Convolution

Vidéo ■ partie 12.1. Convolution d'une matrice
Vidéo ■ partie 12.2. Neurone et couche de convolution
Vidéo ■ partie 12.3. Les différentes couches d'un réseau
Vidéo ■ partie 12.4. Traitement des images

La convolution est une opération mathématique simple sur un tableau de nombres, une matrice ou encore une image afin d'y apporter une transformation ou d'en tirer des caractéristiques principales.

1. Matrice

Nous allons voir ou revoir quelques notions (le minimum vital) sur les matrices.

1.1. Vocabulaire

- Une ***matrice*** est un tableau de n lignes et p colonnes, contenant des nombres réels.

$$\overset{n \text{ lignes}}{\updownarrow} \begin{pmatrix} a_{11} & a_{12} & \dots & a_{1p} \\ a_{21} & a_{22} & \dots & a_{2p} \\ \vdots & \vdots & \ddots & \vdots \\ a_{n1} & a_{n2} & \dots & a_{np} \end{pmatrix}$$
$$\overset{p \text{ colonnes}}{\longleftrightarrow}$$

- Lorsque il y a autant de lignes que de colonnes, on parle de ***matrice carrée***. Voici une matrice 3×3 :

$$\begin{pmatrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{pmatrix}$$

- ***Addition.*** On peut additionner deux matrices de même taille. L'addition se fait terme à terme. Exemple :

$$\begin{pmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{pmatrix} + \begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{pmatrix} = \begin{pmatrix} 1 & 3 & 3 \\ 5 & 1 & 7 \\ 7 & 9 & 9 \end{pmatrix}$$

- ***Multiplication par un scalaire.*** On peut multiplier une matrice par un nombre réel. Exemple :

$$3 \cdot \begin{pmatrix} 1 & 2 & 3 & 4 \\ 8 & 7 & 5 & 6 \end{pmatrix} = \begin{pmatrix} 3 & 6 & 9 & 12 \\ 24 & 21 & 15 & 18 \end{pmatrix}$$

- ***Multiplication.*** On peut multiplier deux matrices. Nous ne donnons pas ici la définition, mais la multiplication *n'est pas* effectuée terme à terme. Voici, sans entrer dans les détails, un exemple pour des

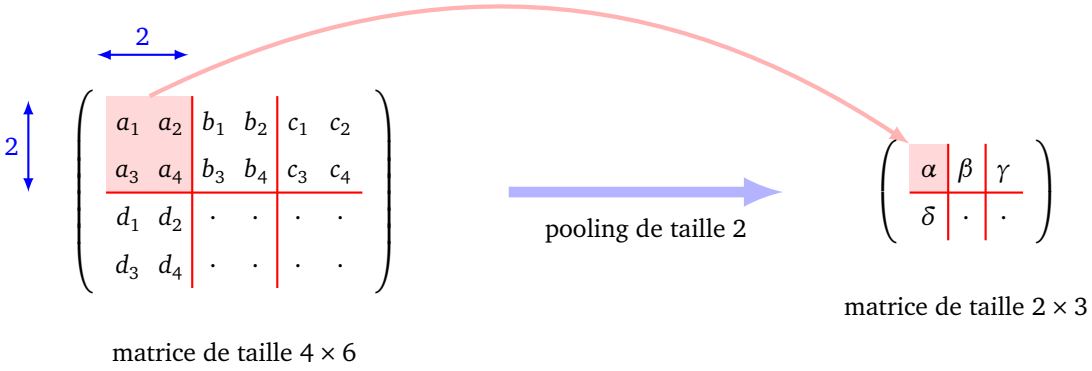
matrices carrées :

$$\begin{pmatrix} -2 & -1 & 0 \\ -1 & 1 & 1 \\ 0 & 1 & 2 \end{pmatrix} \times \begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{pmatrix} = \begin{pmatrix} -6 & -9 & -12 \\ 10 & 11 & 12 \\ 18 & 21 & 24 \end{pmatrix}$$

Nous définirons un peu plus loin la convolution qui est un nouveau type de multiplication de matrices.

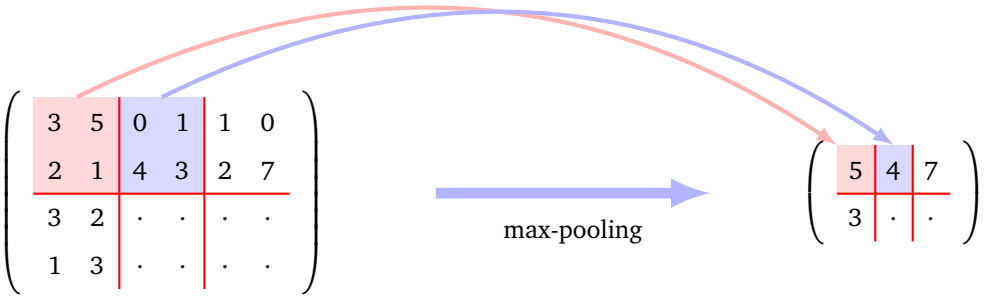
1.2. Pooling

Le *pooling* (regroupement de termes) consiste à transformer une matrice en une matrice plus petite tout en essayant d'en garder les caractéristiques principales.
Un ***pooling*** de taille k transforme une matrice de taille $n \times p$ en une matrice de taille k fois plus petite, c'est-à-dire de taille $n//k \times p//k$. Une sous-matrice de taille $k \times k$ de la matrice de départ produit un seul coefficient de la matrice d'arrivée.

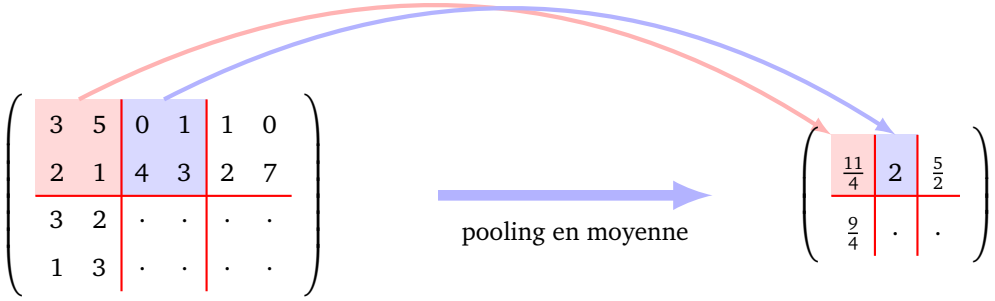


On distingue deux types de pooling.

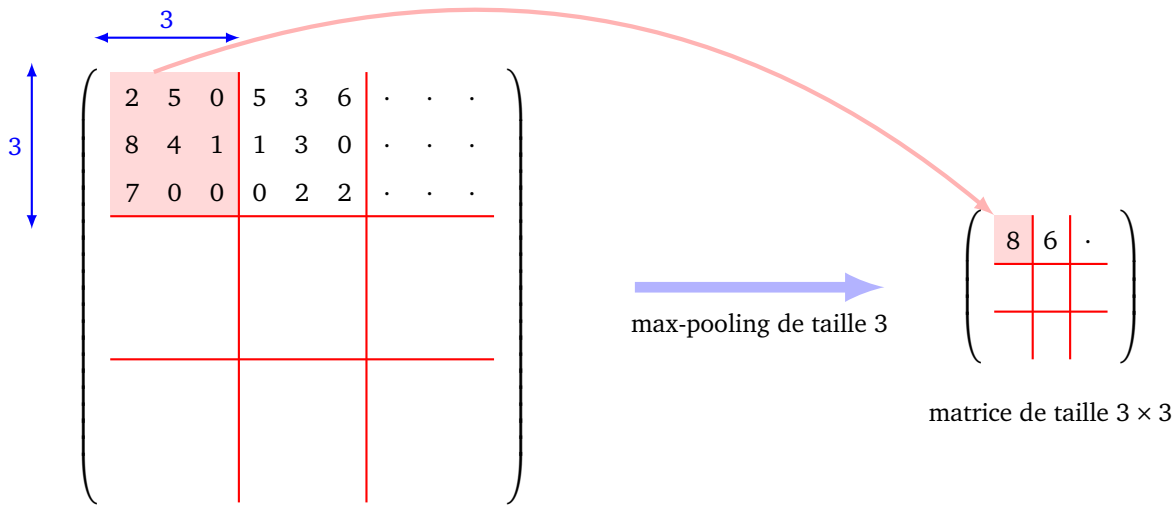
- Le ***max-pooling*** de taille k consiste à retenir le maximum de chaque sous-matrice de taille $k \times k$:



- Le ***pooling en moyenne*** de taille k (*average pooling*) consiste à retenir la moyenne des termes de chaque sous-matrice de taille $k \times k$:



Ci-dessous, voici le début d'un max-pooling de taille 3. La matrice de départ de taille 9×9 est transformée en une matrice de taille 3×3.



matrice de taille 9 × 9

matrice de taille 3 × 3

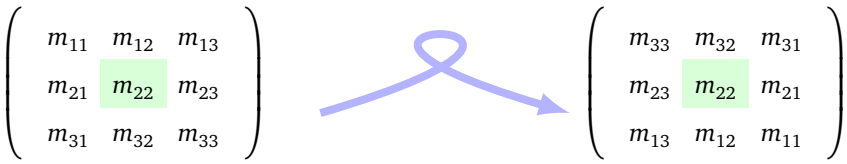
Le max-pooling, qui ne retient que la valeur la plus élevée par sous-matrice, permet de détecter la présence d'une caractéristique (par exemple un pixel blanc dans une image noire). Tandis que le pooling en moyenne prend en compte tous les termes de chaque sous-matrice (par exemple avec 4 pixels d'une image de ciel, on retient la couleur moyenne).

2. Convolution : deux dimensions

2.1. Définition

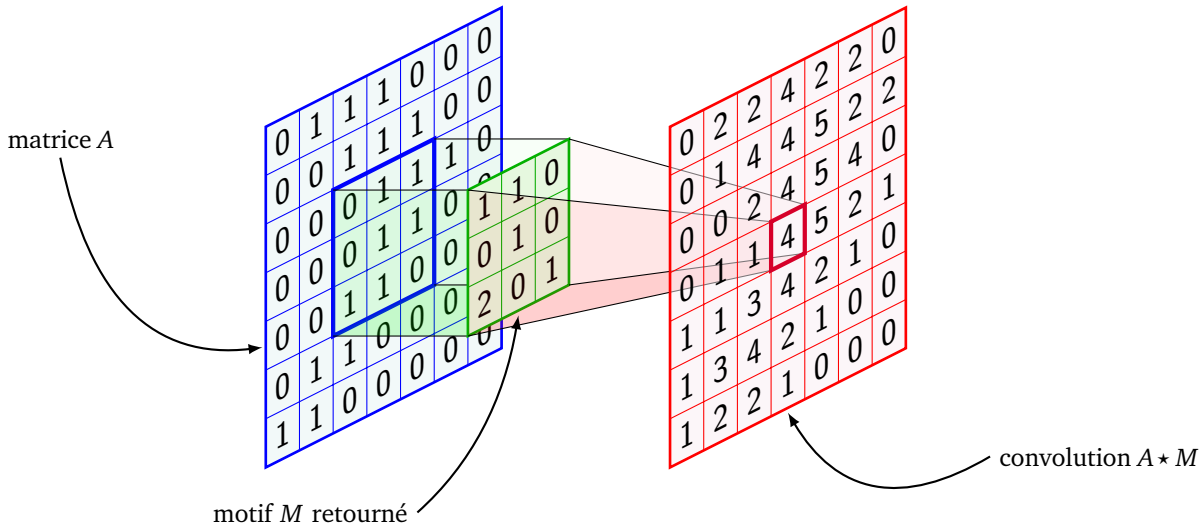
La convolution en deux dimensions est une opération qui :
- à partir d'une matrice d'entrée notée A,
- et d'une matrice d'un motif noté M,

associe une matrice de sortie $A \star M$.
Tout d'abord, il faut retourner la matrice M :

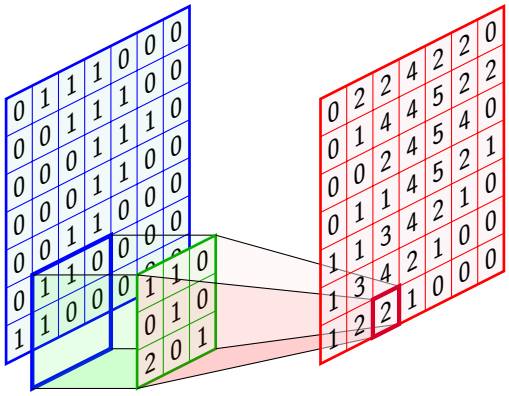


Motif de taille 3 × 3 **Motif retourné**

Le calcul de $A \star M$ s'effectue coefficient par coefficient :
- on centre le motif retourné sur la position du coefficient à calculer,
- on multiplie chaque coefficient de A par le coefficient du motif retourné en face (quitte à ajouter des zéros virtuels sur les bords de A),
- la somme de ces produits donne un coefficient de $A \star M$.



Voici un autre schéma pour le calcul d'un autre coefficient. Pour ce calcul, on rajoute des zéros virtuels autour de la matrice A.



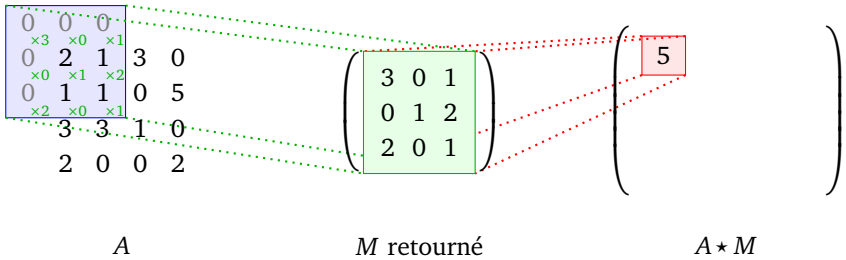
Exemple.
Calculons la convolution $A \star M$ définie par :

$$\begin{pmatrix} 2 & 1 & 3 & 0 \\ 1 & 1 & 0 & 5 \\ 3 & 3 & 1 & 0 \\ 2 & 0 & 0 & 2 \end{pmatrix} \star \begin{pmatrix} 1 & 0 & 2 \\ 2 & 1 & 0 \\ 1 & 0 & 3 \end{pmatrix}.$$

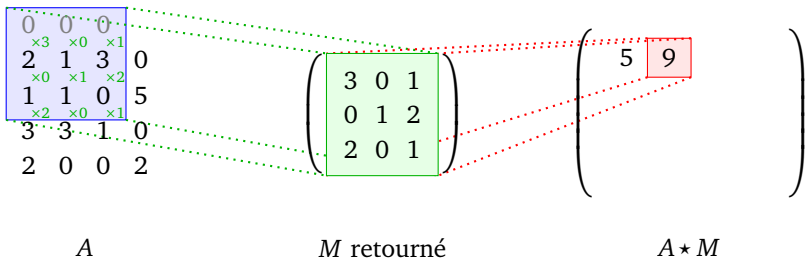
On commence par retourner M. Pour calculer le premier coefficient de $A \star M$, on centre le motif sur le premier coefficient de A, puis on rajoute des zéros virtuels à gauche et en haut. Ensuite on calcule les produits des coefficients de la matrice M retournée avec les coefficients de A correspondants, et on les additionne :

$$0 \times 3 + 0 \times 0 + 0 \times 1 + 0 \times 0 + 2 \times 1 + 1 \times 2 + 0 \times 2 + 1 \times 0 + 1 \times 1.$$

Cette somme vaut 5, c'est le premier coefficient de $A \star M$.

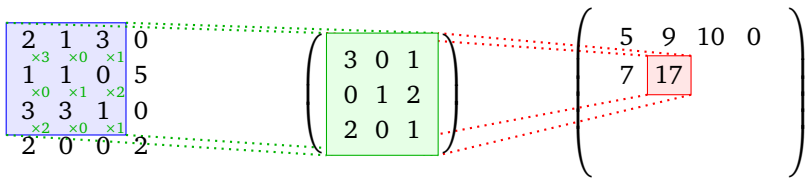


On continue avec le second coefficient.

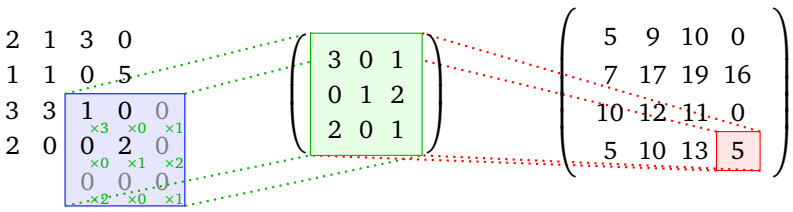


A M retourné $A \star M$

Et ainsi de suite :

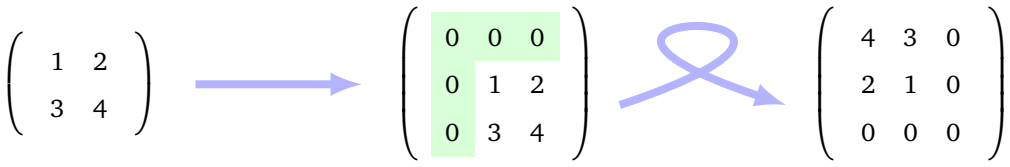


Jusqu'à calculer entièrement la matrice $A \star M$.



Remarque.

- Il ne faut pas confondre le fait de retourner la matrice avec une opération différente qui s'appelle la transposition.
- Dans la plupart de nos situations, le motif sera une matrice de taille 3×3. Par contre la matrice A pourra être de très grande taille (par exemple une matrice 600×400, codant une image).
- Cependant, si la matrice du motif possède une dimension paire on rajoute des zéros virtuels à gauche ou en haut (avant de la retourner).



Motif de taille 2×2 **Motif augmenté** **Motif retourné**

- Contrairement au pooling, les motifs disposés pour des calculs d'éléments voisins se superposent. Sur le dessin ci-dessous le motif est d'abord centré sur le coefficient 8, puis sur le coefficient 9.

$$\begin{pmatrix} 1 & 2 & 3 & 4 & 5 & 6 \\ 7 & 8 & 9 & 10 & 11 & 12 \\ 13 & 14 & 15 & 16 & 17 & 18 \\ & & & & & \\ & & & & & \\ & & & & & \end{pmatrix}$$

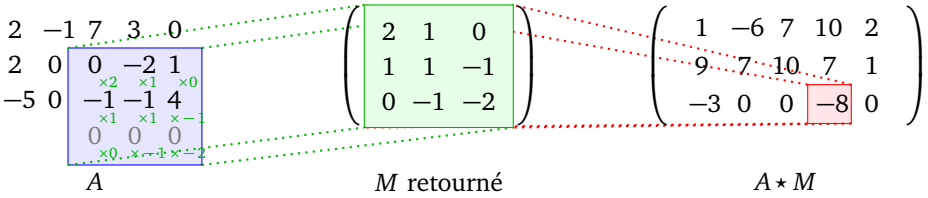
- On verra dans les chapitres « Convolution avec Python » et « Convolution avec tensorflow/keras » plus de paramètres pour la convolution (comment déplacer le motif, quel choix faire au bord de la matrice A, etc.).
- Par définition, il faut retourner la matrice M avant de calculer les produits. Cela complique l'usage mais est cohérent avec la définition donnée pour la dimension 1 et cela permettra d'avoir de bonnes propriétés mathématiques (voir le chapitre « Convolution avec Python »).

2.2. Exemples simples

Exemple 3×3**.** Voici un autre exemple de calcul :

$$\begin{pmatrix} 2 & -1 & 7 & 3 & 0 \\ 2 & 0 & 0 & -2 & 1 \\ -5 & 0 & -1 & -1 & 4 \end{pmatrix} \star \begin{pmatrix} -2 & -1 & 0 \\ -1 & 1 & 1 \\ 0 & 1 & 2 \end{pmatrix}.$$

Il ne faut pas oublier de retourner M avant de commencer !



Translation. Une convolution par la matrice

$$\begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 1 & 0 \end{pmatrix}$$

correspond à une translation des coefficients vers le bas. Par exemple :

$$\begin{pmatrix} 1 & 2 & 3 & 4 \\ 5 & 6 & 7 & 8 \\ 9 & 10 & 11 & 12 \\ 13 & 14 & 15 & 16 \end{pmatrix} \star \begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 1 & 0 \end{pmatrix} = \begin{pmatrix} 0 & 0 & 0 & 0 \\ 1 & 2 & 3 & 4 \\ 5 & 6 & 7 & 8 \\ 9 & 10 & 11 & 12 \end{pmatrix}.$$

De même pour une translation des coefficients vers la droite.

$$\begin{pmatrix} 1 & 2 & 3 & 4 \\ 5 & 6 & 7 & 8 \\ 9 & 10 & 11 & 12 \\ 13 & 14 & 15 & 16 \end{pmatrix} \star \begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 0 \end{pmatrix} = \begin{pmatrix} 0 & 1 & 2 & 3 \\ 0 & 5 & 6 & 7 \\ 0 & 9 & 10 & 11 \\ 0 & 13 & 14 & 15 \end{pmatrix}.$$

Moyenne. On effectue une moyenne locale des coefficients à l'aide du motif :

$$\frac{1}{9}\begin{pmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{pmatrix}.$$

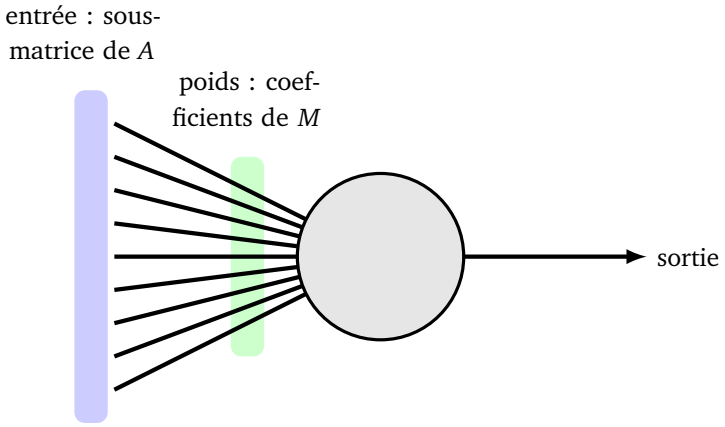
Par exemple :

$$\begin{pmatrix} 1 & 2 & 3 & 4 & 5 \\ 6 & 7 & 8 & 9 & 10 \\ 11 & 12 & 13 & 14 & 15 \\ 16 & 17 & 18 & 19 & 20 \\ 21 & 22 & 23 & 24 & 25 \end{pmatrix} \star \frac{1}{9} \begin{pmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{pmatrix} \simeq \begin{pmatrix} 1.77 & 3 & 3.66 & 4.33 & 3.11 \\ 4.33 & 7 & 8 & 9 & 6.33 \\ 7.66 & 12 & 13 & 14 & 9.66 \\ 11 & 17 & 18 & 19 & 13 \\ 8.44 & 13 & 13.66 & 14.33 & 9.77 \end{pmatrix}.$$

On remarque des phénomènes de bords dus à l'ajout de zéros virtuels.

2.3. Neurone de convolution

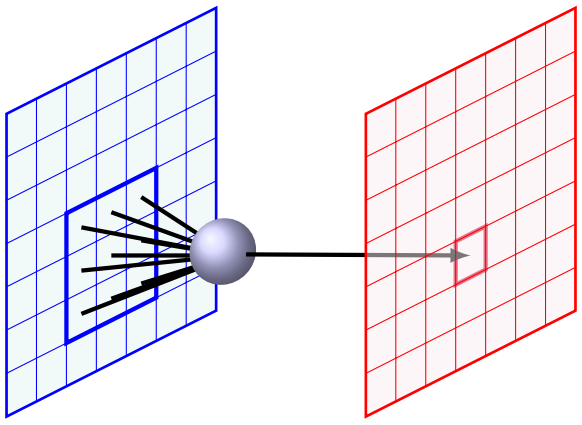
À l'aide de la convolution, nous allons définir un nouveau type de couche de neurones : une « couche de convolution ». Tout d'abord un ***neurone de convolution*** de taille 3×3 est un neurone classique ayant 9 entrées (pour l'instant la fonction d'activation est l'identité).



Les poids du neurone correspondent aux coefficients d'une matrice de convolution :

$$\begin{pmatrix} m_{11} & m_{12} & m_{13} \\ m_{21} & m_{22} & m_{23} \\ m_{31} & m_{32} & m_{33} \end{pmatrix}.$$

Imaginons que l'entrée soit une image ou un tableau de dimension 2, pour nous ce sera une matrice A. Alors un neurone de convolution est relié à une sous-matrice 3×3 de A.



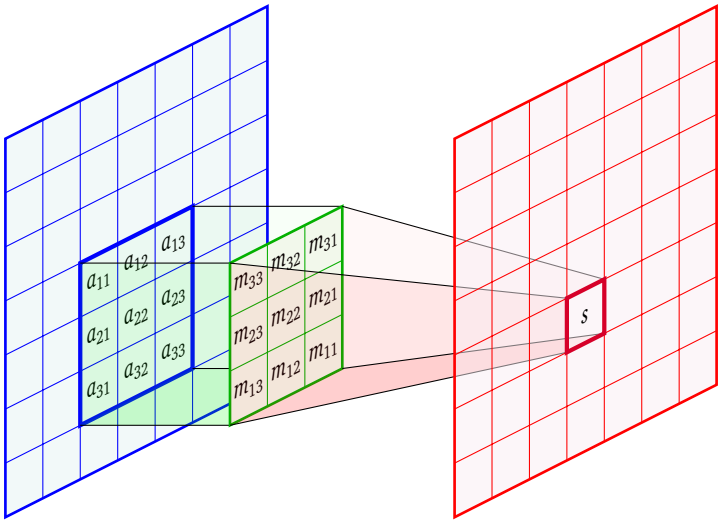
Ce neurone agit comme un élément de convolution. Par exemple si la sous-matrice de A est notée

$$\begin{pmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{pmatrix}$$

alors le neurone produit la sortie :

$$s = a_{11} \cdot m_{33} + a_{12} \cdot m_{32} + a_{13} \cdot m_{31} + a_{21} \cdot m_{23} + a_{22} \cdot m_{22} + a_{23} \cdot m_{21} + a_{31} \cdot m_{13} + a_{32} \cdot m_{12} + a_{33} \cdot m_{11}$$

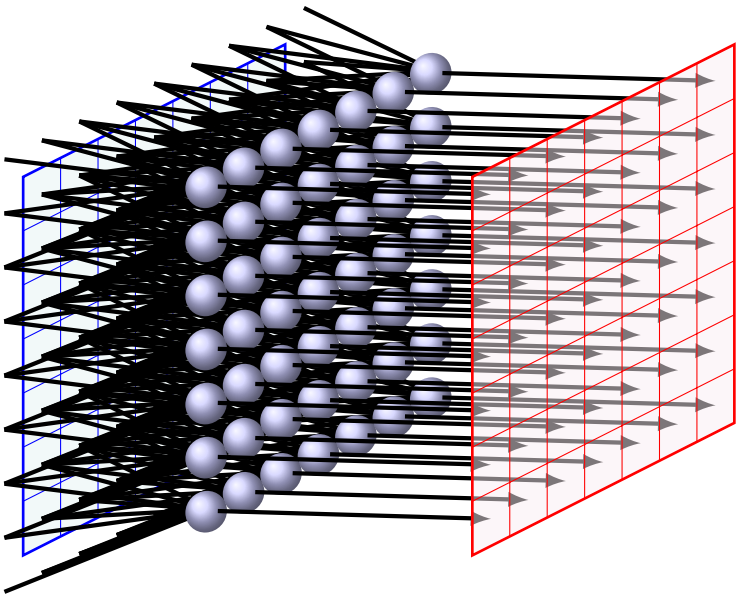
N'oublier pas que dans le produit de convolution, on commence par retourner la matrice M.



On pourrait aussi composer par une fonction d'activation au niveau du neurone, mais on le fera plus tard à l'aide d'une « couche d'activation ». Plus généralement un neurone de convolution de taille $p \times q$ possède pq arêtes et donc pq poids à définir ou à calculer.

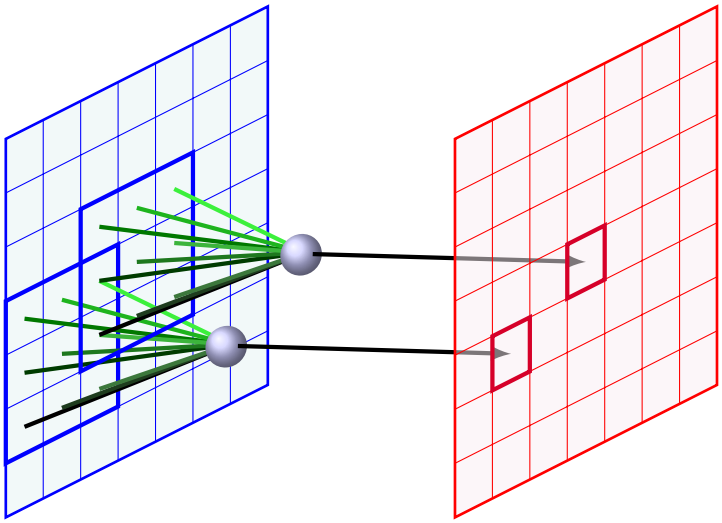
2.4. Couche de convolution

Considérons une entrée A représentée par une matrice de taille $n \times p$.
Une ***couche de convolution*** (pour un seul motif) est la donnée d'une matrice M appelé ***motif*** (par exemple de taille 3×3) et qui renvoie en sortie les coefficients de la matrice $A \star M$.



Pour une entrée de taille $n \times p$, il y a donc np neurones de convolutions, chacun ayant 9 arêtes (car le motif est de taille 3×3). Il est très important de comprendre que les poids sont communs à tous les neurones (ce sont les coefficients de M). Ainsi, pour une couche de convolution, il y a seulement 9 poids à déterminer pour définir la couche de convolution (bien que le nombre de neurones puisse être très grand).
Sur le dessin ci-dessous on ne montre que deux neurones. Noter deux choses importantes : (a) deux neurones différents peuvent avoir certaines entrées communes (sur le dessin les deux carrés 3×3 s'intersectent) et

(b) deux neurones différents ont les mêmes poids (sur le dessin les arêtes entrantes ayant la même couleur, ont les mêmes poids).



Remarque.

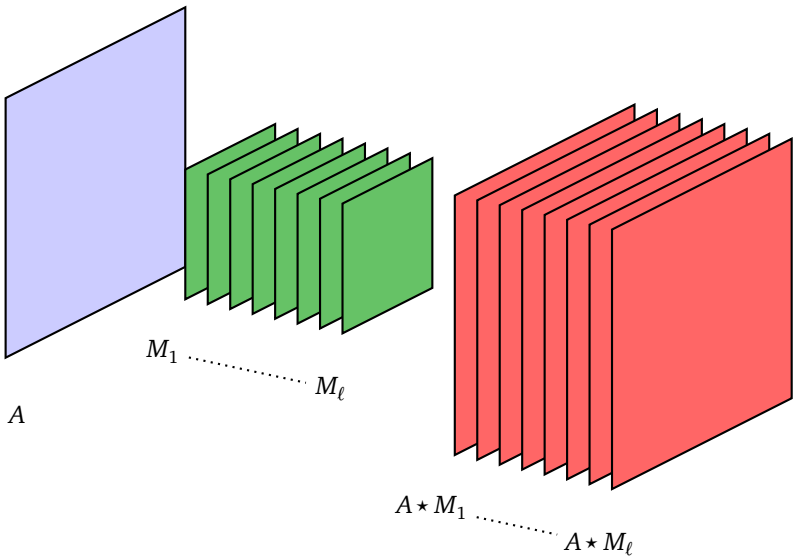
- Terminologie : le motif M s'appelle aussi le ***noyau*** (*kernel*), le ***filtre*** ou encore le ***masque***.
- Le motif peut être d'une taille différente de la taille 3×3 considérée dans les exemples qui est cependant la plus courante.
- Il existe également des variantes avec biais ou fonction d'activation.
- Combinatoire : dans le cas d'une couche complètement connectée, le nombre de poids à calculer serait énorme. En effet, une couche de np neurones complètement connectée à une entrée de taille $n \times p$ amènerait à calculer $(np)^2$ poids. Pour $n = p = 100$, cela fait 10 000 neurones et 100 000 000 poids. Pour rappel, notre couche de convolution est définie par 9 poids (quels que soient n et p).
- D'un point de vue mathématique, une couche de convolution associée au motif M est l'application :

$$\begin{array}{rccc} F : & \mathcal{M}_{n,p} & \longrightarrow & \mathcal{M}_{n,p} \\ & A & \longmapsto & A \star M \end{array}$$

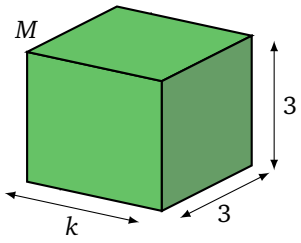
où $\mathcal{M}_{n,p}$ est l'ensemble des matrices de taille $n \times p$.
Si on transforme une matrice en un (grand) vecteur, alors on obtient une application $F : \mathbb{R}^{np} \to \mathbb{R}^{np}$.

2.5. Différentes couches d'un réseau de neurones

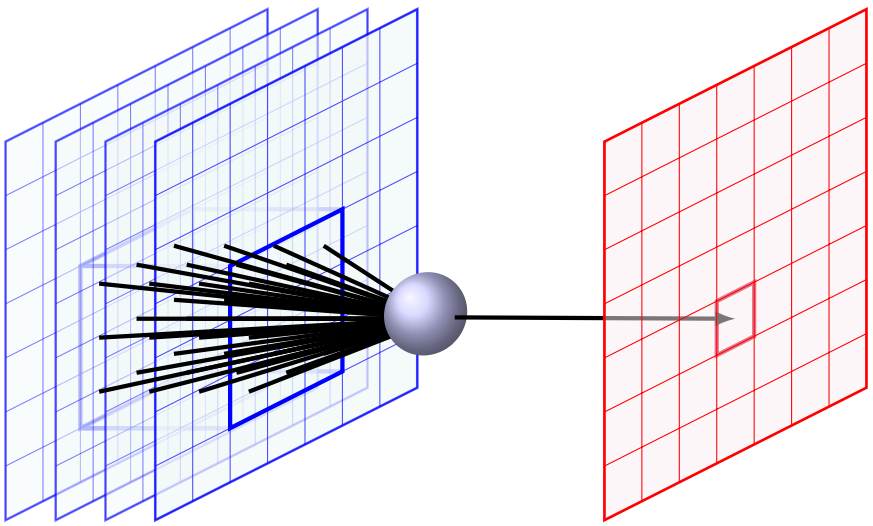
Les couches de convolution sont au cœur des réseaux de neurones modernes. Voici un petit survol des types de couches qui seront mises en pratique dans le chapitre « Convolution avec tensorflow/keras ».
Plusieurs filtres. On verra dans la pratique qu'une couche de convolution est définie avec plusieurs motifs, c'est-à-dire un nombre ℓ de matrices $M_1, M_2, \ldots, M_\ell$. Ainsi pour une entrée A de taille $n \times p$, une couche de convolution à ℓ motifs renvoie une sortie de taille $n \times p \times \ell$, correspondant à $(A \star M_1, A \star M_2, \ldots, A \star M_\ell)$.



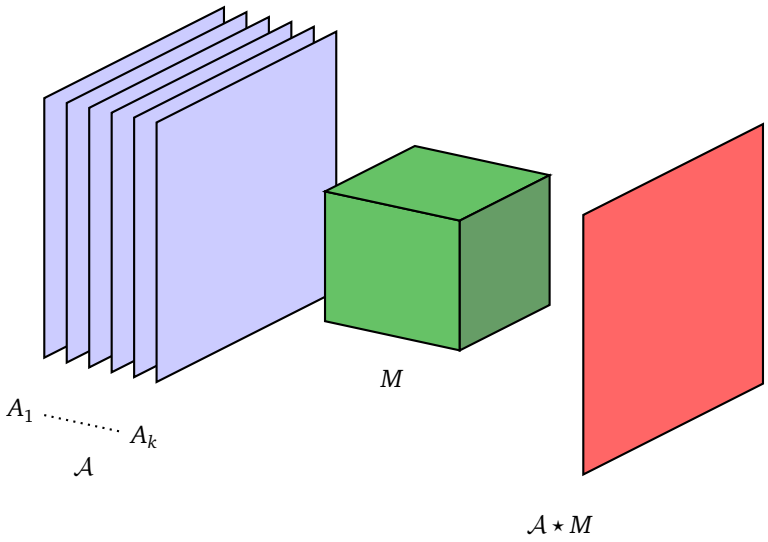
Convolution à partir de plusieurs canaux. Pour traiter une sortie de taille $n \times p \times k$ du type précédent, qui va être l'entrée de la couche suivante on doit définir une convolution ayant plusieurs canaux en entrée. Si les entrées sont les $A_1, A_2, \ldots, A_k$ alors un motif M associé à cette entrée de profondeur k est un 3-tenseur de taille $(3,3,k)$, c'est-à-dire un tableau à 3 dimensions de la forme $3 \times 3 \times k$ (on renvoie au chapitre « Tenseurs » pour la définition).



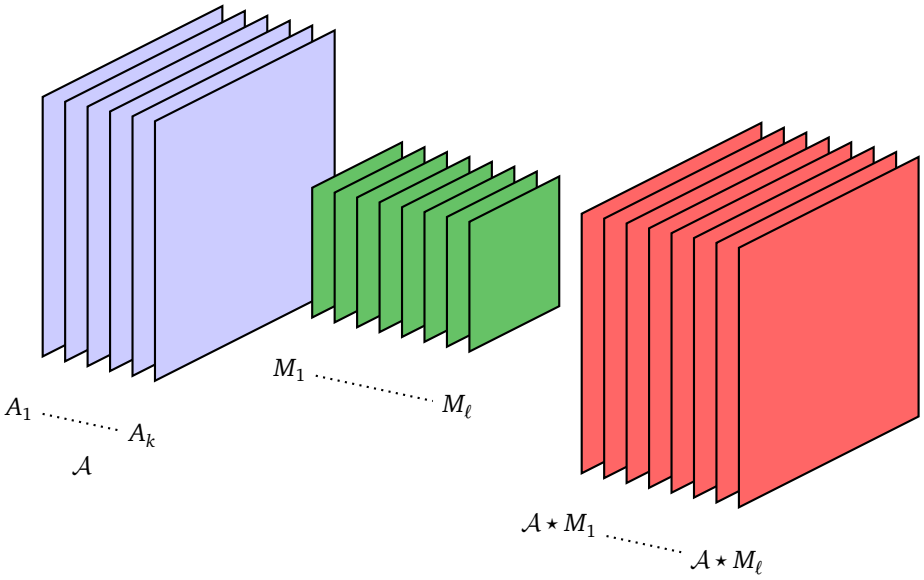
Ce motif M correspond donc à une couche de neurones où chaque neurone possède $3 \times 3 \times k$ arêtes. Chaque neurone est connecté localement à une zone 3×3 dans un plan, mais ceci sur toute la profondeur des k entrées. Le calcul de la sortie du neurone se fait en réalisant la somme de $3 \times 3 \times k$ termes. Pour réaliser une couche de neurones, la zone 3×3 se déplace dans le plan, mais s'étend toujours sur toute la profondeur.



On note $\mathcal{A} = (A_1, A_2, \ldots, A_k)$ et $\mathcal{A} \star M$ le produit de convolution. Si l'entrée $\mathcal{A}$ est de taille (n,p,k) alors la sortie $\mathcal{A} \star M$ associée au motif M est de taille (n,p).



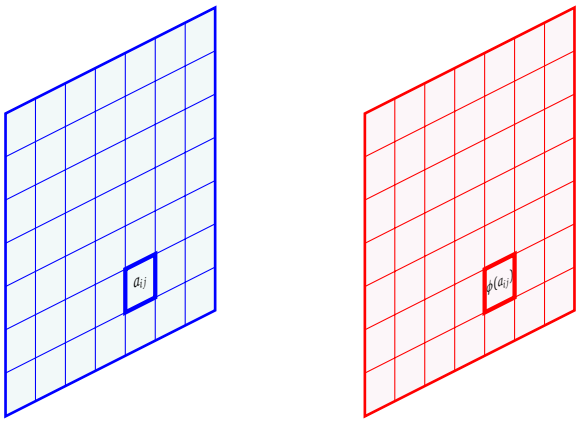
Convolution à plusieurs filtres à partir de plusieurs canaux. C'est le cas général dans la pratique. Une entrée donnée par plusieurs canaux $\mathcal{A} = (A_1, A_2, \ldots, A_k)$, associée à des motifs $M_1, M_2, \ldots, M_\ell$ (qui sont donc chacun des 3-tenseurs de taille $(3,3,k)$) produit une sortie de taille $n \times p \times \ell$, correspondant à $(\mathcal{A} \star M_1, \mathcal{A} \star M_2, \ldots, \mathcal{A} \star M_\ell)$. Si l'entrée $\mathcal{A}$ est de taille (n,p,k) alors la sortie est de taille (n,p,ℓ).



Noter que sur cette figure chaque motif M_i est représenté par un carré 3×3, alors qu'en fait chacun devrait être une boîte en 3 dimensions de taille $3 \times 3 \times k$.

Activation. On peut composer par une fonction d'activation ϕ directement dans le neurone ou bien dans une couche à part (ce qui est équivalent). Comme d'habitude, la fonction d'activation est la même pour tous les neurones d'une couche. Ainsi une ***couche d'activation*** pour la fonction d'activation ϕ, transforme une entrée

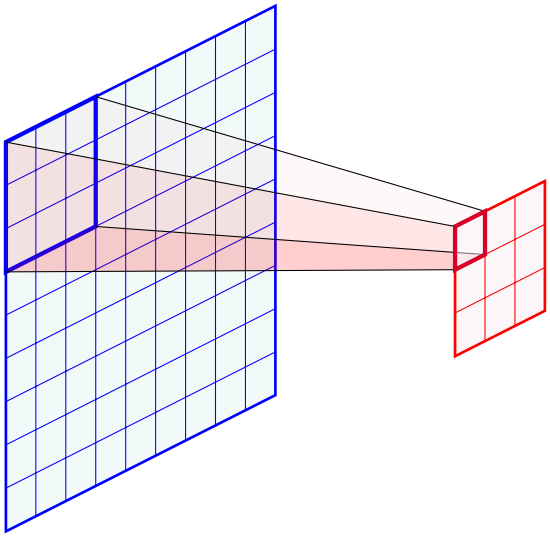
$$\begin{pmatrix} a_{11} & a_{12} & \ldots & a_{1p} \\ a_{21} & a_{22} & \ldots & a_{2p} \\ \vdots & \vdots & \ddots & \vdots \\ a_{n1} & a_{n2} & \ldots & a_{np} \end{pmatrix} \quad \text{en} \quad \begin{pmatrix} \phi(a_{11}) & \phi(a_{12}) & \ldots & \phi(a_{1p}) \\ \phi(a_{21}) & \phi(a_{22}) & \ldots & \phi(a_{2p}) \\ \vdots & \vdots & \ddots & \vdots \\ \phi(a_{n1}) & \phi(a_{n2}) & \ldots & \phi(a_{np}) \end{pmatrix}.$$



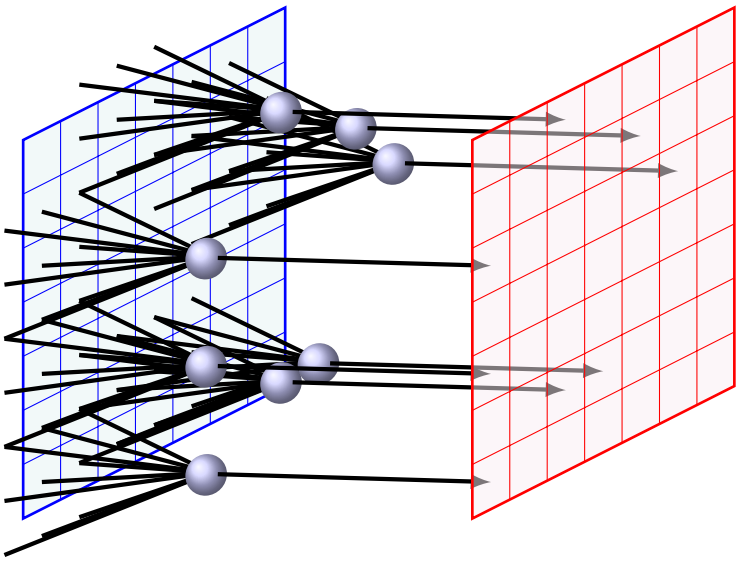
matrice A activation ϕ

Les fonctions d'activation sont les mêmes que celles rencontrées auparavant : ReLU, σ, th...

Pooling. Une couche de pooling de taille k transforme une entrée de taille $n \times p$ en une sortie de taille $n//k \times p//k$. Nous renvoyons à la première section de ce chapitre pour le max-pooling ou le pooling en moyenne. Cette opération permet de réduire les données d'un facteur k^2. Par exemple, une entrée de taille 100×80 (8000 données) avec un pooling de taille 4 fournit une sortie de taille 25×20 (500 données). Ci-dessous un pooling de taille 3 transforme une matrice 9×9 en une matrice 3×3.



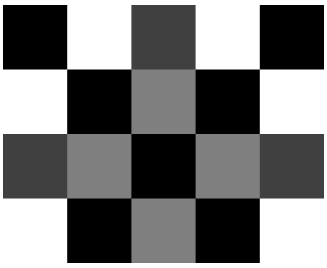
Dropout. Le ***dropout*** (décrochage ou abandon en français) est une technique pour améliorer l'apprentissage d'un réseau et en particulier pour prévenir le sur-apprentissage. L'idée est de désactiver certains neurones d'une couche lors des étapes de l'apprentissage. Ces neurones sont choisis au hasard et sont désactivés temporairement pour une itération (par exemple on peut choisir à chaque itération de désactiver un neurone avec une probabilité $\frac{1}{2}$). Cela signifie que l'on retire toute arête entrante et toute arête sortante de ces neurones, ce qui revient à mettre les poids à zéro tant pour l'évaluation que pour la rétropropagation. Lors de l'itération suivante on choisit de nouveau au hasard les neurones à désactiver. Voir le chapitre « Probabilités » pour plus de détails.



3. Traitement des images

La convolution est un outil puissant dans le domaine de la reconnaissance d'images et est indispensable dans les réseaux de neurones modernes. Pour mieux comprendre ce que permet la convolution, nous allons voir comment certains motifs transforment les images.
Dans cette section, une image en niveaux de gris est une matrice ayant ses entrées entre 0 et 255. Chaque entrée de la matrice représente un pixel, 0 pour un pixel noir, 255 pour un pixel blanc, les valeurs intermédiaires étant des nuances de gris.
Voici un petit exemple.

$$A = \begin{pmatrix} 0 & 255 & 64 & 255 & 0 \\ 255 & 0 & 127 & 0 & 255 \\ 64 & 127 & 0 & 127 & 64 \\ 255 & 0 & 127 & 0 & 255 \end{pmatrix}$$



C'est sur une image/matrice A que l'on va appliquer une convolution.

3.1. Flou

On obtient une image floue par application du motif :

$$M = \frac{1}{9}\begin{pmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{pmatrix}.$$

Image originale

Flou

En effet, la convolution par cette matrice M remplace la couleur d'un pixel par la moyenne de la couleur de ses 9 pixels voisins. Outre l'aspect esthétique, il y a un intérêt fondamental : le flou réduit le bruit. Par exemple, si un pixel est mauvais (par exemple à cause d'une erreur de l'appareil photo), le flou élimine cette erreur.
Une variante est le *flou gaussien* défini par la matrice :

$$M = \frac{1}{16}\begin{pmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{pmatrix}.$$

Il existe aussi des flous définis par des matrices M de taille 5×5, qui tiennent compte des 25 pixels voisins.

3.2. Piqué

C'est en quelque sorte le contraire du flou ! Le motif est :

$$M = \begin{pmatrix} 0 & -1 & 0 \\ -1 & 5 & -1 \\ 0 & -1 & 0 \end{pmatrix}.$$

Voici un exemple avec l'image originale à gauche et l'image transformée à droite qui a l'air plus nette que l'originale !

Image originale

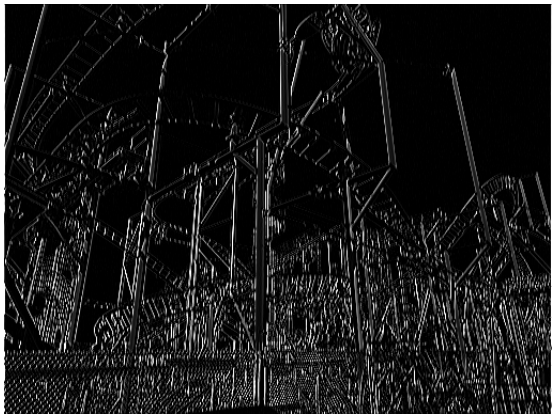
Piqué

3.3. Verticales et horizontales

Le motif suivant renforce les lignes verticales :

$$M = \begin{pmatrix} -1 & 2 & -1 \\ -1 & 2 & -1 \\ -1 & 2 & -1 \end{pmatrix}.$$

Image originale

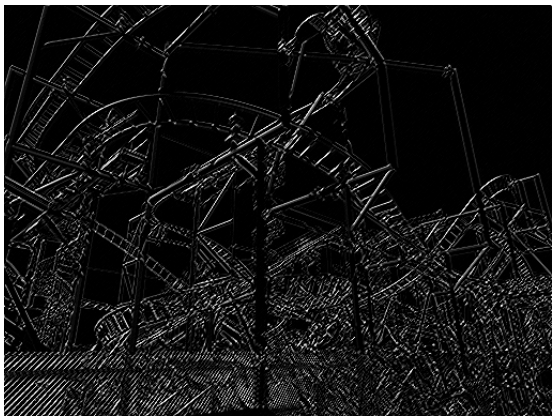


Verticales

De même pour les lignes horizontales (à gauche) ou les lignes à 45° (à droite) :

$$M = \begin{pmatrix} -1 & -1 & -1 \\ 2 & 2 & 2 \\ -1 & -1 & -1 \end{pmatrix}, \qquad M = \begin{pmatrix} -1 & -1 & 2 \\ -1 & 2 & -1 \\ 2 & -1 & -1 \end{pmatrix}.$$

Horizontales



Diagonales

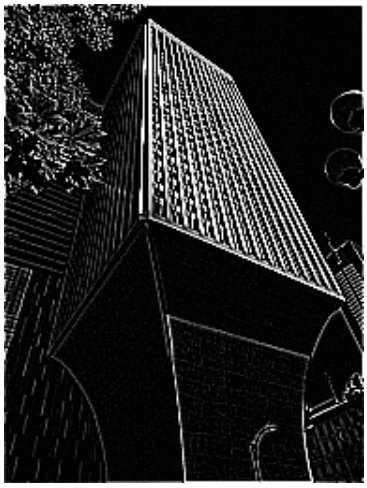
Ce que l'on retient, c'est que la convolution permet de détecter des lignes verticales ou horizontales par exemple et donc d'extraire des caractéristiques abstraites d'une image.

3.4. Bords

On peut plus généralement extraire les bords des formes d'une image avec :

$$M = \begin{pmatrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{pmatrix}.$$

Image originale



Contour

Des variantes sont les matrices :

$$M = \begin{pmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{pmatrix} \quad \text{ou} \quad M = \begin{pmatrix} 1 & 0 & -1 \\ 0 & 0 & 0 \\ -1 & 0 & 1 \end{pmatrix}.$$

3.5. Mise en relief

En jouant sur les contrastes, on peut créer une impression de relief avec :

$$M = \begin{pmatrix} -2 & -1 & 0 \\ -1 & 1 & 1 \\ 0 & 1 & 2 \end{pmatrix}.$$

Image originale

Relief

3.6. Pooling

On termine cette section en illustrant l'action du pooling sur une image.

Image originale

Max-pooling 4 × 4

Pooling 4 × 4 en moyenne

Les images obtenues comportent 16 fois moins de pixels. Les voici agrandies.

Max-pooling 4 × 4

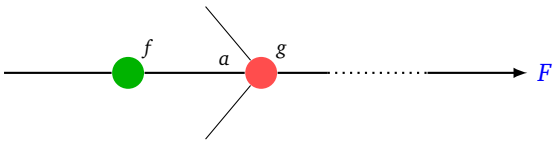
Pooling 4 × 4 en moyenne

4. Rétropropagation

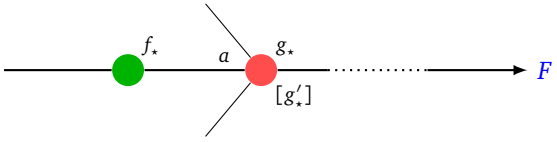
Les formules de la rétropropagation s'étendent sans problème au cas des neurones de convolution. On commence par reprendre les calculs du chapitre « Gradient » pour le cas classique avant de s'occuper du cas de la convolution.

Cas classique.

Voici la situation d'un neurone. On ne s'intéresse qu'à une seule de ses arêtes d'entrée : celle associée au poids a et de valeur d'entrée f. Ce neurone a pour fonction d'activation g. La sortie de ce neurone est utilisée par le reste du réseau. À la fin, on obtient une valeur de sortie pour la fonction F.



On distingue la fonction de sa valeur en un point : on note f la fonction et $f_\star$ la valeur de la fonction à la sortie du neurone correspondant. De même pour g et $g_\star$ et la valeur de la dérivée $g'_\star$.



On reprend la formule du chapitre « Gradient » :

$$\frac{\partial F}{\partial a} = \frac{\partial F}{\partial g} \cdot f_\star \cdot g'_\star.$$

Nous avions vu comment cette formule permettait de calculer le gradient par rapport aux poids.

Preuve. Tout d'abord

$$\frac{\partial F}{\partial a} = \frac{\partial F}{\partial g} \cdot \frac{\partial g}{\partial a}.$$

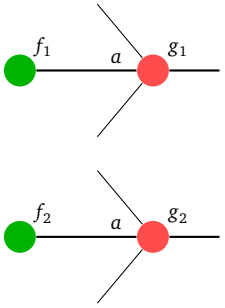
Or

$$\frac{\partial g}{\partial a} = f_\star \cdot g'_\star$$

car $g_\star = g(\cdots + a f_\star + \cdots)$. La formule s'obtient en dérivant $a \mapsto g(\cdots + a f + \cdots)$ par rapport à la variable a.

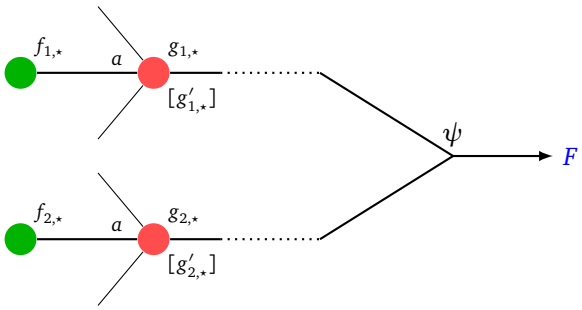
Cas de la convolution.

On se place maintenant dans une couche de convolution. Les neurones de cette couche ont des poids en commun, il faut donc en tenir compte dans les formules. Imaginons une couche de convolution avec deux neurones.



Ces deux neurones ont un poids commun a, par contre les entrées pour ce poids $f_{1,\star}$ et $f_{2,\star}$ peuvent être différentes et les sorties $g_{1,\star}$ et $g_{2,\star}$ également (même si les fonctions d'activation g_1 et g_2 sont les mêmes). Le réseau continue. À la fin, nous obtenons un fonction de sortie F qui dépend des sorties g_1 et g_2 de nos deux neurones :

$$F = \psi(g_1, g_2).$$



Nous avons la nouvelle formule :

$$\frac{\partial F}{\partial a} = \frac{\partial F}{\partial g_1} \cdot f_{1,\star} \cdot g'_{1,\star} + \frac{\partial F}{\partial g_2} \cdot f_{2,\star} \cdot g'_{2,\star}.$$

Preuve. Tout d'abord nous avons $F = \psi(g_1, g_2)$, donc

$$\frac{\partial F}{\partial g_1} = \frac{\partial \psi}{\partial x}(g_1, g_2) \quad \text{et} \quad \frac{\partial F}{\partial g_2} = \frac{\partial \psi}{\partial y}(g_1, g_2).$$

On se souvient de la formule de dérivation de la composition de

$$F(a) = \psi\big(u(a), v(a)\big)$$

qui est

$$F'(a) = u'(a)\frac{\partial \psi}{\partial x}(u(a), v(a)) \; + \; v'(a)\frac{\partial \psi}{\partial y}(u(a), v(a)).$$

Comme on peut aussi écrire que :

$$F = \psi\big(g_1(\cdots + af_1 + \cdots), g_2(\cdots + af_2 + \cdots)\big)$$

alors

$$\begin{aligned}\frac{\partial F}{\partial a} = {} & f_1 \cdot g_1'(\cdots + af_1 + \cdots)\frac{\partial \psi}{\partial x}\big(g_1(\cdots + af_1 + \cdots), g_2(\cdots + af_2 + \cdots)\big) \\ & + f_2 \cdot g_2'(\cdots + af_1 + \cdots)\frac{\partial \psi}{\partial y}\big(g_1(\cdots + af_1 + \cdots), g_2(\cdots + af_2 + \cdots)\big).\end{aligned}$$

Ce qui donne bien :

$$\frac{\partial F}{\partial a} = f_{1,\star} \cdot g'_{1,\star} \cdot \frac{\partial F}{\partial g_1} + f_{2,\star} \cdot g'_{2,\star} \cdot \frac{\partial F}{\partial g_2}.$$

Si les fonctions d'activation de la couche de convolution sont l'identité alors $g'_{1,\star} = 1$ et $g'_{2,\star} = 1$, on obtient dans ce cas la formule :

$$\frac{\partial F}{\partial a} = f_{1,\star} \cdot \frac{\partial F}{\partial g_1} + f_{2,\star} \cdot \frac{\partial F}{\partial g_2}.$$

On retient que chaque entrée associée au poids a contribue proportionnellement à sa valeur dans le calcul de la dérivée partielle par rapport à ce poids a.

Plus généralement, pour des entrées f_i, $i = 1, \ldots, n$, pour n neurones d'une couche de convolution ayant pour sortie g_i, $i = 1, \ldots, n$, et pour un poids a partagé par ces neurones :

$$\frac{\partial F}{\partial a} = \sum_{i=1}^{n} f_{i,\star} \cdot g'_{i,\star} \cdot \frac{\partial F}{\partial g_i}.$$