Vision Image Processing

Rachel Rea, Tibor Krols, Ninell Oldenburg, Marie Mortensen

12th January 2022

1 Introduction

In this assignment a Content Based Image Retrieval (CBIR) system is implemented and described. For that the images from the CalTech 101 image database `http://www.vision.caltech.edu/Image_Datasets/Caltech101` are used. This database contains 101 image categories, where each category has approximately 50 pictures. The categories include all sorts of objects and animals, e.g. airplanes, dolphins and faces.
To create the CBIR system SIFT descriptors are extracted from the images with OpenCV's SIFT. With these SIFT descriptors a codebook is generated. Based on the codebook, similar images can be retrieved. The images will be retrieved based on different distance/similarity measures including euclidean distance, Bhattacharyya distance, common words similarity and TF-IDF similarity.
The performance of the retrieval will be measured as a percentage where the top retrieved image is of the same category as the query image. Besides that the percentage of whether an image of the same category as the query image is included in the top 3 retrieved images is calculated.

2 Results

2.1 Codebook

We initially test the program on a category size of 5 and cluster the descriptors into k=200 based on empirical tests where 200 revealed the highest rank and number of correct predictions as seen in table 2.1. The performance is based on the retrieval with the Euclidean distance measure. In the final evaluation, we'll alternate that value.

After we've initialized all global variables in the `__init__()`, we define a function, `load_images()` that goes through n amount of folders and adds a sub-list of images as well as their filenames from the folder to a list. Afterwards, we firstly defined our own `train_test_split()` function for testing on a lower number of images in both. It selects a specified amount of images (we tested on 5 and 10) from each category and assigns them to a train and test set respectively (lines 103-121). Later we replaced that function by just calling the `train_test_split` function the `scipy`-package comes with as it has the more advances functionality of splitting the set percentage-wise.

Following the introduction in `https://docs.opencv.org/3.4/da/df5/tutorial_py_sift_intro.html`, the images are converted to grayscale, and the sift 'engine' is defined, key points are detected with it and from this the descriptors are computed. This is performed for every image in each category for the train set and all the descriptors are concatenated into the same list. Afterwards, the descriptors are normalized by the standard-deviation and the codebook is created using

K-values	% position in top-3	% position in top-1	Mean rank
100	0.7	0.48	2.42
200	0.82	0.52	1.64
300	0.78	0.52	2.1
400	0.7	0.5	2.38
500	0.7	0.5	2.4

Table 2.1: Performance using different cluster values. Based on retrieval with Euclidean distance.

`kmeans` from `scipy.cluster.vq`. This specific `kmeans` function is chosen based on its efficient clustering (as seen in figure 2.1) and because it directly returns both a codebook which is what is needed in our case.

The descriptors are classified to their closest prototype/cluster center with `vq()` and the BoW is created by counting the occurrence of every prototype in each image. The histogram and additional information (group, image category and filename) are combined in a sub-array and saved.

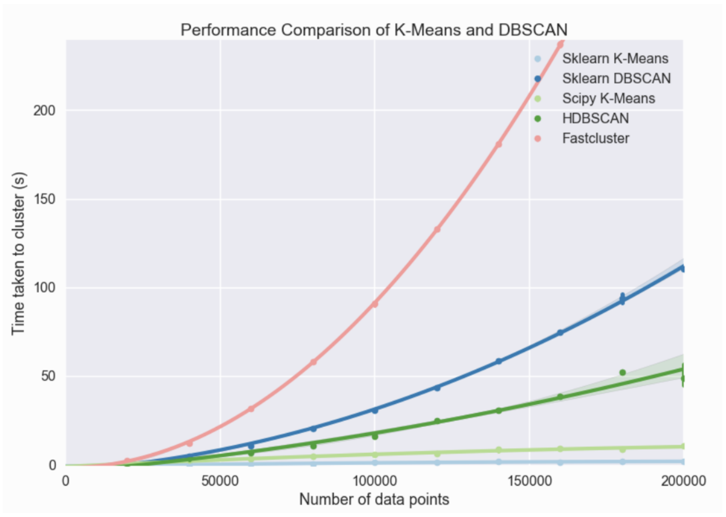

Figure 2.1: Figure of clustering benchmarks (taken from www.hdbscan.readthedocs.io/en/latest/performance_and_scalability.html#comparison-of-high-performance-implementations).

2.2 Indexing

We repeat the steps of extracting SIFT descriptors and project their descriptors using the codebook from the train set and count the occurrence of each prototype descriptor in every image.

2.3 Retrieving

The final part of CBIR program is to define similarity measures and based on these, find the closest train image (histogram) to a query image (histogram). Here we review the following measures: Euclidian distance, TF-IDF similarity, common-words and Bhattarya distance. These measures are extracted when comparing train images to themselves and while comparing a test image set to train images. The performance metrics of the retrieval system are defined in the introduction.

For the output we decided to write a pandas data frame in which we define the requested categories of top3 match, the exact match, and the mean reciprocal rank for each of or distance measures, Euclidean, Bhattarya, common-words, and TF-IDF. Also, we appended a note on the number of image features or descriptors for the train set.

Lastly, we call this whole class with our different test parameters: the number of categories ($i \in \{5, 20\}$) and different k-values ($k \in \{200, 500, 2000\}$) and test both for either the retrieval of the train images and test images. If the requested k exceeds the number of initial descriptors, k will just become this number. Interesting to note here is that the computation time not goes up for the number of categories, but also the number of k-values. That makes sense: the more categories we have for our descriptors, the more choices and therefore decisions must be taken.

2.3.1 Retrieving train

Below are the performance when the test set is the data, which CBIR has been trained on.

Distance	**% position in top-3**	**% position in top-1**	**Mean rank**
Euclidean	0.400000	0.580645	10.870130
Bhattaya	0.509677	0.122581	6.902597
Common words	0.541935	0.109677	6.214286
tf-idf	0.445161	0.251613	6.948052

Table 2.2: Accuracy for 154 train images, 155 test images from train set, 5 categories, kmeans with k=200, 503 features.

Distance	**% position in top-3**	**% position in top-1**	**Mean rank**
Euclidean	0.290323	0.470968	9.207792
Bhattaya	0.341935	0.109677	15.240260
Common words	0.348387	0.109677	17.058442
tf-idf	0.264516	0.245161	10.487013

Table 2.3: Accuracy for 154 train images, 155 test images from train set, 5 categories, kmeans with k=496, 496 features.

Distance	**% position in top-3**	**% position in top-1**	**Mean rank**
Euclidean	0.348387	0.458065	6.207792
Bhattaya	0.329032	0.154839	8.733766
Common words	0.303226	0.109677	14.597403
tf-idf	0.335484	0.354839	9.266234

Table 2.4: Accuracy for 154 train images, 155 test images from train set, 5 categories, kmeans with k=1772, 1772 features.

Distance	**% position in top-3**	**% position in top-1**	**Mean rank**
Euclidean	0.080797	0.463799	54.076433
Bhattaya	0.077650	0.026233	84.592357
Common words	0.056663	0.017838	75.299363
tf-idf	0.095488	0.046170	69.093418

Table 2.5: Accuracy for 942 train images, 953 test images from train set, 20 categories, kmeans with k=200, 645 features.

Distance	**% position in top-3**	**% position in top-1**	**Mean rank**
Euclidean	0.075551	0.467996	43.011677
Bhattaya	0.078699	0.039874	80.401274
Common words	0.076600	0.017838	146.719745
tf-idf	0.104932	0.039874	79.515924

Table 2.6: Accuracy for 942 train images, 953 test images from train set, 20 categories, kmeans with k=500, 601 features

Distance	**% position in top-3**	**% position in top-1**	**Mean rank**
Euclidean	0.44	0.41	30.22
Bhattaya	0.08	0.02	77.69
Common words	0.07	0.02	158.56
tf-idf	0.06	0.17	64.23

Table 2.7: Accuracy for 942 train images, 953 test images from train set, 20 categories, kmeans with k=730, 730 features

2.3.2 Retrieving test images

Below are the performance when the test set is the data, which CBIR has not been trained on.

Distance	**% position in top-3**	**% position in top-1**	**Mean rank**
Euclidean	0.31	0.54	7.54
Bhattaya	0.41	0.29	9.97
Common words	0.48	0.11	11.20
tf-idf	0.42	0.37	10.56

Table 2.8: Accuracy for 154 train images, 155 test images from test set, 5 category, kmeans with k=200, 1332 features.

Distance	**% position in top-3**	**% position in top-1**	**Mean rank**
Euclidean	0.37	0.45	10.28
Bhattaya	0.42	0.11	11.09
Common words	0.39	0.11	12.60
tf-idf	0.58	0.32	9.92

Table 2.9: Accuracy for 154 train images, 155 test images from test set, 5 category, kmeans with k=500, 1772 features.

Distance	**% position in top-3**	**% position in top-1**	**Mean rank**
Euclidean	0.40	0.52	7.37
Bhattaya	0.42	0.17	12.37
Common words	0.41	0.11	11.56
tf-idf	0.55	0.22	7.03

Table 2.10: Accuracy for 154 train images, 155 test images from test set, 5 category, kmeans with k=440, 440 features.

Next we will discuss these results.

2.3.3 Discussion of measures and results

First, we have a couple of important comments about our results. For all measures of the Euclidean top3 vs. top1 values, we can see that the top1 has always a higher value than the top3, which is,

Distance	**% position in top-3**	**% position in top-1**	**Mean rank**
Euclidean	0.07	0.45	37.53
Bhattaya	0.10	0.03	86.10
Common words	0.08	0.02	61.59
tf-idf	0.08	0.03	71

Table 2.11: Accuracy for 942 train images, 953 test images from test set, 20 categories, kmeans with k=200, 503 features.

Distance	**% position in top-3**	**% position in top-1**	**Mean rank**
Euclidean	0.06	0.47	39.35
Bhattaya	0.09	0.02	70.99
Common words	0.07	0.02	98.20
tf-idf	0.11	0.07	58.29

Table 2.12: Accuracy for 942 train images, 953 test images from test set, 20 categories, kmeans with k=264, 264 features

Distance	**% position in top-3**	**% position in top-1**	**Mean rank**
Euclidean	0.45	0.48	25.64
Bhattaya	0.08	0.03	74.11
Common words	0.07	0.018	118.76
tf-idf	0.08	0.19	61.63

Table 2.13: Accuracy for 942 train images, 953 test images from test set, 20 categories, kmeans with k=404, 404 features

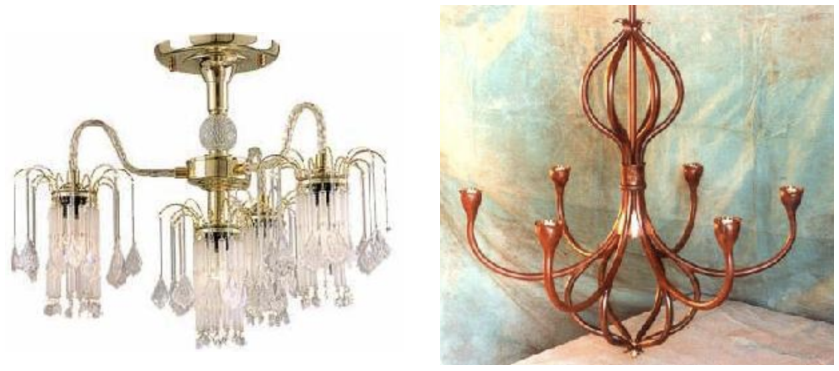

Figure 2.2: Left: Query image. Right: Correctly retrieved category

by definition, not correct. Unfortunately, correcting this error at the late stage of our experiments would have exceeded our resources, but for future investigating we would have to look into this. We suspect, that because the Euclidean distance is measured differently, that is as smaller the value the better the prediction (in contrast to every other measure, where the value corresponds to a positive number), also our results reflect this negative confusion. Secondly, the k in the kmeans clustering do not reach 2000 because the SIFT detector automatically chose the set of descriptors - these are in some cases below 2000, and therefore they cannot be clustered into because k cannot be larger than the number of descriptors. Thirdly, the results for the train images are suspiciously low compared and very similar to the test set (Example in fig 2.3). We had expected a very high score for train images because the histograms that we compare are identical to those used to extract descriptors and prototype; we thought there would always be a distance of 0 / similarity of 1 between two histograms in the sample, i.e., a perfect match. However, this is not the case. We will discuss these results despite that they do not make completely sense.

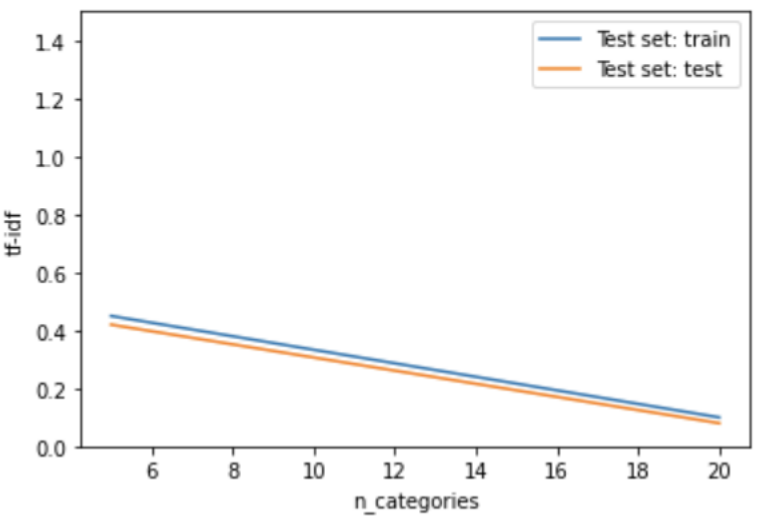

Figure 2.3: In and out of sample performance with k = 200, using tf-idf.

The baseline top-1 percentage for 5 categories based on random guesses would be 0.2. For 20 categories this would be 0.05.
For the 5 categories and k = 200 TF-IDF seems to be the best similarity measure when looking at top-1 percentage (ignoring the Euclidean distance, where there seems to be something incorrect) as can be seen in Tables 2.2 and 2.8. This is also the only distance measure that performs above chance level in this case. The top-1 percentage based on TF-IDF is surprisingly higher for the test data than it is for the train data. Based on that you would say that there is no overfitting on the traindata.The reason that this is higher could be due to the fact that the train and the test set contain different images, which lead to different and a different amount of features. As the optimal K depends on the amount of features it is hard to draw a hard conclusion. However, when looking at the top-3 percentage then the performance is better when looking at the train set compared to the test set. Based on that, you would say that that there still is overfitting on the train data. Besides that, when looking at the top-3 percentage one would say that common-words is the best distance measure.

When looking at 5 categories, it seems that k = 200 is better than k = 500, when looking at the test set as can be seen in tables 2.8 and 2.9.
When looking at 20 categories it seems that k = 500 is better than k = 200, when looking at the train set, as can be seen in tables 2.7 and 2.5. Important to note is that the performance is really bad and even slightly under random chance level. For more categories, a higher k is needed, since there is a higher number of features, which leads to a higher number of categories.

In figure Figure 2.4, we can see an example of a wrongly retrieved image compared to Figure 2.2. This example can reflect some of the limitations of the procedures in CBIR. First of all, there is some information lost when we convert the images to grayscale before the SIFT detector is applied. There is a larger diversity of colors in the dragonfly which do not match the colors in the headphone image. Secondly, the mismatch illustrate the limitations of the bag-of-visual-words approach; with

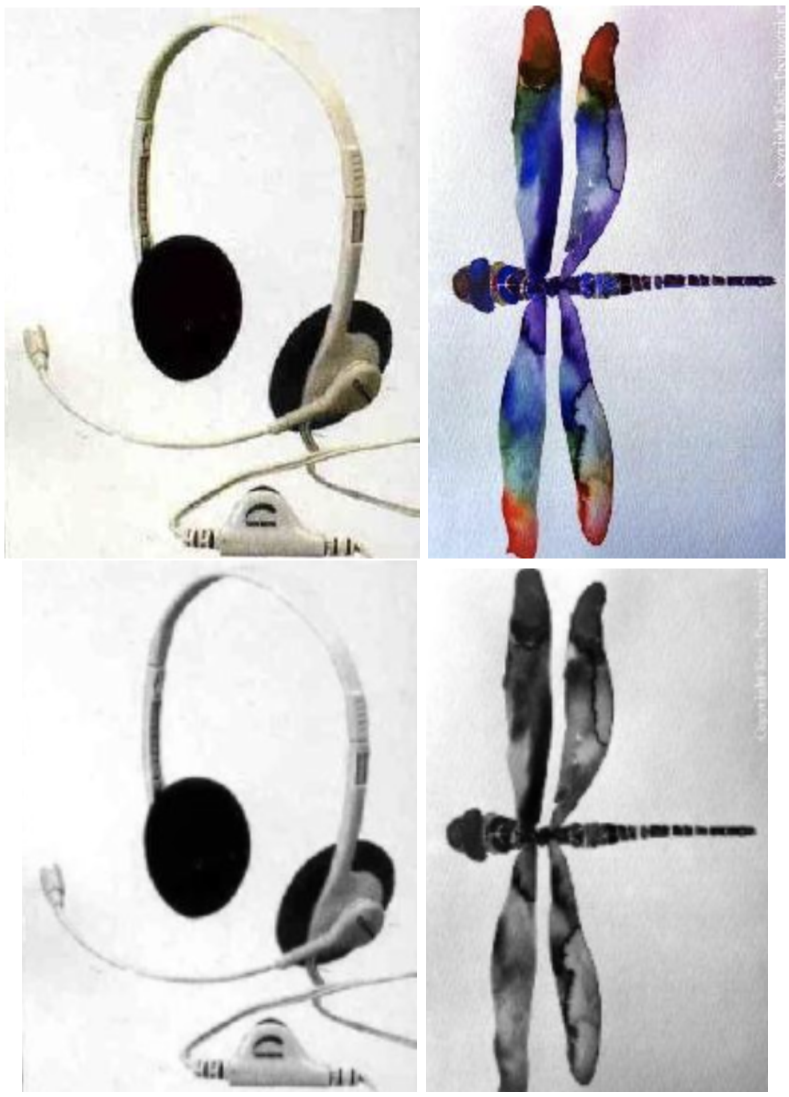

Figure 2.4: Top left and bottom: Query image in RGB and greyscale. Top right: Wrongly retrieved category in RGB and greyscale

this approach, the detected key-points and descriptors loose their spatial arrangement. When these two limitations are considered, we might see how the two images end up being matched, because they are both thin objects on a white background with some quite dark patches and some slightly brighter patches.