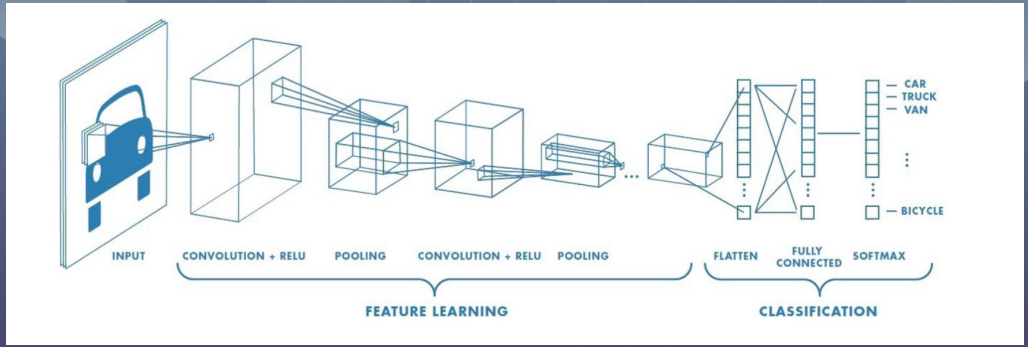
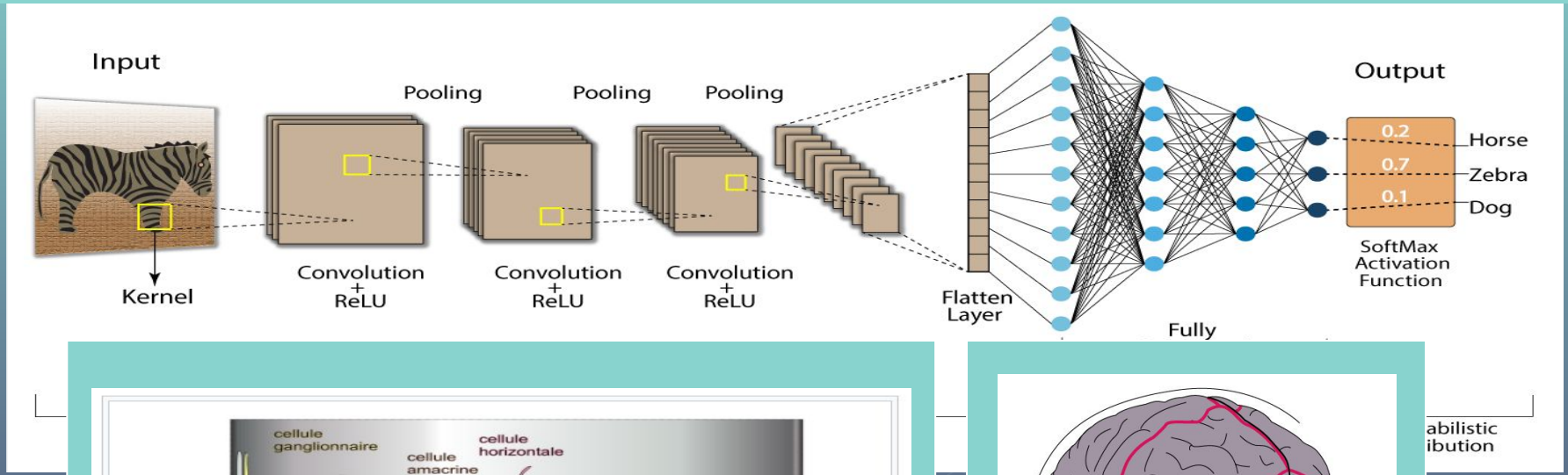
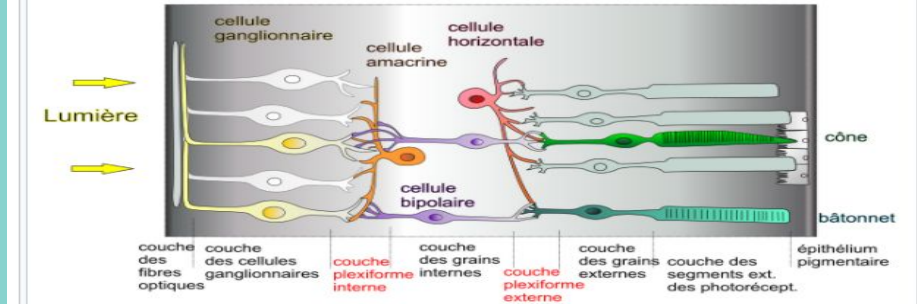
Réseaux de neurones convolutionnels

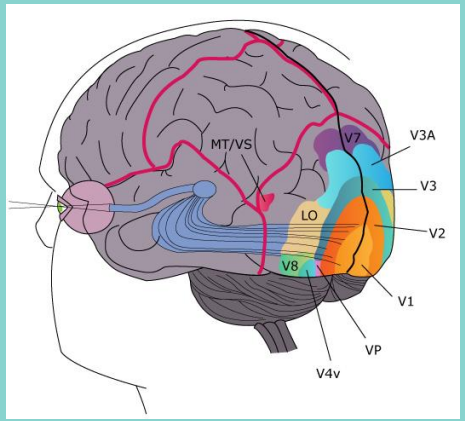


Les réseaux neurones convolutifs et bio inspiration

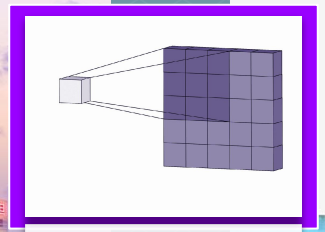


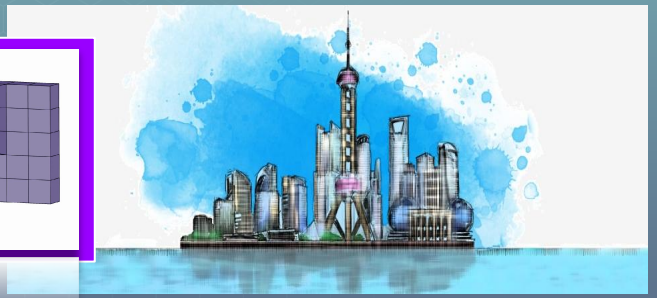


(Fig. 3) Structure de la rétine (dessin modifié de Purves[3]).

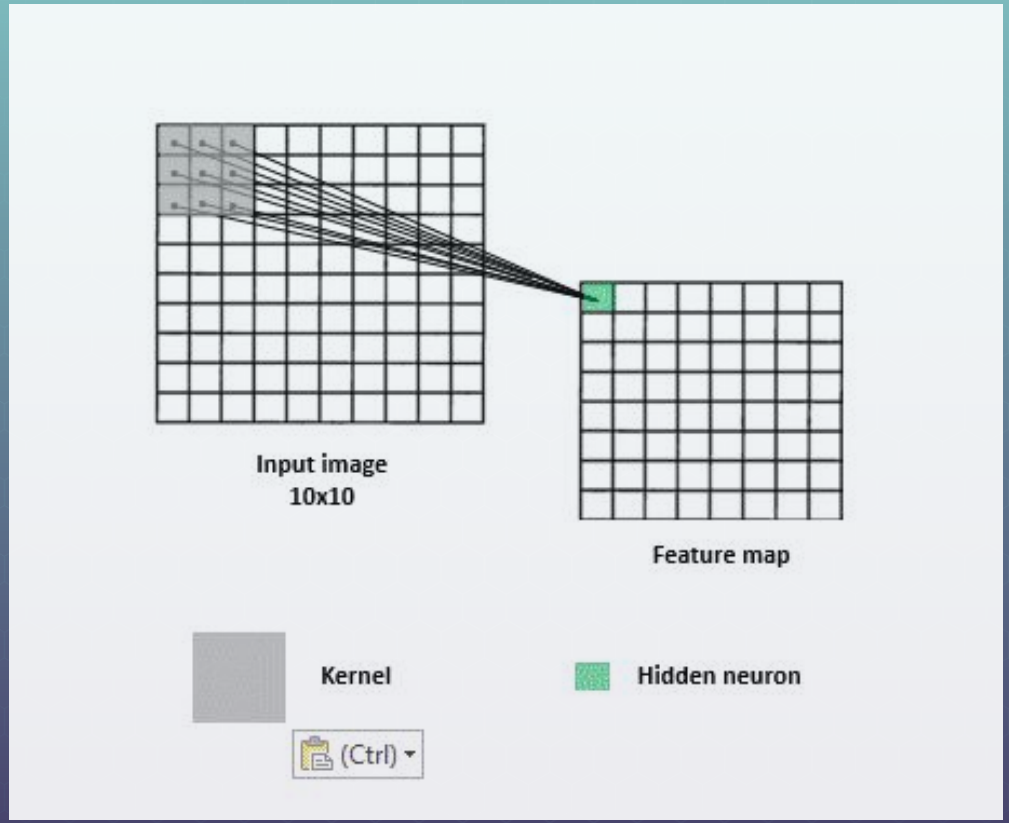


Principe d'un réseau de neurone CNN

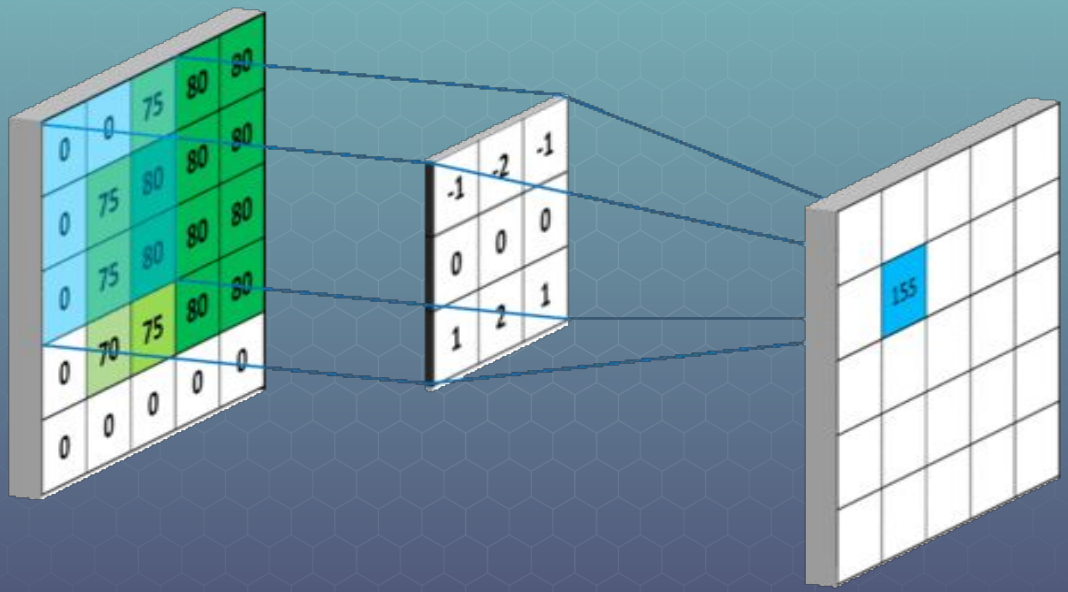




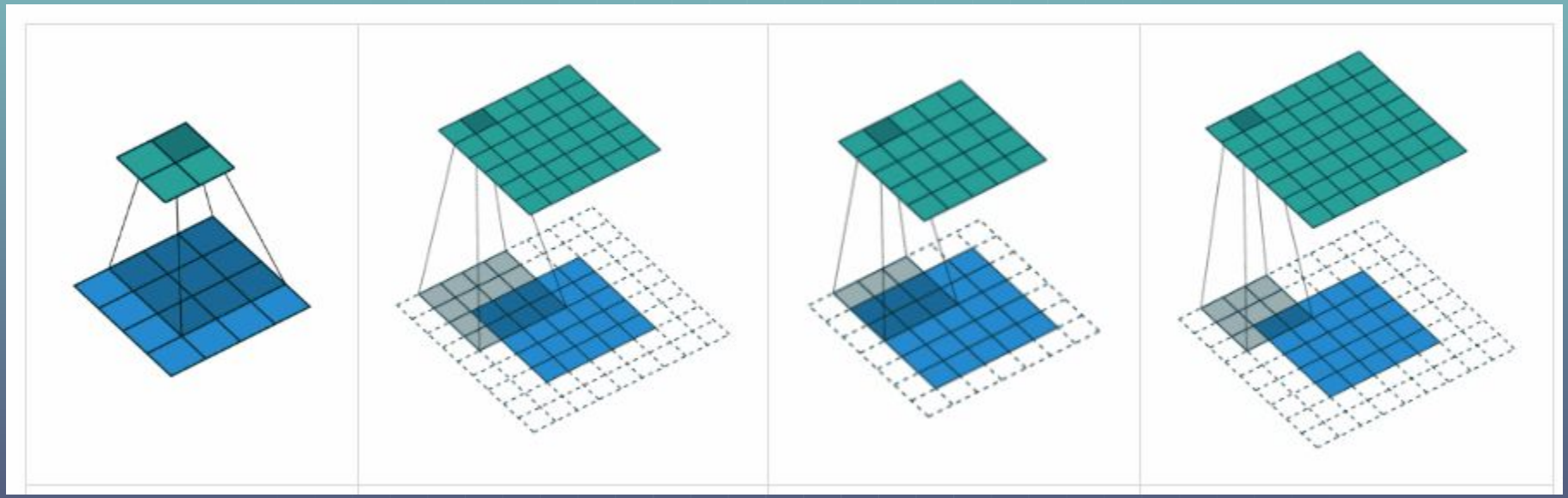
La convolution en détail



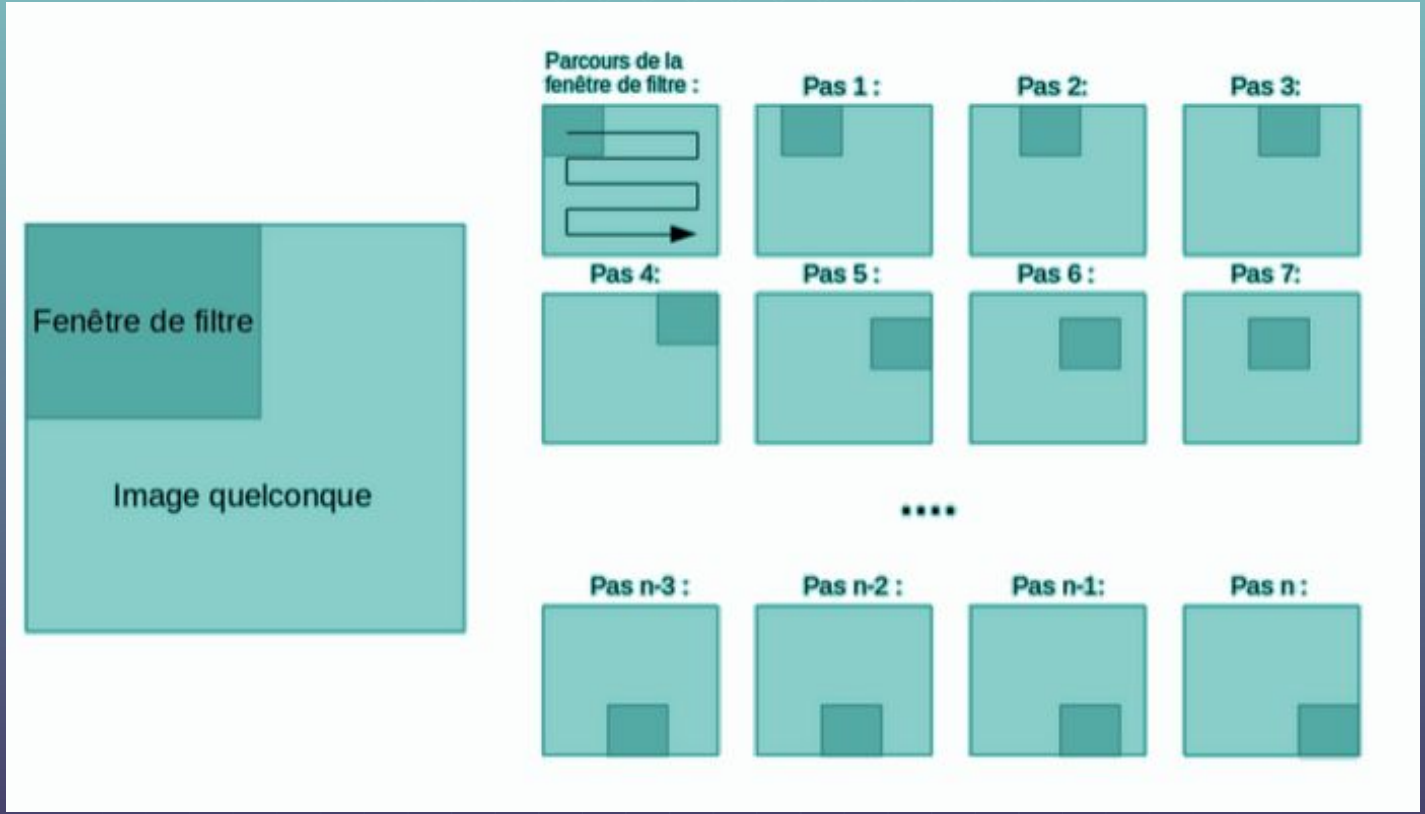
Le kernel


0 0 75 80 80
0 75 80 80 80
0 75 80 80 80
0 70 75 80 80
0 0 0 0 0
-1 -2 -1
0 0 0
1 2 1
155

Padding



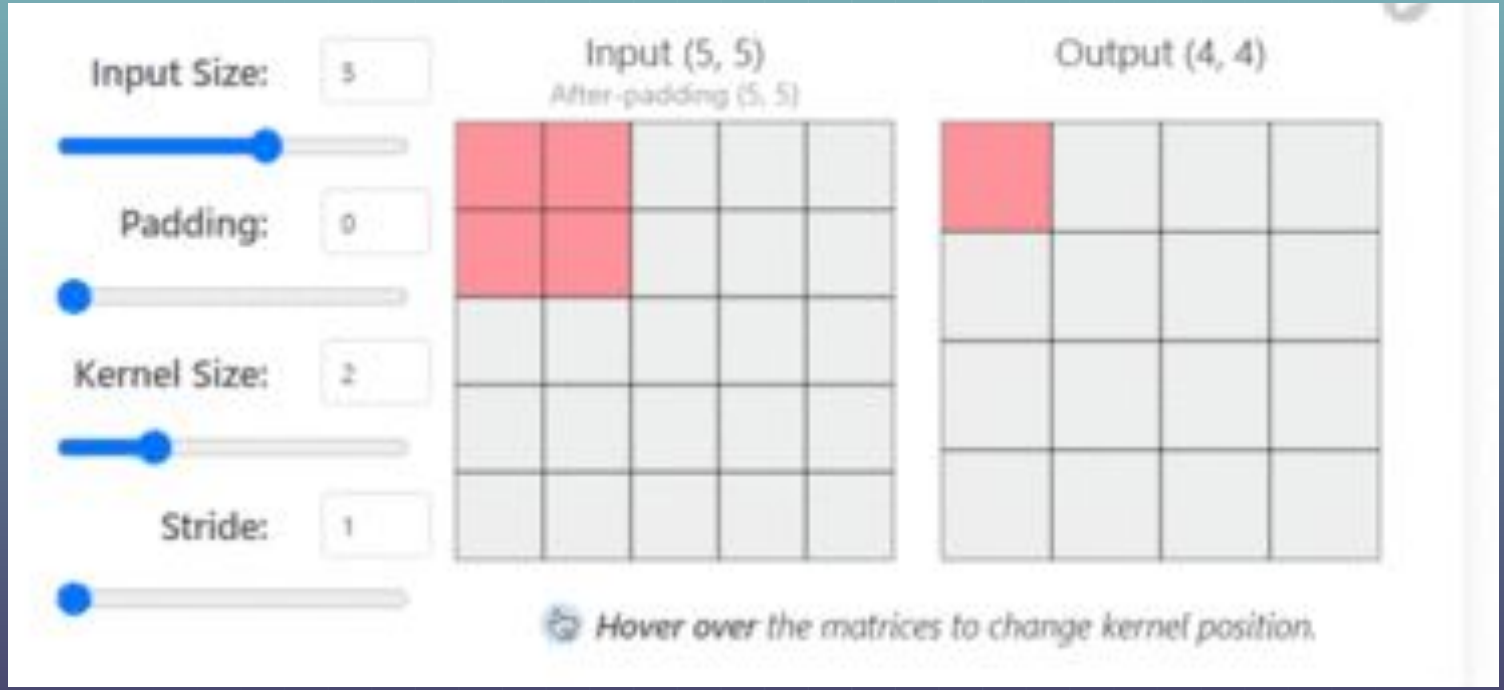
Stride


Fenêtre de filtre
Image quelconque
Parcours de la fenêtre de filtre :
Pas 1 :
Pas 2:
Pas 3:
Pas 4:
Pas 5 :
Pas 6 :
Pas 7:
....
Pas n-3 :
Pas n-2 :
Pas n-1:
Pas n :

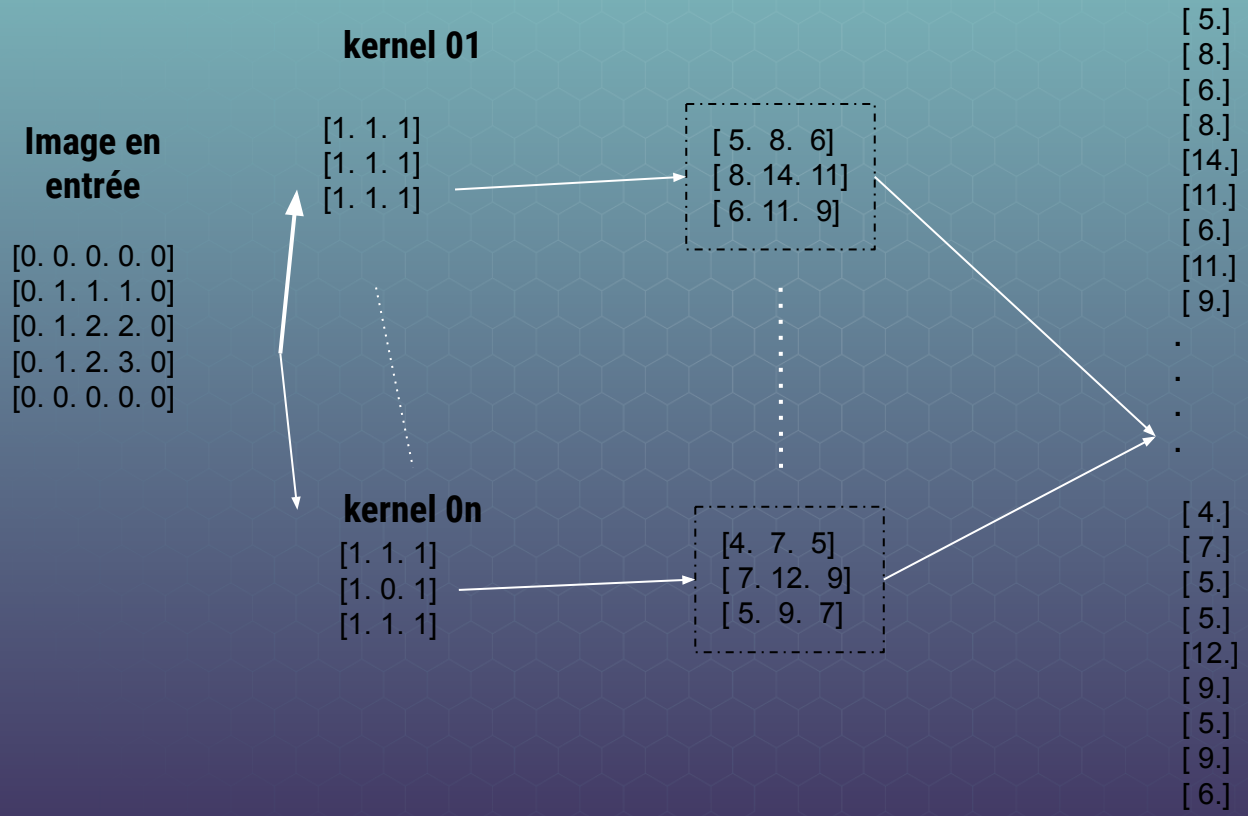
Feature Map

Description de l'image en termes de caractéristique associée à chaque pixel.

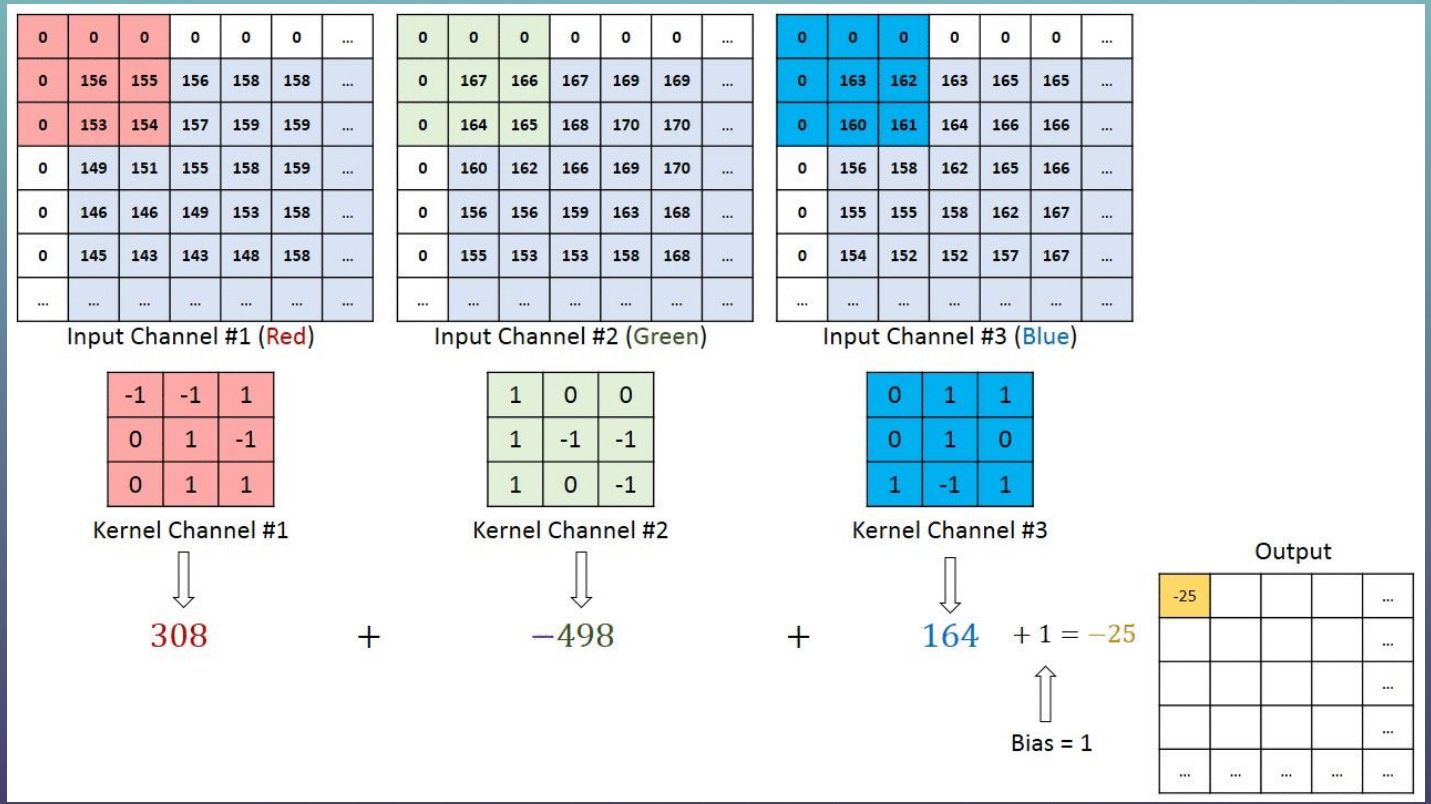
UNE IMAGE EN NB ET UNE CONVOLUTION, CA FAIT QUOI ?



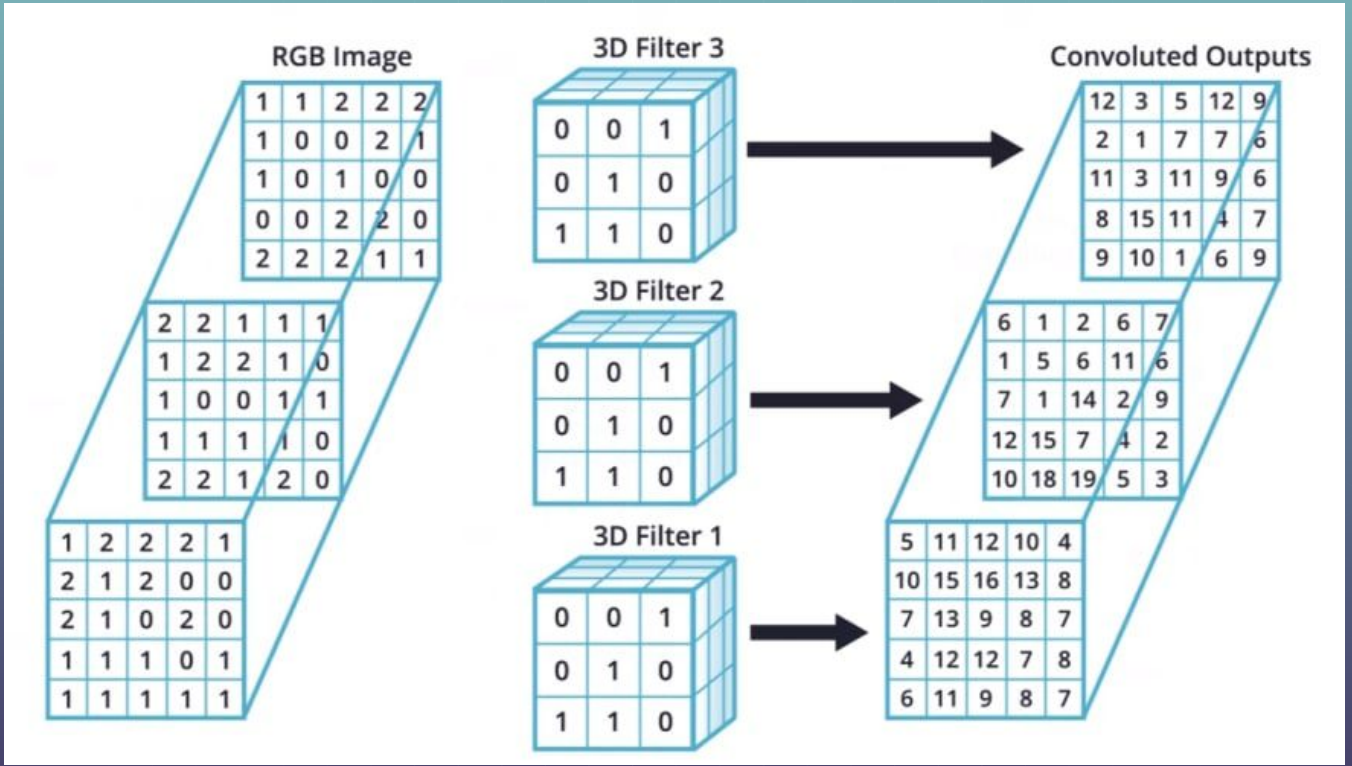
UNE IMAGES AVEC DEUX OU n FILTRES CA DONNE ;



ET EN COULEURS, ALORS !



Plus il y a de kernel, plus on rit !

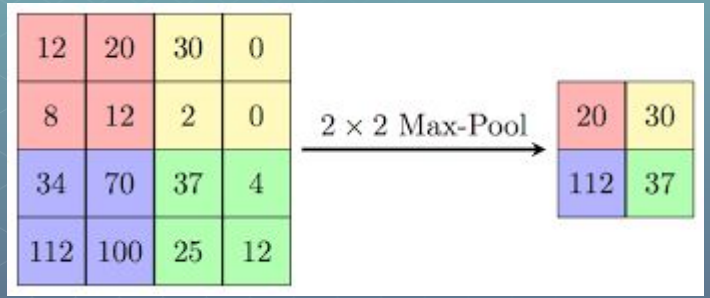


La couche de pooling 2D

- Le pooling s'applique sur la features map et la ré-échantillonne.
- Similaire à la convolution -> la fenêtre se déplace sur la features map.

> Réduit la charge de calcul

> Réduit le nombre de paramètres

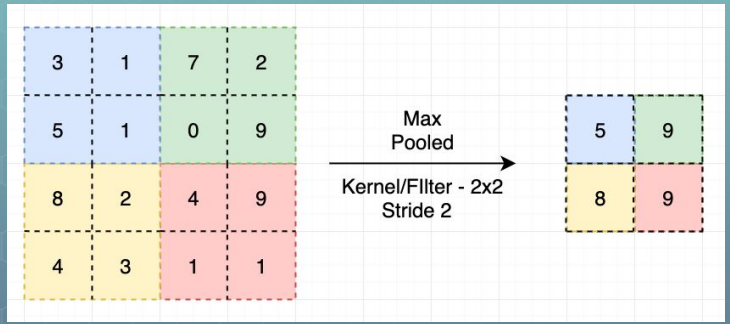


source illustration : https://computersciencewiki.org/index.php/Max-pooling_/_Pooling

La couche de pooling 2D

Définition dans keras :

- Stride (pas)
- Padding
- Size

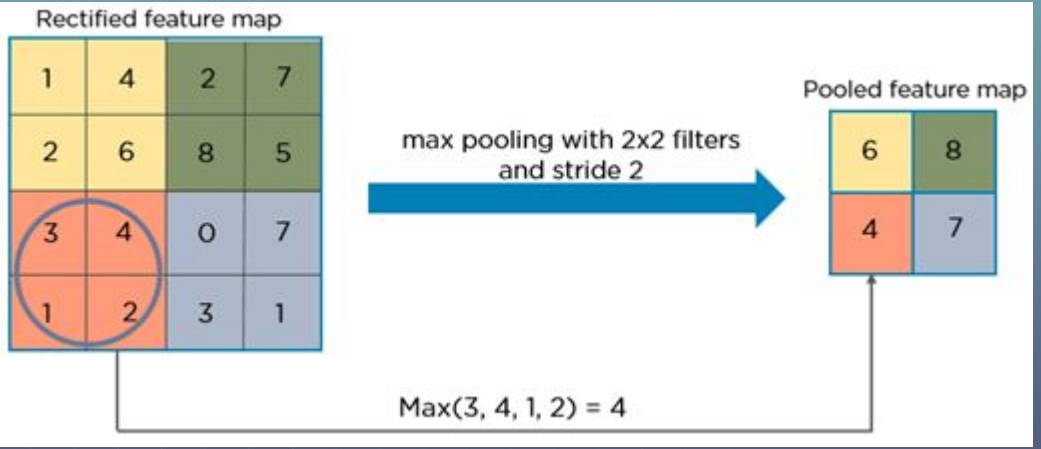



```
model.add(AveragePooling2D(pool_size = (2,2), strides=(2,2), padding="valid"))
```

source illustration : https://medium.com/ai-in-plain-english/pooling-layer-beginner-to-intermediate-fa0dbdce80eb

Maxpooling2D

- Elle sélectionne 4 valeurs et garde en mémoire la plus grande.
- Dans cette configuration, la Pooled feature map est 4x moins grande que la feature map.



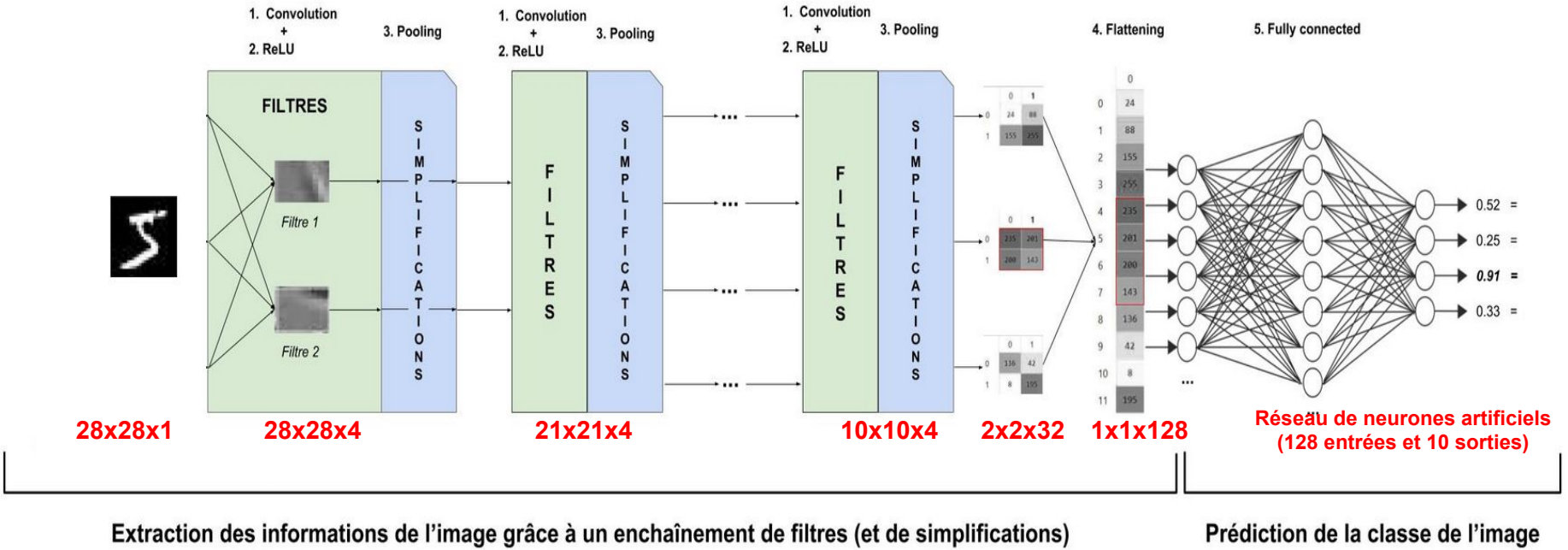
source illustration : https://computersciencewiki.org/index.php/Max-pooling_/_Pooling

Average Pooling 2D

- Identique au Maxpooling, mais au lieu de sélectionner la valeur la plus élevée, l'Average Pooling fait la moyenne de toutes les valeurs observées.

source illustration: https://embarc.org/embarc_mli/doc/build/html/MLI_kernels/pooling_avg.html

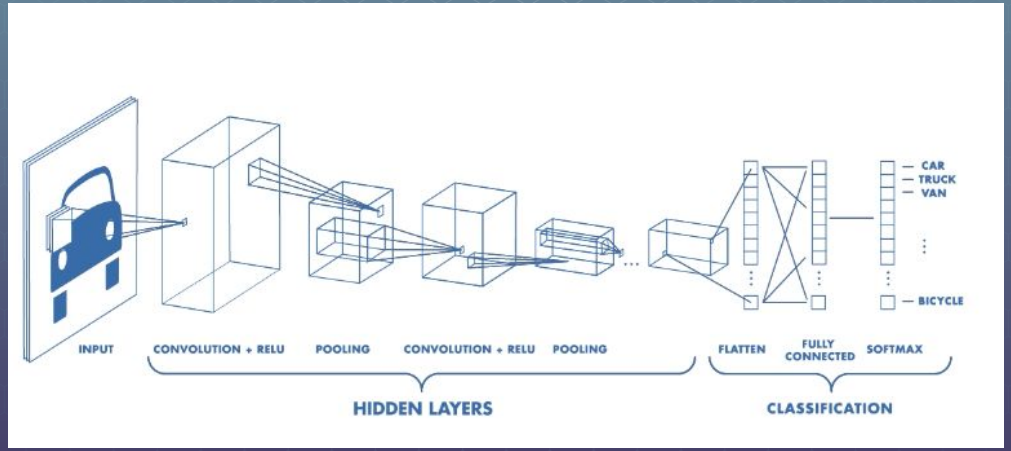
Schéma du fonctionnement du modèle CNN



Architecture typique d'un CNN

Un réseau CNN est constitué de trois parties :

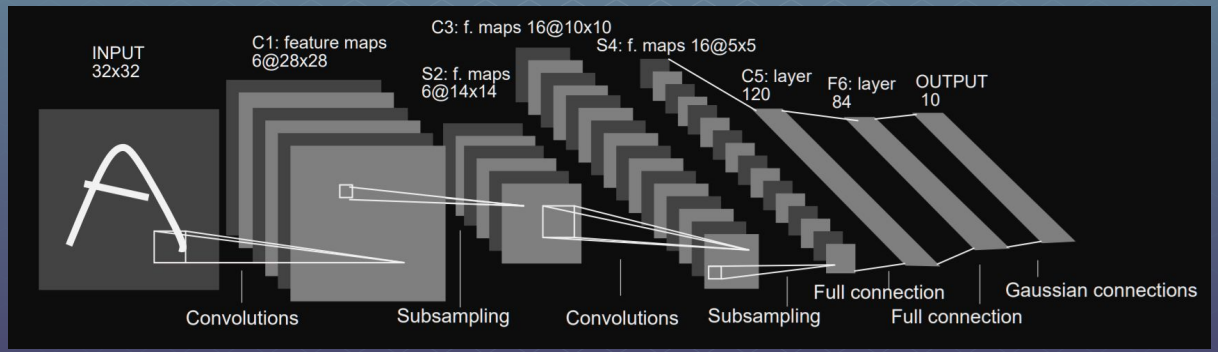
- Une partie convolutive (features extraction)
- Un Flatten()
- Une partie fully-connected



source illustration : http://www.ashokrahulgade.com/skills/DL/CNN/Introduction.html

Architecture LeNet-5

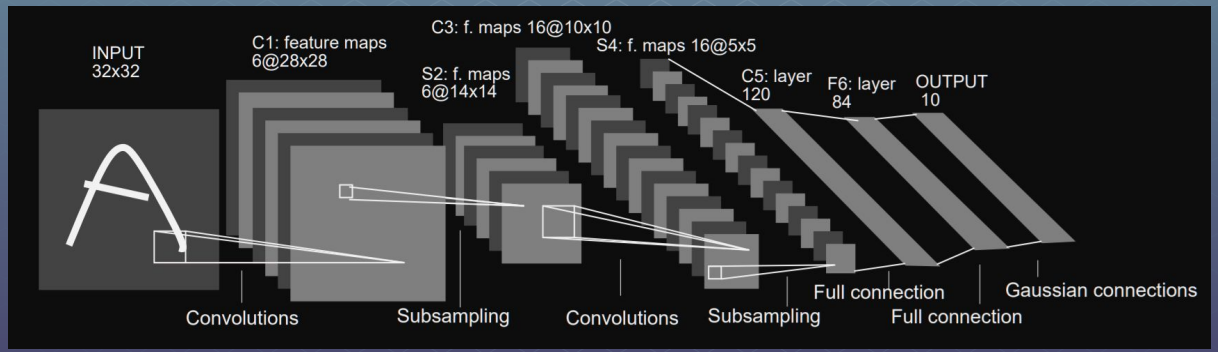
“Gradient-Based Learning Applied To Document Recognition” 1998 par Yann LeCun, Léon Bottou, Yoshua Bengio, et Patrick Haffner



source illustration : https://towardsdatascience.com/understanding-and-implementing-lenet-5-cnn-architecture-deep-learning-a2d531ebc342

Architecture LeNet-5

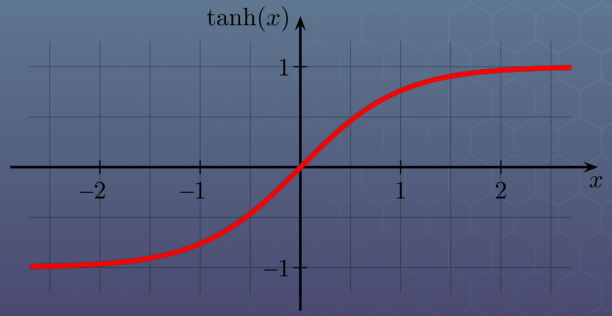
- 3 couches de convolution et 2 couches de sous-échantillonnage (AveragePooling)
- 2 couches fully-connected
- Reprise de l'architecture classique : convolution / fully-connected



source illustration : https://towardsdatascience.com/understanding-and-implementing-lenet-5-cnn-architecture-deep-learning-a2d531ebc342

Architecture LeNet-5 avec Keras

Utilisation de la fonction Tangente hyperbolique.




```python
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Conv2D, Dense, Flatten, AveragePooling2D

model = Sequential()

model.add(Conv2D(filters = 6, kernel_size=(5,5), strides=1, activation='tanh', input_shape=(28, 28, 1), padding="same"))
model.add(AveragePooling2D(pool_size = (2,2), strides=(2,2), padding="valid"))

model.add(Conv2D(filters = 16, kernel_size=(5,5), strides=1, activation='tanh', padding="valid"))
model.add(AveragePooling2D(pool_size = (2,2), strides=(2,2), padding="valid"))

model.add(Conv2D(filters=120, kernel_size=(5, 5), strides=(1, 1), activation='tanh', padding='valid'))

model.add(Flatten())

model.add(Dense(units = 84, activation='tanh'))
model.add(Dense(units = 10, activation='softmax'))


model.compile(optimizer='adam', loss='categorical_crossentropy', metrics=['accuracy'])
```

```python
model.summary()
```

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d (Conv2D)              (None, 28, 28, 6)         156
_________________________________________________________________
average_pooling2d (AveragePo (None, 14, 14, 6)         0
_________________________________________________________________
conv2d_1 (Conv2D)            (None, 10, 10, 16)        2416
_________________________________________________________________
average_pooling2d_1 (Average (None, 5, 5, 16)          0
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 1, 1, 120)         48120
_________________________________________________________________
flatten (Flatten)            (None, 120)               0
_________________________________________________________________
dense (Dense)                (None, 84)                10164
_________________________________________________________________
dense_1 (Dense)              (None, 10)                850
=================================================================
Total params: 61,706
Trainable params: 61,706
Non-trainable params: 0
_________________________________________________________________
```

Architecture LeNet-5

- Son application majeure a été pour la reconnaissance de caractères manuscrits (MNIST)
- Automatisation du classement des chèques de banque.

source illustration: http://yann.lecun.com/exdb/lenet/

Vous n’avez pas de questions?!

Une équipe à votre service