KTH - DD2476 Final Project
Content-Based Image Retrieval System using Deep Convolutional Networks

Antoine Broyelle broyelle@kth.se
Andreas Drangel anddra@kth.se
Wojciech Kryściński wkr@kth.se
Safir Najafi safirn@kth.se

May 19, 2017

Abstract

With the rapid growth of large collections of digital images the problem of efficient organization and retrieval of digital images has become a very popular research topic in recent years. In this project we designed, implemented and tested a Content-Based Image Retrieval (CBIR) system. One of the main parts of the project was to explore the different techniques of extracting meaningful image descriptors, with strong emphasis on the use of deep Convolutional Neural Networks (CNN) as they have recently made major breakthroughs in several areas. Another part has been to design a software architecture and to put components together.

1 Introduction

Image retrieval (IR) systems try to solve the problem of organizing and efficiently serving digital images in large collections. With the recent explosion of digital data, especially images, and the ever growing need for quick and accurate retrieval this field is developing very quickly. There is as variety of areas that benefit from CBIR systems, from the most obvious such as popular web search engines like Google Images[1] to more complex and specialized, such as medical imaging or public security [SSRKMK17].

Because of the broad scope of use, it is easy to see why there are a lot of ongoing projects in the area, both in research and industry[Wik]. Advances in the particular technology could improve the quality of people's lives, in both large and small.

2 Related Work

To begin with, a paper experimenting with color histogram algorithms that was proposed back in 1991. The paper is ground-breaking in the fact that it is the first ever to propose using colors instead of shapes to identify objects in images. The authors claim the color histogram algorithm is superior to the shape algorithms in many aspects, due to the advantage in speed, making it applicable to time critical tasks. Today histograms are a common way to evaluate images, but has flaws when it is the only used feature extractor, because it may generate false positives. [SB91]

Yann LeCun was the first one to successfuly train a large neural networks under supervised learning using back-propagation (gradient based technique)[YL]. More recently, CNNs has shown great results on computer vision related problems such as classification task [KSH12]. There is also a paper investigating how deep learning with Convolutional Neural Networks (CNN) can be used to classify images in content-based image retrieval.

[1] https://images.google.com/

The paper assesses several different methods of deep learning using the CNN and concludes that although not definitive, there are strong indicators that such deep learning techniques will be able to effectively outperform the, at the time, current evaluation methods. [WWH+14]

Lastly, an interesting library that was recently published by Facebook Research is Faiss (Facebook AI Similarity Search). Faiss was created with Facebook-scale in mind, it is optimized to handle billion-document collections while operating directly from RAM. The library was released together with a research paper [JDJ17] that analyzes the performance of a system built using Faiss using the Deep1B dataset that holds 1 billion images. Authors also describe how GPUs can be utilized to enhance the performance of CBIR systems and claim 8.5x speed boost compared to prior state-of-the-art.

3 Method

The architecture of CBIR systems is relatively simple and closely resembles that of text-document search engines. A simplified architecture is presented in figure 1. It consists of feature extractors that transform the images into a more compact and expressive representation, an index that stores feature vectors of all known images, a module that computes the similarity between documents and finally a module that ranks retrieved documents.

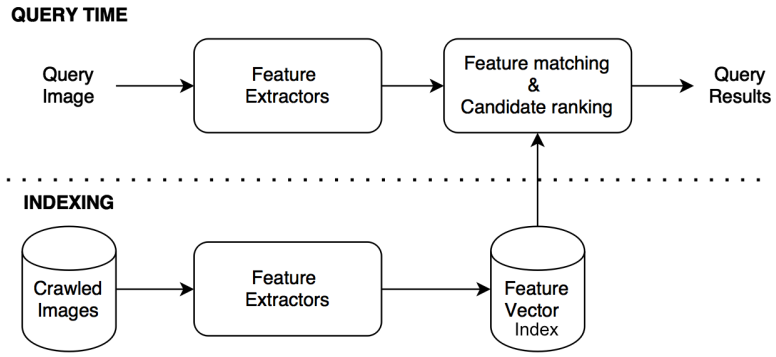

Figure 1: Simplified architecture of a CBIR system

3.1 Image Feature Extractors

One of the central points of any CBIR system is the module that extracts visual descriptors from images (also called image descriptors or image features). Visual descriptors are features of the content of images. They describe the colors, shapes and textures of objects present in a given image. After being extracted from an image, these feature vectors are indexed and used to represent the image in a similar fashion to term-frequency vectors in text-based search engines. Throughout the years there were many different trends in feature extraction. Some recent based on very advanced algorithms from the field of Computer Vision. [Wik04c]

In this project we decided to explore three feature extraction techniques. The most basic one, serving as the baseline for experiments, is a 3D Color histogram. The second one, that has recently gained much popularity in the field of CBIR, is a pre-trained Convolutional Neural Network. We also propose a new technique, inspired by the research frontiers, that combines both previously mentioned extractors. These techniques are described in more detail in the following subsections.

3.1.1 Color Histogram

A color histogram represents the distribution of colors in an image. In digital images it indicates the number of pixels in an image that have a specific color. Color histograms are a general technique and can be built in different color spaces such as RGB, HSV or LAB. Usually, they are computed separately for each channel in the image, for the entire intensity range, resulting in 256 (0-255) bins per channel. If a more compact representation is needed it is possible to quantize the image colors into less bins on a per-channel basis. It is also possible to obtain a more detailed view of the correlations between colors by building a multidimensional histogram that counts the occurrence of combinations of intensities. This type of histogram quickly explodes in size, $(num\_bins)^{(num\_channels)}$, so quantization must be used. [Wik04a]

In CBIR systems color based image descriptors are a very popular choice because of their simplicity and stability. This type of feature extractors is robust to various image transformations, such as scaling, rotation, or slight changes of view angle. Such robustness is a very desirable characteristic in feature extractors. Color histograms are also relatively easy and fast to compare using popular distance metrics such as the Euclidean distance, histogram intersection or χ^2 (chi-squared) test. Unfortunately the simplicity of the technique also limits its ability to take advantage of more complex information present in the image, such as texture or spatial distribution of content. This type of information proves to be crucial in more complex CBIR use-cases. [MGM09, SSRKMK17].

A simple yet very interesting extension of color histograms is *Local Color Histograms*. The idea behind it is that instead of computing one color histogram per entire image (making it a global descriptor) it is better to divide the image into regions and compute a color histogram on a per-region basis. This extension allows to incorporate some spatial information into the extracted features. As a side note it is worth mentioning that this technique is related to how human visual perception works.[SSRKMK17].

3.1.2 Convolutional Neural Networks

Convolutional Neural Networks (CNNs) are a type of artificial neural network that are loosely inspired by the organization of the visual cortex. These type of networks are primarily used for tasks related to computer vision, such as object classification, object detection or semantic segmentation. In comparison to regular neural networks, they have specialized convolutional and pooling layers that are specifically designed to work with multi-dimensional data organized in volumes that has intricate spatial dependencies. The main idea driving these models is to build a hierarchical representation of the input data that grows in complexity (in a positive way) with the depth of the network. When many of these layers are stacked together each layer contributes one representation level that builds upon those coming from previous layers. The network gains the ability to locate very complex and high level features in images such as faces or particular objects in a way that is invariant to scaling, rotation or illumination. Trained convolutional networks can be very easily transformed into feature extractors, simply by using one of the intermediate representations of the input data that the networks build internally. Unfortunately, as in the case of histograms the extremely appealing benefits of the network also have some drawbacks. Convolutional Neural Networks are very complex architectures that require huge amounts of data, computational power and time to train. Only a few institutions with appropriate infrastructure are able to successfully train this type of network from scratch [ZF13, RGC15, Wik04b].

The possible answer to this issue is a technique called "transfer learning". It is a very

active topic and ongoing research in exploring the ability of transferring knowledge of trained models to new domains and similar tasks. It has been shown that Deep Learning models generalize very well, which means that if they are trained to do a certain task they can easily be applied to a very similar task with little or no retraining [PY10]. There are several popular pre-trained models shared for public use by top research groups. One of them is the VGG network created by the Visual Geometry Group from the University of Oxford in 2014 [SZ14]. The architecture of this network is shown in figure 2. It consists of a very deep (as of that time) network with multiple convolutional layers grouped into blocks (indicated by different colors) ending with 3 fully-connected layers. The VGG16 network was trained specifically for the ImageNet contest. It has been used in countless projects as the feature extractor of choice, eg. [LCH+16, SLD16].

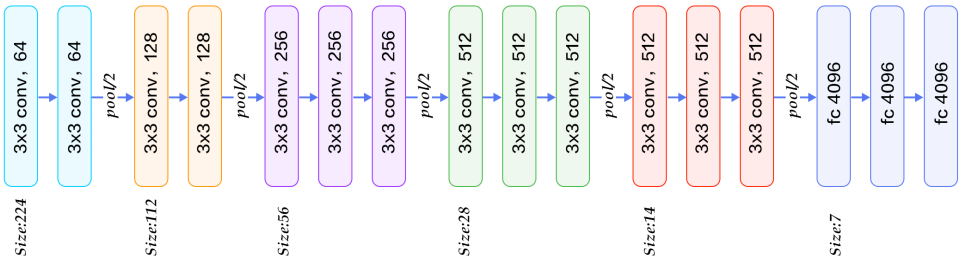

Figure 2: The architecture of the VGG-16 network used in this project as a feature extractor. Image taken from [Bai].

3.1.3 Multi-level feature fusion

Considering the incremental way how CNNs build the hierarchical representation of input data, a reasonable idea would be to not only rely on the late-stage features, that represent very high-level concepts, but also take into consideration those available in earlier stages of the network, which described textures or shapes. This in theory should make the feature extractor more robust and generalize to more types of images [ZF13].

The intermediate representations created as the data flows forward through the network are called *feature maps*. These feature maps can be thought of as two dimensional greyscale images that usually have a large number of layers (64, 256, 512 and more, compared to only 3 for RGB images). Each channel of the feature map can be interpreted as a intensity-map that shows whether (and where) a certain pattern was present in the input coming into the given layer. Even though the spatial size of the feature map decreases in later stages of the network, a naive transformation of simply unrolling these feature maps into vectors would yield an extremely high dimensional feature vector. This would not be very efficient to store or manipulate during retrieval. As part of this project we propose a method that make use of color histograms as a way of summarizing the contents of each channel of feature maps using a small number of bins (preferably 3). Using this technique will make the generated feature vector independent of the spatial size of feature maps and linear in the number of channels ($num\_bins * num\_channels$), which is easier to manage.

The method described in this section was loosely inspired by the following papers [ZF13, SLD16, VS16]. With the given timespan for the project it was impossible to do a thorough review of the entire literature related to this field, but to the best of our knowledge, the method of extracting and combining features described in this section has not been studied or published.

3.2 Indexer and Search Engine

Elasticsearch 5.4 is used to index the extracted contents and to query images against the index. Elasticsearch is an open-source search engine, initially developed for querying data with respect to similarity features. The software has a large user-base and an extensive documentation and is also designed to be versatile. Elasticsearch is by design a service aiming to be flexible to users. This enables users to modify settings, structures and the implementation of modules or subparts of Elasticsearch to function the way they like. For example, scripts may be added to change how elements are scored (ranked retrieval) or configuring how data is managed during indexing. Furthermore, advanced configuration is available for details of the index, for example by configuring the settings of shards, replicas, or clusters. Indexing with Elasticsearch stores data in permanent storage, making it easy and flexible to use Elasticsearch even when the service is somehow interrupted. Naturally, this results in longer indexing times, but the need for re-indexing is only ever needed when the data in the index is insufficient or outdated.

Querying and indexing items in Elasticsearch is done by sending HTTP-requests containing properties in JSON-format. Scoring when querying is performed by using scripts. Several scripting languages may be used to score a query, but we have made use of groovy since it uses very general syntax and is easy to learn for our simple scoring purposes. When a custom script is to be used for scoring, Elasticsearch must be informed about which script to run during query time through the HTTP-request. Also, the script must be present in the Elasticsearch-directory.

3.3 User-facing system

A React-based[2] user-facing front-end system was developed in order to perform image queries. The system communicates through HTTP-based API endpoints with a back-end system which in turn connects to the feature extractor and the Elasticsearch search index. An image is fed as input to the front-end system, together with selections of which feature extractor to use (histograms, CNNs or a combination of both) and which similarity measure to use (euclidean distance, cosine similarity, histogram intersection, etc). These selections are then communicated to the back-end which extracts the selected features from the image according to the selected feature extractor and uses the selected similarity measure for querying the search index. Relevant images are then returned from the index and communicated back to the front-end system where they are displayed to the user.

4 Experiments

4.1 Dataset

All the experiments were conducted using the MIRFLICKR-25k Retrieval Evaluation dataset [HL08]. This collection consists of 25,000 color images downloaded from Flickr. The content of images spans across different classes, which makes it a great collection for evaluating general CBIR systems. Images come with additional metadata, such as camera settings, hashtags and location, but it was not used during this project. Additionally some images in the dataset are annotated with one or several labels that describe the visible content.

[2] https://facebook.github.io/react/

Two levels of annotations are provided. According to the documentation, the first level consists of relevant labels while the second one regroups potential labels and related topics. Only the relevant annotations were used for evaluating the built system.

For testing and evaluation purposes, 100 images were extracted from the original dataset. The final index contains exactly 24.900 images. The test images were selected pseudo-randomly so that the label distribution over the test set is roughly the same as in the original data set (see Figure 1). The consequence is that there is at least one label per image in the test set.

number of images	label
7849	people
3982	female
3829	portrait
3647	male
1350	clouds
1077	flower
669	night
668	tree
590	dog
484	bird
380	car
214	sea
149	river
116	baby

Table 1: Label distribution in FLICKR-25k

4.2 Feature extractors

4.2.1 Histogram

The baseline extractor was a 3D Color Histogram in the HSV color space with (9, 3, 3) bins per H-S-V channel and 5 regions per image. This extractor encoded every indexed image into a vector of length 405. This feature extractor will be referred to as ***hist-basic***. Example features obtained from the convolutional layers are shown in Figure 3.

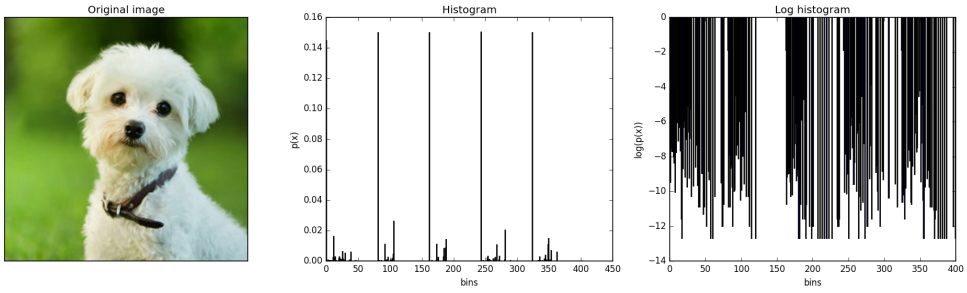

Figure 3: Features extracted with *hist-basic*. The histogram is also shown in log-space for better visualization.

4.2.2 CNN

The second extractor was the VGG-16 Convolutional Neural Network pre-trained for the ImageNet contest. The outputs of the last fully connected layer `FC3` were treated as image descriptors. This extractor encoded every indexed image into a vector of length 4096. Due to insufficient computational power, the network was not fine-tuned for the dataset used in these experiments. This feature extractor will be referred to as ***cnn-basic***. Example features obtained with this extractor are shown in Figure 4

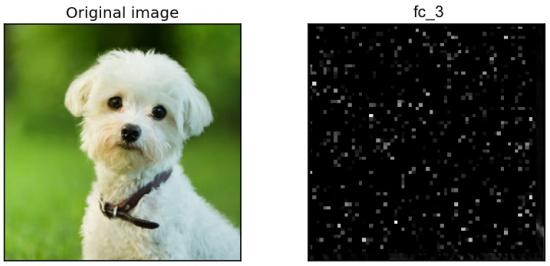

Figure 4: Features extracted with *cnn-basic*. The 4096 element vector (right) was reshaped to a 64x64 grid for better visualization.

4.2.3 Hybrid : Histogram-CNN

The third extractor was the multi-level feature fusion using the same network as for *cnn-basic*. The extractor used feature maps coming from the second convolutional layers in all 5 convolutional blocks, `[BLOCK1_CONV2, BLOCK2_CONV2, BLOCK3_CONV2, BLOCK4_CONV2, BLOCK5_CONV2]`. These feature maps are "summarized" using color histograms with 3 bins. This method yields a feature vector of length 4416. This feature extractor will be referred to as ***cnn-hist***. Example features obtained from the convolutional layers are shown in Figure 5. The image shows features before the histograms were computed.

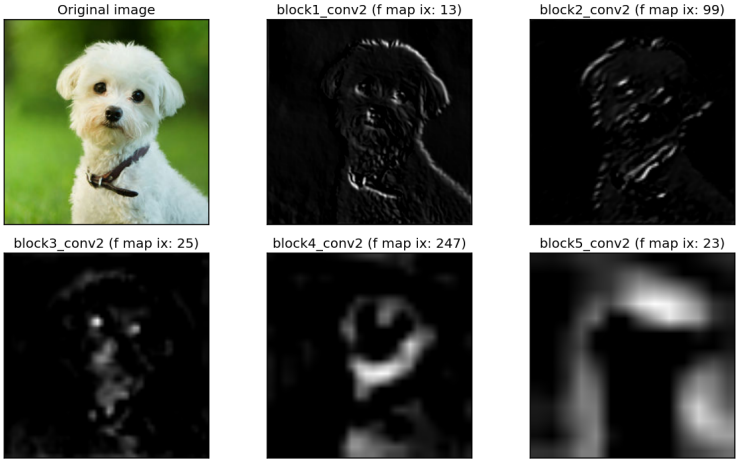

Figure 5: Features extracted from the convolutional layers of the VGG network *cnn-basic*. Such feature maps are then "summarized" by computing the 3-bin histogram.

4.3 Similarity measures

A collection of similarity measures were used for the querying and calculating of the metrics. The similarity measures used in the project are displayed together with their respective mathematical model below. $\boldsymbol{A}$ and $\boldsymbol{B}$ are representations of the vectors used for the similarity metric.

Cosine similarity	$\cos(A,B) = \frac{\sum_{i=1}^{n} A_i B_i}{\sqrt{\sum_{i=1}^{n} A_i^2}\sqrt{\sum_{i=1}^{n} B_i^2}}$ (higher values better)
Euclidean distance	$d(A,B) = \sqrt{\sum_{i=1}^{n}(A_i - B_i)^2}$ (lower values better)
Chi^2	$\chi^2(A,B) = \frac{1}{2}\sum_{i=1}^{n} \frac{(A_i - B_i)^2}{(A_i + B_i)}$ (lower values better)
K-L-divergence	$KL(\boldsymbol{A}\|\boldsymbol{B}) = \sum_{i=1}^{n} A_i \ln(\frac{A_i}{B_i})$ (lower values better)
Bhattacharyya	$D_B(A,B) = -\ln \sum_{i=1}^{n} \sqrt{A_i B_i}$ (lower values better)
Intersection	$S(A,B) = \frac{\sum_{i=1}^{n} \min(A_i, B_i)}{\sum_{i=1}^{n} A_i}$ (higher values better)

4.4 Evaluation

The evaluation of a search engine is a critical point in its development process, as it allows to rate the performance of the system in a quantitative manner. Scores obtained from the evaluation process form the basis for comparing different retrieval systems and deciding whether further improvements are needed.

In this work, we consider that a returned image is relevant if it shares at least one label with the query image. Also, for a given query, we define the number of relevant documents

as the number of images that have at least one label shared with the test image. As the number of relevant document could be quite large, the recall tends to be really small and becomes more complicated to interpret.

To analyze the performance of our system on a test query, we used the evolution of precision over the number of returned documents for the first 100 documents. The interval is quite large for text-document retrieval, but images are marginally faster to process and evaluate for human beings. Thus it is more likely that a person searching for an image will browse 100 images than that a person searching for a text-document will read 100 documents. Figure 6 illustrates an example query and related metrics.

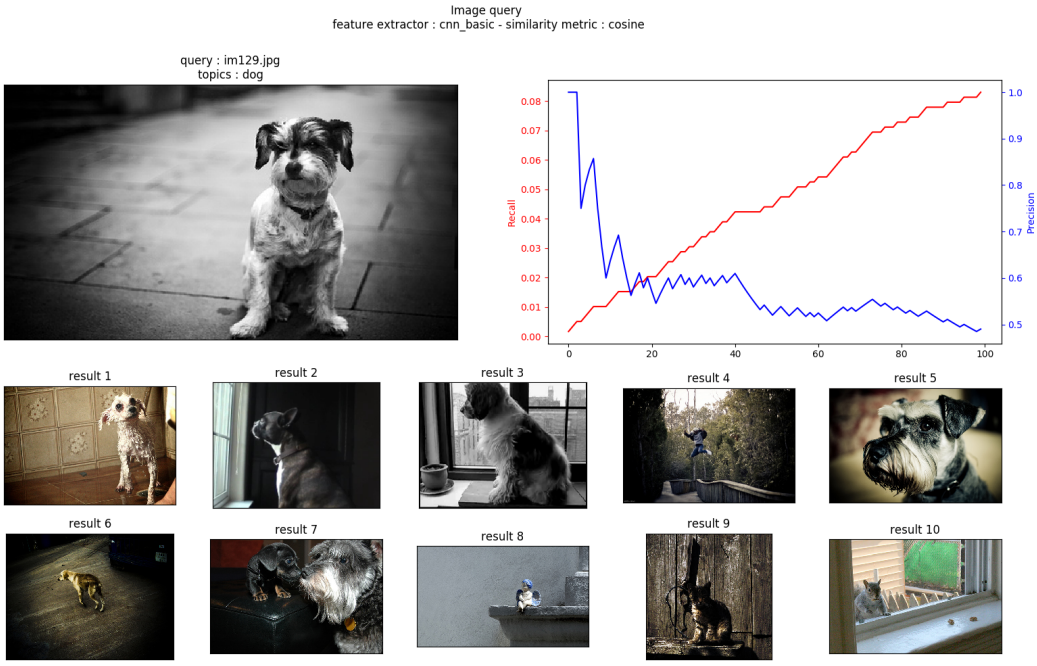

Figure 6: Example query and related metrics

However, the evolution of a curve could be quite hard to follow so we use Average Precision at 100 (AP@100). AP is useful for our purposes because it puts emphasis on the order in which the returned documents are presented. For example, relevant items ranked at the bottom half of the retrieved items and non-relevant items ranked highly will result in a low AP, while the reversed situation would result in higher AP. To evaluate the full system, we made use of the metric Mean Average Precision at 100 (MAP@100). MAP is just an averaging of AP over the whole test set. Figure 7 summarizes the performance for all combination of feature extractors and similarity metrics.

	euclidean	intersection	cosine	chi2	kl	bhattacharyya
cnn-basic	0.5215	0.5098	0.5215	0.5288	0.0108	0.4757
hist-basic	0.0888	0.1000	0.1145	0.1090	0.0271	0.1063
cnn-hist	0.3518	0.3706	0.0440	0.3920	0.3397	0.41245

Figure 7: Mean Average Precision at 100 images

Results of experiments show that the best combinations of feature extractors with similarity metrics are: ***cnn-basic*** with euclidean or cosine distance, ***hist-basic*** with cosine and ***cnn-hist*** with bhattacharyya.

As expected, features extracted using ***cnn-basic*** outperform those extracted with ***hist-basic*** independent of the similarity metric. However, in our experiments ***cnn-hist*** came

close to ***cnn-basic*** in terms of performance. In order to get a better understanding of the result, we ran manual tests and found a tricky one. In this scenario, the query image is a smartphone with an ocean as a background picture. As we can see in Figure 8, the ***cnn-basic*** interprets the image as a screen. The ***hist-basic*** see a blue image (see Figure 9) and the ***cnn-hist*** learns the spacial distribution of colors (see Figure 10).

Figure 8: Phone with ocean background as a query image. ***cnn-basic*** extractor and euclidean distance

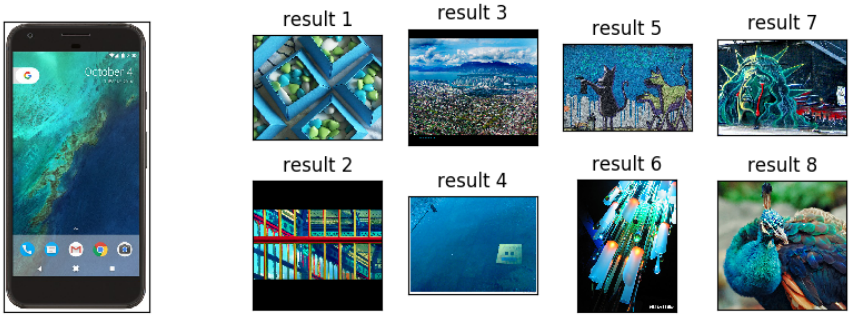

Figure 9: Phone with ocean background as a query image. ***hist-basic*** extractor and bhattacharyya distance

Figure 10: Phone with ocean background as a query image. ***cnn-hist*** extractor and bhattacharyya distance

With any other query, we experienced a better empirical results with images retrieved by the ***cnn-basic*** than the ***cnn-hist***. Here are a few comments and possible improvement on the automatic evaluation system. By using only the relevant annotations, we only use 14 classes of labels and some of them cover more than 1/5 of the dataset. Figure 11 illustrates this problem. The query image is annotated with the label `male` because there is a person

in the bottom left corner. The returned images seems quite interesting but the precision stay around 0 as they all represent buildings but there is no label related to this topic.

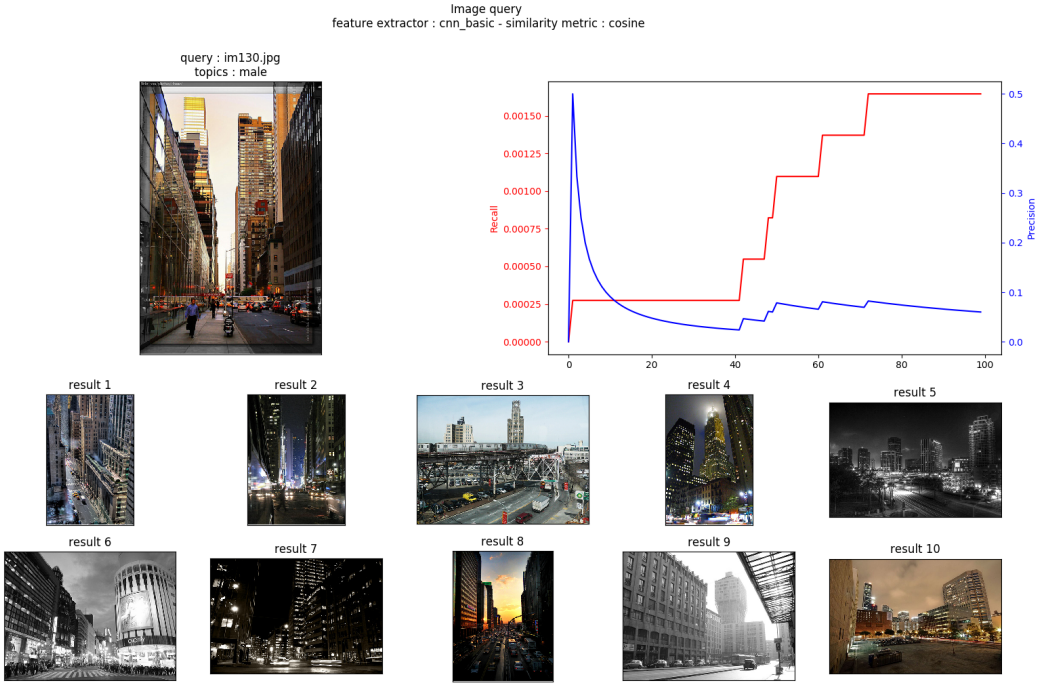

Figure 11: Example of a bad evaluation due to the limited set of labels

5 Conclusions

Our experiments showed that the choice of the feature extractor and the metric used to compare features have a great influence on the overall performance of the CBIR system. Choosing the best combination of the mentioned components proved to be non-trivial.

According to Figure 7, using ***cnn-basic*** outperforms ***cnn-hist*** which in turn outperforms ***hist-basic*** across almost all similarity measures. There are two inconsistent MAP values in the figure which does not follow this pattern, the *cosine similarity* for ***cnn-hist*** and the *Kullback-Leibler distance* for ***cnn-basic***. Due to time constraints on this project we did not investigate these inconsistencies any further. We assume that the real values of these inconsistencies should follow the pattern of the majority of the similarity measures. A change of similarity metric should not influence the MAP by an order of magnitude.

As is apparent from the experiment, using hybrid CNN-histogram features works less well than CNN features alone. We are not convinced that the combination of CNN- and histogram-based features can't work well together, but maybe another method of summarizing the CNN feature maps is needed. Further research could aim to explain this matter by investigating the possibilities of using different methods of summarizing feature maps or extracting feature maps from different layers.

In terms of performance, we were able to implement our solution and make it run on average laptops. The query takes around one second.

References

[Bai] Baidu. Paddle paddle. [Online; accessed 3-May-2017].

[HL08] Mark J. Huiskes and Michael S. Lew. The mir flickr retrieval evaluation. In *MIR '08: Proceedings of the 2008 ACM International Conference on Multimedia Information Retrieval*, New York, NY, USA, 2008. ACM.

[JDJ17] Jeff Johnson, Matthijs Douze, and Hervé Jégou. Billion-scale similarity search with gpus. *CoRR*, abs/1702.08734, 2017.

[KSH12] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 25*, pages 1097–1105. Curran Associates, Inc., 2012.

[LCH+16] Yun Liu, Ming-Ming Cheng, Xiaowei Hu, Kai Wang, and Xiang Bai. Richer convolutional features for edge detection. *CoRR*, 2016.

[MGM09] S. Murala, A. B. Gonde, and R. P. Maheshwari. Color and texture features for image indexing and retrieval. In *2009 IEEE International Advance Computing Conference*, pages 1411–1416, 2009.

[PY10] Sinno Jialin Pan and Qiang Yang. A survey on transfer learning. *IEEE Trans. on Knowl. and Data Eng.*, 22(10):1345–1359, 2010.

[RGC15] Adriana Romero, Carlo Gatta, and Gustau Camps-Valls. Unsupervised deep feature extraction for remote sensing image classification. *CoRR*, abs/1511.08131, 2015.

[SB91] Michael J. Swain and Dana H. Ballard. Color indexing. *International Journal of Computer Vision*, 7(1):11–32, 1991.

[SLD16] Evan Shelhamer, Jonathan Long, and Trevor Darrell. Fully convolutional networks for semantic segmentation. *CoRR*, abs/1605.06211, 2016.

[SSRKMK17] Pallikonda Sarah Suhasini, K. Sri Rama Krishna, and I. V. Murali Krishna. Content based image retrieval based on different global and local color histogram methods: A survey. *Journal of The Institution of Engineers (India): Series B*, 98(1):129–135, 2017.

[SZ14] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. *CoRR*, abs/1409.1556, 2014.

[VS16] Domonkos Varga and Tamás Szirányi. Fast content-based image retrieval using convolutional neural network and hash function. pages 2636–2640, 2016.

[Wik] Wikipedia. List of cbir engines. [Online; accessed 10-May-2017].

[Wik04a] Wikipedia. Color histogram — Wikipedia, the free encyclopedia, 2004. [Online; accessed 3-May-2017].

[Wik04b] Wikipedia. Convolutional neural network — Wikipedia, the free encyclopedia, 2004. [Online; accessed 3-May-2017].

[Wik04c] Wikipedia. Visual descriptor — Wikipedia, the free encyclopedia, 2004. [Online; accessed 3-May-2017].

[WWH+14] Ji Wan, Dayong Wang, Steven Chu Hong Hoi, Pengcheng Wu, Jianke Zhu, Yongdong Zhang, and Jintao Li. Deep learning for content-based image retrieval: A comprehensive study. In *Proceedings of the 22Nd ACM International Conference on Multimedia*, MM '14, pages 157–166, New York, NY, USA, 2014. ACM.

[YL] Y. Bengio P. Haffner Y. Lecun, L. Bottou. Gradient-based learning applied to document recognition.

[ZF13] Matthew D. Zeiler and Rob Fergus. Visualizing and understanding convolutional networks. *CoRR*, abs/1311.2901, 2013.