SADIH: Semantic-Aware DIscrete Hashing

Zheng Zhang,[1] **Guo-sen Xie,**[2] **Yang Li,**[1] **Sheng Li,**[3] **Zi Huang**[1]
[1]The University of Queensland, Australia
[2] Inception Institute of Artificial Intelligence, UAE
[3] University of Georgia, USA
{darrenzz219, gsxiehm}@gmail.com; y.li9@uq.net.au; sheng.li@uga.edu; huang@itee.uq.edu.au

Abstract

Due to its low storage cost and fast query speed, hashing has been recognized to accomplish similarity search in large-scale multimedia retrieval applications. Particularly, supervised hashing has recently received considerable research attention by leveraging the label information to preserve the pairwise similarities of data points in the Hamming space. However, there still remain two crucial bottlenecks: 1) the learning process of the *full pairwise similarity preservation* is computationally unaffordable and unscalable to deal with big data; 2) the available category information of data are not well-explored to learn discriminative hash functions. To overcome these challenges, we propose a unified Semantic-Aware DIscrete Hashing (SADIH) framework, which aims to directly embed the transformed semantic information into the asymmetric similarity approximation and discriminative hashing function learning. Specifically, a semantic-aware latent embedding is introduced to asymmetrically preserve the full pairwise similarities while skillfully handle the cumbersome $n \times n$ pairwise similarity matrix. Meanwhile, a semantic-aware autoencoder is developed to jointly preserve the data structures in the discriminative latent semantic space and perform data reconstruction. Moreover, an efficient alternating optimization algorithm is proposed to solve the resulting discrete optimization problem. Extensive experimental results on multiple large-scale datasets demonstrate that our SADIH can clearly outperform the state-of-the-art baselines with the additional benefit of lower computational costs.

Introduction

In the big data era, recent years have witnessed the ever-growing volume of multimedia data with high dimensionality. This is made possible by the emergence of large-scale similarity measurement technique with high computational efficiency. Different from the traditional indexing technique (Lew et al. 2006), hashing yields a scalable similarity search mechanism with acceptable accuracies in the fast Hamming space (Wang et al. 2018). Technically, hashing generally compresses the high-dimensional data instances into compact binary codes (typically $\leq$ 128-dim), in which the similarity and structural information are preserved from the original data. In this paper, we will mainly focus on the learning-based hashing methods that are formulated by the data-dependent hash encoding strategy, which has shown better retrieval performance than data-independent (or learning-free) hashing schemes, such as locality-sensitive hashing (LSH) (Gionis et al. 1999) and its variants (Kulis and Grauman 2009; Jiang, Que, and Kulis 2015).

A common problem of learning-based hashing methods is to construct similarity-preserving hash functions, which generate similar binary codes for nearby data items. Many such hashing learning methods have been proposed to enable efficient similarity search and can be broadly grouped into two categories: unsupervised and supervised hashing.

Unsupervised methods typically encode samples as binary codes by exploring data distribution without label or relevances (Zhang et al. 2018b; 2018a). They learn hash codes/functions based on the semantic gap principle (Smeulders et al. 2000), *i.e.,* the difference in structures formed within the high- and low-level descriptors. Representative unsupervised hashing methods include manifold learning based hashing and quantization based hashing. Manifold learning based hashing tries to discover the neighborhood relationship of data points in the learned binary codes, such as spectral hashing (SH) (Weiss, Torralba, and Fergus 2009), multiple feature hashing (MFH) (Song et al. 2011), scalable graph hashing (SGH) (Jiang and Li 2015) and ordinal constraint hashing (OCH) (Liu et al. 2018). Quantization based hashing aims to achieve the minimal quantization error, such as iterative quantization (ITQ) (Gong et al. 2013) and quantization-based hashing (Song et al. 2018). Due to the absence of semantic label information, unsupervised hashing is usually inferior to supervised hashing, which can produce state-of-the-art retrieval results.

Supervised hashing generates discriminative and compact hash codes/functions by leveraging the supervised semantic information from data such as pairwise similarity or relevant feedback. Many supervised hashing methods have been proposed to enable efficient similarity search. Representative methods in this group include semi-supervised hashing (SSH) (Wang, Kumar, and Chang 2012), latent factor hashing (LFH) (Zhang, Zhang, and Li 2014), fast supervised hashing (FastH) (Lin et al. 2014), column sampling based discrete supervised hashing (COSDISH) (Kang, Li, and Zhou 2016), etc. It has been studied to construct encoding functions in a designed kernel space, such as binary reconstructive embedding (BRE) (Kulis and Darrell 2009),

kernel-based supervised hashing (KSH) (Liu et al. 2012), the kernel variant of ITQ (Gong et al. 2013), supervised discrete hashing (SDH) (Shen et al. 2015), SDH with relaxation (SDHR) (Gui et al. 2016; Zhang et al. 2018c) and fast SDH (FSDH) (Gui et al. 2018). The kernel based hashing methods have been shown to achieve promising performance.

To further improve the retrieval performance, many deep hashing models (Erin Liong et al. 2015; Shen et al. 2018; Li, Wang, and Kang 2016; Lai et al. 2015; Lin et al. 2015) have been introduced over the past few years, where the nonlinear feature embeddings learned by deep neural networks were typically shown to achieve higher performance than hand-crafted descriptors. As we know, semantic hashing (Salakhutdinov and Hinton 2009) is the pioneering work of using deep machine for hashing. However, these deep hashing models are complicated and need pre-training, which is inefficient in real applications. Moreover, there might be concern about the encoding time of the training and test data.

Although achieving progress, current supervised similarity-preserving hashing methods are still facing severe challenges. First, to *avoid* using the full $n \times n$ pairwise similarity matrix, these methods employ sampling strategies in the training phase to reduce the large computation and memory overhead. In such a case, they fail to capture the full structures residing on the entire data, which inevitably results in information loss and unsatisfactory performance. Their objectives would be suboptimal for realistic search tasks, and such methods become inappropriate for large-scale retrieval tasks. Second, only preserving the pairwise similarities transformed from labels clearly excludes the category information of data from the training step. In this way, these methods can not transfer the discriminative information from labels into the learned binary codes, resulting in inferior performance. Third, since the discrete optimization introduced by binary constraint leads to an NP-hard mixed integer programming problem, most of them usually solve it by relaxing the binary variables into continuous ones, followed by thresholding or quantization. However, such relaxation strategy can amplify the quantization errors, which may greatly influence the quality of the learned binary codes and degrade the performance.

To address the aforementioned problems, we propose a novel discriminative binary code learning framework, dubbed Semantic-Aware DIscrete Hashing (SADIH), for fast scalable supervised hashing. Specifically, we introduce an asymmetric similarity-preserving strategy that can preserve the discrete constraint and reduce the accumulated quantization error between binary code matrix and the well-designed latent semantic-aware embedding. During the learning step, such trick can skillfully handle the huge $n \times n$ pairwise similarity matrix, and preserve the discriminative category information into the learned binary codes. Meanwhile, we also develop a novel semantic-aware encoder-decoder paradigm to guarantee the high-quality latent embedding. In particular, an encoder projects the visual features of an image into a latent semantic space, and in turn consider the latent semantic-aware embedding as an input to a decoder which reconstructs the original visual representation. As such, our learning framework not only can effectively preserve the discriminative semantic information into the learned binary codes and hashing functions, but efficiently approximates the full pairwise similarities without information loss. Furthermore, an alternating algorithm is developed to solve the resulting problem, where each subproblem can be optimized efficiently, yielding satisfactory solutions.To sum up, the main contributions of this work are:

(1) A novel semantic-aware discrete hashing framework is proposed to simultaneously consider the full pairwise similarities ($n \times n$) and the category information into the joint learning objective. SADIH aims to generate discriminative binary codes which can successfully capture the entire pairwise similarities as well as the intrinsic correlations between visual features and semantics from different categories.

(2) We introduce a latent semantic embedding space which can reconcile the structural difference between the visual and semantic spaces, meanwhile preserve the discriminative structures in the learned binary codes.

(3) An asymmetric similarity approximation loss is developed to reduce the accumulated quantization error between the learned binary codes and the latent semantic-aware embeddings. Meanwhile, a supervised semantic-aware autoencoder is constructed to jointly perform the data structural preservation and data reconstruction. The well-designed alternating optimization algorithm with guaranteed convergence is applied to produce the high-quality hash codes.

Semantic-Aware Discrete Hashing

Basic Formulation

This work mainly focuses on supervised hashing to enable efficient semantic similarity search by Hamming ranking of compact hash codes. Suppose we have n d-dimensional data points, denoted as $\boldsymbol{X} = [\boldsymbol{x}_1, \cdots, \boldsymbol{x}_n] \in \Re^{d \times n}$, and their associated semantic labels are $\boldsymbol{Y} = [\boldsymbol{y}_1, \cdots, \boldsymbol{y}_n] \in \{0,1\}^{c \times n}$, where c is the number of classes. The i-th column of matrix $\boldsymbol{Y}$, *i.e.* $\boldsymbol{y}_i = [0, \cdots 1, \cdots, 0]^T \in \Re^c$, is the label vector of the i-th sample, and $y_{ji} = 1$ indicates $\boldsymbol{x}_i$ belongs to the j-th class. Notably, supervised hashing also contains a pairwise similarity matrix $\boldsymbol{S} \in \{-1,1\}^{n \times n}$ obtained from semantic correlations such as labels used in this paper. Specifically, $s_{ij} = 1$ means that data items i and j are semantically similar and share at least one label, while $s_{ij} = -1$ indicates items i and j are semantically dissimilar.

The goal of supervised hashing aims to learn l hashing functions to project the data X into a discriminative Hamming space, and generate a binary code matrix $\boldsymbol{B} = [\boldsymbol{b}_1, \cdots, \boldsymbol{b}_n] \in \{-1,1\}^{l \times n}$. Moreover, the learned binary codes should preserve the semantic similarities indicated in $\boldsymbol{S}$. The commonly-used objective function (Liu et al. 2012) quantizes the approximation error between the Hamming affinity and semantic similarity matrix using

$$\min_{\boldsymbol{B}} \|r\boldsymbol{S} - \boldsymbol{B}^T\boldsymbol{B}\|_F^2 \; s.t. \; \boldsymbol{B} \in \{-1,1\}^{l \times n}, \qquad (1)$$

where $\|\cdot\|_F$ is the Frobenius norm. In this model, the inner product of any two binary codes reflects the opposite of the Hamming distance, and can be used to approximate the corresponding similarity labels. Due to its effectiveness, this model has become a standard formulation for

supervised hashing learning (Zhang, Zhang, and Li 2014; Lin et al. 2014; Kang, Li, and Zhou 2016). However, there are still several deficiencies. **First**, such a symmetric binary affinity form is limited in matching the real-valued ground truth. Importantly, the optimization on symmetric discrete constraint is time-consuming, which makes it hard to adapt for large-scale datasets (Neyshabur et al. 2013). **Second**, owing to its computation and memory prohibition, the full similarity matrix $\boldsymbol{S}$ is usually avoided using in the training step. An alternative strategy is to sample a small subset for training, which inevitably causes information loss and suboptimal results. **Third**, directly transforming labels into pairwise similarities loses the category information of training data, which can not preserve the discriminative characteristics into the learned binary codes. **Finally**, most methods solve the discrete optimization problem by relaxing the discrete constraint by omitting the sign function. However, the approximate solution is obviously suboptimal and often generates low-quality hashing codes.

Therefore, we present an efficient **Semantic-Aware DIscrete Hashing (SADIH)** framework to alleviate the above limitations. In the method, the asymmetric hamming affinity approximation, latent semantic embedding and encoder-decoder paradigm are simultaneously considered to guarantee discriminative binary codes and hashing functions.

Objective Function

Asymmetric Similarity Approximation Loss To fully explore the entire similarities on n available points, we introduce a simple but effective semantic-aware constraint, *i.e.,* $\mathcal{V} = \boldsymbol{W}^\top \boldsymbol{Y}$ where $\boldsymbol{W} \in \Re^{c\times l}$, to asymmetrically approximate the ground-truth affinity. Meanwhile, the label information are embedded into the latent semantic embedding $\mathcal{V}$. Particularly, the matrix $\boldsymbol{W}^\top$ can be viewed as the category-level basis matrix of the latent semantic features $\mathcal{V}$, because $\boldsymbol{v}_i = \boldsymbol{W}^\top \boldsymbol{y}_i$, where each item in $\boldsymbol{v}_i$ contains the category partition factor. Moreover, using the real-valued embeddings can produce more accurate approximation of similarity, and reduce the accumulated quantization error (Dong, Charikar, and Li 2008; Luo, Wu, and Xu 2018). Based on the asymmetric hashing learning (Shrivastava and Li 2014), we replace one of the binary codes $\boldsymbol{B}$ in (1), and consider its robust model

$$\begin{aligned} &\min_{\boldsymbol{B},\boldsymbol{W}} \|l\boldsymbol{S} - \mathcal{V}^T\boldsymbol{B}\|_{21} \\ &s.t.\ \mathcal{V} = \boldsymbol{W}^T\boldsymbol{Y}, \boldsymbol{B} \in \{-1,1\}^{l\times n}, \end{aligned} \tag{2}$$

which $\|\boldsymbol{A}\|_{21} = \sum_{i=1}^{n} \|\boldsymbol{a}^i\|_2$ denotes the l_{21}-norm of matrix $\boldsymbol{A}$ and $\boldsymbol{a}^i$ is the i-th row of matrix $\boldsymbol{A}$. The l_{21}-norm is robust to noise or outliers based on the rotation-invariance property. It is noteworthy that this simple constraint can enable the model to effectively exploit all the n data points for training (shown in optimization) without any sampling. Moreover, it also can more precisely measure the quantization between the given similarities $\boldsymbol{S}$ and the learned asymmetric affinity.

Semantic-Aware Autoencoder The discriminative binary codes for training data can be learned based on model (2), but there still remain two concerns. On one hand, the latent semantic embedding only leverages the label information, while the inherent characteristics embedded in the training data are not well-explored to capture the instance-level features. On the other hand, (2) can not be generalized to the out-of-sample cases for efficient query generation. To this end, we formulate the linear semantic-aware autoencoder scheme, which optimizes against the following objective:

$$\begin{aligned} &\min_{\{\boldsymbol{P}_i,\boldsymbol{c}_i\}_{i=1}^2,\mathcal{V},\boldsymbol{W}} \|\boldsymbol{X} - (\boldsymbol{P}_2(\boldsymbol{P}_1\boldsymbol{X} + \boldsymbol{c}_1\boldsymbol{1}^T) + \boldsymbol{c}_2\boldsymbol{1}^T)\|_F^2 \\ &+ \gamma\mathcal{R}(\boldsymbol{P}_2,\boldsymbol{P}_1)\ s.t.\ \mathcal{V} = \boldsymbol{W}^T\boldsymbol{Y}, \mathcal{V} = \boldsymbol{P}_1\boldsymbol{X} + \boldsymbol{c}_1\boldsymbol{1}^T, \end{aligned} \tag{3}$$

where $\boldsymbol{P}_1 \in \Re^{l\times d}$ and $\boldsymbol{P}_2 \in \Re^{d\times l}$ are the encoding and decoding matrices, $\mathcal{R}(\cdot) = \|\cdot\|_F^2$ is the regularization term to avoid overfitting, $\boldsymbol{c}_1$ and $\boldsymbol{c}_2$ are the biased vectors, and γ is a weighting parameter. It is clear that *this model can make use of semantic attributes in $\mathcal{V}$ as an intermediate level clue to associate low-level visual features with high-level semantic information*. However, when we project the visual d-dim features into the lower l-dim (typically $l \ll d$) semantic space, this may encounter the imbalanced projection problem, *i.e.* the variances of the projected dimensions vary severely (Wang, Kumar, and Chang 2012). To this end, inspired by ITQ (Gong et al. 2013), we may change the coordinates of the whole feature space through an adjustment rotation. For eliminating the bias variables, we reformulate the above problem into a relaxed optimization with an orthogonal transformation:

$$\begin{aligned} &\min_{\boldsymbol{P}_1,\boldsymbol{P}_2,\mathcal{V},\boldsymbol{W}} \|\boldsymbol{X} - \boldsymbol{P}_2\mathcal{V}\|_F^2 + \beta\|\mathcal{V} - \boldsymbol{P}_1\boldsymbol{X}\|_F^2 + \gamma\mathcal{R}(\boldsymbol{P}_2) \\ &s.t.\ \mathcal{V} = \boldsymbol{W}^T\boldsymbol{Y}, \boldsymbol{P}_1^T\boldsymbol{P}_1 = \boldsymbol{I}. \end{aligned} \tag{4}$$

From Eqn. (4), we can see that the preferred latent attributes satisfy $\mathcal{V} = \boldsymbol{P}_1\boldsymbol{X}$ with minimum reconstruction error $\boldsymbol{X} = \boldsymbol{P}_2\mathcal{V}$. Given the orthogonal transformation, the overall variance can be effectively diffused into all the learned dimensions through the adjustment rotation, and the underlying characteristics hidden in data $\boldsymbol{X}$ are uncovered and transferred into the semantic embeddings. Importantly, the encoder can be used as a linear hash function for new queries.

Joint Objective Function To preserve the interconnection between the semantic-aware similarity approximation and preferable latent semantic space construction, SADIH combines the asymmetric similarity approximation loss given in (2) and semantic-aware autoencoder scheme given in (4) into one unified learning framework. Such a learning framework can minimize the intractable full pairwise similarity-preserving error, meanwhile interactively enhances the qualities of the learned binary codes and discriminative latent semantic-aware embeddings:

$$\begin{aligned} &\min_{\boldsymbol{B},\boldsymbol{W},\boldsymbol{P}_1,\boldsymbol{P}_2,\mathcal{V}} \|l\boldsymbol{S} - \mathcal{V}^T\boldsymbol{B}\|_{21} + \alpha\|\boldsymbol{X} - \boldsymbol{P}_2\mathcal{V}\|_F^2 \\ &\qquad + \beta\|\mathcal{V} - \boldsymbol{P}_1\boldsymbol{X}\|_F^2 + \gamma\mathcal{R}(\boldsymbol{P}_2) \\ &s.t.\ \boldsymbol{B} \in \{-1,1\}^{l\times n}, \mathcal{V} = \boldsymbol{W}^T\boldsymbol{Y}, \boldsymbol{P}_1^T\boldsymbol{P}_1 = \boldsymbol{I}, \end{aligned} \tag{5}$$

where α is a weighting parameter. Furthermore, it is clear that the first term makes sure the asymmetric correlations between the discriminative binary codes $\boldsymbol{B}$ and latent semantic representations $\mathcal{V}$. Therefore, the encoding matrix

$\boldsymbol{P}_1$ can capture the discriminative information embedded in the latent semantic space, and the hashing codes for out-of-sample $\boldsymbol{x}_t$ can be directly generated by $\boldsymbol{b} = sgn(\boldsymbol{P}_1\boldsymbol{x}_t)$, where $sgn(\cdot)$ denotes the element-wise sign function.

Optimization

The key of our optimization is to learn discriminative binary codes $\boldsymbol{B}$ and find a proper latent-embedding space $\mathcal{V}$ with the data structural preservation $\boldsymbol{P}_1$ and data reconstruction $\boldsymbol{P}_2$. However, problem (5) is non-convex to all variables, and involves a discrete constraint, which leads to an NP-hard problem. Thus, we propose an alternating optimization algorithm to obtain a local minima.

We first rewrite problem (5) as an equivalent form:

$$\begin{aligned}\min_{\boldsymbol{B},\boldsymbol{W},\boldsymbol{P}_1,\boldsymbol{P}_2} &\|l\boldsymbol{S}-\left(\boldsymbol{W}^T\boldsymbol{Y}\right)^T\boldsymbol{B}\|_{21}+\alpha\|\boldsymbol{X}-\boldsymbol{P}_2\boldsymbol{W}^T\boldsymbol{Y}\|_F^2\\ &+\beta\|\boldsymbol{W}^T\boldsymbol{Y}-\boldsymbol{P}_1\boldsymbol{X}\|_F^2+\gamma\mathcal{R}(\boldsymbol{P}_2,\boldsymbol{W}^T\boldsymbol{Y})\\ s.t.\ &\boldsymbol{B}\in\{-1,1\}^{l\times n},\boldsymbol{P}_1^T\boldsymbol{P}_1=\boldsymbol{I}.\end{aligned} \tag{6}$$

Next, we iteratively update each variable with an alternative manner, *i.e.*, updating one when fixing others.

$\boldsymbol{B}$-Step: When fixing $\boldsymbol{W}$, $\boldsymbol{P}_1$ and $\boldsymbol{P}_2$, we can update $\boldsymbol{B}$ by using two discrete learning strategies. The first learning scheme optimizes $\boldsymbol{B}$ with a constant-approximation solution, inspired by (Kang, Li, and Zhou 2016). Specifically, we push the l_{21}-norm loss to a more strict l_1-norm loss, *i.e.,*

$$\min_{\boldsymbol{B}}\|l\boldsymbol{S}-\left(\boldsymbol{W}^T\boldsymbol{Y}\right)^T\boldsymbol{B}\|_1\ s.t.\ \boldsymbol{B}\in\{-1,1\}^{l\times n}. \tag{7}$$

where $\|\cdot\|_1$ denotes the l_1-norm. Problem (7) reaches its minimum when $\boldsymbol{B} = sgn(\boldsymbol{W}^T\boldsymbol{Q})$, where $\boldsymbol{Q} = l\boldsymbol{Y}\boldsymbol{S}$ can be calculated beforehand. We denote this as ***SADIH-L1***.

Alternatively, we can directly optimize the l_{21}-norm loss with an equivalent transformation. We first define $\boldsymbol{D} \in \Re^{n\times n}$ as a diagonal matrix, the i-th diagonal element of which is defined as $\boldsymbol{d}_{ii} = 1/2\|\boldsymbol{u}^i\|$, where $\boldsymbol{u}^i$ is the i-th row of $\left(l\boldsymbol{S}-\boldsymbol{Y}^T\boldsymbol{W}\boldsymbol{B}\right)$. In this way, we need to solve

$$\begin{aligned}&\min_{\boldsymbol{B}} Tr\left(\left(l\boldsymbol{S}-\boldsymbol{Y}^T\boldsymbol{W}\boldsymbol{B}\right)^T\boldsymbol{D}\left(l\boldsymbol{S}-\boldsymbol{Y}^T\boldsymbol{W}\boldsymbol{B}\right)\right)\\ &s.t.\ \boldsymbol{B}\in\{-1,1\}^{l\times n},\end{aligned} \tag{8}$$

which can be equivalently rewritten as

$$\begin{aligned}&\min_{\boldsymbol{B}} Tr\left(\boldsymbol{R}^T\boldsymbol{B}\boldsymbol{D}\boldsymbol{B}^T\boldsymbol{R}\right)-2tr\left(\boldsymbol{B}^T\boldsymbol{M}\right)\\ &s.t.\ \boldsymbol{B}\in\{-1,1\}^{l\times n},\end{aligned} \tag{9}$$

where $\boldsymbol{R} = \boldsymbol{W}^T\boldsymbol{Y}$, $\boldsymbol{M} = \boldsymbol{W}^T\boldsymbol{D}\boldsymbol{Q}$, and Tr$(\cdot)$ is the trace norm. For this binary quadratic program problem, we employ the discrete cyclic coordinate descent (DCC) method (Shen et al. 2015) to sequentially learn each row of $\boldsymbol{B}$ while fixing other rows. Let $\boldsymbol{b}^\top$ be the k-th row of $\boldsymbol{B}$, $k = 1,\cdots,l$, and $\bar{\boldsymbol{B}}$ be the matrix of $\boldsymbol{B}$ excluding $\boldsymbol{b}$. Similarly, let $\boldsymbol{r}^T$ and $\boldsymbol{q}^T$ be the k-th row of $\boldsymbol{R}$ and $\boldsymbol{M}$, respectively. $\bar{\boldsymbol{R}}$ and $\bar{\boldsymbol{M}}$ are the matrix of $\boldsymbol{R}$ excluding $\boldsymbol{r}$ and the matrix of $\boldsymbol{M}$ excluding $\boldsymbol{q}$, respectively. Then, we have

$$\begin{aligned}&Tr\left(\boldsymbol{R}^\top\boldsymbol{B}\boldsymbol{D}\boldsymbol{B}^\top\boldsymbol{R}\right)\\ &=Tr\left(\left(\bar{\boldsymbol{B}}^\top\bar{\boldsymbol{R}}+\boldsymbol{b}\boldsymbol{r}^\top\right)\boldsymbol{D}\left(\bar{\boldsymbol{R}}^\top\bar{\boldsymbol{B}}+\boldsymbol{r}\boldsymbol{b}^\top\right)\right)\\ &=2Tr\left(\boldsymbol{b}\boldsymbol{r}^\top\boldsymbol{D}\bar{\boldsymbol{R}}^\top\bar{\boldsymbol{B}}\right)+Tr\left(\boldsymbol{b}\boldsymbol{r}^\top\boldsymbol{D}\boldsymbol{r}\boldsymbol{b}^\top\right)+const.\end{aligned} \tag{10}$$

Since $Tr(\boldsymbol{b}\boldsymbol{r}^\top\boldsymbol{D}\boldsymbol{r}\boldsymbol{b}^\top) = Tr(\boldsymbol{r}^\top\boldsymbol{D}\boldsymbol{r}\boldsymbol{b}^\top\boldsymbol{b}) = n\sum_i \boldsymbol{d}_{ii}\boldsymbol{r}^\top\boldsymbol{r}$, this term is a constant *w.r.t.* $\boldsymbol{b}$.

Similarly, $tr\left(\boldsymbol{B}^T\boldsymbol{M}\right) = \begin{bmatrix}\boldsymbol{b}^\top\boldsymbol{q} & \boldsymbol{b}^\top\bar{\boldsymbol{M}}\\ \bar{\boldsymbol{B}}^\top\boldsymbol{q} & \bar{\boldsymbol{B}}^\top\bar{\boldsymbol{M}}\end{bmatrix}$, and then

$$tr\left(\boldsymbol{B}^T\boldsymbol{M}\right) = \boldsymbol{q}^\top\boldsymbol{b}+const. \tag{11}$$

Therefore, Eqn. (9) can be reformulated as

$$\begin{aligned}&\min_{\boldsymbol{b}} Tr\left(\left(\boldsymbol{r}^\top\boldsymbol{D}\bar{\boldsymbol{R}}^\top\bar{\boldsymbol{B}}-\boldsymbol{q}^\top\right)\boldsymbol{b}\right)\\ &s.t.\ \boldsymbol{b}\in\{-1,1\}^l,\end{aligned} \tag{12}$$

which has a closed-form solution:

$$\boldsymbol{b} = sgn\left(\boldsymbol{q}-\bar{\boldsymbol{B}}^\top\bar{\boldsymbol{R}}\boldsymbol{D}\boldsymbol{r}\right). \tag{13}$$

We can see that each bit $\boldsymbol{b}$ is calculated based on the pre-learned (l-1) bits $\bar{\boldsymbol{B}}$. We iteratively update each bit until it converges to a set of optimal codes $\boldsymbol{B}$.

$\boldsymbol{W}$-Step: When fixing $\boldsymbol{B}$, $\boldsymbol{P}_1$ and $\boldsymbol{P}_2$, the objective function (6) *w.r.t.* $\boldsymbol{W}$ is degenerated to

$$\begin{aligned}&\min_{\boldsymbol{W}} Tr\left(\left(l\boldsymbol{S}-\boldsymbol{Y}^T\boldsymbol{W}\boldsymbol{B}\right)^T\boldsymbol{D}\left(l\boldsymbol{S}-\boldsymbol{Y}^T\boldsymbol{W}\boldsymbol{B}\right)\right)\\ &+\alpha\|\boldsymbol{X}-\boldsymbol{P}_2\boldsymbol{W}^T\boldsymbol{Y}\|_F^2+\beta\|\boldsymbol{W}^T\boldsymbol{Y}-\boldsymbol{P}_1\boldsymbol{X}\|_F^2+\gamma\|\boldsymbol{W}^T\boldsymbol{Y}\|_F^2.\end{aligned} \tag{14}$$

The closed-form solution of $\boldsymbol{W}$ is given by setting the derivation of (14) to zero, *i.e.,*

$$\begin{aligned}\boldsymbol{W} = \boldsymbol{L}(&\boldsymbol{Q}\boldsymbol{D}\boldsymbol{B}^\top+\boldsymbol{Y}\boldsymbol{X}^\top(\alpha\boldsymbol{P}_2+\beta\boldsymbol{P}_1))\\ &(\boldsymbol{B}\boldsymbol{D}\boldsymbol{B}^\top+\alpha\boldsymbol{P}_2\boldsymbol{P}_2^T+(\beta+\gamma)\boldsymbol{I})^{-1},\end{aligned} \tag{15}$$

where $\boldsymbol{L} = (\boldsymbol{Y}\boldsymbol{Y}^\top)^{-1}$ can be calculated beforehand.

$\boldsymbol{F}$-Step: When fixing $\boldsymbol{B}$, $\boldsymbol{W}$ and $\boldsymbol{P}_2$, problem (6) *w.r.t.* $\boldsymbol{P}_1$ becomes

$$\min_{\boldsymbol{P}_1}\|\boldsymbol{W}^T\boldsymbol{Y}-\boldsymbol{P}_1\boldsymbol{X}\|_F^2\ s.t.\ \boldsymbol{P}_1^T\boldsymbol{P}_1=\boldsymbol{I}, \tag{16}$$

which can be solved by the following lemma.

Lemma 1. *$\boldsymbol{P}_1 = \boldsymbol{U}\boldsymbol{V}^T$ is the optimal solution to the problem in Eqn. (16), where $\boldsymbol{U}$ and $\boldsymbol{V}$ are the left and right singular matrices of the compact Singular Value Decomposition (SVD) on ($\boldsymbol{X}\boldsymbol{Y}^\top\boldsymbol{W}$).*

$\boldsymbol{P}$-Step: Similarly, when fixing other variables, problem (6) *w.r.t.* $\boldsymbol{P}_2$ can be re-written as:

$$\min_{\boldsymbol{P}_2}\alpha\|\boldsymbol{X}-\boldsymbol{P}_2\boldsymbol{W}^T\boldsymbol{Y}\|_F^2+\gamma\|\boldsymbol{P}_2\|_F^2. \tag{17}$$

The minimal $\boldsymbol{P}_2$ can be obtained by setting the partial derivative of Eqn. (17) to zero, and we have

$$\boldsymbol{P}_2 = \boldsymbol{T}\boldsymbol{X}\boldsymbol{Y}^T\boldsymbol{W}, \tag{18}$$

where $T = (\alpha\boldsymbol{X}\boldsymbol{X}^\top+\gamma I)^{-1}$ can be computed beforehand.

The proposed optimization iteratively updates four variables until satisfying the convergence criteria. The convergence of the proposed optimization algorithm is guaranteed by the following theorem.

Theorem 1. *The alternating optimization steps of our method will monotonously decrease the value of the objective function until it converges to a local optima.*

Table 1: The averaged retrieval result comparison (MAP score and precision of top 100 samples) and computation time efficiency (in seconds) on **CIFAR-10** using 512-dimensional GIST features, and both **SUN397** and **ImageNet** datasets using 4096-dim deep CNN features from VGG19 $fc7$. The best performances have been displayed in boldface.

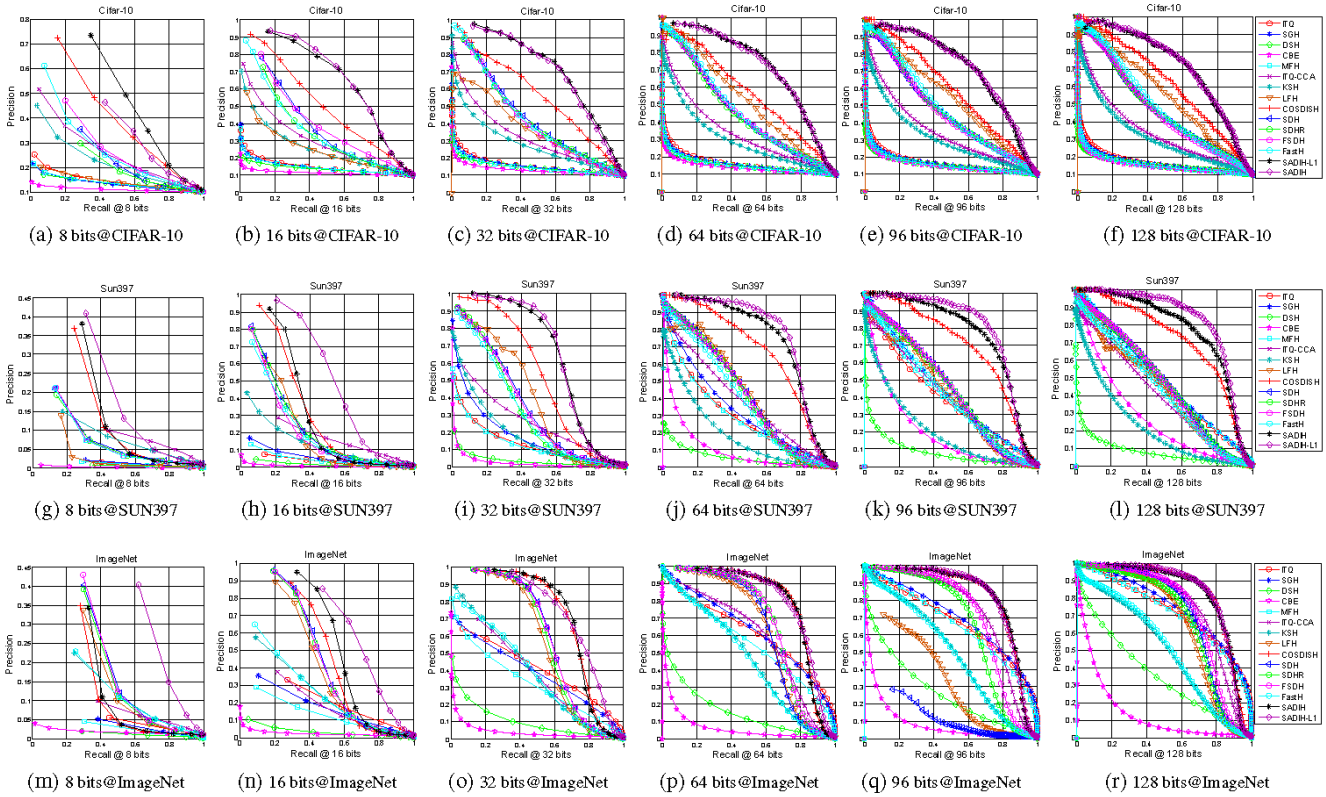
Methods	**CIFAR-10**																	
	MAP Score						**Precision@top100**						**Computation Time (in second)**					
# bits	8-bits	16-bits	32-bits	64-bits	96-bits	128-bits	8-bits	16-bits	32-bits	64-bits	96-bits	128-bits	8-bits	16-bits	32-bits	64-bits	96-bits	128-bits
ITQ	0.1528	0.1552	0.1642	0.1700	0.1719	0.1757	0.2012	0.2341	0.2798	0.3035	0.3146	0.3295	0.98	1.40	2.49	4.35	7.01	10.07
SGH	0.1411	0.1522	0.1643	0.1711	0.1759	0.1781	0.1995	0.2487	0.2866	0.3085	0.3280	0.3329	4.47	6.27	8.30	14.62	18.71	25.16
CBE	0.1134	0.1168	0.1335	0.1454	0.1552	0.1564	0.1370	0.1545	0.1927	0.2449	0.2706	0.2791	29.80	29.92	29.90	29.30	29.53	29.53
DSH	0.1441	0.1437	0.1548	0.1589	0.1622	0.1651	0.1649	0.1986	0.2358	0.2498	0.2680	0.2738	0.35	0.39	0.49	0.81	1.03	1.36
MFH	0.1428	0.1483	0.1543	0.1605	0.1649	0.1638	0.1908	0.2410	0.2645	0.2944	0.3146	0.3124	25.33	26.44	35.30	37.94	49.19	55.42
OCH	0.0744	0.1395	0.2299	0.3048	0.3459	0.3720	0.1530	0.3276	0.4895	0.5994	0.6489	0.6765	9.29	9.25	8.97	13.94	14.53	9.76
ITQ-CCA	0.2736	0.3157	0.3361	0.3519	0.3588	0.3552	0.3815	0.4205	0.4417	0.4574	0.4650	0.4634	1.17	2.14	3.36	6.88	11.25	16.62
KSH	0.2477	0.2786	0.3044	0.3202	0.3330	0.3349	0.2511	0.2849	0.3068	0.3164	0.3283	0.3298	38.25	77.04	171.54	354.14	501.57	729.75
LFH	0.2845	0.3849	0.5013	0.5891	0.6144	0.6109	0.2163	0.2889	0.4126	0.5039	0.5353	0.5279	4.13	7.22	12.07	18.25	29.53	40.03
COSDISH	0.4990	0.5573	0.6263	0.6155	0.6399	0.6551	0.4347	0.4818	0.5483	0.5284	0.5591	0.5728	5.65	14.78	34.84	128.13	302.06	541.51
SDH	0.2855	0.3956	0.4478	0.4610	0.4735	0.4789	0.3633	0.4920	0.5298	0.5419	0.5527	0.5543	29.40	70.30	142.73	278.70	537.74	874.45
SDHR	0.2547	0.3782	0.4422	0.4583	0.4727	0.4782	0.3221	0.4783	0.5306	0.5478	0.5503	0.5561	37.08	41.69	54.94	115.12	240.87	278.52
FSDH	0.3211	0.4196	0.4285	0.4629	0.4703	0.4708	0.4170	0.5046	0.5231	0.5399	0.5433	0.5449	7.58	11.45	11.75	8.28	14.00	15.85
FastH	0.2965	0.3679	0.4358	0.4715	0.4917	0.5018	0.3582	0.4414	0.5107	0.5528	0.5751	0.5875	95.51	219.51	302.46	596.94	1013.51	834.36
SADIH	0.4647	0.6992	0.7263	0.7271	**0.7367**	0.7319	0.4204	0.6404	**0.6682**	0.6609	**0.6703**	0.6666	1.37	4.16	19.88	65.30	158.48	315.40
SADIH-L1	**0.6131**	**0.6994**	**0.7277**	**0.7298**	0.7289	**0.7348**	**0.5487**	**0.6472**	0.6647	**0.6673**	0.6621	**0.6697**	0.36	0.41	0.47	0.70	0.79	0.97
Methods	**SUN397**																	
	MAP Score						**Precision@top100**						**Computation Time (in second)**					
# bits	8-bits	16-bits	32-bits	64-bits	96-bits	128-bits	8-bits	16-bits	32-bits	64-bits	96-bits	128-bits	8-bits	16-bits	32-bits	64-bits	96-bits	128-bits
ITQ	0.0138	0.0449	0.1139	0.2304	0.3166	0.3803	0.0177	0.0579	0.1199	0.1955	0.2377	0.2658	6.26	7.04	9.34	13.65	18.32	23.94
SGH	0.0149	0.0516	0.1229	0.2379	0.3164	0.3830	0.0197	0.0726	0.1408	0.2159	0.2619	0.2903	15.23	17.46	21.83	30.41	38.61	46.81
DSH	0.0069	0.0216	0.0480	0.0878	0.1231	0.1612	0.0090	0.0358	0.0667	0.1081	0.1368	0.1615	1.43	1.62	2.16	3.14	4.33	5.51
CBE	0.0035	0.0094	0.0266	0.0814	0.1480	0.2054	0.0058	0.0183	0.0468	0.1063	0.1597	0.1971	116.35	112.35	113.56	113.35	112.76	112.84
MFH	0.0136	0.0464	0.1135	0.2255	0.3116	0.3804	0.0199	0.0664	0.1317	0.2093	0.2556	0.2874	181.95	178.09	175.22	202.37	227.33	246.25
OCH	0.0519	0.1770	0.2302	0.3291	0.3869	0.3655	0.0418	0.1481	0.1978	0.2960	0.3529	0.3233	58.72	50.17	55.71	57.07	57.58	60.73
ITQ-CCA	0.1426	0.2130	0.2725	0.3958	0.4201	0.4177	0.0659	0.1833	0.3092	0.4926	0.5498	0.5567	14.00	14.96	21.44	37.70	28.45	63.39
KSH	**0.3400**	0.4072	0.4361	0.4276	0.4497	0.4578	0.0605	0.0717	0.0759	0.0745	0.0776	0.0785	58.75	203.49	400.48	770.64	1159.23	1747.01
LFH	0.0530	0.1536	0.2814	0.3533	0.3674	0.4013	0.0454	0.1278	0.2553	0.3183	0.3312	0.3600	58.93	54.11	68.85	92.22	140.28	143.53
COSDISH	0.2135	0.3341	0.5143	0.6832	0.7160	0.7365	0.0796	0.2141	0.4081	0.6096	0.6527	0.6755	33.93	57.69	92.10	258.65	556.82	1026.28
SDH	0.1211	0.2379	0.3415	0.4225	0.4585	0.4867	0.1024	0.3531	0.5070	0.5990	0.6395	0.6672	188.58	191.78	450.55	1014.60	1839.20	1958.06
SDHR	0.1109	0.2249	0.3140	0.4152	0.4688	0.4842	0.0797	0.3212	0.4821	0.6049	0.6489	0.6614	933.43	958.68	1150.00	1692.07	1736.54	1863.96
FSDH	0.1064	0.2112	0.3365	0.4245	0.4604	0.4775	0.0615	0.3224	0.5020	0.6046	0.6451	0.6557	91.21	92.37	91.34	97.81	105.39	116.13
FastH	0.1061	0.2027	0.3070	0.3865	0.4195	0.4386	0.0890	0.2929	0.4676	0.5670	0.5935	0.6203	103.02	198.64	448.35	886.73	785.03	1667.14
SADIH	0.1588	**0.5599**	0.5909	0.6798	0.7332	0.7655	0.0972	**0.5411**	0.5578	0.6411	0.6911	0.7156	12.68	20.78	52.64	186.77	398.34	695.19
SADIH-L1	0.2938	0.5491	**0.6597**	**0.7576**	**0.7762**	**0.8077**	**0.1467**	0.4472	**0.5956**	**0.7050**	**0.7250**	**0.7572**	9.72	10.21	10.49	11.18	11.62	11.79
Methods	**ImageNet**																	
	MAP Score						**Precision@top100**						**Computation Time (in second)**					
# bits	8-bits	16-bits	32-bits	64-bits	96-bits	128-bits	8-bits	16-bits	32-bits	64-bits	96-bits	128-bits	8-bits	16-bits	32-bits	64-bits	96-bits	128-bits
ITQ	0.0442	0.1379	0.2626	0.4194	0.4991	0.5531	0.0543	0.1665	0.2816	0.3817	0.4190	0.4475	38.73	40.45	42.87	50.23	59.28	68.59
SGH	0.0386	0.1246	0.2544	0.4158	0.4952	0.5569	0.0459	0.1509	0.2493	0.3381	0.3728	0.3957	38.28	45.24	49.80	57.90	67.90	86.43
DSH	0.0206	0.0504	0.1149	0.1854	0.2886	0.3295	0.0265	0.0755	0.1485	0.2062	0.2735	0.2966	4.65	5.83	6.91	9.59	13.15	18.02
CBE	0.0240	0.0311	0.0554	0.0821	0.1078	0.1573	0.0346	0.0744	0.1672	0.2441	0.3127	0.4098	287.12	254.74	251.37	243.71	245.01	270.34
MFH	0.0340	0.1118	0.2368	0.3937	0.4908	0.5570	0.0446	0.1347	0.2376	0.3274	0.3698	0.3958	616.86	750.02	799.93	808.44	882.59	900.17
OCH	0.1302	0.2720	0.4187	0.5248	0.5705	0.5969	0.2263	0.3728	0.4711	0.5665	0.6976	0.7141	33.27	35.07	36.89	36.12	36.96	37.01
ITQ-CCA	0.1616	0.2735	0.4090	0.6017	0.7278	0.7441	0.0959	0.2627	0.4428	0.6518	0.7671	0.7773	66.18	65.77	65.63	75.71	81.99	110.89
KSH	0.1549	0.2902	0.4115	0.4870	0.5141	0.5282	0.1478	0.2257	0.2757	0.3079	0.3201	0.3251	136.77	237.76	453.38	834.08	1328.50	1565.16
LFH	0.2512	0.4380	0.5576	0.6347	0.4972	0.6604	0.1638	0.5186	0.6519	0.7138	0.6044	0.7337	100.45	105.73	128.35	184.69	219.43	168.22
COSDISH	0.2392	0.5298	0.7304	0.7937	0.8038	0.8092	0.0996	0.4663	0.6943	0.7640	0.7670	0.7740	39.85	61.05	106.87	425.75	848.97	1271.77
SDH	0.2684	0.4631	0.5975	0.6488	0.3597	0.7040	0.1662	0.5377	0.6752	0.7235	0.4397	0.7550	182.21	187.85	210.91	614.88	936.21	1490.15
SDHR	0.2703	0.4637	0.5843	0.6749	0.6653	0.6827	0.1664	0.5417	0.6694	0.7373	0.7404	0.7494	249.32	258.10	281.94	705.89	1138.48	1645.44
FSDH	0.2818	0.4650	0.5987	0.6768	0.7063	0.7187	0.1879	0.5414	0.6775	0.7382	0.7582	0.7631	136.65	149.12	137.81	188.14	192.19	196.87
FastH	0.1613	0.2983	0.4145	0.4803	0.5119	0.5306	0.1502	0.2251	0.2776	0.3045	0.3176	0.3257	116.31	191.69	327.66	798.27	991.33	1369.75
SADIH	0.2587	0.5790	0.7303	0.7964	**0.8112**	0.8082	0.0817	0.5172	0.7014	0.7660	**0.7770**	0.7710	7.70	17.21	53.38	188.56	410.69	732.02
SADIH-L1	**0.5169**	**0.7057**	**0.7767**	**0.8037**	0.8100	**0.8165**	**0.4254**	**0.6554**	**0.7360**	**0.7664**	0.7745	**0.7848**	5.75	5.69	6.07	6.02	6.63	8.03

In experiments, we found that our algorithm usually can efficiently converge within $t = 5 - 8$ iterations. The main computational complexity of our algorithm comes from calculating $\boldsymbol{B}$ of each iteration in $\mathcal{O}(lcn)$ for SADIH and $\mathcal{O}(nc)$ for SADIH-L1. Additionally, during optimization, the maximum complexity of other steps is $\mathcal{O}(ndc)$ due to the property of matrix product and inversion, which is very efficient in practice. In general, the computational complexities of the proposed optimization algorithm on SADIH learning is linear to the number of samples $\mathcal{O}(n)$.

Experiments

In this section, we conduct extensive evaluations of our discriminative hashing methods on real-world datasets. All the experiments were implemented on Matlab 2013a on a standard Window PC with 64 GB RAM.

Datasets: We evaluate the proposed SADIH and SADIH-L1 on three publicly available benchmark databases: **CIFAR-10** (Krizhevsky and Hinton 2009), **Sun397** (Xiao et al. 2010), and **ImageNet** (Deng et al. 2009). *Cifar-10* is a labeled subset of 80M tiny images, containing $60,000$ tiny color images in 10 classes of objects with $6,000$ images per class. Each image with unique semantic label is represented by 512-dim GIST features on the 32×32 pixels. Moreover, we employ the nonlinear anchor features to improve the interpretation of features as in (Shen et al. 2015). *SUN-397* includes around $108k$ images pictured under 397 different scene conditions. Each category has at least 100 images. Each image is represented by $4,096$-dim deep cnn features from VGG19 $fc7$ (Simonyan and Zisserman 2014). *ImageNet* from ILSVRC 2012 has more than 1.2 million images from $1,000$ categories. In the experiments, we constructed a retrieval image database including $128K$ images selected from the first 100 largest classes from the given training data.



(a) 8 bits@CIFAR-10 (b) 16 bits@CIFAR-10 (c) 32 bits@CIFAR-10 (d) 64 bits@CIFAR-10 (e) 96 bits@CIFAR-10 (f) 128 bits@CIFAR-10

(g) 8 bits@SUN397 (h) 16 bits@SUN397 (i) 32 bits@SUN397 (j) 64 bits@SUN397 (k) 96 bits@SUN397 (l) 128 bits@SUN397

(m) 8 bits@ImageNet (n) 16 bits@ImageNet (o) 32 bits@ImageNet (p) 64 bits@ImageNet (q) 96 bits@ImageNet (r) 128 bits@ImageNet

Figure 1: Precision-Recall curves of different methods with different code lengths on CIFAR-10, SUN397 and ImageNet.

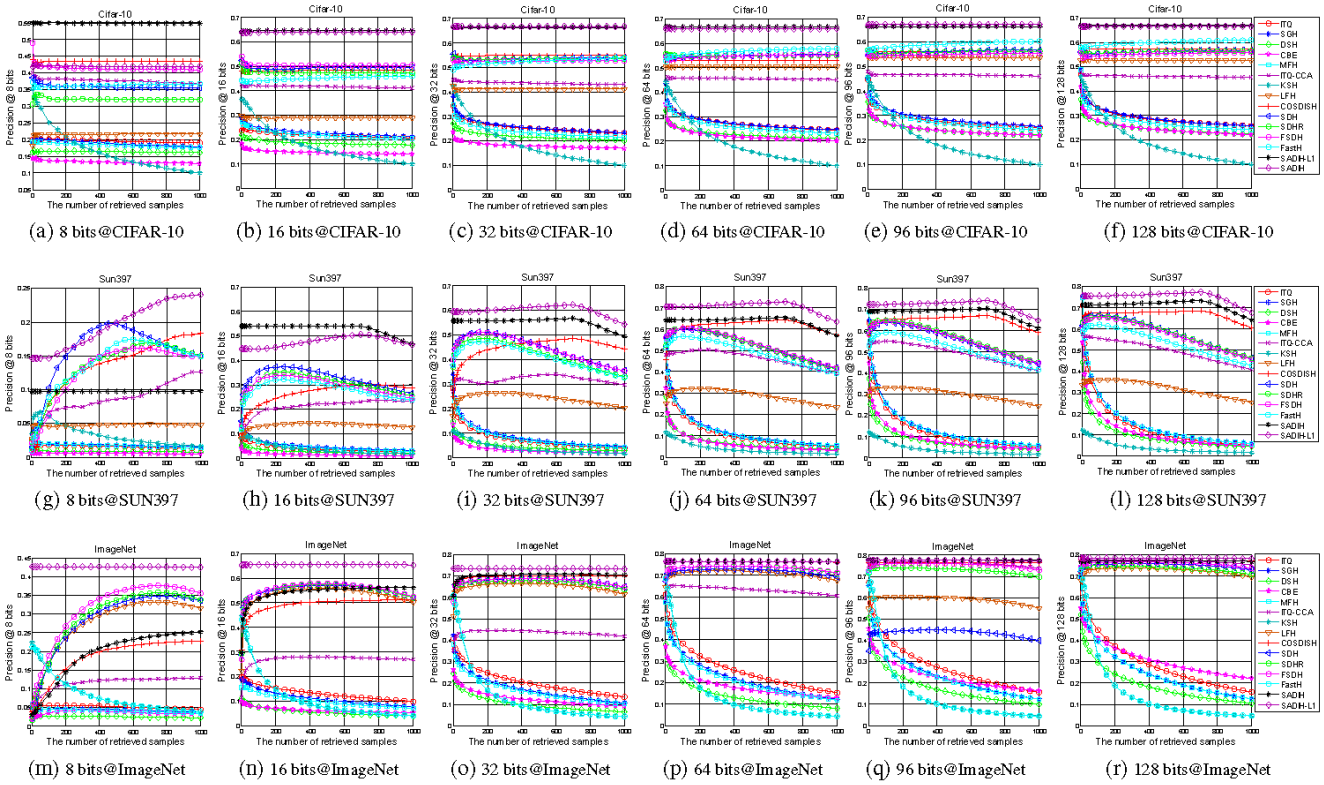
Similarly, the $4,096$-diml deep CNN features from VGG19 $fc7$ were used for evaluation.

Experimental Settings: Following the previous experimental configurations used in (Shen et al. 2015; Kang, Li, and Zhou 2016), we randomly split the CIFAR-10 dataset into a training set ($59K$ images) and a test query set ($1,000$ images), which has 100 images per category. In SUN-397, we randomly sample 100 images from each of the first 18 largest scene categories to formulate $1,800$ query images. For ImageNet, we randomly select 50 images for each category from the validation image dataset to construct the $5,000$ query image dataset. The semantic similarities on these datasets are measured whether two images share the same semantic label. For all datasets, we conduct feature normalization to make each dimension have zero-centered mean and equal variance.

Baseline Methods and Implementation Details: In experiments, we compare our SADIH and SADIH-L1 with 14 hashing methods including six unsupervised hashing methods, (*i.e.*, ITQ, SGH, DSH (Jin et al. 2014), CBE, and OCH) and eight supervised hashing methods (*i.e.*, ITQ-CCA (Gong et al. 2013), KSH, LFH, COSDISH, SDH, SDHR, FSDH, FastH). All the compared algorithms were performed five times with different random initializations, and the averaged experimental results were reported. To make fair comparison, all the compared methods were reimplemented using the released source codes given by the corresponding authors. Specifically, we searched the best parameters carefully for each algorithm by five-folds cross-validation, or directly employed the default parameters suggested by the original papers. For graph based method such as KSH and OCH, using the full semantic information for training is impossible due to the heavy computation complexity, and $5,000$ samples were selected from the training data for model construction. For our SSAH, the parameter γ was empirically set to 0.001. For the parameters α and β, we should tune it by cross-validation from the candidate set $\{0.01, 0.1, 1.0, 5, 10\}$. The maximum iteration number t was set to 5, which could assure the best performance.

Evaluation Measures: We adopted three frequently-used performance metrics (Manning, Raghavan, and others 2008) to evaluate different methods, *i.e.* mean average precision (MAP), the precision-recall curves and mean precision rate curves of top 1000 returned samples. Moreover, we also compared the computation time to show efficiency.

Quantitative Results: Table 1 explicitly illustrates the retrieval results of different algorithms with MAP and Precision@top100 as well as the computation time on CIFAR-10, SUN397 and ImageNet datasets. 1) Generally, supervised methods can achieve higher accuracies than unsupervised methods, while OCH and ITQ can lead to competitive performances. Importantly, the unsupervised methods are unsuitable to deal with large-scale image searching with large number of classes, such as SUN397 and ImageNet, while they can handle datasets with fewer categories such as CIFAR-10. 2) Our SADIH and SADIH-L1 in most cases can



(a) 8 bits@CIFAR-10 (b) 16 bits@CIFAR-10 (c) 32 bits@CIFAR-10 (d) 64 bits@CIFAR-10 (e) 96 bits@CIFAR-10 (f) 128 bits@CIFAR-10

(g) 8 bits@SUN397 (h) 16 bits@SUN397 (i) 32 bits@SUN397 (j) 64 bits@SUN397 (k) 96 bits@SUN397 (l) 128 bits@SUN397

(m) 8 bits@ImageNet (n) 16 bits@ImageNet (o) 32 bits@ImageNet (p) 64 bits@ImageNet (q) 96 bits@ImageNet (r) 128 bits@ImageNet

Figure 2: Precision curves of up to 1000 retrieved samples of different methods with different code lengths on CIFAR-10, SUN397 and ImageNet. (Better to view in color)

achieve the highest MAP scores with different code lengths, which demonstrate the efficacy of the proposed framework. Our SADIH and SADIH-L1 always produce superior performance to SDH, FSDH, FastH and SDHR, since our methods simultaneously consider the category information and pairwise similarities, however, other mentioned methods neglect the similarity preservation. Compared to KSH, LFH and COSDISH, our methods can make use of the $full$ similarities and discriminative category-level information for learning effective binary codes, yielding superior performance. 3) SADIH-L1 has a tendency to outperform SADIH, since l_1-norm regularization can generate more accurate approximation measurement. Increasing the coding lengths can improve the retrieval accuracies, but needs more training time. 4) Supervised methods seem to consume longer computation time compared to unsupervised ones. However, our SADIH-L1 is the fastest one in supervised methods, and SADIH can provide good balance between performance and time.

We further show the precision-recall curves of the compared methods with varying code lengths in Fig. 1. It can be observed that our methods are consistently better than all the competing methods, which indicates that our methods can retrieve more similar samples for a query at any fixed code length. Moreover, the precision variations *w.r.t.* different number of retrieved samples are illustrated in Fig. 2. We can observe that our methods are always superior to other methods, and their precisions are relatively stable with varying number of returned samples.

Conclusion

In this paper, we proposed a novel joint discriminative hashing framework, dubbed semantic-aware DIscrete Hashing (SADIH), which could efficiently guarantee the full semantic similarity preservation and discriminative semantic space construction. SADIH leveraged the asymmetric similarity approximation loss to preserve the full $n \times n$ similarities of the complete dataset. Meanwhile, the supervised semantic-aware autoencoder was designed to construct the discriminative semantic embedding space with full data variation preservation and good data reconstruction. The resulting problem was efficiently solved by the proposed discrete optimization algorithm. Extensive experimental results demonstrated the superiority of our methods on different large-scale datasets in terms of different evaluation protocols.

Acknowledgment This work is partially supported by ARC FT130101530.

References

Deng, J.; Dong, W.; Socher, R.; Li, L.-J.; Li, K.; and Fei-Fei, L. 2009. Imagenet: A large-scale hierarchical image database. In *CVPR*, 248–255. IEEE.

Dong, W.; Charikar, M.; and Li, K. 2008. Asymmetric dis-

tance estimation with sketches for similarity search in high-dimensional spaces. In *SIGIR*, 123–130. ACM.

Erin Liong, V.; Lu, J.; Wang, G.; Moulin, P.; and Zhou, J. 2015. Deep hashing for compact binary codes learning. In *CVPR*, 2475–2483.

Gionis, A.; Indyk, P.; Motwani, R.; et al. 1999. Similarity search in high dimensions via hashing. In *VLDB*, volume 99, 518–529.

Gong, Y.; Lazebnik, S.; Gordo, A.; and Perronnin, F. 2013. Iterative quantization: A procrustean approach to learning binary codes for large-scale image retrieval. *IEEE TPAMI* 35(12):2916–2929.

Gui, J.; Liu, T.; Sun, Z.; Tao, D.; and Tan, T. 2016. Supervised discrete hashing with relaxation. *TNNLS*.

Gui, J.; Liu, T.; Sun, Z.; Tao, D.; and Tan, T. 2018. Fast supervised discrete hashing. *TPAMI* 40(2):490–496.

Jiang, Q.-Y., and Li, W.-J. 2015. Scalable graph hashing with feature transformation. In *IJCAI*, 2248–2254.

Jiang, K.; Que, Q.; and Kulis, B. 2015. Revisiting kernelized locality-sensitive hashing for improved large-scale image retrieval. In *CVPR*, 4933–4941.

Jin, Z.; Li, C.; Lin, Y.; and Cai, D. 2014. Density sensitive hashing. *IEEE Trans. Cybern.* 44(8):1362–1371.

Kang, W.-C.; Li, W.-J.; and Zhou, Z.-H. 2016. Column sampling based discrete supervised hashing. In *AAAI*, 1230–1236.

Krizhevsky, A., and Hinton, G. 2009. Learning multiple layers of features from tiny images. Technical report, Citeseer.

Kulis, B., and Darrell, T. 2009. Learning to hash with binary reconstructive embeddings. In *NIPS*, 1042–1050.

Kulis, B., and Grauman, K. 2009. Kernelized locality-sensitive hashing for scalable image search. In *ICCV*, 2130–2137. IEEE.

Lai, H.; Pan, Y.; Liu, Y.; and Yan, S. 2015. Simultaneous feature learning and hash coding with deep neural networks. In *CVPR*, 3270–3278.

Lew, M.; Sebe, N.; Djeraba, C.; and Jain, R. 2006. Content-based multimedia information retrieval: State of the art and challenges. *ACM TOMM* 2(1):1–19.

Li, W.-J.; Wang, S.; and Kang, W.-C. 2016. Feature learning based deep supervised hashing with pairwise labels. In *IJCAI*, 1711–1717.

Lin, G.; Shen, C.; Shi, Q.; Van den Hengel, A.; and Suter, D. 2014. Fast supervised hashing with decision trees for high-dimensional data. In *CVPR*, 1963–1970.

Lin, K.; Yang, H.-F.; Hsiao, J.-H.; and Chen, C.-S. 2015. Deep learning of binary hash codes for fast image retrieval. In *CVPRW*, 27–35.

Liu, W.; Wang, J.; Ji, R.; Jiang, Y.; and Chang, S. 2012. Supervised hashing with kernels. In *CVPR*, 2074–2081. IEEE.

Liu, H.; Ji, R.; Wang, J.; and Shen, C. 2018. Ordinal constraint binary coding for approximate nearest neighbor search. *TPAMI* (99):1–1.

Luo, X.; Wu, Y.; and Xu, X. 2018. Scalable supervised discrete hashing for large-scale search. In *WWW*, 1603–1612.

Manning, C. D.; Raghavan, P.; et al. 2008. *Introduction to information retrieval*. Cambridge University Press.

Neyshabur, B.; Srebro, N.; Salakhutdinov, R. R.; Makarychev, Y.; and Yadollahpour, P. 2013. The power of asymmetry in binary hashing. In *NIPS*, 2823–2831.

Salakhutdinov, R., and Hinton, G. 2009. Semantic hashing. *IJAR* 50(7):969–978.

Shen, F.; Shen, C.; Liu, W.; and Tao Shen, H. 2015. Supervised discrete hashing. In *CVPR*, 37–45.

Shen, F.; Xu, Y.; Liu, L.; Yang, Y.; Huang, Z.; and Shen, H. T. 2018. Unsupervised deep hashing with similarity-adaptive and discrete optimization. *TPAMI*.

Shrivastava, A., and Li, P. 2014. Asymmetric lsh (alsh) for sublinear time maximum inner product search (mips). In *NIPS*, 2321–2329.

Simonyan, K., and Zisserman, A. 2014. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*.

Smeulders, A. W.; Worring, M.; Santini, S.; Gupta, A.; and Jain, R. 2000. Content-based image retrieval at the end of the early years. *TPAMI* (12):1349–1380.

Song, J.; Yang, Y.; Huang, Z.; Shen, H. T.; and Hong, R. 2011. Multiple feature hashing for real-time large scale near-duplicate video retrieval. In *ACMM*, 423–432. ACM.

Song, J.; Gao, L.; Liu, L.; Zhu, X.; and Sebe, N. 2018. Quantization-based hashing: a general framework for scalable image and video retrieval. *PR* 75:175–187.

Wang, J.; Zhang, T.; Sebe, N.; Shen, H. T.; et al. 2018. A survey on learning to hash. *IEEE TPAMI* 40(4):769–790.

Wang, J.; Kumar, S.; and Chang, S. 2012. Semi-supervised hashing for large-scale search. *TPAMI* (12):2393–2406.

Weiss, Y.; Torralba, A.; and Fergus, R. 2009. Spectral hashing. In *NIPS*, 1753–1760.

Xiao, J.; Hays, J.; Ehinger, K. A.; Oliva, A.; and Torralba, A. 2010. Sun database: Large-scale scene recognition from abbey to zoo. In *CVPR*, 3485–3492. IEEE.

Zhang, Z.; Liu, L.; Qin, J.; Zhu, F.; Shen, F.; Xu, Y.; Shao, L.; and Shen, H. 2018a. Highly-economized multi-view binary compression for scalable image clustering. In *ECCV*, 717–732.

Zhang, Z.; Liu, L.; Shen, F.; Shen, H. T.; and Shao, L. 2018b. Binary multi-view clustering. *IEEE TPAMI* 99:1–1.

Zhang, Z.; Shao, L.; Xu, Y.; Liu, L.; and Yang, J. 2018c. Marginal representation learning with graph structure self-adaptation. *IEEE TNNLS* 29:4645–4659.

Zhang, P.; Zhang, W.; and Li, W.-J. 2014. Supervised hashing with latent factor models. In *SIGIR*, 173–182. ACM.