Memory-efficient non-parametric image classification using compact binary codes

Frederick Tung
Laboratory for Computational Intelligence
University of British Columbia
ftung@cs.ubc.ca

***Abstract*—Non-parametric image classification methods such as Naive Bayes Nearest Neighbour (NBNN) do not require a training phase and scale naturally to a large number of classes. However, they are also memory intensive because they require the storage and indexing of all database image feature descriptors in memory. In this paper, we propose a memory-efficient NBNN image classification algorithm that employs compact binary codes. We evaluate three modern techniques for constructing compact binary codes: spectral hashing, iterative quantization, and k-means hashing. Our experiments show that substantial memory efficiency gains are possible at a moderate cost in classification accuracy. We further discuss two novel extensions for future investigation and present preliminary experimental results for both.**

I. Introduction

In image classification, we are interested in determining the general semantic category of an image, such as 'cat' or 'camera'. Objects in an image can be described and classified based on a variety of features, such as image gradients [1], [2], contour segments [3], [4], or self-similarities [5], [6]. Once feature descriptors are extracted, classification techniques are applied to predict the object's semantic category.

Parametric classification techniques learn a model of the object categories from labelled data in a training phase. In contrast, non-parametric classification techniques, such as nearest neighbour, classify new instances by comparing them directly to the labelled data. Non-parametric approaches do not require a training phase and scale naturally to multiple categories.

Boiman et al. [7] observed that previous non-parametric approaches to image classification were adversely affected by two common practices: the coarse quantization of feature descriptors (into codewords in a bag of visual words framework [8]) and the use of image-to-image distances instead of image-to-class distances. To address these issues, Boiman et al. proposed the Naive Bayes Nearest Neighbour (NBNN) image classifier, in which a query image is classified by selecting the class that minimizes the sum of distances between the query image's feature descriptors and their nearest-neighbour descriptor in the class. McCann and Lowe [9] later proposed a variation called Local NBNN, which compares the query image's feature descriptors to their nearest neighbours in a single pool of descriptors shared over all classes instead of their class-specific nearest neighbours.

In the NBNN framework, feature descriptor quantization is avoided by storing all feature descriptors extracted from all database images, and performing approximate nearest neighbour searches at test time. However, storing all database image feature descriptors makes NBNN image classification highly memory intensive.

In this paper, we propose a memory-efficient NBNN image classification algorithm that replaces the high-dimensional feature descriptors with compact binary codes. The low-dimensional binary codes require less memory and can be evaluated quickly, yet closely preserve the locality relationships of the original feature descriptors. We consider and evaluate three modern techniques for constructing compact binary codes: spectral hashing [10], iterative quantization [11], and k-means hashing [12]. Experiments on the Caltech-101 [13] and Caltech-256 [14] datasets indicate that substantial memory efficiency gains are possible at a moderate cost in image classification accuracy.

In addition, we discuss two novel extensions for future investigation, which aim to further improve the image classification accuracy of our memory-efficient NBNN algorithm. The first extension tries to improve the retrieval performance of the compact binary codes by incorporating feature-space neighbour information. The second extension is a novel technique for learning binary codes that is inspired by iterative quantization [11].

II. Related work

A. Methods for image classification

Parametric image classification methods learn a model of the object categories from labelled data in a training phase. The traditional bag of visual words [8] framework for image classification creates appearance-based vocabularies, or codebooks, by quantizing the feature descriptor space into codewords. Image classification is performed by extracting feature descriptors from the image, building a histogram over the codewords, and classifying the histogram using, for example, a support vector machine. Lazebnik et al. [15] introduced spatial pyramid matching, which extends the bag of visual words approach by encoding coarse spatial information. Bosch et al. [16] proposed computing the spatial pyramid over a region of interest instead of the entire image, and in addition to appearance (visual words), incorporating shape using histograms of oriented gradients. Dalal and Triggs [1] proposed an image representation consisting of a dense overlapping grid of locally normalized histograms of oriented gradient. Shechtman and Irani [6] proposed the local self-similarity descriptor. They observed that two objects of the same class may have different photometric properties, yet can be recognized to be

of the same class by their shared recurring patterns of self-similarity. Deselaers and Ferrari [5] introduced the global self-similarity descriptor to efficiently capture self-similarities across the entire image. Ferrari et al. [3] introduced k Adjacent Segments, a local scale-invariant feature based on groups of approximately straight segments fit to edgel-chains. In a later work, Ferrari et al. [17] developed an explicit shape model, in which a prototypical continuous contour and its principal deformations are learned for each object class. Felzenszwalb et al. [2] introduced a deformable parts model consisting of a root filter that "anchors" a deformable arrangement of higher resolution parts filters. Perronnin et al. [18], [19] proposed Fisher kernel encoding as an alternative to the traditional bag of visual words. Jegou et al. [20] introduced the Vector of Locally Aggregated Descriptors (VLAD) image representation, which can be viewed as a simplification of the Fisher vector.

Non-parametric image classification methods classify new instances by comparing them directly to the labelled data, and do not require a training phase. As previously described, Boiman et al. [7] proposed Naive Bayes Nearest Neighbour (NBNN) image classification. McCann and Lowe [9] developed the Local NBNN algorithm. Kanan and Cottrell [21] proposed learning ICA features (filters) from natural scene images and applying them to describe biologically inspired "fixations" in new images. The fixation features are used instead of the usual SIFT descriptors in an NBNN classifier. Timofte et al. [22] extended the Naive Bayes framework by proposing several alternatives to the usual nearest neighbour classification rule.

B. Methods for learning binary codes

Researchers have made many advances in learning distance-preserving compact binary codes for large-scale computer vision applications. By storing and comparing low-dimensional binary representations of high-dimensional feature descriptors, both memory requirements and query time can be greatly reduced. Comparison between binary codes is particularly efficient because the Hamming distance between two codes can be computed by applying an XOR and counting the ones, which is a fast operation supported directly in modern hardware.

Locality sensitive hashing (LSH) [23], [24] is a traditional technique for quickly finding similar data points in a high-dimensional space and was introduced as a method for approximate nearest neighbour search. LSH relies on collections of hash functions that hash similar data points to the same bin with high probability and hash dissimilar data points to the same bin with low probability.

In computer vision, Torralba et al. [25] performed large-scale image matching using binary embeddings of Gist [26] descriptors. The authors experimented with variants of parameter sensitive hashing [27] and semantic hashing [28] to learn the binary codes. Parameter sensitive hashing was introduced by Shakhnarovich et al. [27] for human pose estimation. The technique extends locality sensitive hashing to find near neighbours in parameter space. Each hash function is a binary classifier that is trained to distinguish between pairs of data points that are close in parameter space and pairs that are distant. Salakhutdinov and Hinton [28] proposed the semantic hashing method in the context of document retrieval. Low-dimensional binary codes are learned by training a deep autoencoder neural network on high-dimensional word count inputs. Later, Weiss et al. [10] demonstrated how the problem of finding compact binary codes is closely related to balanced graph partitioning, and proposed the spectral hashing (SH) algorithm to solve a relaxed version of this problem.

We next describe in some detail two state-of-the-art binary encoding techniques: iterative quantization [11] and k-means hashing [12]. We subsequently evaluate both of these techniques, as well as spectral hashing, in the context of our proposed memory-efficient NBNN image classification algorithm.

1) Iterative quantization (ITQ): Gong and Lazebnik [11] proposed the iterative quantization (ITQ) technique for learning compact binary codes. ITQ finds a rotation of the PCA-projected, zero-centred data such that the quantization error of mapping the data to the closest corners of a zero-centred binary hypercube is minimized. The required rotation is determined in an iterative process that alternates between assigning data points to the closest hypercube corner given the current rotation, and updating the rotation to minimize the quantization error given the current assignments.

Following the notation of Gong and Lazebnik [11], let $B \in \{-1, 1\}^{n \times c}$ be the binary code matrix, where n is the number of data points and c is the code length in bits, and let V be the PCA-projected data matrix. The quantization error for a rotation matrix R is given by [11]

$$Q(B, R) = ||B - VR||_F^2 \tag{1}$$

where $||\cdot||_F$ denotes the Frobenius norm. The iterative process alternates between updating B given the current R, and updating R given the current B. It can be shown that for a fixed R, Q is minimized by updating B by assigning data points to the closest corner of the binary hypercube, and for a fixed B, Q is minimized by $R = \hat{S}S^T$, where $S\Omega\hat{S}^T$ is the SVD of B^TV. The iterative process is initialized with a random rotation.

2) K-means hashing (KMH): He et al. [12] recently proposed the k-means hashing (KMH) method for learning compact binary codes. In KMH, data points are clustered such that the Euclidean distance between two data points can be approximated by the Hamming distance between their cluster indices. KMH trades off quantization error as in the usual k-means algorithm with "affinity error", which refers to the difference between the pairwise Euclidean distances and the corresponding pairwise Hamming distances of the cluster indices. The objective is minimized in an iterative process that alternates between fixing the cluster centroids and assigning data points to their nearest cluster, and fixing the assignments and sequentially optimizing the cluster centroids.

KMH can be scaled to longer binary codes using product quantization [29]. Product quantization splits the feature space into multiple subspaces and quantizes each subspace separately (e.g. by k-means). Applying product quantization to KMH, binary codes are learned for each subspace and the final binary code is obtained by concatenating the binary codes for each subspace.

III. MEMORY-EFFICIENT NBNN

Non-parametric approaches do not require a training phase and scale naturally to multiple classes. However, they incur high memory costs because all database image feature descriptors are retained for comparison at query time. In this section, we first explain the NBNN image classification framework in detail. We then describe our proposed memory-efficient NBNN image classification algorithm that uses compact binary codes to reduce the high memory costs of traditional NBNN.

Boiman et al. [7] noted that the most distinctive and informative feature descriptors for classification appear infrequently in the dataset. However, quantization error is highest for infrequent descriptors. As a result, quantization of descriptors into visual words [8] decreases the discriminative power of descriptors that are individually highly informative. One way to address this problem might be to discard descriptors that are individually less informative. However, there are many such descriptors and in combination they may yet provide a strong cue, as in an ensemble of weak classifiers [7].

Non-parametric classification using image-to-image distances performs well only when the query image closely matches a labelled image in the dataset. When the labelled dataset is small, a close match may not be available. An image-to-class distance, based on all images in the class, improves the generalization of the classifier since it allows the query image to be matched by "composing pieces" from multiple images of the same class [7].

The NBNN image classification algorithm proposed by Boiman et al. addresses both issues: the classifier avoids descriptor quantization and minimizes an image-to-class KL distance. The following is based on the derivation in Boiman et al. [7].

Consider the Maximum Likelihood (ML) classifier for a query image Q:

$$\hat{C} = \arg\max_C p(Q|C) \tag{2}$$

Let $d_1, ..., d_n$ be the feature descriptors in Q. Under the Naive Bayes assumption,

$$p(Q|C) = p(d_1, ..., d_n|C) = \prod_{i=1}^{n} p(d_i|C) \tag{3}$$

Taking the log probability of Eq. (2) gives

$$\hat{C} = \arg\max_C \sum_{i=1}^{n} \log p(d_i|C) \tag{4}$$

The probability density $p(d|C)$ for a feature descriptor d can be approximated using Parzen windows estimation:

$$\hat{p}(d|C) = \frac{1}{L} \sum_{j=1}^{L} K(d - d_j^C) \tag{5}$$

where $d_1^C, ..., d_L^C$ are the descriptors in images of class C, and $K()$ is the Parzen window function (kernel), such as a Gaussian. Most of the terms in the summation are negligible, and Eq. (5) can be approximated by a small number of nearest neighbour descriptors. Boiman et al. showed that using even a single nearest neighbour is feasible, which yields a very simple classification rule, assuming a Gaussian kernel:

$$\begin{aligned} \hat{C} &= \arg\max_C \sum_{i=1}^{n} -||d_i - \text{NN}_C(d_i)||^2 \\ &= \arg\min_C \sum_{i=1}^{n} ||d_i - \text{NN}_C(d_i)||^2 \end{aligned} \tag{6}$$

where $\text{NN}_C(d_i)$ denotes the nearest neighbour descriptor of d_i in class C.

The Local NBNN algorithm [9] achieves a significant speed-up by comparing the query image's feature descriptors to their nearest neighbours in a single pool of descriptors shared over all classes instead of their class-specific nearest neighbours. Intuitively, instead of asking whether a feature descriptor "looks like" each particular class, Local NBNN asks which classes a feature descriptor most looks like [9]. Approximate nearest neighbours are efficiently computed using randomized kd-trees [30] as implemented in FLANN [31]. Local NBNN provides greater scalability to a large number of classes as search time scales logarithmically with the number of classes instead of linearly. The approach also obtains an increase in classification accuracy. For these reasons, we build our memory-efficient NBNN algorithm on top of the Local NBNN framework.

The proposed memory-efficient NBNN algorithm uses compact binary codes to reduce the memory costs of traditional NBNN image classification.

The problem of constructing compact binary codes can be summarized as follows. Given a database $X \in \mathbb{R}^{n \times m}$ of n feature descriptors with dimension m, we would like to compute a mapping to compact binary codes $B \in \{-1, 1\}^{n \times c}$, where c is the code length in bits, such that the locality relationships in the original feature space are closely preserved in the reduced Hamming space. The mapping is applied to all feature descriptors in the database. Instead of storing the full-dimensional descriptors X, we store only the binary codes Y, obtaining significant gains in memory efficiency.

Given a query image, its feature descriptors are similarly mapped to binary codes and the usual Local NBNN classification is performed in the reduced Hamming space.

Techniques for constructing compact binary codes can only imperfectly preserve the locality relationships in the original feature space. We are interested in evaluating the extent to which image classification performance can be preserved under the proposed framework. In our experiments, we evaluate three modern techniques for constructing compact binary codes, as previously described in Section II: spectral hashing [10], iterative quantization [11], and k-means hashing [12].

IV. EXPERIMENTS

We performed experiments on the Caltech-101 and Caltech-256 image classification benchmarks [13], [14], which McCann and Lowe used to evaluate Local NBNN [9]. SIFT [32] features were extracted at four scales along a regular grid

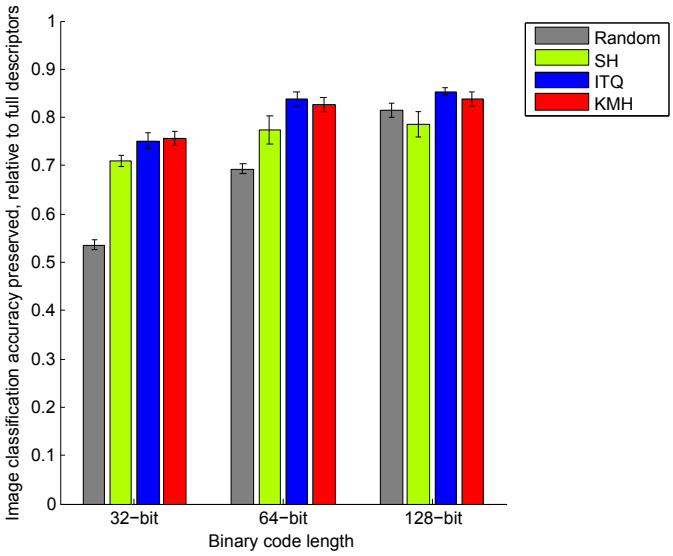


Fig. 1. Experimental results for the Caltech-101 dataset, with a database size of 15 labelled images per class. The coloured bars show the performance of the proposed memory-efficient NBNN algorithm using three modern techniques for constructing binary codes: spectral hashing (SH), iterative quantization (ITQ), and k-means hashing (KMH). The grey bars show the performance of a baseline in which feature descriptors are randomly subsampled to obtain the same reduction in memory as the binary codes.

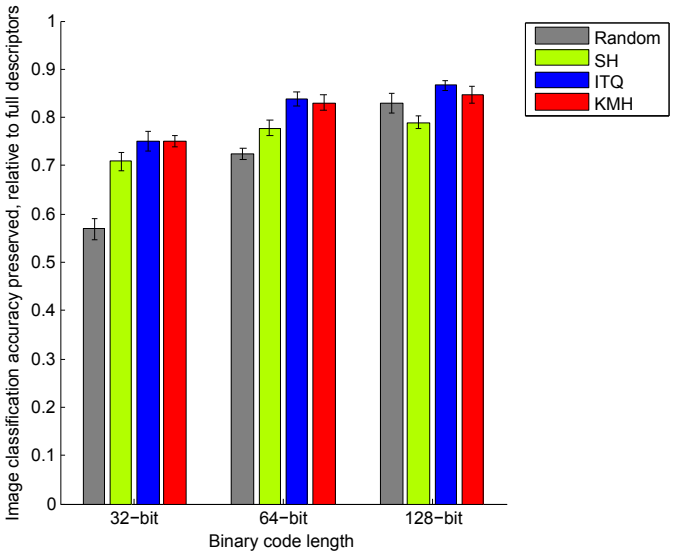


Fig. 2. Experimental results for the Caltech-101 dataset, with a database size of 30 labelled images per class. The coloured bars show the performance of the proposed memory-efficient NBNN algorithm using three modern techniques for constructing binary codes: spectral hashing (SH), iterative quantization (ITQ), and k-means hashing (KMH). The grey bars show the performance of a baseline in which feature descriptors are randomly subsampled to obtain the same reduction in memory as the binary codes.

with a step size of ten pixels. Descriptors with low contrast were discarded following McCann and Lowe [9]. We reproduced the Local NBNN algorithm following the parameter settings in their paper. To accelerate nearest neighbour matching at query time, we used the approximate nearest neighbour search for binary features as implemented in FLANN [31]. McCann and Lowe similarly used FLANN to match full-dimensional SIFT descriptors.

In our Caltech-101 experiments, we evaluated spectral hashing, iterative quantization, and k-means hashing methods for constructing the compact binary codes. We evaluated three binary code lengths: 32, 64, and 128 bits. Of primary interest are the memory efficiency gains that are possible and the extent to which classification performance can be preserved when using these compact codes instead of 128-dimensional SIFT descriptors.

We integrated publicly available implementations of spectral hashing, iterative quantization, and k-means hashing into our implementation of Local NBNN. For k-means hashing, we performed the split into product subspaces following the split used by He et al. [12] in their experimental evaluation on a large database of SIFT descriptors (2 bits per subspace for 32-bit binary codes, and 4 bits per subspace for 64-bit and 128-bit binary codes).

Figure 1 shows experimental results for a database size of 15 training images per class, a common test point in Caltech-101. The vertical axis is the percentage of the mean per-class accuracy preserved by the various binary codes, relative to the full-dimensional SIFT descriptors. The horizontal axis is the binary code length: 32, 64, or 128 bits. For each code length we plot bars for spectral hashing, iterative quantization, and k-means hashing. The error bars indicate standard deviations over five trials.

The plot illustrates the tradeoff between gains in memory efficiency and image classification accuracy. At 128 bits, the binary codes require 8 times less memory than 128-dimensional SIFT descriptors. The best performing methods for constructing compact binary codes, iterative quantization and k-means hashing, achieve 85% and 84%, respectively, of the image classification accuracy obtained by the full-dimensional SIFT descriptors. At 64 bits, these same methods still obtain 84% and 83%, respectively, of the classification accuracy of the full-dimensional SIFT descriptors. Even at 32 bits, where the binary codes require 32 times less memory than 128-dimensional SIFT descriptors, 75% and 76% of the classification accuracy can still be preserved using iterative quantization and k-means hashing. For reference, randomly subsampling 1/8 of the feature descriptors to obtain the same memory savings as the 128-bit binary codes preserves 81% of the image classification accuracy. Randomly subsampling 1/16 and 1/32 of the descriptors to obtain the same memory savings as the 64-bit and 32-bit binary codes preserves 69% and 54% of the image classification accuracy, respectively.

Figure 2 shows similar results when the size of the database is increased to 30 training images per class, another common test point. At 128 bits, the binary codes constructed by iterative quantization and k-means hashing achieve 87% and 85%, respectively, of the image classification accuracy obtained by the full-dimensional SIFT descriptors. At 64 bits, these methods preserve 84% and 83% of the image classification accuracy. Even at 32 bits, representing a 32-fold reduction in memory requirements, 75% of the original classification accuracy can be preserved using iterative quantization and k-means hashing.

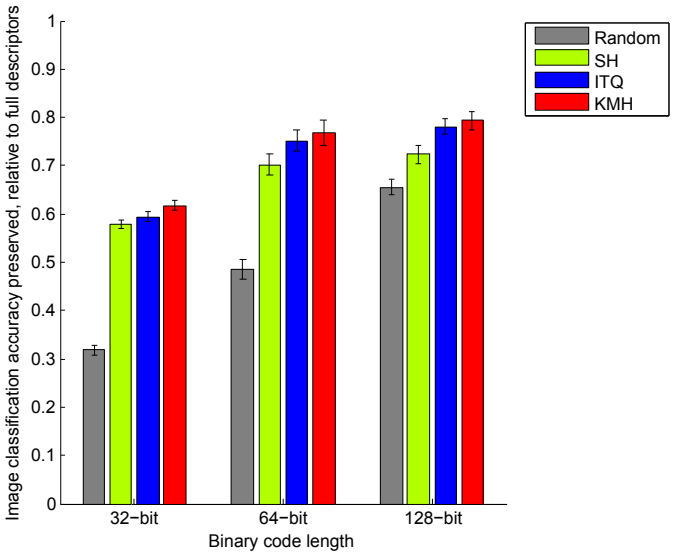


Fig. 3. Experimental results for the Caltech-256 dataset, with a database size of 15 labelled images per class. The coloured bars show the performance of the proposed memory-efficient NBNN algorithm using three modern techniques for constructing binary codes: spectral hashing (SH), iterative quantization (ITQ), and k-means hashing (KMH). The grey bars show the performance of a baseline in which feature descriptors are randomly subsampled to obtain the same reduction in memory as the binary codes.

We performed additional experiments on the Caltech-256 dataset with 15 training images per class. The results are shown in Figure 3. On Caltech-256, 128-bit binary codes constructed by iterative quantization and k-means hashing achieve 78% and 79% of the image classification accuracy possible with full-dimensional SIFT descriptors. At 64 bits, these methods preserve 75% and 77% of the image classification accuracy. Finally, at 32 bits, iterative quantization and k-means hashing preserve 59% and 62% of the image classification accuracy, respectively. For reference, randomly subsampling 1/8, 1/16, and 1/32 of the feature descriptors to obtain the same memory savings as the 128-bit, 64-bit, and 32-bit binary codes preserves only 66%, 49%, and 32% of the image classification accuracy, respectively.

We conclude that, by applying state-of-the-art techniques such as iterative quantization and k-means hashing to construct compact binary codes, our memory-efficient NBNN image classification algorithm can obtain substantial improvements in memory efficiency compared with standard NBNN, at moderate tradeoffs in classification accuracy.

V. Extensions

We next describe two extensions for future investigation: *neighbourhood expansion* and the *bank of random rotations*. Both extensions aim to further improve the classification accuracy of the memory-efficient NBNN algorithm. The first extension attempts to boost the retrieval performance of the compact binary codes by employing feature-space neighbour information. The second extension proposes a novel data-adaptive method for learning binary codes, which we demonstrate to provide improved performance over the natural baseline of iterative quantization (ITQ) on a standard retrieval dataset.

VI. Neighbourhood expansion

A. Introduction

Since methods for constructing compact binary codes imperfectly preserve the original feature-space locality relationships in the reduced Hamming space, true feature-space neighbours are often missed when performing retrieval using compact binary codes. To improve the retrieval performance of compact binary codes, we propose a simple technique called *neighbourhood expansion*, which augments the retrieved list of Hamming-space neighbours with feature-space neighbours. We show that neighbourhood expansion can be applied as a complementary technique to obtain a modest boost in the retrieval performance of iterative quantization and k-means hashing.

B. Proposed technique

True neighbours in the original feature space may be missed when performing retrieval using compact binary codes. That is, given a query feature descriptor x and its corresponding binary code y, a nearest neighbour search of y among the stored binary codes may not produce the same indices as a nearest neighbour search of x among the original feature descriptors. Some true feature-space neighbours of x may be mapped far from y in the Hamming space, while some non-neighbours of x may be mapped close to y in the Hamming space.

To improve the retrieval performance when using compact binary codes, we propose a simple post-processing technique that we refer to as neighbourhood expansion. Neighbourhood expansion first requires the n_f nearest neighbour indices of each data point in the original feature space to be computed. Let $L^{n \times n_f}$ be the lookup table of nearest neighbour indices, such that $L(i, j)$ contains the index of the jth nearest neighbour of the ith feature descriptor in the dataset. This computation is performed as a pre-processing step and is required only once per dataset.

Given a query feature descriptor x, we encode the query descriptor into its compact binary code y and execute the usual search among the database binary codes. Let $\{i(y_1), i(y_2), ..., i(y_N)\}$ be the resulting sorted list of nearest neighbour indices of y. This list of indices would normally be returned as the query result. However, we would like to improve the quality of the list of retrieved indices by incorporating information about the feature-space neighbours of the Hamming-space neighbours.

For each retrieved Hamming-space neighbour, we look up its n_f nearest neighbours in the original feature space. For example, for the nearest Hamming neighbour, the nearest neighbour indices in the original feature space are $L(i(y_1), 1), ..., L(i(y_1), n_f)$. This lookup provides us with a list of *expansion indices* E,

$$\begin{aligned} &\{L(i(y_1), 1), L(i(y_1), 2), ..., L(i(y_1), n_f), \\ &L(i(y_2), 1), L(i(y_2), 2), ..., L(i(y_2), n_f), \\ &..., L(i(y_N), 1), L(i(y_N), 2), ..., L(i(y_N), n_f)\} \end{aligned} \quad (7)$$

Next, the expansion indices are interleaved with the original query result $\{i(y_1), i(y_2), ..., i(y_N)\}$ to form an augmented list of indices. The interleaving operation is parameterized by the spacing at which expansion indices are inserted into the original query result, denoted δ. For example, if $\delta = 1$ then an expansion index is inserted after each original index, and if $\delta = 2$ then an expansion index is inserted every two original indices. To illustrate, if N is even and $\delta = 2$, then the augmented list A is given by

$$\{i(y_1), i(y_2), E(1), i(y_3), i(y_4), E(2), ..., i(y_{N-1}), i(y_N), E(N/2)\} \quad (8)$$

where the remaining expansion indices in E are dropped from further consideration. Finally, the augmented list of indices A is pruned for duplicates and the first N entries are returned as the final query result A'.

Figure 4 shows a graphical summary of the neighbourhood expansion technique with a toy example. Suppose we are interested in performing image retrieval against a large database of natural scene images. We extract Gist [26] feature descriptors for each image and use k-means hashing to learn compact binary codes. Given a query descriptor x, say of a mountain image, we would like to retrieve similar mountain images from the database. We compute the binary code representation y of the query descriptor and search for the $N = 6$ nearest neighbours of y in the database of binary codes. The search produces six indices $\{i(y_1), i(y_2), ..., i(y_6)\}$, each corresponding to an image in the database, some of which are similar mountain images but others of which are irrelevant. To improve the retrieval result, we look up the feature-space neighbour indices of the retrieved indices to form a list of expansion indices E. The expansion indices are combined with the original indices $\{i(y_1), i(y_2), ..., i(y_6)\}$ to form an augmented list of indices A. After removing duplicates, we return the first six indices as the final retrieval result A', providing the user with one additional relevant mountain image and removing one irrelevant forest image.

Neighbourbood expansion requires the table L of feature-space neighbour indices to be loaded in memory. To maintain the memory efficiency gains of using compact binary codes, we limit the size of L to match the memory usage of the database of binary codes. For example, we set $n_f = 2$ for 64-bit binary codes, and $n_f = 4$ for 128-bit binary codes. Since neighbourhood expansion can be performed as a post-processing step after performing all queries against the database of binary codes, and does not require the database of binary codes, this limit on L ensures that if the system has enough memory to load the binary codes, then it also has enough memory to load L.

Though simple, neighbourhood expansion can lift the retrieval performance of state-of-the-art binary encoding methods such as iterative quantization and k-means hashing, as a complementary technique. Neighbourhood expansion is generic and does not assume any particular binary encoding method.

C. Experiments

To evaluate the feasibility of the proposed technique, we followed the experimental evaluation framework of He et al. for k-means hashing [12]. Experiments were performed on the SIFT1M dataset [29], which consists of 1 million SIFT [32] descriptors for the database and 10,000 independent descriptors for queries. We followed the convention of He et al. and considered the ground truth to be the query's 10 nearest neighbours in the database.

The retrieval performance of a binary encoding method can be characterized by a recall@N curve [12], [29], which plots the proportion of true neighbours of x retrieved in the N nearest neighbours of y. Figure 5 shows the recall@N curves for binary codes of length 32, 64, and 128 bits. In each figure we plot the recall of iterative quantization by itself (ITQ), iterative quantization with neighbourhood expansion (ITQ+NE), k-means hashing by itself (KMH), and k-means hashing with neighbourhood expansion (KMH+NE). We found that, for all three binary code lengths, neighbourhood expansion provides a modest boost in the retrieval performance of both iterative quantization and k-means hashing.

Neighbourhood expansion has one free parameter: the spacing parameter δ. Figure 6 shows the effect of varying the spacing parameter for 64-bit binary codes constructed by iterative quantization and k-means hashing. Setting $\delta = 2, 4$ obtains almost identical results, and we generate all other results in this section with $\delta = 2$.

Finally, since computation of the table L of exact feature-space nearest neighbours may be time consuming (though required only once per dataset), we investigated the possibility of substituting an efficient approximate nearest neighbour search. Figure 7 shows the effect of using the randomized kd-trees [30] approximate nearest neighbour search as implemented in FLANN [31] for several values of FLANN's 'checks' parameter with 4 trees. Figure 7 is generated with 64-bit binary codes constructed by iterative quantization and k-means hashing. These results show that, at a modest cost in the retrieval improvement, it is feasible to replace the exact neighbours in L with approximate neighbours.

D. Conclusion

In this section, we have described the neighbourhood expansion technique for improving the retrieval performance of compact binary codes. Our experiments on a dataset of 1 million SIFT descriptors demonstrate the validity of the proposed technique. However, the improvement in retrieval performance is modest and neighbourhood expansion is feasible only when queries can be performed as post-processing. We conclude that a more promising and practical approach to improving memory-efficient NBNN would be to improve the algorithm for constructing binary codes. This is the focus of our second proposed extension, the bank of random rotations.

VII. Bank of random rotations: A data-adaptive approach to learning binary codes

A. Introduction

We propose a novel approach to learning binary codes that uses a *bank of random rotations* and computes Hamming

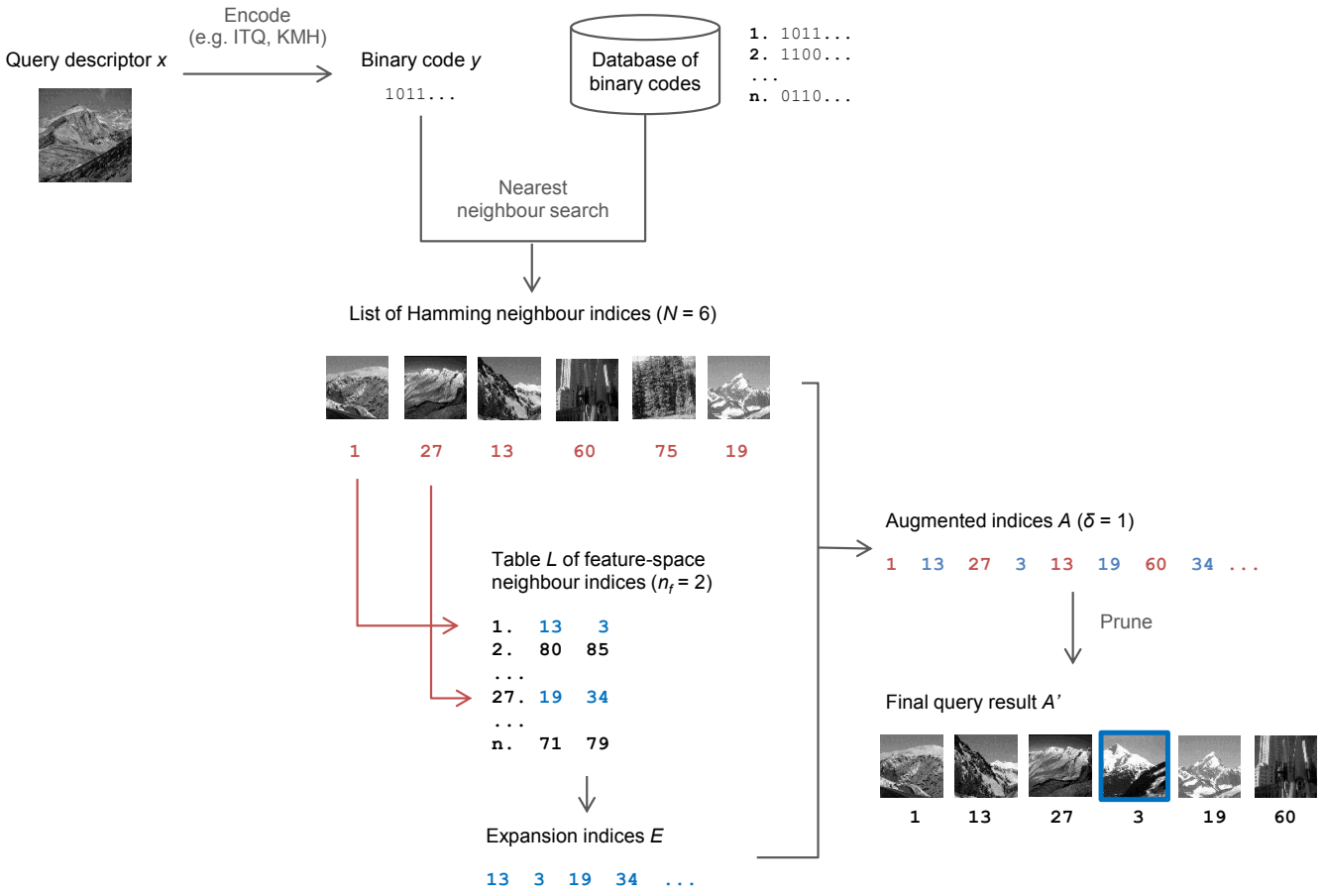


Fig. 4. Graphical summary and toy example of the neighbourhood expansion technique. Images are from the UIUC scene categorization dataset [15].

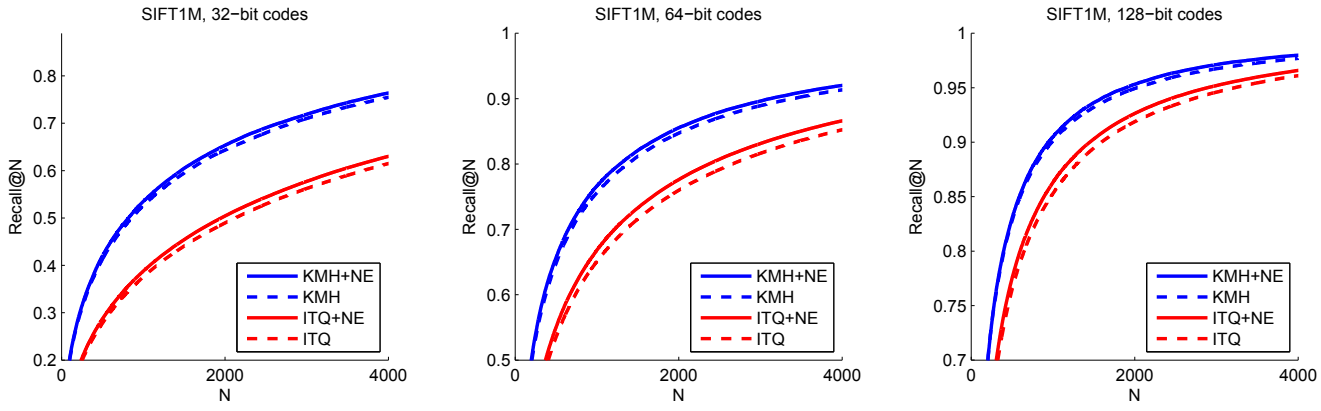
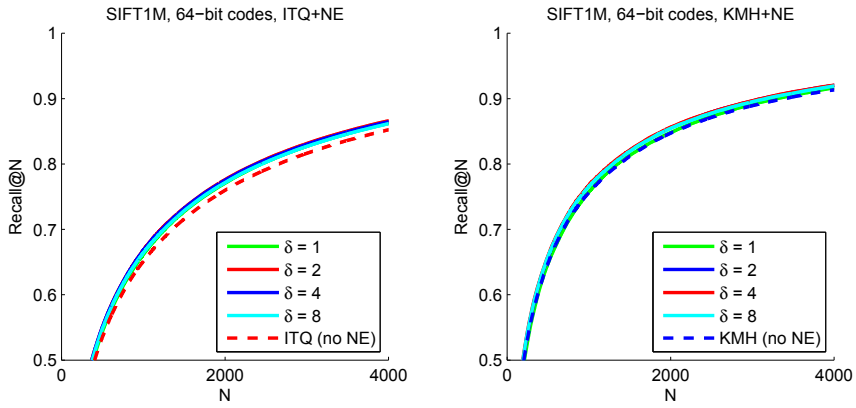
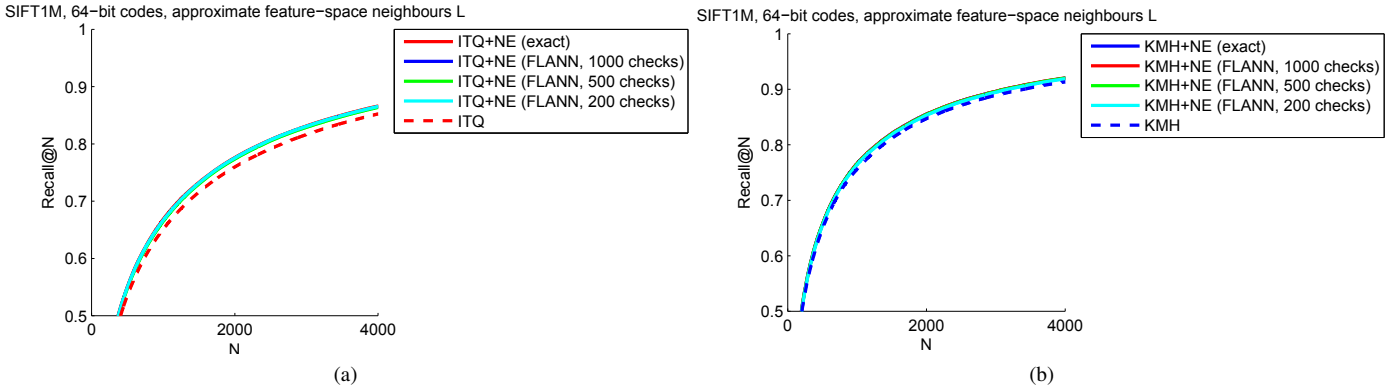


Fig. 5. Retrieval performance on SIFT1M with (a) 32-bit, (b) 64-bit, and (c) 128-bit binary codes, with and without neighbourhood expansion. For both iterative quantization and k-means hashing, neighbourhood expansion provides a modest improvement in retrieval performance.



(a) (b)

Fig. 6. Effect of varying δ on neighbourhood expansion performance, for 64-bit codes constructed by (a) iterative quantization (b) k-means hashing



(a) (b)

Fig. 7. Effect of substituting exact nearest neighbours in L with approximate nearest neighbours, for 64-bit codes constructed by (a) iterative quantization (b) k-means hashing

distances in a data-adaptive manner. Intuitively, instead of using a single globally optimal rotation as in the state-of-the-art iterative quantization method, we adaptively select the best rotation from a collection of random rotations. The proposed approach allows locality relationships in the original feature space to be more closely preserved in the reduced Hamming space.

B. Proposed technique

The problem of learning binary codes can be summarized as follows. Given a database $X \in \mathbb{R}^{n \times m}$ of n feature descriptors with dimension m, we would like to learn compact binary codes $Y \in \{0,1\}^{n \times c}$, where c is the code length in bits, such that the locality relationships in the original feature space are closely preserved in the reduced Hamming space. Given a novel query descriptor x_q, retrieval is efficiently performed by computing its binary representation y_q and comparing y_q against the database of binary codes Y.

Our approach is closely related to orthogonal hashing methods, which produce binary codes using a set of orthogonal hyperplanes. Each hyperplane splits the feature space into two partitions and corresponds to a single bit in the binary code. A data point falling on one side of the hyperplane is assigned value 0 for the bit, and a data point falling on the opposite side is assigned value 1 for the bit. Perhaps the simplest orthogonal hashing method is PCA hashing [33], [11], in which PCA is applied to the zero-centred data points and the binary codes are formed by taking the sign (i.e. thresholding at zero) of the PCA-embedded data points. In this case, the binary hyperplanes are simply the principal components. In iterative quantization (ITQ), a globally optimal rotation of the zero-centred, PCA-embedded data points is found such that the quantization error is minimized (recall Eq. 1). This quantization error minimizing rotation produces higher retrieval performance than a random rotation or no rotation as in PCA hashing, as feature-space locality relationships are better preserved.

Although ITQ finds the globally optimal rotation for minimizing the quantization error, there may be many individual database points for which the ITQ rotation is not well suited for neighbour retrieval. These are points that are mapped close to the binary hyperplanes. Figure 8 illustrates with a toy example. Suppose the globally optimal rotation is marked by the orthogonal axes in Figure 8(a). Consider the data point A. Since it is on the opposite side of the blue hyperplane as point B, points A and B will have opposite values for the bit corresponding to the blue hyperplane. Comparison of

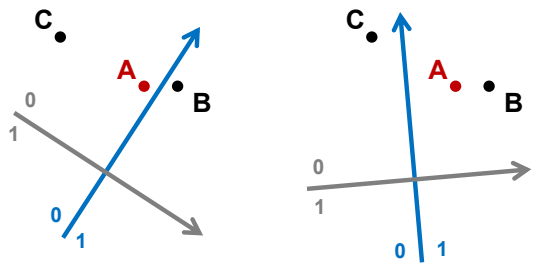


Fig. 8. Toy example illustrating the benefit of data-adaptive rotation selection when computing binary codes. Consider a particular point, A. In (a), the rotation maps neighbouring point B onto the opposite side of the blue hyperplane. The Hamming distance between A and B is a poor approximation of their feature-space distance: C is closer in Hamming space than B. From A's perspective, the alternative rotation in (b) is better for neighbour retrieval.

this bit adds 1 to the Hamming distance between A and B, even though in the original feature space A and B are in fact close neighbours. In this 2D example, B would have Hamming distance 1 from A, and C would be evaluated as closer to A with Hamming distance 0, even though A and C are farther apart in the original feature space.

Suppose instead of applying a single globally optimal rotation to all data points, we have at our disposal a collection of random rotations and can adaptively apply the rotation that is best suited for the individual data point. For example, the rotation in Figure 8(b) preserves point A's neighbour relationship with B, as both points are on the same side of the blue hyperplane. Hence, in this 2D toy example, B is correctly recognized as a closer neighbour than C, with Hamming distance 0 versus Hamming distance 1.

This is the intuition behind the bank of random rotations. We first generate a collection of random rotations $\mathbf{R} = \{R_1, R_2, ..., R_K\}$. For each zero-centred, PCA-embedded data point v_i, $i = 1...n$, we select the best random rotation in the bank, R_j. The best rotation for v_i is defined as the one that, when applied to v_i, results in the maximum sum of margins from the binary hyperplanes. We apply the rotation R_j to v_i and obtain the binary code y_i by the usual method (thresholding at zero) and store the pair (y_i, j).

Given a zero-centred, PCA-embedded query point v_q, we compute its Hamming distance to the database codes Y adaptively, using each database code's associated rotation. We can implement this as follows. For each rotation R in the bank, we obtain the binary code y_q of v_q with respect to the rotation R, and compute the Hamming distance of y_q to all database codes in Y that selected R as their rotation. Repeating this for all rotations in the bank gives the data-adaptive Hamming distances to all binary codes in the database. Figure 9 illustrates with a toy example.

We next show that these data-adaptive Hamming distances provide better preservation of the original feature-space locality relationships than the Hamming distances obtained using the single globally optimal ITQ rotation.

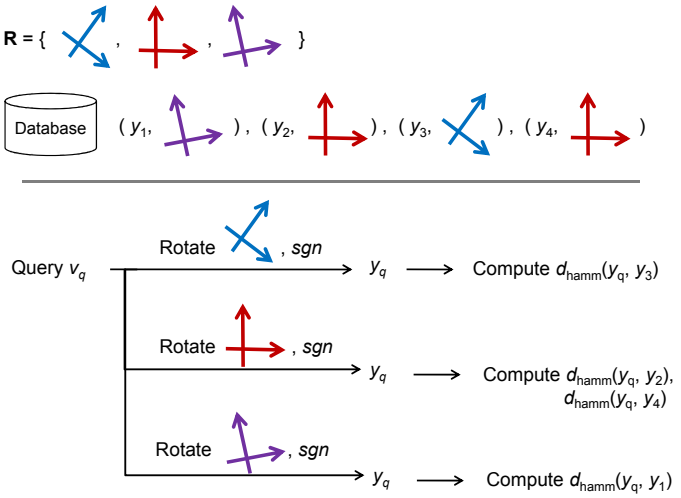


Fig. 9. Toy example illustrating the computation of data-adaptive Hamming distances using a bank of random rotations.

C. Experiments

We present preliminary experimental results on the publicly available SIFT1M [29] dataset. We compare the proposed bank of random rotations with three state-of-the-art techniques for learning binary codes: spectral hashing (SH), iterative quantization (ITQ), and k-means hashing (KMH). Similar to our evaluation of neighbourhood expansion, we show retrieval performance using recall@N curves [12], [29] and considered the ground truth to be the query's 10 nearest Euclidean neighbours.

Since we need to store the identifier of the best random rotation for each data point, for a fair comparison we adjusted the number of bits in our binary codes accordingly. For example, for comparing 64-bit codes and a bank of 4096 random rotations, we computed 52-bit codes since 12 bits are needed to index into the bank of random rotations.

Figure 10 shows the retrieval performance of 64-bit codes and 128-bit codes on the SIFT1M dataset. We observed that the proposed bank of random rotations provides higher retrieval performance than the natural baseline, iterative quantization, which uses a single globally optimal rotation. This encouraging result confirms the intuitions discussed in the previous subsection. Furthermore, the retrieval performance of the bank of random rotations is comparable to the very recently introduced k-means hashing method. Since KMH refines cluster centroids that are initialized using PCA with zero rotation, it would be interesting as future work to investigate whether the general principle of data-adaptive rotations can be extended to KMH to further improve performance. Qualitatively, the bank of random rotations is a computationally simpler approach that does not require any sequential optimization or product-space decomposition.

We found the proposed approach to be robust to the size of the bank of random rotations. Figure 11 shows the retrieval performance over different bank sizes K for 64-bit and 128-bit experiment settings on SIFT1M. Figure 10 is generated with $K = 4096$ random rotations.

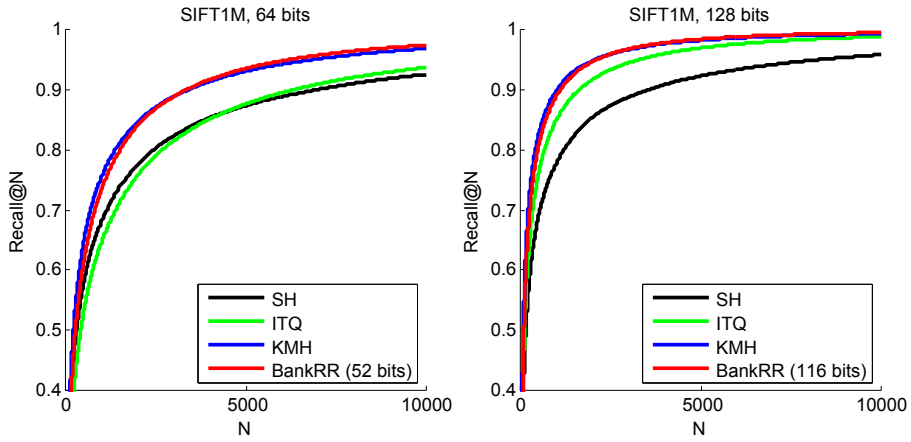
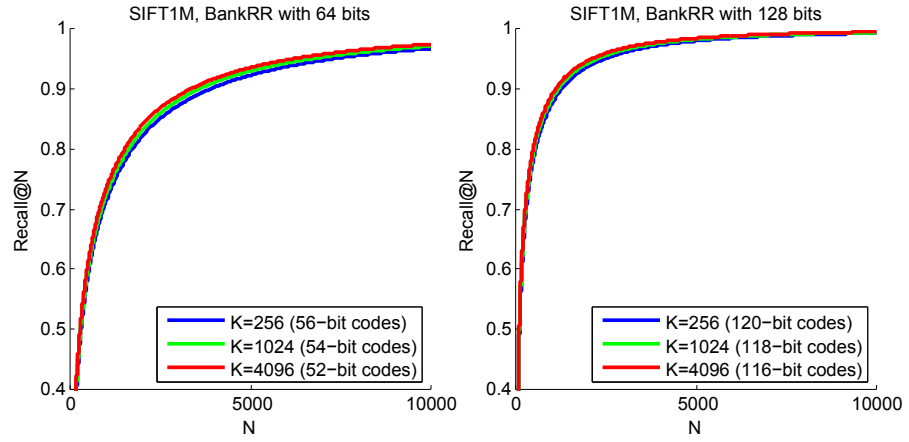


Fig. 10. Retrieval performance for SIFT1M with (a) 64-bit and (b) 128-bit binary codes (SH: Spectral hashing, ITQ: Iterative quantization, KMH: K-means hashing, BankRR: Bank of random rotations). For a fair comparison the BankRR binary code length is reduced to account for the bits needed to index into the bank of random rotations.



(a) (b)

Fig. 11. Effect of varying the size of the bank of random rotations (i.e. the number of random rotations in the bank) for (a) 64-bit and (b) 128-bit binary codes.

In summary, our preliminary experimental results show that the proposed bank of rotations approach provides improved retrieval performance compared with the natural baseline of iterative quantization, which uses a single globally optimal rotation. The proposed approach also performs well in comparison with other state-of-the-art methods for learning binary codes, such as the recently introduced k-means hashing method.

VIII. Conclusion

Although NBNN image classification requires no training phase, scales naturally to a large number of classes, and generalizes well by "composing pieces" using image-to-class distance, it can also be prohibitively expensive in terms of memory requirements. Our proposed memory-efficient NBNN image classification algorithm applies compact binary coding to obtain several-fold gains in memory efficiency at a moderate cost in classification accuracy. We evaluated spectral hashing, iterative quantization, and k-means hashing methods for constructing compact binary codes, and integrated these into the modern Local NBNN framework. In addition, we explored two extensions to improve the accuracy of memory-efficient NBNN image classification, and presented preliminary experimental results for both.

Acknowledgment

This research is sponsored in part by the Natural Sciences and Engineering Research Council of Canada.

References

[1] N. Dalal and B. Triggs, "Histograms of oriented gradients for human detection," in *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, vol. 1, 2005, pp. 886–893.

[2] P. Felzenszwalb, R. Girshick, D. McAllester, and D. Ramanan, "Object detection with discriminatively trained part-based models," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 32, no. 9, pp. 1627–1645, 2010.

[3] V. Ferrari, L. Fevrier, F. Jurie, and C. Schmid, "Groups of adjacent contour segments for object detection," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 30, no. 1, pp. 36–51, 2008.

[4] A. Opelt, A. Pinz, and A. Zisserman, "A boundary-fragment-model for object detection," in *Proc. European Conference on Computer Vision*, 2006, pp. 575–588.

[5] T. Deselaers and V. Ferrari, "Global and efficient self-similarity for object classification and detection," in *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, 2010, pp. 1633–1640.

[6] E. Shechtman and M. Irani, "Matching local self-similarities across images and videos," in *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, 2007.

[7] O. Boiman, E. Shechtman, and M. Irani, "In defense of nearest-neighbor based image classification," in *Proc. IEEE Conference on Computer Vision and Pattern Recognition*, 2008.

[8] G. Csurka, C. R. Dance, L. Fan, J. Williamowski, and C. Bray, "Visual categorization with bags of keypoints," in *Proc. ECCV Workshop on Statistical Learning in Computer Vision*, 2004.

[9] S. McCann and D. Lowe, "Local Naive Bayes Nearest Neighbor for image classification," in *Proc. IEEE Conference on Computer Vision and Pattern Recognition*, 2012, pp. 3650–3656.

[10] Y. Weiss, A. Torralba, and R. Fergus, "Spectral hashing," in *Advances in Neural Information Processing Systems*, 2008.

[11] Y. Gong and S. Lazebnik, "Iterative quantization: a Procrustean approach to learning binary codes," in *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, 2011, pp. 817–824.

[12] K. He, F. Wen, and J. Sun, "K-means hashing: an affinity-preserving quantization method for learning binary compact codes," in *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, 2013.

[13] L. Fei-Fei, R. Fergus, and P. Perona, "Learning generative visual models from few training examples: an incremental Bayesian approach tested on 101 object categories," in *IEEE CVPR Workshop on Generative-Model Based Vision*, 2004.

[14] G. Griffin, A. Holub, and P. Perona, "Caltech-256 object category dataset," California Institute of Technology, Tech. Rep., 2007.

[15] S. Lazebnik, C. Schmid, and J. Ponce, "Beyond bags of features: spatial pyramid matching for recognizing natural scene categories," in *Proc. IEEE Conference on Computer Vision and Pattern Recognition*, vol. 2, 2006, pp. 2169–2178.

[16] A. Bosch, A. Zisserman, and X. Muñoz, "Image classification using random forests and ferns," in *Proc. IEEE International Conference on Computer Vision*, 2007.

[17] V. Ferrari, F. Jurie, and C. Schmid, "From images to shape models for object detection," *International Journal of Computer Vision*, vol. 87, no. 3, pp. 284–303, 2010.

[18] F. Perronnin and C. Dance, "Fisher kernels on visual vocabularies for image categorization," in *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, 2007.

[19] F. Perronnin, J. Sánchez, and T. Mensink, "Improving the Fisher kernel for large-scale image classification," in *Proc. European Conference on Computer Vision*, 2010, pp. 143–156.

[20] H. Jégou, M. Douze, C. Schmid, and P. Pérez, "Aggregating local descriptors into a compact image representation," in *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, 2010, pp. 3304–3311.

[21] C. Kanan and G. Cottrell, "Robust classification of objects, faces, and flowers using natural image statistics," in *Proc. IEEE Conference on Computer Vision and Pattern Recognition*, 2010, pp. 2472–2479.

[22] R. Timofte, T. Tuytelaars, and L. V. Gool, "Naive Bayes image classification: beyond nearest neighbors," in *Proc. Asian Conference on Computer Vision*, 2012, pp. 689–703.

[23] A. Gionis, P. Indyk, and R. Motwani, "Similarity search in high dimensions via hashing," in *Proc. International Conference on Very Large Data Bases*, 1999, pp. 518–529.

[24] P. Indyk and R. Motwani, "Approximate nearest neighbor: towards removing the curse of dimensionality," in *ACM Symposium on Theory of Computing*, 1998, pp. 604–613.

[25] A. Torralba, R. Fergus, and Y. Weiss, "Small codes and large image databases for recognition," in *Proc. IEEE Conference on Computer Vision and Pattern Recognition*, 2008.

[26] A. Oliva and A. Torralba, "Modeling the shape of the scene: a holistic representation of the spatial envelope," *International Journal of Computer Vision*, vol. 42, no. 3, pp. 145–175, 2001.

[27] G. Shakhnarovich, P. Viola, and T. Darrell, "Fast pose estimation with parameter-sensitive hashing," in *Proc. IEEE International Conference on Computer Vision*, 2003, pp. 750–757.

[28] R. Salakhutdinov and G. Hinton, "Semantic hashing," in *Proc. SIGIR workshop on information retrieval and applications of graphical models*, 2007.

[29] 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.

[30] C. Silpa-Anan and R. Hartley, "Optimised KD-trees for fast image descriptor matching," in *Proc. IEEE Conference on Computer Vision and Pattern Recognition*, 2008, pp. 3650–3656.

[31] M. Muja and D. Lowe, "Fast approximate nearest neighbors with automatic algorithm configuration," in *Proc. International Conference on Computer Vision Theory and Applications*, 2009.

[32] D. Lowe, "Distinctive image features from scale-invariant keypoints," *International Journal of Computer Vision*, vol. 60, no. 2, pp. 91–110, 2004.

[33] J. Wang, S. Kumar, and S.-F. Chang, "Semi-supervised hashing for scalable image retrieval," in *Proc. IEEE Conference on Computer Vision and Pattern Recognition*, 2010.