La détection des piétons dans la route

Projet de fin de module

Réalisé par : Ghizlane Hafsi
Encadré par : Mme Sara Baghdadi

Mai 2025

Plan de la présentation

1. Introduction
2. Présentation de YOLOv5
3. Méthodologie
4. Entraînement du Modèle
5. Évaluation et Résultats
6. Démonstration
7. Applications et Limites
8. Conclusion
9. Présentation de YOLOv5
10. Cas Pratique : Détection de Piétons avec HOG + SVM

Introduction – L'importance de la détection des piétons sur la route (Partie 1)

La détection des piétons dans la rue représente un défi majeur pour la sécurité routière. Chaque jour, des millions de piétons traversent les routes, et une proportion importante des accidents de la route implique des piétons.

Ces derniers sont particulièrement vulnérables aux collisions avec des véhicules en raison de leur absence de protection physique.

Introduction – L'importance de la détection des piétons sur la route (Partie 2)

Les technologies de vision par ordinateur, telles que les réseaux neuronaux convolutifs (CNN) et l'apprentissage profond (deep learning), ont transformé cette tâche.

Des algorithmes comme YOLOv5 (You Only Look Once)permettent une détection rapide et précise des piétons à partir d'images et de vidéos en temps réel, capturées par des caméras embarquées.

Introduction – L'importance de la détection des piétons sur la route (Partie 3)

Ce projet vise à implémenter cette technologie de détection des piétons en utilisant YOLOv5 pour aider à la réduction des accidents de la route et des victimes.

L'intégration de cette technologie représente une avancée majeure pour la sécurité des usagers de la route.

Qu'est-ce que YOLO ?

YOLO signifie *You Only Look Once*.

C'est un algorithme de **détection d'objets en temps réel** basé sur les réseaux neuronaux convolutifs (CNN).

Il détecte les objets (*piétons*, *voitures*, *etc.*) et prédit leur classe et leur position **en une seule passe** à travers le réseau.

Pourquoi utiliser YOLO ?

- Rapidité exceptionnelle : parfait pour les applications en temps réel.
- Précision compétitive : bonnes performances même avec des objets multiples.
- Tout-en-un : classification et localisation faites simultanément.
- Utilisé dans la vidéosurveillance, les voitures autonomes, etc.

Étape 1 : Division de l'image

YOLO divise l'image d'entrée en une grille $S \times S$.

- Chaque cellule de la grille est responsable des objets dont le **centre** se trouve dans cette cellule.
- Par exemple, si $S = 7$, l'image est divisée en 49 cellules.

Étape 2 : Prédiction par cellule

Chaque cellule prédit :

- Des **boîtes englobantes (bounding boxes)** : (x, y, w, h)
- Un **score de confiance** : probabilité qu'un objet est présent.
- Les **probabilités de classe** : piéton, voiture, panneau, etc.

Détail d'une Bounding Box

Une boîte est définie par :

- x, y : coordonnées du **centre** de la boîte (normalisées)
- w, h : largeur et hauteur (aussi normalisées)

YOLO prédit aussi :

- Le **score de confiance** : entre 0 et 1
- Les **classes** avec leurs probabilités

Étape 3 : Application de seuil

YOLO rejette les boîtes dont le **score de confiance est trop faible**.

Exemple :

- Boîte A : score 0.91 $\Rightarrow$ gardée
- Boîte B : score 0.42 $\Rightarrow$ rejetée

Seuil classique : 0.5, mais ajustable selon l'application.

Base de données INRIA (version YOLOv5)

- Base adaptée à la détection des piétons
- Contient **902 images** annotées
- Structure du dossier :
  - `train/images` et `train/labels`
  - `valid/images` et `valid/labels`
  - `test/images` et `test/labels`
- Fichier `data.yaml` pour la configuration

Fichier data.yaml — Configuration du Dataset

- Définit les chemins vers les données :
  - `train: ../train/images`
  - `val: ../valid/images`
  - `test: ../test/images`
- Nombre de classes : `nc: 1`
- Noms des classes : `names: ['person']`
- Indispensable pour entraîner YOLOv5

Fichier de labels (format YOLOv5)

- Exemple de fichier `label.txt` :

Contenu

```
0 0.51875 0.40278 0.26250 0.52778
```

- Signification :
  - `0` : ID de la classe (ici, "person")
  - `x_center`, `y_center` : coordonnées du centre
  - `width`, `height` : dimensions de la boîte
- Tous les chiffres sont **normalisés** entre 0 et 1

Utilité du format YOLO

- Permet une annotation légère et rapide à traiter
- Pas besoin des coordonnées exactes de coins
- Idéal pour une détection temps réel
- Compatible avec de nombreux frameworks (YOLOv5, YOLOv8...)

Entraînement du modèle

- Utilisation du modèle pré-entraîné `yolov5s.pt`
- Entraînement sur le dataset INRIA pendant 50 époques
- Données définies dans le fichier `data.yaml`

Code

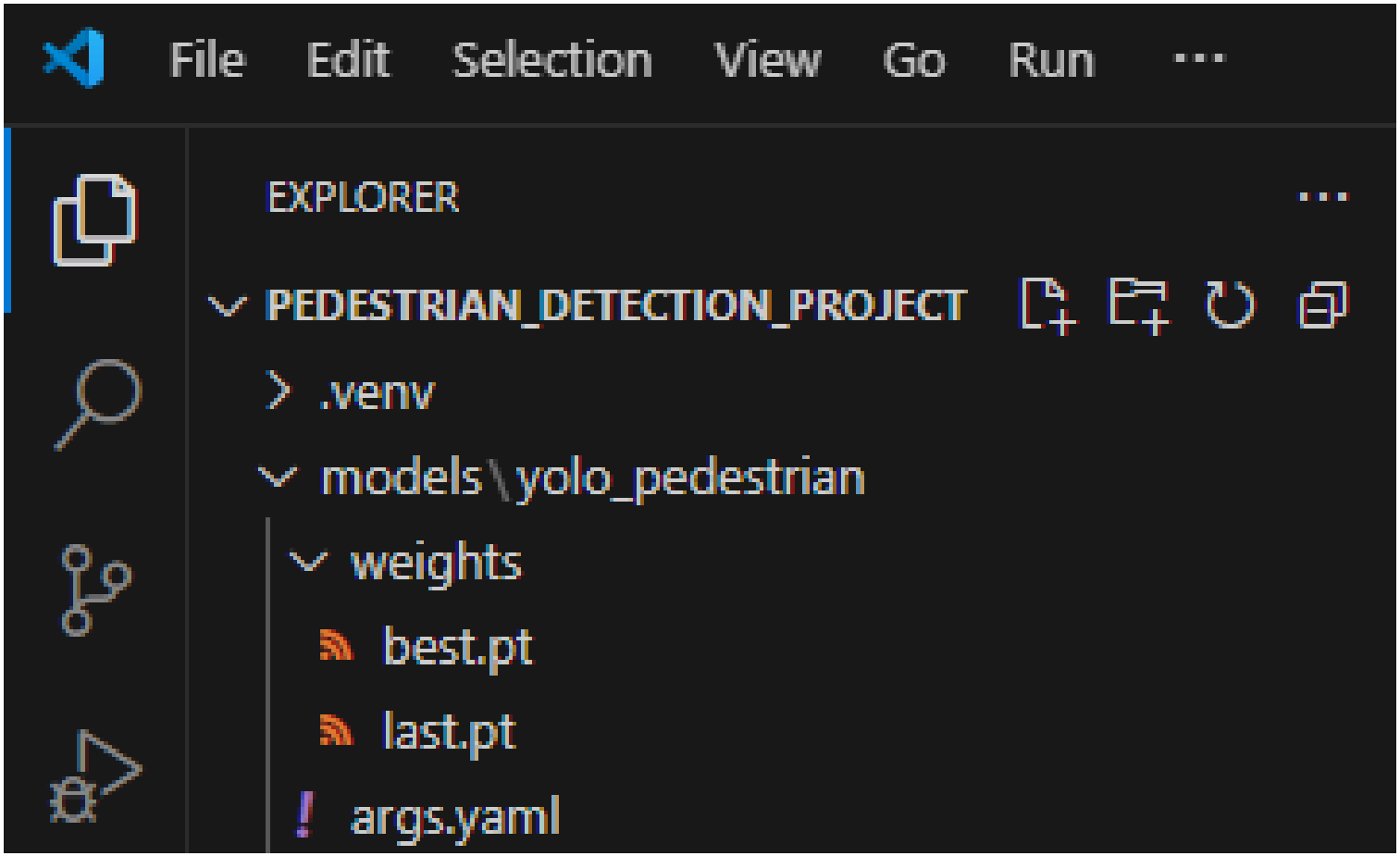
```
from ultralytics import YOLO
model = YOLO("yolov5s.pt")
model.train(
    data="data.yaml",
    epochs=50,
    imgsz=640,
    batch=16,
    project="models",
    name="yolo_pedestrian",
    device="cpu"
)
```

Fichiers de poids sauvegardés

Lors de l'entraînement du modèle YOLOv5, deux fichiers importants sont générés :

1. `best.pt` :
   - Contient le modèle ayant obtenu les meilleures performances sur les données de validation.
   - Sauvegardé automatiquement pendant l'entraînement.
   - À utiliser pour les tests, la démonstration et les prédictions finales.
2. `last.pt` :
   - Contient l'état du modèle à la fin de la dernière époque d'entraînement.
   - Peut être utilisé pour reprendre l'entraînement ou comparaison.

Structure des fichiers du modèle entraîné

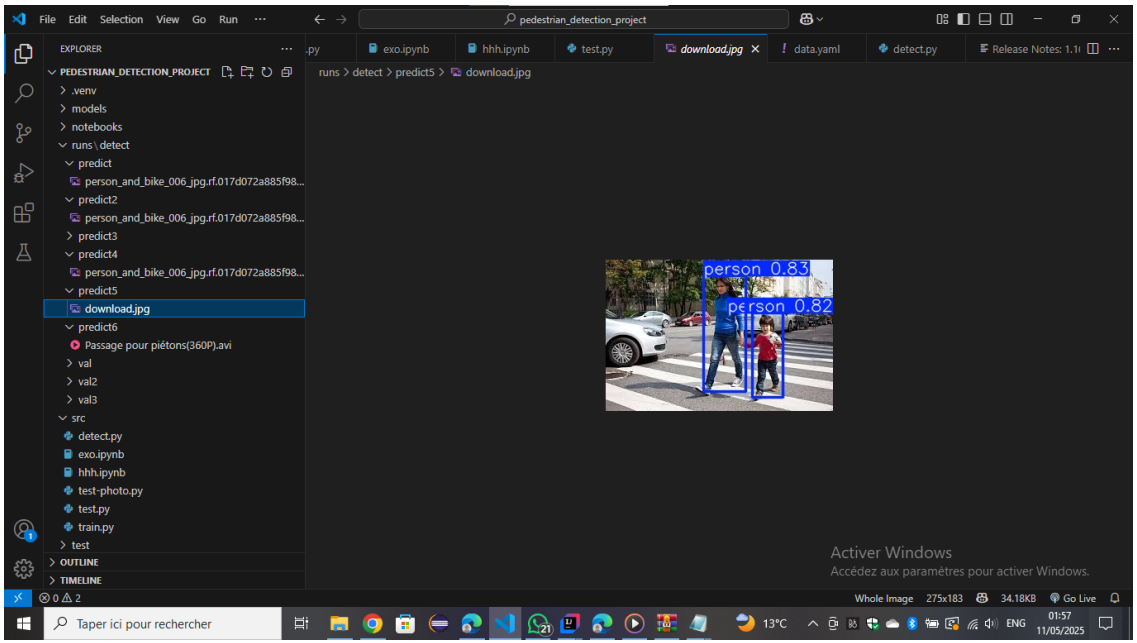


Chemin : `models/yolo_pedestrian/weights/`

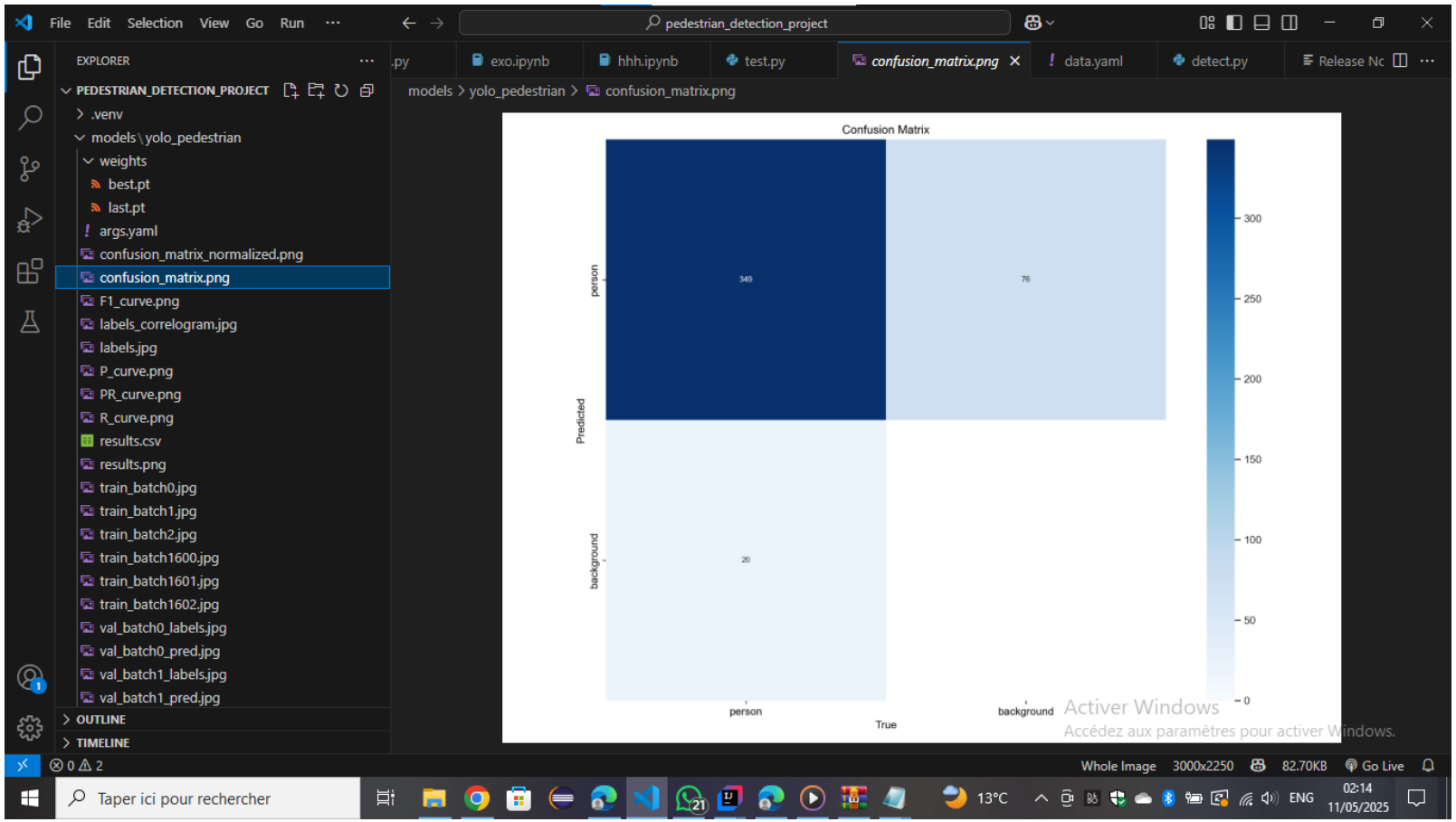
- `best.pt` : Meilleur modèle sauvegardé.
- `last.pt` : Dernier état du modèle à la fin de

Test du modèle sur une nouvelle image

- Le modèle YOLOv5 entraîné a été testé sur une nouvelle image.
- Deux piétons ont été détectés avec des scores de confiance de 0.83 et 0.82.
- La détection est représentée par des boîtes englobantes bleues avec les étiquettes "person".



Matrice de Confusion



figureMatrice de confusion du modèle YOLO pour la détection de piétons

Interprétation

- **349** vrais positifs (piétons bien détectés)
- **76** faux positifs (zones du fond mal classées comme piéton)
- **20** faux négatifs (piétons non détectés)
- **0** vrais négatifs (très peu de fond bien classé, normal en détection)

SVM+HOG

- Détecter les piétons dans des images à l'aide des descripteurs HOG (Histogram of Oriented Gradients) et d'un classifieur SVM (Support Vector Machine).
- Exemple d'approche classique en Machine Learning pour la vision par ordinateur.

Pipeline de détection

1. Chargement des images positives (piétons) et négatives (fond).
2. Extraction des caractéristiques avec `cv2.HOGDescriptor`.
3. Entraînement d'un SVM linéaire.
4. Application du modèle sur de nouvelles images.
5. Affichage des boîtes englobantes autour des piétons détectés.

Base de données : Images Positives et Négatives

- **Images positives** : Contiennent des piétons, utilisées pour apprendre à détecter les formes humaines.
- **Images négatives** : Ne contiennent pas de piétons, servent à apprendre ce qu'il ne faut pas détecter.
- **Prétraitement** :
  - Redimensionnement des images à une taille fixe (par ex. 64x128 pixels).
  - Extraction de caractéristiques HOG à partir de chaque image.

But

Créer un jeu de données équilibré pour entraîner un SVM à distinguer piétons et arrière-plans.

Exemples de la base de données

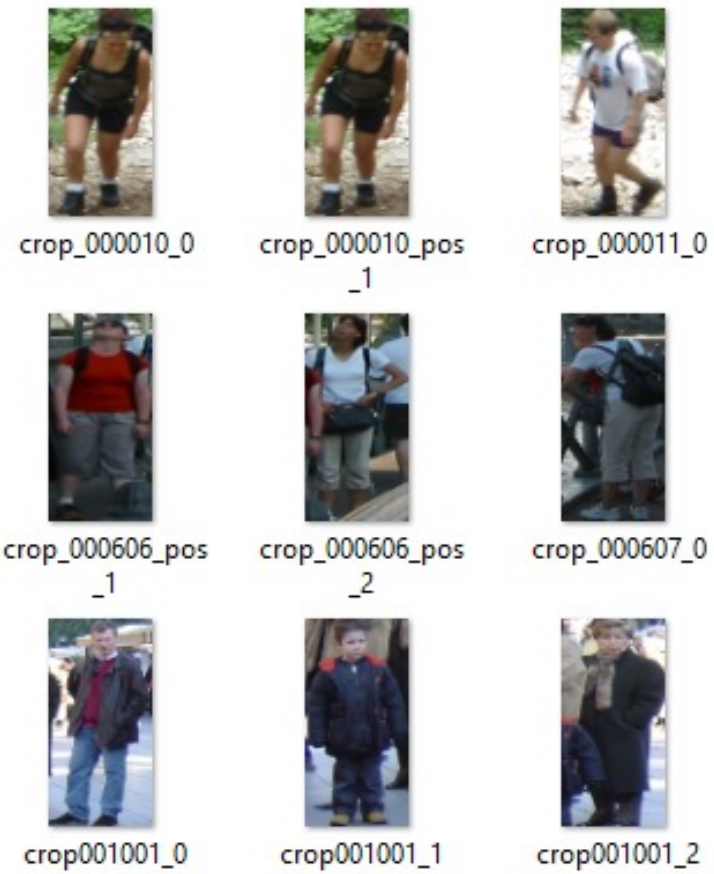


Figure – *

Image Positive (avec piéton)

Figure – *

Image Négative (sans piéton)

Extraction HOG et Entraînement SVM

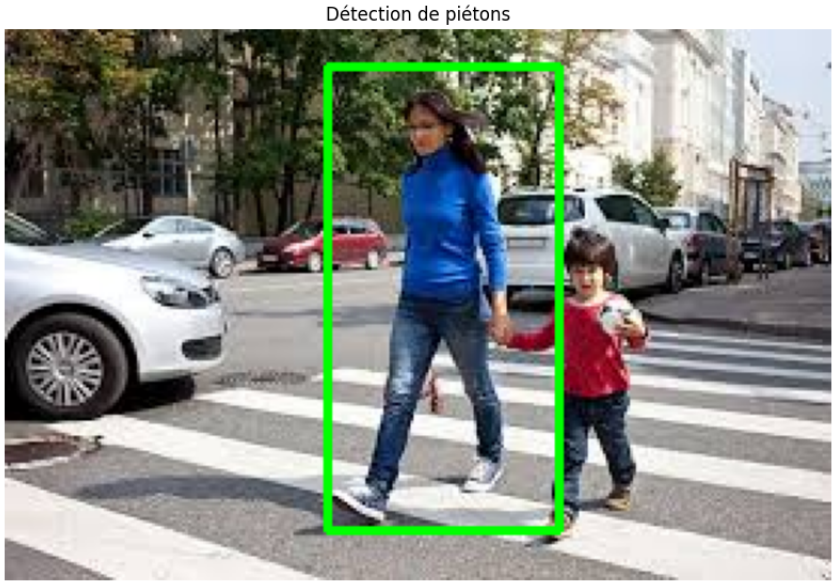
1. Extraction des caractéristiques HOG

- Utilisation de `cv2.HOGDescriptor()` d'OpenCV.
- HOG (Histogram of Oriented Gradients) extrait les bords et les formes dans une image.
- Chaque image est convertie en un vecteur de caractéristiques.
- Exemple de paramètres :
  - Taille de cellule : 8x8
  - Bloc : 2x2 cellules
  - Orientation : 9 bins

2. Entraînement du classifieur SVM

- Les vecteurs HOG sont associés à leurs étiquettes : `+1` pour les images positives (piétons), `-1` pour les négatives.
- SVM linéaire entraîné pour séparer les deux classes.
- Utilisation de `cv2.ml.SVM_create()` ou un SVM de `scikit-learn`.

Résultats de la détection



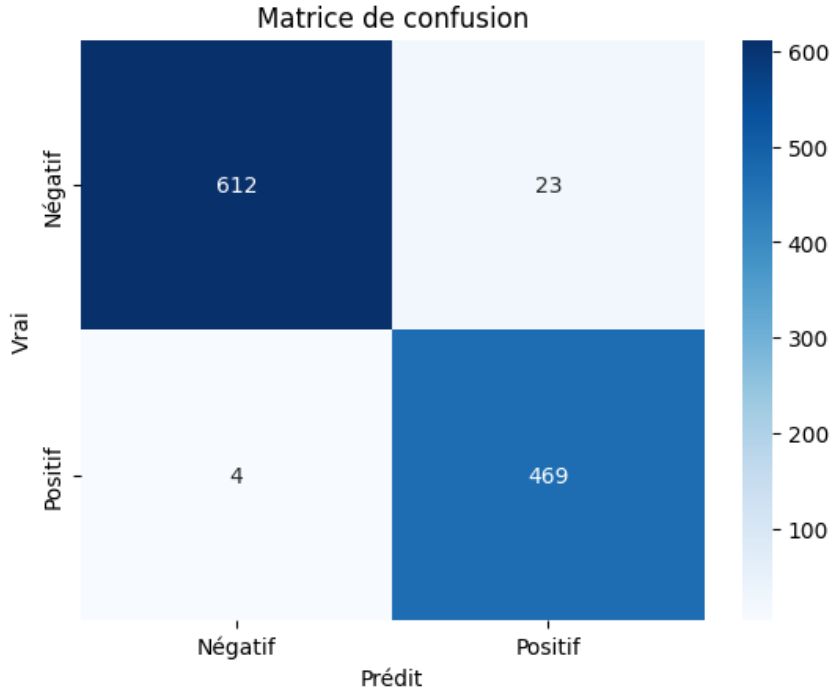
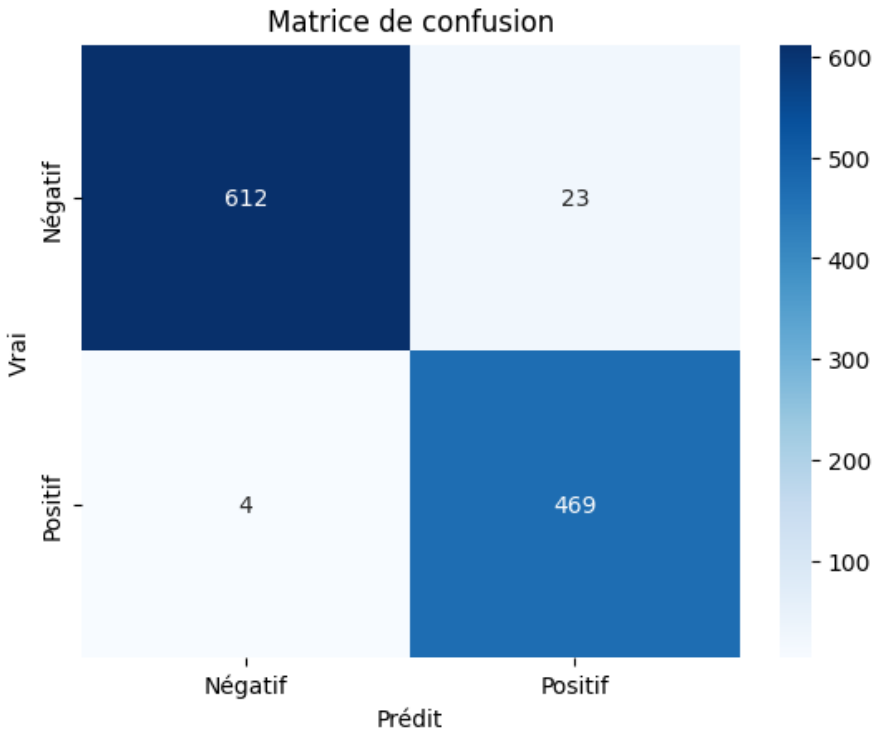


Figure – Détection de piétons avec HOG + SVM

Évaluation du Modèle : Matrice de Confusion

- **Vrais positifs (VP)** : 469
  (Piétons bien détectés)
- **Faux positifs (FP)** : 23
  (Faux piétons détectés)
- **Vrais négatifs (VN)** : 612
  (Absence de piéton bien détectée)
- **Faux négatifs (FN)** : 4
  (Piétons manqués)



*Matrice de confusion du classifieur SVM

Interprétation

Le modèle présente une excellente performance globale avec très peu d'erreurs.

Analyse - HOG vs Deep Learning

Méthode	**Avantages**	**Limites**
HOG + SVM	Rapide, simple, peu de données	Moins précis, pas r
YOLO/CNN	Très précis, robuste	Besoin de GPU et beauco

Conclusion

- La détection des piétons est une tâche essentielle pour améliorer la sécurité routière, notamment dans les systèmes d'assistance à la conduite et les véhicules autonomes.
- Dans ce projet, deux approches complémentaires ont été explorées :
  - Une méthode classique basée sur l'extraction de caractéristiques HOG (Histogram of Oriented Gradients) et l'entraînement d'un SVM linéaire.
  - Une approche avancée avec le modèle de détection temps réel **YOLOv5**, capable de localiser plusieurs piétons avec une grande précision.
- L'utilisation de **HOG + SVM** permet une compréhension fine du pipeline de détection classique, tandis que **YOLOv5** offre une solution puissante et adaptée aux applications modernes en temps réel.
- L'intégration de ces techniques contribue activement au développement de systèmes intelligents de prévention des accidents impliquant des piétons.