TAKAHASHI Vincent
RICHARD Valentin
Date du TP : 19/11/2021

Rapport de TP2 – Lecture automatique de chiffres par analyse d'image

I. Introduction

Le but de ce TP est de reconnaître un chiffre dans une image.

II. Prise en main de l'environnement

III. Travail préparatoire

1. Question (1)

Nous avons pris connaissance du la classe Image. Nous avons relevé que nous allons devoir utiliser impérativement numpy pour travailler avec cette classe.

2. Question (2)

Nous avons complété la fonction Binarisation. Pour cela nous avons créé une nouvelle image, de taille identique à l'image donnée en paramètre et toute noire. Ensuite nous parcourons l'image modèle pour comparer chaque pixel avec le seuil donné en paramètre. En fonction de la comparaison, nous changeons, (ou pas,) la valeur du pixel dans l'image crée (sachant que nous n'affectons que deux valeurs aux pixels de notre image : 0 ou 255).

Ainsi nous obtenons notre nouvelle image.

Nous avons effectué les tests, aucun ne concernant cette fonction ne s'est révélé négatif.

```
#=========================================================================
# Methode de binarisation
# 2 parametres :
#   self : l'image a binariser
#   S : le seuil de binarisation
#   on retourne une nouvelle image binarisee
#=========================================================================
def binarisation(self, S):
    im_bin = Image()
    im_bin.set_pixels(np.zeros((self.H, self.W), dtype = np.uint8))

    for i in range (self.H):
        for j in range (self.W):
            if self.pixels[i][j] >= S:
                im_bin.pixels[i][j] = 255
    return im_bin
```

3. Question (3)

Nous avons complété la fonction Localisation. Pour cela on place les curseurs aux extrémités de l'image. En parcourant l'image en ligne, on détecte les pixels noirs et on rapproche les curseurs au fur et à mesure.

Après cela, on crée une nouvelle image tout noire, que nous dimensionnons avec les curseurs placés précédemment. On parcourt à la fois l'image modèle et la nouvelle image pour copier la partie de l'image modèle qui nous intéresse.

Il ne reste plus qu'à retourner notre nouvelle image.

```python
def localisation(self):
    lmax = 0
    lmin = self.H
    cmax = 0
    cmin = self.W

    for i in range (self.H):
        for j in range (self.W):
            if self.pixels[i][j] == 0:
                if j < cmin:
                    cmin = j
                elif j > cmax :
                    cmax = j
                if i < lmin:
                    lmin = i
                elif i > lmax:
                    lmax = i
    h = lmax - lmin
    w = cmax - cmin

    im_red = Image()

    im_red.set_pixels(np.zeros((h, w), dtype = np.uint8))
    for i in range (h):
        for j in range (w):

            if self.pixels[lmin+i, cmin + j] == 255:
                im_red.pixels[i][j] = 255
            elif self.pixels[lmin +i, cmin+j] == 0:
                im_red.pixels[i][j] = 0

    return im_red
```

IV. Reconnaissance automatique de chiffres

1. Question (1)

Nous avons lancé les tests, aucun ne s'est révélé négatif.

2. Question (2)

Même chose, tous les tests sont bons, après quelques corrections.

3. Question (3)

On a implémenté la fonction resize. On a créé une nouvelle image. On utilise ensuite la fonction resize donnée dans le sujet pour remplir cette image selon la taille voulue.

Nous avons utilisé la conversion donnée dans le sujet (cela nous a valu une erreur, qui ont été corrigé).

```python
#=============================================
# Methode de redimensionnement d'image
#=============================================
def resize(self, new_H, new_W):
    res_im = Image()
    new = resize(self.pixels, (new_H, new_W), 0)
    res_im.set_pixels(np.uint8(new*255))
    return res_im
```

4. Question (4)

Nous avons ajouté la méthode similitude. On parcourt l'image en comptant le nombre de pixels identiques puis on fait le ratio sur le nombre de pixel total.

```
#=============================================================================
# Methode de mesure de similitude entre l'image self et un modele im
#=============================================================================
def similitude(self, im):
    n = 0
    for i in range (self.H):
        for j in range (self.W):
            if self.pixels[i,j] == im.pixels[i,j]:
                n = n+1

    return n/(self.H*self.W)
```

5. Question (5)

Nous avons écrit la fonction reconnaissance_chiffre. On commence par binariser et localiser notre image grâce à nos fonctions précédentes. Puis on initialise deux variables pour garder la mémoire la proportion maximale et l'indice de l'image ayant la proportion maximale.

On fait ensuite une boucle pour comparer la similitude avec chaque image modèle. Si une image correspond mieux que toutes les précédentes, on l'enregistre dans nos variables.

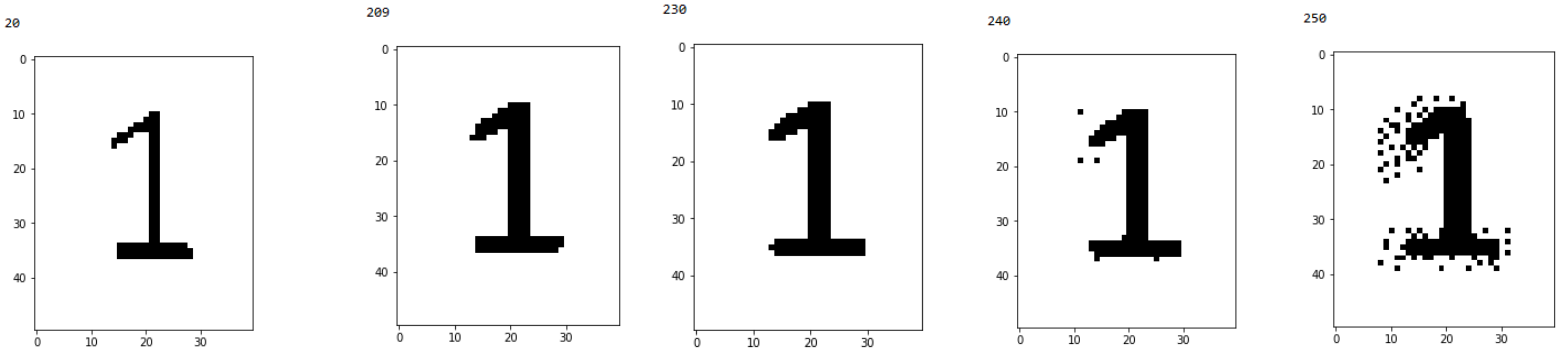
On renvoie enfin l'indice (qui est aussi le chiffre en question) de l'image qui a obtenu la meilleure similitude avec la nôtre.

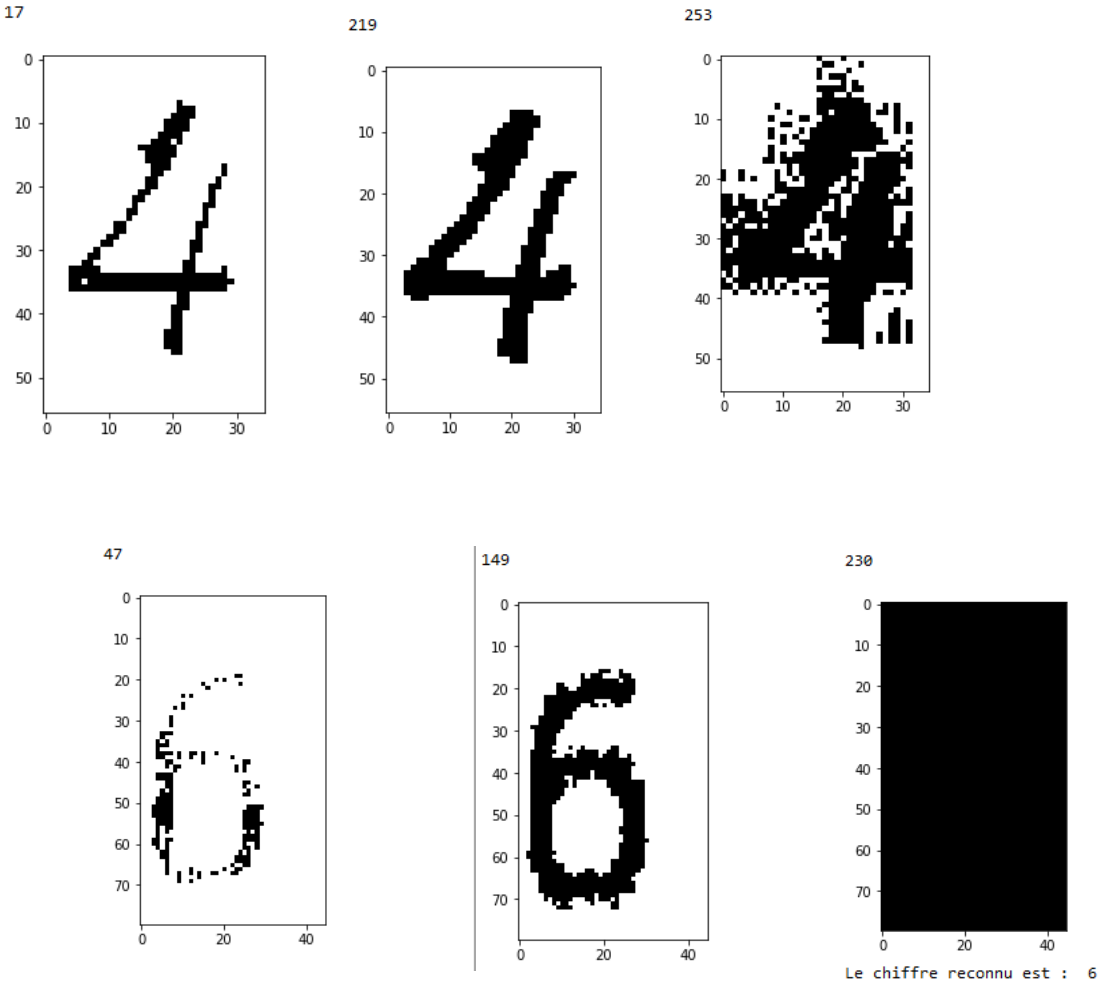
```
def reconnaissance_chiffre(image, liste_modeles, S):
    bin_im = image.binarisation(S)
    loc_im = bin_im.localisation()

    prop = 0
    i_prop = 0

    for i in range (len(liste_modeles)):

        modele_0 = liste_modeles[i]
        W = modele_0.W
        H = modele_0.H
        res = loc_im.resize(H,W)

        if res.similitude(liste_modeles[i])> prop:
            prop = res.similitude( liste_modeles[i])
            i_prop = i

    return i_prop
```

6. Question (6)

Nous avons fait des tests avec plusieurs numéro.

Voici ce que nous avons obtenu :





Le numéro affiché au-dessus de chaque image correspond au seuil.
Le seuil optimal se situe aux alentours des 200, d'après nos observations, même s'il ne peut correspondre à toutes les images. (Exemple avec le 6 qui est sur fond gris, on obtient une image noire si on prend un seuil trop élevé).

V. Conclusion

Nous avons terminé le TP dans les temps, de plus, tous les tests ont étés passés avec succès.