Efficient and compact visual feature descriptors hashing using hierarchical multiple assignment k-means

Conference Paper · July 2017

DOI: 10.1109/ICMEW.2017.8026220

Marco Bertini
University of Florence

EFFICIENT AND COMPACT VISUAL FEATURE DESCRIPTORS HASHING USING HIERARCHICAL MULTIPLE ASSIGNMENT K-MEANS

Simone Ercoli, Marco Bertini, Alberto Del Bimbo

Università degli Studi di Firenze - MICC
simone.ercoli, marco.bertini, alberto.delbimbo @unifi.it

ABSTRACT

In this paper we present an efficient and effective method for visual descriptors hashing based on hierarchical multiple assignment within a k-means framework. The method has been used to address the problem of approximate nearest neighbor (ANN) retrieval, and it has been tested on local and global visual content descriptors, either engineered or learned. The proposed method has been compared to state-of-the-art methods on different standard large-scale datasets composed by millions of visual features: SIFT 1M and GIST 1M (BIGANN), and also on the recent DEEP1M dataset, composed by one million CNN-based features. Experimental results show that, despite its simplicity, the proposed method obtains an excellent performance.

***Index Terms*—** Image retrieval, approximate nearest neighbor retrieval, hashing, SIFT, CNN.

1. INTRODUCTION

The inception of web-scale visual archives, in which content is represented using high-dimensional feature vectors, calls for efficient and effective methods to perform retrieval. Nearest neighbor (NN) retrieval is one of the main tasks for large scale multimedia archives and for many computer vision tasks. Since even methods designed for high-dimensional features indexing obtain a performance that is comparable to that of linear search [1], a solution to speed up retrieval is to compress the dimensionality of the descriptor. This is beneficial also to address the problem of storing image descriptors, either in case of large-scale archives, or in case of systems with limited memory. In this approach, typically, an approximate nearest neighbor (ANN) search is performed computing the Hamming distance on binary features, obtained from feature hashing. In this way it is possible to compress features that consist of hundreds of floats (e.g. SIFT descriptors) or thousands of floats (e.g. CNN descriptors) to a few bytes (e.g. 64 bits). This reduction makes it possible to store large scale archives, e.g. of 1 billion images, in the main memory of a standard PC, and obtain a reasonable performance in terms of speed and retrieval.

In this paper we present a novel method for feature hashing, based on k-means, that is based on a quantized version of soft assignment in which features are associated to multiple cluster centers, and where the selection of these cluster centers is performed hierarchically to compress the binary descriptor from 4096 bits to 64 bits. The method is unsupervised, requires a very limited training set and also the resulting codebook size is very small, resulting in a small memory and computational cost that is suitable for large-scale archives. Evaluation has been performed on two large scale standard datasets, i.e. BIGANN and DEEP1M, each one composed by a million of features; using these datasets shows that the proposed method can be applied both to local and global visual features, both engineered (SIFT and GIST in BIGANN) and learned (CNN in DEEP1M). In terms of retrieval performance the proposed method obtains results that are better or, in a few cases, comparable to those of more complex state-of-the-art approaches.

2. PREVIOUS WORKS

The main and most recent works related to visual feature hashing can be divided in methods based on vector quantization and its many variations and, with the advent of CNN descriptors, on neural networks.

Vector Quantization. The Product Quantization (PQ) method, proposed by Jégou *et al.* [1], consists in the decomposition of the feature space into a Cartesian product of subspaces with lower dimensionality, that are quantized separately. This approach solves the memory issues that arise when using simpler vector quantization methods such as k-means, because it requires a much smaller number of centroids. The method has obtained state-of-the-art results on a large scale SIFT features dataset, improving over previous methods such as Spectral Hashing [2] and Hamming Embedding [3].

Building upon the success of the Product Quantization method, several other variations and improvements have been proposed. Norouzi and Fleet [4] have further explored the idea of compositionality of the PQ approach, building upon it two variations of k-means: Orthogonal k-means and Cartesian k-means (ck-means). The improvement of PQ proposed

978-1-5386-0560-8/17/$31.00 ©2017 IEEE ICMEW 2017

by Ge *et al.* [5], called OPQ, minimizes quantization distortions w.r.t. space decomposition and quantization codebooks; He *et al.* [6] have approximated the Euclidean distance between codewords in k-means method, proposing an affinity-preserving technique. Kalantidis and Avrithis [7] have proposed a simple vector quantizer (LOPQ) using a local optimization over a rotation and a space decomposition and applying a parametric solution that assumes a normal distribution. Guo *et al.* [8] have recently improved over OPQ and LOPQ methods, adding two quantization distortion properties of the Residual Vector Quantization (RVQ) model, with the goal of restoring, instead of reducing, quantization distortion errors. Babenko and Lempitsky have recently proposed in [9] an efficient tree-based dynamic programming method, minimizing the compression error of the descriptors that are hashed. The authors have introduced a new dataset composed by 1 million CNN image descriptors.

A few works have proposed the use of PQ to build indexing structures. Babenko and Lempitsky [10] have proposed an inverted multi-index (IMI), i.e. an efficient similarity search method that generalizes the standard inverted index by replacing vector quantization inside the inverted indices with product quantization; the multi-index is built as a multi-dimensional table (Multi-D-ADC). The same authors have more recently addressed the problem of indexing CNN features [11], observing the inefficiency of IMI to index such features, and proposing two extensions: the Non-Orthogonal Inverted Multi-Index (NO-IMI) and the Generalized Non-Orthogonal Inverted Multi-Index (GNO-IMI).

Neural Networks. Most of the vector quantization methods have been originally proposed and tested on engineered features, typically SIFT descriptors. Since the successful introduction of CNN features also for image retrieval, and not only for classification, several approaches designed specifically for these features have been proposed.

A deep learning framework for the creation of hash-like binary codes for image retrieval has been proposed by Lin *et al.* [12]. A hidden layer is used to represent the latent concepts that dominate the class labels (when these labels are available). This layer learns specific image representations and, at the same time, a set of hash-like functions. also the hashing method proposed by Xia *et al.* [13] simultaneously learns a representation of images and a set of hash functions.

The deep model proposed by Do *et al.* [14], learns binary hash codes for image retrieval by preserving similarity, balance and independence of images. Two sub-optimizations during the learning process allow to efficiently solve binary constraints.

The method proposed by Guo and Li [15] obtains the binary hash code of a given image using binarization of the CNN outputs of specific fully connected layers.

The deep neural network model proposed by Zhang *et al.* [16] for supervised learning of hash codes (called VDSH), is based on a training algorithm inspired by alternating direction method of multipliers (ADMM, originally presented in [17]). The training process is decomposed into independent layer-wise local updates through auxiliary variables.

Hashing of multimodal data has been addressed in the deep learning method by Wang *et al.* [18], where a multimodal Deep Belief Network is used to capture correlation in high-level space during pre-training. This step is followed by learning a cross-modal autoencoder in a fine tuning phase.

A two steps method for CNN features hashing has been proposed by Lin *et al.* [19]. In the first step binary embedding functions are learned by Stacked Restricted Boltzmann Machines, then fine tuning is performed to retain the metric properties of the original feature space.

3. THE PROPOSED METHOD

The proposed method introduces a substantial variation of the multiple assignment k-means vector quantization approach introduced in [20]. This new methods treats the assignment of a visual feature to multiple cluster centers with a hierarchical process during the quantization process. As in this previous method we can, using small training data, greatly reduce the number of required cluster centers, but getting higher performance. The result is a compact hash code of visual features. The computation is divided into three main steps.

First step. Repeats the *multi-k-means* process presented in [20] in which we exploit a typical k-means algorithm to obtain a dictionary (Eq. 1)

$$argmin_{\mathbf{s}} \sum_{i=1}^{k} \sum_{\mathbf{x} \in S_i} \parallel \mathbf{x} - C_i \parallel^2 \tag{1}$$

where we try to minimize the sum of of distance functions of each observations $(\mathbf{x})$, where each observation is a D-dimensional float vector, e.g. a SIFT or CNN feature, to the C_k centers of the cluster. The final intent is to find a good partitioning into k sets $\mathbf{S} = \{S_1, S_2, \ldots, S_k\}$.
Once we obtain our k centroids, we use these points to create hash codes following the Eq. 2:

$$\begin{cases} \parallel x - C_j \parallel \leq \delta & j^{th}\ bit = 1 \\ \parallel x - C_j \parallel > \delta & j^{th}\ bit = 0 \end{cases} \tag{2}$$

where x is the feature point and δ is a threshold measure given by:

$$n^{th}\ nearest\ distance\ \parallel x - C_j \parallel \quad \forall j = 1, ..., k \tag{3}$$

This means that the centroid j is associated to the j^{th} bit of the hash code of length k, and the bit is set to 1 based on a predefined number n of nearest centroids. This first step is illustrated in Fig. 1.

We used $k = 4096$ and $n = 1024$, so to have a very refined initial decomposition of the space of the descriptors.

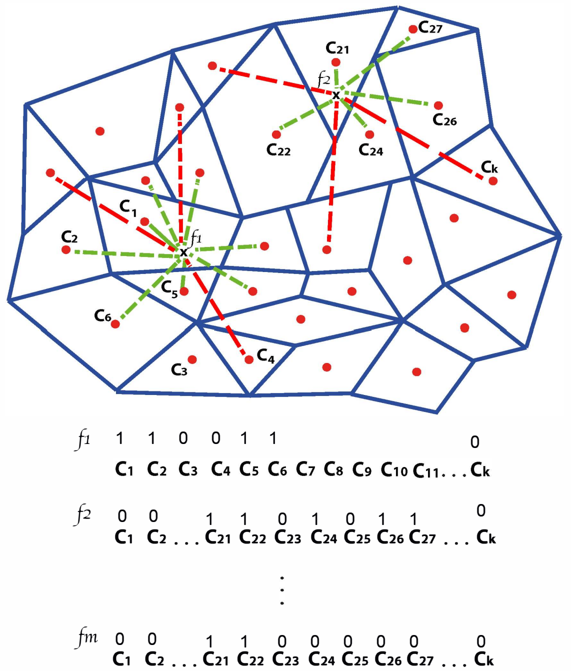

Fig. 1. Illustration of the first step of the process: if a feature is assigned to a centroid (*green line*) the corresponding bit in the hash code is set to 1, otherwise (*red line*) it is set to 0. The number of total centroids k is equal to 4096 while the number n of nearest centroids used as threshold is equal to 1024.

The value of k is quite high to provide a good speedup in feature retrieval, and is reduced in the following step.

Second step. During the second step of our process we compute a global histogram of the bins of the hash codes using this formula:

$$bin_{C_z} = \sum_{i=1}^{m} f_i(C_z) \quad \forall z = 1, ...k \tag{4}$$

where each bin_{C_z} represents the position of each centroid, m is the number of the hash codes and C_z is the z^{th} centroid. Once we obtain the histogram we select only the first 64 most populated bins, that correspond to the associated centroids (Fig. 2).

The goal of this step is to select the most representative centroids obtained in the previous step.

Third step. In this step our intent is to find a final good partitioning of the descriptors space into k sets $\mathbf{S} = \{S_1, S_2, \ldots, S_k\}$ using the 64 centroids coming from the previous step (that are used as "*seeds*"), i.e. as starting point for a new re-training. In this case we are trying to obtain a better decomposition of the descriptor space and consequently a better hash codes representation. The final number of centroid k is still 64, a figure typically used in the scientific literature, but if needed it can be further reduced (eg. to create hashes of just 32 bits). In this case we use the same approach presented in the step 1 to create the hash codes (2) but with a different

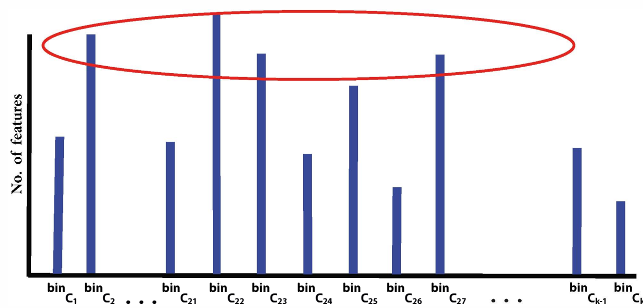

Fig. 2. Histogram generated during the second step of our process. Each bin bin_{C_k} is associated with the position of each centroid C_k, and contains the number of features that have been assigned to the centroid (remind that features are assigned to multiple centroids).

threshold, which is computed using Eq. 5:

$$\delta = \frac{1}{k} \sum_{j=1}^{k} \parallel x - C_j \parallel \tag{5}$$

This step is represented in Figure 3, that shows how the feature indicated by x is assigned to the centroids $C_1, C_3, C_5, C_6, \ldots$; each centroid is again associated with a specific bit position in the resulting hash code..

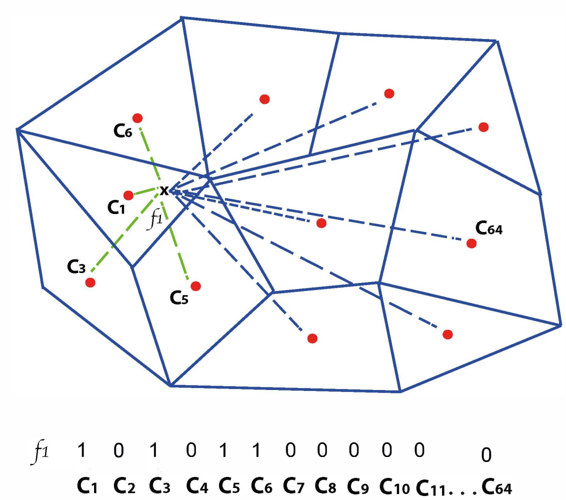

Fig. 3. Assignment example during the third step after the re-training computation. The final length of our has codes is equal to 64.

3.1. Computational Cost

Let us consider a vector with dimensionality D, and desired hash code length of 64 bits. Standard k-means has an assignment complexity of kD, where $k = 2^{64}$. The proposed approach has three steps where in the first stage needs $k' = 4096$

Table 1. Datasets characteristics

vector dataset	SIFT1M	GIST1M	DEEP1M
descriptor dimensionality D	128	960	256
# learning set vectors	100,000	500,000	100,000
# database set vectors	1,000,000	1,000,000	1,000,000
# queries set vectors	10,000	1,000	1,000
# nearest vectors for each query	100	100	1

centroids, the second stage is only an histogram calculation and the third step needs of $k'' = 64$ centroids. This means that this approach has a complexity of $(k'+k'')D$ and requires $(k' + k'')D$ floats to store the codebook. Product Quantization requires instead $k^* \times D$ floats for the codebook and has an assignment complexity of k^*D, where $k^* = k^{1\backslash m}$ using typically $k^* = 256$ and $m = 8$, for a 64 bit length. This means that in a three steps scenario PQ approach needs of a cost $40\times$ greater than our, while a standard PQ has a cost that is one order of magnitude smaller. The complexity of the proposed method at query time, to hash a query descriptor, is the same of Product Quantization.

4. EXPERIMENTAL RESULTS

The proposed method has been thoroughly compared to several state-of-the-art approaches using standard datasets, experimental setups and evaluation metrics.

Datasets

BIGANN Dataset [1, 21] is a large-scale dataset commonly used to compare methods for visual feature hashing and approximate nearest neighbor retrieval [1,4,5,7,10,21,22]. The dataset comprises SIFT and GIST descriptors, and is composed by three subsets, for each of which are provided predefined learning, query and base set: For each query are provided the corresponding ground truth results in the base set, ordered from the most similar to the most different, computed in an exhaustive way with Euclidean distance. The SIFT query and base descriptors have been extracted from the INRIA Holidays images [23], while the learning set has been extracted from Flickr images. GIST query and base descriptors are from INRIA Holidays and Flickr 1M datasets, while learning vectors are from [24]. In all the cases query descriptors are from the query images of INRIA Holidays. In this work we have used the SIFT1M and GIST1M sub sets.

DEEP1M Dataset [9] is a recent dataset produced using a deep CNN based on the AlexNet [25] architecture and trained on ImageNet dataset [26]. Descriptors are extracted from the outputs of the last fully-connected layer, and to reduce its very high dimensionality they have been compressed to 256 dimensions using PCA, then they have been l_2-normalized.

The characteristics of all the datasets used in the experiments are summarized in Table 1.

Evaluation Metrics

The performance of ANN retrieval in BIGANN dataset is typically evaluated using *recall@R*, as shown in the results reported in the literature [1, 4, 5, 7, 10, 21]. It is defined, for varying values of R, as the average rate of queries for which the 1-nearest neighbor is retrieved in the top R positions. In case of $R = 1$ this metric coincides with *precision@1*. The same measure has been used by the authors of the DEEP1M dataset [9]. In the following, all the results reported use *recall@R* to allow comparison with the other approaches.

Configurations and Implementations

BIGANN: A typical hash length used in the other approaches is 64 bits. We follow this choice and use settings which reproduce top performances at 64-bit codes. We perform search with a non-exhaustive approach. For each query 64 bits binary hash code of the feature and Hamming distance measure are used to extract small subsets of candidates from the whole database set (Table 1). Euclidean distance measure is then used to re-rank the nearest feature points, calculating recall@R values in these subsets, an approach used also in [7,10,20].

DEEP1M: We use the CNN features computed in [9] obtained from a convolutional neural network with an L_2 normalization and a PCA compression for a final dimension of D= 256 and finally hashed to 32-bit codes. Searching process is done in a non-exhaustive way, using Hamming distances to reduce the subsets of candidates from the whole database set. After we have extracted a shortlist of candidates we perform a re-rank step based on Euclidean distances and we calculate recall@R values, as in [11].

In all cases, both hashed codes and full descriptors have been stored in main memory.

4.1. Results on BIGANN: SIFT1M, GIST1M

In these experiments the proposed approach is evaluated using the SIFT1M (Table 4) and GIST1M (Table 5) datasets, comparing it against several methods presented in section 2: Product Quantization (ADC and IVFADC) [1], PQ-RO [8], PQ-RR [8], Cartesian k-means [4], OPQ-P [5, 27], OPQ-NP [5, 27], LOPQ [7], a non-exhaustive adaptation of OPQ [5], called I-OPQ [7], RVQ [28] , RVQ-P [8] and RVQ-NP [8], m-k-means-t [20] using as threshold the arithmetic mean of the distances between feature vectors and centroids to compute hash code.

Since we have some randomness due to the k-means clustering in the first and third step of the proposed approach, these experiments are averaged over a set of 10 runs.

The proposed method, in all its variants, obtains the best results when considering the more challenging values of *recall@R*, i.e. with a small number of nearest neighbors, like

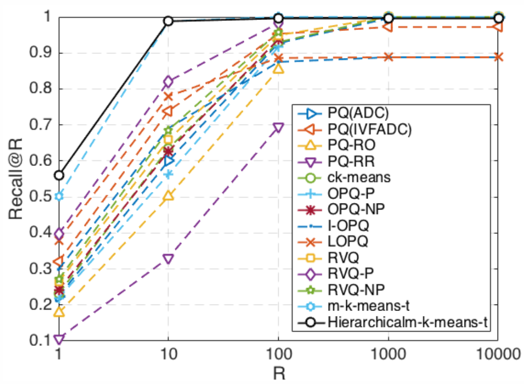

Fig. 4. *Recall@R* on SIFT1M - Comparison between our method with competing state-of-the-art methods (see references).

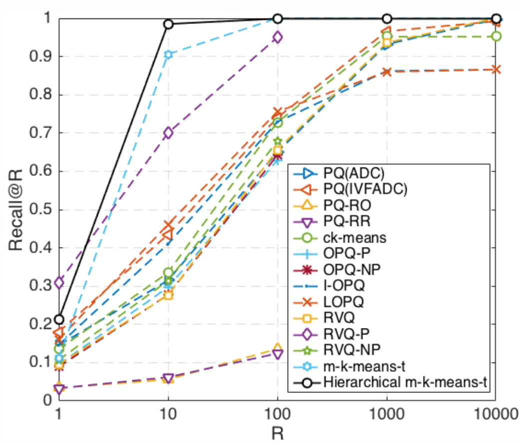

Fig. 5. *Recall@R* on GIST1M - Comparison between our method with competing state-of-the-art methods (see references).

1, 10 and 100. When R goes to 1000 and 10,000 it still obtains the best results and in the case of SIFT1M it is on par with ck-means [4]. Considering GIST1M the method consistently outperforms all the other methods for all the values of R, except for R=1 where RVQ-P [8] is better.

In general the use of the hierarchical approach outperforms the m-k-means-t method of [20], especially in the challenging low values of R; this effect is more visible in the case of the GIST 1M dataset.

4.2. Results on DEEP1M

Experiments on DEEP1M [9] are shown in Figure 6. We use a configuration with a final hash code length of 32 bits. Since the dataset is much newer than BIGANN, only a few methods have been tested on it, that we report. The proposed method is compared against PQ [21], OTQ [9], AQ [29] and OPQ [5] for which we report the results obtained by the authors using hash codes of 32 bits. Following the experimental setup used in [9], we considered $R = 1$, $R = 10$, $R = 10$ and $r = 1000$ for the *recall@R* measure.

The proposed method obtains similar result for *recall@1* respect to the others approach; considering $R = 10, R = 100$ and $R = 1000$ it obtains a results considerably better than the others methods; Especially we obtain better results than m-k-means-t method [20] on all R values.

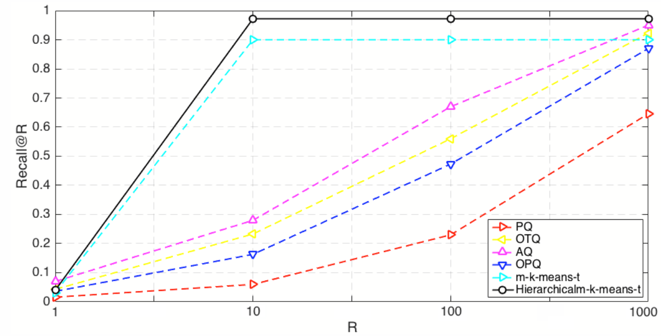

Fig. 6. *Recall@R* on DEEP1M - Comparison between our method with competing state-of-the-art methods (see references).

5. CONCLUSION

We have proposed a new version of the k-means based hashing schema called hierarchical multi-k-means which uses a small number of centroids, has a low computational cost and results in a compact quantizer. These characteristics make it a good choice for large-scale multimedia applications. The hash signature computed with our proposed approach is able to represent high dimensional visual features more than the previous approach called multi-k-means [20] obtaining a very high efficiency in approximate nearest neighbor (ANN) retrieval, both on local and global visual features. The method has been also tested on large scale datasets of engineered (SIFT and GIST) and learned (deep CNN) features, obtaining results that outperform or are comparable to more complex state-of-the-art approaches.

Acknowledgment This work is partially supported by the "Social Museum and Smart Tourism" project (CTN01_00034_231545).

This research is based upon work supported in part by the Office of the Director of National Intelligence (ODNI), Intelligence Advanced Research Projects Activity (IARPA), via IARPA contract number 2014-14071600011. The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of ODNI, IARPA, or the U.S. Government. The U.S. Government is authorized to reproduce and distribute reprints for Governmental purpose notwithstanding any copyright annotation thereon.

6. REFERENCES

[1] H. Jégou, M. Douze, and C. Schmid, "Product quantization for nearest neighbor search," *IEEE Transactions*

on Pattern Analysis and Machine Intelligence, vol. 33, no. 1, pp. 117–128, 2011.

[2] Y. Weiss, A. Torralba, and R. Fergus, "Spectral hashing," in *Proc. of NIPS*, 2009.

[3] M. Jain, H. Jégou, and P. Gros, "Asymmetric Hamming embedding: Taking the best of our bits for large scale image search," in *Proc. of ACM MM*, 2011.

[4] M. Norouzi and D.J. Fleet, "Cartesian k-means," in *Proc. of CVPR*, 2013.

[5] T. Ge, K. He, Q. Ke, and J. Sun, "Optimized product quantization for approximate nearest neighbor search," in *Proc. of CVPR*, 2013.

[6] K. He, F. Wen, and J. Sun, "K-means hashing: An affinity-preserving quantization method for learning binary compact codes," in *Proc. of CVPR*, 2013.

[7] Y. Kalantidis and Y. Avrithis, "Locally optimized product quantization for approximate nearest neighbor search," in *Proc. of CVPR*, 2014.

[8] D. Guo, C. Li, and L. Wu, "Parametric and nonparametric residual vector quantization optimizations for ANN search," *Neurocomputing*, 2016.

[9] A. Babenko and V. Lempitsky, "Tree quantization for large-scale similarity search and classification," in *Proc. of CVPR*, 2015.

[10] A. Babenko and V. Lempitsky, "The inverted multi-index," in *Proc. of CVPR*, 2012.

[11] A. Babenko and V. Lempitsky, "Efficient indexing of billion-scale datasets of deep descriptors," in *Proc. of CVPR*, 2016.

[12] K. Lin, H.-F. Yang, J.-H. Hsiao, and C.-S. Chen, "Deep learning of binary hash codes for fast image retrieval," in *Proc. of CVPR*, 2015.

[13] R. Xia, Y. Pan, H. Lai, C. Liu, and S. Yan, "Supervised hashing for image retrieval via image representation learning.," in *Proc. of AAAI*, 2014.

[14] T.-T. Do, A.-Z. Doan, and N.-M. Cheung, "Discrete hashing with deep neural network," *arXiv preprint arXiv:1508.07148*, 2015.

[15] J. Guo and J. Li, "CNN based hashing for image retrieval," *arXiv preprint arXiv:1509.01354*, 2015.

[16] Z. Zhang, Y. Chen, and V. Saligrama, "Supervised hashing with deep neural networks," *arXiv preprint arXiv:1511.04524*, 2015.

[17] S. Boyd, N. Parikh, E. Chu, B. Peleato, and J. Eckstein, "Distributed optimization and statistical learning via the alternating direction method of multipliers," *Foundations and Trends in Machine Learning*, vol. 3, no. 1, pp. 1–122, 2011.

[18] D. Wang, P. Cui, M. Ou, and W. Zhu, "Learning compact hash codes for multimodal representations using orthogonal deep structure," *IEEE Transactions on Multimedia*, vol. 17, no. 9, pp. 1404–1416, 2015.

[19] J. Lin, O. Morère, J. Petta, V. Chandrasekhar, and A. Veillard, "Tiny descriptors for image retrieval with unsupervised triplet hashing," in *Proc. of DCC*, 2016.

[20] S. Ercoli, M. Bertini, and A. Del Bimbo, "Compact hash codes and data structures for efficient mobile visual search," in *Proc. of International Workshop on Mobile Multimedia Computing (MMC)*, 2015.

[21] H. Jégou, R. Tavenard, M. Douze, and L. Amsaleg, "Searching in one billion vectors: re-rank with source coding," in *Proc. of ICASSP*, 2011.

[22] M. Norouzi, A. Punjani, and D.J. Fleet, "Fast exact search in Hamming space with multi-index hashing," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 36, no. 6, pp. 1107–1119, 2014.

[23] H. Jegou, M. Douze, and C. Schmid, "Hamming embedding and weak geometric consistency for large scale image search," in *Proc. of ECCV*, 2008.

[24] A. Torralba, R. Fergus, and W. T. Freeman, "80 million tiny images: A large data set for nonparametric object and scene recognition," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 30, no. 11, pp. 1958–1970, 2008.

[25] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, "Going deeper with convolutions," in *Proc. of CVPR*, 2015.

[26] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "ImageNet: A large-scale hierarchical image database," in *Proc. of CVPR*, 2009.

[27] T. Ge, K. He, Q. Ke, and J. Sun, "Optimized product quantization," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 36, no. 4, pp. 744–755, 2014.

[28] Y. Chen, T. Guan, and C. Wang, "Approximate nearest neighbor search by residual vector quantization," *Sensors*, vol. 10, no. 12, pp. 11259–11273, 2010.

[29] A. Babenko and V. Lempitsky, "Additive quantization for extreme vector compression," in *Proc. of CVPR*, 2014.