

Rapport de TP2 – Reconnaissance de chiffre

I. Introduction

Ce TP a pour but la reconnaissance de chiffre à partir d'une photo. Pour les reconnaître il faudra filtrer la photo afin de l'améliorer, la redimensionner et la comparer avec une base de données.

II. Section III du TP

1. Question (1).

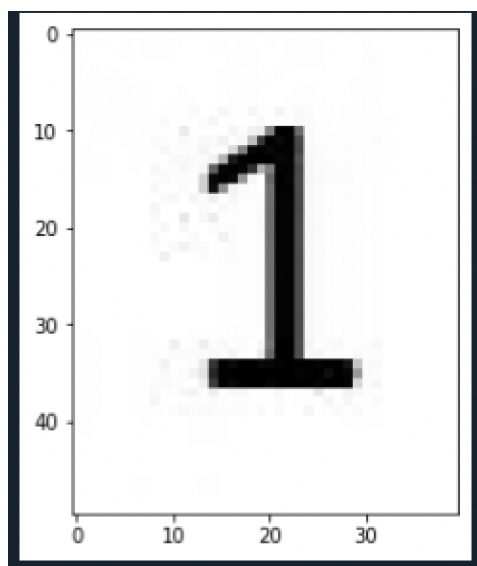
L'attribut H et W décrivent la résolution de l'image. L'attribut pixel correspond au tableau contenant les valeurs de chaque pixel avec H étant le nombre de lignes du tableau et W le nombre de colonnes. De plus, la position du pixel dans le tableau correspond à la position du pixel sur l'image.

2. Question (2).

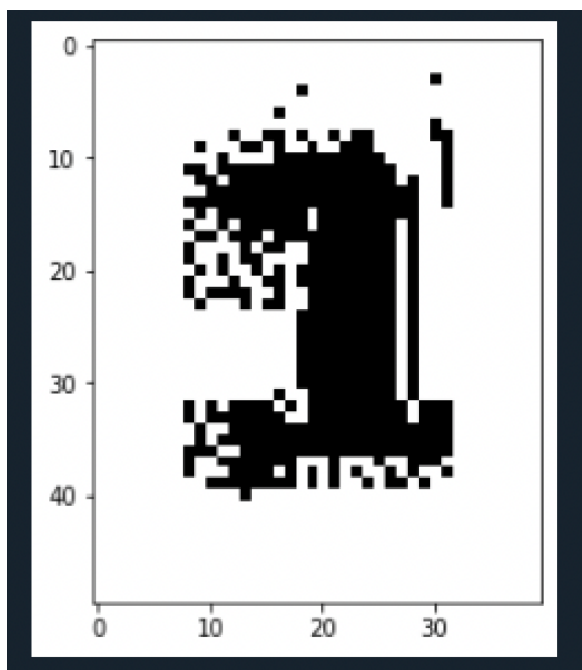
Nous avons créé une nouvelle image. A l'aide de deux boucles « for », nous avons parcouru toute l'image. Lorsque la valeur du pixel était inférieure à une valeur (S), on rentrait dans la nouvelle image, à la même position, la valeur « 0 ». Inversement si la valeur du pixel était supérieure à S, on rentrait « 255 ».

```
#=====
# 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 c in range(self.W):
        for l in range(self.H):
            if self.pixels[l][c] >= S:
                im_bin.pixels[l][c] = 255
            else:
                im_bin.pixels[l][c] = 0
    return(im_bin)
```

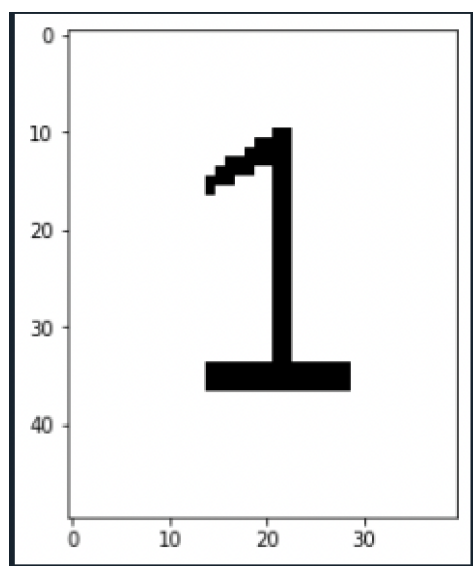
Avant binarisation :



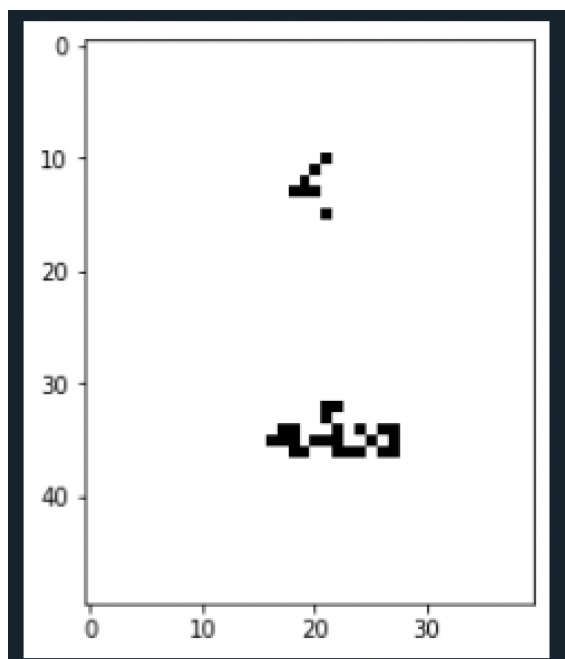
Avec $S=254$ (Dès que la case n'est plus blanche, il retourne une case noir) :



Après binarisation :



Avec $S=2$ (Dès que la case n'est pas tout à fait noir, il retourne une case blanche) :



3. Question (3)

Afin de réduire le tableau, nous l'avons parcouru à l'aide de « for ». Lorsque qu'un pixel est noir (il vaut 0), on agrandit soit la colonne soit la ligne de notre nouveau tableau. Enfin on remet les pixels noirs dans le nouveau tableau avec la bonne place. Cela nous donne le tableau le plus petit possible avec la même image. On a donc supprimé les bords blancs inutiles.

```
#####  
# Dans une image binaire contenant une forme noire sur un fond blanc  
# la methode 'localisation' permet de limiter l'image au rectangle englobant  
# la forme noire  
# 1 parametre :  
# self : l'image binaire que l'on veut recadrer  
# on retourne une nouvelle image recadree  
#####  
def localisation(self):  
    im_bin = self.binarisation(127)  
    c_min = im_bin.W  
    c_max = 0  
    l_min = im_bin.H  
    l_max = 0  
    for l in range(im_bin.H):  
        for c in range(im_bin.W):  
            if im_bin.pixels[l][c] == 0:  
                if c <= c_min:  
                    c_min = c  
                if c >= c_max:  
                    c_max = c  
                if l <= l_min:  
                    l_min = l  
                if l >= l_max:  
                    l_max = l  
    im_res = Image()  
    im_res.set_pixels(np.zeros((l_max-l_min+1,c_max-c_min+1),dtype = np.uint8))  
    im_res.pixels[0:l_max-l_min+1,0:c_max-c_min+1] = im_bin.pixels[l_min:l_max+1,c_min:c_max+1]  
    return im_res
```

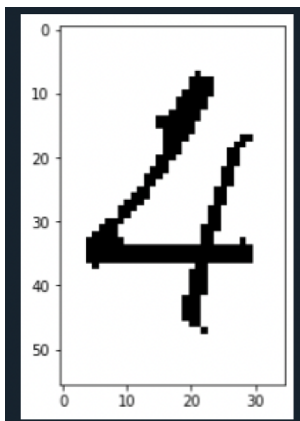
III. Section IV du TP

1.

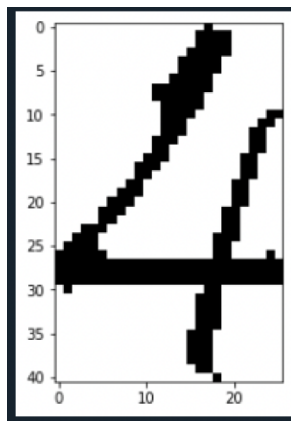
Pour les résultats de la binarisation, ils sont exposés dans la question 2) de la 2^{ème} section du Compte rendu. On peut se rendre compte que la binarisation permet d'écrire plus nettement le chiffre car selon la valeur de S, il permet de supprimer les petits points gris qui parasite l'image. Plus on augmente S, plus les cases grises de plus en plus clair deviendront noir après binarisation. Ce sera l'inverse si l'on diminue la valeur de S. Le bon choix du S permet d'obtenir la meilleur image possible.

2.

Avant localisation :



Après localisation :



Cette méthode nous permet de « recentrer » le chiffre. Il s'agit de la recadrer dans un rectangle. La fonction calcule la taille de ce rectangle. Elle cherche le pixel noir le plus à droite, le plus à gauche, le plus haut et le plus bas. Ces nouvelles données permettent de redimensionner le tableau de pixels et ainsi avoir une image sans bords blanc.

3.

Pour pouvoir comparer l'image avec la base de donnée, il faut que l'image soit à la même résolution que les données. De ce fait, la fonction « resize » change la résolution de l'image en lui donnant le bon nombre de colonnes et le bon nombre de lignes.

```
def resize(self, new_H, new_W):  
    im_resized = Image()  
    im_resized.H = new_H  
    im_resized.W = new_W  
    im_resized.pixels = resize(self.pixels, (new_H, new_W), 0)  
    im_resized.pixels = np.uint8(im_resized.pixels*255)  
    return im_resized
```

4.

Pour compter le nombre de pixels similaire entre l'image et les différentes données, il nous faut que l'image et la donnée aient la même résolution. On utilise donc la méthode « resize ». Ensuite on parcourt les deux images afin de pouvoir compter le nombre de similarités. Enfin la méthode retourne le nombre de similarités divisé par le nombre de pixels totaux.

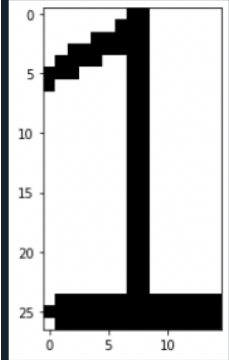
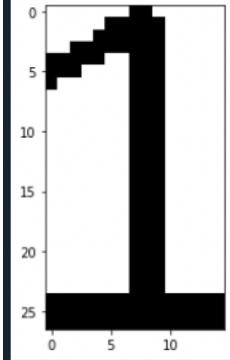
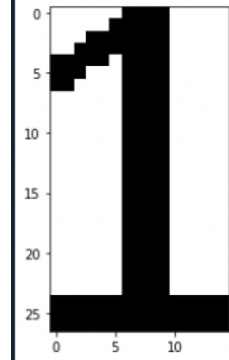
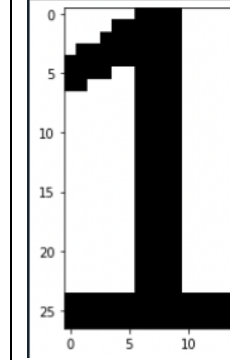
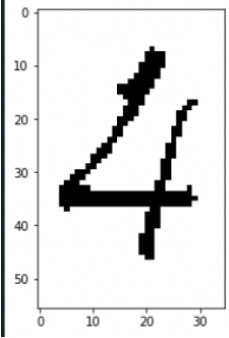
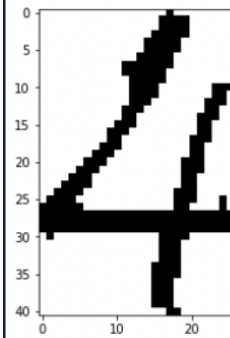
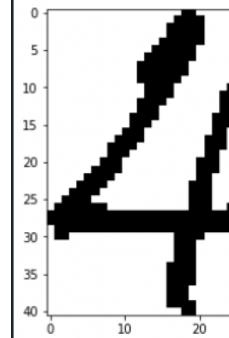
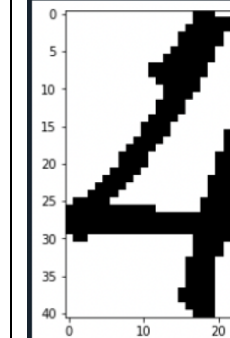
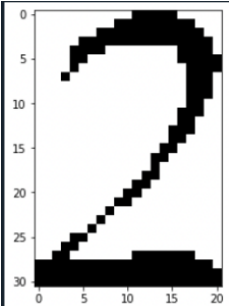
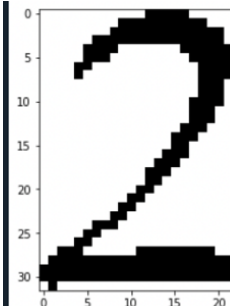
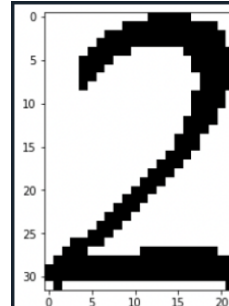
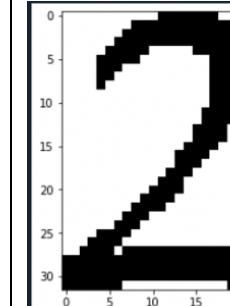
```
def similitude(self, im):  
    im_bin = self.binarisation(127)  
    sim = 0  
    pixels = 0  
    res = 0  
    self.resize(100, 100)  
    im.resize(100, 100)  
    for i in range(im_bin.H):  
        for j in range(im_bin.W):  
            if self.pixels[i][j] == im.pixels[i][j]:  
                sim = sim+1  
                pixels = pixels+1  
    res = sim / pixels  
    return res
```

5.

D'après les similitudes, la méthode reconnaissance_chiffre va nous permettre de reconnaître quel chiffre est à l'écran. On va alors comparer notre tableau de pixels provenant d'une image à des tableaux de références. On calcule pour chaque référence le taux de similitudes. Le taux le plus élevé correspond au chiffre contenu dans le tableau de références. On peut alors reconnaître le chiffre depuis une image.

```
def reconnaissance_chiffre(image, liste_modeles, S):
    res=0
    simmax=0
    im = image.localisation()
    for i in range (len(liste_modeles)):
        H = liste_modeles[i].H
        W = liste_modeles[i].W
        im = im.resize(H, W)
        sim = im.similitude(liste_modeles[i])
        if sim > simmax :
            simmax = sim
            res = i
    return res
```

6.

Chiffre \ Valeur de S	50	100	150	200
1				
4				
2				

D'après le tableau, une valeur de S autour de 100 serait le plus approprié. En effet, en dessous on pourrait manquer d'informations pour voir bien le chiffre. Au-dessus de 150 les traits sont trop épais et admet surement trop d'informations.

IV. Conclusion

Pour conclure, ce TP nous a permis d'acquérir des base dans les traitements de l'image. On a appris à convertir une photo en tableau contenant des pixels. Puis redimensionner et traiter ce tableau afin de reconnaître un chiffre à partir d'une image .