

Réseaux de neurones à convolution

Guillaume Bourmaud

Rappel des ingrédients du « Deep Learning »

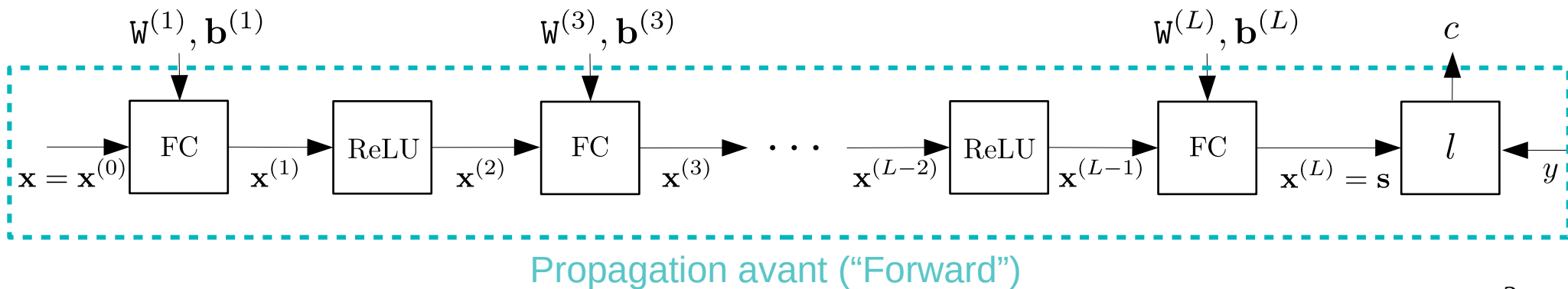
1) Grande base de données étiquetées

2) « Bonne » architecture de réseau de neurones profond

- ▶ « Perceptron » multicouche, Réseau de neurones à convolution, Transformer
- ▶ Et optimisation par descente de gradient stochastique (AdamW, etc.)

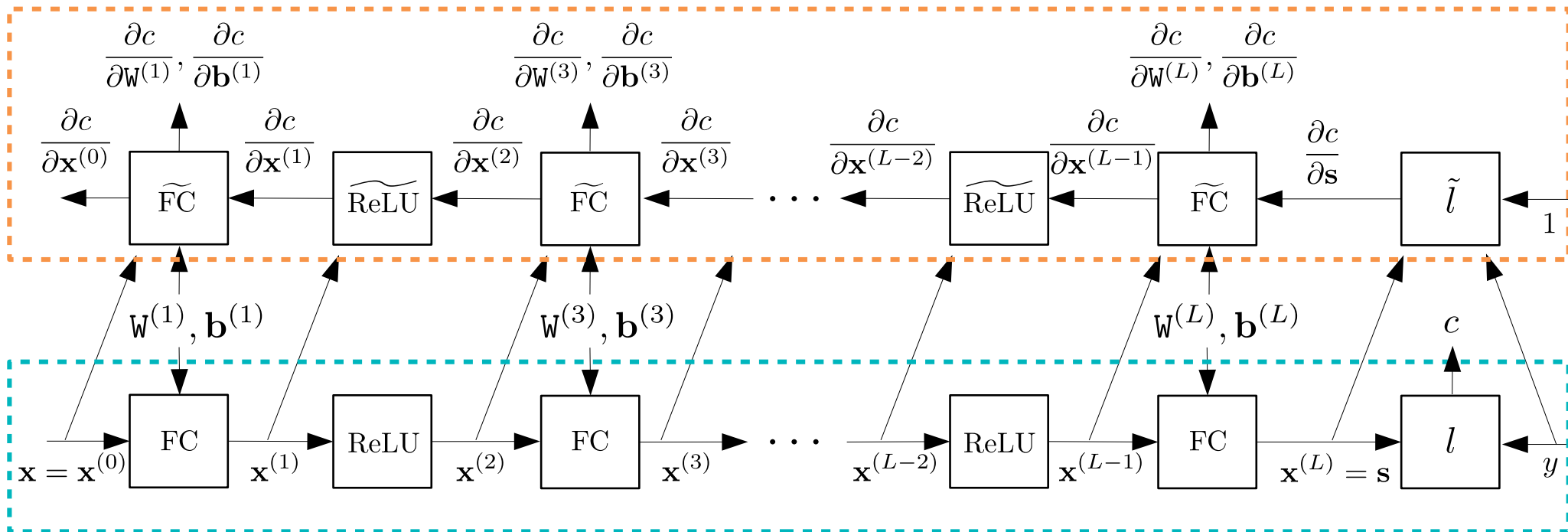
3) Grande capacité de calculs en parallèle (GPU)

Rappel MLP + Rétropropagation du gradient



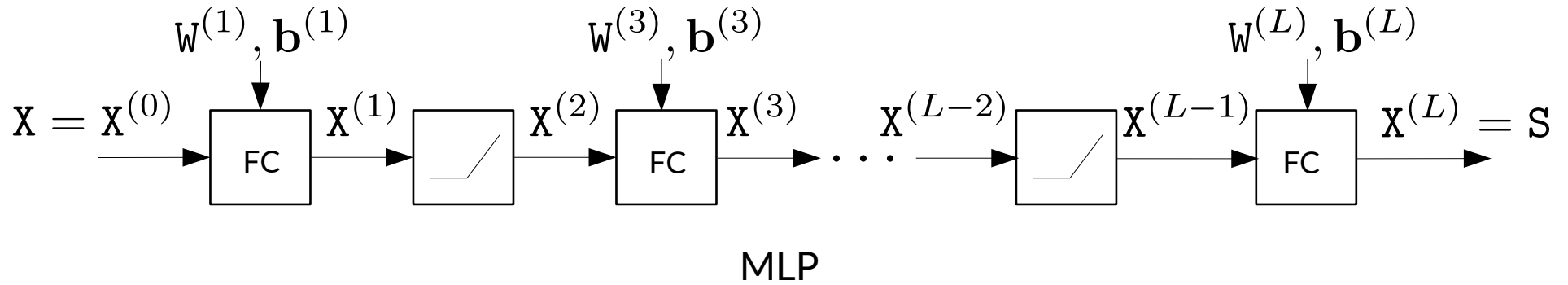
Rappel MLP + Rétropropagation du gradient

Rétropropagation ("Backward")

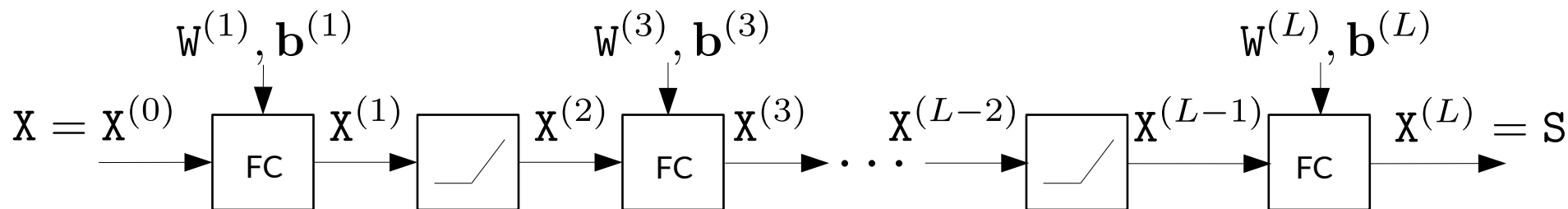


Propagation avant ("Forward")

Introduction



Introduction



MLP

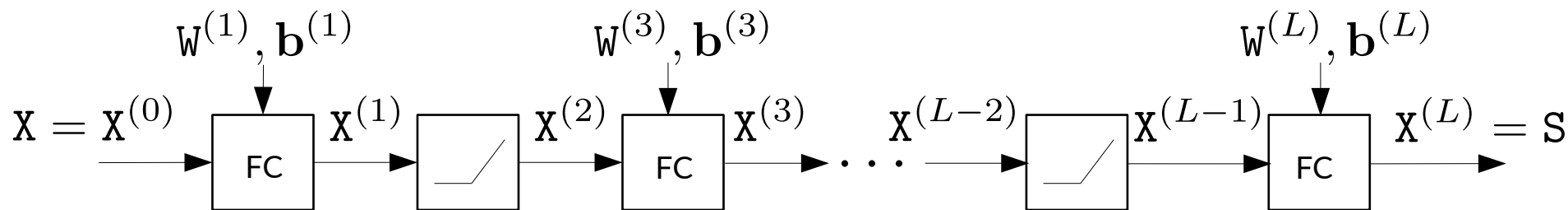
– FC (transformations affines générales)

+ Conv (transformations affines spécifiques)

=

CNN

Introduction



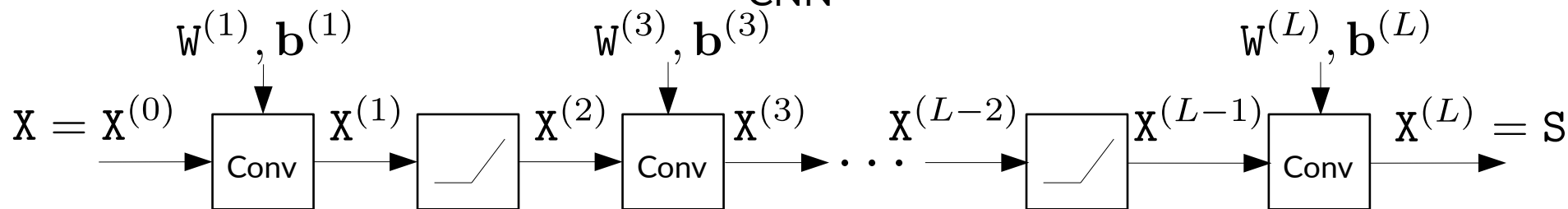
MLP

– FC (transformations affines générales)

+ Conv (transformations affines spécifiques)

=

CNN



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

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

I)

Limites d'une transformation affine générale (FC)

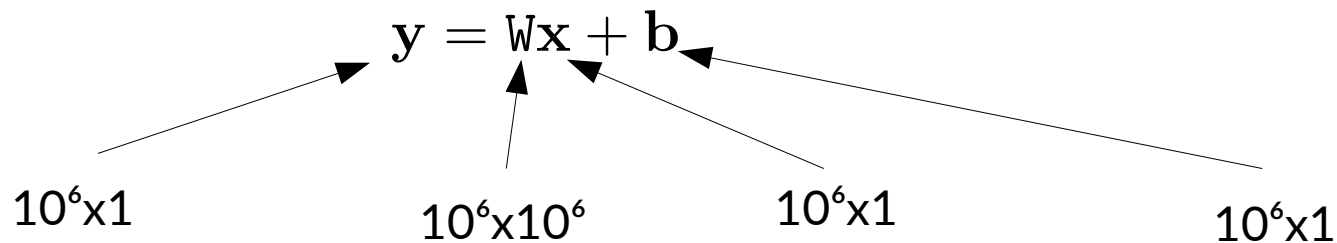
H=480



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



I)

Limites d'une transformation affine générale (FC)

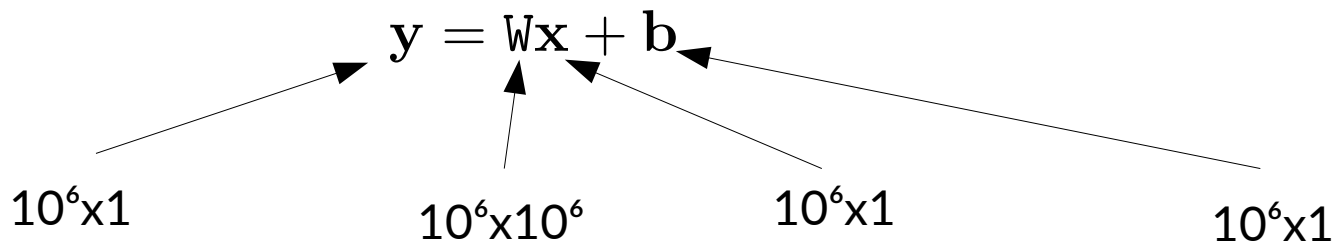
H=480



W=640

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

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



Occupation mémoire de W : 4 octets (32 bits) $\times 10^6 \times 10^6 = 4\text{To}$.

Nombre de « multiplications+additions » également très élevé.

I)

Opération de « convolution » = transformation affine spécifique

“Fully Connected”

=

Transformation affine générale

$$\begin{bmatrix} W_{11} & W_{12} & W_{13} & W_{14} \\ W_{21} & W_{22} & W_{23} & W_{24} \\ W_{31} & W_{32} & W_{33} & W_{34} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} + \begin{bmatrix} b_1 \\ b_2 \\ b_3 \end{bmatrix}$$

I)

Opération de « convolution » = transformation affine spécifique

“Fully Connected”

=

Transformation affine générale

$$\begin{bmatrix} W_{11} & W_{12} & W_{13} & W_{14} \\ W_{21} & W_{22} & W_{23} & W_{24} \\ W_{31} & W_{32} & W_{33} & W_{34} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} + \begin{bmatrix} b_1 \\ b_2 \\ b_3 \end{bmatrix}$$

Localement connecté à un
voisinage de taille 2

$$\begin{bmatrix} W_{11} & W_{12} & 0 & 0 \\ 0 & W_{22} & W_{23} & 0 \\ 0 & 0 & W_{33} & W_{34} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} + \begin{bmatrix} b_1 \\ b_2 \\ b_3 \end{bmatrix}$$

I)

Opération de « convolution » = transformation affine spécifique

“Fully Connected”
=
Transformation affine générale

$$\begin{bmatrix} W_{11} & W_{12} & W_{13} & W_{14} \\ W_{21} & W_{22} & W_{23} & W_{24} \\ W_{31} & W_{32} & W_{33} & W_{34} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} + \begin{bmatrix} b_1 \\ b_2 \\ b_3 \end{bmatrix}$$

Localement connecté à un
voisinage de taille 2

$$\begin{bmatrix} W_{11} & W_{12} & 0 & 0 \\ 0 & W_{22} & W_{23} & 0 \\ 0 & 0 & W_{33} & W_{34} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} + \begin{bmatrix} b_1 \\ b_2 \\ b_3 \end{bmatrix}$$

« Convolution » avec filtre de taille 2
=
Équivariance par translation

$$\begin{bmatrix} W_{11} & W_{12} & 0 & 0 \\ 0 & W_{11} & W_{12} & 0 \\ 0 & 0 & W_{11} & W_{12} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} + \begin{bmatrix} b \\ b \\ b \end{bmatrix}$$

I)

Opération de « convolution » = transformation affine spécifique

“Fully Connected”
=
Transformation affine générale

$$\begin{bmatrix} W_{11} & W_{12} & W_{13} & W_{14} \\ W_{21} & W_{22} & W_{23} & W_{24} \\ W_{31} & W_{32} & W_{33} & W_{34} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} + \begin{bmatrix} b_1 \\ b_2 \\ b_3 \end{bmatrix}$$

Localement connecté à un
voisinage de taille 2

$$\begin{bmatrix} W_{11} & W_{12} & 0 & 0 \\ 0 & W_{22} & W_{23} & 0 \\ 0 & 0 & W_{33} & W_{34} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} + \begin{bmatrix} b_1 \\ b_2 \\ b_3 \end{bmatrix}$$

« Convolution » avec filtre de taille 2
=
Équivariance par translation

$$\begin{bmatrix} W_{11} & W_{12} & 0 & 0 \\ 0 & W_{11} & W_{12} & 0 \\ 0 & 0 & W_{11} & W_{12} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} + \begin{bmatrix} b \\ b \\ b \end{bmatrix}$$

Bonne idée pour tous les types de signaux (son, image, etc.), il faut que la notion de voisinage ait un sens !

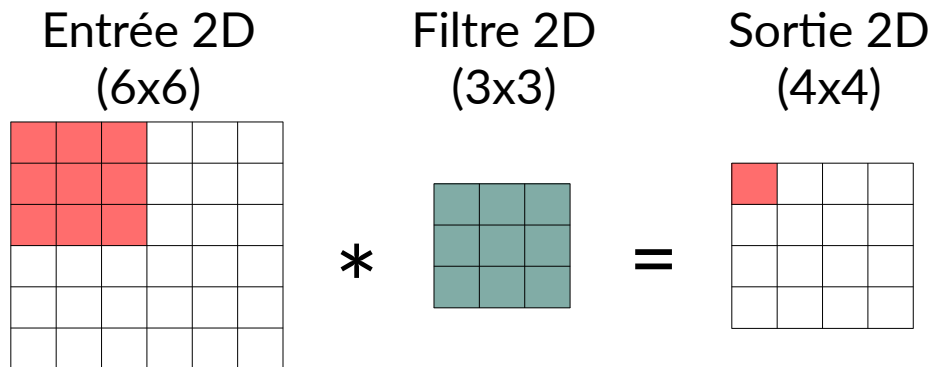
→ Beaucoup moins de paramètres à stocker

→ Beaucoup moins de « multiplications+additions »

I)

Opération de « convolution » en 2D

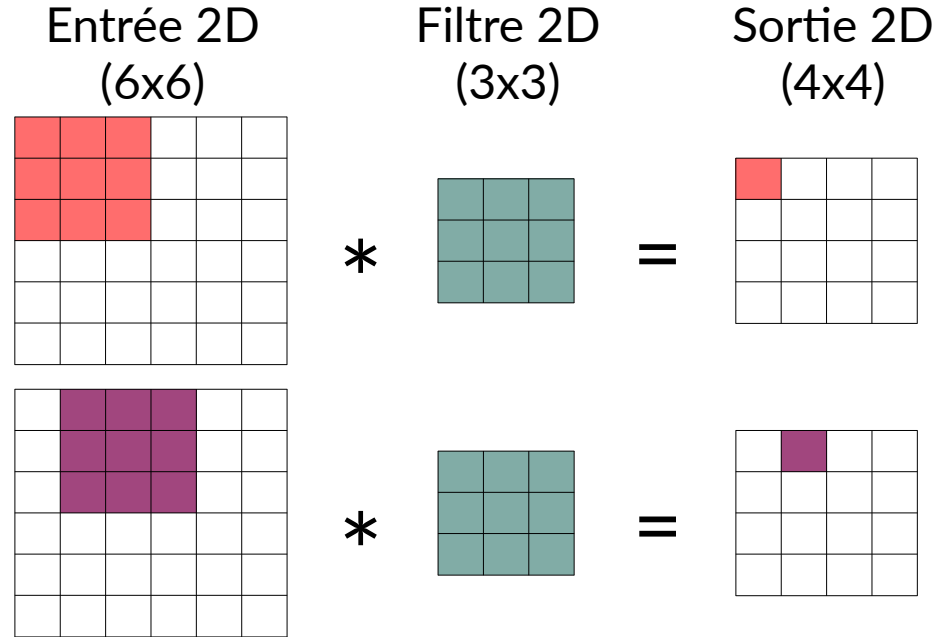
En fait, il s'agit d'une
intercorrélacion



I)

Opération de « convolution » en 2D

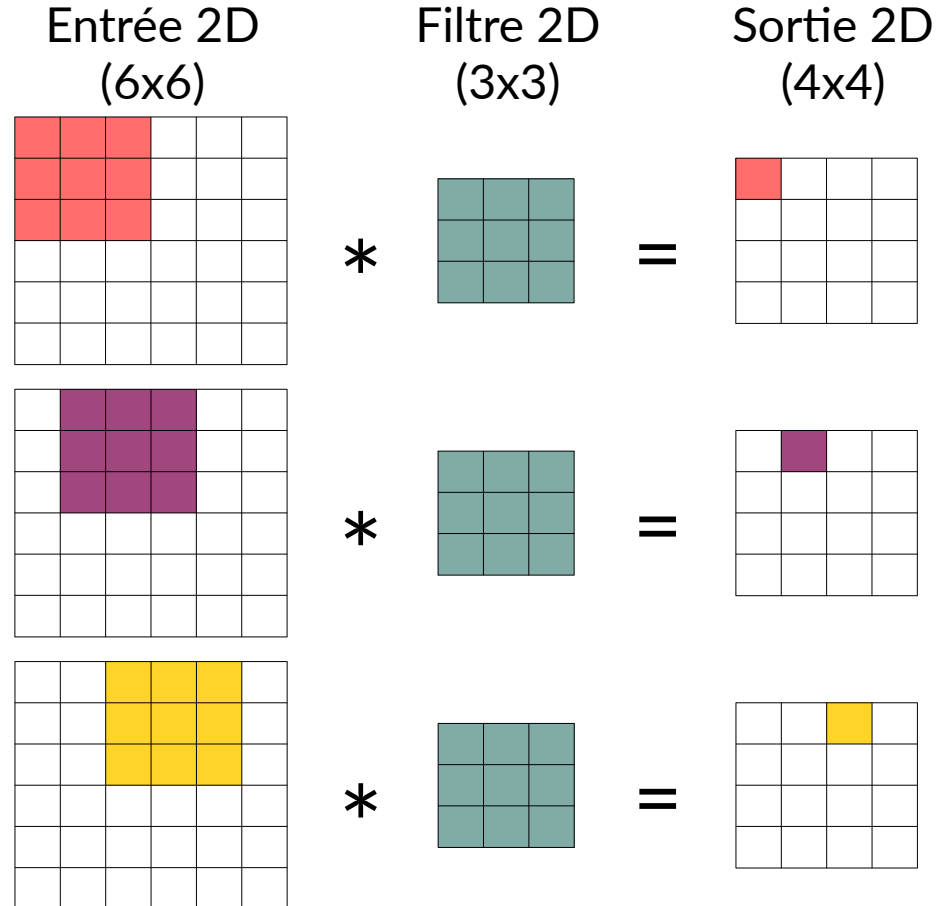
En fait, il s'agit d'une
intercorrélacion



I)

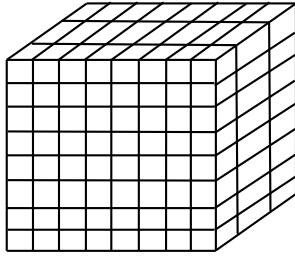
Opération de « convolution » en 2D

En fait, il s'agit d'une
intercorrélacion



I)

Couche de convolution 2D : un seul filtre



$X^{(0)}$

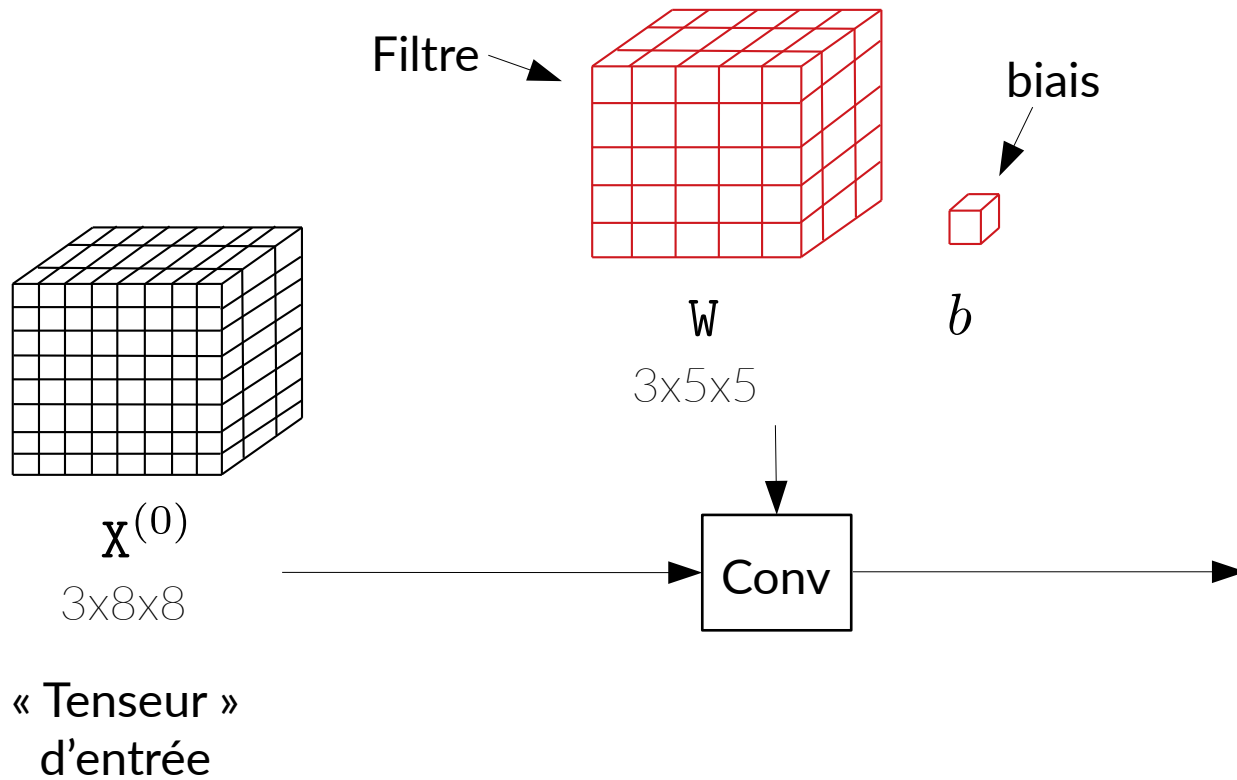
3x8x8

« Tenseur »
d'entrée

« Tenseur » = tableau multi-dimensionnel

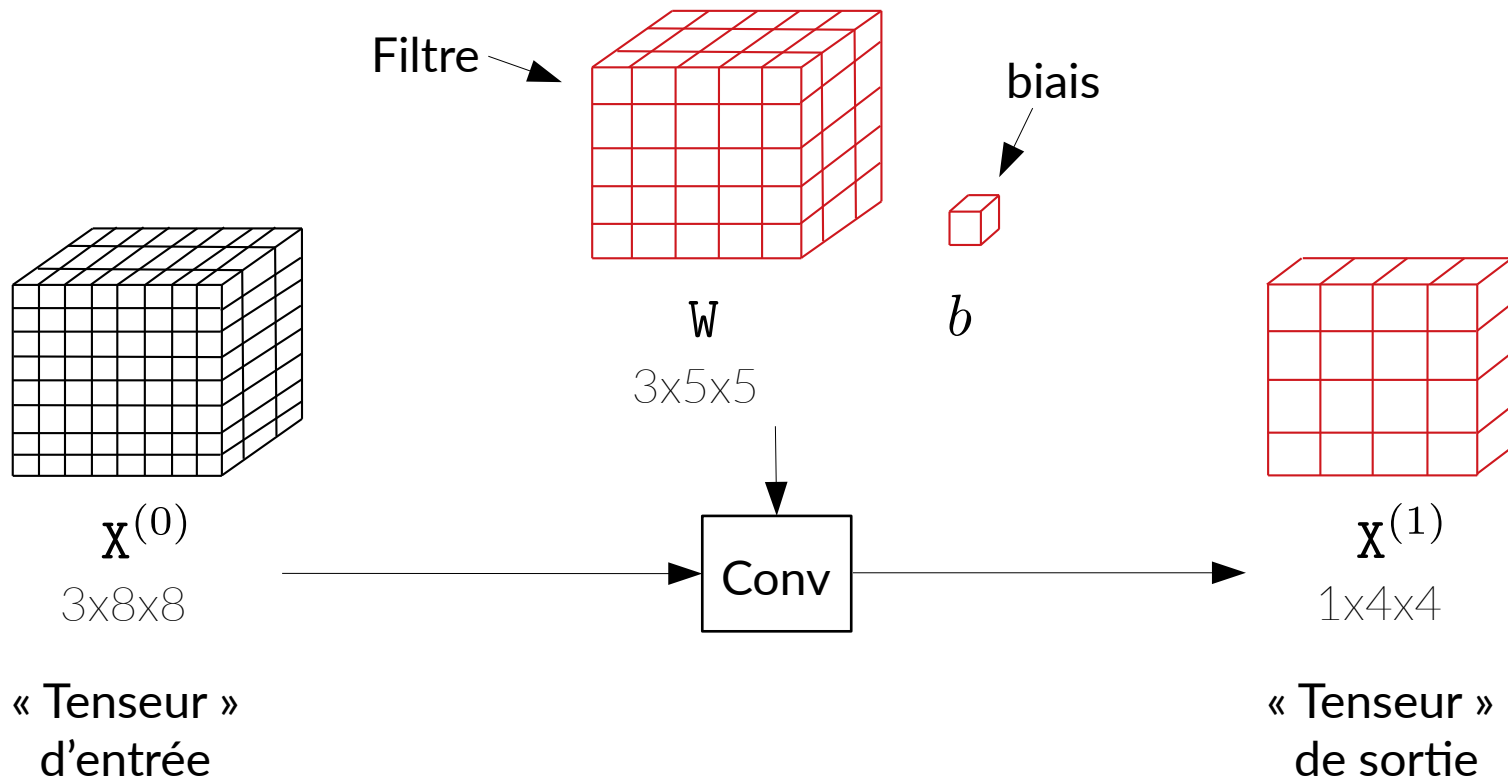
I)

Couche de convolution 2D : un seul filtre



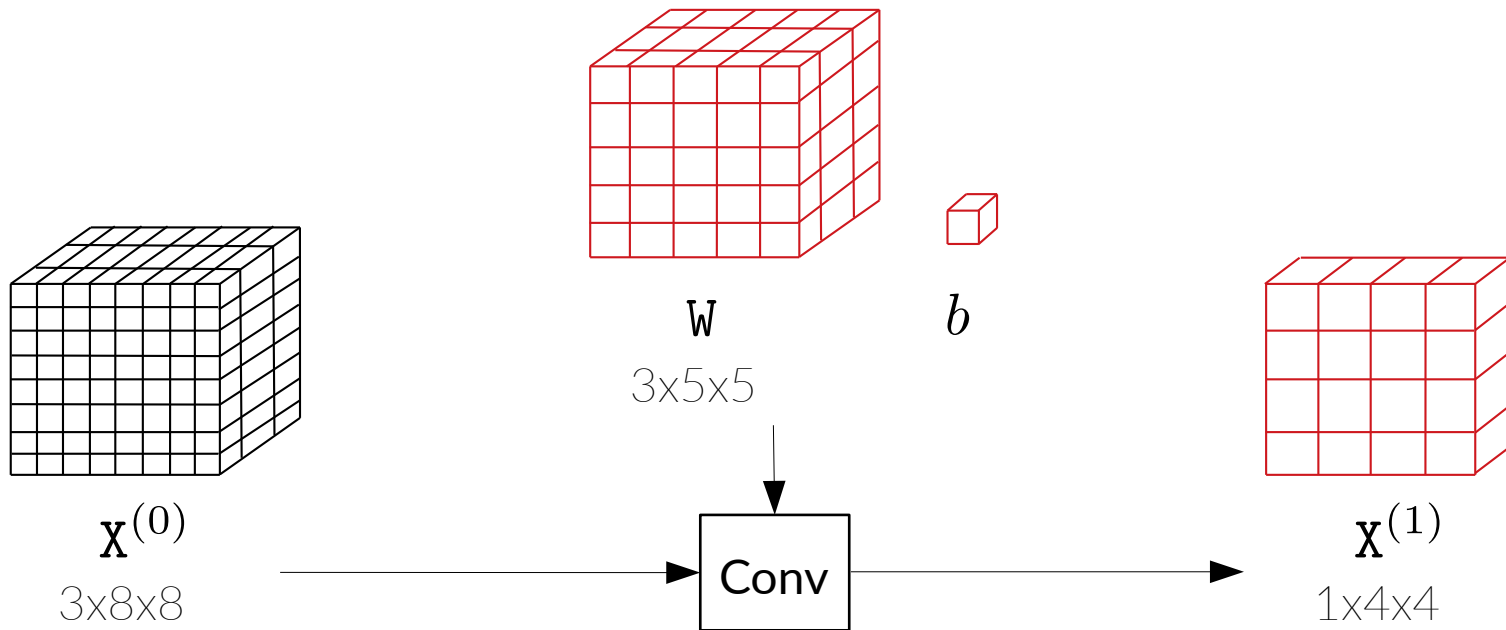
I)

Couche de convolution 2D : un seul filtre



I)

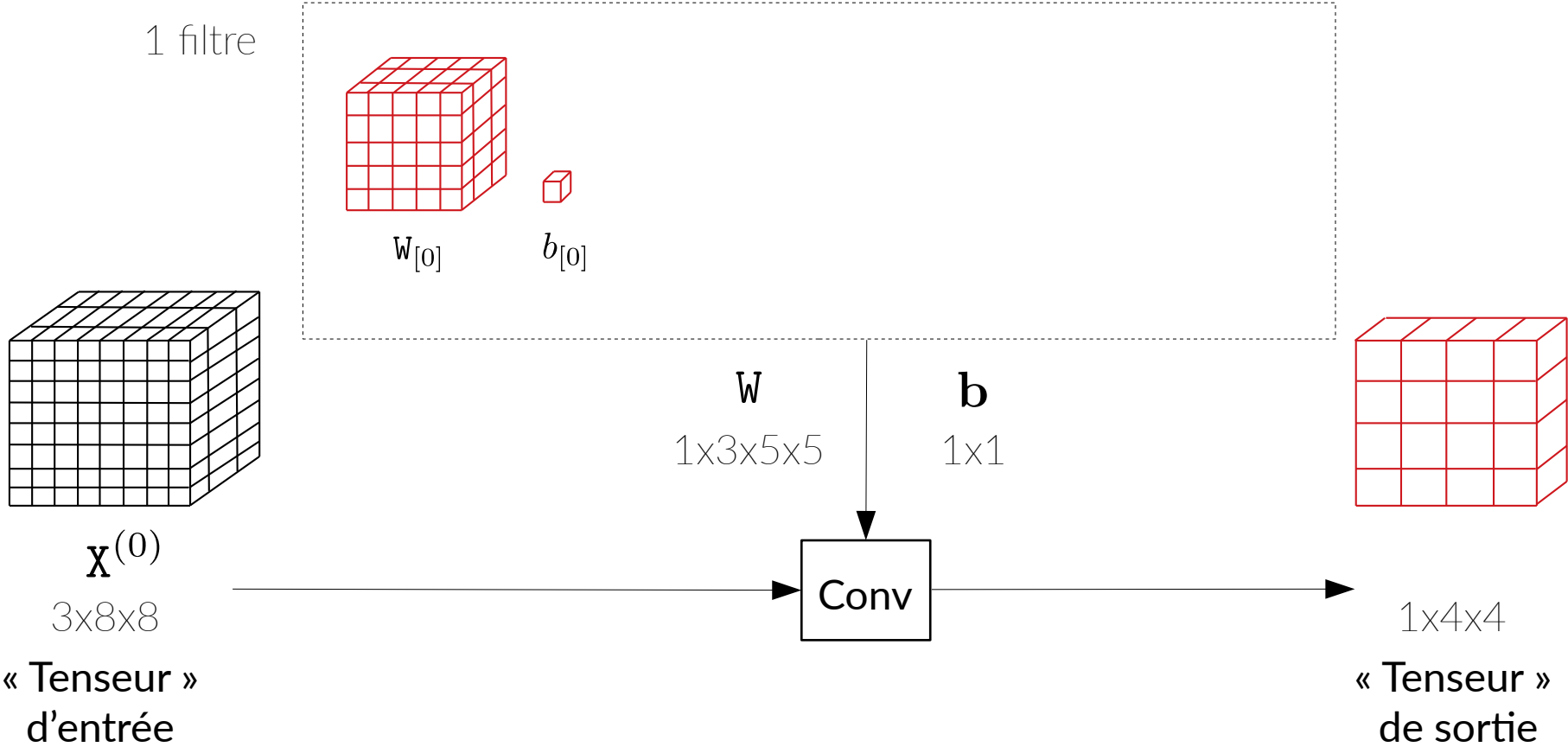
Couche de convolution 2D : un seul filtre



$$X_{i,j}^{(1)} = \sum_{k=0}^2 \sum_{m=0}^4 \sum_{n=0}^4 W_{k,m,n} X_{k,i+m,j+n}^{(0)} + b$$

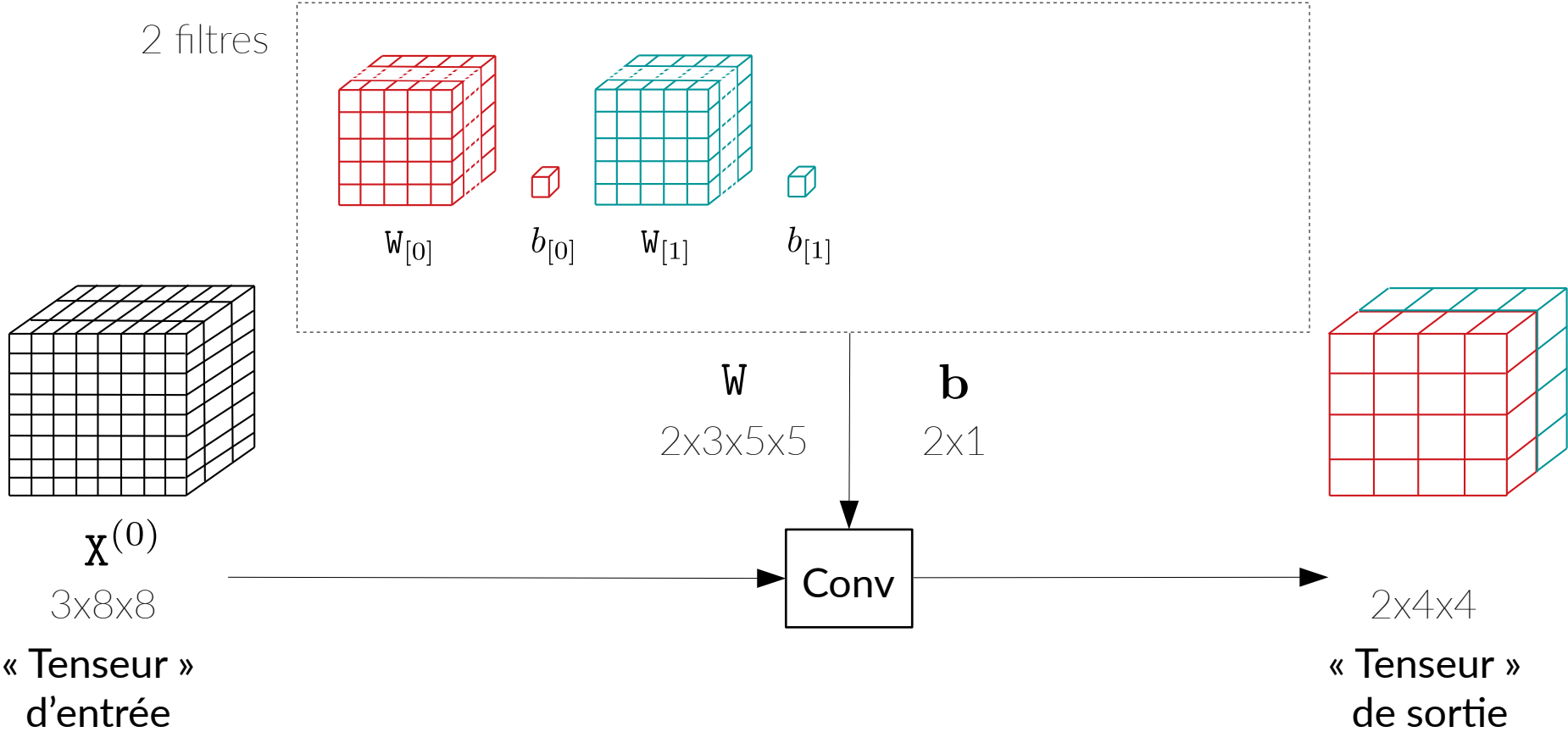
1)

Couche de convolution 2D : K filtres



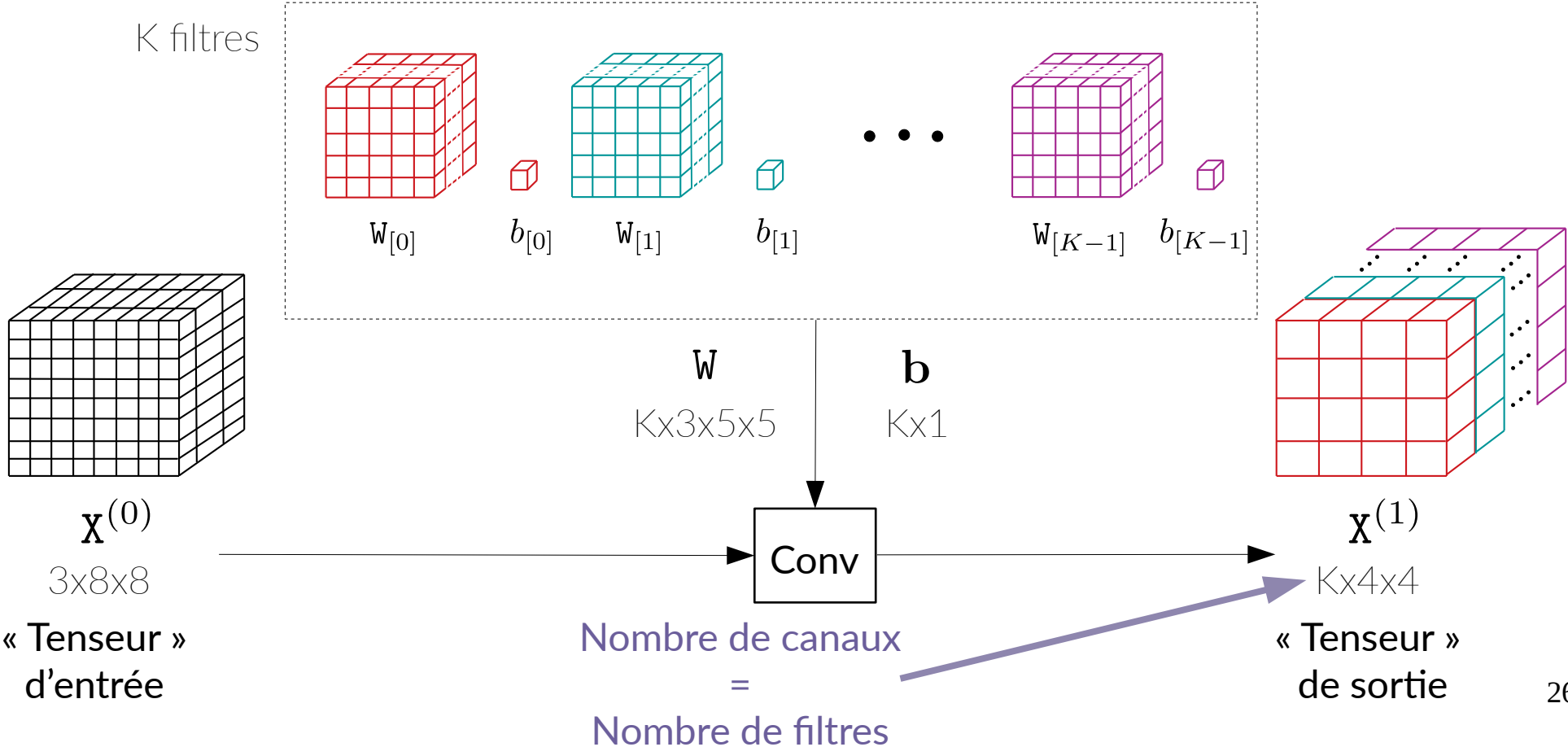
1)

Couche de convolution 2D : K filtres



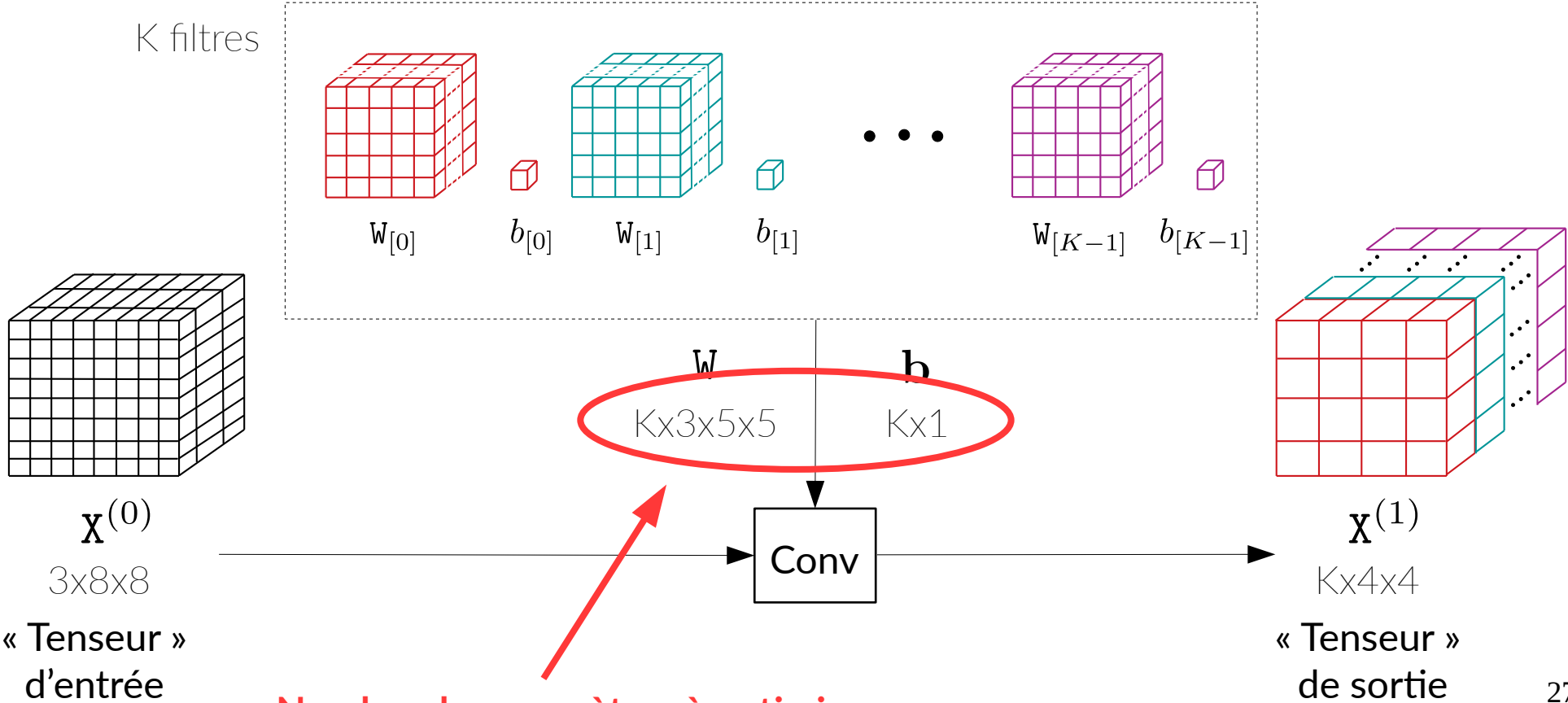
1)

Couche de convolution 2D : K filtres



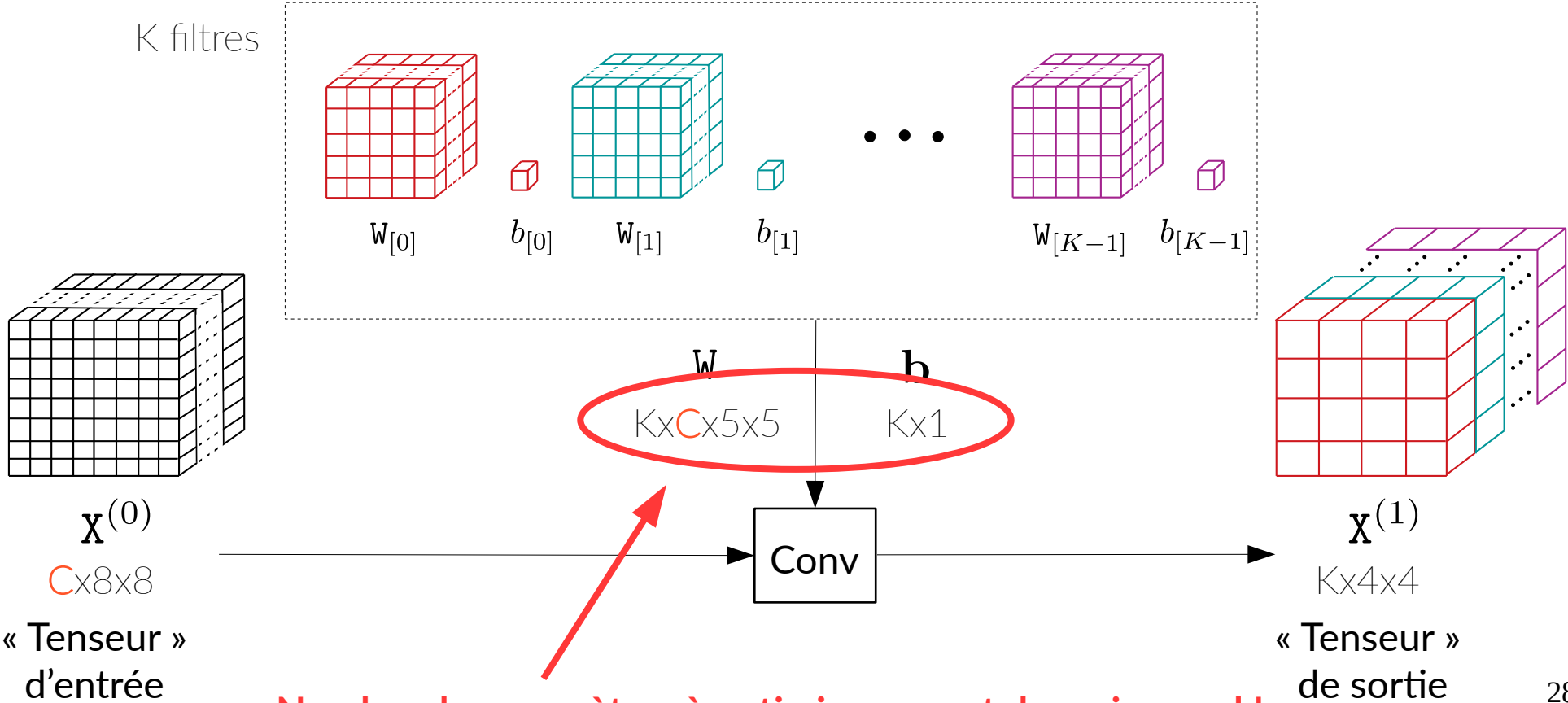
1)

Couche de convolution 2D : K filtres



1)

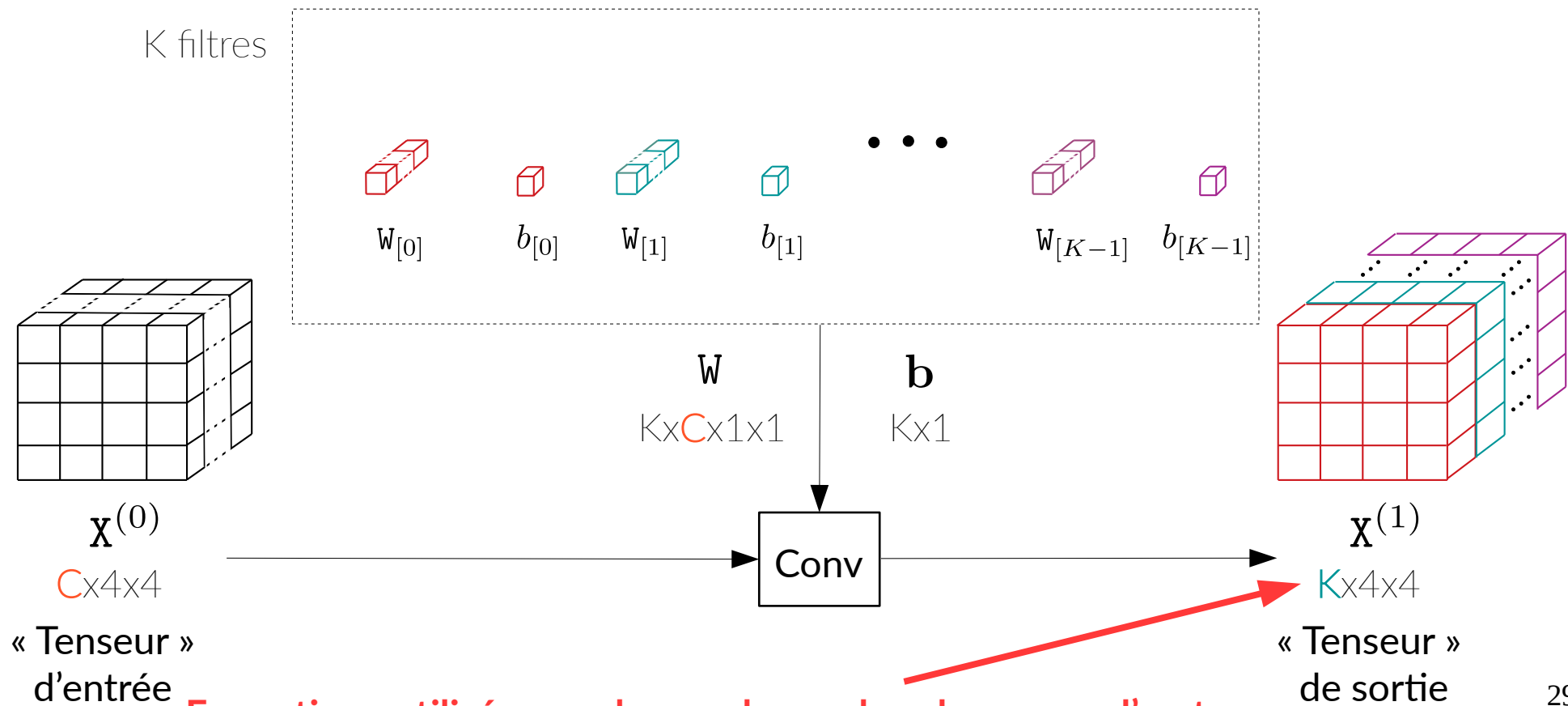
Couche de convolution 2D : K filtres



Nombre de paramètres à optimiser... peut devenir grand !

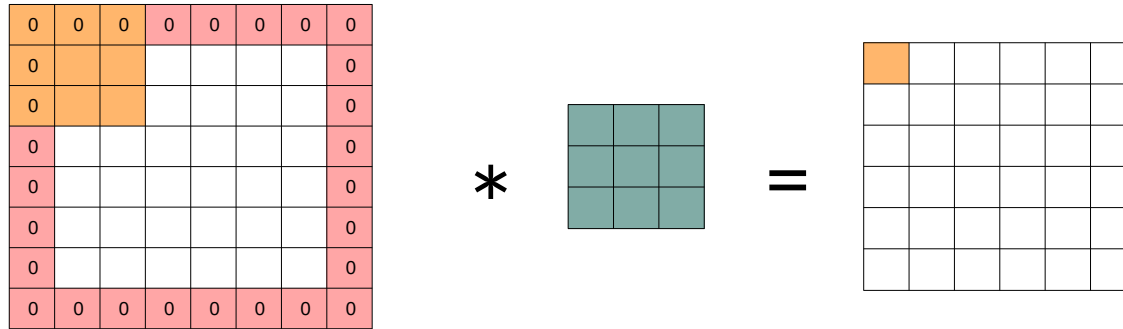
I)

Couche de convolution 2D avec filtre 1x1



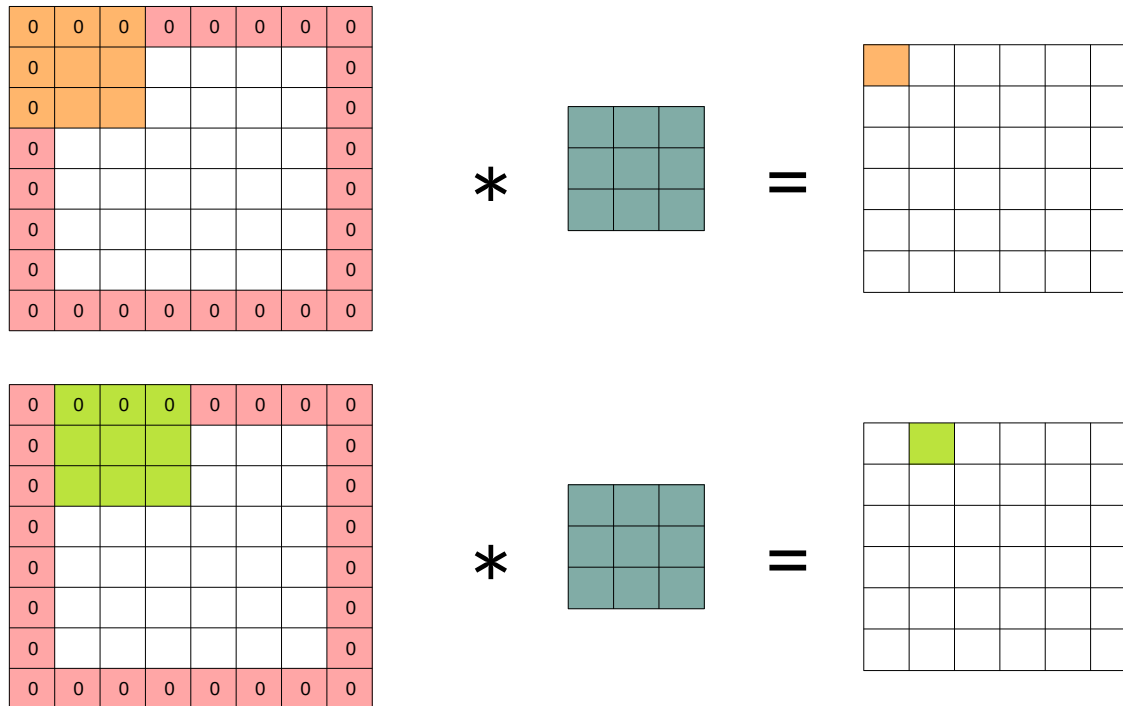
I)

« Zero padding »



I)

« Zero padding »

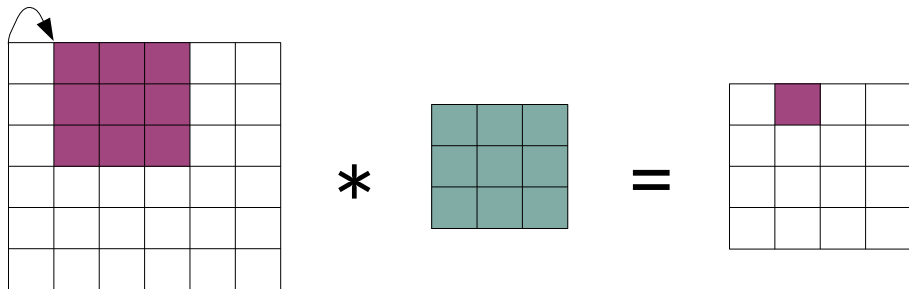
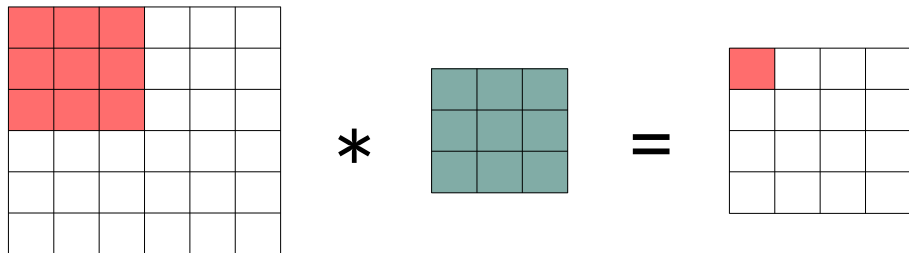


Permet de préserver la taille du tenseur d'entrée

I)

« Stride »

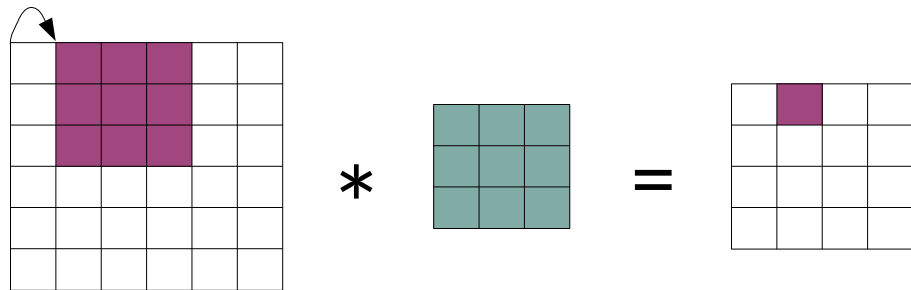
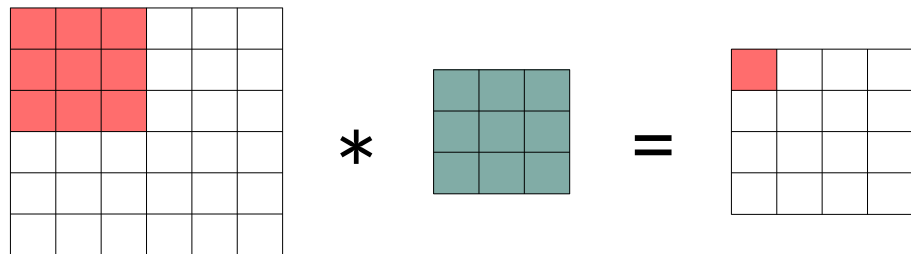
Exemple Stride = 1



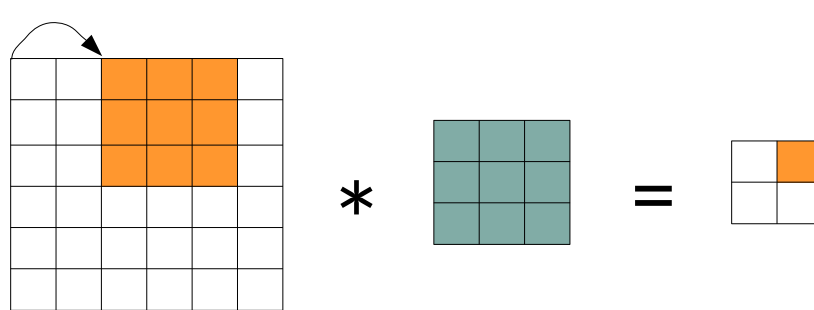
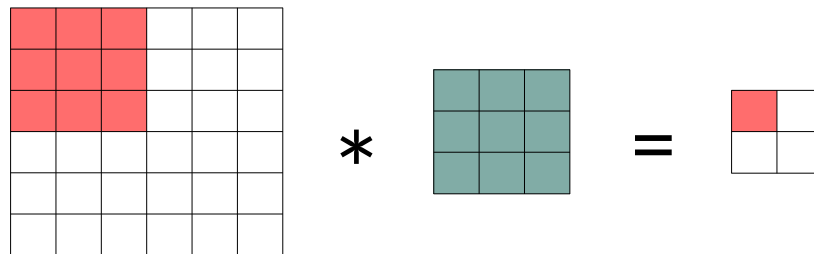
I)

« Stride »

Exemple Stride = 1



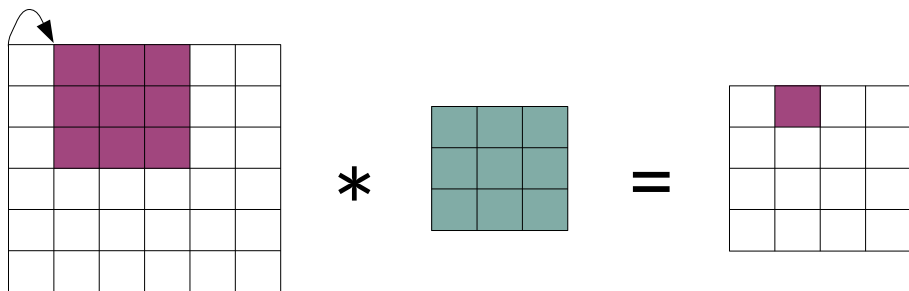
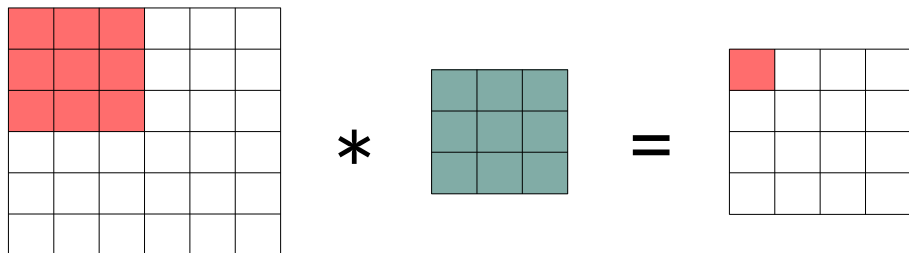
Exemple Stride = 2



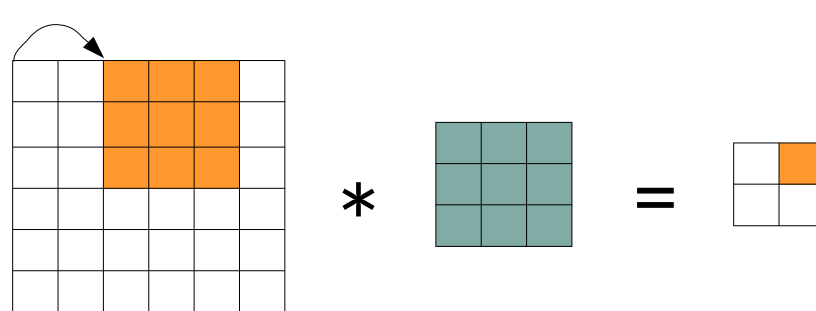
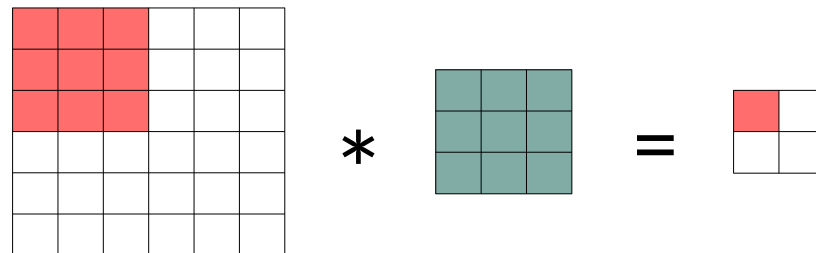
I)

« 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 1x1 ou 3x3, parfois 5x5 ou 7x7

Principaux hyperparamètres d'une couche de convolution

- Taille des filtres
 - En pratique toujours impair
 - Souvent 1x1 ou 3x3, parfois 5x5 ou 7x7
- Nombre de filtres
 - = Nombre de canaux souhaité en sortie

Principaux hyperparamètres d'une couche de convolution

- Taille des filtres
 - En pratique toujours impair
 - Souvent 1x1 ou 3x3, parfois 5x5 ou 7x7
- Nombre de filtres
 - = Nombre de canaux souhaité en sortie
- Quantité de zero-padding
 - Compense la taille du filtre si volonté de préserver la taille de l'entrée
 - 1x1 \rightarrow padding = 0, 3x3 \rightarrow padding = 1, 5x5 \rightarrow padding = 2

I)

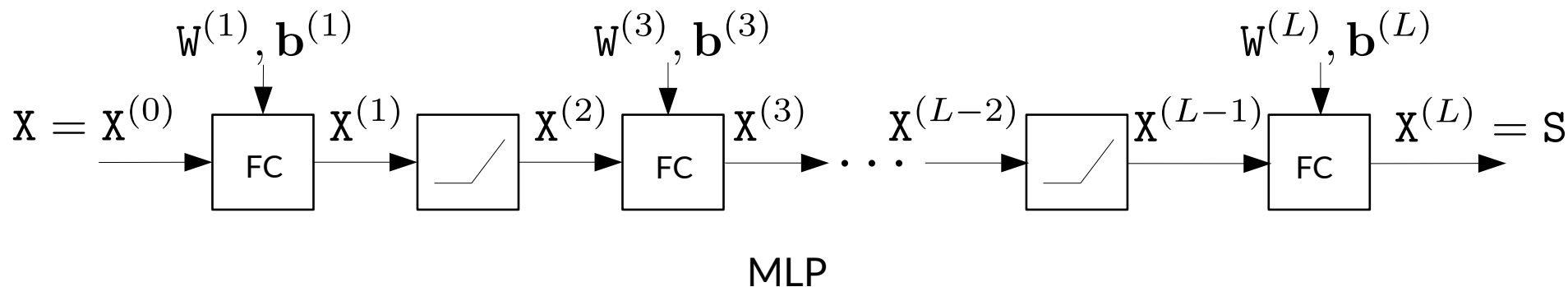
Principaux hyperparamètres d'une couche de convolution

- Taille des filtres
 - En pratique toujours impair
 - Souvent 1x1 ou 3x3, parfois 5x5 ou 7x7
- Nombre de filtres
 - = Nombre de canaux souhaité en sortie
- Quantité de zero-padding
 - Compense la taille du filtre si volonté de préserver la taille de l'entrée
 - 1x1 \rightarrow padding = 0, 3x3 \rightarrow padding = 1, 5x5 \rightarrow padding = 2
- Stride
 - = 1 si volonté de préserver la résolution de l'entrée
 - = 2 si volonté de réduire la résolution de l'entrée

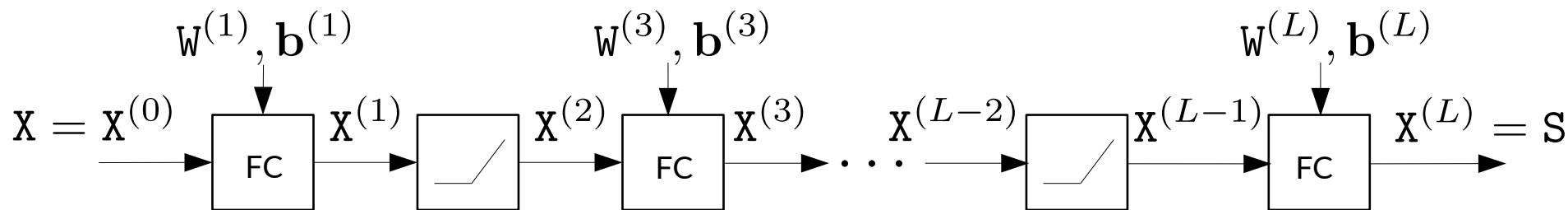
II) Réseau de neurones à convolution

II)

Réseau de neurones à convolution (CNN)



Réseau de neurones à convolution (CNN)



MLP

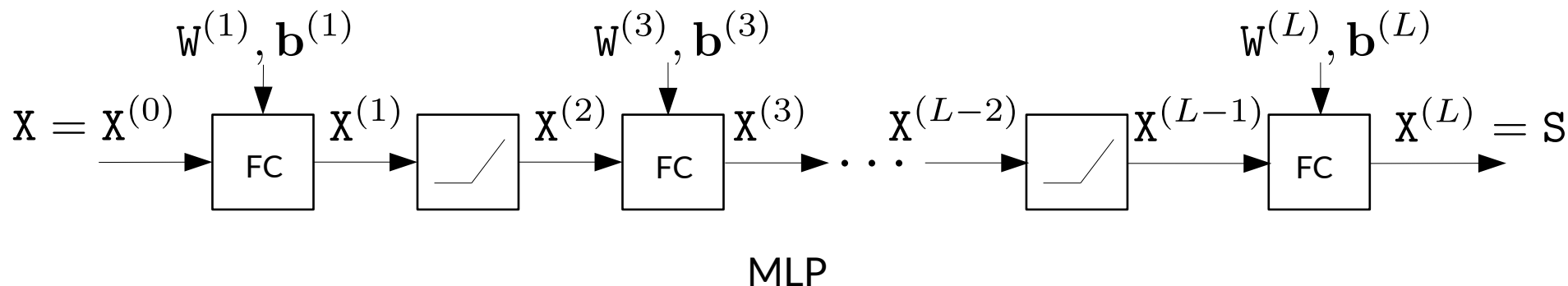
– FC (transformations affines générales)

+ Conv (transformations affines spécifiques)

=

CNN

Réseau de neurones à convolution (CNN)

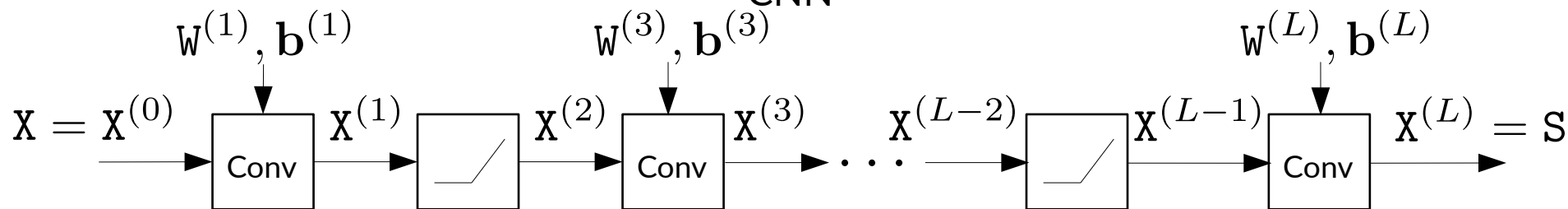


– FC (transformations affines générales)

+ Conv (transformations affines spécifiques)

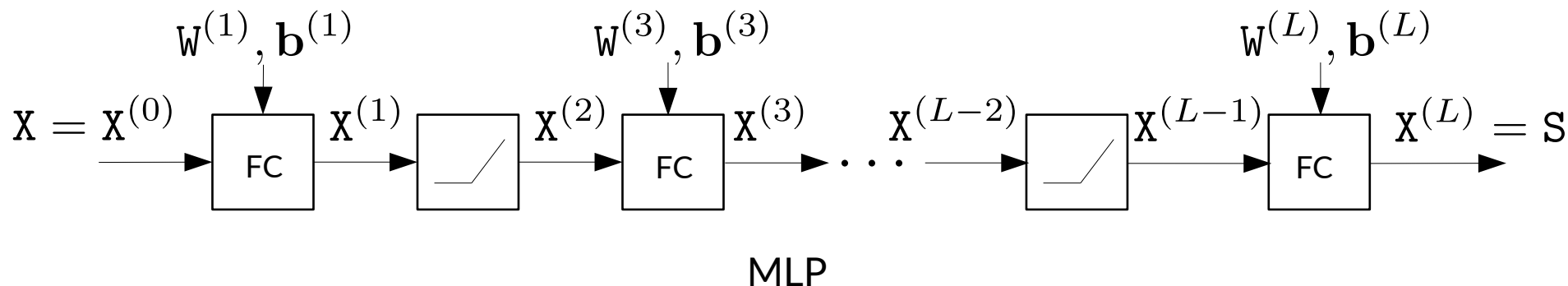
=

CNN



II)

Réseau de neurones à convolution (CNN)

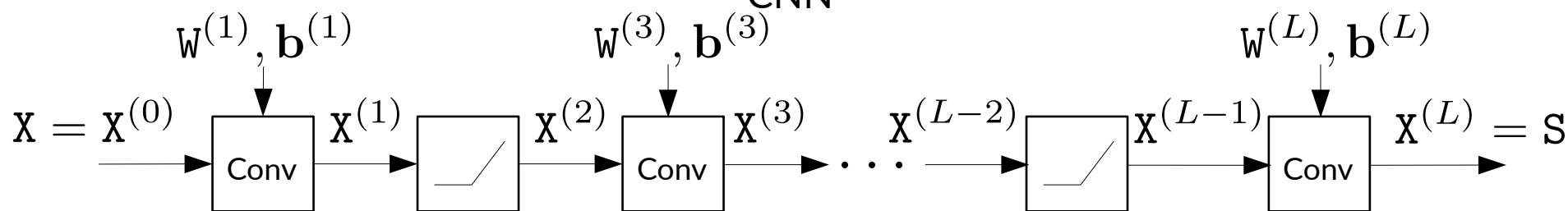


– FC (transformations affines générales)

+ Conv (transformations affines spécifiques)

=

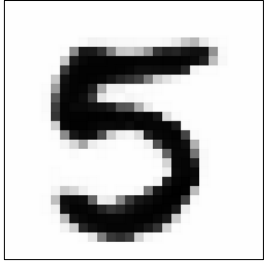
CNN



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

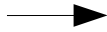
II)

Exemple d'architecture de CNN pour MNIST



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

1x28x28



S



10x1x1

"0"

"1"

"2"

"3"

"4"

"5"

"6"

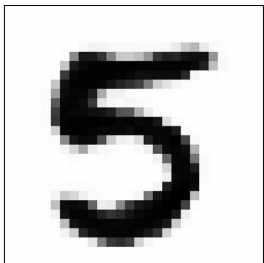
"7"

"8"

"9"

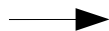
II)

Exemple d'architecture de CNN pour MNIST



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

1x28x28



Objectif : réduire progressivement la résolution de 28x28 à 1x1 et agrandir le nombre de canaux de 1 à 10.

S

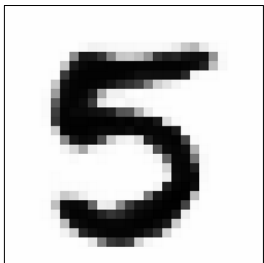


10x1x1

“0”
“1”
“2”
“3”
“4”
“5”
“6”
“7”
“8”
“9”

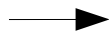
II)

Exemple d'architecture de CNN pour MNIST



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

1x28x28



S



10x1x1

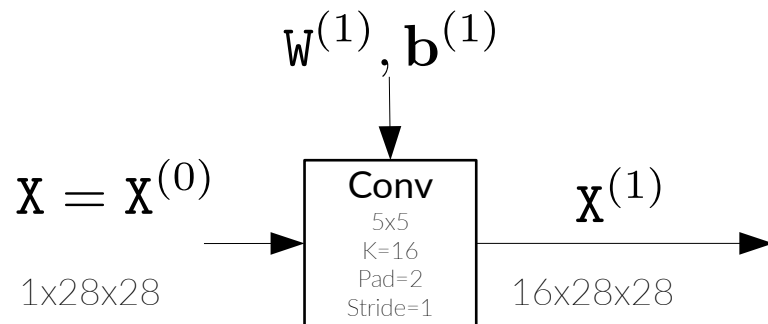
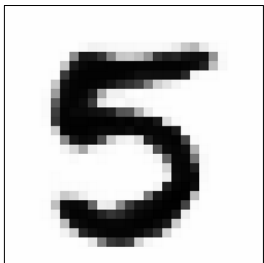
"0"
 "1"
 "2"
 "3"
 "4"
 "5"
 "6"
 "7"
 "8"
 "9"

Objectif : réduire progressivement la résolution de 28x28 à 1x1 et agrandir le nombre de canaux de 1 à 10.

Remarque : en général on augmente beaucoup plus le nombre de canaux au sein du réseau pour ne pas « trop » compresser l'information.

II)

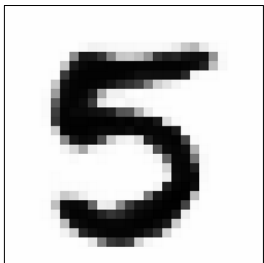
Exemple d'architecture de CNN pour MNIST



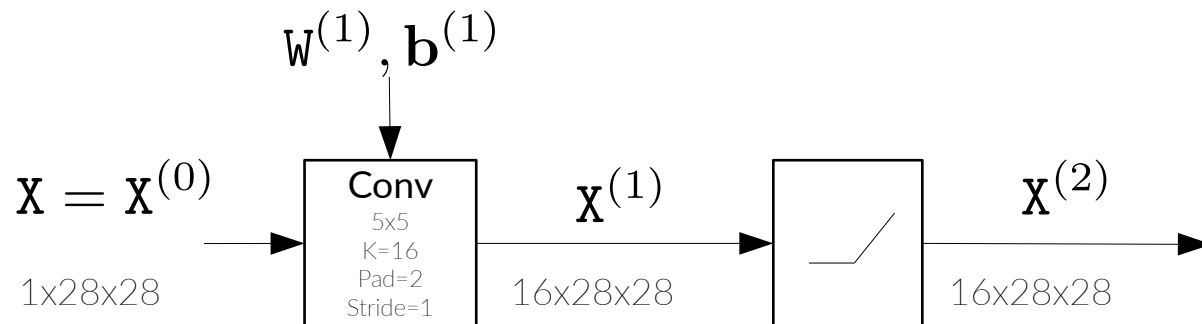
“0”
“1”
“2”
“3”
“4”
“5”
“6”
“7”
“8”
“9”

II)

Exemple d'architecture de CNN pour MNIST

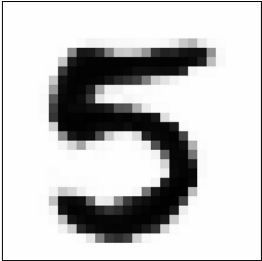


"0"
"1"
"2"
"3"
"4"
"5"
"6"
"7"
"8"
"9"

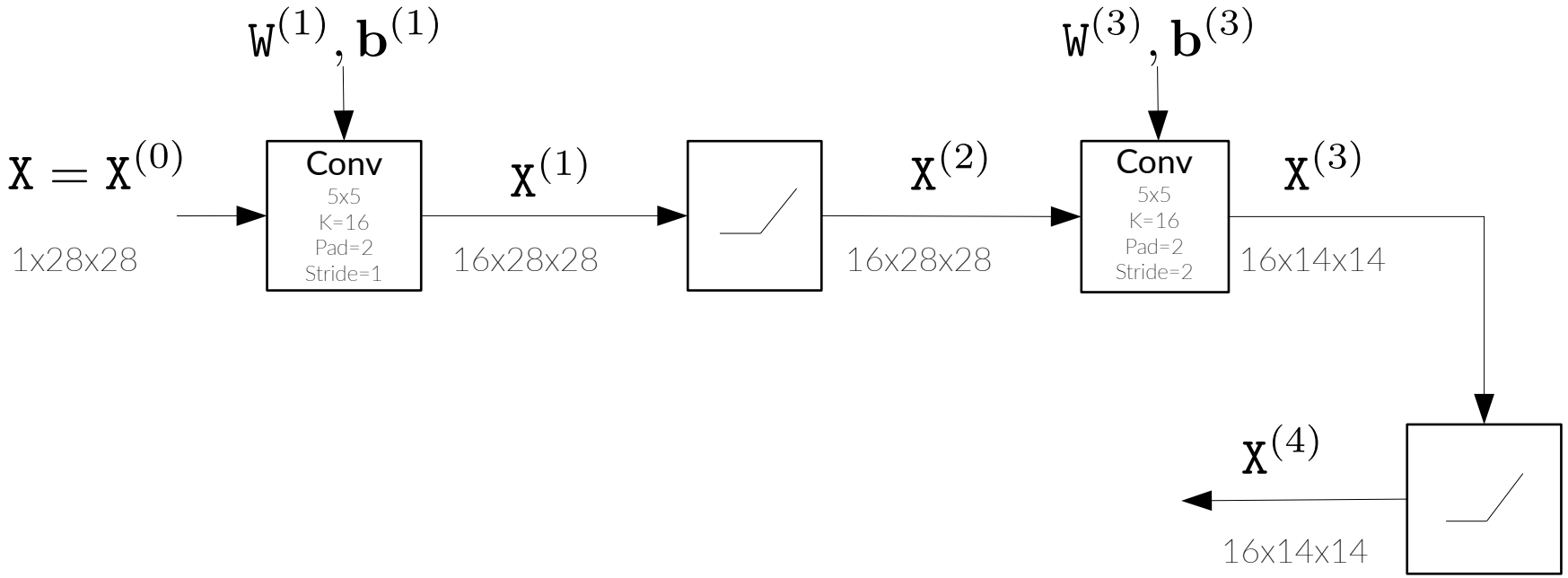


II)

Exemple d'architecture de CNN pour MNIST

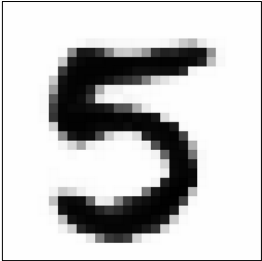


- "0"
- "1"
- "2"
- "3"
- "4"
- "5"
- "6"
- "7"
- "8"
- "9"

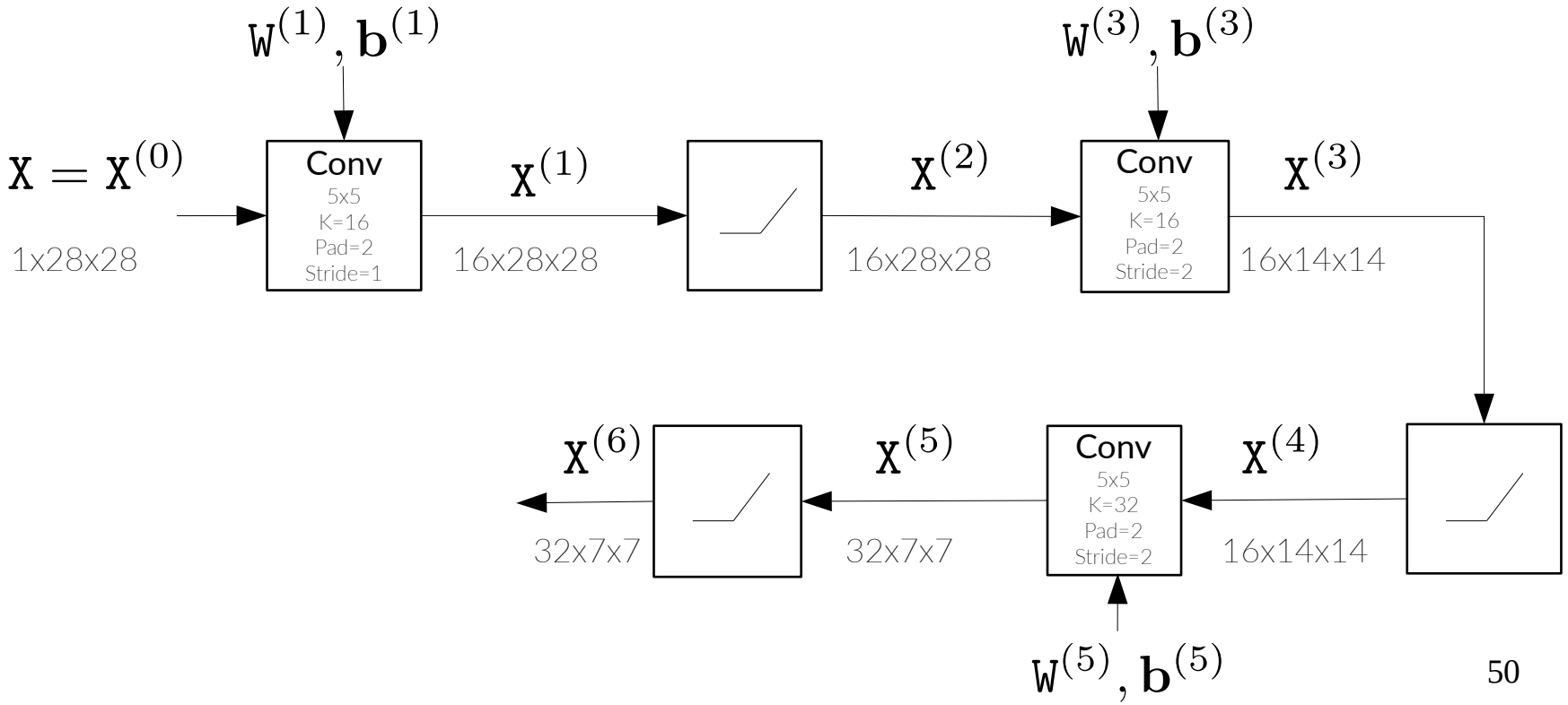


II)

Exemple d'architecture de CNN pour MNIST

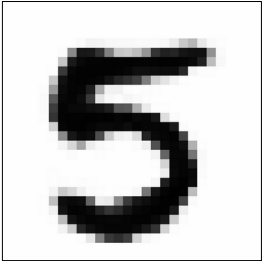


- "0"
- "1"
- "2"
- "3"
- "4"
- "5"
- "6"
- "7"
- "8"
- "9"

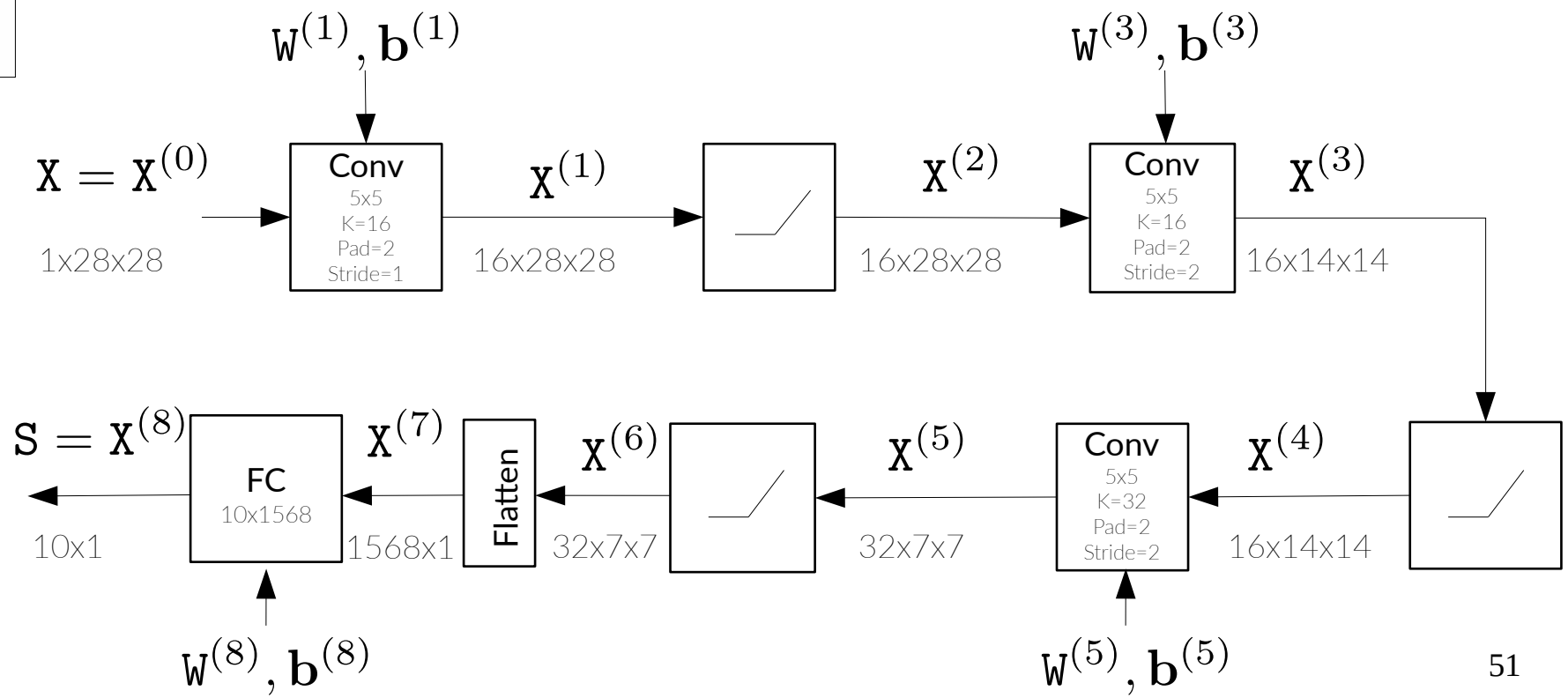


II)

Exemple d'architecture de CNN pour MNIST

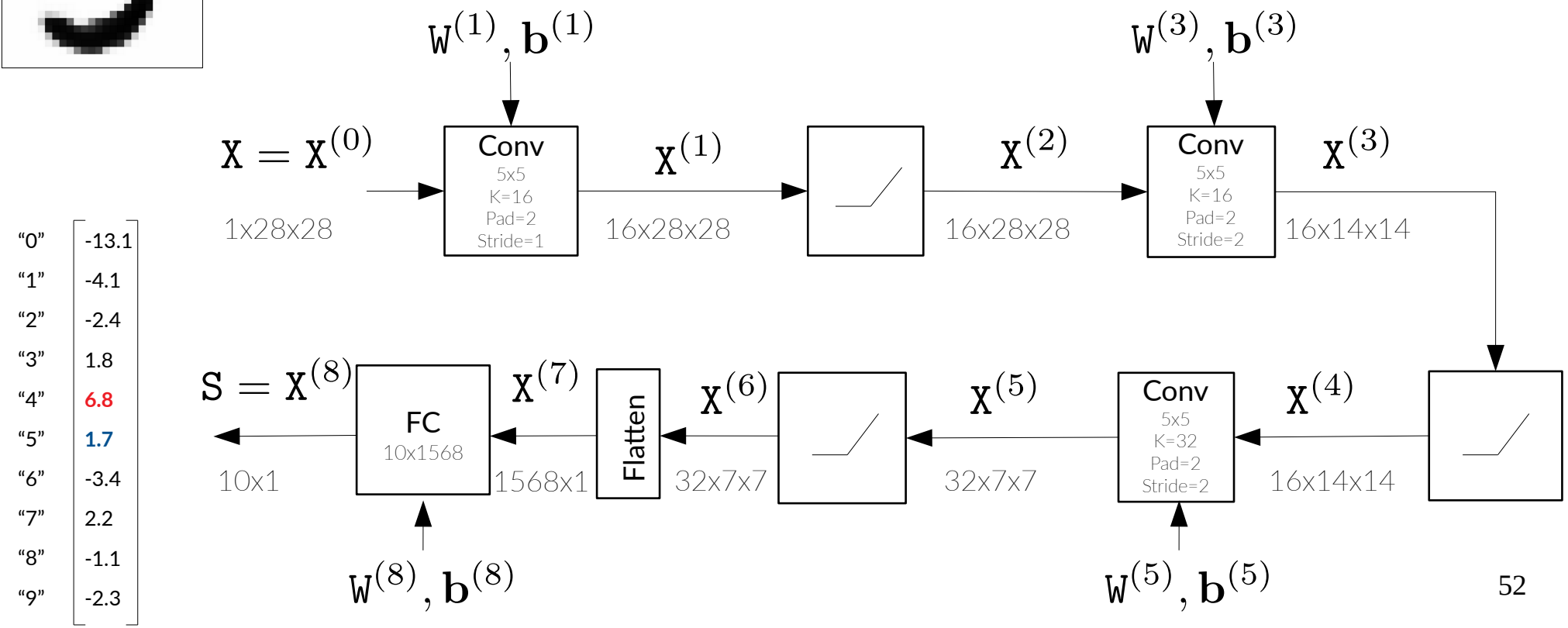
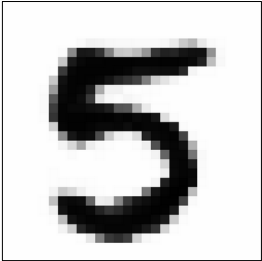


- "0"
- "1"
- "2"
- "3"
- "4"
- "5"
- "6"
- "7"
- "8"
- "9"



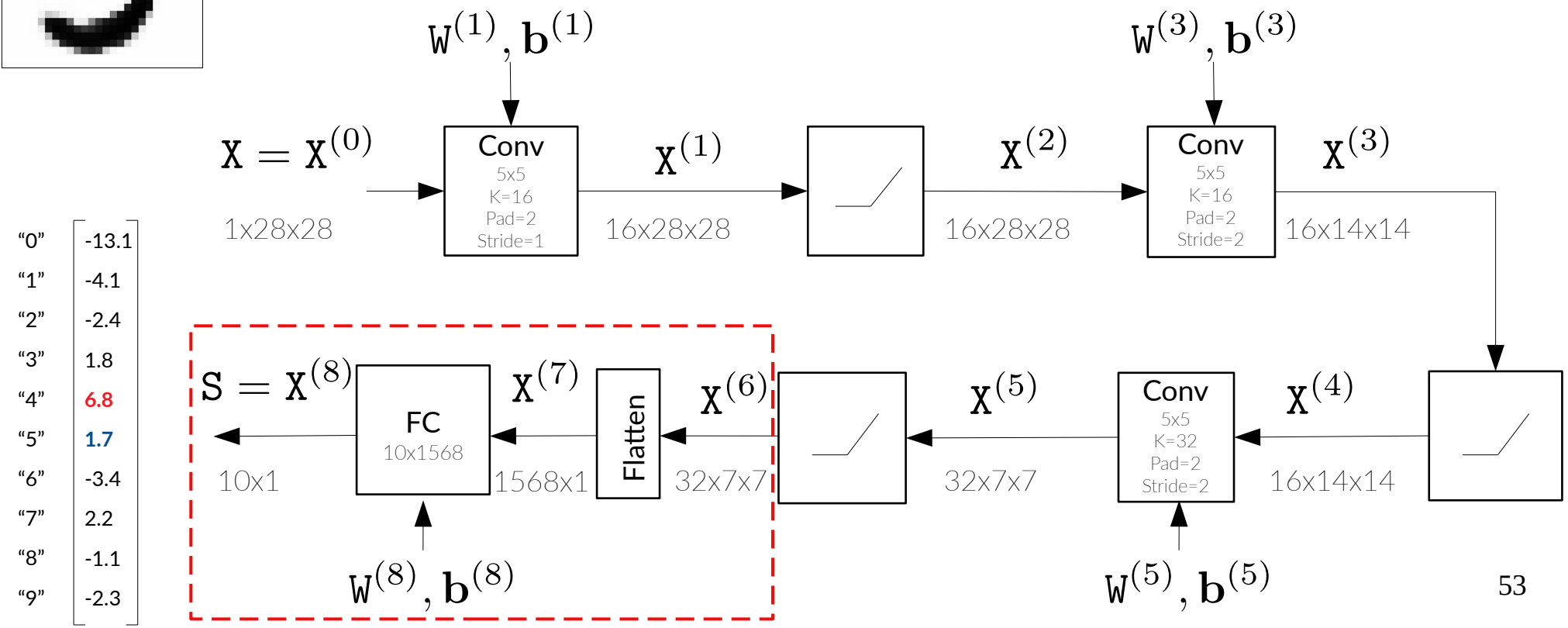
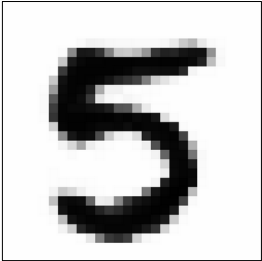
II)

Exemple d'architecture de CNN pour MNIST

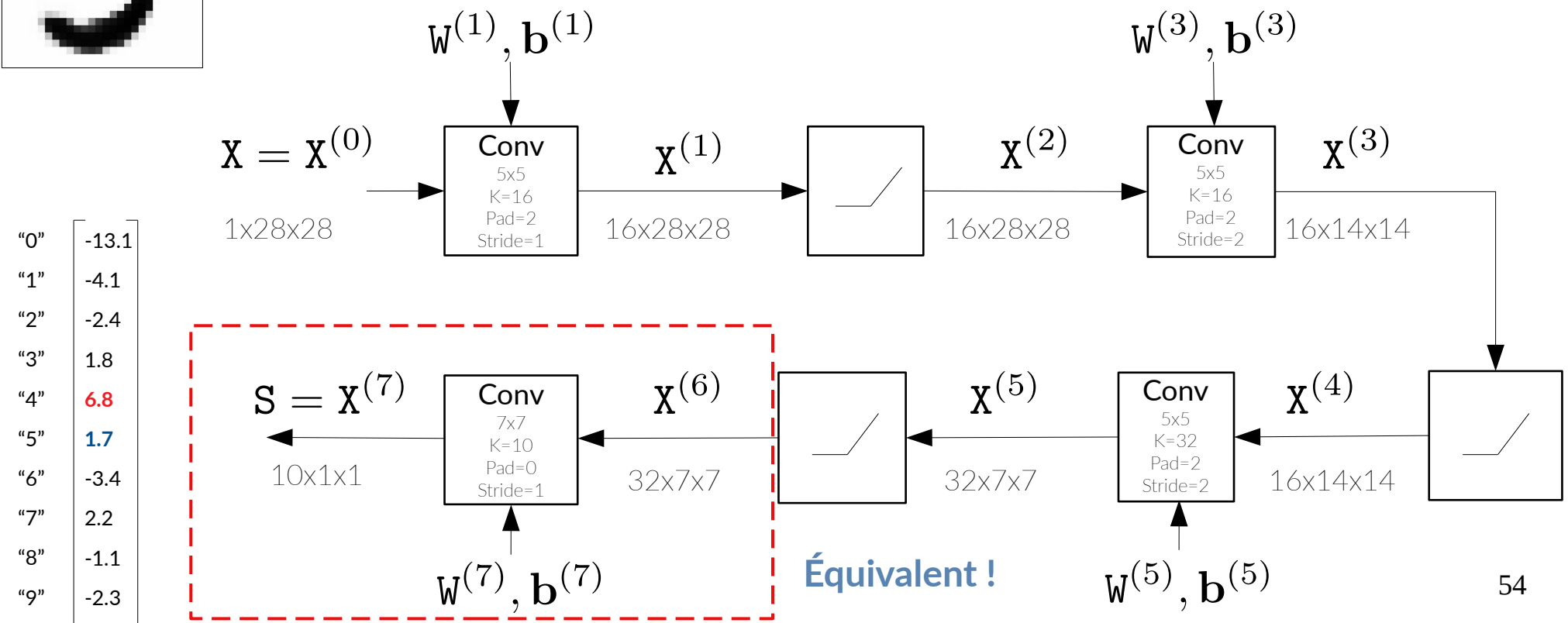
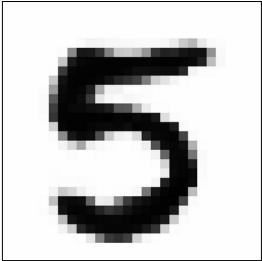


II)

Exemple d'architecture de CNN pour MNIST

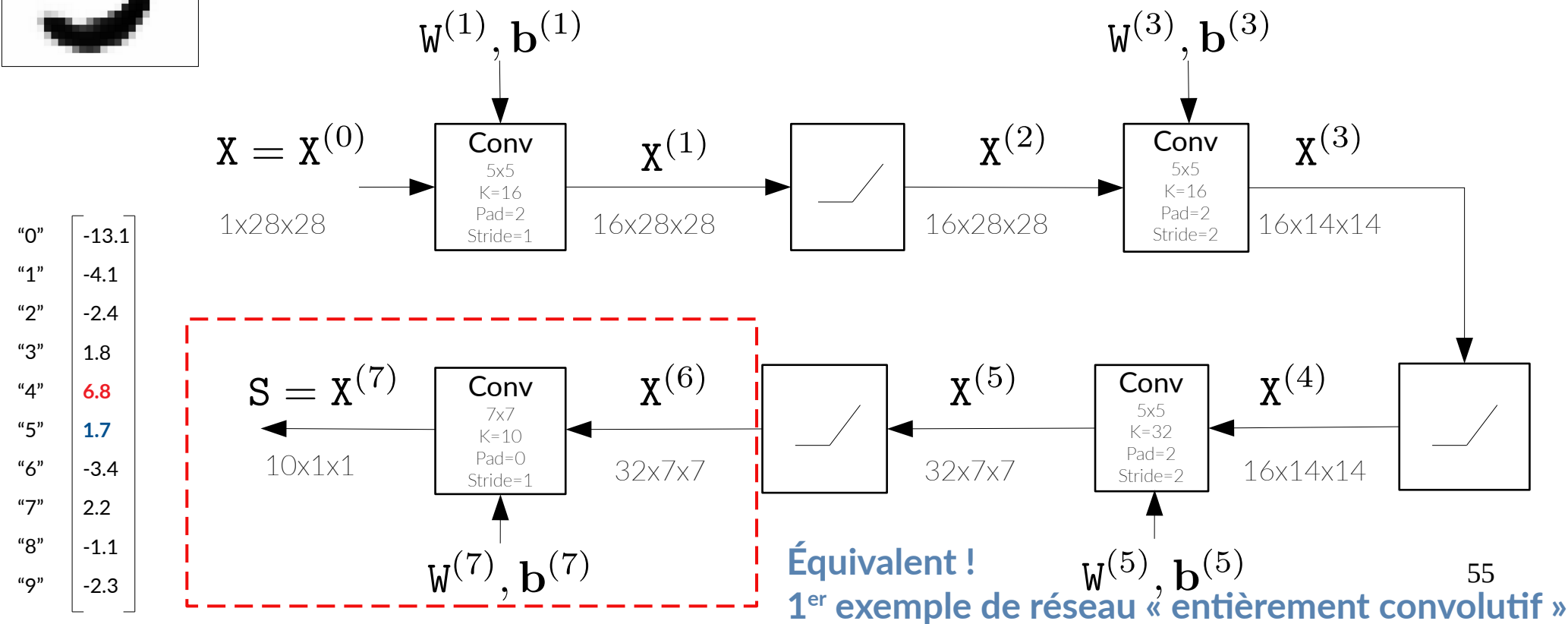
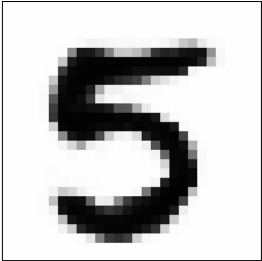


Exemple d'architecture de CNN pour MNIST



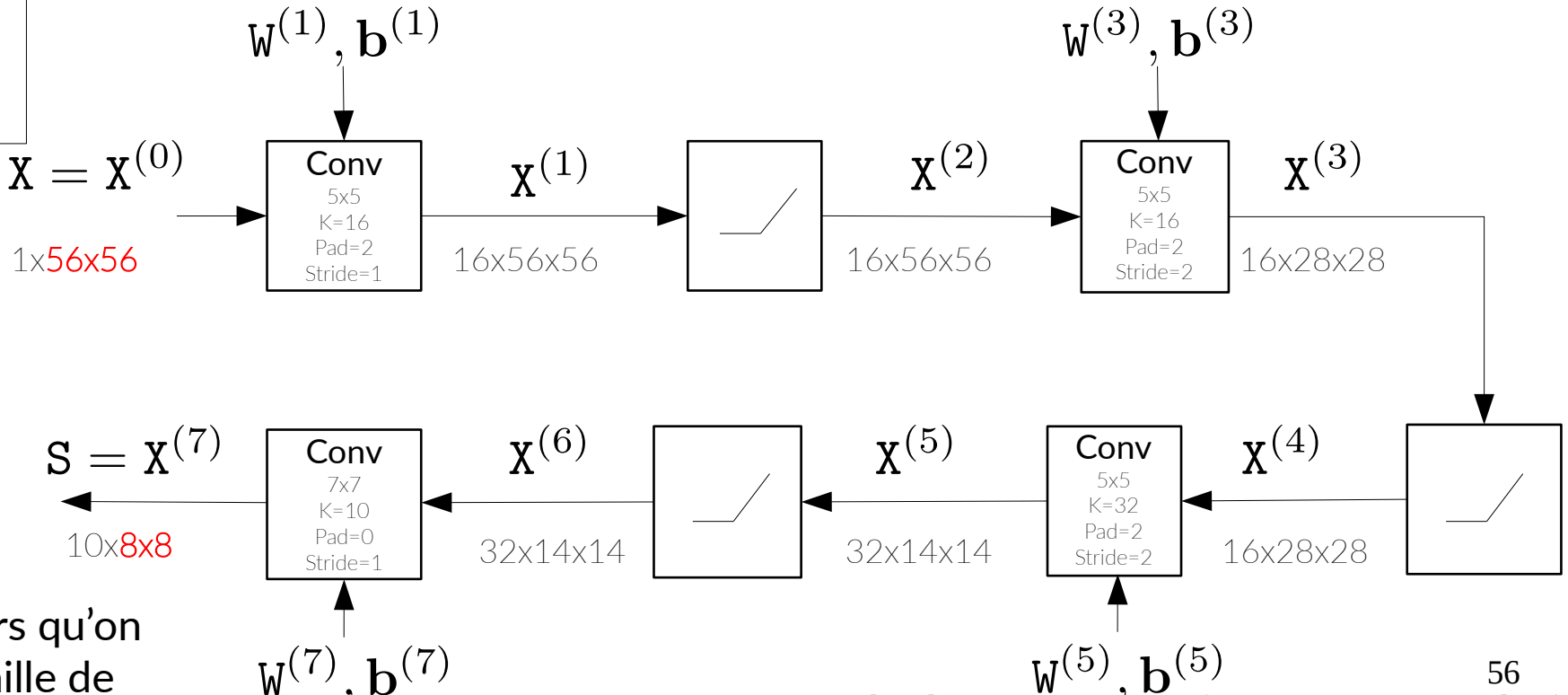
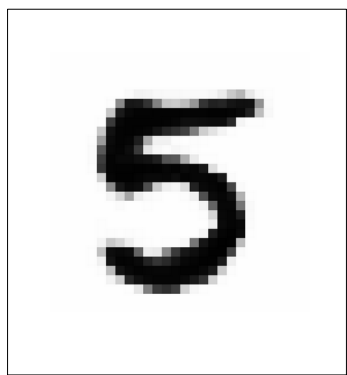
"0"	-13.1
"1"	-4.1
"2"	-2.4
"3"	1.8
"4"	6.8
"5"	1.7
"6"	-3.4
"7"	2.2
"8"	-1.1
"9"	-2.3

Exemple d'architecture de CNN pour MNIST



II)

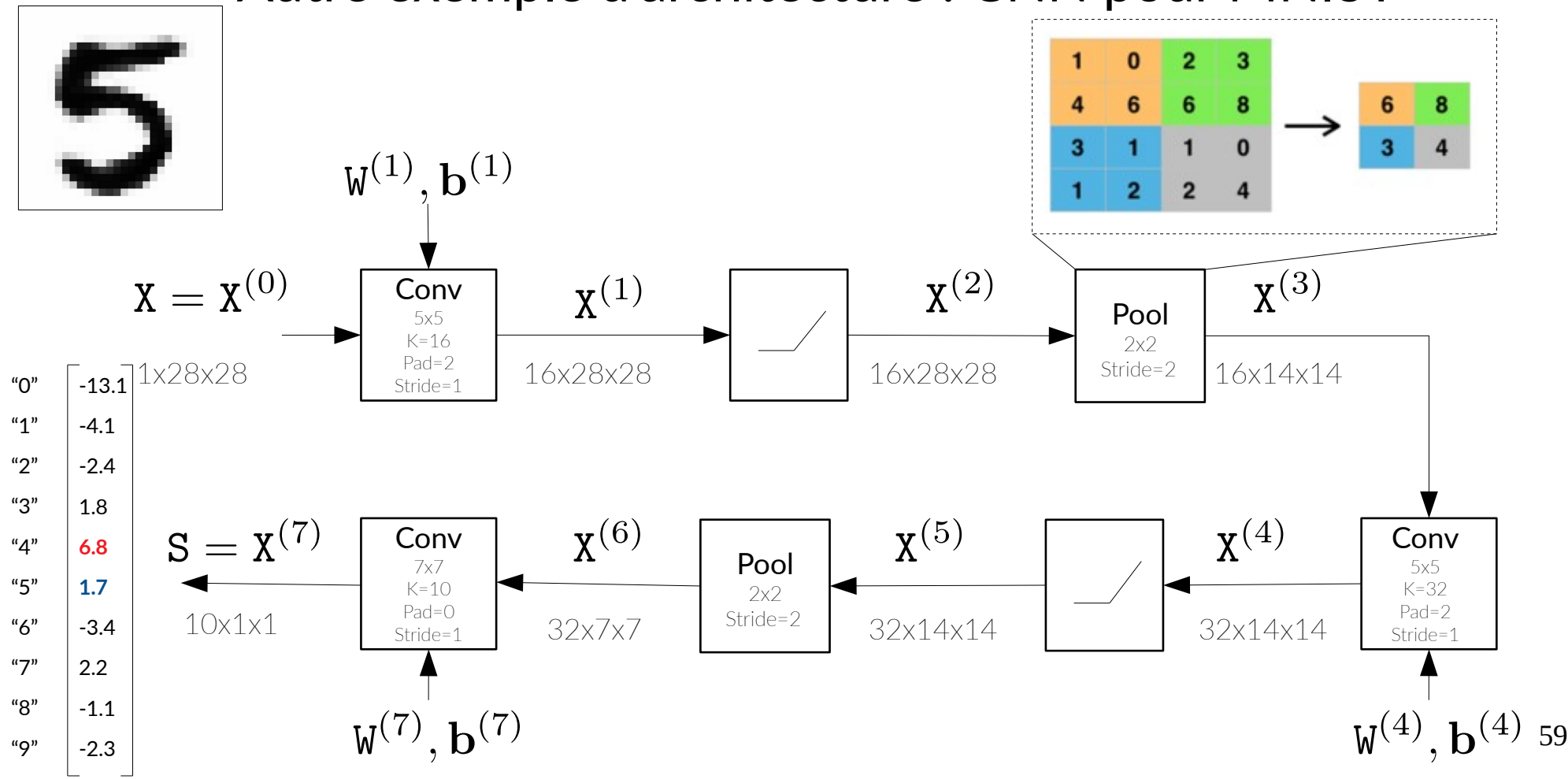
Exemple d'architecture de CNN pour MNIST



S'exécute alors qu'on a changé la taille de l'entrée

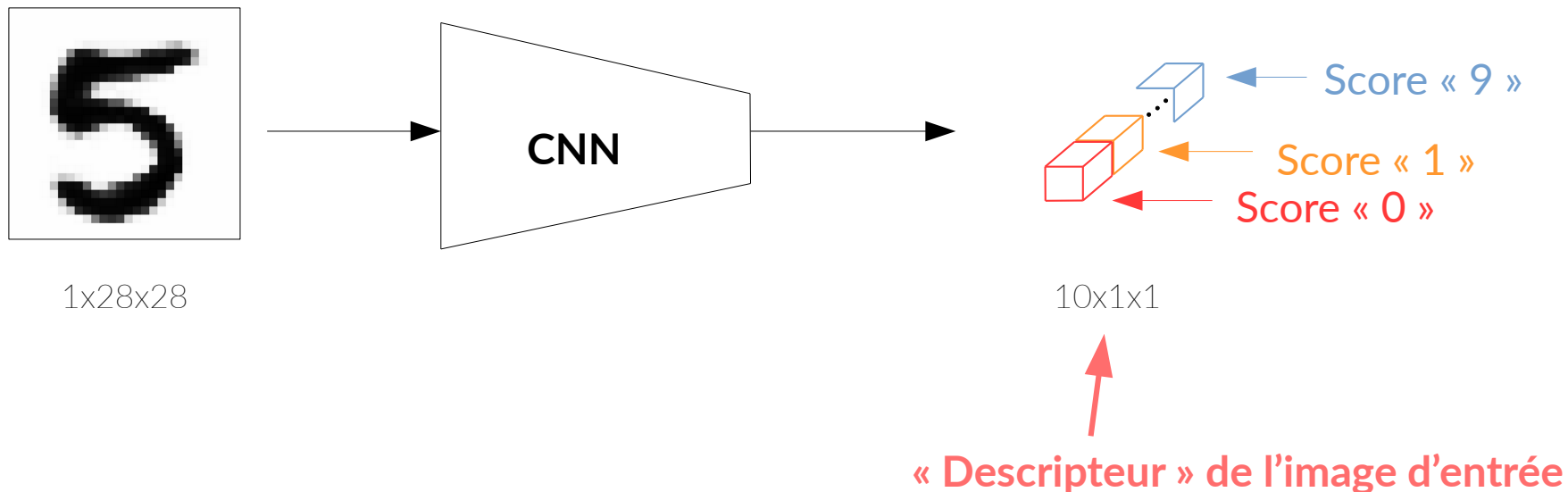
II)

Autre exemple d'architecture : CNN pour MNIST



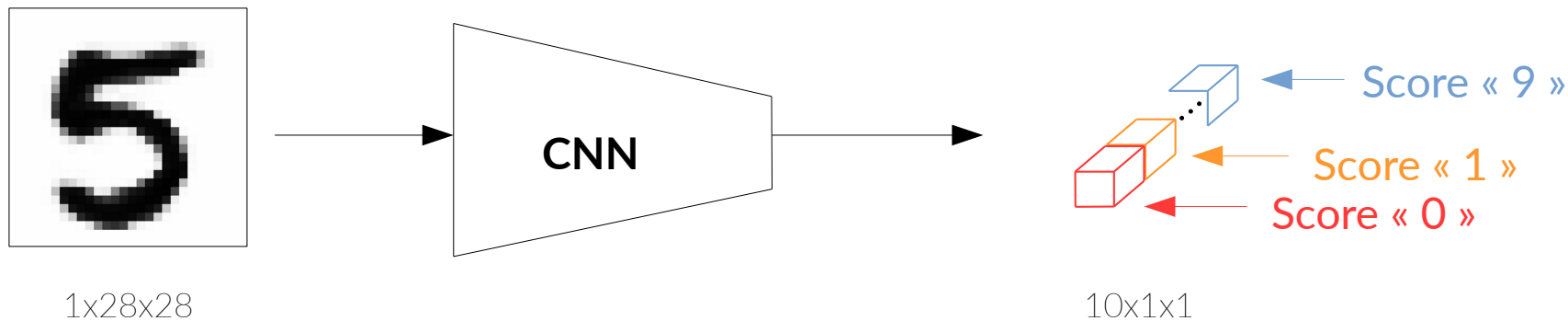
Architectures de CNN : Deux cas extrêmes

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



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 $\text{stride} = 2$

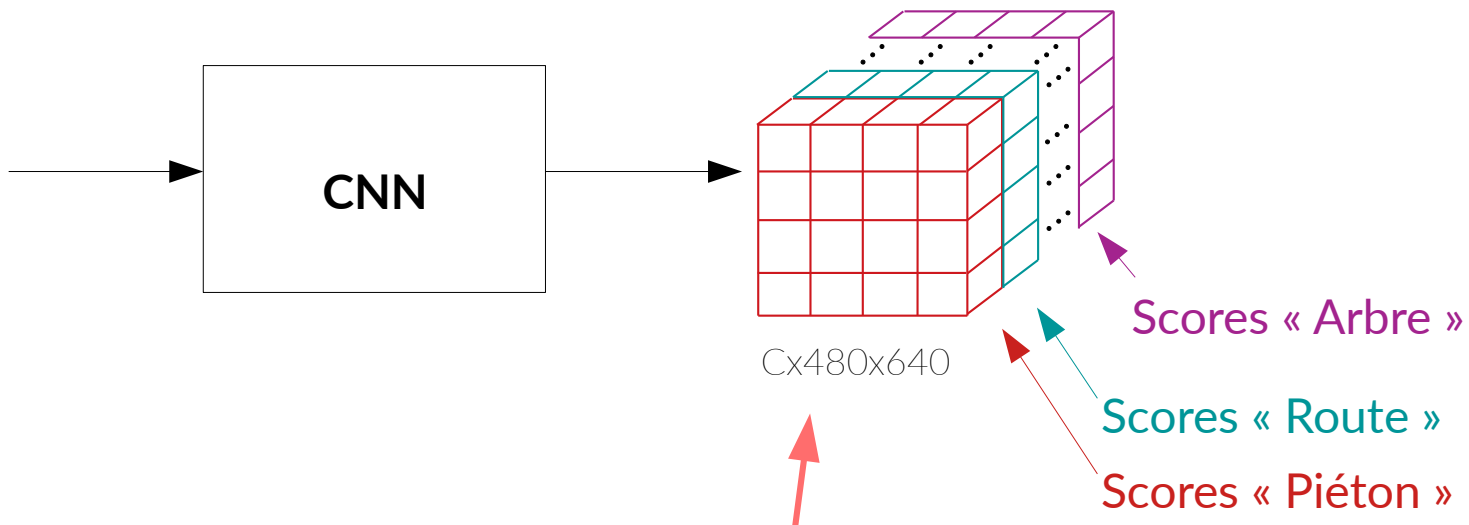
Architectures de CNN : Deux cas extrêmes

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

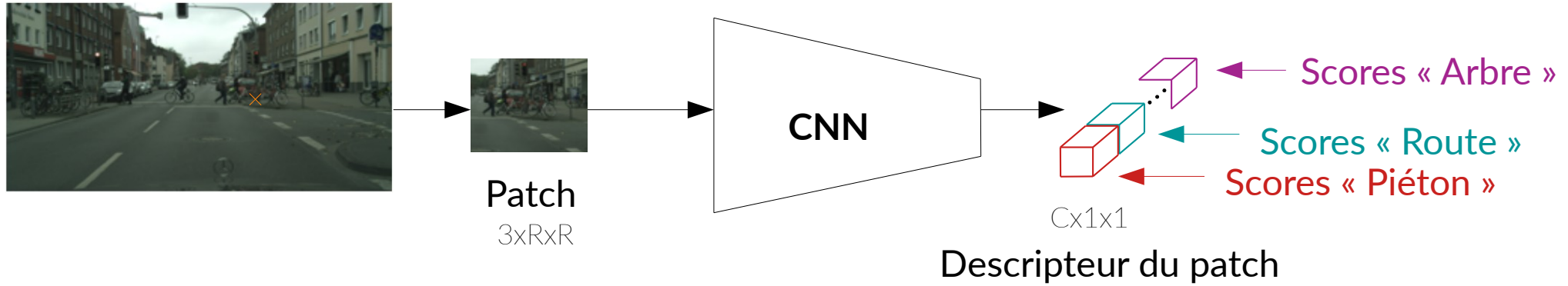


3x480x640



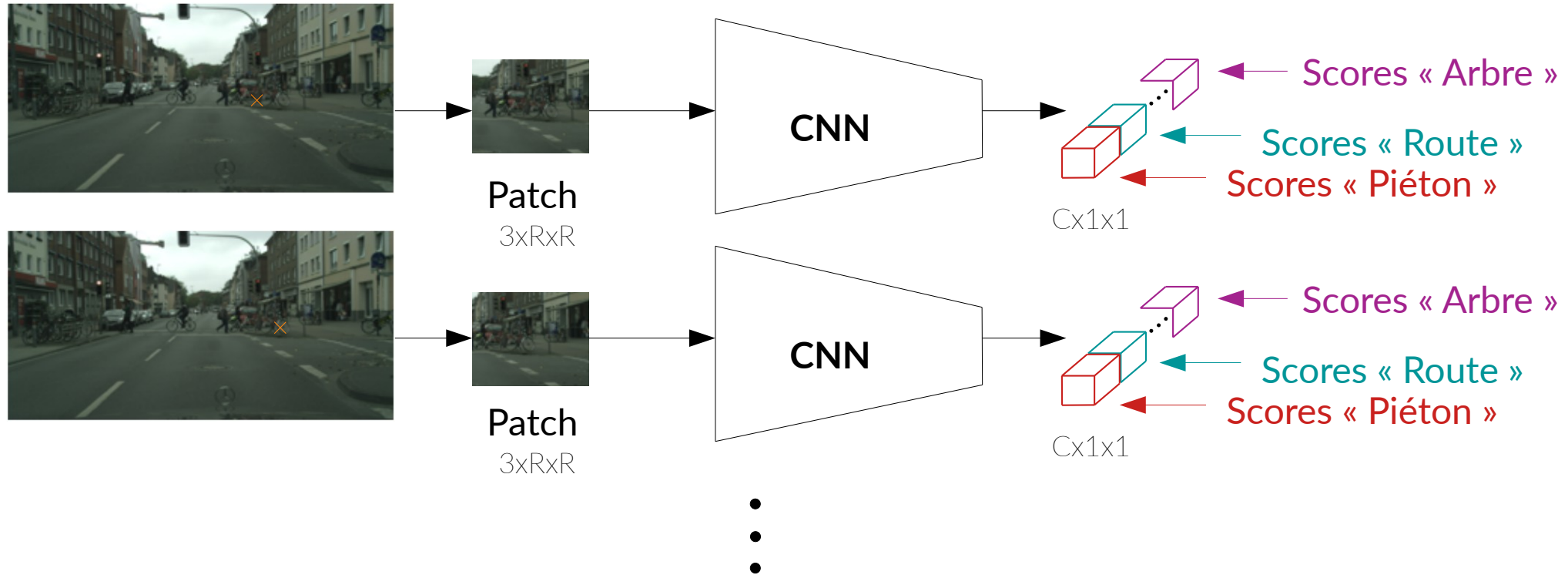
Un « descripteur » par pixel de l'image d'entrée

Comment obtenir un descripteur pour chaque pixel ?



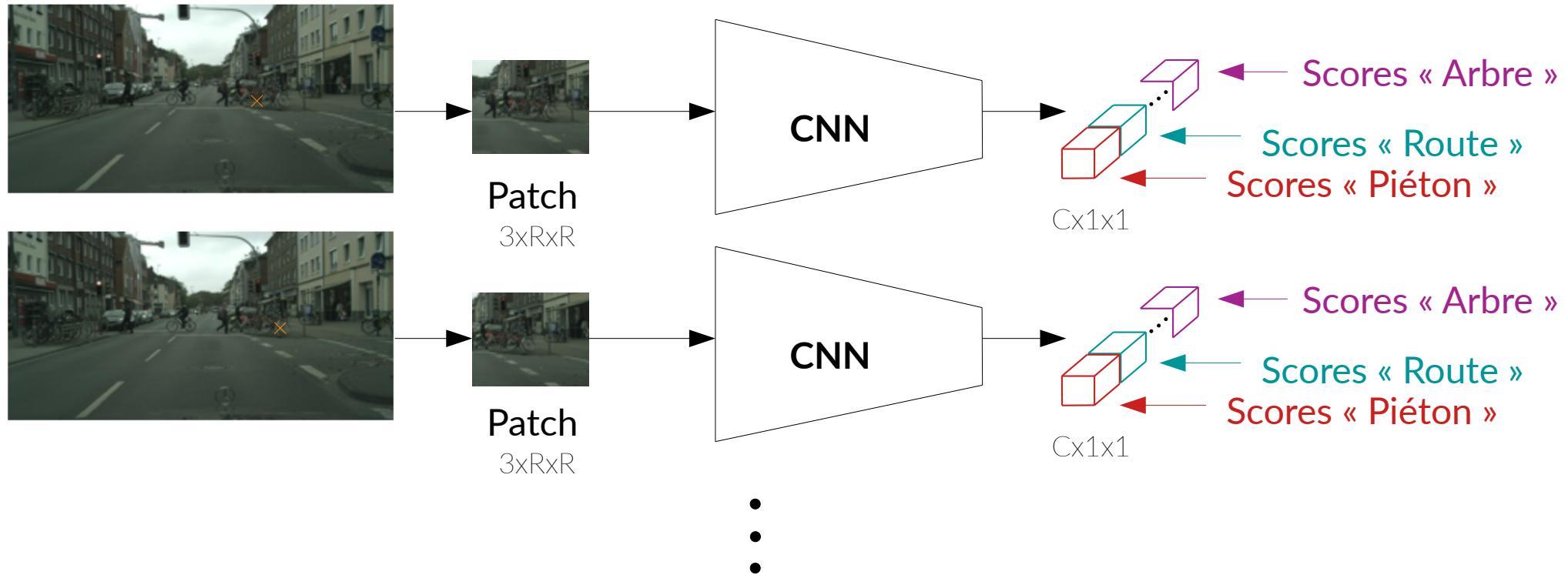
II)

Comment obtenir un descripteur pour chaque pixel ?



II)

Comment obtenir un descripteur pour chaque pixel ?



Les patches voisins ont beaucoup de pixels en commun → calculs redondants

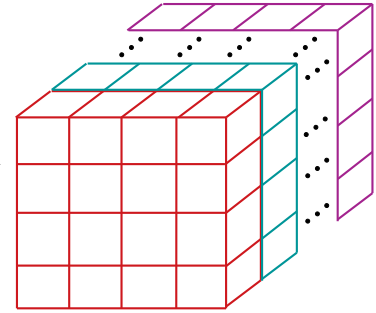
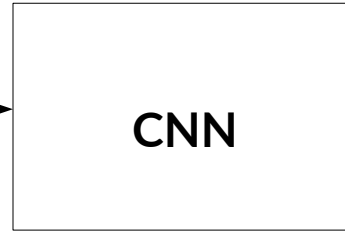
→ Utilisation d'une architecture entièrement convolutive

II)

Architecture entièrement convolutive

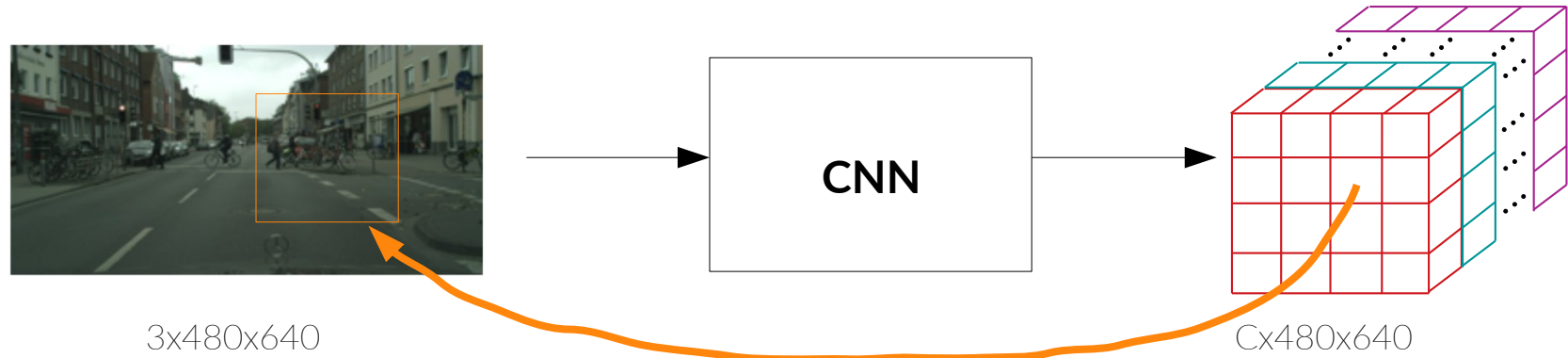


3x480x640



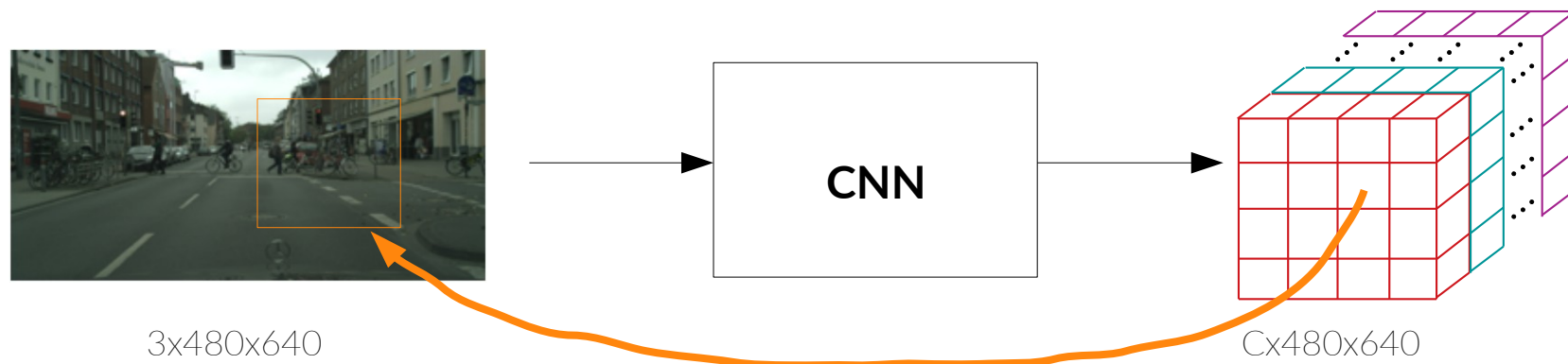
Cx480x640

Architecture entièrement convolutive



Descripteur $C \times 1 \times 1$ d'un patch de taille $R \times R$,
où $R \times R$ s'appelle le Champ Récepteur du CNN (« Receptive Field »)

Architecture entièrement convolutive

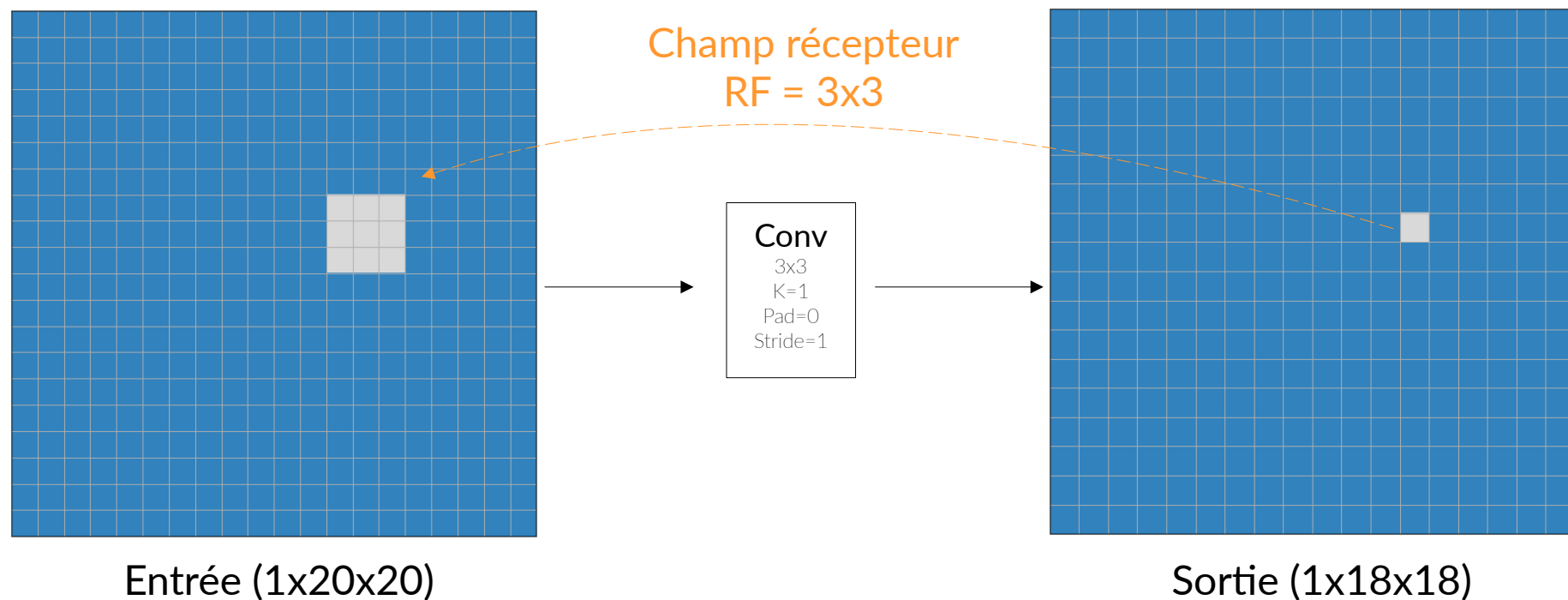


Descripteur $C \times 1 \times 1$ d'un patch de taille $R \times R$,
où $R \times R$ s'appelle le Champ Récepteur du CNN (« Receptive Field »)

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

II)

Champ récepteur d'une couche de convolution



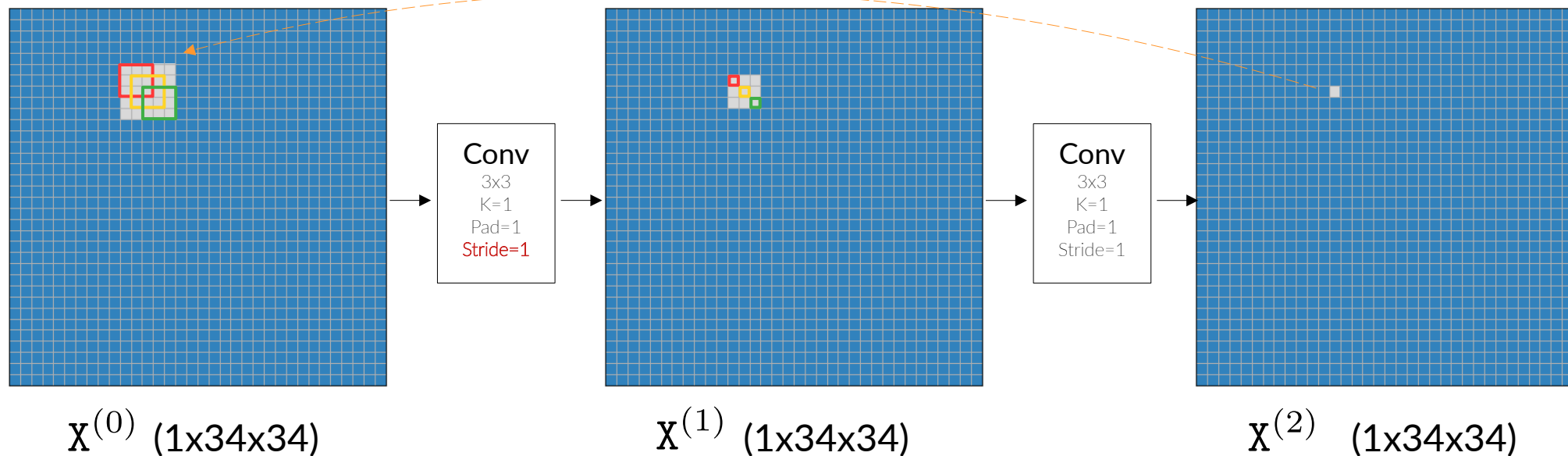
Pour une couche → champ récepteur = taille du filtre

II)

Champ récepteur de deux couches de convolution

Stride = 1

Champ récepteur
RF = 5x5

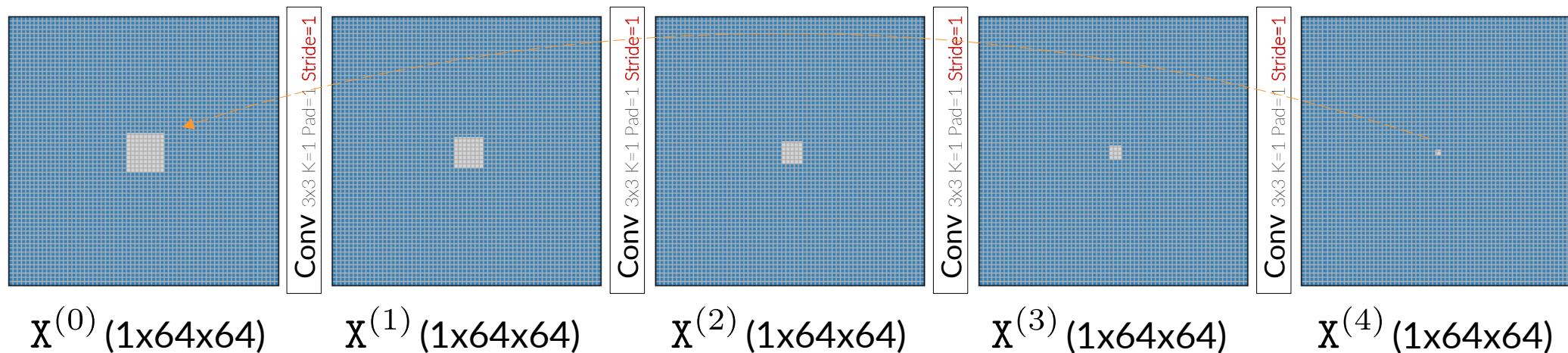


Remarque : les ReLU n'affectent pas le RF donc on ne les représente pas ici.

II)

Champ récepteur de quatre couches de convolution Stride = 1

Champ récepteur
RF = 9x9

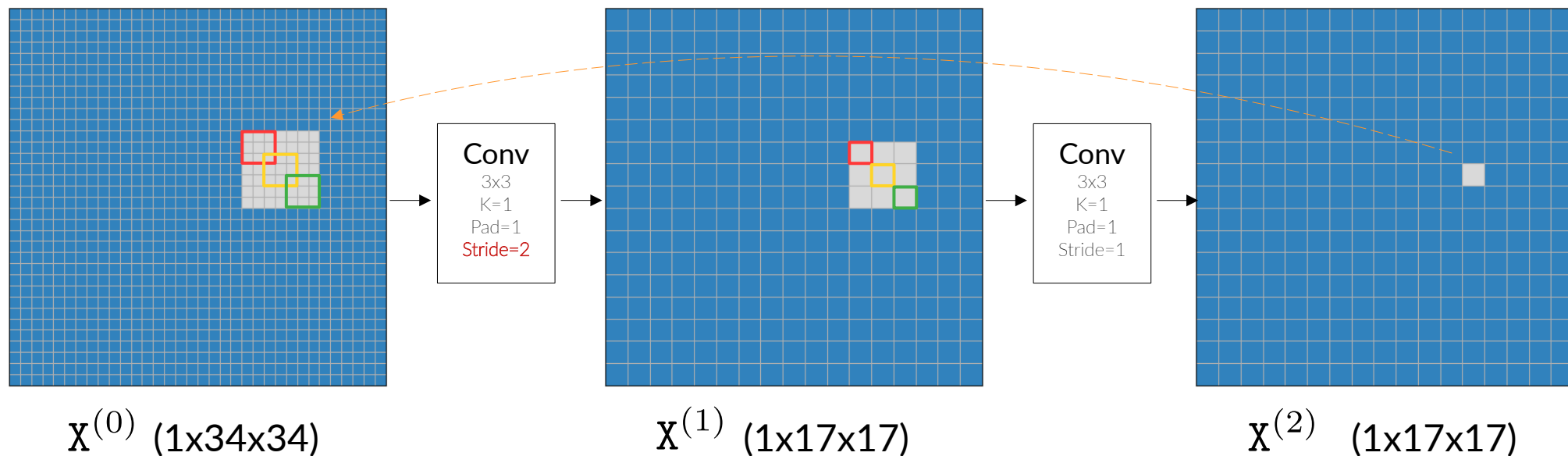


→ Croissance du RF très lente ...

II)

Champ récepteur de deux couches de convolution Stride = 2

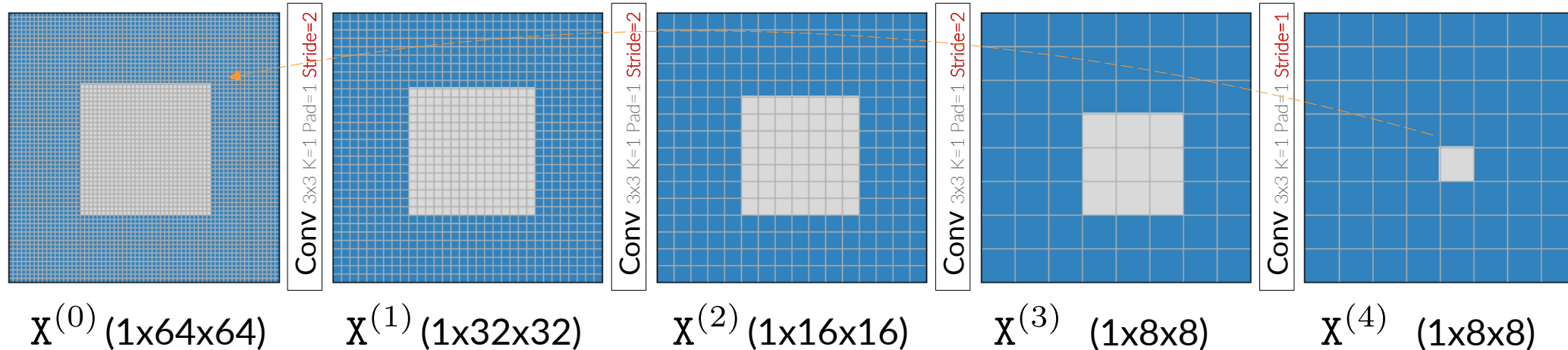
Champ récepteur
RF = 7x7



II)

Champ récepteur de quatre couches de convolution Stride = 2

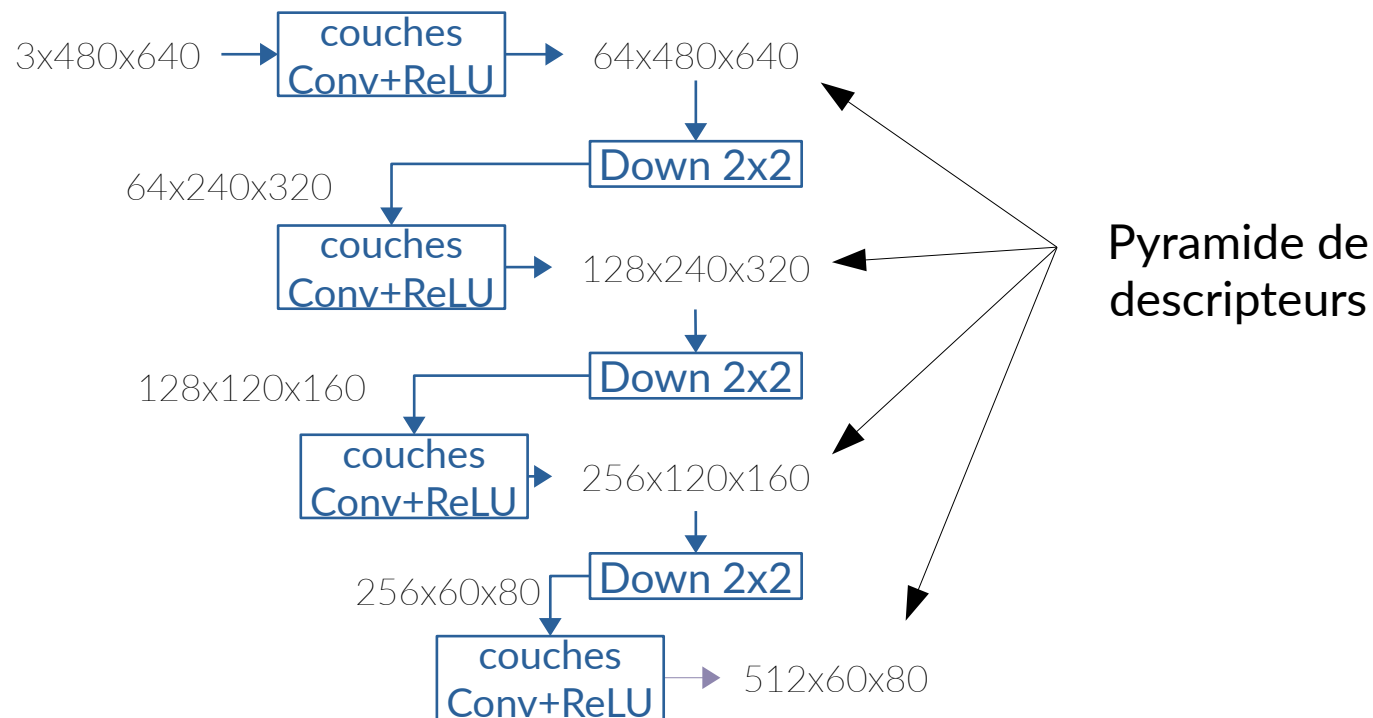
Champ récepteur
RF = 31x31



→ Croissance rapide du RF ... mais baisse de la résolution
→ Architecture U-Net

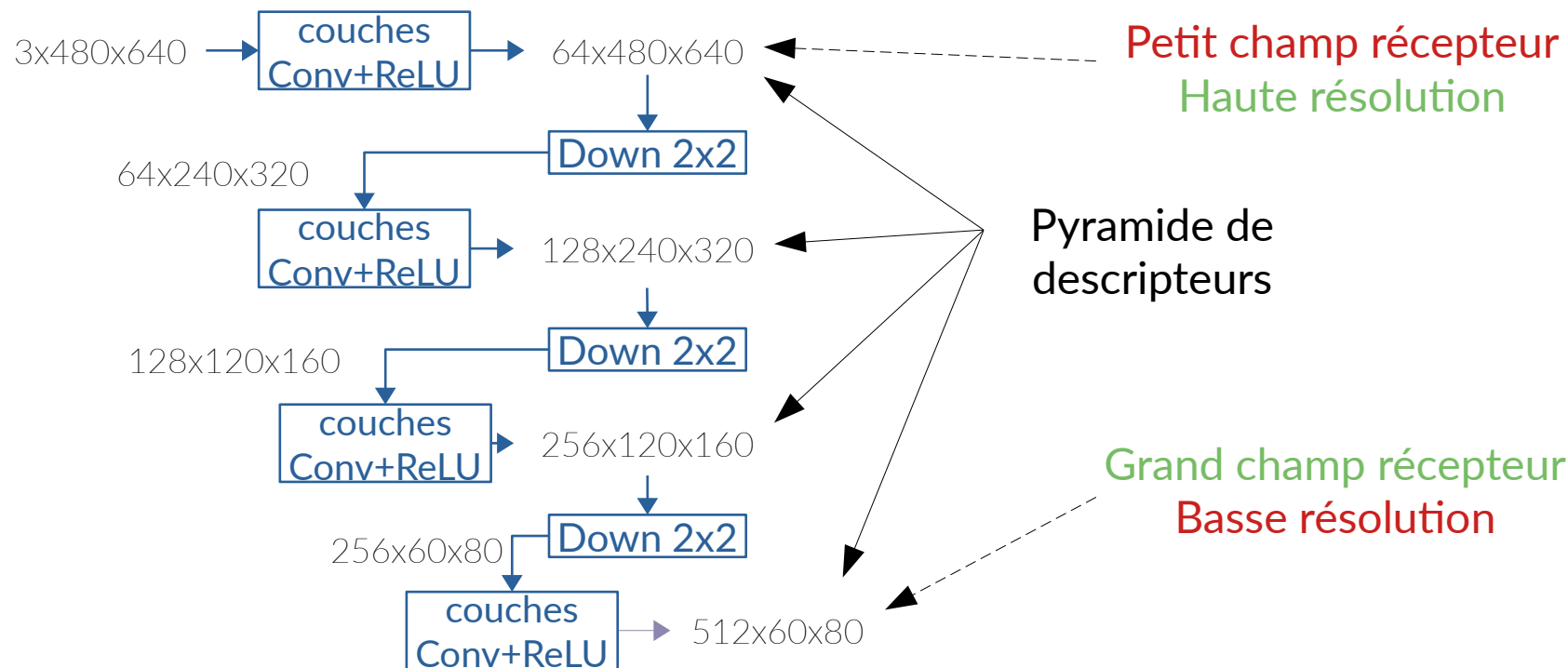
II)

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



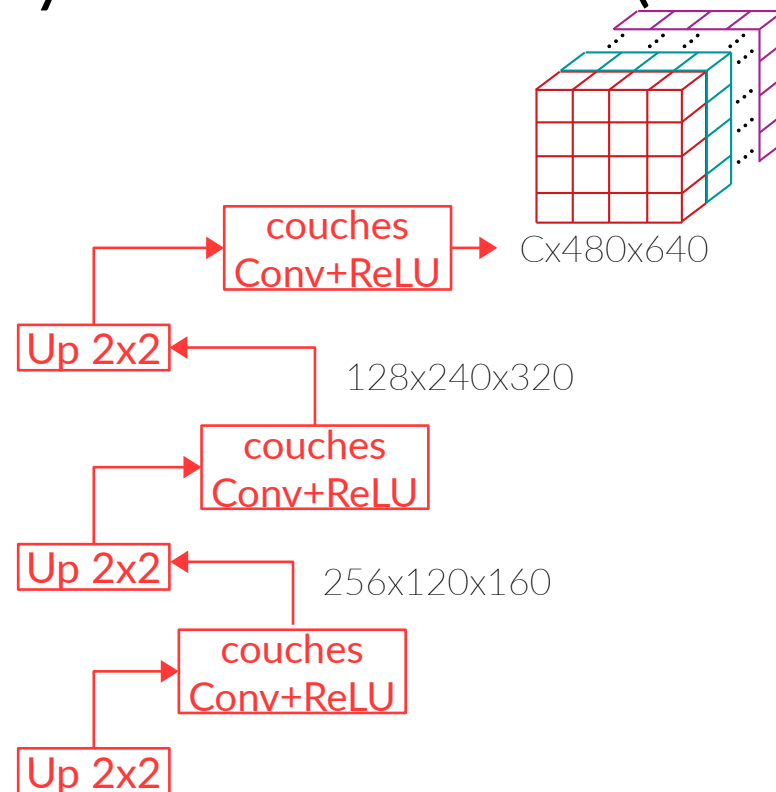
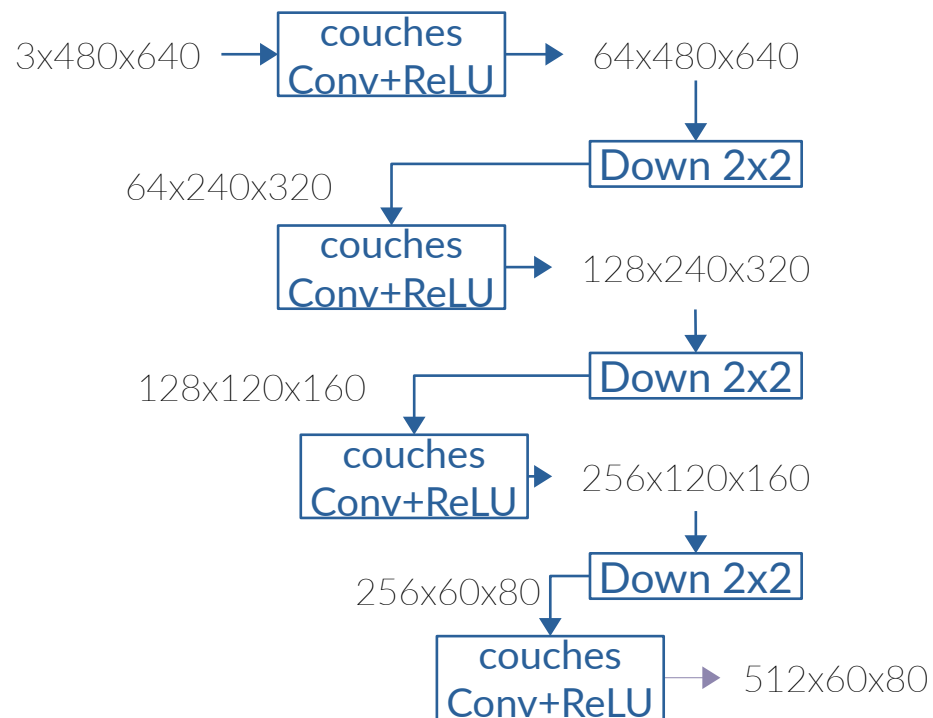
II)

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



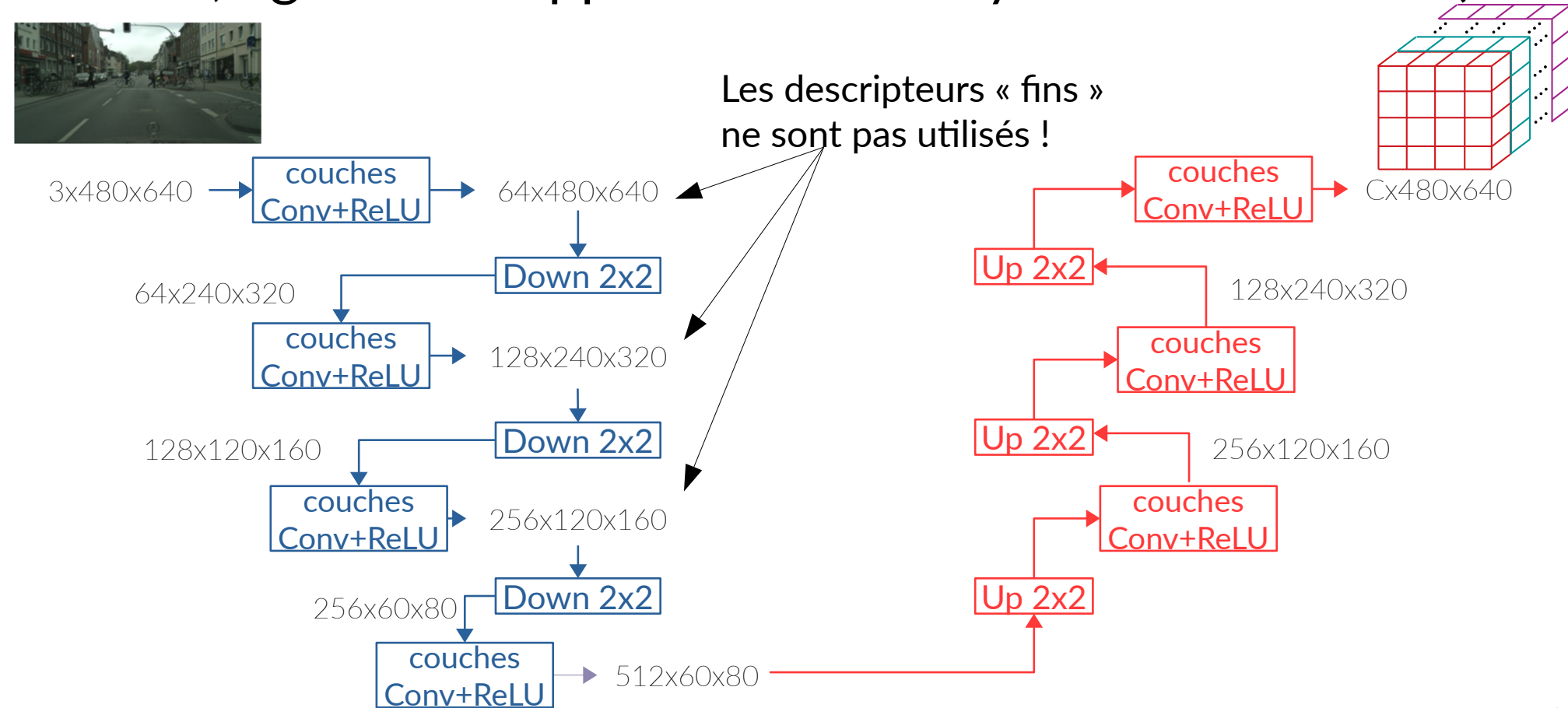
II)

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



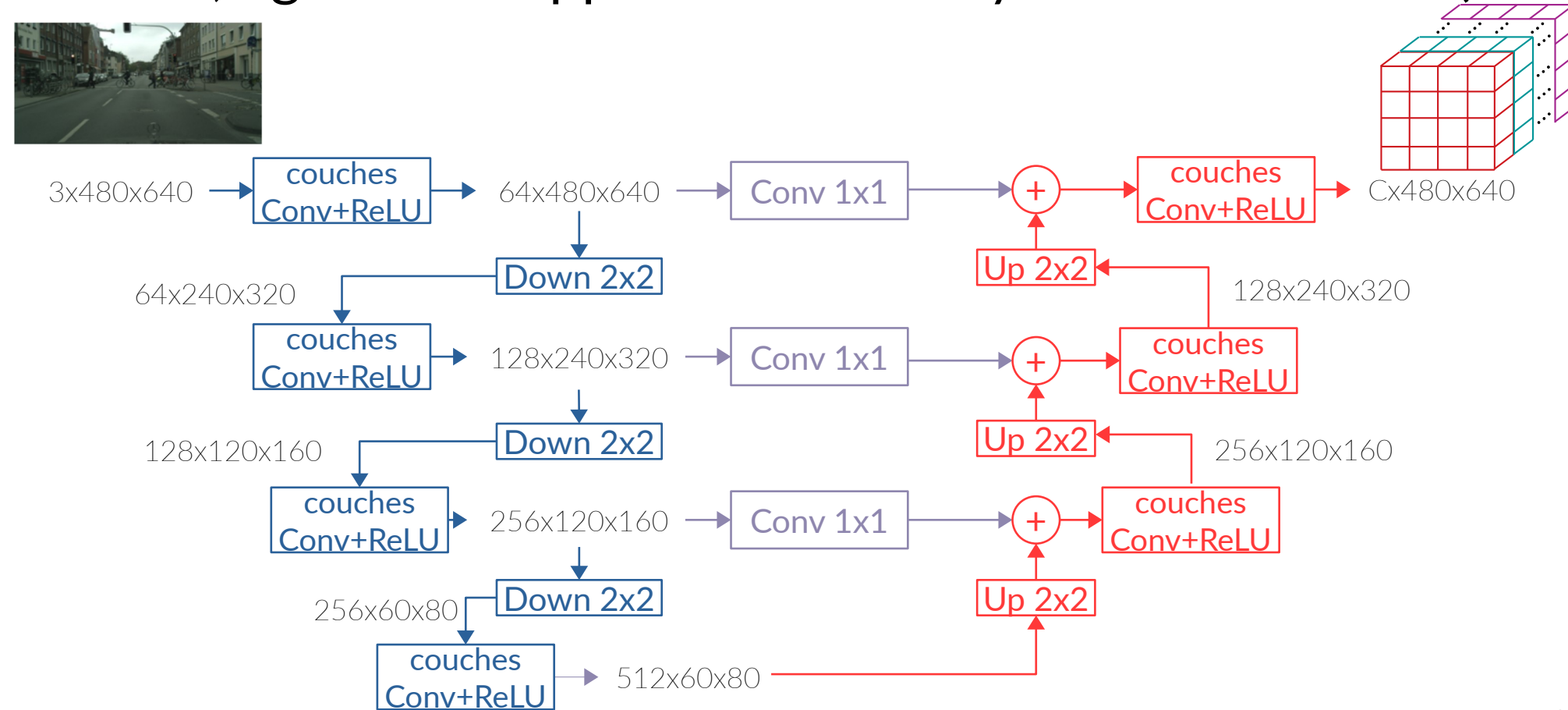
II)

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



II)

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



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

