Réseaux de neurones à convolutions

Objectifs

- Appliquer les réseaux de neurones aux images
- Comprendre le mécanisme des convolutions
- Connaître les couches liées aux convolutions

Limitations des réseaux denses pour les images

Principale faiblesse : chaque input est spécifique pour les poids appris.

$\rightarrow$ Impossible de gérer efficacement les translations

$\rightarrow$ Impossible d'appliquer un neurone détectant une forme à une autre position dans l'image

On pallie ces problèmes avec les réseaux à convolutions.

Réseaux à convolutions

Idées fortes :

- Restreindre l'input des neurones
- Appliquer un même neurone à plusieurs zones restreintes

Neurone dans un réseau à convolutions

- Appelé filtre ou kernel
- Réceptionne souvent entre 1x1 et 7x7 inputs seulement
- Appliqué plusieurs fois au lieu d'une seule
- Un même neurone a donc plusieurs outputs.

Application d'un filtre

Image :

1	2	0	5
3	1	0	-1
0	3	0	1
0	4	5	5

Filtre :

0	1
2	-1

Application successive du filtre sur les différentes zones de l'image

Application d'un filtre

Image :

1	2	0	5
3	1	0	-1
0	3	0	1
0	4	5	5

Filtre :

0	1
2	-1

Résultat :

7		

Application successive du filtre sur les différentes zones de l'image

Application d'un filtre

Image :

1	2	0	5
3	1	0	-1
0	3	0	1
0	4	5	5

Filtre :

0	1
2	-1

Résultat :

7	2	

Application successive du filtre sur les différentes zones de l'image

Application d'un filtre

Image :

1	2	0	5
3	1	0	-1
0	3	0	1
0	4	5	5

Filtre :

0	1
2	-1

Résultat :

7	2	6

Application successive du filtre sur les différentes zones de l'image

Application d'un filtre

Image :

1	2	0	5
3	1	0	-1
0	3	0	1
0	4	5	5

Filtre :

0	1
2	-1

Résultat :

7	2	6
-2		

Application successive du filtre sur les différentes zones de l'image

Application d'un filtre

Image :

1	2	0	5
3	1	0	-1
0	3	0	1
0	4	5	5

Filtre :

0	1
2	-1

Résultat :

7	2	6
-2	6	

Application successive du filtre sur les différentes zones de l'image

Application d'un filtre

Image :

1	2	0	5
3	1	0	-1
0	3	0	1
0	4	5	5

Filtre :

0	1
2	-1

Résultat :

7	2	6
-2	6	-2

Application successive du filtre sur les différentes zones de l'image

Application d'un filtre

Image :

1	2	0	5
3	1	0	-1
0	3	0	1
0	4	5	5

Filtre :

0	1
2	-1

Résultat :

7	2	6
-2	6	-2
-1		

Application successive du filtre sur les différentes zones de l'image

Application d'un filtre

Image :

1	2	0	5
3	1	0	-1
0	3	0	1
0	4	5	5

Filtre :

0	1
2	-1

Résultat :

7	2	6
-2	6	-2
-1	3	

Application successive du filtre sur les différentes zones de l'image

Application d'un filtre

Image :

1	2	0	5
3	1	0	-1
0	3	0	1
0	4	5	5

Filtre :

0	1
2	-1

Résultat :

7	2	6
-2	6	-2
-1	3	6

Application successive du filtre sur les différentes zones de l'image

Exercice

Image :

3	2	-2	-3
1	4	1	2
5	-3	-2	-1
-2	6	7	1

Filtre :

2	-1
-2	3

Biais = 5

1. Calculer le résultat de l'application du filtre à l'image.
2. Que détecte le filtre ?

Solution

Image :

3	2	-2	-3
1	4	1	2
5	-3	-2	-1
-2	6	7	1

Filtre :

2	-1
-2	3

Biais = 5

1. Calculer le résultat de l'application du filtre à l'image.

19	6	8
-16	12	6
40	10	-9

2. Que détecte le filtre ?

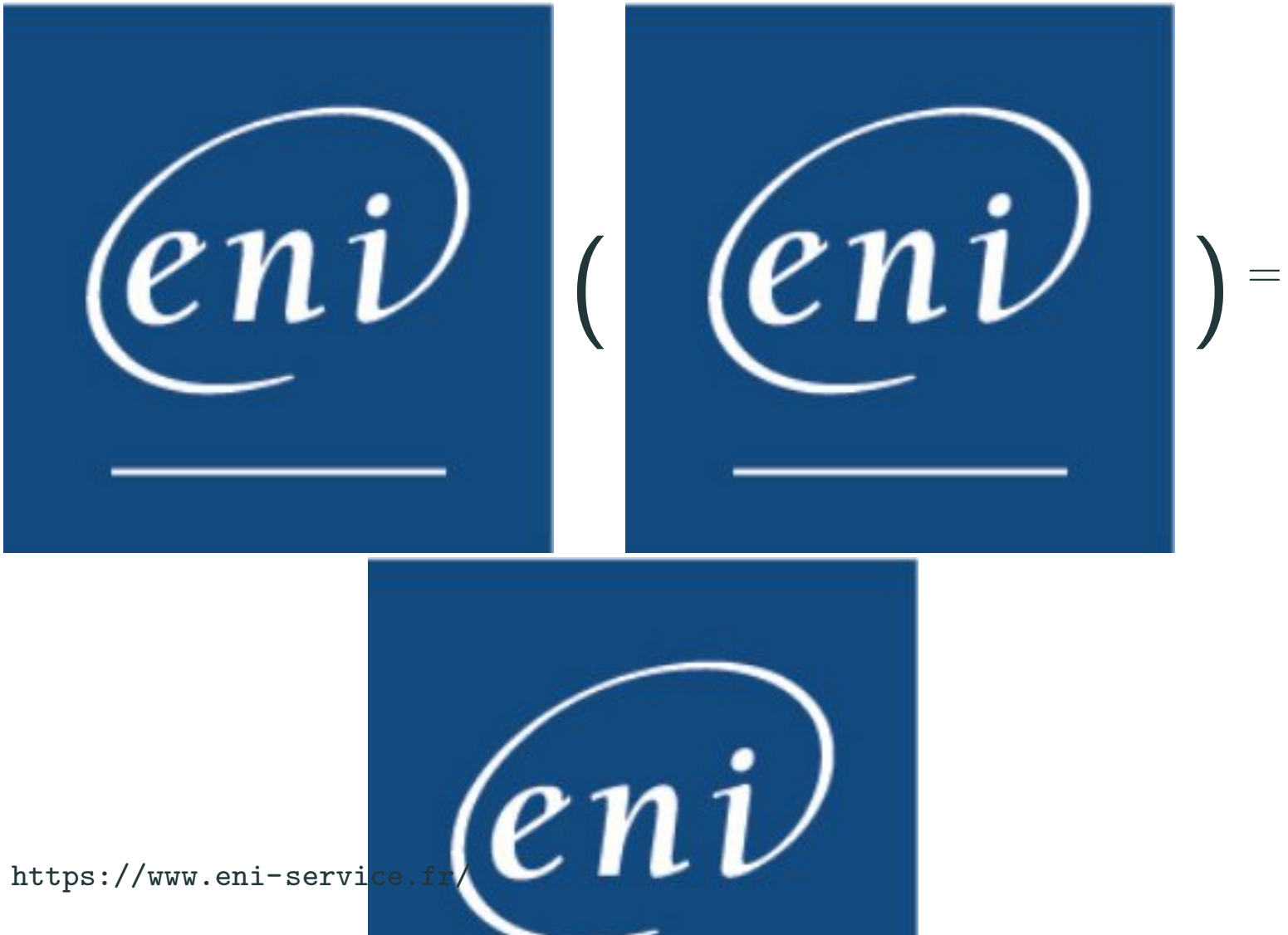
   L'activation de la diagonale Nord-Ouest Sud-Est et la non activation de la diagonale opposée.

Nature des transformations apprises

Un réseau à convolutions transforme un input 3d en un output 3d de forme différente.

Output d'un filtre

(conv-filter) (conv-input) (conv-feature-map)



Couche complète

Une couche complète de convolution contient plusieurs filtres :
(conv-filter) (conv-input) (conv-feature-map)

Réseau complet typique

Entrelacement de couches de convolutions et de non-linéarités :
(conv-filter) (conv-input) (conv-feature-map) (conv-filter2)
(conv-feature-map) (conv-feature-map2)

Première couche : ReLU $\Big($  $\Big($  $\Big)\Big) =$

Deuxième couche : ReLU $\Big($  $\Big($  $\Big)\Big) =$

Exercice

Input	$128 \times 128 \times 3$
Couche 1	256 filtres de $5 \times 5 \times ?$
Couche 2	512 filtres de $3 \times 3 \times ?$
Couche 3	64 filtres de $1 \times 1 \times ?$

1. Compléter les ?
2. Donner la taille de l'output.
3. Combien de poids contient la dernière couche ?

Solution

Input	$128 \times 128 \times 3$
Couche 1	256 filtres de $5 \times 5 \times 3$
Couche 2	512 filtres de $3 \times 3 \times 256$
Couche 3	64 filtres de $1 \times 1 \times 512$

1. Compléter les ?.
2. Donner la taille de l'output.
   $128 \times 128 \times 3$
   $\rightarrow 124 \times 124 \times 256$
   $\rightarrow 122 \times 122 \times 512$
   $\rightarrow 122 \times 122 \times 64$
3. Combien de poids contient la dernière couche ?
   $1 \times 1 \times 512 \times 64 + 64 = 32832$

Plus de détails sur l'application des filtres

Possibilité de modifier l'application des filtres pour construire des architectures diverses :

- Stride n : Parcourir l'input n par n au lieu de 1 par 1. Permet de sous-échantillonner.
- Padding n : Rajouter des 0 à l'input pour contrôler les dimensions de l'output.

Par exemple, des filtres 5×5 avec padding 2 conservent la taille de l'input.

Exemple de padding

Taille 3×3, Stride 1, padding 1 sur un input 3×3 :

0	0	0	0	0
0	1	2	8	0
0	-7	-1	-5	0
0	3	1	1	0
0	0	0	0	0

Exemple de padding

Taille 3×3, Stride 1, padding 1 sur un input 3×3 :

0	0	0	0	0
0	1	2	8	0
0	-7	-1	-5	0
0	3	1	1	0
0	0	0	0	0

Exemple de padding

Taille 3×3, Stride 1, padding 1 sur un input 3×3 :

0	0	0	0	0
0	1	2	8	0
0	-7	-1	-5	0
0	3	1	1	0
0	0	0	0	0

Exemple de padding

Taille 3×3, Stride 1, padding 1 sur un input 3×3 :

0	0	0	0	0
0	1	2	8	0
0	-7	-1	-5	0
0	3	1	1	0
0	0	0	0	0

Exemple de padding

Taille 3×3, Stride 1, padding 1 sur un input 3×3 :

0	0	0	0	0
0	1	2	8	0
0	-7	-1	-5	0
0	3	1	1	0
0	0	0	0	0

Exemple de padding

Taille 3×3, Stride 1, padding 1 sur un input 3×3 :

0	0	0	0	0
0	1	2	8	0
0	-7	-1	-5	0
0	3	1	1	0
0	0	0	0	0

Exemple de padding

Taille 3×3, Stride 1, padding 1 sur un input 3×3 :

0	0	0	0	0
0	1	2	8	0
0	-7	-1	-5	0
0	3	1	1	0
0	0	0	0	0

Exemple de padding

Taille 3×3, Stride 1, padding 1 sur un input 3×3 :

0	0	0	0	0
0	1	2	8	0
0	-7	-1	-5	0
0	3	1	1	0
0	0	0	0	0

Exemple de padding

Taille 3×3, Stride 1, padding 1 sur un input 3×3 :

0	0	0	0	0
0	1	2	8	0
0	-7	-1	-5	0
0	3	1	1	0
0	0	0	0	0

Exemple de padding

Taille 3×3, Stride 1, padding 1 sur un input 3×3 :

0	0	0	0	0
0	1	2	8	0
0	-7	-1	-5	0
0	3	1	1	0
0	0	0	0	0

$\rightarrow$ L'output conserve la dimension de l'input (3×3) malgré la taille du filtre > 1 !

Exemple de stride

Taille 2×2, Stride 2, padding 0 sur un input 4×4 :

5	1	2	8
-1	-7	-1	-5
2	3	1	1
3	0	1	2

Exemple de stride

Taille 2×2, Stride 2, padding 0 sur un input 4×4 :

5	1	2	8
-1	-7	-1	-5
2	3	1	1
3	0	1	2

Exemple de stride

Taille 2×2, Stride 2, padding 0 sur un input 4×4 :

5	1	2	8
-1	-7	-1	-5
2	3	1	1
3	0	1	2

Exemple de stride

Taille 2×2, Stride 2, padding 0 sur un input 4×4 :

5	1	2	8
-1	-7	-1	-5
2	3	1	1
3	0	1	2

Deux autres blocs utilisés

En plus des convolutions et non-linéarités, on utilise :

- Bloc de pooling pour sous-échantillonner
- Bloc de normalisation de batch pour améliorer l'apprentissage

Bloc de pooling

- Réduit la dimensionnalité du volume en aggrégeant les valeurs
- Utilise une moyenne ou un maximum
- Pas de paramètres ! Fonction fixe
- À contraster avec les convolutions à stride > 1.

Bloc de pooling

Pooling max de taille 2×2, Stride 2, padding 0 sur un input 4×4 :

5	1	2	8
-1	-7	-1	-5
2	3	1	1
3	0	1	2

Résultat :

5	8
3	2

Bloc de pooling

Pooling max de taille 2×2, Stride 2, padding 0 sur un input 4×4 :

5	1	2	8
-1	-7	-1	-5
2	3	1	1
3	0	1	2

Résultat :

5	8
3	2

Bloc de pooling

Pooling max de taille 2×2, Stride 2, padding 0 sur un input 4×4 :

5	1	2	8
-1	-7	-1	-5
2	3	1	1
3	0	1	2

Résultat :

5	8
3	2

Bloc de pooling

Pooling max de taille 2×2, Stride 2, padding 0 sur un input 4×4 :

5	1	2	8
-1	-7	-1	-5
2	3	1	1
3	0	1	2

Résultat :

5	8
3	2

Paramètres de pooling

- 2×2, stride 2, padding 0
- 4×4, stride 2, padding 1

Bloc de normalisation de batch

- Lutter contre un effet pervers de la rétropropagation des gradients [**?**]
- Facilite grandement l'apprentissage
- Utilisé dans tous les modèles récents

Bloc de normalisation de batch

- Après un pas de rétropropagation, un layer n change de distribution d'output
- Le layer $n+1$ a appris sur la distribution d'avant

$\rightarrow$ L'entraînement est donc inefficace !

Bloc de normalisation de batch

Solution : normaliser le batch après le layer n pour qu'il soit stable pour le layer $n+1$: diviser par sa variance et soustraire sa moyenne.

Conclusion

- Les convolutions sont des neurones à l'input restreint
- Un réseau à convolutions transforme des tenseurs 3d en d'autres tenseurs 3d
- Le nombre, la taille et l'application des filtres contrôle ces transformations
- Des non-linéarités, des blocs de poolings et de la normalisation par batch sont utilisés en complément des convolutions