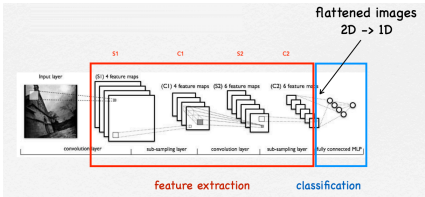


CNN

Les **convolutional neural networks** sont des réseaux de neurones utilisant des opérations de convolution pour extraire des caractéristiques (features) des données d'entrées, en particulier des images.

Couches



Convolutionnelles (S1, S2)

Les couches dites **convolutionnelles** appliquent des filtres (kernel) sur les données d'entrée pour produire des cartes de caractéristiques (feature maps). Ces filtres permettent de détecter des motifs type: **bordure, textures, etc..**

Sous-échantillonnage (C1, C2)

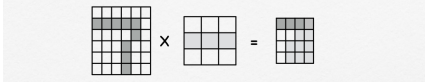
Les couches de sous-échantillonnage réduisent les dimensions des cartes de caractéristiques en appliquant des opérations comme le **max pooling** ou l'**average pooling**. Cela permet de réduire le nombre de paramètres et de contrôler le sur-ajustement

Entièrement connectées (classification)

À la fin des couches convolutionnelles et de pooling, les caractéristiques extraites sont aplaties et passées à travers des couches entièrement connectées (comme dans un MLP) pour effectuer la classification ou la régression.

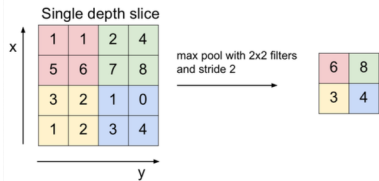
Convolution

Permet de détecter des motifs sur toutes les images d'entrées



Maxpooling

Statistiques récapitulatives de la détection des motifs



CNN shallow

- **Structure** : 1 à 3 couches convolutionnelles et de pooling.
- **Utilisation** : Pour des tâches simples avec des jeux de données de petite taille ou des motifs simples.
- **Avantages** : Moins de paramètres, réduisant le risque de sur-ajustement. Entraînement rapide et moins exigeant en termes de ressources.
- **Inconvénients** : Capacité limitée à capturer des motifs complexes. Moins performant pour des tâches complexes de vision par ordinateur.

CNN deep

- **Structure** : Plus de 10 couches convolutionnelles et de pooling, parfois des centaines (e.g., ResNet, VGG).
- **Utilisation** : Pour des tâches complexes avec des jeux de données volumineux ou des motifs complexes.
- **Avantages** : Capacité supérieure à capturer des motifs complexes et des relations hiérarchiques. Excellente performance sur des tâches complexes de vision par ordinateur.

- **Inconvénients** : Entraînement long et exigeant en termes de ressources. Plus de paramètres, augmentant le risque de sur-ajustement et nécessitant des techniques de régularisation.

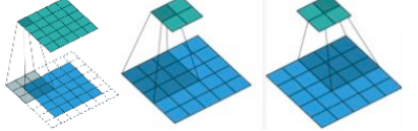
Fonction d'activation

ReLU	Leaky ReLU
$g(z) = \max(0, z)$	$g(z) = \max(0.01 \times z, z)$

Stride et padding

- **Stride**: déplacement du filtre lors de la convolution. Un stride de 1 signifie que le filtre se déplace d'un pixel à la fois, tandis qu'un stride de 2 signifie qu'il se déplace de deux pixels.
- **Padding**: ajout de pixels autour de l'image pour contrôler la taille de la sortie.

Zero-Padding / Stride = 2



CNN Keras

10 = Input(shape=(height,width, 3), name='input')	Model: "model_test"
11 = Convolution2D(16, 3, 3, strides=(1,1), padding='same', activation='relu', name='l1')(input)	Layer (type) Output Shape Param #
12 = MaxPooling2D(pool_size=(2, 2), name='l1_pool')(l1)	InputLayer (None, 28, 28, 3) 0
13 = Convolution2D(32, 3, 3, strides=(1,1), padding='same', activation='relu', name='l2')(l1_pool)	Conv2D (None, 26, 26, 16) 448
14 = MaxPooling2D(pool_size=(2, 2), name='l2_pool')(l2)	MaxPooling2D (None, 13, 13, 16) 0
15 = Convolution2D(64, 3, 3, strides=(1,1), padding='same', activation='relu', name='l3')(l2_pool)	Conv2D (None, 24, 24, 32) 4608
16 = MaxPooling2D(pool_size=(2, 2), name='l3_pool')(l3)	MaxPooling2D (None, 12, 12, 32) 0
17 = Flatten(name='flat')(l3_pool)	Conv2D (None, 7, 7, 64) 18496
18 = Dense(1000, activation='relu', name='l4')(l3_pool)	MaxPooling2D (None, 3, 3, 64) 0
19 = Dense(1000, activation='relu', name='l5')(l4)	Flatten (None, 128) 0
20 = Dense(10, activation='softmax', name='output')(l5)	Dense (None, 1000) 128000
model = Model(input, output)	Dense (None, 10) 101
model.summary()	Total params: 38,272

1. Taille de l'image traitée : 28x28 pixels
2. Nb image sortie L1 vers L2 = 16
3. Taille image L2 = 3x3 pixels
4. Nb filtres L2 = 32 -> chaque filtre à 3x3 coefficients donc 9
5. Dernière couche L3, il y a 64 images de 7x7 pixels
6. Nb neurones cachés en L4 Dense = 25
7. Nb poids reliant L4 à L5 = 14425

Nb paramètres en L1
• il y a un filtre de 5x5 pixels (9), avec 16 images d'entrée (10) + 16 donc 5 * 5 * 16 + 16 = 416

Nb paramètres en L4
• il y a 576 data entrant (8) qui vont dans 25 neurones (6) et chaque neurone a un biais donc 576 * 25 + 25 = 14425

CNN architecture

LeNet-5

- **Quoi** : Une des premières architectures CNN, conçue pour la reconnaissance de chiffres manuscrits
- **Comment** : Utilisé pour des tâches de reconnaissance de motifs simples comme les chiffres
- **Contexte** : Tâches de reconnaissance de caractères manuscrits (ex. codes postaux)
- **Pourquoi** : Simplicité et efficacité pour les tâches simples avec des données limitées

AlexNet (2012)

- **Quoi** : Un CNN plus profond qui a gagné le défi ILSVRC 2012
- **Comment** : Utilisé pour des tâches de classification d'images à grande échelle
- **Contexte** : Concours de vision par ordinateur, applications nécessitant une bonne précision
- **Pourquoi** : Bonne performance pour les tâches complexes grâce à l'utilisation de ReLU et du dropout

ZF Net (2013)

- **Quoi** : Une amélioration d'AlexNet, avec des filtres plus petits et plus de couches

- **Comment** : Classification d'images et détection d'objets
- **Contexte** : Tâches nécessitant une analyse fine des images
- **Pourquoi** : Meilleure performance que AlexNet sur les tâches complexes

VGG Net (2014)

- **Quoi** : Un CNN avec une architecture simple mais profonde utilisant des filtres 3x3
- **Comment** : Classification d'images, segmentation d'images
- **Contexte** : Recherche en vision par ordinateur, applications nécessitant une haute précision
- **Pourquoi** : Simplicité de conception et bonne performance sur des tâches complexes grâce à sa profondeur

GoogLeNet (Inception, 2015)

- **Quoi** : Un CNN avec des modules d'Inception qui permettent de capturer des informations à différentes échelles
- **Comment** : Classification d'images, reconnaissance d'objets
- **Contexte** : Applications nécessitant une haute précision et une efficacité computationnelle
- **Pourquoi** : Efficacité et capacité à capturer des motifs à différentes échelles

ResNet (2015)

- **Quoi** : Un CNN très profond avec des connexions résiduelles pour éviter la dégradation des performances
- **Comment** : Classification d'images, détection d'objets, segmentation d'images
- **Contexte** : Tâches très complexes nécessitant des réseaux profonds
- **Pourquoi** : Permet l'entraînement de réseaux très profonds sans dégradation des performances grâce aux connexions résiduelles

DenseNet (2017)

- **Quoi** : Un CNN où chaque couche est connectée à toutes les couches précédentes
- **Comment** : Classification d'images, détection d'objets, segmentation d'images
- **Contexte** : Applications nécessitant une propagation efficace des caractéristiques
- **Pourquoi** : Réduction du nombre de paramètres et meilleure propagation des caractéristiques

EfficientNet (2020)

- **Quoi** : Une famille de CNN optimisés pour l'efficacité computationnelle en équilibrant profondeur, largeur et résolution
- **Comment** : Classification d'images, détection d'objets, segmentation d'images
- **Contexte** : Applications nécessitant une haute précision et une efficacité computationnelle
- **Pourquoi** : Optimisation des ressources tout en maintenant une haute performance

Transfert learning

Principe du Transfer Learning

Réutiliser un modèle pré-entraîné (ex: sur ImageNet) pour une nouvelle tâche. Motivation : Un modèle entraîné sur une grande base de données a appris des caractéristiques générales utiles pour d'autres tâches.

Étapes du Transfer Learning

1. Chargement du modèle pré-entraîné

Utiliser un modèle entraîné sur une grande base (ImageNet : 1000 objets, 1000 images/catégorie) Garder l'architecture des couches convolutionnelles

2. Freeze des couches

Geler les premières couches (caractéristiques générales) Les poids ne sont pas mis à jour pendant l'entraînement Garde les détecteurs de caractéristiques de bas niveau

3. Modification de la sortie

Remplacer la dernière couche dense Adapter au nombre de classes de la nouvelle tâche Exemple :

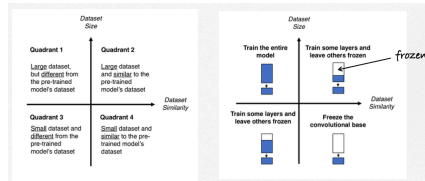
ImageNet (1000 classes) → COVID detection (2 classes)

4. Fine-tuning

Option A : Entraîner seulement les dernières couches
Option B : Dégeler quelques couches et entraîner avec un petit learning rate
Stratégie : Plus on a de données, plus on peut dégeler de couches

Avantages du Transfer Learning

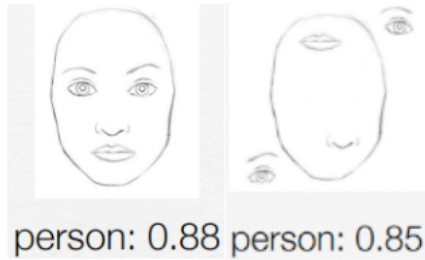
Moins de données nécessaires : Peut fonctionner avec des milliers au lieu de millions d'images
Convergence plus rapide : Initialisation intelligente des poids
Meilleures performances sur petits datasets
Économie de ressources : Pas besoin de réentraîner depuis zéro



Deep troubles

Avantages des CNN

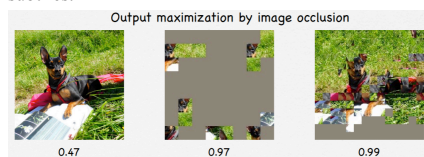
Offre une invariance à la translation spatiale. Il peut donc reconnaître un objet avec des tailles différentes même avec des positions « décalées ».



Corrélations “brute force”

Les CNN ne “comprennent” pas les données comme les humains, mais détectent des caractéristiques spécifiques pour prendre des décisions. Cela peut parfois les tromper.

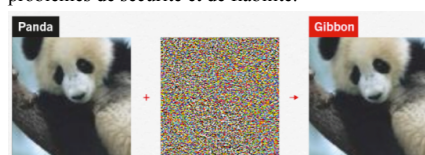
Les CNN peuvent être trompés par des caractéristiques anormales ou bruitées. Ils détectent des motifs sans comprendre le contexte global, ce qui les rend vulnérables à des manipulations subtiles.



Attaques Adversariales

Les CNN peuvent être trompés par des images modifiées de manière imperceptible pour les humains mais classées avec une grande confiance par le modèle.

Les attaques adversariales exploitent les faiblesses des CNN en ajoutant des perturbations spécifiques pour changer la classification. Cela pose des problèmes de sécurité et de fiabilité.



Techniques Contre le Surapprentissage

1. Réduction du Nombre de Paramètres

Principe : Modèle plus simple → moins de risque de surapprentissage
Méthodes :

Réduire le nombre de couches
Réduire le nombre de filtres par couche
Utiliser des filtres plus petits
Architectures plus efficaces (EfficientNet)

2. Data Augmentation

Transformations géométriques :

Rotation : $[x', y'] = [x \cos \theta - y \sin \theta, x \sin \theta + y \cos \theta]$
Translation : $[x', y'] = [x + \Delta x, y + \Delta y]$
Zoom/Recadrage et retournement

Transformations colorimétriques :

Modification luminosité/contraste
Perturbation des canaux RGB
Ajout de bruit

Effet : Augmente artificiellement la taille du dataset d'entraînement

3. Dropout

Principe : Désactiver aléatoirement des neurones pendant l'entraînement
Implémentation : Avec probabilité p (ex: 0.5), mettre le neurone à 0
Effet : Force le réseau à ne pas dépendre d'un seul neurone, améliore la généralisation
Usage : Principalement dans les couches denses, moins dans les couches convolutionnelles

4. Early Stopping

Principe : Arrêter l'entraînement quand la performance sur validation cesse de s'améliorer
Surveillance : Suivre l'erreur de validation à chaque époque
Implémentation : Patience (nombre d'époques sans amélioration avant arrêt)

5. Régularisation L1/L2

L1 : $\text{Loss} = \text{Loss}_{\text{original}} + \lambda \sum |w_i|$
L2 : $\text{Loss} = \text{Loss}_{\text{original}} + \lambda \sum w_i^2$
Effet : Pénalise les poids trop importants, favorise la simplicité

Application deep learning

Types de Problèmes

Classification / Reconnaissance / Identification

Exemples : Détection de cancer, reconnaissance de chiffres, filtrage de spam, identification de visages et de commandes vocales.

But : Attribuer une classe ou une étiquette spécifique à chaque entrée de données.

Régression

Exemples : Prédiction de variables continues comme l'âge, les scores de plongée, le rendement agricole.

But : Prédire une valeur continue à partir des données d'entrée.

Réseaux de Neurones Entièrement Connectés vs. Convolutifs

Réseaux de Neurones Entièrement Connectés (Fully Connected Networks)

Utilisation : Traiter des données où chaque caractéristique est indépendante.

Exemples : Données tabulaires, régression, filtrage collaboratif pour recommandations.

Réseaux de Neurones Convolutifs (CNN)

Utilisation : Traiter des données avec des relations spatiales.

Exemples : Images, séries temporelles (sons, données de capteurs), vidéos.

Types de Convolutions

1D Convolutions : Utilisées pour les séries temporelles.

3D Convolutions : Utilisées pour les vidéos.

Helpful ressources

Calculer le nombre de paramètres d'un MLP

Dans le cas où nous souhaitons calculer le nombre de paramètres d'un MLP pour traiter des images de 10 digits, 28x28 pixels, niveaux de gris. Le MLP est composé d'une seule couche cachée de 25 neurones:

