SY32 – TD Apprentissage 04 : Évaluation d'un détecteur

Exercice 1 : Évaluation d'un détecteur

Il s'agit d'évaluer les performances d'un détecteur de visages avec une base de données de 1000 images dont toutes les annotations sont stockées dans le fichier `label_train.txt`. Le détecteur a été entraîné avec les images `0001.jpg` à `0500.jpg`. Le fichier `results_train_500.txt` contient les résultats du détecteur sur les images `0501.jpg` à `1000.jpg` de la base de données. Le fichier `label_train.txt` des vérités contient cinq valeurs $k,\ i,\ j,\ h,\ l$ par ligne, et le fichier `results_train_500.txt` des prédictions contient six valeurs $k,\ i,\ j,\ h,\ l,\ s$ par ligne :

- k : numéro de l'image
- (i, j) : coordonnées (ligne c-à-d selon l'axe vertical, colonne c-à-d selon l'axe horizontal) du coin supérieur gauche de la boîte
- (h, l) : taille (hauteur, largeur) de la boîte
- s : score de détection

La lecture des données pour l'exercice peut se faire comme suit :

```
import numpy as np

results_train = np.loadtxt('results_train_500.txt')
label_train = np.loadtxt('label_train.txt', skiprows=658) # lire a partir de l'img 501
```

1 . Écrire une fonction pour tracer les boîtes d'une image sur fond blanc. Cela servira à visualiser le bon fonctionnement du filtrage des boîtes de la question 4. (Vous pouvez considérer que vos images font 450×450 pixels.)

2 . Écrire une fonction `IoU` qui calcule l'aire de recouvrement entre deux boîtes, chacune des boîtes b étant représentée par ses coordonnées (i, j, h, l).

3 . Valider la fonction `IoU` avec des données d'entrée typiques dont le résultat attendu est connu, par exemple :

- Soit $b_1 = b_2 = (0, 0, 10, 10) \Rightarrow$ `IoU` $= 1$
- Soit $b_1 = (0, 0, 10, 10)$ et $b_2 = (20, 20, 10, 10) \Rightarrow$ `IoU` $= 0$
- Soit $b_1 = (0, 0, 10, 10)$ et $b_2 = (0, 5, 10, 10) \Rightarrow$ `IoU` $= 0.33...$ (l'intersection vaut $5 \times 10 = 50$ et l'union vaut $15 \times 10 = 150$)
- Soit $b_1 = (21, 4, 140, 10)$ et $b_2 = (30, 12, 15, 12) \Rightarrow$ `IoU` $= 0.01935...$

4 . Pour chaque image, supprimer des prédictions les doublons (boîtes englobantes ayant une aire de recouvrement supérieure à 0.5) en ne gardant que les boîtes de score maximal (technique NMS : Non Maximum Suppression). En principe, cette procédure est appliquée comme post-traitement au détecteur. Le NMS peut être implémenté comme suit :

```
def filtre_nms(results_in, tresh_iou = 0.5):
    # role : si chevauchement trop fort entre deux detections, garder celle dont la confiance est
        maximale
    results_out = np.empty((0,6)) # initialiser un tableau de sortie vide
    unique_ids = np.unique(results_in[:,0])
    for i in unique_ids: # image par image
        results_in_i = results_in[results_in[:,0] == i]
        # trier les boites par score de confiance decroissant
        results_in_i = results_in_i[ results_in_i[:,5].argsort()[::-1] ]

        # liste des boites que l'on garde pour cette image a l'issue du NMS
        results_out_i = np.empty((0,6))
        # on garde forcement la premiere boite, la plus sure
        results_out_i = np.vstack((results_out_i, results_in_i[0]))

        # pour toutes les boites suivantes, les comparer a celles que l'on garde
        for n in range(1,len(results_in_i)):
            for m in range(len(results_out_i)):
                if iou(results_in_i[n,1:5], results_out_i[m,1:5]) > tresh_iou:
                    # recouvrement important,
                    # et la boite de results_out_i a forcement un score plus haut
                    break
                elif m == len(results_out_i)-1:
```

```
                # c'etait le dernier test pour verifier si cette detection est a conserver
                results_out_i = np.vstack((results_out_i, results_in_i[n]))

        # ajouter les boites de cette image a la liste de toutes les boites
        results_out = np.vstack((results_out,results_out_i))

    return results_out
```

5 . Pour chaque image, déterminer quelles sont les boîtes prédites qui correspondent effectivement à un visage et celles qui ne correspondent pas à un visage. On dira qu'une boîte est un VP (vrai positif) s'il existe une boîte dans le fichier des vérités `label_train.txt` avec laquelle l'aire de recouvrement est supérieure à 0.5.

Attention : on prendra garde de ne pas compter plusieurs fois un même visage détecté par plusieurs boîtes. Seule l'une d'elles compte comme VP, les autres comptent comme FP (faux positif).

6 . Pour chaque image, déterminer tous les visages qui ont été détectés et ceux qui ne l'ont pas été.

7 . Calculer la précision, le rappel, et le score F_1 (moyenne harmonique de la précision et du rappel) en considérant toutes les détections.

8 . Tracer la courbe de précision/rappel.

9 . Calculer l'aire sous la courbe de précision/rappel.

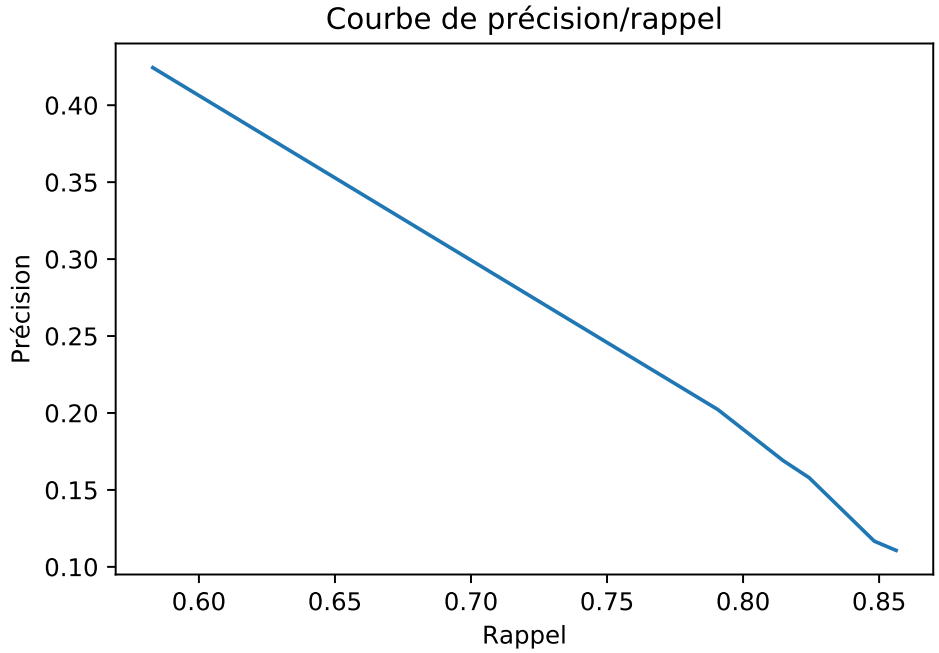

FIGURE 1 – Courbe de précision/rappel.

Note : Étude sur les différences entre les courbes ROC et précision/rappel : `https://ichi.pro/fr/sur-les-courbes-roc-et-precision-recall-213574150750732`

Liste de fonctions utiles :

- `numpy.loadtxt` avec argument `skiprows`
- `numpy.unique`
- `numpy.tolist`
- `numpy.argsort`
- `numpy.empty`
- `numpy.vstack`
- `numpy.concatenate`
- `numpy.logical_and`
- `numpy.zeros`
- `min`
- `max`
- `sort`
- `matplotlib.pyplot.plt`
- `matplotlib.patches.Rectangle`
- `matplotlib.collections.PatchCollection`
- opérateur `~`
- `numpy.count_nonzero`