



MASTER ISI:  
DATA MINING

---

## TP N°4

---

Réalisé par :

ESSADEQ Ayoub

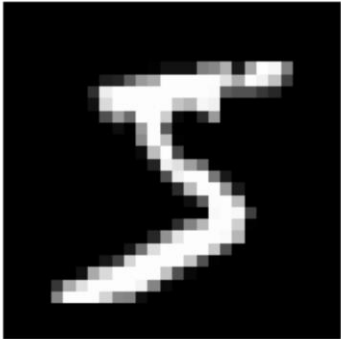
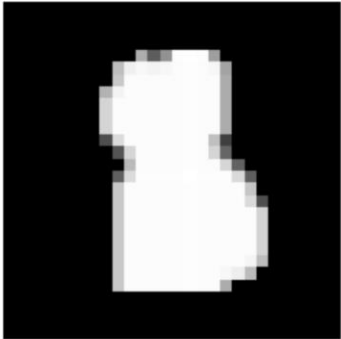






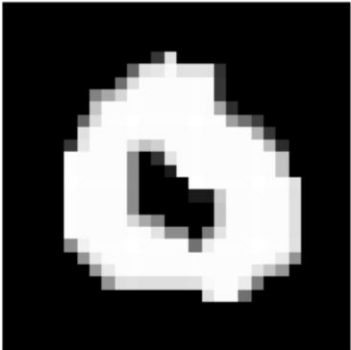
RAHHAOUI ABDESSAMAD

2023-2024

1) Calculer la distance Similarité et corrélation entre une image Requête et les images de la base afficher les 9 premiers résultats (Utiliser la fonction Python Subplot( 3,3, ..) )  
Après un tri par Ordre croissant :

```
import numpy as np
import matplotlib.pyplot as plt
from scipy.spatial.distance import euclidean
from scipy.stats import pearsonr
import keras # Ajout de l'importation Keras
# Charger les données MNIST
(x_train, y_train), (_, _) = keras.datasets.mnist.load_data()
base = x_train
# Définir votre image requête (à remplacer par votre propre image)
requete = base[0]
# Fonction pour calculer la distance euclidienne
def euclidean_distance(img1, img2):
    return euclidean(img1.flatten(), img2.flatten())
# Fonction pour calculer la similarité de corrélation de Pearson
def correlation_similarity(img1, img2):
    corr, _ = pearsonr(img1.flatten(), img2.flatten())
    return corr
# Calculer les distances et similarités
distances = [euclidean_distance(requete, img) for img in base]
similarities = [correlation_similarity(requete, img) for img in base]
# Tri des indices par distance (ordre croissant)
sorted_indices = np.argsort(distances)
# Afficher les 9 premiers résultats
fig, axes = plt.subplots(3, 3, figsize=(8, 8))
for i, ax in enumerate(axes.flat):
    index = sorted_indices[i]
    ax.imshow(base[index], cmap='gray')
    ax.set_title(f'Distance:
{distances[index]:.2f}\nSimilarité:{similarities[index]:.2f}')
    ax.axis('off')
plt.tight_layout()
plt.show()
```

Le résultat :

<p>Distance: 0.00 Similarité:1.00</p> 	<p>Distance: 1787.92 Similarité:0.55</p> 	<p>Distance: 1818.59 Similarité:0.67</p> 
<p>Distance: 1841.44 Similarité:0.58</p> 	<p>Distance: 1849.29 Similarité:0.57</p> 	<p>Distance: 1859.65 Similarité:0.64</p> 
<p>Distance: 1871.29 Similarité:0.73</p> 	<p>Distance: 1872.02 Similarité:0.57</p> 	<p>Distance: 1889.09 Similarité:0.37</p> 

2) Calculer la distance Similarité et corrélation entre l'histogramme d'une image Requête et les histogrammes des images de la base afficher les 9 premiers résultats (Utiliser la fonction Python Subplot( 3,3, ..) ) Apres un tri par Ordre croissant Travailler sur les data sets de KERAS (MNIST ; CIFAR10 et CIFAR100)

```
import numpy as np
import matplotlib.pyplot as plt
from scipy.spatial.distance import euclidean
from scipy.stats import pearsonr
from skimage import exposure
import keras
# Charger les données CIFAR100
(x_train, y_train), (_, _) = keras.datasets.cifar100.load_data()
base = x_train
# Définir votre image requête (à remplacer par votre propre image)
requete = base[0]
# Fonction pour calculer l'histogramme d'une image
def compute_histogram(img):
    return np.histogram(img.flatten(), bins=256, range=[0, 256])[0]
# Fonction pour calculer la distance euclidienne entre deux histogrammes
def euclidean_distance_histogram(hist1, hist2):
    return euclidean(hist1, hist2)
# Fonction pour calculer la similarité de corrélation de Pearson entre
deux histogrammes
def correlation_similarity_histogram(hist1, hist2):
    corr, _ = pearsonr(hist1, hist2)
    return corr
# Calculer les histogrammes
histogram_requete = compute_histogram(requete)
histograms_base = [compute_histogram(img) for img in base]
# Calculer les distances et similarités entre l'histogramme de la requête
et les histogrammes de la base
distances = [euclidean_distance_histogram(histogram_requete, hist) for
hist in
histograms_base]
similarities = [correlation_similarity_histogram(histogram_requete, hist)
for
hist in histograms_base]
# Tri des indices par distance (ordre croissant)
sorted_indices = np.argsort(distances)
# Afficher les 9 premiers résultats
fig, axes = plt.subplots(3, 3, figsize=(8, 8))
for i, ax in enumerate(axes.flat):
    index = sorted_indices[i]
    ax.imshow(base[index])
```

```
ax.set_title(f'Distance: {distances[index]:.2f}\nSimilarité: {similarities[index]:.2f}')
ax.axis('off')
plt.tight_layout()
plt.show()
```

**Le résultat :**

Distance: 0.00  
Similarité: 1.00



Distance: 87.34  
Similarité: 0.94



Distance: 91.35  
Similarité: 0.94



Distance: 92.83  
Similarité: 0.94



Distance: 94.40  
Similarité: 0.93



Distance: 98.55  
Similarité: 0.93



Distance: 99.56  
Similarité: 0.93



Distance: 100.63  
Similarité: 0.92



Distance: 100.77  
Similarité: 0.93

