Image Similarity from Olivetti Faces Dataset

Group Members:

1. Nihar Samirbhai Patel npatel69@asu.edu
2. Deep Vijaykumar Patel dpatel94@asu.edu
3. Dhruvil Deepakbhai Patel dshah47@asu.edu
4. Vijay Aravindh Viswanathan vviswa19@asu.edu
5. Manikanta Mudara mmudara@asu.edu
6. Pramukh Belam pbelam@asu.edu

Abstract

The original Olivetti faces dataset consists of 400 grayscale images of size 64 x 64. It has 10 images for each of the 40 subjects. For some subjects, the images were taken at different times, varying lighting conditions, facial expressions and facial details. All the images were taken against a dark homogeneous background with the subjects in an upright, frontal position with some subjects facing sidewards to add noise. In this phase of the project, it is expected to use three different feature descriptors namely Color Moments, Extended Local Binary Patterns and Histogram of Oriented Gradients to extract features. The extracted features are than used to obtain similar images based on the query image and the model. The similarity is calculated based on various distance measures. The final task is to compute similar images based on all the models by giving different weightage to models. The output is than compared by human to estimate the accuracy of the models alongside the distance.

Keywords: Feature Extraction, Olivetti Faces dataset, Color Moments, Extended Local Binary Patterns, Histogram of Oriented Gradients.

Introduction

The Olivetti Faces dataset is developed by AT&T Laboratories Cambridge and contains set of face images taken between April 1992 and April 1994. The original dataset consists of 400 grayscale images of size 92 x 112. However, the sklearn version has dimension of 64 x 64 and the pixel values are normalized to the interval [0, 1] rather than [0, 255]. The images are taken against a dark background with different lighting, facial expression, facial details and timing. While majority images are front facing, there are some images which face sidewards to create heterogeneity in the dataset. There are 40 subjects and each subject has 10 images in varying conditions. When importing data from sklearn, three arrays are obtained. The X array consists of 400 vectors of dimension (4096 x 1). The Y array is the target name for each vector in X, hence the dimension is (400 x 1). The IMAGES array is similar to X, but the dimension of each array is (64 x 64) for all 400 images.

There are three feature descriptors which needs to be used in this phase for similarity comparison. Color Moments of any image comprise of three components namely Mean, Standard Deviation and Skewness. The 64 x 64 image is transformed to 64 vectors of size 8 x

8. Now on this 8 x 8 matrix, color moments are applied and hence we get a vector of size 3. So, the overall size becomes 64 x 3, which we reshape into 8 x 8 x 3. The Local Binary Pattern are binary values associated to a pixel by comparing its values to all its 8 neighbors. So, for each pixel in the image, it is replaced by the local binary pattern and the updated image is now used for similarity calculation. The extended LBP uses a different method to calculate binary pattern. Histogram of Oriented Gradients is another method for feature extraction. It takes into account the orientation of change in the pixel values. This implementation uses 9 orientations (8 for directions and 1 for no change) for a cell size of 8 x 8, block size of 2 x 2 and L2 norm of 0.2.

Terminology

The following are the frequently used terms of the implementation.

Pixel: A Pixel is simply a color instance. Based on the color model that's used, a pixel represents the most fundamental block of an image. A pixel can take any value from 0 to 255 for three channels namely Red, Green and Blue. Pixels arranged in 2-D space construct an image.

Gray scale: The Grey scale is a representation of a color instance with just one channel that represents the intensity of the color from white to black scale. The values are of the range 0 (Black) and 255 (White), and it allows for 256 different variations in intensity. It is useful for image processing and related tasks.

Feature Selection: Feature selection is also known as variable selection, attribute selection or variable subset selection. It refers to the process of selecting a subset of relevant features for model construction. It helps to avoid "Dimensionality Curse".

Feature Extraction: Feature Extraction is associated with reducing the number of features in an input object. In machine learning and pattern recognition, feature extraction is an important task and ignoring it can lead to disastrous results. Special care has to be taken while reducing features as removing unique features can lead generalization of the data.

Color Moments: Color Moments refers to three features associated with block (square matrix) of data. The first value is the Mean, second value is Standard Deviation and third is Skewness. These features can be used to calculate similarity between images.

Extended Local Binary Pattern: Local Binary Pattern (LBP) is a representation of an image based on the neighbors of a selected pixel. The algorithm takes each pixel and compares it with the value of all the other pixels at a given radius R and yields a binary representation (encoded as decimal) for each pixel. Hence, a 64 x 64 image returns a 64 x 64 LBP matrix. The histogram returned from LBP function is considered as the LBP feature vector.

Histogram of Oriented Gradients: The histogram of oriented gradients (HOG) is a feature descriptor used in computer vision and image processing for the purpose of object detection. The technique counts occurrences of gradient orientation in localized portions of an image. This implementation uses 9 orientations with cell size of 8 x 8 and block size of 2 x 2.

Similarity Metrics: This is used to calculate similarity between two given data objects. There are various methods for distance measurement and some of them are Euclidean, Earth Movers distance, Pearson Correlation, KL Divergence, Mahanalobis Distance, Cosine Similarity, and others.

Problem Specification

This phase consists of four tasks. The tasks are as follows:

- Task 1: Implement a program which, given an image ID and one of the following models, extracts and prints (in a human readable form) the corresponding feature descriptors. Models = {CM8x8, ELBP, HOG}

- Task 2: Implement a program which, given a folder with images, extracts and stores feature descriptors for all the images in the folder.

- Task 3: Implement a program which, given a folder with images and an image ID, a model, and a value "k", returns and visualizes the most similar k images based on the corresponding visual descriptors. For each match, also list the overall matching score.

- Task 4: Implement a program which, given a folder with images and an image ID and a value "k", returns and visualizes the most similar k images based on **all** corresponding visual descriptors. For each match, also list the overall matching score and the contributions of the individual visual models.

Assumptions

The following assumptions were made while implementing this project:

1. All the images in the folder are of same size and in '.png' format.
2. All the images provided are in grayscale.
3. No images are duplicated.
4. The input image is from the same dataset with same size and format.
5. Feature extraction using inbuilt libraries works accurately.
6. Distance functions imported from libraries works accurately.

Description of the proposed solution/implementation

The proposed solution uses Python and its libraries for computation and storage. The dataset was imported from *sklearn.datasets.fetch_olivetti_faces.* This returns arrays of data, target and images into X, Y, imgs respectively. X is of size (400, 4096), Y of (400, 1) and imgs of (400, 64, 64). The images are normalized to scale [0, 1] rather than [0, 255]. All images are in grayscale. All the libraries and functions are imported in the beginning. Image of all 40 subjects is shown in the output.

Task 1:

This task is to extract and print features for a given image and model in human readable form. For this task three models were implemented namely Color Moments, Extended Local Binary Pattern and Histogram of Oriented Gradients.

Color Moments: A function called *calculate_CM(input_img)* is created which expects a 64 x 64 image as input and returns 8 x 8 x 3 sized array. The input image is sliced into 64 vectors of size 8 x 8. Now for each of the 64 vectors three moments (mean, standard deviation, skewness) are calculated using *numpy* functions and appended to a list. The final output is an array of size 64 x 3 which is resized into 8 x 8 x 3 array.

Extended Local Binary Pattern: A function named *calculate_LBP(input_img)* is developed which expects 64 x 64 image as input and produces an array of LBP features. For this implementation, inbuilt function of *numpy* called *local_binary_pattern()* was used to get features. Radius was set to 1, neighbors to 8, number of bins to 256 with rotational invariant *(ror)* method.

Histogram of Oriented Gradients: A function called *calculate_HOG(input_img)* is devised which takes 64 x 64 image as input and generates feature vector of size 1764. For this implementation, inbuilt function of *numpy* called *hog()* was used to get features. Orientations were set to 9, block size to 8 x 8, cell size to 2 x 2 and norm to L2-Hys.

The function named Task_1() takes image and model as input and generates the features based on model.

Sample output for task 1:

```
The feature descriptor for image-0.png and HOG are
 [0.34298747 0.34298747 0.01171484 ... 0.01057377 0.04454767 0.2046834 ]
```

```
The feature descriptor for image-0.png and CM8x8 are
 [[ 1.47078125e+02  4.15042534e+01 -5.50547098e-01]
 [ 1.96093750e+02  4.51289368e+00 -7.12079078e-01]
 [ 2.03921875e+02  3.69672441e+00  4.03322192e-01]
 [ 2.23531250e+02  6.45408964e+00 -5.34948668e-01]
 [ 2.14203125e+02  8.63709259e+00 -1.89562604e-01]
 [ 2.03359375e+02  4.38878965e+00 -1.24893671e+00]
 [ 2.02687500e+02  5.23770905e+00 -9.83680590e-01]
 [ 1.33718750e+02  4.28921127e+01 -1.74879293e-01]]
```

```
The feature descriptor for image-0.png and ELBP are
 [3.22265625e-02 5.98144531e-02 0.00000000e+00 6.37207031e-02
 0.00000000e+00 6.10351562e-03 0.00000000e+00 1.37695312e-01
 0.00000000e+00 7.32421875e-04 0.00000000e+00 5.37109375e-03
 0.00000000e+00 5.37109375e-03 0.00000000e+00 2.74414062e-01
 0.00000000e+00 6.10351562e-03 0.00000000e+00 4.63867187e-03
 0.00000000e+00 1.70898437e-03 0.00000000e+00 3.66210937e-03
 0.00000000e+00 5.12695312e-03 0.00000000e+00 4.15039062e-03
 0.00000000e+00 7.56835937e-03 0.00000000e+00 1.73339844e-01
 0.00000000e+00 0.00000000e+00 0.00000000e+00 0.00000000e+00
 0.00000000e+00 2.44140625e-04 0.00000000e+00 2.19726562e-03
 0.00000000e+00 0.00000000e+00 0.00000000e+00 2.44140625e-04
 0.00000000e+00 2.44140625e-04 0.00000000e+00 4.15039062e-03
 0.00000000e+00 0.00000000e+00 0.00000000e+00 1.22070312e-03
 0.00000000e+00 2.44140625e-04 0.00000000e+00 2.19726562e-03
 0.00000000e+00 0.00000000e+00 0.00000000e+00 3.90625000e-03
```

Task 2:

This task is to compute all the feature vectors for all the images in the given folder, and store them. In this task, input to the function is path to folder and output is a file named *'features.txt'* saved in the same folder. All the images in the given folder are used to calculate features descriptors and then they are written into a text file.

Sample output for task 2:

```
features - Notepad
File  Edit  Format  View  Help
image-0.png: Color Moments
[[[ 1.47078125e+02  4.15042534e+01 -5.50547098e-01]
  [ 1.96093750e+02  4.51289368e+00 -7.12079078e-01]
  [ 2.03921875e+02  3.69672441e+00  4.03322192e-01]
  [ 2.23531250e+02  6.45408964e+00 -5.34948668e-01]
  [ 2.14203125e+02  8.63709259e+00 -1.89562604e-01]
  [ 2.03359375e+02  4.38878965e+00 -1.24893671e+00]
  [ 2.02687500e+02  5.23770905e+00 -9.83680590e-01]
  [ 1.33718750e+02  4.28921127e+01 -1.74879293e-01]]

 [[ 1.48687500e+02  4.19217567e+01 -1.18269598e+00]
```

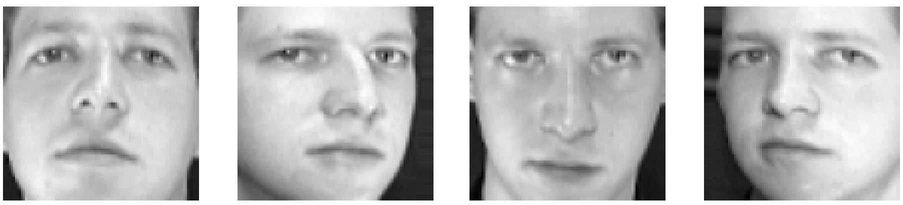
Task 3:

This task was to output most similar *'k'* images from a given folder with their similarity score based on a given model and an input image. After calculating a feature descriptor for the input image, feature descriptors for all images in the folder are calculated for a given model. Now similarity is calculated between input image and all other images in the folder based on model. This list is sorted to give top *k* images based on distance. After thorough analysis, Earth Movers distance was used for comparing features of CM and HOG, while correlation was used for ELBP.

Sample output for task 3:

Set 1, input_img = 'image-0.png', k = 4

```
Model              : CM8x8
Set                : set1
Input Image        : image-0.png
K                  : 4
Rank 1 ---> image-6.png          : 3.1518124541162313
Rank 2 ---> image-8.png          : 3.602674520931225
Rank 3 ---> image-5.png          : 3.938855208687797
Rank 4 ---> image-2.png          : 4.230160387729845
```



```
Model              : ELBP
Set                : set1
Input Image        : image-0.png
K                  : 4
Rank 1 ---> image-6.png          : 0.0010641066735831428
Rank 2 ---> image-4.png          : 0.0023483014619594123
Rank 3 ---> image-2.png          : 0.0030176596182959203
Rank 4 ---> image-3.png          : 0.003063406130106139
```

```
Model              : HOG
Set                : set1
Input Image        : image-0.png
K                  : 4
Rank 1 ---> image-5.png          : 0.004424375571885826
Rank 2 ---> image-2.png          : 0.005033065415461765
Rank 3 ---> image-3.png          : 0.00637070997473491
Rank 4 ---> image-8.png          : 0.006404599830708055
```

Set 2, input_img = ‘image-0.png’, k = 4

```
Model              : CM8x8
Set                : set2
Input Image        : image-0.png
K                  : 4
Rank 1 ---> image-2.png          : 4.230160387729845
Rank 2 ---> image-4.png          : 4.892928305104478
Rank 3 ---> image-1.png          : 6.065992518017445
Rank 4 ---> image-3.png          : 9.014437570025025
```

```
Model              : ELBP
Set                : set2
Input Image        : image-0.png
K                  : 4
Rank 1 ---> image-4.png          : 0.0023483014619594123
Rank 2 ---> image-2.png          : 0.0030176596182959203
Rank 3 ---> image-3.png          : 0.003063406130106139
Rank 4 ---> image-12.png         : 0.0031909608520542B1
```

```
Model              : HOG
Set                : set2
Input Image        : image-0.png
K                  : 4
Rank 1 ---> image-2.png          : 0.005033065415461765
Rank 2 ---> image-3.png          : 0.00637070997473491
Rank 3 ---> image-1.png          : 0.006919707463150785
Rank 4 ---> image-4.png          : 0.011135897654372489
```

Set 3, input_img = ‘image=0.png’, k = 4

```
Model              : CM8x8
Set                : set3
Input Image        : image-0.png
K                  : 4
Rank 1 ---> image-70.png         : 6.317257085030583
Rank 2 ---> image-110.png        : 6.859064387367253
Rank 3 ---> image-60.png         : 8.620595765566605
Rank 4 ---> image-90.png         : 9.78866746558702
```

```
Model              : ELBP
Set                : set3
Input Image        : image-0.png
K                  : 4
Rank 1 ---> image-60.png         : 0.0010330065231424213
Rank 2 ---> image-100.png        : 0.00162204320981707
Rank 3 ---> image-120.png        : 0.0027755385449015346
Rank 4 ---> image-30.png         : 0.0035555642026979806
```

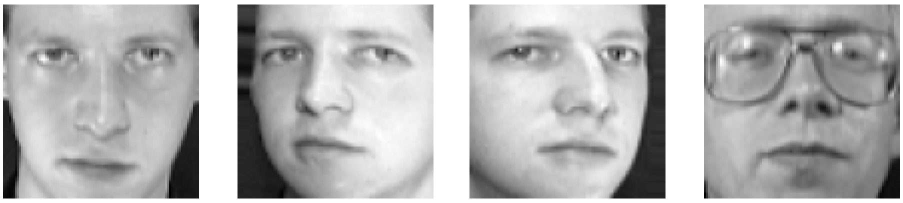
```
Model            : HOG
Set              : set3
Input Image      : image-0.png
K                : 4
Rank 1 ---> image-40.png         : 0.006130899930030032
Rank 2 ---> image-50.png         : 0.007487745525360349
Rank 3 ---> image-100.png        : 0.008935403464432654
Rank 4 ---> image-30.png         : 0.010320816117324608
```

Task 4:

This task was to output most similar *'k'* images from a given folder with their similarity score based on ***all*** models and an input image. After calculating a feature descriptor for the input image, feature descriptors for all images in the folder are calculated for ***all*** models. Now similarity is calculated between input image and all other images in folder for ***all*** models. These similarities are now normalized in range [0,1] using *normalize()* function. Now, weightage is assigned to each feature descriptor such that its sum is 1. For example, CM has weight 0.4, ELBP has weight 0.2 and HOG has weight 0.4. These new weighted distances are now sorted to give top *k* images based on combined distance.

Sample output for task 4:

```
Weights of CM, ELBP, HOG are 0.4, 0.2, 0.4
Set              : set1
Input Image      : image-0.png
K                : 4
Rank 1 ---> image-2.png          : 0.20398707222258006
Rank 2 ---> image-6.png          : 0.20920106607136862
Rank 3 ---> image-8.png          : 0.23247753135997135
Rank 4 ---> image-5.png          : 0.27294604849443954
```



```
Weights of CM, ELBP, HOG are 0.4, 0.2, 0.4
Set                : set2
Input Image        : image-0.png
K                  : 4
Rank 1 ---> image-2.png          : 0.13496785643875087
Rank 2 ---> image-3.png          : 0.20794880963042145
Rank 3 ---> image-4.png          : 0.21360834320645905
Rank 4 ---> image-11.png         : 0.33106463579564466
```

```
Weights of CM, ELBP, HOG are 0.4, 0.2, 0.4
Set                : set3
Input Image        : image-0.png
K                  : 4
Rank 1 ---> image-100.png        : 0.18958372990585884
Rank 2 ---> image-40.png         : 0.19874723380828432
Rank 3 ---> image-50.png         : 0.22532101669521432
Rank 4 ---> image-30.png         : 0.2330556853033822
```

Interface Specification

This file was created in Jupyter Notebook. For running the cell user needs to press (Shift + Enter). Each cell has some functions with necessary comments and headings. This default path was set to my machine. When running on other machine few modifications have to be made for path to folder.

System requirements/installation and execution instructions

The solution is designed to work only with Python 3.6 and above.
This file has *'.ipynb'* extension and needs to be opened in Jupyter Notebook.

The minimum system requirements are:
Processor: 2.2 GHz processor
RAM: 4 GB of RAM
Operating System: Windows (tested)

Requirements:

cv2
os
numpy
pandas
matplotlib
sklearn

Execution steps:

- Library imports, function definitions and Task 0:

  Run the first **8** cells to import the necessary libraries and create helper functions for the remaining tasks of the project. Cell 1 import the libraries. Cell 2 defines a function to slice 64 x 64 images into 64 x 8 x 8. Cell 3 imports the original dataset from sklearn and prints the description of the data. Cell 4 extracts unique subjects from the dataset. Cell 5,6,7, and 8 are the functions which calculate the feature descriptors.

- Task 1:

  Run **all** the cells till Task 1 heading is encountered.
  Run the cell containing function *"def task_1(test_img_path, input_img, model):"*
  Set the parameters listed in the next cell along with the path to folder and execute the function **task_1**.
  The formatted output is printed.

- Task 2:

  Run the cell containing *"def task_2(img_path):"*
  Set the parameters listed in the next cell along with the path to folder and execute the function **task_2**.
  The output feature file is saved in the same folder.

- Task 3:

  Run the cell containing *"def task_3(test_img_path, input_img, model, k):"*
  Set the parameters listed in the next cell along with the path to folder and execute the function **task_3**.
  Images similar to input image are shown along with their similarity score and parameters.

- Task 4:

  Run the cell containing *"def task_4(test_img_path, input_img, k):"*
  Set the parameters listed in the next cell along with the path to folder and execute the function **task_4**.

Images similar to input image are shown along with their similarity score and parameters.

Conclusion

Analysis from this project leads to a conclusion, that different feature extraction methods focus on different aspects of an image and lose some features while comparing. Also, similarity metrics can produce random results based on the data provided to them. Hence, feature selection remains an essential part of any pattern recognition task and requires special care, because removing important features could lead to erroneous results.

Bibliography

https://en.wikipedia.org/wiki/Histogram_of_oriented_gradients
https://en.wikipedia.org/wiki/Local_binary_patterns
https://en.wikipedia.org/wiki/Pearson_correlation_coefficient
https://en.wikipedia.org/wiki/Earth_mover%27s_distance

Appendix

Specific roles of the group members
This phase of the project was performed by each team member individually. The collaboration was limited to the design of the overall solution in terms of functions and sharing of ideas for assigned tasks.