Réseaux de Neurones Convolutionnels

ARN

6 - Convolutional Neural Networks

Abstract

Table des matières

1. Réseaux de Neurones Convolutionnels (CNN)

1.1. Problématique de la Reconnaissance d'Objets

La reconnaissance d'objets par ordinateur constitue un défi majeur en intelligence artificielle. Les approches traditionnelles se heurtent à plusieurs obstacles fondamentaux.

Approche Naïve	Approche CNN
Templates fixes	Caractéristiques adaptatives
Sensible à la position	Invariance spatiale
Pas de variabilité	Gestion de la variabilité
Correspondance exacte	Détection de motifs flexibles

1.1.1. Le Défi de la Variabilité Intra-Classe

Problème central : Les objets appartenant à une même catégorie présentent une énorme diversité d'apparences. Par exemple, tous les stades d'athlétisme ne ressemblent pas exactement au même modèle idéal.

Facteurs de variabilité :
- **Position** : l'objet peut apparaître n'importe où dans l'image
- **Échelle** : taille variable selon la distance de prise de vue
- **Orientation** : rotation possible dans l'espace
- **Éclairage** : conditions lumineuses changeantes
- **Occlusion partielle** : parties de l'objet cachées

Solution adoptée : Au lieu d'utiliser des gabarits rigides, il faut identifier des **caractéristiques discriminantes** robustes : couleurs typiques, textures caractéristiques, formes géométriques particulières, présence de contours spécifiques.

1.2. Inspiration Biologique : Le Système Visuel

1.2.1. Architecture du Cortex Visuel des Mammifères

Le système visuel des mammifères traite l'information de manière hiérarchique et distribuée :

Voie ventrale (reconnaissance d'objets) :
- **Rétine** (20-40 ms) : Capture des photons
- **V1** (40-60 ms) : Détection des contours simples, barres orientées
- **V2** (50-70 ms) : Formes visuelles intermédiaires
- **V4** (70-90 ms) : Groupes de caractéristiques, couleurs
- **IT** (80-100 ms) : Descriptions d'objets de haut niveau

1.2.2. Les Découvertes Révolutionnaires de Hubel & Wiesel

Contexte historique : En 1959, David Hubel et Torsten Wiesel mènent des expériences sur des chats anesthésiés en enregistrant l'activité de neurones individuels du cortex visuel.

Découverte fortuite : Ils découvrent par hasard qu'un neurone ne réagit pas aux points lumineux (qui activent la rétine) mais uniquement aux barres orientées dans une direction précise.

Implications :
- Chaque neurone possède un **champ récepteur** spécialisé
- Les neurones détectent des motifs géométriques spécifiques
- Prix Nobel de médecine 1981 pour cette découverte fondamentale

1.2.3. Exemples de Spécialisation Neuronale

Neurone "Jennifer Aniston" : Des chercheurs ont identifié dans l'hippocampe humain des neurones qui se déclenchent spécifiquement en voyant cette actrice, démontrant l'existence de neurones très spécialisés.

Paréidolie faciale : La région FFA (fusiform face area) détecte les visages même dans des objets inanimés (prises électriques, voitures), illustrant la puissance des détecteurs de motifs biologiques.

1.2.4. Principes Architecturaux Biologiques

1. **Traitement local → global** : extraction progressive de caractéristiques de plus en plus complexes
2. **Hiérarchie de complexité** : des edges simples aux objets complets
3. **Invariance progressive** : tolérance croissante aux transformations géométriques
4. **Parallélisme massif** : traitement simultané en de multiples endroits

1.3. Mathématiques de la Convolution

1.3.1. Principe Fondamental

La convolution est une opération mathématique qui applique un filtre (noyau) sur une image pour extraire des caractéristiques spécifiques.

Formule mathématique générale : $(f * g)(x, y) = \sum_{i=-\infty}^{\infty} \sum_{j=-\infty}^{\infty} f(i, j) \times g(x - i, y - j)$

En pratique pour un noyau 3×3 : $(f * g)(x, y) = \sum_{i=-1}^{1} \sum_{j=-1}^{1} f(x + i, y + j) \times g(i, j)$

Algorithme étape par étape :

1. Placer le noyau 3×3 sur une région de l'image
2. Multiplier élément par élément
3. Sommer tous les produits
4. Le résultat devient la valeur du pixel de sortie
5. Déplacer le noyau et répéter

1.3.2. Bibliothèque de Noyaux Classiques

Détection horizontale : $K_{\text{horizontal}} = \begin{pmatrix} 0 & 0 & 0 \\ \frac{1}{3} & \frac{1}{3} & \frac{1}{3} \\ 0 & 0 & 0 \end{pmatrix}$

Effet : Met en évidence les lignes horizontales

Détection verticale : $K_{\text{vertical}} = \begin{pmatrix} 0 & \frac{1}{3} & 0 \\ 0 & \frac{1}{3} & 0 \\ 0 & \frac{1}{3} & 0 \end{pmatrix}$

Effet : Met en évidence les lignes verticales

Détection de contours (Laplacien) : $K_{\text{edge}} = \begin{pmatrix} 1 & 1 & 1 \\ 1 & -8 & 1 \\ 1 & 1 & 1 \end{pmatrix}$

Effet : Détecte les changements brusques d'intensité

Flou gaussien : $K_{\text{blur}} = \frac{1}{16} \begin{pmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{pmatrix}$

Effet : Lisse l'image en réduisant le bruit

1.3.3. Exemple Pratique de Convolution

Soit une image 5×5 et un noyau de détection horizontale 3×3 :

$$I = \begin{pmatrix} 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{pmatrix} \quad \text{et} \quad K = \begin{pmatrix} 0 & 0 & 0 \\ \frac{1}{3} & \frac{1}{3} & \frac{1}{3} \\ 0 & 0 & 0 \end{pmatrix}$$

Calcul en position (1,1) :

$O(1,1) = \sum_{i=0}^{2} \sum_{j=0}^{2} I(1 + i, 1 + j) \times K(i, j)$

$= I(1,1) \times K(0,0) + I(1,2) \times K(0,1) + I(1,3) \times K(0,2) + I(2,1) \times K(1,0) + I(2,2) \times K(1,1) + I(2,3) \times K(1,2) + I(3,1) \times K(2,0) + I(3,2) \times K(2,1) + I(3,3) \times K(2,2)$

$= 0 \times 0 + 0 \times 0 + 0 \times 0 + 1 \times \frac{1}{3} + 1 \times \frac{1}{3} + 1 \times \frac{1}{3} + 0 \times 0 + 0 \times 0 + 0 \times 0$

$= 0 + 0 + 0 + \frac{1}{3} + \frac{1}{3} + \frac{1}{3} + 0 + 0 + 0 = 1$

1.4. Architecture Complète d'un CNN

1.4.1. Vue d'Ensemble du Processus

Un CNN moderne suit généralement cette architecture en cascade :

Image d'entrée → [Conv + ReLU → Pool] × N → Flatten → Dense → Softmax → Classes

Phase d'extraction de caractéristiques :
- **Couches convolutionnelles** : détection de motifs locaux
- **Fonctions d'activation** : introduction de non-linéarité (ReLU)
- **Couches de pooling** : réduction dimensionnelle et invariance

Phase de classification :
- **Aplatissement** : conversion 2D → 1D
- **Couches denses** : combinaison des caractéristiques
- **Couche de sortie** : probabilités par classe

1.4.2. Exemple Concret : LeNet-5 (1989)

Architecture historique de Yann LeCun pour la reconnaissance de chiffres manuscrits :

$$32 \times 32 \rightarrow \text{Conv}(6 \times 5 \times 5) \rightarrow \text{Pool}(2 \times 2) \rightarrow \text{Conv}(16 \times 5 \times 5) \rightarrow \text{Pool}(2 \times 2) \rightarrow \text{Dense}(120) \rightarrow \text{Dense}(84) \rightarrow \text{Dense}(10)$$

Innovation : Premier réseau convolutionnel appliqué avec succès à un problème réel (tri automatique du courrier postal).

1.5. Fonctionnement Détaillé des Couches

1.5.1. Couches de Convolution : Le Cœur du Système

Objectif : Apprendre automatiquement les filtres optimaux pour chaque tâche spécifique, sans programmation manuelle.

Mécanisme d'apprentissage :
- Les poids du noyau sont initialisés aléatoirement
- L'algorithme de rétropropagation ajuste ces poids
- Le réseau découvre les motifs pertinents dans les données

Exemple pour la détection de chats :
- **Première couche** : détection d'edges, coins, textures de base
- **Deuxième couche** : yeux, oreilles, formes arrondies
- **Troisième couche** : visages complets, postures typiques

1.5.2. Fonction d'Activation ReLU

Définition mathématique : $\text{ReLU}(x) = \max(0, x) = \begin{cases} x \text{ si } x > 0 \\ 0 \text{ si } x \leq 0 \end{cases}$

Avantages :
- **Simplicité** : calcul très rapide
- **Parcimonie** : de nombreuses activations sont nulles
- **Gradient non-saturant** : évite le problème du gradient qui disparaît
- **Inspiration biologique** : les neurones biologiques ne transmettent que des signaux positifs

1.5.3. Partage de Poids : Révolution Conceptuelle

Principe : Le même filtre est appliqué à toutes les positions de l'image.

Formulation mathématique : Pour un filtre W et une image I, la sortie en position (x, y) est : $O(x, y) = \sum_{i,j} I(x+i, y+j) \times W(i, j) + b$

Conséquences :
- **Réduction drastique des paramètres** : un filtre 3×3 au lieu d'une matrice complète
- **Invariance par translation** : une caractéristique est détectée peu importe sa position

- **Généralisation** : ce qui est appris dans une région s'applique partout

1.5.4. Traitement des Images Couleur (RGB)

Pour une image couleur, chaque filtre possède 3 plans correspondant aux canaux R, G, B :

Calcul pour un pixel de sortie : $O(x,y) = \sum_{i,j}\left[R(x+i,y+j) \times W_{R(i,j)} + G(x+i,y+j) \times W_{G(i,j)} + B(x+i,y+j) \times W_{B(i,j)}\right] + b$

Forme vectorielle compacte : $O(x,y) = \langle \begin{pmatrix} R \\ G \\ B \end{pmatrix}, \begin{pmatrix} W_R \\ W_G \\ W_B \end{pmatrix} \rangle + b$

Interprétation : Le filtre peut apprendre à détecter des combinaisons spécifiques de couleurs (ex: le vert de l'herbe, le bleu du ciel).

1.6. Couches de Sous-échantillonnage (Pooling)

1.6.1. Max Pooling : Principe et Calcul

Mécanisme : Dans chaque région 2×2, ne conserver que la valeur maximale.

Formulation mathématique : $\text{MaxPool}(I)(x,y) = \max\{I(2x,2y), I(2x+1,2y), I(2x,2y+1), I(2x+1,2y+1)\}$

Exemple complet :

$$I = \begin{pmatrix} 1 & 1 & 2 & 4 \\ 5 & 6 & 7 & 8 \\ 3 & 2 & 1 & 0 \\ 1 & 2 & 3 & 4 \end{pmatrix} \longrightarrow \text{MaxPool}_{2\times2(I)} = \begin{pmatrix} 6 & 8 \\ 3 & 4 \end{pmatrix}$$

Détail des calculs par région :
- **Région 1** : $\max\{1,1,5,6\} = 6$
- **Région 2** : $\max\{2,4,7,8\} = 8$
- **Région 3** : $\max\{3,2,1,2\} = 3$
- **Région 4** : $\max\{1,0,3,4\} = 4$

Avantages multiples :
- **Réduction de taille** : image divisée par 4 (2×2)
- **Invariance locale** : petites translations sans effet
- **Extraction de l'essentiel** : seules les activations fortes survivent
- **Réduction de calcul** : moins de données pour les couches suivantes

1.6.2. Alternative : Average Pooling

Formulation mathématique : $\text{AvgPool}(I)(x,y) = \frac{1}{4}[I(2x,2y) + I(2x+1,2y) + I(2x,2y+1) + I(2x+1,2y+1)]$

Usage : Moins courant car perd l'information des activations fortes.

1.7. Phase de Classification

1.7.1. Aplatissement (Flattening)

Nécessité : Les couches denses ne peuvent traiter que des vecteurs 1D.

Transformation mathématique : Pour des matrices de features $M_1, M_2, ..., M_n$:

$$\text{Flatten}(M_1, M_2, ..., M_n) = \begin{pmatrix} m_{1,1,1} \\ m_{1,1,2} \\ \cdots \\ m_{1,i,j} \\ \cdots \\ m_{n,i,j} \end{pmatrix}$$

Exemple concret : $\begin{pmatrix} a & b \\ c & d \end{pmatrix}, \begin{pmatrix} e & f \\ g & h \end{pmatrix} \longrightarrow \begin{pmatrix} a \\ b \\ c \\ d \\ e \\ f \\ g \\ h \end{pmatrix}$

Conservation de l'information : Aucune perte, simple réorganisation des données.

1.7.2. Couches Entièrement Connectées

Architecture classique : Perceptron multi-couches traditionnel.

Fonctionnement mathématique : $(y) = f(W(x) + (b))$

Où :
- (x) : vecteur d'entrée (features aplaties)
- W : matrice de poids
- (b) : vecteur de biais
- f : fonction d'activation (ReLU, Softmax)

Fonction Softmax pour la classification : $\text{Softmax}(z_i) = \frac{e^{z_i}}{\sum_{j=1}^{K} e^{z_j}}$

Rôle : Combiner les caractéristiques détectées pour prendre une décision finale.

1.8. Apprentissage par Rétropropagation

1.8.1. Fonction de Coût

Classification multi-classes : Entropie croisée $L = -\frac{1}{N}\sum_{n=1}^{N}\sum_{c=1}^{C} y_{n,c}\log(\hat{y}_{n,c})$

Où :
- N : nombre d'exemples
- C : nombre de classes
- $y_{n,c}$: vraie étiquette (0 ou 1)
- $\hat{y}_{n,c}$: prédiction du modèle

1.8.2. Optimisation par Gradient

Règle de mise à jour générale : $\theta_{t+1} = \theta_t - \eta\nabla_\theta L(\theta_t)$

Où :
- θ : paramètres du modèle (poids, biais)
- η : taux d'apprentissage
- $\nabla_\theta L$: gradient de la fonction de coût

Algorithme Adam : Version avancée avec adaptation automatique : $m_t = \beta_1 m_{t-1} + (1-\beta_1)\nabla_\theta L$ $v_t = \beta_2 v_{t-1} + (1-\beta_2)(\nabla_\theta L)^2$ $\theta_{t+1} = \theta_t - \frac{\eta}{\sqrt{\hat{v}_t}+\varepsilon}\hat{m}_t$

1.8.3. Défis de l'Entraînement

Surapprentissage : Le modèle mémorise les données d'entraînement.

Solution : Dropout avec probabilité p de désactivation aléatoire des neurones.

Temps de calcul : Entraînement très coûteux.

Solution : GPU/TPU, architectures optimisées.

1.9. Implémentation Pratique avec Keras

1.9.1. Code Complet Commenté

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

```
# Construction de l'architecture
model = Sequential([
    # Première couche convolutionnelle
    Conv2D(32, (3,3), activation='relu', input_shape=(28,28,1)),
    # 32 filtres de taille 3×3, images 28×28 en niveaux de gris

    # Deuxième couche convolutionnelle
    Conv2D(32, (3,3), activation='relu'),
    # Mêmes paramètres, détection de motifs plus complexes

    # Sous-échantillonnage
    MaxPooling2D(pool_size=(2,2)),
    # Réduction par facteur 2 en largeur et hauteur

    # Régularisation contre le surapprentissage
    Dropout(0.25),
    # 25% des neurones désactivés aléatoirement

    # Passage aux couches denses
    Flatten(),
    # Conversion matrices → vecteur

    # Classification
    Dense(128, activation='relu'),
    # 128 neurones entièrement connectés

    Dropout(0.5),
    # Régularisation plus forte avant la sortie

    # Couche de sortie
    Dense(10, activation='softmax')
    # 10 classes (chiffres 0-9), probabilités sommant à 1
])

# Configuration de l'entraînement
model.compile(
    loss='categorical_crossentropy',  # Pour classification multi-classes
    optimizer='adam',                 # Optimiseur adaptatif
    metrics=['accuracy']              # Métrique de suivi
)

# Entraînement
history = model.fit(
    X_train, y_train,
    batch_size=32,           # 32 images par mini-batch
    epochs=50,              # 50 passages complets sur les données
    validation_split=0.2,   # 20% des données pour validation
    verbose=1               # Affichage du progrès
)
```

1.9.2. Paramètres Cruciaux

Nombre de filtres : Plus de filtres = plus de motifs détectables, mais plus de calculs.

Taille des filtres :

- 3×3 : standard moderne, bon compromis
- $5 \times 5, 7 \times 7$: champ récepteur plus large
- 1×1 : réduction de dimensionnalité

Taux d'apprentissage :

- Trop élevé : oscillations, pas de convergence
- Trop faible : apprentissage très lent

1.10. Applications et Performances

1.10.1. Domaines d'Application

Vision par ordinateur :
- Classification d'images (ImageNet : 1000 classes, millions d'images)
- Détection d'objets (YOLO, R-CNN)
- Segmentation sémantique (U-Net)

Médical :
- Diagnostic par imagerie (radiologie, dermatologie)
- Analyse histologique
- Détection de tumeurs

Autonomie :
- Véhicules autonomes (détection piétons, panneaux)
- Drones de surveillance
- Robotique industrielle

1.10.2. Architectures Célèbres

AlexNet (2012) : Premier CNN à gagner ImageNet, révolution du deep learning.

VGG (2014) : Architecture simple mais très profonde (16-19 couches).

ResNet (2015) : Connexions résiduelles, réseaux de 152 couches.

EfficientNet (2019) : Optimisation du rapport précision/efficacité.

1.10.3. Limites et Défis Actuels

Données : Besoin de très grandes bases de données étiquetées.

Interprétabilité : Difficulté à comprendre les décisions du réseau.

Robustesse : Vulnérabilité aux attaques adversariales.

Éthique : Biais dans les données, vie privée, emploi.

1.11. Conclusion

Les CNN représentent une révolution dans le traitement d'images, combinant inspiration biologique et puissance computationnelle. Leur capacité à apprendre automatiquement des représentations hiérarchiques a ouvert la voie aux succès actuels de l'intelligence artificielle. Cependant, ils soulèvent aussi des questions importantes sur l'interprétabilité et l'éthique que la recherche continue d'explorer.