Réseaux de neurones à convolution

Guillaume Bourmaud

PLAN

I. Couche de convolution

II. Réseaux de neurones à convolution

I) Couche de convolution

I)

Limites d'une transformation affine générale (FC)

H=480

W=640

$\mathbf{x} : 640 \times 480 \times 3 \approx 10^6$ éléments

I)

Limites d'une transformation affine générale (FC)

H=480

W=640

$\mathbf{x} : 640 \times 480 \times 3 \approx 10^6$ éléments

Exemple d'une seule couche FC préservant la résolution de l'image d'entrée

$$\mathbf{y} = \mathbf{W}\mathbf{x} + \mathbf{b}$$

10^6x1 10^6x10^6 10^6x1 10^6x1

I)

Limites d'une transformation affine générale (FC)

H=480

W=640

$\mathbf{x} : 640 \times 480 \times 3 \approx 10^6$ éléments

Exemple d'une seule couche FC préservant la résolution de l'image d'entrée

$$\mathbf{y} = \mathbf{W}\mathbf{x} + \mathbf{b}$$

10^6x1 10^6x10^6 10^6x1 10^6x1

Occupation mémoire de W : 4 octets (32 bits) x 10^6x10^6 = 4To.

Nombre de « multiplications+additions » également très élevé.

I)

Opération de « convolution » = transformation affine spécifique

"Fully Connected"
=
Transformation affine générale

$$\begin{bmatrix} W_{11} & W_{12} & W_{13} & W_{14} \\ W_{21} & W_{22} & W_{23} & W_{24} \\ W_{31} & W_{32} & W_{33} & W_{34} \end{bmatrix} \begin{bmatrix} \mathbf{x}_1 \\ \mathbf{x}_2 \\ \mathbf{x}_3 \\ \mathbf{x}_4 \end{bmatrix} + \begin{bmatrix} \mathbf{b}_1 \\ \mathbf{b}_2 \\ \mathbf{b}_3 \end{bmatrix}$$

I)

Opération de « convolution » = transformation affine spécifique

“Fully Connected”
=
Transformation affine générale

$$\begin{bmatrix} W_{11} & W_{12} & W_{13} & W_{14} \\ W_{21} & W_{22} & W_{23} & W_{24} \\ W_{31} & W_{32} & W_{33} & W_{34} \end{bmatrix} \begin{bmatrix} \mathbf{x}_1 \\ \mathbf{x}_2 \\ \mathbf{x}_3 \\ \mathbf{x}_4 \end{bmatrix} + \begin{bmatrix} \mathbf{b}_1 \\ \mathbf{b}_2 \\ \mathbf{b}_3 \end{bmatrix}$$

Localement connecté

$$\begin{bmatrix} W_{11} & W_{12} & 0 & 0 \\ 0 & W_{22} & W_{23} & 0 \\ 0 & 0 & W_{33} & W_{34} \end{bmatrix} \begin{bmatrix} \mathbf{x}_1 \\ \mathbf{x}_2 \\ \mathbf{x}_3 \\ \mathbf{x}_4 \end{bmatrix} + \begin{bmatrix} \mathbf{b}_1 \\ \mathbf{b}_2 \\ \mathbf{b}_3 \end{bmatrix}$$

I)

Opération de « convolution » = transformation affine spécifique

“Fully Connected”
=
Transformation affine générale

$$\begin{bmatrix} W_{11} & W_{12} & W_{13} & W_{14} \\ W_{21} & W_{22} & W_{23} & W_{24} \\ W_{31} & W_{32} & W_{33} & W_{34} \end{bmatrix} \begin{bmatrix} \mathbf{x}_1 \\ \mathbf{x}_2 \\ \mathbf{x}_3 \\ \mathbf{x}_4 \end{bmatrix} + \begin{bmatrix} \mathbf{b}_1 \\ \mathbf{b}_2 \\ \mathbf{b}_3 \end{bmatrix}$$

Localement connecté

$$\begin{bmatrix} W_{11} & W_{12} & 0 & 0 \\ 0 & W_{22} & W_{23} & 0 \\ 0 & 0 & W_{33} & W_{34} \end{bmatrix} \begin{bmatrix} \mathbf{x}_1 \\ \mathbf{x}_2 \\ \mathbf{x}_3 \\ \mathbf{x}_4 \end{bmatrix} + \begin{bmatrix} \mathbf{b}_1 \\ \mathbf{b}_2 \\ \mathbf{b}_3 \end{bmatrix}$$

Équivariance par translation
=
« Convolution »

$$\begin{bmatrix} W_{11} & W_{12} & 0 & 0 \\ 0 & W_{11} & W_{12} & 0 \\ 0 & 0 & W_{11} & W_{12} \end{bmatrix} \begin{bmatrix} \mathbf{x}_1 \\ \mathbf{x}_2 \\ \mathbf{x}_3 \\ \mathbf{x}_4 \end{bmatrix} + \begin{bmatrix} b \\ b \\ b \end{bmatrix}$$

I)

Opération de « convolution » = transformation affine spécifique

"Fully Connected"
=
Transformation affine générale

$$\begin{bmatrix} W_{11} & W_{12} & W_{13} & W_{14} \\ W_{21} & W_{22} & W_{23} & W_{24} \\ W_{31} & W_{32} & W_{33} & W_{34} \end{bmatrix} \begin{bmatrix} \mathbf{x}_1 \\ \mathbf{x}_2 \\ \mathbf{x}_3 \\ \mathbf{x}_4 \end{bmatrix} + \begin{bmatrix} \mathbf{b}_1 \\ \mathbf{b}_2 \\ \mathbf{b}_3 \end{bmatrix}$$

Localement connecté

$$\begin{bmatrix} W_{11} & W_{12} & 0 & 0 \\ 0 & W_{22} & W_{23} & 0 \\ 0 & 0 & W_{33} & W_{34} \end{bmatrix} \begin{bmatrix} \mathbf{x}_1 \\ \mathbf{x}_2 \\ \mathbf{x}_3 \\ \mathbf{x}_4 \end{bmatrix} + \begin{bmatrix} \mathbf{b}_1 \\ \mathbf{b}_2 \\ \mathbf{b}_3 \end{bmatrix}$$

Équivariance par translation
=
« Convolution »

$$\begin{bmatrix} W_{11} & W_{12} & 0 & 0 \\ 0 & W_{11} & W_{12} & 0 \\ 0 & 0 & W_{11} & W_{12} \end{bmatrix} \begin{bmatrix} \mathbf{x}_1 \\ \mathbf{x}_2 \\ \mathbf{x}_3 \\ \mathbf{x}_4 \end{bmatrix} + \begin{bmatrix} b \\ b \\ b \end{bmatrix}$$

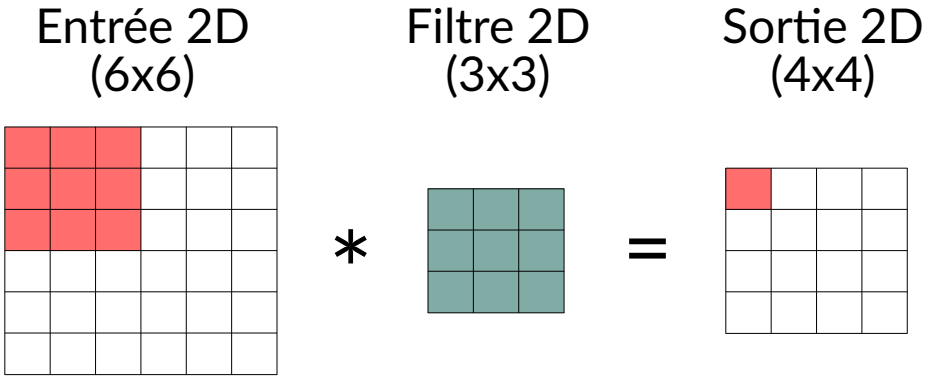
Beaucoup moins de paramètres à stocker

Beaucoup moins de « multiplications+additions »

I)

Opération de « convolution » en 2D

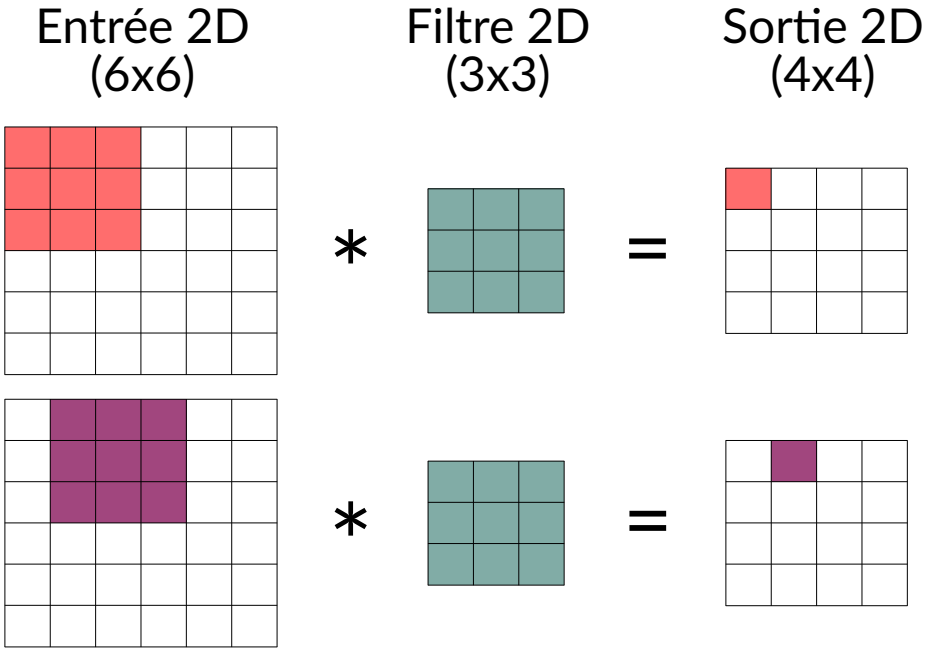
En fait, il s'agit d'une intercorrélation



I)

Opération de « convolution » en 2D

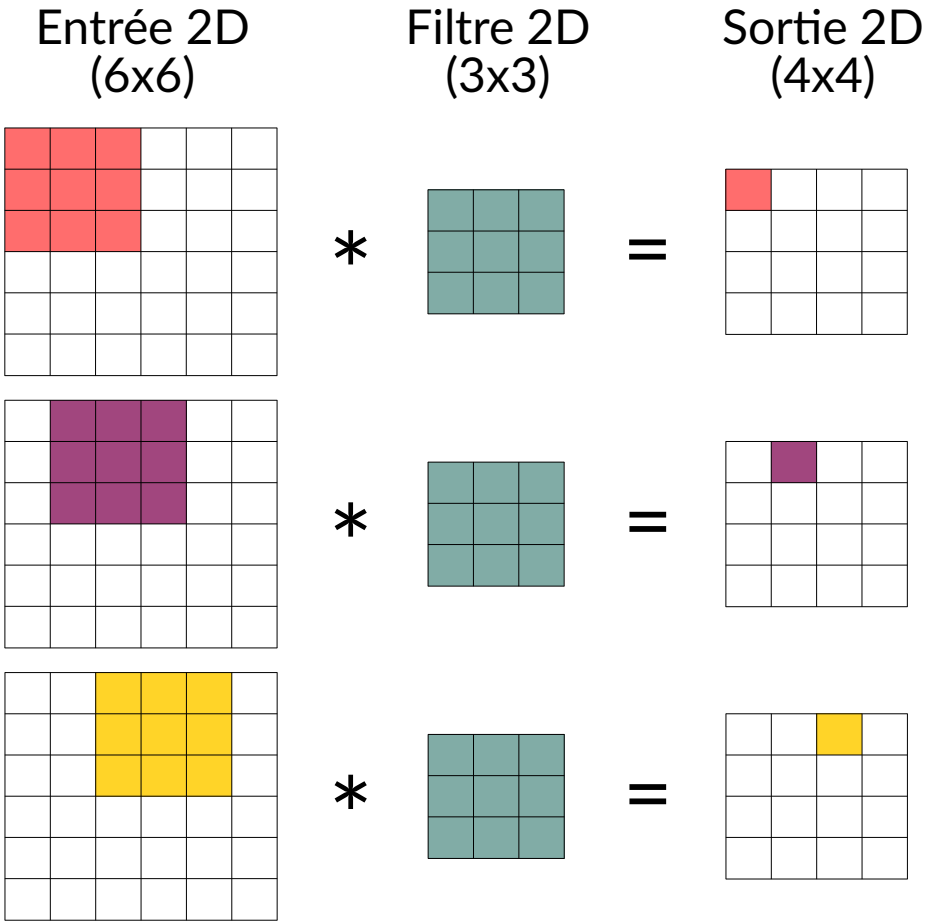
En fait, il s'agit d'une intercorrélation



I)

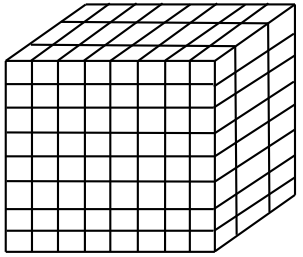
Opération de « convolution » en 2D

En fait, il s'agit d'une intercorrélation



I)

Couche de convolution 2D : un seul filtre



$\mathbf{X}^{(0)}$

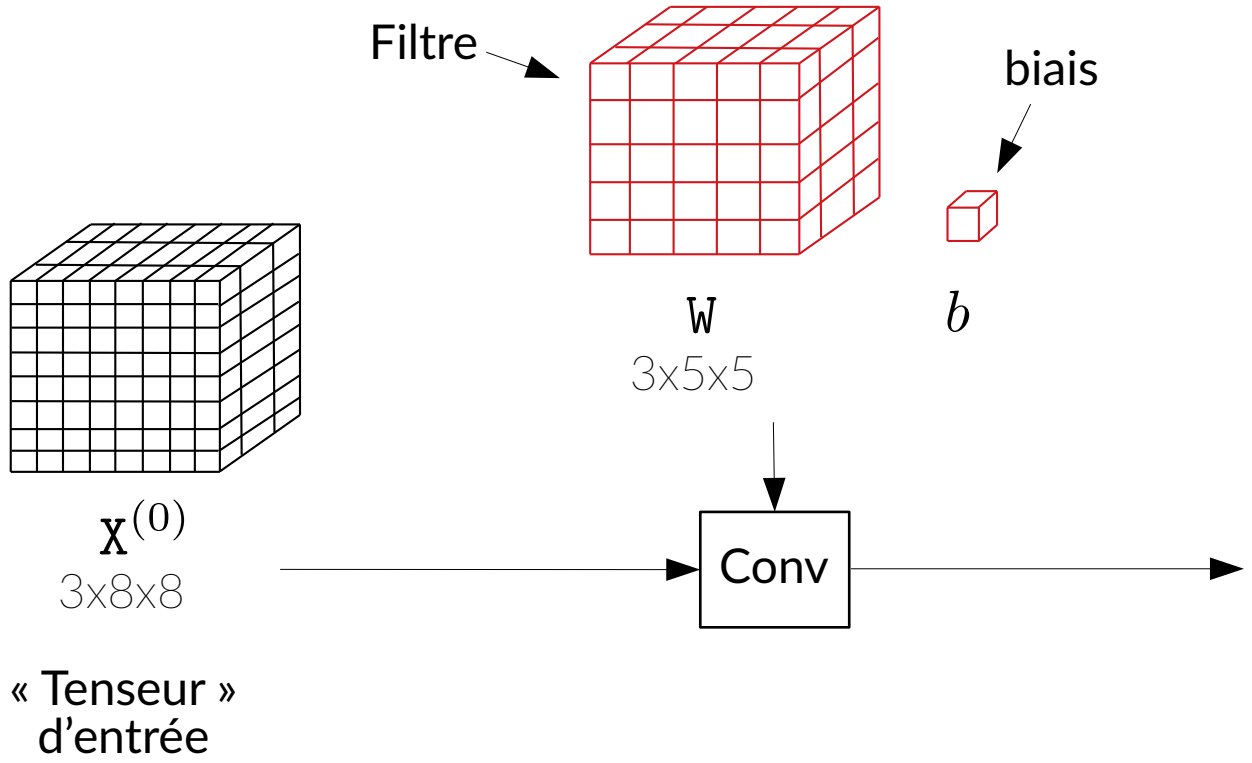
3x8x8

« Tenseur » d'entrée

« Tenseur » = tableau multi-dimensionnel

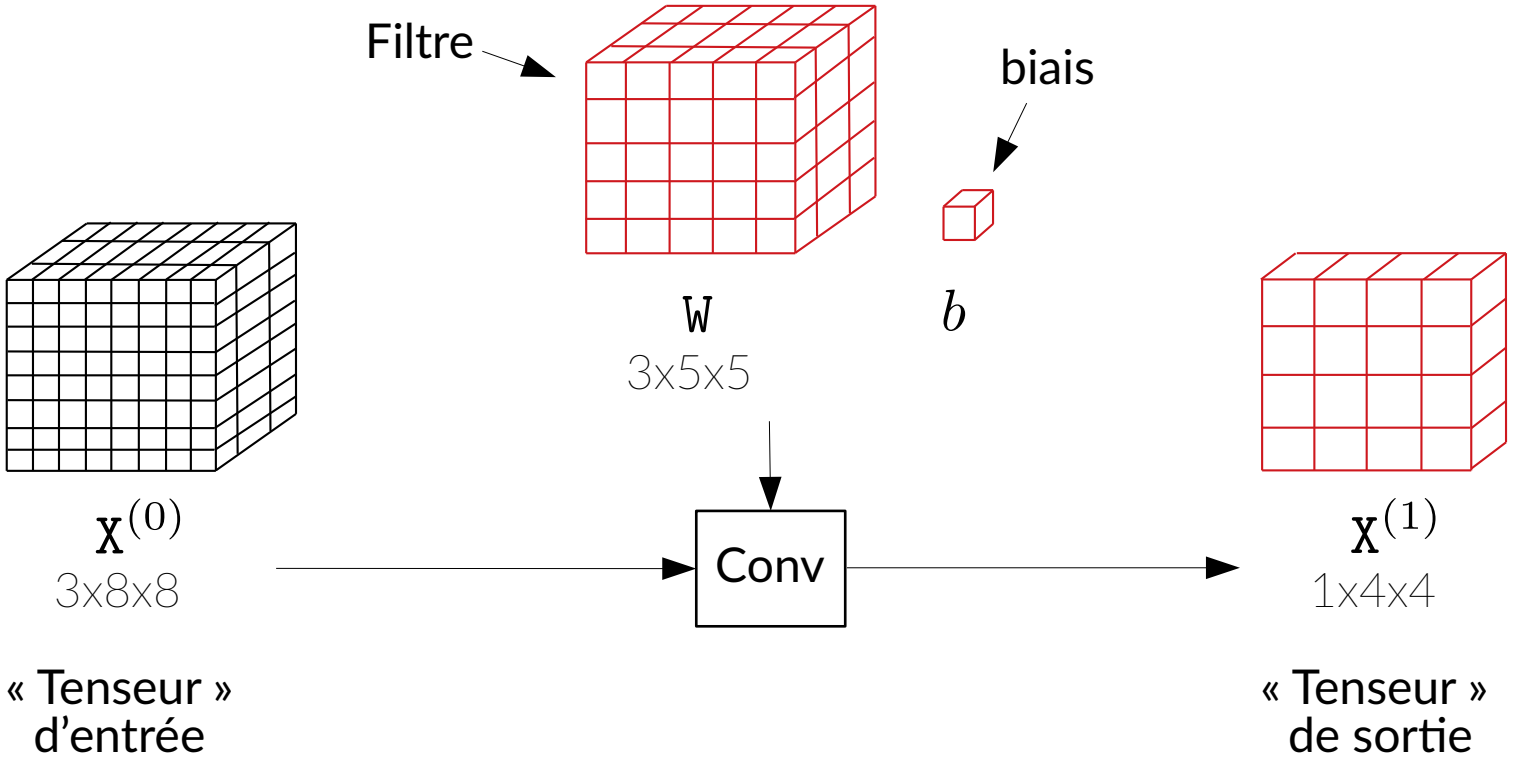
I)

Couche de convolution 2D : un seul filtre



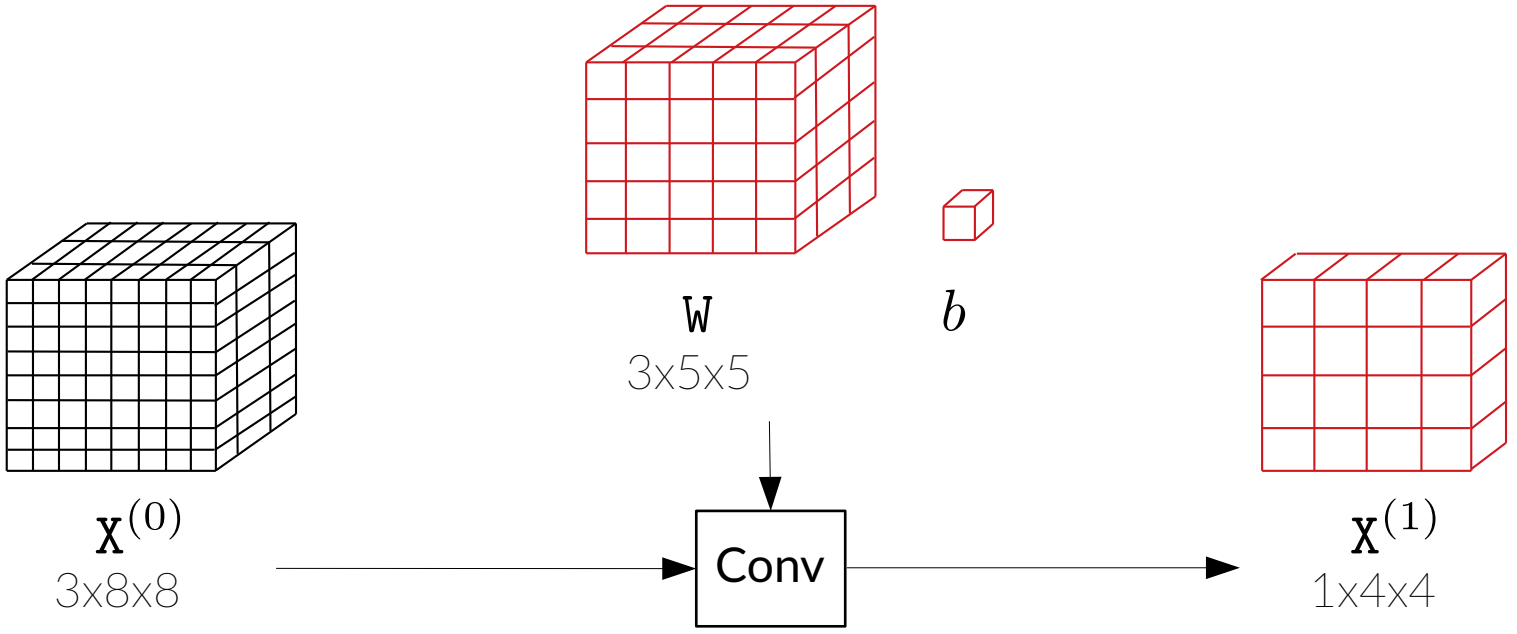
I)

Couche de convolution 2D : un seul filtre



I)

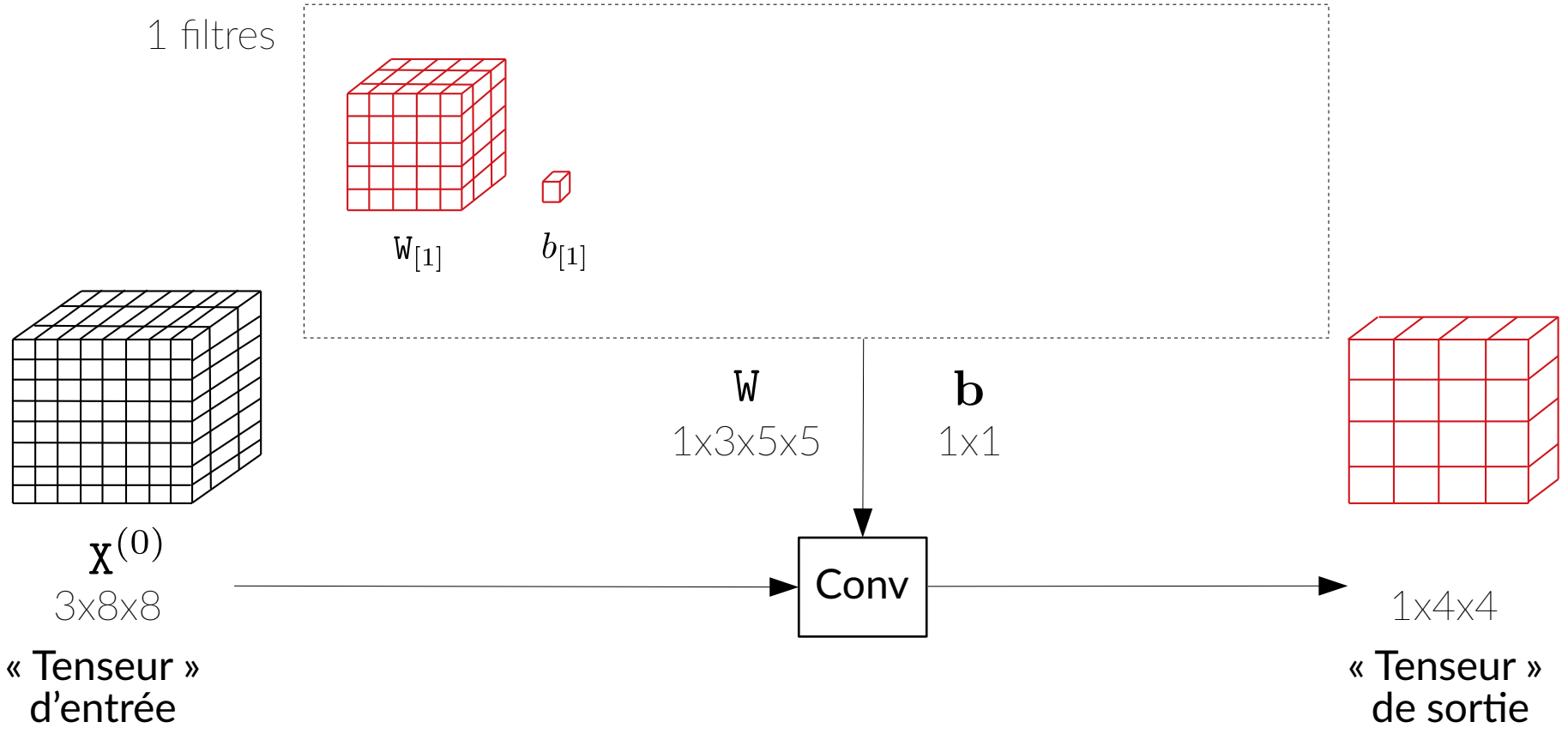
Couche de convolution 2D : un seul filtre



$$\mathbf{X}^{(1)}_{i,j} = \sum_{k=0}^{2} \sum_{m=0}^{4} \sum_{n=0}^{4} \mathbf{W}_{k,m,n} \mathbf{X}^{(0)}_{k,i+m,j+n} + b$$

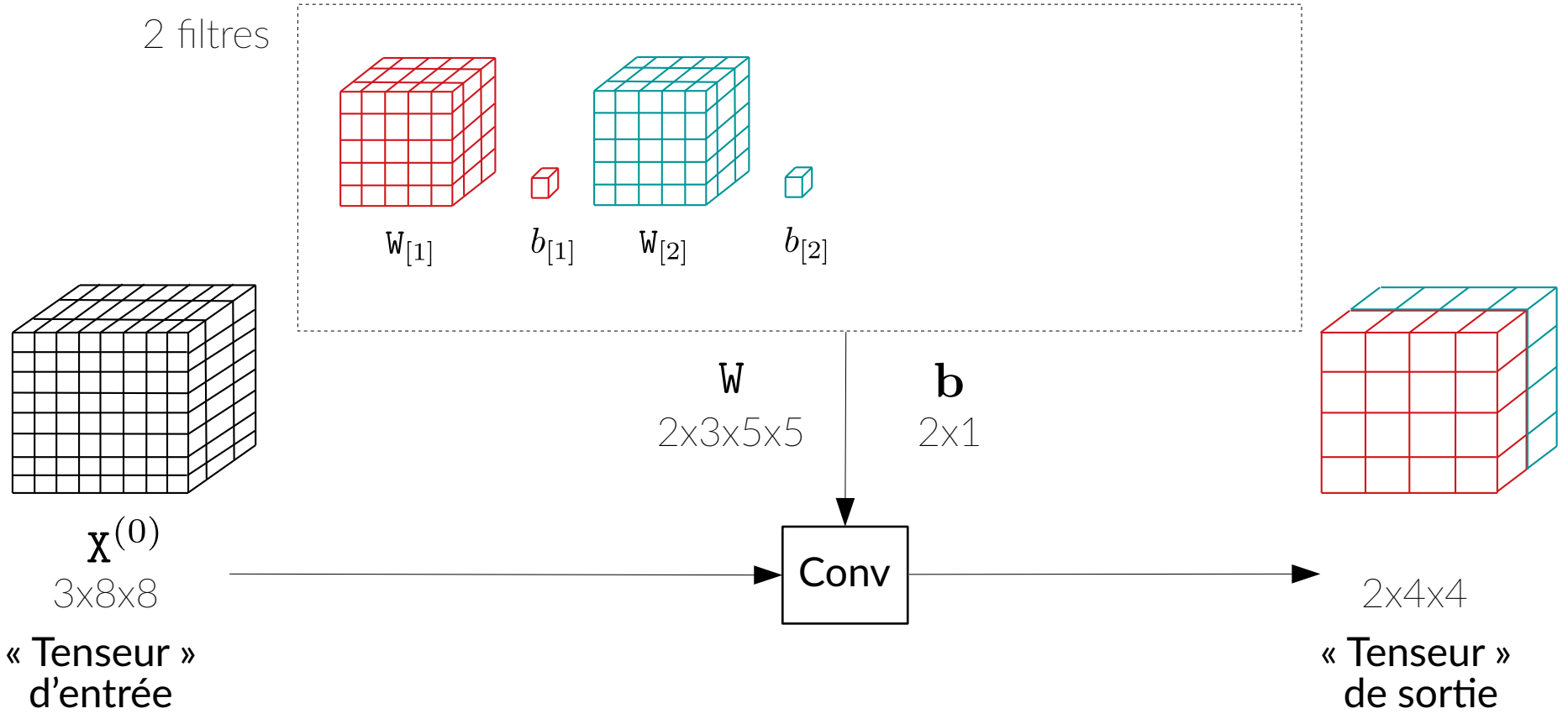
I)

Couche de convolution 2D : K filtres



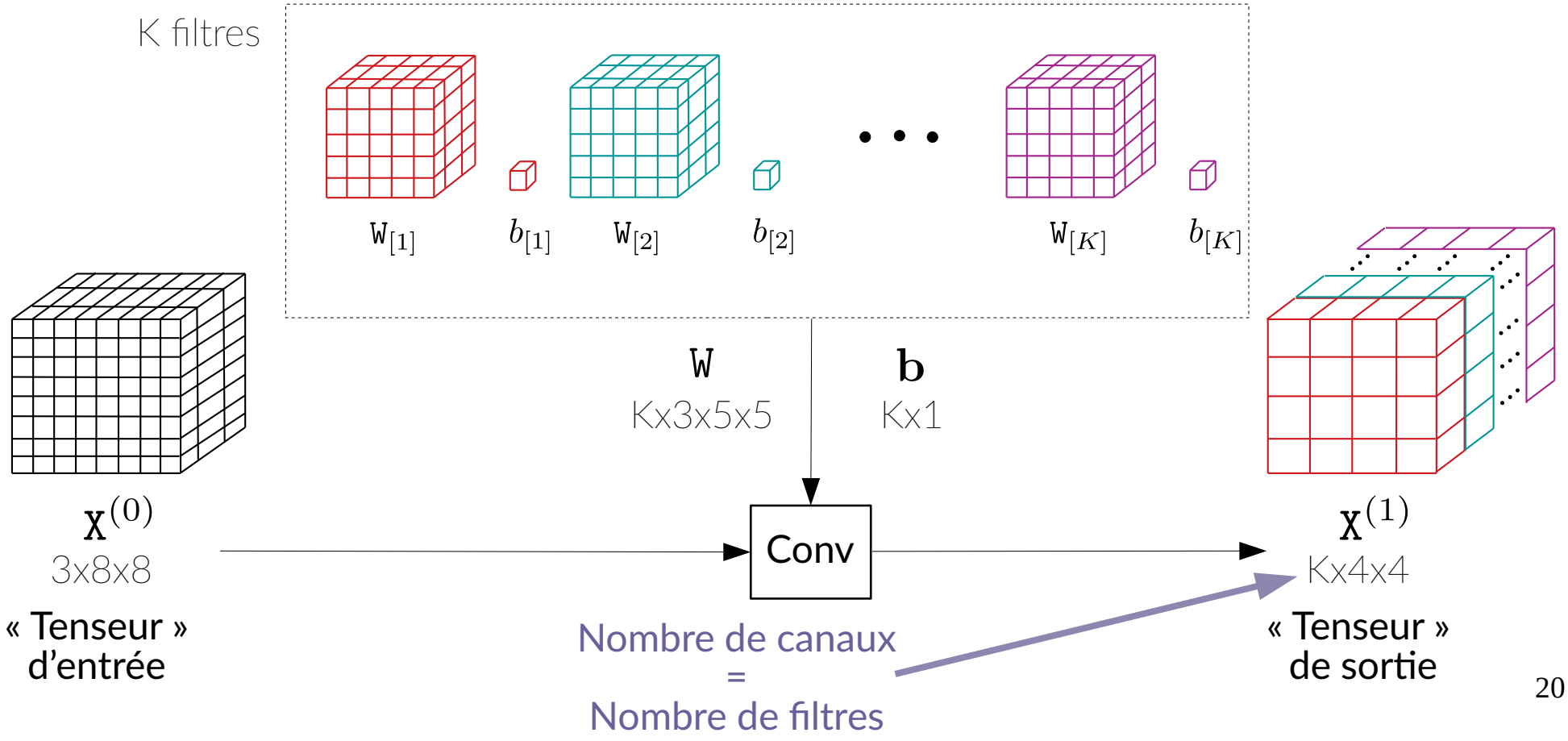
I)

Couche de convolution 2D : K filtres



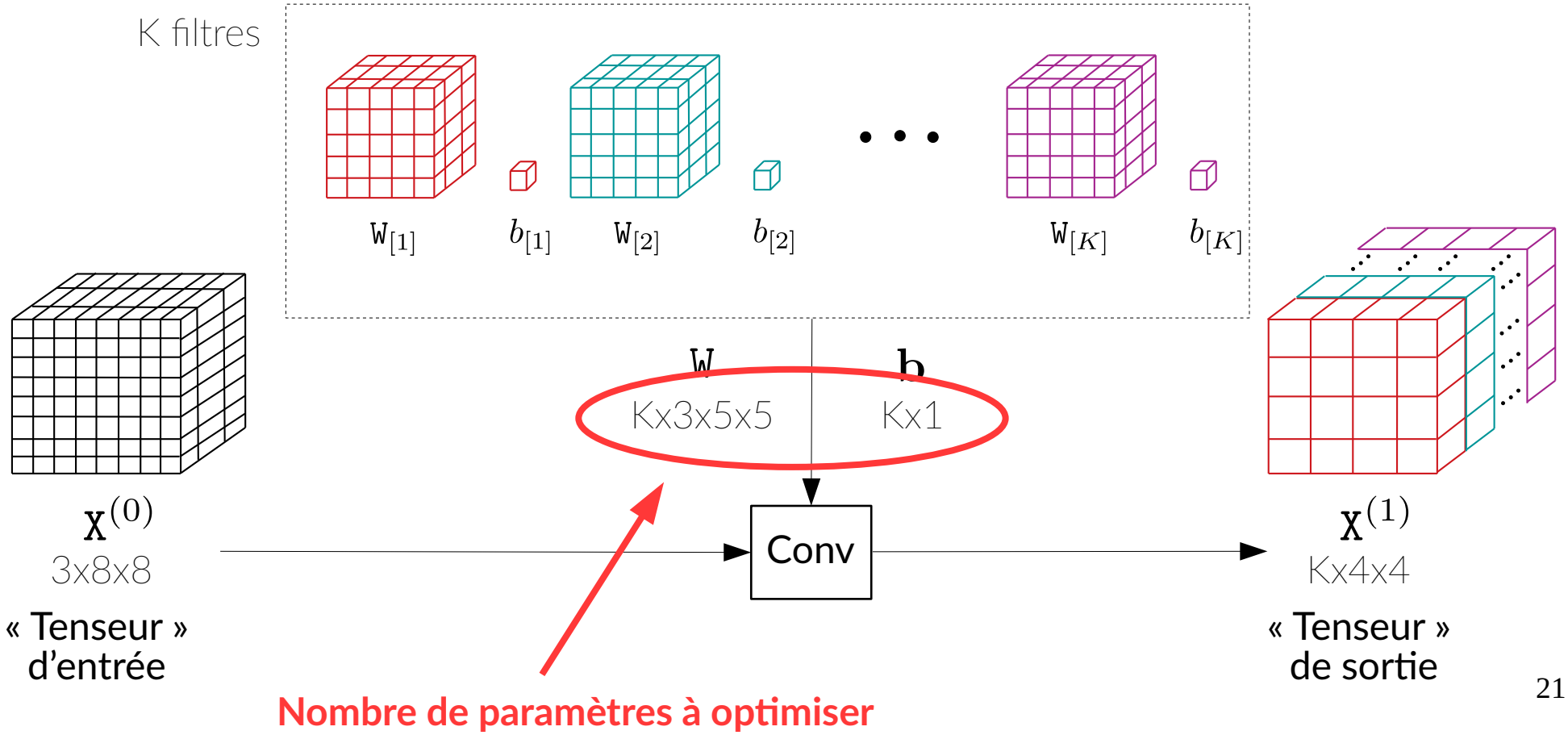
I)

Couche de convolution 2D : K filtres



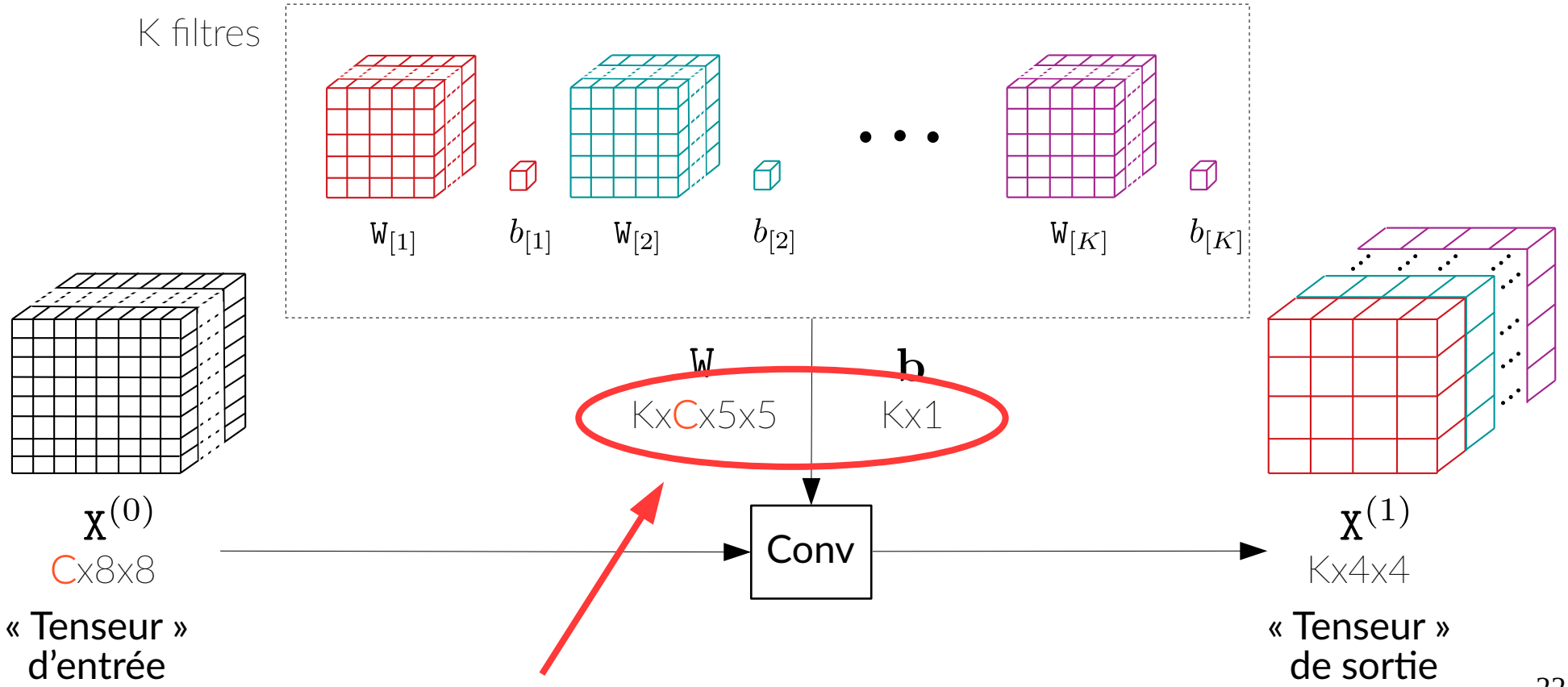
I)

Couche de convolution 2D : K filtres



I)

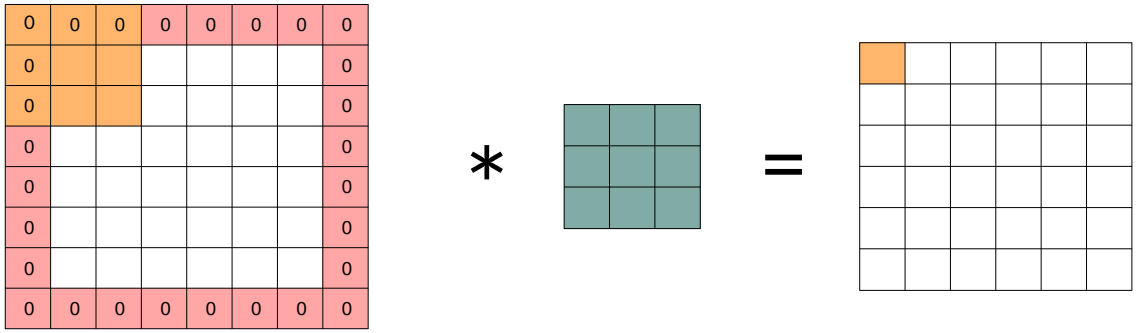
Couche de convolution 2D : K filtres



Nombre de paramètres à optimiser… peut devenir grand !

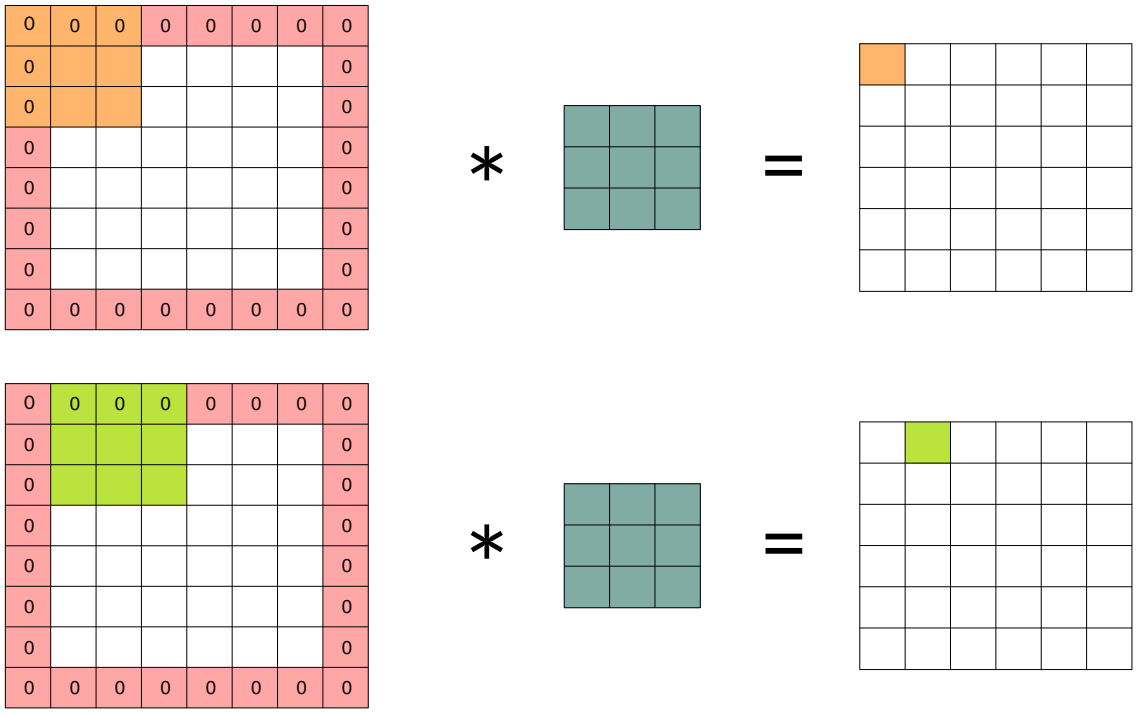
I)

« Zero padding »



I)

« Zero padding »

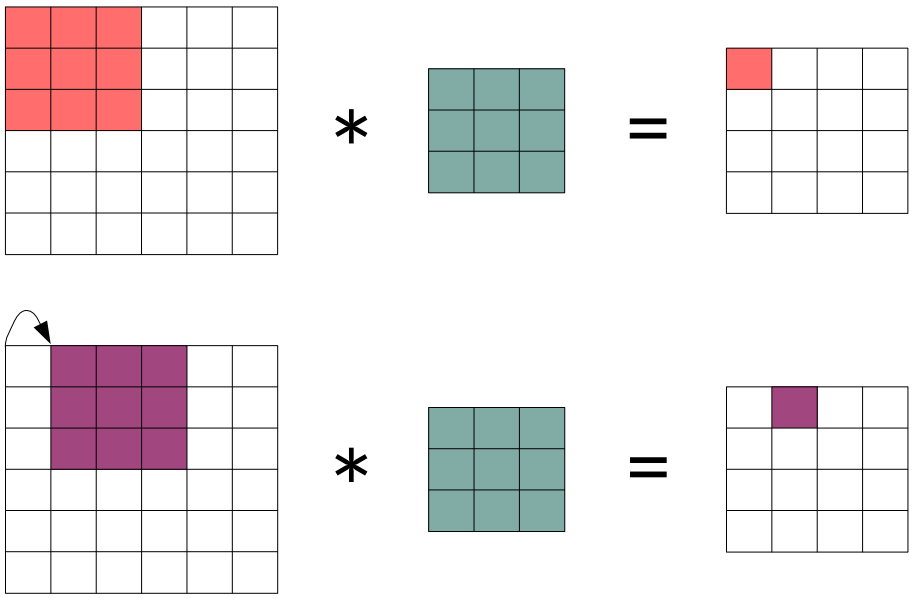


Permet de préserver la taille du tenseur d'entrée = ne pas avoir d'effet de bord

I)

« Stride »

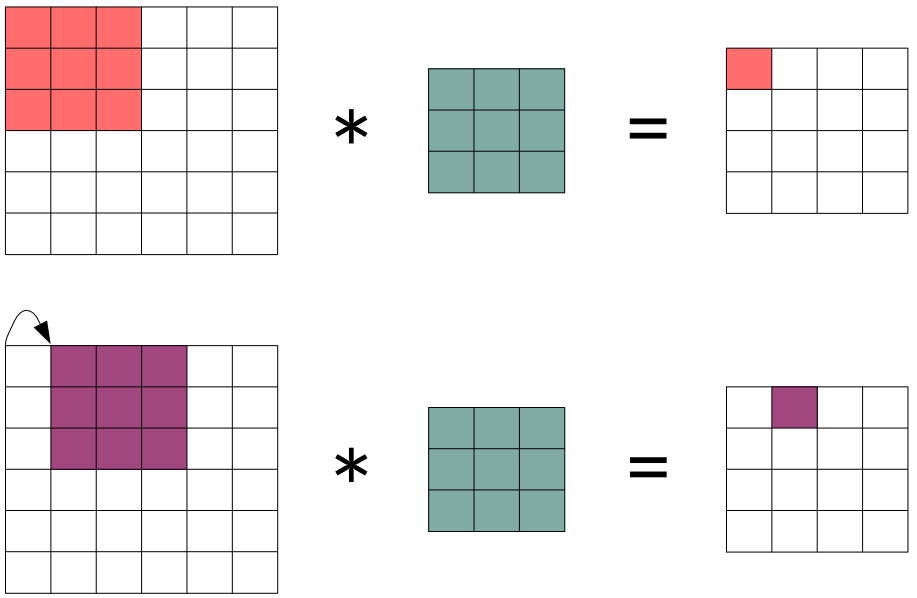
Exemple Stride = 1



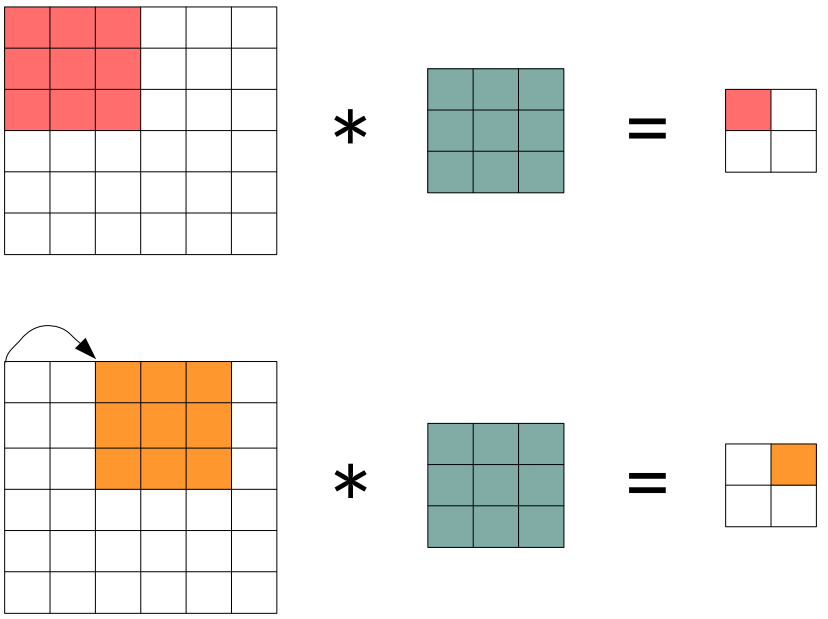
I)

« Stride »

Exemple Stride = 1



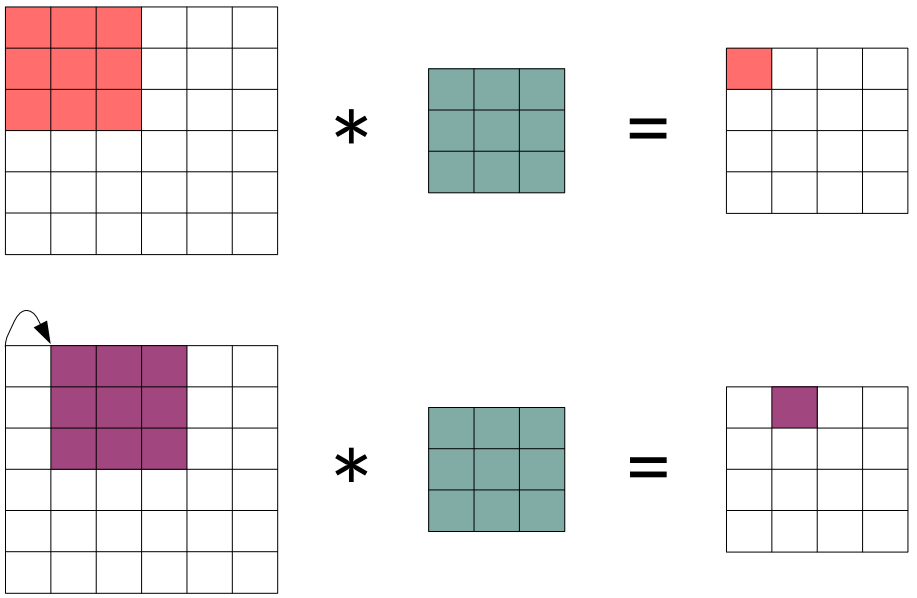
Exemple Stride = 2



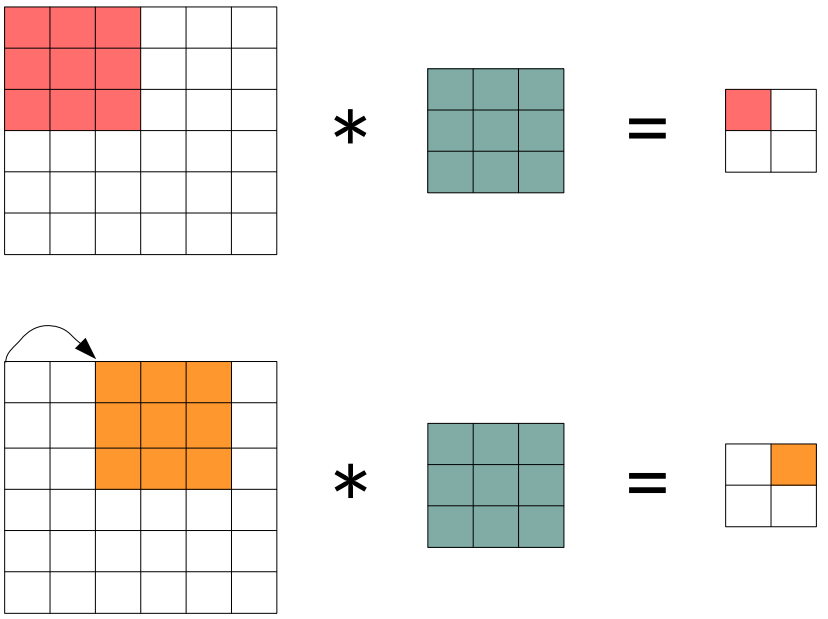
I)

« Stride »

Exemple Stride = 1



Exemple Stride = 2



Réduit la résolution de la sortie par « stride »

Principaux hyperparamètres d'une couche de convolution

- Taille des filtres

  - En pratique toujours impair
  - Souvent 3x3, parfois 5x5 ou 7x7

Principaux hyperparamètres d'une couche de convolution

- Taille des filtres

  - En pratique toujours impair
  - Souvent 3x3, parfois 5x5 ou 7x7

- Nombre de filtres

  = Nombre de canaux souhaités en sortie

Principaux hyperparamètres d'une couche de convolution

- Taille des filtres

  - En pratique toujours impair
  - Souvent 3x3, parfois 5x5 ou 7x7

- Nombre de filtres

  = Nombre de canaux souhaités en sortie

- Quantité de zero-padding

  - Compense la taille du filtre si volonté de préserver la taille de l'entrée
  - 3x3 → padding = 1, 5x5 → padding = 2

Principaux hyperparamètres d'une couche de convolution

- Taille des filtres

  - En pratique toujours impair
  - Souvent 3x3, parfois 5x5 ou 7x7

- Nombre de filtres

  = Nombre de canaux souhaités en sortie

- Quantité de zero-padding

  - Compense la taille du filtre si volonté de préserver la taille de l'entrée
  - 3x3 → padding = 1, 5x5 → padding = 2

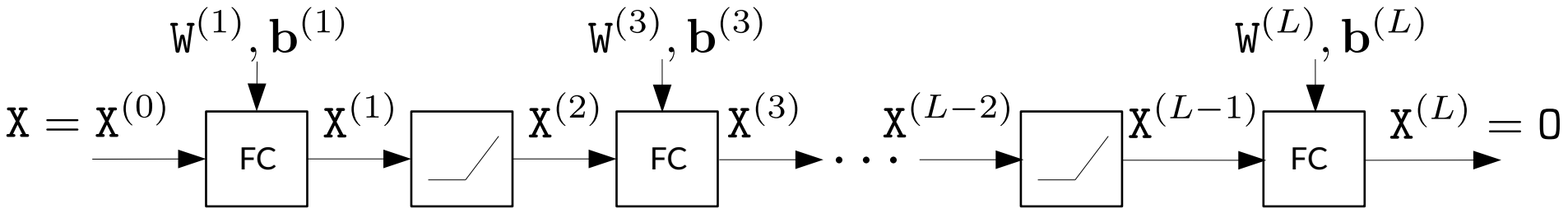
- Stride

  = 1 si volonté de préserver la résolution de l'entrée
  = 2 si volonté de réduire la résolution de l'entrée

II) Réseau de neurones à convolution

II)

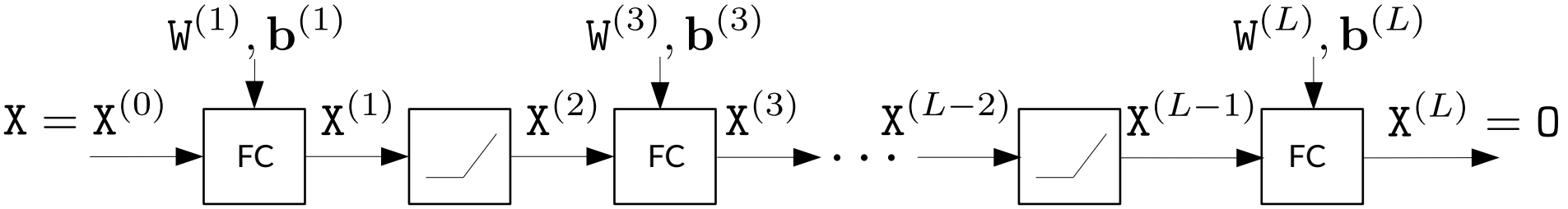
Réseau de neurones à convolution (CNN)



MLP

II)

Réseau de neurones à convolution (CNN)



MLP

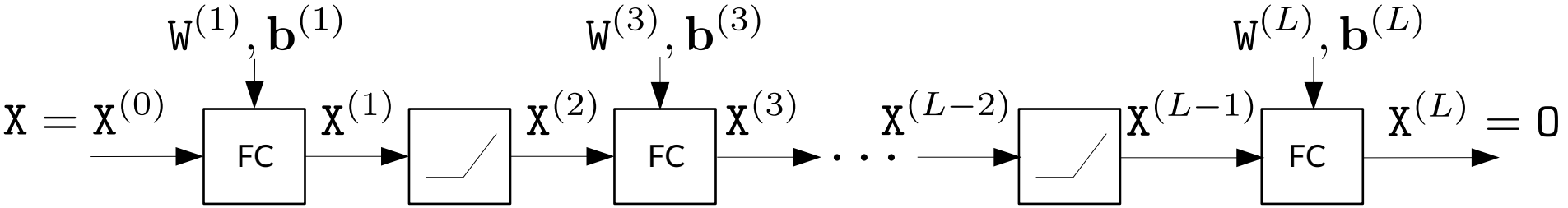
− FC (transformations affines générales)

\+ Conv (transformations affines spécifiques)

= CNN

II)

Réseau de neurones à convolution (CNN)

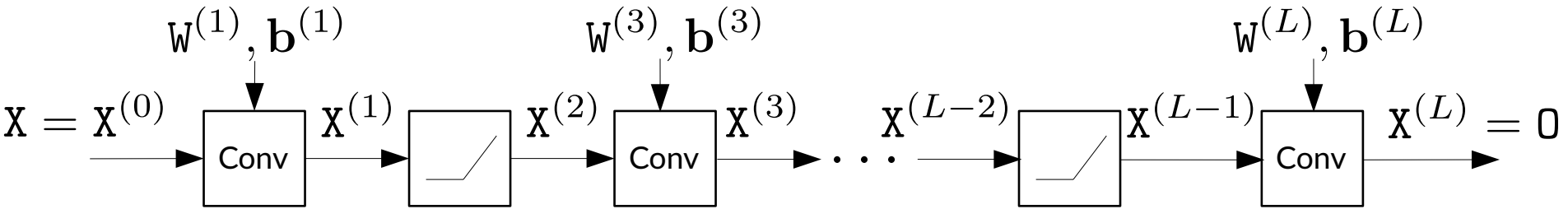


MLP

− FC (transformations affines générales)

\+ Conv (transformations affines spécifiques)

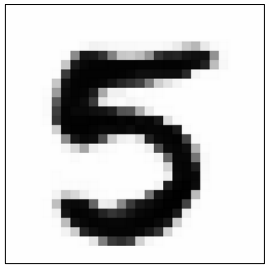
= CNN

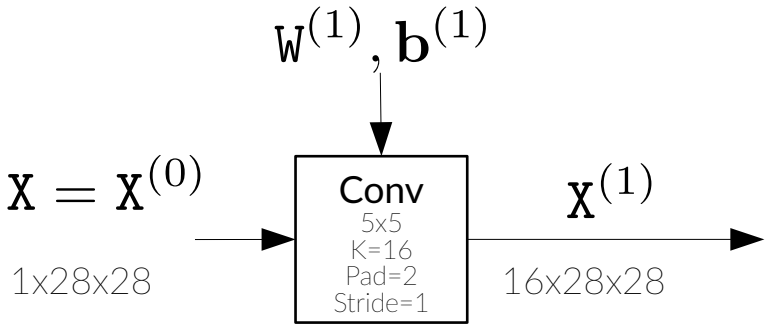


→ Initialisation des paramètres d'une couche de convolution identique à ceux d'une FC !

II)

Exemple d'architecture de CNN pour MNIST

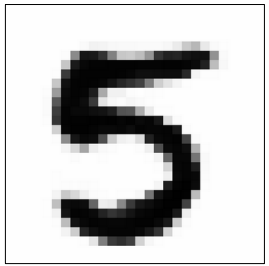


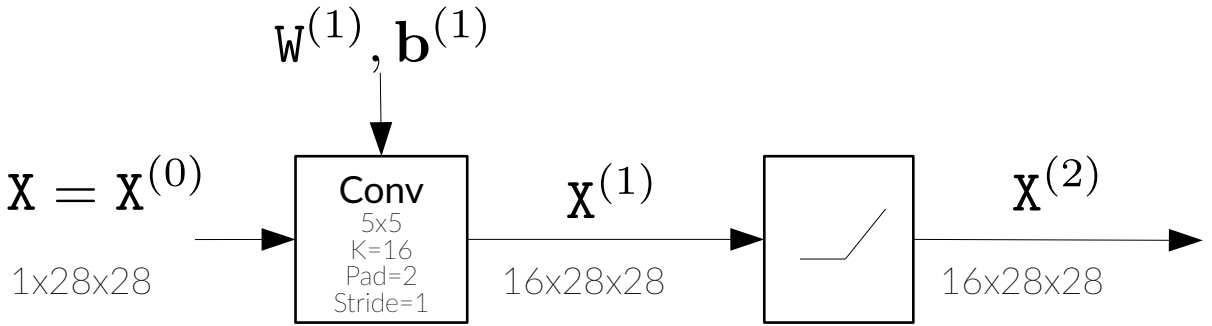


"0"
"1"
"2"
"3"
"4"
"5"
"6"
"7"
"8"
"9"

II)

Exemple d'architecture de CNN pour MNIST

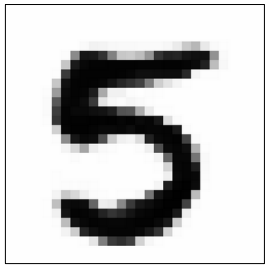


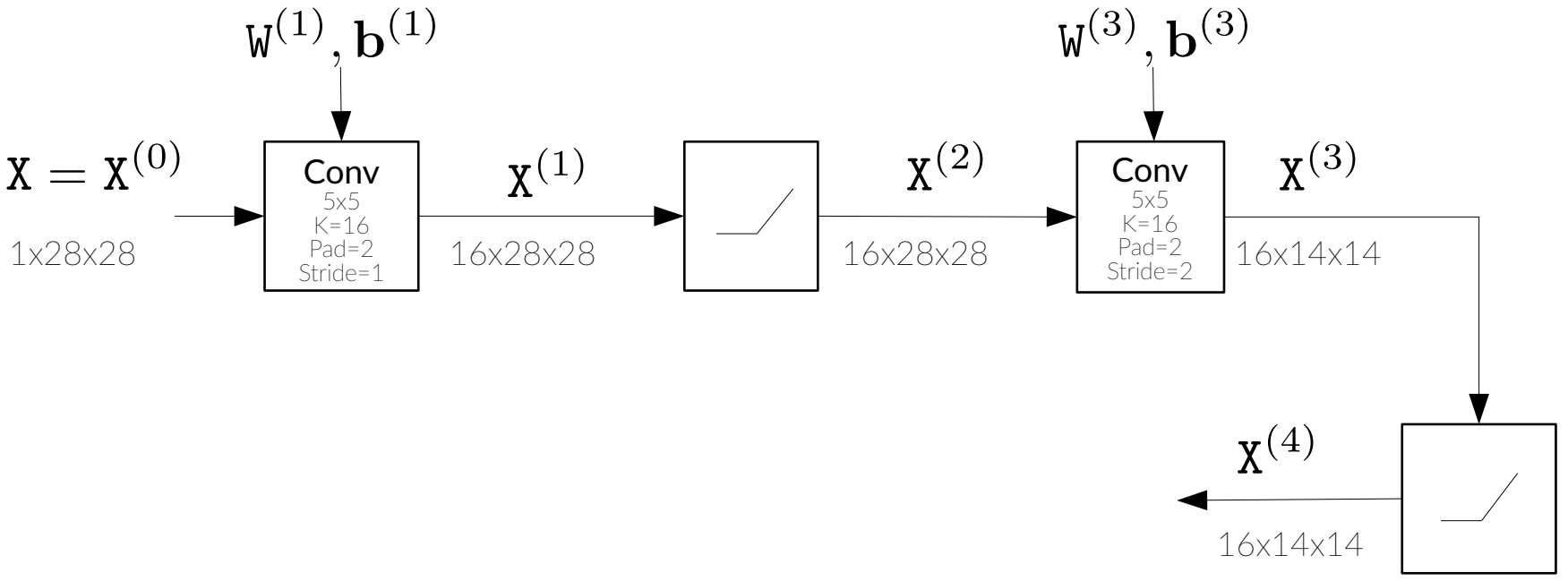


"0"
"1"
"2"
"3"
"4"
"5"
"6"
"7"
"8"
"9"

II)

Exemple d'architecture de CNN pour MNIST

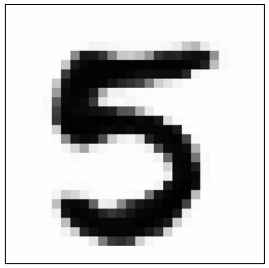


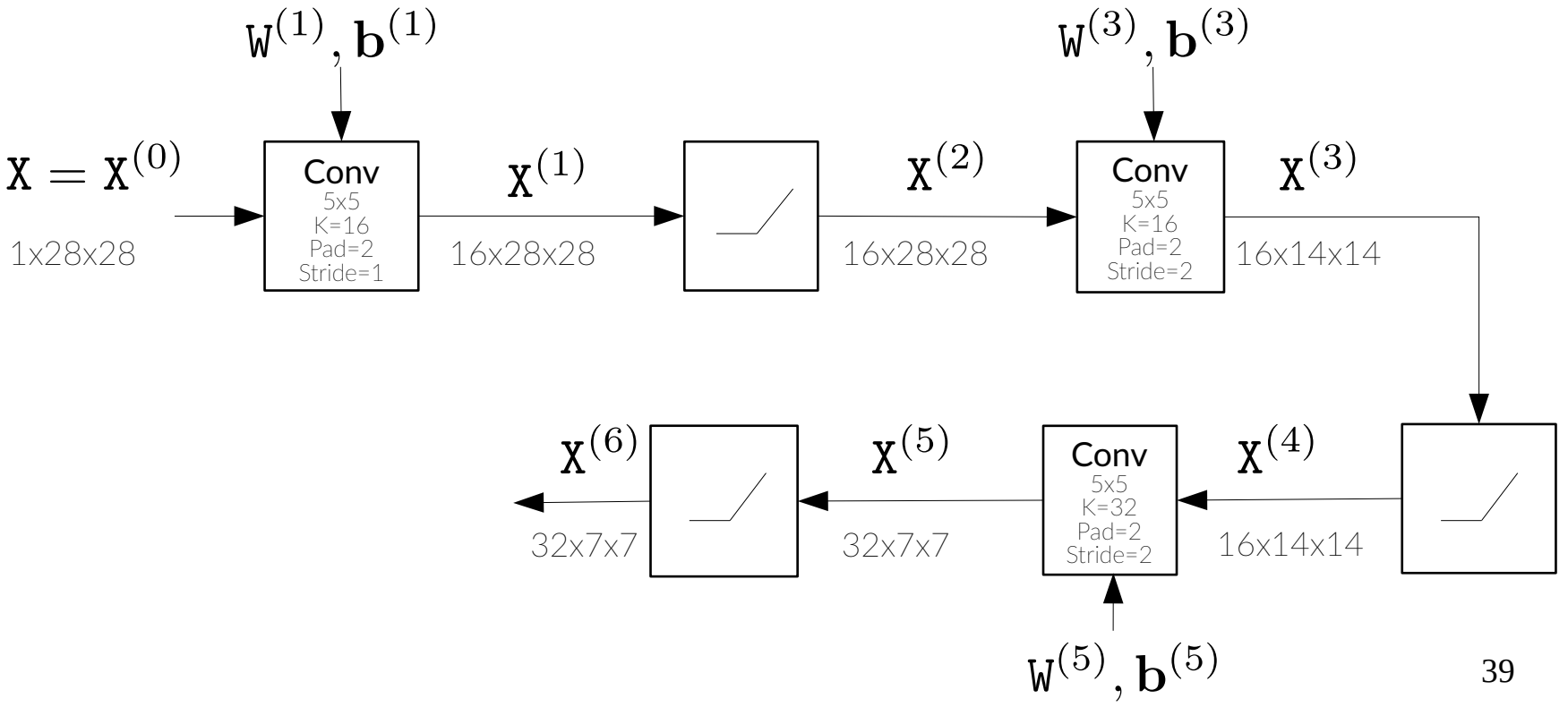


"0"
"1"
"2"
"3"
"4"
"5"
"6"
"7"
"8"
"9"

II)

Exemple d'architecture de CNN pour MNIST

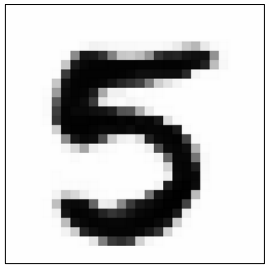


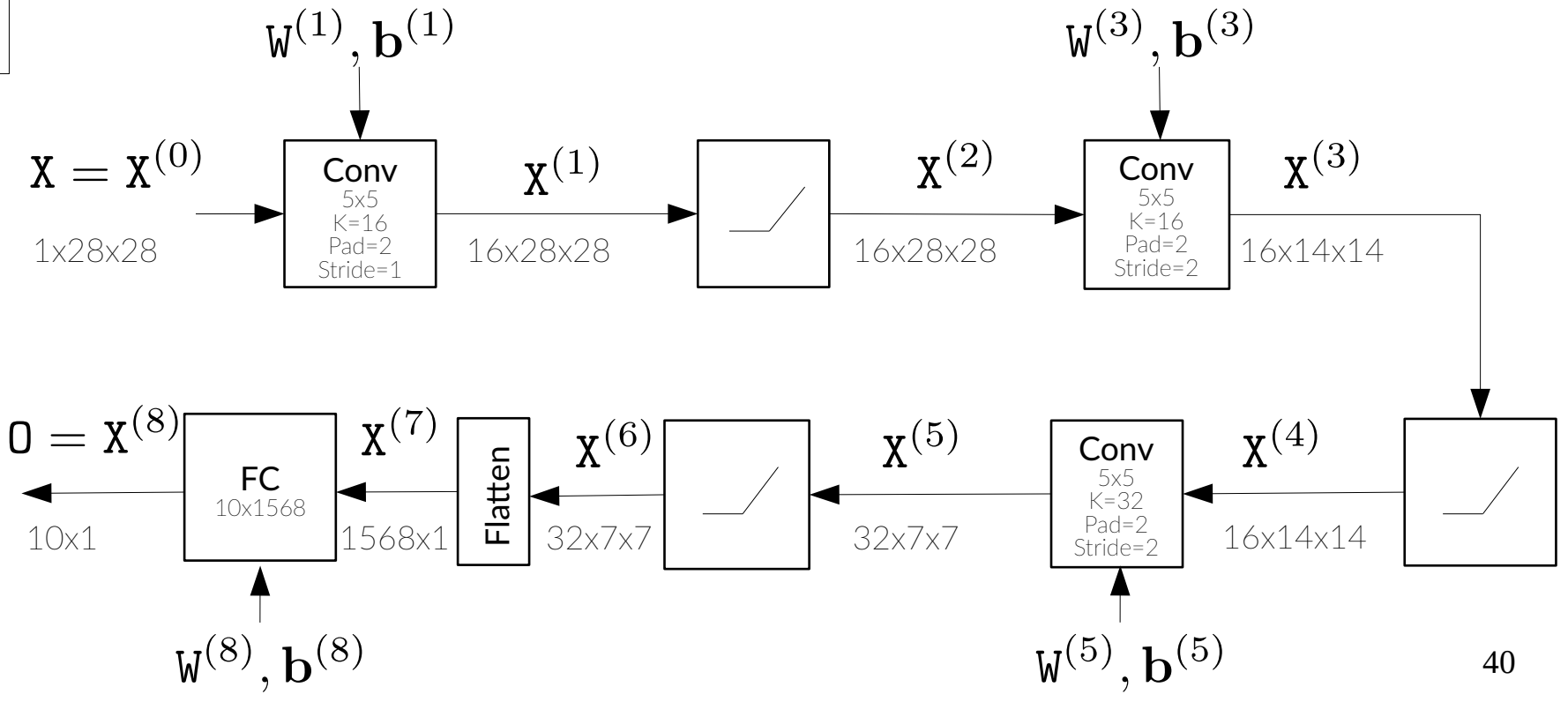


"0"
"1"
"2"
"3"
"4"
"5"
"6"
"7"
"8"
"9"

II)

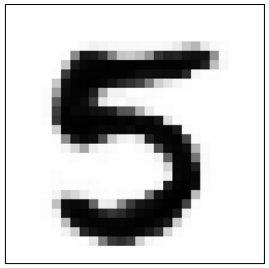
Exemple d'architecture de CNN pour MNIST

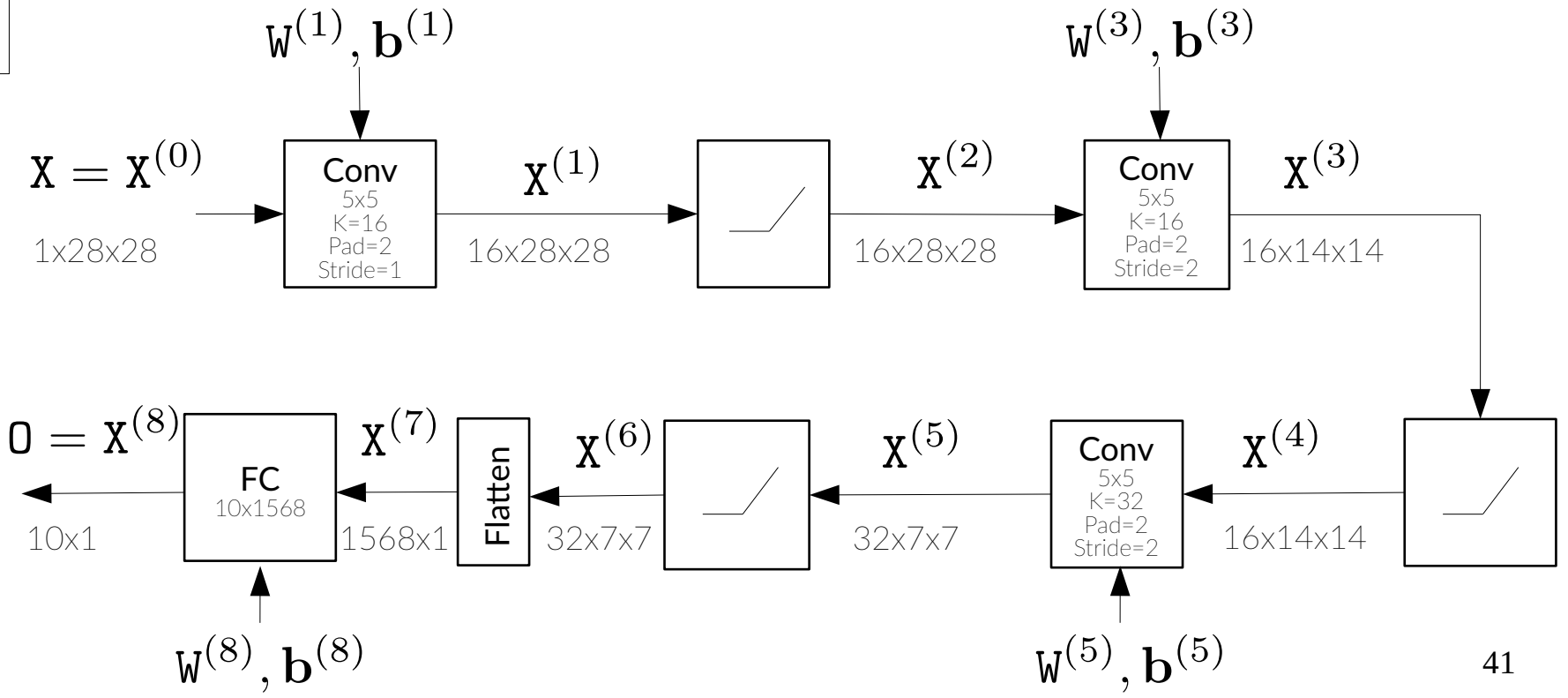




II)

Exemple d'architecture de CNN pour MNIST

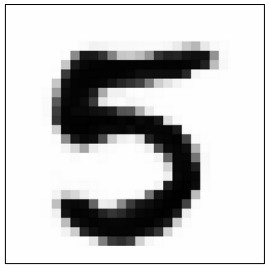


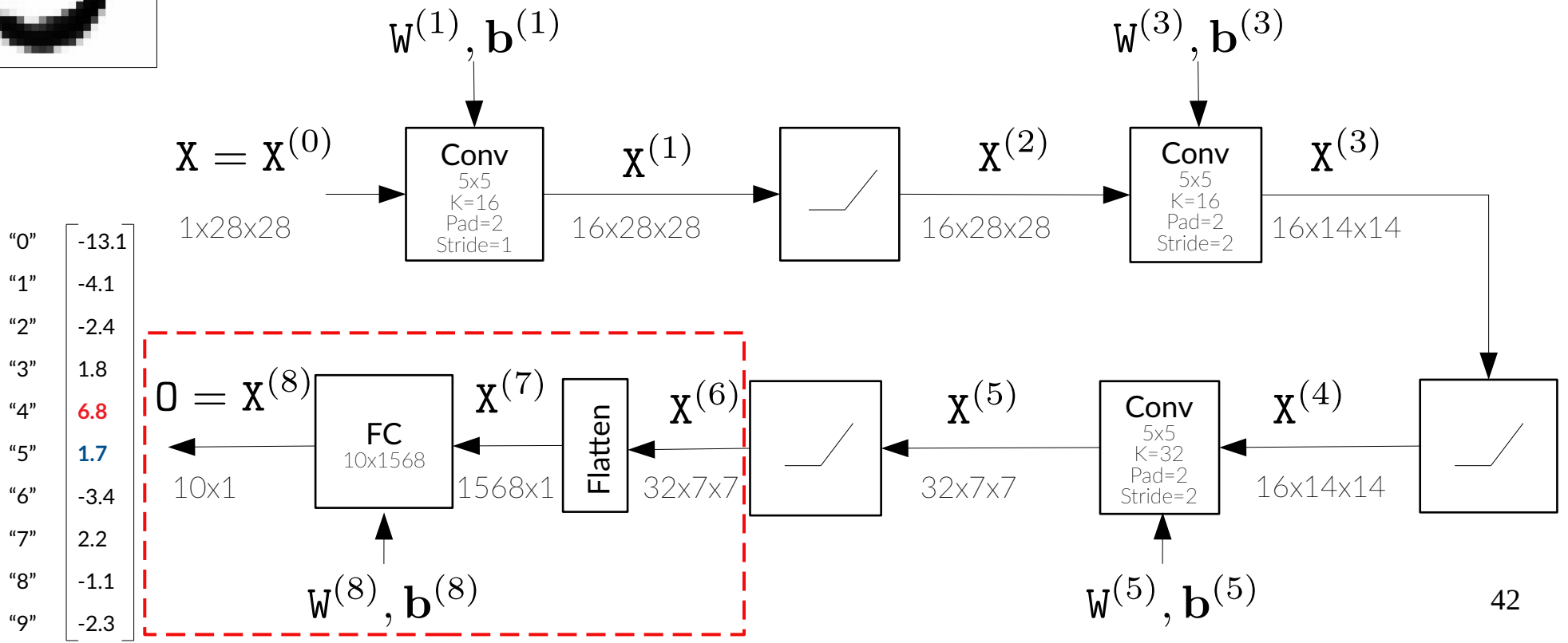


"0"	-13.1
"1"	-4.1
"2"	-2.4
"3"	1.8
"4"	6.8
"5"	1.7
"6"	-3.4
"7"	2.2
"8"	-1.1
"9"	-2.3

II)

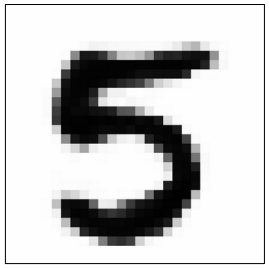
Exemple d'architecture de CNN pour MNIST

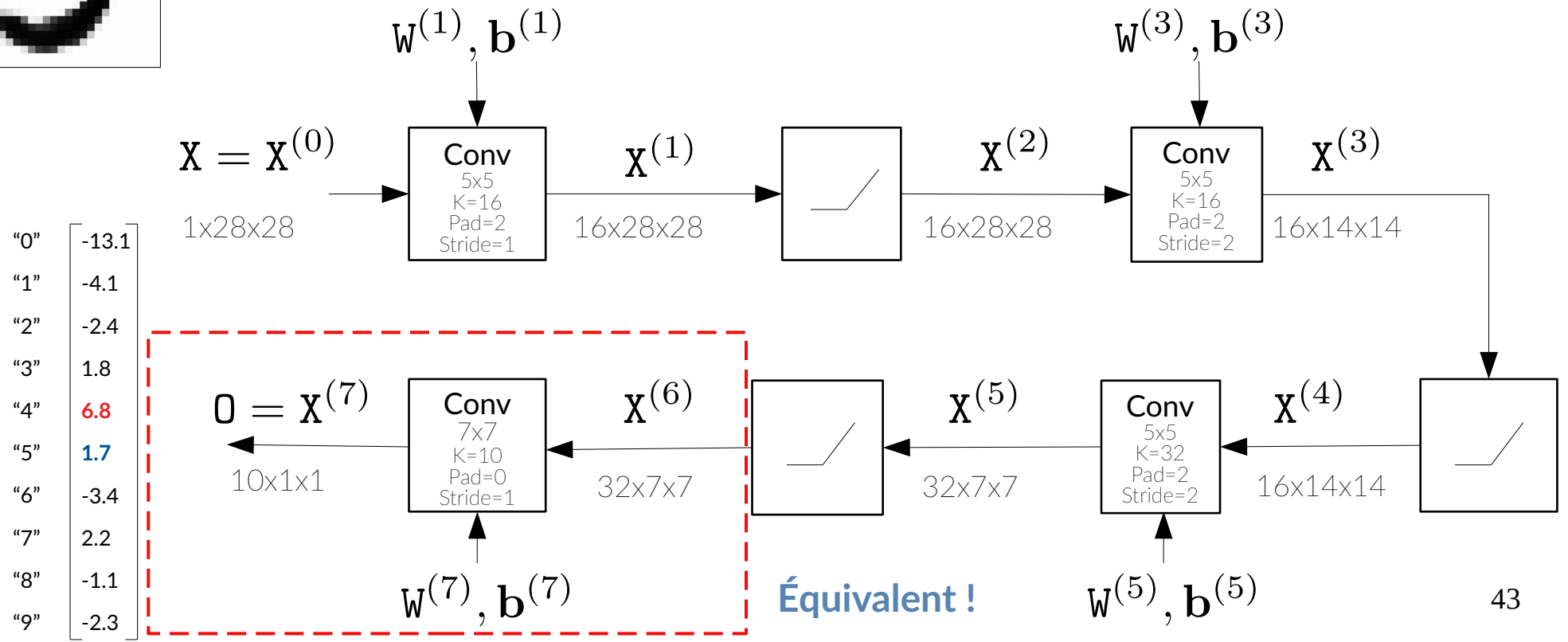




II)

Exemple d'architecture de CNN pour MNIST

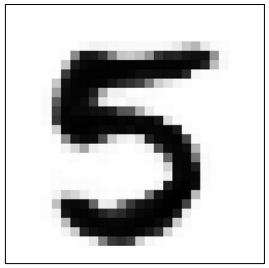


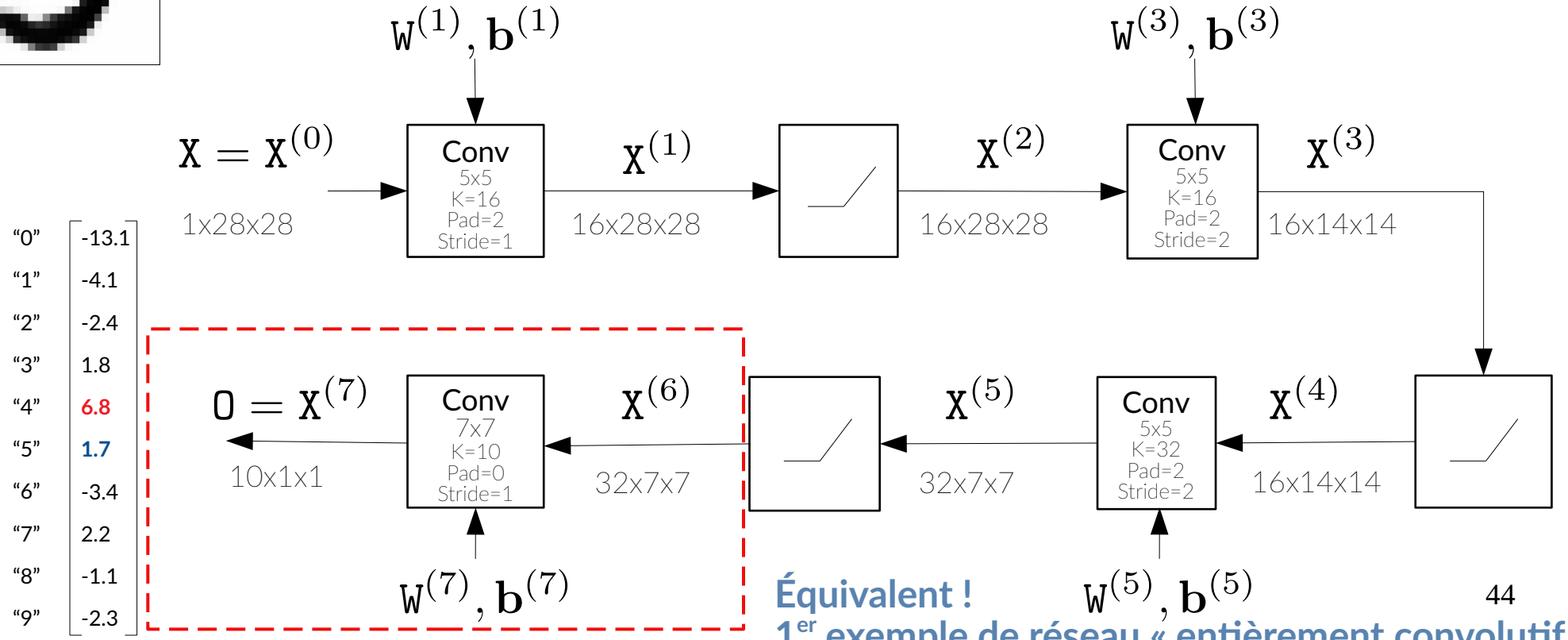


Équivalent !

II)

Exemple d'architecture de CNN pour MNIST



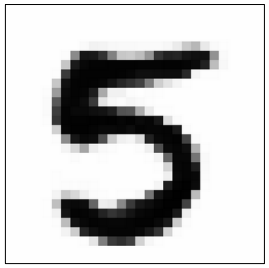


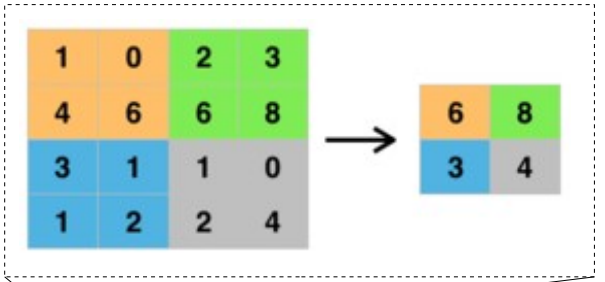
Équivalent !

1er exemple de réseau « entièrement convolutif »

II)

Autre exemple d'architecture : CNN pour MNIST





$\mathbf{W}^{(1)}, \mathbf{b}^{(1)}$

$\mathbf{X} = \mathbf{X}^{(0)}$

1x28x28

Conv
5x5
K=16
Pad=2
Stride=1

$\mathbf{X}^{(1)}$

16x28x28

$\mathbf{X}^{(2)}$

16x28x28

Pool
2x2
Stride=2

$\mathbf{X}^{(3)}$

16x14x14

Conv
5x5
K=32
Pad=2
Stride=1

$\mathbf{W}^{(4)}, \mathbf{b}^{(4)}$

$\mathbf{X}^{(4)}$

32x14x14

$\mathbf{X}^{(5)}$

32x14x14

Pool
2x2
Stride=2

$\mathbf{X}^{(6)}$

32x7x7

Conv
7x7
K=10
Pad=0
Stride=1

$\mathbf{W}^{(7)}, \mathbf{b}^{(7)}$

$\mathbf{O} = \mathbf{X}^{(7)}$

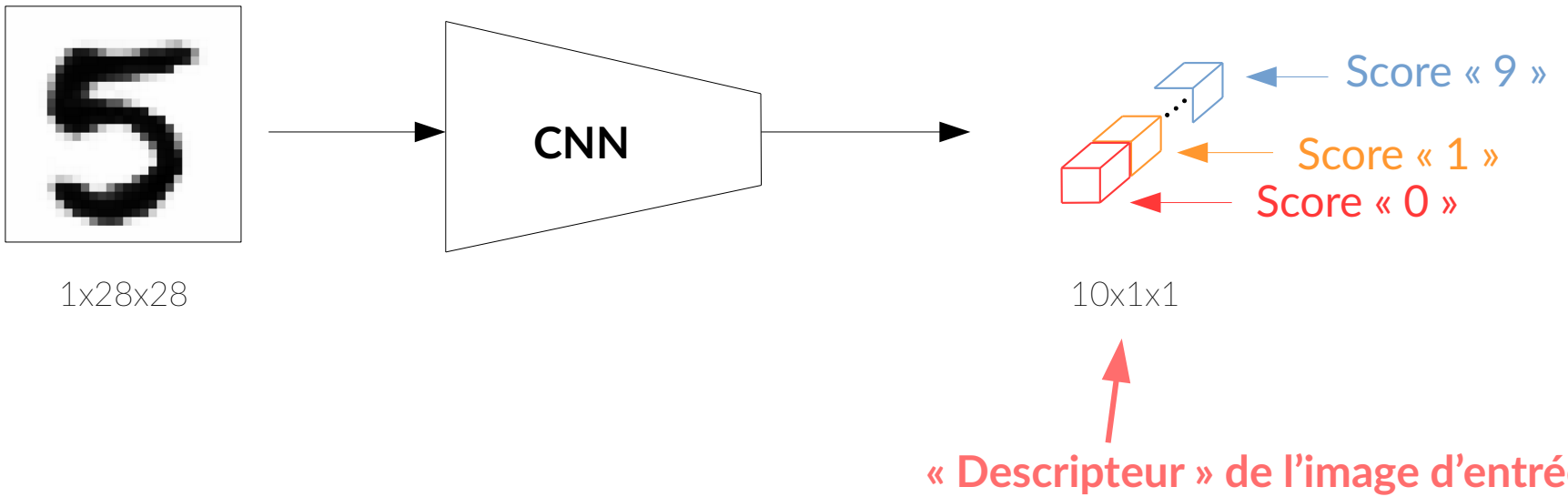
10x1x1

"0"	-13.1
"1"	-4.1
"2"	-2.4
"3"	1.8
"4"	6.8
"5"	1.7
"6"	-3.4
"7"	2.2
"8"	-1.1
"9"	-2.3

II)

Architectures de CNN : Deux cas extrêmes

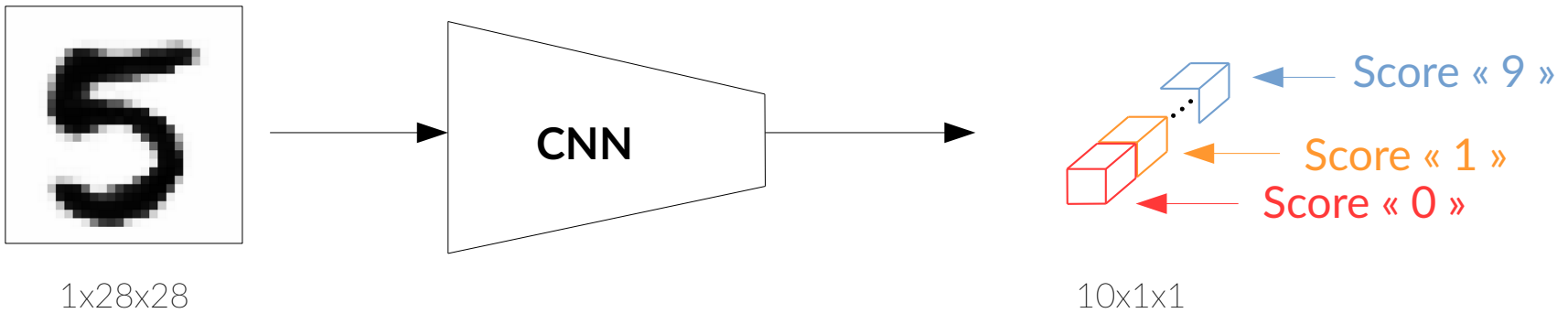
Cas 1 : Extraire une information **globale** présente dans l'image d'entrée



II)

Architectures de CNN : Deux cas extrêmes

Cas 1 : Extraire une information **globale** présente dans l'image d'entrée



Réduction **progressive** de la résolution

↓

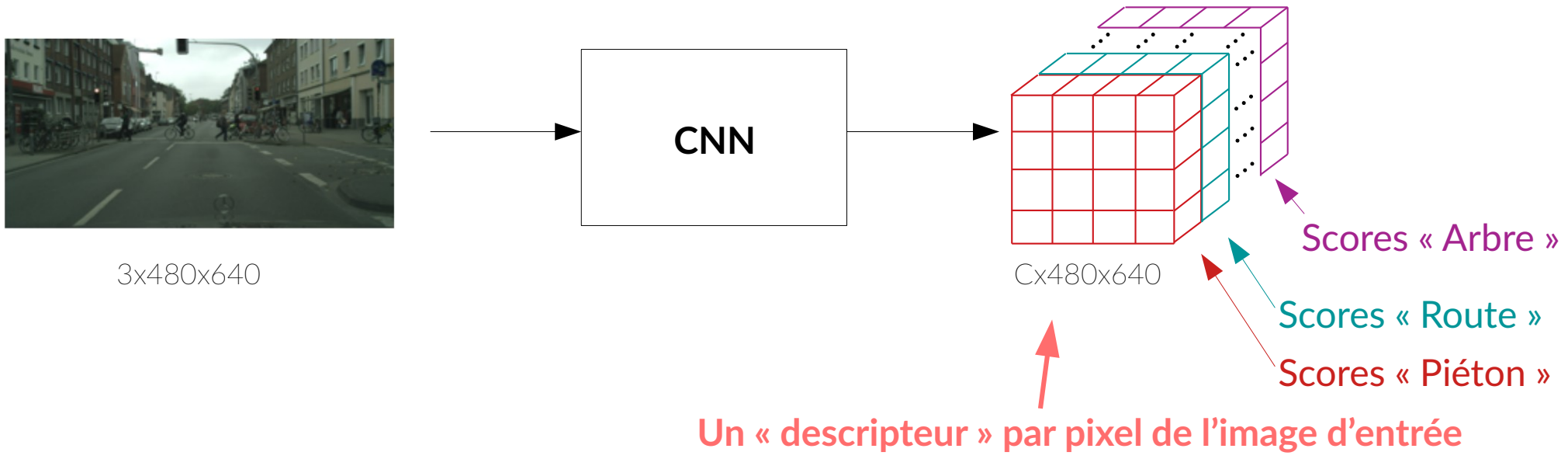
Utilisation de couches de conv ou pooling avec stride = 2

II)

Architectures de CNN : Deux cas extrêmes

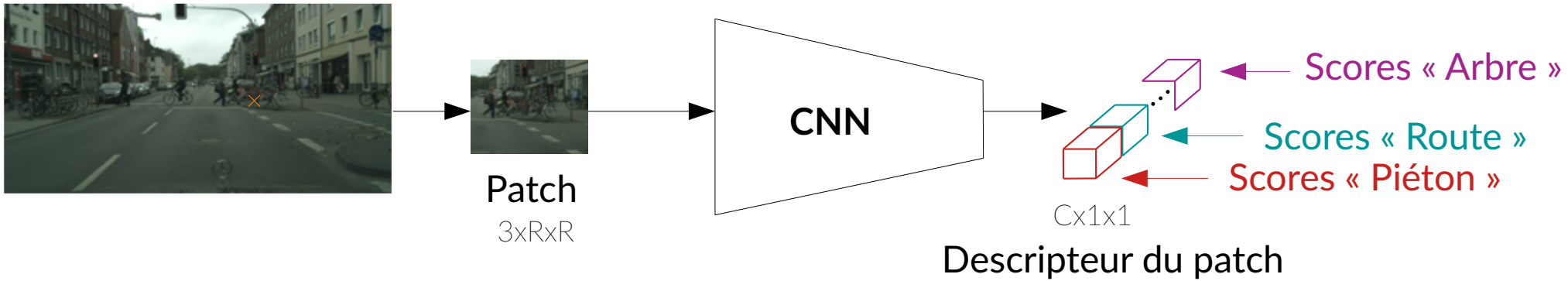
Cas 1 : Extraire une information **globale** présente dans l'image d'entrée

Cas 2 : Extraire une information **pour chaque pixel** de l'image d'entrée



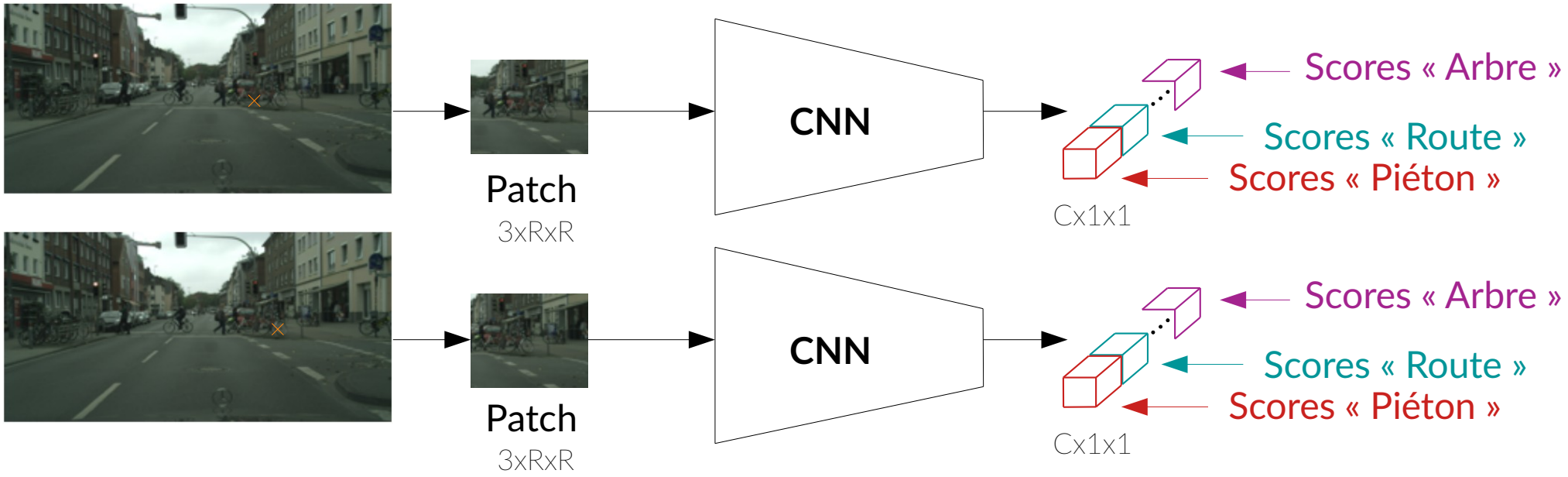
II)

Comment obtenir un descripteur pour chaque pixel ?



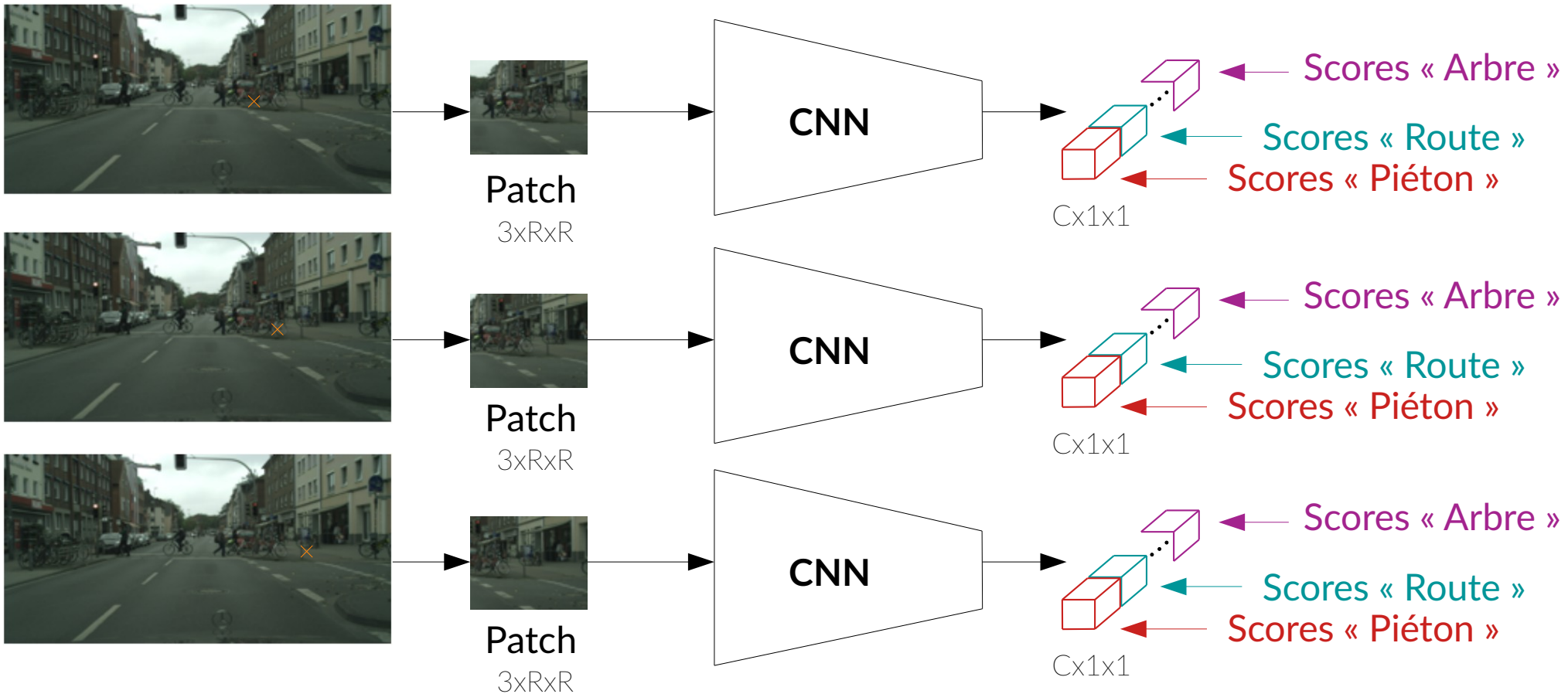
II)

Comment obtenir un descripteur pour chaque pixel ?



II)

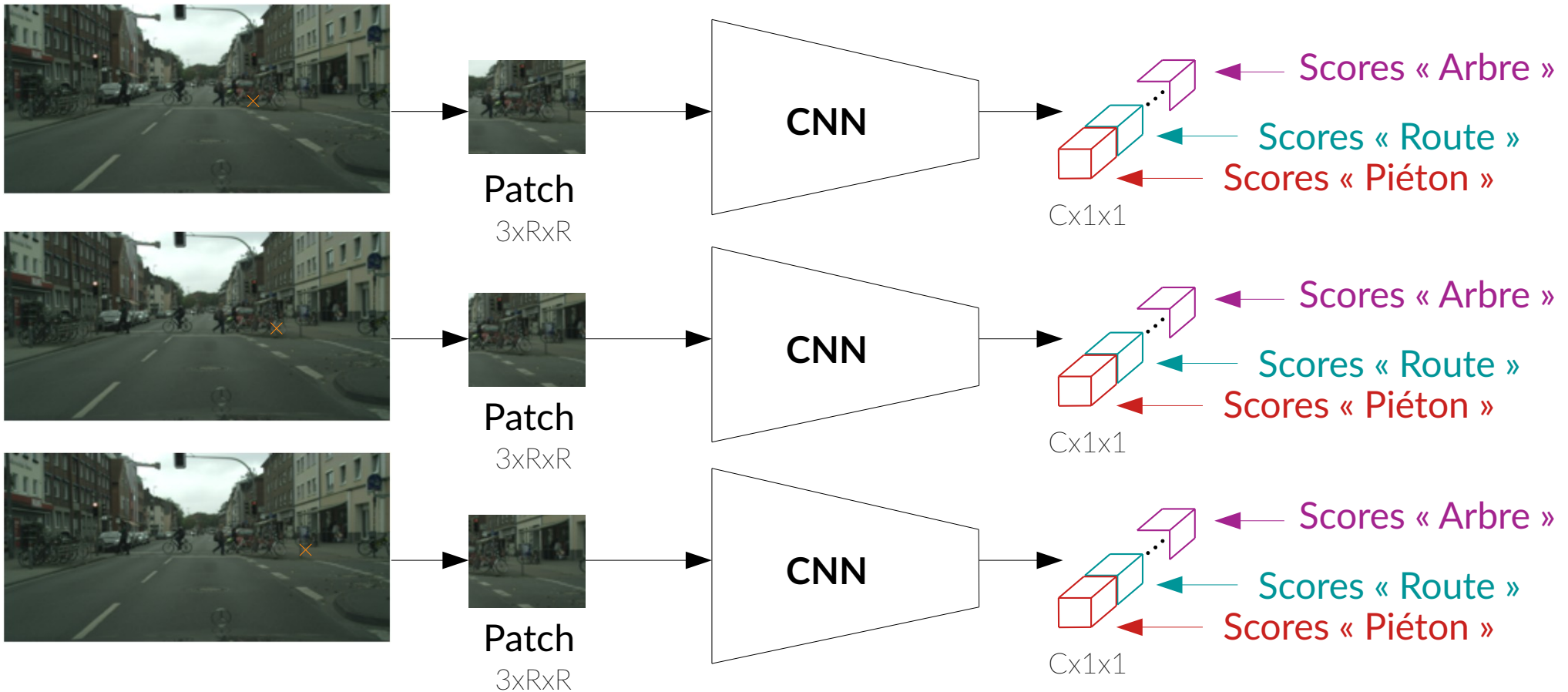
Comment obtenir un descripteur pour chaque pixel ?



Très inefficace en mémoire et en calculs !

II)

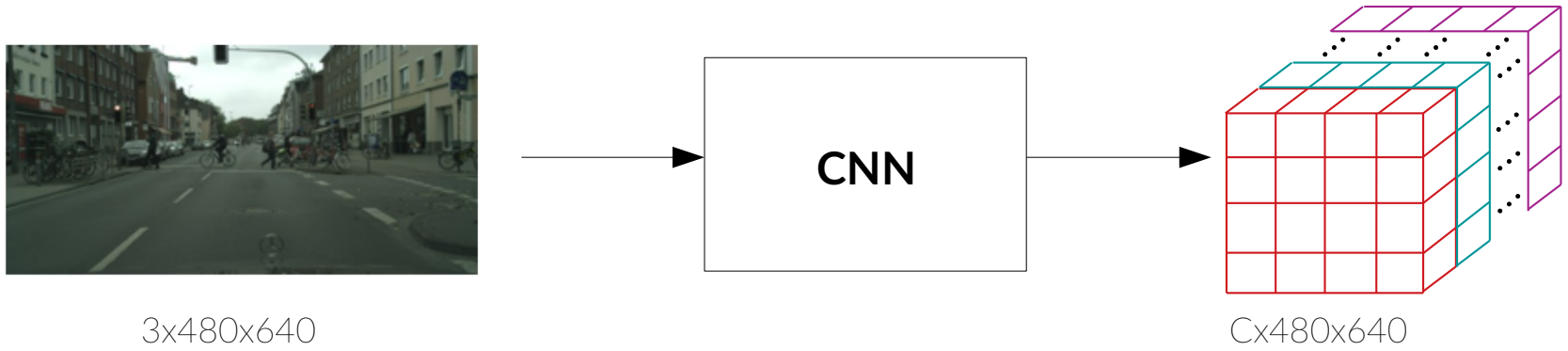
Comment obtenir un descripteur pour chaque pixel ?



Très inefficace en mémoire et en calculs ! → Utilisation d'une architecture entièrement convolutive

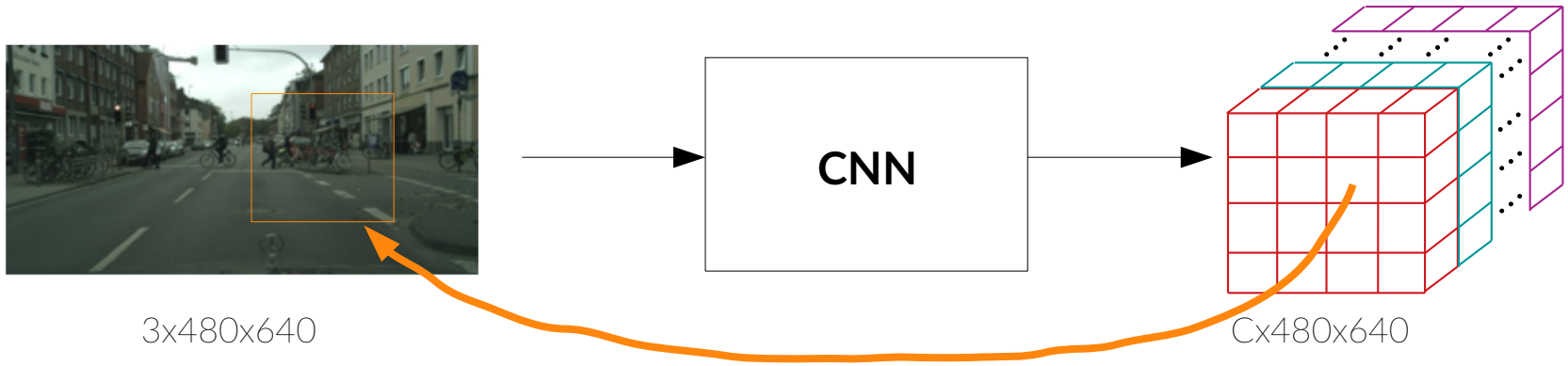
II)

Architecture entièrement convolutive



II)

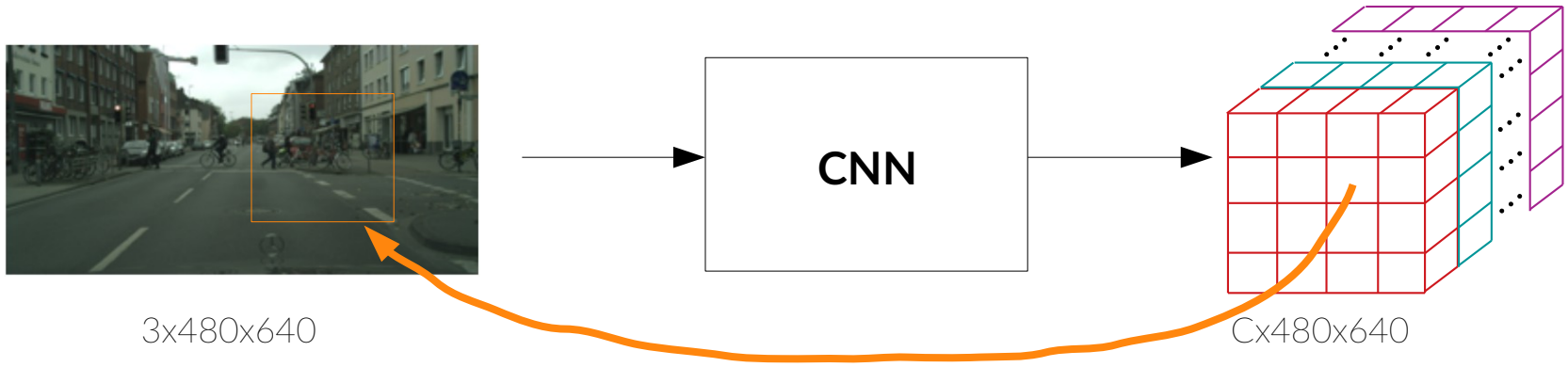
Architecture entièrement convolutive



Descripteur Cx1x1 d'un patch de taille RxR,
où RxR s'appelle le Champ Récepteur du CNN (« Receptive Field »)

II)

Architecture entièrement convolutive

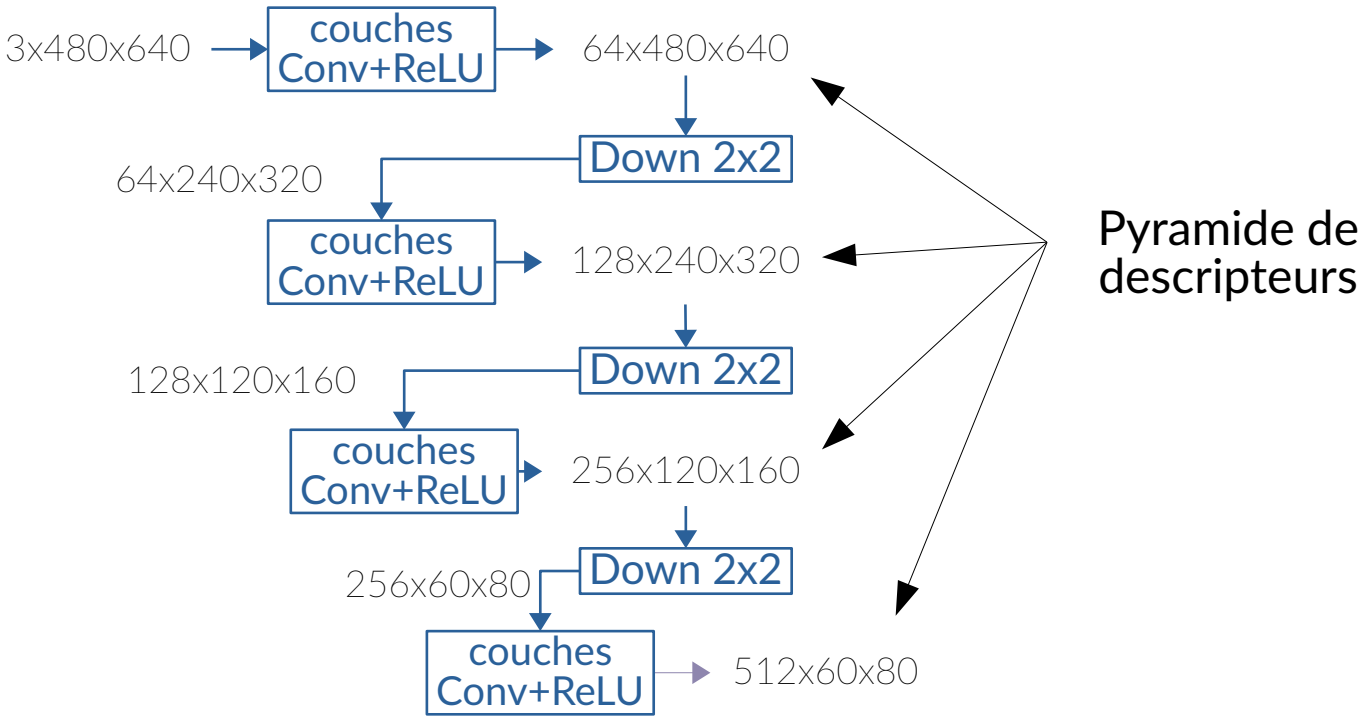


Descripteur Cx1x1 d'un patch de taille RxR,
où RxR s'appelle le Champ Récepteur du CNN (« Receptive Field »)

Comment avoir un grand champ récepteur tout en restant raisonnable en mémoire et temps de calculs ?

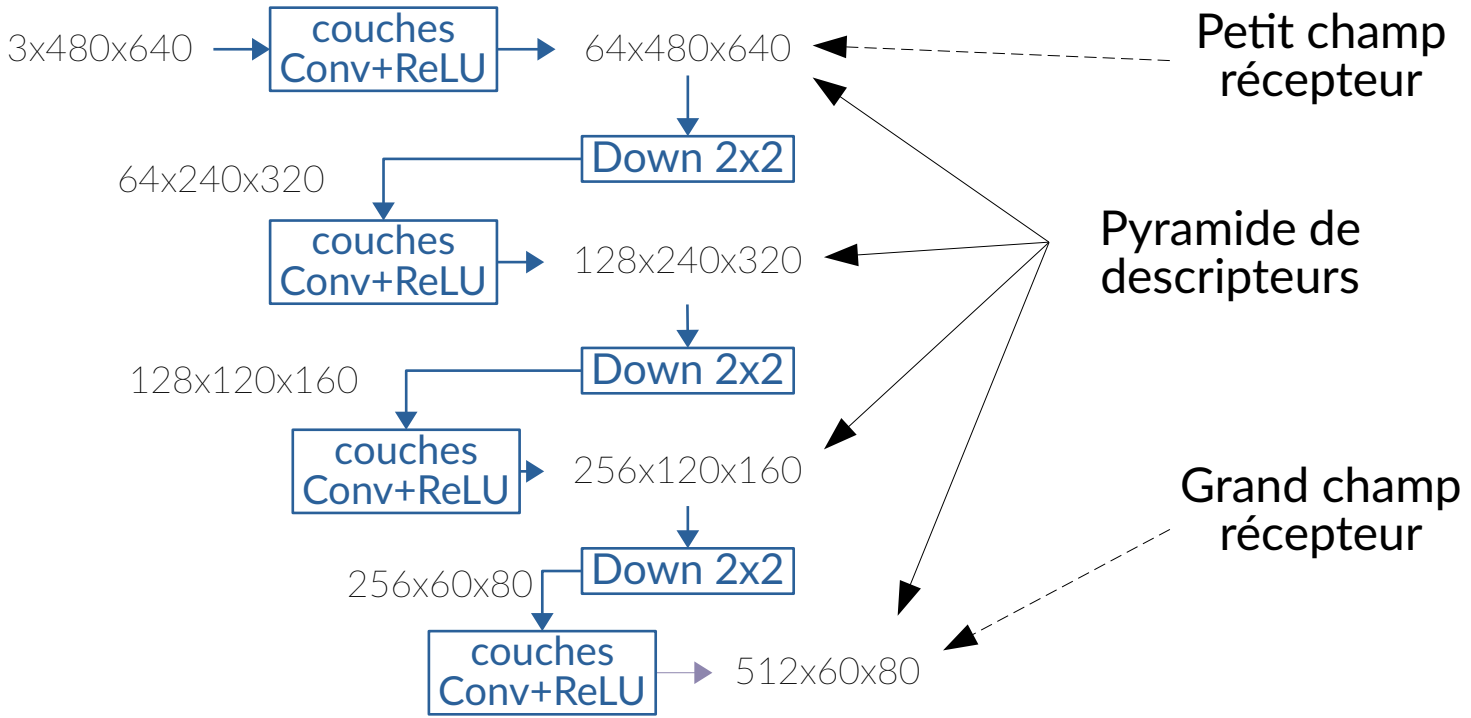
II)

U-Net, également appelé « Feature Pyramid Network » (FPN)



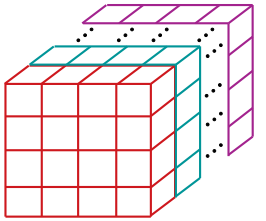
II)

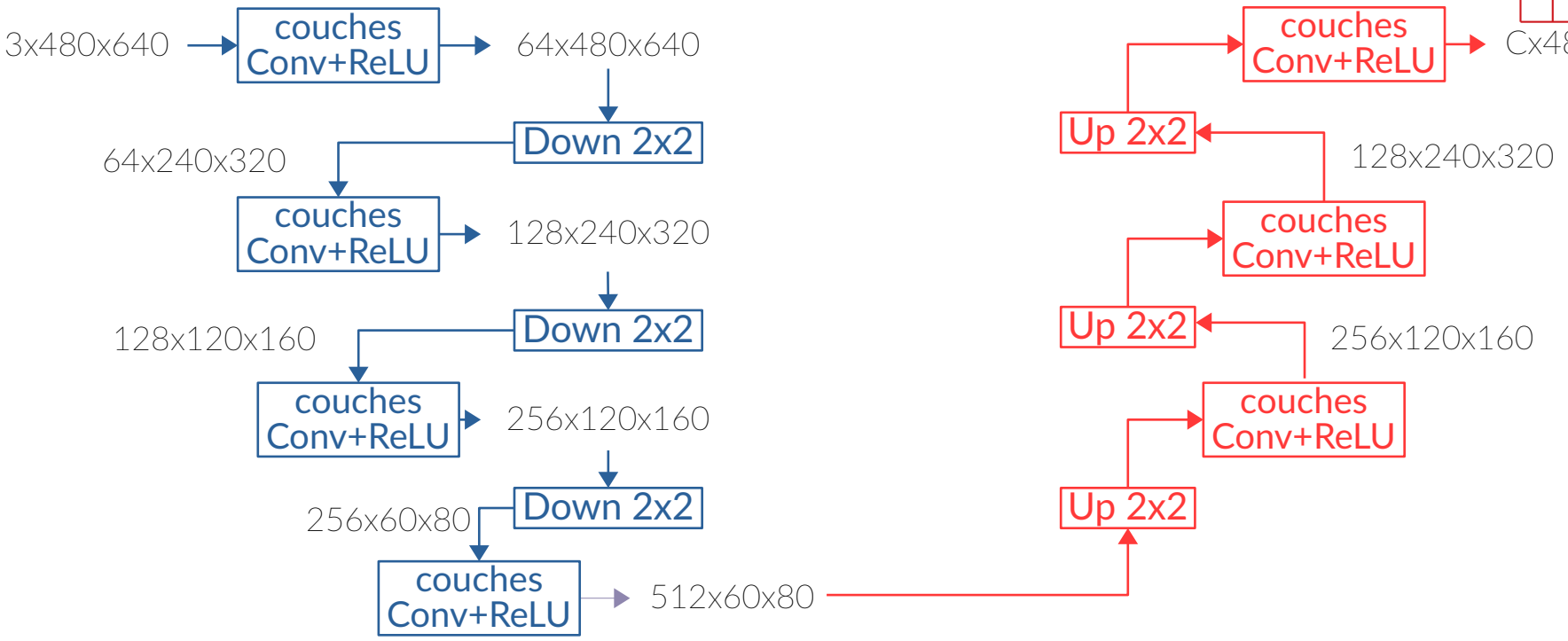
U-Net, également appelé « Feature Pyramid Network » (FPN)



II)

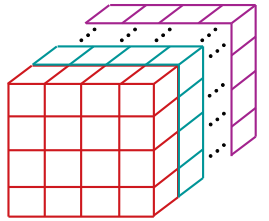
U-Net, également appelé « Feature Pyramid Network » (FPN)





II)

U-Net, également appelé « Feature Pyramid Network » (FPN)



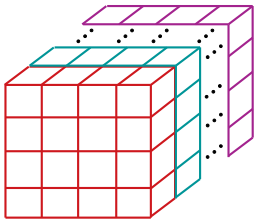
Les descripteurs « fins »
ne sont pas utilisés !

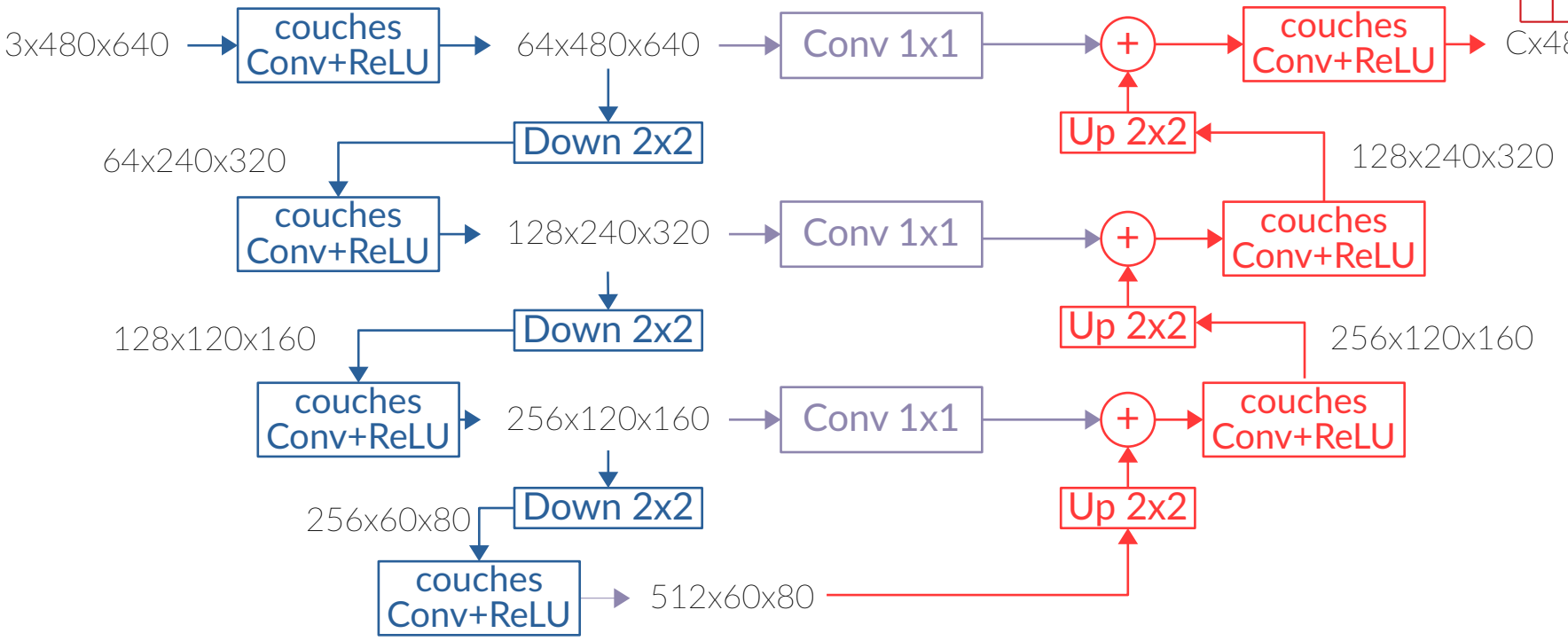
3x480x640 → couches Conv+ReLU → 64x480x640 → Down 2x2 → 64x240x320 → couches Conv+ReLU → 128x240x320 → Down 2x2 → 128x120x160 → couches Conv+ReLU → 256x120x160 → Down 2x2 → 256x60x80 → couches Conv+ReLU → 512x60x80

512x60x80 → Up 2x2 → couches Conv+ReLU → 256x120x160 → Up 2x2 → couches Conv+ReLU → 128x240x320 → Up 2x2 → couches Conv+ReLU → Cx480x640

II)

U-Net, également appelé « Feature Pyramid Network » (FPN)





II)

U-Net, également appelé « Feature Pyramid Network » (FPN)