



## Rapport du Projet :

Détection de visages en utilisant le Transfer Learning  
avec MediaPipe model maker et MobileNet\_V2

**Matière :** Deep Learning

**Professeur :** Dounia Awad

**Etudiant :** Es-Sadany Yassine

**Groupe :** ING3 IA 2

06 Décembre 2023

# Contents

|          |  |           |
|----------|--|-----------|
| <b>1</b> | <b>Introduction</b>                            | <b>2</b>  |
| 1.1      | Contexte . . . . .                             | 2         |
| 1.2      | Objectif du projet . . . . .                   | 2         |
| 1.3      | Structure du rapport . . . . .                 | 2         |
| <b>2</b> | <b>Méthodologie</b>                            | <b>2</b>  |
| 2.1      | Transfer Learning . . . . .                    | 2         |
| 2.2      | Outils et technologies utilisés . . . . .      | 3         |
| 2.2.1    | MediaPipe . . . . .                            | 3         |
| 2.2.2    | MobileNet . . . . .                            | 4         |
| <b>3</b> | <b>Mise en œuvre</b>                           | <b>5</b>  |
| 3.1      | Entraînement du modèle . . . . .               | 5         |
| 3.1.1    | Préparation des données . . . . .              | 5         |
| 3.1.2    | Configuration de l'environnement . . . . .     | 7         |
| 3.1.3    | Processus d'entraînement . . . . .             | 7         |
| <b>4</b> | <b>Tests et résultats</b>                      | <b>7</b>  |
| 4.1      | Evaluation des performance du modèle . . . . . | 7         |
| 4.2      | Procédure de test . . . . .                    | 8         |
| 4.3      | Résultats . . . . .                            | 8         |
| 4.4      | Discussion . . . . .                           | 14        |
| <b>5</b> | <b>Conclusion</b>                              | <b>14</b> |

# 1 Introduction

## 1.1 Contexte

La détection de visages est un domaine clé de la vision par ordinateur, avec des applications allant de la sécurité aux interfaces utilisateurs. L'approche moderne utilise le deep learning pour améliorer la précision et l'efficacité.

## 1.2 Objectif du projet

Ce projet vise à développer un modèle de détection de visages en utilisant le transfer learning avec MediaPipe model maker avec le modèle préentraînée MobileNet\_V2, exploitant les capacités de traitement du GPU sur Google Colab.

## 1.3 Structure du rapport

Ce rapport détaille les étapes de développement, de la configuration de l'environnement Colab à l'entraînement et aux tests du modèle.

# 2 Méthodologie

## 2.1 Transfer Learning

Le transfer learning implique l'utilisation de modèles pré-entraînés et leur adaptation à de nouvelles tâches. Cette méthode permet d'économiser des ressources et du temps, et améliore la performance sur des ensembles de données de petite taille.



Figure 1: Langage d'execution

## 2.2 Outils et technologies utilisés

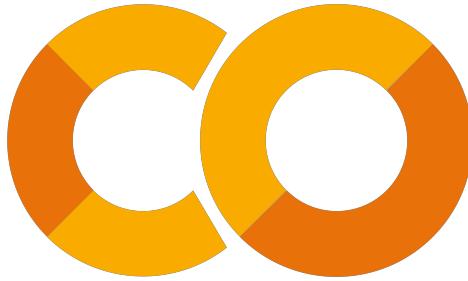


Figure 2: Environnement d'execuition

### 2.2.1 MediaPipe

MediaPipe est un framework de vision par ordinateur proposant des solutions pré-construites pour des tâches comme la détection de visages. Il est reconnu pour sa facilité d'utilisation et sa performance.

l'utilisation de mediapipe\_model\_maker pour entraîner un modèle de détection d'objet est un exemple de transfert d'apprentissage (transfer learning). Le

transfert d'apprentissage est une technique courante en apprentissage automatique où un modèle pré-entraîné sur une tâche spécifique est utilisé comme point de départ pour entraîner un nouveau modèle sur une tâche similaire ou apparentée.

Dans le cas de mediapipe\_model\_maker, il utilise généralement un modèle pré-entraîné sur un grand ensemble de données pour extraire des caractéristiques génériques, puis il ajuste ces caractéristiques pour la tâche spécifique de détection d'objet sur nos propres données.

L'avantage du transfert d'apprentissage est qu'il permet d'exploiter les connaissances acquises par un modèle sur une tâche pour améliorer les performances sur une tâche similaire, même si les ensembles de données sont différents. Cela peut conduire à une convergence plus rapide et à de meilleures performances avec moins de données d'entraînement que nécessaire pour un apprentissage à partir de zéro.

Dans mon projet le modèle préentraînée choisi est **MobileNet\_V2** qui sera expliqué dans la partie suivante.



Figure 3: Logo de Mediapipe

### 2.2.2 MobileNet

MobileNet est une famille d'architectures de réseaux de neurones convolutifs conçue pour être légère et efficace en termes de calcul, ce qui les rend adaptés aux appareils avec des ressources limitées, tels que les smartphones et les systèmes embarqués. L'architecture MobileNetV2 est une version améliorée de MobileNet, optimisée pour des performances encore meilleures.

Dans le contexte de la détection d'objets, MobileNetV2 est souvent utilisé comme une architecture de base pour extraire des caractéristiques à partir

d'images. Ces caractéristiques extraites sont ensuite utilisées par un modèle plus complexe pour effectuer la tâche spécifique de détection d'objets (dans notre cas de visages).

## 3 Mise en œuvre

### 3.1 Entraînement du modèle

#### 3.1.1 Préparation des données

Pour l'entraînement du modèle, un dataset spécifique à la détection de visages a été utilisé, provenant de RobotFlow. Ce dataset est formaté selon le standard COCO (**C**ommon **O**bjects **i**n **C**ontext) JSON, qui est couramment utilisé pour les tâches de détection d'objets. Le format COCO (Common Objects in Context) structure les données en un fichier JSON contenant des informations telles que les emplacements des objets (définis par des bounding boxes) et les catégories d'objets dans les images. Le dataset est structuré comme la figure 4 montre :

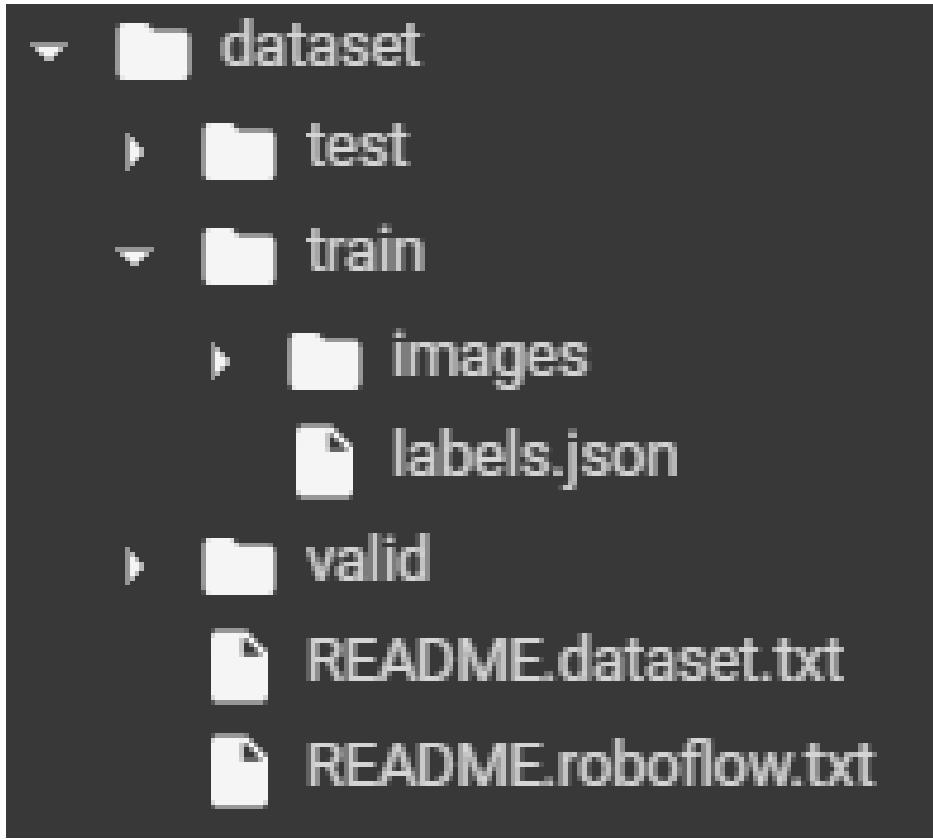


Figure 4: Format COCO

où labels.json est formaté comme suit :

```
{
  "categories": [
    {"id":1, "name":<nom_cat>},
    ...
  ],
  "images": [
    {"id":0, "file_name":<img0>.jpg},
    ...
  ],
  "annotations": [
    {"id":0, "image_id":0, "category_id":1, "bbox": [x-coordonnée en haut à gauche, y-coordonnée en haut à gauche, largeur, hauteur]},
    ...
  ]
}
```

Figure 5: Format labels.json



Figure 6: Source du dataset utilisé : <https://universe.roboflow.com/face-detection-and-recognition-dataset/person-faces>

### 3.1.2 Configuration de l'environnement

Google Colab est utilisé pour son accès facile aux ressources de calcul, notamment aux GPU. Les scripts Python sont exécutés dans des notebooks Colab, où le GPU est activé via les paramètres de runtime.

### 3.1.3 Processus d'entraînement

Le modèle MobileNet est entraîné sur les données de visages du dataset RobotFlow en utilisant le transfer learning. Les paramètres du modèle sont ajustés pour optimiser les performances, en tenant compte des limitations de la mémoire GPU disponible sur Colab. L'entraînement est réalisé en utilisant le format COCO pour exploiter efficacement les annotations et les métadonnées fournies avec le dataset.

## 4 Tests et résultats

### 4.1 Evaluation des performance du modèle

à partir du notebook `face_detector.ipynb` et après avoir entrainée le modèle, on évalue sa performance et on obtient les résultats suivants :

```

loss, coco_metrics = model.evaluate(validation_data, batch_size=20)
print("validation coco metrics: (coco_metrics)")

3/25 [=====] - 2s/step - total_loss: 0.4403 - cls_loss: 0.2306 - box_loss: 0.0032 - model_loss: 0.3884
  creating index...
  creating index...
  nodes created
  number of image evaluation...
  Evaluation type: "bbox"
  (25, 50)
  Accumulating evaluation results...
  (25, 50)
  Average Precision (AP) @ (IoU=0.50-0.95 | area= all | maxDet=100 ) = 0.500
  Average Precision (AP) @ (IoU=0.50-0.95 | area= small | maxDet=100 ) = 0.499
  Average Precision (AP) @ (IoU=0.50-0.95 | area= medium | maxDet=100 ) = 0.570
  Average Precision (AP) @ (IoU=0.50-0.95 | area= large | maxDet=100 ) = 0.441
  Average Precision (AP) @ (IoU=0.50-0.95 | area=medium | maxDet=100 ) = 0.463
  Average Precision (AP) @ (IoU=0.50-0.95 | area=large | maxDet=100 ) = 0.396
  Average Recall (AR) @ (IoU=0.50-0.95 | area= all | maxDet= 10 ) = 0.548
  Average Recall (AR) @ (IoU=0.50-0.95 | area= small | maxDet= 10 ) = 0.525
  Average Recall (AR) @ (IoU=0.50-0.95 | area= medium | maxDet= 10 ) = 0.723
  Average Recall (AR) @ (IoU=0.50-0.95 | area= large | maxDet= 10 ) = 0.059
  validation coco metrics: {'AP': 0.500, 'AP50': 0.499, 'AP95': 0.570, 'AR1': 0.548, 'AR10': 0.525, 'AR100': 0.723, 'ARmax100': 0.059}
Validation coco metrics: {'AP': 0.500, 'AP50': 0.499, 'AP95': 0.570, 'AR1': 0.548, 'AR10': 0.525, 'AR100': 0.723, 'ARmax100': 0.059}

```

Figure 7: Evaluation des performances du modèle entraîné

## 4.2 Procédure de test

Le test du modèle a été fait dans le notebook nommé Test\_model.ipynb, où on a testé un ensemble des images contenus dans un dossier localisé dans le drive, tel que j'ai testé sur deux types d'images, d'abord celles dans le drive et puis y a une section du code qui permet de capturer l'image du web caméra en temps réel et détecter les visages. Enfin la dernière ligne du code c'est pour tester le modèle en local par exemple sur PyCharm, il suffit d'importer le modèle téléchargé **model.tflite** avec ce code et ça va lancer la caméra du pc avec la détection en temps réel des visages sur la caméra. En plus de la détection j'ai rajouté la condition du score (score  $\geq$  score\_max-3 pour chaque détection valide) ou la probabilité de détection pour vérifier s'il s'agit bien de visage, car le modèle détecte quelques objets qui ne sont pas des visages mais avec un score faible.

## 4.3 Résultats

Les résultats montrent une haute précision de détection avec un temps de réponse acceptable, indiquant l'efficacité du modèle dans des conditions réelles.

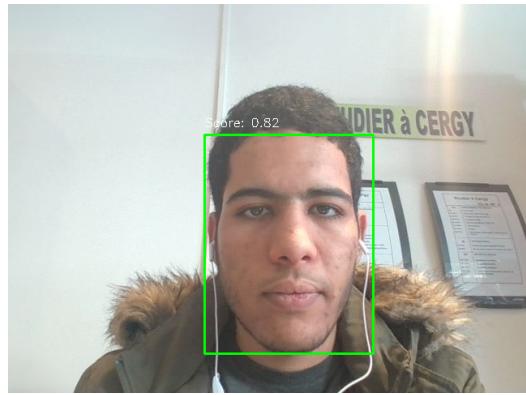


Figure 8: Test du modèle avec la web cam

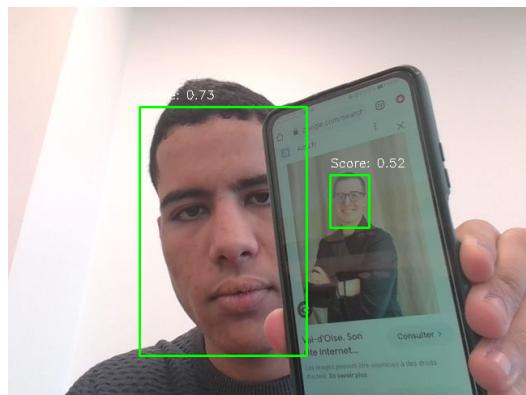


Figure 9: Test du modèle avec la web cam

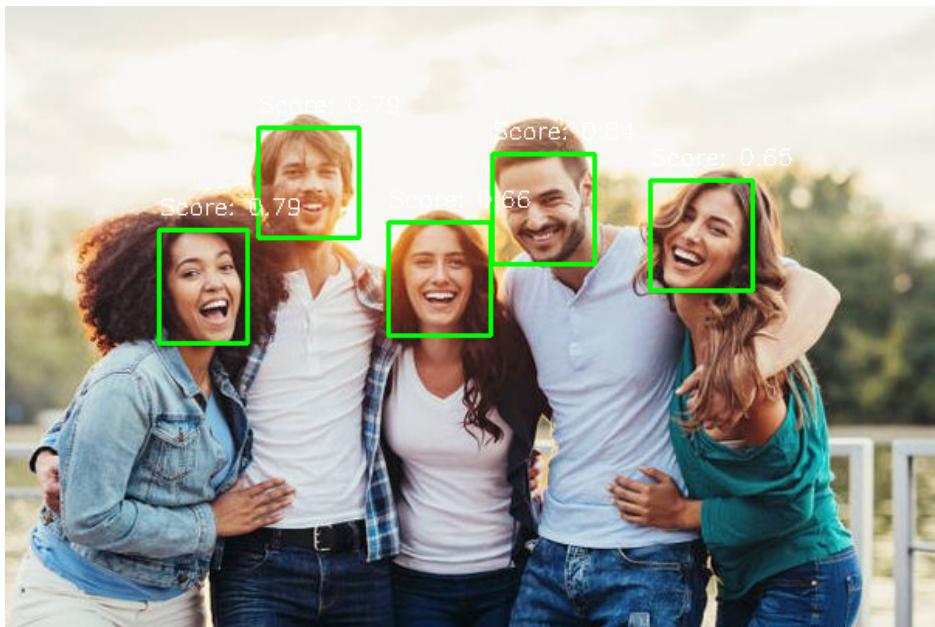


Figure 10: Test du modèle sur une image en local



Figure 11: Test du modèle sur une image en local

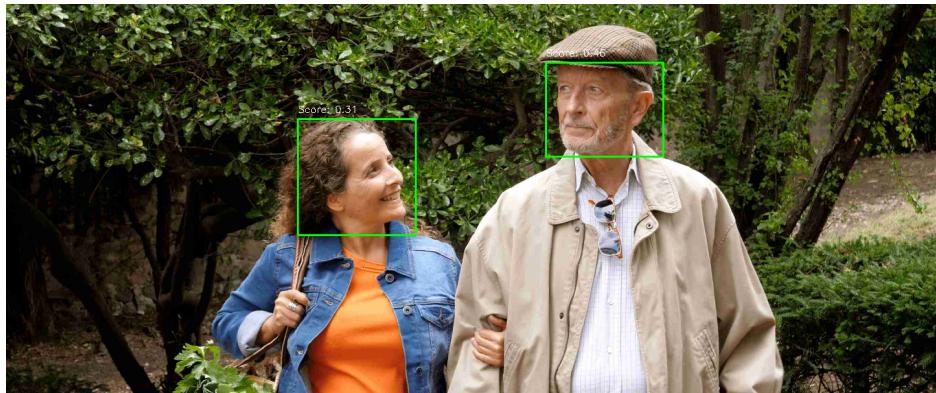


Figure 12: Test du modèle sur une image en local

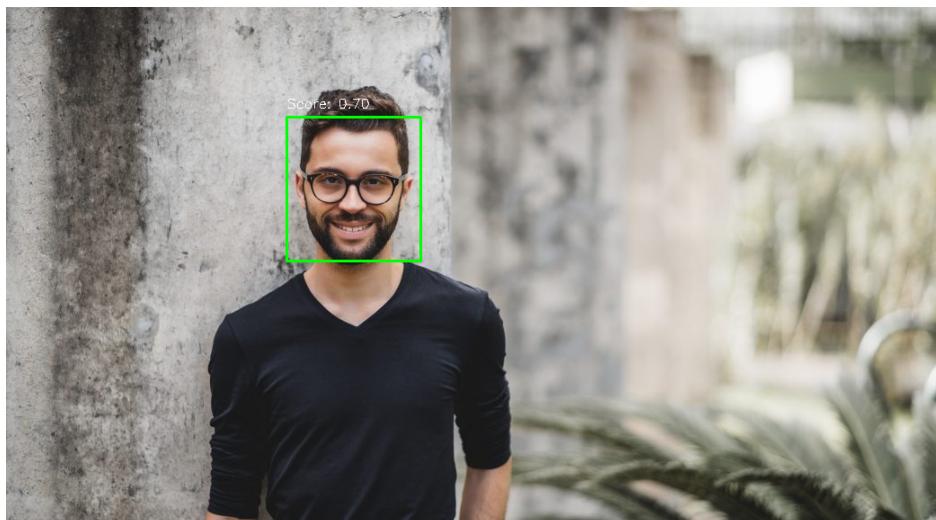


Figure 13: Test du modèle sur une image en local

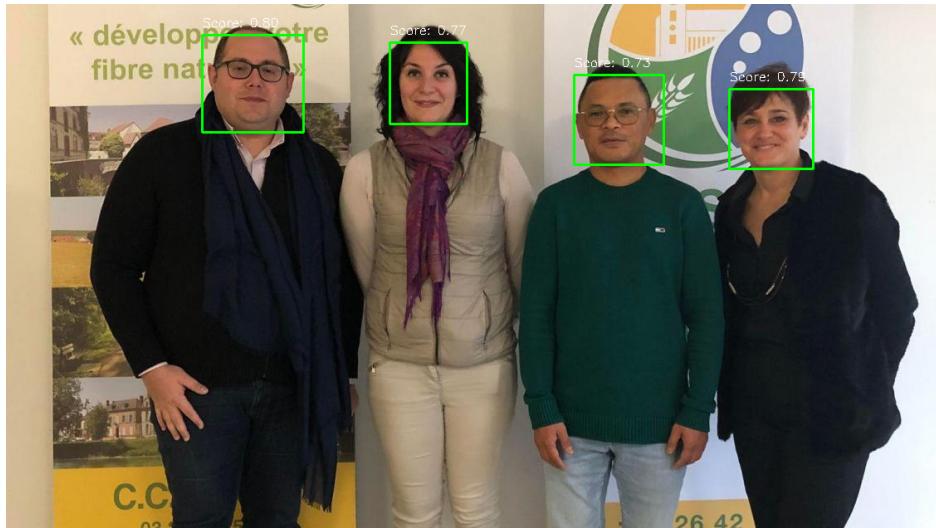


Figure 14: Test du modèle sur une image en local



Figure 15: Test du modèle sur une image en local



Figure 16: Test du modèle sur une image en local

#### 4.4 Discussion

Les résultats montre que la détection de visage sur la caméra en temps réel est plus performant que sur des images importé du local, parce que des fois il peut ne pas détecter quelques visages si l'images contient plus de 5 visages mais je pense que ça et dû au données du dataset qui ne couvre pas les images ou y a plusieurs visages, et aussi des fois il peut détecter des objets qui ne sont pas des visages mais avec une probabilité faible.

Après des recherches j'ai trouvé que la taille des images du dataset est très important et que toutes les images doivent avoir la même taille. et C'est pour cela la performance du modèle a était affecté.

### 5 Conclusion

Ce projet démontre l'efficacité du transfer learning en combinant MediaPipe et MobileNet pour la détection de visages. Et a permis de savoir que l'ensemble de données du dataset peut affecté les performance du modèle.