

Exercice 1 : VGG16

Question 1. On considère un réseau de neurones convolutif qui prend en entrée des images couleur RGB de taille 224×224 pixels. Si la première couche de convolution a 16 filtres de dimension 3×3 , combien de poids au total comporte cette couche ?

Question 2.

- Pour réaliser une classification binaire, quelle fonction d'activation va-t-on utiliser pour l'unique neurone de la dernière couche d'un réseau ? Peut-on utiliser deux neurones en sortie pour faire la même tâche ?
- Même question pour une classification avec 10 classes mutuellement exclusives à distinguer.
- Toujours avec 10 classes différentes, comment peut-on coder/représenter en pratique ces 10 classes pour un réseau de neurones ? Comment s'appelle ce type d'encodage d'étiquettes ?
- Donner deux pré-traitements possibles sur des images données en entrée d'un réseau de neurones.

Question 3. On considère le réseau de classification d'objets dans les images appelé VGG16. Son architecture est illustrée dans la figure 1.

- Combien de classes d'objets ce réseau peut-il prédire ?
- Sachant qu'il permet de détecter des classes mutuellement exclusives, avec quelle fonction de coût peut-on entraîner ce modèle ?
- Quel est le rôle des couches « maxpooling » ? Détaillez leur fonctionnement.
- Ce réseau est constitué de deux blocs : le premier composé d'une succession de couches de convolutions, le deuxième de couches denses (MLP). Quel rôle a chacun de ces blocs ?

$$1) \text{ poids total : } (3 \times 3 \times 3) \times 16 + 16 = 448, \text{ le } +16 \text{ car biais}$$

2) - une sigmoïde. Oui, deux neurones en sortie pour la même tâche mais alors on ne prend pas sigmoïde mais softmax.

- softmax si 10 classes mutuellement exclusives à distinguer

- 10 classes, y va être encodé en "one-hot".

ex: $y = [0, 0, 1, 0, 0 \dots 0]$
 ^{classe positive (indice 2)}

- On peut normaliser entre 0 et 1 pour moindre variance
On peut standardiser : image $\leftarrow \frac{\text{image} - \text{moyenne}}{\text{écart-type}}$

3) - Le réseau peut prédire 1000 classes d'objets

- On peut entraîner ce modèle avec la fonction de coût "cross-entropy"

- Le rôle des couches "maxpooling" est de réduire la dimension spatiale (taille de l'image).
Le but est de traiter moins de données, temps de calcul ainsi que de réduire le bruit.
Ca augmente aussi le champs réceptif

- La succession de couches de convolutions + max pooling permet d'extraire des informations (extraction des features)

Puisque fully-connecting (couches denses) est responsable de la classification.

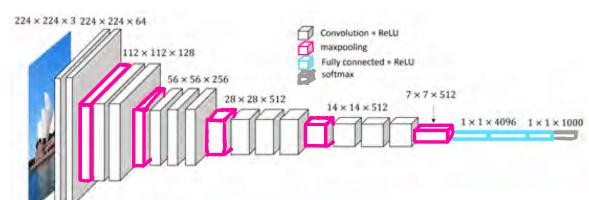


FIGURE 1 – Architecture de VGG16.

Exercice 2 : convolution 1-d

Question 1. Exemple

Soit le tenseur $x = [2, 3, 0, -1]$ et le kernel $w = [1, 2, -1]$

- Calculer le résultat de la convolution sur x avec le kernel w .
- Donner la matrice W utilisant les poids de w qui permet de faire l'opération équivalente par une multiplication matricielle $y = Wx$.
- Quelle est la longueur du résultat de cette convolution ?

Question 2. Transposons

Considérons maintenant le résultat précédent y comme une entrée

- Calculer le résultat de la multiplication matricielle $z = W^T y$.
- Quelle est la longueur du résultat z ? Commenter.

$$1) \quad x = [2, 3, 0, -1] \text{ et } w = [1, 2, -1]$$

- convolution de x avec le kernel w : $s(t) = (x * w)(t)$

$$y = [2 \times 1 + 3 \times 2 + 0, 3 \times 1 + 0 + (-1)] = [8, 5]$$

$$S(i, j) = (K * I)(i, j) = \sum_{m, n} I(i + m, j + n)K(m, n)$$

$$- y = Wx \text{ avec } W = \begin{bmatrix} 1 & 2 & -1 & 0 \\ 0 & 1 & 2 & -1 \end{bmatrix} \begin{bmatrix} 2 \\ 3 \\ 0 \\ -1 \end{bmatrix} = \begin{bmatrix} 8 \\ 5 \end{bmatrix}$$

- La longueur du résultat de cette convolution est 2

$$2) \quad y = [8, 5]$$

$$- z = W^T y = \begin{bmatrix} 1 & 2 & -1 & 0 \\ 0 & 1 & 2 & -1 \end{bmatrix}^T \begin{bmatrix} 8 \\ 5 \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 2 & 1 \\ -1 & 2 \\ 0 & -1 \end{bmatrix} \begin{bmatrix} 8 \\ 5 \end{bmatrix} = \begin{bmatrix} 8 \\ 20 \\ 0 \\ -5 \end{bmatrix}$$

- La longueur de z est 4. Il y a eu une augmentation.

Il s'agit d'une convolution transposée.

↳ nn.ConvTransposeD

Exercice 3 : convolutions 2d standard, depthwise et pointwise

Notations

Les tenseurs sont aplatis pour pouvoir être lus facilement. Nous avons les canaux de sortie (C_{out}) de haut en bas et les canaux d'entrée de gauche à droite (C_{in}), et une ligne horizontale (resp. verticale) délimite les deux.

Question 1. Conv2d standard

Voici la définition du noyau (kernel) w de notre première convolution :

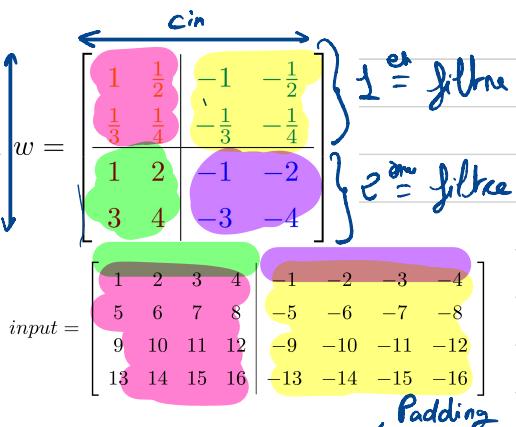
$w.size() = (C_{out}, C_{in}, k_h, k_w) = (2, 2, 2, 2)$

et ses valeurs :

Soit m une couche Conv2D telle que :

```
m = nn.Conv2d(Cin, Cout, (kh, kw), bias=False, stride=2)
```

- Quelles sont les dimensions du résultat de cette convolution appliquée à l'input?
 - Quel est le tenseur résultat $m(input, w)$?
 - Nous avons vu en cours que l'implantation de la convolution dans PyTorch est faite à l'aide d'une multiplication matricielle. Cet algorithme, appelé Im2Col, redimensionne à la fois le kernel et l'input pour obtenir le résultat désiré (un reshape final est nécessaire pour revenir à un tenseur 4-d en sortie).
 - Le batch d'inputs subit le redimensionnement suivant : $(B, C_{in}, h_{in}, w_{in})$ en $(C_{in} * k_h * k_w, B * H_{out} * W_{out})$
 - Et pour le kernel : $(C_{out}, C_{in}, k_h, k_w)$ en $(C_{out}, C_{in} * k_h * k_w)$
- Donner les deux matrices input et kernel redimensionnées par Im2Col pour notre exemple.



$$h_{out} = \frac{h_{in} + 2P - k_h}{\text{stride}} + 1$$

$$= \frac{4 + 0 - 2}{2} + 1 = 2$$

1) - dimension du résultat de cette convolution appliquée à l'input: $(B, Cout, h_{out}, w_{out}) = (1, 2, 2, 2)$

- tenseur résultat $m(input, w) = \begin{bmatrix} a & b \\ c & d \\ \hline e & f \\ g & h \end{bmatrix} \uparrow \downarrow \text{Cout}$

$a = 10, 33$

- Deux matrices input et kernel redimensionnées par Im2Col:

$$w = \begin{bmatrix} 1 & \frac{1}{2} & \frac{1}{3} & \frac{1}{4} \\ 1 & 2 & 3 & 4 \end{bmatrix} \quad \begin{bmatrix} -1 & -\frac{1}{2} & -\frac{1}{3} & -\frac{1}{4} \\ -1 & -2 & -3 & -4 \end{bmatrix}$$

$$x = \begin{bmatrix} 1 & 3 & 9 & 11 \\ 2 & 6 & 10 & 12 \\ 5 & 7 & 13 & 15 \\ 8 & 14 & 16 \\ 6 & -1 & -3 & -9 \\ -1 & -2 & -4 & -10 \\ -5 & -7 & -13 & -15 \\ -6 & -8 & -14 & -16 \end{bmatrix}$$

$$W \cdot x = \begin{bmatrix} 3 \frac{1}{3} & b & c & d \\ e & f & g & h \end{bmatrix}$$

Question 2. Conv2d séparable en profondeur ou Depthwise Conv2d

Pour pouvoir utiliser ce type de convolution, on utilise l'option de PyTorch $groups = C_{in}$. On a en outre besoin que C_{out} soit divisible par la valeur de $groups$ choisie. Dans notre exemple, nous aurions une possibilité :

$w.size() = (C_{out}, C_{in}/groups, k_h, k_w) = (2, 1, 2, 2)$

Considérons le noyau suivant :

$$w = \begin{bmatrix} 1 & \frac{1}{2} \\ \frac{1}{3} & \frac{1}{4} \\ 1 & 2 \\ 3 & 4 \end{bmatrix}$$

Soit m la couche depthwise Conv2D utilisant ce noyau :

```
m = nn.Conv2d(Cin, Cout, (kh, kw), groups=Cin, bias=False, stride=2)
```

- Quelles sont les dimensions et le tenseur résultat de $m(input, w)$?

2) dimensions: $(1, 2, 2, 2)$

tenseur résultat: $m(input, w) =$