Deep Permutations: Deep Convolutional Neural Networks and Permutation-Based Indexing

Giuseppe Amato, Fabrizio Falchi, Claudio Gennaro, and Lucia Vadicamo

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

Abstract. The activation of the Deep Convolutional Neural Networks hidden layers can be successfully used as features, often referred as Deep Features, in generic visual similarity search tasks.
Recently scientists have shown that permutation-based methods offer very good performance in indexing and supporting approximate similarity search on large database of objects. Permutation-based approaches represent metric objects as sequences (permutations) of reference objects, chosen from a predefined set of data. However, associating objects with permutations might have a high cost due to the distance calculation between the data objects and the reference objects.
In this work, we propose a new approach to generate permutations at a very low computational cost, when objects to be indexed are Deep Features. We show that the permutations generated using the proposed method are more effective than those obtained using pivot selection criteria specifically developed for permutation-based methods.

Keywords: Similarity Search, Permutation-Based Indexing, Deep Convolutional Neural Network

1 Introduction

The activation of the Deep Convolutional Neural Networks (DCNNs) hidden layers has been used in the context of transfer learning and content-based image retrieval [10, 23]. 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" [19]. These representations can be successfully used as features in generic recognition or visual similarity search tasks. The first layers 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.

However, DCNN features are typically of high dimensionality. For instance, in the well-known AlexNet architecture [18] the output of the sixth layer (fc6) has 4,096 dimensions, while the fifth layer (pool5) has 9,216 dimensions. This

represents a major obstacle to the use of DCNN features on large scale, due to the well-known dimensionality curse [13].

An effective approach to tackle the dimensionality curse problem is the application of approximate access methods. Permutation-based approaches [4, 9, 11, 22] are promising access methods for approximate similarity search. They represent metric objects as sequences (permutations) of reference objects, chosen from a predefined set of objects. Similarity queries are executed by searching for data objects whose permutation representations are similar to the query permutation representation. Each permutation is generated by sorting the entire set of reference objects according to their distances from the object to be represented.

The total number of reference objects, to be used for building permutations, depends on the size of the dataset to be indexed, and can amount to tens of thousands [4]. In these cases, both indexing time and searching time is affected by the cost of generating permutations for objects being inserted, or for the queries.

In this paper, we propose an approach to generate permutations for Deep Features at a very low computational cost since it does not require the distance calculation between the reference objects and the objects to be represented. Moreover, we show that the permutations generated using the proposed method are more effective than those obtained using pivot selection criteria specifically developed for permutation-based methods.

The rest of the paper is organized as follows. In Section 2, we briefly describe related work. Section 3 provides background for the reader. In Section 4, we introduce our approach to generate permutations for Deep Features. Section 5 presents some experimental results using real-life datasets. Section 6 concludes the paper.

2 Related Work

Pivot selection strategies for permutation-based methods were discussed in [2]. In the paper the Farthest-First Traversal (FFT) technique was identified as the one providing a set of reference objects such that the sorting performed with similarity computed among the permutations was the most correlated to sorting performed using the original distance. We will see that the techniques proposed here for Deep Features outperform also the FFT technique.

The permutation-based approach was used in PPP-Codes index [21] to index a collection of 20 million images processed by a deep convolutional neural network. However, no special techniques was used to generate permutations for Deep Features.

Some recent works try to treat the features in a convolutional layer as local features [5, 25]. This way, a single forward pass of the entire image through the DCNN is enough to obtain the activation of its local patches, which are then encoded using *Vector of Locally Aggregated Descriptors* (VLAD). A similar approach uses *Bag of Words* (BoW) encoding instead of VLAD to take advantage of sparse representations for fast retrieval in large-scale databases. However,

although authors claim that their approach is very scalable in terms of search time, they did not report any efficiency measurements and experiments have been carried out on datasets of limited size.

Liu et al. [20] proposed a framework that adapts Bag-of-Word model and inverted table to DCNN feature indexing, which is similar to the one we propose. However, for large-scale datasets, Liu et al. have to build a large-scale visual dictionary that employs the 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.

An approach, called *LuQ* and introduced in [1], exploits the quantization of the vector components of the DCNN features that allows one to use a text retrieval engine to perform image similarity search. In LuQ, each real-valued vector component x_i of the deep feature 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 documents representing the feature vectors.

3 Background

In the following we introduce the needed notions of permutation-based similarity search approach and Deep Features.

3.1 Permutation-Based Indexing

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* or *reference objects*, 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 [4, 9, 11, 22].

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 Π_o^{-1}:

$$\Pi_o^{-1} = (\Pi_o^{-1}(1), \dots, \Pi_o^{-1}(n)).$$

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} 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. As an example given two permutations Π_x and Π_y, Spearman rho distance is defined as:

$$S_\rho(\Pi_x, \Pi_y) = \sqrt{\sum_{1 \leq i \leq n} (\Pi_x^{-1}(i) - \Pi_y^{-1}(i))^2}$$

Following the intuition that the most relevant information of the permutation Π_o is in the very first, i.e. nearest, pivots [4], the Spearman rho distance with location parameter $S_{\rho,l}$ is a generalization intended to compare top-l lists (i.e., truncated permutations). It was defined in [12] as:

$$S_{\rho,l}(\Pi_x, \Pi_y) = \sqrt{\sum_{1 \leq i \leq n} (\tilde{\Pi}_{x,l}^{-1}(i) - \tilde{\Pi}_{y,l}^{-1}(i))^2}$$

$S_{\rho,l}$ differs from S_ρ for the use of an inverted top-l permutation $\tilde{\Pi}_{o,l}^{-1}$, which assumes that pivots further than $p_{\Pi_o(l)}$ from o are assigned to position $l+1$. Formally, $\tilde{\Pi}_{o,l}^{-1}(i) = \Pi_o^{-1}(i)$ if $\Pi_o^{-1}(i) \leq l$ and $\tilde{\Pi}_{o,l}^{-1}(i) = l+1$ otherwise.

It is worth noting that only the first l elements of the permutation Π_o are used, in order to compare any two objects with the $S_{\rho,l}$.

3.2 Deep Features

Recently, a new class of image descriptor, built upon Deep Convolutional Neural Networks, have been used as effective alternative to descriptors built using local features such as SIFT, SURF, ORB, BRIEF, etc. DCNNs have attracted enormous interest within the Computer Vision community because of the state-of-the-art results [18] achieved in challenging image classification challenges such as 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 [10, 7] and object detection [14], 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) can be used to transfer learning across tasks [10, 23]. The activation of neurons of a specific layers, in particular the last ones, can be used as features for describing the visual content.

In order to extract Deep Features, we used a trained model publicly available for the popular Caffe framework [17]. Many deep neural network models, in particular trained models, are available for this framework[1] . Among them, we chose the HybridNet for several reasons: first, its architecture is the very same of the famous AlexNet [18]; 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 [26]; last, but not least, experiments conducted on various datasets demonstrate the good transferability of the learning [26, 8, 6].

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

We decided to use the activation of the first fully connected layer, the fc6 layer, given the results reported on [10, 7, 8].

The activations at the fc6 layer is a vector of 4,096 of floats. Generally, the rectified linear unit (ReLU) is used to bring to zero all negative activation values. In this way, feature vectors contain only values greater or equal to zero. Feature vectors are sparse, so that in average about 75% of elements are zero.

4 Permutation Representation for Deep Features

As introduced in Section 3.1 the basic idea of permutation-based indexing techniques is to represent data objects with permutations built using a set of reference object identifiers as permutants. Given an object o, its permutation-based representation Π_o is the list of reference object identifiers, sorted in ascending order with respect to the distance between o and the various reference objects.

Using the permutation-based representation, similarity between two objects is estimated computing the similarity between the two corresponding permutations, rather than using the original distance function. The rationale behind this is that, when permutations are built using this strategy, objects that are very close one to the other, have similar permutation representations as well. In other words, if two objects are very close one to the other, they will sort the set of reference objects in a very similar way.

Notice however that, the relevant aspect, when building permutations, is the capability of generating sequences of identifiers (permutations) in such a way that similar objects have similar permutations as well. Sorting a set of reference objects, according to their distance with the object to be represented is just one, yet effective, approach.

Here, we propose an approach to generate sequence of identifiers, not necessarily associated with reference objects, when objects to be indexed are Deep Features. 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 $fv = [0.1, 0.3, 0.4, 0, 0.2]$[2]. 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, permutant 5 is in position 3, etc. The inverted representation, introduced in Section 3.1 is $\Pi^{-1}_{fv} = (4, 2, 1, 5, 3)$, that is permutant (index) 1 is in position 4, permutant 2 is in position 2, permutant 3 is in position 1, 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.

[2] In reality, the number of dimensions is 4,096 or more.

More formally, let $fv = [v_1, \ldots, v_n]$ be a deep feature vector (where $n = 4,096$, in our case). The corresponding permutation is $\Pi_{fv} = (\Pi_{fv}(1), \ldots, \Pi_{fv}(n))$ such that $\forall i \in \{1, \ldots, n-1\}, fv[\Pi_{fv}(i)] \geq fv[\Pi_{fv}(i+1)]$. Using the inverted representation, introduced in Section 3.1, we have that $\Pi^{-1}_{fv} = (\Pi^{-1}_{fv}(1), \ldots, \Pi^{-1}_{fv}(n))$ such that if $\Pi^{-1}_{fv}(i) \leq \Pi^{-1}_{fv}(j)$ then $fv[i] \geq fv[j]$, that is if index i of the vector appears before index j in the permutation, then the value of element i of the vector is greater than that of element j.

Let us discuss more in details the process of creating permutations with the activations of the deep neural network. Note that when two elements of a deep feature vectors have the same value, their position in the permutation cannot be uniquely assigned. This is very rare, with elements having a value different than zero, since we are using real values. However, as we said in Section 3.2, on average, 75% of the dimension have value equal to zero. This means that order of these elements is not unique.

In order to face this problem, we define and compare two different strategies:

- The first strategy, which we call *zeros-to-l*, assigns all elements having value equal to zero to position $l + 1$, where l is the location parameters.
- The second strategy, which we call *no-ReLU* does not use the ReLU (See section 3.2) so that negative values are not flattened to 0 and vector components with the same activation values occur very rarely.

If we restrict to the case of Sperman rho distance and considering deep feature vectors L2-normalized to the unit length, it easy to see that our strategy of generating permutations is equivalent to the following permutation generation strategy.

- Create a set of 4,096 reference vectors (pivots) such that the i-th reference object has 1 in dimension i and 0 in all other elements of the vectors.
- Given an object o sort all reference objects in ascending order to their distance from o, as described in Section 3.1.

A question that might arise after this description is what is the benefit of this approach given that vector of permutations are of the same dimension of DCNN vectors. The advantage of the proposed approach is that permutation vectors can be easily encoded into an inverted index, which exhibits high efficiency as shown in [3].

5 Experiments

The similarity search paradigm employs a similarity (or a distance) function to retrieve objects similar to a query. The similarity function and the object representations are chosen so that they reflect the user (or the application) requirements for the task being executed. However, generally, the similarity function does not capture precisely the semantic of the indexed objects, and some errors occur in the similarity search results.

In our case, we are testing an approximate similarity search algorithm. That is, an algorithm that returns a result that is approximate with respect to the exact similarity search result, which in turns tries to satisfy the user retrieval requirements. The assumption is that, although the approximate similarity search result is an approximation of the exact similarity search results, the user does not notice the possible degradation of accuracy, given that also the exact similarity search algorithm is already an approximation of his/her intuition of similarity.

In this respect, we performed two type of experiments. We first evaluated the performance of the proposed technique in a pure similarity search task, where we use an exact similarity search ground-truth to assess the quality of the approximate similarity search, obtained with the permutation-based approach. Then, we evaluated the performance in a multimedia information retrieval task. Here, the ground-truth was manually generated associating each query with a set of results pertinent to the query. In this way, we were able to evaluate both the approximation introduced with respect to the exact similarity search algorithm, and the impact of this approximation with respect to the user perception of the retrieval task.

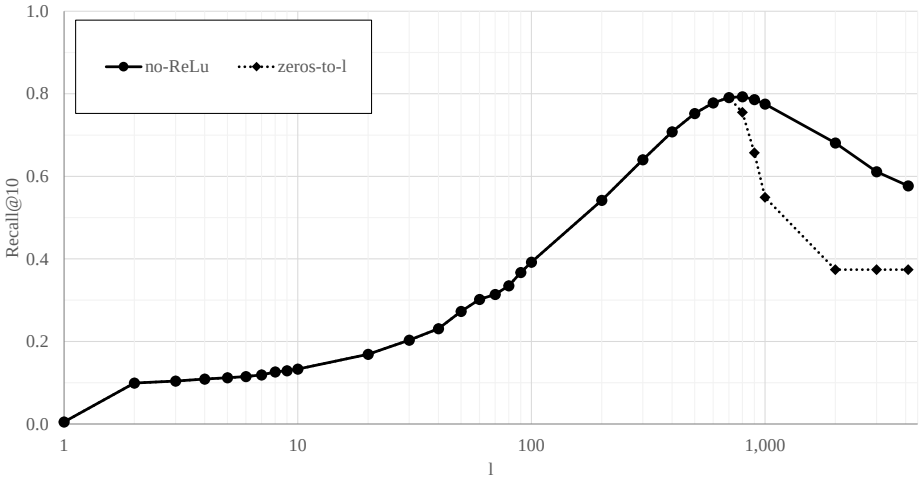

Fig. 1. Comparison between the *no-ReLU* and the *zeros-to-l* techniques, varying the location parameter l (length of the truncated permutations).

5.1 Experimental Settings

For assessing the proposed technique in a pure similarity search task, we used the Deep Features extracted as discussed in Section 3.2 from the *Yahoo Flickr Creative Commons 100 Million dataset (YFCC100M)* [24].

The assessment of the proposed algorithm in a multimedia information retrieval task was performed using the Deep Features extracted from the *INRIA Holidays dataset* [16].

The YFCC100M dataset [24] contains almost 100M of the images, all uploaded to Flickr between 2004 and 2014 and published under a CC commercial or noncommercial license. The ground-truth was built selecting 1,000 different queries and executing an exact similarity search on these queries using the euclidean distance to compare Deep Features.

INRIA Holidays [16] is a collection of 1,491 images, which mainly contains personal holidays photos. The images are of high resolution and represent a large variety of scene type (natural, man-made, water, fire effects,etc). The authors selected 500 queries and manually identified a list of qualified results for each of them. As in [15], we merged the Holidays dataset with the distraction dataset MIRFlickr including 1M images [3].

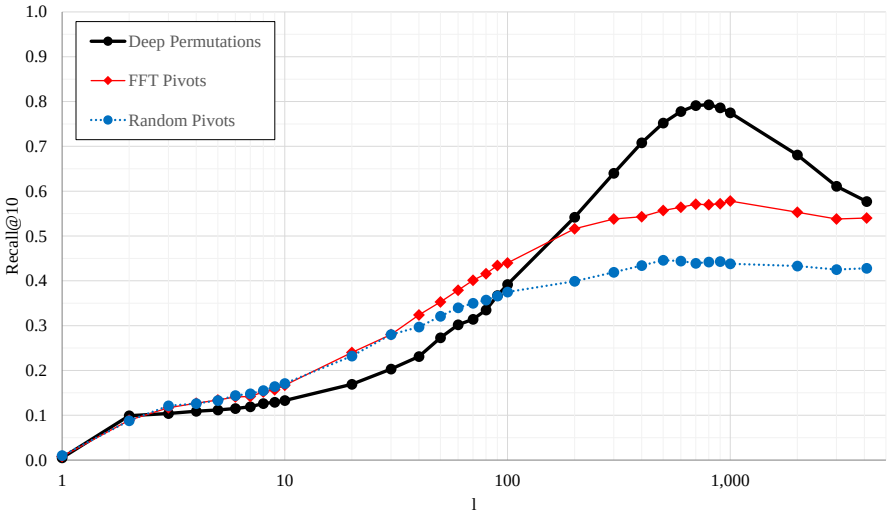

Fig. 2. Comparisons of the proposed Deep Permutation approach, with standard permutation-based methods using random selection of pivots, and Farthest-First Traversal (FFT) pivot selection strategy.

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

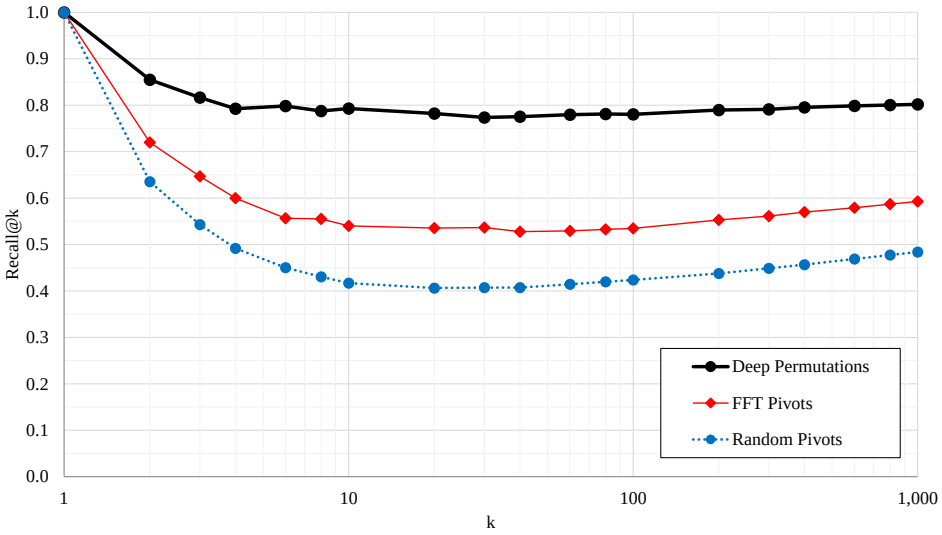

Fig. 3. *Recall@k* varying k for our approach with l=800 and 4,096 random and FFT pivots.

5.2 Evaluation in a Similarity Search Task

In order to assess quality of search results of our approach, we use the measure called *recall@k*, which determines the ratio of correct results for a given query in the top-k results returned. Let $ER_Q(k)$ and $AR_Q(k)$ be the top-k sets of results returned by exact and approximate similarity search, respectively. The recall@k 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:

$$recall@k = \frac{|AR_Q(k) \cap ER_Q(k)|}{|ER_Q(k)|}.$$

Where $|\cdot|$ denote the size of a set.

We first discuss the comparison of the two approaches that we defined for handling elements of the vectors having zero as value: *no-ReLU* and the *zeros-to-l*. These tests were executed on a subset of the YFCC100M dataset of size 1M and the results are shown in Figure 1. In experiments, we vary the location parameter l, that is the length of the truncated top-l permutation, and we compute the recall@10.

The figure shows that the plots corresponding to the two strategies are overlapped until $l = 700$. Them, the *zeros-to-l* degrades with respect to the other. At $l = 900$ also the *no-ReLU* starts degrading, remaining always higher than the

other. This behavior is due to the presence of elements with value equal to zero. As we said in Section 4, it is not possible to distinguish and to sort the elements having value equal to zero and, on average, about the 75% of elements of fc6 vectors are zeros. This means that when l approaches to 1,000, there are no more elements with non-zero values, which up to now were correctly sorted, and we encounter elements having value equal to zero. These elements are all assigned to position $l+1$ in the *zeros-to-l* approach, and are replaced by the negative activation values, seen before applying the ReLU, in the *no-ReLU* approach. The graphs show that using negative value for sorting these elements helps, until a certain degree. For larger values of l the quality degrades. It is worth mentioning that when the neural network was trained, the ReLU was used. Therefore, negative values were never seen at the output and they were always flattened to zero. Therefore, the negative values were not subject of fine tuning during the learning phase, and were always all treated as zeros. This is the reason why we see a degradation also using negative values. It would be interesting comparing with a network trained without using ReLU. However, this was out of the scope of this paper, where we wanted to use a standard DCNN, and we leave it to future investigation.

Figure 2 compares the proposed approach, with the *no-ReLU* strategy using a standard permutation-based approach, where pivots where both selected randomly and using Farthest-First Traversal (FFT), which in [2] was shown to be the best pivot selection method for permutation-based searching. Random selection and FFT offer better performance for values of l up to 200. Then the Deep Permutation approach is much better, reaching a recall of 80%. The FFT approach is always lower than 60% and the random approach reaches just 45% recall.

Figure 3 compares our approach against random selection and FFT, computing the $recall@k$ for k ranging from 1 to 1,000. Also in this case, we can see that the new proposed approach outperforms the others and remains practically stable for all ks.

Tests discussed above were executed on a subset of size 1M of the entire YFCC100M dataset. Figure 4 shows the performance of the Deep Permutations approach increasing the size of the indexed dataset up to 100M. Here, we compute recall for k equal to 10, 100, and 1,000. Also in this case we do not see significant differences for different values of k, and the recall remains also stable around 80% for the various tested sizes of the dataset.

5.3 Evaluation in a Multimedia Information Retrieval Task

In this set of experiments, we only test the *no-ReLU* approach. Figure 5 shows the graph of the mean average precision (mAP) varying the location parameter l, that is the length of the truncated permutation. We can see that the mAP improves rapidly until l is 100, then remains stable slightly below 0.8. The maximum is reached when $l = 800$, where the mAP is 0.77.

These values of mAP are rather surprising and competing with state of the art methods tested on the INRIA Holidays dataset. To further investigate this

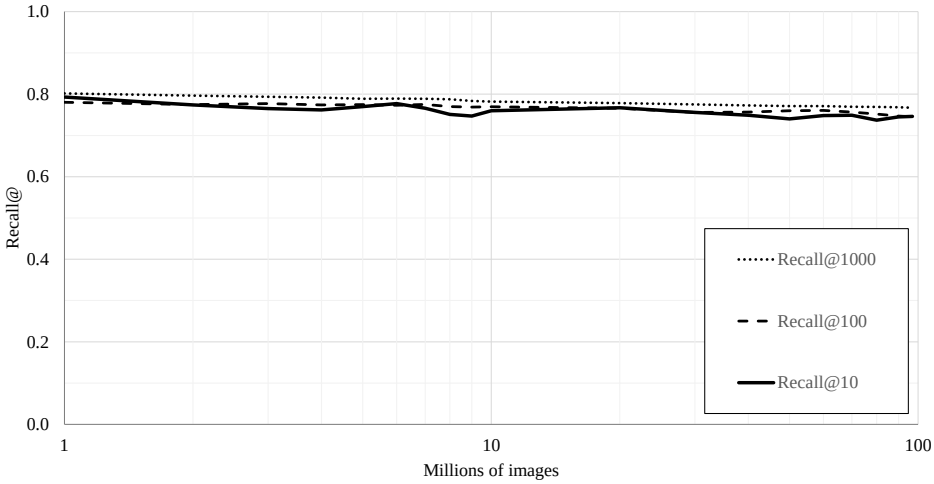

Fig. 4. *Recall@k* for various k varying dataset size (expressed in millions) obtained by the proposed approach for l=800.

we have compared the obtained results with the direct use of the Deep Features, using the Eucledian distance ($L2$) as distance, and with the LuQ method [1]. The comparison was performed on the INRIA Holidays dataset alone and together with the MIRFlickr dataset.

Results are reported in Table 1. The direct use of the Deep Features on the INRIA Holidays dataset, using the $L2$ distance, exhibits a mAP of 0.75 with $ReLU$, 0.76 without $ReLU$. On the INRIA Holidays dataset with the MIRFlickr distraction set, it exhibits a mAP of 0.69 with $ReLU$, 0.62 without $ReLU$.

Our approach on the INRIA Holidays dataset, exhibits a mAP of 0.75 with full permutations, 0.77 with $l = 800$. On the INRIA Holidays dataset with MIRFlickr distraction set, we obtain a mAP of 0.60 with with full permutations, 0.62 with $l = 800$. The results obtained using $l = 800$ are always greater or equal to the one obtained directly using the Deep Features, and equal to the results obtained by LuQ.

Looking at these results we can make an additional observation. Deep Features are generally compared using the $L2$ distance. However, results above suggest that possibly this is not the best distance function to be used. In fact, transforming Deep Features into permutations and comparing them using the Spearman rho distance has slightly better performance. Thus, investigations on better distance functions to be used with Deep Features is worth being considered.

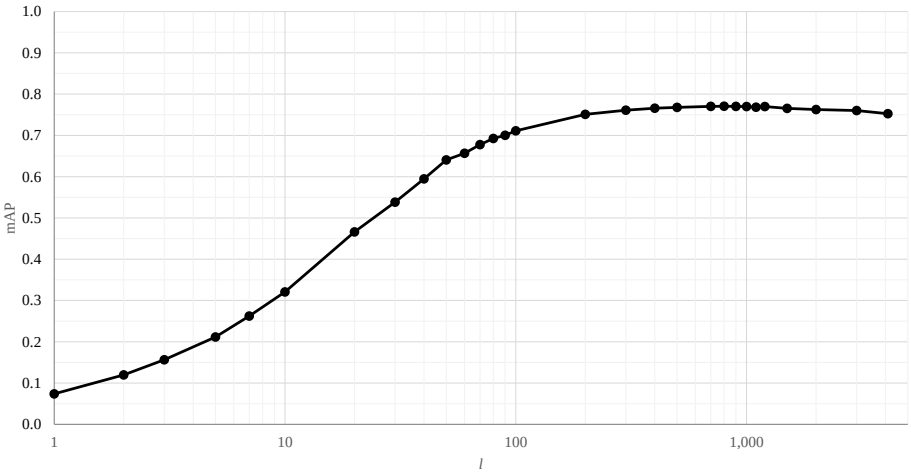

Fig. 5. mAP obtained on INRIA Holidays varying l.

Table 1.

	L2		Deep Permutations		LuQ[1]
	ReLu	no-ReLu	full	$l = 800$	
Holidays	0.75	0.76	0.75	0.77	0.77
Holidays+MIRFlickr	0.60	0.62	0.60	0.62	0.62

6 Conclusion

In this paper, we presented an approach for representing and fast indexing Deep Convolutional Neural Network Features as permutations. Compared to the classical approach based on permutation, this technique does not need computing distances between pivots and data objects but uses the same activation values of the neural network as a source for associating Deep Feature vectors with permutations.

The proposed technique when evaluated in a pure similarity search task offers a recall up to 80%, much higher than other permutation-based methods. We also evaluated this technique in a multimedia information retrieval context. Here, surprisingly, the proposed technique offers a mean average precision of 0.77, slightly higher than the direct use of the Deep Features with the $L2$ distance.

This suggests that probably the $L2$ is not the most effective distance function to be used with Deep Features, given that permutation representation together with the Spearman rho distance provide better performance.

Note also that, probably, the same approach can be applied to any feature represented as a vector, not just DCNN features, provided its dimensionality is high. We are going to investigate how this idea generalizes to other features and to other distance functions as future work.

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

References

1. 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
2. Amato, G., Esuli, A., Falchi, F.: Pivot selection strategies for permutation-based similarity search. In: Brisaboa, N., Pedreira, O., Zezula, P. (eds.) Similarity Search and Applications: 6th International Conference, SISAP 2013, A Coruña, Spain, October 2-4, 2013, Proceedings. pp. 91–102. Springer Berlin Heidelberg, Berlin, Heidelberg (2013), http://dx.doi.org/10.1007/978-3-642-41062-8_10
3. Amato, G., Gennaro, C., Savino, P.: Mi-file: using inverted files for scalable approximate similarity search. Multimedia Tools and Applications pp. 1–30 (2012)
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. Arandjelović, R., Gronat, P., Torii, A., Pajdla, T., Sivic, J.: Netvlad: Cnn architecture for weakly supervised place recognition. arXiv preprint arXiv:1511.07247 (2015)
6. 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)
7. Babenko, A., Slesarev, A., Chigorin, A., Lempitsky, V.: Neural codes for image retrieval. In: Computer Vision–ECCV 2014, pp. 584–599. Springer (2014)
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., Figueroa, K., Navarro, G.: Effective proximity retrieval by ordering permutations. Pattern Analysis and Machine Intelligence, IEEE Transactions on 30(9), 1647–1658 (2008)
10. 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)

11. Esuli, A.: Use of permutation prefixes for efficient and scalable approximate similarity search. Information Processing & Management 48(5), 889–902 (2012)
12. Fagin, R., Kumar, R., Sivakumar, D.: Comparing top k lists. In: Proceedings of the Fourteenth Annual ACM-SIAM Symposium on Discrete Algorithms. pp. 28–36. SODA '03, Society for Industrial and Applied Mathematics (2003)
13. Ge, Z., McCool, C., Sanderson, C., Corke, P.: Modelling local deep convolutional neural network features to improve fine-grained image classification. In: Image Processing (ICIP), 2015 IEEE International Conference on. pp. 4112–4116. IEEE (2015)
14. Girshick, R., Donahue, J., Darrell, T., Malik, J.: Rich feature hierarchies for accurate object detection and semantic segmentation. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 580–587 (2014)
15. Jégou, H., Douze, M., Schmid, C.: Packing bag-of-features. In: Computer Vision, 2009 IEEE 12th International Conference on. pp. 2357 –2364 (29 2009-oct 2 2009)
16. Jégou, H., Douze, M., Schmid, C.: Hamming embedding and weak geometric consistency for large scale image search. In: Forsyth, D., Torr, P., Zisserman, A. (eds.) Computer Vision – ECCV 2008, Lecture Notes in Computer Science, vol. 5302, pp. 304–317. Springer Berlin Heidelberg (2008), http://dx.doi.org/10.1007/978-3-540-88682-2_24
17. 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)
18. 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)
19. LeCun, Y., Bengio, Y., Hinton, G.: Deep learning. Nature 521(7553), 436–444 (2015)
20. Liu, R., Zhao, Y., Wei, S., Zhu, Z., Liao, L., Qiu, S.: Indexing of cnn features for large scale image search. arXiv preprint arXiv:1508.00217 (2015)
21. Novak, D., Batko, M., Zezula, P.: Large-scale image retrieval using neural net descriptors. In: Proceedings of the 38th International ACM SIGIR Conference on Research and Development in Information Retrieval. pp. 1039–1040. ACM (2015)
22. Novak, D., Kyselak, M., Zezula, P.: On locality-sensitive indexing in generic metric spaces. In: Proceedings of the Third International Conference on Similarity Search and Applications. pp. 59–66. SISAP '10, ACM (2010)
23. 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)
24. 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)
25. Yue-Hei Ng, J., Yang, F., Davis, L.S.: Exploiting local features from deep networks for image retrieval. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops. pp. 53–61 (2015)
26. 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)