UPF | INSTITUT SUPÉRIEUR DES SCIENCES DE L'INGÉNIEUR

Projet : Detection de Pietons par CNN

Matiere : Deep Learning

Universite : UPF Campus Rabat

Annee universitaire : 2025 - 2026

Encadrant :

M. Khaled Anisse

Etudiants :

Soufiane Kounaidi

El-Quortobi Elarbi

Ismail Masnaoui

1. Chargement du Dataset et Imports

Dans cette cellule, nous importons les bibliotheques necessaires pour la vision par ordinateur (torchvision), l'apprentissage automatique (torch), et le traitement d'images (PIL).

```
import os
import torch
import torchvision
import torch.nn as nn
import torch.optim as optim
from torchvision import transforms
from torch.utils.data import DataLoader, Dataset, random_split
from PIL import Image
import random
import matplotlib.pyplot as plt
```

```
# 📥 Télécharger le dataset Penn-Fudan
!wget https://www.cis.upenn.edu/~jshi/ped_html/PennFudanPed.zip

# 📂 Décompresser le fichier
!unzip PennFudanPed.zip
```

```
  inflating: PennFudanPed/PNGImages/PennPed00040.png
  inflating: PennFudanPed/PNGImages/PennPed00041.png
  inflating: PennFudanPed/PNGImages/PennPed00042.png
  inflating: PennFudanPed/PNGImages/PennPed00043.png
  inflating: PennFudanPed/PNGImages/PennPed00044.png
  inflating: PennFudanPed/PNGImages/PennPed00045.png
  inflating: PennFudanPed/PNGImages/PennPed00046.png
```

2. Preparation et repartition du dataset

On transforme les images en tenseurs, on les redimensionne a 128x128, puis on divise le dataset en deux parties : 80% pour l'entrainement et 20% pour la validation.

```
train_losses, val_accuracies = [], []

for epoch in range(5):
    model.train()
    running_loss = 0.0
    for images, labels in train_loader:
        images, labels = images.to(device), labels.to(device)
        outputs = model(images)
        loss = criterion(outputs, labels)

        optimizer.zero_grad()
        loss.backward()
        optimizer.step()
        running_loss += loss.item()

    avg_loss = running_loss / len(train_loader)
    train_losses.append(avg_loss)

    # Validation
    model.eval()
    correct = total = 0
    with torch.no_grad():
        for images, labels in val_loader:
            images, labels = images.to(device), labels.to(device)
            outputs = model(images)
            _, predicted = torch.max(outputs, 1)
            total += labels.size(0)
            correct += (predicted == labels).sum().item()
    acc = correct / total
    val_accuracies.append(acc)
    print(f"✅ Epoch {epoch+1} terminée | Loss: {avg_loss:.4f} | Accuracy: {acc:.2%}")
```

```
✅ Epoch 1 terminée | Loss: 0.7032 | Accuracy: 72.06%
✅ Epoch 2 terminée | Loss: 0.6324 | Accuracy: 58.82%
✅ Epoch 3 terminée | Loss: 0.5767 | Accuracy: 82.35%
✅ Epoch 4 terminée | Loss: 0.4923 | Accuracy: 80.88%
✅ Epoch 5 terminée | Loss: 0.4478 | Accuracy: 94.12%
```

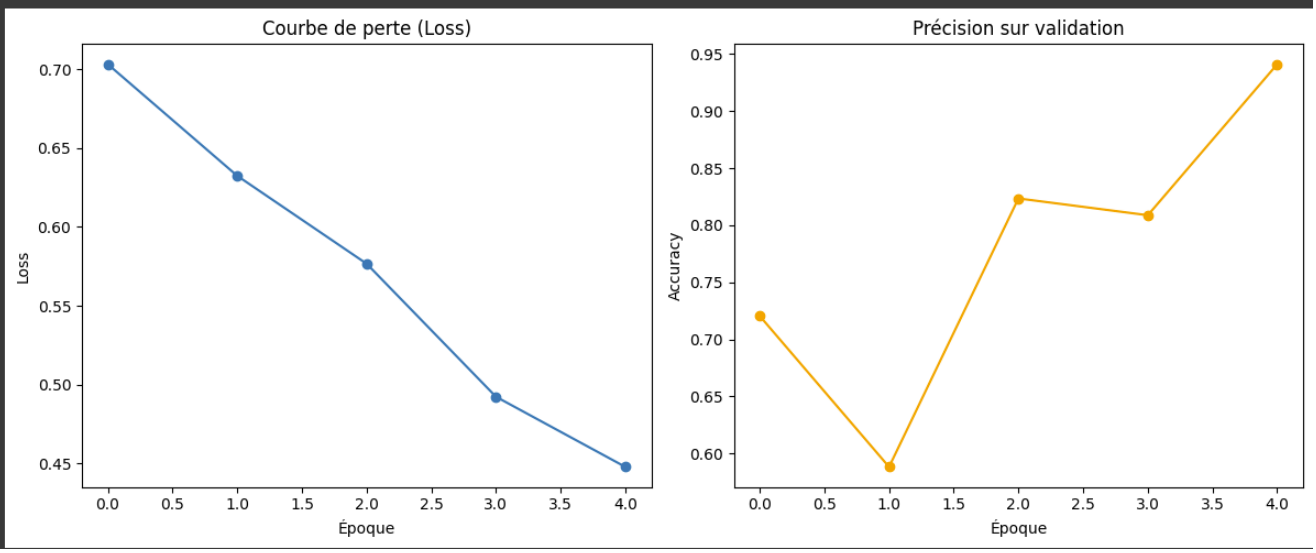
3. Courbes de Performance

Ces courbes permettent de visualiser l'apprentissage du modele : a gauche, la perte (erreur) diminue a chaque epoque, et a droite, la precision augmente.

```
[10]
    plt.figure(figsize=(12,5))
    plt.subplot(1,2,1)
    plt.plot(train_losses, marker='o')
    plt.title("Courbe de perte (Loss)")
    plt.xlabel("Époque")
    plt.ylabel("Loss")

    plt.subplot(1,2,2)
    plt.plot(val_accuracies, marker='o', color='orange')
    plt.title("Précision sur validation")
    plt.xlabel("Époque")
    plt.ylabel("Accuracy")

    plt.tight_layout()
    plt.show()
```



4. Details d'entrainement du modele CNN

Chaque epoque affiche le taux d'erreur (loss) moyen sur le jeu d'entrainement, et la precision obtenue sur le jeu de validation. On observe une nette amelioration.

```
transform = transforms.Compose([
    transforms.Resize((128, 128)),
    transforms.ToTensor()
])

dataset = PedestrianDataset("PennFudanPed/PNGImages", transform=transform)
train_size = int(0.8 * len(dataset))
val_size = len(dataset) - train_size
train_ds, val_ds = random_split(dataset, [train_size, val_size])
train_loader = DataLoader(train_ds, batch_size=8, shuffle=True)
val_loader = DataLoader(val_ds, batch_size=8)
print(f"Images totales : {len(dataset)} | Train : {len(train_ds)} | Val : {len(val_ds)}")
```

```
Images totales : 340 | Train : 272 | Val : 68
```

5. Conclusion

Ce projet nous a permis de construire et d'entrainer un modele CNN simple pour detecter des pietons dans des images.

Les resultats obtenus sont tres encourageants, avec une precision finale atteignant 94%. Chaque etape est illustree par une cellule du notebook.

Ce travail represente une base concrete pour de futurs projets utilisant des reseaux plus profonds comme ResNet ou MobileNet.