Réseaux de neurones

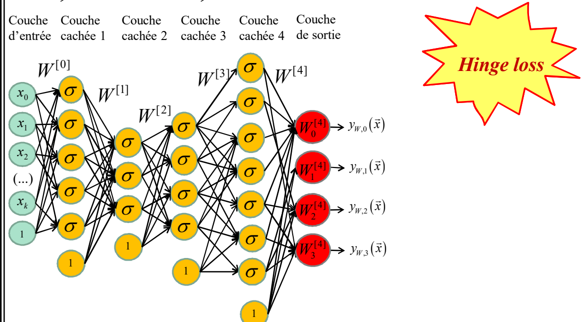
IFT 780

Réseaux à convolution

Par

Pierre-Marc Jodoin

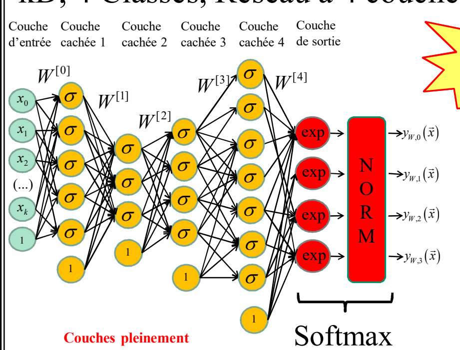
kD, 4 Classes, Réseau à 4 couches cachées



Couches pleinement connectées ***(fully-connected layers)***

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

kD, 4 Classes, Réseau à 4 couches cachées



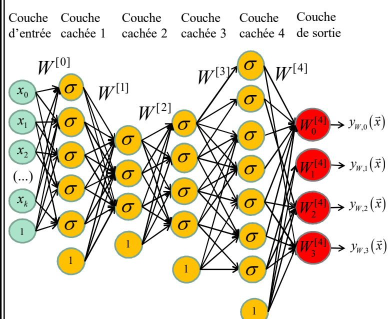
Softmax

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

Couches pleinement connectées ***(fully-connected layers)***

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

kD, 4 Classes, Réseau à 4 couches cachées

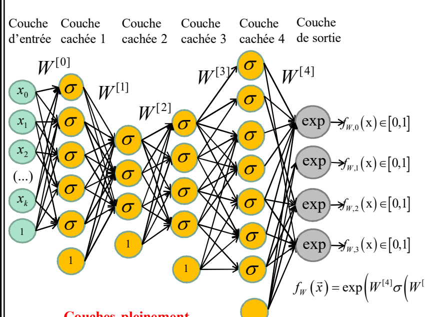


$$W^{[0]} \in R^{5\times k+1}$$
$$W^{[1]} \in R^{3\times 6}$$
$$W^{[2]} \in R^{4\times 4}$$
$$W^{[3]} \in R^{5\times 7}$$
$$W^{[4]} \in R^{4\times 8}$$

Couches pleinement connectées ***(fully-connected layers)***

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

kD, 4 Classes, Réseau à 4 couches cachées



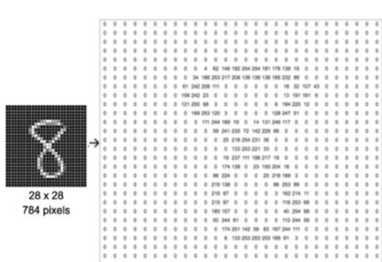
$$W^{[0]} \in R^{5\times k+1}$$
$$W^{[1]} \in R^{3\times 6}$$
$$W^{[2]} \in R^{4\times 4}$$
$$W^{[3]} \in R^{5\times 7}$$
$$W^{[4]} \in R^{4\times 8}$$

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

Couches pleinement connectées ***(fully-connected layers)***

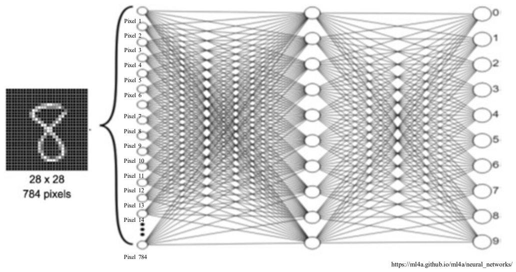
Softmax $y_{W,i}(\mathrm{x}) = \dfrac{f_{W,i}}{\sum_k f_{W,k}}$

Comment classifier des images?

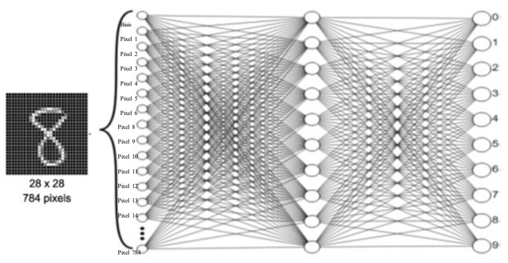


https://ml4a.github.io/ml4a/neural_networks/

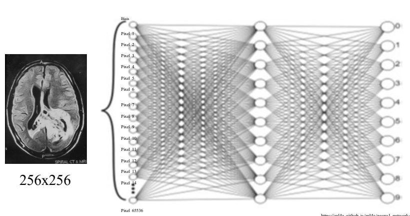
Comment classifier des images?



Beaucoup de paramètres (7850 dans la couche 1)



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



256x256

Image médicale (IRM de cerveau)

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

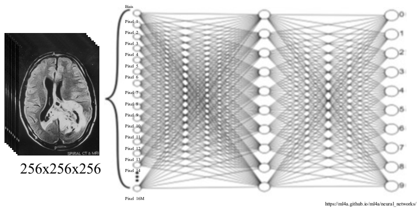


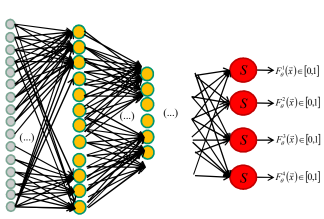
Image médicale 3D (IRM de cerveau)

Comment réduire le nombre de connections?

11

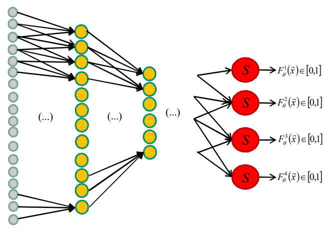
Comment réduire le nombre de connections?

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



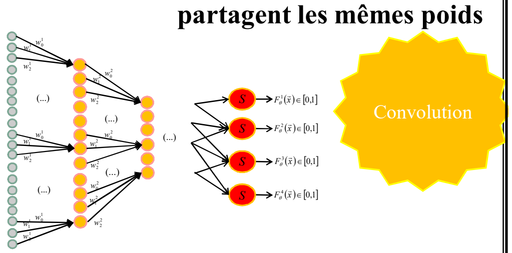
150-D en entrée avec 150 neurones dans la 1ère couche => **22,200 parametres dans la couche d'entrée!!**

Solution : connexions partielles



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

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!

Convolution
et
couche convolutionnelle
1D

Exemple 1D de la convolution

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

(signal d'entrée) f(u)	(filtre) W(u)	(filtre) W(-u)
10 20 -30 40 -50	.1 .2 .3	.3 .2 .1

(f*W)(1)	(f*W)(2)	(f*W)(3)
10 20 -30 40 -50	10 20 -30 40 -50	10 20 -30 40 -50
× × ×	× × ×	× × ×
.3 .2 .1	.3 .2 .1	.3 .2 .1
3+4+-3	6-6+4	-9+8+-5
4	4 4	4 4 -6

16

En gros

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

La convolution des réseaux de neurones = corrélation

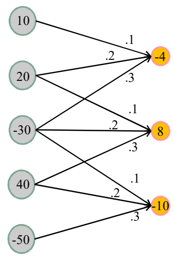
$$(f*W)(v)=\sum_{u=-\infty}^{\infty} f(u)W(v+u)$$

(signal d'entrée) f(u)	(filtre) W(u)	(filtre) W(+u)
10 20 -30 40 -50	.1 .2 .3	.1 .2 .3

(f*W)(1)	(f*W)(2)	(f*W)(3)
10 20 -30 40 -50	10 20 -30 40 -50	10 20 -30 40 -50
× × ×	× × ×	× × ×
.1 .2 .3	.1 .2 .3	.1 .2 .3
1+4-9	2-6+12	-3+8-15
-4	-4 8	-4 8 -10

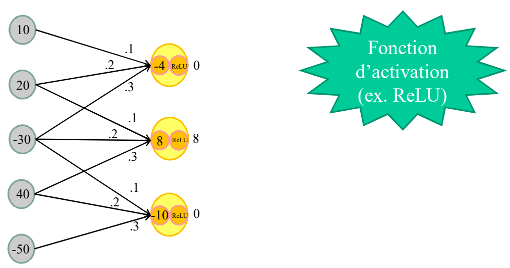
18

L’opération de la page précédente est équivalente à



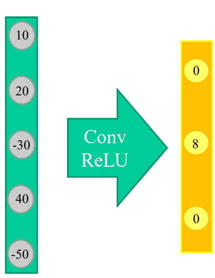
19

L’opération de la page précédente est équivalente à



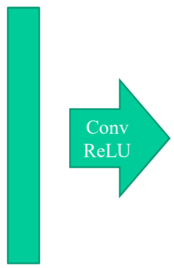
20

Représentation graphique courante (simple)

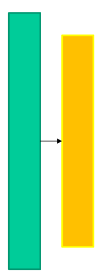


21

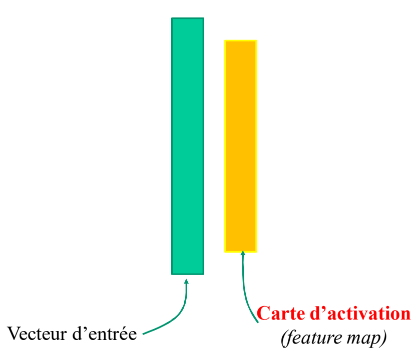
Représentation graphique courante (encore plus simple)



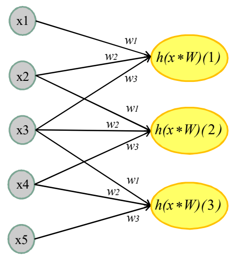
Représentation graphique courante (vraiment ultra simple)



Représentation graphique courante (eehhh…)



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



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

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

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

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

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

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

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

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

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 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

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

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

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

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

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

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

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

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

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

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

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

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

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

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* »)

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

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* »)

De loin l'option la plus utilisée

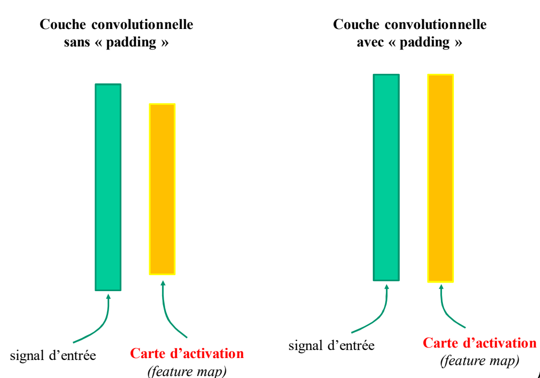
f(u)

10	10	20	-30	40	-50	-50

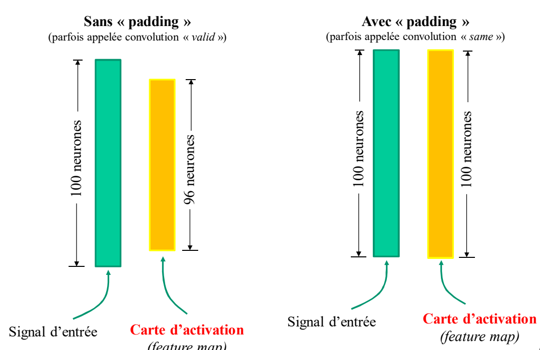
(f*W)(u)

9	-4	8	-10	-21

49

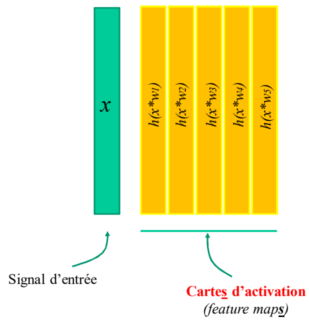


Exemple : taille de filtre = 5, stride=1



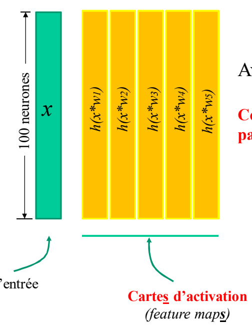
Il est possible d'apprendre **plusieurs filtres par couche**

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



Taille de filtre = 5

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

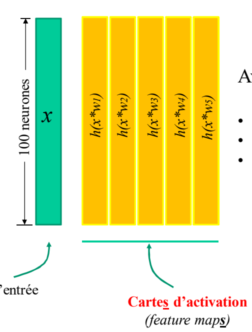


Avec « *padding* »

Combien de neurones et de paramètres au total?

Taille de filtre = 5

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



Avec « *padding* »:

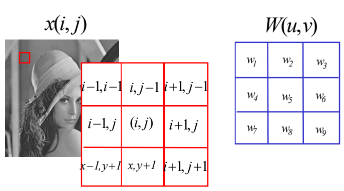
- 100 neurones par carte d'activation
- 500 neurones au total
- 25 (5x5) paramètres au total

Convolution et couche convolutionnelle 2D

Filtage 2D

(sans flip de filtre)

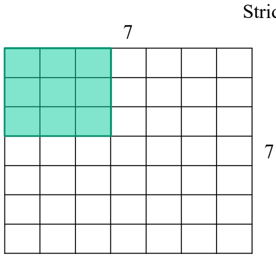
$$(x*W)(i,j)=\sum_u\sum_v f(i+u,j+v)W(u,v)$$



$$\begin{aligned}(x*W)(i,j)&=w_1x(i-1,j-1)+w_2x(i,j-1)+w_3x(i+1,j-1)\\&+w_4x(i-1,j)\quad+w_5x(i,j)\quad+w_6x(i+1,j)\\&+w_7x(i-1,j+1)+w_8x(i,j+1)+w_9x(i+1,j+1)\end{aligned}$$

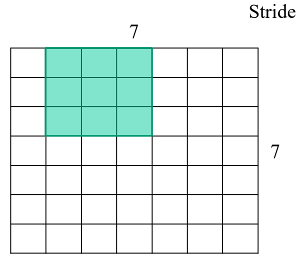
Convolution 2D

Filtre = 3x3
Stride = 1



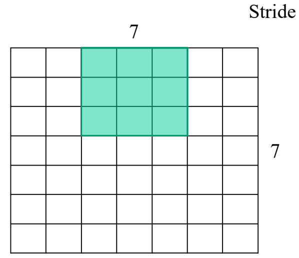
Convolution 2D

Filtre = 3x3
Stride = 1



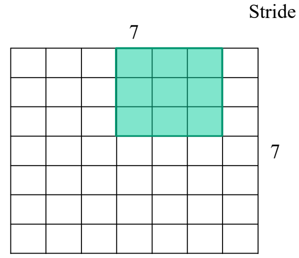
Convolution 2D

Filtre = 3x3
Stride = 1



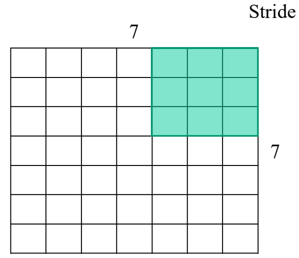
Convolution 2D

Filtre = 3x3
Stride = 1



Convolution 2D

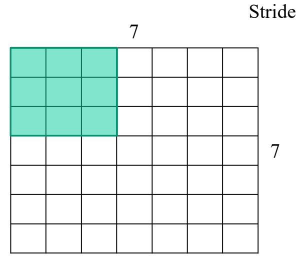
Filtre = 3x3
Stride = 1



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

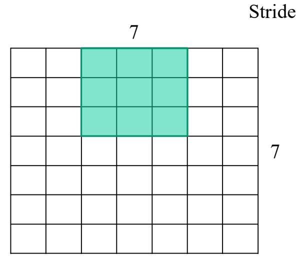
Convolution 2D

Filtre = 3x3
Stride = 2



Convolution 2D

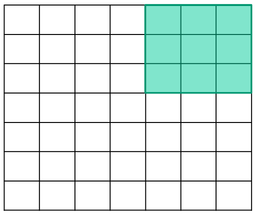
Filtre = 3x3
Stride = 2



Convolution 2D

Filtre = 3x3
Stride = 2

7



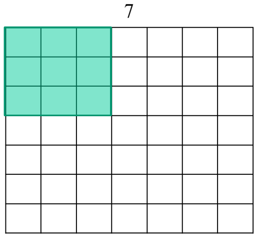
7

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

Convolution 2D

Filtre = 3x3
Stride = 3

7

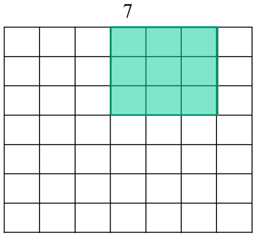


7

Convolution 2D

Filtre = 3x3
Stride = 3

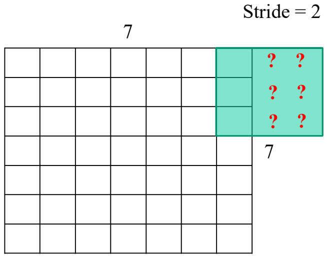
7



7

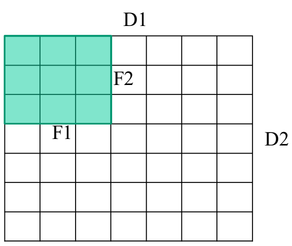
Convolution 2D

Filtre = 3x3
Stride = 2



Combinaison D-F-S invalide!

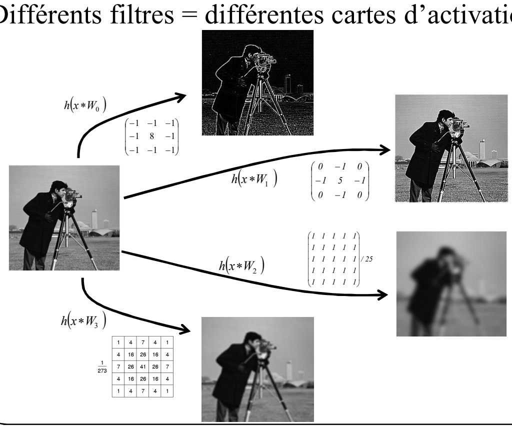
Convolution 2D



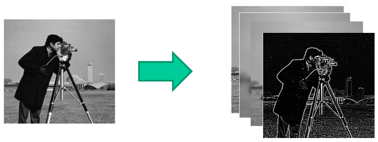
Taille de la carte d'activation :

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

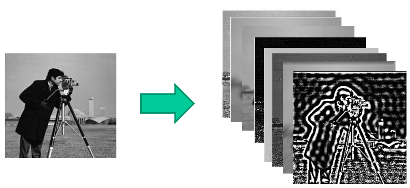
Différents filtres = différentes cartes d'activation



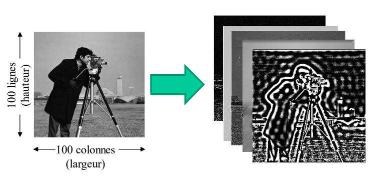
4 filtres = Couche convolutive avec **4 cartes d'activation**



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

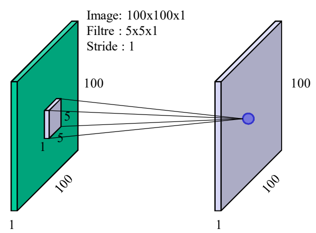


Ex.: taille de filtre : 5x5, 5 cartes d'activation, convolution « same »

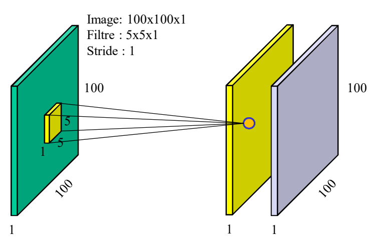


- 10,000 neurones par carte d'activation
- 50,000 neurones au total
- 5x5x5 = 125 paramètres au total

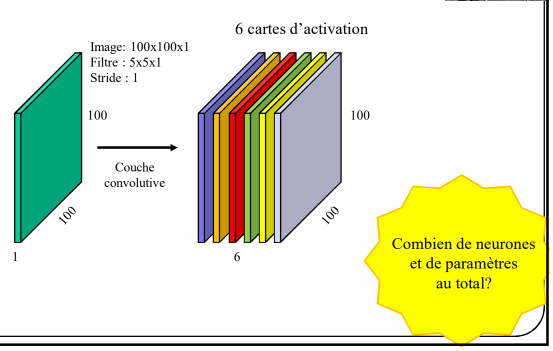
Représentation schématique
(1 filtre et 1 carte d'activation, convolution « same »)



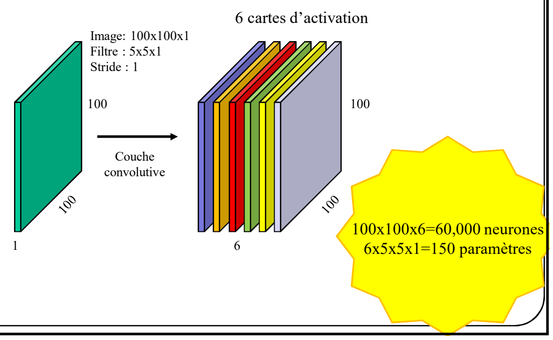
Représentation schématique
(2 filtres et 2 cartes d'activation, convolution « *same* »)



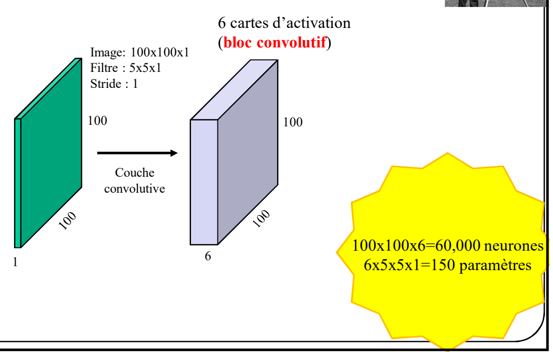
Représentation schématique
(6 filtres et 6 cartes d'activation, convolution « same »)



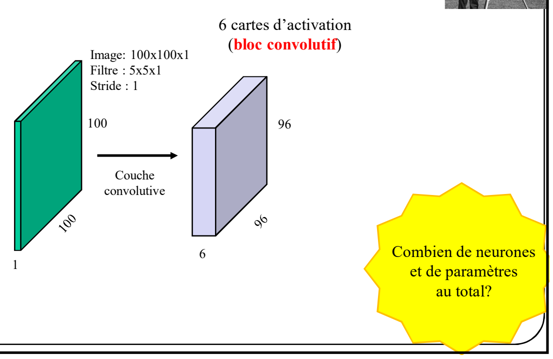
Représentation schématique
(6 filtres et 6 carte d'activation, convolution « *same* »)



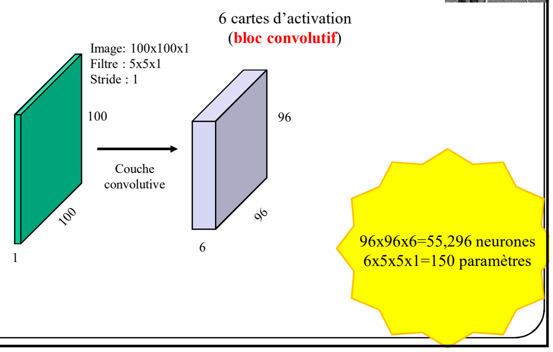
Représentation schématique **simplifiée**
(6 filtres et 6 carte d'activation, convolution « *same* »)



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

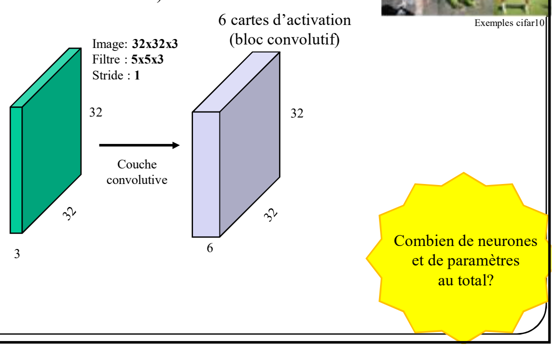


Représentation schématique simplifiée
(6 filtres et 6 cartes d'activation, convolution « ***valid*** »)

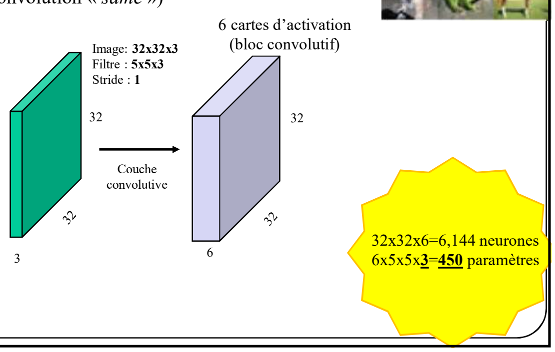


Représentation schématique **images couleurs**
(ex.: images RGB de CIFAR10
convolution « *same* »)

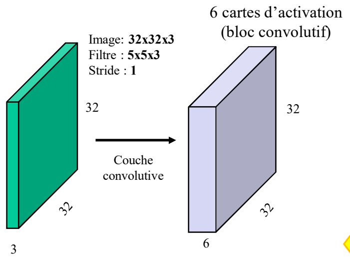
Exemples cifar10



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

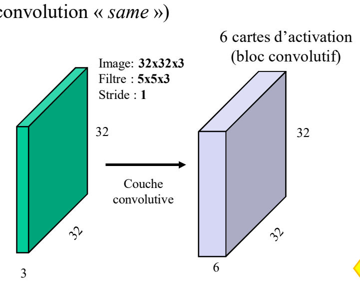


Représentation schématique **images couleurs** (ex.: images RGB de CIFAR10 convolution « *same* »)



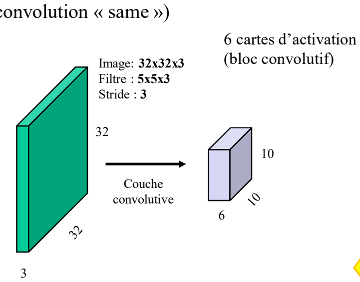
Qu'arrivera-t-il si on utilise une **stride de 3?**

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



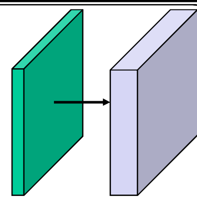
(D-F)/S+1
=
(32-5)/3+1=10

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



10x10x6=**600 neurones**
6x5x5x**3**=**450** paramètres

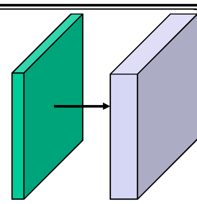
Exemple



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

Combien de paramètres dans cette couche?

Exemple

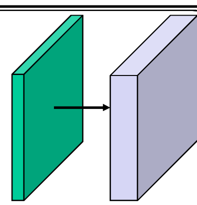


Volume en entrée : **32 x 32 x 3**
10 filtres 5x5 avec **stride = 1**
et convolution **«** ***same*** **»**

Combien de paramètres dans cette couche?

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

Exemple

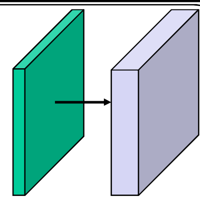


Volume en entrée : **32 x 32 x 3**
10 filtres 5x5 avec **stride = 1**
et convolution **«** ***same*** **»**

Combien de paramètres dans cette couche?

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

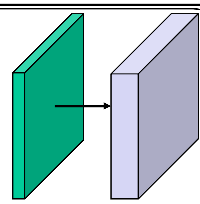
Exemple



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

Combien de paramètres dans cette couche?

Exemple

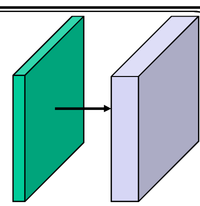


Volume en entrée : **32 x 32 x 3**
10 filtres 5x5 avec **stride = 1**
et convolution « ***valid*** »

Combien de paramètres dans cette couche?

Même chose, cela ne change pas la conformité des filtres

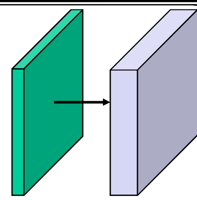
Exemple



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?

Exemple

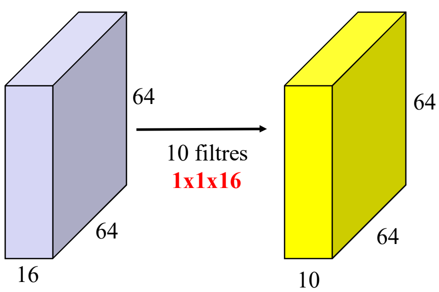


Volume en entrée : **32 x 32 x 3**
10 filtres 5x5 avec **stride = 1**
et convolution « ***valid*** »

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

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

Des filtres 1x1? Oui ça marche



Exemple simple d'un filtre 1x1

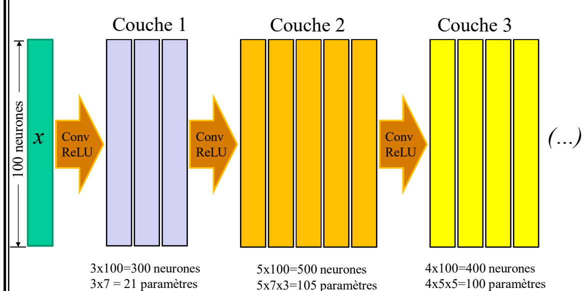
$$\left[\frac{1}{3}, \frac{1}{3}, \frac{1}{3}\right]$$

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

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

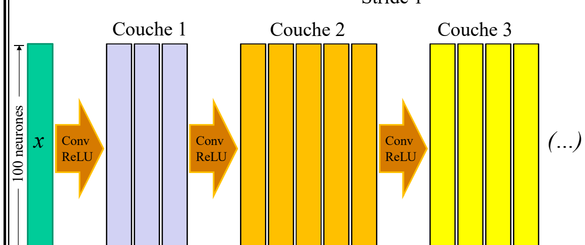
Exemple : 3 filtres couche 1 : taille 7
5 filtres couche 2 : taille 7
4 filtres couche 3 : taille 5

Convolution « *same* »
Stride 1



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

Convolution « *same* »
Stride 1

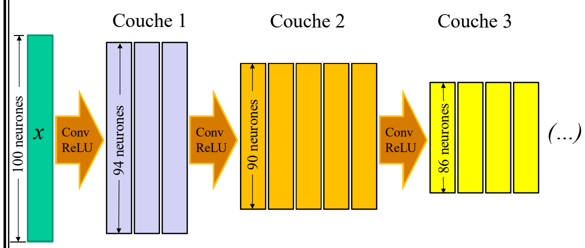


Pour ces 3 couches convolutionnelles

- 1200 neurones au total
- 226 paramètres au total

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

Convolution « ***valid*** »
Stride 1



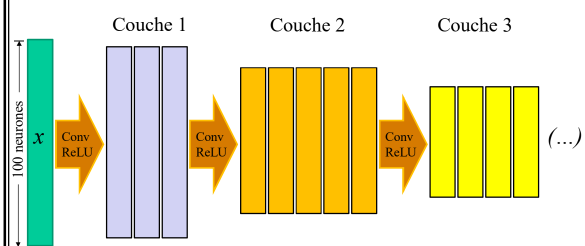
3x(100-7+1)=282 neurones
3x7 = 21 paramètres

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

4x(90-5+1)=344 neurones
4x5x5 = 100 paramètres

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

Convolution « ***valid*** »
Stride 1

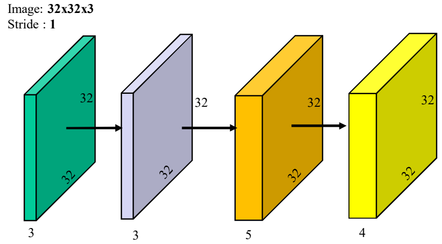


Pour ces 3 couches convolutionnelles

- 1066 neurones au total
- 226 paramètres au total

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**



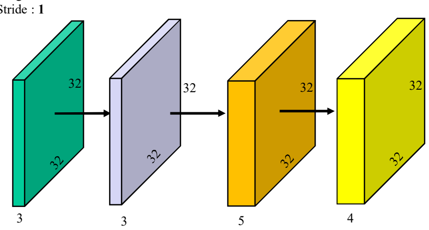
3x(32x32)=3,072 neurones
3x7x7x3 = 441 paramètres

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

4x(32x32)=4,096 neurones
4x11x11x5 = 2,420 paramètres

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* »

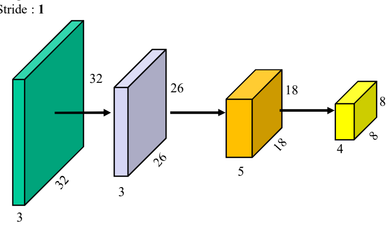
Image: **32x32x3**
Stride : **1**



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

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**



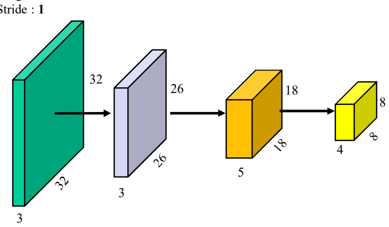
3x(26x26)=2,028 neurones
3x7x7x3 = 441 paramètres

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

4x(8x8)=256 neurones
4x11x11x5 = 2,420 paramètres

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

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

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

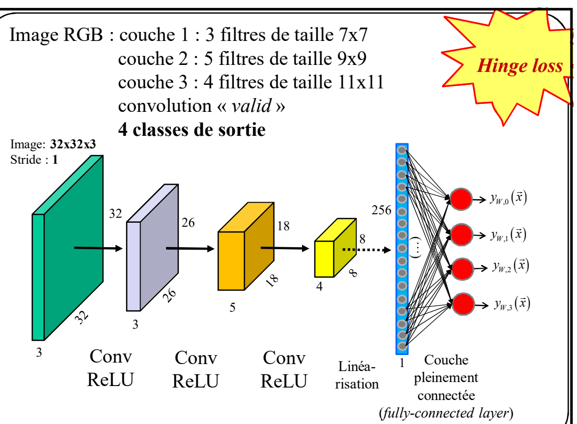
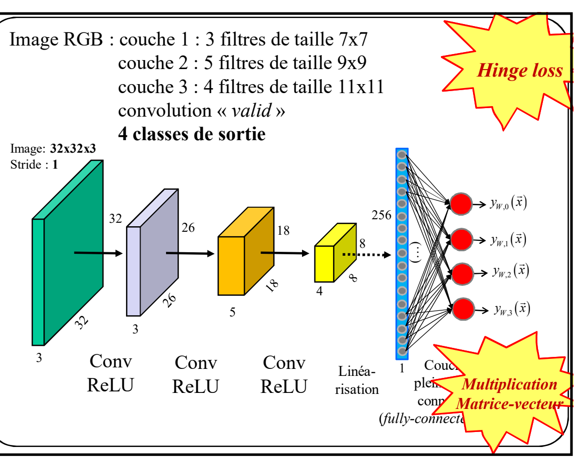
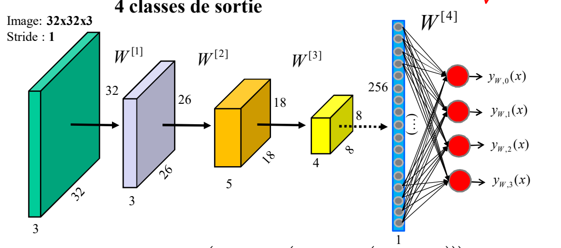


Image RGB : couche 1 : 3 filtres de taille 7x7
couche 2 : 5 filtres de taille 9x9
couche 3 : 4 filtres de taille 11x11
convolution « *valid* »
4 classes de sortie



Hinge loss

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



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

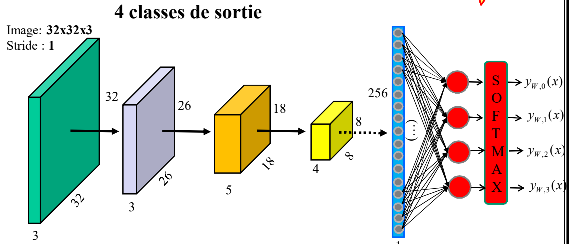
$$y_W(x) = W^{[4]}\left(W^{[3]} * h\left(W^{[2]} * h\left(W^{[1]} * x\right)\right)\right)$$

Filtres convolutifs

Matrice 4x256

Cross-entropy loss

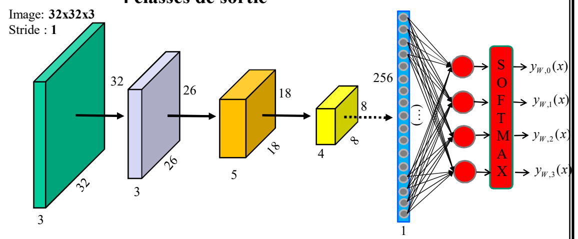
Image RGB : couche 1 : 3 filtres de taille 7x7
couche 2 : 5 filtres de taille 9x9
couche 3 : 4 filtres de taille 11x11
convolution « *valid* »
4 classes de sortie



Nombre total de neurones?

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

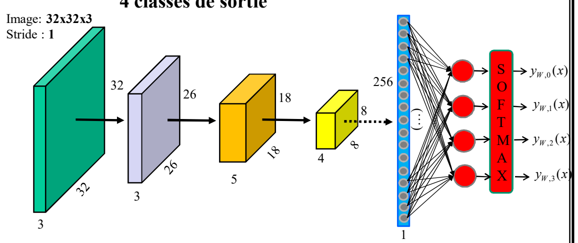
Cross-entropy loss



26x26x3 + 18x18x5 + 8x8x4 + 4

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

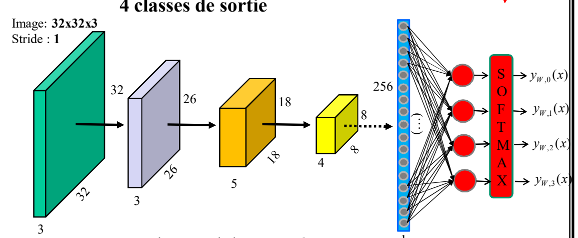
Cross-entropy loss



3,908 neurones

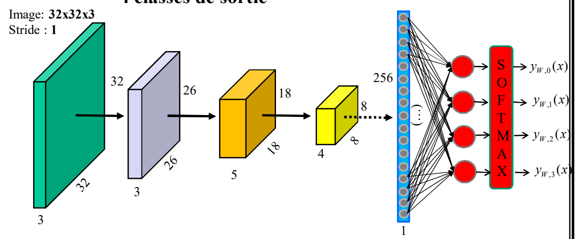
Image RGB : couche 1 : 3 filtres de taille 7x7
couche 2 : 5 filtres de taille 9x9
couche 3 : 4 filtres de taille 11x11
convolution « *valid* »
4 classes de sortie

Cross-entropy loss



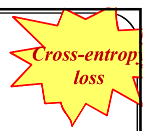
Nombre total de paramètres?

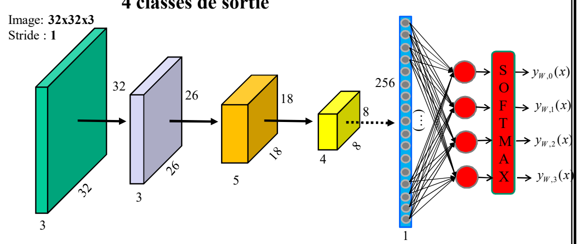
Image RGB : couche 1 : 3 filtres de taille 7x7
couche 2 : 5 filtres de taille 9x9
couche 3 : 4 filtres de taille 11x11
convolution « *valid* »
4 classes de sortie



3x7x7x3 + 5x9x9x3 + 4x11x11x5 + 256x4

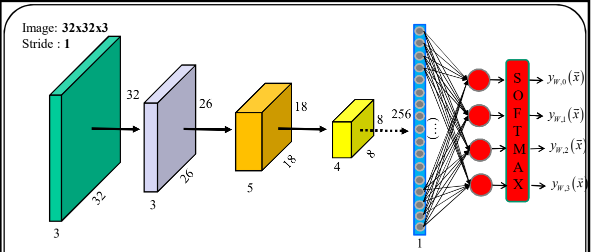
Image RGB : couche 1 : 3 filtres de taille 7x7
couche 2 : 5 filtres de taille 9x9
couche 3 : 4 filtres de taille 11x11
convolution « *valid* »
4 classes de sortie





5,100 paramètres

Réseaux à convolution
vs
Réseaux **pleinement** convolutifs



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

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

Exemple 1d
(convolution « valid »)

9

40	50	70	80	90	10	20	30	40

$*$

3

.2	-.3	.4

$=$

7

21	21	26	-7	23	8	11

Exemple 1d
(convolution « valid »)

9

40	50	70	80	90	10	20	30	40

$*$

5

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

$=$

5

35	-18	33	1	32

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

Taille filtre = nb de neurones couche précédente

9 — 9 — 1

40	50	70	80	90	10	20	30	40

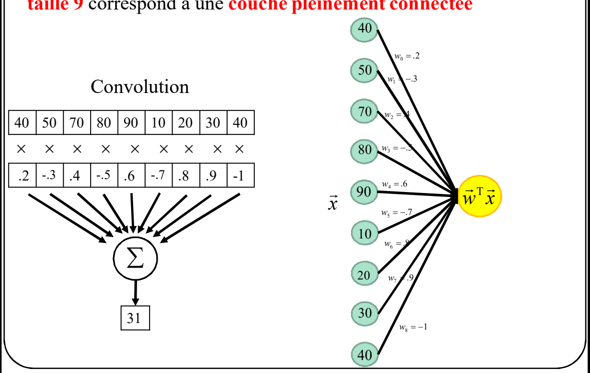
$*$

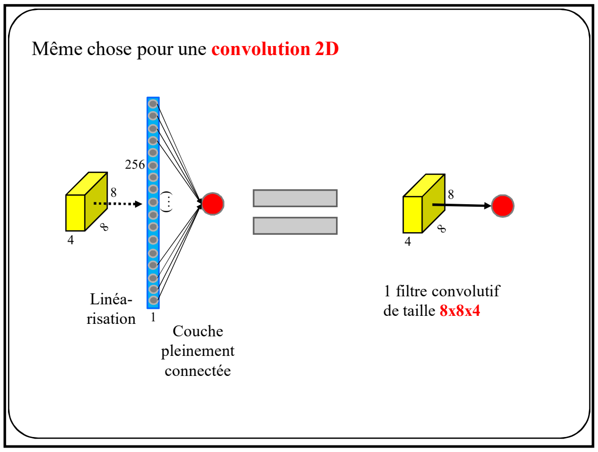
.2	-.3	.4	-.5	.6	-.7	.8	.9	-1

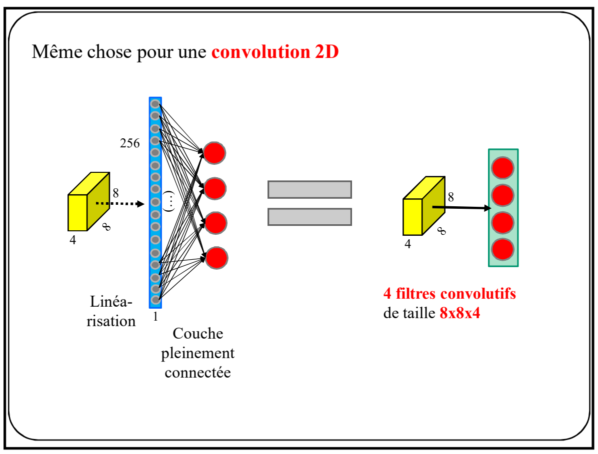
$=$

31

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




Même chose pour une convolution 2D
256
8
(...)
4
8
Linéa-
risation
1
Couche
pleinement
connectée
8
4
8
1 filtre convolutif
de taille 8x8x4


Même chose pour une convolution 2D
256
8
(...)
4
8
Linéa-
risation
1
Couche
pleinement
connectée
8
4
8
4 filtres convolutifs
de taille 8x8x4

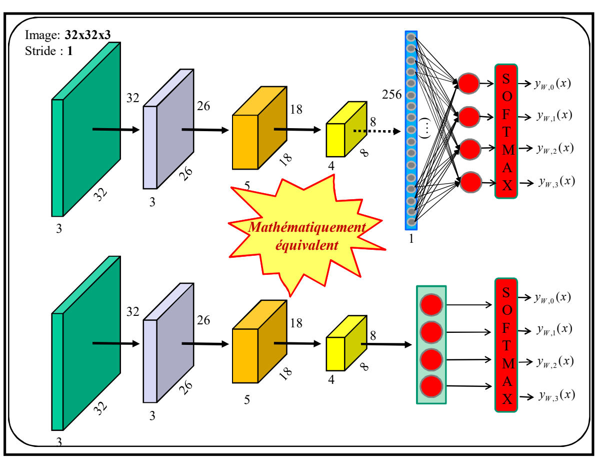
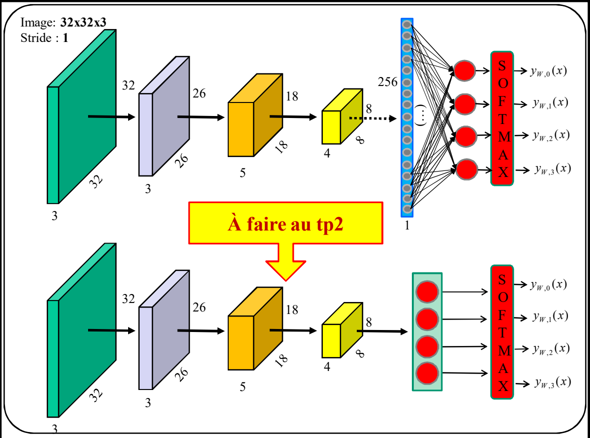


Image: 32x32x3
Stride : 1
32
32
3
26
26
3
18
18
5
8
8
4
256
(...)
1
SOFTMAX
$y_{W,0}(x)$
$y_{W,1}(x)$
$y_{W,2}(x)$
$y_{W,3}(x)$
Mathématiquement
équivalent



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

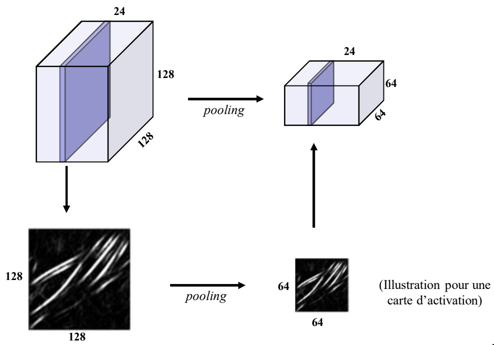
couche 1 : 3 filtres de taille 7x7
couche 2 : 5 filtres de taille 9x9
couche 3 : 4 filtres de taille 11x11
couche 4 : 4 filtres de taille 8x8
Softmax

En fait, presque équivalent …

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

Pooling

Réduction de la taille des cartes d’activation



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 2x2 avec stride =2

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

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 2x2 avec stride =2

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

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**

?

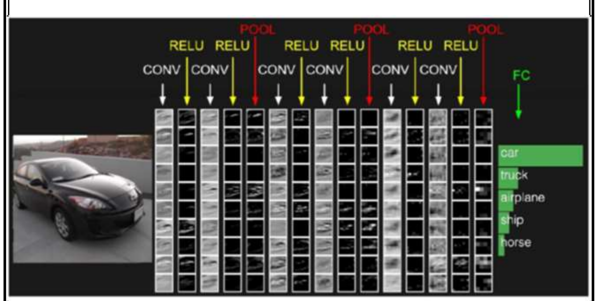
Max pooling

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**

?

Illustration d'un CNN complet

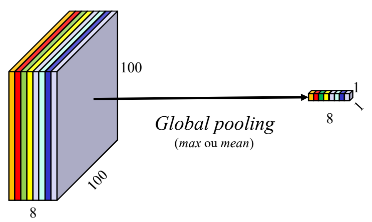


Crédit : cs231 Stanford

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



Multiplication matricielle parcimonieuse

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

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

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

=

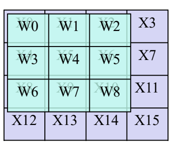
Y0	Y1
Y2	Y3

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

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

Rappel

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

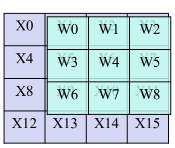


Y0	Y1
Y2	Y3

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

Rappel

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

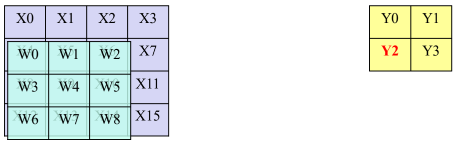


Y0	**Y1**
Y2	Y3

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

Rappel

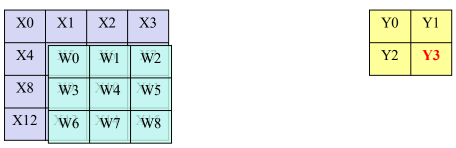
Ex.: convolution « *valid* », un canal d'entrée et une carte d'activation, filtre 3x3



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

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

Linéarisation de l'entrée et « matriçage » du filtre

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

$$\begin{bmatrix} W0 & W1 & W2 & 0 & W3 & W4 & W5 & 0 & W6 & W7 & W8 & 0 & 0 & 0 & 0 & 0 \\ 0 & W0 & W1 & W2 & 0 & W3 & W4 & W5 & 0 & W6 & W7 & W8 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & W0 & W1 & W2 & 0 & W3 & W4 & W5 & 0 & W6 & W7 & W8 & 0 \\ 0 & 0 & 0 & 0 & 0 & W0 & W1 & W2 & 0 & W3 & W4 & W5 & 0 & W6 & W7 & W8 \end{bmatrix} \text{ x } \begin{bmatrix} X0 \\ X1 \\ X2 \\ X3 \\ X4 \\ X5 \\ X6 \\ X7 \\ X8 \\ X9 \\ X10 \\ X11 \\ X12 \\ X13 \\ X14 \\ X15 \end{bmatrix} = \begin{bmatrix} Y0 \\ Y1 \\ Y2 \\ Y3 \end{bmatrix}$$

Linéarisation de l'entrée et « matriçage » du filtre

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

Entrée

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

*

Filtre

W0	W1
W2	W3

=

Y0	Y1	Y2
Y3	Y4	Y5
Y6	Y7	Y8

Linéarisation de l'entrée et « matriçage » du filtre

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

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

x

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

=

Y0
Y1
Y2
Y3
Y4
Y5
Y6
Y7
Y8

Linéarisation de l'entrée et « matriçage » du filtre

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

Entrée

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

*

Filtre

W0	W1
W2	W3

W4	W5
W6	W7

=

Y0	Y1	Y2
Y3	Y4	Y5
Y6	Y7	Y8

Y9	Y10	Y11
Y12	Y13	Y14
Y15	Y16	Y17

Linéarisation de l’entrée et « matriçage » du filtre

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

W4	W5	0	0	W6	W7	0	0	0	0	0	0	0	0	0	0
0															
0															
0															
0															
0															
0															
0															
0															

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

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

=

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

Linéarisation de l’entrée et « matriçage » du filtre

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

Entrée

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

=

Y0	Y1	Y2
Y3	Y4	Y5
Y6	Y7	Y8

Y9	Y10	Y11
Y12	Y13	Y14
Y15	Y16	Y17

Linéarisation de l’entrée et « matriçage » du filtre

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

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

x

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

=

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

Linéarisation de l'entrée et « matriçage » du filtre

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

Entrée

X0	X1	X2
X3	X4	X5
X6	X7	X8

X9	X10	X11
X12	X13	X14
X15	X16	X17

*

Filtre

W0	W1
W2	W3

W4	W5
W6	W7

=

Y0	Y1
Y2	Y3

Linéarisation de l'entrée et « matriçage » du filtre

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

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

x

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

=

Y0
Y1
Y2
Y3

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

Ex.: convolution « *valid* »

Entrée

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

*

Filtre

W0	W1	W2
W3	W4	W5
W6	W7	W8

=

Y0	Y1
Y2	Y3

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

Ex.: convolution « *valid* »

Entrée

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

X0
X1
X2
X4
X5
X6
X8
X9
X10

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

Ex.: convolution « *valid* »

Entrée

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

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

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

Ex.: convolution « *valid* »

Entrée

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

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

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

Ex.: convolution « *valid* »

Entrée

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

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

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

Ex.: convolution « *valid* »

W0	W1	W2	W3	W4	W5	W6	W7	W8

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

=

Y1	Y2	Y3	Y4

Ou encore…

Ex.: convolution « *valid* »

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

x

W0
W1
W2
W3
W4
W5
W6
W7
W8

=

Y0
Y1
Y2
Y3

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**.

Comment calculer la rétropropagation dans un CNN?

À faire au TP2