

Computer Vision

Introduction

Bassam Kurdy Ph.D

<bassam.kurdy@apinum.fr>

La *Vision par ordinateur* CV - kézako ?

- La **computer vision** désigne une technique **d'intelligence artificielle** permettant d'analyser des images captées par un équipement tel qu'une **caméra**.
- Concrètement, la computer vision se présente comme un outil basé sur l'IA capable de **reconnaître** une image, de la **comprendre**, et de **traiter** les informations qui en découlent.
- Pour beaucoup, la vision par ordinateur est l'équivalent, en termes d'IA, des **yeux humains** et de la capacité de notre **cerveau** à **traiter** et **analyser** les images perçues.
- La reproduction de la vision humaine par des ordinateurs constitue d'ailleurs l'un des grands objectifs de la computer vision.

Deep Learning & Computer Vision



« We should stop training radiologists now... in five years deeplearning is going to do better. Radiologists are like the coyote already over the edge of the cliff who hasn't yet looked down »

<< Il faut arrêter de former les radiologues maintenant... dans cinq ans le Deep Learning ira mieux. Les radiologues sont comme le coyote déjà au bord de la falaise qui n'a pas encore regardé vers le bas >>

*Quelles sont les **applications** de la computer vision?*

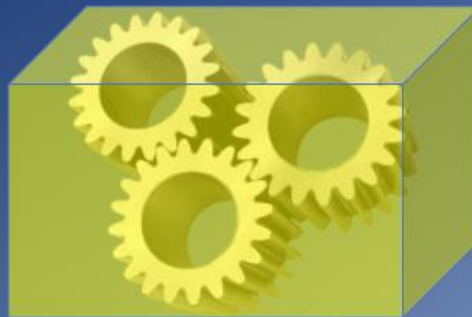


On recense aujourd'hui de nombreux domaines d'application :

- **Les voitures autonomes**
- **Les systèmes de reconnaissance faciale**
- **L'industrie**
- **Assurance qualité**
- **La Santé (Diagnostic médical)**
- **Reconnaissance de caractères (OCR)**
- **Gérer les archives d'images**
- **Trademark et de copyright**
- **Problèmes de sécurité**
- **Problèmes militaires**
- **SIG et télédétection**
- **... etc**

Machine Learning

= Classifieur



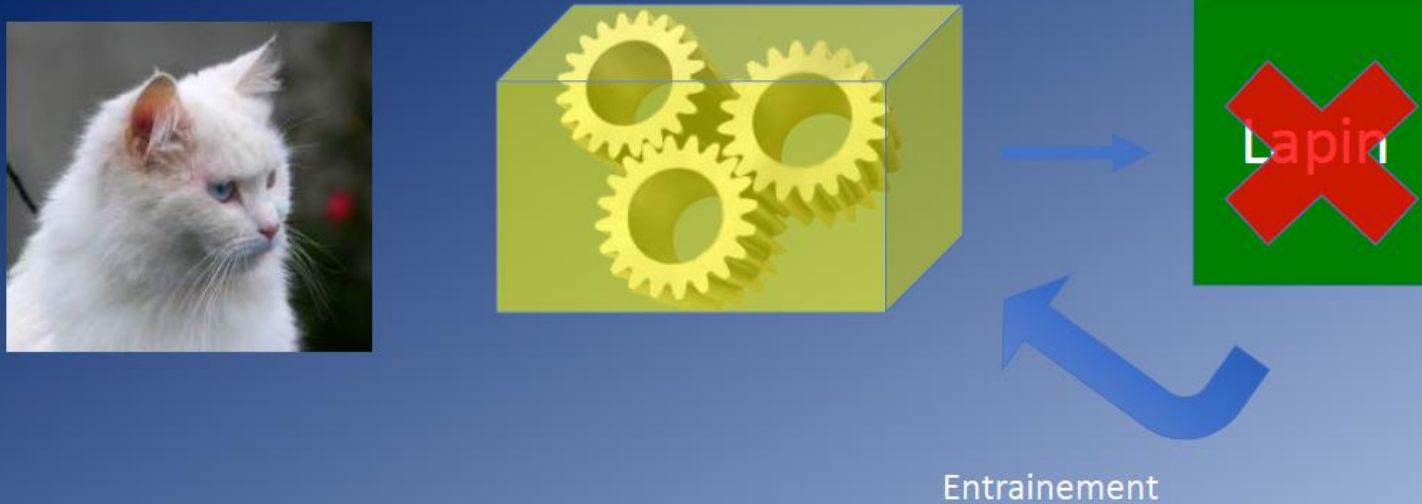
Chat



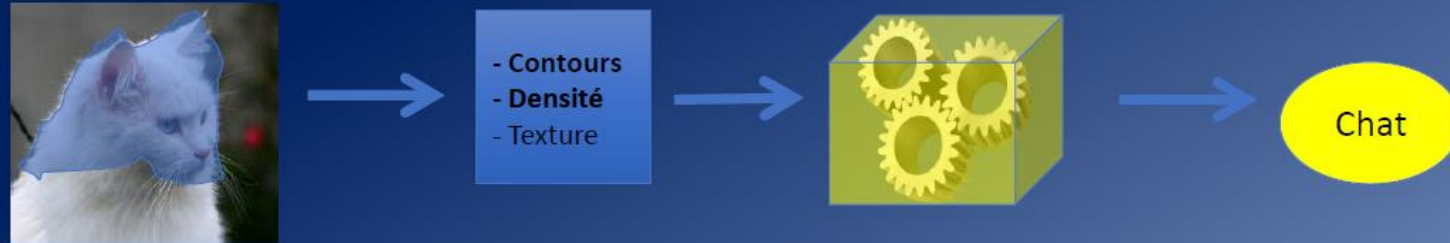
Lapin

Machine Learning

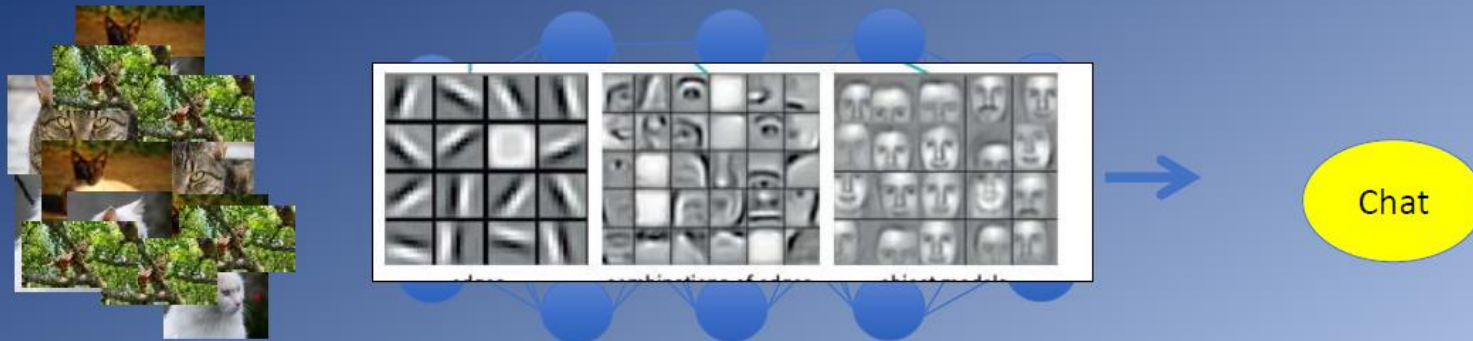
= Classifieur



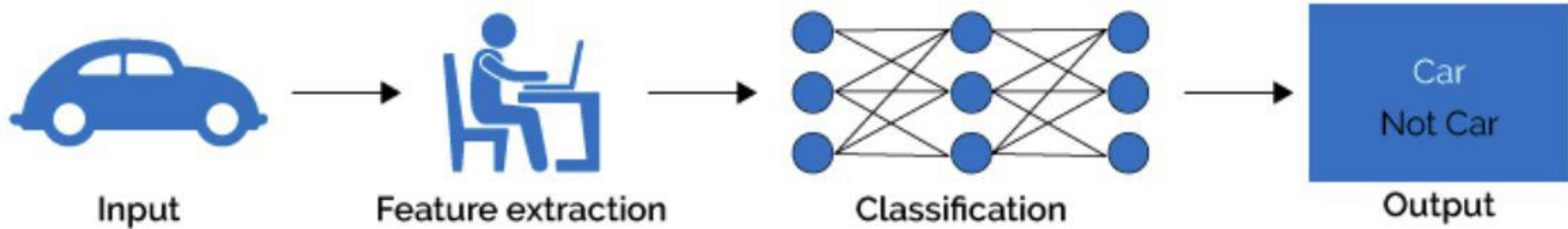
Machine Learning



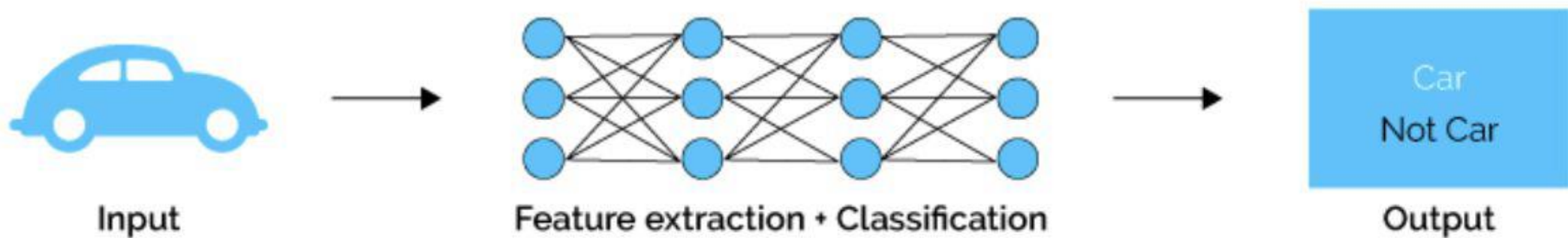
Deep Neural Network



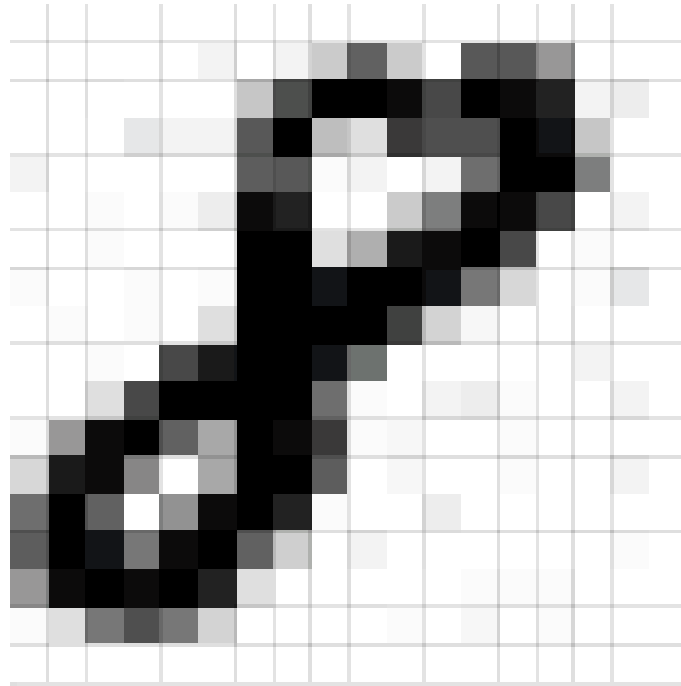
Machine Learning



Deep Learning



Une image est une matrice de valeurs de pixels



Pourquoi utiliser OpenCV ?

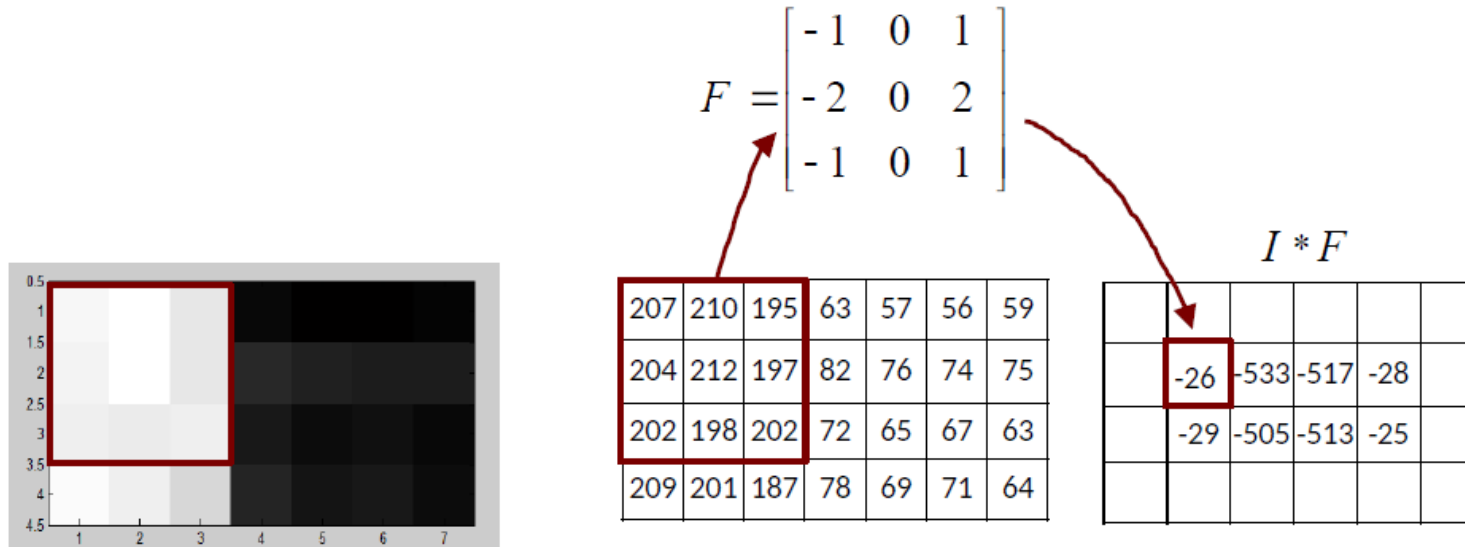
- Les algorithmes *d'OpenCV* permettent d'appliquer divers traitements sur les images pour faciliter la détection d'éléments précis dans celles-ci.
- Voici quelques exemples :
 - **Le Thresholding (seuillage)** d'une image permet de définir une valeur de pixel (correspondant à une couleur ou niveau de gris) qui servira de seuil. Au-dessus ou en dessous de cette valeur (selon l'algorithme), tous les pixels se verront assigner une autre valeur.
 - **Les filtres de détection de contours**
 - **Les filtres de lissage**
 - **Les filtres morphologiques**
 - **... etc.**

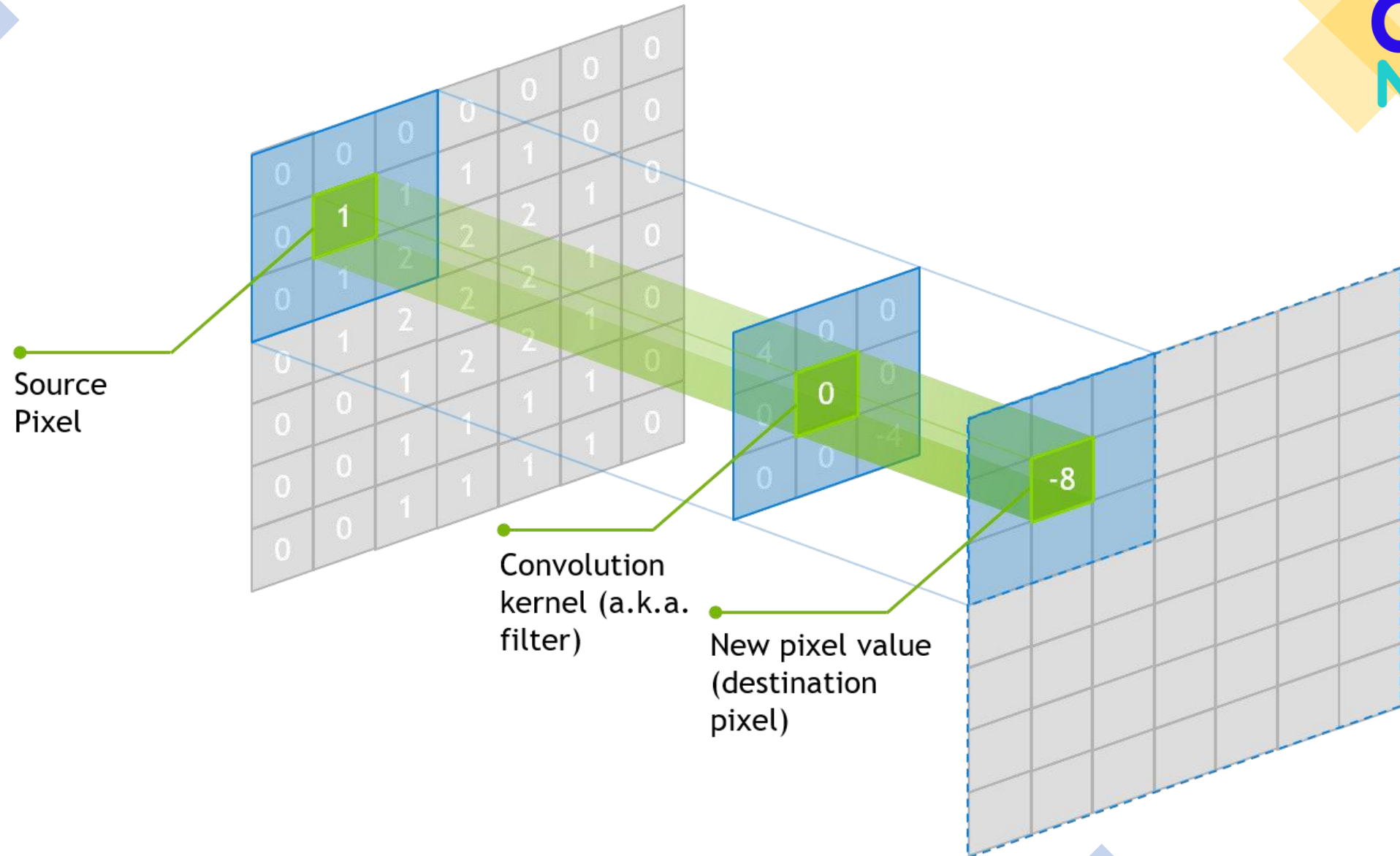
Exemple « convolution »

Appellation de Filtre (*Filter*), ou Noyau (*Kernel*)

Soit une image $I \in \mathbb{R}^{m \times n}$ et un filtre de convolution $F \in \mathbb{R}^{k \times k}$ de taille impaire $k = 2d + 1$ (avec $d \in \mathbb{N}^+$):

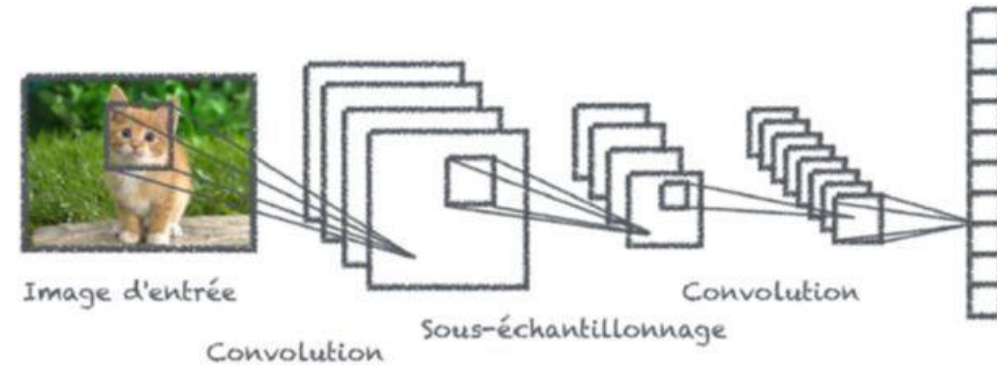
$$(I * F)[x, y] = \sum_{i=-d}^{+d} \sum_{j=-d}^{+d} I[x+i, y+j] \times F[i+d+1, j+d+1]$$





Les réseaux de neurones convolutionnels

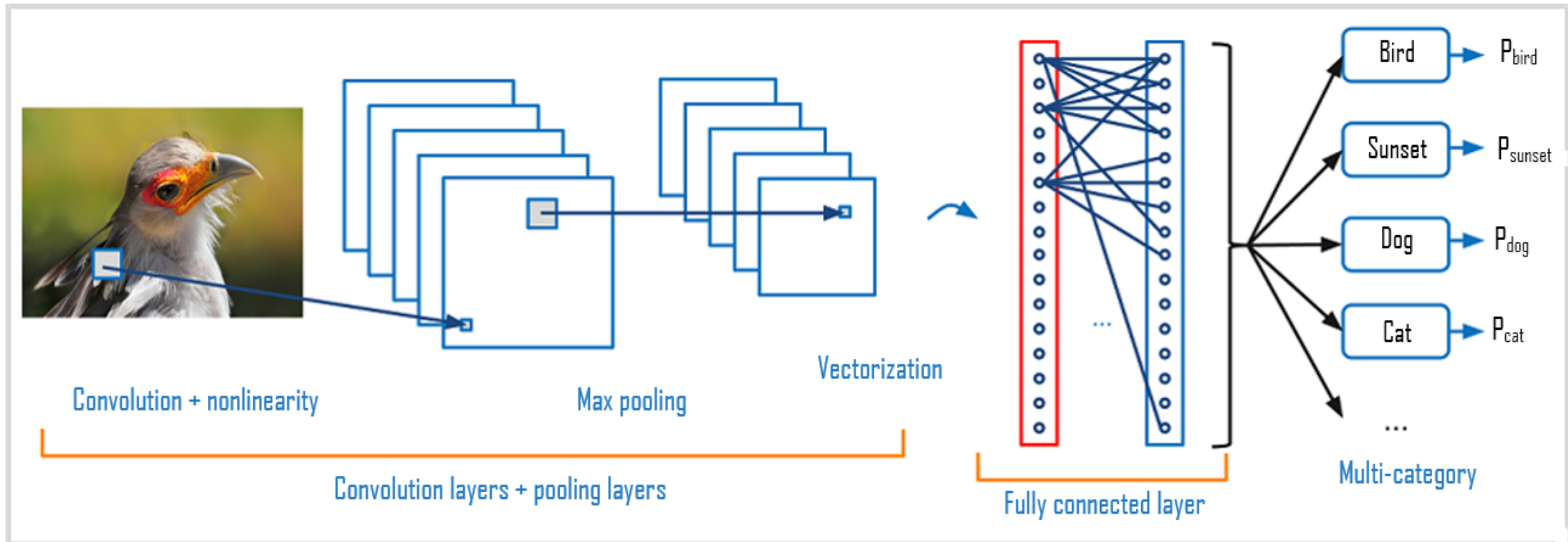
En apprentissage automatique, un **réseau de neurones convolutifs** ou **réseau de neurones à convolution** (en anglais *CNN* ou *ConvNet* pour *Convolutional Neural Networks*) est un type de réseau de neurones artificiels acycliques (*feed-forward*), dans lequel le motif de connexion entre les neurones est inspiré par le cortex visuel des animaux.

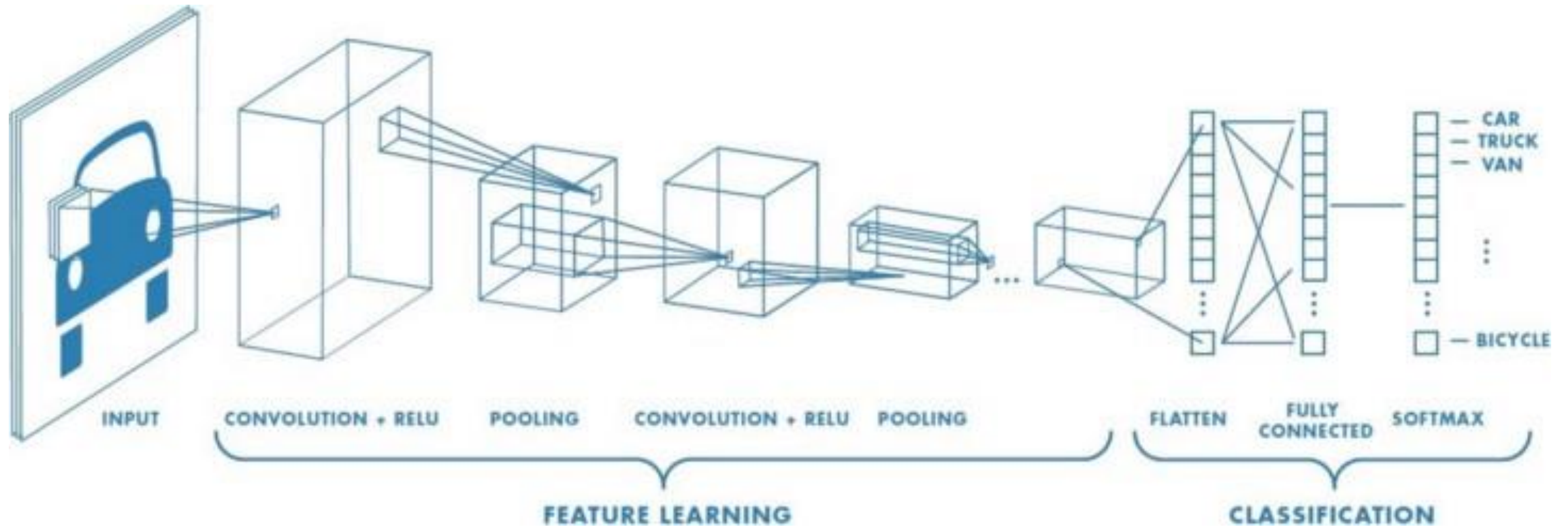


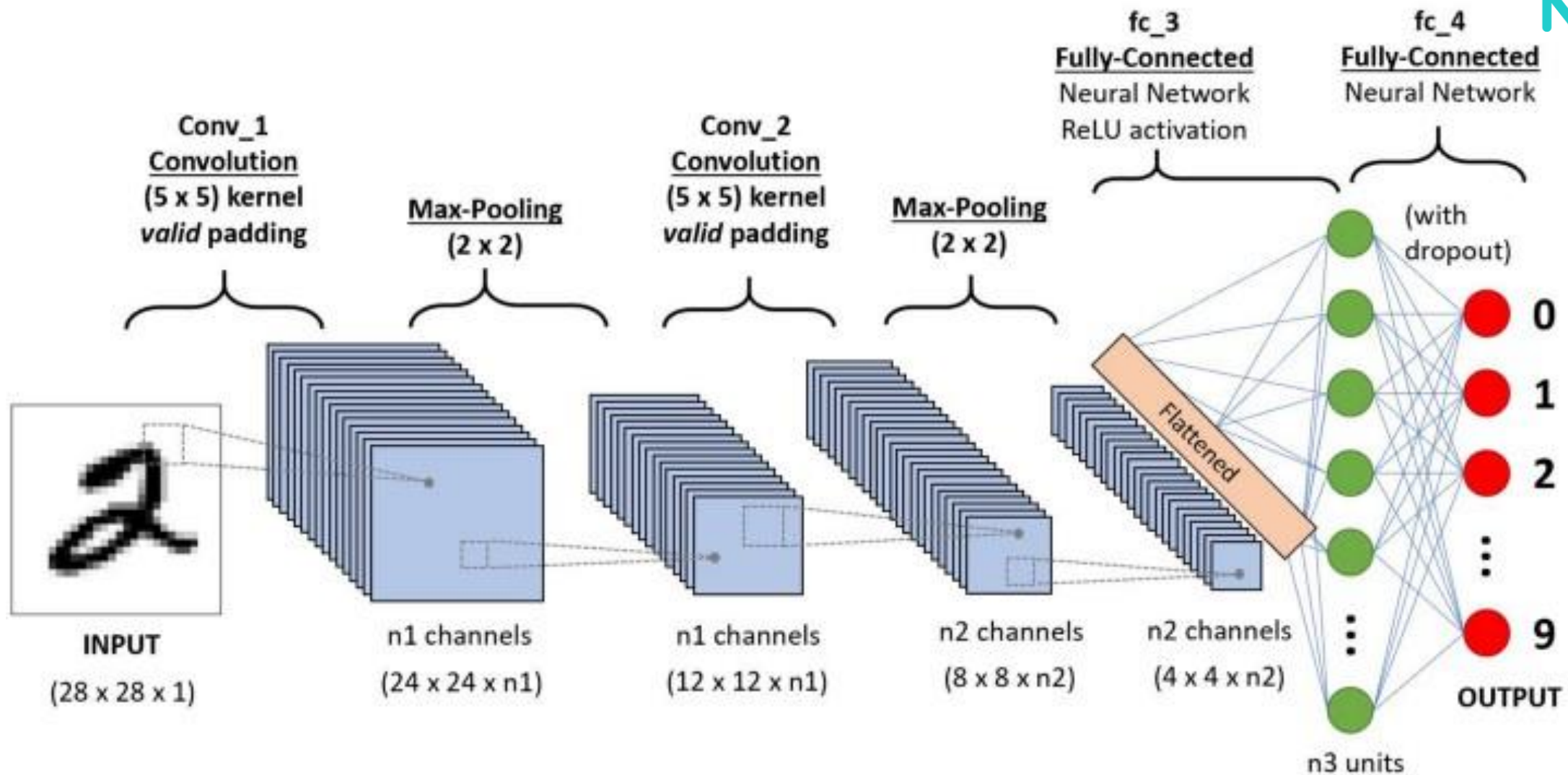
33

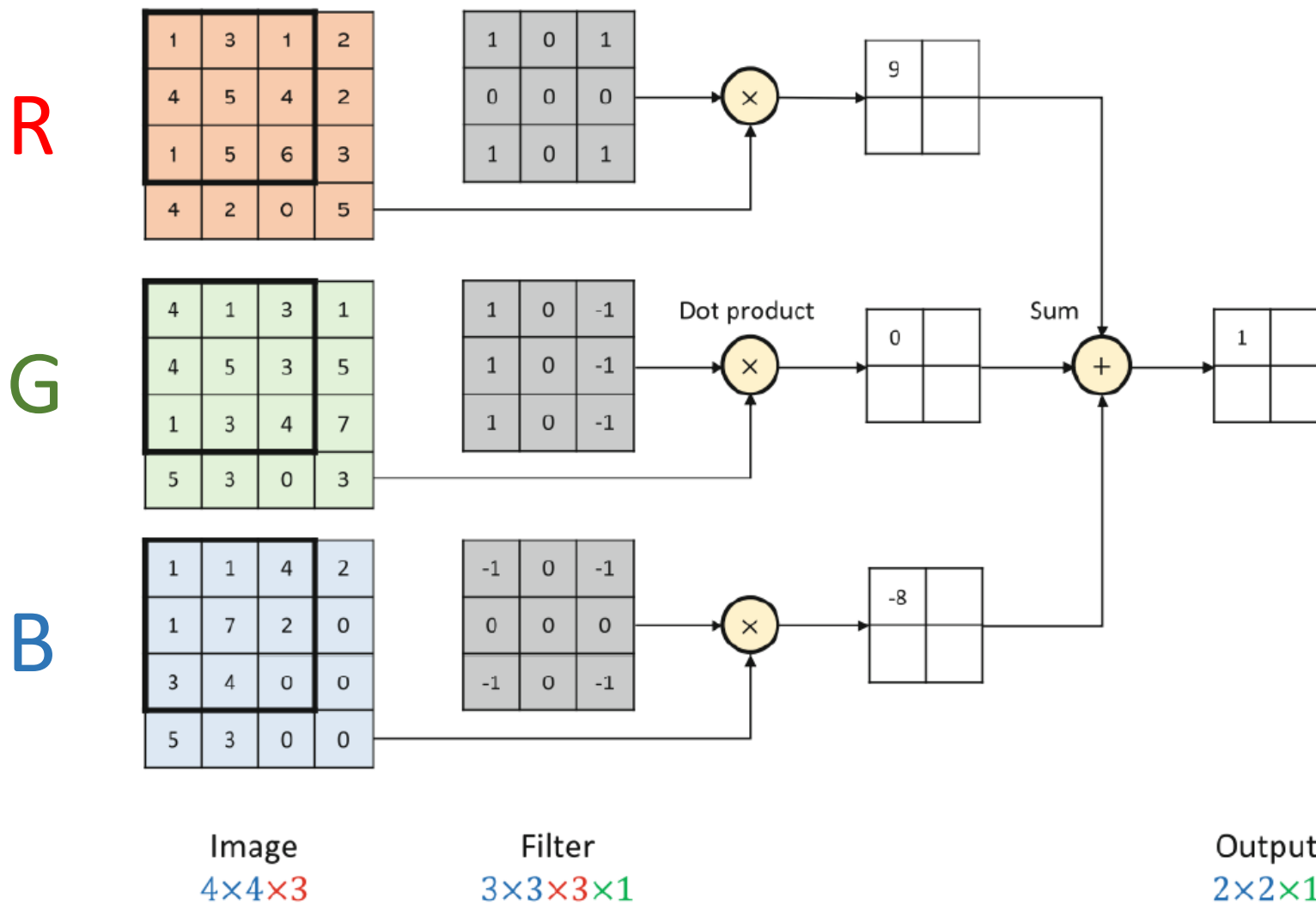
Les réseaux de neurones convolutifs (CNN)

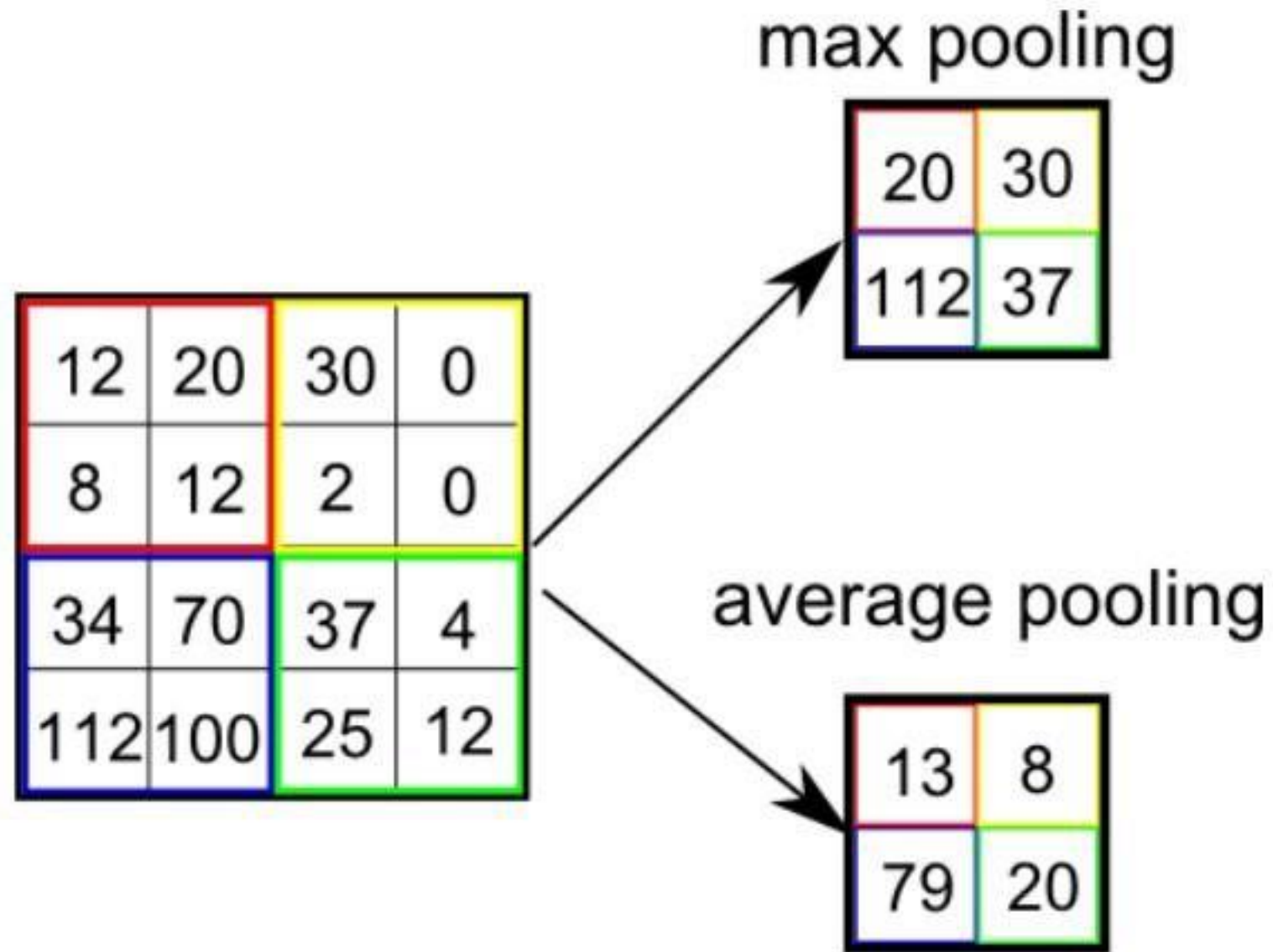
- Motivations sur les images :
 - Corrélation locale des données
 - Détection de formes
 - Invariance par translation : les formes peuvent se trouver à différents endroits de l'image
- La connaissance du problème influence la conception de l'architecture :
 - Les images possèdent une structure 2D
 - Forte corrélation locale dans les valeurs des pixels











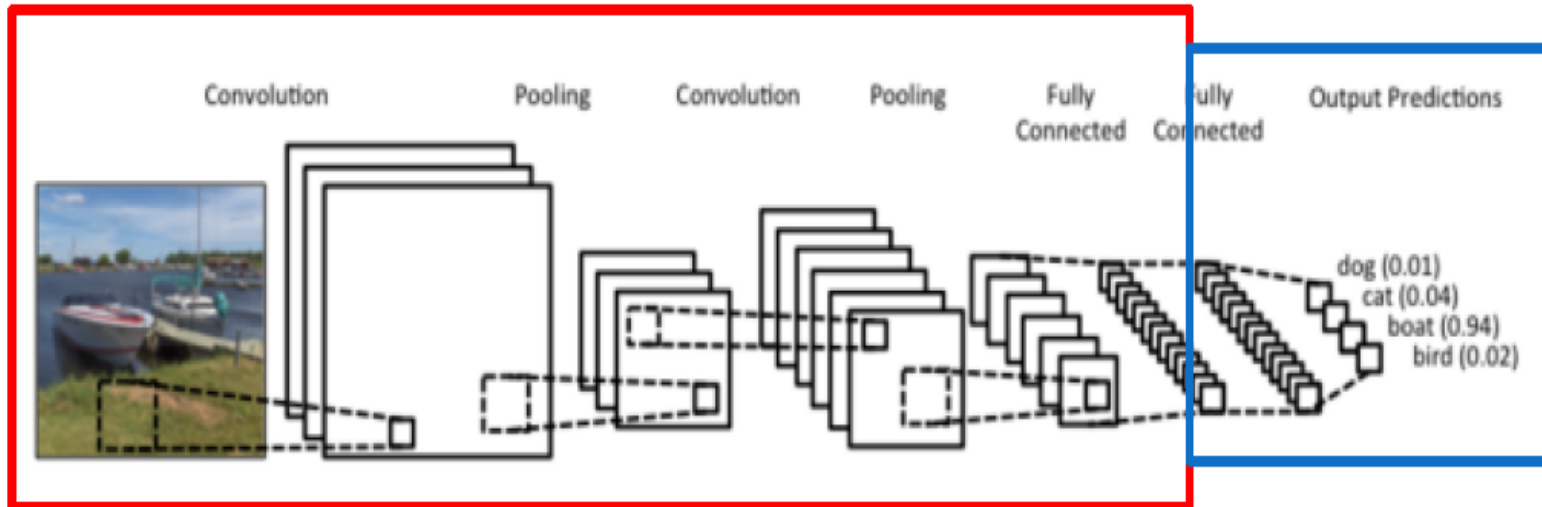
Les réseaux de neurones convolutifs (CNN)

- Sous Keras (en **gras** : ce qui est nouveau, en *italique*, ce qui est modifiable) :
- `keras.layers.Conv2D(filters, kernel_size, strides=(1, 1), padding='valid', activation=None, use_bias=True, kernel_initializer='glorot_uniform', bias_initializer='zeros', kernel_regularizer=None, bias_regularizer=None)`
- `keras.layers.MaxPooling2D(pool_size=(2, 2), strides=None, padding='valid')`
- `keras.layers.UpSampling2D(size=(2, 2), interpolation='nearest')` : opération inverse du pooling
- Ces trois couches existent aussi en version 1D et 3D
- `keras.layers.Flatten()` (pour mettre les données sur une seule ligne avant des fully-connected layers)
- [Conv2D layer \(keras.io\)](https://keras.io)

Transfer Learning

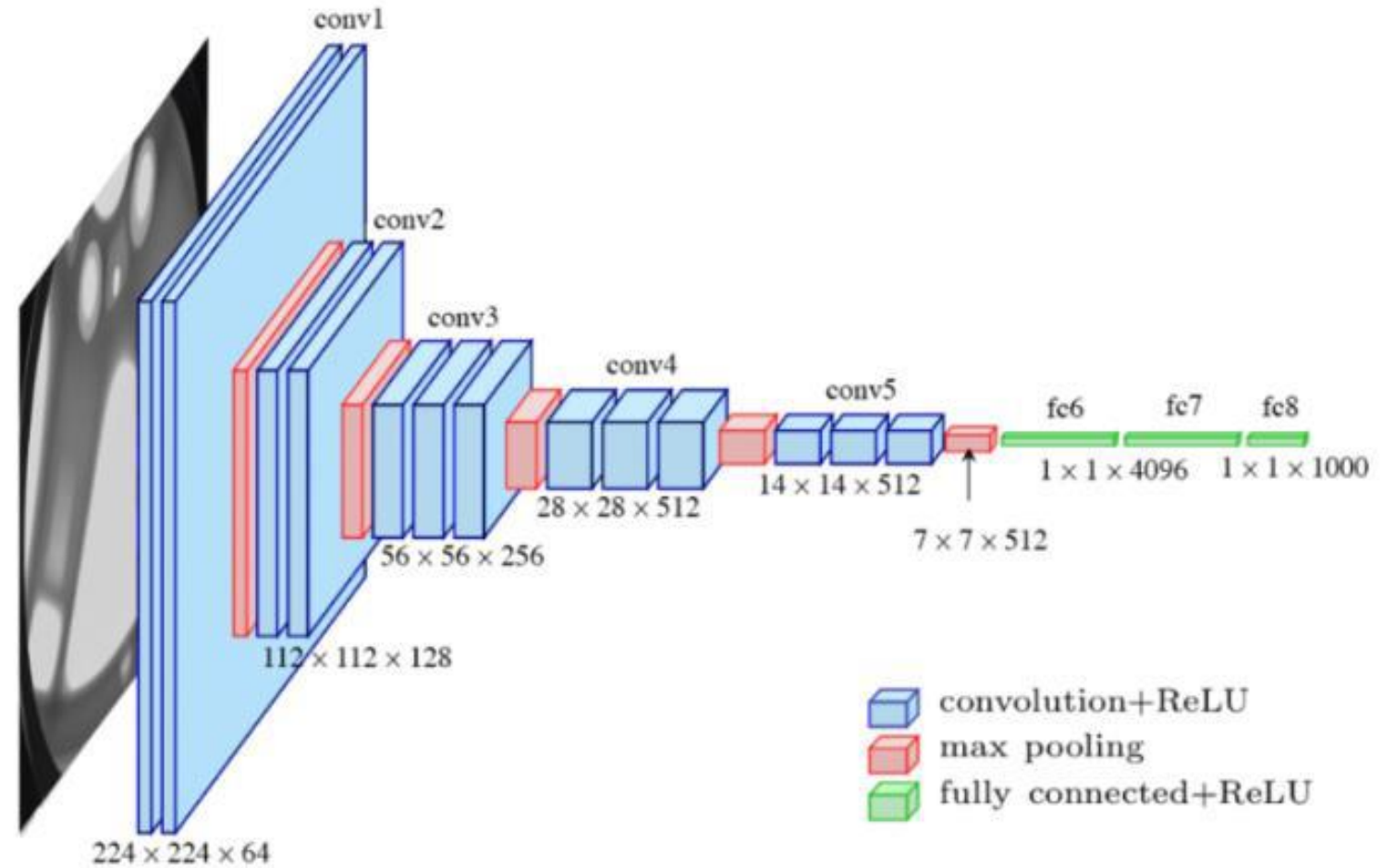
- ▶ Idée : conserver l'extraction des caractéristiques apprise sur d'autres problématiques
 - ▶ Revient à conserver des couches de convolution apprises sur un problème similaire
 - ▶ On ne change que la (ou éventuellement les) dernière(s) couche(s) d'identification

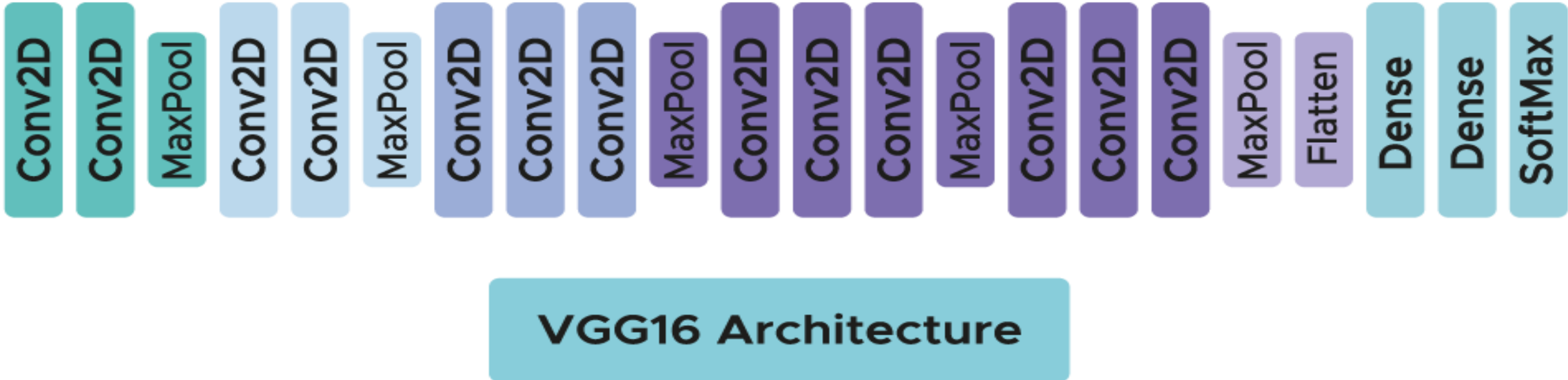
Fixée
(déjà
apprise)



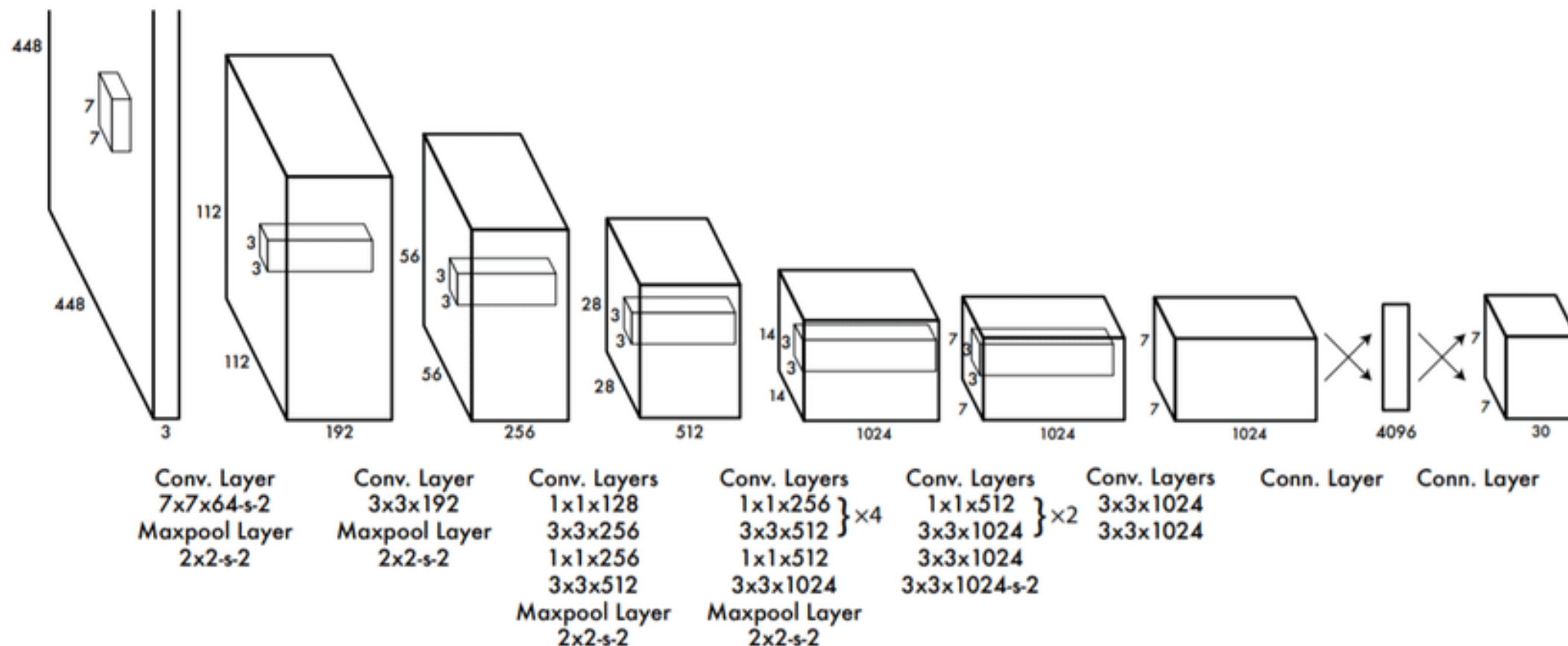
On apprend
seulement ces
couches

VGG16

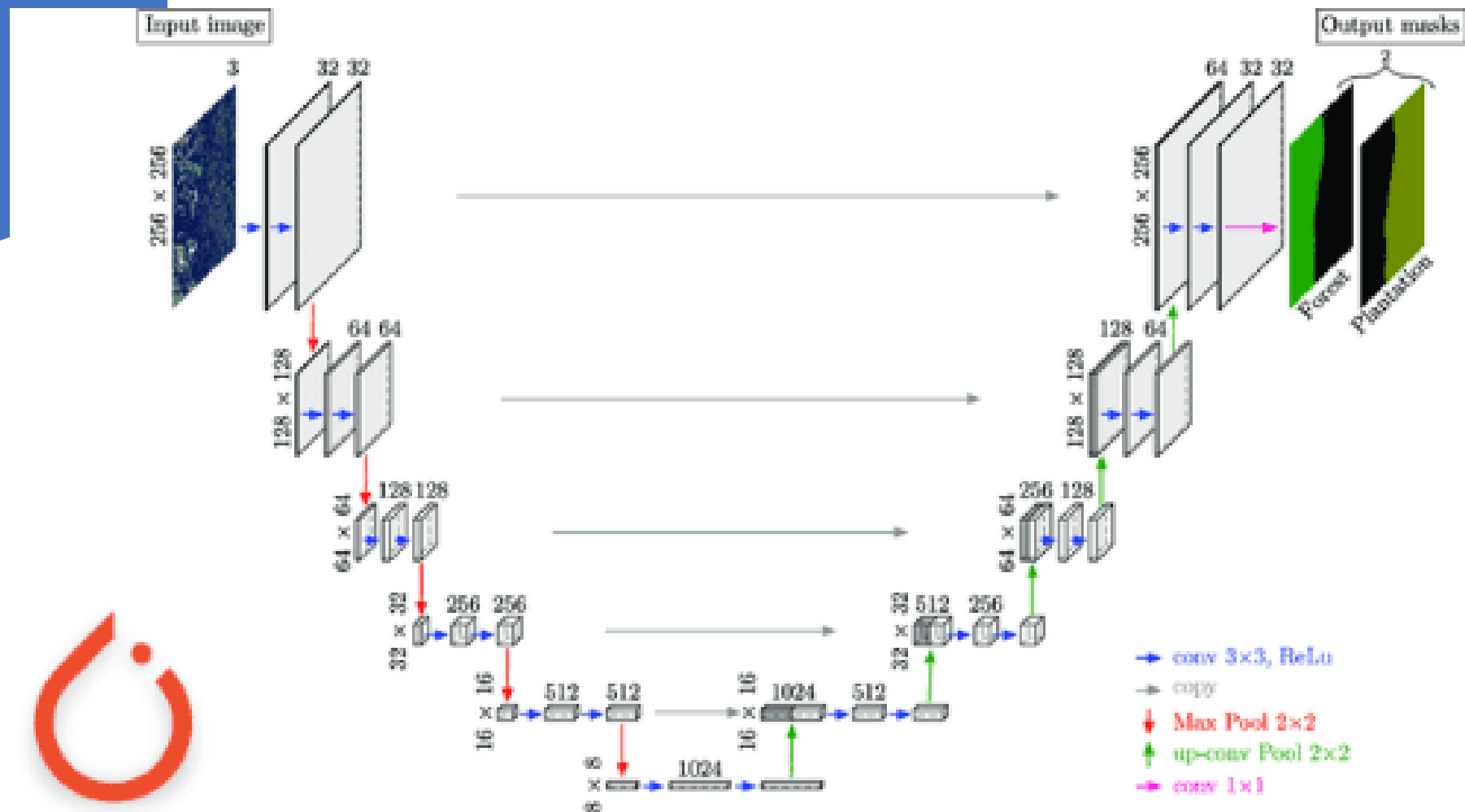




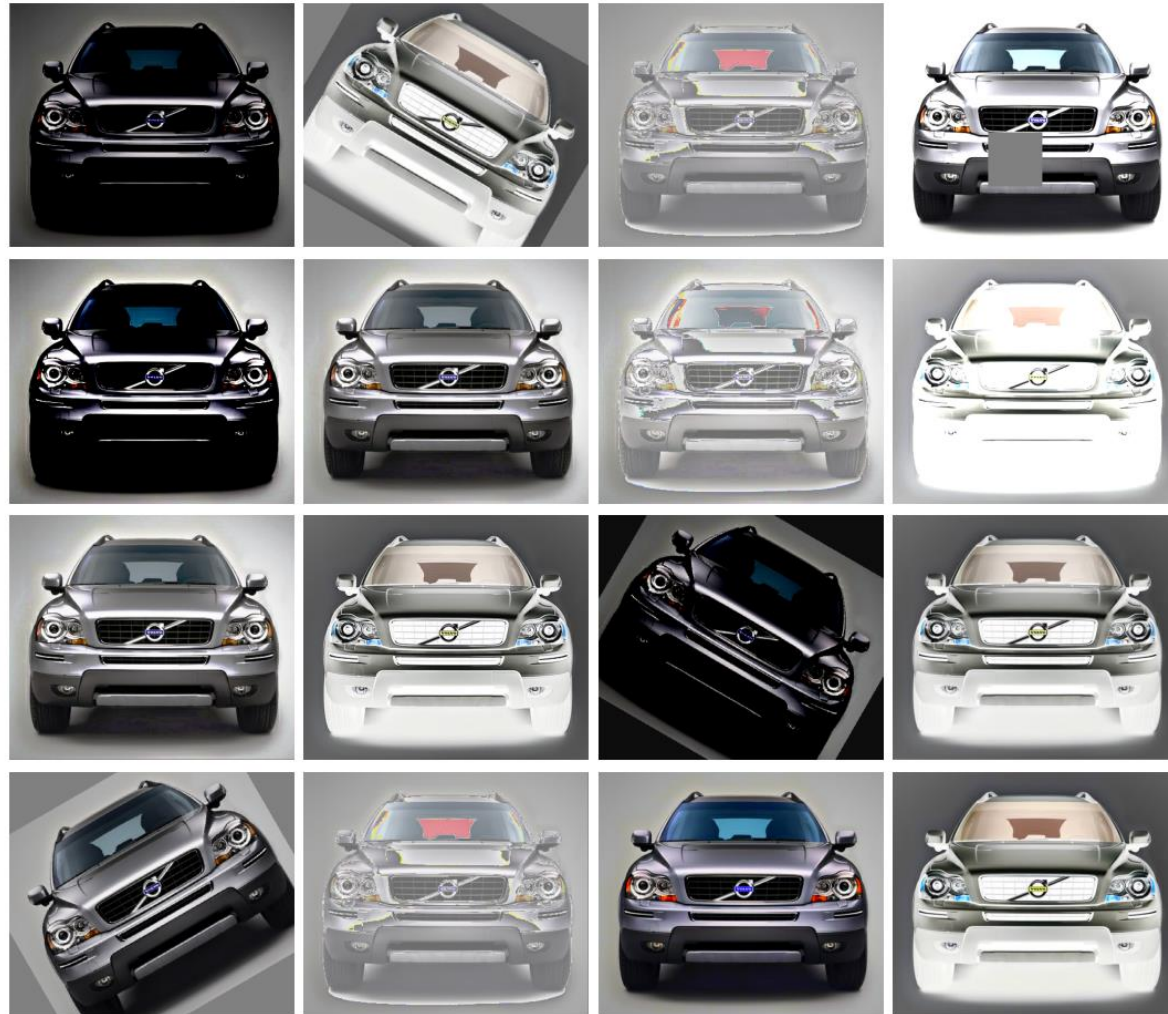
YOLO



U-Net



Data Augmentation



Data Augmentation

```
from keras.applications.vgg16 import preprocess_input  
from keras.preprocessing.image import ImageDataGenerator
```

```
train_data_generator = ImageDataGenerator(  
    Preprocessing_function = preprocess_input,  
    # data augmentation  
    rotation_range = 10,  
    width_shift_range = 0.1,  
    height_shift_range = 0.1,  
    zoom_range = 1.1,  
    horizontal_flip = True  
)
```

```
test_data_generator = ImageDataGenerator(  
    preprocessing_function = preprocess_input)
```