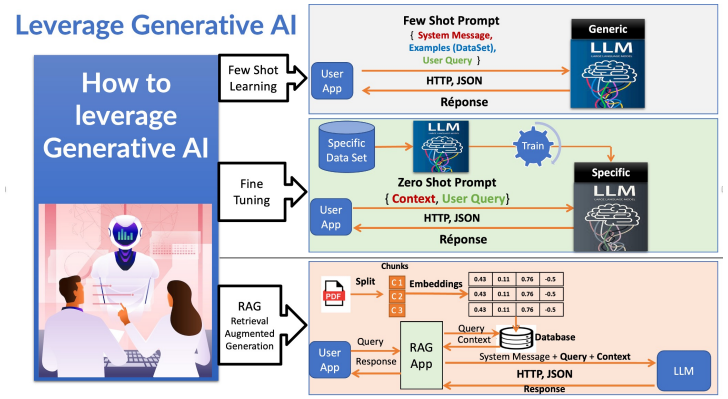
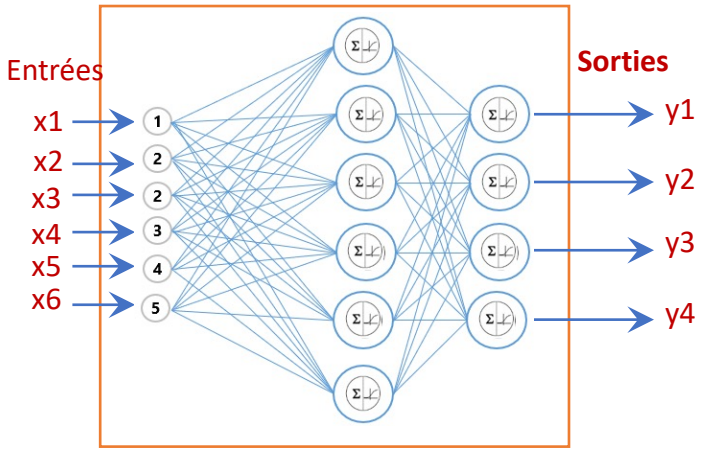
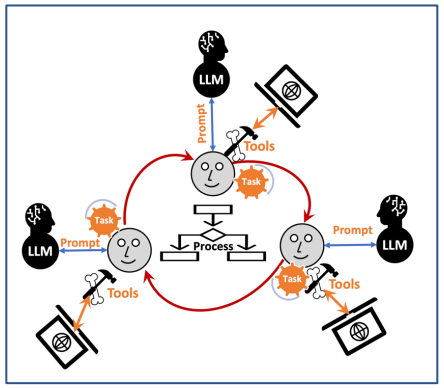
Intelligence Artificielle & IA Générative

Mohamed Youssfi, Enseignant Chercheur, ENSET Mohammedia, Directeur Laboratoire Informatique, Intelligence Artificielle et Cyber Sécurité

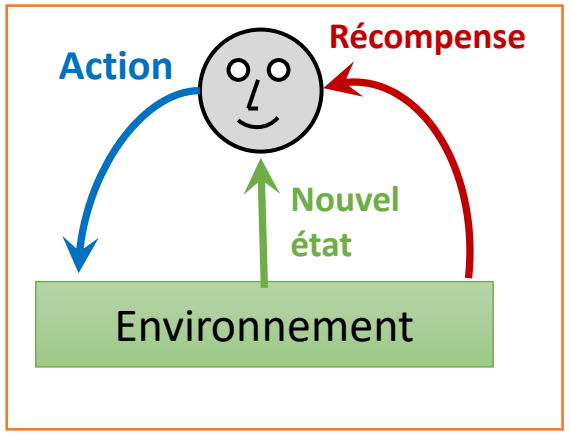


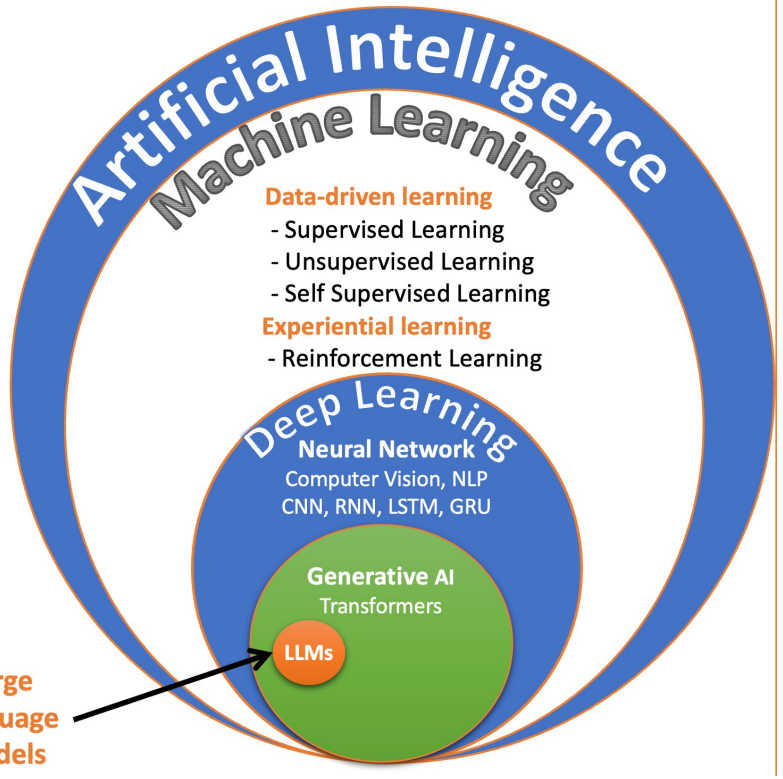
Apprentissage Supervisé





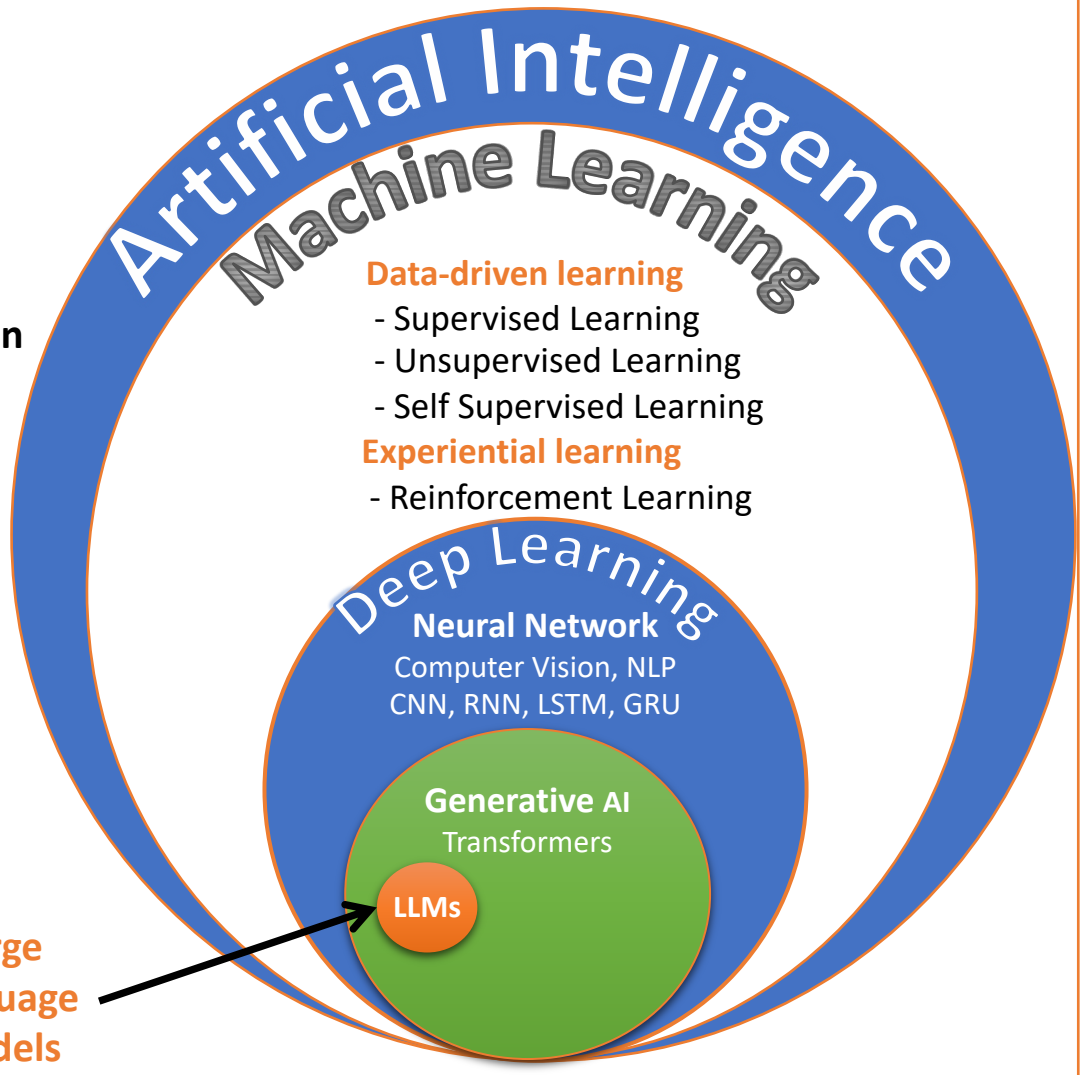
Apprentissage Par Renforcement





Intelligence Artificielle

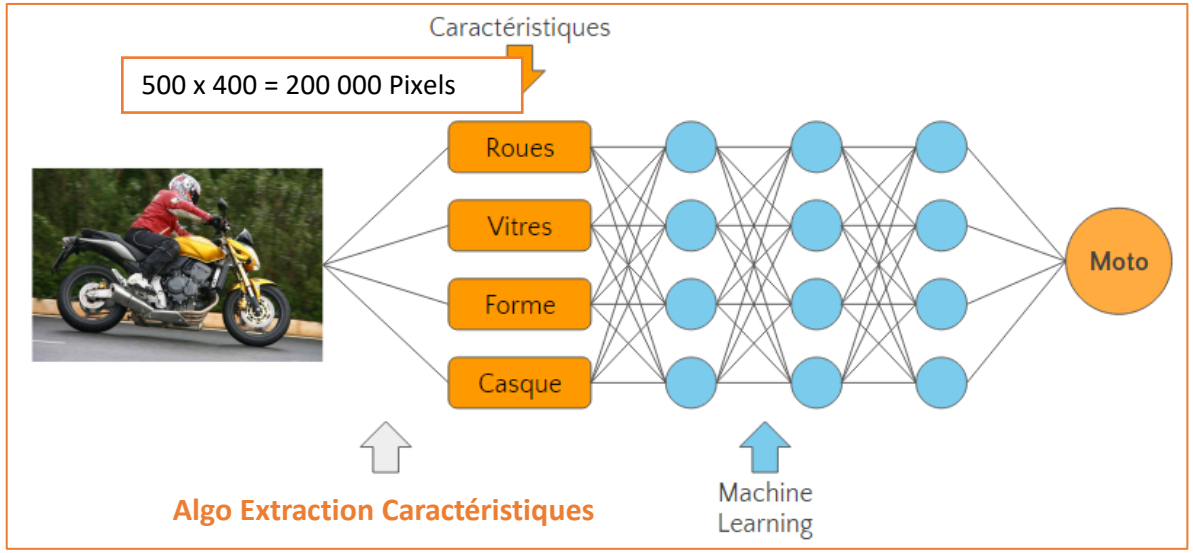
- L'**intelligence artificielle**
- **Intelligence Artificielle Distribuée** =
  - **IA** : Pour des agents intelligent (Modéliser le savoir et le comportement)
  - + **Distribuée** : Modéliser leurs interactions => **Intelligence Collective**
- **IA = IA symbolique (5%) + Machine Learning (95%)**
- **Techniques d'apprentissage :**
  - **Piloté par les données**
    - **Apprentissage Supervisé**
      - **Régression**
      - **Classification**
    - **Apprentissage Non Supervisé**
    - **Apprentissage Auto Supervisé**
  - **Piloté par l'expérience**
- **Deep Learning**
- **IA Générative**

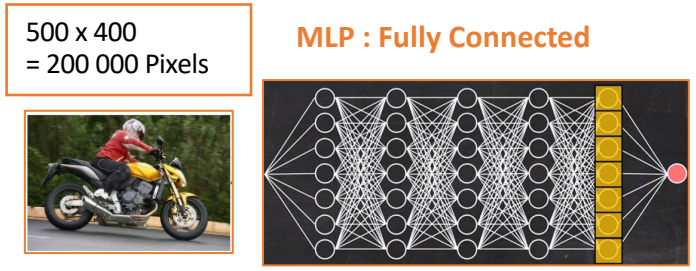


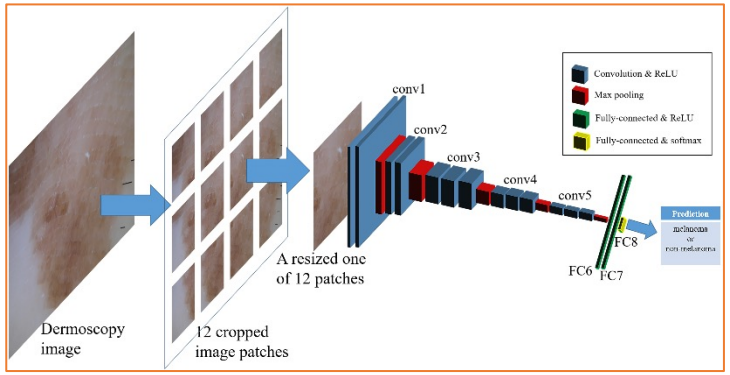
Deep Learning

Deep Learning

- L'apprentissage profond (Deep Learning) est un sous-domaine de l'intelligence artificielle qui utilise des réseaux neuronaux artificiels formant de nombreuses couches pour résoudre des tâches complexes.
- L'apprentissage profond permet des progrès importants et rapides dans les domaines de l'analyse du signal sonore ou visuel, notamment :
  - La reconnaissance faciale,
  - La reconnaissance vocale,
  - Computer Vision
  - Natural Language Processing (NLP).

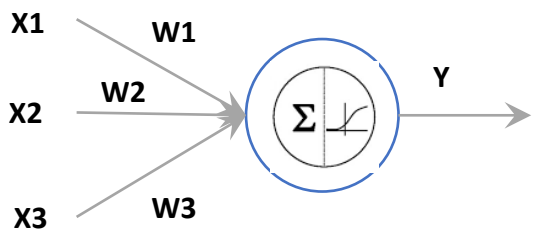




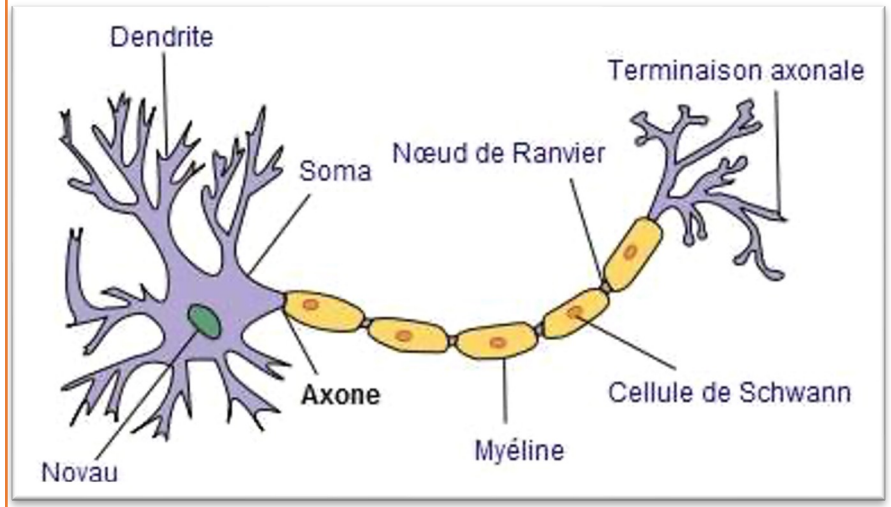


Deep Learning

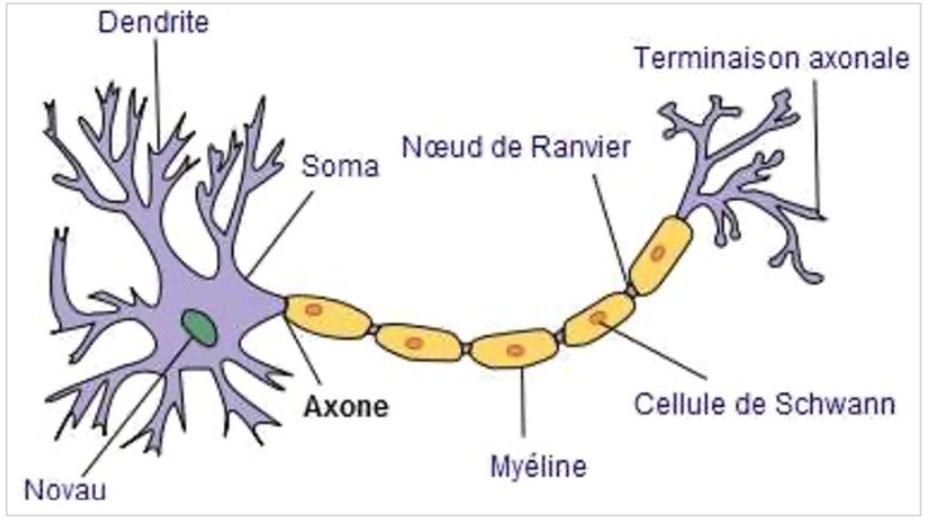
- Un neurone artificiel (NA) est une unité de calcul élémentaire (fonction mathématique) permet de mettre en relations des entrées Xi avec une sortie Y
- Un neurone artificiel est représentation approximative d'un neurone biologique
  - Additionne ses entrées x_i pondérées par des poids w_i ,
  - compare 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).
- 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.

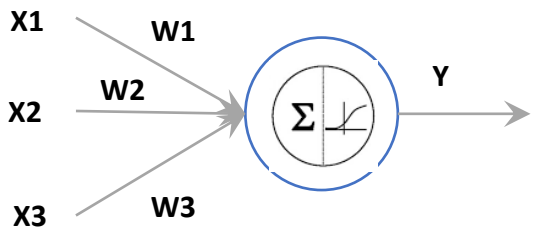


- $Sum=x_1w_1+x_2w_2+x_3w_3$
- Si Sum>Seuil => Y=1 Si non Y=0

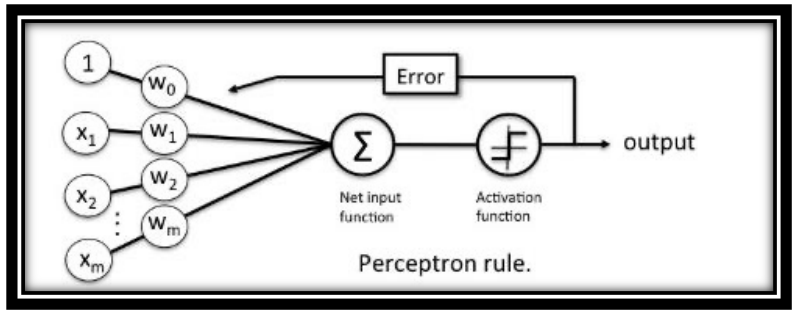


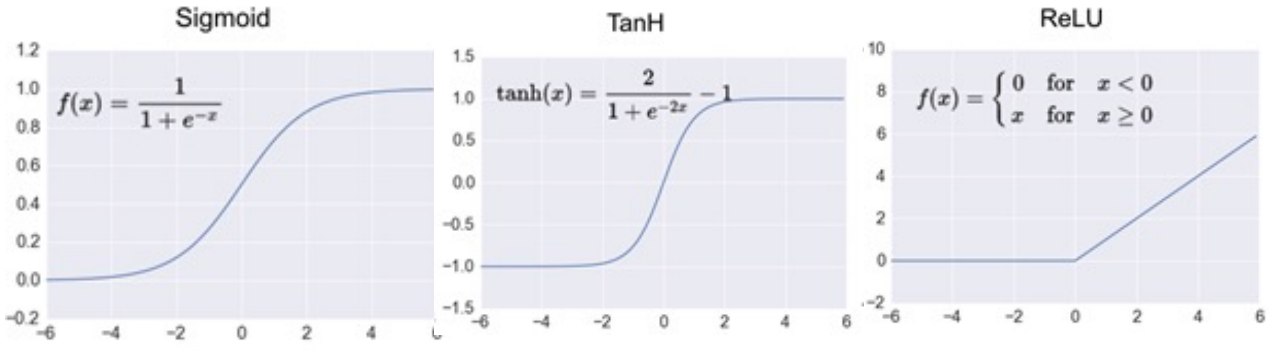
Deep Learning





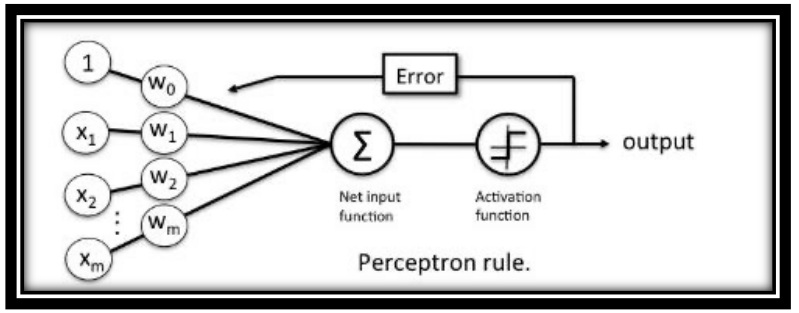
- $Sum=x_1w_1+x_2w_2+x_3w_3$
- Si Sum>Seuil => Y=1 Si non Y=0



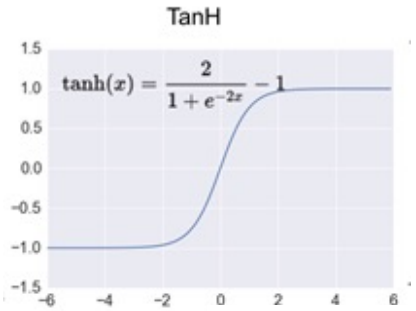


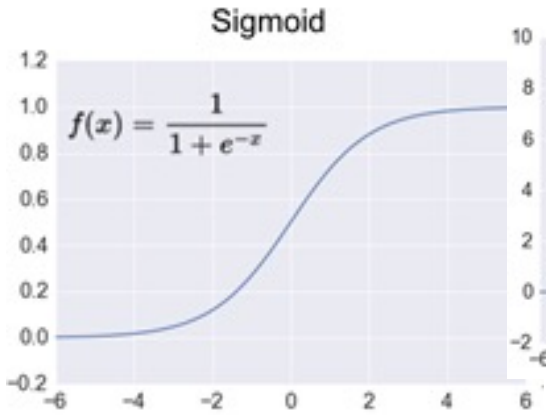
Apprentissage supervisé : Classification (Neural Network)

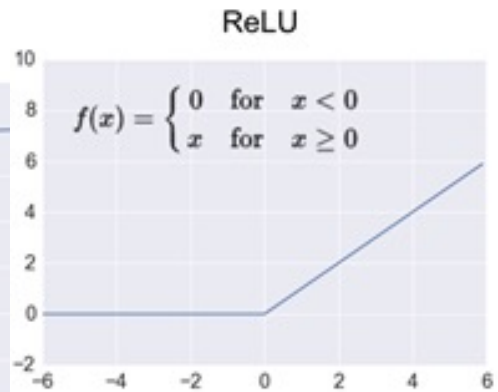
Perceptron [Frank Rosenblatt ,1957]



- 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 jour des poids : **$w_i(t)=w_i(t-1)+\alpha$ err.Xi**
- **α** : vitesse d'apprentissage (entre 0 et 1)

Apprentissage supervisé : Classification (Neural Network)

Multi Layer Perceptron (MLP)

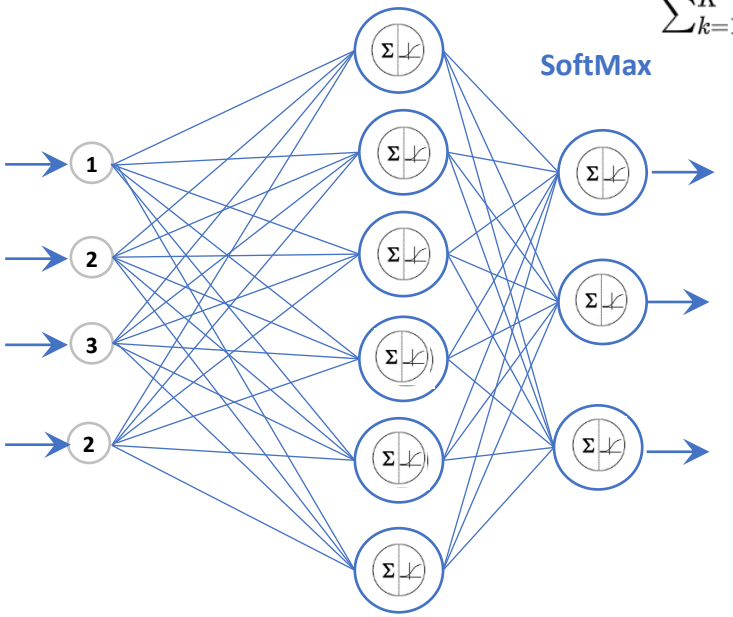
- Un Multi Layer Perceptron (MLP) est un type de réseau de neurones artificiels composé de plusieurs couches de neurones. Voici ses principales caractéristiques :
- **Couches** : Il comprend une couche d'entrée, une ou plusieurs couches cachées, et une couche de sortie.
- **Neurones** : Les neurones dans chaque couche sont connectés à tous les neurones de la couche précédente (connexion "fully connected").
- **Fonction d'activation** : Chaque neurone applique une fonction d'activation (comme ReLU, sigmoïde, ou tanh) pour introduire de la non-linéarité.
- **Entraînement par rétropropagation** : L'apprentissage du MLP se fait par rétropropagation de l'erreur à l'aide d'un algorithme d'optimisation, comme l'Adam ou le SGD.
- Un MLP est souvent utilisé pour des tâches de classification ou de régression et est une forme de réseau de neurones feed forward.

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

Sigmoid

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

SoftMax



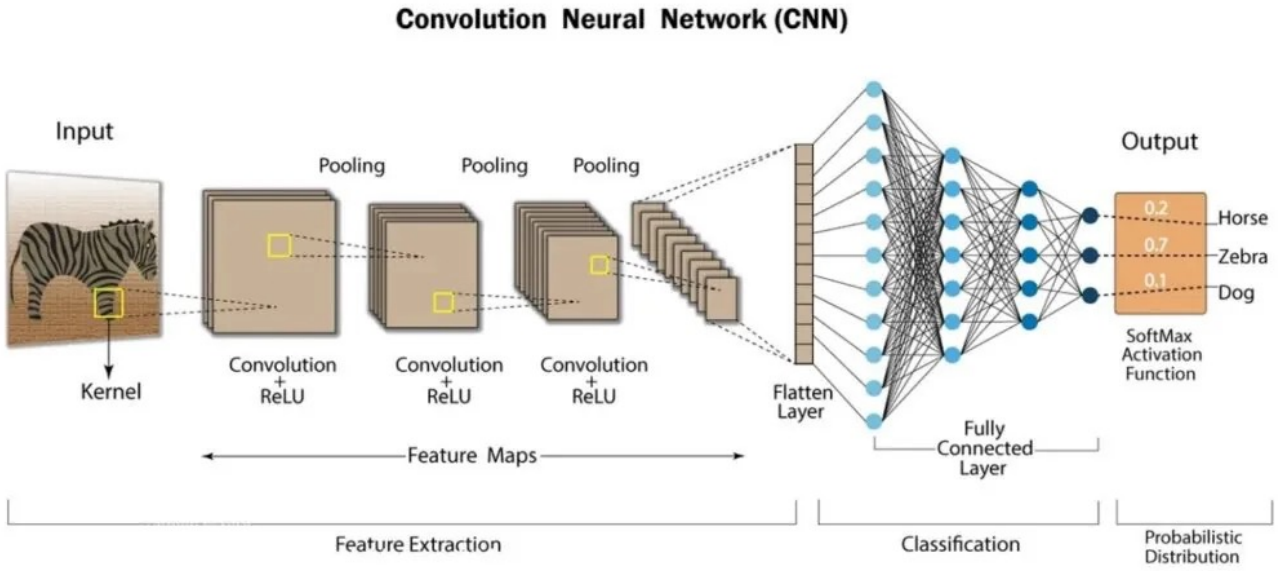
Learning Rate : 0.001 Optimiser : ADAM, SGD

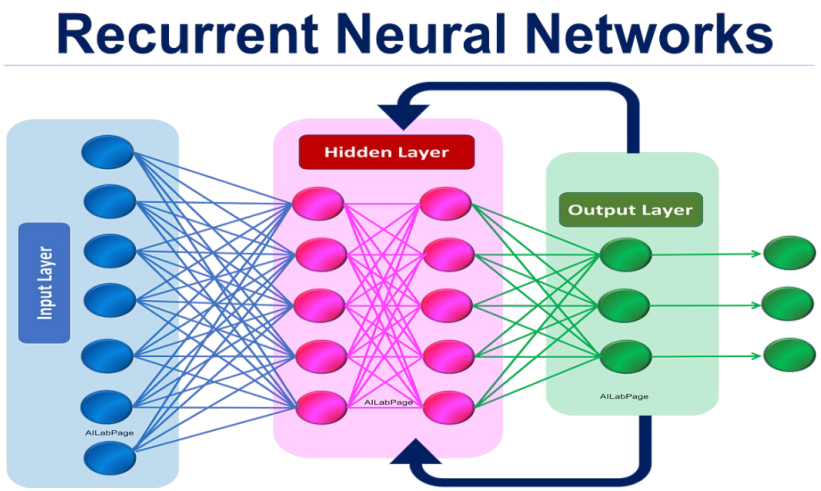
Loss function : MearnSquaredError

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

Modèles de Deep learning

- ANN (Artificial Neural Network) ou MLP (Multi Layer Perceptron); Réseaux de neurones entièrement connectés
- CNN (Convolutional Neural network); Réseaux de Convolutionnels : Dédiés pour les problèmes de computer vision comme la reconnaissance faciale, la classification des images, la détection des objets dans des images, etc.
- RNN (Recurrent Neural Network), LSTM (Long Short Time Memory), GRU (Gated Recurrent Unit) : Dédiés pour la prédiction dans dans des séquences temporelles comme les séries temporelles et pour NLP (Natural Language processing)



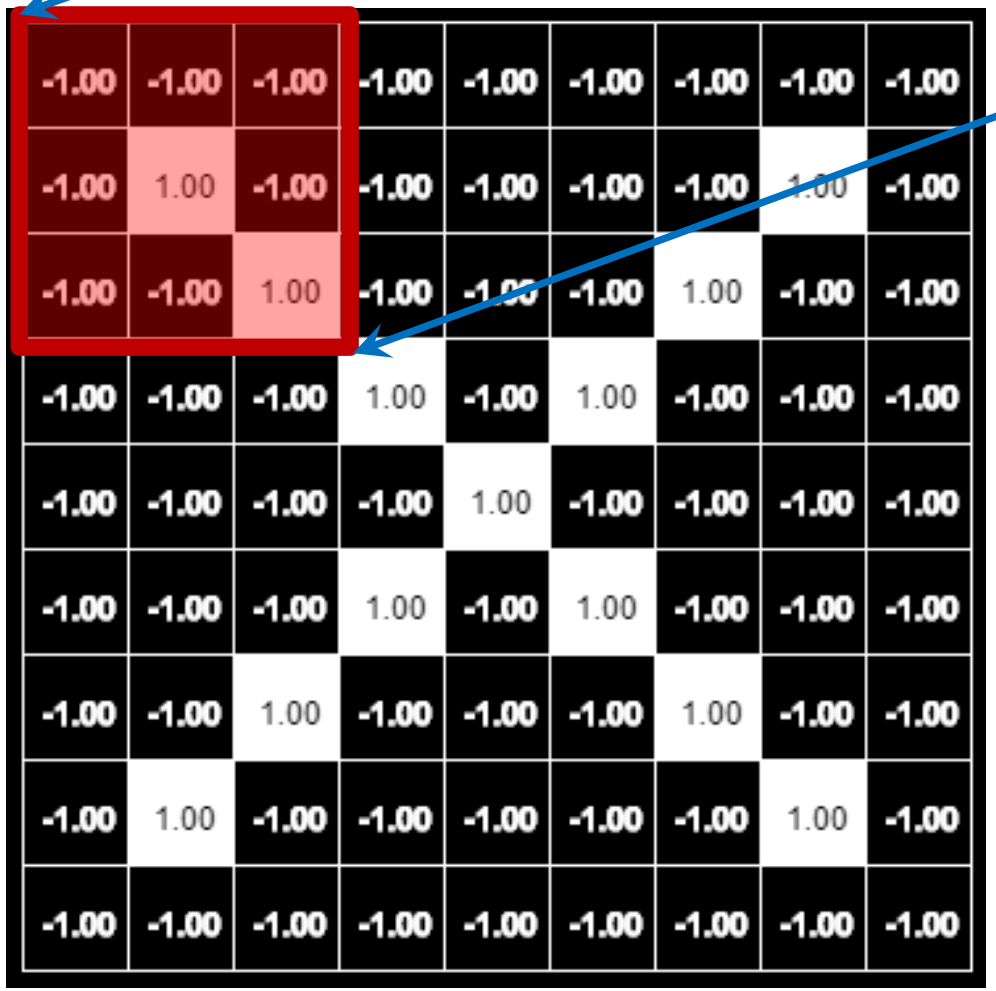


Covolution Layer

Kernel

1.00	-1.00	-1.00
-1.00	1.00	-1.00
-1.00	-1.00	1.00

Image Originale



Kernel size : 3 x 3

Stride : 1

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

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

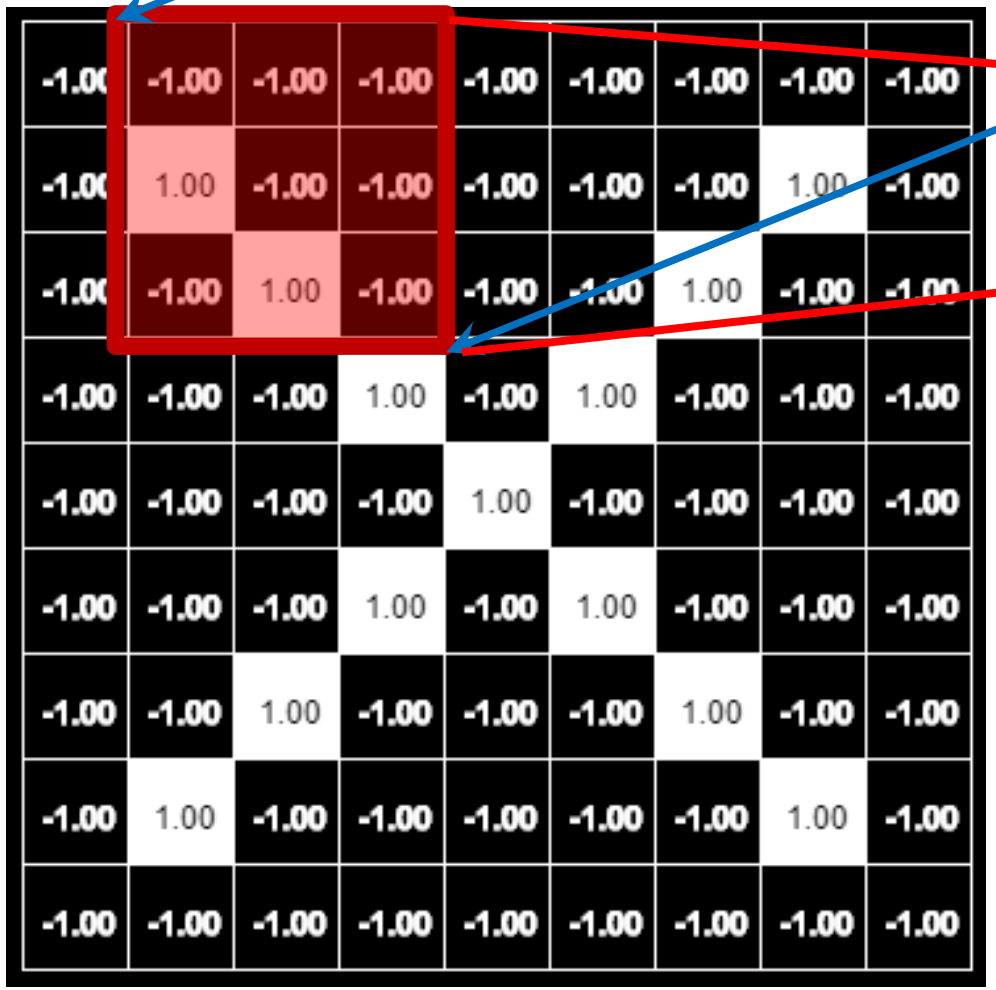
Stride : 1

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



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

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

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

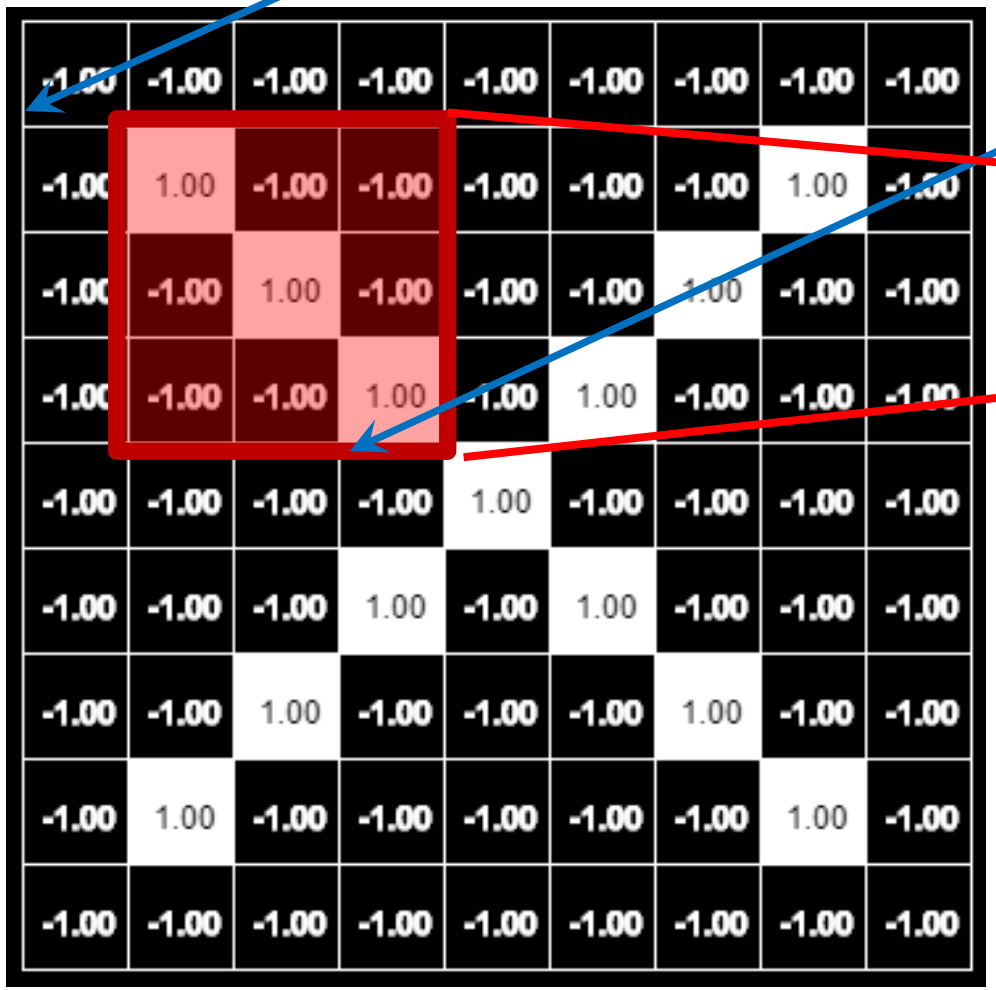
Stride : 1

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

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

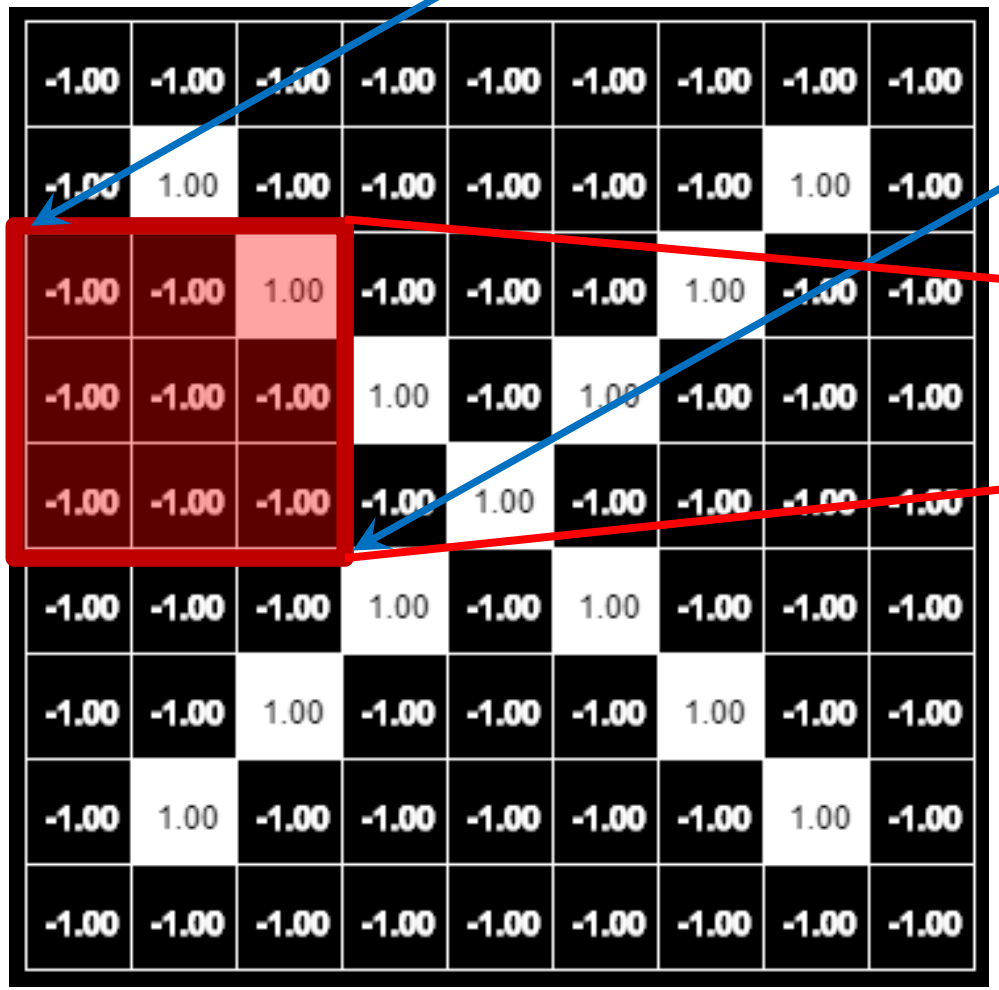
Stride : 1

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



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

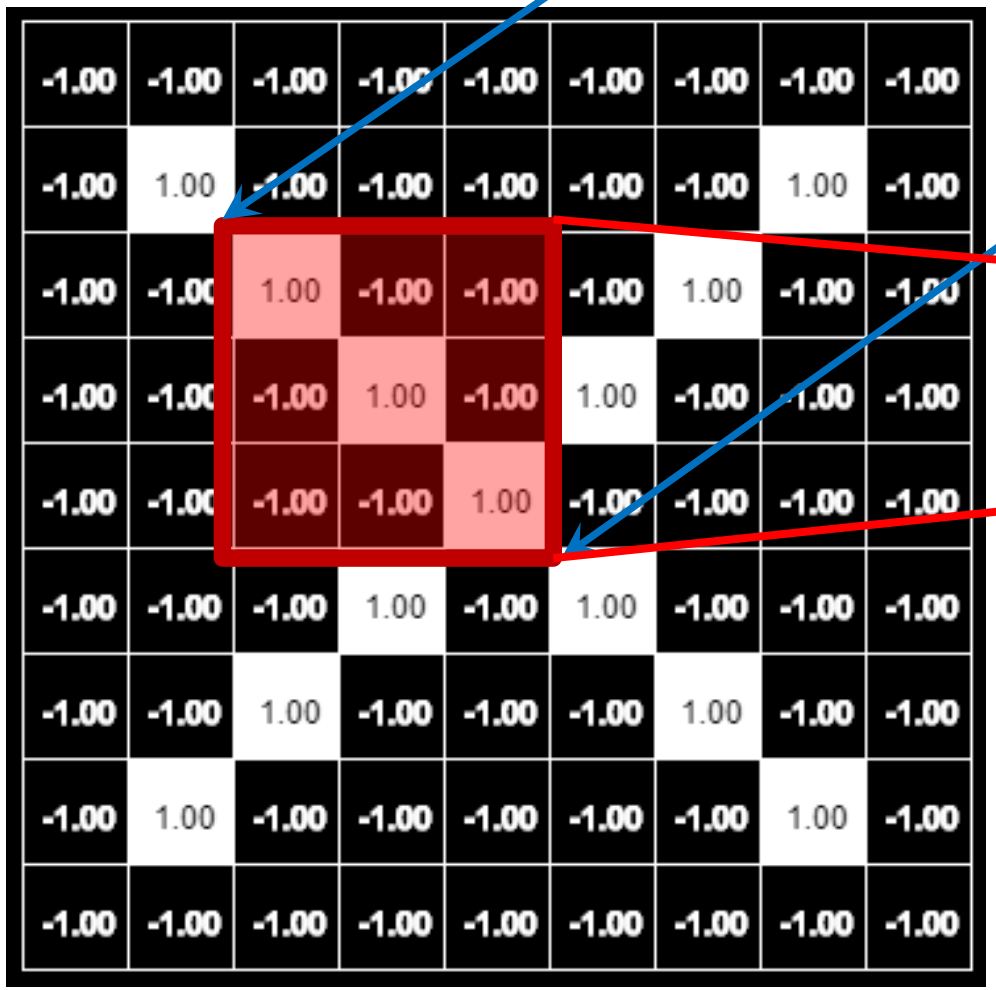
Stride : 1

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 => -0.33

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

-1.00	-1.00	-1.00	-1.00	-1.00	-1.00	-1.00	-1.00	-1.00
-1.00	1.00	-1.00	-1.00	-1.00	-1.00	-1.00	1.00	-1.00
-1.00	-1.00	1.00	-1.00	-1.00	-1.00	1.00	-1.00	-1.00
-1.00	-1.00	-1.00	1.00	-1.00	1.00	-1.00	-1.00	-1.00
-1.00	-1.00	-1.00	-1.00	1.00	-1.00	-1.00	-1.00	-1.00
-1.00	-1.00	-1.00	1.00	-1.00	1.00	-1.00	-1.00	-1.00
-1.00	-1.00	1.00	-1.00	-1.00	-1.00	1.00	-1.00	-1.00
-1.00	1.00	-1.00	-1.00	-1.00	-1.00	-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

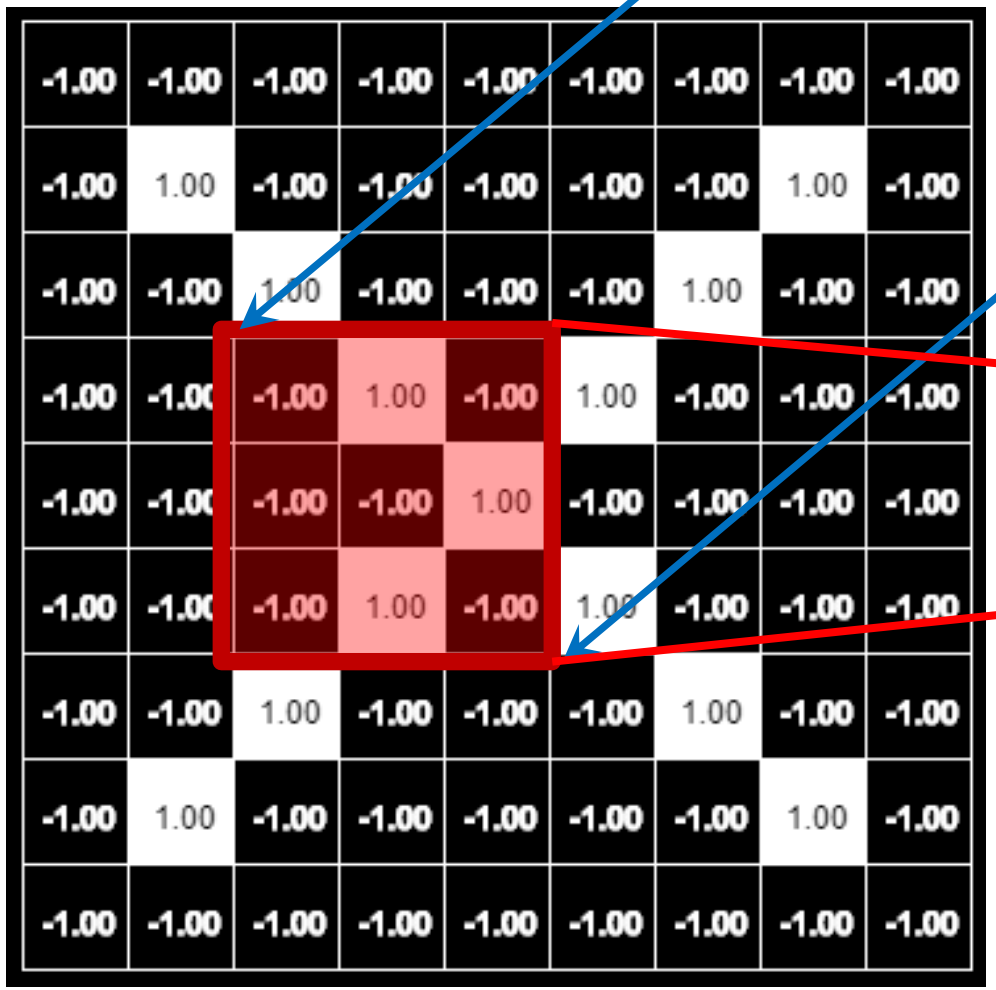
Stride : 1

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



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

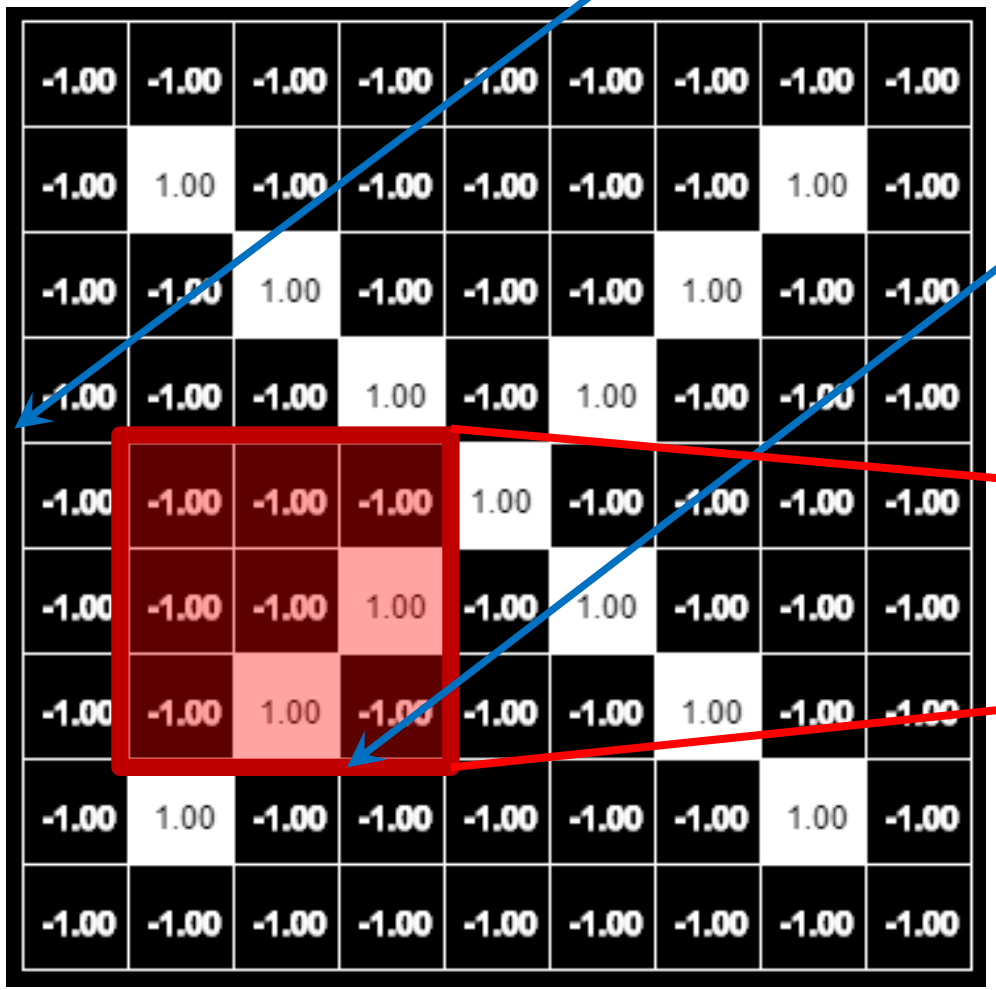
Stride : 1

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



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

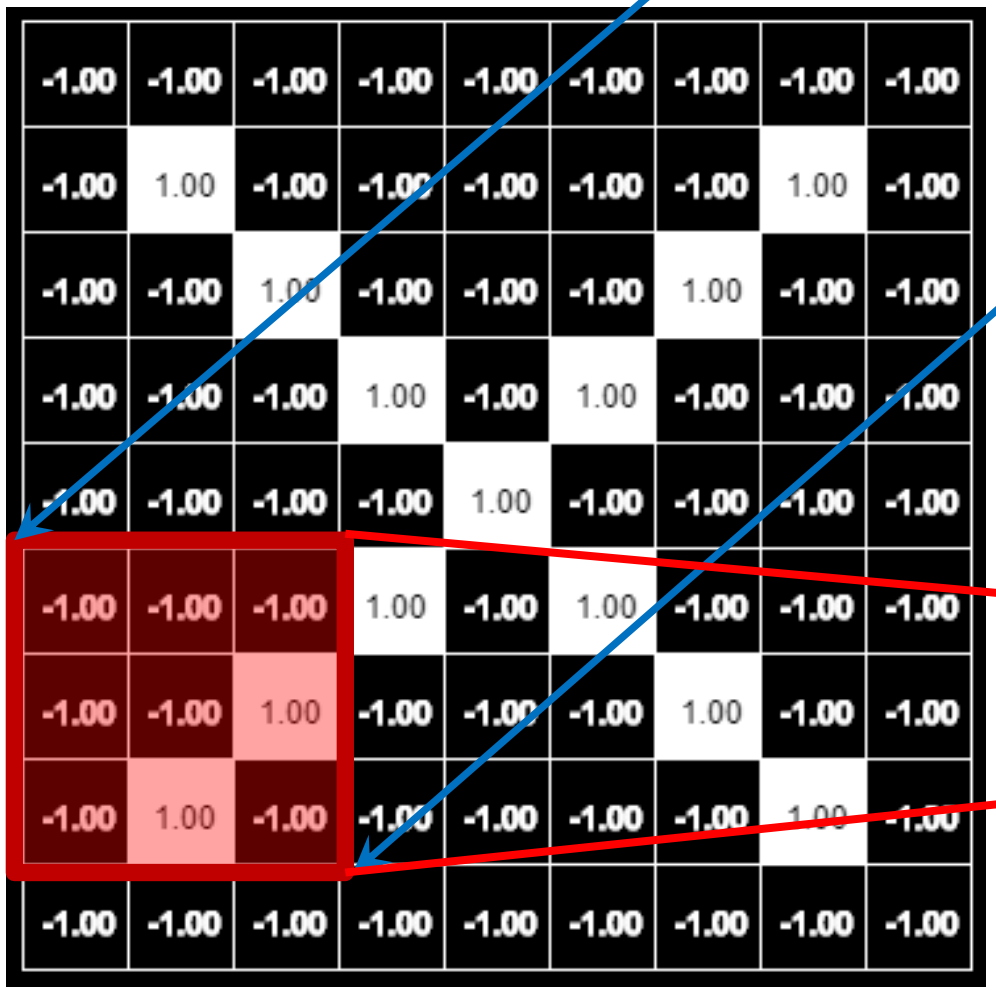
Stride : 1

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



Convolution

Image Filtrée

Kernel size : 3 x 3

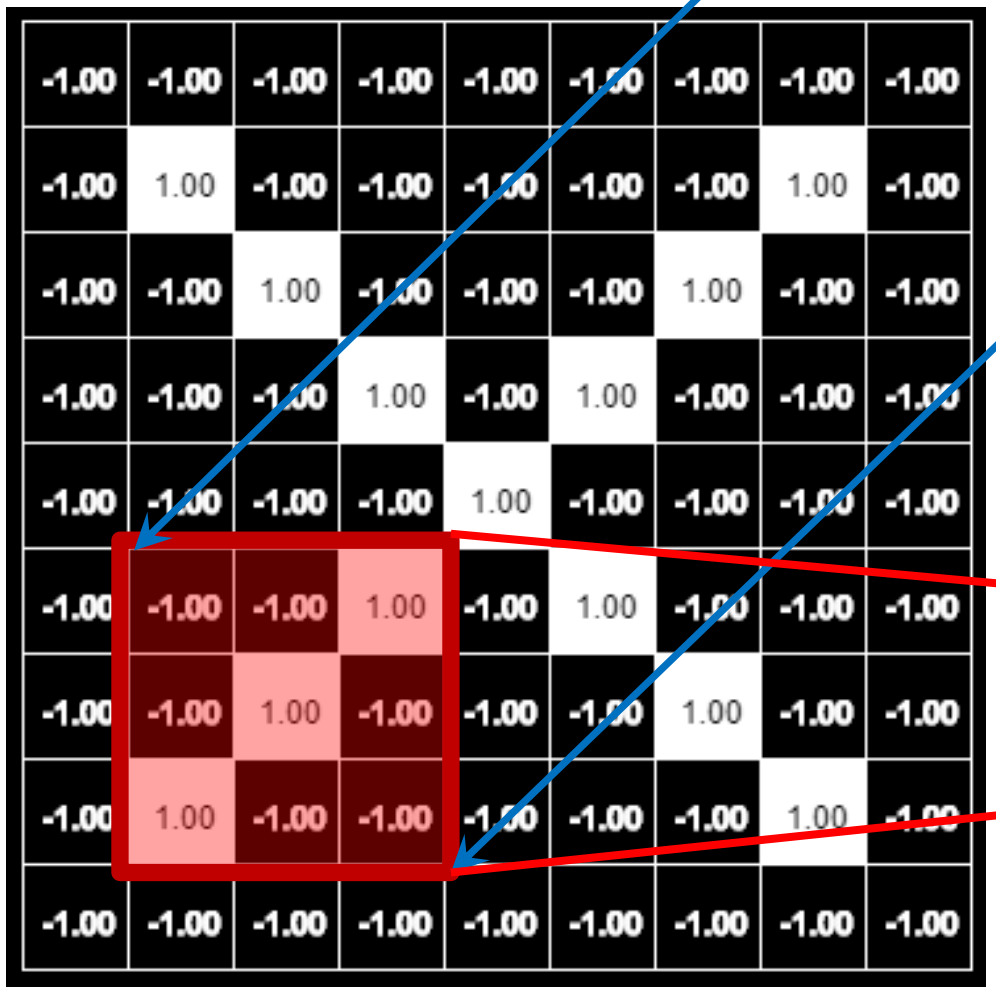
Stride : 1

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 => 0.78

Image Originale



Convolution

Image Filtrée

Kernel size : 3 x 3

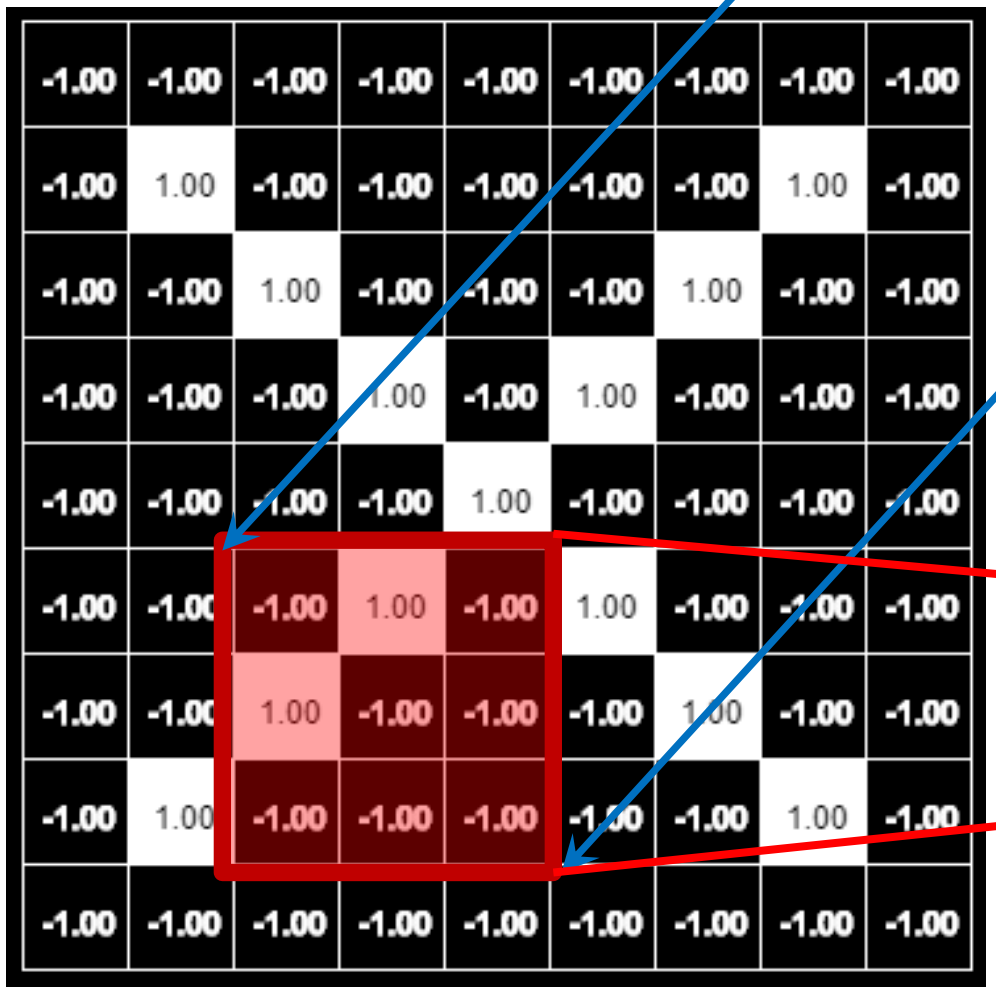
Stride : 1

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



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

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

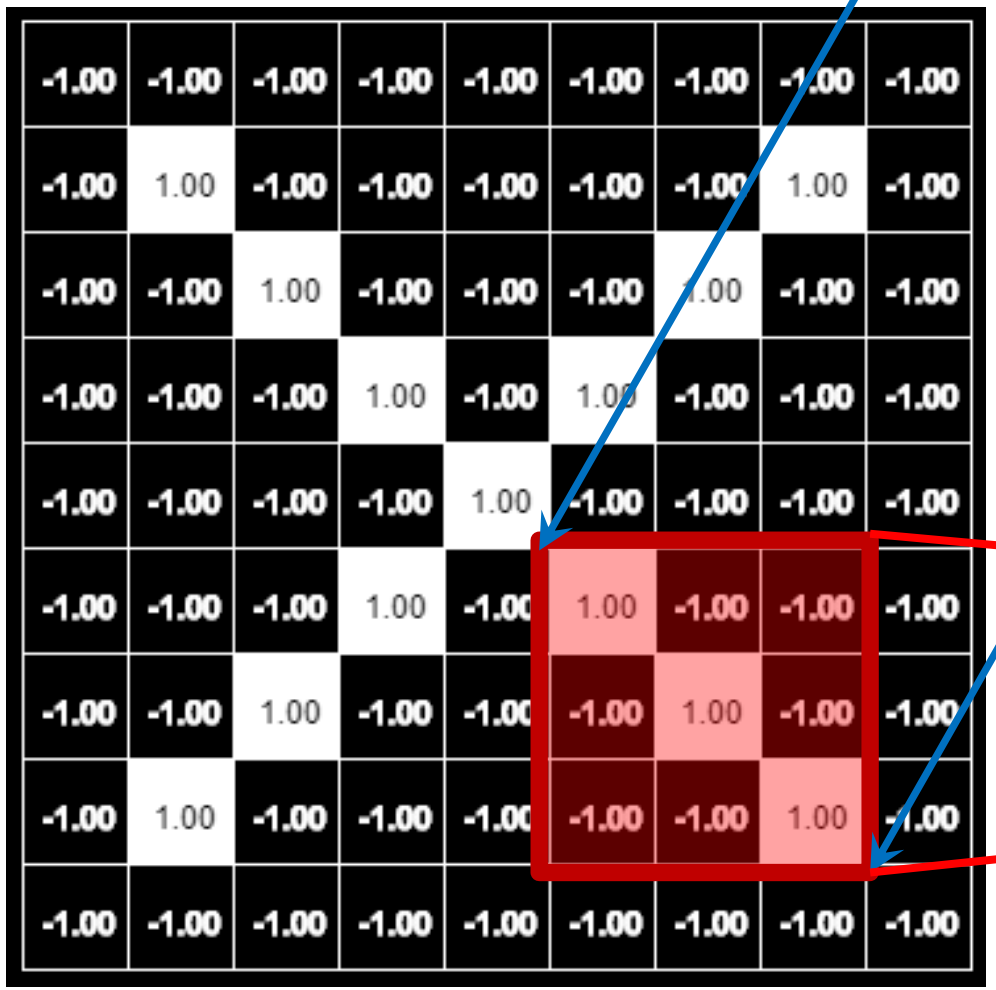
Stride : 1

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

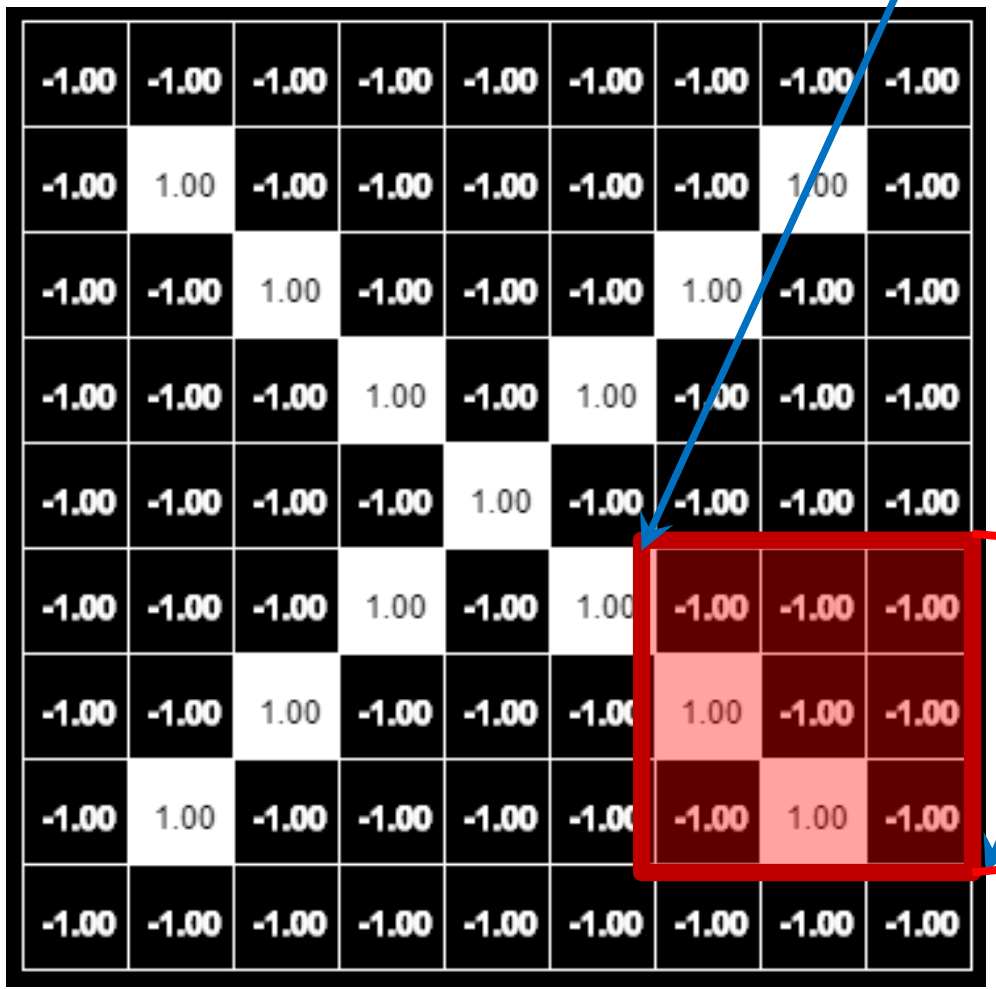
Stride : 1

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



Convolution

Image Filtrée

Kernel size : 3 x 3

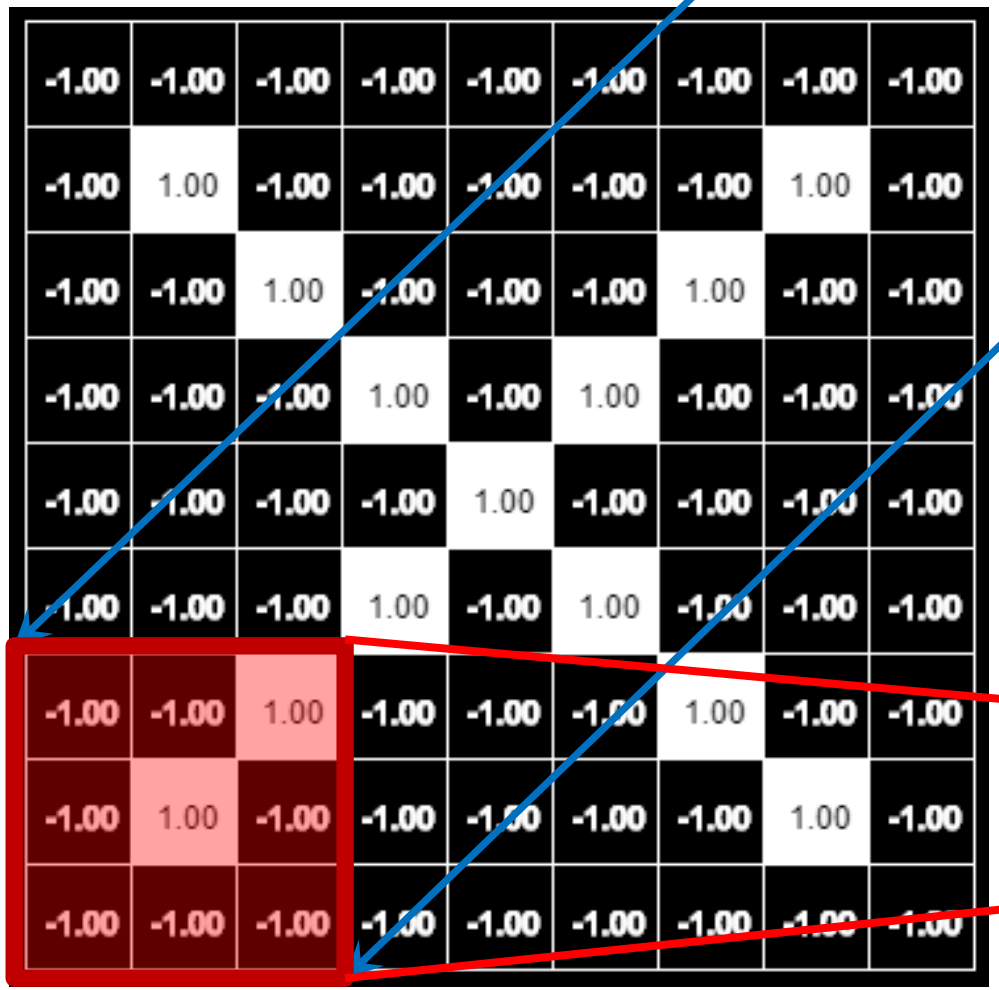
Stride : 1

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 => 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

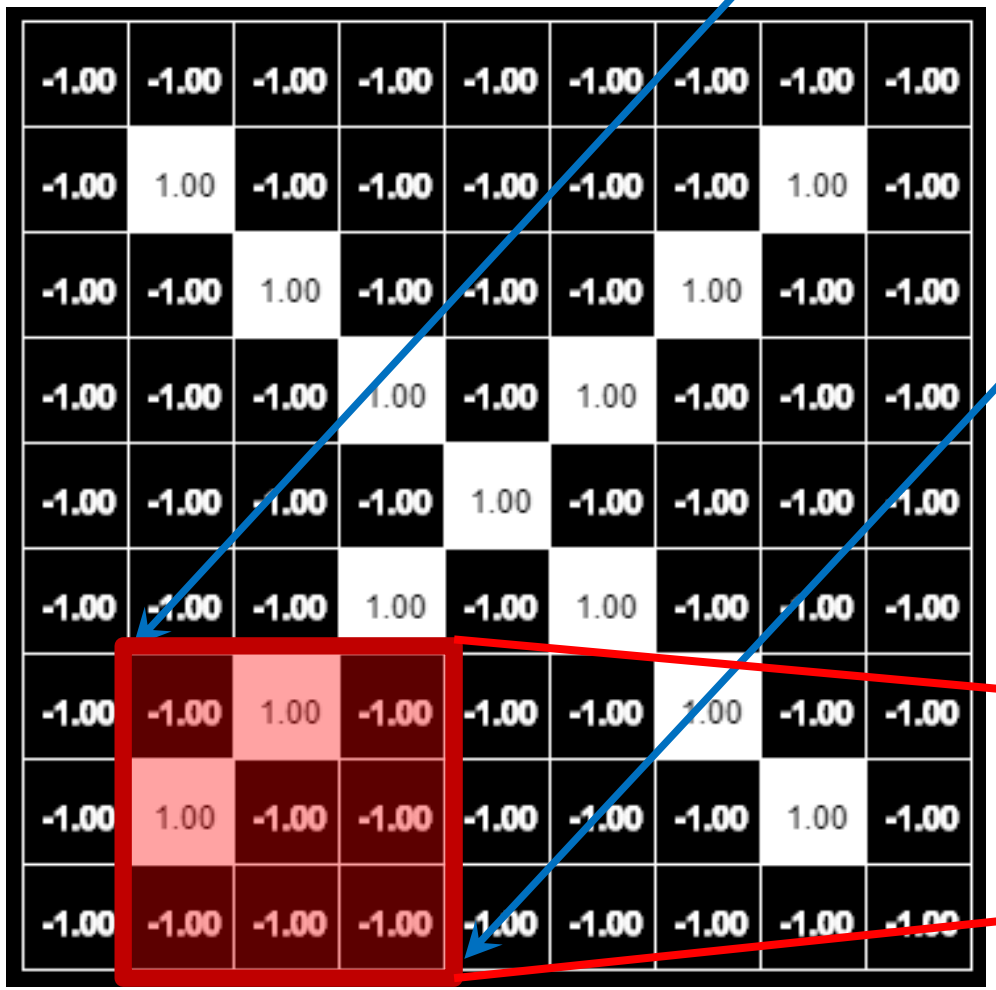
Stride : 1

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



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

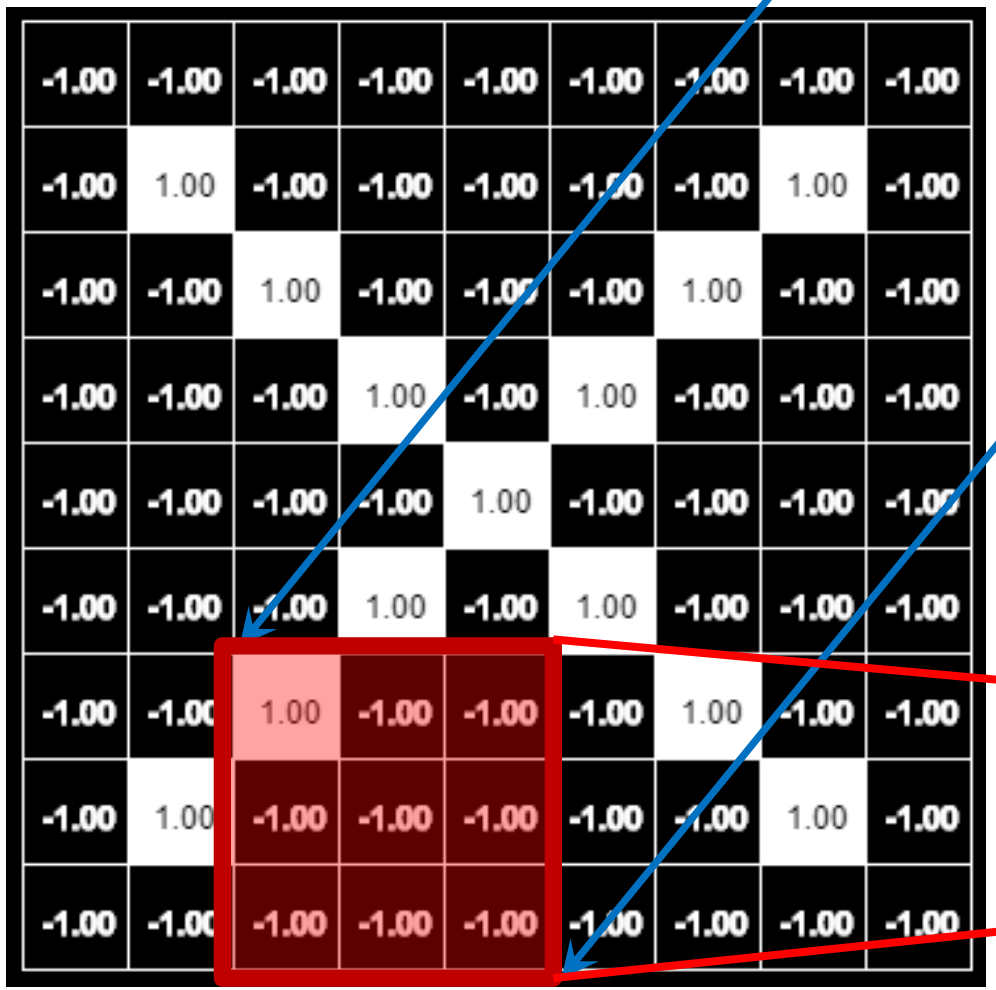
Stride : 1

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



Convolution

Image Filtrée

Kernel size : 3 x 3

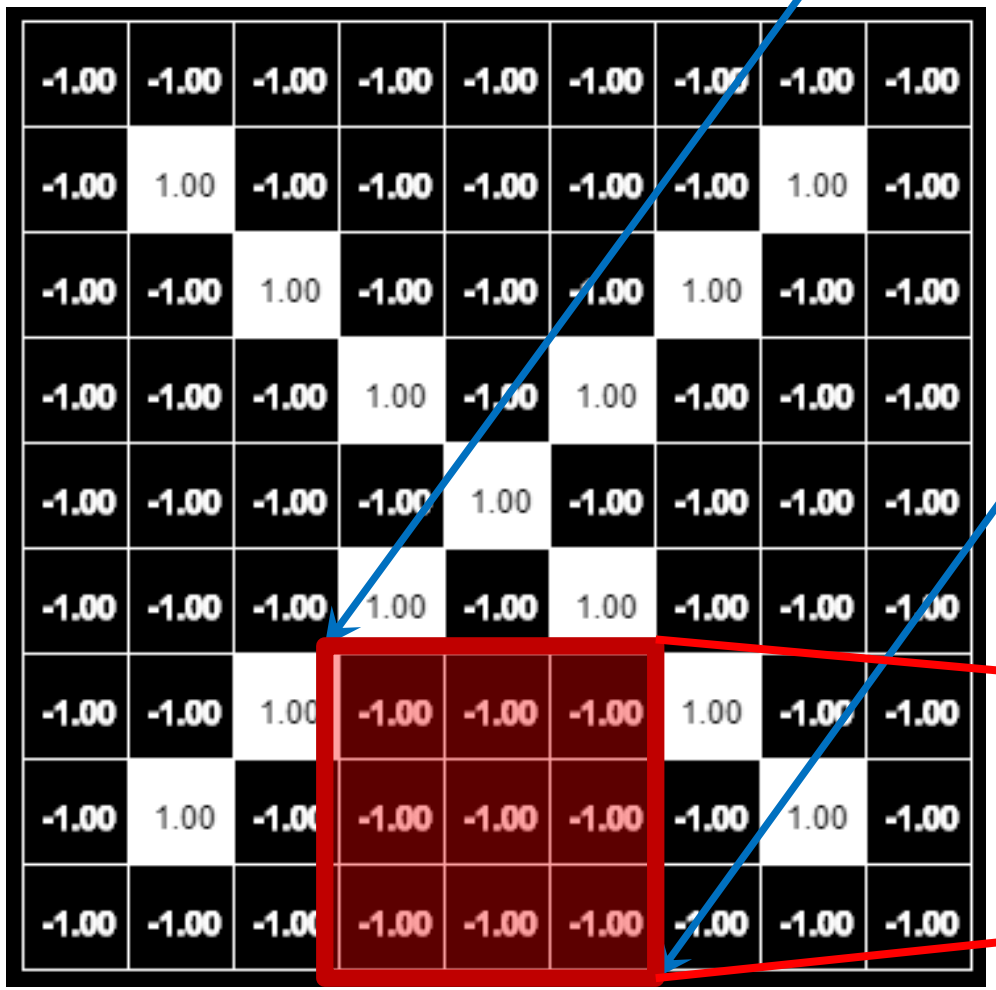
Stride : 1

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



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

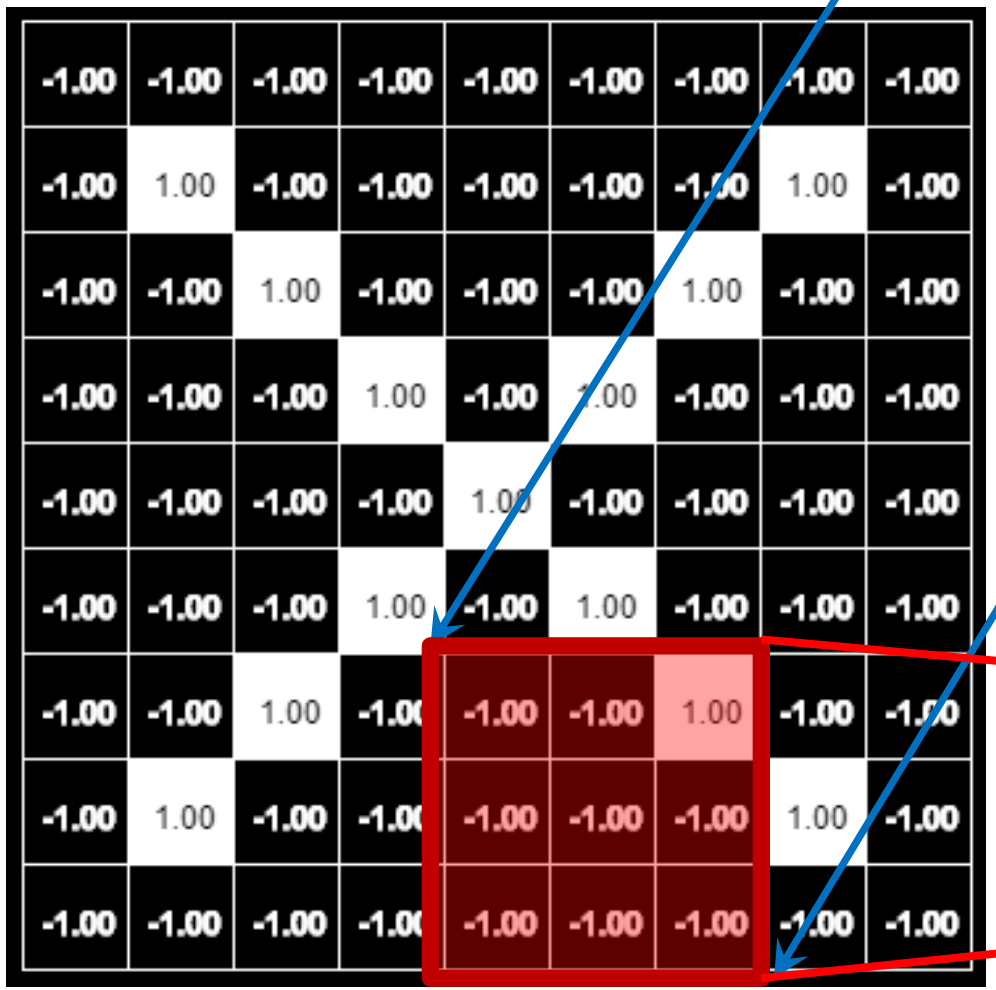
Stride : 1

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



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

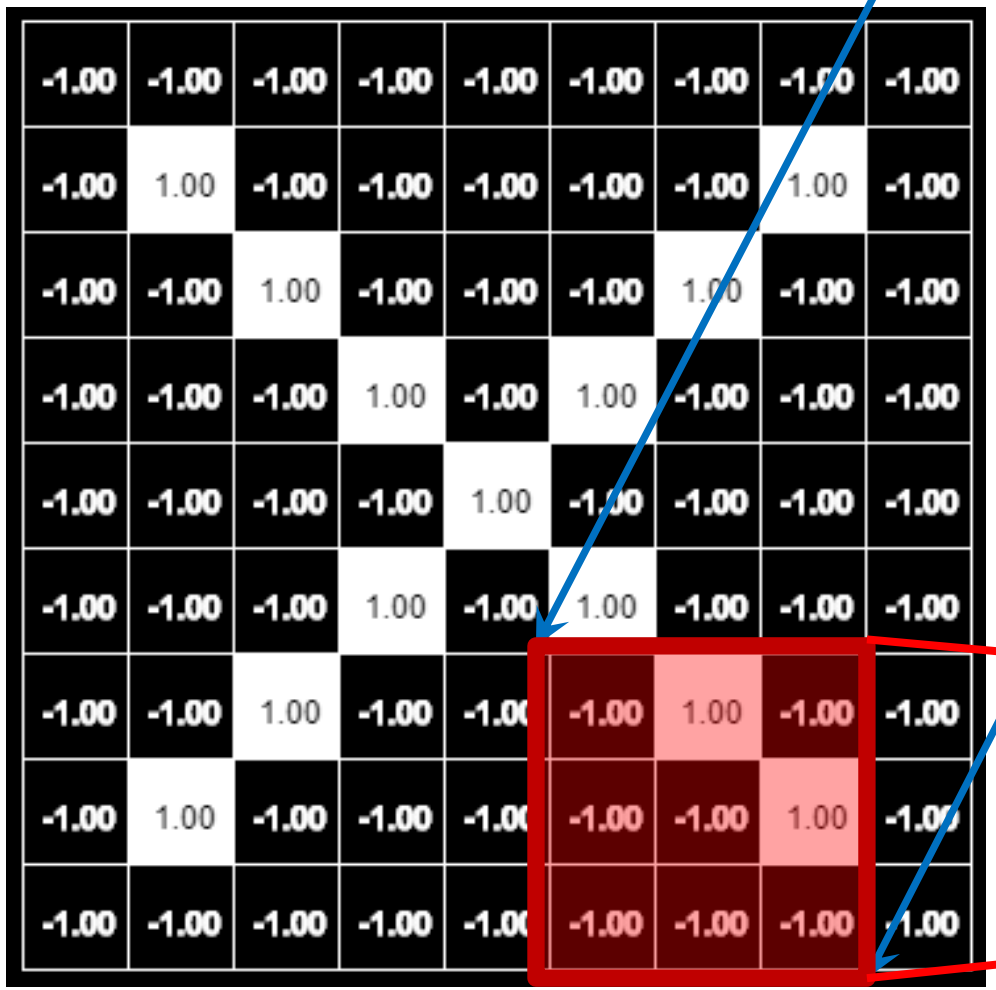
Stride : 1

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



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

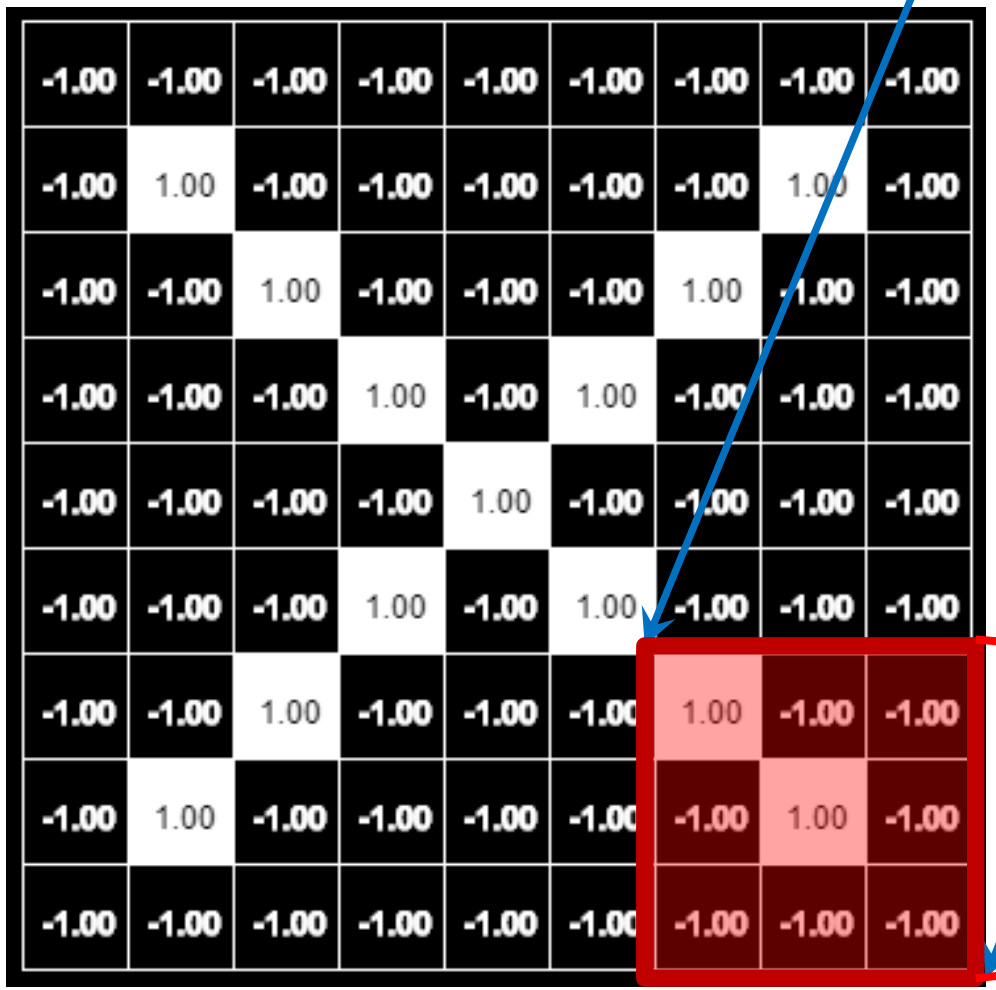
Stride : 1

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



Convolution

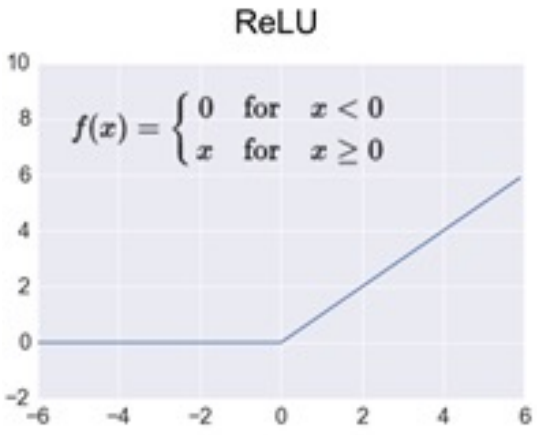
Image Filtrée

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

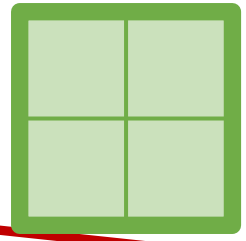
0.78	0.00	0.11	0.33	0.56	0.00	0.33
0.00	1.00	0.00	0.33	0.00	0.11	0.00
0.11	0.00	1.00	0.00	0.11	0.00	0.56
0.33	0.33	0.00	0.56	0.00	0.33	0.33
0.56	0.00	0.11	0.00	1.00	0.00	0.11
0.00	0.11	0.00	0.33	0.00	1.00	0.00
0.33	0.00	0.56	0.33	0.11	0.00	0.78

POOLING Layer (POOL MAX)

Kernel size : 2 x 2

Stride : 2

Image Filtrée avec RELU



POOL MAX

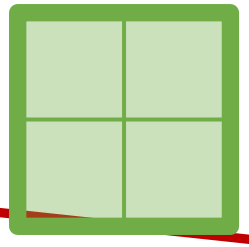
Image Après Pool Max

POOLING (POOL MAX)

Kernel size : 2 x 2

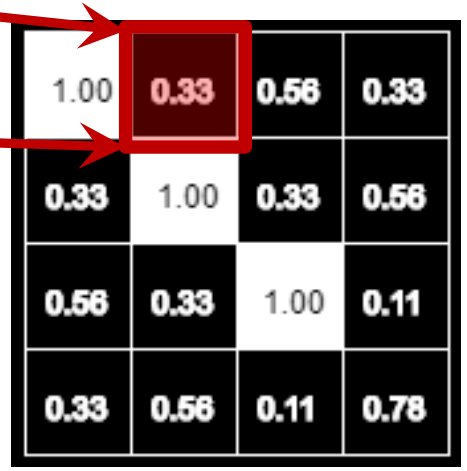
Stride : 2

Image Filtrée avec RELU



POOL MAX

Image Après Pool Max

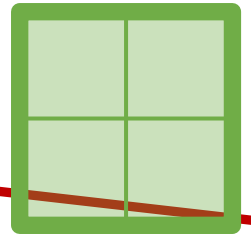


POOLING (POOL MAX)

Kernel size : 2 x 2

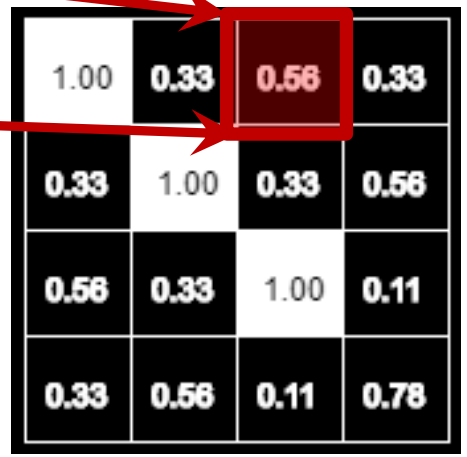
Stride : 2

Image Filtrée avec RELU



POOL MAX

Image Après Pool Max



POOLING (POOL MAX)

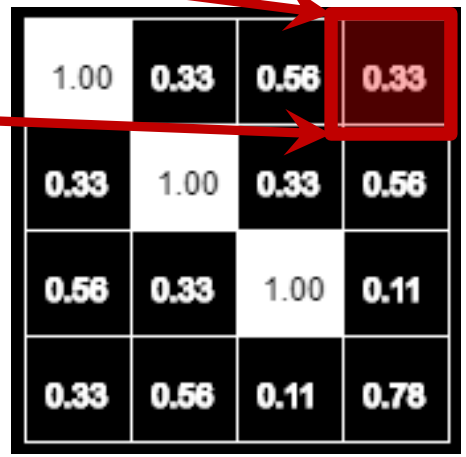
Kernel size : 2 x 2

Stride : 2

Image Filtrée avec RELU

POOL MAX

Image Après Pool Max

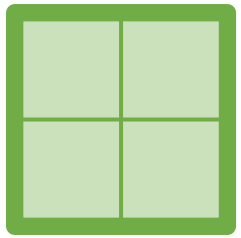


POOLING (POOL MAX)

Kernel size : 2 x 2

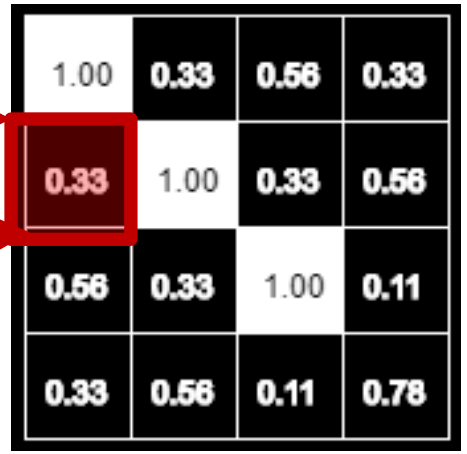
Stride : 2

Image Filtrée avec RELU



POOL MAX

Image Après Pool Max

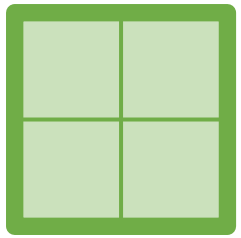


POOLING (POOL MAX)

Kernel size : 2 x 2

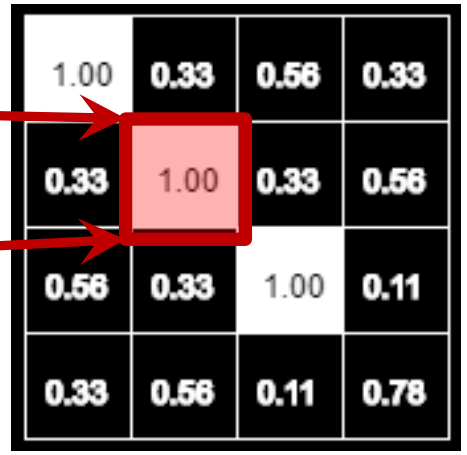
Stride : 2

Image Filtrée avec RELU



POOL MAX

Image Après Pool Max

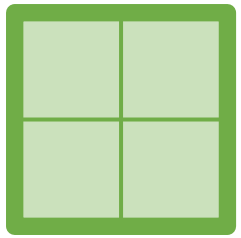


POOLING (POOL MAX)

Kernel size : 2 x 2

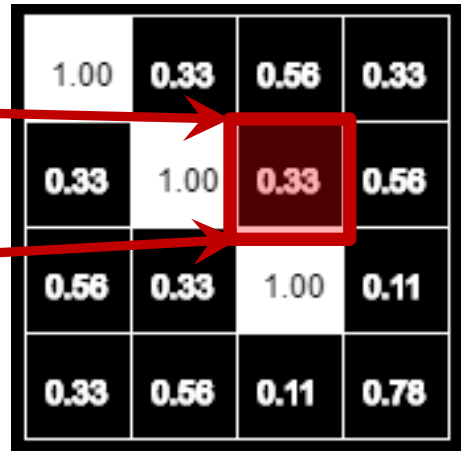
Stride : 2

Image Filtrée avec RELU



POOL MAX

Image Après Pool Max

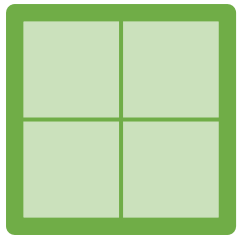


POOLING (POOL MAX)

Kernel size : 2 x 2

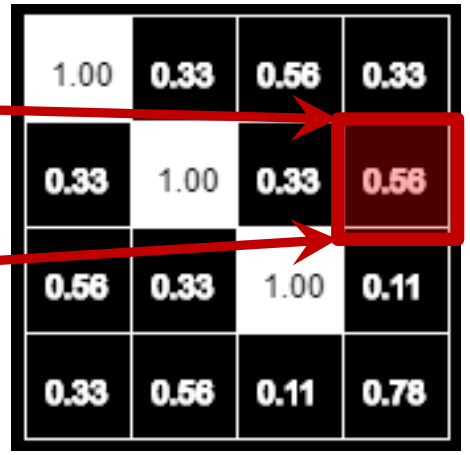
Stride : 2

Image Filtrée avec RELU



POOL MAX

Image Après Pool Max

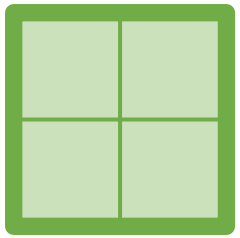


POOLING (POOL MAX)

Kernel size : 2 x 2

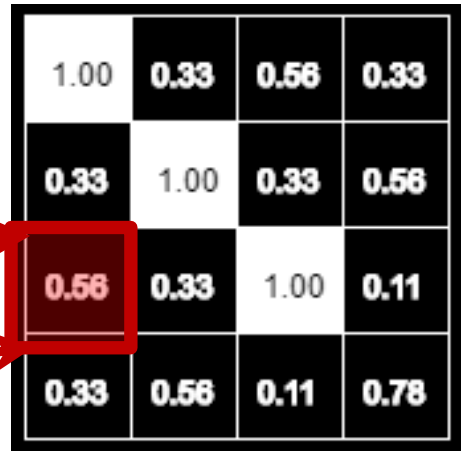
Stride : 2

Image Filtrée avec RELU



POOL MAX

Image Après Pool Max



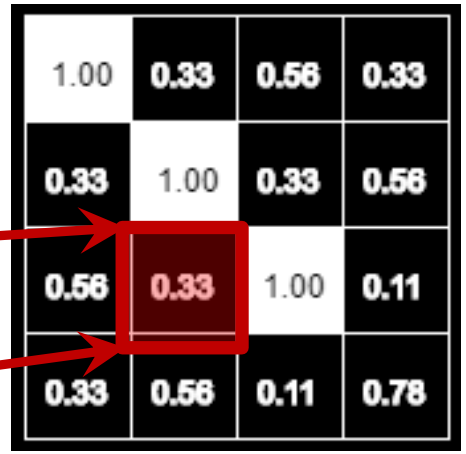
POOLING (POOL MAX)

Kernel size : 2 x 2

Stride : 2

Image Filtrée avec RELU

Image Après Pool Max



POOLING (POOL MAX)

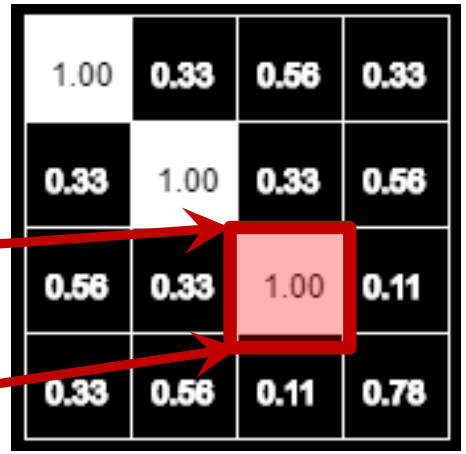
Kernel size : 2 x 2

Stride : 2

Image Filtrée avec RELU

POOL MAX

Image Après Pool Max

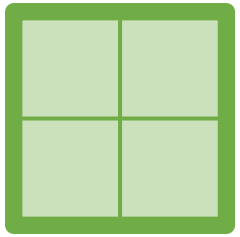


POOLING (POOL MAX)

Kernel size : 2 x 2

Stride : 2

Image Filtrée avec RELU



POOL MAX

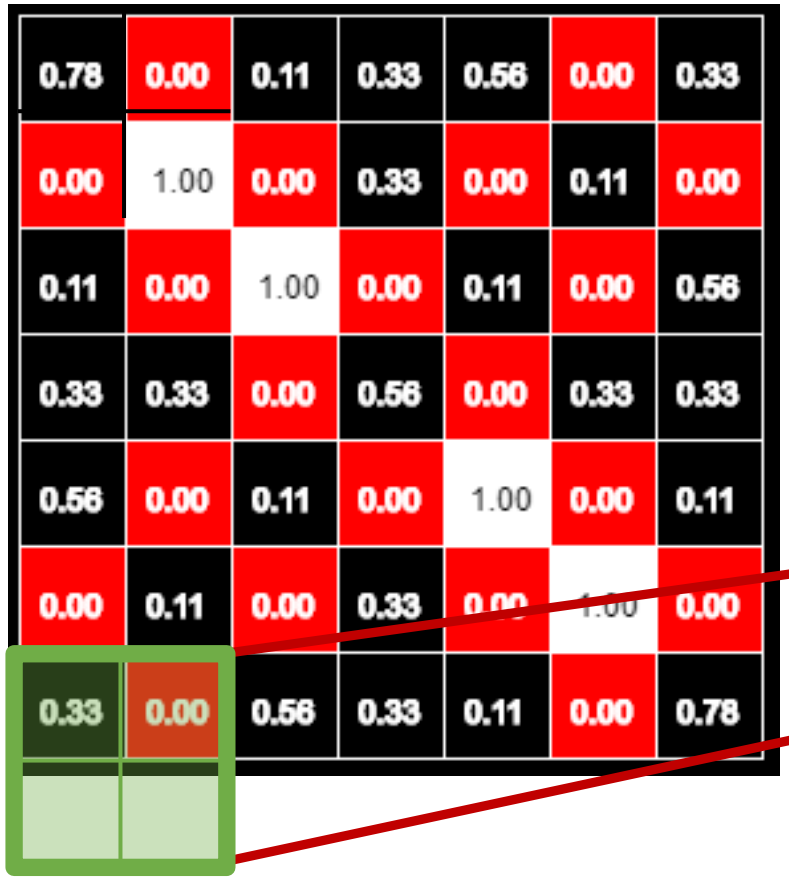
Image Après Pool Max

POOLING (POOL MAX)

Kernel size : 2 x 2

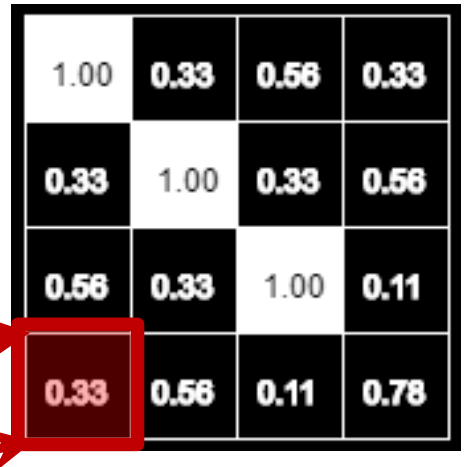
Stride : 2

Image Filtrée avec RELU



POOL MAX

Image Après Pool Max



POOLING (POOL MAX)

Kernel size : 2 x 2

Stride : 2

Image Filtrée avec RELU

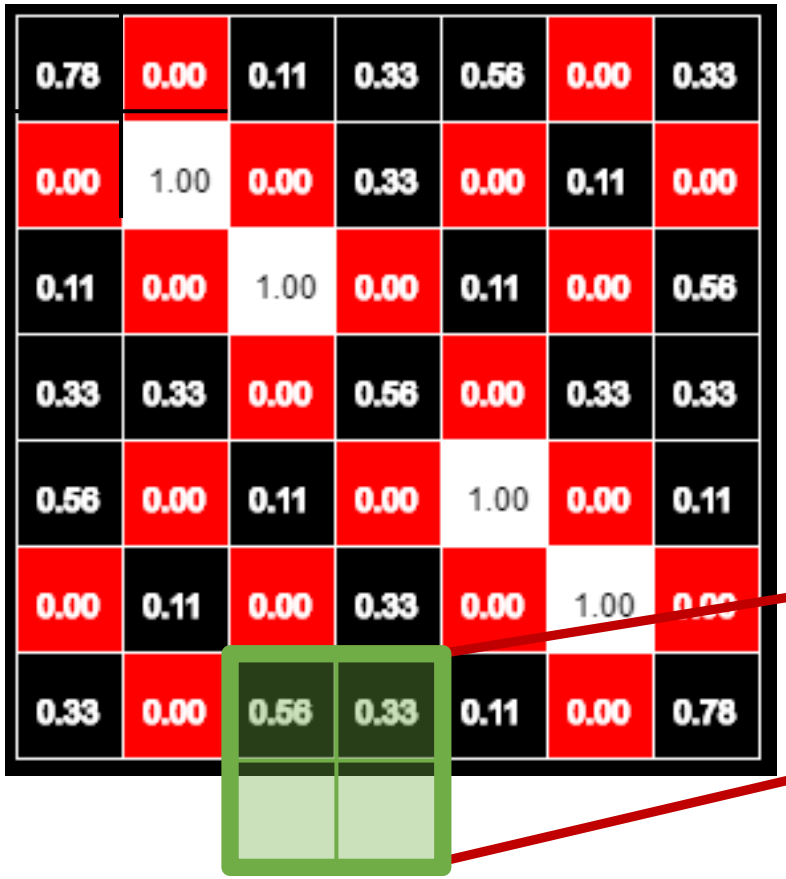


Image Après Pool Max

POOLING (POOL MAX)

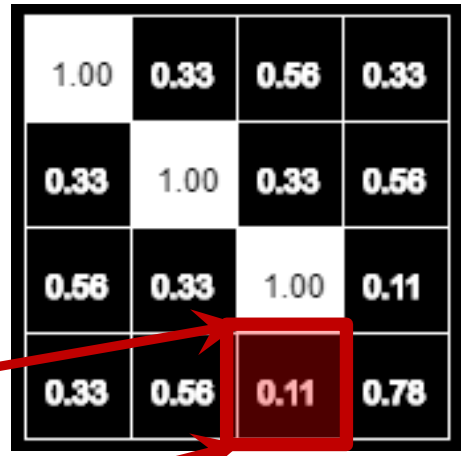
Kernel size : 2 x 2

Stride : 2

Image Filtrée avec RELU

POOL MAX

Image Après Pool Max



POOLING (POOL MAX)

Kernel size : 2 x 2

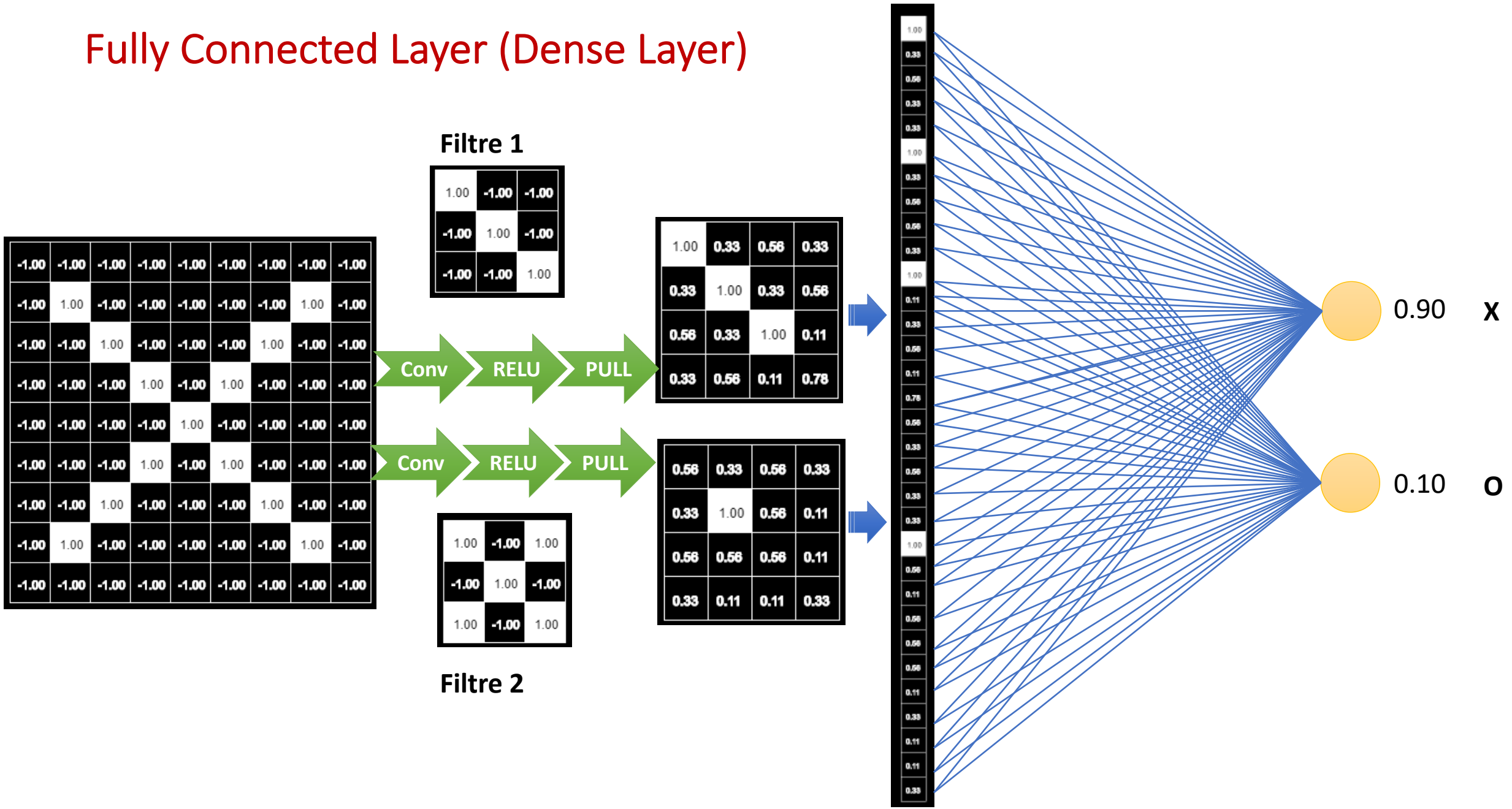
Stride : 2

Image Filtrée avec RELU

POOL MAX

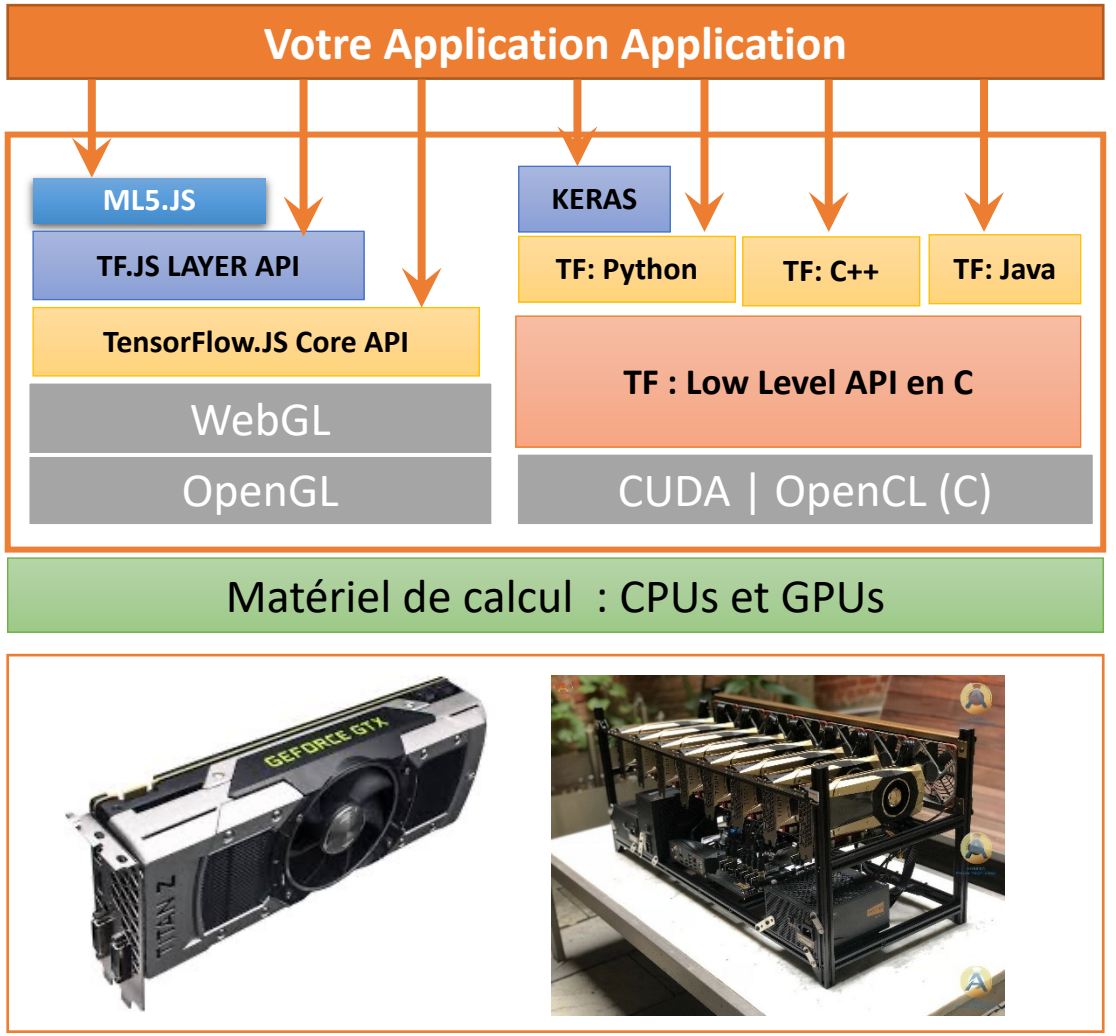
Image Après Pool Max

Fully Connected Layer (Dense Layer)



Principaux Framework de Deep Learning

- **TensorFlow**
  - **Développeur** : Google
  - **Langages principaux** : Python, C++, Java Script
  - Support de déploiement sur mobile, cloud, ou navigateur web (TensorFlow Lite, TF.js).
  - Support du calcul parallèle et du GPU/TPU.
- **PyTorch**
  - **Développeur** : Facebook (Meta) AI Research
  - **Langages principaux** : Python, C++
  - **Présentation** : PyTorch est un framework open source pour le deep learning qui privilégie la simplicité et l'intuitivité.
  - Support du calcul parallèle et du GPU/TPU.
- **Keras**
  - **Développeur** : Initialement par François Chollet, maintenant maintenu par la communauté TensorFlow.
  - **Langages principaux** : Python
  - **Présentation** : Keras fonctionne principalement comme une API par-dessus TensorFlow. Keras est aujourd'hui intégré nativement à TensorFlow (tf.keras).
- **DeepLearning4J (DL4J) :**
  - **Développeur** : Skymind (maintenant Konduit AI)
  - **Langage principal** : Java (et Scala)
  - **Présentation** : Conçu pour l'écosystème **Java** et **JVM** (Java Virtual Machine).
  - Support du calcul parallèle et du GPU/TPU.



ANN

```
!pip install -q tensorflow
!pip install -q pandas
!pip install -q seaborn
!pip install -q matplotlib
```

```
import tensorflow as tf
from tensorflow import keras
```

✓ 0.0s

```
(X_train,y_train),(X_test, y_test) = keras.datasets.mnist.load_data()
```

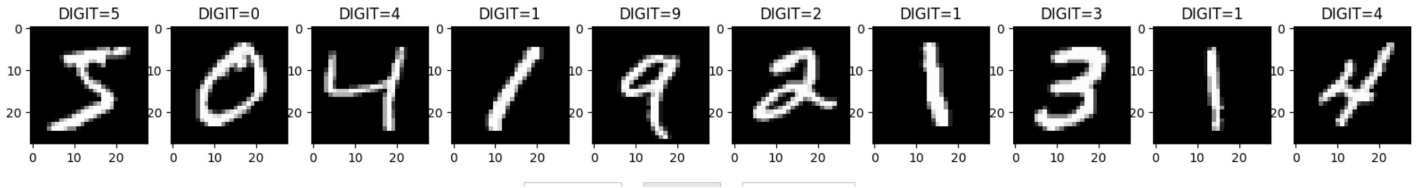
✓ 0.1s

```
print(X_train.shape)
print(X_test.shape)
```

✓ 0.0s

```
(60000, 28, 28)
(10000, 28, 28)
```

```
import matplotlib.pyplot as plt
plt.gray()
figure, axis = plt.subplots(ncols=10, figsize=(20,20))
for index in range(10):
    axis[index].imshow(X_train[index])
    axis[index].set_title(f"DIGIT={y_train[index]}")
plt.show()
```



Mnist Dataset

ANN

```
gpus = tf.config.experimental.list_logical_devices('GPU')
for gpu in gpus:
    tf.config.experimental.set_memory_growth(gpu,True)
```

✓ 0.0s

```
print(gpus)
```

✓ 0.0s

```
[]
```

```
ann = keras.Sequential([
    keras.layers.Input((28,28)),
    keras.layers.Flatten(),
    keras.layers.Dense(100, activation='sigmoid'),
    keras.layers.Dense(10, activation='sigmoid')
])
ann.compile(
    optimizer='adam',
    loss= 'sparse_categorical_crossentropy',
    metrics=['accuracy']
    )
```

```
ann.summary()
```

✓ 0.0s

Model: "sequential_25"

Layer (type)	Output Shape	Param #
flatten_16 (Flatten)	(None, 784)	0
dense_41 (Dense)	(None, 100)	78,500
dense_42 (Dense)	(None, 10)	1,010

Total params: 79,510 (310.59 KB)

Trainable params: 79,510 (310.59 KB)

Non-trainable params: 0 (0.00 B)

ANN

```
history1 = ann.fit(X_train, y_train, epochs=10, validation_split=0.3)
```

✓ 14.1s

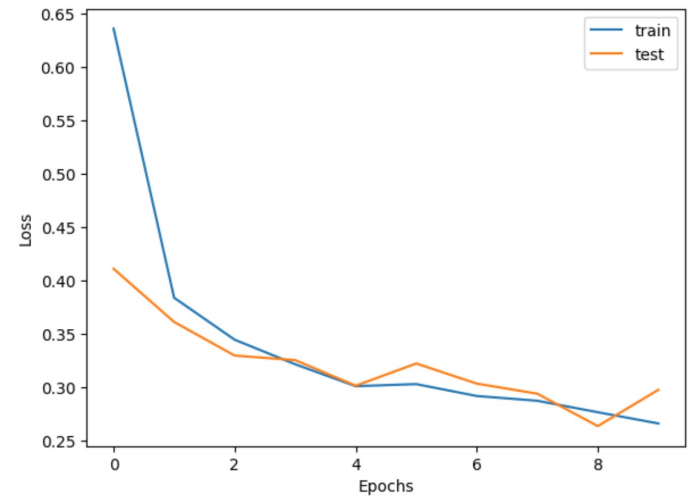
```
Epoch 1/10
1313/1313 ━━━━━━━━━━━━━━━━━━━━ 2s 1ms/step - accuracy: 0.7418 - loss: 0.9331 - val_accuracy: 0.8904 - val_loss: 0.
Epoch 2/10
1313/1313 ━━━━━━━━━━━━━━━━━━━━ 1s 1ms/step - accuracy: 0.8874 - loss: 0.3995 - val_accuracy: 0.9004 - val_loss: 0.
Epoch 3/10
1313/1313 ━━━━━━━━━━━━━━━━━━━━ 1s 1ms/step - accuracy: 0.8990 - loss: 0.3477 - val_accuracy: 0.9051 - val_loss: 0.
Epoch 4/10
1313/1313 ━━━━━━━━━━━━━━━━━━━━ 1s 1ms/step - accuracy: 0.9060 - loss: 0.3228 - val_accuracy: 0.9112 - val_loss: 0.
Epoch 5/10
1313/1313 ━━━━━━━━━━━━━━━━━━━━ 1s 1ms/step - accuracy: 0.9046 - loss: 0.3248 - val_accuracy: 0.9057 - val_loss: 0.
Epoch 6/10
1313/1313 ━━━━━━━━━━━━━━━━━━━━ 1s 982us/step - accuracy: 0.9087 - loss: 0.3047 - val_accuracy: 0.9159 - val_loss:
Epoch 7/10
1313/1313 ━━━━━━━━━━━━━━━━━━━━ 1s 1ms/step - accuracy: 0.9157 - loss: 0.2846 - val_accuracy: 0.9186 - val_loss: 0.
Epoch 8/10
1313/1313 ━━━━━━━━━━━━━━━━━━━━ 1s 1ms/step - accuracy: 0.9165 - loss: 0.2709 - val_accuracy: 0.9152 - val_loss: 0.
Epoch 9/10
1313/1313 ━━━━━━━━━━━━━━━━━━━━ 1s 1ms/step - accuracy: 0.9157 - loss: 0.2789 - val_accuracy: 0.9182 - val_loss: 0.
Epoch 10/10
1313/1313 ━━━━━━━━━━━━━━━━━━━━ 1s 1ms/step - accuracy: 0.9208 - loss: 0.2635 - val_accuracy: 0.9161 - val_loss: 0.
```

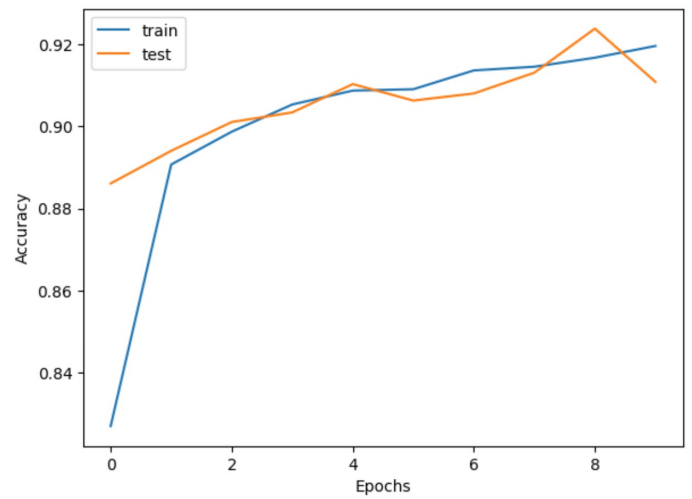
```
train_evaluation =ann.evaluate(X_train, y_train)
test_evaluation= ann.evaluate(X_test, y_test)
print("-"*80)
print(f"Train Loss = {train_evaluation[0]}, Train Accuracy = {train_evaluation[1]}")
print("-"*80)
print(f"Test Loss = {test_evaluation[0]}, Test Accuracy = {test_evaluation[1]}")
print("-"*80)
```

✓ 1.0s

```
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 1s 451us/step - accuracy: 0.9197 - loss: 0.2700
313/313 ━━━━━━━━━━━━━━━━━━━━ 0s 497us/step - accuracy: 0.9099 - loss: 0.3057
--------------------------------------------------------------------------------
Train Loss = 0.27229562401771545, Train Accuracy = 0.9186999797821045
--------------------------------------------------------------------------------
Test Loss = 0.27174338698387146, Test Accuracy = 0.9207000136375427
--------------------------------------------------------------------------------
```

```
figure, axis = plt.subplots(ncols=2, figsize=(15,5))
axis[0].plot(history1.history['loss'])
axis[0].plot(history1.history['val_loss'])
axis[0].set_xlabel('Epochs')
axis[0].set_ylabel('Loss')
axis[0].legend(['train','test'])
axis[1].plot(history1.history['accuracy'])
axis[1].plot(history1.history['val_accuracy'])
axis[1].set_xlabel('Epochs')
axis[1].set_ylabel('Accuracy')
axis[1].legend(['train','test'])
plt.show()
```





ANN

```
from sklearn.metrics import classification_report
import numpy as np
import seaborn as sns
```

✓ 0.1s

```
pred = ann.predict(X_test)
predictions = np.argmax(pred, axis=1)
predictions
```

✓ 0.1s

313/313 ━━━━━━━━━━━━━━━━━━━━ **0s** 358us/step

```
array([7, 2, 1, ..., 4, 5, 6])
```

```
print(classification_report(y_test, predictions))
```

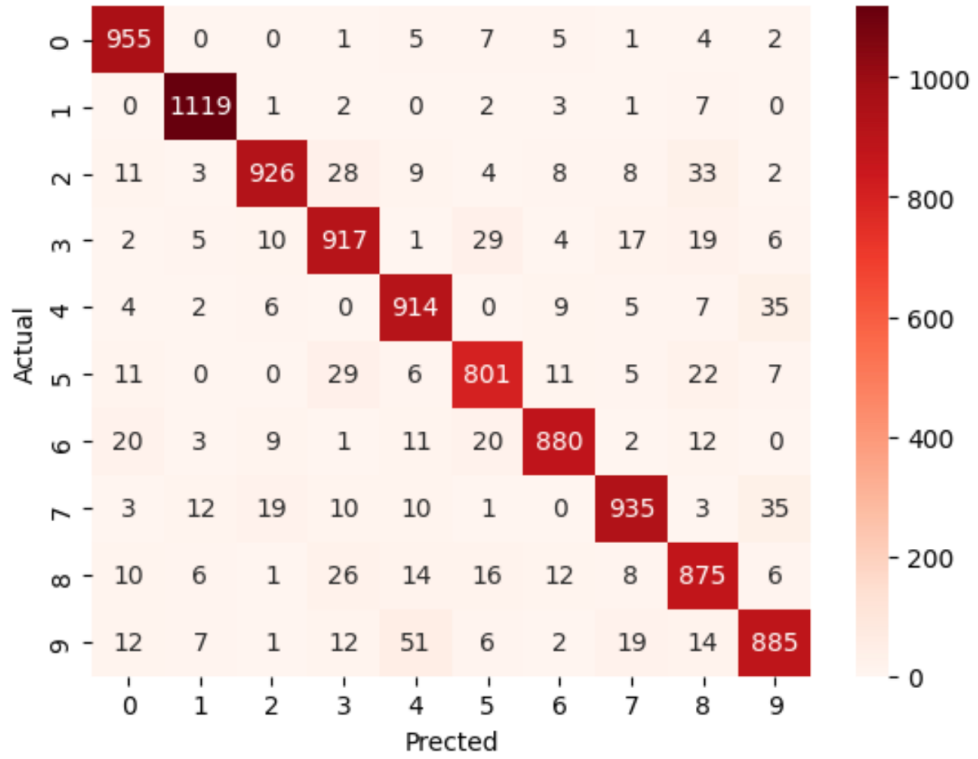
✓ 0.0s

	precision	recall	f1-score	support
0	0.93	0.97	0.95	980
1	0.97	0.99	0.98	1135
2	0.95	0.90	0.92	1032
3	0.89	0.91	0.90	1010
4	0.90	0.93	0.91	982
5	0.90	0.90	0.90	892
6	0.94	0.92	0.93	958
7	0.93	0.91	0.92	1028
8	0.88	0.90	0.89	974
9	0.90	0.88	0.89	1009
accuracy			0.92	10000
macro avg	0.92	0.92	0.92	10000
weighted avg	0.92	0.92	0.92	10000

```
cm = tf.math.confusion_matrix(labels=y_test, predictions=predictions)
sns.heatmap(cm, annot=True, fmt='d', cmap='Reds')
plt.xlabel('Prected')
plt.ylabel('Actual')
```

✓ 0.1s

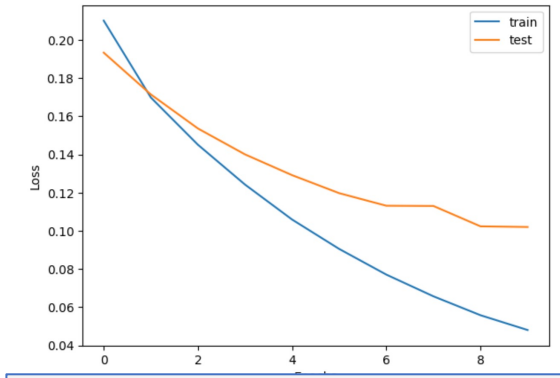
```
Text(50.722222222222214, 0.5, 'Actual')
```

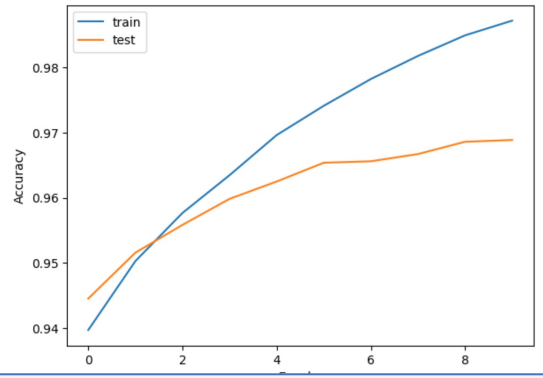


ANN

Scale Data

```
X_train_scaled = X_train / 255.0
X_test_scaled = X_test / 255.0
history2 = ann.fit(X_train_scaled, y_train, epochs=10, validation_split=0.3)
```






```
train_evaluation =ann.evaluate(X_train, y_train)
test_evaluation= ann.evaluate(X_test, y_test)
print("-"*80)
print(f"Train Loss = {train_evaluation[0]}, Train Accuracy = {train_evaluation[1]}")
print("-"*80)
print(f"Test Loss = {test_evaluation[0]}, Test Accuracy = {test_evaluation[1]}")
print("-"*80)
```

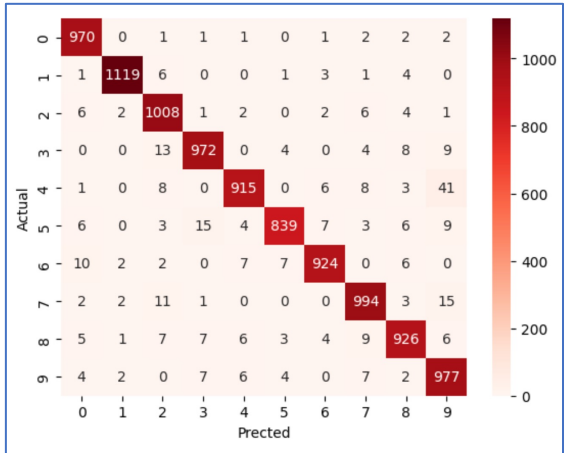
✓ 1.0s

```
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 1s 455us/step - accuracy: 0.9832 - loss: 0.0544
313/313 ━━━━━━━━━━━━━━━━━━━━ 0s 459us/step - accuracy: 0.9609 - loss: 0.1376
--------------------------------------------------------------------------------
Train Loss = 0.07728223502635956, Train Accuracy = 0.9763666391372681
--------------------------------------------------------------------------------
Test Loss = 0.12216612696647644, Test Accuracy = 0.9643999934196472
--------------------------------------------------------------------------------
```

```
train_evaluation =ann.evaluate(X_train, y_train)
test_evaluation= ann.evaluate(X_test, y_test)
print("-"*80)
print(f"Train Loss = {train_evaluation[0]}, Train Accuracy = {train_evaluation[1]}")
print("-"*80)
print(f"Test Loss = {test_evaluation[0]}, Test Accuracy = {test_evaluation[1]}")
print("-"*80)

pred = ann.predict(X_test)
predictions = np.argmax(pred, axis=1)
print(classification_report(y_test, predictions))

cm = tf.math.confusion_matrix(labels=y_test, predictions=predictions)
sns.heatmap(cm, annot=True, fmt='d', cmap='Reds')
plt.xlabel('Prected')
plt.ylabel('Actual')
plt.show()
```

```
figure, axis = plt.subplots(ncols=2, figsize=(15,5))
axis[0].plot(history2.history['loss'])
axis[0].plot(history2.history['val_loss'])
axis[0].set_xlabel('Epochs')
axis[0].set_ylabel('Loss')
axis[0].legend(['train','test'])
axis[1].plot(history2.history['accuracy'])
axis[1].plot(history2.history['val_accuracy'])
axis[1].set_xlabel('Epochs')
axis[1].set_ylabel('Accuracy')
axis[1].legend(['train','test'])
plt.show()
```

```
313/313 ━━━━━━━━━━━━━━━━━━━━ 0s 412us/step
              precision    recall  f1-score   support

           0       0.97      0.99      0.98       980
           1       0.99      0.99      0.99      1135
           2       0.95      0.98      0.96      1032
           3       0.97      0.96      0.97      1010
           4       0.97      0.93      0.95       982
           5       0.98      0.94      0.96       892
           6       0.98      0.96      0.97       958
           7       0.96      0.97      0.96      1028
           8       0.96      0.95      0.96       974
           9       0.92      0.97      0.94      1009

    accuracy                           0.96     10000
   macro avg       0.96      0.96      0.96     10000
weighted avg       0.96      0.96      0.96     10000
```



CNN

CNN

```
cnn = keras.Sequential([
    keras.layers.Input((28,28,1)),
    keras.layers.Conv2D(filters=32, kernel_size=(3,3), activation='relu'),
    keras.layers.MaxPool2D((2,2)),
    keras.layers.Conv2D(filters=64, kernel_size=(3,3), activation='relu'),
    keras.layers.MaxPool2D((2,2)),
    keras.layers.Flatten(),
    #keras.layers.Dropout(0.2),
    keras.layers.Dense(100, activation='relu'),
    keras.layers.Dense(10, activation='softmax')
])
cnn.compile(
    optimizer='adam',
    loss= 'sparse_categorical_crossentropy',
    metrics=['accuracy']
    )
```

✓ 0.0s

```
cnn.summary()
```

✓ 0.0s

Model: "sequential_27"

Layer (type)	Output Shape	Param #
conv2d_14 (Conv2D)	(None, 26, 26, 32)	320
max_pooling2d_14 (MaxPooling2D)	(None, 13, 13, 32)	0
conv2d_15 (Conv2D)	(None, 11, 11, 64)	18,496
max_pooling2d_15 (MaxPooling2D)	(None, 5, 5, 64)	0
flatten_18 (Flatten)	(None, 1600)	0
dense_45 (Dense)	(None, 100)	160,100
dense_46 (Dense)	(None, 10)	1,010

Total params: 179,926 (702.84 KB)

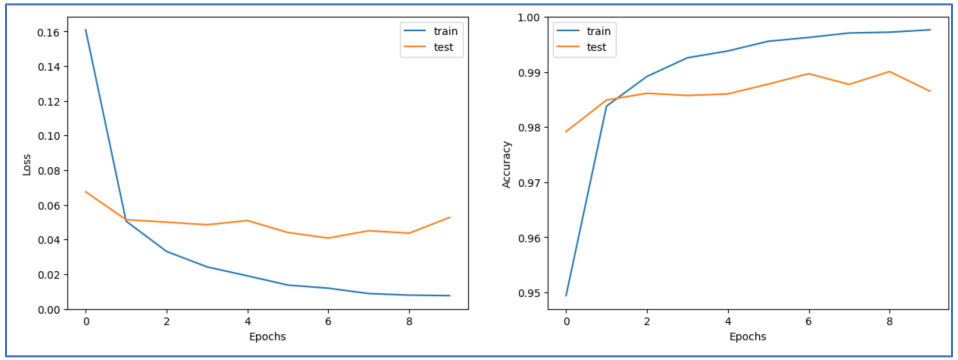
Trainable params: 179,926 (702.84 KB)

Non-trainable params: 0 (0.00 B)

CNN

```
X_train_scaled = X_train / 255.0
X_test_scaled = X_test / 255.0
history3 = cnn.fit(X_train_scaled, y_train, epochs=10, validation_split=0.3)
```

✓ 1m 29.2s

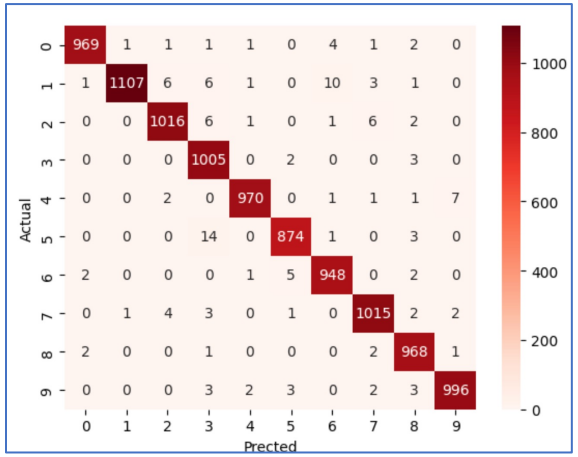



```
figure, axis = plt.subplots(ncols=2, figsize=(15,5))
axis[0].plot(history3.history['loss'])
axis[0].plot(history3.history['val_loss'])
axis[0].set_xlabel('Epochs')
axis[0].set_ylabel('Loss')
axis[0].legend(['train','test'])
axis[1].plot(history3.history['accuracy'])
axis[1].plot(history3.history['val_accuracy'])
axis[1].set_xlabel('Epochs')
axis[1].set_ylabel('Accuracy')
axis[1].legend(['train','test'])
plt.show()
```

```
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 4s 2ms/step - accuracy: 0.9954 - loss: 2.8304
313/313 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - accuracy: 0.9823 - loss: 17.6253
--------------------------------------------------------------------------------
Train Loss = 5.735527992248535, Train Accuracy = 0.9932833313941956
--------------------------------------------------------------------------------
Test Loss = 12.59516716003418, Test Accuracy = 0.9868000149726868
--------------------------------------------------------------------------------
```

```
train_evaluation =cnn.evaluate(X_train, y_train)
test_evaluation= cnn.evaluate(X_test, y_test)
print("-"*80)
print(f"Train Loss = {train_evaluation[0]}, Train Accuracy = {train_evaluation[1]}")
print("-"*80)
print(f"Test Loss = {test_evaluation[0]}, Test Accuracy = {test_evaluation[1]}")
print("-"*80)

pred = cnn.predict(X_test)
predictions = np.argmax(pred, axis=1)
print(classification_report(y_test, predictions))

cm = tf.math.confusion_matrix(labels=y_test, predictions=predictions)
sns.heatmap(cm, annot=True, fmt='d', cmap='Reds')
plt.xlabel('Prected')
plt.ylabel('Actual')
plt.show()
```

	precision	recall	f1-score	support
0	0.99	0.99	0.99	980
1	1.00	0.98	0.99	1135
2	0.99	0.98	0.99	1032
3	0.97	1.00	0.98	1010
4	0.99	0.99	0.99	982
5	0.99	0.98	0.98	892
6	0.98	0.99	0.99	958
7	0.99	0.99	0.99	1028
8	0.98	0.99	0.99	974
9	0.99	0.99	0.99	1009
accuracy			0.99	10000
macro avg	0.99	0.99	0.99	10000
weighted avg	0.99	0.99	0.99	10000



CNN

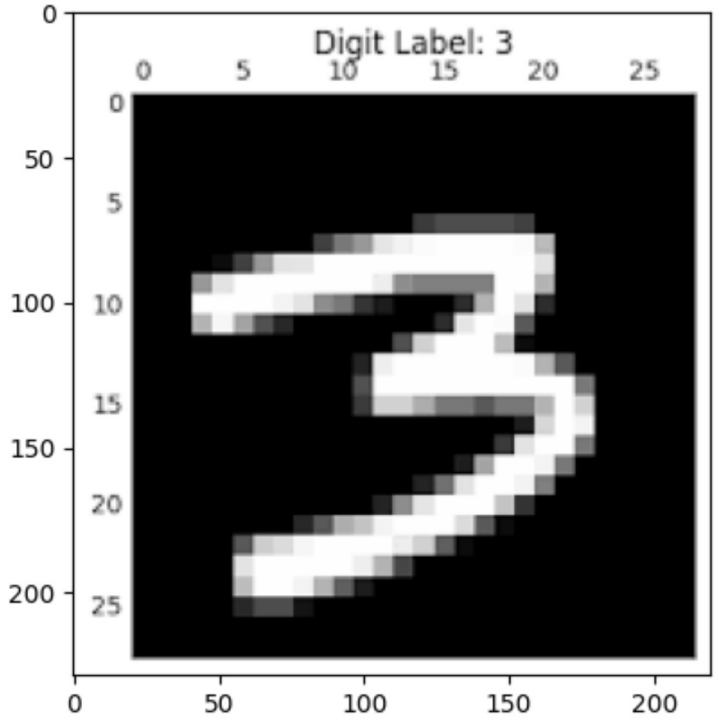
Prediction

```
!pip install opencv-python
```

✓ 1.6s

```
import cv2
img = cv2.imread('3.png')
plt.imshow(img)
plt.show()
```

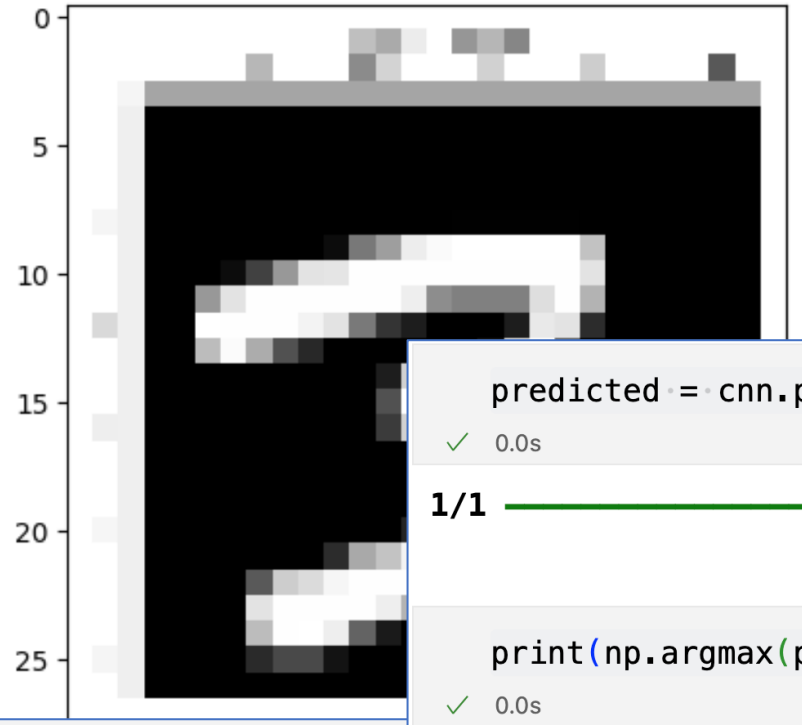
✓ 0.0s




```
img_resized = tf.image.resize(img,size=(28,28))
gray_image = tf.image.rgb_to_grayscale(img_resized)
plt.imshow(gray_image)
```

✓ 0.0s

```
<matplotlib.image.AxesImage at 0x3a4331fd0>
```



```
gray_image.shape
```

✓ 0.0s

```
TensorShape([28, 28, 1])
```

```
predicted = cnn.predict(np.expand_dims(gray_image/255,0))
```

✓ 0.0s

```
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 23ms/step
```

```
print(np.argmax(predicted))
```

✓ 0.0s

```
3
```

CNN

Saving the trained model

```
import os
```

✓ 0.0s

```
ann.save(os.path.join('models','ann_image_classifier_v1.h5'))
cnn.save(os.path.join('models','cnn_image_classifier_V1.h5'))
```

✓ 0.0s

Loading the trained model

```
from tensorflow.keras.models import load_model
new_model = load_model('models/cnn_image_classifier_V1.h5')
```

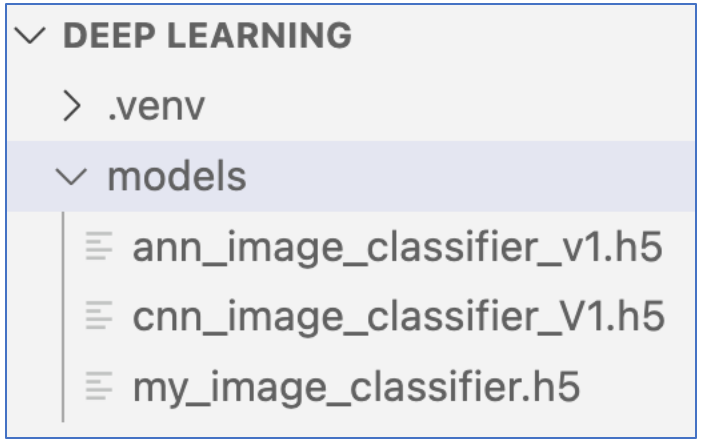
✓ 0.0s

```
WARNING:absl:Compiled the loaded model, but the compiled metrics
```

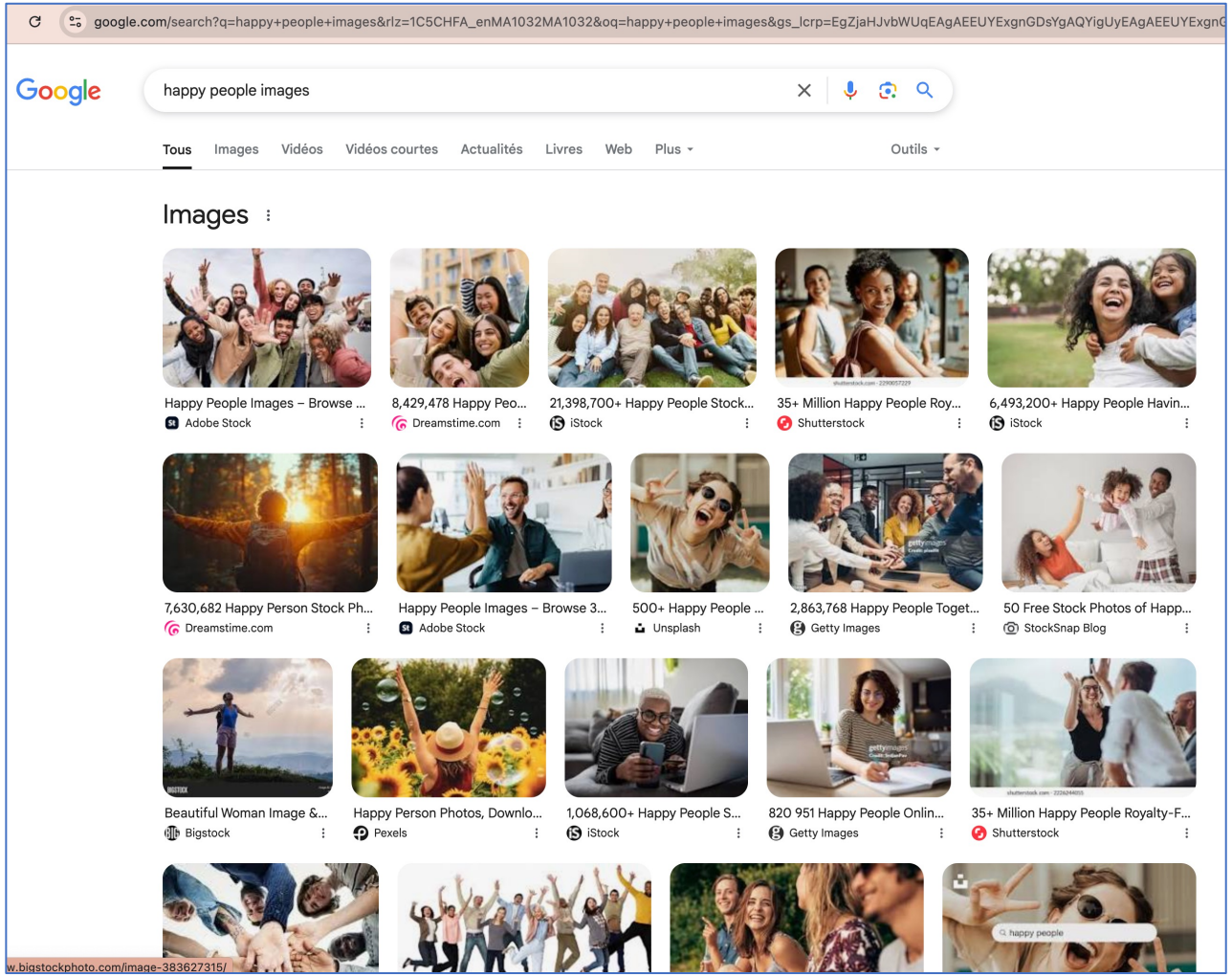
```
pred = new_model.predict(np.expand_dims(gray_image/255,0))
print(np.argmax(pred))
```

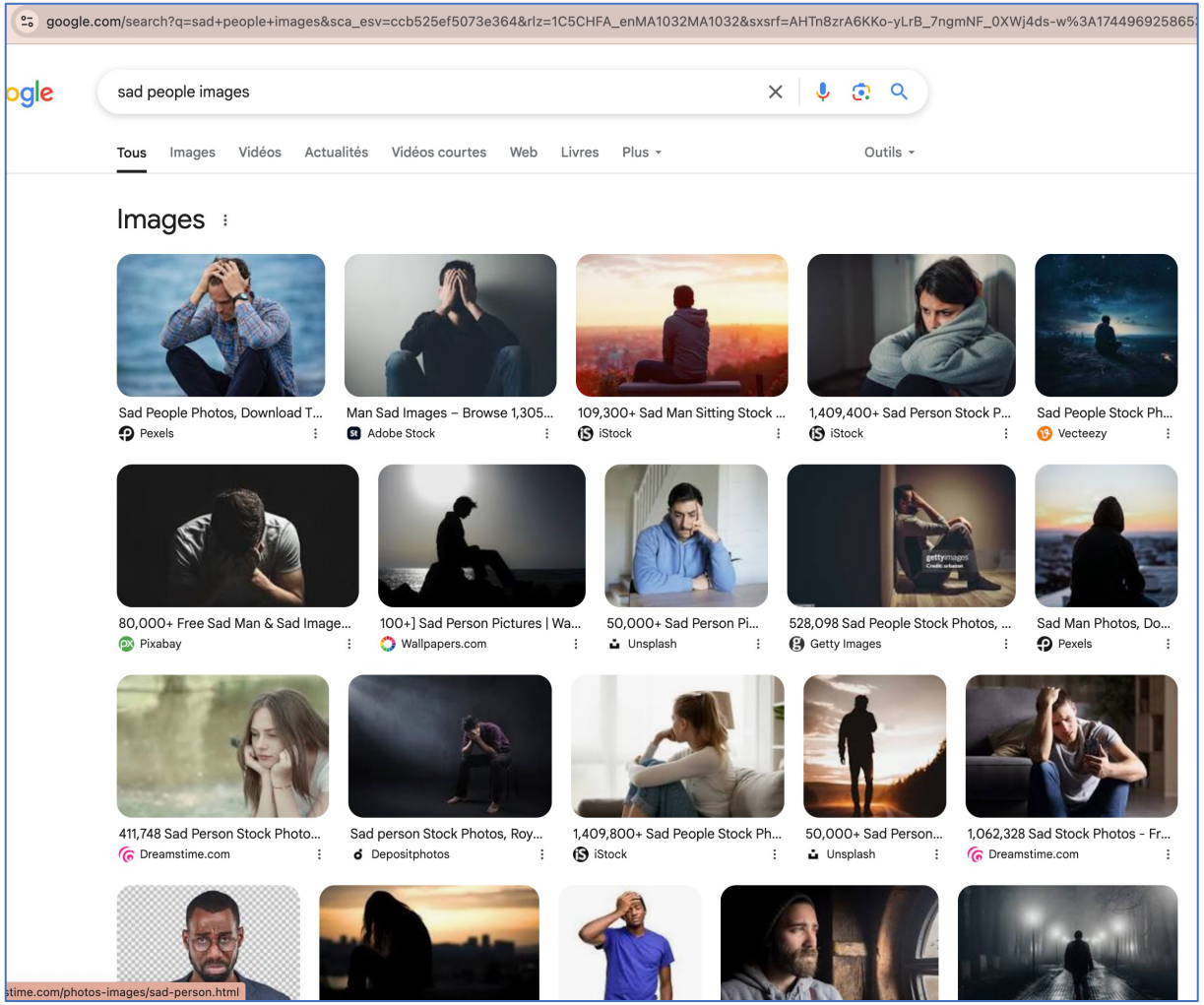
✓ 0.0s

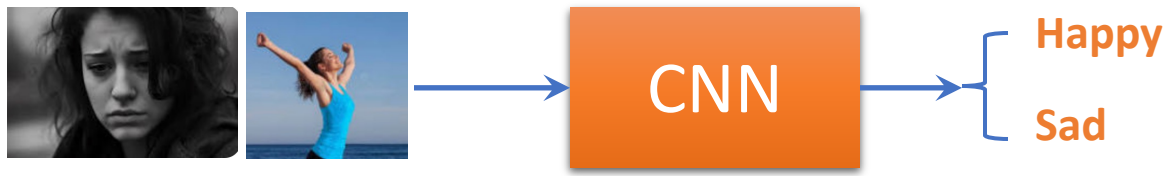
```
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 20ms/step
3
```



CNN : Application Image sentiment classification







CNN : Application Image sentiment classification

```
!pip -q install  tensorflow
```
✓ 0.7s

```
!pip -q install opencv-python
```
✓ 0.5s

```
#!pip install tensorflow-gpu
```
✓ 0.0s

```
!pip -q install matplotlib
```
✓ 0.5s

```
!pip list
```
✓ 0.4s

```
!pip list
```
✓ 0.4s

Package	Version
absl-py	2.2.2
appnope	0.1.4
asttokens	3.0.0
astunparse	1.6.3
certifi	2025.1.31
charset-normalizer	3.4.1
comm	0.2.2
debugpy	1.8.14
decorator	5.2.1
executing	2.2.0
flatbuffers	25.2.10
gast	0.6.0
google-pasta	0.2.0
grpcio	1.71.0
h5py	3.13.0
idna	3.10
ipykernel	6.29.5
ipython	9.1.0
ipython_pygments_lexers	1.1.1
jedi	0.19.2
jupyter_client	8.6.3
jupyter_core	5.7.2
keras	3.9.2
...	
wcwidth	0.2.13
Werkzeug	3.1.3
wheel	0.45.1
wrapt	1.17.2

```
import tensorflow as tf
import os
```
✓ 0.0s

```
# Avoid OOM errors by setting GPU Memory Consumption Growth
gpus = tf.config.experimental.list_physical_devices('GPU')
for gpu in gpus:
    tf.config.experimental.set_memory_growth(gpu, True)
```
✓ 0.0s

```
tf.config.list_physical_devices('GPU')
```
✓ 0.0s

```
[]
```

CNN : Application Image sentiment classification

Supprimer les images ayant autres que jpeg, jpg, bmp et png

```
import cv2
import imghdr
```

```
data_dir = 'data'
image_extentions_to_keep = ['jpeg','jpg', 'bmp', 'png']
```

✓ 0.0s

```
for image_class in os.listdir(data_dir):
    for image in os.listdir(os.path.join(data_dir, image_class)):
        image_path = os.path.join(data_dir, image_class, image)
        try:
            img = cv2.imread(image_path)
            tip = imghdr.what(image_path)
            if tip not in image_extentions_to_keep:
                os.remove(image_path)
        except Exception as e:
            print('Issue with image {}'.format(image_path))
```

✓ 1.6s

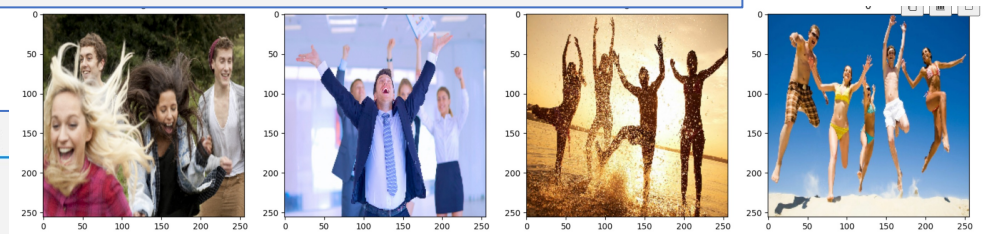
IMAGECLASSIFI...
- .venv
- data
  - happy
  - sad

```
data_iterator = data.as_numpy_iterator()
batch = data_iterator.next()
```

✓ 0.3s

```
fig, ax = plt.subplots(ncols=4, figsize=(20,20))
for idx, img in enumerate(batch[0][5:9]):
    ax[idx].imshow(img.astype(int))
    ax[idx].title.set_text(batch[1][idx])
```

✓ 0.2s



```
tf.keras.utils.image_dataset_from_directory??
```

✓ 0.0s

```
Signature:
tf.keras.utils.image_dataset_from_directory(
    directory,
    labels='inferred',
    label_mode='int',
    class_names=None,
    color_mode='rgb',
    batch_size=32,
    image_size=(256, 256),
    shuffle=True,
    seed=None,
    validation_split=None,
    subset=None,
    interpolation='bilinear',
    follow_links=False,
    crop_to_aspect_ratio=False,
    pad_to_aspect_ratio=False,
    data_format=None,
    verbose=True,
)
```

Charger les images

```
import numpy as np
from matplotlib import pyplot as plt
data = tf.keras.utils.image_dataset_from_directory('data', shuffle=False)
```

✓ 0.0s

CNN : Application Image sentiment classification

Normalisation des données

```
data = data.map(lambda x,y: (x/255, y))
```

✓ 0.0s

Split Data

```
train_size = int(len(data)*.7)
val_size = int(len(data)*.2)
test_size = int(len(data)*.1)
```

✓ 0.0s

```
train = data.take(train_size)
val = data.skip(train_size).take(val_size)
test = data.skip(train_size+val_size).take(test_size)
```

✓ 0.0s

CNN : Application Image sentiment classification

Construction du modèle CNN

```
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Input, Conv2D, MaxPooling2D, Dense, Flatten, Dropout
```

[1] ✓ 0.0s

```
model = Sequential()
model.add(Input((256,256,3)))
model.add(Conv2D(32, (3,3), 1, activation='relu'))
model.add(MaxPooling2D())
model.add(Conv2D(64, (3,3), 1, activation='relu'))
model.add(MaxPooling2D())
model.add(Conv2D(32, (3,3), 1, activation='relu'))
model.add(MaxPooling2D())
model.add(Flatten())
model.add(Dense(256, activation='relu'))
model.add(Dense(1, activation='sigmoid'))
```

```
model.compile('adam', loss=tf.losses.BinaryCrossentropy(), metrics=['accuracy'])
```

✓ 0.0s

```
model.summary()
```

Model: "sequential_13"

Layer (type)	Output Shape	Param #
conv2d_36 (Conv2D)	(None, 254, 254, 32)	896
max_pooling2d_36 (MaxPooling2D)	(None, 127, 127, 32)	0
conv2d_37 (Conv2D)	(None, 125, 125, 64)	18,496
max_pooling2d_37 (MaxPooling2D)	(None, 62, 62, 64)	0
conv2d_38 (Conv2D)	(None, 60, 60, 32)	18,464
max_pooling2d_38 (MaxPooling2D)	(None, 30, 30, 32)	0
flatten_12 (Flatten)	(None, 28800)	0
dense_24 (Dense)	(None, 256)	7,373,056
dense_25 (Dense)	(None, 1)	257

Total params: 7,411,169 (28.27 MB)

Trainable params: 7,411,169 (28.27 MB)

Non-trainable params: 0 (0.00 B)

CNN : Application Image sentiment classification

Entrainer le modèle

```
history = model.fit(train, epochs=20, validation_data=val)
```

✓ 41.9s

```
Epoch 1/20
7/7 ━━━━━━━━━━━━━━━━━━━━ 4s 479ms/step - accuracy: 0.5479 - loss: 1.8532 - val_accuracy: 0.0000e+00 - val_loss: 2.1496
Epoch 2/20
7/7 ━━━━━━━━━━━━━━━━━━━━ 3s 461ms/step - accuracy: 0.6408 - loss: 0.4672 - val_accuracy: 0.3750 - val_loss: 0.6935
Epoch 3/20
7/7 ━━━━━━━━━━━━━━━━━━━━ 3s 455ms/step - accuracy: 0.6919 - loss: 0.6918 - val_accuracy: 0.0469 - val_loss: 0.7227
Epoch 4/20
7/7 ━━━━━━━━━━━━━━━━━━━━ 3s 456ms/step - accuracy: 0.8899 - loss: 0.7067 - val_accuracy: 0.0000e+00 - val_loss: 1.3316
Epoch 5/20
7/7 ━━━━━━━━━━━━━━━━━━━━ 3s 454ms/step - accuracy: 0.9051 - loss: 0.3828 - val_accuracy: 0.2500 - val_loss: 0.7237
Epoch 6/20
7/7 ━━━━━━━━━━━━━━━━━━━━ 3s 457ms/step - accuracy: 0.9086 - loss: 0.6095 - val_accuracy: 0.2344 - val_loss: 0.7552
Epoch 7/20
7/7 ━━━━━━━━━━━━━━━━━━━━ 3s 454ms/step - accuracy: 0.8957 - loss: 0.5710 - val_accuracy: 0.20
Epoch 8/20
7/7 ━━━━━━━━━━━━━━━━━━━━ 3s 450ms/step - accuracy: 0.9155 - loss: 0.3921 - val_accuracy: 0.39
Epoch 9/20
7/7 ━━━━━━━━━━━━━━━━━━━━ 3s 451ms/step - accuracy: 0.9071 - loss: 0.3928 - val_accuracy: 0.40
Epoch 10/20
7/7 ━━━━━━━━━━━━━━━━━━━━ 3s 455ms/step - accuracy: 0.9210 - loss: 0.3290 - val_accuracy: 0.50
Epoch 11/20
7/7 ━━━━━━━━━━━━━━━━━━━━ 3s 451ms/step - accuracy: 0.9078 - loss: 0.3167 - val_accuracy: 0.64
Epoch 12/20
7/7 ━━━━━━━━━━━━━━━━━━━━ 3s 456ms/step - accuracy: 0.8983 - loss: 0.3005 - val_accuracy: 0.67
Epoch 13/20
...
Epoch 19/20
7/7 ━━━━━━━━━━━━━━━━━━━━ 3s 455ms/step - accuracy: 0.9811 - loss: 0.1140 - val_accuracy: 0.59
Epoch 20/20
7/7 ━━━━━━━━━━━━━━━━━━━━ 3s 452ms/step - accuracy: 0.9823 - loss: 0.0316 - val_accuracy: 0.73
```

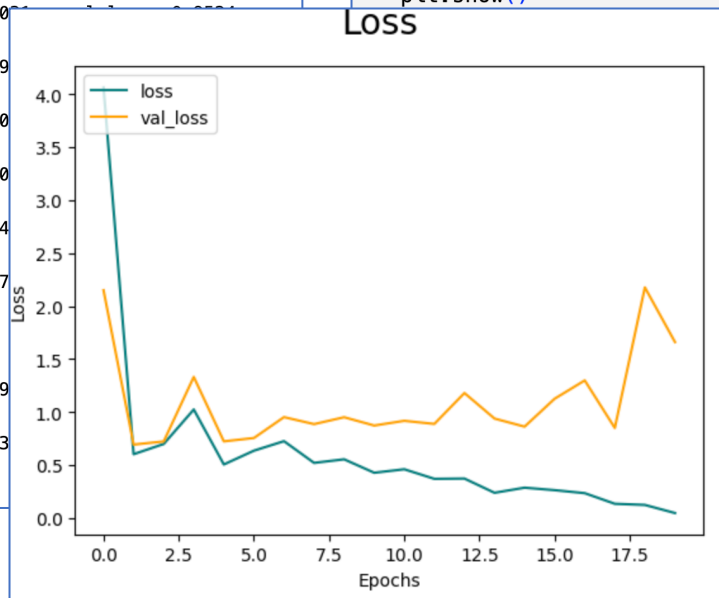
Output is truncated. View as a scrollable element or open in a text editor. Adjust cell output settings...

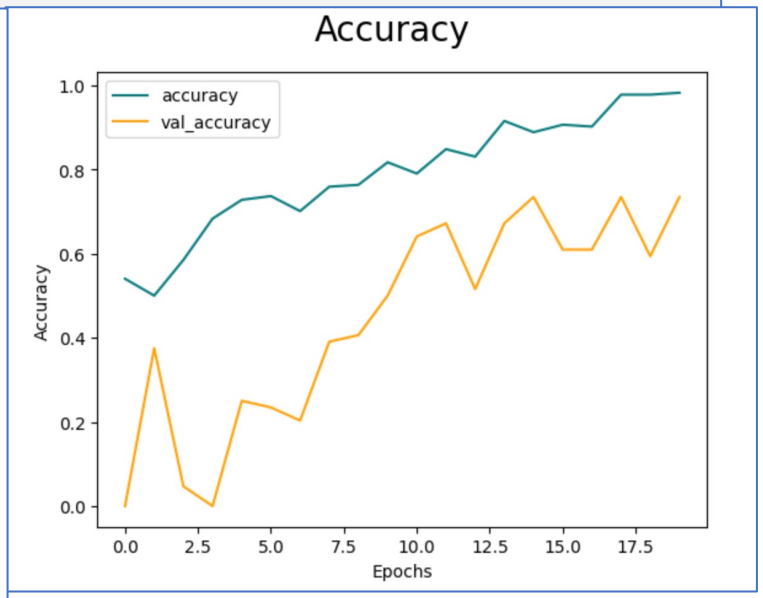
Plot Performance

```
fig = plt.figure()
plt.plot(history.history['loss'], color='teal', label='loss')
plt.plot(history.history['val_loss'], color='orange', label='val_loss')
fig.suptitle('Loss', fontsize=20)
plt.xlabel('Epochs')
plt.ylabel('Loss')
plt.legend(loc="upper left")
plt.show()
```

✓ 0.0s

```
fig = plt.figure()
plt.plot(history.history['accuracy'], color='teal', label='accuracy')
plt.plot(history.history['val_accuracy'], color='orange', label='val_accuracy')
fig.suptitle('Accuracy', fontsize=20)
plt.xlabel('Epochs')
plt.ylabel('Accuracy')
plt.legend(loc="upper left")
plt.show()
```





CNN : Application Image sentiment classification

Evaluation du modèle CNN

```
from tensorflow.keras.metrics import Precision, Recall, BinaryAccuracy
```

✓ 0.0s

```
pre = Precision()
re = Recall()
acc = BinaryAccuracy()
```

✓ 0.0s

```
for batch in test.as_numpy_iterator():
    X, y = batch
    yhat = model.predict(X)
    pre.update_state(y, yhat)
    re.update_state(y, yhat)
    acc.update_state(y, yhat)
```

✓ 0.4s

```
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 94ms/step
```

```
print(pre.result(), re.result(), acc.result())
```

✓ 0.0s

```
tf.Tensor(1.0, shape=(), dtype=float32) tf.Tensor(0.5294118, shape=(), dtype=float32) tf.Tensor(0.5294118, shape=(), dtype=float32)
```

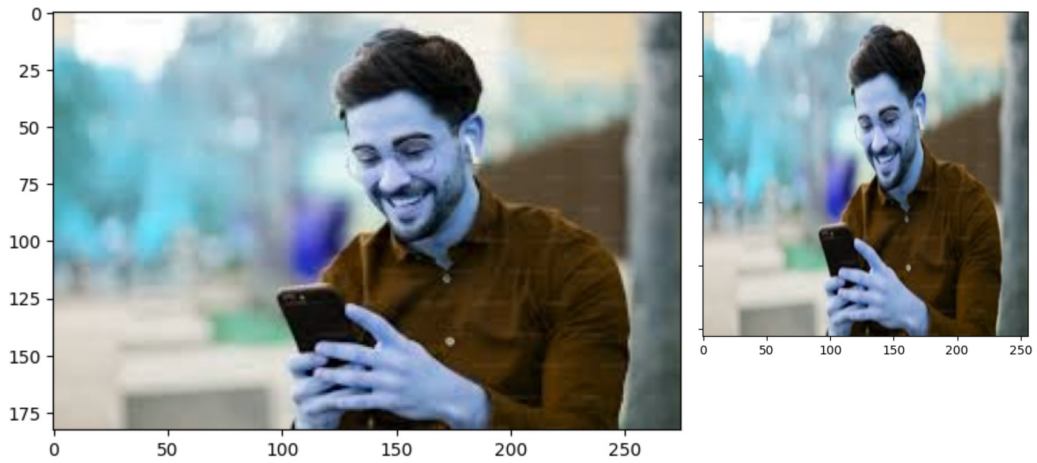
CNN : Application Image sentiment classification

Tester le modèle CNN avec une prédiction quelconque

```
import cv2
```
✓ 0.0s

```
img = cv2.imread('sample.jpeg')
plt.imshow(img)
plt.show()
```
✓ 0.0s

```
resize = tf.image.resize(img, (256,256))
plt.imshow(resize.numpy().astype(int))
plt.show()
```
✓ 0.0s




```
yhat = model.predict(np.expand_dims(resize/255, 0))
```
✓ 0.0s

```
yhat
```
✓ 0.0s

```
array([[0.06212844]], dtype=float32)
```

```
if yhat > 0.5:
    print(f'Predicted class is Sad')
else:
    print(f'Predicted class is Happy')
```
✓ 0.0s

```
Predicted class is Happy
```

CNN : Application Image sentiment classification

Enregistrer le modèle

```
[204]  model.save(os.path.join('models','cnn1.h5'))
       ✓ 0.0s
```

Charger le modèle

```
[5]  from tensorflow.keras.models import load_model
     ✓ 0.0s
```

```
[6]  new_model = load_model('models/cnn1.h5')
     ✓ 0.0s
```

```
pred =new_model.predict(np.expand_dims(resize/255, 0))
✓ 0.0s
```

```
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 28ms/step
```

```
if pred > 0.5:
    print(f'Predicted class is Sad')
else:
    print(f'Predicted class is Happy')
✓ 0.0s
```

```
Predicted class is Happy
```

Quelques Rappels en statistiques

Analyse de données statistiques : espérance, variance, écart-type

L'**espérance** d'une série statistique $X = (x_i)_{i=1,\dots,N}$ est la moyenne des valeurs de cette série statistique :

$$E(X) = \bar{X} = \frac{x_1 + \cdots + x_N}{N}.$$

La **variance** et l'écart-type mesurent eux la dispersion des valeurs de cette série statistique autour de sa moyenne. La variance $V(X)$ est définie par

$$V(X) = \frac{1}{N}\left((x_1 - \bar{X})^2 + \cdots + (x_N - \bar{X})^2\right) = \frac{1}{N}\sum_{k=1}^{N}\left(x_k - \bar{X}\right)^2.$$

Elle est donc égale à la moyenne des carrés des différences entre les observations x_i et leur moyenne $\bar{X}$.

L'écart-type, noté σ_X, est la racine carrée de la variance :

$$\sigma(X) = \sqrt{V(X)} = \sqrt{\frac{1}{N}\sum_{k=1}^{N}\left(x_k - \bar{X}\right)^2}.$$

L'écart-type s'exprime dans les mêmes unités que X : si X désigne des longueurs en m, alors $\sigma(X)$ s'exprime aussi en m.

Si la série statistique est donnée par un tableau statistique (x_i, n_i), ce qui signifie que la valeur x_i est prise n_i fois, on peut directement calculer la variance par la formule :

$$V(X) = \frac{1}{n_1 + \cdots + n_N}\sum_{i=1}^{N} n_i (x_i - \bar{X})^2.$$

Voyons ce que signifie l'écart-type. Dans une classe de 25 élèves, à un devoir, on observe les notes suivantes :

- 5 notes à 8.
- 5 notes à 9.
- 5 notes à 10.
- 5 notes à 11.
- 5 notes à 12.

L'espérance (ou moyenne des notes!) est 10. La variance vaut

$$V = \frac{1}{25}\left(5 \times 2^2 + 5 \times 1^2 + 5 \times 0^2 + 5 \times 1^2 + 5 \times 2^2\right) = 2.$$

Elle est assez faible, les notes sont donc très centrées autour de la moyenne.

Dans une autre classe de 25 élèves, au même devoir, les élèves ont obtenu :

- 5 notes à 0.
- 5 notes à 5.
- 5 notes à 10.
- 5 notes à 15.
- 5 notes à 20.

L'espérance vaut toujours 10, mais pourtant il est clair que les deux classes sont très différentes! Calculons la variance : elle vaut

$$V = \frac{1}{25}\left(5 \times 10^2 + 5 \times 5^2 + 5 \times 0^2 + 5 \times 5^2 + 5 \times 10^2\right) = 50.$$

La variance est beaucoup plus grande que dans le premier cas, ce qui signifie que les notes sont très espacées!

Corrélation des variables aléatoires

Covariance de deux séries statistiques

La **covariance** de deux séries statistiques $X = (X_i)_{i=1,\ldots,N}$ et $(Y_i)_{i=1,\ldots,N}$ ayant le même nombre d'éléments est défini par

$$\mathrm{Cov}(X,Y) = \sum_{i=1}^{N} \frac{(X_i - \bar{X})(Y_i - \bar{Y})}{N}.$$

Il s'agit donc de la moyenne des produits des écarts des valeurs à la moyenne de chaque série.

Le **coefficient de corrélation linéaire** de ces deux séries est défini par

$$\rho(X,Y) = \frac{\mathrm{Cov}(X,Y)}{\sqrt{\mathrm{Var}(X)}\sqrt{\mathrm{Var}(Y)}}$$

où $\mathrm{Var}(X)$ et $\mathrm{Var}(Y)$ désignent respectivement les variances de X et de Y.

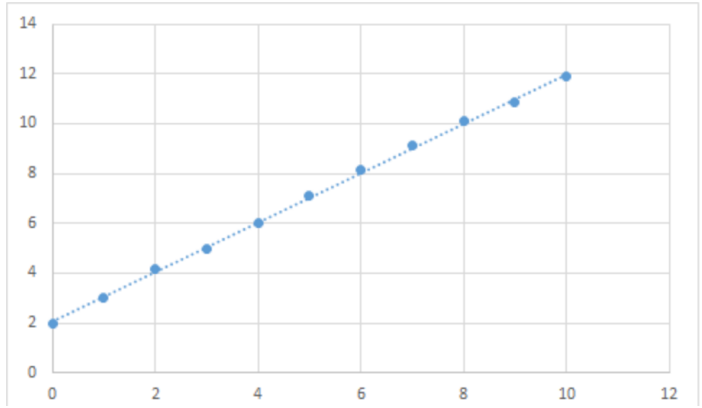
Le coefficient de relation linéaire quantifie la force du lien linéaire entre X et Y. Si $r = 1$ ou $r = -1$, alors il existe deux réels a et b tels que $Y = aX + b$: Y dépend affinement de X . Si au contraire r est proche de 0, alors X et Y ne dépendent pas affinement l'un de l'autre.

Étudions trois exemples pour voir l'influence du coefficient de corrélation linéaire.

Exemple 1 :

X	0	1	2	3	4	5	6	7	8	9	10
Y	2,04	3,02	4,15	4,97	6,02	7,08	8,12	9,14	10,13	10,86	11,89

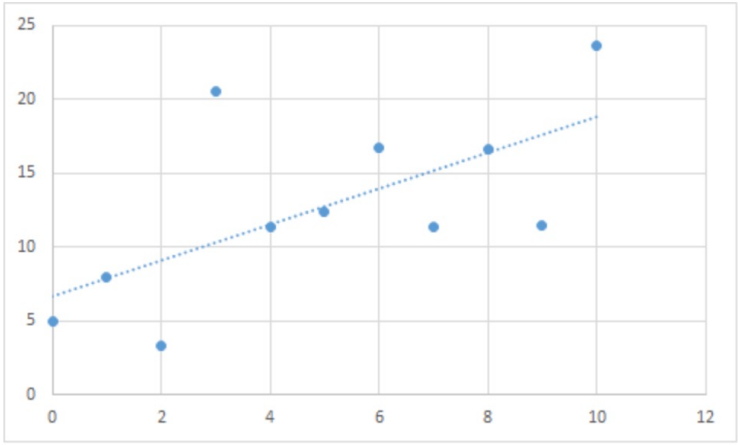
Le coefficient de corrélation linéaire est presque égal à 1. Les deux séries statistiques dépendent affinement l'une de l'autre, ce que confirme le nuage de points.



Exemple 2 :

X	0	1	2	3	4	5	6	7	8	9	10
Y	5,01	7,93	3,27	20,50	11,33	12,41	16,74	11,39	16,61	11,48	23,59

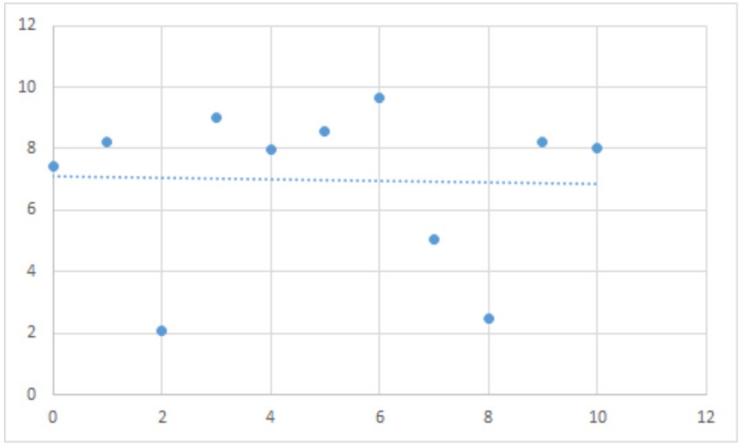
Le coefficient de corrélation linéaire est environ égal à $0{,}65$. Il y a une certaine corrélation entre les deux séries statistiques mais elles ne dépendent pas affinement l'une de l'autre. Voici le nuage de points associé.



Exemple 3 :

X	0	1	2	3	4	5	6	7	8	9	10
Y	7,45	8,20	2,09	9,03	7,99	8,58	9,63	5,06	2,49	8,22	8,01

Le coefficient de corrélation linéaire est environ égal à $-0{,}02$, il est donc très faible. Les deux caractères étudiés ne semblent pas corrélés. Voici le nuage de points associé.



Si X et Y admettent une covariance et des variances non nulles, alors leur **coefficient de corrélation linéaire** est défini par :

$$\rho(X,Y) = \frac{\mathrm{Cov}(X,Y)}{\sqrt{V(X)V(Y)}} = \frac{\mathrm{Cov}(X,Y)}{\sigma(X)\sigma(Y)} \in [-1,1].$$

Les variables aléatoires X et Y sont dites **non corrélées** si $\mathrm{Cov}(X,Y) = 0$. Elles sont dites **corrélées** dans le cas contraire.

Interprétation : Le coefficient de corrélation linéaire mesure la dépendance affine de X et Y. Ainsi, si $\rho(X,Y) = 1$, il existe des réels a et b tels que $Y = aX + b$. Inversement, si X et Y sont indépendantes, $\mathrm{Cov}(X,Y) = 0$.

Si $(X_i)_{i=1,\ldots,n}$ est une suite finie de variables aléatoires, la **matrice des variances/covariances** des (X_i) est la matrice carrée dont le coefficient en (i,j) est donné par :

$$c_{i,j} = \mathrm{Cov}(X_i, X_j).$$

Par exemple, la matrice de covariance du couple (X,Y) est :

$$\begin{pmatrix} V(X) & \mathrm{Cov}(X,Y) \\ \mathrm{Cov}(X,Y) & V(Y) \end{pmatrix}.$$

Une matrice de covariance est toujours symétrique.