CAPSTONE PROJECT REPORT

Hashing: Querying in Face Datasets

Name - Paleti Samuel Yashaswi
Course - Machine learning and AI
Duration - 24 months
Question - 4

Implement a basic hashing model from scratch that hashes the images. You can use any dataset of few images and can implement a-hash or any other hashing algorithm of your choice. For a-hash, given any images, first resize the image to a suitable size, followed by grayscale conversion of the image. Then mean normalize the image to obtain a binary image, whose sum can be used as a hash value. Using the hash model, encode all the images present inside your directory and then search for images similar to the query image.

In [5]:
```
# Importing the necessary libraries
import os
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import matplotlib.image as mpimg
```

In [8]:
```
# Reading multiple images from a folder
mydir = os.getcwd()
data_dir = os.path.join(mydir,'yalefaces')
#print(data_dir)
imgs = os.listdir(data_dir)
images = []
for file in imgs:
    file_path = os.path.join(data_dir,file)
    img = mpimg.imread(file_path)
    if img is not None:
        images.append(img)
print(len(images))
```

```
166
```

In [9]:
```
# Vectorizing the images and storing it in a list
imgs_vec = []
for image in images:
    row,col = image.shape
    img_vec = image.reshape(row*col)
    img_vec_norm = img_vec/(np.linalg.norm(img_vec))
    imgs_vec.append(img_vec_norm)
print(len(imgs_vec))
print(row*col)
```

```
166
77760
```

n [10]:
```
# Generator Function to generate random unit vectors for Hashing
def genRandomHashVec(m,length):
    hash_vec = []
    for i in range(m):
        v = np.random.uniform(-1,1,length)
        v_ =  v/(np.linalg.norm(v))
        hash_vec.append(v_)
    return hash_vec
```

:
```
# Creating a Image Dictionary using the hash as the keys
image_dict = {}
for i in range(len(imgs_vec)):
    hash_code = LSH(hash_vector,imgs_vec[i])
    str_hash_code = ''.join(hash_code)
    if str_hash_code not in image_dict.keys():
        image_dict[str_hash_code] = [i]
    else:
        image_dict[str_hash_code].append(i)
```

:
```
# Displaying the Hashes
cols_names = ['Hash_codes','Image_Index']
df = pd.DataFrame(image_dict.items(),columns = cols_names)
df.head(30)
```

:

	Hash_codes	Image_Index
0	0001001100	[0, 1, 2, 3, 5, 6, 8, 9, 10, 11, 12, 13, 14, 1...
1	1001001100	[4, 15, 26, 70, 81, 92, 125, 136]
2	0001011100	[7, 40, 84, 106, 111, 112, 117, 118, 120, 121,...
3	0000011100	[29, 161]
4	0101001100	[37, 42, 48]
5	0001001101	[57, 62]
6	0000001100	[73, 139]
7	0001001110	[126, 127]

```
# Getting the keys and values of the Dictionary
keys = list(image_dict.keys())
values = list(image_dict.values())
```

```
# Plotting images with same hash code
igs = [images[i] for i in range(len(images)) if i in values[2]]
fig = plt.figure()
cols = 2
n_images = len(igs)
for n,image in zip(range(n_images),igs):
    ax = fig.add_subplot(cols,np.ceil(n_images/float(cols)),n+1)
    plt.gray()
    plt.imshow(image)
fig.set_size_inches(np.array(fig.get_size_inches())*n_images)
plt.show()
```