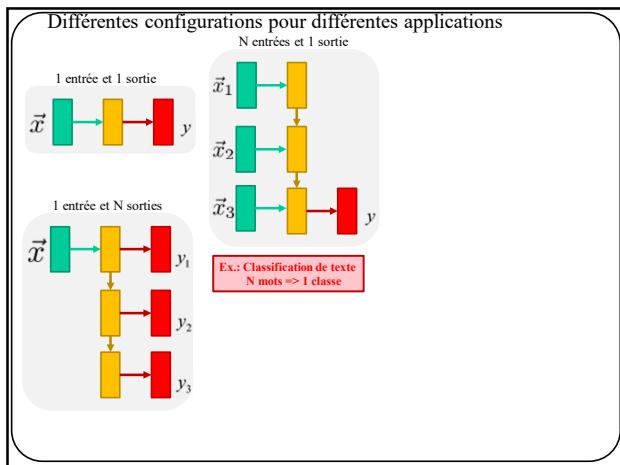
198

Différentes configurations pour différentes applications



**Ex.: description d'une image**  
1 image => N mots

199



200

---

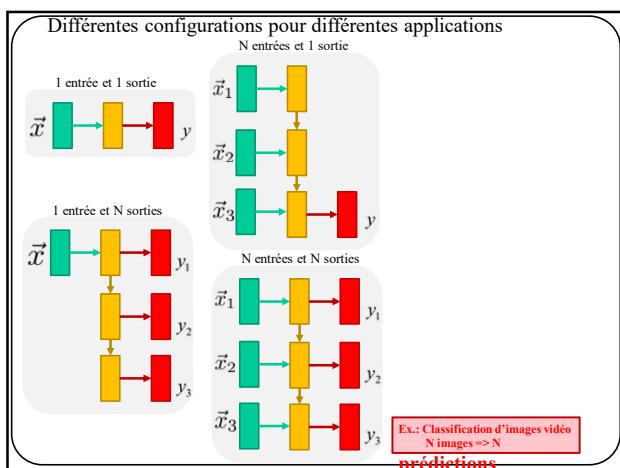
---

---

---

---

---



201

---

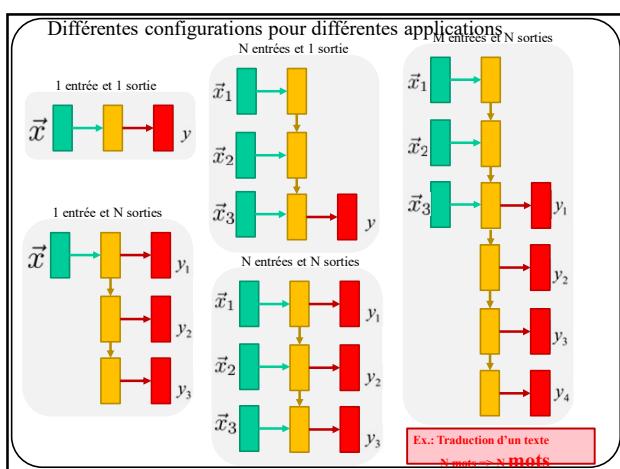
---

---

---

---

---



202

---

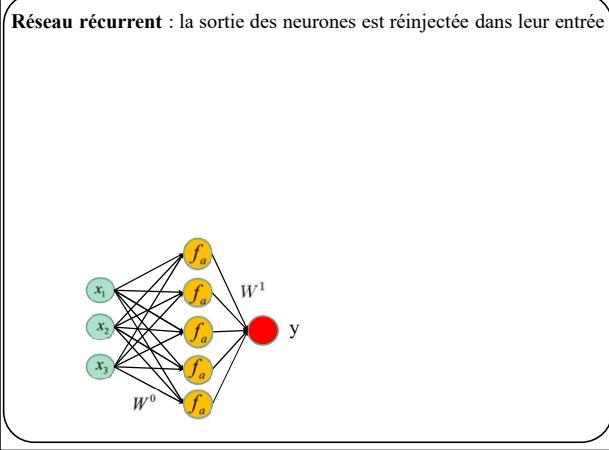
---

---

---

---

---



203

---

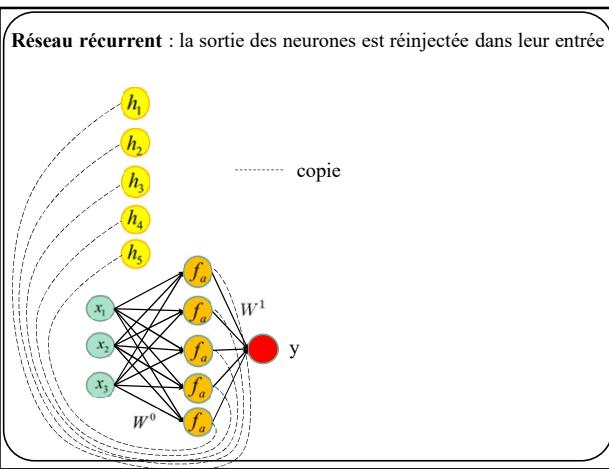
---

---

---

---

---



204

---

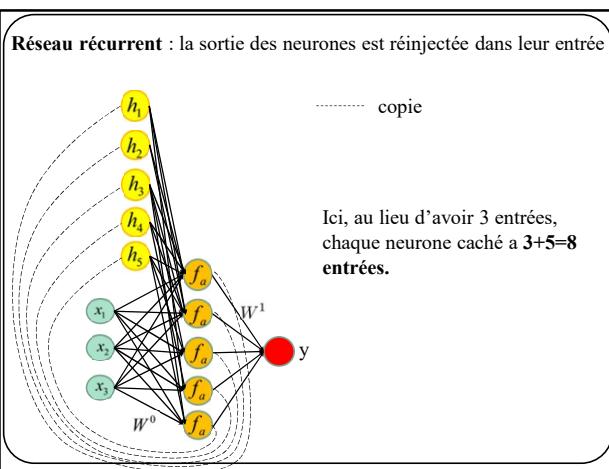
---

---

---

---

---



205

---

---

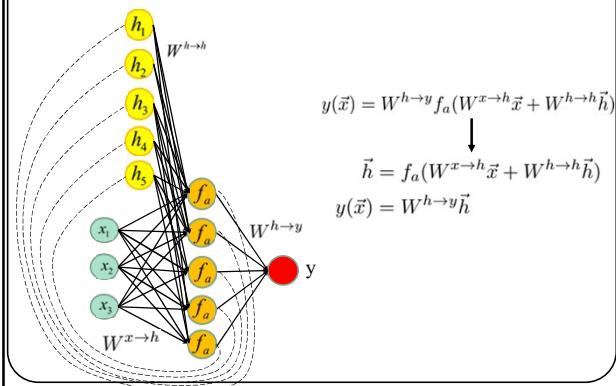
---

---

---

---

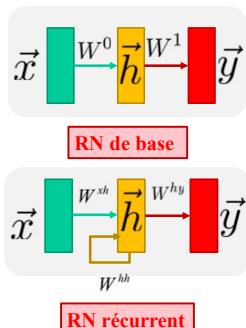
**Réseau récurrent** : la sortie des neurones est réinjectée dans leur entrée



$$\begin{aligned} y(\vec{x}) &= W^{h \rightarrow y} f_a(W^{x \rightarrow h} \vec{x} + W^{h \rightarrow h} \vec{h}) \\ &\quad \downarrow \\ \vec{h} &= f_a(W^{x \rightarrow h} \vec{x} + W^{h \rightarrow h} \vec{h}) \\ y(\vec{x}) &= W^{h \rightarrow y} \vec{h} \end{aligned}$$

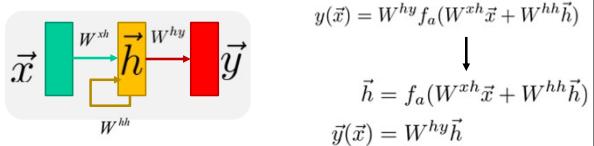
206

### Illustration simplifiée



207

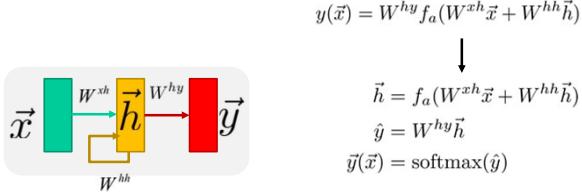
Dans le cas général avec K sorties (régression)



$$\begin{array}{c} y(\vec{x}) = W^{hy} f_a(W^{xh}\vec{x} + W^{hh}\vec{h}) \\ \downarrow \\ \vec{h} = f_a(W^{xh}\vec{x} + W^{hh}\vec{h}) \\ \vec{y}(\vec{x}) = W^{hy}\vec{h} \end{array}$$

208

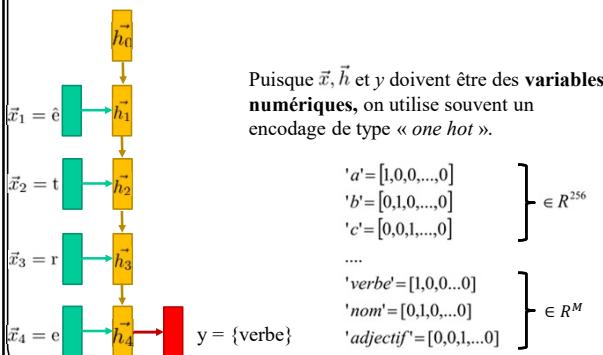
Dans le cas général avec K sorties (classification)



209

Exemple pour N entrées et 1 sortie:

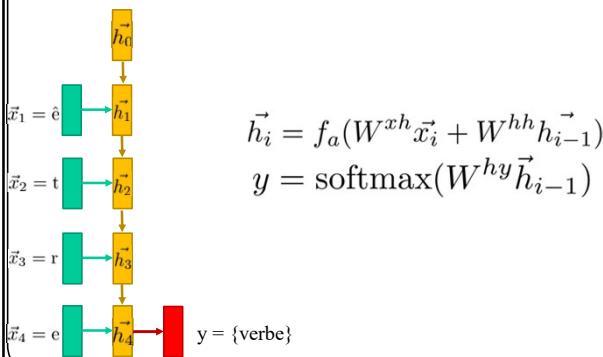
Analyse grammaticale (classification) : (é.t.r.e)=> {verbe}



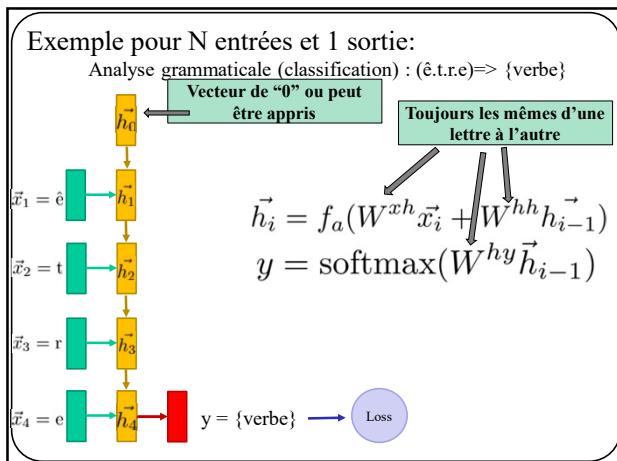
210

Exemple pour N entrées et 1 sortie:

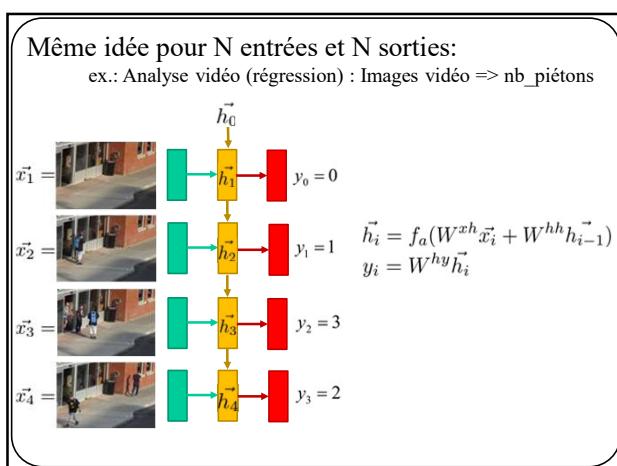
Analyse grammaticale (classification) : (é.t.r.e)=> {verbe}



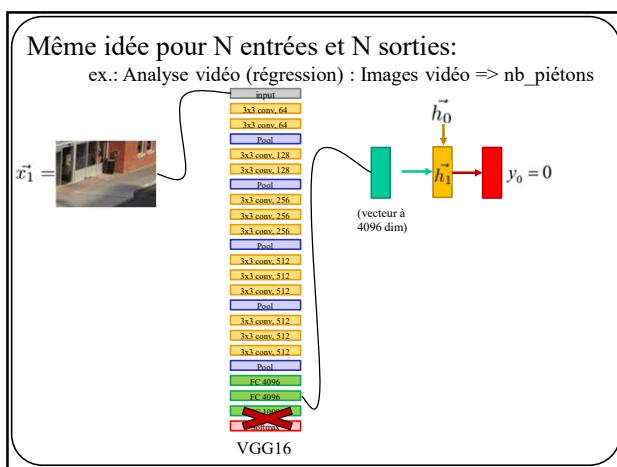
211



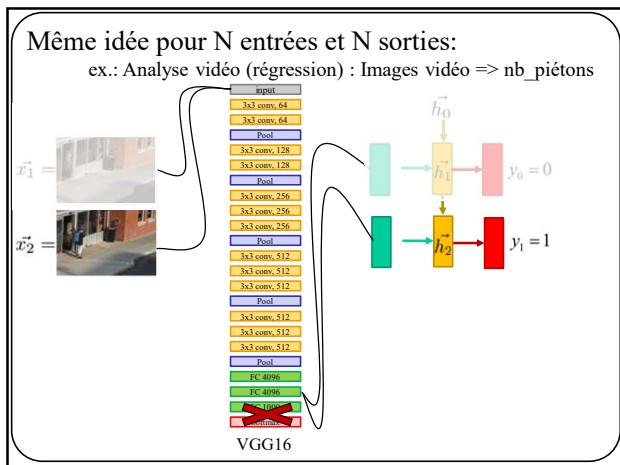
212



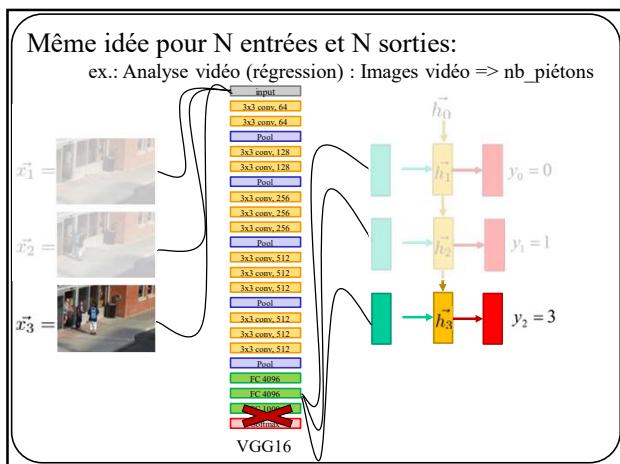
213



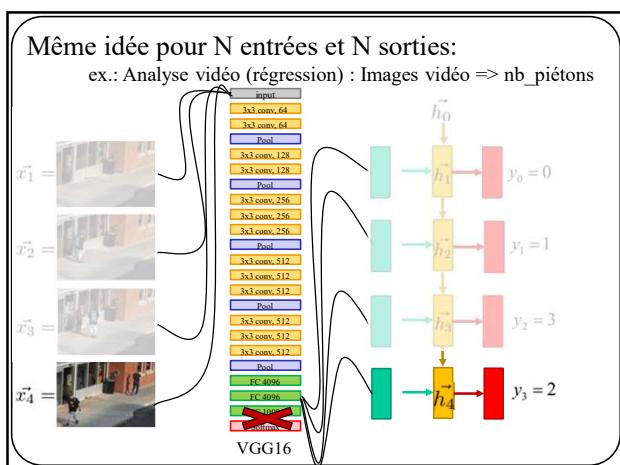
214



215



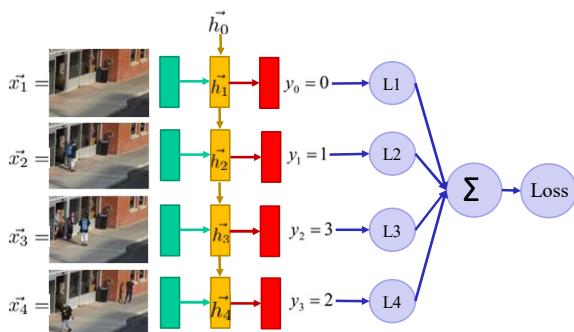
216



217

Même idée pour N entrées et N sorties:

ex.: Analyse vidéo (régression) : Images vidéo => nb piétons



---

---

---

---

---

---

---

---

---

---

218

Autre exemple: **prédition de caractères** (modèle de langue)

**Alphabet jouet :[a,e,m,s]**

### Représentation « one hot » jouet:

$$\begin{aligned} 'a' &= [1, 0, 0, 0] \\ 'e' &= [0, 1, 0, 0] \\ 'm' &= [0, 0, 1, 0] \\ 's' &= [0, 0, 0, 1] \end{aligned}$$

**But :** Entrainer un modèle à prédire les lettres du mot « **masse** ».

---

---

---

---

---

---

---

---

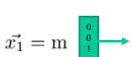
---

219

Autre exemple: **prédition de caractères** (modèle de langue)

Alphabet :[a,e,m,s]

**Entrainer** un modèle à prédire les lettres du mot « **masse** ».



---

---

---

---

---

---

---

---

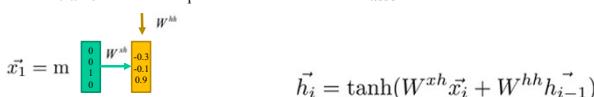
---

220

Autre exemple: **prédition de caractères** (modèle de langue)

Alphabet :[a,e,m,s]

**Entraîner** un modèle à prédire les lettres du mot « **masse** ».

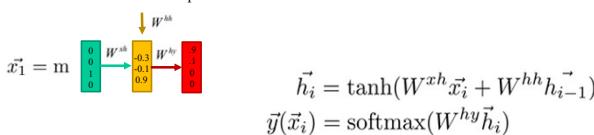


221

Autre exemple: **prédition de caractères** (modèle de langue)

Alphabet :[a,e,m,s]

**Entrainer** un modèle à prédire les lettres du mot « **masse** ».

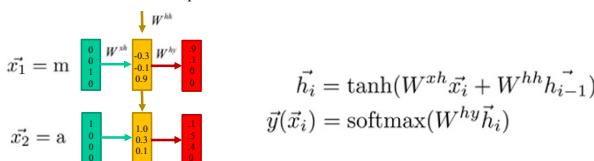


222

Autre exemple: **prédition de caractères** (modèle de langue)

Alphabet :[**a,e,m,s**]

**Entraîner** un modèle à prédire les lettres du mot « **masse** ».

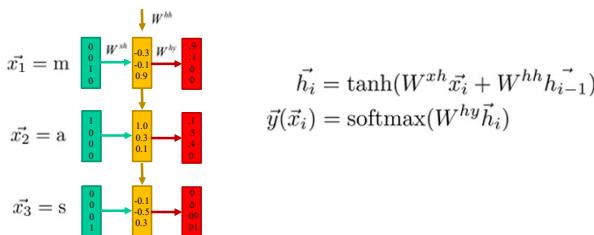


223

### Autre exemple: prédiction de caractères (modèle de langue)

Alphabet :{a,e,m,s}

**Entrainer** un modèle à prédire les lettres du mot « **masse** ».

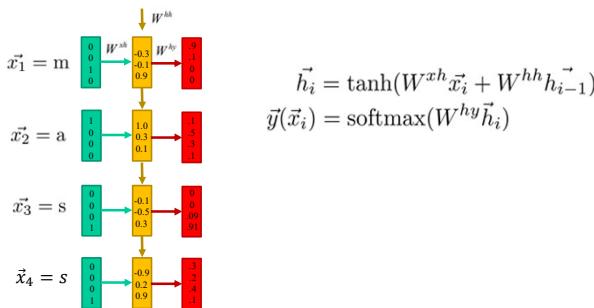


224

### Autre exemple: prédiction de caractères (modèle de langue)

Alphabet :{a,e,m,s}

**Entrainer** un modèle à prédire les lettres du mot « **masse** ».

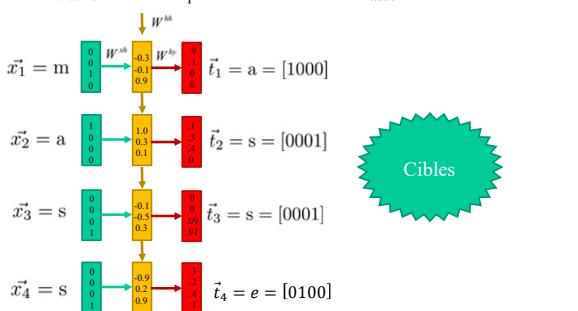


225

### Autre exemple: prédiction de caractères (modèle de langue)

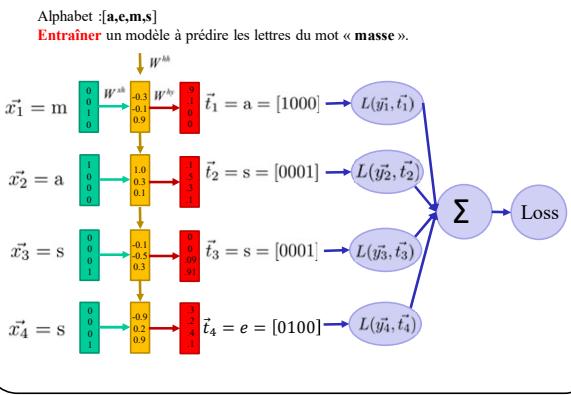
Alphabet :{a,e,m,s}

**Entrainer** un modèle à prédire les lettres du mot « **masse** ».



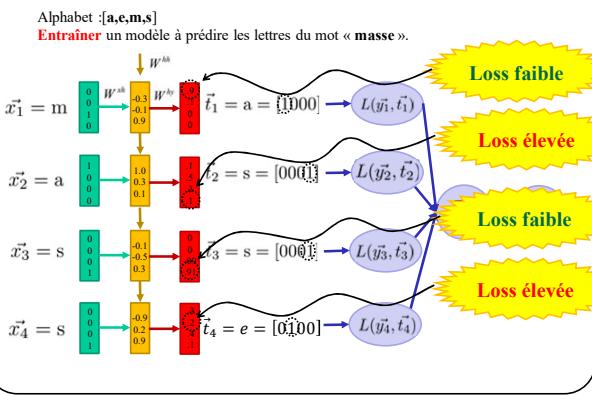
226

Autre exemple: **prédition de caractères** (modèle de langue)



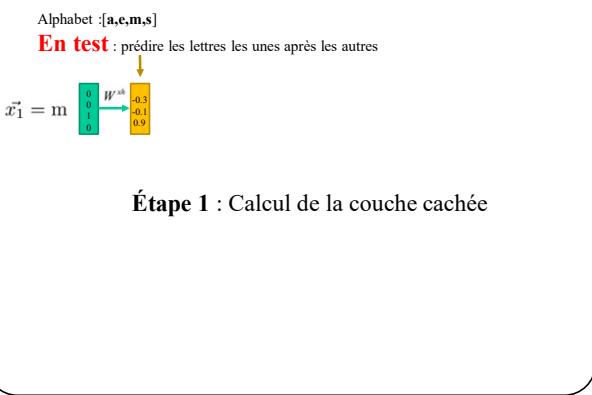
227

Autre exemple: **prédition de caractères** (modèle de langue)



228

Autre exemple: **prédition de caractères** (modèle de langue)

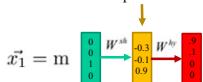


229

Autre exemple: **prédition de caractères** (modèle de langue)

Alphabet :[a,e,m,s]

**En test** : prédire les lettres les unes après les autres



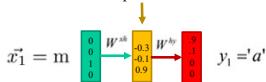
### **Étape 2 : Calcul de la sortie (softmax)**

230

Autre exemple: **prédition de caractères** (modèle de langue)

Alphabet :[a,e,m,s]

**En test**: prédire les lettres les unes après les autres



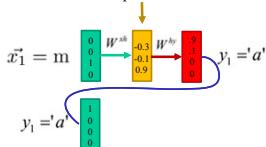
### **Étape 3 : Sélectionner le caractère le plus probable**

231

Autre exemple: **prédition de caractères** (modèle de langue)

Alphabet :[**a,e,m,s**]

**En test** : prédire les lettres les unes après les autres



**Étape 4 :** Injecter le caractère prédit au début du réseau

232

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

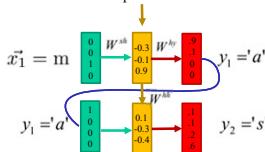
---

---

Autre exemple: **prédition de caractères** (modèle de langue)

Alphabet :[a,e,m,s]

**En test** : prédire les lettres les unes après les autres



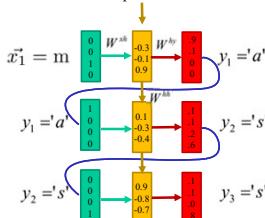
Et on recommence!

233

Autre exemple: **prédition de caractères** (modèle de langue)

Alphabet :[a,e,m,s]

**En test**: prédire les lettres les unes après les autres

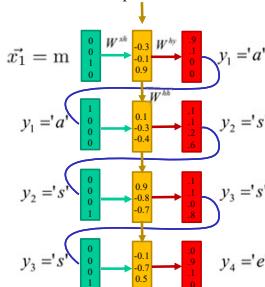


234

Autre exemple: **prédition de caractères** (modèle de langue)

Alphabet :[**a,e,m,s**]

**En test** : prédire les lettres les unes après les autres



235



Texte généré une fois le modèle entraîné

<p>PANDARUS:</p> <p>Alas, I think he shall be come approached and the day When little brain would be attain'd into being never fed, And who is but a chain and subjects of his death, I should not sleep.</p> <p>Second Senator: They are away this miseries, produced upon my soul, Breaking and strongly should be buried, when I perish The earth and thoughts of many states.</p> <p>DUKE VINCENTIO: Well, your wit is in the case of side and that.</p> <p>Second Lord: They would be ruled after this chamber, and my fair names begun out of the fact, to be conveyed, Whose noble souls I'll have the heart of the vars.</p> <p>CLOWN: Come, sir, I will make did behold your worship.</p> <p>VIOGLIA: I'll drink it.</p>	<p>VIOLA: Why, Salisbury must find his flesh and though That which I am not age, not a man and in fire, To shew the hand of a man and a ware, To grace my hand-reach within, and not a fair are hand, The Caesar and my goodly father's world. When I was heaven of presence and our fleets, We spare with hours, but cut thy council I am great, Murdered and by thy master's ready there My power to give thee but so much as hell; Some service in the noble bondman here, Would show him to her wine.</p> <p>KING LEAR: O, if you were a feeble sight, the courtesy of your law, Your sight and several breath, will wear the gods With his heads, and my ladies are wonder'd at the deeds, So drop upon your lordship's head, and your opinion Shall be against your honour.</p>
---	---

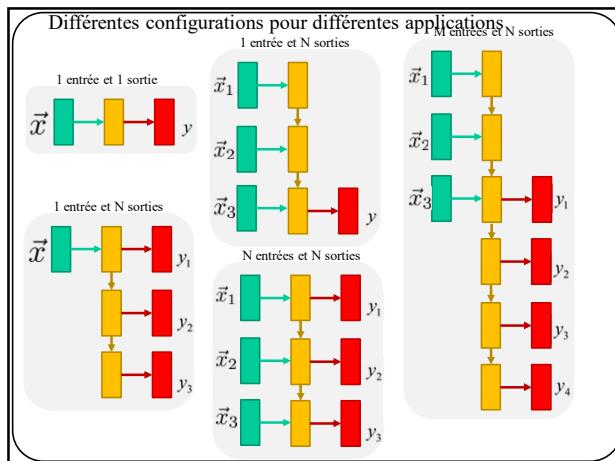
Crédit: A. Karpathy, CS231

239

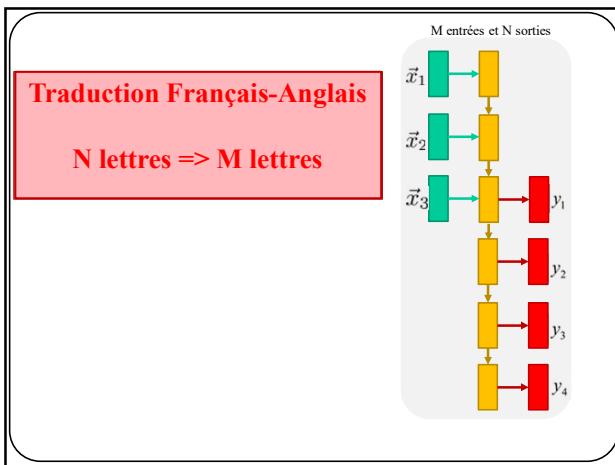
Entraînement sur le code source de Linux en C++

Crédit: A. Karpathy, CS231

240



241



242

---

---

---

---

---

---

---

---

**Autre exemple: traduction**

Traduire 'assez' > 'enough'  
Alphabet fr: [<BoS>, a, e, s, z, <EoS>]  
Alphabet en: [<BoS>, e, g, h, n, o, u, <EoS>]

Sutskever, I., Vinyals, O., & Le, Q. V. (2014). Sequence to sequence learning with neural networks. *Advances in neural information processing systems*, 27.

Pas le même nombre d'entrées que de sorties !  
(BoS : Begining of Sentence, EoS:End of Sentence).

---

---

---

---

---

---

---

---

243

**Autre exemple: traduction**

Traduire 'assez' -> 'enough'  
Alphabet fr: [<BoS>, a, e, s, z, <EoS>]  
Alphabet en: [<BoS>, e, g, h, n, o, u, <EoS>]

Sutskever, I., Vinyals, O., & Le, Q. V. (2014). Sequence to sequence learning with neural networks. *Advances in neural information processing systems*, 27.

$\vec{x}_1 = \text{<BoS>}$

$W^h$

---

---

---

---

---

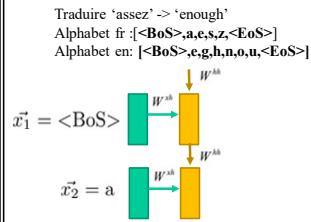
---

---

---

244

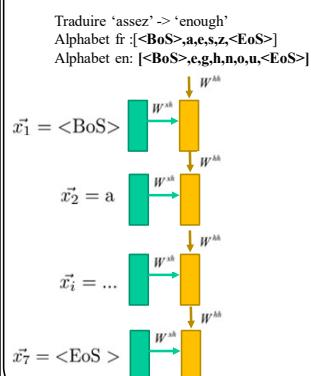
### Autre exemple: traduction



Sutskever, I., Vinyals, O., & Le, Q. V. (2014). Sequence to sequence learning with neural networks. *Advances in neural information processing systems*, 27.

245

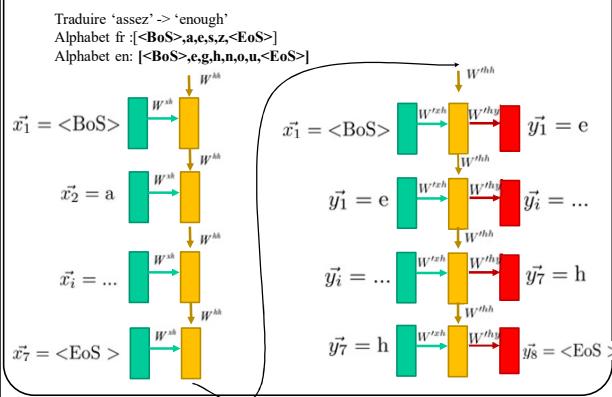
### Autre exemple: traduction



Sutskever, I., Vinyals, O., & Le, Q. V. (2014). Sequence to sequence learning with neural networks. *Advances in neural information processing systems*, 27.

246

### Autre exemple: traduction

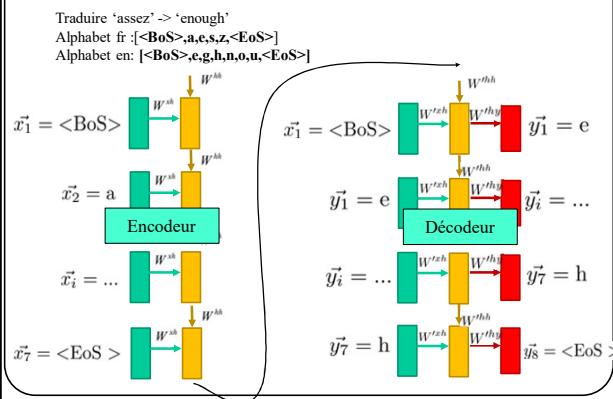


Sutskever, I., Vinyals, O., & Le, Q. V. (2014). Sequence to sequence learning with neural networks. *Advances in neural information processing systems*, 27.

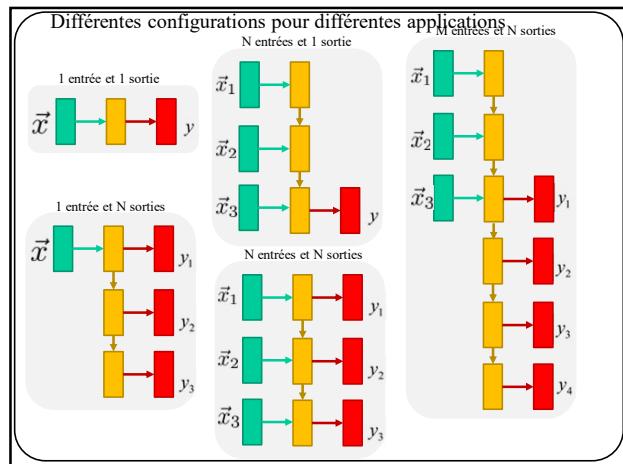
247

### Autre exemple: traduction

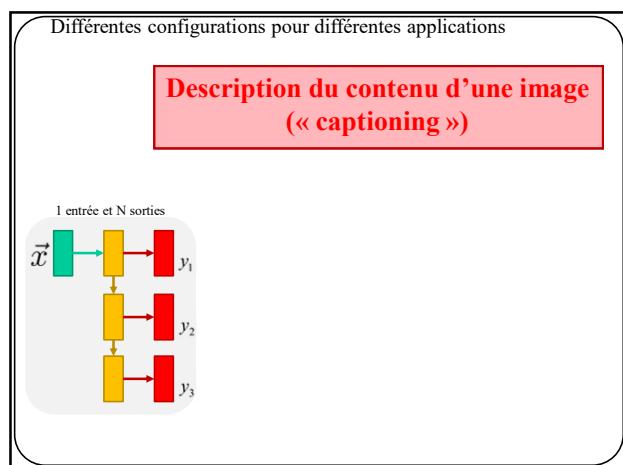
Sutskever, I., Vinyals, O., & Le, Q. V. (2014). Sequence to sequence learning with neural networks. *Advances in neural information processing systems*, 27.



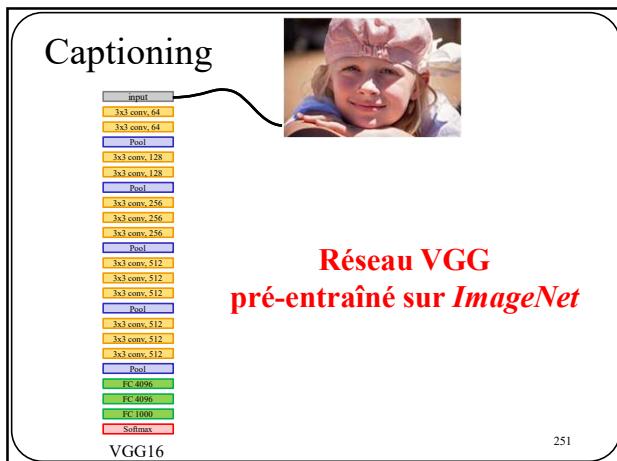
248



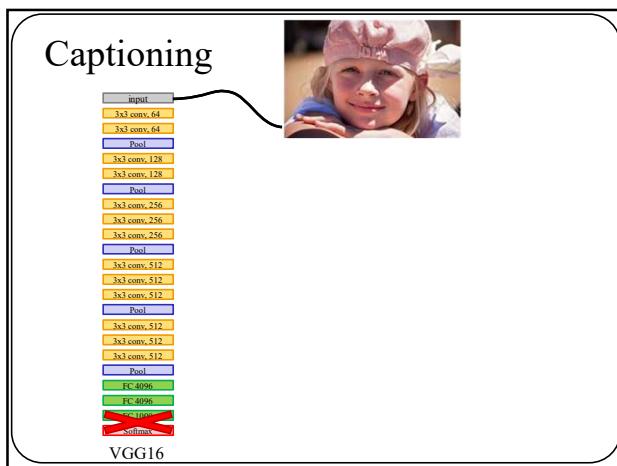
249



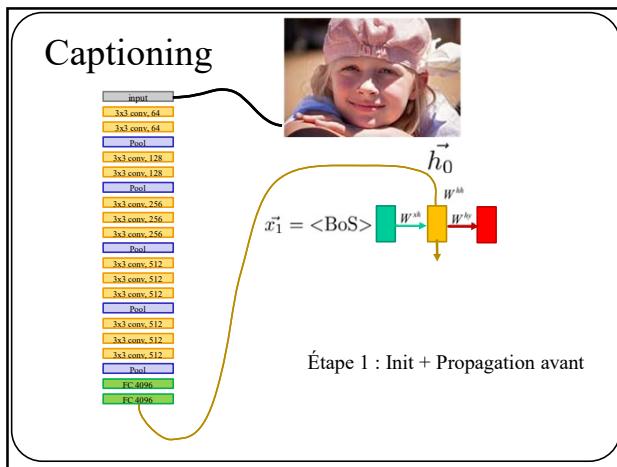
250



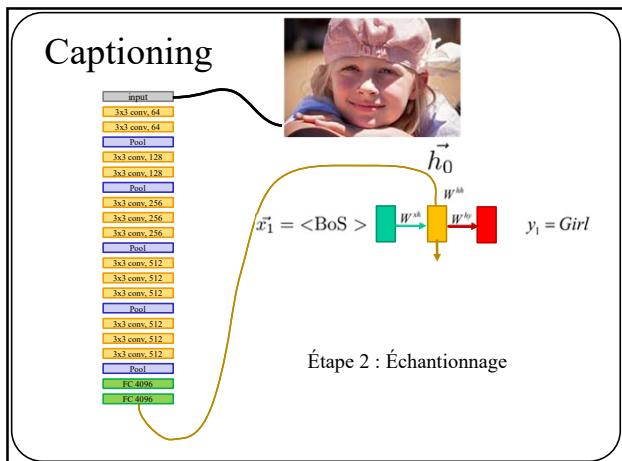
251



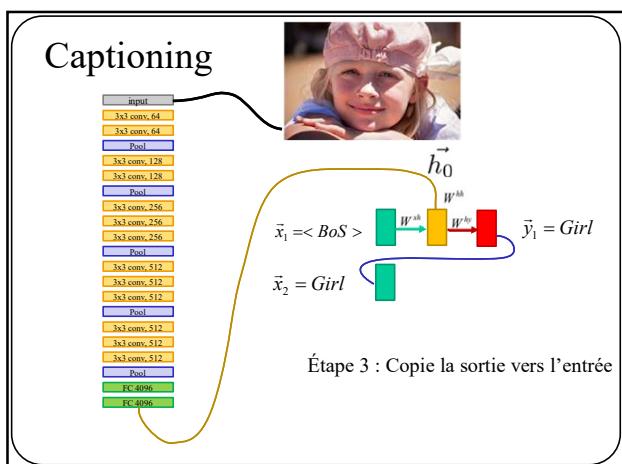
252



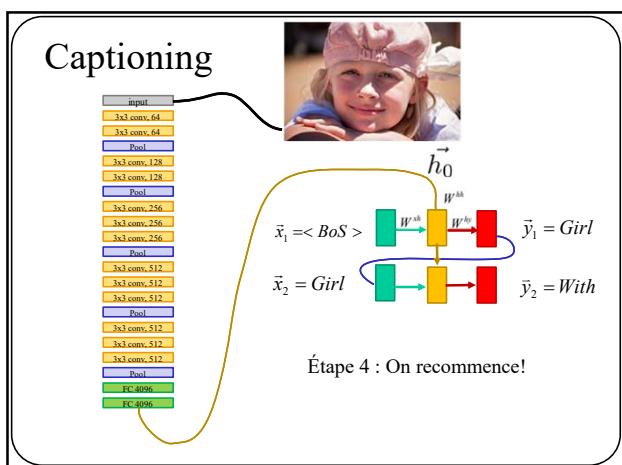
253



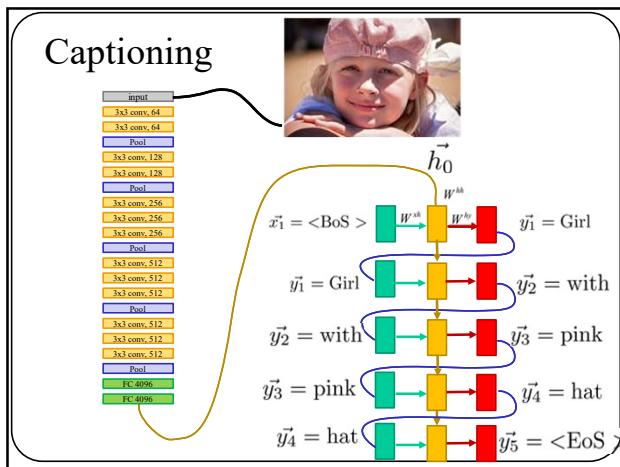
254



255



256



257



258



259