TP IA : Algorithme KNN & Machine Learning : Identification automatique de panneaux

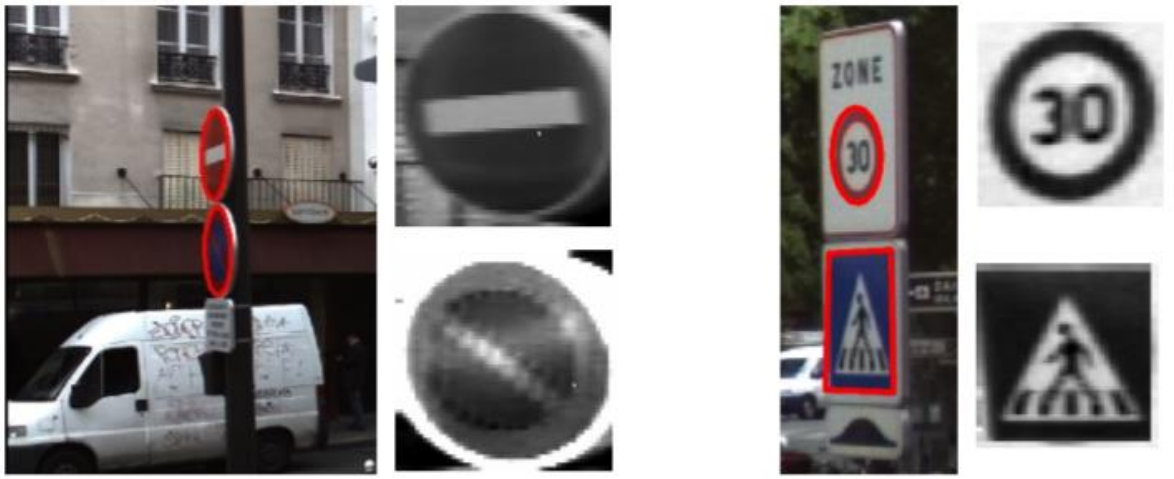
1. Contexte

Nous allons développer un algorithme capable de reconnaitre une image automatiquement à partir de l'apprentissage d'une base d'images sources. Pour cela, nous allons utiliser l'algorithme KNN, dit de recherche des « k plus proches voisins » en utilisant la norme euclidienne.

Nos images recherchées seront supposées prétraitées afin qu'elles ressemblent à cela :

Le pré-traitement numérique aura donc réalisé les étapes suivantes :

- Transformation projective en couleur afin d'avoir une image « vue de face » et cadrée



Source : Page 51 de l'ouvrage intitulé « Détection et reconnaissance de la signalisation verticale par analyse d'image » en lien ici.

- Redimensionnement des images avec 100 lignes et 100 colonnes, soit 10 000 pixels RGB
- Enregistrement du résultat au format BMP

Les images sources permettant l'apprentissage auront subi un traitement supplémentaire de mise en blanc du fond en dehors du panneau, et nous nous limiterons à des panneaux circulaires.

2. Téléchargement du dossier élèves

Téléchargez et décompressez le « dossier eleve » sur cahier de prépa. Vous aurez les images sources, les images recherchées, et un code Python à compléter.

3. Mise en place de l'algorithme KNN

Nous allons travailler avec des n-uplets (listes de n termes) qui seront associées à chaque image. On suppose que le nombre de termes n de chaque n-uplets est identique dans tout le TP.

On rappelle que la distance euclidienne entre deux n-uplets $u = (u_0, u_1, \dots, u_{n-1})$ et $v = (v_0, v_1, \dots, v_{n-1})$ est le résultat du calcul suivant :

$$D = \sqrt{\sum_{i=0}^{n-1} (v_i - u_i)^2}$$

Question 1: Créer une fonction Distance_uv(u,v) calculant la distance euclidienne entre les deux n-uplets sous forme de listes u et v

Vérifier :

```
>>> u = [1,3,5]

>>> v = [2,4,6]

>>> Distance_uv(u,v)
1.7320508075688772
```

Question 2: Créer une fonction Distance(u,Lv) renvoyant une liste des distances euclidiennes entre u et tous les n-uplets v de la liste Lv ainsi que l'indice associé sous la forme [Distance entre u et v, indice de v dans Lv]

Vérifier :

```
>>> u = [1,3,5]

>>> v1 = [1,2,3]

>>> v2 = [2,4,6]

>>> v3 =[1,3,5]

>>> Lv = [v1,v2,v3]

>>> Distance(u,Lv)
[[2.23606797749979, 0], [1.7320508075688772, 1], [0.0, 2]]
```

On donne la fonction Proches(u,Lv,k) qui renvoie une liste des k plus proches voisins de u dans la liste Lv au sens de la norme euclidienne, soit les k listes [dst,ind]

Remarques :

- On supposera que k est plus petit que le nombre de n-uplets de Lv
- L.sort() permet de trier les éléments de L par rapport à la première composante de tous ses éléments (les distances ici)

A la suite du code précédent, vérifier :

```
>>> Proches(u,Lv,1)
[[0.0, 2]]
```

4. Lecture des images

On donne le code suivant :

```
import matplotlib.pyplot as plt
plt.close('all')

def Affiche(image):
plt.figure()
plt.imshow(image)
plt.axis('off')
plt.show()
plt.pause(0.00001)
```

A partir d'une image au format array d'entiers codés sur 8 bits, la fonction Affiche affiche cette image sur une figure.

On rappelle que l'ouverture au format array d'une image se réalise avec la commande :

```
plt.imread(Chemin)
```

Attention, les chemins dans Python s'écrivent : Sources\\1\\0.bmp (dossier sources puis dossier 1 puis image 0).

Question 3: Créer une fonction Lecture(Chemin) qui renvoie l'array associé à l'image de chemin contenu dans la variable Chemin

5. Fonctions d'analyse des images

On utilisera la méthode suivante : pour chaque image, on crée une liste L_RGB des valeurs de R, G et B de chacun de ses pixels. Ainsi, avec des images ayant toujours la même dimension de 100x100 pixels, on obtient une liste de 30 000 valeurs pour chacune. Comme chaque n-uplet doit avoir la même taille, vous comprenez pourquoi toutes les images ont le même nombre de pixels.

Question 4: Créer une fonction Analyse(Image) qui, à partir d'une image sous forme d'array, renvoie la liste L_RGB associée – Attention, transformer les R, G et B en flottants, sinon il y aura overflow avec les uint8 lors des calculs de distances (pour ce faire, on pourra utiliser : « R = float(R) »).

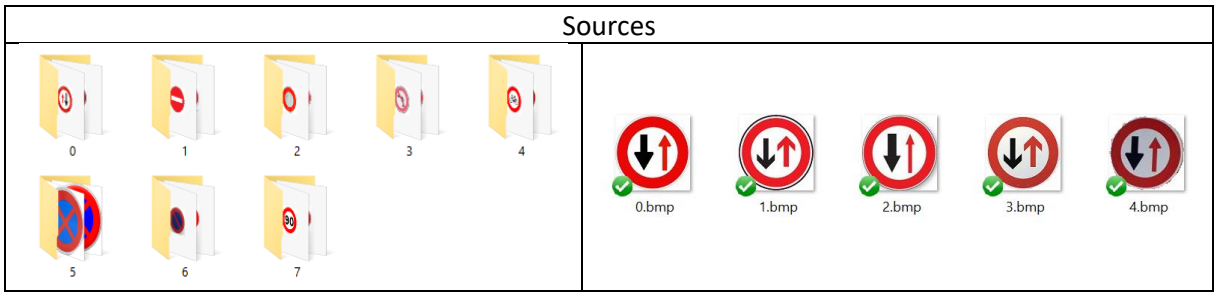
Question 5: Créer une fonction Analyse_Globale(L_Chemin) qui, à partir d'une liste des chemins des images à analyser, renvoie la liste des listes L_RGB de chacune des images concernées

Remarque : on pourra faire en sorte que cette fonction affiche l'état d'avancement du traitement des images, par exemple :

```
Apprentissage image 1 sur 40
Apprentissage image 2 sur 40
Apprentissage image 3 sur 40
Apprentissage image 4 sur 40
Apprentissage image 5 sur 40
Apprentissage image 6 sur 40
Apprentissage image 7 sur 40
Apprentissage image 8 sur 40
```

6. Création de la base des données

Vous avez à votre disposition un dossier nommé « Sources » dans lequel apparaissent 8 dossiers numérotés de 0 à 7. Chacun de ces dossiers contient 5 images sources du même panneau numérotées de 0 à 4, qui vont servir à l'apprentissage :



Soient les listes :

```
Dossiers =[0,1,2,3,4,5,6,7]
Nb_Images_Dossiers=[5,5,5,5,5,5,5,5]
```

La liste Dossiers liste les numéros des dossiers présents dans le répertoire Sources, et la liste Nb_Images_Dossiers répertorie le nombre d'images contenues dans chacun de ces dossiers.

Chaque image contenue dans le dossier source possède :

- Un chemin
- Un numéro de dossier
- Un numéro d'image

On souhaite créer les trois listes Liste_Chemin, Liste_Dossier et Liste_Num, qui pour un même indice et donc, pour une même image, contiennent ces trois informations. Pour la suite, on appellera « **Indice d'une image** » son indice dans ces trois listes.

Question 6: Mettre en place un code utilisant Dossiers et Nb_Images_Dossiers créant les listes Liste_Chemin, Liste_Dossier et Liste_Num

Vérifiez :

```
>>> Liste_Chemin
['Sources\\0\\0.bmp', 'Sources\\0\\1.bmp'
'Sources\\0\\2.bmp', 'Sources\\0\\3.bmp',
'Sources\\0\\4.bmp', 'Sources\\1\\0.bmp',
'Sources\\1\\1.bmp', 'Sources\\1\\2.bmp',
'Sources\\1\\3.bmp', 'Sources\\1\\4.bmp',
'Sources\\2\\0.bmp', 'Sources\\2\\1.bmp',
'Sources\\2\\2.bmp', 'Sources\\2\\3.bmp',
'Sources\\2\\4.bmp', 'Sources\\3\\0.bmp',
'Sources\\3\\1.bmp', 'Sources\\3\\2.bmp',
'Sources\\3\\3.bmp', 'Sources\\3\\4.bmp',
'Sources\\4\\0.bmp', 'Sources\\4\\1.bmp',
'Sources\\4\\2.bmp', 'Sources\\4\\3.bmp',
'Sources\\4\\4.bmp', 'Sources\\5\\0.bmp',
'Sources\\5\\1.bmp', 'Sources\\5\\2.bmp',
'Sources\\5\\3.bmp', 'Sources\\5\\4.bmp',
'Sources\\6\\0.bmp', 'Sources\\6\\1.bmp',
'Sources\\6\\2.bmp', 'Sources\\6\\3.bmp',
'Sources\\6\\4.bmp', 'Sources\\7\\0.bmp',
'Sources\\7\\1.bmp', 'Sources\\7\\2.bmp',
'Sources\\7\\3.bmp', 'Sources\\7\\4.bmp']
```

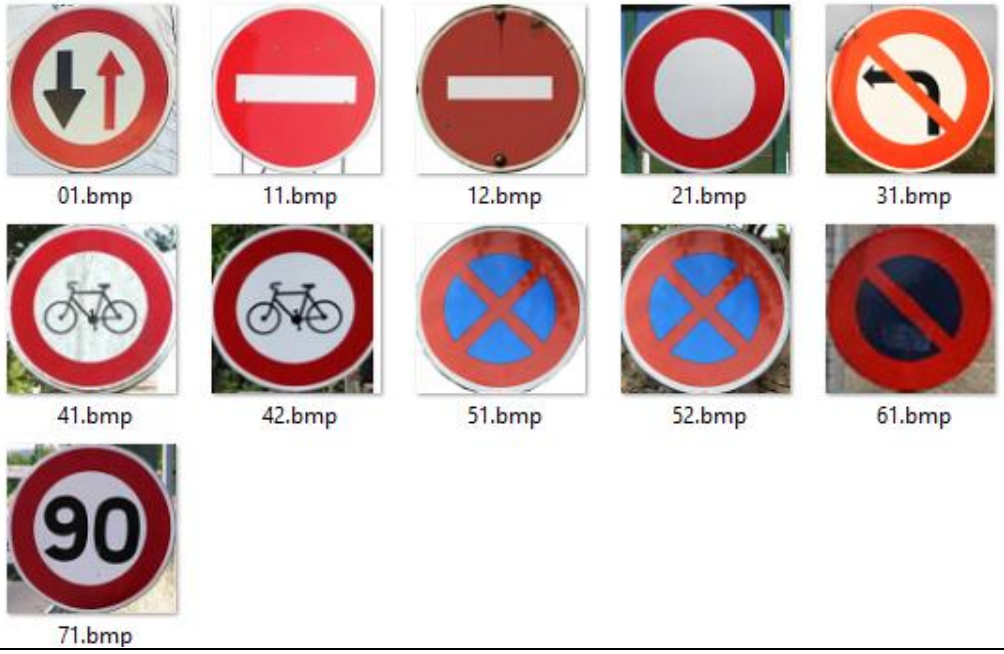
```
>>> Liste_Dossier
[0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 2, 2, 2, 2,
2, 3, 3, 3, 3, 3, 4, 4, 4, 4, 4, 5, 5, 5,
5, 5, 6, 6, 6, 6, 6, 7, 7, 7, 7, 7]

>>> Liste_Num
[0, 1, 2, 3, 4, 0, 1, 2, 3, 4, 0, 1, 2, 3,
4, 0, 1, 2, 3, 4, 0, 1, 2, 3, 4, 0, 1, 2,
3, 4, 0, 1, 2, 3, 4, 0, 1, 2, 3, 4]
```

Question 7: Écrire le code nécessaire à la création de la liste de listes « Donnees », contenant les listes L_RGB de toutes les images sources

7. Reconnaissance automatique

Vous avez à votre disposition un dossier nommé « Recherche » contenant des images à rechercher automatiquement à l'aide de l'algorithme mis en place. Elles sont toutes issues d'une photo en situation réelle et le premier numéro de leurs noms correspond au dossier auquel elles devraient appartenir.

Recherche



Les images sources ayant un fond blanc, notre algorithme s'adapte automatiquement à n'importe quel fond. En effet, une image recherchée ayant un fond quelconque sera à la même « distance » que toutes les images sources sur la partie extérieur, l'algorithme sélectionnera alors celle qui se rapproche le plus dans la comparaison du contenu intérieur du panneau.

Question 8: Créer un code qui permet d'ouvrir, d'afficher et d'analyser (création de sa liste L_RGB) l'une des images du dossier Recherche à choisir

Question 9: Créer un code qui détermine les k=5 plus proches voisins de l'image recherchée et crée les listesResultat_Ind(indices des images résultats), Resultat_Dossiers(dossiers correspondants) et Resultat_Num (numéros des images dans les dossiers) et qui affiche dans la console dossiers et numéros des images trouvées

Question 10: Créer une fonction Max_Occurences(L) qui renvoie le terme apparaissant le plus dans L, et le premier s'il y a des exæquo

Vérifiez :

```
>>> L = [3,3,1,2,3]
>>> Max_Occurences(L)
3
```

```
>>> L = [1,3,1,3,4]
>>> Max_Occurences(L)
1
```

Question 11: Créer un code permettant de déterminer le dossier résultat, qui l'affiche dans la console et affiche l'une des images de ce dossier

Vérifiez la réussite de l'identification de toutes les images à rechercher.

8. Matrice de confusion

Pour évaluer le succès de l'algorithme KNN, on utilise une matrice de confusion. Dans notre application, nous avons deux types de données :

- Les types de panneaux appris
- Les types de panneaux recherchés

Ainsi, nous avons 8 types de panneaux appris et nous avons recherché les mêmes 8 types de panneaux. La matrice est définie ainsi :

- Chaque ligne correspond à un type de panneau recherché
- Chaque colonne représente le type de panneau trouvé

En nous adaptant aux données de ce TP, cela donne :

- Ligne l : premier chiffre du nom de l'image recherchée
- Colonne c : numéro du dossier trouvé

Alors, le terme de la matrice en $M(l,c)$ avec $c = l$ est le nombre d'images sources trouvées comme images cibles. Autrement dit, si l'algorithme trouve toutes les images correctement dans notre exemple, on aura la matrice ci-contre.

$$M = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 2 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 2 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 2 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 2 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$

Question 12: Indiquer ce que voudrait dire $M(1,0) = 1$

Question 13: Créer la liste LR des numéros des images à rechercher sous forme de str

```
>>> LR
['01', '11', '12', '21', '22', '31', '41', '42', '51', '52', '61', '71']
```

Question 14: Mettre en place le code créant la liste des chemins LCR des images recherchées

```
>>> LCR
['Recherche\\01.bmp', 'Recherche\\11.bmp', 'Recherche\\12.bmp', 'Recherche\\21.bmp',
'Recherche\\22.bmp', 'Recherche\\31.bmp', 'Recherche\\41.bmp', 'Recherche\\42.bmp',
'Recherche\\51.bmp', 'Recherche\\52.bmp', 'Recherche\\61.bmp', 'Recherche\\71.bmp']
```

Question 15: Créer la liste Donnees_R des données des images recherchées en utilisant Analyse_Globale

Question 16: Mettre en place une fonction Resolution(DR,k) prenant en argument les données d'une image recherchée DR et le nombre de voisins k utilisés, et renvoyant le numéro de dossier trouvé

Vérifier :

```
>>> Resolution(Donnees_R[0],5)
0

>>> Resolution(Donnees_R[2],5)
1

>>> Resolution(Donnees_R[11],5)
7
```

Question 17: Créer la fonction Etude(k) réalisant la matrice de confusion de notre algorithme pour une valeur de k donnée

Remarque : les listes LR et Dossiers sont utilisées de manière globale

Question 18: Déterminer la matrice de confusion de notre algorithme pour k allant de 1 à 5 et conclure

9. Utilisation de sklearn

Il existe une bibliothèque qui permet de réaliser la résolution knn rapidement, elle n'est pas attendue dans le programme d'informatique mais introduite dans le programme de SI. Vous avez le code ci-dessous dans le dossier élèves :

```
# Données

x = Donnees
y = Liste_Dossier
DR = Donnees_R[0]
k = 5

# Apprentissage

from sklearn.model_selection import train_test_split
from sklearn.neighbors import KNeighborsClassifier
knn = KNeighborsClassifier(k)
knn.fit(x,y)
Score = knn.score(x_test,y_test)
print("Score:",Score)

# Reconnaissance

Sol = knn.predict([DR])[0]
print(Sol)

# Matrice de confusion

from sklearn.metrics import confusion_matrix
y_true = []
y_pred = []
for i in range(N):
    y_true.append(######)
    y_pred.append(######)
Mat = confusion_matrix(y_true,y_pred)
print(Mat)
```

Il faut définir :

- La liste x des données x
- La liste y des résultats attendus de chaque donnée x
- La liste DR des données de l'image recherchée
- Pour la matrice de confusion, les listes y_true et y_pred des solutions réelles et des solutions obtenues pour la reconnaissance de toutes les images dans Donnees_R

Question 19: Utiliser ce code pour visualiser la prédiction sur les images recherchées et la matrice de confusion pour k=5

Source : D. Defauchy