

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

### I. Introduction

Ce TP a pour but de nous faire manipuler les classes et les fonctions sur Python afin de permettre de créer un algorithme qui puisse reconnaître le chiffre sur une image.

### II. Travail préparatoire

#### 1. Question (1).

#### 2. Question (2).

Pour commencer on a complété le programme « binarisation » en ajoutant deux boucles « for » afin de parcourir et de comparer pixel par pixel.

On compare les pixels de l'image en utilisant le nombre attribué à leur couleur (de 0 à 255) avec le seuil que l'on a fixé dans la commande.

Si la valeur du pixel est supérieur au seuil alors on affecte au pixel de coordonnées (y,x) la valeur 255, si elle est inférieur ou égale au seuil alors on lui affecte la valeur 0 (noir).

```
def binarisation(self, S):  
  
    im_bin = Image()  
  
    im_bin.set_pixels(np.zeros((self.H, self.W), dtype=np.uint8))  
    for y in range(self.H):  
        for x in range(self.W):  
            if self.pixels[y][x] > S :  
                im_bin.pixels[y][x] = 255  
            if self.pixels[y][x] < S :  
                im_bin.pixels[y][x] = 0  
            if self.pixels[y][x] == S:  
                im_bin.pixels[y][x] = 0  
  
    return im_bin
```

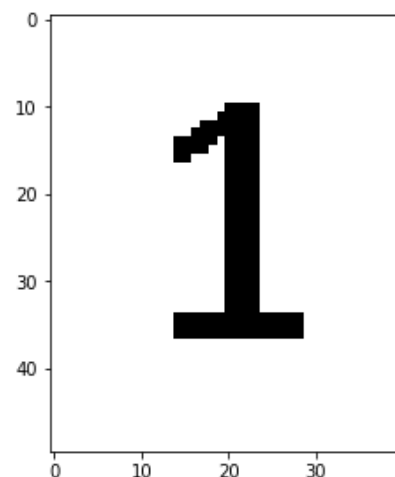
Par la suite pour afficher la nouvelle image, on appelle d'abord l'image de base grâce aux trois commandes ci-dessous.

Puis on donne un nom à la nouvelle image (i) et on applique le programme binarisation à (im) avec un seuil de 150, qu'on affiche à l'aide de display :

```
In [8]: im=Image()  
  
In [9]: im.load(path_to_assets + "test2.jpg")  
lecture image : ../assets/test2.jpg (50x40)  
  
In [35]: im.display(1)
```

```
In [38]: i=im.binarisation(150)
```

```
In [39]: i.display(1)
```



### 3. Question (3).

```
def localisation(self):  
  
    im_loc = Image()  
    c_min=0  
    c_max=0  
    l_min=0  
    l_max=0  
  
    im_loc.set_pixels(self.pixels)  
    for y in range (self.H):  
        for x in range (self.W):  
            if self.pixels[y][x] == 0:  
                l_max=y+1  
  
    for x in range (self.W):  
        for y in range (self.H,):  
            if self.pixels[y][x] == 0:  
                c_max=x+1  
  
    for y in range (self.H-1,0,-1):  
        for x in range (self.W-1,0,-1):  
            if self.pixels[y][x] == 0:  
                l_min=y  
  
    for x in range (self.W-1,0,-1):  
        for y in range (self.H-1,0,-1):  
            if self.pixels[y][x] == 0:  
                c_min=x  
  
    im_loc.pixels=im_loc.pixels[l_min:l_max,c_min:c_max]  
    return im_loc
```

On effectue quatre boucles « for » dans le but d'obtenir les coordonnées du rectangle qui englobe la forme noire.

Dans la première boucle par exemple, on parcourt ligne par ligne l'image et on regarde si le pixel est noir c'est-à-dire s'il vaut 0, dans ce cas on affecte la valeur de l'ordonné +1 à l\_max puis la boucle continue. A la fin l\_max correspond à l'ordonné +1 du dernier pixel noir. La deuxième boucle répond au même principe, cependant elle parcourt les pixels colonne par colonne et affecte donc à c\_max l'abscisse +1 du dernier pixel noir rencontré.

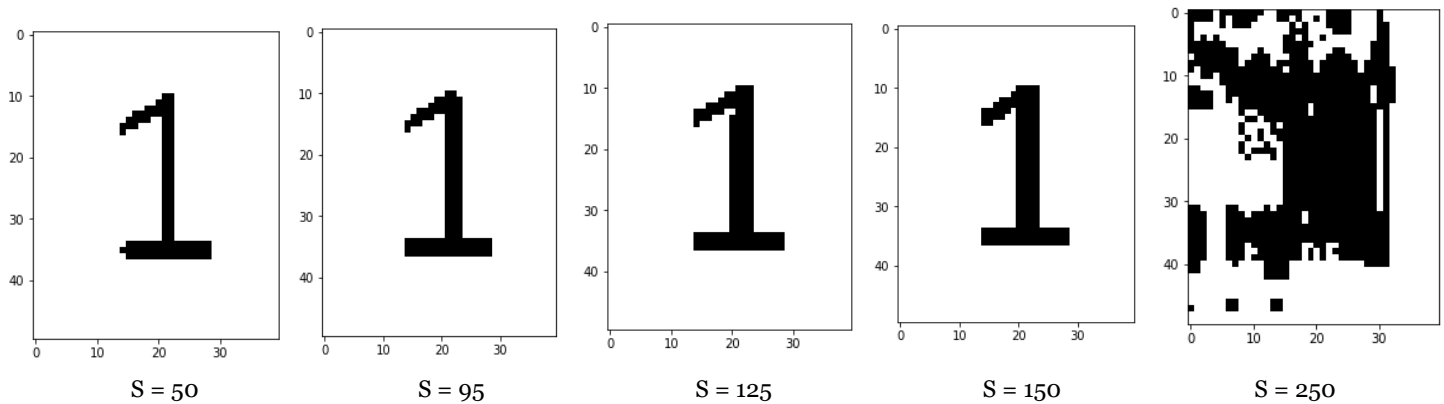
Les deux dernières boucles conservent le même raisonnement que les deux premières néanmoins elles traversent les pixels de bas en haut pour obtenir l\_min et de gauche à droite pour avoir c\_min.

La commande `im_loc.pixels=im_loc.pixels[l_min:l_max,c_min:c_max]` nous donne la partie de l'image limitée aux l\_min à l\_max-1 et aux colonnes c\_min à c\_max-1 c'est pourquoi dans les deux premières boucles nous avons ajouté +1 à y et x.

### III. Reconnaissance automatique de chiffres

#### 1. Question (1).

Après plusieurs tests avec des valeurs différentes, nous obtenons les images ci-dessous :

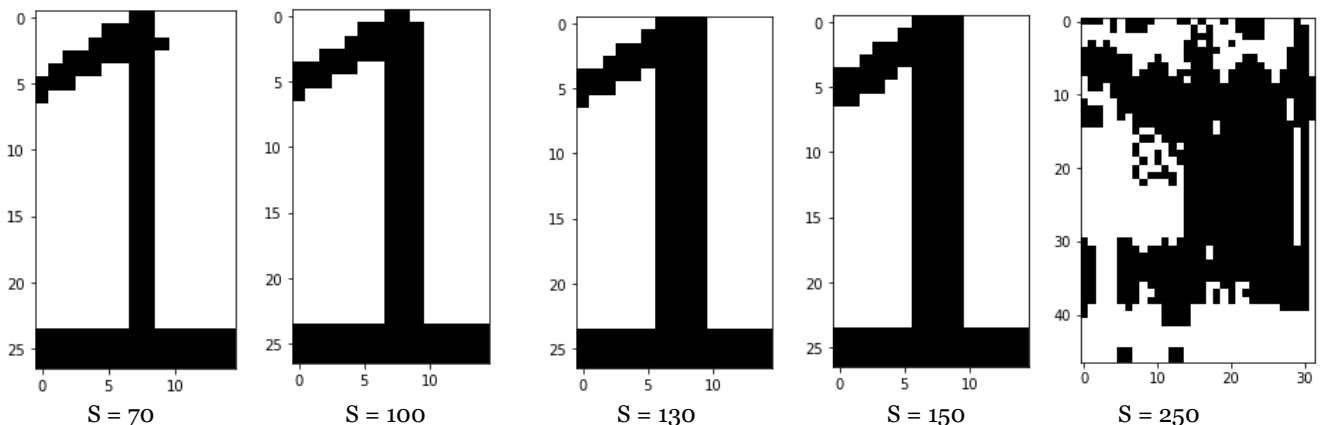


Donc d'après les résultats obtenus, nous constatons que le seuil choisi ne doit pas être trop élevé et également que la valeur la plus adaptée semble être 150.

Les tests sur la méthode de binarisation ne renvoient pas d'erreur, nous ne modifions donc pas notre code.

#### 2. Question (2).

Après plusieurs tests avec des valeurs différentes, nous obtenons les images ci-dessous :



Les tests sur la méthode de localisation ne renvoient pas d'erreur, nous ne modifions donc pas notre code.

#### 3. Question (3).

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

On définit l'image `im_res` par un tableau de pixels obtenus grâce à la fonction `resize`.

Puis on transforme le codage float en int pour se ramener à une image.

On attribue à la hauteur de l'image la valeur `new_H` et à la largeur de l'image la valeur `new_W`.

#### 4. Question (4).

```
def similitude(self, im):  
    k = 0  
    n = 0  
    im_sim = Image()  
    im_sim.set_pixels(self.pixels)  
    im.set_pixels(im.pixels)  
    for y in range (self.H) :  
        for x in range (self.W):  
            n = n+1  
            if self.pixels[y][x]==im.pixels[y][x]:  
                k = k+1  
    return k/n
```

A l'aide de deux boucles for on a parcouru tous les pixels des deux images un par un.

Les boucles vont comparer la couleur des pixels s'ils sont noirs (0) ou blancs (255), et s'ils sont de même couleur et au même endroit alors on ajoutera 1 à la variable k.

La variable n sert à compter le nombre total de pixel pour l'image.

A la fin on retourne k/n pour avoir un ratio entre 0 et 1.

Après avoir effectué le test, il ne nous renvoie aucune erreur.

#### 5. Question (5).

```
def reconnaissance_chiffre(image, liste_modeles, S):  
    l=[]  
    s=0  
    n=-1  
    indice=0  
    image = image.binarisation(S)  
    image=image.localisation()  
    for models in liste_modeles:  
        models=models.binarisation(S)  
        models=models.localisation()  
        models=models.resize(100, 60)  
        image = image.resize(100, 60)  
        i=image.similitude(models)  
        l.append(i)  
    for j in l:  
        n=n+1  
        if j>s:  
            s=j  
            indice=n  
    return indice
```

Premièrement, nous avons effectué les méthodes de binarisation et de localisation sur l'image « image ».

Nous avons également ajouté image = Image() or cela a provoqué une disfonction dans notre code, car notre liste obtenue ne changeait jamais. En effet, elle se basait systématiquement sur la même nouvelle image ( image = Image() ) donnée. Nous avons donc supprimé cette partie.

Après nous avons fait une boucle « for » pour prendre chaque chiffre « models » de la liste. Ainsi pour chacun des modèles nous avons utilisé les fonctions binarisation et localisation.

Ensuite la fonction « resize » a été appliquée pour mettre l'image et le modèle à la même taille.

Puis nous les avons comparés grâce à la fonction similitude.

Par la suite nous avons ajouté à une liste « l » tous les ratios.

A l'aide d'une boucle « for », les ratios ont été comparés et l'indice du plus grand retourné.

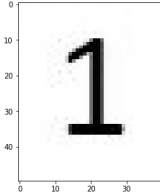
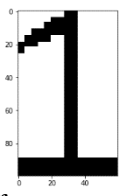
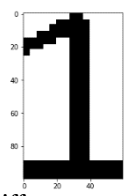
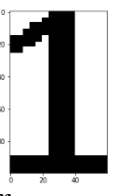
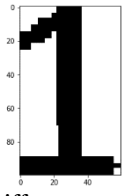
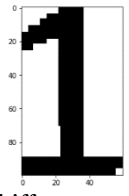
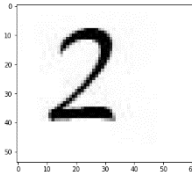
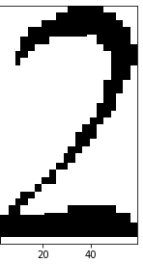
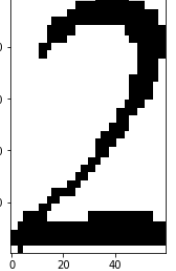
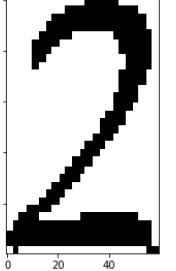
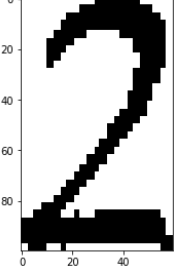
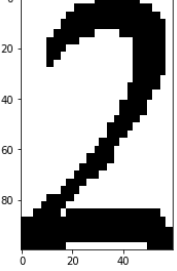
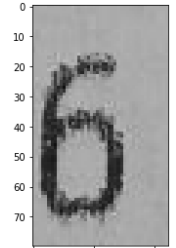

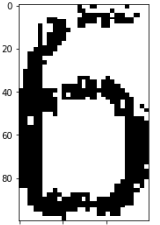
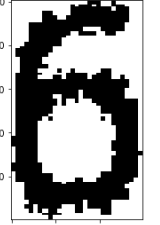

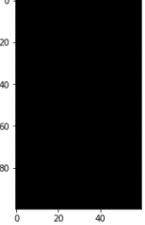
Pour le n = n+1, la valeur a été initialisée à -1 pour qu'au premier tour on obtienne n=0 pour l'indice 0.

Pour finir, on renvoie l'indice puisqu'il correspond au chiffre représenté sur l'image.



Après avoir effectué le test, nous n'avons trouvé aucune erreur.

## 6. Question (6).

Images	Seuils				
	60	100	150	170	200
	 Chiffre reconnu : 1	 Chiffre reconnu : 1	 Chiffre reconnu : 1	 Chiffre reconnu : 1	 Chiffre reconnu : 1
	 Chiffre reconnu : 2	 Chiffre reconnu : 2	 Chiffre reconnu : 2	 Chiffre reconnu : 2	 Chiffre reconnu : 2
	 Chiffre reconnu : 6	 Chiffre reconnu : 6	 Chiffre reconnu : 6	 Chiffre reconnu : 6	 Chiffre reconnu : 8

D'après les résultats obtenus, on détermine que le meilleur seuil est de 150.

## IV. Conclusion :

Grâce à ce TP nous avons réussi à mieux appréhender les classes et ainsi trouver des solutions. Ce TP nous a appris comment modifier et comparer des images entre elles.