Clothing Image Retrieval for Smarter Shopping

(Team Data Scientists)
Shivangi Goel(201505532)
Tanushri Sharma(201505518)
Vanaja Penumatsa(201505505)

Scope:

The goal of this project is to use image processing to aid in shopping. Our application allows the user to submit a photo of an article of clothing that they are interested in, and that will then return a list of similar items using image retrieval techniques. This helps the user explore their options and decide exactly what they want.

Challenges:

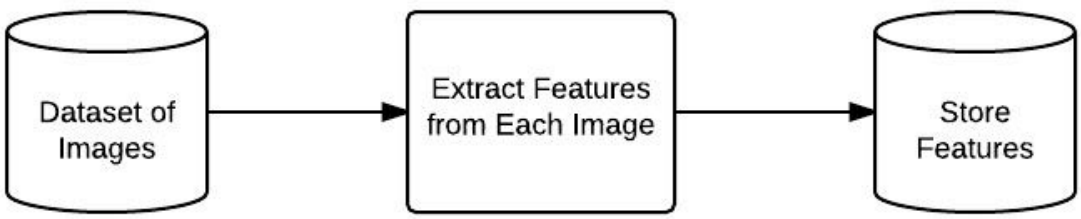
we're all familiar with text based search engines such as Google, Bing, and DuckGoGo — you simply enter a few keywords related to the content you want to find (i.e., your "query"), and then your results are returned to you. But for image search engines, things work a little differently — you're not using *text* as your query, you are instead using an *image*.Here the challenge is how to quantify the contents of an image to make it searchable.

Goal:

Image search engines that quantify the contents of an image are called **Content-Based Image Retrieval** (CBIR) systems. Our goal is to build such system using various image processing and clustering techniques.

Steps:

1. Define image descriptor: At this phase we need to decide what aspect of the image we want to describe. Are we interested in the color of the image? Texture of the Image? Or the shape of an object in the image.
2. Indexing your dataset: Now that we have our image descriptor defined, our job is to apply this image descriptor to each image in our dataset, extract features from these images, and write the features to storage (ex. CSV file, RDBMS, Redis, etc.) so that they can be later compared for similarity.



3. Define similarity metric: Now we have a bunch of feature vectors. But how are we going to compare them? Popular choices include the Euclidean distance, Cosine distance, and chi-squared distance, but the actual choice is highly dependent on (1) your dataset and (2) the types of features you extracted. We particularly used chi-squared on our image histogram representation.
4. Searching: The final step is to perform an actual search. A user will submit a query image to our system and our job will be to (1) extract features from this query image and then (2) apply the similarity function to compare the query features to the features already indexed. From there, you simply return the most relevant results according to the similarity function.

Feature Extraction:

We have extracted the following features:

- ❏ Color Extraction
- ❏ Texture extraction
- ❏ SIFT feature extraction

(i)Color Extraction:

Images that have similar color distributions will be considered relevant to each other. By utilizing a color histogram as our image descriptor, we'll be we'll be relying on the *color distribution* of the image. Because of this, we have to make an important

assumption regarding our image search engine. Our image descriptor is a color histogram in the RGB color space.

Histograms are used to give a (rough) sense of the density of pixel intensities in an image. Essentially, our histogram will estimate the probability density of the underlying function, or in this case, the probability *P* of a pixel color *C* occurring in our image *I*.

It's important to note that there is a trade-off with the number of bins you select for your histogram. If you select *too few bins*, then your histogram will have less components and unable to disambiguate between images with substantially different color distributions. Likewise, if you use *too many bins* your histogram will have many components and images with very similar contents may be regarded and "not similar" when in reality they are.

Histogram is one of the representations to quantify the contents an image. We save the histogram representation of all the images in a file. And they are compared with test image histogram representation using chi-square distance.
Chi-SquareDistance : sqrt(sum(abs((a-b).^2)))

(ii)Texture Extraction:

Problem with color:

Even if images have dramatically different contents, they will still be considered "similar" provided that their color distributions are similar.

What is Texture: Image texture gives us information about the spatial arrangement of color or intensities in an image or selected region of an image.

The first step in constructing the texture descriptor is to convert the image into grayscale. For each pixel in the grayscale image, we select a neighborhood of size *r* (8) surrounding the center pixel. A LBP(Local Binary Patterns, visual descriptor) value is then calculated for this center pixel and stored in the output 2D array with the same width and height as the input image.

We need to calculate the LBP value for the center pixel. We can start from any neighboring pixel and work our way clockwise or counter-clockwise, but our ordering must be kept *consistent* for all pixels in our image and all images in our dataset. Given a *3 x 3* neighborhood, we thus have 8 neighbors that we must perform a binary test on. The results of this binary test are stored in an 8-bit array, which we then convert to decimal, like this:

The last step is to compute a histogram over the output LBP array. Since a *3 x 3* neighborhood has *2 ^ 8 = 256* possible patterns, our LBP 2D array thus has a *minimum value of 0* and a *maximum value of 255*, allowing us to construct a 256-bin histogram of LBP codes as our final feature vector.

(iii)SIFT feature extraction:

Scale-invariant feature transform is an algorithm in computer vision to detect and describe local features in images. For any object in an image, interesting points on the object can be extracted to provide a "feature description" of the object.

This description, extracted from a training image, can then be used to identify the object when attempting to locate the object in a test image containing many other objects. To perform reliable recognition, it is important that the features extracted from the training image be detectable even under changes in image scale, noise and illumination. Such points usually lie on high-contrast regions of the image, such as object edges.

SIFT keypoints of objects are first extracted from a set of reference images and stored in a database. An object is recognized in a new image by individually comparing each feature from the new image to this database and finding candidate matching features based on Euclidean distance of their feature vectors. From the full set of matches, subsets of keypoints that agree on the object and its location, scale, and orientation in the new image are identified to filter out good matches.

Clustering:

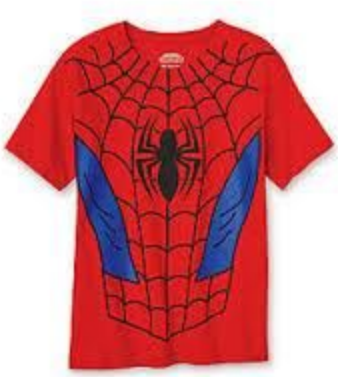
1. First we are dividing each training image into 12 patches (4 X 3)
2. Then for each patch we are calculating the SIFT feature that is a 128 X 1 vector and storing them in a file.
3. Then over each patch of each training image we are applying k-means with k=100. Thus each training image can be represented by a 12 X 1 vector. The i^{th} position of this vector represents the cluster number of the i^{th} of image. In short, each training image is represented by 12 words, each word is a number from 0-99. (Bag of Words)
4. When the test image comes, we first make its background white (same as training images) by using edge detection techniques so that, the background color does not impact the color histogram of image .
5. Then we calculate the color and feature descriptors of image using the process explained before. These descriptors are compared with the training

images using chi-squared distances and a list containing the comparison results is prepared.

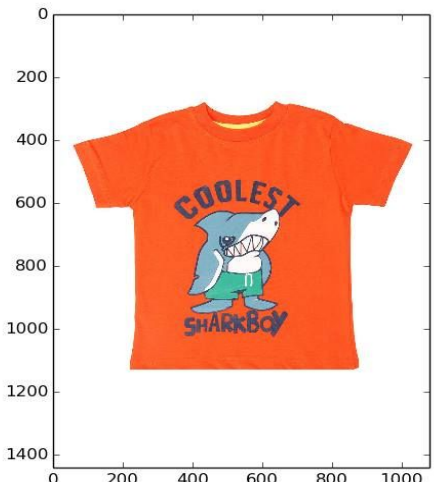
6. After this we perform steps 1-3 over the test image and compare the 12 X 1 vector obtained with the training images bag of words.
7. The results from step 5 and 6 are combined to give top results.

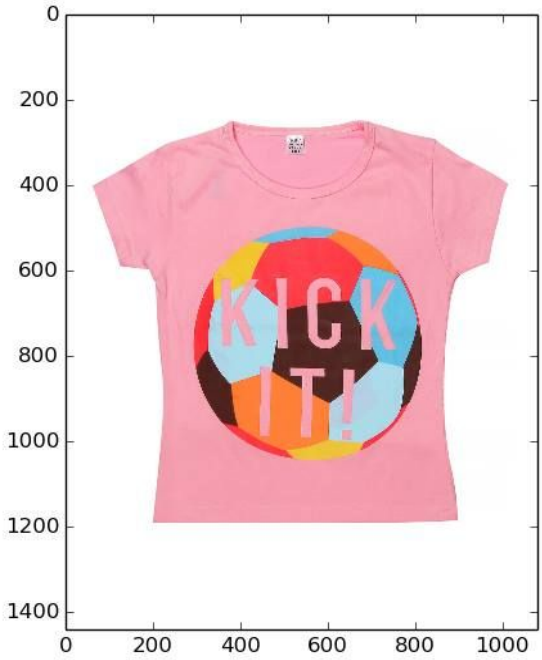
Results:

Input Image:



Output:





Future Work:

- ➔ In color extraction, RGB color space fails to mimic how humans perceive color instead of RGB color space HSV color space can be used.
- ➔ Segmentation can be done to separate foreground from background.

References

YouTube Link:

https://www.youtube.com/watch?v=KQcgBmYrSbs&feature=youtu.be

GitHub Link:

https://github.com/TanushriSharma/Clothin-Image-Retrieval.git