YFCC100M-HNfc6: a Large-Scale Deep Features Benchmark for Similarity Search

Giuseppe Amato, Fabrizio Falchi, Claudio Gennaro, and Fausto Rabitti

ISTI-CNR, via G. Moruzzi 1, 56124 Pisa, Italy
<firstname>.<lastname>@isti.cnr.it

Abstract. In this paper, we present YFCC100M-HNfc6, a benchmark consisting of 97M deep features extracted from the Yahoo Creative Commons 100M (YFCC100M) dataset. Three type of features were extracted using a state-of-the-art Convolutional Neural Network trained on the ImageNet and Places datasets. Together with the features, we made publicly available a set of 1,000 queries and k-NN results obtained by sequential scan. We first report detailed statistical information on both the features and search results. Then, we show an example of performance evaluation, performed using this benchmark, on the MI-File approximate similarity access method.

Keywords: Similarity Search, Deep Features, Content-Based Image Retrieval, Convolutional Neural Networks, YFCC100M

1 Introduction

The ability to efficiently search for similarity in large databases of images is a critical aspect for a number of content-based retrieval applications like web search engines, e-commerce, museum collections, medical image processing, etc. To address this problem, several approaches based on index methods have been proposed in the literature, such as approximate access methods based permutation-based indices [22, 10, 4, 12, 16]. However, an important issue in comparing performance of different access methods is the availability of realistic benchmarks, of very large size.

In this paper, we present YFCC100M-HNfc6, a similarity-search benchmark consisting of three types of features extracted from 97M images, and pre-computed similarity search results for k-NN searches (k=10,001) on 1000 queries. To make scalability assessment of access methods, precomputed results for the 1000 queries were generated for increasing sizes of the dataset at intermediate steps of 1M. The features of the benchmark were extracted from the YFCC100M [21] dataset using state of the art Deep Convolutional Neural Networks.

Deep learning methods are "representation-learning methods with multiple levels of representation, obtained by composing simple but non-linear modules that each transform the representation at one level (starting with the raw input) into a representation at a higher, slightly more abstract level" [15]. Starting from 2012 [14], Deep Convolutional Neural Networks (DCCNs) have attracted

enormous interest within the Computer Vision community because of the state-of-the-art results achieved in image classification tasks. The relevance of the internal representation learned by the neural network during training have been proved by recent works that the activation produced by an image within the intermediate layers can be used as a high-level descriptor of the image visual content [19, 6, 8, 17].

The importance of having a very large dataset of publicly available features has been proven by the Content-based Photo Image Retrieval (CoPhIR) [7][1] we released on 2009, which has been used by many scientists working in the field of very large scale similarity search algorithms. The CoPhIR dataset consists of MPEG-7 features extracted from about 107M Flickr! images. Given the impressive improvement recently achieved in many Content-Based Image Retrieval applications by using Deep Features, we decided to create a new benchmark based on features obtained as activations of DCNNs. To accomplish to this task we had the opportunity to have access to an already publicly available image dataset, i.e., the YFCC100M [21], from which we extracted the deep features, and we collaborated with the team of the Multimedia Commons Initiative [2] that made our deep features also available through their website.

The availability of extracted features from large datasets contributes to the research in the field in three ways. First, it allows a fair comparison between similarity access methods. In fact, details in the feature extractions could result in slightly different features obtained by various research group making impossible comparing the performance measures obtained. Second, the extraction of some features, as the deep features, is computational demanding. Extracting features from a collection of 100M documents can take months on a standard PC or special hardware, as clusters of GPUs, are required to do it in days. Third, when the images of the dataset are public available online (as for CoPhIR and YFCC100M), having the features allows researchers to index without storing them locally but just pointing to them whenever results have to be shown.

The rest of the paper is organized as follows. In Section 2, we briefly describe related work. Detailed information about the dataset and the features extracted are given in Section 3. In Section 4, we give statistical information about both features and search results. Section 5 provides some general metrics for measuring the quality of approximate results and a case study of their application. Section 6 concludes the paper.

2 Related Work

There are other very large datasets that can be used as a benchmark for assessing the performance of similarity search access methods. Among the most relevant we mention the CoPhIR and the TEXMEX dataset.

The CoPhIR dataset [7] consists of 107 millions MPEG-7 features extracted from images. Not all the images in CoPhIR have a creative common license.

[1] `http://cophir.isti.cnr.it/]`

Moreover, Deep Features have been proved to outperform previous approach as MPEG-7 in many tasks [17].

The ANN_SIFT1B dataset from the TEXMEX corpus [2] consists of one billion SIFT local features of 128 dimensions extracted from about 1 million images. Any image has about 1,000 features and each of them would be a query for the system. Moreover, a ground truth with image as queries was note defined. It is worth to mention that, in terms of images, our proposed dataset is 2 order of magnitude bigger and the proposed deep features have larger dimensionality.

Deep Convolutional Neural Networks (DCNNs) have recently become state-of-the-art approach for many computer vision task such as image classification [14, 20],image retrieval [11, 6, 17, 14, 20]and object recognition [11]. The use of the activation of intermediate layers as a high-level descriptor of the image visual content has been also proved to be effective by many recent works [19, 6, 8, 17]. Rectified Linear Unit (ReLU) is part of almost all of the DCNN models and is typically applied also for extracting deep features from images [11, 8]. However, there are works in which the ReLU was omitted [19, 6, 17]. The L2 Normalization of the feature in order to and compare using the Euclidean distance is a standard de-facto for deep features [19, 8]. It is worth to mention, that the resulting ranking of similarity search is equivalent to the cosine similarity. Principal Component Analysis has been successfully used in [18, 6].

3 The YFCC100M-HNfc6 dataset

The Yahoo Flickr Creative Commons 100M (YFCC100M) [21] consists of approximately 99.2 million photos and 0.8 million videos, all uploaded to Flickr between 2004 and 2014 and published under a Creative Commons commercial or non commercial license. Metadata for the YFCC100M dataset are publicly available through Yahoo! Webscope[3]. YFCC100M images can be obtained directly from Flickr! using the information reported in the metadata or through the Multimedia Commons Initiative website[4] using the hash of the original image also reported in the metadata.

The YFCC100M-HNfc6 was obtained by extracting, from the YFCC100M images, the HNfc6 Deep Features, described in Section 3.1, and by pre-computing the k-NN results for 1,000 queries. We selected the first 1,000 images of our ordering (which is random) as similarity search queries. We performed k-NN search with $k = 10,001$ sequentially scanning the data. We report results at intermediate steps of 1 million objects in order to allow scalability measures of access methods. The query itself is between the results generated at any intermediate step. The results of the k-NN queries can be used as ground-truth results for ranges up to 0.629 for the Euclidean distance and up to 606 for the Hamming. We did not to consider images smaller than 4KB; therefore, we extracted features from 96,976,180 of the 99,206,564 images in YFCC100M [21]. YFCC100M-

[2] `http://corpus-texmex.irisa.fr/`

[3] `https://webscope.sandbox.yahoo.com/catalog.php?datatype=i&did=67`

[4] `https://multimediacommons.wordpress.com`

HNFc6 features are available in 97 zipped text files in which the objects have been randomly ordered. Thus, any subset of the full dataset contains random objects from the full dataset. The features, the k-NN results, demos of on-line systems indexing the features, and other information is available on the deep features website [1]. As a result of different processing of the neuron activation, we give three distinct features: *ReLU-L2Norm*, which is the reference feature for the dataset; *Binary*, which is intended for high efficiency and *Raw*, which allows other researchers to test other type of processing of the neurons activations. Information about the extracted features is given in following section.

3.1 The HNfc6 Deep Features

Deep features have been extracted with a trained model publicly available for the popular Caffe framework [13]. Many deep neural network models and in particular trained models are available for this framework at[5] . Among them, we chose the HybridNet for several reasons: first, its architecture is the same of the famous AlexNet [14]; second, the HybridNet has been trained not only on the ImageNet subset used for ILSVRC competitions (as many others), but also on the Places Database [23]; last, but not least, experiments conducted on various datasets demonstrate the good transferability of the learning [23, 8, 5]. We decided to use the activation of the first fully connected layer (i.e., fc6) given the results reported on [11, 6, 8]. It is worth to mention that the activation of the second fully connected layer (i.e., fc7) can be obtained from the fc6 activation with a simple matrix operation using the pre-trained weights [1].

As a result of different processing of the neuron activation, we give three distinct features described in the following.

ReLU-L2Norm The reference features of our YFCC100M-HNfc6 data are 4,096 dimensional L2 Normalized vectors corresponding to the activation of the neurons of the HybridNet *fc6* layer after the ReLU. This activation function, which is part of the HybridNet Convolutional Neural Network, simply sets to zero all the elements of the vectors that are negative. The distance to be used to compare is the Euclidean (aka L2 distance). The ranking of the results for a query using a combination of the L2 normalization and Euclidean distance is the same to the ones using the Cosine similarity of the original feature vectors.

Binary We provide a binary version of the deep features consisting of 4,096 bits (i.e., 512 bytes). We simply encoded the positive values of the activations as 1s, while zeros and negative values as 0s. We evaluated k-NN results also for these features using the Hamming distance.

Raw The activations of the fc6 neurons without the ReLU are given for anyone that would like to try approaches different from the previous ones and can be

[5] `https://github.com/BVLC/caffe/wiki/Model-Zoo`

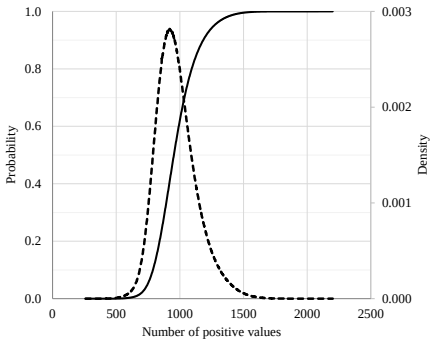


Fig. 1: Cumulative distribution function and probability density of n. positive elements per image.

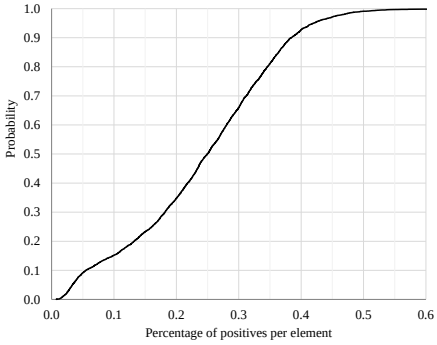


Fig. 2: Cumulative distribution function of percentage of positives per feature element

useful to researchers willing to test new processing of neuron activations and experiments. We also give a k-NN results for these features comparing them with the Euclidean distance after the L2 Normalization. Please note that the values given on the deep features website [1] are not L2 Normalized.

Please note that the fc7 layer activation can be obtained from the raw fc6 data quite easily and without the use of any external library. On the deep features website, we give all the necessary information and a code example for anyone interested.

4 Statistical Analysis

In this Section, we report detailed statical information for the proposed benchmark. As reported in the previous Section, YFCC100M-HNfc6 consists of three version of deep features, which are the results of different processing of the exact values of the neuron activation.

We first show some statistics about the sign of the activation of the neurons that constitute the deep features. Each neuron of a hidden layer, as the *fc6* we considered, basically reports whether a specific high level feature, learned during the training phase, is found in the input image. The sign of the activation reports if the specific feature was found or not, while the absolute value is a measure of the confidence on this information. These statistic also illustrate the sparsity of both the *ReLU-L2Norm* and *Binary* features. Please note that the sparsity of the features is relevant for any access methods that would consider compression or techniques that require sparse information as inverted files.

In Figure 1, we report the cumulative distribution function and probability density function for the number of positive values in features. This statistic is the very same for the *ReLU-L2Norm*, *Raw*, and *Binary*. In particular, the amount of positives has the following statistics: *min*=221, *mode*=919, *median*=972, *mean*=972 and *max*=2,201. The results show that on average, each

image has about 25% of positive elements. Thus, the *ReLU-L2Norm* vectors are quite sparse.

We now consider each vector component (i.e., each neuron in the *fc6* layer) individually. The goal of this analysis is evaluating the sparsity and estimating the usefulness of each particular element of the feature vector. As an example, an element whose values is always zero would be useless. Moreover, the overall sparsity of the data (shown in Figure 1) could be the results of very different distribution of the values across the vector elements. In Figure 2, we report the cumulative distribution related to the percentage of positives for an element all over the dataset. The graph shows that there is a 10% of the vector components that have positive values in less than 5% of the images, while there is 10% of elements that are positive in more than 40% of the images (that is, there is 90% of elements that are positive in less than 40% of the images). Overall, the results show that the sparsity is not equally spread across the feature element. Moreover, there are neurons which are almost never activated by the input images. The opposite, neurons activated by almost any image, do not exist.

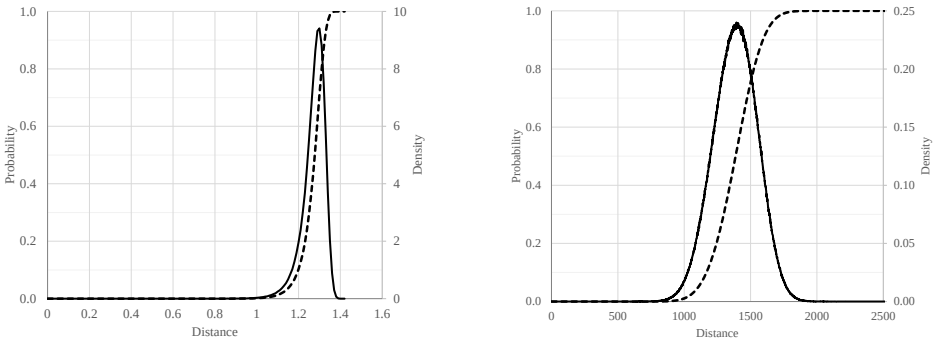
We now focus on the distribution of distances. In Figure 3, we report the cumulative and probability density functions for both the Euclidean distance applied to the *ReLU-L2Norm* features (a) and the Hamming distance applied to the features of *Binary*. In Table 1, we report the metric space intrinsic dimensionality [9] defined as $\mu^2/(2\sigma^2)$, and other information also related to these distributions.

Table 1: Intrinsic Dimensionality

	Euclidean	Hamming
Intrinsic Dimensionality ($\mu^2/(2\sigma^2)$)	276	35
Variance	0.0029	27057
Standard Deviation (σ)	0.054	164.5
Mean (μ)	1.27	1383
Mode	1.28	1388

The *ReLU-L2Norm* features in conjunction with the Euclidean distance appear to be very hard to index. The course of dimensionality is revealed in the graph and confirmed by the high *intrinsic dimensionality*. On the contrary, the *Binary* features combined with the Hamming distance reveal an intrinsic dimensionality of only 35 and the distribution is very similar to a Gaussian.

In Figure 4, we analyse the amount of intersection between the results of the 1,000 k-NN queries we performed by sequentially scanning the dataset for the aim of creating the ground-truths that we made public available on our website. We compare the results obtained with the *ReLU-L2Norm* and *Binary* features reporting the average intersection varying k. We also considered the cases in which the query is between the results or is removed. When the query is in the result, at least the query itself is in the intersection. So, for instance, we have



(a) Euclidean (b) Hamming

Fig. 3: Cumulative distribution function (probability density as dotted line) for both Euclidean and Hamming distances

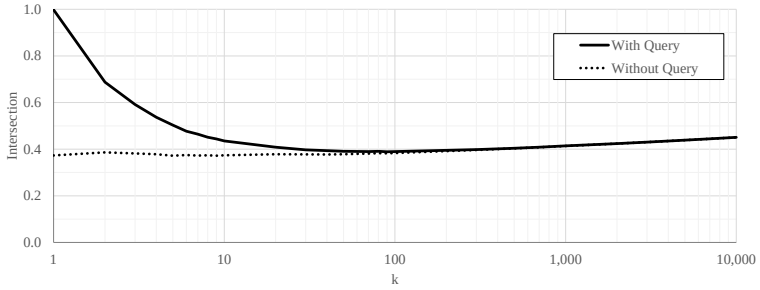


Fig. 4: Intersection between Euclidean and Hamming k-NN results varying k used for the k-NN search (with and without the query in the dataset).

100% intersection for $k = 1$. The most interesting curve is the one in which we do not consider the query itself in the results. We obtained an intersection of about 0.4 for k between 1 and 1,000. It is also worth to note that the intersection increase with k.

In Figure 5, we consider the distance of the results in our k-NN queries over the *ReLU-L2Norm* feature varying k reporting *min*, *mean*, *max*, *10-th* and *90-th* percentiles. The graph shows that 90% of the first 10 results for each query have a distance below 1.0. Unfortunately this distance is the same we obtain as mean for the 10,000 result. Moreover, for 10% of the queries we have a first result at a distance greater than 0.9, which is the mean value we obtained for the result at position 1,000. In other words, while deep features have been proven to be effective in ranking, the value of the distance itself does not appear to be meaningful. This is also an effect of the high intrinsic dimensionality of the space and the curse of dimensionality.

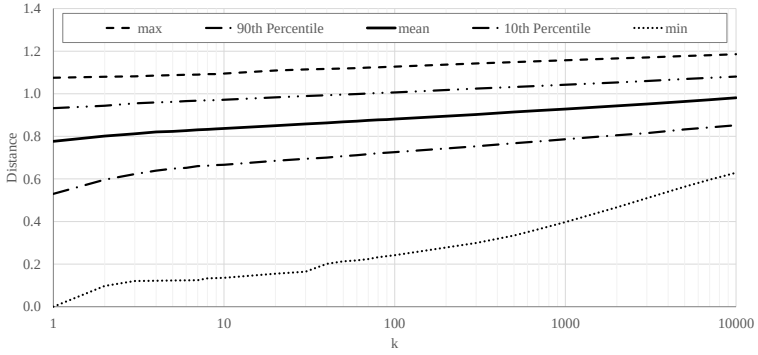


Fig. 5: *Mean*, *min*, *max*, 10-th and 90-th the k-th results distances.

4.1 Online CBIR systems using YFCC100M-HNfc6

We have also created and put on-line two different CBIR systems that use the YFCC100M-HNfc6 benchmark. One system is based on the the Metric Inverted File (MI-File) technique [4]. MI-File uses an inverted file to store relationships between permutations, and some approximations and optimizations to improve both efficiency and effectiveness. The basic idea is that entries (the lexicon) of the inverted file are the set of permutants (or pivots) P. The posting list associated with an entry $p_i \in P$ is a list of pairs $(o, \Pi_o^{-1}(i))$, $o \in C$, i.e. a list where each object o of the dataset C is associated with the position of the pivot p_i in Π_o.

The second system is based on the *LuQ* approach [3]. *LuQ* represents each DNCC feature as a text document and uses a NoSQL database (Apache Lucene) for efficiently indexing and searching purposes. It exploits the quantization of the vector components of the DCNN features, in which each real-valued vector component x_i is transformed in a natural numbers n_i given by $\lfloor Qx_i \rfloor$; where $\lfloor \rfloor$ denotes the floor function and Q is a multiplication factor > 1 that works as a *quantization factor*. n_i are then used as term frequencies for the "term-components" of the text document representing the feature vectors.

All the 97M features vectors of YFCC100M-HNfc6 were indexed using MI-File and LuQ approaches. The corresponding on-line demos are available at `http://mifile.deepfeatures.org` and `http://melisandre.deepfeatures.org`.

The whole Lucene 5.5 archive of LuQ approach is also available for download from the deep features website [1]. The advantage of this representation is that can be directly queried with Lucene by simply extracting the term vectors from the archive.

5 Performance evaluation of similarity search techniques

Performance assessment and comparison of similarity search techniques requires, in addition to a common dataset, also a common methodology to the execution

of the experiments. Experiments must be executed using objective measures and should be reproducible so that other researcher can validate them and compare against other techniques.

In the following, we discuss some useful performance measures that can be used to assess similarity search methods. Then, we show an example of performance assessment of a similarity search technique using the YFCC100M-HNfc6 benchmark and some of these performance measures.

Performance measures can be broadly classified into measures for assessing the efficiency and measures for assessing the accuracy of similarity search algorithms. Exact similarity search algorithms, that is algorithms that retrieve all objects that satisfy a similarity query, require just assessment of their efficiency. However, many popular similarity search methods are approximate, i.e., might loose some qualifying objects and can retrieve some non-qualifying objects. In this case, therefore, the assessment of the quality of the results is relevant, to determine the trade-off between efficiency and accuracy.

A very obvious measure to assess efficiency of similarity search algorithms is measuring the *average query processing time* computed on a reasonable number of different queries. Unfortunately, this measures is not easily reproducible and objective. Query processing time depends on several aspects that are not easily controllable. Different hardware architectures, software installed, operating system, running environments, programming languages, can significantly affect the results.

A more objective measure, when data to be searched cannot be stored in main memory, is the *number of disk block reads*. All hard disks read data in blocks, for instance of size 4K bytes. Reading more disk blocks means more time spent transferring data from disk to memory. This measure can give significant information both for traditional hard disks and for solid state disks. In case of traditional hard disks, it can be useful to distinguish between *consecutively stored disk block reads* and *randomly stored disk block reads*. Consecutively stored disk block reads do not require disk seeks, so they are order of magnitude faster than randomly stored disk block reads. However, this differentiation is not relevant in case of solid state disks, where there is practically no disk seek cost.

Another useful objective measure is the *number of object reads*. Similarity search algorithms generally access data from the disk either for accessing data structures of the index or for retrieving objects to be checked against the query conditions. The number of object reads gives an idea of the improvement of performance with respect to an exhaustive sequential scan of the entire dataset. A similar performance measure is the *number of distance computations*. In many realistic cases, the distance computation has an high cost as well. Counting the number of computed distances is useful as well, especially in those cases where part or all objects are stored in main memory.

Approximate similarity search methods offer improvement of efficient of some orders of magnitude with respect to exact similarity search algorithms, at the expense of some degradation of the accuracy. To assess the quality of approximate similarity search methods, a ground-truth must be available for the dataset, built

using exact similarity search queries. In YFCC100M-HNfc6 the ground-truth is composed of 1,000 different queries for which the 10,001 nearest neighbours were retrieved. Ground-truth was generated using the entire dataset, and also smaller portions with size multiple of one million, to allow researchers to both test on a portion of the dataset and study scalability. A discussion on measures for assessing the quality of approximate similarity search can be found in [22]. Here we report some of those measures.

Two popular measures to assess quality of search results are the *Precision* and *Recall* measures. These can also be effectively used in our case. Let Q be a similarity query, for instance a range search or a k-NN search. Let ER_Q be the sorted exact similarity search result set, and AR_Q be the sorted approximate similarity search result set. Let $|\cdot|$ denote the size of a set. The precision P is the ratio between the number of correct results in the approximate result set, by the total size of the approximate result set:

$$P = \frac{|AR_Q \cap ER_Q|}{|AR_Q|}.$$

The recall R is the ratio between the number of correct results in the approximate result set and the number of correct results that should have been retrieved:

$$R = \frac{|AR_Q \cap ER_Q|}{|ER_Q|}.$$

An additional useful way to assess the quality of approximate results is to evaluate the discrepancy between the sorted approximate result set and the exact result set. This can be measured in terms of the difference in position of the objects between these sets. Let X be the entire dataset, X_Q the entire dataset sorted according to the distance from query object in query Q, $o = AR_Q[i]$ is the i-th object in the sorted approximate result set AR_Q, $X_Q(o)$ is the position of object o in sorted sets X_Q. The *Error on the Position*, EP, can be defined as a normalized version of the Induced Sperman Footrule distance as follows:

$$EP = \frac{\sum_{i=1}^{|AR_Q|} |X_Q(AR_Q[i]) - i|}{|AR_Q| \cdot |X|}.$$

The error on position measures the difference in positions between the approximate result and the exact result, averaged for all retrieved objects, and normalized dividing by the size of the dataset. Suppose the error on position is $EP = 10^{-5}$, on a dataset of 1 millions objects. This means that on average, the difference in position of retrieved objects, between the approximate and exact results, is $10^{-5} \cdot 1,000,000 = 10$.

Clearly all above measures should be averaged on several queries. As we said before, YFCC100M-HNfc6 offers k-NN results for 1,000 queries, so these measures can be averaged on those queries.

5.1 Performance Evaluation Example

In the following, we show some an example of performance assessments obtained using the full binary YFCC100M-HNfc6 benchmark and some of the performance measures mentioned in the previous section. The method that we test is the MI-File approximate similarity search index [4]. MI-File is a permutation based method that uses inverted files to perform fast approximate execution of k-NN queries. MI-File offers the following parameters to trade efficiency with accuracy:

- Amplification factor *amp*: when searching for the k-NN the MI-File retrieves a candidate set of $k' = amp \cdot k$ objects, reorders it according to the original distance function, and returns the top-k objects. The larger *amp*, the higher the search cost, and the higher the accuracy.
- data object permutation length k_i: the permutation representing a data object is obtained using the the k_i closest reference objects out of the total set of reference objects. The value of k_i determines the number of posting lists containing a reference to the object being inserted.
- Query permutation length k_s: the permutation representing the query is obtained using the the k_s closest reference objects out of the total set of reference objects. The value of k_s determines the number of posting lists accessed during a query execution.
- Maximum position difference *mpd*: posting lists are scanned considering entries referring objects whose reference objects position difference in their permutation, with respect to the query permutation, is at most *mpd*. The higher *mpd*, the more entries are retrieved from the posting lists.

Please see [4] for further details on the MI-File and its parameters usage.

In our experiments, we indexed the entire binary YFCC100M-HNfc6 dataset, using $k_i = 100$. The total number of reference objects for building permutations is 20,000. The queries were executed with *amp* ranging from 1 to 70. The values used for k_s ranged from 1 to 50 and $mpd = k_s$. We executed 100-NN queries using the 1,000 queries of the ground truth, and performance measures were obtained as average of the measures computed for all query. Results are shown in Figure 6.

The two upper figures show the relationships between the number of disk blocks accessed and the quality of results. Disk block size is 4K bytes. Every plot corresponds to different setting for *amp*, and the amount of disk blocks accessed was tuned by setting the k_s parameter. MI-File reaches a recall of 75% with a number of disk block accesses around 30,000. For the same disk access cost, we have an error on position of $2 \cdot 10^{-6}$. This means that the on average the difference in position of retrieved objects, between the approximate and exact results, on a dataset of 100M objects, is around 200 positions. However, given that the recall is 0.75%, so 75 out of 100 objects are correctly retrieved and their difference in position can be at most 25, most of the position error is due to the few objects (25%) that were erroneously retrieved in place of the exact results.

The two bottom graphs show the relationships between the number of database objects accessed and the quality of the results. Here, the index access cost is not

taken into account. In this case, 7,000 objects out of 100M total objects have to be accessed to have a recall of almost 75% and a position error of $2 \cdot 10^{-6}$.

An objective estimation of the average query processing time can be obtained from the number of disk blocks accessed and the Input/Output Operations Per Second (IOPS) of the disk being used. For instance with a solid state disk having 8,600 IOPS, the expected elapsed time of a query reading 30,000 disk blocks of 4K each, is around 3.5 seconds.

It is also worth mentioning that, in this example of performance evaluation, precision is always equal to recall. In fact the denominator, in the precision and recall definitions, is equal to the total number of retrieved objects, which is $k = 100$ both for approximate and exact search, and the numerator is always the number of correct objects retrieved.

6 Conclusion

In this paper, we presented YFCC100M-HNfc6: a benchmark for evaluating content-based image retrieval systems consisting of 97M deep features extracted from the YFCC100M dataset. Together with detailed statical information for the proposed dataset, we reported performance assessment and comparison that can be used to assess similarity search methods with common methodology to the execution of the experiments. The benchmark is publicly available on the Deep Features website [1].

Acknowledgments

This work was partially founded by: EAGLE, Europeana network of Ancient Greek and Latin Epigraphy, co-founded by the European Commision, CIP-ICT-PSP.2012.2.1 - Europeana and creativity, Grant Agreement n. 325122; and Smart News, Social sensing for breakingnews, co-founded by the Tuscany region under the FAR-FAS 2014 program, CUP CIPE D58C15000270008.

References

1. Deep features. `http://www.deepfeatures.org`, accessed: 2016-05-23
2. The multimedia commons initiative. `https://multimediacommons.wordpress.com/`, accessed: 2016-05-23
3. Amato, G., Debole, F., Falchi, F., Gennaro, C., Rabitti, F.: Large scale indexing and searching deep convolutional neural network features. In: Proceeding of the 18th International Conference on Big Data Analytics and Knowledge Discovery (DaWaK 2016). Springer (2016), to appear
4. Amato, G., Gennaro, C., Savino, P.: MI-File: using inverted files for scalable approximate similarity search. Multimedia Tools and Applications 71(3), 1333–1362 (2014), `http://dx.doi.org/10.1007/s11042-012-1271-1`

5. Azizpour, H., Razavian, A., Sullivan, J., Maki, A., Carlsson, S.: From generic to specific deep representations for visual recognition. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops. pp. 36–45 (2015)
6. Babenko, A., Slesarev, A., Chigorin, A., Lempitsky, V.: Neural codes for image retrieval. In: Computer Vision–ECCV 2014, pp. 584–599. Springer (2014)
7. Bolettieri, P., Esuli, A., Falchi, F., Lucchese, C., Perego, R., Piccioli, T., Rabitti, F.: CoPhIR: a test collection for content-based image retrieval. CoRR abs/0905.4627v2 (2009), http://cophir.isti.cnr.it
8. Chandrasekhar, V., Lin, J., Morère, O., Goh, H., Veillard, A.: A practical guide to cnns and fisher vectors for image instance retrieval. arXiv preprint arXiv:1508.02496 (2015)
9. Chávez, E., Navarro, G., Baeza-Yates, R., Marroquín, J.L.: Searching in metric spaces. ACM computing surveys (CSUR) 33(3), 273–321 (2001)
10. Chavez, G., Figueroa, K., Navarro, G.: Effective proximity retrieval by ordering permutations. Pattern Analysis and Machine Intelligence, IEEE Transactions on 30(9), 1647 –1658 (sept 2008)
11. Donahue, J., Jia, Y., Vinyals, O., Hoffman, J., Zhang, N., Tzeng, E., Darrell, T.: Decaf: A deep convolutional activation feature for generic visual recognition. arXiv preprint arXiv:1310.1531 (2013)
12. Gennaro, C., Amato, G., Bolettieri, P., Savino, P.: An approach to content-based image retrieval based on the lucene search engine library. In: Research and Advanced Technology for Digital Libraries, pp. 55–66. Springer (2010)
13. Jia, Y., Shelhamer, E., Donahue, J., Karayev, S., Long, J., Girshick, R., Guadarrama, S., Darrell, T.: Caffe: Convolutional architecture for fast feature embedding. arXiv preprint arXiv:1408.5093 (2014)
14. Krizhevsky, A., Sutskever, I., Hinton, G.E.: Imagenet classification with deep convolutional neural networks. In: Advances in neural information processing systems. pp. 1097–1105 (2012)
15. LeCun, Y., Bengio, Y., Hinton, G.: Deep learning. Nature 521(7553), 436–444 (2015)
16. Mohamed, H., Marchand-Maillet, S.: Quantized ranking for permutation-based indexing. Information Systems 52, 163–175 (2015)
17. Razavian, A.S., Azizpour, H., Sullivan, J., Carlsson, S.: CNN features off-the-shelf: an astounding baseline for recognition. In: Computer Vision and Pattern Recognition Workshops (CVPRW), 2014 IEEE Conference on. pp. 512–519. IEEE (2014)
18. Razavian, A.S., Sullivan, J., Maki, A., Carlsson, S.: A baseline for visual instance retrieval with deep convolutional networks. arXiv preprint arXiv:1412.6574 (2014)
19. Sermanet, P., Eigen, D., Zhang, X., Mathieu, M., Fergus, R., LeCun, Y.: Overfeat: Integrated recognition, localization and detection using convolutional networks. arXiv preprint arXiv:1312.6229 (2013)
20. Simonyan, K., Zisserman, A.: Very deep convolutional networks for large-scale image recognition. arXiv preprint arXiv:1409.1556 (2014)
21. Thomee, B., Elizalde, B., Shamma, D.A., Ni, K., Friedland, G., Poland, D., Borth, D., Li, L.J.: Yfcc100m: The new data in multimedia research. Communications of the ACM 59(2), 64–73 (2016)
22. Zezula, P., Amato, G., Dohnal, V., Batko, M.: Similarity Search: The Metric Space Approach, Advances in Database Systems, vol. 32. Springer-Verlag (2006)
23. Zhou, B., Lapedriza, A., Xiao, J., Torralba, A., Oliva, A.: Learning deep features for scene recognition using places database. In: Advances in neural information processing systems. pp. 487–495 (2014)

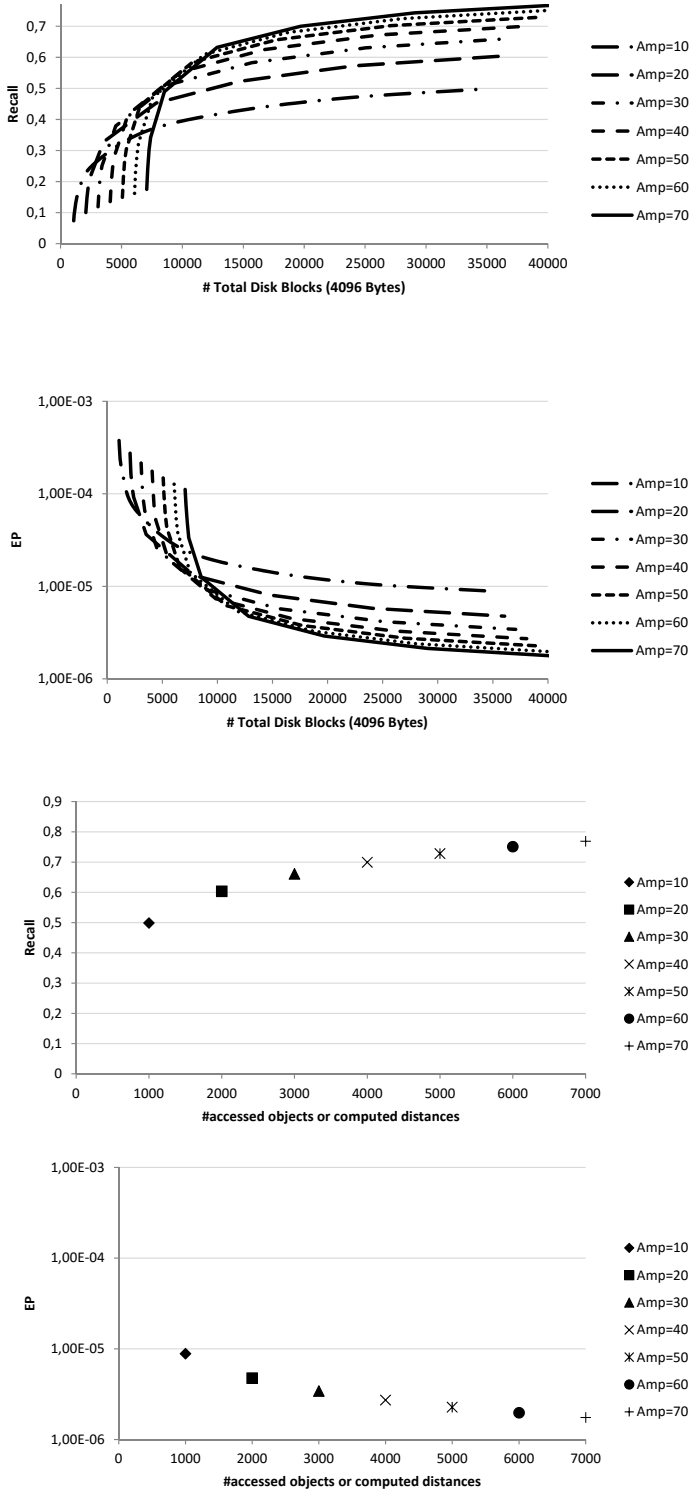


Fig. 6: Experiments executed indexing the YFCC100M-HNfc6 *binary* features with the MI-File considering k-NN search with $k = 100$