Pattern Recognition Letters 107 (2018) 105–113

Contents lists available at ScienceDirect

Pattern Recognition Letters

journal homepage: www.elsevier.com/locate/patrec

SHISS: Supervised hashing with informative set selection

Chao Ma[a], Chen Gong[b,*], Yun Gu[a], Jie Yang[a,*], Deying Feng[c]

[a] Institute of Image Processing and Pattern Recognition, Shanghai Jiao Tong University, Shanghai, China
[b] School of Computer Science and Engineering, Nanjing University of Science and Technology, Nanjing, China
[c] Liaocheng University, China

ARTICLE INFO

Article history:
Available online 12 August 2017

JEL classification:
41A05
41A10
65D05
65D17

Keywords:
Biometrics
Image retrieval
Supervised hashing
Information set selection

ABSTRACT

In recent years, there has been increasing demand for social security and identity authentication, which leads to the booming of many biometrics involved large-scale problems such as recognition, retrieval, and identification. In this case, traditional models are infeasible due to the limited capability for handling large-scale data. Therefore, hashing technique is becoming prevalent due to its low storage cost and fast query speed. Recently, researchers have shown that supervised hashing can achieve higher accuracy than unsupervised hashing by incorporating tag or label information of data for learning hashing function. However, existing supervised methods treat all training examples equally, ignoring the different impacts of various training examples on the learning process. As a result, their performance is not satisfactory under some practical situations. As an improvement, this paper proposes a new method called "Supervised Hashing with Informative Set Selection" (SHISS), which assumes that different training examples have different influence on the learning process, and their usage should follow a logic way during optimization. In particular, we propose two criteria, certainty and diversity, to evaluate the informativeness of the subsets of training examples and encourage the more informative subsets to be learned ahead of the less informative ones. The experiments on two typical image retrieval datasets and one face dataset demonstrate that the proposed SHISS obtains higher mean average precision and shows faster convergence rate compared with the state-of-the-art hashing methods.

1. Introduction

The use of biometrics for identity authentication has gained significant popularity in various applications, such as intelligent surveillance, boarder control, corporate governance, etc [4–6]. Biometrics can be an ideal tool to automatically identify individuals based on their physiological (e.g., fingerprint, iris, palmprint, and vein) and behavioral (e.g., voice, signature, and gait) characteristics [15]. For example, in a face identification system, a query face should be compared with all gallery face images in the system. Therefore, the direct comparison by traditional exhaustive techniques would be intractable to such large-scale face dataset due to their large computational and storage costs in the presence of the high-dimensional feature. Moreover, the examples belonging to the same class may appear quite differently in practical applications. For example, although the face images in Fig. 1 represent the same person, they look very dissimilar due to different poses, lighting conditions and facial expressions. At this time, hashing provides an ideal solution to find a high-level, compact and accurate representation of examples to understand their essential semantic information.

Existing hashing methods can be divided into two categories: data-independent and data-dependent methods. For data-independent methods, such as locality-sensitive hashing (LSH) [8,17,18], the hashing function is randomly generated without using any training data. For data-dependent methods, the hashing functions are learned from a set of training data, therefore data-dependent methods are also called Learning to Hashing (L2H). As this paper studies data-dependent methods, next we briefly review some representative hashing approaches belonging to this category. L2H methods can be unsupervised, semi-supervised, or supervised.

Unsupervised data-dependent hashing methods mainly utilize the data distribution or the underlying manifold structure to design effective indexing schemes. For example, Weiss et al. [36] presented a spectral hashing (SH) algorithm in which the objective function was similar to Laplacian eigenmaps [1]. Gong et al. [11] proposed an iterative quantization (ITQ) algorithm that minimized the binarization loss between hashing codes and the orig-

* Corresponding authors.
E-mail addresses: sjtu_machao@sjtu.edu.cn (C. Ma), chen.gong@njust.edu.cn (C. Gong), jieyang@sjtu.edu.cn (J. Yang).

http://dx.doi.org/10.1016/j.patrec.2017.08.012
0167-8655/© 2017 Elsevier B.V. All rights reserved.

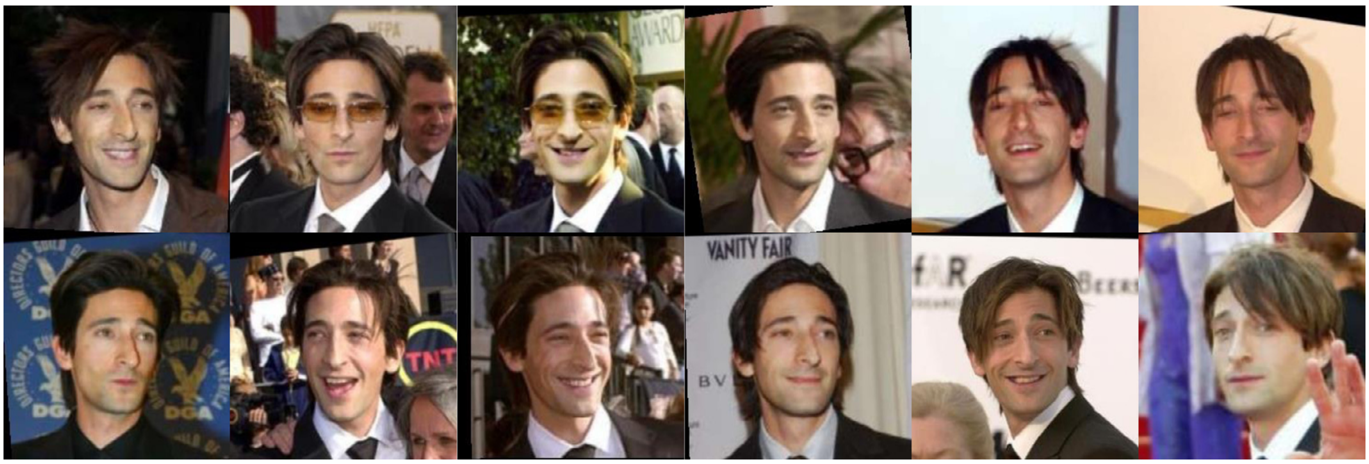


Fig. 1. Face images of a single person. Note that these images are different in pose variations, lighting conditions, and facial expressions.

inal examples. Anchor graph hashing (AGH) [26] and inductive manifold hashing (IMH) [31] mainly utilized the manifold learning algorithm to learn hashing functions.

Semi-supervised data-dependent hashing methods leverage both supervised information and unsupervised information. For example, Kulis and Darrell [20] proposed a binary reconstructive embedding (BRE) algorithm that minimized the reconstruction error between the learning Hamming distance and the original Euclidean distance. Wang et al. [34] proposed a semi-supervised hashing (SSH) method that minimized the empirical error on the labeled data while maximized the entropy of the generated hashing bits over unlabeled data.

Supervised data-dependent methods employ supervised information (e.g., labels or tags) of examples to obtain the discriminative hashing codes. For example, Liu et al. [25] proposed a kernel-based supervised hashing (KSH) algorithm that required a limited amount of label information, i.e. similar and dissimilar example pairs. Zhang et al. [39] proposed a latent factor hashing (LFH) algorithm to learn similarity-preserving binary codes based on the latent factor model. Shen et al. [30] proposed a new supervised hashing framework, where the learning objective was to generate the optimal binary hash codes for linear classification.

Recently, many deep learning algorithms have been proposed, some of which have been successfully applied to hashing tasks [3,23,24,37,40]. For example, Lin et al. [24] and Li et al. [23] utilized convolutional neural networks (CNNs) [22] for supervised hashing. Carreira-Perpinán and Raziperchikolaei [3] proposed a hashing model with binary autoencoder, which sought to reconstruct an image from the binary code produced by the hashing function.

Among the above methods, supervised hashing has been intensively used in various real-world applications because it achieves very impressive accuracy with short hashing codes. However, existing supervised hashing methods treat all the training examples equally when learning the hashing function, and thus the examples are used in an imperfect sequence. To be specific, during their optimization, they simply select a subset of training examples at random and then blindly learn from these examples without considering whether they are beneficial for the ongoing learning process. For example, Zhang et al. [39] proposed a latent factor hashing model (LFH) which is optimized via a stochastic learning solver with linear-time complexity. In this stochastic learning process, the examples are uniformly sampled from the whole training set to form the batches for iterations. As a result, LFH does not consider the influence of every example to the learning process. However, the impacts of various training examples on establishing the hashing function are quite different. For example, selecting the training examples far from potential decision boundary helps little on efficiently deciding the accurate hashing function; picking up the examples all belonging to one class is harmful as in this case the examples are imbalanced and reveal very little class information. Hence, it is necessary to find a way to evaluate the "values" of examples so that they can be rearranged for the learning process.

Similar to our motivation, some prior works [2,9,10,27] have also found that properly organizing the examples for learning is critical to boosting the performance. Curriculum Learning [2], which is inspired by the cognitive process of humans and animals, favors of learning simple examples first, and then gradually taking more complex examples into consideration [7,16]. Gong et al. [9,10] proposed a teaching-to-learn and learning-to-teach framework, which shows that the results of graph-based label diffusion can be improved via a simple-to-difficult propagation sequence. Loshchilov et al. [27] developed an online batch selection method to help stochastic gradient descent (SGD) focus on the most relevant training examples according to their loss values. However, all these methods are not applicable to supervised hashing problems. In supervised hashing, loss functions usually model the similarities between different examples based on the Hamming distance, therefore they cannot reflect the loss regarding every individual example. Besides, above methods do not take measures to avoid the class imbalance issue, which is adverse for supervised hashing to obtain the satisfactory performance.

Based on this consideration, we propose a novel hashing method, dubbed "Supervised Hashing with Informative Set Selection" (SHISS), which aims to perform supervised hashing by establishing a sequence of sets of informative examples during the learning process. Specifically, the informativeness of an example set is evaluated by two criteria including "Certainty" and "Diversity". In particular, the certainty of a set is measured by the distances of its examples to the decision boundary, which depicts how certain the feature vectors of these examples are encoded into the corresponding hashing codes. The diversity criterion evaluates how much class information is potentially included by a set of examples. Therefore, an informative set should be uncertain and diverse. The entire objective function of our model is then integrated by such two criteria and can be optimized via the method of augmented Lagrangian multiplier (ALM). The ability of the proposed algorithm is systematically evaluated on two typical natural image datasets (i.e. CIFAR-10 [19] and MIR-FLICKR [13]) and one challenging large-scale face dataset LFW [12,21]. Experimental results indicate that the proposed algorithm outperforms the existing hashing algorithms. This demonstrates that the proposed SHISS is effective in obtaining compact and accurate hashing codes, and thus it is suitable for various identity authentication applications.

2. Problem description

Suppose we have a data collection $\mathcal{X} = \{\mathbf{x}_i\}_{i=1}^N$, where $\mathbf{x}_i \in \mathbb{R}^d$ is the d-dimensional feature vector of the ith example. Besides, s_{ij} denotes the available pairwise labels between examples, where $s_{ij} = 1$ means $\mathbf{x}_i$ and $\mathbf{x}_j$ are similar and $s_{ij} = 0$ means $\mathbf{x}_i$ and $\mathbf{x}_j$ are dissimilar. The goal of hashing is to learn the hashing codes $\mathbf{b}_i \in \{-1, 1\}^{k\times 1}$ for each $\mathbf{x}_i$ with length k. Ideally, we hope that the Hamming distance between $\mathbf{b}_i$ and $\mathbf{b}_j$ should be small when $s_{ij} = 1$, while the Hamming distance should be large when $s_{ij} = 0$. In general, we can write the hashing code as $\mathbf{b}_i = sgn(\mathbf{W}^T\mathbf{x}_i) = sgn(\mathbf{u}_i)$, where $\mathbf{W} \in \mathbb{R}^{d\times k}$ denotes the projection matrix, $\mathbf{u}_i$ denotes the relaxed codes calculated as $\mathbf{u}_i = \mathbf{W}^T\mathbf{x}_i$, the element of $\mathbf{b}_i$ is 1 if the corresponding element of $\mathbf{u}_i$ is greater than 0, and -1 otherwise. Besides, we use the set $\mathcal{S} = \{s_{ij}\}$ to include all pairs of examples that are similar or dissimilar, and employ the matrix $\mathbf{B} = [\mathbf{b}_1, \cdots, \mathbf{b}_N]^T \in \{-1, 1\}^{N\times k}$ to represent the hashing codes of all N examples.

In this paper, we use LFH model [39] to learn the hashing function as it achieves state-of-the-art performance. In this method, the likelihood F of the pairwise similarity between any two examples is modeled as the function of the Hamming distance, which is:

$$\begin{aligned} F &= \log p(\mathbf{U}|\mathcal{S}) \\ &= \sum_{s_{ij}\in\mathcal{S}} (s_{ij}\Theta_{ij} - \log(1 + e^{\Theta_{ij}})) - \frac{1}{2\alpha}\|\mathbf{U}\|_F^2 + c, \end{aligned} \tag{1}$$

where $\Theta_{ij} = \frac{1}{2}\mathbf{u}_i^T\mathbf{u}_j$, $\mathbf{U} = [\mathbf{u}_1, \cdots, \mathbf{u}_N]^T$, α is a normalized hyper-parameter that is set to a pre-computed optimal value, $\|\cdot\|_F$ denotes the Frobenius norm of a matrix, and c is a constant term independent of $\mathbf{U}$. LFH then learns the optimal $\mathbf{U}$ by maximizing F in Eq. (1). The gradient and the Hessian matrix of the objective function F in Eq. (1) with respect to $\mathbf{U}_{i*}$ can be derived as:

$$\begin{aligned} \frac{\partial F}{\partial \mathbf{U}_{i*}^T} &= \frac{1}{2}\sum_{j:s_{ij}\in\mathcal{S}} (s_{ij} - a_{ij})\mathbf{U}_{j*}^T \\ &\quad + \frac{1}{2}\sum_{j:s_{ji}\in\mathcal{S}} (s_{ji} - a_{ji})\mathbf{U}_{j*}^T - \frac{1}{\alpha}\mathbf{U}_{i*}^T, \end{aligned} \tag{2}$$

$$\begin{aligned} \frac{\partial^2 F}{\partial \mathbf{U}_{i*}^T \partial \mathbf{U}_{i*}} &= -\frac{1}{4}\sum_{j:s_{ij}\in\mathcal{S}} a_{ij}(1 - a_{ij})\mathbf{U}_{j*}^T\mathbf{U}_{j*} \\ &\quad - \frac{1}{4}\sum_{j:s_{ji}\in\mathcal{S}} a_{ji}(1 - a_{ji})\mathbf{U}_{j*}^T\mathbf{U}_{j*} - \frac{1}{\alpha}\mathbf{I}, \end{aligned} \tag{3}$$

where $\mathbf{U}_{i*}$ denotes the ith row of the matrix $\mathbf{U}$.

LFH defines $\mathbf{H}_i$ as:

$$\mathbf{H}_i = -\frac{1}{16}\sum_{j:s_{ij}\in\mathcal{S}} \mathbf{U}_{j*}^T\mathbf{U}_{j*} - \frac{1}{16}\sum_{j:s_{ji}\in\mathcal{S}} \mathbf{U}_{j*}^T\mathbf{U}_{j*} - \frac{1}{\alpha}\mathbf{I}. \tag{4}$$

It can be proved that

$$\frac{\partial^2 F}{\partial \mathbf{U}_{i*}^T \partial \mathbf{U}_{i*}} \succeq \mathbf{H}_i, \tag{5}$$

where $A \succeq B$ denotes that $A - B$ is a positive semi-definite matrix.

Since directly optimizing $\mathbf{U}$ can be very time-consuming, LFH then constructs a lower bound of Eq. (1) and maximizes this bound iteratively. The lower bound $\tilde{F}(\cdot)$ is defined as:

$$\begin{aligned} \tilde{F}(\mathbf{U}_{i*}) &= F(\mathbf{U}_{i*}(t)) + (\mathbf{U}_{i*} - \mathbf{U}_{i*}(t))\frac{\partial F}{\partial \mathbf{U}_{i*}^T}(t) \\ &\quad + \frac{1}{2}(\mathbf{U}_{i*} - \mathbf{U}_{i*}(t))\mathbf{H}_i(t)(\mathbf{U}_{i*} - \mathbf{U}_{i*}(t))^T \end{aligned} \tag{6}$$

where the variables with (t) denote their values in the tth iteration. $\mathbf{U}$ can be iteratively updated by using the rule below:

$$\mathbf{U}_{i*}(t+1) = \mathbf{U}_{i*}(t) - \left[\frac{\partial F}{\partial \mathbf{U}_{i*}^T}(t)\right]^T \mathbf{H}_i(t)^{-1}. \tag{7}$$

From Eq. (7), we see that for a set of N examples, LFH can only update one row of $\mathbf{U}$ at one time, which is time-consuming when N grows large. To handle the large-scale problems, LFH proposes a variant of SGD method with linear-time complexity to solve this model. During the iteration t, LFH randomly selects M examples from the whole training set $\mathcal{X}$ to form the batch $\mathcal{X}_t$ and then constructs $\mathcal{S}_t = \{s_{ij}\}$, where $\mathbf{x}_i \in \mathcal{X}_t$ or $\mathbf{x}_j \in \mathcal{X}_t$. Therefore, $\mathbf{H}_i$ defined in Eq. (4) can be simplified as:

$$\mathbf{H}_i = -\frac{1}{8}\sum_{j:s_{ij}\in\mathcal{S}_t} \mathbf{U}_{j*}^T\mathbf{U}_{j*} - \frac{1}{\alpha}\mathbf{I}. \tag{8}$$

For all $\{i : \mathbf{x}_i \notin \mathcal{X}_t\}$, the associated $\mathbf{H}_i$ are kept unchanged, therefore LFH computes $\mathbf{H}_i^{-1}$ of these examples in a preprocessing step and updates $\mathbf{U}_{i*}$ through only once calculation. By doing so, LFH can efficiently update $\mathbf{U}$ with acceptable computational cost.

From above description, we see that $\mathcal{X}_t$ is randomly generated without considering the informativeness of different examples, so our goal is to improve the optimization process of LFH by advancing the most informative examples among the entire training set for learning, such that the selected examples are consistently helpful for efficient and accurate supervised hashing.

3. Selection of informative examples

In this section, we discuss the details of our SHISS method for deciding the most informative example set for optimization. Specifically, we utilize certainty and diversity criteria to pick up the informative set of examples.

3.1. Certainty

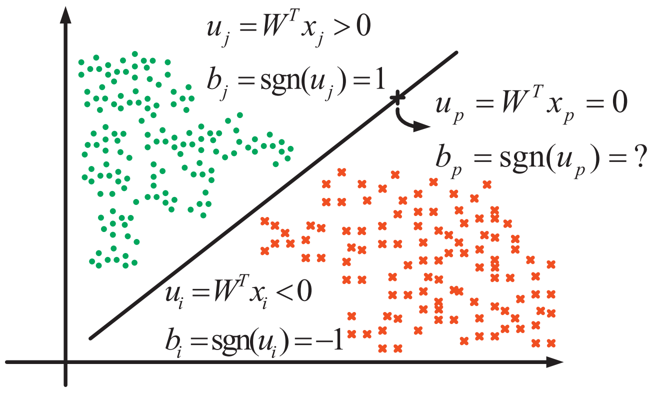
Recall that in our supervised hashing model, the binary hashing code $\mathbf{b}_i$ is obtained by thresholding a projection of an example $\mathbf{x}_i$, which is:

$$\begin{aligned} \mathbf{b}_i &= sgn(\mathbf{W}^T\mathbf{x}_i) \\ &= sgn(\mathbf{w}_1^T\mathbf{x}_i, \cdots, \mathbf{w}_k^T\mathbf{x}_i)^T \\ &= [sgn(\mathbf{u}_i^1), \cdots, sgn(\mathbf{u}_i^k)]^T, \end{aligned} \tag{9}$$

where $\mathbf{W} = [\mathbf{w}_1, \mathbf{w}_2, \cdots, \mathbf{w}_k] \in \mathbb{R}^{d\times k}$, and $\mathbf{u}_i^j$ is the jth element of $\mathbf{u}_i$. Therefore, the linear hashing function based on $\mathbf{W}$ can be viewed as k decision hyperplanes, and each of them is used to generate one bit hashing code. More precisely, if a data point sits on the positive side of a decision hyperplane, its corresponding hashing bit is 1, otherwise -1. Inspired by [35,41], the certainty of an example for obtaining a hashing code can be measured by its distance to the associated hyperplane. Consequently, for the ith example, its distance to the jth hyperplane can be calculated as:

$$dis(\mathbf{x}_i)_j = \frac{|\mathbf{w}_j^T\mathbf{x}_i|}{\|\mathbf{w}_j\|_2} = \frac{|\mathbf{u}_i^j|}{\|\mathbf{w}_j\|_2}. \tag{10}$$

We can find that the shorter the distance of an example to a hyperplane, the more uncertain it is. Fig. 2 shows an extreme case, where an example lies exactly on a decision hyperplane, so it is highly uncertain to decide such example's corresponding bit as 1 or -1.



$u_j = W^T x_j > 0$
$b_j = \text{sgn}(u_j) = 1$
$u_p = W^T x_p = 0$
$b_p = \text{sgn}(u_p) = ?$
$u_i = W^T x_i < 0$
$b_i = \text{sgn}(u_i) = -1$

Fig. 2. An illustration of certainty criterion in $\mathbb{R}^2$ space with two classes and a decision hyperplane. Red crosses represent the examples on the negative side of decision hyperplane with hashing bit -1, green dots denote the examples on the positive side with hashing bit 1. Besides, an example $\mathbf{x}_p$ lies exactly on the decision hyperplane, and thus it is quite uncertain to decide its hashing bit as 1 or -1. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

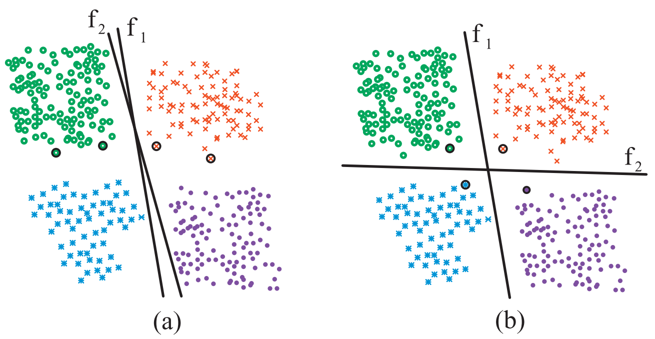


Fig. 3. A setup in $\mathbb{R}^2$ with four classes and two hashing functions f_1 and f_2. The points with different colors belong to different classes and the selected points have been circled in black. Fig. 3(a) shows that if four points from two classes are selected, both f_1 and f_2 partition the entire feature space into two parts. Fig. 3(b) reveals that if diverse examples are chosen, the four classes will be successfully detected by f_1 and f_2.

Note that for all examples, the values of $\|\mathbf{w}_j\|_2$ are the same, so we may directly use $dis(\mathbf{x}_i)_j = |\mathbf{u}_i^j|$ to model the distance of $\mathbf{x}_i$ to the j-th hyperplane. Since there are totally k decision hyperplanes, the certainty c_i of an example $\boldsymbol{x}_i$ is defined by:

$$c_i = \left(\sum_{j=1}^{k} dis(\mathbf{x}_i)_j^2\right)^{\frac{1}{2}} = \|\mathbf{u}_i\|_2. \tag{11}$$

Based on Eq. (11), the certainty of the selected set of examples (denoted by $\mathcal{X}_s$) can be calculated as:

$$Certainty(\mathcal{X}_s) = \sum_{\mathbf{x}_i \in \mathcal{X}_s} c_i. \tag{12}$$

3.2. Diversity

Jiang et al. [14] reveal that diversity among examples should be deployed to enrich the information carried out by an example set. In other words, the selected examples should be sufficiently dissimilar to each other to alleviate the information redundancy for learning. This is quite similar to the cognitive mechanism of humans, who need to learn many different subjects to form a comprehensive knowledge system.

For better illustration, we use a toy example to demonstrate the superiority of selecting diverse data points. As shown in Fig. 3, the entire dataset contains the examples from four classes. Suppose we choose the four examples illustrated in Fig. 3(a), the hashing function f_2 will play the same role as f_1 in deciding the hashing codes of the four data points. In contrast, if the diverse examples are selected as shown in Fig. 3(b), f_1 and f_2 altogether will assign different 2-bit hashing codes to the four examples belonging to four different classes.

Based on this consideration, we propose to measure the diversity of a set according to the pairwise similarities of its included examples. To be specific, the similarity d_{ij} between $\mathbf{x}_i$ and $\mathbf{x}_j$ is defined as:

$$d_{ij} = \frac{\mathbf{u}_i^T \mathbf{u}_j}{\|\mathbf{u}_i\|_2 \|\mathbf{u}_j\|_2}, \tag{13}$$

then the diversity of the selected set of examples is calculated as:

$$Diversity(\mathcal{X}_s) = \sum_{\mathbf{x}_i, \mathbf{x}_j \in \mathcal{X}_s} d_{ij}. \tag{14}$$

Note that here we do not use $\mathbf{x}_i$ and $\mathbf{x}_j$ in the original feature space to characterize the pairwise similarity as in [38]. Instead, we use the projection results of $\mathbf{x}_i$ and $\mathbf{x}_j$ (i.e. $\mathbf{u}_i$ and $\mathbf{u}_j$) in hashing space to assess the similarity between two examples, as this representation is concise, high-level and also related to the hashing scenario.

3.3. Overall objective and optimization

The entire objective function for example selection integrates the certainty component in Eq. (12) and the diversity component given in Eq. (14), which is:

$$\min_{\mathcal{X}_s} \sum_{\mathbf{x}_i \in \mathcal{X}_s} c_i + \beta \sum_{\mathbf{x}_i, \mathbf{x}_j \in \mathcal{X}_s} d_{ij}, \tag{15}$$

where $\beta > 0$ is a trade-off parameter. By solving the above optimization problem, the most informative set $\mathcal{X}_s$ of examples can be found.

However, Eq. (15) is symbolic and cannot be solved directly. To formalize the objective, we introduce an indicating vector $\boldsymbol{\mu} \in \{0, 1\}^{N\times 1}$ in which the ith element $\boldsymbol{\mu}_i$ specifies whether or not the corresponding example should be selected, i.e., $\boldsymbol{\mu}_i = 1$ when $\mathbf{x}_i$ is selected and $\boldsymbol{\mu}_i = 0$ otherwise. Eq. (15) can then be rewritten as:

$$\begin{aligned} \min_{\boldsymbol{\mu}} \quad & \boldsymbol{\mu}^T \mathbf{C}_d + \beta \boldsymbol{\mu}^T \mathbf{D}_s \boldsymbol{\mu} \\ s.t. \quad & \boldsymbol{\mu} \in \{0, 1\}^{N\times 1}, \\ & \boldsymbol{\mu}^T \mathbf{1} = L, \end{aligned} \tag{16}$$

where $\mathbf{C}_d = [c_1, c_2, \cdots, c_N]^T$ is the data certainty vector and $\mathbf{D}_s = \{d_{ij}\}^{N\times N}$ is the data diversity matrix, $\mathbf{1}$ is an all-one vector, and the constraint $\boldsymbol{\mu}^T\mathbf{1} = L$ means that we wish to select exact L ($L = M \times T$, where M is the batch size, and T is the maximal number of iterations.) examples to form the most informative set $\mathcal{X}_s$.

Directly solving the problem Eq. (16) is intractable due to the integer constraint that will make the problem NP-hard. Therefore, we relax the constraint to the continuous constraint $\mathbf{0} \le \boldsymbol{\mu} \le \mathbf{1}$, and thus Eq. (16) can be reformulated as:

$$\begin{aligned} \min_{\boldsymbol{\mu}} \quad & \boldsymbol{\mu}^T \mathbf{C}_d + \beta \boldsymbol{\mu}^T \mathbf{D}_s \boldsymbol{\mu} \\ s.t. \quad & \mathbf{0} \le \boldsymbol{\mu} \le \mathbf{1}, \\ & \boldsymbol{\mu}^T \mathbf{1} = L. \end{aligned} \tag{17}$$

Eq. (17) can be easily solved by the method of ALM. The main idea of ALM is to transform a constrained optimization problem into a non-constrained problem by incorporating penalty terms. Compared with the traditional Lagrangian method, ALM adds an additional quadratic penalty function to the objective, which leads to faster convergence rate and lower computational cost.

The augmented Lagrangian function of Eq. (17) is:

$$\begin{aligned} &L(\boldsymbol{\mu}, \boldsymbol{\lambda}_1, \boldsymbol{\lambda}_2, \lambda_3, \mathbf{T}_1, \mathbf{T}_2, \sigma) \\ &= \boldsymbol{\mu}^T \mathbf{C}_d + \beta \boldsymbol{\mu}^T \mathbf{D}_s \boldsymbol{\mu} + \frac{\sigma}{2} \|\boldsymbol{\mu} - \mathbf{T}_1\|_2^2 - \boldsymbol{\lambda}_1^T (\boldsymbol{\mu} - \mathbf{T}_1) \\ &\quad + \frac{\sigma}{2} \|\mathbf{1} - \boldsymbol{\mu} - \mathbf{T}_2\|_2^2 - \boldsymbol{\lambda}_2^T (\mathbf{1} - \boldsymbol{\mu} - \mathbf{T}_2) \\ &\quad + \frac{\sigma}{2} (\boldsymbol{\mu}^T \mathbf{1} - L)^2 - \lambda_3 (\boldsymbol{\mu}^T \mathbf{1} - L), \end{aligned} \tag{18}$$

where $\boldsymbol{\lambda}_1$, $\boldsymbol{\lambda}_2$, λ_3 are the Lagrangian multipliers, $\mathbf{T}_1$, $\mathbf{T}_2$ are the non-negative auxiliary vectors, and σ is the penalty coefficient.

The derivative of the augmented Lagrangian function $L(\boldsymbol{\mu}, \boldsymbol{\lambda}_1, \boldsymbol{\lambda}_2, \lambda_3, \mathbf{T}_1, \mathbf{T}_2, \sigma)$ with respect to $\boldsymbol{\mu}$ is calculated as follow:

$$\begin{aligned} \frac{\partial L}{\partial \boldsymbol{\mu}} &= (2\beta \mathbf{D}_s + 2\sigma \mathbf{I} + \sigma \mathbf{1}^T \mathbf{1}) \boldsymbol{\mu} \\ &\quad - [\sigma (\mathbf{T}_1 - \mathbf{T}_2 + \mathbf{1} + L\mathbf{1}) + \boldsymbol{\lambda}_1 - \boldsymbol{\lambda}_2 + \lambda_3 \mathbf{1} - \mathbf{C}_d]. \end{aligned} \tag{19}$$

Let

$$\mathbf{A} = 2\beta \mathbf{D}_s + 2\sigma \mathbf{I} + \sigma \mathbf{1}^T \mathbf{1}, \tag{20}$$

and

$$\mathbf{p} = \sigma (\mathbf{T}_1 - \mathbf{T}_2 + \mathbf{1} + L\mathbf{1}) + \boldsymbol{\lambda}_1 - \boldsymbol{\lambda}_2 + \lambda_3 \mathbf{1} - \mathbf{C}_d, \tag{21}$$

and then the derivative of Lagrangian function with respect to $\boldsymbol{\mu}$ can be rewritten as:

$$\frac{\partial L}{\partial \boldsymbol{\mu}} = \mathbf{A}\boldsymbol{\mu} - \mathbf{p}. \tag{22}$$

By setting the value of Eq. (22) to zero, the indicating vector $\boldsymbol{\mu}$ is updated by:

$$\boldsymbol{\mu} = \mathbf{A}^{-1} \mathbf{p}. \tag{23}$$

The entire iterative ALM process for solving Eq. (17) is summarized in Algorithm 1, which is guaranteed to converge according to [29]. After obtaining the solution of Eq. (17), we select L examples with the largest $\boldsymbol{\mu}$ to form the optimal set $\mathcal{X}_s$, in which the examples are sorted by the value of $\boldsymbol{\mu}$ in descending order.

Algorithm 1 The algorithm for solving Eq. (17).

```
Input: $\mathbf{C}_d$, $\mathbf{D}_s$, $L$, $\beta$, $\sigma = 1$, $\rho = 1.5$, $iter = 0$
 1: repeat
 2:     // Compute $\mathbf{T}_1, \mathbf{T}_2$
 3:     $\mathbf{T}_1 = max(\mathbf{0}, \boldsymbol{\mu} - \boldsymbol{\lambda}_1/\sigma)$;
 4:     $\mathbf{T}_2 = max(\mathbf{0}, \mathbf{1} - \boldsymbol{\mu} - \boldsymbol{\lambda}_2/\sigma)$;
 5:     // Update $\boldsymbol{\mu}$ by using Eq.(23)
 6:     $\boldsymbol{\mu} := \mathbf{A}^{-1}\mathbf{p}$;
 7:     // Update variables
 8:     $\boldsymbol{\lambda}_1 := max(\mathbf{0}, \boldsymbol{\lambda}_1 - \sigma\boldsymbol{\mu})$;
 9:     $\boldsymbol{\lambda}_2 := max(\mathbf{0}, \boldsymbol{\lambda}_2 - \sigma(\mathbf{1} - \boldsymbol{\mu}))$;
10:     $\lambda_3 := \lambda_3 - \sigma(\boldsymbol{\mu}^T\mathbf{1} - L)$;
11:     $\sigma := min(\rho\sigma, 10^{10})$;
12:     $iter := iter + 1$;
13: until Convergence
Output: $\boldsymbol{\mu}$ that minimizes Eq.(17)
```

The full SHISS algorithm is summarized in Algorithm 2. Firstly, we initialize $\mathbf{U}$ by performing PCA on $\mathcal{X}$, which is common in many popular hashing methods such as ITQ [11] and LFH [39]. Secondly, we solve the optimization problem in Eq. (17) to form the optimal set $\mathcal{X}_s$. Next, we sequentially select M examples from $\mathcal{X}_s$ to update $\mathbf{U}$. Finally, the hashing codes $\mathbf{B}$ are obtained by binarizing the relaxed codes $\mathbf{U}$.

Algorithm 2 Learning algorithm for SHISS.

```
Input: $\mathcal{X}$, $\mathcal{S}$, $\varepsilon$, $M$, $T$
1: Initialize $\mathbf{U}$ by performing PCA on $\mathcal{X}$.
2: Compute $\mathbf{C}_d$ and $\mathbf{D}_s$ by using Eqs.(11) and (13).
3: Solve the optimization problem in Eq.(17) to form the optimal
   set $\mathcal{X}_s$.
4: for $t = 1 \to T$ do
5:     Update $\mathbf{U}$ in LFH with $M$ examples by using Eq. (7).
6:     Compute loss function $F$ in Eq.(1) using the updated $\mathbf{U}$.
7:     Terminate the iterative process when the change of $F$ is
   smaller than $\varepsilon$.
8: end for
9: $\mathbf{B} = sgn(\mathbf{U})$.
Output: Relaxed codes $\mathbf{U}$, hashing codes $\mathbf{B}$.
```

3.4. Out-of-sample extension

In order to perform the retrieval, the binary code $\mathbf{b}_q$ for a query $\mathbf{x}_q$ need to be computed. We achieve this by finding the projection matrix $\mathbf{W} \in \mathbb{R}^{d \times k}$ that maps $\mathbf{x}$ to $\mathbf{u}$ in the following way:

$$\mathbf{u} = \mathbf{W}^T \mathbf{x}. \tag{24}$$

The liner regression is used to train $\mathbf{W}$ over the training set. The squared loss L_e with regularization term is shown below:

$$L_e = \|\mathbf{U} - \mathbf{X}\mathbf{W}\|_F^2 + \lambda_e \|\mathbf{W}\|_F^2. \tag{25}$$

The optimal $\mathbf{W}$ can be calculated as:

$$\mathbf{W} = (\mathbf{X}^T \mathbf{X} + \lambda_e \mathbf{I})^{-1} \mathbf{X}^T \mathbf{U}. \tag{26}$$

4. Experiments

This section presents the experimental results and demonstrates the effectiveness of the proposed approach. We start by introducing the datasets and setting, and then provide the comparison results of our method with existing approaches.

4.1. Dataset and setting

We demonstrate the effectiveness of our proposed method by conducting experiments on two typical image datasets (CIFAR-10 [19] and MIR-FLICKR [13]), and one challenging and large-scale face dataset LFW [12,21]. As a subset of well-known 80M tiny image collection [32], CIFAR-10 contains 60,000 images which are categorized into 10 classes (6,000 images per class) and the original size of the contained color images is 32×32. It is a single-label dataset in which each image belongs to one of ten classes. MIR-FLICKR contains 25,000 images, and it is a multi-label dataset in which each image is annotated with one or multiple class labels from 38 classes. LFW is a challenging and large-scale face dataset designed for studying the problem of unconstrained face recognition. The dataset contains more than 13,000 images of faces collected from the web and each face has been labeled with the name of the person pictured. In our experiment, only persons represented by over 10 images in the dataset are used (4,272 images from 154 persons). Each image is cropped and normalized to 250×250.

For all datasets, we represent each image by a 512-dimensional GIST vector [28]. We perform normalization on the feature vectors of all examples to make each dimension have zero mean and equal variance. Two images are considered to be semantically similar if they share the same class label, otherwise they are dissimilar. As in LFH [39], we randomly choose 1000 examples as validation set and 1000 examples as test set for all datasets. The rest of examples form the training set. We run the compared methods ten times

Table 1
MAP results of various compared methods. The best MAP under each setting is shown in boldface.

Methods	CIFAR-10				MIR-FLICKR				LFW			
	8-bits	16-bits	32-bits	64-bits	8-bits	16-bits	32-bits	64-bits	8-bits	16-bits	32-bits	64-bits
SHISS	**0.3638**	**0.4925**	**0.5417**	**0.5932**	**0.6725**	**0.6806**	**0.6968**	**0.7038**	**0.1482**	**0.1828**	0.1891	0.2085
LFH	0.3372	0.4532	0.5176	0.5678	0.6402	0.6590	0.6699	0.6770	0.1312	0.1651	0.1658	0.1955
BA	0.1349	0.1458	0.1500	0.1553	0.5678	0.5458	0.5476	0.5597	0.0308	0.0355	0.0373	0.0419
SDH	0.2652	0.3964	0.4145	0.4346	0.6031	0.6066	0.6096	0.6148	0.1037	0.1470	**0.2058**	**0.2293**
KSH	0.2334	0.2662	0.2933	0.3128	0.5702	0.5744	0.5701	0.5742	0.0412	0.0423	0.0470	0.0452
ITQ	0.1388	0.1105	0.1212	0.1216	0.4427	0.4381	0.4591	0.4643	0.0264	0.0283	0.0293	0.0299
PCAH	0.1562	0.1406	0.1226	0.1207	0.4690	0.4576	0.4601	0.4555	0.0269	0.0288	0.0299	0.0311
SH	0.1601	0.1444	0.1185	0.1226	0.4696	0.4691	0.4581	0.4550	0.0271	0.0261	0.0272	0.0282
LSH	0.1511	0.1249	0.1257	0.1210	0.4646	0.4645	0.4617	0.4539	0.0259	0.0278	0.0287	0.0294

with different splits of training / validation / test sets, and the reported accuracies are calculated as the mean value of the outputs of these ten independent runs.

We compare our SHISS method with several state-of-the-art hashing methods, which include:

- Data-independent method: locality-sensitive hashing (LSH) [8];
- Unsupervised hashing methods: spectral hashing (SH) [36], principal component analysis based hashing (PCAH) [33], iteration quantization (ITQ) [11];
- Supervised hashing methods: kernel-based supervised hashing (KSH) [25], supervised hashing with latent factor models (LFH) [39], supervised discrete hashing (SDH) [30].
- Deep hashing method: hashing with binary autoencoders (BA) [3].

The source codes of all the baselines are provided by the corresponding authors. For LFH, the batch size is set to $M = 100$ on CIFAR-10 and MIR-FLICKR, and $M = 50$ on LFW. For KSH, the number of support vectors is 300 as suggested by [25]. For our SHISS, the trade-off parameter is set to $\beta = 1$ on all three datasets. The batch size is set to $M = 100$ on CIFAR-10 and MIR-FLICKR, and $M = 50$ on LFW. The reasons will be explained in Section 4.4. All experiments are conducted on a workstation with Intel Xeon E5 CPU @3.70 Ghz and 16GB RAM.

We report the experimental results using Mean Average Precision (MAP). MAP is a widely used metric for evaluating the performance of hashing methods. For a query q, all the points in the training set are ranked according to the Hamming distance between their hashing codes and the query's. The average precision $AP(q)$ is then defined as:

$$AP(q) = \frac{1}{L_q} \sum_{r=1}^{R} P_q(r)\delta_q(r), \tag{27}$$

where L_q is the number of groundtruth neighbors in the ranked list, $P_q(r)$ is the precision of the top r results, and $\delta_q(r) = 1$ if the r-th result is the real neighbor and 0 otherwise.

4.2. Effect of informative set selection

By comparing the MAP results of SHISS and LFH shown in Table 1, we find that by carefully selecting the most informative examples for optimization, our method performs better than LFH with random example selection. This convincingly demonstrates the effectiveness of our informative example selection strategy by considering both certainty and diversity.

Besides, since SHISS improves the iterative optimization process of LFH, we also compare the convergence rates of the two methods. Specifically, in Fig. 4, we plot the convergence curves of SHISS and LFH for generating 16-bit hashing codes on all three datasets. It can be observed that SHISS converges much faster than LFH. This is because the selected examples with small certainty are usually close to the potential decision boundary, so updating with them will help the iterative optimization process converge to a critical solution quickly.

Moreover, we also present the variations of MAP output by SHISS and LFH on the validation set during optimization process (see Fig. 5). We find that due to the example selection, SHISS yields higher MAP than LFH when the iteration proceeds, which reflects that the hashing codes determined by SHISS can faithfully preserve the real similarity relationship among examples.

4.3. Results

We perform the MAP results of compared methods with different code lengths on three datasets in Table 1. We can find that our SHISS method consistently outperforms all the other baselines on CIFAR-10 and MIR-FLICKR. On LFW dataset, SHISS performs the best with 8-bit and 16-bit hashing codes compared with other baselines. When the number of hashing bits is 32 and 64, SDH performs the best on LFW, but SHISS still outperforms other methods significantly. These results demonstrate the remarkable ability of SHISS for obtaining better hashing function and preserving the real similarity relationship among examples.

Furthermore, in order to illustrate the effectiveness of SHISS, we present the face retrieval results with 16-bit binary codes on LFW dataset. The top ten images for a query returned by four different hashing methods (e.g., SHISS, LFH, SDH, and ITQ) are shown in Fig. 8. The incorrect retrieval results are outlined in red rectangles. As we can see, our SHISS method is able to find the most relevant face images when compared with other hashing methods.

4.4. Parameter sensitivity

In this section, we study the impacts of batch size M and trade-off parameter β on the performance of SHISS. We firstly examine the influence of M on the MAP output by SHISS. To this end, we fix β to 1 and vary M from 10 to 100 on CIFAR-10 and MIR-FLICKR, and vary M from 10 to 50 on LFW. The obtained MAPs with 16-bit codes are shown in Fig. 6. We observe that the performances increase as the batch size M increases on all datasets. This is because large M implies that SHISS incorporates many examples for one iteration, so the performance of SHISS will be improved. However, if M is too large, which means that many examples are to be selected, the selection process (i.e. the optimization of Eq. (17)) will be very time-consuming. Therefore, we make a trade-off between efficiency and MAP, and set M as indicated in Section 4.1.

In order to investigate the impact of the selection of β to the final performance, we fix $M = 100$ on CIFAR-10 and MIR-FLICKR, and $M = 50$ on LFW. We then vary β from 0.01 to 100. The change of the obtained MAP on all three datasets are reported in Fig. 7, which reveals that the performance of SHISS is not sensitive to the selection of β, so we simply set this parameter to 1 as introduced in Section 4.1 as this parameter in our method can be easily tuned.

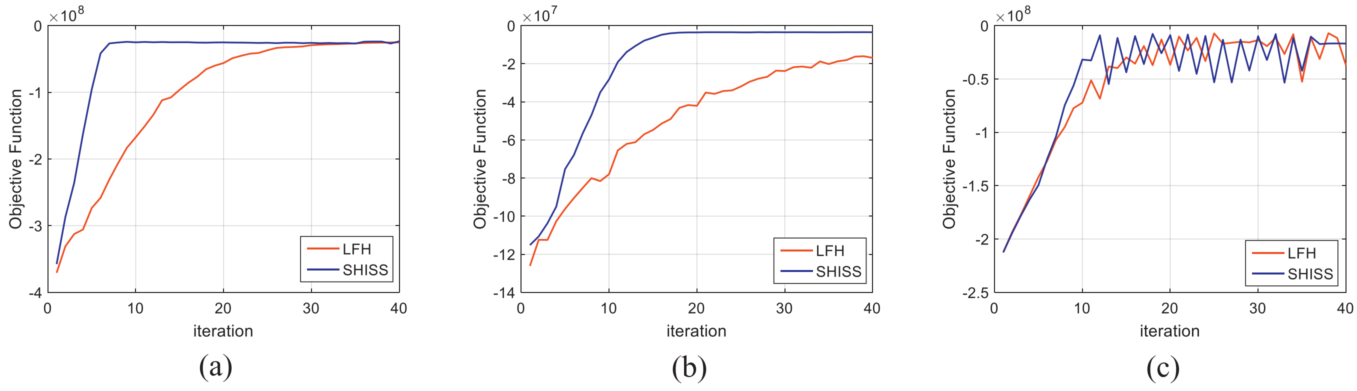


Fig. 4. Convergence curves of LFH and SHISS with 16 hashing bits. (a) is CIFAR-10. (b) is MIR-FLICKR. (c) is LFW.

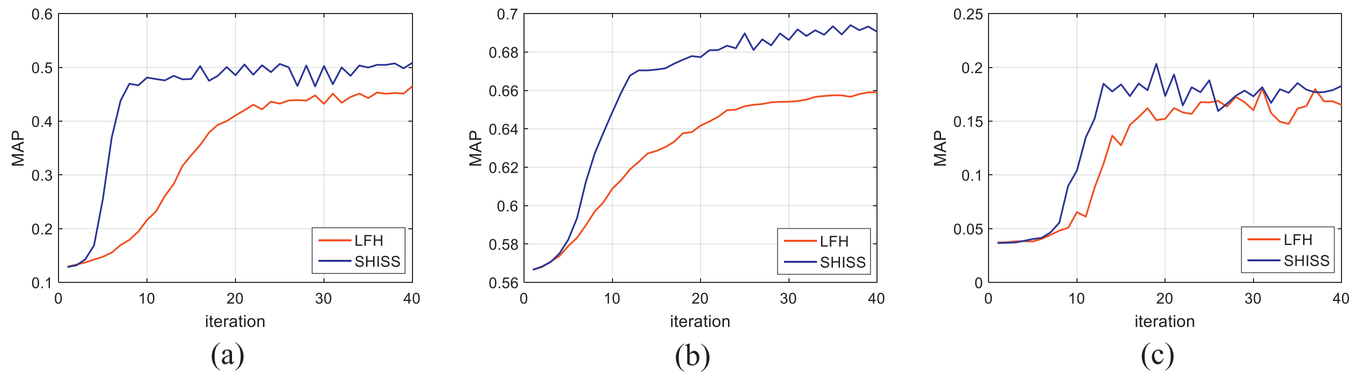


Fig. 5. The MAP variations of LFH and SHISS during the iterations. (a) is CIFAR-10. (b) is MIR-FLICKR. (c) is LFW.

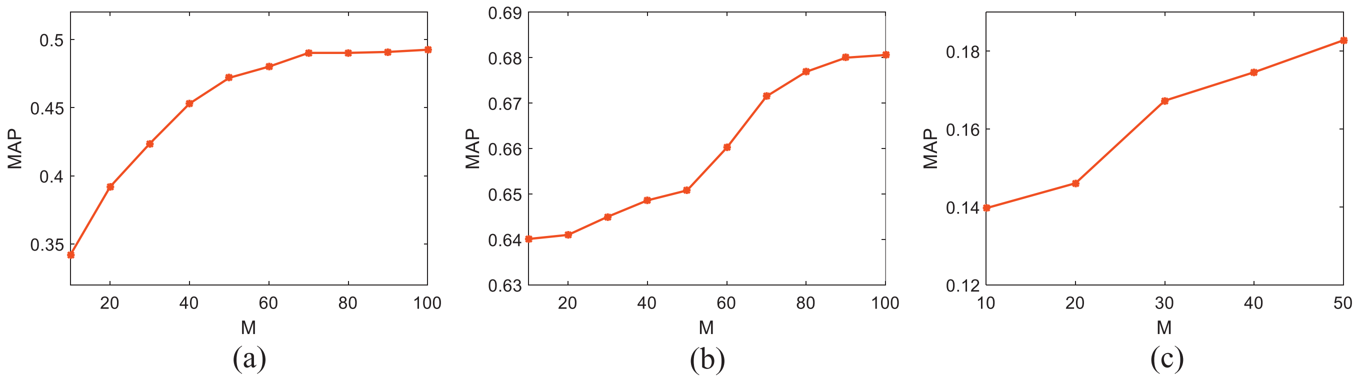


Fig. 6. MAP results obtained under different batch sizes. (a) is CIFAR-10. (b) is MIR-FLICKR. (c) is LFW.

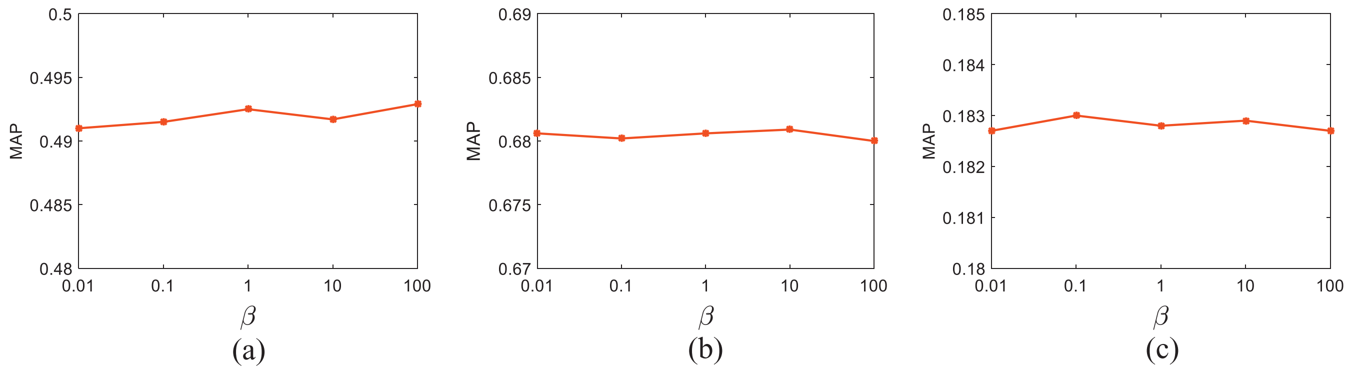


Fig. 7. Parameter sensitivity of β. (a) is CIFAR-10. (b) is MIR-FLICKR. (c) is LFW.

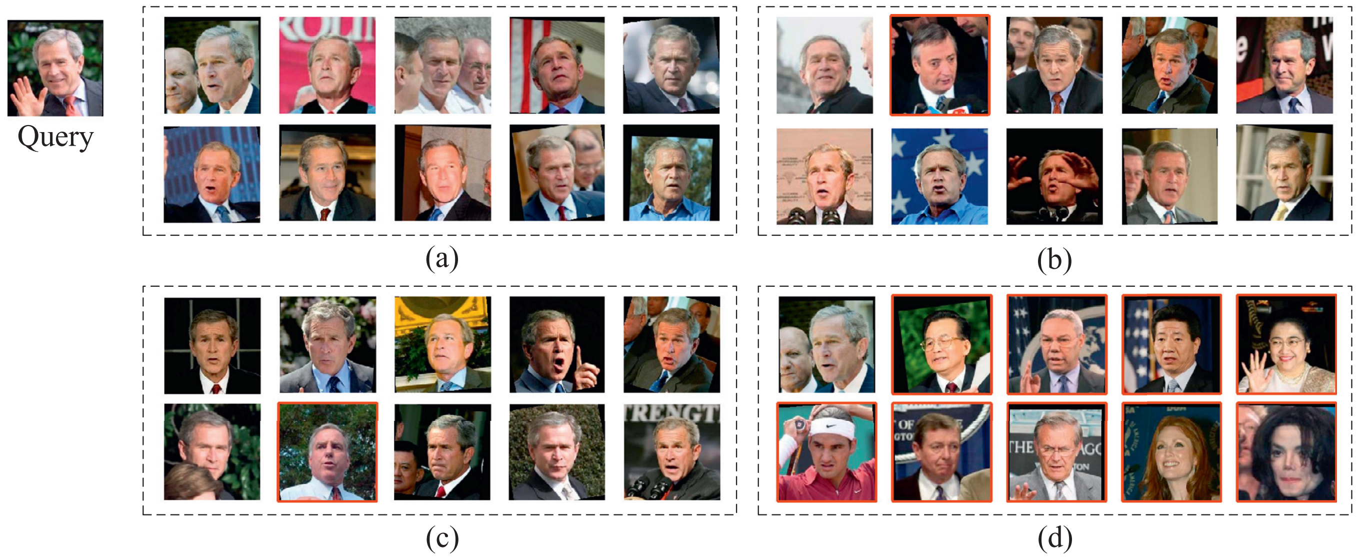


Fig. 8. Face retrieval results with 16-bit hashing codes on the LFW dataset, where the incorrect retrieval results are marked by red rectangles. (a) SHISS; (b) LFH; (c) SDH; (d) ITQ.

5. Conclusion

This paper proposes a novel supervised hashing method named "Supervised Hashing with Informative Set Selection" (SHISS). Different from the existing supervised methods that randomly select examples in each iteration, SHISS selects the most informative subset of training examples for obtaining better hashing functions. Our method is simple, efficient, and effective, and the experiments on three typical datasets demonstrate that SHISS achieves very encouraging performance as well as significantly boost the convergence speed.

Note that the proposed SHISS is a general hashing method. Therefore, in the future, we plan to apply SHISS to other identity authentication systems with different biometric characteristics such as gait, palmprint, and iris.

Acknowledgments

This research is partly supported by NSFC, China (No:61572315, 6151101179, 61602246), and Natural Science Foundation of Jiangsu Province (No. BK20171430), and National Basic Research Program, China (No. 2015CB856004).

References

[1] M. Belkin, P. Niyogi, Laplacian eigenmaps for dimensionality reduction and data representation, Neural Comput. 15 (6) (2003) 1373–1396.
[2] Y. Bengio, J. Louradour, R. Collobert, J. Weston, Curriculum learning, in: International Conference on Machine Learning, 2009, pp. 41–48.
[3] M.A. Carreira-Perpinán, R. Raziperchikolaei, Hashing with binary autoencoders, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2015, pp. 557–566.
[4] C. Ding, J. Choi, D. Tao, L.S. Davis, Multi-directional multi-level dual-cross patterns for robust face recognition, IEEE Trans. Pattern Anal. Mach. Intell. 38 (3) (2016) 518–531.
[5] C. Ding, D. Tao, Robust face recognition via multimodal deep face representation, IEEE Trans. Multimedia 17 (11) (2015) 2049–2058.
[6] C. Ding, D. Tao, Trunk-branch ensemble convolutional neural networks for video-based face recognition, IEEE Trans. Pattern Anal. Mach. Intell. (2017).
[7] J.L. Elman, Learning and development in neural networks: the importance of starting small, Cognition 48 (1) (1993) 71–99.
[8] A. Gionis, P. Indyk, R. Motwani, et al., Similarity search in high dimensions via hashing, in: VLDB, 99, 1999, pp. 518–529.
[9] C. Gong, D. Tao, W. Liu, L. Liu, J. Yang, Label propagation via teaching-to-learn and learning-to-teach., IEEE Trans. Neural Netw. Learn. Syst. (2016) 1–14.
[10] C. Gong, D. Tao, J. Yang, W. Liu, Teaching-to-learn and learning-to-teach for multi-label propagation., in: AAAI, 2016, pp. 1610–1616.
[11] Y. Gong, S. Lazebnik, Iterative quantization: a procrustean approach to learning binary codes, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, IEEE, 2011, pp. 817–824.
[12] G.B. Huang, M. Ramesh, T. Berg, E. Learned-Miller, Labeled Faces in the Wild: A Database for Studying Face Recognition in Unconstrained Environments, Technical Report, University of Massachusetts, Amherst, 2007.
[13] M.J. Huiskes, B. Thomee, M.S. Lew, New trends and ideas in visual concept detection: the mir flickr retrieval evaluation initiative, in: Proceedings of the international conference on Multimedia information retrieval, ACM, 2010, pp. 527–536.
[14] L. Jiang, D. Meng, S. Yu, Z. Lan, S. Shan, A. Hauptmann, Self-paced learning with diversity, Adv. Neural Inf. Process. Syst. (2014) 2078–2086.
[15] I. Kavati, M.V. Prasad, C. Bhagvati, A score-based indexing and retrieval technique for biometric databases, Int. J. Pattern Recognit. Artif. Intell. 31 (06) (2017) 1756009.
[16] F. Khan, B. Mutlu, X. Zhu, How do humans teach: on curriculum learning and teaching dimension, in: Advances in Neural Information Processing Systems, 2011, pp. 1449–1457.
[17] S. Korman, S. Avidan, Coherency sensitive hashing, in: Computer Vision (ICCV), 2011 IEEE International Conference on, IEEE, 2011, pp. 1607–1614.
[18] S. Korman, S. Avidan, Coherency sensitive hashing, IEEE Trans. Pattern Anal. Mach. Intell. 38 (6) (2016) 1099–1112.
[19] A. Krizhevsky, G. Hinton, Learning multiple layers of features from tiny images (2009).
[20] B. Kulis, T. Darrell, Learning to hash with binary reconstructive embeddings, in: Advances in neural information processing systems, 2009, pp. 1042–1050.
[21] G.B.H.E. Learned-Miller, Labeled Faces in the Wild: Updates and New Reporting Procedures, Technical Report, University of Massachusetts, Amherst, 2014.
[22] Y. LeCun, B. Boser, J.S. Denker, D. Henderson, R.E. Howard, W. Hubbard, L.D. Jackel, Backpropagation applied to handwritten zip code recognition, Neural Comput. 1 (4) (1989) 541–551.
[23] W.-J. Li, S. Wang, W.-C. Kang, Feature learning based deep supervised hashing with pairwise labels, arXiv preprint (2015). arXiv:1511.03855.
[24] K. Lin, H.-F. Yang, J.-H. Hsiao, C.-S. Chen, Deep learning of binary hash codes for fast image retrieval, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops, 2015, pp. 27–35.
[25] W. Liu, J. Wang, R. Ji, Y.-G. Jiang, S.-F. Chang, Supervised hashing with kernels, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, IEEE, 2012, pp. 2074–2081.
[26] W. Liu, J. Wang, S. Kumar, S.-F. Chang, Hashing with graphs, in: International Conference on Machine Learning, 2011, pp. 1–8.
[27] I. Loshchilov, F. Hutter, Online batch selection for faster training of neural networks, Mathematics (2015).
[28] A. Oliva, A. Torralba, Modeling the shape of the scene: a holistic representation of the spatial envelope, Int. J. Comput. Vis. 42 (3) (2001) 145–175.
[29] F. Pompili, N. Gillis, P.A. Absil, F. Glineur, Two algorithms for orthogonal nonnegative matrix factorization with application to clustering, Neurocomputing 141 (4) (2012) 15–25.
[30] F. Shen, C. Shen, W. Liu, H. Tao Shen, Supervised discrete hashing, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2015, pp. 37–45.
[31] F. Shen, C. Shen, Q. Shi, A. Van Den Hengel, Z. Tang, Inductive hashing on manifolds, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2013, pp. 1562–1569.

[32] A. Torralba, R. Fergus, W.T. Freeman, 80 million tiny images: a large data set for nonparametric object and scene recognition, IEEE Trans. Pattern Anal. Mach. Intell. 30 (11) (2008) 1958–1970.
[33] J. Wang, S. Kumar, S.F. Chang, Sequential projection learning for hashing with compact codes, in: International Conference on Machine Learning, 2010, pp. 1127–1134.
[34] J. Wang, S. Kumar, S.-F. Chang, Semi-supervised hashing for large-scale search, IEEE Trans. Pattern Anal. Mach. Intell. 34 (12) (2012) 2393–2406.
[35] Q. Wang, L. Si, Z. Zhang, N. Zhang, Active hashing with joint data example and tag selection, in: Proceedings of the 37th international ACM SIGIR conference on Research and development in information retrieval, ACM, 2014, pp. 405–414.
[36] Y. Weiss, A. Torralba, R. Fergus, Spectral hashing, in: D. Koller, D. Schuurmans, Y. Bengio, L. Bottou (Eds.), Advances in Neural Information Processing Systems 21, Curran Associates, Inc., 2009, pp. 1753–1760.
[37] R. Xia, Y. Pan, H. Lai, C. Liu, S. Yan, Supervised hashing for image retrieval via image representation learning., in: AAAI, 1, 2014, p. 2.
[38] D. Zhang, J. Wang, D. Cai, J. Lu, Self-taught hashing for fast similarity search, in: Proceedings of the 33rd international ACM SIGIR conference on Research and development in information retrieval, ACM, 2010, pp. 18–25.
[39] P. Zhang, W. Zhang, W.J. Li, M. Guo, Supervised hashing with latent factor models, in: Proceedings of the 37th international ACM SIGIR conference on Research and development in information retrieval, 2014, pp. 173–182.
[40] R. Zhang, L. Lin, R. Zhang, W. Zuo, L. Zhang, Bit-scalable deep hashing with regularized similarity learning for image retrieval and person re-identification, IEEE Trans. Image Process. 24 (12) (2015) 4766–4779.
[41] Y. Zhen, D.Y. Yeung, Active hashing and its application to image and text retrieval, Data Min. Knowl. Discov. 26 (2) (2013) 255–274.