Phase #2 Project Report

By

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

CSE 515 Multimedia and Web Databases

Arizona State University

20th October, 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 #2 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

<FILL IN YOUR PSUDOCODES FOR PHASE 2>

# Design Decisions

<FILL IN YOUR DESIGN DECISIONS FOR PHASE 2>

**Interface specifications**

<FROM KRISHNA’S REPORT >

**System requirements**

<ADD FIRST PARAGRAPH FOR SYSTEM AND PYTHON>

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**

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 &amp; 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/