This ICCV paper is the Open Access version, provided by the Computer Vision Foundation.
Except for this watermark, it is identical to the version available on IEEE Xplore.

Top Rank Supervised Binary Coding for Visual Search

Dongjin Song*
Department of ECE
UC San Diego
dosong@ucsd.edu

Wei Liu*
School of Electronic Engineering
Xidian University
wliu@ee.columbia.edu

Rongrong Ji
School of Information Science
and Engineering
Xiamen University
rrji@xmu.edu.cn

David A. Meyer
Department of Mathematics
UC San Diego
dmeyer@math.ucsd.edu

John R. Smith
IBM T. J. Watson
Research Center
jsmith@us.ibm.com

Abstract

In recent years, binary coding techniques are becoming increasingly popular because of their high efficiency in handling large-scale computer vision applications. It has been demonstrated that supervised binary coding techniques that leverage supervised information can significantly enhance the coding quality, and hence greatly benefit visual search tasks. Typically, a modern binary coding method seeks to learn a group of coding functions which compress data samples into binary codes. However, few methods pursued the coding functions such that the precision at the top of a ranking list according to Hamming distances of the generated binary codes is optimized. In this paper, we propose a novel supervised binary coding approach, namely Top Rank Supervised Binary Coding (Top-RSBC), which explicitly focuses on optimizing the precision of top positions in a Hamming-distance ranking list towards preserving the supervision information. The core idea is to train the disciplined coding functions, by which the mistakes at the top of a Hamming-distance ranking list are penalized more than those at the bottom. To solve such coding functions, we relax the original discrete optimization objective with a continuous surrogate, and derive a stochastic gradient descent to optimize the surrogate objective. To further reduce the training time cost, we also design an online learning algorithm to optimize the surrogate objective more efficiently. Empirical studies based upon three benchmark image datasets demonstrate that the proposed binary coding approach achieves superior image search accuracy over the state-of-the-arts.

* Wei Liu was with IBM Research when this work was conducted.

1. Introduction

With the rapid development of massive image collections such as Facebook, Instagram, and Flickr, how to search for visually relevant images effectively and efficiently has become overwhelmingly important. Instead of exhaustively searching for the most similar images with respect to a query in one high-dimensional feature space, binary coding techniques encode images with binary codes via proper coding functions and perform efficient searches in the generated low-dimensional code space (*i.e.*, Hamming space), therefore drastically accelerating search procedures and also saving storage. During the past few years, binary coding (also known as *hashing* [26] in the literature) techniques have been widely used in a variety of computer vision and machine learning applications, including object recognition [23, 24, 21], scene classification [22], image retrieval [9, 14, 8, 33, 34, 20], linear classifier training [10, 11], active learning [15], multi-task learning [28], *etc*.

Conceptually, a typical binary coding method exploits a group of coding functions $\left\{h_k : \mathbb{R}^d \mapsto \mathbb{H}\right\}_{k=1}^{r}$ to map data samples from a d-dimensional data space $\mathbb{R}^d$ to an r-dimensional Hamming space $\mathbb{H}^r$, such that original samples are represented as binary codes of length r. Early explorations in binary coding, such as Locality-Sensitive Hashing (LSH) [2] and Min-wise Hashing (MinHash) [3], construct coding functions with random permutations or projections. These randomized binary coding methods, however, require long code lengths ($r \geq 1,000$) to meet search requirements, and usually incur inferior search accuracy for large-scale image search [13].

Unlike the early randomized binary coding methods that are independent of data, a great number of data-dependent binary coding techniques have been proposed during the

recent decade. Generally speaking, these techniques can be categorized into two main types: unsupervised and supervised (including semi-supervised) approaches. Unsupervised approaches, such as Spectral Hashing (SH) [29], Iterative Quantization (ITQ) [4], Isotropic Hashing (ISOH) [6], Discrete Graph Hashing (DGH) [12] *etc.*, seek to learn coding functions by taking into account underlying data structures, distributions, or topological information. Differently, supervised approaches aim to learn coding functions by incorporating supervised information, *e.g.*, instance-level labels, pair-level labels, or triplet-level ranks. The representative techniques include Binary Reconstructive Embedding (BRE) [7], Minimal Loss Hashing (MLH) [16], Kernel-based Supervised Hashing (KSH) [13], Hamming Distance Metric Learning (HDML) [17], Ranking-based Supervised Hashing (RSH) [27], Column Generation Hashing (CGH) [11], and Rank Preserving Hashing (RPH) [20].

In particular, pointwise supervised methods (*e.g.*, BRE [7]) and pairwise supervised methods (*e.g.*, MLH [16] and KSH [13]) respectively leverage labels of instances and pairwise labels of instance-pairs to train the coding functions such that the label information can be preserved in the produced Hamming space. Their objectives, however, may be suboptimal for visual search task because the ranking information was not fully exploited. To tackle this issue, rank supervised approaches (*e.g.*, HDML [17], RSH [27], CGH [11], and RPH [20]) which employ triplet-level ranks are developed.

Although the aforementioned rank supervised binary coding approaches have shown their efficacy for large-scale image search tasks, we would argue that few of them explored optimizing the precision at the top positions of a ranking list according to Hamming distances of the generated binary codes since most of them treat triplet-level violations equally (independent of their positions in the Hamming distance ranking list), which is indeed one of the most practical concerns with high-performance image search. To this end, in this paper we propose a novel Top Rank Supervised Binary Coding (Top-RSBC) technique specially designed for optimizing the precision of top positions in a Hamming distance ranking list. The core idea is to train the disciplined coding functions by which the mistakes at the top of a Hamming-distance ranking list are penalized more than those at the bottom. Since our introduced rank-preserving objective is discrete in nature and its associated optimization problem is combinatorially difficult, we relax the original discrete objective to a continuous and differentiable surrogate, and accordingly derive a stochastic gradient descent method to optimize the surrogate objective. To further reduce the training time cost, we also design an online learning algorithm to optimize the surrogate objective efficiently. We compare the proposed approach, *i.e.*, Top-RSBC, against various state-of-the-art binary coding methods through extensive experiments conducted on three benchmark image datasets, *i.e.*, SUN397 [32], ImageNet100 [19], and YouTube Faces [31], among which the largest dataset contains more than 600K images. The experimental results demonstrate that Top-RSBC outperforms the state-of-the-arts in executing rapid image search with binary codes.

The rest of this paper is organized as follows. In Section 2, we describe the learning model of Top-RSBC. In Section 3, we study a surrogate optimization objective of Top-RSBC, accordingly introduce a stochastic gradient descent method to optimize this objective, and further design an online learning algorithm to tackle this objective efficiently. We present the experimental results in Section 4, and finally conclude this work in Section 5.

2. Top Rank Supervised Binary Coding

In this section, we first introduce some main notations used in the paper. Let $\mathbf{X} \in \mathbb{R}^{d\times n}$ be a data matrix of n data samples with d dimensions, $\boldsymbol{x}_i \in \mathbb{R}^d$ be the i-th column of $\mathbf{X}$, and X_{ij} be the entry in i-th row and j-th column of $\mathbf{X}$, respectively. Moreover, we use $\|\cdot\|_{\mathrm{F}}$ to denote the Frobenius norm of matrices, and $\|\boldsymbol{x}\|_{\mathrm{H}}$ to represent the Hamming norm of a vector $\boldsymbol{x}$, which is defined as the number of nonzero entries in $\boldsymbol{x}$, *i.e.*, ℓ_0 norm. We use $\|\boldsymbol{x}\|_1$ to represent the ℓ_1 norm of vector $\boldsymbol{x}$, which is defined as the sum of absolute values of the entries in $\boldsymbol{x}$.

Given a data matrix $\mathbf{X} \in \mathbb{R}^{d\times n}$, the purpose of our proposed supervised binary coding model Top-RSBC is to learn a group of mapping functions $\{h_c(\boldsymbol{x})\}_{c=1}^r$ such that a d-dimensional floating-point input $\boldsymbol{x} \in \mathbb{R}^d$ is compressed into an r-bit binary code $\boldsymbol{b} = \big[h_1(\boldsymbol{x}), \cdots, h_r(\boldsymbol{x})\big]^\top \in \mathbb{H}^r \equiv \{1, -1\}^r$. This mapping, known as coding or hash function in the literature, is formulated by

$$h_c(\boldsymbol{x}) = \mathrm{sgn}\big(f_c(\boldsymbol{x})\big), \quad c = 1, \cdots, r, \tag{1}$$

where $\mathrm{sgn}(x)$ is the sign function that returns 1 if input variable $x > 0$ and -1 otherwise, and $f_c : \mathbb{R}^d \mapsto \mathbb{R}$ is a proper prediction function. A variety of mathematical forms for f_c (*e.g.*, linear or nonlinear) can be utilized to serve to specific data domains and practical applications. In this work, we focus on using a linear prediction function, that is, $f_c(\boldsymbol{x}) = \boldsymbol{w}_c^\top \boldsymbol{x} + t_c$ (where $\boldsymbol{w}_c \in \mathbb{R}^d$ and $t_c \in \mathbb{R}$) for simplicity. Following the previous work [4, 6, 13, 27], we set the bias term $t_c = -\boldsymbol{w}_c^\top \boldsymbol{u}$ by using the mean vector $\boldsymbol{u} = \sum_{i=1}^n \boldsymbol{x}_i/n$, which will make each generated binary bit $\{h_c(\boldsymbol{x}_i)\}_{c=1}^r$ for $c \in [1 : r]$ be nearly balanced and hence exhibit maximum entropy. For brevity, we further define the whole coding function $\boldsymbol{h} : \mathbb{R}^d \mapsto \mathbb{H}^r$ to comprise the functionality of r individual coding functions $\{h_c\}_{c=1}^r$, that is,

$$\boldsymbol{h}(\boldsymbol{x}, \mathbf{W}) = \mathrm{sgn}\big(\mathbf{W}^\top(\boldsymbol{x} - \boldsymbol{u})\big), \tag{2}$$

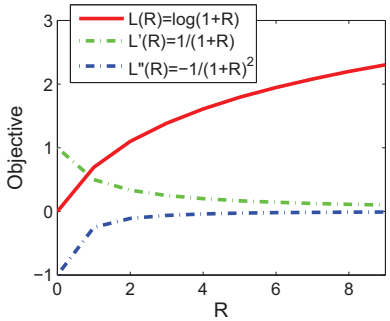
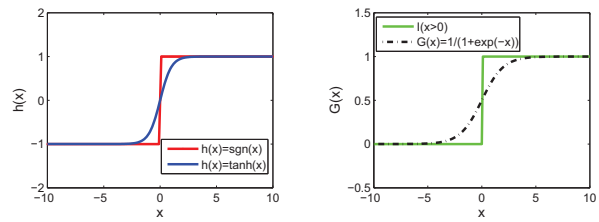


Figure 1. Loss function $L(\mathrm{R})$, the gradient of $L(\mathrm{R})$, *i.e.*, $L'(\mathrm{R})$, and $L''(\mathrm{R})$.



(a) (b)

Figure 2. Relaxation of the objective function. (a) $\tanh(x)$ is a relaxation of $\mathrm{sgn}(x)$; (b) sigmoid loss $G(x) = \frac{1}{1+exp(-x)}$ is a good approximation for indicator function $I(x > 0)$.

which is parameterized by a matrix $\mathbf{W} = [\boldsymbol{w}_1, \cdots, \boldsymbol{w}_r] \in \mathbb{R}^{d\times r}$. Note that Eq. (2) applies the sign function $\mathrm{sgn}(\cdot)$ in the element-wise way. For simplicity, we will abbreviate $\mathbf{W}$ and use $\boldsymbol{h}(\boldsymbol{x}) = \boldsymbol{h}(\boldsymbol{x}, \mathbf{W})$ in the following description.

To pursue such a coding function $\boldsymbol{h}(\cdot)$, rather than considering pairwise data similarities (*i.e.*, pair-level labels) as in [16, 13], we leverage relative data similarities in the form of triplets $\mathcal{D} = \big\{(\boldsymbol{x}_i, \boldsymbol{x}_j, \boldsymbol{x}_s)\big\}$, in which the pair $(\boldsymbol{x}_i, \boldsymbol{x}_j)$ is more similar than the pair $(\boldsymbol{x}_i, \boldsymbol{x}_s)$ (*e.g.*, $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$ belong to the same class while $\boldsymbol{x}_i$ and $\boldsymbol{x}_s$ belong to different classes). Intuitively, we would expect that these relative similar relationships revealed by $\mathcal{D}$ can be preserved within the Hamming space by virtue of a good coding function $\boldsymbol{h}(\cdot)$, which makes the Hamming distance between the codes $\boldsymbol{h}(\boldsymbol{x}_i)$ and $\boldsymbol{h}(\boldsymbol{x}_j)$ smaller than that between the codes $\boldsymbol{h}(\boldsymbol{x}_i)$ and $\boldsymbol{h}(\boldsymbol{x}_s)$. Suppose that $\boldsymbol{x}_i$ is a query, $\boldsymbol{x}_j$ is its similar sample, and $\boldsymbol{x}_s$ is its dissimilar sample. Then the "rank" of $\boldsymbol{x}_j$ with respect to the query $\boldsymbol{x}_i$ can be defined as the number of dissimilar samples $\boldsymbol{x}_s$ (when s varies) which are more closer to the query $\boldsymbol{x}_i$ than $\boldsymbol{x}_j$ within Hamming space. Specifically, the "rank" can be written as:

$$\mathrm{R}(\boldsymbol{x}_i, \boldsymbol{x}_j) = \sum_s I\Big(\big\|\boldsymbol{h}(\boldsymbol{x}_i) - \boldsymbol{h}(\boldsymbol{x}_s)\big\|_{\mathrm{H}} \leq \big\|\boldsymbol{h}(\boldsymbol{x}_i) - \boldsymbol{h}(\boldsymbol{x}_j)\big\|_{\mathrm{H}}\Big), \tag{3}$$

where $I(\cdot)$ is an indicator function which returns 1 if the condition in the parenthesis is satisfied and returns 0 otherwise. Intuitively, the function $\mathrm{R}(\boldsymbol{x}_i, \boldsymbol{x}_j)$ explicitly measures the number of the incorrectly ranked dissimilar samples $\boldsymbol{x}_s$'s which are closer to the query $\boldsymbol{x}_i$ than the similar sample $\boldsymbol{x}_j$ in terms of Hamming distance and therefore indicates the position of $\boldsymbol{x}_j$ in a Hamming distance ranking list with respect to the query $\boldsymbol{x}_i$. In order to explicitly optimize the search precision at top positions of a ranking list, we introduce a ranking loss as:

$$L\big(\mathrm{R}(\boldsymbol{x}_i, \boldsymbol{x}_j)\big) = \log\big(1 + \mathrm{R}(\boldsymbol{x}_i, \boldsymbol{x}_j)\big) \tag{4}$$

which penalizes the samples (*i.e.*, $\boldsymbol{x}_s$'s) that are incorrectly ranked at the top of a Hamming-distance ranking list more than those at the bottom. This is because the increment of $L(\mathrm{R})$ gradually decays as R increases linearly. The detailed properties of the ranking loss are shown in Figure 1. As we can notice, $L(R)$ is a one to one monotonic increasing function with first order derivative $L'(R)$ large than zero and second order derivative $L''(R)$ smaller than zero. Since $L(R)$ can be seen as an integral of its gradient, intuitively, $L'(R) > 0$ preserves the rank by penalizing the "rank" (*i.e.*, R) we defined more severe at the top (*i.e.*, when R is small) than at the bottom (*i.e.*, when R is large).

Our model Top-RSBC makes use of the above ranking loss and the learning objective is formulated as follows:

$$\mathcal{O}(\mathbf{X}, \mathbf{W}) = \sum_i \sum_j \log\big(1 + \mathrm{R}(\boldsymbol{x}_i, \boldsymbol{x}_j)\big) + \frac{\lambda}{2}\|\mathbf{W}\|_{\mathrm{F}}^2, \tag{5}$$

where the first term is the proposed ranking loss, the second term enforces regularization, and $\lambda > 0$ is a positive parameter controlling the trade-off between the ranking loss and the regularization term. By optimizing this objective with respect to $\mathbf{W}$, we expect to optimize the visual search precision at top positions of a Hamming distance ranking list.

We are aware that similar idea of our ranking loss in Eq. (4) was previously used for information retrieval [25] and image annotation [30]. Our work differs from the them because (a) the relative similarity of the triplets are measured with Hamming distance which is discrete and discontinuous; and (b) our ranking loss is continuous differentiable while the previous ones are discrete and difficult to optimize.

3. Optimization

Although the ranking loss in Eq. (4) is continuous and differentiable, our objective in Eq. (5) is still difficult to optimize. This is because: (a) the coding function is a discrete mapping; and (b) the Hamming norm lies in a discrete space. Therefore, our introduced Top-RSBC objective is discrete in nature and the associated optimization problem is combinatorially difficult.

Algorithm 1 SGD learning for Top-RSBC.

Input: data $\mathcal{D} = \{(\boldsymbol{x}_i, \boldsymbol{x}_j, \boldsymbol{x}_s)\}$, α, $\mathbf{W}$, and λ
Output: $\mathbf{W} \in \mathbb{R}^{d\times r}$
repeat
  Randomly pick up an query $\boldsymbol{x}_i$.
  Fix $\boldsymbol{x}_i$ and randomly select an similar example $\boldsymbol{x}_j$.
  Fix $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$, and randomly draw $p \leq |\mathcal{N}|$ dissimilar $\boldsymbol{x}_s$'s when s varies to form $\{\boldsymbol{x}_i, \boldsymbol{x}_j, \boldsymbol{x}_s\}_{s=1}^{p}$.
  Calculate the gradient in Eq. (13).
  Make a gradient descent based upon Eq. (14).
until validation error does not improve or maximum iteration number is achieved.

To tackle this issue, we need relax the original discrete objective to a continuous and differentiable surrogate.

3.1. Relaxation

Specifically, the target coding function $\boldsymbol{h}(\boldsymbol{x}) = \mathrm{sgn}\big(\mathbf{W}^\top(\boldsymbol{x}-\boldsymbol{u})\big)$ can be relaxed as:

$$\overline{\boldsymbol{h}}(\boldsymbol{x}) = \tanh\big(\mathbf{W}^\top(\boldsymbol{x}-\boldsymbol{u})\big), \tag{6}$$

which is continuous and differentiable as shown in Figure 2(a). $\tanh(\cdot)$ is a good approximation for $\mathrm{sgn}(\cdot)$ function because it transforms the value in the parenthesis to be in between of -1 and $+1$.

Next, the Hamming norm in Eq. (3) is relaxed to ℓ_1 norm which is convex.

Finally, we relax the indicator function in Eq. (3) with sigmoid loss (as shown in Figure 2(b)). Accordingly, Eq. (3) can be can be approximated with::

$$\begin{aligned} &I\Big(\big\|\overline{\boldsymbol{h}}(\boldsymbol{x}_i)-\overline{\boldsymbol{h}}(\boldsymbol{x}_s)\big\|_1 \leq \big\|\overline{\boldsymbol{h}}(\boldsymbol{x}_i)-\overline{\boldsymbol{h}}(\boldsymbol{x}_j)\big\|_1\Big) \\ \approx &G\Big(\big\|\overline{\boldsymbol{h}}(\boldsymbol{x}_i)-\overline{\boldsymbol{h}}(\boldsymbol{x}_j)\big\|_1 - \big\|\overline{\boldsymbol{h}}(\boldsymbol{x}_i)-\overline{\boldsymbol{h}}(\boldsymbol{x}_s)\big\|_1\Big) \end{aligned} \tag{7}$$

where $G(z) = \frac{1}{1+\exp(-z)}$ is the sigmoid function as shown in Figure 2(b).

Based upon these relaxations, the objective in Eq. (5) can be approximated with:

$$\overline{\mathcal{O}(\mathbf{X},\mathbf{W})} = \lambda\|\mathbf{W}\|_{\mathrm{F}}^2 + \sum_i\sum_j \log(1+\overline{\mathrm{R}}(\boldsymbol{x}_i,\boldsymbol{x}_j)). \tag{8}$$

where $\overline{\mathrm{R}}(\boldsymbol{x}_i,\boldsymbol{x}_j)$ is a soft-approximated rank of $\boldsymbol{x}_j$ with respect to the query $\boldsymbol{x}_i$ which can be given by:

$$\overline{\mathrm{R}}(\boldsymbol{x}_i,\boldsymbol{x}_j) = \sum_s G\Big(\mathrm{T}_{ij}-\mathrm{T}_{is}\Big), \tag{9}$$

where T_{ij} is written as

$$\mathrm{T}_{ij} = \big\|\overline{\boldsymbol{h}}(\boldsymbol{x}_i)-\overline{\boldsymbol{h}}(\boldsymbol{x}_j)\big\|_1, \tag{10}$$

and T_{is} is denoted as

$$\mathrm{T}_{is} = \big\|\overline{\boldsymbol{h}}(\boldsymbol{x}_i)-\overline{\boldsymbol{h}}(\boldsymbol{x}_s)\big\|_1. \tag{11}$$

Although sub-gradient descent approach can be derived to optimize Eq. (8), it may converge slowly or even will be infeasible because of the expensive computational for the full gradient at each iteration. Therefore, a stochastic gradient descent method is derived to resolve this issue.

3.2. Stochastic Gradient Descent (SGD)

To optimize Top-RSBC with stochastic gradient descent algorithm, given a collection of triplets $\mathcal{D}$, we first randomly select a query $\boldsymbol{x}_i$ and its similar example $\boldsymbol{x}_j$. Then we fix $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$, and randomly draw p ($p \leq |\mathcal{N}|$) different $\boldsymbol{x}_s$'s when s varies to form a set of triplets $\{\boldsymbol{x}_i, \boldsymbol{x}_j, \boldsymbol{x}_s\}_{s=1}^{p}$. Note that $|\mathcal{N}|$ is the total number of possible choices of s. Assuming that the violated examples are uniformly distributed, then $\overline{\mathrm{R}}(\boldsymbol{x}_i,\boldsymbol{x}_j)$ can be approximated with $\lfloor\frac{|\mathcal{N}|}{p}\rfloor\cdot\sum_{s=1}^{p} G\Big(\mathrm{T}_{ij}-\mathrm{T}_{is}\Big)$ where $\lfloor\cdot\rfloor$ is the floor function.

In this way, the objective of these chosen triplets can be written as:

$$\begin{aligned} &\overline{\mathcal{O}(\boldsymbol{x}_i,\boldsymbol{x}_j,\mathbf{W})} \\ =&\frac{\lambda}{2}\|\mathbf{W}\|_{\mathrm{F}}^2 + \log\Big(1+\Big\lfloor\frac{|\mathcal{N}|}{p}\Big\rfloor\cdot\sum_{s=1}^{p} G\Big(\mathrm{T}_{ij}-\mathrm{T}_{is}\Big)\Big), \end{aligned} \tag{12}$$

and its associated gradient is given by:

$$\begin{aligned} &\frac{\partial\overline{\mathcal{O}(\boldsymbol{x}_i,\boldsymbol{x}_j,\mathbf{W})}}{\partial\mathbf{W}} = \\ &\lambda\mathbf{W} + \frac{\Big\lfloor\frac{|\mathcal{N}|}{p}\Big\rfloor\sum_{s=1}^{p} G\Big(\mathrm{T}_{ij}-\mathrm{T}_{is}\Big)G\Big(-\mathrm{T}_{ij}+\mathrm{T}_{is}\Big)}{1+\Big\lfloor\frac{|\mathcal{N}|}{p}\Big\rfloor\cdot\sum_{s=1}^{p} G\Big(\mathrm{T}_{ij}-\mathrm{T}_{is}\Big)}\cdot \\ &\Big\{(\boldsymbol{x}_i-\boldsymbol{u})\Big[\mathrm{sgn}\Big(\overline{\boldsymbol{h}}(\boldsymbol{x}_i)-\overline{\boldsymbol{h}}(\boldsymbol{x}_j)\Big)\odot\Big(1-\overline{\boldsymbol{h}}^2(\boldsymbol{x}_i)\Big)\Big]^\top \\ &-(\boldsymbol{x}_j-\boldsymbol{u})\Big[\mathrm{sgn}\Big(\overline{\boldsymbol{h}}(\boldsymbol{x}_i)-\overline{\boldsymbol{h}}(\boldsymbol{x}_j)\Big)\odot\Big(1-\overline{\boldsymbol{h}}^2(\boldsymbol{x}_j)\Big)\Big]^\top \\ &-(\boldsymbol{x}_i-\boldsymbol{u})\Big[\mathrm{sgn}\Big(\overline{\boldsymbol{h}}(\boldsymbol{x}_i)-\overline{\boldsymbol{h}}(\boldsymbol{x}_s)\Big)\odot\Big(1-\overline{\boldsymbol{h}}^2(\boldsymbol{x}_i)\Big)\Big]^\top \\ &+(\boldsymbol{x}_s-\boldsymbol{u})\Big[\mathrm{sgn}\Big(\overline{\boldsymbol{h}}(\boldsymbol{x}_i)-\overline{\boldsymbol{h}}(\boldsymbol{x}_s)\Big)\odot\Big(1-\overline{\boldsymbol{h}}^2(\boldsymbol{x}_s)\Big)\Big]^\top\Big\}, \end{aligned} \tag{13}$$

where the symbol $\odot$ denotes Hadamard product (*i.e.,* elementwise product).

Based upon this gradient, we can perform the following stochastic gradient update rule over the parameter $\mathbf{W}$, which is given as:

$$\mathbf{W}_{t+1} = \mathbf{W}_t - \alpha\frac{\partial\overline{\mathcal{O}(\boldsymbol{x}_i,\boldsymbol{x}_j,\mathbf{W})}}{\partial\mathbf{W}_t}, \tag{14}$$

where α is the learning rate and t is the iteration index.

Although the ideal case is that within these p sampled triplets $\{\boldsymbol{x}_i, \boldsymbol{x}_j, \boldsymbol{x}_s\}_{s=1}^{p}$ at least one violation (*i.e.*, the ℓ_1

Algorithm 2 Online learning for Top-RSBC.

Input: data $\mathcal{D} = \{(\boldsymbol{x}_i, \boldsymbol{x}_j, \boldsymbol{x}_s)\}$, α, $\mathbf{W}$, and λ
Output: $\mathbf{W} \in \mathbb{R}^{d \times r}$
repeat
  Randomly draw an query $\boldsymbol{x}_i$.
  Fix $\boldsymbol{x}_i$ and randomly draw an similar example $\boldsymbol{x}_j$.
  Set q=0
  repeat
    Fix both $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$, and pick up a random triplets $(\boldsymbol{x}_i, \boldsymbol{x}_j, \boldsymbol{x}_s)$ when s varies.
    $q = q + 1$
  until $\left\|\overline{\boldsymbol{h}}(\boldsymbol{x}_i) - \overline{\boldsymbol{h}}(\boldsymbol{x}_j)\right\|_1 \geq \left\|\overline{\boldsymbol{h}}(\boldsymbol{x}_i) - \overline{\boldsymbol{h}}(\boldsymbol{x}_s)\right\|_1$ or $q \geq |\mathcal{N}|$.
  if $\left\|\overline{\boldsymbol{h}}(\boldsymbol{x}_i) - \overline{\boldsymbol{h}}(\boldsymbol{x}_j)\right\|_1 \geq \left\|\overline{\boldsymbol{h}}(\boldsymbol{x}_i) - \overline{\boldsymbol{h}}(\boldsymbol{x}_s)\right\|_1$. **then**
    Calculate the gradient of Eq. (15).
    Make a gradient descent based upon Eq. (16)
  end if
until validation error does not improve or maximum iteration number is achieved.

distance between $h(\boldsymbol{x}_i)$ and $h(\boldsymbol{x}_s)$ is smaller than that between $h(\boldsymbol{x}_i)$ and $h(\boldsymbol{x}_j)$) exists, in practical applications we find that stochastic gradient descent works well even when none violation exists in these p triplets. This is because $\overline{\mathrm{R}}(\boldsymbol{x}_i, \boldsymbol{x}_j)$ is a soft-approximated rank of $\boldsymbol{x}_j$ with respect to the query $\boldsymbol{x}_i$ and making a gradient update can still increase the margin in $\mathrm{R}(\cdot)$ which helps minimize the objective.

The detailed Stochastic Gradient Descent (SGD) learning for Top-RSBC is provided in Algorithm 1. The main computation is the gradient updating and the associated computational complexity is $O(d^2rp)$.

3.3. Online Learning

Since the time complexity of SGD is linearly proportional to p, we develop an online learning algorithm to further reduce the time cost for training. The main idea is to find one violated triplets and make gradient descent only based upon this particular violated triplets. Specifically, we first randomly draw an example $\boldsymbol{x}_i$ as the query. Then we fix $\boldsymbol{x}_i$ and randomly select a similar example $\boldsymbol{x}_j$. Finally, we fix both $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$, and then uniformly draw dissimilar example $\boldsymbol{x}_s$, $s = 1, ..., q$ ($q \leq |\mathcal{N}|$) until we find a violation triplets $\{\boldsymbol{x}_i, \boldsymbol{x}_j, \boldsymbol{x}_{s=q}\}$ which satisfies $\left\|\overline{\boldsymbol{h}}(\boldsymbol{x}_i) - \overline{\boldsymbol{h}}(\boldsymbol{x}_j)\right\|_1 \geq \left\|\overline{\boldsymbol{h}}(\boldsymbol{x}_i) - \overline{\boldsymbol{h}}(\boldsymbol{x}_s)\right\|_1$. Note that $|\mathcal{N}|$ is the total number of possible choices of s.

Therefore, the objective of the chosen triplets can be written as:

$$\begin{aligned}&\overline{\mathcal{O}(\boldsymbol{x}_i, \boldsymbol{x}_j, \boldsymbol{x}_s, \mathbf{W})}\\ =&\frac{\lambda}{2}\|\mathbf{W}\|_{\mathrm{F}}^2 + \log\Big(1 + \Big\lfloor\frac{|\mathcal{N}|}{q}\Big\rfloor \cdot G\Big(\mathrm{T}_{ij} - \mathrm{T}_{is}\Big)\Big),\end{aligned} \tag{15}$$

and the associated gradient can be calculated similar to Eq. (13) (without summation over s).

Table 1. The detailed statistics of three datasets.

Datasets	**SUN397** [32]	**ImageNet100** [19]	**YouTube Faces** [31]
♯ Queries	1,800	10,000	6,500
♯ Database samples	106,953	201,054	614,626
♯ Classes	397	100	1,595
♯ Dimensions	1,600	4,096	1,770

Based upon its gradient, the online learning update is given as:

$$\mathbf{W}_{t+1} = \mathbf{W}_t - \alpha\frac{\partial\overline{\mathcal{O}(\boldsymbol{x}_i, \boldsymbol{x}_j, \boldsymbol{x}_s, \mathbf{W})}}{\partial\mathbf{W}_t}, \tag{16}$$

where α is the learning rate and t is the iteration index.

The detailed online learning procedure for Top-RSBC is provided in Algorithm 2. The main computation is the gradient updating and the searching for violation. The associated computational complexity is $O(d^2r + drq)$ which is less than SGD when q is relatively small. After getting $\mathbf{W}$ with Algorithm 2, the compact binary codes for each sample can be generated via Eq. (2) with complexity of $O(dr)$.

In practical applications of both Algorithm 1 and Algorithm 2, $\mathbf{W}$ is randomly initialized with Gaussian distribution of mean 0, standard deviation 1, which is a common choice. The learning rate α is set as 0.1 in all our experiments and regularization hyper-parameter λ are chosen from $\{1e-6, 1e-5, 1e-4, 1e-3, 1e-2, 1e-1, 1\}$ according to a validation set. The validation error (*i.e.,* Mean Average Precision (MAP) in our work) is only evaluated every a few hundreds or thousands steps on the validation set for computational efficiency.

4. Experiments

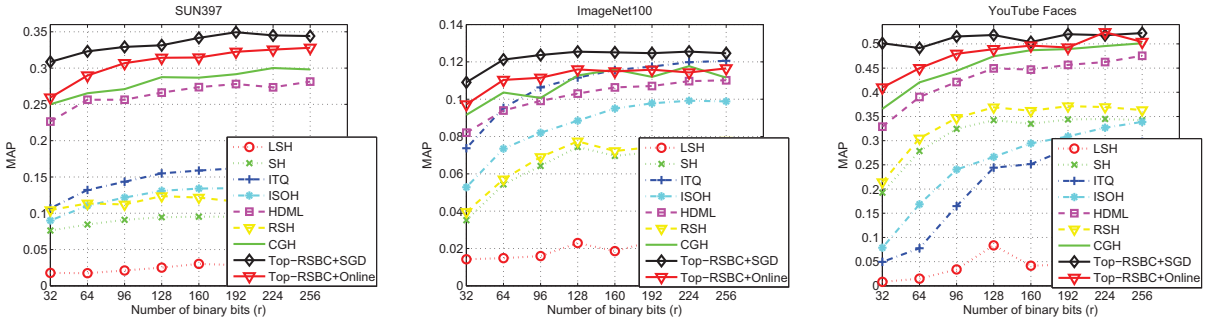
In this section, we first describe three publicly available datasets for empirical studies. Then, we introduce the competing methods as well as the evaluation metrics used in our experiments. Finally, we compare the proposed Top Rank Supervised Binary Coding (Top-RSBC) technique against several state-of-the-art binary coding and hashing algorithms to demonstrate the effectiveness in large-scale image search.

4.1. Datasets and Setup

In the experiments, we conduct image search over three datasets, *i.e.*, SUN397 [32], ImageNet100 [19], and YouTube Faces [31]. SUN397 consists of around 108K images from 397 scene categories. In SUN397, each image is represented by a 1,600-dimensional feature vector extracted by principle component analysis (PCA) from 12,288-dimensional Deep Convolutional Activation Features [5]. ImageNet100 consists of the 100 largest object categories from the famous ImageNet data challenge [19]. In ImageNet100, each object category has at least 1,600 images and each image is represented as a 4,096-dimensional Deep Convolutional Neural Network feature vector [18].

Table 2. Image search performance (**MAP** and **Precision@100**) on **SUN397** when $r = 64$, 128, and 256. All training and test times are recorded in second. The best **MAP** or **Precision@100** is displayed in bold-face type.

Algorithms	**SUN397**							
	MAP			**Precision@100**			**Training Time**	**Test Time**
♯ Bits	r =64	r =128	r =256	$r = 64$	$r = 128$	$r = 256$	$r = 256$	$r = 256$
LSH [2]	0.0174	0.0251	0.0310	0.0370	0.0715	0.1042	2.03	1.91×10^{-5}
SH [29]	0.0845	0.0945	0.0968	0.2326	0.2710	0.2947	61.57	1.42×10^{-4}
ITQ [4]	0.1321	0.1549	0.1581	0.2690	0.3125	0.3289	56.18	1.90×10^{-5}
ISOH [6]	0.1104	0.1309	0.1385	0.2454	0.2887	0.3078	11.57	2.02×10^{-5}
HDML [17]	0.2564	0.2662	0.2814	0.4200	0.4505	0.4748	5308.74	1.89×10^{-5}
RSH [27]	0.1103	0.1132	0.1376	0.2458	0.2836	0.3049	1071.83	2.22×10^{-5}
CGH [11]	0.2654	0.2875	0.2982	0.4189	0.4653	0.4868	4341.32	1.90×10^{-5}
Top-RSBC+SGD	**0.3230**	**0.3315**	**0.3441**	**0.4850**	**0.5132**	**0.5342**	4663.71	1.86×10^{-5}
Top-RSBC+Online	0.2898	0.3142	0.3280	0.4431	0.4886	0.5118	2002.46	1.91×10^{-5}



(a) MAP vs. r on SUN397 (b) MAP vs. r on ImageNet100 (c) MAP vs. r on YouTube Faces

Figure 3. MAP vs. a varying number of binary bits ($r = \{32, 64, 96, 128, 160, 192, 224, 256\}$) for nine different binary coding and hashing algorithms on SUN397, ImageNet100, and YouTube Faces.

The YouTube Faces dataset contains 614,626 face images of 1,595 different people. In YouTube Faces, each face image is represented by a 1,770-dimensional LBP feature vector [1]. The detailed statistics of these three datasets are shown in Table 1.

In SUN397, 100 images are randomly selected from each of the 18 largest scene categories to form a test set of 1,800 query images. For unsupervised binary coding and hashing algorithms, all the database samples are used for training. For supervised binary coding and hashing algorithms, we randomly select 200 images from each of the 18 scene categories to form a training set of 3,600 images; we randomly choose additional 50 images from each of the 18 scene categories to form a validation set of 900 query images. All the rest images in the 397 categories are then treated as the database samples.

Similarly, in ImageNet100, 100 images from each object category are randomly chosen to form a separate test set of 10,000 query images. All the database samples are used for training for unsupervised binary coding and hashing algorithms. To conduct supervised learning, we randomly select additional 500 images from each object category to form a training set of 50,000 images, and 50 images from each object category to form a validation set of 5,000 query images, respectively. The rest images are treated as the database samples for retrieval.

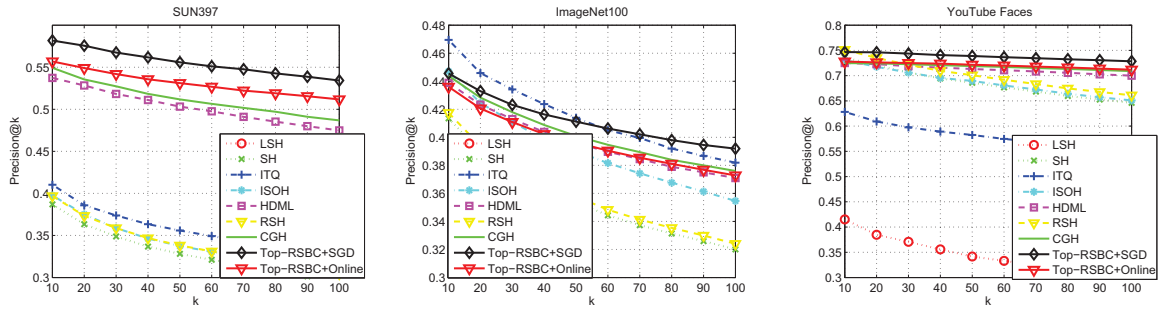
In YouTube Faces, 100 face images from each of the 65 largest face classes are randomly selected to compose a test set of 6,500 query images. To perform unsupervised learning, all the database images are used for training. For supervised binary coding and hashing algorithms, 1,000 images from each of the 65 face classes are randomly draw to form a training set of 65,000 face images. All the rest face images in the 1,770 face classes are treated as the database samples for retrieval.

4.2. Compared Algorithms and Evaluation Metrics

In the experiments, we aim to evaluate the effectiveness of the proposed Top-RSBC for visual search over the three image datasets. For this purpose, we compare Top-RSBC against seven representative binary coding and hashing algorithms. Among them, four are unsupervised algorithms, including one randomized method Locality-Sensitive Hashing (LSH) [2], one spectral method Spectral Hashing (SH) [29], and two linear projection methods Iterative Quantization (ITQ) [4] and Isotropic Hashing (ISOH) [6]. The other three are supervised algorithms which use triplets to encode the rank information (similar to the setting of our approach Top-RSBC). They are Hamming

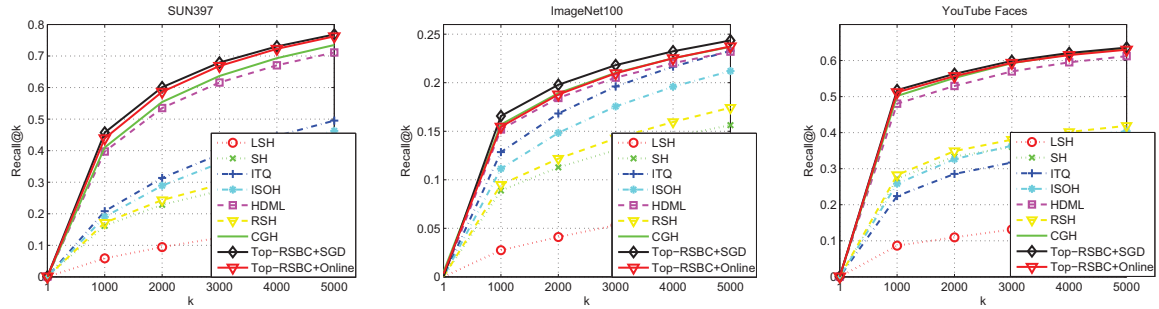
Table 3. Image search performance (**MAP** and **Precision@100**) on **ImageNet100** when $r = 64$, 128, and 256. All training and test times are recorded in second. The best **MAP** or **Precision@100** is displayed in bold-face type.

Algorithms	**ImageNet100**							
	MAP			**Precision@100**			**Training Time**	**Test Time**
♯ Bits	$r = 64$	$r = 128$	$r = 256$	$r = 64$	$r = 128$	$r = 256$	$r = 256$	$r = 256$
LSH [2]	0.0148	0.0230	0.0243	0.0375	0.0892	0.0992	12.23	5.01×10^{-5}
SH [29]	0.0543	0.0744	0.0765	0.2188	0.3146	0.3201	257.90	3.43×10^{-4}
ITQ [4]	0.0954	0.1116	0.1206	0.3010	0.3531	0.3819	171.52	4.99×10^{-5}
ISOH [6]	0.0735	0.0885	0.0988	0.2651	0.3215	0.3546	89.19	4.82×10^{-5}
HDML [17]	0.0939	0.1030	0.1101	0.2982	0.3397	0.3710	20885.24	4.92×10^{-5}
RSH [27]	0.0591	0.0792	0.0801	0.2204	0.3201	0.3241	7217.36	4.97×10^{-5}
CGH [11]	0.0997	0.1053	0.1106	0.3032	0.3447	0.3760	26320.37	4.88×10^{-5}
Top-RSBC+SGD	**0.1212**	**0.1254**	**0.1246**	**0.3332**	**0.3659**	**0.3919**	40929.83	4.89×10^{-5}
Top-RSBC+Online	0.1103	0.1160	0.1165	0.3306	0.3569	0.3712	15192.24	4.86×10^{-5}



(a) Precision@k on SUN397 (b) Precision@k on ImageNet100 (c) Precision@k on YouTube Faces

Figure 4. Precision@k with 256 binary bits on SUN397, ImageNet100, and YouTube Faces.



(a) Recall@k on SUN397 (b) Recall@k on ImageNet100 (c) Recall@k on YouTube Faces

Figure 5. Recall@k with 256 binary bits on SUN397, ImageNet100, and YouTube Faces.

Distance Metric Learning (HDML) [17], Column Generation Hashing (CGH) [11], and Ranking-based Supervised Hashing (RSH) [27].

Note that although nonlinear binary coding or hashing algorithms [8, 13] may be more effective for image search, we limit the compared algorithms to be linear approaches for fair comparison (we notice that the SH algorithm, though nonlinear, is based on and very similar to linear PCA hashing).

To measure the effectiveness of various binary coding and hashing techniques for visual search, we consider three evaluation metrics, *i.e.*, precision at top-k positions (Precision@k), recall at top-k positions (Recall@k), and Mean Average Precision (MAP).

4.3. Results

To demonstrate the effectiveness of the proposed binary coding technique in large-scale image search, we compare Top-RSBC against seven competing binary coding and hashing algorithms on the three image datasets, SUN397, ImageNet100, and YouTube Faces. For all the compared algorithms, the number of binary bits r varies from 32 to 256, as shown in Figure 3. We observe that Top-RSBC consistently achieves superior MAP than the other algorithms on the three datasets. This is because Top-RSBC is specially designed to optimize the precision at top positions of

Table 4. Image search performance (**MAP** and **Precision@100**) on **YouTube Faces** when $r = 64$, 128, and 256. All training and test times are recorded in second. The best **MAP** or **Precision@100** is displayed in bold-face type.

Algorithms	**YouTube Faces**							
	MAP			**Precision@100**			**Training Time**	**Test Time**
♯ Bits	$r = 64$	$r = 128$	$r = 256$	$r = 64$	$r = 128$	$r = 256$	$r = 256$	$r = 256$
LSH [2]	0.0148	0.0837	0.0845	0.0496	0.3097	0.3097	381.90	2.30×10^{-5}
SH [29]	0.2786	0.3424	0.3422	0.5823	0.6403	0.6461	951.67	1.61×10^{-4}
ITQ [4]	0.0777	0.2441	0.2976	0.1206	0.4227	0.5504	660.43	2.11×10^{-5}
ISOH [6]	0.1685	0.2664	0.3387	0.3496	0.4967	0.6510	240.78	2.18×10^{-5}
HDML [17]	0.3903	0.4494	0.4755	0.6066	0.6762	0.6999	5159.28	2.63×10^{-5}
RSH [27]	0.3042	0.3587	0.3604	0.5823	0.6536	0.6611	1217.33	2.37×10^{-5}
CGH [11]	0.4205	0.4743	0.5012	0.6276	0.6903	0.7090	5946.98	2.33×10^{-5}
Top-RSBC+SGD	**0.4914**	**0.5038**	**0.5224**	**0.6960**	**0.7272**	**0.7285**	6958.91	2.47×10^{-5}
Top-RSBC+Online	0.4498	0.4965	0.5140	0.6459	0.6875	0.7120	4892.36	2.29×10^{-5}

a Hamming distance ranking list by penalizing the mistakes at the top of the ranking list more than those at the bottom. The three supervised algorithms including HDML, RSH and CGH, however, treat such mistakes equally. Among the compared algorithms, we notice that supervised algorithms (*e.g.*, HDML and CGH) generally outperform unsupervised algorithms since the former leverage supervised label information to learn discriminative hash/coding functions for binary code generation. For those unsupervised algorithms, we observe that SH, ITQ and ISOH consistently and significantly outperform LSH. This implies that exploring and exploiting underlying data structures, distributions, or topological information can yield more effective codes for visual search tasks. The detailed image search performance in terms of MAP and Precision@100 over the three datasets is shown in Tables 2, 3, and 4.

We further investigate the effectiveness of the proposed Top-RSBC by comparing its Precision@k and Recall@k (when k varies) to those of the competing algorithms in Figures 4 and 5, respectively. When the position k increases, we find that both Top-RSBC+SGD and Top-RSBC+Online generally outperform the other algorithms over these three datasets when the number of bits is fixed to $r = 256$. This indicates that penalizing the mistakes at the top of a ranking list more than those at the bottom can significantly improve the top-k visual search accuracy. This also suggests that both stochastic gradient descent (SGD) and online learning (Online) are effective for optimizing the proposed objective of Top-RSBC.

Note that both the training time and test time of the proposed Top-RSBC and compared algorithms over the three different datasets are provided in Tables 2, 3, and 4, respectively. We observe that the offline training time of Top-RSBC+SGD is comparable with those of HDML and CGH and longer than the other algorithms, because they all use label information in the form of triplet-level ranks. Since in Top-RSBC+online the gradient update is based upon triplet streaming, it consumes less time than the other algorithms, but its performance is inferior to Top-RSBC+SGD. In contrast to the training time, the online code generation time is more crucial for visual search applications. In term of binary code generation, the main computational cost of Top-RSBC depends on the linear projection and binarization operations. Hence, the test time of Top-RSBC is as efficient as typical linear binary coding or hashing algorithms.

5. Conclusion

In this paper, we proposed a novel supervised binary coding technique, dubbed Top Rank Supervised Binary Coding (Top-RSBC), to conduct large-scale visual search. Unlike the previous supervised binary coding methods, the proposed Top-RSBC aims to learn the disciplined coding functions through explicitly optimizing the precision at the top positions of a Hamming-distance ranking list, thereby preserving the supervised rank information. Since the objective of Top-RSBC is discrete and its associated optimization problem is combinatorially difficult, we relaxed the original discrete objective to a continuous surrogate, and then derived a stochastic gradient descent method to optimize the surrogate objective. To make the optimization more efficient, we also developed an online learning algorithm whose update is based upon triplet streaming. Our experiments on three benchmark image datasets demonstrated the superiority of Top-RSBC over the state-of-the-arts in image search.

Acknowledgements

The work of Dongjin Song and David A. Meyer was supported by the U.S. Department of Defense Minerva Research Initiative/Army under Grant W911NF-09-1-0081 and in part by the National Science Foundation-Division of Mathematical Sciences under Grant 1223137. The work of Rongrong Ji was supported by the Nature Science Foundation of China No. 61422210 and No. 61373076.

References

[1] T. Ahonen, A. Hadid, and M. Pietikainen. Face description with local binary patterns: Application to face recognition. *IEEE Trans. on Pattern Analysis and Machine Interlligence*, 28(12):2037–2041, 2006. 6

[2] A. Andoni and P. Indyk. Near-optimal hashing algorithms for approximate nearest neighbor in high dimensions. *Communications of the ACM*, 51(1):117–122, 2008. 1, 6, 7, 8

[3] A. Z. Broder, M. Charikar, A. M. Frieze, and M. Mitzenmacher. Min-wise independent permuations. In *Proc. of ACM Symposium on Theory of Computing*, 1998. 1

[4] Y. Gong, S. Lazebnik, A. Gordo, and F. Perronnin. Iterative quantization: a procrustean approach to learning binary codes for large-scale image retrieval. *IEEE Trans. on Pattern Analysis and Machine Interlligence*, 2012. 2, 6, 7, 8

[5] Y. Gong, L. Wang, R. Guo, and S. Lazebnik. Multi-scale orderless pooling of deep convolutional activation features. In *Proc. of ECCV*, 2014. 5

[6] W. Kong and W.-J. Li. Istropic hashing. In *Proc. of NIPS 25*, 2012. 2, 6, 7, 8

[7] B. Kulis and T. Darrell. Learning to hash with binary reconstructive embeddings. In *Proc. of NIPS 22*, 2009. 2

[8] B. Kulis and K. Grauman. Kernelized locality-sensitive hashing. *IEEE Trans. on Pattern Analysis and Machine Interlligence*, 34(6):1092 – 1104, June 2012. 1, 7

[9] B. Kulis, P. Jain, and K. Grauman. Fast similarity search for learned metrics. *IEEE Trans. on Pattern Analysis and Machine Interlligence*, 31(12):2143–2157, 2009. 1

[10] P. Li, A. Shrivastava, J. Moore, and A. C. Konig. Hashing algorithms for large-scale learning. In *Proc. of NIPS 24*, 2011. 1

[11] X. Li, G. Lin, C. Shen, A. Hengel, and A. Dick. Learning hash functions using column generation. In *Proc. of ICML*, 2013. 1, 2, 6, 7, 8

[12] W. Liu, C. Mu, S. Kumar, and S.-F. Chang. Discrete graph hashing. In *Proc. of NIPS 27*, 2014. 2

[13] W. Liu, J. Wang, R. Ji, Y.-G. Jiang, and S.-F. Chang. Supervised hashing with kernels. In *Proc. of IEEE CVPR*, 2012. 1, 2, 3, 7

[14] W. Liu, J. Wang, S. Kumar, and S.-F. Chang. Hashing with graphs. In *Proc. of ICML*, 2011. 1

[15] W. Liu, J. Wang, Y. Mu, S. Kumar, and S.-F. Chang. Compact hyperplane hashing with bilinear functions. In *Proc. of ICML*, 2012. 1

[16] M. Norouzi and D. J. Fleet. Minimal loss hashing for compact binary codes. In *Proc. of ICML*, 2011. 2, 3

[17] M. Norouzi, D. J. Fleet, and R. Salakhutdinov. Hamming distance metric learning. In *Proc. of NIPS 25*, 2012. 2, 6, 7, 8

[18] A. Rizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural network. In *Proc. of NIPS 25*, 2012. 5

[19] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A. C. Berg, and L. Fei-Fei. Imagenet large scale visual recognition challenge. In *arXiv:1409.0575*, 2014. 2, 5

[20] D. Song, W. Liu, D. A. Meyer, D. Tao, and R. Ji. Rank preserving hashing for rapid image search. In *Proc. of Data Compression Conference*, pages 353–362, Snowbird, Utah, USA, 2015. 1, 2

[21] D. Song, W. Liu, T. Zhou, D. Tao, and D. A. Meyer. Efficient robust conditional random fields. *IEEE Trans. on Image Processing*, 24(10):3124–3136, 2015. 1

[22] D. Song and D. Tao. Biologically inspired feature manifold for scene classification. *IEEE Trans. on Image Processing*, 19(1):174–184, 2010. 1

[23] A. Torralba, R. Fergus, and W. T. Freeman. 80 million tiny images: a large dataset for non-parametric object and scene recognition. *IEEE Trans. on Pattern Analysis and Machine Interlligence*, 30(11):1958–1970, 2008. 1

[24] A. Torralba, R. Fergus, and Y. Weiss. Small codes and large databases for recognition. In *Proc. of IEEE CVPR*, 2008. 1

[25] N. Usunier, D. Buffoni, and P. Gallinari. Ranking with ordered weighted pairwise classification. In *Proc. of ICML*, pages 1057–1064, 2009. 3

[26] J. Wang, W. Liu, S. Kumar, and S.-F. Chang. Learning to hash for indexing big data - a survey. *To appear in Proc. of the IEEE*, 2015. 1

[27] J. Wang, W. Liu, A. X. Sun, and Y. Jiang. Learning hash codes with listwise supervision. In *Proc. of IEEE ICCV*, 2013. 2, 6, 7, 8

[28] K. Q. Weinberger, A. Dasgupta, J. Langford, A. J. Smola, and S. V. N. Vishwanathan. Feature hashing for large scale multitask learning. In *Proc. of ICML*, 2009. 1

[29] Y. Weiss, A. Torralba, and R. Fergus. Spectral hashing. In *Proc. of NIPS 21*, 2008. 2, 6, 7, 8

[30] J. Weston, S. Bengio, and N. Usunier. Large scale image annotation: learning to rank with joint world-image embeddings. *Machine Learning*, 81(1):21–35, 2012. 3

[31] L. Wolf, T. Hassner, and I. Maoz. Face recognition in unconstrained videos with matched background similarity. In *Proc. of IEEE CVPR*, 2011. 2, 5

[32] J. Xiao, J. Hays, K. A. Ehinger, A. Oliva, and A. Torralba. Sun database: Large-scale scene recognition from abbey to zoo. In *Proc. of IEEE CVPR*, 2010. 2, 5

[33] T. Zhang, C. Du, and J. Wang. Composite quantization for approximate nearest neighbor search. In *Proc. of ICML*, 2014. 1

[34] T. Zhang, G.-J. Qi, J. Tang, and J. Wang. Sparse composite quantization. In *Proc. of IEEE CVPR*, 2015. 1