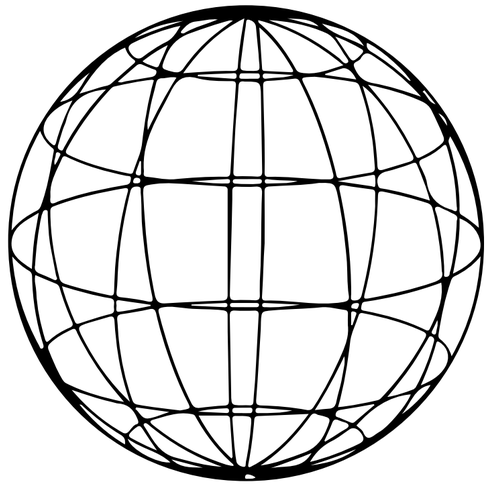
UNIVERSITÉ PARIS 8 VINCENNES-SAINT-DENIS

Master 1 Informatique et Big Data

IMPLÉMENTATION DE RÉSEAUX DE NEURONES CONVOLUTIFS (CNN) EN PYTHON

PROJET DE RECHERCHE EN BIG DATA



Réalisé par **ANTON NELCON Steve**

SOMMAIRE

Réalisé par **ANTON NELCON Steve**

01

INTRODUCTION

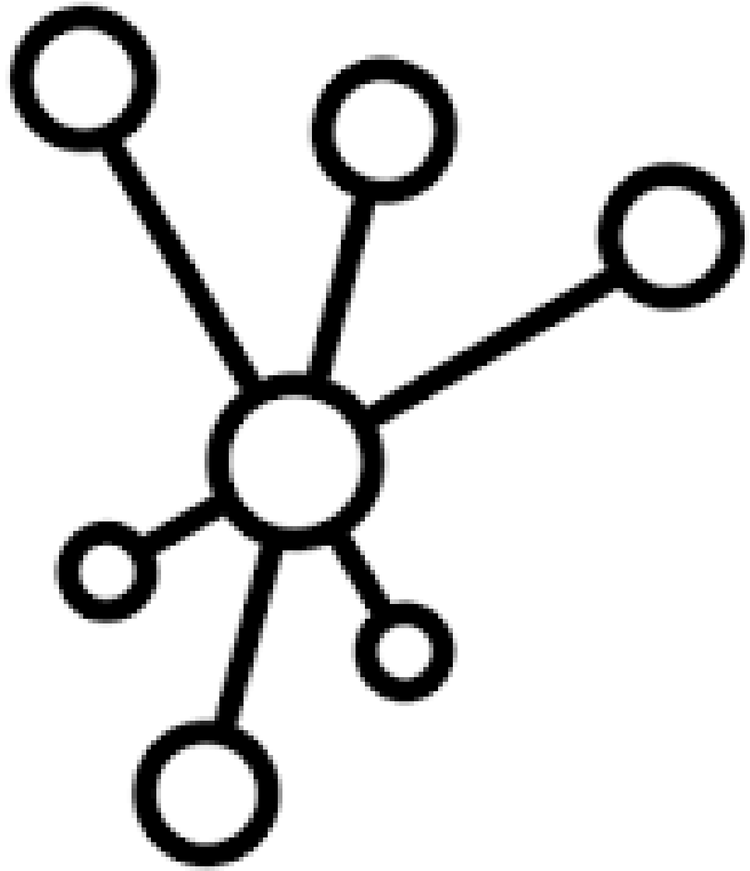
→ Définition d’un (CNN)

→ Contexte du Projet

→ Technologie et outils utilisés

UNIVERSITÉ PARIS 8 VINCENNES-SAINT-DENIS

QU'EST CE QU'UN **RÉSEAUX DE NEURONES CONVOLUTIFS (CNN)** ?



Un Réseau de Neurones Convolutif (CNN) est un modèle de deep learning utilisé pour l'analyse d'images.

Il permet de détecter automatiquement des éléments visuels comme les contours et les formes.

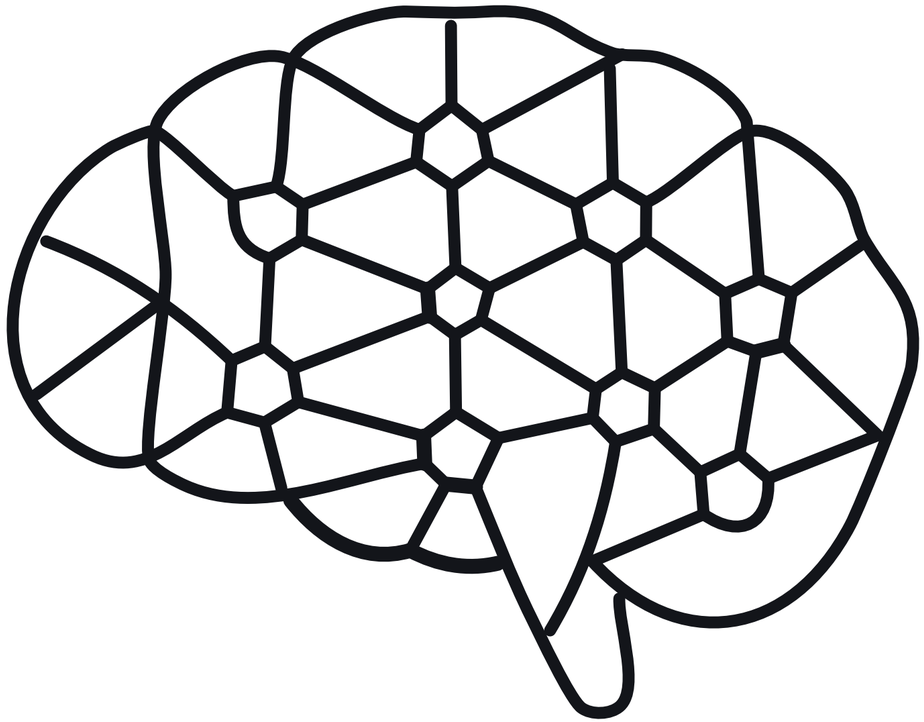
Les CNN sont utilisés pour la classification d'images et la reconnaissance d'objets, de visages ou de chiffres.

Réalisé par **ANTON NELCON Steve**

L’objectif principal d’un Réseau de Neurones Convolutif (CNN) est de réaliser des prédictions précises à partir d’images.

Le projet est organisé en sept sections principales, correspondant aux sept projets développés :

→ Classification d’images du dataset CIFAR-10
→ Reconnaissance de chiffres manuscrits (MNIST)
→ Reconnaissance de formes géométriques
→ Classification de vêtements
→ Reconnaissance de couleurs
→ Classification de fruits et légumes
→ Reconnaissance des émotions sur un visage



Réalisé par **ANTON NELCON Steve**

1. INTRODUCTION

→ Technologie et Outils utilisés

UNIVERSITÉ PARIS 8 VINCENNES-SAINT-DENIS

ENVIRONNEMENT DE DÉVELOPPEMENT

kaggle

LANGAGE DE PROGRAMMATION

FRAMEWORKS

BIBLIOTHEQUES

matplotlib

Réalisé par **ANTON NELCON Steve**

02

LES PROJETS

→ Classification d’images du dataset CIFAR-10

→ Reconnaissance de chiffres manuscrits (MNIST)

→ Reconnaissance de formes géométriques

→ Classification de vêtements

→ Reconnaissance de couleurs

→ Classification de fruits et légumes

→ Reconnaissance des émotions sur un visage

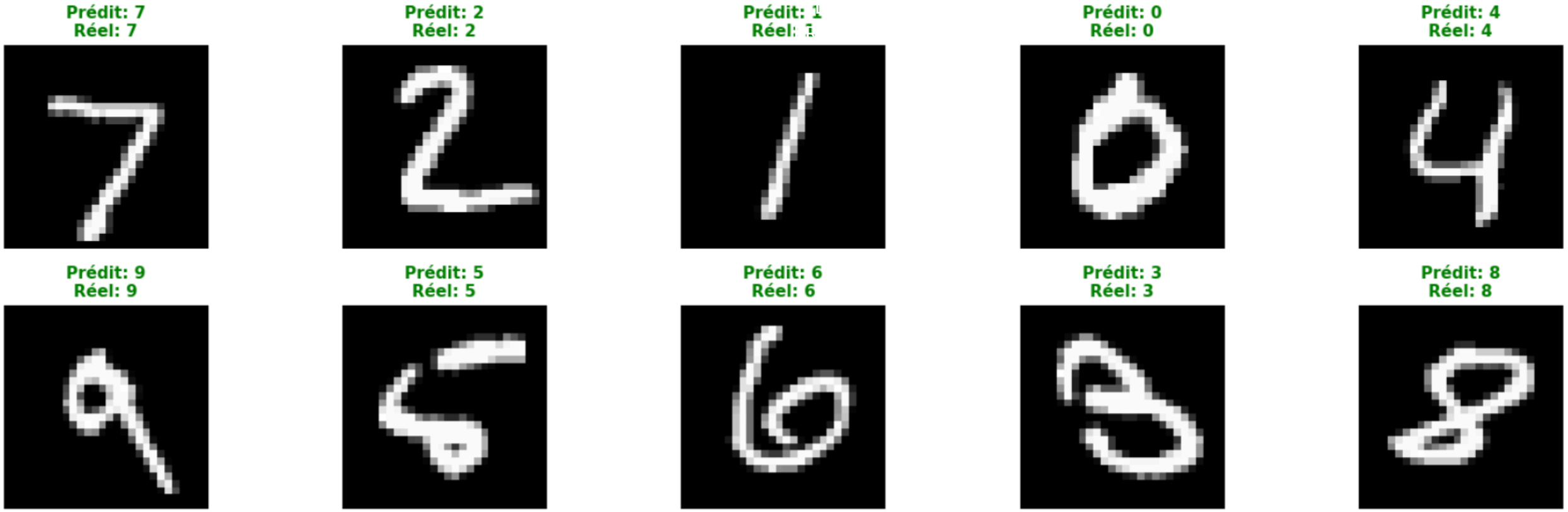
LES PROJETS

→ Classification d'images du dataset CIFAR-10

Détection d'objet dans le dataset CIFAR-10

UNIVERSITÉ PARIS 8 VINCENNES-SAINT-DENIS

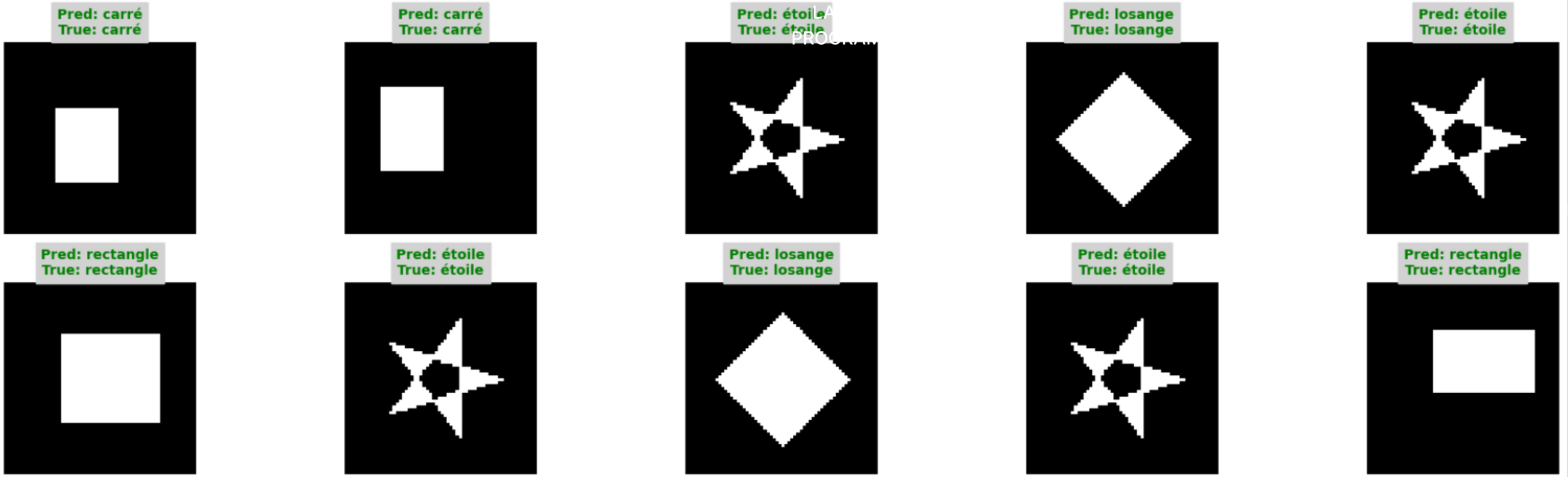
Reconnaissance de chiffres manuscrits (MNIST) - CNN



Fait par ANTON NELCON Steve - M1 IBD P8

UNIVERSITÉ PARIS 8 VINCENNES-SAINT-DENIS

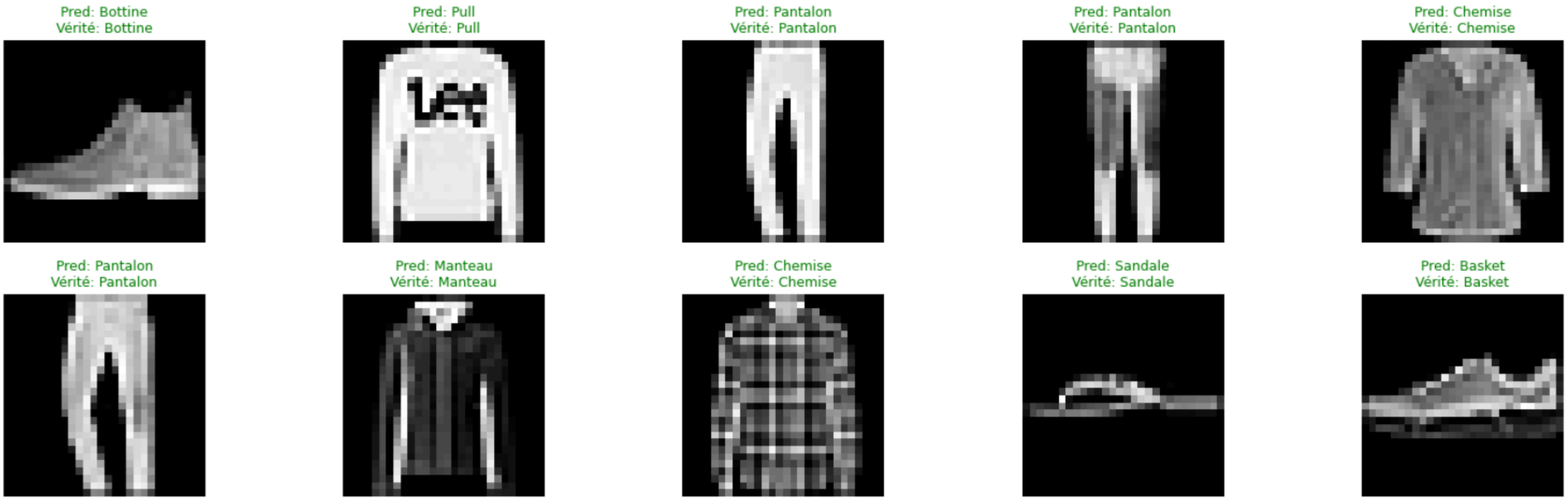
Reconnaissance de formes géométriques



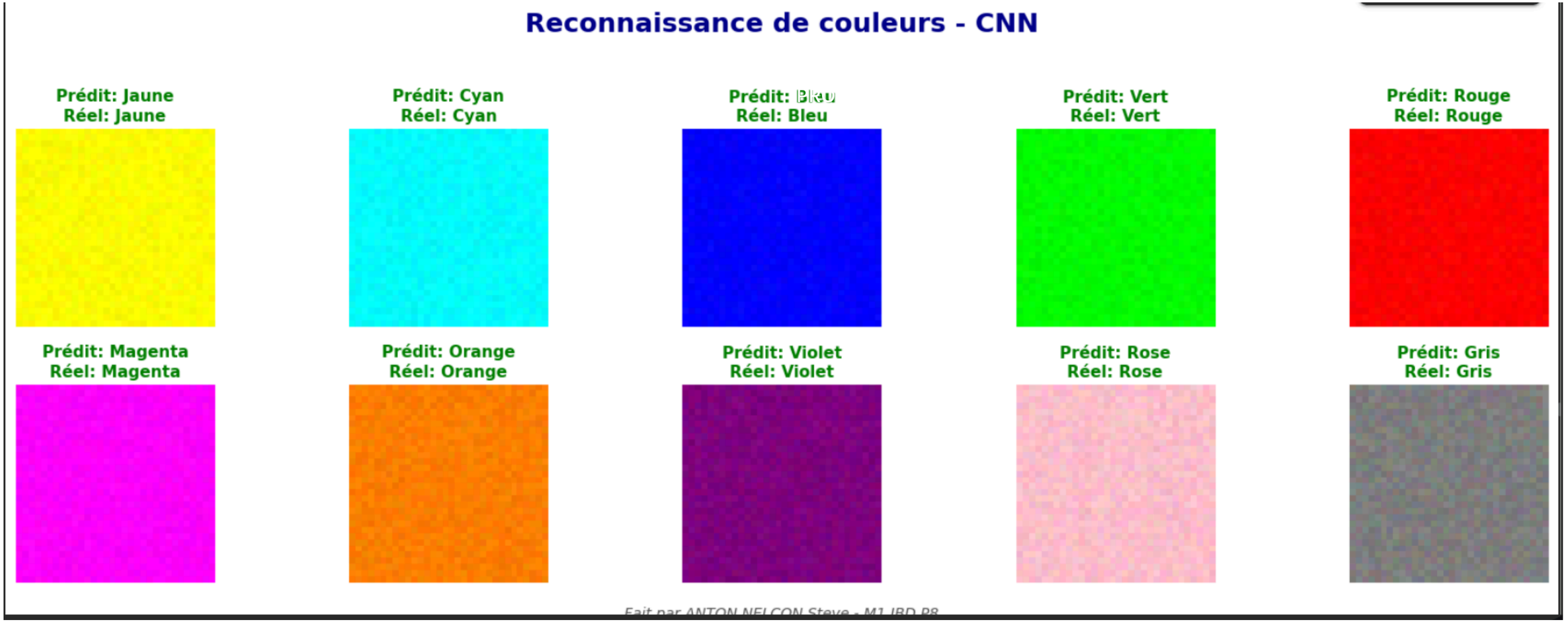
Fait par ANTON NELCON Steve - M1 IBD P8

UNIVERSITÉ PARIS 8 VINCENNES-SAINT-DENIS

🧠 Prédictions du modèle CNN sur le dataset Zalando Fashion Images



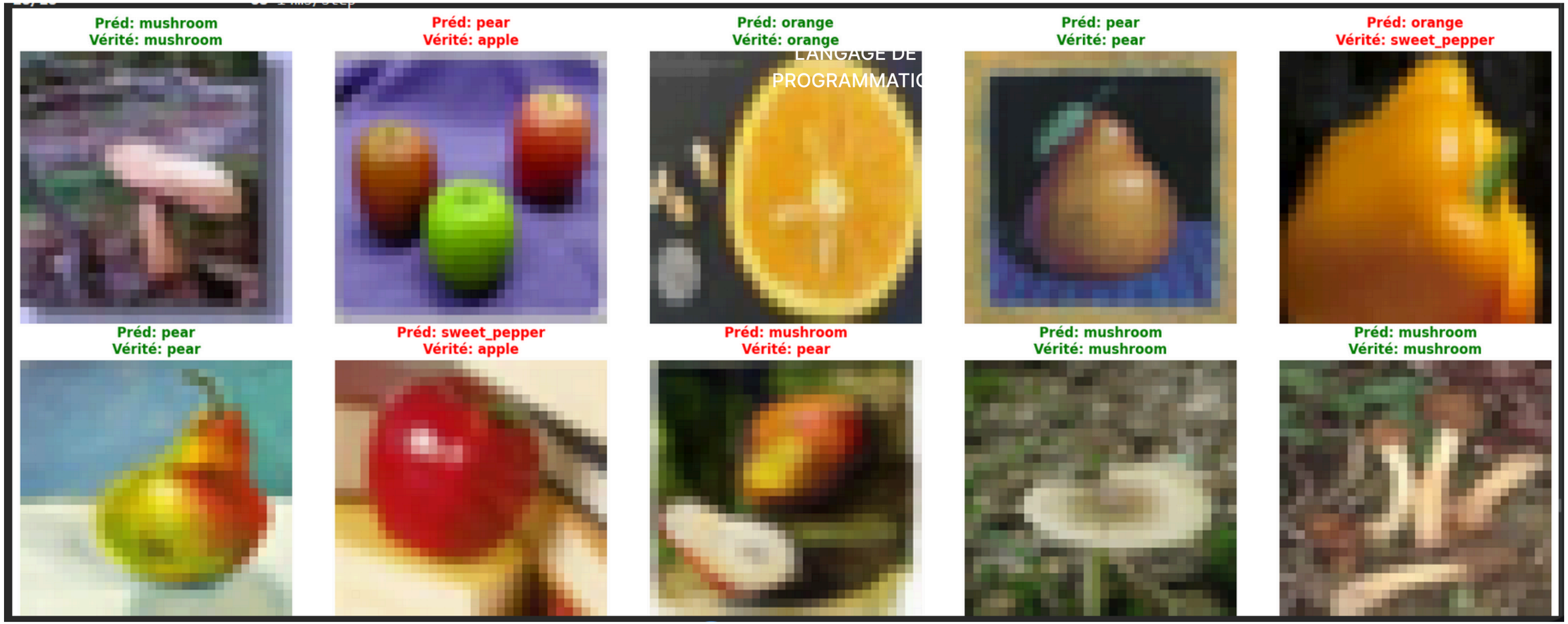
Fait par Anton Nelcon Steve - M1 IBD P8


Reconnaissance de couleurs - CNN
Prédit: Jaune
Réel: Jaune
Prédit: Cyan
Réel: Cyan
Prédit: Bleu
Réel: Bleu
Prédit: Vert
Réel: Vert
Prédit: Rouge
Réel: Rouge
Prédit: Magenta
Réel: Magenta
Prédit: Orange
Réel: Orange
Prédit: Violet
Réel: Violet
Prédit: Rose
Réel: Rose
Prédit: Gris
Réel: Gris
Fait par ANTON NELCON Steve - M1 IBD P8

2.

LES PROJETS

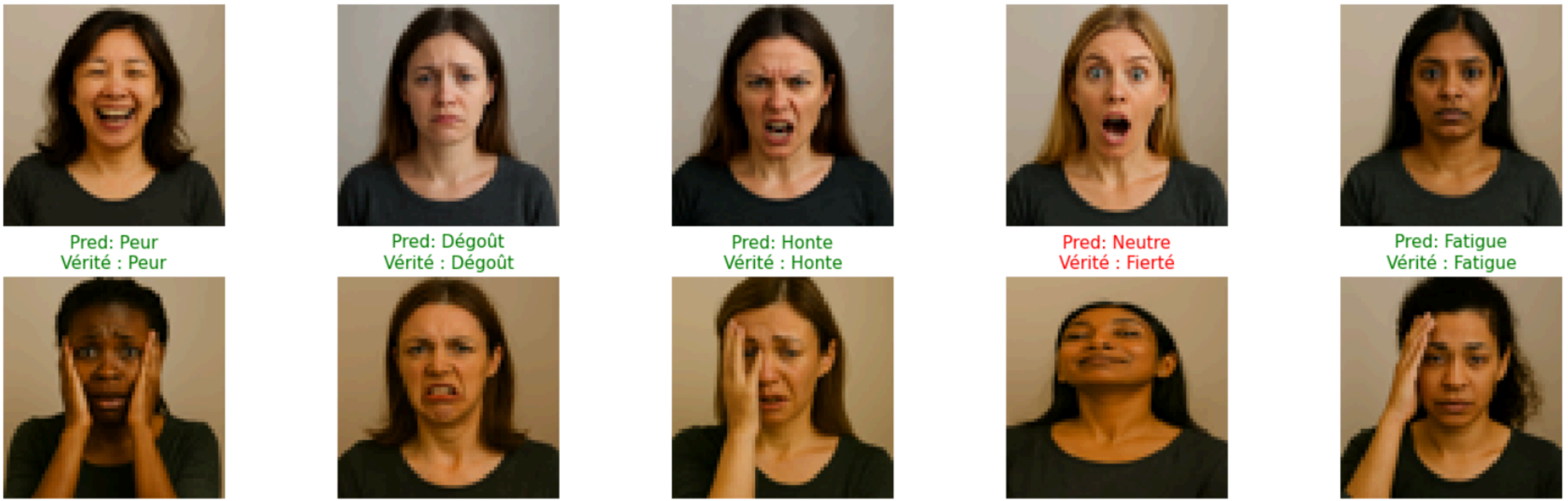
→ Reconnaissance de fruits et légumes



Réalisé par **ANTON NELCON Steve**

UNIVERSITÉ PARIS 8 VINCENNES-SAINT-DENIS

Prédictions CNN sur des visages d'émotions humaines



Réalisé par **ANTON NELCON Steve**

03

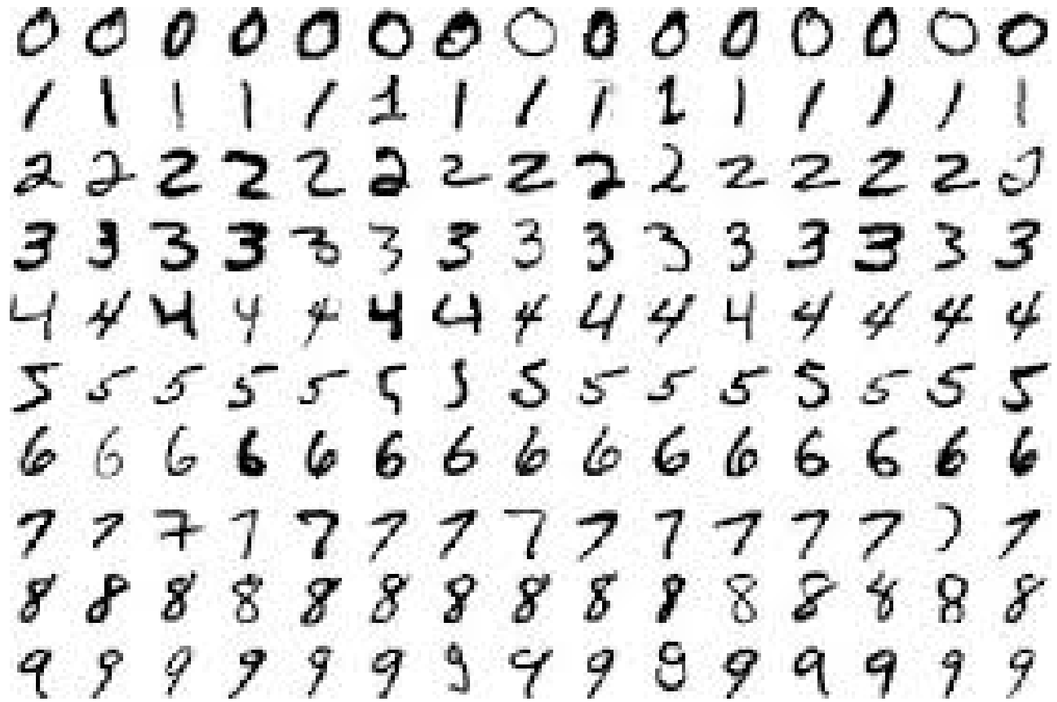
ANALYSE DES PROJETS

→ Reconnaissance de chiffres manuscrits (MNIST

→ Reconnaissance des émotions sur un visage

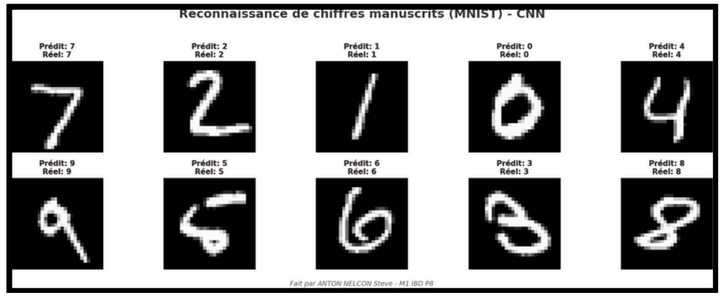
CHIFFRE MANUSCRIT

MNIST



3. ANALYSE DES PROJETS

→ Reconnaissance de chiffres manuscrits (MNIST)



Importer toute les bibliothèques

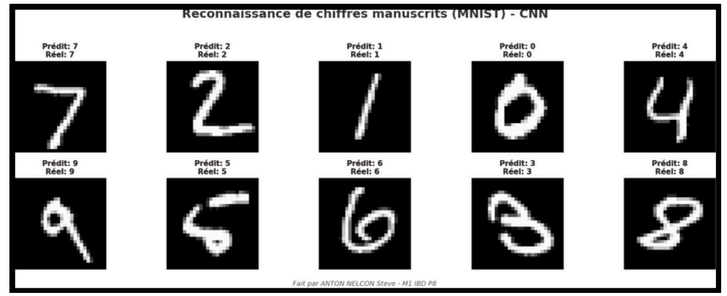
LANGAGE DE PROGRAMMATION

```python
import tensorflow as tf
from tensorflow.keras.datasets import mnist
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Conv2D, MaxPooling2D, Flatten, Dense, Dropout
import matplotlib.pyplot as plt
import numpy as np
```

Explication : J'importe toutes les bibliothèques dont j'ai besoin.
TensorFlow/Keras pour créer le modèle, Matplotlib pour visualiser les résultats, et NumPy pour manipuler les données.

3. ANALYSE DES PROJETS

→ Reconnaissance de chiffres manuscrits (MNIST)



PROGRAMMATION

```
# Charger le dataset MNIST
(x_train, y_train), (x_test, y_test) = mnist.load_data()
```

Explication : Je charge les images de chiffres manuscrits MNIST qui contient 70 000 images (60 000 pour l'entraînement, 10 000 pour le test).

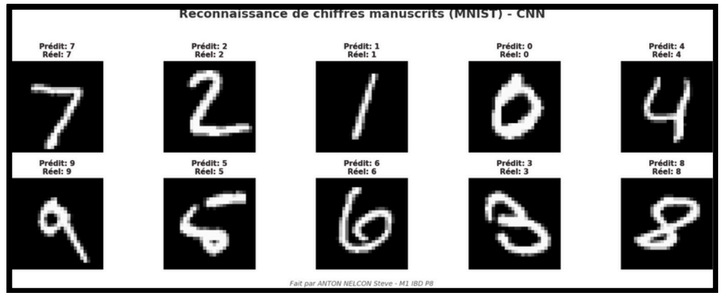
Explication : Je transforme chaque image 28×28 en un tableau 28×28×1 (car les images sont en niveaux de gris), et je divise par 255 pour normaliser les pixels entre 0 et 1.

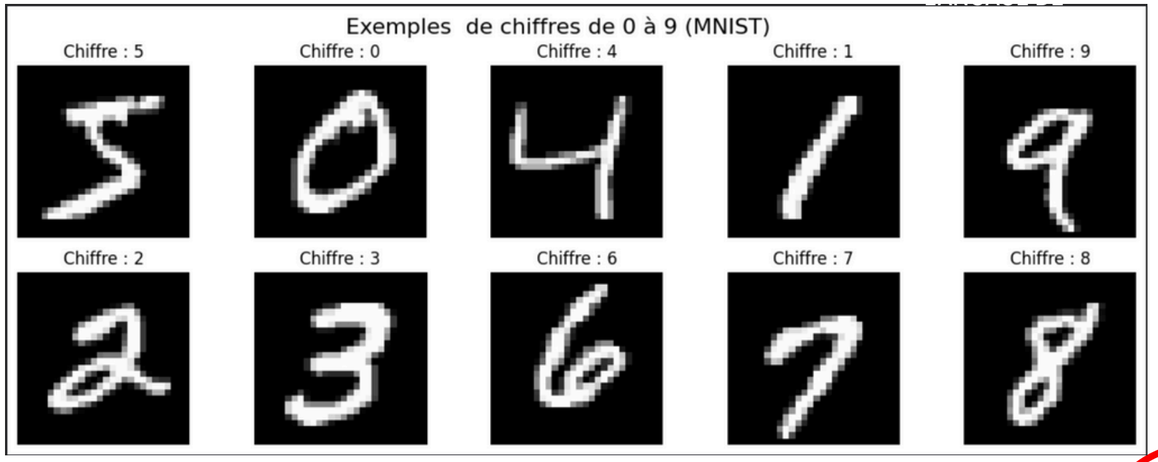
Prétraitement des données

```
# Redimensionner les images (ajouter la dimension du canal)
x_train = x_train.reshape((x_train.shape[0], 28, 28, 1))
x_test = x_test.reshape((x_test.shape[0], 28, 28, 1))

# Normaliser les pixels (0-255 -> 0-1)
x_train, x_test = x_train / 255.0, x_test / 255.0

# Noms des classes MNIST
classes = [str(i) for i in range(10)]
```

3. ANALYSE DES PROJETS

→ Reconnaissance de chiffres manuscrits (MNIST)





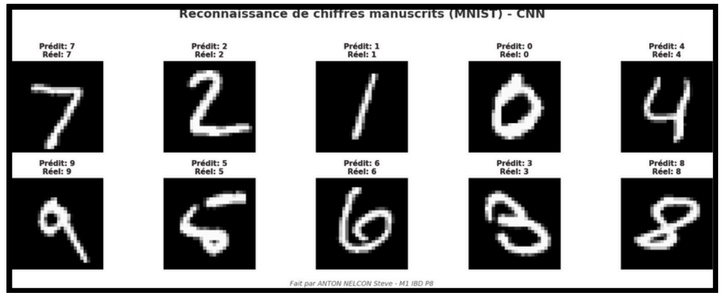
Explication : Je visualise 10 images d'entraînement avec leurs labels, pour m'assurer que les données sont bien chargées.

Afficher les 10 images du Dataset

```
# Trouver un index pour chaque chiffre unique
unique_digits = []
indices = []

for i, label in enumerate(y_train):
    if label not in unique_digits:
        unique_digits.append(label)
        indices.append(i)
    if len(unique_digits) == 10:
        break  # on s'arrête dès qu'on a trouvé les 10 chiffres uniques

# Afficher une image pour chaque chiffre
plt.figure(figsize=(15,5))

for i, idx in enumerate(indices):
    plt.subplot(2,5,i+1)
    plt.imshow(x_train[idx].reshape(28,28), cmap='gray')
    plt.title(f"Chiffre : {y_train[idx]}")
    plt.axis('off')

plt.suptitle("Exemples  de chiffres de 0 à 9 (MNIST)", fontsize=16)
plt.show()
```

3. ANALYSE DES PROJETS

→ Reconnaissance de chiffres manuscrits (MNIST)



EXPLICATION DÉTAILLÉE :

CONV2D(32, (3, 3)) :
- 32 filtres de convolution 3x3
- relu : Fonction d'activation (Rectified Linear Unit)
- Extrait des caractéristiques locales (bords, formes)

MAXPOOLING2D((2, 2)) :
- Réduit la dimension spatiale de moitié
- Garde les informations les plus importantes
- Réduit le nombre de paramètres

CONV2D(64, (3, 3)) :
- Augmente à 64 filtres pour capturer des motifs plus complexes

FLATTEN() :
- Transforme les matrices 2D en vecteur 1D
- Prépare pour les couches denses

DENSE(64) :
- Couche complètement connectée
- Apprend les combinaisons complexes des caractéristiques

DROPOUT(0.5) :
- Désactive aléatoirement 50% des neurones pendant l'entraînement
- Empêche l'overfitting (mémorisation des données d'entraînement)

DENSE(10, ACTIVATION='SOFTMAX') :
- 10 neurones (un par chiffre)
- softmax : Convertit les scores en probabilités (somme = 1)

Création du CNN

```
# Créer le modèle CNN
model = Sequential([
    Conv2D(32, (3,3), activation='relu', input_shape=(28,28,1)),
    MaxPooling2D((2,2)),
    Conv2D(64, (3,3), activation='relu'),
    MaxPooling2D((2,2)),
    Flatten(),
    Dense(64, activation='relu'),
    Dropout(0.5),
    Dense(10, activation='softmax')
])
```

Première couche de convolution

Réduction de dimension

Deuxième couche de convolution

Deuxième pooling

Aplatir les données pour les couches denses

Couche dense complètement connectée

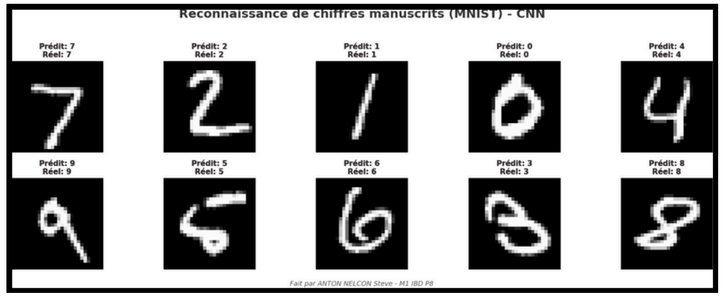
Dropout pour éviter l'overfitting

Couche de sortie (10 classes : chiffres 0-9)

Réalisé par **ANTON NELCON Steve**

3. ANALYSE DES PROJETS

→ Reconnaissance de chiffres manuscrits (MNIST)



Model: "sequential_4"

Layer (type)	Output Shape	Param #
conv2d_11 (Conv2D)	(None, 26, 26, 32)	320
max_pooling2d_8 (MaxPooling2D)	(None, 13, 13, 32)	0
conv2d_12 (Conv2D)	(None, 11, 11, 64)	18,496
max_pooling2d_9 (MaxPooling2D)	(None, 5, 5, 64)	0
flatten_4 (Flatten)	(None, 1600)	0
dense_8 (Dense)	(None, 64)	102,464
dropout_4 (Dropout)	(None, 64)	0
dense_9 (Dense)	(None, 10)	650

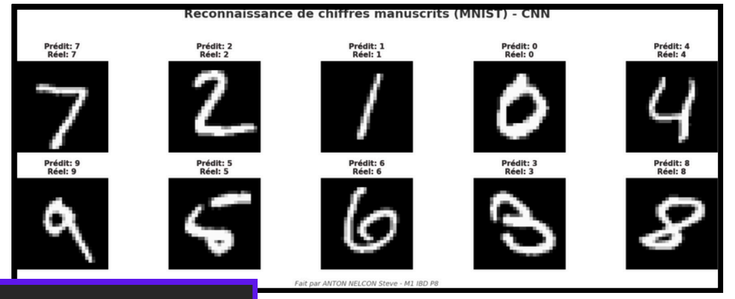
Total params: 121,930 (476.29 KB)
Trainable params: 121,930 (476.29 KB)
Non-trainable params: 0 (0.00 B)

```python
# Compiler le modèle
model.compile(optimizer='adam',
              loss='sparse_categorical_crossentropy',
              metrics=['accuracy'])

# Résumé du modèle
model.summary()
```

Explication : Je définis l'optimiseur, la fonction de coût et les métriques d'évaluation, puis j'affiche un résumé de l'architecture.

3. ANALYSE DES PROJETS

→ Reconnaissance de chiffres manuscrits (MNIST)



Entrainement du modèle

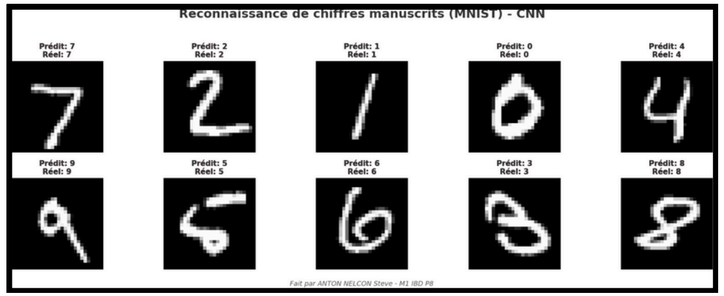
```
history = model.fit(x_train, y_train, epochs=10, batch_size=64, validation_split=0.2)
```

```
Epoch 1/10
750/750 ━━━━━━━━━━━━━━━━━━━━ 42s 54ms/step - accuracy: 0.7630 - loss: 0.7379 - val_accuracy: 0.9751 - val_loss: 0.0781
Epoch 2/10
750/750 ━━━━━━━━━━━━━━━━━━━━ 41s 54ms/step - accuracy: 0.9456 - loss: 0.1809 - val_accuracy: 0.9799 - val_loss: 0.0651
Epoch 3/10
750/750 ━━━━━━━━━━━━━━━━━━━━ 41s 55ms/step - accuracy: 0.9618 - loss: 0.1273 - val_accuracy: 0.9849 - val_loss: 0.0487
Epoch 4/10
750/750 ━━━━━━━━━━━━━━━━━━━━ 40s 54ms/step - accuracy: 0.9687 - loss: 0.1030 - val_accuracy: 0.9867 - val_loss: 0.0446
Epoch 5/10
750/750 ━━━━━━━━━━━━━━━━━━━━ 40s 54ms/step - accuracy: 0.9740 - loss: 0.0841 - val_accuracy: 0.9883 - val_loss: 0.0403
Epoch 6/10
750/750 ━━━━━━━━━━━━━━━━━━━━ 41s 54ms/step - accuracy: 0.9771 - loss: 0.0736 - val_accuracy: 0.9890 - val_loss: 0.0391
Epoch 7/10
750/750 ━━━━━━━━━━━━━━━━━━━━ 40s 54ms/step - accuracy: 0.9789 - loss: 0.0642 - val_accuracy: 0.9888 - val_loss: 0.0402
Epoch 8/10
750/750 ━━━━━━━━━━━━━━━━━━━━ 41s 54ms/step - accuracy: 0.9803 - loss: 0.0609 - val_accuracy: 0.9892 - val_loss: 0.0422
Epoch 9/10
750/750 ━━━━━━━━━━━━━━━━━━━━ 42s 56ms/step - accuracy: 0.9822 - loss: 0.0559 - val_accuracy: 0.9902 - val_loss: 0.0393
Epoch 10/10
750/750 ━━━━━━━━━━━━━━━━━━━━ 80s 54ms/step - accuracy: 0.9857 - loss: 0.0474 - val_accuracy: 0.9906 - val_loss: 0.0354
```

Explication : J'entraîne le modèle pendant 10 époques pour valider pendant l'entraînement.

3. ANALYSE DES PROJETS

→ Reconnaissance de chiffres manuscrits (MNIST)



Évaluation du modèle

```
test_loss, test_acc = model.evaluate(x_test, y_test)
print("Précision sur le test:", test_acc)
```

```
313/313 ━━━━━━━━━━━━━━━━━━━━ 3s 9ms/step - accuracy: 0.9894 - loss: 0.0370
Précision sur le test: 0.9918000102043152
```

Explication : J'évalue les performances finales sur l'ensemble de test pour voir si le modèle généralise bien.

Explication : Je fais des prédictions sur 10 images de test

Faire des prédictions

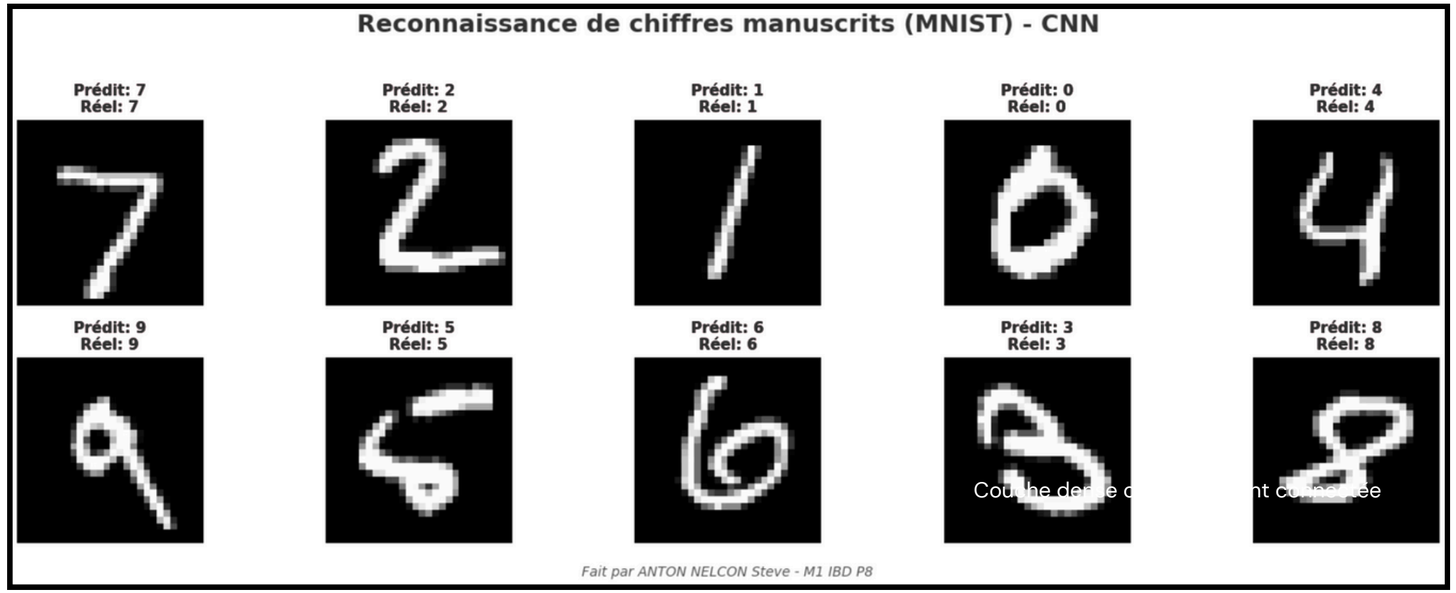
```
# Faire des prédictions sur quelques images de test
predictions = model.predict(x_test)
```

```
313/313 ━━━━━━━━━━━━━━━━━━━━ 3s 8ms/step
```

3. ANALYSE DES PROJETS

→ Reconnaissance de chiffres manuscrits (MNIST)

```python
# Trouver un index unique pour chaque chiffre 0 à 9
unique_digits = []
indices = []

for i, label in enumerate(y_test):
    if label not in unique_digits:
        unique_digits.append(label)
        indices.append(i)
  # Afficher les images avec style
  for i, idx in enumerate(indices):
      plt.subplot(2,5,i+1)
      plt.imshow(x_test[idx].reshape(28,28), cmap='gray')

      pred = np.argmax(predictions[idx])
      true = y_test[idx]

      # Couleur selon la justesse de la prédiction
      color = 'green' if pred == true else 'red'

      plt.title(f"Prédit: {classes[pred]}\nRéel: {classes[true]}",
                color=color, fontsize=11, fontweight='bold')

      plt.axis('off')
      plt.gca().set_facecolor("#f0f0f0")  # fond doux autour de chaque image
      plt.gca().set_frame_on(True)
      plt.gca().spines[:].set_visible(False)
      plt.gca().set_aspect('equal')

  plt.suptitle("Reconnaissance de chiffres manuscrits (MNIST) - CNN",
               fontsize=18, fontweight='bold', color='darkred')

  # Ajouter ton texte personnalisé en bas au centre
  plt.figtext(0.5, 0.02, "Fait par ANTON NELCON Steve - M1 IBD P8",
              ha='center', fontsize=10, style='italic', color='#555')

  plt.tight_layout(rect=[0, 0.05, 1, 0.95])
  plt.show()
```



Explication :Affiche le résultat

RECONNAISSANCE D'EMOTIONS SUR UN VISAGE

3. ANALYSE DES PROJETS

→ Reconnaissance d' émotion sur un visage

```python
# ======================================================================
# Importer les bibliothèques
# ======================================================================
import matplotlib.pyplot as plt
import urllib.request
from PIL import Image
import io
import numpy as np
import tensorflow as tf
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Conv2D, MaxPooling2D, Flatten, Dense, Dropout
```

Explication :Je commence par importer toutes les bibliothèques nécessaires. TensorFlow/Keras pour le modèle et le traitement d'images, Matplotlib pour la visualisation, et Scikit-learn pour l'évaluation.

Explication : Je configure l'organisation des données, avec un dossier par émotion dans les dossiers train, test et validation.

```python
# ======================================================================
# Dictionnaire d'images d'émotions humaines
# ======================================================================
images = {
    "Joie": "https://copilot.microsoft.com/th/id/BCO.65943621-3103-48c7-bfb5-6ed179b8179b.png",
    "Tristesse": "https://copilot.microsoft.com/th/id/BCO.e4d7c443-4a3f-460f-a141-6014d7f16a92.png",
    "Colère": "https://copilot.microsoft.com/th/id/BCO.8f398056-8b9d-4817-b902-5ce4f634ae5a.png",
    "Surprise": "https://copilot.microsoft.com/th/id/BCO.301ef11c-f5a3-4bc8-88f5-5e2ba4612345.png",
    "Neutre": "https://copilot.microsoft.com/th/id/BCO.2000e504-174f-4377-9f91-3a66b761a188.png",
    "Peur": "https://copilot.microsoft.com/th/id/BCO.424fd68b-cfd4-40bc-a305-c5cb99058b64.png",
    "Dégoût": "https://copilot.microsoft.com/th/id/BCO.90a3e800-6e18-48b6-ab96-fd6cbdc49ca4.png",
    "Honte": "https://copilot.microsoft.com/th/id/BCO.8a9f1439-04f9-4701-92ad-4bee54dec61d.png",
    "Fierté": "https://copilot.microsoft.com/th/id/BCO.6049dcd1-cbe6-41be-8ec0-3b30c239de94.png",
    "Fatigue": "https://copilot.microsoft.com/th/id/BCO.71a9cc48-023e-462a-973f-0a460fea43b9.png"
}
```

3. ANALYSE DES PROJETS

→ Reconnaissance d' émotion sur un visage

```python
# ================================================================
# Préparation du "jeu de données" factice (pour la démonstration)
# ================================================================
emotion_labels = list(images.keys())  # Récupère la liste des émotions (les clés du dictionnaire "images")
num_classes = len(emotion_labels) # Nombre total de classes → taille de la dernière couche du modèle

x_data, y_data = [], []  # Tableaux : x_data = images, y_data = labels
for idx, (emotion, url) in enumerate(images.items()):
    with urllib.request.urlopen(url) as response: # Téléchargement de l'image via son URL
        img_data = response.read()
    img = Image.open(io.BytesIO(img_data)).convert('RGB').resize((64, 64))
    x_data.append(np.array(img))
    y_data.append(idx)    # Ajout du label correspondant (idx) dans y_data

x_data = np.array(x_data) / 255.0  # Conversion des listes en tableaux numpy + normalisation des pixels (0-255 → 0-1)
y_data = np.array(y_data)  # Conversion des labels en tableau numpy
```

1. **Explication :** Je récupère la liste des 10 émotions et je compte combien il y a de classes. Ce nombre sera important pour la dernière couche du modèle.

Je parcours chaque émotion et son URL. Pour chaque image, je télécharge depuis internet, je la convertis en RGB, je la redimensionne à 64×64 pixels, et je l'ajoute à mes données.

Explication : Normalisation des données

```
x_data = np.array(x_data) /
y_data = np.array(y_data)  #
```

```python
# ================================================================
# Création d'un petit modèle CNN
# ================================================================
model = Sequential([
    Conv2D(16, (3,3), activation='relu', input_shape=(64,64,3)), ###  l'image (64x64 en couleur : 3 canaux)
    MaxPooling2D((2,2)),  ##  Première couche de MaxPooling : réduit la taille de l'image en prenant les valeurs max
    Conv2D(32, (3,3), activation='relu'), ## # Deuxième couche de convolution : maintenant 32 filtres pour détecter des motifs plus complexes
    MaxPooling2D((2,2)), ##  Deuxième MaxPooling pour réduire encore la dimension
    Flatten(), ## # Aplatissement : transforme la matrice en vecteur pour la connecter au réseau dense
    Dense(64, activation='relu'), ##  # Couche Dense avec 64 neurones et activation ReLU : apprend les patterns extraits
    Dropout(0.3), ## # Dropout à 30% pour éviter le surapprentissage
    Dense(num_classes, activation='softmax') ##  # Dernière couche : autant de neurones que de classes, activation softmax → probas
])
```

Explication :

Je construis mon modèle CNN. La première couche a 16 filtres de taille 3×3 avec activation ReLU. L'image d'entrée fait 64×64×3.

J'ajoute un max-pooling qui réduit la dimension de l'image de moitié (de 64×64 à 32×32).

Je mets une deuxième couche avec 32 filtres pour détecter des motifs plus complexes.

Encore un max-pooling qui réduit de 32×32 à 16×16.

J'utilise Flatten() pour transformer les matrices 2D en un vecteur 1D, nécessaire pour les couches denses.

J'ajoute une couche dense avec 64 neurones qui apprend les combinaisons des caractéristiques extraites.

Je mets un dropout de 30% pour éviter que le modèle mémorise trop les données d'entraînement.

La dernière couche a 10 neurones (un par émotion) avec activation softmax pour avoir des probabilités.

```
model.compile(optimizer='adam',
              loss='sparse_categorical_crossentropy',  # Pour des labels sous forme de chiffres (0,1,2...)
              metrics=['accuracy']) ##  # On veut mesurer la précision du modèle
```

Explication :
Je configure l'apprentissage avec l'optimiseur Adam, la fonction de perte adaptée, et l'accuracy comme métrique

```
# Entraînement rapide (démonstration)
history = model.fit(x_data, y_data, epochs=10, verbose=0)
```

Explication :
J'entraîne le modèle sur mes 10 images pendant 10 époques.

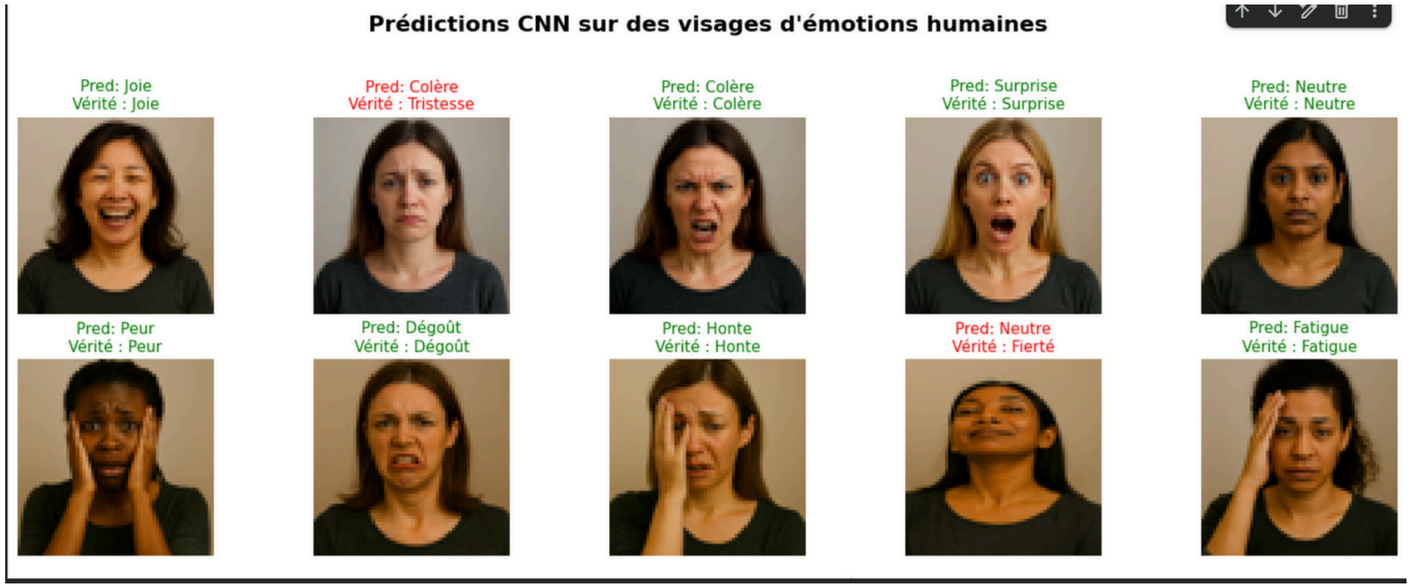
```
# ============================================================
# Prédiction des émotions
# ============================================================
predictions = model.predict(x_data)
```

Explication :
Je fais prédire au modèle l'émotion de chaque image. Chaque prédiction donne 10 probabilités.

3. ANALYSE DES PROJETS

→ Reconnaissance d'emotions sur le visage

```
# ======================================================================
# Affichage des images avec leurs prédictions
# ======================================================================
plt.figure(figsize=(15,6))
for i in range(len(x_data)):
    plt.subplot(2,5,i+1)
    plt.imshow(x_data[i])
    pred_label = np.argmax(predictions[i])
    true_label = y_data[i]
    color = 'green' if pred_label == true_label else 'red'
    plt.title(f"Pred: {emotion_labels[pred_label]}\nVérité : {emotion_labels[true_label]}", color=color, fontsize=11)
    plt.axis('off')

plt.suptitle("Prédictions CNN sur des visages d'émotions humaines", fontsize=16, fontweight='bold')
plt.figtext(0.5, 0.02, "Fait par Anton Nelcon Steve - M1 IBD - Université Paris 8", ha='center', style='italic', fontsize=10)
plt.tight_layout(rect=[0, 0.05, 1, 0.95])
plt.show()
```



Explication :Affiche le résultat

04

CONCLUSION

→ Mon ressentie

DEEP
LEARNING

NOUVELLE
BIBLIOTHEQUE

05

RESSOURCES

→ Ressources

Réalisé par **ANTON NELCON Steve**

1.4 Fondamentaux des CNN

1.5 Architecture des CNN

Un réseau de neurones convolutif est composé de plusieurs types de couches :

1.6 Couches de convolution

Les couches de convolution appliquent des filtres (ou kernels) sur l'image d'entrée pour extraire des caractéristiques locales. Mathématiquement, l'opération de convolution s'écrit :

$$(f * g)(x, y) = \sum_{i=-\infty}^{\infty} \sum_{j=-\infty}^{\infty} f(i, j) \cdot g(x - i, y - j) \tag{1}$$

1.7 Couches de pooling

Les couches de pooling réduisent la dimensionnalité spatiale des features maps, généralement par :

- **Max Pooling :** Sélection de la valeur maximale dans une fenêtre
- **Average Pooling :** Calcul de la moyenne des valeurs

1.8 Couches fully connected

Les couches entièrement connectées (dense layers) effectuent la classification finale en combinant toutes les features extraites.

1.9 Fonctions d'activation

Plusieurs fonctions d'activation sont utilisées :

- **ReLU** (Rectified Linear Unit) : $f(x) = \max(0, x)$
- **Softmax** : $\sigma(z)_i = \frac{e^{z_i}}{\sum_{j=1}^{K} e^{z_j}}$

1.10 Fonction de coût et optimisation

La fonction de coût utilisée est généralement l'entropie croisée catégorielle :

$$L = -\sum_{i=1}^{N} \sum_{c=1}^{C} y_{i,c} \log(\hat{y}_{i,c}) \tag{2}$$

où $y_{i,c}$ est la vraie étiquette et $\hat{y}_{i,c}$ la prédiction.

FIN

MERCI DE VOTRE ECOUTE !!!

Présenté par **ANTON NELCON Steve**