Analysis of the 11k Hands dataset

Group Members

1. Tarun Chinmai Sekar / tsekar@asu.edu
2. Utkarsh Agarwal / uagarwa5@asu.edu
3. Dhruv Patel / dpatel49@asu.edu
4. Satyak Patel / smpate12@asu.edu
5. Harsh Virani / hvirani@asu.edu
6. Jaykumar Vaghasiya / jvaghasi@asu.edu

Abstract

The 11khands dataset is a collection of 11076 hand images from a collection of 190 subjects. The dataset features hands in both the dorsal and palmar positions against a uniform white background. In this phase of the project, the dataset is subject to four different feature extraction techniques. The extracted features are then used to compute similar images. Each team member independently worked on implementing two of the four different feature extraction techniques. I worked on implementing the Color Moments and Local Binary Patterns features. After feature extraction is completed, the extracted features are then used in calculating distance scores, which are then used to find similar images. The objective of this phase is to obtain 'k' images similar to a test image based on any of the two implemented feature models. The reported "similar" images are then compared visually to the test image to gauge the accuracy of similarity comparison.

Keywords

Local Binary Patterns, Color Moments, Feature Extraction, 11kHands.

Introduction

The 11khands dataset is a comprehensive dataset of 11076 hand images from 190 diverse subjects created to aid in gender recognition or biometric identification research. The hand images are of a 1200 x 1600 resolution, and have a uniform white background. The images are all taken from a similar angle and distance from the camera. The dataset also has metadata associated with it, which has a number of features identifying images such as - accessories, modifications present on the hand (nail polish etc), gender, age, skin color etc. The metadata is not used in this phase of the project.

There are two different features that are extracted from the images, and used for similarity comparison. They are - Color Moments and Local Binary Patterns. The Color Moments comprise of the three standard moments of the given image, such as the Mean, Standard Deviation and Skew. The image is converted into a YUV space before calculating the moments. The Local Binary Patterns are binary encoded values generated by comparing the neighbors of a pixel and associating binary values to the pixel values. The Local binary patterns yield an image with the pixel values replaced with the LBP values, and the LBP values are used for similarity calculation. The image is converted into a grayscale image before computing LBP.

The features of all images in the dataset are extracted, and then these extracted features are used to compute the similarity/distance scores given an index image. The similarity/distance function varies based on the feature chosen.

Terminology

This section describes the most often used terms and provides background on some of the techniques used in this phase.

Pixels

A Pixel is a color instance. Based on the color model that's used, a pixel represents the most fundamental block of an image. Pixels arranged in the form of a 2-D vector construct an image.

RGB Space

The RGB Space is a representation of a Color instance using three channels - Red, Green and Blue. Each channel has values ranging from 0 to 255. Using three channels, and 256 values for them allows for the representation of 16M colors. The RGB space is primarily used for displaying images digitally.

GreyScale Space

The GreyScale space is a representation of a Color instance with just one channel that represents the Intensity of the color in a grey scale - White to black scale. The values are of the range 0 and 255, and it allows for 256 different variations in intensity. It is useful for image processing.

YUV Space

The YUV space represents a Color instance with three channels - Y, U, and V and is a direct transformation from the RGB space. The transformation is as follows.

$$Y = 0.299R + 0.587G + 0.114B$$

U = 0.492 (B-Y)
V = 0.877 (R-Y)

The Y channel contains the intensity of the light in the image, and the U and V channels contain the color information. The advantage of YUV space is that it allows for efficient storage and compression of the image, by allocating less space to the U and V channels without any human perceptible loss of clarity.

Color Moments

The color moments are the first, second and third moments of the three channels under consideration. Calculating the color moments allows one to derive simple similarity/distance functions to compute the similarity and distance of images. The color moments are the Mean, Variance and Skew of each channel. Hence, an entire image can be represented as a vector of 9 elements. Color moments are typically calculated after converting the given image into a YUV space.

LBP

Local Binary Patterns, or LBP is a representation of an image, based on its neighbors. The algorithm works by taking each pixel and comparing it with the value of the other pixels at a given radius and arriving at a binary representation (encoded as decimal) for each pixel. Hence, a 100x100 image yields a 100x100 LBP matrix. The histogram of the LBP array is considered as the LBP feature vector.

Histogram

A histogram is a graphical representation of data, classified into buckets. Usually, histograms are calculated by taking a set size of bins, and classifying the given data into those bins. This yields a representation of the distribution of the data in the dataset.

Distance Score

The distance score is a discrete value that is computed by applying a distance function on two vectors. The distance score is a scale on which the similarity of the image is represented on and smaller the score is, the more similar the images are. The distance score in this phase of this project is computed by measuring the euclidean distance between two vectors.

Euclidean Distance

Euclidean distance is defined as the L2 norm of two vectors, and is calculated as follows:

$$d(p, q) = \sqrt{\sum_{i=1}^{n} (q_i - p_i)^2}$$

Problem Specification

This phase of the project comprises three distinct tasks. The tasks are as follows

1. Task 1: Given an image ID, extract the color moments and the LBP feature and print the extracted feature descriptors.
2. Task 2: Given a path to the directory containing all the images, compute the feature descriptors of all the images in the directory, and store the feature descriptors.
3. Task 3: Given a query image id, and a value “k”, return the k most similar images to the given query image in the database.

Assumptions

The following assumptions are made during the implementation of this phase of the project.

1. All images in the directory are of the same size and resolution
2. The loss introduced by the transformations are acceptable for the project.
3. Extracted features are a good representation of the data set
4. When two images have the same distance/similarity score, then the image with the numerically lower image ID is chosen as the most similar one.

Description of the proposed solution/implementation

The proposed solution uses MongoDB as the datastore for the features and as an intermediate storage area for computation. The MongoDB datastore was chosen due to its flexible schema and ease of use. A MongoDB database called - “mwdb_project” and a collection called “image_features” is created beforehand for the storage of the images and the features. The schema of the collection is defined as follows

{

“image_id” : string, #Filename of the image without the extension

“image_path” : string, #Path to the image on the disk

“ColorMoments” : array(Float) # the array of floating point numbers representing the color moments of the image**.**

“LBP” : array(Float) # the array of floating point numbers representing the lbp feature of the image.

“query_image_id”: string, # The query image ID to compute the similarity against.

“distance_score” : float, # The distance between the given image & query image.

}

This implementation also leverages the parallel processing capabilities of Python using the PoolManager() interface for the second and third tasks. The parallel processing interface is used to speed up computation, and to leverage all cores of the host system.

Task 1

This task requires the extraction of features from a single image in the dataset, and displaying the extracted features. The two models implemented in this task are the Color Moments and Local Binary Patterns.

Color Moments

The image is first read from the path on disk into a numpy array using the cv2 library. The image is then converted into a YUV image using the cv2 library's cvtColor method. This converts the three channels (RGB) of the image, into three channels (YUV). This converted image array is then split into 100 x 100 windows. The input image size is 1200 x 1600 consisting of 3 channels, and splitting it into 100 x 100 windows yields 192 such windows for each channel. For each window in the list of 192 x 3 windows, the first, second and third moments (mean, variance and skew) are calculated using standard numpy library functions, and appended into a list. The color moments will comprise of 192 x 3 x 3 values for each image. The output list is displayed, and also inserted into the database.

Local Binary Patterns

The image is read from the path on disk into a numpy array using the cv2 library and then converted into a grayscale image using the cv2 library's cvtColor method. This transformation, the three channels of a 1200 x 1600 RGB image are converted into a 1200 x 1600 image with a single channel. The local binary patterns are computed after splitting the given image into 100 x 100 windows. This splitting yields 192 such windows. The local binary patterns of the 100 x 100 window is calculated using scikit-image's local_binary_patterns library method that yields a 100 x 100 image of local binary patterns. After experimenting with a number of values, and comparing the outputs visually, the value for the radius and points is chosen as 1, 3. The histogram of the resulting LBP is computed with the bin size set to 256 and range of values to 0, 255. These values are chosen in order to obtain histograms of the same shape for all images. These histograms are printed on the screen and inserted into the database.

Task 2

The objective of task 2 is to compute the feature vectors for all the images in the given folder, and store them. In this task, the paths of all images in the given directory are computed, and stored in a list.

The list of paths to the images is parallely processed by the methods defined in task 1 to extract features for a single image, and the resulting feature vectors are stored in the database.

Task 3

The objective of this task is to compute the “k” most similar images to a given query image “q”. The similarity can be calculated by multiple different distance metrics, including the manhattan distance, euclidean distance, the mahalanobis distance etc. After experimenting with different distance metrics, and comparing the results visually, and taking into account the run time, the euclidean distance was chosen as the distance function to use, due to its relatively fast runtime and high visual accuracy.

This task also leverages parallel processing heavily as follows. First, the given query image_id is set in all the rows of the table, and the distance score is set to -1. Then, a list of python dictionaries is fetched from the database containing the fields - image_id and query_image_id. The PoolManager is again leveraged to parallely iterate over the list of dictionaries and run a helper method that computes the distance.Based on the model used, the distance function varies slightly in its implementation.

ColorMoments

The input to the distance function is two vectors 192 x 9 elements, and the distance is computed for the entire image as a whole, by applying the euclidean distance for the three moments across the three channels.

LBP

The input to the distance function is two histograms of length 192 x 256, and the distance is computed for the entire image as a whole, by applying the euclidean distance to each bin of the histogram, across the 192 windows.

The computed distance scores are then updated in the distance_score column of the db table. The lesser the distance score, the greater the similarity.

Once the parallel methods are done processing, the top “k” images are returned using standard database select , order by and limit methods, (i.e) - the mongo db equivalent of

the query “select * from images ordered by distance_score asc limit k” is run to obtain the results.

Interface specifications

The solution uses standard python command line parsing utilities to extract the command line arguments passed to the solution such as the path to the image directory, the query image id, the value of “k” etc.

The sample outputs for task 1 - the feature vectors are present in the outputs/Task1 directory of the submission. The files are named in the format <ImageID> - <ModelName> - <Features>.txt.

The sample outputs of task 3 are present in the outputs/Task3 directory of the submission. The files are named in the format - <QueryImageId>-<Model>-<K>.png

System requirements /installation and execution instructions

The solution is designed to work only with Python 3.6+.
The minimum system requirements are:
Processor: 2.2Ghz processor
RAM: 4 GB of RAM
Operating System: macOS or linux (tested only on these platforms)
Python: 3.6

The recommended system requirements are:
Processor: 2.2Ghz processor with at least 4 cores
RAM: 16 GB of RAM
Operating System: macOS or linux
Python: 3.7

The solution also requires the installation of mongoDB, and the creation of a database named “mwdb_project” in the database.
The following python libraries are required to run the program.

1. opencv-python
2. scipy
3. pymongo
4. skimage
5. Scikit-image
6. matplotlib

Alternatively, after installing python 3.7 and mongoDb, one can run the bootstrap.sh shell script to bootstrap and prepare the system.

To run the solution, the following invocations are used.

Task 1
./phase1.py --image_id <ImageID> --model <MODEL NAME - CM/LBP> <PATH TO FOLDER> 1

Task 2
./phase1.py <PATH TO FOLDER> 2

Task 3
./phase1.py --image_id <ImageID> --model <MODEL NAME - CM/LBP> --k <Value of K> <PATH TO FOLDER> 3

Related work
In [1] the Local Binary Patterns of the 11k hands datasets are used as one of the features analysed by the SVM and the layers of Convolutional Neural Network.

Conclusions

From the analysis conducted in this study, it can be seen that different feature models contain varying amounts of information, and that loss of information plays a vital role for similarity comparisons. Using different models sometimes yield different results, and the differences are due to the inherent loss caused by the model chosen.

Bibliography

1. Afifi, Mahmoud. "11K Hands: gender recognition and biometric identification using a large dataset of hand images." *Multimedia Tools and Applications* 78.15 (2019): 20835-20854.
2. https://en.wikipedia.org/wiki/Color_moments
3. https://en.wikipedia.org/wiki/Local_binary_patterns

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 and sharing of ideas.