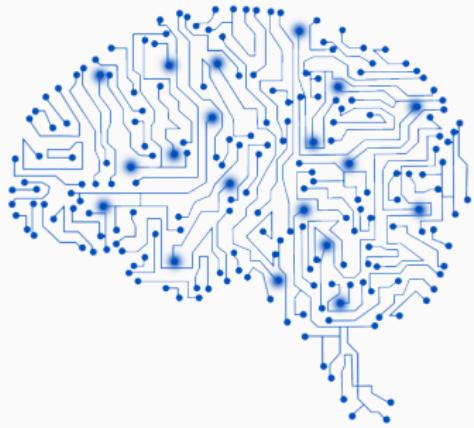


Apprentissage pour l'image Machine learning for image processing

Course III – Introduction to Artificial Neural Networks: Deep neural networks and Convolutional Neural Networks (CNN)

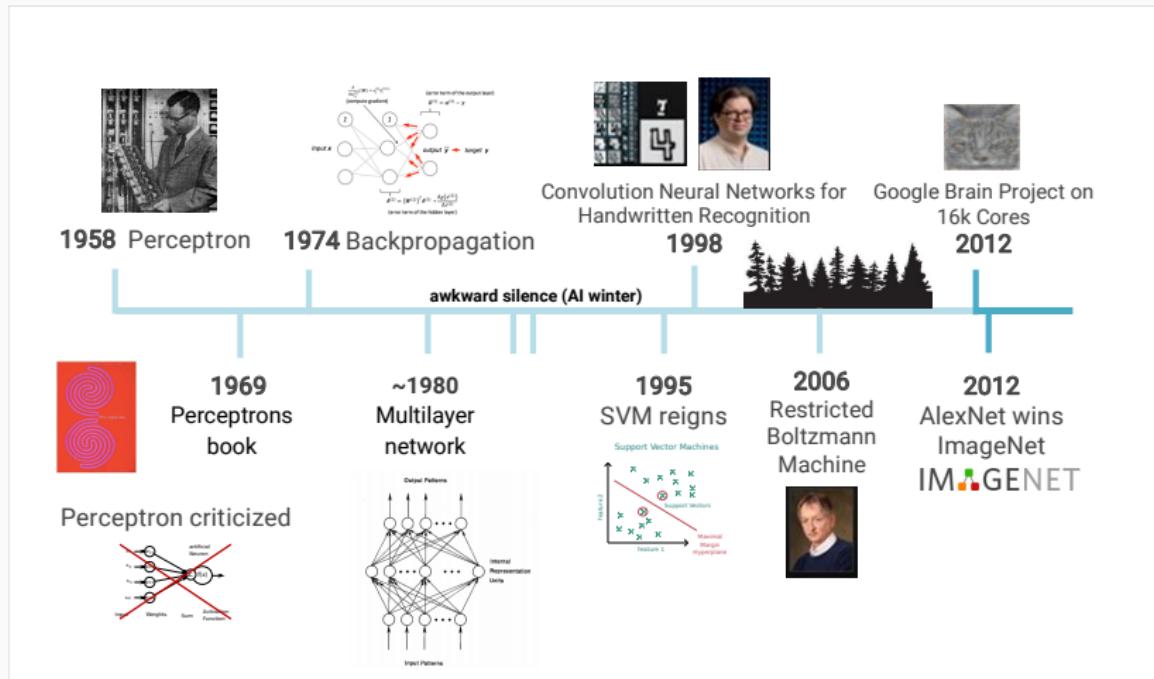
Emile Pierret

Lundi 7 avril 2025



Machine learning – Timeline

Timeline of (deep) learning



(Source: Lucas Masuch & Vincent Lepetit)

À la fin du cours :

- Comprendre ce qu'est un CNN (Réseau de Neurones Convolutifs).
- Implémenter l'entraînement d'un CNN pour la classification avec PyTorch.

Dernière séance :

- Comprendre le mécanisme de calcul des gradients dans un Réseau de Neurones (rétropropagation/backpropagation).
- Implémenter des tenseurs et la backpropagation avec PyTorch.
- Implémenter une descente de gradient avec PyTorch.

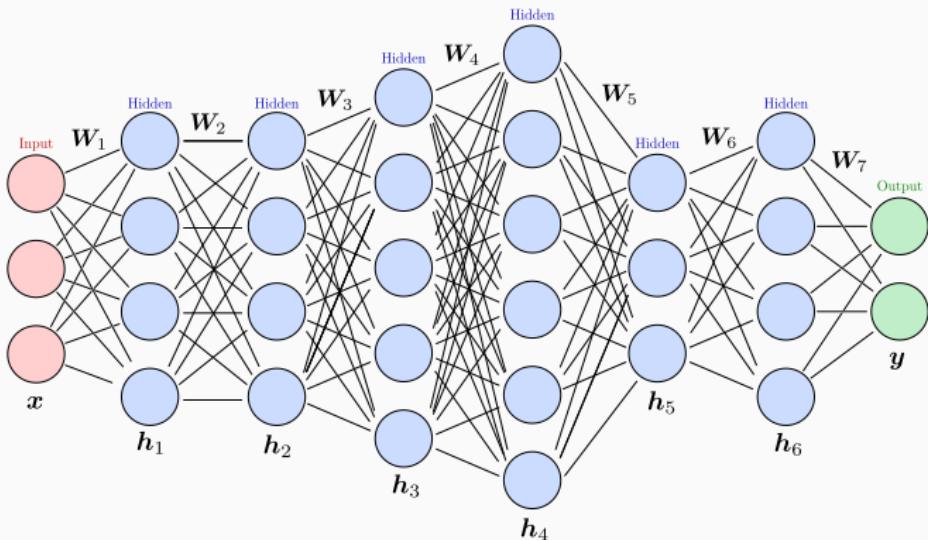
Pendant cette séance :

- Comprendre ce qu'est un CNN.
- Savoir coder un CNN avec PyTorch.

Prochaine séance :

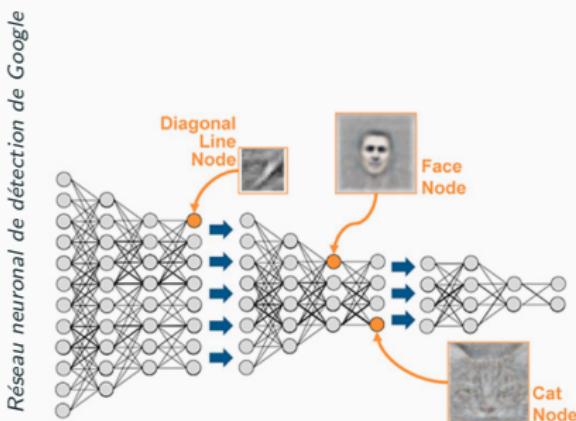
- Entraîner un CNN pour classifier les images de CIFAR-10.

ANNs recap



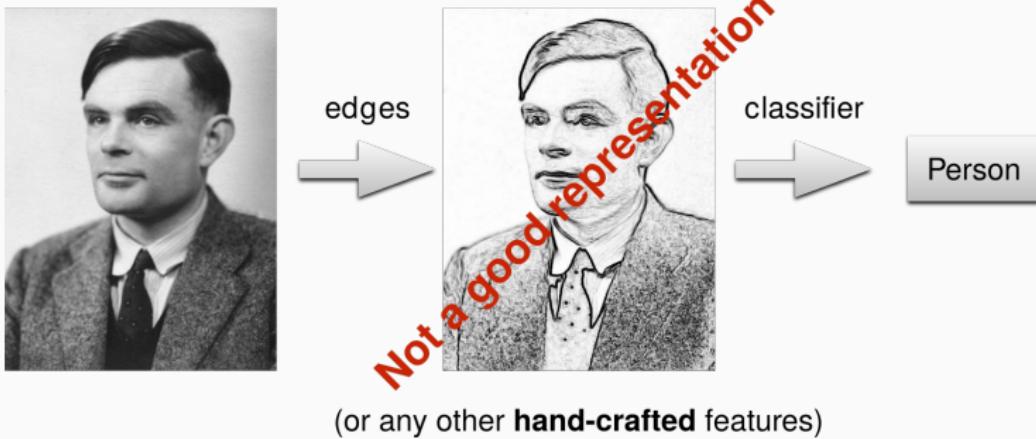
- Interconnexions des neurones (sommations et activations),
- Architecture feedforward : entrée → couche(s) cachée(s) → sortie,
- Paramétré par un ensemble de poids W_k et biais b_k ,
- Backpropagation : paramètres appris à partir d'un ensemble de données étiquetées.

- Apprentissage de représentations à l'aide de réseaux de neurones artificiels
 - Apprendre automatiquement de bonnes "features" à partir de données brutes.
- Apprentissage de représentations de données avec plusieurs niveaux d'abstraction
 - Hiérarchie de couches imitant les réseaux neuronaux de notre cerveau,



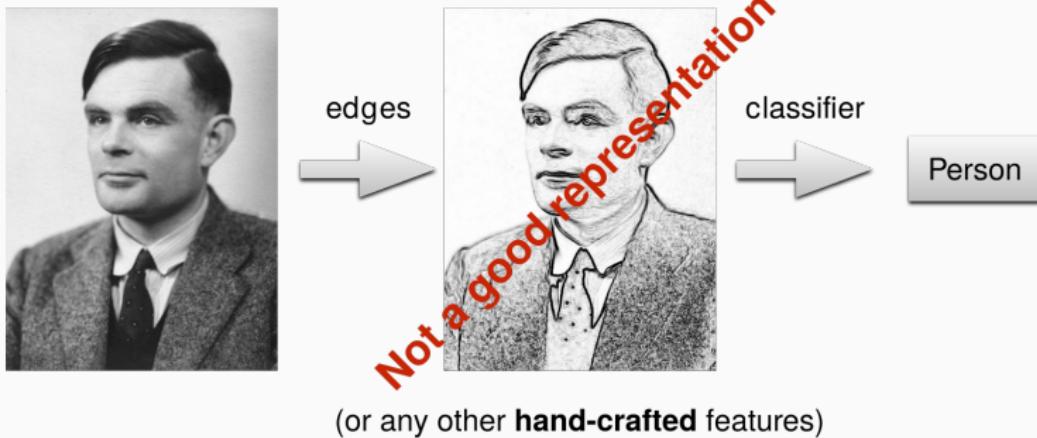
- Si vous fournissez au système une grande quantité d'informations, il commence à les comprendre et à y répondre de manière utile.

Comment enseigner à une machine ?



De bonnes représentations sont souvent très complexes à définir.

Comment enseigner à une machine ?

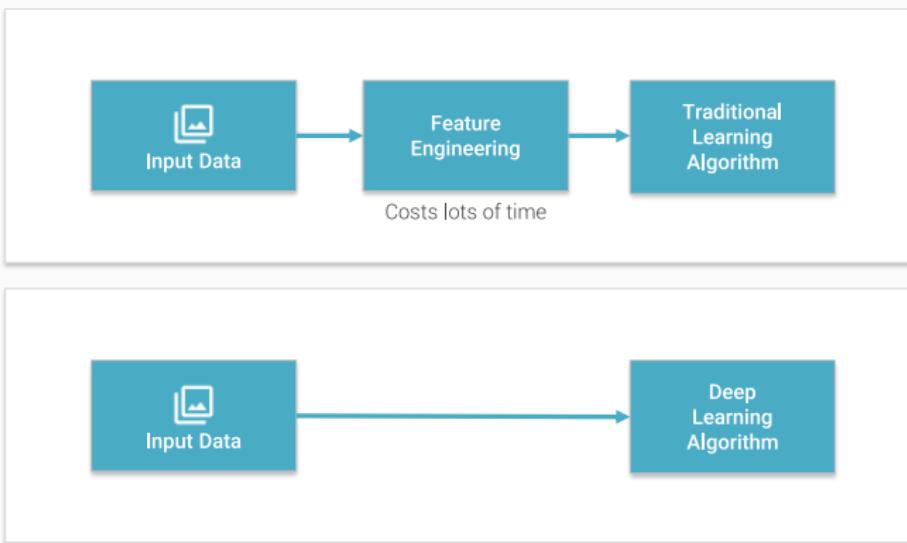


De bonnes représentations sont souvent très complexes à définir.

NB: Pour nos ensembles 2D (TP1), c'était assez facile.

(Source : Caner Hazırbaş)

Deep learning : Plus besoin d'ingénierie des caractéristiques



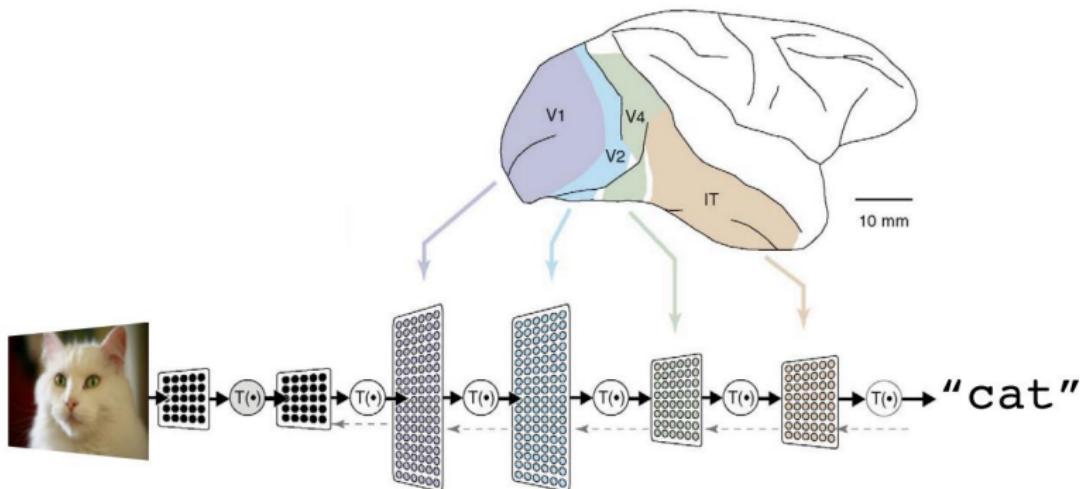
(Source : Lucas Masuch)

Inspiré par le cerveau

- La première hiérarchie de neurones qui reçoit l'information dans le cortex visuel est sensible à des bords spécifiques, tandis que les régions du cerveau plus bas dans la pipeline visuelle sont sensibles à des structures plus complexes, comme les visages.
- Notre cerveau possède de nombreux neurones connectés entre eux, et la **force des connexions** entre les neurones représente **la connaissance à long terme**.

(Source : Lucas Masuch)

Deep learning – Basic architecture



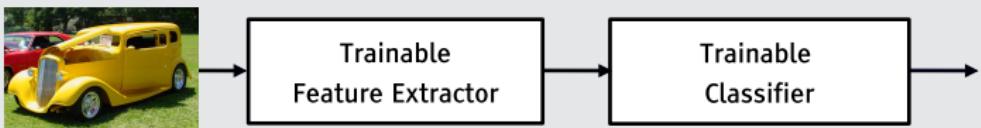
A deep neural network consists of a **hierarchy of layers**, whereby each layer **transforms the input data** into more abstract representations (e.g. edge -> nose -> face). The output layer combines those features to make predictions.

Apprentissage profond : Plus besoin d'ingénierie des caractéristiques

- Le modèle traditionnel de reconnaissance des motifs (Cf TP1)
 - Caractéristiques/Features fixes/ingénierie + classificateur entraînable



- Apprentissage profond
 - Caractéristiques/Features entraînable + classificateur entraînable



(Source : Yann LeCun & Marc'Aurelio Ranzato)

Hiérarchie des caractéristiques entraînables

- Hiérarchie des représentations avec des niveaux croissants d'abstraction.
- Chaque étape est une sorte de transformation de caractéristiques entraînables.

Reconnaissance d'images

- Pixel → bord → motif → partie → objet

Texte

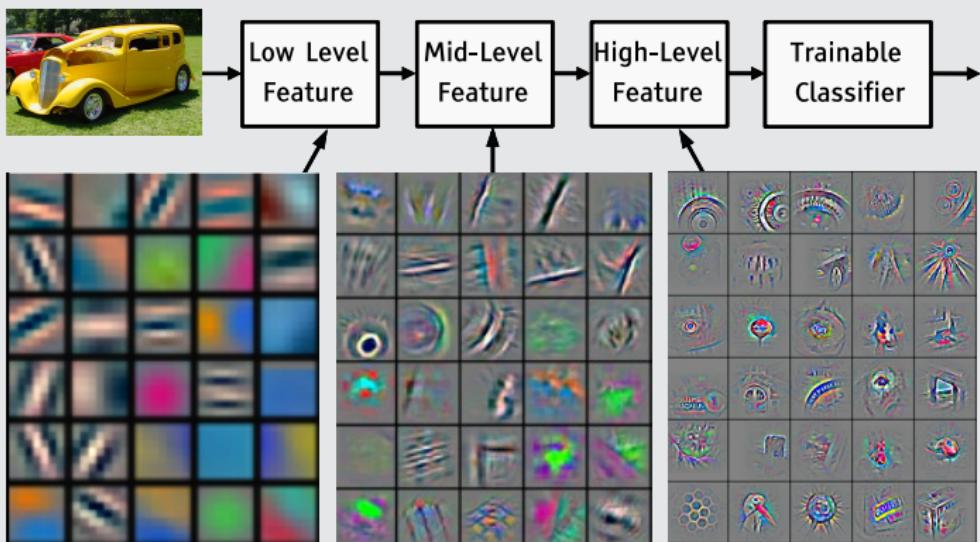
- Caractère → mot → groupe de mots → proposition → phrase → histoire

L'Apprentissage profond résout le problème de l'apprentissage de représentations hiérarchiques.

(Source : Yann LeCun & Marc'Aurelio Ranzato)

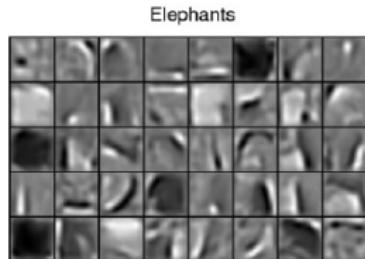
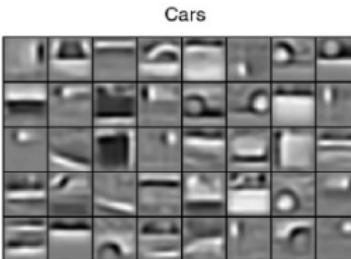
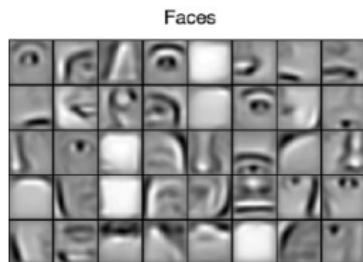
Apprentissage profond – Hiérarchie des caractéristiques

- L'apprentissage est **profond** s'il possède plus d'une étape de transformation non-linéaire des caractéristiques



Visualisation des caractéristiques du réseau convolutionnel entraîné sur ImageNet (Zeiler & Fergus, 2013)

Apprentissage profond – Hiérarchie des caractéristiques



- Les caractéristiques de niveau intermédiaire à élevé sont spécifiques à la tâche donnée.
- Les représentations de bas niveau contiennent des caractéristiques moins spécifiques.
(peuvent être partagées pour de nombreuses applications différentes)

Apprentissage profond – Entraînement

La tendance actuelle : rendre le modèle de plus en plus profond

- 2012 : 8 couches (AlexNet – Krizhevsky *et al.*, 2012)
- 2014 : 19 couches (VGG Net – Simonyan & Zisserman, 2014)
- 2014 : 22 couches (GoogLeNet – Szegedy *et al.*, 2014)
- 2015 : 152 couches (ResNet – He *et al.*, 2015)
- 2016 : 201 couches (DenseNet – Huang *et al.*, 2017)

Mais souvenez-vous, avec la rétropropagation :

- Nous nous retrouvons bloqués dans des optima locaux ou des points selles
- Le temps d'apprentissage ne se comporte pas bien en fonction de la taille
 - il est très lent pour les réseaux profonds et peut être instable.

Comment les réseaux sont-ils devenus aussi profonds ? D'abord, pourquoi la rétropropagation échoue-t-elle ?

Apprentissage profond – Problèmes de l'annulation du gradient (vanishing gradient)

Backpropagation et problèmes de l'annulation du gradient

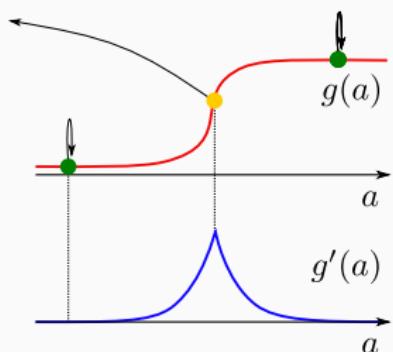
Mise à jour: $\mathbf{W}_k = \mathbf{W}_k - \gamma \nabla_{\mathbf{W}_k} E$ avec $\nabla_{\mathbf{W}_k} E = \boldsymbol{\delta}_k \mathbf{h}_{k-1}^T$
avec $\boldsymbol{\delta}_k = \nabla_{\mathbf{a}_k} E = \nabla_{\mathbf{h}_k} E \odot g'_k(\mathbf{a}_k)$.

- Avec des réseaux profonds, le gradient s'annule rapidement.
- Malheureusement, cela se produit même si nous sommes loin de la solution.
- Les mises à jour deviennent insignifiantes, ce qui entraîne des entraînements lents.
- Le gradient peut également exploser, entraînant des instabilités :
→ problème d'explosion du gradient.

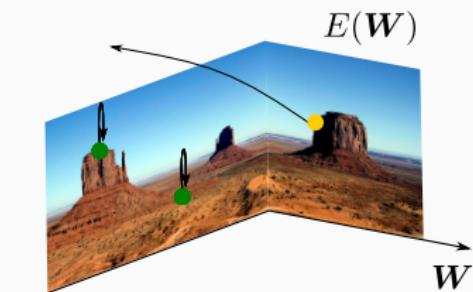
Apprentissage profond – Problème d'annulation du gradient

À mesure que le réseau devient plus profond, le paysage de E devient :

- très accidenté → beaucoup de points stationnaires,
- avec de larges plateaux → problème de disparition du gradient,
- et délimité par des falaises → problème d'explosion du gradient.



Activation function



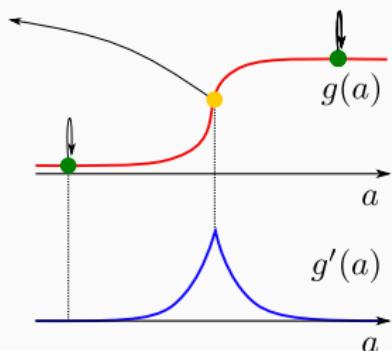
Cost landscape

Alors, comment s'en sortir ?

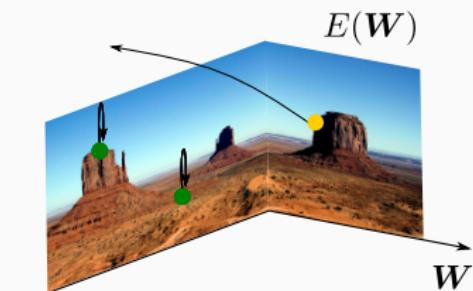
Apprentissage profond – Problème d'annulation du gradient

À mesure que le réseau devient plus profond, le paysage de E devient :

- très accidenté → beaucoup de points stationnaires,
- avec de larges plateaux → problème de disparition du gradient,
- et délimité par des falaises → problème d'explosion du gradient.



Activation function

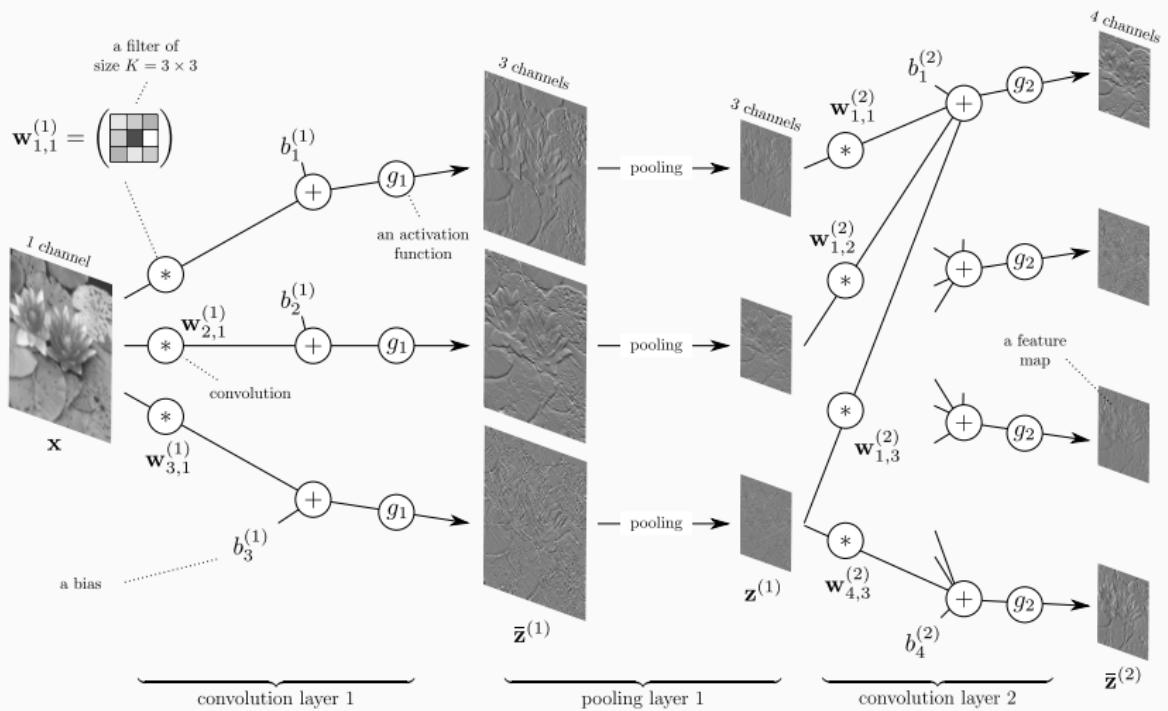


Cost landscape

Alors, comment s'en sortir ?

→ Il faut réduire le nombre de paramètres ! (ou plutôt leur taille)

CNN for image processing



Objectif: Réduire le nombre de paramètres

Afin de faciliter l'apprentissage/l'optimisation d'un réseau, on introduit deux types de couches:

- Des couches de convolution: opérations linéaires impliquant de plus petites matrices.
- Des couches de pooling: visant à réduire la taille des neurones.

Convolution : Un seul canal

Pour une image $x = x(i, j)$ ayant **un seul canal**, la convolution avec un noyau κ de taille $[-s, s] \times [-s, s]$

- la **convolution** est :

$$x * \kappa(i, j) = \sum_{(k, \ell) \in [-s, s] \times [-s, s]} \kappa(k, \ell) x(i - k, j - \ell)$$

$x * \kappa$ = moyenne locale de x avec le poids de κ en **position opposée**

- la **corrélation croisée** est :

$$x \otimes \kappa(i, j) = \sum_{(k, \ell) \in [-s, s] \times [-s, s]} \kappa(k, \ell) x(i + k, j + \ell).$$

$x \otimes \kappa$ = moyenne locale de x avec le poids de κ en **même position**

Une convolution est une somme pondérée des données, une "moyenne" locale.
(Voir illustration plus loin)

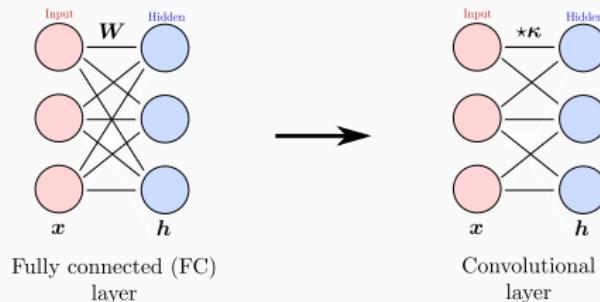
Convolution

En apprentissage automatique
“convolution”
signifie
“corrélation croisée + biais”

- Rappelons que les couches “linéaires” sont des applications affines $x \mapsto Wx + b$, donc les couches de convolution sont un cas spécifique où $x \mapsto Wx$ est une corrélation croisée.

Qu'est-ce que les CNN ?

- Essentiellement des réseaux neuronaux qui utilisent la convolution à la place des multiplications matricielles générales, au moins pour les premières couches.



- Les CNN sont conçus pour traiter des données sous forme de tableaux/tensors multidimensionnels (*par exemple*, images 2D, vidéos/images volumétriques 3D).
- Composés de séries d'étapes : couches convolutionnelles et couches de pooling.
- Les unités sont connectées à des régions locales des cartes de caractéristiques de la couche précédente.
- Miment le cortex visuel.

Convolutions

1 <small>x1</small>	1 <small>x0</small>	1 <small>x1</small>	0	0
0 <small>x0</small>	1 <small>x1</small>	1 <small>x0</small>	1	0
0 <small>x1</small>	0 <small>x0</small>	1 <small>x1</small>	1	1
0	0	1	1	0
0	1	1	0	0

Image

4		

Convolved
Feature

+bias

Convolutions

1	1 _{x1}	1 _{x0}	0 _{x1}	0
0	1 _{x0}	1 _{x1}	1 _{x0}	0
0	0 _{x1}	1 _{x0}	1 _{x1}	1
0	0	1	1	0
0	1	1	0	0

Image

4	3	

Convolved
Feature

+bias

Convolutions

1	1	1	0	0
0	1	1	1	0
0	0	1	1	1
0	0	1	1	0
0	1	1	0	0

Image

4	3	4

Convolved
Feature

+bias

Convolutions

1	1	1	0	0
0 x1	1 x0	1 x1	1	0
0 x0	0 x1	1 x0	1	1
0 x1	0 x0	1 x1	1	0
0	1	1	0	0

Image

4	3	4
2		

Convolved
Feature

+bias

Convolutions

1	1	1	0	0
0	1 _{x1}	1 _{x0}	1 _{x1}	0
0	0 _{x0}	1 _{x1}	1 _{x0}	1
0	0 _{x1}	1 _{x0}	1 _{x1}	0
0	1	1	0	0

Image

4	3	4
2	4	

Convolved
Feature

+bias

Convolutions

1	1	1	0	0
0	1	1 _{×1}	1 _{×0}	0 _{×1}
0	0	1 _{×0}	1 _{×1}	1 _{×0}
0	0	1 _{×1}	1 _{×0}	0 _{×1}
0	1	1	0	0

Image

4	3	4
2	4	3

Convolved
Feature

+bias

Convolutions

1	1	1	0	0
0	1	1	1	0
0 x1	0 x0	1 x1	1	1
0 x0	0 x1	1 x0	1	0
0 x1	1 x0	1 x1	0	0

Image

4	3	4
2	4	3
2		

Convolved
Feature

+bias

Convolutions

1	1	1	0	0
0	1	1	1	0
0	0 <small>×1</small>	1 <small>×0</small>	1 <small>×1</small>	1
0	0 <small>×0</small>	1 <small>×1</small>	1 <small>×0</small>	0
0	1 <small>×1</small>	1 <small>×0</small>	0 <small>×1</small>	0

Image

4	3	4
2	4	3
2	3	

Convolved
Feature

+bias

Convolutions

1	1	1	0	0
0	1	1	1	0
0	0	1 _{×1}	1 _{×0}	1 _{×1}
0	0	1 _{×0}	1 _{×1}	0 _{×0}
0	1	1 _{×1}	0 _{×0}	0 _{×1}

Image

4	3	4
2	4	3
2	3	4

Convolved
Feature

+bias

Convolutions

1	1	1	0	0
0	1	1	1	0
0	0	1 _{×1}	1 _{×0}	1 _{×1}
0	0	1 _{×0}	1 _{×1}	0 _{×0}
0	1	1 _{×1}	0 _{×0}	0 _{×1}

Image

4	3	4
2	4	3
2	3	4

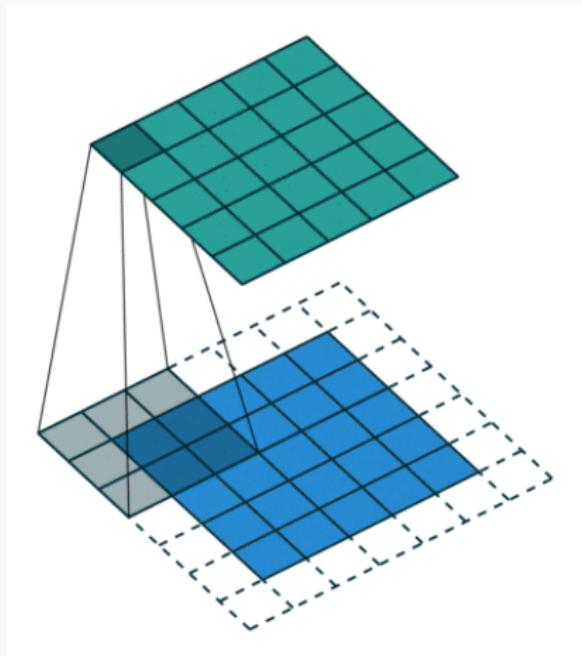
Convolved
Feature

+biais

→ Pour une taille d'entrée (M, N) , la taille de la sortie de la convolution par un noyau de taille $N_\kappa = 2k + 1$ est

$$(M - (N_\kappa - 1), N - (N_\kappa - 1)x) = (M - 2k, N - 2k)$$

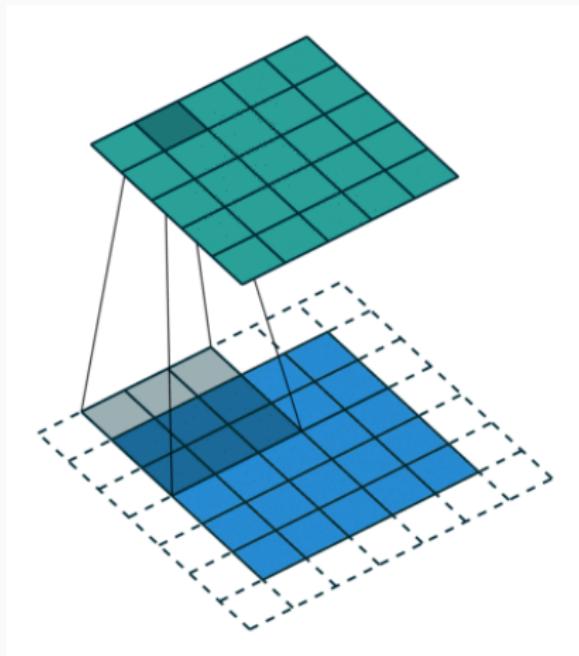
Convolutions with a padding



Différents types de padding existent :

- zéro
- bord
- réflexion

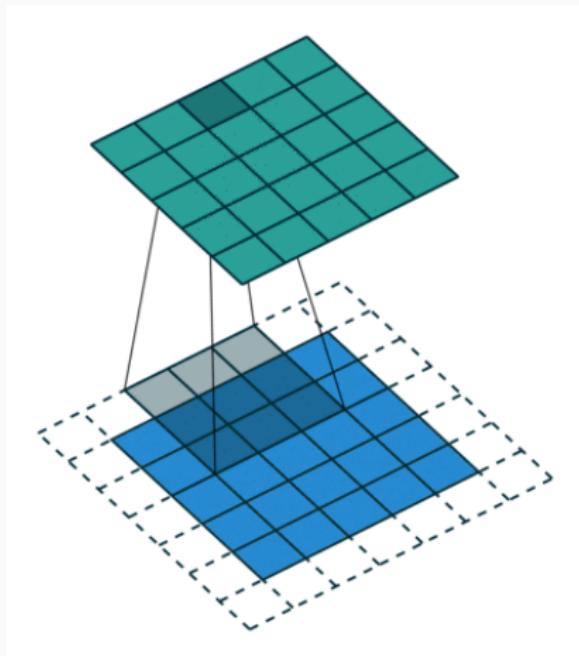
Convolutions with a padding



Différents types de padding existent :

- zéro
- bord
- réflexion

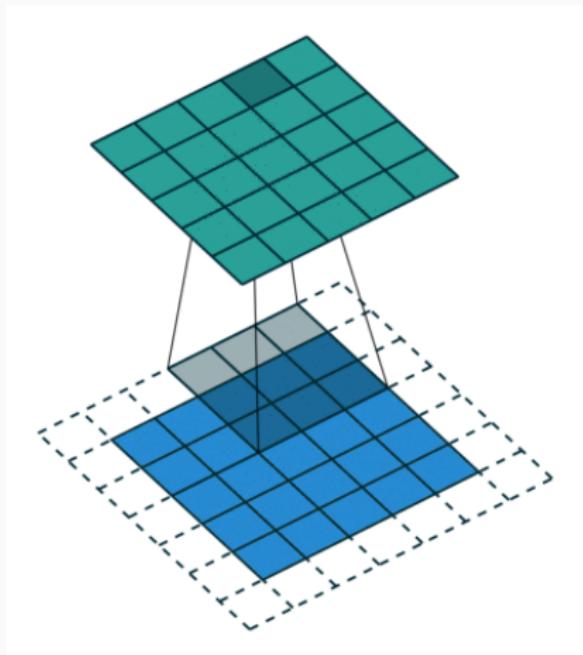
Convolutions with a padding



Différents types de padding existent :

- zéro
- bord
- réflexion

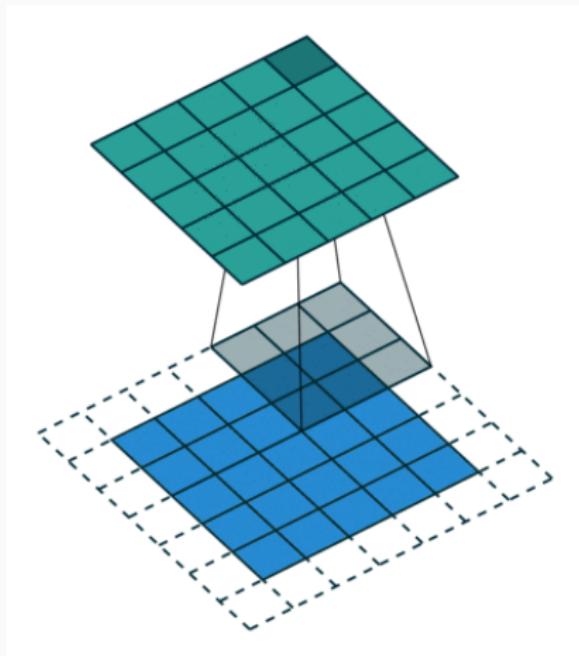
Convolutions with a padding



Différents types de padding existent :

- zéro
- bord
- réflexion

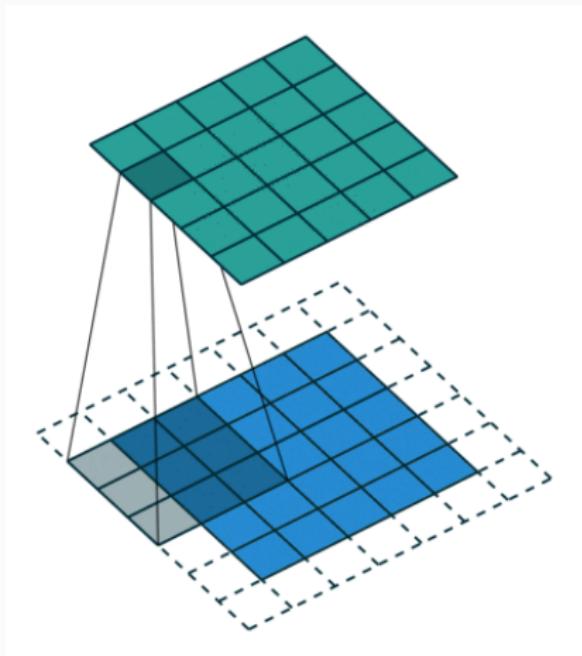
Convolutions with a padding



Différents types de padding existent :

- zéro
- bord
- réflexion

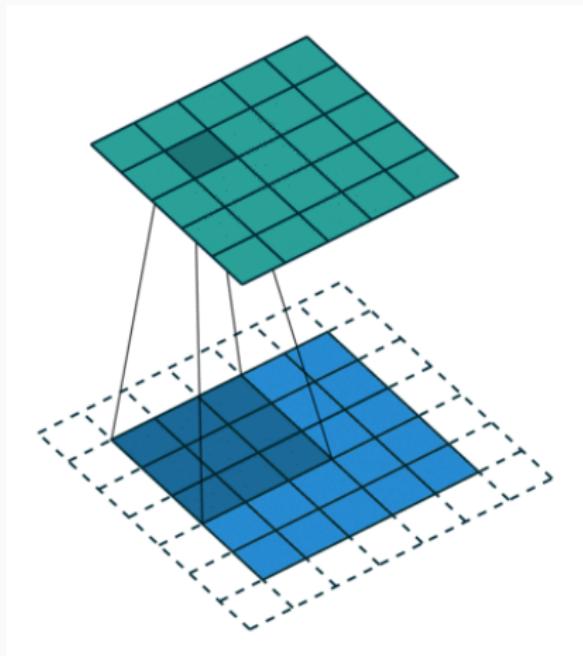
Convolutions with a padding



Différents types de padding existent :

- zéro
- bord
- réflexion

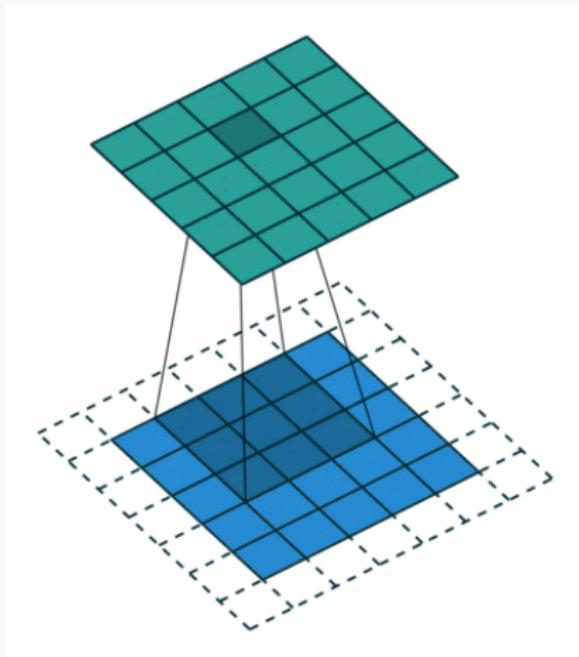
Convolutions with a padding



Différents types de padding existent :

- zéro
- bord
- réflexion

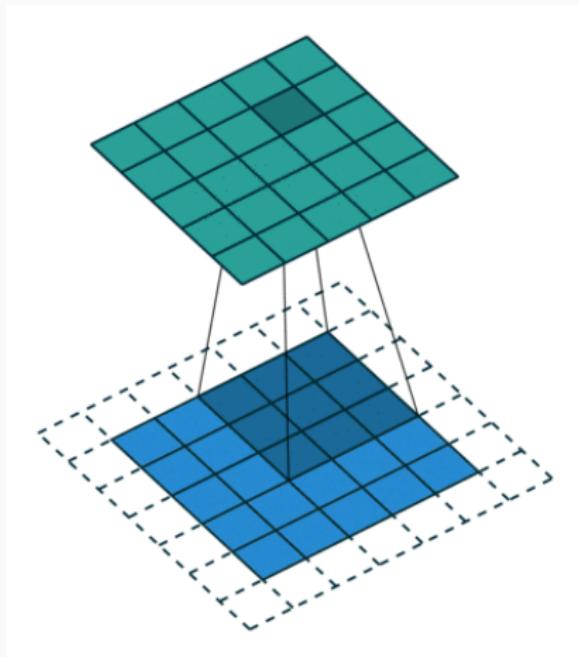
Convolutions with a padding



Différents types de padding existent :

- zéro
- bord
- réflexion

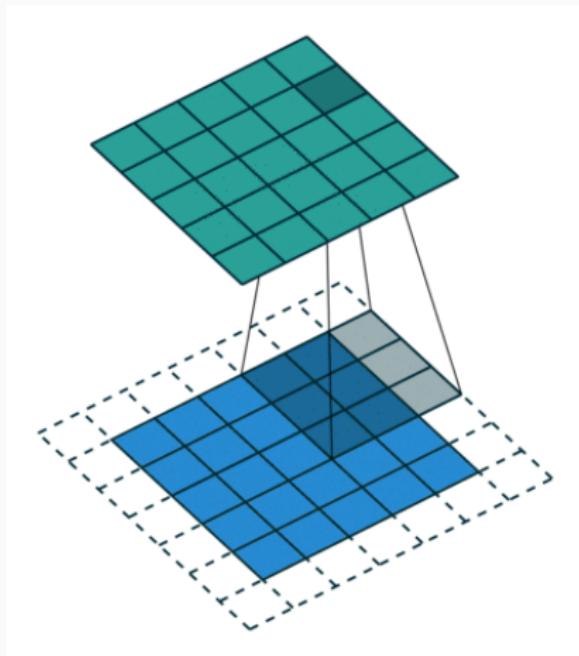
Convolutions with a padding



Différents types de padding existent :

- zéro
- bord
- réflexion

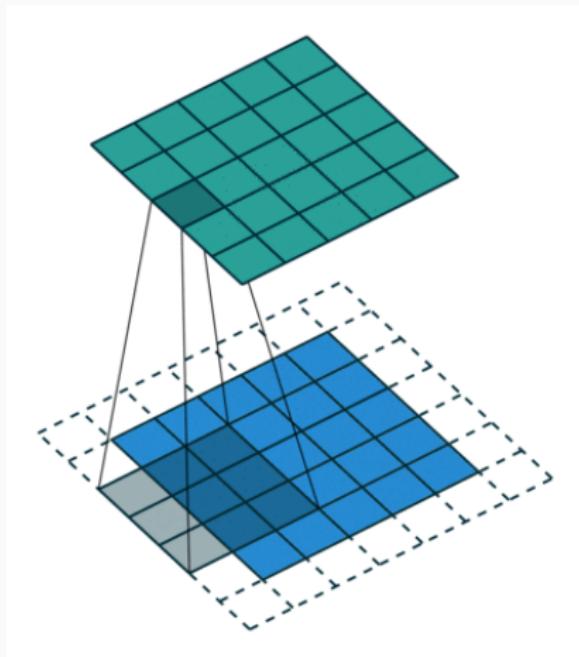
Convolutions with a padding



Différents types de padding existent :

- zéro
- bord
- réflexion

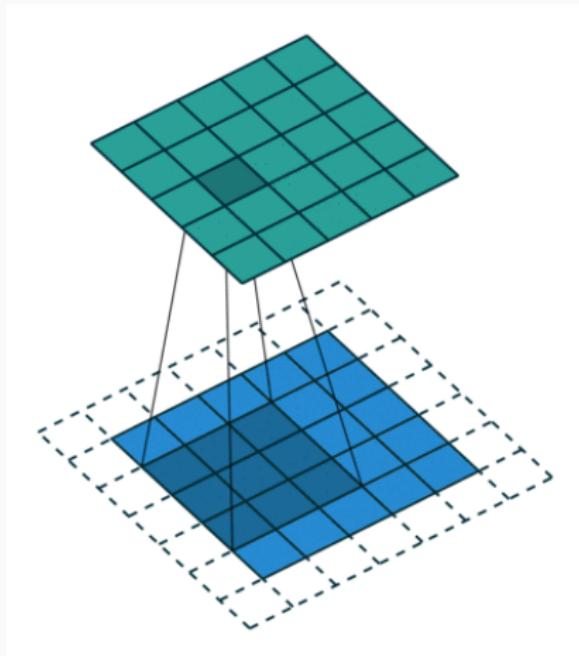
Convolutions with a padding



Différents types de padding existent :

- zéro
- bord
- réflexion

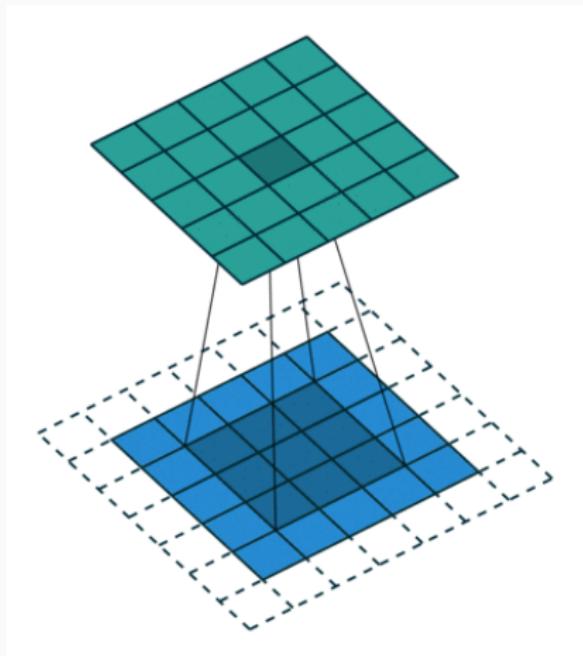
Convolutions with a padding



Différents types de padding existent :

- zéro
- bord
- réflexion

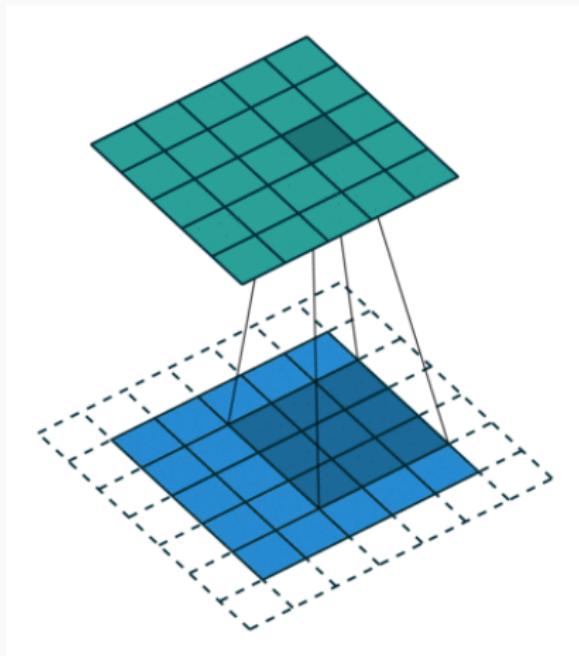
Convolutions with a padding



Différents types de padding existent :

- zéro
- bord
- réflexion

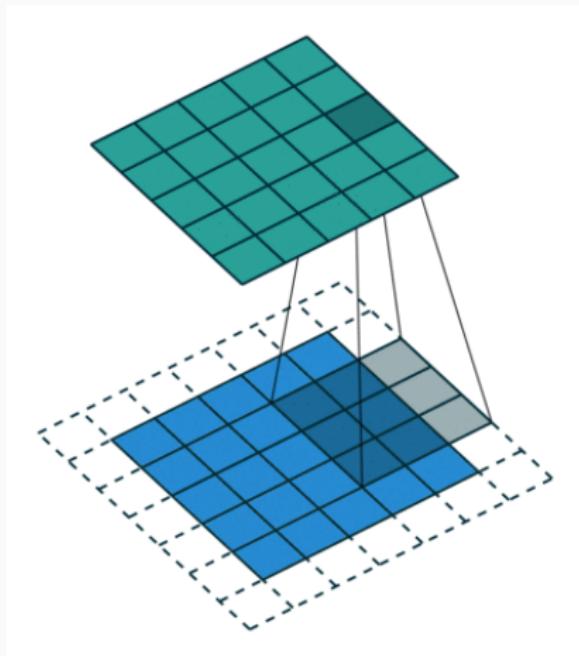
Convolutions with a padding



Différents types de padding existent :

- zéro
- bord
- réflexion

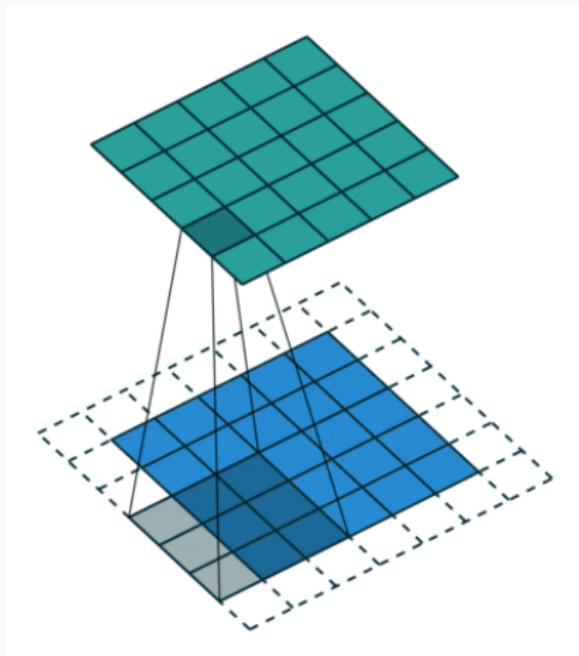
Convolutions with a padding



Différents types de padding existent :

- zéro
- bord
- réflexion

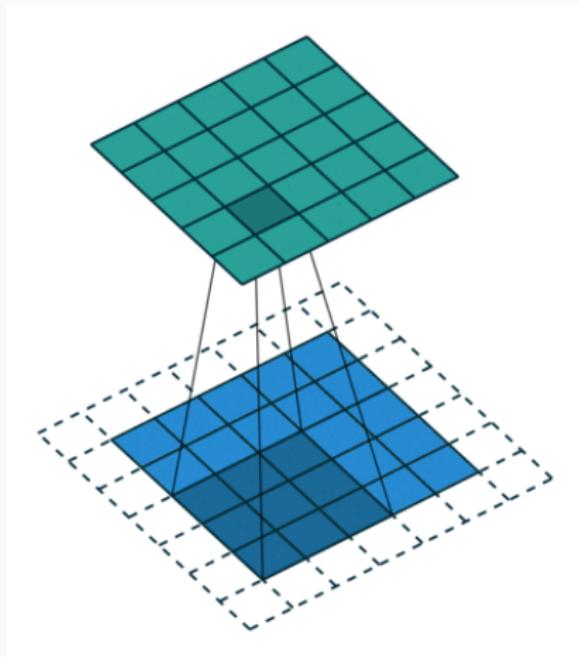
Convolutions with a padding



Différents types de padding existent :

- zéro
- bord
- réflexion

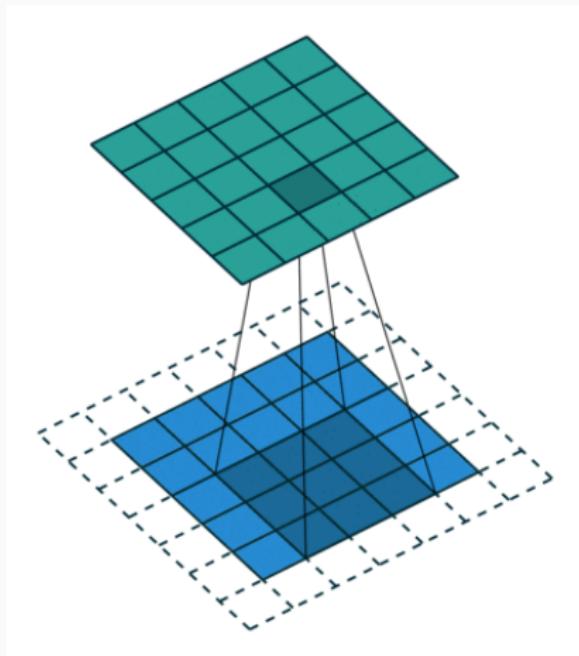
Convolutions with a padding



Différents types de padding existent :

- zéro
- bord
- réflexion

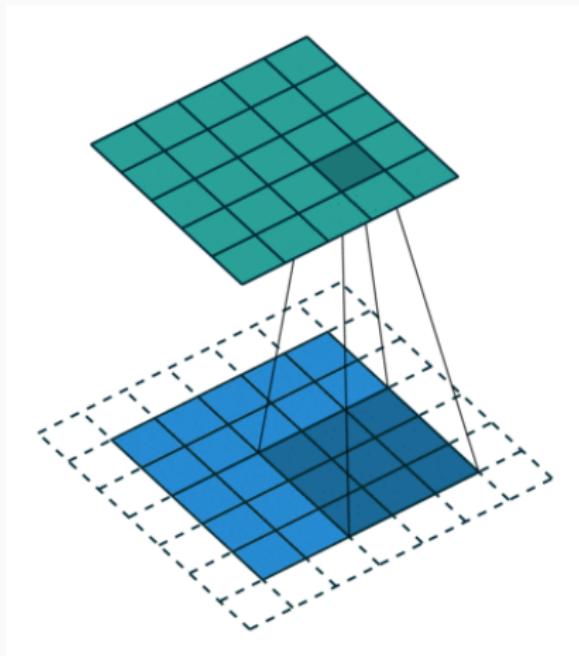
Convolutions with a padding



Différents types de padding existent :

- zéro
- bord
- réflexion

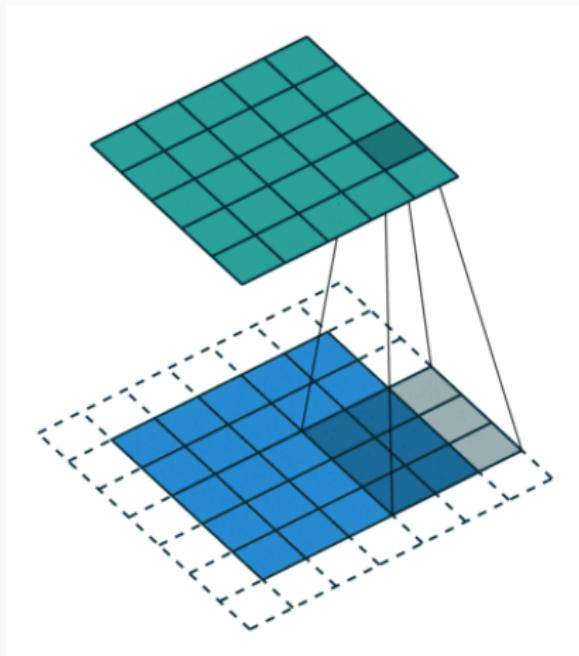
Convolutions with a padding



Différents types de padding existent :

- zéro
- bord
- réflexion

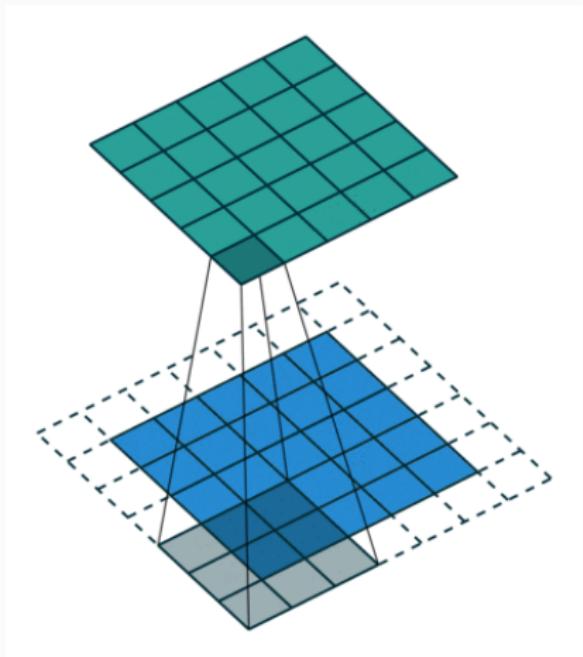
Convolutions with a padding



Différents types de padding existent :

- zéro
- bord
- réflexion

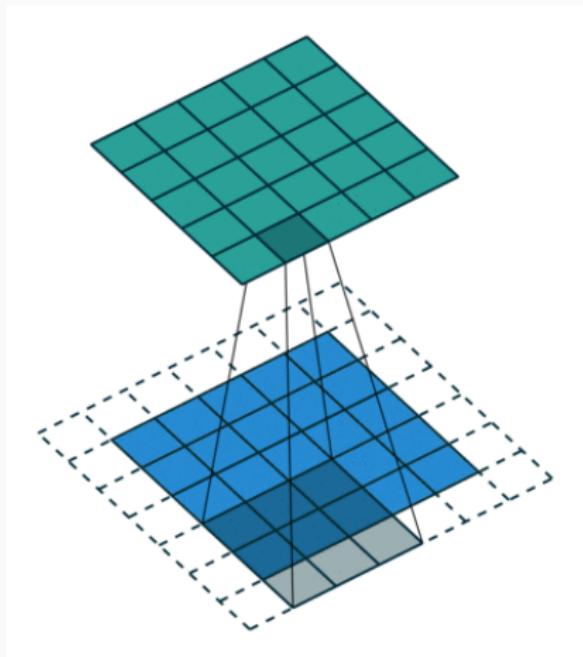
Convolutions with a padding



Différents types de padding existent :

- zéro
- bord
- réflexion

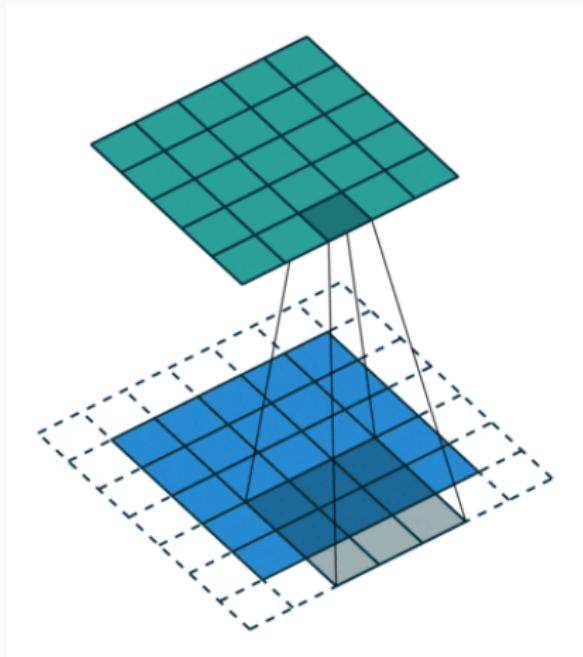
Convolutions with a padding



Différents types de padding existent :

- zéro
- bord
- réflexion

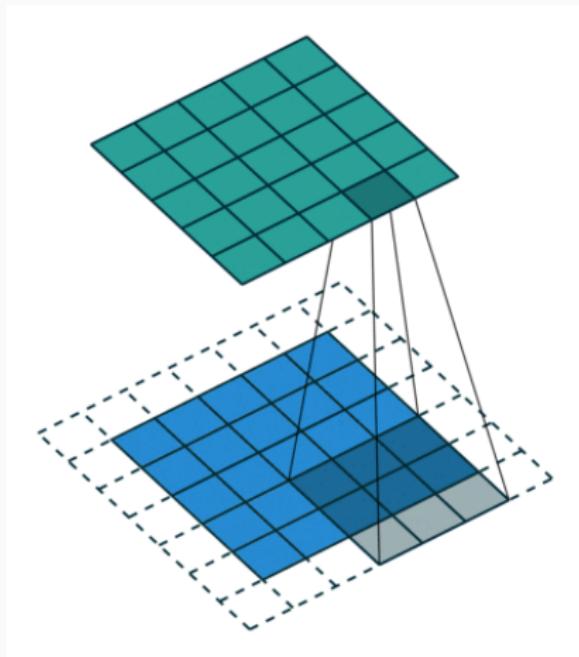
Convolutions with a padding



Différents types de padding existent :

- zéro
- bord
- réflexion

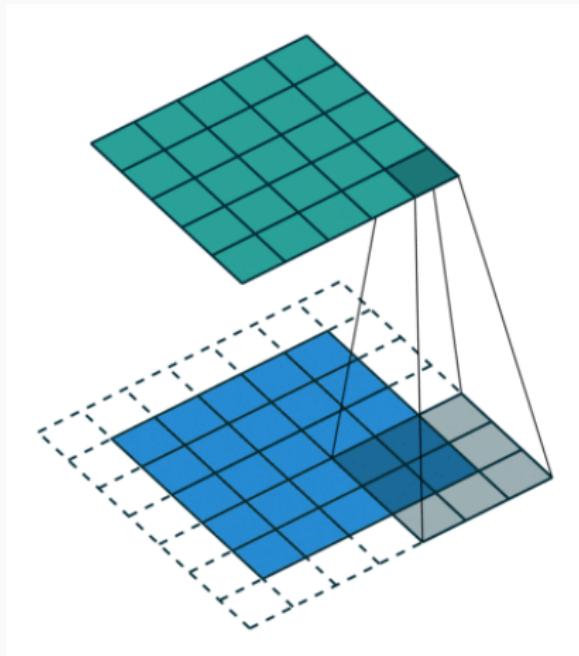
Convolutions with a padding



Différents types de padding existent :

- zéro
- bord
- réflexion

Convolutions with a padding

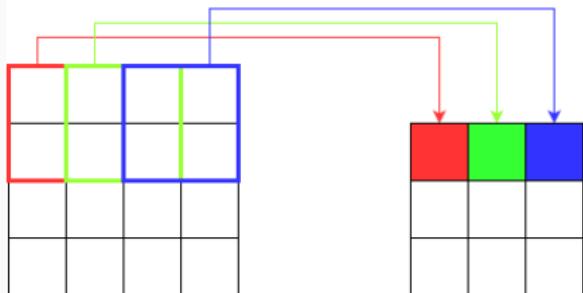


Différents types de padding existent :

- zéro
- bord
- réflexion

Convolutions with a stride

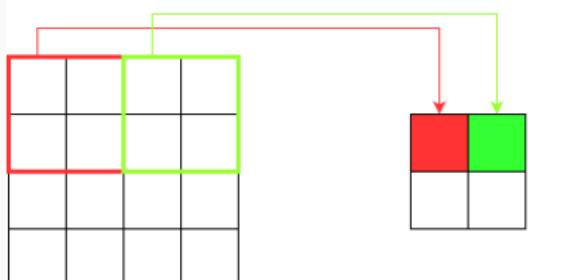
Stride = 1



- Éviter la redondance

- Prendre un pixel parmi s pixels pour une taille s

Stride = 2



Convolutions with channels

0	0	0	0	0	0	...
0	156	155	156	158	158	...
0	153	154	157	159	159	...
0	149	151	155	158	159	...
0	146	146	149	153	158	...
0	145	143	143	148	158	...
...

Input Channel #1 (Red)

-1	-1	1
0	1	-1
0	1	1

Kernel Channel #1



310

+

0	0	0	0	0	0	...
0	167	166	167	169	169	...
0	164	165	168	170	170	...
0	160	162	166	169	170	...
0	156	156	159	163	168	...
0	155	153	153	158	168	...
...

Input Channel #2 (Green)

1	0	0
1	-1	-1
1	0	-1

Kernel Channel #2



-170

0	0	0	0	0	0	...
0	163	162	163	165	165	...
0	160	161	164	166	166	...
0	156	158	162	165	166	...
0	155	155	158	162	167	...
0	154	152	152	157	167	...
...

Input Channel #3 (Blue)

0	1	1
0	1	0
1	-1	1

Kernel Channel #3



325

+

+ 1 = 466

Bias = 1
↑

-25	466			...
				...
				...
				...
...

Output

Convolutions with channels

0	0	0	0	0	0	0	...
0	156	155	156	158	158	158	...
0	153	154	157	159	159	159	...
0	149	151	155	158	159	159	...
0	146	146	149	153	158	158	...
0	145	143	143	148	158	158	...
...

Input Channel #1 (Red)

-1	-1	1
0	1	-1
0	1	1

Kernel Channel #1



314

+

1	0	0
1	-1	-1
1	0	-1

Kernel Channel #2



-175

0	0	0	0	0	0	0	...
0	163	162	163	165	165	165	...
0	160	161	164	166	166	166	...
0	156	158	162	165	166	166	...
0	155	155	158	162	167	167	...
0	154	152	152	157	167	167	...
...

Input Channel #3 (Blue)

0	1	1
0	1	0
1	-1	1

Kernel Channel #3



326

+

+ 1 = 466

Bias = 1

-25	466	466			...
					...
					...
					...
...

Convolutions with channels

0	0	0	0	0	0	0	...
0	156	155	156	158	158	158	...
0	153	154	157	159	159	159	...
0	149	151	155	158	159	159	...
0	146	146	149	153	158	158	...
0	145	143	143	148	158	158	...
...

Input Channel #1 (Red)

-1	-1	1
0	1	-1
0	1	1

Kernel Channel #1



318

+

0	0	0	0	0	0	0	...
0	167	166	167	169	169	169	...
0	164	165	168	170	170	170	...
0	160	162	166	169	170	170	...
0	156	156	159	163	168	168	...
0	155	153	153	158	168	168	...
...

Input Channel #2 (Green)

1	0	0
1	-1	-1
1	0	-1

Kernel Channel #2



-173

0	0	0	0	0	0	0	...
0	163	162	163	165	165	165	...
0	160	161	164	166	166	166	...
0	156	158	162	165	166	166	...
0	155	155	158	162	167	167	...
0	154	152	152	157	167	167	...
...

Input Channel #3 (Blue)

0	1	1
0	1	0
1	-1	1

Kernel Channel #3

1	-1	1
0	1	0
1	-1	1

Kernel Channel #3

$$\begin{array}{r} \downarrow \\ 318 \end{array} + \begin{array}{r} \downarrow \\ -173 \end{array} + \begin{array}{r} \downarrow \\ 329 \end{array} + 1 = 475$$

Bias = 1

-25	466	466	475	...
...
...
...
...

Output

Convolutions with channels

0	0	0	0	0	0	0	...
0	156	155	156	158	158	158	...
0	153	154	157	159	159	159	...
0	149	151	155	158	159	159	...
0	146	146	149	153	158	158	...
0	145	143	143	148	158	158	...
...

Input Channel #1 (Red)

-1	-1	1
0	1	-1
0	1	1

Kernel Channel #1



298

+

0	0	0	0	0	0	0	...
0	167	166	167	169	169	169	...
0	164	165	168	170	170	170	...
0	160	162	166	169	170	170	...
0	156	156	159	163	168	168	...
0	155	153	153	158	168	168	...
...

Input Channel #2 (Green)

1	0	0
1	-1	-1
1	0	-1

Kernel Channel #2



-491

0	0	0	0	0	0	0	...
0	163	162	163	165	165	165	...
0	160	161	164	166	166	166	...
0	156	158	162	165	166	166	...
0	155	155	158	162	167	167	...
0	154	152	152	157	167	167	...
...

Input Channel #3 (Blue)

0	1	1
0	1	0
1	-1	1

Kernel Channel #3

1	0	0
0	1	0
1	-1	1

Kernel Channel #3



487

+ 1 = 295

Bias = 1

-25	466	466	475	...
295				...
				...
				...
...

Output

Convolutions with channels

0	0	0	0	0	0	0	...
0	156	155	156	158	158	158	...
0	153	154	157	159	159	159	...
0	149	151	155	158	159	159	...
0	146	146	149	153	158	158	...
0	145	143	143	148	158	158	...
...

Input Channel #1 (Red)

-1	-1	1
0	1	-1
0	1	1

Kernel Channel #1



148

0	0	0	0	0	0	0	...
0	167	166	167	169	169	169	...
0	164	165	168	170	170	170	...
0	160	162	166	169	170	170	...
0	156	156	159	163	168	168	...
0	155	153	153	158	168	168	...
...

Input Channel #2 (Green)

1	0	0
1	-1	-1
1	0	-1

Kernel Channel #2



-8

0	0	0	0	0	0	0	...
0	163	162	163	165	165	165	...
0	160	161	164	166	166	166	...
0	156	158	162	165	166	166	...
0	155	155	158	162	167	167	...
0	154	152	152	157	167	167	...
...

Input Channel #3 (Blue)

0	1	1
0	1	0
1	-1	1

Kernel Channel #3



646

$$+ 1 = 787$$

Bias = 1

-25	466	466	475	...
295	787			...
				...
				...
...

Convolutions with channels

0	0	0	0	0	0	0	...
0	156	155	156	158	158	158	...
0	153	154	157	159	159	159	...
0	149	151	155	158	159	159	...
0	146	146	149	153	158	158	...
0	145	143	143	148	158	158	...
...

Input Channel #1 (Red)

-1	-1	1
0	1	-1
0	1	1

Kernel Channel #1



158

0	0	0	0	0	0	0	...
0	167	166	167	169	169	169	...
0	164	165	168	170	170	170	...
0	160	162	166	169	170	170	...
0	156	156	159	163	168	168	...
0	155	153	153	158	168	168	...
...

Input Channel #2 (Green)

1	0	0
1	-1	-1
1	0	-1

Kernel Channel #2



-14

0	0	0	0	0	0	0	...
0	163	162	163	165	165	165	...
0	160	161	164	166	166	166	...
0	156	158	162	165	166	166	...
0	155	155	158	162	167	167	...
0	154	152	152	157	167	167	...
...

Input Channel #3 (Blue)

0	1	1
0	1	0
1	-1	1

Kernel Channel #3



653

+ 1 = 798

↑
Bias = 1

-25	466	466	475	...
295	787	798	798	...
...
...
...

Output

Convolutions with channels

0	0	0	0	0	0	...
0	156	155	156	158	158	...
0	153	154	157	159	159	...
0	149	151	155	158	159	...
0	146	146	149	153	158	...
0	145	143	143	148	158	...
...

Input Channel #1 (Red)

0	0	0	0	0	0	...
0	167	166	167	169	169	...
0	164	165	168	170	170	...
0	160	162	166	169	170	...
0	156	156	159	163	168	...
0	155	153	153	158	168	...
...

Input Channel #2 (Green)

0	0	0	0	0	0	...
0	163	162	163	165	165	...
0	160	161	164	166	166	...
0	156	158	162	165	166	...
0	155	155	158	162	167	...
0	154	152	152	157	167	...
...

Input Channel #3 (Blue)

-1	-1	1
0	1	-1
0	1	1

Kernel Channel #1



161

+

1	0	0
1	-1	-1
1	0	-1

Kernel Channel #2



-9

0	1	1
0	1	0
1	-1	1

Kernel Channel #3



659

+

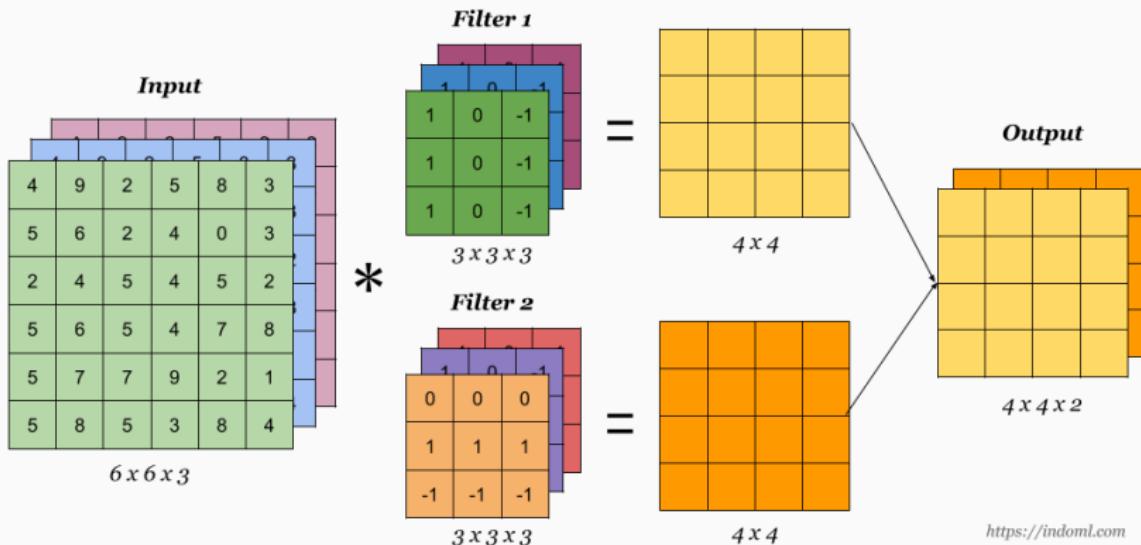
+

+ 1 = 812

Bias = 1

-25	466	466	475	...
295	787	798	812	...
...
...
...

Convolutions avec plusieurs noyaux



Couche de convolution

Couche de convolution avec c_{in} canaux d'entrée et c_{out} canaux de sortie :

- Image d'entrée \mathbf{x} avec c_{in} **canaux**
- Image de sortie \mathbf{y} avec c_{out} **canaux**.
- Noyau : κ tel que pour tout $(k, \ell) \in [-k, k] \times [-k, k]$

$\kappa(k, \ell) \in \mathbb{R}^{c_{\text{out}} \times c_{\text{in}}}$, est une matrice de taille $c_{\text{out}} \times c_{\text{in}}$

- Biais : $\mathbf{b} \in \mathbb{R}^{c_{\text{out}}}$

$$\mathbf{y}(i, j) = \text{Conv}(\mathbf{x}; \kappa, \mathbf{b})(i, j)$$

$$= \left[\sum_{(k, \ell) \in [-s, s] \times [-s, s]} \kappa(k, \ell) \mathbf{x}(i + k, j + \ell) \right] + \mathbf{b} \in \mathbb{R}^{c_{\text{out}}}$$

Comptage du nombre de paramètres

Prenons une convolution qui sera appliquée à une entrée avec c_{in} canaux, ayant des noyaux de taille $2k + 1$ et produisant c_{out} canaux en sortie.

Entrée de taille $(c_{\text{in}}, M, N) \rightarrow$ Convolution avec des noyaux de taille $(2k + 1) \times (2k + 1) \rightarrow$ Sortie de taille $(c_{\text{out}}, M - 2k, N - 2k)$

Combien de paramètres sont nécessaires pour définir cette convolution ?

Comptage du nombre de paramètres

Prenons une convolution qui sera appliquée à une entrée avec c_{in} canaux, ayant des noyaux de taille $2k + 1$ et produisant c_{out} canaux en sortie.

Entrée de taille $(c_{\text{in}}, M, N) \rightarrow$ Convolution avec des noyaux de taille $(2k + 1) \times (2k + 1) \rightarrow$ Sortie de taille $(c_{\text{out}}, M - 2k, N - 2k)$

Combien de paramètres sont nécessaires pour définir cette convolution ?

Il y a c_{out} groupes de noyaux c_{in} de taille $(2k + 1) \times (2k + 1)$ à définir.

Comptage du nombre de paramètres

Prenons une convolution qui sera appliquée à une entrée avec c_{in} canaux, ayant des noyaux de taille $2k + 1$ et produisant c_{out} canaux en sortie.

Entrée de taille $(c_{\text{in}}, M, N) \rightarrow$ Convolution avec des noyaux de taille $(2k + 1) \times (2k + 1) \rightarrow$ Sortie de taille $(c_{\text{out}}, M - 2k, N - 2k)$

Combien de paramètres sont nécessaires pour définir cette convolution ?

Il y a c_{out} groupes de noyaux c_{in} de taille $(2k + 1) \times (2k + 1)$ à définir.

Conclusion : $c_{\text{out}} \times c_{\text{in}} \times (2k + 1) \times (2k + 1) + c_{\text{out}}$ paramètres.

Les CNN sont composés de trois ingrédients principaux :

① Champs réceptifs locaux

- les unités cachées sont connectées uniquement à une petite région de leur entrée,

② Poids partagés

- mêmes poids et biais pour toutes les unités d'une couche cachée,

mais aussi

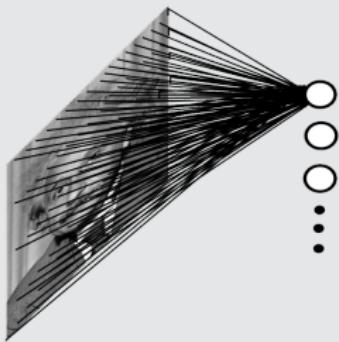
③ Redondance: plus d'unités dans une couche cachée que d'entrées,

④ Sparsité: les unités ne doivent pas toutes s'activer pour le même stimulus. (un noyau par motif)

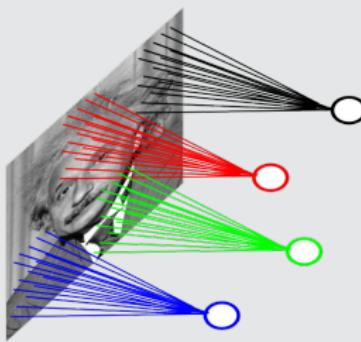
Tout cela s'inspire du cortex visuel.

Champs réceptifs locaux → Couche localement connectée

- Chaque unité dans une couche cachée ne peut voir qu'un petit voisinage de son entrée,
- Cela capte le concept de spatialité.



Connecté entièrement



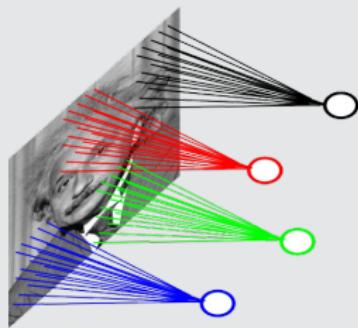
Connecté localement

Pour une image de 200×200 et 40 000 unités cachées :

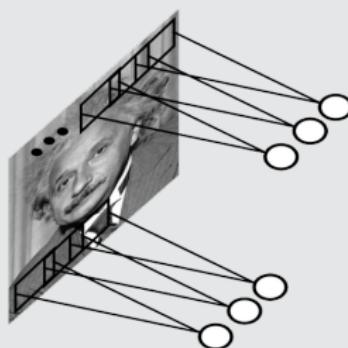
- Fully connected : 1.6 milliard de paramètres $((200 \times 200)^2)$
- Connecté localement (Noyaux de taille 10×10) : 4 millions de paramètres. $(200 \times 200 \times 10 \times 10)$

Champs réceptifs auto-similaires → Poids partagés

- Déetecter des caractéristiques indépendamment de leur position (invariance de translation),
- Utiliser des convolutions pour apprendre des motifs simples de l'entrée.



Connecté localement



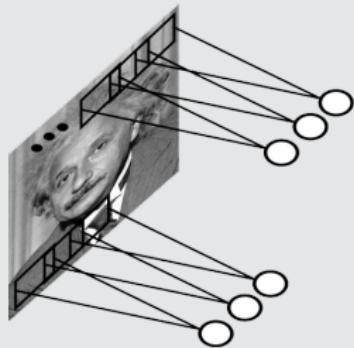
Poids partagés

Pour une image de 200×200 et 40 000 unités cachées :

- Connecté localement (champs de 10×10) : 4 millions de paramètres,
- & Poids partagés : 100 paramètres (10×10).

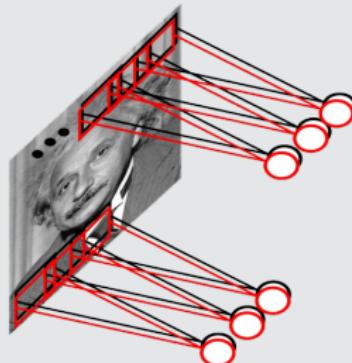
Cellules spécialisées → Banque de filtres

- Utiliser une banque de filtres pour détecter plusieurs motifs à chaque endroit,
- Plusieurs convolutions avec des noyaux différents,
- Le résultat est un tableau 3D, où chaque tranche est une carte de caractéristiques.



Poids partagés

(1 entrée → 1 carte de caractéristiques)

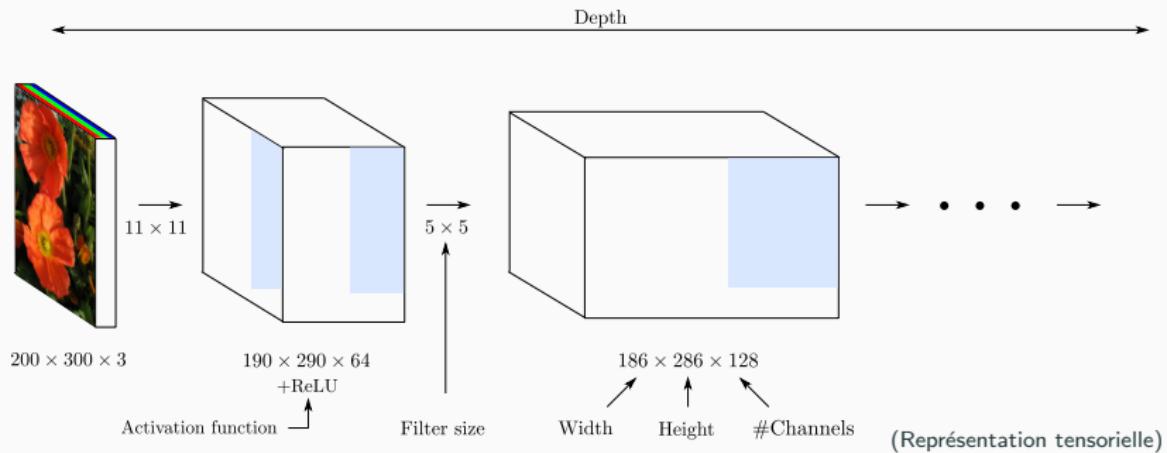


Banque de filtres

(1 entrée → 2 cartes de caractéristiques)

- Champs de 10×10 & 10 caractéristiques en sortie : 1 000 paramètres.

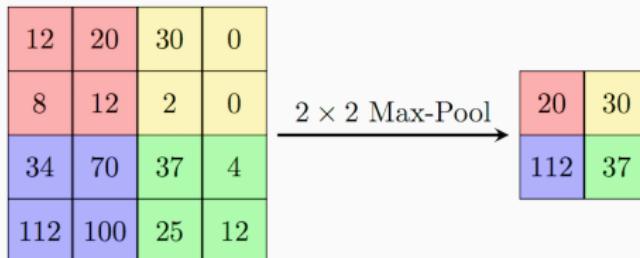
Surcomplétude → augmenter le nombre de canaux



- **Redondance** : augmenter le nombre de canaux entre les couches.
- Pouvons-nous encore mieux contrôler la taille des cartes de features ?

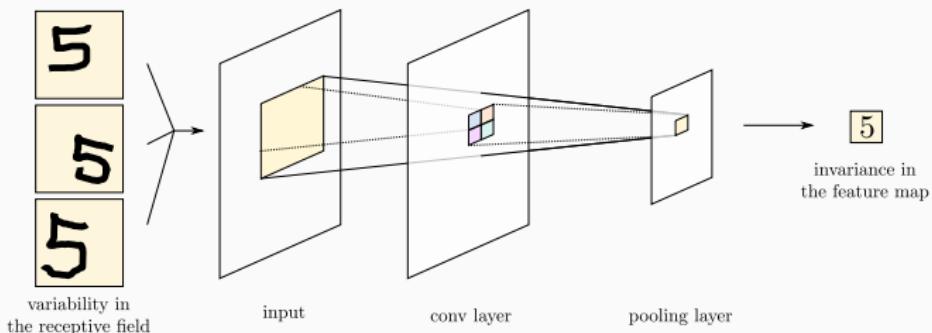
Couche de Pooling

- Utilisée après chaque couche de convolution pour imiter les **cellules complexes**,
 - Partitionner l'image en une grille de fenêtres de $z \times z$ (généralement $z = 2$),
 - **max-pooling** : prendre le max dans la fenêtre



- **average-pooling** : prendre la moyenne

Couche de Pooling



- Rend la sortie **inchangée** même si l'entrée est légèrement modifiée,
- Permet une certaine invariance/robustesse par rapport à la position exacte,
- Simplifie/Condense/Résume la sortie des couches cachées,
- **Augmente les champs réceptifs effectifs** (par rapport à la première couche),
- Pour une entrée (c_{in}, M, N) , la taille de sortie d'une couche de pooling de taille z est $(c_{\text{in}}, M/z, N/z)$.

Paramétrisation des CNNs

Configurer une **couche de convolution** nécessite de choisir :

- Taille du filtre : $n \times n$
- Canaux de sortie : C
- (Stride : s)
- (Padding : p)

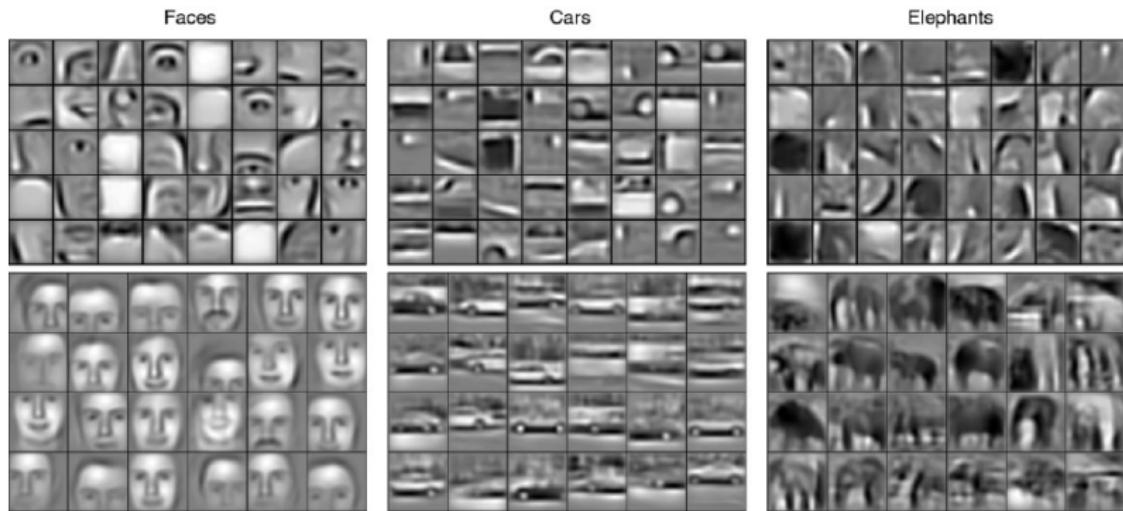
Les poids du filtre κ et le biais b sont appris par rétropropagation.

Configurer une **couche de pooling** nécessite de choisir :

- Taille du pooling : $z \times z$
- Règle d'agrégation : **max-pooling**, average-pooling, ...

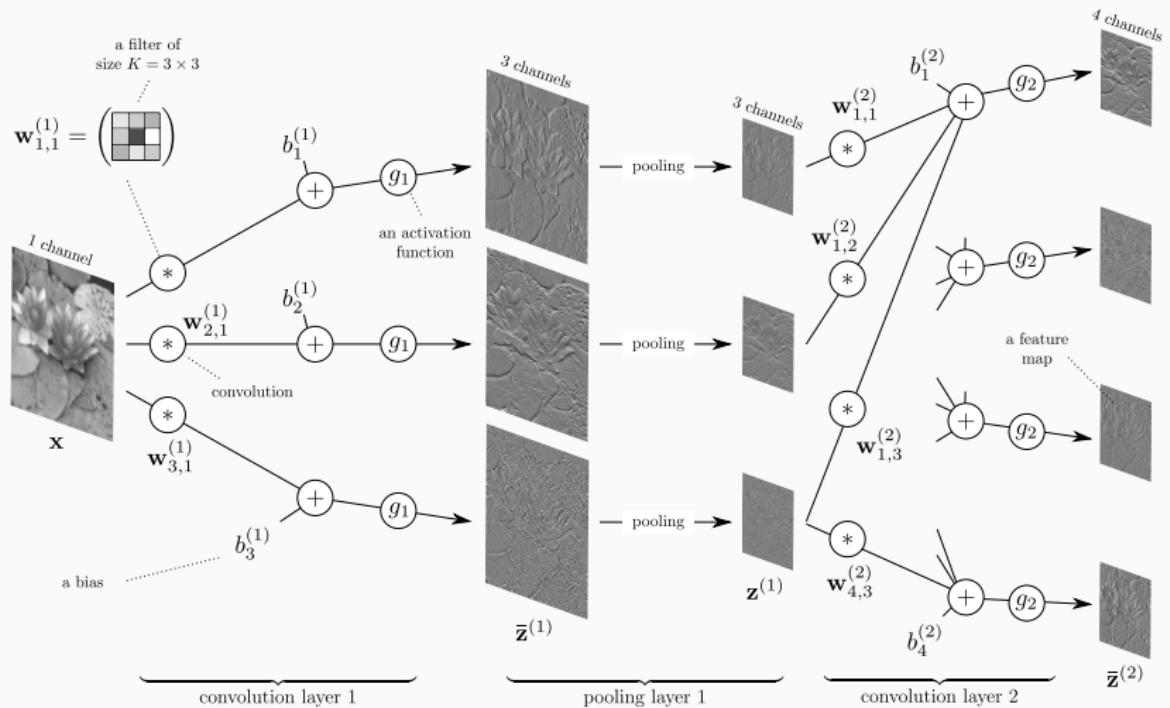
Il n'y a pas de paramètres à apprendre ici.

Apprentissage profond – Hiérarchie des caractéristiques

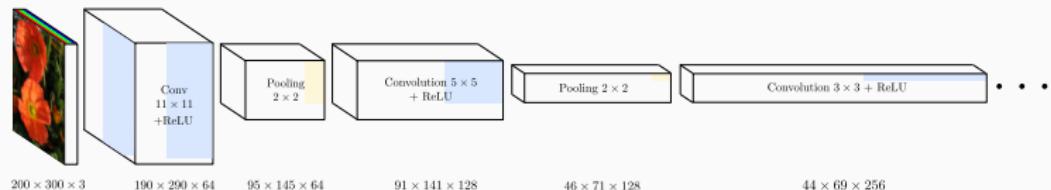


- Les caractéristiques de niveau intermédiaire à élevé sont spécifiques à la tâche donnée.
- Les représentations de bas niveau contiennent des caractéristiques moins spécifiques.
(peuvent être partagées pour de nombreuses applications différentes)

Tous les concepts ensemble



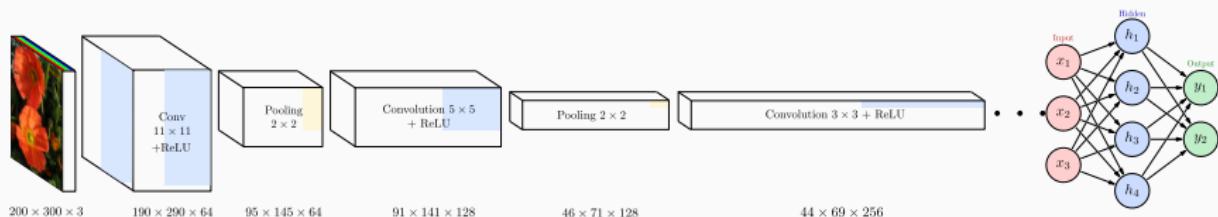
Tous les concepts ensemble avec représentation tensorielle



CNN : Alternatif :

Conv + ReLU + pooling

Tous les concepts ensemble avec représentation tensorielle



CNN : Alternatif :

Conv + ReLU + pooling

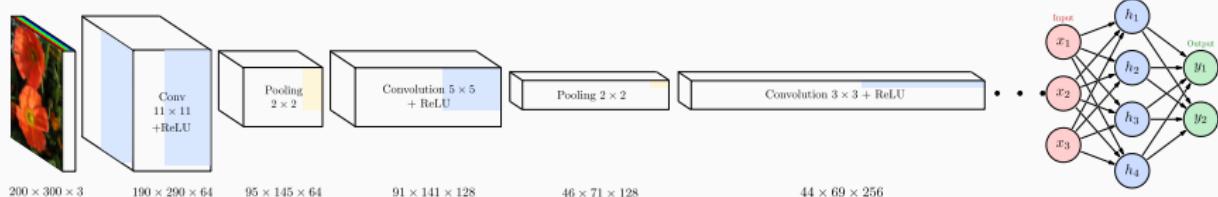
Fin du réseau :

Brancher un réseau de neurones

classique :

Couches cachées entièrement
connectées (linéaires) + ReLU

Tous les concepts ensemble avec représentation tensorielle



Réseau complet :

- **CNN** : Extraire des caractéristiques/features spécifiques aux données spatiales
- **Partie entièrement connectée (fully connected)** : Utiliser les features du CNN pour une tâche de classification spécifique
- **Entraînement** : Apprendre à la fois la classification et l'extraction de features ensemble

CNN : Alternatif :

Conv + ReLU + pooling

Fin du réseau :

Brancher un réseau de neurones classique :

Couches cachées entièrement connectées (linéaires) + ReLU

Questions?

Slides from Charles Deledalle

Sources, images courtesy and acknowledgment

K. Chatfield

P. Gallinari,

C. Hazırbaş

A. Horodniceanu

Y. LeCun

V. Lepetit

L. Masuch

A. Ng

M. Ranzato