Programming Assignment-4

In [7]:
```
import pathlib
from tqdm import tqdm
import cv2
import os.path
import os
import numpy as np
import pandas as pd
from glob import glob
from pathlib import Path
import warnings
warnings.filterwarnings("ignore")
```

In [2]:
```
desktop = pathlib.Path("Cropped")
```

In [3]:
```
import torch
from torchvision.models import resnet18
from PIL import Image
from torchvision import transforms
```

In [4]:
```
model=resnet18(pretrained=True)
model=torch.nn.Sequential(*(list(model.children())[:-1]))
model.eval()
```

Out[4]:

```
Sequential(
  (0): Conv2d(3, 64, kernel_size=(7, 7), stride=(2, 2), padding=(3, 3), bias=False)
  (1): BatchNorm2d(64, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
  (2): ReLU(inplace=True)
  (3): MaxPool2d(kernel_size=3, stride=2, padding=1, dilation=1, ceil_mode=False)
  (4): Sequential(
    (0): BasicBlock(
      (conv1): Conv2d(64, 64, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=False)
      (bn1): BatchNorm2d(64, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
      (relu): ReLU(inplace=True)
      (conv2): Conv2d(64, 64, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=False)
      (bn2): BatchNorm2d(64, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    )
    (1): BasicBlock(
      (conv1): Conv2d(64, 64, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=False)
      (bn1): BatchNorm2d(64, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
      (relu): ReLU(inplace=True)
      (conv2): Conv2d(64, 64, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=False)
      (bn2): BatchNorm2d(64, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    )
  )
  (5): Sequential(
    (0): BasicBlock(
      (conv1): Conv2d(64, 128, kernel_size=(3, 3), stride=(2, 2), padding=(1, 1), bias=False)
      (bn1): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
      (relu): ReLU(inplace=True)
      (conv2): Conv2d(128, 128, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=False)
      (bn2): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
      (downsample): Sequential(
        (0): Conv2d(64, 128, kernel_size=(1, 1), stride=(2, 2), bias=False)
        (1): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
      )
    )
    (1): BasicBlock(
      (conv1): Conv2d(128, 128, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=False)
      (bn1): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
      (relu): ReLU(inplace=True)
      (conv2): Conv2d(128, 128, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=False)
      (bn2): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    )
  )
  (6): Sequential(
    (0): BasicBlock(
      (conv1): Conv2d(128, 256, kernel_size=(3, 3), stride=(2, 2), padding=(1, 1), bias=False)
      (bn1): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
      (relu): ReLU(inplace=True)
      (conv2): Conv2d(256, 256, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=False)
      (bn2): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
      (downsample): Sequential(
        (0): Conv2d(128, 256, kernel_size=(1, 1), stride=(2, 2), bias=False)
```

```
      (1): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    )
  )
  (1): BasicBlock(
    (conv1): Conv2d(256, 256, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=False)
    (bn1): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (relu): ReLU(inplace=True)
    (conv2): Conv2d(256, 256, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=False)
    (bn2): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
  )
)
(7): Sequential(
  (0): BasicBlock(
    (conv1): Conv2d(256, 512, kernel_size=(3, 3), stride=(2, 2), padding=(1, 1), bias=False)
    (bn1): BatchNorm2d(512, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (relu): ReLU(inplace=True)
    (conv2): Conv2d(512, 512, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=False)
    (bn2): BatchNorm2d(512, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (downsample): Sequential(
      (0): Conv2d(256, 512, kernel_size=(1, 1), stride=(2, 2), bias=False)
      (1): BatchNorm2d(512, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    )
  )
  (1): BasicBlock(
    (conv1): Conv2d(512, 512, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=False)
    (bn1): BatchNorm2d(512, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (relu): ReLU(inplace=True)
    (conv2): Conv2d(512, 512, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=False)
    (bn2): BatchNorm2d(512, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
  )
)
(8): AdaptiveAvgPool2d(output_size=(1, 1))
)
```

In [6]:

```
preprocess = transforms.Compose([
    transforms.Resize((224,224)),
    transforms.ToTensor(),
    transforms.Normalize(mean=[0.485, 0.456, 0.406], std=[0.229, 0.224, 0.225]),
])
```

input_images = [] classes = [] four_label= sorted(os.listdir(desktop)) for index, name in enumerate(four_label): folder = os.path.join(desktop, name) files = glob(os.path.join(folder, '*.jpg')) for file in files: image_path = os.path.join(folder, file) image = Image.open(file).convert("RGB") image = preprocess(image) image = image.unsqueeze(0) # create a mini-batch as expected by the model image=model(image).squeeze().detach().numpy() input_images.append(image) classes.append(index)

In [10]:

```
input_images=np.array(input_images)
classes=np.array(classes)
```

In [12]:
```
#reducing components
from sklearn.decomposition import PCA
reduced=PCA(2)
reduced_input=reduced.fit_transform(input_images)
```

In [15]:
```
from sklearn.cluster import KMeans,BisectingKMeans,SpectralClustering,AgglomerativeClustering,DBSCAN
from sklearn.metrics import fowlkes_mallows_score, silhouette_score
```

In [21]:
```
fowlkes_mallow,silhouette_cof=[],[]
```

In [22]:
```
for init in ['random','k-means++']:
    model=KMeans(n_clusters=4, init= init, random_state=42).fit_predict(reduced_input)
    fowlkes_mallow.append((fowlkes_mallows_score(classes, model),f'{init}')),silhouette_cof.append((silhouette_score(reduced_input,model),f'{init}'))
```

In [24]:
```
model=SpectralClustering(n_clusters=4,random_state=42).fit_predict(reduced_input)
fowlkes_mallow.append((fowlkes_mallows_score(classes, model),'Specctral_clustering')),silhouette_cof.append((silhouette_score(reduced_input,model),'Specctral_clustering')
```

Out[24]:
```
(None, None)
```

In [26]:
```
model=BisectingKMeans(n_clusters=4,init='random',random_state=42).fit_predict(reduced_input)
fowlkes_mallow.append((fowlkes_mallows_score(classes, model),'Bisecting_kmeans')),silhouette_cof.append((silhouette_score(reduced_input,model),'Bisecting_kmeans'))
```

Out[26]:
```
(None, None)
```

In [48]:
```
model = DBSCAN(eps=0.55, min_samples=10).fit_predict(reduced_input)
fowlkes_mallow.append((fowlkes_mallows_score(classes, model),'DBSCAN')),silhouette_cof.append((silhouette_score(reduced_input,model),'DBSCAN'))
```

Out[48]:
```
(None, None)
```

In [49]:
```
clusters = len(set(model)) - (1 if -1 in model else 0)
print(f"Number of clusters: {clusters}")
```

```
Number of clusters: 4
```

In [50]:
```
for link in ['single','complete','average','ward']:
    model=AgglomerativeClustering(n_clusters=4, linkage= link).fit_predict(reduced_input)
    fowlkes_mallow.append((fowlkes_mallows_score(classes, model),f'{link}')),silhouette_cof.append((silhouette_score(reduced_input,model),f'{link}'))
```

best to worst silhouette_coefficient :

In [52]:
```
sorted(silhouette_cof,key=lambda x:x[0],reverse=True)
```

Out[52]:
```
[(0.6235918, 'random'),
 (0.6235918, 'k-means++'),
 (0.62225515, 'ward'),
 (0.6221962, 'Specctral_clustering'),
 (0.6218227, 'complete'),
 (0.6210831, 'Bisecting_kmeans'),
 (0.61992186, 'average'),
 (-0.04253666, 'single'),
 (-0.44324148, 'DBSCAN')]
```

best to worst fowlkes_scores :

In [54]:
```
sorted(fowlkes_mallow,key=lambda x:x[0],reverse=True)
```

Out[54]:
```
[(0.9696193882431478, 'complete'),
 (0.9667587820631002, 'random'),
 (0.9667587820631002, 'k-means++'),
 (0.9643100149450072, 'Specctral_clustering'),
 (0.9641399569249511, 'ward'),
 (0.9620179763497949, 'average'),
 (0.9617040486162625, 'Bisecting_kmeans'),
 (0.4986377685134068, 'single'),
 (0.47379697292390527, 'DBSCAN')]
```

In [24]:
```
sorted_silhouette = sorted(silhouette_cof, key=lambda x: x[0], reverse=True)
print("\nRanking based on Silhouette Coefficient:")
for rank, (score, method) in enumerate(sorted_silhouette, start=1):
    print(f"{rank}. {method} - Silhouette Coefficient: {score}")
```

```
Ranking based on Silhouette Coefficient:
1. random - Silhouette Coefficient: 0.6229459643363953
2. k-means++ - Silhouette Coefficient: 0.6229459643363953
3. Specctral_clustering - Silhouette Coefficient: 0.6218364238739014
4. Bisecting_kmeans - Silhouette Coefficient: 0.6210035681724548
5. average - Silhouette Coefficient: 0.6196712851524353
6. ward - Silhouette Coefficient: 0.6196315884590149
7. complete - Silhouette Coefficient: 0.6155784130096436
8. single - Silhouette Coefficient: -0.05801675468683243
9. DBSCAN - Silhouette Coefficient: -0.24252061545848846
```

In [25]:
```
sorted_fowlkes = sorted(fowlkes_mallow, key=lambda x: x[0], reverse=True)
print("\nRanking based on Fowlkes-Mallows Index:")
for rank, (score, method) in enumerate(sorted_fowlkes, start=1):
    print(f"{rank}. {method} - Fowlkes-Mallows Index: {score}")
```

```
Ranking based on Fowlkes-Mallows Index:
1. Specctral_clustering - Fowlkes-Mallows Index: 0.9723869207977962
2. Bisecting_kmeans - Fowlkes-Mallows Index: 0.9668210291034108
3. random - Fowlkes-Mallows Index: 0.9642248536253374
4. k-means++ - Fowlkes-Mallows Index: 0.9642248536253374
5. average - Fowlkes-Mallows Index: 0.9641790636313344
6. ward - Fowlkes-Mallows Index: 0.953188710705752
7. complete - Fowlkes-Mallows Index: 0.9480596231176764
8. single - Fowlkes-Mallows Index: 0.4986377685134068
9. DBSCAN - Fowlkes-Mallows Index: 0.4777854411290982
```

References

https://scikit-learn.org/stable/modules/clustering.html

https://pytorch.org/hub/pytorch_vision_resnet

https://scikit-learn.sourceforge.net/stable/modules/generated/sklearn.cluster.Ward.html