



Détection et reconnaissance faciale

Bastien Henry - Rayane Nadaud - Florian Poscente

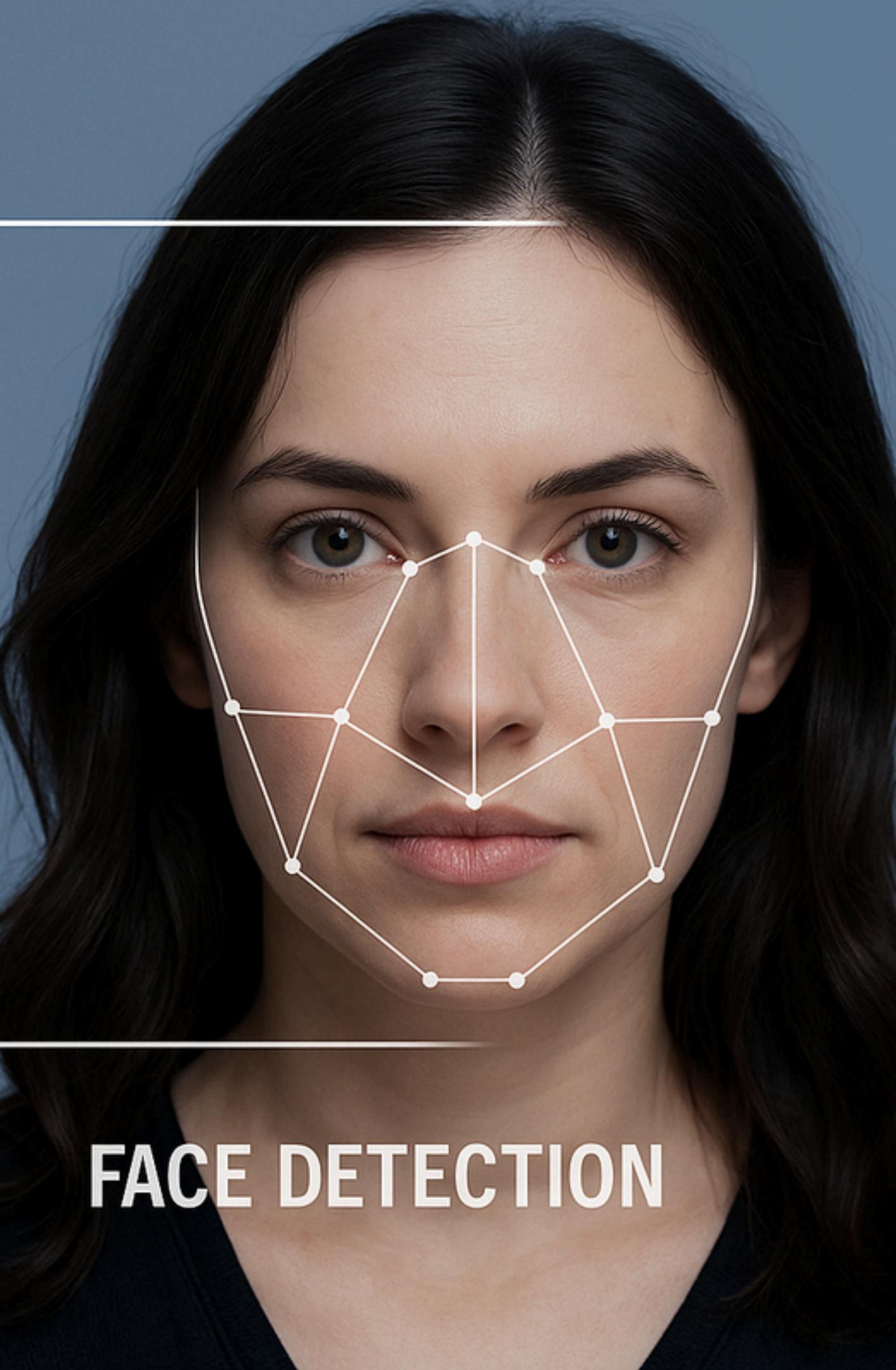
Module : ST2IAI – M1 T2IA

Enseignante : Faten Chakchouk



Contexte

- La reconnaissance faciale est aujourd'hui utilisée dans de nombreux domaines (sécurité, déverrouillage de smartphones, paiements sans contact, etc.).
- Notre projet compare méthodes classiques et modernes pour améliorer ses performances.

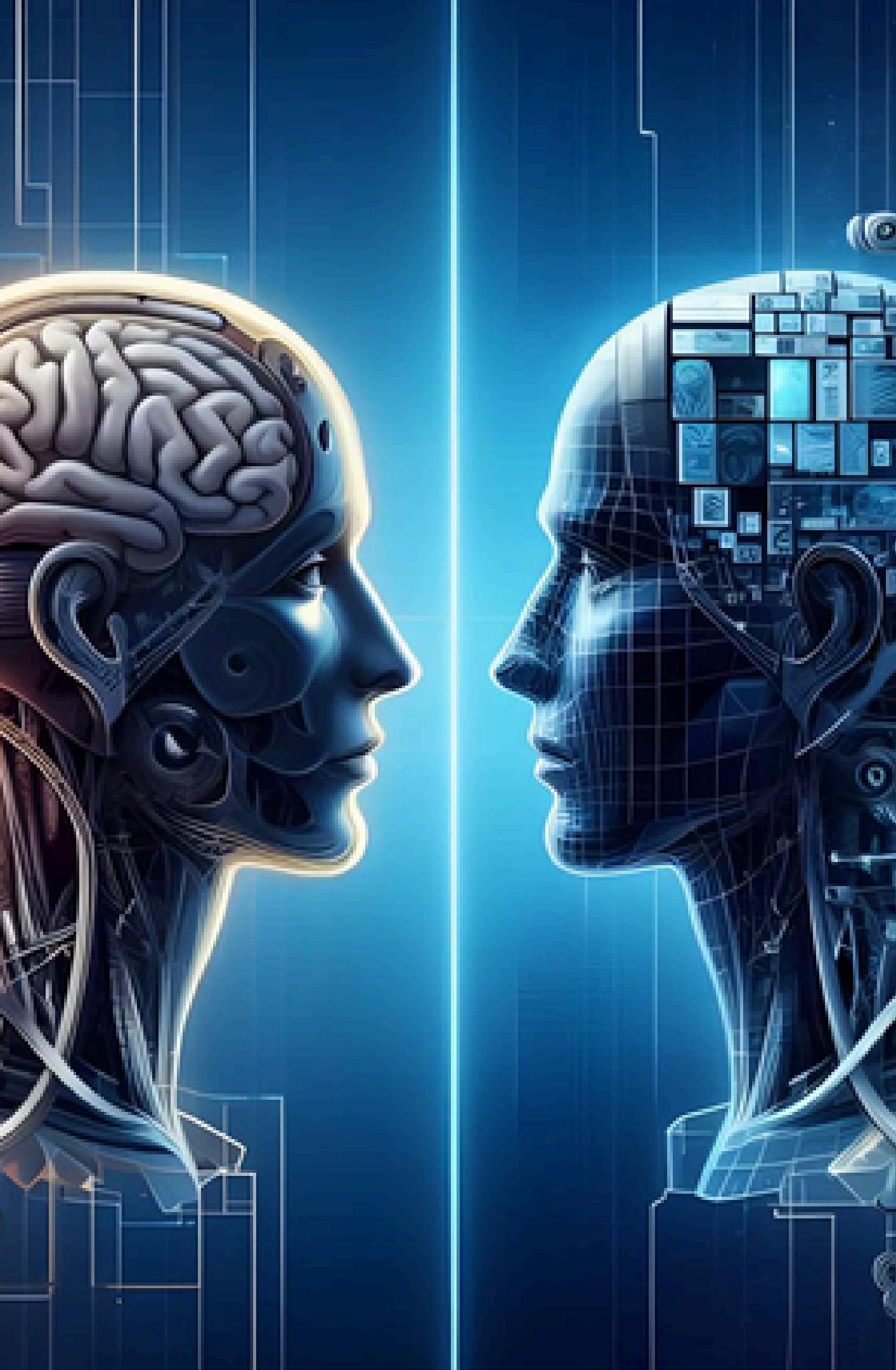


FACE DETECTION

Objectif spécifique du projet

Concevoir un pipeline complet de reconnaissance faciale en temps réel :

- Commencer par une méthode classique pour la détection : Haar Cascade.
- Tester ensuite des approches plus avancées pour la reconnaissance (HOG + Machine LearningCNN Deep Learning)
- Comparer ces méthodes en termes de précision et de rapidité.



État de l'Art

Méthodes Classiques

- Haar Cascade: détection rapide mais sensible aux variations
- HOG : meilleure robustesse aux changements d'illumination

Avantages: simplicité, rapidité d'exécution, faibles ressources requises

Inconvénients: précision limitée dans des conditions variables

Approches Modernes

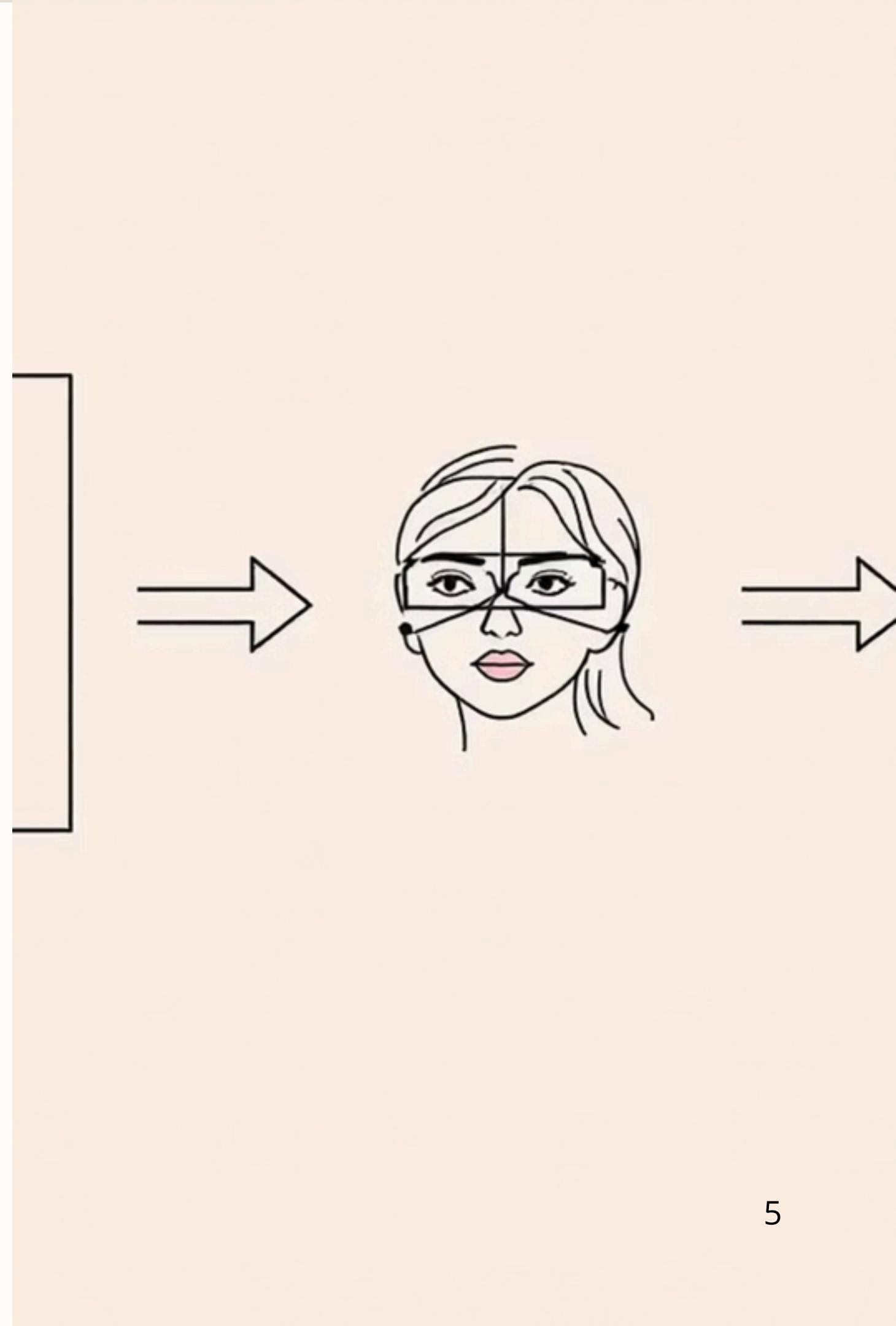
- CNN (Réseaux de Neurones Convolutifs): MTCNN, RetinaFace

Avantages: haute précision, robustesse aux variations

Inconvénients: coût computationnel élevé, besoin de données massives

Pipeline Proposé

- Détection de visage
 - Haar Cascade (OpenCV)
 - MediaPipe Face Detection
- Méthode de classification
 - Decision Tree
 - SVM
 - Random forest
- Descripteur
 - HOG
- Réduction de dimension
 - PCA

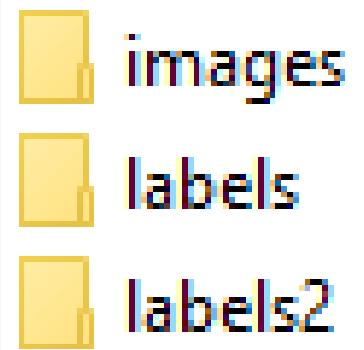


Datasets utilisés



Partie Detection de visage

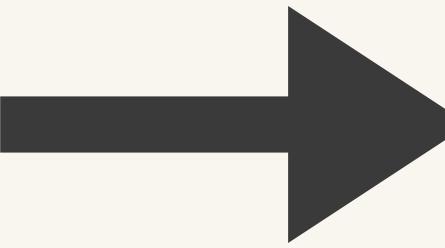
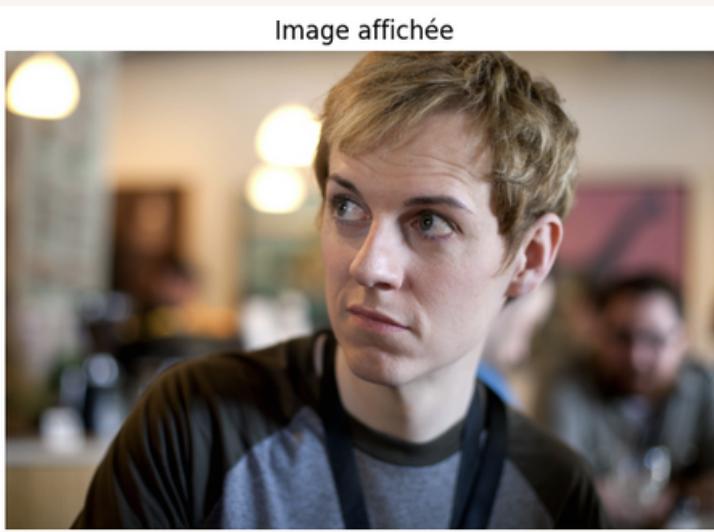
- Kaggle Face Detection Dataset
- Contient de nombreuses images en format JPG
- Possède l'emplacement des visage (ground-truth) dans fichier TXT



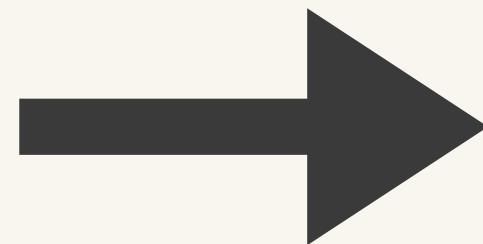
Partie Reconnaissance

- fetch_lfw_people - Labeled Faces in the Wild :
- Environ 13 000 images

État d'avancement



Utilisation de Haar Cascade



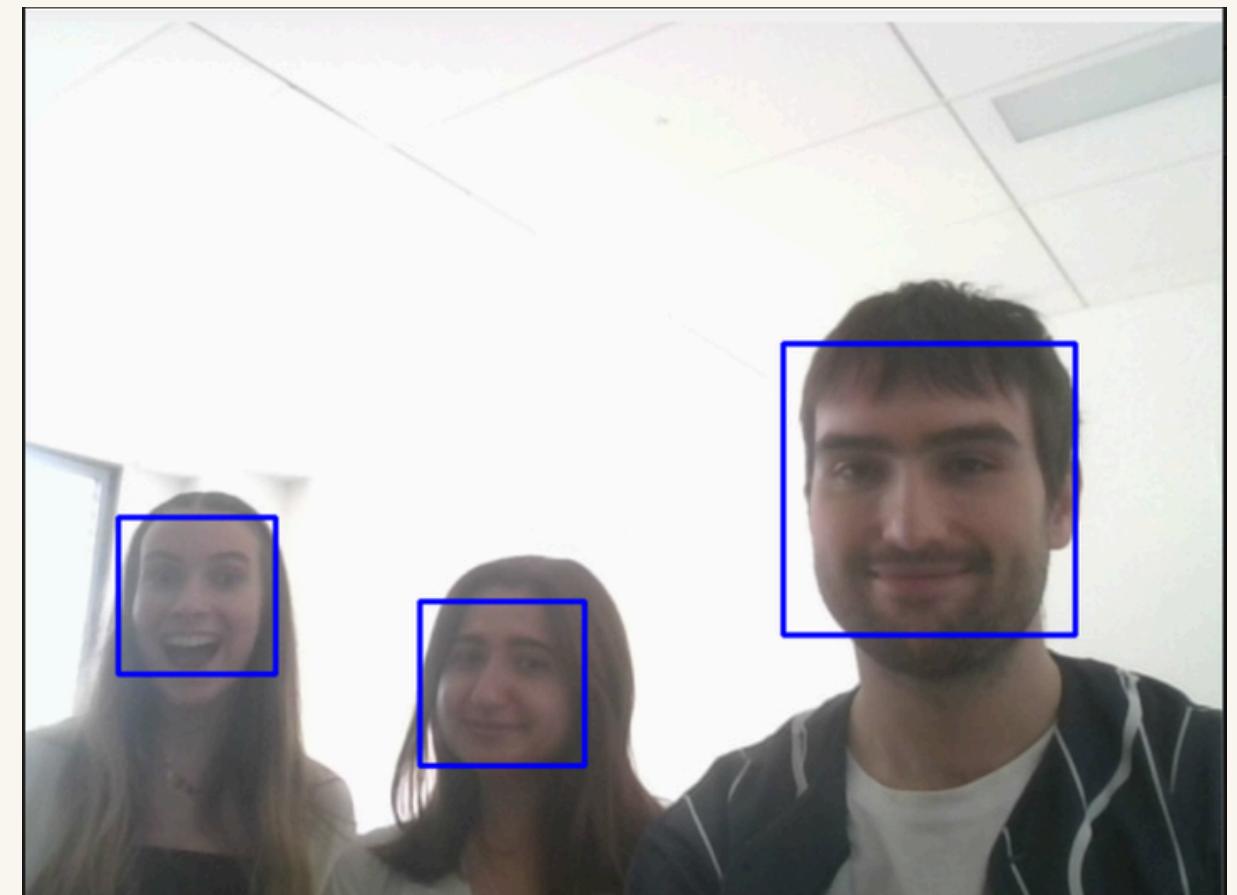
État d'avancement

Lire à partir de la WebCam

```
video_capture = cv2.VideoCapture(0)
```

```
while True:  
    ret, frame = video_capture.read()  
    if not ret:  
        break  
    gray_frame = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)  
    faces = face_classifier.detectMultiScale(  
        gray_frame,  
        scaleFactor=1.1,  
        minNeighbors=5,  
        minSize=(40, 40))  
    for (x, y, w, h) in faces:  
        cv2.rectangle(frame, (x, y), (x + w, y + h), (255, 0, 0), 2)  
    cv2.imshow("Video", frame)  
    # Quitter sur la touche 'q'  
    if cv2.waitKey(1) & 0xFF == ord('q'):  
        break  
  
video_capture.release()  
cv2.destroyAllWindows()
```

```
cv2.imshow("Video", frame)
```



État d'avancement

Partie Détection de visage

- Utilisation du DataSet

```
data_dir    = "C:/Users/Utilisateur/Desktop/Data"  
images_dir = os.path.join(data_dir, "images/train")  
labels_dir = os.path.join(data_dir, "labels/train")
```

- Lecture des boîtes ground-truth au format YOLO

```
label_path = os.path.join(labels_dir,  
                           os.path.splitext(filename)[0] + ".txt")  
  
gt_boxes = []  
if os.path.exists(label_path):  
    with open(label_path, "r") as f:  
        for line in f:  
            cls, xc, yc, w, h = map(float, line.split())  
            xc *= w_img; yc *= h_img  
            w *= w_img; h *= h_img  
            x = xc - w/2; y = yc - h/2  
            gt_boxes.append((x, y, w, h))  
  
else:  
    print(f"{filename}: pas de label")  
    continue
```

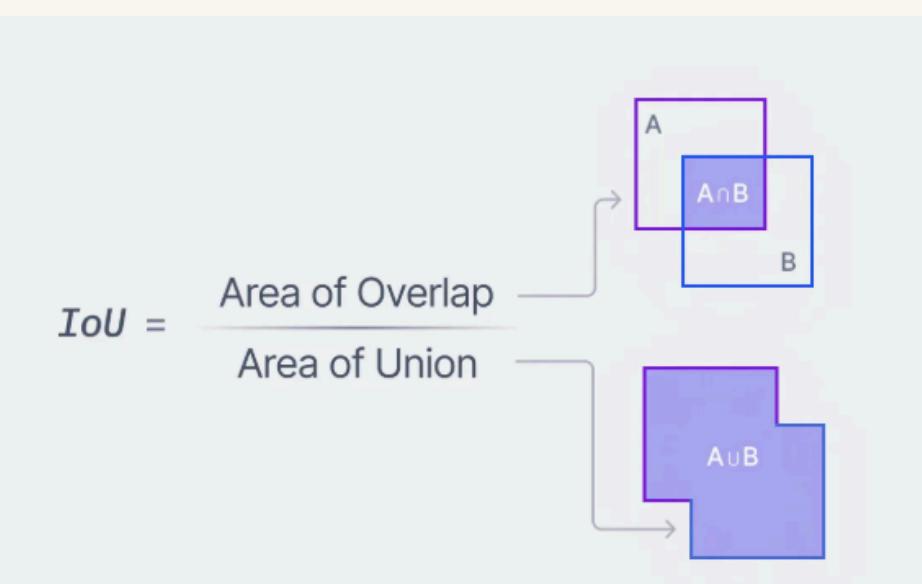


- Affichage boîte englobante sur les images du dataset:

- Haar Cascade
- Ground Truth

1 3
2 4

- Comparaison des boites englobantes à l'aide d'une fonction IOU

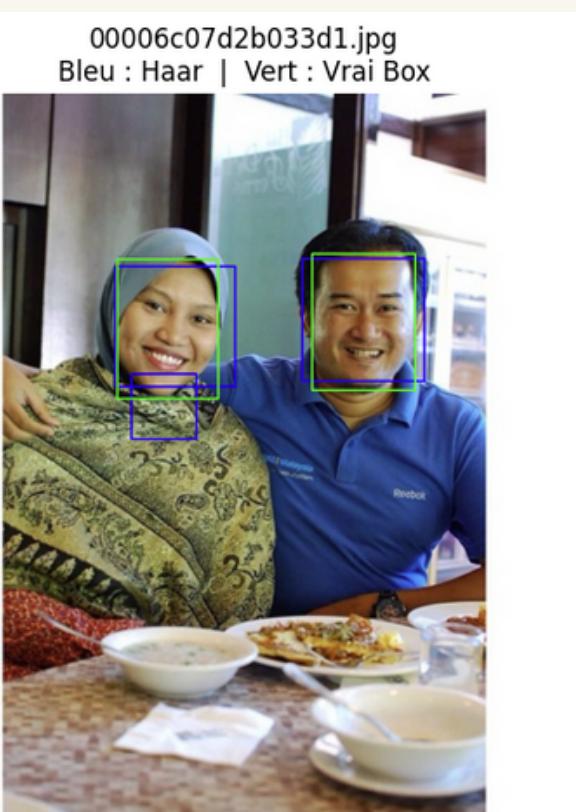


État d'avancement

Métriques d'analyse

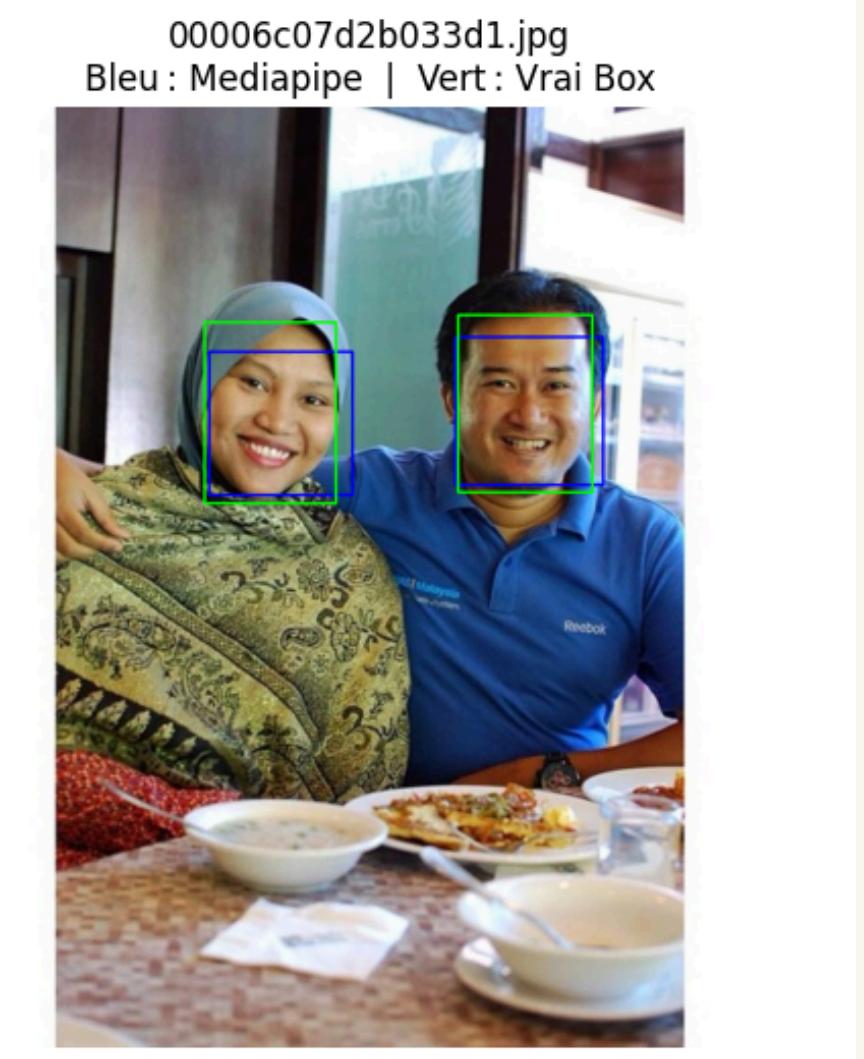
- Résultat cascade Haar

```
IoU moyen (seuil 0.5) : 0.729  
Précision : 0.583  
Rappel : 0.538  
F1-score : 0.560
```



- Résultat Mediapipe

```
MediaPipe –  
Précision: 1.000  
Rappel: 0.615  
F1: 0.762  
IoU moyen: 0.723
```



Tests de différent modèle de détection de visage (dlib, DNN SSD)

État d'avancement

Partie Reconnaissance de visage

- Application ACP

```
n_components = 150
pca = PCA(n_components=n_components, svd_solver='randomized', whiten=True)
pca.fit(X_train)

eigenfaces = pca.components_.reshape((n_components, images.shape[1], images.shape[2]))
print(f"Variabilité expliquée par {n_components} composantes : "
      f"{np.sum(pca.explained_variance_ratio_):.2%}")
✓ 0.1s

Variabilité expliquée par 150 composantes : 94.61%
```

```
X_train_pca = pca.transform(X_train)
X_test_pca = pca.transform(X_test)

svc = SVC(kernel='rbf', class_weight='balanced', C=10, gamma=0.001, random_state=42)
svc.fit(X_train_pca, y_train)
```

Rapport de Classification:				
	precision	recall	f1-score	support
Ariel Sharon	0.71	0.79	0.75	19
Colin Powell	0.78	0.86	0.82	59
Donald Rumsfeld	0.78	0.70	0.74	30
George W Bush	0.90	0.92	0.91	133
Gerhard Schroeder	0.75	0.67	0.71	27
Hugo Chavez	0.91	0.56	0.69	18
Tony Blair	0.76	0.81	0.78	36
accuracy			0.83	322
macro avg	0.80	0.76	0.77	322
weighted avg	0.83	0.83	0.82	322

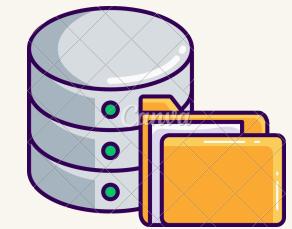
Utilisation de SVM pour interprétation et pour une comparaison prochaine

Prochaine Objectif ?

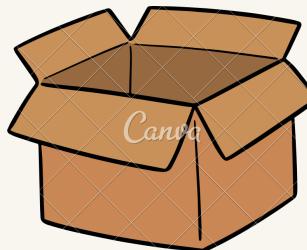
Partie Reconnaissance de visage

- Utilisation de HOG
- Comparaison via SVM de ACP / HOG
- Essaie de différents classifieurs (DecisionTree / Random Forest)
- Application CNN ?

Difficultés rencontrées



Compréhension
du Data Set
Kaggle

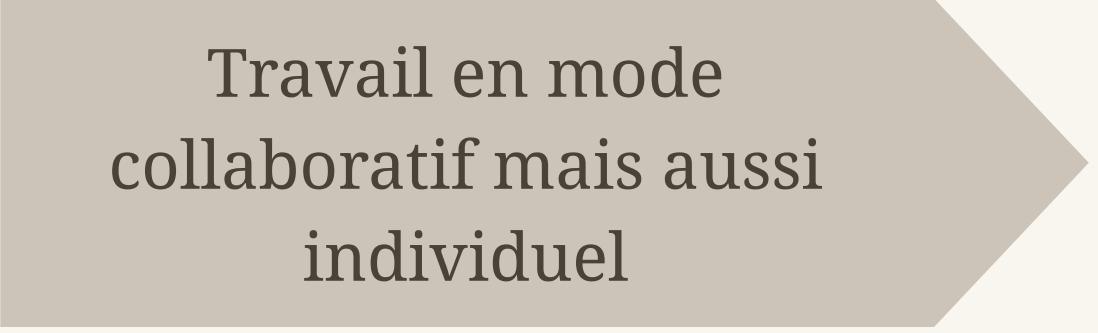


Harmonisation
des formats de
boîtes
englobantes

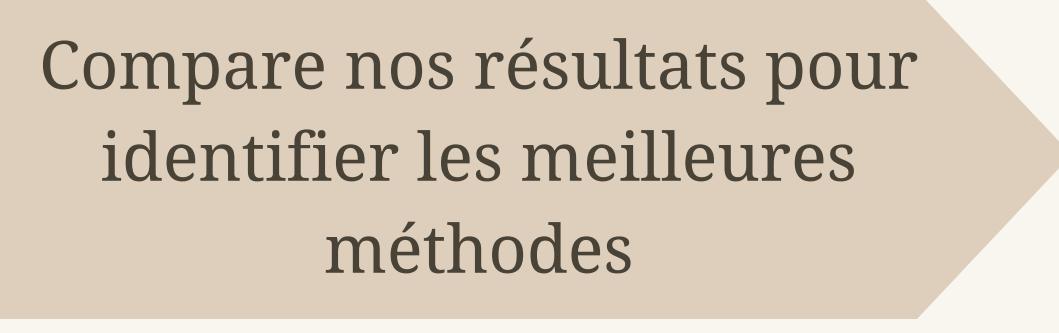


Interprétation des
métriques

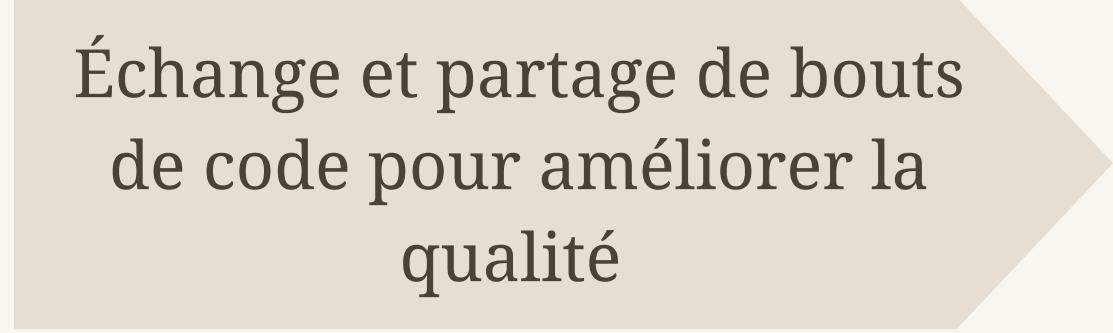
Repartitions des tâches



Travail en mode
collaboratif mais aussi
individuel



Compare nos résultats pour
identifier les meilleures
méthodes



Échange et partage de bouts
de code pour améliorer la
qualité