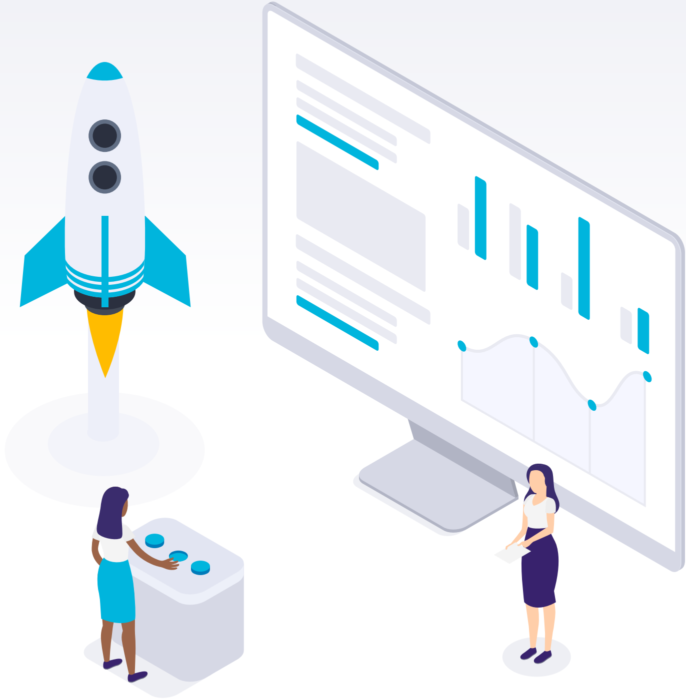
Deux applications potentielles de l'intelligence artificielle en médecine

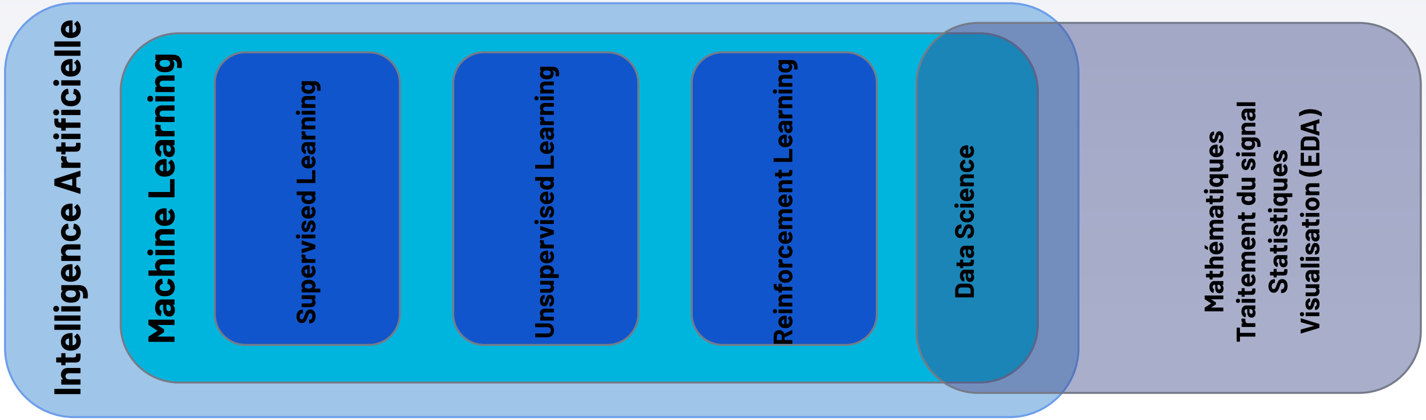
Plan de la présentation

1. Introduction
2. ML vs DL
3. Maladie de Crohn, Rectocolite hémorragique
4. Classification des signaux EEG
5. Décomposition des signaux
6. Ressources
7. Conclusion

1 Introduction

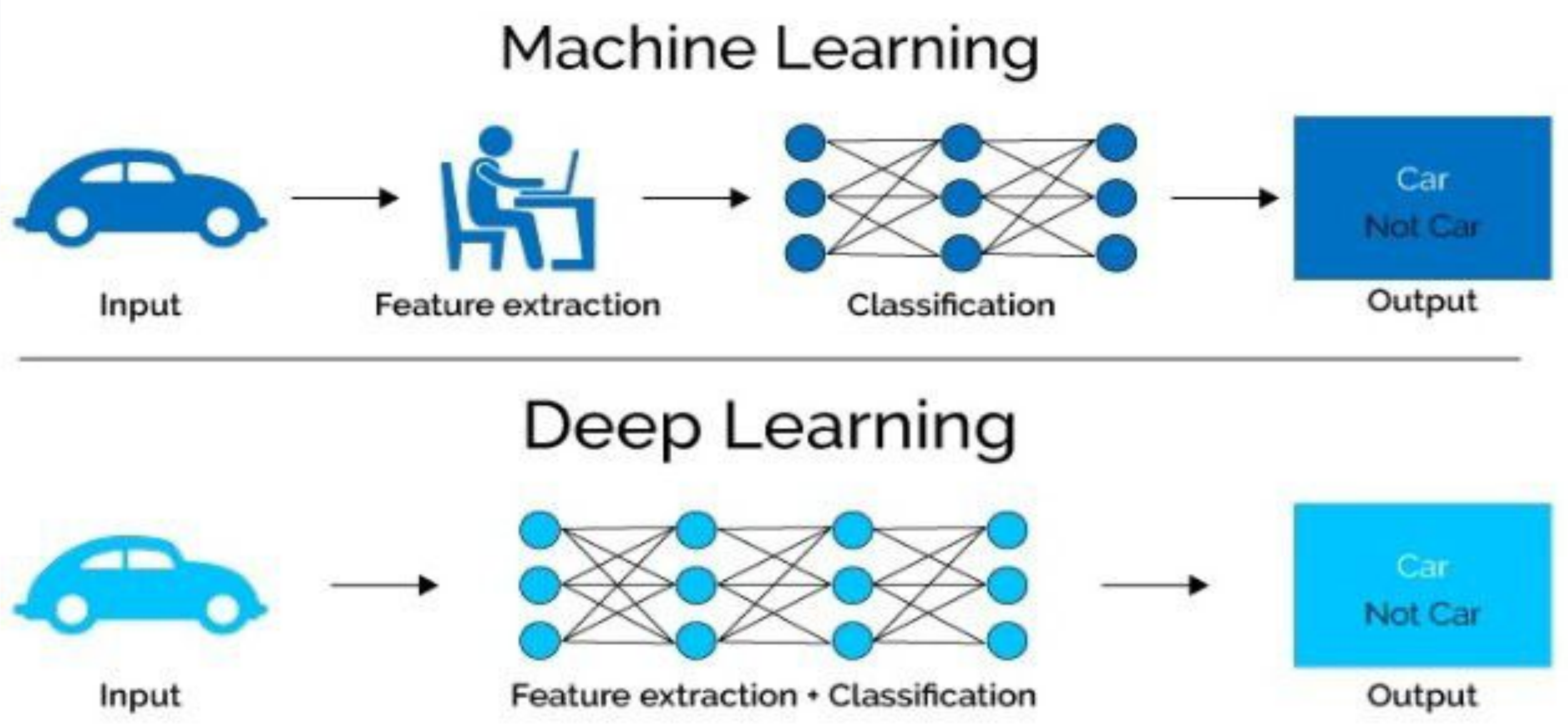


AI, ML, DL, RL, DS… Quelques définitions



- **L'intelligence artificielle** – permet aux machines d'effectuer des tâches sans programmation préalable.
- **Le machine learning** – est basé sur l'apprentissage statistique paramétrique ou non-paramétrique à partir des données ainsi il permet de réaliser des prédictions.
- **Le deep learning** – est une branche du machine learning utilisant des réseaux de neurones.
- **Le reinforcement learning** – est la seule branche où l'IA pourrait dépasser celle de l'humain.

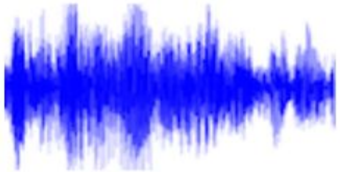
ML vs DL


Machine Learning
Car
Not Car
Input
Feature extraction
Classification
Output
Deep Learning
Car
Not Car
Input
Feature extraction + Classification
Output

Vers les grandes dimensions

Données diverses mais avec un grand nombre d de variables

$d \sim 10^6/mn$



Audio

$d \sim 10^6$

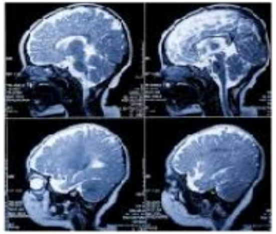


Image 2D/3D

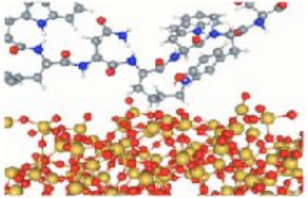
$d \sim 10^6$

Texte

$d \sim 10^9$

Reseau social

$d \sim 10^{24}$



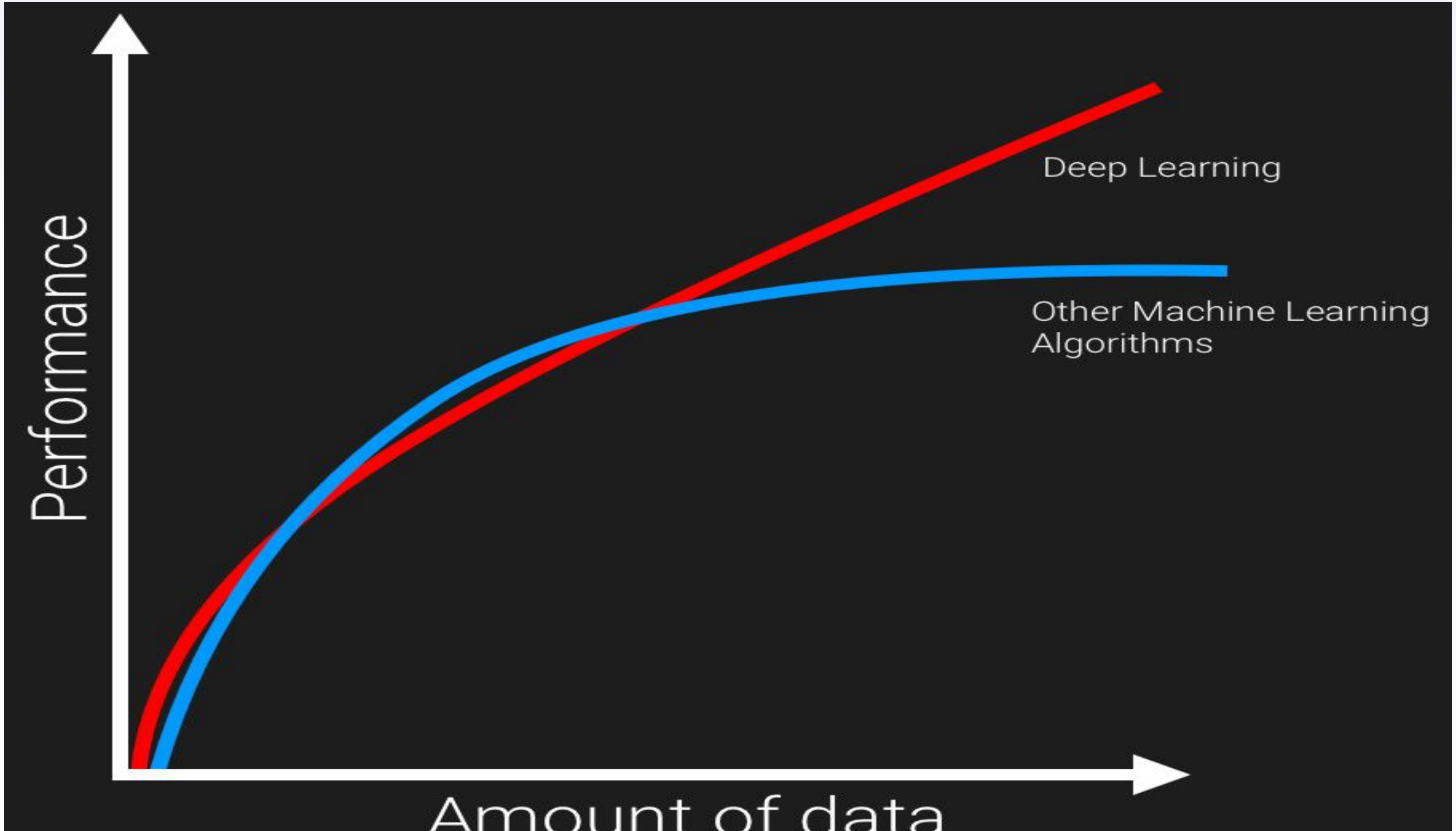
mole de matière

- *Modeliser*: capturer la nature et la variabilité des données
  - compression, restauration, synthèse

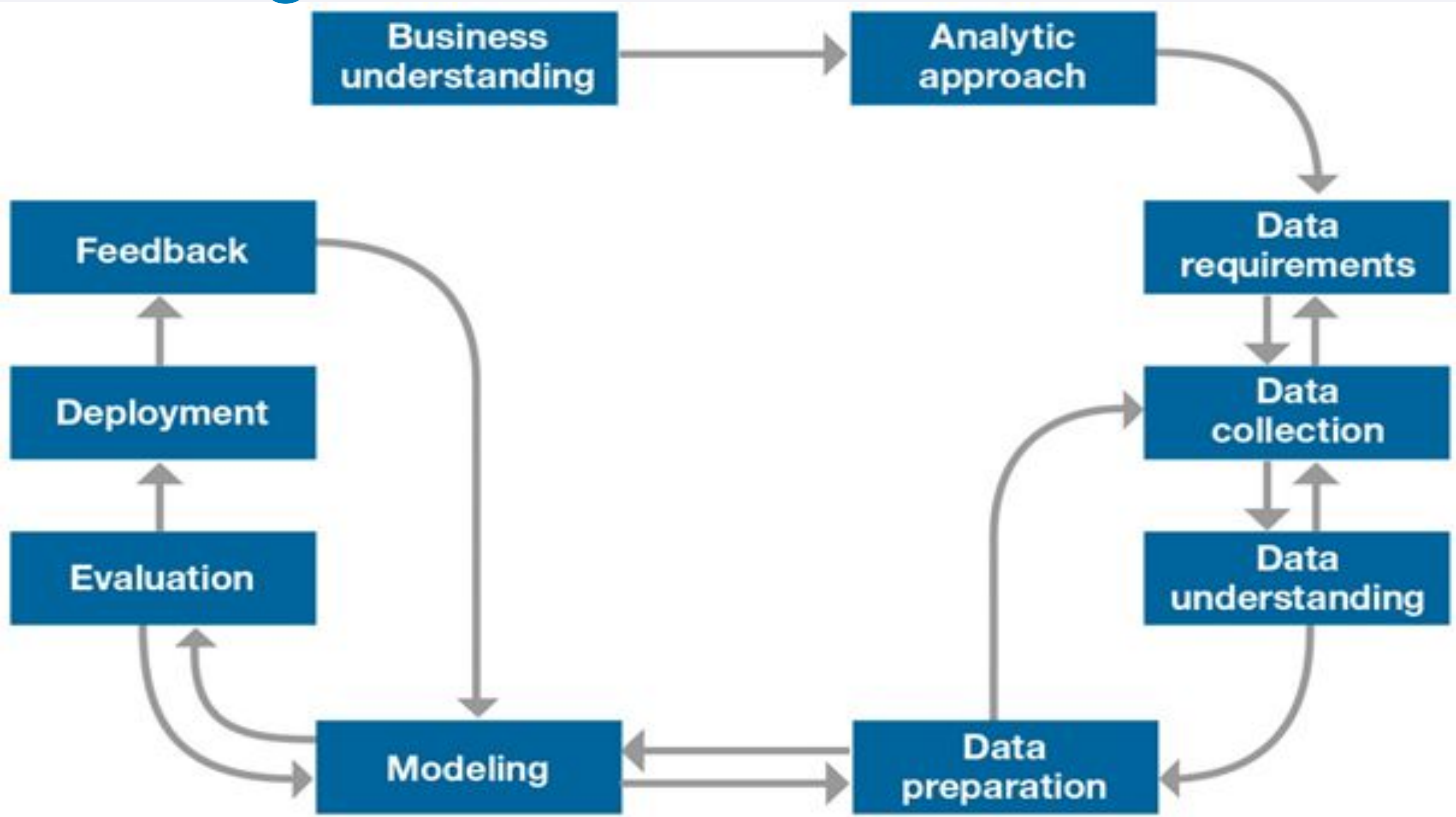
- *Prédire:* estimer la réponse d'une question à partir de données
  - Reconnaissance d'images ou de sons, diagnostics médicaux, analyse de textes ou traductions, prédire la physique...

Apprentissage statistique $\longrightarrow$ Intelligence artificielle

ML vs DL

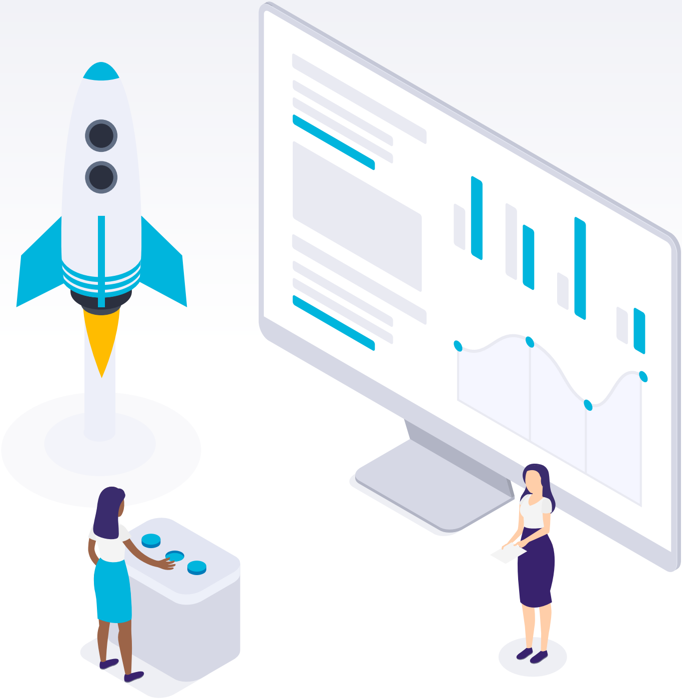

Performance
Deep Learning
Other Machine Learning Algorithms
Amount of data

Méthodologie



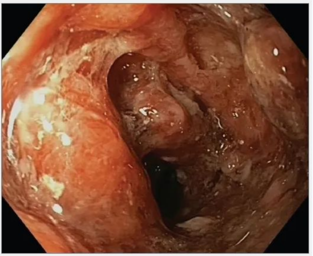
2

Application 1: Maladie de Crohn, Rectocolite hémorragique



التهاب الأمعاء الناحي , التهاب القولون التقرحي

Maladie de Crohn, Rectocolite hémorragique



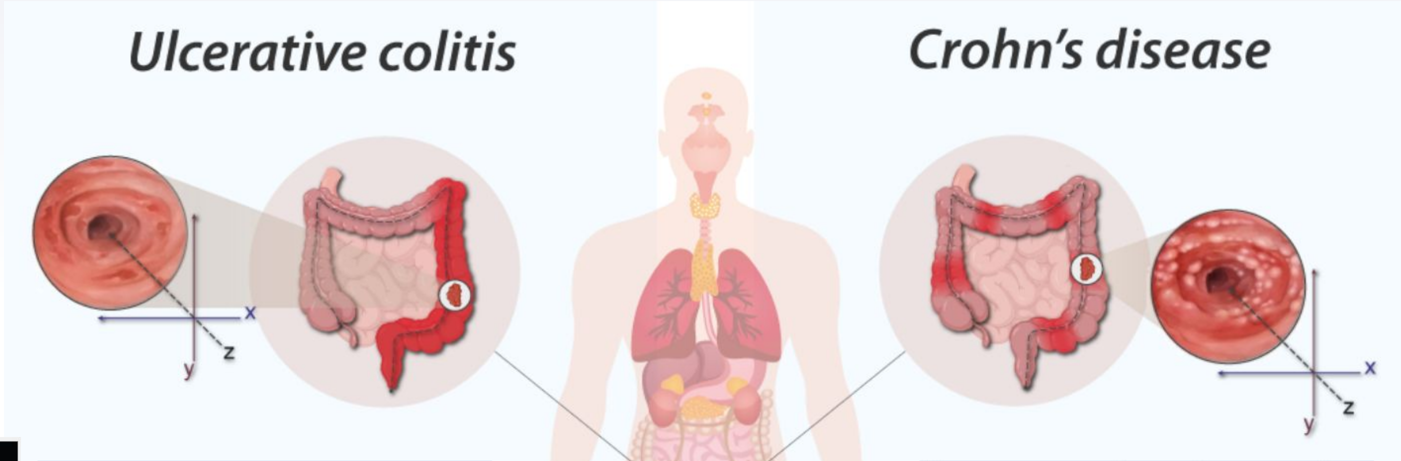
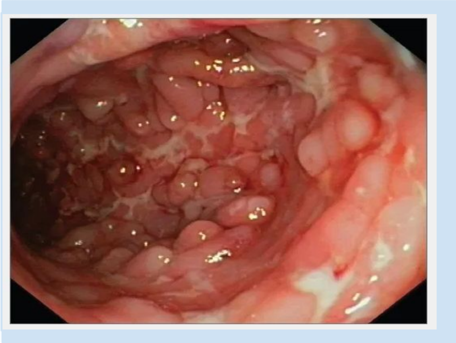
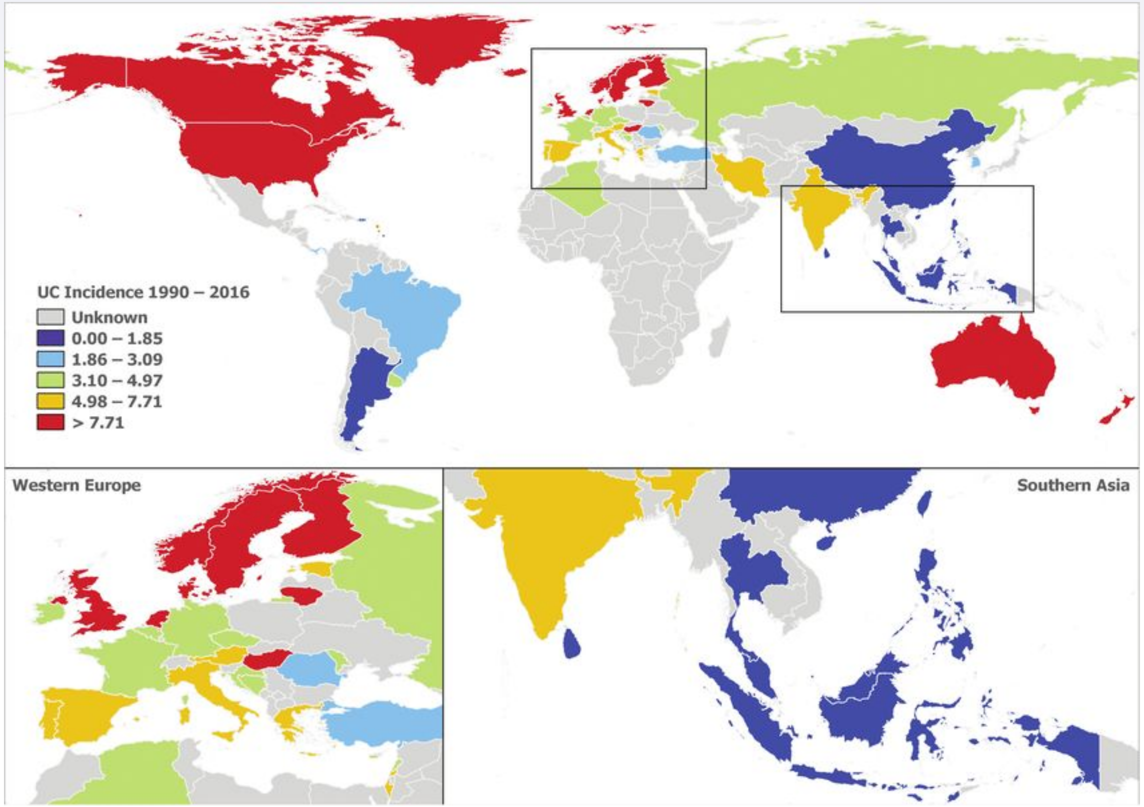


Illustration réalisée par Radhi Toujani et Louay Boukhris



Carte mondiale: absence de statistique en Tunisie!



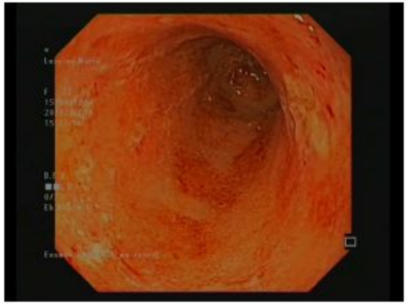
Collaboration avec Professeur **Saber Rebii**, service de chirurgie générale, centre de traumatologie et des grands brûlés, Ben Arous.

*Collecte des données
*Anonymisation de données
*Analyse sur SPSS avec des modèles mathématiques simples...

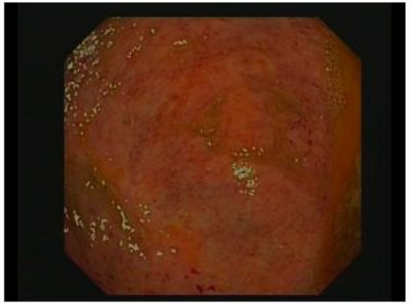
Endoscopic abnormalities that are signs of UC disease found during a colonoscopy video

(a) Ulceration

(b) Bleeding



(c) Erosion



(d) Fibrosis

Collecte des données

Indicateurs cliniques, indicateurs biologique et chimique, Données images, Données vidéos

For both diseases, doctors usually use a set of biological, clinical and spatial indicators, to evaluate the patient's conditions and its probable evolution and to choose the most appropriate treatment and evaluate the response to the proposed one.
As clinical indicators one can cite:

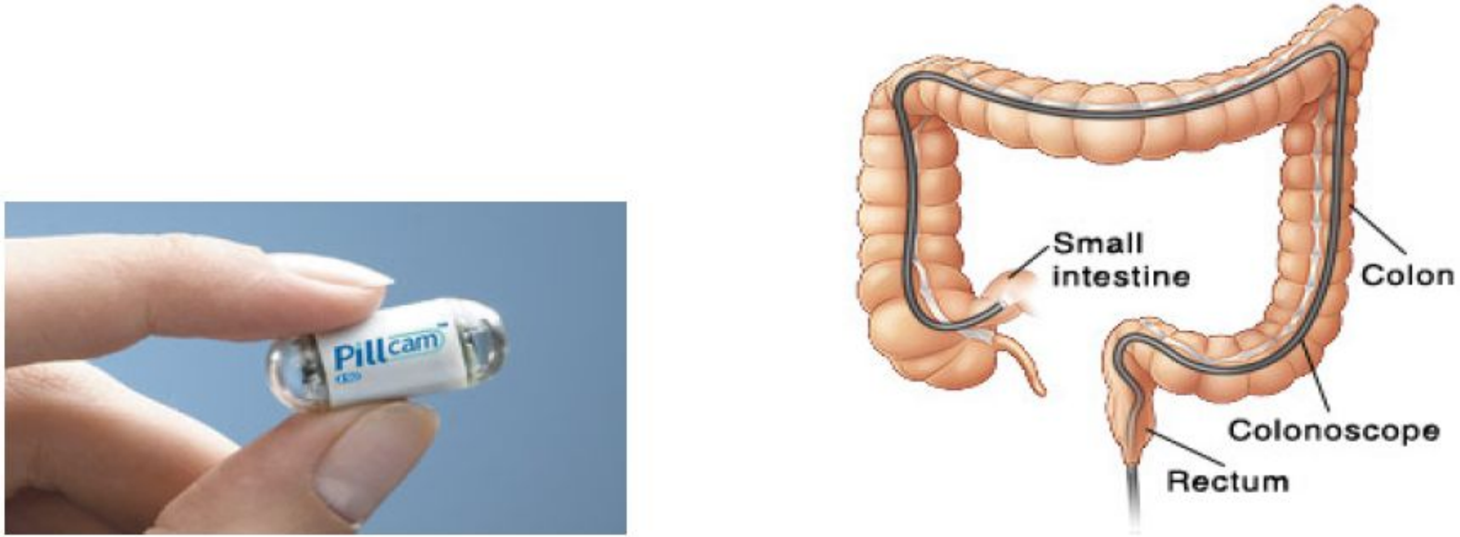
- Doctors questioning and examination of the patient,
- Colonoscopy video examination,
- Histological images corresponding to the biopsies taken during the colonoscopy examination.

The biological (chemical) indicators are:

- Dosage of calprotectin in the C-reactive protein stools which are inflammation markers,
- DNA and RNA analyzes of the intestinal microbiota,
- Analyzes of RNA expressed in the intestine.

These indicators are not spatial and the given information is never localized in a precise position then doctors and surgeons are prevented from having spatial information on anatomical locations.
The diagnosis of these diseases is based on the analysis of colonoscopy videos. Thus, physicians assess the severity of the disease according to the presence of inflammation, bleeding or ulcers on the intestinal wall. In the same way, the extent of the lesions is currently ignored in medical practice, for lack of a validated method for analyzing this information.

Collecte des données vidéos et images



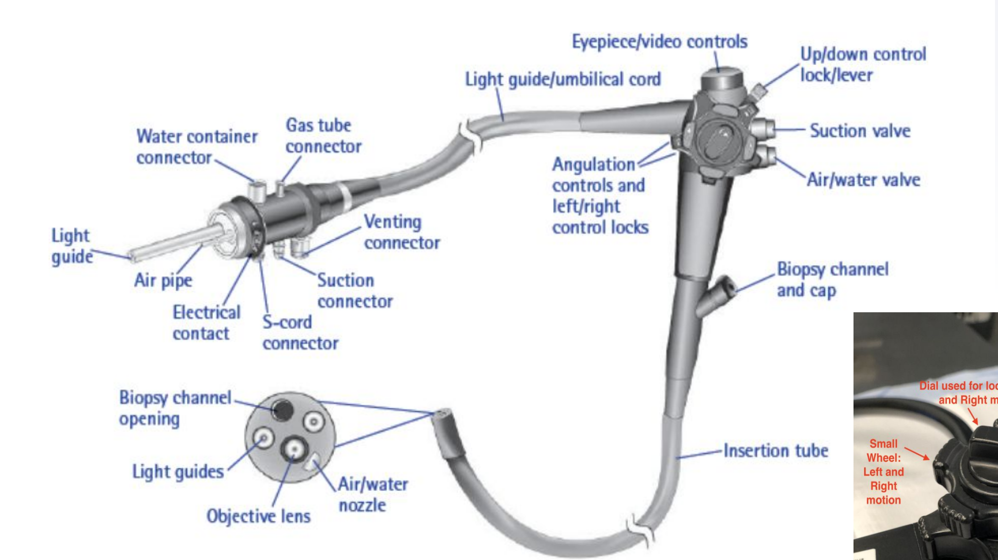
(a) Wireless Capsule Endoscopy device (b) Colonoscope

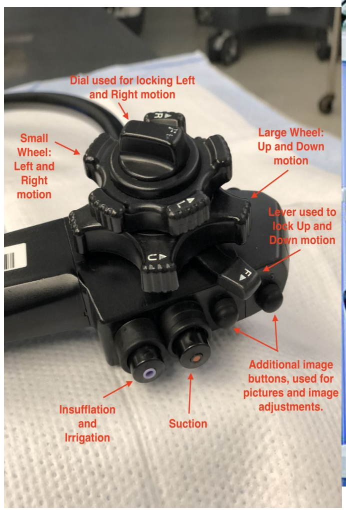
Figure 3.2: Medical devices used during the exploration of the digestive tract.

Medical device: Wireless Capsule Endoscopy

1. Optical Dome used for liquid filtration and gastrointestinal enzymes balancing,
2. Lens Holder which tightly holds the lens,
3. Lens,
4. light-emitting diode (LED) responsible for the light illumination around the passage area of the body for easier identification of the affected tissues,
5. Complementary Metal Oxide Semiconductor (CMOS) Sensor able to detect
tiny objects with 140-degree accuracy and very high-quality images,
6. Battery made up of silver oxid, made to be unharmful to the body and it is used to feed the CMOS detector, LED and transmitter. And it can work for about 8 hours,
7. Application Specific Integrated Circuit (ASIC Transmitter),
8. Antenna composed of coated polyethene, it ensures the information/images
communication between the belt receiver and the capsule.

Medical instrument: endoscope





Medical instrument: endoscope; biopsy forceps to collect biopsies

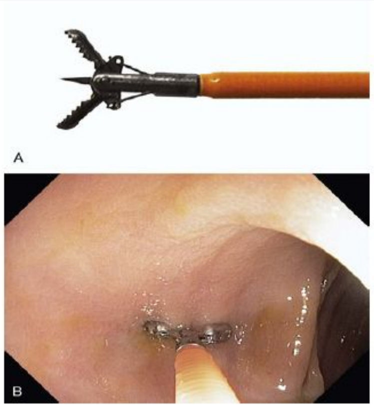


Figure 3.14: A. Biopsy forceps, B. Application of the forceps to collect biopsies during the colonoscopy

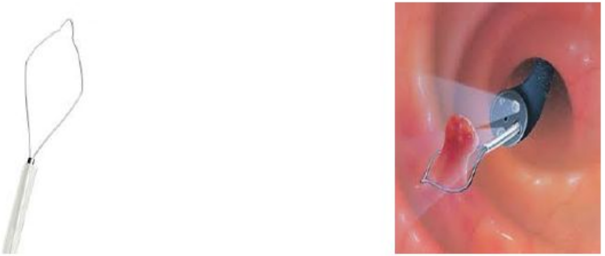
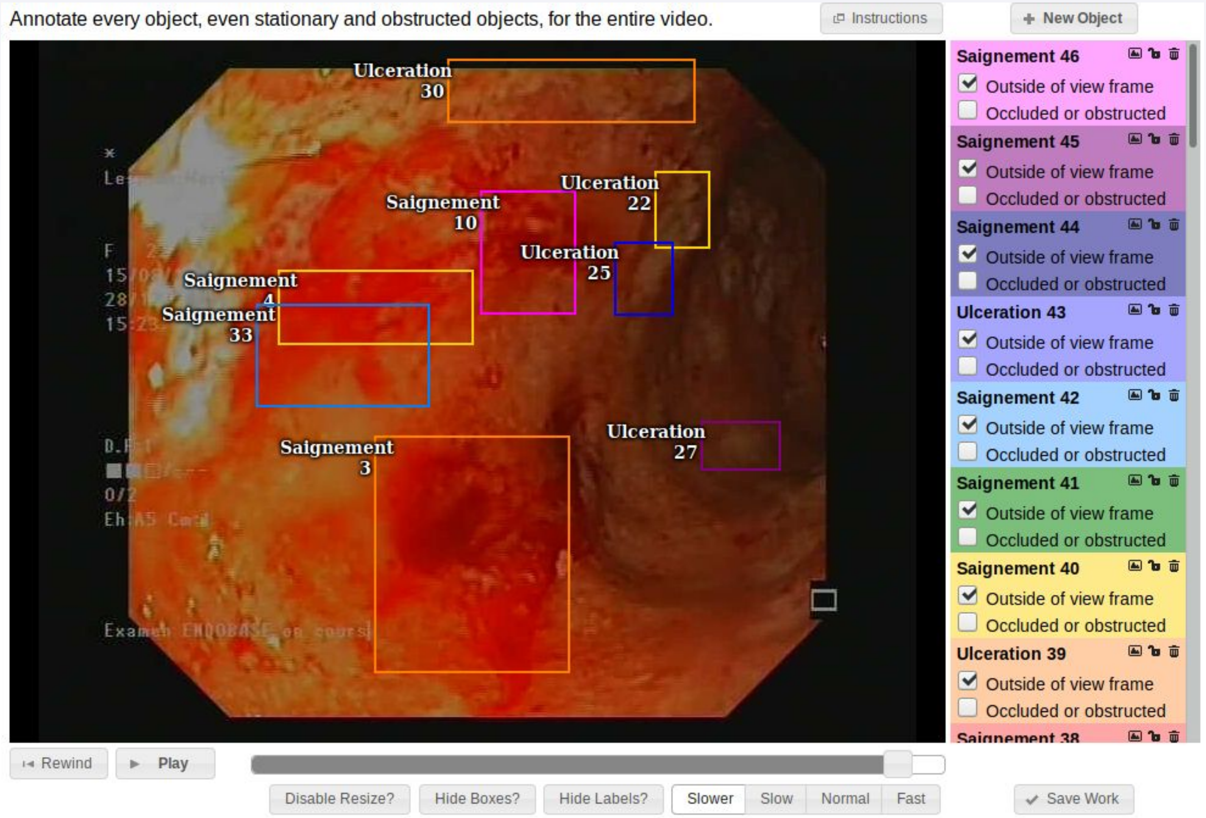
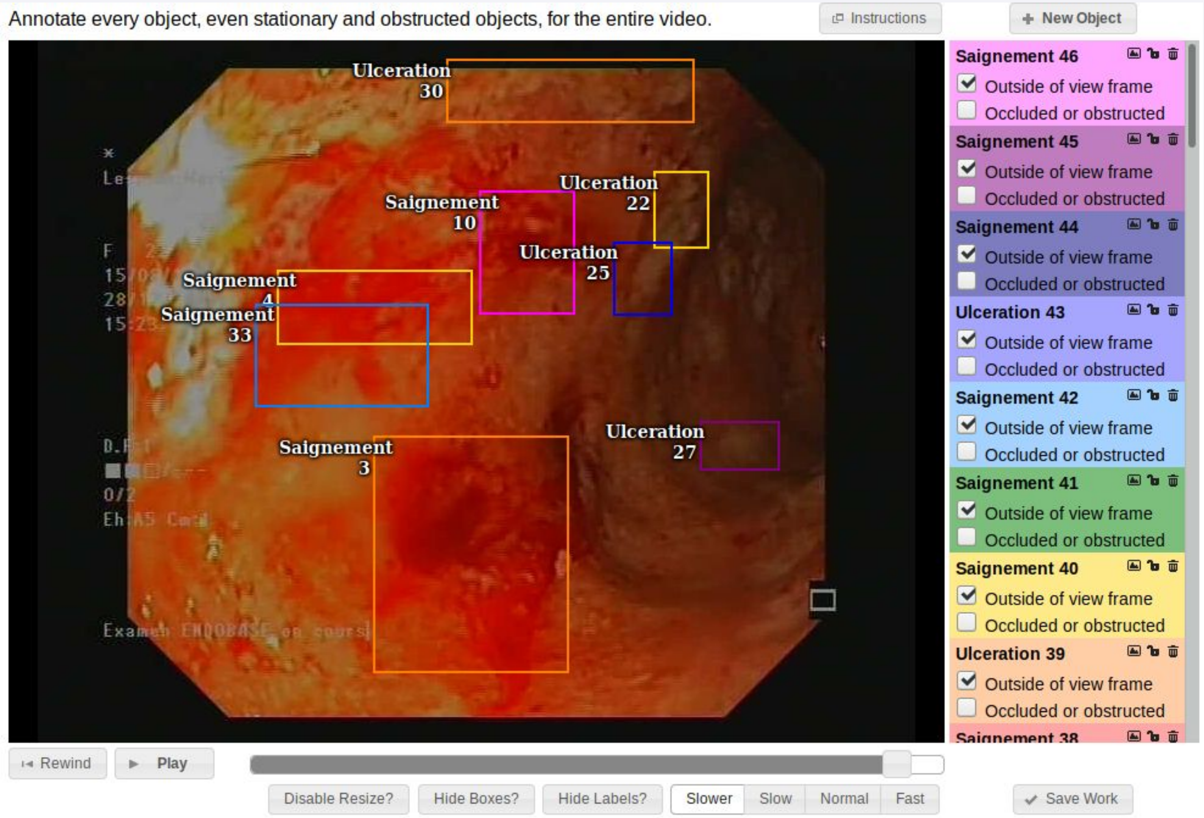


Figure 3.15: Snare forceps (left) and it is application during the colonoscopy (right)

Intelligence artificielle pour l'aide à la décision en temps réel



Intelligence artificielle pour l'aide à la décision en temps réel



Deep Learning avec CNN (Convolutional Neural Network)

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

CNN

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

Covolution Layer

Kernel

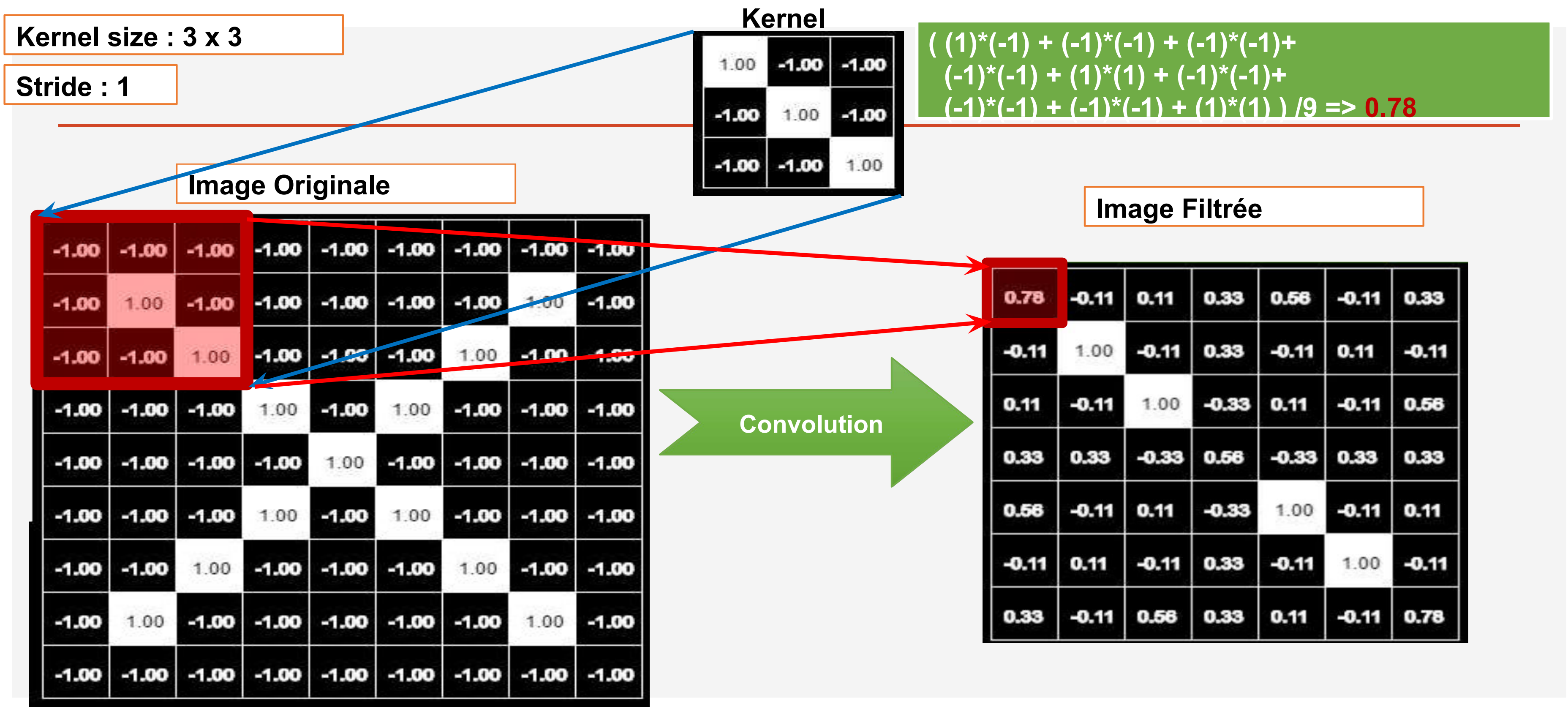
1.00	-1.00	-1.00
-1.00	1.00	-1.00
-1.00	-1.00	1.00

Kernel size : 3 x 3

Stride : 1

Image Originale

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


Kernel size : 3 x 3
Stride : 1
Kernel
((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

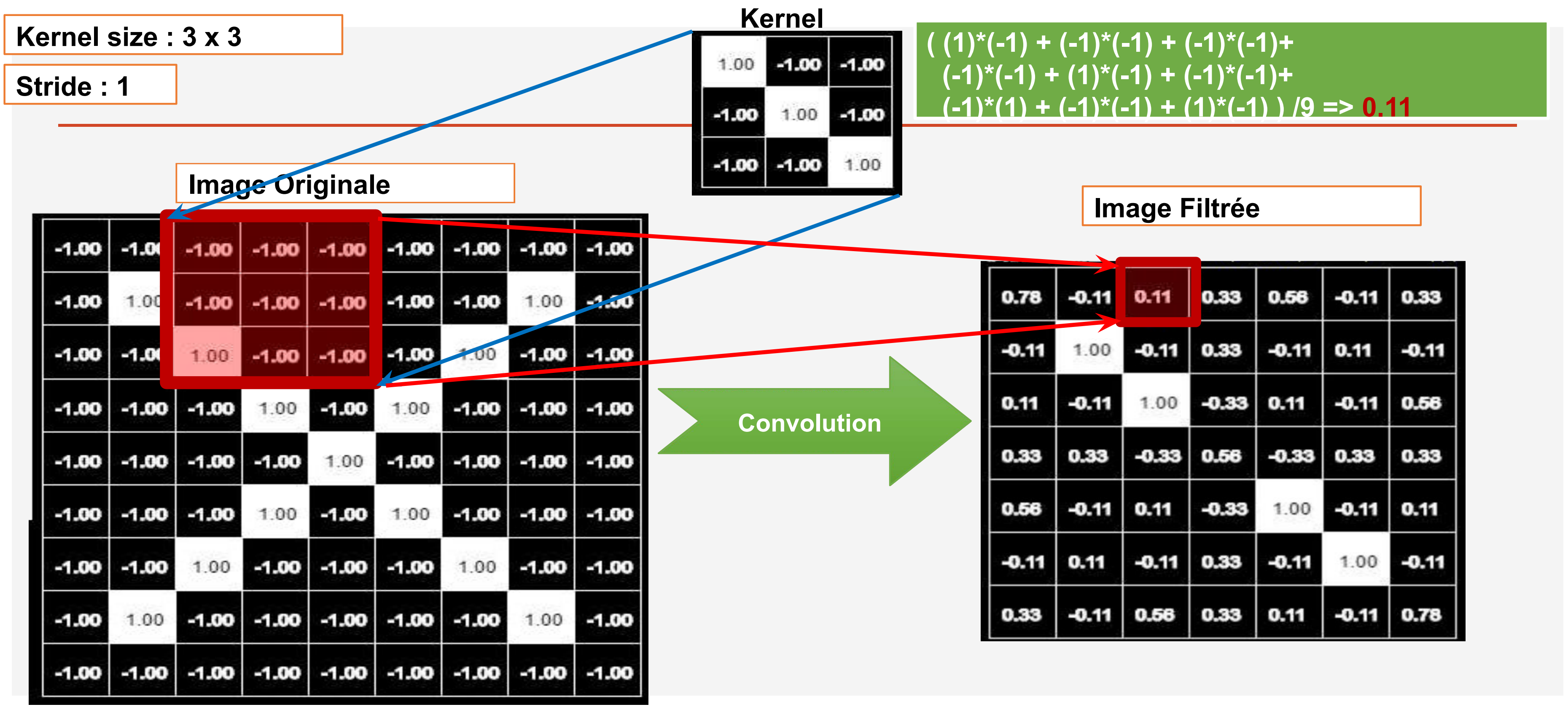
Image Originale

-1.0	-1.00	-1.00	-1.00	-1.00	-1.00	-1.00	-1.00	-1.00
-1.0	1.00	-1.00	-1.00	-1.00	-1.00	-1.00	1.00	-1.00
-1.0	-1.00	1.00	-1.00	-1.00	-1.00	1.00	-1.00	-1.00
-1.00	-1.00	-1.00	1.00	-1.00	1.00	-1.00	-1.00	-1.00
-1.00	-1.00	-1.00	-1.00	1.00	-1.00	-1.00	-1.00	-1.00
-1.00	-1.00	-1.00	1.00	-1.00	1.00	-1.00	-1.00	-1.00
-1.00	-1.00	1.00	-1.00	-1.00	-1.00	1.00	-1.00	-1.00
-1.00	1.00	-1.00	-1.00	-1.00	-1.00	-1.00	1.00	-1.00
-1.00	-1.00	-1.00	-1.00	-1.00	-1.00	-1.00	-1.00	-1.00

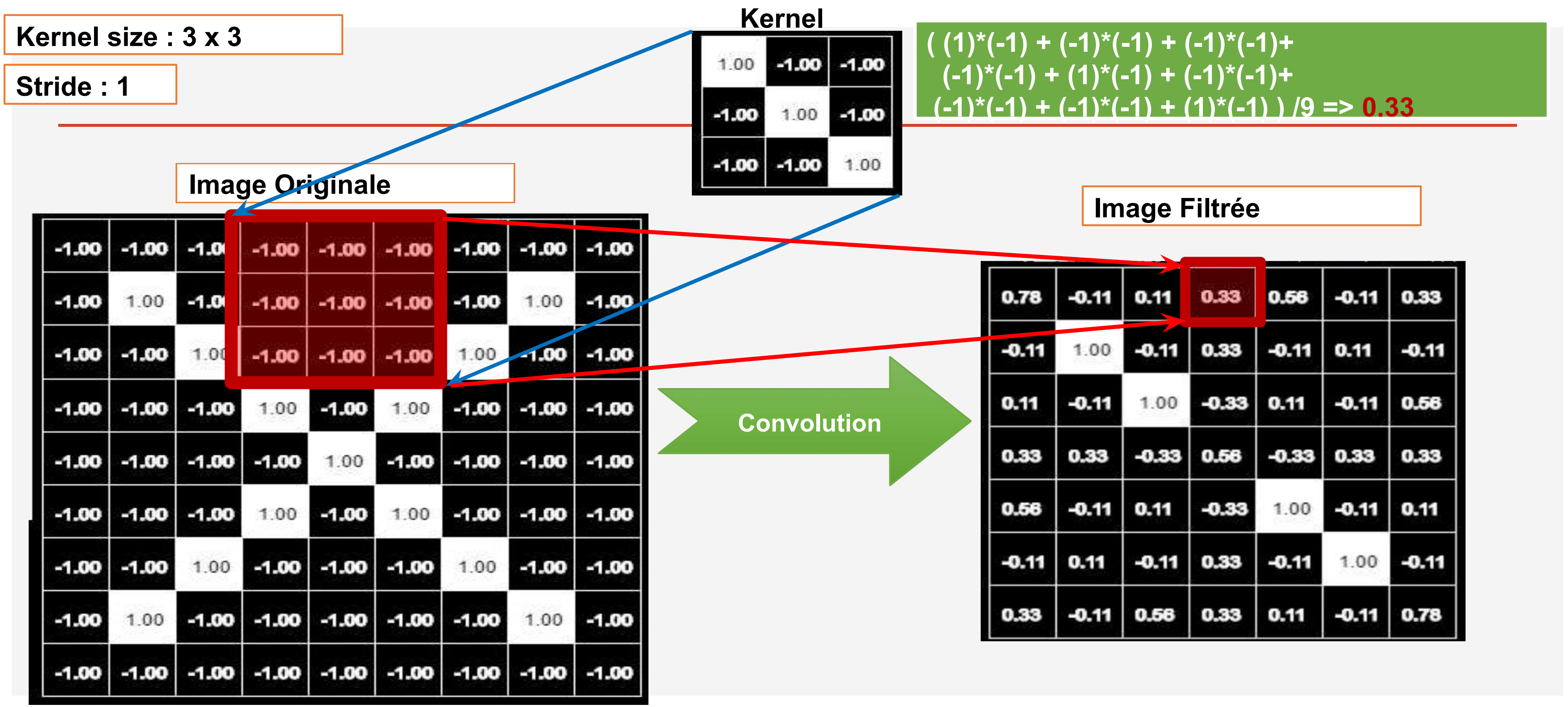
Convolution

Image Filtrée

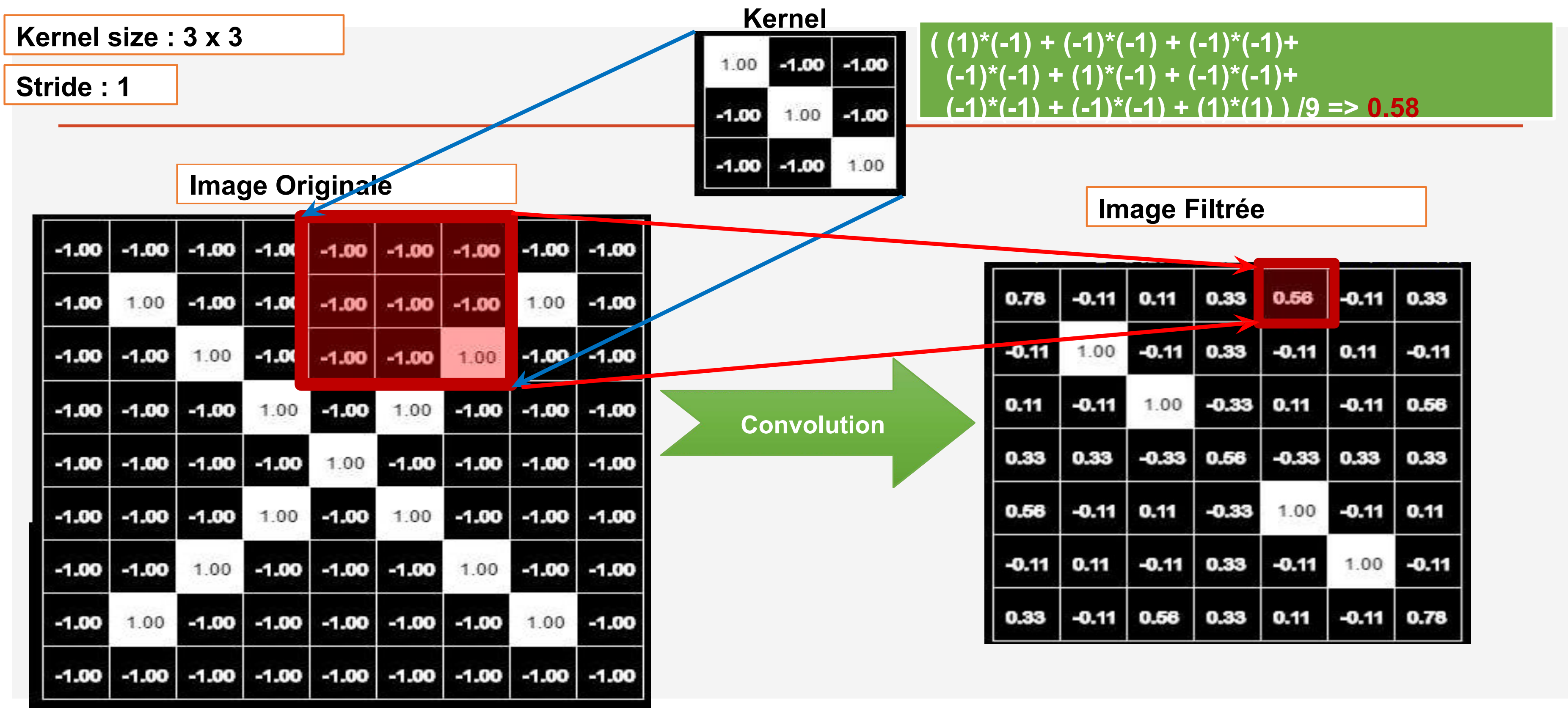
0.78	-0.11	0.11	0.33	0.56	-0.11	0.33
-0.11	1.00	-0.11	0.33	-0.11	0.11	-0.11
0.11	-0.11	1.00	-0.33	0.11	-0.11	0.56
0.33	0.33	-0.33	0.56	-0.33	0.33	0.33
0.56	-0.11	0.11	-0.33	1.00	-0.11	0.11
-0.11	0.11	-0.11	0.33	-0.11	1.00	-0.11
0.33	-0.11	0.56	0.33	0.11	-0.11	0.78



Kernel size : 3 x 3
Stride : 1
Kernel
((1)*(-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 => 0.33
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)*(-1) + (-1)*(-1) + (-1)*(-1)+
(-1)*(-1) + (1)*(-1) + (-1)*(-1)+
(-1)*(-1) + (-1)*(-1) + (1)*(1)) /9 => 0.58
Image Originale
Convolution
Image Filtrée

Kernel size : 3 x 3

Stride : 1

Kernel

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

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

Image Originale

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

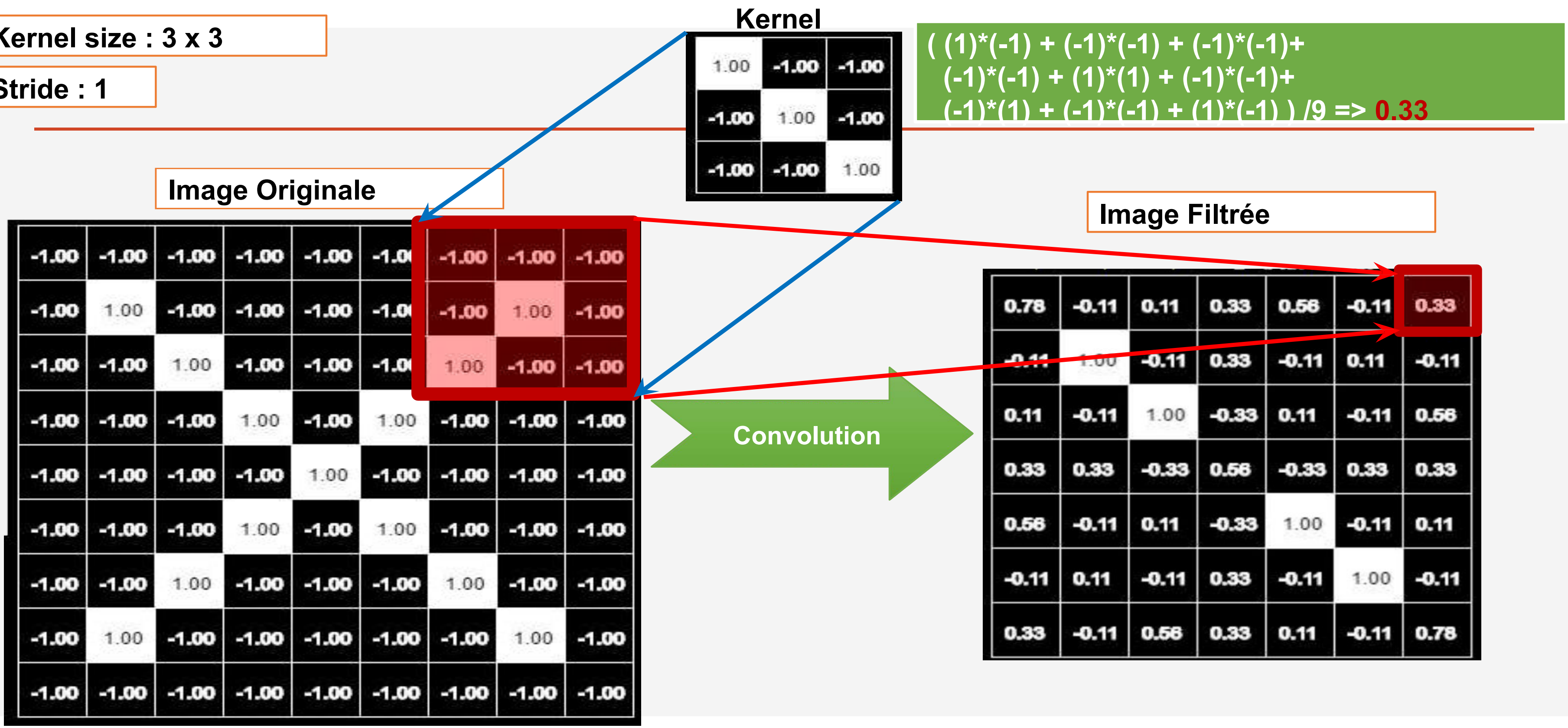
Convolution

Image Filtrée

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

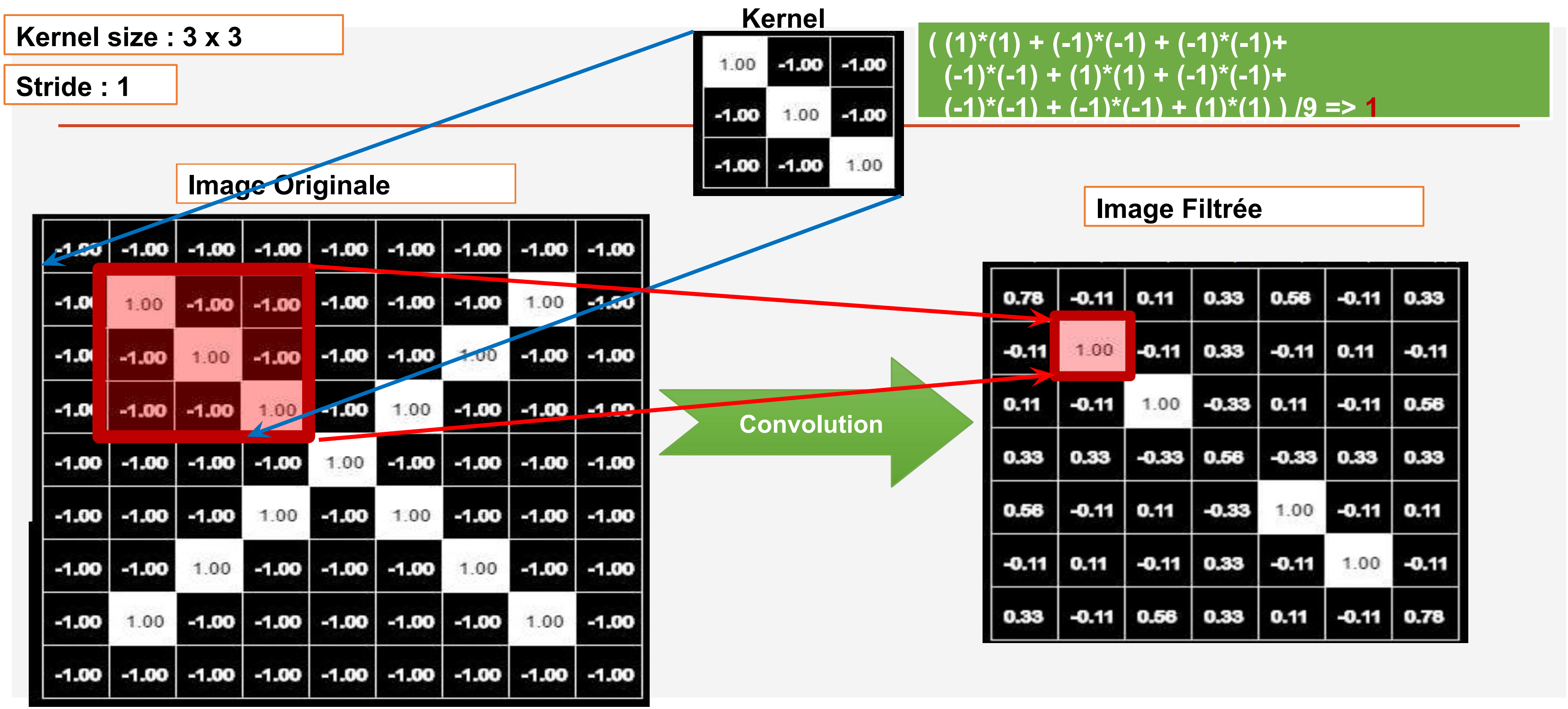
Image Originale

-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

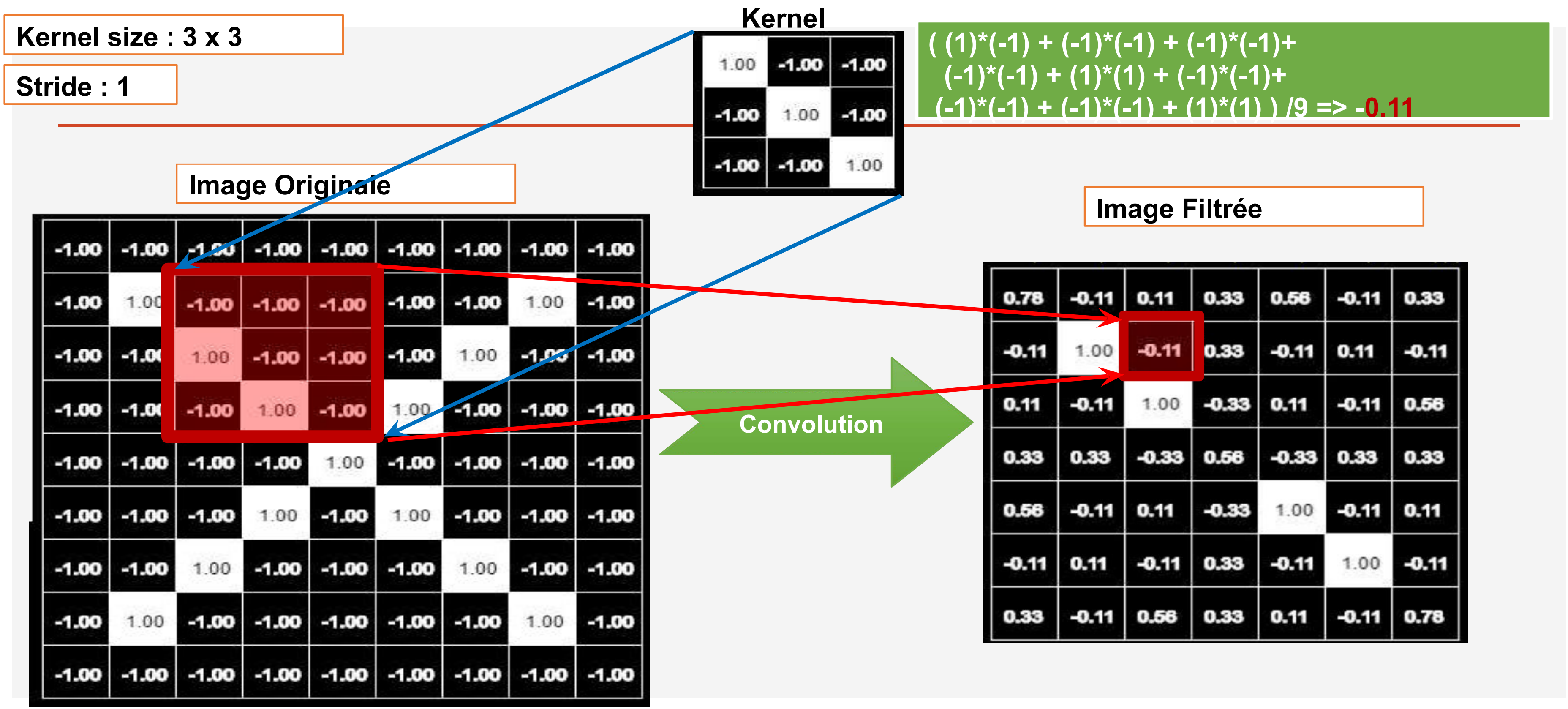
Convolution

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)*(1) + (-1)*(-1) + (-1)*(-1)+
(-1)*(-1) + (1)*(1) + (-1)*(-1)+
(-1)*(-1) + (-1)*(-1) + (1)*(1)) /9 => 1
Image Originale
Image Filtrée
Convolution



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

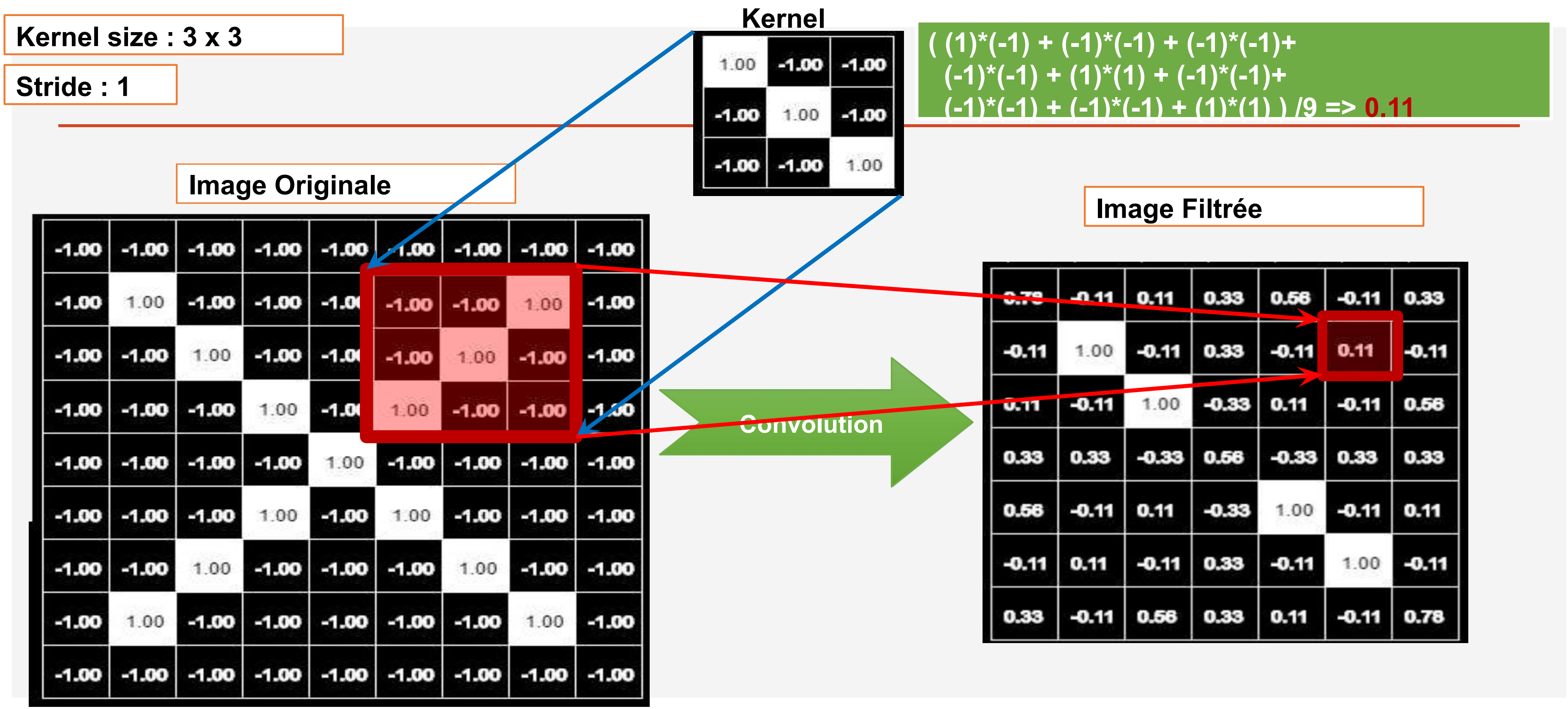
Image Originale

-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)*(-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 => -0.11

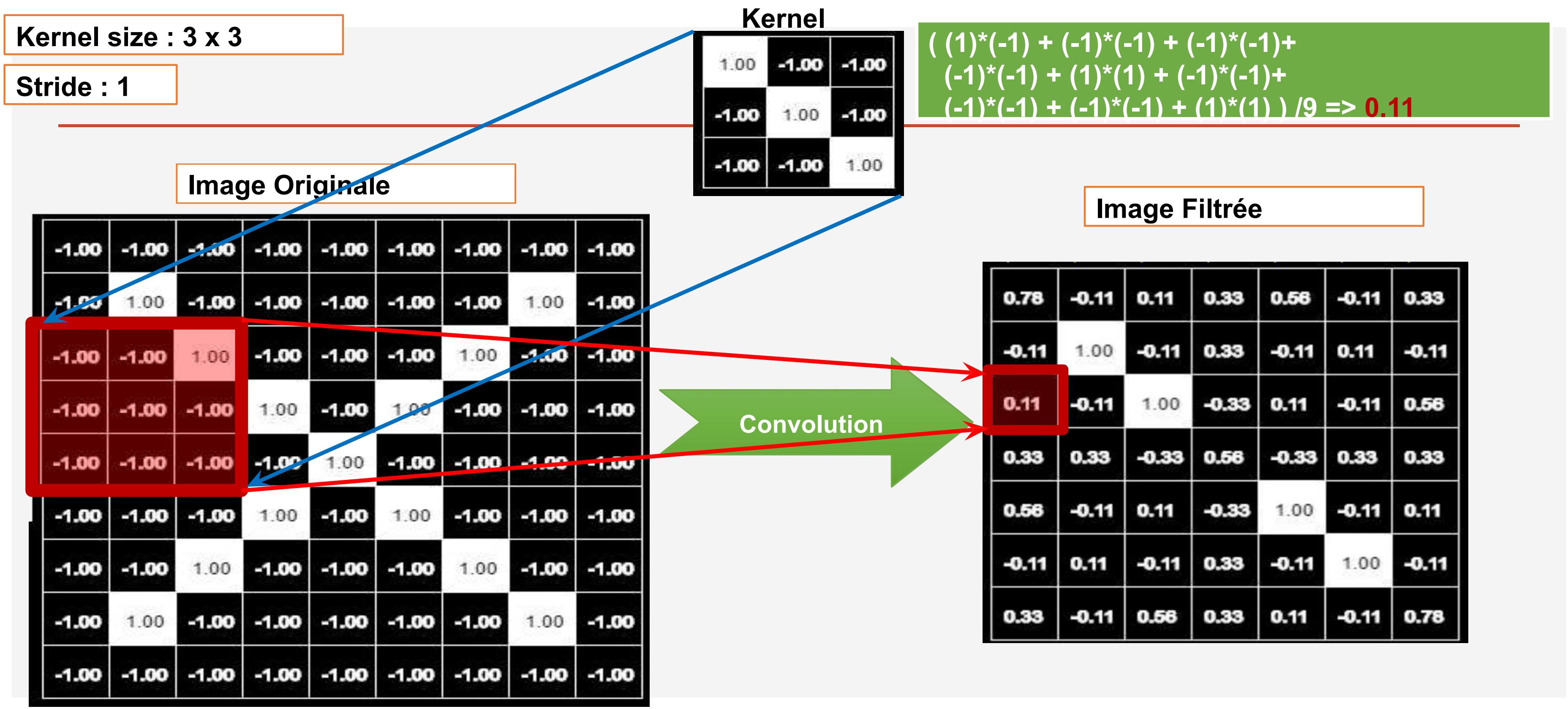
Image Originale

-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

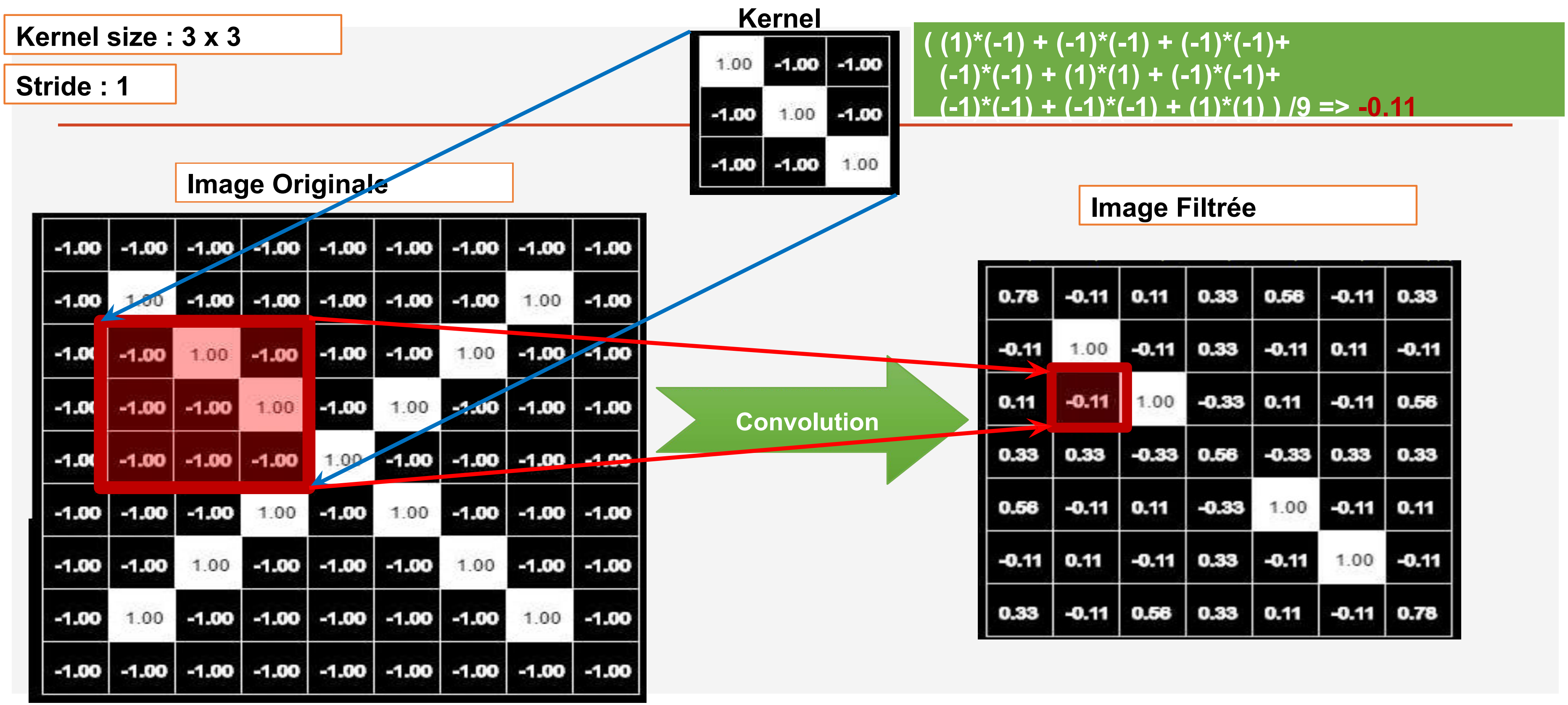
Convolution

Image Filtrée

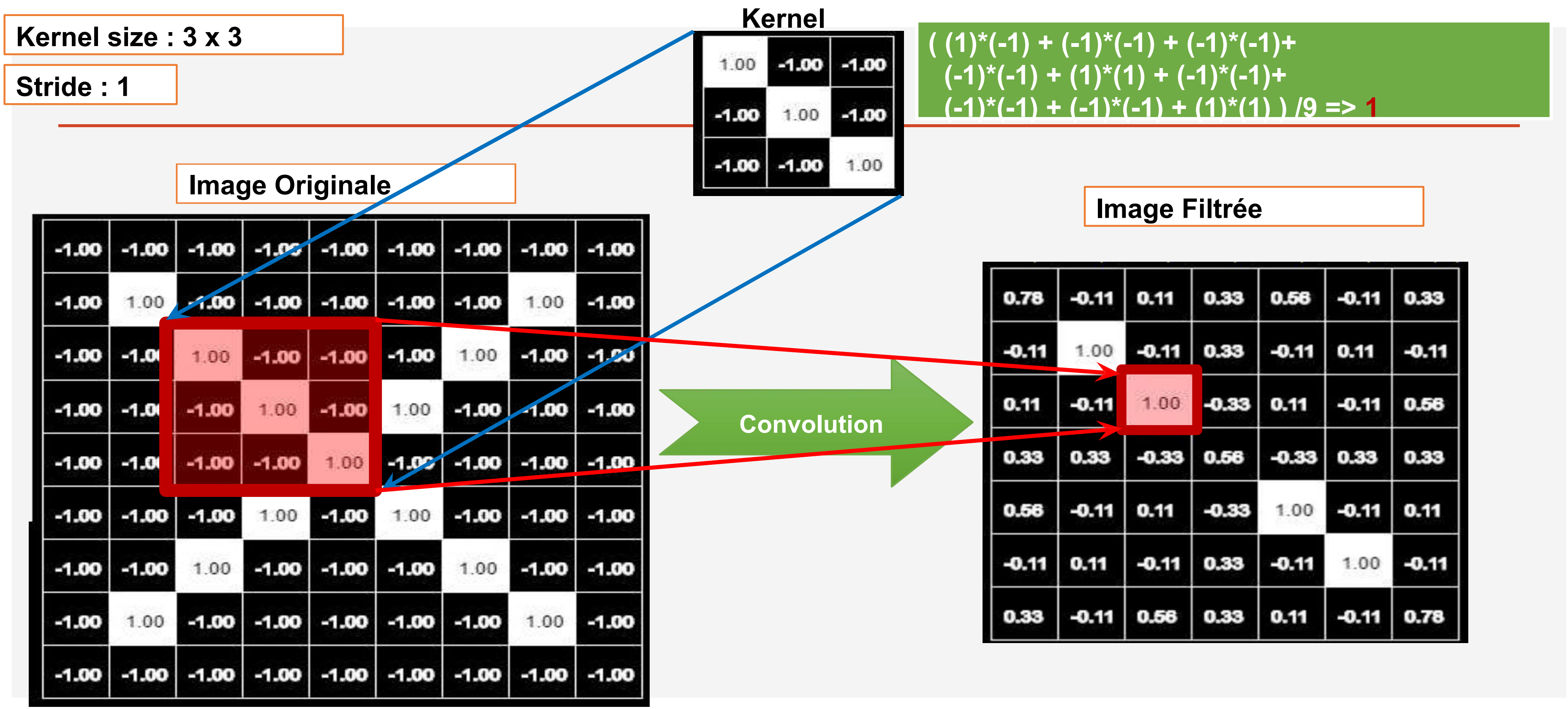
0.78	-0.11	0.11	0.33	0.56	-0.11	0.33
-0.11	1.00	-0.11	0.33	-0.11	0.11	-0.11
0.11	-0.11	1.00	-0.33	0.11	-0.11	0.56
0.33	0.33	-0.33	0.56	-0.33	0.33	0.33
0.56	-0.11	0.11	-0.33	1.00	-0.11	0.11
-0.11	0.11	-0.11	0.33	-0.11	1.00	-0.11
0.33	-0.11	0.56	0.33	0.11	-0.11	0.78

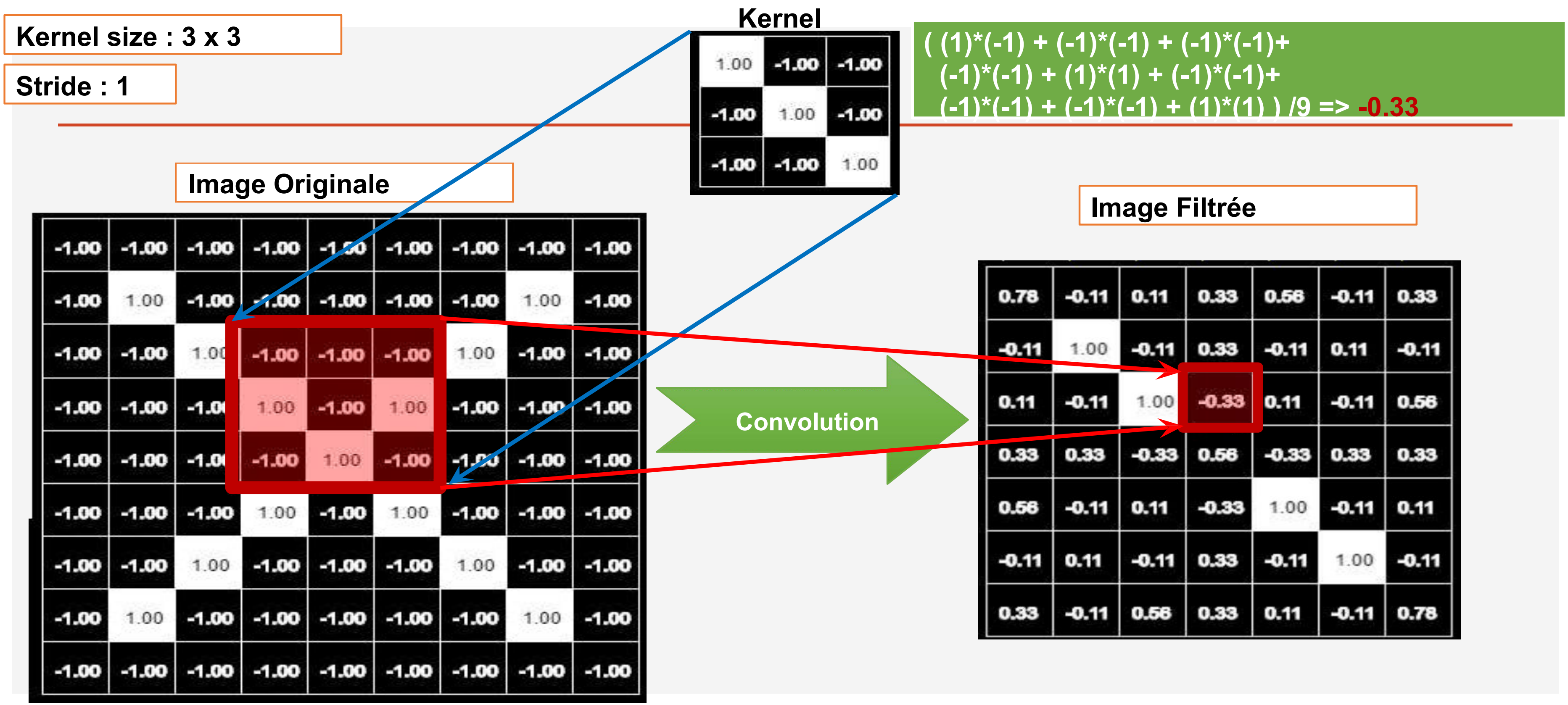


Kernel size : 3 x 3
Stride : 1
Kernel
1.00 -1.00 -1.00
-1.00 1.00 -1.00
-1.00 -1.00 1.00
((1)*(-1) + (-1)*(-1) + (-1)*(-1)+
(-1)*(-1) + (1)*(1) + (-1)*(-1)+
(-1)*(-1) + (-1)*(-1) + (1)*(1)) /9 => 0.11
Image Originale
-1.00 -1.00 -1.00 -1.00 -1.00 -1.00 -1.00 -1.00 -1.00
-1.00 1.00 -1.00 -1.00 -1.00 -1.00 -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



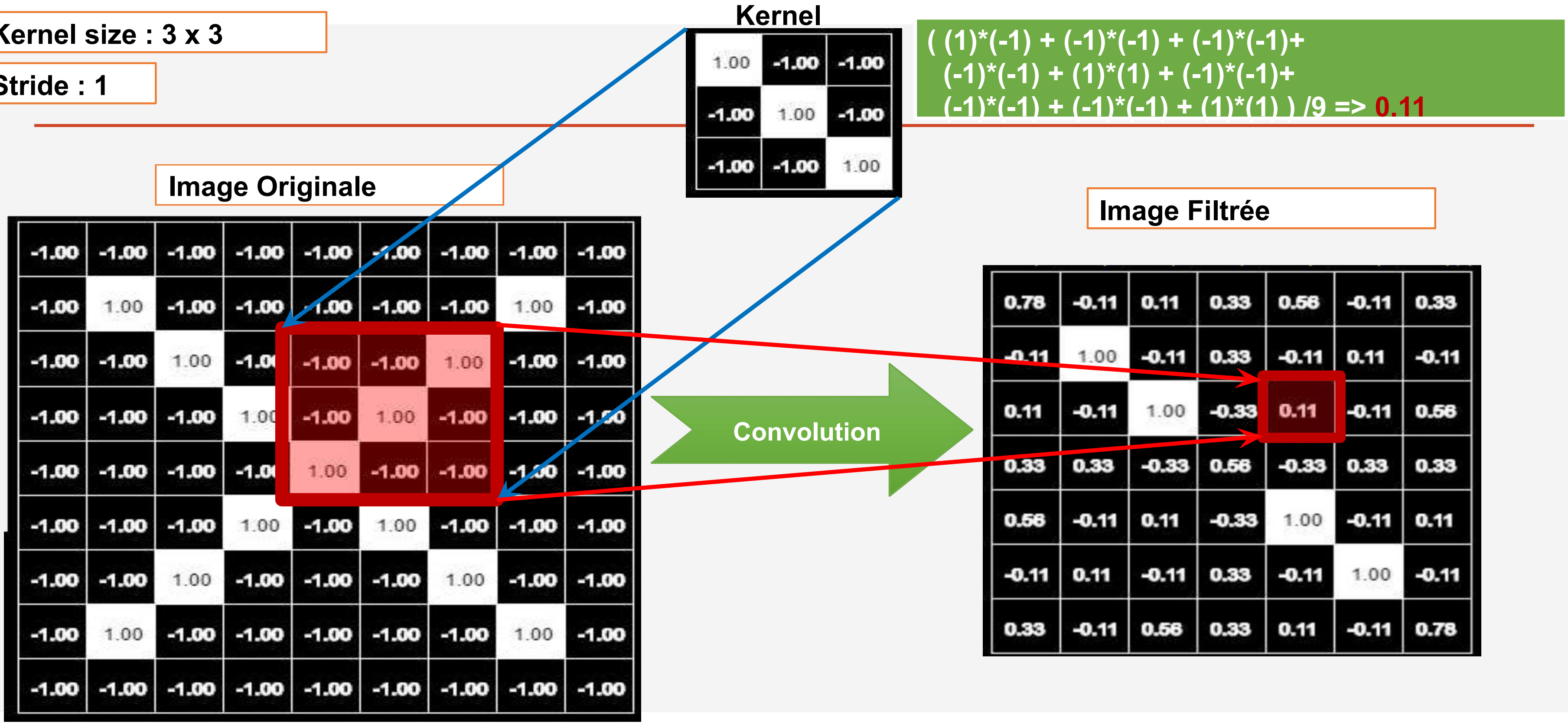
Kernel size : 3 x 3
Stride : 1
Kernel
((1)*(-1) + (-1)*(-1) + (-1)*(-1)+
(-1)*(-1) + (1)*(1) + (-1)*(-1)+
(-1)*(-1) + (-1)*(-1) + (1)*(1)) /9 => -0.11
Image Originale
Convolution
Image Filtrée

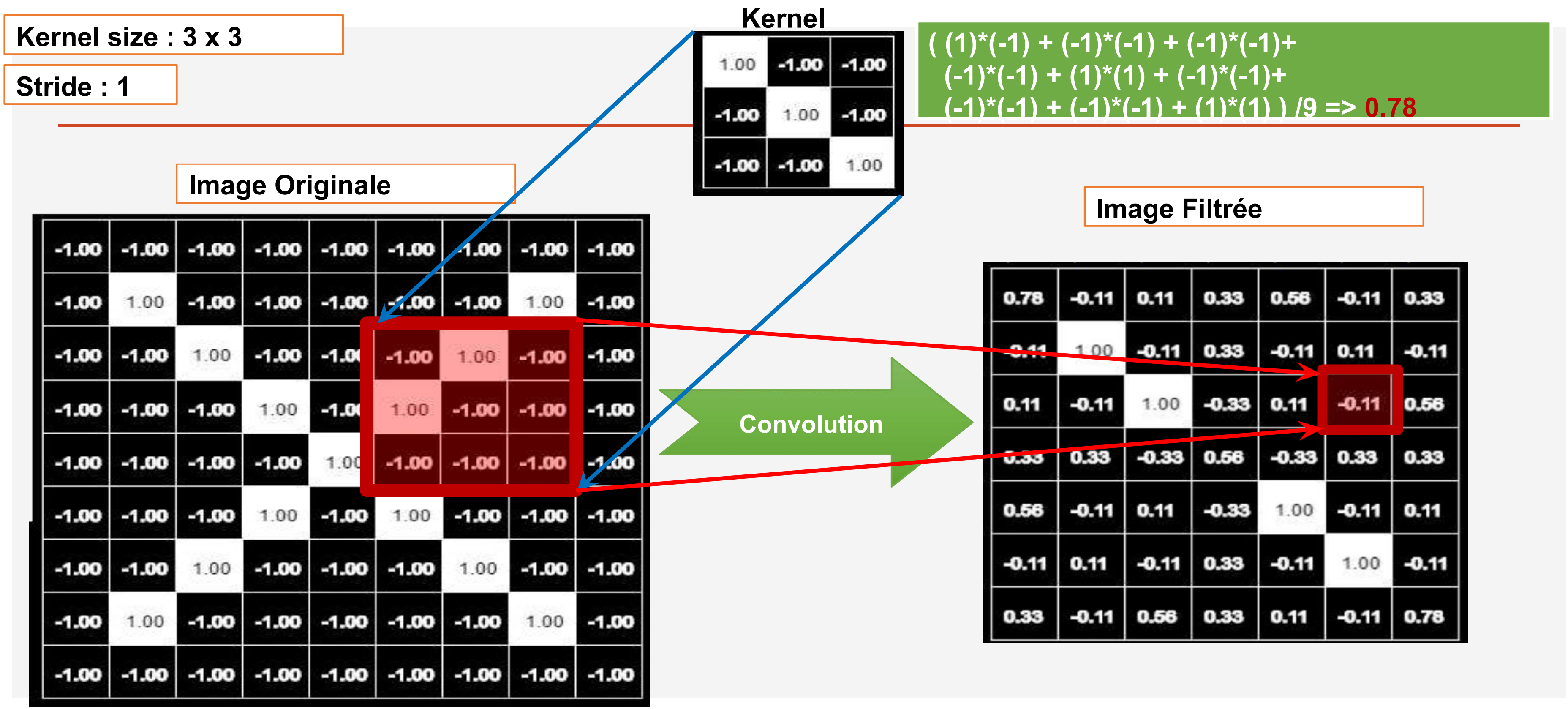

Kernel size : 3 x 3
Stride : 1
Kernel
((1)*(-1) + (-1)*(-1) + (-1)*(-1)+
(-1)*(-1) + (1)*(1) + (-1)*(-1)+
(-1)*(-1) + (-1)*(-1) + (1)*(1)) /9 => 1
Image Originale
Convolution
Image Filtrée


Kernel size : 3 x 3
Stride : 1
Kernel
((1)*(-1) + (-1)*(-1) + (-1)*(-1)+
(-1)*(-1) + (1)*(1) + (-1)*(-1)+
(-1)*(-1) + (-1)*(-1) + (1)*(1)) /9 => -0.33
Image Originale
Convolution
Image Filtrée

Kernel size : 3 x 3

Stride : 1




Kernel 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 => 0.58

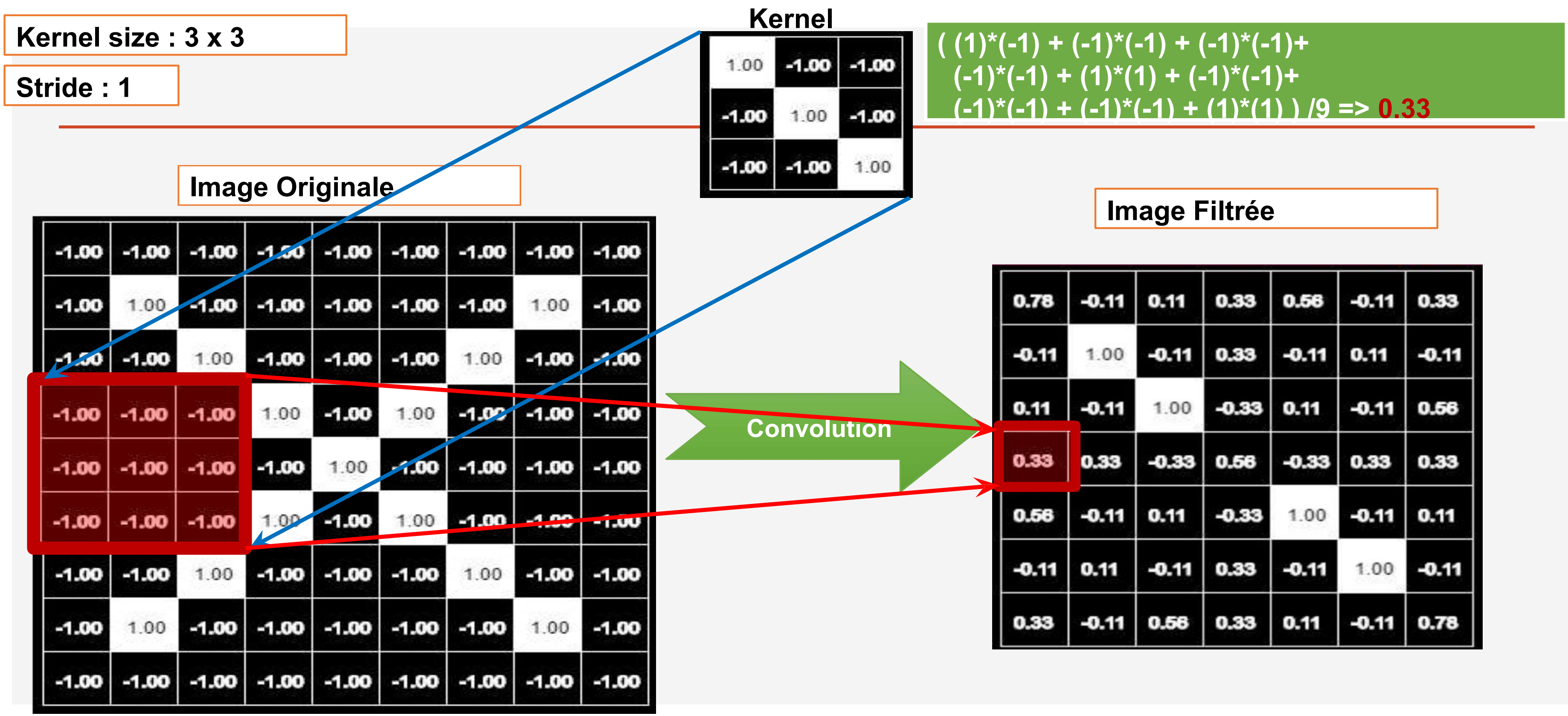
Image Originale

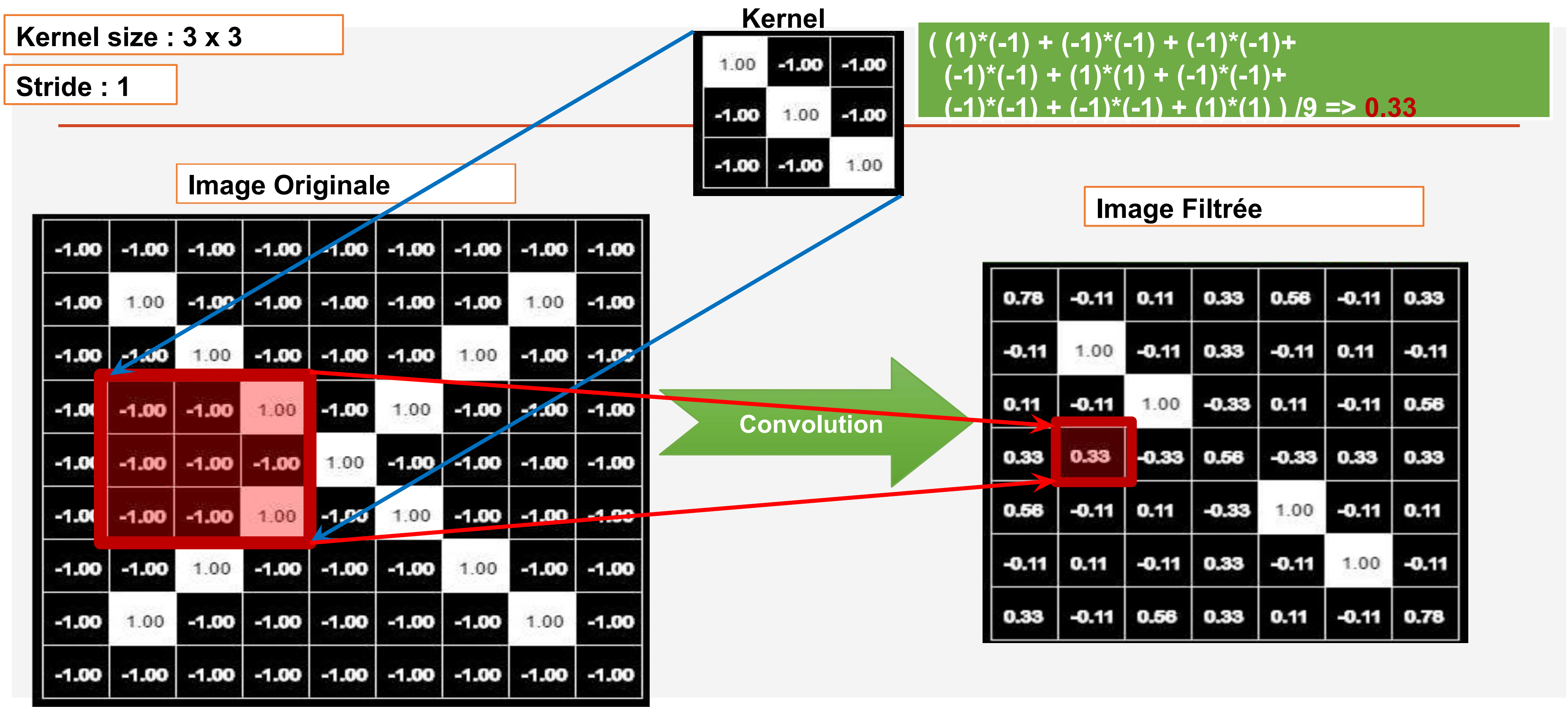
-1.00	-1.00	-1.00	-1.00	-1.00	-1.00	-1.00	-1.00	-1.00
-1.00	1.00	-1.00	-1.00	-1.00	-1.00	-1.00	1.00	-1.00
-1.00	-1.00	1.00	-1.00	-1.00	-1.00	1.00	-1.00	-1.00
-1.00	-1.00	-1.00	1.00	-1.00	1.00	-1.00	-1.00	-1.00
-1.00	-1.00	-1.00	-1.00	1.00	-1.00	-1.00	-1.00	-1.00
-1.00	-1.00	-1.00	1.00	-1.00	1.00	-1.00	-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)*(-1) + (-1)*(-1) + (-1)*(-1)+
(-1)*(-1) + (1)*(1) + (-1)*(-1)+
(-1)*(-1) + (-1)*(-1) + (1)*(1)) /9 => 0.33
Image Originale
Convolution
Image Filtrée



Kernel

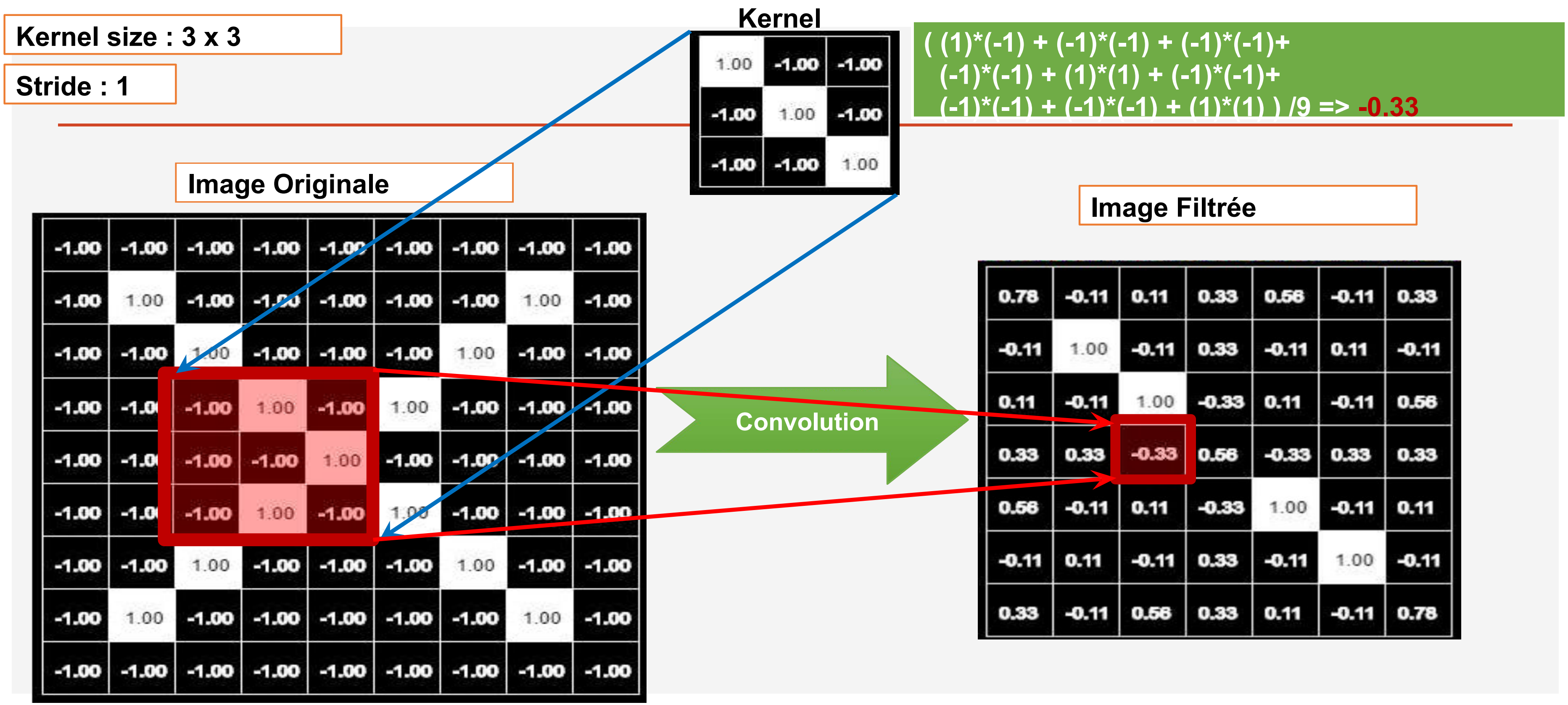
1.00	-1.00	-1.00
-1.00	1.00	-1.00
-1.00	-1.00	1.00

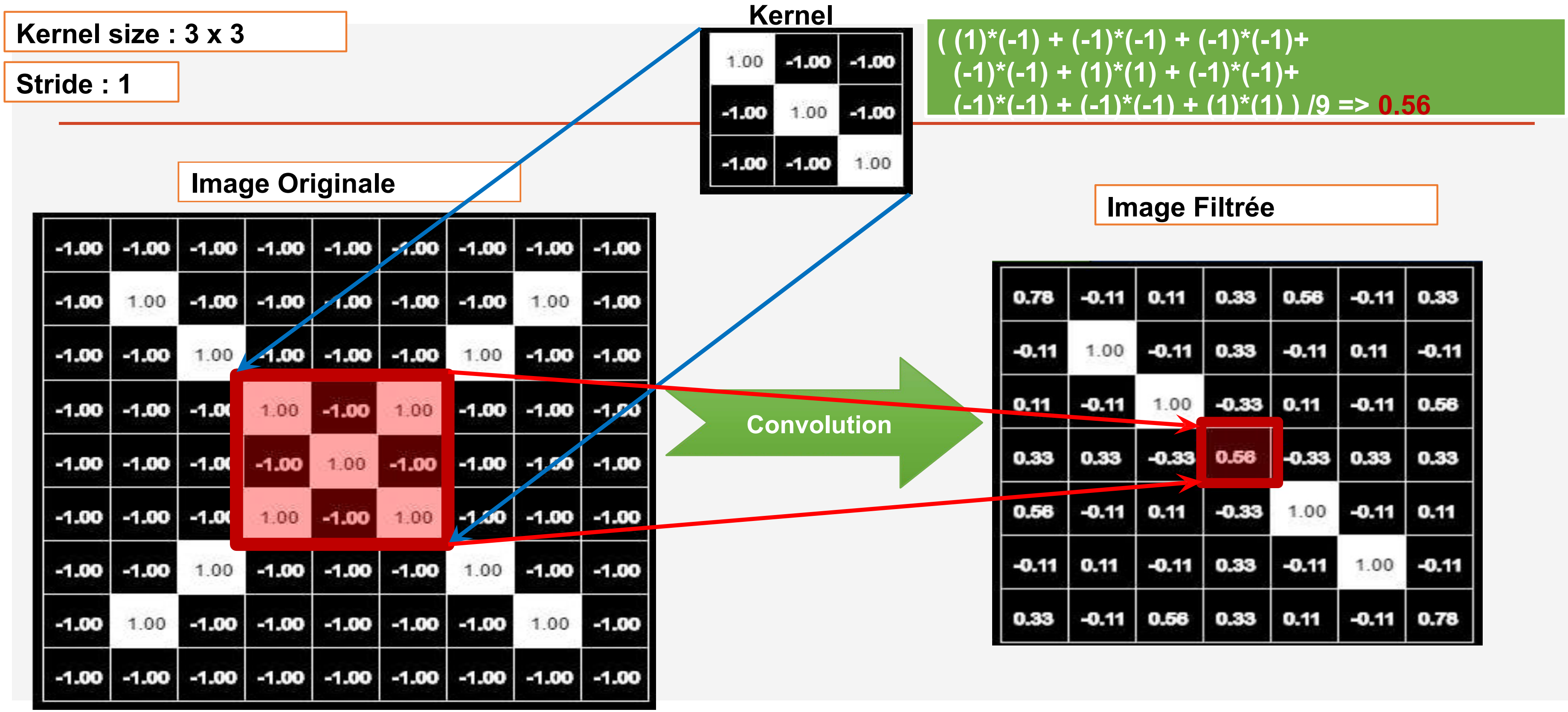
Image Originale

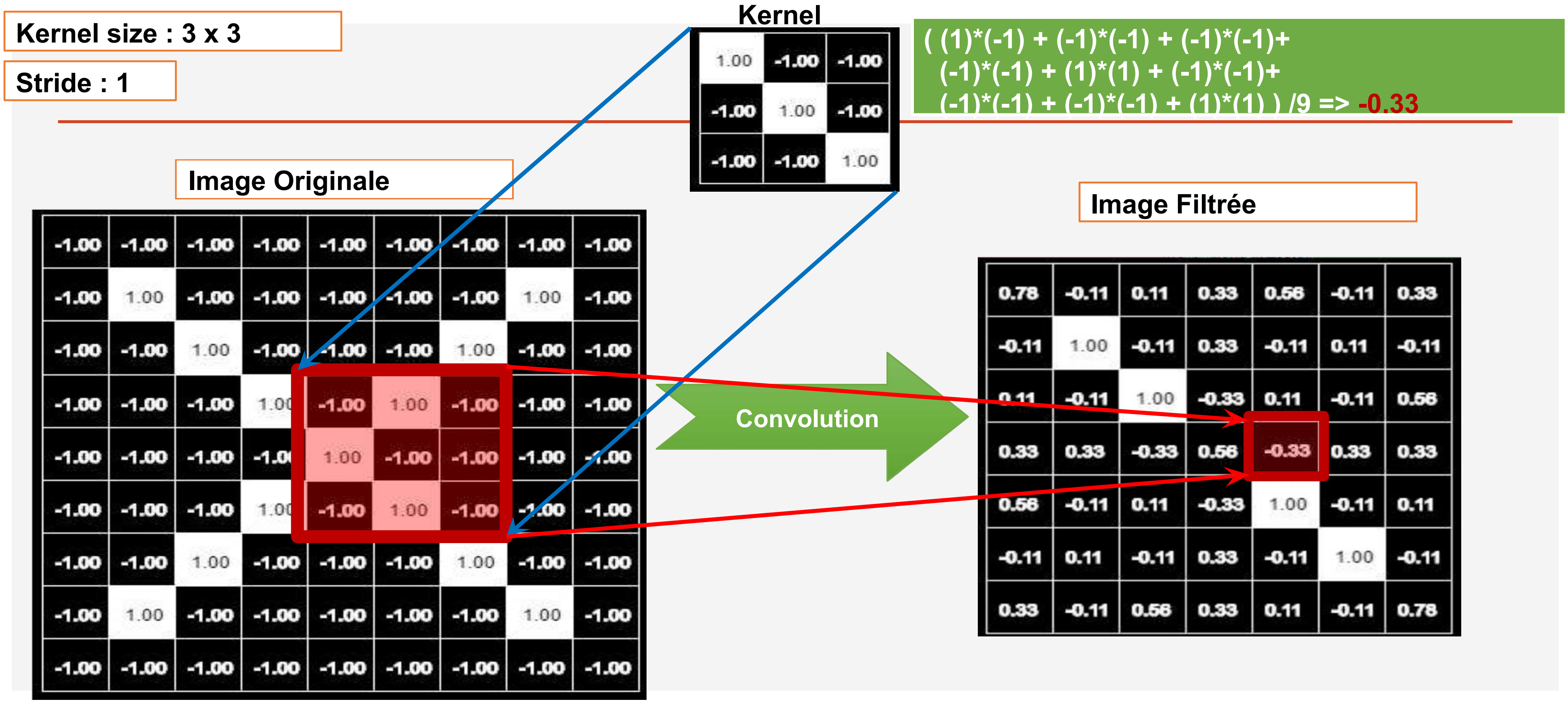
-1.00	-1.00	-1.00	-1.00	-1.00	-1.00	-1.00	-1.00	-1.00
-1.00	1.00	-1.00	-1.00	-1.00	-1.00	-1.00	1.00	-1.00
-1.00	-1.00	1.00	-1.00	-1.00	-1.00	1.00	-1.00	-1.00
-1.00	-1.00	-1.00	1.00	-1.00	1.00	-1.00	-1.00	-1.00
-1.00	-1.00	-1.00	-1.00	1.00	-1.00	-1.00	-1.00	-1.00
-1.00	-1.00	-1.00	1.00	-1.00	1.00	-1.00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

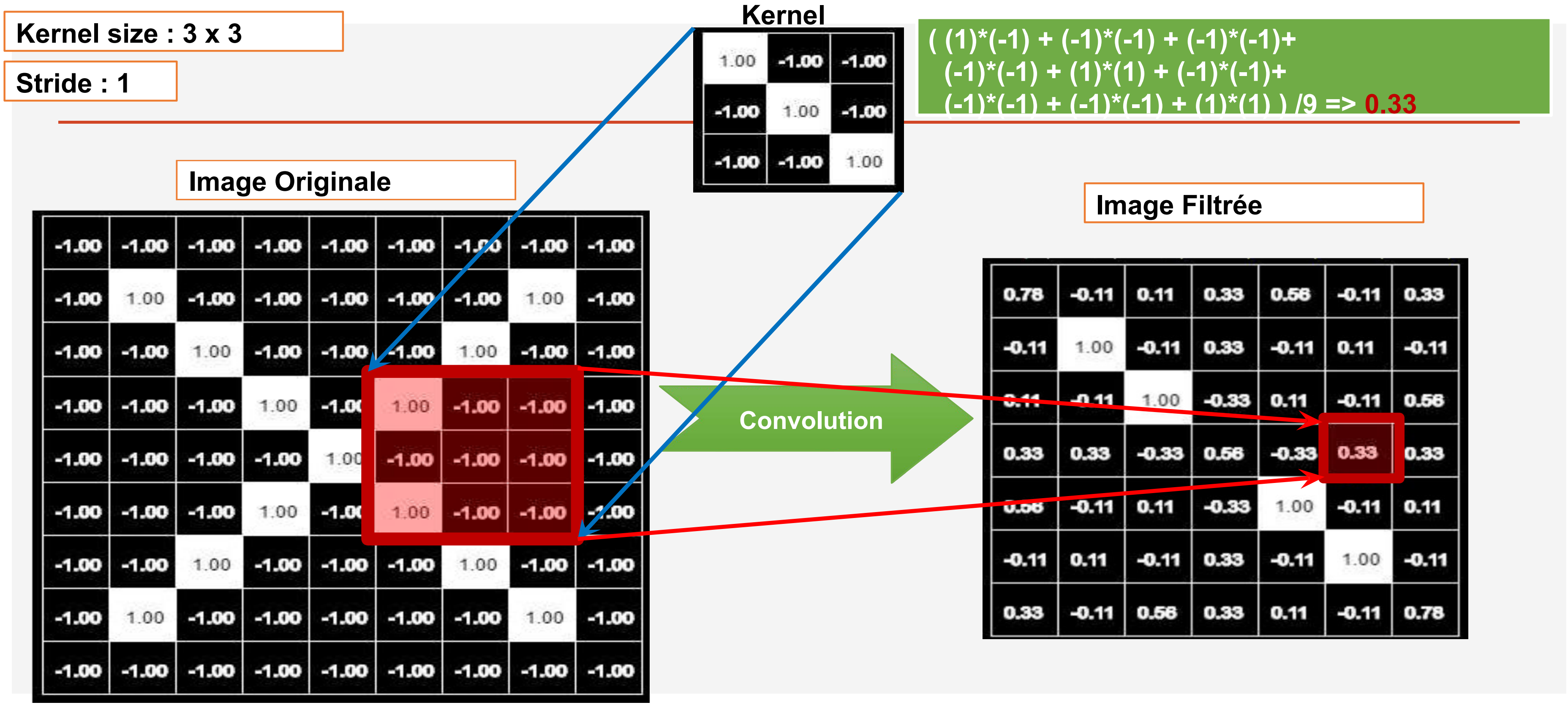
Image Filtrée

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

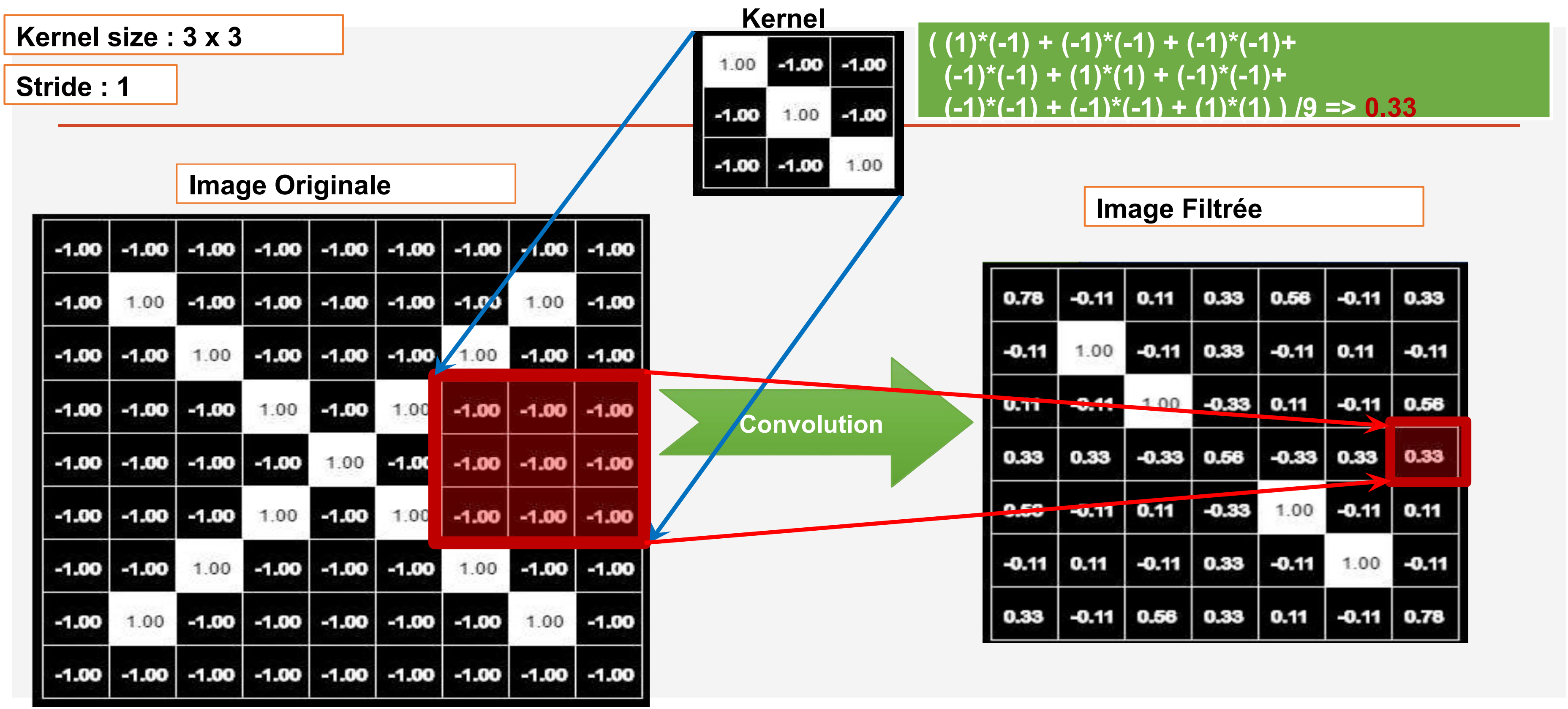

Kernel size : 3 x 3
Stride : 1
Kernel
1.00 -1.00 -1.00
-1.00 1.00 -1.00
-1.00 -1.00 1.00
((1)*(-1) + (-1)*(-1) + (-1)*(-1)+
(-1)*(-1) + (1)*(1) + (-1)*(-1)+
(-1)*(-1) + (-1)*(-1) + (1)*(1)) /9 => -0.33
Image Originale
-1.00 -1.00 -1.00 -1.00 -1.00 -1.00 -1.00 -1.00 -1.00
-1.00 1.00 -1.00 -1.00 -1.00 -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)*(-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

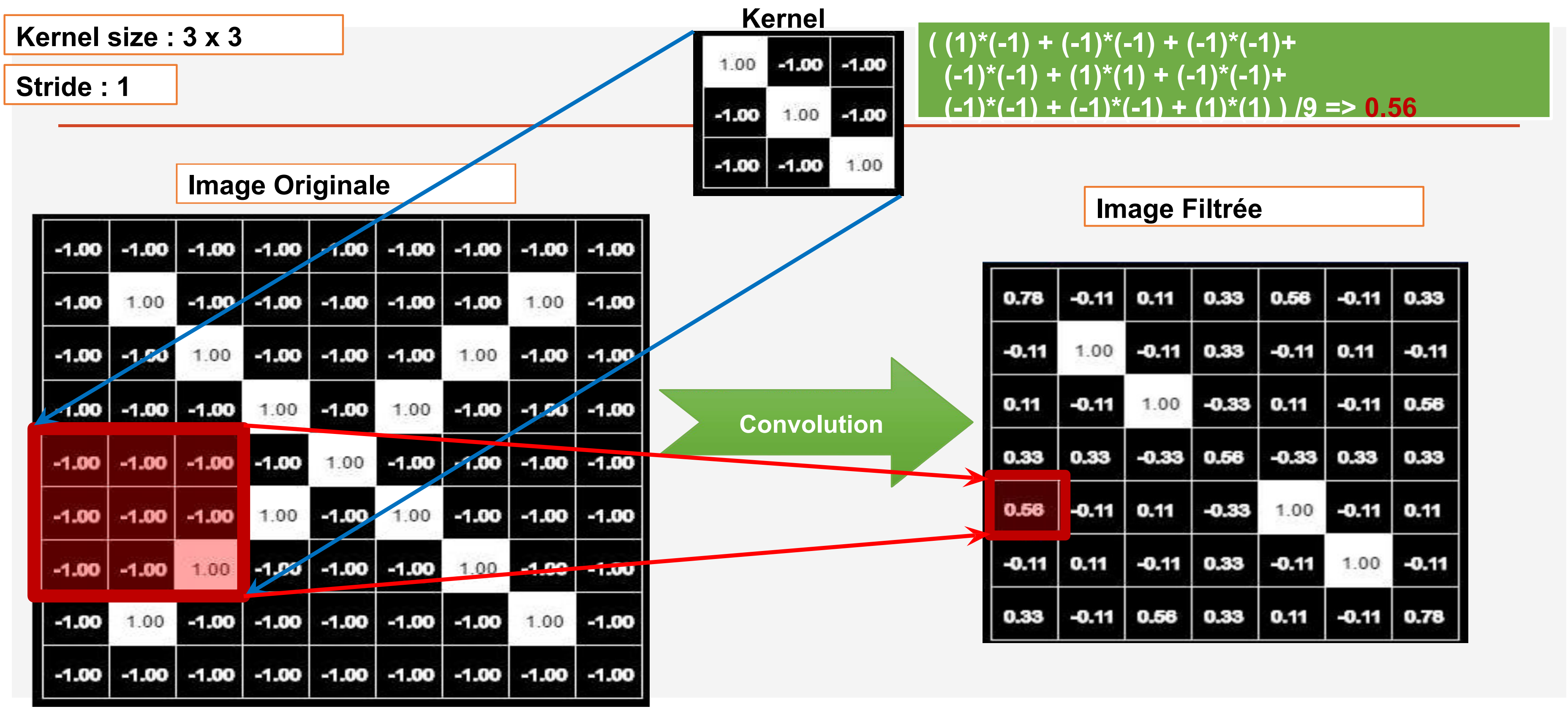

Kernel size : 3 x 3
Stride : 1
Kernel
1.00 -1.00 -1.00
-1.00 1.00 -1.00
-1.00 -1.00 1.00
((1)*(-1) + (-1)*(-1) + (-1)*(-1)+
(-1)*(-1) + (1)*(1) + (-1)*(-1)+
(-1)*(-1) + (-1)*(-1) + (1)*(1)) /9 => -0.33
Image Originale
-1.00 -1.00 -1.00 -1.00 -1.00 -1.00 -1.00 -1.00 -1.00
-1.00 1.00 -1.00 -1.00 -1.00 -1.00 -1.00 1.00 -1.00
-1.00 -1.00 1.00 -1.00 -1.00 -1.00 1.00 -1.00 -1.00
-1.00 -1.00 -1.00 1.00 -1.00 1.00 -1.00 -1.00 -1.00
-1.00 -1.00 -1.00 -1.00 1.00 -1.00 -1.00 -1.00 -1.00
-1.00 -1.00 -1.00 1.00 -1.00 1.00 -1.00 -1.00 -1.00
-1.00 -1.00 1.00 -1.00 -1.00 -1.00 1.00 -1.00 -1.00
-1.00 1.00 -1.00 -1.00 -1.00 -1.00 -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)*(-1) + (-1)*(-1) + (-1)*(-1)+
(-1)*(-1) + (1)*(1) + (-1)*(-1)+
(-1)*(-1) + (-1)*(-1) + (1)*(1)) /9 => 0.33
Image Originale
Convolution
Image Filtrée



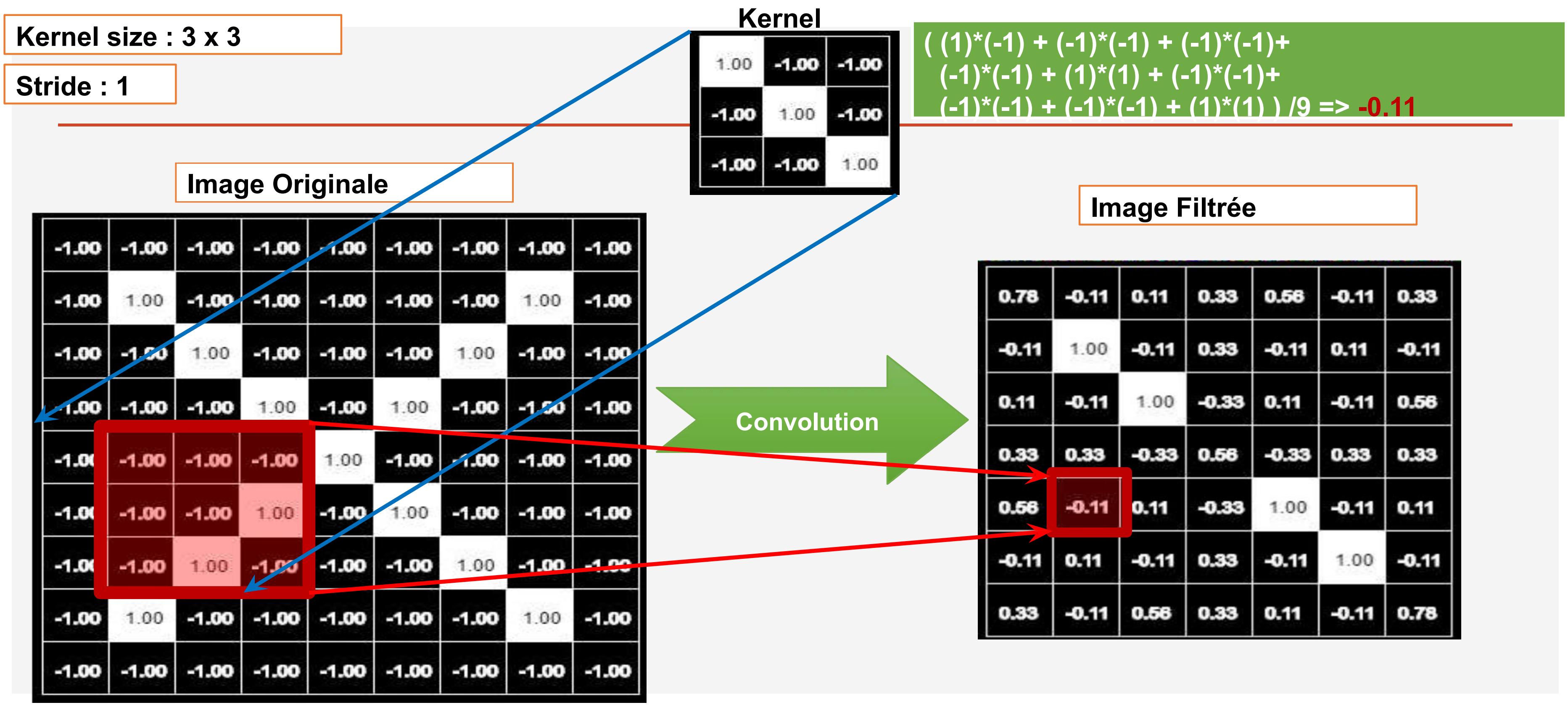
Kernel size : 3 x 3
Stride : 1
Kernel
1.00 -1.00 -1.00
-1.00 1.00 -1.00
-1.00 -1.00 1.00
((1)*(-1) + (-1)*(-1) + (-1)*(-1)+
(-1)*(-1) + (1)*(1) + (-1)*(-1)+
(-1)*(-1) + (-1)*(-1) + (1)*(1)) /9 => 0.33
Image Originale
-1.00 -1.00 -1.00 -1.00 -1.00 -1.00 -1.00 -1.00 -1.00
-1.00 1.00 -1.00 -1.00 -1.00 -1.00 -1.00 1.00 -1.00
-1.00 -1.00 1.00 -1.00 -1.00 -1.00 1.00 -1.00 -1.00
-1.00 -1.00 -1.00 1.00 -1.00 1.00 -1.00 -1.00 -1.00
-1.00 -1.00 -1.00 -1.00 1.00 -1.00 -1.00 -1.00 -1.00
-1.00 -1.00 -1.00 1.00 -1.00 1.00 -1.00 -1.00 -1.00
-1.00 -1.00 1.00 -1.00 -1.00 -1.00 1.00 -1.00 -1.00
-1.00 1.00 -1.00 -1.00 -1.00 -1.00 -1.00 1.00 -1.00
-1.00 -1.00 -1.00 -1.00 -1.00 -1.00 -1.00 -1.00 -1.00
Convolution
Image Filtrée
0.78 -0.11 0.11 0.33 0.56 -0.11 0.33
-0.11 1.00 -0.11 0.33 -0.11 0.11 -0.11
0.11 -0.11 1.00 -0.33 0.11 -0.11 0.56
0.33 0.33 -0.33 0.56 -0.33 0.33 0.33
0.56 -0.11 0.11 -0.33 1.00 -0.11 0.11
-0.11 0.11 -0.11 0.33 -0.11 1.00 -0.11
0.33 -0.11 0.56 0.33 0.11 -0.11 0.78



Kernel size : 3 x 3
Stride : 1
Kernel
1.00 -1.00 -1.00
-1.00 1.00 -1.00
-1.00 -1.00 1.00
((1)*(-1) + (-1)*(-1) + (-1)*(-1)+
(-1)*(-1) + (1)*(1) + (-1)*(-1)+
(-1)*(-1) + (-1)*(-1) + (1)*(1)) /9 => 0.56
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 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

$$((1)*(-1) + (-1)*(-1) + (-1)*(-1) + (-1)*(-1) + (1)*(1) + (-1)*(-1) + (-1)*(-1) + (-1)*(-1) + (1)*(1)) /9 \Rightarrow 0.11$$

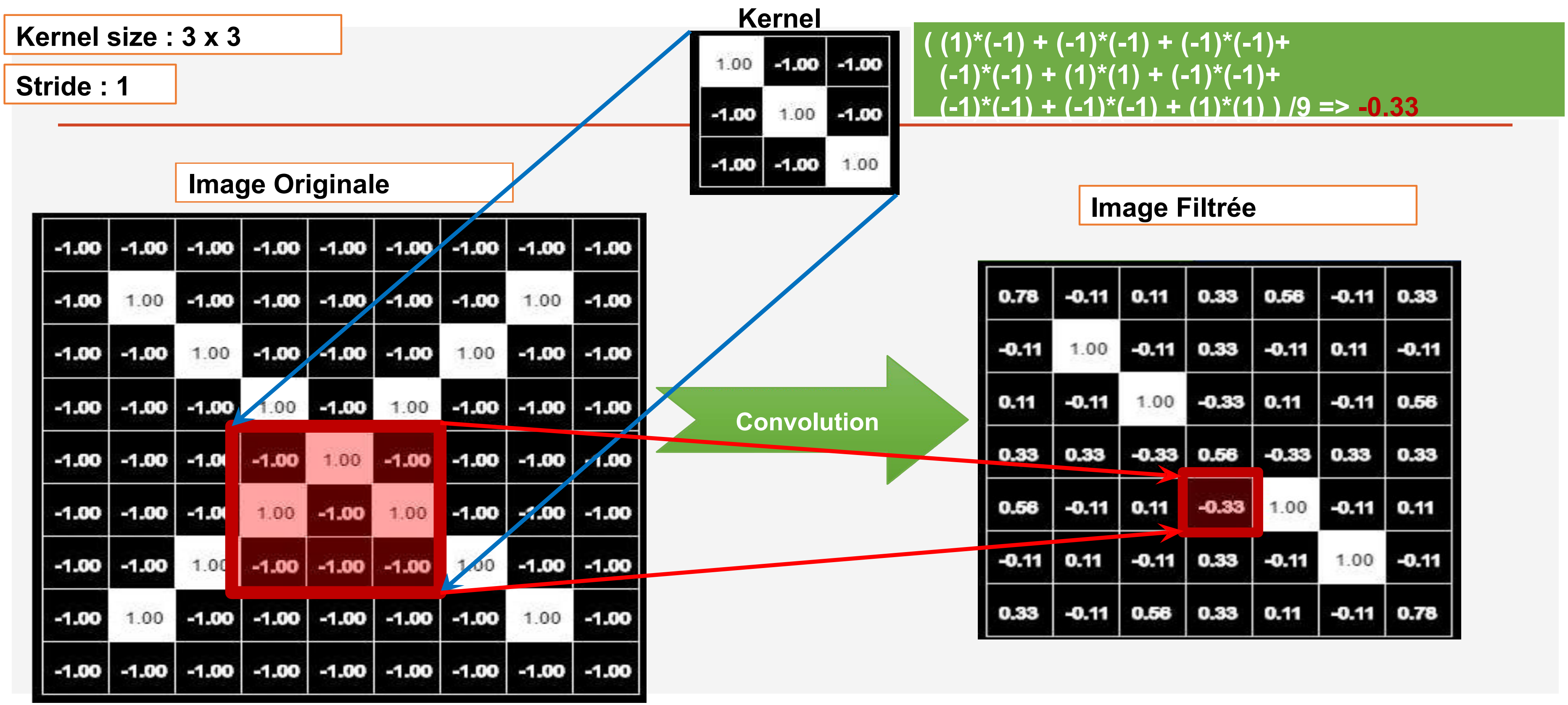
Image Originale

-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

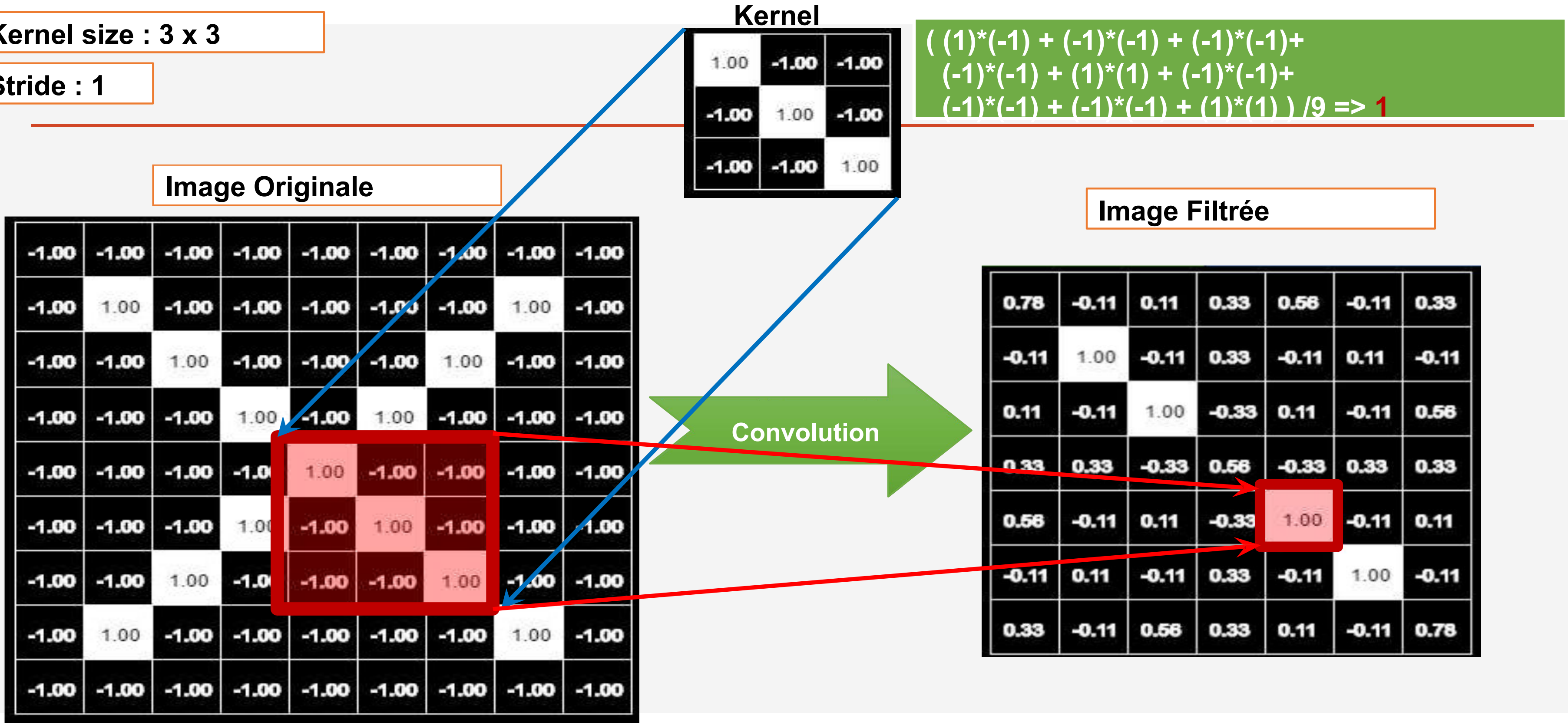
Image Filtrée

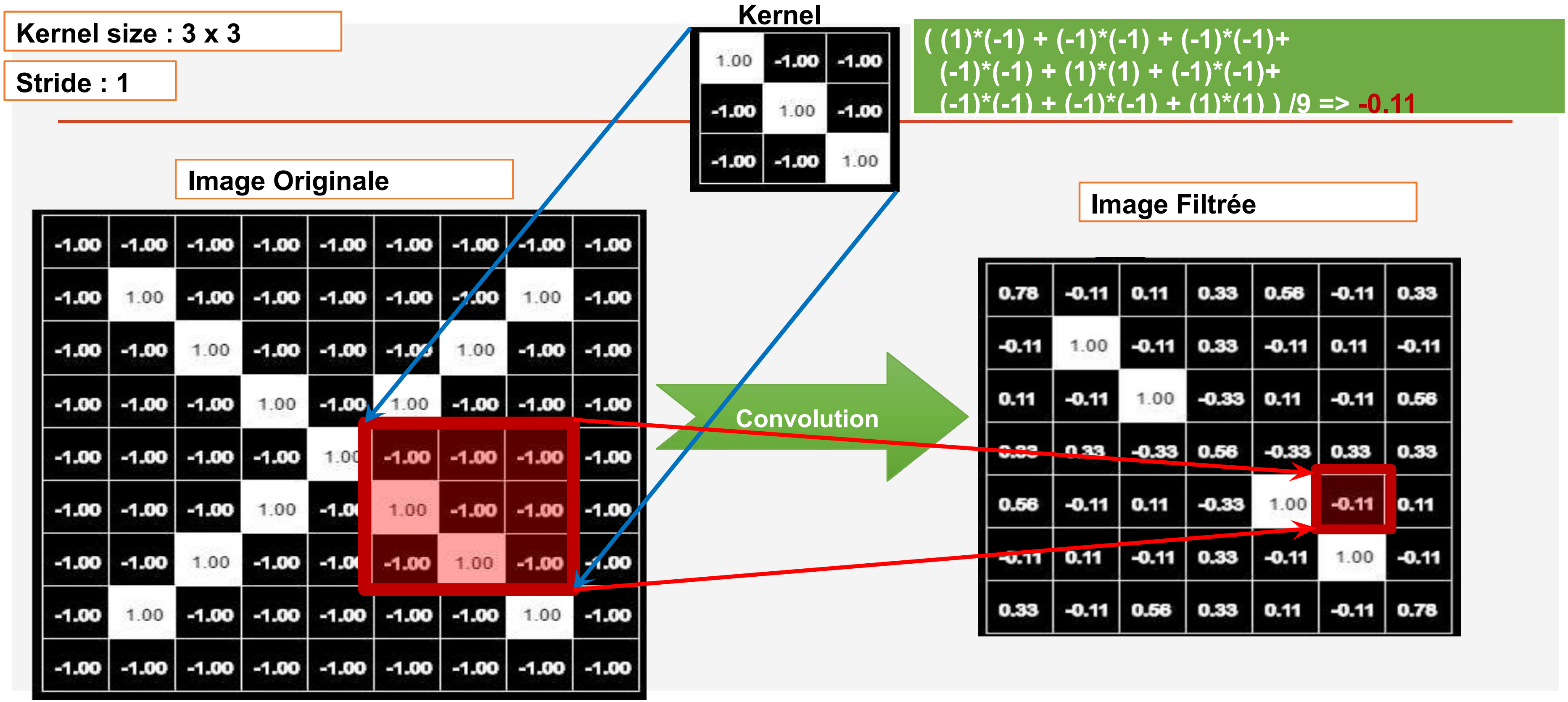
0.78	-0.11	0.11	0.33	0.56	-0.11	0.33
-0.11	1.00	-0.11	0.33	-0.11	0.11	-0.11
0.11	-0.11	1.00	-0.33	0.11	-0.11	0.56
0.33	0.33	-0.33	0.56	-0.33	0.33	0.33
0.56	-0.11	0.11	-0.33	1.00	-0.11	0.11
-0.11	0.11	-0.11	0.33	-0.11	1.00	-0.11
0.33	-0.11	0.56	0.33	0.11	-0.11	0.78


Kernel size : 3 x 3
Stride : 1
Kernel
((1)*(-1) + (-1)*(-1) + (-1)*(-1)+
(-1)*(-1) + (1)*(1) + (-1)*(-1)+
(-1)*(-1) + (-1)*(-1) + (1)*(1)) /9 => -0.33
Image Originale
Convolution
Image Filtrée

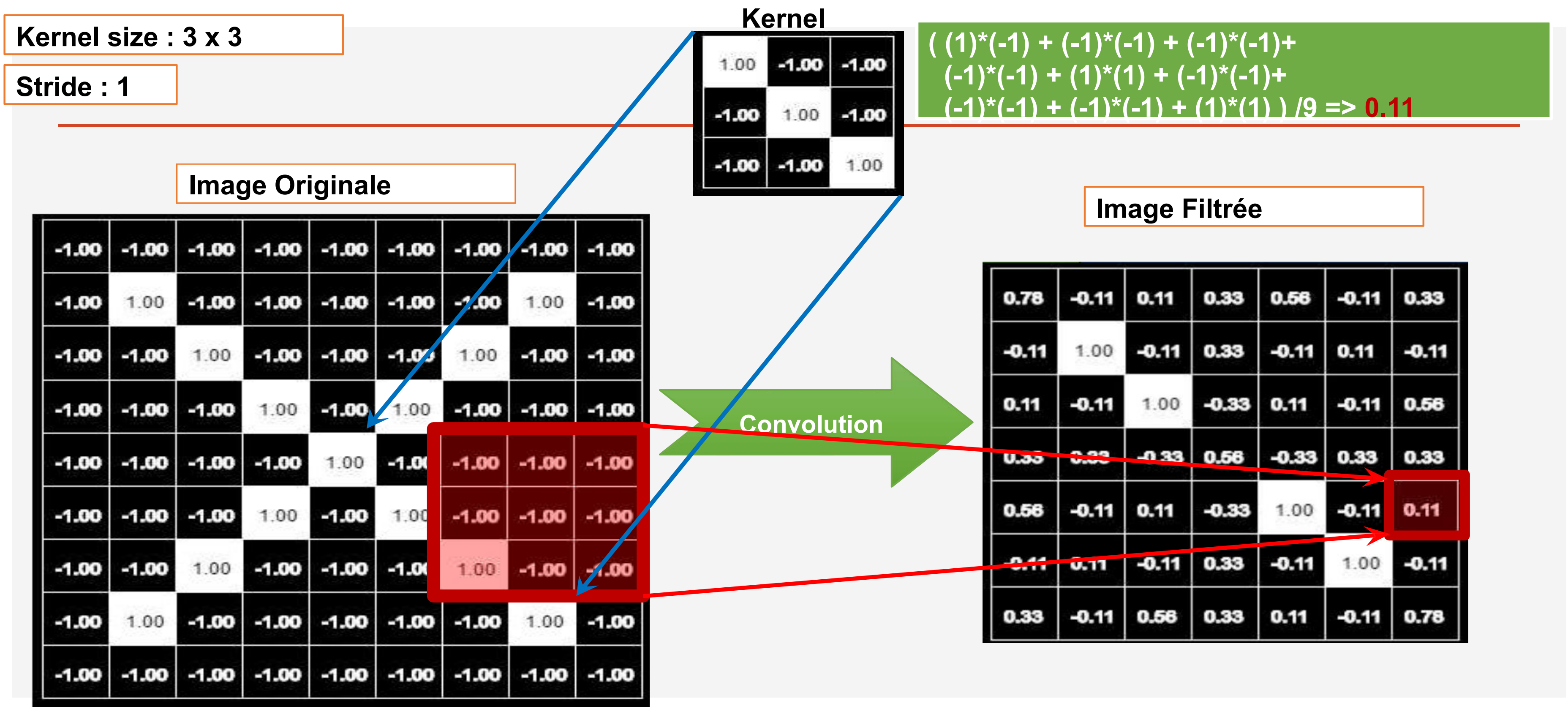
Kernel size : 3 x 3

Stride : 1





Kernel 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)*(-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

$$((1)*(-1) + (-1)*(-1) + (-1)*(-1)+ (-1)*(-1) + (1)*(1) + (-1)*(-1)+ (-1)*(-1) + (-1)*(-1) + (1)*(1)) /9 \Rightarrow -0.11$$

Image Originale

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

Convolution

Image Filtrée

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

Kernel size : 3 x 3

Stride : 1

Kernel

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

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

Image Originale

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

Convolution

Image Filtrée

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

Kernel size : 3 x 3

Stride : 1

Kernel

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

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

Image Originale

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

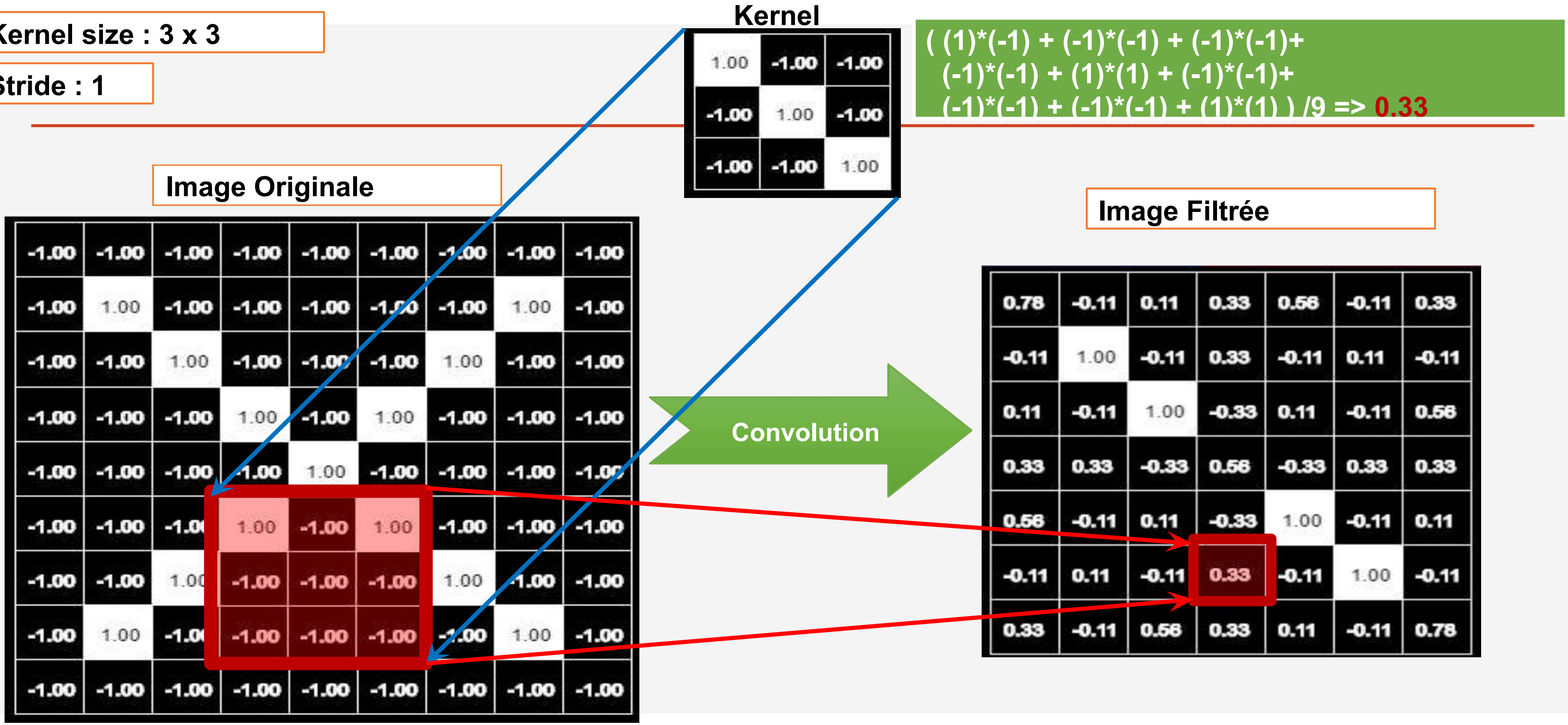
Convolution

Image Filtrée

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

Kernel size : 3 x 3

Stride : 1



Kernel 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

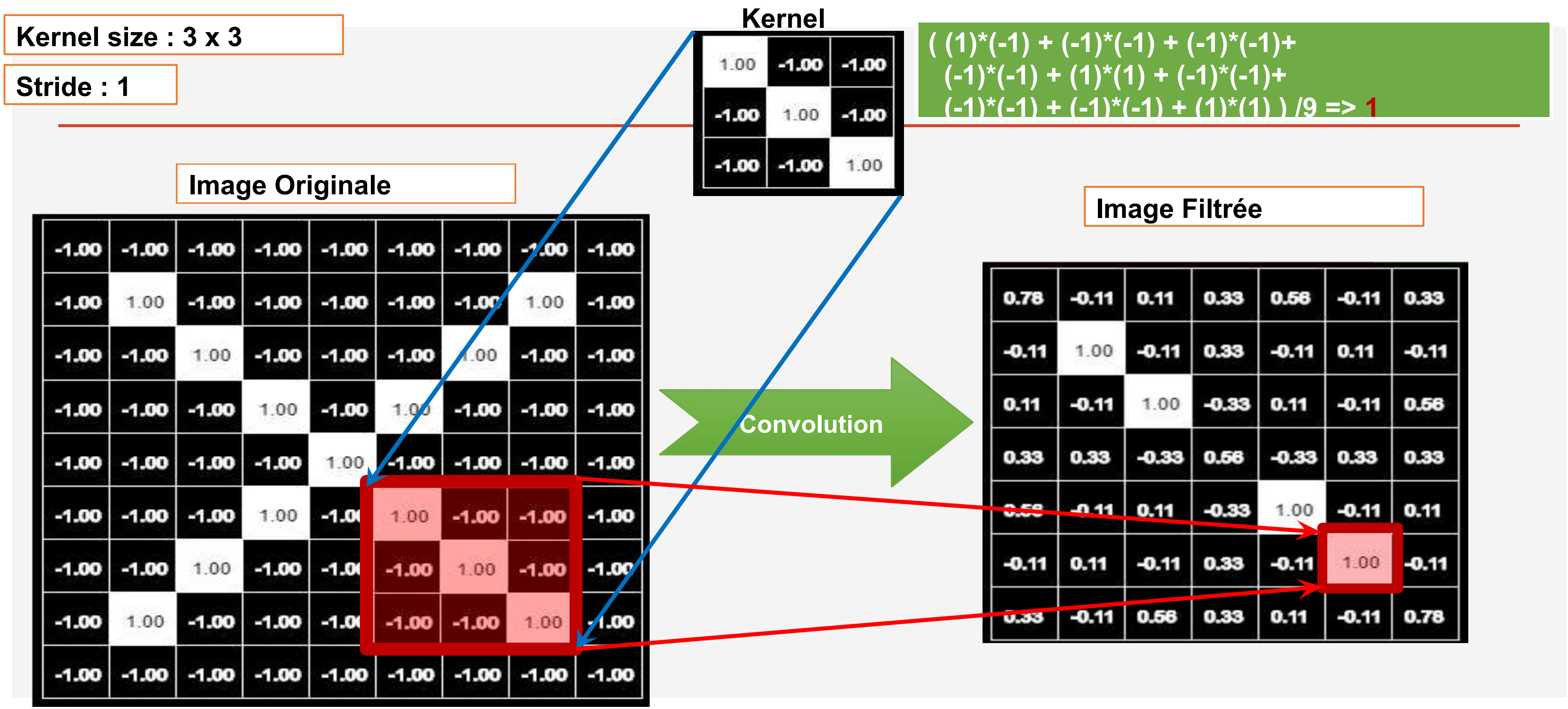
Image Originale

-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

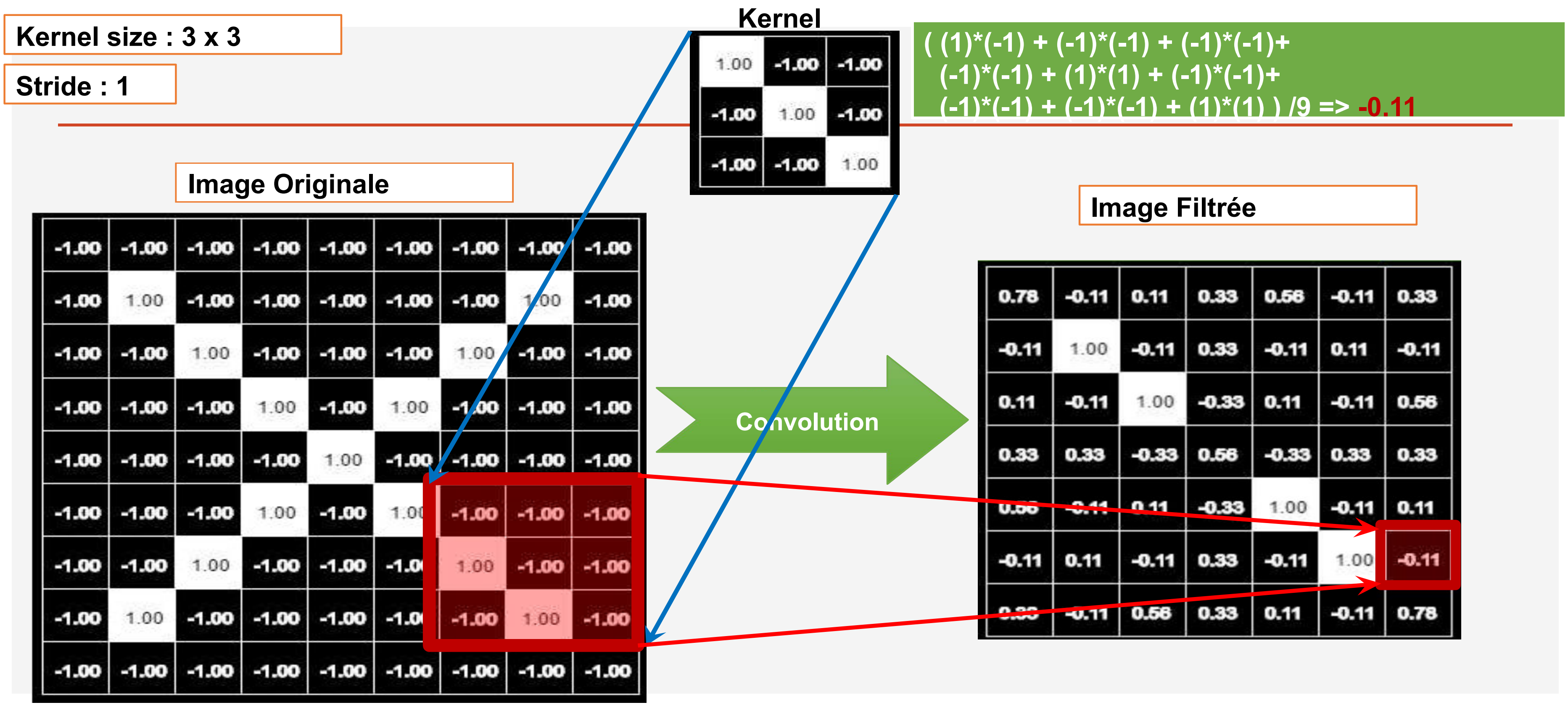
Convolution

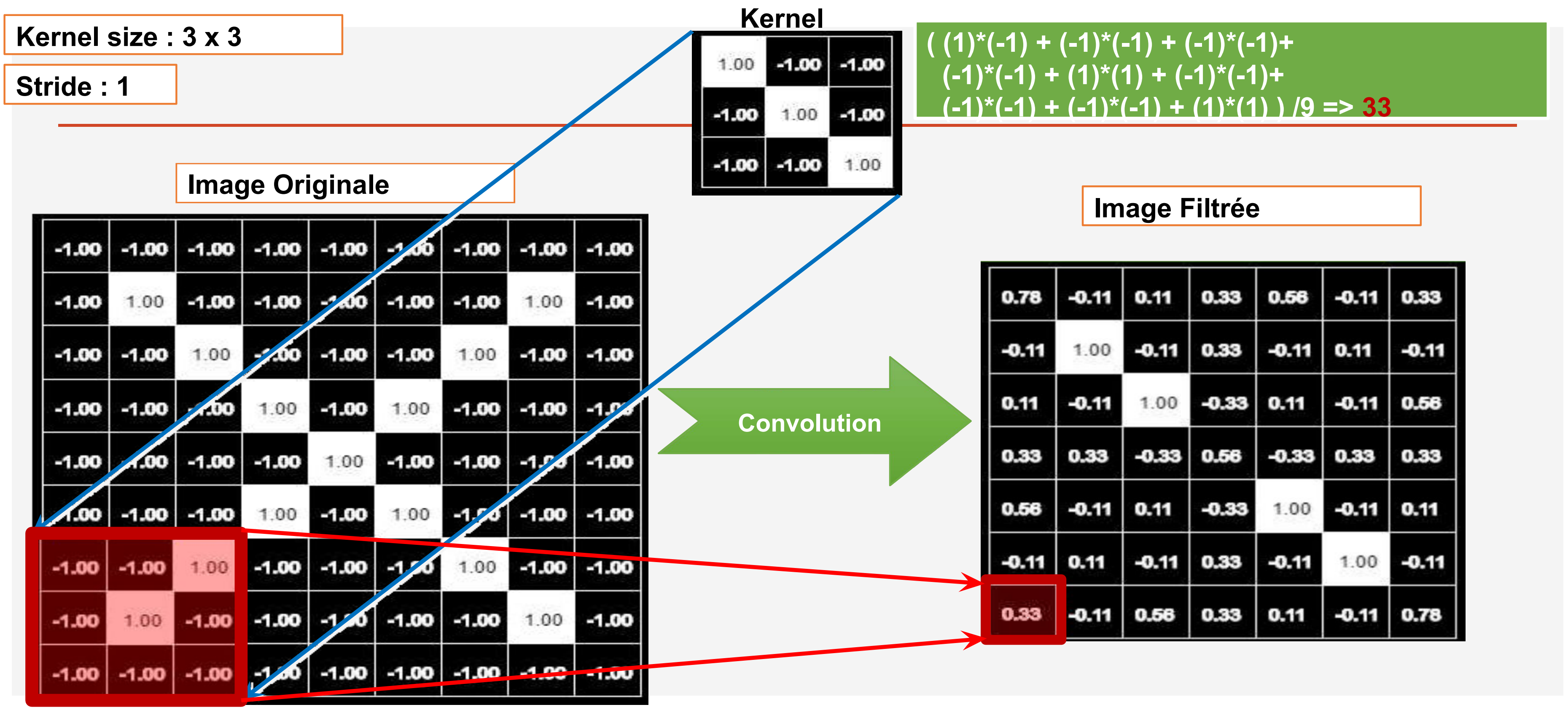
Image Filtrée

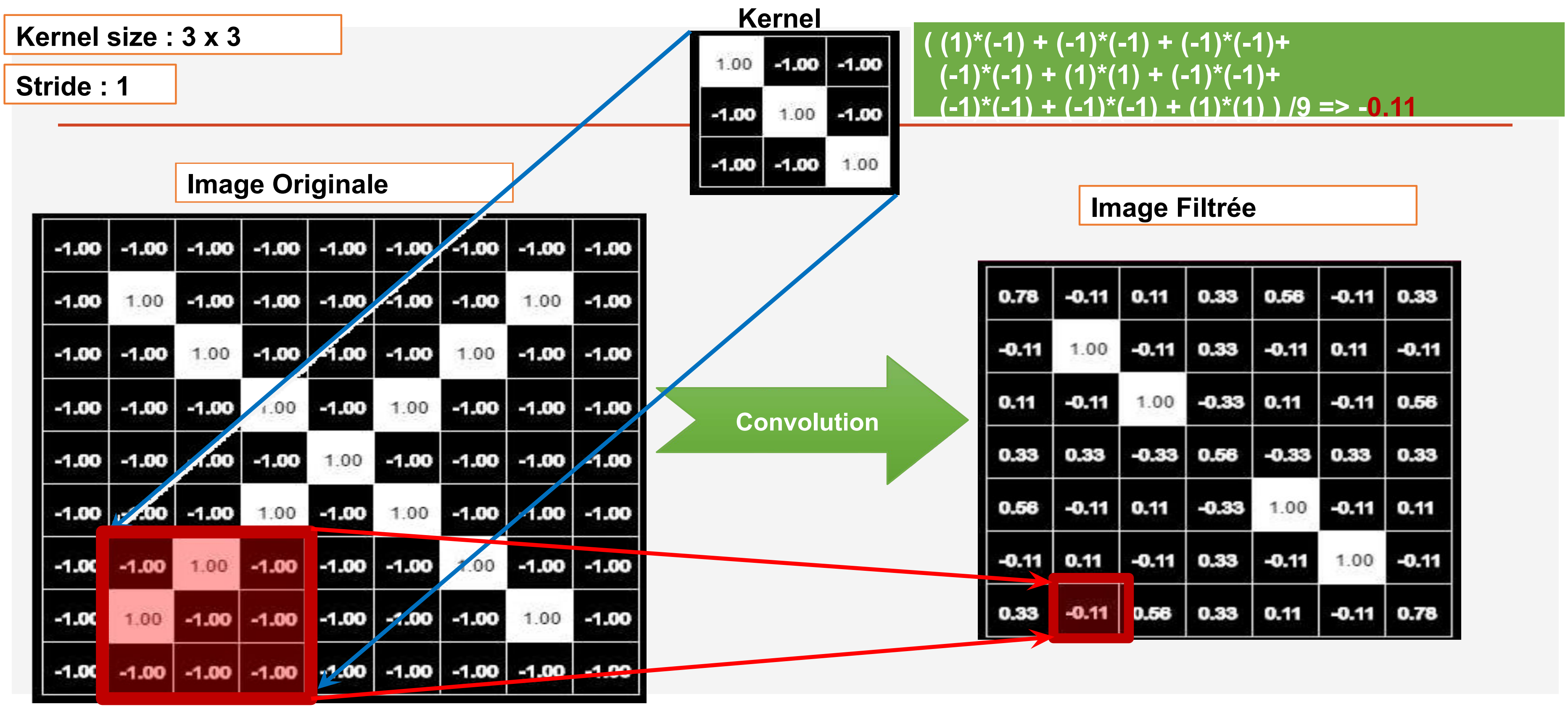
0.78	-0.11	0.11	0.33	0.56	-0.11	0.33
-0.11	1.00	-0.11	0.33	-0.11	0.11	-0.11
0.11	-0.11	1.00	-0.33	0.11	-0.11	0.56
0.33	0.33	-0.33	0.56	-0.33	0.33	0.33
0.56	-0.11	0.11	-0.33	1.00	-0.11	0.11
-0.11	0.11	-0.11	0.33	-0.11	1.00	-0.11
0.33	-0.11	0.56	0.33	0.11	-0.11	0.78



Kernel size : 3 x 3
Stride : 1
Kernel
((1)*(-1) + (-1)*(-1) + (-1)*(-1)+
(-1)*(-1) + (1)*(1) + (-1)*(-1)+
(-1)*(-1) + (-1)*(-1) + (1)*(1)) /9 => 1
Image Originale
Convolution
Image Filtrée

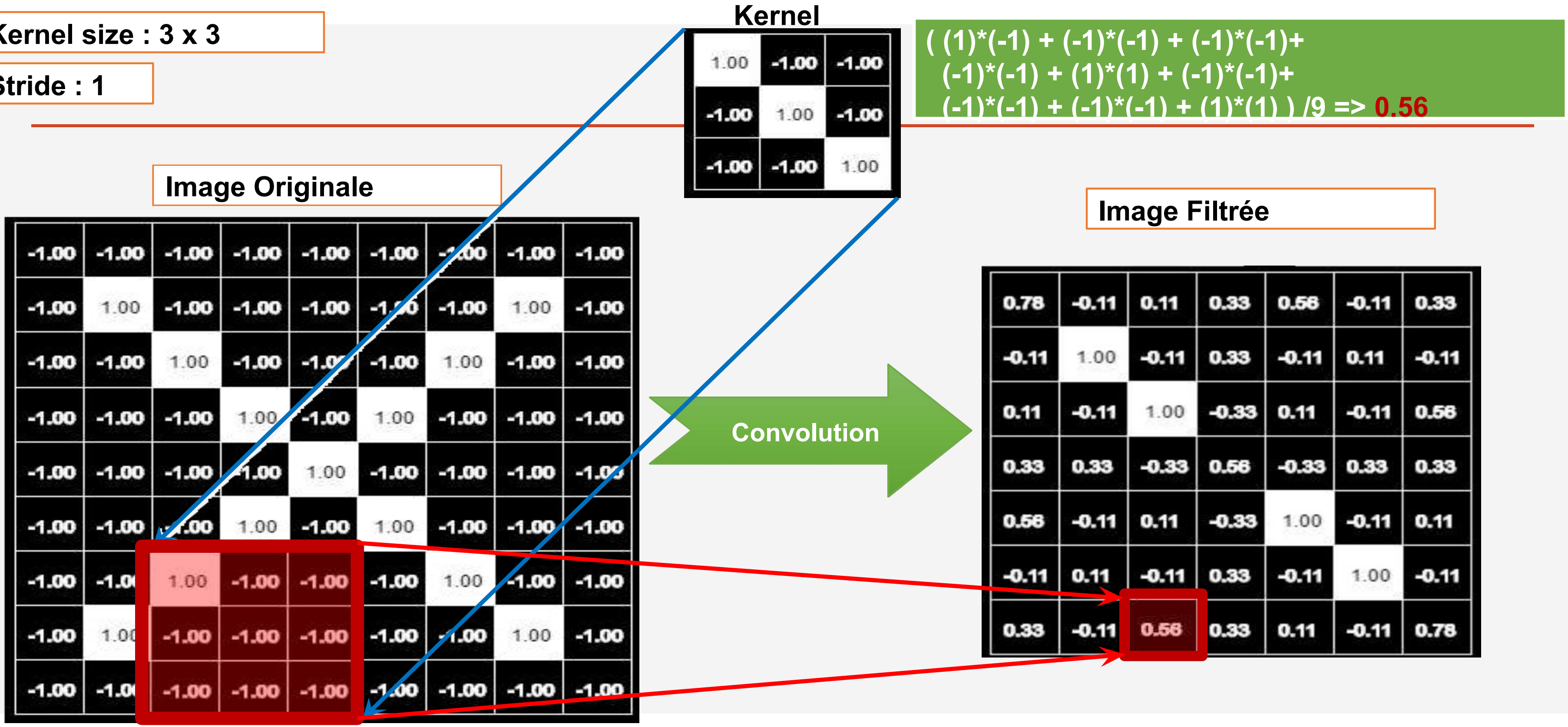

Kernel size : 3 x 3
Stride : 1
Kernel
((1)*(-1) + (-1)*(-1) + (-1)*(-1)+
(-1)*(-1) + (1)*(1) + (-1)*(-1)+
(-1)*(-1) + (-1)*(-1) + (1)*(1)) /9 => -0.11
Image Originale
Convolution
Image Filtrée

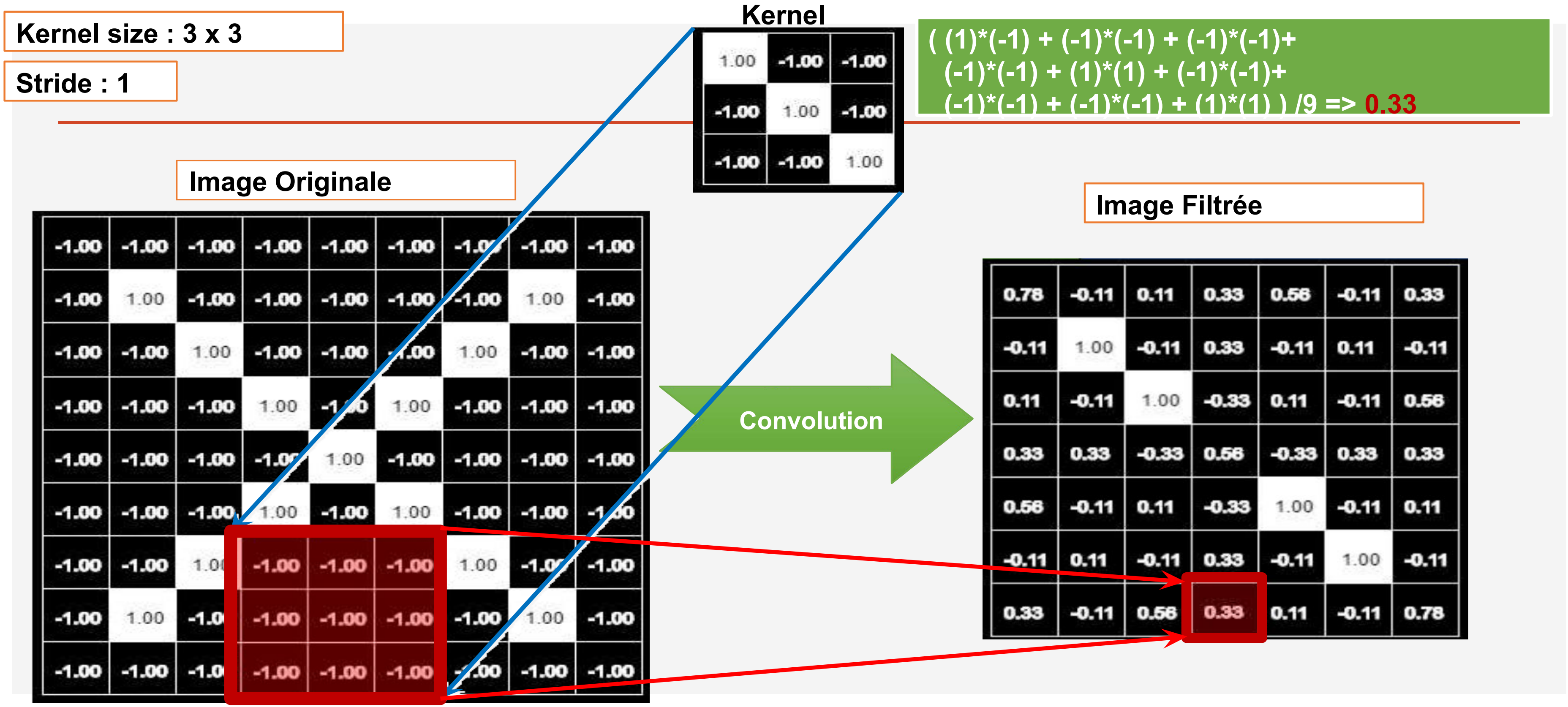

Kernel size : 3 x 3
Stride : 1
Kernel
((1)*(-1) + (-1)*(-1) + (-1)*(-1)+
(-1)*(-1) + (1)*(1) + (-1)*(-1)+
(-1)*(-1) + (-1)*(-1) + (1)*(1)) /9 => 33
Image Originale
Convolution
Image Filtrée


Kernel size : 3 x 3
Stride : 1
Kernel
1.00 -1.00 -1.00
-1.00 1.00 -1.00
-1.00 -1.00 1.00
((1)*(-1) + (-1)*(-1) + (-1)*(-1)+
(-1)*(-1) + (1)*(1) + (-1)*(-1)+
(-1)*(-1) + (-1)*(-1) + (1)*(1)) /9 => -0.11
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

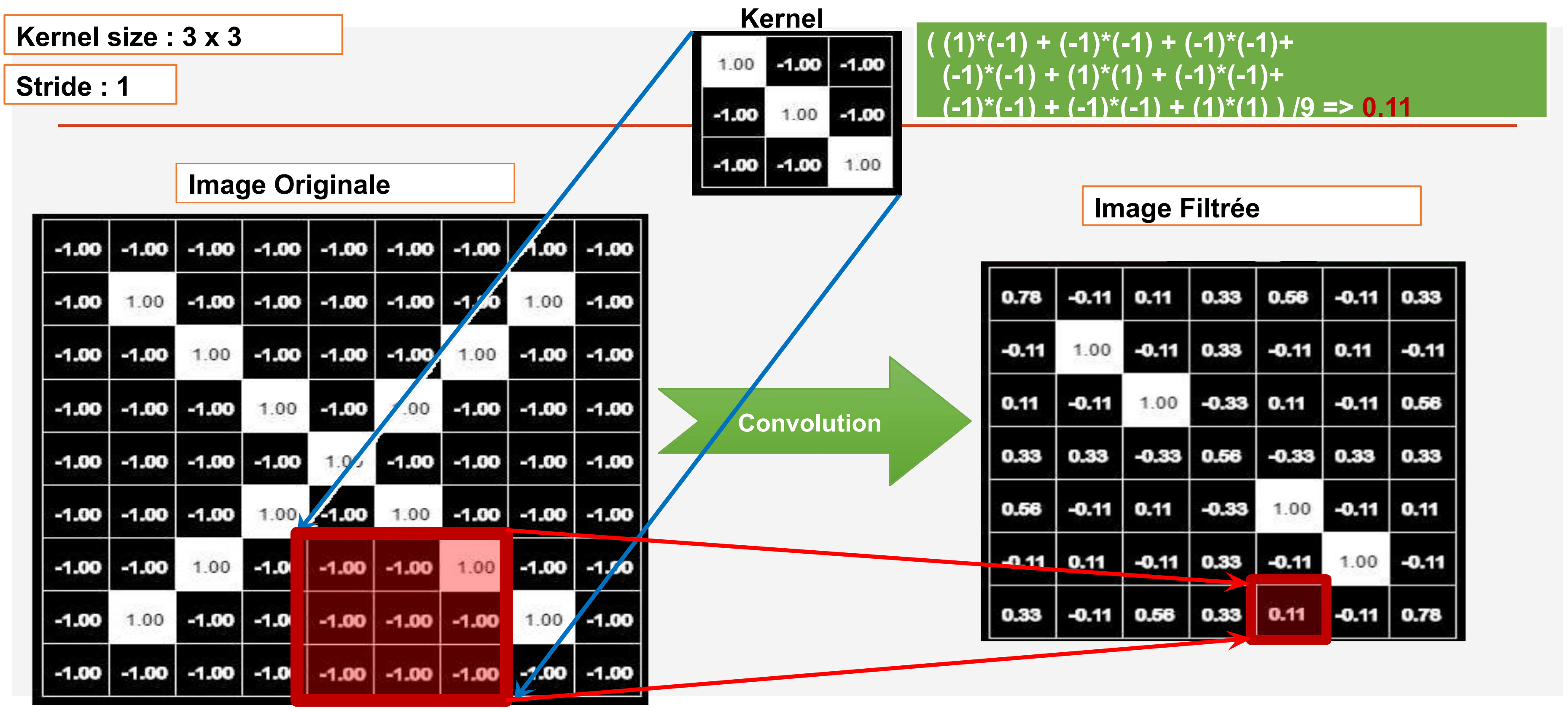
Kernel size : 3 x 3

Stride : 1

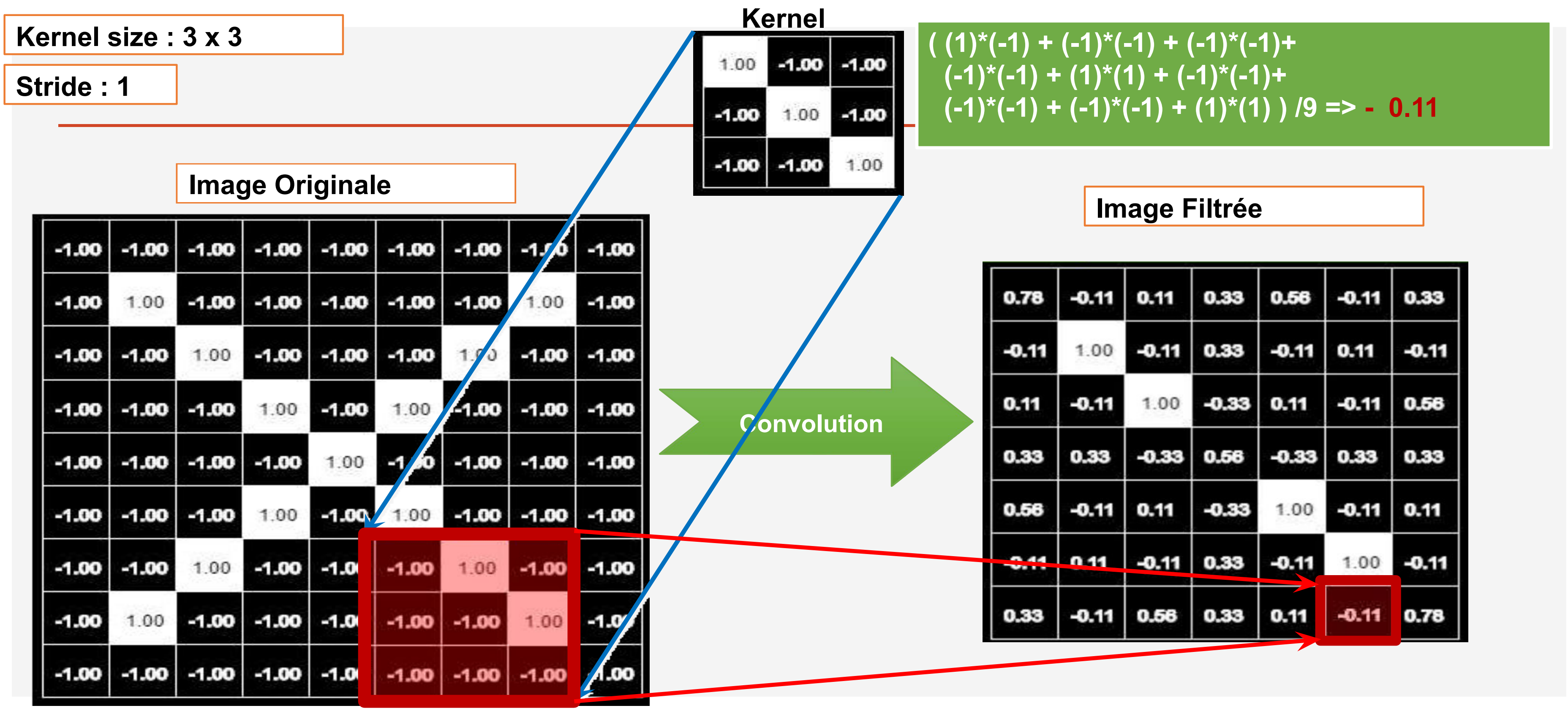




Kernel size : 3 x 3
Stride : 1
Kernel
1.00 -1.00 -1.00
-1.00 1.00 -1.00
-1.00 -1.00 1.00
((1)*(-1) + (-1)*(-1) + (-1)*(-1)+
(-1)*(-1) + (1)*(1) + (-1)*(-1)+
(-1)*(-1) + (-1)*(-1) + (1)*(1)) /9 => 0.33
Image Originale
-1.00 -1.00 -1.00 -1.00 -1.00 -1.00 -1.00 -1.00 -1.00
-1.00 1.00 -1.00 -1.00 -1.00 -1.00 -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)*(-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 => - 0.11
Image Originale
-1.00 -1.00 -1.00 -1.00 -1.00 -1.00 -1.00 -1.00 -1.00
-1.00 1.00 -1.00 -1.00 -1.00 -1.00 -1.00 1.00 -1.00
-1.00 -1.00 1.00 -1.00 -1.00 -1.00 1.00 -1.00 -1.00
-1.00 -1.00 -1.00 1.00 -1.00 1.00 -1.00 -1.00 -1.00
-1.00 -1.00 -1.00 -1.00 1.00 -1.00 -1.00 -1.00 -1.00
-1.00 -1.00 -1.00 1.00 -1.00 1.00 -1.00 -1.00 -1.00
-1.00 -1.00 1.00 -1.00 -1.00 -1.00 1.00 -1.00 -1.00
-1.00 1.00 -1.00 -1.00 -1.00 -1.00 -1.00 1.00 -1.00
-1.00 -1.00 -1.00 -1.00 -1.00 -1.00 -1.00 -1.00 -1.00
Convolution
Image Filtrée
0.78 -0.11 0.11 0.33 0.56 -0.11 0.33
-0.11 1.00 -0.11 0.33 -0.11 0.11 -0.11
0.11 -0.11 1.00 -0.33 0.11 -0.11 0.56
0.33 0.33 -0.33 0.56 -0.33 0.33 0.33
0.56 -0.11 0.11 -0.33 1.00 -0.11 0.11
-0.11 0.11 -0.11 0.33 -0.11 1.00 -0.11
0.33 -0.11 0.56 0.33 0.11 -0.11 0.78

Kernel size : 3 x 3

Stride : 1

Kernel

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

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

Image Originale

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

Convolution

Image Filtrée

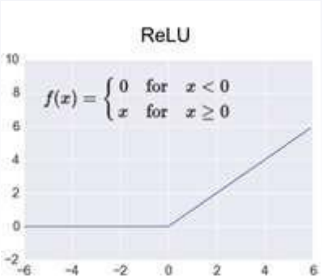
0.78	-0.11	0.11	0.33	0.56	-0.11	0.33
-0.11	1.00	-0.11	0.33	-0.11	0.11	-0.11
0.11	-0.11	1.00	-0.33	0.11	-0.11	0.56
0.33	0.33	-0.33	0.56	-0.33	0.33	0.33
0.56	-0.11	0.11	-0.33	1.00	-0.11	0.11
-0.11	-0.11	-0.11	0.33	-0.11	1.00	-0.11
0.33	-0.11	0.56	0.33	0.11	-0.11	0.78

RELU Activation Function

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

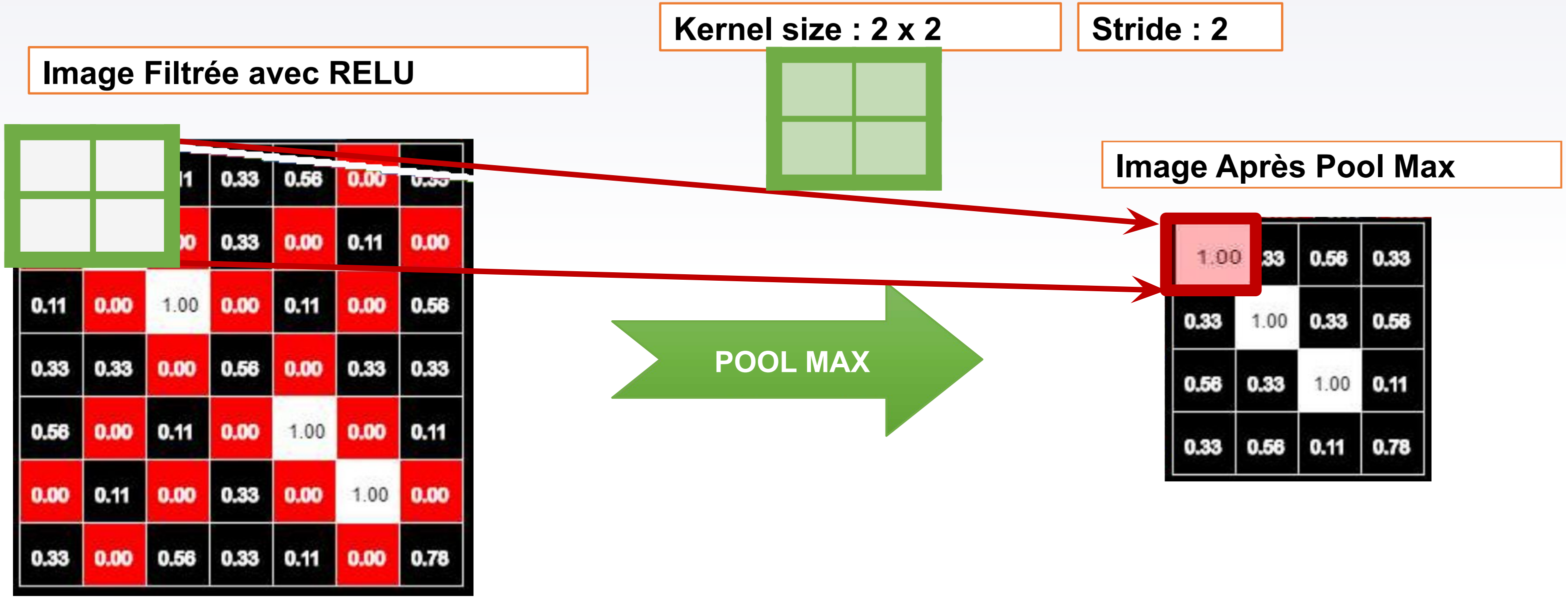


RELU Activation

Image Filtrée avec RELU

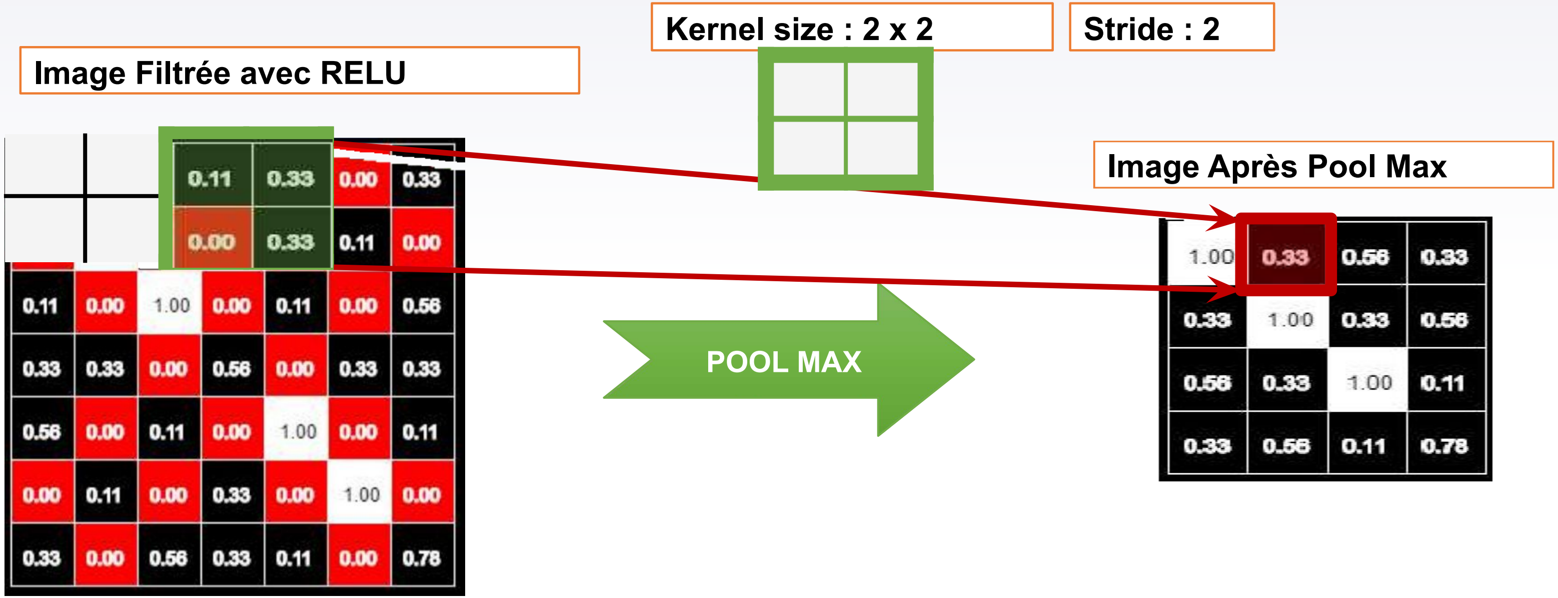
0.78	0.00	0.11	0.33	0.56	0.00	0.33
0.00	1.00	0.00	0.33	0.00	0.11	0.00
0.11	0.00	1.00	0.00	0.11	0.00	0.56
0.33	0.33	0.00	0.56	0.00	0.33	0.33
0.56	0.00	0.11	0.00	1.00	0.00	0.11
0.00	0.11	0.00	0.33	0.00	1.00	0.00
0.33	0.00	0.56	0.33	0.11	0.00	0.78

POOLING Layer (POOL MAX)



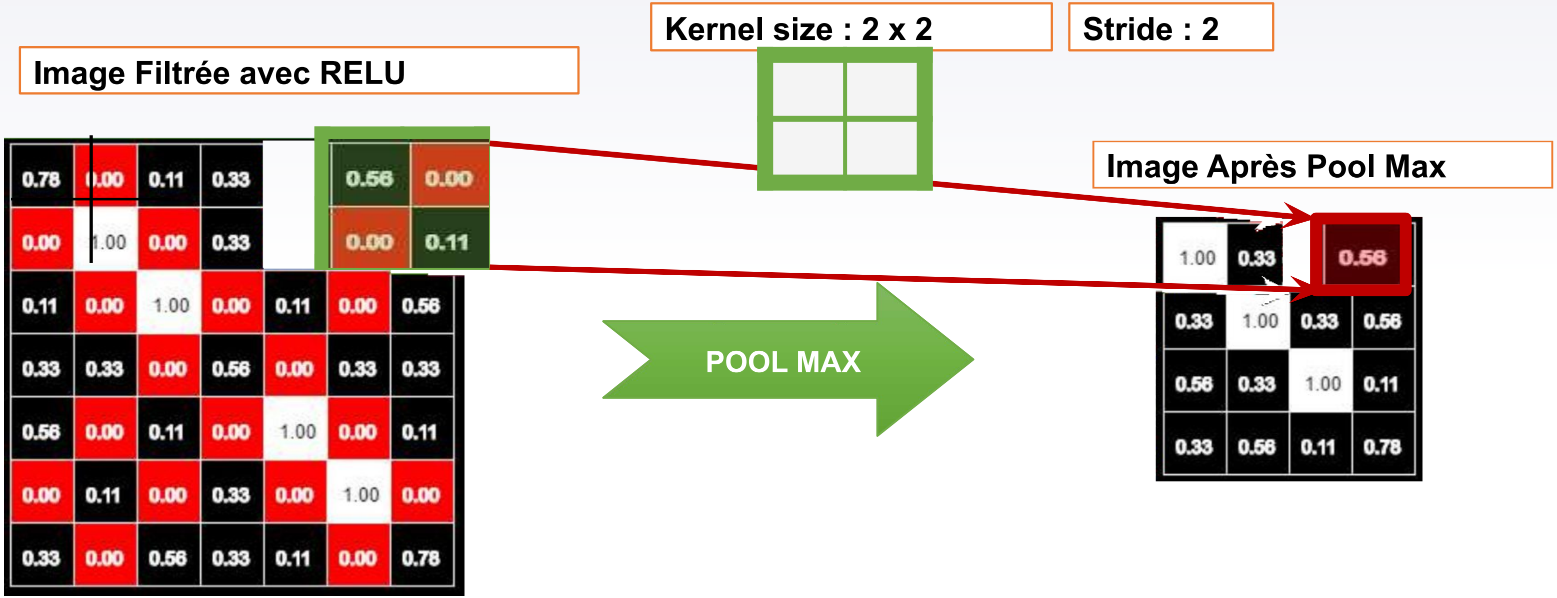
POOLING (POOL MAX)

POOLING (POOL MAX)



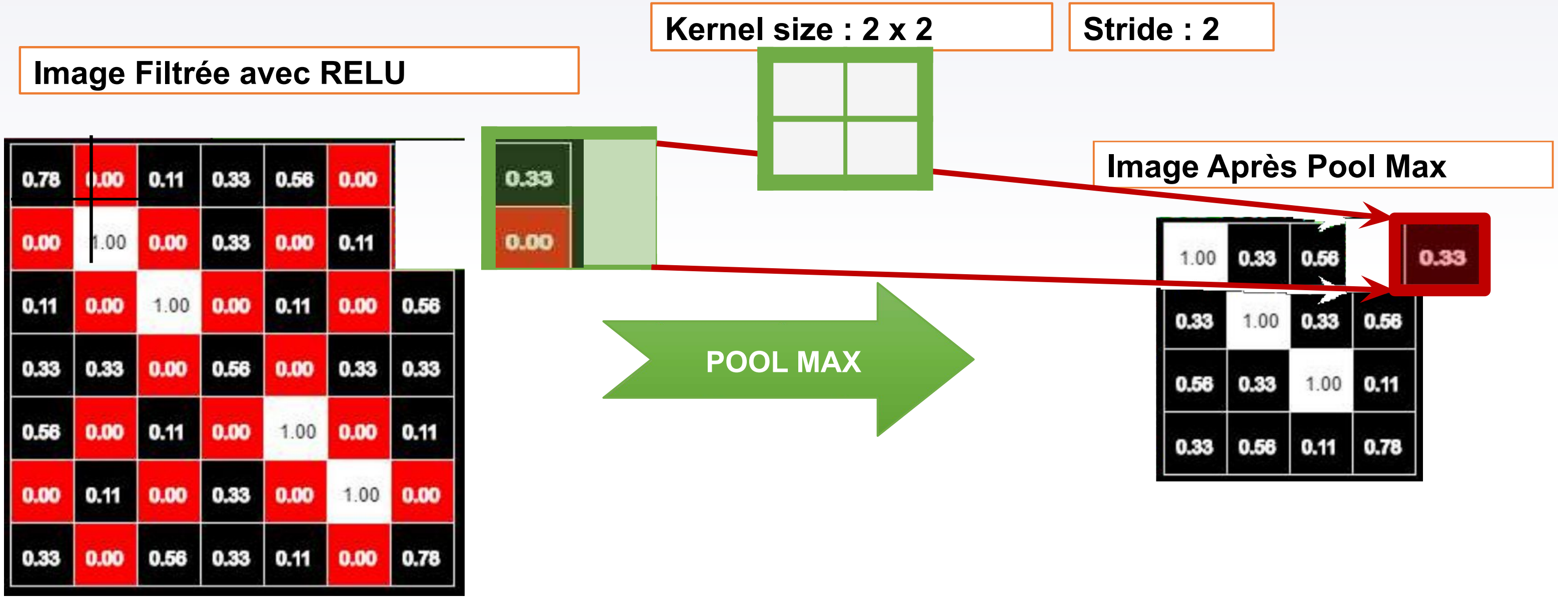
POOLING (POOL MAX)

POOLING (POOL MAX)

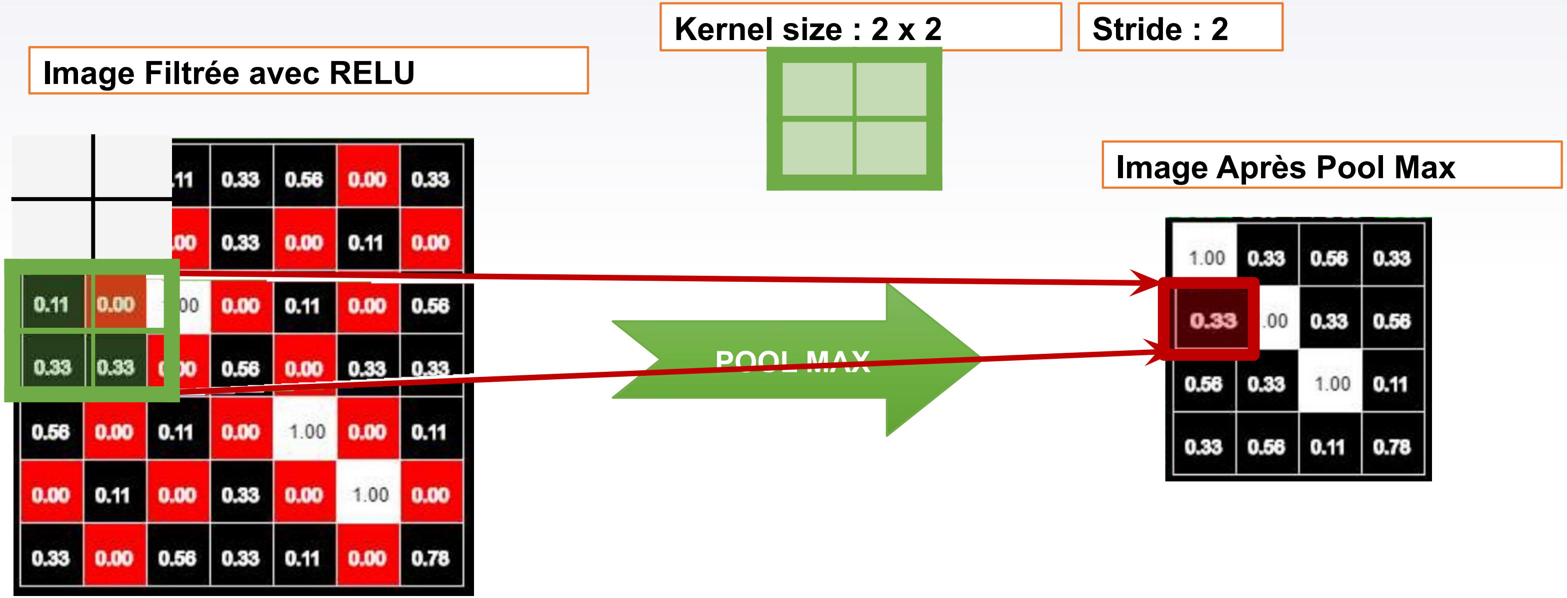


POOLING (POOL MAX)

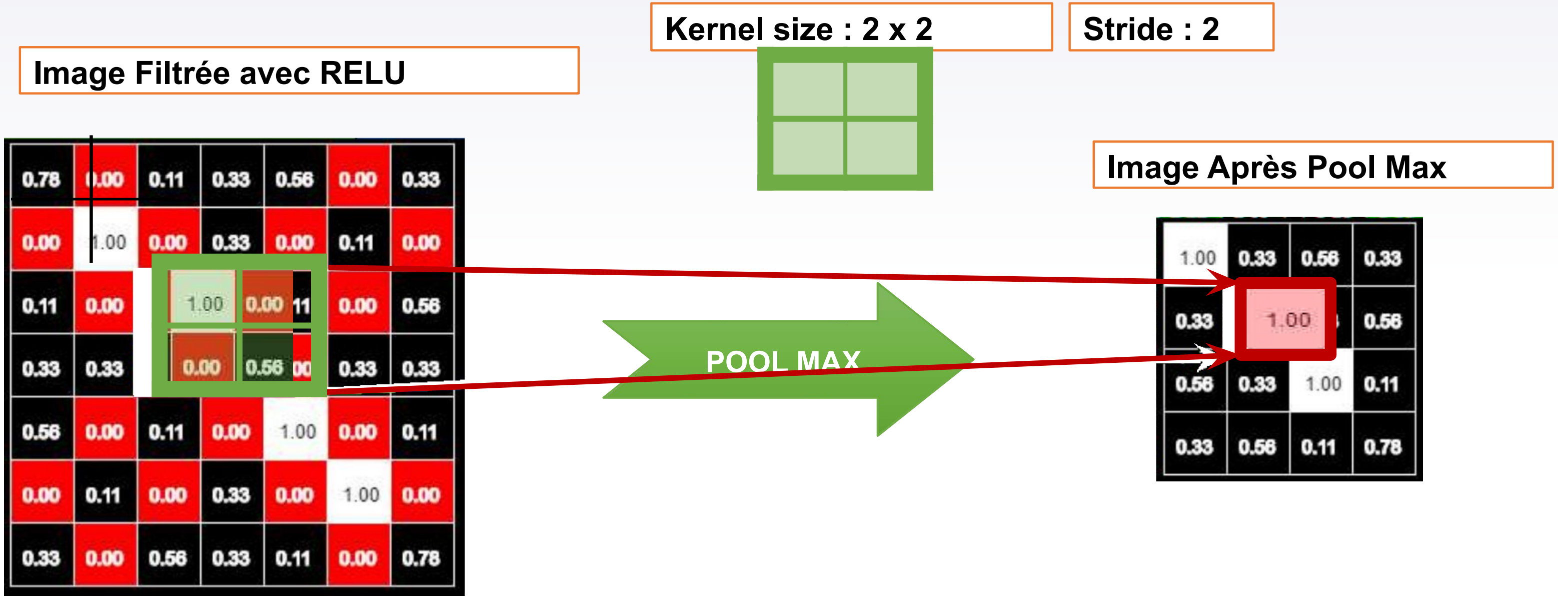
POOLING (POOL MAX)



POOLING (POOL MAX)

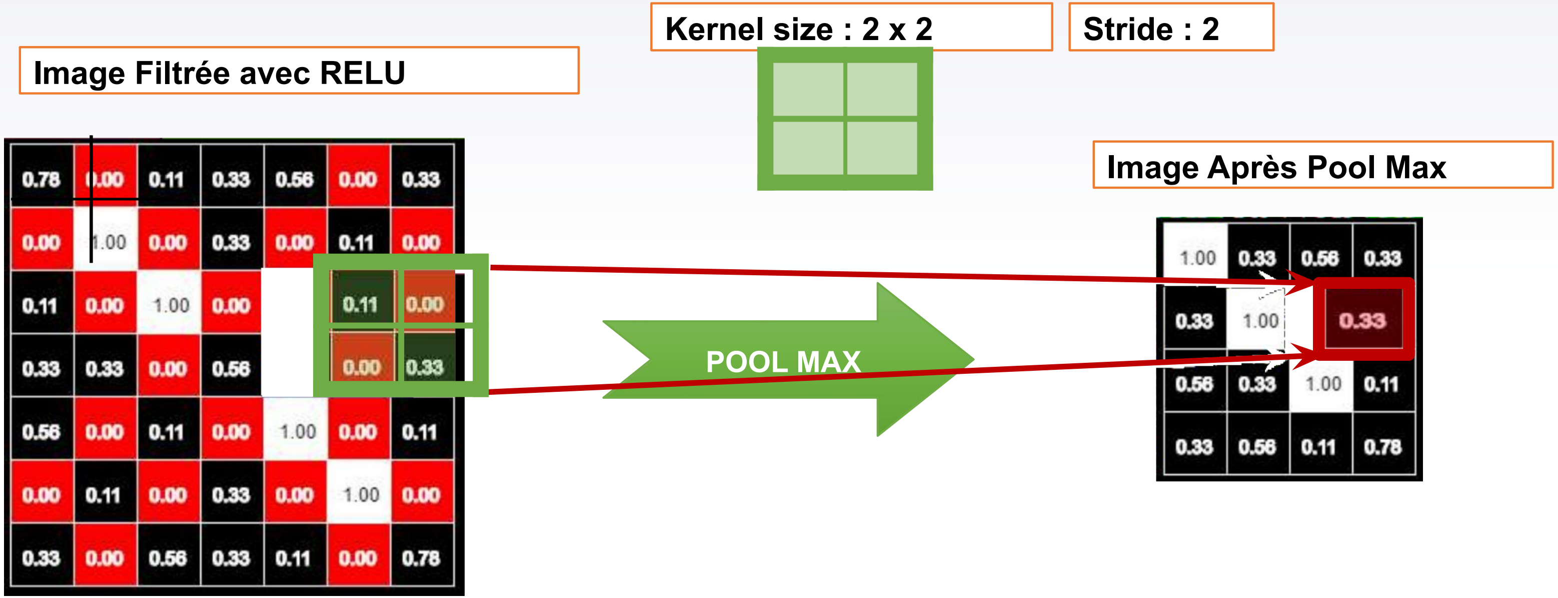


POOLING (POOL MAX)
POOLING (POOL MAX)



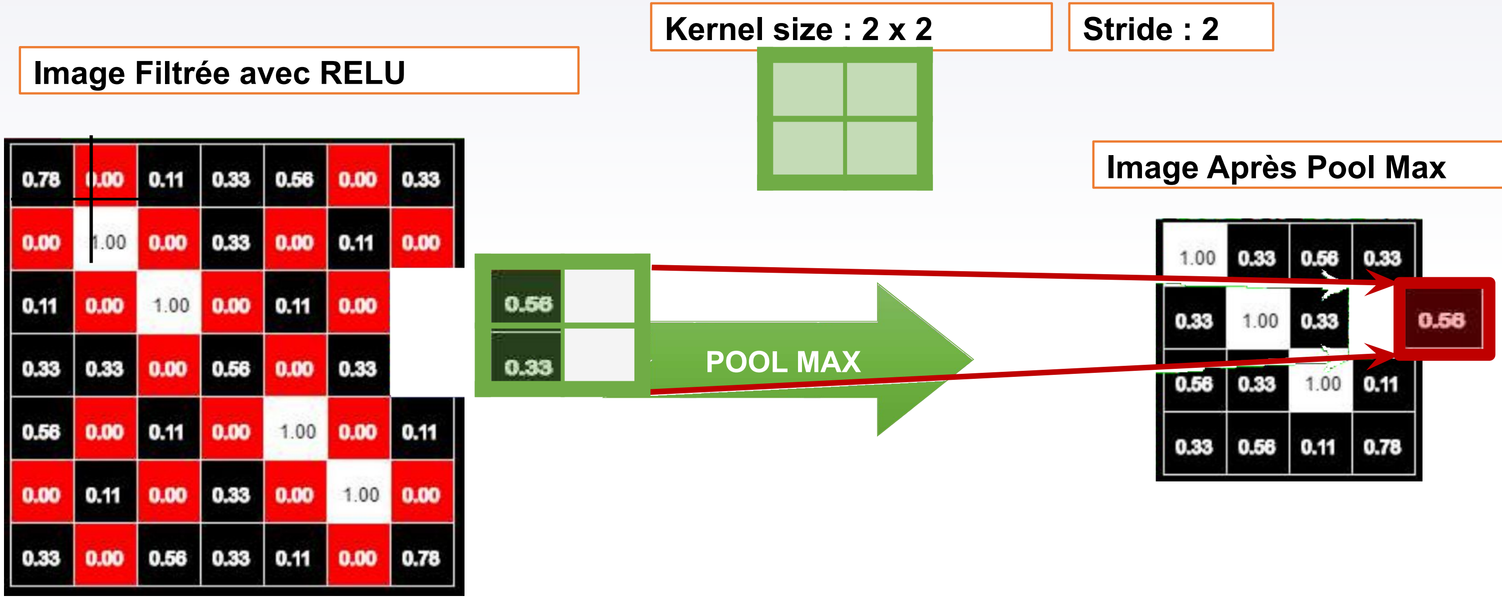
POOLING (POOL MAX)

POOLING (POOL MAX)



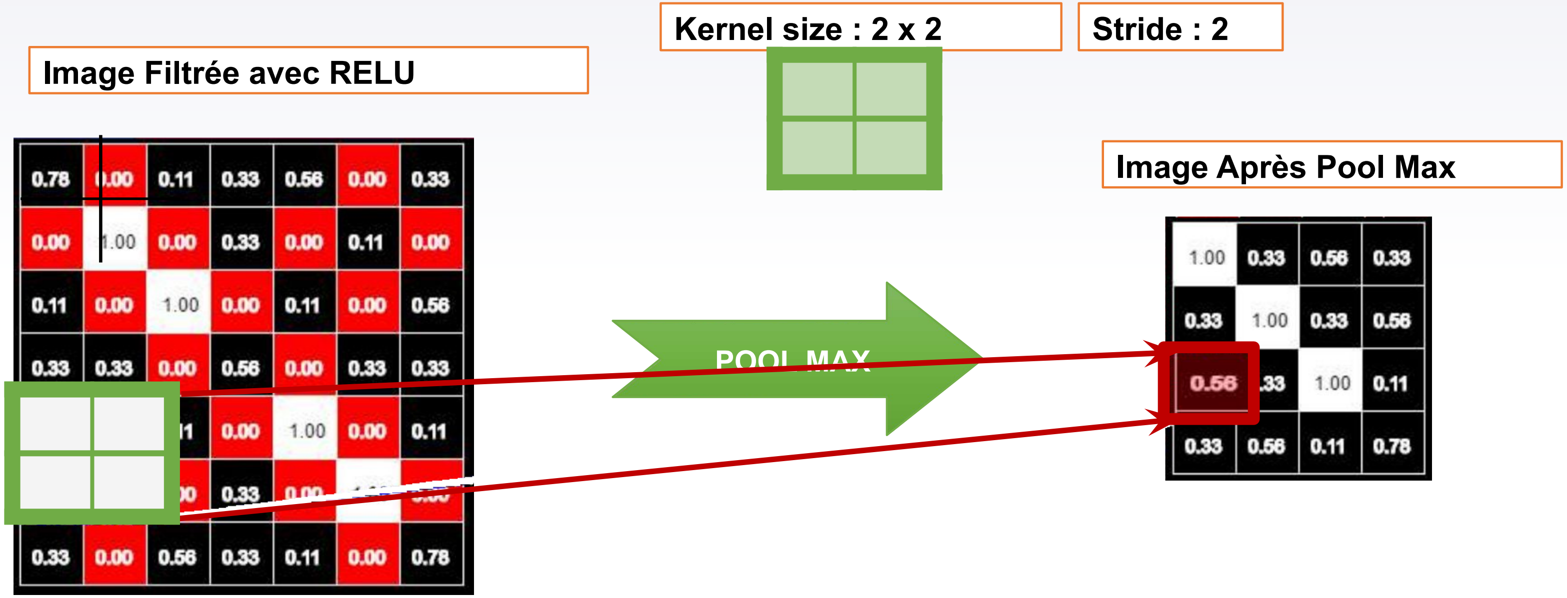
POOLING (POOL MAX)

POOLING (POOL MAX)



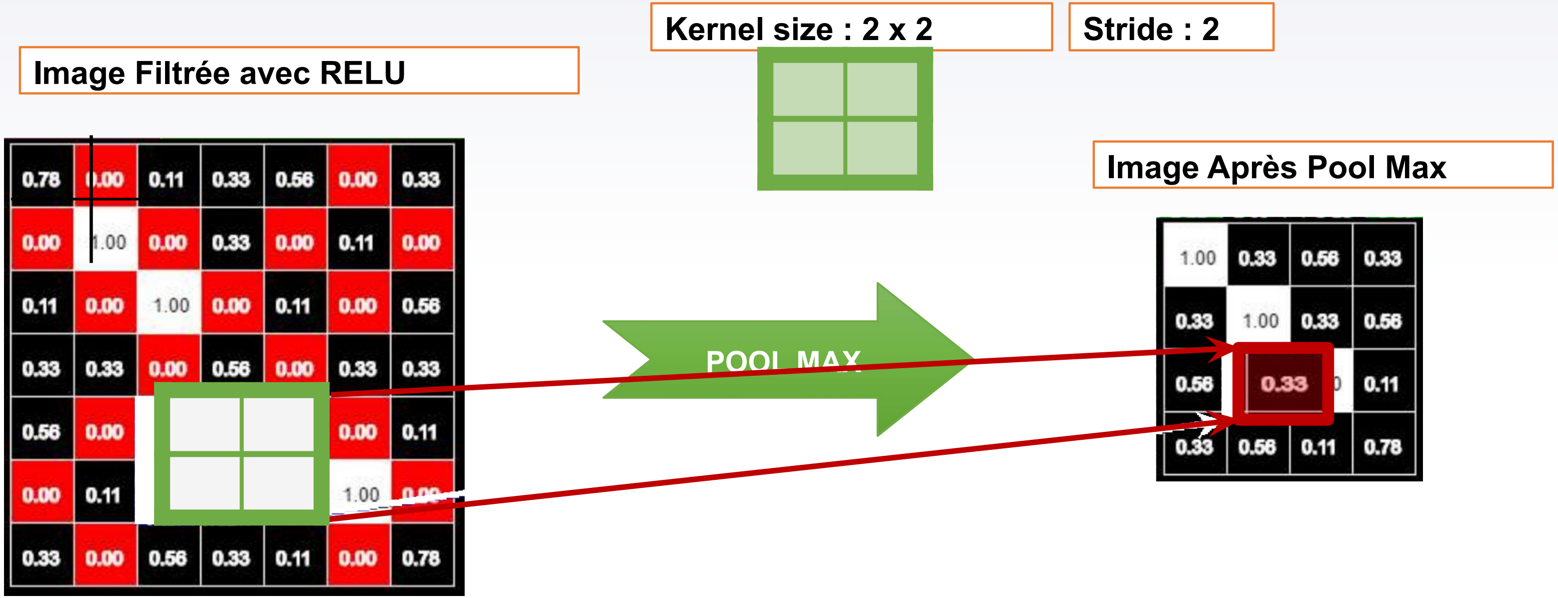
POOLING (POOL MAX)

POOLING (POOL MAX)



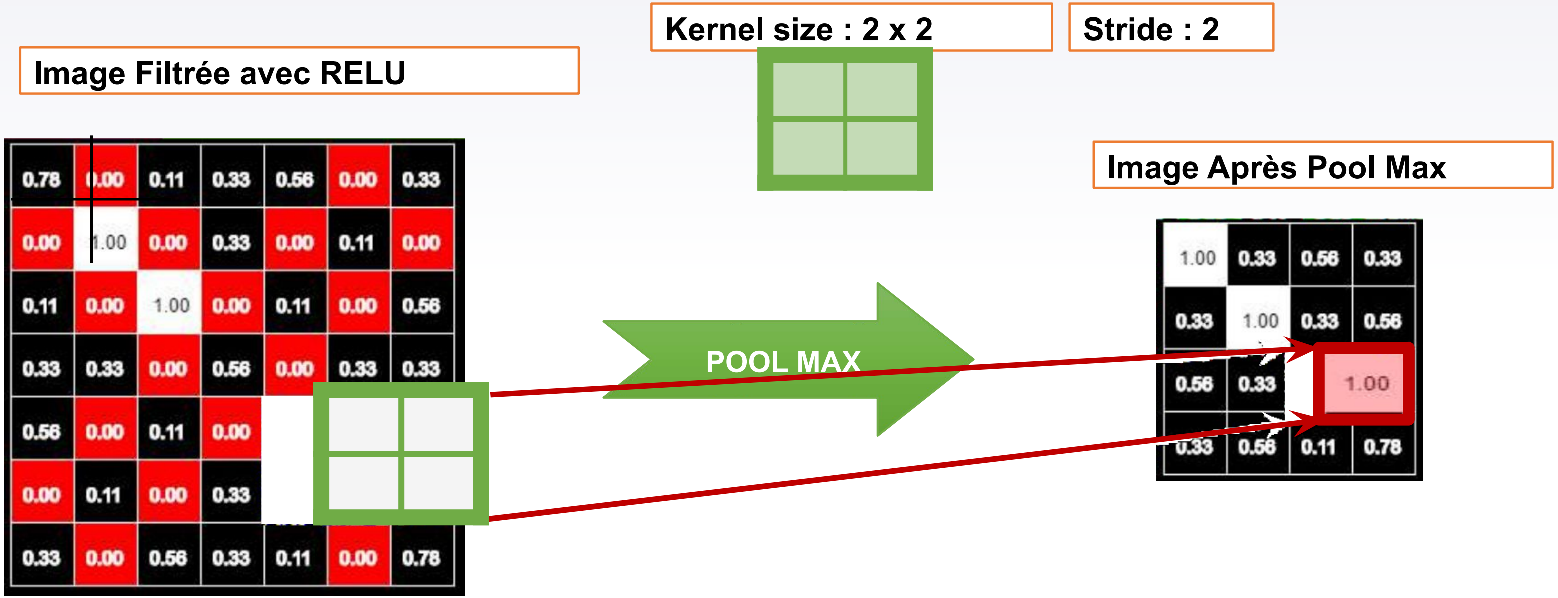
POOLING (POOL MAX)

POOLING (POOL MAX)



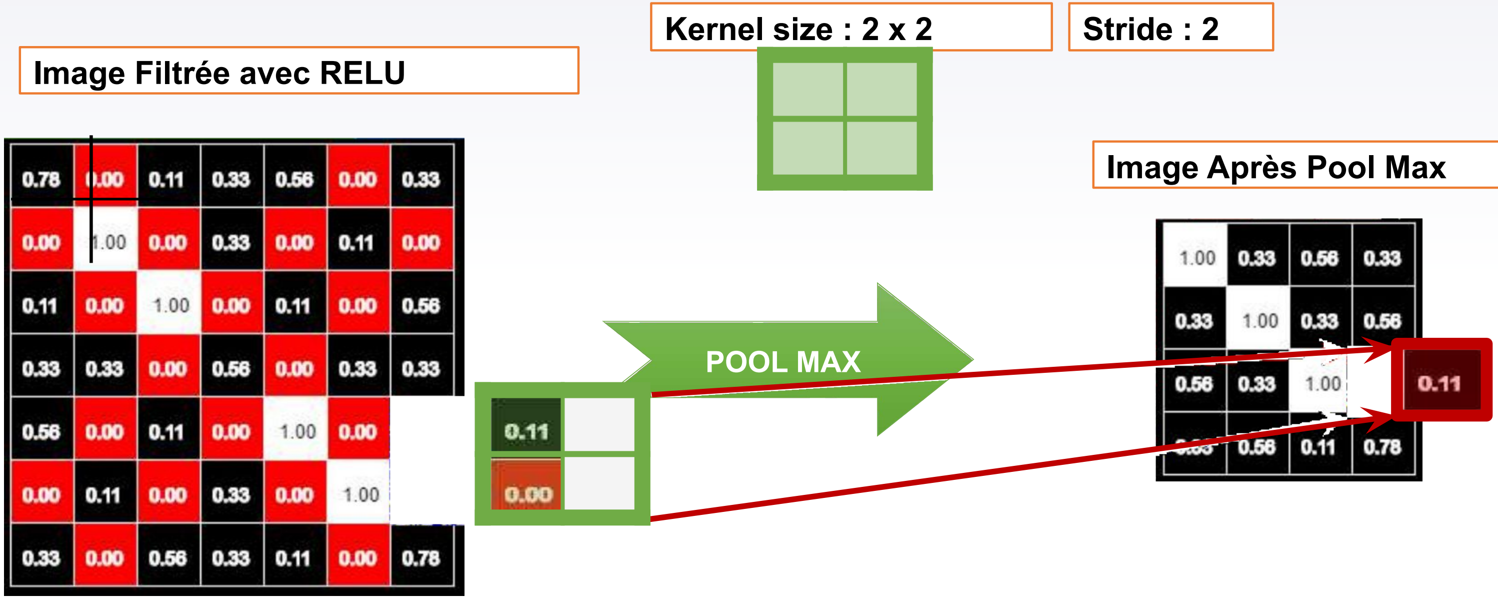
POOLING (POOL MAX)

POOLING (POOl MAX)



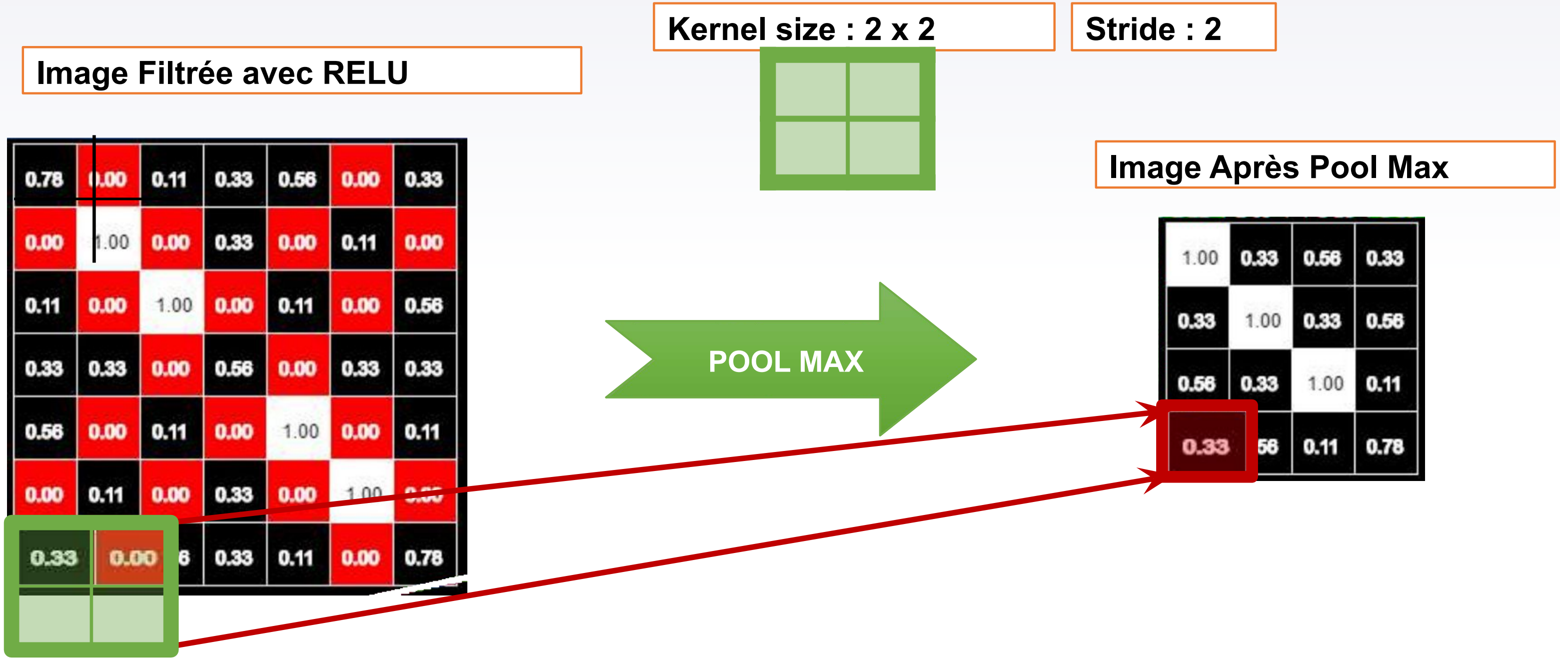
POOLING (POOL MAX)

POOLING (POOL MAX)



POOLING (POOL MAX)

POOLING (POOL MAX)



POOLING (POOL MAX)

POOLING (POOL MAX)

Kernel size : 2 x 2

Stride : 2

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		0.00	0.78

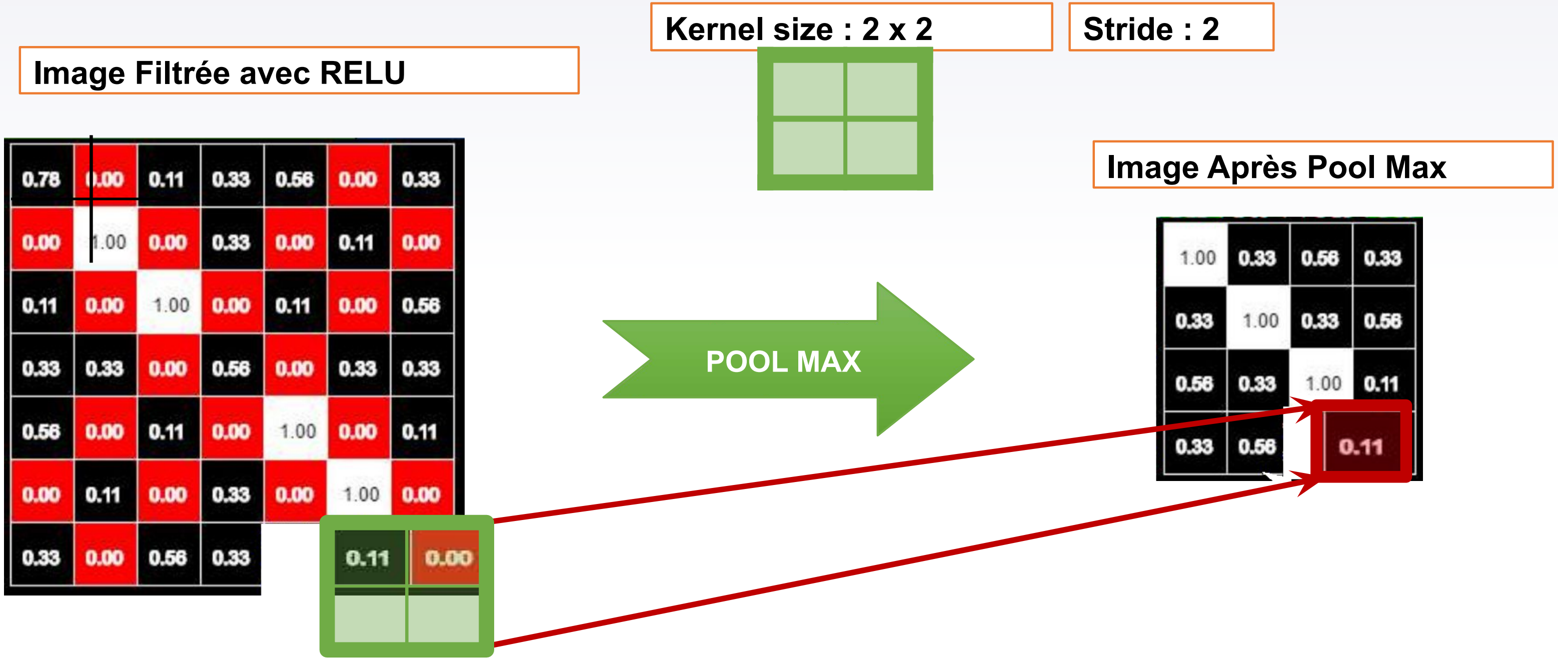
POOL MAX

Image Après Pool Max

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

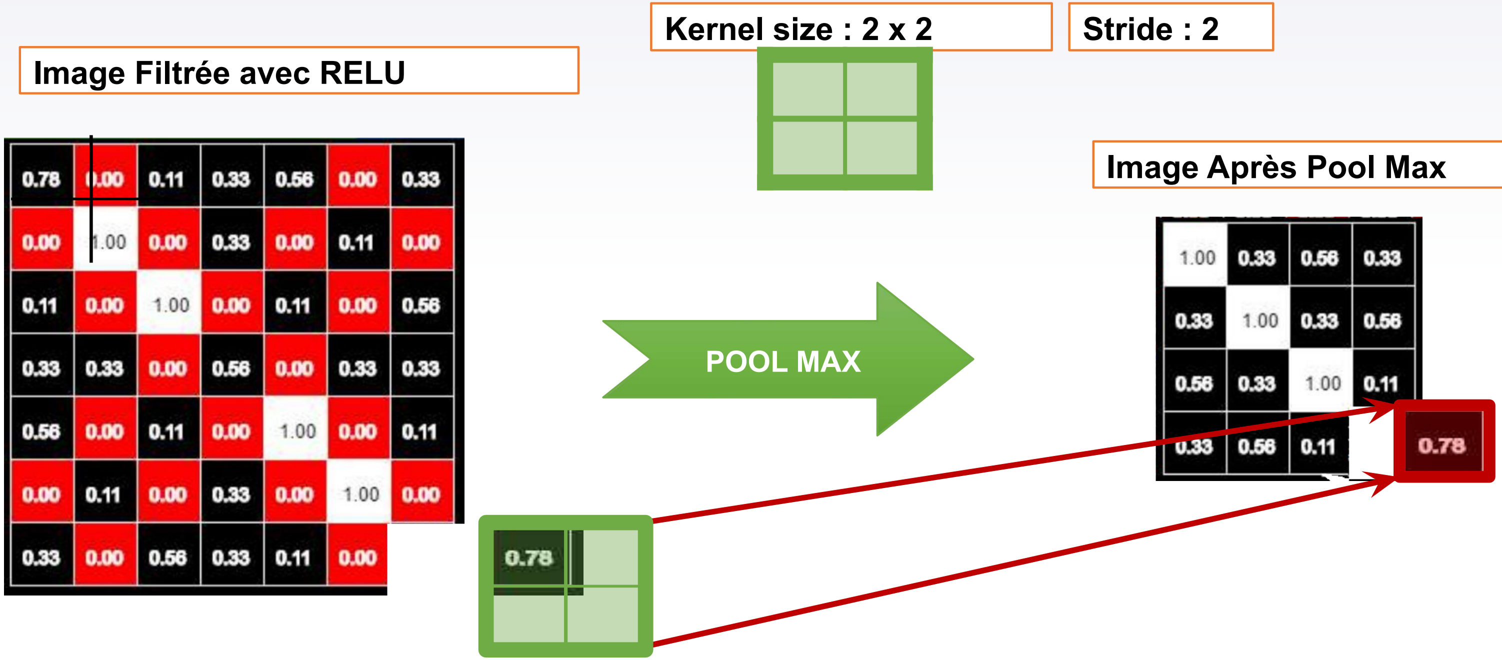
POOLING (POOL MAX)

POOLING (POOL MAX)

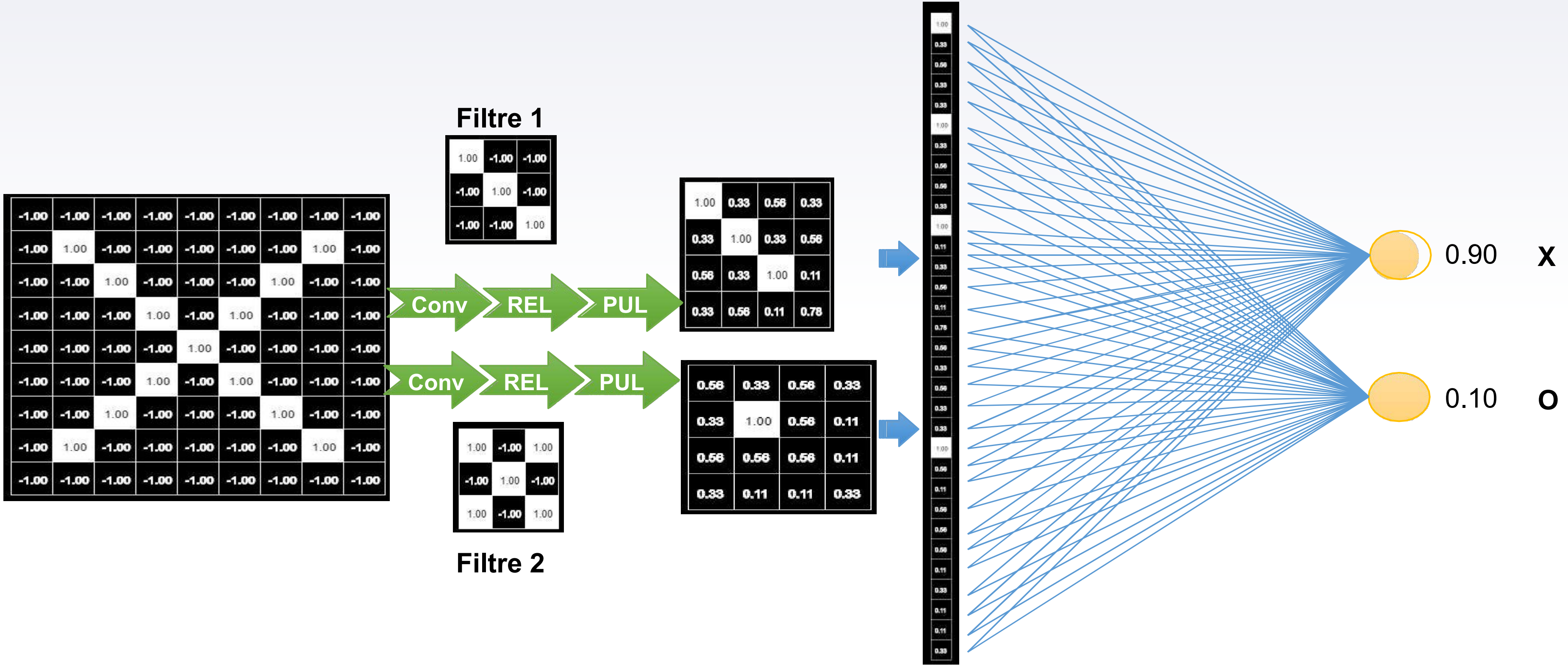


POOLING (POOL MAX)

POOLING (POOL MAX)



Fully Connected Layer (Dense Layer)



Datasets publics et privés!

- Prise de contact avec un laboratoire nantais (CrohnIPI) ayant une base de données contenant 4000 images x 2 maladies (en cours).
- Chargement d'une base de données fournie par un laboratoire norvégien kvasir (1,2 Go/23,5 Go/8 maladies)

CrohnIPI: An endoscopic image database for the evaluation of automatic Crohn's disease lesions recognition algorithms

R. Vallée[a], A. De Maissin[b], A. Coutrot[a], H. Mouchère[a], A. Bourceille[b], and N. Normand[a]

[a]Université de Nantes, CNRS, LS2N, F-44000 Nantes, France
[b]CHU Nantes, Institut des Maladies de l'Appareil Digestif, CIC Inserm 1413, Université de Nantes, Nantes, France

ABSTRACT

Wireless capsule endoscopy (WCE) allows medical doctors to examine the interior of the small intestine with a noninvasive procedure. This methodology is particularly important for Crohn's disease (CD), where an early diagnosis improves treatment outcomes. The counting and identification of CD lesions in WCE videos is a time-consuming process for medical experts. In the era of deep-learning many automatic WCE lesion classifiers, requiring annotated data, have been developed. However, benchmarking classifiers is difficult due to the lack of standard evaluation data. Most detection algorithms are evaluated on private datasets or on unspecified subsets of public databases. To help the development and comparison of automatic CD lesion classifiers, we release CrohnIPI, a dataset of 3498 images, independently reviewed by several experts. It contains 60.55% of non-pathological images and 38.85% of pathological images with 7 different types of CD lesions. A part of these images are multilabeled. The dataset is balanced between pathological images and non-pathological ones and split into two subsets for training and testing models. This database will progressively be enriched over the next few years in aim to make the automatic detection algorithms converge to the most accurate system possible and to consolidate their evaluation.

Keywords: Dataset, Video Capsule Endoscopic, Deep learning, Crohn's disease

simula

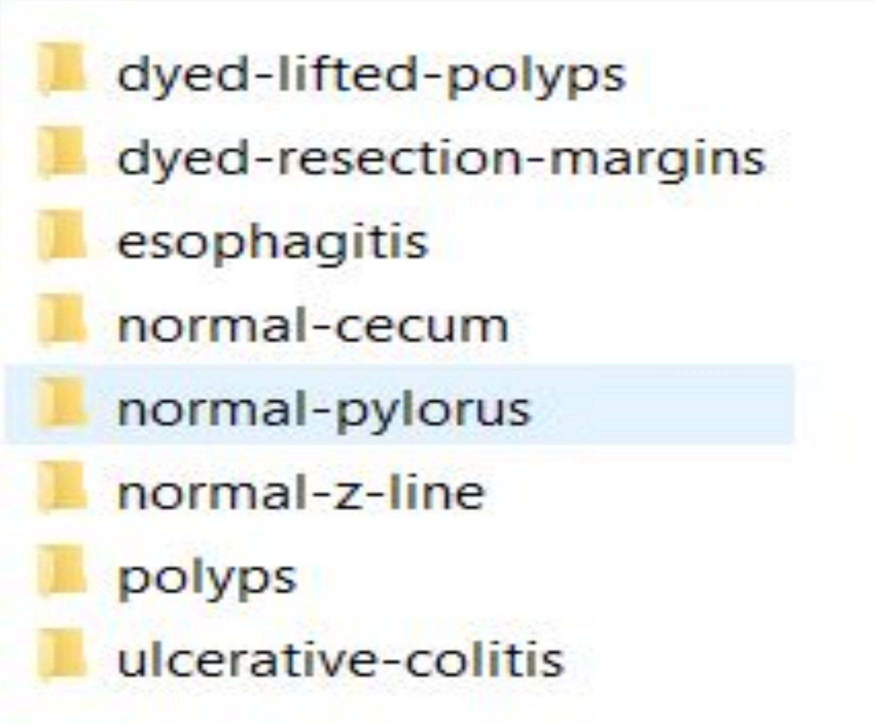
Kvasir

A Multi-Class Image-Dataset for Computer Aided Gastrointestinal Disease Detection.

Automatic detection of diseases by use of computers is an important, but still unexplored field of research. Such innovations may improve medical practice and refine health care systems all over the world. However, datasets containing medical images are hardly available, making reproducibility and comparison of approaches almost impossible. Here, we present Kvasir, a dataset containing images from inside the gastrointestinal (GI) tract. The collection of images are classified into three important anatomical landmarks an three clinically significant findings. In addition, it contains two categories of images related to endoscopic polyp removal. Sorting and annotation of the dataset is performed by medical doctors (ex- perienced endoscopists). In this respect, Kvasir is important for researc on both single- and multi-disease computer aided detec- tion. By providing it, we invite and enable multimedia researcher into the medical domain of detection and retrieval.

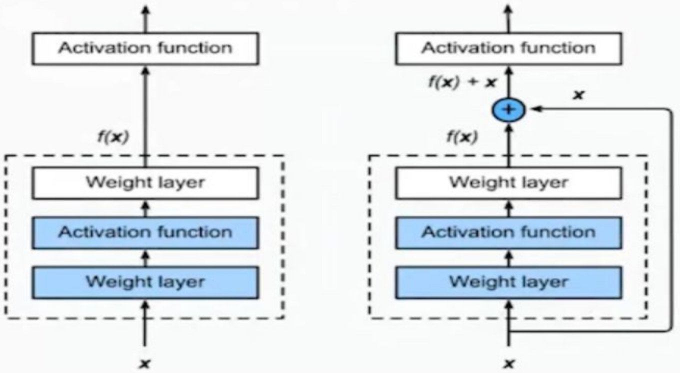
The Kvasir-SEG Dataset

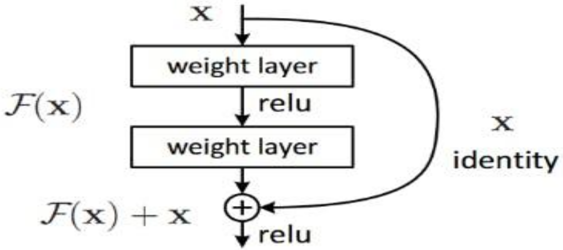
- The ***kvasir-dataset.zip (size 1.2 GB)*** archive contains 4,000 images, 8 classes, 500 images for each class. The images are stored in the separate folders named accordingly to the name of the class images belongs to. The image files are encoded using JPEG compression. The encoding settings can vary across the dataset and they reflecting the a priori unknown endoscopic equipment settings. The extension of the image files is ".jpg".

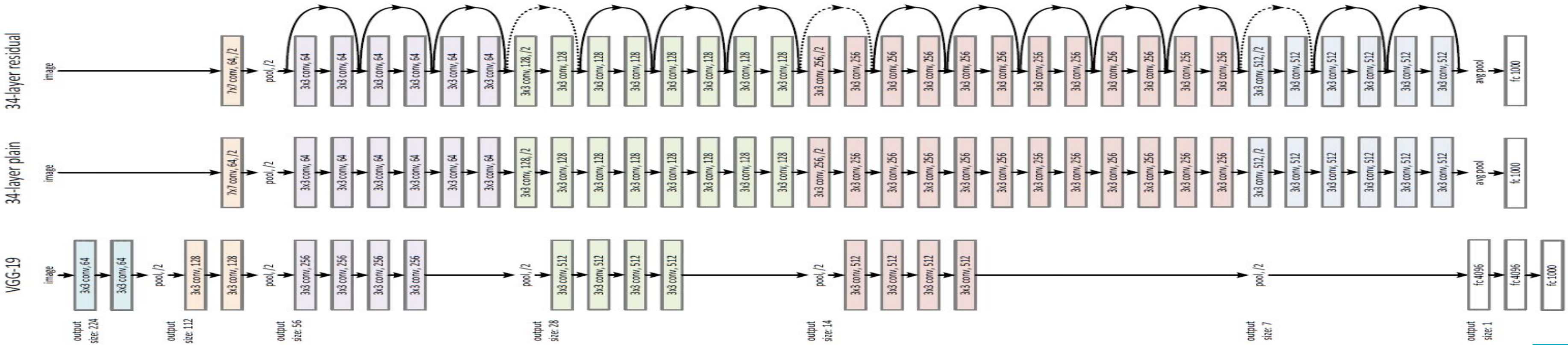


Architecture Resnet

- ResNet is one of the machine learning models that use skip connections between layers.
- ResNet was introduced in 2015 and increases the accuracy.
- Skip connections help in reducing the number of layers.
- $x_{t+1} = x_t + f(x_t)$







Resnet 50

```
to_pil = transforms.ToPILImage()
images, labels = get_random_images(8)
classes=trainloader.dataset.classes
results=[]
fig=plt.figure(figsize=(20,20))
for ii in range(len(images)):
    image = to_pil(images[ii])
    index = predict_image(image)
    sub = fig.add_subplot(1, len(images), ii+1)
    res = int(labels[ii]) == index
    results.append(res)
    sub.set_title(str(classes[index]) + ":" + str(res))
    plt.axis('off')
    plt.imshow(image)
```

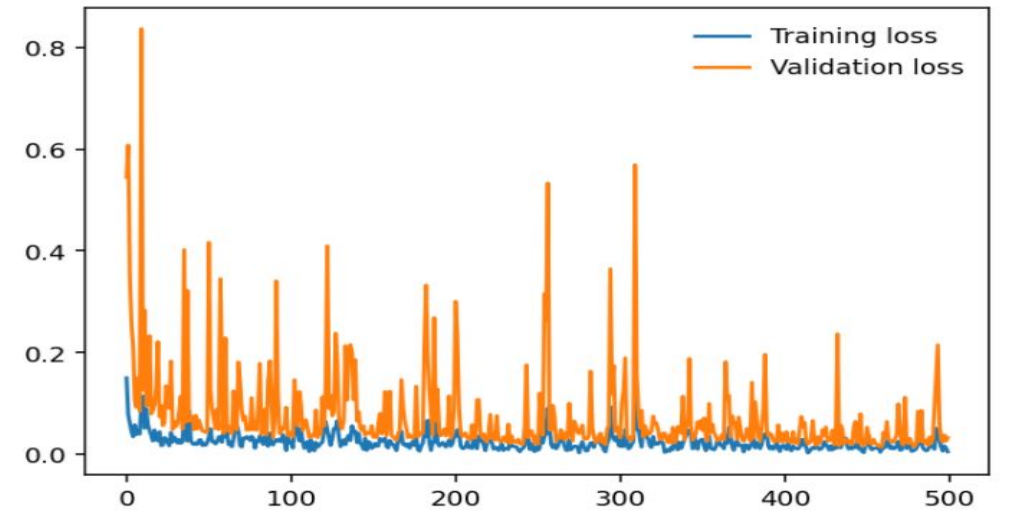
```
def predict_image(image):
    image_tensor = test_transforms(image).float()
    image_tensor = image_tensor.unsqueeze_(0)
    #input = Variable(image_tensor)
    image_tensor = image_tensor.to(device)
    output = model(image_tensor)
    #print(output)
    _, predicted = torch.max(output.data, 1)
    print(predicted)
    print(_)
    #print(torch.exp(output.data.cpu()))
    index = output.data.cpu().numpy().argmax()
    return index
```

```
[True, True, True, True, True, True, True, True]
```

Resnet 50

The negative log-likelihood loss:

$$\text{loss}(x, y) = -(\log y)$$

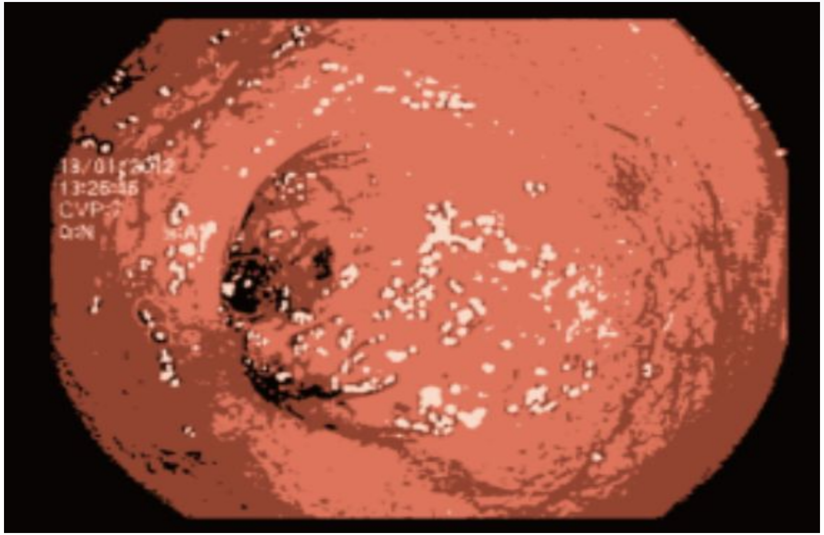


`Epoch 100/100.. Train loss: 0.028.. Test loss: 0.033.. Test accuracy: 0.990`

Kmeans on original image

Original image

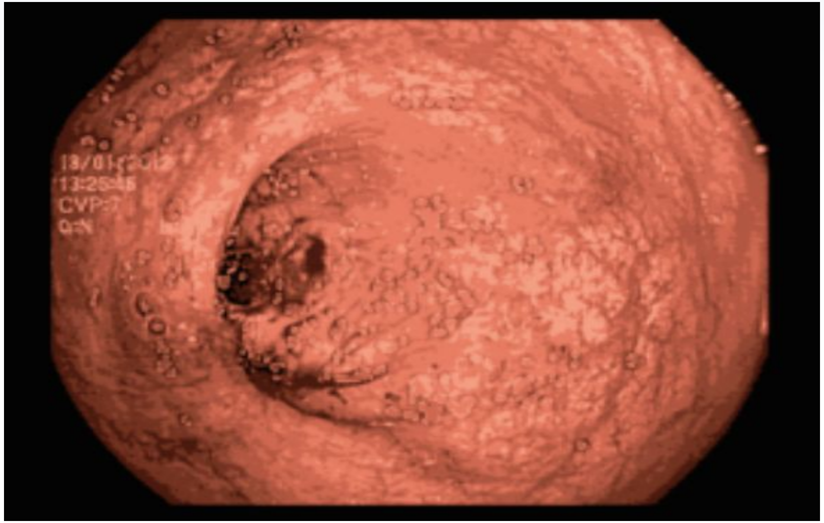
Kmeans applied



Kmeans on original image

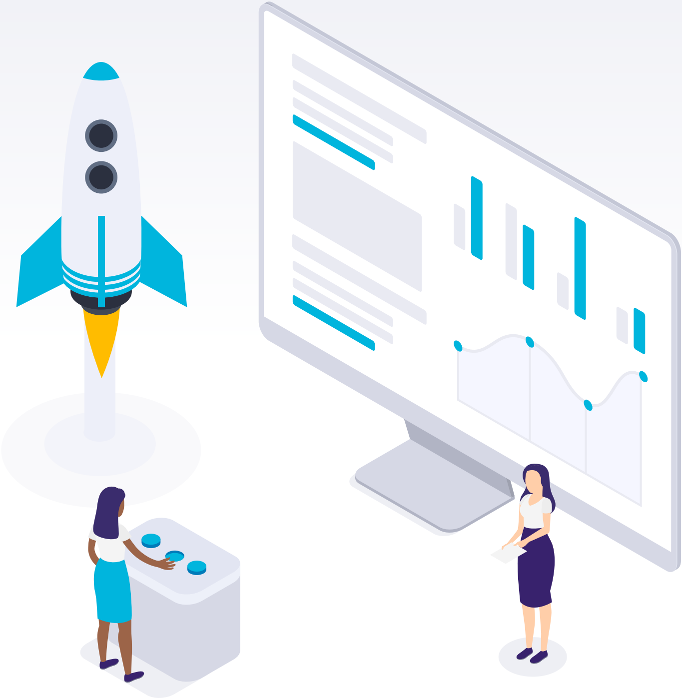
Inpainted original image

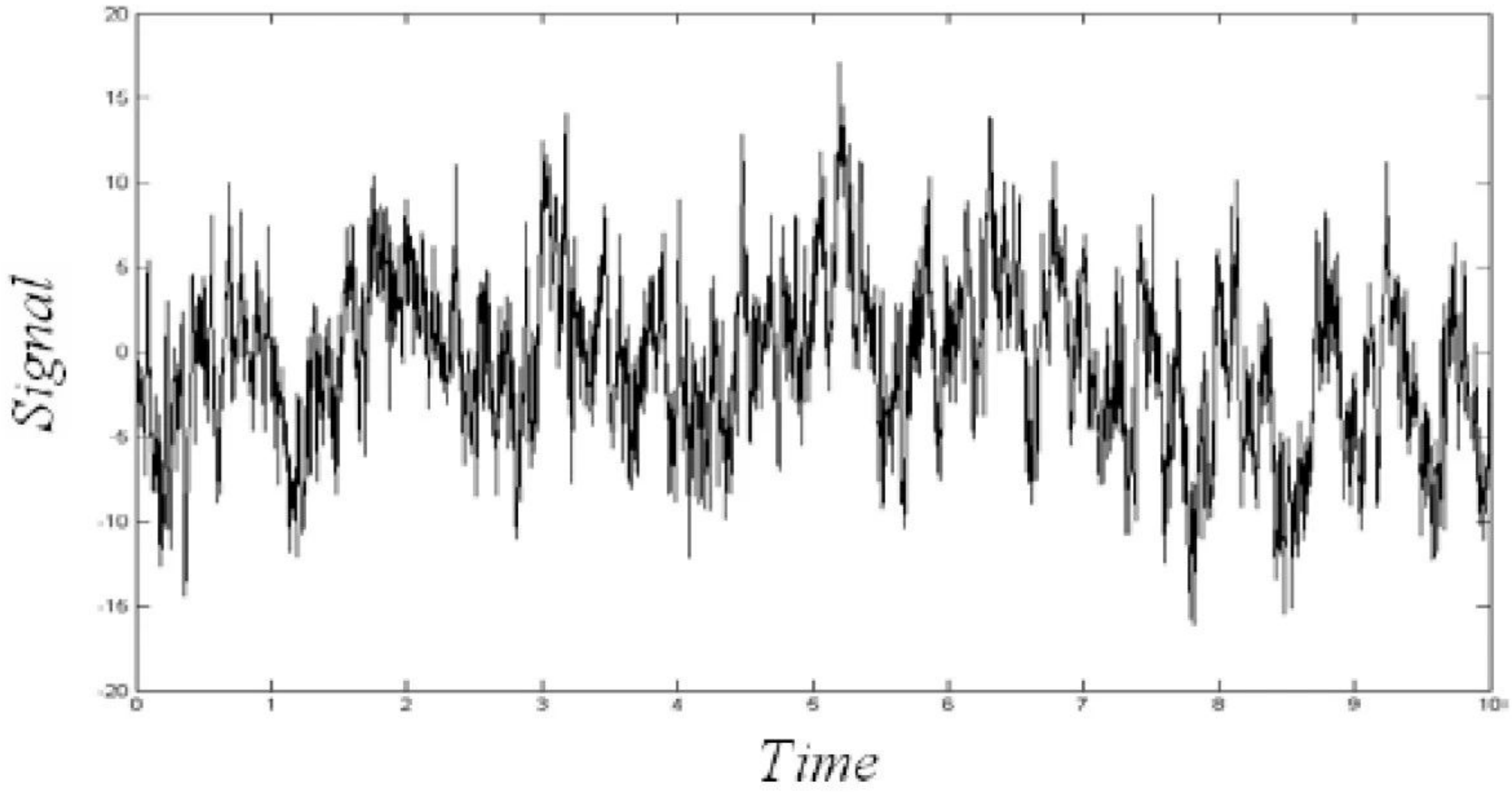
Kmeans applied on inpainting

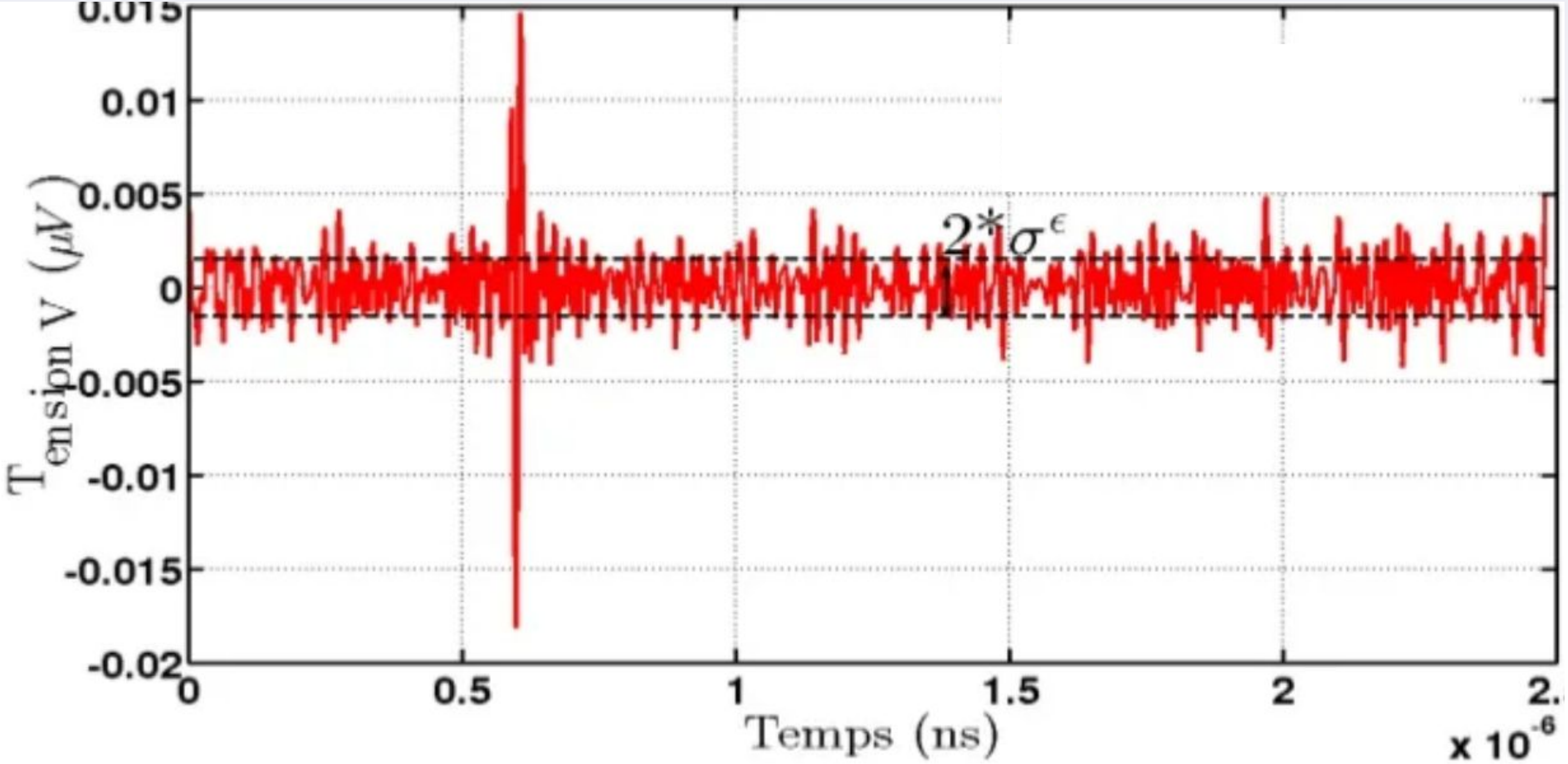


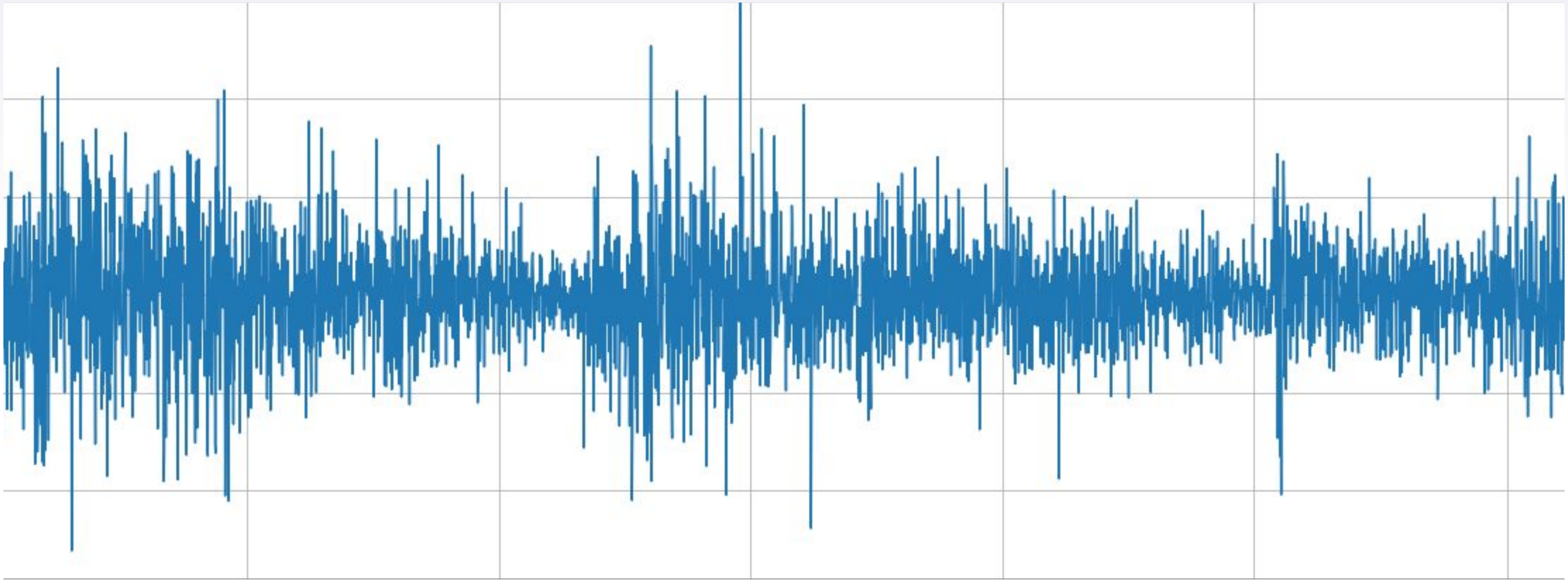
2

Classification des signaux EEG et détection des états de stress




20
15
10
5
0
-5
-10
-15
-20
0
1
2
3
4
5
6
7
8
9
10
Signal
Time

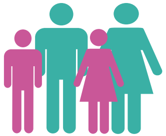

0.015
0.01
0.005
0
-0.005
-0.01
-0.015
-0.02
Tension V (μV)
$2^*\sigma^\epsilon$
0
0.5
1
1.5
2
2.
Temps (ns)
x 10^{-6}



✔ OMS 2018

L'épilepsie

50 millions



Dans le monde

Chaque année

2,4 millions

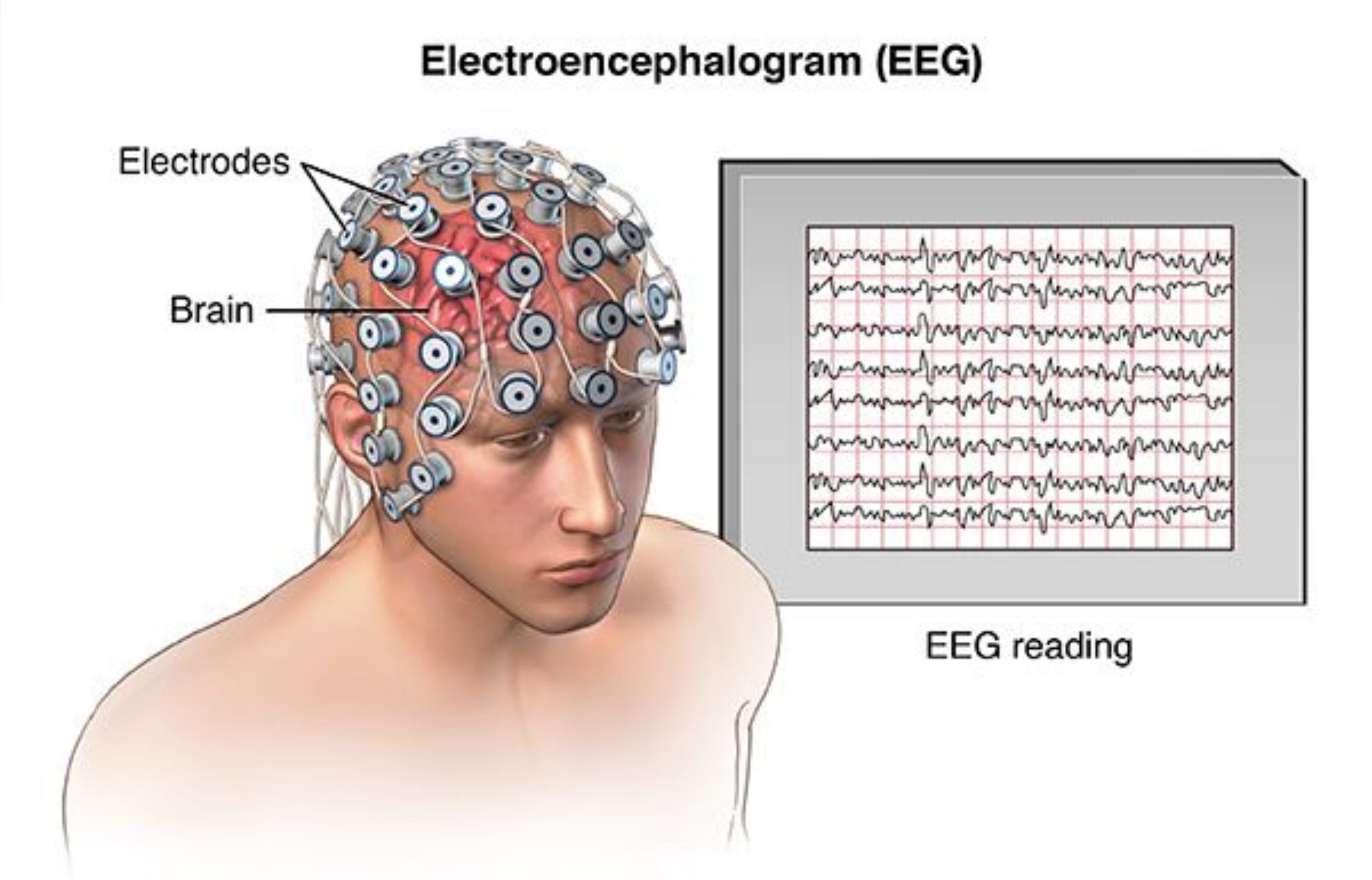
Problématique

Perte du temps

Solution proposée

Le développement d’un système automatisé pour aider les neurologues à classifier les signaux du cerveau EEG

Electroencéphalogramme



Langage de programmation

Outils et environnements

Bibliothèques utilisées

Compréhension du métier

- ✔ De nombreuses applications médicales nécessitent la détection rapide et précise des crises

- ✔ l’EEG associé à une crise chez un patient peut ressembler à un schéma bénin pour autre patient

- ✔ Une petite quantité de données sur l’épilepsie disponibles en raison de la fréquence des crises

Objectifs Data science

Classifier les profils épileptiques
et non épileptiques

Classifier les phases:
normal - intecrise - crise

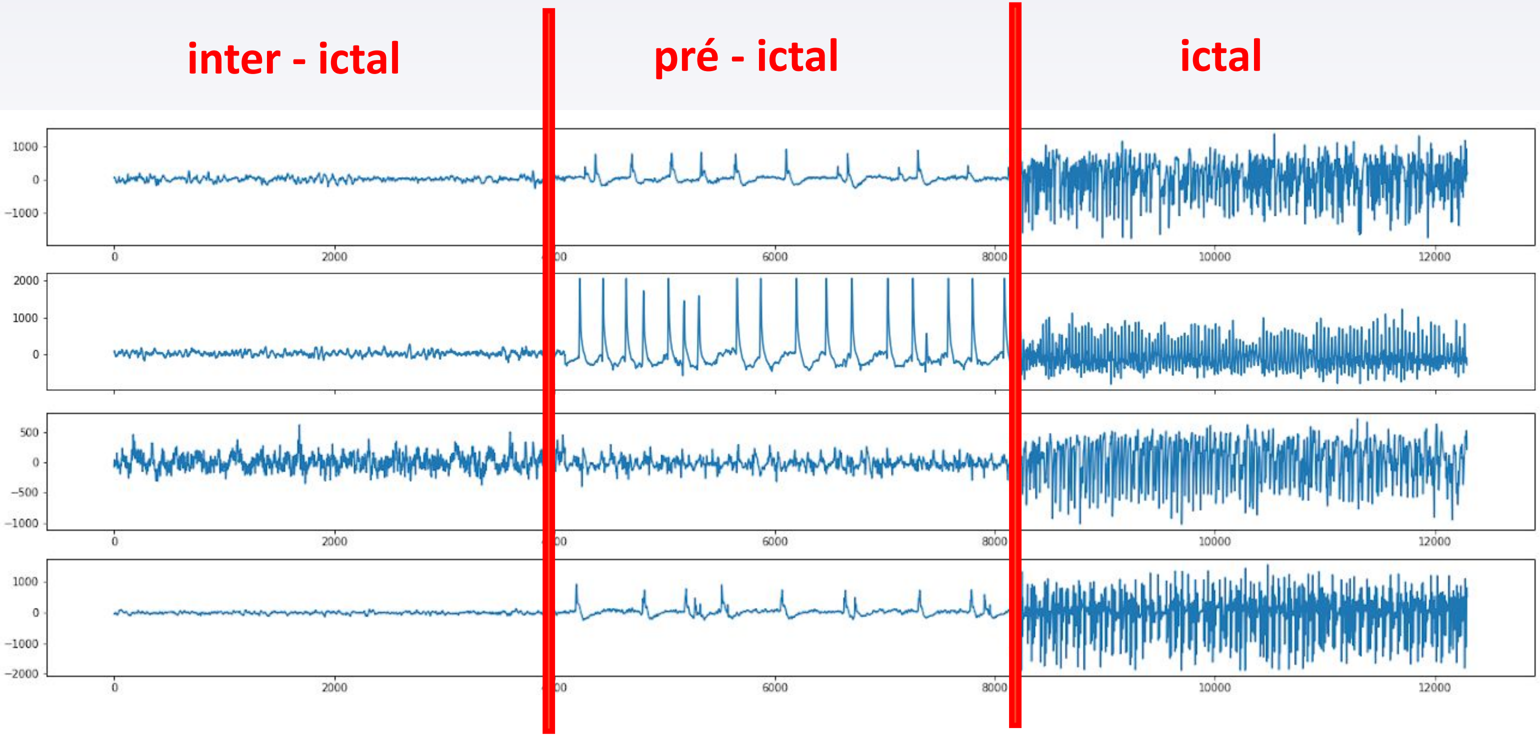
Collecte des données

- L’ensemble de données sur l’épilepsie de l’Université de Bonn en Allemagne

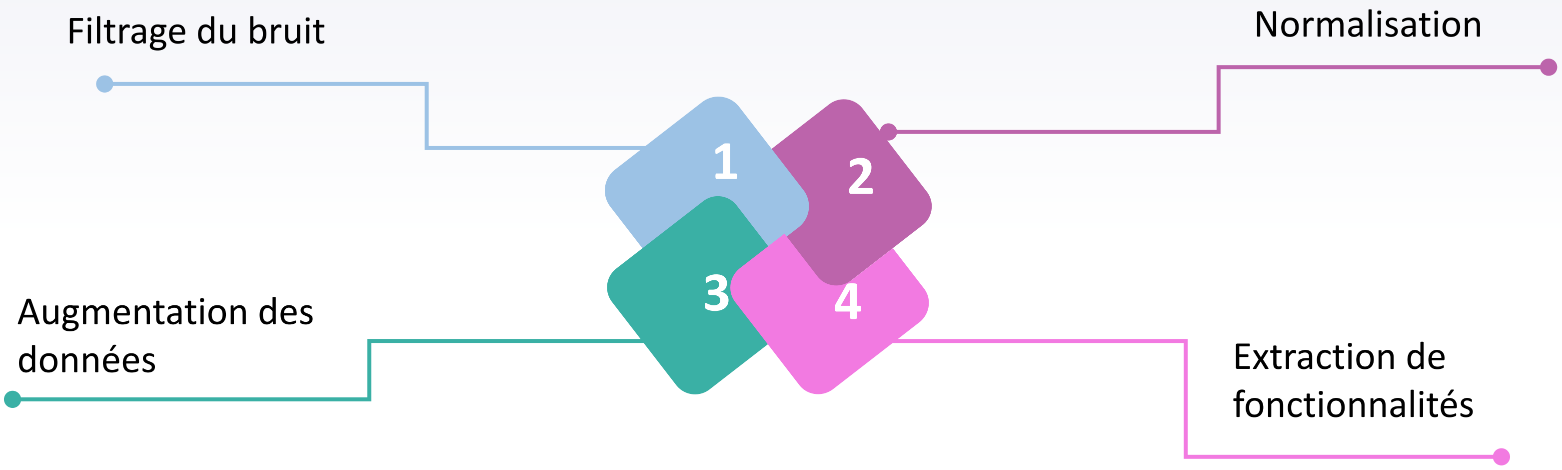
- 100 échantillons chacun

A	B	C	D	E
Non-Epileptic	Non-Epileptic	Epileptic	Epileptic	Epileptic
Eyes opened	Eyes closed	Interictal	Interictal	Ictal

Exploration de données



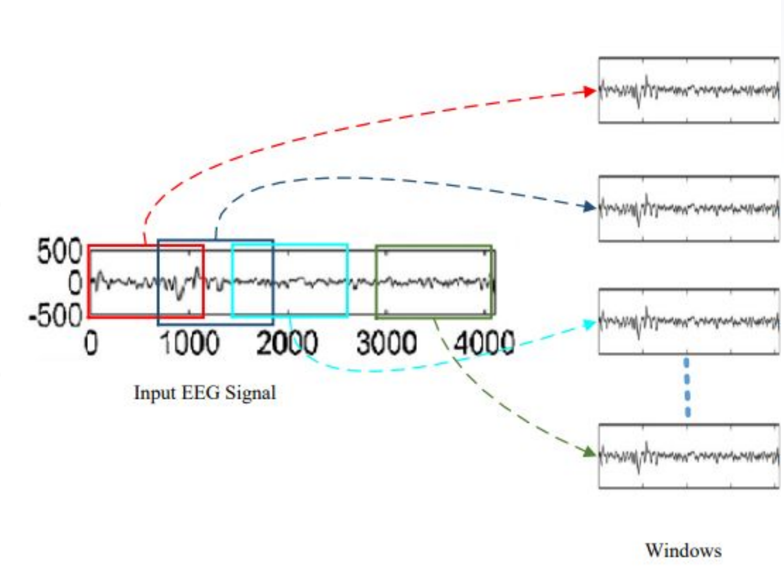
Préparation des données



Augmentation des données

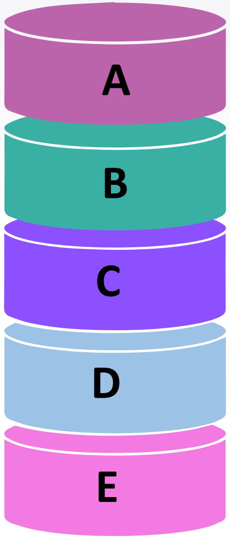
- Par fenêtre coulissante
- Une taille de fenêtre de 512 et un pas de 64

1 signal avec 4097 points



57 sous-signaux avec 512 points chacun

Augmentation des données

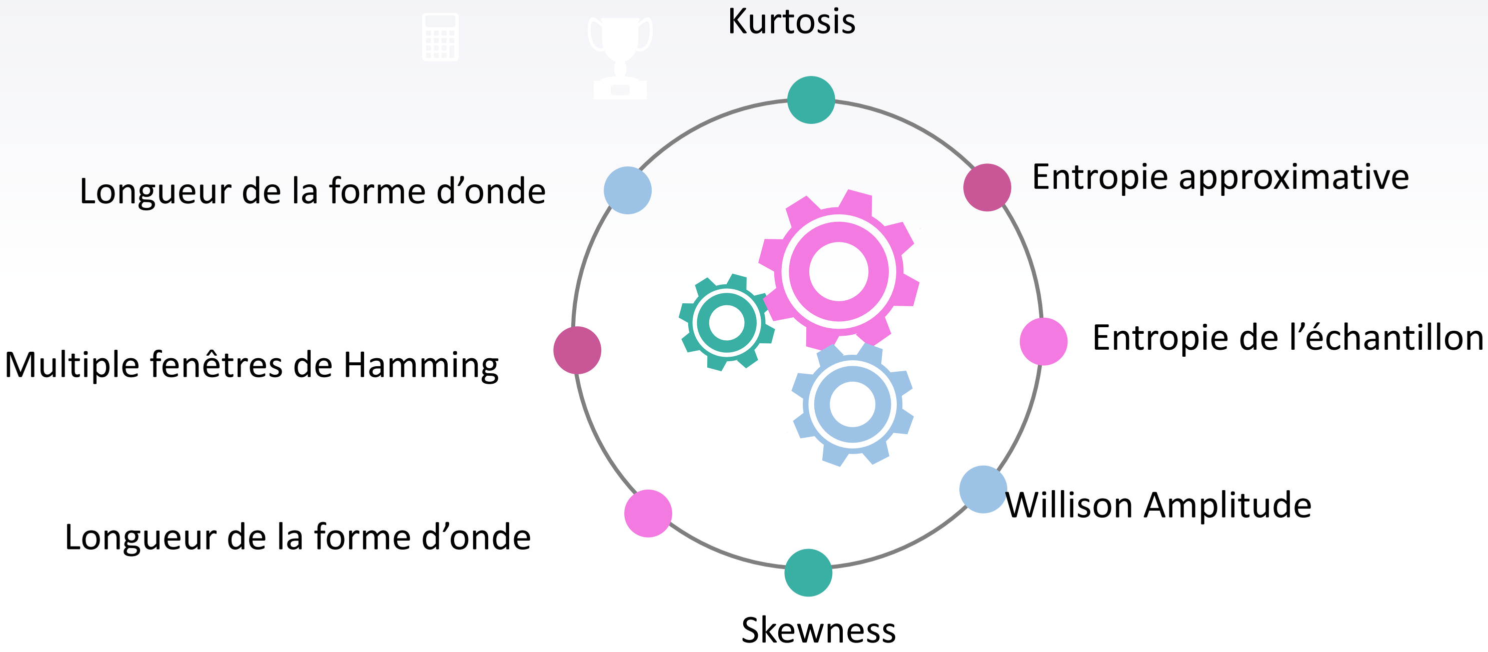


5700 échantillons chacun

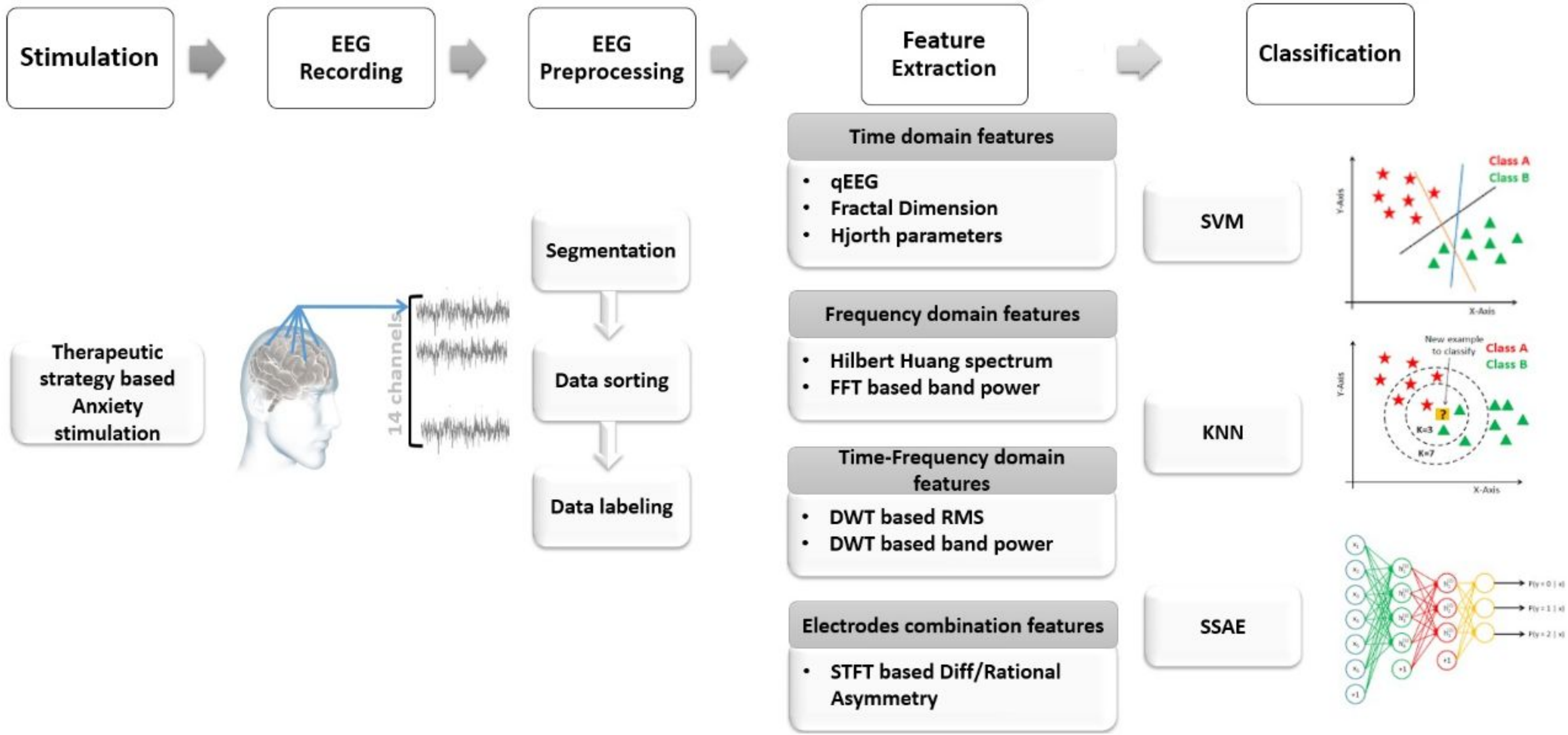
de 500 signaux

à 28500 signaux au total

Features extraction



Architecture proposée: article scientifique arxiv.org/abs/1901.02942v2



Features extraction dans le domaine temporel

1. **Mean absolute value:** it gives information about muscle contraction levels. It is defined as:

$$MAV_k = \frac{1}{N}\sum_{i=1}^{N}|x_i|$$

2. **Root mean square:** it reflects the mean power of the signal and is related to the constant force and non-fatiguing contractions. It is defined as:

$$RMS_k = \sqrt{\frac{1}{N}\sum_{i=1}^{N}x_i^{\;2}}$$

3. **Variance:** it is a measure of the power density of the signal. It is defined as:

$$VAR_k = \frac{1}{N}\sum_{i=1}^{N}(x_i - \overline{x})^2$$

Features extraction dans le domaine temporel

4. **Simple square integral:** it gives a measure of the energy of the EMG signal. It is defined as:

$$SSI_k = \sum_{i=1}^{N} (\left| x_i^{\ 2} \right|)$$

5. **Zero crossings**: it is the number of times the waveform crosses zero. This feature provides an approximate estimation of frequency domain properties. The threshold avoids counting zero crossings induced by noise. It is calculated as follows

   Increment one unit the zero crossings count value if

$$\{x_i > 0 \; and \; x_{i+1} < 0\} \; or \; \{x_i < 0 \; and \; x_{i+1} > 0\} \; and \; \left| x_i - x_{i+1} \right| \geq threshold$$

   Which can be rewritten as:

$$ZC_k = \sum_{i=1}^{N-1} \{f(x_i * x_{i+1})\};$$

$$f(x) \; = \; 1 \; if \; x \; < \; 0 \; and \; \left| x_i - x_{i+1} \right| \geq threshold$$

$$f(x) \; = \; 0 \; otherwise$$

Features extraction dans le domaine temporel

6. **Waveform length:** it is the cumulative length of the waveform over the segment. This feature is related to the signal amplitude, frequency and time. It is defined as:

$$WL_k = \sum_{i=1}^{N-1} \left| x_{i+1} - x_i \right|$$

7. **Slope sign changes:** it is similar to the zero crossings feature. It also provides information about the frequency content of the signal. It is calculated as follows Increment one unit the slope sign change value if:

$$\{x_i > x_{i-1} and\ x_i > x_{i+1}\}\ or\ \{x_i < x_{i-1} and\ x_i < x_{i+1}\}\ and\ \left| x_i - x_{i+1} \right| \geq\ threshold\ or\ \left| x_i - x_{i-1} \right| \geq threshold$$

Which can be rewritten as:

$$SSC_k = \sum_{i=2}^{N-1} [f[(x_i - x_{i-1})(x_i - x_{i+1})]];$$

$$f(x) \ = \ 1\ if\ x \geq threshold$$

$$f(x) \ = \ 0\ otherwise$$

8. **Willison amplitude:** it is the number of times that the difference in amplitude between adjacent data points exceed a predefined threshold. This feature provides information about the muscle contraction level. It is defined as:

$$WAMP_k = \sum_{i=1}^{N-1} f(\left| x_i - x_{i+1} \right|)$$

$$f(x) \ = \ 1\ if\ x\ \geq threshold$$

$$f(x) \ = \ 0\ otherwise$$

Features extraction

6. **Waveform length:** it is the cumulative length of the waveform over the segment. This feature is related to the signal amplitude, frequency and time. It is defined as:

$$WL_k = \sum_{i=1}^{N-1} \left| x_{i+1} - x_i \right|$$

7. **Slope sign changes:** it is similar to the zero crossings feature. It also provides information about the frequency content of the signal. It is calculated as follows Increment one unit the slope sign change value if:

$$\{x_i > x_{i-1} and\ x_i > x_{i+1}\}\ or\ \{x_i < x_{i-1} and\ x_i < x_{i+1}\}\ and\ \left| x_i - x_{i+1} \right| \geq threshold\ or\ \left| x_i - x_{i-1} \right| \geq threshold$$

Which can be rewritten as:

$$SSC_k = \sum_{i=2}^{N-1} [f[(x_i - x_{i-1})(x_i - x_{i+1})]];$$

$$f(x) = 1\ if\ x \geq threshold$$

$$f(x) = 0\ otherwise$$

8. **Willison amplitude:** it is the number of times that the difference in amplitude between adjacent data points exceed a predefined threshold. This feature provides information about the muscle contraction level. It is defined as:

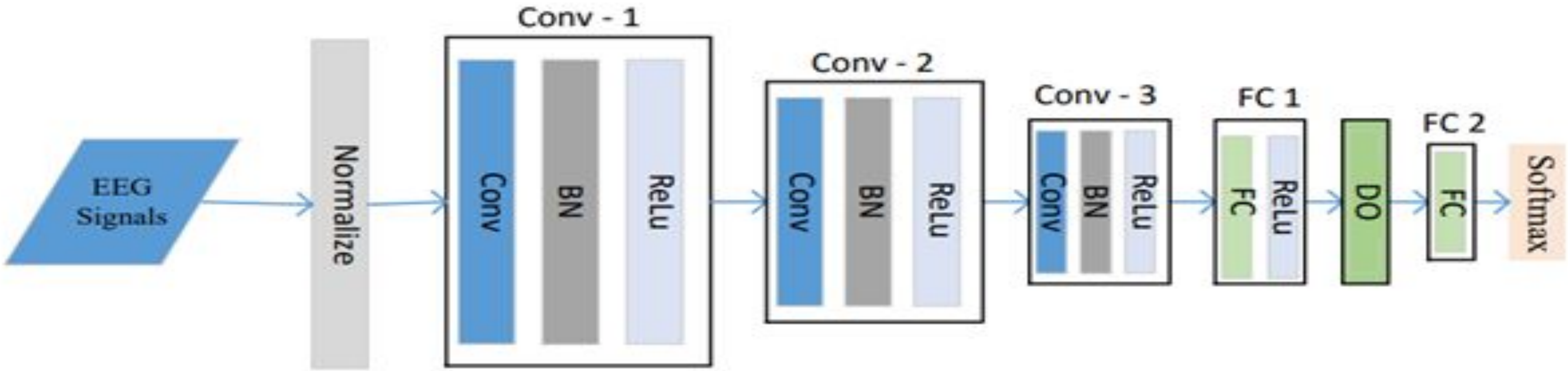
$$WAMP_k = \sum_{i=1}^{N-1} f(\left| x_i - x_{i+1} \right|)$$

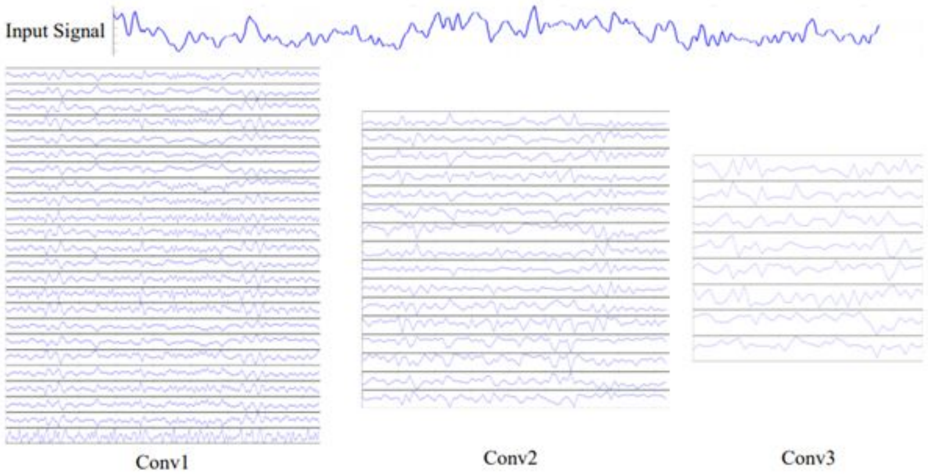
$$f(x) = 1\ if\ x \geq threshold$$

$$f(x) = 0\ otherwise$$

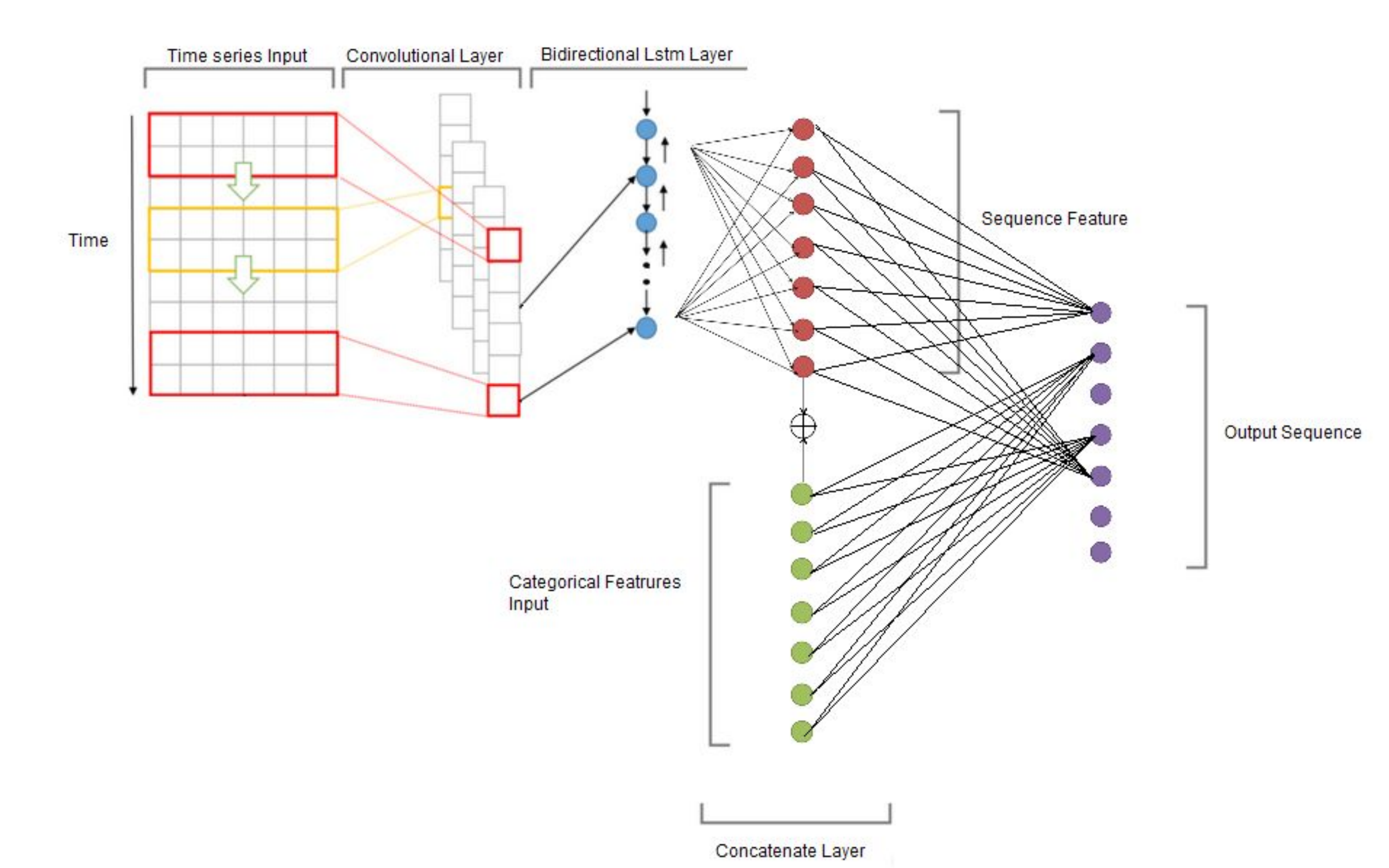
Résultat d'une segmentation + LDA

P-1D-CNN en utilisant le Deep Learning





Modélisation



Modélisation

Classifier les profils épileptiques et non épileptiques

Un problème de **classification binaire**

Machine learning

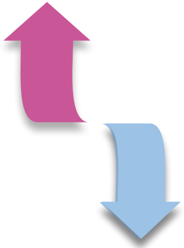
Modèle	Accuracy	Precision	Recall
KNN	90%	83%	61%
Random Forest	94%	88%	80%
Naïf bayes	79%	48%	70%
SVM	91%	84%	70%
LDA	84%	77%	31%
Decision tree	91%	78%	80%
XGBoost	84%	89%	21%
LightGBM	94,5%	88%	82%

Deep learning

Modèle	Accuracy
LSTM	95%

Modélisation

Classifier les phases:
normal - inter crise - crise



Un problème de
classification à 3 classes

Machine learning

Modèle	...	...call
KNN		%
Random Fore		%
Naïf bayes		%
SVM		%
LDA		%
Decision tree		%
XGBoost		%
LightGBM		%

Deep learning

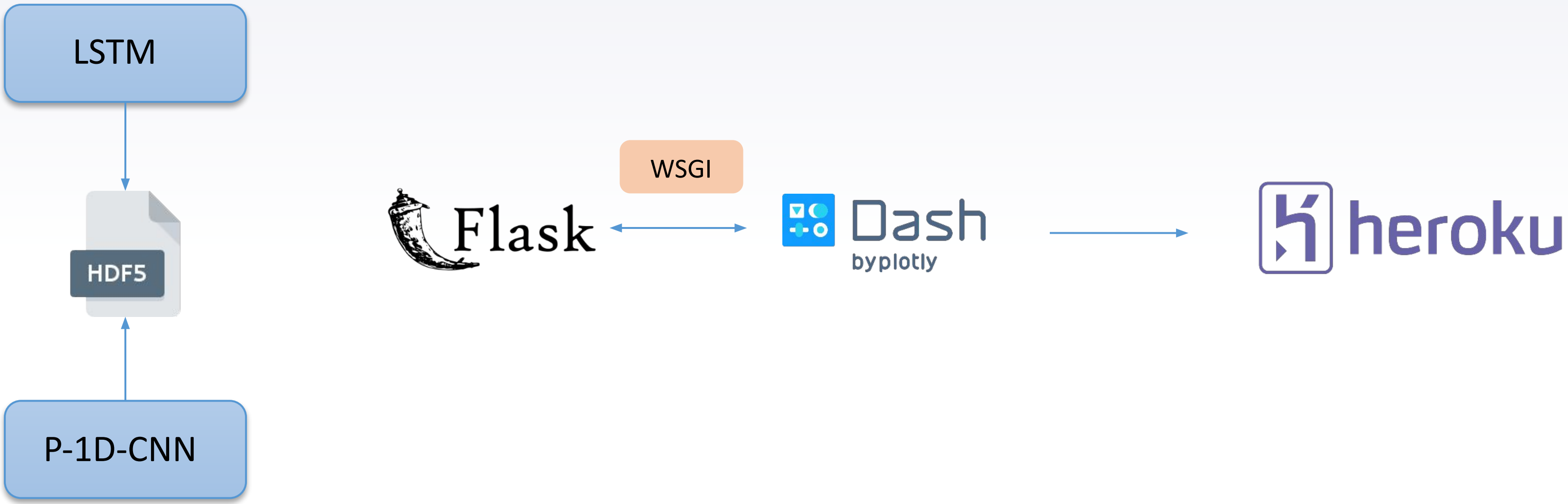
- Un signal EEG est une série chronologique à une dimension:

✔ P1D-CNN

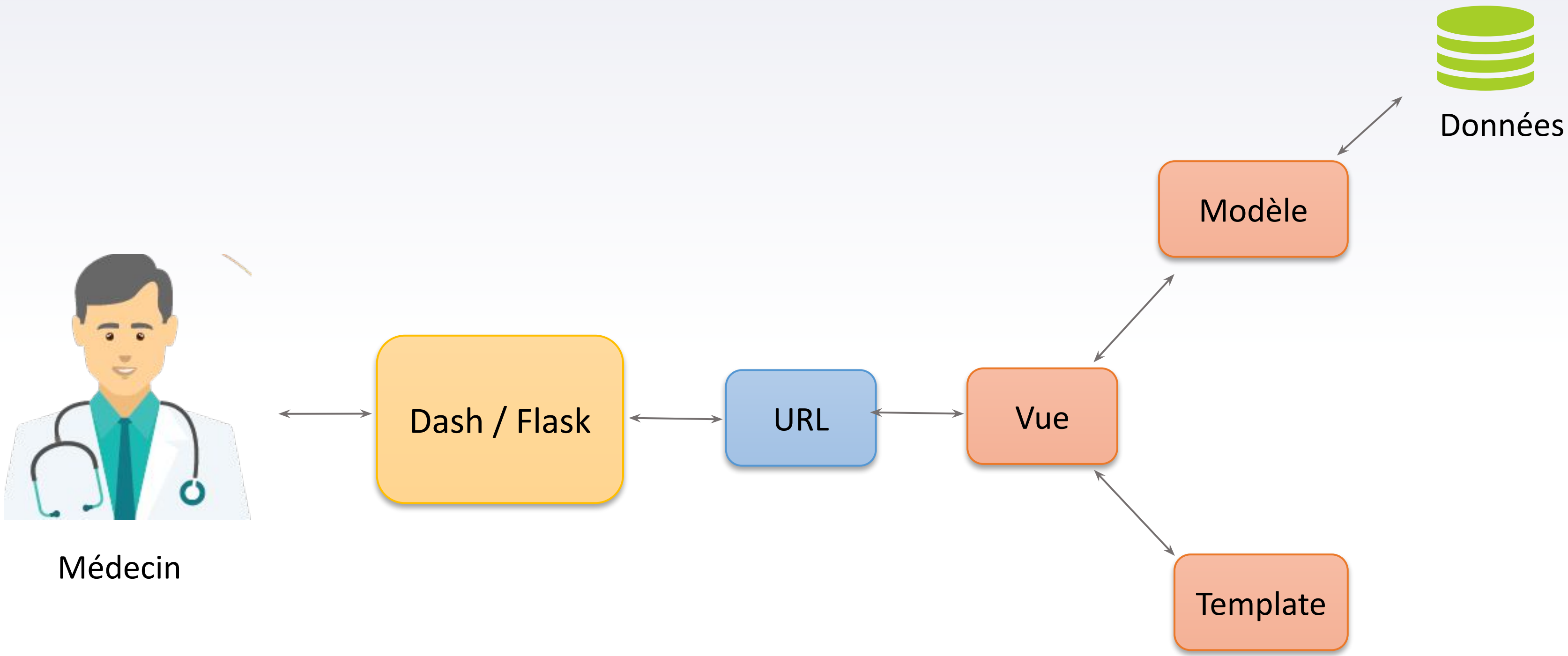
- Pour son évaluation, nous avons adopté une validation croisée par 10

Accuracy
96,1%

Déploiement

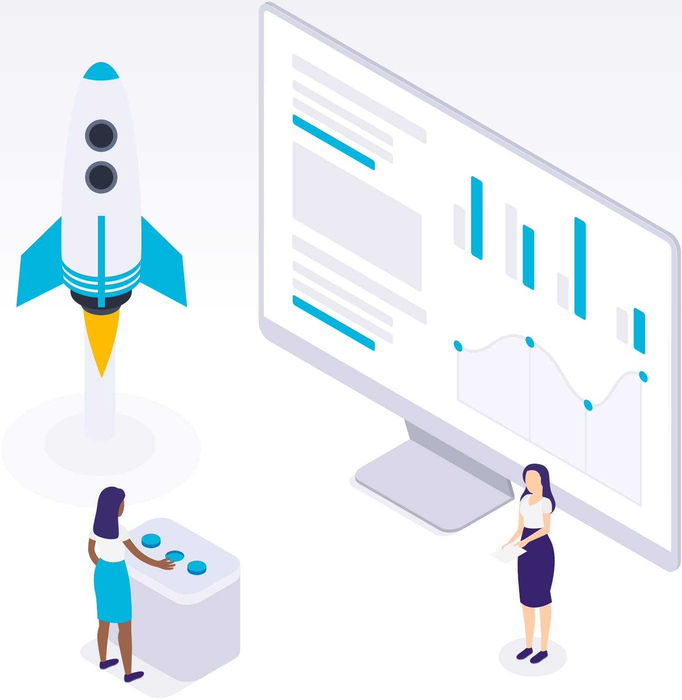


Architecture



4

Décomposition



Principe mathématique de la décomposition

Décomposition discrète:

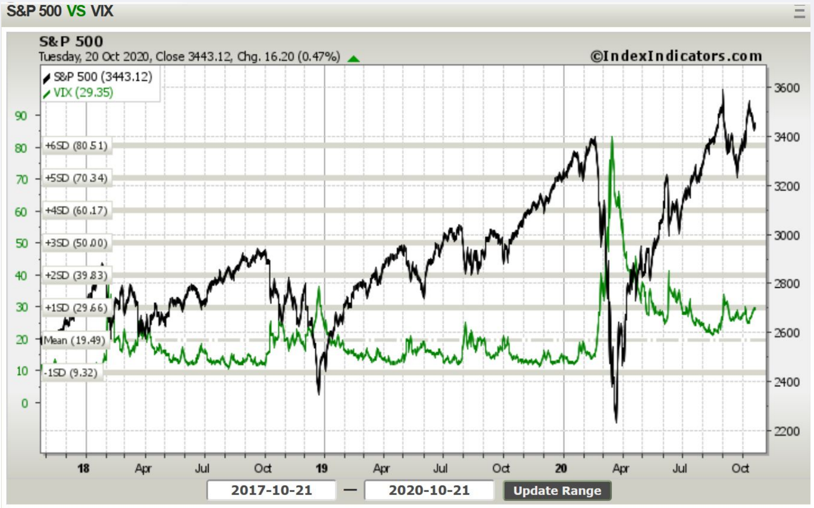
Un signal continu et périodique dans le temps peut être décomposé en une série des fonctions trigonométriques sinus et cosinus.

Ainsi, on parle d'une décomposition en Série de Fourier qui exprime le signal périodique comme **une combinaison linéaire** de signaux sinusoïdaux

$$x(t)=a_0+\sum_{n=1}^{+\infty} a_n \cos\left(n\frac{2\pi}{T}t\right)+b_n \sin\left(n\frac{2\pi}{T}t\right)$$

Somme de sinus et de cosinus : facile à interpréter

Principe mathématique de la décomposition



Problème :

Les séries temporelles financières sont continues mais non périodiques.

Solution pour les signaux continus et non périodiques

Décomposition continue:

$$X(f) = \int_{-\infty}^{+\infty} x(t)\, e^{-j2\pi ft}\, dt$$

Application avec un code en Python:

- **Signal sans bruit ayant une seule fréquence**
- **Signal sans bruit composé de deux fréquences**
- **Signal avec bruit composé de deux fréquences**

numpy.fft.fft

numpy.fft.fft(*a*, *n=None*, *axis=-1*, *norm=None*)

Compute the one-dimensional discrete Fourier Transform.

Meilleure performance avec numpy.fft.fft

numpy.fft.fft

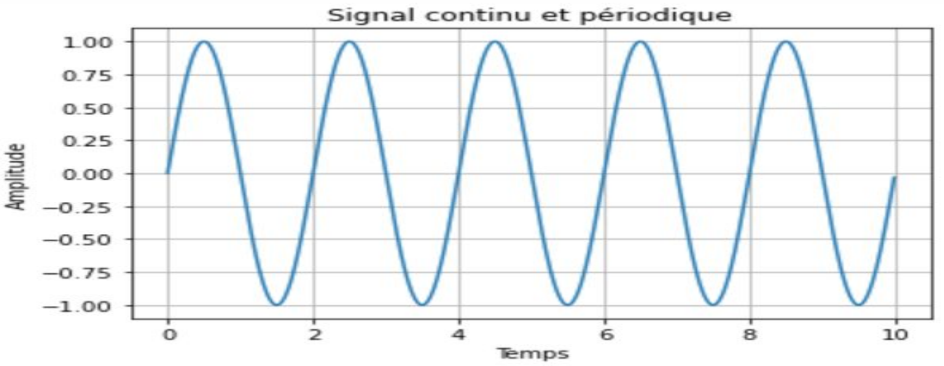
numpy.fft.fft(*a*, *n=None*, *axis=-1*, *norm=None*)

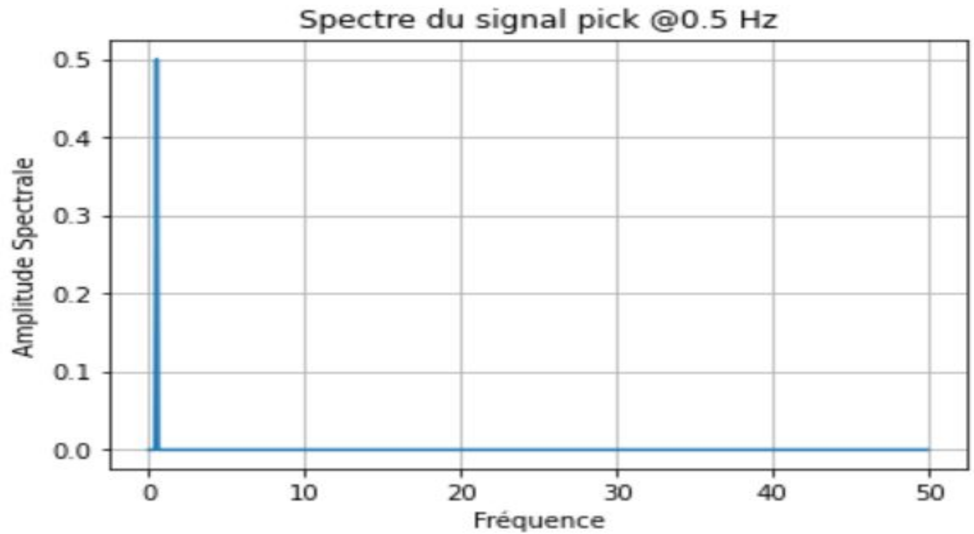
Compute the one-dimensional discrete Fourier Transform.

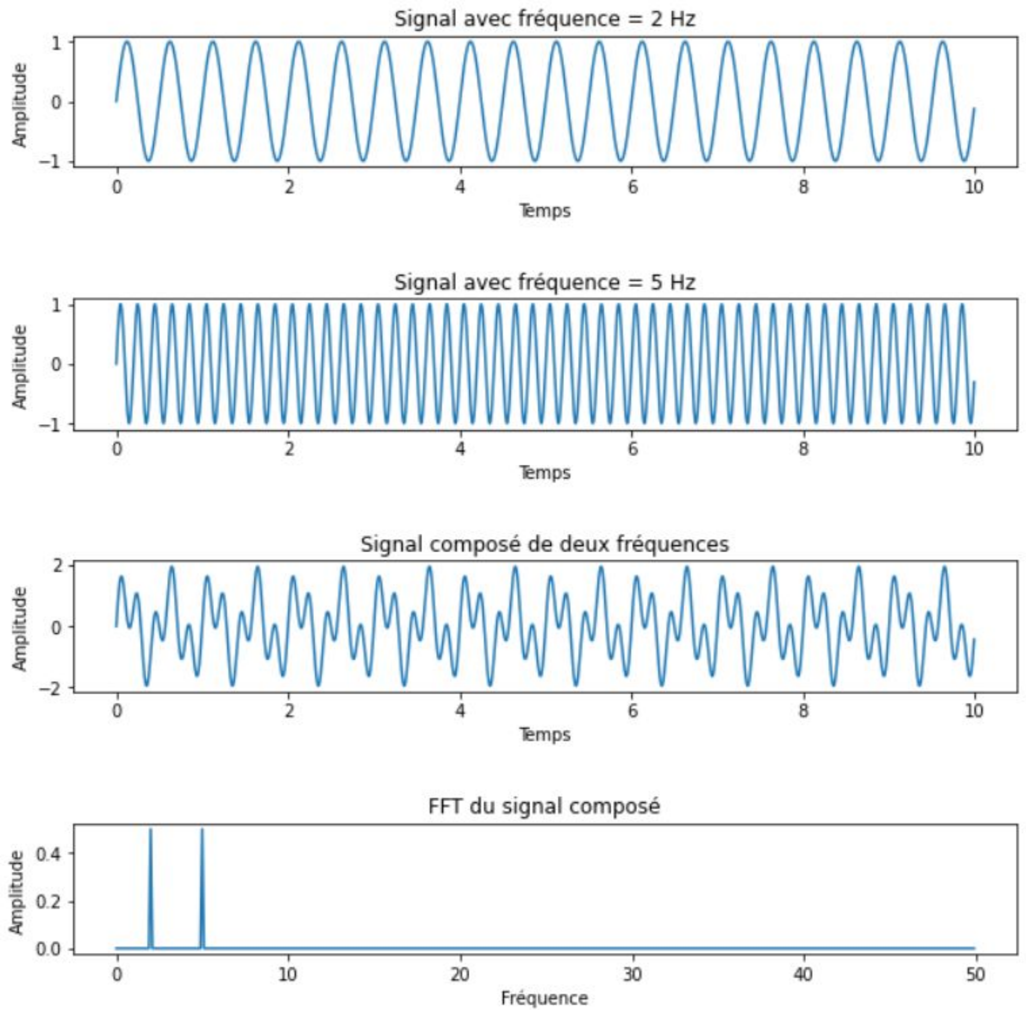
```
%timeit np.fft.fft(signal)
49.5 µs ± 3.17 µs per loop (mean ± std. dev. of 7 runs, 10000 loops each)

%timeit DFT_lent(signal)
969 ms ± 7.94 ms per loop (mean ± std. dev. of 7 runs, 1 loop each)

%timeit FFT(signal)
10.6 ms ± 425 µs per loop (mean ± std. dev. of 7 runs, 100 loops each)

%timeit FFT_vectorized(signal)
862 µs ± 23.5 µs per loop (mean ± std. dev. of 7 runs, 1000 loops each)
```

```python
# Chargement des modules
import numpy as np
import matplotlib.pyplot as plt
%matplotlib inline
# Fréquence d'échantillonnage
freq_sampling = 100
# Intervalles d'échantillonnage
interval_sampling = 1 / freq_sampling
# Temps initial
ti = 0
# Temps final
tf = 10
# Fréquence du signal
freq = 0.5
# Intervalle temporel
time = np.arange(ti, tf, interval_sampling)
# Construction du signal
amplitude = np.sin(2 * np.pi * freq * time)
# Calcul de la transformation de Fourier
trans_fourier = np.fft.fft(amplitude)
# Normalisation de la transformation
trans_fourier = trans_fourier / len(amplitude)
# On supprime la fréquence d'échantillonnage
trans_fourier = trans_fourier[range( int(len(amplitude) / 2 )  )]
# Paramètres graphiques
compteur   = len(amplitude)
values      = np.arange(int(compteur/2))
periode_temporelle  = compteur / freq_sampling
frequences = values/periode_temporelle
```

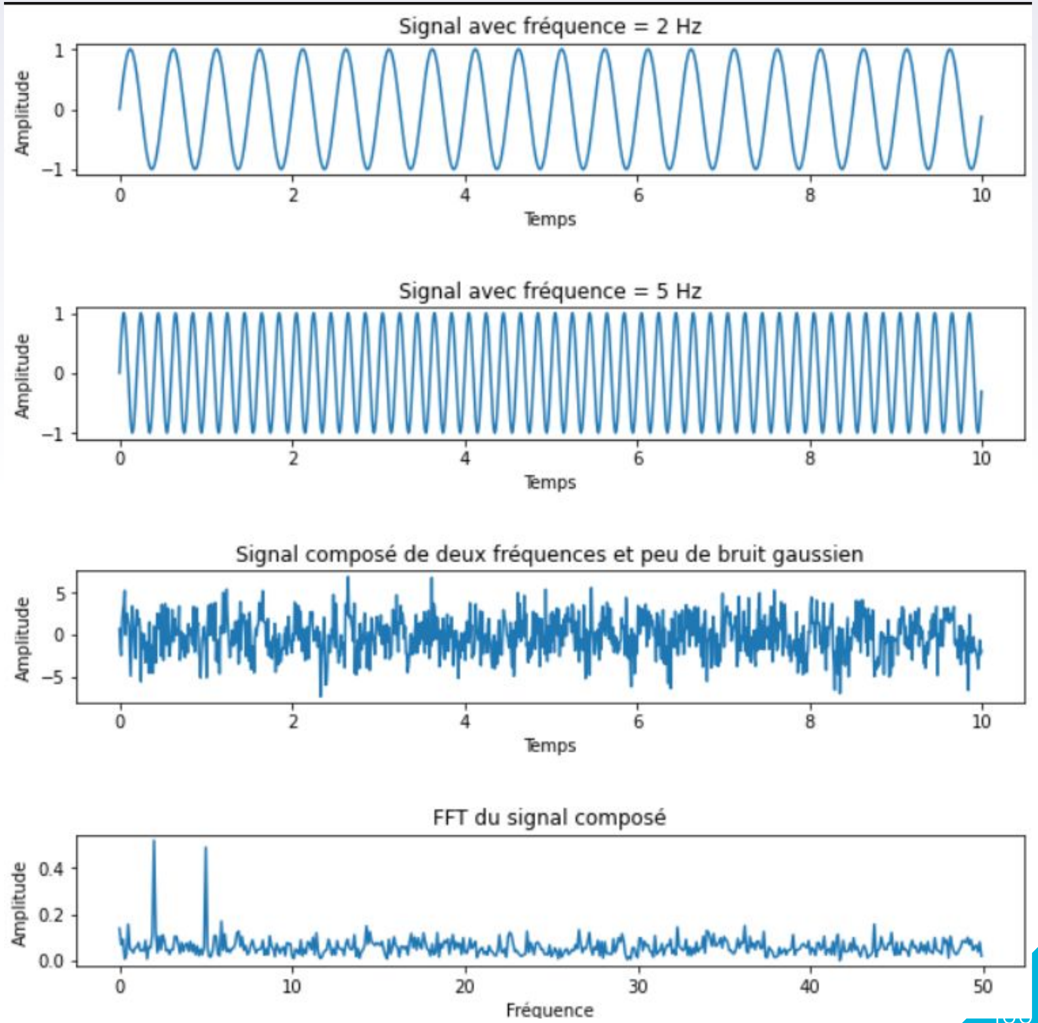
```python
plt.plot(time, amplitude)
plt.xlabel("Temps")
plt.ylabel("Amplitude")
plt.title("Signal continu et périodique")
plt.grid()
plt.show()
```



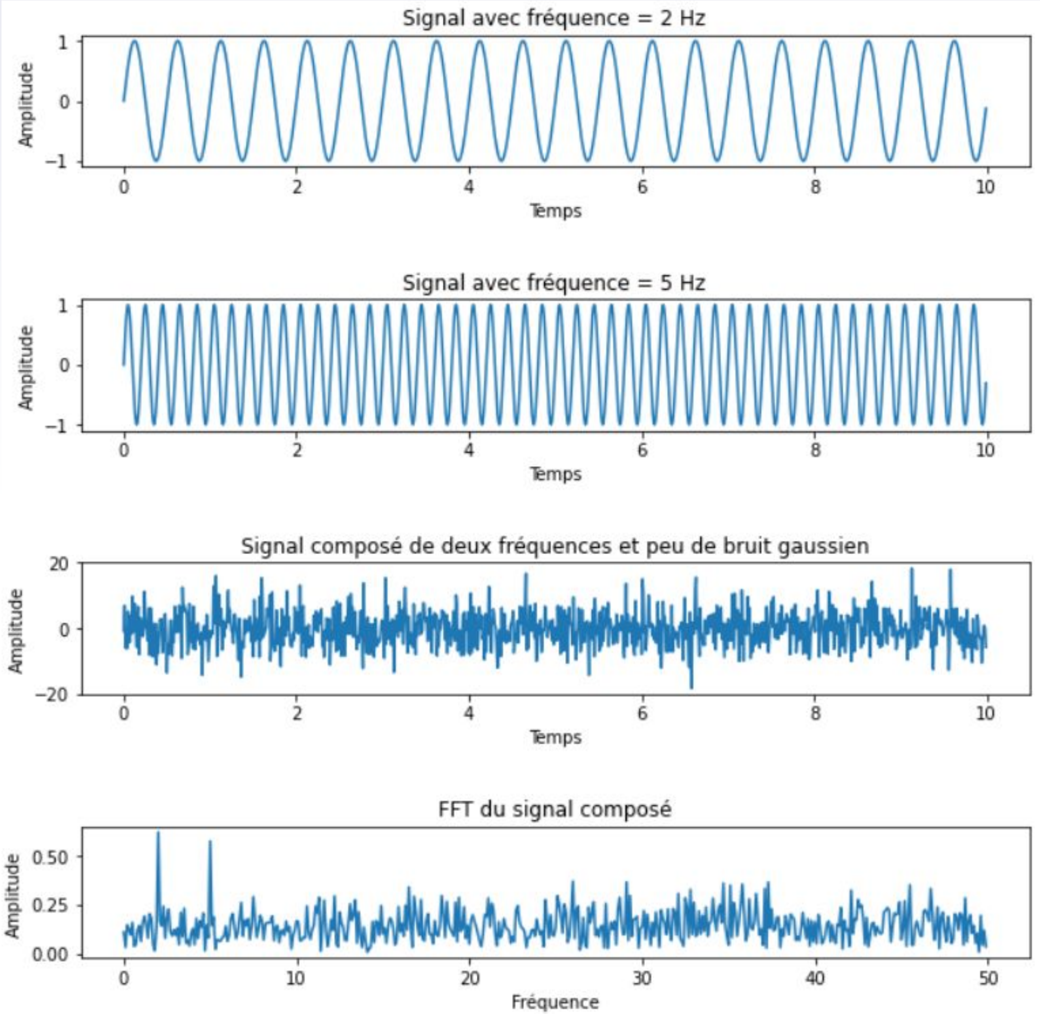




- La transformation de Fourier permet de détecter les deux fréquences composant le signal.

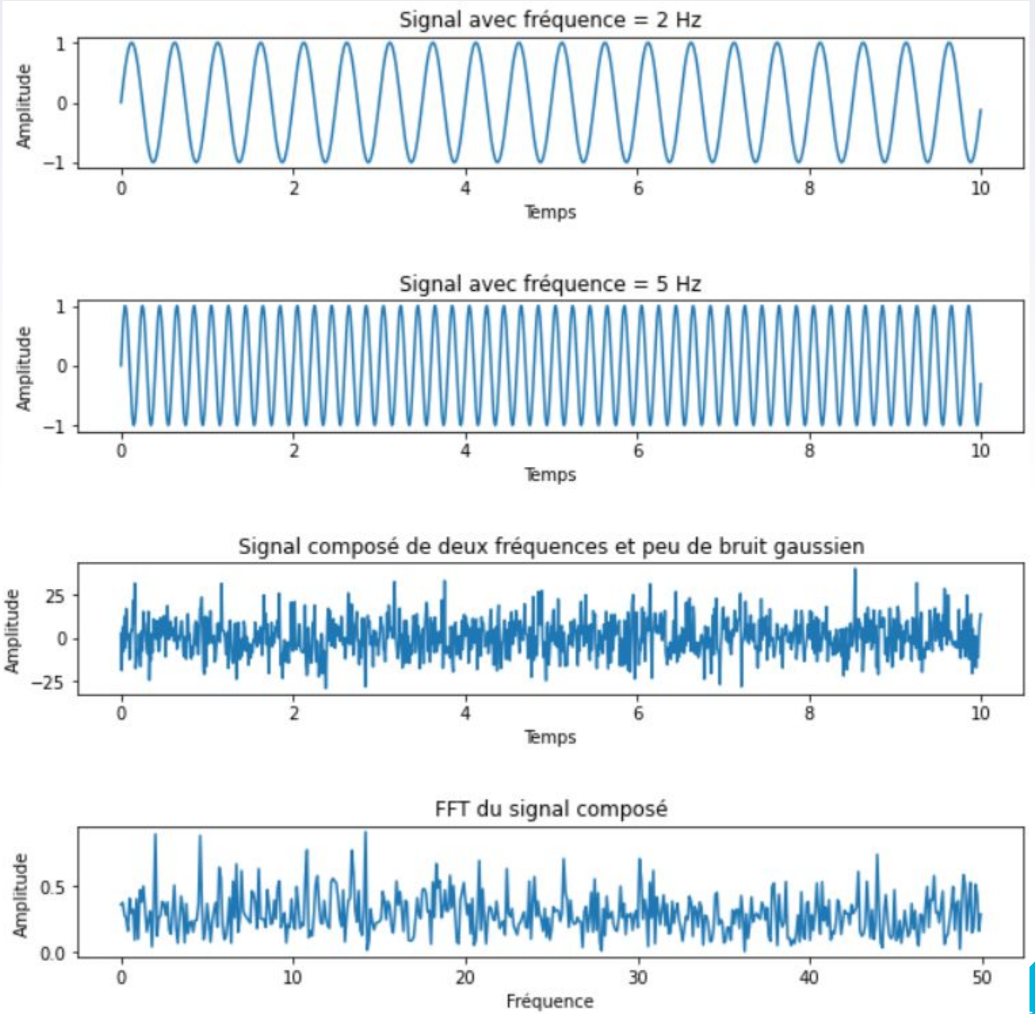


- Avec peu de bruit, la transformation de Fourier permet de détecter les deux fréquences composant le signal.



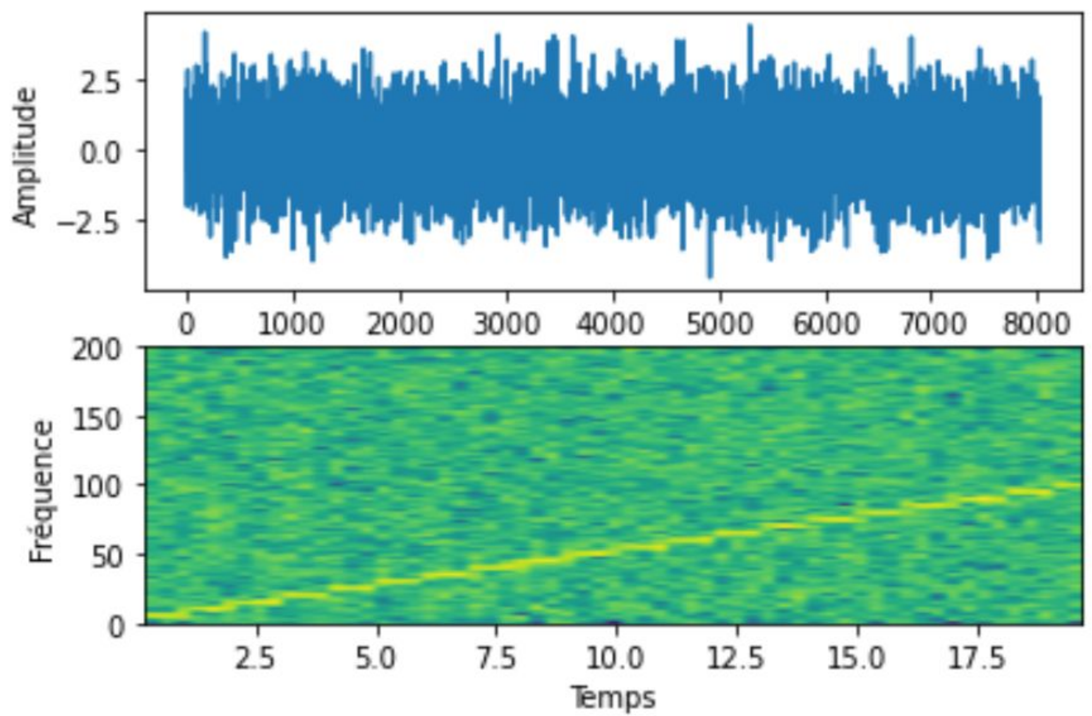
- Avec un bruit moyen, la transformation de Fourier arrive quand même à détecter les deux fréquences composant le signal.

- Avec beaucoup de bruit, il est presque impossible de détecter les deux fréquences



Interprétation à l'aide d'un spectrogramme

```
fig, (ax1, ax2) = plt.subplots(nrows=2)
ax1.plot(t, x)
Pxx, freqs, bins, im = ax2.specgram(x, NFFT=NFFT, Fs=Fs, noverlap=900)
# The `specgram` method returns 4 objects. They are:
# - Pxx: the periodogram
# - freqs: the frequency vector
# - bins: the centers of the time bins
# - im: the .image.AxesImage instance represe
plt.show()
```



by Dr Time and Brother Frequency

Integrate your function times a complex exponential It's really not so hard you can do it with your pencil And when you're done with this calculation You've got a brand new function - the Fourier Transformation

What a prism does to sunlight, what the ear does to sound

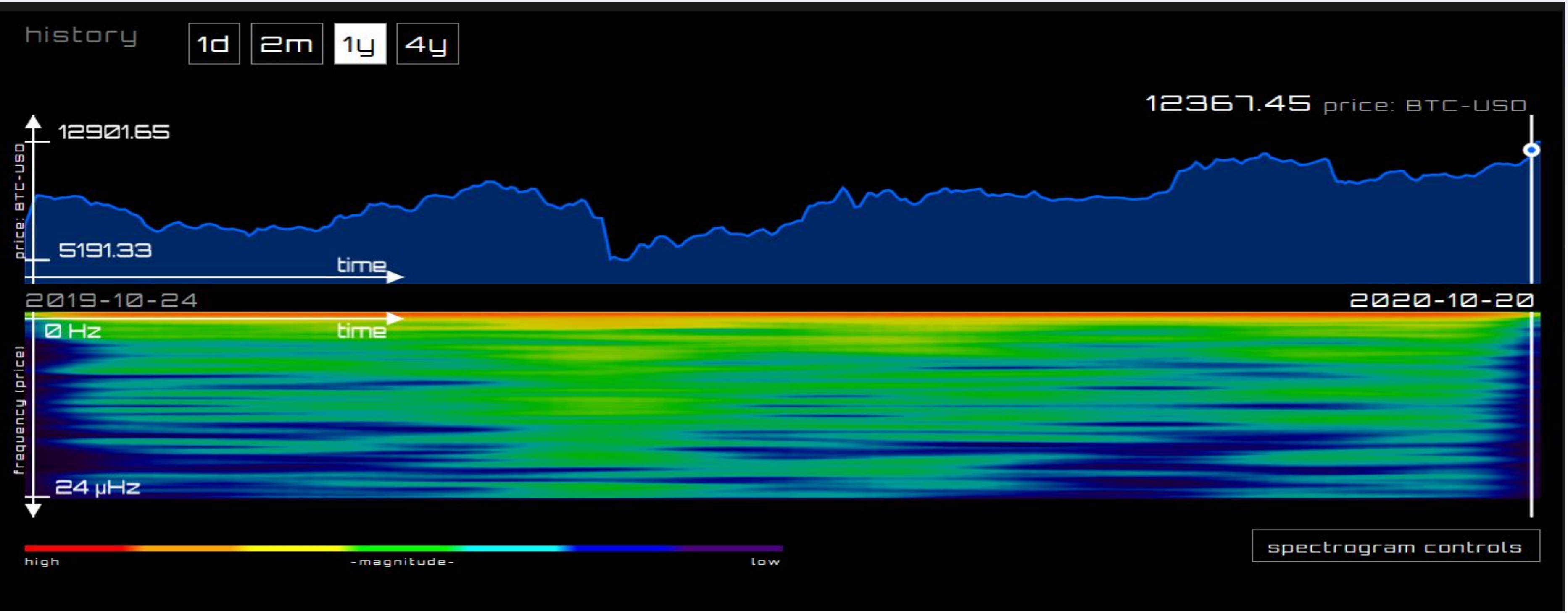
Fourier does to signals, it's the coolest trick around

Now filtering is easy, you don't need to convolve

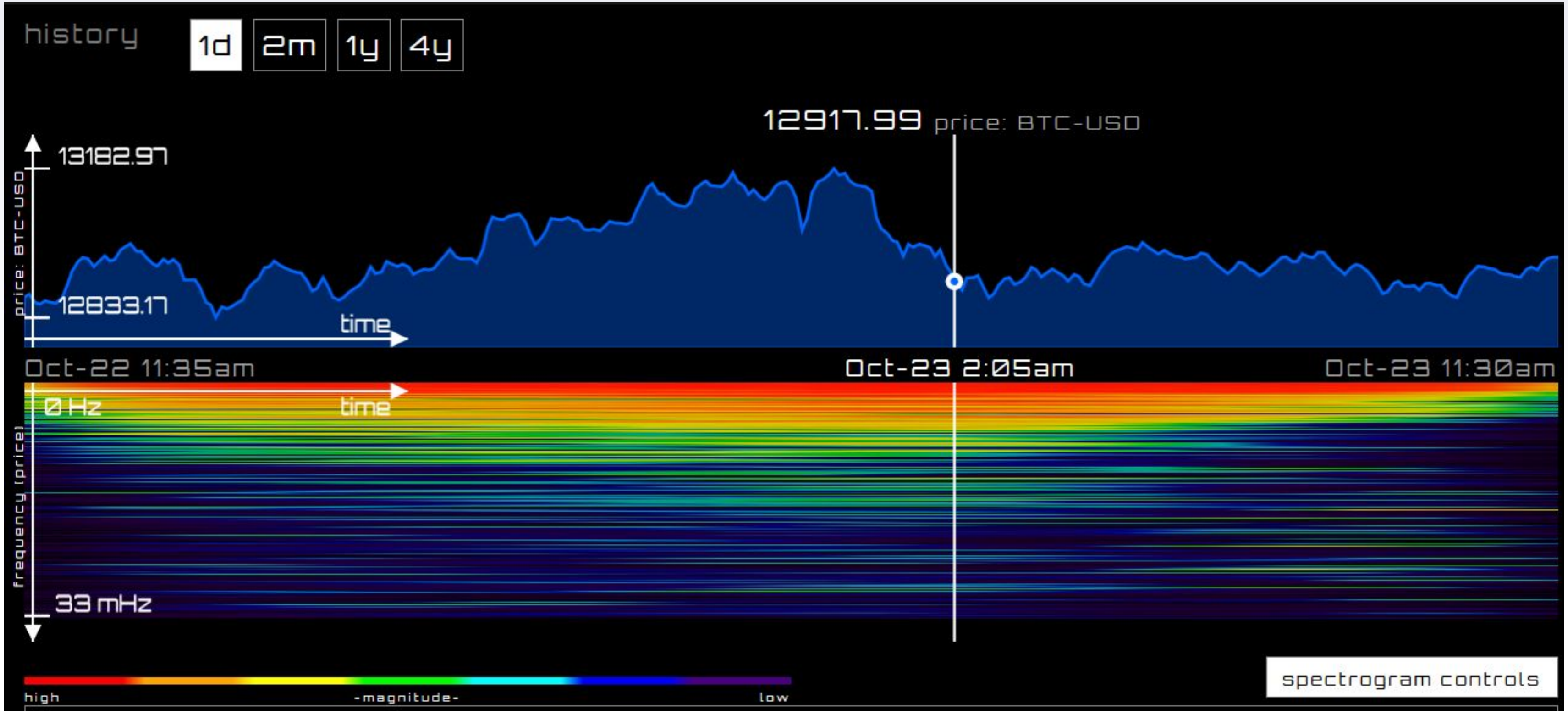
All you do is multiply in order to solve. From time into frequency - from frequency to time Every operation in the time domain Has a Fourier analog - that's what I claim Think of a delay, a simple shift in time It becomes a phase rotation - now that's truly sublime!And to differentiate, here's a simple trick Just multiply by J omega, ain't that slick?Integration is the inverse, what you gonna do?

Divide instead of multiply - you can do it too. From time into frequency - from frequency to time etc …

La suite + l'interprétation sur http://www.jmlg.org/lyrics/Fouriers_Song.htm

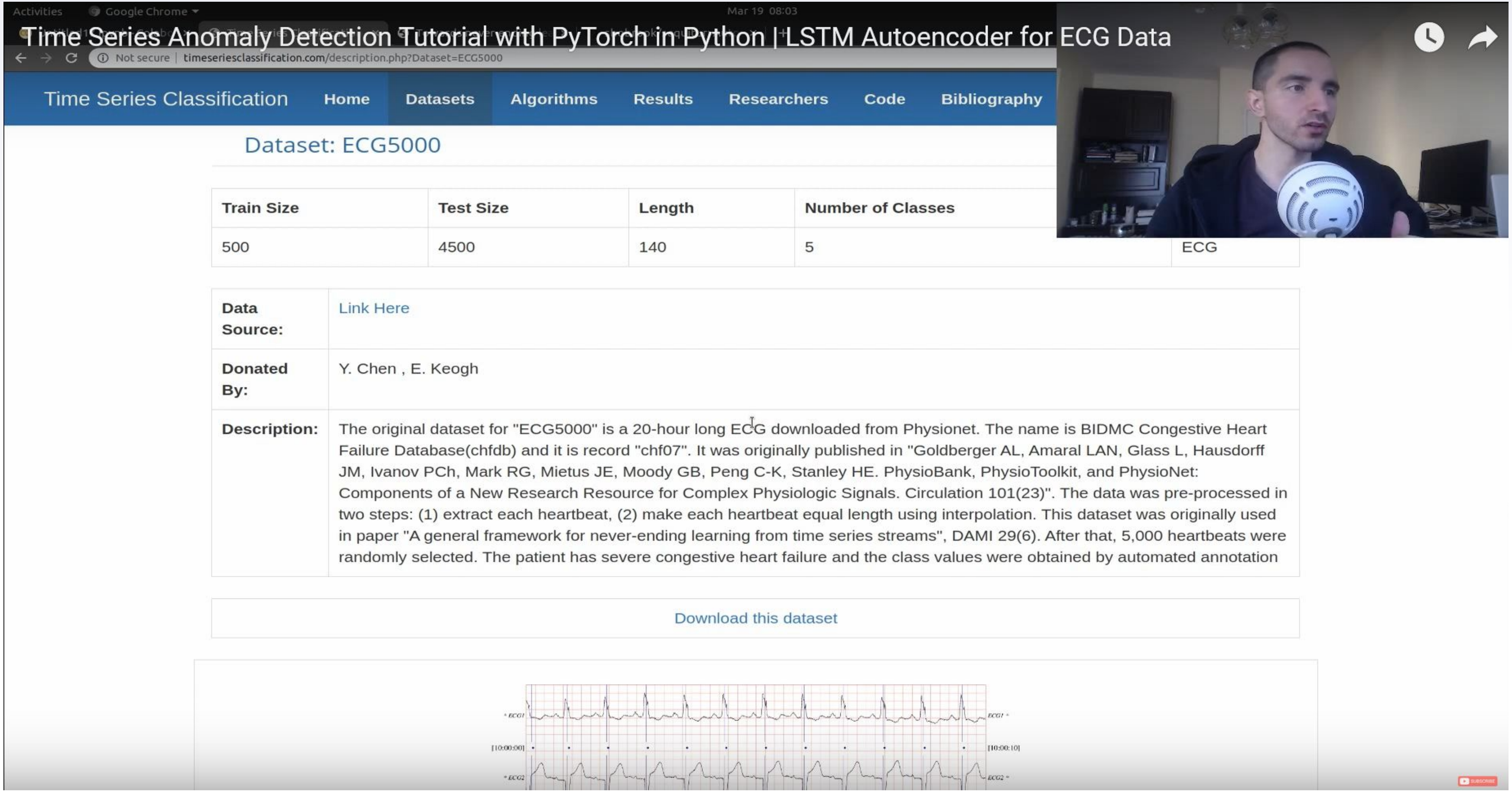


**** Quand les variations sont relativement stables ==> le spectre contient principalement relativement des faibles fréquences.
**** Quand les variations sont grandes (il y a de la volatilité) ==> tout le spectre s'illumine et ces événements ressemblent à des flammes.



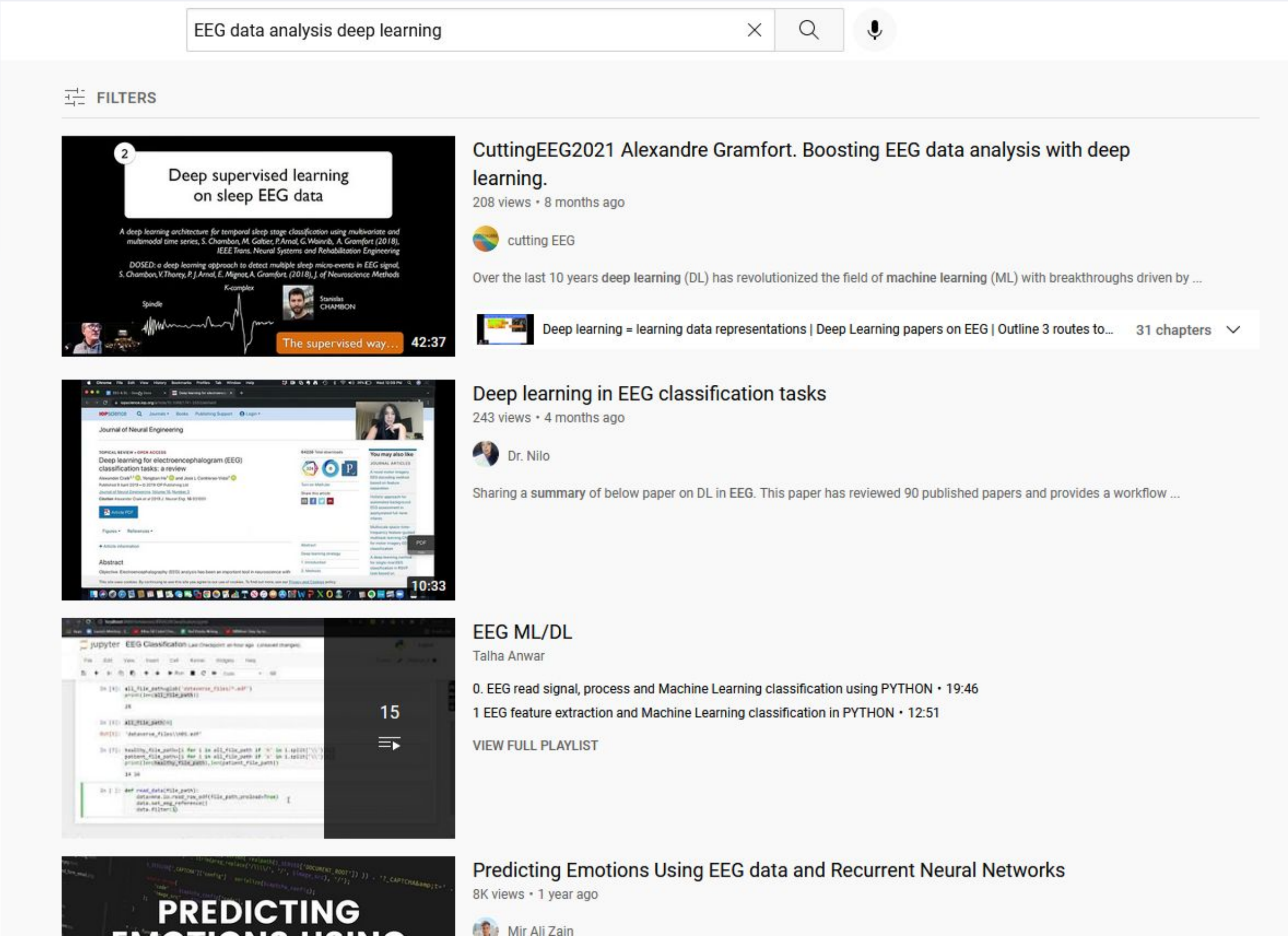
**** Quand les variations sont relativement stables ==> le spectre contient principalement relativement des faibles fréquences.
**** Quand les variations sont grandes (il y a de la volatilité) ==> tout le spectre s'illumine et ces événements ressemblent à des flammes.

Ressources



Activities
Google Chrome
Mar 19 08:03
Time Series Anomaly Detection Tutorial with PyTorch in Python | LSTM Autoencoder for ECG Data
Not secure | timeseriesclassification.com/description.php?Dataset=ECG5000
Time Series Classification
Home
Datasets
Algorithms
Results
Researchers
Code
Bibliography
Dataset: ECG5000
Train Size
Test Size
Length
Number of Classes
500
4500
140
5
ECG
Data Source:
Link Here
Donated By:
Y. Chen , E. Keogh
Description:
The original dataset for "ECG5000" is a 20-hour long ECG downloaded from Physionet. The name is BIDMC Congestive Heart Failure Database(chfdb) and it is record "chf07". It was originally published in "Goldberger AL, Amaral LAN, Glass L, Hausdorff JM, Ivanov PCh, Mark RG, Mietus JE, Moody GB, Peng C-K, Stanley HE. PhysioBank, PhysioToolkit, and PhysioNet: Components of a New Research Resource for Complex Physiologic Signals. Circulation 101(23)". The data was pre-processed in two steps: (1) extract each heartbeat, (2) make each heartbeat equal length using interpolation. This dataset was originally used in paper "A general framework for never-ending learning from time series streams", DAMI 29(6). After that, 5,000 heartbeats were randomly selected. The patient has severe congestive heart failure and the class values were obtained by automated annotation
Download this dataset

Ressources


EEG data analysis deep learning
FILTERS
Deep supervised learning on sleep EEG data
The supervised way...
42:37
CuttingEEG2021 Alexandre Gramfort. Boosting EEG data analysis with deep learning.
208 views • 8 months ago
cutting EEG
Over the last 10 years deep learning (DL) has revolutionized the field of machine learning (ML) with breakthroughs driven by ...
Deep learning = learning data representations | Deep Learning papers on EEG | Outline 3 routes to...
31 chapters
10:33
Deep learning in EEG classification tasks
243 views • 4 months ago
Dr. Nilo
Sharing a summary of below paper on DL in EEG. This paper has reviewed 90 published papers and provides a workflow ...
15
EEG ML/DL
Talha Anwar
0. EEG read signal, process and Machine Learning classification using PYTHON • 19:46
1 EEG feature extraction and Machine Learning classification in PYTHON • 12:51
VIEW FULL PLAYLIST
PREDICTING
Predicting Emotions Using EEG data and Recurrent Neural Networks
8K views • 1 year ago
Mir Ali Zain

Ressources

Search

Browse State-of-the-Art | Datasets | Methods | More

EEGNet: A Compact Convolutional Network for EEG-based Brain-Computer Interfaces

23 Nov 2016 · Vernon J. Lawhern, Amelia J. Solon, Nicholas R. Waytowich, Stephen M. Gordon, Chou P. Hung, Brent J. Lance · Edit social preview

Brain computer interfaces (BCI) enable direct communication with a computer, using neural activity as the control signal. This neural signal is generally chosen from a variety of well-studied electroencephalogram (EEG) signals. For a given BCI paradigm, feature extractors and classifiers are tailored to the distinct characteristics of its expected EEG control signal, limiting its application to that specific signal. Convolutional Neural Networks (CNNs), which have been used in computer vision and speech recognition, have successfully been applied to EEG-based BCIs; however, they have mainly been applied to single BCI paradigms and thus it remains unclear how these architectures generalize to other paradigms. Here, we ask if we can design a single CNN architecture to accurately classify EEG signals from different BCI paradigms, while simultaneously being as compact as possible. In this work we introduce EEGNet, a compact convolutional network for EEG-based BCIs. We introduce the use of depthwise and separable convolutions to construct an EEG-specific model which encapsulates well-known EEG feature extraction concepts for BCI. We compare EEGNet to current state-of-the-art approaches across four BCI paradigms: P300 visual-evoked potentials, error-related negativity responses (ERN), movement-related cortical potentials (MRCP), and sensory motor rhythms (SMR). We show that EEGNet generalizes across paradigms better than the reference algorithms when only limited training data is available. We demonstrate three different approaches to visualize the contents of a trained EEGNet model to enable interpretation of the learned features. Our results suggest that EEGNet is robust enough to learn a wide variety of interpretable features over a range of BCI tasks, suggesting that the observed performances were not due to artifact or noise sources in the data.

PDF | Abstract

Code

Edit

vlawhern/arl-eegmodels (official)	★ 666	TensorFlow
aliasvishnu/EEGNet	★ 137	PyTorch
gibranfp/P300-CNNT	★ 41	TensorFlow

Tasks

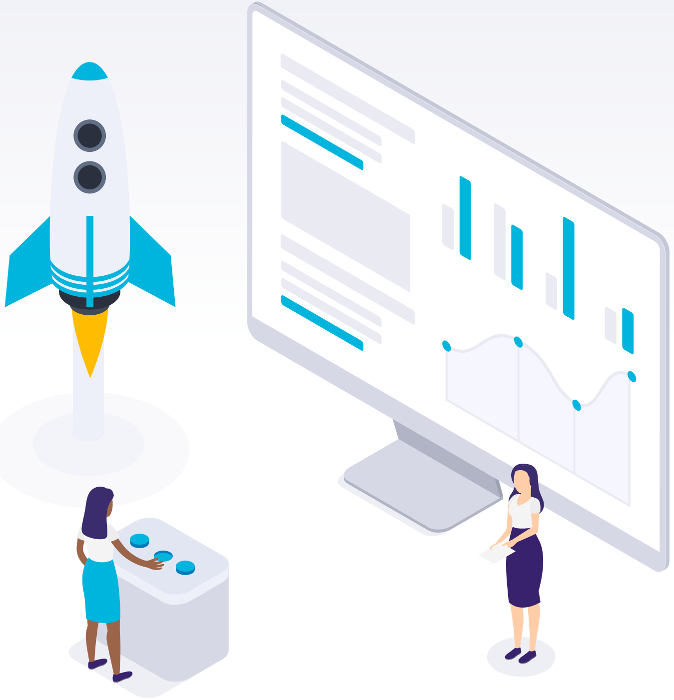
Edit

Computer Vision | EEG | speech-recognition

Ressources


1
NeuroTech
ML Methods and Kaggle
May 21, 2020
Slides by Zeyun Wu and Yundong Wang
2
About the club
- Affiliated with NeuroTechX
- Annual Student NTX Competition
- Faculty Advisor
- Connections with Wearable Sensing
- Multiple headsets for project groups next year!
3
Poll
- How much do you know about Machine Learning?
- Have you attended a Kaggle Competition before?
4
Presentation Outline
- EEG signals as time series data
- Machine Learning algorithms for EEG signal
- Deep Learning algorithms (essentially CNN) for EEG signal
- A Kaggle Project of BCI Challenge
5
EEG Signal as Time-series data
- Generally people use Machine Learning algorithms as a method to learn the feature of the data, make prediction or decision on the data
- EEG signals are n by k vectors, where n small is channel numbers and k large is the time points: high dimensional
https://tinyurl.com
/ntx-gbm-ML
University of California, San Diego
NeuroTech
ML Methods and Kaggle
May 21, 2020
Slides by Zeyun Wu and Yundong Wang

4 Conclusion



Measure	Description
	Statistical measures
MeanRR	Mean of RRIs
RMSSD	Square root of the mean squared differences of successive RRIs
VAR	Variance of RRIs
SDNN	Standard deviation of RRIs
SDSD	Standard deviation of differences between adjacent RRIs
NN50	Number of intervals differences of successive NN intervals greater than 50ms
pNN50	Proportion of NN50 to the total number of NN intervals
NN20	Number of intervals differences of successive NN intervals greater than 20ms
pNN20	Proportion of NN20 to the total number of NN intervals
	Geometric measures
HRV Triangular Index	Total number of NN intervals divided by the height of the histogram of all NN intervals.
TINN	Baseline width of the minimal squared differences of the interpolation of the highest peak I the histogram of all NN intervals.

Measure	Description
VLF power	Power of the very low frequency band ($\leq$ 0.04Hz) of the PSD.
LF Power	Power of the low frequency band (0.04 – 0.15 Hz) of the PSD.
HF power	Power of the high frequency band (0.15 – 0.4 Hz) of the PSD.
LF/HF	Ratio of LF to HF.