Phase #1 Project Report

By

Krima Doshi, Krishna Vamsi Nandamuru, Nisarg Shah, Harshita Verma, Chinmay Sai Krishna Atluri, Eshita Khandelwal

CSE 515 Multimedia and Web Databases

Arizona State University

15th September, 2021

Abstract

Images contain a thousand words. Humans can gain many insights from images, but it is not so simple for computers to get the same. To enable computers to gain such insights, we must first determine and store the features which are an important factor in enabling machines to process this data to gain similar insights. Here, one such task of identifying similar images for a given image has been performed by extracting features like color moments, hog and elbp using the Olivetti face dataset. The similarity results have been evaluated using distance measures like manhattan, cosine, earth movers and L2. It has been observed that the manhattan distance had the best performance overall for this vector space of 400 images.

Keywords: Olivetti face, feature extraction, color moments, hog, elbp, manhattan, cosine, earth movers, L2.

Phase #1 Project Report

Images store a lot of information. In the current digital scenario, the inflow of data, along with images, has vastly increased. Along with the increasing HD quality of image, there is a rise in the amount the space required for storage. Hence, it becomes necessary to efficiently store the data so that it is not only compact but has less latency at query time retrieval.

Compact data can be achieved by using optimum features to characterize these images and form a basis to obtain information from them. From the features available for performing this task, color moments, elbp and hog have been chosen for characterization of images. These features are then compared using various distance metric such as Manhattan, L2, Earth Movers and Cosine. An optimum metric from the ones listed above has been chosen for each feature and the top k most similar images are returned.

The following assumptions have been for the given project:

1. Each image given is frontal.

2. Each image is a 64x64 grayscale image.

Terminologies

Color Moments: This vector contains the color moments of the image. They are used to differentiate images based on some color feature like mean, standard deviation and skewness.

Mean: The average value of a block in the image. Here, 64x64 images have been divided into 8x8 blocks and a mean is calculated on each of these 8x8 blocks

Standard Deviation: Standard deviation in a block is the measure of how different the pixels are from the mean in that block.

Skewness: Skewness in a block measures the asymmetry of the color in that block.

LBP: Local Binary Patterns is a texturing measure for an image. It considers the neighbouring pixels for a pixel A, applies a threshold function by comparing them all with pixel A and assigning a binary value to them and then converting that binary value to decimal and assigning in to a lbp matrix at A.

HOG: Histogram of oriented gradients shows the direction of color intensity in a block. It basically splits the image into 8x8 blocks with 50% overlap, i.e. for an image of 64x64, 14x14 matrix will be formed if block size is 8x8. The gradient and orientation is calculated for each of these blocks. The orientation is calculated using 9 bins, so one bin will measure a range of 20 degrees.

Manhattan distance: Manhattan distance is the sum of the absolute differences of two vectors.

L2 distance: L2 distance is the square root sum of the squared differences of two vectors.
Cosine distance: Cosine distance is the measure of angle between two vectors.

Earth Mover's distance: It is the measure of the weights that need to be shifted in one image histogram to convert it into another.

Proposed Solution

The proposed solution consists of 4 stages – feature extraction, feature storage, feature comparison and returning top k.

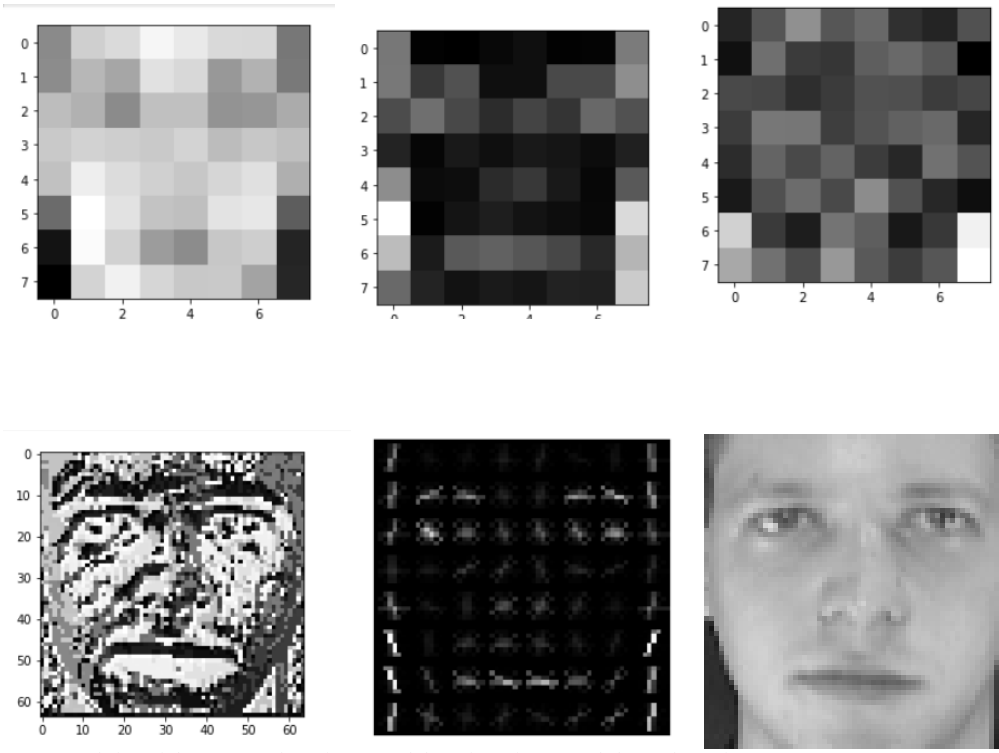
1. First three features, namely color moments, elbp and hog, are extracted from an image.
2. Next, these features are stored in a csv file.
3. After that, different distance functions are applied to determine the best fit for each feature. Consequently, the best distance metric is decided for all features.
4. Lastly, a pipeline is created to return all similar images for an image given its image path, reference images folder, values of k and feature.

They are implemented as follows:

Task 1: Implementation of features onto the given image

This task deals with computing individual features for any one image.

- `get_mean_img`: Calculate the mean of image by dividing the image into blocks of 8x8
- `get_std_dev_img`: Calculate the standard deviation of image by dividing the image into blocks of 8x8
- `get_skew_img`: Calculate the median of image by dividing the image into blocks of 8x8. Then use the Pearson's Coefficient of Skewness to calculate the skewness.
- `get_cm8x8`: Obtain the color moments of the image by calling the above three functions and combining them into one array
- `get_elbp`: Calculate the elbp of the image using radius 1 and neighbours 8.
- `get_hog`: Calculate the hog of the image using 9 orientation bins, 2x2 cell size and 8x8 cells per block using L2-Hys (setting maximum value to 0.2)



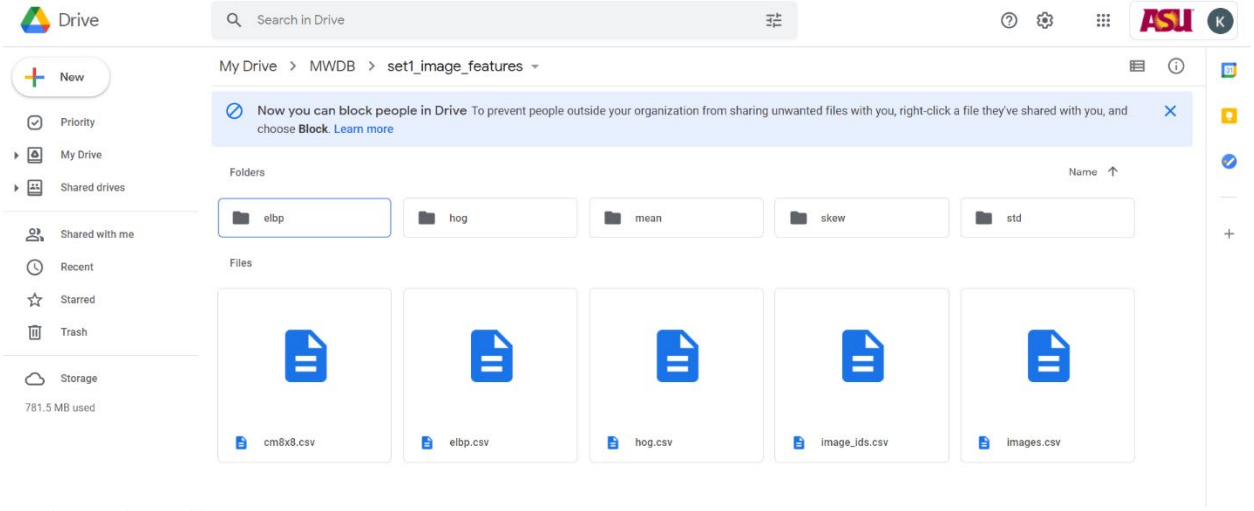
Top(From left to right): Mean of 8x8 image, std dev of 8x8 image and skew of 8x8 image

Bottom(From left to right): LBP, hog and original image

Task 2: Implementation of extraction and storage of feature descriptors for all images in a folder.

This task deals with computing all features for a given folder and storing them in a csv format.

- `save_all_img_features`: Call all the feature creation functions for each image and store all the features in a corresponding csv file in the image features folder



Saved output feature files

Task 3

This task deals evaluating the query on the dataset and returning the top k values for each individual feature.

- `Euclidean`: Obtain the top k vectors with minimum distance using faiss L2 distance
- `cosine`: Obtain the top k vectors with minimum distance using cosine from scipy
- `manhattan`: Obtain the top k vectors with minimum distance using Manhattan absolute distance with numpy
- `top_k_match`: Return the top k vectors with the chosen metric
- `get_image_file`: Return the image files corresponding to the top k indices

Example Output screen of cm8x8 image similarity selection using Manhattan

Task 4

This task deals evaluating the query on the dataset and returning the top k values for all features. It also returns the average weightage of the vectors in the calculation of top k.

- `get_image_file`: Return the image files corresponding to the top k indices

Example Output screen of all similarity selection using Manhattan

Interface specifications

Google Colab is used to create this code. Steps to run the code:

1. Open Google Colab at https://research.google.com/colaboratory/
2. Select “File -> Upload Notebook” and upload this notebook
3. On your drive create a folder named “MWDB” and upload your input folder onto it.
4. Run the cells one by one. One of the cells asks you to connect your notebook to your drive, simply click on that link and sign in.
5. Input the full path (if loading from drive start with “/content/drive/MyDrive/MWDB”) when asked for folder and image and continue running all the inputs

Example input:

```
Requirement already satisfied: faiss-cpu in /usr/local/lib/python3.7/dist-packages (1.7.1.post2)
Enter path of images directory:/content/drive/MyDrive/MWDB/set1/
Enter value of k:4
Enter input image path:/content/drive/MyDrive/MWDB/set1/image-0.png
Enter feature to compare:cm8x8
```

6. Your output will be shown on screen as well as saved in a result folder

System requirements

The technology used for this project is Google Colab. Google Colab is an online platform that allocates the user a kernel to run their python notebook and save/link it to google drive. The basic version needed for this project is available online for free and just needs a gmail account to use. So, all that is required is a working laptop with a good internet speed, a browser to run the notebook on, a gmail account and your google drive.

The libraries used in this project are as listed below:

- **Sklearn.datasets[11]** Python machine learning dataset library containing Olivetti faces data
- **Numpy[7]** Python library for arrays and matrices with optimum time-efficient functions to apply on them
- **Pandas[8]** Python library that deals with dateframes for csvs, joins and concatenations
- **os** Python library used for file functions used in the code

- **matplotlib.pyplot[10]** Python graph plot library for images and graphs
- **skimage.feature.local_binary_pattern**[9] Python machine learning library for lbp
- **skimage.feature.hog**[9] Python machine learning library for hog data

- **google.colab.drive** Python library to connect drive storage to google colab session
- **PIL[12]** Library to quickly load and save images
- **Faiss[13]** Facebook similarity search library for L2 metrics and
- **Glob** Library for accessing all images in a particular folder
- **scipy.spatial.distance.cosine**[6] Distance metric calculation function for cosine distance

Related Work

Vadivel et al.[1] compared 4 distances: Manhattan distance, Euclidean distance, Vector Cosine Angle distance and Histogram Intersection distance on a huge image dataset and arrived at the conclusion that Manhattan performed better for that data. Ponnmoli et al.[2] further verified this fact using image segmentation and state Manhattan as a better metrics not only because of its high accuracy but also because of its lesser dimensionalilty when compared to Euclidean. Ahonen et al.[5] explain how LBP is applied on images and prove its recognition rate higher than PCA MahCosine. Ahonen et al.[3] shows how the performance of cosine with lbp outperforms all considered methods even against different lighting conditions.

Conclusion

Image source folder	Feature used	result image	Distance
set1	all	task4/image-0	0
set1	all	task4/image-7	1312.948
set1	all	task4/image-2	1461.191
set1	all	task4/image-6	1651.13
set1	cm8x8	task3/image-0	1.04E-06
set1	cm8x8	task3/image-2	5.016234
set1	cm8x8	task3/image-7	9.303502
set1	elbp	task3/image-0	2.39E-05

set1	elbp	task3/image-7	1090.129
set1	elbp	task3/image-2	1226.224
set1	elbp	task3/image-6	1387.851
set1	hog	task3/image-0	1.08E-05
set1	hog	task3/image-7	213.515
set1	hog	task3/image-2	229.9512
set1	hog	task3/image-6	256.3038
set2	all	task4/image-0	0
set2	all	task4/image-2	1461.191
set2	all	task4/image-4	1753.002
set2	all	task4/image-1	1765.695
set2	cm8x8	task3/image-0	1.04E-06
set2	cm8x8	task3/image-2	5.016234
set2	cm8x8	task3/image-10	11.62216
set2	cm8x8	task3/image-4	11.95799
set2	elbp	task3/image-0	2.39E-05
set2	elbp	task3/image-2	1226.224
set2	elbp	task3/image-4	1441.329
set2	elbp	task3/image-3	1444.706
set2	hog	task3/image-0	1.08E-05
set2	hog	task3/image-2	229.9512
set2	hog	task3/image-1	285.813
set2	hog	task3/image-4	299.7149
set3	elbp	task3/image-0	2.86E-06
set3	elbp	task3/image-0	2021.718
set3	elbp	task3/image-70	2024.235
set3	elbp	task3/image-10	2069.78
set3	hog	task3/image-0	9.66E-06
set3	hog	task3/image-0	414.8085
set3	hog	task3/image-70	428.0601
set3	hog	task3/image-110	447.777
set1	cm8x8	task3/image-6	6.97522

As seen in the confusion matrix above, the similarity search returns the image itself with the minimum distance since it is present in the source folder.

	euclidean	cosine	manhattan
mean	48.5	49.58	47.87
std	46	45.41	49.23
skew	35.28	36.38	38.72
cm8x8	51.09	52.03	52.23
elbp	48.20	48.31	50.78
hog	54.36	57.04	55.17

all	53.4	55.3	56.55

The above matrix has been calculated on the given olivetti faces by taking the average of results of all the images with k=10. It has been observed that Manhattan and Euclidean perform better for all the features overall. Additionally, Manhattan has lesser dimensions than Euclidean so it can be said a better choice. It has also been observed that cosine performs significantly better with hog vectors.

Biblography

[1] Vadivel, A. K. M. S. S. A., A. K. Majumdar, and Shamik Sural. "Performance comparison of distance metrics in content-based image retrieval applications." *International Conference on Information Technology (CIT), Bhubaneswar, India*. 2003.

[2] Ponnmoli, K. M., and Dr S. Selvamuthukumaran. "Analysis of Face Recognition using Manhattan Distance Algorithm with Image Segmentation." *International Journal of Computer Science and Mobile Computing* 3.7 (2014): 18-27.

[3]Ahonen, Timo, Abdenour Hadid, and Matti Pietikäinen. "Face recognition with local binary patterns." *European conference on computer vision. Springer, Berlin, Heidelberg*, 2004.

[4]Sinha, Pawan, and Richard Russell. "A perceptually based comparison of image similarity metrics." Perception 40.11 (2011): 1269-1281.

[5]Ahonen, Timo, Abdenour Hadid, and Matti Pietikainen. "Face description with local binary patterns: Application to face recognition." IEEE transactions on pattern analysis and machine intelligence 28.12 (2006): 2037-2041.

[6] Pauli Virtanen, Ralf Gommers, Travis E. Oliphant, Matt Haberland, Tyler Reddy, David Cournapeau, Evgeni Burovski, Pearu Peterson, Warren Weckesser, Jonathan Bright, Stéfan J. van der Walt, Matthew Brett, Joshua Wilson, K. Jarrod Millman, Nikolay Mayorov, Andrew R. J. Nelson, Eric Jones, Robert Kern, Eric Larson, CJ Carey, İlhan Polat, Yu Feng, Eric W. Moore, Jake VanderPlas, Denis Laxalde, Josef Perktold, Robert Cimrman, Ian Henriksen, E.A. Quintero, Charles R Harris, Anne M. Archibald, Antônio H. Ribeiro, Fabian Pedregosa, Paul van Mulbregt, and SciPy 1.0 Contributors. (2020) SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python. Nature Methods, 17(3), 261-272.

[7] Harris, C.R., Millman, K.J., van der Walt, S.J. et al. Array programming with NumPy. Nature 585, 357–362 (2020). DOI: 0.1038/s41586-020-2649-2.

[8]McKinney W, others. Data structures for statistical computing in python. In: Proceedings of the 9th Python in Science Conference. 2010. p. 51–6.

[9] Van der Walt S, Sch"onberger, Johannes L, Nunez-Iglesias J, Boulogne, Franccois, Warner JD, Yager N, et al. scikit-image: image processing in Python. PeerJ. 2014;2:e453.

[10] Hunter JD. Matplotlib: A 2D graphics environment. Computing in science & engineering. 2007;9(3):90–5.

[11] Pedregosa F, Varoquaux, Ga"el, Gramfort A, Michel V, Thirion B, Grisel O, et al. Scikit-learn: Machine learning in Python. Journal of machine learning research. 2011;12(Oct):2825–30.

[12] Clark A. Pillow (PIL Fork) Documentation [Internet]. readthedocs; 2015. Available from: https://buildmedia.readthedocs.org/media/pdf/pillow/latest/pillow.pdf

[13] Hervé Jegou, Matthijs Douze, Jeff Johnson, Faiss: A library for efficient similarity search, 2017. https://engineering.fb.com/2017/03/29/data-infrastructure/faiss-a-library-for-efficient-similarity-search/