Programme du cours

- Introduction et contexte
- Réseaux de neurones et optimisation
- Deep Computer Vision
- Deep Sequence model
- Deep Generative model

Qu’est-ce que la vision par ordinateur ?

- La vision par ordinateur est un sous domaine de l’Intelligence Artificielle qui vise à permettre aux ordinateurs d’interpréter et comprendre les informations visuelles du monde, comme le font les humains avec leurs yeux et cerveaux.

Qu’est-ce que la vision par ordinateur ?

- La vision par ordinateur est un sous domaine de l’Intelligence Artificielle qui vise à permettre aux ordinateurs d’interpréter et comprendre les informations visuelles du monde, comme le font les humains avec leurs yeux et cerveaux.

- Implique le développement d’algorithmes permettant aux machines d’analyser et traiter et de donner sens aux images numériques et aux vidéos.

Qu’est-ce que la vision par ordinateur ?

- La vision par ordinateur est un sous domaine de l’Intelligence Artificielle qui vise à permettre aux ordinateurs d’interpréter et comprendre les informations visuelles du monde, comme le font les humains avec leurs yeux et cerveaux.

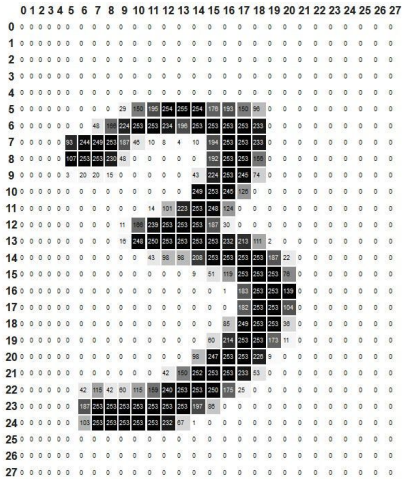
- Implique le développement d’algorithmes permettant aux machines d’analyser et traiter et de donner sens aux images numériques et aux vidéos.

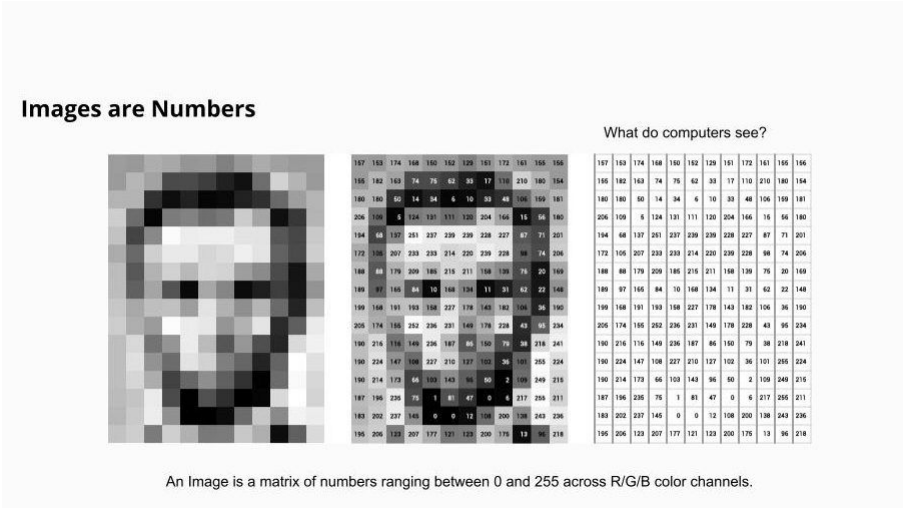
- Le but premier du domaine de la vision par ordinateur est de permettre aux ordinateurs d’extraire des patterns à partir de flux visuel, qui peut comprendre des tâches telles que :

  - Reconnaissance d'images
  - Détection d’objects
  - Segmentation d’images
  - Reconnaissance faciale
  - Reconnaissance des gestes
  - …

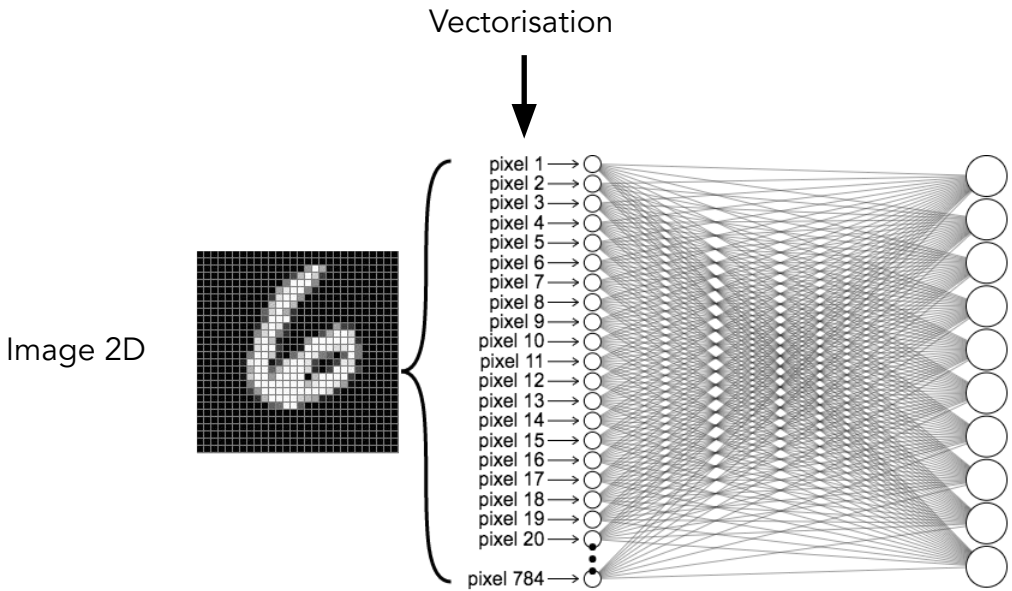
Qu’est-ce qu’une image ?

Une image est généralement définie comme une grille bidimensionnelle de valeurs de pixels. Chaque pixel représente un seul point de l'image et contient des informations sur la couleur de ce point. L'organisation des pixels dans cette grille est ce qui donne à une image son contenu visuel.





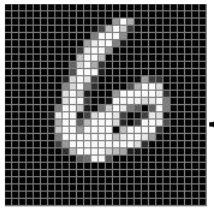
Réseau fully-connect pour la classification d'image

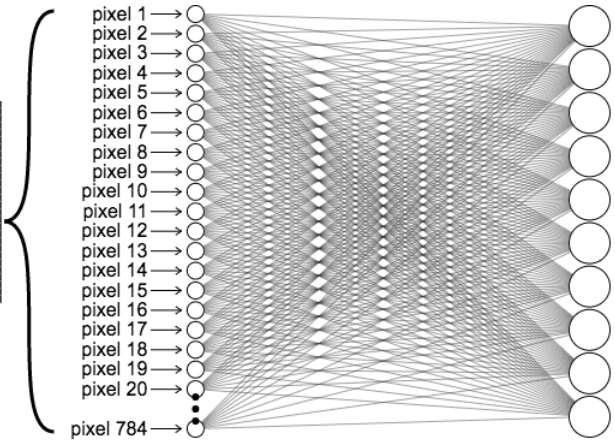


Réseau fully-connect pour la classification d'image

Vectorisation

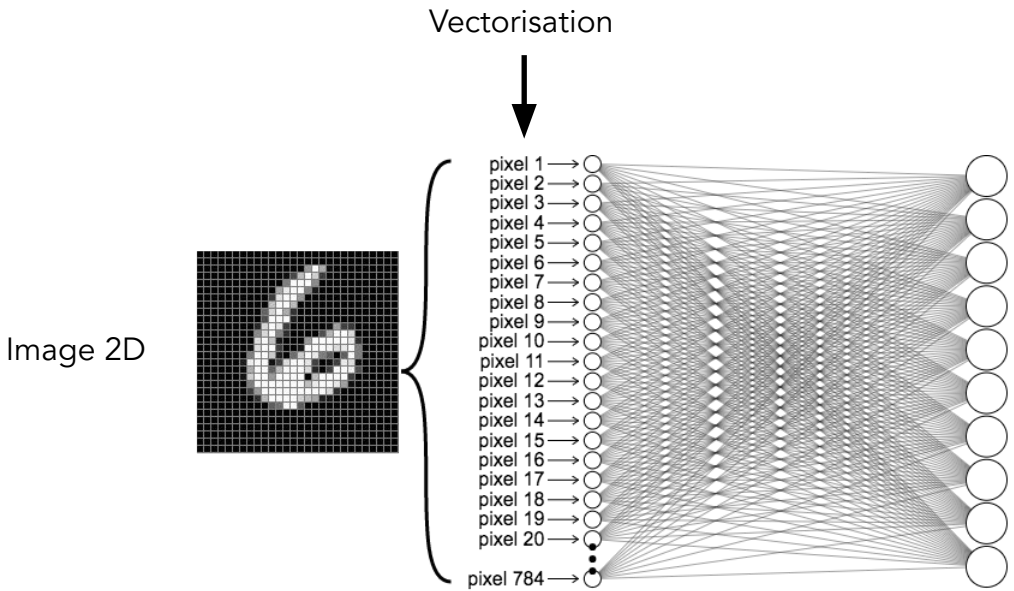
Image 2D





Problèmes :

Réseau fully-connect pour la classification d'image



Problèmes :

- Les neurones de la première couche sont connecté à tous les pixels —> trop de paramètre à apprendre !
- Pas d'information spatial
- Détruit concept de couleurs

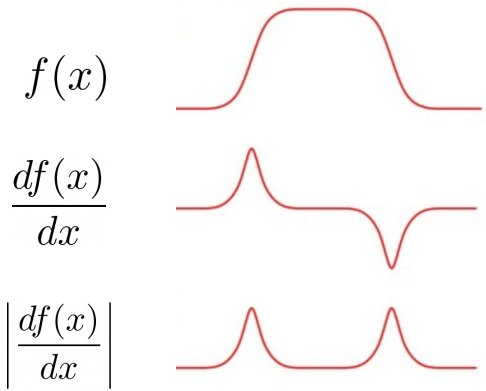
Détection de contours - motivation

- Beaucoup de détails important au sein d'une image sont contenu dans ses bords ou contours.
- Les contours jouent un rôle crucial dans la perception humaine lorsqu'il s'agit de traiter et de comprendre les images.
- La perception des contours est considérée comme l'une des premières étapes du traitement de l'information visuelle par le cerveau humain.
- Lorsque nous regardons une image ou une scène, notre cerveau commence par détecter les contours des objets et des formes, ce qui nous permet de distinguer les différentes parties de ce que nous voyons.
- Cette détection des contours est essentielle pour la reconnaissance des objets, la segmentation des éléments visuels, et la compréhension générale de notre environnement visuel.

Source :
https://en.wikipedia.org/wiki/Edge_detection

Détection de contours

Le but est de détecter les changements brusques de “valeurs” dans une image afin de capturer des événements importants et des modifications dans les propriétés sémantiques.



$$\left|\frac{df(x)}{dx}\right| > T \Longrightarrow \text{Bords}$$

Adapté de :
https://www.projectrhea.org/rhea/index.php/Edge_Detection_with_Gaussian_Blur

Opérateur de Sobel

- Quantification de la variation d'intensité au niveau des pixels par l'utilisation du gradient de l'image.
- Les zones au sein d'une image présentant une forte dérivée seront associé à des contours.

Opérateur de Sobel

- Quantification de la variation d'intensité au niveau des pixels par l'utilisation du gradient de l'image.
- Les zones au sein d'une image présentant une forte dérivée seront associé à des contours.

Opérateur de Sobel

- Quantification de la variation d'intensité au niveau des pixels par l'utilisation du gradient de l'image.
- Les zones au sein d'une image présentant une forte dérivée seront associé à des contours.

Considérons l'ensemble de pixel suivant représenté par des valeurs d'intensité allant de a à i :

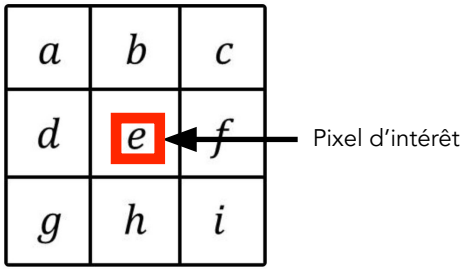
a	b	c
d	e	f
g	h	i

Adapté de :
https://nrsyed.com/2018/02/18/edge-detection-in-images-how-to-derive-the-sobel-operator/

Opérateur de Sobel

- Quantification de la variation d'intensité au niveau des pixels par l'utilisation du gradient de l'image.
- Les zones au sein d'une image présentant une forte dérivée seront associé à des contours.

Considérons l'ensemble de pixel suivant représenté par des valeurs d'intensité allant de a à i :

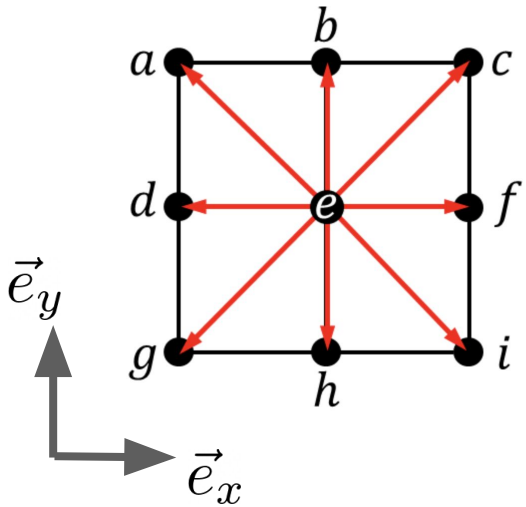


Distance de 1 entre chaque pixel adjacent

Adapté de :
https://nrsyed.com/2018/02/18/edge-detection-in-images-how-to-derive-the-sobel-operator/

Opérateur de Sobel

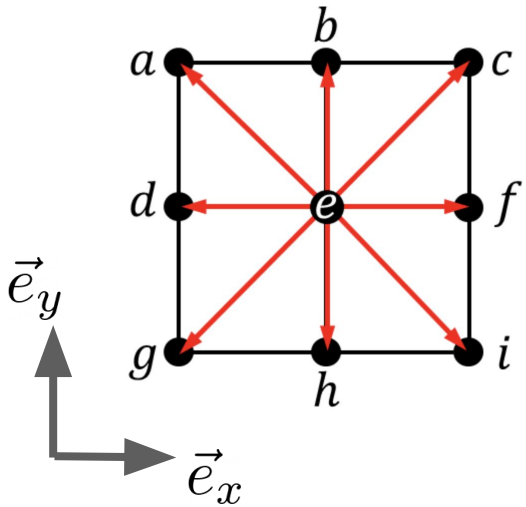
La dérivée du pixel central *e* est égal à la somme des dérivées directionnelles de ses 8 pixels voisin.



Adapté de :
https://nrsyed.com/2018/02/18/edge-detection-in-images-how-to-derive-the-sobel-operator/

Opérateur de Sobel

La dérivée du pixel central *e* est égal à la somme des dérivées directionnelles de ses 8 pixels voisin.



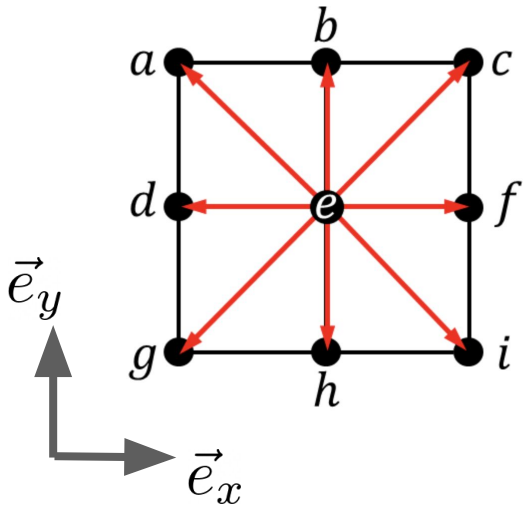
$$(f-e)\vec{e}_x \qquad (b-e)\vec{e}_y$$

$$-(d-e)\vec{e}_x \qquad -(h-e)\vec{e}_y$$

Adapté de :
https://nrsyed.com/2018/02/18/edge-detection-in-images-how-to-derive-the-sobel-operator/

Opérateur de Sobel

La dérivée du pixel central *e* est égal à la somme des dérivées directionnelles de ses 8 pixels voisin.



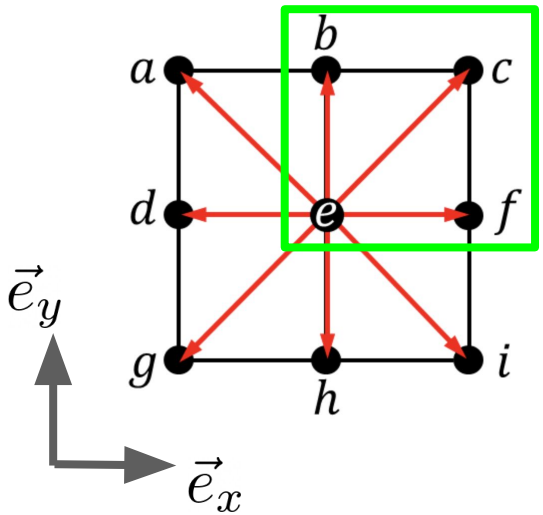
$$(f-e)\vec{e}_x \qquad (b-e)\vec{e}_y$$

$$-(d-e)\vec{e}_x \qquad -(h-e)\vec{e}_y$$

Adapté de :
https://nrsyed.com/2018/02/18/edge-detection-in-images-how-to-derive-the-sobel-operator/

Opérateur de Sobel

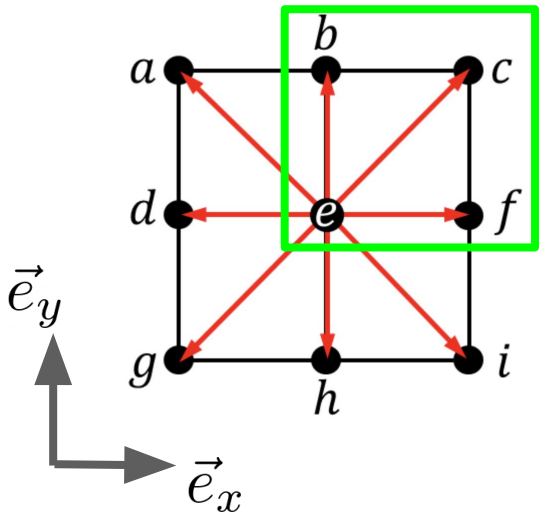
La dérivée du pixel central *e* est égal à la somme des dérivées directionnelles de ses 8 pixels voisin.

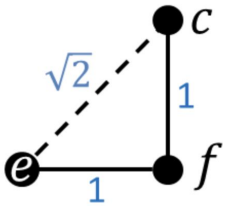


Adapté de :
https://nrsyed.com/2018/02/18/edge-detection-in-images-how-to-derive-the-sobel-operator/

Opérateur de Sobel

La dérivée du pixel central *e* est égal à la somme des dérivées directionnelles de ses 8 pixels voisin.

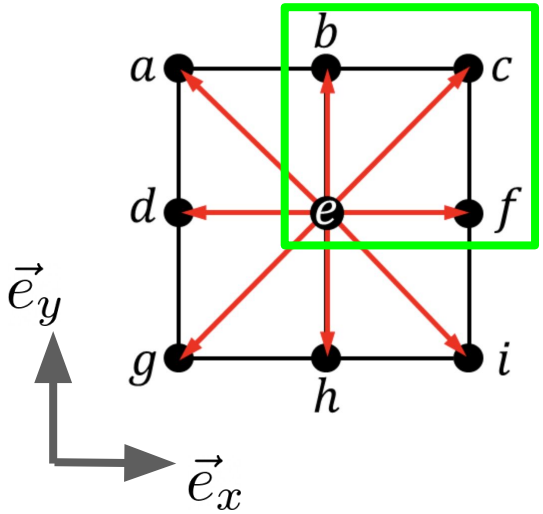


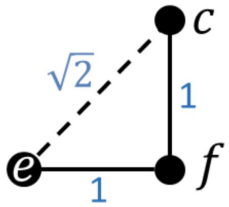


Adapté de :
https://nrsyed.com/2018/02/18/edge-detection-in-images-how-to-derive-the-sobel-operator/

Opérateur de Sobel

La dérivée du pixel central *e* est égal à la somme des dérivées directionnelles de ses 8 pixels voisin.





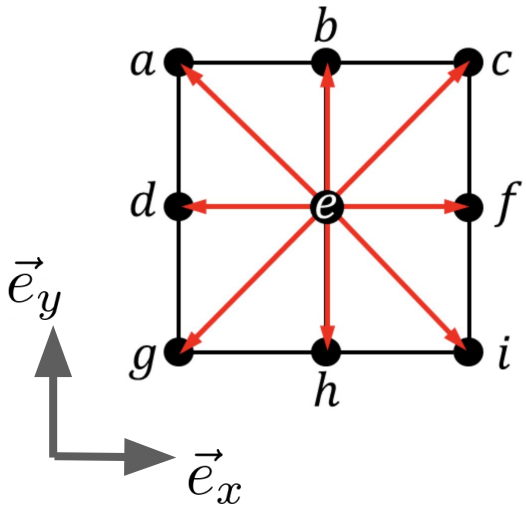
$$\frac{1}{\sqrt{(2)}}[(c-e)\cos(45^o)\vec{e}_x + (c-e)\sin(45^o)\vec{e}_y]$$

$$\frac{1}{2}(c-e)(\vec{e}_x + \vec{e}_y)$$

Adapté de :
https://nrsyed.com/2018/02/18/edge-detection-in-images-how-to-derive-the-sobel-operator/

Opérateur de Sobel

La dérivée du pixel central *e* est égal à la somme des dérivées directionnelles de ses 8 pixels voisin.

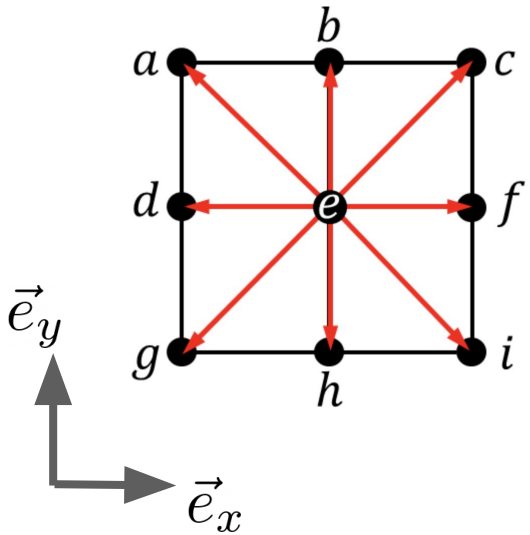


$$[(f-d)+\frac{1}{2}(-a+c-g+i)]\vec{e}_x$$

$$[(b-h)+\frac{1}{2}(a+c-g-i)]\vec{e}_y$$

Adapté de :
https://nrsyed.com/2018/02/18/edge-detection-in-images-how-to-derive-the-sobel-operator/

Opérateur de Sobel



$$G_x = \frac{1}{2}\begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix}$$

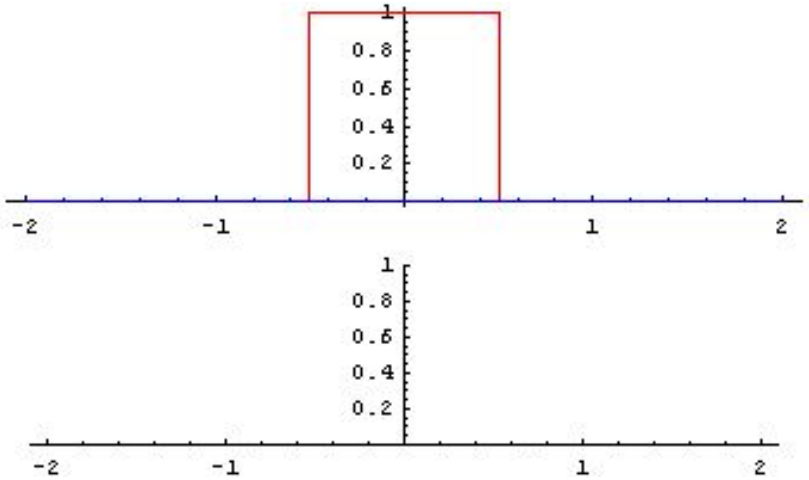
$$G_y = \frac{1}{2}\begin{bmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix}$$

Adapté de :
https://nrsyed.com/2018/02/18/edge-detection-in-images-how-to-derive-the-sobel-operator/

L’opération de Convolution

L’opération de Convolution

Une convolution est une opération mathématique qui généralise le concept de moyenne glissante



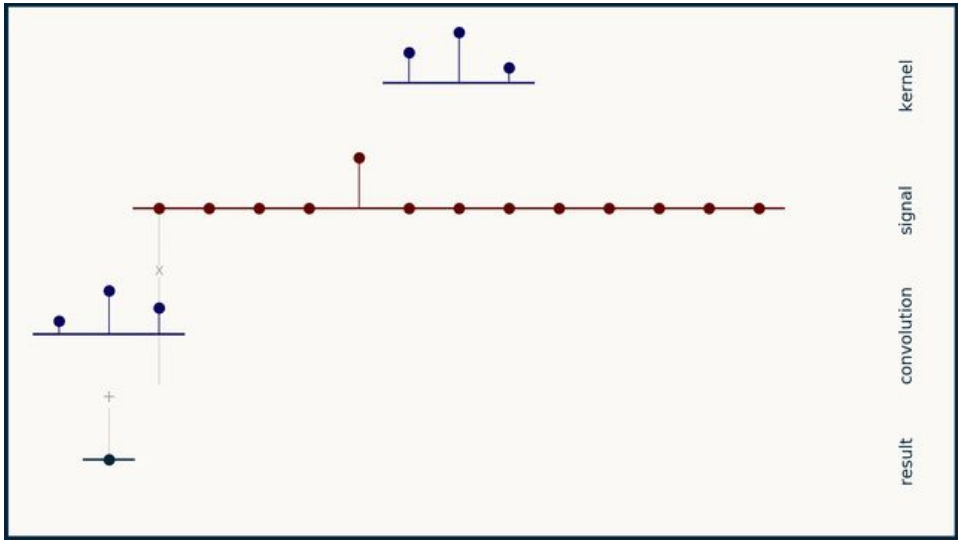
Continu:

$$(f \star g)(x) = \int f(x-t)g(t)dt$$

Discret:

L’opération de Convolution

Une convolution est une opération mathématique qui généralise le concept de moyenne glissante



Continu:

$$(f \star g)(x) = \int f(x-t)g(t)dt$$

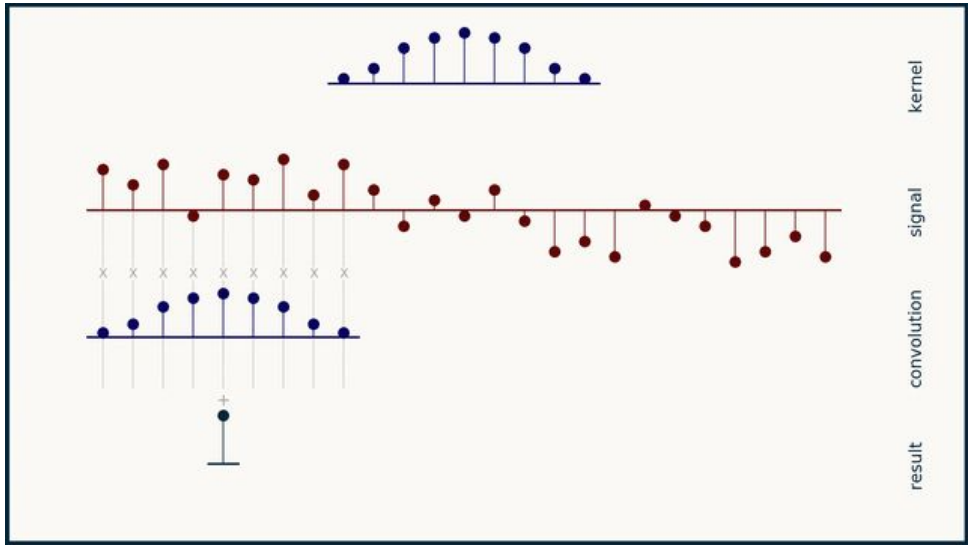
Discret:

$$(f \star g)[n] = \sum f[m]g[n-m]$$

https://e2eml.school/convolution_one_d.html

L’opération de Convolution

Une convolution est une opération mathématique qui généralise le concept de moyenne glissante



Continu:

$$(f \star g)(x) = \int f(x-t)g(t)dt$$

Discret:

$$(f \star g)[n] = \sum f[m]g[n-m]$$

https://e2eml.school/convolution_one_d.html

L’opération de Convolution

Exemple de convolution d’une image 5x5 par un filtre 3x3

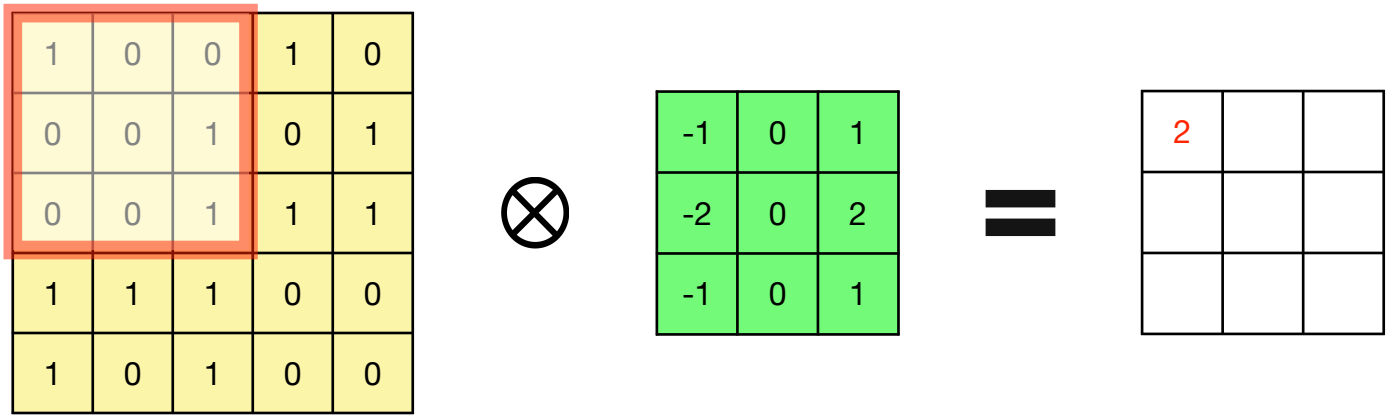
1	0	0	1	0
0	0	1	0	1
0	0	1	1	1
1	1	1	0	0
1	0	1	0	0

Image 5 x 5 pixels

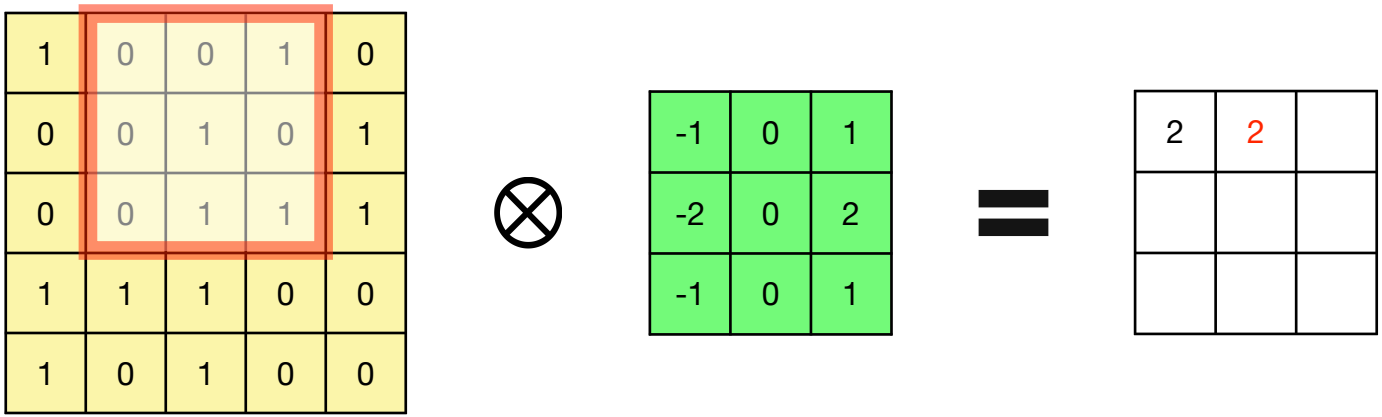
-1	0	1
-2	0	2
-1	0	1

Filtre de Sobel horizontal

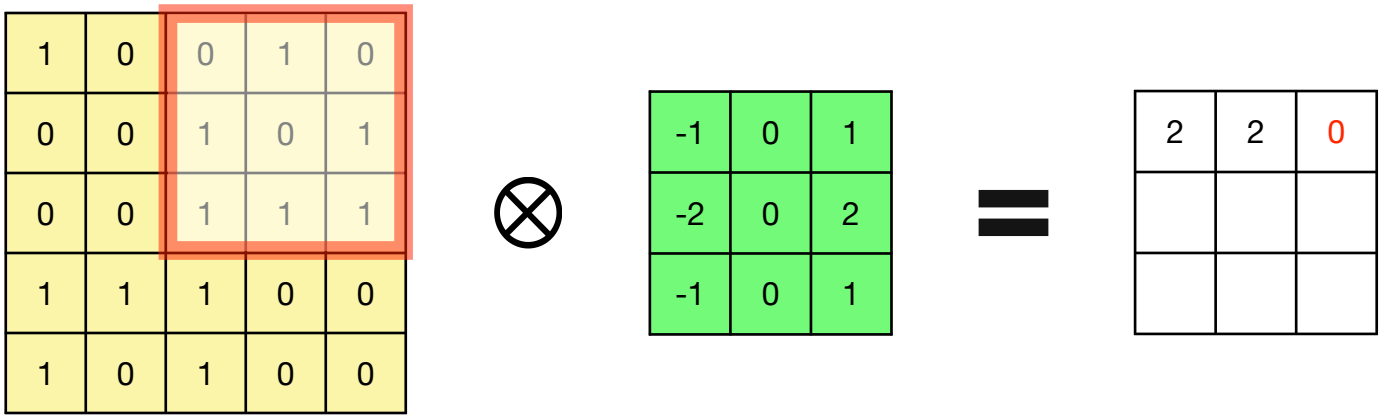
L’opération de Convolution



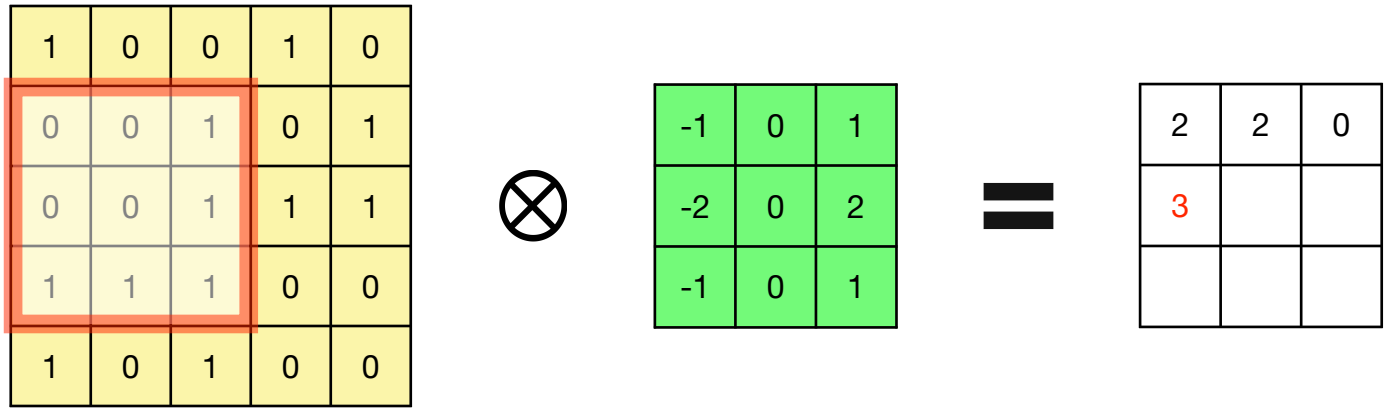
L’opération de Convolution



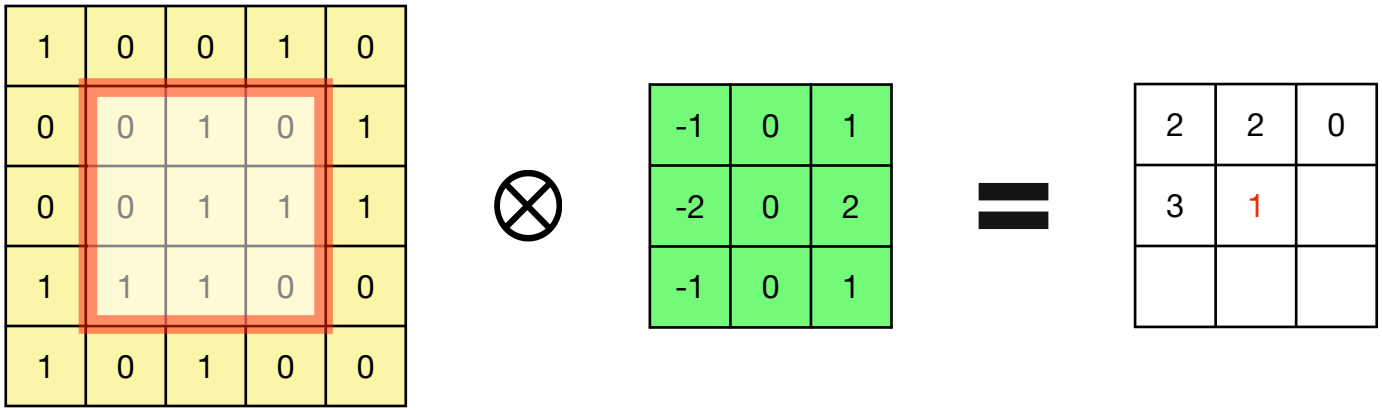
L’opération de Convolution



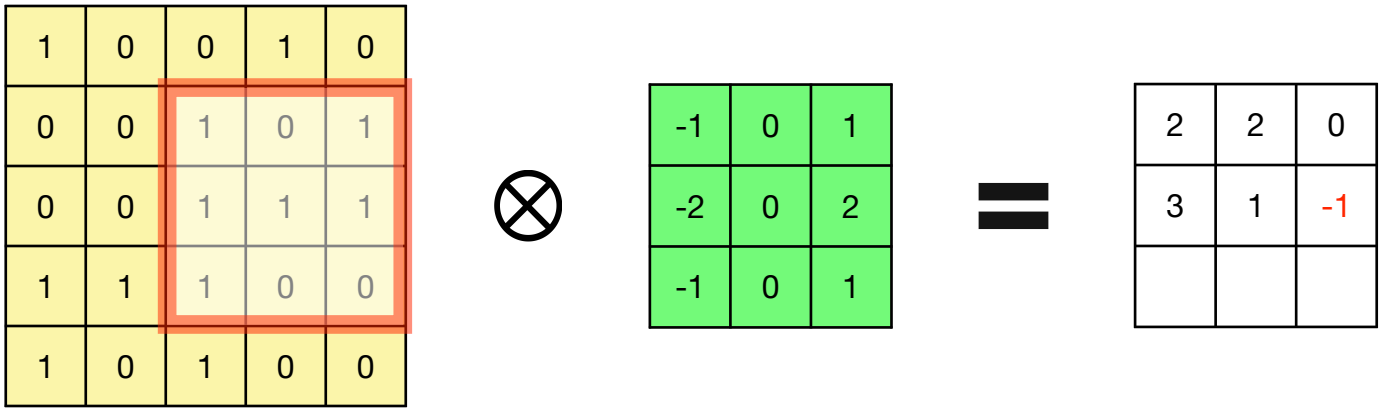
L’opération de Convolution



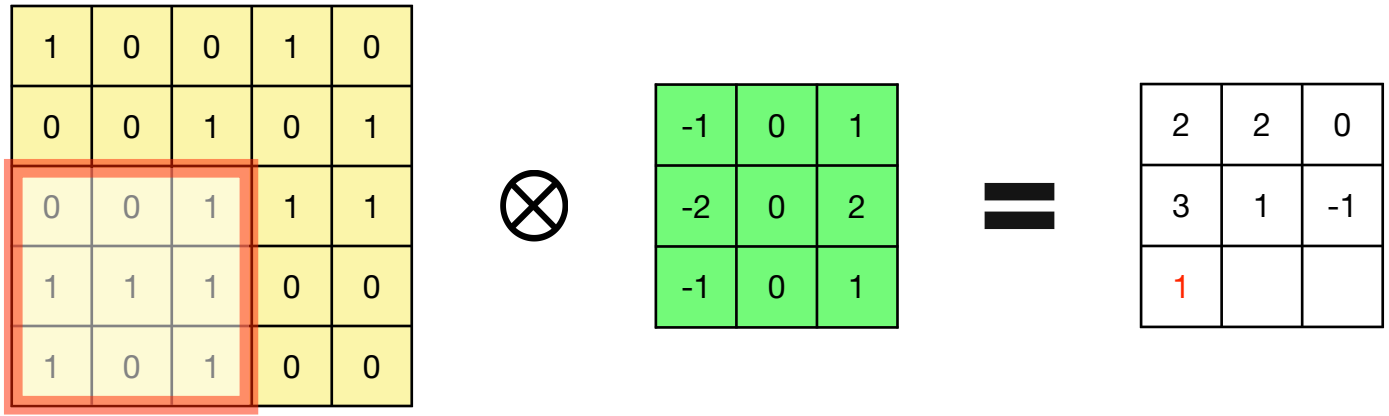
L’opération de Convolution



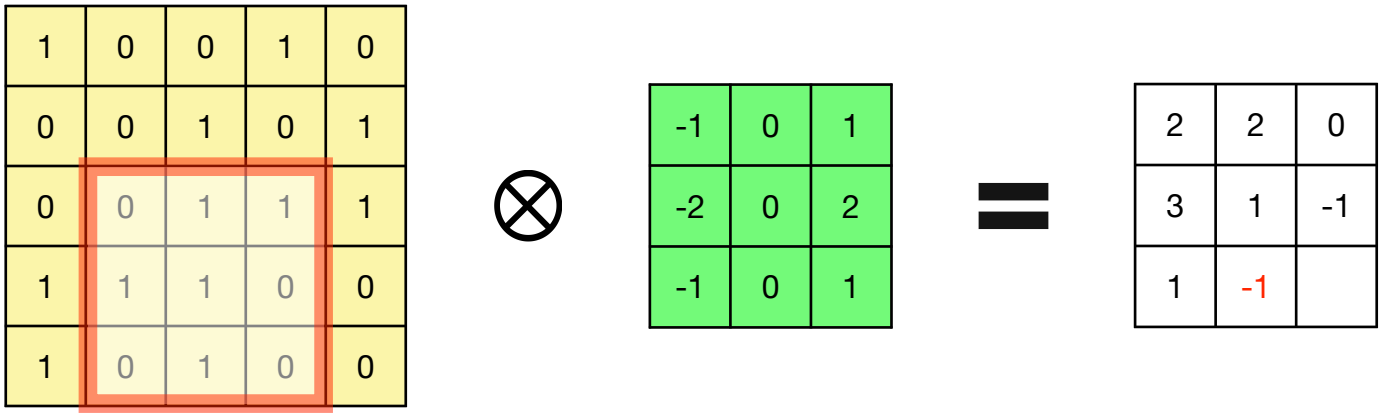
L’opération de Convolution



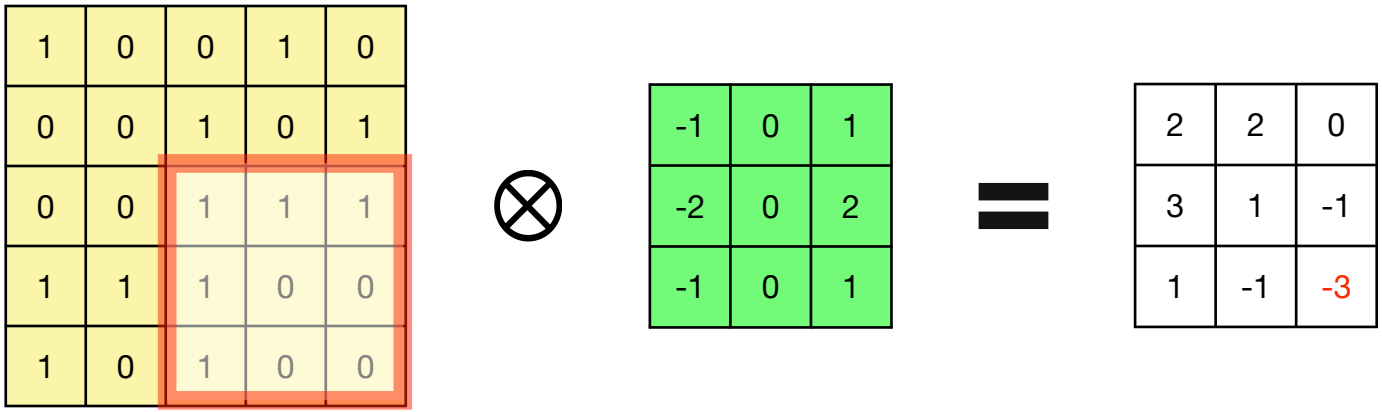
L’opération de Convolution



L’opération de Convolution



L’opération de Convolution



Filtre de Sobel

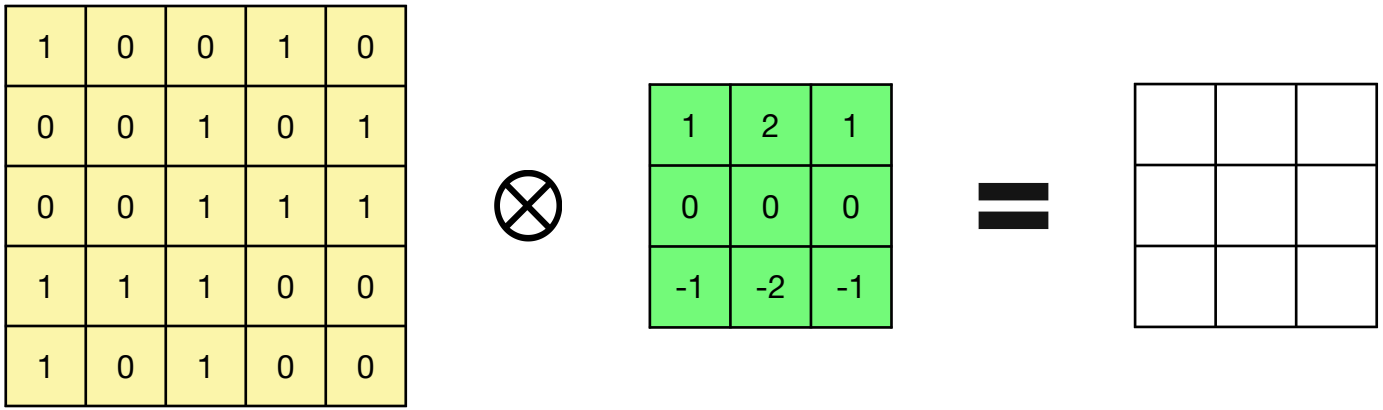
Original image

Sobel X

Sobel Y

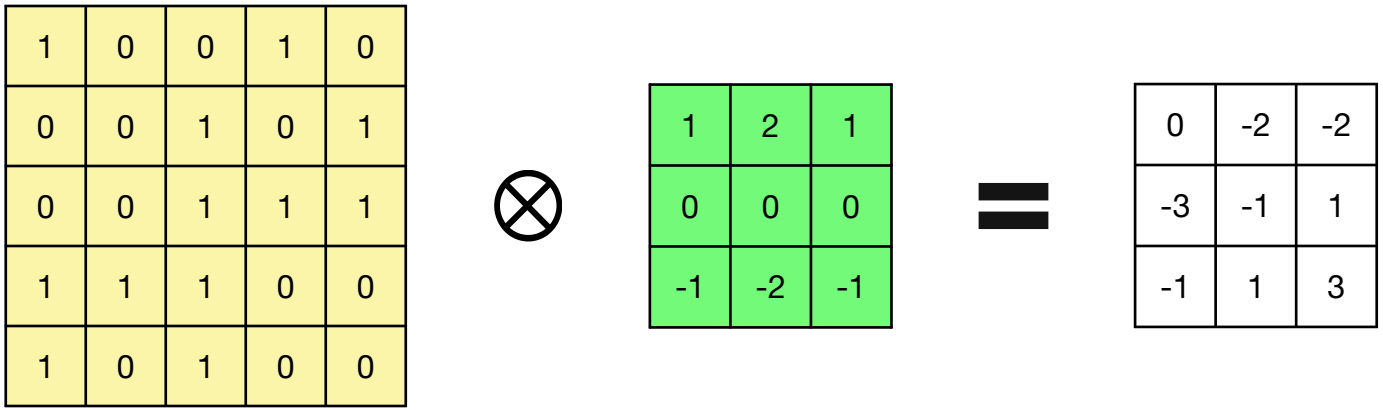
L’opération de Convolution

Exercice: Calcul de convolution



L’opération de Convolution

Exercice: Calcul de convolution



D’autre type de filtres

Original	*Gaussian Blur*	*Sharpen*	*Edge Detection*
$\begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix}$	$\frac{1}{16}\begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix}$	$\begin{bmatrix} 0 & -1 & 0 \\ -1 & 5 & -1 \\ 0 & -1 & 0 \end{bmatrix}$	$\begin{bmatrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{bmatrix}$
			

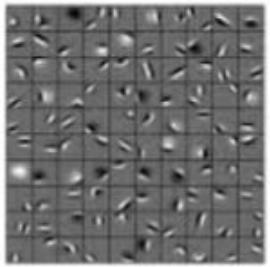
Exemple de Convolution

https://setosa.io/ev/image-kernels/

Extraction de caractéristiques

- Des filtres pourraient être utilisés pour extraire diverses caractéristiques (features) à partir d'images.
- Ces caractéristiques peuvent être simples, comme les bords et les contours, ou plus complexes, comme un nez ou même des structures de visages.

Features bas niveau

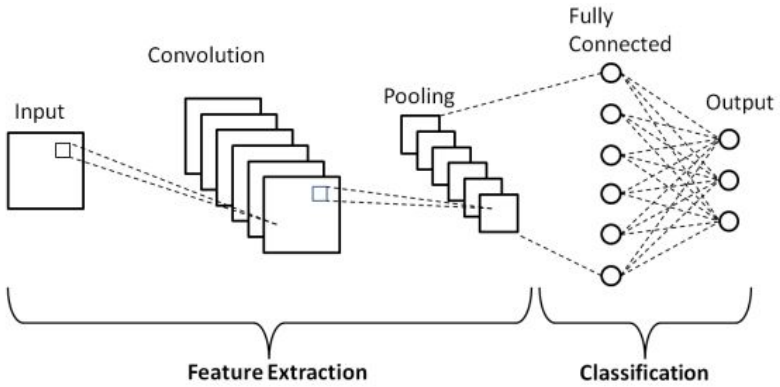


Features de niveau moyen

Features de haut niveau

Réseau de neurones à Convolution (CNNs)

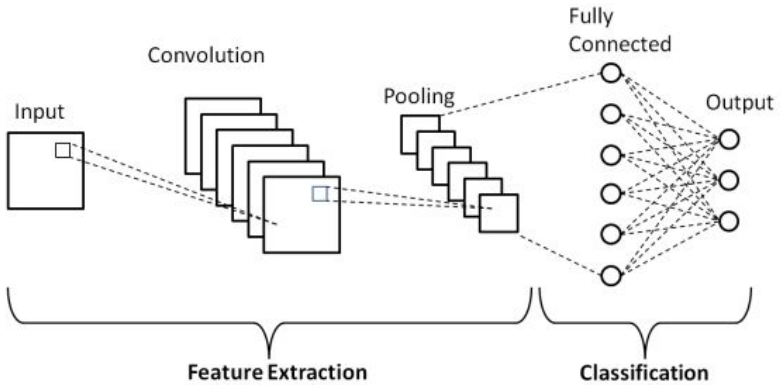
- Une architecture de type CNN est composée d'une succession de couches distinctes transformant un volume d'entrée en volume de sortie



Source image :
https://www.researchgate.net/publication/336805909_A_High-Accuracy_Model_Average_Ensemble_of_Convolutional_Neural_Networks_for_Classification_of_Cloud_Image_Patches_on_Small_Datasets

Réseau de neurones à Convolution (CNNs)

- Une architecture de type CNN est composée d'une succession de couches distinctes transformant un volume d'entrée en volume de sortie
- Les différentes types de couches sont :
  - Fully-Connected
  - Convolution : génération de features maps à partir de filtres
  - Pooling : opération de downsampling



Source image :
https://www.researchgate.net/publication/336805909_A_High-Accuracy_Model_Average_Ensemble_of_Convolutional_Neural_Networks_for_Classification_of_Cloud_Image_Patches_on_Small_Datasets

Couche de Convolution

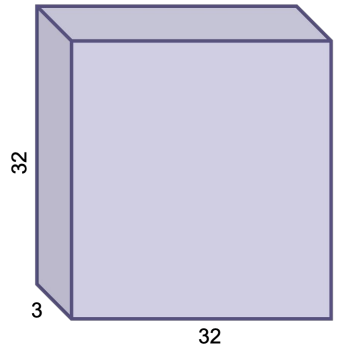
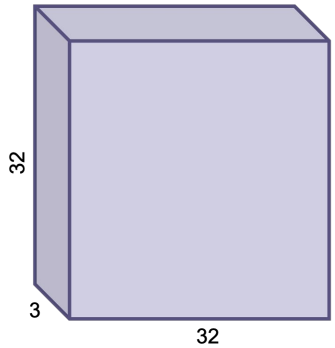
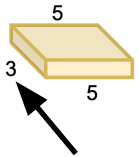


Image RGB de résolution
32x32

Couche de Convolution

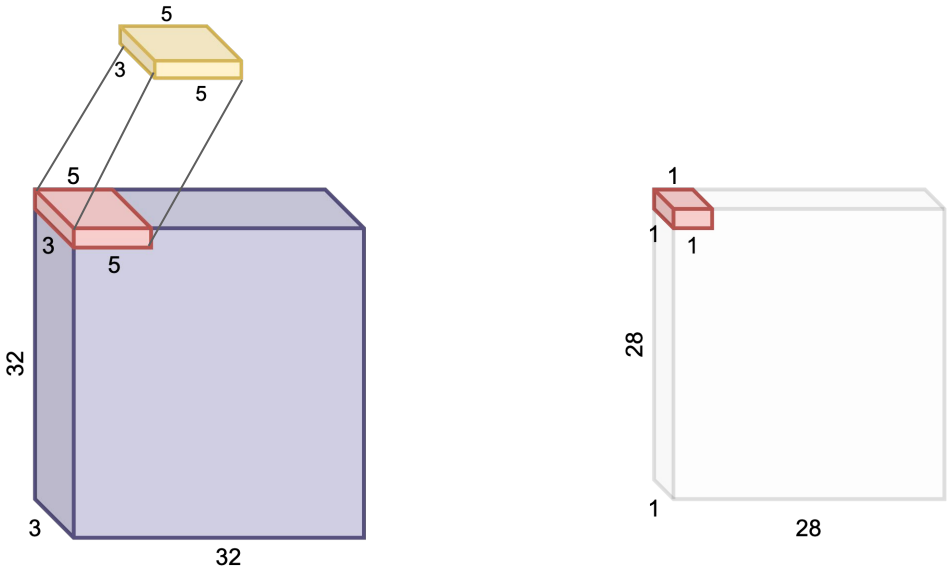




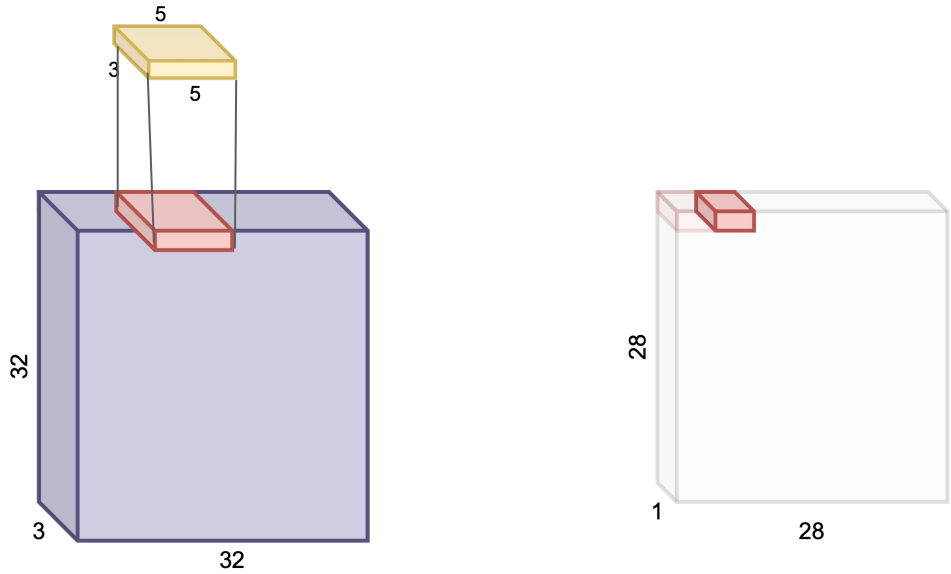
Filtre 1 (F1)

Profondeur d'un filtre est toujours égale à la profondeur de l'image ou du **feature maps**

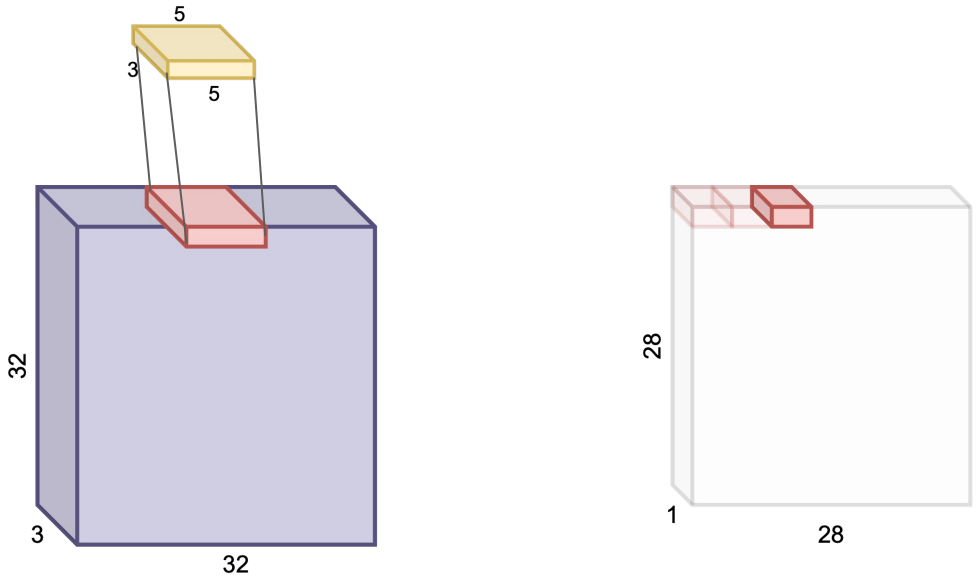
Couche de Convolution



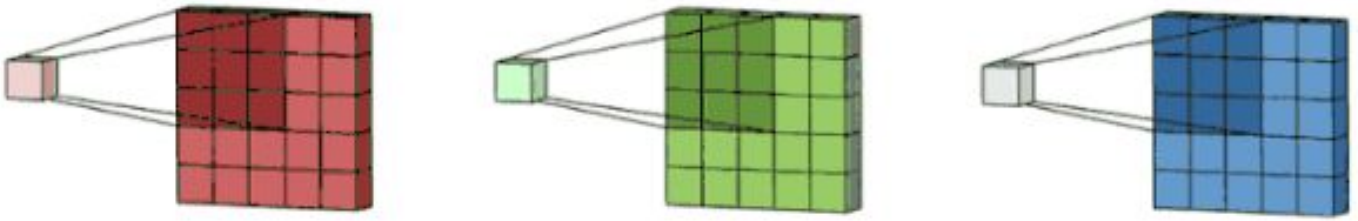
Couche de Convolution



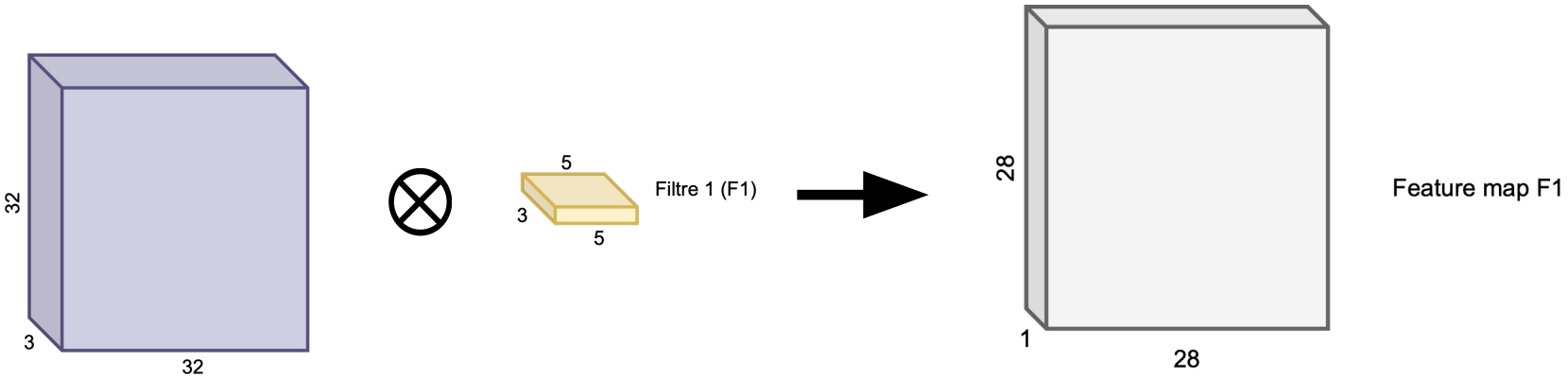
Couche de Convolution



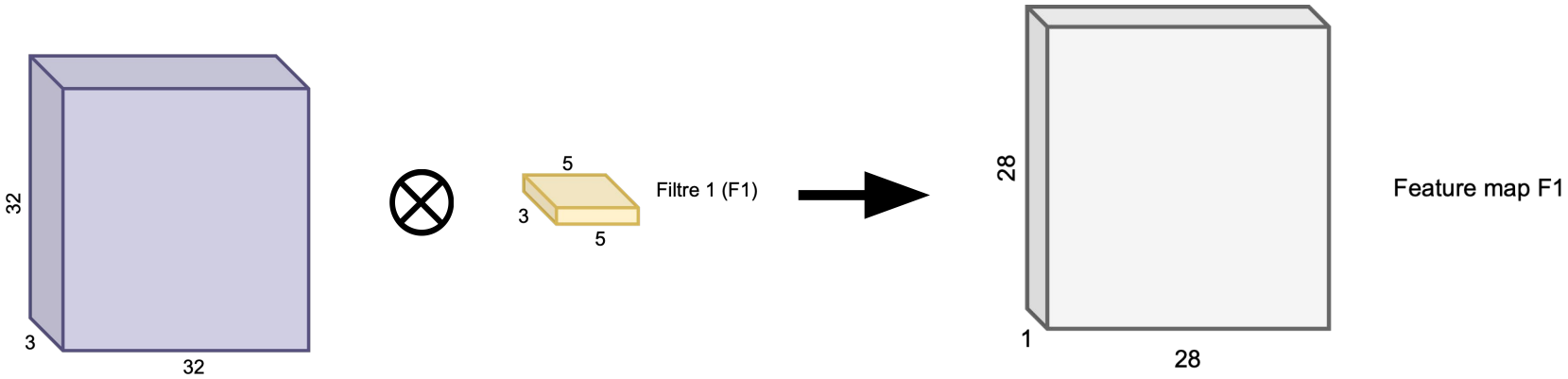
Couche de Convolution



Couche de Convolution



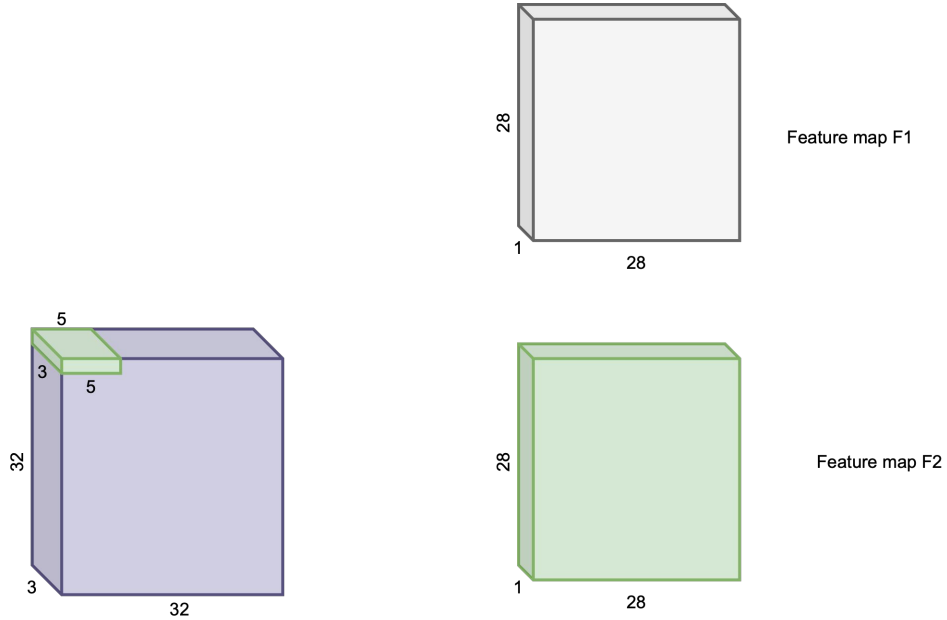
Couche de Convolution



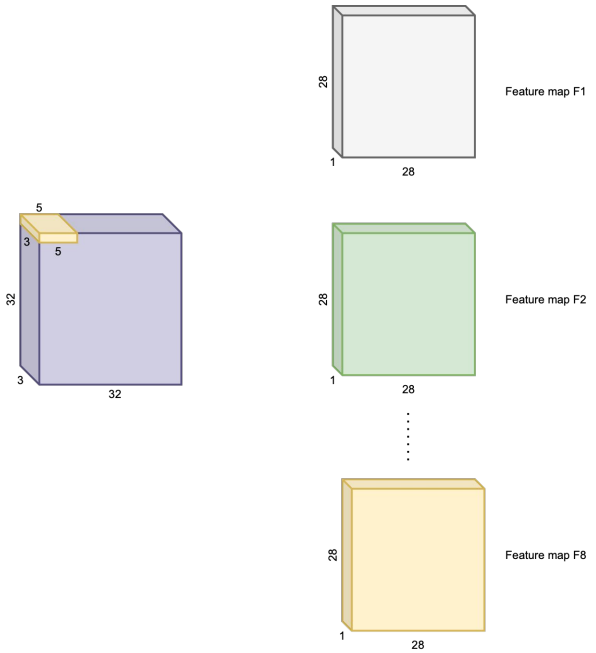
$$H_{out} = H_{in} - \text{largeur du filtre} + 1$$

$$W_{out} = W_{in} - \text{largeur du filtre} + 1$$

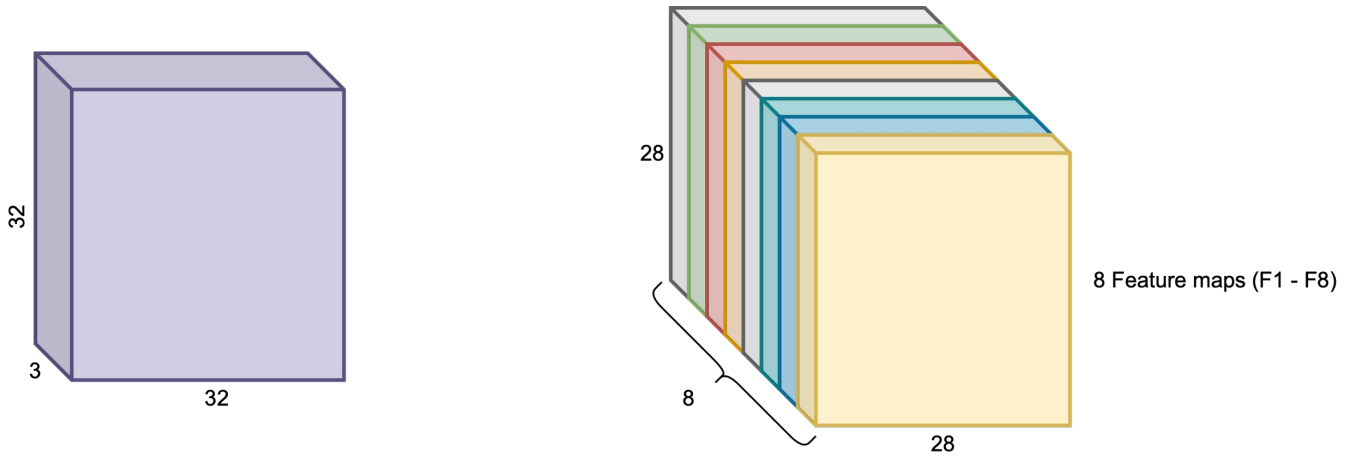
Couche de Convolution



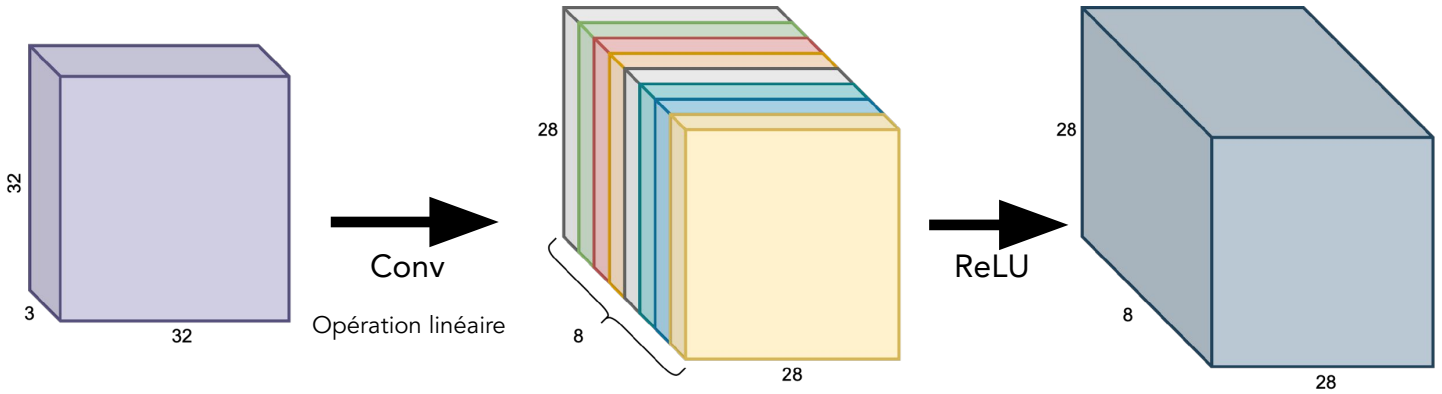
Couche de Convolution



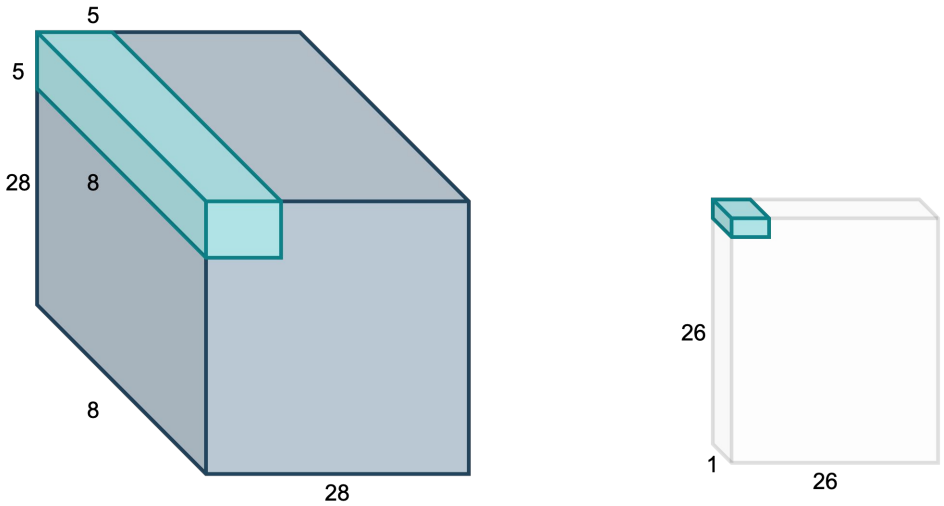
Couche de Convolution



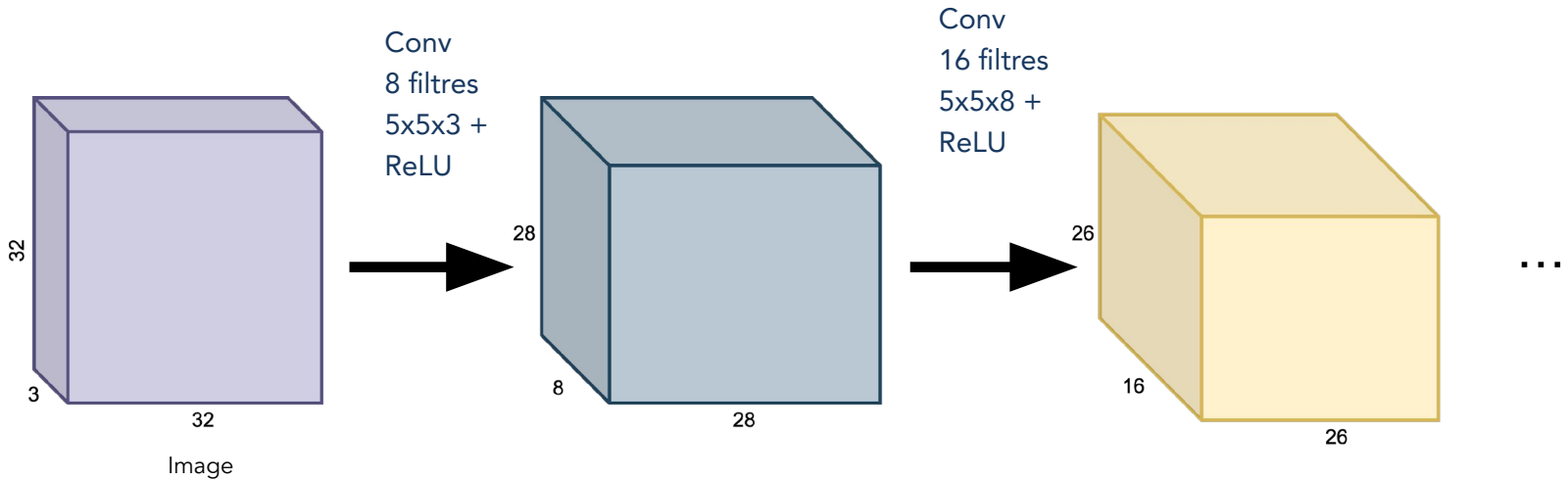
Non-Linéarité



Convolution à partir de feature maps



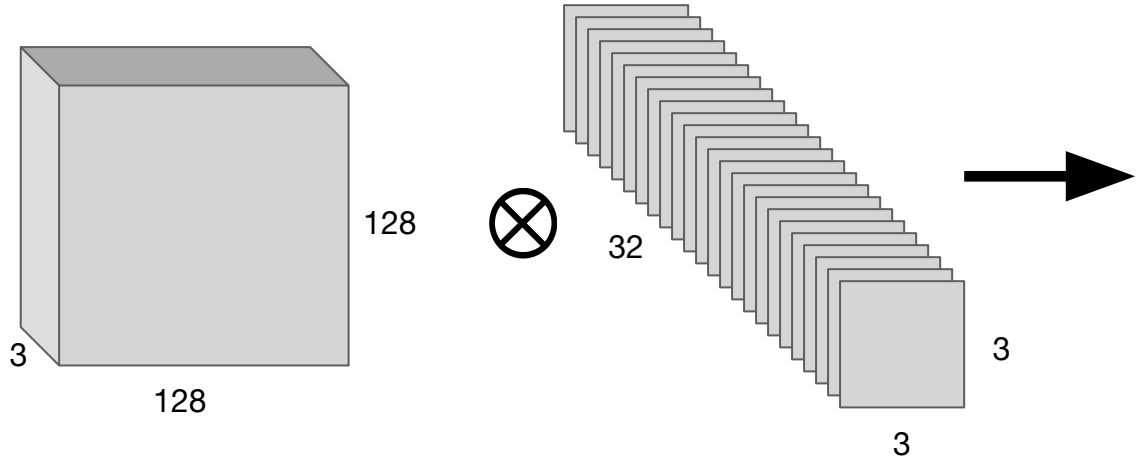
Convolution Block



Convolution Block

Exercice: Calcul de taille

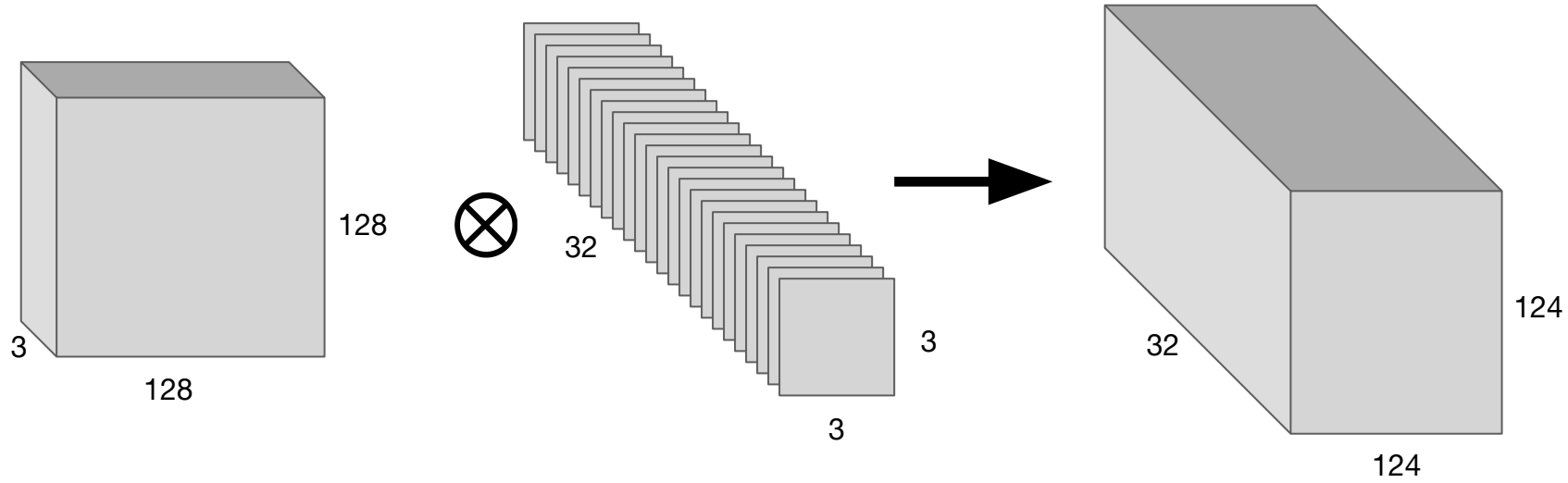
On considère une image en couleur de taille 128x128. On lui applique 32 filtres de taille 3x3. On obtient des features maps de taille intermédiaire. Quelle est la nouvelle taille obtenue ?



Convolution Block

Exercice: Calcul de taille

On considère une image en couleur de taille 128x128. On lui applique 32 filtres de taille 3x3. On obtient des features maps de taille intermédiaire. Quelle est la nouvelle taille obtenue ?



Forces

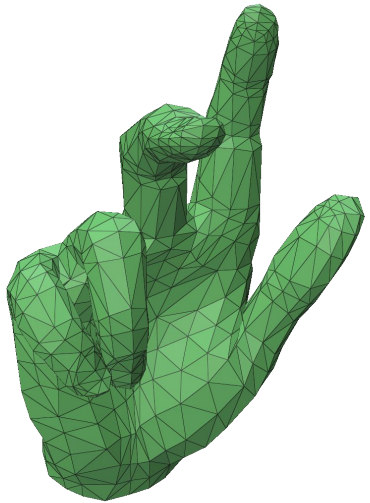
- Connectivité locale avec pixels adjacent. Exploitation de la structure spatial.
- Faible nombre de paramètres comparé à un réseau fully-connected.
- Les CNNs sont excellents pour apprendre automatiquement des caractéristiques hiérarchiques à partir de données d'entrée.

Faiblesse

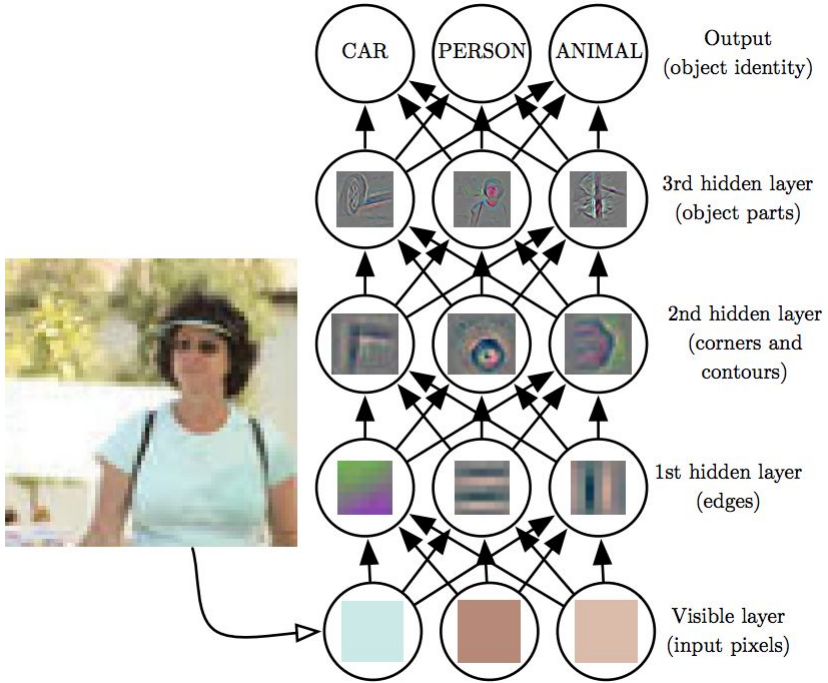
- Sensibilité aux translations / rotations
- Capture les informations locales mais peut être limité pour le contexte globale
- Difficultés d’interprétation (dangers de certains biais liés aux faciès par exemple)
- Difficultés à généraliser à d’autres types de données (3D par exemple)

Faiblesse

- Sensibilité aux translations / rotations
- Capture les informations locales mais peut être limité pour le contexte globale
- Difficultés d’interprétation (dangers de certains biais liés aux faciès par exemple)
- Difficultés à généraliser à d’autres types de données (3D par exemple)



Apprentissage de Caractéristiques Hiérarchiques



Source image :
https://www.deeplearningbook.org

Visualisation des features

https://www.youtube.com/watch?v=AgkfIQ4IGaM

Zéro Padding

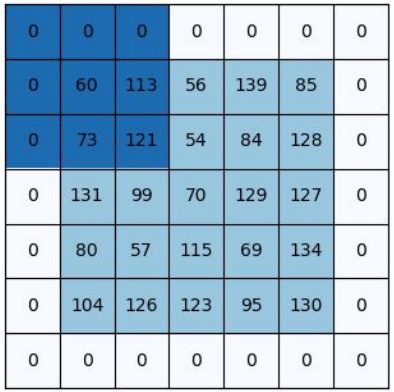
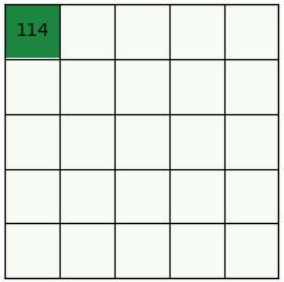


Image originale 5x5

Kernel

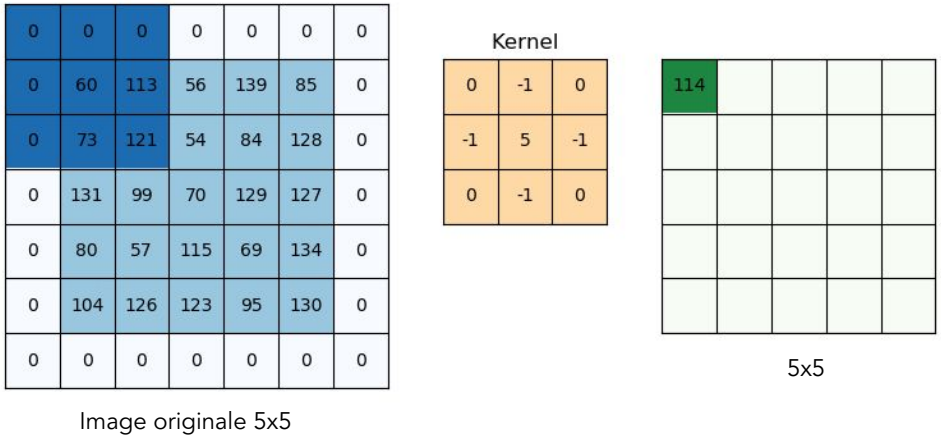
0	-1	0
-1	5	-1
0	-1	0



5x5

Source image :
https://medium.com/@draj0718/zero-padding-in-convolutional-neural-networks-bf1410438e99

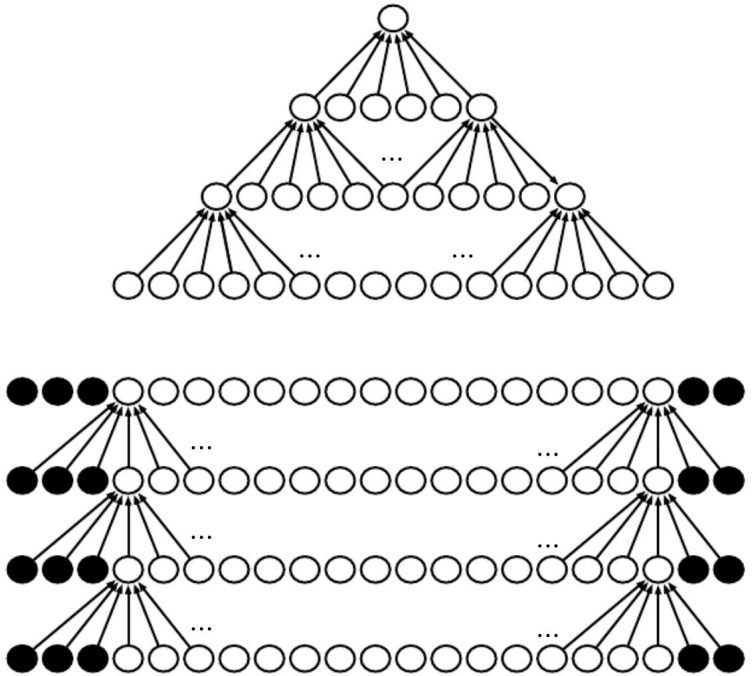
Zéro Padding



- Permet de préserver la taille en terme de hauteur et largeur au niveau des features map.
- Bords sont moins informatifs.

Source image :
https://medium.com/@draj0718/zero-padding-in-convolutional-neural-networks-bf1410438e99

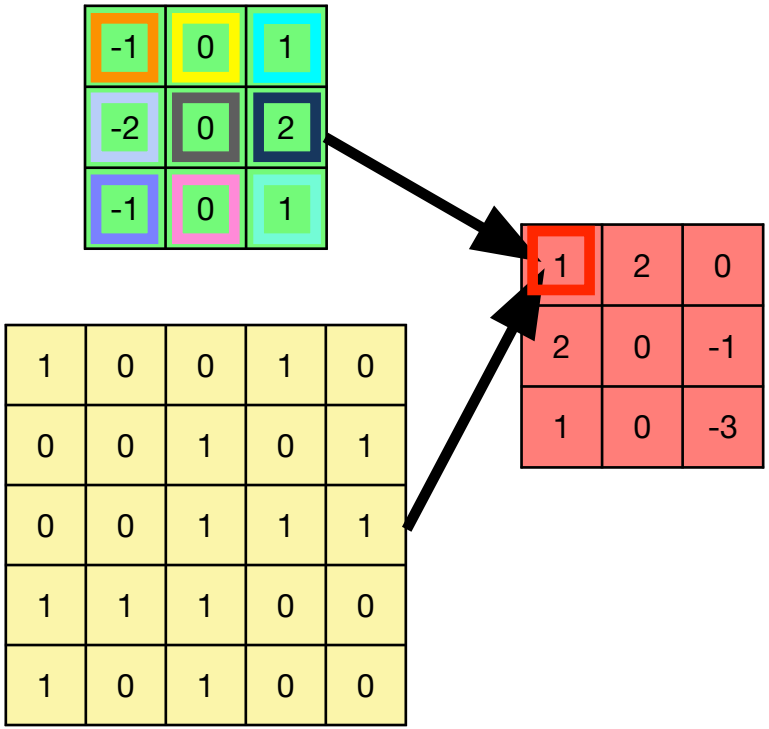
Zéro Padding

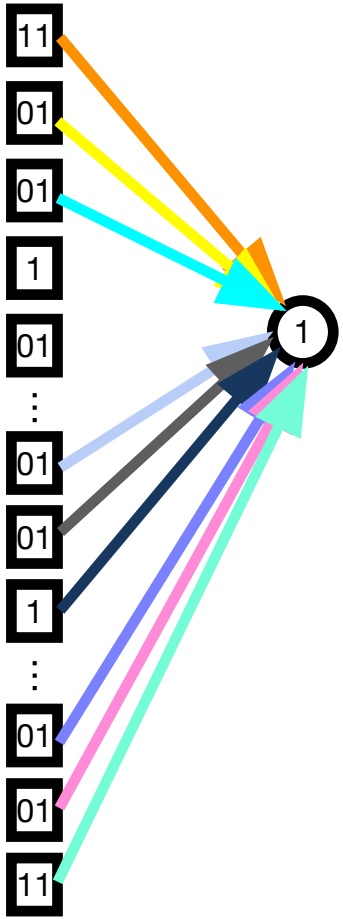


Effet du padding sur la taille du réseau de neurones

Source image :
https://www.deeplearningbook.org

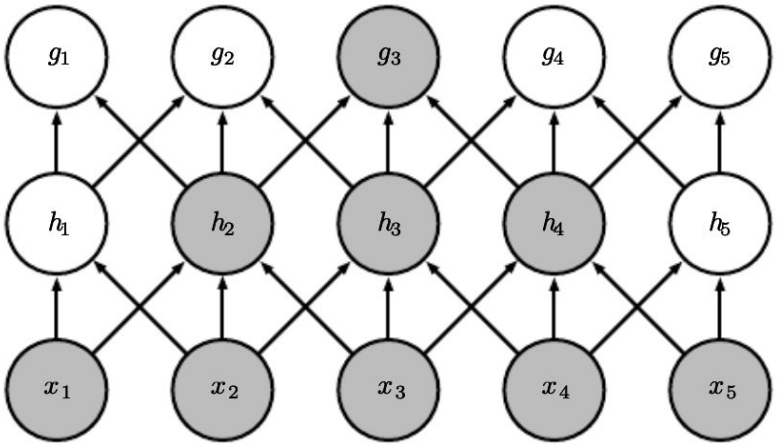
Point de vue d'un neurone





Champs récepteur

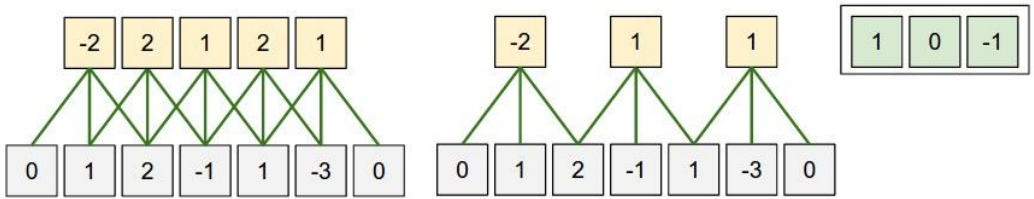
- Le champs récepteur des activations au sein des couches profondes d'un CNN est plus grand que celui des activations au sein des premières couches.
- Cela augmente avec l'utilisation d'un « strides convolution » ou de pooling.



Source image :
https://www.deeplearningbook.org

Pas (stride)

- Le pas ou stride fait référence à la taille de l'étape qu'un filtre prend lorsqu'il se déplace sur des données d'entrée ou features maps pendant la convolution.
- Des pas plus grands réduisent les dimensions spatiales.
- Joue un rôle important dans le contrôle du champ récepteur du réseau.



Source image :
https://cs231n.github.io/convolutional-networks/

L’opération de Convolution

Exemple de convolution d’une image 5x5 par un filtre 3x3 avec un stride de 2

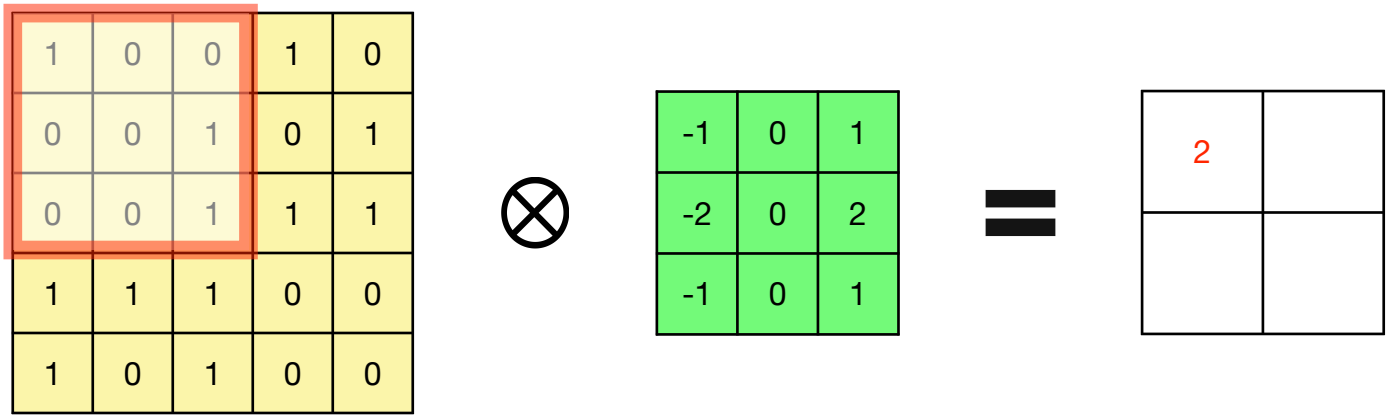
1	0	0	1	0
0	0	1	0	1
0	0	1	1	1
1	1	1	0	0
1	0	1	0	0

Image 5 x 5 pixels

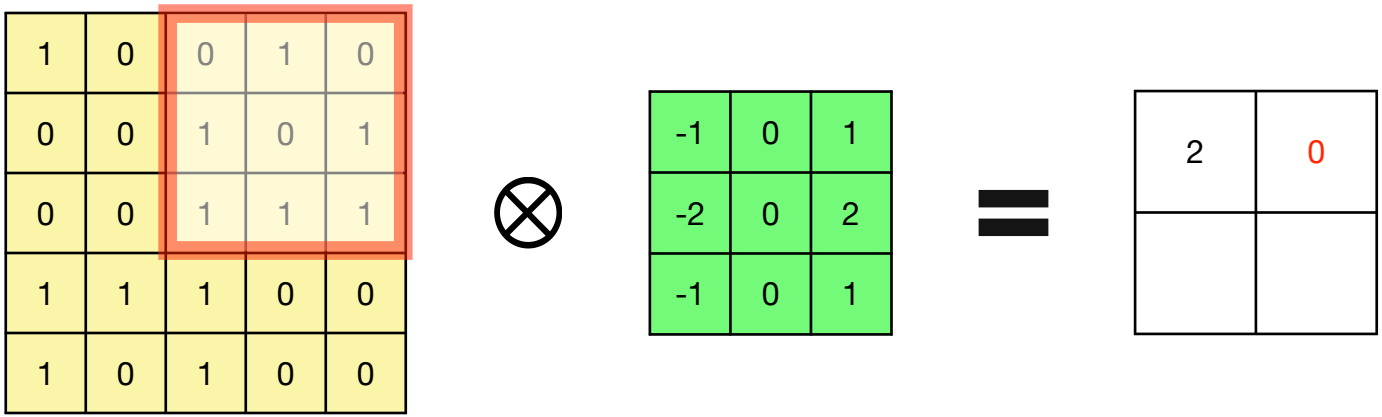
-1	0	1
-2	0	2
-1	0	1

Filtre de Sobel horizontal

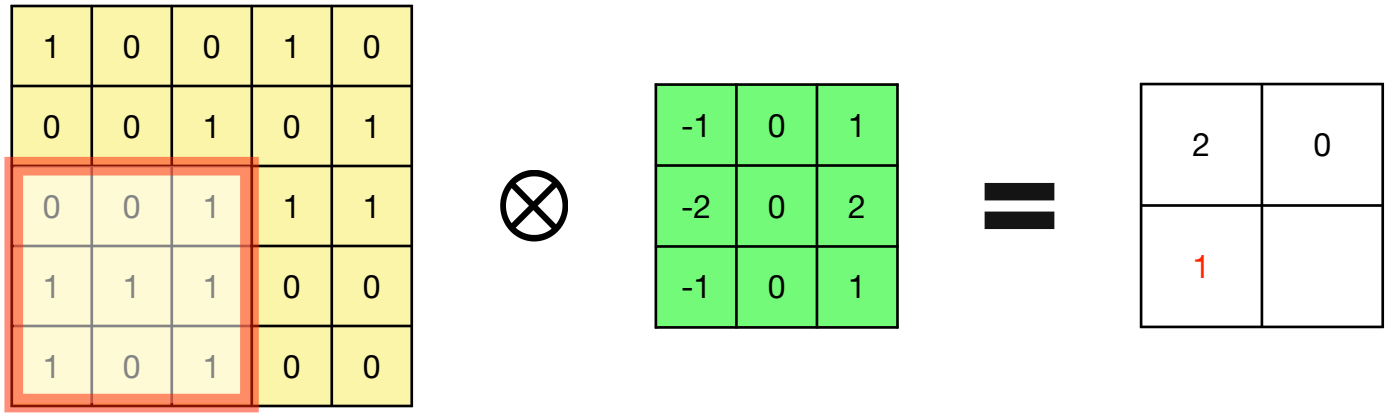
L’opération de Convolution



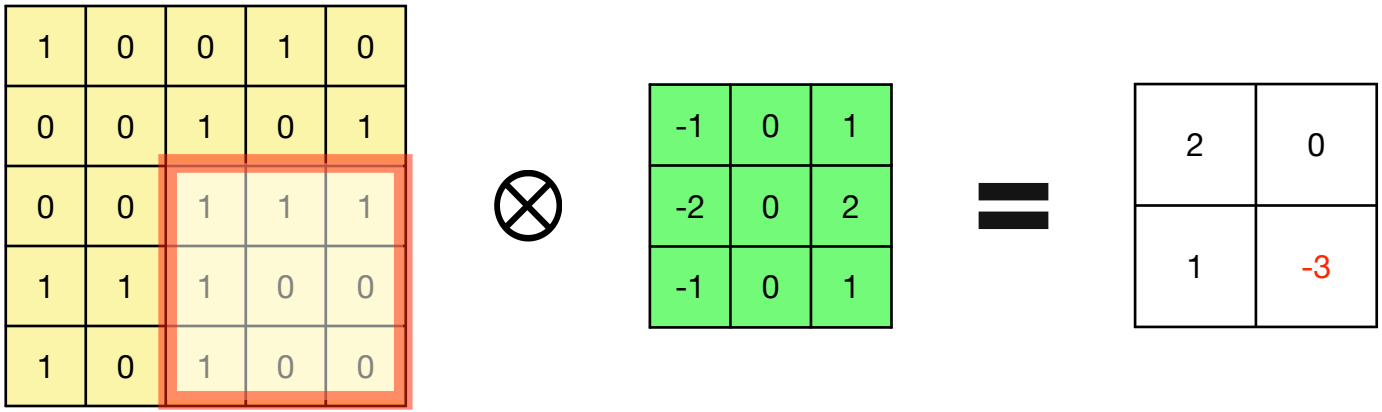
L’opération de Convolution



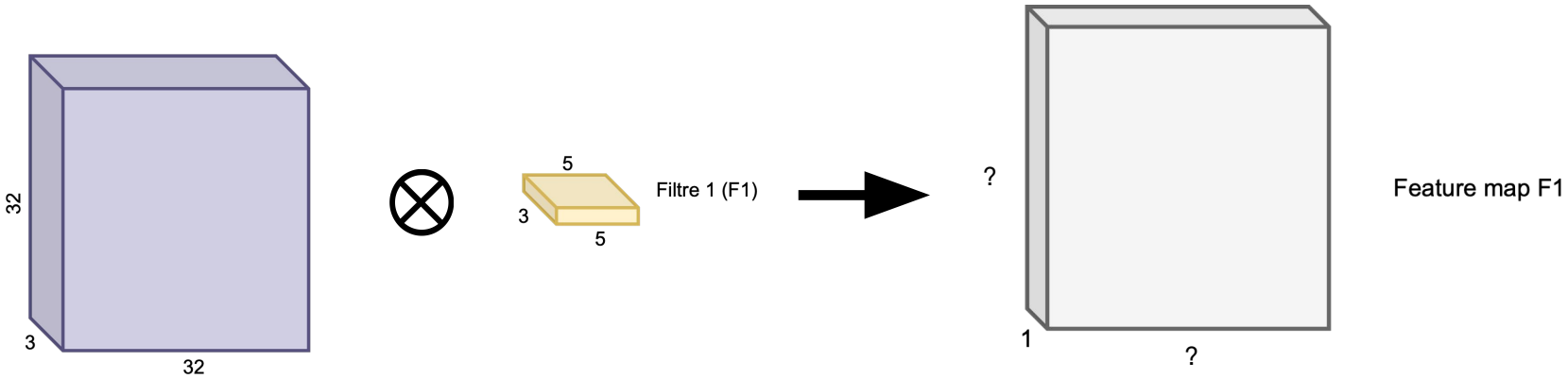
L’opération de Convolution



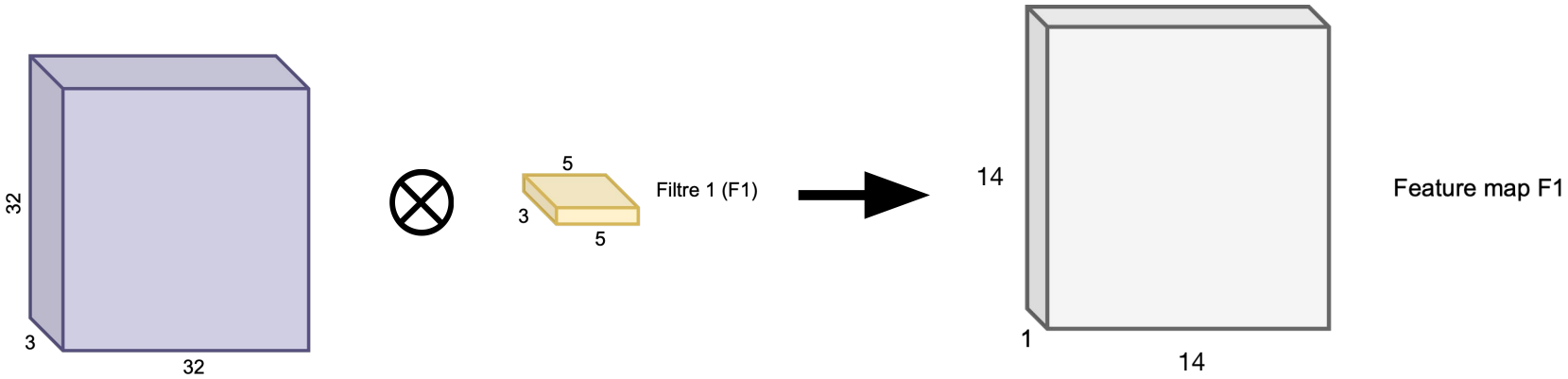
L’opération de Convolution



Taille features map avec stride



Taille features map avec stride

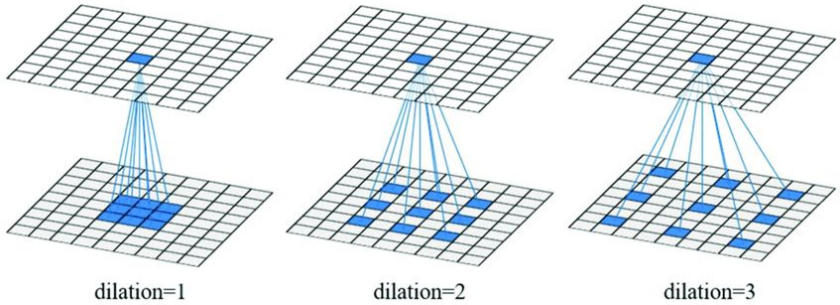


$$H_{out} = \frac{H_{in} - \text{hauteur du filtre}}{\text{stride}} + 1$$

$$W_{out} = \frac{W_{in} - \text{largeur du filtre}}{\text{stride}} + 1$$

Dilated Convolution

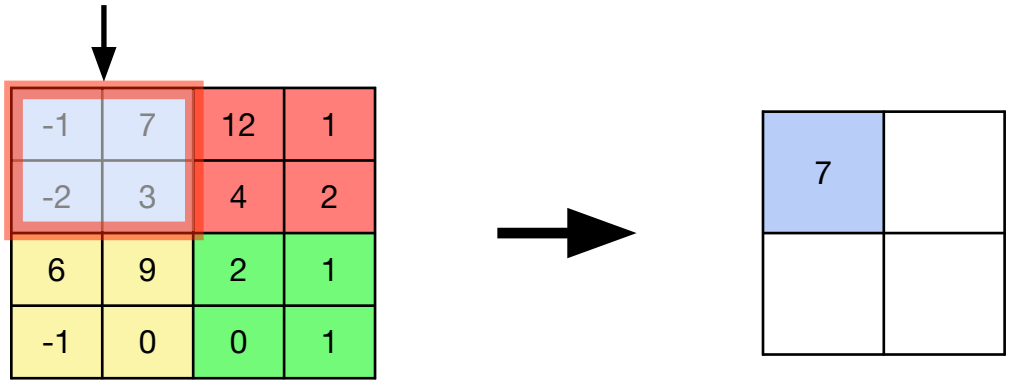
- La convolution dilatée peut capturer des caractéristiques à différentes échelles spatiales sans augmenter le nombre de paramètres.
- Le champ récepteur en convolution dilatée croît de manière exponentielle avec le nombre de couches de dilation. Cela permet au réseau d'avoir un champ récepteur plus large sans utiliser des tailles de filtre excessivement grandes.



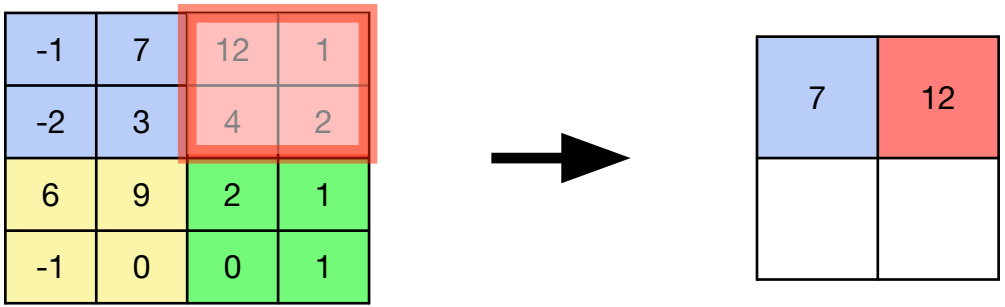
Cui et al. Multiscale Spatial-Spectral Convolutional Network with Image-Based Framework for Hyperspectral Imagery Classification, 2019

Max Pooling

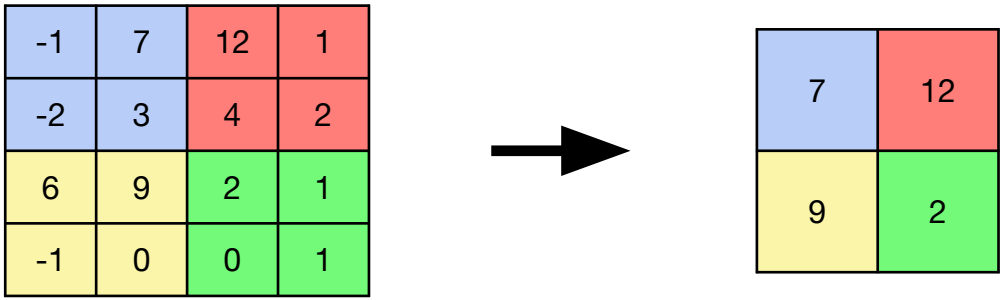
Taille 2x2 et pas de 2



Max Pooling

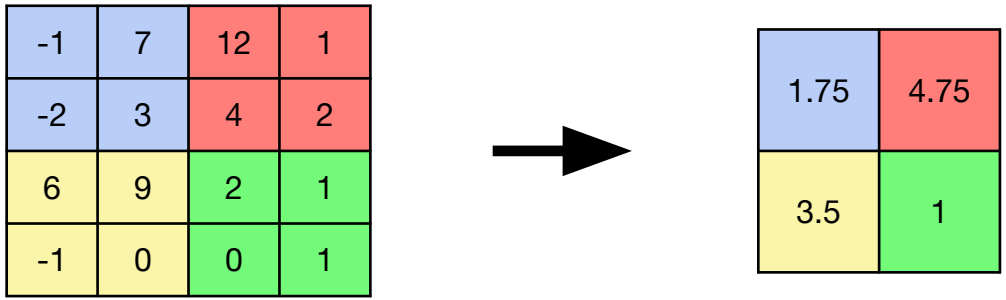


Max Pooling

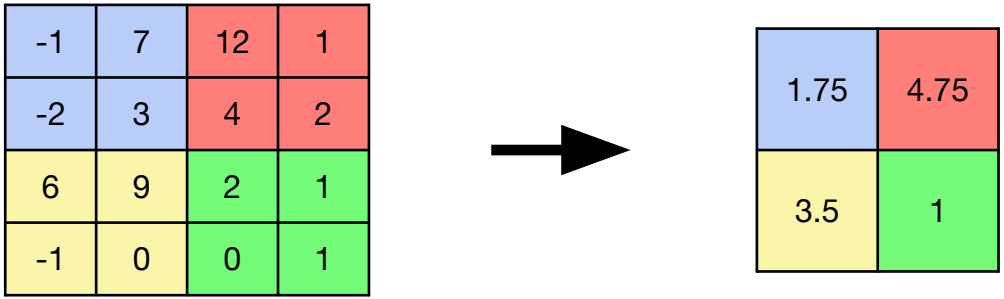


- Conserve les caractéristiques les plus marquantes d'une région et élimine les informations moins importantes.
- Moins sensible aux petites variations dans les données d'entrée, ce qui peut améliorer la robustesse.
- Réduit la complexité car réduit la dimension du features map.

Average Pooling



Average Pooling

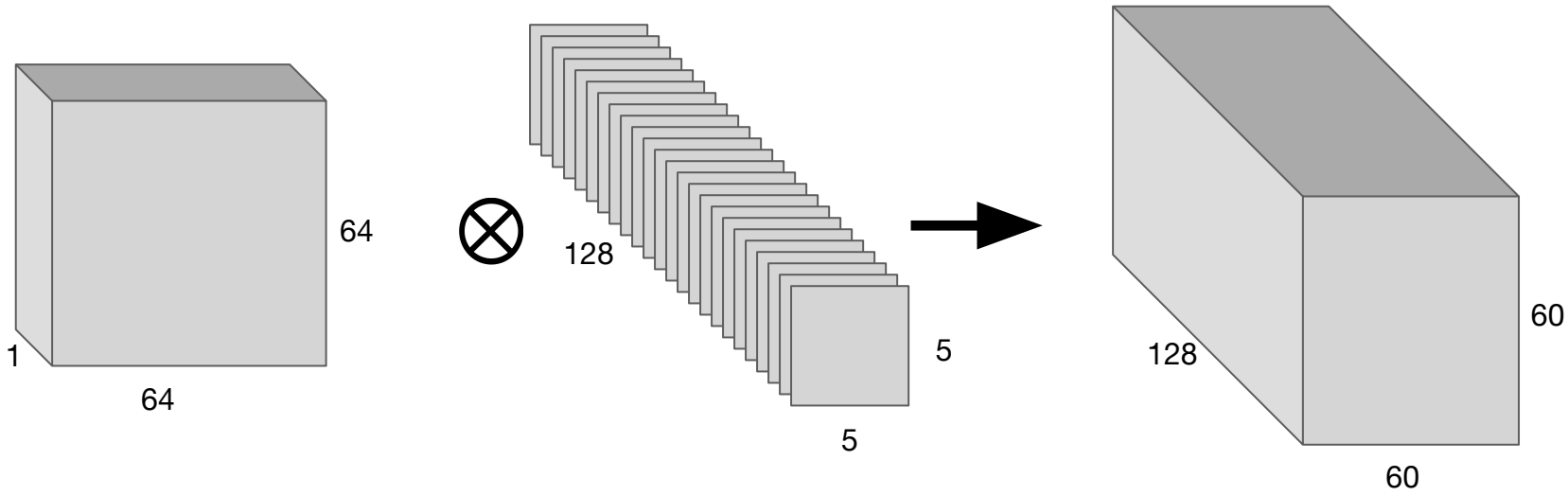


- Produit une représentation plus lissée des données d'entrée.
- Contribue à réduire les effets du bruit et des valeurs aberrantes.

Convolution Block

Exercice: Calcul de taille

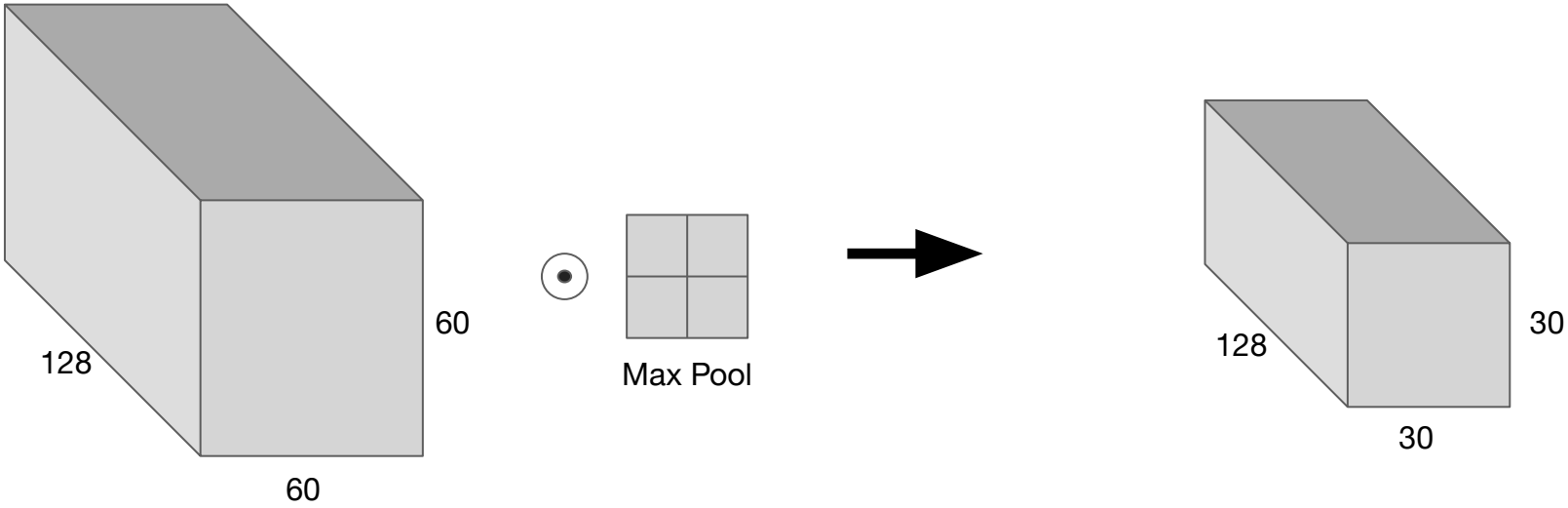
On considère une image en noir et blanc de taille 64x64. On lui applique 128 filtres de taille 5x5. On obtient des features maps de taille intermédiaire. On applique ensuite un max pooling de taille 2. Quelle est la nouvelle taille obtenue ?



Convolution Block

Exercice: Calcul de taille

On considère une image en couleur de taille 64x64. On lui applique 128 filtres de taille 5x5. On obtient des features maps de taille intermédiaire. On applique ensuite un max pooling de taille 2. Quelle est la nouvelle taille obtenue ?



Pooling

- Réduit la dimensionnalité spatiale ce qui réduit la complexité du modèle.
- Invariant à la translation.
- Augmente le champs réceptif.
- Réduit le sur-apprentissage.

Avec Pytorch

```python
class CNN2D(nn.Module):  1 usage  new *
    def __init__(self):  new *
        super(CNN2D, self).__init__()

        self.conv1 = nn.Conv2d(in_channels=1, out_channels=32, kernel_size=3, stride=1, padding=1)
        self.conv2 = nn.Conv2d(in_channels=32, out_channels=64, kernel_size=3, stride=1, padding=1)
        self.pool = nn.MaxPool2d(kernel_size=2, stride=2, padding=0)
        self.conv3 = nn.Conv2d(in_channels=64, out_channels=128, kernel_size=3, stride=1, padding=1)

        self.fc1 = nn.Linear(128 * 3 * 3,  out_features: 128)
        self.fc2 = nn.Linear( in_features: 128,  out_features: 10)
        self.softmax = nn.Softmax(dim=1)

    def forward(self, x):  new *
        x = self.pool(F.relu(self.conv1(x)))
        x = self.pool(F.relu(self.conv2(x)))
        x = self.pool(F.relu(self.conv3(x)))
        x = x.view(-1, 128 * 3 * 3)

        x = F.relu(self.fc1(x))
        x = self.fc2(x)
        x = self.softmax(x)
        _, predicted_class = torch.max(x, 1)

        return predicted_class
```

Avec Pytorch

```python
class CNN2D(nn.Module):  1 usage  new *
    def __init__(self):  new *
        super(CNN2D, self).__init__()

        self.conv1 = nn.Conv2d(in_channels=1, out_channels=32, kernel_size=3, stride=1, padding=1)
        self.conv2 = nn.Conv2d(in_channels=32, out_channels=64, kernel_size=3, stride=1, padding=1)
        self.pool = nn.MaxPool2d(kernel_size=2, stride=2, padding=0)
        self.conv3 = nn.Conv2d(in_channels=64, out_channels=128, kernel_size=3, stride=1, padding=1)

        self.fc1 = nn.Linear(128 * 3 * 3,  out_features: 128)
        self.fc2 = nn.Linear( in_features: 128,  out_features: 10)
        self.softmax = nn.Softmax(dim=1)

    def forward(self, x):  new *
        x = self.pool(F.relu(self.conv1(x)))
        x = self.pool(F.relu(self.conv2(x)))
        x = self.pool(F.relu(self.conv3(x)))
        x = x.view(-1, 128 * 3 * 3)

        x = F.relu(self.fc1(x))
        x = self.fc2(x)
        x = self.softmax(x)
        _, predicted_class = torch.max(x, 1)

        return predicted_class
```

Convolution d'une image en noir et blanc

Max pooling

Fully connected

Softmax pour prédire une probabilité pour chaque classe

Retourne la classe avec probabilité max

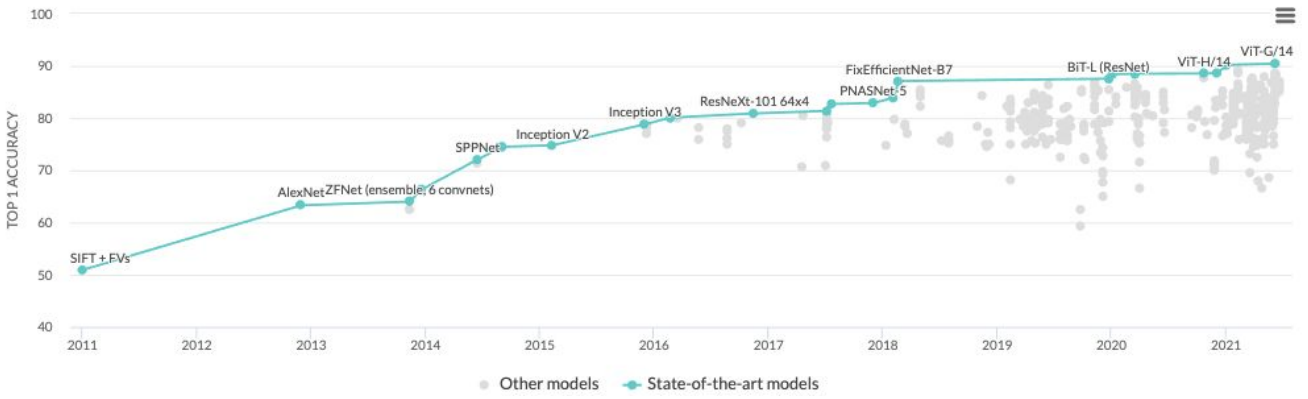
Deep architectures

IMAGENET

Image Classification on ImageNet

Leaderboard Dataset

View Top 1 Accuracy by Date for All models



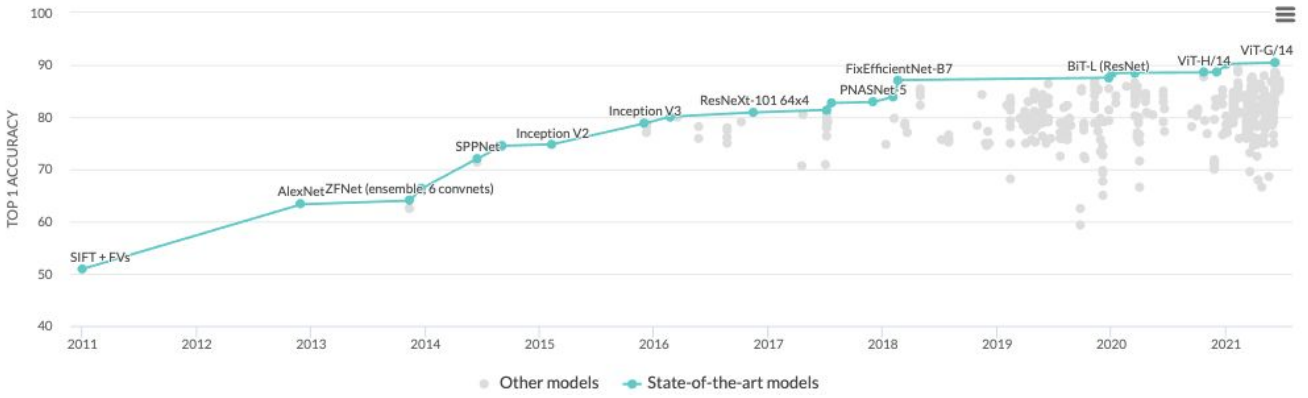
Deep architectures

IMAGENET

Image Classification on ImageNet (2009)

Leaderboard Dataset

View Top 1 Accuracy by Date for All models

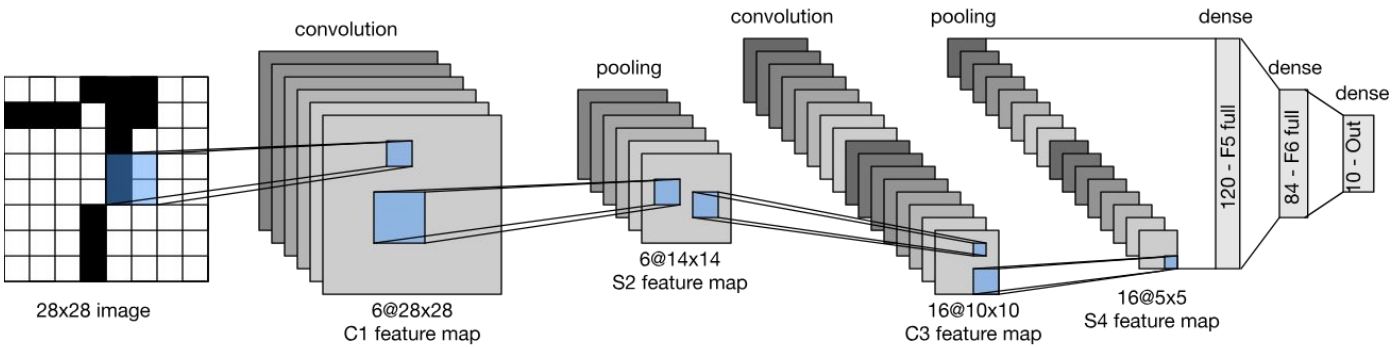


Fei Fei Li

Deep architectures

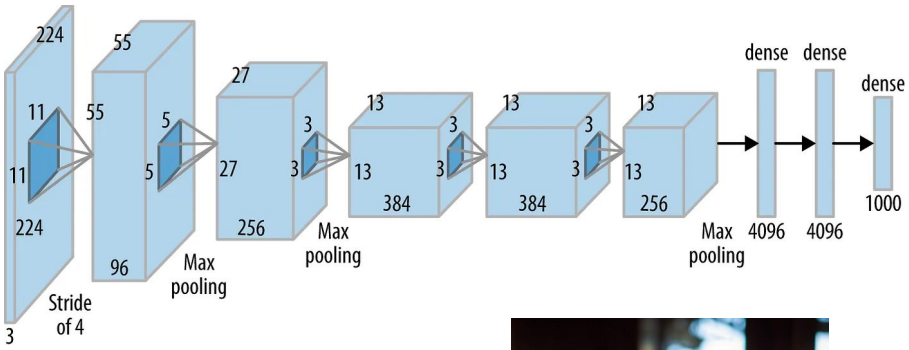
LeNet-5 (1998)

LeNet-1 (1989)



Yann LeCun

AlexNet (2012)



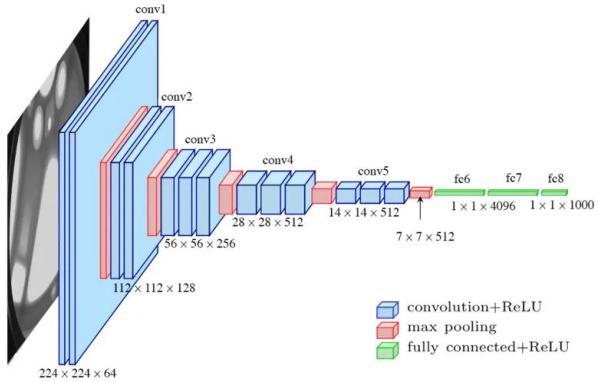
Alex Krizhevsky

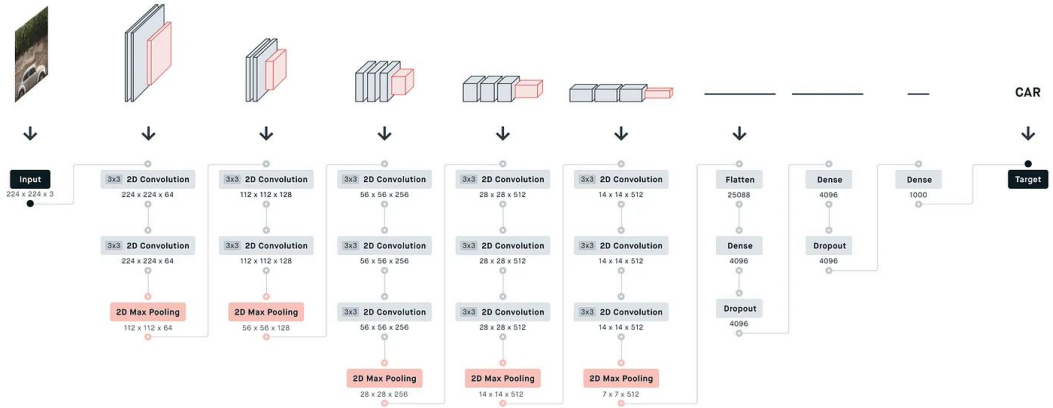
AlexNet

Image: 224 (height) × 224 (width) × 3 (channels)
↓
Convolution with 11×11 kernel+4 stride: 54×54×96
↓ ReLu
Pool with 3×3 max. kernel+2 stride: 26×26×96
↓
Convolution with 5×5 kernel+2 pad: 26×26×256
↓ ReLu
Pool with 3×3 max.kernel+2 stride: 12×12×256
↓
Convolution with 3×3 kernel+1 pad: 12×12×384
↓ ReLu
Convolution with 3×3 kernel+1 pad: 12×12×384
↓ ReLu
Convolution with 3×3 kernel+1 pad: 12×12×256
↓ ReLu
Pool with 3×3 max.kernel+2 stride: 5×5×256
↓ flatten
Dense: 4096 fully connected neurons
↓ ReLu, dropout p=0.5
Dense: 4096 fully connected neurons
↓ ReLu, dropout p=0.5
Dense: 1000 fully connected neurons
↓
Output: 1 of 1000 classes

Alexnet Block Diagram
(source:oreilly.com)
https://en.wikipedia.org/wiki/Alex Net

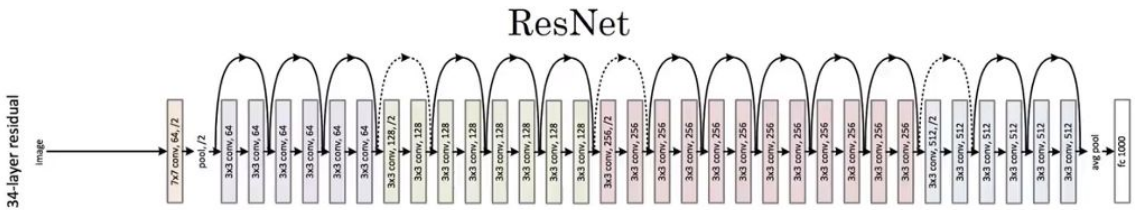
VGG (2014)

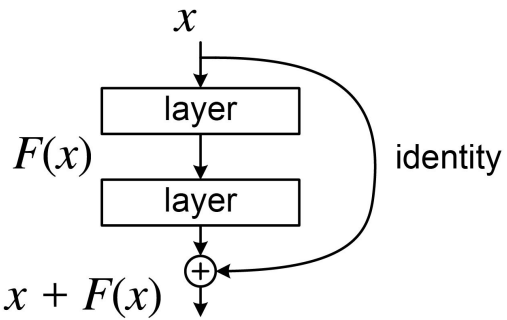




https://medium.com/@siddheshb008/vgg-net-architecture-explained-71179310050f

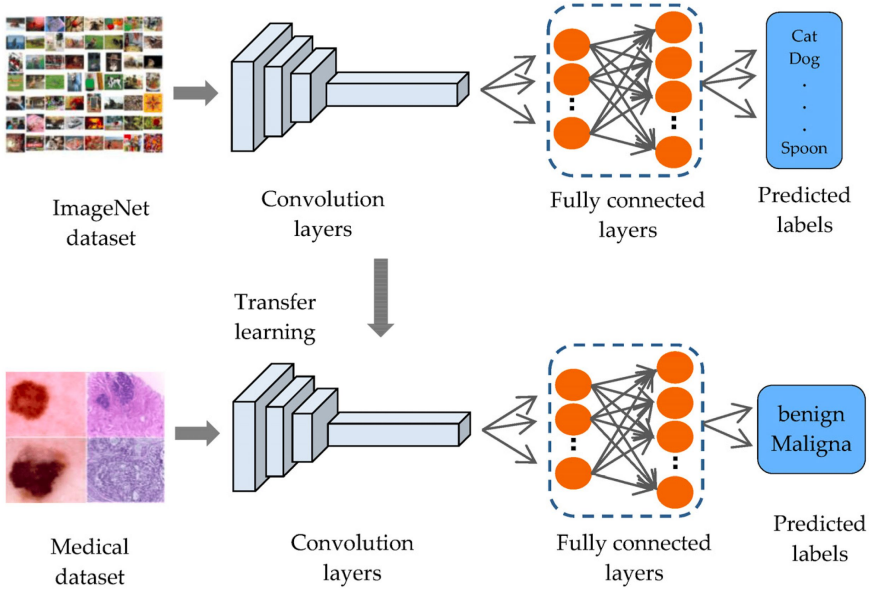
ResNet (Residual Network)





He et al. Deep Residual Learning for Image Recognition, 2015.

Transfer Learning



https://www.mdpi.com/1424-8220/23/2/570

TP 2: CNNs