**ML5 \_ image recognition**

**0 \_ DATASET**

* **Préparer l’arborescence suivantes**

|  |
| --- |
| /dataset/  ├── /train/  │ ├── target/ (800 images)  │ ├── others/ (800 images, optionnel)  ├── /test/  │ ├── target/ (200 images)  │ ├── others/ (200 images, optionnel) |

* **Préparer les images**

Pour un bon entraînement, toutes les images doivent avoir :

✔ Le même format (ex: JPG ou PNG)

✔ La même taille (ex: 224x224 px, adapté à MobileNet)

✔ Une normalisation des pixels (valeurs entre 0 et 1) (fait par imageNet)

Programme python pour préparer les images

|  |
| --- |
| import cv2  import os  import numpy as np  # Dossiers des images  input\_dirs = ["dataset/train/target", "dataset/train/others",  "dataset/test/target", "dataset/test/others"]  output\_size = 224 # Taille finale  def resize\_with\_padding(img, size):  h, w = img.shape[:2]  # Calcul du ratio pour garder les proportions  scale = size / max(h, w)  new\_w, new\_h = int(w \* scale), int(h \* scale)  # Redimensionnement sans déformation  resized = cv2.resize(img, (new\_w, new\_h), interpolation=cv2.INTER\_AREA)  # Création d'un fond noir (ou blanc : remplacer 0 par 255)  delta\_w = size - new\_w  delta\_h = size - new\_h  top, bottom = delta\_h // 2, delta\_h - (delta\_h // 2)  left, right = delta\_w // 2, delta\_w - (delta\_w // 2)  padded = cv2.copyMakeBorder(resized, top, bottom, left, right, cv2.BORDER\_CONSTANT, value=(0, 0, 0)) # Noir  return padded  # Parcourir les dossiers et appliquer le padding  for input\_dir in input\_dirs:  for filename in os.listdir(input\_dir):  img\_path = os.path.join(input\_dir, filename)  img = cv2.imread(img\_path)  if img is None:  continue  img\_padded = resize\_with\_padding(img, output\_size)  # Sauvegarde (remplace l’original)  cv2.imwrite(img\_path, img\_padded)  print("Redimensionnement terminé sans distorsion ! ✅") |

**1\_ Entrainement du modèle**

On utilise MobileNet et ml5.NeuralNetwork pour entraîner un modèle simple de reconnaissance faciale.

Étapes de l'entraînement

1. Charger et préparer les images dans ml5.js
2. Utiliser MobileNet pour extraire les features
3. Entraîner un modèle personnalisé sur tes images
4. Sauvegarder le modèle pour une utilisation avec la webcam

Hébergement local des images nécessite un server : node.js ou python

En python : python -m http.server 8000 puis http://localhost:8000/ dans le navigateur

3 fichiers générés :

* model.json : Architecture du modèle (structure, type de réseau).
* model.weights.bin : Poids appris du modèle.
* model\_meta.json : Métadonnées et informations sur le modèle.

**2\_ Tester le modèle**

Résumé des étapes pour tester un modèle

* Charger le modèle entraîné avec ml5.NeuralNetwork.load() ou tf.loadLayersModel().
* Tester une image avec la méthode classify de ml5.js ou predict avec TensorFlow.js.
* Analyser les résultats (vérification de la classe prédite vs la classe réelle).
* Tester sur plusieurs images et calculer la précision globale du modèle.
* Visualiser les résultats sous forme de graphiques pour mieux comprendre la performance.