TP – Classification d'images avec CNN et Keras

Jeu de données CIFAR-10

Formation Data & Intelligence Artificielle

Objectifs pédagogiques

À l'issue de ce TP, l'apprenant sera capable de :

- Comprendre le fonctionnement d'un réseau de neurones convolutionnel (CNN)
- Charger et explorer le jeu de données CIFAR-10
- Prétraiter des images pour l'apprentissage profond
- Construire un modèle CNN avec Keras / TensorFlow
- Entraîner et évaluer un modèle de classification d'images
- Analyser les performances d'un modèle

Contexte

La classification d'images est l'un des principaux cas d'usage du Deep Learning. Dans ce TP, nous utilisons le jeu de données **CIFAR-10**, composé de 60 000 images couleur de taille 32×32 réparties en 10 classes.

Description du jeu de données CIFAR-10

- Nombre total d'images : 60 000
- Images d'entraînement : 50 000
- Images de test : 10 000
- Dimensions : $32 \times 32 \times 3$ (RGB)
- Problème : classification multiclasse

Classe	Description
0	Avion
1	Automobile
2	Oiseau
3	Chat
4	Cerf
5	Chien
6	Grenouille
7	Cheval
8	Bateau
9	Camion

Travail demandé

Le TP est structuré en plusieurs parties. Chaque partie doit être exécutée, commentée et analysée.

1 Importation des bibliothèques

1. Importer les bibliothèques nécessaires :
   - `numpy`
   - `pandas`
   - `matplotlib`
   - `seaborn`
   - `tensorflow.keras`
2. Configurer l'environnement graphique.

Question : Expliquer brièvement le rôle de chaque bibliothèque.

2 Chargement des données

1. Charger le jeu de données CIFAR-10 depuis Keras
2. Séparer les données en ensemble d'entraînement et de test
3. Vérifier les dimensions des tableaux

Question : Décrire la structure des données chargées.

3 Exploration visuelle des données

1. Afficher plusieurs images du jeu d'entraînement
2. Associer chaque image à son label
3. Observer la diversité des classes

Question : Quelles difficultés la classification de ces images peut-elle poser ?

4 Prétraitement des données

1. Normaliser les pixels des images
2. Vérifier le type des données
3. Vérifier la compatibilité avec la fonction de perte

Question : Pourquoi la normalisation est-elle indispensable en Deep Learning ?

5 Construction du modèle CNN

Construire un réseau CNN respectant l'architecture suivante :

- Deux blocs convolutionnels :
  - Convolution 2D
  - Convolution 2D
  - MaxPooling
  - Dropout
- Un classifieur :
  - Flatten
  - Dense
  - Dropout
  - Dense avec Softmax

Question : Expliquer le rôle de chaque type de couche.

6 Compilation du modèle

1. Compiler le modèle avec :
   - Optimiseur : Adam
   - Fonction de perte : sparse_categorical_crossentropy
   - Métrique : accuracy

Question : Justifier le choix de la fonction de perte.

7 Entraînement du modèle

1. Entraîner le modèle sur le jeu d'entraînement
2. Utiliser une validation interne
3. Choisir le nombre d'épochs et la taille de batch

8 Évaluation du modèle

1. Évaluer le modèle sur le jeu de test
2. Afficher l'accuracy finale
3. Comparer entraînement et test

Question : Le modèle généralise-t-il correctement ? Justifier.

9 Analyse des performances

1. Tracer les courbes de loss et d'accuracy
2. Identifier un éventuel surapprentissage

10 Questions de réflexion

1. Pourquoi les CNN sont-ils adaptés aux images ?
2. Quel est le rôle du MaxPooling ?
3. Pourquoi utiliser le Dropout ?
4. Comment améliorer les performances du modèle ?

Bonus (optionnel)

- Ajouter de la data augmentation
- Ajouter de la Batch Normalization
- Tester un autre optimiseur

Livrables attendus

- Notebook complété
- Codes exécutables
- Commentaires et analyses
- Réponses aux questions