

Mixed Image-Keyword Query Adaptive Hashing over Multilabel Images

XIANGLONG LIU, Beihang University
YADONG MU, Columbia University
BO LANG, Beihang University
SHIH-FU CHANG, Columbia University

This article defines a new hashing task motivated by real-world applications in content-based image retrieval, that is, effective data indexing and retrieval given mixed query (query image together with user-provided keywords). Our work is distinguished from state-of-the-art hashing research by two unique features: (1) Unlike conventional image retrieval systems, the input query is a combination of an exemplar image and several descriptive keywords, and (2) the input image data are often associated with multiple labels. It is an assumption that is more consistent with the realistic scenarios. The mixed image-keyword query significantly extends traditional image-based query and better explicates the user intention. Meanwhile it complicates semantics-based indexing on the multilabel data. Though several existing hashing methods can be adapted to solve the indexing task, unfortunately they all prove to suffer from low effectiveness. To enhance the hashing efficiency, we propose a novel scheme “boosted shared hashing”. Unlike prior works that learn the hashing functions on either all image labels or a single label, we observe that the hashing function can be more effective if it is designed to index over an optimal label subset. In other words, the association between labels and hash bits are moderately sparse. The sparsity of the bit-label association indicates greatly reduced computation and storage complexities for indexing a new sample, since only limited number of hashing functions will become active for the specific sample. We develop a Boosting style algorithm for simultaneously optimizing both the optimal label subsets and hashing functions in a unified formulation, and further propose a query-adaptive retrieval mechanism based on hash bit selection for mixed queries, no matter whether or not the query words exist in the training data. Moreover, we show that the proposed method can be easily extended to the case where the data similarity is gauged by nonlinear kernel functions. Extensive experiments are conducted on standard image benchmarks like CIFAR-10, NUS-WIDE and a-TRECVID. The results validate both the sparsity of the bit-label association and the convergence of the proposed algorithm, and demonstrate that the proposed hashing scheme achieves substantially superior performances over state-of-the-art methods under the same hash bit budget.

Categories and Subject Descriptors: H.3.1 [Information Storage and Retrieval]: Content Analysis and Indexing; H.3.3 [Information Storage and Retrieval]: Information Search and Retrieval; I.2.6 [Artificial Intelligence]: Learning

General Terms: Algorithms, Experimentation

Additional Key Words and Phrases: Multilabel images, mixed image-keyword query, query adaptive hashing, boosting, locality-sensitive hashing

ACM Reference Format:

Xianglong Liu, Yadong Mu, Bo Lang, and Shih-Fu Chang. 2014. Mixed image-keyword query adaptive hashing over multilabel images. *ACM Trans. Multimedia Comput. Commun. Appl.* 10, 2, Article 22 (February 2014), 21 pages.
DOI: <http://dx.doi.org/10.1145/2540990>

X. Liu and B. Lang are supported by the National Major Project of China (2010ZX01042-002-001-00), the SKLSDE Foundation (SKLSDE-2013ZX-05), and the NSFC (61370125).

Authors' address: X. Liu, Beihang University, 37 Xueyuan Road, Haidian, Beijing 100191, China; email: xlliu@nlsde.buaa.edu.cn; Y. Mu, Electrical Engineering Department, Columbia University, New York, NY, 10027; email: muyadong@gmail.com.

Permission to make digital or hard copies of part or all of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies show this notice on the first page or initial screen of a display along with the full citation. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, to republish, to post on servers, to redistribute to lists, or to use any component of this work in other works requires prior specific permission and/or a fee. Permissions may be requested from Publications Dept., ACM, Inc., 2 Penn Plaza, Suite 701, New York, NY 10121-0701 USA, fax +1 (212) 869-0481, or permissions@acm.org.

© 2014 ACM 1551-6857/2014/02-ART22 \$15.00

DOI: <http://dx.doi.org/10.1145/2540990>

1. INTRODUCTION

The deluge of visual data explosively aggregated at online information sites brings great challenges to their storage, indexing and retrieval. In practice many tasks upon such data (e.g., content-based image retrieval) can be formulated as a nearest neighbor search problem in some well-defined feature space. An empirical observation is that finding exact nearest neighbors is returning desirable results, yet computationally expensive. To reduce the complexity, approximate nearest neighbor (ANN) search, giving a good balance between the performance of multimedia system and computational complexity, has been proposed and studied intensively in the literature [Bentley 1975; Indyk and Motwani 1998].

Traditional ANN methods are usually based on tree structures like k -D tree [Bentley 1975], which can achieve efficient search by intelligently eliminating large portions of the search space. However theoretic analysis shows that the complexity will significantly increase when working with high dimensional features [Goodman et al. 2004]. Recently locality sensitive hashing methods (LSH) attract much attention owing to its theoretic property. These methods find a number of hashing functions, mapping the high dimensional data into compact binary codes for both efficient storage and search. In the past decade, various hashing methods are proposed for different scenarios such as unsupervised [Weiss et al. 2008; Gong and Lazebnik 2011; Liu et al. 2011], (semi-)supervised [Mu et al. 2010; Wang et al. 2010a], kernelized [Kulis and Darrell 2009; Kulis and Grauman 2009; He et al. 2010], and multiple features [Song et al. 2011; Zhang et al. 2011; Liu et al. 2012b].

Despite the aforementioned progress in hashing based ANN, developing practical hashing methods for multimedia data still remains a challenging task, due to various complications mainly stemming from the diverse multimedia semantics. The multimedia data show several important characteristics.

- (1) *Multiple Categories*. The number of semantic categories varies for different image benchmarks and multimedia tasks. For instance, ImageNet dataset [Deng et al. 2009] contains more than 10,000 object categories, which significantly complicates the visual retrieval operation.
- (2) *Multiple Labels*. An image or video is associated with several semantic labels, as seen in large-scale image benchmark NUS-WIDE [Chua et al. 2009] and concept-based video set a-TRECVID [Yu et al. 2012]. Note that the notations “multiple categories” and “multiple labels” have subtle differences on the possible number of labels associated with each multimedia data. Since multiple categories can be regarded as a special case of multilabel input, for brevity, we abuse the notation multiple labels to denote both until otherwise mentioned. It is a nontrivial task to design a new indexing algorithm to handle the massive data with such rich semantics.

Prior works have already partly shown the power of taking the diverse multimedia semantics into account during the hashing function construction. For instance, the semi-supervised hashing methods [Mu et al. 2010; Wang et al. 2010a] take advantages of the relative similarity relationship among data and obtain a significant performance leap.

Due to the rich semantics contained in each image, the user intension in image retrieval is possibly ambiguous when only using the image as the query. Recent trends have witnessed more complex, yet effective search paradigm (e.g., the “image+text” mixed query) that better captures the intention of the users. Google has released the “search by image” function (<http://www.google.com/searchbyimage>). The input of the function is a mixture of a query image and several descriptive textual words. These words, clarifying the user intention, could be either provided by the users or inferred by the retrieval system. The final rank of an image is determined by jointly considering two factors: the visual proximity to the query image, and the match between the image’s surrounding text in the web page and the given query words. In practice, the additional textual hint greatly helps reduce the ambiguity of the user intention, and therefore refines the retrieval accuracy. The new image retrieval paradigm as illustrated by the

Table I. Description of Problem Setting
(See text for more explanation.)

	Labels	Visual Features
Database Data	×	✓
Training Data	✓	✓
Query	✓	✓

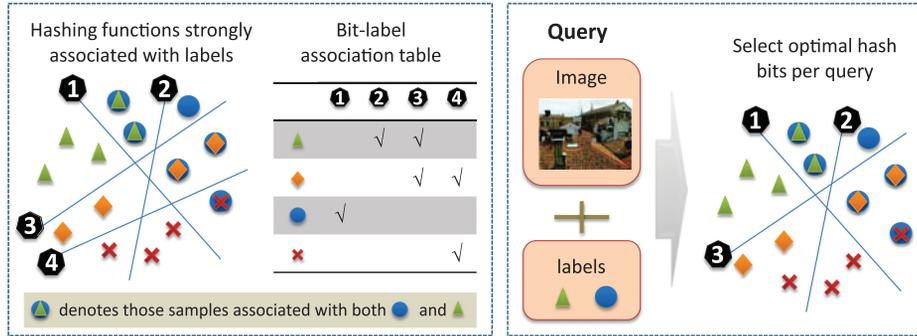


Fig. 1. We propose a novel learning framework to design compact hashing functions sparsely associated with labels. The left panel shows several hashing functions learned over multilabel (denoted by different marks with different colors) training data. Each hashing function is designed especially for only a subset of labels (e.g., function 4 preserves both orange diamond and red cross labels). The final association between labels and hashing functions is summarized in the table. The right panel shows the retrieval with a query consisting of an image and two labels. According to the specific query labels, a subset of most effective hashing functions (function 1, 2, and 3) are selected (note that function 4 will not be chosen since it is not designed for the two labels in the query).

Google product pioneers a new solution to circumvent the semantic gap, yet also proposes serious challenges to effective multimedia (typically in multilabel form) indexing.

In this article, we will concentrate on designing a new hashing method for the new search paradigm over multilabel data. The description of our problem setting is summarized in Table I. There are three types of data in our system: database data, training data, and queries. We use the notation “database data” to refer to a large-scale data pool (typically in millions or even larger scale) from which the most similar images to the query are retrieved. In real-world scenarios such as the Web-sharing images, the label information for most data is too noisy to be reliable and also often partially missing. To make our work applicable for generic data, under our problem setting the label information of database data is assumed to be unavailable. To well model the semantics, a relatively small training set is collected and manually annotated, which enables training high-quality hashing functions and afterwards applying these functions to the database data. When it comes to the retrieval stage, we assume that a query is given as the mixture of the visual feature and a few user-specified labels.

Rare work has been devoted to the new topic in the hashing literature. Though several existing hashing methods can be applicable after proper modification, they suffer from inferior effectiveness. Briefly speaking, prior methods either optimize each individual hashing function over all labels [Mu et al. 2010; Wang et al. 2010b], or select the optimal ones for each label from a large pool of pre-learned hashing functions [Mu et al. 2011]. Unfortunately, neither of them is the most effective way to accomplish the interested task in this article. In the problem setting, the query contains a few labels of interest, and the task somewhat boils down to exploiting hashing functions that are most effective for those labels. In this article we propose a new hashing scheme for better efficacy. The key idea, as illustrated in Figure 1, is to pursue each 1-D subspace that best separates only a subset of correlated labels, rather than all labels or single label. The size of the label subset is adaptively determined.

Compared with alternative solutions, our proposed scheme shows significant advantages in terms of both the computational efficiency (fewer active hash bits) and the retrieval accuracy (mainly owing to the query-adaptive property). In brief, our contributions is threefold.

- (1) We propose a new problem (compact hashing for mixed query over multilabel data) and a reasonable solution with empirical evaluation. It exploits the shared subspaces among the labels to help index multilabel data. Specifically, each hashing function projects the data onto a 1-D linear subspace, where a subset of labels (check Figure 1 for an example) are well separated from other labels. In this way it sparsifies the association between the hashing functions and labels. The sparsification is crucial for improving the multilabel hashing performance since it enables query-adaptive hashing via label-oriented hash bit selection.
- (2) We develop an efficient Boosting-style algorithm to sequentially learn both the hashing functions and their associated subset of labels (referred to as active labels hereafter). The optimization is very efficient and converges fast. Moreover, under the same framework our algorithm can be easily extended to nonlinear kernel hashing.
- (3) At the retrieval stage, given the mixture of image and descriptive labels, we design an efficient mechanism for selecting the most effective hashing functions, based on the learned sparse association between the hash bits and labels.

Note that the whole article extends upon a previous conference publication [Liu et al. 2012c] with additional exploration on the algorithm generalization (general retrieval mechanism, nonlinear hashing extension, etc.), detailed analysis of various algorithmic properties (solution sparsity and convergence rate), and amplified experimental results. The remaining sections are organized as follows. Section 2 reviews related works. In Section 3 we first provide a sketch of the basic idea, and then elaborate on the algorithm, including the multilabel boosted-shared hashing algorithm, a quadratic complexity method for optimized label subset selection, and a query-adaptive bit selection method at retrieval stage. Section 4 analyzes the convergence of the proposed method and its connections to prior research, and then gives its extension to nonlinear hashing. In Section 5, comprehensive experimental results demonstrate the properties of the algorithm (e.g., controllable sparsity and convergence rate), and validate its effectiveness. Finally Section 6 concludes this article.

2. RELATED WORK

The seminal work of Indyk et al. [Indyk and Motwani 1998] attacks the problem of approximate nearest neighbor retrieval for binary vectors equipped with Hamming distance. The so-called locality-sensitive property refers to the fact that LSH algorithms embed similar data under specific metric into similar codes in Hamming space. They use the data structure of multiple hash tables to accomplish a constructive proof for the sub-linear complexity. The most important factor in LSH algorithm design is the underlying metric to gauge data similarity. In the past decade researchers from the theoretic computer science community have explored hashing on a diverse set of metrics, including l_p -norm ($p \in (0, 2]$) [Datar et al. 2004], Jaccard index [Broder et al. 1998], etc. The idea of LSH is also tailored for other scenarios, like kernelized hashing [Kulis and Darrell 2009; Mu et al. 2010; He et al. 2010; Liu et al. 2012b], unsupervised hashing [Weiss et al. 2008; Liu et al. 2011], and (semi-)supervised hashing [Wang et al. 2010a; Mu et al. 2010; Liu et al. 2012a].

2.1 Existing Hashing Methods

Several state-of-the-art algorithms can be used after proper modification to address the multilabel hashing problem given the mixed image-keyword query. As aforementioned in Section 1, these methods can be roughly cast into two categories according to the association between hashing functions and

labels: universal hashing that learns hashing functions universal for all labels, and per-label hashing that respectively learns hashing functions for each specific label.

Universal Hashing. Several works have been devoted to indexing multicategory image data, which are also immediately extended for multilabel data. Representative works include kernel-based hashing with weak supervision [Mu et al. 2010] and semi-supervised hashing [Wang et al. 2010a, 2010b]. The key idea in these works is pushing same-label samples as close as possible in the hash-bit Hamming space, and simultaneously maximizing the separation between different semantic labels. The work of Mu et al. [2010] also supports sparsely-specified pairwise similarity (or dissimilarity) constraints, which are common in many real-world applications.

Sequential projection learning for hashing (SPLH) [Wang et al. 2010b], one of the most related methods in the direction, employs the linear hashing function on centered data $X = \{x_i, i = 1, 2, \dots, N\}$: $h_b(x) = \text{sign}(w^T x)$. Then a series of data-dependent hashing functions $H = \{h_1, h_2, \dots, h_B\}$ is sequentially learned in a Boosting-like procedure by maximizing their empirical accuracy on data X :

$$\sum_b \sum_{(i,j)} S_{ij} \cdot h_b(x_i) h_b(x_j) = \frac{1}{2} \text{tr}(H_b(X) S H_b(X)^T), \quad (1)$$

where $S_{ij} = 1$ if x_i and x_j form a nearest neighbor pair with same labels, and $S_{ij} = -1$ otherwise. S will be empirically updated by imposing larger weights on pairs violated by the current hashing function.

Due to the semantic diversity, it is difficult to have each hashing function significantly benefit all labels. However, great efforts desired to learn a good hashing function can be saved if only a small label subset is targeted. Under the multilabel setting, the targeted label subset also enables query-adaptive hashing schemes, where hashing function selection depends on the query labels (they are assumed to be known as in Google search by image).

Per-Label Hashing. The basic idea is bit selection or bit reweighing. It builds a large pool of random hashing functions and selects a subset of optimal hash bits (possibly reweighing them) for each label. The selection procedure is independent for different labels, therefore this method is scalable for large set of labels and extensible to those labels unseen during training. Representative works include the learned reconfigurable hashing (LRH) [Mu et al. 2011] and the reweighted hashing [Mu et al. 2012].

These methods are effortlessly adaptive to semantic changes by selecting hash bits or adjusting corresponding weights. However, they will be inefficient when processing the multilabel data using the over-complete hash bit pool. As aforementioned, for each label a subset of bits is selected from the large pool of over-complete hash bits, and typically the subsets of different labels seldom heavily overlap with each other. Therefore the number of bits required to store each sample will show a roughly linear increase with respect to the number of labels, which is far from being optimal.

2.2 Shared Subspace Learning

Another line of work that is most relevant to our work is shared subspace learning for multilabel data. Early investigation on shared subspace utilizes the multitask learning framework [Caruana 1997]. The idea also has been exploited in other applications like the boosted multiclass object detector [Torralla et al. 2004], which learns the shared feature across different classes. Moreover, with the abundant multilabel multimedia data, researchers also propose shared subspace learning methods to facilitate multilabel image classification or annotation. For example, Yan et al. [2007a] proposed a model-shared subspace boosting method [Friedman et al. 1998] to reduce the information redundancy in the learning process and Ji et al. [2010] extract the shared structure in multiple categories.

Our work focuses on the query-adaptive hashing over multilabel data by discovering the shared subspace across different semantics. It is motivated by the label-aided image retrieval paradigm and significantly reduces the inter-bit redundancy through the idea of shared subspace learning. This work

can be treated as a flexible and general framework for supervised hashing, under which the typical methods of prior research SPLH and LRH are two special cases by varying the size of the label subset. As far as we know, this is the first work that comprehensively studies the hashing problem for multilabel data with mixed image-keyword query.

3. MULTILABEL BOOSTED SHARED HASHING

In this section we elaborate on the proposed multilabel hashing method. As aforementioned, the key idea is to encourage sparse association between hash bits and labels by exploiting shared subspaces among the labels. First some notations will be introduced in Section 3.1. To generate compact hash codes, Section 3.2 presents a formulation in Boosting style. In each boosting round, when the active labels are known, the pursuit of optimal hashing function is shown to boil down to an eig-decomposition problem, which is efficient to solve. It will further describe a quadratic-complexity strategy to determine the active labels for each hashing function in Section 3.3. A matching-score based bit selection method will be proposed for query-adaptive search at the retrieval stage in Section 3.4.

3.1 Notations

Denote $\langle x_i, l_i \rangle \in \mathbb{R}^D \times \{0, 1\}^L$, $i = 1 \dots N$ to be the i -th feature vector together with the corresponding label vector, where D , N , L are the feature dimension, the numbers of data and labels respectively. Let $l_i(k)$ be the k -th element of l_i . $l_i(k) = 1$ reflects that x_i is associated with the k -th label. For multilabel data, the number of nonzero elements is allowed to be more than one.

Hashing functions. We want to learn a series of hashing functions $H^{(B)}(\cdot) = [h_1(\cdot), h_2(\cdot), \dots, h_B(\cdot)]^T$, where $h_b(\cdot)$ is the b -th hashing function. For tractability we adopt the linear form, that is,

$$h_b(x) = \text{sign}(w^T x), \quad (2)$$

where $\text{sign}(\cdot)$ is the signum function and w is the normal vector of the hashing hyperplane. We omit the thresholding parameters by zero-centering the feature vectors.

The hashing vector w is randomly generated in original p -stable distribution based LSH algorithms [Datar et al. 2004]. Recent years have witnessed the generation of compact hash codes by optimizing w according to specific criterion [Weiss et al. 2008; Wang et al. 2010a; He et al. 2010]. We propose to use a multilabel graph based learning framework to squeeze the most out of each hash bit and preserve the neighbor relationships.

Multilabel graph. The multilabel graph characterizes the neighbor relationships of each sample with respect to each label associated with the sample. It considers two types of nearest neighbors [Yan et al. 2007b], that is, homogeneous neighborhood—nearest neighbors with the same label, and heterogenous neighborhood—nearest neighbors with different labels. Thus with respect to each label there will be one graph depicting the two types of neighbor relationships simultaneously. Figure 2 illustrates the construction of the proposed multilabel graphs for a sample with two labels.

Specifically, for the k -th label, assume that the i -th sample has $l_i(k) = 1$. Let $\mathcal{N}_k^o(i)$ denote its homogeneous neighborhood, which comprises the indices of its k^o -nearest neighbors of the same label, and $\mathcal{N}_k^e(i)$ be the heterogenous neighborhood containing its k^e -nearest neighbors of different labels. The subscript k in $\mathcal{N}_k^o(i)$, $\mathcal{N}_k^e(i)$ reflects that they pertain to the graph of the k -th label.

Active label set. One defining feature of the proposed method is the sparse association between hash bits and the labels. We use the term “active label set” and variable \mathcal{S}_b to indicate those labels associated with the b -th hash bit, namely the b -th hashing function is learned to target at these labels and preserve their neighbor relationships. Figure 7(b) gives such an example on real-world image benchmark by applying the proposed hashing scheme. We postpone algorithmic details of choosing active labels in Section 3.3. In the following presentation, \mathcal{S}_b is assumed to be known for clarity.

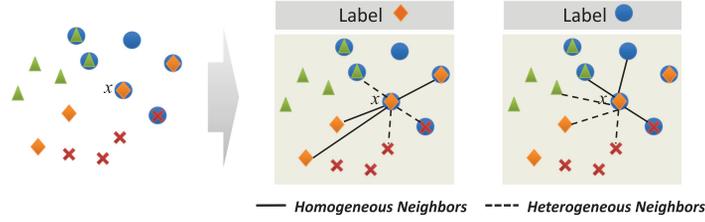


Fig. 2. Illustration of multilabel graph construction on the same training data as Figure 1. The sample x in the left sub-figure has two labels (displayed in blue round and brown diamond), therefore it has outgoing edges in two graphs corresponding to aforementioned labels respectively. For each label, x is connected to its homogeneous neighbors using solid lines and heterogeneous neighbors using dotted lines as shown in the right panel.

3.2 Formulation

Our goal is to optimally learn B hashing functions that can be shared across a label subset. Based on the multilabel graphs and active label set S_b , we first employ the exponential loss function for all hash bits on the graphs of S_b .

Exponential-loss oriented cost function. Intuitively, a sample is encouraged to share the same hash values with its homogeneous neighbors (in other words, small Hamming distance between their hash codes), and have significant different hash codes compared with its heterogeneous neighbors. This likelihood can be measured via the following function:

$$F^{(B)}(x_i, x_j, k) = \sum_{b=1}^B \beta \cdot \delta[k \in S_b] \cdot h_b(x_i) \cdot h_b(x_j), \quad (3)$$

where k denotes one of the labels that x_i is associated with, β is a constant weighted parameter for each hashing function that controls the convergence rate (more detailed discussion will be given in next section), and $\delta(x)$ is the delta function that returns 1 if x is true, otherwise 0. Obviously the delta function serves as the gating function on S_b for hash bit selection, which is consistent to the idea illustrated in Figure 7(b). If we define $f^{(b)}(x_i, x_j, k) = \beta \cdot \delta[k \in S_b] h_b(x_i) \cdot h_b(x_j)$ and let $F^{(0)}(x_i, x_j, k) = 0$, then $F(x_i, x_j, k)$ will be additive. At the b -th iteration it can be written as

$$F^{(b)}(x_i, x_j, k) = F^{(b-1)}(x_i, x_j, k) + f^{(b)}(x_i, x_j, k). \quad (4)$$

Following the margin maximization widely adopted in distance metric learning [Yan et al. 2007b; Weinberger and Saul 2009], we consider the Hamming distance between homogeneous and heterogeneous neighbors, and define the cost function $c(x_i)$ on x_i and its neighbors:

$$c(x_i) = \sum_{\forall k, l_k(i)=1} \left(\frac{1}{k^o} \sum_{j \in \mathcal{N}_k^o(i)} e^{-F(x_i, x_j, k)} + \frac{1}{k^e} \sum_{j \in \mathcal{N}_k^e(i)} e^{F(x_i, x_j, k)} \right). \quad (5)$$

Exponential function is adopted in this equation owing to its robustness and the convenience for incremental update. Let $\mathcal{N}_k(i) = \mathcal{N}_k^o(i) \cup \mathcal{N}_k^e(i)$. For conciseness consideration, we ignore the effect of k^o, k^e and introduce an auxiliary variable z_{ij} that takes value -1 if $j \in \mathcal{N}_k^e(i)$, otherwise 1. The cost function for all samples can be represented as follows:

$$\mathcal{J} = \sum_{i=1}^N c(x_i) = \sum_{i=1}^N \sum_{\forall k, l_k(i)=1} \sum_{j \in \mathcal{N}_k(i)} e^{-z_{ij} F(x_i, x_j, k)} = \sum_{(i, j, k) \in \mathcal{I}} e^{-z_{ij} F(x_i, x_j, k)}, \quad (6)$$

where \mathcal{I} denotes the index set of all M valid triple sets (i, j, k) corresponding to all neighbor pairs.

Taylor expansion and updating rule. Similar to classification, a powerful way to minimize such exponential loss function is boosting. Following the spirit of the well-known AdaBoost algorithm [Freund and Schapire 1995; Friedman et al. 1998], the hashing function $h_b(\cdot)$, $b = 1 \dots B$ is learned in a sequential manner. Since $F(x_i, x_j, k)$ is additive, any binary term in Equation (6) can be simplified as

$$e^{-z_{ij}F^{(b)}(x_i, x_j, k)} = e^{-z_{ij}F^{(b-1)}(x_i, x_j, k)} \cdot e^{-z_{ij}f^{(b)}(x_i, x_j, k)} = \pi^{(b-1)}(i, j, k) \cdot e^{-z_{ij}f^{(b)}(x_i, x_j, k)}, \quad (7)$$

where the variable π is introduced to maintain the weights over all neighbor pairs. It can be normalized to be a probabilistic distribution and estimated from previous $b - 1$ iterations:

$$\pi^{(b)}(i, j, k) = \frac{1}{\mathcal{T}^{(b)}} \pi^{(b-1)}(i, j, k) \cdot e^{-z_{ij}f^{(b)}(x_i, x_j, k)}, \quad (8)$$

where $\mathcal{T}^{(b)} = \sum_{(i,j,k) \in \mathcal{I}} \pi^{(b-1)}(i, j, k) \cdot e^{-z_{ij}f^{(b)}(x_i, x_j, k)}$ and $\pi^{(0)}(i, j, k) = \frac{1}{M}$.

From the Taylor expansion, we can further obtain (the following approximation results from abandoning high-order terms in Taylor expansion):

$$e^{-z_{ij}f^{(b)}(x_i, x_j, k)} \approx 1 - z_{ij}f^{(b)}(x_i, x_j, k) + \frac{z_{ij}^2 (f^{(b)}(x_i, x_j, k))^2}{2}. \quad (9)$$

After Taylor expansion, the loss of the b -th function on the active label set S_b can be rewritten as

$$\sum_{(i,j,k) \in \mathcal{I}, k \in S_b} \pi^{(b-1)}(x_i, x_j, k) \cdot \left(1 - z_{ij}f^{(b)}(x_i, x_j, k) + \frac{z_{ij}^2 (f^{(b)}(x_i, x_j, k))^2}{2} \right). \quad (10)$$

3.3 Optimization

Under the boosting framework, both the optimal active label set S_b and the optimal hashing function should be learned to minimize the loss in the previous equation. We first present how to find the optimal hashing function when given S_b .

Optimal hashing function. Given $k \in S_b$ and β , since both the variable z_{ij} and function $f^{(b)}(x_i, x_j, k)$ are discrete, ranging over $\{-1, 1\}$ and $\{-\beta, \beta\}$, the equation can be further written as

$$e^{-z_{ij}f^{(b)}(x_i, x_j, k)} \approx -z_{ij}f^{(b)}(x_i, x_j, k) + \text{const}. \quad (11)$$

We relax the signum function to be continuous following Mu et al. [2010] and Wang et al. [2012], that is, $f^{(b)}(x_i, x_j, k) = \beta \cdot \text{sign}(w^T x_i) \cdot \text{sign}(w^T x_j)$ is relaxed to $\beta w^T x_i x_j^T w$. The optimization problem for the b -th iteration is as follows:

$$\min_w \mathcal{J}^c = w^T \left(- \sum_{(i,j,k) \in \mathcal{I}, k \in S_b} \pi^{(b-1)}(x_i, x_j, k) \cdot z_{ij} \cdot x_i x_j^T \right) w. \quad (12)$$

Such a problem can be efficiently solved by eig-decomposition, where w is the eigenvector corresponding to the smallest eigenvalues of $-\sum_{(i,j,k) \in \mathcal{I}, k \in S_b} \pi^{(b-1)}(x_i, x_j, k) \cdot z_{ij} \cdot x_i x_j^T$.

Active label set selection. To find the optimal active label set, an intuitive way is to exhaustively compare all loss functions on possible 2^L subsets (recall that L is the number of labels). However, due to the exponential complexity, it is impractical for large-scale retrieval with a number of labels. We elaborate on a scalable algorithm for selecting S_b in $O(L^2)$ steps instead of $O(2^L)$. The scalability to large label set mainly stems from a greedy selection strategy. At the beginning, the active label set S_b is initialized to be the label that best minimizes Problem (12), and the optimal w^* is recorded. At each following iteration, it expands S_b by adding another label. Thus there are at most L iterations, and in

each iteration \mathcal{J} in Equation (6) is calculated using the relaxation in Equation (11) with the temporary active label set $S_b \cup \{k\}$, $k \in \{1, \dots, L\} \setminus S_b$ (i.e., totally $O(L)$ attempts in each iteration). The one that most decreases \mathcal{J} is selected, and w^* is recorded. The procedure is terminated when the gain is below τ (e.g., $\tau = 5\%$) compared with the previous iteration. Totally we need solve Problem (12) less than $O(L^2)$ times. Algorithm 1 lists the selection method.

ALGORITHM 1: Active Label Set Selection.

```

1:  $S_b = \emptyset$ ;
2: while  $|S_b| \leq L$  do
3:   for any un-selected label  $k \in \{1, \dots, L\} \setminus S_b$  do
4:     Calculate  $w^*$  by solving Problem (12) with active label set  $S_b \cup \{k\}$ ;
5:     Calculate the decrease amount of  $\mathcal{J}$  in Equation (6) with  $S_b \cup \{k\}$  and  $w^*$ ;
6:   end for
7:   Terminate if the decrease is below prespecified threshold  $\tau$ ;
8:   Update  $S_b = S_b \cup \{k^*\}$  by select the label  $k^*$  with maximal decrease;
9: end while

```

The proposed selection procedure is quite similar to these of the greedy based sparse coding algorithms like (orthogonal) matching pursuit [Pati et al. 1993] and the active set selection [Lee et al. 2007]. Similar to the aforementioned methods, ours naturally ensures the sparsity of the label-bit association. Nevertheless, our method distinguishes from others by adopting different selection criterion for next sample or label. Particularly, we adopt the neighbor prediction error of hash codes, unlike the criteria in others' work (e.g., reconstruction error of sparse codes).

Finally, in the boosting framework, both hashing functions and their active label sets can be learned sequentially. The whole boosted shared hashing algorithm is shown in Algorithm 2.

ALGORITHM 2: Boosted Shared Hashing.

```

1: Initialize the weights  $\pi^{(0)}(i, j, k) = \frac{1}{M}$  for  $(i, j, k) \in \mathcal{I}$  and normalize them to form a probability distribution;
   Set  $F(x_i, x_j, k) = 0$ ;
2: for  $b = 1, 2, \dots, B$  do
3:   Learn the active label set  $S_b$  and the hashing vector  $w_b$  using Algorithm 1;
4:   Update the weights and normalize them:  $\pi^{(b)}(i, j, k) = \frac{1}{T^{(b)}} \pi^{(b-1)}(i, j, k) \cdot e^{-z_{ij} f^{(b)}(x_i, x_j, k)}$ ;
5:   Update  $F(x_i, x_j, k)$  via Equation (4);
6: end for

```

3.4 The Retrieval Stage

Section 3.3 has presented the algorithms for learning the hashing functions and active label sets, which encourage sparse association between the hash bits and labels. We denote the learned bit-label association matrix by $A \in \{0, 1\}^{L \times B}$, whose nonzero elements in the b -th column correspond to the active label set S_b of the b -th hashing function. When a query (x_q, l_q) comes, the retrieval task boils down to selecting a specific number of hash bits from the learned B hash bits.

A straightforward selecting way is to find the associated bits for each query label in l_q and then taking their union. However, this strategy usually fails to rank the candidate bits, especially when more than the budgeted number of candidates are available. Moreover, the number of associated bits in matrix A is varying for different labels, which complicates adaptively choosing proper number of hash bits for the retrieval. To address this nontrivial task, we give an efficient scoring scheme to assess the match between the query and each bit.

3.4.1 Retrieval Using Database Labels. Supposing that the query labels come from the labels in the training set (named database labels), we propose to use a simple matching-score based bit ranking method for bit selection. A matching score is computed between b -th bit-label association vector $A^{(b)}$ (b -th column of matrix A) and l_q , as follows:

$$s_J(A^{(b)}, l_q) = \frac{|A^{(b)} \cap l_q|}{|A^{(b)} \cup l_q|}, \quad (13)$$

where $|\cdot|$ denotes the number of nonzero elements in a set. \cap, \cup represent the intersection and union operations on two sets respectively. Recall that both $A^{(b)}$ and l_q are binary vectors. Intuitively, s_J calculates the common labels between them, penalized by their union. It is known as the Jaccard index in the hashing literature. The more labels the two binary vectors share, the more confident the hashing function will be active. Therefore, given a query (x_q, l_q) , the algorithm greedily selects specific number of hash bits with highest scores from A by sorting the scores $s_J(A^{(b)}, l_q)$ in descending order.

3.4.2 Retrieval Using Unseen Labels. In practice, it is infeasible for hashing methods to simultaneously learn good hash functions for a large number of labels at the training stage, due to both the difficulty of collecting sufficient training data and the complexity of semantic labels. To support those new labels (named unseen labels) that are unknown when training (one typical scenario is that the data with unseen labels appear sequentially), we propose a simple, yet efficient scheme to update the retrieval system by incorporating the new data based on those learned hashing functions on the database labels, so that our method can also handle queries with unseen labels. First the correlations between unseen and database labels are discovered, and then at retrieval stage the database labels that are highly correlated with the unseen labels are selected as the most proximal semantics for queries, which will help achieve fast hash function adaption for unseen labels.

Suppose we can collect a small training data set with L_u unseen labels, and in practice L_u might be much larger than L . Let $\omega_i \in \mathbb{R}^D, i = 1, \dots, L_u$ be the statistical mean vector of all the feature vectors associated with the i -th unseen label, and $\Omega \in \mathbb{R}^{D \times L}$ be a matrix, whose columns correspond to the statistical mean vectors of the training data associated with each database label. Motivated by the idea of sparse subspace clustering [Elhamifar and Vidal 2009] that each data point in certain subspace should be represented as a linear combination of other points in the same subspace, we formulate and solve the following problem to learn the correlations between unseen labels and database labels:

$$\min_{\Theta \geq 0} \sum_{i=1}^{L_u} (\|\omega_i - \Omega \theta_i\|_F^2 + \lambda \|\theta_i\|_1) - \mu \text{tr} \left(\Theta \left(I - \frac{1}{L_u} \mathbf{1}\mathbf{1}^T \right) \Theta^T \right). \quad (14)$$

In the first term of the given formulation, $\Theta = (\theta_1, \dots, \theta_{L_u}) \in \mathbb{R}^{L \times L_u}$ are the recovery coefficients for each unseen label, and $\sum_{i=1}^{L_u} (\|\omega_i - \Omega \theta_i\|_F^2)$ is the reconstructing error, minimizing which leads to the desirable Θ that can well reconstruct each ω_i of unseen labels using Ω of all database labels. In addition, an ℓ_1 regularizer on the coefficients is appended to make the correlations between unseen and database labels sparse. It is noted that different unseen labels possess varied correlations with the same database label owing to their semantic diversity, which implies a large variance on their corresponding coefficients. Therefore, the last term intuitively encourages the diversity over different columns of Θ by maximizing their variance $\text{tr}(\Theta(I - \frac{1}{L_u} \mathbf{1}\mathbf{1}^T)\Theta^T)$, where I is an identity matrix.

Though the entire objective function is nonconvex due to the last term, it can be recognized as a Concave-Convex Procedure (CCCP) [Yuille 2002] and efficiently obtains a local optimum by iteratively solving a series of convex sub-problems. In practice, we use a simple alternating optimization strategy.

In each iteration, only one column of Θ is active, and others are kept as known. A local optimum of the whole problem can be obtained efficiently by iteratively solving quadratic programming.

The learned θ_i weights the informativeness of each database label for recovering the i -th unseen label. Given a query containing unseen labels, we can form a query vector l_q by selecting the most informative database labels S_q with largest coefficients. This strategy also works for retrieval using database labels, where their most correlated database labels will be themselves. With l_q , the aforementioned bit selection method for database label retrieval (see Algorithm 3) can be applied immediately.

ALGORITHM 3: Query-Adaptive Retrieval.

- 1: **for** $b = 1, 2, \dots, B$ **do**
 - 2: Calculate $s_j(A^{(b)}, l_q)$ for b -th hashing function and query l_q according to Equation (13);
 - 3: **end for**
 - 4: Select budgeted k_b hashing functions with highest scores and corresponding hash bits for database images;
 - 5: Generate hash bits for query using selected hashing functions;
 - 6: Retrieve top ranked images using hash-lookup or Hamming ranking.
-

4. ANALYSIS AND EXTENSION

4.1 Convergence

The proposed algorithm updates the weights on neighbor pairs in each round and sequentially learns all budgeted hashing functions. It turns out that when setting the additive weighted parameter β dynamically as Adaboost does, we can guarantee a fast convergence using our boosting-style algorithm.

First let define $r^{(b)} = \sum_{(i,j,k) \in \mathcal{I}, k \in \mathcal{S}_b} \pi^{(b-1)}(i, j, k)$, which is the sum of all weights on pairs with the label in \mathcal{S}_b . The error rate of b -th bit on these data can be calculated by

$$\epsilon^{(b)} = \frac{1}{r^{(b)}} \sum_{\substack{(i,j,k) \in \mathcal{I}, k \in \mathcal{S}_b \\ z_{ij} f^{(b)}(x_i, x_j, k) < 0}} \pi^{(b-1)}(i, j, k), \quad (15)$$

and the accuracy rate is $\xi^{(b)} = 1 - \epsilon^{(b)}$ ($\xi^{(b)} \geq \epsilon^{(b)}$). By minimizing the loss in (7), we get

$$\beta^{(b)} = \frac{1}{2} \ln \left(\frac{\xi^{(b)}}{\epsilon^{(b)}} \right). \quad (16)$$

We adopt the definition of the training error that is very similar to binary classification error:

$$\mathcal{E} = \sum_{(i,j,k) \in \mathcal{I}} \delta \left[z_{ij} F^{(B)}(x_i, x_j, k) \leq 0 \right]. \quad (17)$$

Following Freund and Schapire [1995], it can be shown that the training error is bounded above by

$$\mathcal{E} \leq M e^{-\sum_{b=1}^B r^{(b)} (\sqrt{\xi^{(b)}} - \sqrt{\epsilon^{(b)}})^2} \quad (18)$$

with $\xi^{(b)} + \epsilon^{(b)} = 1$ and $r^{(b)} < 1$. As the number of learned hashing functions increases, the upper bound of the training error will decrease monotonically, and guarantee the convergence.

4.2 Connections with Previous Work

As we claimed before, the proposed algorithm serves as a general framework for supervised hashing. For instance, it has connections to existing algorithms like SPLH [Wang et al. 2010b]. Particularly, by

setting $R_{ij} = -\sum_{k \in S_b} \pi^{(b-1)}(x_i, x_j, k) \cdot z_{ij}$ and $H_b(X) = [h_b(x_1), \dots, h_b(x_N)]^T$, the optimization problem in Equation (12) can be reformulated as

$$\min_w \mathcal{J}^c = - \sum_{(i,j,k) \in \mathcal{I}, k \in S_b} \pi^{(b-1)}(x_i, x_j, k) \cdot z_{ij} \cdot w^T x_i x_j^T w = \frac{1}{2} \text{tr}(H_b(X) R H_b(X)^T). \quad (19)$$

The formulation is very similar to that of SPLH [Wang et al. 2010b] (see Equation (1)), where we can also incorporate unsupervised information. Both methods sequentially learn the hashing functions in a boosting(-like) way. However, there exist two main differences between them:

- (1) Since each hashing function in our proposed method only concentrates on a subset of labels, R will be very sparse. On the contrary, SPLH tries to benefit all labels simultaneously with a dense S in Equation (1). Sharing bits only across a small number of targeted labels makes our method both query-adaptive and computationally efficient.
- (2) Instead of explicitly updating the matrix R in a heuristic way in SPLH, our method derives the optimal update from the exponential loss function following Adaboost, and thus guarantees the convergence. Therefore, compared to SPLH the proposed hashing approach possesses significant superiorities in terms of both the computational efficiency and the retrieval accuracy.

4.3 Nonlinear Hashing Extension

The optimization problem in (19) can be generalized to handle kernelized data, whose pairwise similarity is gauged by the inner product in specific reproducing kernel Hilbert space. Formally, the hashing function can be written in the form of

$$h_b(x) = \text{sign}(w^T \varphi(x)), \quad (20)$$

where $\varphi(\cdot)$ is a mapping function that can implicitly and nonlinearly up-lifts the feature dimension. We also abuse the notation w here to denote the hashing vector in the kernel space. The kernel-based representation potentially improves the data characteristics [He et al. 2012]. Prior research [Kulis and Darrell 2009; Mu et al. 2010; He et al. 2010; Liu et al. 2011] has indicated that applying such nonlinear feature map helps learn high-quality hashing functions and achieve better performances.

According to the representer theorem, an optimal solution w of such a problem is in the span of the training data. It will be computationally expensive to use all training data, and thus we employ an approach following prior research [Kulis and Grauman 2009; He et al. 2010; Liu et al. 2011] to represent w as a linear combination of $\varphi(\hat{x}_i)$ in kernel space:

$$w = \sum_{i=1}^{N_l} v_i \varphi(\hat{x}_i), \quad (21)$$

where $\hat{x}_i \in \mathbb{R}^D$, $i = 1, \dots, N_l$ are the landmarks in the feature space, and usually generated by sampling or clustering. In practice, $N_l \ll N$, and is fixed to a proper value from 300 to 1,000 for a tradeoff between the computation and the accuracy [He et al. 2010; Liu et al. 2011].

Given the kernel function $K(x_i, x_j) = \varphi(x_i)^T \varphi(x_j)$, we can rewrite the hashing functions using the kernel definition instead of the explicit nonlinear mapping:

$$h_b(x) = \text{sign} \left(\sum_{i=1}^{N_l} v_i K(x, \hat{x}_i) \right). \quad (22)$$

Substituting the kernelized hashing function into the optimization problem in Equation (19), we get the following problem with a kernel matrix $K \in \mathbb{R}^{N_i \times N}$ between N samples and N_i landmarks:

$$\min_v \mathcal{J}^c = \frac{1}{2} \text{tr}(v^T K R K^T v). \quad (23)$$

The optimal v turns to the eigenvector of $K R K^T$ corresponding to the smallest eigenvalue, and thus the whole boosted shared hashing algorithm with adaptive retrieval can be adopted directly.

5. EVALUATION AND RESULTS

In this section we evaluate our proposed boosted-shared hashing (denoted as BSH hereafter) on both multicategory and multilabel image benchmarks. Specifically, we choose three large-scale image benchmarks: one multicategory dataset CIFAR-10 [Krizhevsky 2009] (60K images), and two multilabel benchmarks NUS-WIDE [Chua et al. 2009] (270K images) and a-TRECVID [Yu et al. 2012] (260K images). In all experiments we follow the widely-used feature types of each dataset. As the preprocessing, feature vectors are all zero-centered and unit-normalized.

Since there is no hashing scheme specially designed for multilabel data, we adopt five state-of-the-art universal hashing methods: sequential projection learning for hashing (SPLH) [Wang et al. 2010b], iterative quantization (ITQ) [Gong and Lazebnik 2011], binary reconstruction embedding (BRE) [Kulis and Darrell 2009], spectral hashing (SH) [Weiss et al. 2008], and random projection based locality sensitive hashing (LSH) [Datar et al. 2004], and a per-hashing method: learned reconfigurable hashing (LRH) LRH [Mu et al. 2011] as the baseline algorithms. All methods are properly tailored before they are applied to the multilabel data. It is well-known that hashing performance heavily relies on the number of hash bits used for a query. Therefore, towards a fair comparison, all hashing schemes in the experiments are contrasted with respect to the same budget of hash bits per query.

Following Yan et al. [2007b], the heterogenous neighbor number k^e is set twice of the homogeneous neighbor number k^o to compensate the heavy unbalance between the numbers of the two types of neighbor pairs. In our experiments, k^o , k^e are set to be 15, 30 respectively without tuning. The regularization parameter in SPLH is critical for its performance, and thus will be tuned before comparison. To reduce the effect of randomness, we estimate the performance by averaging 10 independent runs.

5.1 Retrieval Using Database Labels

We first comprehensively study the proposed hashing for the retrieval scenario using database labels. Both the multicategory and the multilabel datasets are considered in this experiment.

5.1.1 Multicategory Retrieval. CIFAR-10 is a subset of the 80-million tiny images originally constructed for learning meaningful recognition-related image filters whose responses resemble the behavior of human visual cortex [Krizhevsky 2009]. The dataset is comprised of 60,000 32×32 color images from 10 semantic categories (e.g., ‘airplane’, ‘frog’, etc.). There are 6,000 images for each category, from which 3,000 images are randomly sampled for training the hashing functions. We also sample 1,000 samples in total as the queries. Regarding the visual features, we extract 384-D GIST features [Oliva and Torralba 2001] from these images as Mu et al. [2010] and Liu et al. [2012a] did.

Generally, the performance of a hashing scheme is gauged based on various statistics about “good neighbors.” On CIFAR-10, we define “good neighbors” as those samples from the same category to the query. In Figure 3(a) and (b), we report the recall performances with 16 and 32 hash bits for each query in terms of the number of “good neighbors”. It can be observed that both BSH and its kernel version (KBSH for short, using a gaussian kernel with 500 landmarks generated by k-means clustering on

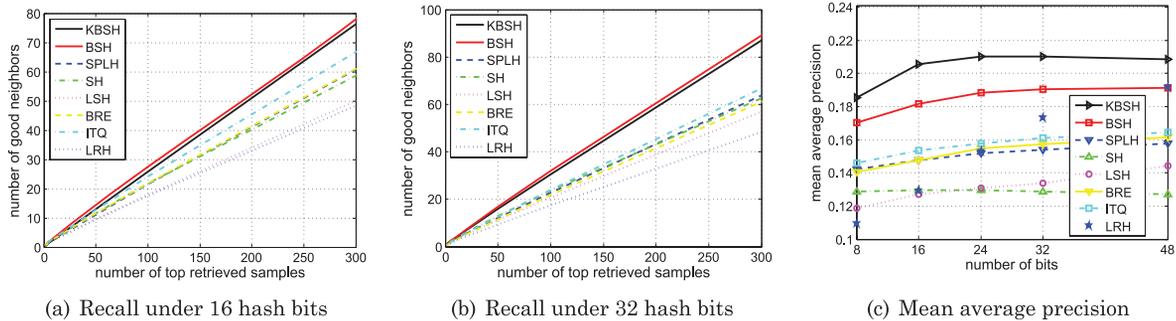


Fig. 3. Performances comparison of different hashing schemes on CIFAR-10. (a) and (b) are the recall results using 16 and 32 hash bits per query, and (c) presents the performances with varying numbers of hash bits.

the training data) significantly outperform all universal hashing baselines, and KBSH performs best, owing to its powerful data representation. For per-label hashing LRH, if we encode each semantic space of the ten categories of CIFAR-10 using 16 bits, then totally 160 bits are required to store each image, which is quite close to the total bit budget we used for both KBSH and BSH (i.e., 150 hash bits). But from Figure 3 we can observe that in this case LRH is much inferior to BSH. Though LRH might achieve better performance when using more than 48 bits per query, the total number of bits for indexing becomes much larger than that of (K)BSH. Note that in our experiments we set $\beta = 0.3$ without fine tuning. Better hashing performance can be obtained using a more proper β .

It is also important to study the tendency of the performance with respect to the increasing number of hash bits per query. Figure 3(c) summarizes the mean average precision (MAP) of all methods using hash bits ranging from 8 to 48. All performances increase as more hash bits are used, but our proposed scheme is consistently superior to others with more than 8 bits, especially when using the kernelized extension. Here, compared SPLH that spends 1.1s on learning each hash function, although BSH consumes more training time (5.3s), it produces more discriminate functions. Moreover, its testing time (2.5×10^{-2} s) is close to that of SPLH. Therefore on the whole, we conclude that BSH and KBSH can attain significant performance gains over other methods like ITQ in terms of both recall and precision.

5.1.2 Multilabel Retrieval. We choose the large-scale NUS-WIDE benchmark [Chua et al. 2009] for the multilabel image retrieval. There are totally 81 tags (hereafter we use the term “tag” and “label” interchangeably until otherwise mentioned) annotated manually across the entire dataset, and each image is associated with a varying number of tags. We represent each image by concatenating the provided features including 500-D SIFT-based bag-of-words feature and 225-D block-wise color moment feature. These features are widely adopted in image search and classification research [Li et al. 2013]. Furthermore, following Liu et al. [2011] we select 25 most-frequent tags (‘sky’, ‘clouds’, ‘person’, etc.), each of which has several thousands of associated images. We randomly select 5,000 images to build a training set and 1,000 images as a query set. In the experiments, we set the regularization parameter η of SPLH to be 8×10^{-3} after fine tuning. The label matrix S in SPLH is created by summarizing all the neighbor relationships of different classes.

One defining feature of our work is the new retrieval paradigm with mixed image-keyword queries. To quantitatively study our proposed solution to this problem, we generate queries from those images associated with at least one tags. For each query image, one and two tags are randomly selected from all its tags to form one- and two-label queries (the proposed method is trivially extended to queries

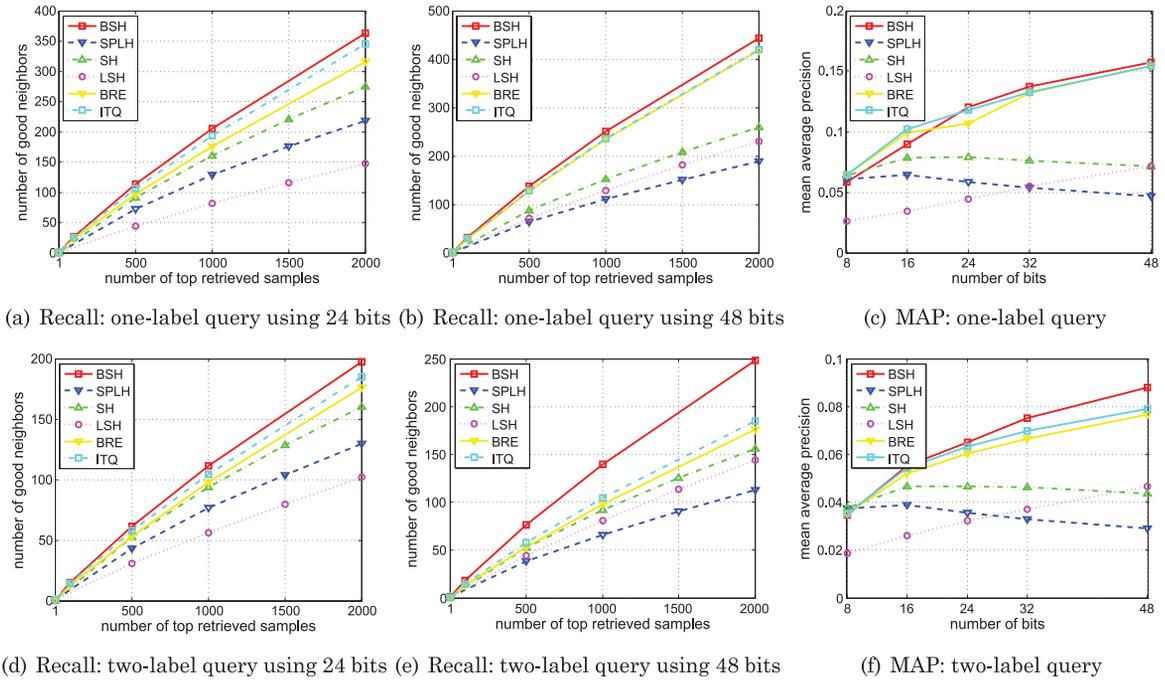


Fig. 4. Performances of BSH and baselines on NUS-WIDE image benchmark with one- and two-label queries.

with more than two labels). The groundtruth for each query is defined as the set of images that satisfy two requirements:

- (1) The images should be associated with same tags of the query;
- (2) Their distances to the query in terms of visual features should be short enough (we use 3,000 nearest neighbors).

Figure 4 shows the performances of our proposed method and five baselines. Suppose k_b hash bits are used for each query, our proposed BSH first learns less than $6k_b$ hash bits, which serve as the bit pool for query-adaptive bit selection. We study the recall performances in terms of good neighbor number for one- and two-label queries using 24 and 48 hash bits, presented in Figure 4(a), (b), (d), and (e) respectively. Figure 4(c) and (f) plot the MAP values under varying hash bits per query. It is noticed that as the bit number increases, both the recall and precision performances are improved, and our BSH method outperforms all other baselines in all experiments.

As aforementioned, we adopt the retrieval paradigm that takes the mixture of image and keywords as the query. It models the user intention more accurately and addresses the problems of conventional hashing schemes that neglect the multilabel information. An illustrating example from NUS-WIDE is found in Figure 5, where the image is associated with three tags. Assume the users are only interested in part of the semantic tags associated with the query image (which is the common case in real-world image retrieval), conventional hashing schemes tend to fail since they are not query-adaptive. In Figure 5, we generate two different queries, for instance, “image visual feature + ‘Ocean’ + ‘Sky’” and “image visual feature + ‘Ocean’ + ‘Person’” for the query image. Top 10 returned images obtained by different hashing schemes are displayed. Our method is able to select query-adaptive hash bits and therefore gets more reasonable results matching both the query image and its keywords.

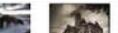
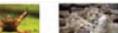
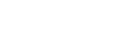
Query Image	Algo.	Label	Top-10 Retrieval Results									
 	BSH											
												
	SPLH											
												
	SH											
												

Fig. 5. An illustrating example of image retrieval using mixed image-keyword query over multilabel data.

5.2 Retrieval Using Unseen Labels

In the following experiments, we evaluate the performance of our hashing method with the adaptive retrieval using unseen labels. Here we employ **a-TRECVID** [Yu et al. 2012], a large-scale image dataset with more labels than either CIFAR-10 or NUS-WIDE. It is built from the TRECVID 2011 Semantic Indexing (SIN) annotation set, including 126 fully labeled concepts on 0.26 million images. Each image is associated with a varying number of concepts.

Following Yu et al. [2012], we extract dense SIFT local features [Vedaldi and Fulkerson 2008] and quantize them to form 1,500-D spatial pyramid bag-of-words features [Lazebnik et al. 2006; Chatfield et al. 2011] for each image. The spatial regions are obtained by dividing the image in 1×1 and 2×2 grids for a total of 5 regions. We select 50 most-frequent concepts that are associated with thousands of images, of which 25 are randomly used as database labels while the rest as the unseen labels. Again we compare our method with the five baselines: SPLH, SH, LSH, BRE and ITQ. Similar to NUS-WIDE experiments, for all methods 5,000 images are sampled as a training set and another 1,000 images as testing queries. For SPLH, the regularization parameter η is fine tuned to a proper value 1×10^{-3} .

We evaluate the mixed image-keyword retrieval paradigm again. Similar to experiments on NUS-WIDE, our testing set includes two kinds of queries: query images associated with one and two sampled labels. The groundtruth for each query is required to be semantically and visually similar to the query. The difference from previous experiments is that the query labels come from unseen labels, rather than database labels. For our method, we use the retrieval mechanism for unseen labels proposed in Section 3.4, namely reconstruction coefficients for each unseen label will be learned first, and then the most relevant database label will be used as a proxy to select the active hashing functions.

In Figure 6 we show comprehensive results for the two kinds of queries: the recall performances under both 24 and 48 hash bits per query, and MAP scores under varying hash bits. Here we totally learn 300 hash bits as the hash pool for query-adaptive hash bit selection. It can be observed from the figure that our method achieves much better performances than other methods using the same number of hash bits. As unsupervised methods, LSH and SH give the lowest performances, but obtain slight performance improvements with an increasing number of hash bits. SPLH, utilizing the supervised information, gives competing performance at first, but then drops below other baselines. One reason is that SPLH learns the hashing functions only for database labels, and thus loses the discriminative power for unseen labels. For one-label query, ITQ archives a very satisfying performance. However, when using more labels in the query, it fails to compete with our BSH. Compared with all baselines, our method can find good proxy database labels for query labels, and thus is generalized well for unseen labels. It can be noticed in Figure 6 that the proposed hashing method achieves significantly

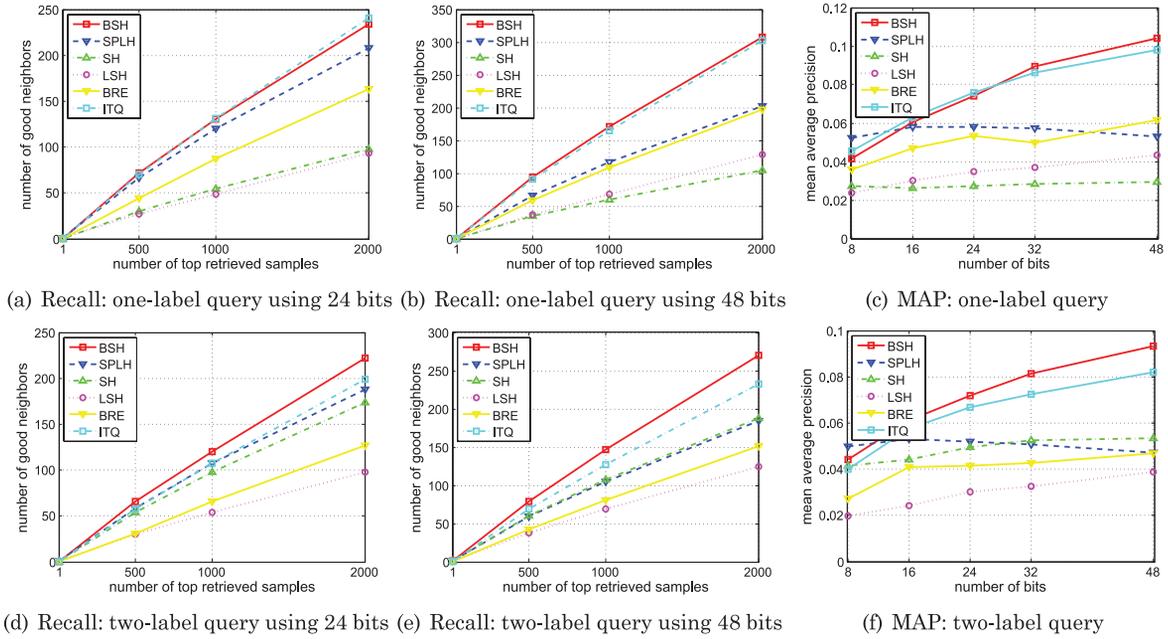


Fig. 6. Performances of BSH and baselines on a-TRECVID image benchmark with one- and two-label queries.

superior and dramatically increasing performances when using long hash codes, owing to both the label-targeting hashing functions and the adaptive retrieval mechanism.

5.3 Shared Methods

This article has proposed a greedy strategy for active label selection (i.e., forward selection, denoted by “greedy sharing” in Figure 7(a)), greatly accelerating the exhaustive search which has an exponential complexity $O(2^L)$ (L is the label number). Other native strategies like “all sharing” (i.e., all labels are selected for each bit), and “random sharing” (i.e., a number of labels are randomly selected for each bit) are contrasted with “greedy sharing” as the baselines. The “all sharing” strategy has been used by other conventional hashing schemes [Wang et al. 2010a; Weiss et al. 2008] but differs in the way to learn the hashing functions. The “random sharing” strategy uniformly samples specific number (we use 3, the rounded averaged number of active labels using “greedy sharing”) of labels to be active.

We perform experimental comparison on CIFAR-10 among different strategies for active label set discovery as illustrated in Figure 7. The comparison is based on the recall performance in terms of the number of good neighbors in top-300 retrieved samples for all strategies. To make it comprehensive, the performances with different hash bits per query (specifically, 8–32 bits) are reported. From Figure 7(a), it is observed that both “random sharing” and “greedy sharing” outperform “all sharing”, which indicates that the idea of sparsifying bit-label association is helpful. Moreover, our proposed “greedy sharing”, greedily exploiting the near-optimal label subset for each hashing function, is shown to be the best. An example of the learned bit-label association matrix using “greedy sharing” is plotted in Figure 7(b). The white blob indicates that the specific bit and label are associated (see detailed discussion in Section 5.4.1). Besides “all sharing” and “random sharing,” we also experimented with other popular alternative like the backward elimination. Compared with “greedy sharing,” the backward elimination leads to more dense associations between labels and bits (5.7 labels share a bit) and more computation cost (13.3 s for learning a hash function) at the training stage.

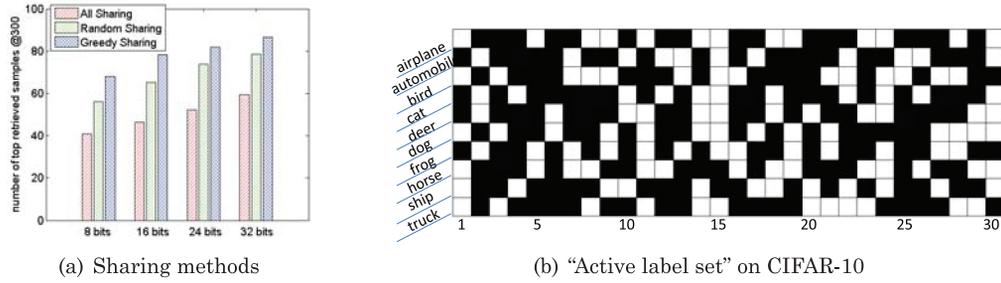


Fig. 7. Sharing hashing functions across "active label set". (a) Performance comparison of different sharing methods. (b) Illustration of "active label set" learned by the proposed method on CIFAR-10: the horizontal and vertical axes correspond to 30 hash bits and 10 semantic labels respectively, and the white cells denote the active labels.

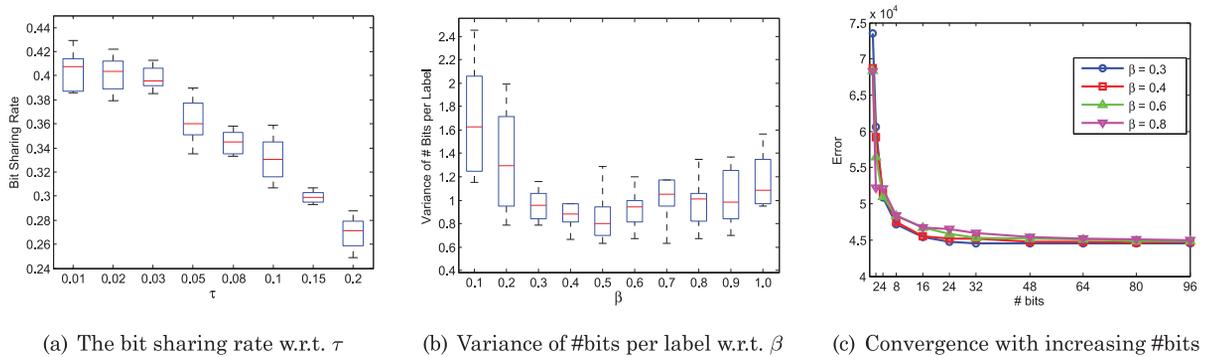


Fig. 8. Sparsity and convergence analysis of the proposed algorithm.

5.4 Sparsity and Convergence

5.4.1 Sparsity. The key idea of our method is to sequentially exploit the sparse association between hashing functions and labels. In Algorithm 1, the size of the active label set S_b is controlled by the parameter τ , which has a straightforward effect on the sparsity of the bit-label association.

To study the impact of τ , first we introduce the bit sharing rate for each bit: $r_{bs}(b) = \frac{|S_b|}{L}$, which reflects how many labels are targeted by the b -th hash bit. According to Algorithm 1, it is easy to conclude that a large τ will lead to a small S_b and thus a low bit sharing rate, because it terminates the greedy search process early. We investigate the average sharing rate of all bits as a sparsity measure of the association: a small average value indicates significant sparsity. Figure 8(a) shows the average of $r_{bs}(b)$ on CIFAR-10 dataset (10 labels and 100 bits) with respect to different τ . It is obvious that the average bit sharing rate decreases as τ increases from 1% to 20%. For our algorithm setting where $\tau = 5\%$, it is 0.3614, namely on average 3.614 labels share a bit. Therefore, if we want to use $k_b = 8$ hash bits for each label on CIFAR-10, totally $k_b \lceil \frac{L}{r_s} \rceil = 24$ hashing functions need to learn. This will significantly reduce both the storage and computation compared with state-of-the-art query-adaptive methods like LRH for which at least 80 hashing functions are required.

5.4.2 Convergence. Besides the sparsity, the convergence rate is another important property, especially for Boosting-style methods. In our algorithm, it is determined by parameter β which is simply set to a constant value in Algorithm 2 (adaptive β is discussed in Section 4.1). β has effect on the update of π in each boosting round: a large β amplifies the weights of the mis-separated neighbor pairs and thus

makes the algorithm concentrate on these pairs by selecting the same active labels in the next round. Similarly, it is highly possible that with a very small β the algorithm acts identically in the successive rounds, since the weights change quite slightly. To make the learned hashing functions cover all labels in a few rounds, and thus speedup the convergence, a proper β should be chosen.

We investigate the numbers of hashing functions associated with each label in the first few boosting rounds. Intuitively we expect a small variance among them, which means that the hashing functions can uniformly target at all labels within a small total budget. Figure 8(b) shows the variances using different β in the first 24 rounds on CIFAR-10. The trend is very consistent with our analysis, namely too small or too large β will lead to similar active label sets and thus unbalanced coverage. According to our observation, on CIFAR-10 a proper β for a balanced coverage should fall in [0.3, 0.8]. Considering the training error defined in Equation (17), Figure 8(c) demonstrates a fast convergence with β from 0.3 to 0.8. The error decreases dramatically at first and then tends to keep a steady state, where each hashing function gives a stable performance on the training data.

6. CONCLUSIONS

In this article, we have attempted to address the mixed image-keyword query adaptive hashing problem over multilabel images, and proposed a general supervised hashing framework based on boosting. Learning a hashing function for all data with different semantic similarities is very difficult, while the proposed method only focuses on the subset of active labels to overcome the difficulty. It exploits the shared subspaces among related labels and learns the shared hashing functions for these labels in the boosting procedure. An efficient query-adaptive bit selection method, integrated into this framework for both database and unseen label retrieval, enhances both the adaptivity and the scalability of the work. As discussed, such framework is considerably robust in terms of sparsity and convergence, and easily extended to nonlinear hashing. Our experimental results on several standard benchmarks demonstrate the superiority of the proposed method can achieve better performance than state-of-the-art methods when using the same budgeted number of hashing functions.

REFERENCES

- J. L. Bentley. 1975. Multidimensional binary search trees used for associative searching. *Commun. ACM* 18, 509–517.
- A. Broder, M. Charikar, A. Frieze, and M. Mitzenmacher. 1998. Min-wise independent permutations. In *Proceedings of the Symposium on Theory of Computing*.
- R. Caruana. 1997. Multitask learning. *Mach. Learn.* 28, 41–75.
- K. Chatfield, V. Lempitsky, A. Vedaldi, and A. Zisserman. 2011. The devil is in the details: An evaluation of recent feature encoding methods. In *Proceedings of the British Machine Vision Conference*.
- T. -S. Chua, J. Tang, R. Hong, H. Li, Z. Luo, and Y. Zheng. 2009. Nus-wide: A real-world web image database from National University of Singapore. In *Proceedings of the ACM Conference on Image and Video Retrieval*.
- M. Datar, N. Immorlica, P. Indyk, and V. S. Mirrokni. 2004. Locality-sensitive hashing scheme based on p-stable distributions. In *Proceedings of the Symposium on Computational Geometry*.
- J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. 2009. ImageNet: A large-scale hierarchical image database. In *Proceedings of the IEEE International Conference on Computer Vision and Pattern Recognition*.
- E. Elhamifar and R. Vidal. 2009. Sparse subspace clustering. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. IEEE, 2790–2797.
- Y. Freund and R. E. Schapire. 1995. A decision-theoretic generalization of on-line learning and an application to boosting. In *Proceedings of the 2nd European Conference on Computational Learning Theory (EuroCOLT'95)*. Springer, 23–37.
- J. Friedman, T. Hastie, and R. Tibshirani. 1998. Additive logistic regression: a statistical view of boosting. In *Annals of Statistics*.
- Y. Gong and S. Lazebnik. 2011. Iterative quantization: A Procrustean approach to learning binary codes. In *Proceedings of the IEEE International Conference on Computer Vision and Pattern Recognition*. 817–824.
- J. He, S. Kumar, and S.-F. Chang. 2012. On the difficulty of nearest neighbor search. In *Proceedings of the International Conference on Machine Learning*.

- J. He, W. Liu, and S.-F. Chang. 2010. Scalable similarity search with optimized kernel hashing. In *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*.
- P. Indyk. 2004. Nearest neighbours in high-dimensional spaces. In *Handbook of Discrete and Computational Geometry 2nd Ed.*, E. Goodman and J. O'Rourke, Eds. *Chapter 39*, CRC Press.
- P. Indyk and R. Motwani. 1998. Approximate nearest neighbors: towards removing the curse of dimensionality. In *Proceedings of the ACM Symposium on Theory of Computing*.
- S. Ji, L. Tang, S. Yu, and J. Ye. 2010. A shared-subspace learning framework for multi-label classification. *ACM Trans. Knowl. Discov. Data* 4, 8:1–8:29.
- A. Krizhevsky. 2009. Learning multiple layers of features from tiny images. Tech. rep.
- B. Kulis and T. Darrell. 2009. Learning to hash with binary reconstructive embeddings. In *Advances in Neural Information Processing Systems*.
- B. Kulis and K. Grauman. 2009. Kernelized locality-sensitive hashing for scalable image search. In *Proceedings of the IEEE International Conference on Computer Vision*.
- S. Lazebnik, C. Schmid, and J. Ponce. 2006. Beyond bags of features: Spatial pyramid matching for recognizing natural scene categories. In *Proceedings of the IEEE International Conference on Computer Vision and Pattern Recognition (CVPR'06)*. IEEE, 2169–2178.
- H. Lee, A. Battle, R. Raina, and A. Y. Ng. 2007. Efficient sparse coding algorithms. In *Advances in Neural Information Processing Systems*, B. Schölkopf, J. Platt, and T. Hoffman, Eds., 801–808.
- P. Li, M. Wang, J. Cheng, C. Xu, and H. Lu. 2013. Spectral hashing with semantically consistent graph for image indexing. *IEEE Trans. Multimedia* 15, 1, 141–152.
- W. Liu, J. Wang, R. Ji, Y.-G. Jiang, and S.-F. Chang. 2012a. Supervised hashing with kernels. In *Proceedings of the IEEE International Conference on Computer Vision and Pattern Recognition*.
- W. Liu, J. Wang, S. Kumar, and S.-F. Chang. 2011. Hashing with graphs. In *Proceedings of the International Conference on Machine Learning*.
- X. Liu, J. He, D. Liu, and B. Lang. 2012b. Compact kernel hashing with multiple features. In *Proceedings of the ACM International Conference on Multimedia*.
- X. Liu, Y. Mu, B. Lang, and S.-F. Chang. 2012c. Compact hashing for mixed image-keyword query over multi-label images. In *Proceedings of the ACM International Conference on Multimedia Retrieval*.
- Y. Mu, X. Chen, T.-S. Chua, and S. Yan. 2011. Learning reconfigurable hashing for diverse semantics. In *Proceedings of the ACM International Conference on Multimedia Retrieval*.
- Y. Mu, X. Chen, X. Liu, T.-S. Chua, and S. Yan. 2012. Multimedia semantics-aware query-adaptive hashing with bits reconfigurability. *Int. J. Multimedia Information Retrieval*, 1–12.
- Y. Mu, J. Shen, and S. Yan. 2010. Weakly-supervised hashing in kernel space. In *Proceedings of the IEEE International Conference on Computer Vision and Pattern Recognition*.
- A. Oliva and A. Torralba. 2001. Modeling the shape of the scene: A holistic representation of the spatial envelope. *Int. J. Comput. Vision* 42, 145–175.
- Y. Pati, R. Rezaifar, and P. S. Krishnaprasad. 1993. Orthogonal matching pursuit: recursive function approximation with applications to wavelet decomposition. In *Conference Record of the 27th Asilomar Conference on Signals, Systems and Computers*. Vol. 1, 40–44.
- J. Song, Y. Yang, Z. Huang, H. T. Shen, and R. Hong. 2011. Multiple feature hashing for real-time large scale near-duplicate video retrieval. In *Proceedings of the ACM International Conference on Multimedia*.
- A. Torralba, K. Murphy, and W. Freeman. 2004. Sharing features: efficient boosting procedures for multiclass object detection. In *Proceedings of the IEEE International Conference on Computer Vision and Pattern Recognition*.
- A. Vedaldi and B. Fulkerson. 2008. VLFeat: An open and portable library of computer vision algorithms. <http://www.vlfeat.org/>.
- J. Wang, S. Kumar, and S.-F. Chang. 2010a. Semi-supervised hashing for scalable image retrieval. In *Proceedings of the IEEE International Conference on Computer Vision and Pattern Recognition*.
- J. Wang, S. Kumar, and S.-F. Chang. 2010b. Sequential projection learning for hashing with compact codes. In *Proceedings of the International Conference on Machine Learning*.
- J. Wang, S. Kumar, and S.-F. Chang. 2012. Semi-supervised hashing for large scale search. *IEEE Trans. Pattern Anal. Mach. Intell.*
- K. Q. Weinberger and L. K. Saul. 2009. Distance metric learning for large margin nearest neighbor classification. *J. Mach. Learn. Res.* 10, 207–244.
- Y. Weiss, A. Torralba, and R. Fergus. 2008. Spectral hashing. In *Advances in Neural Information Processing Systems*.
- ACM Transactions on Multimedia Computing, Communications and Applications, Vol. 10, No. 2, Article 22, Publication date: February 2014.

- R. Yan, J. Tesic, and J. R. Smith. 2007a. Model-shared subspace boosting for multi-label classification. In *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*.
- S. Yan, D. Xu, B. Zhang, H. Zhang, Q. Yang, and S. Lin. 2007b. Graph embedding and extensions: A general framework for dimensionality reduction. *IEEE Trans. Pattern Anal. Mach. Intell.* 29, 1, 40–51.
- F. Yu, R. Ji, M.-H. Tsai, G. Ye, and S.-F. Chang. 2012. Weak attributes for large-scale image retrieval. In *Proceedings of the IEEE International Conference on Computer Vision and Pattern Recognition*.
- A. L. Yuille. 2002. The concave-convex procedure (cccp). In *Advances in Neural Information Processing Systems*. MIT Press.
- D. Zhang, F. Wang, and L. Si. 2011. Composite hashing with multiple information sources. In *Proceedings of the ACM SIGIR Conference on Research and Development in Information Retrieval*.

Received January 2013; revised May 2013; accepted September 2013