Large Scale Indexing and Searching Deep Convolutional Neural Network Features

Giuseppe Amato, Franca Debole, Fabrizio Falchi,
Claudio Gennaro, and Fausto Rabitti *

ISTI-CNR
Via G. Moruzzi 1
56124 Pisa - Italy
name.surname@isti.cnr.it

Abstract. Content-based image retrieval using Deep Learning has become very popular during the last few years. In this work, we propose an approach to index Deep Convolutional Neural Network Features to support efficient retrieval on very large image databases. The idea is to provide a text encoding for these features enabling the use of a text retrieval engine to perform image similarity search. In this way, we built *LuQ* a robust retrieval system that combines full-text search with content-based image retrieval capabilities. In order to optimize the index occupation and the query response time, we evaluated various tuning parameters to generate the text encoding. To this end, we have developed a web-based prototype to efficiently search through a dataset of 100 million of images.

Keywords: Convolutional Neural Network, Deep Learning, Inverted Index, Image Retrieval

1 Introduction

Deep Convolutional Neural Networks (DCNNs) have recently shown impressive performance in the Computer Vision area, such as image classification and object recognition [15, 21, 9]. The activation of the DCNN hidden layers has been also used in the context of transfer learning and content-based image retrieval [6, 19]. In fact, 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" [16]. These representations can be successfully used as *features* in generic recognition or visual similarity search tasks.

* This work was partially founded by: EAGLE, Europeana network of Ancient Greek and Latin Epigraphy, co-founded by the European Commission, 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.

The first layers of DCNN are typically useful in recognizing low-level characteristics of images such as edges and blobs, while higher levels have demonstrated to be more suitable for semantic similarity search. One major obstacle to the use of DCNN features for indexing large datasets of images is its internal representation, which is high dimensional leading to the curse of dimensionality [7]. For instance, in the well-known AlexNet architecture [15] the output of the sixth layer (fc6) has 4,096 dimensions, while the fifth layer (pool5) has 9,216 dimensions. An effective approach to tackle the dimensionality curse problem is the application of approximate access methods such as permutation-based indexes [5, 1].

A drawback of these approaches is on one hand the dependence of the index on a set of reference objects (or pivots); on the other hand the need to reorder the result set of the search according to the original feature distance. The former issue requires the selection of a set of reference objects, which represents well the variety of datasets that we want to index. The latter issue forces to use a support database to store the features, which requires efficient random I/O on a fast secondary memory (such as flash-based storages).

In this paper, we propose an approach to specifically index DCNN Features to support efficient content-based on large datasets, which we refer to as *LuQ*. The proposed approach exploits the ability of inverted files to deal with the sparsity of the Convolutional features. To this end, we make use of the efficient and robust full-text search library Lucene[1]. The idea is to associate each component of the feature vector with a unique alphanumeric keyword and to generate a textual representation in which we boost the relative term proportionally to its intensity.

A browser-based application that provides an interface for combined textual and visual searching into a dataset of about 100 million of images is available at `http://melisandre.deepfeatures.org`.

The paper is organized as follows. Section 2 makes a survey of the related works. Section 3 presents the proposed approach. Section 4 discusses the validation tests. Section 5 concludes.

2 Related Work

Recently, a new class of image descriptors, built upon Deep Convolutional Neural Networks (DCNNs), have been used as effective alternative to descriptors built using local features such as SIFT, SURF, ORB, BRIEF, etc. Starting from 2012 [15], DCNNs have attracted enormous interest within the Computer Vision community because of the state-of-the-art results achieved in the image classification challenge ImageNet Large Scale Visual Recognition Challenge (ILSVRC). In Computer Vision, DCNN have been used to perform several tasks, including not only image classification, but also image retrieval [6, 2] and object detection [8], to cite some. In particular, it has been proved that the multiple levels of representation which are learned by DCNN on specific task (typically supervised)

[1] `http://lucene.apache.org`

can be used to transfer learning across tasks. This means that the activation of neurons of a specific layers, preferably the last ones, can be used as features for describing the visual content [19].

Liu et al. [17] proposed a framework that adapts Bag-of-Word model and inverted table to DCNN feature indexing, which is similar to LuQ. However, for large-scale datasets, Liu et al. have to build a large-scale visual dictionary that employs product quantization method to learn a large-scale visual dictionary from a training set of global DCNN features. In any case, using this approach the authors reported a search time that is one order higher than in our case for the same dataset.

3 Text Encoding for Deep Convolutional Neural Network Features

3.1 Feature Indexing

The main idea of LuQ is to index DCNN features using a text encoding that allows us to use a text retrieval engine to perform image similarity search. As discussed later, we implemented this idea on top of the Lucene text retrieval engine; however any full-text engine supporting vector space model can be used for this purpose.

In principle, to perform similarity search on DCNN feature vectors we should compare the vector extracted from the query with all the vectors extracted from the images of the dataset and take the nearest images according to L2 distance. This is sometimes called brute-force approach. However, if the database is large, brute-force approach can become time-consuming.

Starting from this approach, we observed that given the sparsity of the DCNN features, which contain mostly zeros (about 75%), we are able to use a well-known IR technique, i.e., an inverted index.

In fact, we note that since typically DCNN feature vectors exhibits better results if they are L2-normalized to the unit length [20, 4], if two vectors $\mathbf{x}$ and $\mathbf{y}$ have length equal to 1 (such as in our case), the following relationship between the L2 distance $d_2(\mathbf{x}, \mathbf{y})$ and the inner product $\mathbf{x} * \mathbf{y}$ exists:

$$d_2(\mathbf{x}, \mathbf{y})^2 = 2(1 - \mathbf{x} * \mathbf{y})$$

The advantage of computing the similarity between vectors in terms of inner product is that we can efficiently exploit the sparsity of the vectors by accumulating the product of non-zeroes entries in the vector $\mathbf{x}$ and their corresponding non-zeros entries in the vector $\mathbf{y}$. Moreover, Lucene, as other search engines, computes the similarity between documents using the cosine similarity, which is the inner product of the two vectors divided by their lengths product. Therefore, in our case, cosine similarity and inner product are the same. Ascertained this behave, our idea is to fill the inverted index of Lucene with the DCNN

features vectors. For space-saving reasons, however, text search engines do not store float numbers in the posting entries of the inverted index representing documents, rather they store the term frequencies, which are represented as integers. Therefore, we must guarantee that posting entries will contain numeric values proportional to the float values of the deep feature entries.

To employ this idea, in LuQ, we provide a text encoding for the DCNN feature vectors that guarantees the direct proportionality between the feature components and the term frequencies. Let $\mathbf{w} = (w_1, \ldots, w_m)$ denote the L2-normalized DCNN vector of m dimensions. Firstly, we associated each of its component w_i with a unique alphanumeric term τ_i (for instance, the prefix 'f' followed by the numeric values corresponding to the index i). The text encoding $doc(\mathbf{w})$ corresponding to the vector $\mathbf{w}$ is given by:

$$doc(\mathbf{w}) = \bigcup_{i=1}^{m} \bigcup_{j=1}^{\lfloor Qw_i \rfloor} \tau_i$$

Where $\lfloor\rfloor$ denotes the floor function and Q is a multiplication factor > 1 that works as a *quantization factor*[2].

Therefore, we form the text encoding of w_i by repeating the term τ_i for the non-zero components a number of times directly proportional to w_i. This process introduces a quantization error due to the representation of float components in integers. However, as we will see, this error does not affect the retrieval effectiveness. The accuracy of this approximation depends on the factor Q, used to transform the vector $\mathbf{w}$. For instance, if we fix $Q = 2$, for $w_i < 0.5$, $\lfloor Qw_i \rfloor = 0$, while for $w_i \geq 0.5$, $\lfloor Qw_i \rfloor \geq 1$. In contrast, the smaller we set Q the smaller the inverted index will be. This is because the floor function will set to zero more entries of the posting lists. Hence, we have to find a good trade-off between the effectiveness of the retrieval system and its space occupation.

For example, if we set $Q = 30$ and we have for instance a feature vector with just three components $\mathbf{w} = (0.01, 0.15, 0.09)$ the corresponding integer-representation of the vector will be $(0, 4, 2)$ and its text encoding will be: $doc(\mathbf{w})$ ="f2 f2 f2 f2 f3 f3".

Since on average the 25% of the DCNN features are non-zero (in our specific case the fc6 layer), the size of their corresponding text encoding will have a small fraction of the unique terms present in the whole dictionary (composed of 4,096 terms). In our case, on average a document contains about 275 unique terms, which is about 6.7% of the dictionary because of quantization that set to zero the feature components smaller than $1/Q$.

3.2 Query Reduction

When we have to process similarity search, therefore the search engine have to treat query of that size. These unusual long queries, however, can affect the response time if the inverted index contains million of items.

[2] By abuse of notation, we denote the space-separated concatenation of keywords with the union operator $\cup$.

A quite intuitive way to overcome this issue is to reduce the size of the query by exploiting the knowledge of the *tf*idf* (i.e., term frequency * inverse document frequency) statistic of the text encoding, which comes for free in standard full-text retrieval engines. We can retain the elements of the query that exhibit greater values of *tf*idf* and eliminate the others. For instance, for a query of about 275 unique term on average, we can take the first ten terms that exhibits the highest *tf*idf*, we obtain a query time reduction of about 96%.

This query reduction comes, however, with a price: it decreases the precision of results. To attenuate this problem, for a top-k query, we reorder the results using the cosine similarity between the original query (i.e., the one without reduction) and the first $C_r \times k$ candidates documents retrieved. Where C_r is an amplification factor that we refer to as *reordering factor*. For instance, if we have to return $k = 100$ images and we set $C_r = 10$, we take and reorder the first $10 \times 100 = 1000$ candidate documents retrieved by the reduced query.

In order to calculate the cosine similarity of the original query and the $C_r \times k$ candidates, we have to reconstruct the quantized features by accessing to the posting list of the document returned by the search engine. As we will see, this approach does not affect significantly the efficiency of the query but can offer great improvements in terms of effectiveness.

Figure 1 summarizes the indexing and searching phases of LuQ.

4 Experiments

4.1 Setup

In order to test efficiency of LuQ, we used the Yahoo Flickr Creative Commons 100 Million (YFCC100M) dataset[3]. This dataset was created in 2014 as part of the Yahoo Webscope program. YFCC100M consists of 99.2 million photos and 0.8 million videos uploaded to Flickr between 2004 and 2014 and Creative Commons commercial or noncommercial license. More information about the dataset can be found in the article in Communications of the ACM [22].

For extracting deep features we used the Caffe [14] deep learning framework. In particular we used the neural network Hybrid-DCNN whose model and weights are public available in the Caffe Model Zoo[4]. The Hybrid-DCNN was trained on 1,183 categories (205 scene categories from Places Database and 978 object categories from the train data of ILSVRC2012 (ImageNet) with 3.6 million images [23]). The architecture is the same as Caffe reference network.

The deep features we have extracted are the activations of the fc6 layer. We have made them public available at `http://www.deepfeatures.org` and they will be soon included in the Multimedia Commons initiative corpus.

A ground-truth is not yet available for the YFCC100M dataset. Therefore, effectiveness of the proposed approach was evaluated using the INRIA Holidays

[3] `http://bit.ly/yfcc100md`
[4] `http://github.com/BVLC/caffe/wiki/Model-Zoo`

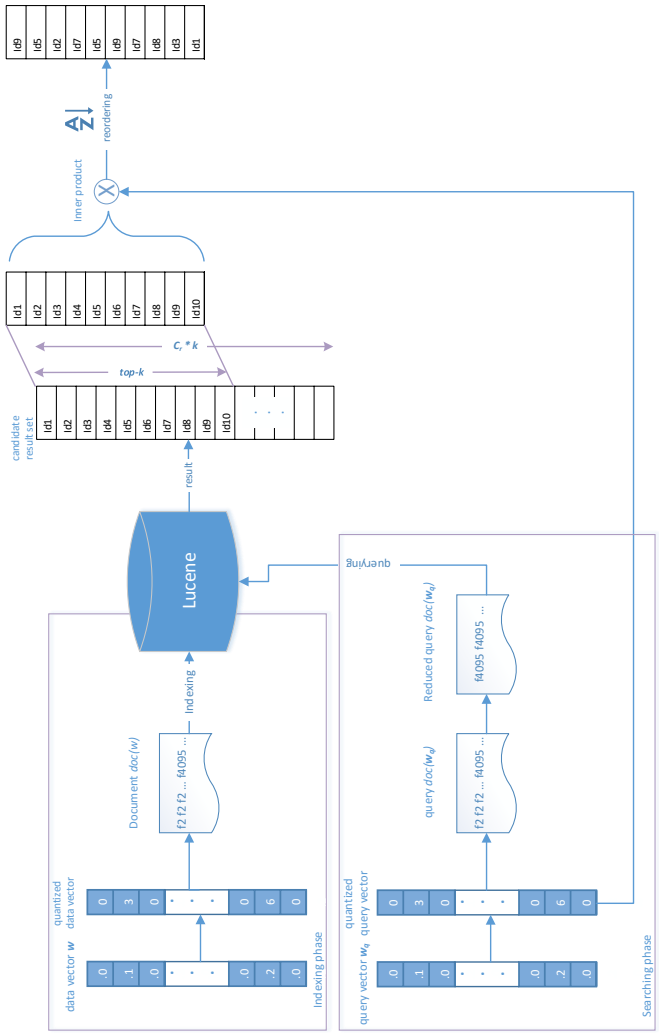

Fig. 1. Diagram showing the indexing and searching phases of LuQ.

dataset [13, 11]. It is a collection of 1,491 holiday images. The authors selected 500 queries and for each of them a list of positive results. As in [10, 12, 11], to evaluate the approach on a large scale, we merged the Holidays dataset with the

Flickr1M collection[5]. We extracted DCNN features also from these datasets in order to test our technique.

All experiments were conducted on a Intel Core i7 CPU, 2.67 GHz with 12.0 GB of RAM a 2TB 7200 RPM HD for the Lucene index. We used Lucene v5.5 running on Java 8 64 bit.The quality of the retrieved images is typically evaluated by means of precision and recall measures. As in many other papers [10, 18, 11], we combined this information by means of the mean Average Precision (mAP), which represents the area below the precision and recall curve.

4.2 Effectiveness and Quantization factor Q

In a first experimental analysis, we have evaluated the optimal value of Q over the Flickr1M dataset. As explained above, by keeping the value Q to the minimum, we can reduce the space occupation of the inverted index. Figure 2 shows the mAP as function of Q and the corresponding space occupation of the inverted index. From this analysis, we conclude that an optimal choice of the quantization factor Q is 30, which leads to a mAP of 0.62 and a space occupation of 2.31GB. We stress that the mAP using the brute-force approach (on the exact DCNN feature vectors) is about 0.60. This means that our quantization error leads to a slight improvement of the precision for $Q \geq 30$. Another important aspect is that this effectiveness was obtained forcing Lucene to use the standard inner product on *tf* weight without *idf* and any other document normalization. A further improvement can be obtained using the similarity function of Lucene called LMDirichlet, which provided a mAP of about 0.64.

Figure 3 shows the document frequency distribution, in the Flickr1M dataset, of the terms τ_i (i.e., component w_i), sorted in decreasing order. As can be seen, the distribution is quite skewed and some terms are much more frequent than others ranging from 313 to 378,876 in a collection of about 1 million features. This aspect has some impact on the performance of the inverted index, since it means that the posting list of the term τ_x has 378,876 items since it appears in many image features.

The observation about document frequency leads to the idea of using *tf*idf* to reduce the query length by cutting off terms with lower *tf*idf* weight. Since in inverted files the query time is usually proportional with the length of the query [3], this approach gives a great improvement in terms of query response time.

Figure 4 shows the mAP values at different levels of reordering factors C_r and query lengths L_q. Note that, $C_r = 0$ means no reordering, $C_r = 1$, reordering of the first k candidates, $C_r = 2$, reordering of the first $2k$ candidates, and so on. Concerning, L_q, we have considered a range of values between 2 and 50. Since the average document length is about 275, this corresponds to an average length reduction from 0.3% to 80%. In the graph of Figure 4, we have also plotted the mAP level obtained with query without reduction (namely fullQ) and the mAP obtained with sequential scan using L2 (brute-force). In all experiments we have

[5] `http://press.liacs.nl/mirflickr/`

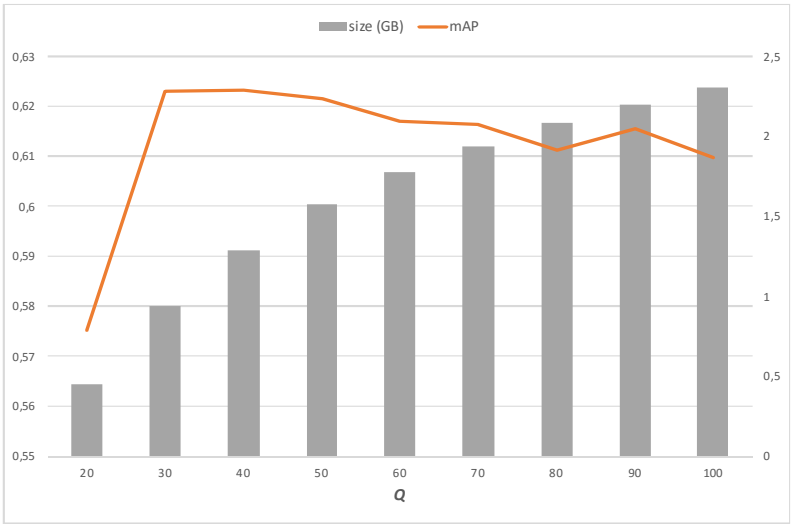

Fig. 2. Effectiveness (mAP) vs space occupation for increasing values of the quantization factor Q.

set $k = 100$. As these experiments show, the configuration $C_r = 10$ and $L_q = 8$ exhibits a mAP comparable to the case of brute-force using L2.

4.3 Evaluation of the Efficiency

In order to evaluate the efficiency of LuQ, in Figure 5, we have plotted the average search time for the same queries of the previous experiments. Please, note that the y-axis is in logarithmic scale. When there is no reordering (i.e., $C_r = 0$), the length of the query has a great impact to the search time (more than one order of magnitude). For increasing values of the the reordering factor C_r, L_q has a lower and lower influence on the search time. Clearly, increasing C_r, we increase the cost of the search. However, as we can see, there is still a big improvement in efficiency even for the case in which the reorder factor is maximum, i.e., $C_r = 10$.

In order to further validate LuQ, we test our index on the much larger dataset YFCC100M. We used the DCNN features of the fc6 layer and indexed them with Lucene using a quantization factor $Q = 30$. Since for this dataset a ground-truth is not available, we only reported the performance of the query in terms of average search time (see Figure 6). From this experiment, we see that for instance for the configuration $L_q = 10$ and $C_r = 10$, we have an average query time of less than 4 seconds (without any parallelization), which is quite encouraging considering that for the same dataset the query time was of the order of 10 minutes using the brute-force approach. In this case, we observe that C_r does not practically affect the efficiency of LuQ.

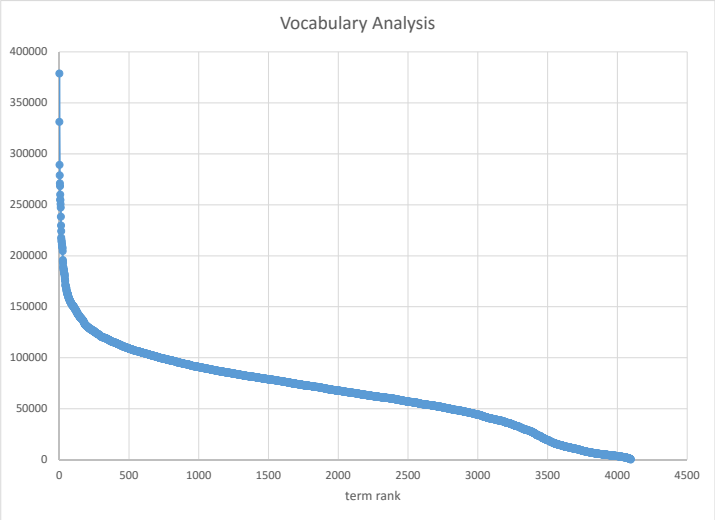

Fig. 3. Distribution of the vector components of the DCNN features ($Q = 30$).

5 Conclusions and Future Work

In this work, we propose LuQ, an efficient approach to build a CBIR, on top of a text search engine, specifically developed for Deep Convolutional Neural Network Features. This approach is very straightforward and does not demand costly elaborations during the indexing process, as for instance the permutation-based approaches such as [1], which requires to order a set of predefined reference features for each feature to be indexed. Moreover, in our approach, we can tune the query costs versus the quality of the approximation by specifying the length of the query, without the need of maintaining the original features for reordering the result set.

We evaluated different implementation strategies to balance index occupation, effectiveness, and the query response time. In order to show that our approach is able to handle large datasets, we have developed a browser-based application that provides an interface for combined textual and visual searching on a dataset of about 100 million of images, available at `http://melisandre.deepfeatures.org`. The whole Lucene 5.5 archive of LuQ approach is also available for download from the same location. This index can be directly queried by simply extracting the term vectors from the archive.

Acknowledgments

This work was partially founded by: EAGLE, Europeana network of Ancient Greek and Latin Epigraphy, co-founded by the European Commission, CIP-ICT-PSP.2012.2.1 - Europeana and creativity, Grant Agreement n. 325122; and

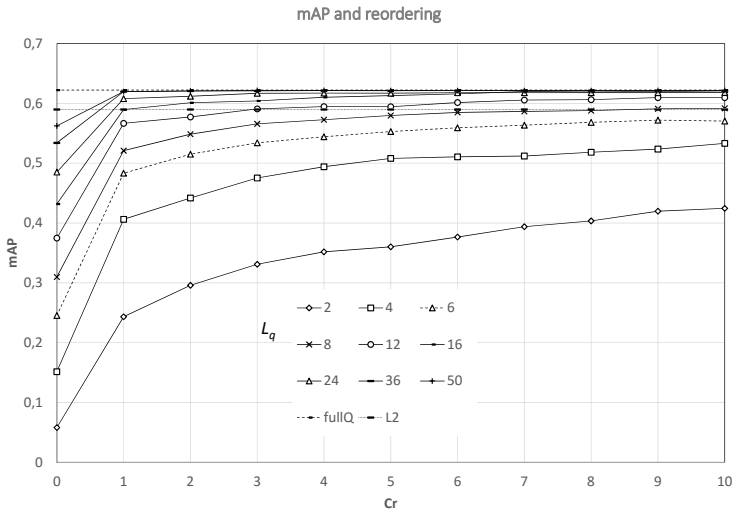

Fig. 4. Effectiveness (mAP) for various level of query lengths L_q and reordering factor C_r (with $k = 100$ and $C_r = 0$ means no reordering) vs the query without reduction (fullQ), and the mAP obtained with sequential scan using L2.

Smart News, Social sensing for breakingnews, co-founded by the Tuscany region under the FAR-FAS 2014 program, CUP CIPE D58C15000270008.

References

1. Giuseppe Amato, Claudio Gennaro, and Pasquale Savino. MI-File: using inverted files for scalable approximate similarity search. *Multimedia Tools and Applications*, 71(3):1333–1362, 2014.
2. Artem Babenko, Anton Slesarev, Alexandr Chigorin, and Victor Lempitsky. Neural codes for image retrieval. In *Computer Vision–ECCV 2014*, pages 584–599. Springer, 2014.
3. Stefan Büttcher, Charles LA Clarke, and Gordon V Cormack. *Information retrieval: Implementing and evaluating search engines*. Mit Press, 2010.
4. Ken Chatfield, Karen Simonyan, Andrea Vedaldi, and Andrew Zisserman. Return of the devil in the details: Delving deep into convolutional nets. *arXiv preprint arXiv:1405.3531*, 2014.
5. G.E. Chavez, K. Figueroa, and G. Navarro. Effective proximity retrieval by ordering permutations. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 30(9):1647 –1658, sept. 2008.
6. Jeff Donahue, Yangqing Jia, Oriol Vinyals, Judy Hoffman, Ning Zhang, Eric Tzeng, and Trevor Darrell. Decaf: A deep convolutional activation feature for generic visual recognition. *arXiv preprint arXiv:1310.1531*, 2013.
7. ZongYuan Ge, Chris McCool, Conrad Sanderson, and Peter Corke. Modelling local deep convolutional neural network features to improve fine-grained image

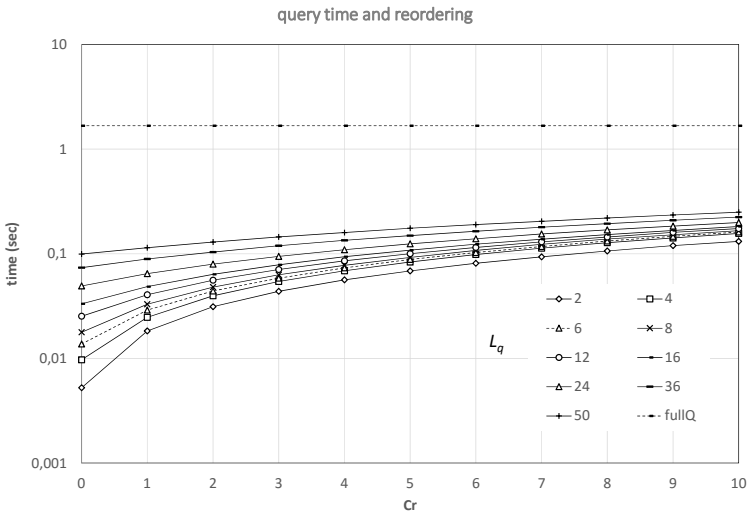

Fig. 5. Average search time (sec) for various level of query lengths L_q and reordering factor C_r (with $k = 100$ and $C_r = 0$ means no reordering) vs the query without reduction (fullQ), and the mAP obtained with sequential scan using L2.

classification. In *Image Processing (ICIP), 2015 IEEE International Conference on*, pages 4112–4116. IEEE, 2015.

8. Ross Girshick, Jeff Donahue, Trevor Darrell, and Jitendra Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 580–587, 2014.
9. Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. *arXiv preprint arXiv:1512.03385*, 2015.
10. H. Jégou, M. Douze, and C. Schmid. Packing bag-of-features. In *Computer Vision, 2009 IEEE 12th International Conference on*, pages 2357 –2364, 29 2009-oct. 2 2009.
11. H. Jégou, F. Perronnin, M. Douze, J. Sànchez, P. Pérez, and C. Schmid. Aggregating local image descriptors into compact codes. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 34(9):1704–1716, Sept 2012.
12. Hervé Jégou, Matthijs Douze, and Cordelia Schmid. Improving bag-of-features for large scale image search. *International Journal of Computer Vision*, 87:316–336, May 2010.
13. Hervé Jégou, Matthijs Douze, Cordelia Schmid, and Patrick Pérez. Aggregating local descriptors into a compact image representation. In *IEEE Conference on Computer Vision & Pattern Recognition*, pages 3304–3311, jun 2010.
14. Yangqing Jia, Evan Shelhamer, Jeff Donahue, Sergey Karayev, Jonathan Long, Ross Girshick, Sergio Guadarrama, and Trevor Darrell. Caffe: Convolutional architecture for fast feature embedding. *arXiv preprint arXiv:1408.5093*, 2014.
15. Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012.

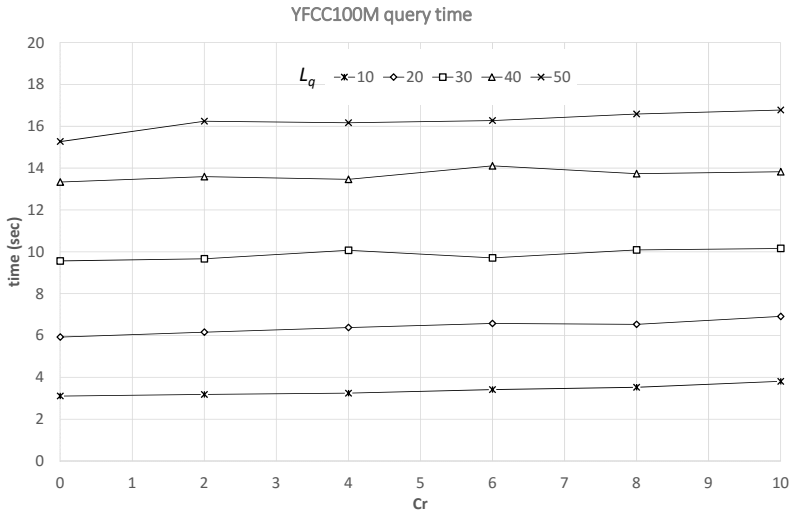

Fig. 6. Average search time (sec) for the YFCC100M dataset using lucene.

16. Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *Nature*, 521(7553):436–444, 2015.
17. Ruoyu Liu, Yao Zhao, Shikui Wei, Zhenfeng Zhu, Lixin Liao, and Shuang Qiu. Indexing of cnn features for large scale image search. *arXiv preprint arXiv:1508.00217*, 2015.
18. F. Perronnin, Yan Liu, J. Sanchez, and H. Poirier. Large-scale image retrieval with compressed fisher vectors. In *Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on*, pages 3384 –3391, june 2010.
19. Ali Razavian, Hossein Azizpour, Josephine Sullivan, and Stefan Carlsson. Cnn features off-the-shelf: an astounding baseline for recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pages 806–813, 2014.
20. Ali Sharif Razavian, Hossein Azizpour, Josephine Sullivan, and Stefan Carlsson. Cnn features off-the-shelf: An astounding baseline for recognition. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, June 2014.
21. Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.
22. Bart Thomee, Benjamin Elizalde, David A Shamma, Karl Ni, Gerald Friedland, Douglas Poland, Damian Borth, and Li-Jia Li. Yfcc100m: The new data in multimedia research. *Communications of the ACM*, 59(2):64–73, 2016.
23. Bolei Zhou, Agata Lapedriza, Jianxiong Xiao, Antonio Torralba, and Aude Oliva. Learning deep features for scene recognition using places database. In *Advances in neural information processing systems*, pages 487–495, 2014.