This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication.

Iterative Quantization: A Procrustean Approach to Learning Binary Codes for Large-scale Image Retrieval

Yunchao Gong, Svetlana Lazebnik, Albert Gordo, Florent Perronnin

Abstract—This paper addresses the problem of learning similarity-preserving binary codes for efficient similarity search in large-scale image collections. We formulate this problem in terms of finding a rotation of zero-centered data so as to minimize the quantization error of mapping this data to the vertices of a zero-centered binary hypercube, and propose a simple and efficient alternating minimization algorithm to accomplish this task. This algorithm, dubbed iterative quantization (ITQ), has connections to multi-class spectral clustering and to the orthogonal Procrustes problem, and it can be used both with unsupervised data embeddings such as PCA and supervised embeddings such as canonical correlation analysis (CCA). The resulting binary codes significantly outperform several other state-of-the-art methods. We also show that further performance improvements can result from transforming the data with a nonlinear kernel mapping prior to PCA or CCA. Finally, we demonstrate an application of ITQ to learning binary attributes or "classemes" on the ImageNet dataset.

Index Terms—Large-scale image search, binary codes, hashing, quantization.

1 Introduction

Recently, the vision community has devoted a lot of attention to the problem of learning similarity-preserving binary codes for representing large-scale image collections [4], [5], [13], [15], [20], [22], [23], [29], [31], [36], [39], [43], [45], [49], [50], [51]. An effective scheme for learning binary codes should have several properties. First, the codes should be short to enable the storage of large amounts of images in memory. For example, for an ordinary workstation with 16GB of RAM, to fit 250 million images in memory, we could only use about 64 bits per image. Second, the codes should map images that are similar (either in terms of feature space distance or semantic distance) to binary strings with a low Hamming distance. Finally, the algorithms for learning the parameters of the binary code and for encoding a new test image should be efficient and scalable.

A common initial step in many binary coding methods is to perform principal component analysis (PCA) to ensure good generalization ability for small code sizes [14], [49], [51]. However, since the variance of the data in each PCA direction is different – in particular, higher-variance directions carry much more information – encoding each direction with the same number of bits is bound to produce poor performance. To deal with this issue, spectral hashing (SH) [51] uses a separable Laplacian eigenfunction formulation that ends up assigning more bits to directions along which the data has a greater range. However, this approach is somewhat heuristic and relies on an assumption that the data is uniformly distributed in a high-dimensional rectangle. Semi-supervised hashing (SSH) [49] relaxes the orthogonality constraints of PCA to allow successive projection directions to capture more of the data variance. While this approach produces promising results, the optimization problem requires careful regularization to avoid degenerate solutions.

In this work, we start with PCA-projected data and formulate the problem of learning a good binary code in terms of directly minimizing the quantization error of mapping this data to vertices of the binary hypercube. Figure 1 illustrates the key steps of our approach. First, we show that simply applying a random orthogonal transformation, as suggested by Jégou et al. [19], already does a very good job of balancing the variance of different PCA directions and outperforms both SH [51] and non-orthogonal relaxation [49]. Next, we propose an alternating minimization approach for refining the initial orthogonal transformation to reduce quantization error. This approach, dubbed **iterative quantization (ITQ)**, is described in Section 3. It has connections to the orthogonal Procrustes problem [41] and to eigenvector discretization for multi-class spectral partitioning [52], and in the experiments of Section 4 it outperforms the methods

- *Y. Gong is with the computer science department, University of North Carolina at Chapel Hill, USA. E-mail: yunchao@cs.unc.edu. Code is available at: http://www.unc.edu/~yunchao/itq.htm.*
- *S. Lazebnik is with the computer science department, University of Illinois at Urbana-Champaign, USA. E-mail: slazebni@uiuc.edu.*
- *A. Gordo is with computer vision center, Universitat Autonoma de Barcelona, Spain. E-mail: agordo@cvc.uab.es.*
- *F. Perronnin is with Textual Visual Pattern Analysis, Xerox Research Centre Europe, France. E-mail: florent.perronnin@xrce.xerox.com.*

Digital Object Indentifier 10.1109/TPAMI.2012.193 0162-8828/12/$31.00 © 2012 IEEE

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication.

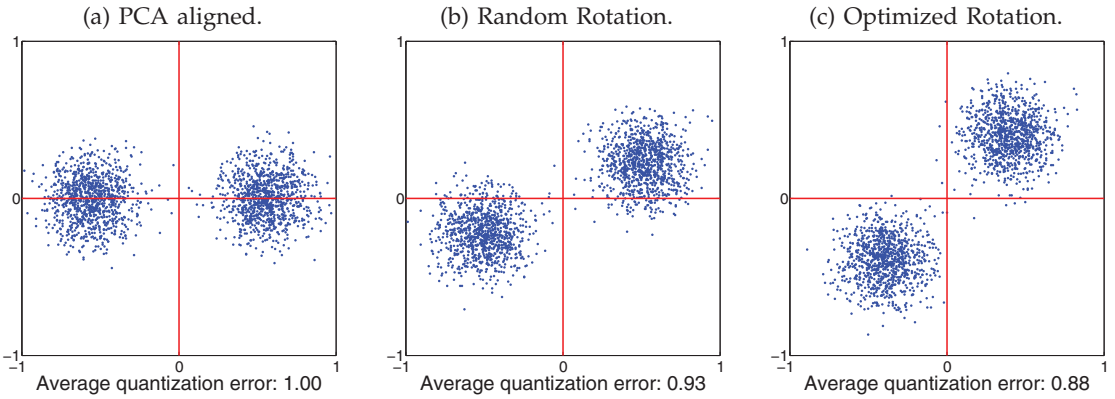


Figure 1. Toy illustration of our ITQ method (see Section 3 for details). The basic binary encoding scheme is to quantize each data point to the closest vertex of the binary cube, $(\pm 1, \pm 1)$ (this is equivalent to quantizing points according to their quadrant). (a) The x and y axes correspond to the PCA directions of the data. Note that quantization assigns points in the same cluster to different vertices. (b) Randomly rotated data – the variance is more balanced and the quantization error is lower. (c) Optimized rotation found by ITQ – quantization error is lowest, and the partitioning respects the cluster structure.

of [36], [49], [51]. Moreover, ITQ does not need to be coupled with PCA, but can be used with many other dimensionality reducing embeddings. In Section 5, we show how to combine ITQ with canonical correlation analysis (CCA) to incorporate information from clean or noisy class labels in order to improve the semantic consistency of the code. Section 6 shows that further improvements can be achieved by transforming the data using a randomized nonlinear embedding [37] that approximates the Gaussian kernel. As an additional application, Section 7 demonstrates the usefulness of ITQ for learning binary visual attributes or "classemes" [46] on the ImageNet dataset [8]. We show the resulted binary descriptor is discriminative, and can be effectively used for object category retrieval.

This paper is an extended version of the work initially published in CVPR 2011 [13]. Novel contributions over [13] include the nonlinear kernel extension (Section 6) and the application to object categorization and retrieval (Section 7).

2 Related Work

Nearest neighbor search is a fundamental operation underlying many computer vision approaches. Brute-force search, or comparing a query point with every point in the database, becomes prohibitively expensive as the dataset size and the dimensionality of the features grows. To reduce search complexity, a number of algorithms and data structures have been proposed [42]. These methods, such as hierarchical feature space decompositions and locality sensitive hashing (LSH), usually aim to provide sub-linear running time for approximate near-neighbor lookup. By contrast, similarity-preserving binary codes do not necessarily offer sub-linear search complexity. However, they can reduce memory requirements and the running time of brute-force search by significant constant factors, and they do not involve building and maintaining complex data structures.

The first step of computing binary codes usually involves finding an intermediate continuous embedding of the original data. For example, a popular hash function for LSH that preserves dot-product similarity uses random projections drawn from a Gaussian distribution [1]. Kulis et al. [22] generalize this LSH formulation to the setting where the similarity is given by a "black-box" kernel function. Raginsky and Lazebnik [36] use random Fourier features [37] to approximate the Gaussian kernel. All these methods are based on randomized embeddings, which tend to be too noisy for a small number of bits. Another family of methods uses dimensionality reduction prior to binary coding, with PCA being the most common choice [13], [14], [18], [49], [50], [51]. Beyond PCA, other embeddings can be applied. For example, Liu et al. [24] use a spectral embedding, He et al. [15] use a method similar to Independent Component Analysis (ICA), Strecha et al. [43] use Linear Discriminant Analysis (LDA). Most recently, Liu et al. [25] have developed a principled formulation for learning hash codes using kernels and category labels.

After the embedding, the next step is to binarize the data. For randomized embeddings, this can simply be done by appropriately thresholding the projected data [1], [22], [36]. For PCA and related techniques, a more complex binarization scheme is necessary to alleviate the problem of unbalanced variance discussed in the Introduction. The works by Wang et al. [49], [50] use non-orthogonal relaxation or sequential projections. Spectral Hashing [51] allocates bits based on separable Laplacian eigenfunctions. In this paper, we propose a simpler idea of rotating the data in order to balance the variance. We have been inspired by the

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication.

approach of Yu and Shi [52] for discretizing relaxed solutions to multi-class spectral clustering, where an orthogonal transformation is applied to the continuous eigenvectors to bring them as close as possible to a discrete solution. One important difference between [52] and our approach is that [52] allows discretization only to the c orthogonal hypercube vertices with exactly one positive entry, while we use all the 2^c vertices as targets.

At retrieval time, given a new query image, it is necessary to compute the distances from that query to every image in the database. Most methods directly compute the Hamming distance between the respective binary codes, which can be done very efficiently using low-level hardware operations. An alternative to this is given by asymmetric distance [14], [18], in which the database points are quantized but the query is not. In this paper, we use only the Hamming distance. If nearest-neighbor lookup by linear scan is not satisfactory, it is possible to treat the binary strings representing images as hash keys [1], [22]. Note that this is only feasible with very compact codes, i.e., 32 bits at most. Section 4.5 will look at this scenario and evaluate the performance of binary codes for hashing.

Besides similarity-preserving binary codes, other representations have been proposed for efficient large-scale image retrieval. One of these is Product Quantization (PQ) [18], [19], where the feature space is decomposed into a Cartesian product of low-dimensional subspaces, each subspace is quantized separately, and asymmetric distance is computed between the query and the quantized codes with the help of lookup tables. Other state-of-the-art methods include min-hashing for bags of features [7] and vocabulary trees [28]. Though such schemes are capable of achieving very high performance for tasks such as near-duplicate detection and object instance retrieval, binary coding schemes remain attractive due to their flexibility. They do not require building of hash tables or inverted indices, and they can be compared using simple Hamming distance. As a consequence of this, binary codes can easily support other dataset operations besides near-neighbor retrieval. For example, clustering of binary codes can be implemented using a standard k-medoids algorithm [12], while clustering in the min-hash framework is much less straightforward [7]. As another example, in Section 7 we will train linear SVM classifiers directly on top of similarity-preserving binary codes (see also [33]), whereas this would be impossible to do with PQ-compressed visual features without decoding them first.

3 Unsupervised Code Learning

In this section, we address the problem of learning binary codes without any supervisory information (e.g., class labels or tags). We first apply linear dimensionality reduction to the data, and then perform binary quantization in the resulting space. For the first step, discussed in Section 3.1, we follow the maximum variance formulation of [49], [51], which yields PCA projections. The major novelty of our method is in the second step (Section 3.2), where we try to preserve the locality structure of the projected data by rotating it so as to minimize the quantization error.

Let us first introduce our notation. We have a set of n data points $\{\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_n\}$, $\boldsymbol{x}_i \in \mathbb{R}^d$, that form the rows of the data matrix $X \in \mathbb{R}^{n \times d}$. We assume that the points are zero-centered, i.e., $\sum_{i=1}^{n} \boldsymbol{x}_i = 0$. Our goal is to learn a binary code matrix $B \in \{-1, 1\}^{n \times c}$, where c denotes the code length.[1] For each bit $k = 1, \ldots, c$, the binary encoding function is defined by $h_k(\boldsymbol{x}) = \operatorname{sgn}(\boldsymbol{x}\boldsymbol{w}_k)$, where $\boldsymbol{w}_k$ is a column vector of hyperplane coefficients and

$$\operatorname{sgn}(v) = \begin{cases} 1, & \text{if } v \geq 0; \\ -1, & \text{otherwise.} \end{cases}$$

For a matrix or a vector, $\operatorname{sgn}(\cdot)$ will denote the result of element-wise application of the above function. Thus, we can write the entire encoding process as $B = \operatorname{sgn}(XW)$, where $W \in \mathbb{R}^{d \times c}$ is the matrix with columns $\boldsymbol{w}_k$.

3.1 Unsupervised PCA Embedding

Following the formulation of [49], [50], [51], we want to produce an efficient code in which the variance of each bit is maximized and the bits are pairwise uncorrelated. We can do this by maximizing the following objective function:

$$\begin{aligned} \mathcal{I}(W) &= \sum_k \operatorname{var}(h_k(\boldsymbol{x})) = \sum_k \operatorname{var}(\operatorname{sgn}(\boldsymbol{x}\boldsymbol{w}_k)) \,, \\ & \frac{1}{n} B^T B = I \,. \end{aligned}$$

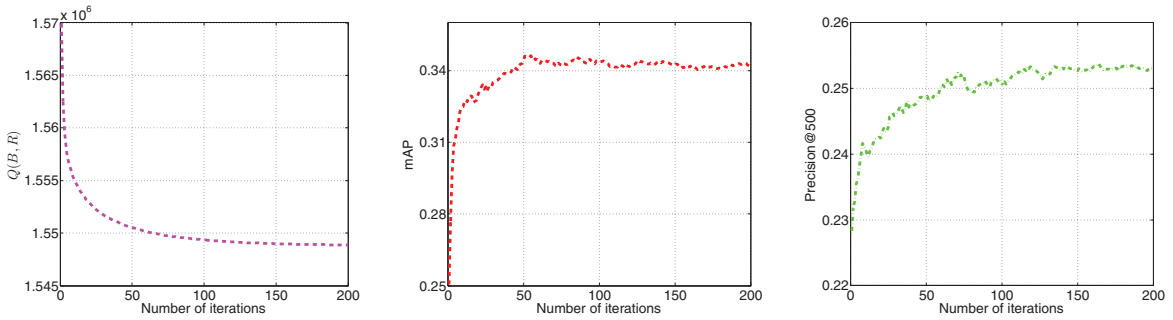
As shown in [49], the variance is maximized by encoding functions that produce exactly balanced bits, i.e., when $h_k(\boldsymbol{x}) = 1$ for exactly half of the data points and -1 for the other half. However, the requirement of exact balancedness makes the above objective function intractable. Adopting the same signed magnitude relaxation as in [49], we get the following continuous objective function:

$$\begin{aligned} \widetilde{\mathcal{I}}(W) &= \sum_k \mathbb{E}(\|\boldsymbol{x}\boldsymbol{w}_k\|_2^2) = \frac{1}{n} \sum_k \boldsymbol{w}_k^T X^T X \boldsymbol{w}_k \\ &= \frac{1}{n} \operatorname{tr}(W^T X^T X W) \,, \quad W^T W = I \,. \end{aligned} \tag{1}$$

The constraint $W^T W = I$ requires the hashing hyperplanes to be orthogonal to each other, which is a relaxed version of the requirement that code bits be pairwise decorrelated. This objective function is

1. In our formulation, the entries of B take on values $\{-1, 1\}$ instead of $\{0, 1\}$ because the proposed quantization-based scheme of Section 3.2 requires both the data and the binary cube to be zero-centered.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication.



(a) Objective function value (b) Euclidean neighbor precision (c) Class label precision

Figure 2. Performance of ITQ (32 bits) as a function of the number of iterations on the CIFAR dataset (refer to Section 4 for details of the dataset and evaluation protocol). (a) Objective function value vs. the number of iterations; (b) Mean average precision (mAP) of Euclidean neighbor retrieval vs. the number of iterations; (c) Average class label precision at top 500 retrieved images vs. the number of iterations.

exactly the same as that of PCA. For a code of c bits, we obtain W by taking the top c eigenvectors of the data covariance matrix $X^T X$.

3.2 Iterative Quantization

Let $\boldsymbol{v} \in \mathbb{R}^c$ be a vector in the projected space. It is easy to show (see below) that $\mathrm{sgn}(\boldsymbol{v})$ is the vertex of the hypercube $\{-1, 1\}^c$ closest to $\boldsymbol{v}$ in terms of Euclidean distance. The smaller the quantization loss $\|\mathrm{sgn}(\boldsymbol{v}) - \boldsymbol{v}\|^2$, the better the resulting binary code will preserve the original locality structure of the data. Now, going back to eq. (1), it is clear that if W is an optimal solution, then so is $\widetilde{W} = WR$ for any orthogonal $c \times c$ matrix R. Therefore, we are free to orthogonally transform the projected data $V = XW$ in such a way as to minimize the quantization loss

$$Q(B, R) = \|B - VR\|_F^2, \qquad (2)$$

where $\|\cdot\|_F$ denotes the Frobenius norm.

The idea of rotating the data to minimize quantization loss can be found in Jégou et al. [17], [19]. However, the approach of [17] does not include a dimensionality reduction step, and the approach of [19] is based not on binary codes, but on product quantization with asymmetric distance computation. Unlike in our formulation, direct minimization of quantization loss for asymmetric distance is impractical, so Jégou et al. instead suggest solving an easier problem, that of finding a rotation (or, more precisely, an orthogonal transformation) to balance the variance of the different dimensions of the data. In practice, they find that a random rotation works well. Based on this observation, a natural baseline for our method is given by initializing R to a random orthogonal matrix.

Beginning with the random initialization of R, we adopt a k-means-like procedure that we call ITQ to find a local minimum of the quantization loss (2). In each iteration, each data point is first assigned to the nearest vertex of the binary hypercube, and then R is updated to minimize the quantization loss given this assignment. These two alternating steps are described in detail below.

Fix R and update B. Expanding (2), we have

$$\begin{aligned} Q(B, R) &= \|B\|_F^2 + \|V\|_F^2 - 2\,\mathrm{tr}(BR^T V^T) \\ &= nc + \|V\|_F^2 - 2\,\mathrm{tr}(BR^T V^T)\,. \end{aligned} \qquad (3)$$

Because the projected data matrix $V = XW$ is fixed, minimizing (3) is equivalent to maximizing

$$\mathrm{tr}(BR^T V^T) = \sum_{i=1}^{n} \sum_{j=1}^{c} B_{ij} \widetilde{V}_{ij},$$

where $\widetilde{V}_{ij}$ denote the elements of $\widetilde{V} = VR$. To maximize this expression with respect to B, we need to have $B_{ij} = 1$ whenever $\widetilde{V}_{ij} \geq 0$ and -1 otherwise. In other words, $B = \mathrm{sgn}(VR)$ as claimed in the beginning of this section.

Note that scaling the original data X by a constant factor changes the additive and multiplicative constants in (3), but does not affect the optimal value of B or R. Thus, while our method requires the data to be zero-centered, it does not care at all about the scaling. In other words, the quantization formulation (2) makes sense regardless of whether the average magnitude of the feature vectors matches the radius of the binary cube.

Fix B and update R. For a fixed B, the objective function (2) corresponds to the classic Orthogonal Procrustes problem [41], in which one tries to find a rotation to align one point set with another. In our case, the two point sets are given by the projected data V and the target binary code matrix B. For a fixed B, (2) is minimized as follows: first compute the SVD of the $c \times c$ matrix $B^T V$ as $B^T V = S\Omega\hat{S}^T$ and then let $R = \hat{S}S^T$.

We alternate between updates to B and R for several iterations to find a locally optimal solution. The convergence is guaranteed as the global optimal solution of each subproblem can be achieved, thus each

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication.

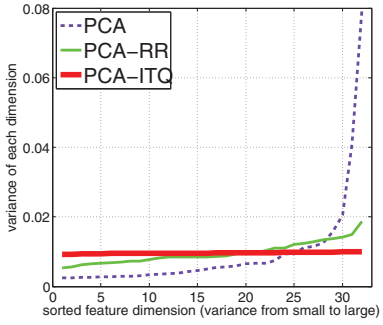


Figure 3. Variance of different dimensions for PCA, random rotation (RR), and ITQ using a 32-bit code.

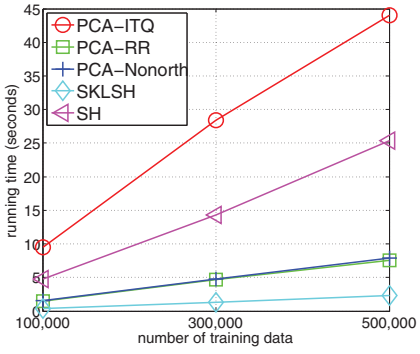


Figure 4. Training time for different binary encoding methods with 32 bits. Refer to section 4.2 for the details of different baseline methods.

step will never increase the objective function value. Figure 2 shows the typical behavior of the objective function and of our two operational criteria, precision for Euclidean neighbor retrieval and for class label retrieval, as a function of the number of iterations. In practice, we have found that we do not need to iterate until convergence to get good results, so we use 50 iterations in all our subsequent experiments. Figure 3 plots the variance of each dimension before and after ITQ. It is clear that ITQ leads to the most balanced variances. Finally, Figure 4 shows the training times for 32-bits codes for ITQ and competing baselines described in Section 4.2. All the methods scale linearly with the number of images, and although ours has the slowest training time, it is still very practical in absolute terms.

4 Evaluating Unsupervised ITQ

In this section, we evaluate the basic unsupervised ITQ framework introduced in Section 3. Once the promise of this framework has been established, we will then go on to describe and evaluate its supervised extension (Section 5), nonlinear kernel extension (Section 6), and application to object categorization and retrieval (Section 7).

4.1 Datasets

We evaluate ITQ on two subsets of the Tiny Images dataset [44], both of which come from [10]. The first one is a version of the CIFAR dataset [21], and it consists of 64,185 images that have been manually grouped into 11 ground-truth classes: airplane, automobile, bird, boat, cat, deer, dog, frog, horse, ship and truck. The second subset consists of 580,000 Tiny Images. Apart from the CIFAR images, which are included in the larger subset, all the other images lack manually supplied ground truth labels, but they come associated with one of 388 Internet search keywords. In this section, we use the CIFAR ground-truth labels to evaluate the semantic consistency of our codes, and in Section 5, we will use the "noisy" keyword information associated with the remaining Tiny Images to train a supervised linear embedding.

The original Tiny Images are 32×32 pixels. We represent them with grayscale GIST descriptors [30] computed at three different scales $(8, 8, 4)$, resulting in 320-dimensional feature vectors. Because ITQ cannot use more bits than the original dimension of the data, in this section we evaluate code sizes up to 256 bits. However, in Section 6 we will show an extension that will allow us to use more bits than data dimensions.

4.2 Protocols and Baseline Methods

We follow two evaluation protocols widely used in recent papers [36], [49], [51]. The first one is to evaluate performance of nearest neighbor search using Euclidean neighbors as ground truth. As in [36], a nominal threshold of the average distance to the 50th nearest neighbor is used to determine whether a database point returned for a given query is considered a true positive. Then, based on the Euclidean ground truth, we compute the recall-precision curve and the mean average precision (mAP), or the area under the recall-precision curve. Second, we evaluate the semantic consistency of codes produced by different methods by using class labels as ground truth. For this case, we report the averaged precision of top 500 ranked images for each query as in [50] (note that the CIFAR dataset has about 6,000 images per class). For all experiments, we randomly select 1000 points to serve as test queries. The remaining images form the training set on which the code parameters are learned, as well as the database against which the queries are performed. All the experiments reported in this paper are averaged over five random training/test partitions.

We compare our ITQ method to three baseline methods that follow the basic hashing scheme $B = \mathrm{sgn}(X\widetilde{W})$, where the projection matrix $\widetilde{W}$ is defined in different ways:

1) **LSH:** $\widetilde{W}$ is a Gaussian random matrix [1]. Note that in theory, this scheme has locality preserving guarantees only for unit-norm vectors. While

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication.

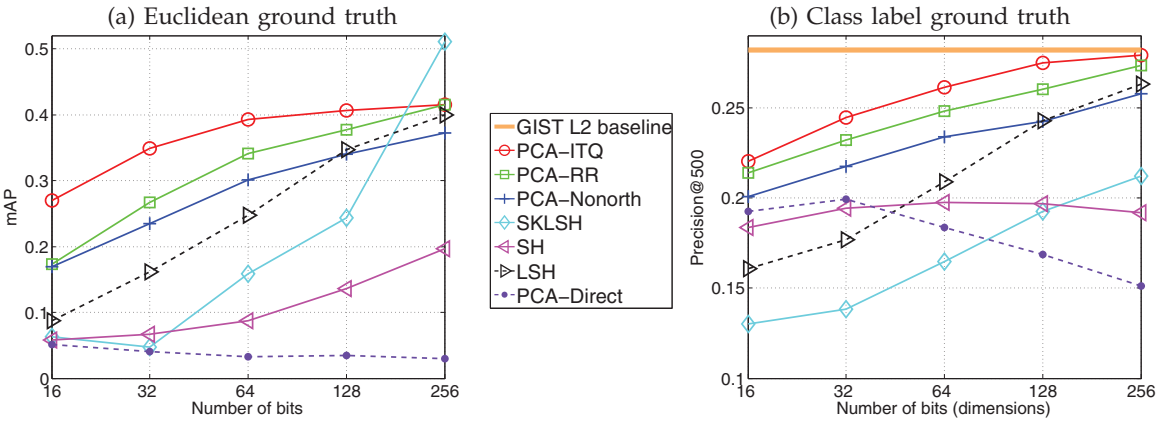
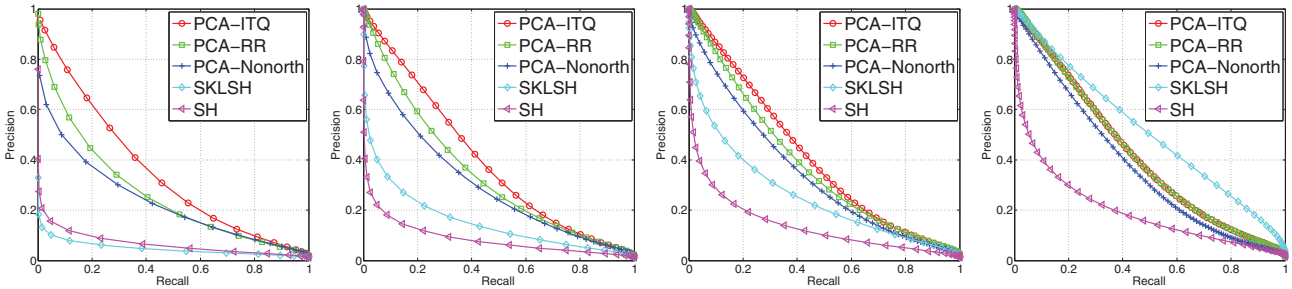
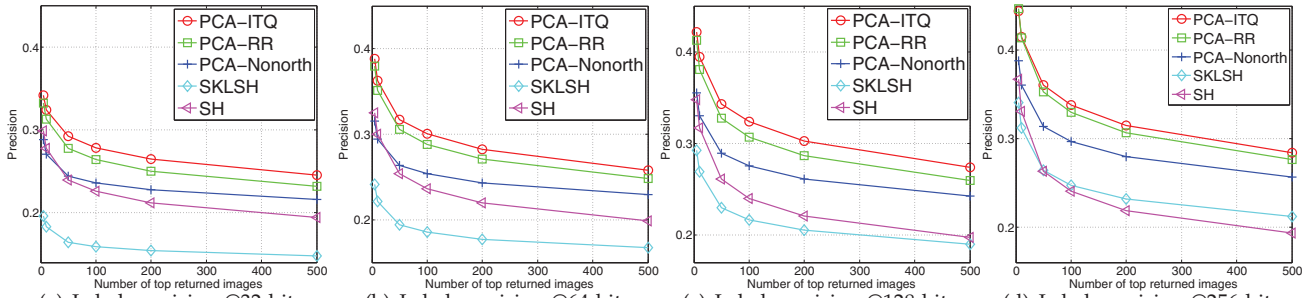


Figure 5. Comparative evaluation on CIFAR dataset. (a) Performance is measured by mean average precision (mAP) for retrieval of Euclidean neighbors within a target radius (see text). Refer to Figure 6 for the complete recall-precision curves for the state-of-the-art methods. (b) Performance is measured by the averaged precision of top 500 ranked images for each query where ground truth is defined by semantic class labels. Refer to Figure 7 for the complete class label precision curves for the state-of-the-art methods.



(a) Recall precision @32 bits. (b) Recall precision @64 bits. (c) Recall precision @128 bits. (d) Recall precision @256 bits.

Figure 6. Comparison with state-of-the-art methods on CIFAR dataset using Euclidean neighbors as ground truth. Refer to Figure 5(a) for a summary of the mAP of these curves as a function of code size.



(a) Label precision @32 bits. (b) Label precision @64 bits. (c) Label precision @128 bits. (d) Label precision @256 bits.

Figure 7. Comparison with state-of-the-art methods on CIFAR dataset using semantic labels as ground truth. Figure 5(b) shows the summary plot of average precision as a function of code size.

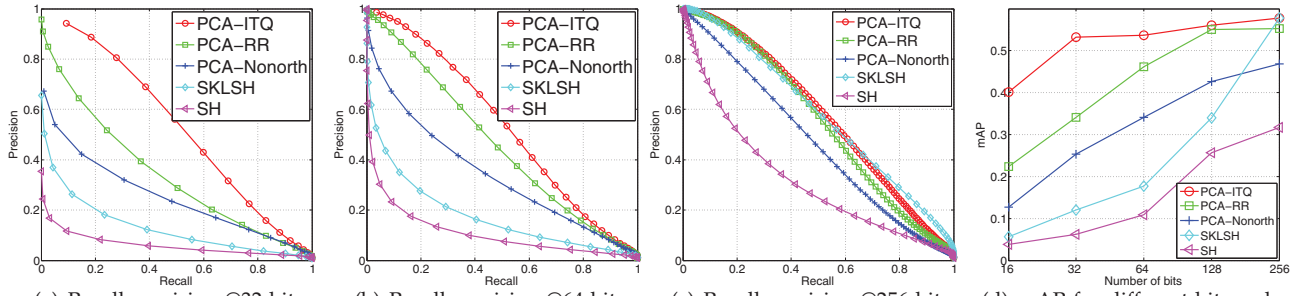
we do not normalize our data to unit norm, we have found that it still works well as long as the data is zero centered.

2) **PCA-Direct:** $\widetilde{W}$ is simply the matrix of top c PCA directions. This baseline is included to show what happens when we do not rotate the PCA-projected data prior to quantization.
3) **PCA-RR:** $\widetilde{W} = WR$, where W is the matrix of PCA directions and R is a random orthogonal matrix. This is the initialization of ITQ, as described in Section 3.

We also compare ITQ to three state-of-the-art methods using code provided by the authors:

1) **SH** [51]: Spectral Hashing. This method is based on quantizing the values of analytical eigenfunctions computed along PCA directions of the data.
2) **SKLSH** [36]: This method is based on random Fourier features for approximating the Gaussian kernel [37]. In [36], this method is reported to outperform SH for code sizes larger than 64 bits. We use a Gaussian kernel with bandwidth set to

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication.



(a) Recall precision @32 bits. (b) Recall precision @64 bits. (c) Recall precision @256 bits. (d) mAP for different bits code.

Figure 8. Results on the 580,000 Tiny Image subset. Ground truth is defined by Euclidean neighbors.

the average distance to the 50th nearest neighbor as in [36].

3) **PCA-Nonorth** [49]: Non-orthogonal relaxation of PCA. This method is reported in [49] to outperform SH. Note that instead of using semi-supervised PCA as in [49], the evaluation of this section uses standard unsupervised PCA since we assume there is no class label information available (although label information will be used in Section 5).

Note that of all the six methods above, LSH and SKLSH are the only ones that rely on randomized data-independent linear projections. All the other methods, including our PCA-RR and PCA-ITQ, use PCA (or a non-orthogonal relaxation of PCA) as an intermediate dimensionality reduction step.

4.3 Results on CIFAR Dataset

Figure 5(a) compares all the methods based on their mean average precision for Euclidean neighbor ground truth. Perhaps surprisingly, the natural baseline for our method, PCA-RR, already outperforms everything except PCA-ITQ for most code sizes. The only exception is SKLSH, which has a strongly upward trajectory and gets the best performance for a code size of 256. This behavior may be due to the theoretical convergence guarantee of SKLSH [36]. LSH, which is data-independent just like SKLSH, also improves as the code size increases, and it almost reaches the performance level of PCA-RR at 256 bits. As for PCA-ITQ, it consistently performs better than PCA-RR, although the advantage becomes smaller as the code size increases. Thus, adapting to the data distribution seems especially important when the code size is small. In particular, doing the ITQ refinement for a 64-bit code raises its performance almost to the level of the 256-bit PCA-RR code.

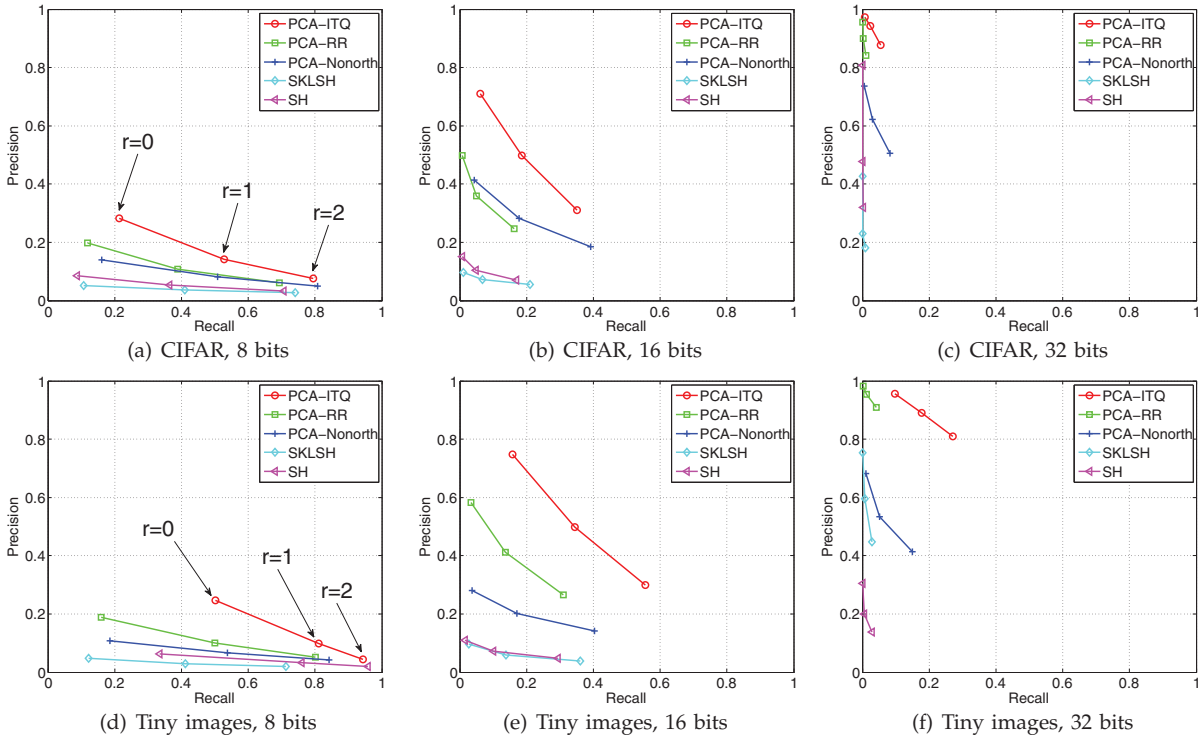
Figure 5(b) evaluates the semantic consistency of the codes using class labels as ground truth. For each method, it shows the precision for top 500 returned images as a function of code size. As in Figure 5(a), PCA-RR and PCA-ITQ outperform all the other methods, and PCA-ITQ has a small but consistent advantage over PCA-RR. There are some interesting differences among the other methods, however. Unlike in Figure 5(a), PCA-Direct works relatively well for the smallest code sizes (32 and 64 bits), while SKLSH works surprisingly poorly. This may be due to the fact that unlike most of the other methods, SKLSH does not rely on PCA. Our results seem to indicate that PCA really helps to preserve semantic consistency for the smallest code sizes. Even at 256 bits, while SKLSH had by far the best performance for Euclidean neighbor retrieval, it lags behind most of the other methods in terms of class label precision. This underscores the fact that the best Euclidean neighbors are not necessarily the most semantically consistent, and that it is necessary to evaluate performance using *both* Euclidean and class label ground truth. Another observation worth making is that the most heuristic methods, namely PCA-Direct and SH, can actually get *worse* as the number of bits increases.

Figures 6 and 7 show complete recall-precision and class label precision curves corresponding to the summary numbers in Figures 5(a,b). To avoid clutter, these curves (and all the subsequent results reported in this paper) omit the two baseline methods LSH and PCA-Direct. The complete curves confirm the trends seen in Figures 5(a,b). One thing that becomes especially apparent from looking at Figure 6(d) is that the data-dependent methods (PCA-Nonorth, PCA-RR, PCA-ITQ) seem to hit a ceiling of performance as code size increases, while the data-independent SKLSH method does not have a similar limitation (in fact, in the limit of infinitely many bits, SKLSH is guaranteed to yield exact Euclidean neighbors). Once again, the message seems to be that adapting binary codes to the data can give the biggest gain for small code sizes.

4.4 Results on 580,000 Tiny Images

Figure 8 shows precision-recall curves and mAP for Euclidean neighbor retrieval on the 580,000 Tiny Images. As explained in Section 4.1, there are no ground truth class labels for this dataset, so we cannot evaluate class label precision. The relative ordering of the different methods is largely consistent with results on CIFAR, with PCA-ITQ getting an even bigger performance advantage at small code sizes.

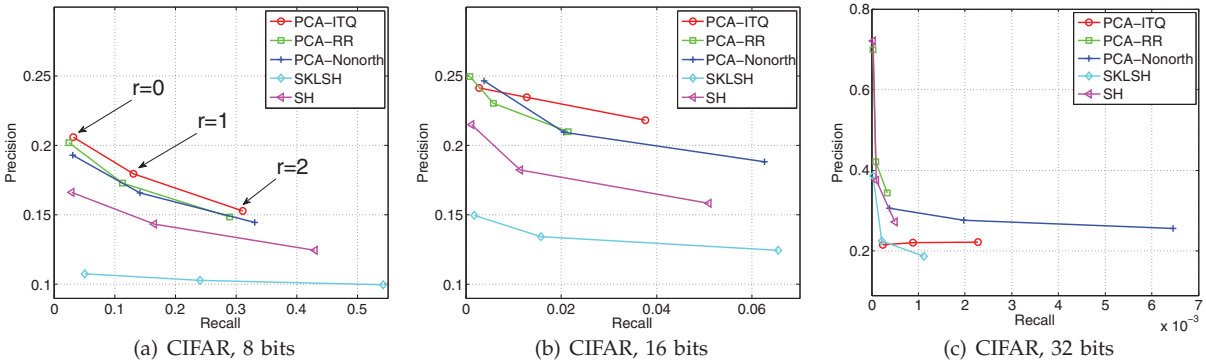
This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication.



(a) CIFAR, 8 bits (b) CIFAR, 16 bits (c) CIFAR, 32 bits

(d) Tiny images, 8 bits (e) Tiny images, 16 bits (f) Tiny images, 32 bits

Figure 9. Hashing performance for different Hamming radii r with Euclidean neighbor ground truth. (a-c) CIFAR dataset; (d-f) 580,000 Tiny Images dataset.



(a) CIFAR, 8 bits (b) CIFAR, 16 bits (c) CIFAR, 32 bits

Figure 10. Hashing performance on the CIFAR dataset with class label ground truth. r is Hamming radius. Note the progressively decreasing recall from (a) to (c), and the different vertical scale in (c).

Moreover, comparing Figure 8(d) with Figure 5(a), we can see that at 256 bits, SKLSH, PCA-Nonorth, PCA-RR, and PCA-ITQ converge to a higher level of mAP performance than on the smaller CIFAR dataset. This may be because the larger dataset samples the feature space more densely, making it easier to find good image matches.

4.5 Evaluation of Hashing Performance

To fully realize the potential of binary codes for large-scale datasets, we would like to be able to use them for hashing or indexing as opposed to exhaustive search. For this, we would need a very small code (32 bits or less) to yield reasonably high precision and recall among retrieved points that lie within a Hamming distance of 0 to 2 from the query. Figure 9 shows the recall and precision of 8, 16 and 32-bit codes at Hamming radii $r = 0$, 1, and 2 for several methods on CIFAR and 580,000 Tiny Images with Euclidean neighbor ground truth. PCA-ITQ is almost always the best in terms of both recall and precision. Furthermore, comparing Figure 9(a-c) with (d-f), we can find that the recall of PCA-ITQ improves substantially when we increase the dataset size by an order of magnitude.

Figure 10 reports hashing performance in terms of class label retrieval. In this case, ITQ has good precision for 8 and 16 bits, but not for 32 bits. At first glance, this seems to contradict the result of Figure 5(b) where ITQ has the highest class label precision at top 500 retrieved images for every code size. To

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication.

explain this, note that some query points have many matches in a small Hamming radius (corresponding to dense areas of the feature space) and some queries have very few. The precision for a very small Hamming ball is significantly reduced by the queries with many matches since the matches usually do not all belong to the same class. Note that the same finding does not hold for Euclidean neighbor precision since the queries having many matches in a small Hamming ball also have many Euclidean neighbors. In any case, the semantic label recall of all the methods is dismally low, indicating that a lot more research needs to be done before similarity-preserving binary codes become truly useful for hashing.

5 Leveraging Label Information

5.1 Supervised CCA Embedding

When label information is available for the images that are used to train the embedding, we can choose a supervised dimensionality reduction method to better capture the semantic structure of the dataset. In our work, we use the Canonical Correlation Analysis (CCA) [16], which has proven to be an effective tool for extracting a common latent space from two views [11] and is robust to noise [3]. We assume that each training image descriptor $\boldsymbol{x}_i \in \mathbb{R}^d$ has associated with it a label vector $\boldsymbol{y}_i \in \{0,1\}^t$, where t is the total number of labels (search keywords, tags) available, and a given entry of $\boldsymbol{y}_i$ is 1 if the image is associated with the corresponding label and 0 otherwise. Note that the labels do not have to be mutually exclusive and may be noisy. The label vectors form the rows of a label matrix $Y \in \{0,1\}^{n \times t}$. The goal of CCA is to find projection directions $\boldsymbol{w}_k$ and $\boldsymbol{u}_k$ for feature and label vectors to maximize the correlation between the projected data $X\boldsymbol{w}_k$ and $Y\boldsymbol{u}_k$:

$$\begin{aligned} & \mathcal{C}(\boldsymbol{w}_k, \boldsymbol{u}_k) = \boldsymbol{w}_k^T X^T Y \boldsymbol{u}_k \\ s.t. \quad & \boldsymbol{w}_k^T X^T X \boldsymbol{w}_k = 1, \ \boldsymbol{u}_k^T Y^T Y \boldsymbol{u}_k = 1 . \end{aligned}$$

Maximizing the above objective function involves solving the following generalized eigenvalue problem to get $\boldsymbol{w}_k$:

$$X^T Y (Y^T Y + \rho I)^{-1} Y^T X \boldsymbol{w}_k = \lambda_k^2 (X^T X + \rho I) \boldsymbol{w}_k, \quad (4)$$

in which ρ is a small regularization constant used to prevent a trivial solution [11] (we set ρ to be 0.0001 in our experiments). The leading generalized eigenvectors of (4) then give us a sequence of orthogonal $\boldsymbol{w}_k$ directions that span the solution space, just as for PCA. Note that once we have $\boldsymbol{w}_k$, we can also solve for the corresponding $\boldsymbol{u}_k$, but in our case, we only care about the projection directions in the data space, since we assume that label information will be unavailable at test time.

For a c-bit code, we form a projection matrix $\widehat{W} \in \mathbb{R}^{d \times c}$ whose columns are given by the leading eigenvectors $\boldsymbol{w}_k$ scaled by the corresponding eigenvalues λ_k. The reason for the scaling is that for supervised embeddings such as CCA, the most discriminative information tends to concentrate in the top eigenvectors. Since we usually have more bits in the code than the number of most informative eigenvectors, scaling the trailing eigenvectors can reduce noise and empirically produce better performance.[2] More generally, it is possible to scale the eigenvectors by the tth power of the eigenvalues [6], but in our experiments, we have found that $t = 1$ works the best. Finally we obtain the embedded dataset $V = X\widehat{W}$ in the new latent space that preserves both visual and semantic similarity.

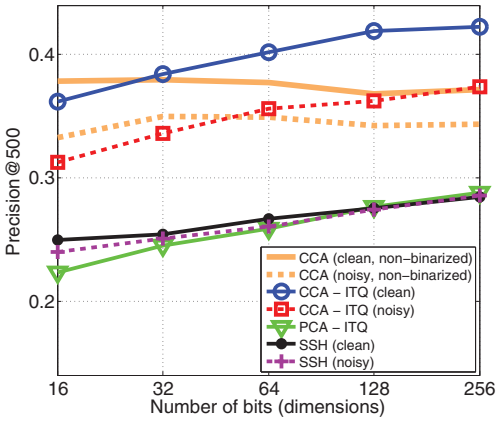


Figure 11. Average precision for top 500 retrieved images for supervised data embeddings based on clean and noisy labels.

5.2 Results

Recall from Section 4.1 that the CIFAR dataset comes with manually verified keywords, while the 580,000 Tiny Images subset comes with noisy keywords that have not been verified by humans. These two different kinds of annotation allow us to explore the power of the CCA embedding given both "clean" and "noisy" supervisory information. For the "clean" scenario, we use a setup analogous to that of Section 4.3: namely, we set aside 1,000 query images from the CIFAR dataset and use the remaining CIFAR images as the training set and the database against which the queries are run. The labels in the training set are used to train the CCA embedding. For the query images, the class labels are used only for benchmarking. For the "noisy" scenario, we learn the CCA embedding from all the Tiny Images that are disjoint from the CIFAR dataset using their unverified search keywords as the supervisory information. Then we apply the resulting embedding to the CIFAR dataset, split it into query images and reference database as in the clean

2. We have found this is not true for PCA. CCA is a supervised embedding, so the top eigenvectors carry much more discriminative information than in the case of PCA.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication.

Figure 12. Image search (32-bit binary code) results on CIFAR. Red border means false positive.

scenario, and use the "clean" ground-truth labels for benchmarking.

As a baseline, we use the semi-supervised approach of [49], in which the label information of the n data points is used to construct an $n \times n$ matrix S that modulates the data covariance matrix. We set $S_{ij} = 1$ if two data points $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$ have the same label, and 0 otherwise. Then we find the projection matrix W by taking the eigendecomposition of $X^T S X$. Note that [49], which assumes that few labeled images are available, regularizes $X^T S X$ by adding to it a small multiple of the covariance matrix $X^T X$. In our case, we have found this regularization to be unnecessary. We then take the data-dependent embedding W and perform ITQ refinement. We call the resulting method SSH-ITQ. Note that in [49], the semi-supervised embedding is combined with nonorthogonal relaxation (SSH-Nonorth), however, just as in Section 4, we have found that SSH-ITQ works better than SSH-Nonorth, so we only reproduce the SSH-ITQ results here.

Figure 11 shows the averaged precision at top 500 retrieved images for the "clean" and "noisy" versions of the CCA and SSH embeddings. For reference, we also include the performance of the unsupervised PCA embedding. We can see that CCA-ITQ with clean labels achieves the highest performance, while CCA-ITQ with noisy labels still gives a big improvement over the unsupervised PCA-ITQ. On the other hand, both versions of SSH fail to improve much over the PCA baseline. For reference, this figure also shows retrieval precision curves for uncompressed CCA-projected data with both "clean" and "noisy" supervisory information. Interestingly, after 32 bits, the ITQ-compressed data actually begins to give better retrieval performance than the uncompressed data! It seems that clustering the discriminative CCA outputs to vertices of the binary hypercube is actually accomplishing some sort of "semantic hashing" [39]. However, the precise explanation for this effect remains unclear, and investigating it is an important problem for the future. Finally, Figure 12 shows retrieved images for several sample query images. We can clearly see that when labels are incorporated, the results are much more semantically consistent.

6 ITQ WITH A KERNEL EMBEDDING

6.1 Random Fourier Features

A big limitation of PCA and CCA is that they can only capture linear structure in the data. In order to introduce nonlinearity into the embedding process, we can use kernel PCA (KPCA) [40]. Finding the KPCA embedding for n feature vectors involves

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication.

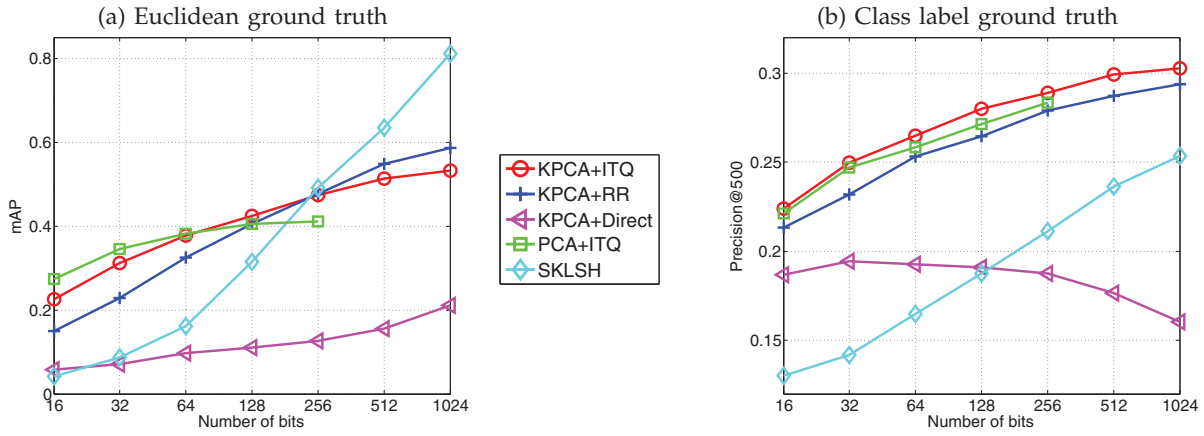


Figure 13. Comparative evaluation of kernel ITQ on CIFAR dataset. (a) Performance is measured by mean average precision (mAP) for Euclidean neighbor retrieval. (b) Performance is measured by the averaged precision of top 500 ranked images for each query where ground truth is defined by semantic class labels.

computing the $n \times n$ kernel matrix and performing eigendecomposition on it. However, for large-scale image databases, these operations are prohibitively expensive, so we have to resort to approximation schemes.

In this paper, we are particularly interested in the Gaussian kernel, whose radius can be used to control the neighborhood size for nearest-neighbor search. To approximate the Gaussian kernel $K(\boldsymbol{x}, \boldsymbol{y}) = \exp(-\|\boldsymbol{x} - \boldsymbol{y}\|^2/(2\sigma^2))$, we can use the explicit *random Fourier feature* (RFF) mapping [37]. For a data point $\boldsymbol{x}$, each coordinate of this mapping is given by

$$\Phi_{\boldsymbol{w},b}(\boldsymbol{x}) = \sqrt{2}\cos(\boldsymbol{x}\boldsymbol{w} + b)\,,$$

where the random projection vector $\boldsymbol{w}$ is drawn from Normal$(0, \frac{1}{\sigma^2}I)$ and b is drawn from Unif$[0, 2\pi]$. A D-dimensional embedding is given by

$$\Phi^D(\boldsymbol{x}) = [\Phi_{\boldsymbol{w}_1,b_1}(\boldsymbol{x}), \Phi_{\boldsymbol{w}_2,b_2}(\boldsymbol{x}), \ldots, \Phi_{\boldsymbol{w}_D,b_D}(\boldsymbol{x})].$$

The inner product of the mapped data approximates the Gaussian kernel as $K(\boldsymbol{x}, \boldsymbol{y}) \approx \Phi^D(\boldsymbol{x})\Phi^D(\boldsymbol{y})^T$. When D goes to infinity, the mapping becomes exact. In our experiments, we use $D = 3,000$. After the random Fourier mapping, we simply perform linear PCA on top of $\Phi^D(X)$ to obtain an approximate KPCA embedding for the data. Given the points in our dataset, we first transform them using RFF, then perform KPCA to reduce the dimensionality, and finally binarize the data in the same way as before.

Note that while we solely focus on approximation of the Gaussian kernel, there also exist explicit mappings for other popular kernels [27], [34], [47].

6.2 Results

We first compare KPCA with ITQ to other baseline methods, including linear PCA with ITQ. Results for Euclidean neighbor retrieval are reported in Figure 13 (a). One of the advantages of RFF is that it allows us to learn binary codes whose dimension is higher than that of the original data (320 in the experiments so far). Thus, we report results up to 1,024 bits for the kernel methods, and for PCA-ITQ, we report results up to 256 bits. For RFF, we set the radius of the Gaussian kernel to the average distance to the 50th nearest neighbor. From Figure 13(a), in terms of Euclidean neighbor retrieval, KPCA-ITQ starts to have an advantage over PCA-ITQ beginning with 128 bits (though we have found that it is possible to tune the radius of the Gaussian kernel to match the performance of PCA-ITQ for shorter codes), and KPCA-RR seems to work the best for the longest code sizes. Figure 13 (b) reports performance in terms of class label retrieval. For this case, KPCA-ITQ is consistently better than PCA-ITQ or KPCA-RR.

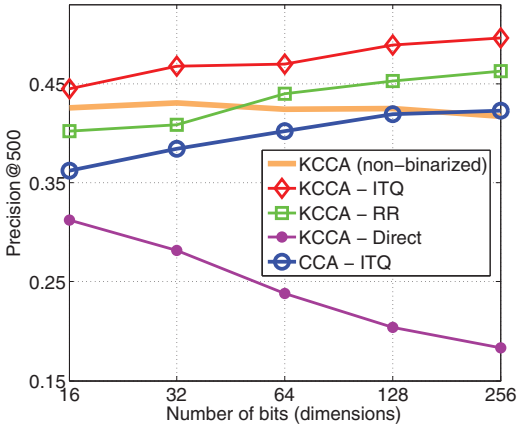


Figure 14. Comparison of KCCA methods with CCA-ITQ on the CIFAR dataset. Performance is evaluated in terms of average class label precision for top 500 retrieved images. Results are generated using clean labels (refer to Section 5 for a description of the label information in the CIFAR dataset).

It is interesting to compare KPCA-RR/ITQ and

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication.

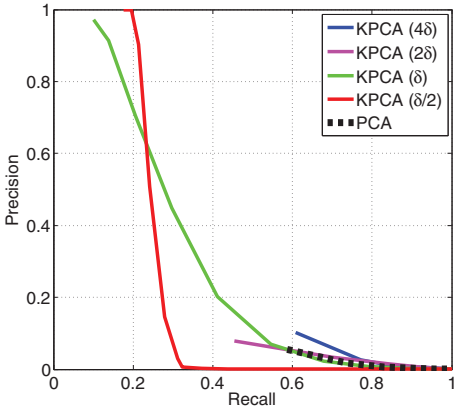


Figure 15. Performance of kernel ITQ (32 bits) with a ground truth radius δ equal to half the distance to the 50th nearest neighbor. The numbers in parentheses show the value of the kernel width σ used in the RFF mapping.

SKLSH [36], since the latter is also based on RFF. To obtain a c-bit code, SKLSH first computes a c-dimensional RFF embedding of the data and then quantizes each dimension after adding a random threshold. By contrast, our KPCA-based methods start with a 3,000-dimensional RFF embedding and then use PCA to reduce it to top c dimensions. Based on the results of Figure 13, this process makes an especially big difference for class label precision.

Next, Figure 14 shows results for performing CCA on top of RFF to learn a supervised embedding. As a baseline, this figure includes the curve for CCA-ITQ, which was also shown in Figure 11 (where it was, in turn, a big improvement over the unsupervised PCA-ITQ). Now, we can see that KCCA-RR and KCCA-ITQ both significantly improve over CCA-ITQ, with KCCA-ITQ giving the best results for every code size. Moreover, KCCA-ITQ outperforms the continuous KCCA baseline even for 16 bits! This is even stronger than the finding of Figure 11, where we first saw that performing ITQ on top of a supervised embedding can actually improve its semantic precision. Note that a recent work [25] has proposed a supervised kernel hashing scheme that shares a similar motivation to ours here. In the future, it will be interesting to compare our method with that of [25].

Finally, we want to demonstrate the effect of the radius σ of the Gaussian kernel in the RFF mapping. Intuitively, by tuning this parameter, we can control how the binary codes approximate the neighbors: a smaller radius can give us better approximation for very close neighbors, while a larger one can give us codes that better reflect the global structure of the data. To demonstrate this, we use a ground-truth radius δ defined as *half* the distance to the 50th nearest neighbor. This results in only very close points (near duplicates) being defined as ground truth neighbors. Figure 15 shows recall and precision of KPCA-ITQ for $\sigma = [\delta/2,\ \delta,\ 2\delta,\ 4\delta]$. The best performance is obtained for δ and $\delta/2$, which match the desired neighborhood size the most closely. By contrast, KPCA-ITQ with a larger radius, or PCA-ITQ, which does not have a radius parameter, work very poorly in this regime.

Method	Dimensionality	FV Precision@50
Fisher Vector	4096 × 32 bits	33.40
Classeme	950 × 32 bits	**39.24**
Classeme-Threshold	950 × 1 bits	31.54
Classeme-ITQ	950 × 1 bits	**38.98**
Classeme-RR	950 × 1 bits	37.06
Classeme-SH	950 × 1 bits	22.66
Classeme-LSH	950 × 1 bits	36.27
Classeme-SKLSH	950 × 1 bits	33.35

Table 1
Image retrieval results (average class label precision at top 50 returned images) on 50 classes from ILSVRC2010.

	Percent of training set				
Method	1%	5%	10%	50%	100%
Fisher Vector	35.55	52.21	**57.11**	**66.21**	**69.16**
Classeme	38.54	51.49	56.18	64.31	66.77
Classeme-Threshold	34.17	47.93	51.16	56.75	58.72
Classeme-ITQ	**40.58**	**53.32**	56.62	61.12	62.68
Classeme-RR	37.33	50.39	54.06	58.83	60.55

Table 2
Image classification results (%) on 50 classes from ILSVRC2010 using Classemes trained on Fisher Vectors. Standard deviations are around 1%.

Let us summarize the benefits of using the RFF embedding in combination with ITQ. First, it allows us to use more bits than original feature dimensions to get better code accuracy. Second, it significantly improves class label precision, especially when combined with CCA. Third, it has a tunable radius parameter that can be changed to obtain much better performance on tasks such as near-duplicate image retrieval.

7 Learning Classemes from ImageNet

Finally, we present an application of ITQ to learning binary *classeme* features [2], [38], [46], [48]. The idea of classemes is to train large numbers of visual classifiers and then use their outputs on a new test image as a high-level feature for novel category recognition.

For the experiments of this section, we collect data from the ImageNet database [8], which contains more than 14 million labeled images from more than 22,000 categories. Specifically, we use the ILSVRC2010 subset containing 1.2 million images from 1000 categories. We represent images using state-of-the-art Fisher vector (FV) [32] descriptors, which are computed as follows. Images are resized to have an area of 100,000 pixels (if larger). SIFT [26] and color descriptors [35] are extracted from 24×24 patches every 6 pixels

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication.

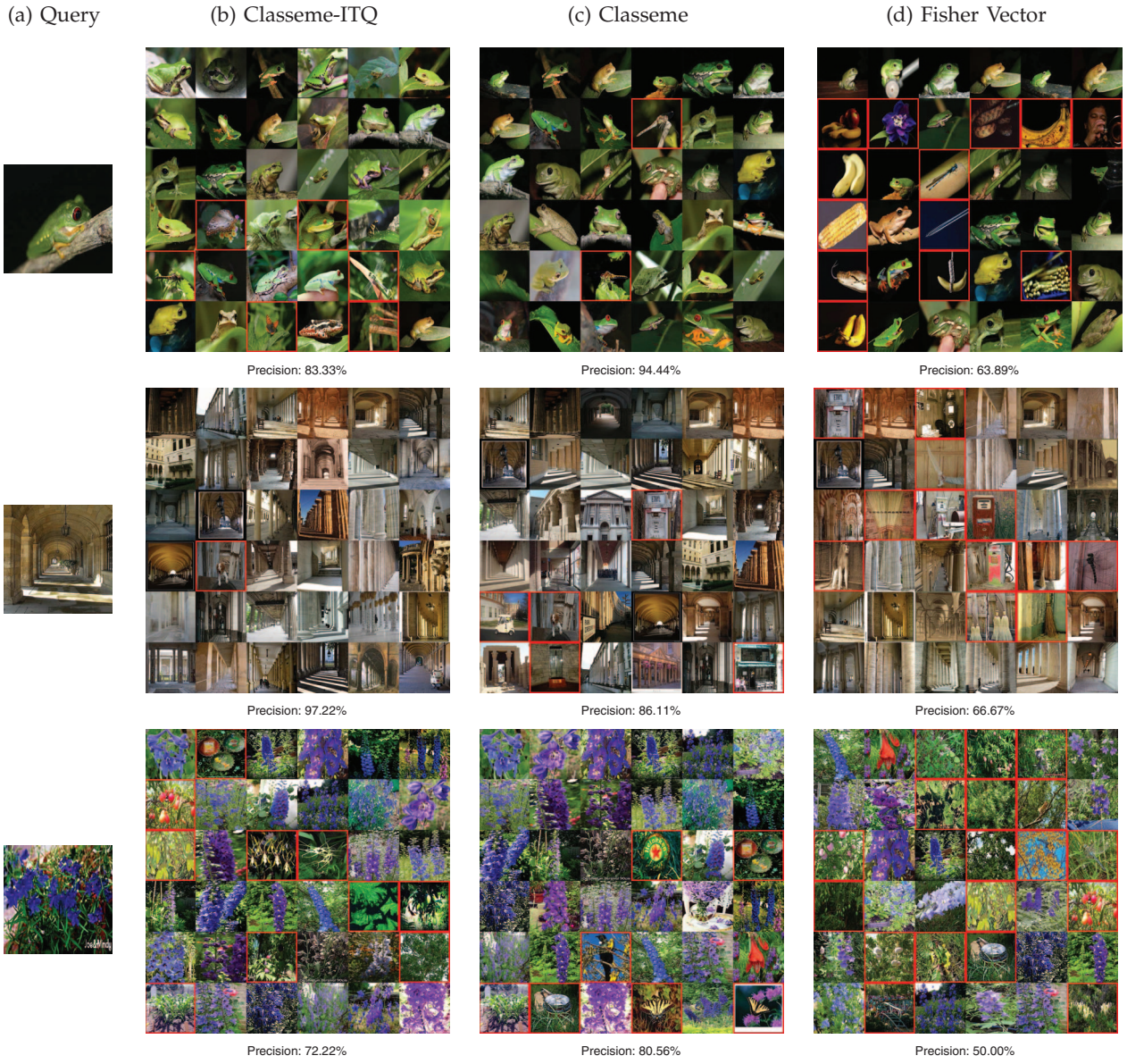


Figure 16. Sample image search results on ImageNet (50 testing categories). We qualitatively compare Fisher Vector, Classeme, and binary Classeme (950 bits) obtained by ITQ. Red border means false positive. Note that in some of the examples, Classeme-ITQ has lower precision than the continuous Classeme.

at 5 scales (by resizing the image by a factor $\sqrt{2}$ between two scales). The dimensionality of both SIFT and color descriptors is reduced to 64 by PCA. For each descriptor type, a Gaussian Mixture Model with 16 components is trained. Given an image, a 2048-dimensional FV is computed for SIFT and color by taking the gradient with respect to the mean and standard deviation parameters. The FVs are power- and L2-normalized as suggested in [35]. The final descriptor is the concatenation of the SIFT and color FV's, and is 4096-dimensional.

To learn the classemes, we randomly pick 950 classes from ILSVRC2010 and train LIBLINEAR SVM classifiers [9] on them. We use all the positive training data for each class, and randomly sample the same amount of negative training data from the remaining classes. The regularization parameter C of LIBLINEAR SVM is set to 2. Then the output scores of the 950 classifiers are used as classeme features to recognize the remaining 50 categories. To convert classemes to binary codes, we simply run ITQ (or any other binarization method) on top of the classifier outputs without any additional dimensionality reduction.

For our first experiment, we want to see how well binary classemes work as a representation for image retrieval. There are a total of 68,295 images in our 50 "testing" categories. We set aside 1,000 of those as "queries" and find their nearest neighbors among the remaining 67,295 images. Table 1 compares the average class label precision of the top 50 returned

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication.

images[3] for a number of coding methods. First of all, we can see that the continuous 950-dimensional classemes give better retrieval performance than the low-level FV features. Binarizing the classemes by directly thresholding them drops the performance significantly. However, binarizing them with ITQ results in almost no degradation (we have randomly sampled five other 950/50 class splits, and found that classeme-ITQ is consistently about 0.5% worse than the uncompressed classeme). For comparison, the table includes a number of other binary coding methods, all of which are worse than ITQ. Qualitative image retrieval results are shown in Figure 16.

Next, we would like to use classemes for novel category recognition, as suggested in [2], [46]. The motivating scenario is as follows: suppose we have already learned visual models for a number of categories using a large amount of training data, and then we are given a much smaller amount of training data for a never before seen category. Can we leverage our previous knowledge to quickly and efficiently learn a model for the new category?

In our experimental setup, we use the classifiers for the 950 categories as "previous knowledge." For the remaining 50 categories, we set aside 80% of the data for training, 10% for validation, and 10% for testing. We randomly sample different amounts of training data from the 80% training set for five trials, and train LIBLINEAR SVMs on the binary classeme features. The regularization parameter of linear SVM is tuned on the validation set using the grid of $[0.0002,\ 0.002,\ 0.02,\ 0.2,\ 2,\ 20]$. Table 2 shows the resulting classification accuracies. As before, the classeme-ITQ representation outperforms all the other binary embeddings. More interestingly, classeme-ITQ works better than continuous classeme or Fisher vectors for small training set size, where the higher-dimensional features may be overfitting. For larger training sizes, overfitting is no longer an issue, and the original Fisher vectors work the best. Thus we conclude that ITQ-based binary classemes may be well suited for recognition of novel categories when a very small amount of training data is available.

8 Discussion

The main insight of our work is that the problem of learning similarity-preserving binary codes can be successfully formulated in terms of rotating zero-centered PCA-projected data so as to minimize the quantization error of mapping that data to the vertices of a zero-centered binary hypercube. Our ITQ method for finding this rotation is simple and efficient, and produces compelling improvements over the state of the art. Along the way, we have also shown that a *random* rotation [19] already works better than more elaborate schemes like non-orthogonal relaxation [49], and thus makes a strong baseline for evaluating ITQ. In Section 5, we have shown that the classic CCA embedding [16] gives a very effective way of utilizing clean or noisy labels for supervised training of binary codes. Intriguingly, our experiments even suggest that binarizing CCA-projected data with ITQ can actually *improve* the semantic precision of retrieval. This implies that ITQ on top of a supervised embedding does a good job of finding a binary code structure that accurately reflects the underlying class label structure – a phenomenon that needs more study in the future. In Section 6, we have demonstrated that even further improvements can result from using a nonlinear kernel embedding prior to PCA or CCA. Finally, Section 7 has given an application of ITQ to creating a compressed binary "classeme" representation suitable for retrieval and novel category recognition on Internet-scale datasets.

3. Recall that in our CIFAR retrieval experiments, we were evaluating precision at top 500 returned images. That number was chosen because CIFAR has 6,000 images per class. ILSVRC2010 has only 300-1,200 images per class, so evaluating precision at top 50 is more appropriate.

Acknowledgments

We thank Jun Wang, Rob Fergus, Joe Tighe, Ruiqi Guo, and Hervé Jégou for helpful discussions and for sharing their code and data. Gong and Lazebnik are partially supported by NSF CAREER Award IIS 0845629, Microsoft Research Faculty Fellowship, Xerox, DARPA Computer Science Study Group, and ARO. Gordo is partially supported by the Spanish projects TIN2009-14633-C03-03 and CONSOLIDER-INGENIO 2010 (CSD2007-00018).

References

[1] A. Andoni and P. Indyk. Near-optimal hashing algorithms for approximate nearest neighbor in high dimensions. *Communication of the ACM*, 2008.
[2] A. Bergamo, L. Torresani, and A. Fitzgibbon. Picodes: Learning a compact code for novel-category recognition. *NIPS*, 2011.
[3] M. B. Blaschko and C. H. Lampert. Correlational spectral clustering. *CVPR*, 2008.
[4] P. Brasnett and M. Bober. Fast and robust image identification. *ICPR*, 2008.
[5] M. Bronstein, A. Bronstein, N. Paragios, and F. Michel. Data fusion through cross-modality metric learning using similarity-sensitive hashing. *CVPR*, 2010.
[6] O. Chapelle, J. Weston, and B. Schoelkopf. Cluster kernels for semi-supervised learning. *NIPS*, 2002.
[7] O. Chum and J. Matas. Large scale discovery of spatilly related images. *PAMI*, 2010.
[8] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. Imagenet: A large-scale hierarchical image database. *CVPR*, 2009.
[9] R.-E. Fan, K.-W. Chang, C.-J. Hsieh, X.-R. Wang, and C.-J. Lin. Liblinear: A library for large linear classification. *Journal of Machine Learning Research*, 2008.
[10] R. Fergus, A. Torralba, and Y. Weiss. Semi-supervised learning in gigantic image collections. *NIPS*, 2009.
[11] D. P. Foster, R. Johnson, S. M. Kakade, and T. Zhang. Multi-view dimensionality reduction via canonical correlation analysis. *Tech Report. Rutgers University*, 2010.
[12] J.-M. Frahm, P. Georgel, D. Gallup, T. Johnson, R. Raguram, C. Wu, Y.-H. Jen, E. Dunn, B. Clipp, S. Lazebnik, and M. Pollefeys. Building Rome on a cloudless day. In *ECCV*, 2010.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication.

[13] Y. Gong and S. Lazebnik. Iterative quantization: A Procrustean approach to learning binary codes. *CVPR*, 2011.
[14] A. Gordo and F. Perronnin. Asymmetric distances for binary embeddings. *CVPR*, 2011.
[15] J. He, R. Radhakrishnan, S.-F. Chang, and C. Bauer. Compact hashing with joint optimization of search accuracy and time. *CVPR*, 2011.
[16] H. Hotelling. Relations between two sets of variables. *Biometrika*, 28:312–377, 1936.
[17] H. Jégou, M. Douze, and C. Schmid. Hamming embedding and weak geometric consistency for large-scale image search. *ECCV*, 2008.
[18] H. Jégou, M. Douze, and C. Schmid. Product quantization for nearest neighbor search. *IEEE TPAMI*, 2010.
[19] H. Jégou, M. Douze, C. Schmid, and P. Perez. Aggregating local descriptors into a compact image representation. *CVPR*, 2010.
[20] A. Joly and O. Buisson. Random maximum margin hashing. *CVPR*, 2011.
[21] A. Krizhevsky. Learning multiple layers of features from tiny images. *Tech Report. University of Toronto*, 2009.
[22] B. Kulis and K. Grauman. Kernelized locality-sensitive hashing for scalable image search. In *ICCV*, 2009.
[23] R.-S. Lin, D. Ross, and J. Yagnik. Spec hashing: Similarity preserving algorithm for entropy-based coding. *CVPR*, 2010.
[24] W. Liu, S. Kumar, and S.-F. Chang. Hashing with graphs. *ICML*, 2011.
[25] W. Liu, J. Wang, R. Ji, Y.-G. Jiang, and S.-F. Chang. Supervised hashing with kernels. *CVPR*, 2012.
[26] D. G. Lowe. Distinctive image features from scale-invariant keypoints. *IJCV*, 2004.
[27] S. Maji, A. C. Berg, and J. Malik. Classification using intersection kernel support vector machines is efficient. In *CVPR*, 2008.
[28] D. Nister and H. Stewenius. Scalable recognition with a vocabulary tree. In *CVPR*, 2006.
[29] M. Norouzi and D. J. Fleet. Minimal loss hashing for compact binary codes. *ICML*, 2011.
[30] A. Oliva and A. Torralba. Modeling the shape of the scene: a holistic representation of the spatial envelope. *IJCV*, 2001.
[31] L. Pauleve, H. Jegou, and L. Amsaleg. Locality sensitive hashing: a comparison of hash function types and querying mechanisms. *Pattern Recognition Letters*, 2010.
[32] F. Perronnin and C. R. Dance. Fisher kernels on visual vocabularies for image categorization. *CVPR*, 2007.
[33] F. Perronnin, Y. Liu, J. Sánchez, and H. Poirier. Large-scale image retrieval with compressed Fisher vectors. In *CVPR*, pages 3384–3391, 2010.
[34] F. Perronnin, J. Sanchez, , and Y. Liu. Large-scale image categorization with explicit data embedding. *CVPR*, 2010.
[35] F. Perronnin, J. Sanchez, and T. Mensink. Improving the fisher kernel for large-scale image classification. *ECCV*, 2010.
[36] M. Raginsky and S. Lazebnik. Locality sensitive binary codes from sift-invariant kernels. *NIPS*, 2009.
[37] A. Rahimi and B. Recht. Random features for large-scale kernel machines. *NIPS*, 2007.
[38] N. Rasiwasia, P. Moreno, and N. Vasconcelos. Bridging the gap: Query by semantic example. *IEEE Transactions on Multimedia*, 2007.
[39] R. Salakhutdinov and G. Hinton. Semantic hashing. *International Journal of Approximate Reasoning*, 2009.
[40] B. Schölkopf, A. Smola, and K.-R. Müller. Kernel principal component analysis. In *ICANN*, 1997.
[41] P. Schonemann. A generalized solution of the orthogonal procrustes problem. *Psychometrika*, 31, 1966.
[42] G. Shakhnarovich, T. Darrell, and P. Indyk, editors. *Nearest-Neighbors methods in Learning and Vision: Theory and Practice*. MIT Press, 2006.
[43] C. Strecha, A. M. Bronstein, M. M. Bronstein, and P. Fua. Ldahash: Improved matching with smaller descriptors. *PAMI*, 2010.
[44] A. Torralba, R. Fergus, and W. Freenman. 80 million tiny images: a large dataset for non-parametric object and scene recognition. *PAMI*, 2008.
[45] A. Torralba, R. Fergus, and Y. Weiss. Small codes and large image databases for recognition. *CVPR*, 2008.
[46] L. Torresani, M. Szummer, , and A. Fitzgibbon. Efficient object category recognition using classemes. *ECCV*, 2010.
[47] A. Vedaldi and A. Zisserman. Efficient additive kernels via explicit feature maps. *CVPR*, 2010.
[48] G. Wang, D. Hoiem, and D. Forsyth. Learning image similarity from flickr groups using stochastic intersection kernel machines. *ICCV*, 2009.
[49] J. Wang, S. Kumar, and S.-F. Chang. Semi-supervised hashing for scalable image retrieval. *CVPR*, 2010.
[50] J. Wang, S. Kumar, and S.-F. Chang. Sequential projection learning for hashing with compact codes. *ICML*, 2010.
[51] Y. Weiss, A. Torralba, and R. Fergus. Spectral hashing. *NIPS*, 2008.
[52] S. X. Yu and J. Shi. Multiclass spectral clustering. *ICCV*, 2003.

Yunchao Gong received the BE degree in software engineering from Nanjing University, China, and the MS degree in computer science from the University of North Carolina at Chapel Hill in 2009 and 2012, respectively. He is currently working toward the PhD degree in Department of Computer Science at UNC Chapel Hill. His research interests include object recognition, image retrieval, locality sensitive hashing, and machine learning. He is a student member of the IEEE.

Svetlana Lazebnik received her Ph.D. in 2006 at the University of Illinois at Urbana-Champaign. From 2007 to 2012, she was an assistant professor of computer science at the University of North Carolina at Chapel Hill. As of January 2012, she has moved back to UIUC as an assistant professor. She is the recipient of an NSF CAREER award and a Microsoft Research Faculty Fellowship, a member of the 2011 DARPA Computer Science Study Group, and a member of the editorial board of the International Journal of Computer Vision. Her research interests include computer vision, image understanding, and machine learning techniques for visual data.

Albert Gordo received his BSc degree in Computer Engineering and his MSc degrees in Intelligent Systems from the University Jaume I in Castelln, Spain, in 2007 and 2009, respectively. He also holds a degree in Computational Math obtained in 2007. He is currently pursuing his PhD in the Computer Vision Center of Barcelona, Spain in collaboration with the Xerox Research Centre Europe in Grenoble, France, under the supervision of Dr. Ernest Valveny and Dr. Florent Perronnin. His main research interests include document image analysis and image retrieval, particularly focusing on large-scale problems.

Florent Perronnin received his Engineering degree in 2000 from the ENST (Paris, France) and his Ph.D. degree in 2004 from the EPFL (Lausanne, Switzerland). From 2000 to 2001 he was a Research Engineer with the Panasonic Speech Technology Laboratory (Santa Barbara, California) working on speech and speaker recognition. In 2005, he joined the Xerox Research Centre Europe (Grenoble, France). His main interests are in the application of machine learning to computer vision tasks such as image classification, retrieval or segmentation.