Large-Scale Image Retrieval with Elasticsearch

Giuseppe Amato, Paolo Bolettieri, Fabio Carrara, Fabrizio Falchi, Claudio Gennaro
ISTI-CNR, via G. Moruzzi 1, 56124, Pisa, Italy
{name.surname}@isti.cnr.it

ABSTRACT

Content-Based Image Retrieval in large archives through the use of visual features has become a very attractive research topic in recent years. The cause of this strong impulse in this area of research is certainly to be attributed to the use of Convolutional Neural Network (CNN) activations as features and their outstanding performance. However, practically all the available image retrieval systems are implemented in main memory, limiting their applicability and preventing their usage in big-data applications. In this paper, we propose to transform CNN features into textual representations and index them with the well-known full-text retrieval engine Elasticsearch. We validate our approach on a novel CNN feature, namely Regional Maximum Activations of Convolutions. A preliminary experimental evaluation, conducted on the standard benchmark INRIA Holidays, shows the effectiveness and efficiency of the proposed approach and how it compares to state-of-the-art main-memory indexes.

ACM Reference Format:
Giuseppe Amato, Paolo Bolettieri, Fabio Carrara, Fabrizio Falchi, Claudio Gennaro. 2018. Large-Scale Image Retrieval with Elasticsearch. In *SIGIR '18: The 41st International ACM SIGIR Conference on Research and Development in Information Retrieval, July 8-12, 2018, Ann Arbor, MI, USA.* ACM, New York, NY, USA, 4 pages. https://doi.org/10.1145/3209978.3210089

1 INTRODUCTION

Convolutional Neural Networks (CNNs) are increasingly used as feature extractors to support efficient Content-Based Image Retrieval (CBIR) systems. One of the obstacles to use these features is, however, in their high dimensionality, which prevents the use of standard space-partitioning data structures. For instance, in the well-known AlexNet architecture [13] the output of the sixth layer (fc6) has 4,096 dimensions. To overcome this problem, various partitioning methods have been proposed. For example, the inverted multi-index [6], which outperforms the state of the art by a large margin, uses product quantization both to define the coarse level and to code residual vectors combined with binary compressed techniques.

In this paper, we propose an alternative approach suitable to be implemented in secondary memory. We exploit a surrogate text representation of the features that allows us to exploit an existing text retrieval engine to build a content-based image retrieval system. The key idea is to represent the CNN features as permutations and, in turn, to transform them into surrogate text using the basic approach developed in [7].

The CNN feature that we have used in this work is the Regional Maximum Activation of Convolutions (R-MAC) [16], which is a very effective image representation for instance-level retrieval in CBIR systems. This feature is the result of the spatial aggregation of the activations of a convolution layer of a CNN and therefore robust to scale and translation. Gordo et al. [9] extended the R-MAC representation by improving the region pooling mechanism and including it in a differentiable pipeline trained end-to-end for retrieval tasks.

In this paper, we extend and improve our previous work on surrogate text representation [1] based on the approach of deep permutation [2], by adopting the Concatenated Rectified Linear Unit (CReLU) transformation [15]. The advantage of this approach is a better estimate of the matching score among R-MAC features by preserving both positive and negative activation information, which leads to an improvement of effectiveness at the same cost when using the conventional deep permutation approach. We have also improved the quality of the experimental evaluation providing a comparison with the FAISS library [12], which implements PQ-compressed inverted-file indexes in main memory. Moreover, we have tested the scalability of our solution, by distributing our index across multiple nodes with Elasticsearch (https://www.elastic.co).

The rest of the paper is organized as follows. Section 2 provides background for the reader. In Section 3, we introduce our approach to generate permutations for R-MAC features. Section 4 presents some experimental results using a real-life dataset. Section 5 concludes the paper.

2 BACKGROUND

Recently, a new class of image descriptor, built upon Convolutional Neural Networks, have been used as an effective alternative to descriptors built using local features such as SIFT, ORB, BRIEF, etc. CNNs have attracted an enormous interest within the Computer Vision community because of the state-of-the-art results achieved in challenging image classification tasks such as the ImageNet Large Scale Visual Recognition Challenge (http://www.image-net.org). In computer vision, CNNs have been used to perform several tasks, including image classification, as well as image retrieval [4, 5] and object detection [8], to cite some. Moreover, it has been proved that the representations learned by CNNs on specific tasks (typically supervised) can be transferred successfully across tasks [5, 14]. The activation of neurons of specific layers, in particular the last ones, can be used as features to semantically describe the visual content of an image.

Tolias et al. [16] proposed the R-MAC feature representation, which encodes and aggregates several regions of the image in a

SIGIR '18, July 8–12, 2018, Ann Arbor, MI, USA

ACM ISBN 978-1-4503-5657-2/18/07…$15.00
https://doi.org/10.1145/3209978.3210089

dense and compact global image representation. To compute an R-MAC feature, an input image is fed to a fully convolutional network pre-trained on ImageNet. The output of the last convolutional layer is max-pooled over different spatial regions at different position and scales, obtaining a feature vector for each region. These vectors are then $l2$-normalized, PCA-whitened, $l2$-normalized again, and finally aggregated by summing them together and $l2$-normalizing the final result. The obtained representation is an effective aggregation of local convolutional features at multiple position and scales that it can be compared with the cosine similarity function.

In our work, we used the ResNet-101 trained model provided by Gordo et al. [9] as an R-MAC feature extractor, which has been shown to achieve the best performance on standard benchmarks. We extracted the R-MAC features using fixed regions at two different scales as proposed in [16] instead of using the region proposal network. Defined S as the size in pixel of the minimum side of an image, we computed the multi-resolution descriptor aggregating the ones extracted at $S = 550$, 800 and $1,050$, resulting in a dense 2.048-dimensional real-valued image descriptor.

3 SURROGATE TEXT REPRESENTATION FOR CNN FEATURES

The basic idea of permutation-based indexing techniques is to represent feature objects as permutations built using a set of reference object identifiers as permutants.

Given a domain $\mathcal{D}$, a *distance function* $d : \mathcal{D} \times \mathcal{D} \rightarrow \mathbb{R}$, and a fixed set of reference objects $P = \{p_1 \dots p_n\} \subset \mathcal{D}$ that we call *pivots*, we define a permutation-based representation Π_o (briefly *permutation*) of an object $o \in \mathcal{D}$ as the sequence of pivots identifiers sorted in ascending order by their distance from o [3].

Formally, the permutation-based representation $\Pi_o = (\Pi_o(1), \dots, \Pi_o(n))$ lists the pivot identifiers in an order such that $\forall j \in \{1, \dots, n-1\},\ d(o, p_{\Pi_o(j)}) \leq d(o, p_{\Pi_o(j+1)})$, where $p_{\Pi_o(j)}$ indicates the pivot at position j in the permutation associated with object o. If we denote as $\Pi_o^{-1}(i)$ the position of a pivot p_i, in the permutation of an object $o \in \mathcal{D}$, so that $\Pi_o(\Pi_o^{-1}(i)) = i$, we obtain the equivalent *inverted* representation of permutations $\Pi_o^{-1} = (\Pi_o^{-1}(1), \dots, \Pi_o^{-1}(n))$. In Π_o the value in each position of the sequence is the identifier of the pivot in that position. In the inverted representation Π_o^{-1}, each position corresponds to a pivot and the value in each position corresponds to the rank of the corresponding pivot. The inverted representation of permutations Π_o^{-1}, is a vector that we refer to as *vector of permutations*, and which allows us to easily define most of the distance functions between permutations. Permutations are generally compared using Spearman rho, Kendall Tau, or Spearman Footrule distances.

So far, we have presented the general approach of permutation-based indexing. However, when objects to be indexed are *real-valued vectors* as in the case of deep features, we can use the approach presented in [2]. It allows us to generate a sequence of identifiers not associated with pivots. The basic idea is as follows. Permutants are the indexes of elements of the deep feature vectors. Given a deep feature vector, the corresponding permutation is obtained by sorting the indexes of the elements of the vector, in descending order with respect to the values of the corresponding elements. Suppose for instance the feature vector is $fv = [0.1, 0.3, 0.4, 0, 0.2]$ (in reality, the number of dimensions is 2,048 or more). The permutation-based representation of fv is $\Pi_{fv} = (3, 2, 5, 1, 4)$, that is permutant (index) 3 is in position 1, permutant 2 is in position 2, etc. The vector of permutations introduced above is therefore $\Pi_{fv}^{-1} = (4, 2, 1, 5, 3)$, that is permutant (index) 1 is in position 4, permutant 2 is in position 2, etc.

The intuition behind this is that features in the high levels of the neural network carry out some sort of high-level visual information. We can imagine that individual dimensions of the deep feature vectors represent some sort of visual concept and that the value of each dimension specifies the importance of that visual concept in the image. Similar deep feature vectors sort the visual concepts (the dimensions) in the same way, according to the activation values.

Without entering the technical details of this approach, let us just stress the fact that although the vector of permutations is of the same dimension of CNN vectors, the advantage is in that they be easily encoded into an inverted index. Moreover, following the intuition that the most relevant information of the permutation is in the very first, we can truncate the vector of permutations to the top-K elements (i.e., truncated permutations at K). The elements of the vectors beyond K can be ignored, allowing us to modulate the size of vectors and reduce the size of the index.

A key aspect of the R-MAC vector is the presence of a comparable number of positive and negative elements (activations) and, more important, both negative and positive elements contribute to informativeness. However, negative activations are practically neglected, when using conventional deep permutations, since we sort the indexes of the elements of the vector, in descending order with respect to the values of the corresponding elements. In order to prevent this imbalance towards positive activations at the expense of negative ones, we use the Concatenated Rectified Linear Unit (CReLU) transformation [15]. It simply makes an identical copy of vector elements, negate it, concatenate both original vector and its negation, and then apply ReLU altogether. More precisely, we denote ReLU as $[\cdot]+ = max(\cdot, 0)$, and define CReLU of the vector fv as $fv^+ = ([fv]+, [-fv]+)$. For instance, if $fv = [0.1, -0.3, -0.4, 0, 0.2]$, the CReLU applied on fv is $fv^+ = [0.1, 0, 0, 0, 0.2, 0, 0.3, 0.4, 0, 0]$. Finally, the vector permutations is then $\Pi_{fv+}^{-1} = (4, 5, 6, 7, 3, 8, 2, 1, 9, 10)$ (ties take random positions). Notice that in general, this operation is lossy since the dot product between vectors is not preserved, i.e., $fv_1 \cdot fv_2 \leq fv_1^+ \cdot fv_2^+$. However, this transformation allows us to apply the deep permutation approach without completely neglecting the negative activations of the R-MAC features. After applying the CReLU to the vectors fv, the rest of the procedure is the same as the deep permutation on vectors $fv+$, and everything else proceeds as for the conventional deep permutation approach.

Finally, in order to index the permutation vectors with a text-retrieval engine as Elasticsearch, we use the surrogate text representation introduced in [7], which simply consists in assigning a distinct codeword to each item of the permutation vector Π^{-1} and repeating the codewords a number of times equal to the complement of the rank of each item within the permutation. However, since our permutation vectors are of 4,096 dimensions ($2 \times 2,048$ due to CReLU), and since we want to promote a sparse representation, we also truncate permutations at their top-K elements. In

our running example, using $K = 4$ we have that the new truncated vector is $\Pi^{-1}_{fv+} = (4, 5, 5, 5, 3, 5, 2, 1, 5, 5)$. This means that all the positions greater than four are clipped to position five. To represent this vector using a surrogate text, let first τ_i be the codeword corresponding to the i-th component of the permutation vector. We then consider only the first four positions and ignore the elements of the vector greater then four, i.e., we generate the following surrogate text: "$\tau_1\ \tau_5\ \tau_5\ \tau_7\ \tau_7\ \tau_7\ \tau_8\ \tau_8\ \tau_8\ \tau_8$". We eventually repeat the codeword corresponding to the i-th component a number of times proportional to its, say, importance in the vector (i.e., if i-th component is at position 1, we repeat τ_i K times, if j-th component is at position 2, we repeat τ_j $K - 1$ times, and so on). The idea is based on the fact that the scalar product adopted by the search engine at query time will return the same ranking obtained by the Spearman rho distance applied to the original permutation vectors (see [7] for more clarifications).

4 EXPERIMENTAL EVALUATION

In this section, we evaluate the performance of the proposed solution in a multimedia information retrieval task. In particular, we extracted the R-MAC features from *INRIA Holidays* [10] dataset, which is a standard benchmark for image retrieval consisting in a collection of 1,491 images representing a large variety of scene type (natural, man-made, water, etc). The authors of INRIA Holidays selected 500 queries and manually identified a list of qualified results for each of them. In the literature, this benchmark is extended with the distractor dataset MIRFlickr including 1M images called MIRFlickr1M (http://press.liacs.nl/mirflickr/).

We compared the performance of our approach with state-of-the-art inverted-file-based approximate nearest neighbor algorithms based on product quantization (PQ) [11]. In PQ, the original vector is divided into M sub-vectors which are then independently quantized. A codebook is learned via k-means for each of the M sub-division, and each sub-vector is compressed by storing the nearest centroid index. We used the FAISS library [12] as an implementation of PQ-compressed inverted-file indexes, denoted IVFPQ. In IVFPQ, the feature space is partitioned into N Voronoi cells, each associated with a particular posting list of the inverted file. Each posting list contains the PQ-compressed difference between samples belonging to that cell and the cell centroid. When building the index, both the cell centroids and the PQ-compression codebooks have to be pretrained on a subset of the data. When querying the index, we probe the posting lists of the P Voronoi cells nearest to the query, and we reconstruct the samples using the codebooks. The number of Voronoi cells N controls the number and length of the postings lists, while the number of PQ sub-quantizer M controls the amount of memory occupied by a sample, also known as code size C. In the implementation of FAISS, we directly choose the code size C, and M is set accordingly by the software. Since FAISS is an in-memory index, C is usually set to the maximum code size to fit the whole dataset we want to index in main memory. While N and C are fixed parameter chosen at indexing time, the number of nearest neighbor Voronoi cells to probe P can be adjusted at query time and can be tuned to control the effectiveness-efficiency trade-off.

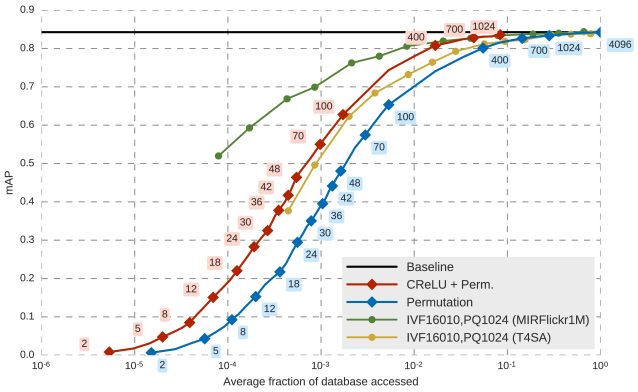

Figure 1: mAP vs AFDA. Lines with diamond markers show the trade-off of Deep Permutation approaches for increasing values of K (reported near each point). Lines with circle markers correspond to PQ-compressed inverted-file-based indexes (FAISS) for increasing number of Voronoi cell accessed. The horizontal line represents the baseline *mAP*, i.e. the one computed using the original R-MAC vectors.

For a fair comparison, we used a configuration for FAISS which gives the best trade-off between effectiveness and efficiency, choosing a relatively big code size $C = 1024$ and number of Voronoi cells $N = 16k$. However, this configuration has long index training times and a large memory footprint (for 1M images, FAISS ($C = 1024$) requires about 1GB in main memory, against about 0.7GB of our solution in secondary memory in the larger configuration for $K = 400$), and this could limit its scalability, especially in systems with limited main memory.

Evaluation of the Effectiveness. We generated different sets of permutations from the original features using different values of K (i.e., we consider top-K truncated permutations), and for each configuration, we measured the *mAP* (mean Average Precision) obtained and the query execution times for each configuration. In Figure 1, we report the mAP (on the y-axis) in function of the *Average Fraction of Database elements Accessed* (AFDA in brief, on the x-axis). The AFDA for surrogate text representation techniques can easily be derived as follows. Let D the dimension of the vector (2,048 for fv and 4,096 for $fv+$ in the case of R-MAC), and d_i the density of i-th dimension (computed as the fraction of samples in the database having a non-zero i-th dimension). Given a query q, we access the i-th posting list only if the i-th dimension of q is non-zero, which is true with probability d_i. The i-th posting list contains the elements of the database having non-zero i-th dimension, that is a d_i fraction of the database. Hence, we can compute $AFDA = \sum_i^D d_i^2$, because the i-th posting list contains a d_i fraction of the database, and we access it with d_i probability. Note that, in a surrogate text representation, we associate a distinct codeword to each dimension of the vector, and the smaller the AFDA is, the more balanced are the posting lists of the inverted index. The greater is K, the lower is the sparsity, and hence the greater is the *mAP* and the average fraction of database accessed. Since we are dealing with an approximate approach, the baseline mAP computed with the

K	42	48	70	100	400
avg. time (msec)	51	60	93	155	1,381

Table 1: Average query times using Elasticsearch on MIRFlickr1M.

# nodes (imgs.)	1 (1M)	2 (2M)	3 (3M)	4 (4M)
avg. time (msec)	274	193	189	227

Table 2: Scalability of the queries for $K = 100$ using Elasticsearch on a cluster of up to four nodes.

original features can be considered as an upper-bound. We can see that we reach satisfactory levels of effectiveness for an AFDA of 10^{-2}. Moreover, we have shown the remarkable advantage of the CReLU preprocessing on R-MAC vectors in comparison with the deep permutation method applied directly on the original vectors.

Figure 1 also reports the performance of FAISS indexes when varying the number of Voronoi cells probed. The two curves related to FAISS correspond to two different datasets used for learning the codebooks: MIRFlickr1M and T4SA [17]. The former is the one on which the mAP evaluation is performed, while the latter is a large collection of images collected from the live stream of random 5% of global tweets using the Twitter Streaming API. The reason for this choice is to show how the performance of FAISS is sensitive to the specific dataset distribution on which it has been trained. Indeed, we see the impact of this aspect is really strong, and it could, in real applications, influence the scalability of the system or require continuous codebook adjustments, forcing to re-indexing the data periodically. Our solution has an intermediate performance but does not require any training procedure and therefore any readjustments.

Evaluation of the Efficiency. In order to see the impact of truncation of permutations on efficiency, we report in Table 1 the average query time when varying the parameter K. Clearly, by increasing K the query time increases. The times reported in Table 1 are measured using only the NoSQL database (Apache Lucene) upon which Elasticsearch is built around. When we take into consideration the execution time of the overall Elasticsearch queries, a little performance degradation due to the communication and computation overheads of Elasticsearch has also to be taken into account. However, one of the advantages of Elasticsearch consists in its ability of horizontal scaling and balancing the loading between the nodes in a cluster.

Table 2 shows the scalability of the average query time using Elasticsearch and permutations with truncation at $K = 100$ as we increase the number of nodes to accommodate the increment in size of the search database. In particular, we increased the size of the database from one to four million items and incremented the number of nodes from one to four, accordingly. As can be seen, the average query response time remains fairly stable; the oscillations are mainly due to nodes having different types of commodity hardware.

5 CONCLUSION

In this paper, we present an approach for indexing CNN features as permutations to build a CBIR system. We rely upon the full-text retrieval engine Elasticsearch, which works in secondary memory and provides horizontal scalability and reliability.

In a nutshell, the idea is to exploit the same activation values of the neural network as a means to associate CNN feature vectors with permutations. Specifically, we have explored the impact of introducing a (CReLU) preprocessing phase on R-MAC dense descriptors, which allowed us to regain the informative contribution of the negative elements of the vectors. We also observed how our approach exhibits interesting performance in terms of efficiency and effectiveness compared to state-of-the-art approaches, which operate in main memory and hardly scale to large-scale applications.

REFERENCES

[1] Giuseppe Amato, Fabio Carrara, Fabrizio Falchi, and Claudio Gennaro. 2017. Efficient Indexing of Regional Maximum Activations of Convolutions using Full-Text Search Engines. In *ACM on International Conference on Multimedia Retrieval*. ACM, 420–423.

[2] Giuseppe Amato, Fabrizio Falchi, Claudio Gennaro, and Lucia Vadicamo. 2016. Deep Permutations: Deep Convolutional Neural Networks and Permutation-Based Indexing. In *International Conference on Similarity Search and Applications*. Springer, 93–106.

[3] Giuseppe Amato, Claudio Gennaro, and Pasquale Savino. 2014. MI-File: using inverted files for scalable approximate similarity search. *Multimedia Tools and Applications* 71, 3 (2014), 1333–1362.

[4] Artem Babenko, Anton Slesarev, Alexandr Chigorin, and Victor Lempitsky. 2014. Neural codes for image retrieval. In *Computer Vision–ECCV 2014*. Springer, 584–599.

[5] Jeff Donahue, Yangqing Jia, Oriol Vinyals, Judy Hoffman, Ning Zhang, Eric Tzeng, and Trevor Darrell. 2013. Decaf: A deep convolutional activation feature for generic visual recognition. *arXiv preprint arXiv:1310.1531* (2013).

[6] Matthijs Douze, Hervé Jégou, and Florent Perronnin. 2016. *Polysemous Codes*. Springer International Publishing, Cham, 785–801.

[7] Claudio Gennaro, Giuseppe Amato, Paolo Bolettieri, and Pasquale Savino. 2010. An approach to content-based image retrieval based on the Lucene search engine library. In *International Conference on Theory and Practice of Digital Libraries*. 55–66.

[8] Ross Girshick, Jeff Donahue, Trevor Darrell, and Jitendra Malik. 2014. Rich feature hierarchies for accurate object detection and semantic segmentation. In *IEEE conference on computer vision and pattern recognition*. 580–587.

[9] Albert Gordo, Jon Almazán, Jérôme Revaud, and Diane Larlus. 2017. End-to-End Learning of Deep Visual Representations for Image Retrieval. *International Journal of Computer Vision* 124, 2 (2017), 237–254.

[10] Hervé Jégou, Matthijs Douze, and Cordelia Schmid. 2008. Hamming Embedding and Weak Geometric Consistency for Large Scale Image Search. In *Computer Vision – ECCV 2008*. LNCS, Vol. 5302. Springer, 304–317.

[11] Herve Jegou, Matthijs Douze, and Cordelia Schmid. 2011. Product quantization for nearest neighbor search. *IEEE transactions on pattern analysis and machine intelligence* 33, 1 (2011), 117–128.

[12] Jeff Johnson, Matthijs Douze, and Hervé Jégou. 2017. Billion-scale similarity search with GPUs. *arXiv preprint arXiv:1702.08734* (2017).

[13] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. 2012. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*. 1097–1105.

[14] Ali S Razavian, Hossein Azizpour, Josephine Sullivan, and Stefan Carlsson. 2014. CNN features off-the-shelf: an astounding baseline for recognition. In *Computer Vision and Pattern Recognition Workshops (CVPRW), 2014 IEEE Conference on*. IEEE, 512–519.

[15] Wenling Shang, Kihyuk Sohn, Diogo Almeida, and Honglak Lee. 2016. Understanding and improving convolutional neural networks via concatenated rectified linear units. In *International Conference on Machine Learning*. 2217–2225.

[16] Giorgos Tolias, Ronan Sicre, and Hervé Jégou. 2015. Particular object retrieval with integral max-pooling of CNN activations. *arXiv preprint arXiv:1511.05879* (2015).

[17] Lucia Vadicamo, Fabio Carrara, Andrea Cimino, Stefano Cresci, Felice Dell'Orletta, Fabrizio Falchi, and Maurizio Tesconi. 2017. Cross-Media Learning for Image Sentiment Analysis in the Wild. In *The IEEE International Conference on Computer Vision (ICCV)*.