Compact Hash Codes for Efficient Visual Descriptors Retrieval in Large Scale Databases

Simone Ercoli, Marco Bertini, *Member, IEEE*, and Alberto Del Bimbo, *Member, IEEE*

***Abstract*—In this paper, we present an efficient method for visual descriptors retrieval based on compact hash codes computed using a multiple k-means assignment. The method has been applied to the problem of approximate nearest neighbor (ANN) search of local and global visual content descriptors, and it has been tested on different datasets: three large scale standard datasets of engineered features of up to one billion descriptors (BIGANN) and, supported by recent progress in convolutional neural networks (CNNs), on CIFAR-10, MNIST, INRIA Holidays, Oxford 5K, and Paris 6K datasets; also, the recent DEEP1B dataset, composed by one billion CNN-based features, has been used. Experimental results show that, despite its simplicity, the proposed method obtains a very high performance that makes it superior to more complex state-of-the-art methods.**

***Index Terms*—Convolutional neural network (CNN), hashing, nearest neighbor search, retrieval, SIFT.**

I. Introduction

EFFICIENT nearest neighbor (NN) search is one of the main issues in large scale information retrieval for multimedia and computer vision tasks. Also methods designed for multidimensional indexing obtain a performance that is only comparable to exhaustive search, when they are used to index high dimensional features [1]. A typical solution to avoid an exhaustive comparison is to employ methods that perform approximate nearest neighbor (ANN) search, that is finding the elements of a database that have a high probability to be nearest neighbors. ANN algorithms have been typically evaluated based on the trade-off between efficiency and search quality, but the inception of web-scale datasets have introduced also the problems of memory usage and speed. For these reasons the most recent approaches typically use feature hashing to reduce the dimensionality of the descriptors, and compute Hamming distances over these hash codes. These methods generally use inverted files, e.g. implemented using hash tables, to index the database, and require to use hash codes with a length of several tens of bits to obtain a reasonable performance in retrieval, a typical length is 64 bits. Their application to systems with relatively limited memory (e.g. mobile devices still have 1–2 GB RAM only) or to systems that involve a large-scale media indexing, that need to maintain the index in main memory to provide an adequate response time, requires to use compact hash codes.

In this paper we present a novel method for feature hashing, based on multiple k-means assignments. This method is unsupervised, requires a very limited codebook size and obtains a very good performance in retrieval, even with very compact hash codes (e.g. also with 32 bits); it can be applied to hash local and global visual features, either engineered (e.g. SIFT) or learned (e.g. CNNs). The proposed approach greatly reduces the need of training data and memory requirements for the quantizer, and obtains a retrieval performance similar or superior to more complex state-of-the-art approaches on standard large scale datasets. This makes it suitable, in terms of computational cost, for mobile devices, large-scale media analysis and content-based image retrieval in general.

The paper is organized as follows: in Section II we provide a thorough review of works related to visual feature hashing and indexing, highlighting the main differences of the proposed method. Our approach is presented in Section III, including a discussion about its computational complexity. Experimental results on several large scale standard datasets of visual features and images, and comparison with the current state-of-the-art, are reported and discussed in Section IV. Finally, conclusions are drawn in Section V.

II. Previous Works

Previous works on visual feature hashing can be divided in methods based on hashing functions, scalar quantization, vector quantization and, more recently, neural networks.

A. Hashing Functions

Weiss *et al.* [2] have proposed to treat the problem of hashing as a particular form of graph partitioning, in their Spectral Hashing (SH) algorithm. Li *et al.* [3] have improved the application of SH to image retrieval optimizing the graph Laplacian that is built based on pairwise similarities of images during the hash function learning process, without requiring to learn a distance metric in a separate step. Heo *et al.* [4] have proposed to encode high-dimensional data points using hyperspheres instead of hyperplanes; Jin *et al.* [5] have proposed a variation of LSH, called Density Sensitive Hashing, that does

Manuscript received August 26, 2016; revised December 23, 2016; accepted March 21, 2017. Date of publication April 25, 2017; date of current version October 13, 2017. This work was supported in part by the "Social Museum and Smart Tourism" under Project CTN01_00034_231545, in part by the Office of the Director of National Intelligence, and in part by the Intelligence Advanced Research Projects Activity, via IARPA under Contract 2014-14071600011. The associate editor coordinating the review of this manuscript and approving it for publication was Prof. Zhu Li. *(Corresponding author: Marco Bertini.)*

The authors are with the Media Integration and Communication Center, University of Florence, Florence 50139, Italy (e-mail: simone.ercoli@unifi.it; marco.bertini@unifi.it; alberto.delbimbo@unifi.it).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TMM.2017.2697824

not use random projections but instead uses projective functions that are more suitable for the distribution of the data. Du *et al.* [6] have proposed the use of Random Forests to perform linear projections, along with a metric that is not based on Hamming distance. Lv *et al.* [7] address the problem of large scale image retrieval learning two hashes in their Asymmetric Cyclical Hashing (ACH) method: a short one (k bits) for the images of the database and a longer one (mk bits) for the query. Hashes are obtained using similarity preserving Random Fourier Features, and computing the Hamming distance between the long query hash and the cyclically m-times concatenated compact hash code of the stored image.

Paulevé *et al.* [8] have compared structured quantization algorithms with unstructured quantizers (i.e. k-means and hierarchical k-means clustering). Experimental results on SIFT descriptors have shown that unstructured quantizers provide significantly superior performances with respect to structured quantizers.

B. Scalar Quantization

Zhou *et al.* [9] have proposed an approach based on scalar quantization of SIFT descriptors. The median and the third quartile of the bins of each descriptor are computed and used as thresholds, hashing is then computed coding the value of each bin of the descriptor with 2 bits, depending on this subdivision. The final hash code has a dimension of 256 bits, but only the first 32 bits are used to index the code in an inverted file; thus differences in the following bits, associated with the remaining bins of the SIFT descriptor, are not taken into account when querying the index. On the other hand methods that follow this approach, like the three following ones, do not require training data. The method of [9] has been extended by Ren *et al.* [10], including an evaluation of the reliability of bits, depending on their quantization errors. Unreliable bits are then flipped when performing search, as a form of query expansion. To avoid using codebooks, in the context of memory limited devices such as mobile phones, Zhou *et al.* [11] have proposed the use of scalable cascaded hashing (SCH), performing sequentially scalar quantization on the principal components, obtained using PCA, of SIFT descriptors. Chen and Hsieh [12] have recently proposed an approach that quantizes the differences of the bins of the SIFT descriptor, using the median computed on all the SIFT descriptors of a training set as a threshold.

C. Vector Quantization

Jégou *et al.* [1] have proposed to decompose the feature space into a Cartesian product of subspaces with lower dimensionality, that are quantized separately. This Product Quantization (PQ) method is efficient in solving memory issues that arise when using vector quantization methods such as k-means, since it requires a much reduced number of centroids to obtain a code of the desired length. The method has obtained state-of-the-art results on a large scale SIFT features dataset, improving over methods such as SH [2] and Hamming Embedding [13]. This result is confirmed in the work of Chandrasekhar *et al.* [14], that have compared several compression schemes for SIFT features. However, its performance is dependent on the choice of the subspaces used.

The efficiency and efficacy of the Product Quantization method has led to development of several variations and improvements. The idea of compositionality of the PQ approach has been further analyzed by Norouzi and Fleet [15], that have built upon it proposing two variations of k-means: Orthogonal k-means and Cartesian k-means (ck-means). Also Ge *et al.* [16] have proposed another improvement of PQ, called OPQ, that minimizes quantization distortions w.r.t. space decomposition and quantization codebooks; He *et al.* [17] have proposed an affinity-preserving technique to approximate the Euclidean distance between codewords in k-means method. Kalantidis and Avrithis [18] have presented a simple vector quantizer (LOPQ) which uses a local optimization over a rotation and a space decomposition and apply a parametric solution that assumes a normal distribution. More recently, Guo *et al.* [19] have improved over OPQ and LOPQ adding two quantization distortion properties of the Residual Vector Quantization (RVQ) model, that tries to restore quantization distortion errors instead of reducing it.

A few works have addressed the problem of indexing. Babenko and Lempitsky [20] have proposed an efficient similarity search method, called inverted multi-index (IMI); this approach generalizes the inverted index by replacing vector quantization inside inverted indices with product quantization, and building the multi-index as a multi-dimensional table (Multi-D-ADC). Another multi-index strategy has been proposed by Zheng *et al.* [21], where complementary features, i.e. binarized SIFT descriptors and local color features are indexed in a coupled Multi-Index (c-MI), performing feature fusion at indexing level for content-based image retrieval. More recently, Babenko and Lempitsky [22] have addressed the problem of indexing CNN features, observing that IMI is inefficient to index such features, and proposing two extensions of IMI: the Non-Orthogonal Inverted Multi-Index (NO-IMI) and the Generalized Non-Orthogonal Inverted Multi-Index (GNO-IMI). Multi-scale matching of SIFT and CNN features binarized with LSH, has been recently addressed in the work of Zheng *et al.* [23], proposing an indexing structure that uses compact pointers to associate the local features with the regional and global ones.

D. Neural Networks

Lin *et al.* [24] have proposed a deep learning framework to create hash-like binary codes for fast image retrieval. Hash codes are learned in a point-wise manner by employing a hidden layer for representing the latent concepts that dominate the class labels (when the data labels are available). This layer learns specific image representations and a set of hash-like functions.

Do *et al.* [25] have addressed the problem of learning binary hash codes for large scale image search using a deep model which tries to preserve similarity, balance and independence of images. Two sub-optimizations during the learning process allow to efficiently solve binary constraints.

Guo and Li [26] have proposed a method to obtain the binary hash code of a given image using binarization of the CNN outputs of a certain fully connected layer.

TABLE I
NOTATION TABLE

x_i	feature to be hashed
C_i	generic centroid
S_i	cluster
k	number of centroids; length of the hash code
C_j	centroid associated to the j^{th} bit of the hash code
D	dimension of the feature

Zhang *et al.* [27] have proposed a very deep neural network (DNN) model for supervised learning of hash codes (VDSH). They use a training algorithm inspired by alternating direction method of multipliers (ADMM) [28]. The method decomposes the training process into independent layer-wise local updates through auxiliary variables.

Xia *et al.* [29] have proposed an hashing method for image retrieval which simultaneously learns a representation of images and a set of hash functions.

A deep learning framework for hashing of multimodal data has been proposed by Wang *et al.* [30], using a multimodal Deep Belief Network to capture correlation in high-level space during pre-training, followed by learning a cross-modal autoencoder in fine tuning phase.

Lin *et al.* [31] have proposed to use unsupervised two steps hashing of CNN features. In the first step Stacked Restricted Boltzmann Machines learn binary embedding functions, then fine tuning is performed to retain the metric properties of the original feature space.

The method proposed in this paper belongs to the family of methods based on vector quantization; compared to recent hashing algorithms based on neural networks this allows to learn hash codes with a much reduced computational cost, without loss in performance. Unlike previous vector quantization approaches it associates the features to multiple codebook words, reducing quantization errors due to wrong associations to nearby codewords. Another difference, considering in particular the family of methods based on Product Quantization, is the fact that codewords are associated to single bits of the hash and not to portions of the feature; this avoids the need to partition features so to reflect the structure of the descriptor, as shown in [1].

III. THE PROPOSED METHOD

The proposed method exploits a novel version of the k-means vector quantization approach, introducing the possibility of assignment of a visual feature to multiple cluster centers during the quantization process. This approach greatly reduces the number of required cluster centers, as well as the required training data, performing a sort of quantized codebook soft assignment for an extremely compact hash code of visual features. Table I summarizes the symbols used in the following.

The first step of the computation is a typical k-means algorithm for clustering. Given a set of observations $(\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n)$ where each observation is a D-dimensional real vector, k-means clustering partitions the n observations into $k(\leq n)$ sets $\mathbf{S} = \{S_1, S_2, \ldots, S_k\}$ so as to minimize the sum of distance functions of each point in the cluster to the C_k centers. Its objective is to find

$$\text{argmin}_{\mathbf{s}} \sum_{i=1}^{k} \sum_{\mathbf{x} \in S_i} \| \mathbf{x} - C_i \|^2 . \quad (1)$$

This process is convergent (to some local optimum) but the quality of the local optimum strongly depends on the initial assignment. We use the *k-means++* [32] algorithm for choosing the initial values, to avoid the poor clusterings sometimes found by the standard k-means algorithm.

A. Multi-k-means Hashing

K-means is typically used to compute the hash code of visual feature in unstructured vector quantization, because it minimizes the quantization error by satisfying the two Lloyd optimality conditions [1]. In a first step a dictionary is learned over a training set and then hash codes of features are obtained by computing their distance from each cluster center. Vectors are assigned to the nearest cluster center, whose code is used as hash code. Considering the case of 128-dimensional visual content descriptors, like SIFT or the FC7 layer of the VGG-M-128 CNN [33], this means that compressing them to 64 bits codes requires to use $k = 2^{64}$ centroids. In this case the computational cost of learning a k-means based quantizer becomes expensive in terms of memory and time because: *i)* there is the need of a quantity of training data that is several times larger than k, and *ii)* the execution time of the algorithm becomes unfeasible. Using hierarchical k-means (HKM) makes it possible to reduce execution time, but the problem of memory usage and size of the required learning set affects also this approach. Since the quantizer is defined by the k centroids, the use of quantizers with a large number of centroids may not be practical or efficient: if a feature has a dimension D, there is need to store $k \times D$ values to represent the codebook of the quantizer. A possible solution to this problem is to reduce the length of the hash signature, but this typically affects negatively retrieval performance. The use of product k-means quantization, proposed originally by Jégou *et al.* [1], overcomes this issue.

In our approach, instead, we propose to compute a sort of soft assignment within the k-means framework, to obtain very compact signatures and dimension of the quantizer, thus reducing its memory requirements, while maintaining a retrieval performance similar to that of [1].

The proposed method, called *multi-k-means* (in the following abbreviated as *m-k-means*), starts learning a standard k-means dictionary as shown in (1), using a very small number k of centroids to maintain a low computational cost. Once we obtained our $\mathbf{C}_1, \ldots, \mathbf{C}_k$ centroids, the main difference resides in the assignment and creation of the hash code. Each centroid is associated to a specific bit of the hash code

$$\begin{cases} \| x - C_j \| \leq \delta & j^{th}\ bit = 1 \\ \| x - C_j \| > \delta & j^{th}\ bit = 0 \end{cases} \quad (2)$$

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

$$\delta = \begin{cases} (\prod_{j=1}^{k} \parallel x - C_j \parallel)^{\frac{1}{k}} & \text{geometric mean} \\ \frac{1}{k} \sum_{j=1}^{k} \parallel x - C_j \parallel & \text{arithmetic mean} \\ n^{th} \text{ nearest distance } \parallel x - C_j \parallel & \forall j = 1, ..., k \end{cases} \tag{3}$$

i.e. centroid j is associated to the j^{th} bit of the hash code of length k; the bit is set to 1 if the feature to be quantized is assigned to its centroid, or to 0 otherwise.

A feature can be assigned to more than one centroid using two main different approaches:

i) m-k-means-t_1 - using (2) and one of the first two thresholds of (3). In this case the feature vector is considered as belonging to all the centroids from which its distance is below the threshold. Experiments have shown that the arithmetic mean is more efficient with respect to the geometric one, and all the experiments will report results obtained with it.

ii) m-k-means-n_1 - using (2) and the third threshold of (3), i.e. assigning the feature to a predefined number n of nearest centroids.

We also introduce two variants (*m-k-means-t_2* and *m-k-means-n_2*) to the previous approaches by randomly splitting the training data into two groups and creating two different codebooks for each feature vector. The final hash code is given by the union of these two codes.

With the proposed approach it is possible to create hash signatures using a much smaller number of centroids than using the usual k-means baseline, since each centroid is directly associated to a bit of the hash code. This approach can be considered a quantized version of codebook soft assignment [34] and, similarly, it alleviates the problem of codeword ambiguity while reducing the quantization error.

Fig. 1 illustrates the quantization process and the resulting hash codes in three cases: one in which a vector is assigned to a variable number of centroids (*m-k-means-t_1*), one in which a vector is assigned to a predefined number of centroids (*m-k-means-n_1*) and one in which the resulting code is created by the union of two different codes created using two different codebooks (*m-k-means-t_2* and *m-k-means-n_2*). In all cases the feature is assigned to more than one centroid. An evaluation of these two approaches is reported in Section IV.

Typically a multi probe approach is used to solve the problem of ambiguous assignment to a codebook centroid (in case of vector quantization, as in the coarse quantization step of PQ [1]) or quantization error (e.g. in case of scalar quantization, as in [9], [10]); this technique stems from the approach originally proposed in [35], to reduce the need of creating a large number of hash tables in LSH. The idea is that if a object is close to a query object q, but is not hashed to the same bucket of q, it is still likely hashed to a bucket that is near, i.e. to a bucket associated with an hash that has a small difference w.r.t. the hash of q. With this approach one or more bits of the query hash code are flipped to perform a query expansion, improving recall at the expense of computational cost and search time. In fact, if we chose to try all the hashes within an Hamming distance of 1 we have to create variations of the original hash of q flipping all the bits of the hash, one at a time. This means that for a hash code of length k we need to repeat the query with additional k hashes. In the proposed method this need of multi probe queries is greatly reduced, because of the possibility of assignment of features to more than one centroid. For example, consider either Fig. 1 *(top)* or *(middle)*: if a query point nearby f_1 or f_2 falls in the Voronoi cell of centroid C_6, using standard k-means it could be retrieved only using a multi probe query, instead the proposed approach maintains the same hash code.

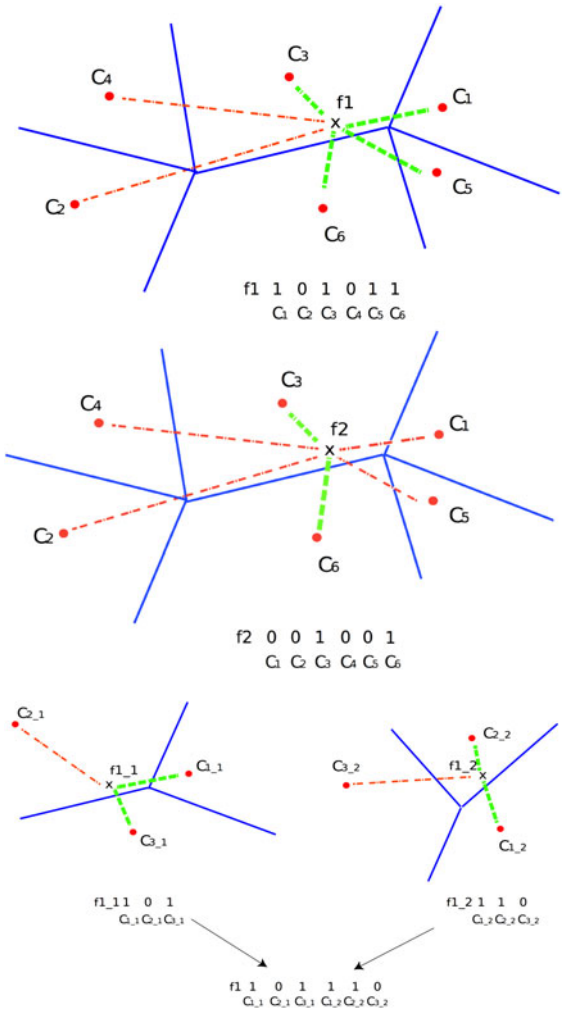

Fig. 1. Toy examples illustrating the proposed method: (top) features can be assigned (green line) to a variable number of nearest clusters (e.g. those with distances below the mean δ, i.e., *m-k-means-t_1*); (middle) features can be assigned to a fixed number of clusters (e.g. the 2 nearest clusters, i.e., *m-k-means-n_1*); (bottom) hash code created from two different codebooks (*m-k-means-x_2*, where x can be either t or n). If a feature is assigned to a centroid, the corresponding bit in the hash code is set to 1.

B. Computational Complexity

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 $k = 2^{64}$, while the proposed approach instead needs $k' = 64$ centroids, has a complexity of $k'D$ and requires $k'D$ floats to store the codebook. Product Quantization

Fig. 2. Sample images from the INRIA Holiday dataset. Left column shows the query images, the other columns show similar images.

requires $k^* \times D$ floats for the codebook and has an assignment complexity of $k^* D$, where $k^* = k^{1/m}$, using typically $k^* = 256$ and $m = 8$ values, for a 64 bit length [1]; in this case the cost of the proposed method is a quarter of the cost of PQ.

IV. Experimental Results

The variants of the proposed method (*m-k-means-*t_1, *m-k-means-*n_1, *m-k-means-*t_2 and *m-k-means-*n_2) have been thoroughly compared to several state-of-the-art approaches using standard datasets, experimental setups and evaluation metrics.

A. Datasets

BIGANN Dataset [1], [36] is a large-scale dataset commonly used to compare methods for visual feature hashing and approximate nearest neighbor search [1], [15], [16], [18], [20], [36], [37]. The dataset is composed by three different sets of SIFT and GIST descriptors, each one divided in three subsets: a learning set, a query set and base set; each query has corresponding ground truth results in the base set, computed in an exhaustive way with Euclidean distance, ordered from the most similar to the most different. For SIFT1M and SIFT1B query and base descriptors have been extracted from the INRIA Holidays images [38], while the learning set has been extracted from Flickr images. For GIST1M query and base descriptors are from INRIA Holidays and Flickr 1M datasets, while learning vectors are from [39]. In all the cases query descriptors are from the query images of INRIA Holidays (see Fig. 2). The characteristics of the dataset are summarized in Table II.

DEEP1B Dataset [22] is a recent dataset produced using a deep CNN based on the GoogLeNet [40] architecture and trained on ImageNet dataset [41]. Descriptors are extracted from the outputs of the last fully-connected layer, compressed using PCA to 96 dimensions, and l_2-normalized. The characteristics of the dataset are summarized in Table III.

TABLE II
BIGANN Datasets Characteristics

vector dataset	SIFT 1M	SIFT 1B	GIST 1M
descriptor dimensionality D	128	128	960
# learning set vectors	100,000	100,000,000	500,000
# database set vectors	1,000,000	1,000,000,000	1,000,000
# queries set vectors	10,000	10,000	1,000
# nearest vectors for each query	100	1000	100

TABLE III
DEEP1B Datasets Characteristics

descriptor dimensionality D	96
# learning set vectors	358,480,000
# database set vectors	1,000,000,000
# queries set vectors	10,000
# nearest vectors for each query	1

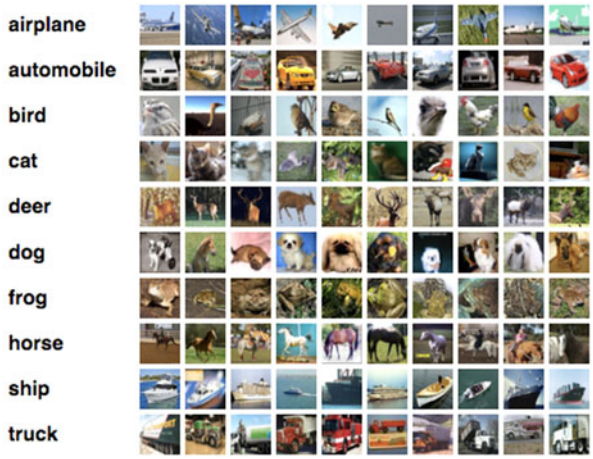

Fig. 3. Sample images from the CIFAR-10 dataset.

CIFAR-10 Dataset [42] consists of 60,000 colour images (32×32 pixels) in 10 classes, with 6,000 images per class (see Fig. 3). The dataset is split into training and test sets, composed by 50,000 and 10,000 images respectively. A retrieved image is considered relevant for the query if it belongs to the same class. This dataset has been used for ANN retrieval using hash codes in [24], [29].

MNIST Dataset [43] consists of 70,000 handwritten digits images (28×28 pixels, see Fig. 4). The dataset is split into 60,000 training examples and 10,000 test examples. Similarly to CIFAR-10, a retrieved image is considered relevant if it belongs to the same class of the query. This dataset has been used for ANN retrieval in [24], [29].

Image retrieval datasets INRIA Holidays [38], Oxford 5K [44] and Paris 6K [45] are three datasets typically used to evaluate image retrieval systems. For each dataset are given a number of query images, and the associated ground truth. INRIA Holidays is composed by 1,491 images, of which 500 are used as queries; Oxford 5K is composed by 5,062 images with 55 query images, and Paris 6 K is made of 6,412 images with 55 query images. We used the query images and ground truth provided

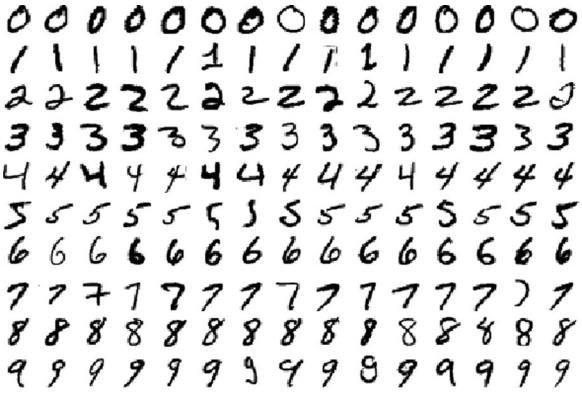

Fig. 4. Sample images from the MNIST dataset.

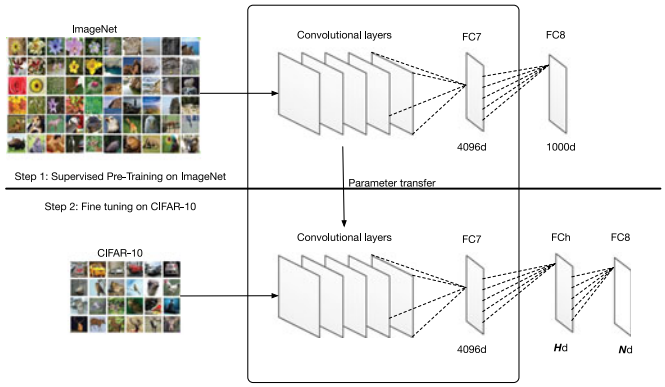

Fig. 5. Framework used for the CNN feature extraction on CIFAR-10 [24]: we use the values of the nodes of the FCh layer as feature (48 dimensions).

for each dataset, adding 100,000 distractor images from *Flickr 100K* [44].

B. Evaluation Metrics

The performance of ANN retrieval in BIGANN dataset is evaluated using *recall@R*, which is used in most of the results reported in the literature [1], [15], [16], [18], [20], [36] and it is, for varying values of R, 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 DEEP1B dataset [22].

Performance of image retrieval in CIFAR-10, MNIST, INRIA Holidays, Oxford 5K and Paris 6K is measured following the setup of [29], using Mean Average Precision

$$MAP = \frac{\sum_{q=1}^{Q} AveP(q)}{Q} \tag{4}$$

where

$$AveP = \int_0^1 p(r)dr \tag{5}$$

is the area under the precision-recall curve and Q is the number of queries.

C. Configurations and Implementations

1) BIGANN: We 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 II). Euclidean distance measure is then used to re-rank the nearest feature points, calculating recall@R values in these subsets.

2) DEEP1B: We use the CNN features computed in [22], hashed to 64-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.

3) CIFAR-10: We use features computed with the framework proposed in [24] (Fig. 5). The process is carried out in two steps: in the first step a supervised pre-training on the large-scale ImageNet dataset [46] is performed. In the second step fine-tuning of the network is performed, with the insertion of a latent layer that simultaneously learns domain specific feature representations and a set of hash-like functions. The authors used the pre-trained CNN model proposed by Krizhevsky *et al.* [46] and implemented in the Caffe CNN library [47]. In our experiments we use features coming from the FCh Layer (Latent Layer H), which has a size of 48 nodes.

4) MNIST: We use LeNet CNN to compute our features in MNIST. This is a network architecture developed by LeCun [48] that was especially designed for recognizing handwritten digits, reading zip codes, etc. It is a 8-layer network with 1 input layer, 2 convolutional layers, 2 non-linear down-sampling layers, 2 fully connected layers and a Gaussian connected layer with 10 output classes. We used a modified version of LeNet [47] and we obtain features from the first fully connected layer.

We perform search with a non-exhaustive approach on both CIFAR-10 and MNIST datasets. For each image we extract a 48-dimensional feature vector for CIFAR-10, and 500-dimensional feature vector for MNIST, from the respective network and then we generate a 48 bits binary hash code using the proposed methods of Section III. Hamming distance is used to select the nearest hash codes for each query and similarity measure given by

$$\text{similarity} = \cos(\theta) = \frac{\mathbf{A} \cdot \mathbf{B}}{\| \mathbf{A} \| \| \mathbf{B} \|} = \frac{\sum_{i=1}^{n} A_i B_i}{\sqrt{\sum_{i=1}^{n} A_i^2} \sqrt{\sum_{i=1}^{n} B_i^2}} \tag{6}$$

where A_i and B_i are the components of the original feature vectors A and B, is used to re-rank the nearest visual features.

i) Holidays, Oxford 5K, Paris 6K: We used CNN features extracted using the 1024d average pooling layer of GoogLeNet [40], that in initial experiments has proven to be more effective than the FC7 layer of VGG [49] used in [50]. When testing on a dataset, training is performed using the other two datasets. Features have been hashed to 64 bits binary codes, a length that has proved to be the best compromise between compactness and

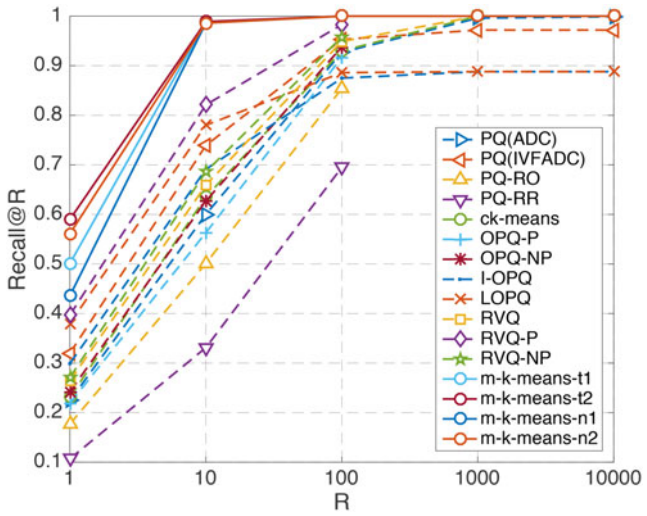

Fig. 6. *Recall@R* on SIFT1M. Comparison between our method (m-k-means-t_1, m-k-means-n_1 with $n = 32$, m-k-means-t_2 and m-k-means-n_2 with $n = 32$), the Product Quantization method (PQ ADC and PQ IVFADC) [1], Cartesian k-means method (ck-means) [15], a non-exhaustive adaptation of the optimized product quantization method (I-OPQ), a locally optimized product quantization method (LOPQ) [18], OPQ-P, and OPQ-NP [16], [51], and PQ-RO, PQ-RR, RVQ-P, and RVQ-NP [19].

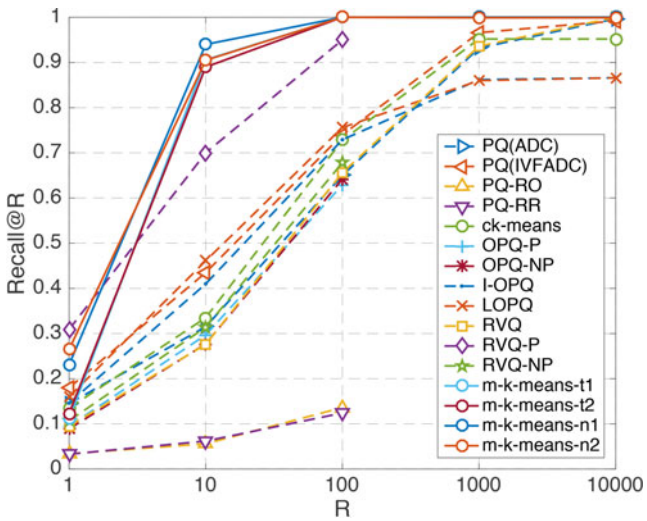

Fig. 7. *Recall@R* on GIST1M. Comparison between our method (m-k-means-t_1, m-k-means-n_1 with $n = 48$, m-k-means-t_2 and m-k-means-n_2 with $n = 48$), the product quantization method (ADC and IVFADC) [1], Cartesian k-means method (ck-means) [15], a non-exhaustive adaptation of the optimized product quantization method (I-OPQ) [18], a locally optimized product quantization method (LOPQ) [18], OPQ-P, and OPQ-NP [16], [51], and PQ-RO, PQ-RR, RVQ-P, and RVQ-NP [19].

representativeness, and allow us to compare with a number of competing approaches.

D. Results on BIGANN: SIFT1M, GIST1M

In this set of experiments the proposed approach and its variants are compared on the SIFT1M (Fig. 6) and GIST1M (Fig. 7) datasets against several methods presented in Section II: Product Quantization (ADC and IVFADC) [1], PQ-RO [19], PQ-RR [19], Cartesian k-means [15], OPQ-P [16], [51], OPQ-NP [16], [51], LOPQ [18], a non-exhaustive adaptation of OPQ [16], called I-OPQ [18], RVQ [52] , RVQ-P [19] and RVQ-NP [19].

ADC (*Asymmetric Distance Computation*) is characterized by the number of sub vectors m and the number of quantizers per sub vectors k^*, and produces a code of length $m \times log_2 k^*$.

IVFADC (*Inverted File with Asymmetric Distance Computation*) is characterized by the codebook size k' (number of centroids associated to each quantizer), the number of neighbouring cells w visited during the multiple assignment, the number of sub vectors m and the number of quantizers per sub vectors k^* which is in this case fixed to $k^* = 256$. The length of the final code is given by $m \times log_2 k^*$.

PQ-RO [19] is the Product Quantization approach with data projection by randomly order dimensions.

PQ-RR [19] is the Product Quantization approach with data projection by both PCA and randomly rotation.

Cartesian k-means (ck-means) [15] models region center as an additive combinations of subcenters. Let m be the number of subcenters, with h elements, then the total number of model centers is $k = h^m$, but the total number of subcenters is $h \times m$, and the number of bits of the signature is $m \times log_2 h$.

OPQ-P [16], [51] is the parametric version of Optimized Product Quantization (OPQ), that assumes a parametric Gaussian distribution of features and performs space decomposition using an orthonormal matrix computed from the covariance matrix of data.

OPQ-NP [16], [51] is the non-parametric version of OPQ, that does not assume any data distribution and alternatively optimizes sub-codebooks and space decomposition.

LOPQ (Locally optimized product quantization) [18] is a vector quantizer that combines low distortion with fast search applying a local optimization over rotation and space decomposition.

I-OPQ [18] is a non-exhaustive adaptation of OPQ (Optimized Product Quantization [16]) which use either OPQ-P or OPQ-NP global optimization.

RVQ [52] approximates the quantization error by another quantizer instead of discarding it. In this method several stage-quantizers, each one with its corresponding stage-codebook, are connected sequentially. Each stage-quantizer approximates the residual vector of the preceding stage by one of centroids of its stage-codebook and generates a new residual vector for the next stage.

RVQ-P [19] is a parametric version of RVQ, where stage-codebooks and space decomposition of RVQ are optimized using SVD.

RVQ-NP [19] is a non-parametric version of RVQ, using the same techniques of RVQ-P, but optimizing a space decomposition for all the stages.

The parameters of the proposed methods are set as follows: for *m-k-means-*t_1 we use as threshold the arithmetic mean of the distances between feature vectors and centroids to compute hash code; *m-k-means-*n_1 creates hash code by setting to 1 the corresponding position of the first 32 (SIFT1M) and first 48 (GIST1M) nearest centroids for each feature; *m-k-means-*t_2 and *m-k-means-*n_2 create two different sub hash codes for each feature by splitting into two parts the training phase and

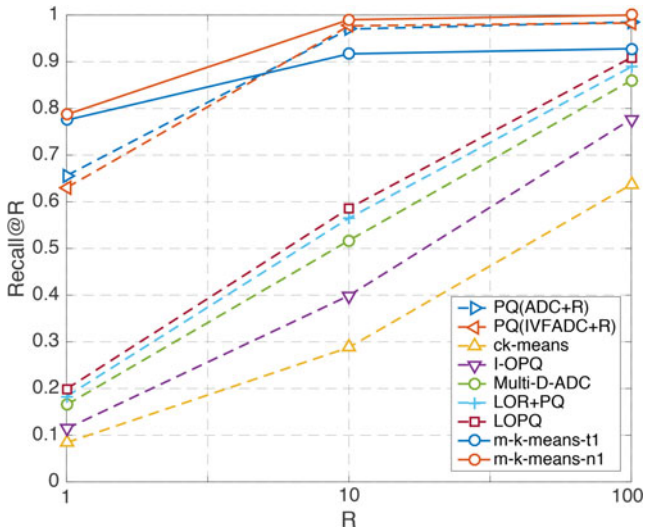

Fig. 8. *Recall@R* on SIFT1B. Comparison between our method (m-k-means-t_1, m-k-means-n_1 with $n = 24$), the product quantization method [36], a non-exhaustive adaptation of the optimized product quantization method (I-OPQ) [18], a multi-index method (Multi-D-ADC), and a locally optimized product quantization method (LOPQ) with a sub-optimal variant (LOR+PQ) [18].

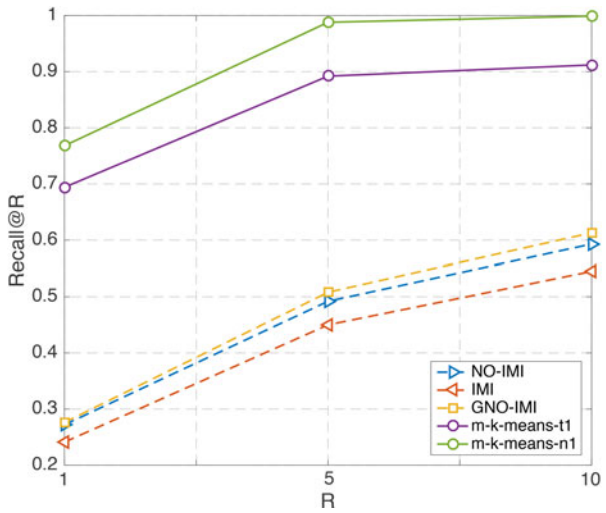

Fig. 9. *Recall@R* on DEEP1B. Comparison between our method (m-k-means-t_1, m-k-means-n_1 with $n = 24$), inverted multi-index (IMI) [22], non-orthogonal inverted multi-index (NO-IMI) [22] and generalized non-orthogonal inverted multi-index (GNO-IMI) [22].

combine these two sub parts into one single code to create the final signature. Since we have a random splitting during the training phase, 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 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 [15]. Considering GIST1M the method consistently outperforms all the other methods for all the values of R, except for $R = 1$ where RVQ-P [19] is better.

E. Results on BIGANN: SIFT1B

In this experiment we compare our method on the large scale SIFT1B dataset (Fig. 8) against LOPQ and a sub-optimal variant LOR+PQ [18], single index PQ approaches IVFADC+R [36] and ADC+R [36], I-OPQ [16] and ck-means [15], and a multi-index method Multi-D-ADC [20]. [36] differ from the standard IVFADC [1] and ADC [1] in using short quantization codes to re-rank the NN candidates. *m-k-means-*t_1 uses the same setup of the previous experiment; *m-k-means-*n_1 uses the first 24 nearest centroids for each feature.

Also in this experiment the proposed method obtains the best results, in particular when considering the more challenging small value of R for the *recall@R* measure ($R = 1$), with an improvement between 10% and 20% with respect to the best results of the compared methods.

F. Results on DEEP1B

Experiments on DEEP1B [22] are shown in Fig. 9. We use a configuration with an hash code length of 64 bits for the *m-k-means-*t_1 and *m-k-means-*n_1 variants. The comparison is made against IMI [22], NO-IMI [22] and GNO-IMI [22], for which we report the results obtained by the authors using a rerank approach for codes of 64 bits. Following the experimental setup used in [22], we considered $R = 1$, $R = 5$ and $R = 10$ for the *recall@R* measure.

The proposed method obtains best results in both configurations (*m-k-means-*t_1 and *m-k-means-*n_1) and considering $R = 1$ obtains a result approximately three times greater than the others methods; for the other values of R the improvement is between 2 and 1.5×.

TABLE IV
MAP RESULTS ON CIFAR-10 AND MNIST

Method	CIFAR-10 (MAP)	MNIST (MAP)
LSH [57]	0.120	0.243
SH [2]	0.130	0.250
ITQ [54]	0.175	0.429
BRE [56]	0.196	0.634
MLH [55]	0.211	0.654
ITQ-CCA [54]	0.295	0.726
KSH [53]	0.356	0.900
CNNH [29]	0.522	0.960
CNNH+[29]	0.532	0.975
ACH [7]	0.600	-
KevinNet [24]	0.894	**0.985**
m-k-means-$\mathbf{t_1}$	0.953	0.972
m-k-means-$\mathbf{t_2}$	0.849	0.964
m-k-means-$\mathbf{n_1}$	**0.972**	0.969
m-k-means-$\mathbf{n_2}$	0.901	0.959

Comparison between our method (m-k-means-t_1, m-k-means-n_1 with $n = 24$, m-k-means-t_2 and m-k-means-n_2 with $n = 24$) with KSH [53], ITQ-CCA [54], MLH [55], BRE [56], CNNH [29], CNNH+[29], KevinNet [24], LSH [57], SH [2], ITQ [54]. Results from [7], [24], [29].

G. Results on CIFAR-10, MNIST

In the experiments on CIFAR-10 [42] and MNIST [43] images dataset (Table IV) we use the following configurations for the proposed method: hash code length of 48 bits (the same

length used by the compared methods), arithmetic mean for the *m-k-means-*t_1 variant, $n = 24$ for *m-k-means-*n_1.

Queries are performed using a random selection of 1,000 query images (100 images for each class), considering a category labels ground truth relevance ($Rel(i)$) between a query q and the i^{th} ranked image. So $Rel(i) \in \{0, 1\}$ with 1 for the query and i^{th} images with the same label and 0 otherwise. This setup has been used in [24], [29]. Since we select queries in a random way the results of these experiments are averaged over a set of 10 runs.

We compared the proposed approach with several state-of-the-art hashing methods including supervised (KSH [53], ITQ-CCA [54], MLH [55], BRE [56], CNNH [29], CNNH+[29], KevinNet [24]) and unsupervised methods (LSH [57], SH [2], ITQ [54]).

The proposed method obtains the best results on the more challenging of the two datasets, i.e. CIFAR-10. The comparison with the KevinNet [24] method is interesting since we use the same features, but we obtain better results for all the variants of the proposed method except one. On MNIST dataset the best results of our approach are comparable with the second best method [29], and anyway are not far from the best approach [24].

H. Results on INRIA Holidays, Oxford 5K and Paris 6K

In this experiment we evaluate the effects of the method parameters, i.e. number of nearest centroids n for the *m-k-means-*n_1 method, and Hamming distance threshold (H). The proposed approaches are compared with several state-of-the-art methods, among which the recent UTH method [31]; while some of these methods were originally proposed for engineered features, they have been evaluated on CNN features (results reported from [31]). Retrieval is performed using the coarse-to-fine approach used in [24], where the hash is used to select a candidate list of images and CNN descriptors are used to re-rank this list. For the sake of brevity only some combinations of the parameters of the proposed methods are reported. The *m-k-means-*n_1 method, with an Hamming distance $H \geq 6$ and $n = 6$ greatly outperforms any state-of-the-art hashing method. Also *m-k-means-*t_1 outperforms competing methods, although obtaining a smaller improvement. In general both methods are robust with respect to the parameters.

Of course using methods that use the full CNN descriptors, without hashing, it is possible to obtain better results, for example as in the schemes tested in the work by Wan *et al.* [58], or using the global CNN descriptor proposed by Gordo *et al.* [59], but this comes at the expense of much larger memory occupation and computational costs for retrieval (e.g. the descriptor proposed by Gordo is 512 floats long). In this case the benefit of using a hashing schema is to allow better scaling to larger datasets and improved computational costs.

I. Results Varying Hash Code Length and n

In this experiment (Table V) we compare the behavior of retrieval performance for different lengths of the hash code (for *m-kmeans-*t_1) and for different values of n nearest neighbors (for *m-kmeans-*n_1). Experiments were made for SIFT1M for different values of *recall@R*. Fig. 10 reports the results. We can observe how the performances are good for each signature length and how they converge to 1 from recall@10 onward; differences in performance are already small for $R = 5$. This means that our binary coding method maintains a good representation along different signatures length and, for the *m-k-means-*n_1 variant, also for different values of n.

TABLE V
MAP RESULTS ON HOLIDAYS, OXFORD 5K, AND PARIS 6K DATASETS

Method	Holidays	Oxford 5K	Paris 6K
ITQ [54]	0.537	0.230	-
BPBC [60]	0.381	0.225	-
PCAHash [54]	0.528	0.239	-
LSH [61]	0.431	0.239	-
SKLSH [62]	0.241	0.134	-
SH [2]	0.522	0.232	-
SRBM [63]	0.516	0.212	-
UTH [31]	0.571	0.240	-
m-k-means-$\mathbf{n_1}$ ($n = 6, H = 10$)	**0.756**	**0.460**	0.676
m-k-means-$\mathbf{n_1}$ ($n = 6, H = 16$)	**0.756**	**0.461**	**0.678**
m-k-means-$\mathbf{n_1}$ ($n = 10, H = 10$)	0.743	0.456	0.608
m-k-means-$\mathbf{n_1}$ ($n = 10, H = 16$)	**0.756**	0.460	0.677
m-k-means-$\mathbf{t_1}$ ($n = 10$)	0.683	0.362	0.480

The proposed methods method outperforms all the current state-of-the-art methods. All hashes are 64 bit long.

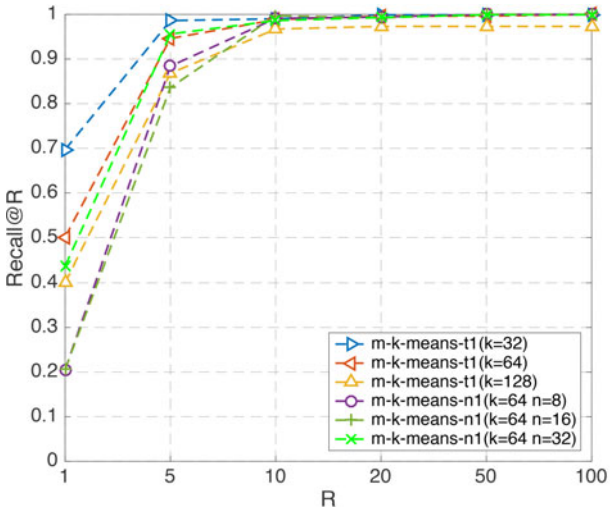

Fig. 10. Experiments on SIFT1M with different signatures length and values of n. We use $k = 32$, $k = 64$, and $k = 128$ for *m-k-means-*t_1 variant; for *m-k-means-*n_1 we use $k = 64$ with $n = 8$, $n = 16$, and $n = 32$.

V. CONCLUSION

We have proposed a new version of the k-means based hashing schema called multi-k-means – with 4 variants: *m-k-means-*t_1, *m-k-means-*t_2, *m-k-means-*n_1 and *m-k-means-*n_2 – which uses a small number of centroids, guarantees a low computational cost and results in a compact quantizer. These characteristics are achieved thanks to the association of the centroids to the bits of the hash code, that greatly reduce the need of a large number of centroids to produce a code of the needed length. Another advantage of the method is that it has no parameters

in its *m-k-means-t_1* and *m-k-means-t_2* variants, and only one parameter for the other two variants; anyway, as shown by the experiments, it is quite robust to variations of such parameter, as well as hash code length.

Our compact hash signature is able to represent high dimensional visual features obtaining a very high efficiency in approximate nearest neighbor (ANN) retrieval, both, on local and global visual features. This characteristic stems from the multiple-assignment strategy, that reduces the need of multi probe strategy to retrieve hash codes that differ by few bits, typically due to quantization errors, and results in better approximated nearest neighbour estimation using Hamming distance. The method has been 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. The *m-k-means-n_1* variant typically performs better than *m-k-means-t_1*, especially when dealing with modern CNN features.

ACKNOWLEDGMENT

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.

V. APPENDIX

Experimental Results on BIGANN and DEEP1B Reported as Tables. See Tables VI–IX here.

TABLE VI
Recall@R ON SIFT1M - COMPARISON BETWEEN OUR METHOD (M-K-MEANS-t_1, M-K-MEANS-n_1 WITH $n = 32$, M-K-MEANS-t_2, AND M-K-MEANS-n_2 WITH $n = 32$), THE PRODUCT QUANTIZATION METHOD (PQ ADC AND PQ IVFADC) [1], CARTESIAN K-MEANS METHOD (CK-MEANS) [15], A NON-EXHAUSTIVE ADAPTATION OF THE OPTIMIZED PRODUCT QUANTIZATION METHOD (I-OPQ), A LOCALLY OPTIMIZED PRODUCT QUANTIZATION METHOD (LOPQ) [18], OPQ-P AND OPQ-NP [16], [51], AND PQ-RO, PQ-RR, RVQ-P, AND RVQ-NP [19]

Method	R@1	R@10	R@100	R@1000	R@10000
PQ (ADC) [1]	0.224	0.600	0.927	0.996	0.999
PQ (IVFADC) [1]	0.320	0.739	0.953	0.972	0.972
PQ-RO [19]	0.177	0.501	0.854	N/A	N/A
PQ-RR [19]	0.107	0.331	0.695	N/A	N/A
ck-means [15]	0.231	0.635	0.930	**1**	**1**
OPQ-P [16], [51]	0.219	0.563	0.917	N/A	N/A
OPQ-NP [16], [51]	0.242	0.627	0.938	N/A	N/A
I-OPQ [18]	0.299	0.691	0.875	0.888	0.888
LOPQ [18]	0.380	0.780	0.886	0.888	0.888
RVQ [52]	0.264	0.659	0.949	**1**	**1**
RVQ-P [19]	0.397	0.821	0.983	N/A	N/A
RVQ-NP [19]	0.271	0.686	0.958	N/A	N/A
m-k-means-$\mathbf{t_1}$	0.501	0.988	**1**	**1**	**1**
m-k-means-$\mathbf{t_2}$	**0.590**	**0.989**	**1**	**1**	**1**
m-k-means-$\mathbf{n_1}$	0.436	0.986	**1**	**1**	**1**
m-k-means-$\mathbf{n_2}$	0.561	0.986	**1**	**1**	**1**

TABLE VII
Recall@R ON GIST1M - COMPARISON BETWEEN OUR METHOD (M-K-MEANS-t_1, M-K-MEANS-n_1 WITH $n = 48$, M-K-MEANS-t_2, AND M-K-MEANS-n_2 WITH $n = 48$), THE PRODUCT QUANTIZATION METHOD (ADC AND IVFADC) [1], CARTESIAN K-MEANS METHOD (CK-MEANS) [15], A NON-EXHAUSTIVE ADAPTATION OF THE OPTIMIZED PRODUCT QUANTIZATION METHOD (I-OPQ) [18], A LOCALLY OPTIMIZED PRODUCT QUANTIZATION METHOD (LOPQ) [18], OPQ-P AND OPQ-NP [16], [51], AND PQ-RO, PQ-RR, RVQ-P, AND RVQ-NP [19]

Method	R@1	R@10	R@100	R@1000	R@10000
PQ (ADC) [1]	0.145	0.315	0.650	0.932	0.997
PQ (IVFADC) [1]	0.180	0.435	0.740	0.966	0.992
PQ-RO [19]	0.034	0.056	0.136	N/A	N/A
PQ-RR [19]	0.033	0.062	0.124	N/A	N/A
ck-means [15]	0.135	0.335	0.728	0.952	0.985
OPQ-P [16], [51]	0.095	0.297	0.629	N/A	N/A
OPQ-NP [16], [51]	0.089	0.277	0.642	N/A	N/A
I-OPQ [18]	0.146	0.410	0.729	0.862	0.866
LOPQ [18]	0.160	0.461	0.756	0.860	0.866
RVQ [52]	0.095	0.276	0.656	0.936	**1**
RVQ-P [19]	**0.309**	0.700	0.950	N/A	N/A
RVQ-NP [19]	0.107	0.314	0.678	N/A	N/A
m-k-means-$\mathbf{t_1}$	0.111	0.906	**1**	**1**	**1**
m-k-means-$\mathbf{t_2}$	0.123	0.890	**1**	**1**	**1**
m-k-means-$\mathbf{n_1}$	0,231	**0,940**	**1**	**1**	**1**
m-k-means-$\mathbf{n_2}$	0,265	0,905	**1**	0,999	0,999

TABLE VIII
Recall@R ON SIFT1B - COMPARISON BETWEEN OUR METHOD (M-K-MEANS-t_1, M-K-MEANS-n_1 WITH $n = 24$), THE PRODUCT QUANTIZATION METHOD [36], A NON-EXHAUSTIVE ADAPTATION OF THE OPTIMIZED PRODUCT QUANTIZATION METHOD (I-OPQ) [18], A MULTI-INDEX METHOD (MULTI-D-ADC), AND A LOCALLY OPTIMIZED PRODUCT QUANTIZATION METHOD (LOPQ) WITH A SUB-OPTIMAL VARIANT (LOR+PQ) [18]

Method	R@1	R@10	R@100
PQ (ADC+R) [36]	0.656	0.970	0.985
PQ (IVFADC+R) [36]	0.630	0.977	0.983
ck-means [15]	0.084	0.288	0.637
I-OPQ [18]	0.114	0.399	0.777
Multi-D-ADC [20]	0.165	0.517	0.860
LOR+PQ [18]	0.183	0.565	0.889
LOPQ [18]	0.199	0.586	0.909
m-k-means-$\mathbf{t_1}$	0.775	0.917	0.928
m-k-means-$\mathbf{n_1}$	**0.787**	**0.990**	**1**

TABLE IX
Recall@R ON DEEP1B - COMPARISON BETWEEN OUR METHOD (M-K-MEANS-t_1, M-K-MEANS-n_1 WITH $n = 24$), INVERTED MULTI-INDEX (IMI) [22], NON-ORTHOGONAL INVERTED MULTI-INDEX (NO-IMI) [22], AND GENERALIZED NON-ORTHOGONAL INVERTED MULTI-INDEX (GNO-IMI) [22]

Method	R@1	R@5	R@10
NO-IMI [22]	0.272	0.492	0.593
IMI [22]	0.241	0.450	0.545
GNO-IMI [22]	0.276	0.508	0.613
m-k-means-$\mathbf{t_1}$	0.694	0.892	0.912
m-k-means-$\mathbf{n_1}$	**0.768**	**0.988**	**0.999**

REFERENCES

[1] H. Jégou, M. Douze, and C. Schmid, "Product quantization for nearest neighbor search," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 33, no. 1, pp. 117–128, Jan. 2011.

[2] Y. Weiss, A. Torralba, and R. Fergus, "Spectral hashing," in *Proc. Neural Inf. Process. Syst.*, 2009, pp. 1753–1760.

[3] P. Li, M. Wang, J. Cheng, C. Xu, and H. Lu, "Spectral hashing with semantically consistent graph for image indexing," *IEEE Trans. Multimedia*, vol. 15, no. 1, pp. 141–152, Jan. 2013.

[4] J. P. Heo, Y. Lee, J. He, S. F. Chang, and S. E. Yoon, "Spherical hashing: Binary code embedding with hyperspheres," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 37, no. 11, pp. 2304–2316, Nov. 2015.

[5] Z. Jin, C. Li, Y. Lin, and D. Cai, "Density sensitive hashing," *IEEE Trans. Cybern.*, vol. 44, no. 8, pp. 1362–1371, Aug. 2014.

[6] S. Du, W. Zhang, S. Chen, and Y. Wen, "Learning flexible binary code for linear projection based hashing with random forest," in *Proc. 22nd IEEE Int. Conf. Pattern Recog.*, Aug. 2014, pp. 2685–2690.

[7] Y. Lv, W. W. Y. Ng, Z. Zeng, D. S. Yeung, and P. P. K. Chan, "Asymmetric cyclical hashing for large scale image retrieval," *IEEE Trans. Multimedia*, vol. 17, no. 8, pp. 1225–1235, Aug. 2015.

[8] L. Paulevé, H. Jégou, and L. Amsaleg, "Locality sensitive hashing: A comparison of hash function types and querying mechanisms," *Pattern Recog. Lett.*, vol. 31, no. 11, pp. 1348–1358, 2010.

[9] W. Zhou, Y. Lu, H. Li, and Q. Tian, "Scalar quantization for large scale image search," in *Proc. ACM Int. Conf. Multimedia*, 2012, pp. 169–178.

[10] G. Ren, J. Cai, S. Li, N. Yu, and Q. Tian, "Scalable image search with reliable binary code," in *Proc. ACM Int. Conf. Multimedia*, 2014, pp. 769–772.

[11] W. Zhou *et al.*, "Towards codebook-free: Scalable cascaded hashing for mobile image search," *IEEE Trans. Multimedia*, vol. 16, no. 3, pp. 601–611, Apr. 2014.

[12] C.-C. Chen and S.-L. Hsieh, "Using binarization and hashing for efficient SIFT matching," *J. Vis. Commun. Image Represent.*, vol. 30, pp. 86–93, 2015.

[13] M. Jain, H. Jégou, and P. Gros, "Asymmetric Hamming embedding: Taking the best of our bits for large scale image search," in *Proc. ACM Int. Conf. Multimedia*, 2011, pp. 1441–1444.

[14] V. Chandrasekhar *et al.*, "Survey of SIFT compression schemes," in *Proc. Int. Workshop Mobile Multimedia Process.*, 2010, pp. 35–40.

[15] M. Norouzi and D. Fleet, "Cartesian k-means," in *Proc. IEEE Conf. Comput. Vis. Pattern Recog.*, Jun. 2013, pp. 3017–3024.

[16] T. Ge, K. He, Q. Ke, and J. Sun, "Optimized product quantization for approximate nearest neighbor search," in *Proc. IEEE Conf. Comput. Vis. Pattern Recog.*, Jun. 2013, pp. 2946–2953.

[17] K. He, F. Wen, and J. Sun, "K-means hashing: An affinity-preserving quantization method for learning binary compact codes," in *Proc. IEEE Conf. Comput. Vis. Pattern Recog.*, Jun. 2013, pp. 2938–2945.

[18] Y. Kalantidis and Y. Avrithis, "Locally optimized product quantization for approximate nearest neighbor search," in *Proc. IEEE Conf. Comput. Vis. Pattern Recog.*, Jun. 2014, pp. 2329–2336.

[19] D. Guo, C. Li, and L. Wu, "Parametric and nonparametric residual vector quantization optimizations for ANN search," *Neurocomputing*, vol. 217, pp. 92–102, 2016.

[20] A. Babenko and V. Lempitsky, "The inverted multi-index," in *Proc. IEEE Conf. Comput. Vis. Pattern Recog.*, Jun. 2012, pp. 3069–3076.

[21] L. Zheng, S. Wang, Z. Liu, and Q. Tian, "Packing and padding: Coupled multi-index for accurate image retrieval," in *Proc. IEEE Conf. Comput. Vis. Pattern Recog.*, Jun. 2014, pp. 1947–1954.

[22] A. Babenko and V. Lempitsky, "Efficient indexing of billion-scale datasets of deep descriptors," in *Proc. IEEE Conf. Comput. Vis. Pattern Recog.*, Jun. 2016, pp. 2055–2063.

[23] L. Zheng, S. Wang, J. Wang, and Q. Tian, "Accurate image search with multi-scale contextual evidences," *Int. J. Comput. Vis.*, vol. 120, no. 1, pp. 1–13, Oct. 2016.

[24] K. Lin, H.-F. Yang, J.-H. Hsiao, and C.-S. Chen, "Deep learning of binary hash codes for fast image retrieval," in *Proc. IEEE Conf. Comput. Vis. Pattern Recog. Workshops*, Jun. 2015, pp. 27–35.

[25] T.-T. Do, A.-Z. Doan, and N.-M. Cheung, "Discrete hashing with deep neural network," *CoRR*, 2015. [Online]. Available: http://arxiv.org/abs/1508.07148

[26] J. Guo and J. Li, "CNN based hashing for image retrieval," *CoRR*, 2015. [Online]. Available: http://arxiv.org/abs/1509.01354

[27] Z. Zhang, Y. Chen, and V. Saligrama, "Supervised hashing with deep neural networks," *CoRR*, 2015. [Online]. Available: http://arxiv.org/abs/1511.04524

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

[29] R. Xia, Y. Pan, H. Lai, C. Liu, and S. Yan, "Supervised hashing for image retrieval via image representation learning," in *Proc. AAAI Conf. Artif. Intell.*, 2014, pp. 2156–2162.

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

[31] J. Lin, O. Morère, J. Petta, V. Chandrasekhar, and A. Veillard, "Tiny descriptors for image retrieval with unsupervised triplet hashing," in *Proc. Data Compression Conf.*, 2016, pp. 397–406.

[32] D. Arthur and S. Vassilvitskii, "k-means++: The advantages of careful seeding," in *Proc. ACM-SIAM Symp. Discrete Algorithms*, 2007, pp. 1027–1035.

[33] K. Chatfield, K. Simonyan, A. Vedaldi, and A. Zisserman, "Return of the devil in the details: Delving deep into convolutional nets," in *Proc. Brit. Mach. Vis. Conf.*, 2014.

[34] J. van Gemert, C. Veenman, A. Smeulders, and J.-M. Geusebroek, "Visual word ambiguity," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 32, no. 7, pp. 1271–1283, Jul. 2010.

[35] Q. Lv, W. Josephson, Z. Wang, M. Charikar, and K. Li, "Multi-probe LSH: Efficient indexing for high-dimensional similarity search," in *Proc. Int. Conf. Very Large Data Bases*, 2007, pp. 950–961.

[36] H. Jégou, R. Tavenard, M. Douze, and L. Amsaleg, "Searching in one billion vectors: Re-rank with source coding," in *Proc. Int. Conf. Acoust., Speech, Signal Process.*, 2011, pp. 861–864.

[37] M. Norouzi, A. Punjani, and D. Fleet, "Fast exact search in Hamming space with multi-index hashing," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 36, no. 6, pp. 1107–1119, Jun. 2014.

[38] H. Jegou, M. Douze, and C. Schmid, "Hamming embedding and weak geometric consistency for large scale image search," in *Proc. Eur. Conf. Comput. Vis.*, 2008, pp. 304–317.

[39] A. Torralba, R. Fergus, and W. T. Freeman, "80 million tiny images: A large data set for nonparametric object and scene recognition," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 30, no. 11, pp. 1958–1970, Nov. 2008.

[40] C. Szegedy *et al.*, "Going deeper with convolutions," in *Proc. IEEE Conf. Comput. Vis. Pattern Recog.*, Jun. 2015, pp. 1–9.

[41] J. Deng *et al.*, "ImageNet: A large-scale hierarchical image database," in *Proc. IEEE Conf. Comput. Vis. Pattern Recog.*, Jun. 2009, pp. 248–255.

[42] A. Krizhevsky and G. Hinton, "Learning multiple layers of features from tiny images," M.S. thesis, Dept. Comput. Sci., Univ. Toronto, Toronto, ON, Canada, 2009.

[43] Y. LeCun, C. Cortes, and C. J. Burges, "The MNIST database of handwritten digits," *Proc. IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.

[44] J. Philbin, O. Chum, M. Isard, J. Sivic, and A. Zisserman, "Object retrieval with large vocabularies and fast spatial matching," in *Proc. IEEE Conf. Comput. Vis. Pattern Recog.*, Jun. 2007, pp. 1–8.

[45] J. Philbin, O. Chum, M. Isard, J. Sivic, and A. Zisserman, "Lost in quantization: Improving particular object retrieval in large scale image databases," in *Proc. IEEE Conf. Comput. Vis. Pattern Recog.*, Jun. 2008, pp. 1–8.

[46] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Proc. Neural Inf. Process. Syst.*, 2012, pp. 1097–1105.

[47] Y. Jia *et al.*, "Caffe: Convolutional architecture for fast feature embedding," in *Proc. ACM Int. Conf. Multimedia*, 2014, pp. 675–678.

[48] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proc. IEEE*, vol. 86, no. 11, pp. 2278–2324, Nov. 1998.

[49] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," in *Proc. Int. Conf. Learn. Represent.*, 2015.

[50] T. Uricchio, M. Bertini, L. Seidenari, and A. Del Bimbo, "Fisher encoded convolutional bag-of-windows for efficient image retrieval and social image tagging," in *Proc. IEEE Int. Conf. Comput. Vis. Workshops*, Dec. 2015, pp. 1020–1026.

[51] T. Ge, K. He, Q. Ke, and J. Sun, "Optimized product quantization," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 36, no. 4, pp. 744–755, Apr. 2014.

[52] Y. Chen, T. Guan, and C. Wang, "Approximate nearest neighbor search by residual vector quantization," *Sensors*, vol. 10, no. 12, pp. 11 259–11 273, 2010.

[53] W. Liu, J. Wang, R. Ji, Y.-G. Jiang, and S.-F. Chang, "Supervised hashing with kernels," in *Proc. IEEE Conf. Comput. Vis. Pattern Recog.*, Jun. 2012, pp. 2074–2081.

[54] Y. Gong, S. Lazebnik, A. Gordo, and F. Perronnin, "Iterative quantization: A procrustean approach to learning binary codes for large-scale image retrieval," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 35, no. 12, pp. 2916–2929, Dec. 2013.

[55] M. Norouzi and D. J. Fleet, "Minimal loss hashing for compact binary codes," in *Proc. Int. Conf. Mach. Learn.*, 2011, pp. 353–360.
[56] B. Kulis and T. Darrell, "Learning to hash with binary reconstructive embeddings," in *Proc. Int. Conf. Neural Inf. Process. Syst.*, 2009, pp. 1042–1050.
[57] A. Gionis, P. Indyk, and R. Motwani, "Similarity search in high dimensions via hashing," in *Proc. Int. Conf. Very Large Data Bases*, 1999, pp. 518–529.
[58] J. Wan *et al.*, "Deep learning for content-based image retrieval: A comprehensive study," in *Proc. ACM Int. Conf. Multimedia*, 2014, pp. 157–166.
[59] A. Gordo, J. Almazán, J. Revaud, and D. Larlus, "Deep image retrieval: Learning global representations for image search," in *Proc. Eur. Conf. Comput. Vis.*, 2016, pp. 241–257.
[60] Y. Gong, S. Kumar, H. Rowley, and S. Lazebnik, "Learning binary codes for high-dimensional data using bilinear projections," in *Proc. IEEE Conf. Comput. Vis. Pattern Recog.*, Jun. 2013, pp. 484–491.
[61] M. Datar, N. Immorlica, P. Indyk, and V. S. Mirrokni, "Locality-sensitive hashing scheme based on p-stable distributions," in *Proc. Symp. Comput. Geom.*, 2004, pp. 253–262.
[62] M. Raginsky and S. Lazebnik, "Locality-sensitive binary codes from shift-invariant kernels," in *Proc. Int. Conf. Neural Inf. Process. Syst.*, 2009, pp. 1509–1517.
[63] V. Chandrasekhar, J. Lin, O. Morere, A. Veillard, and H. Goh, "Compact global descriptors for visual search," in *Proc. Data Compression Conf.*, 2015, pp. 333–342.

Authors' photographs and biographies not available at the time of publication.