**University Mohamed Boudiaf of M'sila**
**Faculty of Mathematics and Computer Science**

# Introduction to Artificial Intelligence

## Chapitre 3. Apprentissage Profond

***Dr. SAID KADRI***

**Associate Professor "Class A"**

**Department of Computer Science, Faculty of Mathematics and Informatics,**

**University Mohamed Boudiaf of M'sila**

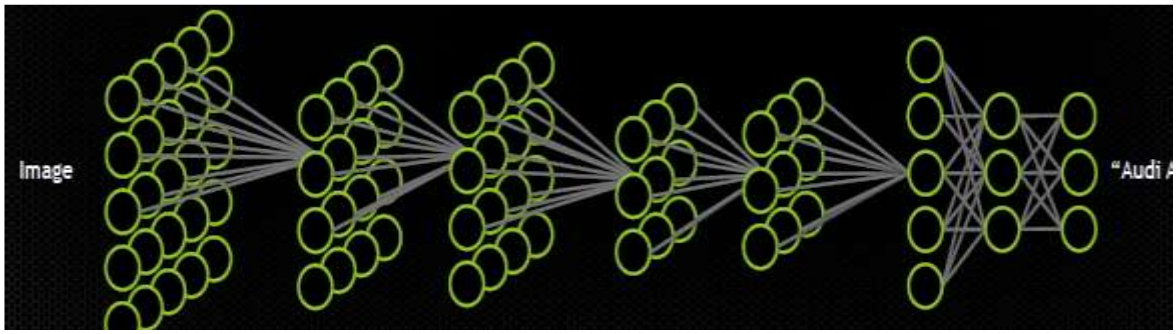
**E-mail:** kadri.said28@gmail.com

**Website:** https://kadrisaid28.wixsite.com/sgadri

**2020– 2021**

# 3. Apprentissage Profond (Deep Learning)

- L'apprentissage profond ou approfondi (Deep learning DL) est un sous domaine de l'apprentissage automatique (Machine Learning ML) pour lequel un modèle apprend à effectuer des taches diverses de classification en se basant directement sur des images ou des textes.
- En Deep learning, on utilise uniquement des réseaux de neurones artificiels RNA pour construire le modèle approprié.
- Le terme "Deep" vient de l'utilisation de plusieurs couches de neurones, alors plus qu'on a des couches plus qu'on a un réseau profond.
- Un réseau de neurone traditionnel contient uniquement 2 ou 3 couches, tandis qu'un réseau profond peut avoir des centaines ou des milliers de couches.
- Une architecture de Deep Learning est constituée de: une couche d'entrée, plusieurs couches cachées, est une couche de sortie. Les différentes couches sont interconnectées de sorte que les sorties de chaque couche deviennent des entrées pour la couche suivante.




- Le Deep Learning a prouvé son succès dans plusieurs domaines vitaux, tels que :
  - Classification des images médicales
  - Reconnaissance de la parole
  - Reconnaissance des caractères
  - Reconnaissance et détection visuelle des objets

## Difference entre Deep Learning et Machine Learning

- L'apprentissage profond (Deep learning) est un sous domaine de l'apprentissage automatique (machine learning).
- En machine learning, on effectue la tâche d'extraction des éléments caractéristiques (feature extraction) manuellement, par contre, en deep learning, c'est le réseau construit qui se charge de cette tâche d'une manière automatique lors du traitement de milliers ou de millions d'occurrences (ex : pixels d'une image) du dataset en plusieurs étapes.

## Motivation de l'utilisation du Deep Learning

- Large disponibilité de Datasets.
- Evolution des équipements graphiques (cartes GPU)
- Prix relativement raisonnable du matériel.
- Evolution des techniques et des méthodes de traitement

## Champs d'application:

- Classification des images.
- Traduction automatique : d'une langue à une autre.
- Jeux (ex: jeu d'échec).
- Reconnaissace des visages
- Reconnaissance des gestes.
- Reconnaissance de la parole (Conversion de la parole en textes).
- Recherche de vidéos (recherche& Analyse)
- Moteurs de recommandation (Robots, Cars)
- Indexation et Recherche (ex: recherche de documents sur Web)

## Exigences de Deep Learning

- Plus de données (Large datasets, Big Data)
- Meilleurs modèles d'apprentissage (e.g., DNN, CNN, RNN, Auto-encodeurs, …)
- Accélérateurs GPU puissants (Nvidia drive™ px auto-pilot car computer, tegra x1 mobile superchip 256-core GPU | 8-core 64-bit CPU | 1 Tera-FLOPS)

## Les plates-formes de Deep Learning

### 1. Caffe

- Interface: C++, Python
- L'une des premières plates-formes qui supportent une GPU
- Orientée surtout au domaine du computer vision (and CNNs)

### 2. TensorFlow (Google, 2015)

- Interface: Python, C++
- Utilise des Multi GPUs
- Très populaire

### 3. Keras (Google, 2015)

- Installable sur Tensorflow and Theano
- Interface: Python
- But: Fournir une interface simple

### 4. Theano

- Interface: Python
- L'une des premières plates-formes qui supportent une carte GPU

**5. Autres platesformes:** Microsoft CNTK, Torch, MXN, Deeplearning4j

**Datasets utiliséss**

- ***ImageNet:*** 14 million images (21841 catégories), utilisée pour la reconnaissance de l'image.
- Architectures des réseaux de neurones disponibles: AlexNet (2012), ZFNet (2013), GoogleNet (2014), ResNet (2015), CUImage (2016), Squeeze and excitation Networks (2017)

## Quelques types de réseaux de neurones profonds

### *1. Un simple réseau de neurones profond (Deep Neural Networks DNN)*

Un simple réseau de neurones profond appelé "Deep Neural Network" est un réseau de neurones avec plus de deux couches (une couche d'entrée, plusieurs couches cachées, une couche de sortie). Les réseaux de neurones de ce type utilisent une modélisation mathématique sophistiquée pour traiter les données. Ces modèles ajustent les sorties en fonction des entrées.

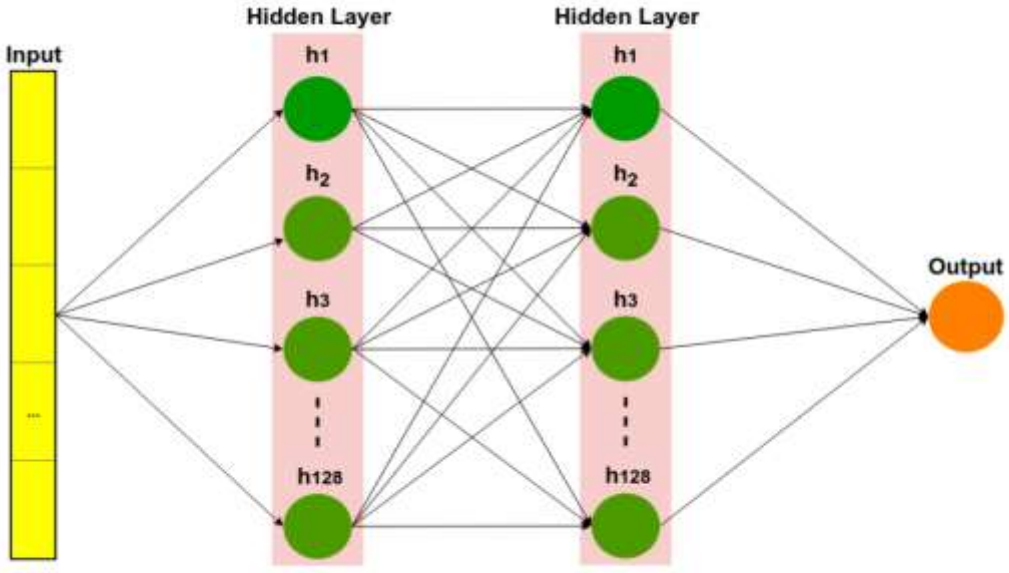



**Figure 2.** Deep neural network (DNN).

## *2. Réseaux de neurones convolutifs (Convolutional Neural Networks CNN)*

Un réseau de neurones convolutif ou de convolution (Convolutional Neural Network CNN) est un modèle spécial des réseaux de neurones de type « feed-forward neural network » utilisé dans différents domaines tels que : computer vision CV, recommender systems RS, et Natural Language Processing NLP. C'est une architecture profonde de réseaux de neurones. Un reseau CNN est composé de plusieurs couches de différents types: des couches de convolution (convolutional layers), des couches d'échantillonnage (pooling or subsampling layers) et une couche totalement connectée (Fully-Connected Layer). Le rôle des couches de convolution est de filtrer les données qu'elles reçoivent pour extraire les éléments les plus pertinents, par contre les couches d'échantillonnage (Pooling Layers) est de réduire la résolution (la dimension) des éléments sélectionnés par les couches de convolution ce qui réduit la complexité, éviter le phénomène de overfitting et par conséquent augmente la robustesse et les performances du réseau CNN surtout devant les parasites.

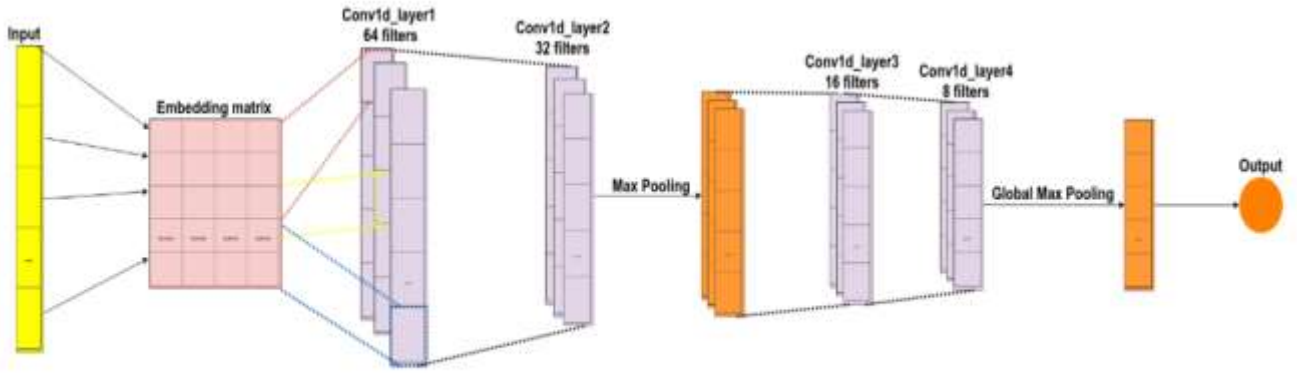



Figure 3. A convolutional neural network.

### *3. Réseaux de neurones récurrents (Recurrent Neural Networks RNN)*

Les réseaux de neurones récurrents sont une classe de réseaux de neurones dont les connections entre les neurones forment un cycle orienté ce qui crée des boucles de rétroaction dans le réseau RNN. La fonction principale d'un RNN est le traitement des informations séquentielles en se basant sur la mémoire interne capturée par les cycles orientés. Contrairement aux réseaux de neurones simples, les RNNs peuvent rappeler les calculs antérieurs des informations et les réutiliser en les appliquant sur l'élément suivant dans la séquence des entrées. Un type spécial des RNNs est le réseau appelé LSTM (Long Short-Term Memory), qui est capable d'utiliser une mémoire longue comme fonction d'activation dans les couches cachées. L'architecture LSTM était proposée par Hochreiter and Schmidhuber (1997).

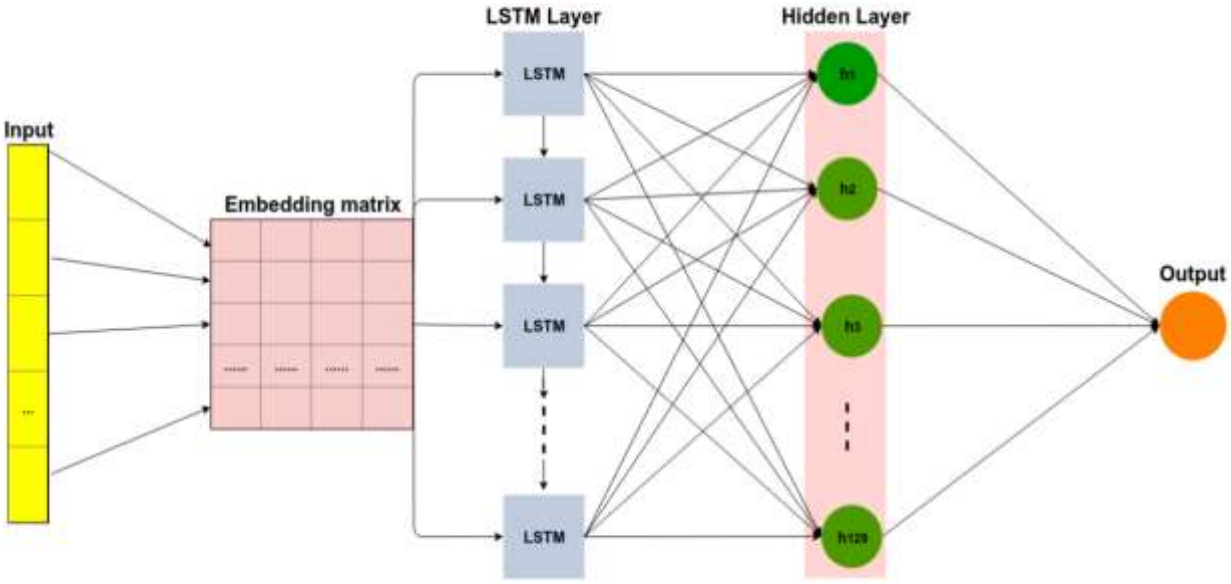



**Figure 4.** A long short-term memory network. LSTM, long short-term memory.

Dans Fig 4, les données d'entrée sont prétraitées pour les redimensionner (le même principe des réseaux CNNs). La couche suivante est composée des LSTM constitués de 200 neurones. La couche finale est une couche totalement connectée (fully connected layer), composée de 128 neurones pour la classification des textes. La dernière couche utilise une function d'activation sigmoid pour réduire un vecteur de dimension 128 à une seule sortie, sachant qu'on a deux classes à prédire (positive, negative).

### *4. Réseaux de neurones récursifs (Recursive Neural Network RecNN):*

Est un réseau de neurones qui peut être considéré comme une généralisation des réseaux RNNs. Les réseaux RecNNs sont généralement utilisés pour apprendre une structure de graphe acyclique orienté à partir de données. Les vecteurs d'état cachés des nœuds fils gauches et droits dans le graphe peuvent être utilisés pour calculer le vecteur d'état caché du nœud actuel.

### *5. Réseaux de croyance profonds (Deep Belief Network DBN)*

Un réseau DBN contient plusieurs couches constituées de modèles graphiques ayant des arêtes orientées et non orientées à la fois. Chaque réseau est composé de plusieurs couches dont chacune est constituée d'une collection de neurones reliés aux neurones de la couche suivante, mais non connectés entre eux. L'apprentissage d'un réseau DBN est effectué par un algorithme spécifique appelé « greedy layer-wise learning algorithm/ algorithme d'apprentissage gourmand par couche ».

### *6. Hybrid Deep Learning Network HDLN*

Une autre catégorie de réseau de neurone est appelé "réseau de neurones profonde hybride » (Hybrid Deep Learning Network), qui combine deux modèles profonds ensemble ou plus, tels que: un modèle CNN, un modèle LSTM, un modèle probabiliste PNN ou même un modèle RBM (RBM).

# Réseaux de neurones convolutifs (Convolutional Neural Networks CNNs)

UN réseau CNN est un algorithme de deep learning qui a souvent comme entrée, une image et qui donne plus d'importance aux différents aspects/objets dans l'image et d'être capable de les différentier les uns des autres.

Une utilisation classique des CNNs est de classifier une collection d'images dans le but de détecter ce que représente chaque image (voiture, humain, chat, bateau, avion, …)

## Motivations des CNNs

Les images utilisées ont généralement des hautes résolutions (ex : 224X224 pixels ou plus), d'où la conception d'un RNA simple pour traiter des images 224X224 colorées (les 3 couleurs de base RGB) demande l'introduction de 223X224X3=150.528 éléments en entrée du modèle !!! et si on a 1024 neurones dans la première couche cachée de notre modèle ➔on doit calculer (apprendre) 150.528x1024 poids (plus de 150 millions poids !!!) uniquement pour la première couche cachée !!! ce qui est très couteux en mémoire et en temps d'apprentissage (il parait impossible d'apprendre un tel réseau comme ceci).

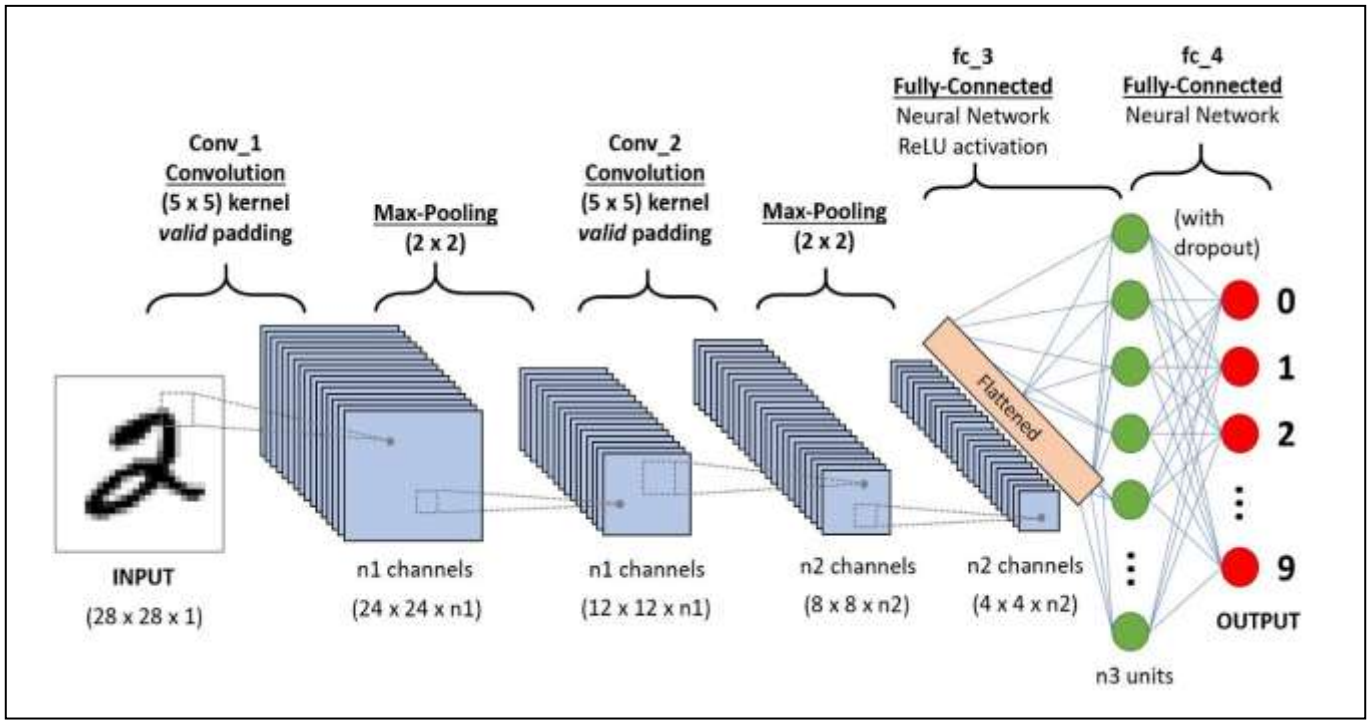
A cet effet, on peut mettre l'accent sur les parties (pixels) les plus significatives de l'image ce qui est réalisé par les CNNs.

## Domaines d'application des CNNs

1. Classification des images (Computer Vision)
2. Détection des objets (Computer Vision)
3. Traitement de langages (NLP)

**Architecture d'un réseau CNN**

Un modèle CNN est constitué de plusieurs types de couches, notamment : des couches de convolution (Conv Layers), des couches d'échantillonnage (Pooling Layers), une couche pleinement connectée (Fully Connected Layer).




## Principe de la convolution :

La couche de convolution Conv.Layer du réseau CNN est basée sur l'opération mathématique convolution. Il s'agit d'utiliser un ensemble de filtres ou chaque filtre (appelé aussi noyau) est une matrice de deux dimensions 2D contenant des valeurs entières

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

**Un filtre 3X3**

On applique ce filtre sur une image en entrée pour produire une image réduite (c'est très important) en convoluant le filtre avec l'image suivant la procédure suivante :

1. Superposer le filtre sur l'image d'entrée à un endroit bien déterminé.
2. Effectuer une multiplication élément par élément entre les valeurs du filtre et les valeurs correspondantes de l'image.
3. Calculer la somme de tous les produits élémentaires. Le résultat de cette somme est la valeur de sortie pour le pixel de l'image résultante en sortie.
4. Déplacer le filtre sur l'image vers un autre endroit
5. Répéter (2) et (3)

## Avantages d'une couche CONV

1. Réduire les dimensions des images d'entrée pour faciliter leur traitement (on peut garder les dimensions si c'est nécessaire).
2. Extraire les éléments les plus significatifs dans l'image (arêtes, couleurs, …) sans perdre l'information.

## Exemple

On considère une image d'entrée 4x4 à niveau de gris (noir et blanc) et un filtre 3x3 comme suit :

| | | | |
|---|---|---|---|
| 0 | 50 | 0 | 29 |
| 0 | 80 | 31 | 2 |
| 33 | 90 | 0 | 75 |
| 0 | 9 | 0 | 95 |

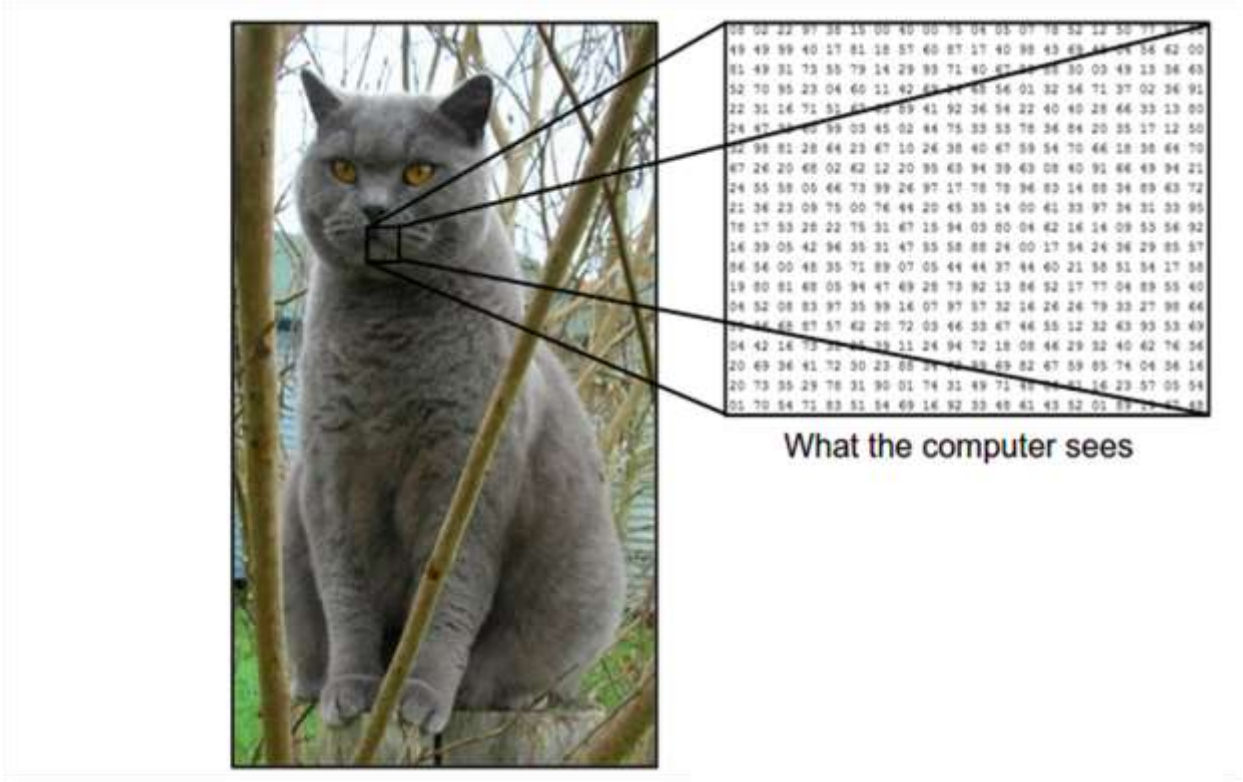
**Une Image 4X4**

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

**Un filtre 3X3**

Les nombres dans l'image représentent les intensités des pixels (les couleurs). E.g., la valeur 0 représente la couleur noire, tandis que la valeur 255 représentent la couleur blanche.

En général, les pixels ayant des valeurs élevées représentent la partie la plus claire de l'image et les pixels ayant des valeurs basses représentent les parties les plus sombres.



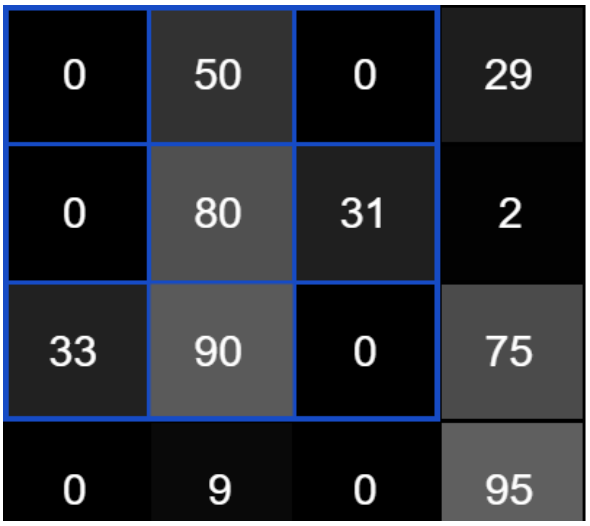

**Représentation d’une image sur la machine**

On veut convoluer l'image et le filtre pour avoir une image de sortie de dimension 2x2

| | |
|---|---|
| ? | ? |
| ? | ? |

**Une image de sortie 2X2**

Comme s'est expliqué précédemment, on superpose notre filtre sur le coin supérieur gauche de l'image




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

Ensuite nous effectuons un produit élément par élément entre les valeurs de l'image et les valeurs du filtre qui se chevauchent en partant du coin supérieur gauche et en allant vers la droite, puis vers le bas.

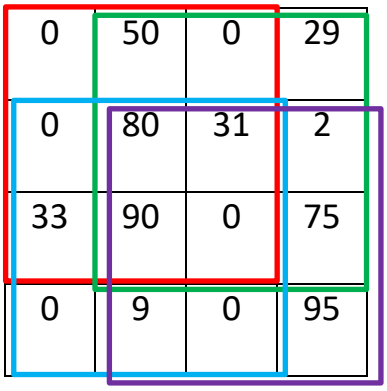
0x(-1) + 50x0 + 0x1 + 0x(-2) + 80x0 + 31x2 + 33x(-1) + 90x0 + 0x1

0+0+0+0+0+62-33+0+0 = 29

| | |
|---|---|
| 29 | ? |
| ? | ? |

**Une image de sortie 2X2**

On fait la même chose en déplaçant le filtre horizontalement et verticalement sur l'image d'entrée en calculant les sommes correspondantes pour trouver les valeurs des pixels destination de l'image de sortie.




**Une Image 4X4**

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

**Un filtre 3X3**

| | |
|---|---|
| 29 | -192 |
| -35 | -22 |

**Une image de sortie 2X2**

## Types de filtres utilisés dans la convolution ?

Le type des filtres que nous choisissons, permet de détecter les arêtes verticales ou horizontales de l'image.

| | | |
|---|---|---|
| 1 | 0 | -1 |
| 1 | 0 | -1 |
| 1 | 0 | -1 |

**Un filtre 3X3 vertical**

| | | |
|---|---|---|
| 1 | 1 | 1 |
| 0 | 0 | 0 |
| -1 | -1 | -1 |

**Un filtre 3X3 horizontal**

## Exemples d filtres les plus utilisés

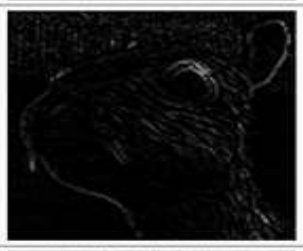
| | | |
|---|---|---|
| -1 | 0 | 1 |
| -2 | 0 | 2 |
| -1 | 0 | 1 |

**Un filtre de Sobel vertical**

| | | |
|---|---|---|
| 3 | 0 | -3 |
| 10 | 0 | -10 |
| 3 | 0 | -3 |

**Un filtre de Scharr**

## Influence du filtre

| Operation | Filter | Convolved Image |
|---|---|---|
| **Identity** | $\begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix}$ |  |
| **Edge detection** | $\begin{bmatrix} 1 & 0 & -1 \\ 0 & 0 & 0 \\ -1 & 0 & 1 \end{bmatrix}$ |  |
| | $\begin{bmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{bmatrix}$ |  |
| | $\begin{bmatrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{bmatrix}$ |  |
| **Sharpen** | $\begin{bmatrix} 0 & -1 & 0 \\ -1 & 5 & -1 \\ 0 & -1 & 0 \end{bmatrix}$ |  |
| **Box blur** (normalized) | $\frac{1}{9}\begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$ |  |
| **Gaussian blur** (approximation) | $\frac{1}{16}\begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix}$ |  |

**Que donne la convolution d'une image ?**

Par application du filtre vertical de Sobel 3x3 que nous avons déjà utilisé précédemment sur l'image, nous obtenons :

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

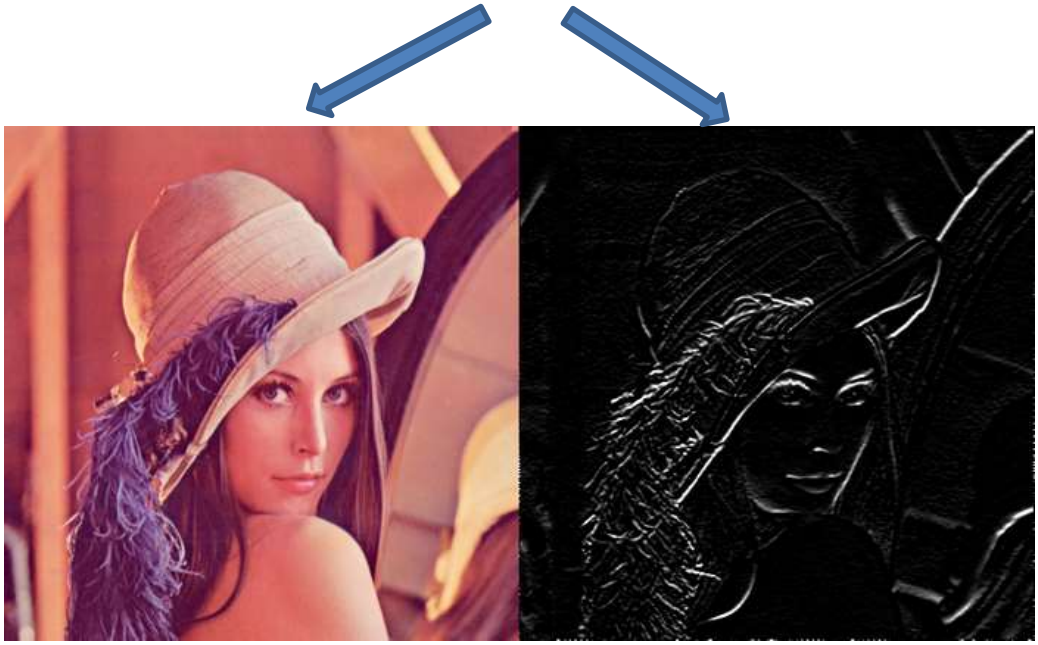
**Un filtre de Sobel vertical**

**Une image convoluée avec un filtre vertical de Sobel**

De même, si on applique un filtre horizontal de Sobel, on obtient l'image suivante :

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

**Un filtre de Sobel horizontal**



**Une image convoluée avec un filtre horizontal de Sobel**

On peut voir facilement que c'est ce que se passe ? les filtres de Sobel sont des détecteurs de contours. Le filtre de Sobel vertical détecte les contours verticaux, par contre le filtre horizontal de Sobel détecte les contours horizontaux. Les images résultantes sont maintenant clairement interprétées, un pixel clair (ayant une valeur élevée) dans l'image résultante indique qu'il y a un contour fort autour cette zone dans l'image d'entrée (originale).

## Rembourrage (padding)

Rappelons que la convolution d'une image 4x4 avec un filtre 3x3 peut produire une image de sortie 2x2.

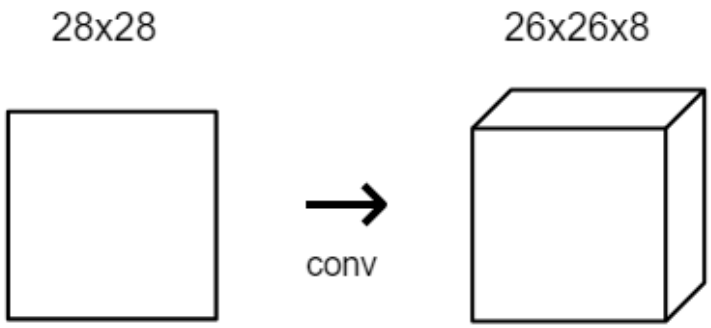
Si on veut obtenir une image de sortie de mêmes dimensions que celles de l'image d'entrée (4x4), nous devons ajouter des 0 autour l'image afin de pouvoir superposer le filtre à plusieurs endroits. Notons qu'un filtre 3x3 nécessite un pixel de rembourrage (padding). Dans ce cas on parle d'un

même embourrage (the same padding), car les images d'entrée et de sortie ont les mêmes dimensions. On peut aussi avoir un rembourrage qui réduit les dimensions de l'image de sortie, c'est le cas de rembourrage valide.

### *1. Les couches de convolution (Conv.Layers)*

Comme c'est expliqué précédemment, les CNNs incluent des couches de convolution (Conv.Layers) qui utilisent un ensemble de filtres pour transformer les images d'entrée en images de sortie. Alors, le paramètre principal d'une couche Conv est le nombre de filtres utilisés.

Par exemple un modèle CNN appliqué sur une image MNIST (représentant un chiffre), on utilise une petite couche CONV initiale avec 08 filtres, ce qui transforme l'image d'entrée 28x28 en un volume de sortie 26x26x8 (08 images 26x26 empilées ensemble).




### *2. Couches d'échantillonnage/Mise en commun (Pooling/Sampling Layers)*

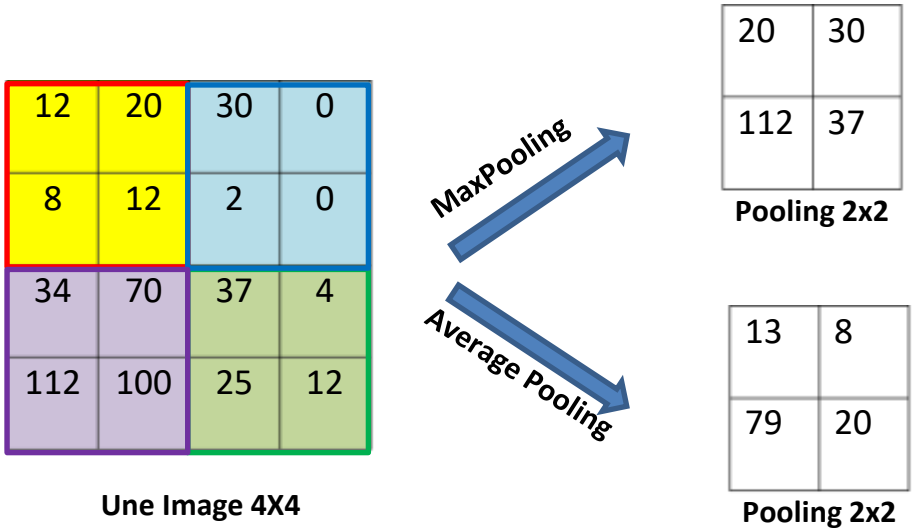
Généralement, les pixels voisins dans les images ont des valeurs similaires d'où la convolution produit aussi des pixels de valeurs similaires. Alors pourquoi pas regrouper ces pixels similaires et prendre uniquement un pixel représentatif de ces pixels. C'est exactement l'idée derrière l'échantillonnage (pooling).

L'objectif est de réduire les dimensions des images résultantes dans la couche CONV afin de en ne prendre que les pixels significatifs et accélérer le traitement des images.

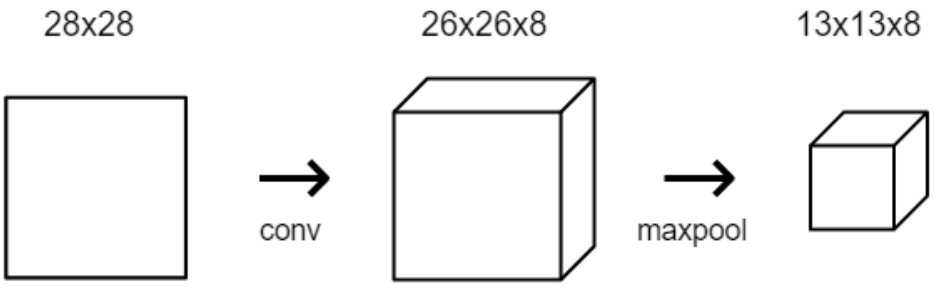
On distingue 02 types d'échantillonnage (pooling) :

1. Echantillonnage maximal (MaxPooling) : prendre la valeur maximale de chaque partie de l'image
2. Echantillonnage moyen (Average Pooling) : prendre la valeur moyenne de chaque partie de l'image.

**Exemple :** ci-après un exemple de pooling layer (MaxPooling – Average Pooling) de taille 2x2

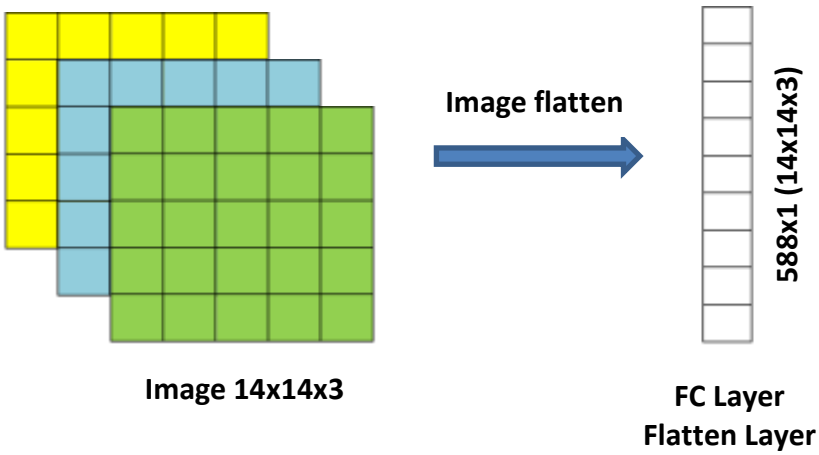



Notons que le pooling divise la largeur et la hauteur de l'image d'entrée par la taille du pool. Pour l'exemple précédent (le modèle MNIST), on place une couche Maxpooling de taille 2x2 juste après notre couche CONV initilale, ce qui transformera l'image d'entrée 26x26x8 en une sortie 13x13x8.




### 3. *Couche pleinement connectée (Fully Connected Layer/FC Layer/Flatten Layer)*

Cette couche transforme l'image en un vecteur monodimensionnel (1-dim vector) pour préparer à la phase de classification proprement dite.



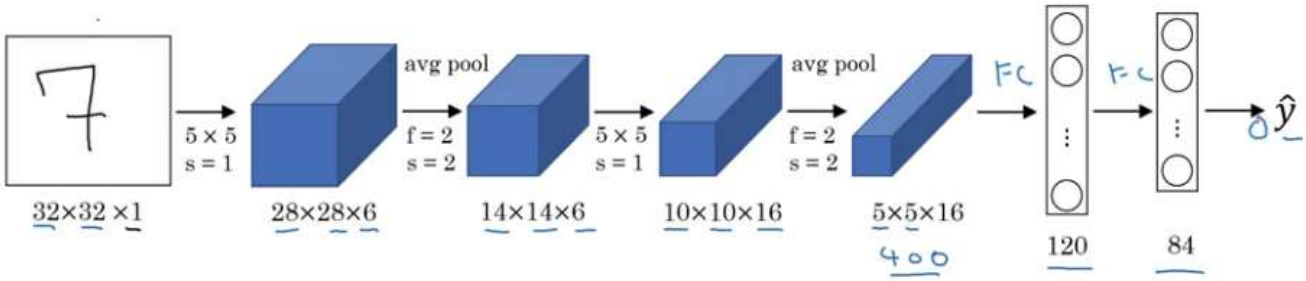
## Architectures populaires des réseaux CNNs

Dans cette section, on va voir les architectures les plus populaires des réseaux CNNs, à noter:

1. L'architecture LeNet-5
2. L'architecture AlexNet
3. L'architecture VGG

### *1. L'architecture LeNet-5*

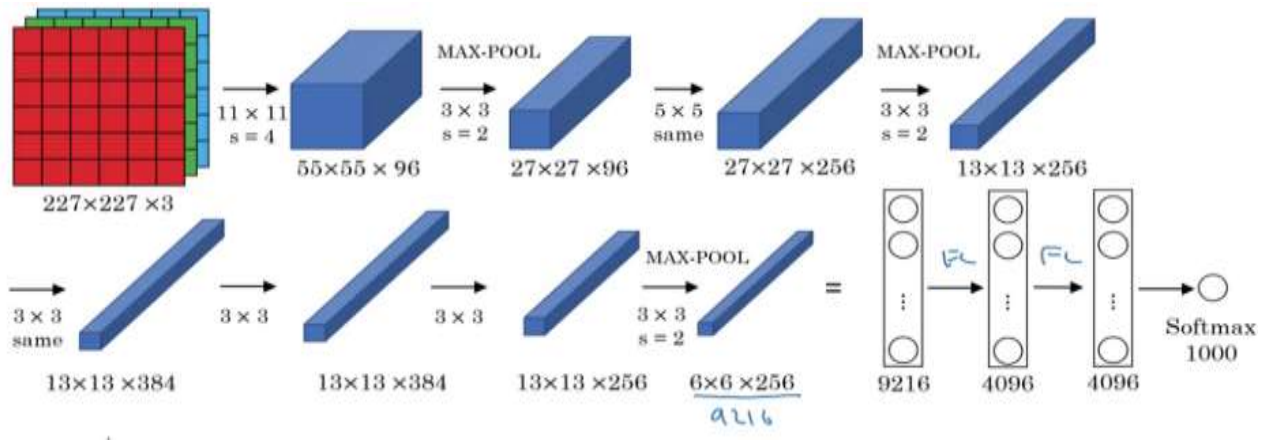
Peut être représentée comme suit :




- **Paramètres:** 60k
- **Couches utilisées:** Conv -> Pool -> Conv -> Pool -> FC -> FC -> Output
- **Fonction d'activation:** Sigmoid/tanh and ReLu

### *2. L'architecture AlexNet*

Peut être illustrée comme suit:



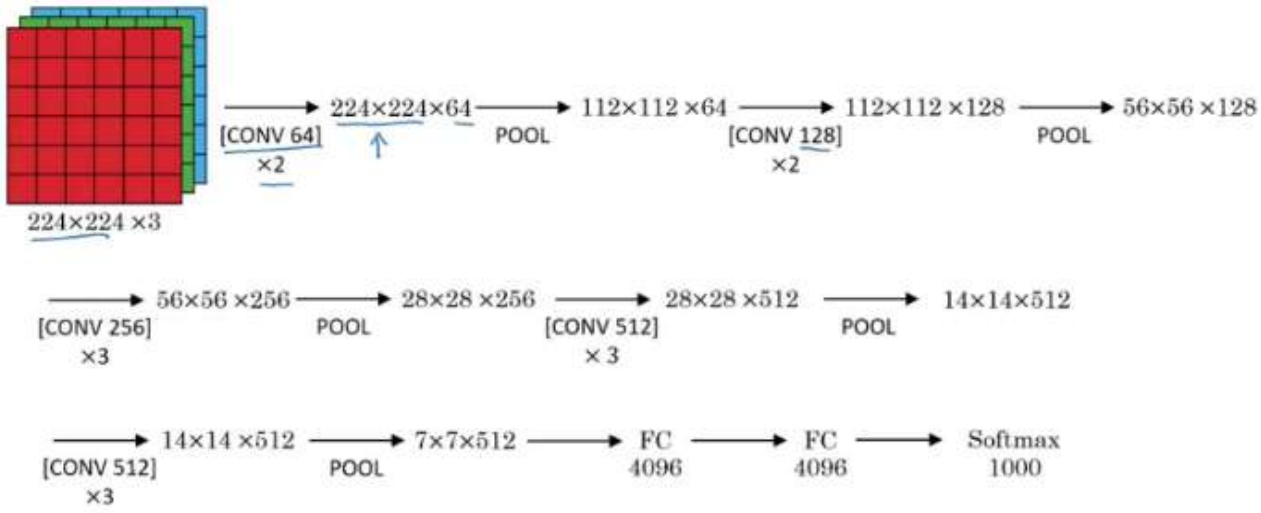
Ce modèle est similaire au modèle LeNet-5 avec plus de couches de convolution et de pooling

- **Paramètres:** 60 million
- **Couches utilisées :** comme s'est expliqué sur la figure.
- **Fonction d'activation:** ReLu

### *3. L'architecture VGG-16*

l'architecture VGG-16 est basée sur un reseau plus simple qui met l'accent sur des couches CONV de taille (utilise aussi the same padding). Une couche MaxPooling de taille 2x2 est utilisée après chaque couche de convolution. L'architecture VGG-16 peut être résumée comme suit:

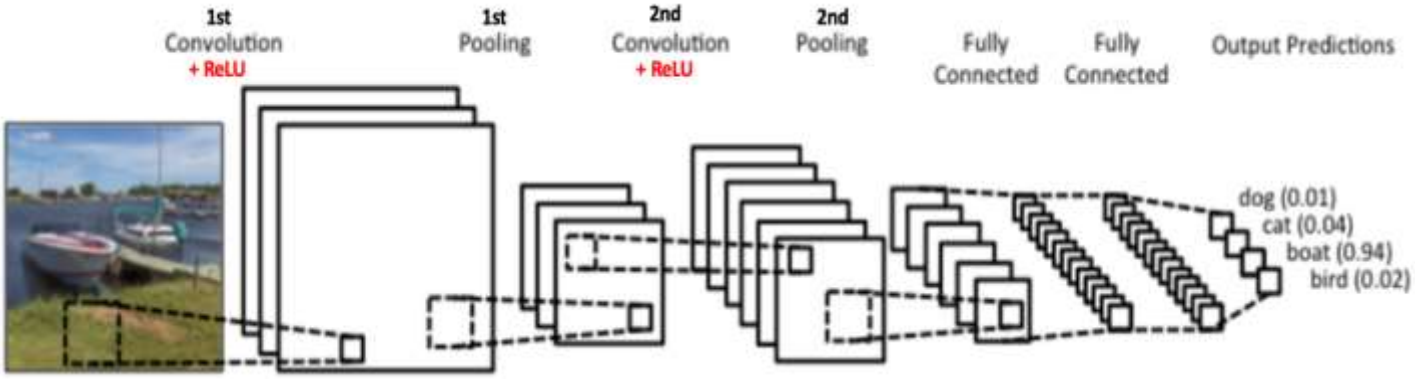
.



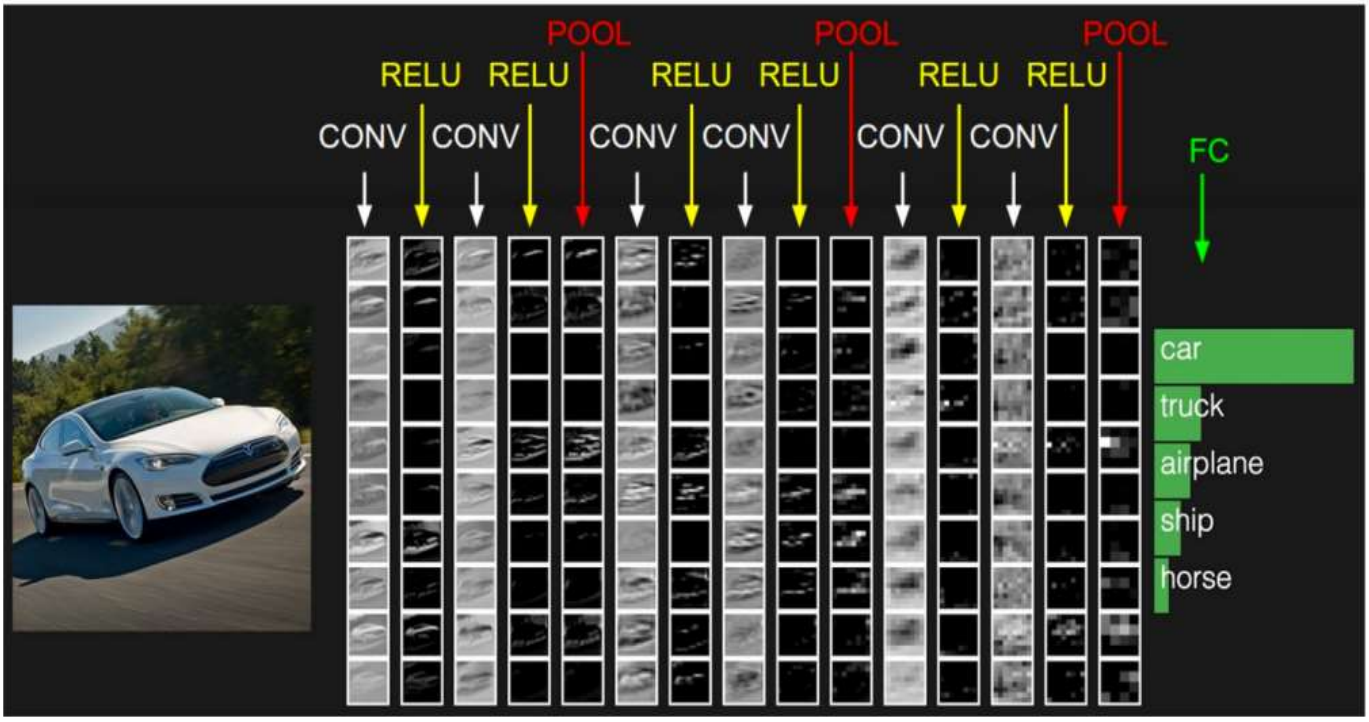

- **Paramètres:** 138 million

# Quelques illustraions de CNNs modèles

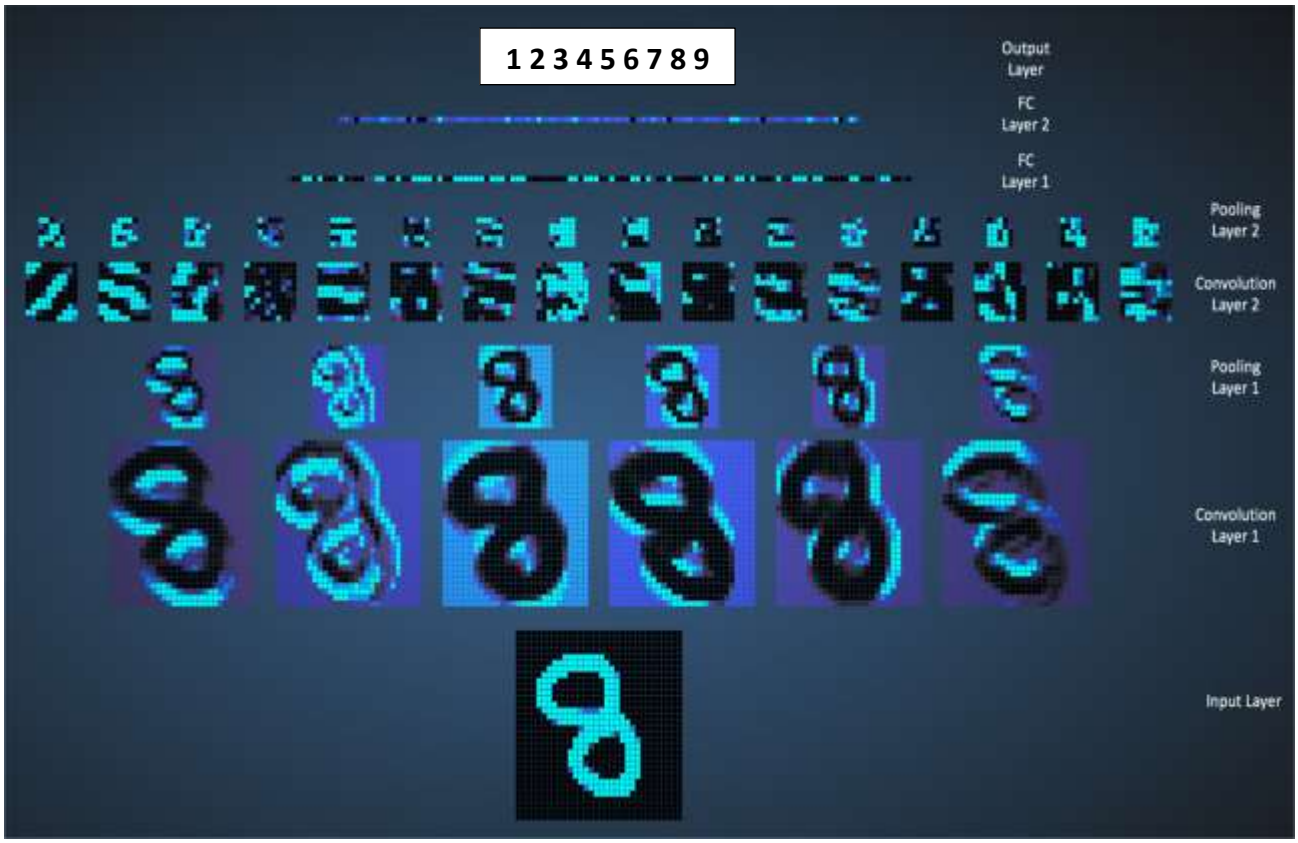
## *1. Classification des d'animaux*




## *2. Classification des objets différents*



## 3. *Reconnaissance des chiffres*



## Papiers et articles publiés

## Papiers Conférences

1. Said G. Diabetic Patient Classification: An Efficient Algorithm Based on Deep Learning Approach, International Conference on Science, Engineering&Technology (ICSET) held in Istanbul, Turkey, 20-21 December, 2019 (Winner of Best Conference Paper)
2. Said G. Efficient Arabic handwritten character recognition based on machine learning and deep learning approaches, The 2nd International Conference on Advances in Intelligent Systems, Soft Computing and Optimization Techniques 2020 (ICAISCO 2020) held in Penang, Malaysia, 01 - 02, April 2020 (**Scopus Indexed Conference**).
3. Said G., Erich N: Building Best Predictive Models Using ML and DL Approaches to Categorize Fashion Clothes, The 19th International Conference on Artificial Intelligence and Soft Computing ICAISC 2020 (H5-index = 20) (**Springer Conference**). held in Zakopane, Poland, 12 - 14, October 2020.
   Available on: https://link.springer.com/book/10.1007/978-3-030-61401-0
4. Hassina T, Said Gadr, Baya L, Nour El-Houda A, Nadjet B: Handwritten Digit Recognition: Developing an Efficient ML and DL Model to Recognize Handwritten Digits, CNIATI'21 Conférence Nationale sur l 'Intelligence artificielle et les technologies de l'information, 24 Mai 2021, University Chadli Ben Djedid, Al-Taref.
5. Said G, Sara O.M, Khadidja H, Safia C. An Efficient System to Predict Customers' Satisfaction on Touristic Services Using ML and DL Approaches. 22rd Arabic International Conference on Information Technology (ACIT 2021), 21-23 Dec, **IEEE Conference**, Quabos University, Sultanate Oman. Available on: https://ieeexplore.ieee.org/document/9677167.
   DOI: 10.1109/ACIT53391.2021.9677167
6. Gadri S., Adouane N.E. (2022) Efficient Traffic Signs Recognition Based on CNN Model for Self-Driving Cars. In: Vasant P., Zelinka I., Weber GW. (eds) Intelligent Computing & Optimization. ICO 2021. 30-31 Dec, 2021 (**LNNS Springer Conference)**. Lecture Notes in Networks and Systems, vol 371. Springer, Cham. https://doi.org/10.1007/978-3-030-93247-3_5
   Available on: https://link.springer.com/chapter/10.1007/978-3-030-93247-3_5
7. Gadri S., Chabira S., Ould Mehieddine S., Herizi Kh. (2022) Sentiment Analysis: Developing an Efficient Model Based on Machine Learning and Deep Learning Approaches. In: Vasant P., Zelinka I., Weber GW. (eds) Intelligent Computing & Optimization. ICO 2021. 30-31 Dec, 2021 (**LNNS Springer Conference)**. Lecture Notes in Networks and Systems, vol 371. Springer, Cham. https://doi.org/10.1007/978-3-030-93247-3_24
   Available on: https://link.springer.com/chapter/10.1007/978-3-030-93247-3_24

## Articles Journaux:

8. Said G. Efficient Arabic handwritten character recognition based on machine learning and deep learning approaches, Journal of Advanced Research in Dynamical & Control Systems, (Scopus indexed Journal), online version Vol. 12, 07-Special Issue, 2020, Pages: 9-17, DOI: 10.14311/NNW.2017.27.011, August 2020, DOI: 10.5373/JARDCS/V12SP7/20202076, ISSN 1943-023X, https://www.jardcs.org/archivesview.php?volume=3&issue=39
9. Said G. Diabetic patient classification: an efficient algorithm based on deep learning approach, International Journal of Advances in Electronics and Computer Science IJAECS, ISSN(p): 2394-2835, Volume-7, Issue-3, April 2020, http://iraj.in
10. Said G. Developing an efficient Predictive model based on ML and DL approaches to detect diabetes, The International Journal of Computing and Informatics Informatica, ISSN(p): 0350-5596, Volume-45, Issue-3, 2021, https://www.informatica.si/index.php/informatica/article/view/3041