# Towards Large-scale Image Retrieval with a Disk-only Index

Daniel Manger[1], Dieter Willersinn[1] and Jürgen Beyerer[1,2]
[1] *Fraunhofer IOSB, Karlsruhe, Germany*
[2] *Karlsruhe Institute of Technology KIT, Vision and Fusion Lab, Karlsruhe, Germany*

Keywords: Content-based Image Retrieval, Bag-of-words, Convolutional Neural Networks, Index.

Abstract: Facing ever-growing image databases, the focus of research in content-based image retrieval, where a query image is used to search for those images in a large database that show the same object or scene, has shifted in the last decade. Instead of using local features such as SIFT together with quantization and inverted file indexing schemes, models working with global features and exhaustive search have been proposed to encounter limited main memory and increasing query times. This, however, impairs the capability to find small objects in images with cluttered background. In this paper, we argue, that it is worth reconsidering image retrieval with local features because since then, two crucial ingredients became available: large solid-state disks providing dramatically shorter access times, and more discriminative models enhancing the local features, for example, by encoding their spatial neighborhood using features from convolutional neural networks resulting in way fewer random read memory accesses. We show that properly combining both insights renders it possible to keep the index of the database images on the disk rather than in the main memory which allows even larger databases on today's hardware. As proof of concept we support our arguments with experiments on established public datasets for large-scale image retrieval.

## 1 INTRODUCTION

While being a lively field of research in computer vision, content-based image retrieval (CBIR) has also become apparent in many applications, including, for example, countless apps for recognizing specific items based on photos taken with mobile devices. However, today's typical CBIR setups are restricted to a limited amount of database images to be searched in - typically not exceeding a few million images. In other words, even 15 years after the introduction of local features such as SIFT (Lowe, 2004), which boosted computer vision in many aspects, it has not yet been achieved to scale that original accuracy of comparing a pair of images with local features towards current huge web-scale databases containing billions of images. The main reason for that seems, that the models typically are based on a codebook-based quantization of feature descriptors into visual words (Sivic and Zisserman, 2003). This representation of local features, commonly termed Bag-of-Words (BoW), allows to manage the image retrieval problem with established methods from text retrieval, i.e. to leverage an inverted file indexing scheme with the resulting index being stored in main memory for speedup reasons. As the quantization step impairs the discriminative power of the local features, the retrieval accuracy in large-scale datasets is limited. As a consequence, research turned towards global features where all local features of an image are aggregated into a high-dimensional embedding such as VLAD (Jégou et al., 2010) or Fisher Vectors (Perronnin and Dance, 2007) followed by a compression step - usually PCA and whitening - resulting in a compact fixed-length code per image. Eventually, after the sustained success of convolutional neural networks (CNN), deep-learned features have been proposed which, for example, compile global features from pooling the responses of CNN layers (Babenko and Lempitsky, 2015; Kalantidis et al., 2015; Razavian et al., 2014; Tolias et al., 2015).
Since those global image representations are targeted to yield very compact codes suitable for fast exhaustive search, retrieving small objects surrounded by plenty of cluttered background becomes difficult. In this work, we argue that two recent technical and algorithmic advances offer themselves to be combined with the bag-of-words model for image retrieval: First, the latest hardware progress in solid-state disks enables extremely fast random read access to large on-disk datasets. Second, extending the discriminability of local features by considering their larger

Manger, D., Willersinn, D. and Beyerer, J.
Towards Large-scale Image Retrieval with a Disk-only Index.
DOI: 10.5220/0006631303670372
In *Proceedings of the 13th International Joint Conference on Computer Vision, Imaging and Computer Graphics Theory and Applications (VISIGRAPP 2018) - Volume 5: VISAPP*, pages 367-372
ISBN: 978-989-758-290-5

neighborhood utilizing representations from CNNs, can be used to induce a second dimension into the index which results in far fewer memory accesses for processing one query image. Leveraging both circumstances allows to implement a retrieval system whose index no longer needs to reside in the main memory but can remain on the larger and less expensive solid-state drive.

This paper is structured as follows: Section 2 presents related work on extensions of the BoW model, Section 3 outlines the relevant technical details for using solid-state disks for storing the index. Section 4 then describes the experiments starting with insights of the 2D index with CNN features and Section 5 focuses on the evaluation of the proposed retrieval system and its results in terms of accuracy, speed and memory usage.

## 2 RELATED WORK

Many advancements of the bag-of-words model have been proposed which aim at different aspects of the image retrieval pipeline in order to incorporate more information into the retrieval process. In this work, we neglect methods refining the shortlist (re-ranking) and focus on those approaches that incorporate additional information into the index so that *all* images can benefit from. These can be separated into three strategies:

*Extending the accumulator* which holds bins for the scores of all the database images by new dimensions assuming that irrelevant features will spread along multiple bins of one database image while corresponding features will accumulate in one or few of the bins of a similar image. For instance, (Jegou et al., 2008) use orientation and scale information of SIFT features to push database images with features having consistent differences in scale and orientation compared to the query image.

*Filtering of features:* This keeps the accumulator compact (still one bin per database image) and adds additional information into the index to filter matches prior to casting votes into the accumulator. (Zhang et al., 2013) integrates information about the four closest features in the image coordinate space and during retrieval, each BoW match is further examined as to how many of the four neighboring features are consistent.

*2D-Index:* In order to overcome the runtime, performance and storage limits of both accumulator extension and filtering of features, (Zheng et al., 2014) uses a multi-index. The first dimension of the index is still dedicated to the BoW vectors while the second dimension is based on the color name descriptor (Khan et al., 2012), which is an 11-dimensional descriptor mapping color values to 11 categories. Using a Color-Codebook of size 200, every feature in the index is assigned up to the 100 closest Color-Words which however obviously eliminates the advantages of the second dimension because still up to 50% of the index has to be traversed. Recently, (Manger and Willersinn, 2017) proposed to use features pooled from convolutional neural networks to represent the features' larger neighborhoods inducing the second dimension. More precisely, they sum- or max-pool the activations in the 512 channels of the last convolutional layer of the VGG16 network (Simonyan and Zisserman, 2014). The pooling is performed in a rectangular region defined by the local feature's position and scale to preserve the invariance w.r.t. translation, scale and rotation of the underlying local features.

In this paper, we focus on the latter strategy and adopt the approach of (Manger and Willersinn, 2017) since when targeting the index to reside on a solid-state disk, adding such a new dimension to the index is attractive in multiple aspects: In contrast to the filtering strategy, the runtime during retrieval can be optimized because only features which match both dimensions have to be considered for the accumulator leading to fewer memory accesses. Furthermore, retrieval accuracy can benefit from the second dimension because many incorrect matches of features are discarded that match w.r.t. the first dimension only (the quantized local feature descriptor) but not in the second dimension (the larger context of the feature described by the CNN feature). We also consider the combination of max- and sum-pooled CNN features using `OR` and `AND` combinations, e.g. for the latter, both the max-pooled and the sum-pooled values must match after quantization (and of course the respective visual words) in order to cast votes in the accumulator.

## 3 SOLID-STATE DISKS FOR RETRIEVAL

Solid-state drives (SSD) are storage devices based on flash-memory and without any moving parts like there are in hard drives. Bits are stored in cells which are grouped into blocks consisting of a number of pages which is the smallest unit for reading or writing.

In order to provide the same host interface as hard-drives and since flash-cells are wearing off, SSDs require a controller with a flash translation layer which maps the logical block adresses to the physical blocks of the flash memory space and which manages the wearing off. See (Goossaert, 2014) for more details in SSDs in view of coding.

In this paper, we focus on the characteristics with respect to random read access since this is the relevant property when using SSDs for retrieval[1]. In our scenario, the most important property of a SSD for retrieval is the number of input/output operations per second (IOPS). However, the performance values in the data sheets are usually presented for random reads of larger chunks, e.g. 4,096 bytes. For smaller sizes such as 16 bytes, the read-ahead buffering technique is not effective and the controller has to perform much more address mappings resulting in lower IOPS. Furthermore, random reads can be performed in parallel (referred to as query depth in the data sheets) in order to take advantage of the internal parallelism based on spreading data across several flash chips. Finally, the data to be read should be aligned to the size of a page since otherwise, even reading two bytes could require reading two full pages instead of one. Figure 1 shows the IOPS for 10,000 random read operations in a 200 GiB file using different query depths and chunk sizes. For each curve, we average the IOPS from ten runs and clear the cache of the operating system each time to prevent distortion from caching effects. See Section 4.2 for details on the hardware. As can be seen, performance saturates at 20 parallel random read operations and the page size turns out to be 512 bytes since reading even one more byte (513 bytes) takes almost twice as long. For our experiments, we therefore use a query depth of 20 in the following.

## 4 EXPERIMENTS

We first describe the setup used for our experiments, followed by statistics on the features to be included into the 2D index and the design decisions taken to derive the final retrieval system.

### 4.1 Setup and Datasets

For the 2D index, we follow the setup of (Manger and Willersinn, 2017) using a visual Codebook of size 100,000 generated with SIFT features (Lowe, 2004) and RootSift normalization (Arandjelović and Zisserman, 2012). For the second dimension characterizing the larger neighborhood of the features, we adopt

[1]However, since SSDs provide internal parallelism, including by spreading data across several NAND-flash chips, the read performance also correlates with the write pattern, e.g. related data should be written together. Since, in our retrieval scenario, we don't know in advance which data will be related, we leave this as a footnote in this work.

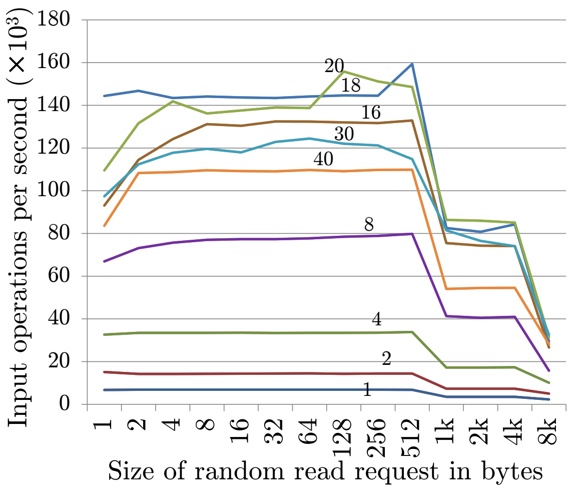

Figure 1: Performance of the tested SSD for 10,000 random reads using different query depths (1 to 40) and for varying sizes of read requests (1 to 8,196 bytes).

the Codebook size of 10,000 for the max- and sum-pooled CNN features. All models (Codebooks) are generated using the Oxford5k dataset (Philbin et al., 2007) which contains 5,062 images. For evaluation, we use the Paris6k dataset (Philbin et al., 2008) with 6,392 images including five query images and several corresponding images for each of 11 distinct buildings. Furthermore, we add one million images from the MIRFlickr1M dataset (Mark J. Huiskes and Lew, 2010) as distractors. In this setup containing over 1M images yielding over 1 billion quantized local features in the index, the benefit of distributing them along two dimensions becomes obvious in Table (top) 1. Compared to a plain BoW index, the second dimension cuts down the number of required features to be processed by a factor of at least 1,500.

### 4.2 Hardware

All experiments in this paper are run using implementations in C++ on a system with two Intel Xeon E5-2630v4 10-Core CPUs using all cores. For the separate SSD which only stores the index, we use a Samsung SSD 960 Pro M.2 1TB with NVMe protocol and PCIe 3.0 x4.

### 4.3 Discriminative 2D Index

In traditional Bag-of-Words-based image retrieval, the index containing all quantized (using the Codebook) features of the database images is read from disk to main memory during the start of the software. The index is composed of a set of lists, where one list of variable length for each visual word of the

Table 1: Number of quantized features in the index (containing Paris6k and 1M distractor images) which are accessed for processing one query image for different index configurations. In total, the index in main memory contains $1.27 \times 10^9$ features. The numbers represent the median for the 55 query images of Paris6k since some of them contain very much features due to heavily cluttered backgrounds.

| **Index configuration** | **number of features processed** | | | | |
|---|---|---|---|---|---|
| Index in main memory | | | | | |
| BoW only | 44,720,650 | | | | |
| BoW + CNN-max | 15,296 | | | | |
| BoW + CNN-sum | 17,238 | | | | |
| BoW + CNN-sum AND CNN-max | 2,945 | | | | |
| BoW + CNN-sum OR CNN-max | 29,321 | | | | |
| Index on SSD using a maximum number of features per cell of: | 16 | 8 | 4 | 2 | 1 |
| BoW + CNN-max | 9,119 | 7,187 | 5,310 | 3,657 | 2,299 |
| BoW + CNN-sum AND CNN-max | 1,665 | 1,263 | 866 | 560 | 336 |

Codebook contains all features assigned to that visual word. However, when adding the second dimension to the index based on the CNN features and aiming at keeping the new 2D-index on the solid-state disk only, the resulting 2D structure forces a fixed maximum number of features per 2D cell in order to quickly locate each of the $10^5 \times 10^4 = 10^9$ cell on the disk. Unfortunately, the features do not spread evenly across both dimensions[2], which raises the question about the maximum number of features per cell. Too low values will discard many features whereas too high values will require too much memory and lower the 'filling level' of the cells. We therefore analyze the retrieval accuracy for different values.

For each of the CNN feature variants (max-pooled and sum-pooled) we create a separate index file. When using the `OR` combination of both variants, we read the appropriate features using both files but refrain from filtering out the few overlapping features which match in both variants (and thus voting twice in the accumulator). Similarly, in case of the `AND` combination, we also avoid to read both sets of features and then search for duplicates. Instead, we use the filtering approach (see. Section 2), i.e. in the index designated to the max-pooled features, we store for each feature not only its image index in the database but also its sum-pooled value which is then used for filtering. Please note that, in contrast to the filtering strategies described in Section 2 such as Hamming Embedding, this virtually requires no overhead since the number of features is tiny compared to BoW-only, and the filtering step is just comparing one integer value per feature.

[2]which can be observed using the Gini coefficient which is 0.2 for the distribution of features across the visual words, and 0.6 for the CNN features, respectively for our database setup.

In the index file, we use 12 bytes for each feature: 4 bytes for the image number in the database, 2 bytes for the quantized CNN feature of the respective other pooling variant (for realizing the `AND` combination as mentioned in the last paragraph) and reserve 6 more bytes for additional payload (which is however not used in this work, e.g. scale and orientation of a feature or Hamming Embedding of its descriptor). In each cell we allow up to 16 features which leads to a final size of the SSD index file of $10^5 \times 10^4 \times 16 \times 12$ bytes $= 178$ GiB. Using these parameters, the cell size is $16 \times 12 = 192$ bytes and thus below the SSD page size of 512 bytes (see Section 3) which allows to read all of the up to 16 features in the cell using one random read operation.

## 5 EVALUATION

Figure 2 presents the results of the large-scale retrieval experiment on the Paris6k dataset together with 1 million distractor images and using standard protocols including the mean average precision (MAP) metric for evaluation. MAP results are shown as a function of the maximum number of features per cell (x-coordinate). For each value, we also determined how many features hence fit into the SDD index (see 'Features used', aligned to the right axis). On the other hand, despite for some cells, not all features can be saved, nearly 60% of the cells remain empty since no feature takes their visual word and CNN-feature values. Using the `OR` combination of CNN features and taking just three features per cell already outperforms the BoW-model, although only 40% of the features can be stored in the SSD index in this configuration. Furthermore, the performance approaches saturation when including at least ten features per cell

(70% of all features). Interestingly, using only 80% of all features, the SSD index can even slightly outperform the original retrieval using all features in the main memory. This suggests, that by restricting the number of features in terms of their quantized CNN feature, one can filter out features with a frequently occurring larger neighborhood which negatively affect the retrieval performance. Potentially, these are the features arising on watermark symbols or timestamps in the images.

Table 1 (bottom) shows, how the different cell sizes affect the number of features processed during one query. As only several thousand features generate votes into the accumulator for over one million database images, the accumulator could be realized with a sparse implementation which speeds up both initialization and the final sorting to obtain the similarity scores.

Finally, Table 2 presents retrieval times for the different setups. We exclude durations for feature extraction of the query image and accumulator evaluation thus focussing on the retrieval of the features in the index. For the variants using SSD index, we present numbers for a maximum cell size of 16 features since, due to the page size of the SSD, access times for smaller cell sizes are identical anyway (see Section 3). Astonishingly, the models accessing the index on the solid-state disk are one order of magnitude faster than the BoW model working with the main memory and all features. Of course, this is only possible because, in our proposed setup, far less features have to be accessed which is in turn a consequence of the more discriminative representation based on CNN features. For a rough comparison, we also report the processing time using our hardware (again using all cores) for image retrieval with global features, i.e. exhaustively comparing the query vector with all 1M database vectors (in main memory) assuming a 256 dimensional global feature vector for each image.

Please note that, ultimately, the configuration with the AND-combination of CNN features could even be one magnitude faster if the SSD was large enough to allow indexing along all three dimensions (BoW $\times$ CNN-max $\times$ CNN-sum; see Section 4.3 for our workaround implementation). However, even when only using one feature per cell and storing just the image index inside, such a setup would need $10^5 \times 10^4 \times 10^4 \times 4$ bytes = 36 TiB. Nevertheless, quantizing the CNN-features with smaller codebooks can offer flexible trade-offs between retrieval speed and SSD memory usage.

Table 2: Retrieval time for one query on Paris6k + 1M dataset (mean of all 55 query images).

| **Index configuration** | **time in ms** |
|---|---|
| BoW only (index in main memory) | 369.8 |
| Global features with exhaustive search: $10^6$ vectors, 256D | 28.1 |
| BoW + max (SSD) | 24.8 |
| BoW + max AND sum (SSD) | 36.6 |
| BoW + max OR sum (SSD) | 36.1 |

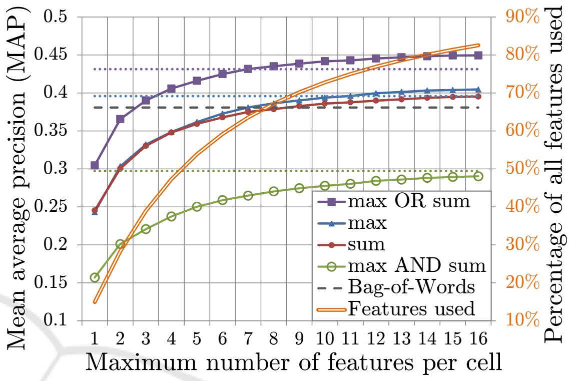

Figure 2: Retrieval performance for the Paris6k dataset with 1 million distractor images. Results of using the proposed SSD index (incorporating max- and sum-pooled CNN features and their combinations) are shown in solid lines. For comparism, dotted lines indicate the conventional BoW setup + CNN features, i.e. the index being located in the main memory, all features are used, and their CNN feature extensions are used via filtering. The dashed line represents the plain BoW setup working with visual words only, i.e. without any CNN features.

## 6 CONCLUSION

In this work, we have outlined the benefits for local feature-based image retrieval when leveraging the combination of a discriminative CNN-based feature extension together with the ultrashort random read access times of modern solid-state disks. Beyond being faster, the proposed method gets rid of the necessity of keeping the index of all features in the main memory.

Underpinned by our experiments retrieving similar images in a database of over one million images in less than 40 milliseconds, we showed that the proposed combination can enable retrieval times which are similar to the methods using ultra-compact signatures derived from global image features. While enabling still larger databases on computers and servers, this could, in principle and for some applications, even pave the way for running large-scale image retrieval on mobile devices where large flash drives are becoming common but computational power and main memory is still limited.

## ACKNOWLEDGEMENTS

This work was supported by the German Ministry of Education and Research (BMBF) (grant number 13N14028).

## REFERENCES

Arandjelović, R. and Zisserman, A. (2012). Three things everyone should know to improve object retrieval. In *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on*, pages 2911–2918. IEEE.

Babenko, A. and Lempitsky, V. (2015). Aggregating local deep features for image retrieval. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1269–1277.

Goossaert, E. (2014). Coding for ssds. http://codecapsule.com/2014/02/12/coding-for-ssds-part-1-introduction-and-table-of-contents/.

Jegou, H., Douze, M., and Schmid, C. (2008). Hamming embedding and weak geometric consistency for large scale image search. In *European conference on computer vision*, pages 304–317. Springer.

Jégou, H., Douze, M., Schmid, C., and Pérez, P. (2010). Aggregating local descriptors into a compact image representation. In *Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on*, pages 3304–3311. IEEE.

Kalantidis, Y., Mellina, C., and Osindero, S. (2015). Cross-dimensional weighting for aggregated deep convolutional features. *arXiv preprint arXiv:1512.04065*.

Khan, F. S., Anwer, R. M., Van De Weijer, J., Bagdanov, A. D., Vanrell, M., and Lopez, A. M. (2012). Color attributes for object detection. In *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on*, pages 3306–3313. IEEE.

Lowe, D. G. (2004). Distinctive image features from scale-invariant keypoints. *International journal of computer vision*, 60(2):91–110.

Manger, D. and Willersinn, D. (2017). Extending the bag-of-words representation with neighboring local features and deep convolutional features. In *2017 Irish Machine Vision and Image Processing Conference (IMVIP)*.

Mark J. Huiskes, B. T. and Lew, M. S. (2010). New trends and ideas in visual concept detection: The mir flickr retrieval evaluation initiative. In *MIR '10: Proceedings of the 2010 ACM International Conference on Multimedia Information Retrieval*, pages 527–536, New York, NY, USA. ACM.

Perronnin, F. and Dance, C. (2007). Fisher kernels on visual vocabularies for image categorization. In *2007 IEEE Conference on Computer Vision and Pattern Recognition*, pages 1–8. IEEE.

Philbin, J., Chum, O., Isard, M., Sivic, J., and Zisserman, A. (2007). Object retrieval with large vocabularies and fast spatial matching. In *2007 IEEE Conference on Computer Vision and Pattern Recognition*, pages 1–8. IEEE.

Philbin, J., Chum, O., Isard, M., Sivic, J., and Zisserman, A. (2008). Lost in quantization: Improving particular object retrieval in large scale image databases. In *Computer Vision and Pattern Recognition, 2008. CVPR 2008. IEEE Conference on*, pages 1–8. IEEE.

Razavian, A. S., Sullivan, J., Maki, A., and Carlsson, S. (2014). A baseline for visual instance retrieval with deep convolutional networks. *arXiv preprint arXiv:1412.6574*.

Simonyan, K. and Zisserman, A. (2014). Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*.

Sivic, J. and Zisserman, A. (2003). Video google: A text retrieval approach to object matching in videos. In *Computer Vision, 2003. Proceedings. Ninth IEEE International Conference on*, pages 1470–1477. IEEE.

Tolias, G., Sicre, R., and Jégou, H. (2015). Particular object retrieval with integral max-pooling of cnn activations. *arXiv preprint arXiv:1511.05879*.

Zhang, S., Tian, Q., Huang, Q., Gao, W., and Rui, Y. (2013). Multi-order visual phrase for scalable image search. In *Proceedings of the Fifth International Conference on Internet Multimedia Computing and Service*, pages 145–149. ACM.

Zheng, L., Wang, S., Liu, Z., and Tian, Q. (2014). Packing and padding: Coupled multi-index for accurate image retrieval. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1939–1946.