CybSPEED

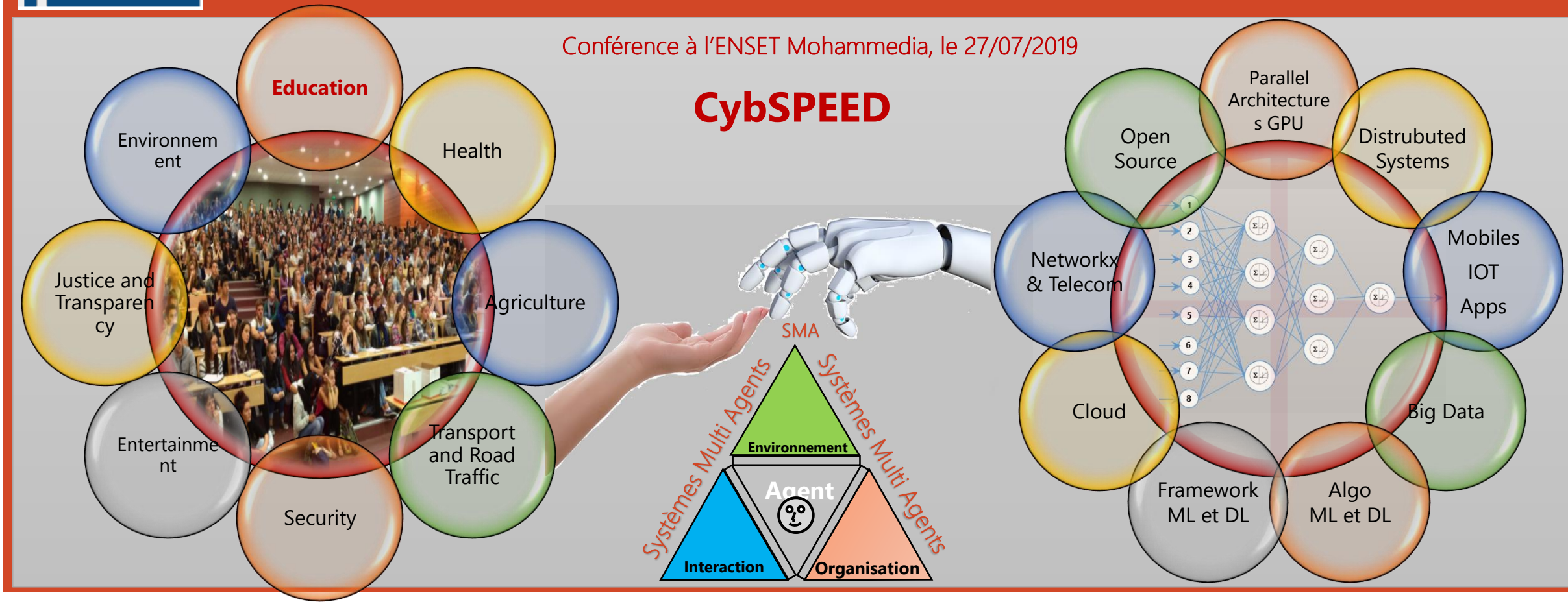
# Machines and Deep Learning for Web, Mobile and embedded Applications using DeepLearning4J & TensorFlowJS Frameworks

- https://www.youtube.com/user/mohamedYoussfi
- https://www.slideshare.net/mohamedYoussfi9
- https://www.researchgate.net/profile /Youssfi_Mohamed
- https://github.com/mohamedYoussfi/

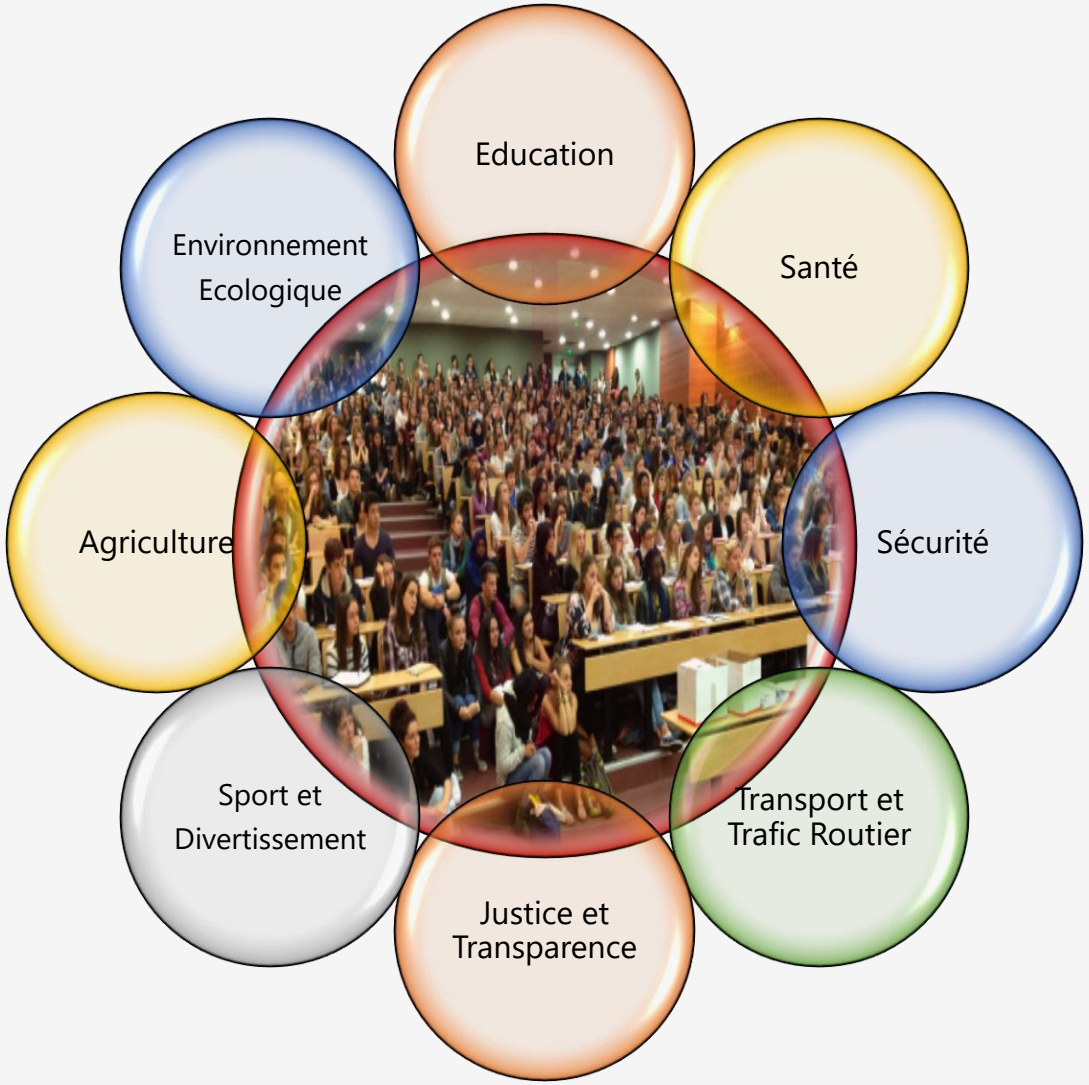
Par : Mohamed YOUSSFI
Lab. SSDIA, ENSET Mohammedia,
Université **Hassan II** de Casablanca Maroc
med@youssfi.net

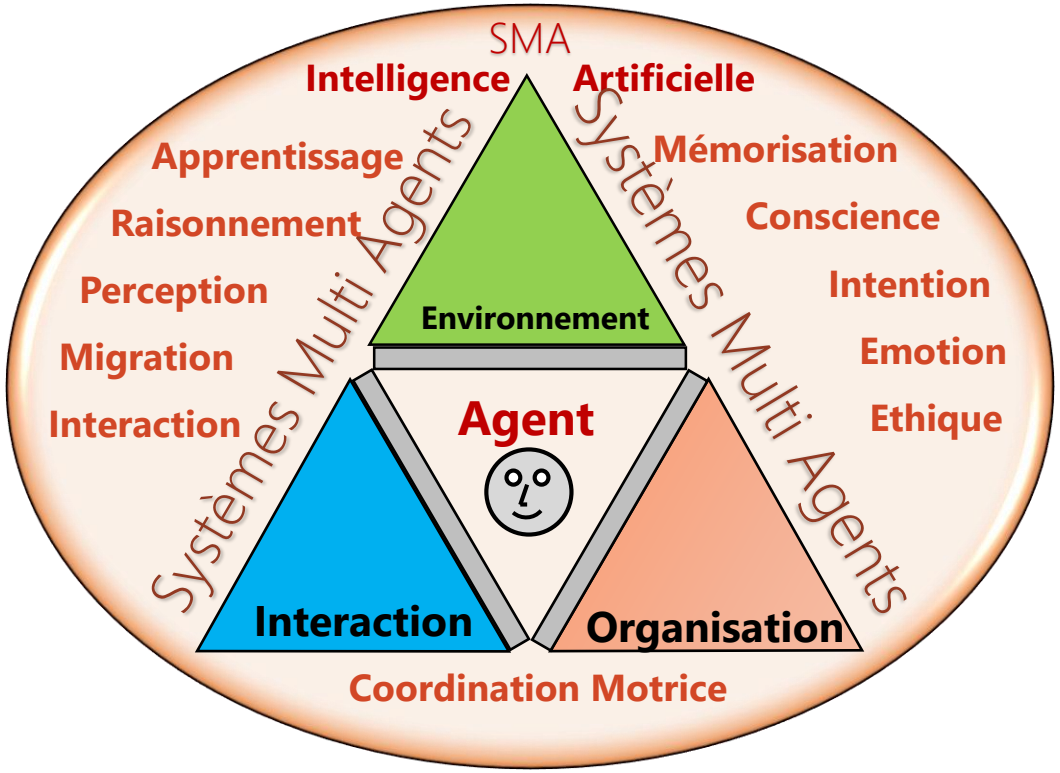
Conférence à l'ENSET Mohammedia, le 27/07/2019

## CybSPEED



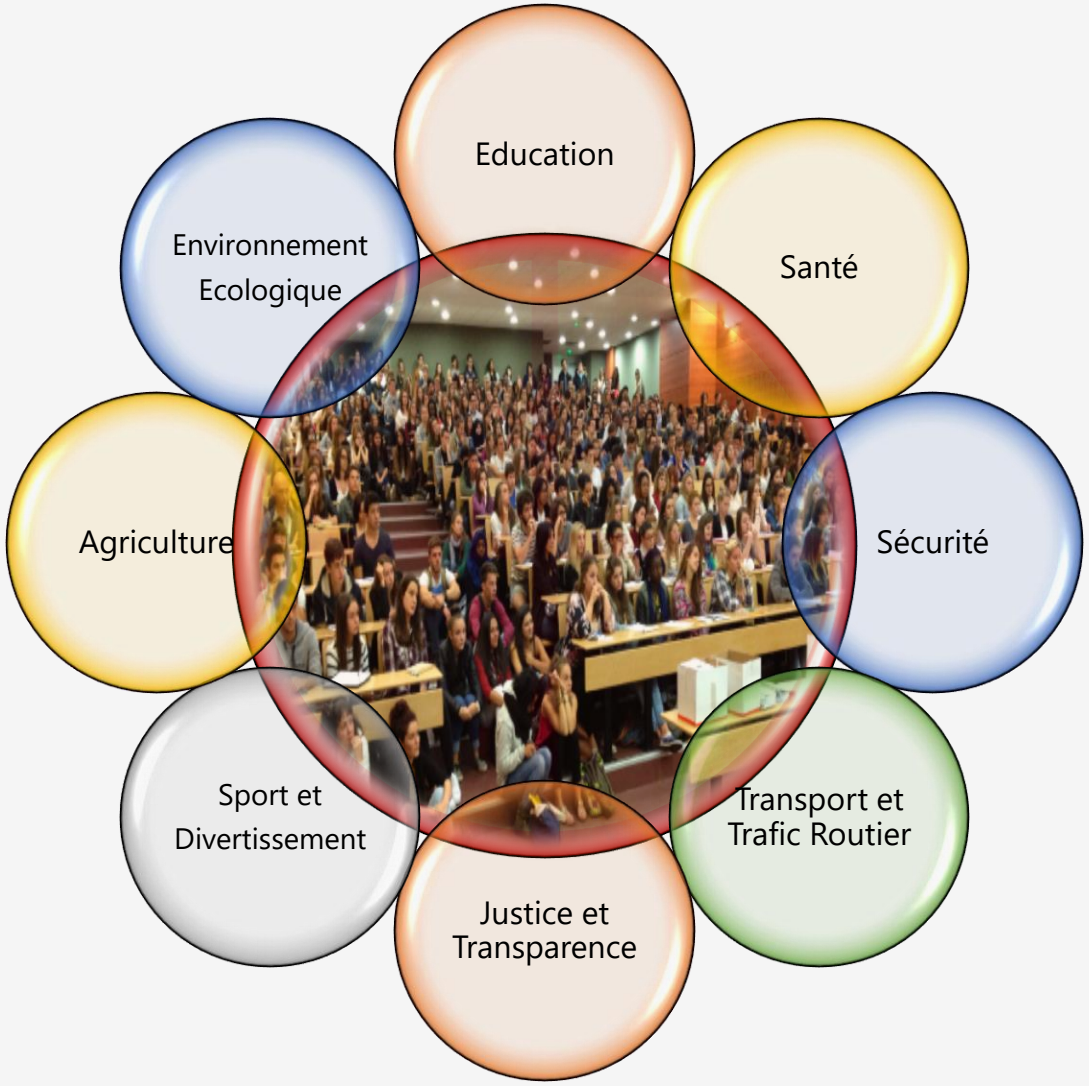
# Intelligence Artificielle?

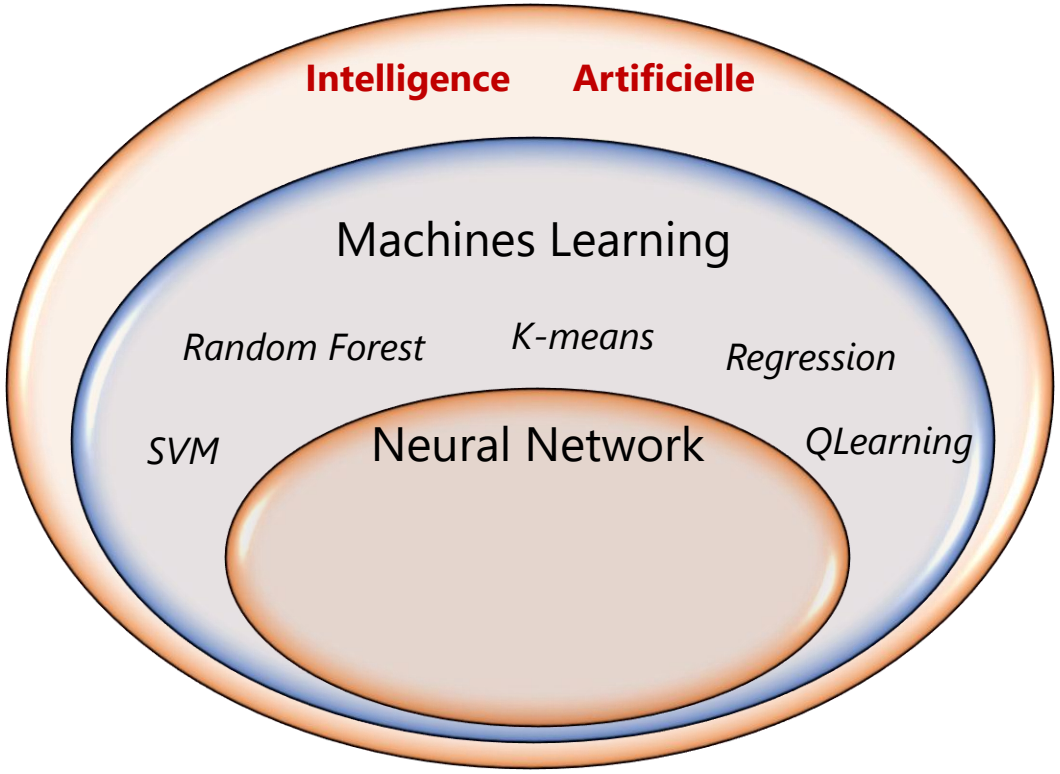


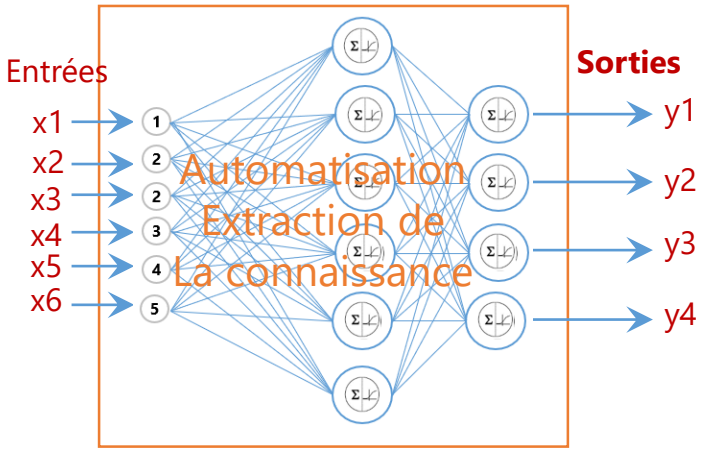

- L'**intelligence artificielle** : cherche à doter les systèmes informatiques de capacités intellectuelles comparables à celles des êtres humains »
- **Intelligence Artificielle Distribuée** =
  - **IA** : Pour des agents intelligent (Modéliser le savoir et le comportement)
  - \+ **Distribution** : Modéliser leurs interactions => **Intelligence Collective**
- **1985 : Systèmes Multi Agents**
- **IA Faible / IA Forte**
- **Transfert du raisonnement**

# Intelligence Artificielle, Machines Learning, Deep Learning

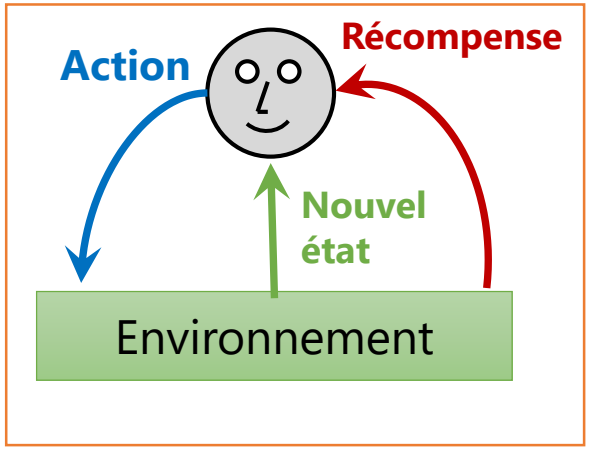


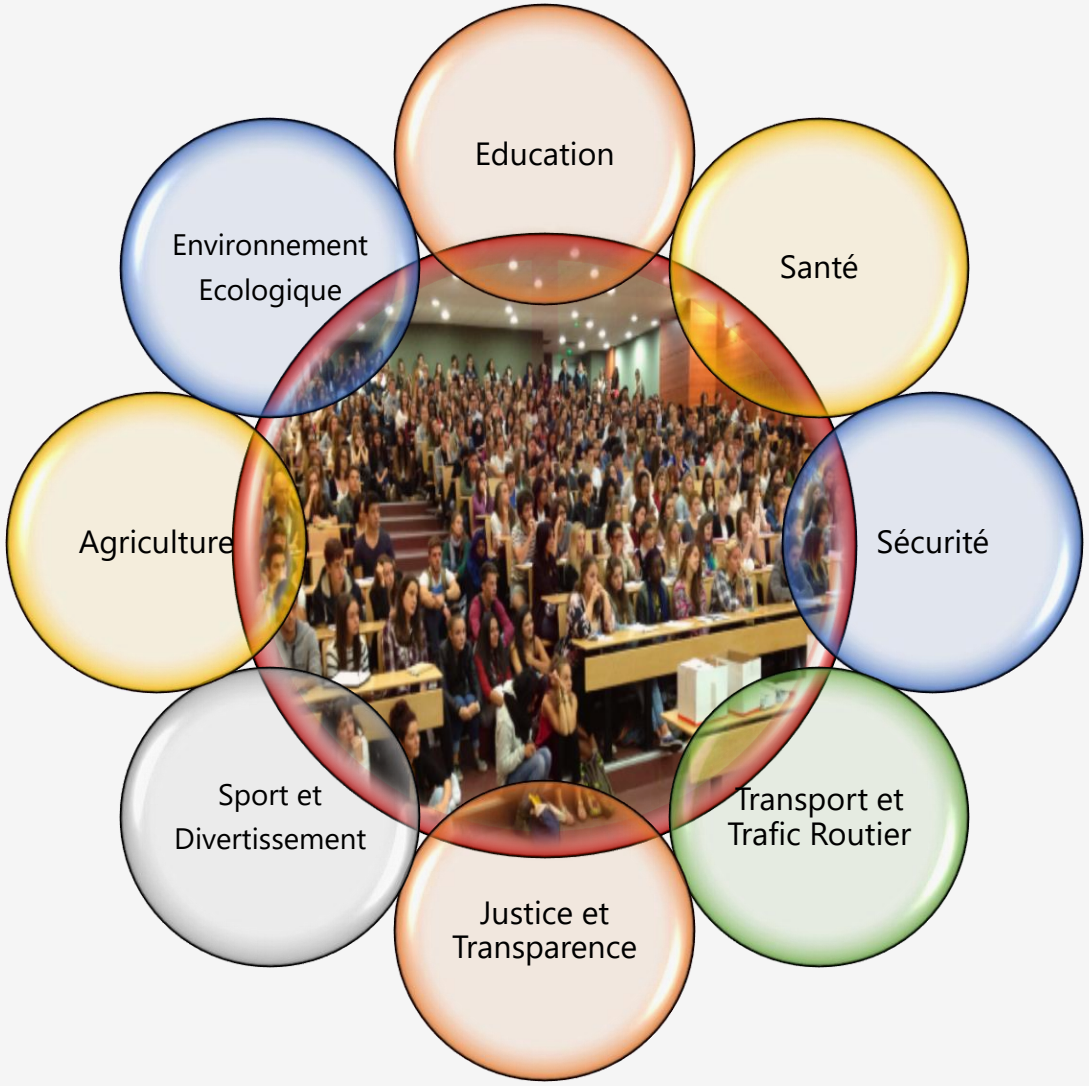

## Apprentissage Supervisé

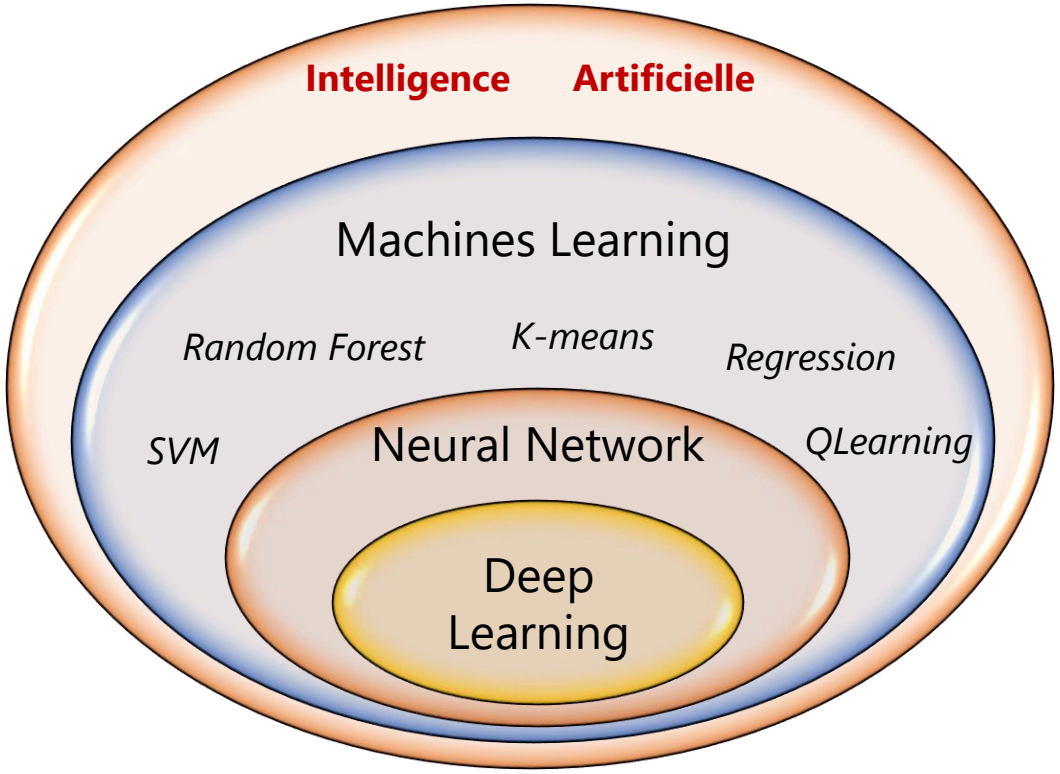



## Apprentissage Par Renforcement



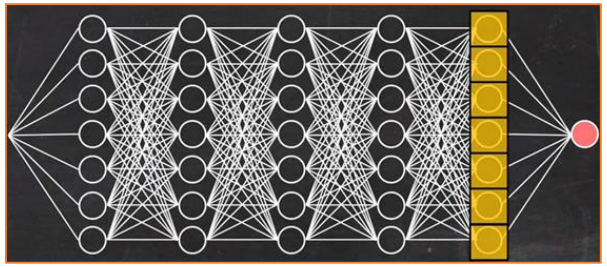
# Intelligence Artificielle, Machines Learning, Deep Learning




500 x 400 = 200 000 Pixels

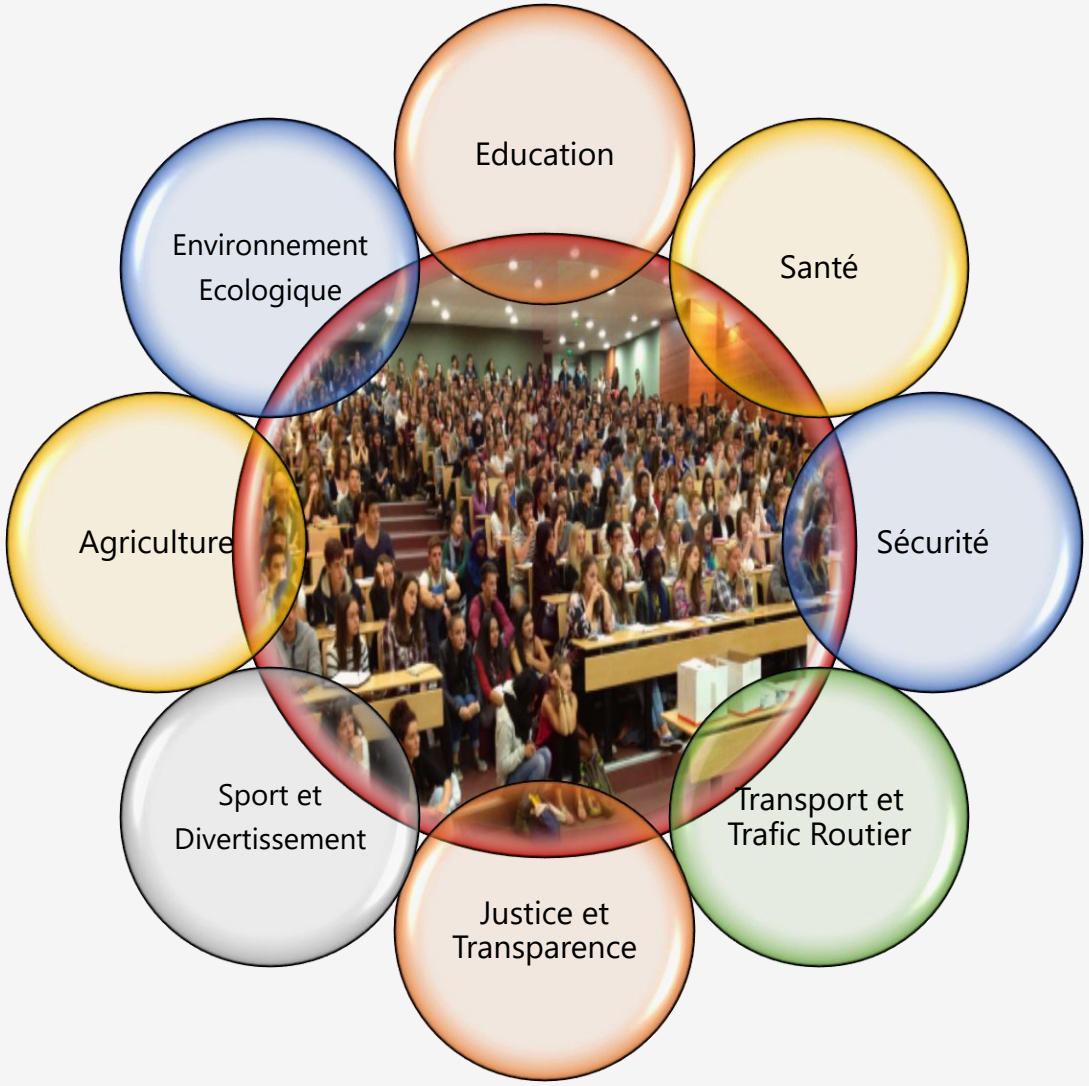
**Deep Learning**

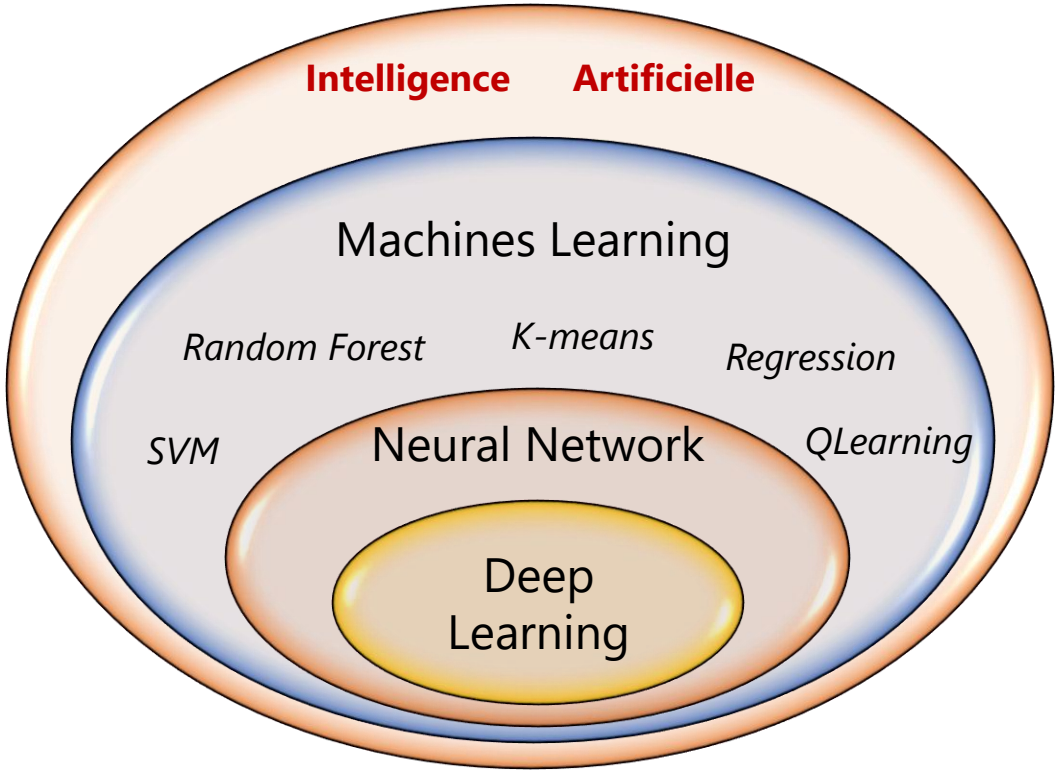


**CNN : [ Convolution, RELU, MAX PULLING ] [ Fully Connected ]**

**RNN : [ Recurent Neural Network]**

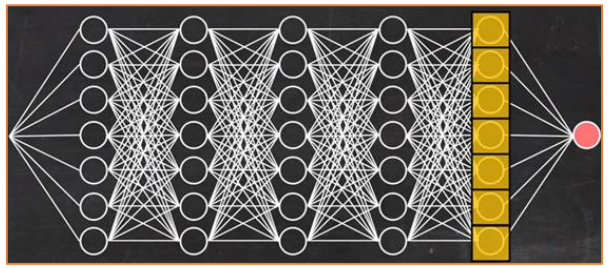
# Deep Reinforcement Learning



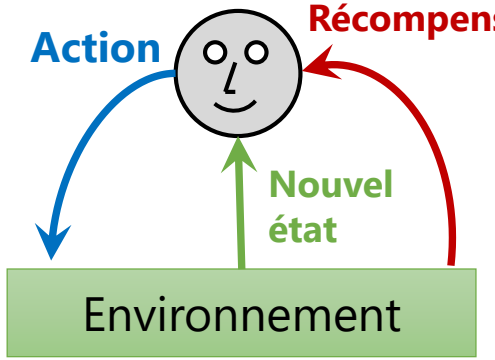

**Deep Reinforcement Learning**

**Deep Learning**



+

**Apprentissage Par Renforcement**

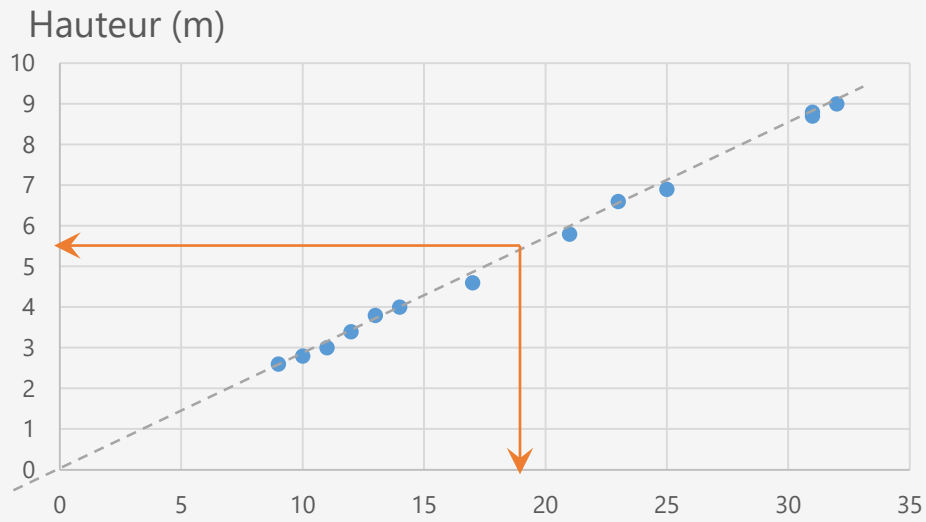


# Exemple très simple de machines learning expliqué aux artisans

Imaginez que vous êtes un botaniste et que vous vous intéresser à déterminer la relation qui existe entre le diamètre et la hauteur des troncs d'une catégorie d'arbres.

Mesures Expérimentales

| Diamètre (cm) | Hauteur (m) |
|---|---|
| 10 | 2,8 |
| 17 | 4,6 |
| 14 | 4 |
| 9 | 2,6 |
| 21 | 5,8 |
| 25 | 6,9 |
| 31 | 8,8 |
| 12 | 3,4 |
| 23 | 6,6 |
| 32 | 9 |
| 31 | 8,7 |
| 11 | 3 |
| 13 | 3,8 |



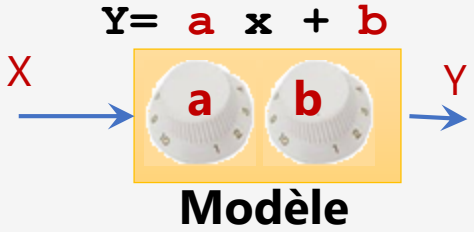

| | |
|---|---|
| 19 | ? |

Prédiction / Deviner

Tracer une droite comme modèle pour généraliser à partir des données de l'expérience : Apprentissage

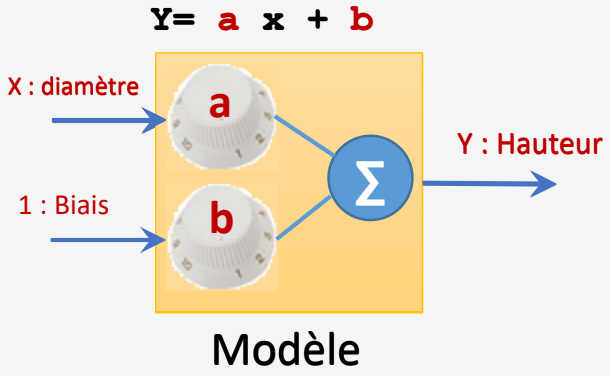
- Machines Learning c'est faire la même chose pour des problèmes complexes en utilisant des algorithmes

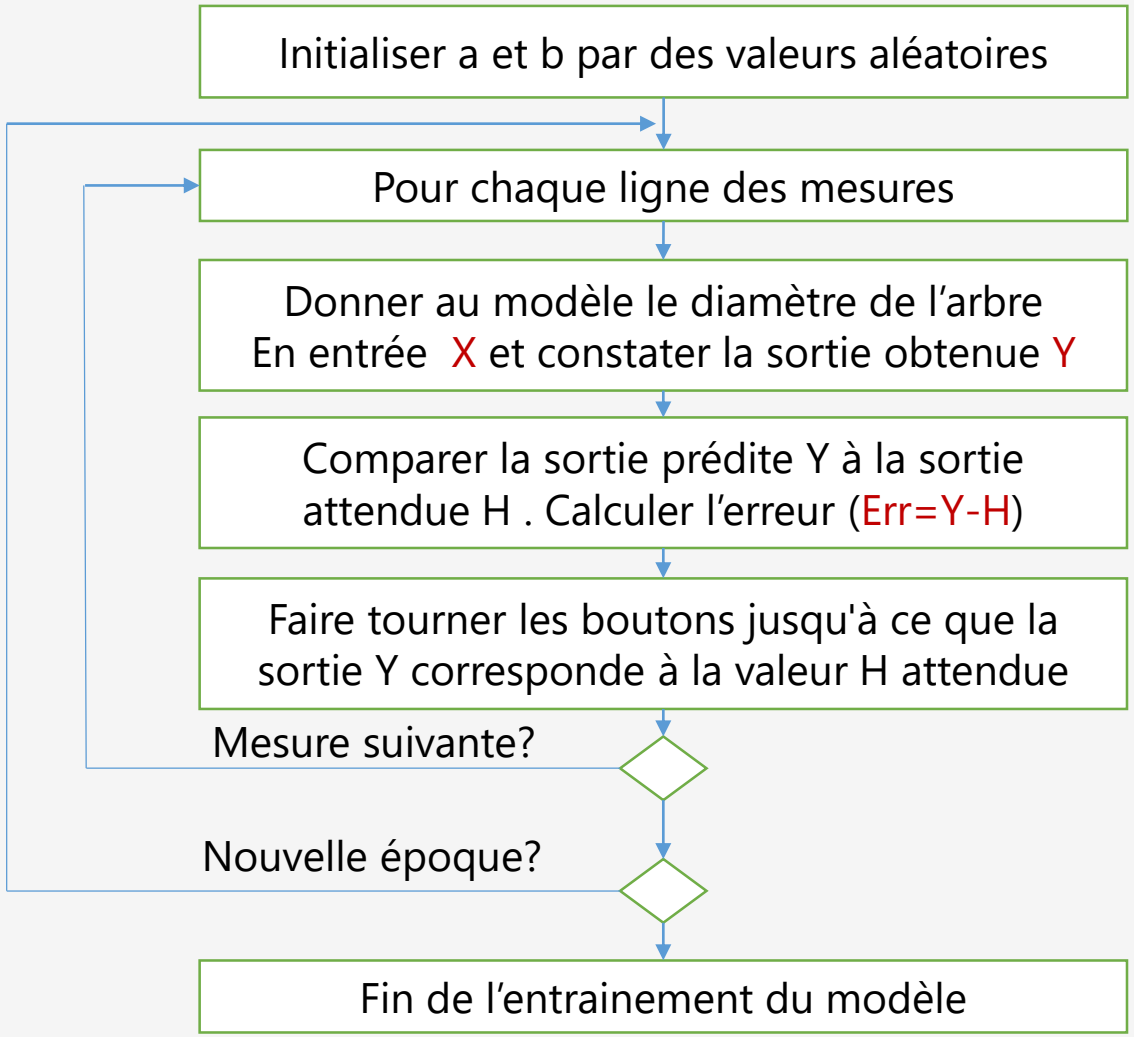
- **Phase d'apprentissage :** Chercher les valeurs de a et b qui permettent de définir une droite qui colle aux nuages de points
- **Phase de prédiction :** Pour une entrée x quelconque, le modèle calcule la sortie y correspondante



# Comment paramétrer ce modèle :

| Diamètre (cm) | Hauteur (m) |
|---|---|
| 10 | 2,8 |
| 17 | 4,6 |
| 14 | 4 |
| 9 | 2,6 |
| 21 | 5,8 |
| 25 | 6,9 |
| 31 | 8,8 |
| 12 | 3,4 |
| 23 | 6,6 |
| 32 | 9 |
| 31 | 8,7 |
| 11 | 3 |
| 13 | 3,8 |




Avec beaucoup d'expérience ou un peu de maths

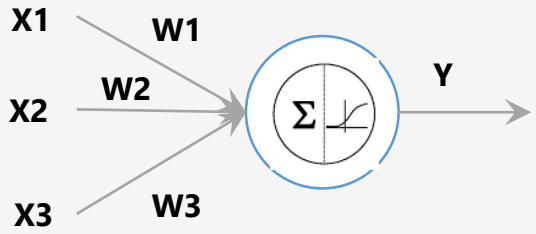
- Pour le réglage du bouton a : **a= a + X*erreur * μ**
  - μ entre 0 et 1 : Learning rate (Vitesse d'apprentissage)
- Pour le réglage du bouton b : **b= b + β*erreur * μ**
  - β s'appelle le BIAIS
- Au lieu de régler les boutons a et b pour chaque mesure, prendre le temps de noter les erreurs obtenues pour **N mesures** et par la suite trouver une formule qui permet de trouver le bon réglage à faire sur les boutons a et b qui permet de minimiser l'erreur pour toutes les mesures : N représente le **Batch size**

# Neurone Artificiel

Un neurone artificiel (NA) est une unité de calcul élémentaire (fonction mathématique )

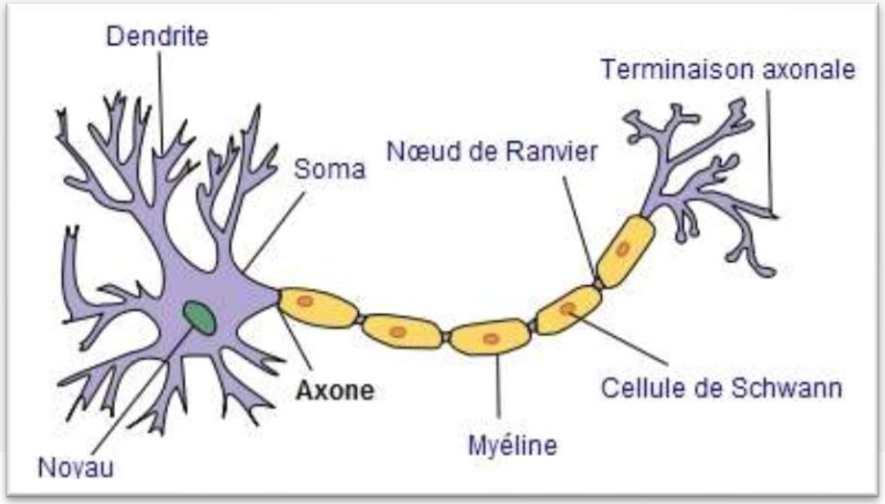
permet de mettre en relations des entrées Xi avec une sortie Y

- Un neurone artificiel est représentation approximative d'un neurone biologique
  - Additionne ses entrées $x_i$ pondérées par des poids $w_i$ ,
  - compare la somme résultante à une valeur seuil selon un fonction d'activation,
  - répond en émettant un signal si cette somme est supérieure ou égale à ce seuil (modèle ultra-simplifié du fonctionnement d'un neurone biologique).
  - Ces neurones sont par ailleurs associés en réseaux dont la topologie des connexions est variable : réseaux proactifs, récurrents, etc.
  - Enfin, l'efficacité de la transmission des signaux d'un neurone à l'autre peut varier : on parle de « poids synaptique », et ces poids peuvent être modulés par des règles d'apprentissage (ce qui mime la plasticité synaptique des réseaux biologiques).

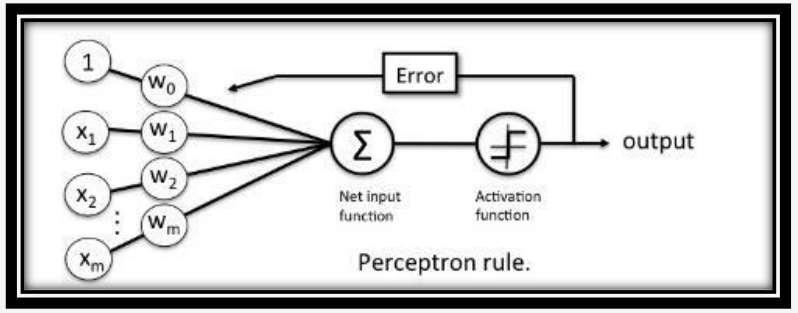



- $Sum=x_1w_1+x_2w+x_3w_3$
- Si Sum>Seuil => Y=1 Si non Y=0

- Un neurone reçoit des signaux venant de d'autres neurones à travers ses dendrites(connexions)
- Emit un signal ou non à travers son Axone selon les entrées reçues et les poids synaptiques.



# Perceptron [Frank Rosenblatt ,1957]

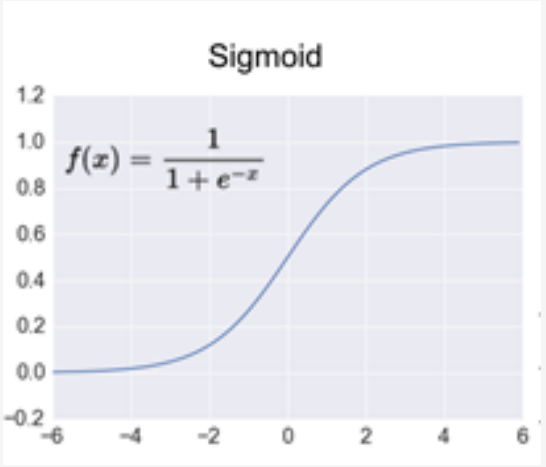


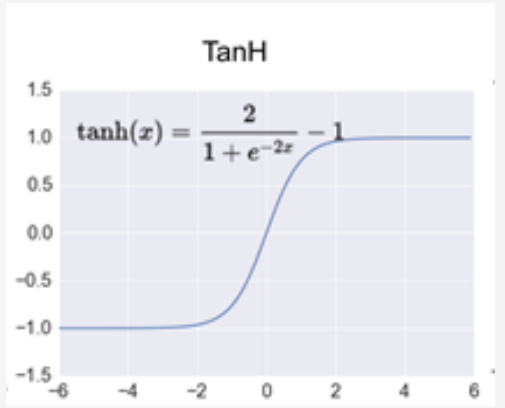

Modèle d'apprentissage supervisé, classifieurs binaires (séparant deux classes).

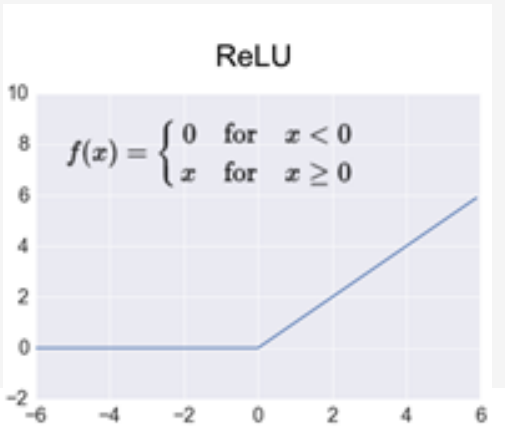
Le perceptron reçoie un ensemble d'entrées pondérées par des poids synaptiques $w_i$ et produit une sortie binaire en sortie.

La sortie est produite en :

- Additionnant les entrées xi pondérées par les poids synaptiques du perceptron wi
  - $S = \sum_{i=1}^{m} x_i w_i$
- Ensuite on applique à cette somme une **fonction d'activation non linéaire** qui produit une sortie binaire. : Exemple Seuillage
  - $Y = \begin{cases} 1 \; si \; S \geq \theta \\ -1 \; si \; S < \theta \end{cases}$



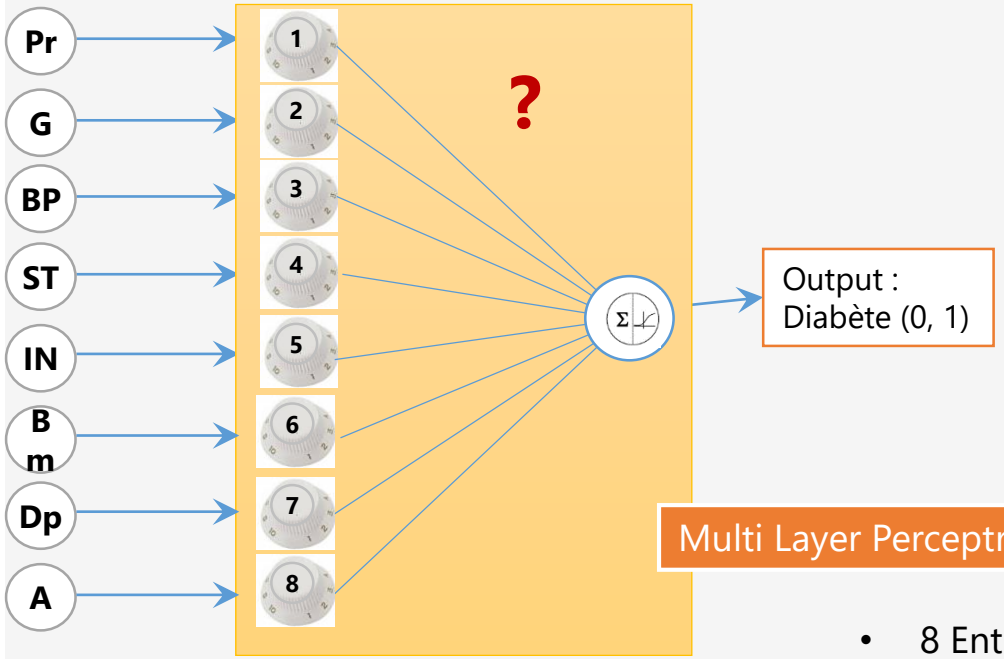




## Dans la phase d'apprentissage,

- **err= $Y_p$ - $Y_t$**
- mise à ajour des poids : $w_i(t) = w_i(t-1) + \mu \; err$
- **μ** : vitesse d'apprentissage (entre 0 et 1)

# Exemple un peu plus complexe : Prédiction du diabète



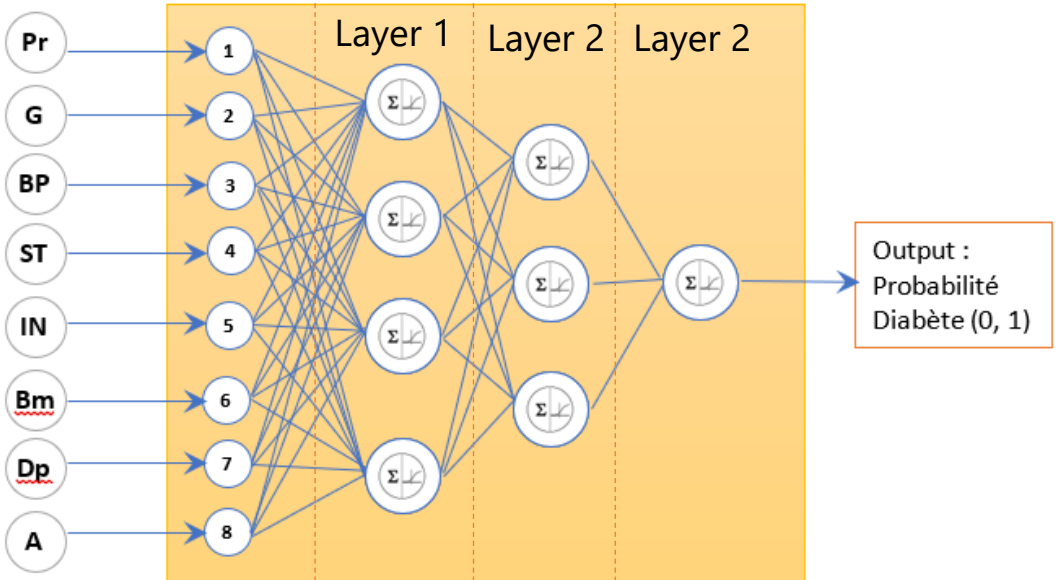

- **Inputs :**
1. **Pregnancies** : Nombre de grossesses
2. **Glucose** : La taux de Glucose dans le sang
3. **Blood Pressure** : La pression du sang
4. **Skin Thickness** : Epaisseur de la peau
5. **Insulin** : Le taux de l'insuline présent dans le sang
6. **BMI** : Indice de masse corporel
7. **Diabetes Pedigree Function** Antécédents de diabète sucrégénitique.
8. **Age** : Age du patient

- **Output : Présence ou non du diabète**

| Pr | Gluc | BP | ST | Ins | BMI | DPF | Age | Output |
|---|---|---|---|---|---|---|---|---|
| 6 | 148 | 72 | 35 | 0 | 33.6 | 0.627 | 50 | 1 |
| 1 | 85 | 66 | 29 | 0 | 26.6 | 0.351 | 31 | 0 |
| 8 | 183 | 64 | 0 | 0 | 23.3 | 0.672 | 32 | 1 |
| 1 | 89 | 66 | 23 | 94 | 28.1 | 0.167 | 21 | 0 |
| 0 | 137 | 40 | 35 | 168 | 43.1 | 2.288 | 33 | 1 |
| 5 | 116 | 74 | 0 | 0 | 25.6 | 0.201 | 30 | 0 |
| 3 | 78 | 50 | 32 | 88 | 31 | 0.248 | 26 | 1 |
| 10 | 115 | 0 | 0 | 0 | 35.3 | 0.134 | 29 | 0 |

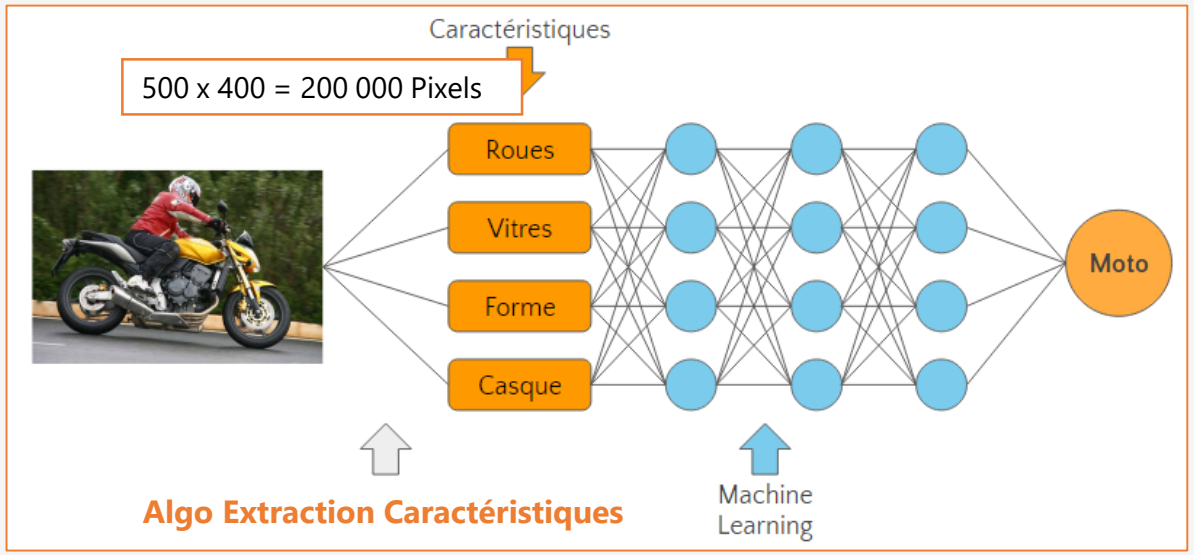
## Multi Layer Perceptron : Fully Connected

- 8 Entrées
- 1 Sortie
- 3 Couches de neurones
- Layer 1 : 4 neurones
- Layer 2 : 3 neurones
- Layer 3 : 1 neurone
- Nre Cxions : 8x4+4x3+3=47



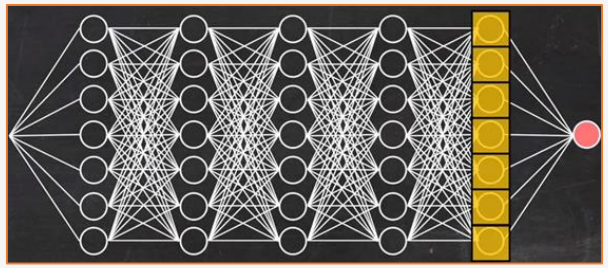

Pour ajuster les poids de connexions: Algo de Rétro Propagation du Gradient : (SGD, ADAM, etc.)

# Qu'en est ils, pour le cas de classification des images ?

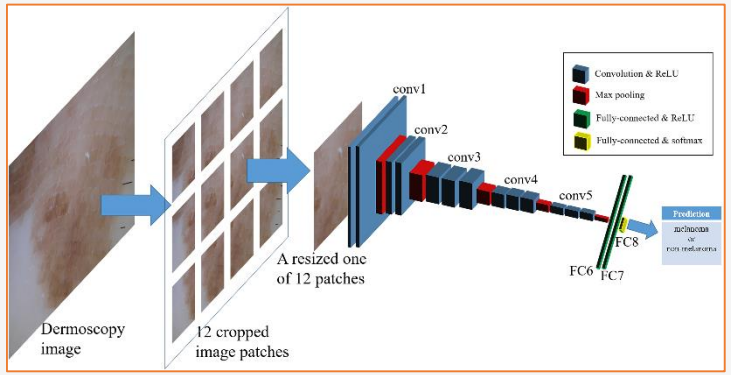



500 x 400 = 200 000 Pixels

MLP : Fully Connected



CNN : [ Convolution, RELU, MAX PULLING ] [ Fully Connected ]

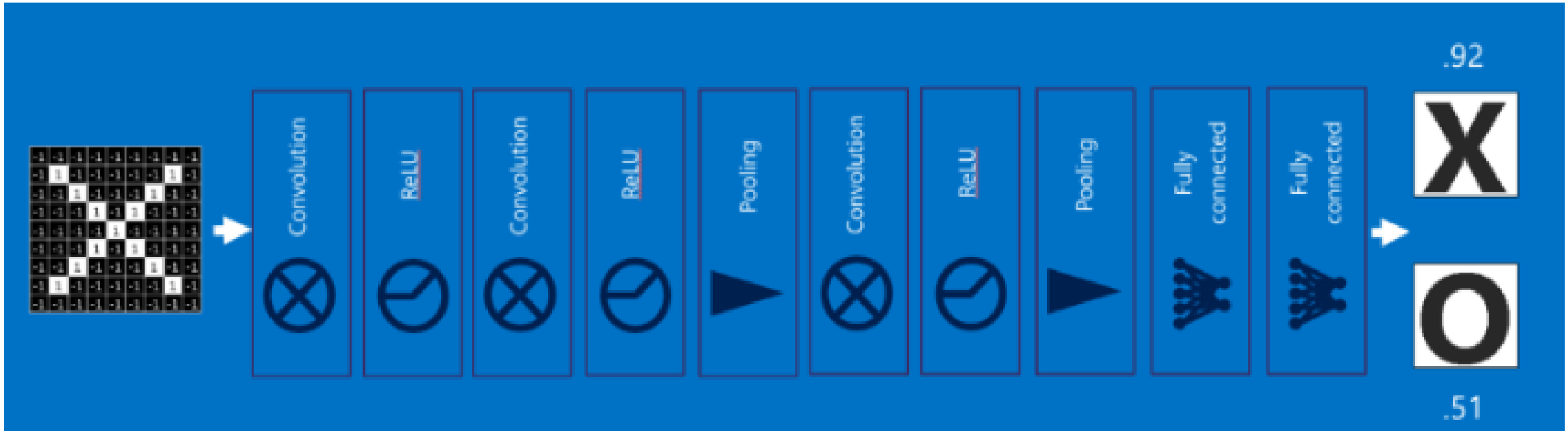



- Pour le cas des images deux approches:
  - Machines Learning :
    - Passer par un algo d'extraction de caractéristiques qui vont alimenter le réseaux (Réseaux de dimension faible)
  - DeepLearning :
    - Alimenter le réseaux avec tous les pixels des images brutes en ajoutant beaucoup de couches cachées au réseau
    - Fully Connected (MLP)
    - **Convolution Neural Network (CNN)**

# Deep Learning avec CNN (Convolutional Neural Network)

- En apprentissage profond, un réseau de neurones convolutifs ou réseau de neurones à convolution est un type de réseau de neurones artificiels, dans lequel le motif de connexion entre les neurones est inspiré par le cortex visuel des animaux.
- Utilisés dans la reconnaissance d'image et vidéo, les systèmes de recommandation et le traitement du langage naturel.
- Ils reçoivent des images en entrée, détectent les features (caractéristiques) de chacune d'entre elles, puis entraînent un classifieur.
- Un CNN se compose des différentes couches suivantes :
  - Couches de convolution qui consiste à appliquer un filtre de convolution à l'image pour détecter des caractéristiques de l'image
  - Couche de correction ReLU (Rectified Linear Unit) qui consiste à remplacer les nombres négatifs des images filtrées par des zéros.
  - Couches de Pooling qui consiste à réduire la taille de l'image en remplaçant en ne gardant les informations les plus importantes. Par exemple pour chaque groupe de 4 pixel le pixel ayant la valeur maximale (Max Pooling)
  - Couche entièrement connectée (Fully Connexted) qui reçoie un vecteur en entrée contenant les pixels aplatis de toutes les images filtrées, corrigées et réduites par le pooling.

# CNN


Convolution
ReLU
Convolution
ReLU
Pooling
Convolution
ReLU
Pooling
Fully connected
Fully connected
.92
.51

# Covolution Layer

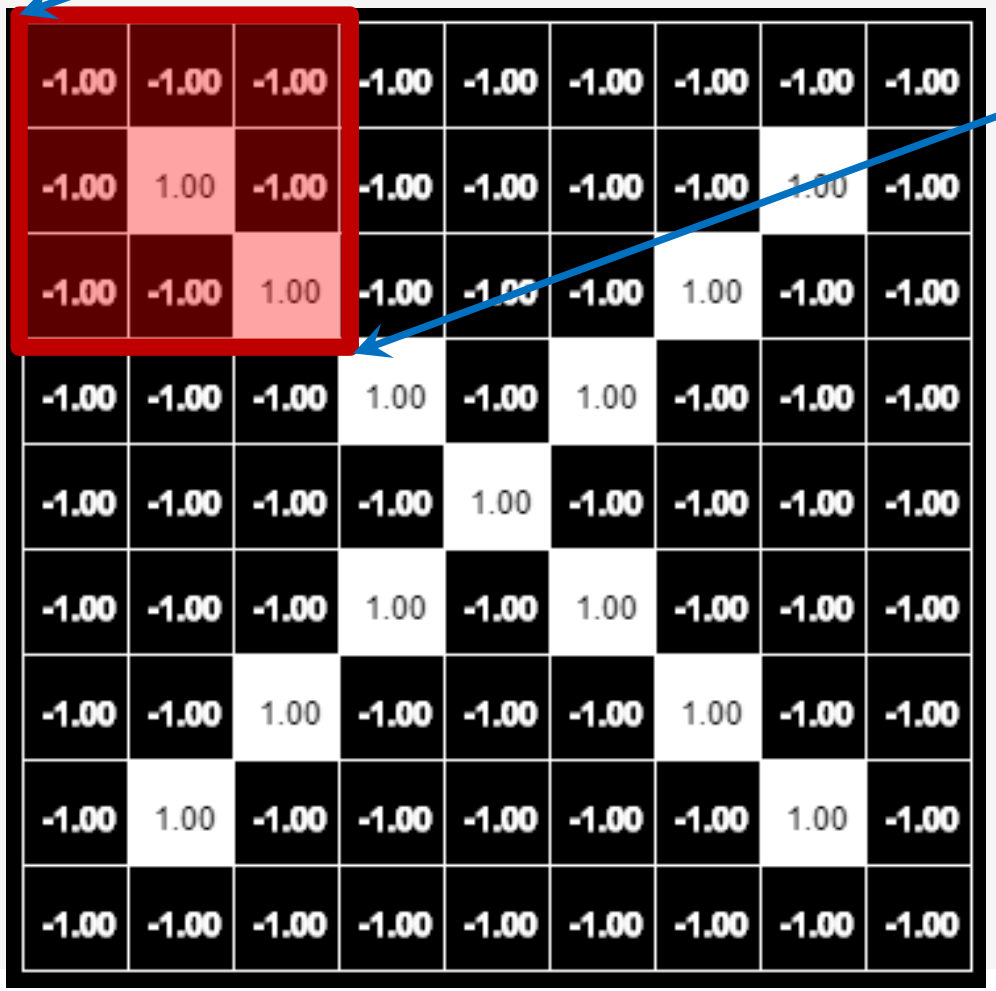
**Kernel**

| | | |
|---|---|---|
| 1.00 | -1.00 | -1.00 |
| -1.00 | 1.00 | -1.00 |
| -1.00 | -1.00 | 1.00 |

**Kernel size : 3 x 3**

**Stride : 1**

**Image Originale**



**Kernel size : 3 x 3**

**Stride : 1**

**Kernel**

| | | |
|---|---|---|
| 1.00 | -1.00 | -1.00 |
| -1.00 | 1.00 | -1.00 |
| -1.00 | -1.00 | 1.00 |

( (1)*(-1) + (-1)*(-1) + (-1)*(-1)+
(-1)*(-1) + (1)*(1) + (-1)*(-1)+
(-1)*(-1) + (-1)*(-1) + (1)*(1) ) /9 => **0.78**

**Image Originale**

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| -1.00 | -1.00 | -1.00 | -1.00 | -1.00 | -1.00 | -1.00 | -1.00 | -1.00 |
| -1.00 | 1.00 | -1.00 | -1.00 | -1.00 | -1.00 | -1.00 | 1.00 | -1.00 |
| -1.00 | -1.00 | 1.00 | -1.00 | -1.00 | -1.00 | 1.00 | -1.00 | -1.00 |
| -1.00 | -1.00 | -1.00 | 1.00 | -1.00 | 1.00 | -1.00 | -1.00 | -1.00 |
| -1.00 | -1.00 | -1.00 | -1.00 | 1.00 | -1.00 | -1.00 | -1.00 | -1.00 |
| -1.00 | -1.00 | -1.00 | 1.00 | -1.00 | 1.00 | -1.00 | -1.00 | -1.00 |
| -1.00 | -1.00 | 1.00 | -1.00 | -1.00 | -1.00 | 1.00 | -1.00 | -1.00 |
| -1.00 | 1.00 | -1.00 | -1.00 | -1.00 | -1.00 | -1.00 | 1.00 | -1.00 |
| -1.00 | -1.00 | -1.00 | -1.00 | -1.00 | -1.00 | -1.00 | -1.00 | -1.00 |

**Convolution**

**Image Filtrée**

| | | | | | | |
|---|---|---|---|---|---|---|
| 0.78 | -0.11 | 0.11 | 0.33 | 0.56 | -0.11 | 0.33 |
| -0.11 | 1.00 | -0.11 | 0.33 | -0.11 | 0.11 | -0.11 |
| 0.11 | -0.11 | 1.00 | -0.33 | 0.11 | -0.11 | 0.56 |
| 0.33 | 0.33 | -0.33 | 0.56 | -0.33 | 0.33 | 0.33 |
| 0.56 | -0.11 | 0.11 | -0.33 | 1.00 | -0.11 | 0.11 |
| -0.11 | 0.11 | -0.11 | 0.33 | -0.11 | 1.00 | -0.11 |
| 0.33 | -0.11 | 0.56 | 0.33 | 0.11 | -0.11 | 0.78 |

**Kernel size : 3 x 3**

**Stride : 1**

**Kernel**

| | | |
|---|---|---|
| 1.00 | -1.00 | -1.00 |
| -1.00 | 1.00 | -1.00 |
| -1.00 | -1.00 | 1.00 |

( (1)*(-1) + (-1)*(-1) + (-1)*(-1)+
(-1)*(1) + (1)*(-1) + (-1)*(-1)+
(-1)*(-1) + (-1)*(1) + (1)*(-1) ) /9 => **-0.11**

**Image Originale**

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| -1.00 | -1.00 | -1.00 | -1.00 | -1.00 | -1.00 | -1.00 | -1.00 |
| -1.00 | 1.00 | -1.00 | -1.00 | -1.00 | -1.00 | 1.00 | -1.00 |
| -1.00 | -1.00 | 1.00 | -1.00 | -1.00 | 1.00 | -1.00 | -1.00 |
| -1.00 | -1.00 | -1.00 | 1.00 | -1.00 | 1.00 | -1.00 | -1.00 |
| -1.00 | -1.00 | -1.00 | -1.00 | 1.00 | -1.00 | -1.00 | -1.00 |
| -1.00 | -1.00 | -1.00 | 1.00 | -1.00 | 1.00 | -1.00 | -1.00 |
| -1.00 | -1.00 | 1.00 | -1.00 | -1.00 | -1.00 | 1.00 | -1.00 |
| -1.00 | 1.00 | -1.00 | -1.00 | -1.00 | -1.00 | 1.00 | -1.00 |
| -1.00 | -1.00 | -1.00 | -1.00 | -1.00 | -1.00 | -1.00 | -1.00 |

**Convolution**

**Image Filtrée**

| | | | | | | |
|---|---|---|---|---|---|---|
| 0.78 | -0.11 | 0.11 | 0.33 | 0.56 | -0.11 | 0.33 |
| -0.11 | 1.00 | -0.11 | 0.33 | -0.11 | 0.11 | -0.11 |
| 0.11 | -0.11 | 1.00 | -0.33 | 0.11 | -0.11 | 0.56 |
| 0.33 | 0.33 | -0.33 | 0.56 | -0.33 | 0.33 | 0.33 |
| 0.56 | -0.11 | 0.11 | -0.33 | 1.00 | -0.11 | 0.11 |
| -0.11 | 0.11 | -0.11 | 0.33 | -0.11 | 1.00 | -0.11 |
| 0.33 | -0.11 | 0.56 | 0.33 | 0.11 | -0.11 | 0.78 |

**Kernel size : 3 x 3**

**Stride : 1**

**Kernel**

| | | |
|---|---|---|
| 1.00 | -1.00 | -1.00 |
| -1.00 | 1.00 | -1.00 |
| -1.00 | -1.00 | 1.00 |

( (1)*(-1) + (-1)*(-1) + (-1)*(-1)+
(-1)*(-1) + (1)*(-1) + (-1)*(-1)+
(-1)*(1) + (-1)*(-1) + (1)*(-1) ) /9 => **0.11**

**Image Originale**

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| -1.00 | -1.00 | -1.00 | -1.00 | -1.00 | -1.00 | -1.00 | -1.00 | -1.00 |
| -1.00 | 1.00 | -1.00 | -1.00 | -1.00 | -1.00 | -1.00 | 1.00 | -1.00 |
| -1.00 | -1.00 | 1.00 | -1.00 | -1.00 | -1.00 | 1.00 | -1.00 | -1.00 |
| -1.00 | -1.00 | -1.00 | 1.00 | -1.00 | 1.00 | -1.00 | -1.00 | -1.00 |
| -1.00 | -1.00 | -1.00 | -1.00 | 1.00 | -1.00 | -1.00 | -1.00 | -1.00 |
| -1.00 | -1.00 | -1.00 | 1.00 | -1.00 | 1.00 | -1.00 | -1.00 | -1.00 |
| -1.00 | -1.00 | 1.00 | -1.00 | -1.00 | -1.00 | 1.00 | -1.00 | -1.00 |
| -1.00 | 1.00 | -1.00 | -1.00 | -1.00 | -1.00 | -1.00 | 1.00 | -1.00 |
| -1.00 | -1.00 | -1.00 | -1.00 | -1.00 | -1.00 | -1.00 | -1.00 | -1.00 |

**Convolution**

**Image Filtrée**

| | | | | | | |
|---|---|---|---|---|---|---|
| 0.78 | -0.11 | 0.11 | 0.33 | 0.56 | -0.11 | 0.33 |
| -0.11 | 1.00 | -0.11 | 0.33 | -0.11 | 0.11 | -0.11 |
| 0.11 | -0.11 | 1.00 | -0.33 | 0.11 | -0.11 | 0.56 |
| 0.33 | 0.33 | -0.33 | 0.56 | -0.33 | 0.33 | 0.33 |
| 0.56 | -0.11 | 0.11 | -0.33 | 1.00 | -0.11 | 0.11 |
| -0.11 | 0.11 | -0.11 | 0.33 | -0.11 | 1.00 | -0.11 |
| 0.33 | -0.11 | 0.56 | 0.33 | 0.11 | -0.11 | 0.78 |

**Kernel size : 3 x 3**

**Stride : 1**

**Kernel**

| | | |
|---|---|---|
| 1.00 | -1.00 | -1.00 |
| -1.00 | 1.00 | -1.00 |
| -1.00 | -1.00 | 1.00 |

( (1)*(-1) + (-1)*(-1) + (-1)*(-1)+
(-1)*(-1) + (1)*(-1) + (-1)*(-1)+
(-1)*(-1) + (-1)*(-1) + (1)*(-1) ) /9 => **0.33**

**Image Originale**

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| -1.00 | -1.00 | -1.00 | -1.00 | -1.00 | -1.00 | -1.00 | -1.00 | -1.00 |
| -1.00 | 1.00 | -1.00 | -1.00 | -1.00 | -1.00 | -1.00 | 1.00 | -1.00 |
| -1.00 | -1.00 | 1.00 | -1.00 | -1.00 | -1.00 | 1.00 | -1.00 | -1.00 |
| -1.00 | -1.00 | -1.00 | 1.00 | -1.00 | 1.00 | -1.00 | -1.00 | -1.00 |
| -1.00 | -1.00 | -1.00 | -1.00 | 1.00 | -1.00 | -1.00 | -1.00 | -1.00 |
| -1.00 | -1.00 | -1.00 | 1.00 | -1.00 | 1.00 | -1.00 | -1.00 | -1.00 |
| -1.00 | -1.00 | 1.00 | -1.00 | -1.00 | -1.00 | 1.00 | -1.00 | -1.00 |
| -1.00 | 1.00 | -1.00 | -1.00 | -1.00 | -1.00 | -1.00 | 1.00 | -1.00 |
| -1.00 | -1.00 | -1.00 | -1.00 | -1.00 | -1.00 | -1.00 | -1.00 | -1.00 |

**Convolution**

**Image Filtrée**

| | | | | | | |
|---|---|---|---|---|---|---|
| 0.78 | -0.11 | 0.11 | 0.33 | 0.56 | -0.11 | 0.33 |
| -0.11 | 1.00 | -0.11 | 0.33 | -0.11 | 0.11 | -0.11 |
| 0.11 | -0.11 | 1.00 | -0.33 | 0.11 | -0.11 | 0.56 |
| 0.33 | 0.33 | -0.33 | 0.56 | -0.33 | 0.33 | 0.33 |
| 0.56 | -0.11 | 0.11 | -0.33 | 1.00 | -0.11 | 0.11 |
| -0.11 | 0.11 | -0.11 | 0.33 | -0.11 | 1.00 | -0.11 |
| 0.33 | -0.11 | 0.56 | 0.33 | 0.11 | -0.11 | 0.78 |

**Kernel size : 3 x 3**

**Stride : 1**

**Kernel**

| | | |
|---|---|---|
| 1.00 | -1.00 | -1.00 |
| -1.00 | 1.00 | -1.00 |
| -1.00 | -1.00 | 1.00 |

( (1)*(-1) + (-1)*(-1) + (-1)*(-1)+
(-1)*(-1) + (1)*(-1) + (-1)*(-1)+
(-1)*(-1) + (-1)*(-1) + (1)*(1) ) /9 => **0.58**

**Image Originale**

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| -1.00 | -1.00 | -1.00 | -1.00 | -1.00 | -1.00 | -1.00 | -1.00 | -1.00 |
| -1.00 | 1.00 | -1.00 | -1.00 | -1.00 | -1.00 | -1.00 | 1.00 | -1.00 |
| -1.00 | -1.00 | 1.00 | -1.00 | -1.00 | -1.00 | 1.00 | -1.00 | -1.00 |
| -1.00 | -1.00 | -1.00 | 1.00 | -1.00 | 1.00 | -1.00 | -1.00 | -1.00 |
| -1.00 | -1.00 | -1.00 | -1.00 | 1.00 | -1.00 | -1.00 | -1.00 | -1.00 |
| -1.00 | -1.00 | -1.00 | 1.00 | -1.00 | 1.00 | -1.00 | -1.00 | -1.00 |
| -1.00 | -1.00 | 1.00 | -1.00 | -1.00 | -1.00 | 1.00 | -1.00 | -1.00 |
| -1.00 | 1.00 | -1.00 | -1.00 | -1.00 | -1.00 | -1.00 | 1.00 | -1.00 |
| -1.00 | -1.00 | -1.00 | -1.00 | -1.00 | -1.00 | -1.00 | -1.00 | -1.00 |

Convolution

**Image Filtrée**

| | | | | | | |
|---|---|---|---|---|---|---|
| 0.78 | -0.11 | 0.11 | 0.33 | 0.56 | -0.11 | 0.33 |
| -0.11 | 1.00 | -0.11 | 0.33 | -0.11 | 0.11 | -0.11 |
| 0.11 | -0.11 | 1.00 | -0.33 | 0.11 | -0.11 | 0.56 |
| 0.33 | 0.33 | -0.33 | 0.56 | -0.33 | 0.33 | 0.33 |
| 0.56 | -0.11 | 0.11 | -0.33 | 1.00 | -0.11 | 0.11 |
| -0.11 | 0.11 | -0.11 | 0.33 | -0.11 | 1.00 | -0.11 |
| 0.33 | -0.11 | 0.56 | 0.33 | 0.11 | -0.11 | 0.78 |

**Kernel size : 3 x 3**

**Stride : 1**

**Kernel**

| | | |
|---|---|---|
| 1.00 | -1.00 | -1.00 |
| -1.00 | 1.00 | -1.00 |
| -1.00 | -1.00 | 1.00 |

( (1)*(-1) + (-1)*(-1) + (-1)*(-1)+
(-1)*(-1) + (1)*(1) + (-1)*(1)+
(-1)*(-1) + (-1)*(1) + (1)*(-1) ) /9 => -0.11

**Image Originale**

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| -1.00 | -1.00 | -1.00 | -1.00 | -1.00 | -1.00 | -1.00 | -1.00 | -1.00 |
| -1.00 | 1.00 | -1.00 | -1.00 | -1.00 | -1.00 | -1.00 | 1.00 | -1.00 |
| -1.00 | -1.00 | 1.00 | -1.00 | -1.00 | -1.00 | 1.00 | -1.00 | -1.00 |
| -1.00 | -1.00 | -1.00 | 1.00 | -1.00 | 1.00 | -1.00 | -1.00 | -1.00 |
| -1.00 | -1.00 | -1.00 | -1.00 | 1.00 | -1.00 | -1.00 | -1.00 | -1.00 |
| -1.00 | -1.00 | -1.00 | 1.00 | -1.00 | 1.00 | -1.00 | -1.00 | -1.00 |
| -1.00 | -1.00 | 1.00 | -1.00 | -1.00 | -1.00 | 1.00 | -1.00 | -1.00 |
| -1.00 | 1.00 | -1.00 | -1.00 | -1.00 | -1.00 | -1.00 | 1.00 | -1.00 |
| -1.00 | -1.00 | -1.00 | -1.00 | -1.00 | -1.00 | -1.00 | -1.00 | -1.00 |

Convolution

**Image Filtrée**

| | | | | | | |
|---|---|---|---|---|---|---|
| 0.78 | -0.11 | 0.11 | 0.33 | 0.56 | -0.11 | 0.33 |
| -0.11 | 1.00 | -0.11 | 0.33 | -0.11 | 0.11 | -0.11 |
| 0.11 | -0.11 | 1.00 | -0.33 | 0.11 | -0.11 | 0.56 |
| 0.33 | 0.33 | -0.33 | 0.56 | -0.33 | 0.33 | 0.33 |
| 0.56 | -0.11 | 0.11 | -0.33 | 1.00 | -0.11 | 0.11 |
| -0.11 | 0.11 | -0.11 | 0.33 | -0.11 | 1.00 | -0.11 |
| 0.33 | -0.11 | 0.56 | 0.33 | 0.11 | -0.11 | 0.78 |

**Kernel size : 3 x 3**

**Stride : 1**

**Kernel**

| | | |
|---|---|---|
| 1.00 | -1.00 | -1.00 |
| -1.00 | 1.00 | -1.00 |
| -1.00 | -1.00 | 1.00 |

( (1)*(-1) + (-1)*(-1) + (-1)*(-1)+
(-1)*(-1) + (1)*(1) + (-1)*(-1)+
(-1)*(1) + (-1)*(-1) + (1)*(-1) ) /9 => **0.33**

**Image Originale**

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| -1.00 | -1.00 | -1.00 | -1.00 | -1.00 | -1.00 | -1.00 | -1.00 | -1.00 |
| -1.00 | 1.00 | -1.00 | -1.00 | -1.00 | -1.00 | -1.00 | 1.00 | -1.00 |
| -1.00 | -1.00 | 1.00 | -1.00 | -1.00 | -1.00 | 1.00 | -1.00 | -1.00 |
| -1.00 | -1.00 | -1.00 | 1.00 | -1.00 | 1.00 | -1.00 | -1.00 | -1.00 |
| -1.00 | -1.00 | -1.00 | -1.00 | 1.00 | -1.00 | -1.00 | -1.00 | -1.00 |
| -1.00 | -1.00 | -1.00 | 1.00 | -1.00 | 1.00 | -1.00 | -1.00 | -1.00 |
| -1.00 | -1.00 | 1.00 | -1.00 | -1.00 | -1.00 | 1.00 | -1.00 | -1.00 |
| -1.00 | 1.00 | -1.00 | -1.00 | -1.00 | -1.00 | -1.00 | 1.00 | -1.00 |
| -1.00 | -1.00 | -1.00 | -1.00 | -1.00 | -1.00 | -1.00 | -1.00 | -1.00 |

Convolution

**Image Filtrée**

| | | | | | | |
|---|---|---|---|---|---|---|
| 0.78 | -0.11 | 0.11 | 0.33 | 0.56 | -0.11 | 0.33 |
| -0.11 | 1.00 | -0.11 | 0.33 | -0.11 | 0.11 | -0.11 |
| 0.11 | -0.11 | 1.00 | -0.33 | 0.11 | -0.11 | 0.56 |
| 0.33 | 0.33 | -0.33 | 0.56 | -0.33 | 0.33 | 0.33 |
| 0.56 | -0.11 | 0.11 | -0.33 | 1.00 | -0.11 | 0.11 |
| -0.11 | 0.11 | -0.11 | 0.33 | -0.11 | 1.00 | -0.11 |
| 0.33 | -0.11 | 0.56 | 0.33 | 0.11 | -0.11 | 0.78 |

**Kernel size : 3 x 3**

**Stride : 1**

**Kernel**

| | | |
|---|---|---|
| 1.00 | -1.00 | -1.00 |
| -1.00 | 1.00 | -1.00 |
| -1.00 | -1.00 | 1.00 |

( (1)*(-1) + (-1)*(1) + (-1)*(-1)+
(-1)*(-1) + (1)*(-1) + (-1)*(1)+
(-1)*(-1) + (-1)*(-1) + (1)*(-1) ) /9 => **-0.11**

**Image Originale**

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| -1.00 | -1.00 | -1.00 | -1.00 | -1.00 | -1.00 | -1.00 | -1.00 | -1.00 |
| -1.00 | 1.00 | -1.00 | -1.00 | -1.00 | -1.00 | -1.00 | 1.00 | -1.00 |
| -1.00 | -1.00 | 1.00 | -1.00 | -1.00 | -1.00 | 1.00 | -1.00 | -1.00 |
| -1.00 | -1.00 | -1.00 | 1.00 | -1.00 | 1.00 | -1.00 | -1.00 | -1.00 |
| -1.00 | -1.00 | -1.00 | -1.00 | 1.00 | -1.00 | -1.00 | -1.00 | -1.00 |
| -1.00 | -1.00 | -1.00 | 1.00 | -1.00 | 1.00 | -1.00 | -1.00 | -1.00 |
| -1.00 | -1.00 | 1.00 | -1.00 | -1.00 | -1.00 | 1.00 | -1.00 | -1.00 |
| -1.00 | 1.00 | -1.00 | -1.00 | -1.00 | -1.00 | -1.00 | 1.00 | -1.00 |
| -1.00 | -1.00 | -1.00 | -1.00 | -1.00 | -1.00 | -1.00 | -1.00 | -1.00 |

Convolution

**Image Filtrée**

| | | | | | | |
|---|---|---|---|---|---|---|
| 0.78 | -0.11 | 0.11 | 0.33 | 0.56 | -0.11 | 0.33 |
| -0.11 | 1.00 | -0.11 | 0.33 | -0.11 | 0.11 | -0.11 |
| 0.11 | -0.11 | 1.00 | -0.33 | 0.11 | -0.11 | 0.56 |
| 0.33 | 0.33 | -0.33 | 0.56 | -0.33 | 0.33 | 0.33 |
| 0.56 | -0.11 | 0.11 | -0.33 | 1.00 | -0.11 | 0.11 |
| -0.11 | 0.11 | -0.11 | 0.33 | -0.11 | 1.00 | -0.11 |
| 0.33 | -0.11 | 0.56 | 0.33 | 0.11 | -0.11 | 0.78 |

**Kernel size : 3 x 3**

**Stride : 1**

**Kernel**

| | | |
|---|---|---|
| 1.00 | -1.00 | -1.00 |
| -1.00 | 1.00 | -1.00 |
| -1.00 | -1.00 | 1.00 |

( (1)*(1) + (-1)*(-1) + (-1)*(-1)+
(-1)*(-1) + (1)*(1) + (-1)*(-1)+
(-1)*(-1) + (-1)*(-1) + (1)*(1) ) /9 => 1

**Image Originale**

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| -1.00 | -1.00 | -1.00 | -1.00 | -1.00 | -1.00 | -1.00 | -1.00 | -1.00 |
| -1.00 | 1.00 | -1.00 | -1.00 | -1.00 | -1.00 | -1.00 | 1.00 | -1.00 |
| -1.00 | -1.00 | 1.00 | -1.00 | -1.00 | -1.00 | 1.00 | -1.00 | -1.00 |
| -1.00 | -1.00 | -1.00 | -1.00 | 1.00 | -1.00 | -1.00 | -1.00 | -1.00 |
| -1.00 | -1.00 | -1.00 | 1.00 | -1.00 | 1.00 | -1.00 | -1.00 | -1.00 |
| -1.00 | -1.00 | 1.00 | -1.00 | -1.00 | -1.00 | 1.00 | -1.00 | -1.00 |
| -1.00 | 1.00 | -1.00 | -1.00 | -1.00 | -1.00 | -1.00 | 1.00 | -1.00 |
| -1.00 | -1.00 | -1.00 | -1.00 | -1.00 | -1.00 | -1.00 | -1.00 | -1.00 |

Convolution

**Image Filtrée**

| | | | | | | |
|---|---|---|---|---|---|---|
| 0.78 | -0.11 | 0.11 | 0.33 | 0.56 | -0.11 | 0.33 |
| -0.11 | 1.00 | -0.11 | 0.33 | -0.11 | 0.11 | -0.11 |
| 0.11 | -0.11 | 1.00 | -0.33 | 0.11 | -0.11 | 0.56 |
| 0.33 | 0.33 | -0.33 | 0.56 | -0.33 | 0.33 | 0.33 |
| 0.56 | -0.11 | 0.11 | -0.33 | 1.00 | -0.11 | 0.11 |
| -0.11 | 0.11 | -0.11 | 0.33 | -0.11 | 1.00 | -0.11 |
| 0.33 | -0.11 | 0.56 | 0.33 | 0.11 | -0.11 | 0.78 |

**Kernel size : 3 x 3**

**Stride : 1**

**Kernel**

| | | |
|---|---|---|
| 1.00 | -1.00 | -1.00 |
| -1.00 | 1.00 | -1.00 |
| -1.00 | -1.00 | 1.00 |

( (1)*(-1) + (-1)*(-1) + (-1)*(-1)+
(-1)*(-1) + (1)*(1) + (-1)*(-1)+
(-1)*(-1) + (-1)*(-1) + (1)*(1) ) /9 => -0.11

**Image Originale**

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| -1.00 | -1.00 | -1.00 | -1.00 | -1.00 | -1.00 | -1.00 | -1.00 | -1.00 |
| -1.00 | 1.00 | -1.00 | -1.00 | -1.00 | -1.00 | -1.00 | 1.00 | -1.00 |
| -1.00 | -1.00 | 1.00 | -1.00 | -1.00 | -1.00 | 1.00 | -1.00 | -1.00 |
| -1.00 | -1.00 | -1.00 | 1.00 | -1.00 | 1.00 | -1.00 | -1.00 | -1.00 |
| -1.00 | -1.00 | -1.00 | -1.00 | 1.00 | -1.00 | -1.00 | -1.00 | -1.00 |
| -1.00 | -1.00 | -1.00 | 1.00 | -1.00 | 1.00 | -1.00 | -1.00 | -1.00 |
| -1.00 | -1.00 | 1.00 | -1.00 | -1.00 | -1.00 | 1.00 | -1.00 | -1.00 |
| -1.00 | 1.00 | -1.00 | -1.00 | -1.00 | -1.00 | -1.00 | 1.00 | -1.00 |
| -1.00 | -1.00 | -1.00 | -1.00 | -1.00 | -1.00 | -1.00 | -1.00 | -1.00 |

Convolution

**Image Filtrée**

| | | | | | | |
|---|---|---|---|---|---|---|
| 0.78 | -0.11 | 0.11 | 0.33 | 0.56 | -0.11 | 0.33 |
| -0.11 | 1.00 | -0.11 | 0.33 | -0.11 | 0.11 | -0.11 |
| 0.11 | -0.11 | 1.00 | -0.33 | 0.11 | -0.11 | 0.56 |
| 0.33 | 0.33 | -0.33 | 0.56 | -0.33 | 0.33 | 0.33 |
| 0.56 | -0.11 | 0.11 | -0.33 | 1.00 | -0.11 | 0.11 |
| -0.11 | 0.11 | -0.11 | 0.33 | -0.11 | 1.00 | -0.11 |
| 0.33 | -0.11 | 0.56 | 0.33 | 0.11 | -0.11 | 0.78 |

**Kernel size : 3 x 3**

**Stride : 1**

**Kernel**

| | | |
|---|---|---|
| 1.00 | -1.00 | -1.00 |
| -1.00 | 1.00 | -1.00 |
| -1.00 | -1.00 | 1.00 |

( (1)*(-1) + (-1)*(-1) + (-1)*(-1)+
(-1)*(-1) + (1)*(1) + (-1)*(-1)+
(-1)*(-1) + (-1)*(-1) + (1)*(1) ) /9 => **0.33**

**Image Originale**

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| -1.00 | -1.00 | -1.00 | -1.00 | -1.00 | -1.00 | -1.00 | -1.00 | -1.00 |
| -1.00 | 1.00 | -1.00 | -1.00 | -1.00 | -1.00 | -1.00 | 1.00 | -1.00 |
| -1.00 | -1.00 | 1.00 | -1.00 | -1.00 | -1.00 | 1.00 | -1.00 | -1.00 |
| -1.00 | -1.00 | -1.00 | 1.00 | -1.00 | 1.00 | -1.00 | -1.00 | -1.00 |
| -1.00 | -1.00 | -1.00 | -1.00 | 1.00 | -1.00 | -1.00 | -1.00 | -1.00 |
| -1.00 | -1.00 | -1.00 | 1.00 | -1.00 | 1.00 | -1.00 | -1.00 | -1.00 |
| -1.00 | -1.00 | 1.00 | -1.00 | -1.00 | -1.00 | 1.00 | -1.00 | -1.00 |
| -1.00 | 1.00 | -1.00 | -1.00 | -1.00 | -1.00 | -1.00 | 1.00 | -1.00 |
| -1.00 | -1.00 | -1.00 | -1.00 | -1.00 | -1.00 | -1.00 | -1.00 | -1.00 |

Convolution

**Image Filtrée**

| | | | | | | |
|---|---|---|---|---|---|---|
| 0.78 | -0.11 | 0.11 | 0.33 | 0.56 | -0.11 | 0.33 |
| -0.11 | 1.00 | -0.11 | 0.33 | -0.11 | 0.11 | -0.11 |
| 0.11 | -0.11 | 1.00 | -0.33 | 0.11 | -0.11 | 0.56 |
| 0.33 | 0.33 | -0.33 | 0.56 | -0.33 | 0.33 | 0.33 |
| 0.56 | -0.11 | 0.11 | -0.33 | 1.00 | -0.11 | 0.11 |
| -0.11 | 0.11 | -0.11 | 0.33 | -0.11 | 1.00 | -0.11 |
| 0.33 | -0.11 | 0.56 | 0.33 | 0.11 | -0.11 | 0.78 |

**Kernel size : 3 x 3**

**Stride : 1**

**Kernel**

| | | |
|---|---|---|
| 1.00 | -1.00 | -1.00 |
| -1.00 | 1.00 | -1.00 |
| -1.00 | -1.00 | 1.00 |

( (1)*(-1) + (-1)*(-1) + (-1)*(-1)+
(-1)*(-1) + (1)*(1) + (-1)*(-1)+
(-1)*(-1) + (-1)*(-1) + (1)*(1) ) /9 => **-0.11**

**Image Originale**

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| -1.00 | -1.00 | -1.00 | -1.00 | -1.00 | -1.00 | -1.00 | -1.00 | -1.00 |
| -1.00 | 1.00 | -1.00 | -1.00 | -1.00 | -1.00 | -1.00 | 1.00 | -1.00 |
| -1.00 | -1.00 | 1.00 | -1.00 | -1.00 | -1.00 | 1.00 | -1.00 | -1.00 |
| -1.00 | -1.00 | -1.00 | 1.00 | -1.00 | 1.00 | -1.00 | -1.00 | -1.00 |
| -1.00 | -1.00 | -1.00 | -1.00 | 1.00 | -1.00 | -1.00 | -1.00 | -1.00 |
| -1.00 | -1.00 | -1.00 | 1.00 | -1.00 | 1.00 | -1.00 | -1.00 | -1.00 |
| -1.00 | -1.00 | 1.00 | -1.00 | -1.00 | -1.00 | 1.00 | -1.00 | -1.00 |
| -1.00 | 1.00 | -1.00 | -1.00 | -1.00 | -1.00 | -1.00 | 1.00 | -1.00 |
| -1.00 | -1.00 | -1.00 | -1.00 | -1.00 | -1.00 | -1.00 | -1.00 | -1.00 |

Convolution

**Image Filtrée**

| | | | | | | |
|---|---|---|---|---|---|---|
| 0.78 | -0.11 | 0.11 | 0.33 | 0.56 | -0.11 | 0.33 |
| -0.11 | 1.00 | -0.11 | 0.33 | -0.11 | 0.11 | -0.11 |
| 0.11 | -0.11 | 1.00 | -0.33 | 0.11 | -0.11 | 0.56 |
| 0.33 | 0.33 | -0.33 | 0.56 | -0.33 | 0.33 | 0.33 |
| 0.56 | -0.11 | 0.11 | -0.33 | 1.00 | -0.11 | 0.11 |
| -0.11 | 0.11 | -0.11 | 0.33 | -0.11 | 1.00 | -0.11 |
| 0.33 | -0.11 | 0.56 | 0.33 | 0.11 | -0.11 | 0.78 |

**Kernel size : 3 x 3**

**Stride : 1**

**Kernel**

| | | |
|---|---|---|
| 1.00 | -1.00 | -1.00 |
| -1.00 | 1.00 | -1.00 |
| -1.00 | -1.00 | 1.00 |

( (1)*(-1) + (-1)*(-1) + (-1)*(-1)+
(-1)*(-1) + (1)*(1) + (-1)*(-1)+
(-1)*(-1) + (-1)*(-1) + (1)*(1) ) /9 => **0.11**

**Image Originale**

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| -1.00 | -1.00 | -1.00 | -1.00 | -1.00 | -1.00 | -1.00 | -1.00 | -1.00 |
| -1.00 | 1.00 | -1.00 | -1.00 | -1.00 | -1.00 | -1.00 | 1.00 | -1.00 |
| -1.00 | -1.00 | 1.00 | -1.00 | -1.00 | -1.00 | 1.00 | -1.00 | -1.00 |
| -1.00 | -1.00 | -1.00 | 1.00 | -1.00 | 1.00 | -1.00 | -1.00 | -1.00 |
| -1.00 | -1.00 | -1.00 | -1.00 | 1.00 | -1.00 | -1.00 | -1.00 | -1.00 |
| -1.00 | -1.00 | -1.00 | 1.00 | -1.00 | 1.00 | -1.00 | -1.00 | -1.00 |
| -1.00 | -1.00 | 1.00 | -1.00 | -1.00 | -1.00 | 1.00 | -1.00 | -1.00 |
| -1.00 | 1.00 | -1.00 | -1.00 | -1.00 | -1.00 | -1.00 | 1.00 | -1.00 |
| -1.00 | -1.00 | -1.00 | -1.00 | -1.00 | -1.00 | -1.00 | -1.00 | -1.00 |

Convolution

**Image Filtrée**

| | | | | | | |
|---|---|---|---|---|---|---|
| 0.78 | -0.11 | 0.11 | 0.33 | 0.56 | -0.11 | 0.33 |
| -0.11 | 1.00 | -0.11 | 0.33 | -0.11 | 0.11 | -0.11 |
| 0.11 | -0.11 | 1.00 | -0.33 | 0.11 | -0.11 | 0.56 |
| 0.33 | 0.33 | -0.33 | 0.56 | -0.33 | 0.33 | 0.33 |
| 0.56 | -0.11 | 0.11 | -0.33 | 1.00 | -0.11 | 0.11 |
| -0.11 | 0.11 | -0.11 | 0.33 | -0.11 | 1.00 | -0.11 |
| 0.33 | -0.11 | 0.56 | 0.33 | 0.11 | -0.11 | 0.78 |

**Kernel size : 3 x 3**

**Stride : 1**

**Kernel**

| | | |
|---|---|---|
| 1.00 | -1.00 | -1.00 |
| -1.00 | 1.00 | -1.00 |
| -1.00 | -1.00 | 1.00 |

( (1)*(-1) + (-1)*(-1) + (-1)*(-1)+
(-1)*(-1) + (1)*(1) + (-1)*(-1)+
(-1)*(-1) + (-1)*(-1) + (1)*(1) ) /9 => **-0.11**

**Image Originale**

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| -1.00 | -1.00 | -1.00 | -1.00 | -1.00 | -1.00 | -1.00 | -1.00 | -1.00 |
| -1.00 | 1.00 | -1.00 | -1.00 | -1.00 | -1.00 | -1.00 | 1.00 | -1.00 |
| -1.00 | -1.00 | 1.00 | -1.00 | -1.00 | -1.00 | 1.00 | -1.00 | -1.00 |
| -1.00 | -1.00 | -1.00 | 1.00 | -1.00 | 1.00 | -1.00 | -1.00 | -1.00 |
| -1.00 | -1.00 | -1.00 | -1.00 | 1.00 | -1.00 | -1.00 | -1.00 | -1.00 |
| -1.00 | -1.00 | -1.00 | 1.00 | -1.00 | 1.00 | -1.00 | -1.00 | -1.00 |
| -1.00 | -1.00 | 1.00 | -1.00 | -1.00 | -1.00 | 1.00 | -1.00 | -1.00 |
| -1.00 | 1.00 | -1.00 | -1.00 | -1.00 | -1.00 | -1.00 | 1.00 | -1.00 |
| -1.00 | -1.00 | -1.00 | -1.00 | -1.00 | -1.00 | -1.00 | -1.00 | -1.00 |

Convolution

**Image Filtrée**

| | | | | | | |
|---|---|---|---|---|---|---|
| 0.78 | -0.11 | 0.11 | 0.33 | 0.56 | -0.11 | 0.33 |
| -0.11 | 1.00 | -0.11 | 0.33 | -0.11 | 0.11 | -0.11 |
| 0.11 | -0.11 | 1.00 | -0.33 | 0.11 | -0.11 | 0.56 |
| 0.33 | 0.33 | -0.33 | 0.56 | -0.33 | 0.33 | 0.33 |
| 0.56 | -0.11 | 0.11 | -0.33 | 1.00 | -0.11 | 0.11 |
| -0.11 | 0.11 | -0.11 | 0.33 | -0.11 | 1.00 | -0.11 |
| 0.33 | -0.11 | 0.56 | 0.33 | 0.11 | -0.11 | 0.78 |

**Kernel size : 3 x 3**

**Stride : 1**

**Kernel**

| | | |
|---|---|---|
| 1.00 | -1.00 | -1.00 |
| -1.00 | 1.00 | -1.00 |
| -1.00 | -1.00 | 1.00 |

( (1)*(-1) + (-1)*(-1) + (-1)*(-1)+
(-1)*(-1) + (1)*(1) + (-1)*(-1)+
(-1)*(-1) + (-1)*(-1) + (1)*(1) ) /9 => **0.11**

**Image Originale**

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| -1.00 | -1.00 | -1.00 | -1.00 | -1.00 | -1.00 | -1.00 | -1.00 | -1.00 |
| -1.00 | 1.00 | -1.00 | -1.00 | -1.00 | -1.00 | -1.00 | 1.00 | -1.00 |
| -1.00 | -1.00 | 1.00 | -1.00 | -1.00 | -1.00 | 1.00 | -1.00 | -1.00 |
| -1.00 | -1.00 | -1.00 | 1.00 | -1.00 | 1.00 | -1.00 | -1.00 | -1.00 |
| -1.00 | -1.00 | -1.00 | -1.00 | 1.00 | -1.00 | -1.00 | -1.00 | -1.00 |
| -1.00 | -1.00 | -1.00 | 1.00 | -1.00 | 1.00 | -1.00 | -1.00 | -1.00 |
| -1.00 | -1.00 | 1.00 | -1.00 | -1.00 | -1.00 | 1.00 | -1.00 | -1.00 |
| -1.00 | 1.00 | -1.00 | -1.00 | -1.00 | -1.00 | -1.00 | 1.00 | -1.00 |
| -1.00 | -1.00 | -1.00 | -1.00 | -1.00 | -1.00 | -1.00 | -1.00 | -1.00 |

Convolution

**Image Filtrée**

| | | | | | | |
|---|---|---|---|---|---|---|
| 0.78 | -0.11 | 0.11 | 0.33 | 0.56 | -0.11 | 0.33 |
| -0.11 | 1.00 | -0.11 | 0.33 | -0.11 | 0.11 | -0.11 |
| 0.11 | -0.11 | 1.00 | -0.33 | 0.11 | -0.11 | 0.56 |
| 0.33 | 0.33 | -0.33 | 0.56 | -0.33 | 0.33 | 0.33 |
| 0.56 | -0.11 | 0.11 | -0.33 | 1.00 | -0.11 | 0.11 |
| -0.11 | 0.11 | -0.11 | 0.33 | -0.11 | 1.00 | -0.11 |
| 0.33 | -0.11 | 0.56 | 0.33 | 0.11 | -0.11 | 0.78 |

**Kernel size : 3 x 3**

**Stride : 1**

**Kernel**

| | | |
|---|---|---|
| 1.00 | -1.00 | -1.00 |
| -1.00 | 1.00 | -1.00 |
| -1.00 | -1.00 | 1.00 |

( (1)*(-1) + (-1)*(-1) + (-1)*(-1)+
(-1)*(-1) + (1)*(1) + (-1)*(-1)+
(-1)*(-1) + (-1)*(-1) + (1)*(1) ) /9 => **-0.11**

**Image Originale**

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| -1.00 | -1.00 | -1.00 | -1.00 | -1.00 | -1.00 | -1.00 | -1.00 | -1.00 |
| -1.00 | 1.00 | -1.00 | -1.00 | -1.00 | -1.00 | -1.00 | 1.00 | -1.00 |
| -1.00 | -1.00 | 1.00 | -1.00 | -1.00 | -1.00 | 1.00 | -1.00 | -1.00 |
| -1.00 | -1.00 | -1.00 | 1.00 | -1.00 | 1.00 | -1.00 | -1.00 | -1.00 |
| -1.00 | -1.00 | -1.00 | -1.00 | 1.00 | -1.00 | -1.00 | -1.00 | -1.00 |
| -1.00 | -1.00 | -1.00 | 1.00 | -1.00 | 1.00 | -1.00 | -1.00 | -1.00 |
| -1.00 | -1.00 | 1.00 | -1.00 | -1.00 | -1.00 | 1.00 | -1.00 | -1.00 |
| -1.00 | 1.00 | -1.00 | -1.00 | -1.00 | -1.00 | -1.00 | 1.00 | -1.00 |
| -1.00 | -1.00 | -1.00 | -1.00 | -1.00 | -1.00 | -1.00 | -1.00 | -1.00 |

Convolution

**Image Filtrée**

| | | | | | | |
|---|---|---|---|---|---|---|
| 0.78 | -0.11 | 0.11 | 0.33 | 0.56 | -0.11 | 0.33 |
| -0.11 | 1.00 | -0.11 | 0.33 | -0.11 | 0.11 | -0.11 |
| 0.11 | -0.11 | 1.00 | -0.33 | 0.11 | -0.11 | 0.56 |
| 0.33 | 0.33 | -0.33 | 0.56 | -0.33 | 0.33 | 0.33 |
| 0.56 | -0.11 | 0.11 | -0.33 | 1.00 | -0.11 | 0.11 |
| -0.11 | 0.11 | -0.11 | 0.33 | -0.11 | 1.00 | -0.11 |
| 0.33 | -0.11 | 0.56 | 0.33 | 0.11 | -0.11 | 0.78 |

**Kernel size : 3 x 3**

**Stride : 1**

**Kernel**

| | | |
|---|---|---|
| 1.00 | -1.00 | -1.00 |
| -1.00 | 1.00 | -1.00 |
| -1.00 | -1.00 | 1.00 |

( (1)*(-1) + (-1)*(-1) + (-1)*(-1)+
(-1)*(-1) + (1)*(1) + (-1)*(-1)+
(-1)*(-1) + (-1)*(-1) + (1)*(1) ) /9 => **1**

**Image Originale**

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| -1.00 | -1.00 | -1.00 | -1.00 | -1.00 | -1.00 | -1.00 | -1.00 | -1.00 |
| -1.00 | 1.00 | -1.00 | -1.00 | -1.00 | -1.00 | -1.00 | 1.00 | -1.00 |
| -1.00 | -1.00 | 1.00 | -1.00 | -1.00 | -1.00 | 1.00 | -1.00 | -1.00 |
| -1.00 | -1.00 | -1.00 | 1.00 | -1.00 | 1.00 | -1.00 | -1.00 | -1.00 |
| -1.00 | -1.00 | -1.00 | -1.00 | 1.00 | -1.00 | -1.00 | -1.00 | -1.00 |
| -1.00 | -1.00 | -1.00 | 1.00 | -1.00 | 1.00 | -1.00 | -1.00 | -1.00 |
| -1.00 | -1.00 | 1.00 | -1.00 | -1.00 | -1.00 | 1.00 | -1.00 | -1.00 |
| -1.00 | 1.00 | -1.00 | -1.00 | -1.00 | -1.00 | -1.00 | 1.00 | -1.00 |
| -1.00 | -1.00 | -1.00 | -1.00 | -1.00 | -1.00 | -1.00 | -1.00 | -1.00 |

Convolution

**Image Filtrée**

| | | | | | | |
|---|---|---|---|---|---|---|
| 0.78 | -0.11 | 0.11 | 0.33 | 0.56 | -0.11 | 0.33 |
| -0.11 | 1.00 | -0.11 | 0.33 | -0.11 | 0.11 | -0.11 |
| 0.11 | -0.11 | 1.00 | -0.33 | 0.11 | -0.11 | 0.56 |
| 0.33 | 0.33 | -0.33 | 0.56 | -0.33 | 0.33 | 0.33 |
| 0.56 | -0.11 | 0.11 | -0.33 | 1.00 | -0.11 | 0.11 |
| -0.11 | 0.11 | -0.11 | 0.33 | -0.11 | 1.00 | -0.11 |
| 0.33 | -0.11 | 0.56 | 0.33 | 0.11 | -0.11 | 0.78 |

**Kernel size : 3 x 3**

**Stride : 1**

**Kernel**

| | | |
|---|---|---|
| 1.00 | -1.00 | -1.00 |
| -1.00 | 1.00 | -1.00 |
| -1.00 | -1.00 | 1.00 |

( (1)*(-1) + (-1)*(-1) + (-1)*(-1)+
(-1)*(-1) + (1)*(1) + (-1)*(-1)+
(-1)*(-1) + (-1)*(-1) + (1)*(1) ) /9 => **-0.33**

**Image Originale**

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| -1.00 | -1.00 | -1.00 | -1.00 | -1.00 | -1.00 | -1.00 | -1.00 | -1.00 |
| -1.00 | 1.00 | -1.00 | -1.00 | -1.00 | -1.00 | -1.00 | 1.00 | -1.00 |
| -1.00 | -1.00 | 1.00 | -1.00 | -1.00 | -1.00 | 1.00 | -1.00 | -1.00 |
| -1.00 | -1.00 | -1.00 | 1.00 | -1.00 | 1.00 | -1.00 | -1.00 | -1.00 |
| -1.00 | -1.00 | -1.00 | -1.00 | 1.00 | -1.00 | -1.00 | -1.00 | -1.00 |
| -1.00 | -1.00 | -1.00 | 1.00 | -1.00 | 1.00 | -1.00 | -1.00 | -1.00 |
| -1.00 | -1.00 | 1.00 | -1.00 | -1.00 | -1.00 | 1.00 | -1.00 | -1.00 |
| -1.00 | 1.00 | -1.00 | -1.00 | -1.00 | -1.00 | -1.00 | 1.00 | -1.00 |
| -1.00 | -1.00 | -1.00 | -1.00 | -1.00 | -1.00 | -1.00 | -1.00 | -1.00 |

Convolution

**Image Filtrée**

| | | | | | | |
|---|---|---|---|---|---|---|
| 0.78 | -0.11 | 0.11 | 0.33 | 0.56 | -0.11 | 0.33 |
| -0.11 | 1.00 | -0.11 | 0.33 | -0.11 | 0.11 | -0.11 |
| 0.11 | -0.11 | 1.00 | -0.33 | 0.11 | -0.11 | 0.56 |
| 0.33 | 0.33 | -0.33 | 0.56 | -0.33 | 0.33 | 0.33 |
| 0.56 | -0.11 | 0.11 | -0.33 | 1.00 | -0.11 | 0.11 |
| -0.11 | 0.11 | -0.11 | 0.33 | -0.11 | 1.00 | -0.11 |
| 0.33 | -0.11 | 0.56 | 0.33 | 0.11 | -0.11 | 0.78 |

**Kernel size : 3 x 3**

**Stride : 1**

**Kernel**

| | | |
|---|---|---|
| 1.00 | -1.00 | -1.00 |
| -1.00 | 1.00 | -1.00 |
| -1.00 | -1.00 | 1.00 |

( (1)*(-1) + (-1)*(-1) + (-1)*(-1)+
(-1)*(-1) + (1)*(1) + (-1)*(-1)+
(-1)*(-1) + (-1)*(-1) + (1)*(1) ) /9 => **0.11**

**Image Originale**

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| -1.00 | -1.00 | -1.00 | -1.00 | -1.00 | -1.00 | -1.00 | -1.00 | -1.00 |
| -1.00 | 1.00 | -1.00 | -1.00 | -1.00 | -1.00 | -1.00 | 1.00 | -1.00 |
| -1.00 | -1.00 | 1.00 | -1.00 | -1.00 | -1.00 | 1.00 | -1.00 | -1.00 |
| -1.00 | -1.00 | -1.00 | 1.00 | -1.00 | 1.00 | -1.00 | -1.00 | -1.00 |
| -1.00 | -1.00 | -1.00 | -1.00 | 1.00 | -1.00 | -1.00 | -1.00 | -1.00 |
| -1.00 | -1.00 | -1.00 | 1.00 | -1.00 | 1.00 | -1.00 | -1.00 | -1.00 |
| -1.00 | -1.00 | 1.00 | -1.00 | -1.00 | -1.00 | 1.00 | -1.00 | -1.00 |
| -1.00 | 1.00 | -1.00 | -1.00 | -1.00 | -1.00 | -1.00 | 1.00 | -1.00 |
| -1.00 | -1.00 | -1.00 | -1.00 | -1.00 | -1.00 | -1.00 | -1.00 | -1.00 |

Convolution

**Image Filtrée**

| | | | | | | |
|---|---|---|---|---|---|---|
| 0.78 | -0.11 | 0.11 | 0.33 | 0.56 | -0.11 | 0.33 |
| -0.11 | 1.00 | -0.11 | 0.33 | -0.11 | 0.11 | -0.11 |
| 0.11 | -0.11 | 1.00 | -0.33 | 0.11 | -0.11 | 0.56 |
| 0.33 | 0.33 | -0.33 | 0.56 | -0.33 | 0.33 | 0.33 |
| 0.56 | -0.11 | 0.11 | -0.33 | 1.00 | -0.11 | 0.11 |
| -0.11 | 0.11 | -0.11 | 0.33 | -0.11 | 1.00 | -0.11 |
| 0.33 | -0.11 | 0.56 | 0.33 | 0.11 | -0.11 | 0.78 |

**Kernel size : 3 x 3**

**Stride : 1**

**Kernel**

| | | |
|---|---|---|
| 1.00 | -1.00 | -1.00 |
| -1.00 | 1.00 | -1.00 |
| -1.00 | -1.00 | 1.00 |

( (1)*(-1) + (-1)*(-1) + (-1)*(-1)+
(-1)*(-1) + (1)*(1) + (-1)*(-1)+
(-1)*(-1) + (-1)*(-1) + (1)*(1) ) /9 => **0.78**

**Image Originale**

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| -1.00 | -1.00 | -1.00 | -1.00 | -1.00 | -1.00 | -1.00 | -1.00 | -1.00 |
| -1.00 | 1.00 | -1.00 | -1.00 | -1.00 | -1.00 | -1.00 | 1.00 | -1.00 |
| -1.00 | -1.00 | 1.00 | -1.00 | -1.00 | -1.00 | 1.00 | -1.00 | -1.00 |
| -1.00 | -1.00 | -1.00 | 1.00 | -1.00 | 1.00 | -1.00 | -1.00 | -1.00 |
| -1.00 | -1.00 | -1.00 | -1.00 | 1.00 | -1.00 | -1.00 | -1.00 | -1.00 |
| -1.00 | -1.00 | -1.00 | 1.00 | -1.00 | 1.00 | -1.00 | -1.00 | -1.00 |
| -1.00 | -1.00 | 1.00 | -1.00 | -1.00 | -1.00 | 1.00 | -1.00 | -1.00 |
| -1.00 | 1.00 | -1.00 | -1.00 | -1.00 | -1.00 | -1.00 | 1.00 | -1.00 |
| -1.00 | -1.00 | -1.00 | -1.00 | -1.00 | -1.00 | -1.00 | -1.00 | -1.00 |

Convolution

**Image Filtrée**

| | | | | | | |
|---|---|---|---|---|---|---|
| 0.78 | -0.11 | 0.11 | 0.33 | 0.56 | -0.11 | 0.33 |
| -0.11 | 1.00 | -0.11 | 0.33 | -0.11 | 0.11 | -0.11 |
| 0.11 | -0.11 | 1.00 | -0.33 | 0.11 | -0.11 | 0.56 |
| 0.33 | 0.33 | -0.33 | 0.56 | -0.33 | 0.33 | 0.33 |
| 0.56 | -0.11 | 0.11 | -0.33 | 1.00 | -0.11 | 0.11 |
| -0.11 | 0.11 | -0.11 | 0.33 | -0.11 | 1.00 | -0.11 |
| 0.33 | -0.11 | 0.56 | 0.33 | 0.11 | -0.11 | 0.78 |

**Kernel size : 3 x 3**

**Stride : 1**

**Kernel**

| | | |
|---|---|---|
| 1.00 | -1.00 | -1.00 |
| -1.00 | 1.00 | -1.00 |
| -1.00 | -1.00 | 1.00 |

( (1)*(-1) + (-1)*(-1) + (-1)*(-1)+
(-1)*(-1) + (1)*(1) + (-1)*(-1)+
(-1)*(-1) + (-1)*(-1) + (1)*(1) ) /9 => **0.58**

**Image Originale**

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| -1.00 | -1.00 | -1.00 | -1.00 | -1.00 | -1.00 | -1.00 | -1.00 | -1.00 |
| -1.00 | 1.00 | -1.00 | -1.00 | -1.00 | -1.00 | 1.00 | 1.00 | -1.00 |
| -1.00 | -1.00 | 1.00 | -1.00 | -1.00 | -1.00 | 1.00 | -1.00 | -1.00 |
| -1.00 | -1.00 | -1.00 | 1.00 | -1.00 | 1.00 | -1.00 | -1.00 | -1.00 |
| -1.00 | -1.00 | -1.00 | -1.00 | 1.00 | -1.00 | -1.00 | -1.00 | -1.00 |
| -1.00 | -1.00 | -1.00 | 1.00 | -1.00 | 1.00 | -1.00 | -1.00 | -1.00 |
| -1.00 | -1.00 | 1.00 | -1.00 | -1.00 | -1.00 | 1.00 | -1.00 | -1.00 |
| -1.00 | 1.00 | -1.00 | -1.00 | -1.00 | -1.00 | -1.00 | 1.00 | -1.00 |
| -1.00 | -1.00 | -1.00 | -1.00 | -1.00 | -1.00 | -1.00 | -1.00 | -1.00 |

Convolution

**Image Filtrée**

| | | | | | | |
|---|---|---|---|---|---|---|
| 0.78 | -0.11 | 0.11 | 0.33 | 0.56 | -0.11 | 0.33 |
| -0.11 | 1.00 | -0.11 | 0.33 | -0.11 | 0.11 | -0.11 |
| 0.11 | -0.11 | 1.00 | -0.33 | 0.11 | -0.11 | 0.56 |
| 0.33 | 0.33 | -0.33 | 0.56 | -0.33 | 0.33 | 0.33 |
| 0.56 | -0.11 | 0.11 | -0.33 | 1.00 | -0.11 | 0.11 |
| -0.11 | 0.11 | -0.11 | 0.33 | -0.11 | 1.00 | -0.11 |
| 0.33 | -0.11 | 0.56 | 0.33 | 0.11 | -0.11 | 0.78 |

**Kernel size : 3 x 3**

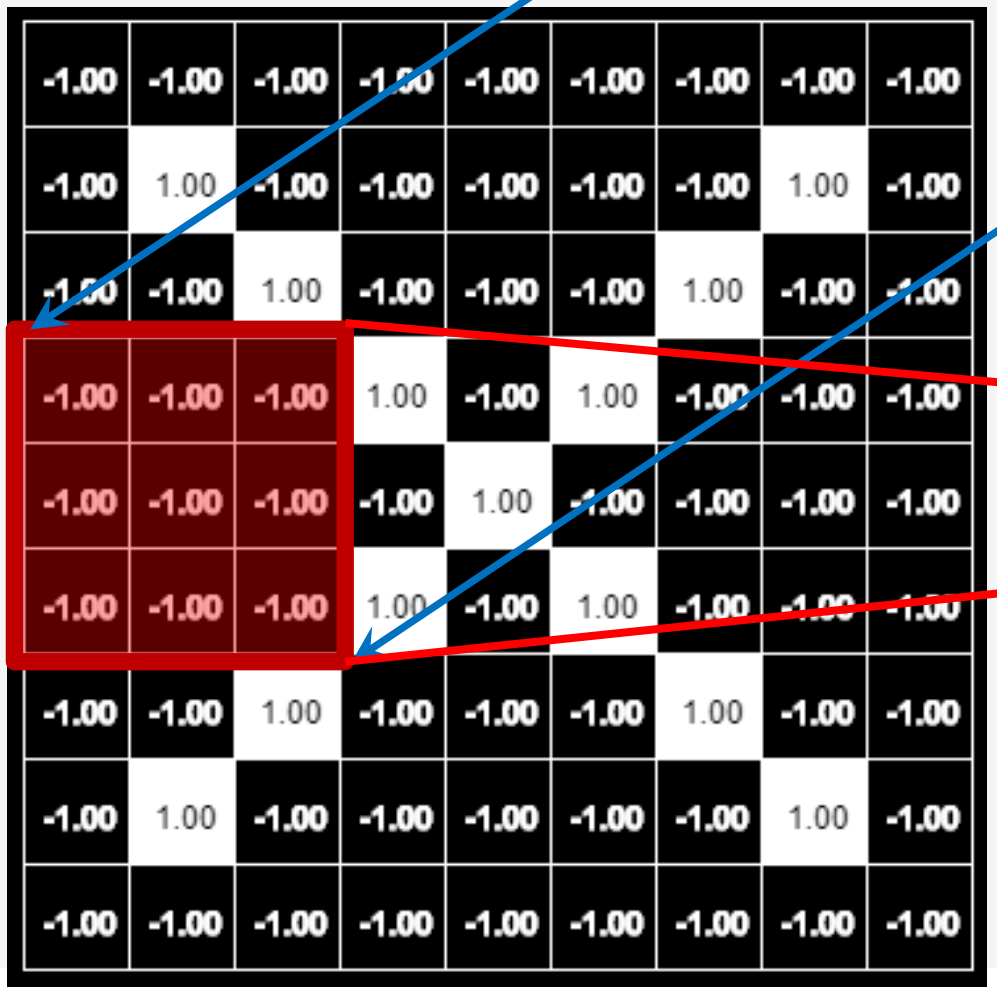
**Stride : 1**

**Kernel**

| | | |
|---|---|---|
| 1.00 | -1.00 | -1.00 |
| -1.00 | 1.00 | -1.00 |
| -1.00 | -1.00 | 1.00 |

( (1)*(-1) + (-1)*(-1) + (-1)*(-1)+
(-1)*(-1) + (1)*(1) + (-1)*(-1)+
(-1)*(-1) + (-1)*(-1) + (1)*(1) ) /9 => **0.33**

**Image Originale**




Convolution

**Image Filtrée**

**Kernel size : 3 x 3**

**Stride : 1**

**Kernel**

| | | |
|---|---|---|
| 1.00 | -1.00 | -1.00 |
| -1.00 | 1.00 | -1.00 |
| -1.00 | -1.00 | 1.00 |

( (1)*(-1) + (-1)*(-1) + (-1)*(-1)+
(-1)*(-1) + (1)*(1) + (-1)*(-1)+
(-1)*(-1) + (-1)*(-1) + (1)*(1) ) /9 => **0.33**

**Image Originale**

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| -1.00 | -1.00 | -1.00 | -1.00 | -1.00 | -1.00 | -1.00 | -1.00 | -1.00 |
| -1.00 | 1.00 | -1.00 | -1.00 | -1.00 | -1.00 | -1.00 | 1.00 | -1.00 |
| -1.00 | -1.00 | 1.00 | -1.00 | -1.00 | -1.00 | 1.00 | -1.00 | -1.00 |
| -1.00 | -1.00 | -1.00 | 1.00 | -1.00 | 1.00 | -1.00 | -1.00 | -1.00 |
| -1.00 | -1.00 | -1.00 | -1.00 | 1.00 | -1.00 | -1.00 | -1.00 | -1.00 |
| -1.00 | -1.00 | -1.00 | 1.00 | -1.00 | 1.00 | -1.00 | -1.00 | -1.00 |
| -1.00 | -1.00 | 1.00 | -1.00 | -1.00 | -1.00 | 1.00 | -1.00 | -1.00 |
| -1.00 | 1.00 | -1.00 | -1.00 | -1.00 | -1.00 | -1.00 | 1.00 | -1.00 |
| -1.00 | -1.00 | -1.00 | -1.00 | -1.00 | -1.00 | -1.00 | -1.00 | -1.00 |

Convolution

**Image Filtrée**

| | | | | | | |
|---|---|---|---|---|---|---|
| 0.78 | -0.11 | 0.11 | 0.33 | 0.56 | -0.11 | 0.33 |
| -0.11 | 1.00 | -0.11 | 0.33 | -0.11 | 0.11 | -0.11 |
| 0.11 | -0.11 | 1.00 | -0.33 | 0.11 | -0.11 | 0.56 |
| 0.33 | 0.33 | -0.33 | 0.56 | -0.33 | 0.33 | 0.33 |
| 0.56 | -0.11 | 0.11 | -0.33 | 1.00 | -0.11 | 0.11 |
| -0.11 | 0.11 | -0.11 | 0.33 | -0.11 | 1.00 | -0.11 |
| 0.33 | -0.11 | 0.56 | 0.33 | 0.11 | -0.11 | 0.78 |

**Kernel size : 3 x 3**

**Stride : 1**

**Kernel**

| | | |
|---|---|---|
| 1.00 | -1.00 | -1.00 |
| -1.00 | 1.00 | -1.00 |
| -1.00 | -1.00 | 1.00 |

( (1)*(-1) + (-1)*(-1) + (-1)*(-1)+
(-1)*(-1) + (1)*(1) + (-1)*(-1)+
(-1)*(-1) + (-1)*(-1) + (1)*(1) ) /9 => **-0.33**

**Image Originale**

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| -1.00 | -1.00 | -1.00 | -1.00 | -1.00 | -1.00 | -1.00 | -1.00 | -1.00 |
| -1.00 | 1.00 | -1.00 | -1.00 | -1.00 | -1.00 | -1.00 | 1.00 | -1.00 |
| -1.00 | -1.00 | 1.00 | -1.00 | -1.00 | -1.00 | 1.00 | -1.00 | -1.00 |
| -1.00 | -1.00 | -1.00 | 1.00 | -1.00 | 1.00 | -1.00 | -1.00 | -1.00 |
| -1.00 | -1.00 | -1.00 | -1.00 | 1.00 | -1.00 | -1.00 | -1.00 | -1.00 |
| -1.00 | -1.00 | -1.00 | 1.00 | -1.00 | 1.00 | -1.00 | -1.00 | -1.00 |
| -1.00 | -1.00 | 1.00 | -1.00 | -1.00 | -1.00 | 1.00 | -1.00 | -1.00 |
| -1.00 | 1.00 | -1.00 | -1.00 | -1.00 | -1.00 | -1.00 | 1.00 | -1.00 |
| -1.00 | -1.00 | -1.00 | -1.00 | -1.00 | -1.00 | -1.00 | -1.00 | -1.00 |

Convolution

**Image Filtrée**

| | | | | | | |
|---|---|---|---|---|---|---|
| 0.78 | -0.11 | 0.11 | 0.33 | 0.56 | -0.11 | 0.33 |
| -0.11 | 1.00 | -0.11 | 0.33 | -0.11 | 0.11 | -0.11 |
| 0.11 | -0.11 | 1.00 | -0.33 | 0.11 | -0.11 | 0.56 |
| 0.33 | 0.33 | -0.33 | 0.56 | -0.33 | 0.33 | 0.33 |
| 0.56 | -0.11 | 0.11 | -0.33 | 1.00 | -0.11 | 0.11 |
| -0.11 | 0.11 | -0.11 | 0.33 | -0.11 | 1.00 | -0.11 |
| 0.33 | -0.11 | 0.56 | 0.33 | 0.11 | -0.11 | 0.78 |

**Kernel size : 3 x 3**

**Stride : 1**

**Kernel**

| | | |
|---|---|---|
| 1.00 | -1.00 | -1.00 |
| -1.00 | 1.00 | -1.00 |
| -1.00 | -1.00 | 1.00 |

( (1)*(-1) + (-1)*(-1) + (-1)*(-1)+
(-1)*(-1) + (1)*(1) + (-1)*(-1)+
(-1)*(-1) + (-1)*(-1) + (1)*(1) ) /9 => **0.56**

**Image Originale**

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| -1.00 | -1.00 | -1.00 | -1.00 | -1.00 | -1.00 | -1.00 | -1.00 | -1.00 |
| -1.00 | 1.00 | -1.00 | -1.00 | -1.00 | -1.00 | -1.00 | 1.00 | -1.00 |
| -1.00 | -1.00 | 1.00 | -1.00 | -1.00 | -1.00 | 1.00 | -1.00 | -1.00 |
| -1.00 | -1.00 | -1.00 | 1.00 | -1.00 | 1.00 | -1.00 | -1.00 | -1.00 |
| -1.00 | -1.00 | -1.00 | -1.00 | 1.00 | -1.00 | -1.00 | -1.00 | -1.00 |
| -1.00 | -1.00 | -1.00 | 1.00 | -1.00 | 1.00 | -1.00 | -1.00 | -1.00 |
| -1.00 | -1.00 | 1.00 | -1.00 | -1.00 | -1.00 | 1.00 | -1.00 | -1.00 |
| -1.00 | 1.00 | -1.00 | -1.00 | -1.00 | -1.00 | -1.00 | 1.00 | -1.00 |
| -1.00 | -1.00 | -1.00 | -1.00 | -1.00 | -1.00 | -1.00 | -1.00 | -1.00 |

Convolution

**Image Filtrée**

| | | | | | | |
|---|---|---|---|---|---|---|
| 0.78 | -0.11 | 0.11 | 0.33 | 0.56 | -0.11 | 0.33 |
| -0.11 | 1.00 | -0.11 | 0.33 | -0.11 | 0.11 | -0.11 |
| 0.11 | -0.11 | 1.00 | -0.33 | 0.11 | -0.11 | 0.56 |
| 0.33 | 0.33 | -0.33 | 0.56 | -0.33 | 0.33 | 0.33 |
| 0.56 | -0.11 | 0.11 | -0.33 | 1.00 | -0.11 | 0.11 |
| -0.11 | 0.11 | -0.11 | 0.33 | -0.11 | 1.00 | -0.11 |
| 0.33 | -0.11 | 0.56 | 0.33 | 0.11 | -0.11 | 0.78 |

**Kernel size : 3 x 3**

**Stride : 1**

**Kernel**

| | | |
|---|---|---|
| 1.00 | -1.00 | -1.00 |
| -1.00 | 1.00 | -1.00 |
| -1.00 | -1.00 | 1.00 |

( (1)*(-1) + (-1)*(-1) + (-1)*(-1)+
(-1)*(-1) + (1)*(1) + (-1)*(-1)+
(-1)*(-1) + (-1)*(-1) + (1)*(1) ) /9 => **-0.33**

**Image Originale**

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| -1.00 | -1.00 | -1.00 | -1.00 | -1.00 | -1.00 | -1.00 | -1.00 | -1.00 |
| -1.00 | 1.00 | -1.00 | -1.00 | -1.00 | -1.00 | -1.00 | 1.00 | -1.00 |
| -1.00 | -1.00 | 1.00 | -1.00 | -1.00 | -1.00 | 1.00 | -1.00 | -1.00 |
| -1.00 | -1.00 | -1.00 | 1.00 | -1.00 | 1.00 | -1.00 | -1.00 | -1.00 |
| -1.00 | -1.00 | -1.00 | -1.00 | 1.00 | -1.00 | -1.00 | -1.00 | -1.00 |
| -1.00 | -1.00 | -1.00 | 1.00 | -1.00 | 1.00 | -1.00 | -1.00 | -1.00 |
| -1.00 | -1.00 | 1.00 | -1.00 | -1.00 | -1.00 | 1.00 | -1.00 | -1.00 |
| -1.00 | 1.00 | -1.00 | -1.00 | -1.00 | -1.00 | -1.00 | 1.00 | -1.00 |
| -1.00 | -1.00 | -1.00 | -1.00 | -1.00 | -1.00 | -1.00 | -1.00 | -1.00 |

Convolution

**Image Filtrée**

| | | | | | | |
|---|---|---|---|---|---|---|
| 0.78 | -0.11 | 0.11 | 0.33 | 0.56 | -0.11 | 0.33 |
| -0.11 | 1.00 | -0.11 | 0.33 | -0.11 | 0.11 | -0.11 |
| 0.11 | -0.11 | 1.00 | -0.33 | 0.11 | -0.11 | 0.56 |
| 0.33 | 0.33 | -0.33 | 0.56 | -0.33 | 0.33 | 0.33 |
| 0.56 | -0.11 | 0.11 | -0.33 | 1.00 | -0.11 | 0.11 |
| -0.11 | 0.11 | -0.11 | 0.33 | -0.11 | 1.00 | -0.11 |
| 0.33 | -0.11 | 0.56 | 0.33 | 0.11 | -0.11 | 0.78 |

**Kernel size : 3 x 3**

**Stride : 1**

**Kernel**

| | | |
|---|---|---|
| 1.00 | -1.00 | -1.00 |
| -1.00 | 1.00 | -1.00 |
| -1.00 | -1.00 | 1.00 |

( (1)*(-1) + (-1)*(-1) + (-1)*(-1)+
(-1)*(-1) + (1)*(1) + (-1)*(-1)+
(-1)*(-1) + (-1)*(-1) + (1)*(1) ) /9 => **0.33**

**Image Originale**

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| -1.00 | -1.00 | -1.00 | -1.00 | -1.00 | -1.00 | -1.00 | -1.00 | -1.00 |
| -1.00 | 1.00 | -1.00 | -1.00 | -1.00 | -1.00 | -1.00 | 1.00 | -1.00 |
| -1.00 | -1.00 | 1.00 | -1.00 | -1.00 | -1.00 | 1.00 | -1.00 | -1.00 |
| -1.00 | -1.00 | -1.00 | 1.00 | -1.00 | 1.00 | -1.00 | -1.00 | -1.00 |
| -1.00 | -1.00 | -1.00 | -1.00 | 1.00 | -1.00 | -1.00 | -1.00 | -1.00 |
| -1.00 | -1.00 | -1.00 | 1.00 | -1.00 | 1.00 | -1.00 | -1.00 | -1.00 |
| -1.00 | -1.00 | 1.00 | -1.00 | -1.00 | -1.00 | 1.00 | -1.00 | -1.00 |
| -1.00 | 1.00 | -1.00 | -1.00 | -1.00 | -1.00 | -1.00 | 1.00 | -1.00 |
| -1.00 | -1.00 | -1.00 | -1.00 | -1.00 | -1.00 | -1.00 | -1.00 | -1.00 |

Convolution

**Image Filtrée**

| | | | | | | |
|---|---|---|---|---|---|---|
| 0.78 | -0.11 | 0.11 | 0.33 | 0.56 | -0.11 | 0.33 |
| -0.11 | 1.00 | -0.11 | 0.33 | -0.11 | 0.11 | -0.11 |
| 0.11 | -0.11 | 1.00 | -0.33 | 0.11 | -0.11 | 0.56 |
| 0.33 | 0.33 | -0.33 | 0.56 | -0.33 | 0.33 | 0.33 |
| 0.56 | -0.11 | 0.11 | -0.33 | 1.00 | -0.11 | 0.11 |
| -0.11 | 0.11 | -0.11 | 0.33 | -0.11 | 1.00 | -0.11 |
| 0.33 | -0.11 | 0.56 | 0.33 | 0.11 | -0.11 | 0.78 |

**Kernel size : 3 x 3**

**Stride : 1**

**Kernel**

| | | |
|---|---|---|
| 1.00 | -1.00 | -1.00 |
| -1.00 | 1.00 | -1.00 |
| -1.00 | -1.00 | 1.00 |

( (1)*(-1) + (-1)*(-1) + (-1)*(-1)+
(-1)*(-1) + (1)*(1) + (-1)*(-1)+
(-1)*(-1) + (-1)*(-1) + (1)*(1) ) /9 => **0.33**

**Image Originale**

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| -1.00 | -1.00 | -1.00 | -1.00 | -1.00 | -1.00 | -1.00 | -1.00 | -1.00 |
| -1.00 | 1.00 | -1.00 | -1.00 | -1.00 | -1.00 | -1.00 | 1.00 | -1.00 |
| -1.00 | -1.00 | 1.00 | -1.00 | -1.00 | -1.00 | 1.00 | -1.00 | -1.00 |
| -1.00 | -1.00 | -1.00 | 1.00 | -1.00 | 1.00 | -1.00 | -1.00 | -1.00 |
| -1.00 | -1.00 | -1.00 | -1.00 | 1.00 | -1.00 | -1.00 | -1.00 | -1.00 |
| -1.00 | -1.00 | -1.00 | 1.00 | -1.00 | 1.00 | -1.00 | -1.00 | -1.00 |
| -1.00 | -1.00 | 1.00 | -1.00 | -1.00 | -1.00 | 1.00 | -1.00 | -1.00 |
| -1.00 | 1.00 | -1.00 | -1.00 | -1.00 | -1.00 | -1.00 | 1.00 | -1.00 |
| -1.00 | -1.00 | -1.00 | -1.00 | -1.00 | -1.00 | -1.00 | -1.00 | -1.00 |

Convolution

**Image Filtrée**

| | | | | | | |
|---|---|---|---|---|---|---|
| 0.78 | -0.11 | 0.11 | 0.33 | 0.56 | -0.11 | 0.33 |
| -0.11 | 1.00 | -0.11 | 0.33 | -0.11 | 0.11 | -0.11 |
| 0.11 | 0.11 | 1.00 | -0.33 | 0.11 | -0.11 | 0.56 |
| 0.33 | 0.33 | -0.33 | 0.56 | -0.33 | 0.33 | 0.33 |
| 0.56 | -0.11 | 0.11 | -0.33 | 1.00 | -0.11 | 0.11 |
| -0.11 | 0.11 | -0.11 | 0.33 | -0.11 | 1.00 | -0.11 |
| 0.33 | -0.11 | 0.56 | 0.33 | 0.11 | -0.11 | 0.78 |

**Kernel size : 3 x 3**

**Stride : 1**

**Kernel**

| | | |
|---|---|---|
| 1.00 | -1.00 | -1.00 |
| -1.00 | 1.00 | -1.00 |
| -1.00 | -1.00 | 1.00 |

( (1)*(-1) + (-1)*(-1) + (-1)*(-1)+
(-1)*(-1) + (1)*(1) + (-1)*(-1)+
(-1)*(-1) + (-1)*(-1) + (1)*(1) ) /9 => **0.56**

**Image Originale**

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| -1.00 | -1.00 | -1.00 | -1.00 | -1.00 | -1.00 | -1.00 | -1.00 | -1.00 |
| -1.00 | 1.00 | -1.00 | -1.00 | -1.00 | -1.00 | -1.00 | 1.00 | -1.00 |
| -1.00 | -1.00 | 1.00 | -1.00 | -1.00 | -1.00 | 1.00 | -1.00 | -1.00 |
| -1.00 | -1.00 | -1.00 | 1.00 | -1.00 | 1.00 | -1.00 | -1.00 | -1.00 |
| -1.00 | -1.00 | -1.00 | -1.00 | 1.00 | -1.00 | -1.00 | -1.00 | -1.00 |
| -1.00 | -1.00 | -1.00 | 1.00 | -1.00 | 1.00 | -1.00 | -1.00 | -1.00 |
| -1.00 | -1.00 | 1.00 | -1.00 | -1.00 | -1.00 | 1.00 | -1.00 | -1.00 |
| -1.00 | 1.00 | -1.00 | -1.00 | -1.00 | -1.00 | -1.00 | 1.00 | -1.00 |
| -1.00 | -1.00 | -1.00 | -1.00 | -1.00 | -1.00 | -1.00 | -1.00 | -1.00 |

Convolution

**Image Filtrée**

| | | | | | | |
|---|---|---|---|---|---|---|
| 0.78 | -0.11 | 0.11 | 0.33 | 0.56 | -0.11 | 0.33 |
| -0.11 | 1.00 | -0.11 | 0.33 | -0.11 | 0.11 | -0.11 |
| 0.11 | -0.11 | 1.00 | -0.33 | 0.11 | -0.11 | 0.56 |
| 0.33 | 0.33 | -0.33 | 0.56 | -0.33 | 0.33 | 0.33 |
| 0.56 | -0.11 | 0.11 | -0.33 | 1.00 | -0.11 | 0.11 |
| -0.11 | 0.11 | -0.11 | 0.33 | -0.11 | 1.00 | -0.11 |
| 0.33 | -0.11 | 0.56 | 0.33 | 0.11 | -0.11 | 0.78 |

**Kernel size : 3 x 3**

**Stride : 1**

**Kernel**

| | | |
|---|---|---|
| 1.00 | -1.00 | -1.00 |
| -1.00 | 1.00 | -1.00 |
| -1.00 | -1.00 | 1.00 |

( (1)*(-1) + (-1)*(-1) + (-1)*(-1)+
(-1)*(-1) + (1)*(1) + (-1)*(-1)+
(-1)*(-1) + (-1)*(-1) + (1)*(1) ) /9 => **-0.11**

**Image Originale**

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| -1.00 | -1.00 | -1.00 | -1.00 | -1.00 | -1.00 | -1.00 | -1.00 | -1.00 |
| -1.00 | 1.00 | -1.00 | -1.00 | -1.00 | -1.00 | -1.00 | 1.00 | -1.00 |
| -1.00 | -1.00 | 1.00 | -1.00 | -1.00 | -1.00 | 1.00 | -1.00 | -1.00 |
| -1.00 | -1.00 | -1.00 | 1.00 | -1.00 | 1.00 | -1.00 | -1.00 | -1.00 |
| -1.00 | -1.00 | -1.00 | -1.00 | 1.00 | -1.00 | -1.00 | -1.00 | -1.00 |
| -1.00 | -1.00 | -1.00 | 1.00 | -1.00 | 1.00 | -1.00 | -1.00 | -1.00 |
| -1.00 | -1.00 | 1.00 | -1.00 | -1.00 | 1.00 | -1.00 | -1.00 | -1.00 |
| -1.00 | 1.00 | -1.00 | -1.00 | -1.00 | -1.00 | -1.00 | 1.00 | -1.00 |
| -1.00 | -1.00 | -1.00 | -1.00 | -1.00 | -1.00 | -1.00 | -1.00 | -1.00 |

Convolution

**Image Filtrée**

| | | | | | | |
|---|---|---|---|---|---|---|
| 0.78 | -0.11 | 0.11 | 0.33 | 0.56 | -0.11 | 0.33 |
| -0.11 | 1.00 | -0.11 | 0.33 | -0.11 | 0.11 | -0.11 |
| 0.11 | -0.11 | 1.00 | -0.33 | 0.11 | -0.11 | 0.56 |
| 0.33 | 0.33 | -0.33 | 0.56 | -0.33 | 0.33 | 0.33 |
| 0.56 | -0.11 | 0.11 | -0.33 | 1.00 | -0.11 | 0.11 |
| -0.11 | 0.11 | -0.11 | 0.33 | -0.11 | 1.00 | -0.11 |
| 0.33 | -0.11 | 0.56 | 0.33 | 0.11 | -0.11 | 0.78 |

**Kernel size : 3 x 3**

**Stride : 1**

**Kernel**

| | | |
|---|---|---|
| 1.00 | -1.00 | -1.00 |
| -1.00 | 1.00 | -1.00 |
| -1.00 | -1.00 | 1.00 |

( (1)*(-1) + (-1)*(-1) + (-1)*(-1)+
(-1)*(-1) + (1)*(1) + (-1)*(-1)+
(-1)*(-1) + (-1)*(-1) + (1)*(1) ) /9 => **0.11**

**Image Originale**

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| -1.00 | -1.00 | -1.00 | -1.00 | -1.00 | -1.00 | -1.00 | -1.00 | -1.00 |
| -1.00 | 1.00 | -1.00 | -1.00 | -1.00 | -1.00 | -1.00 | 1.00 | -1.00 |
| -1.00 | -1.00 | 1.00 | -1.00 | -1.00 | -1.00 | 1.00 | -1.00 | -1.00 |
| -1.00 | -1.00 | -1.00 | 1.00 | -1.00 | 1.00 | -1.00 | -1.00 | -1.00 |
| -1.00 | -1.00 | -1.00 | -1.00 | 1.00 | -1.00 | -1.00 | -1.00 | -1.00 |
| -1.00 | -1.00 | -1.00 | 1.00 | -1.00 | 1.00 | -1.00 | -1.00 | -1.00 |
| -1.00 | -1.00 | 1.00 | -1.00 | -1.00 | -1.00 | 1.00 | -1.00 | -1.00 |
| -1.00 | 1.00 | -1.00 | -1.00 | -1.00 | -1.00 | -1.00 | 1.00 | -1.00 |
| -1.00 | -1.00 | -1.00 | -1.00 | -1.00 | -1.00 | -1.00 | -1.00 | -1.00 |

Convolution

**Image Filtrée**

| | | | | | | |
|---|---|---|---|---|---|---|
| 0.78 | -0.11 | 0.11 | 0.33 | 0.56 | -0.11 | 0.33 |
| -0.11 | 1.00 | -0.11 | 0.33 | -0.11 | 0.11 | -0.11 |
| 0.11 | -0.11 | 1.00 | -0.33 | 0.11 | -0.11 | 0.56 |
| 0.33 | 0.33 | -0.33 | 0.56 | -0.33 | 0.33 | 0.33 |
| 0.56 | -0.11 | 0.11 | -0.33 | 1.00 | -0.11 | 0.11 |
| -0.11 | 0.11 | -0.11 | 0.33 | -0.11 | 1.00 | -0.11 |
| 0.33 | -0.11 | 0.56 | 0.33 | 0.11 | -0.11 | 0.78 |

**Kernel size : 3 x 3**

**Stride : 1**

**Kernel**

| | | |
|---|---|---|
| 1.00 | -1.00 | -1.00 |
| -1.00 | 1.00 | -1.00 |
| -1.00 | -1.00 | 1.00 |

( (1)*(-1) + (-1)*(-1) + (-1)*(-1)+
(-1)*(-1) + (1)*(1) + (-1)*(-1)+
(-1)*(-1) + (-1)*(-1) + (1)*(1) ) /9 => **-0.33**

**Image Originale**

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| -1.00 | -1.00 | -1.00 | -1.00 | -1.00 | -1.00 | -1.00 | -1.00 | -1.00 |
| -1.00 | 1.00 | -1.00 | -1.00 | -1.00 | -1.00 | -1.00 | 1.00 | -1.00 |
| -1.00 | -1.00 | 1.00 | -1.00 | -1.00 | -1.00 | 1.00 | -1.00 | -1.00 |
| -1.00 | -1.00 | -1.00 | 1.00 | -1.00 | 1.00 | -1.00 | -1.00 | -1.00 |
| -1.00 | -1.00 | -1.00 | -1.00 | 1.00 | -1.00 | -1.00 | -1.00 | -1.00 |
| -1.00 | -1.00 | -1.00 | 1.00 | -1.00 | 1.00 | -1.00 | -1.00 | -1.00 |
| -1.00 | -1.00 | 1.00 | -1.00 | -1.00 | -1.00 | 1.00 | -1.00 | -1.00 |
| -1.00 | 1.00 | -1.00 | -1.00 | -1.00 | -1.00 | -1.00 | 1.00 | -1.00 |
| -1.00 | -1.00 | -1.00 | -1.00 | -1.00 | -1.00 | -1.00 | -1.00 | -1.00 |

Convolution

**Image Filtrée**

| | | | | | | |
|---|---|---|---|---|---|---|
| 0.78 | -0.11 | 0.11 | 0.33 | 0.56 | -0.11 | 0.33 |
| -0.11 | 1.00 | -0.11 | 0.33 | -0.11 | 0.11 | -0.11 |
| 0.11 | -0.11 | 1.00 | -0.33 | 0.11 | -0.11 | 0.56 |
| 0.33 | 0.33 | -0.33 | 0.56 | -0.33 | 0.33 | 0.33 |
| 0.56 | -0.11 | 0.11 | -0.33 | 1.00 | -0.11 | 0.11 |
| -0.11 | 0.11 | -0.11 | 0.33 | -0.11 | 1.00 | -0.11 |
| 0.33 | -0.11 | 0.56 | 0.33 | 0.11 | -0.11 | 0.78 |

**Kernel size : 3 x 3**

**Stride : 1**

**Kernel**

| | | |
|---|---|---|
| 1.00 | -1.00 | -1.00 |
| -1.00 | 1.00 | -1.00 |
| -1.00 | -1.00 | 1.00 |

( (1)*(-1) + (-1)*(-1) + (-1)*(-1)+
(-1)*(-1) + (1)*(1) + (-1)*(-1)+
(-1)*(-1) + (-1)*(-1) + (1)*(1) ) /9 => 1

**Image Originale**

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| -1.00 | -1.00 | -1.00 | -1.00 | -1.00 | -1.00 | -1.00 | -1.00 | -1.00 |
| -1.00 | 1.00 | -1.00 | -1.00 | -1.00 | -1.00 | -1.00 | 1.00 | -1.00 |
| -1.00 | -1.00 | 1.00 | -1.00 | -1.00 | -1.00 | 1.00 | -1.00 | -1.00 |
| -1.00 | -1.00 | -1.00 | 1.00 | -1.00 | 1.00 | -1.00 | -1.00 | -1.00 |
| -1.00 | -1.00 | -1.00 | -1.00 | 1.00 | -1.00 | -1.00 | -1.00 | -1.00 |
| -1.00 | -1.00 | -1.00 | 1.00 | -1.00 | 1.00 | -1.00 | -1.00 | -1.00 |
| -1.00 | -1.00 | 1.00 | -1.00 | -1.00 | -1.00 | 1.00 | -1.00 | -1.00 |
| -1.00 | 1.00 | -1.00 | -1.00 | -1.00 | -1.00 | -1.00 | 1.00 | -1.00 |
| -1.00 | -1.00 | -1.00 | -1.00 | -1.00 | -1.00 | -1.00 | -1.00 | -1.00 |

Convolution

**Image Filtrée**

| | | | | | | |
|---|---|---|---|---|---|---|
| 0.78 | -0.11 | 0.11 | 0.33 | 0.56 | -0.11 | 0.33 |
| -0.11 | 1.00 | -0.11 | 0.33 | -0.11 | 0.11 | -0.11 |
| 0.11 | -0.11 | 1.00 | -0.33 | 0.11 | -0.11 | 0.56 |
| 0.33 | 0.33 | -0.33 | 0.56 | -0.33 | 0.33 | 0.33 |
| 0.56 | -0.11 | 0.11 | -0.33 | 1.00 | -0.11 | 0.11 |
| -0.11 | 0.11 | -0.11 | 0.33 | -0.11 | 1.00 | -0.11 |
| 0.33 | -0.11 | 0.56 | 0.33 | 0.11 | -0.11 | 0.78 |

**Kernel size : 3 x 3**

**Stride : 1**

**Kernel**

| | | |
|---|---|---|
| 1.00 | -1.00 | -1.00 |
| -1.00 | 1.00 | -1.00 |
| -1.00 | -1.00 | 1.00 |

( (1)*(-1) + (-1)*(-1) + (-1)*(-1)+
(-1)*(-1) + (1)*(1) + (-1)*(-1)+
(-1)*(-1) + (-1)*(-1) + (1)*(1) ) /9 => **-0.11**

**Image Originale**

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| -1.00 | -1.00 | -1.00 | -1.00 | -1.00 | -1.00 | -1.00 | -1.00 | -1.00 |
| -1.00 | 1.00 | -1.00 | -1.00 | -1.00 | -1.00 | -1.00 | 1.00 | -1.00 |
| -1.00 | -1.00 | 1.00 | -1.00 | -1.00 | -1.00 | 1.00 | -1.00 | -1.00 |
| -1.00 | -1.00 | -1.00 | 1.00 | -1.00 | 1.00 | -1.00 | -1.00 | -1.00 |
| -1.00 | -1.00 | -1.00 | -1.00 | 1.00 | -1.00 | -1.00 | -1.00 | -1.00 |
| -1.00 | -1.00 | -1.00 | 1.00 | -1.00 | 1.00 | -1.00 | -1.00 | -1.00 |
| -1.00 | -1.00 | 1.00 | -1.00 | -1.00 | -1.00 | 1.00 | -1.00 | -1.00 |
| -1.00 | 1.00 | -1.00 | -1.00 | -1.00 | -1.00 | -1.00 | 1.00 | -1.00 |
| -1.00 | -1.00 | -1.00 | -1.00 | -1.00 | -1.00 | -1.00 | -1.00 | -1.00 |

Convolution

**Image Filtrée**

| | | | | | | |
|---|---|---|---|---|---|---|
| 0.78 | -0.11 | 0.11 | 0.33 | 0.56 | -0.11 | 0.33 |
| -0.11 | 1.00 | -0.11 | 0.33 | -0.11 | 0.11 | -0.11 |
| 0.11 | -0.11 | 1.00 | -0.33 | 0.11 | -0.11 | 0.56 |
| 0.33 | 0.33 | -0.33 | 0.56 | -0.33 | 0.33 | 0.33 |
| 0.56 | -0.11 | 0.11 | -0.33 | 1.00 | -0.11 | 0.11 |
| -0.11 | 0.11 | -0.11 | 0.33 | -0.11 | 1.00 | -0.11 |
| 0.33 | -0.11 | 0.56 | 0.33 | 0.11 | -0.11 | 0.78 |

**Kernel size : 3 x 3**

**Stride : 1**

**Kernel**

| | | |
|---|---|---|
| 1.00 | -1.00 | -1.00 |
| -1.00 | 1.00 | -1.00 |
| -1.00 | -1.00 | 1.00 |

( (1)*(-1) + (-1)*(-1) + (-1)*(-1)+
(-1)*(-1) + (1)*(1) + (-1)*(-1)+
(-1)*(-1) + (-1)*(-1) + (1)*(1) ) /9 => **0.11**

**Image Originale**

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| -1.00 | -1.00 | -1.00 | -1.00 | -1.00 | -1.00 | -1.00 | -1.00 | -1.00 |
| -1.00 | 1.00 | -1.00 | -1.00 | -1.00 | -1.00 | -1.00 | 1.00 | -1.00 |
| -1.00 | -1.00 | 1.00 | -1.00 | -1.00 | -1.00 | 1.00 | -1.00 | -1.00 |
| -1.00 | -1.00 | -1.00 | 1.00 | -1.00 | 1.00 | -1.00 | -1.00 | -1.00 |
| -1.00 | -1.00 | -1.00 | -1.00 | 1.00 | -1.00 | -1.00 | -1.00 | -1.00 |
| -1.00 | -1.00 | -1.00 | 1.00 | -1.00 | 1.00 | -1.00 | -1.00 | -1.00 |
| -1.00 | -1.00 | 1.00 | -1.00 | -1.00 | -1.00 | 1.00 | -1.00 | -1.00 |
| -1.00 | 1.00 | -1.00 | -1.00 | -1.00 | -1.00 | -1.00 | 1.00 | -1.00 |
| -1.00 | -1.00 | -1.00 | -1.00 | -1.00 | -1.00 | -1.00 | -1.00 | -1.00 |

Convolution

**Image Filtrée**

| | | | | | | |
|---|---|---|---|---|---|---|
| 0.78 | -0.11 | 0.11 | 0.33 | 0.56 | -0.11 | 0.33 |
| -0.11 | 1.00 | -0.11 | 0.33 | -0.11 | 0.11 | -0.11 |
| 0.11 | -0.11 | 1.00 | -0.33 | 0.11 | -0.11 | 0.56 |
| 0.33 | 0.33 | -0.33 | 0.56 | -0.33 | 0.33 | 0.33 |
| 0.56 | -0.11 | 0.11 | -0.33 | 1.00 | -0.11 | 0.11 |
| -0.11 | 0.11 | -0.11 | 0.33 | -0.11 | 1.00 | -0.11 |
| 0.33 | -0.11 | 0.56 | 0.33 | 0.11 | -0.11 | 0.78 |

**Kernel size : 3 x 3**

**Stride : 1**

**Kernel**

| | | |
|---|---|---|
| 1.00 | -1.00 | -1.00 |
| -1.00 | 1.00 | -1.00 |
| -1.00 | -1.00 | 1.00 |

( (1)*(-1) + (-1)*(-1) + (-1)*(-1)+
(-1)*(-1) + (1)*(1) + (-1)*(-1)+
(-1)*(-1) + (-1)*(-1) + (1)*(1) ) /9 => **-0.11**

**Image Originale**

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| -1.00 | -1.00 | -1.00 | -1.00 | -1.00 | -1.00 | -1.00 | -1.00 | -1.00 |
| -1.00 | 1.00 | -1.00 | -1.00 | -1.00 | -1.00 | -1.00 | 1.00 | -1.00 |
| -1.00 | -1.00 | 1.00 | -1.00 | -1.00 | -1.00 | 1.00 | -1.00 | -1.00 |
| -1.00 | -1.00 | -1.00 | 1.00 | -1.00 | 1.00 | -1.00 | -1.00 | -1.00 |
| -1.00 | -1.00 | -1.00 | -1.00 | 1.00 | -1.00 | -1.00 | -1.00 | -1.00 |
| -1.00 | -1.00 | -1.00 | 1.00 | -1.00 | 1.00 | -1.00 | -1.00 | -1.00 |
| -1.00 | -1.00 | 1.00 | -1.00 | -1.00 | -1.00 | 1.00 | -1.00 | -1.00 |
| -1.00 | 1.00 | -1.00 | -1.00 | -1.00 | -1.00 | -1.00 | 1.00 | -1.00 |
| -1.00 | -1.00 | -1.00 | -1.00 | -1.00 | -1.00 | -1.00 | -1.00 | -1.00 |

Convolution

**Image Filtrée**

| | | | | | | |
|---|---|---|---|---|---|---|
| 0.78 | -0.11 | 0.11 | 0.33 | 0.56 | -0.11 | 0.33 |
| -0.11 | 1.00 | -0.11 | 0.33 | -0.11 | 0.11 | -0.11 |
| 0.11 | -0.11 | 1.00 | -0.33 | 0.11 | -0.11 | 0.56 |
| 0.33 | 0.33 | -0.33 | 0.56 | -0.33 | 0.33 | 0.33 |
| 0.56 | -0.11 | 0.11 | -0.33 | 1.00 | -0.11 | 0.11 |
| -0.11 | 0.11 | -0.11 | 0.33 | -0.11 | 1.00 | -0.11 |
| 0.33 | -0.11 | 0.56 | 0.33 | 0.11 | -0.11 | 0.78 |

**Kernel size : 3 x 3**

**Stride : 1**

**Kernel**

| | | |
|---|---|---|
| 1.00 | -1.00 | -1.00 |
| -1.00 | 1.00 | -1.00 |
| -1.00 | -1.00 | 1.00 |

( (1)*(-1) + (-1)*(-1) + (-1)*(-1)+
(-1)*(-1) + (1)*(1) + (-1)*(-1)+
(-1)*(-1) + (-1)*(-1) + (1)*(1) ) /9 => **0.78**

**Image Originale**

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| -1.00 | -1.00 | -1.00 | -1.00 | -1.00 | -1.00 | -1.00 | -1.00 | -1.00 |
| -1.00 | 1.00 | -1.00 | -1.00 | -1.00 | -1.00 | -1.00 | 1.00 | -1.00 |
| -1.00 | -1.00 | 1.00 | -1.00 | -1.00 | -1.00 | 1.00 | -1.00 | -1.00 |
| -1.00 | -1.00 | -1.00 | 1.00 | -1.00 | 1.00 | -1.00 | -1.00 | -1.00 |
| -1.00 | -1.00 | -1.00 | -1.00 | 1.00 | -1.00 | -1.00 | -1.00 | -1.00 |
| -1.00 | -1.00 | -1.00 | 1.00 | -1.00 | 1.00 | -1.00 | -1.00 | -1.00 |
| -1.00 | -1.00 | 1.00 | -1.00 | -1.00 | -1.00 | 1.00 | -1.00 | -1.00 |
| -1.00 | 1.00 | -1.00 | -1.00 | -1.00 | -1.00 | -1.00 | 1.00 | -1.00 |
| -1.00 | -1.00 | -1.00 | -1.00 | -1.00 | -1.00 | -1.00 | -1.00 | -1.00 |

Convolution

**Image Filtrée**

| | | | | | | |
|---|---|---|---|---|---|---|
| 0.78 | -0.11 | 0.11 | 0.33 | 0.56 | -0.11 | 0.33 |
| -0.11 | 1.00 | -0.11 | 0.33 | -0.11 | 0.11 | -0.11 |
| 0.11 | -0.11 | 1.00 | -0.33 | 0.11 | -0.11 | 0.56 |
| 0.33 | 0.33 | -0.33 | 0.56 | -0.33 | 0.33 | 0.33 |
| 0.56 | -0.11 | 0.11 | -0.33 | 1.00 | -0.11 | 0.11 |
| -0.11 | 0.11 | -0.11 | 0.33 | -0.11 | 1.00 | -0.11 |
| 0.33 | -0.11 | 0.56 | 0.33 | 0.11 | -0.11 | 0.78 |

**Kernel size : 3 x 3**

**Stride : 1**

**Kernel**

| | | |
|---|---|---|
| 1.00 | -1.00 | -1.00 |
| -1.00 | 1.00 | -1.00 |
| -1.00 | -1.00 | 1.00 |

( (1)*(-1) + (-1)*(-1) + (-1)*(-1)+
(-1)*(-1) + (1)*(1) + (-1)*(-1)+
(-1)*(-1) + (-1)*(-1) + (1)*(1) ) /9 => **-0.11**

**Image Originale**

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| -1.00 | -1.00 | -1.00 | -1.00 | -1.00 | -1.00 | -1.00 | -1.00 | -1.00 |
| -1.00 | 1.00 | -1.00 | -1.00 | -1.00 | -1.00 | -1.00 | 1.00 | -1.00 |
| -1.00 | -1.00 | 1.00 | -1.00 | -1.00 | -1.00 | 1.00 | -1.00 | -1.00 |
| -1.00 | -1.00 | -1.00 | 1.00 | -1.00 | 1.00 | -1.00 | -1.00 | -1.00 |
| -1.00 | -1.00 | -1.00 | -1.00 | 1.00 | -1.00 | -1.00 | -1.00 | -1.00 |
| -1.00 | -1.00 | -1.00 | 1.00 | -1.00 | 1.00 | -1.00 | -1.00 | -1.00 |
| -1.00 | -1.00 | 1.00 | -1.00 | -1.00 | -1.00 | 1.00 | -1.00 | -1.00 |
| -1.00 | 1.00 | -1.00 | -1.00 | -1.00 | -1.00 | -1.00 | 1.00 | -1.00 |
| -1.00 | -1.00 | -1.00 | -1.00 | -1.00 | -1.00 | -1.00 | -1.00 | -1.00 |

Convolution

**Image Filtrée**

| | | | | | | |
|---|---|---|---|---|---|---|
| 0.78 | -0.11 | 0.11 | 0.33 | 0.56 | -0.11 | 0.33 |
| -0.11 | 1.00 | -0.11 | 0.33 | -0.11 | 0.11 | -0.11 |
| 0.11 | -0.11 | 1.00 | -0.33 | 0.11 | -0.11 | 0.56 |
| 0.33 | 0.33 | -0.33 | 0.56 | -0.33 | 0.33 | 0.33 |
| 0.56 | -0.11 | 0.11 | -0.33 | 1.00 | -0.11 | 0.11 |
| -0.11 | 0.11 | -0.11 | 0.33 | -0.11 | 1.00 | -0.11 |
| 0.33 | -0.11 | 0.56 | 0.33 | 0.11 | -0.11 | 0.78 |

**Kernel size : 3 x 3**

**Stride : 1**

**Kernel**

| | | |
|---|---|---|
| 1.00 | -1.00 | -1.00 |
| -1.00 | 1.00 | -1.00 |
| -1.00 | -1.00 | 1.00 |

( (1)*(-1) + (-1)*(-1) + (-1)*(-1)+
(-1)*(-1) + (1)*(1) + (-1)*(-1)+
(-1)*(-1) + (-1)*(-1) + (1)*(1) ) /9 => **0.33**

**Image Originale**

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| -1.00 | -1.00 | -1.00 | -1.00 | -1.00 | -1.00 | -1.00 | -1.00 | -1.00 |
| -1.00 | 1.00 | -1.00 | -1.00 | -1.00 | -1.00 | -1.00 | 1.00 | -1.00 |
| -1.00 | -1.00 | 1.00 | -1.00 | -1.00 | -1.00 | 1.00 | -1.00 | -1.00 |
| -1.00 | -1.00 | -1.00 | 1.00 | -1.00 | 1.00 | -1.00 | -1.00 | -1.00 |
| -1.00 | -1.00 | -1.00 | -1.00 | 1.00 | -1.00 | -1.00 | -1.00 | -1.00 |
| -1.00 | -1.00 | -1.00 | 1.00 | -1.00 | 1.00 | -1.00 | -1.00 | -1.00 |
| -1.00 | -1.00 | 1.00 | -1.00 | -1.00 | -1.00 | 1.00 | -1.00 | -1.00 |
| -1.00 | 1.00 | -1.00 | -1.00 | -1.00 | -1.00 | -1.00 | 1.00 | -1.00 |
| -1.00 | -1.00 | -1.00 | -1.00 | -1.00 | -1.00 | -1.00 | -1.00 | -1.00 |

Convolution

**Image Filtrée**

| | | | | | | |
|---|---|---|---|---|---|---|
| 0.78 | -0.11 | 0.11 | 0.33 | 0.56 | -0.11 | 0.33 |
| -0.11 | 1.00 | -0.11 | 0.33 | -0.11 | 0.11 | -0.11 |
| 0.11 | -0.11 | 1.00 | -0.33 | 0.11 | -0.11 | 0.56 |
| 0.33 | 0.33 | -0.33 | 0.56 | -0.33 | 0.33 | 0.33 |
| 0.56 | -0.11 | 0.11 | -0.33 | 1.00 | -0.11 | 0.11 |
| -0.11 | 0.11 | -0.11 | 0.33 | -0.11 | 1.00 | -0.11 |
| 0.33 | -0.11 | 0.56 | 0.33 | 0.11 | -0.11 | 0.78 |

**Kernel size : 3 x 3**

**Stride : 1**

**Kernel**

| | | |
|---|---|---|
| 1.00 | -1.00 | -1.00 |
| -1.00 | 1.00 | -1.00 |
| -1.00 | -1.00 | 1.00 |

( (1)*(-1) + (-1)*(-1) + (-1)*(-1)+
(-1)*(-1) + (1)*(1) + (-1)*(-1)+
(-1)*(-1) + (-1)*(-1) + (1)*(1) ) /9 => **-0.11**

**Image Originale**

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| -1.00 | -1.00 | -1.00 | -1.00 | -1.00 | -1.00 | -1.00 | -1.00 | -1.00 |
| -1.00 | 1.00 | -1.00 | -1.00 | -1.00 | -1.00 | 1.00 | 1.00 | -1.00 |
| -1.00 | -1.00 | 1.00 | -1.00 | -1.00 | 1.00 | -1.00 | -1.00 | -1.00 |
| -1.00 | -1.00 | -1.00 | 1.00 | -1.00 | 1.00 | -1.00 | -1.00 | -1.00 |
| -1.00 | -1.00 | -1.00 | -1.00 | 1.00 | -1.00 | -1.00 | -1.00 | -1.00 |
| -1.00 | -1.00 | -1.00 | 1.00 | -1.00 | 1.00 | -1.00 | -1.00 | -1.00 |
| -1.00 | -1.00 | 1.00 | -1.00 | -1.00 | -1.00 | 1.00 | -1.00 | -1.00 |
| -1.00 | 1.00 | -1.00 | -1.00 | -1.00 | -1.00 | -1.00 | 1.00 | -1.00 |
| -1.00 | -1.00 | -1.00 | -1.00 | -1.00 | -1.00 | -1.00 | -1.00 | -1.00 |

Convolution

**Image Filtrée**

| | | | | | | |
|---|---|---|---|---|---|---|
| 0.78 | -0.11 | 0.11 | 0.33 | 0.56 | -0.11 | 0.33 |
| -0.11 | 1.00 | -0.11 | 0.33 | -0.11 | 0.11 | -0.11 |
| 0.11 | -0.11 | 1.00 | -0.33 | 0.11 | -0.11 | 0.56 |
| 0.33 | 0.33 | -0.33 | 0.56 | -0.33 | 0.33 | 0.33 |
| 0.56 | -0.11 | 0.11 | -0.33 | 1.00 | -0.11 | 0.11 |
| -0.11 | 0.11 | -0.11 | 0.33 | -0.11 | 1.00 | -0.11 |
| 0.33 | -0.11 | 0.56 | 0.33 | 0.11 | -0.11 | 0.78 |

**Kernel size : 3 x 3**

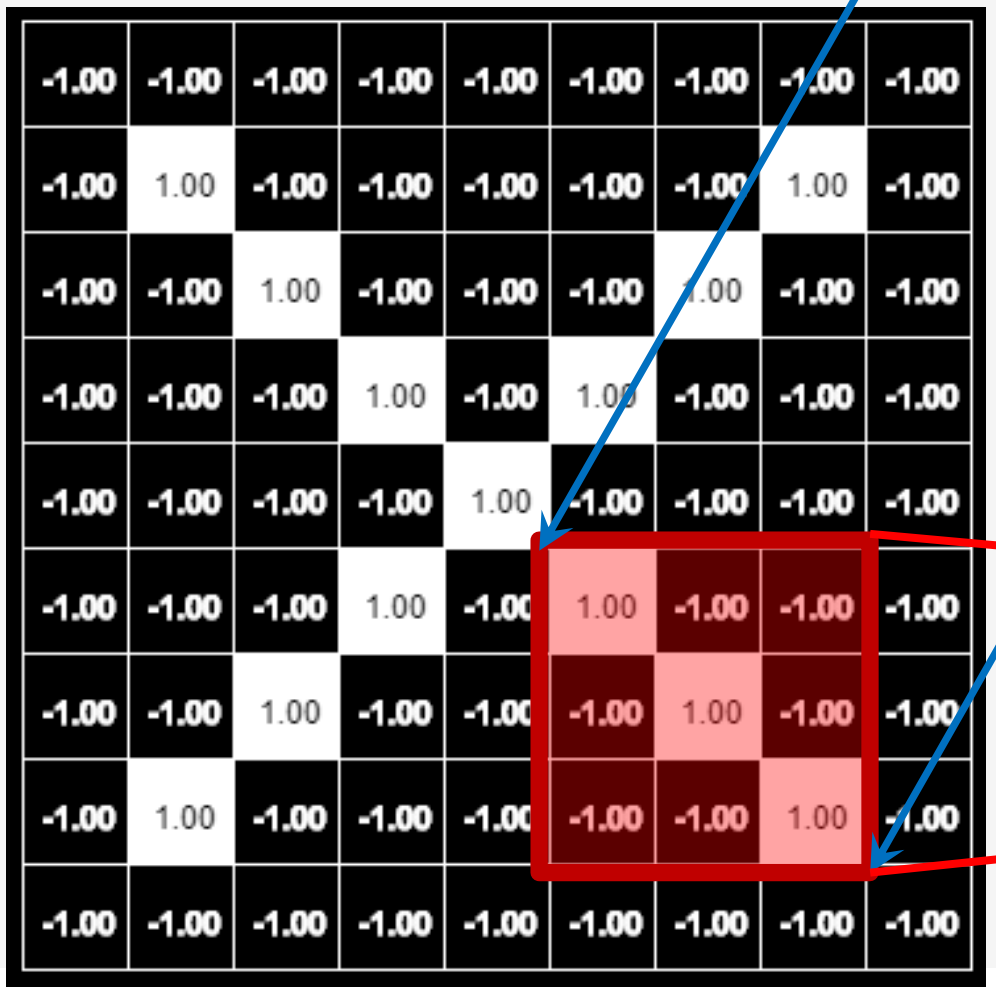
**Stride : 1**

**Kernel**

| | | |
|---|---|---|
| 1.00 | -1.00 | -1.00 |
| -1.00 | 1.00 | -1.00 |
| -1.00 | -1.00 | 1.00 |

( (1)*(-1) + (-1)*(-1) + (-1)*(-1)+
(-1)*(-1) + (1)*(1) + (-1)*(-1)+
(-1)*(-1) + (-1)*(-1) + (1)*(1) ) /9 => 1

**Image Originale**




Convolution

**Image Filtrée**

**Kernel size : 3 x 3**

**Stride : 1**

**Kernel**

| | | |
|---|---|---|
| 1.00 | -1.00 | -1.00 |
| -1.00 | 1.00 | -1.00 |
| -1.00 | -1.00 | 1.00 |

( (1)*(-1) + (-1)*(-1) + (-1)*(-1)+
(-1)*(-1) + (1)*(1) + (-1)*(-1)+
(-1)*(-1) + (-1)*(-1) + (1)*(1) ) /9 => **-0.11**

**Image Originale**

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| -1.00 | -1.00 | -1.00 | -1.00 | -1.00 | -1.00 | -1.00 | -1.00 | -1.00 |
| -1.00 | 1.00 | -1.00 | -1.00 | -1.00 | -1.00 | -1.00 | 1.00 | -1.00 |
| -1.00 | -1.00 | 1.00 | -1.00 | -1.00 | -1.00 | 1.00 | -1.00 | -1.00 |
| -1.00 | -1.00 | -1.00 | 1.00 | -1.00 | 1.00 | -1.00 | -1.00 | -1.00 |
| -1.00 | -1.00 | -1.00 | -1.00 | 1.00 | -1.00 | -1.00 | -1.00 | -1.00 |
| -1.00 | -1.00 | -1.00 | 1.00 | -1.00 | 1.00 | -1.00 | -1.00 | -1.00 |
| -1.00 | -1.00 | 1.00 | -1.00 | -1.00 | -1.00 | 1.00 | -1.00 | -1.00 |
| -1.00 | 1.00 | -1.00 | -1.00 | -1.00 | -1.00 | -1.00 | 1.00 | -1.00 |
| -1.00 | -1.00 | -1.00 | -1.00 | -1.00 | -1.00 | -1.00 | -1.00 | -1.00 |

Convolution

**Image Filtrée**

| | | | | | | |
|---|---|---|---|---|---|---|
| 0.78 | -0.11 | 0.11 | 0.33 | 0.56 | -0.11 | 0.33 |
| -0.11 | 1.00 | -0.11 | 0.33 | -0.11 | 0.11 | -0.11 |
| 0.11 | -0.11 | 1.00 | -0.33 | 0.11 | -0.11 | 0.56 |
| 0.33 | 0.33 | -0.33 | 0.56 | -0.33 | 0.33 | 0.33 |
| [illegible] | [illegible] | 0.11 | -0.33 | 1.00 | -0.11 | 0.11 |
| -0.11 | 0.11 | -0.11 | 0.33 | -0.11 | 1.00 | -0.11 |
| [illegible] | [illegible] | 0.56 | 0.33 | 0.11 | -0.11 | 0.78 |

**Kernel size : 3 x 3**

**Stride : 1**

**Kernel**

| | | |
|---|---|---|
| 1.00 | -1.00 | -1.00 |
| -1.00 | 1.00 | -1.00 |
| -1.00 | -1.00 | 1.00 |

( (1)*(-1) + (-1)*(-1) + (-1)*(-1)+
(-1)*(-1) + (1)*(1) + (-1)*(-1)+
(-1)*(-1) + (-1)*(-1) + (1)*(1) ) /9 => **33**

**Image Originale**

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| -1.00 | -1.00 | -1.00 | -1.00 | -1.00 | -1.00 | -1.00 | -1.00 | -1.00 |
| -1.00 | 1.00 | -1.00 | -1.00 | -1.00 | -1.00 | -1.00 | 1.00 | -1.00 |
| -1.00 | -1.00 | 1.00 | -1.00 | -1.00 | -1.00 | 1.00 | -1.00 | -1.00 |
| -1.00 | -1.00 | -1.00 | 1.00 | -1.00 | 1.00 | -1.00 | -1.00 | -1.00 |
| -1.00 | -1.00 | -1.00 | -1.00 | 1.00 | -1.00 | -1.00 | -1.00 | -1.00 |
| -1.00 | -1.00 | -1.00 | 1.00 | -1.00 | 1.00 | -1.00 | -1.00 | -1.00 |
| -1.00 | -1.00 | 1.00 | -1.00 | -1.00 | -1.00 | 1.00 | -1.00 | -1.00 |
| -1.00 | 1.00 | -1.00 | -1.00 | -1.00 | -1.00 | -1.00 | 1.00 | -1.00 |
| -1.00 | -1.00 | -1.00 | -1.00 | -1.00 | -1.00 | -1.00 | -1.00 | -1.00 |

Convolution

**Image Filtrée**

| | | | | | | |
|---|---|---|---|---|---|---|
| 0.78 | -0.11 | 0.11 | 0.33 | 0.56 | -0.11 | 0.33 |
| -0.11 | 1.00 | -0.11 | 0.33 | -0.11 | 0.11 | -0.11 |
| 0.11 | -0.11 | 1.00 | -0.33 | 0.11 | -0.11 | 0.56 |
| 0.33 | 0.33 | -0.33 | 0.56 | -0.33 | 0.33 | 0.33 |
| 0.56 | -0.11 | 0.11 | -0.33 | 1.00 | -0.11 | 0.11 |
| -0.11 | 0.11 | -0.11 | 0.33 | -0.11 | 1.00 | -0.11 |
| 0.33 | -0.11 | 0.56 | 0.33 | 0.11 | -0.11 | 0.78 |

**Kernel size : 3 x 3**

**Stride : 1**

**Kernel**

| | | |
|---|---|---|
| 1.00 | -1.00 | -1.00 |
| -1.00 | 1.00 | -1.00 |
| -1.00 | -1.00 | 1.00 |

( (1)*(-1) + (-1)*(-1) + (-1)*(-1)+
(-1)*(-1) + (1)*(1) + (-1)*(-1)+
(-1)*(-1) + (-1)*(-1) + (1)*(1) ) /9 => -0.11

**Image Originale**

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| -1.00 | -1.00 | -1.00 | -1.00 | -1.00 | -1.00 | -1.00 | -1.00 | -1.00 |
| -1.00 | 1.00 | -1.00 | -1.00 | -1.00 | -1.00 | -1.00 | 1.00 | -1.00 |
| -1.00 | -1.00 | 1.00 | -1.00 | -1.00 | -1.00 | 1.00 | -1.00 | -1.00 |
| -1.00 | -1.00 | -1.00 | 1.00 | -1.00 | 1.00 | -1.00 | -1.00 | -1.00 |
| -1.00 | -1.00 | -1.00 | -1.00 | 1.00 | -1.00 | -1.00 | -1.00 | -1.00 |
| -1.00 | -1.00 | -1.00 | 1.00 | -1.00 | 1.00 | -1.00 | -1.00 | -1.00 |
| -1.00 | -1.00 | 1.00 | -1.00 | -1.00 | -1.00 | -1.00 | -1.00 | -1.00 |
| -1.00 | 1.00 | -1.00 | -1.00 | -1.00 | -1.00 | -1.00 | 1.00 | -1.00 |
| -1.00 | -1.00 | -1.00 | -1.00 | -1.00 | -1.00 | -1.00 | -1.00 | -1.00 |

Convolution

**Image Filtrée**

| | | | | | | |
|---|---|---|---|---|---|---|
| 0.78 | -0.11 | 0.11 | 0.33 | 0.56 | -0.11 | 0.33 |
| -0.11 | 1.00 | -0.11 | 0.33 | -0.11 | 0.11 | -0.11 |
| 0.11 | -0.11 | 1.00 | -0.33 | 0.11 | -0.11 | 0.56 |
| 0.33 | 0.33 | -0.33 | 0.56 | -0.33 | 0.33 | 0.33 |
| 0.56 | -0.11 | 0.11 | -0.33 | 1.00 | -0.11 | 0.11 |
| -0.11 | 0.11 | -0.11 | 0.33 | -0.11 | 1.00 | -0.11 |
| 0.33 | -0.11 | 0.56 | 0.33 | 0.11 | -0.11 | 0.78 |

**Kernel size : 3 x 3**

**Stride : 1**

**Kernel**

| | | |
|---|---|---|
| 1.00 | -1.00 | -1.00 |
| -1.00 | 1.00 | -1.00 |
| -1.00 | -1.00 | 1.00 |

( (1)*(-1) + (-1)*(-1) + (-1)*(-1)+
(-1)*(-1) + (1)*(1) + (-1)*(-1)+
(-1)*(-1) + (-1)*(-1) + (1)*(1) ) /9 => **0.56**

**Image Originale**

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| -1.00 | -1.00 | -1.00 | -1.00 | -1.00 | -1.00 | -1.00 | -1.00 | -1.00 |
| -1.00 | 1.00 | -1.00 | -1.00 | -1.00 | -1.00 | -1.00 | 1.00 | -1.00 |
| -1.00 | -1.00 | 1.00 | -1.00 | -1.00 | -1.00 | 1.00 | -1.00 | -1.00 |
| -1.00 | -1.00 | -1.00 | 1.00 | -1.00 | 1.00 | -1.00 | -1.00 | -1.00 |
| -1.00 | -1.00 | -1.00 | -1.00 | 1.00 | -1.00 | -1.00 | -1.00 | -1.00 |
| -1.00 | -1.00 | -1.00 | 1.00 | -1.00 | 1.00 | -1.00 | -1.00 | -1.00 |
| -1.00 | -1.00 | 1.00 | -1.00 | -1.00 | -1.00 | 1.00 | -1.00 | -1.00 |
| -1.00 | 1.00 | -1.00 | -1.00 | -1.00 | -1.00 | -1.00 | 1.00 | -1.00 |
| -1.00 | -1.00 | -1.00 | -1.00 | -1.00 | -1.00 | -1.00 | -1.00 | -1.00 |

Convolution

**Image Filtrée**

| | | | | | | |
|---|---|---|---|---|---|---|
| 0.78 | -0.11 | 0.11 | 0.33 | 0.56 | -0.11 | 0.33 |
| -0.11 | 1.00 | -0.11 | 0.33 | -0.11 | 0.11 | -0.11 |
| 0.11 | -0.11 | 1.00 | -0.33 | 0.11 | -0.11 | 0.56 |
| 0.33 | 0.33 | -0.33 | 0.56 | -0.33 | 0.33 | 0.33 |
| 0.56 | -0.11 | 0.11 | -0.33 | 1.00 | -0.11 | 0.11 |
| -0.11 | 0.11 | -0.11 | 0.33 | -0.11 | 1.00 | -0.11 |
| 0.33 | -0.11 | 0.56 | 0.33 | 0.11 | -0.11 | 0.78 |

**Kernel size : 3 x 3**

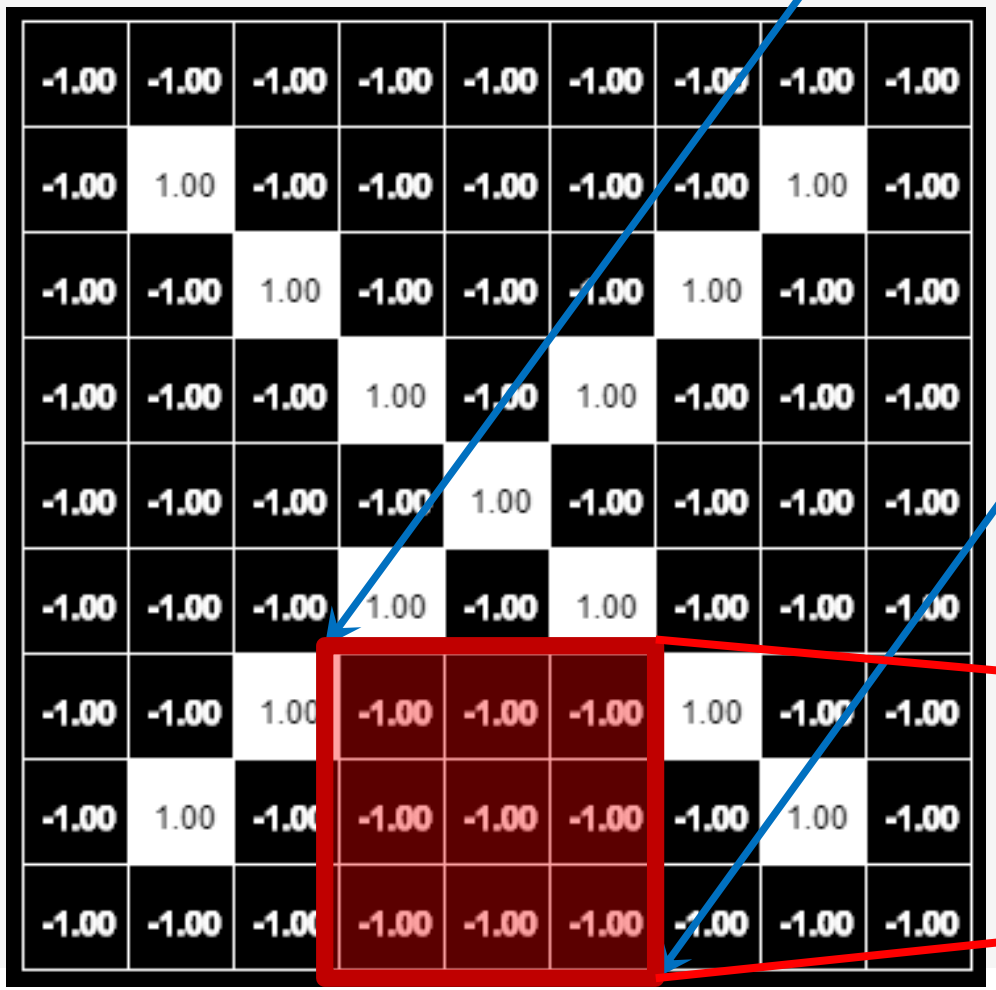
**Stride : 1**

**Kernel**

| | | |
|---|---|---|
| 1.00 | -1.00 | -1.00 |
| -1.00 | 1.00 | -1.00 |
| -1.00 | -1.00 | 1.00 |

( (1)*(-1) + (-1)*(-1) + (-1)*(-1)+
(-1)*(-1) + (1)*(1) + (-1)*(-1)+
(-1)*(-1) + (-1)*(-1) + (1)*(1) ) /9 => **0.33**

**Image Originale**



**Convolution**

**Image Filtrée**

| | | | | | | |
|---|---|---|---|---|---|---|
| 0.78 | -0.11 | 0.11 | 0.33 | 0.56 | -0.11 | 0.33 |
| -0.11 | 1.00 | -0.11 | 0.33 | -0.11 | 0.11 | -0.11 |
| 0.11 | -0.11 | 1.00 | -0.33 | 0.11 | -0.11 | 0.56 |
| 0.33 | 0.33 | -0.33 | 0.56 | -0.33 | 0.33 | 0.33 |
| 0.56 | -0.11 | 0.11 | -0.33 | 1.00 | -0.11 | 0.11 |
| -0.11 | 0.11 | -0.11 | 0.33 | -0.11 | 1.00 | -0.11 |
| 0.33 | -0.11 | 0.56 | 0.33 | 0.11 | -0.11 | 0.78 |

**Kernel size : 3 x 3**

**Stride : 1**

**Kernel**

| | | |
|---|---|---|
| 1.00 | -1.00 | -1.00 |
| -1.00 | 1.00 | -1.00 |
| -1.00 | -1.00 | 1.00 |

( (1)*(-1) + (-1)*(-1) + (-1)*(-1)+
(-1)*(-1) + (1)*(1) + (-1)*(-1)+
(-1)*(-1) + (-1)*(-1) + (1)*(1) ) /9 => **0.11**

**Image Originale**

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| -1.00 | -1.00 | -1.00 | -1.00 | -1.00 | -1.00 | -1.00 | -1.00 | -1.00 |
| -1.00 | 1.00 | -1.00 | -1.00 | -1.00 | -1.00 | -1.00 | 1.00 | -1.00 |
| -1.00 | -1.00 | 1.00 | -1.00 | -1.00 | -1.00 | 1.00 | -1.00 | -1.00 |
| -1.00 | -1.00 | -1.00 | 1.00 | -1.00 | 1.00 | -1.00 | -1.00 | -1.00 |
| -1.00 | -1.00 | -1.00 | -1.00 | 1.00 | -1.00 | -1.00 | -1.00 | -1.00 |
| -1.00 | -1.00 | -1.00 | 1.00 | -1.00 | 1.00 | -1.00 | -1.00 | -1.00 |
| -1.00 | -1.00 | 1.00 | -1.00 | -1.00 | -1.00 | 1.00 | -1.00 | -1.00 |
| -1.00 | 1.00 | -1.00 | -1.00 | -1.00 | -1.00 | -1.00 | 1.00 | -1.00 |
| -1.00 | -1.00 | -1.00 | -1.00 | -1.00 | -1.00 | -1.00 | -1.00 | -1.00 |

Convolution

**Image Filtrée**

| | | | | | | |
|---|---|---|---|---|---|---|
| 0.78 | -0.11 | 0.11 | 0.33 | 0.56 | -0.11 | 0.33 |
| -0.11 | 1.00 | -0.11 | 0.33 | -0.11 | 0.11 | -0.11 |
| 0.11 | -0.11 | 1.00 | -0.33 | 0.11 | -0.11 | 0.56 |
| 0.33 | 0.33 | -0.33 | 0.56 | -0.33 | 0.33 | 0.33 |
| 0.56 | -0.11 | 0.11 | -0.33 | 1.00 | -0.11 | 0.11 |
| -0.11 | 0.11 | -0.11 | 0.33 | -0.11 | 1.00 | -0.11 |
| 0.33 | -0.11 | 0.56 | 0.33 | 0.11 | -0.11 | 0.78 |

**Kernel size : 3 x 3**

**Stride : 1**

**Kernel**

| | | |
|---|---|---|
| 1.00 | -1.00 | -1.00 |
| -1.00 | 1.00 | -1.00 |
| -1.00 | -1.00 | 1.00 |

( (1)*(-1) + (-1)*(-1) + (-1)*(-1)+
(-1)*(-1) + (1)*(1) + (-1)*(-1)+
(-1)*(-1) + (-1)*(-1) + (1)*(1) ) /9 => -0.11

**Image Originale**

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| -1.00 | -1.00 | -1.00 | -1.00 | -1.00 | -1.00 | -1.00 | -1.00 | -1.00 |
| -1.00 | 1.00 | -1.00 | -1.00 | -1.00 | -1.00 | -1.00 | 1.00 | -1.00 |
| -1.00 | -1.00 | 1.00 | -1.00 | -1.00 | -1.00 | 1.00 | -1.00 | -1.00 |
| -1.00 | -1.00 | -1.00 | 1.00 | -1.00 | 1.00 | -1.00 | -1.00 | -1.00 |
| -1.00 | -1.00 | -1.00 | -1.00 | 1.00 | -1.00 | -1.00 | -1.00 | -1.00 |
| -1.00 | -1.00 | -1.00 | 1.00 | -1.00 | 1.00 | -1.00 | -1.00 | -1.00 |
| -1.00 | -1.00 | 1.00 | -1.00 | -1.00 | -1.00 | 1.00 | -1.00 | -1.00 |
| -1.00 | 1.00 | -1.00 | -1.00 | -1.00 | -1.00 | -1.00 | 1.00 | -1.00 |
| -1.00 | -1.00 | -1.00 | -1.00 | -1.00 | -1.00 | -1.00 | -1.00 | -1.00 |

Convolution

**Image Filtrée**

| | | | | | | |
|---|---|---|---|---|---|---|
| 0.78 | -0.11 | 0.11 | 0.33 | 0.56 | -0.11 | 0.33 |
| -0.11 | 1.00 | -0.11 | 0.33 | -0.11 | 0.11 | -0.11 |
| 0.11 | -0.11 | 1.00 | -0.33 | 0.11 | -0.11 | 0.56 |
| 0.33 | 0.33 | -0.33 | 0.56 | -0.33 | 0.33 | 0.33 |
| 0.56 | -0.11 | 0.11 | -0.33 | 1.00 | -0.11 | 0.11 |
| -0.11 | 0.11 | -0.11 | 0.33 | -0.11 | 1.00 | -0.11 |
| 0.33 | -0.11 | 0.56 | 0.33 | 0.11 | -0.11 | 0.78 |

**Kernel size : 3 x 3**

**Stride : 1**

**Kernel**

| | | |
|---|---|---|
| 1.00 | -1.00 | -1.00 |
| -1.00 | 1.00 | -1.00 |
| -1.00 | -1.00 | 1.00 |

( (1)*(-1) + (-1)*(-1) + (-1)*(-1)+
(-1)*(-1) + (1)*(1) + (-1)*(-1)+
(-1)*(-1) + (-1)*(-1) + (1)*(1) ) /9 => **0.78**

**Image Originale**

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| -1.00 | -1.00 | -1.00 | -1.00 | -1.00 | -1.00 | -1.00 | -1.00 | -1.00 |
| -1.00 | 1.00 | -1.00 | -1.00 | -1.00 | -1.00 | -1.00 | 1.00 | -1.00 |
| -1.00 | -1.00 | 1.00 | -1.00 | -1.00 | -1.00 | 1.00 | -1.00 | -1.00 |
| -1.00 | -1.00 | -1.00 | 1.00 | -1.00 | 1.00 | -1.00 | -1.00 | -1.00 |
| -1.00 | -1.00 | -1.00 | -1.00 | 1.00 | -1.00 | -1.00 | -1.00 | -1.00 |
| -1.00 | -1.00 | -1.00 | 1.00 | -1.00 | 1.00 | -1.00 | -1.00 | -1.00 |
| -1.00 | -1.00 | 1.00 | -1.00 | -1.00 | -1.00 | 1.00 | -1.00 | -1.00 |
| -1.00 | 1.00 | -1.00 | -1.00 | -1.00 | -1.00 | -1.00 | 1.00 | -1.00 |
| -1.00 | -1.00 | -1.00 | -1.00 | -1.00 | -1.00 | -1.00 | -1.00 | -1.00 |

Convolution

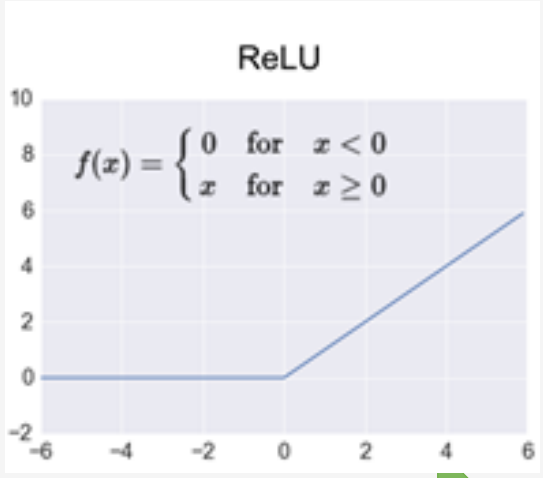
**Image Filtrée**

| | | | | | | |
|---|---|---|---|---|---|---|
| 0.78 | -0.11 | 0.11 | 0.33 | 0.56 | -0.11 | 0.33 |
| -0.11 | 1.00 | -0.11 | 0.33 | -0.11 | 0.11 | -0.11 |
| 0.11 | -0.11 | 1.00 | -0.33 | 0.11 | -0.11 | 0.56 |
| 0.33 | 0.33 | -0.33 | 0.56 | -0.33 | 0.33 | 0.33 |
| 0.56 | -0.11 | 0.11 | -0.33 | 1.00 | -0.11 | 0.11 |
| -0.11 | 0.11 | -0.11 | 0.33 | -0.11 | 1.00 | -0.11 |
| 0.33 | -0.11 | 0.56 | 0.33 | 0.11 | -0.11 | 0.78 |

# RELU Activation Function

**Image Filtrée**

| | | | | | | |
|---|---|---|---|---|---|---|
| 0.78 | -0.11 | 0.11 | 0.33 | 0.56 | -0.11 | 0.33 |
| -0.11 | 1.00 | -0.11 | 0.33 | -0.11 | 0.11 | -0.11 |
| 0.11 | -0.11 | 1.00 | -0.33 | 0.11 | -0.11 | 0.56 |
| 0.33 | 0.33 | -0.33 | 0.56 | -0.33 | 0.33 | 0.33 |
| 0.56 | -0.11 | 0.11 | -0.33 | 1.00 | -0.11 | 0.11 |
| -0.11 | 0.11 | -0.11 | 0.33 | -0.11 | 1.00 | -0.11 |
| 0.33 | -0.11 | 0.56 | 0.33 | 0.11 | -0.11 | 0.78 |




**RELU Activation**

**Image Filtrée avec RELU**

| | | | | | | |
|---|---|---|---|---|---|---|
| 0.78 | 0.00 | 0.11 | 0.33 | 0.56 | 0.00 | 0.33 |
| 0.00 | 1.00 | 0.00 | 0.33 | 0.00 | 0.11 | 0.00 |
| 0.11 | 0.00 | 1.00 | 0.00 | 0.11 | 0.00 | 0.56 |
| 0.33 | 0.33 | 0.00 | 0.56 | 0.00 | 0.33 | 0.33 |
| 0.56 | 0.00 | 0.11 | 0.00 | 1.00 | 0.00 | 0.11 |
| 0.00 | 0.11 | 0.00 | 0.33 | 0.00 | 1.00 | 0.00 |
| 0.33 | 0.00 | 0.56 | 0.33 | 0.11 | 0.00 | 0.78 |

# POOLING Layer (POOL MAX)

**Kernel size : 2 x 2**

**Stride : 2**

**Image Filtrée avec RELU**


POOL MAX

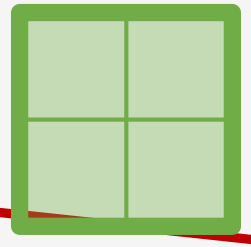
**Image Après Pool Max**

# POOLING (POOL MAX)

**Kernel size : 2 x 2**

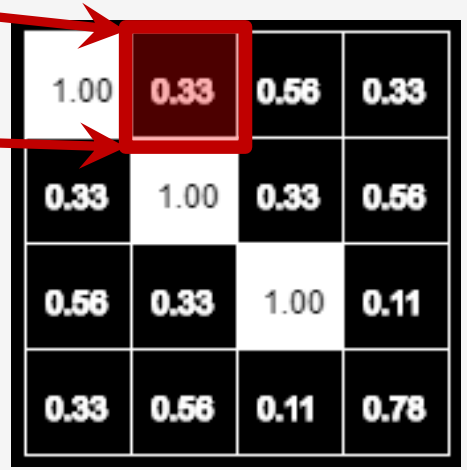
**Stride : 2**

**Image Filtrée avec RELU**

POOL MAX

**Image Après Pool Max**

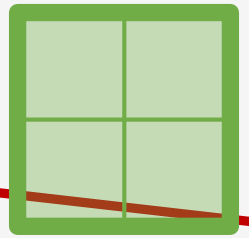


# POOLING (POOL MAX)

**Kernel size : 2 x 2**

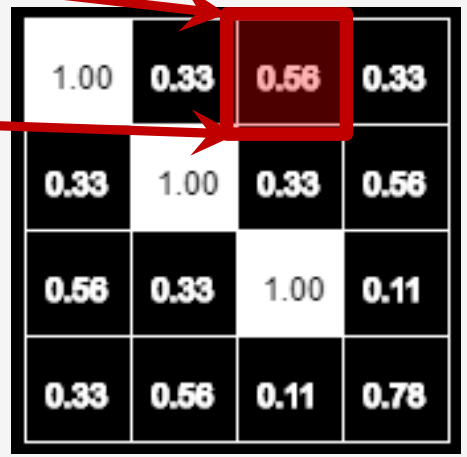
**Stride : 2**

**Image Filtrée avec RELU**

POOL MAX

**Image Après Pool Max**



# POOLING (POOL MAX)

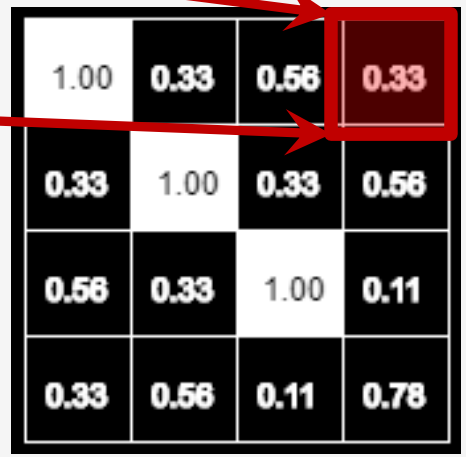
**Kernel size : 2 x 2**

**Stride : 2**

**Image Filtrée avec RELU**


POOL MAX

**Image Après Pool Max**

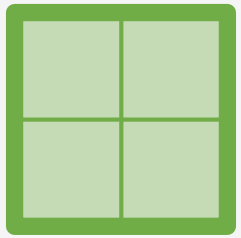


# POOLING (POOL MAX)

**Kernel size : 2 x 2**

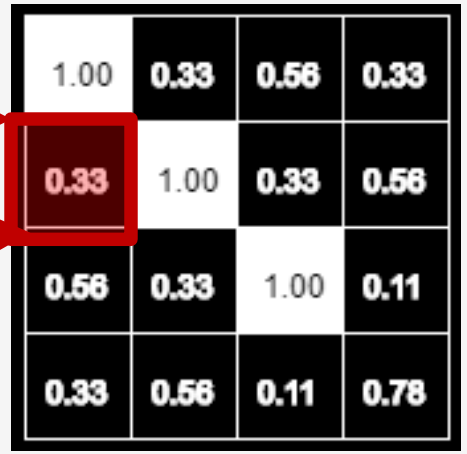
**Stride : 2**

**Image Filtrée avec RELU**

POOL MAX

**Image Après Pool Max**

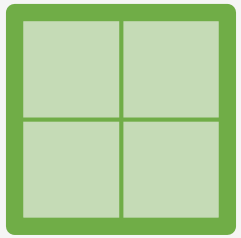


# POOLING (POOL MAX)

**Kernel size : 2 x 2**

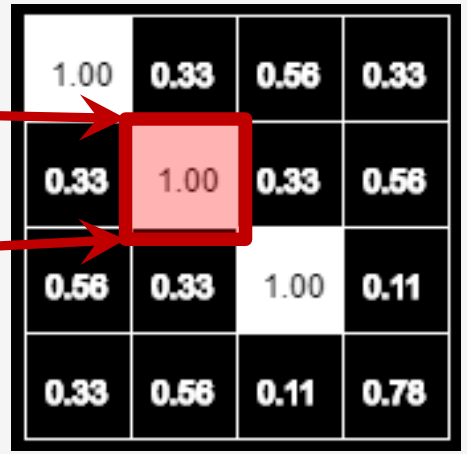
**Stride : 2**

**Image Filtrée avec RELU**

POOL MAX

**Image Après Pool Max**

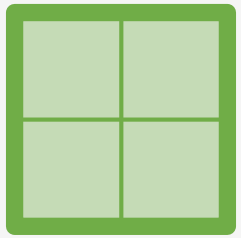


# POOLING (POOL MAX)

**Kernel size : 2 x 2**

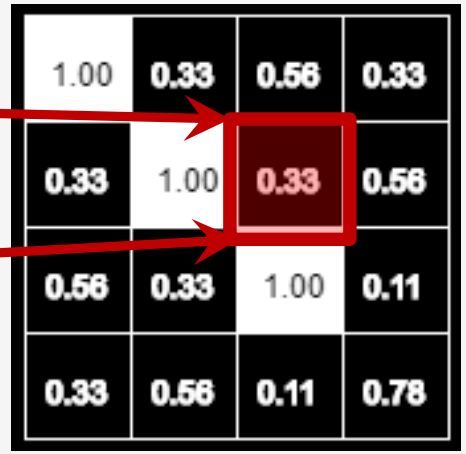
**Stride : 2**

**Image Filtrée avec RELU**

POOL MAX

**Image Après Pool Max**

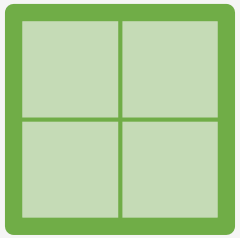


# POOLING (POOL MAX)

**Kernel size : 2 x 2**

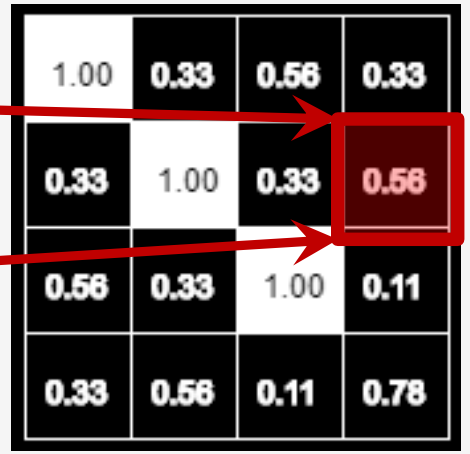
**Stride : 2**

**Image Filtrée avec RELU**

POOL MAX

**Image Après Pool Max**

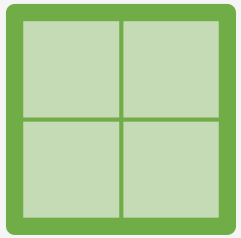


# POOLING (POOL MAX)

**Kernel size : 2 x 2**

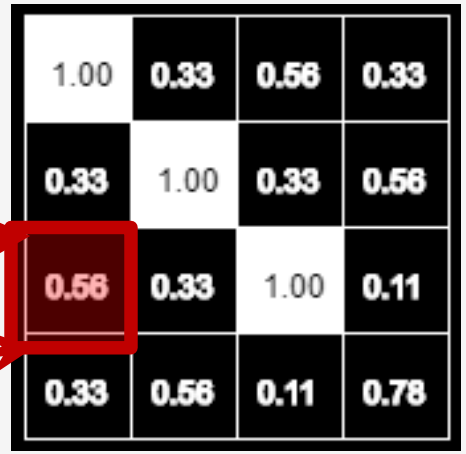
**Stride : 2**

**Image Filtrée avec RELU**

POOL MAX

**Image Après Pool Max**



# POOLING (POOL MAX)

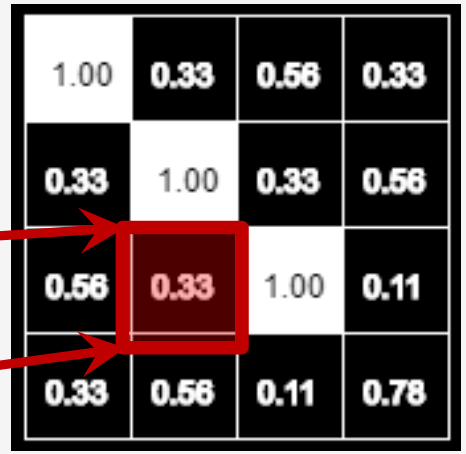
**Kernel size : 2 x 2**

**Stride : 2**

**Image Filtrée avec RELU**



**Image Après Pool Max**



# POOLING (POOL MAX)

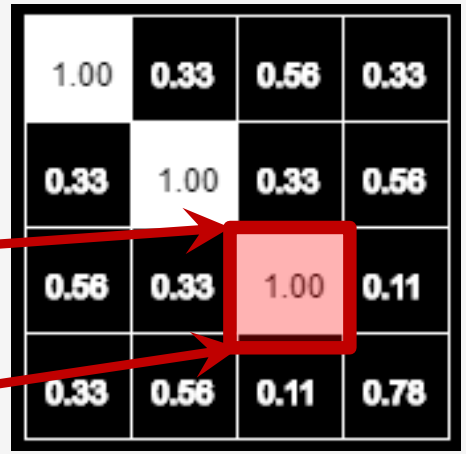
**Kernel size : 2 x 2**

**Stride : 2**

**Image Filtrée avec RELU**


POOL MAX

**Image Après Pool Max**

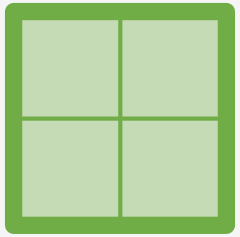


# POOLING (POOL MAX)

**Kernel size : 2 x 2**

**Stride : 2**

**Image Filtrée avec RELU**

POOL MAX

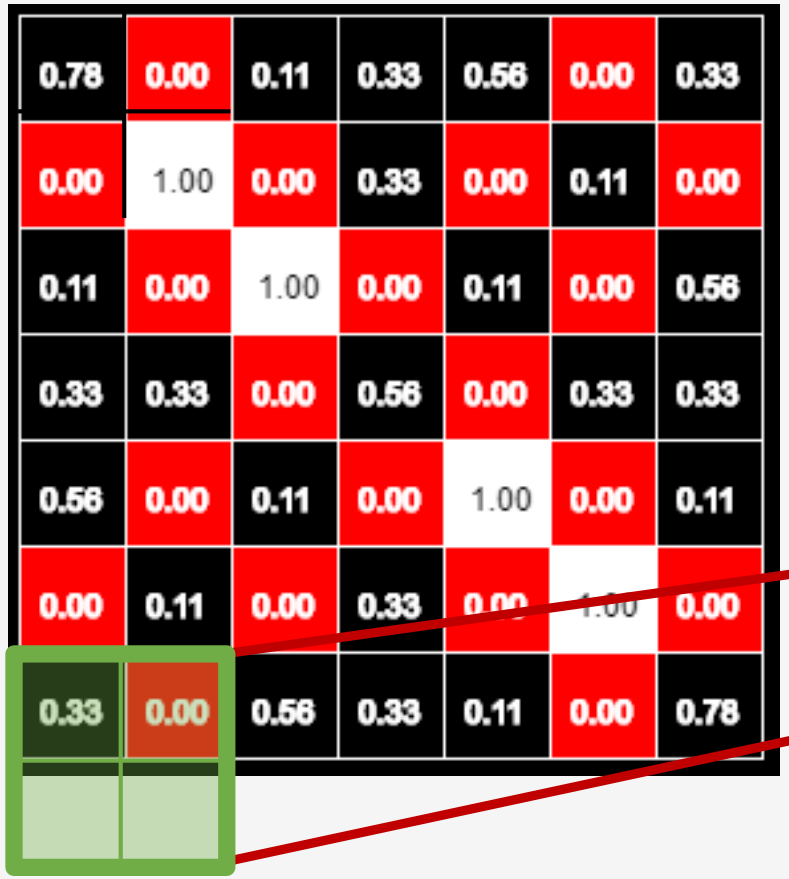
**Image Après Pool Max**

# POOLING (POOL MAX)

**Kernel size : 2 x 2**

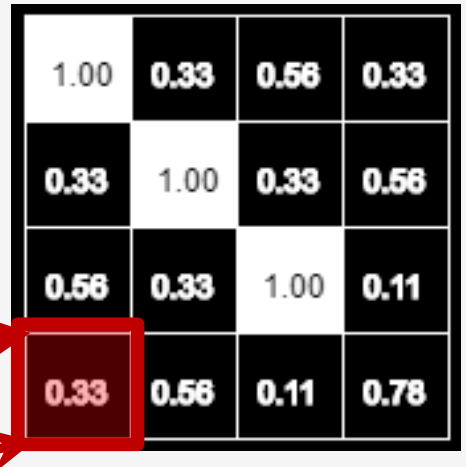
**Stride : 2**

**Image Filtrée avec RELU**




POOL MAX

**Image Après Pool Max**

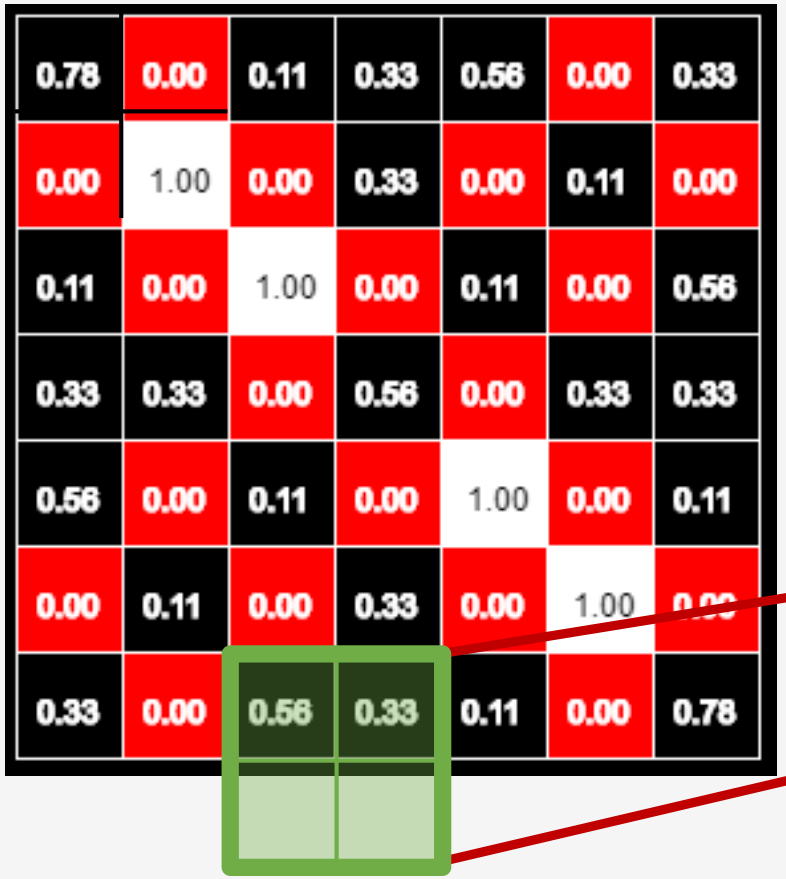


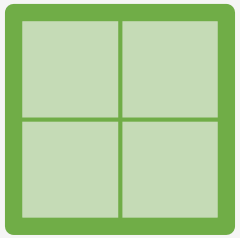
# POOLING (POOL MAX)

**Kernel size : 2 x 2**

**Stride : 2**

**Image Filtrée avec RELU**



POOL MAX

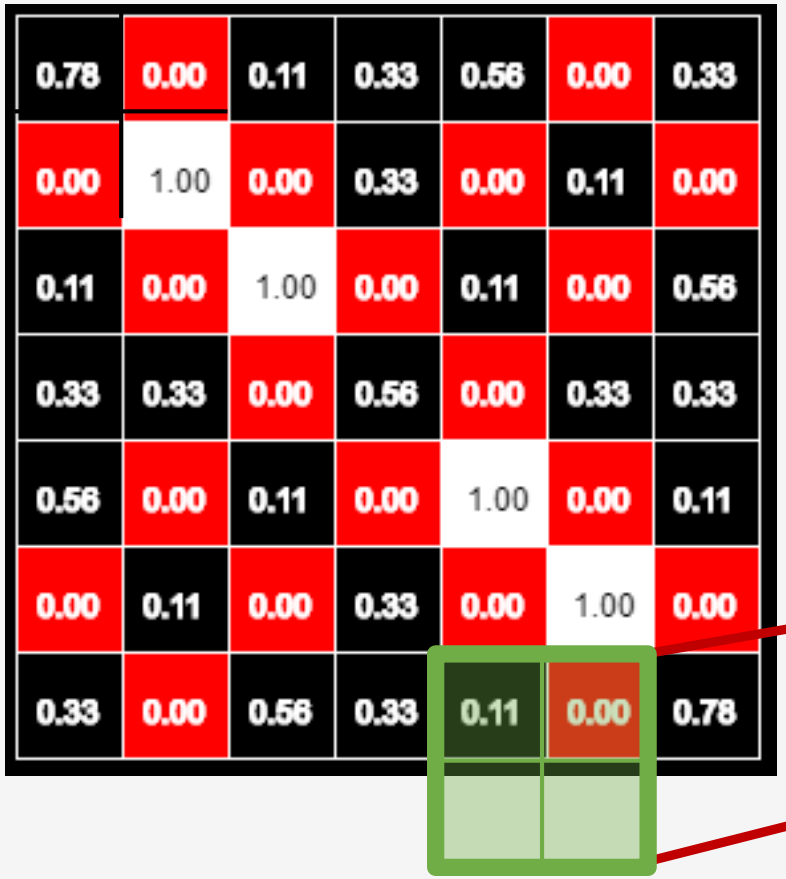
**Image Après Pool Max**

# POOLING (POOL MAX)

**Kernel size : 2 x 2**

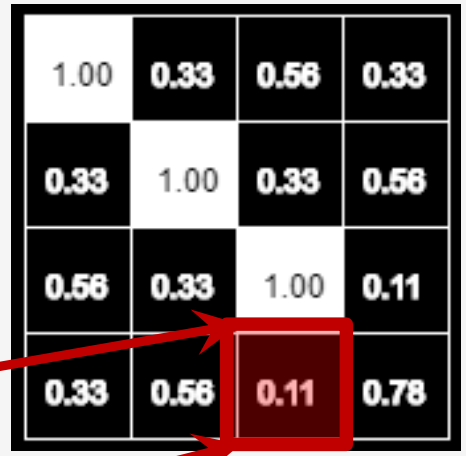
**Stride : 2**

**Image Filtrée avec RELU**




POOL MAX

**Image Après Pool Max**



# POOLING (POOL MAX)

**Kernel size : 2 x 2**

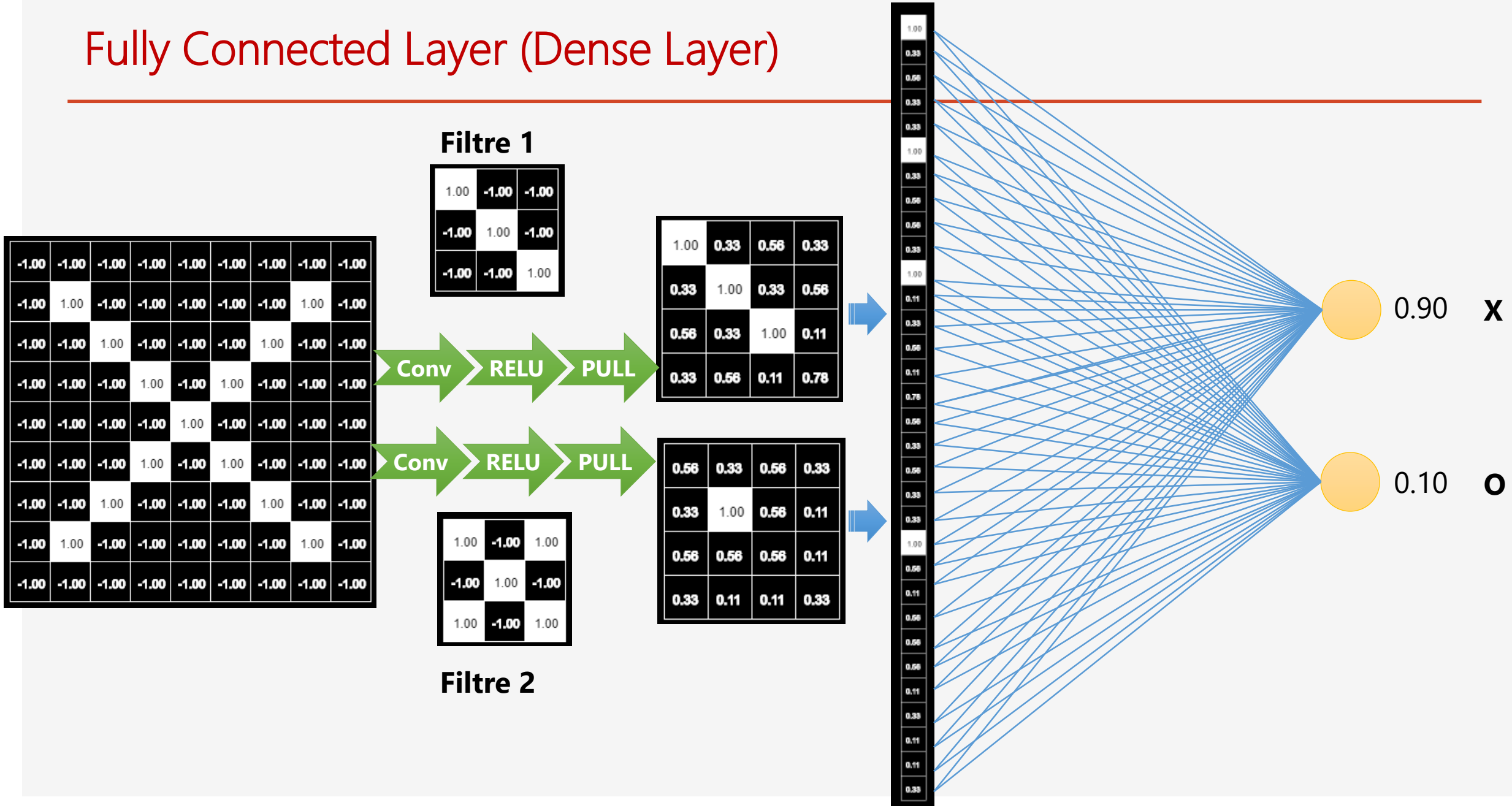
**Stride : 2**

**Image Filtrée avec RELU**

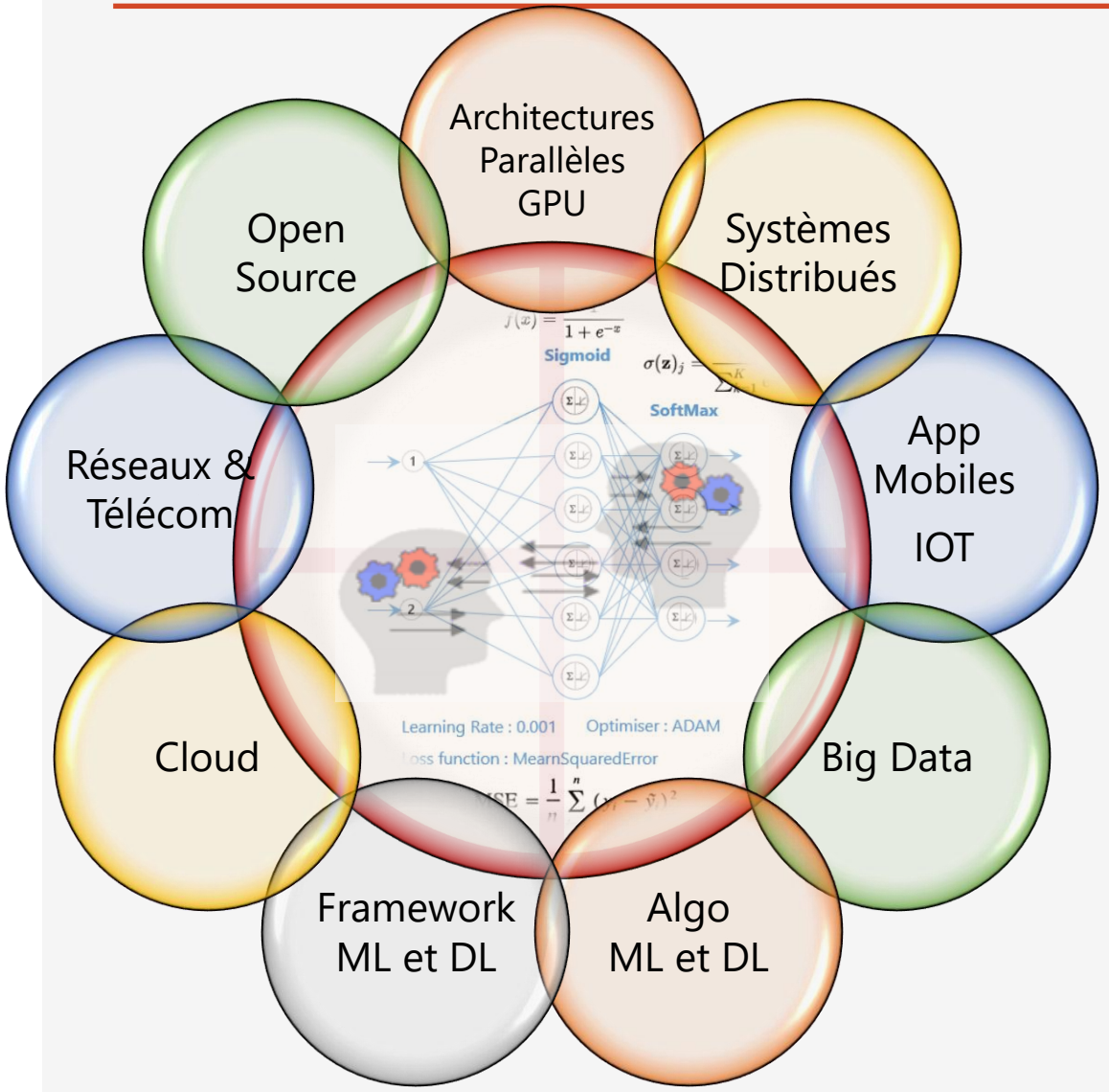

POOL MAX

**Image Après Pool Max**

# Fully Connected Layer (Dense Layer)



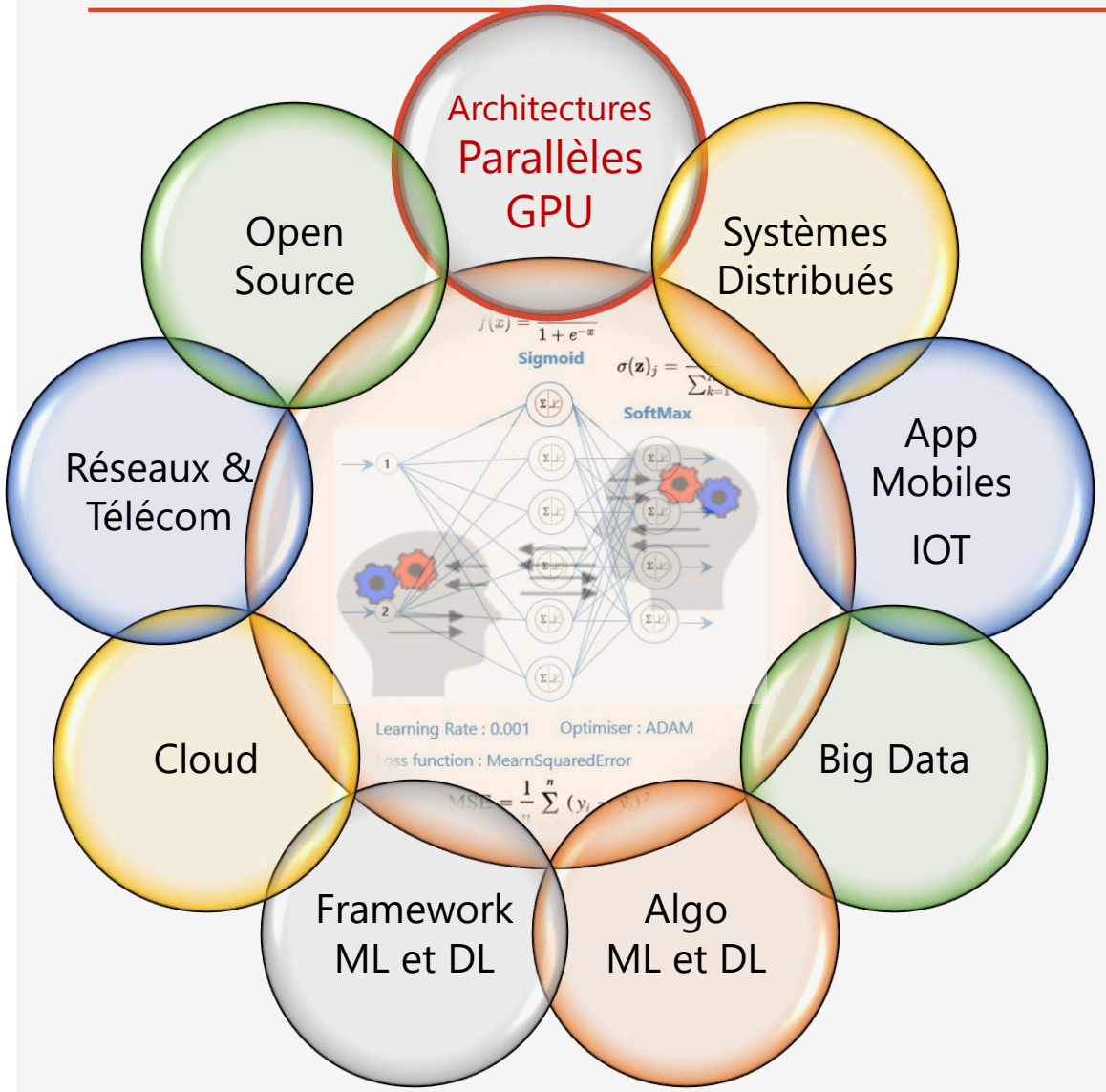
# Période d'Incertitude de  l'Intelligence Artificielle




Pour avoir un modèle qui représente plus la réalité, il faudrait bien choisir :

- Le bon type du modèle (MLP, CNN, RNN,  etc.)
- Le bon nombre de couches de neurones
- Pour chaque couche il faut choisir :
  - Le nombre de neurones adéquat
  - La bonne fonction d'activation (RELU, SIGMOID, …)
  - Le nombre de filtres, la taille des filtres, types filtres
- La bonne valeur de la vitesse d'apprentissage (Entre 0 et 1)
- La bonne fonction de perte (Loss Function) produisant l'erreur à minimiser (Mean squared error, Cross entropy, etc.)
- Le bon algorithme de rétropropagation du gradient permettant de mettre à jour les poids des connexions en minimisant la fonction de perte  : SGD, ADAM, etc...
- Et surtout un data set de très grande taille pour représenter le maximum d'échantillons de la réalité.
- Ce qui fait que pendant la phase d'apprentissage => **Beaucoup de temps de calcul**

# Catalyseurs de l'Intelligence Artificielle : Architectures Massivement Parallèles

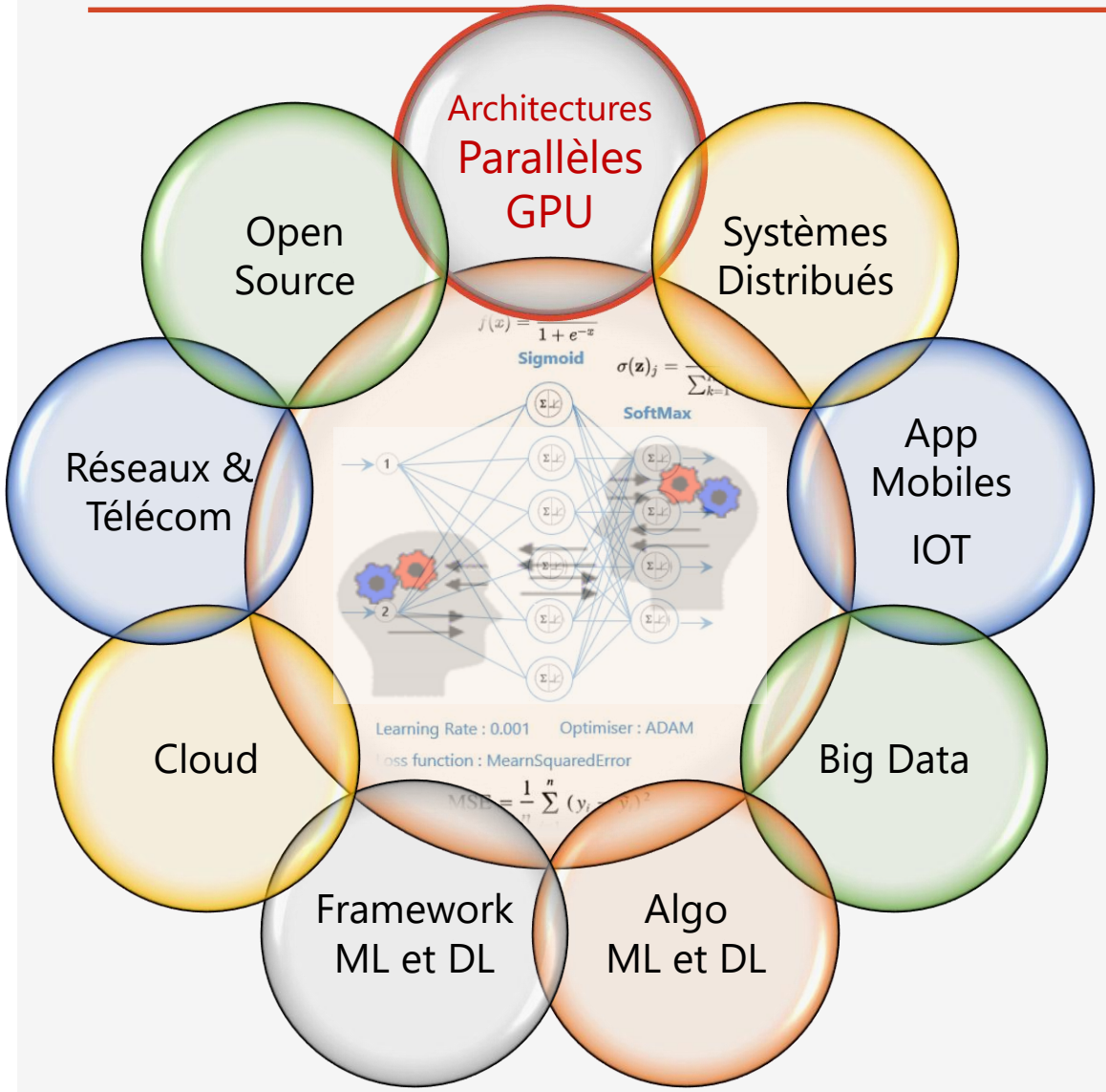



## ➢ Architectures Massivement parallèles avec GPU :

- **Vers 2010, Nvidia GeForce 3 ouvre ses GPU pou GP GPU avec CUDA**
- **AMD avec sa carte Radeon**
- **IntelHD : Circuits intégrés dans le CPU**
- **OpenCL pour GPGPU Cross Platform**
- **OpenGL pour GPGPU pour les systèmes embarqués**
- **WebGL pour GPGPU dans les navigateurs Web**


Carte GPU Nvidia GeForce GTX Titan Z : ( prix $2999 )
5760 stream processors, 12GB of fast GDDR5 memory

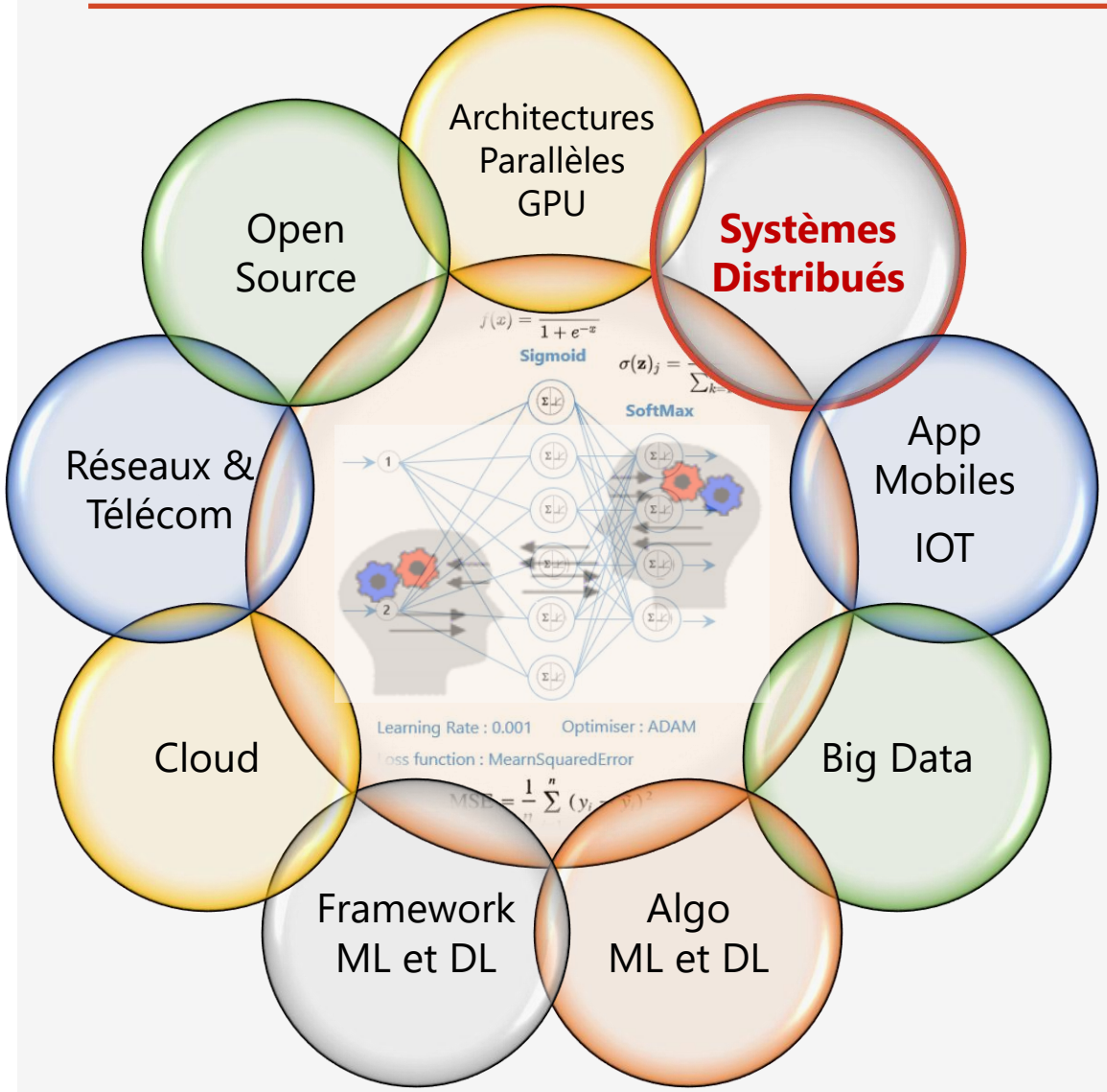
# Catalyseurs de l'Intelligence Artificielle : Architectures Massivement Parallèles




➢ **Architectures Massivement parallèles avec GPU :**

8 GPU TITAN RTX RIG -Crypto Mining for altcoins & super-computing, 40,000+ cores ( Prix US $29,990.00 )

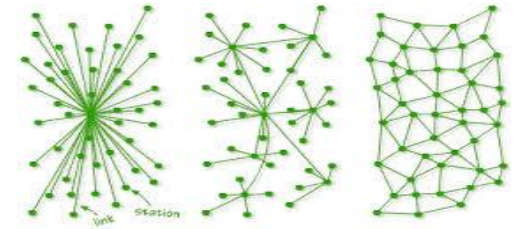
# Catalyseurs de l'Intelligence Artificielle : Systèmes Distribués




## Systèmes Distribués

- Middlewares RMI, CORBA, JMS, MPI
- Grilles de Clacul
- Protocoles de messageries asynchrones:
- **AMQP, MQTT, STOMP**
- Brokers : **RabbitMQ, ActiveMQ**
- Caches mémoires Distribués : **Hazelcast**
- Middlewares **SMA : JADE**

**Systèmes Distribués Middlewares**



CORBA Java RMI MPI

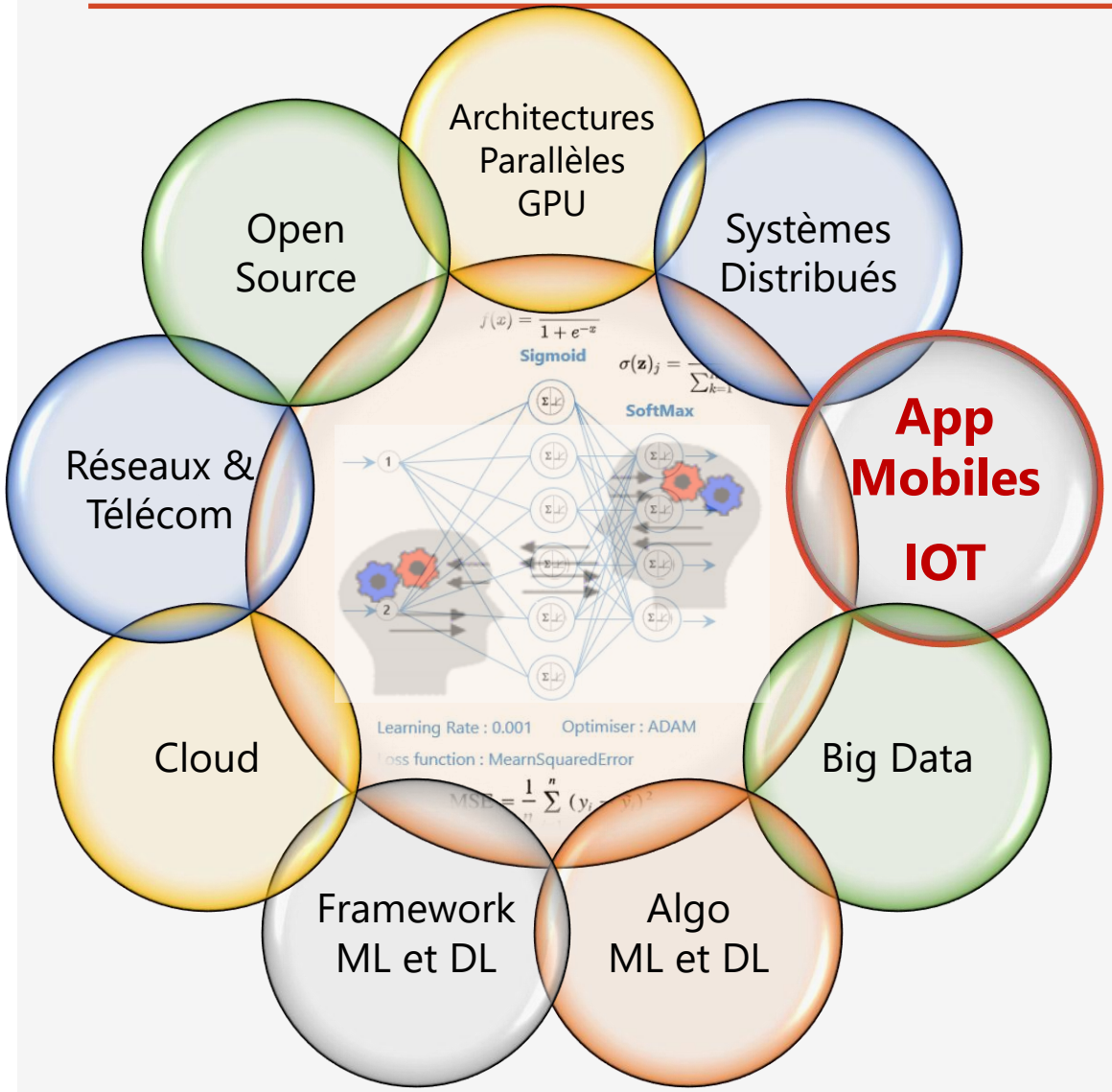
RabbitMQ Apache ACTIVEMQ

AMQP MQTT

Stomp

hazelcast

# Catalyseurs de l'Intelligence Artificielle : Applications Mobiles et IOT



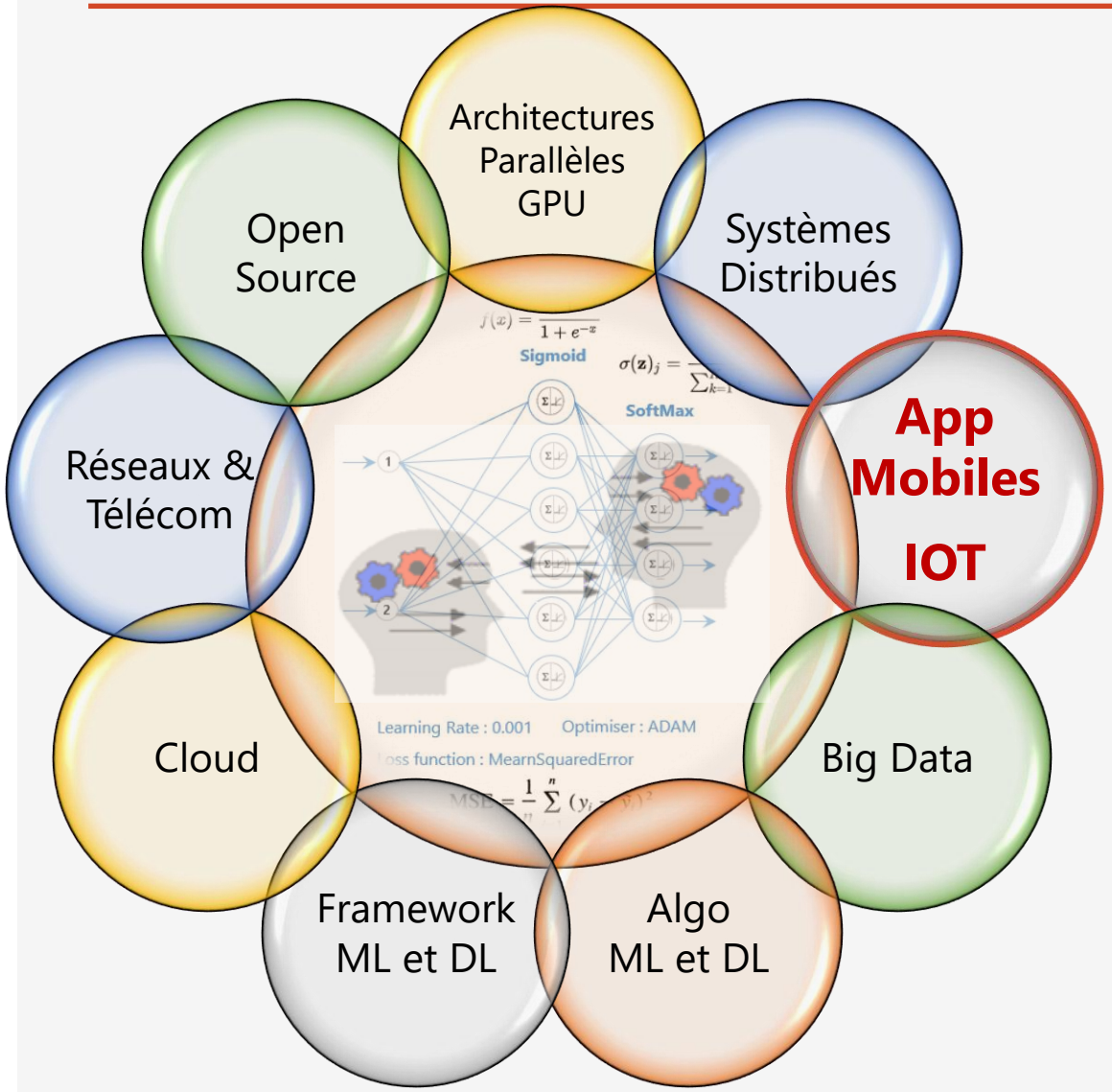

## ➢ Capteurs d'un Smartphone

- Caméras (Avant et Arrière)
- Ecran tactile
- Microphones (1 à 3)
- Détecteur de proximité
- Capteur de luminosité
- Capteur effet hall
- Magnétomètres : 3 capteurs (Boussole, guidage GPS, détecteur de métaux, Réalité Augmentée.)
- Gyromètre ( gyroscope) : 3 Capteurs
- Accéléromètre : 3 capteurs
- Thermomètre et Hygromètre
- Capteur de geste infrarouge (Détection du Mvt de la main)
- Capteur de pression (Baromètre, altimètre. )
- Scanner d'empreintes digitales
- Capteur d'impulsions cardiaques
- …

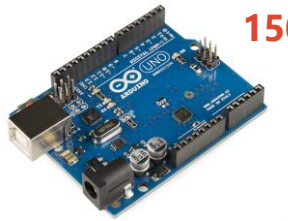
## ➢ Développement Mobile

- Développement Mobile Natif (Android, IOS, WindowsPhone)
- Développement Mobile Cross Platform (IONIC, React Native, Flutter)

# Catalyseurs de l'Intelligence Artificielle : IOT et Applications Mobile



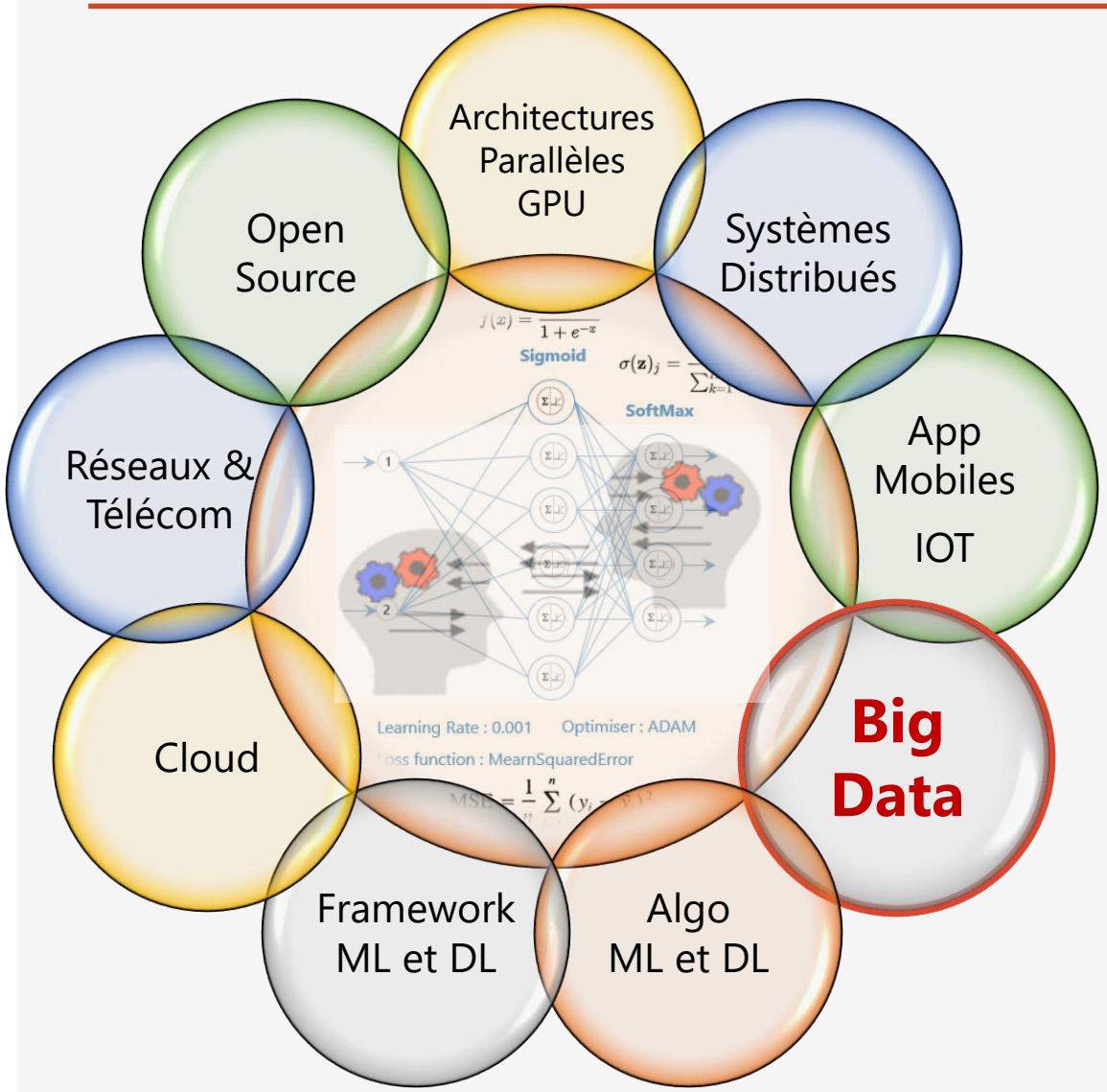

- **IOT**
  - Cartes Raspburry
  - Cartes Arduino
  - Des milliers de capteurs



**150 DH**

**250 DH**

# Catalyseurs de l'Intelligence Artificielle : Big Data



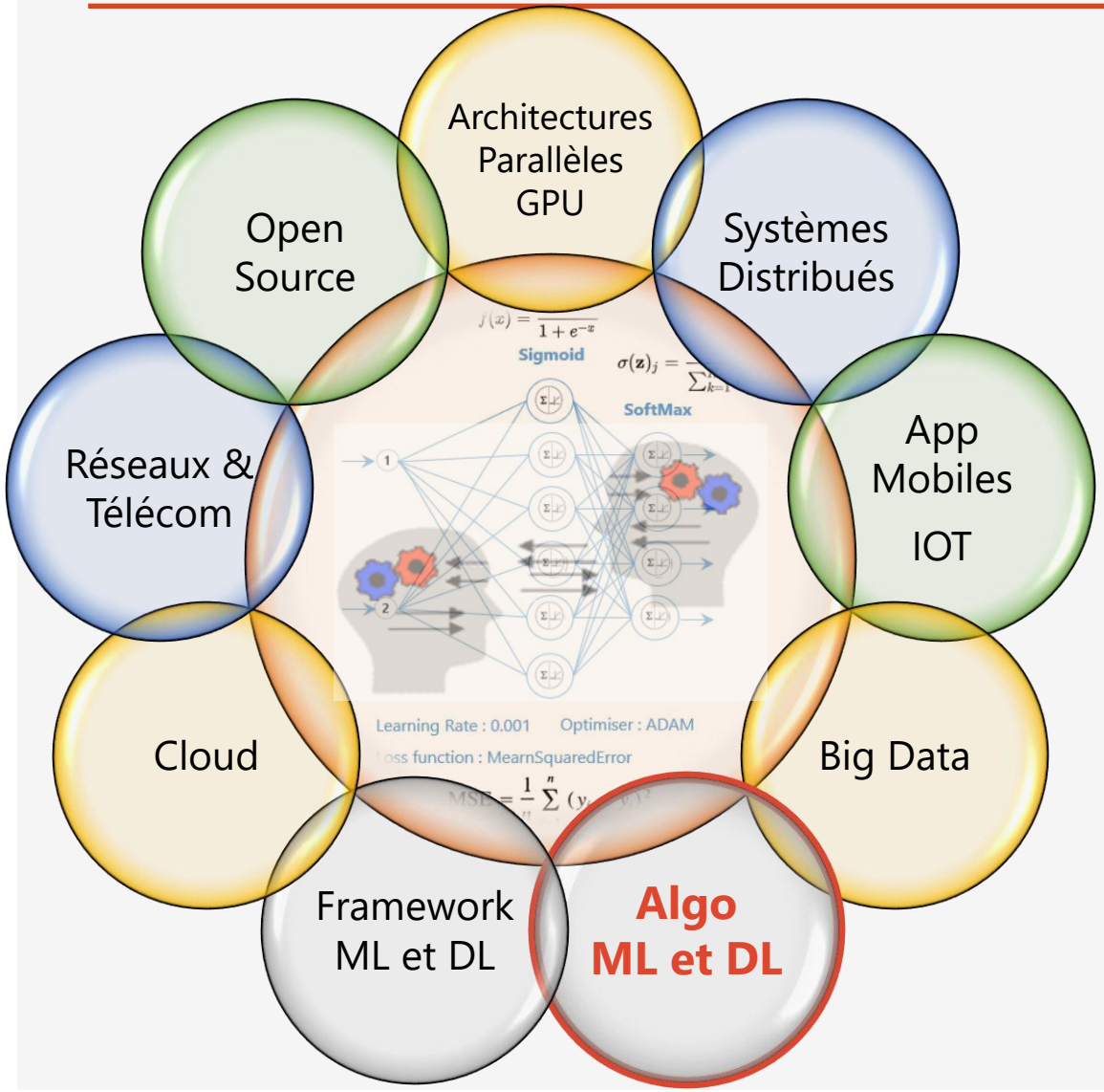

- Stockage :
  - HDFS : Hadoop
  - NoSQL : MongoDB, Cassandra,..
  - BD Orientés Graphes : Neo4J
- Batch Processing : MapReduce,
- Micro-Batch Processing: Spark,
- Stream Processing : KAFKA Stream
- Caches Mémoires Distribués : Hazelcast

- Programmation fonctionnelle :
  - Java script
  - Python
  - Scala, Kotlin
- Programmation Réactive
  - ReactiveX
  - Reactor

**Big Data Et Systèmes Distribués Middlewares**

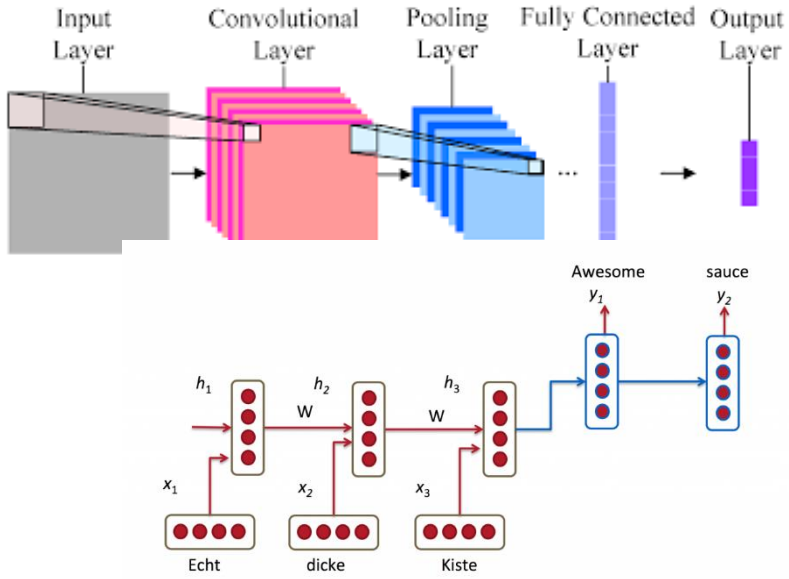
hadoop, APACHE Spark, MapReduce, kafka, RabbitMQ, Apache ACTIVEMQ, mongoDB, cassandra, AMQP, MQTT, hazelcast, Stomp

# Catalyseurs de l'Intelligence Artificielle : Algorithmes DL

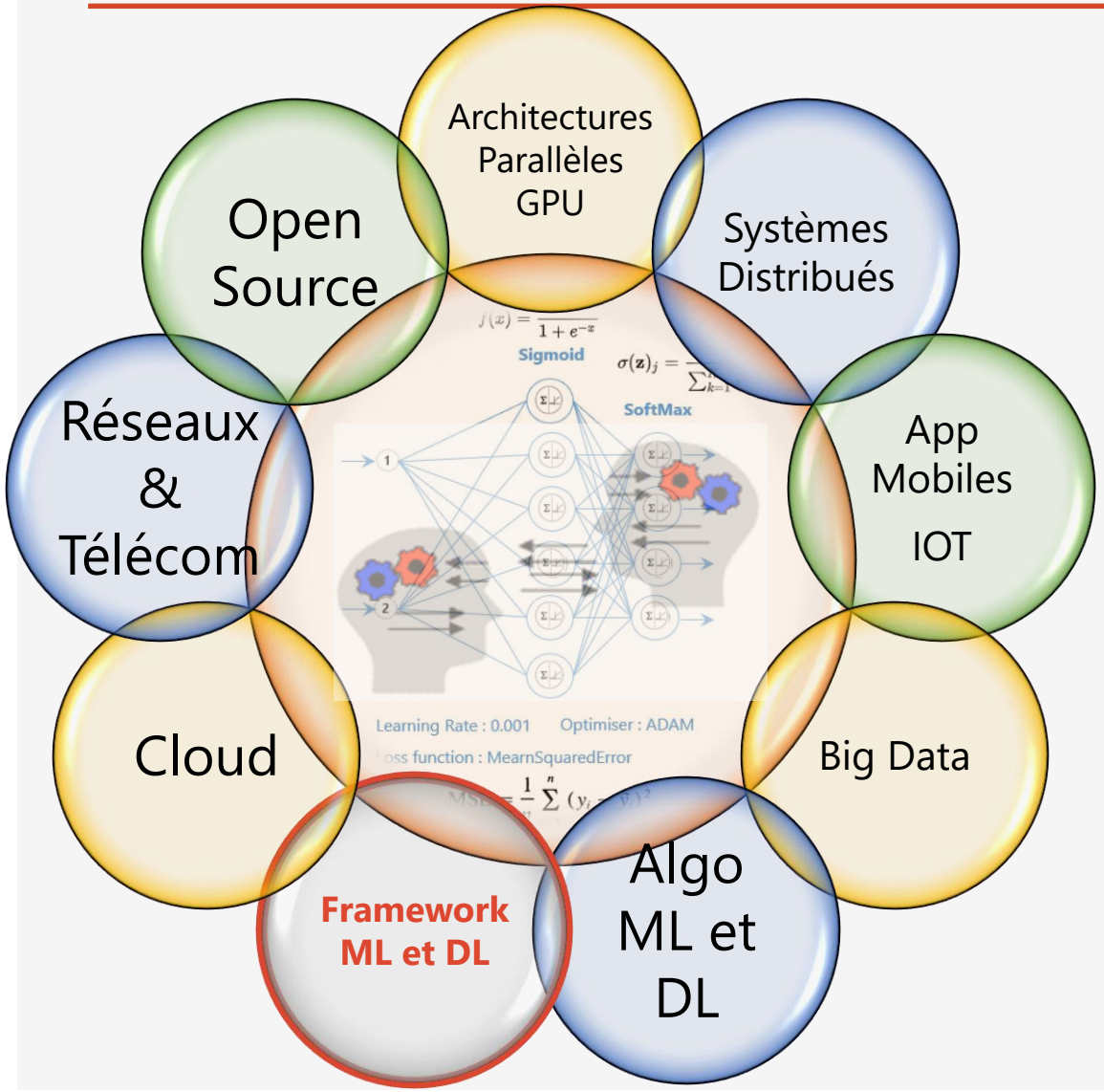



- Deep Learning (Apprentissage par les données)
  - Convolution Neural Network (**CNN** )
  - Recurrent Neural Network (**RNN**)
- **Deep Reinforcement Learning** (CNN+QLearning)

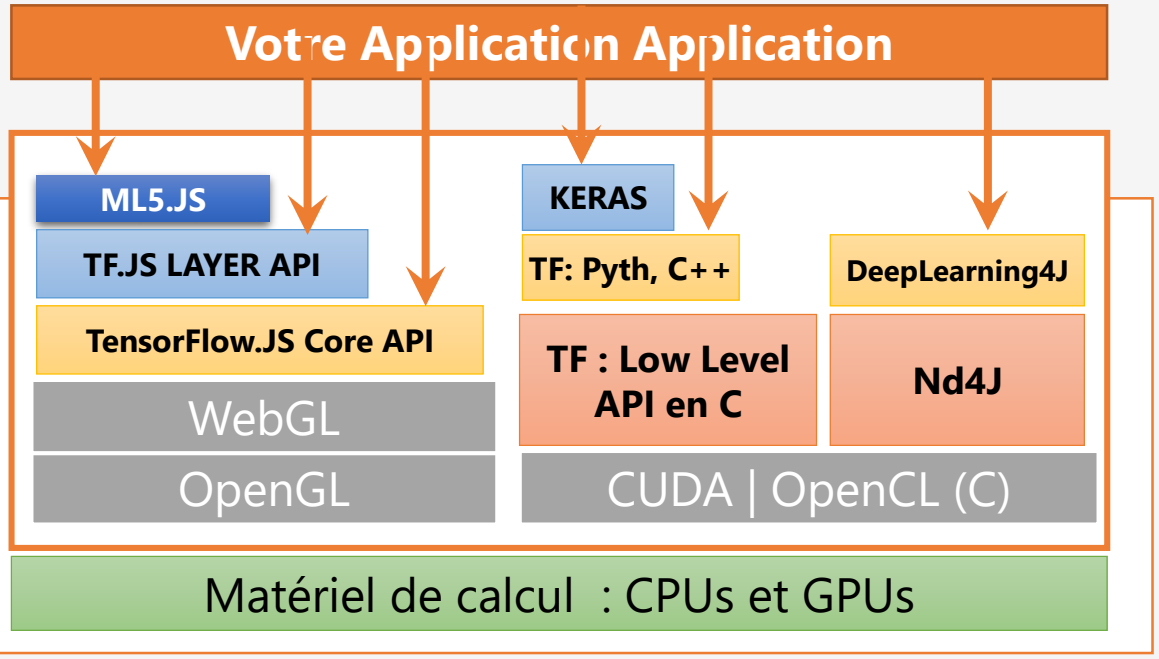
(RNN+QLearning)



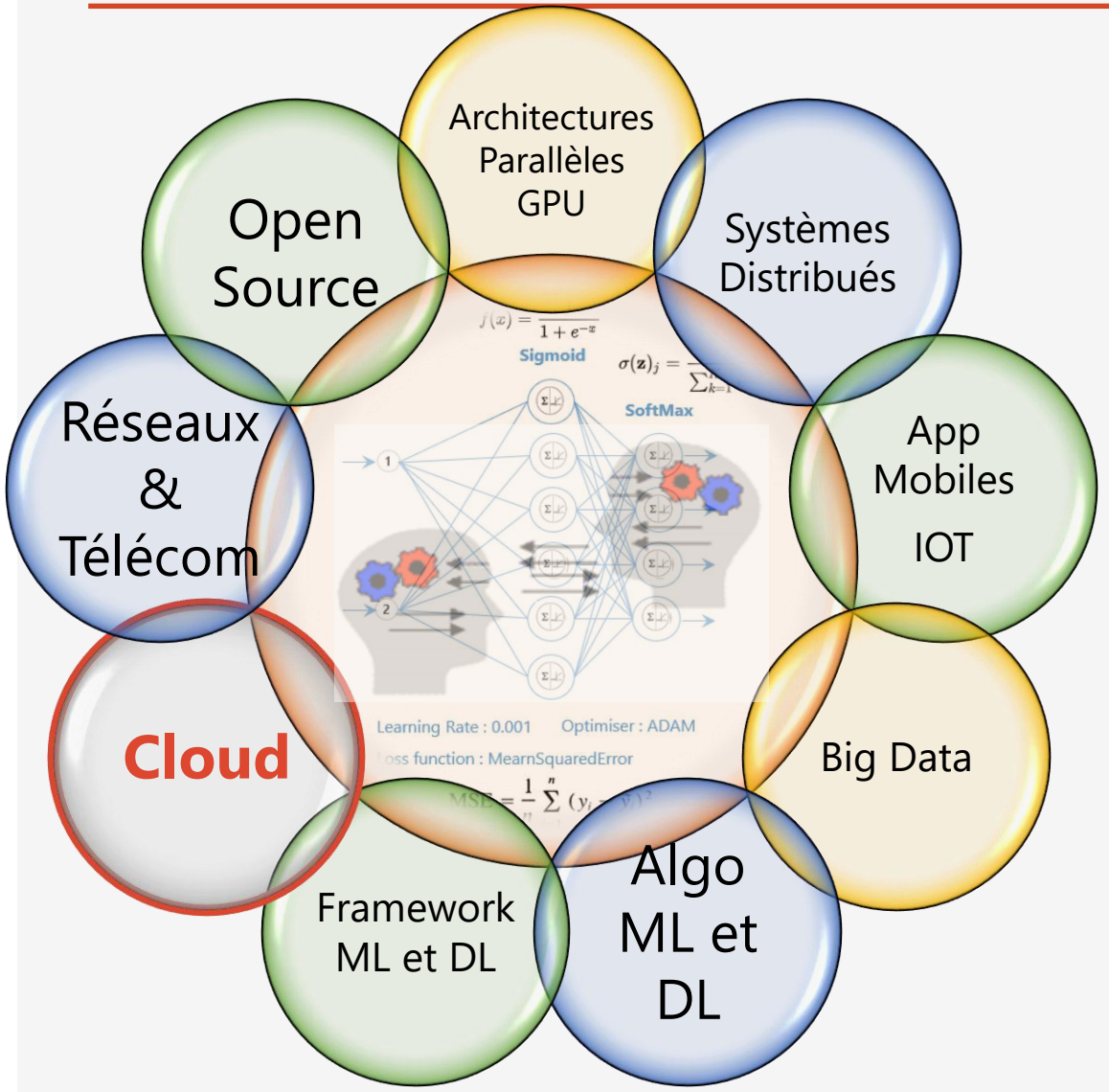
# Catalyseurs de l'Intelligence Artificielle : Framework de ML et DL




- Mise en pratique les algo de l'IA en exploitant les GPUs et les SD.
- API de calcul parallèle sur GPU des algo de l'algèbre linéaire basée sur C (CUDA, OpenCL)
- API de haut niveau avec Python, C++, Java
- 2015 : Framework de ML : TensorFlow
- API de haut niveau : KERAS (Python et C++)
- PyTorch, ML.Net(C#), DL4J (Java), TensorFlow.JS
- Algo basés sur les RN: MLP, CNN et RNN



# Catalyseurs de l'Intelligence Artificielle : Cloud Computing



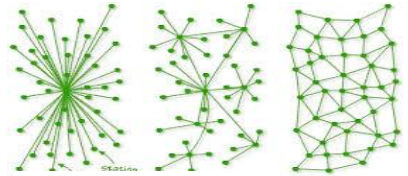

- Virtualisation des machines : VMWare
- Cloud Computing (Systèmes Distribués)
- SaaS, IaaS, PaaS
- Google, Microsoft, Amazon, Facebook, IBM,…
- Cloud Micro-services avec Spring Cloud
- Docker : Conteneurs légers
- Kubernetes pour l'orchestration des micro-services

**Virtualisation & Cloud Computing**



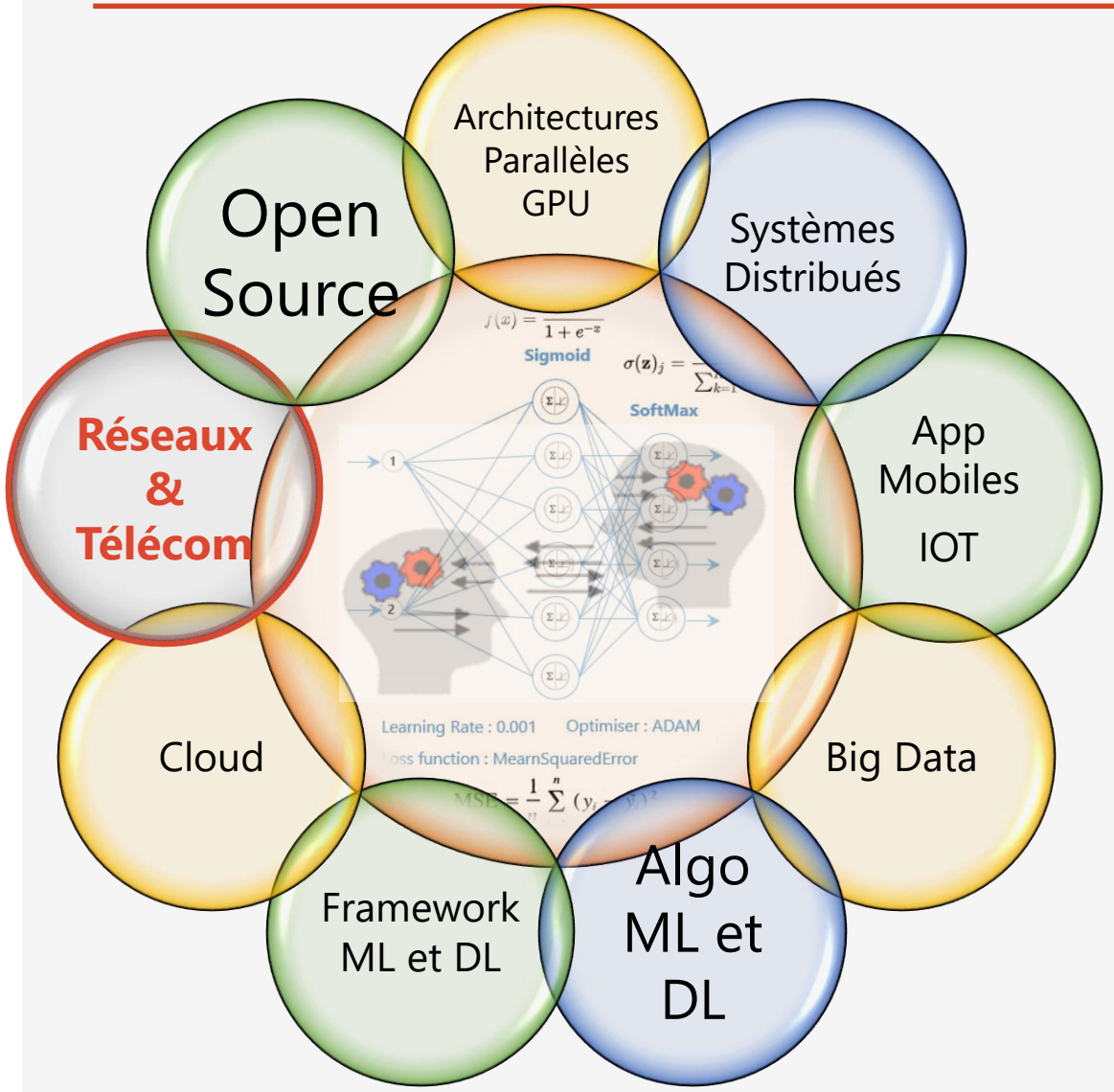




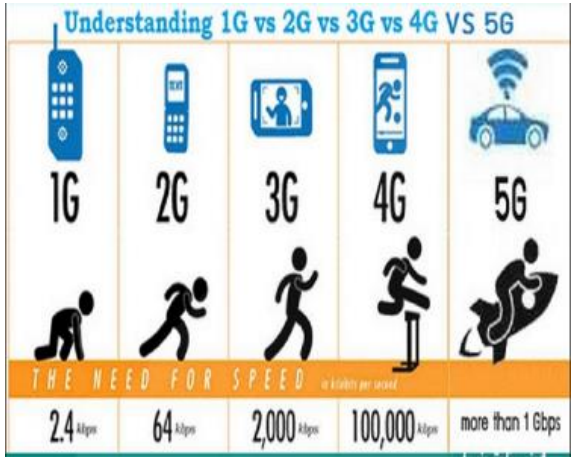



# Catalyseurs de l'Intelligence Artificielle : Réseaux Télécom

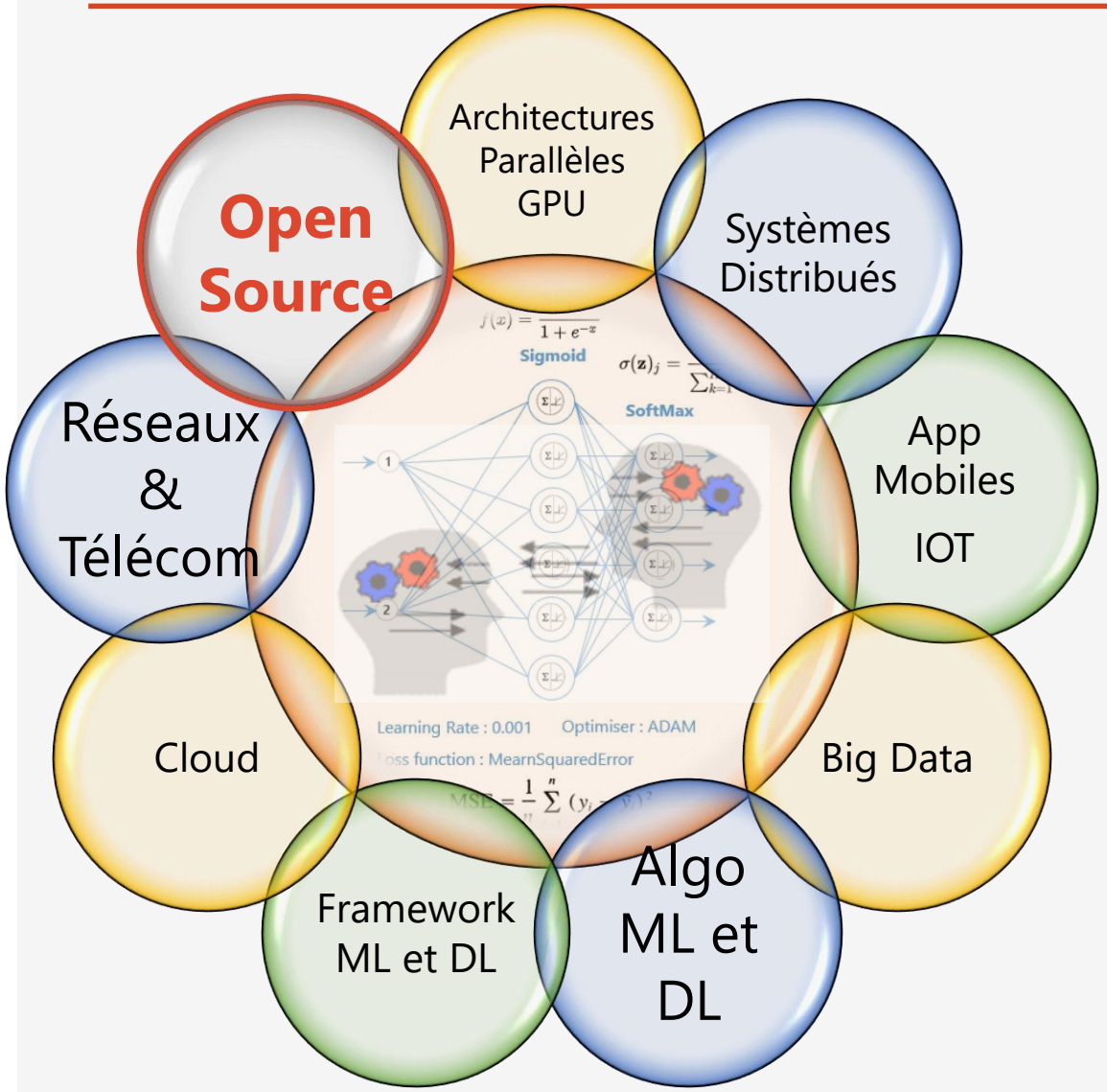



- 3G, 4G, 5G,
- Wireless,
- Fibre Optique )

**Virtualisation & Cloud Computing**



# Catalyseurs de l'Intelligence Artificielle : Open Source




# TensorFlow.JS

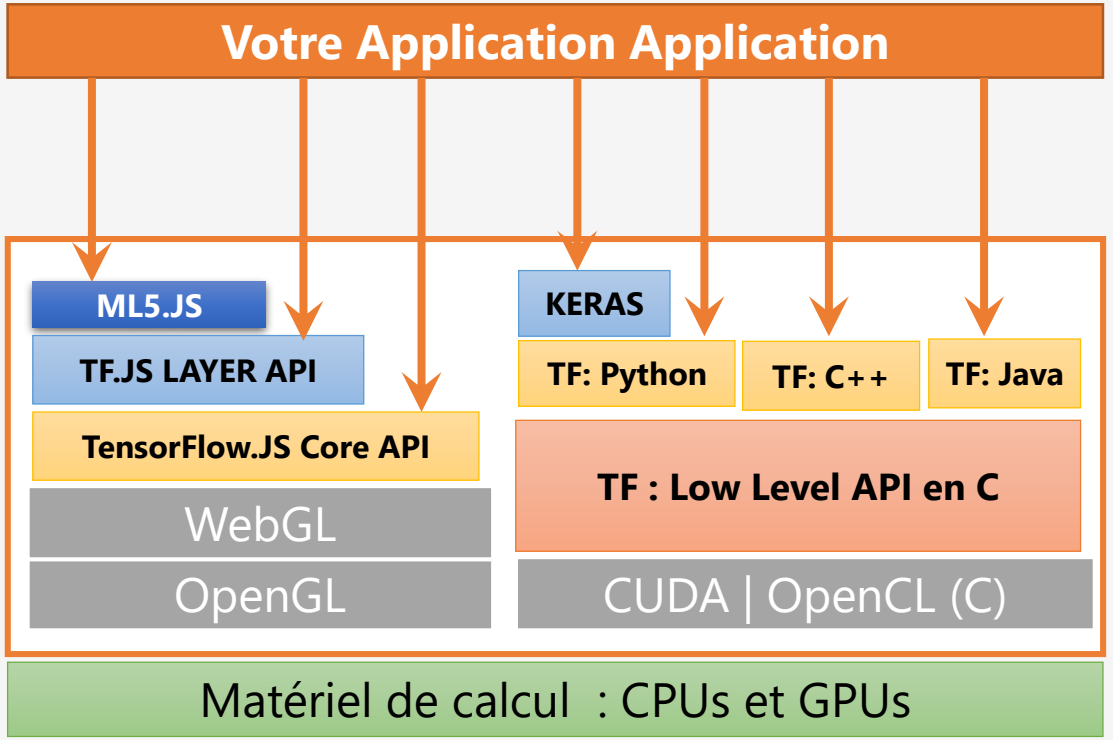
- TensorFlowJS est un Framework de machines et Deep Learning basée sur JavaScript

- Ce Framework s'appuie sur WebGL qui permet d'utiliser les GPUs pour accélérer les calcul

# Architecture de TensorFlow

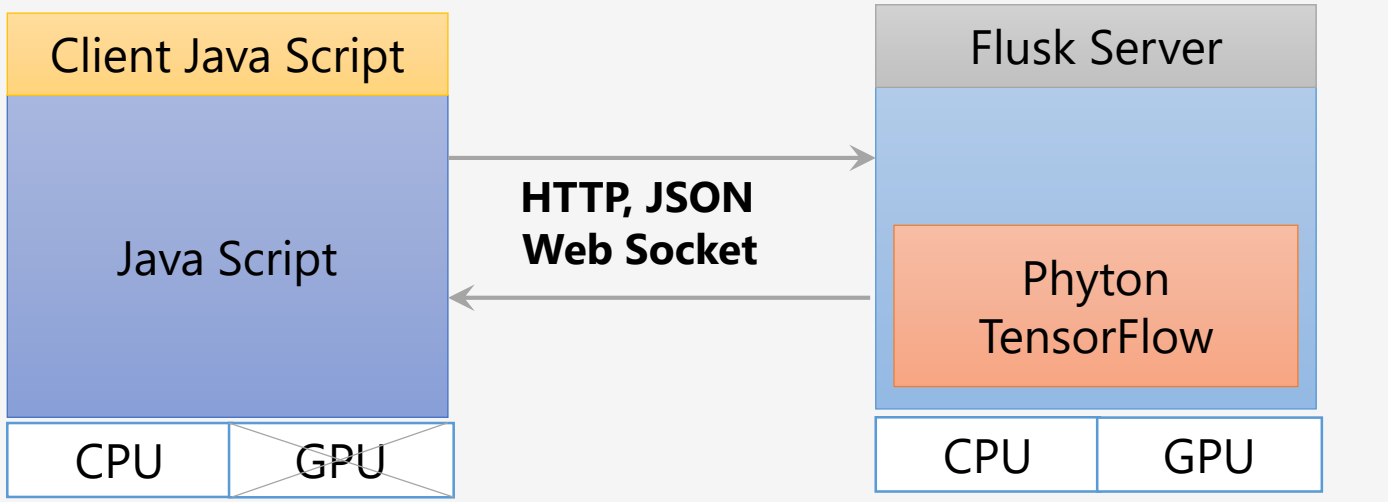
**Signification :**

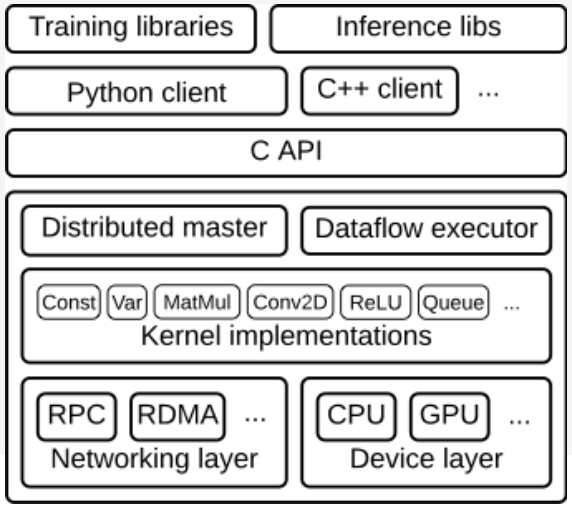
- **Tensor :**
  - Object représentant une structure mathématique qui peut représenter :
    - un nombre scalaire, Un vecteur, Une matrice 2D, Une matrice 3D, 4D, … nD
  - Avec des opérations de l'algèbre linéaire permettant différents calcul matriciels pouvant exploiter les CPUs et les architectures massivement parallèles à base des GPUs.
- **Flow** : signifie Débit ou Flux




- La couche bas niveau de TensorFlow est développée en C car la programmation parallèle sur les GPU de Nvidea utilise CUDA qui est basé sur le langage C.
- Il existe plusieurs implémentations de API de haut niveau permettant programmer des applications de faire du machines learning basées sur TensorFlow :
  - API Python, API C++, API Java
- KERAS est une API de haut niveau qui permet de faire du machines Learning avec TensorFlow d'une manière plus simple
- TensorFlow.JS est une implémentation Java Script de TensorFlow basée sur WebGL

# Comment utiliser TensorFlow avec Java Script

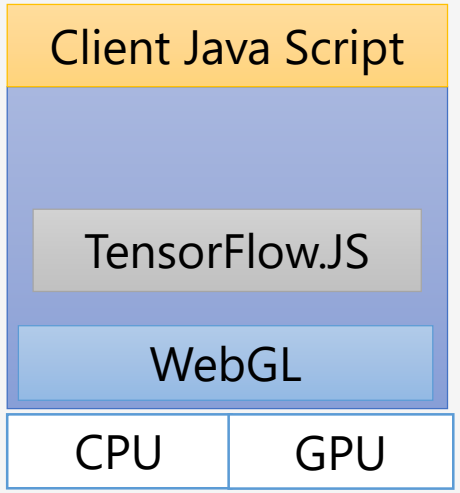


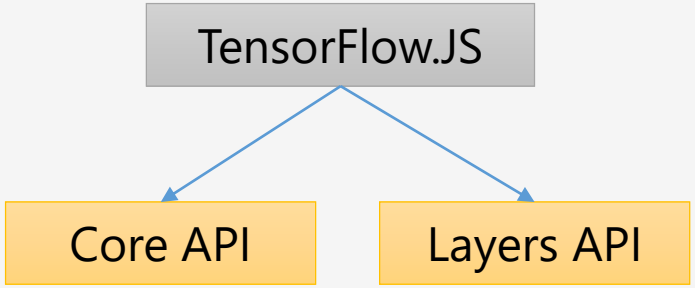




Solution classique : Machines Learning Sever Side

- Par exemple Utiliser Python Flusk Server

# TensorFlow.JS




- Core API : Low Level API
  - Basic buildin blocs of the TensorFlow.JS
- Layers API : Hi Level API
  - Pour créer facilement différents types de modèles
  - Equivalent à KERAS

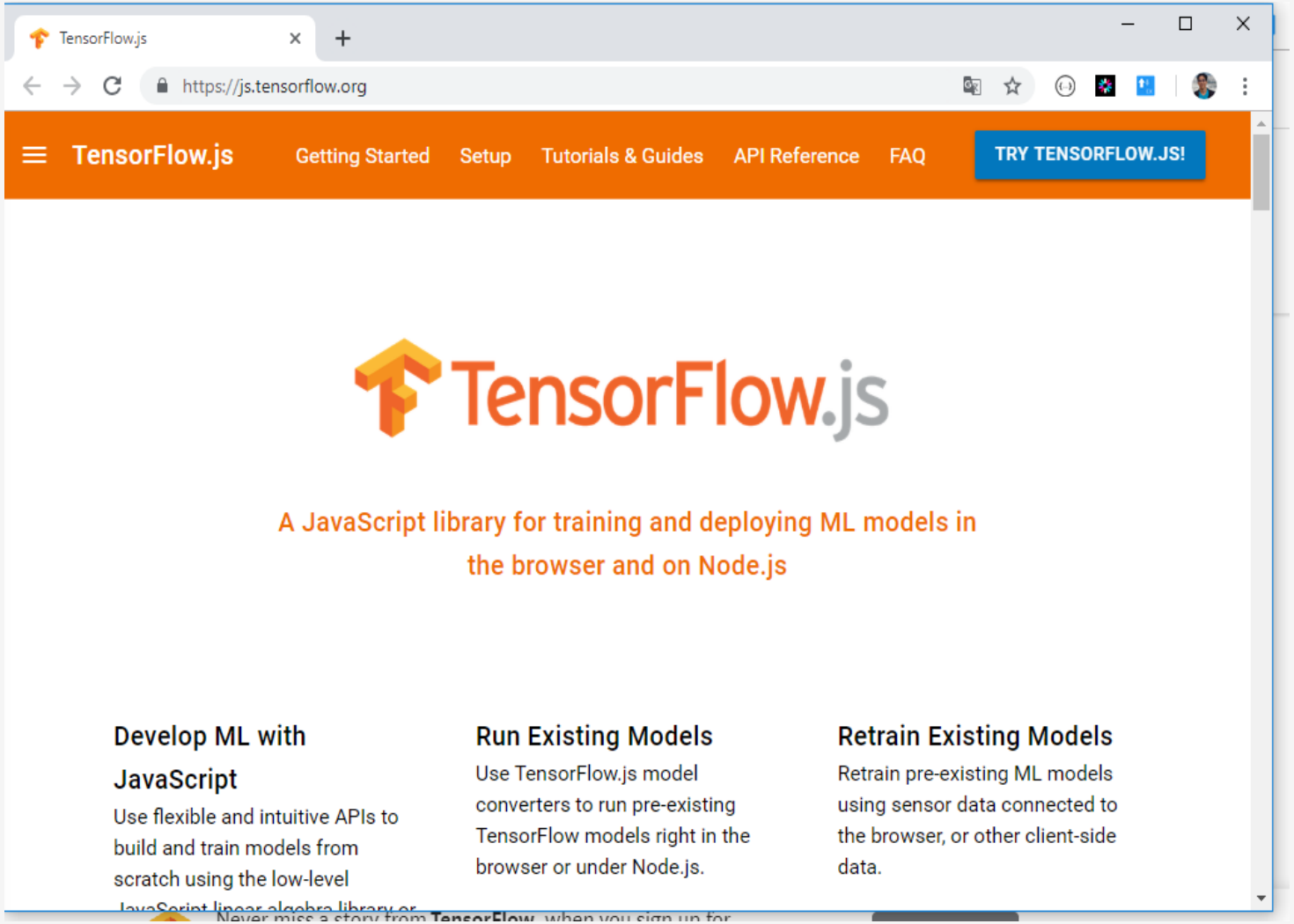
- WebGL est interface des Browsers web vers :
  - OpenGL ES 2.0 (OpenGL for Embedded Systems )
  - qui permet d'effectuer des calculs sur les GPUs.

# TensorFlow.JS

- **Tensorflow.js** est une bibliothèque construite sur **deeplearn.js** pour créer des modules d'apprentissage en profondeur directement sur le navigateur.

- Créer des modèles d'apprentissage CNN, RNN, etc.  Les entrainer directement sur le navigateur en utilisant la puissance de traitement du **GPU du client**.

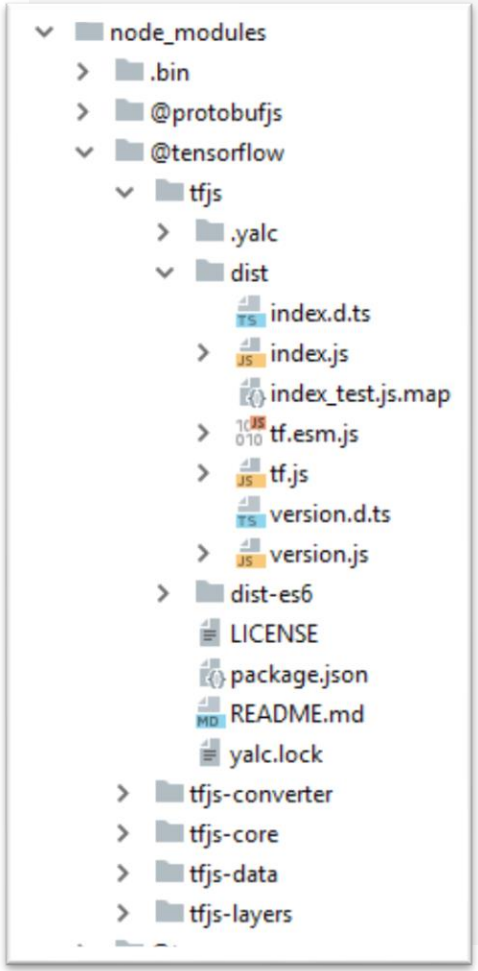
- Par conséquent, un **GPU de serveur n'est pas nécessaire** pour former le NN.



# Intégration de TensorFlow.JS dans une page HTML

## Intégration de TensorFlow.JS  dans HTML

```
<script src="https://cdn.jsdelivr.net/npm/@tensorflow/tfjs@0.14.1/dist/tf.min.js"> </script>
```




- Télécharger la librairie TensorFlowJS :
  - npm install --save @tensorflow/tfjs

```
<script src="node_modules/@tensorflow/tfjs/dist/tf.min.js"></script>
```

# Tensors.JS : Core API => Tensors tf.tensor (values, shape?, dtype?)

TensorFlow.js

```
let t1=tf.tensor([2,6,7,8,9,5],[2,3]);
console.log("Tensor t1 [2x3]:");
t1.print();

let t2=tf.tensor([2.7,6,1,2,1,4],[3,2]);
console.log("Tensor t2 [3x2]:");
t1.print();

console.log("Tensor t3 transposée de t1:");
let t3=t1.transpose();
t3.print();

console.log("Tensor t4: t1 x t2 ");
let t4=t1.matMul(t2);
t4.print();

t1.dispose();t2.dispose();t3.dispose();t4.dispose();
```

**t1**

| | | |
|---|---|---|
| 2 | 6 | 7 |
| 8 | 9 | 5 |

**x**

**t2**

| | |
|---|---|
| 2.7 | 6 |
| 1 | 2 |
| 1 | 4 |

```
Tensor t1 [2x3]:
[ [2, 6, 7],
  [8, 9, 5] ]
Tensor t2 [3x2]:
[[2, 6, 7],
 [8, 9, 5]]
Tensor t3 transposée de t1:
[[2, 8],
 [6, 9],
 [7, 5]]
Tensor t4: t1 x t2
[[18.3999996, 52],
 [35.5999985, 86]]
```

# Libérer les Tensors :

TensorFlow.js

**`tensor.dispose();` ou bien `tf.tidy(()=>{});`**

```
tf.tidy(()=>{
    let t1=tf.tensor([2,6,7,8,9,5],[2,3]);
    console.log("Tensor t1 [2x3]:");
    t1.print();
    let t2=tf.tensor([2.7,6,1,2,1,4],[3,2]);
    console.log("Tensor t2 [3x2]:");
    t1.print();
    console.log("Tensor t3 transposée de t1:");
    let t3=t1.transpose();
    t3.print();
    console.log("Tensor t4: t1 x t2 ");
    let t4=t1.matMul(t2);
    t4.print();
});
```

**t1**

| | | |
|---|---|---|
| 2 | 6 | 7 |
| 8 | 9 | 5 |

**x**

**t2**

| | |
|---|---|
| 2.7 | 6 |
| 1 | 2 |
| 1 | 4 |

```
Tensor t1 [2x3]:
[ [2, 6, 7],
  [8, 9, 5] ]
Tensor t2 [3x2]:
[[2, 6, 7],
 [8, 9, 5]]
Tensor t3 transposée de t1:
[[2, 8],
 [6, 9],
 [7, 5]]
Tensor t4: t1 x t2
[[18.3999996, 52],
 [35.5999985, 86]]
```

# TensorFlow.JS : Layers API

Layers API est la couche de haut niveau du Framework TensorFlow.JS

Cette couche s'appuie sur la couche de bas niveau qui est Core API

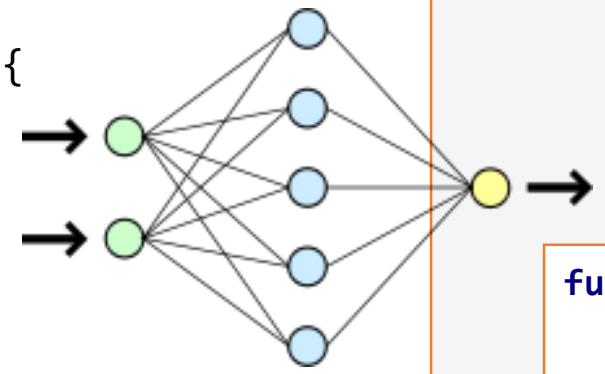
Cette API définit toutes les opérations nécessaire pour machines Learning:

- Créer des modèles de réseaux neurones
- Configurer les modèles
- Entrainer les modèles
- Tester les modèles
- Effectuer les prédictions
- Etc…

# TensorFlow.js Layers API

## 1. Création d'un Modèle

```
const  model=tf.sequential();

model.add(tf.layers.dense( {
    units:5,
    activation:'sigmoid',
    inputShape:[2]
}));

model.add(tf.layers.dense({
    units:1,
    activation:'sigmoid'
}));

let learningRare=0.1;

model.compile({
    optimizer:tf.train.sgd(learningRare),
    loss:tf.losses.meanSquaredError
});
```



## 2. Entrainement du modèle

```
const xs=tf.tensor2d([ [0,0], [0.5,0.5], [1,1] ]);
const ys=tf.tensor1d([ 1, 0.5, 0]);
const numberOfEpochs=1000;

model.fit(xs,ys,{
    epochs:numberOfEpochs,
    callbacks:{onEpochEnd:logProgress }
}).then(modelTrained);

function logProgress(epoch,resp){
    console.log(epoch+"=>"+resp.loss);
}
```

## 3. Prédiction

```
function modelTrained(){
    console.log("Model trained");
    let data=[0.5,0.5] ;
    const input=tf.tensor2d([data],[1,2]);
    let predicted=model.predict(input);
     let output=predicted.dataSync()[0];
 console.log("x1="+data[0]+",x2="+data[1]+", output="+output);
}
```

# TensorFlow.JS : Layers API

## 4. Enregistrement du modèle

```
function onSaveModel(){
 model.save('localstorage://my-model-1’) .then(

   (result)=>{
   console.log("model saved :"+result);
    });
}
```

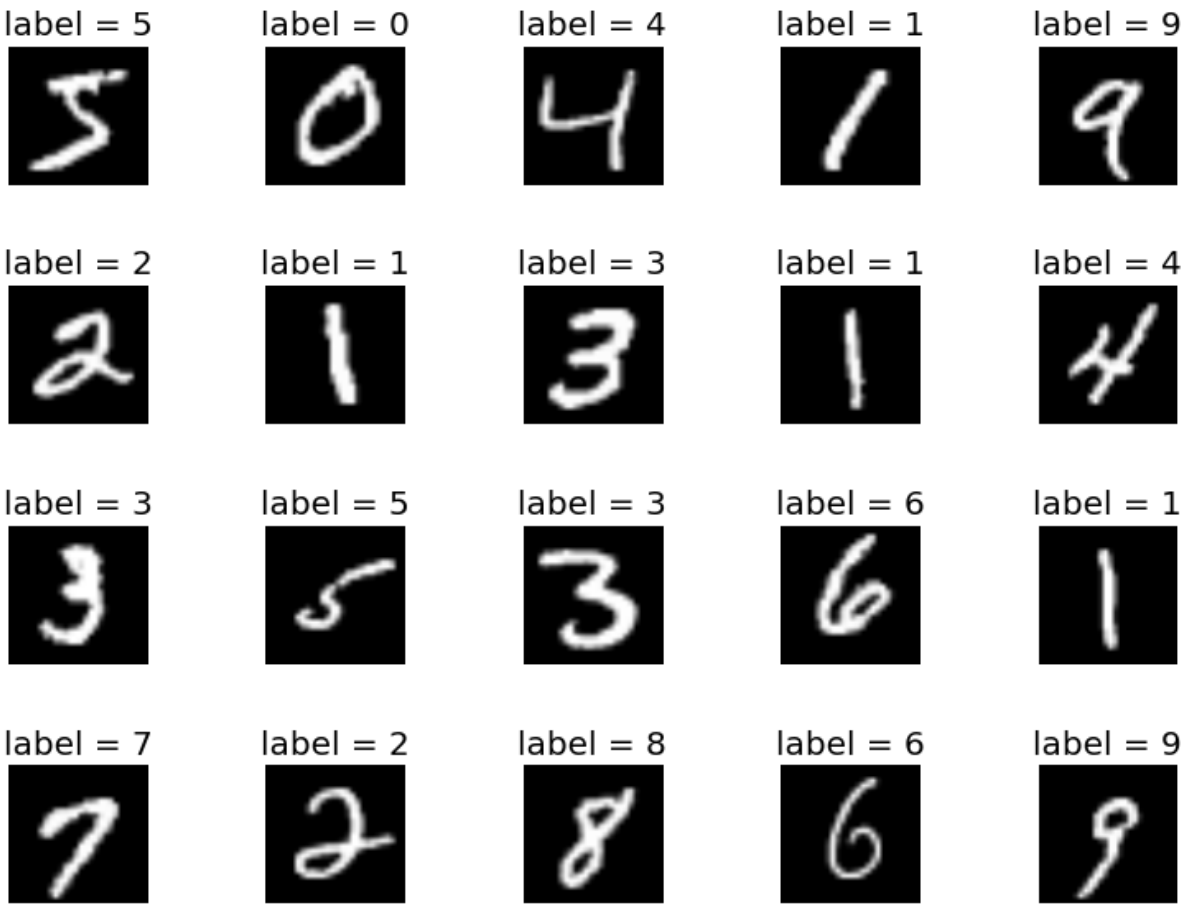
## 5. Chargement du mdoèle

```
function onLoadModel(){
    tf.loadLayersModel('localstorage://my-model-1’)
.then(m=>{
        model=m;
        let config={
            optimizer:tf.train.sgd(learningRate),
            loss:'meanSquaredError'
        }
        model.compile(config);
    });
}
```

# Pour Deep Learning avec CNN

```
async createCNNModel(){
    // Create the model
    let cnnModel=tf.sequential();
    // Add the First Convolution Layer
    cnnModel.add(tf.layers.conv2d({
      inputShape: [28, 28, 1],
      kernelSize: 5,
      filters: 8,
      strides: 1,
      activation: 'relu',
    }));
```

EMNIST Data Set : images de taille 28 x 28 pixels 1 bit



# Pour Deep Learning avec CNN

```
//Adding The Second Layer : MaxPooling Layer
   cnnModel.add(tf.layers.maxPooling2d({
     poolSize: [2, 2],
     strides: [2, 2]
   }));
   // Adding Another Convolutional Layer
   cnnModel.add(tf.layers.conv2d({
     kernelSize: 5,
     filters: 16,
     strides: 1,
     activation: 'relu',
   }));
```

# Pour Deep Learning avec CNN

```
//Adding Another MaxPooling Layer
   cnnModel.add(tf.layers.maxPooling2d({
     poolSize: [2, 2],
     strides: [2, 2]
   }));
```

```
// Adding A Flatten Layer
   cnnModel.add(tf.layers.flatten());
  // Adding A Dense Layer (Fully Connected Layer) For Performing The Final Classification
   cnnModel.add(tf.layers.dense({
     units: 10,
     kernelInitializer: 'VarianceScaling',
     activation: 'softmax'
   }));
```

# Pour Deep Learning avec CNN

```
// Compiling The Model
   cnnModel.compile({
     optimizer: tf.train.sgd(0.15),
     loss: 'categoricalCrossentropy'
   });
```

```
// Train the model
   let images=[[]];  let labels=[[]];
   let xs=tf.tensor2d(images);
   let ys=tf.tensor2d(labels)
   await cnnModel.fit(xs, labels, {
     batchSize: 60,
     epochs: 1
   });
   // Prediction
   let inputImage=[[]];
   let outputLabel=cnnModel.predict(inputImage);
   console.log(outputLabel);
}
```

# Application 1 : Iris Data Set

https://github.com/mohamedYoussfi/iris_tfjs_angular

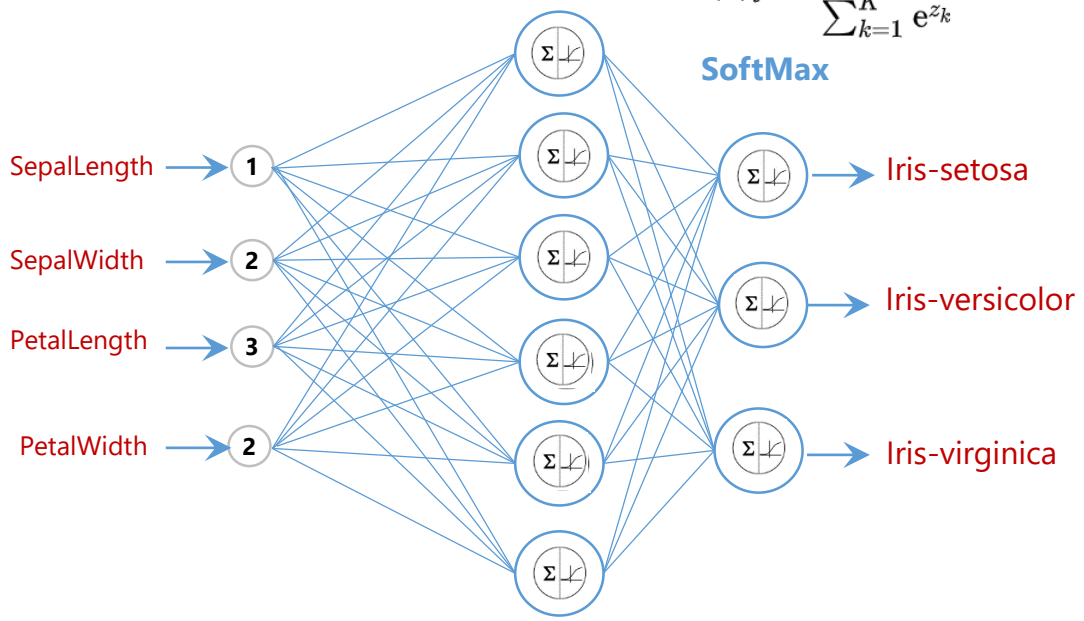
https://www.kaggle.com/uciml/iris

$$f(x) = \frac{1}{1 + e^{-x}}$$

Sigmoid

$$\sigma(\mathbf{z})_j = \frac{e^{z_j}}{\sum_{k=1}^{K} e^{z_k}}$$

SoftMax




Learning Rate : 0.001 Optimiser : ADAM

Loss function : MearnSquaredError

$$\text{MSE} = \frac{1}{n}\sum_{i=1}^{n}(y_i - \tilde{y}_i)^2$$

**Iris_train.csv (120 instances)**

```
5.1,3.5,1.4,0.2,0
4.9,3.0,1.4,0.2,0
4.7,3.2,1.3,0.2,0
…
6.4,3.2,4.5,1.5,1
6.9,3.1,4.9,1.5,1
5.5,2.3,4.0,1.3,1
…
6.3,3.3,6.0,2.5,2
5.8,2.7,5.1,1.9,2
7.1,3.0,5.9,2.1,2
```

**Iris_test.csv (30 instances)**

```
5.0,3.5,1.3,0.3,0
4.5,2.3,1.3,0.3,0
4.4,3.2,1.3,0.2,0
…
5.5,2.6,4.4,1.2,1
6.1,3.0,4.6,1.4,1
5.8,2.6,4.0,1.2,1
…
6.7,3.1,5.6,2.4,2
6.9,3.1,5.1,2.3,2
5.8,2.7,5.1,1.9,2
```


Iris-setosa.png


Iris-versicolor.pn


Iris-virginica.png

# Application : Iris Data Set

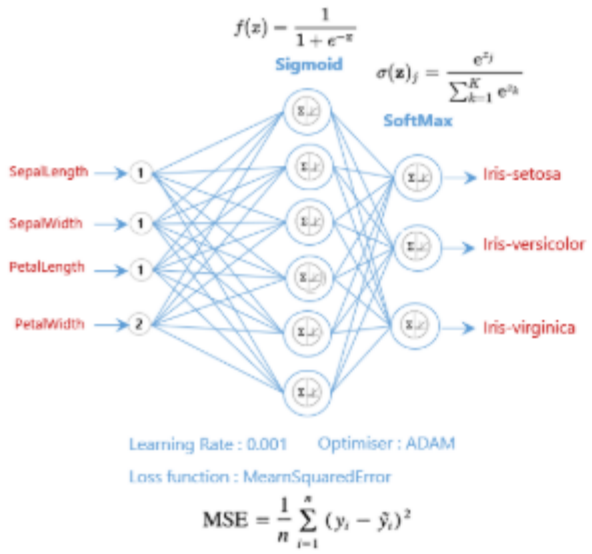
localhost:4200

Neural Net Model

https://github.com/mohamedYoussfi/iris_tfjs_angular

Create Model | Load Data | Train | Save Model | Load Model | Evaluate Model

99%




Prediction :**Iris-setosa**

**SepalLengthCm:**

5.1

**SepalWidthCm:**

3.5

**PetalLengthCm:**

1.4

**PetalWidthCm:**

0.2

Predict

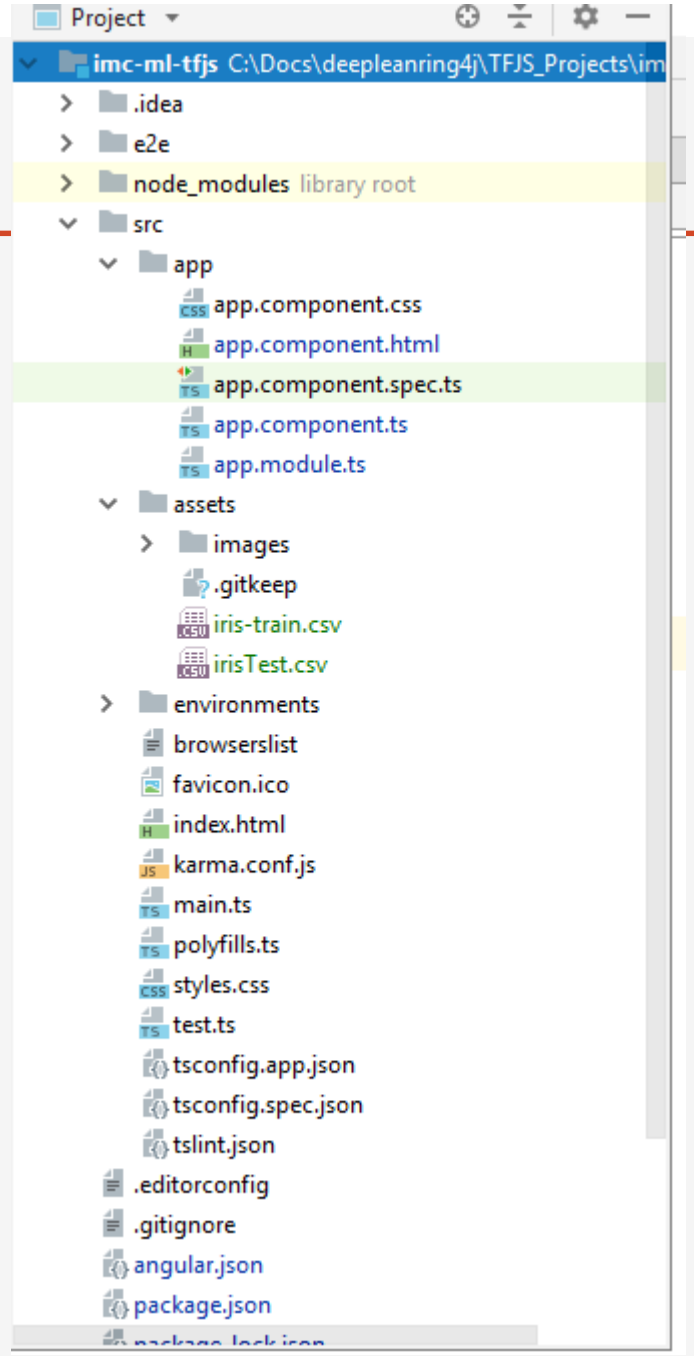
**Loss: 0.01703817769885063**

**Accuracy: 0.9749999642372131Evaluation: 100 %**

| SepalLengthCm | SepalWidthCm | PetalLengthCm | PetalWidthCm | Species |
|---|---|---|---|---|
| 5.1 | 3.5 | 1.4 | 0.2 | Iris-setosa |

# Application

- Outils à installer :
  - NodeJS (npm)
  - Git
  - Angular : > **`npm install -g @angular/cli`**
- Création d'un projet Angular :
  - **> ng new iris-ml-tfjs**
- **Démendances à installer :**
  - **TensorFlow.JS :**
    - **`> npm install --save @tensorflow/tfjs`**
  - **Framework CSS Bootstrap :**
    - **`> npm install --save bootstrap@3`**
  - **Angular Progress Bar :**
    - **`> npm install --save angular-progress-bar`**



# Application : dépendances et configuration

**package.json**

```
"dependencies": {
  "@angular/animations": "~7.2.0",
  "@angular/common": "~7.2.0",
  "@angular/compiler": "~7.2.0",
  "@angular/core": "~7.2.0",
  "@angular/forms": "~7.2.0",
  "@angular/platform-browser": "~7.2.0",
  "@angular/platform-browser-dynamic": "~7.2.0",
  "@angular/router": "~7.2.0",
  "@tensorflow/tfjs": "^1.0.2",
  "angular-progress-bar": "^1.0.9",
  "bootstrap": "^3.4.1",
  "core-js": "^2.5.4",
  "rxjs": "~6.3.3",
  "tslib": "^1.9.0",
  "zone.js": "~0.8.26"
}
```

**app.module.ts**

```
import { BrowserModule } from '@angular/platform-browser';
import { NgModule } from '@angular/core';
import { AppRoutingModule } from './app-routing.module';
import { AppComponent } from './app.component';
import {FormsModule} from '@angular/forms';
import {ProgressBarModule} from 'angular-progress-bar';
@NgModule({
  declarations: [
    AppComponent
  ],
  imports: [
    BrowserModule,
    AppRoutingModule, FormsModule, ProgressBarModule
  ],
  providers: [],
  bootstrap: [AppComponent]
})
export class AppModule { }
```

**angular.json**

```
"styles": [  "src/styles.css", "node_modules/bootstrap/dist/css/bootstrap.min.css" ]
```

# app.component.ts : Déclaration des attributs du composant web

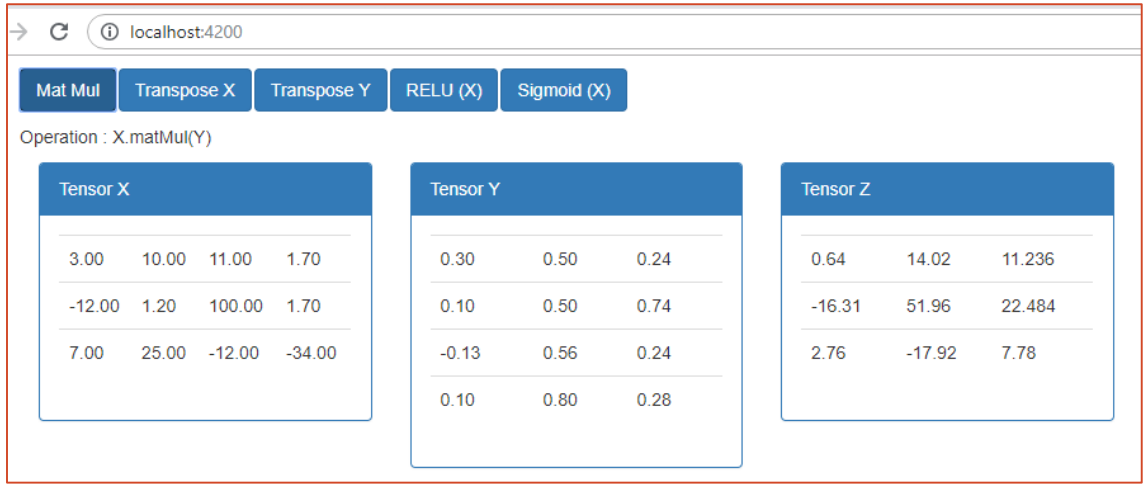
```
import {Component, OnInit} from '@angular/core';
import * as tf from '@tensorflow/tfjs';
import {HttpClient} from '@angular/common/http';
@Component({
  selector: 'app-root',
  templateUrl: './app.component.html',
  styleUrls: ['./app.component.css']
})
export class AppComponent implements OnInit{

  public tensors:tf.Tensor2D[]=[];    public operation:string;
  public model:tf.Sequential;   public learningRate:number=0.01;
  private modelCreated: boolean;  public xs:Array<number[]>=[];
  public ys:Array<number[]>=[];  public xsTest:Array<number[]>=[];
  public ysTest:Array<number[]>=[];
  public irisClasses=['Iris-setosa','Iris-versicolor','Iris-virginica'];
  currentEpoch: number=0;  currentAccuracy: number=0;
  currentLoss: number=0;  correctEval:number=0;
  wrongEval:number=0;

  example={SepalLengthCm:5.1,SepalWidthCm:3.5,PetalLengthCm:1.4,PetalWidthCm:0.2};
  prediction: any;
  epochs:number=100;

  constructor(private httpClient:HttpClient){}
```

# app.component.ts : les opérations sur les tensors

```
ngOnInit(): void {
  this.tensors['X']=tf.tensor([3,10,11,1.70,-12,1.20,100,1.70,7, 25,-12,-34],[3,4]);
  this.tensors['Y']=tf.tensor([[0.3,0.5,0.24],[0.1,0.5,0.74],[-0.13,0.56,0.24],
    [0.1,0.8,0.28]]);
  this.tensors['X'].print();
  this.tensors['Y'].print();
}
```

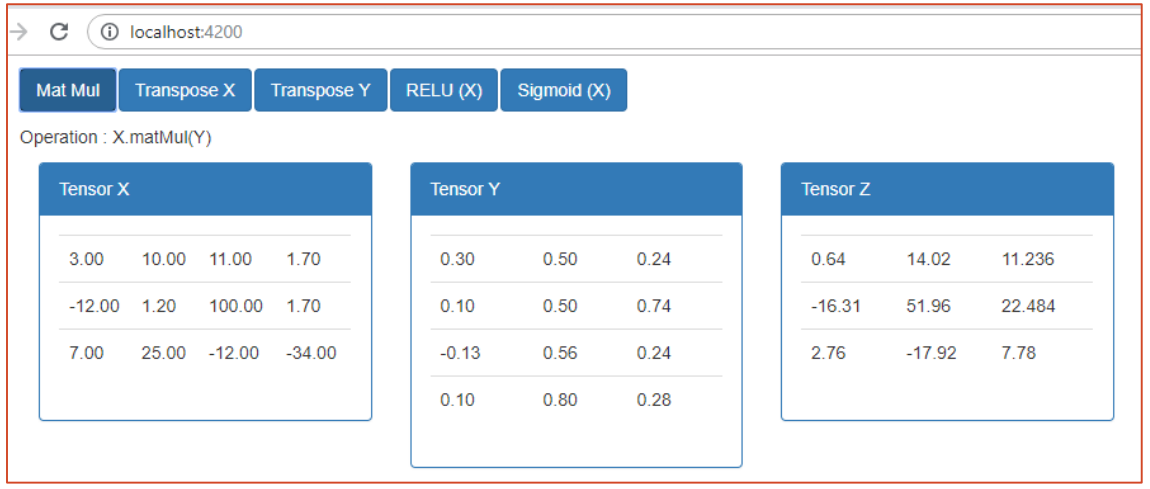
# app.component.ts : les opérations sur les tensors

```
onMult() {
  this.tensors['Z']=this.tensors['X'].matMul(this.tensors['Y']);
  this.operation="X.matMul(Y)";
}
onTranspose(t: string) {
  this.tensors['Z']=this.tensors[t].transpose();
  this.operation="X.transose()";
}
```



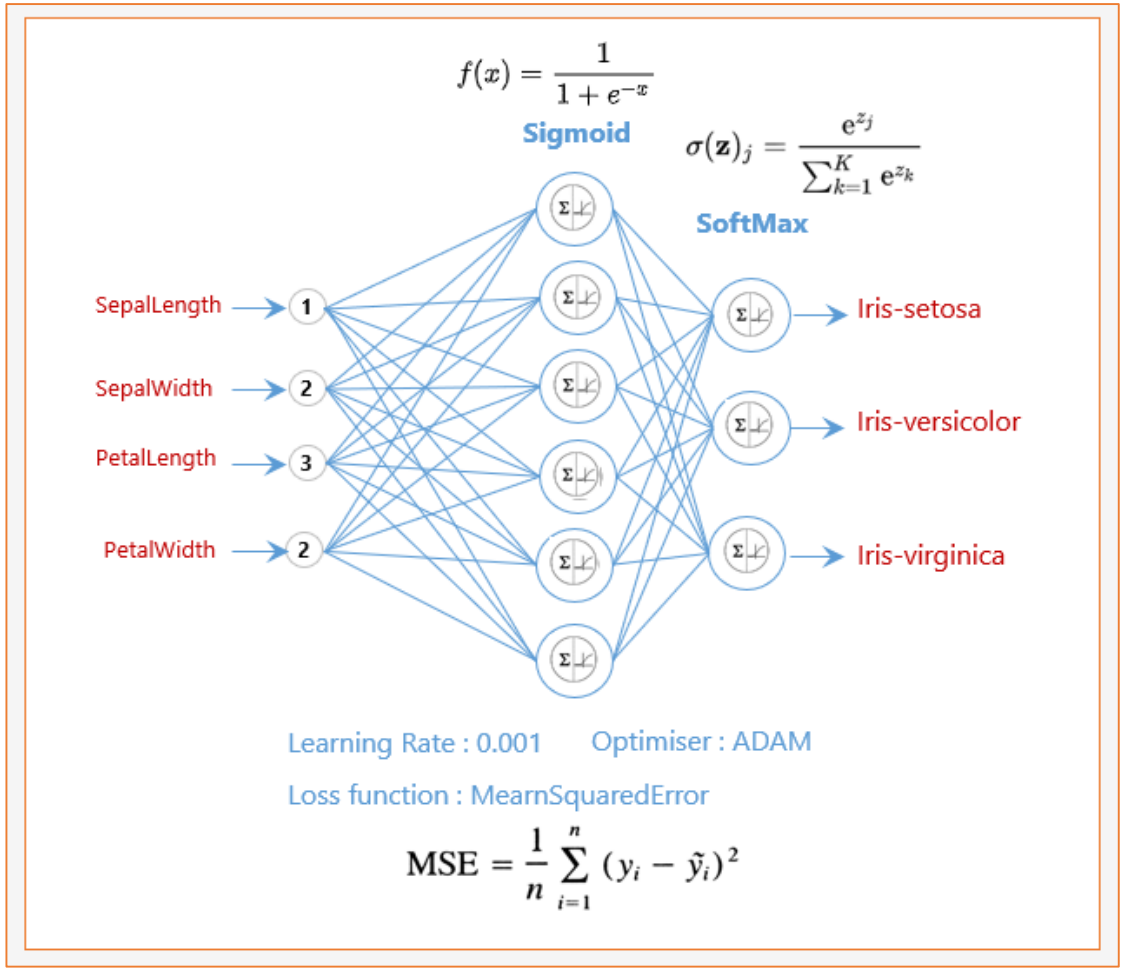
# app.component.ts : les opérations sur les tensors

```
onRelu(t: string) {
  this.tensors['Z']=tf.relu(this.tensors[t]);
  this.operation="X.relu()";
}
onSigmoid(t: string) {
  this.tensors['Z']=tf.sigmoid(this.tensors[t]);
  this.operation="X.sigmoid()";
}
```



# app.component.ts : Création du modèle de réseaux de neurones

```
onCreateModel() {
  this.model=tf.sequential();
  this.model.add(tf.layers.dense({
    units:10, inputShape:[4],activation:'sigmoid'
  }));
  this.model.add(tf.layers.dense({
    units:3,activation:'softmax'
  }));
  this.model.compile({
    optimizer:tf.train.adam(this.learningRate),
    loss:tf.losses.meanSquaredError,
    metrics:['accuracy']
  });
  this.modelCreated=true;
}
```



# app.component.ts : Préparation du Data Set

```
onLoadData() {
  this.httpClient.get("assets/iris-train.csv",{responseType:'text'})
    .subscribe(data=>{
      let parsedData=this.parseData(data);
      this.xs=parsedData.xs;
      this.ys=parsedData.ys;
    },err=>{
      console.log(err);
    }); this.httpClient.get("assets/irisTest.csv",{responseType:'text'})
    .subscribe(data=>{
      let parsedData=this.parseData(data);
      this.xsTest=parsedData.xs;
      this.ysTest=parsedData.ys;
    },err=>{
      console.log(err);
    }); }
```

```
5.1,3.5,1.4,0.2,0
4.9,3.0,1.4,0.2,0
4.7,3.2,1.3,0.2,0
…
6.4,3.2,4.5,1.5,1
6.9,3.1,4.9,1.5,1
5.5,2.3,4.0,1.3,1
…
6.3,3.3,6.0,2.5,2
5.8,2.7,5.1,1.9,2
7.1,3.0,5.9,2.1,2
```

# app.component.ts : Préparation du Data Set

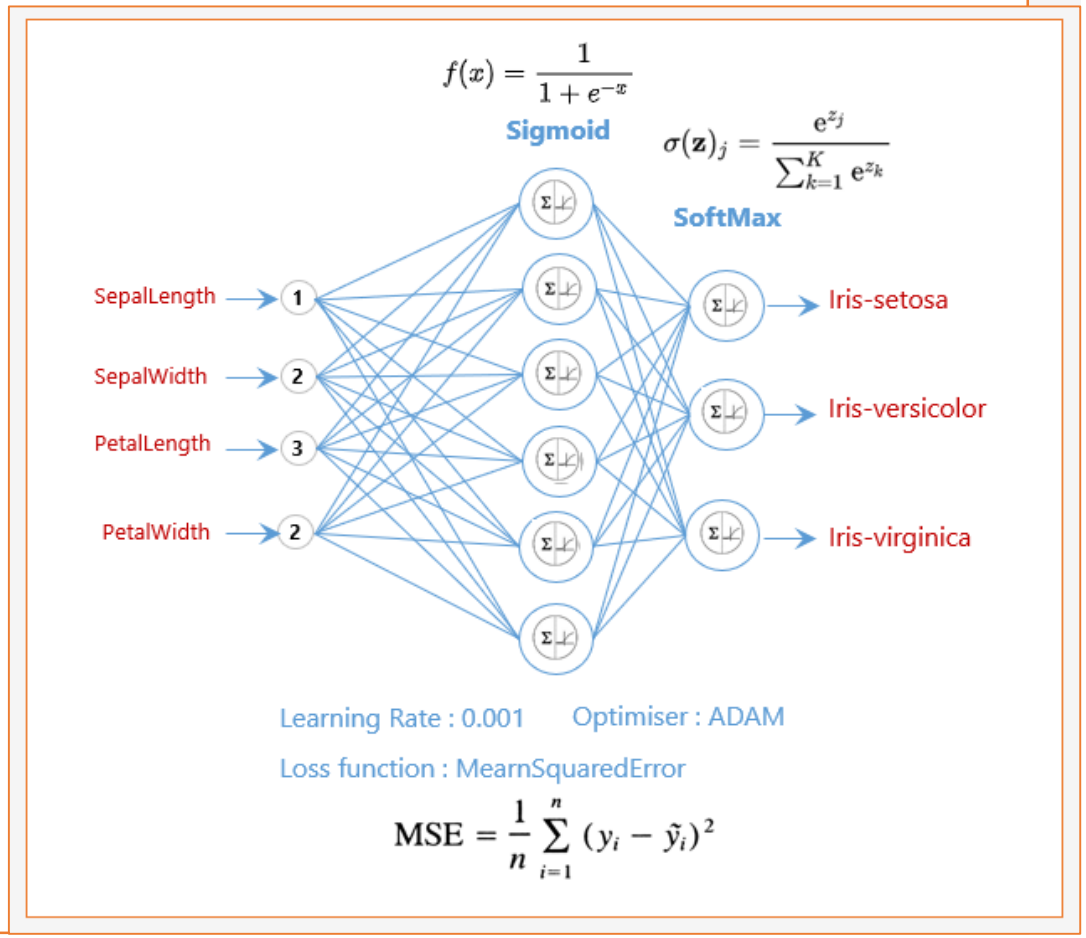
```
parseData(data){
  let inputs=[];  let outputs=[];
  let lines=data.split(/\n/);
  for (let i = 0; i < lines.length; i++) {
    if(lines[i]!==''){
      let line=lines[i].split(",");
      inputs.push([
        parseFloat(line[0]), parseFloat(line[1]),
        parseFloat(line[2]), parseFloat(line[3])]);
      let output=[0,0,0];output[parseInt(line[4])]=1;
      outputs.push(output);
    }
  }
  return {xs:inputs,ys:outputs };
}
```

```
5.1,3.5,1.4,0.2,0
4.9,3.0,1.4,0.2,0
4.7,3.2,1.3,0.2,0
…
6.4,3.2,4.5,1.5,1
6.9,3.1,4.9,1.5,1
5.5,2.3,4.0,1.3,1
…
6.3,3.3,6.0,2.5,2
5.8,2.7,5.1,1.9,2
7.1,3.0,5.9,2.1,2
```

```
getIrisClass(data){
  //let t=tf.tensor1d(data);
  let index=data.indexOf(1);
  return this.irisClasses[index];
}
```

# app.component.ts : Entrainer le modèle

```
onTrainModel() {
  const inputs:tf.Tensor2D=tf.tensor2d(this.xs);
  const targets:tf.Tensor2D=tf.tensor2d(this.ys);
  const inputsTest:tf.Tensor2D=tf.tensor2d(this.xsTest);
  const targetsTest:tf.Tensor2D=tf.tensor2d(this.ysTest);
    this.model.fit(inputs,targets,{
      epochs:this.epochs,
      validationData:[inputsTest,targetsTest],
      callbacks:{
        onEpochEnd:(epoch,logs)=>{
          this.currentEpoch=epoch;
          this.currentLoss=logs.loss;
          this.currentAccuracy=logs.acc;
        }
      }
    }).then(resp=>{    }); }
```



# app.component.ts : Save the model

```
onSaveModel() {
  this.model.save('localstorage://irisModel')
    .then(result=>{
      alert('Success saving model!');
    },err=>{
      alert('Error saving Model!');
    })
}
```

# app.component.ts : Load the model

```
onLoadModel() {
  tf.loadLayersModel('localstorage://irisModel')
    .then(m=>{
      this.model=m;
      this.model.compile({
        optimizer:tf.train.adam(this.learningRate),
        loss:tf.losses.meanSquaredError,
        metrics:['accuracy']
      });
      this.modelCreated=true;
      alert("Model loaded!");
    })
}
```

# app.component.ts : Evaluate the model

```
onEvalModel() {
  let inputsTest=tf.tensor2d(this.xsTest);
  let targetTest=tf.tensor2d(this.ysTest);
  let yTrue=targetTest.argMax(-1).dataSync();
  let predictions=this.model.predict(inputsTest);
  // @ts-ignore
  let yPredictions=predictions.argMax(-1).dataSync();
  this.correctEval=0;
  this.wrongEval=0;
  for (let i = 0; i < yPredictions.length; i++) {
    if(yTrue[i]==yPredictions[i]) ++this.correctEval
    else ++this.wrongEval;
  }}
```

# app.component.ts : Prediction

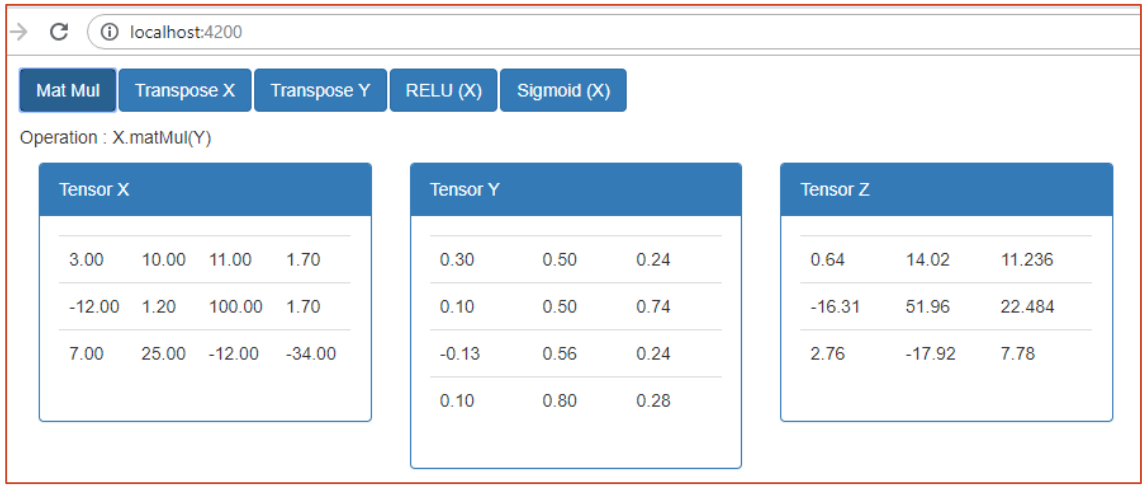
```
onPredict(value: any) {
    let x1=parseFloat(value.SepalLengthCm);
    let x2=parseFloat(value.SepalWidthCm);
    let x3=parseFloat(value.PetalLengthCm);
    let x4=parseFloat(value.PetalWidthCm);
    let input=tf.tensor2d([[x1,x2,x3,x4]]);
    const prediction=this.model.predict( input);
    // @ts-ignore
    let index=prediction.argMax(-1).dataSync()[0];
    this.prediction=this.irisClasses[index];
  }
}
```

# app.component.html : Boutons des opérations sur le tensors

```
<div class="container">
  <p></p>
  <div class="row">
    <ul class="nav nav-pills">
      <li> <button class="btn btn-primary" (click)="onMult()">Mat Mul</button>  </li>
      <li> <button class="btn btn-primary" (click)="onTranspose('X')">Transpose X</button>  </li>
      <li> <button class="btn btn-primary" (click)="onTranspose('Y')">Transpose Y</button> </li>
      <li> <button class="btn btn-primary" (click)="onRelu('X')">RELU (X)</button> </li>
      <li> <button class="btn btn-primary" (click)="onSigmoid('X')">Sigmoid (X)</button>  </li>
    </ul>
  </div>
  <p></p>
```

# app.component.html : Affichage des données des tensors

```
<div class="row">
    <p>   Operation : {{operation}} </p>
    <div class="col-md-5" *ngFor="let t of tensors | keyvalue">
      <div class="panel panel-primary">
        <div class="panel-heading">Tensor {{t.key}}</div>
        <div class="panel-body">
          <table class="table">
            <tr *ngFor="let line of t.value.arraySync()">
              <td *ngFor="let v of line">{{v|number:'0.2'}}</td>
            </tr>
          </table>
        </div>
      </div>
    </div>
  </div>
</div>
```



# app.component.html : Modèle

```
<div class="container">
  <div class="panel panel-default">
    <div class="panel-heading">Neural Net Model</div>
    <div class="panel-body">
      <ul class="nav nav-pills">
        <li><button class="btn btn-success" (click)="onCreateModel()">Create Model</button></li>
        <li><button class="btn btn-success" (click)="onLoadData()">Load Data</button></li>
        <li><button class="btn btn-success" (click)="onTrainModel()">Train</button></li>
        <li><button class="btn btn-success" (click)="onSaveModel()">Save Model</button></li>
        <li><button class="btn btn-success" (click)="onLoadModel()">Load Model</button></li>
        <li><button class="btn btn-success" (click)="onEvalModel()">Evaluate Model</button></li>
      </ul>
```

# app.component.html : Modèle

```
<div class="row">
  <div class="col-md-8">
    <progress-bar [progress]="currentEpoch" [color-degraded]="{'0': '#00cbcb',  '15': '#f9c3d3', '25': '#fd8c8e'}"></progress-bar>
  </div>
  <div class="col-md-4">
    <label>Loss: {{currentLoss}}</label>
    <label>Accuracy: {{currentAccuracy}}</label>
    <label>Evaluation: {{100*correctEval/(correctEval+wrongEval)}} %</label>
  </div>
</div>
```

# app.component.html : Modèle

```
<div class="row" *ngIf="modelCreated">
  <div class="row">
    <div class="col-md-4">
      <img src="assets/images/model.png" width="300" >
    </div>
    <div class="col-md-4">
      <div class="panel panel-default">
        <div class="panel-heading">Prediction :<strong>{{prediction}}</strong></div>
        <div class="panel-body">
          <form #f="ngForm" (ngSubmit)="onPredict(f.value)">
            <div class="form-group">
              <label class="control-label">SepalLengthCm:</label>
              <input class="form-control" type="text" name="SepalLengthCm" ngModel=""
[(ngModel)]="example.SepalLengthCm">
            </div>
```

# app.component.html : Modèle

```
<div class="form-group">
              <label class="control-label">SepalWidthCm:</label>
              <input class="form-control" type="text" name="SepalWidthCm" ngModel=""
[(ngModel)]="example.SepalWidthCm">
            </div>
            <div class="form-group">
              <label class="control-label">PetalLengthCm:</label>
              <input class="form-control" type="text" name="PetalLengthCm" ngModel=""
[(ngModel)]="example.PetalLengthCm">
            </div>
            <div class="form-group">
              <label class="control-label">PetalWidthCm:</label>
              <input class="form-control" type="text" name="PetalWidthCm" ngModel=""
[(ngModel)]="example.PetalWidthCm">
            </div>
            <button type="submit">Predict</button>
```

# app.component.html : Modèle

```
</form>
        </div>
      </div>
    </div>
    <div class="col-md-3">
      <img src="assets/images/{{prediction}}.png" width="200">
    </div>
  </div>
```

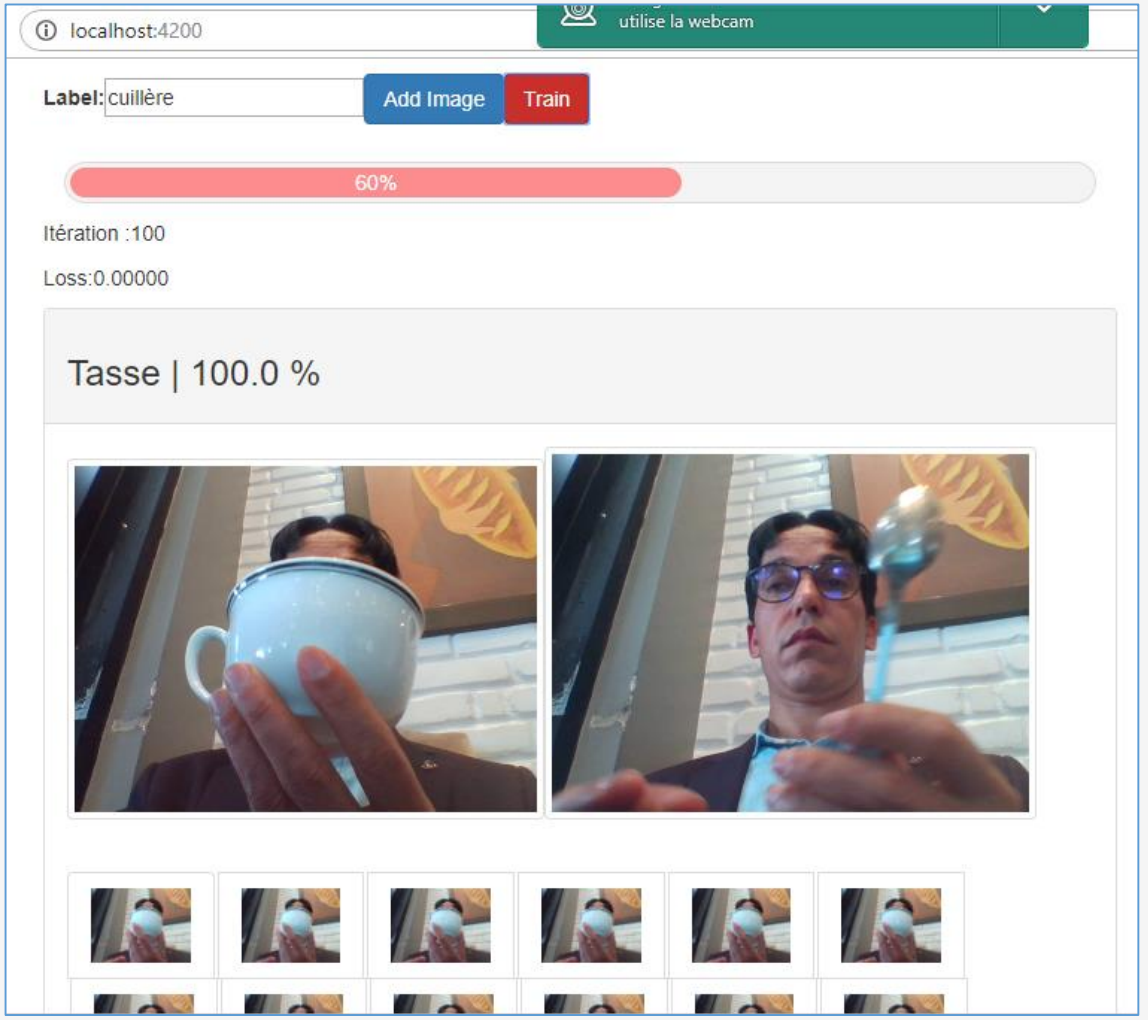
# app.component.html : Modèle

```
<div class="col-md-6">
          <table class="table" *ngIf="xs">
            <tr>
              <th>SepalLengthCm</th>    <th>SepalWidthCm</th>
              <th>PetalLengthCm</th>    <th>PetalWidthCm</th>
              <th>Species</th>
            </tr>
            <tr *ngFor="let d of xs; let i=index">
              <td>{{xs[i][0]}}</td>  <td>{{xs[i][1]}}</td>
              <td>{{xs[i][2]}}</td>  <td>{{xs[i][3]}}</td>
              <td>{{getIrisClass(ys[i])}}</td>
            </tr>
          </table>
        </div>  </div></div>
  </div>
</div>
```

https://github.com/mohamedYoussfi/angular-ml5.js-mobilenet-feature-extractor

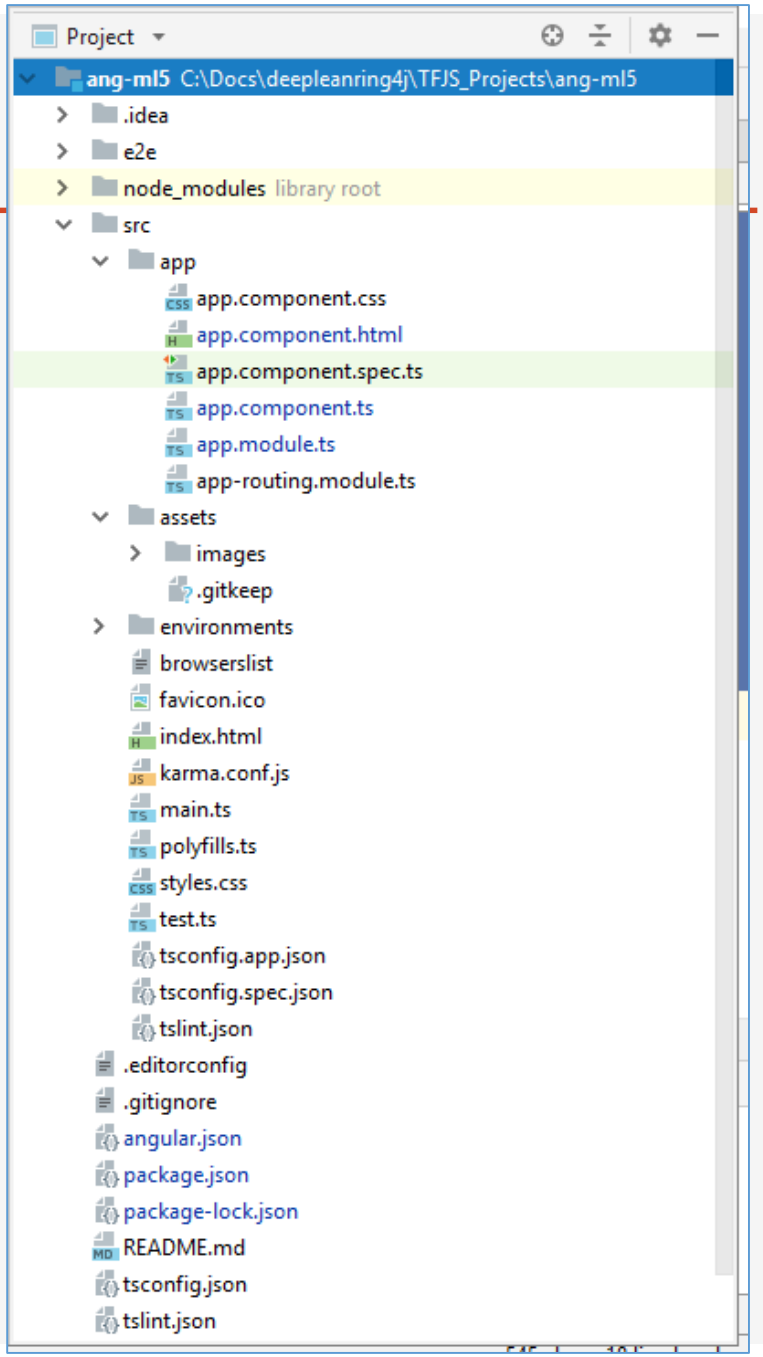
# Application 2 : Deep learning en utilisant ML5.JS et des modèles pré-entrainés

- Cette application montre comment créer d'extraction des caractéristiques « FeatureExtractor » à partir à partir des images capturée par une la caméra du PC.
- Cette application est basée sur TensorFlow.JS et **ML5.JS**.
- Elle charge un modèle pré-entrainé « **MobileNet** » utilisant les techniques de deep learning avec Convolution Neural Network (**CNN**)
- Une fois le modèle MobileNet chargé avec ML5.JS,
  - On présente des objet à la caméra
  - On on ajouter au modèle des images capturées de la caméra en leurs associant un label.
  - On lance l'entrainement du modèle.
  - Une fois entrainé,
    - On présente des objet à la caméra
    - L'application détecte de quel objet s'ajit-il.



# Application

- Outils à installer :
  - NodeJS (npm)
  - Git
  - Angular : > `npm install -g @angular/cli`
- Création d'un projet Angular :
  - **> ng new medico-ml-tfjs**
- **Démendances à installer :**
  - **TensorFlow.JS :**
    - `> npm install --save @tensorflow/tfjs`
  - **ML5.JS :**
    - `> npm install --save ml5`
  - **Framework CSS Bootstrap :**
    - `> npm install --save bootstrap@3`
  - **Angular Progress Bar :**
    - `> npm install --save angular-progress-bar`



# Application : dépendances et configuration

**package.json**

```
"dependencies": {
  "@angular/animations": "~7.2.0",
  "@angular/common": "~7.2.0",
  "@angular/compiler": "~7.2.0",
  "@angular/core": "~7.2.0",
  "@angular/forms": "~7.2.0",
  "@angular/platform-browser": "~7.2.0",
  "@angular/platform-browser-dynamic": "~7.2.0",
  "@angular/router": "~7.2.0",
  "angular-progress-bar": "^1.0.9",
  "bootstrap": "^3.4.1",
  "core-js": "^2.5.4",
  "ml5": "^0.2.2",
  "rxjs": "~6.3.3",
  "tslib": "^1.9.0",
  "zone.js": "~0.8.26"
}
```

**app.module.ts**

```
import { BrowserModule } from '@angular/platform-browser';
import { NgModule } from '@angular/core';
import { AppRoutingModule } from './app-routing.module';
import { AppComponent } from './app.component';
import {FormsModule} from '@angular/forms';
import {ProgressBarModule} from 'angular-progress-bar';

@NgModule({
  declarations: [
    AppComponent
  ],
  imports: [
    BrowserModule,
    AppRoutingModule, FormsModule, ProgressBarModule
  ],
  providers: [],
  bootstrap: [AppComponent]
})
export class AppModule { }
```

**angular.json**

```
"styles": [  "src/styles.css", "node_modules/bootstrap/dist/css/bootstrap.min.css" ]
```

# app.component.ts : Déclaration des attributs du composant

```
import {AfterViewInit, Component, ElementRef, NgZone, OnInit, ViewChild} from '@angular/core';
declare let ml5: any;
@Component({
  selector: 'app-root',
  templateUrl: './app.component.html',
  styleUrls: ['./app.component.css']
})
export class AppComponent implements OnInit, AfterViewInit {
  public mobileNetFeatureExtractor;  public featureClassifier;
  public label;  public confidence;
  public newLabel;  public currentProgress = 0;
  public loss: number;  public iteration: number;
  @ViewChild('video')  public video: ElementRef;
  @ViewChild('canvas')  public canvas: ElementRef;
  public captures: Array<any>;
  constructor(private zone: NgZone) {
    this.captures = [];
  }
```

# app.component.ts : Charger le modèle MobileNet en utilisant featureExtractor de ML5

```
ngOnInit(): void {
   this.mobileNetFeatureExtractor = ml5.featureExtractor('MobileNet', () => {
this.featureClassifier = this.mobileNetFeatureExtractor.classification(this.video.nativeElement, () => {
      console.log('Vidéo ready');
    });
  });
}
```

# app.component.ts : Ajouter une image au modèle capturée par la caméra

```
addImage() {
    this.featureClassifier.addImage(this.newLabel);
    this.capture();
 }
```

# app.component.ts : Entrainer le modèle

```
train() {
    this.iteration = 0; this.loss = 0;    this.currentProgress = 0;
    this.featureClassifier.train((loss) => {
      if (loss == null) {
        this.iteration = 100;
        this.mobileNetFeatureExtractor.classify((e, r) => {
          this.gotResults(e, r);
        });
      } else {
        this.zone.run(() => {
          ++this.currentProgress; ++this.iteration; this.loss = loss;
        });
      }
    });
  }
```

# app.component.ts : Afficher la capture de la améra

```
public ngAfterViewInit() {
    if ( navigator.mediaDevices && navigator.mediaDevices.getUserMedia) {
      navigator.mediaDevices.getUserMedia({ video: true })
        .then(stream => {
                    this.video.nativeElement.srcObject = stream;
                    this.video.nativeElement.play();
      });
    }
  }
```

# app.component.ts : Faire une capture d'une image avec la UserCam

```
public capture() {
    const context = this.canvas.nativeElement.getContext('2d’)
       .drawImage(this.video.nativeElement, 0, 0, 320, 240);
    this.captures.push(this.canvas.nativeElement.toDataURL('image/png'));
  }
```

# app.component.ts : Prédiction de l'objet présenté à la caméra

```
gotResults(err, results) {
    if (err) {   console.log(err); } else {
      this.zone.run(() => {
        this.label = results[0].label;
        this.confidence = results[0].confidence;
      });
      this.mobileNetFeatureExtractor.classify((e, r) => {
        this.gotResults(e, r);
      });
    }
  }
}
```

## app.component.html:

```
<p></p>
<div class="container">
<div class="form-group">
  <label class="control-label">Label:</label>
  <input type="text" [(ngModel)]="newLabel">
  <button (click)="addImage()" class="btn btn-primary">Add Image</button>
  <button (click)="train()" class="btn btn-danger">Train</button>
</div>
```

# app.component.html: Progress Bar et affichage de itération et loss

```
<div class="row">
    <div class="col-md-4">
      <progress-bar [progress]="currentProgress" [color-degraded]="{'0': '#00cbcb',  '15': '#f9c3d3', '25': '#fd8c8e'}"></progress-bar>
    </div>
    <div class="col-md-4">
      <p>Itération :{{iteration}}</p>
      <p>Loss:{{loss}}</p>
    </div>
  </div>
```

# app.component.html: Affichage de la WebCam et des prédictions label et confidence

```
<div>
  <div class="panel panel-default">
    <div class="panel-heading">
      <h3>{{label}}    |    {{confidence*100 | number : '0.1'}} %</h3>
    </div>
  <div class="panel-body">
    <video #video id="video" width="320" height="240" autoplay class="img-thumbnail"></video>
    <canvas #canvas id="canvas" width="320" height="240" class="img-thumbnail"></canvas>
  </div>
```

# app.component.html: Affichage de la liste des images capturées

```
<div class="panel-body">
    <ul class="nav nav-pills">
      <li *ngFor="let c of captures" class="list-group-item">
        <img src="{{ c }}" height="50"  />
      </li>
    </ul>
  </div>
  </div>
</div>
<div>
</div>
</div>
```

# Deeplearning4J : Machines et Deep Learning four les applications Java

- **Deep Learning 4j** est une Framework open source (licence Apache) qui permet de construire, entraîner et tester une grande diversité d'algorithmes de Deep Learning (depuis les réseaux standard, jusqu'aux réseaux à convolutions, en passant par des architectures plus complexes).

- Il se base sur sa structure de données (Nd4j) permettant d'effectuer les opérations de l'algèbre linéaires sur les architectures massivement parallèles GPU et les architectures distribuées

- **Nd4j** utilise du code natif ( Cuda oblige), et alloue de l'espace or du tas Java. Ceci est impérativement à prendre en compte lorsque la volumétrie des données est importante.

- DL4J utilise **DataVec** pour la vectorisation et la transformation des données.

# Nd4J

- **ND4J** est une bibliothèque de calcul scientifique et d'algèbre linéaire, écrite en langage de programmation Java et compatible avec d'autres langages tels que Scala et Cotlin. Le ND4J a été versé à la fondation Eclipse en octobre 2017.

- ND4J permet de réaliser des manipulations des calculs linéaires et matriciel dans un environnement de production.
- Elle s'intègre à Apache Hadoop et **Spark** pour fonctionner avec des unités centrales de traitement (UC) ou des unités de traitement graphiques (**GPU**). Il prend en charge les tableaux n-dimensionnels pour les langages basés sur JVM.

- ND4J est un logiciel gratuit et à code source ouvert, publié sous Apache License 2.0, et développé principalement par le groupe basé à San Francisco qui a créé Deeplearning4j, sous une licence Apache.

# Dépendances Maven

```
<!- Cœur de DL4J -->
<dependency>
    <groupId>org.deeplearning4j</groupId>
    <artifactId>deeplearning4j-core</artifactId>
    <version1.0.0-beta3</version>
</dependency>
<!- ND4J Pour basée sur CUDA  pour les GPUs-->
<dependency>
    <groupId>org.nd4j</groupId>
    <artifactId>nd4j-cuda-8.0-
platform</artifactId>
    <version1.0.0-beta3</version>
</dependency>
```

```
<!- ND4J Natif pour CPU -->
<dependency>
    <groupId>org.nd4j</groupId>
    <artifactId>nd4j-native-platform</artifactId>
    <version>1.0.0-beta3</version>
</dependency>
<!- User Interface de DL4J -->
<dependency>
    <groupId>org.deeplearning4j</groupId>
    <artifactId>deeplearning4j-ui_2.11</artifactId>
    <version>1.0.0-beta3</version>
</dependency>

<dependency>
    <groupId>org.slf4j</groupId>
    <artifactId>slf4j-simple</artifactId>
    <version>1.6.1</version>
</dependency>
```

# Dépendances Maven

- Selon la documentation de DL4J, ce POM configure la librairie pour utiliser les deux « backends » (CPU & GPU) disponibles.
- Ainsi, dès que le premier est utilisé à 100 %, alors le second backend est utilisé.
- Si on souhaite prioriser un des deux backends alors on peut fixer les deux variables d'environnement suivantes :
  - **BACKEND_PRIORITY_CPU**
  - **BACKEND_PRIORITY_GPU.**
- Il est cependant bien plus pratique de simplement retirer du POM le backend que l'on ne souhaite pas utiliser.

# ND4J

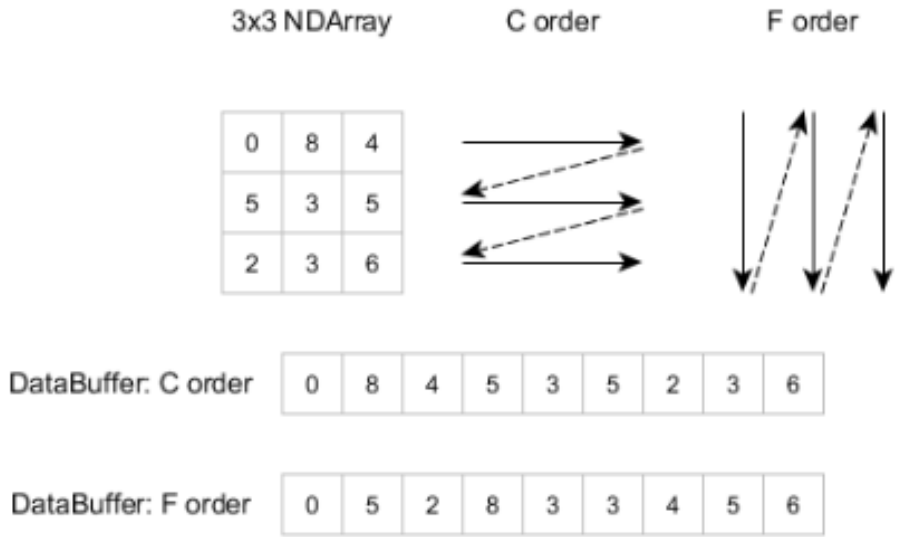
- ND4J est une bibliothèque de calcul scientifique et d'algèbre linéaire, écrite en langage de programmation Java, fonctionnant sur la machine virtuelle Java (JVM) et compatible avec d'autres langages tels que Scala et Clojure.
- ND4J a été à la base de la contribution de la fondation Eclipse en octobre 2017.
- ND4J permet de réaliser des manipulations de calcul linéaire et matriciel dans un environnement de production.
- Il s'intègre à Apache Hadoop et Spark pour fonctionner avec des unités centrales de traitement (UC) ou des unités de traitement graphiques massivement parallèle (GPU).
- Il prend en charge les tableaux n-dimensionnels pour les langages basés sur JVM.
- ND4J est une laibririe Open source, publié sous Apache License 2.0, et développé principalement par le groupe basé à San Francisco qui a créé Deeplearning4j.
- Les opérations de ND4J incluent des versions parallèles distribuées.
- Les opérations peuvent se dérouler dans un cluster et traiter d'énormes quantités de données.
- La manipulation matricielle se produit en parallèle sur les processeurs ou les GPU sur le cloud computing et peut fonctionner dans les clusters Apache Spark ou Hadoop.

# Utilisation de ND4J : Maven dependencies

```
<dependencies>
    <!-- https://mvnrepository.com/artifact/org.nd4j/nd4j-native-platform -->
    <dependency>
        <groupId>org.nd4j</groupId>
        <artifactId>nd4j-native-platform</artifactId>
        <version>1.0.0-beta3</version>
    </dependency>
    <!-- https://mvnrepository.com/artifact/org.nd4j/nd4j-cuda-9.2-platform -->
    <dependency>
        <groupId>org.nd4j</groupId>
        <artifactId>nd4j-cuda-9.2-platform</artifactId>
        <version>1.0.0-beta3</version>
    </dependency>
    <dependency>
        <groupId>org.slf4j</groupId>
        <artifactId>slf4j-simple</artifactId>
        <version>1.6.1</version>
    </dependency>
</dependencies>
```

# Utilisation de ND4J

- En termes d'encodage, un NDArray peut être encodé dans l'ordre C (row-major) ou Fortran (column-major).
- Nd4J peut utiliser simultanément une combinaison de tableaux d'ordres C et F.
- La plupart des utilisateurs peuvent simplement utiliser l'ordre de tableau par défaut, mais notez qu'il est possible d'utiliser un ordre spécifique pour un tableau donné, le cas échéant.




- In the above array, we have:
  - **Shape** = [3,3] (3 rows, 3 columns)
  - **Rank** = 2 (2 dimensions)
  - **Length** = 9 (3x3=9)
  - **Stride**
    - C order stride: [3,1]: the values in consecutive rows are separated in the buffer by 3, and the values consecutive columns are separated in the buffer by 1
    - F order stride: [1,3]: the values in consecutive rows are separated in the buffer by 1, and the values in consecutive columns are separated in the buffer by 3

# Utilisation de ND4J : Exemple 1 : Création des NDArray

```
INDArray tens1= Nd4j.ones(3,2);
INDArray tens2=Nd4j.zeros(3,2);
INDArray tens3=Nd4j.rand(new int[]{3,2});
INDArray tens4=Nd4j.zeros(3,2).add(10);
INDArray tens5=Nd4j.create(new
double[]{6,7,9,4,5,1},new int[]{3,2},'c');
System.out.println("****************");
System.out.println(tens1.toString());
System.out.println("****************");
System.out.println(tens2.toString());
System.out.println("****************");
System.out.println(tens3.toString());
System.out.println("****************");
System.out.println(tens4.toString());
System.out.println("****************");
System.out.println(tens5.toString());
```

```
****************
[[    1.0000,    1.0000],
 [    1.0000,    1.0000],
 [    1.0000,    1.0000]]
****************
[[         0,         0],
 [         0,         0],
 [         0,         0]]
****************
[[   10.0000,   10.0000],
 [   10.0000,   10.0000],
 [   10.0000,   10.0000]]
****************
[[    0.3125,    0.3767],
 [    0.4294,    0.7398],
 [    0.2499,    0.4250]]
****************
[[    6.0000,    7.0000],
 [    9.0000,    4.0000],
 [    5.0000,    1.0000]]
```

# Utilisation de ND4J : Exemple 2 Multiplication de deux matrices

```
Nd4j.setDataType(DataBuffer.Type.DOUBLE);

INDArray t1=Nd4j.create(new
double[][]{{5,7},{8,9},{3,4},{1,3}});
System.out.println("t1,"+t1.shapeInfoToString());
System.out.println(t1);
INDArray t2=Nd4j.create(new
double[][]{{5,7,5},{8,9,5}});
System.out.println("t2,"+t2.shapeInfoToString());
System.out.println(t2);
INDArray t3=t1.mmul(t2);
System.out.println("t3,"+t3.shapeInfoToString());
System.out.println(t3);
```

```
t1,Rank: 2,Offset: 0
 Order: c Shape: [4,2],  stride: [2,1]
[[    5.0000,    7.0000],
 [    8.0000,    9.0000],
 [    3.0000,    4.0000],
 [    1.0000,    3.0000]]
t2,Rank: 2,Offset: 0
 Order: c Shape: [2,3],  stride: [3,1]
[[    5.0000,    7.0000,    5.0000],
 [    8.0000,    9.0000,    5.0000]]
t3,Rank: 2,Offset: 0
 Order: f Shape: [4,3],  stride: [1,4]
[[   81.0000,   98.0000,   60.0000],
 [  112.0000,  137.0000,   85.0000],
 [   47.0000,   57.0000,   35.0000],
 [   29.0000,   34.0000,   20.0000]]
```

# Utilisation de ND4J , Création des NDArray => vstack, hstack

- Création de NDArrays à partir d'autres NDArrays
- Il existe trois méthodes principales pour créer des NDArrays à partir d'autres table NDArrays:
  - Création d'une copie exacte d'un NDArray existant à l'aide de INDArray.dup ()
  - Créer le tableau en tant que sous-ensemble d'un NDArray existant
  - Combinez plusieurs NDArrays existants pour créer un nouveau NDArray

```
**********
[[ 1.0000, 1.0000,      0,          0],
 [ 1.0000, 1.0000,      0,          0],
 [ 1.0000,    1.0000,   0,          0]]
**********
[[    1.0000,    1.0000],
 [    1.0000,    1.0000],
 [    1.0000,    1.0000],
 [         0,         0],
 [         0,         0],
 [         0,         0]]
```

```
INDArray tens6=Nd4j.hstack(tens1,tens2);

INDArray tens7=Nd4j.vstack(tens1,tens2);

System.out.println("**********");

System.out.println(tens6.toString());

System.out.println("**********");

System.out.println(tens7.toString());
```

# Utilisation de ND4J : Création des NDArray => combine

```
int nRows = 2;
        int nColumns = 2;
//INDArray of zeros
        INDArray zeros = Nd4j.zeros(nRows, nColumns);
// Create one of all ones
        INDArray ones = Nd4j.ones(nRows, nColumns);
// Concat on dimension 0
        INDArray combined = Nd4j.concat(0,zeros,ones);
  System.out.println("### COMBINED dimension 0####");
        System.out.println(combined);
//Concat on dimension 1
        INDArray combined2 = Nd4j.concat(1,zeros,ones);
  System.out.println("### COMBINED dimension 1 ####");
        System.out.println(combined2);
```

```
### COMBINED dimension 0####
[[0.00, 0.00],
 [0.00, 0.00],
 [1.00, 1.00],
 [1.00, 1.00]]
### COMBINED dimension 1 ####
[[0.00, 0.00, 1.00, 1.00],
 [0.00, 0.00, 1.00, 1.00]]
```

# Utilisation de ND4J : Création des NDArray => pad

```
 int nRows = 2;
int nColumns = 2;
// Create INDArray of all ones
  INDArray ones = Nd4j.ones(nRows, nColumns);
// pad the INDArray
  INDArray padded = Nd4j.pad(ones, new
int[]{1,1}, Nd4j.PadMode.CONSTANT );
        System.out.println("### Padded ####");
        System.out.println(padded);
```

```
### Padded ####
[[0.00, 0.00, 0.00, 0.00],
 [0.00, 1.00, 1.00, 0.00],
 [0.00, 1.00, 1.00, 0.00],
 [0.00, 0.00, 0.00, 0.00]]
```

# Utilisation de ND4J : Création des NDArray => diag

```
INDArray a=Nd4j.rand(new int []{3,3});
INDArray b=Nd4j.diag(a);
INDArray c=Nd4j.diag(b);
System.out.println("**********a=");
System.out.println(a);
System.out.println("**********b=");
System.out.println(b);
System.out.println("**********c=");
System.out.println(c);
```

```
**********a=

[[    0.1007,    0.0651,    0.7231],
 [    0.7003,    0.6176,    0.3784],
 [    0.0006,    0.3673,    0.4608]]

**********b=

[    0.1007,    0.6176,    0.4608]

**********c=

[[    0.1007,         0,         0],
 [         0,    0.6176,         0],
 [         0,         0,    0.4608]]
```

# Utilisation de ND4J : Création des NDArray => eye(), linespace(),

```
INDArray identityMatrix=Nd4j.eye(3);
System.out.println(identityMatrix);
/*
To create a row vector with elements [a, a+1,
a+2, ..., b]
you can use the linspace command:
Nd4j.linspace(a, b, b-a+1)
 */
INDArray data=Nd4j.linspace(0,8,9).reshape(new
int[]{3,3});
 System.out.println(data.toString());
```

```
[[    1.0000,         0,         0],
 [         0,    1.0000,         0],
 [         0,         0,    1.0000]]


[[         0,    1.0000,    2.0000],
 [    3.0000,    4.0000,    5.0000],
 [    6.0000,    7.0000,    8.0000]]
```

# Utilisation de ND4J : Création des NDArray => get(), put(),

```
INDArray data=Nd4j.rand(new int[]{3,3});
data.putScalar(1,1,10);
System.out.println(data);


NdIndexIterator iterator=new
NdIndexIterator(new int[]{3,3});
while (iterator.hasNext()){
   long[] index=iterator.next();
    System.out.println(data.getDouble(index));
}
```

```
[[    0.4446,    0.3177,    0.1541],
 [    0.0768,   10.0000,    0.0525],
 [    0.4882,    0.6861,    0.6925]]

0.4446452856063843
0.3176351793289185
0.1541199831546783
0.0768395885825157 2
10.0
0.0525495335459709 2
0.488236963748931 9
0.686082780361175 5
0.692496120929718
```

# Utilisation de ND4J : Création des NDArray => getRow() and putRow()

```
INDArray data=Nd4j.rand(new int[]{3,3});
INDArray d1=Nd4j.ones(3);
data.putRow(0,d1);
System.out.println(data);
System.out.println("**************");
System.out.println(data.getRow(2));
```

```
[[    1.0000,    1.0000,    1.0000],
 [    0.1713,    0.9745,    0.8285],
 [    0.4688,    0.4689,    0.0496]]

**************

[[    0.4688,    0.4689,    0.0496]]
```

# Utilisation de ND4J : Création des NDArray => tensorAlongDimension

```
INDArray t=Nd4j.create(new double[]{5,7,8,9,3,4,1,3,5},new
int[]{3,3});
 System.out.println(t);
INDArray t1=t.tensorAlongDimension(0,0);
INDArray t2=t.tensorAlongDimension(1,0);
 INDArray t3=t.tensorAlongDimension(2,0);
 INDArray t4=t.tensorAlongDimension(0,1);
 INDArray t5=t.tensorAlongDimension(1,1);
 INDArray t6=t.tensorAlongDimension(2,1);
 System.out.println("***********"); System.out.println(t1);
 System.out.println("***********"); System.out.println(t2);
 System.out.println("***********"); System.out.println(t3);
 System.out.println("***********"); System.out.println(t4);
 System.out.println("***********"); System.out.println(t5);
 System.out.println("***********"); System.out.println(t6);
```

```
[[    5.0000,    7.0000,    8.0000],
 [    9.0000,    3.0000,    4.0000],
 [    1.0000,    3.0000,    5.0000]]
***********
[[    5.0000,    9.0000,    1.0000]]
***********
[[    7.0000,    3.0000,    3.0000]]
***********
[[    8.0000,    4.0000,    5.0000]]
***********
[[    5.0000,    7.0000,    8.0000]]
***********
[[    9.0000,    3.0000,    4.0000]]
***********
[[    1.0000,    3.0000,    5.0000]]
```

# Utilisation de ND4J : Operations

- ND4J définit :
  - 5 catégories d'oprations qui peuvent être appliquées aux tensors :
    - Scalar
    - Transform
    - Accumulation
    - Index Accumulation
    - Broadcast
  - Et deux méthodes execution pour chacune:
    - Sur la totalité d'un INDArray,
    - Ou sur une dimention d'un INDArray
  - Les opérations peuvent agir de deux manières "in-place" qui modifie le contenu du tensor ou "Copy" qui crée une copie résultat de l'opération.
  - Par exemple pour additionner deux INDArrays. Nd4j ddéfinie deux méthodes :
    - `INDArray.add(INDArray)`
    - `INDArray.addi(INDArray).`

# Utilisation de ND4J : Scalar Ops

```
INDArray t=Nd4j.create(new
double[]{5,7,8,9,3,4,1,3,5},new int[]{3,3});
System.out.println(t);
t.addi(10);
INDArray t2=t.mul(2);
INDArray t3=t2.div(10);
INDArray t4=Nd4j.getExecutioner().execAndReturn(new
ScalarAdd(t,100));
System.out.println(t);
System.out.println(t2);
System.out.println(t3);
System.out.println(t4);
```

```
[[    5.0000,    7.0000,    8.0000],
 [    9.0000,    3.0000,    4.0000],
 [    1.0000,    3.0000,    5.0000]]

[[  115.0000,  117.0000,  118.0000],
 [  119.0000,  113.0000,  114.0000],
 [  111.0000,  113.0000,  115.0000]]

[[   30.0000,   34.0000,   36.0000],
 [   38.0000,   26.0000,   28.0000],
 [   22.0000,   26.0000,   30.0000]]

[[    3.0000,    3.4000,    3.6000],
 [    3.8000,    2.6000,    2.8000],
 [    2.2000,    2.6000,    3.0000]]

[[  115.0000,  117.0000,  118.0000],
 [  119.0000,  113.0000,  114.0000],
 [  111.0000,  113.0000,  115.0000]]
```

# Utilisation de ND4J : Transform Ops

```
INDArray t=Nd4j.create(new
double[]{5,7,8,9,3,4,1,3,5},new int[]{3,3});



INDArray tanh=Nd4j.getExecutioner().execAndReturn(new
Sigmoid(t));



System.out.println(t);
System.out.println(tanh);
```

```
[[    0.9933,    0.9991,    0.9997],
 [    0.9999,    0.9526,    0.9820],
 [    0.7311,    0.9526,    0.9933]]


[[    0.9933,    0.9991,    0.9997],
 [    0.9999,    0.9526,    0.9820],
 [    0.7311,    0.9526,    0.9933]]
```

# Utilisation de ND4J : Accumulation (Reduction) Ops

```
INDArray t=Nd4j.create(new double[]{5,7,8,9,3,4,1,3,5},new
int[]{3,3});
System.out.println(t);
Accumulation sum=Nd4j.getExecutioner().execAndReturn(new Sum(t));
System.out.println(sum.toString()+"="+sum.getFinalResult().doubleVa
lue());
Accumulation max=Nd4j.getExecutioner().execAndReturn(new Max(t));
System.out.println(max.toString()+"="+max.getFinalResult().doubleVa
lue());
System.out.println("SUM="+t.sumNumber().doubleValue());
System.out.println("MAX="+t.maxNumber().doubleValue());
INDArray sumColumns=t.sum(0);
INDArray suRows=t.sum(1);
System.out.println(sumColumns);
System.out.println(suRows);
```

```
[[    5.0000,    7.0000,    8.0000],
 [    9.0000,    3.0000,    4.0000],
 [    1.0000,    3.0000,    5.0000]]

reduce_sum=45.0

reduce_max=9.0

SUM=45.0

MAX=9.0

[[   15.0000,   13.0000,   17.0000]]

[20.0000,
 16.0000,
 9.0000]
```

# Utilisation de ND4J : Index Accumulation Ops

```
INDArray t=Nd4j.create(new
double[]{5,7,8,9,3,4,1,3,5},new int[]{3,3});
System.out.println(t);


INDArray argMaxCols=t.argMax(0);
INDArray argMaxRows=t.argMax(1);


System.out.println(argMaxCols);
System.out.println(argMaxRows);
```

```
[[    5.0000,    7.0000,    8.0000],
 [    9.0000,    3.0000,    4.0000],
 [    1.0000,    3.0000,    5.0000]]

[[    1.0000,         0,         0]]

[2.0000,
 0,
 2.0000]
```

# Utilisation de ND4J : Broadcast and Vector Ops

```
Nd4j.setDataType(DataBuffer.Type.DOUBLE);
INDArray t=Nd4j.create(new
double[]{5,7,8,9,3,4,1,3,5},new int[]{3,3});
System.out.println(t);
INDArray vector=Nd4j.create(new double[]{3,7,5});
System.out.println(vector);
INDArray res=t.addRowVector(vector);
System.out.println(res);


System.out.println(res.dataType());
System.out.println(res.data());
```

```
[[    5.0000,    7.0000,    8.0000],
 [    9.0000,    3.0000,    4.0000],
 [    1.0000,    3.0000,    5.0000]]


[[    3.0000,    7.0000,    5.0000]]


[[    8.0000,   14.0000,   13.0000],
 [   12.0000,   10.0000,    9.0000],
 [    4.0000,   10.0000,   10.0000]]

DOUBLE

[8.0,14.0,13.0,12.0,10.0,9.0,4.0,10.0,10.0]
```

# Utilisation de ND4J : Sérialisation

```
INDArray t1=Nd4j.create(new double[]{5,7,8,9,3,4,1,3,5},new
int[]{3,3});
System.out.println(t1);

//Binary format
DataOutputStream sWrite = new DataOutputStream(new
FileOutputStream(new File("tmp.bin")));
Nd4j.write(t1, sWrite);
sWrite.close();

DataInputStream sRead = new DataInputStream(new
FileInputStream(new File("tmp.bin")));
INDArray t2 = Nd4j.read(sRead);
sRead.close();
System.out.println(t2);

//Text format
Nd4j.writeTxt(t1, "data.txt");
t2 = Nd4j.readTxt("tmp.txt");
// To read csv format:
INDArray t3=Nd4j.readNumpy("data.csv", ", ");
System.out.println(t3);
```

**data.csv**

```
6, 7, 8
3, 4, 0
2, 1, 4
```

```
[[    5.0000,    7.0000,    8.0000],
 [    9.0000,    3.0000,    4.0000],
 [    1.0000,    3.0000,    5.0000]]

[[    5.0000,    7.0000,    8.0000],
 [    9.0000,    3.0000,    4.0000],
 [    1.0000,    3.0000,    5.0000]]

[[    6.0000,    7.0000,    8.0000],
 [    3.0000,    4.0000,         0],
 [    2.0000,    1.0000,    4.0000]]
```

**tmp.txt**

```
{
"filefrom": "dl4j",
"ordering": "c",
"shape": [1, 10],
"data":
[[   1.000000000000000000E0,
2.000000000000000000E0,   3.000000000000000000E0,
4.000000000000000000E0,   5.000000000000000000E0,
6.000000000000000000E0,   7.000000000000000000E0,
8.000000000000000000E0,   9.000000000000000000E0,
1.000000000000000000E1]]
}
```

# Application 2 avec DL4L : Iris Data Set

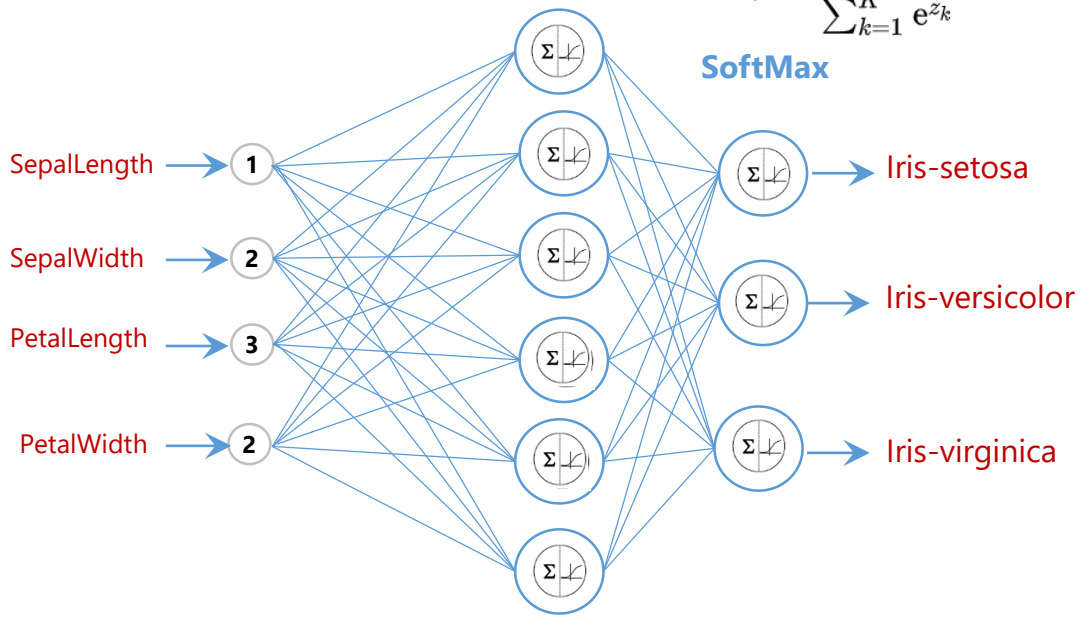
https://github.com/mohamedYoussfi/iris-dl4j-javafx

$$f(x) = \frac{1}{1 + e^{-x}}$$

Sigmoid

$$\sigma(\mathbf{z})_j = \frac{e^{z_j}}{\sum_{k=1}^{K} e^{z_k}}$$

SoftMax




Learning Rate : 0.001 Optimiser : ADAM

Loss function : MearnSquaredError

$$\text{MSE} = \frac{1}{n}\sum_{i=1}^{n}(y_i - \tilde{y}_i)^2$$

**Iris_train.csv (120 instances)**

```
5.1,3.5,1.4,0.2,0
4.9,3.0,1.4,0.2,0
4.7,3.2,1.3,0.2,0
…
6.4,3.2,4.5,1.5,1
6.9,3.1,4.9,1.5,1
5.5,2.3,4.0,1.3,1
…
6.3,3.3,6.0,2.5,2
5.8,2.7,5.1,1.9,2
7.1,3.0,5.9,2.1,2
```

**Iris_test.csv (30 instances)**

```
5.0,3.5,1.3,0.3,0
4.5,2.3,1.3,0.3,0
4.4,3.2,1.3,0.2,0
…
5.5,2.6,4.4,1.2,1
6.1,3.0,4.6,1.4,1
5.8,2.6,4.0,1.2,1
…
6.7,3.1,5.6,2.4,2
6.9,3.1,5.1,2.3,2
5.8,2.7,5.1,1.9,2
```


Iris-setosa.png

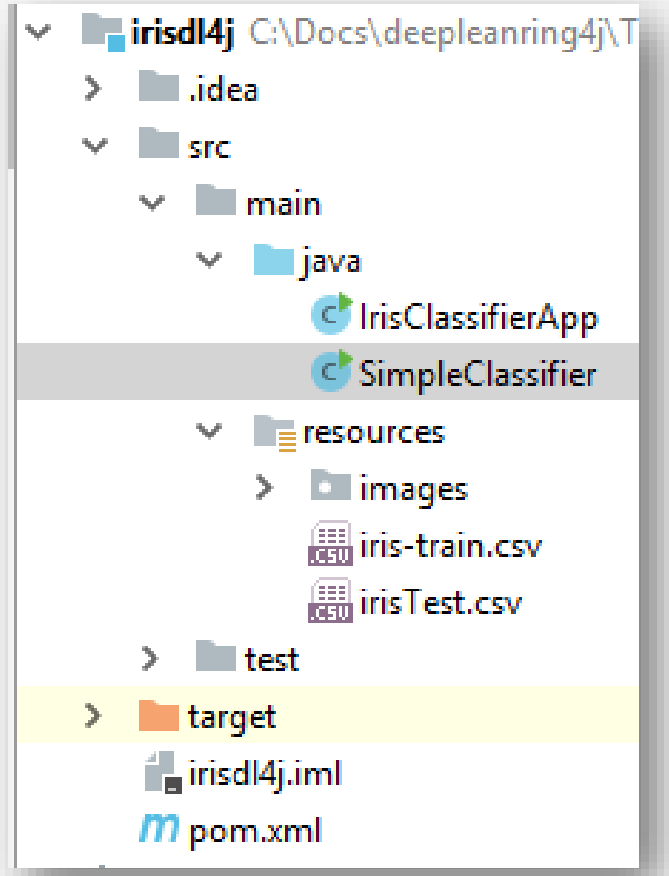

Iris-versicolor.pn


Iris-virginica.png

# Application : Classifieur Multi-classes :

**Paramètres:**

```
int batchSize=1;   int outputSize=3;   int classIndex=4;
double learninRate=0.001;
int inputSize=4;    int numHiddenNodes=10;
int nEpochs=100;
String[] labels={"Iris-setosa","Iris-versicolor","Iris-
virginica"};
```



# Application : Classifieur Multi-classes :

## Création du modèle

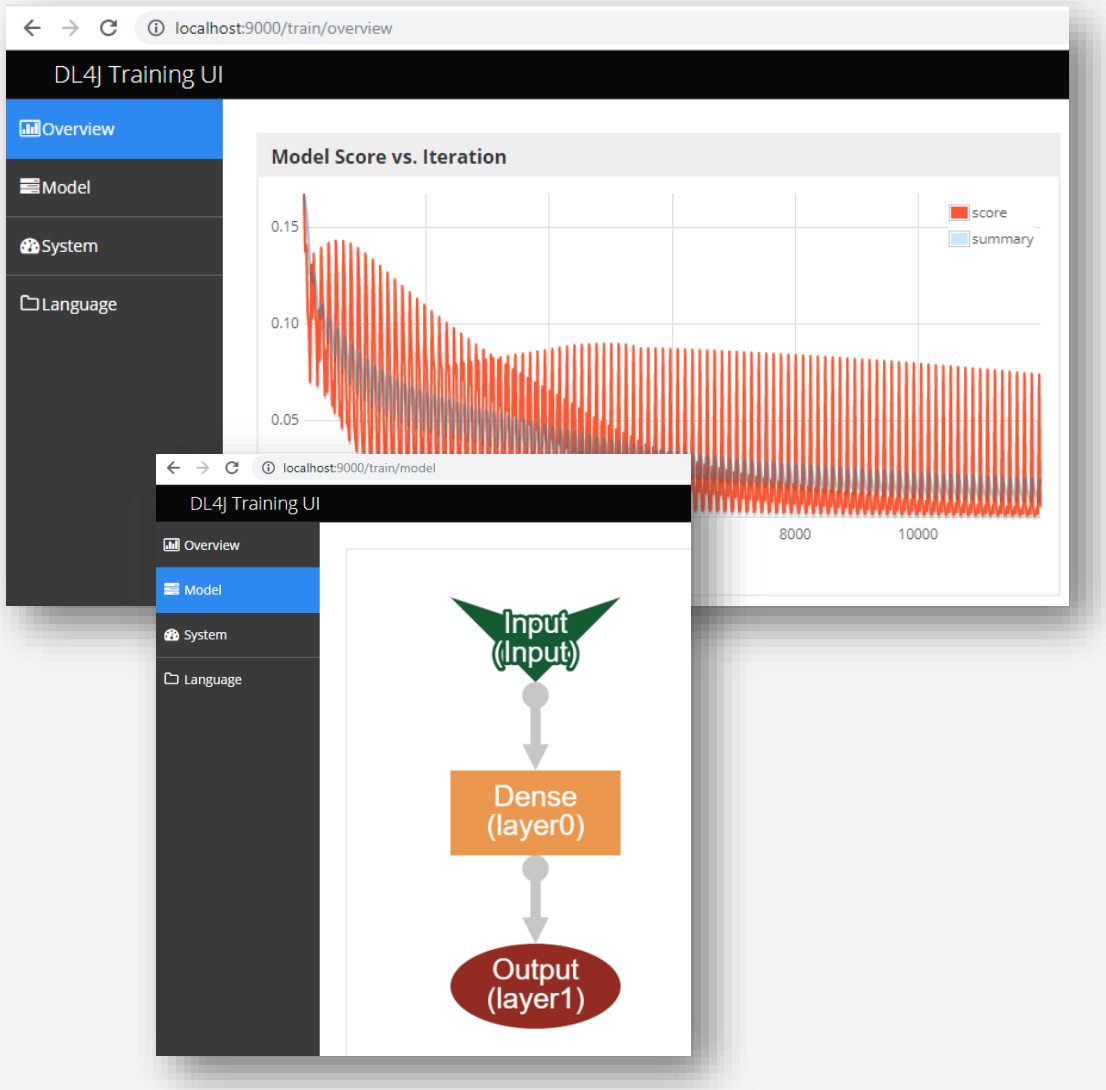
```
MultiLayerConfiguration configuration=new NeuralNetConfiguration.Builder()
        .seed(123)
        .updater(new Adam(learninRate))
        .list()
        .layer(0,new DenseLayer.Builder()
                .nIn(inputSize).nOut(numHiddenNodes).activation(Activation.SIGMOID).build())
        .layer(1,new OutputLayer.Builder()
                .nIn(numHiddenNodes).nOut(outputSize)
                .lossFunction(LossFunctions.LossFunction.MEAN_SQUARED_LOGARITHMIC_ERROR)
                .activation(Activation.SOFTMAX).build())
        .build();
MultiLayerNetwork model=new MultiLayerNetwork(configuration);
model.init();
```

# Application : Classifieur Multi-classes :

http://localhost:9000

## Entrainement du modèle :

```
UIServer uiServer=UIServer.getInstance();
InMemoryStatsStorage inMemoryStatsStorage=new
InMemoryStatsStorage();
uiServer.attach(inMemoryStatsStorage);
File fileTrain=new ClassPathResource("iris-
train.csv").getFile();
RecordReader recordReaderTrain=new CSVRecordReader();
recordReaderTrain.initialize(new FileSplit(fileTrain));
DataSetIterator dataSetIteratorTrain=
        new
RecordReaderDataSetIterator(recordReaderTrain,batchSize,class
Index,outputSize);
for (int i = 0; i <nEpochs ; i++) {
    model.fit(dataSetIteratorTrain);
}
```



# Application : Classifieur Multi-classes :

## Evaluation du modèle :

```
System.out.println("Model Evaluation");
File fileTest=new ClassPathResource("irisTest.csv").getFile();
RecordReader recordReaderTest=new CSVRecordReader();
recordReaderTest.initialize(new FileSplit(fileTest));
DataSetIterator dataSetIteratorTest=
        new RecordReaderDataSetIterator(recordReaderTest,batchSize,classIndex,outputSize);
Evaluation evaluation=new Evaluation(outputSize);

while (dataSetIteratorTest.hasNext()){
    DataSet dataSet = dataSetIteratorTest.next();
    INDArray features=dataSet.getFeatures();
    INDArray labels=dataSet.getLabels();
    INDArray predicted=model.output(features);
    evaluation.eval(labels,predicted);
}
System.out.println(evaluation.stats());
```

```
Model Evaluation
================Evaluation
Metrics===============
 # of classes:    3
 Accuracy:        1,0000
 Precision:       1,0000
 Recall:          1,0000
 F1 Score:        1,0000
Precision, recall & F1: macro-
averaged (equally weighted avg.
of 3 classes)

=====Confusion Matrix========
 0 1 2
-------
10   0   0 | 0 = 0
 0  10   0 | 1 = 1
 0  0   10 | 2 = 2
```

# Application : Classifieur Multi-classes :

## Prédiction :

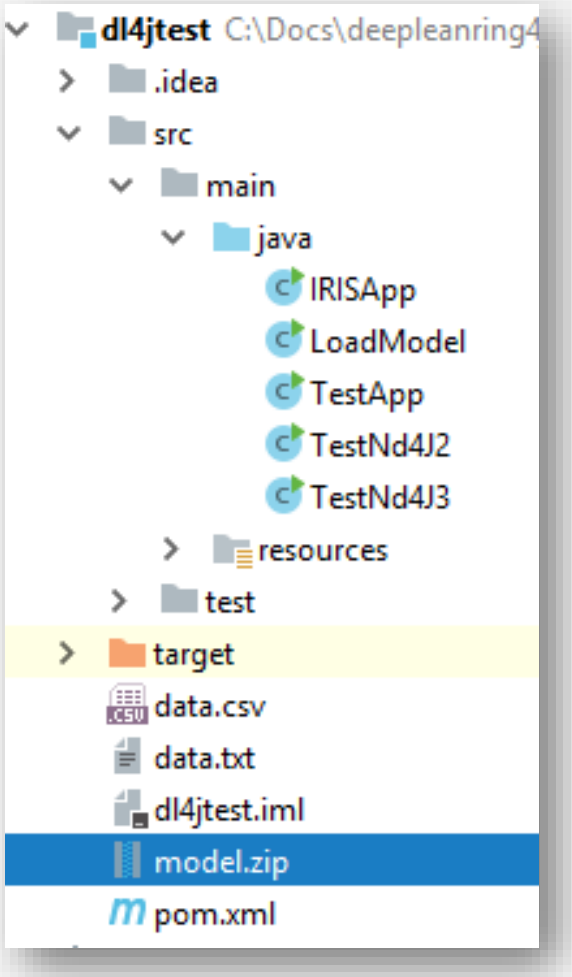
```
System.out.println("Prédiction");
INDArray input= Nd4j.create(new double[][]{
        {5.1,3.5,1.4,0.2}, {4.9,3.0,1.4,0.2},
        {6.7,3.1,4.4,1.4}, {5.6,3.0,4.5,1.5},
        {6.0,3.0,4.8,1.8}, {6.9,3.1,5.4,2.1}
});
System.out.println("**************");
INDArray output=model.output(input);
INDArray classes=output.argMax(1);
System.out.println(output);
System.out.println("-----------------");
System.out.println(classes);
System.out.println("***************");

int[] predictions=classes.toIntVector();
for (int i = 0; i < predictions.length; i++) {
    System.out.println(labels[predictions[i]]);
}
```

```
Prédiction
**************
[[    0.9880,    0.0120, 4.8603e-7],
 [    0.9852,    0.0148, 6.8171e-7],
 [    0.0119,    0.9657,    0.0224],
 [    0.0008,    0.6436,    0.3556],
 [ 9.1299e-5,    0.2578,    0.7421],
 [  1.543e-5,    0.1004,    0.8996]]
-----------------
[0,
 0,
 1.0000,
 1.0000,
 2.0000,
 2.0000]
***************
Iris-setosa
Iris-setosa
Iris-versicolor
Iris-versicolor
Iris-virginica
Iris-virginica
```

# Application : Classifieur Multi-classes :

**Saving the model :**

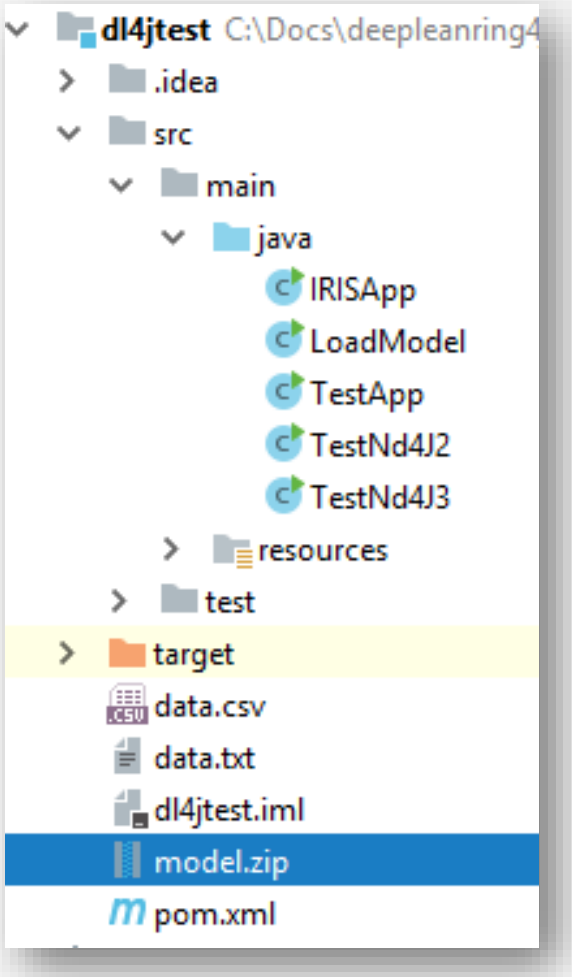
```
System.out.println("Saving Model:");

ModelSerializer.writeModel(model,new File("model.zip"),true);
```



# Application : Classifieur Multi-classes :

**Saving the model :**

```
System.out.println("Saving Model:");

ModelSerializer.writeModel(model,new File("model.zip"),true);
```



# Application : Classifieur Multi-classes :

## Loading the model :

```
System.out.println("Loading Model");
MultiLayerNetwork model= ModelSerializer.restoreMultiLayerNetwork(new
File("model.zip"));
System.out.println("Prédiction");
String[] labels={"Iris-setosa","Iris-versicolor","Iris-virginica"};
INDArray input= Nd4j.create(new double[][]{
        {5.1,3.5,1.4,0.2}
});
INDArray output=model.output(input);
int classIndex =output.argMax(1).getInt(0);
System.out.println(labels[classIndex]);
```

```
Prédiction
Iris-setosa
```

# Application Complète avec Interface graphique

```
import javafx.application.Application;
import javafx.application.Platform;
import javafx.beans.property.SimpleObjectProperty;
import javafx.geometry.Insets;
import javafx.scene.Scene;
import javafx.scene.control.*;
import javafx.scene.image.Image;
import javafx.scene.image.ImageView;
import javafx.scene.layout.BorderPane;
import javafx.scene.layout.GridPane;
import javafx.scene.layout.HBox;
import javafx.scene.layout.VBox;
import javafx.scene.web.WebEngine;
import javafx.scene.web.WebView;
import javafx.stage.Stage;
import org.datavec.api.records.reader.RecordReader;
import org.datavec.api.records.reader.impl.csv.CSVRecordReader;
import org.datavec.api.split.FileSplit;
import org.deeplearning4j.datasets.datavec.RecordReaderDataSetIterator;
import org.deeplearning4j.nn.conf.MultiLayerConfiguration;
import org.deeplearning4j.nn.conf.NeuralNetConfiguration;
import org.deeplearning4j.nn.conf.layers.DenseLayer;
import org.deeplearning4j.nn.conf.layers.OutputLayer;
import org.deeplearning4j.nn.multilayer.MultiLayerNetwork;
```

```
import org.deeplearning4j.nn.weights.WeightInit;
import org.deeplearning4j.optimize.listeners.ScoreIterationListener;
import org.deeplearning4j.ui.api.UIServer;
import org.deeplearning4j.ui.stats.StatsListener;
import org.deeplearning4j.ui.storage.InMemoryStatsStorage;
import org.nd4j.evaluation.classification.Evaluation;
import org.nd4j.linalg.activations.Activation;
import org.nd4j.linalg.api.ndarray.INDArray;
import org.nd4j.linalg.api.ops.impl.loss.MeanSquaredErrorLoss;
import org.nd4j.linalg.dataset.DataSet;
import org.nd4j.linalg.dataset.api.iterator.DataSetIterator;
import org.nd4j.linalg.factory.Nd4j;
import org.nd4j.linalg.io.ClassPathResource;
import org.nd4j.linalg.learning.config.Adam;
import org.nd4j.linalg.lossfunctions.LossFunctions;
import java.io.File;
import java.io.FileInputStream;
import java.io.IOException;
import java.text.DecimalFormat;
import java.text.NumberFormat;
import java.util.ArrayList;
import java.util.Collection;
import java.util.HashMap;
import java.util.Map;
```

# Application Complète avec Interface graphique

```
public class IrisClassifierApp extends Application {
    int batchSize=1;   int outputSize=3;   int classIndex=4; double learninRate=0.001;
    int inputSize=4;    int numHiddenNodes=10;
    MultiLayerNetwork model;
    int nEpochs=100;
    private double progressValue=0;
    InMemoryStatsStorage inMemoryStatsStorage;
    String[] featuresLabels=new String[]{"SepalLength","SepalWidth","PetalLength","PetalWidth"};
    String [] labels=new String[]{"Iris-setosa","Iris-versicolor","Iris-virginica"};
    Collection<Map<String,Object>> trainingDataSet=new ArrayList<>();
    Collection<Map<String,Object>> testDataSet=new ArrayList<>();

    public static void main(String[] args) throws Exception {
        launch();
    }
```

# Application Complète avec Interface graphique

```
@Override
    public void start(Stage primaryStage) throws Exception {
        BorderPane borderPane=new BorderPane();
        HBox hBoxTop=new HBox(10); hBoxTop.setPadding(new Insets(10));
        Button buttonCreateModel=new Button("Create Model");
        Button buttonLoadData=new Button("Load data");
        Button buttonTrain=new Button("Train Model");
        Button buttonEval=new Button("Evaluate Model");
        Button buttonPrection=new Button("Predict");
        Button buttonSaveModel=new Button("Save Model");
        Button buttonLoadModel=new Button("Load Model");


hBoxTop.getChildren().addAll(buttonCreateModel,buttonLoadData,buttonTrain,buttonEval,buttonPrection,buttonSaveMod
el,buttonLoadModel);
        borderPane.setTop(hBoxTop);
```

# Application Complète avec Interface graphique

```
Image image=new Image(new FileInputStream(new ClassPathResource("images/model.png").getFile()));
        ImageView imageView=new ImageView(image);



        TabPane tabPane=new TabPane();
       WebView webView=new WebView();
        WebEngine webEngine=webView.getEngine();
        Tab tabWebView=new Tab("Web View");
        tabWebView.setContent(webView);
        tabPane.getTabs().add(tabWebView);

        TableView<Map<String,Object>> tableViewData=new TableView<>();

        Tab tabData=new Tab("Input Data");
        BorderPane borderPane2=new BorderPane();
```

# Application Complète avec Interface graphique

```
HBox hBox2=new HBox(10);hBox2.setPadding(new Insets(10));
        Button buttonDataTrain=new Button("Train Data");
        Button buttonDataTest=new Button("Test Data");
        hBox2.getChildren().addAll(buttonDataTrain,buttonDataTest);
        borderPane2.setTop(hBox2);
        borderPane2.setCenter(tableViewData);
        tabData.setContent(borderPane2);
        tabPane.getTabs().add(tabData);

        Tab tabConsole=new Tab("Console");
        TextArea textAreaConsole=new TextArea();
        tabConsole.setContent(textAreaConsole);
        tabPane.getTabs().add(tabConsole);
```

# Application Complète avec Interface graphique

```
Tab tabPredictions=new Tab("Predictions");
        HBox hBoxPrediction=new HBox(10);hBoxPrediction.setPadding(new Insets(10));
        GridPane gridPane=new GridPane();gridPane.setPadding(new Insets(10));
        gridPane.setHgap(10);gridPane.setVgap(10);
        Label labelSepalLength=new Label("Sepal Length:");
        TextField textFieldSepalLength=new TextField("5.0");
        Label labelSepalWidth=new Label("Sepal Length:");
        TextField textFieldSepalWidth=new TextField("3.5");
        Label labelPetalLength=new Label("Petal Length:");
        TextField textFieldPetalLength=new TextField("1.3");
        Label labelPetalWidth=new Label("Petal Length:");
        TextField textFieldPetalWidth=new TextField("0.3");
        Button buttonPredict=new Button("Predict");
        Label labelPrection=new Label("?");
```

# Application Complète avec Interface graphique

```
gridPane.add(labelSepalLength,0,0); gridPane.add(textFieldSepalLength,1,0);
        gridPane.add(labelSepalWidth,0,1); gridPane.add(textFieldSepalWidth,1,1);
        gridPane.add(labelPetalLength,0,2); gridPane.add(textFieldPetalLength,1,2);
        gridPane.add(labelPetalWidth,0,3); gridPane.add(textFieldPetalWidth,1,3);
        gridPane.add(buttonPredict,0,4);
        gridPane.add(labelPrection,0,5);
        Image imagePrediction=new Image(new FileInputStream(new
ClassPathResource("images/unknown.png").getFile()));
        ImageView imageViewPrediction=new ImageView(imagePrediction);
        hBoxPrediction.getChildren().add(gridPane);
        hBoxPrediction.getChildren().add(imageViewPrediction);
        tabPredictions.setContent(hBoxPrediction);
        tabPane.getTabs().add(tabPredictions);
        HBox hBoxCenter=new HBox(10);hBoxCenter.setPadding(new Insets(10));
        VBox vBox1=new VBox(10);vBox1.setPadding(new Insets(10));
```

# Application Complète avec Interface graphique

```
ProgressBar progressBar=new ProgressBar();
        progressBar.setPrefWidth(image.getWidth());
        progressBar.setProgress(0);
        vBox1.getChildren().add(progressBar);
        vBox1.getChildren().add(imageView);
        vBox1.setVisible(false);
        hBoxCenter.getChildren().add(vBox1);
        hBoxCenter.getChildren().add(tabPane);
        borderPane.setCenter(hBoxCenter);
        Scene scene=new Scene(borderPane,800,600);
        primaryStage.setScene(scene);
        primaryStage.show();
```

# Application Complète avec Interface graphique

```
buttonCreateModel.setOnAction(evr->{
            new Thread(()->{
                MultiLayerConfiguration configuration=new NeuralNetConfiguration.Builder()
                        .seed(123)
                        .updater(new Adam(learninRate))
                        .list()
                        .layer(0,new DenseLayer.Builder()
                                .nIn(inputSize)
                                .nOut(numHiddenNodes)
                                .activation(Activation.SIGMOID).build())
                        .layer(1,new OutputLayer.Builder()
                                .nIn(numHiddenNodes)
                                .nOut(outputSize)
                                .lossFunction(LossFunctions.LossFunction.MEAN_SQUARED_LOGARITHMIC_ERROR)
                                .activation(Activation.SOFTMAX).build())
                        .build();
                model=new MultiLayerNetwork(configuration);
                model.init();
                imageView.setVisible(true);
            }).start();
        });
```

# Application Complète avec Interface graphique

```
buttonLoadData.setOnAction(evt->{
            NumberFormat numberFormat=new DecimalFormat("#0.00");
            for (int i = 0; i < featuresLabels.length; i++) {
                TableColumn<Map<String,Object>,String> column=new TableColumn<>(featuresLabels[i]);
                column.setCellValueFactory(p->{
                    return new SimpleObjectProperty(p.getValue().get(p.getTableColumn().getText()));
                });
                column.setCellFactory(p->{
                    TableCell tableCell=new TableCell(){
                        @Override
                        protected void updateItem(Object item, boolean empty) {
                            super.updateItem(item, empty);      if(item instanceof Number)       this.setText(numberFormat.format(item));
                            else if(item==null) this.setText("NULL");  else  this.setText(item.toString());
                        }
                    };
                    return tableCell;
                });
                tableViewData.getColumns().add(column);
            }
            TableColumn<Map<String,Object>,String> tableColumnLabel=new TableColumn<>("Label");
            tableColumnLabel.setCellValueFactory(p->{
                return new SimpleObjectProperty(p.getValue().get(p.getTableColumn().getText()));
            });
            tableViewData.getColumns().add(tableColumnLabel);
            viewDataSet("iris-train.csv",tableViewData);
        });
```

# Application Complète avec Interface graphique

```
buttonTrain.setOnAction(evt->{
          new Thread(()->{
              try {
                  progressBar.setProgress(0); progressValue=0; UIServer uiServer=UIServer.getInstance();
                  inMemoryStatsStorage=new InMemoryStatsStorage(); uiServer.attach(inMemoryStatsStorage);
                  //model.setListeners(new ScoreIterationListener(10));
                  model.setListeners(new StatsListener(inMemoryStatsStorage));
                  Platform.runLater(()->{
                      vBox1.setVisible(true);  webEngine.load("http://localhost:9000");
                  });
                  File fileTrain=new ClassPathResource("iris-train.csv").getFile();
                  RecordReader recordReaderTrain=new CSVRecordReader();
                  recordReaderTrain.initialize(new FileSplit(fileTrain));
                  DataSetIterator dataSetIteratorTrain=
                          new RecordReaderDataSetIterator(recordReaderTrain,batchSize,classIndex,outputSize);
              for (int i = 0; i <nEpochs ; i++) {
                  model.fit(dataSetIteratorTrain);
                  Platform.runLater(()->{
                      progressValue+=(1.0/nEpochs);
                      progressBar.setProgress(progressValue);
                  });
              }
              } catch (Exception e) {
                  e.printStackTrace();
              }
          }) .start();
      });
```

# Application Complète avec Interface graphique

```
buttonEval.setOnAction(evt->{
            new Thread(()->{
                try {
                    System.out.println("Model Evaluation");
                    File fileTest=new ClassPathResource("irisTest.csv").getFile();
                    RecordReader recordReaderTest=new CSVRecordReader();
                    recordReaderTest.initialize(new FileSplit(fileTest));
                    DataSetIterator dataSetIteratorTest=
                            new RecordReaderDataSetIterator(recordReaderTest,batchSize,classIndex,outputSize);
                    Evaluation evaluation=new Evaluation(outputSize);

                    while (dataSetIteratorTest.hasNext()){
                        DataSet dataSet = dataSetIteratorTest.next();
                        INDArray features=dataSet.getFeatures();
                        INDArray labels=dataSet.getLabels();
                        INDArray predicted=model.output(features);
                        evaluation.eval(labels,predicted);
                    }
                    textAreaConsole.appendText(evaluation.stats());
                } catch (IOException e) {
                    e.printStackTrace();
                } catch (InterruptedException e) {
                    e.printStackTrace();
                }
            }).start();
        });
```

# Application Complète avec Interface graphique

```
buttonDataTrain.setOnAction(evt->{
            viewDataSet("iris-train.csv",tableViewData);
        });
        buttonDataTest.setOnAction(evt->{
            viewDataSet("irisTest.csv",tableViewData);
        });
        buttonPredict.setOnAction(evt->{
            try {
                double sl=Double.parseDouble(textFieldSepalLength.getText());
                double sw=Double.parseDouble(textFieldSepalWidth.getText());
                double pl=Double.parseDouble(textFieldPetalLength.getText());
                double pw=Double.parseDouble(textFieldPetalWidth.getText());
                System.out.println("Prediction :");
                INDArray input= Nd4j.create(new double[][]{{sl,sw,pl,pw}});
                INDArray ouput=model.output(input);
                textAreaConsole.appendText(ouput.toString());
                String labelOutput=labels[Nd4j.argMax(ouput).getInt(0)];
                labelPrection.setText(labelOutput);
                imageViewPrediction.setImage(new Image(new FileInputStream(new ClassPathResource("images/"+labelOutput+".png").getFile())));
            } catch (IOException e) {
                e.printStackTrace();
            }
        });
    }
```

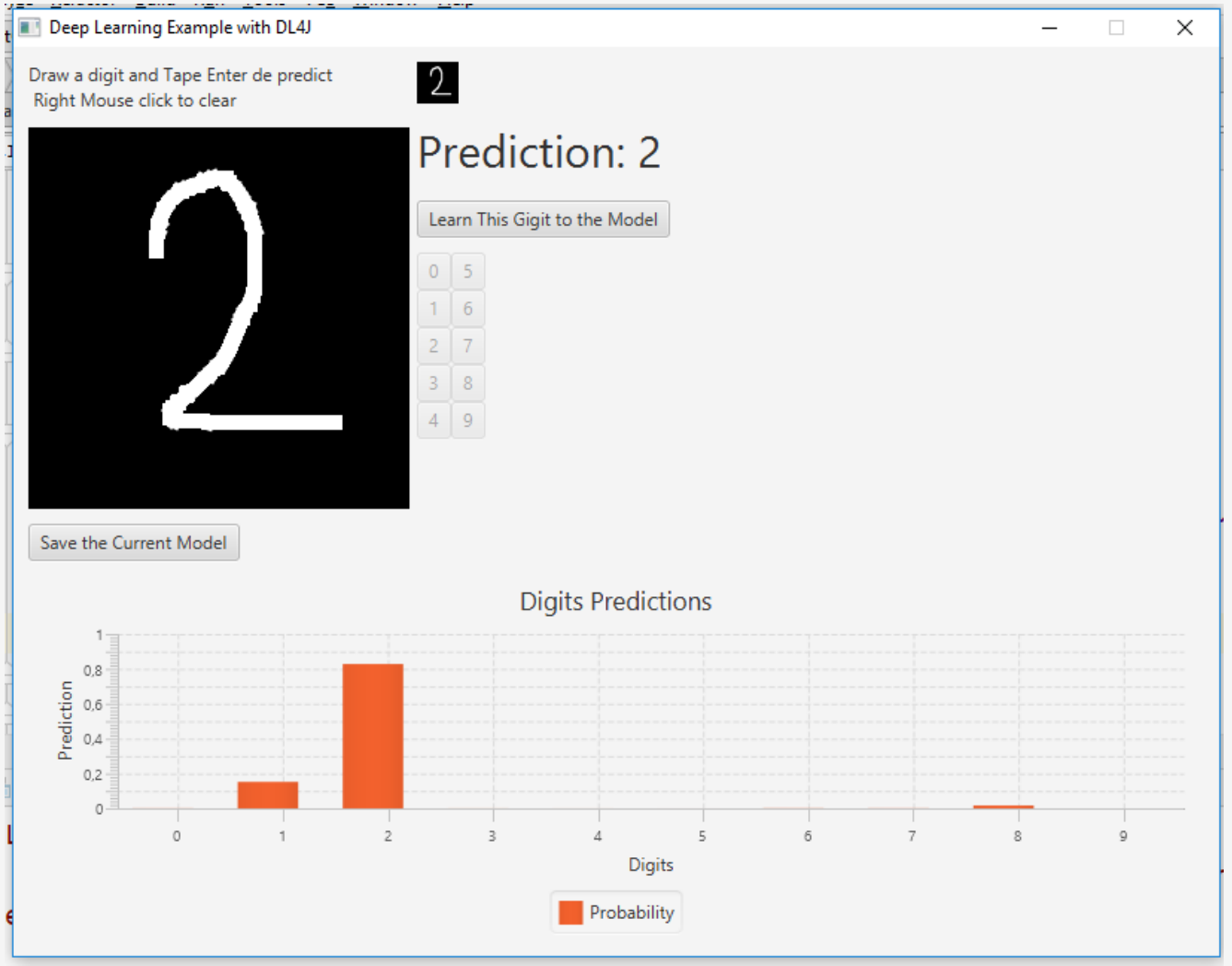
# Application Complète avec Interface graphique

```
private void viewDataSet(String fileName, TableView tableView) {
        try {
            File file=new ClassPathResource(fileName).getFile();
            RecordReader recordReader=new CSVRecordReader();   recordReader.initialize(new FileSplit(file));
            DataSetIterator dataSetIterator=
                    new RecordReaderDataSetIterator(recordReader,batchSize,classIndex,outputSize);
           tableView.getItems().clear();

        while (dataSetIterator.hasNext()){
            DataSet dataSet=dataSetIterator.next();
            INDArray features=dataSet.getFeatures();
            INDArray targets=dataSet.getLabels();
            for (int k = 0; k <batchSize ; k++) {
                INDArray batchFeature=features.getRow(k);  INDArray batchLabel=targets.getRow(k);
                Map<String,Object> data=new HashMap<>();  double[] row=batchFeature.toDoubleVector();
                INDArray rowLabels=batchLabel.getRow(0);
                for (int j = 0; j <row.length ; j++) {
                    data.put(featuresLabels[j],row[j]);
                }
                data.put("Label",labels[rowLabels.argMax().getInt(0)]);  tableView.getItems().add(data);
            }
        }
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

# Application 2 : Classification des images en utilisant CNN

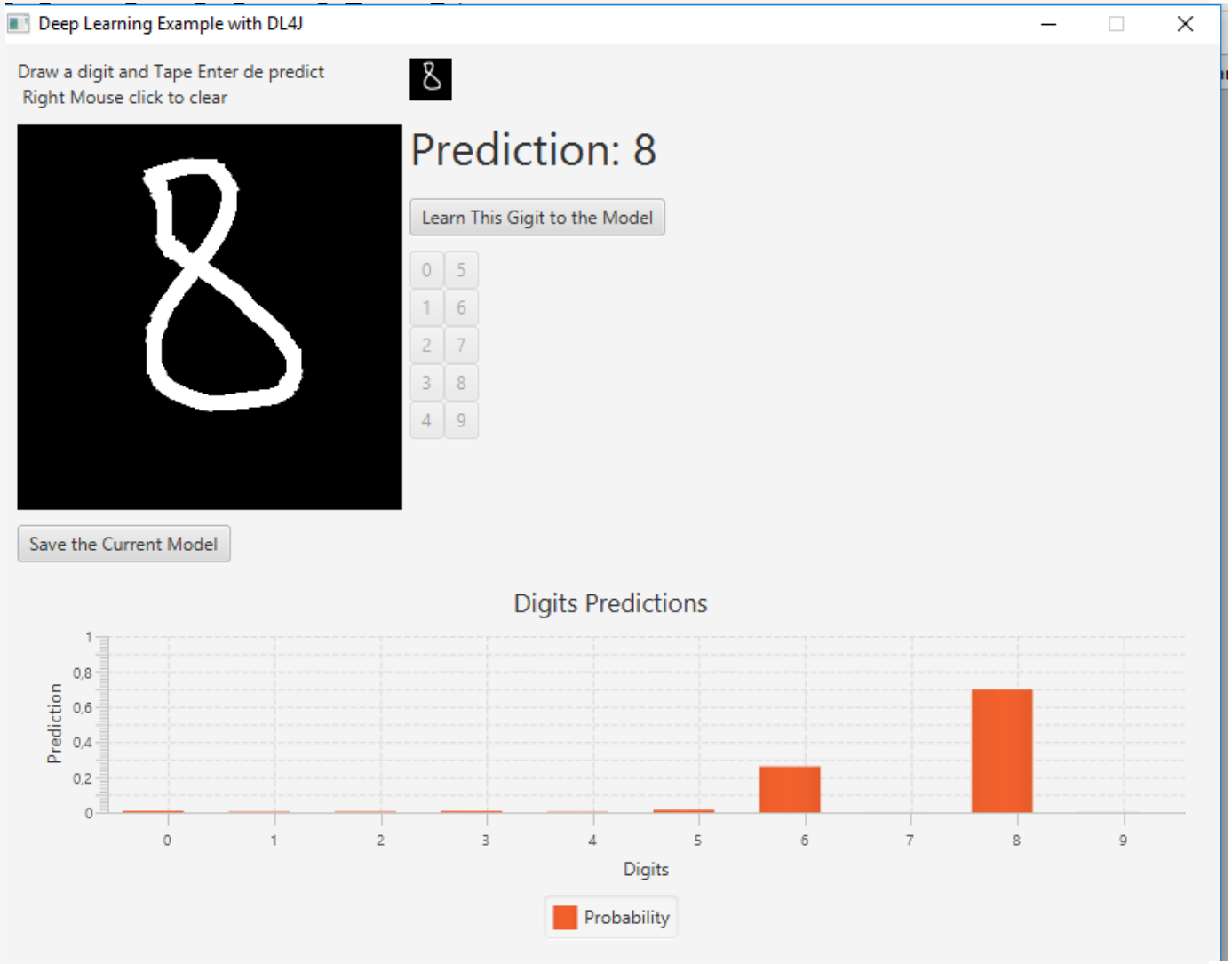
https://github.com/mohamedYoussfi/deeplearning4j-cnn-mnist-app

- L'objectif de cette application étant de créer un modèle de reconnaissance des formes en utilisant Deep Learning avec les Réseaux de neurones à convolution CNN.
- Pour l'entrainement du réseaux, nous utilisons le data set Mninst. Dans cet exemple on s'intéresse à reconnaitre les digits de 0 à 9 dessiné manuellement dans une image.



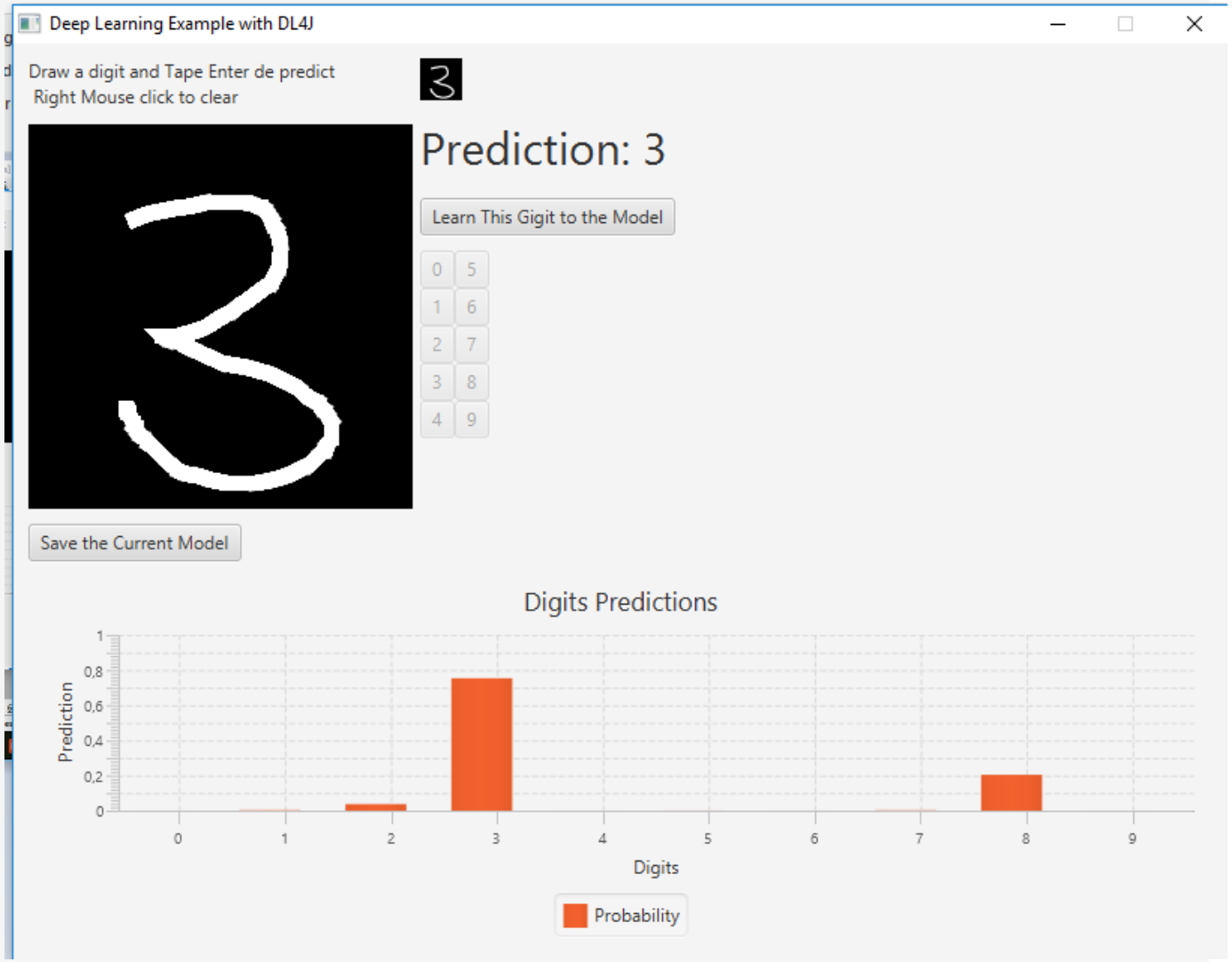
# Application 2 : Classification des images en utilisant CNN

- L'objectif de cette application étant de créer un modèle de reconnaissance des formes en utilisant Deep Learning avec les Réseaux de neurones à convolution CNN.
- Pour l'entrainement du réseaux, nous utilisons le data set Mninst. Dans cet exemple on s'intéresse à reconnaitre les digits de 0 à 9 dessiné manuellement dans une image.



# Application 2 : Classification des images en utilisant CNN

- L'objectif de cette application étant de créer un modèle de reconnaissance des formes en utilisant Deep Learning avec les Réseaux de neurones à convolution CNN.
- Pour l'entrainement du réseaux, nous utilisons le data set Mninst. Dans cet exemple on s'intéresse à reconnaitre les digits de 0 à 9 dessiné manuellement dans une image.



# Application 2 : Classification des images en utilisant CNN

## Packages

```
import org.datavec.api.io.labels.ParentPathLabelGenerator; import org.datavec.api.split.FileSplit;
import org.datavec.image.loader.NativeImageLoader; import org.datavec.image.recordreader.ImageRecordReader;
import org.deeplearning4j.api.storage.StatsStorage; import org.deeplearning4j.datasets.datavec.RecordReaderDataSetIterator;
import org.deeplearning4j.nn.api.OptimizationAlgorithm;
import org.deeplearning4j.nn.conf.BackpropType; import org.deeplearning4j.nn.conf.MultiLayerConfiguration;
import org.deeplearning4j.nn.conf.NeuralNetConfiguration; import org.deeplearning4j.nn.conf.inputs.InputType;
import org.deeplearning4j.nn.conf.layers.ConvolutionLayer; import org.deeplearning4j.nn.conf.layers.DenseLayer;
import org.deeplearning4j.nn.conf.layers.OutputLayer; import org.deeplearning4j.nn.conf.layers.SubsamplingLayer;
import org.deeplearning4j.nn.multilayer.MultiLayerNetwork; import org.deeplearning4j.nn.weights.WeightInit;
import org.deeplearning4j.ui.api.UIServer; import org.deeplearning4j.ui.stats.StatsListener;
import org.deeplearning4j.ui.storage.InMemoryStatsStorage; import org.deeplearning4j.util.ModelSerializer;
import org.nd4j.evaluation.classification.Evaluation; import org.nd4j.linalg.activations.Activation;
import org.nd4j.linalg.dataset.api.iterator.DataSetIterator; import org.nd4j.linalg.dataset.api.preprocessor.DataNormalization;
import org.nd4j.linalg.dataset.api.preprocessor.ImagePreProcessingScaler; import org.nd4j.linalg.learning.config.Nesterovs;
import org.nd4j.linalg.lossfunctions.LossFunctions; import org.nd4j.linalg.schedule.MapSchedule;
import org.nd4j.linalg.schedule.ScheduleType; import org.slf4j.Logger; import org.slf4j.LoggerFactory; import java.io.File;
import java.io.IOException; import java.util.HashMap; import java.util.Map; import java.util.Random;
```

# Application 2 : Classification des images en utilisant CNN

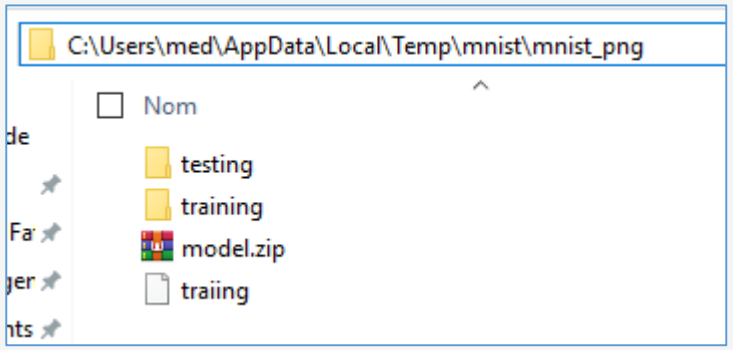
**Paramètres du modèle**

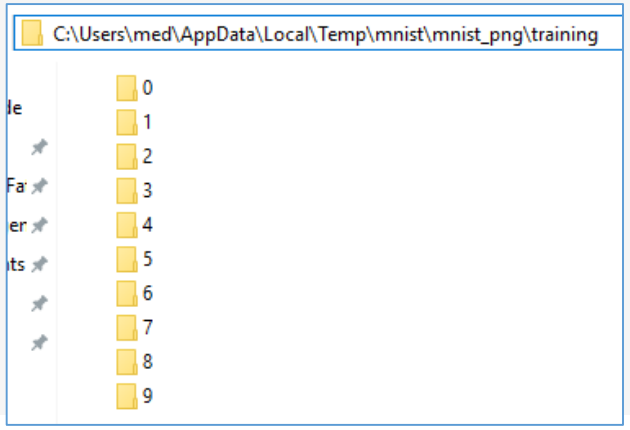
```
public class ConvNeuralNatMnist {
    private static Logger logger= LoggerFactory.getLogger(ConvNeuralNatMnist.class);
    public static void main(String[] args) throws IOException {
        String basePath=System.getProperty("java.io.tmpdir")+"/mnist/mnist_png";
        int height=28;int width=28;   int channels=1;// signe channel for graysacle image
        int outputNum=10;  int batchSize=54;  int epochCount=1;
        int seed =1234;   Random randomGenNum=new Random(seed);
Map<Integer,Double> learningRateByIterations=new HashMap<>();
learningRateByIterations.put(0,0.06);    learningRateByIterations.put(200,0.05);
learningRateByIterations.put(600,0.028); learningRateByIterations.put(800,0.006);
learningRateByIterations.put(1000,0.001);   double quadraticError=0.0005;
double momentum=0.9;
```

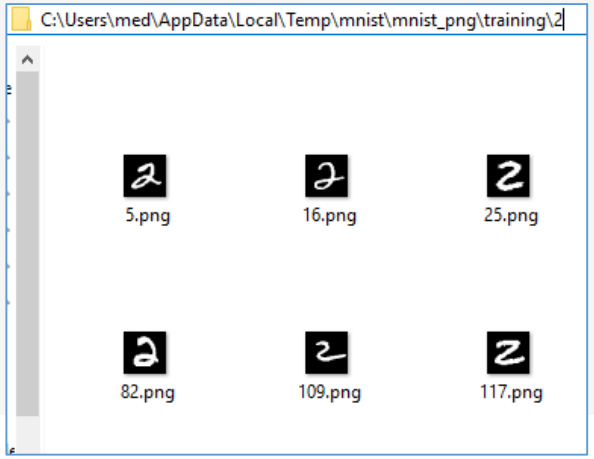
# Application 2 : Classification des images en utilisant CNN

## Chargement et Vectorisation des données

```
Logger.info("Data Load and vectorisation");
File trainDataFile=new File(basePath+"/training");
FileSplit trainFileSplit=new FileSplit(trainDataFile,NativeImageLoader.ALLOWED_FORMATS,randomGenNum);
ParentPathLabelGenerator labelMarker=new ParentPathLabelGenerator();
ImageRecordReader trainImageRecordReader= new ImageRecordReader(height,width,channels,labelMarker);
trainImageRecordReader.initialize(trainFileSplit);
```



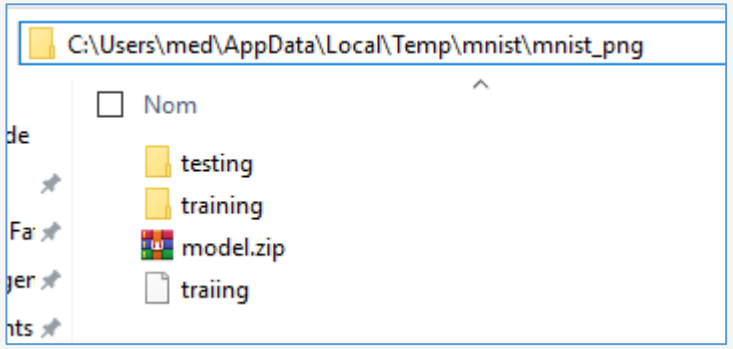



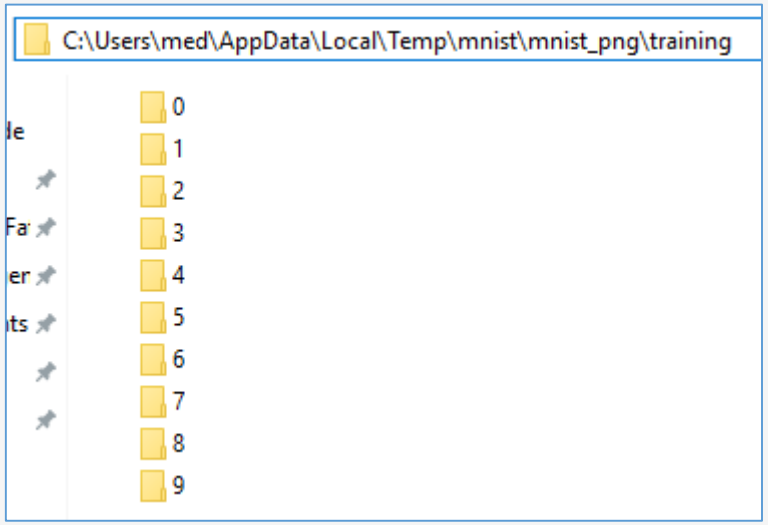



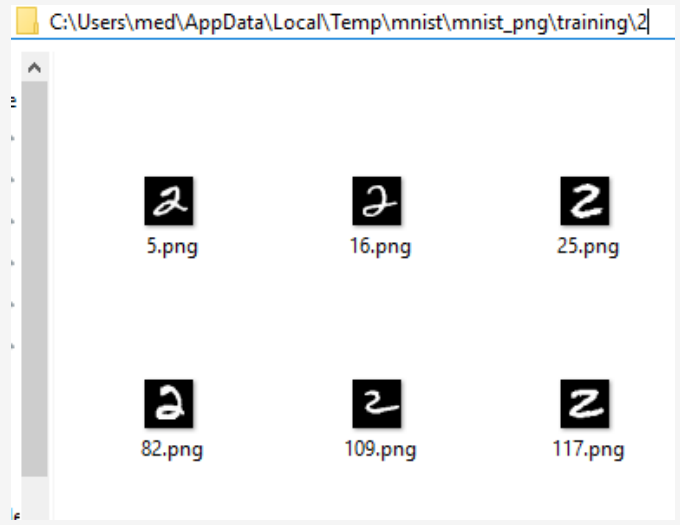
# Application 2 : Classification des images en utilisant CNN

## Chargement et Vectorisation des données

```
int labelIndex=1;
DataSetIterator trainDataSetIterator=new
RecordReaderDataSetIterator(trainImageRecordReader,batchSize,labelIndex,outputNum);
DataNormalization scaler=new ImagePreProcessingScaler(0,1);
scaler.fit(trainDataSetIterator);
trainDataSetIterator.setPreProcessor(scaler);
```



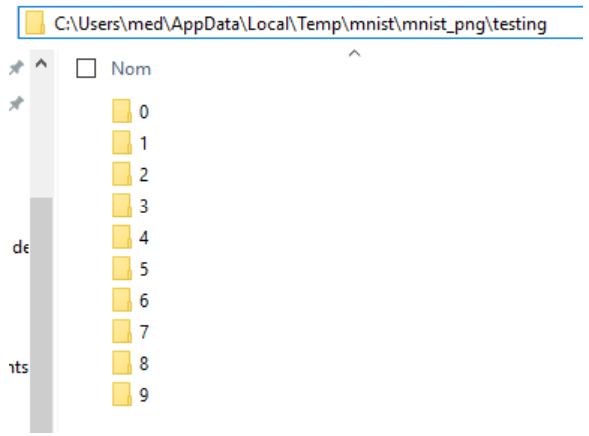



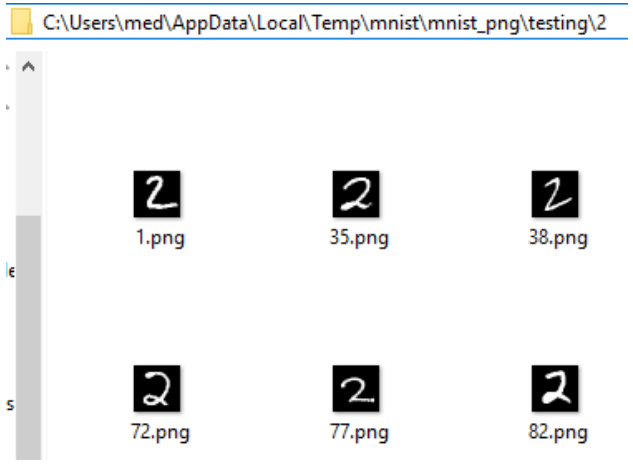



# Application 2 : Classification des images en utilisant CNN

## Chargement et Vectorisation des données

```
 File testDataFile=new File(basePath+"/testing");
FileSplit testFileSplit=new FileSplit(testDataFile, NativeImageLoader.ALLOWED_FORMATS,randomGenNum);
ImageRecordReader testImageRecordReader=new ImageRecordReader(height,width,channels,labelMarker);
testImageRecordReader.initialize(testFileSplit);
DataSetIterator testDataSetIterator=new
RecordReaderDataSetIterator(testImageRecordReader,batchSize,labelIndex,outputNum);
        trainDataSetIterator.setPreProcessor(scaler);
```



# Application 2 : Classification des images en utilisant CNN

## Configuration du modèle

```
Logger.info("Neural Network Model Configuation");
MultiLayerConfiguration configuration=new NeuralNetConfiguration.Builder()
  .seed(seed)
  .l2(quadraticError)
  .optimizationAlgo(OptimizationAlgorithm.STOCHASTIC_GRADIENT_DESCENT)
  .updater(new Nesterovs(new MapSchedule(ScheduleType.ITERATION,learningRateByIterations),momentum))
        .weightInit(WeightInit.XAVIER)
```

# Application 2 : Classification des images en utilisant CNN

## Configuration du modèle

```
        .list()
            .layer(0,new ConvolutionLayer.Builder()
                    .kernelSize(5,5)
                    .nIn(channels)
                    .stride(1,1)
                    .nOut(20)
                    .activation(Activation.RELU).build())
```

# Application 2 : Classification des images en utilisant CNN

## Configuration du modèle

```
.layer(1, new SubsamplingLayer.Builder()
          .poolingType(SubsamplingLayer.PoolingType.MAX)
           .kernelSize(2,2)
           .stride(2,2)
           .build())
```

# Application 2 : Classification des images en utilisant CNN

## Configuration du modèle

```
.layer(2, new ConvolutionLayer.Builder(5,5)
        .stride(1,1)
        .nOut(50)
        .activation(Activation.RELU)
        .build())
```

# Application 2 : Classification des images en utilisant CNN

## Configuration du modèle

```
.layer(3, new SubsamplingLayer.Builder(SubsamplingLayer.PoolingType.MAX)
        .kernelSize(2,2)
        .stride(2,2)
        .build())
```

# Application 2 : Classification des images en utilisant CNN

## Configuration du modèle

```
.layer(4, new DenseLayer.Builder()
        .activation(Activation.RELU)
        .nOut(500)
        .build())
```

# Application 2 : Classification des images en utilisant CNN

## Configuration du modèle

```
.layer(5,new OutputLayer.Builder(LossFunctions.LossFunction.NEGATIVELOGLIKELIHOOD)
        .activation(Activation.SOFTMAX)
        .nOut(outputNum)
        .build())
```

# Application 2 : Classification des images en utilisant CNN

## Configuration du modèle

```
.setInputType(InputType.convolutionalFlat(height,width,channels))
        .backpropType(BackpropType.Standard)
        .build();
```

# Application 2 : Classification des images en utilisant CNN

## Chargement et Vectorisation des données

```
System.out.println(configuration.toJson());
MultiLayerNetwork model=new MultiLayerNetwork(configuration);
model.init();
```

# Application 2 : Classification des images en utilisant CNN

**Démarre un listener pour suivre l'évolution du processus d'apprentissage**

```
UIServer uiServer=UIServer.getInstance();
        StatsStorage statsStorage=new InMemoryStatsStorage();
        uiServer.attach(statsStorage);
        model.setListeners(new StatsListener(statsStorage));
```

# Application 2 : Classification des images en utilisant CNN

**Entrainement et évaluation du modèle pour chaque époque**

```
 Logger.info("Total params:"+model.numParams());
        for (int i = 0; i < epochCount; i++) {
            model.fit(trainDataSetIterator);
            Logger.info("End of epoch "+i);
            Evaluation evaluation=model.evaluate(testDataSetIterator);
            Logger.info(evaluation.stats());
            trainDataSetIterator.reset();
            testDataSetIterator.reset();
        }
        Logger.info("Saving model ....");
        ModelSerializer.writeModel(model,new File(basePath+"/model.zip"),true);
    }
}
```

# Application 2 : Classification des images en utilisant CNN

**Entrainement et évaluation du modèle pour chaque époque**

```
18665 [main] INFO ConvNeuralNatMnist - Total params:431080
124484 [main] INFO ConvNeuralNatMnist - End of epoch 0
128756 [main] INFO ConvNeuralNatMnist -


========================Evaluation Metrics========================
 # of classes:    10
 Accuracy:        0,9851
 Precision:       0,9853
 Recall:          0,9849
 F1 Score:        0,9850
Precision, recall & F1: macro-averaged (equally weighted avg. of 10 classes)
```

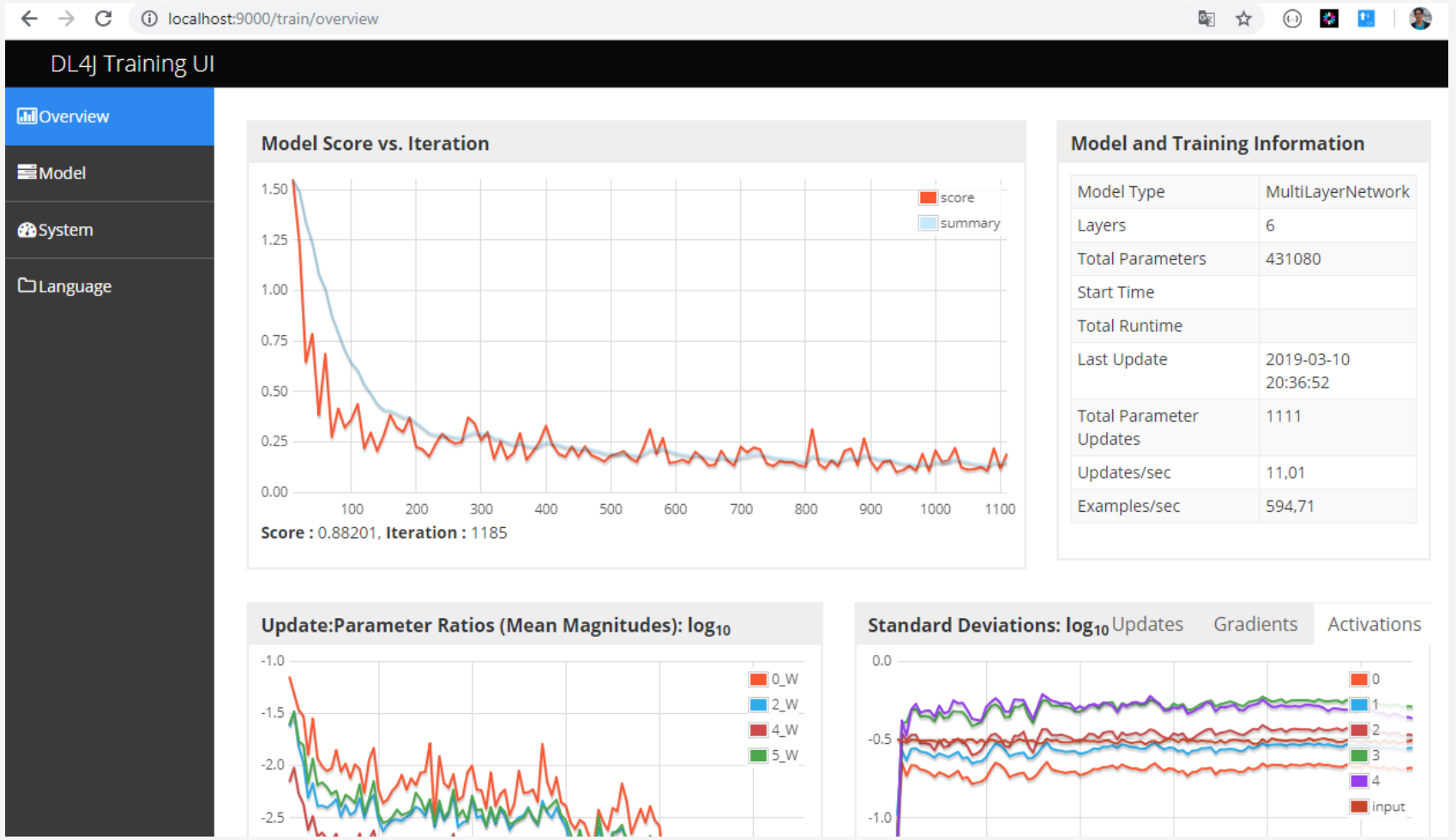
# Application 2 : Classification des images en utilisant CNN

## Entrainement et évaluation du modèle pour chaque époque

```
========================Confusion Matrix=========================

    0    1    2    3    4    5    6    7    8    9

-----------------------------------------------------

  973    2    0    0    1    0    0    2    0    2 | 0 = 0

    0 1131    0    1    0    0    2    0    1    0 | 1 = 1

    3    2 1012    0    2    0    0    8    5    0 | 2 = 2

    1    0    2  992    0    4    0    4    7    0 | 3 = 3

    0    0    0    0  981    0    0    0    1    0 | 4 = 4

    3    0    0    5    0  870    4    3    4    3 | 5 = 5

    9    4    0    0    6    0  939    0    0    0 | 6 = 6

    0    4    6    0    0    0    0 1016    1    1 | 7 = 7

    6    1    1    0    2    0    0    2  961    1 | 8 = 8

    3    4    0    1   14    2    0    6    3  976 | 9 = 9


Confusion matrix format: Actual (rowClass) predicted as (columnClass) N times

=================================================================
```

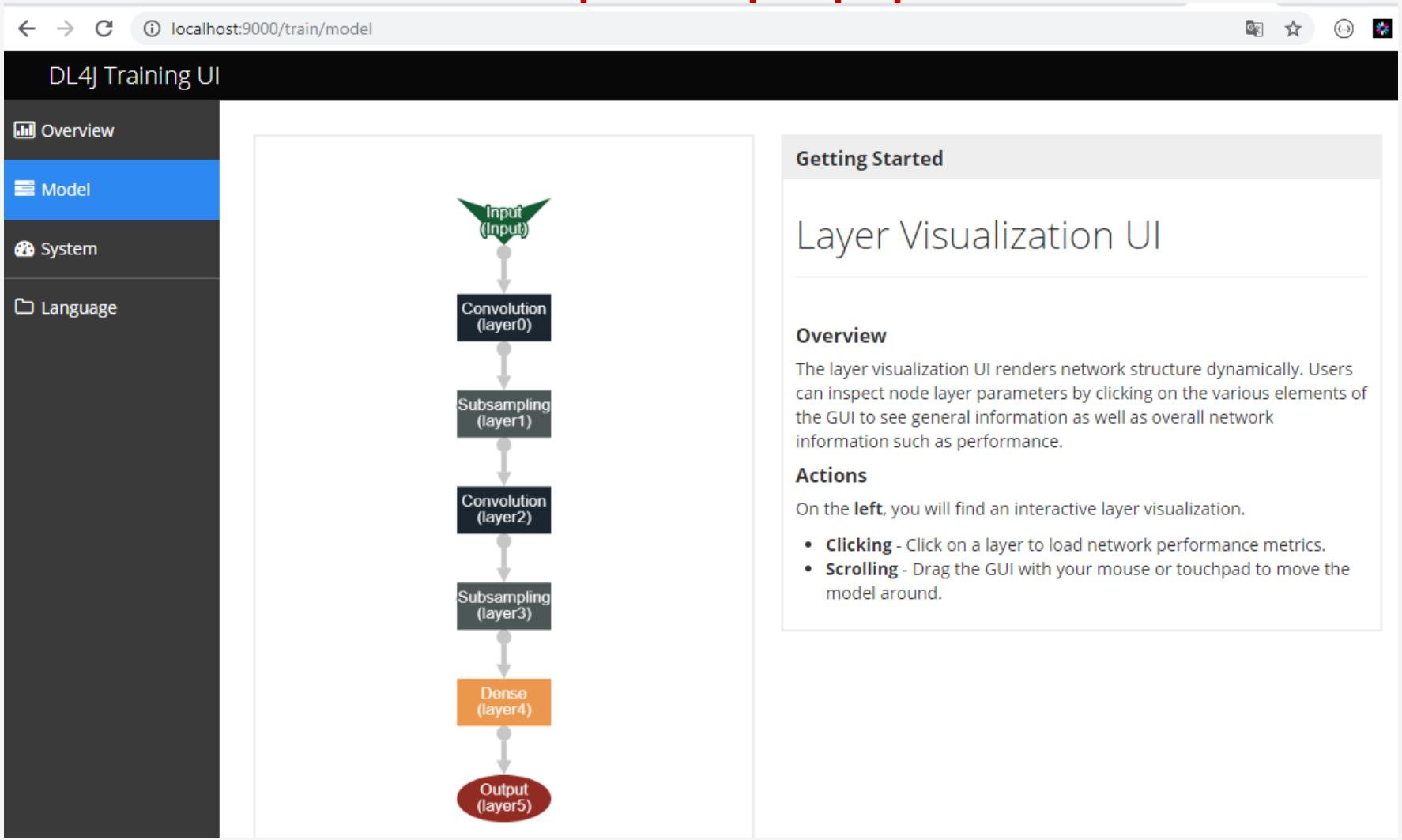
# Application 2 : Classification des images en utilisant CNN

## Entrainement et évaluation du modèle pour chaque époque



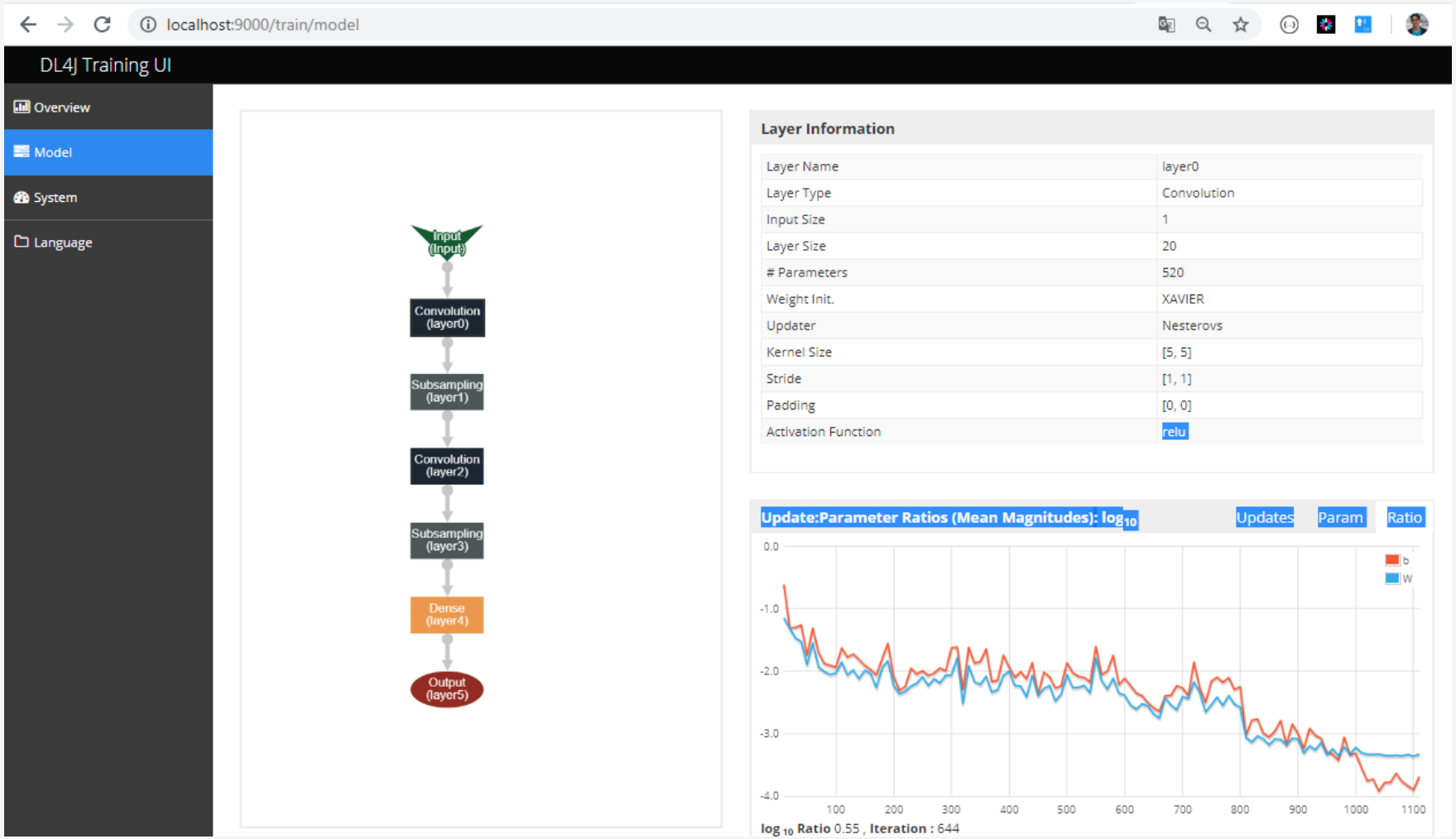
# Application 2 : Classification des images en utilisant CNN

## Entrainement et évaluation du modèle pour chaque époque



# Application 2 : Classification des images en utilisant CNN

## Entrainement et évaluation du modèle pour chaque époque



# UI Application avec JavaFX

## Pacakges

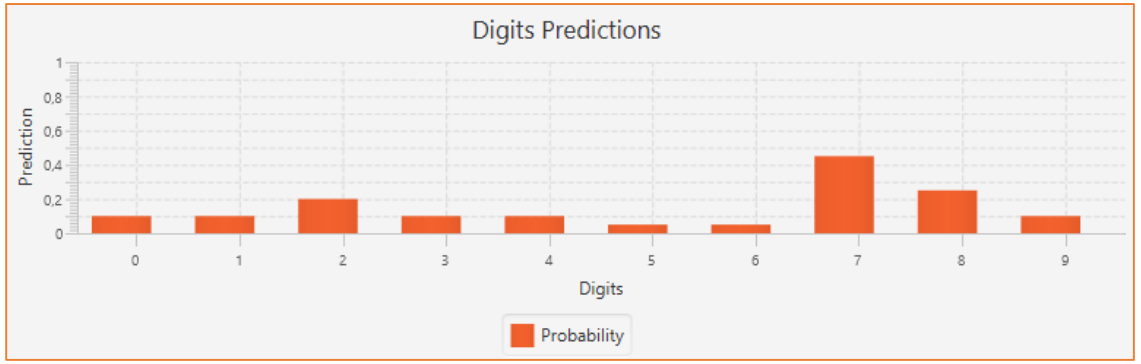
```
import javafx.application.Application; import javafx.embed.swing.SwingFXUtils;
import javafx.event.ActionEvent; import javafx.geometry.Insets; import javafx.geometry.Pos;
import javafx.scene.Scene; import javafx.scene.canvas.Canvas; import javafx.scene.canvas.GraphicsContext;
import javafx.scene.chart.BarChart; import javafx.scene.chart.CategoryAxis; import javafx.scene.chart.NumberAxis;
import javafx.scene.chart.XYChart; import javafx.scene.control.Alert; import javafx.scene.control.Button;
import javafx.scene.control.ButtonType; import javafx.scene.control.Label; import javafx.scene.image.ImageView;
import javafx.scene.image.WritableImage; import javafx.scene.input.KeyCode; import javafx.scene.input.MouseButton;
import javafx.scene.layout.GridPane; import javafx.scene.layout.HBox; import javafx.scene.layout.VBox;
import javafx.scene.paint.Color; import javafx.scene.shape.StrokeLineCap; import javafx.scene.text.Font;
import javafx.stage.Stage; import org.datavec.image.loader.NativeImageLoader;
import org.deeplearning4j.nn.multilayer.MultiLayerNetwork; import org.deeplearning4j.util.ModelSerializer;
import org.nd4j.linalg.api.ndarray.INDArray;
import org.nd4j.linalg.dataset.api.preprocessor.ImagePreProcessingScaler; import org.nd4j.linalg.factory.Nd4j;
import java.awt.*; import java.awt.image.BufferedImage; import java.io.File; import java.io.IOException;
import java.util.*;
```

# UI Application avec JavaFX

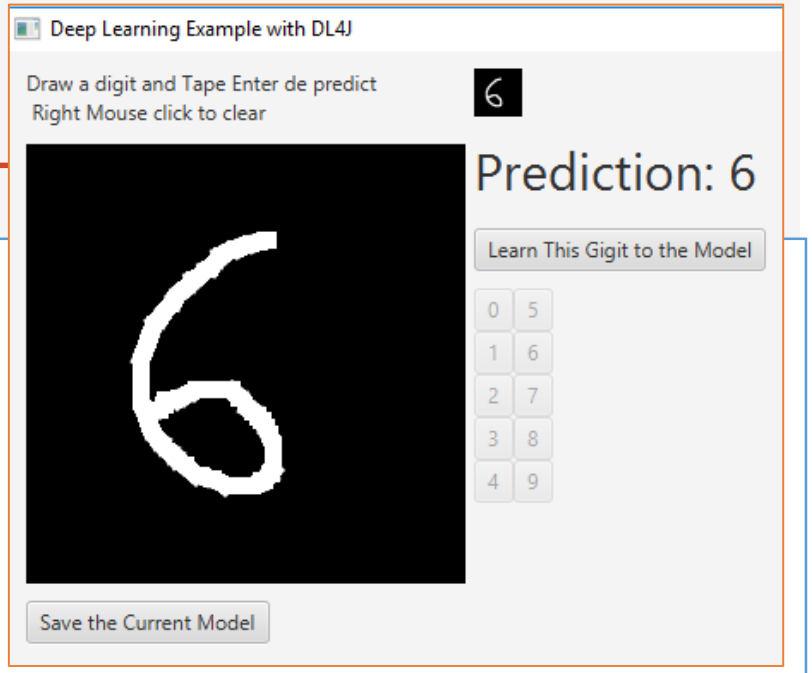
```
public class MnistClassifierUI extends Application {
  private static final String basePath = System.getProperty("java.io.tmpdir") + "/mnist/mnist_png";
  private final int canvasWidth = 256;  private final int canvasHeight = 256;
  private MultiLayerNetwork net;  private INDArray currentImage;
  private Canvas canvas;  private List<XYChart.Data> data;
  private BarChart<String,Double> barChart;

 public MnistClassifierUI() throws IOException {
    File model = new File(basePath + "/model.zip");
    if (!model.exists())      throw new IOException("Can't find the model");
    net = ModelSerializer.restoreMultiLayerNetwork(model);
  }
  public static void main(String[] args) throws Exception {
    launch();
  }
```

# UI Application avec JavaFX : BarChart

```
public void drawChart(){
  CategoryAxis xAxis=new CategoryAxis();   NumberAxis yAxis=new NumberAxis(0,1,0.1);
  barChart=new BarChart(xAxis,yAxis);   barChart.setCategoryGap(10); barChart.setBarGap(10);
  barChart.setTitle("Digits Predictions");   xAxis.setLabel("Digits");  yAxis.setLabel("Prediction");
  XYChart.Series series1=new XYChart.Series();   series1.setName("Probability");
  double[]values=new double[]{0.1,0.1,0.2,0.1,0.1,0.05,0.05,0.45,0.25,0.1} ;
  final String[] categoriesNames = new String[] {"0", "1", "2","3", "4", "5","6", "7", "8","9"};
  xAxis.getCategories().setAll(categoriesNames);
  data=new ArrayList<>();
  for(int i=0;i<categoriesNames.length;i++){
    data.add(new XYChart.Data(categoriesNames[i],values[i]));
  }
  series1.getData().addAll(data);
  barChart.getData().addAll(series1);
}
```



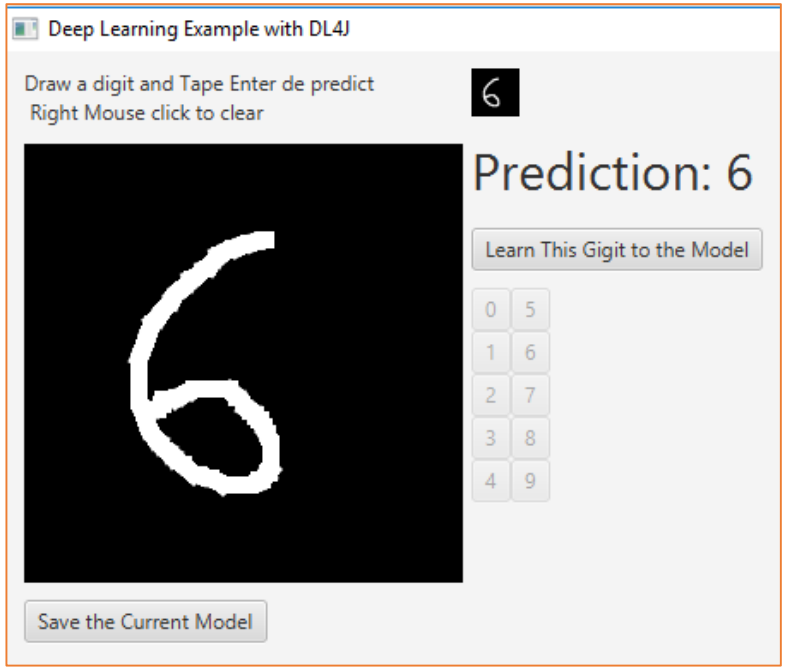
# UI Application avec JavaFX : Construire l'interface




```
  @Override
  public void start(Stage stage) throws Exception {
      canvas = new Canvas(canvasWidth, canvasHeight);
      GraphicsContext ctx = canvas.getGraphicsContext2D();
      ImageView imgView = new ImageView(); imgView.setFitHeight(28);
      imgView.setFitWidth(28);
      ctx.setLineWidth(10);    ctx.setLineCap(StrokeLineCap.SQUARE);
      Label labelMessage=new Label("Draw a digit and Tape Enter de predict"+System.lineSeparator()+" Right Mouse
click to clear");
      Label lblResult = new Label(); lblResult.setFont(new Font(30));
      Button buttonLearn=new Button("Learn This Gigit to the Model");
      Button buttonSaveModel=new Button("Save the Current Model");
```

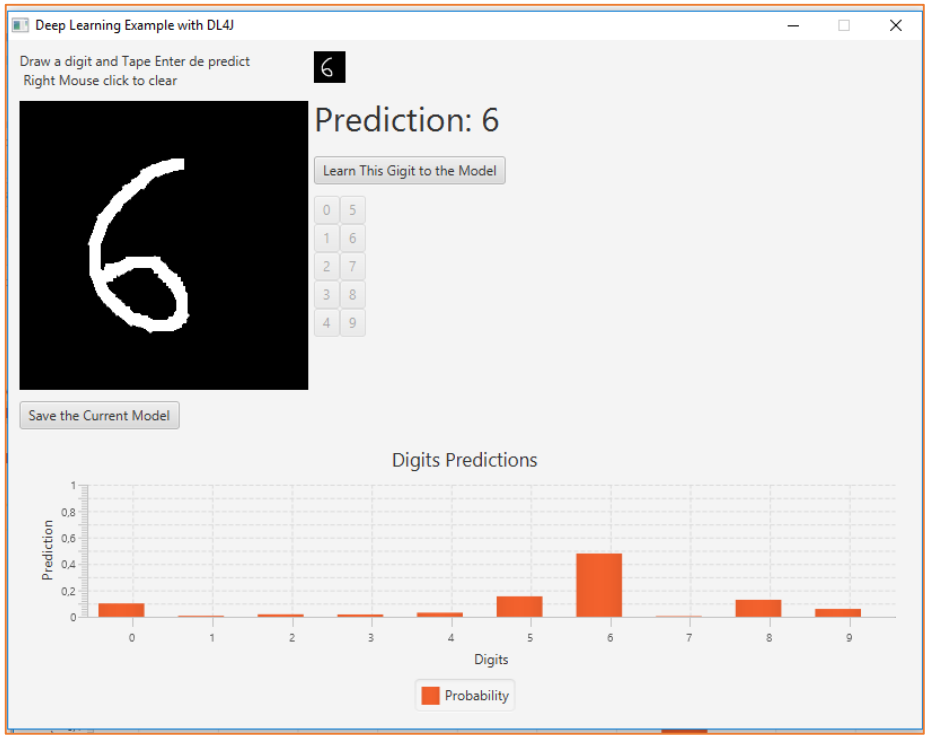
# UI Application avec JavaFX : Construire l'interface

```
GridPane gridPane=new GridPane();int index=-1;
gridPane.setDisable(true);
   for(int i=0;i<2;i++)
     for(int j=0;j<5;j++){
       Button button=new Button(String.valueOf(++index));
       gridPane.add(button,i,j);
       button.setOnAction((evt)->{
         learnThisDigitToModel(evt);
       });
   }
```



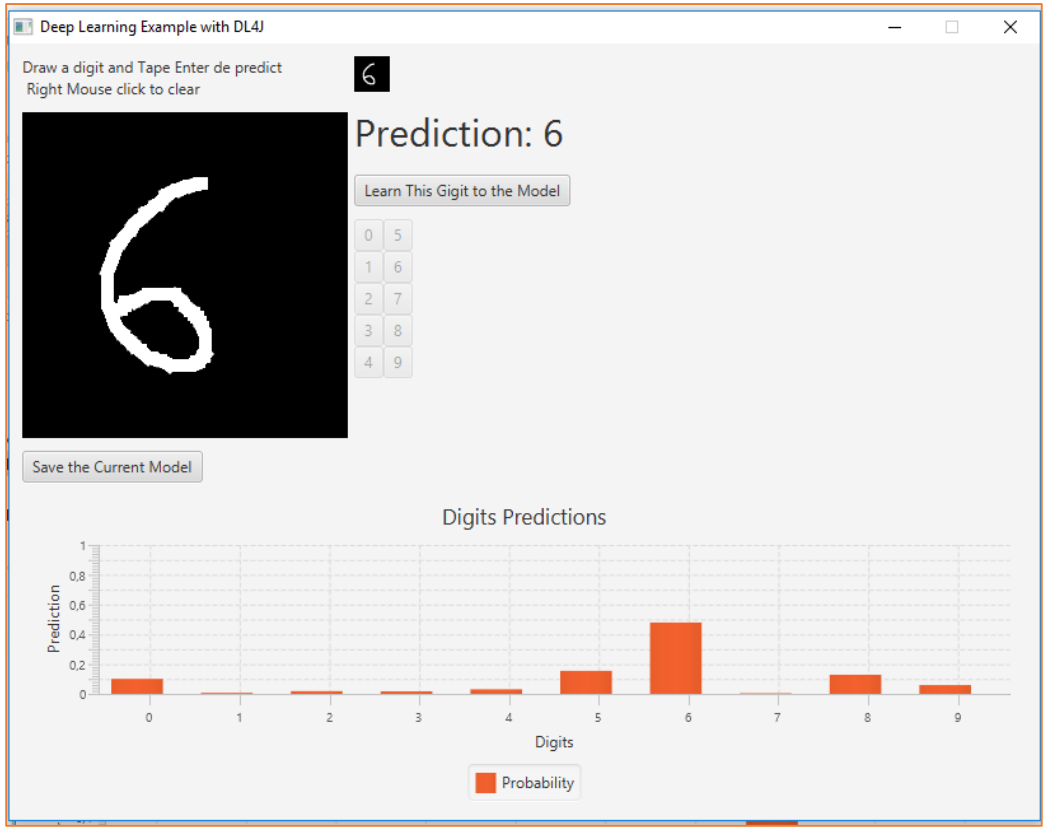
# UI Application avec JavaFX : Construire l'interface

```
VBox vbBottom = new VBox(10, imgView, lblResult,buttonLearn,gridPane);
vbBottom.setAlignment(Pos.TOP_LEFT);
VBox vBoxCanvas=new VBox(10,labelMessage,canvas,buttonSaveModel);
HBox hBox2=new HBox(5, vBoxCanvas, vbBottom);
VBox root = new VBox(10,hBox2);
root.setAlignment(Pos.CENTER);
root.setPadding(new Insets(10));
drawChart();
root.getChildren().add(barChart);
Scene scene = new Scene(root, 800, 600);
stage.setScene(scene);
stage.setTitle("Deep Learning Example with DL4J");
stage.setResizable(false);
stage.show();
```



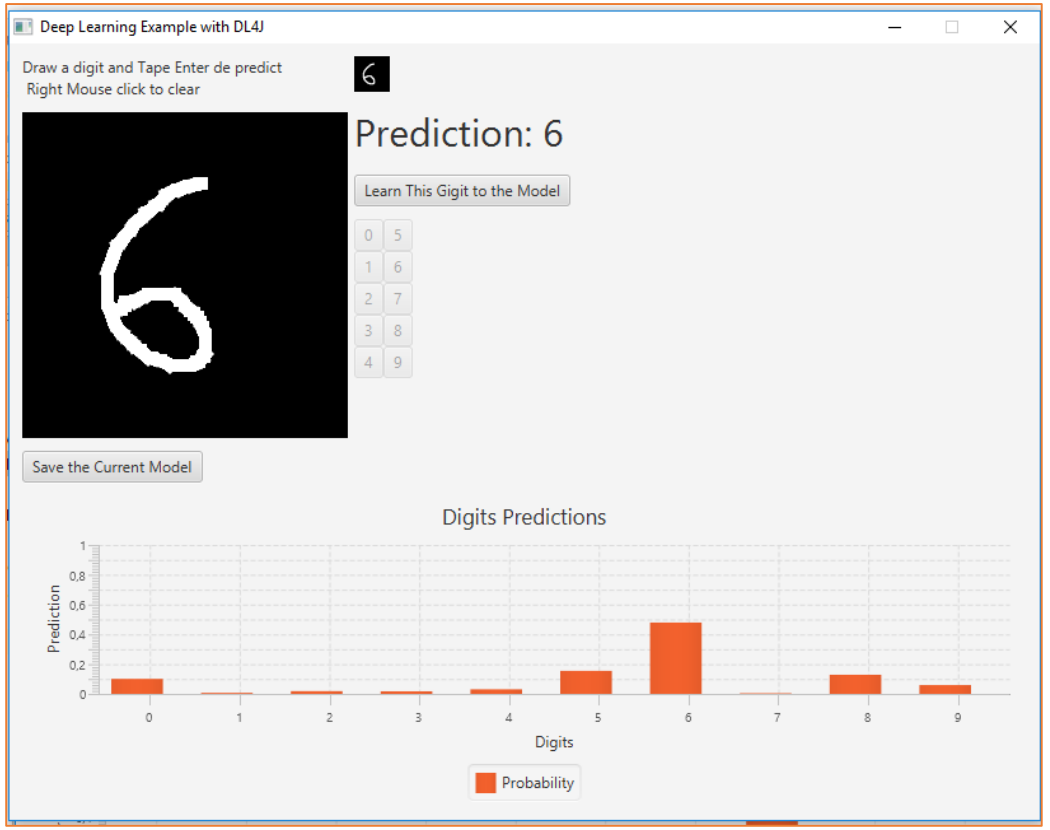
# UI Application avec JavaFX : Commencer le dessin

```
canvas.setOnMousePressed(e -> {
        ctx.setStroke(Color.WHITE);  ctx.beginPath();
        ctx.moveTo(e.getX(), e.getY());
        ctx.stroke();
        canvas.requestFocus();
 });
```



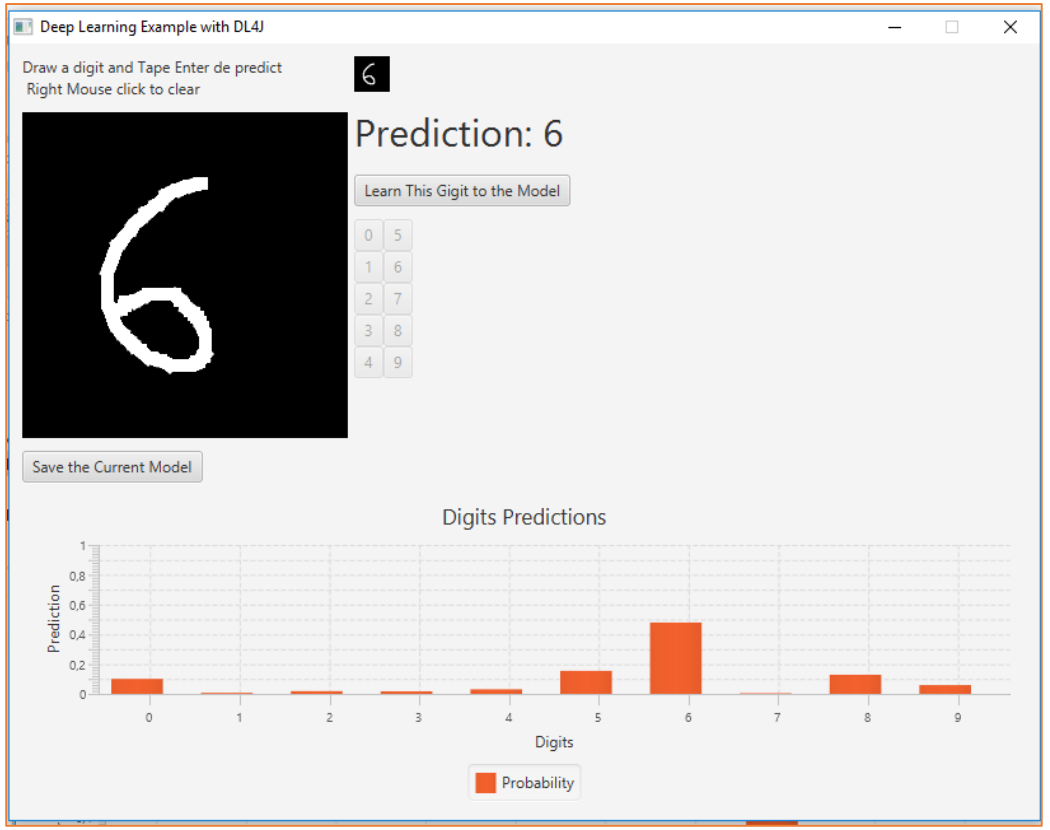
# UI Application avec JavaFX : Dessiner

```
canvas.setOnMouseDragged(e -> {
        ctx.setStroke(Color.WHITE);
        ctx.lineTo(e.getX(), e.getY());
        ctx.stroke();
        canvas.requestFocus();
});
```



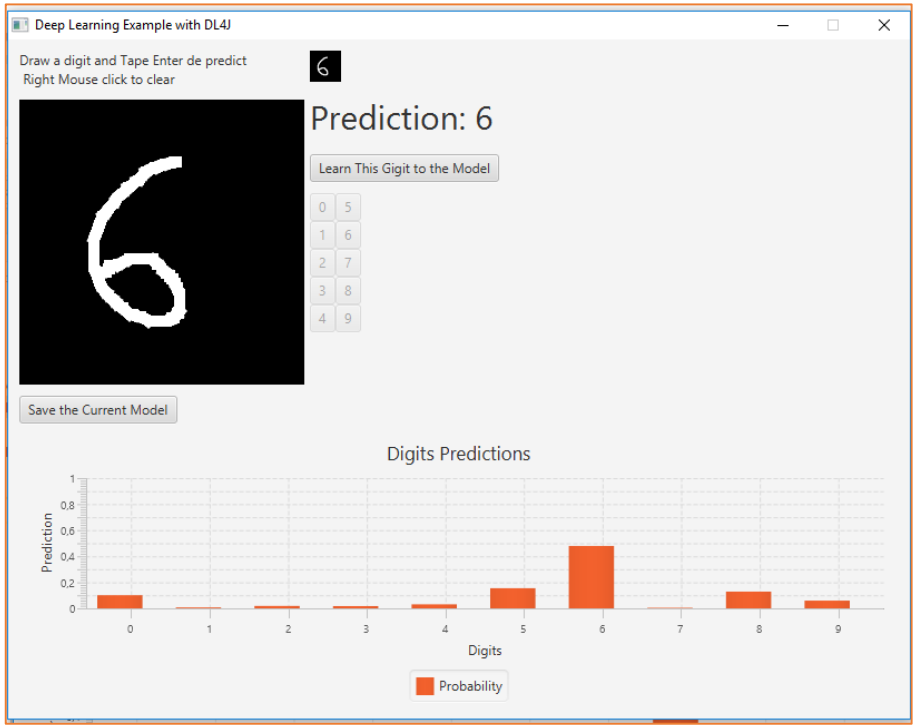
# UI Application avec JavaFX : Effacer le canvas

```
canvas.setOnMouseClicked(e -> {
      if (e.getButton() == MouseButton.SECONDARY) {
        clear(ctx);
      }
      canvas.requestFocus();
});

private void clear(GraphicsContext ctx) {
    ctx.setFill(Color.BLACK);
    ctx.fillRect(0, 0, 300, 300);
}
```



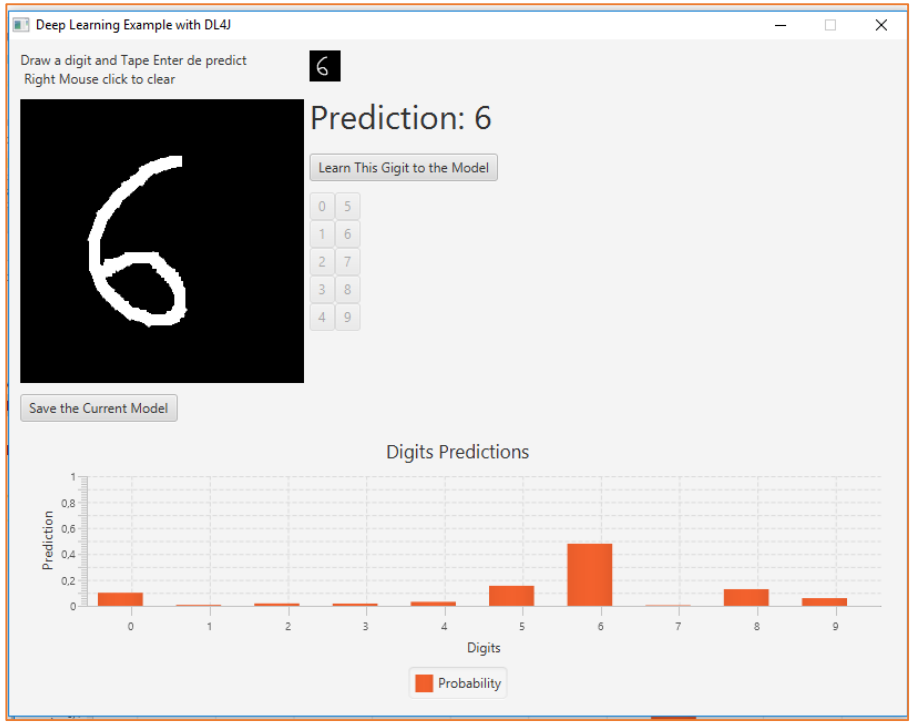
# UI Application avec JavaFX : Prédiction de l'image

```
canvas.setOnKeyReleased(e -> {
      if (e.getCode() == KeyCode.ENTER) {
        BufferedImage scaledImg = getScaledImage(canvas);
        imgView.setImage(SwingFXUtils.toFXImage(scaledImg, null));
        try {
          predictImage(scaledImg, lblResult);
        } catch (Exception e1) {
          e1.printStackTrace();
        }
      }
    });
    clear(ctx);
    canvas.requestFocus();
```



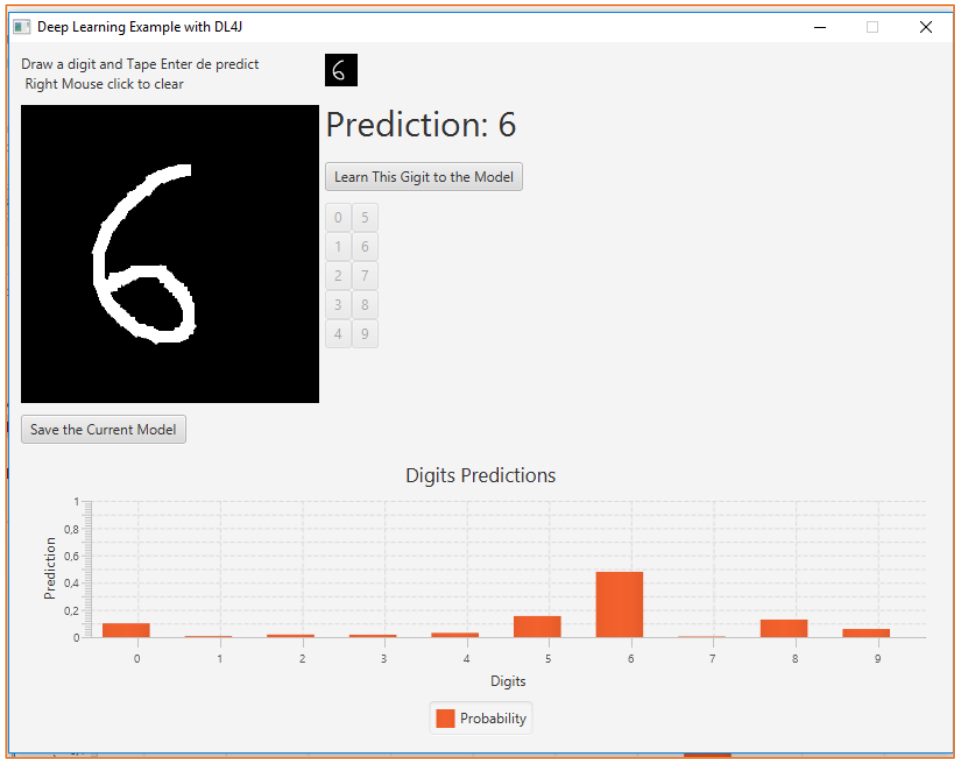
# UI Application avec JavaFX : Prédiction de l'image dessinée

```
private void predictImage(BufferedImage img, Label lbl) throws IOException {
    NativeImageLoader loader = new NativeImageLoader(28, 28, 1, true);
    currentImage = loader.asRowVector(img);
    ImagePreProcessingScaler scaler = new ImagePreProcessingScaler(0, 1);
    scaler.transform(currentImage);
    INDArray output = net.output(currentImage);
    //lbl.setText("Prediction: " + net.predict(image)[0] + "\n " + output);
    lbl.setText("Prediction: " + net.predict(currentImage)[0]);
    double[] d=output.toDoubleVector();
    for(int i=0;i<data.size();i++){
        data.get(i).setYValue(d[i]);
    }
  }
}
```



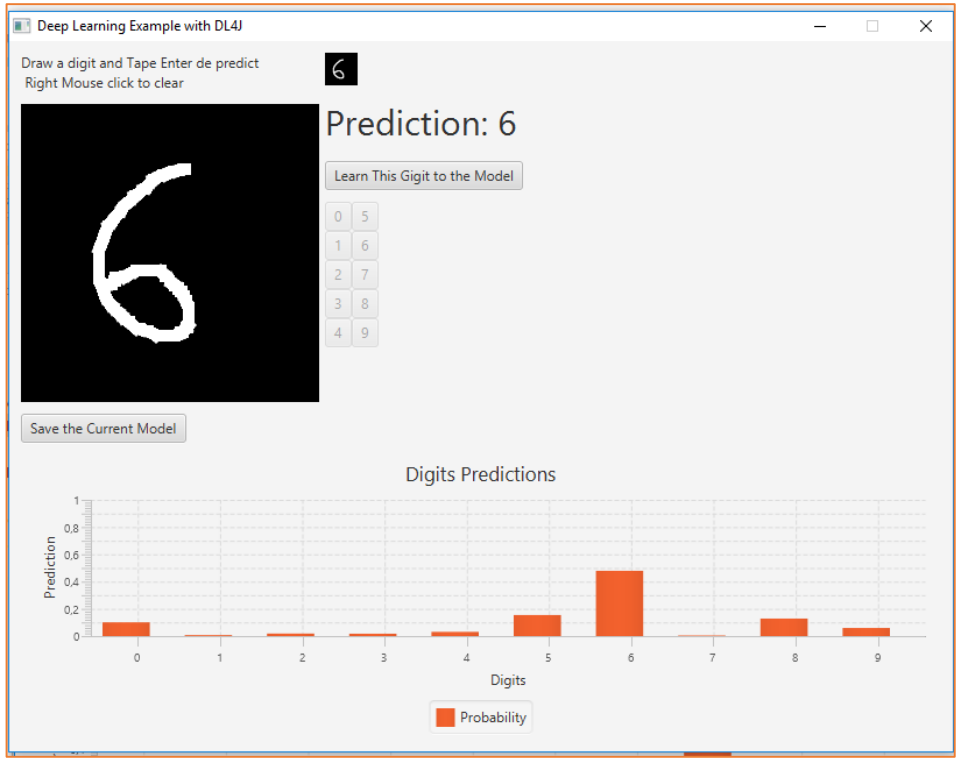
# UI Application avec JavaFX : Activer les boutons pour corriger la pérdiction

```
buttonLearn.setOnAction(evt->{
        if(currentImage!=null) gridPane.setDisable(false);
 });
```



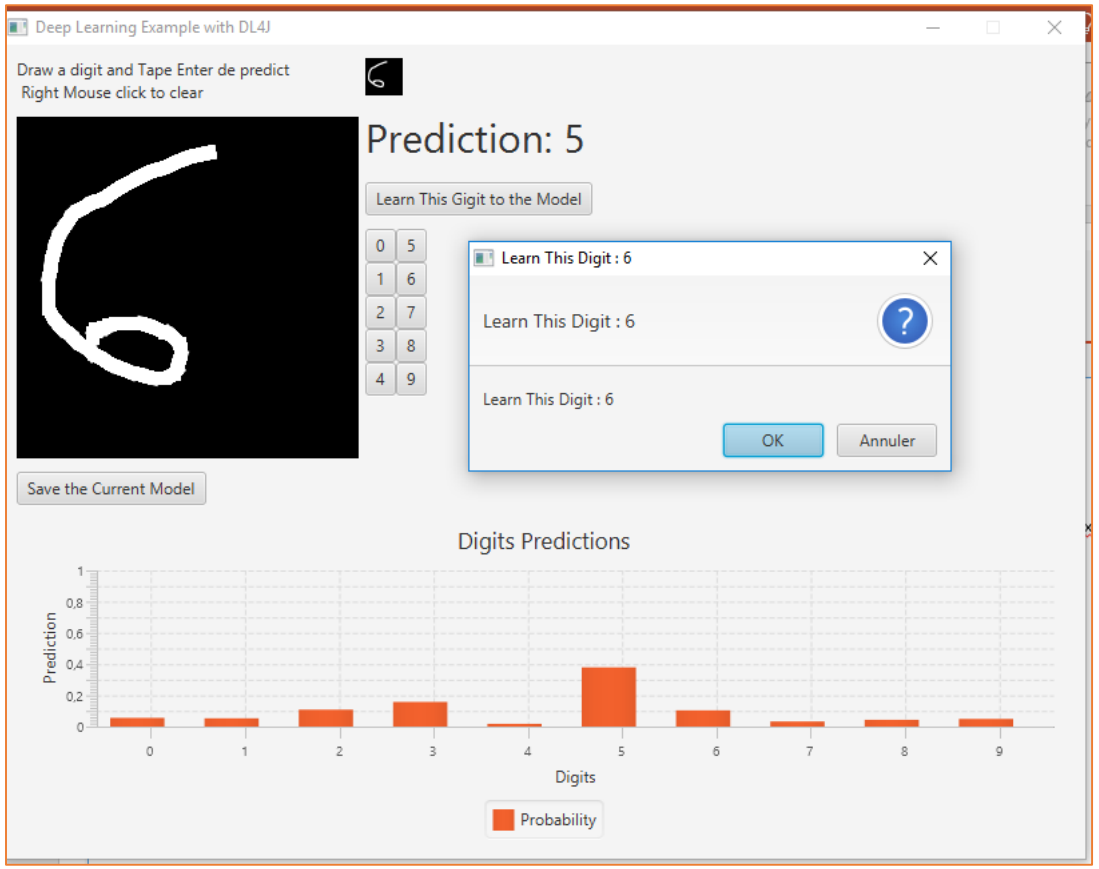
# UI Application avec JavaFX : Enregistrer le modèle

```
buttonSaveModel.setOnAction(evt->{
      try {
        ModelSerializer.writeModel(net,new File(basePath+"/model.zip"),true);
      } catch (IOException e) {
        e.printStackTrace();
      }
    });
  }
```



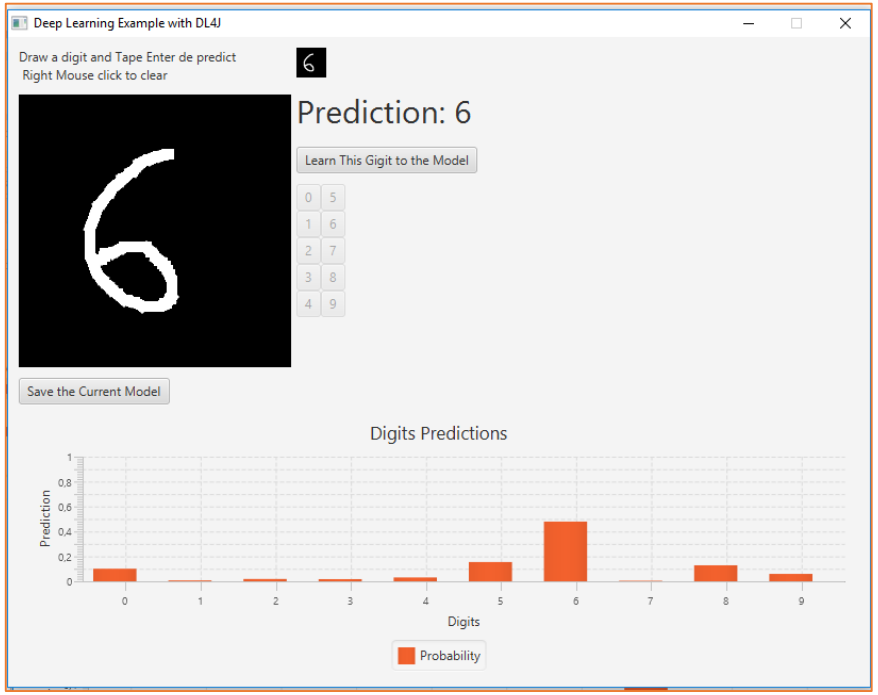
# UI Application avec JavaFX : Apprendre une nouvelle forme

```
private void learnThisDigitToModel(ActionEvent evt) {
    Button b=(Button) evt.getSource();
    int digit=Integer.parseInt(b.getText());
    Alert alert = new Alert(Alert.AlertType.CONFIRMATION);
    alert.setTitle("Learn This Digit : "+digit);
    alert.setHeaderText("Learn This Digit : "+b.getText());
    alert.setContentText("Learn This Digit : "+b.getText());
    Optional<ButtonType> result = alert.showAndWait();
    double[] digits=new double[10];
    digits[digit]=1;
    if (result.get() == ButtonType.OK){
      INDArray ys= Nd4j.create(digits);
      net.fit(currentImage,ys);
    } else { }
    canvas.requestFocus();
  }
```

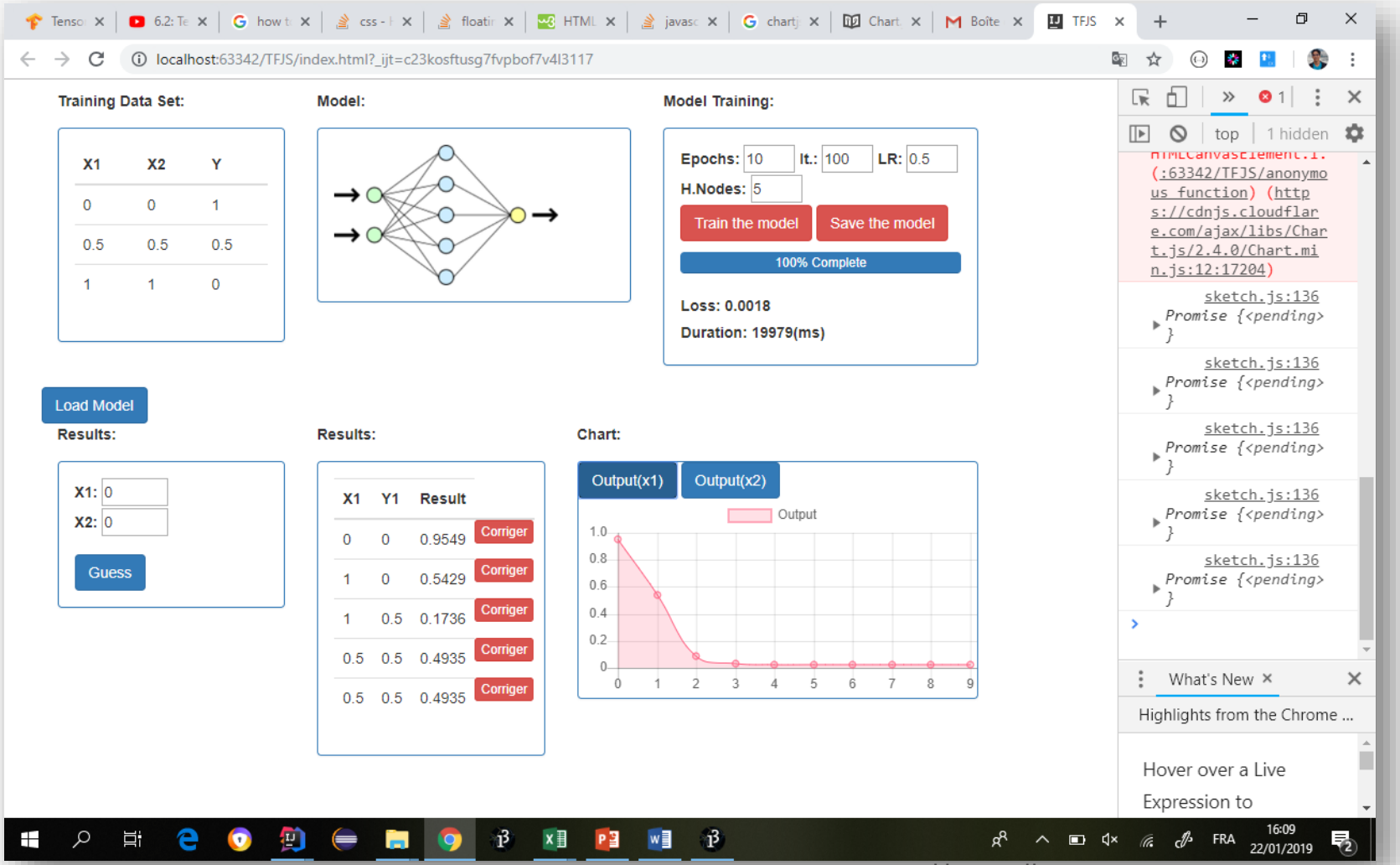


# UI Application avec JavaFX : Création de l'imge à partir du canvas

```
private BufferedImage getScaledImage(Canvas canvas) {
    WritableImage writableImage = new WritableImage(canvasWidth, canvasHeight);
    canvas.snapshot(null, writableImage);
    Image tmp = SwingFXUtils.fromFXImage(writableImage, null).getScaledInstance(28, 28, Image.SCALE_SMOOTH);
    BufferedImage scaledImg = new BufferedImage(28, 28, BufferedImage.TYPE_BYTE_GRAY);
    Graphics graphics = scaledImg.getGraphics();
    graphics.drawImage(tmp, 0, 0, null);
    graphics.dispose();
    return scaledImg;
  }
```



# Application TensorFlow.JS : https://github.com/mohamedYoussfi/machines-learning-tensorflow-js



# Exemple de classification : DataSet IRIS

https://github.com/mohamedYoussfi/machines_learning_TFJS_IRIS

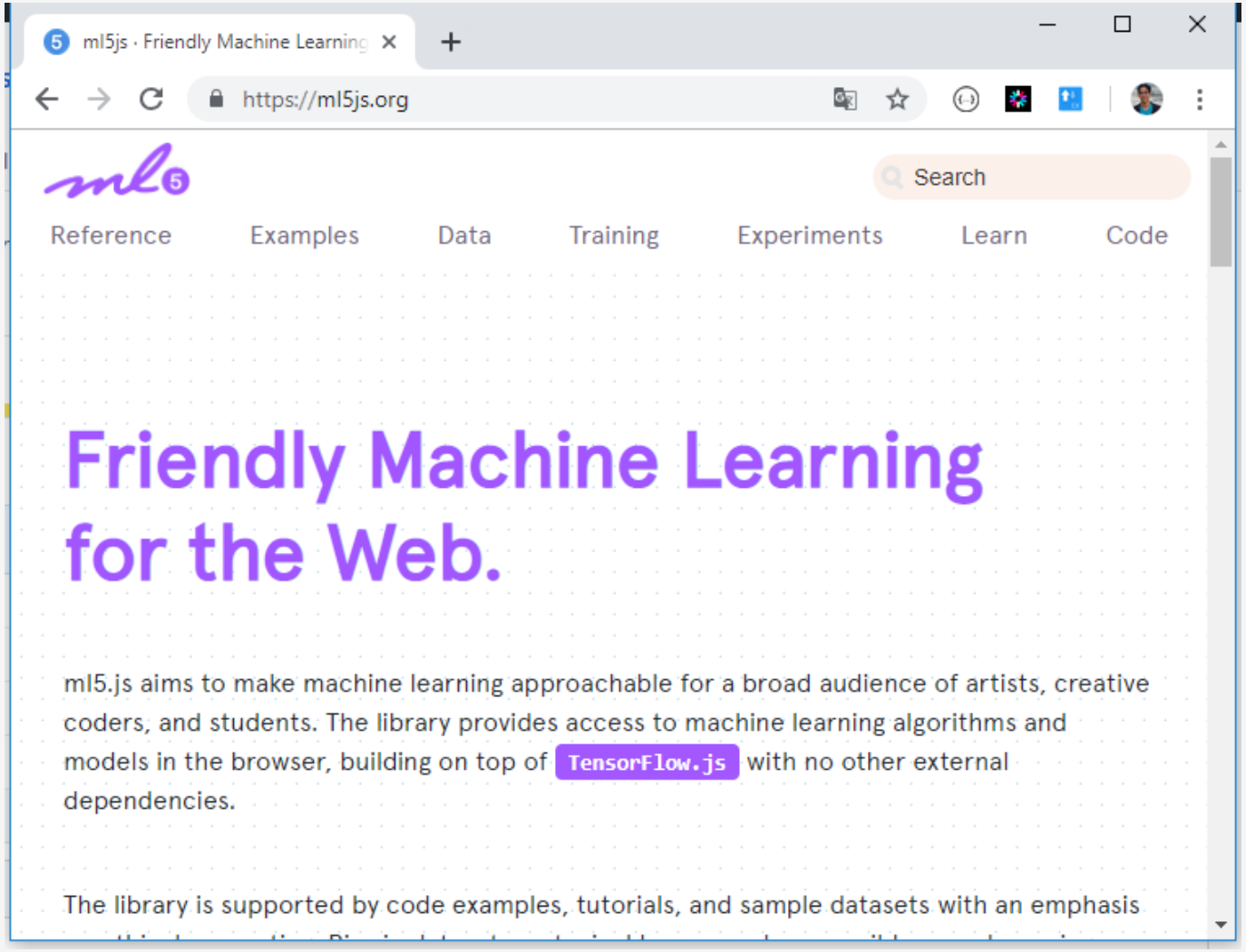
Iris Plants Database

- Attribute Information:
- 1. sepal length in cm
- 2. sepal width in cm
- 3. petal length in cm
- 4. petal width in cm
- 5. class:
- -- Iris Setosa
- -- Iris Versicolour
- -- Iris Virginica

```
5.1,3.5,1.4,0.2,Iris-setosa
4.9,3.0,1.4,0.2,Iris-setosa
4.7,3.2,1.3,0.2,Iris-setosa
4.6,3.1,1.5,0.2,Iris-setosa
....
7.0,3.2,4.7,1.4,Iris-versicolor
6.4,3.2,4.5,1.5,Iris-versicolor
6.9,3.1,4.9,1.5,Iris-versicolor
5.5,2.3,4.0,1.3,Iris-versicolor
6.5,2.8,4.6,1.5,Iris-versicolor
…
6.3,3.3,6.0,2.5,Iris-virginica
5.8,2.7,5.1,1.9,Iris-virginica
7.1,3.0,5.9,2.1,Iris-virginica
6.3,2.9,5.6,1.8,Iris-virginica
6.5,3.0,5.8,2.2,Iris-virginica
…….
```

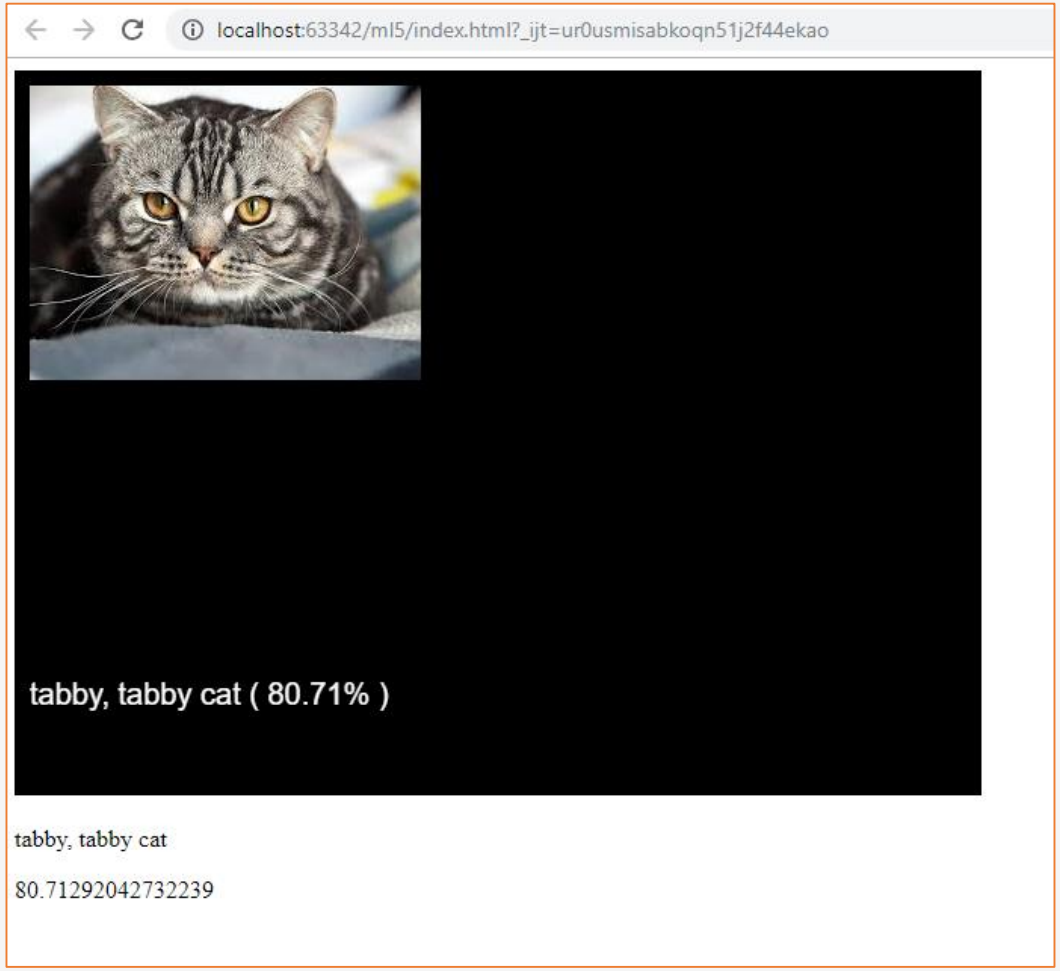
# ML5.JS

- ml5.js est une librairie de haut niveau qui s'appuie sur TensorFlow.js
- Elle permet de faciliter la création des application de Machines et Deep Learning



# Exemple d'application avec ML5.JS : Réutilisation de modèles pré-entrainés

Classifieur d'images : **MobileNet**



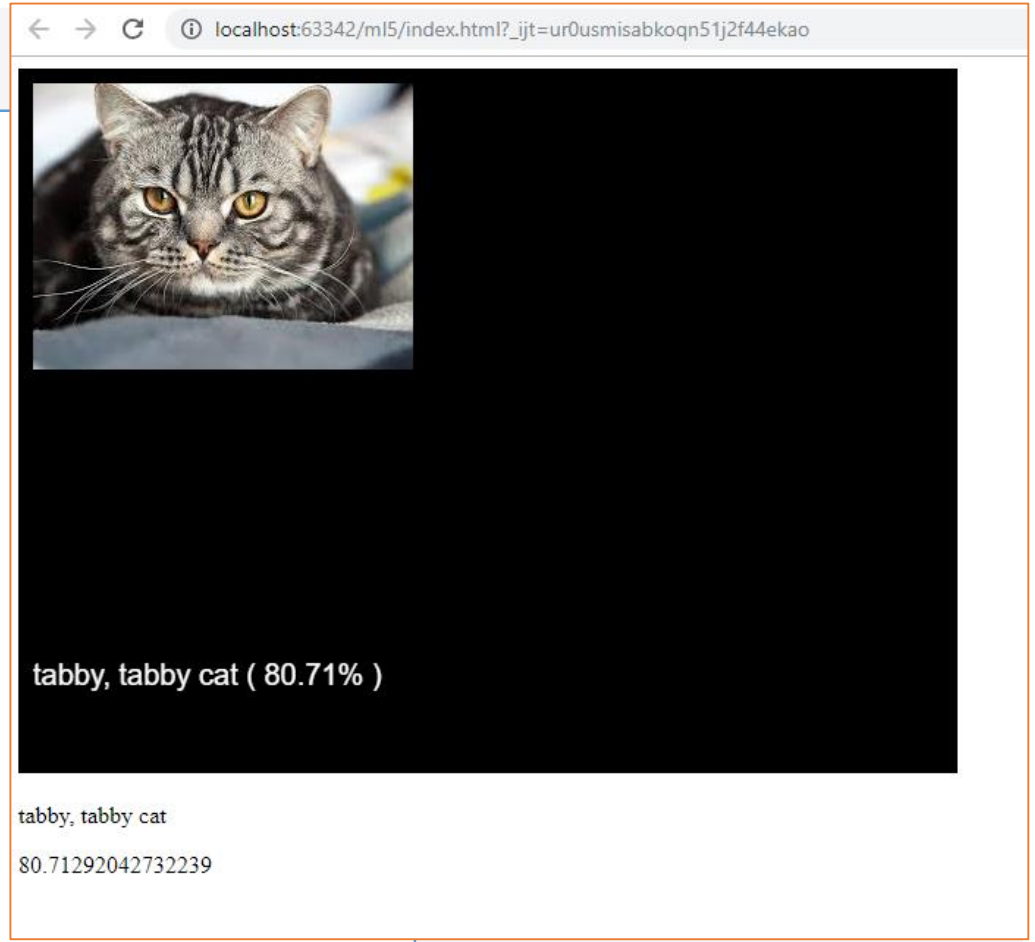

- MobileNets, Présente des modèles de réseaux de neurones convolutionnels (CNN) pré-entrainés basés sur TensorFlow :
  - Détection d'objet (personne, bus, avion, voiture, chien, chat, écran TV, etc…)
  - Visages (attributs d'un visage notamment)
  - Classification « fine-grained » , c'est-à-dire entre des espèces d'une même race (chien, chat, rouge-gorge, …)
  - Reconnaissance de lieux et de paysages
- Caractéristiques :
  - Extrêmement *légers* et petits (en termes de ligne de code et de poids des modèles)
  - Infiniment *rapides*
  - P*récision redoutable*, surtout pour les ressources consommées
  - Facilement *configurables* pour améliorer la précision de la détection
  - *Dédiés à l'embarqué et aux smartphones*, pour déporter les calculs comme on l'a vu avec l'avènement d'AngularJS et du Front Office

# Index.html

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>ML5</title>
    <script src="https://cdnjs.cloudflare.com/ajax/libs/p5.js/0.7.3/p5.js"></script>
    <script src="https://cdnjs.cloudflare.com/ajax/libs/p5.js/0.7.3/addons/p5.dom.min.js"></script>
    <script src="app.js"></script>
    <script src="https://unpkg.com/ml5@0.1.1/dist/ml5.min.js"></script>
</head>
<body>

</body>
</html>
```

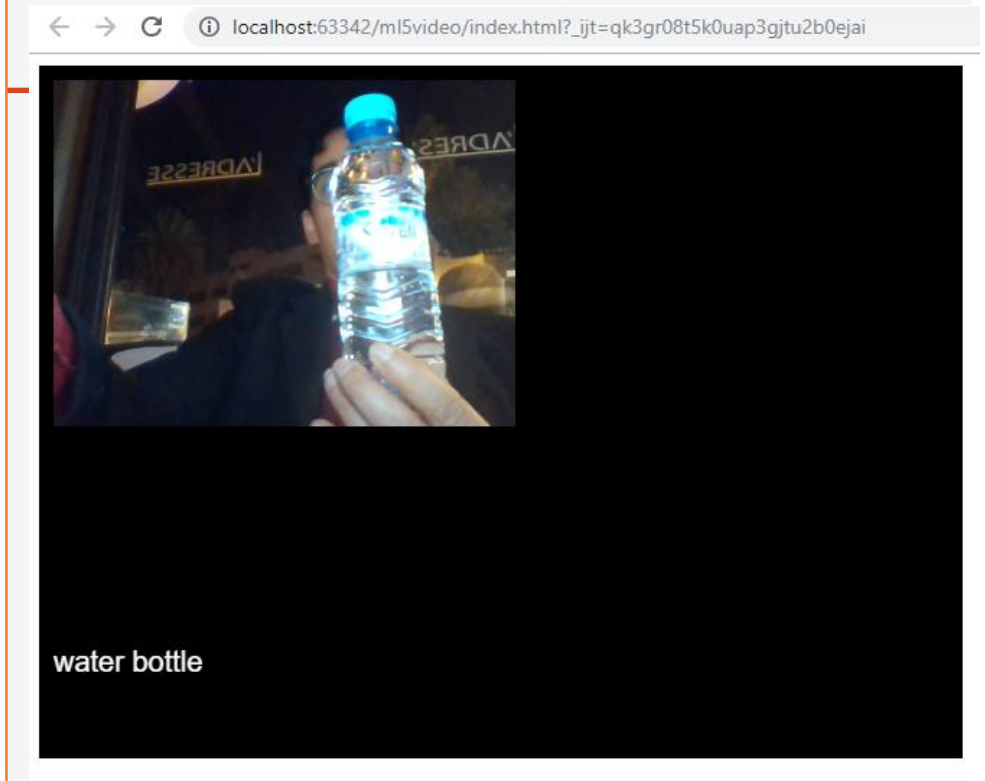
# app.js

```
let mobilenet
let image;
function setup(){
    createCanvas(640,480);
    image=createImg('images/cat3.jpg',function(){
      image(image,10,10);
     }); image.hide();      background(0);

mobilenet=ml5.imageClassifier('MobileNet', function(){
  mobilenet.predict(image,function(err,result){
        if(err)   console.log(err);
        else{
          console.log(result);    let label=result[0].className;
          let p=result[0].probability*100;  fill(255);
          textSize(20);  text(label+" ( "+p.toFixed(2)+"% )",10,height-60);
          createP(label); createP(p);
}}})})
}
```

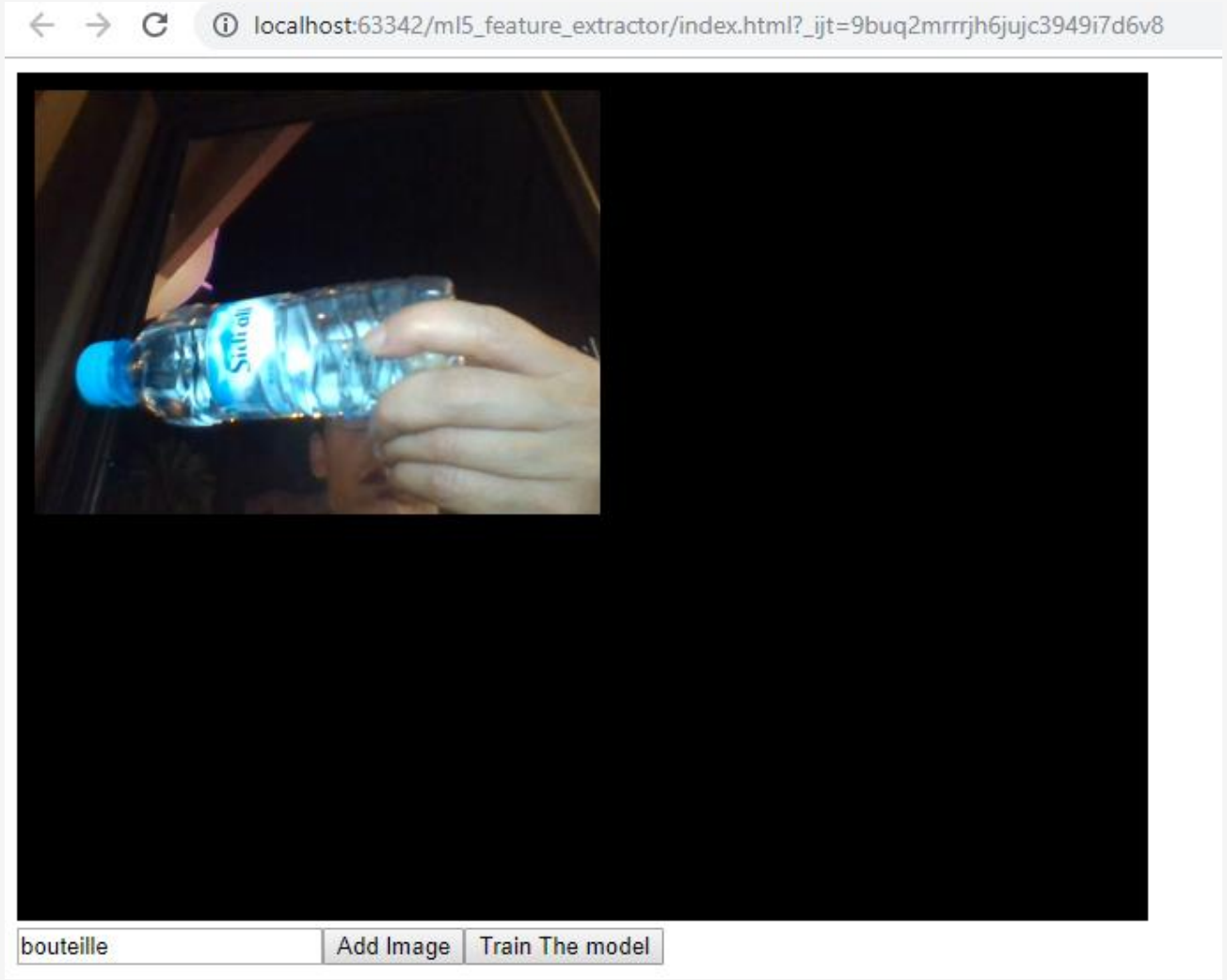


# Version avec Vidéo

```
let mobilenet; let video; let label="";
function setup(){
 createCanvas(640,480);
 video=createCapture(VIDEO); video.hide();
 background(0);
 mobilenet=ml5.imageClassifier('MobileNet',video,ready);
}
function ready(){  mobilenet.predict(gotResults); }
function gotResults(err,results){
   if(err) console.log(err);
    else{
      label=results[0].className;
      let p=results[0].probability*100;
      mobilenet.predict(gotResults);
    }
}
function draw(){
  background(0); image(video,10,10,320,240);
  fill(255); textSize(20);
  text(label,10,height-60);
}
```



# Extraction des caractéristiques avec MobileNet : Images Extraite d'une vidéo



# Index.html

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>ML5</title>
    <script
src="https://cdnjs.cloudflare.com/ajax/libs/p5.js/0.7.3/p5.js"></script>
    <script
src="https://cdnjs.cloudflare.com/ajax/libs/p5.js/0.7.3/addons/p5.dom.min.js"></sc
ript>
    <script src="sketch.js"></script>
    <script src="https://unpkg.com/ml5@0.1.1/dist/ml5.min.js"></script>
</head>
<body>

</body>
</html>
```

# Version avec Vidéo

```
let mobilenet; let video; let label="";
let classifier; let addImageButton; let textInputLabel;
let imagesCount=0; let trainButton;

 function setup(){
     createCanvas(640,480);  video=createCapture(VIDEO);
     video.hide();  background(0);
     mobilenet=ml5.featureExtractor('MobileNet',modelReady);
     textInputLabel=createInput("bouteille");
     addImageButton=createButton("Add Image");
     trainButton=createButton("Train The model");
     addImageButton.mousePressed(function(){
         let labelValue=textInputLabel.value();
        classifier.addImage(labelValue);  ++imagesCount;
        console.log(imagesCount+" : "+labelValue);
     });
     trainButton.mousePressed(function(){
        classifier.train(function(loss){
            if(loss==null){
                console.log("Model trained");
                classifier.classify(gotResults());
            }
            else{    console.log(loss);   }
        })});
 }
```

```
function draw(){
 background(0);
 image(video,10,10,320,240);
    fill(255);
    textSize(20);
    text(label,10,height-60);
}


function modelReady(){
    console.log("model ready");
    classifier=mobilenet.classification(video,videoReady);
    //mobilenet.predict(gotResults);
}


function videoReady(){
    console.log("video ready");
}


function gotResults(err,results){
    if(err)
        console.log(err);
    else{
        label=results;
        mobilenet.classify(gotResults);
    }
}
```

# Quelques Applications de Deep Learning

- Reconnaissance de formes
- Reconnaissance vocale
- Traduction vocale
- Modèles génératifs (Donner une série d'images à un réseaux profond et le forcer à y voir autres choses)
- Etc.

http://www.divertissonsnous.com/2017/11/24/un-systeme-de-reconnaissance-des-formes-applique-a-times-square/


https://deepdreamgenerator.com/


# Références :


- **Yann LeCun** : https://www.youtube.com/watch?v=2WiPx6thH2E
- https://deeplearning4j.org/
- https://js.tensorflow.org/
- **Daniel Shiffman** : https://www.youtube.com/watch?v=XJ7HLz9VYz0&list=PLRqwX-V7Uu6Y7MdSCaIfsxc561QI0U0Tb&index=1
- https://ml5js.org/
- https://keras.io/
- https://p5js.org/
- https://www.lafabriquedunet.fr/blog/definition-data-science/
- https://www.youtube.com/watch?v=oU7tSpvDOIs
- https://www.youtube.com/watch?v=trWrEWfhTVg

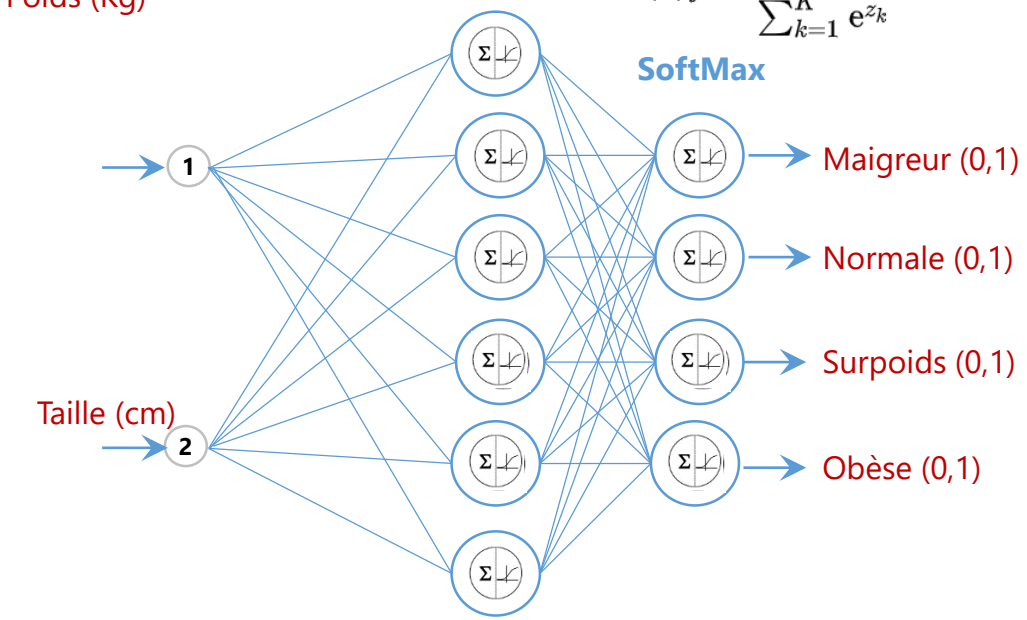
te

$$f(x) = \frac{1}{1 + e^{-x}}$$

Sigmoid

$$\sigma(\mathbf{z})_j = \frac{e^{z_j}}{\sum_{k=1}^{K} e^{z_k}}$$

SoftMax




Learning Rate : 0.001  Optimiser : ADAM

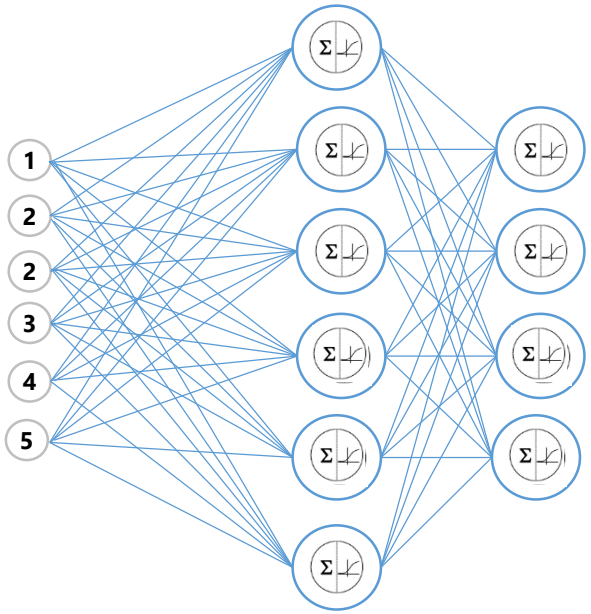
Loss function : MearnSquaredError

$$\mathrm{MSE} = \frac{1}{n} \sum_{i=1}^{n} (y_i - \tilde{y}_i)^2$$

te

$$f(x) = \frac{1}{1 + e^{-x}}$$

**SoftMax** Poids (Kg)




Learning Rate : 0.001 Optimiser : ADAM

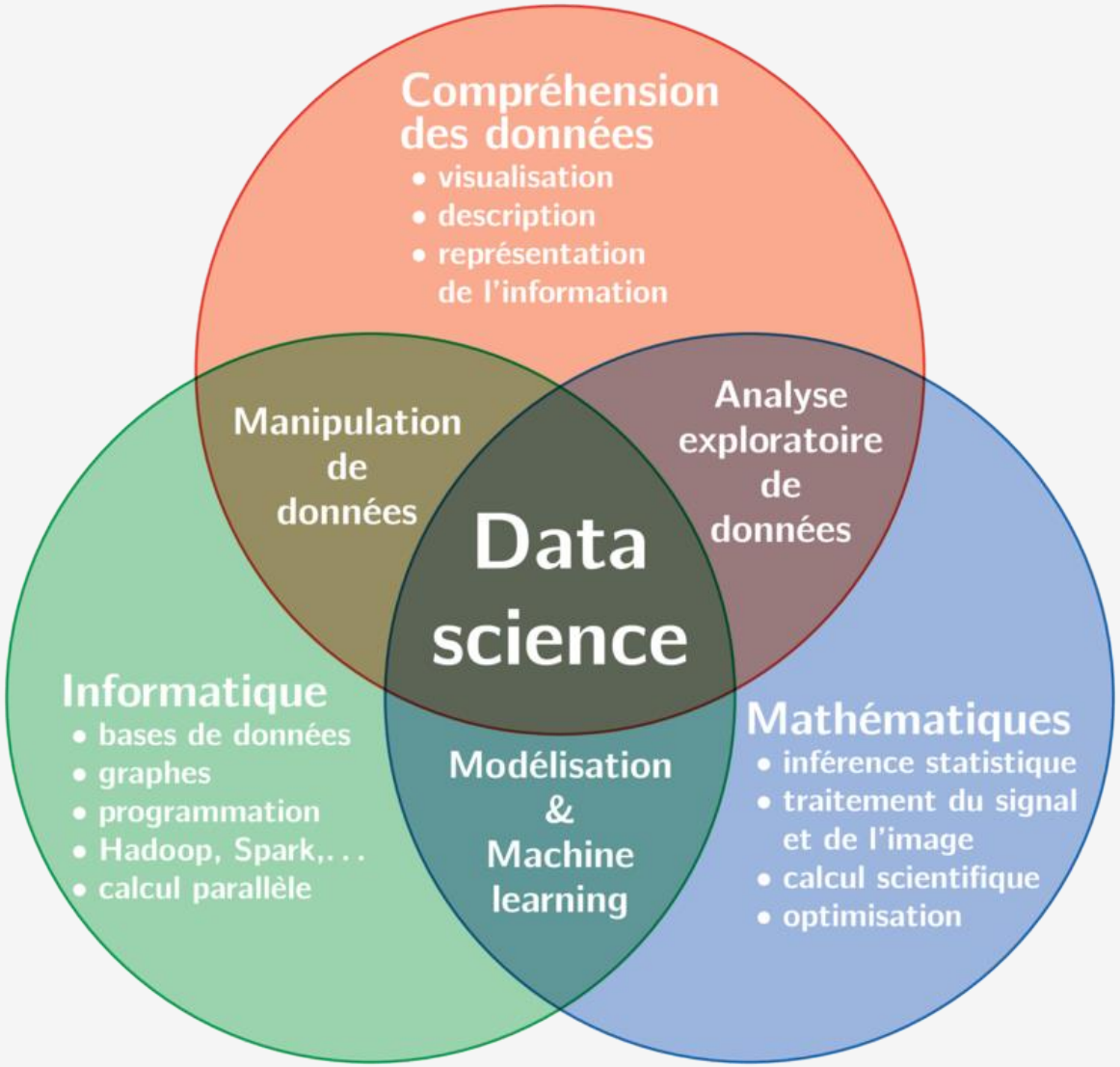
Loss function : MearnSquaredError

$$\mathrm{MSE} = \frac{1}{n}\sum_{i=1}^{n}(y_i - \tilde{y}_i)^2$$

# Data science

*Automatisation et Généralisation de l'extraction des connaissance à partir des données massives et complexes provenant des sources de données hétérogènes*

# Métier Data Scientiste

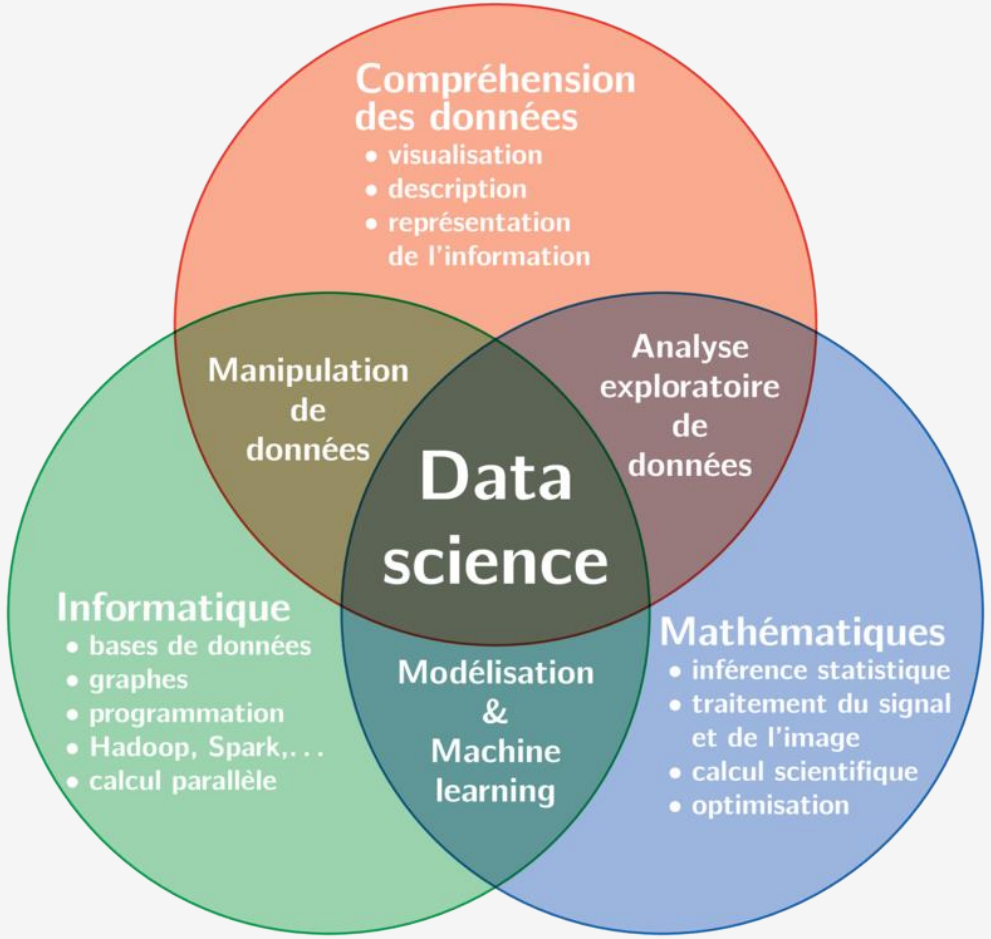



- **Objectif du parcours Data Science** :
  - Générer, Explorer, Traiter les données
  - Visualiser et analyser les données,
  - Concevoir et analyser des modèles et des algorithmes de
    - Traitements de données.
    - Apprentissage automatique
    - Prédictions
- **Prérequis :**
  - Mathématiques appliquées (Probabilités, Optimisation, Algèbre linéaire, Recherche opérationnelle)
  - Statistique et optimisation statistique
  - Informatique:
    - Algorithmique et Programmation
    - Bases de données et Big Data
    - Technologie Web et Mobile
    - IOT
  - Traitement du signal et de l'image.
  - Intelligence artificielle :
    - Machine et Deep Learning,

# Métier Data Scientiste

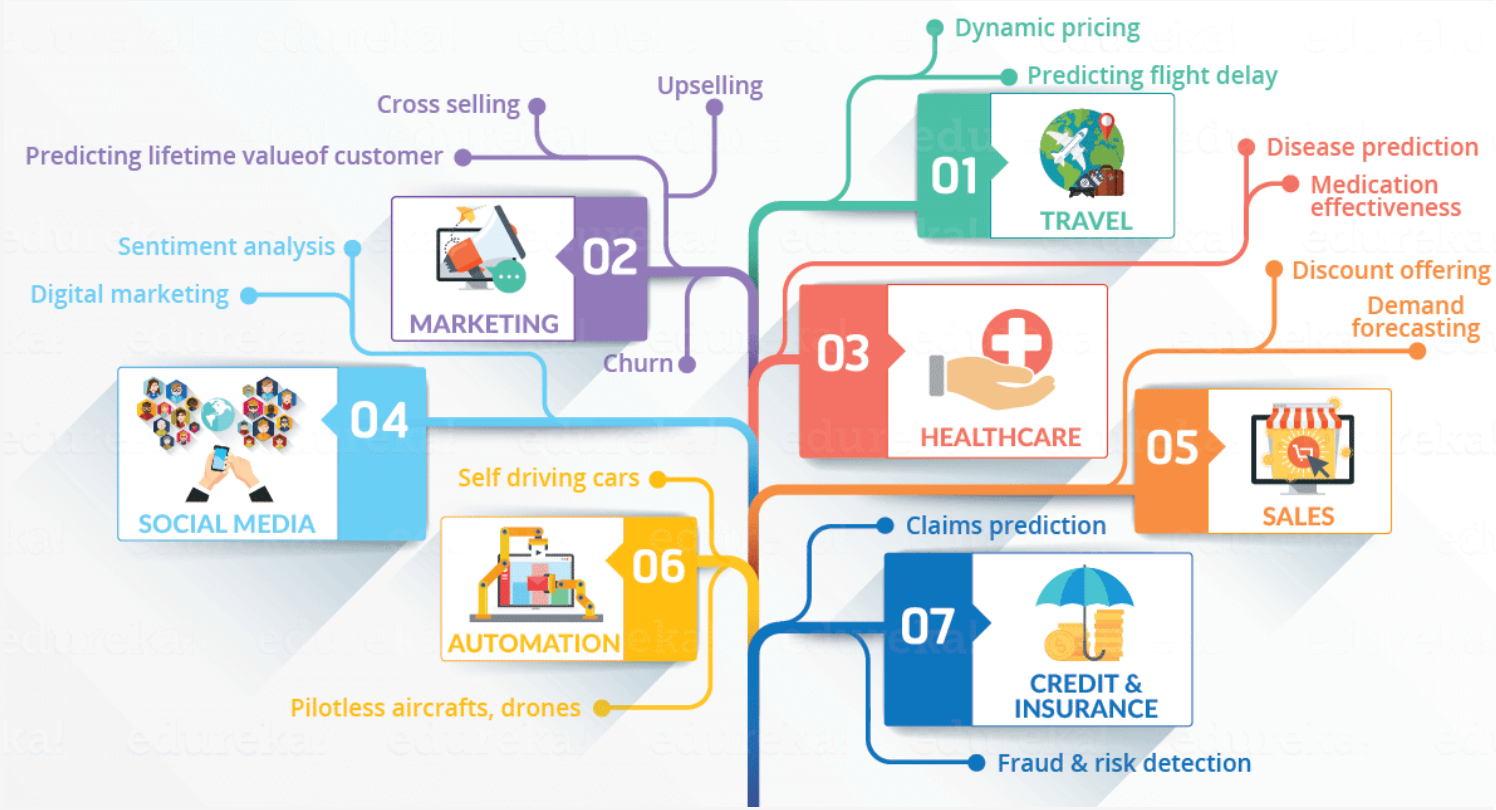
Le métier de data Scientist est apparu pour trois raisons principales :

- L'explosion de la quantité de données produites et collectées (Big Data, IOT, Applications Mobiles)
- L'amélioration et l'accessibilité plus grande des algorithmes de machine Learning ( Frameworks de Machines Learning)
- L'augmentation exponentielle des capacités de calcul des ordinateurs
    - Architectures massivement parallèles (GPUs)
    - Middlewares pour les Systèmes Distribués

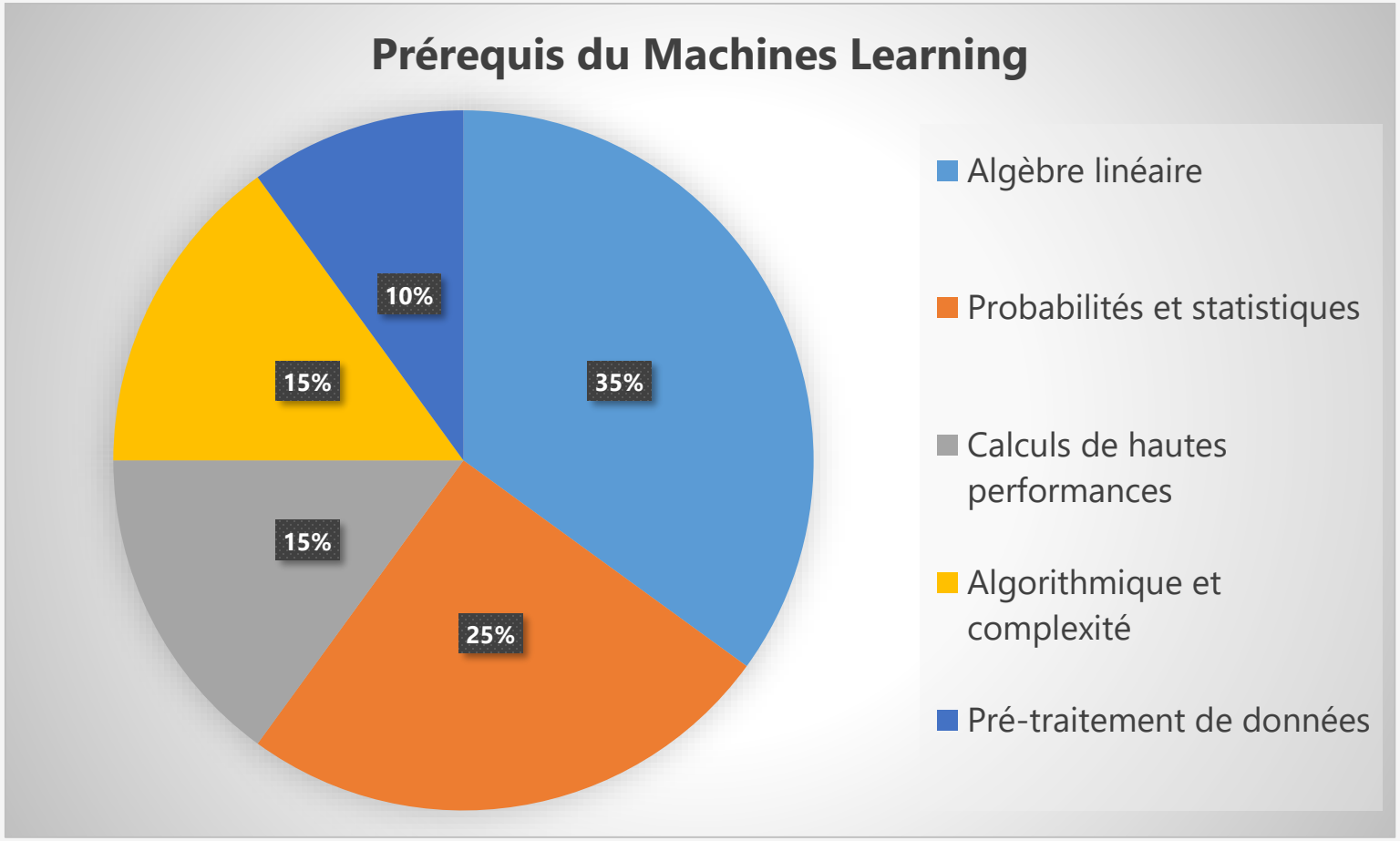


# Métier Data Scientiste

- Un Baromètre LinkedIn de Juillet 2017 indique que :
  - **Le métier de  Data Scientist** est l'un des métiers les plus recherchés et les plus rémunérés
- En terme de salaire, à la sortie de l'école un data Scientiste peut toucher un **salaire mensuel** autour de **4000 Euro** (En Europe)
- Le nombre de lauréats dans cette discipline est encore très faible
- Tous les secteurs sont concernés (Médecine, Agriculture, Justice, Education, Industrie , Finance, Politique, etc..)



# Prérequis du Machines Learning



# Evénement important : Compétitions de reconnaissance des images : Large Scale Visual Recognition Challenge

2010 :

- NEC : 28%
- XRCE : 34%
- ISIL : 45%
- UCI : 47%
- Hminmax : 54%

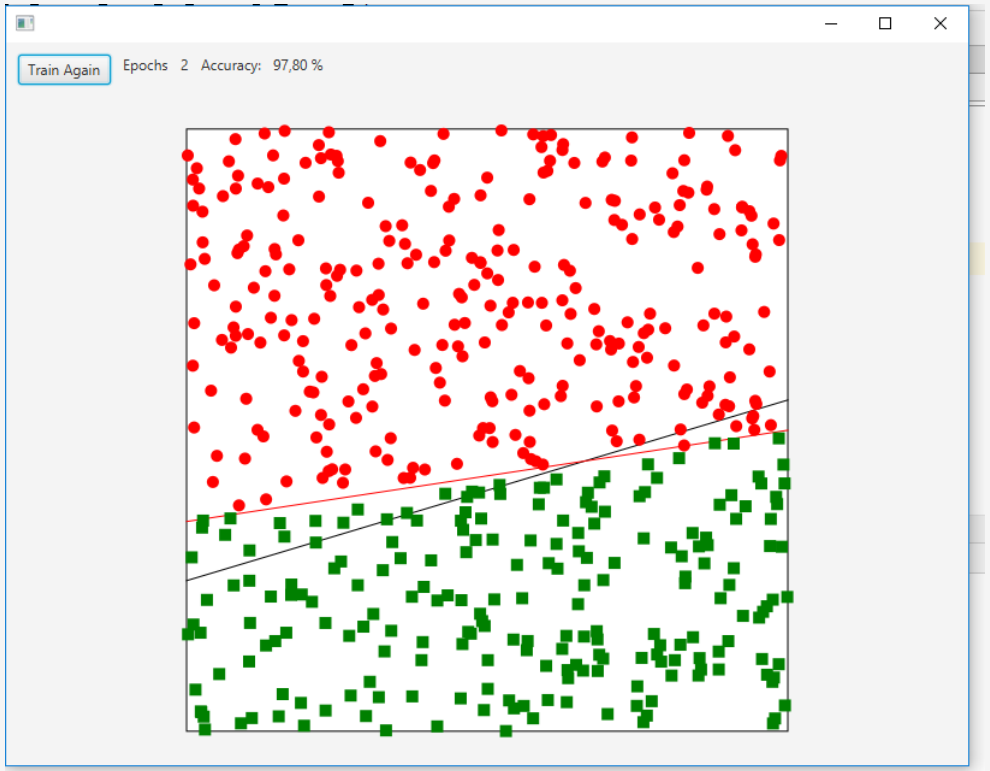
- 2011 :
  - XRCE : 26%
  - UV A : 31%
  - ISI : 36%
  - NII : 50%

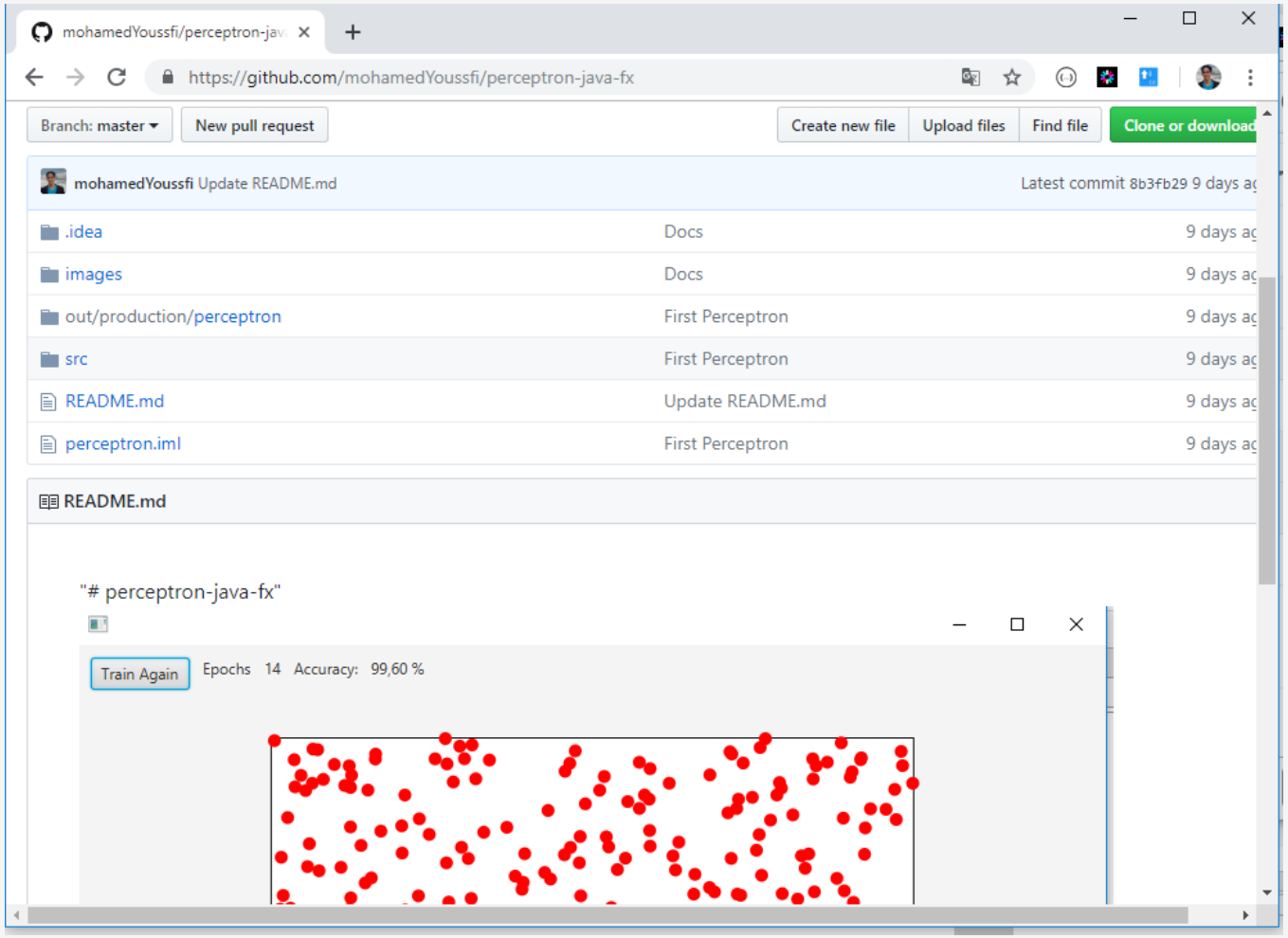
- 2012 : **(Algo de Deep Learning l'emporte)**
  - Super Vision : 16%  (DL)
  - ISI : 26%
  - VGG : 27%
  - XRCE :  27 %
  - UV A :  30%

- 2013 : **(Algos de Deep Learning écrase la compétition)**
  - Clarifi (DL) : 12%
  - NUS (DL) : 13%
  - Zeiler fergus (DL) : 13%
  - A.HOWARD (DL) : 13%
  - OverFeat (DL) : 14%

# Exemple pour débuter: Implémentation java du perceptron

https://github.com/mohamedYoussfi/perceptron-java-fx

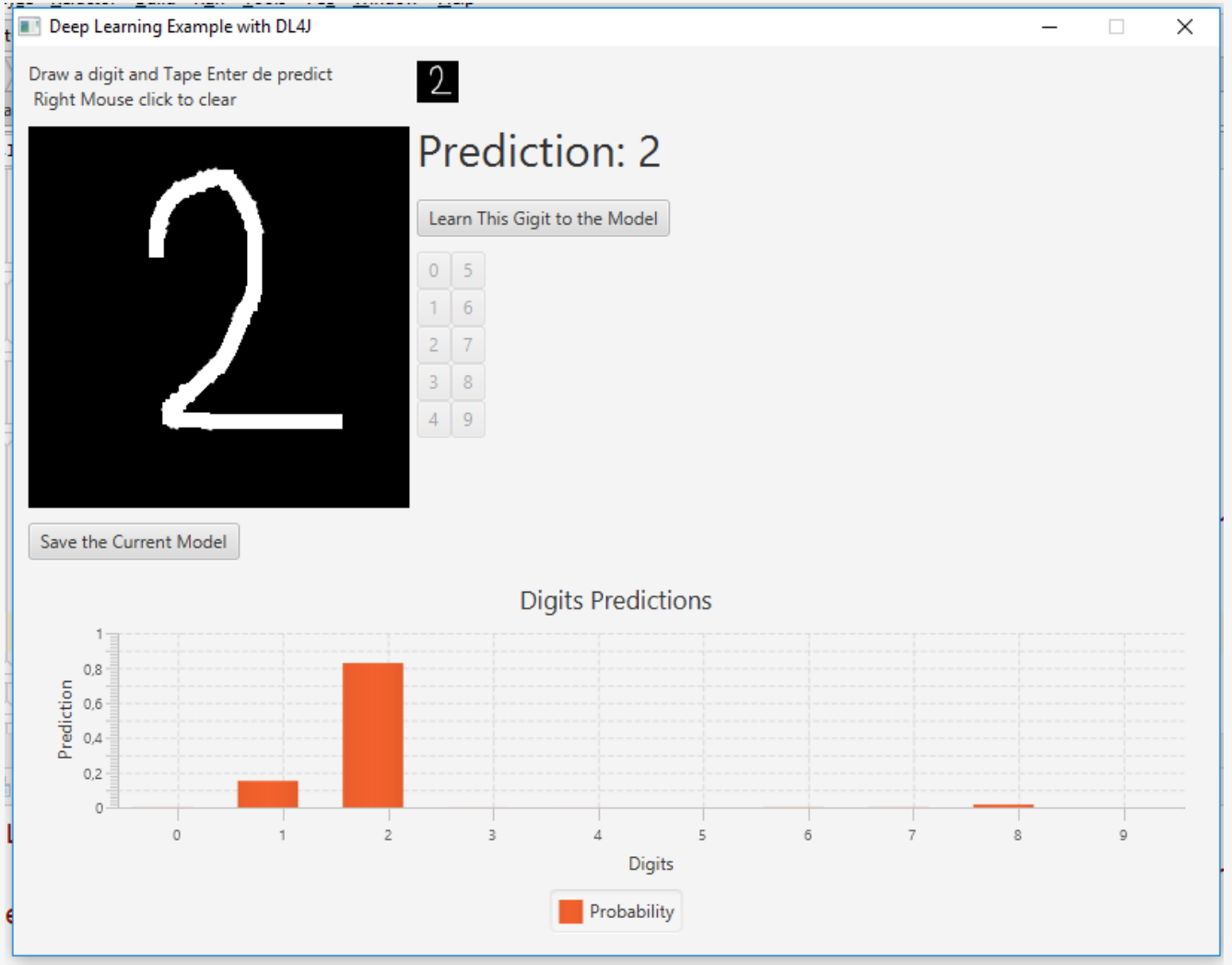


# Application 1 : Classification des images en utilisant CNN

https://github.com/mohamedYoussfi/deeplearning4j-cnn-mnist-app

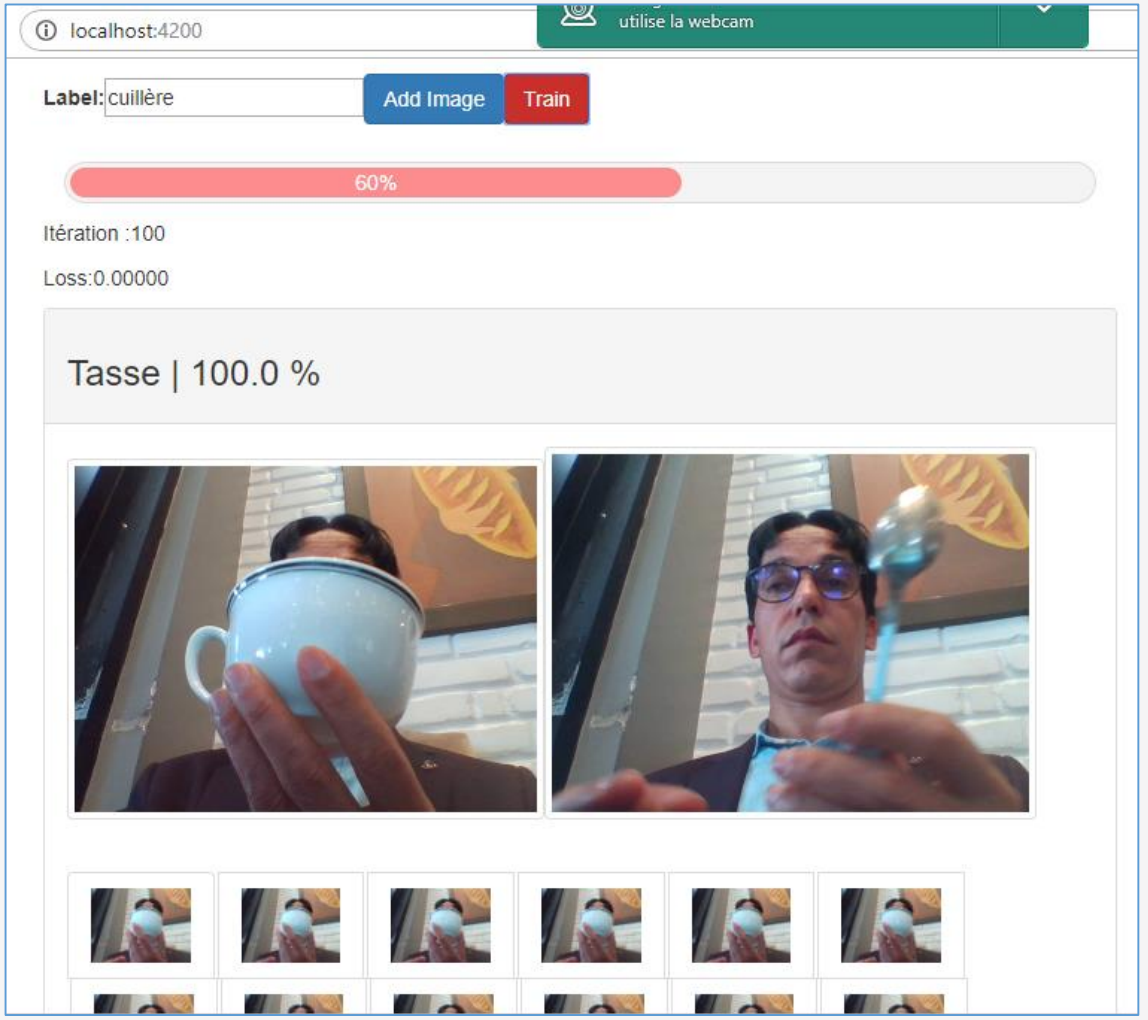
- L'objectif de cette application étant de créer un modèle de reconnaissance des formes en utilisant Deep Learning avec les Réseaux de neurones à convolution CNN.
- Pour l'entrainement du réseaux, nous utilisons le data set Mninst. Dans cet exemple on s'intéresse à reconnaitre les digits de 0 à 9 dessiné manuellement dans une image.



https://github.com/mohamedYoussfi/angular-ml5.js-mobilenet-feature-extractor

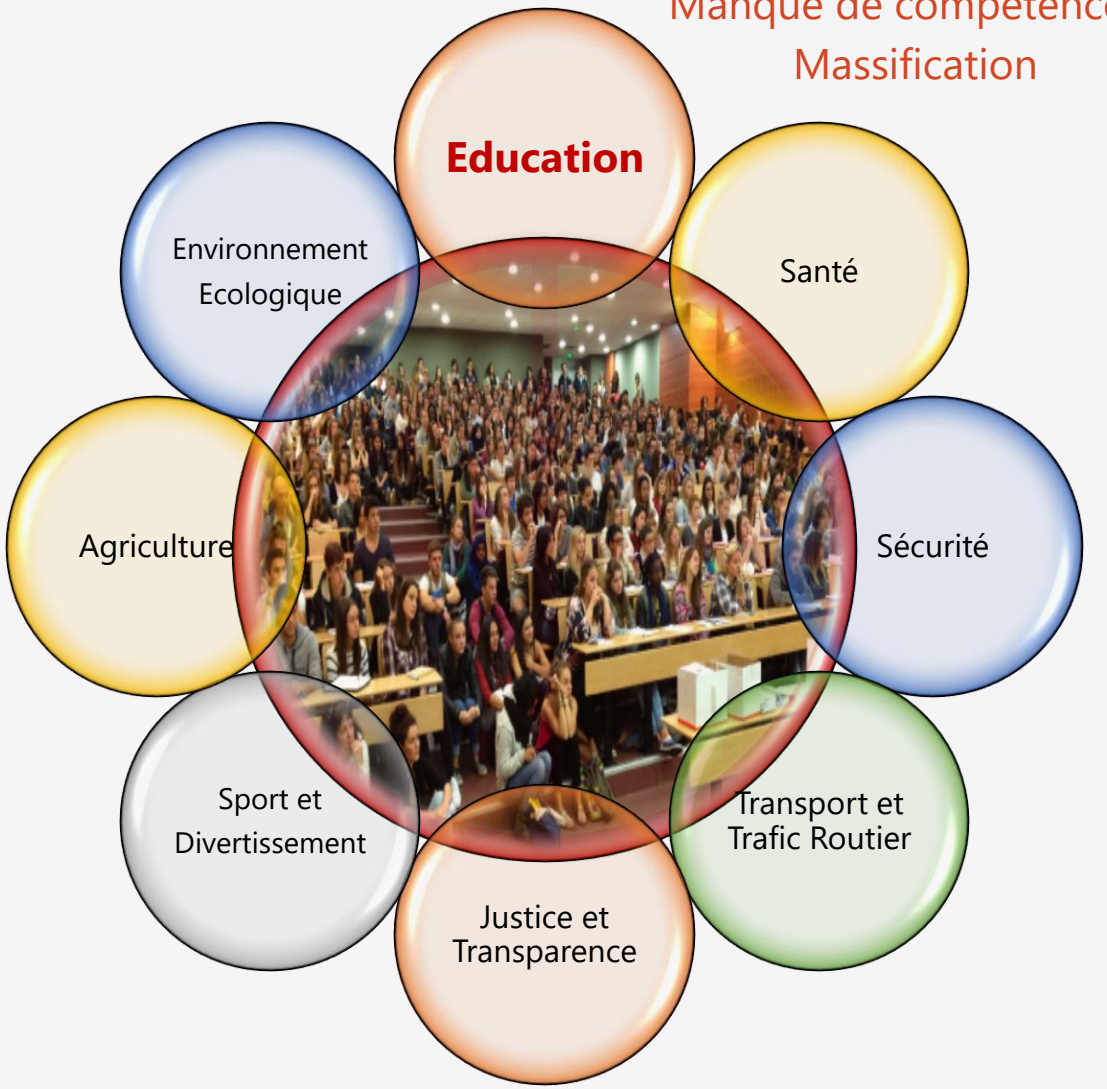
# Application 2 : Deep learning en utilisant ML5.JS et des modèles pré-entrainés

- Cette application montre comment créer d'extraction des caractéristiques « FeatureExtractor » à partir à partir des images capturée par une la caméra du PC.
- Cette application est basée sur TensorFlow.JS et **ML5.JS**.
- Elle charge un modèle pré-entrainé « **MobileNet** » utilisant les techniques de deep learning avec Convolution Neural Network (**CNN**)
- Une fois le modèle MobileNet chargé avec ML5.JS,
  - On présente des objet à la caméra
  - On on ajouter au modèle des images capturées de la caméra en leurs associant un label.
  - On lance l'entrainement du modèle.
  - Une fois entrainé,
    - On présente des objet à la caméra
    - L'application détecte de quel objet s'ajit-il.



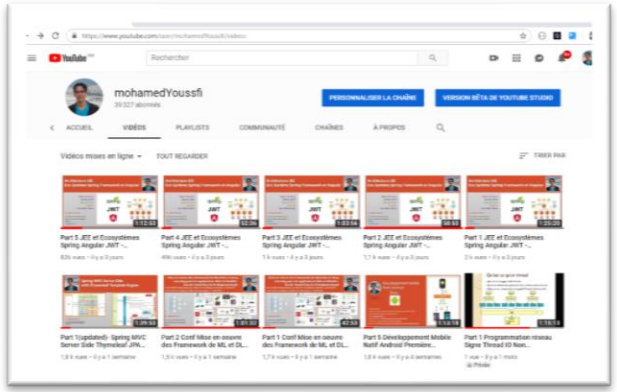
# Education Citoyenne : vers une intelligence collective digitale

Manque de compétences
Massification

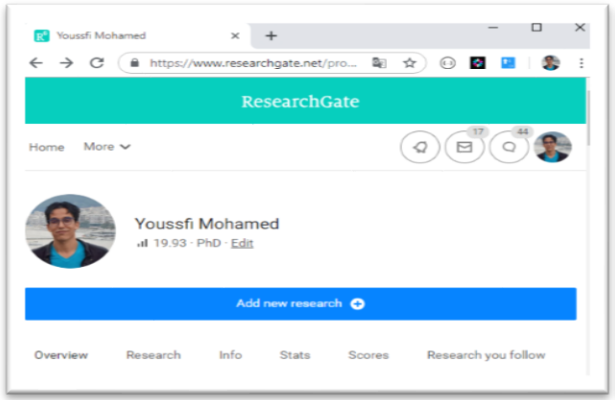


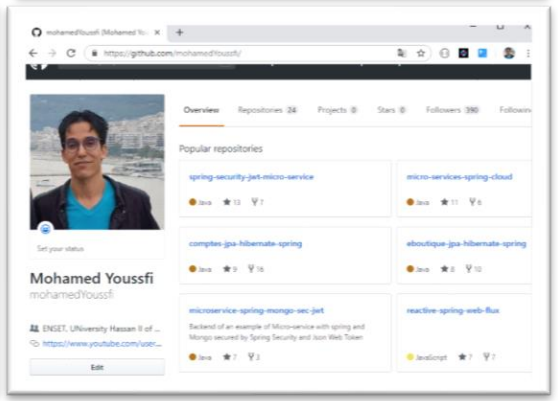

## Mes Références pour le partage digitale :

- **Chaine vidéo :** https://www.youtube.com/user/mohamedYoussfi
  - 40 000 Abonnés, 340 vidéos (Durée Moyenne : 1H30 mn)
- **Supports :** https://www.slideshare.net/mohamedYoussfi9
- **Recherche :** https://www.researchgate.net/profile/Youssfi_Mohamed
- **Code source** : https://github.com/mohamedYoussfi/





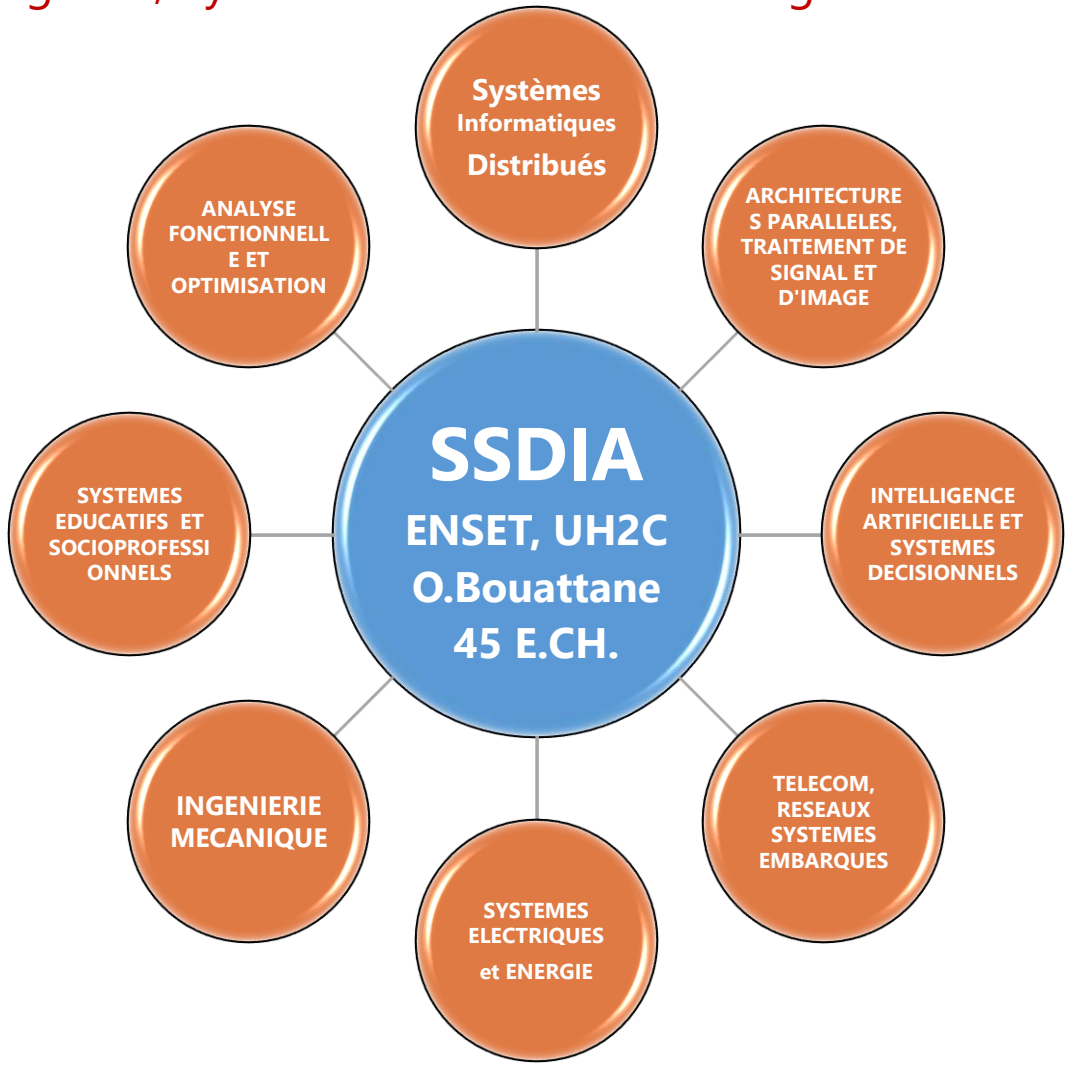


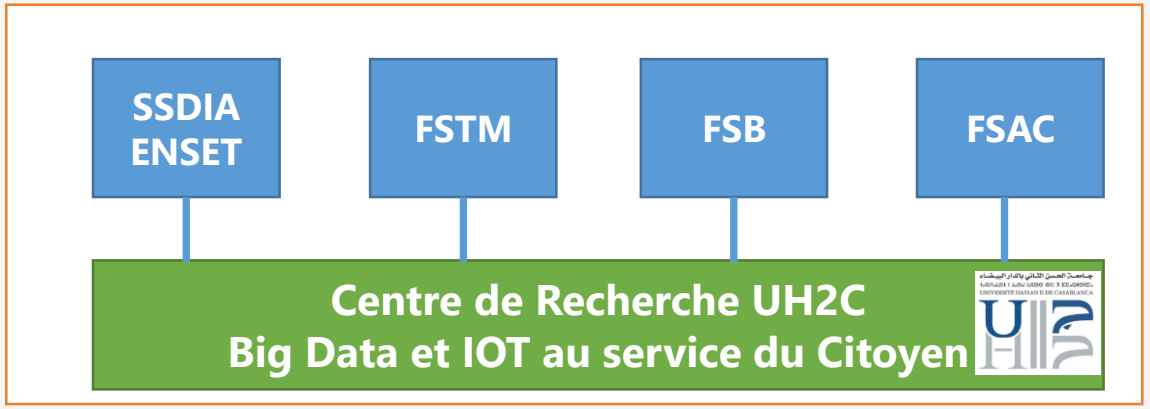



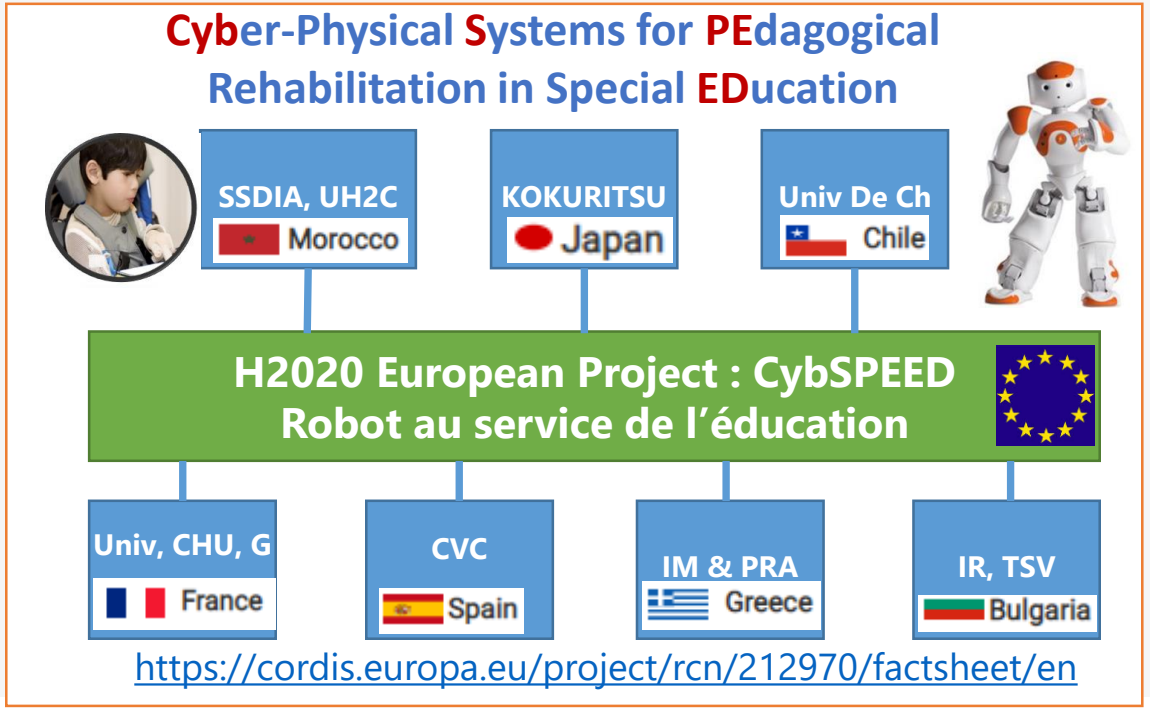

Digitalisation des contenus pédagogiques et Partage de la connaissance

# Recherche : vers la transdisciplinarité collective innovante

## Signaux, Systèmes Distribués et Intelligence Artificielle



# Intelligence Artificielle pour le citoyen, au service des citoyens et par le citoyen