Réseaux de neurones

IFT 780

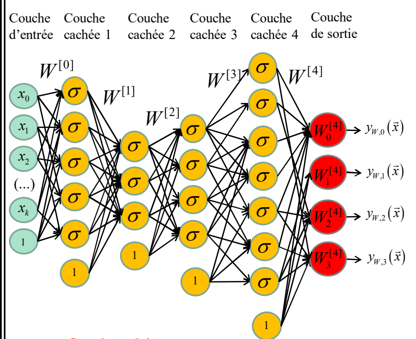
Réseaux à convolution

Par

Pierre-Marc Jodoin

1

kD, 4 Classes, Réseau à 4 couches cachées



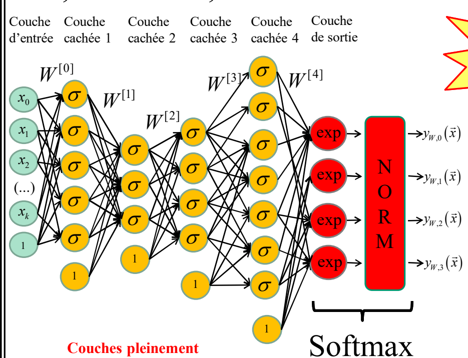
Hinge loss

Couches pleinement connectées (*fully-connected layers*)

$$y_W(\vec{x}) = W^{[4]}\sigma\left(W^{[3]}\sigma\left(W^{[2]}\sigma\left(W^{[1]}\sigma\left(W^{[0]}\vec{x}\right)\right)\right)\right)$$

2

kD, 4 Classes, Réseau à 4 couches cachées



Entropie Croisée

Softmax

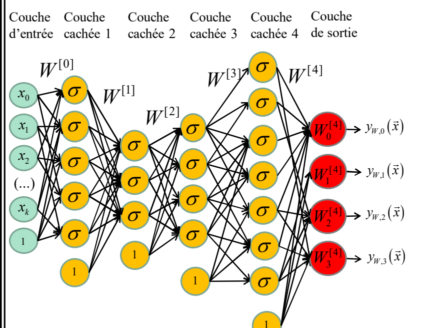
$$y_{W,i}(\mathrm{x}) = \frac{f_{W,i}}{\sum_k f_{W,k}}$$

Couches pleinement connectées (*fully-connected layers*)

$$y_W(\vec{x}) = \text{softmax}\left(W^{[4]}\sigma\left(W^{[3]}\sigma\left(W^{[2]}\sigma\left(W^{[1]}\sigma\left(W^{[0]}\vec{x}\right)\right)\right)\right)\right)$$

3

kD, 4 Classes, Réseau à 4 couches cachées



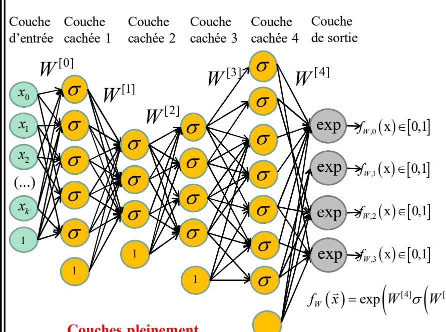
$W^{[0]} \in R^{5 \times k+1}$
$W^{[1]} \in R^{3 \times 6}$
$W^{[2]} \in R^{4 \times 4}$
$W^{[3]} \in R^{5 \times 7}$
$W^{[4]} \in R^{4 \times 8}$

Couches pleinement connectées (*fully-connected layers*)

$$y_W(\vec{x}) = W^{[4]}\sigma\left(W^{[3]}\sigma\left(W^{[2]}\sigma\left(W^{[1]}\sigma\left(W^{[0]}\vec{x}\right)\right)\right)\right)$$

4

kD, 4 Classes, Réseau à 4 couches cachées



$W^{[0]} \in R^{5 \times k+1}$
$W^{[1]} \in R^{3 \times 6}$
$W^{[2]} \in R^{4 \times 4}$
$W^{[3]} \in R^{5 \times 7}$
$W^{[4]} \in R^{4 \times 8}$

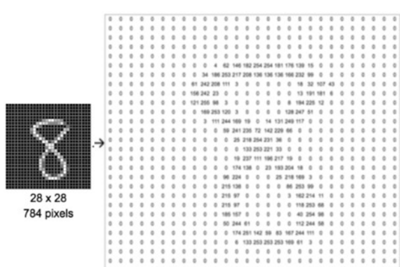
$$f_W(\vec{x}) = \exp\left(W^{[4]}\sigma\left(W^{[3]}\sigma\left(W^{[2]}\sigma\left(W^{[1]}\sigma\left(W^{[0]}\vec{x}\right)\right)\right)\right)\right)$$

Couches pleinement connectées (*fully-connected layers*)

Softmax $y_{W,i}(\mathrm{x}) = \dfrac{f_{W,i}}{\sum_k f_{W,k}}$

5

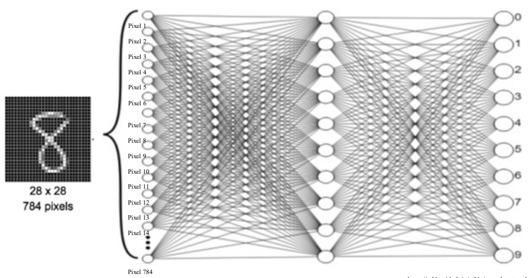
Comment classifier des images?



https://ml4a.github.io/ml4a/neural_networks/

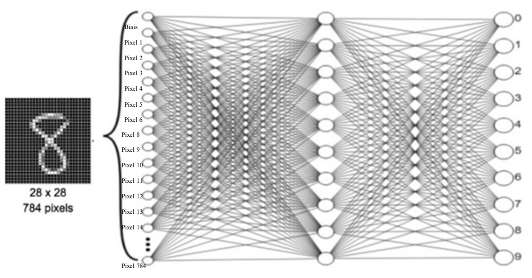
6

Comment classifier des images?



7

Beaucoup de paramètres (7850 dans la couche 1)



8

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

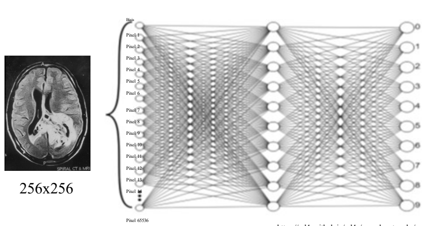


Image médicale (IRM de cerveau)

9

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

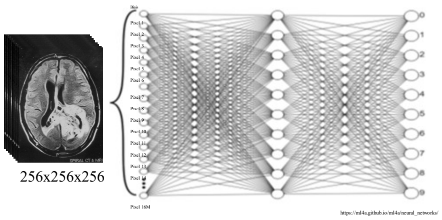


Image médicale 3D (IRM de cerveau)

10

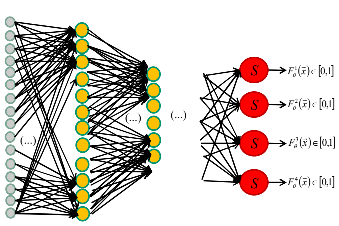
Comment réduire le nombre de connections?

11

11

Comment réduire le nombre de connections?

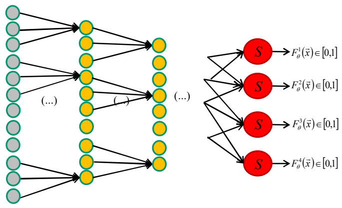
Les **couches pleinement connectées (*fully-connected layers*)** sont problématiques lorsque le **nombre de neurones est élevé.**



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

12

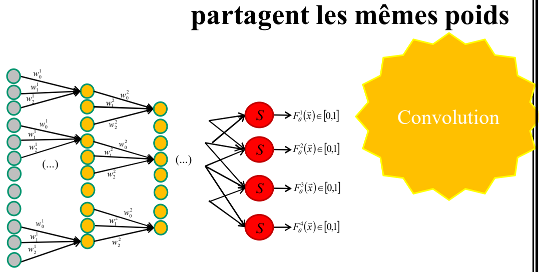
Solution : connexions partielles



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**



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 |
× × ×
| .3 | .2 | .1 |

3+4+-3 → | 4 | | |

(f*W)(2)

| 10 | 20 | -30 | 40 | -50 |
× × ×
| .3 | .2 | .1 |

6-6+4 → | 4 | 4 | |

(f*W)(3)

| 10 | 20 | -30 | 40 | -50 |
× × ×
| .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 |
× × ×
| .1 | .2 | .3 |

1+4-9 → | -4 | | |

(f*W)(2)

| 10 | 20 | -30 | 40 | -50 |
× × ×
| .1 | .2 | .3 |

2-6+12 → | -4 | 8 | |

(f*W)(3)

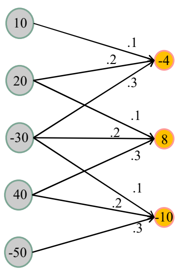
| 10 | 20 | -30 | 40 | -50 |
× × ×
| .1 | .2 | .3 |

-3+8-15 → | -4 | 8 | -10 |

18

18

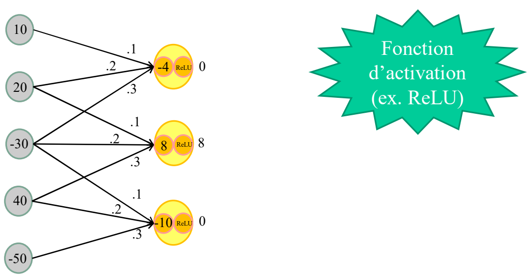
L'opération de la page précédente est équivalente à



19

19

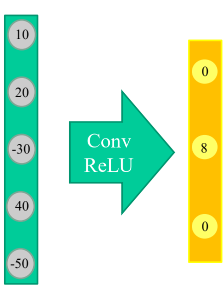
L'opération de la page précédente est équivalente à



20

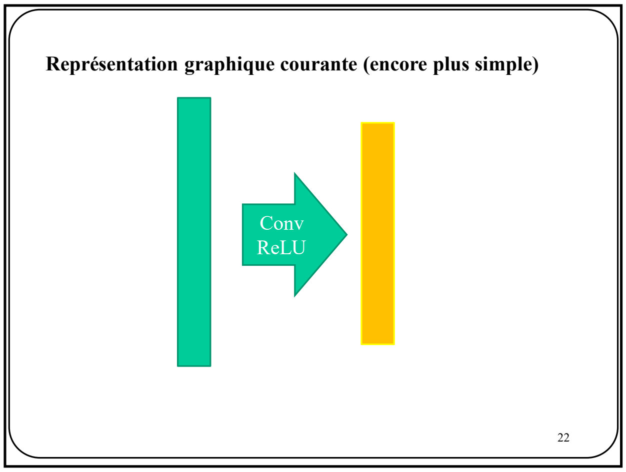
20

Représentation graphique courante (simple)

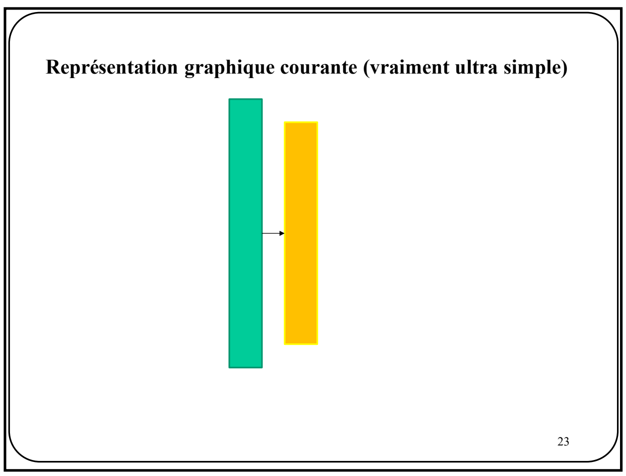


21

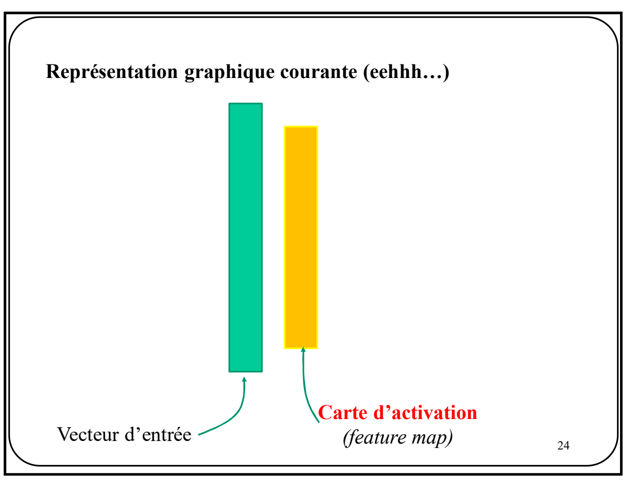
21


Représentation graphique courante (encore plus simple)
Conv
ReLU
22

22

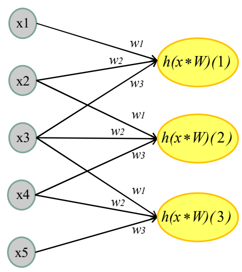

Représentation graphique courante (vraiment ultra simple)
23

23


Représentation graphique courante (eehhh…)
Vecteur d’entrée
Carte d’activation
(feature map)
24

24

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



h: fonction d'activation
(tanh, ReLU, elu, etc.)

25

25

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

D=9

40	**50**	**70**	80	90	10	20	30	40

prod. scal.

F=3

.2	**-.3**	**.4**

21

26

26

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

D=9

40	**50**	**70**	**80**	90	10	20	30	40

prod. scal.

F=3

.2	**-.3**	**.4**

21	**21**

Stride = 1

27

27

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

D=9

40	50	**70**	**80**	**90**	10	20	30	40

prod. scal.

F=3

.2	**-.3**	**.4**

21	21	**26**

28

28

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

D=9

40	50	70	**80**	**90**	**10**	20	30	40

prod. scal.

F=3

.2	**-.3**	**.4**

21	21	26	**-7**

29

29

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

D=9

40	50	70	80	**90**	**10**	**20**	30	40

prod. scal.

F=3

.2	**-.3**	**.4**

21	21	26	-7	**23**

30

30

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

D=9

40	50	70	80	90	**10**	**20**	**30**	40

prod. scal.

F=3

.2	**-.3**	**.4**

21	21	26	-7	23	**8**

31

31

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

D=9

40	50	70	80	90	10	**20**	**30**	**40**

prod. scal.

F=3

.2	**-.3**	**.4**

21	21	26	-7	23	8	**11**

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

32

32

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

D=9

40	**50**	**70**	**80**	**90**	10	20	30	40

prod. scal.

F=5

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

35

33

33

Stride et calcul de la taille de la carte d’activation

D=9

40	**50**	**70**	**80**	**90**	**10**	20	30	40

prod. scal.

F=5

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

35	**-18**

Stride = 1

34

34

Stride et calcul de la taille de la carte d’activation

D=9

40	50	**70**	**80**	**90**	**10**	**20**	30	40

prod. scal.

F=5

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

35	-18	**33**

35

35

Stride et calcul de la taille de la carte d’activation

D=9

40	50	70	**80**	**90**	**10**	**20**	**30**	40

prod. scal.

F=5

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

35	-18	33	**1**

36

36

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

D=9

40	50	70	80	**90**	**10**	**20**	**30**	**40**

prod. scal.

F=5

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

35	-18	33	1	**32**

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

37

37

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

D=9

40	**50**	**70**	**80**	**90**	**10**	**20**	30	40

prod. scal.

F=7

.2	**-.3**	**.4**	**-.5**	**.6**	**-.7**	**.8**

44

38

38

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

D=9

40	**50**	**70**	**80**	**90**	**10**	**20**	**30**	40

prod. scal.

F=7

.2	**-.3**	**.4**	**-.5**	**.6**	**-.7**	**.8**

44	**-8**

Stride = 1

39

39

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

D=9

40	50	**70**	**80**	**90**	**10**	**20**	**30**	**40**

prod. scal.

F=7

.2	-.3	.4	-.5	.6	-.7	.8

44	-8	**44**

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

40

40

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

D=9

40	**50**	**70**	**80**	**90**	10	20	30	40

prod. scal.

F=5

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

35

41

41

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

D=9

40	50	**70**	**80**	**90**	**10**	**20**	30	40

prod. scal.

F=5

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

35	**33**

Stride = 2

42

42

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

D=9

40	50	70	80	**90**	**10**	**20**	**30**	**40**

prod. scal.

F=5

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

35	33	**32**

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

43

43

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

D=9

40	**50**	**70**	**80**	**90**	10	20	30	40

prod. scal.

F=5

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

35

44

44

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

D=9

40	50	70	**80**	**90**	**10**	**20**	**30**	40

prod. scal.

F=5

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

35	**1**

Stride = 3

45

45

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

D=9

40	50	70	80	90	10	**20**	**30**	**40**	**?**	**?**

prod. scal.

F=5

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

35	**1**

ERREUR! Combinaison D-F-S invalide

46

46

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

Taille de la carte d'activation = **(D-F)/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

?	10	20	-30	40	-50
×	×	×			
.1	.2	.3			

Comment gérer les bords?

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

f(u)

0	10	20	-30	40	-50	0

(f*W)(u)

8	-4	8	-10	-6

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

f(u)

20	10	20	-30	40	-50	40

(f*W)(u)

10	-4	8	-10	2

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

f(u)

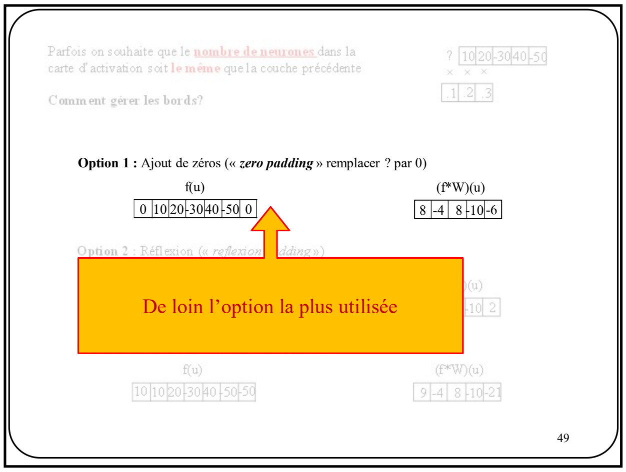
10	10	20	-30	40	-50	-50

(f*W)(u)

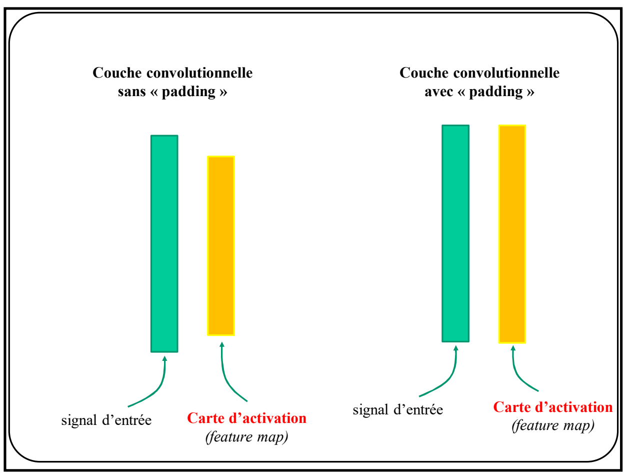
9	-4	8	-10	-21

48

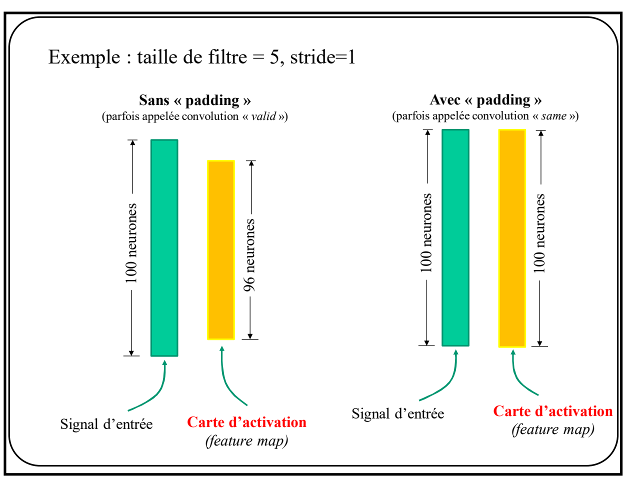
48


Parfois on souhaite que le nombre de neurones dans la carte d'activation soit le même que la couche précédente
Comment gérer les bords?
? 10 20 -30 40 -50
× × ×
.1 .2 .3
Option 1 : Ajout de zéros (« zero padding » remplacer ? par 0)
f(u)
0 10 20 -30 40 -50 0
(f*W)(u)
8 -4 8 -10 -6
Option 2 : Réflexion (« reflexion padding »)
De loin l'option la plus utilisée
-10 2
f(u)
10 10 20 -30 40 -50 -50
(f*W)(u)
9 -4 8 -10 -21
49

49


Couche convolutionnelle
sans « padding »
Couche convolutionnelle
avec « padding »
signal d'entrée
Carte d'activation
(feature map)
signal d'entrée
Carte d'activation
(feature map)

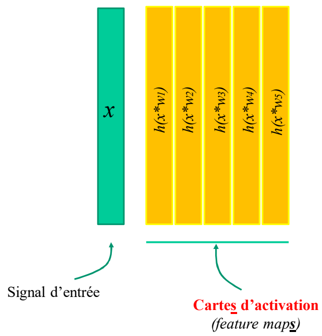
50


Exemple : taille de filtre = 5, stride=1
Sans « padding »
(parfois appelée convolution « valid »)
Avec « padding »
(parfois appelée convolution « same »)
100 neurones
96 neurones
100 neurones
100 neurones
Signal d'entrée
Carte d'activation
(feature map)
Signal d'entrée
Carte d'activation
(feature map)

51

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

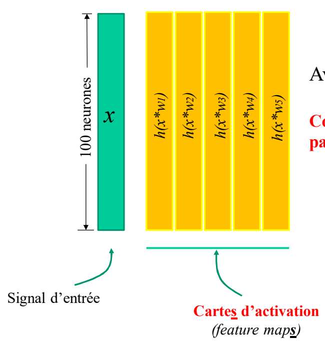
(ex. 5 filtres donnant 5 cartes d'activation)



52

Taille de filtre = 5

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



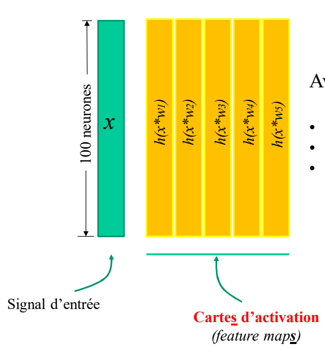
Avec « *padding* »

Combien de neurones et de paramètres au total?

53

Taille de filtre = 5

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



Avec « *padding* »

- 100 neurones par carte d'activation
- 500 neurones au total
- 25 (5x5) paramètres au total

54

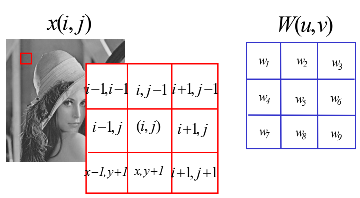
Convolution
et
couche convolutionnelle
2D

55

Filtage 2D

(sans flip de filtre)

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

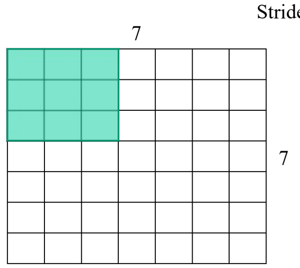


$$\begin{aligned}(x*W)(i,j) = & \; w_1 x(i-1, j-1) + w_2 x(i, j-1) + w_3 x(i+1, j-1) \\ & + w_4 x(i-1, j) \quad + w_5 x(i, j) \quad + w_6 x(i+1, j) \\ & + w_7 x(i-1, j+1) + w_8 x(i, j+1) + w_9 x(i+1, j+1)\end{aligned}$$

56

Convolution 2D

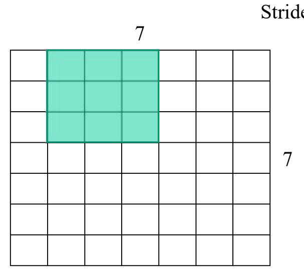
Filtre = 3x3
Stride = 1



57

Convolution 2D

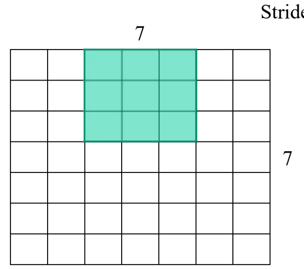
Filtre = 3x3
Stride = 1



58

Convolution 2D

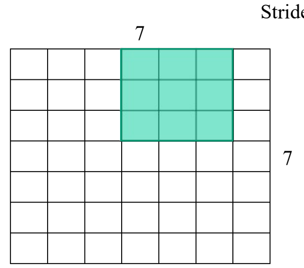
Filtre = 3x3
Stride = 1



59

Convolution 2D

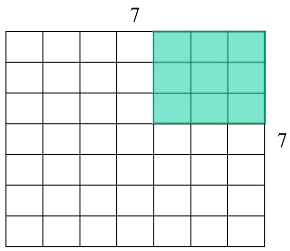
Filtre = 3x3
Stride = 1



60

Convolution 2D

Filtre = 3x3
Stride = 1

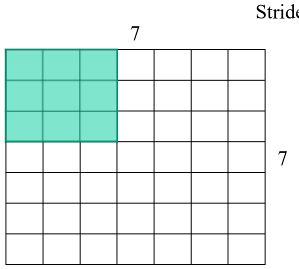


Taille de la carte d'activation (pour stride 1) = **5x5**

61

Convolution 2D

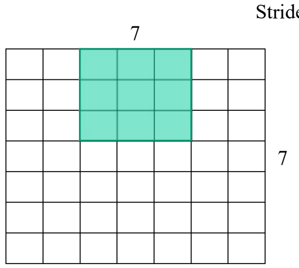
Filtre = 3x3
Stride = 2



62

Convolution 2D

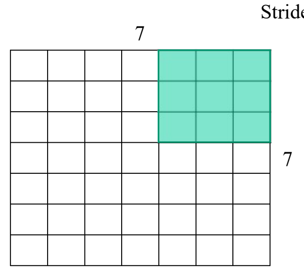
Filtre = 3x3
Stride = 2



63

Convolution 2D

Filtre = 3x3
Stride = 2

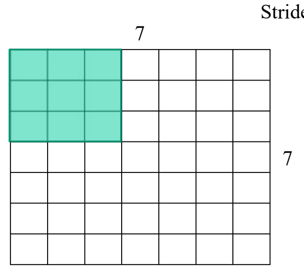


Taille de la carte d'activation (pour stride 2) = **3x3**

64

Convolution 2D

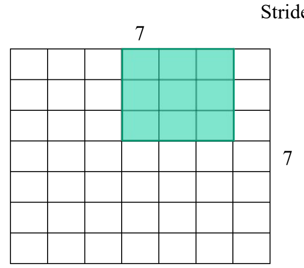
Filtre = 3x3
Stride = 3



65

Convolution 2D

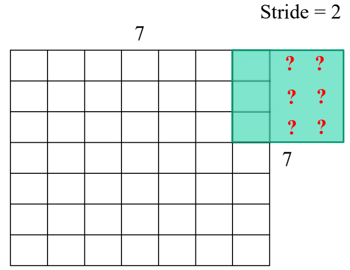
Filtre = 3x3
Stride = 3



66

Convolution 2D

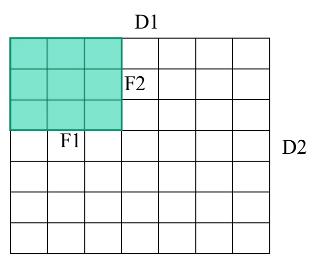
Filtre = 3x3
Stride = 2



Combinaison D-F-S invalide!

67

Convolution 2D

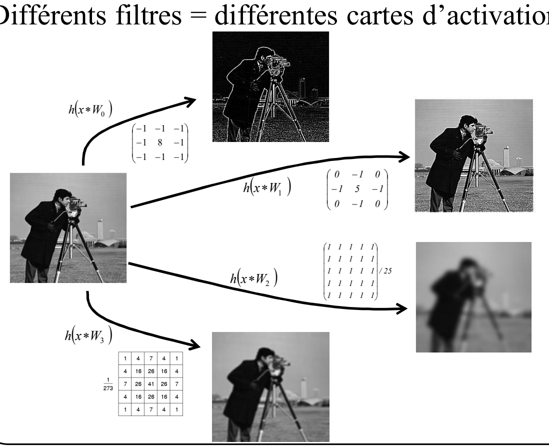


Taille de la carte d'activation :

(D1-F1)/S+1 x (D2-F2)/S+1

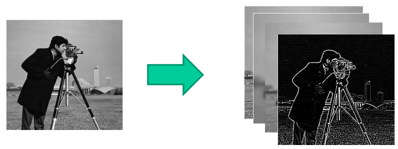
68

Différents filtres = différentes cartes d'activation



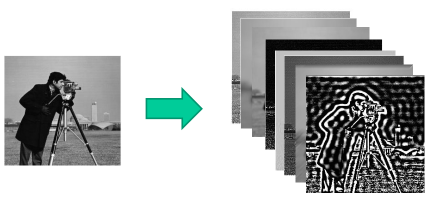
69

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



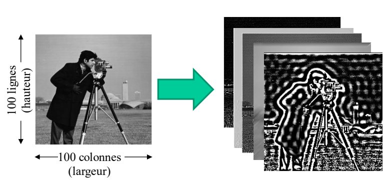
70

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



71

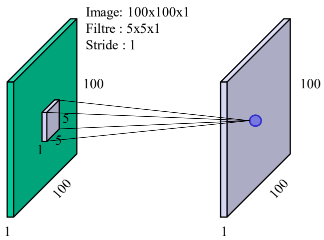
Ex.: taille de filtre : 5x5, 5 cartes d'activation, convolution « same »



- 10,000 neurones par carte d'activation
- 50,000 neurones au total
- 5x5x5 = 125 paramètres au total

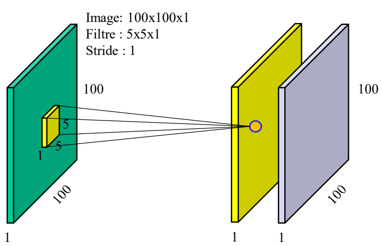
72

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



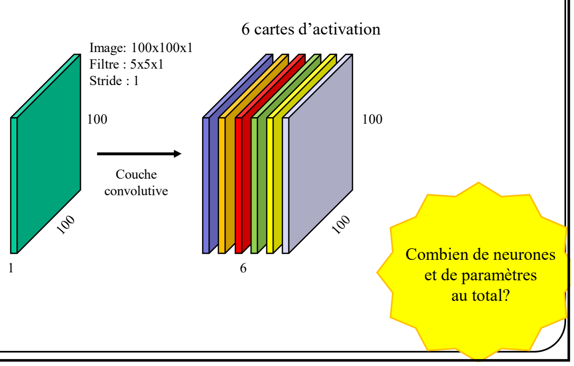
73

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



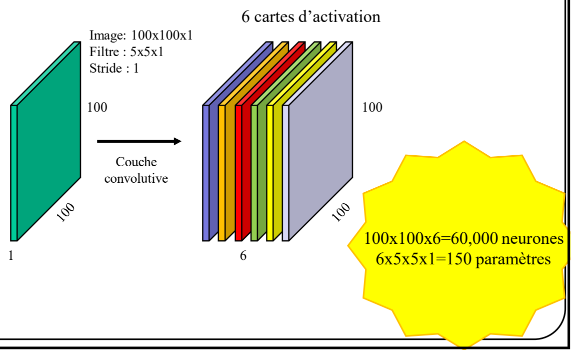
74

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



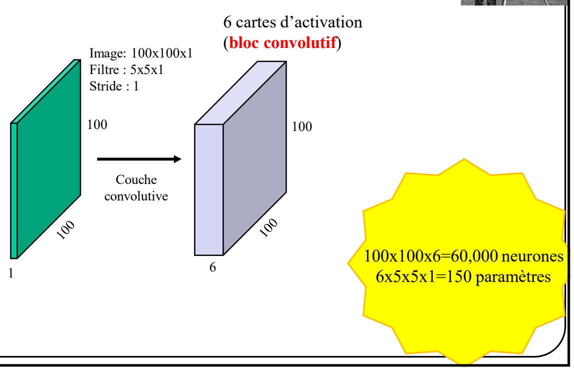
75

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



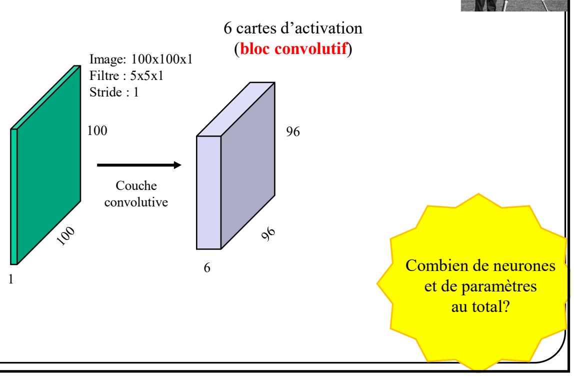
76

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

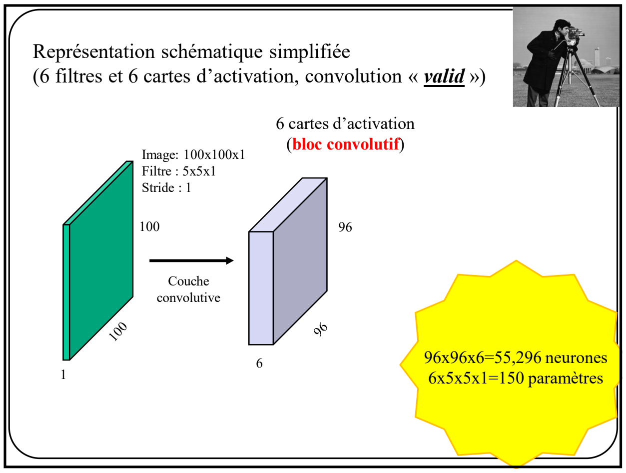


77

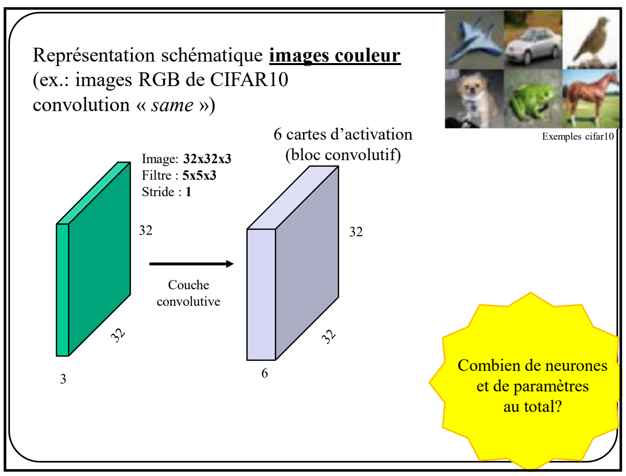
Représentation schématique simplifiée
(6 filtres et 6 cartes d'activation, convolution « ***valid*** »)



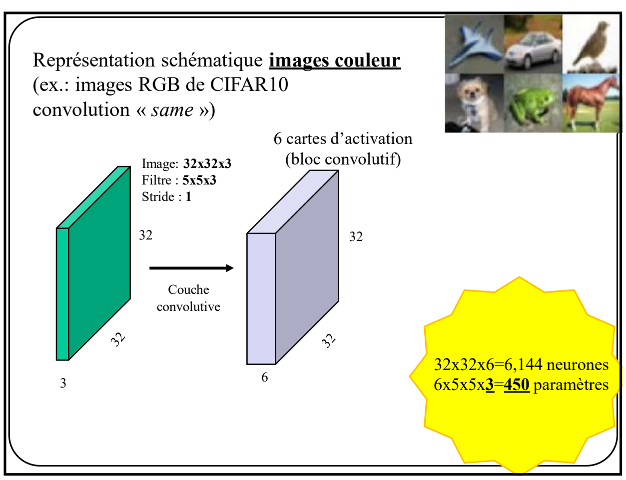
78


Représentation schématique simplifiée
(6 filtres et 6 cartes d'activation, convolution « valid »)
6 cartes d'activation
(bloc convolutif)
Image: 100x100x1
Filtre : 5x5x1
Stride : 1
100
100
1
Couche
convolutive
96
96
6
96x96x6=55,296 neurones
6x5x5x1=150 paramètres

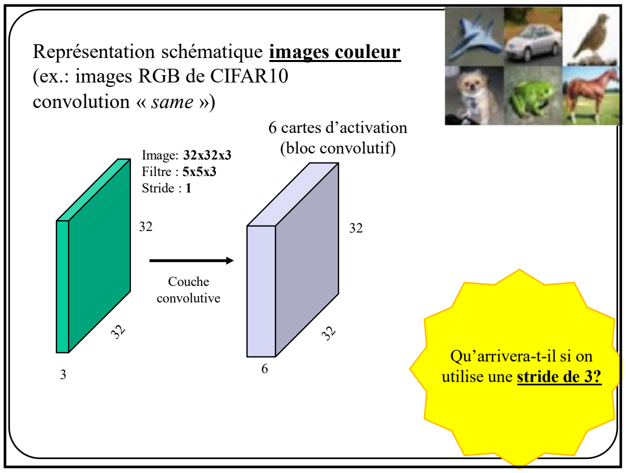
79


Représentation schématique images couleur
(ex.: images RGB de CIFAR10
convolution « same »)
Exemples cifar10
6 cartes d'activation
(bloc convolutif)
Image: 32x32x3
Filtre : 5x5x3
Stride : 1
32
32
3
Couche
convolutive
32
32
6
Combien de neurones
et de paramètres
au total?

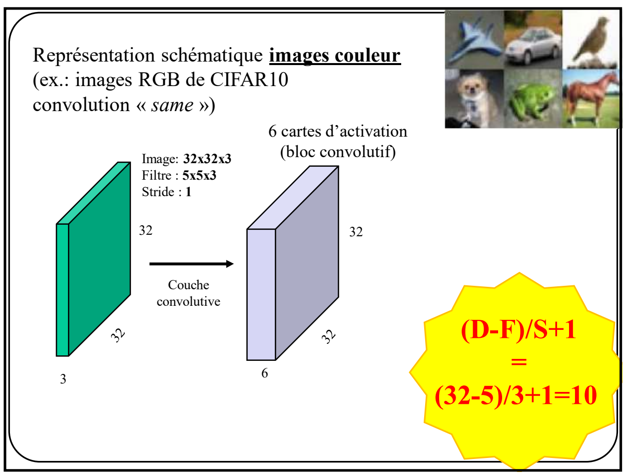
80


Représentation schématique images couleur
(ex.: images RGB de CIFAR10
convolution « same »)
6 cartes d'activation
(bloc convolutif)
Image: 32x32x3
Filtre : 5x5x3
Stride : 1
32
32
3
Couche
convolutive
32
32
6
32x32x6=6,144 neurones
6x5x5x3=450 paramètres

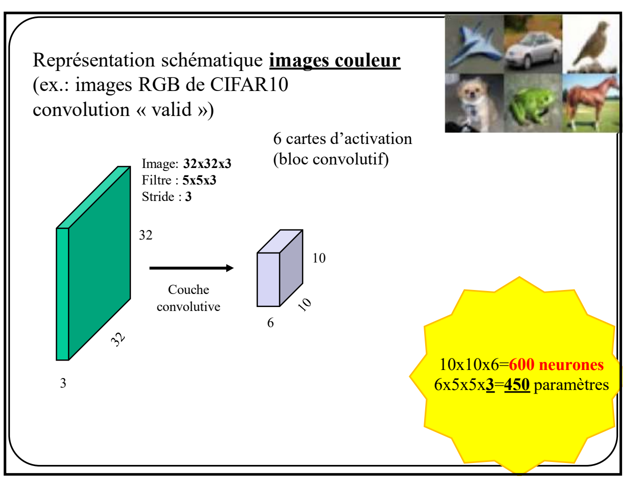
81


Représentation schématique images couleur
(ex.: images RGB de CIFAR10
convolution « same »)
Image: 32x32x3
Filtre : 5x5x3
Stride : 1
32
32
3
Couche
convolutive
6 cartes d'activation
(bloc convolutif)
32
32
6
Qu'arrivera-t-il si on
utilise une stride de 3?

82

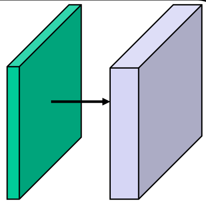

Représentation schématique images couleur
(ex.: images RGB de CIFAR10
convolution « same »)
Image: 32x32x3
Filtre : 5x5x3
Stride : 1
32
32
3
Couche
convolutive
6 cartes d'activation
(bloc convolutif)
32
32
6
(D-F)/S+1
=
(32-5)/3+1=10

83


Représentation schématique images couleur
(ex.: images RGB de CIFAR10
convolution « valid »)
Image: 32x32x3
Filtre : 5x5x3
Stride : 3
32
32
3
Couche
convolutive
6 cartes d'activation
(bloc convolutif)
10
10
6
10x10x6=600 neurones
6x5x5x3=450 paramètres

84

Exemple

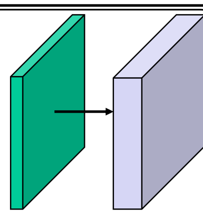


Volume en entrée : **32 x 32 x 3**
10 filtres 5x5 avec **stride = 1**
et convolution « ***same*** »

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

85

Exemple



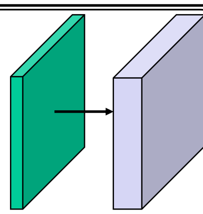
Volume en entrée : **32 x 32 x 3**
10 filtres 5x5 avec **stride = 1**
et convolution « ***same*** »

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

Chaque filtre a **5x5x3** = **75** paramètres
Comme il y a **10 filtres** : **750** paramètres

86

Exemple



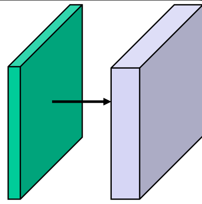
Volume en entrée : **32 x 32 x 3**
10 filtres 5x5 avec **stride = 1**
et convolution « ***same*** ».

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

Chaque filtre a **5x5x3+1** = **76** paramètres (+1 pour le biais)
Comme il y a **10 filtres** : **760** paramètres

87

Exemple

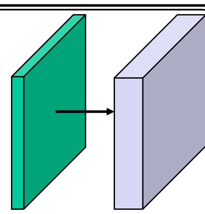


Volume en entrée : **32 x 32 x 3**
10 filtres 5x5 avec **stride = 1**
et convolution « ***valid*** »

Combien de paramètres dans cette couche?

88

Exemple



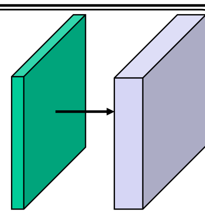
Volume en entrée : **32 x 32 x 3**
10 filtres 5x5 avec **stride = 1**
et convolution « ***valid*** »

Combien de paramètres dans cette couche?

Même chose, cela ne change pas la conformité des filtres

89

Exemple

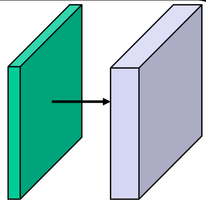


Volume en entrée : **32 x 32 x 3**
10 filtres 5x5 avec **stride = 1**
et convolution « ***valid*** »

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

90

Exemple



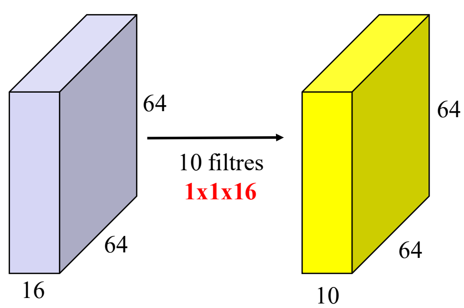
Volume en entrée : **32 x 32 x 3**
10 filtres 5x5 avec **stride = 1**
et convolution « ***valid*** »

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

(32-5+1) x (32-5+1) x 10 = 7,840

91

Des filtres 1x1? Oui ça marche



92

Exemple simple d'un filtre 1x1

$\left[\frac{1}{3}, \frac{1}{3}, \frac{1}{3}\right]$

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

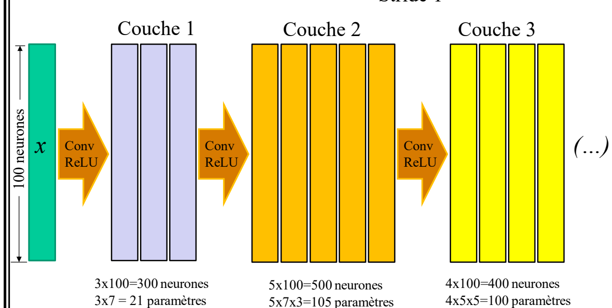
93

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

94

Exemple : 3 filtres couche 1 : taille 7
5 filtres couche 2 : taille 7
4 filtres couche 3 : taille 5

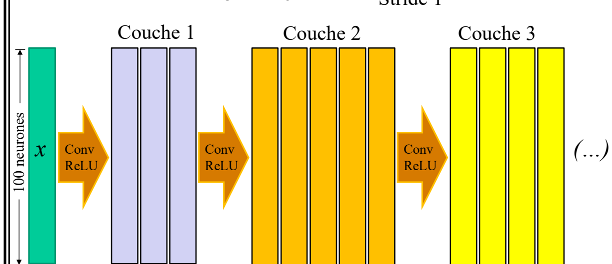
Convolution « *same* »
Stride 1



95

Exemple : 3 filtres couche 1 : taille 7
5 filtres couche 2 : taille 7
4 filtres couche 3 : taille 5

Convolution « *same* »
Stride 1



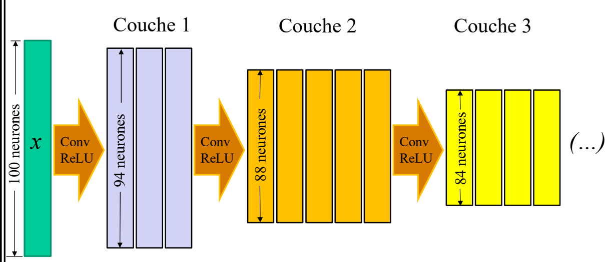
Pour ces 3 couches convolutionnelles

- 1200 neurones au total
- 226 paramètres au total

96

Exemple : 3 filtres couche 1 : taille 7
5 filtres couche 2 : taille 7
4 filtres couche 3 : taille 5

Convolution « ***valid*** »
Stride 1



3x(100-7+1)=282 neurones
3x7 = 21 paramètres

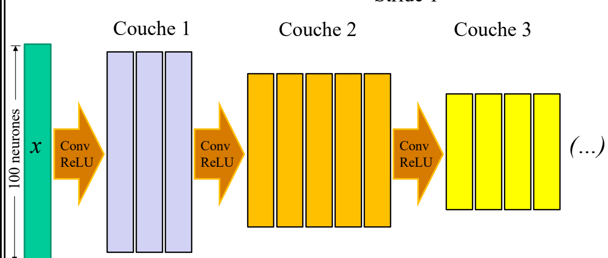
5x(94-7+1)=440 neurones
5x7x3 = 75 paramètres

4x(88-5+1)=336 neurones
4x5x5 = 100 paramètres

97

Exemple : 3 filtres couche 1 : taille 7
5 filtres couche 2 : taille 7
4 filtres couche 3 : taille 5

Convolution « ***valid*** »
Stride 1



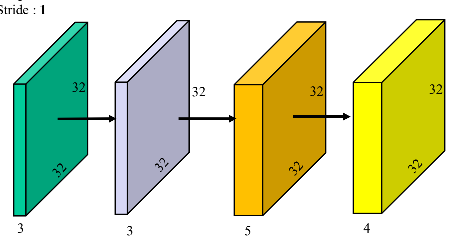
Pour ces 3 couches convolutionnelles

- 1066 neurones au total
- 226 paramètres au total

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**



3x(32x32)=3,072 neurones
3x7x7x3 = 441 paramètres

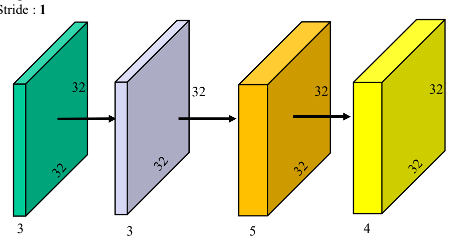
5x(32x32)=5,120 neurones
5x9x9x3 = 1,215 paramètres

4x(32x32)=4,096 neurones
4x11x11x5 = 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**

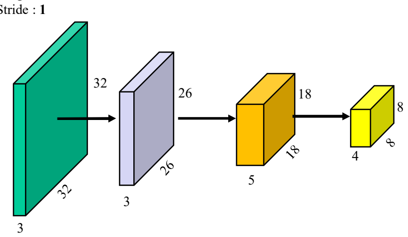


12,288 neurones au total
4,076 paramètres au total

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**



3x(26x26)=2,028 neurones
3x7x7x3 = 441 paramètres

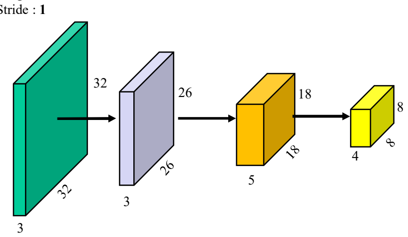
5x(18x18)=1,620 neurones
5x9x9x3 = 1,215 paramètres

4x(8x8)=256 neurones
4x11x11x5 = 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

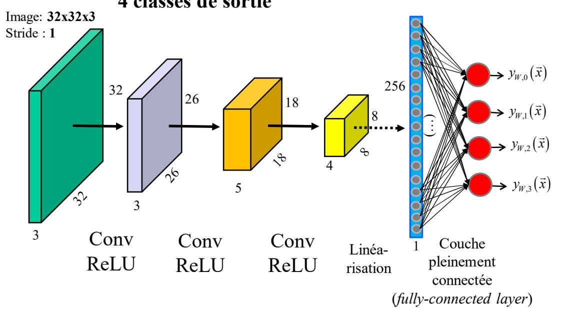
102

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

103

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* »
4 classes de sortie

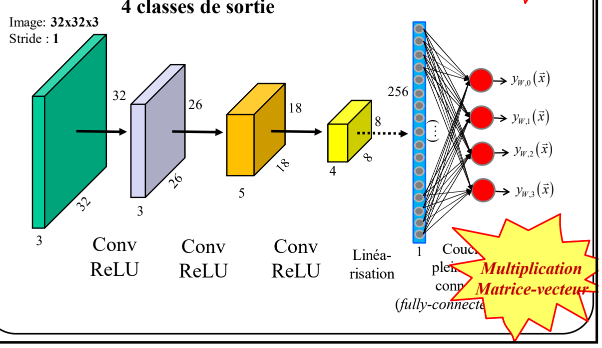
Hinge loss



104

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* »
4 classes de sortie

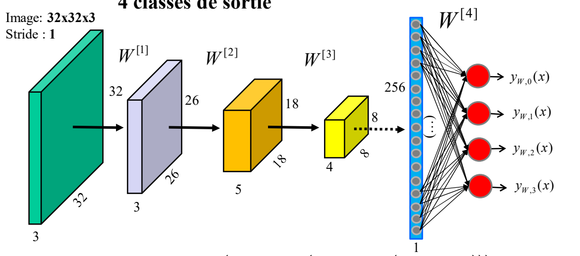
Hinge loss



105

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* »
4 classes de sortie

Hinge loss



$$y_W(x) = W^{[4]} h\Big(W^{[3]} * h\Big(W^{[2]} * h\Big(W^{[1]} * x\Big)\Big)\Big)$$

106

$$y_W(x) = W^{[4]} h\Big(W^{[3]} * h\Big(W^{[2]} * h\Big(W^{[1]} * x\Big)\Big)\Big)$$

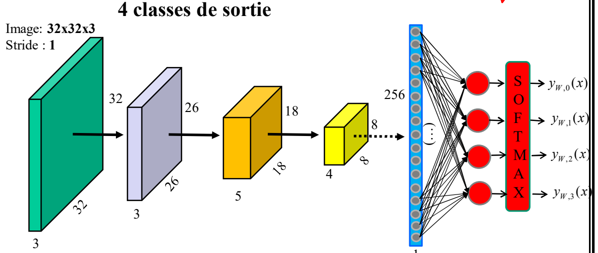
Filtres convolutifs

Matrice 4x256

107

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* »
4 classes de sortie

Cross-entropy loss



Nombre total de neurones?

108

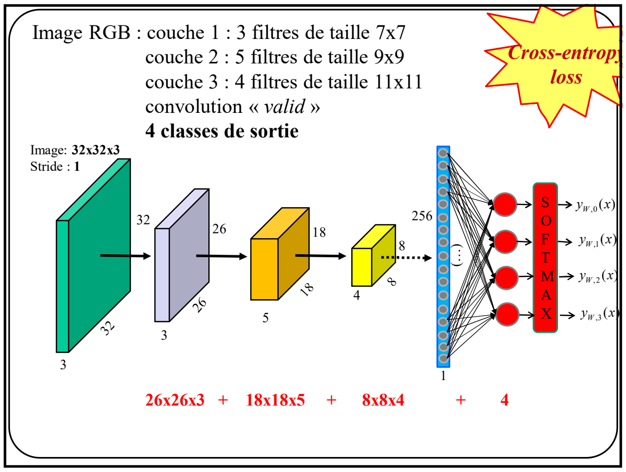


Image RGB : couche 1 : 3 filtres de taille 7x7
couche 2 : 5 filtres de taille 9x9
couche 3 : 4 filtres de taille 11x11
convolution « *valid* »
4 classes de sortie
Cross-entropy loss
Image: 32x32x3
Stride : 1
SOFTMAX
26x26x3 + 18x18x5 + 8x8x4 + 4

109

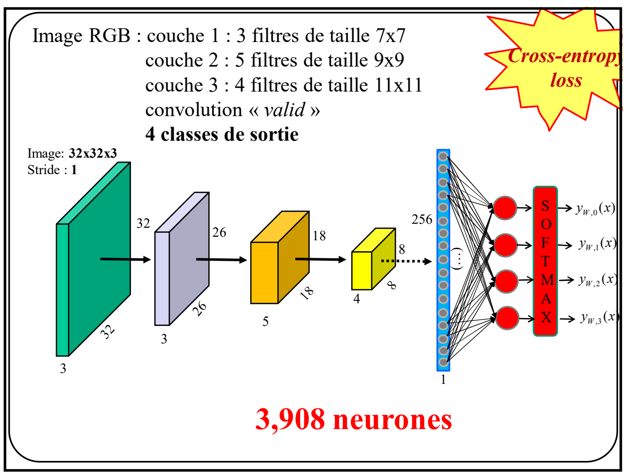


Image RGB : couche 1 : 3 filtres de taille 7x7
couche 2 : 5 filtres de taille 9x9
couche 3 : 4 filtres de taille 11x11
convolution « *valid* »
4 classes de sortie
Cross-entropy loss
Image: 32x32x3
Stride : 1
SOFTMAX
3,908 neurones

110

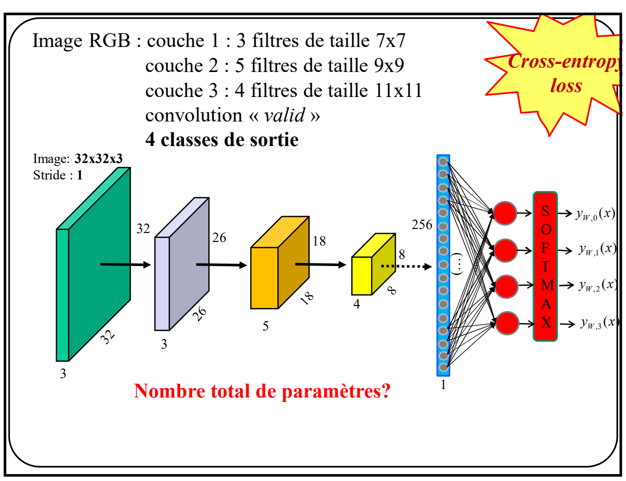
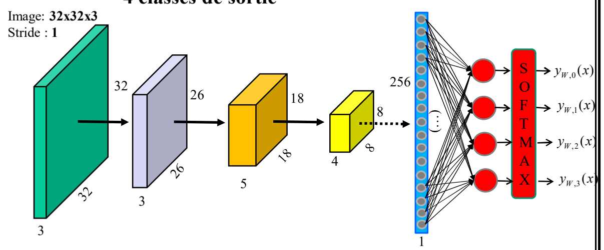


Image RGB : couche 1 : 3 filtres de taille 7x7
couche 2 : 5 filtres de taille 9x9
couche 3 : 4 filtres de taille 11x11
convolution « *valid* »
4 classes de sortie
Cross-entropy loss
Image: 32x32x3
Stride : 1
SOFTMAX
Nombre total de paramètres?

111

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* »
4 classes de sortie

Cross-entropy loss

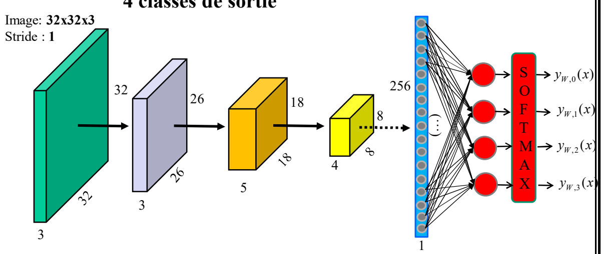


3x7x7x3 + 5x9x9x3 + 4x11x11x5 + 256x4

112

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* »
4 classes de sortie

Cross-entropy loss

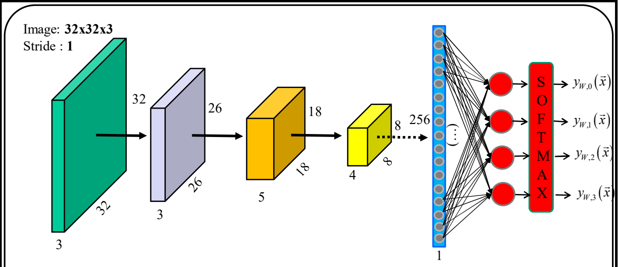


5,100 paramètres

113

Réseaux à convolution
vs
Réseaux **pleinement** convolutifs

114



$$y_W(x) = W^{[4]} h\left(W^{[3]} * h\left(W^{[2]} * h\left(W^{[1]} * x\right)\right)\right)$$

Il est possible de **remplacer la multiplication matrice-vecteur** de la fin par une **convolution**

115

Exemple 1d
(convolution « valid »)

$$\overbrace{\boxed{40\,|\,50\,|\,70\,|\,80\,|\,90\,|\,10\,|\,20\,|\,30\,|\,40}}^{9} * \overbrace{\boxed{.2\,|\,-.3\,|\,.4}}^{3} = \overbrace{\boxed{21\,|\,21\,|\,26\,|\,-7\,|\,23\,|\,8\,|\,11}}^{7}$$

116

Exemple 1d
(convolution « valid »)

$$\overbrace{\boxed{40\,|\,50\,|\,70\,|\,80\,|\,90\,|\,10\,|\,20\,|\,30\,|\,40}}^{9} * \overbrace{\boxed{.2\,|\,-.3\,|\,.4\,|\,-.5\,|\,.6}}^{5} = \overbrace{\boxed{35\,|\,-18\,|\,33\,|\,1\,|\,32}}^{5}$$

117

Exemple 1d
(convolution « valid »)

9										7								3		
40	50	70	80	90	10	20	30	40	*	.2	-.3	.4	-.5	.6	-.7	.8	=	44	-8	44

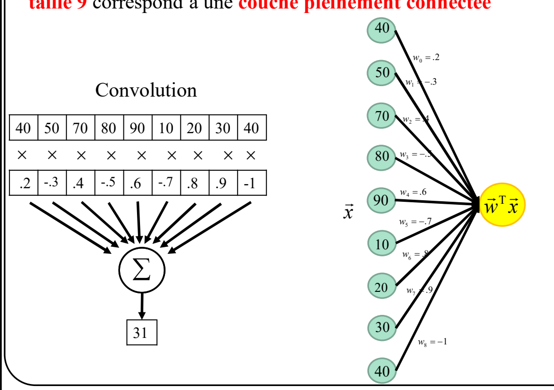
118

Taille filtre = nb de neurones couche précédente

9										9										1
40	50	70	80	90	10	20	30	40	*	.2	-.3	.4	-.5	.6	-.7	.8	.9	-1	=	31

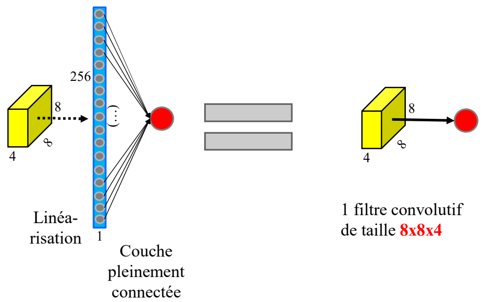
119

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



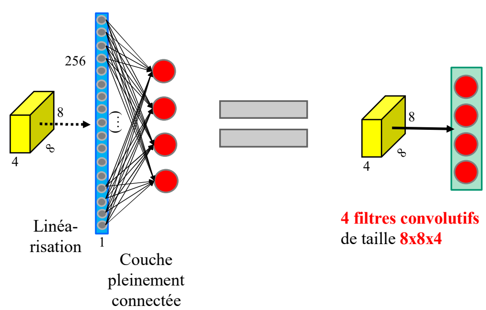
120

Même chose pour une **convolution 2D**

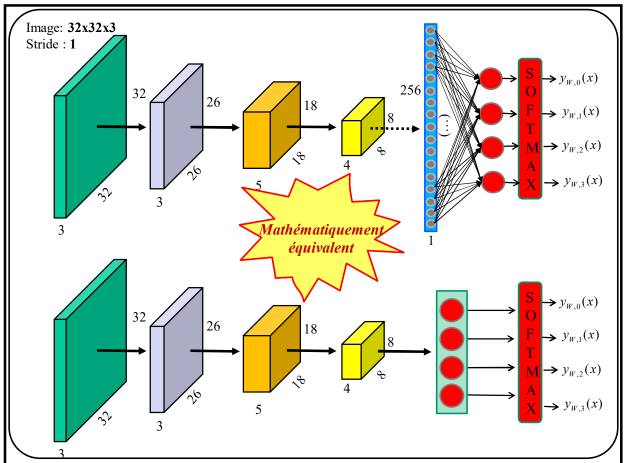


121

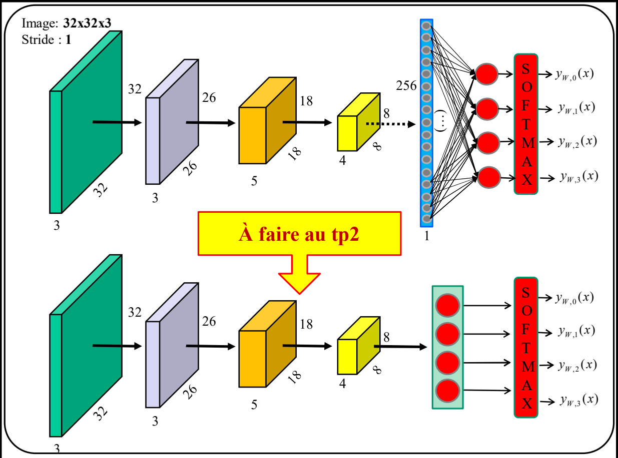
Même chose pour une **convolution 2D**



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 …

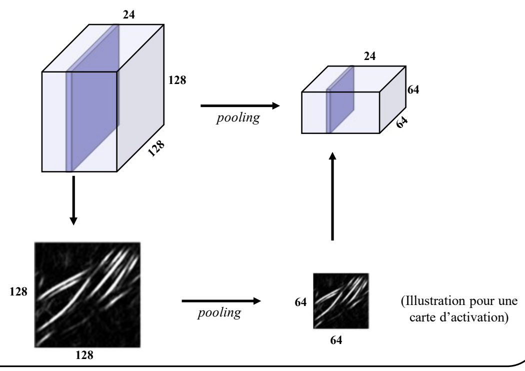
Question : qu'arrive-t-il si on remplace l'image 32x32x3 par une image 64x64x3?

125

Pooling

126

Réduction de la taille des cartes d’activation



127

Max pooling

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

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

128

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

129

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**

?

130

Max pooling

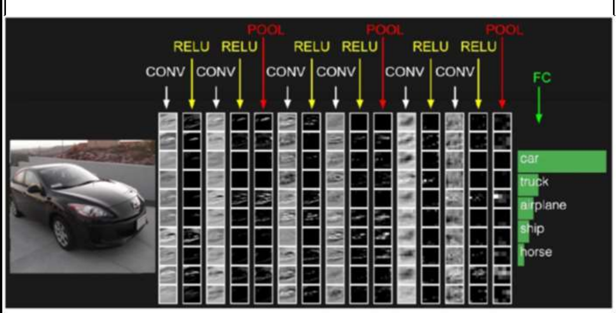
1	2	4	4	9	3	1	2
6	7	8	4	-3	-3	6	3
9	-9	8	-4	5	5	3	0
8	-8	9	-9	5	5	0	1
0	0	1	2	7	9	7	8
-1	-3	3	6	8	8	7	6
9	9	8	2	1	5	-1	-1
1	1	-2	8	3	7	4	-2

→ *Max pooling* **3x3**
avec **stride =2**

?

131

Illustration d'un CNN complet



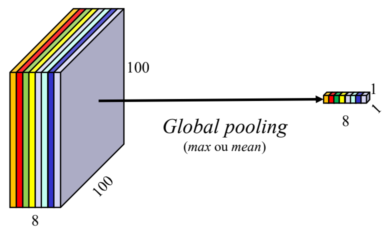
Crédit : cs231 Stanford

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

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

=

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

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

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

Y0	Y1
Y2	Y3

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

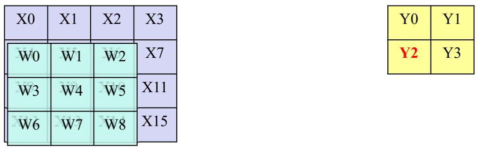
Y0	**Y1**
Y2	Y3

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

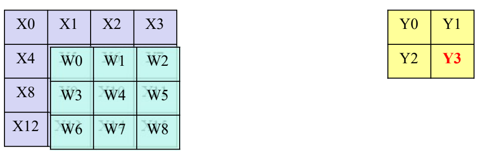


Y2=W0.X4+W1.X5+W2.X6+W3.X8+W4.X9+W5.X10+W6.X12+W7.X13+W8.X14

139

Rappel

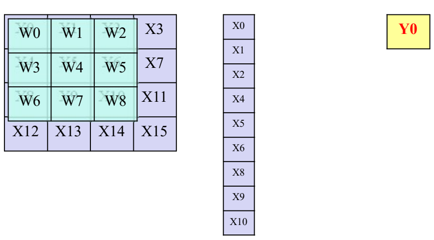
Ex.: convolution « *valid* », un canal d'entrée et une carte d'activation, filtre 3x3



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

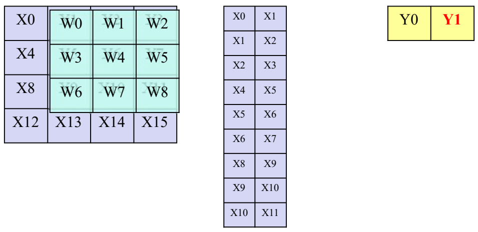
140

Autrement dit…



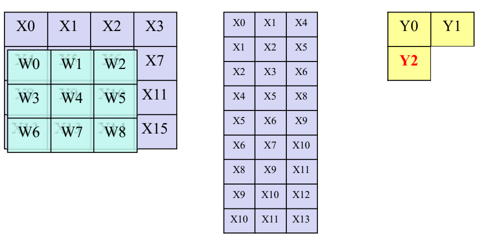
141

Autrement dit…



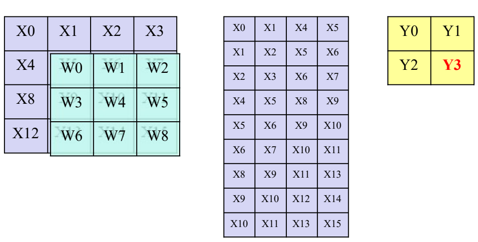
142

Autrement dit…



143

Autrement dit…



144

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

W0	W1	W2	W3	W4	W5	W6	W7	W8

x

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

145

Autre exemple

conv « valid », mini-batch de 2 entrées

2 données en 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	X21

*

Filtre

W0	W1	W2
W3	W4	W5
W6	W7	W8

=

Y0	Y1
Y2	Y3

Y4	Y5
Y6	Y7

146

Autre exemple

conv « valid », mini-batch de 2 entrées

W0	W1	W2	W3	W4	W5	W6	W7	W8

x

X16	X17	X20	X21

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
Y4	Y5	Y6	Y7

147

Autre exemple

conv « valid », une entrée, deux filtres et 2 *features maps* en sortie

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

148

Autre exemple

conv « valid », une entrée, deux filtres et 2 *features maps* en sortie

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

x

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
Y4	Y5	Y6	Y7

149

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

150

Autre exemple

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

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

X

X0	X1	X4	X5
X1	X2	X5	X6
X2	X3	X6	X7
X4	X5	X8	X9
(…)	(…)	(…)	(…)
X22	X23	X26	X27
X24	X25	X27	X29
X25	X26	X28	X30
X26	X27	X29	X31

=

Y0	Y1	Y2	Y3

151

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

Exercice à la maison, voir comment cette 2e 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**.

152

Comment calculer la rétropropagation dans un CNN?

À faire au TP2

153