Convolution 2D en Deep Learning

Introduction

La convolution en Deep Learning a été introduit/popularisé par Yann LeCun et ses collègues dans les années 80 et 90. En mathématiques, la convolution apparaît dans de nombreux domaines mais celui qui nous intéresse est le traitement d'images.

1 Définition et Exemples

Dans le contexte de la convolution 2D en Deep Learning, nous considérons une image/matrice M $\in \mathbb{R}^{h\times w\times c}$(ou $\mathbb{M}_{h,w,c}(\mathbb{R})$), avec w,h,c $\in \mathbb{N}^*$ représentant respectivement les colonnes, les lignes de l'image et le nombre de canaux (généralement $c = 3$). Soit un filtre F $\in [-1;1]^{k_h\times k_w\times c}$, avec $k_h, k_w, c \in \mathbb{N}^*$ et $k_h \leq$ h (et) $k_w \leq$ w.

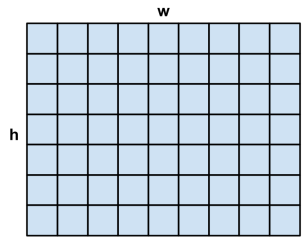

FIGURE 1 – Image/Matrice M

La convolution 2D de l'image M par le filtre F est une matrice C $\in \mathbb{R}^{n\times p}$ avec n,p $\in \mathbb{N}^*$:

$\forall(i,j) \in [|1,n|] \times [|1,p|]$

$$C_{i,j} = \sum_{k=1}^{k_h}\sum_{l=1}^{k_w}\sum_{m=1}^{c} F_{k,l,m} \times M_{i+k,j+l,m}$$

Les expressions de n et p seront détaillées dans la partie suivante car ces dernières dépendent de caractéristiques. Elles seront notées h_{fm} et w_{fm}, "fm" pour features map (en français : "**carte caractéristiques**").

FIGURE 2 – Exemple de convolution 2D avec un filtre F $\in [-1;1]^{7\times7\times3}$ généré aléatoirement avec $s = 8$ et $p = 0$

En Deep Learning, la couche convolutive permet de réduire l'information et d'extraire des caractéristiques locales avec la convolution et d'autres méthodes que nous verrons. Cette réduction de l'information peut-être visible sur l'image par l'effet pixelisé et le passage de l'image de RGB à NB.

2 Arithmétique de Convolution 2D

Maintenant, nous devons définir deux concepts, le "**Zero-Padding**" et le "**Stride**". Le Zero-Padding (en français : "**Remplissage de zéros**") consiste à ajouter p (entier naturel) zéros de chaque côté de la matrice. Le Zero-Padding est constitué de deux valeurs, une pour l'axe des abscisses (largeur), noté p_w, et une autre pour l'axe des ordonnées (hauteur), noté p_h. Le Stride (en français : "**pas**") est un paramètre qui représente le déplacement du filtre sur la matrice après chaque opération. Le Stride est constitué également de deux valeurs, noté s_w et s_h.

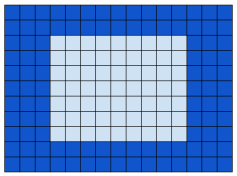

FIGURE 3 – Exemple de Padding ($p_w = 3$, $p_h = 2$, $w = 9$, $h = 7$)

Le filtre est appliqué horizontalement de gauche à droite, puis verticalement de haut en bas.

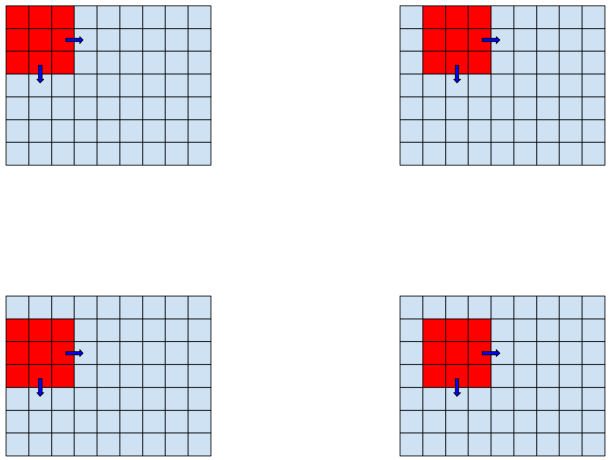

FIGURE 4 – Exemple de stride ($s_w = s_h = 1$, $k_h = k_w = 3$, $w = 9$, $h = 7$)

2.1 Stride Unitaire et Absence de Zero-Padding

Considérons $s_w = s_h = 1$, $p_w = p_h = 0$ et plaçons-nous sur l'axe des abscisses. Le filtre commence à partir du sommet supérieur gauche (origine image/matrice) et se déplace de gauche à droite. La largeur de la sortie résultant de la convolution d'une image est déterminée par le nombre de pas effectués et de la position initiale du filtre.

$\forall\ h, k_h$ tel que $k_h \leq h$

$$h_{fm} = h - k_h + 1 \tag{1}$$

Par la même logique, selon l'axe des abscisses.

$\forall\ w, k_w$ tel que $k_w \leq w$

$$w_{fm} = w - k_w + 1 \tag{2}$$

2.2 Stride Unitaire et Zero-Padding

En ajoutant p zéros de chaque côté de notre matrice revient à avoir une nouvelle matrice M' $\in \mathbb{R}^{(h+2p_h)\times(w+2p_w)\times c}$. En reprenant les équations développées précédemment pour cette matrice, nous obtenons :

$\forall\ h, k_h, p_h$ tel que $k_h \leq h$

$$h_{fm} = h + 2p_h - k_h + 1 \tag{3}$$

$\forall\ w, k_w, p_w$ tel que $k_w \leq w$

$$w_{fm} = w + 2p_w - k_w + 1 \tag{4}$$

2.3 Stride Unitaire et Same Padding

Le **same padding**, aussi appelé **half padding**, est une technique permettant d'avoir une taille de sortie égale à celle de l'entrée ($h_{fm} = h$ et $w_{fm} = w$). À partir de ces conditions et des équations (3) et (4) :

$$h_{fm} = h + 2p_h - k_h + 1$$

$$2p_h - k_h + 1 = 0 \quad \text{car } h_{fm} = h$$

$$k_h = 2p_h + 1$$

$$p_h = \frac{k_h - 1}{2}$$

Ainsi,

$$h_{fm} = h \iff k_h \text{ impair et } p_h = \frac{k_h - 1}{2} \tag{5}$$

$$w_{fm} = w + 2p_w - k_w + 1$$

$$2p_w - k_w + 1 = 0 \text{ car } w_{fm} = w$$

$$k_w = 2p_w + 1$$

$$p_w = \frac{k_w - 1}{2}$$

Ainsi,

$$w_{fm} = w \iff k_w \text{ impair et } p_w = \frac{k_w - 1}{2} \tag{6}$$

Les équivalences (5) et (6) nous indiquent que k_h et k_w doivent être impair. Cependant nous pouvons faire en sorte qu'avec une ou les deux dimensions k_h, k_w pairs pour le filtre, en ajoutant une colonne de zéros à gauche ou à droite ou bien une ligne de zéros en haut ou en bas selon l'axe en question pour obtenir une sortie de dimension égale à celle de l'entrée.
Ainsi, la matrice M devient la matrice M' $\in \mathbb{R}^{(h+1)\times(w+1)\times c}$ et nous obtenons :

$$h_{fm} = h + 1 + 2p_h - k_h + 1$$

$$2p_h - k_h + 2 = 0 \text{ car } h_{fm} = h$$

$$2(p_h + 1) = k_h$$

$$p_h = \frac{k_h}{2} - 1$$

Ainsi,

$$h_{fm} = h_{M'} - 1 = h \iff k_h \text{ pair et } p_h = \frac{k_h}{2} - 1 \tag{7}$$

$$w_{fm} = w + 1 + 2p_w - k_w + 1$$

$$2p_w - k_w + 2 = 0 \text{ car } w_{fm} = w$$

$$2(p_w + 1) = k_w$$

$$p_w = \frac{k_w}{2} - 1$$

Ainsi,

$$w_{fm} = w_{M'} - 1 = w \iff k_w \text{ pair et } p_w = \frac{k_w}{2} - 1 \tag{8}$$

2.4 Full Padding et Stride unitaire

Le **full padding** est une technique où chaque élément du filtre puisse se superposer entièrement à chaque élément de la matrice d'entrée, y compris les bords. Ainsi, nous devons ajouter $p_h = k_h - 1$ zéros et $p_w = k_w - 1$ zéros afin que le filtre puisse accomplir cela.

$\forall\ w, k_h, p_h$ tel que $k_h \leq h$

$$w_{fm} = w + k_w - 1 \tag{9}$$

$\forall\ h, k_h, p_h$ tel que $k_h \leq h$

$$h_{fm} = h + k_h - 1 \tag{10}$$

2.5 Stride non Unitaire et Absence de Zero-Padding

De manière analogue à 1.1, sauf que nous devons diviser le nombre de pas effectués par s_w ou s_h et prendre la partie entière de ce nombre dans le cas où il n'est pas entier. $\forall\ w, k_w$ tel que $k_w \leq w$

$$w_{fm} = \lfloor \frac{w - k_w}{s_w} \rfloor + 1 \tag{11}$$

Par la même logique, selon l'axe des ordonnées.

$\forall\ h, k_h$ tel que $k_h \leq h$

$$h_{fm} = \lfloor \frac{h - k_h}{s_h} \rfloor + 1 \tag{12}$$

2.6 Stride non Unitaire et Zero-Padding

De manière analogue à 2.2, nous obtenons : $\forall\ w, k_w, p_w$ tel que $k_w \leq w$

$$w_{fm} = \lfloor \frac{w + 2p_w - k_w}{s_w} \rfloor + 1 \tag{13}$$

$\forall\ h, k_h, p_h$ tel que $k_h \leq h$

$$h_{fm} = \lfloor \frac{h + 2p_h - k_h}{s_h} \rfloor + 1 \tag{14}$$

2.7 Convolution 2D généralisée

Nous pouvons réécrire la définition de la convolution 2D de l'image M par le filtre F est une matrice C $\in \mathbb{R}^{n \times p}$ avec n,p $\in \mathbb{N}^*$:

$\forall (i,j) \in [|1,n|] \times [|1,p|]$

$$C_{i,j} = \sum_{k=1}^{k_h} \sum_{l=1}^{k_w} \sum_{m=1}^{c} F_{k,l,m} \times M_{i \times s_h + k, j \times s_w + l, m}$$

2.8 D'autres techniques

Nous avons pu voir les caractéristiques principales des techniques utilisées pour la convolution 2D en Deep Learning, mais bien sûr il en existe d'autres :

— **Dilation** : la dilatation est l'espace entre les élements du filtre, dans nos démonstation ce dernier est égale à 1.

— **Reflect Padding** : les bordures de l'image sont remplies en reflétant les pixels adjacents.

— **Replicate Padding** : les bordures de l'image sont remplies en répétant les pixels les plus proches des bords de l'image.

— **Circular Padding** : les bordures de l'image sont remplies en répétant les pixels de manière circulaire. Cela signifie que les pixels au début de l'image sont utilisés pour remplir les bordures de la fin de l'image, et vice versa.

3 Pooling 2D

3.1 Définition et Exemples

L'idée de pooling a également été développée et utilisée dans les premiers réseaux de LeCun, tels que LeNet-5. Cette méthode permet de réduire la dimensionnalité tout en conservant les informations importantes. Il existe plusieurs versions de pooling, nous en verrons que deux **Max-Pooling** et **Average-Pooling**.

Le **Max-Pooling 2D** de l'image M par le filtre $F = H_{k_h,k_w}$ (constitué que de 1) est une matrice C $\in \mathbb{R}^{n \times p}$ avec n,p $\in \mathbb{N}^*$:

$\forall (i,j) \in [|1,n|] \times [|1,p|]$

$$C_{i,j} = \max_{k=1,l=1,m=1}^{k_h,k_w,c} M_{i \times s_h + k, j \times s_w + l, m}$$

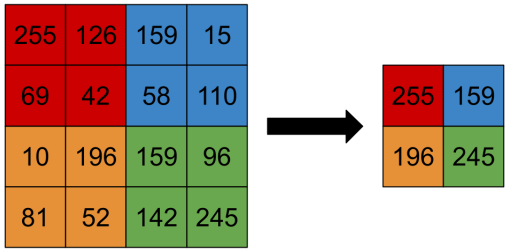

FIGURE 5 – Exemple de Fractional Max-Pooling

FIGURE 6 – Exemple de Max-Pooling 2D avec $s = 8$, $p = 0$ et un filtre $H_{7,7}$

Le **Average Pooling** de l'image M par le filtre $F = H_{k_h,k_w}$ est une matrice C $\in \mathbb{R}^{n\times p\times c}$ avec n,p,c $\in \mathbb{N}^*$:

$\forall (i,j) \in [|1,n|] \times [|1,p|]$

$$C_{i,j} = \frac{1}{k_h \times k_w} \sum_{k=1}^{k_h} \sum_{l=1}^{k_w} M_{i\times s_h+k, j\times s_w+l}$$

FIGURE 7 – Exemple de Average Pooling 2D avec $s = 8$, $p = 0$ et un filtre $H_{7,7}$

3.2 Arithmétique Pooling 2D

De manière analogue à 2.2 et 2.6, nous obtenons : $\forall\ w, k_w, p_w$ tel que $k_w \leq w$

$$w_{fm} = \lfloor \frac{w + 2p_w - k_w}{s_w} \rfloor + 1$$

$\forall\ h, k_h, p_h$ tel que $k_h \leq h$

$$h_{fm} = \lfloor \frac{h + 2p_h - k_h}{s_h} \rfloor + 1$$

4 Bloc convolutif

En cours!!!!!!!!!

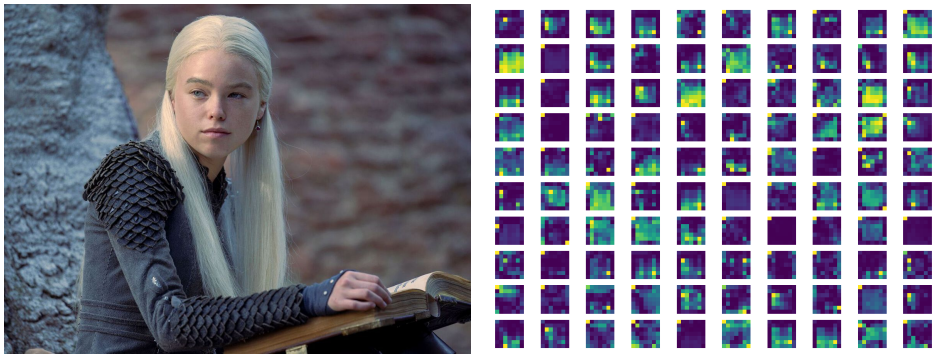

FIGURE 8 – Exemple d'une sortie partielle du backbone du modèle EfficientNetV2S sur notre image