Bishop's University
Department of Computer Science
CS462/562
Winter 2024 – Course project – Face Image Retrieval

Dataset: Olivetti **faces, AT&T Laboratories Cambridge.**

This dataset contains a set of face images taken between April 1992 and April 1994 at AT&T Laboratories Cambridge. As described on the original website:

"There are ten different images of each of 40 distinct subjects. For some subjects, the images were taken at different times, varying the lighting, facial expressions (open / closed eyes, smiling / not smiling) and facial details (glasses / no glasses). All the images were taken against a dark homogeneous background with the subjects in an upright, frontal position (with tolerance for some side movement)."

The image is quantized to 256 grey levels and stored as unsigned 8-bit integers; the loader will convert these to floating point values on the interval [0, 1], which are easier to work with for many algorithms.

The "target" for this database is an integer from 0 to 39 indicating the identity of the person pictured; however, with only 10 examples per class, this relatively small dataset is more interesting from an unsupervised or semi-supervised perspective.

The original dataset consisted of 92 x 112, while the version available here consists of 64x64 images.

Some of the face images are shown here:

The following code snippet loads the dataset, extracts the face image cube into a 3D matrix `X`, and stores the class labels in vector `Y`.

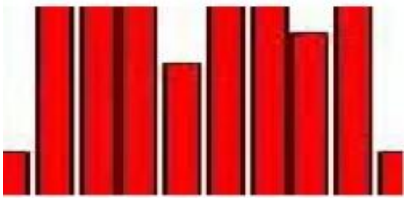
```
# Load the faces datasets
from sklearn.datasets import fetch_olivetti_faces
import matplotlib.pyplot as plt
data = fetch_olivetti_faces()
X = data.images
Y = data.target
```

The shape of the matrix `X` is (400, 64, 64), where 400 represents the total number of face images. These images are divided into 40 classes, each corresponding to a different person, with 10 images per class. Each image has a size of 64x64 pixels. Vector `Y` contains 400 class labels, ranging from 0 to 9, representing the identities of the individuals in the dataset. Each digit in **Y** corresponds to a specific class label.

Objective:
Develop a Python program for retrieving face images (**query images**) from a dataset of images (**target images**). The program comprises two main modules: feature extraction and feature matching. The feature extraction module is designed to apply the Local Binary Pattern (LBP) technique to extract feature maps from the images and then compute histograms of these feature maps. The feature matching module is designed to match histograms by computing the similarity between them. The following figure shows, from left to right, the input face image, the LBP map and the histogram of LBP intensity distribution (credit: Shekhar Karanwal).


Face image


LBP transformation


Histogram of LBP

Feature Extraction:

1. Use existing Python Local Binary Patterns (LBP) algorithm to extract texture features from face images.
2. Implement a function called **`histo_lbp()`** that takes as input a face image, compute its LBP texture features and the histograms of the LBP features. The function returns as output the histogram of LBP. The histogram returned should be normalized. The following is the header of the function.

   ```
   h = histo_lbp(image_name, parameters_of_lbp, n_bins_of_histogram)
   ```

3. Implement a function called **`feature_extraction()`**, which takes as input the face image cube **X** and returns the matrix **H** of the LBP features' histograms. Each row of the matrix **H** represents an LBP histogram of one face image. Add one column to the matrix **H** to indicate the class of the face images. Figure 1 shows an example of a matrix **H** storing the 6-bin histograms of 15 face images. The last column indicates the class number (label) of each image. There are four images in class 0, four images in class 1, three images in class 2, three images in class 3, and one image in class 4.

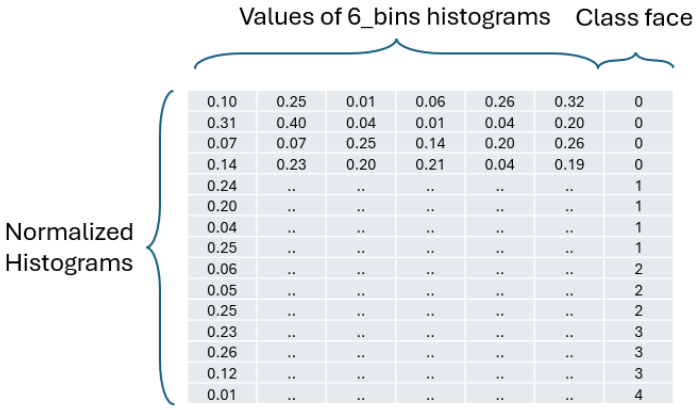


Figure 1. Matrix **H**.

The following is the header of the function **`feature_extraction()`**.

```
H = feature_extraction(X, n_bins_histograms, params_lbp)
# This function calls the above functions.
```

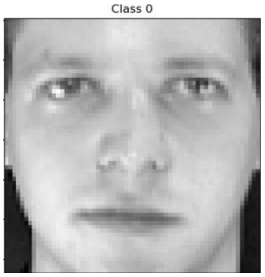
Histogram Matching:

1. Implement a function named **`histo_distance()`** that takes as input the histogram (**P**) of the query face image and the histogram (**Q**) of the target face image in the dataset. This function computes the similarity (**`sim`**) measure between **P** and **Q** based on the Bhattacharyya metric and returns **`sim`**.

```
sim = histo_distance(P, Q)
# P: histogram of the query face image.
# Q: histogram of the target image.
```

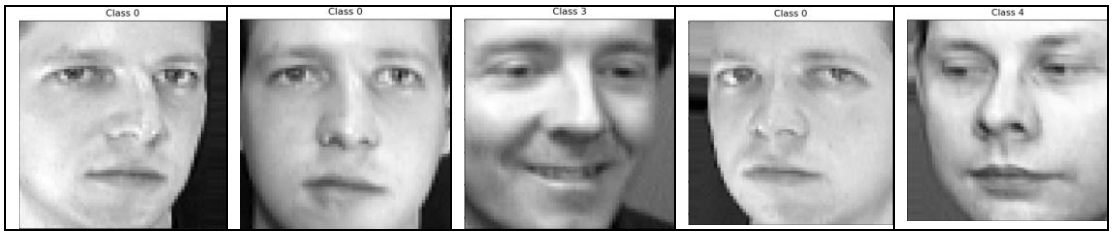
`Main module`

1. Randomly select an image from the dataset **X** to use as a query.
2. Display the query image and indicate its class label in the title. Sample output is shown below.



1. Implement a function called **`image_retrieval()`** that takes the query image and the dataset **X** as input. This function should display the five most similar images to the query image in order of similarity. The query image should be excluded from the target images.

2. The function should compute and return the performance of the method as the number of correct retrieved images divided by 5. A retrieved image is considered correct if it belongs to the same class as the query image.
   - Sample output is shown here:



   In the sample output provided, 3 out of 5 target images are correct, meaning they belong to the same class (class 0) as the query image. Therefore, the performance of the retrieval method is calculated as 3/5, resulting in 60%.

3. Investigate the robustness of the image retrieval method against noise:
   - Add Gaussian noise to images in the dataset **X** with various standard deviation values: std = 0.05, 0.1, 0.3.
   - Repeat the same image retrieval process described above.
   - Display the results for the same query image before and after adding noise.
   - Briefly discuss the effect of the noise on the retrieval process.

Bhattacharyya Coefficient

The Bhattacharyya coefficient is a measure of overlap between two statistical samples or populations. It quantifies the similarity between two probability distributions. Mathematically, the Bhattacharyya coefficient between two discrete probability distributions can be calculated as follows:

$$BC(P,Q) = \sum_i \sqrt{P(i)Q(i)}$$

P and Q represent two discrete distributions obtained by normalizing two distinct histograms. P(i) and Q(i) denote the probabilities associated with the i^{th} bin. The higher the coefficient, the higher the similarity between two histograms.

To normalize a histogram, divide the height of each bin by the total number of values.

For more information about the Bhattacharyya coefficient, read the following tutorial:

https://safjan.com/understanding-bhattacharyya-distance-and-coefficient-for-probability-distributions