Efficient Image Search with Neural Net Features

David Novak[1](✉), Jan Cech[2], and Pavel Zezula[1]

[1] Masaryk University, Brno, Czech Republic
{david.novak,zezula}@fi.muni.cz
[2] Czech Technical University, Prague, Czech Republic
cechj@cmp.felk.cvut.cz

Abstract. We present an efficiency evaluation of similarity search techniques applied on visual features from deep neural networks. Our test collection consists of 20 million 4096-dimensional descriptors (320 GB of data). We test approximate k-NN search using several techniques, specifically FLANN library (a popular in-memory implementation of k-d tree forest), M-Index (that uses recursive Voronoi partitioning of a metric space), and PPP-Codes, which work with memory codes of metric objects and use disk storage for candidate refinement. Our evaluation shows that as long as the data fit in main memory, the FLANN and the M-Index have practically the same ratio between precision and response time. The PPP-Codes identify candidate sets ten times smaller then the other techniques and the response times are around 500 ms for the whole 20M dataset stored on the disk. The visual search with this index is available as an online demo application. The collection of 20M descriptors is provided as a public dataset to academic community.

1 Introduction: Content-Based Image Retrieval

The content-based image retrieval (CBIR) is an area that naturally requires similarity techniques to match the image data. A successful CBIR system must stand on two pillars: *effective* image processing to achieve high quality of the retrieval, and *efficient* search to make the system work in real time and on a large scale. Specifically, the image processing typically leads to certain features (descriptors, stimuli) that capture the application-driven characteristics of the image data; the actual searching process is then realized in space of these features.

Currently, the state-of-the-art image recognition approach is based on deep convolutional neural networks (details follow). The objective of this paper is to study how current similarity techniques can manage visual features obtained by this approach. These features are 4096-dimensional real vectors, which makes it a challenge to efficiently search in large collections. These vectors can be compared using various distance functions, and thus we focus mainly on metric-based search techniques. We analyze the space properties of the visual features extracted from a 20 million image collection and we evaluate the search efficiency of different approaches.

This work was supported by Czech Research Foundation project P103/12/G084.

G. Amato et al. (Eds.): SISAP 2015, LNCS 9371, pp. 237–243, 2015.
DOI: 10.1007/978-3-319-25087-8_22

The Deep Convolutional Neural Networks (DCNN) have a long history in computer vision and machine learning. LeCun et al. [6] introduced an architecture with many hidden layers where weights of lower-level layers are shared as convolution filters. Two decades later, a similar architecture was applied to a large scale visual recognition problem by Krizhevsky et al. [5] and won the ILSVRC 2012 challenge by a large margin. The network was trained to recognize 1k selected ImageNet categories from more than 1M training images. The network consisting of 5 convolutional and 3 fully connected layers takes raw size-normalized images as an input. The network having about 60M parameters represents a very flexible classifier with a great class capacity.

After the great success of [5], researches started to consider a possibility to re-use the representation power of the Krizhevsky's network to solve other recognition problems, i.e. to adapt the classifier to recognize classes the network was not trained for, by using a much smaller dataset [3,12]. The last network layer that outputs the class scores is in fact a linear classifier taking a non-linear representation of an image (the previous layer response). The response of the last hidden layer of Krizhevsky's network is coined the DeCAF feature in [3]. The feature is demonstrated to hold a great representation power and ability to generalize to other recognition tasks by training a simple linear classifier. Moreover, a semantic information is carried implicitly which is shown by tendency of the DeCAF features to cluster semantically similar images of categories on which the network was never explicitly trained.

The DCNN methods were also studied in the context of CBIR [13], using the deep features and a suitable metric. A good generalization to various datasets is shown, but attempts to learn a metric are reported to have a minor effect.

2 Similarity Indexing and Searching

In this section, we describe selected techniques for similarity-based indexing.

FLANN. FLANN is a popular library for performing fast approximate nearest neighbour search developed by Muja and Lowe [7]. It is often used in various computer vision problems where a large dataset is involved and it is incorporated into OpenCV. FLANN contains two main algorithms: (1) a forest of randomized k-d trees, and (2) a hierarchical k-means tree. Additionally, FLANN includes a method for an automatic selection of the most suitable algorithm and its parameters given the target dataset. The choice depends on the nature of the set of features where the search is performed; on its size, on the dimensionality, on the structure of the data, and of course on the desired precision of the approximate nearest neighbour search. The auto-tuning algorithm optimizes a score that consists of a weighted combination of the search time, tree build time and memory overhead. The weights are optionally set by a user. The optimization can run on a representative fraction of the dataset only, which further speeds the auto-tuning up and makes the approximate search system easy to set up.

Metric-Based Approaches. Further, we focus on the *metric access methods* (MAMs), that model the data as a metric space $(\mathcal{D}, \delta)$, where $\mathcal{D}$ is the domain

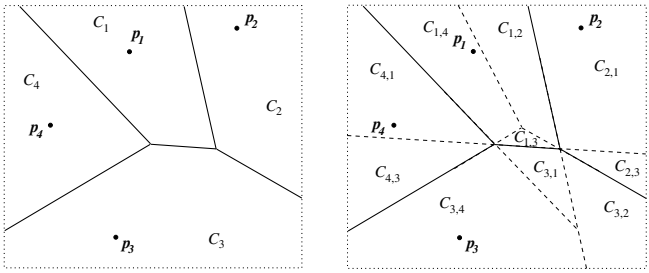

Fig. 1. Voronoi partitioning (left) and of second level Voronoi partitioning (right).

and δ is a metric (distance) function. Specifically, we describe a class of MAMs based on recursive Voronoi partitioning of the space as depicted in Figure 1 for pivots $p_1, \ldots, p_4$. The left part shows standard partitioning which generates four Voronoi cells $C_1, \ldots, C_4$; on the right, each of these cells is partitioned using the rest of the pivots. Cell $C_{i,j}$ then contains objects for which p_i is the closest pivot and p_j is the second closest. This principle can be used recursively l-times and it is often formalized as *permutations of pivots*: objects from Voronoi cell $C_{i_1,\ldots,i_l}$ can be "mapped" onto a vector $\langle i_1, \ldots, i_l \rangle$, which is an l-prefix of a certain permutation of the pivot indexes [8].

This principle has been successfully applied by several MAMs, for instance by a structure called M-Index [8]. This index builds a dynamic trie-like structure over the recursive Voronoi diagram, so that only the *overfilled* cells are partitioned to another level. Given a k-nearest neighbor query k-NN(q), the M-Index forms a *candidate set* of indexed objects by accessing data objects x from the "most promising" Voronoi cells; these candidate objects are refined by evaluation of $\delta(q, x)$ and the best k objects are returned. The Voronoi cell data can be stored either in memory or on the disk in continuous chunks. This approach can be further improved by combining several independent Voronoi partitions [10] in a similar way as in case of randomized k-d forest.

The same space partitioning is used also in a recent technique called PPP-Codes [11]. This MAM defines a mapping of the metric objects onto small codes composed of the pivot permutation prefixes from several pivot spaces. These codes are kept in memory; given a k-NN query, the PPP-Codes search algorithm combines candidate sets from the independent pivot spaces into a small but very accurate candidate set. Only objects from this candidate set are retrieved from the disk and refined. As these objects are read one-by-one (via their identifiers), this approach assumes an efficient key-value store, ideally kept on an SSD disk.

3 Efficiency Evaluation

According to the state of the art in computer vision, we use the DeCAF$_7$ feature produced by the last hidden layer of the neural network model provided by the Caffe project[1] [4], which was trained according to [5]. This 4096-dimensional float

[1] http://caffe.berkeleyvision.org

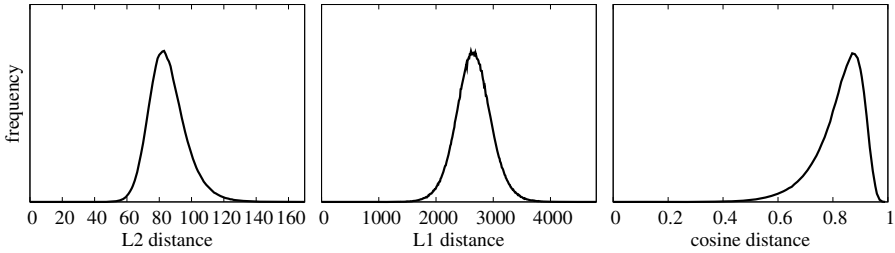

Fig. 2. Distance histograms of the data space with different metric functions; corresponding intrinsic dimensionality values are: L_2: 26.8, L_1: 36.0, cosine distance: 46.9.

vector was extracted from a collection called Profiset [1] consisting of 20 million images provided for research purposes by a microstock photography company. This set of 20M features is public for research purposes at http://disa.fi.muni.cz/profiset/.

In the beginning of this evaluation, we we provide analysis of the feature space properties. Figure 2 shows histogram of distances calculated on a sample of 1M images with three metrics: L_2, L_1 and cosine distance; the figure caption shows also respective values of *intrinsic dimensionality* calculated as $\mu^2/(2 \cdot \sigma^2)$, where μ and σ^2 are the mean and variance of the distance histogram [2].

The core of this section is evaluation of k-NN processing efficiency using L_2. Denoting A the approximate k-NN search answer and A^P the precise NN answer, the answer quality is measured by $recall(A) = precision(A) = \frac{|A \cap A^P|}{K} \cdot 100\%$. The key performance indicator is the wall-clock time of the query processing. All results were averaged over 1,000 queries from the outside of the dataset. We use several subsets of the collection of sizes from 100K to 20M. The evaluation was realized on a 12-core Intel Xeon @ 2.0 GHz machine with 60 GB of main memory, and SSD disk with transfer rate about 270 MB/s with random accesses.

3.1 In-memory Indexes

First, the in-memory FLANN and M-Index were tested on subsets up to 3M objects (48 GB in main memory). The FLANN auto-tuning procedure (running on a 100K sample) chose the randomized k-d tree forest with 32 trees; parameter "number of accessed leaves" varied for different required values of 1-NN recall. The M-Index was configured to use four Voronoi trees [10], each with 512 pivots and the parameter "number of accessed objects" was altered.

Plots in Figure 3 show dependence between the single-thread search times and k-NN recall for FLANN and M-Index on 1M dataset (various values of k). We can see that the results are very similar; this is quite surprising since the partitioning principles and the implementation platforms (C++ vs. Java) of the two indexes differ significantly. FLANN is able to return some results within milliseconds while the M-Index has a minimum response time about 20 ms. This is caused especially by initial calculation of distances between the query object and the set of 4×512 pivots. On the other hand, the recall values grow faster for M-Index, especially for higher values of k.

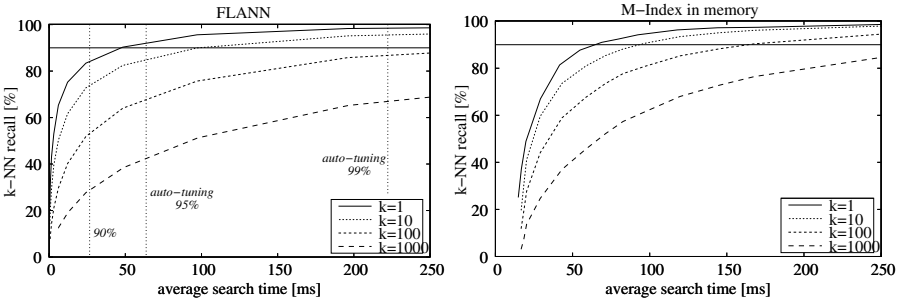

Fig. 3. Recall of k-NN queries vs. search time (single thread) for FLANN and main memory M-Index on 1M data collection.

Figure 4 shows the same type of dependence for 10-NN and collection sizes varying between 100K and 3M. We can see that both indexes scale quite well; the M-Index has again "slower start" but it outruns FLANN in the end.

3.2 Disk-Oriented Indexes

Finally, we analyze how disk-oriented indexes perform on collections up to 20M. First, we test the disk version of M-Index with the same configuration as in the memory case (four indexes, each with 512 pivots, 1M dataset). Since we use four independent space partitioning (in a similar way as locality-sensitive hashing approaches do), the physical data is now replicated *four times*. The left graph in Figure 5 shows the k-NN recall with respect to percentage of data accessed by the index; these results are independent of the memory/data implementation. The right graph compares the search time for various implementations: memory vs. disk and single- vs. multi-thread query evaluation. We can see that the disk variant is feasible with multi-treading. For all disk-oriented experiments, the disk caches were dropped before running every 1000-query batch.

Further, we focus on efficiency of the PPP-Codes index, which has been designed for larger collections of voluminous data objects [11]. In our case, it

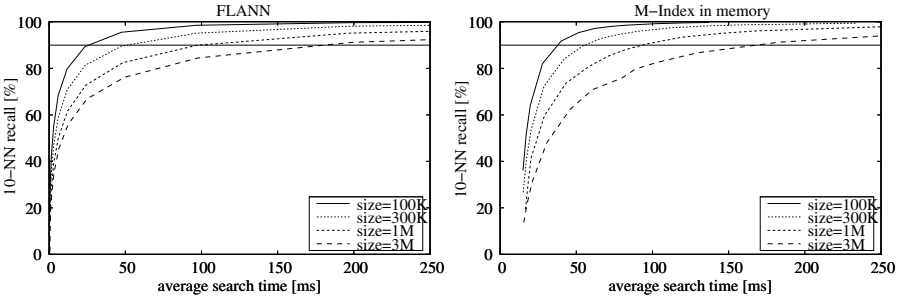

Fig. 4. Recall of 10-NN operations vs. search time (single thread) for FLANN and main memory M-Index on collections of different sizes.

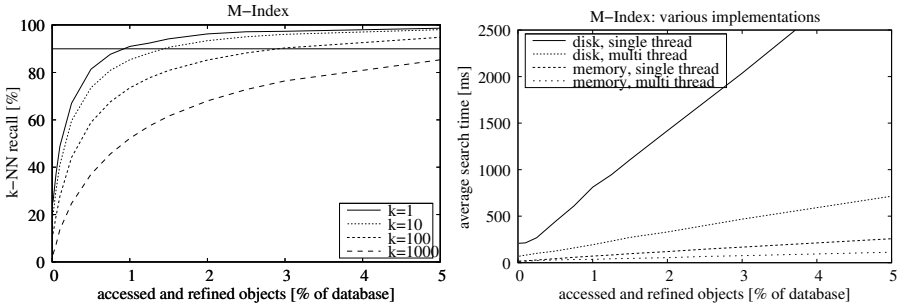

Fig. 5. k-NN recall (left) and search times (right) for different settings (memory/disk and single-/multi-thread processing) for variable perc. of accessed data (1M collection).

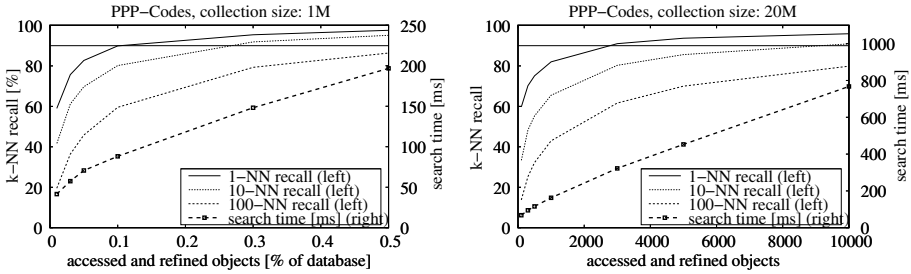

Fig. 6. k-NN recall and search times with respect to accessed objects: 1M and 20M.

uses the same four sets of 512 pivots as the M-Index, it keeps a memory structure (about 1 GB for the 20M collection) and the actual data objects are compressed on the disk (124 GB for the 20M collection).

For comparison with the results above, left graph in Figure 6 shows values of k-NN recall (left vertical axis) and of search times (right axis) with respect to percentage of accessed objects of 1M collection. We can see that PPP-Codes access one order of magnitude fewer objects than M-Index and the search times are about 1/3 of the M-Index with multi-thread processing on the disk. The search time improvement is not proportional to candidate set reduction since the PPP-Codes have more demanding in-memory processing phase and the candidate set objects on the disk are accessed one-by-one [11].

The right graph presents the results on the 20M collection. In this case, the horizontal axis shows the absolute number of accessed objects (out of 20M) and we can see that high recall values are achieved for response times around 500 ms. In practice, lower response times are achieved by not dropping the disk caches.

4 Conclusions

The fusion of the deep neural networks and similarity-based indexing has many good applications in the area of content-based image retrieval. The high dimensionality and bulkiness of the visual features from the neural networks

calls for analysis of actual search efficiency of current indexing techniques on large datasets of this data. We have introduced a test collection with 20M 4096-dimensional image features and tested the k-NN search efficiency of selected indexing techniques. The results indicate that if the data fit into main memory, the metric-based structure M-Index [8] is as efficient as the FLANN [7] library. With the disk version of M-Index, the search would not stay real-time for large datasets because the index accesses over 1 % of the data to produce good results.

The PPP-Codes index [11] better fits this type of datasets as it can achieve fine results accessing around 0.02 % out of the 20M dataset; the search times are around 500 ms. There is an online demonstration application available at http://disa.fi.muni.cz/demos/profiset-decaf/ which presents k-NN visual search on this 20M dataset with the PPP-Codes index [9]. The collection of the 20M descriptors is publicly available at http://disa.fi.muni.cz/profiset/.

References

1. Budikova, P., Batko, M., Zezula, P.: Evaluation platform for content-based image retrieval systems. In: Gradmann, S., Borri, F., Meghini, C., Schuldt, H. (eds.) TPDL 2011. LNCS, vol. 6966, pp. 130–142. Springer, Heidelberg (2011)
2. Chávez, E., Navarro, G.: Measuring the dimensionality of general metric spaces. Technical report, Department of Computer Science, University of Chile (2000)
3. Donahue, J., Jia, Y., Vinyals, O., Hoffman, J., Zhang, N., Tzeng, E., Darrell, T.: DeCAF: a deep convolutional activation feature for generic visual recognition. In: International Conference in Machine Learning (ICML), pp. 647–655 (2014)
4. Jia, Y., Shelhamer, E., Donahue, J., Karayev, S., Long, J., Girshick, R., Guadarrama, S., Darrell, T.: Caffe: convolutional architecture for fast feature embedding. In: International Conference on Multimedia (2014)
5. Krizhevsky, A., Sutskever, I., Hinton, G.E.: Imagenet classification with deep convolutional neural networks. Advances In Neural Information Processing Systems **25**, 1097–1105 (2012)
6. LeCun, Y., Boser, B., Denker, J., Henderson, D., Howard, R., Hubbard, W., Jackel, L.: Backpropagation applied to handwritten zip code recognition. Neural Computation **1**(4) (1989)
7. Muja, M., Lowe, D.G.: Scalable Nearest Neighbour Algorithms for High Dimensional Data. IEEE Trans. on PAMI **36**(11), 2227–2240 (2014)
8. Novak, D., Batko, M., Zezula, P.: Metric Index: An Efficient and Scalable Solution for Precise and Approximate Similarity Search. Information Systems **36**(4), 721–733 (2011)
9. Novak, D., Batko, M., Zezula, P.: Large-scale image retrieval using neural net descriptors. In: Proceedings of SIGIR 2015 (to appear, 2015)
10. Novak, D., Zezula, P.: Performance Study of Independent Anchor Spaces for Similarity Searching. The Computer Journal **57**(11), 1741–1755 (2014)
11. Novak, D., Zezula, P.: Rank aggregation of candidate sets for efficient similarity search. In: Decker, H., Lhotská, L., Link, S., Spies, M., Wagner, R.R. (eds.) DEXA 2014, Part II. LNCS, vol. 8645, pp. 42–58. Springer, Heidelberg (2014)
12. Oquab, M., Bottou, L., Laptev, I., Sivic, J.: Learning and transferring mid-level image representations using convolutional neural networks. In: CVPR (2014)
13. Wan, J., Wang, D., Hoi, S., Wu, P., Zhu, J., Zhang, Y., Li, J.: Deep learning for content-based image retrieval: a comprehensive study. In: Proc. of 22nd ACM International Conference on Multimedia (2014)