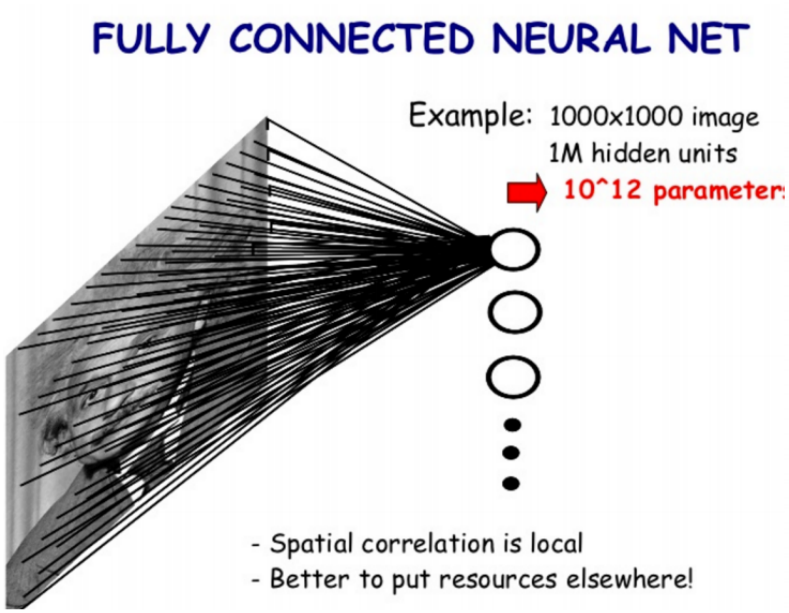
Deep Learning

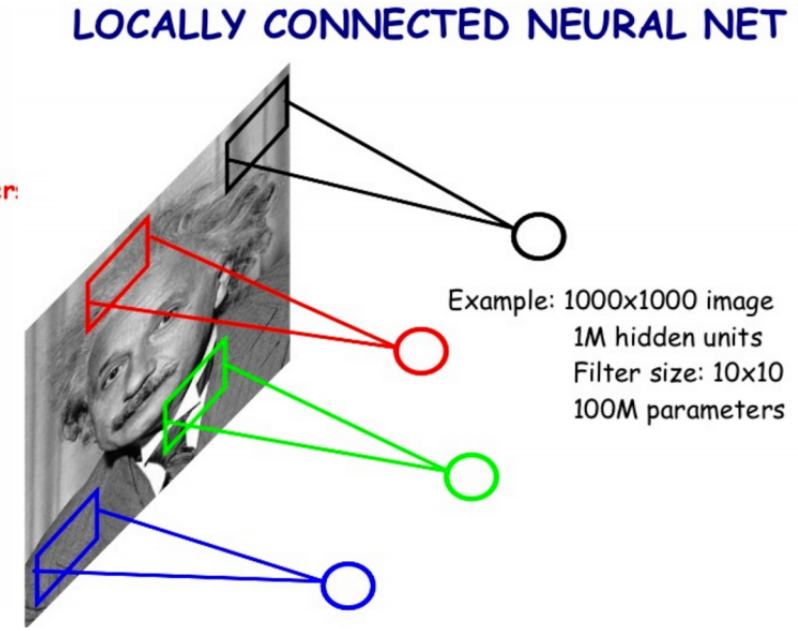
Jérôme Pasquet

December 5, 2019

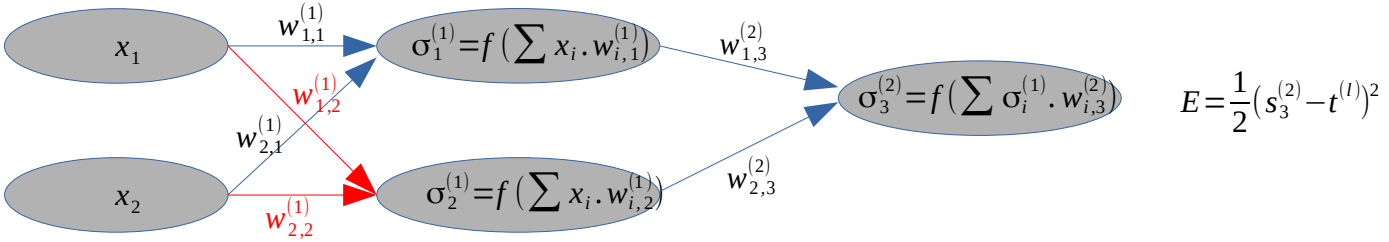
Une inspiration biologique ?

Du MLP au CNN...transition





Poids partagés



Rappel :

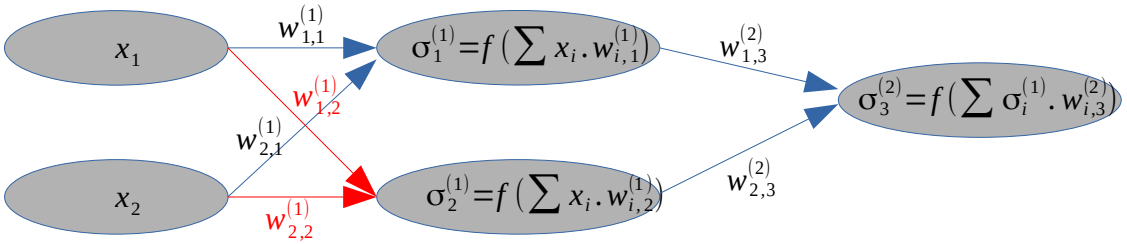
$$\frac{\delta E}{\delta w^{(l)}_{ij}} = \sigma^{(l-1)}_i \phi^{(l)}_j$$

$$\phi^{(l)}_j = \sigma'^{(l)}_j \begin{cases} (\sigma^{(l)}_j - t^{(l)}_j) & \text{Si } l \text{ couche de sortie} \\ \sum_{k \in C_{l+1}} (w^{(l+1)}_{jk} \phi^{(l+1)}_k) & \text{Si couche cachée} \end{cases}$$

Détailler les formules pour $w^{(1)}_{i,1}$, que se passe-t-il si $w^{(1)}_{1,1} = w^{(1)}_{1,2}$?

Attention à la notation qui change ! [cf cours 1R]

Poids partagés



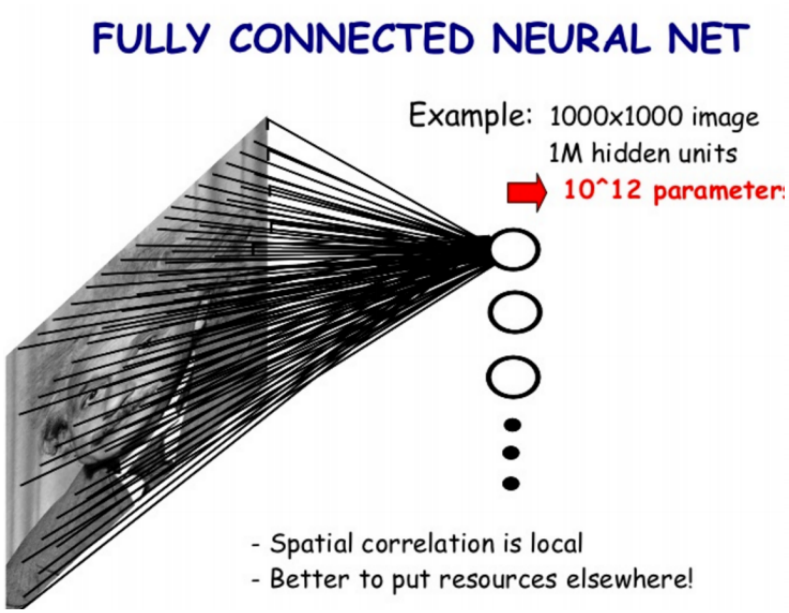
$$E = \frac{1}{2}(s_3^{(2)} - t^{(l)})^2$$

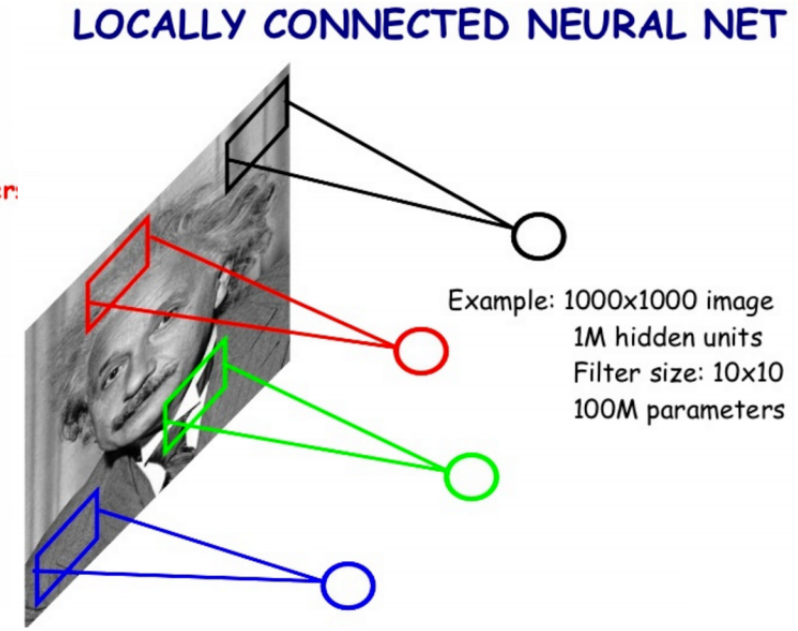
$$\frac{\delta E}{\delta w^{(1)}_{1,1}} = x_1 \phi_1^{(1)} = x_1 \sigma_1'^{(l)} . (w^{(2)}_{13} . \phi_3^{(2)})$$

$$\frac{\delta E}{\delta w^{(1)}_{1,2}} = x_1 \phi_2^{(1)} = x_1 \sigma_2'^{(l)} . (w^{(2)}_{23} . \phi_3^{(2)})$$

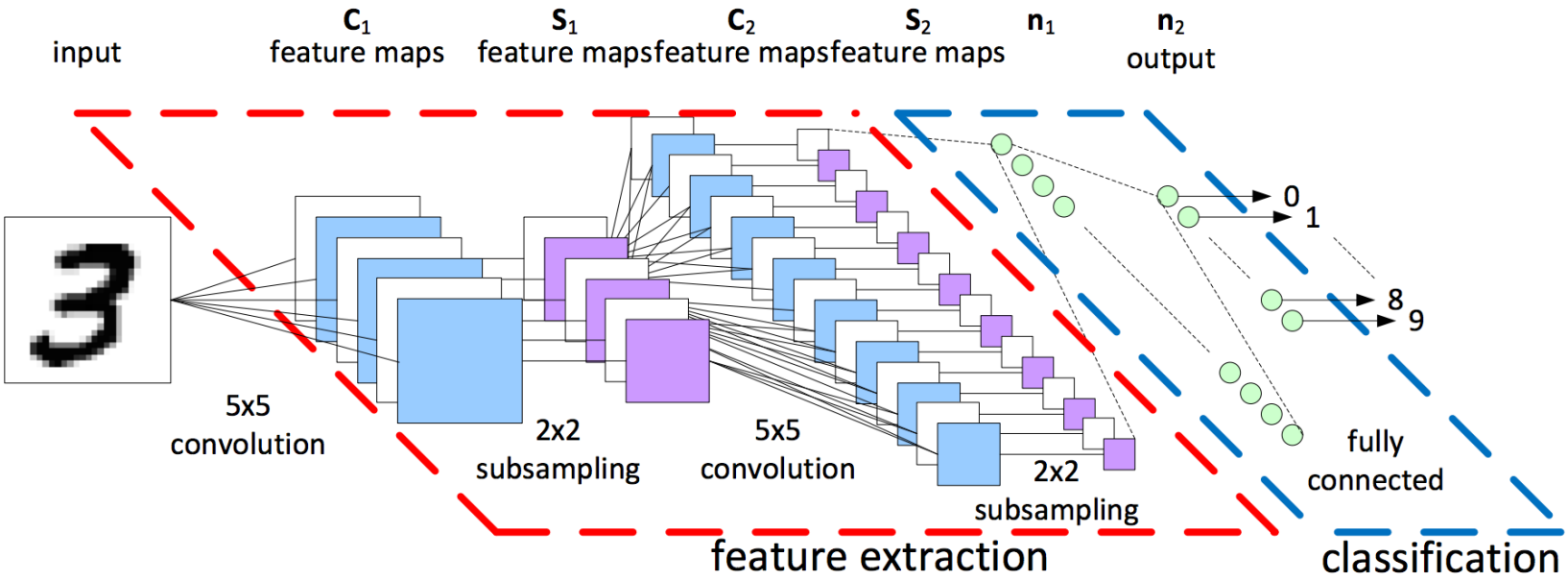
$$\frac{\delta E}{\delta w^{(1)}_{*,1}} = x_1 \sum_{p=Partage} \sigma_p'^{(l)} f'(w^{(2)}_{p3}) \phi_3^{(2)}$$

Du MLP au CNN...transition



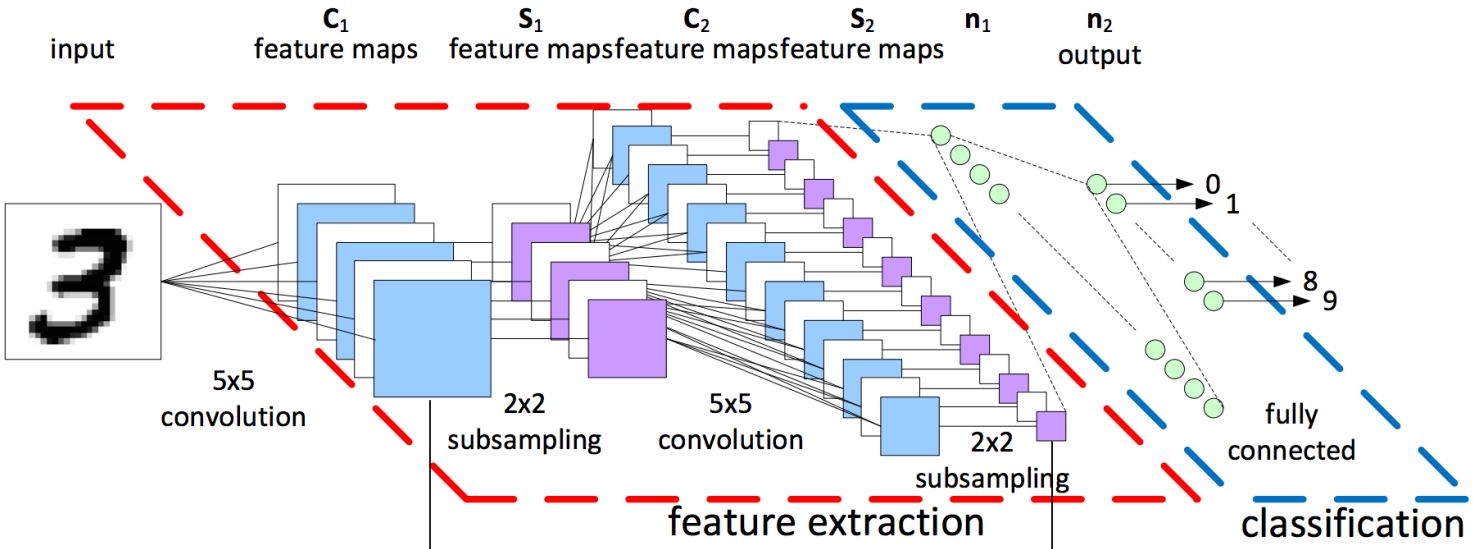


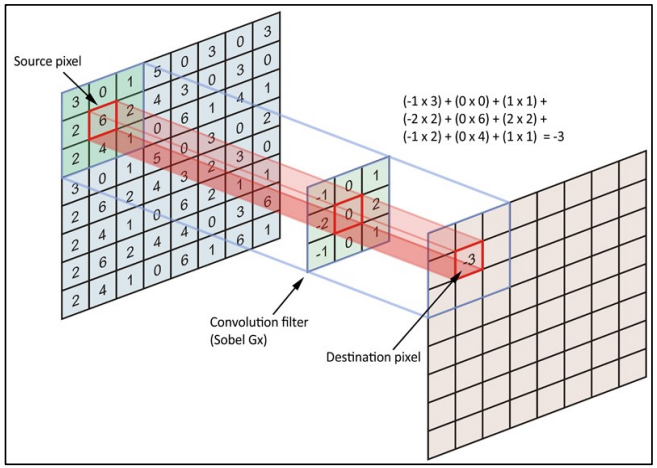
Les CNN

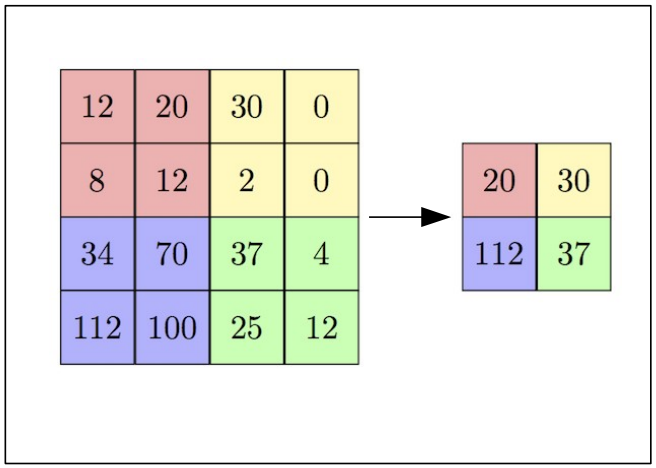


Les cartes de caractéristiques sont les images résultantes des convolutions.

Le CNN



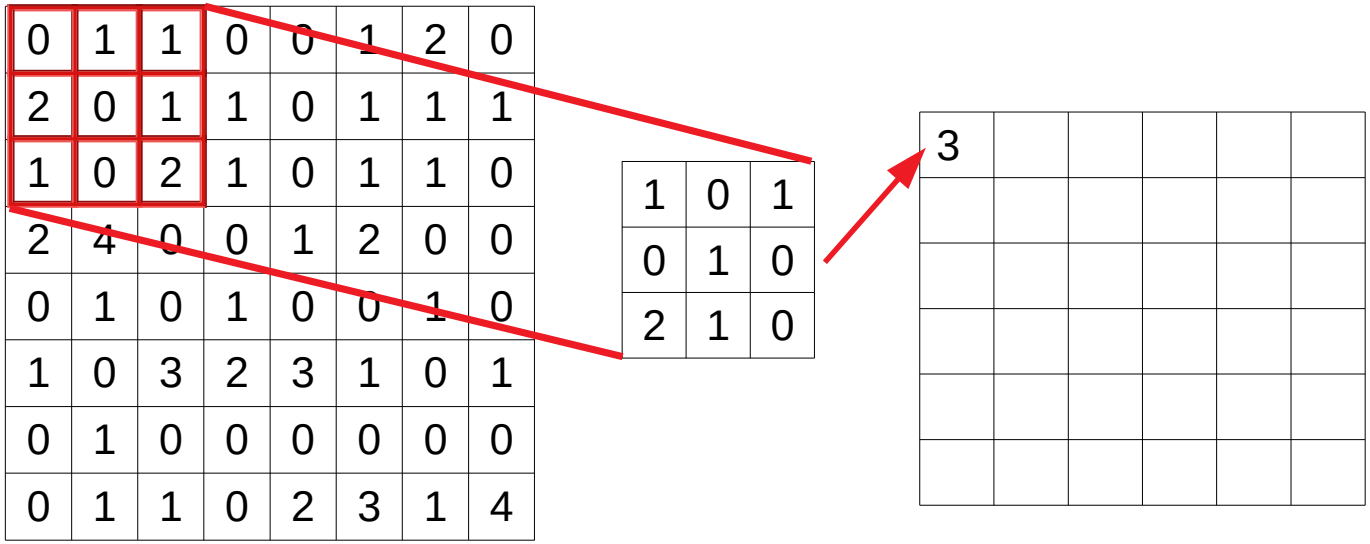




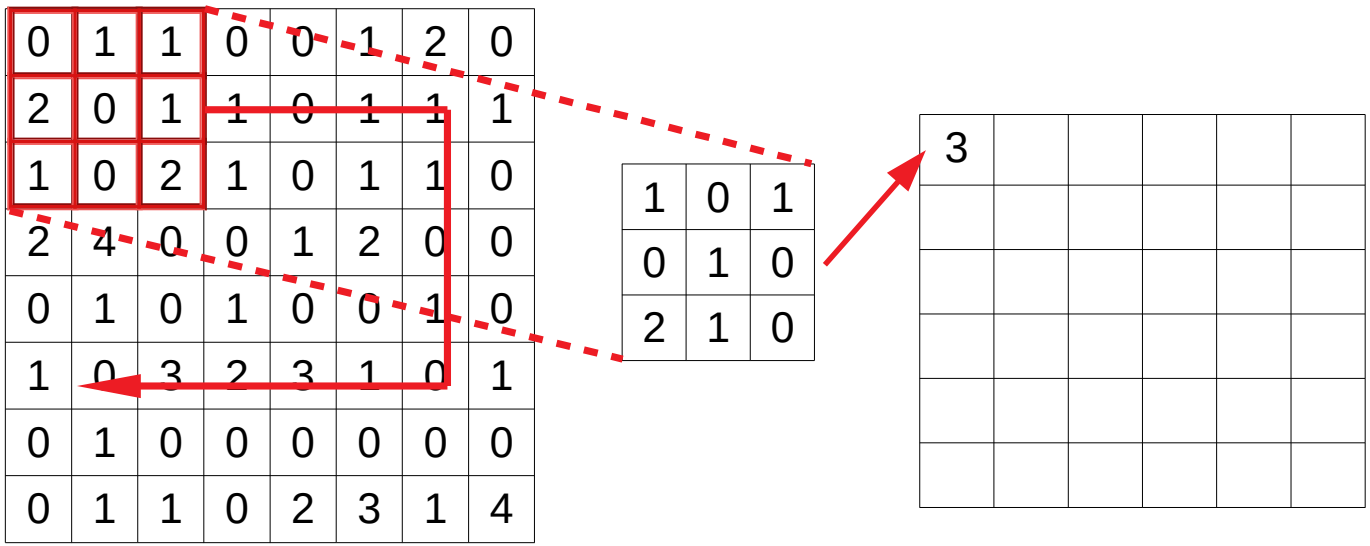
Différents types de pooling...

La convolution

Un peu de vocabulaire : le stride (le pas), le padding (la marge) et le kernel (le noyau).



La convolution



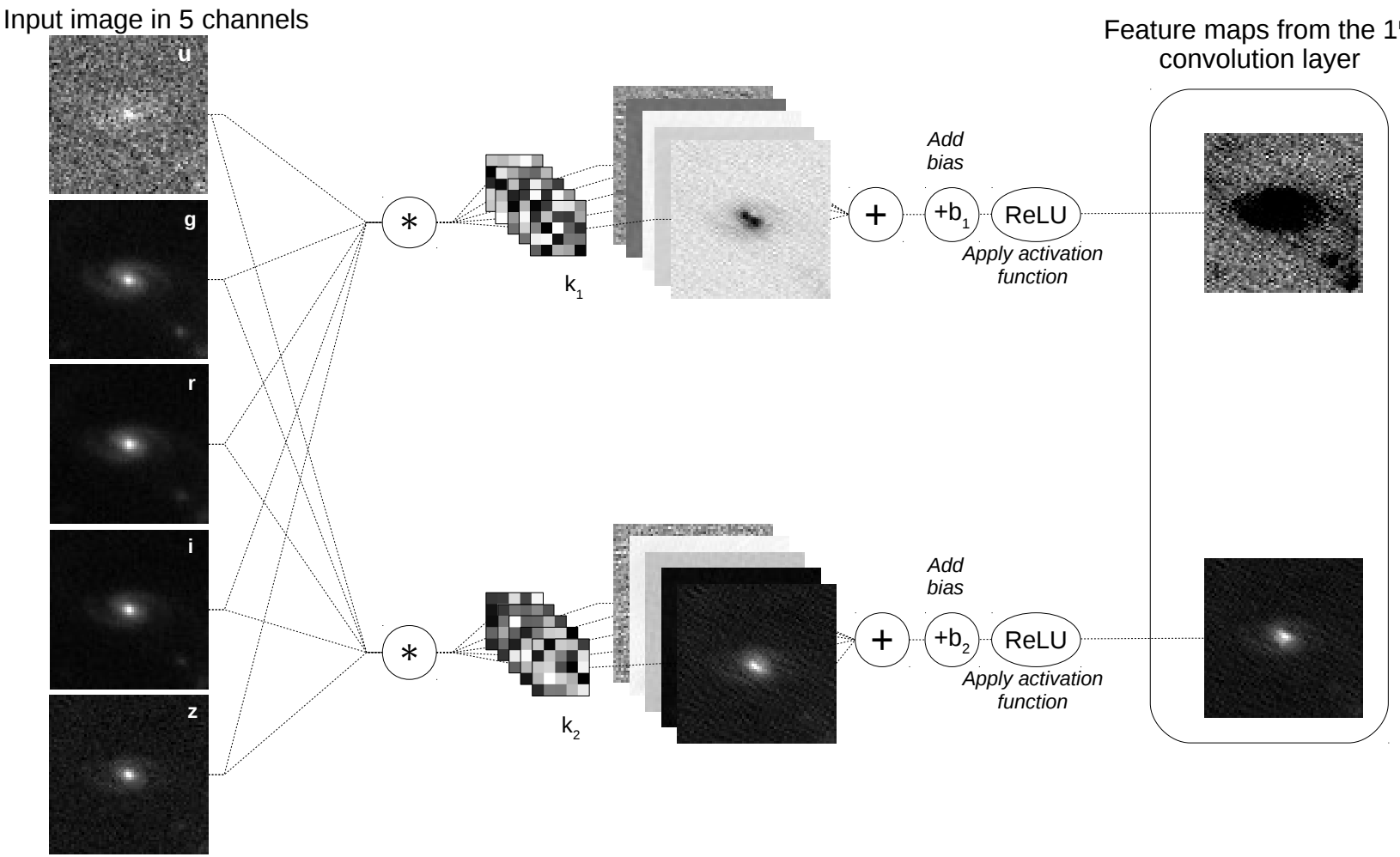
La convolution

0	1	1	0	0	1	2	0
2	0	1	1	0	1	1	1
1	0	2	1	0	1	1	0
2	4	0	0	1	2	0	0
0	1	0	1	0	0	1	0
1	0	3	2	3	1	0	1
0	1	0	0	0	0	0	0
0	1	1	0	2	3	1	4

1	0	1
0	1	0
2	1	0

3	4	7	3	4	5
11	11	2	3	6	7
8	3	3	5	3	2
5	7	7	7	8	5
1	7	2	4	2	0
6	5	8	5	10	9

La convolution étendue



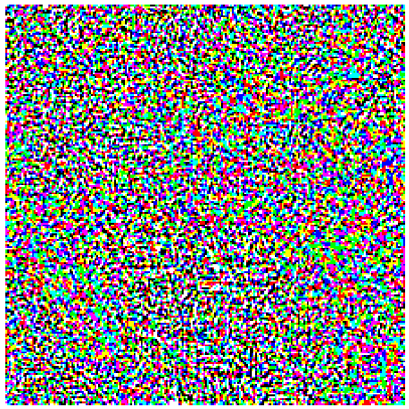
Une classification robuste ?

A lire au second semestre : *Explaining and harnessing adversarial examples* - Ian J. Goodfellow, Jonathon Shlens & Christian Szegedy

$\boldsymbol{x}$

"panda"
57.7% confidence

$+ .007 \times$



$\text{sign}(\nabla_{\boldsymbol{x}} J(\boldsymbol{\theta}, \boldsymbol{x}, y))$

"nematode"
8.2% confidence

$=$

$\boldsymbol{x} + \epsilon\text{sign}(\nabla_{\boldsymbol{x}} J(\boldsymbol{\theta}, \boldsymbol{x}, y))$

"gibbon"
99.3 % confidence

Tensorflow

tf.nn.conv2d(: fonction allouant le poids et faisant la convolution.
input, : tenseur en entrée
filters, : taille des filtres
strides, : pas du kernel
padding, : “VALID” ou “SAME”
data_format=’NHWC’)
tf.nn.**bias_add**(: rajouter le biais à des features map
value, : tenseur auquel on souhaite appliquer un biais
bias) : tenseur de biais.

Tensorflow

tf.nn.conv2d(: fonction allouant le poids et faisant la convolution.
input, : tenseur en entrée
filters, : taille des filtres
strides, : pas du kernel
padding, : “VALID” ou “SAME”
data_format=’NHWC’)
tf.nn.**bias_add**(: rajouter le biais à des features map
value, : tenseur auquel on souhaite appliquer un biais
bias) : tenseur de biais.