Iterative Quantization: A Procrustean Approach to Learning Binary Codes

Yunchao Gong and Svetlana Lazebnik
Department of Computer Science, UNC Chapel Hill, NC, 27599.
{yunchao,lazebnik}@cs.unc.edu

Abstract

This paper addresses the problem of learning similarity-preserving binary codes for efficient retrieval in large-scale image collections. We propose a simple and efficient alternating minimization scheme for 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. This method, 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). Our experiments show that the resulting binary coding schemes decisively outperform several other state-of-the-art methods.

1. Introduction

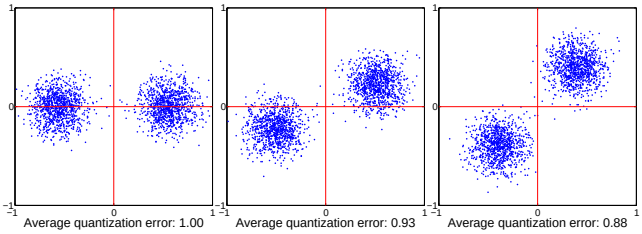
Recently, the vision community has devoted a lot of attention to the problem of learning similarity-preserving binary codes for representing large-scale image collections [12, 16, 18, 19, 20, 21]. Encoding high-dimensional image descriptors as compact binary strings can enable large efficiency gains in storage and computation speed for similarity search, and it can be accomplished with much simpler data structures and algorithms than alternative large-scale indexing methods [3, 7, 9, 10].

As discussed in [21], an effective scheme for learning binary codes should have several properties. First, the codes should be short so that we could store large amount of images in memory. For example, for an ordinary workstation with 16G memory, to store 250 million images in memory, we could only use about 64 bits for each image. Second, the codes should map images that are similar (either perceptually or semantically) 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 very efficient. The need to simultaneously satisfy all three constraints makes the binary code learning problem quite challenging.

Torralba et al. [18] have introduced the binary coding problem to the vision community and compared several methods based on boosting, restricted Boltzmann machines [14], and locality sensitive hashing (LSH) [1]. To further improve the performance and scalability, Weiss et al. have proposed Spectral Hashing (SH) [21], a method motivated by spectral graph partitioning. Raginsky and Lazebnik [12] have proposed a distribution-free method based on the random features mapping for shift-invariant kernels [13]. This method has theoretical convergence guarantees and has demonstrated superior performance to spectral hashing for relatively large code sizes (64 bits and above). Wang et al. [19] have proposed a semi-supervised hashing method (SSH) that incorporates pairwise semantic similarity and dissimilarity constraints from labeled data.

A common initial step in many binary coding methods is to perform principal component analysis (PCA) to reduce the dimensionality of the data. 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 address this problem, SH [21] 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 unrealistic assumption that the data is uniformly distributed in a high-dimensional rectangle. SSH [19] relaxes the orthogonality constraints of PCA to allow successive projections 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 paper, 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. First, we show that simply applying a random orthogonal transformation to the PCA-projected data, as suggested by Jégou et al. [7], already does a very good job of balancing the variance of different PCA directions and outperforms both SH [21] and non-orthogonal relaxation [19]. Next, we propose an al-



(a) PCA aligned. (b) Random Rotation. (c) Optimized Rotation.

Figure 1. Toy illustration of the proposed ITQ method (see Section 2 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.

ternating minimization approach for refining the initial orthogonal transformation to reduce quantization error. This approach, dubbed **iterative quantization (ITQ)** has connections to the orthogonal Procrustes problem [15] and to eigenvector discretization for multi-class spectral partitioning [22], and in our experiments it outperforms the methods of [12, 19, 21]. Moreover, ITQ can be coupled not only with PCA, but with any projection onto an orthogonal basis. In particular, 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.

The rest of this paper is organized as follows. The ITQ method is described in Section 2. The experimental evaluation presented in Section 3 shows results for the unsupervised scenario, where ITQ is applied to PCA-projected data. Section 4 describes the supervised version of our method based on CCA.

2. Unsupervised Code Learning

In this section, we address the problem of learning binary codes without any supervisory information in the form of class labels. 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 2.1, we follow the maximum variance formulation of [19, 21], which yields PCA projections. The major novelty of our method is in the second step (Section 2.2), where we try to preserve the locality structure of the projected data by rotating it so as to minimize the discretization error. Figure 1 illustrates the idea behind our method.

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 $\operatorname{sgn}(v) = 1$ if $v \geq 0$ and 0 otherwise. 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 $\boldsymbol{w}_k$.

2.1. Dimensionality Reduction

Following the formulation of [19, 21], 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:

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

As shown in [19], 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 [19], 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 exactly the same as that of Principal Component Analysis (PCA). For a code of c bits, we obtain W by taking the top c eigenvectors of the data covariance matrix $X^T X$.

2.2. Binary Quantization

Let $\boldsymbol{v} \in \mathbb{R}^c$ be a vector in the projected space. It is easy to show (see below) that $\operatorname{sgn}(\boldsymbol{v})$ is the vertex of the hypercube $\{-1, 1\}^c$ closest to $\boldsymbol{v}$ in terms of Euclidean distance. The smaller the quantization loss $\|\operatorname{sgn}(\boldsymbol{v}) - \boldsymbol{v}\|^2$, the better the resulting binary code will preserve the original locality structure of the data. Now, going back to eq. (1), it is clear

[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 2.2 requires both the data and the binary cube to be zero-centered.

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

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

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

The idea of rotating the data to minimize quantization loss can be found in Jégou et al. [7]. However, the approach of [7] is based not on binary codes, but on product quantization with asymmetric distance computation (ADC). Unlike in our formulation, direct minimization of quantization loss for ADC 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 for this purpose. 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 iterative quantization (ITQ) procedure 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

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

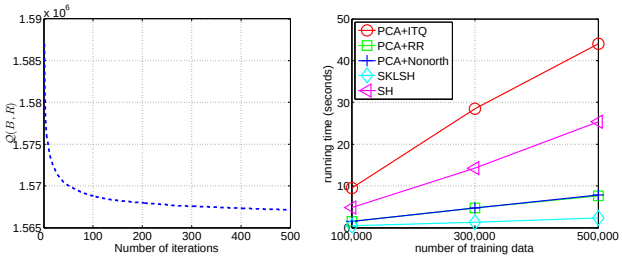
Because the projected data matrix $V = XW$ is fixed, minimizing (3) is equivalent to maximizing

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

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 is compatible with 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 [15], 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 $S\Omega\hat{S}^T$ and then let $R = \hat{S}S^T$.



(a) Quantization error. (b) Training time.

Figure 2. (a) Quantization error for learning a 32-bit ITQ code on the CIFAR dataset (see Section 3.1). (b) Running time for learning different 32-bit encodings on the Tiny Images dataset. The timings were obtained on a workstation with a 2-core Xeon 3.33GHZ CPU and 32G memory.

We alternate between updates to B and R for several iterations to find a locally optimal solution. The typical behavior of the error (2) is shown in Figure 2(a). In practice, we have found that we do not need to iterate until convergence to achieve good performance, and we use 50 iterations for all experiments. Figure 2(b) shows the training time for 32-bit codes for our method and several competing methods that are evaluated in the next section. All the methods scale linearly with the number of images. Although our method is somewhat slower than others, it is still very fast and practical in absolute terms.

Note that the ITQ algorithm has been inspired by the approach of Yu and Shi [22] for discretizing relaxed solutions to multi-class spectral clustering, which is based on finding an orthogonal transformation of the continuous eigenvectors to bring them as close as possible to a discrete solution. One important difference between [22] and our approach is that [22] allows discretization only to the c orthogonal hypercube vertices with exactly one positive entry, while we use all the 2^c vertices as targets. This enables us to learn efficient codes that preserve the locality structure of the data.

3. Evaluation of Unsupervised Code Learning

3.1. Datasets

We evaluate our method on two subsets of the Tiny Images dataset [17]. Both of these subsets come from [4]. The first subset is a version of the CIFAR dataset [8], and it consists of 64,800 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, larger subset consists of 580,000 Tiny Images. Apart from

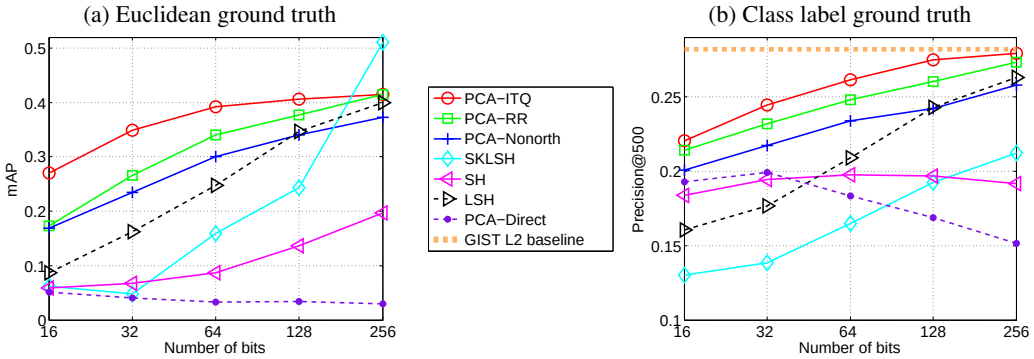


Figure 3. Comparative evaluation on CIFAR dataset. (a) Performance is measured by mean average precision (mAP) for retrieval using top 50 Euclidean neighbors of each query point as true positives. Refer to Figure 4 for the complete recall-precision curves for the state-of-the-art methods. (b) Performance is measured by the averaged precision of top p ranked images for each query where ground truth is defined by semantic class labels. Refer to Figure 5 for the complete class label precision curves for the state-of-the-art methods.

the CIFAR images, which are included among the 580,000 images, 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 4, 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 [11] computed at 8 orientations and 4 different scales, resulting in 320-dimensional feature vectors. Because our method (as well as many other state-of-the-art methods) cannot use more bits than the original dimension of the data, we limit ourselves to evaluating code sizes up to 256 bits.

3.2. Protocols and Baseline Methods

We follow two evaluation protocols widely used in recent papers [12, 19, 21]. The first one is to evaluate performance of nearest neighbor search using Euclidean neighbors as ground truth. As in [12], 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 [20]. 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 5 random training/test partitions.

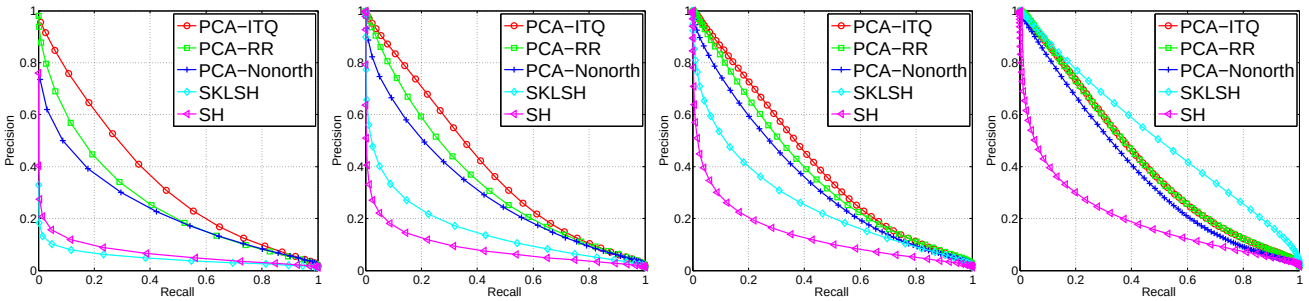
We compare our ITQ method to three baseline methods that follow the basic hashing scheme $H(X) = \mathrm{sgn}(X\widetilde{W})$, where the projection matrix $\widetilde{W}$ is defined in different ways:

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 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 2.2.

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

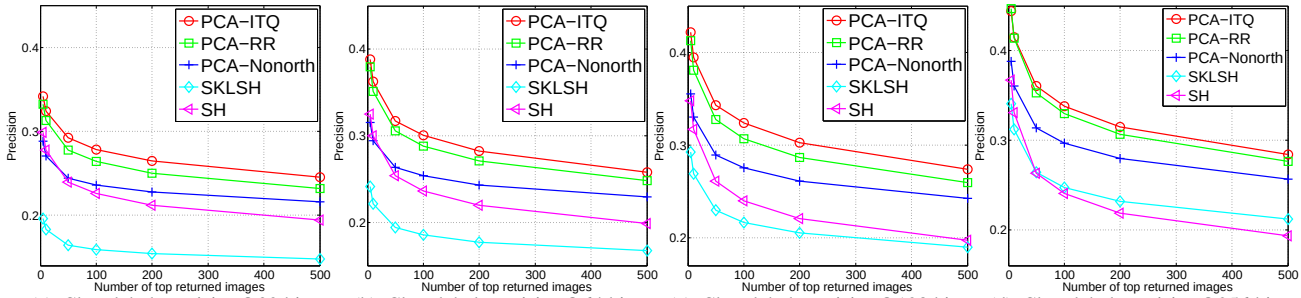
1. **SH** [21]: Spectral Hashing. This method is based on quantizing the values of analytical eigenfunctions computed along PCA directions of the data.
2. **SKLSH** [12]: This method is based on the random features mapping for approximating shift-invariant kernels [13]. In [12], this method is reported to outperform SH for code sizes larger than 64 bits. We use a Gaussian kernel with bandwidth set to the average distance to the 50th nearest neighbor as in [12].
3. **PCA-Nonorth** [19]: Non-orthogonal relaxation of PCA. This method is reported in [19] to outperform SH. Note that instead of using semi-supervised PCA as in [19], the evaluation of this section uses standard unsupervised PCA (a supervised embedding will be used in Section 4).

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-



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

Figure 4. Comparison with state-of-the-art methods on CIFAR dataset using Euclidean neighbors as ground truth. Refer to Figure 3(a) for a summary of the mean average precision of these curves as a function of code size.



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

Figure 5. Comparison with state-of-the-art methods on CIFAR dataset using semantic labels as ground truth. Figure 3(b) shows the summary plot of average precision as a function of code size.

RR and PCA-ITQ, use PCA (or a non-orthogonal relaxation of PCA) as an intermediate dimensionality reduction step.

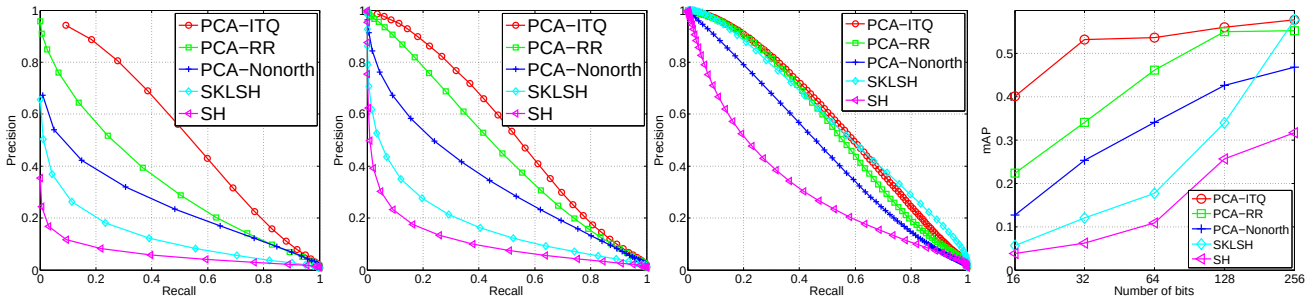
3.3. Results on CIFAR Dataset

Figure 3(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 that when enough bits are assigned, Hamming distance between binary codes approximates distance in the kernel space with high quality. 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 our proposed PCA-ITQ method, 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 3(b) evaluates the semantic consistency of the codes using class labels as ground truth. For each method, it shows retrieval precision for top 500 returned images as a function of code size. It also shows the "upper bound" for the performance of any method, which is given by the retrieval precision of the original uncompressed GIST features with Euclidean distance. As in Figure 3(a), PCA-RR and PCA-ITQ outperform all the other methods, and PCA-ITQ has a small but consistent advantage over PCA-RR. By 256 bits, the precision of PCA-ITQ approaches that of the uncompressed upper bound.

From Figure 3(b), we can also make some interesting observations about the performance of the other methods. Unlike in Figure 3(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 important to apply dimensionality reduction to the data in order to capture the its class structure. Another observation worth making is that the two methods lacking a solid theoretical basis, namely PCA-Direct and SH, can actually *decrease* in performance as the number of bits increases.

Figures 4 and 5 show complete recall-precision and class



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

Figure 6. Results on the 580,000 Tiny Image subset. Ground truth is defined by Euclidean neighbors.

label precision curves corresponding to the summary numbers in Figures 3(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 3 (a,b). One thing that becomes especially apparent from looking at Figure 4(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.

3.4. Results on 580,000 Tiny Images

Figure 6 shows precision-recall curves and mAP for Euclidean neighbor retrieval on the 580,000 Tiny Images. As explained in Section 3.1, 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. Moreover, comparing Figure 6(d) with Figure 3(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.

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. However, even the existing methods that have the best locality preserving properties suffer from very low recall at a low Hamming radius. Table 1 shows the recall and precision of 32-bit codes at Hamming radii 0, 1, and 2 for several methods. We can see that for PCA-ITQ, exact matches already have over 9% recall and over 94% precision – much better

		$r = 0$	$r = 1$	$r = 2$
PCA-ITQ	Recall (%)	**9.31**	**18.43**	**27.82**
	Precision (%)	**94.29**	**88.65**	**80.62**
PCA-RR	Recall (%)	0.10	0.68	2.54
	Precision (%)	95.65	91.00	84.95
PCA-Nonorth	Recall (%)	0.92	5.09	14.99
	Precision (%)	67.30	54.01	42.24
SKLSH	Recall (%)	0.16	1.10	4.26
	Precision (%)	65.56	50.38	36.97
SH	Recall (%)	0.07	0.64	3.01
	Precision (%)	35.35	24.40	16.84

Table 1. Recall and precision for small Hamming radius r (32 bits).

than any of the other methods, including PCA-RR. This is very significant, as it shows that Procrustean adaptation to the data distribution is very effective in bringing close Euclidean neighbors within a small Hamming distance of each other, making the resulting codes usable for hashing.

4. Leveraging Label Information

As discussed earlier, RR and ITQ can be used with any orthogonal basis projection method. In particular, if we have a training dataset with label information available, we can choose a supervised dimensionality reduction method to better capture the semantic structure of the dataset. In this section, we show how to refine our codes in a supervised setting using Canonical Correlation Analysis (CCA) [6], which has proven to be an effective tool for extracting a common latent space from two views [5] and is robust to noise [2]. While the idea of using CCA to perform supervised linear dimensionality reduction prior to binary coding is very simple, to our knowledge, it has not yet been tested in the literature.

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

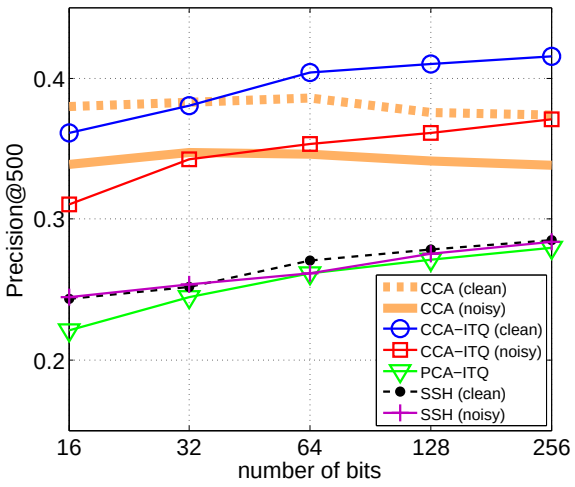


Figure 7. Average precision for top 500 retrieved images for supervised data embeddings based on clean and noisy labels. CCA (clean) and CCA (noisy) are reference curves for uncompressed CCA-projected data.

the correlation between the projected data $X\boldsymbol{w}_k$ and $Y\boldsymbol{u}_k$:

$$\mathcal{C}(\boldsymbol{w}_k, \boldsymbol{u}_k) = \frac{\boldsymbol{w}_k^T X^T Y \boldsymbol{u}_k}{\sqrt{\boldsymbol{w}_k^T X^T X \boldsymbol{w}_k \; \boldsymbol{u}_k^T Y^T Y \boldsymbol{u}_k}}$$
$$s.t. \quad \boldsymbol{w}_k^T X^T X \boldsymbol{w}_k = 1, \; \boldsymbol{u}_k^T Y^T Y \boldsymbol{u}_k = 1 \, .$$

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 [5]. We set ρ to be 0.0001 in this paper. The leading generalized eigenvectors of (4) then give us a sequence of orthogonal $\boldsymbol{w}_k$ directions that span the solution space, just as in the case of 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.

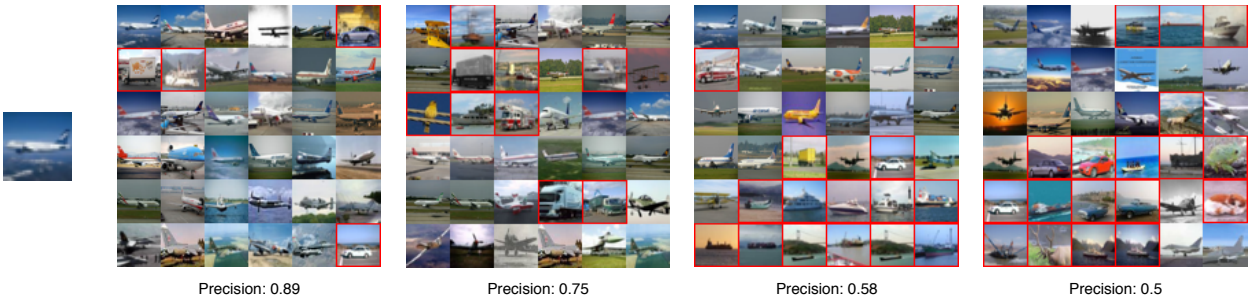
For a c-bit code, we form a projection matrix $\widehat{W} \in \mathbb{R}^{d \times c}$ whose columns are given by the leading eigenvectors $\boldsymbol{w}_k$ scaled by their eigenvalues λ_k.[2] Then we obtain the embedded dataset $V = X\widehat{W}$ in the new latent space that preserves both visual and semantic similarity. Finally, we use the RR and ITQ methods from Section 2.2 to rotate the data to minimize quantization error as before. We refer to the resulting methods as CCA-RR and CCA-ITQ, respectively.

Recall from Section 3.1 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 3.3: namely, we set aside 1000 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 as described above. 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 scenario, and use the "clean" ground-truth labels for benchmarking.

We compare the above approach to the semi-supervised approach of [19], 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 [19], 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 [19], the semi-supervised embedding is combined with nonorthogonal relaxation (SSH-Nonorth), however, just as in Section 3, we have found that SSH-ITQ works better than SSH-Nonorth, so we only reproduce the SSH-ITQ results here.

Figure 7 shows the averaged precision at top 500 retrieved images for the "clean" and "noisy" versions of the CCA and SSH embeddings. As a baseline, we also include the performance of the unsupervised PCA-ITQ 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, SSH produces a very small improvement over PCA, and there is almost no difference in the power of the SSH embeddings learned from clean and noisy data. For reference, this figure also shows retrieval precision curves for uncompressed CCA-projected data for 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 binarization in this case may actually be accomplishing some sort of "semantic hashing" [14], bringing images from the same class to the same hypercube vertex. In the future,

[2] We have found that scaling eigenvectors by their eigenvalues always improves performance for CCA; however, this is not true for PCA.



(a) Query (b) CCA-ITQ. (c) CCA-RR. (d) PCA-ITQ. (e) PCA-RR.

Figure 8. Sample top retrieved images for query in (a) using 32 bits. Red rectangle denotes false positive. Best viewed in color.

we plan to investigate this phenomenon in more detail.

Finally, Figure 8 shows the results of our methods on a sample query. We can clearly see that when labels are incorporated, the results are much more semantically consistent.

5. Discussion

This paper makes several useful findings. First, we show that the performance of PCA-based binary coding schemes can be greatly improved by simply rotating the projected data. Even a random rotation already works better than more elaborate schemes like non-orthogonal relaxation [19]. Second, we demonstrate an iterative quantization method for refining this rotation that is very natural and effective. We evaluate the performance of our method *both* in terms of preserving Euclidean neighbors in the feature space and in terms of retrieving semantically similar images. This evaluation reveals that methods that do very well on the first task, like SKLSH, can actually do quite poorly on the second one. We also show that the classic CCA method [6] gives a good way of utilizing both clean and noisy label information for improving semantic precision. The code and data will be made available online.[3]

At present, one limitation of our method is that it uses one bit per projected data dimension. Unlike randomized data-independent methods such as SKLSH, it cannot use more bits than data dimensions, and converge to the performance of the uncompressed data when enough bits are used. In the future, we would like to bridge the gap between data-dependent methods like ours and data-independent methods like SKLSH to get the best possible performance both for for very small and very large code sizes.

Acknowledgements: We thank Jun Wang, Rob Fergus, Florent Perronnin and Joe Tighe for helpful discussions and for sharing their code and data. This research was supported in part by NSF CAREER Award IIS 0845629, Microsoft Research Faculty Fellowship, Xerox, and ARO.

[3] http://www.unc.edu/∼yunchao/itq.htm.

References

[1] A. Andoni and P. Indyk. Near-optimal hashing algorithms for approximate nearest neighbor in high dimensions. *Communications of the ACM*, 2008.

[2] M. B. Blaschko and C. H. Lampert. Correlational spectral clustering. *CVPR*, 2008.

[3] O. Chum, M. Perdoch, and J. Matas. Geometric min-hashing: Finding a (thick) needle in a haystack. In *CVPR*, 2009.

[4] R. Fergus, A. Torralba, and Y. Weiss. Semi-supervised learning in gigantic image collections. *NIPS*, 2009.

[5] D. P. Foster, R. Johnson, S. M. Kakade, and T. Zhang. Multi-view dimensionality reduction via canonical correlation analysis. *Tech Report. Rutgers University*, 2010.

[6] H. Hotelling. Relations between two sets of variables. *Biometrika*, 28:312–377, 1936.

[7] H. Jégou, M. Douze, C. Schmid, and P. Perez. Aggregating local descriptors into a compact image representation. *CVPR*, 2010.

[8] A. Krizhevsky. Learning multiple layers of features from tiny images. *Tech Report. University of Toronto*, 2009.

[9] B. Kulis and K. Grauman. Kernelized locality-sensitive hashing for scalable image search. In *ICCV*, 2009.

[10] D. Nister and H. Stewenius. Scalable recognition with a vocabulary tree. In *CVPR*, 2006.

[11] A. Oliva and A. Torralba. Modeling the shape of the scene: a holistic representation of the spatial envelope. *IJCV*, 2001.

[12] M. Raginsky and S. Lazebnik. Locality sensitive binary codes from shift-invariant kernels. *NIPS*, 2009.

[13] A. Rahimi and B. Recht. Random features for large-scale kernel machines. *NIPS*, 2007.

[14] R. Salakhutdinov and G. Hinton. Semantic hashing. *SIGIR*, 2007.

[15] P. Schonemann. A generalized solution of the orthogonal Procrustes problem. *Psychometrika*, 31, 1966.

[16] C. Strecha, A. M. Bronstein, M. M. Bronstein, and P. Fua. LDAHash: Improved matching with smaller descriptors. *PAMI*, 2010.

[17] A. Torralba, R. Fergus, and W. Freeman. 80 million tiny images: a large dataset for non-parametric object and scene recognition. *PAMI*, 2008.

[18] A. Torralba, R. Fergus, and Y. Weiss. Small codes and large image databases for recognition. *CVPR*, 2008.

[19] J. Wang, S. Kumar, and S.-F. Chang. Semi-supervised hashing for large-scale image retrieval. *CVPR*, 2010.

[20] J. Wang, S. Kumar, and S.-F. Chang. Sequential projection learning for hashing with compact codes. *ICML*, 2010.

[21] Y. Weiss, A. Torralba, and R. Fergus. Spectral hashing. *NIPS*, 2008.

[22] S. X. Yu and J. Shi. Multiclass spectral clustering. *ICCV*, 2003.