Fast Supervised Discrete Hashing

Jie Gui, *Senior Member, IEEE*,
Tongliang Liu, *Member, IEEE*,
Zhenan Sun, *Member, IEEE*,
Dacheng Tao, *Fellow, IEEE*,
and Tieniu Tan, *Fellow, IEEE*

Abstract—Learning-based hashing algorithms are "hot topics" because they can greatly increase the scale at which existing methods operate. In this paper, we propose a new learning-based hashing method called "fast supervised discrete hashing" (FSDH) based on "supervised discrete hashing" (SDH). Regressing the training examples (or hash code) to the corresponding class labels is widely used in ordinary least squares regression. Rather than adopting this method, FSDH uses a very simple yet effective regression of the class labels of training examples to the corresponding hash code to accelerate the algorithm. To the best of our knowledge, this strategy has not previously been used for hashing. Traditional SDH decomposes the optimization into three sub-problems, with the most critical sub-problem - discrete optimization for binary hash codes - solved using iterative discrete cyclic coordinate descent (DCC), which is time-consuming. However, FSDH has a closed-form solution and only requires a single rather than iterative hash code-solving step, which is highly efficient. Furthermore, FSDH is usually faster than SDH for solving the projection matrix for least squares regression, making FSDH generally faster than SDH. For example, our results show that FSDH is about 12-times faster than SDH when the number of hashing bits is 128 on the CIFAR-10 data base, and FSDH is about 151-times faster than FastHash when the number of hashing bits is 64 on the MNIST data-base. Our experimental results show that FSDH is not only fast, but also outperforms other comparative methods.

Index Terms—Fast supervised discrete hashing, supervised discrete hashing, learning-based hashing, least squares regression

1 Introduction

There is increasing interest in large-scale visual searching in computer vision, information retrieval, and related areas due to its wide practical utility. Hashing [1], [2], [3], [4], [5], [6], [7], [8], [9] is a powerful and well-established large-scale visual search technique. Hashing generally involves generating a series of hash functions to map each example into a binary feature vector such that the produced hash codes preserve the structure of the original space (e.g., similarities between the original examples).

Existing hashing-based algorithms can be classified into two main categories: data-independent and data-dependent (learning-based). Data-independent methods do not depend on training data, instead using random projections to map examples into a feature space before binarization. Exemplars in this category include locality sensitive hashing (LSH) [10], [11] and its discriminative or kernelized variants [12].

- *J. Gui is with the Institute of Intelligent Machines, Chinese Academy of Sciences, Hefei 230031, People's Republic of China. E-mail: guijie@ustc.edu.*
- *T. Liu and D. Tao are with the UBTech Sydney Artificial Intelligence Institute, and the School of Information Technologies, in the Faculty of Engineering and Information Technologies, The University of Sydney, J12 Cleveland St, Darlington, NSW 2008, Australia. E-mail: tliang.liu@gmail.com, dacheng.tao@sydney.edu.au.*
- *Z. Sun and T. Tan are with the Center for Research on Intelligent Perception and Computing, National Laboratory of Pattern Recognition, Institute of Automation, Chinese Academy of Sciences, Beijing 100190, China and CAS Center for Excellence in Brain Science and Intelligence Technology, Beijing 100190, China. E-mail: {znsun, tnt}@nlpr.ia.ac.cn.*

Manuscript received 10 July 2016; revised 28 Dec. 2016; accepted 28 Feb. 2017. Date of publication 6 Mar. 2017; date of current version 5 Jan. 2018.
Recommended for acceptance by K. Weinberger.
For information on obtaining reprints of this article, please send e-mail to: reprints@ieee.org, and reference the Digital Object Identifier below.
Digital Object Identifier no. 10.1109/TPAMI.2017.2678475

In contrast, data-dependent algorithms take full advantage of training data characteristics. Various statistical learning methods have been used to map examples into binary codes for hash function learning in data-dependent hashing algorithms. Existing data-dependent hashing methods can be divided into: unsupervised, semi-supervised, and supervised methods.

In unsupervised data-dependent hashing methods, the training example labels are not required for learning. For instance, Weiss et al. [13] presented a spectral hashing (SH) algorithm in which the objective function was similar to Laplacian eigenmaps [14]. Gong et al. [15] proposed an iterative quantization (ITQ) algorithm that minimized the binarization loss between hash codes and the original examples. Other unsupervised data-dependent hashing methods include anchor graph hashing (AGH) [16] and inductive manifold hashing (IMH) [17] with t-distributed stochastic neighbor embedding (t-SNE) [18].

Semi-supervised data-dependent hashing algorithms exploit pairwise label information for hash function learning. For instance, Wang et al. [19] proposed a semi-supervised hashing (SSH) algorithm that simultaneously minimized the empirical loss for pairwise labeled training examples and maximized the variance of all training examples (both labeled and unlabeled). Kulis and Darrell [20] proposed a binary reconstructive embedding (BRE) method that minimized the reconstruction error between the learned Hamming distance and the original euclidean distance.

Supervised data-dependent hashing algorithms use training example labels in hash function learning. For instance, Liu et al. [21] proposed a kernel-based supervised hashing (KSH) method that required a limited amount of label information, i.e., similar and dissimilar example pairs. Predictable dual-view hashing [22] was proposed and incorporated the idea of support vector machines (SVMs) in hash learning. Other supervised learning-based hashing algorithms such as fast supervised hashing using graph cuts and decision trees (FastHash) [23], [24] and linear discriminant analysis based hashing (LDAHash) [25] have also been proposed.

Many deep learning algorithms have been proposed over the last few years, some of which have been successfully applied to many practical applications such as image classification and action recognition. Newer methods integrate deep learning and hashing [26], [27], [28] for large-scale visual searching. For example, Liong et al. [29] used deep back-propagation neural networks for hashing, while Lin et al. [30] and Zhang et al. [31] utilized deep convolutional neural networks (CNNs) for hashing.

Hash codes are generally composed of 0 and 1 or -1 and 1. The discrete constraints imposed on the hash codes lead to mixed integer optimization problems, which are generally NP-hard. To simplify the optimization in hash learning, most hashing methods first discard discrete constraints, solve a relaxed problem, and then turn real values into the approximate hash codes by quantization (or thresholding). This relaxation strategy obviously simplifies the original binary optimization problem. However, the approximate solution is suboptimal and reduces the effectiveness of the final hash code, possibly due to the accumulated quantization error, especially when learning long hash codes. Most existing hashing algorithms fail to consider the significance of discrete optimization. In [32], a novel supervised discrete hashing (SDH) algorithm was proposed that directly learned the binary hash codes without relaxation. To make full use of label information, SDH was formulated as a least squares classification that regressed each hash code to its corresponding label.

The ordinary least squares regression may not, however, be optimal for classification. To further improve the performance and speed of SDH, here we propose "fast supervised discrete hashing" (FSDH), a simple method that regresses each label to its corresponding hash code. To the best of our knowledge, this strategy has not previously been utilized in hashing. In SDH, the optimization problem is decomposed into three sub-problems, discrete optimization for hash codes being the most critical. SDH uses discrete cyclic coordinate descent (DCC) iteratively to solve discrete optimization, which is time-consuming. However, FSDH has a closed-form solution for hash learning that only requires a single step instead of iteration to solve the hash code; it is, therefore, highly efficient. When solving the projection matrix for least squares regression (another sub-problem), FSDH is usually faster than SDH. Finally, when solving the projection matrix that projects the nonlinear embedding into low-dimensional space, SDH and FSDH have similar time complexity. Therefore, FSDH is generally faster than SDH. Note that there is only a term change in the objective function of FSDH. FSDH is still non-convex and hence reaches only local minima. Our experimental results show that FSDH not only accelerates SDH but also generally outperforms SDH.

The remainder of the paper is organized as follows. We describe our proposed method in Section 2. Experimental results are presented in Section 3, and we conclude in Section 4.

2 Our Proposed Method

We first introduce the background to FSDH, i.e., SDH [32], in Section 2.1. We then introduce our proposed "fast supervised discrete hashing" (FSDH) method in Section 2.2. Finally, theoretical analysis of FSDH is given in Section 2.3.

2.1 Supervised Discrete Hashing

Given n instances $X = \{x_i\}_{i=1}^n$, our aim is to learn a set of hash codes $B = \{b_i\}_{i=1}^n \in \{-1,1\}^{n\times l}$ to preserve their similarities in the original space, where the ith row vector b_i is the l-bits hash codes for x_i. The labels for all training instances are $Y = \{y_i\}_{i=1}^n \in R^{n\times c}$, where c is the number of classes and $y_{ik} = 1$ if x_i comes from class k and 0 otherwise. The term y_{ik} is the kth element of y_i.

The objective function of SDH is,

$$\begin{aligned}&\min_{B,F,W} \sum_{i=1}^{n} \|y_i - b_i W\|_2^2 + \lambda\|W\|_F^2 + v\sum_{i=1}^{n}\|b_i - F(x_i)\|_2^2 \\ &s.t.\ \forall i \quad b_i \in \{-1,1\}^l.\end{aligned} \tag{1}$$

That is,

$$\begin{aligned}&\min_{B,F,W} \|Y - BW\|_F^2 + \lambda\|W\|_F^2 + v\|B - F(X)\|_F^2 \\ &s.t.\ B \in \{-1,1\}^{n\times l},\end{aligned} \tag{2}$$

where $\|\cdot\|_F$ is the Frobenius norm of a matrix. The first term of (1) is the ordinary least squares regression, which regress each hash code to its corresponding label. The term W is the projection matrix for hash codes. The second term of (1) is for regularization. The term $F(\cdot)$ in the last term of (1) is a simple yet powerful nonlinear embedding to approximate the hash code

$$F(x) = \phi(x)P, \tag{3}$$

where $\phi(x)$ is an m-dimensional row vector obtained by the Gaussian kernel $\phi(x) = [\exp\left(\|x - a_1\|^2\sigma\right), \cdots, \exp\left(\|x - a_m\|^2\sigma\right)]$. The terms $\{a_i\}_{i=1}^m$ are the randomly selected m anchor examples from the training instances, and σ is the Gaussian kernel parameter. The matrix $P \in R^{m\times l}$ projects $\phi(x)$ onto the low-dimensional space. Similar formulations to Equation (3) are widely utilized in other methods such as BRE [20] and KSH [21].

The optimization of (2) has three steps: the F-step, which solves P; the G-step, which solves W; and the B-step, which solves B.

F-Step. By fixing all other variables, the projection matrix P is easily computed,

$$P = \left(\phi(X)^T\phi(X)\right)^{-1}\phi(X)^T B. \tag{4}$$

G-Step. If all other variables are fixed, it is easy to solve W,

$$W = \left(B^T B + \lambda I\right)^{-1} B^T Y. \tag{5}$$

B-Step. By fixing other variables, B also has a closed-form solution. The details can be found in [32].

2.2 Fast Supervised Discrete Hashing

To speed up and further improve SDH's performance, we propose a simple yet effective method called "fast supervised discrete hashing" (FSDH). FSDH's objective function is defined as follows:

$$\begin{aligned}&\min_{B,F,W} \|B - YW\|_F^2 + \lambda\|W\|_F^2 + v\|B - F(X)\|_F^2 \\ &s.t.\ B \in \{-1,1\}^{n\times l}.\end{aligned} \tag{6}$$

SDH and FSDH only differ in the first term. SDH regresses B to Y, while FSDH regresses Y to B. In our view, regressing Y to B is the same as regressing B to Y; the motivation for regressing Y to B is to accelerate the algorithm. Only the first term of FSDH's objective function makes the binary code of each class the same but, due to the third term, the binary code within each class will be different. The first term contributes to the between-class binary code differences while the third term contributes to the binary code differences of all examples.

The problem formulated in (6) is a mixed binary integer program with three unknown variables. We use alternating optimization to iteratively solve the problem. Each iteration alternately updates W, P, B ; thus, the optimization of FSDH also involves three steps, similar to SDH. The details are given below.

F-Step. The F-step of FSDH is the same as that of SDH,

$$P = \left(\phi(X)^T\phi(X)\right)^{-1}\phi(X)^T B. \tag{7}$$

G-Step. If B and P are fixed, (6) can be rewritten as,

$$\begin{aligned}&\min_W\ tr\left((YW - B)^T(YW - B)\right) + \lambda tr\left(W^T W\right) \\ &= \min_W\ tr\left(W^T\left(Y^T Y + \lambda I\right)W\right) - 2tr\left(W^T Y^T B\right).\end{aligned} \tag{8}$$

By setting the derivative of (8) with respect to W to zero, W can be solved with a closed-form solution,

$$W = \left(Y^T Y + \lambda I\right)^{-1} Y^T B. \tag{9}$$

The time complexity of the G-step of FSDH is $O(nc^2 + ncl)$, while the time complexity of the G-step of SDH is $O(nl^2 + l^3 + ncl)$. Generally, l is much larger than c. Thus, the G-step of FSDH is usually faster than that of SDH.

B-Step. When F and W are fixed, let us rewrite (6),

$$\begin{aligned}&\min_B\ tr\left((B - YW)^T(B - YW)\right) \\ &\quad + vtr\left((B - F(X))^T(B - F(X))\right) \\ &s.t.\ B \in \{-1,1\}^{n\times l}.\end{aligned} \tag{10}$$

Since $tr\left(B^T B\right)$ is a constant, (10) is equivalent to

$$\begin{aligned}&\min_B\ -tr\left(B^T(YW + \nu F(X))\right) \\ &s.t.\ B \in \{-1,1\}^{n\times l}.\end{aligned} \tag{11}$$

Thus, B can be solved with a closed-form solution as follows:

$$B = \mathrm{sgn}(YW + \nu F(X)). \tag{12}$$

The B-step of SDH involves discrete cyclic coordinate descent, so the hash code is learnt bit by bit. In contrast, the B-step of FSDH has only a single step to solve all bits, making it much faster than SDH (verified in Section 3). We present the algorithm for solving FSDH in Algorithm 1.

Algorithm 1. Fast Supervised Discrete Hashing (FSDH)

```
Inputs: training examples $\{x_i, y_i\}_{i=1}^n$; code length $l$;
maximum iteration number $t$; parameter $\lambda$
Output: binary codes $\{b_i\}_{i=1}^n \in \{-1,1\}^{n\times l}$
Randomly select $m$ examples $\{a_i\}_{i=1}^m$ from the training
examples and get the $\phi(x)$ via the RBF kernel;
Initialize $b_i$ as a $\{-1,1\}^l$ vector randomly;
Initialize $Y$ as $Y = \{Y_{ij}\} \in R^{n\times c}$ where $Y_{ij} = \begin{cases} 1, & if\ y_i = j \\ 0, & otherwise \end{cases}$;
Use (9) to initialize $W$;
Use (7) to initialize $P$;
repeat
  B-step Use (12) to solve $B$;
  G-step Use (9) to solve $W$;
  F-step Use (7) to solve $P$;
until convergence
```

2.3 Theoretical Analysis of FSDH

In this section, we provide theoretical analysis of FSDH. Specifically, we discuss: (1) why we can replace $\|Y - BW\|_F^2$ with $\|B - YW\|_F^2$; and (2) why the proposed model FSDH is stable while the baseline SDH is not stable for learning the hash code B.

The proposed method replaces the term $\|Y - BW\|_F^2$ with $\|B - YW\|_F^2$ in SDH. We have already discussed how the new term reduces time complexity. However, we must also discuss other similarities and dissimilarities between the two terms. It can be seen that both the $\|B - YW\|_F^2$ and $\|Y - BW\|_F^2$ terms encourage the learned binary code to have the within-class and between-class properties that codes from the same class are similar and dissimilar otherwise. The term $\|B - YW\|_F^2$ achieves this because it encourages each code b_i of x_i to be picked up from W according to the label y_i. Thus, we conclude that $\|B - YW\|_F^2$ and $\|Y - BW\|_F^2$ are the same in the sense for generating the within-class and between-class properties and therefore could be replaced.

We discuss another beneficial property of the newly proposed term $\|B - YW\|_F^2$: it stabilizes the hashing coding algorithm. In a stable algorithm, the output hash codes do not change much if a training example is deleted or replaced with an independent and identically distributed (iid) one. Let $S = (X, Y) = \{z_i = (x_i, y_i)\}_{i=1}^n$ be the training sample for FSDH and S^i be the sample with the ith example $z_i = (x_i, y_i), i = 1, \ldots, n$ in S replaced with an iid one $z'_i = (x'_i, y'_i)$.

Definition 1. *A hashing coding algorithm is $\beta(n)$-stable if the following holds*

$$\forall S, S^i, z_i, z'_i, i = 1, \ldots, n, \|B(S) - B(S^i)\|_F \leq \beta(n),$$

where $B(S)$ and $B(S^i)$ are the hash codes learned by employing S and S^i, respectively, and $\beta(n)$ converges to zero with respect to the sample size n.

FSDH is optimized by employing an alternating iteration method. We assume that the optimization algorithm stops with K iterations and in the kth iteration, where $k = 1, \ldots, K$, B^k, W^k, F^k are obtained. We can prove that FSDH for learning W is stable in each iteration because of the ℓ_2-regularization $\|W\|_F^2$.

Before presenting our result, we first modify the objective function in (6) to ensure that the regularization parameters are invariant to the sample size n, class size c, and code length l. The modified model is as follows:

$$\min_{B,F,W} \frac{1}{nl}\|B - YW\|_F^2 + \frac{\lambda'}{cl}\|W\|_F^2 + \frac{\nu'}{nl}\|B - F(X)\|_F^2. \tag{13}$$

Note that the objective function in (13) is identical to that in (6) by letting $\lambda = \lambda' n/c$ and $\nu = \nu'$. We now prove that FSDH is stable with respect to W in each iteration.

Theorem 1. *In the kth iteration, given B^k and F^{k-1}, FSDH defined in (13) is stable when learning W^k. For any S and S^i, let $W^k(S)$ and $W^k(S^i)$ be learned by employing the sample S and S^i in the kth iteration, respectively. Assume that for any learned B and W, we have $\|b - yW\|_2 \leq M$, where M is a universal constant. Then,*

$$\|W^k(S) - W^k(S^i)\|_F \leq 2cM/\lambda' n. \tag{14}$$

See the proof in the supplementary material, which can be found on the Computer Society Digital Library at http://doi.ieeecomputersociety.org/10.1109/TPAMI.2017.2678475.

Since the newly proposed term $\|B - YW\|_F^2$ encourages each code b_i of x_i to be picked up from W according to the label y_i, the learning algorithm being stable with respect to W implies that it is also stable with respect to code B. Theorem 1 then implies that the difference between $W^k(S)$ and $W^k(S^i)$, as well as the difference between and $B^k(S)$ and $B^k(S^i)$, will decrease as the sample size n increases. Note that although the SDH algorithm is stable with respect to learning W in each step, it is not stable for learning the hash code B because least squares solution is not stable [33]. Note also that Bousquet and Elisseeff [34] proved that stable algorithms will generalize well and that our empirical results in Section 3 support our theoretical analysis by showing that the newly proposed methods generalize well on the test samples.

3 Experiments

In this section, we demonstrate the effectiveness of our proposed method by conducting experiments on two large-scale image datasets (CIFAR-10[1] and MNIST[2]) and a challenging and large-scale face dataset FRGC. Experiments are performed on a server with an Intel Xeon processor (2.80 GHz), 128 GB RAM, and configured with Microsoft Windows Server 2008 and MATLAB 2014b.

We compare our proposed method with representative hashing algorithms including BRE [20], SSH [19], KSH [21], FastHash [23], [24], AGH [16], and IMH [17] with t-SNE [18]. For iterative quantization (ITQ) [15], [35] both its supervised (CCA-ITQ) and unsupervised (PCA-ITQ) versions are utilized. CCA-ITQ uses canonical correlation analysis (CCA) for preprocessing. The public MATLAB codes and model parameters suggested by the corresponding authors are used. For fair comparison, in FSDH and SDH, we empirically set λ, v, and the maximum iteration number t to 1, 1e-5, and 5, respectively, as in [32]. For AGH, IMH, SDH, and FSDH, 1,000 randomly sampled anchor points are utilized.

We report the experimental results using Hamming ranking (mean of average precision, MAP), hash lookup (precision, recall, and F-measure of Hamming radius 2), accuracy, training time, and test time. The F-measure is defined as 2×precision×recall/(precision + recall). We also use the following evaluation metric to measure performance: precision at N examples which is the percentage of true neighbors among the top N retrieved examples. Note that a query is considered as a false instance if no example is returned when calculating precisions. The labels of the examples are defined as the ground truths.

1. http://www.cs.toronto.edu/~kriz/cifar.html
2. http://yann.lecun.com/exdb/mnist/

TABLE 1
Experimental Results on the CIFAR-10 Database When the Number of Hashing Bits Is 128

Method	precision@r=2	recall@r=2	F-measure@r=2	MAP	accuracy	training time	test time
FSDH	**0.3142**	**0.0793**	**0.1266**	0.4639	0.658	32.8	7.4e-6
SDH	0.3017	0.0675	0.1103	0.4668	0.649	406.4	6.8e-6
BRE	0.0080	1.4e-6	2.7e-6	0.1640	0.454	3107.5	4.2e-5
KSH	0.0291	3.8e-4	7.5e-4	0.4823	0.57	10534	8.4e-5
SSH	0.1472	1.2e-4	2.5e-4	0.2222	0.463	132.7	1.1e-5
CCA-ITQ	0.1899	0.0029	0.0056	0.3410	0.568	35.9	3.1e-7
FastHash	0.1010	0.0138	0.0243	**0.6802**	**0.683**	1183.3	3.8e-4
PCA-ITQ	1.0e-3	1.7e-7	3.4e-7	0.1803	0.483	20.0	**2.7e-7**
AGH	0.2502	3.3e-4	6.6e-4	0.1478	0.441	**8.4**	1.1e-4
IMH	0.1599	0.0021	0.0042	0.1811	0.365	59.8	5.3e-5

Training and test times are in seconds. The best results are highlighted in bold face.

3.1 Experimental Results on CIFAR-10

As a subset of the well-known 80M tiny image collection [36], CIFAR-10 contains 60,000 images from 10 classes with 6,000 instances for each class. Each image is represented by a 512-dimensional GIST feature vector [37]. The entire dataset is split into a test set with 1,000 examples and a training set with all remaining examples.

The experimental results on CIFAR-10 are shown in Table 1 when the number of hashing bits is 128. Precision, recall, F-measure of Hamming distance within radius 2, MAP, accuracy, training time, and test time are presented. For SSH, we utilize 5,000 labeled instances for similarity matrix construction. FSDH outperforms SDH in terms of precision, recall, F-measure, and accuracy. FSDH takes only about half a minute to train on all 59,000 training examples. In contrast, KSH and FastHash take about 3 hours and 20 minutes, respectively. CCA-ITQ, SSH, PCA-ITQ, AGH, and IMH are also very efficient; however, their performance is generally worse than FSDH. The precision at 500 examples, precision of Hamming radius 2, and accuracy versus the number of hashing bits are shown in Figs. 1, 2, and 3, respectively (only some comparison methods are shown due to space limitations). With respect to precision of Hamming radius 2, FSDH outperforms the other methods when the number of hashing bits is larger than 32, and KSH performs the best when the number of hashing bits is 16. FSDH outperforms the other methods in terms of accuracy and precision at 500 examples, highlighting the effectiveness of our method.

A critical advantage of FSDH is that it is very fast. For example, FSDH and SDH take 32.8 and 406.4 seconds, respectively, when the number of hashing bits is 128. Thus, FSDH is about 12-times faster than SDH in this case. FastHash also performs very well; however, it is much slower than FSDH. For example, FastHash takes 1183.3 seconds with 128 hashing bits. Thus, FSDH is about 36-times faster than FastHash in this case.

3.2 Experimental Results on MNIST

MNIST contains 70,000 784-dimensional handwritten digit images from 0 to 9. Each image is cropped and normalized to 28×28. The dataset is split into a training set with 69,000 examples and a test set with all remaining examples. The experimental results on MNIST are shown in Table 2. FSDH performs best in terms of recall and F-measure, SDH performs the best in terms of precision, and FastHash performs the best in terms of MAP and accuracy. However,

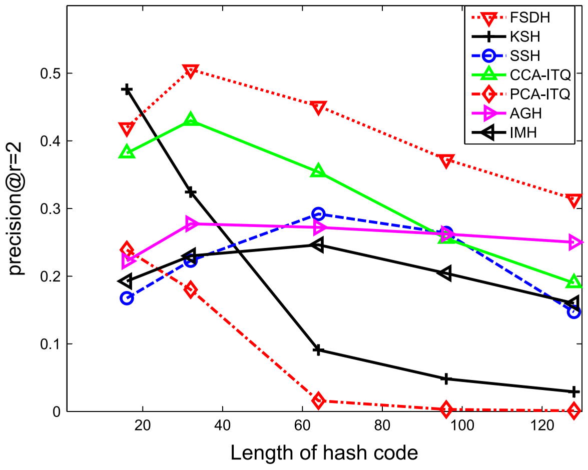

Fig. 2. Precision of Hamming radius 2 as functions of the number of hashing bits (16, 32, 64, 96, 128) on the CIFAR-10 database.

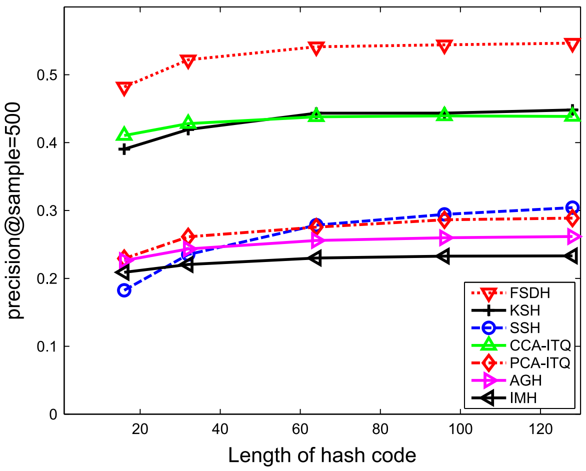

Fig. 1. Precision@sample=500 as functions of the number of hashing bits (16, 32, 64, 96, 128) on the CIFAR-10 database.

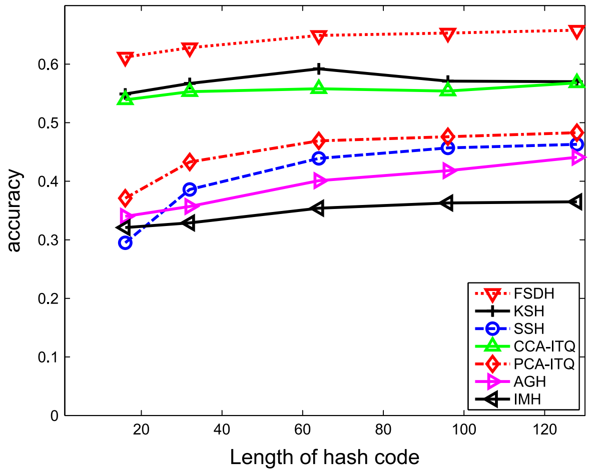

Fig. 3. Accuracy as functions of the number of hashing bits (16, 32, 64, 96, 128) on the CIFAR-10 database.

TABLE 2
Experimental Results on the MNIST Database When the Number of Hashing Bits Is 64

Method	precision@r=2	recall@r=2	F-measure@r=2	MAP	accuracy	training time	test time
FSDH	0.9256	**0.7881**	**0.8513**	0.9410	0.965	30.7	4.3e-6
SDH	**0.9269**	0.7711	0.8419	0.9397	0.963	128	5.1e-6
BRE	0.3850	0.0011	0.0021	0.4211	0.839	24060.6	9.3e-5
KSH	0.6454	0.2539	0.3644	0.9103	0.927	1324.7	7.6e-5
SSH	0.6883	0.0738	0.1332	0.4787	0.734	260.2	5.7e-6
CCA-ITQ	0.7575	0.2196	0.3405	0.7978	0.894	16.6	**4.1e-7**
FastHash	0.8680	0.6735	0.7585	**0.9813**	**0.972**	4661.1	0.0012
PCA-ITQ	0.1680	9.3e-4	0.0018	0.4581	0.886	10.1	4.5e-7
AGH	0.8568	0.0131	0.0258	0.5984	0.899	**6.9**	6.4e-5
IMH	0.8258	0.0889	0.1606	0.6916	0.897	32.2	6.4e-5

Training and test times are in seconds. The best results are highlighted in bold face.

FastHash is much slower than FSDH, taking 4661.1 and 30.7 seconds to train, respectively. Thus, FSDH is about 151-times faster than FastHash in this setting. The precision@sample = 500, precision of Hamming radius 2, recall of Hamming radius 2, F-measure of Hamming radius 2, MAP, and accuracy curves are shown in Figs. 4, 5, 6, 7, 8, and 9, respectively (only some methods are shown due to space limitations). FSDH outperforms all other methods.

3.3 Experimental Results on the FRGC Face Database

The FRGC version two face database [38] is a challenging and large-scale benchmark face database with 8014 face images from 466 individuals in the query set for FRGC experiment 4. These uncontrolled images demonstrate variations in blurring, illumination, expression, and time. In our experiment, only individuals represented by over 10 images in the database are used (3160 images from 316 individuals). Each image is cropped and resized to 32×32 pixels by fixing the eye positions in all experiments, with 256 gray levels per pixel. For each person, seven images are randomly selected for training and the remainder used for testing.

Experimental results on FRGC are shown in Table 3. FSDH performs the best in terms of precision, recall, and F-measure, while SDH performs best for MAP and precision. CCA-ITQ performs the

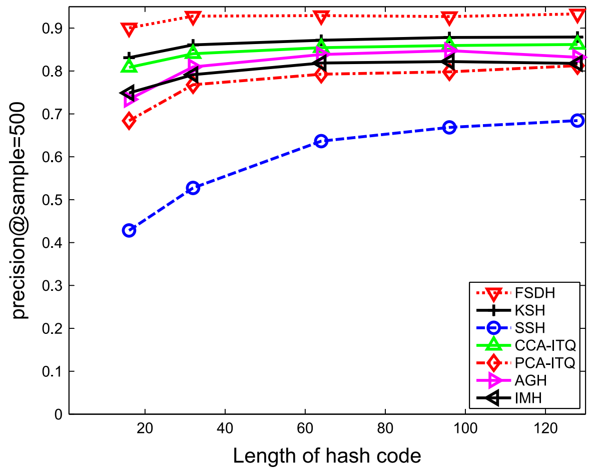

Fig. 4. Precision@sample=500 as functions of the number of hashing bits (16, 32, 64, 96, 128) on the MNIST database.

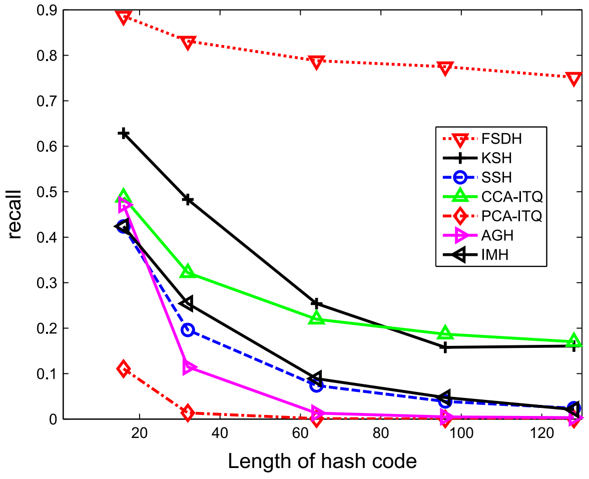

Fig. 6. Recall of Hamming radius 2 as functions of the number of hashing bits (16, 32, 64, 96, 128) on the MNIST database.

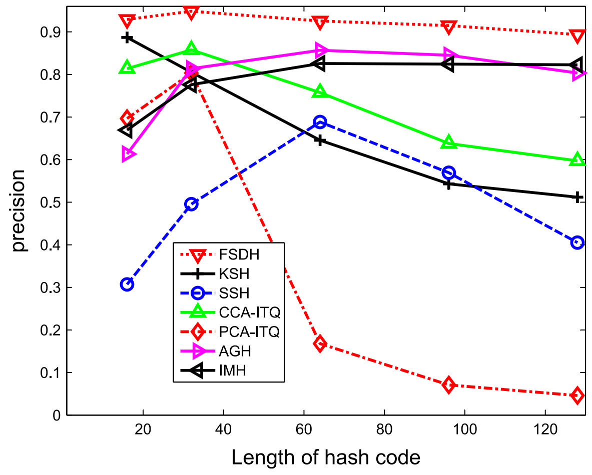

Fig. 5. Precision of Hamming radius 2 as functions of the number of hashing bits (16, 32, 64, 96, 128) on the MNIST database.

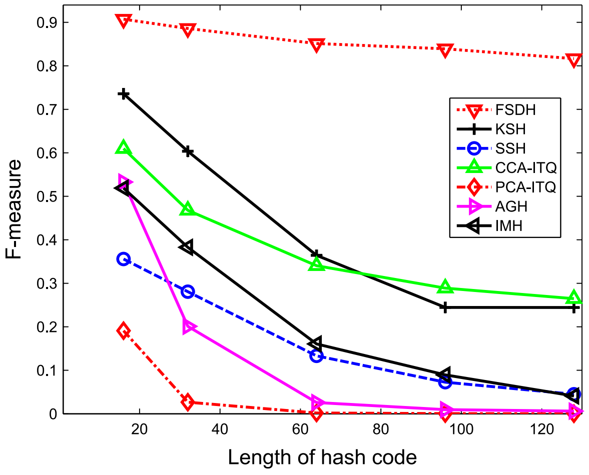

Fig. 7. F-measure of Hamming radius 2 as functions of the number of hashing bits (16, 32, 64, 96, 128) on the MNIST database.

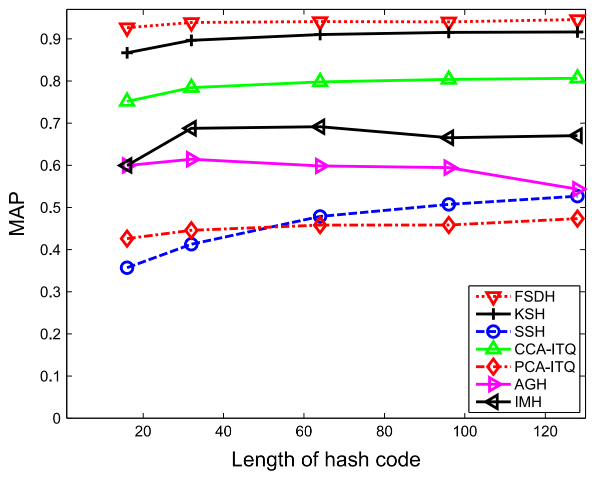

Fig. 8. MAP as functions of the number of hashing bits (16, 32, 64, 96, 128) on the MNIST database.

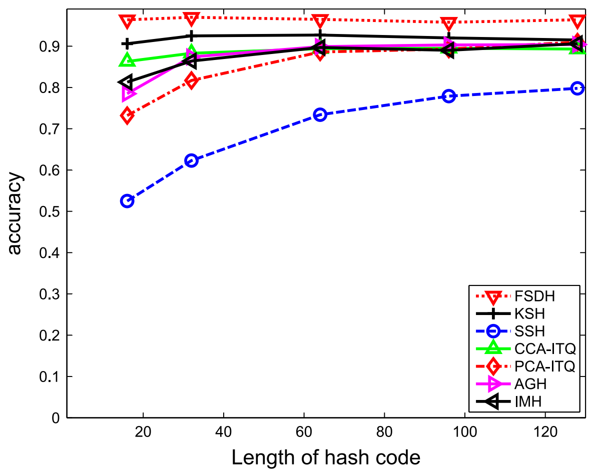

Fig. 9. Accuracy as functions of the number of hashing bits (16, 32, 64, 96, 128) on the MNIST database.

best in terms of accuracy. The MAP versus the number of hashing bits is presented in Fig. 10; due to space limitations, only representative methods are shown. FSDH performs the best with less than or equal to 64 hashing bits, while KSH outperforms the other methods when the number of hashing bits is greater than or equals 128. Furthermore, the MAP of all methods increases as the number of hashing bits increases, perhaps due to more information being encoded in the hash code as the number of hashing bits increases. Thus, the face image is represented by the hash code in a more discriminative and informative way. Several sample query images

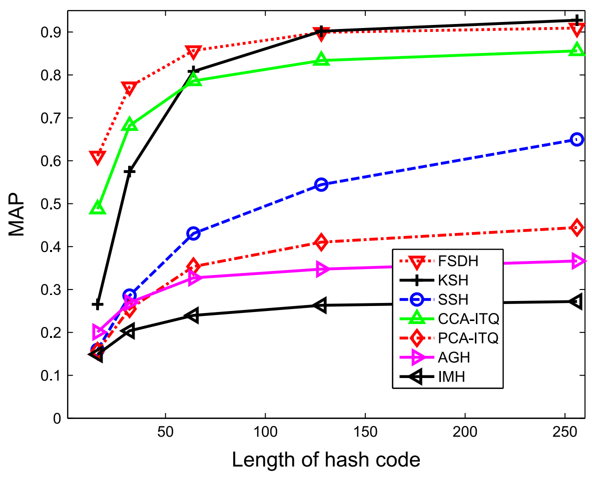

Fig. 10. MAP as functions of the number of hashing bits (16, 32, 64, 128, 256) on the FRGC database.

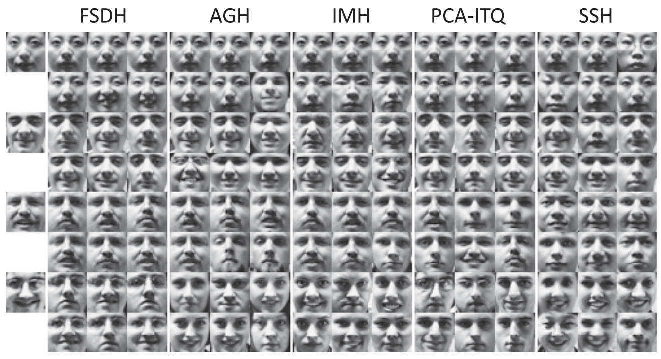

Fig. 11. Top retrieved 6 images of 4 queries returned by various hashing methods on the FRGC data base. The image on the first column is the query instance. From left to right are the retrieved images by FSDH, AGH, IMH, PCA-ITQ and SSH when 16-bit binary codes are utilized for search.

and the retrieved neighbors when 16 bits are utilized to learn binary codes for various hashing algorithms are shown in Fig. 11. FSDH delivers better search results since higher semantic relevance is obtained in the top retrieved instances.

4 Conclusion

In this paper, we present a new data-dependent hashing algorithm called "fast supervised discrete hashing" (FSDH) based on "supervised discrete hashing" (SDH). FSDH regresses the class label to the corresponding hash code, which not only makes it faster but also improves overall performance compared to SDH. Experimental results on image classification and face recognition datasets show that FSDH is very efficient and effective.

TABLE 3
Experimental Results on the FRGC Face Database When the Number of Hashing Bits Is 32

Method	precision@r=2	recall@r=2	F-measure@r=2	MAP	accuracy	training time	test time
FSDH	**0.4400**	**0.3810**	**0.4084**	0.7725	0.753	1.2	2.1e-6
SDH	**0.4400**	0.3740	0.4044	**0.7775**	0.743	2.0	2.2e-6
BRE	0.1690	0.0881	0.1158	0.1458	0.252	216.7	1.8e-5
KSH	0.2383	0.0752	0.1144	0.5748	0.642	736.2	1.2e-4
SSH	0.1856	0.0464	0.0743	0.2858	0.45	9.3	7.4e-6
CCA-ITQ	0.3333	0.1202	0.1767	0.6819	**0.783**	1.0	**1.7e-7**
FastHash	0.0400	0.0057	0.0100	0.2523	0.51	133.6	1.4e-3
PCA-ITQ	0.2172	0.0726	0.1088	0.2548	0.452	**0.3**	2.2e-7
AGH	0.2366	0.3398	0.2789	0.2696	0.391	2.6	1.6e-4
IMH	0.1447	0.3143	0.1982	0.2038	0.287	36.8	9.1e-5

Training and test times are in seconds. The best results are highlighted in bold face.

As an effective and efficient nonlinear feature extraction algorithm, this method can also be applied to other practical applications, especially those involving large-scale data, for example, large-scale mobile video retrieval and visual tracking. Another interesting application of FSDH would be compressing the high-dimensional features into short binary codes, which could significantly speed up large-scale visual tasks such as ImageNet image classification.

In the supervised discrete hashing framework, the hash code is approximated via nonlinear embedding. Deep learning is a current "hot topic", and using deep learning as a nonlinear embedding tool in the SDH framework would be intuitive. However, embedding deep learning in the SDH framework as the nonlinear embedding technique slows the original method. We are now investigating how to effectively and efficiently combine them.

Acknowledgments

This work was supported in part by the grant of the National Science Foundation of China under Grant 61572463 and Grant 61573360, in part by the "Thirteenth Five-Year" National Key Research and Development Program of China under Grant 2016YFD0702002, in part by the grant of Strategic Priority Research Program of the Chinese Academy of Sciences under Grant XDB02080007, in part by the grant of the Open Project Program of the National Laboratory of Pattern Recognition (NLPR) under Grant 201700027, in part by the grant of the Open Project Program of the State Key Lab of CAD&CG under Grant A1709, Zhejiang University, in part by the grant of the Shanghai Key Laboratory of Intelligent Information Processing, China under Grant IIPL-2016-003, and in part by the grant of Australian Research Council Projects under Grant FT-130101457, Grant DP-140102164, and Grant LP-150100671. All correspondence should be directed to Zhenan Sun. Jie Gui and Tongliang Liu contribute equally to this paper.

References

[1] D. Zhang, F. Wang, and L. Si, "Composite hashing with multiple information sources," in *Proc. ACM SIGIR Conf. Res. Develop. Inform. Retrieval*, 2011, pp. 225–234.

[2] J.-P. Heo, Y. Lee, J. He, S.-F. Chang, and S.-E. Yoon, "Spherical hashing: Binary code embedding with hyperspheres," *IEEE Trans. Pattern Anal. and Mach. Intell.*, vol. 37, no. 11, pp. 2304–2316, Nov. 2015.

[3] M. Yu, L. Liu, and L. Shao, "Structure-preserving binary representations for RGB-D action recognition," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 38, no. 8, pp. 1651–1664, Aug. 2016.

[4] W. Zhou, M. Yang, X. Wang, H. Li, Y. Lin, and Q. Tian, "Scalable feature matching by dual cascaded scalar quantization for image retrieval," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 38, no. 1, pp. 159–171, Jan. 2016.

[5] L. Liu, M. Yu, and L. Shao, "Multiview alignment hashing for efficient image search," *IEEE Trans. Image Process.*, vol. 24, no. 3, pp. 956–966, Mar. 2015.

[6] H. Liu, R. Ji, Y. Wu, and W. Liu, "Towards optimal binary code learning via ordinal embedding," in *Proc. AAAI Conf. Artif. Intell.*, 2016, pp. 674–685.

[7] X. Liu, B. Du, C. Deng, M. Liu, and B. Lang, "Structure sensitive hashing with adaptive product quantization," *IEEE Trans. Cybern.*, vol. 46, no. 10, pp. 2252–2264, Oct. 2016.

[8] J. Gui, T. Liu, Z. Sun, D. Tao, and T. Tan, "Supervised discrete hashing with relaxation," *IEEE Trans. Neural Netw. Learning Syst.*, 2016, Doi: 10.1109/TNNLS.2016.2636870.

[9] M. Ou, P. Cui, F. Wang, J. Wang, W. Zhu, and S. Yang, "Comparing apples to oranges: A scalable solution with heterogeneous hashing," in *Proc. ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, 2013, pp. 230–238.

[10] A. Gionis, et al., "Similarity search in high dimensions via hashing," in *Proc. 25th Int. Conf. Very Large Data Bases*, 1999, pp. 518–529.

[11] S. Korman and S. Avidan, "Coherency sensitive hashing," *IEEE Trans. Pattern Anal. and Mach. Intell.*, vol. 38, no. 6, pp. 1099–1112, Jun. 2016.

[12] M. Raginsky and S. Lazebnik, "Locality-sensitive binary codes from shift-invariant kernels," in *Proc. Neural Inform. Process. Syst.*, 2009, pp. 1509–1517.

[13] Y. Weiss, A. Torralba, and R. Fergus, "Spectral hashing," in *Proc. Neural Inform. Process. Syst.*, 2009, pp. 1753–1760.

[14] M. Belkin and P. Niyogi, "Laplacian eigenmaps for dimensionality reduction and data representation," *Neural Comput.*, vol. 15, no. 6, pp. 1373–1396, 2003.

[15] 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. Pattern Anal. and Mach. Intell.*, vol. 35, no. 12, pp. 2916–2929, Dec. 2013.

[16] W. Liu, J. Wang, S. Kumar, and S.-F. Chang, "Hashing with graphs," in *Proc. Int. Conf. Mach. Learning*, 2011, pp. 1–8.

[17] F. Shen, C. Shen, Q. Shi, A. Van Den Hengel, and Z. Tang, "Inductive hashing on manifolds," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2013, pp. 1562–1569.

[18] L. V. D. Maaten and G. Hinton, "Visualizing data using t-SNE," *J. Mach. Learning Res.*, vol. 9, pp. 2579–2605, 2008.

[19] J. Wang, S. Kumar, and S.-F. Chang, "Semi-supervised hashing for large-scale search," *IEEE Trans. Pattern Anal. and Mach. Intell.*, vol. 34, no. 12, pp. 2393–2406, Dec. 2012.

[20] B. Kulis and T. Darrell, "Learning to hash with binary reconstructive embeddings," in *Proc. Neural Inform. Process. Syst.*, 2009, pp. 1042–1050.

[21] W. Liu, J. Wang, R. Ji, Y.-G. Jiang, and S.-F. Chang, "Supervised hashing with kernels," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2012, pp. 2074–2081.

[22] M. Rastegari, J. Choi, S. Fakhraei, H. Daumé III, and L. S. Davis, "Predictable dual-view hashing," in *Proc. Int. Conf. Mach. Learning*, 2013, pp. 1328–1336.

[23] G. Lin, C. Shen, and A. van den Hengel, "Supervised hashing using graph cuts and boosted decision trees," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 37, no. 11, pp. 2317–2331, Nov. 2015.

[24] G. Lin, C. Shen, Q. Shi, A. van den Hengel, and D. Suter, "Fast supervised hashing with decision trees for high-dimensional data," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2014, pp. 1963–1970.

[25] C. Strecha, A. Bronstein, M. Bronstein, and P. Fua, "LDAHash: Improved matching with smaller descriptors," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 34, no. 1, pp. 66–78, Jan. 2012.

[26] H. Lai, Y. Pan, Y. Liu, and S. Yan, "Simultaneous feature learning and hash coding with deep neural networks," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2015, pp. 3270–3278.

[27] W.-J. Li, S. Wang, and W.-C. Kang, "Feature learning based deep supervised hashing with pairwise labels," in *Proc. 25th Int. Joint Conf. Artif. Intell. (IJCAI)*, 2016, pp. 1711–1717.

[28] H. Liu, R. Wang, S. Shan, and X. Chen, "Deep supervised hashing for fast image retrieval," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2016, pp. 2064–2072.

[29] V. Erin Liong, J. Lu, G. Wang, P. Moulin, and J. Zhou, "Deep hashing for compact binary codes learning," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2015, pp. 2475–2483.

[30] K. Lin, H.-F. Yang, J.-H. Hsiao, and C.-S. Chen, "Deep learning of binary hash codes for fast image retrieval," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. Workshops*, 2015, pp. 27–35.

[31] R. Zhang, L. Lin, R. Zhang, W. Zuo, and L. Zhang, "Bit-scalable deep hashing with regularized similarity learning for image retrieval and person re-identification," *IEEE Trans. Image Process.*, vol. 24, no. 12, pp. 4766–4779, Dec. 2015.

[32] F. Shen, C. Shen, W. Liu, and H. Tao Shen , "Supervised discrete hashing," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2015, pp. 37–45.

[33] A. E. Hoerl and R. W. Kennard, "Ridge regression: Applications to nonorthogonal problems," *Technometrics*, vol. 12, no. 1, pp. 69–82, 1970.

[34] O. Bousquet and A. Elisseeff, "Stability and generalization," *J. Mach. Learning Res.*, vol. 2, pp. 499–526, 2002.

[35] Y. Gong and S. Lazebnik, "Iterative quantization: A procrustean approach to learning binary codes," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2011, pp. 817–824.

[36] A. Torralba, R. Fergus, and W. T. Freeman, "80 million tiny images: A large data set for nonparametric object and scene recognition," *IEEE Trans. Pattern Anal. and Mach. Intell.*, vol. 30, no. 11, pp. 1958–1970, Nov. 2008.

[37] A. Oliva and A. Torralba, "Modeling the shape of the scene: A holistic representation of the spatial envelope," *Int. J. Comput. Vis.*, vol. 42, no. 3, pp. 145–175, 2001.

[38] P. J. Phillips, et al., "Overview of the face recognition grand challenge," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2005, pp. 947–954.