

Cours de Vision par Ordinateur

Florian Valade

January 29, 2026

Présentation

Florian Valade

florian.valade@univ-eiffel.fr

Data Scientist chez Fujitsu

Chercheur au LAMA



Objectif du cours

Comprendre la vision par ordinateur et ses applications industrielles
Permettre aux étudiants d'évoluer en tant que :

- Data Scientist
- Machine Learning Engineer
- Chercheur en Computer Vision

Structure du cours

- **Introduction** : Fondamentaux du traitement d'images
 - Représentation numérique des images
 - Concepts clés en vision par ordinateur
- **Méthodes classiques**
 - Filtres et convolutions
 - Détection de contours et segmentation
 - Travaux pratiques sur des cas réels
- **Deep Learning en Vision**
 - Réseaux de neurones convolutifs (CNN)
 - Architecture Transformer
 - Applications modernes

Notation du cours

Évaluation par projet

- **Projet de Computer Vision** (100% de la note finale)
 - Développement d'une solution complète de vision par ordinateur
 - Application des concepts vus en cours sur un cas concret
 - Travail en groupe de 3-4 étudiants
- **Livrables attendus**
 - Code source documenté
 - Rapport technique détaillé
- **Date de rendu** : Fin du semestre

Pourquoi la vision par ordinateur ?

Relier deux mondes :

- Monde numérique : automatisation et manipulation aisée des données
- Monde physique : complexité de compréhension et d'automatisation

Impact et Applications de la Vision par Ordinateur

Permet d'étudier et d'influencer le monde physique

- Détection de plants malades dans un champ
- Conduite autonome
- Robotique
- Surveillance industrielle
- Reconnaissance faciale
- Détection d'objets en temps réel

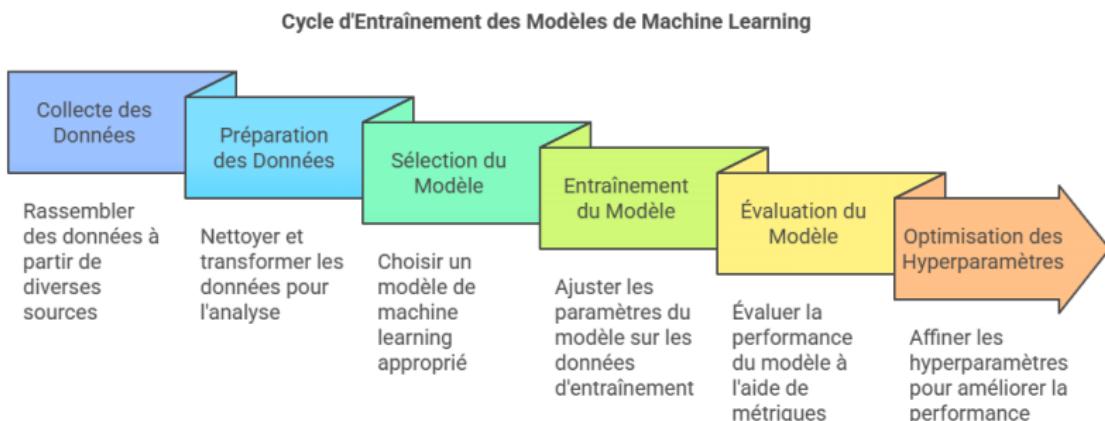


Exemple Concret: Reconnaissance de Fruits et Légumes



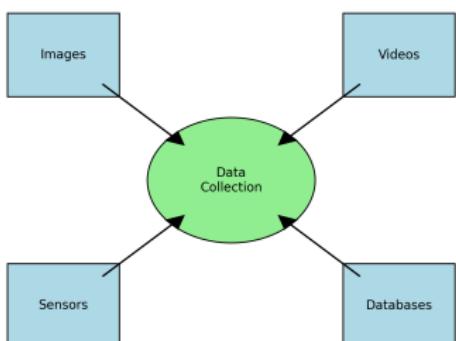
- **Contexte:** Automatisation en grande surface
- **Bénéfices:**
 - Réduction des erreurs de saisie
 - Accélération du processus de pesée
 - Prévention de la fraude
- **Impact:** Amélioration significative de l'expérience client et de l'efficacité opérationnelle

Pipeline d'un Projet de Data Science



- Processus itératif et structuré
- Chaque étape est cruciale pour le succès du projet
- Importance de la validation continue
- Nécessité d'une approche méthodique

Étape 1 : Collecte des Données



• Sources:

- Bases de données publiques
- Images/vidéos collectées
- Données synthétiques générées
- Données externes annotées

• Considérations:

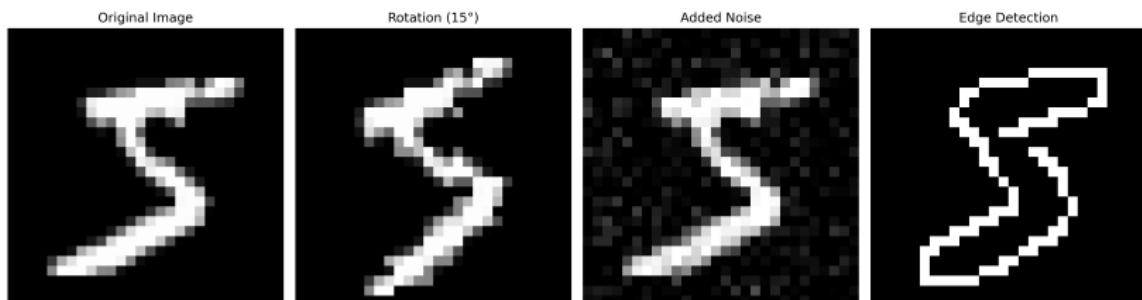
- Qualité et quantité
- Distribution représentative
- Annotations adéquates
- Contraintes éthiques/légales

Étape 2 : Traitement des Données



- **Nettoyage** : Suppression du bruit, correction des erreurs
- **Augmentation** : Génération de nouvelles données synthétiques
- **Feature Engineering** : Extraction des caractéristiques pertinentes
- **Normalisation** : Standardisation des données

Exemple de Traitement des Données : MNIST



• Data Augmentation :

- Rotation : améliore la robustesse aux variations d'orientation
- Bruit : renforce la résistance aux perturbations
- Techniques additionnelles : translation, zoom, flou, retournement, changement de luminosité/contraste ...

• Feature Extraction :

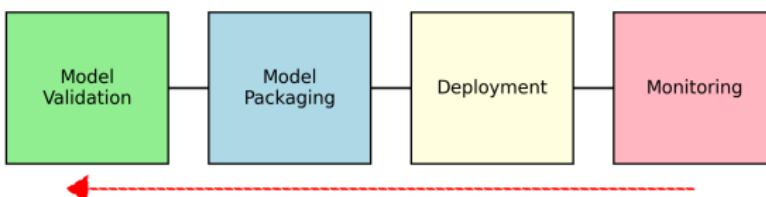
- Détection de contours : met en évidence les caractéristiques structurelles
- Autres techniques : filtres de Gabor, transformée de Fourier, wavelets ...

Étape 3 : Entraînement et Validation

- **Cycle d'entraînement :**

- Choix du modèle
- Définition de la loss
- Optimisation
- Validation

Étape 4 : Déploiement et Production



- **Validation finale** : Tests sur données réelles
- **Packaging** : Containerisation, API REST
- **Déploiement** : Cloud, Edge, On-premise
- **Monitoring** :
 - Surveillance des performances
 - Détection de drift
 - Maintenance continue

Passons aux bases

Représentation Numérique des Images

Qu'est-ce qu'un pixel ?

- **Pixel** = Picture Element
- Plus petite unité d'une image numérique
- En couleur (RGB) :
 - 3 valeurs : Rouge, Vert, Bleu
 - Chaque valeur entre 0 et 255
 - $256^3 \approx 16.7$ millions de couleurs possibles
- En niveaux de gris :
 - 1 seule valeur entre 0 (noir) et 255 (blanc)

Exemple de pixel RGB

RGB = [120, 50, 240]

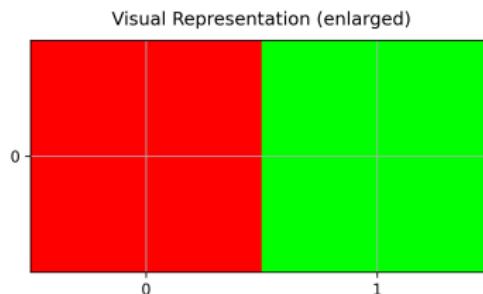
R = 120
G = 50
B = 240



- Un pixel RGB est un triplet de valeurs (R, G, B)
- Chaque canal a une valeur entre 0 et 255
- Combinaison = couleur spécifique
- Exemple : R=120, G=50, B=240 (teinte violette)

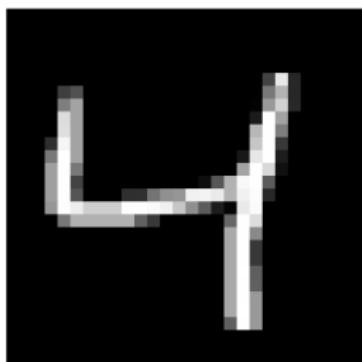
De pixel à image : le plus simple exemple

Image (1x2x3) =
[[[255, 0, 0],
 [0, 255, 0]]]



- Une image est une matrice de pixels
- Le plus simple exemple : une image 1×2
- Dimensions de cette image minimale :
 - Hauteur : 1 pixel
 - Largeur : 2 pixels
 - Profondeur : 3 canaux (RGB)

Exemple : Image MNIST



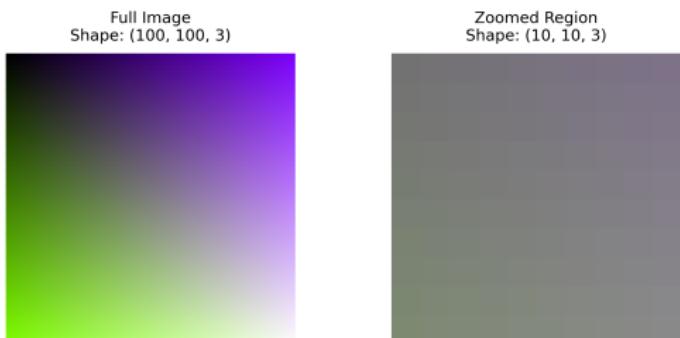
Full matrix shape: 28×28

Center 5×5 subset:

```
[[ 0   0   0   0   0]
 [ 0   0   0  14  86]
 [144 150 241 243 234]
 [240 198 143  91  28]
 [ 0   0   0   0   0]]
```

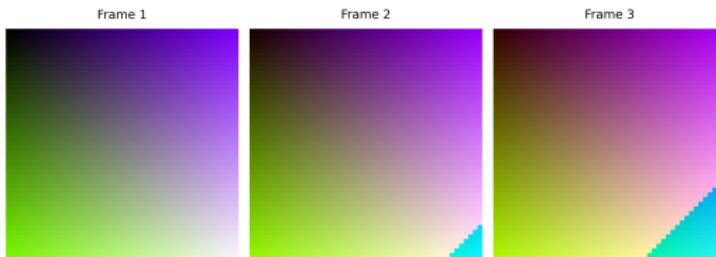
- MNIST : base de données de chiffres manuscrits
- Chaque image est une matrice 28×28 pixels (grayscale)
- Valeurs comprises entre 0 (noir) et 255 (blanc)
- Simple mais suffisant pour illustrer les concepts fondamentaux

Image naturelle haute résolution



- Images réelles : matrices beaucoup plus grandes
- Exemple : photo 4K = matrice $3840 \times 2160 \times 3$
- ≈ 24.9 millions de pixels !

Vidéo : L'ajout de la dimension temporelle



Video = 4D Tensor
Dimensions:
1. Temps (nombre de frames)
2. Hauteur
3. Largeur
4. Canaux (RGB)
Exemple HD 30fps:
→ 30 frames/sec
→ 1920×1080 pixels
→ 3 canaux
= 1.86M valeurs/sec !

- Vidéo = séquence d'images (frames) au cours du temps
- Tenseur 4D : Temps × Hauteur × Largeur × Canaux
- Exemple pour une vidéo HD 30fps, 10 secondes:
 - $30 \times 10 = 300$ frames
 - $300 \times 1920 \times 1080 \times 3 = 1.86$ milliards de valeurs !

Le défi de l'interprétation

- Pour l'ordinateur, une image n'est qu'une matrice de nombres
- Pas de notion intuitive de :
 - Formes
 - Objets
 - Contexte
 - Relations spatiales
- **Challenge de la Vision par Ordinateur :**
 - Extraire du sens de ces matrices de nombres
 - Reproduire (et dépasser) la vision humaine

Techniques traditionnelles

Méthodes Classiques

Classification par Seuillage de Couleur

- **Principe de la classification d'objets :**

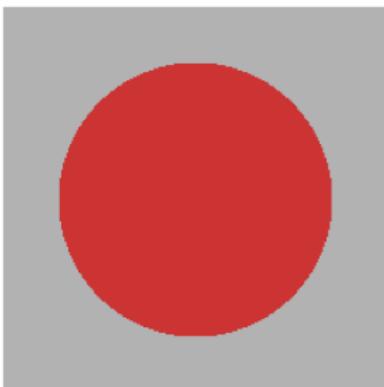
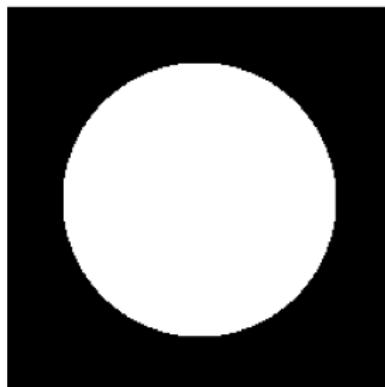
- Identifier et catégoriser des objets dans une image
- Méthode simple : utilisation des caractéristiques de couleur

- **Seuillage de couleur :**

- Définition de plages de valeurs pour chaque canal (R,G,B)
- Création d'un masque binaire
- Avantages : Simple, rapide
- Limites : Sensible aux variations de luminosité, angles, etc.

Exemple de Classification par Seuillage

Image originale

Masque après seuillage
(déttection du rouge)

- **Principe:** Définir un seuil pour séparer les classes
- **Exemple:** Détection d'objets rouges
 - Si $R \geq 0.6$ et $G \leq 0.3$ et $B \leq 0.3 \rightarrow$ objet rouge
 - Sinon \rightarrow fond

Détection de Changements par Soustraction d'Images

- **Principe :**

- Comparaison de deux images similaires
- Soustraction pixel par pixel
- Détection des différences significatives

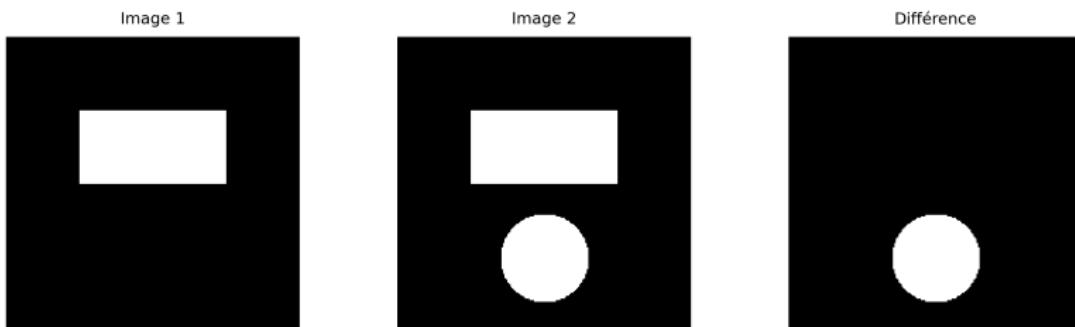
- **Applications :**

- Détection de mouvement
- Surveillance vidéo
- Contrôle qualité industriel

- **Avantages :**

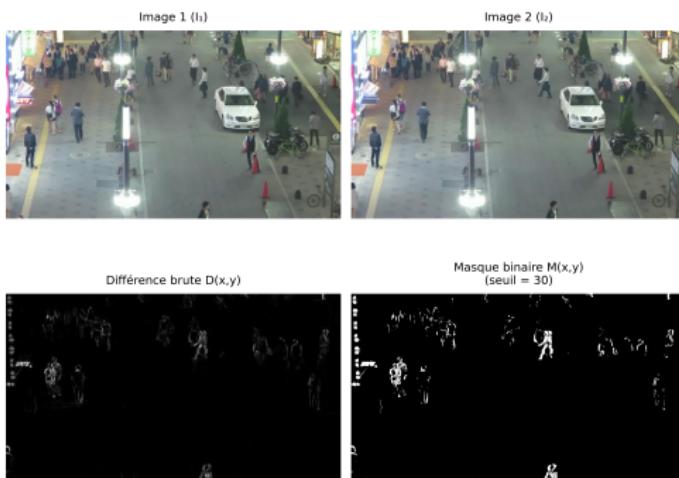
- Simple à mettre en œuvre
- Efficace pour les changements nets
- Peu coûteux en calcul

Exemple Simple de Soustraction d'Images



- **Principe:** Soustraire deux images pour détecter les différences
- **Exemple ci-dessus:**
 - Image 1 : contient un rectangle
 - Image 2 : contient un rectangle + un cercle
 - Différence : fait ressortir uniquement le cercle
- **Applications:** Détection de mouvement, surveillance, suivi d'objets

Limites de la Soustraction Simple - Partie 1



- Exemple avec vidéo réelle (I_1 et I_2 = deux frames)
- Différence brute $D(x,y) = |I_2(x,y) - I_1(x,y)|$
- Masque par seuillage : $M(x,y) = 1$ si $D(x,y) >$ seuil
- **Problèmes observés:** Bruit, trous, zones fragmentées

Limites de la Soustraction Simple - Partie 2

- **Masque de différence brute :**

$$D(x, y) = |I_2(x, y) - I_1(x, y)|$$

- **Masque binaire par seuillage :**

$$M(x, y) = \begin{cases} 1 & \text{si } D(x, y) > \text{seuil} \\ 0 & \text{sinon} \end{cases}$$

- **Problèmes observés :**

- Bruit dans le masque de différence
- Faux positifs après seuillage
- Détections fragmentées

- **Solution proposée :** Post-traitement par opérations morphologiques

Amélioration par Opérations Morphologiques

- **Opérations morphologiques :**

- Transformation d'images basée sur la forme
- Utilisation d'un kernel

- **Processus en deux étapes :**

- **1. Dilatation :** Expansion des objets et connexion des régions

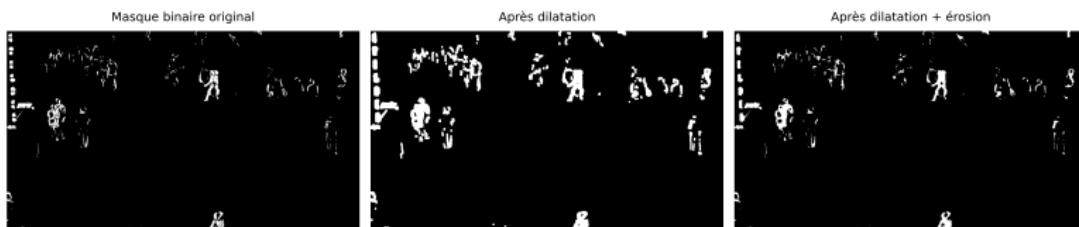
$$[D(M)]_{x,y} = \max_{(i,j) \in K} M_{x+i,y+j}$$

- **2. Érosion :** Suppression du bruit et lissage des contours

$$[E(M)]_{x,y} = \min_{(i,j) \in K} M_{x+i,y+j}$$

où K est l'ensemble des coordonnées du kernel

Application sur le Masque de Soustraction



- **Dilatation:** Agrandit les régions blanches, comble les trous
- **Érosion:** Rétrécit les régions blanches, élimine le bruit
- **Combinaison:** Dilatation suivie d'érosion (fermeture)
- **Résultat:** Objets plus cohérents, moins bruités

Détails des Opérations Morphologiques

● Propriétés :

- Cette combinaison (D puis E) est appelée "fermeture morphologique"
- Fermeture : comble les trous et connecte les régions proches
- Ouverture : élimine les petits objets et le bruit (E puis D)

Opération fondamentale

Convolutions

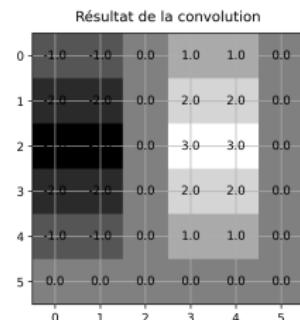
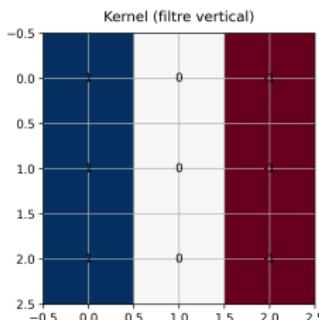
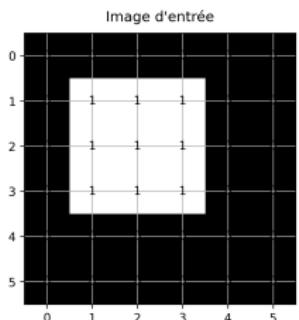
La Convolution : Une Opération Fondamentale

- **Définition** : Opération mathématique combinant deux fonctions
- En vision par ordinateur :
 - Combine une image avec un filtre (kernel)
 - Extrait des caractéristiques spécifiques
 - Base des réseaux de neurones convolutifs
- **Formule mathématique en 2D :**

$$(f * k)(x, y) = \sum_{i=-a}^a \sum_{j=-b}^b f(x - i, y - j)k(i, j)$$

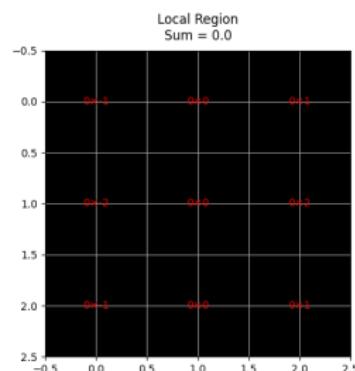
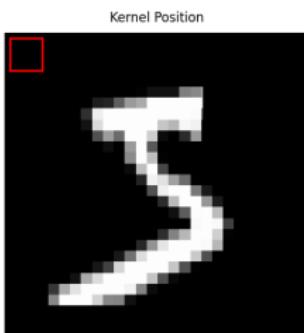
où f est l'image et k est le kernel

Exemple de Convolution 2D



- Convolution = opération fondamentale en CV
- Exemple ci-dessus avec filtre vertical simple
- Chaque position : somme pondérée du voisinage et du noyau
- Résultat : détection des transitions verticales

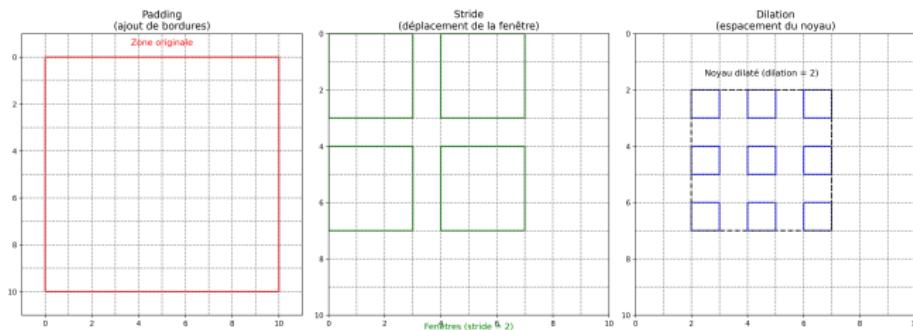
Animation du Processus de Convolution



- Le noyau glisse sur l'image d'entrée
- À chaque position : multiplication élément par élément et somme
- L'exemple utilise un filtre de Sobel vertical (déetecte les bords verticaux)
- Résultat : carte d'activation des contours verticaux

Paramètres d'une Convolution

- **Taille du noyau:** contrôle la zone d'influence (3×3 , 5×5 , $7 \times 7 \dots$)
- **Padding:** Préservation ou non des dimensions de l'image
- **Stride:** Pas de déplacement du noyau (généralement 1 ou 2)
- **Dilation:** Espacement entre les éléments du noyau

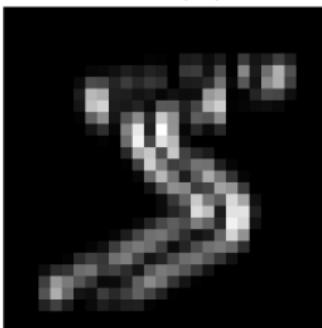


Filtres de Convolution Classiques

Image originale



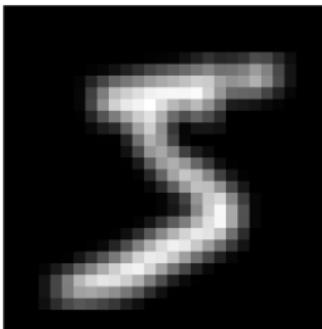
Filtre vertical (Sobel)



Filtre horizontal (Sobel)



Filtre de flou (moyenne)



Kernels utilisés :

Sobel vertical :

$$\begin{matrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{matrix}$$

Sobel horizontal :

$$\begin{matrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{matrix}$$

Flou (moyenne) :

$$\frac{1}{9} \begin{matrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{matrix} \frac{1}{9}$$

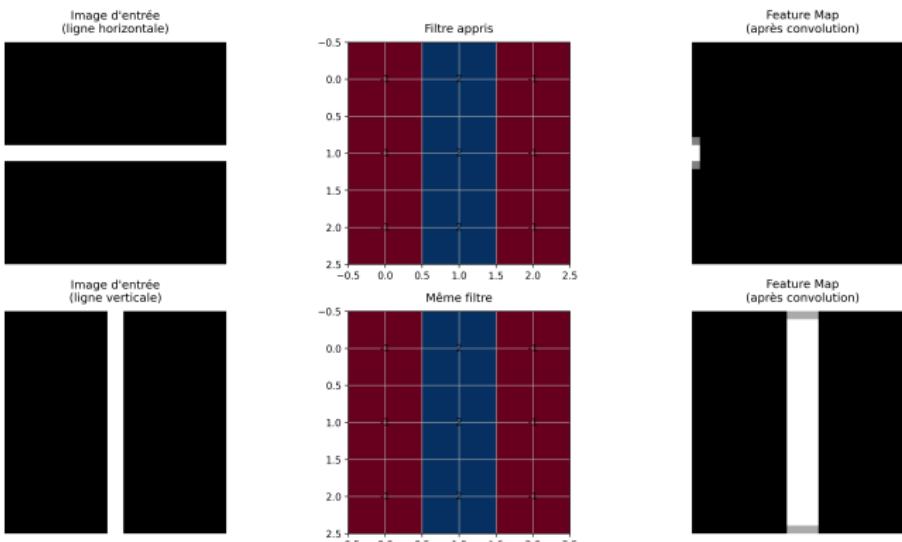
Apprentissage des convolutions

Réseaux CNN

Des Filtres Fixes aux Filtres Appris

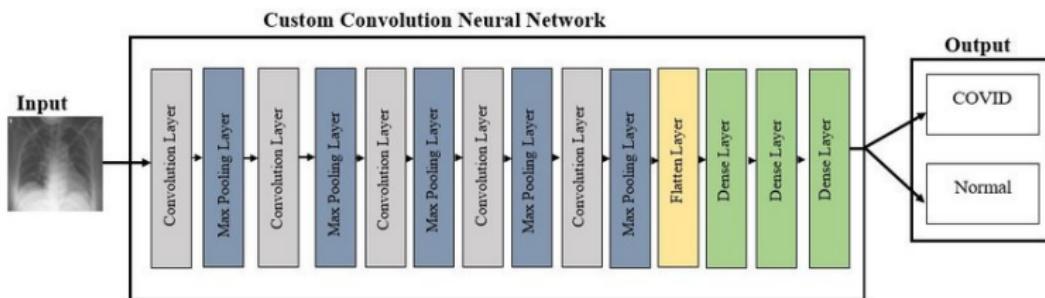
- **Limitation des filtres prédefinis :**
 - Choix manuel des filtres
 - Pas d'adaptation aux données
 - Nombre limité de features extraites
- **Solution : Apprentissage automatique des filtres**
 - Les filtres deviennent des paramètres à optimiser
 - Adaptation aux caractéristiques importantes des données
 - Minimisation d'une fonction de perte

Exemple Simple : Détection de Lignes Verticales



- **Observation** : Le CNN apprend des filtres adaptés aux tâches
- **Exemple** : Filtre appris pour la détection de lignes verticales
- **Résultat** : Activation forte pour les lignes verticales, faible pour horizontales

Vers des Architectures Plus Profondes



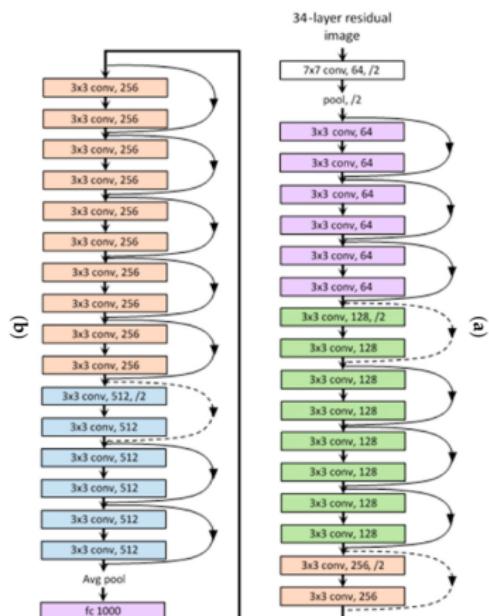
- **Avantages de la profondeur :**

- Extraction de caractéristiques plus complexes
- Hiérarchie de features (des simples aux complexes)
- Meilleure capacité de généralisation

Le Rôle du Pooling

- **Objectifs du pooling :**
 - Réduction de la dimension spatiale
 - Invariance aux petites translations
 - Réduction du nombre de paramètres
- **Types de pooling :**
 - Max pooling : valeur maximale de la région
 - Average pooling : moyenne de la région
- **Paramètres :**
 - Taille de la fenêtre (généralement 2×2)
 - Stride (généralement égal à la taille)

ResNet : L'Innovation des Skip Connections



Avantages :

- Facilite l'entraînement des réseaux profonds
- Atténue le problème de la disparition du gradient
- Permet l'apprentissage résiduel

Pourquoi les Skip Connections ?

- **Problème des réseaux profonds :**

- Difficulté d'apprentissage
- Dégradation des performances

- **Solution ResNet :**

- $F(x) + x$ au lieu de $F(x)$
- Si $F(x) = 0$, on préserve l'identité
- Apprentissage des résidus

- **Impact :**

- Entraînement plus stable
- Convergence plus rapide
- Meilleures performances

Ressources et Questions

Retrouvez tous les supports de cours sur GitHub:



[github.com/fvalade/Computer-vision-
Course-Materials](https://github.com/fvalade/Computer-vision-Course-Materials)

Des questions ?