# TP7 : Classification d'Images avec CNN et MNIST

Ce TP vise à vous initier aux concepts fondamentaux des réseaux de neurones convolutifs (CNN) en vous permettant de construire, entraîner et évaluer un modèle de classification d'images avec le dataset MNIST.



**Objectifs**

* Comprendre les concepts de base des réseaux de neurones convolutifs.
* Apprendre à utiliser Keras pour construire et entraîner un CNN.
* Évaluer la performance du modèle sur le dataset MNIST.

**Matériel**

* Un ordinateur avec Python 3.x installé.
* Les bibliothèques Python installées : Pandas, numpy, scikit-learn, keras (tensorflow), matplotlib.

**Prérequis**

* Connaissances de base en Python.
* Notions fondamentales en deep learning.

**Données**

* Le jeu de données MNIST contient +70000 images en chiffres manuscrits en noir sur blanc.
* Taille des images 28x28 pixels.
* Chaque pixel a une valeur entre 0 (blanc) et 255 (noir).
* Vous pouvez l’importer directement depuis keras :

from tensorflow.keras.datasets import mnist

**Parties**

1. **Chargement des données (Importation) :**
   * Importer les librairies nécessaires (pandas, numpy, sklearn, keras, pyplot, Conv2D, MaxPooling2D, Flatten, Dense).
   * Charger le dataset MNIST.
     1. Penser à divisier directement en entrainement et test.
   * Afficher quelques exemples d'images + étiquettes pour comprendre les données.
     1. Pensez à utiliser le subplot
2. **Préparation des données :**
   * Normaliser les images d’entrainement et testpour que le pixel soit entre 0 et 255
     1. Pensez à diviser par 255.
   * Reshape des données pour les adapter à l'entrée du CNN
     1. Transformées de 28x28 en 28x28x1 pour indiquer qu'elles ont une seule couche de couleur.
3. **Construction du modèle (CNN) :**
   * Créer un modèle séquentiel.
   * Ajouter une première couche de convolution (Conv2D) avec 32 filtres de taille 3x3 et activation ReLU
     1. Pensez à mettre le input\_shape=(28, 28, 1))
   * Ajouter une première couche de pooling (MaxPooling2D) pour réduire la taille de l'image de moitié.
   * Ajouter une deuxième couche de convolution (Conv2D) avec 64 filtres de taille 3x3 et activation ReLU
   * Ajouter une deuxième couche de pooling (MaxPooling2D) pour réduire encore la taille de l'image de moitié
   * Ajouter une couche de flattening (Flatten) pour convertir les matrices 2D en vecteurs 1D
   * Ajouter une couche dense entièrement connectée avec 64 neurones et activation ReLU
   * Ajouter une couche de sortie avec 10 neurones (pour les 10 classes de chiffres) et activation softmax
   * Compiler le modèle avec l’optimisateur ‘adam’, la fonction de perte ‘sparse\_categorical\_crossentropy’ et la métrique d’évaluation ‘accuracy’.
4. **Entraînement du modèle:**
   * Entraîner le modèle sur les données d'entraînement.
     1. Utiliser 10 époques
     2. Utiliser le param validation\_data pour faire la validation des données x\_test,y\_test en même temps.
5. **Évaluation du modèle sur les données de test :**
   * Évaluer le modèle sur les données de test.
   * Afficher la précision du modèle ‘accuracy’.
6. **Visualisation:**
   * Tracer les courbes de précision d'entraînement et de validation pour chaque époque.
     1. Pour comparer accuracy et val\_accuracy

**Travail à rendre :**

Appliquer le modèle CNN sur la dataset CIFAR-10.

\* Elle contient 60 000 images en couleur de 32x32 pixels

\* Réparties en 10 classes (avions, voitures, oiseaux, chats, etc.).

\* A importer en utilisant :

from tensorflow.keras.datasets import cifar10