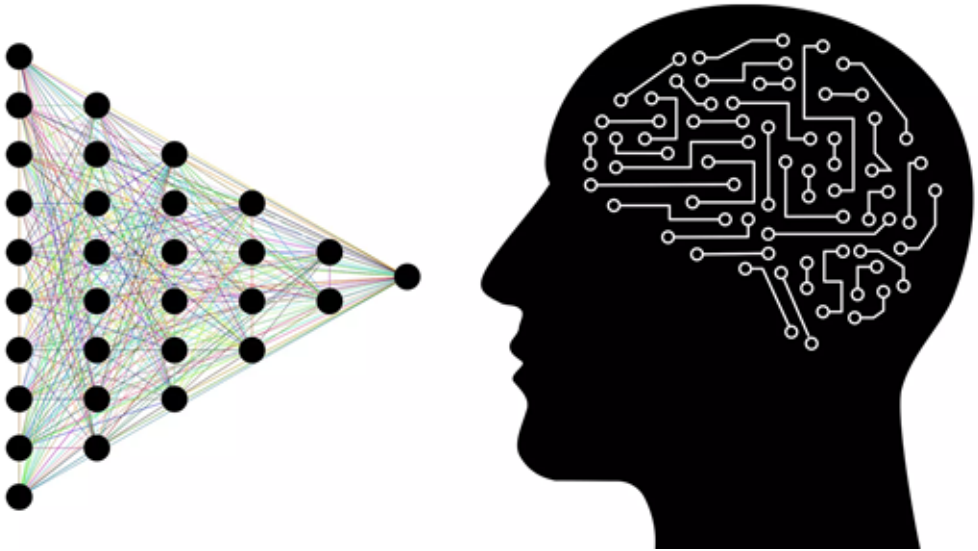
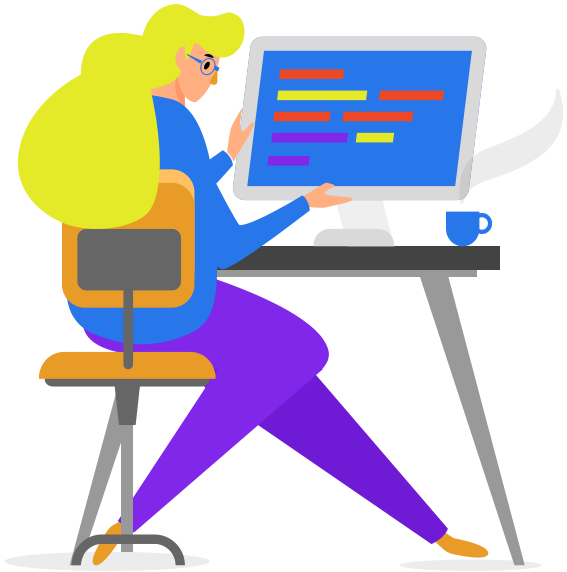
Les réseaux de neurones Convolutifs(CNN)



Mouhammed Niah

Plan



01 Introduction

02 Fonctionnement de base des CNN

03 Architecture des CNN

04 Fonctions clés des couches

05 Fonctions d'activation et techniques de régularisation

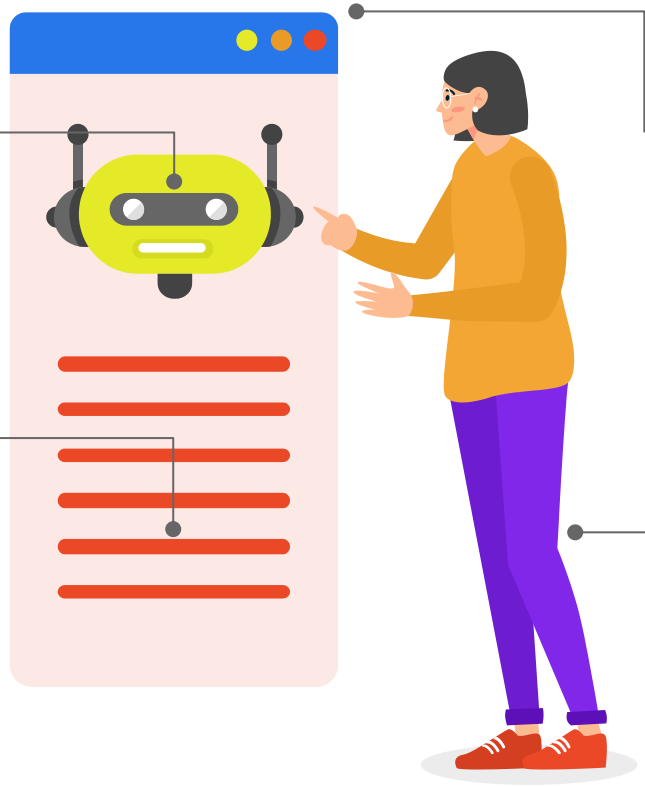
06 Implementation

Introduction

Qu'est-ce qu'un réseau de neurones convolutifs ?

01

Les réseaux de neurones convolutifs (CNN) représentent une catégorie essentielle du deep learning.



02

Ils partagent des similitudes avec les réseaux de neurones classiques, utilisant des paramètres d'apprentissage tels que les poids, les biais, etc.

03

Les CNN jouent un rôle majeur dans des domaines tels que la reconnaissance d'images, la détection d'objets, et la reconnaissance faciale.

04

Les composants fondamentaux des CNN sont :

- La couche de convolution
- La couche de mise en commun
- La couche de sortie

Composants de base de couches du CNN

Input Layer

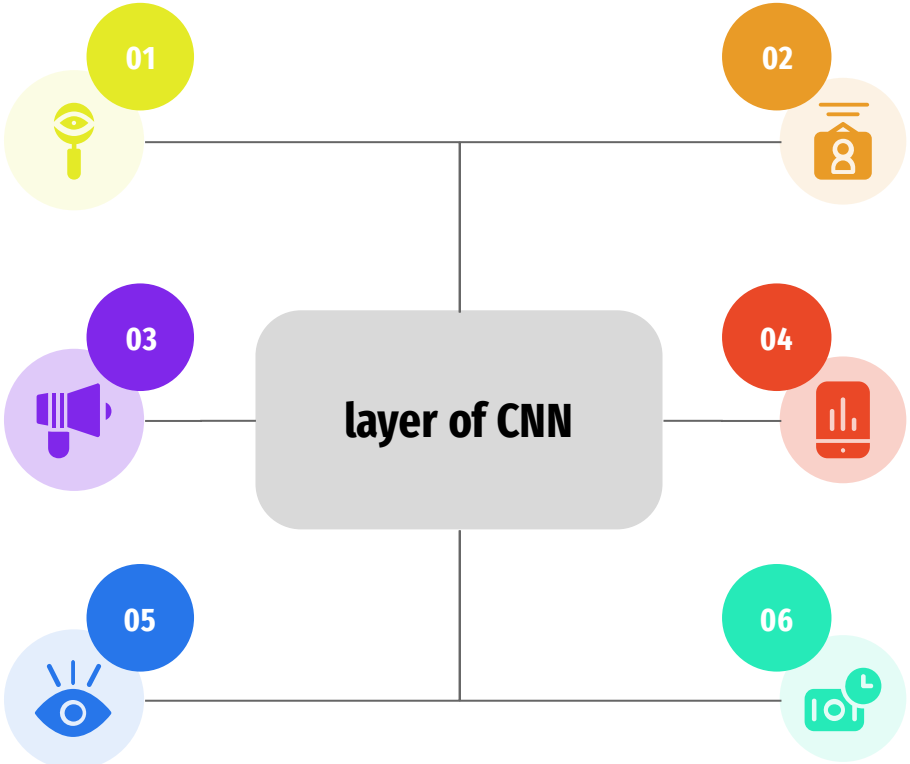
Représente les données d'entrée

Convolutional Layer

Applique des filtres pour détecter des motifs locaux et extraire des caractéristiques importantes de l'image

Pooling Layer

Réduit la dimension spatiale des caractéristiques extraites, en conservant les informations importantes



Fully Connected Layer

Aplatit les données

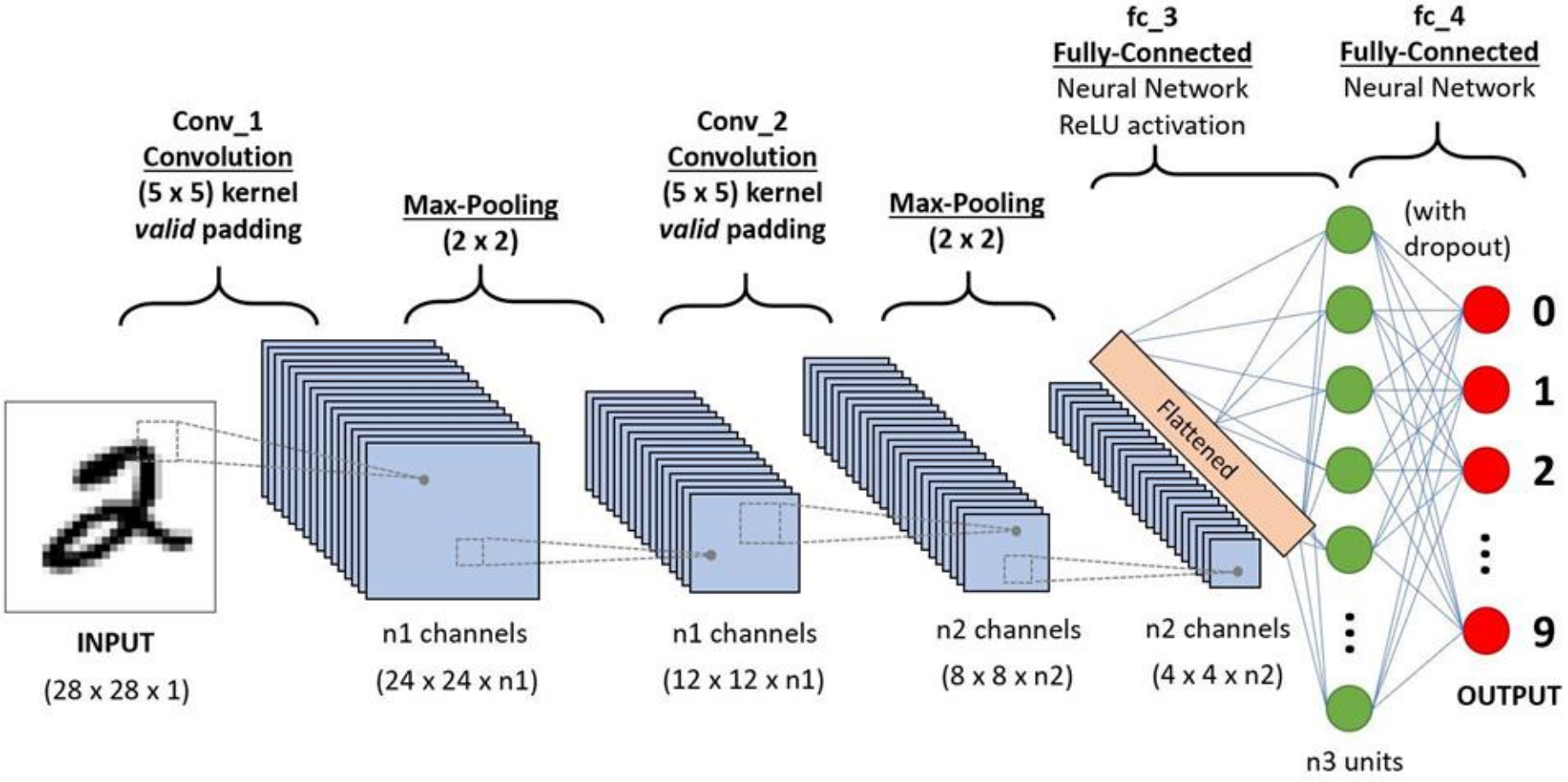
Output Layer

Produit les résultats finaux

Batch Normalization Layer

Normalise les activations de la couche précédente

Architecture des CNN


fc_3
Fully-Connected
Neural Network
ReLU activation
fc_4
Fully-Connected
Neural Network
Conv_1
Convolution
(5 x 5) kernel
valid padding
Max-Pooling
(2 x 2)
Conv_2
Convolution
(5 x 5) kernel
valid padding
Max-Pooling
(2 x 2)
(with
dropout)
Flattened
0
1
2
9
INPUT
(28 x 28 x 1)
n1 channels
(24 x 24 x n1)
n1 channels
(12 x 12 x n1)
n2 channels
(8 x 8 x n2)
n2 channels
(4 x 4 x n2)
n3 units
OUTPUT

Fonctions clés des couches

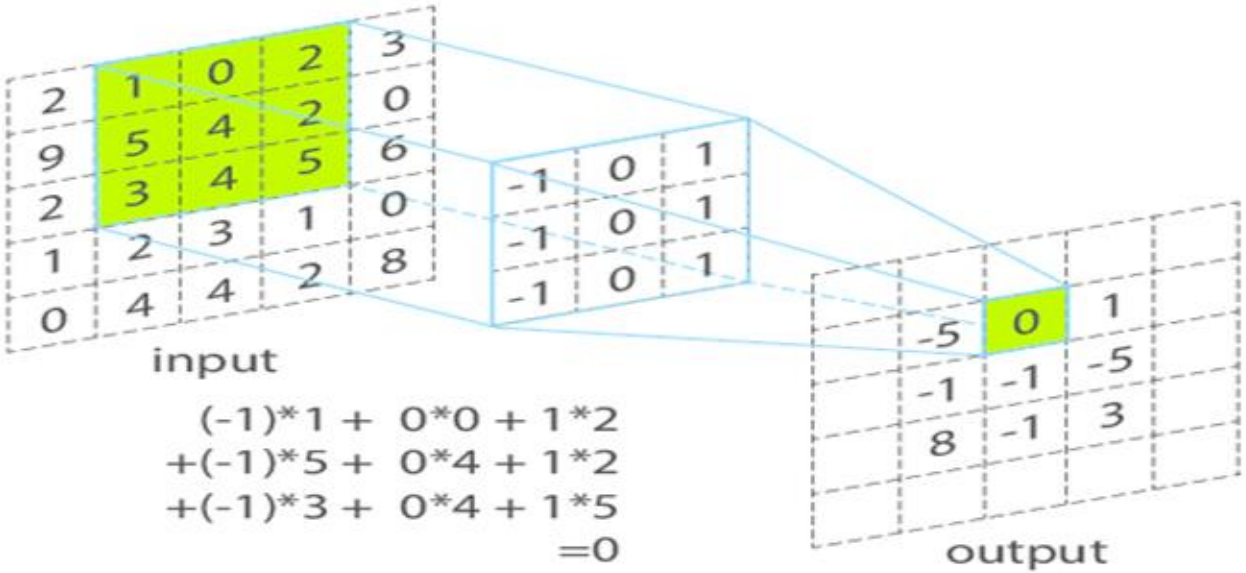
Couche d'Entrée (Input Layer)

la couche d'entrée est la première étape où les données d'entrée sont présentées au réseau, marquant le début du processus d'apprentissage et de traitement des informations par les couches subséquentes

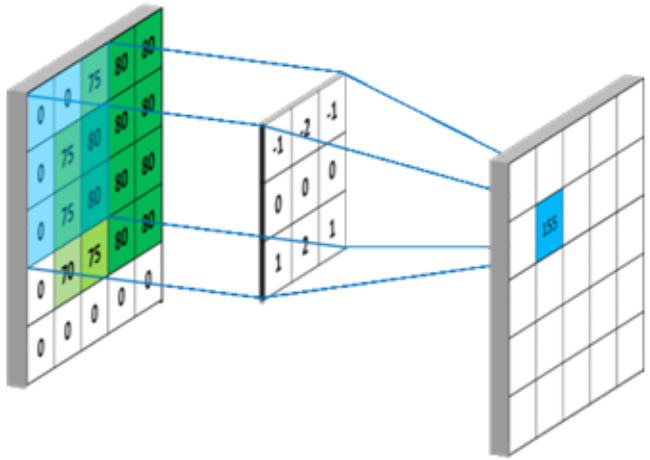
Couche de Convolution

D'un point de vue simpliste, la **Couche de Convolution** dans un CNN revient à appliquer un filtre mathématique à chaque pixel d'une image. Ce filtre parcourt l'image pour détecter des motifs locaux, tels que des bords ou des textures, permettant au réseau de capturer des caractéristiques visuelles significatives

Couche de Convolution

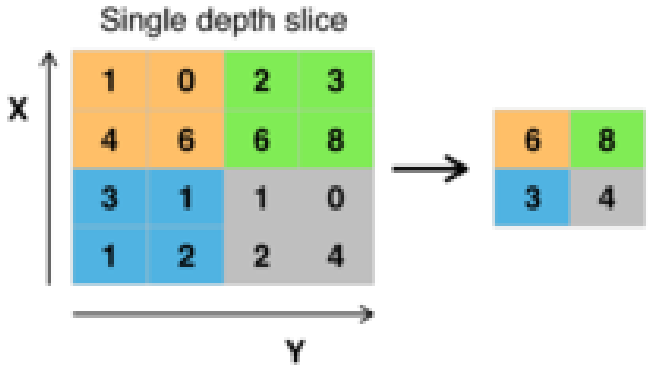


Couche de Convolution



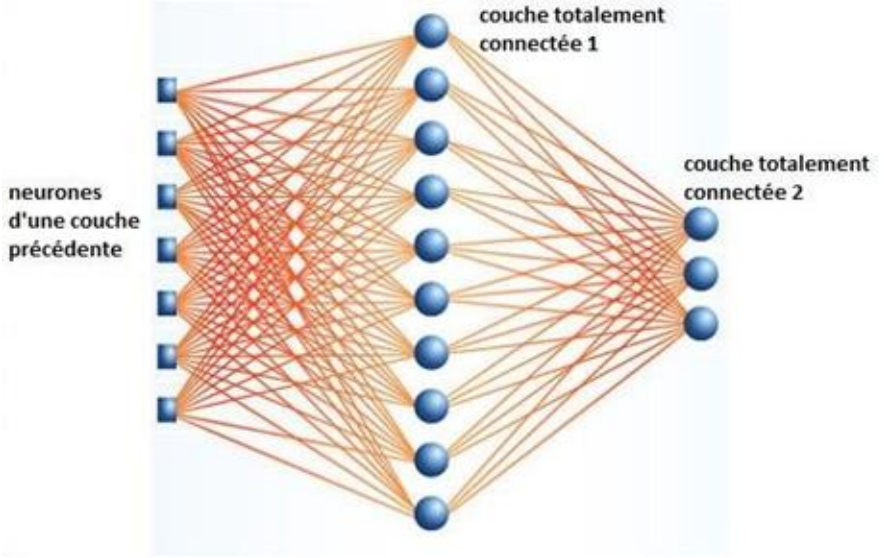
Couche de Mise en Commun (Pooling Layer)

dans un réseau de neurones convolutif (CNN) simplifie et réduit les informations extraites par les couches de convolution. Elle le fait en regroupant les valeurs voisines, souvent par une opération comme la moyenne ou la prise du maximum.

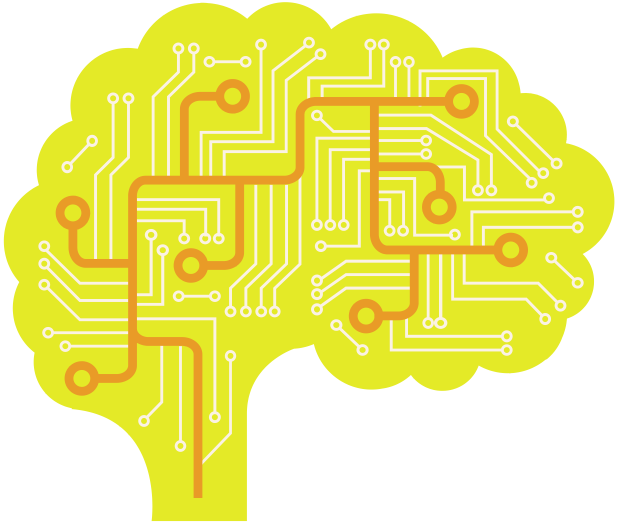


Couche Entièrement Connectée
(Fully Connected Layer)

a couche entièrement connectée joue un rôle clé dans la capacité du modèle à combiner et interpréter les caractéristiques extraites, permettant ainsi de générer des prédictions significatives pour la tâche à accomplir.

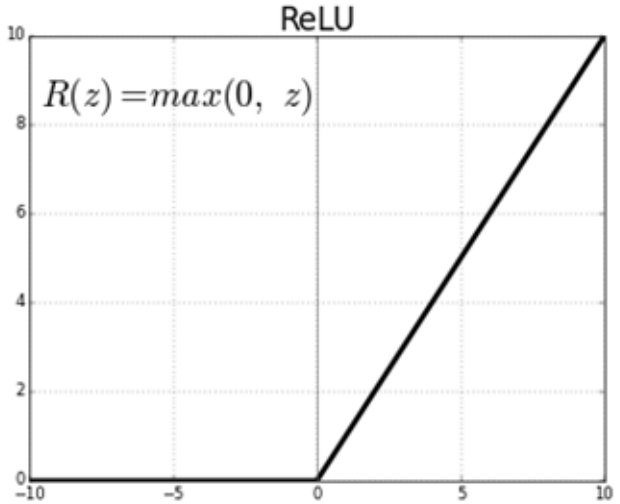


Fonctions d'activation et techniques de régularisation



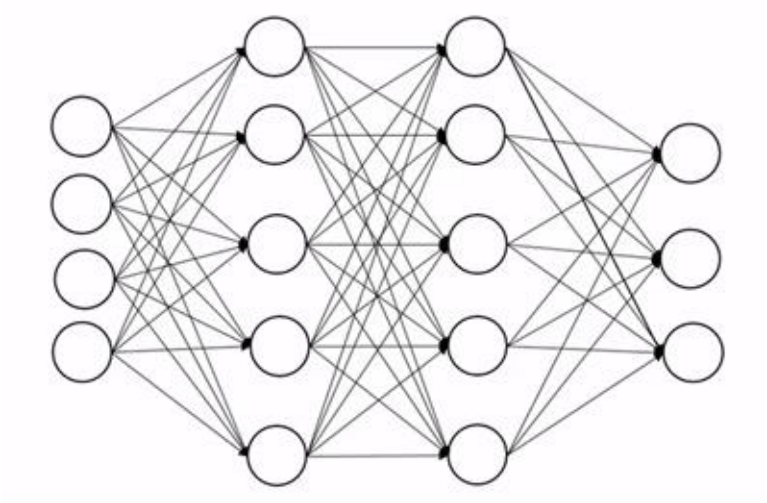
Fonction d'Activation ReLU

La fonction d'activation ReLU (Rectified Linear Unit) est couramment utilisée dans les réseaux de neurones, y compris les CNN, pour introduire de la non-linéarité dans le modèle. Elle est définie comme suit : f(x)=max(0,x)f(x)=max(0,x)



Techniques de Régularisation - Dropout

Le dropout est une technique de régularisation fréquemment utilisée pour prévenir le surajustement dans les réseaux de neurones, y compris les CNN.



Fonctions d'activation et techniques de régularisation

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

# ...

# Utiliser le module Sequential de Keras pour construire le modèle
model = Sequential()

# Ajouter des couches de convolution et de mise en commun
model.add(Conv2D(32, (3, 3), padding="same", activation="relu", input_shape=(32, 32, 3)))
model.add(Conv2D(32, (3, 3), activation="relu"))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Dropout(0.25))

model.add(Conv2D(64, (3, 3), padding="same", activation="relu"))
model.add(Conv2D(64, (3, 3), activation="relu"))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Dropout(0.25))

# Aplatir les données avant de passer aux couches entièrement connectées
model.add(Flatten())

# Ajouter des couches entièrement connectées
model.add(Dense(500, activation="relu"))
model.add(Dense(250, activation="relu"))
model.add(Dense(150, activation="relu"))
model.add(Dropout(0.5))

# Couche de sortie avec activation softmax pour la classification
model.add(Dense(3, activation="softmax"))

# Compiler le modèle avec un optimiseur, une fonction de perte et une métrique
model.compile(optimizer=Adam(lr=0.001), loss="sparse_categorical_crossentropy", metrics=["accuracy"])

# Afficher un résumé du modèle
model.summary()
```