Deep Hashing for Scalable Image Search

Jiwen Lu, *Senior Member, IEEE*, Venice Erin Liong, and Jie Zhou, *Senior Member, IEEE*

***Abstract*— In this paper, we propose a new deep hashing (DH) approach to learn compact binary codes for scalable image search. Unlike most existing binary codes learning methods, which usually seek a single linear projection to map each sample into a binary feature vector, we develop a deep neural network to seek multiple hierarchical non-linear transformations to learn these binary codes, so that the non-linear relationship of samples can be well exploited. Our model is learned under three constraints at the top layer of the developed deep network: 1) the loss between the compact real-valued code and the learned binary vector is minimized, 2) the binary codes distribute evenly on each bit, and 3) different bits are as independent as possible. To further improve the discriminative power of the learned binary codes, we extend DH into supervised DH (SDH) and multi-label SDH by including a discriminative term into the objective function of DH, which simultaneously maximizes the inter-class variations and minimizes the intra-class variations of the learned binary codes with the single-label and multi-label settings, respectively. Extensive experimental results on eight widely used image search data sets show that our proposed methods achieve very competitive results with the state-of-the-arts.**

***Index Terms*— Scalable image search, fast similarity search, hashing, deep learning, multi-label learning.**

I. Introduction

LARGE scale visual search has attracted great attention in computer vision in recent years due to the rapid growth of web data in forms of images and videos. The objective of large scale visual search aims to retrieve the most relevant visual content from large datasets in an accurate and efficient manner. While the conventional similarity search methods such as the nearest neighbor search and tree-based techniques have been widely used for low-dimensional data search, they are not scalable to high-dimensional data such as images and videos because they are usually represented as high-dimensional features. To address this, many hashing based approximate nearest neighbor (ANN) search methods [13], [14], [42], [74], [76], [81] have been proposed in recent years.

The basic idea of hashing-based approaches is to construct a series of hash functions to map each visual object into a binary feature vector so that visually similar samples are mapped into similar binary codes. By encoding high-dimensional real-valued feature vector as low-dimensional compact binary codes, we can significantly speed up the similarity computation and save storage space in memory during the search procedure. Existing hashing-based methods can be mainly classified into two categories: *data-independent* [1], [5], [22], [27], [51] and *data-dependent* [12], [13], [15], [32], [37], [48], [60], [74]. For the first category, random projections are usually employed to map samples into a feature space and then binarization is performed to compute binary codes. Representative methods in this category are locality sensitive hashing (LSH) [1] and its kernelized or discriminative extensions [22], [31], [51]. For the second category, various statistical learning techniques are used to learn hashing functions to map samples into binary codes. State-of-the-art methods include spectral hashing [81], binary reconstructive embedding (BRE) [32], iterative quantization (ITQ) [13], K-means hashing (KMH) [15], minimal-loss hashing (MLH) [48], and sequential projection learning hashing (SPLH) [74]. Recently, several nonlinear hashing methods have also been proposed to exploit the nonlinear structure and relationship of visual samples to improve the search performance [14], [21], [37], [40], [60]. More recently, there have been some attempts on learning-based hashing methods with multi-modal data such as images, texts, and videos [71], [72].

In this paper, we propose a new deep hashing (DH) method to learn compact binary codes for scalable image search. Fig. 1 illustrates the basic idea of the proposed approach. Unlike most existing binary codes learning methods which usually seek a single linear projection to map each sample into a binary vector, we develop a deep neural network to seek multiple hierarchical non-linear transformations to learn compact binary codes. Our model is learned under three constraints at the top layer of the deep network: 1) the loss between the compact real-valued code and the learned binary vector is minimized, 2) the binary codes distribute evenly on each bit, and 3) different bits are as independent as possible. To further improve the discriminative power of the learned binary codes, we extend DH into supervised DH (SDH) by including one discriminative term into the objective function of DH which simultaneously maximizes the inter-class variations and minimizes the intra-class variations of the learned binary

Manuscript received September 16, 2016; revised February 12, 2017; accepted March 1, 2017. Date of publication March 12, 2017; date of current version March 27, 2017. This work was supported in part by the National Key Research and Development Program of China under Grant 2016YFB1001001, in part by the National Natural Science Foundation of China under Grant 61672306, Grant 61225008, Grant 61572271, Grant 61527808, Grant 61373074, and Grant 61373090, in part by the National 1000 Young Talents Plan Program, in part by the National Basic Research Program of China under Grant 2014CB349304, in part by the Ministry of Education of China under Grant 20120002110033, and in part by the Tsinghua University Initiative Scientific Research Program. The associate editor coordinating the review of this manuscript and approving it for publication was Prof. Ling Shao.

J. Lu and J. Zhou are with the Department of Automation, Tsinghua University, Beijing 100084, China, also with the State Key Lab of Intelligent Technologies and Systems, Tsinghua University, Beijing 100084, China, and also with the Tsinghua National Laboratory for Information Science and Technology, Tsinghua University, Beijing 100084, China (e-mail: lujiwen@tsinghua.edu.cn; jzhou@tsinghua.edu.cn).

V. E. Liong is with the Rapid-Rich Object Search Laboratory, Interdisciplinary Graduate School, Nanyang Technological University, Singapore 639798 (e-mail: veniceer001@e.ntu.edu.sg).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TIP.2017.2678163

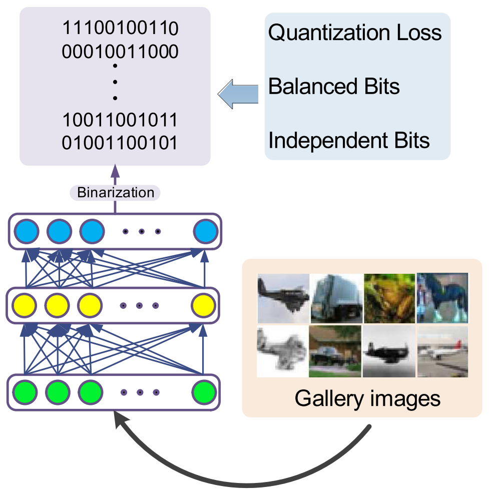


Fig. 1. The basic idea of our proposed deep hashing approach for compact binary codes learning. Given a gallery image set, we develop a deep neural network and learn the parameters of the network by using three criteria for the codes obtained at the top layer of the network: 1) minimizing loss between the compact real-valued code and the learned binary vector; 2) binary codes distribute evenly on each bit, and 3) each bit is as independent as possible. The parameters of the networks are updated by back-propagation based on the optimization objective function at the top layer.

codes. Since images are usually associated with multiple labels in real worlds, we develop a multi-label supervised DH (MSDH) by including a discriminative term into DH to define the between-class and within-class similarity of samples in the multi-label setting manner. Experimental results on eight widely used image retrieval datasets are presented to show the efficacy of the proposed methods.

The contributions of this work are summarized as follows:

1) We propose an unsupervised learning-based hashing method called deep hashing (DH) to learn multiple layers of projections to compute compact binary codes, so that the nonlinear relationship of samples can be well exploited.
2) We develop a supervised learning-based hashing method called supervised deep hashing (SDH) to learn discriminative binary codes with a deep network, under which conceptually similar images are projected as similar binary codes, which is helpful to scalable visual search.
3) We extend our SDH into the multi-label supervised deep hashing (MSDH) by computing the image similarity in a multi-label setting manner, so that scalable image search can be performed for large-scale multi-label image objects.
4) We conduct extensive image retrieval experiments on eight benchmark datasets to demonstrate the efficacy of our proposed methods. Experimental results show that our methods outperform most state-of-the-art hashing methods in both the unsupervised and supervised settings.

II. Background

In this section, we briefly review two related topics: 1) scalable image search and 2) deep learning.

A. Scalable Image Search

Efficient approximate nearest neighbor search algorithms are important to scalable image search. Representative methods include tree-based methods [3], [20], [46], [47], [57], [61] and quantization-based methods [6], [8], [12], [25], [49], [80], [85]. Many hashing-based methods have been proposed in recent years due to their excellent efficiency in both the storage and the search speed, which are suitable for large scale image search. While quantization-based methods provide high search accuracy gains due to their low quantization error and the employed table lookups, hashing-based methods provide faster retrieval speed since the Hamming distance computation only requires bit-wise operations. Hashing techniques have been used in many computer vision applications such as object recognition [68], [69], image retrieval [32], image matching [26], [64] and face recognition [44]. Existing hashing-based methods can be generally categorized into two classes: data-independent and data-dependent. Data-independent methods construct randomized hash functions by using random projections, where the most representative one is the LSH method. LSH preserves the cosine similarity of samples by using random projections obtained from Gaussian distributions to map samples into binary features [1]. However, it can only achieve satisfactory performance with longer codes, which is inefficient in many practical applications. In recent years, LSH has also been extended with some other similarity measure metrics. For example, Ji *et al.* [27] proposed an unbiased similarity estimation method by performing orthogonal random projections in a batch manner. Kulis *et al.* used the kernel similarity [31] and the Mahalanobis distance metric [32] with LSH to learn locality sensitive binary codes. Raginsky and Lazebnik used Gaussian kernels approximated by random fourier transforms [51], [52] to improve LSH. While Wang *et al.* [74] extended it to multiple kernels, their approach still used data-independent random projections, which cannot effectively exploit the geometrical and discriminative information of samples in constructing the hash functions. Hence, data-dependent hashing methods are more desirable and many such algorithms have proposed more recently.

Existing data-dependent hashing methods [13], [14], [42], [74], [81] can be mainly classified into three categories: unsupervised, semi-supervised, and supervised. For the first category, label information of the training set is not required in the learning procedure. For example, Weiss *et al.* [81] presented a spectral hashing method to obtain balanced binary codes by solving a spectral graph partitioning problem. This method has been extended to different versions such as kernel-based [14], hypergraph-based [43], and sparse PCA-based [58]. Gong and Lazebnik [13] developed an ITQ method by simultaneously maximizing the variance of each binary bit and minimizing the binarization loss. Liu *et al.* [41] proposed an anchor graph hashing method to preserve the neighborhood structure of samples to learn hash functions. He *et al.* [15] developed a KMH method by minimizing the hamming distance between the quantized cells and the cluster centers. Heo *et al.* [16] proposed a hypersphere-based hashing method by minimizing the spherical distance between the original real-valued features and the learned binary features. For the second category, the pairwise label information is used to learn hashing functions. For example, Wang *et al.* [73]

developed a semi-supervised hashing (SSH) method by minimizing the empirical error for pairwise labeled training samples and maximizing the variance of both labeled and unlabeled training samples. Kulis *et al.* [32] presented a BRE method by minimizing the reconstruction error between the original Euclidean distance and the learned hamming distance. Norouzi and Fleet [48] presented a MLH method by minimizing the loss between the learned Hamming distance and the quantization error. For the third category, the class label information of each sample is used in hashing function learning. For example, Stretcha *et al.* [64] developed a LDA hashing by minimizing the intra-class variations and maximizing the inter-class variations of binary codes. Rastegari *et al.* [54] proposed a discriminative hashing method by learning multiple linear-SVMs with the large margin criterion.

While these learning-based hashing methods have achieved reasonably good performance in many applications, most of them only seek a single linear projection, which may not be powerful enough to capture the nonlinear structure of samples. To address this, several non-linear hashing methods have been proposed recently. For example, Liu *et al.* [40] employed a kernel formulation and utilized supervised information through minimizing the distance of similar pairs and maximizing the distance of dissimilar pairs. Shen *et al.* [60] performed a non-parametric manifold learning to learn the hash functions to preserve the local structure of the training data. Shen *et al.* [59] also proposed a supervised discrete hashing method to learn binary codes by formulating hashing learning as a linear classification problem with binary constraints and discrete optimization. Lin *et al.* [37] proposed FastHash which performed binary code inference through graph cuts and learned the hash functions through a greedy boosted decision tree framework.

Real-world images are usually annotated with multiple labels due to their complex multilevel semantic structure. Hence, there also exists some hashing work to exploit the ranking order based on multiple labels. For example, Wang *et al.* [78] proposed a linear order-preserving hashing algorithm which maximizes the alignment between the order of the original feature representation and binary code. Lin *et al.* [37] proposed a StructHash algorithm which uses a structured SVM framework to directly optimize the rank-specific evaluation criterions [23]. Wang *et al.* [75] proposed a ranking-based supervised hashing using rank triplets to learn linear hash functions. Similarly, these models mostly perform linear transformations which is not representative enough except [86] which performed a deep training to optimize the multilabel criterions to learn the hash functions.

B. Deep Learning

Deep learning aims to learn hierarchical feature representations by building high-level features from raw data. In recent years, a variety of deep learning algorithms have been proposed in computer vision and machine learning [2], [17], [28], [34], [36], [53], [67], and some of them have successfully applied to many visual analysis applications image classification [30], object detection [65], action recognition [34], face verification [66], and visual tracking [79]. Representative deep learning methods include deep stacked auto-encoder [34], deep convolutional neural networks [28], and deep belief network [17].

Deep learning has achieved great success in various visual applications including scalable image search. To our knowledge, Semantic hashing [56] is the first work on using deep learning techniques to learn hashing functions for scalable image search. They applied the stacked Restricted Boltzmann Machine (RBM) learn compact binary codes for document search. However, their model is complex and requires pre-training, which is not efficient for practical applications. Srivastava and Salakhutdinov [63] proposed a deep Boltzmann machine approach for multi-modal retrieval. However, their method ignored the binary constraints in learning hashing functions. Recently, Xia *et al.* [82] proposed a supervised hashing method by learning image representations via convolutional neural networks. Feng *et al.* [9] presented a deep learning based hashing method by using stacked restricted Boltzmann machines, where semantic similarity of samples are used to fine tune the hash functions. Masci *et al.* [45] proposed a coupled siamese neural network to jointly maximize the intra-similarity and inter-similarity metric for cross-modality retrieval tasks, however, it is also effective for unimodal tasks. More recently, Lai *et al.* [33] and Zhao *et al.* [86] proposed a supervised hashing method to jointly learn image representations and binary codes into one stage with convolutional neural networks, where the optimization objective functions were formulated as a single-label learning and a multi-label learning problem, respectively. Lin *et al.* [38] learned binary codes by using a deep architecture which utilizes the hidden layers to represent the latent concepts dominated by the class labels. Liu *et al.* [39] proposed a deep supervised hashing (DSH) to encode pairwise images and impose regularization on the real-valued codes to approximate discrete binary values. Xia *et al.* [82] proposed a deep convolutional network to learn the hash codes by exploiting pairwise similarity matrix. In this work, we present a general framework of neural networks by seeking multiple hierarchical non-linear transformations to learn compact binary codes, where the hashing functions can be learned in unsupervised, supervised and multi-label supervised settings, which are desirable for various real applications where labeled samples are acquired in different ways.

III. Proposed Approach

In this section, we first present some basic knowledge of the learning-based hashing methods and then detail our proposed DH, SDH, and MSDH methods, respectively.

A. Learning-Based Hashing

Let $\mathbf{X} = [\mathbf{x}_1, \mathbf{x}_2, \cdots, \mathbf{x}_N] \in \mathbb{R}^{d\times N}$ be the training set which contains N samples, where $\mathbf{x}_n \in \mathbb{R}^d$ $(1 \le n \le N)$ is the nth sample in $\mathbf{X}$. Learning-based hashing methods aim to seek multiple hash functions to map and quantize each sample into a compact binary vector. Assume there are K hashing functions to be learned, which map each $\mathbf{x}_n$ into a K-bit binary codes vector $\mathbf{b}_n = [\mathbf{b}_{n1}, \cdots, \mathbf{b}_{nK}] \in \{-1, 1\}^{K\times 1}$, and the kth binary bit $\mathbf{b}_{nk}$ of $\mathbf{x}_n$ is computed as follows:

$$\mathrm{b}_{nk} = f_k(\mathbf{x}_n) = \mathrm{sgn}\,(g_k(\mathbf{x}_n)) = \mathrm{sgn}(\mathrm{w}_k^\top \mathbf{x}_n) \quad (1)$$

where f_k is the kth hashing function, and $\mathrm{w}_k \in \mathbb{R}^d$ is the projection in f_k, $\mathrm{sgn}(v)$ returns 1 if $v > 0$ and -1 otherwise.

Let $\mathrm{W} = [\mathrm{w}_1, \mathrm{w}_2, \cdots, \mathrm{w}_K] \in \mathbb{R}^{d \times K}$ be the projection matrix. Then, the mapping of $\mathbf{x}_n$ can be computed as: $g(\mathbf{x}_n) = \mathrm{W}^\top \mathbf{x}_n$, which can be further binarized to obtain the binary codes as follows:

$$\mathrm{b}_n = \mathrm{sgn}(\mathrm{W}^\top \mathbf{x}_n) \tag{2}$$

B. Deep Hashing

While a variety of learning-based hashing methods have been proposed in recent years [12], [13], [15], [32], [48], [74], most of them aim to learn a single projection matrix W, which cannot effectively capture the nonlinear relationship of samples. In this work, we propose a deep learning approach to learn multiple nonlinear transformations to seek compact binary codes for scalable image search.

As shown in Fig. 1, for a given sample $\mathbf{x}_n$, we obtain a binary vector $\mathbf{b}_n$ by passing it to a network which contains multiple stacked layers of nonlinear transformations. Assume there are $M+1$ layers in our deep network, and there are p^m units for the mth layer, where $m = 1, 2, \cdots, M$. For a given sample $\mathbf{x}_n \in \mathbb{R}^d$, the output of the first layer is: $\mathbf{h}_n^1 = s(\mathbf{W}^1\mathbf{x}_n + \mathbf{c}^1) \in \mathbb{R}^{p^1}$, where $\mathbf{W}^1 \in \mathbb{R}^{p^1 \times d}$ is the projection matrix to be learned at the first layer of the network, $\mathbf{c}^1 \in \mathbb{R}^{p^1}$ is the bias, and $s(\cdot)$ is a non-linear activation function. The output of the first layer is then considered as the input for the second layer, so that $\mathbf{h}_n^2 = s(\mathbf{W}^2\mathbf{h}_n^1 + \mathbf{c}^2) \in \mathbb{R}^{p^2}$, where $\mathbf{W}^2 \in \mathbb{R}^{p^2 \times p^1}$ and $\mathbf{c}^2 \in \mathbb{R}^{p^2}$ are the projection matrix and bias vector for the second layer, respectively. Similarly, the output for the mth layer is: $\mathbf{h}_n^m = s(\mathbf{W}^m\mathbf{h}_n^{m-1} + \mathbf{c}^m)$, and the output at the top layer of our network is:

$$g_{DH}(\mathbf{x}_n) = \mathbf{h}^M = s(\mathbf{W}^M\mathbf{h}_n^{M-1} + \mathbf{c}^M) \tag{3}$$

where the mapping $g_{DH} : \mathbb{R}^d \rightarrow \mathbb{R}^{p^M}$ is parameterized by $\{\mathbf{W}^m, \mathbf{c}^m\}_{m=1}^M$, $1 \le m \le M$, $\mathbf{h}^M$ is defined as the compact real-valued code learned from several nonlinear transformations of the original feature with specific constraints.

Now, we perform hashing for the output $\mathbf{h}^M$ at the top layer of the network to obtain binary codes as follows:

$$\mathbf{b}_n = \mathrm{sgn}(\mathbf{h}_n^M) \tag{4}$$

Let $\mathbf{B} = [\mathbf{b}_1, \ldots, \mathbf{b}_N] \in \{-1, 1\}^{K \times N}$ and $\mathbf{H}^m = [\mathbf{h}_1^m, \mathbf{h}_2^m, \cdots \mathbf{h}_N^m] \in \mathbb{R}^{p^m \times N}$ be the matrix representation of the binary codes vectors and the output of the mth layer of the network, we formulate the following optimization problem to learn the parameters of the network used in our deep hashing model:

$$\begin{aligned}\min_{\mathbf{W},\mathbf{c}} J &= J_1 - \lambda_1 J_2 + \lambda_2 J_3 + \lambda_3 J_4 \\ &= \frac{1}{2}\|\mathbf{B} - \mathbf{H}^M\|_F^2 \\ &\quad - \frac{\lambda_1}{2N}\mathrm{tr}(\mathbf{H}^M(\mathbf{H}^M)^\top) \\ &\quad + \frac{\lambda_2}{2}\sum_{m=1}^M \|\mathbf{W}^m(\mathbf{W}^m)^\top - \mathbf{I}\|_F^2 \\ &\quad + \frac{\lambda_3}{2}\sum_{m=1}^M (\|\mathbf{W}^m\|_F^2 + \|\mathbf{c}^m\|_2^2)\end{aligned} \tag{5}$$

where λ_1, λ_2, and λ_3 are three parameters to balance the effect of different terms. In (5), the first term J_1 aims to minimize the quantization loss between the learned binary vectors and the compact real-valued vectors, so that the energy of the learned compact features can be well preserved in the binary codes. The second term J_2 aims to maximize the variance of learned binary vectors to ensure balanced bits, so that the codes can be enforced as compact as possible. The third term J_3 enforces a relaxed orthogonality constrain on those projection matrices so that the independence of each transform is maximized [77], [84]. The last term J_4 is the regularization to control the scales of the parameters.

To solve this optimization problem, we employ the stochastic gradient descent method to learn parameters $\{\mathbf{W}^m, \mathbf{c}^m\}_{m=1}^M$. The gradient of the objective function in (5) with respect to different parameters are computed as follows:

$$\begin{aligned}\frac{\partial J}{\partial \mathbf{W}^m} &= \Delta^m(\mathbf{H}^{m-1})^\top \\ &\quad + \lambda_2(\mathbf{W}^m(\mathbf{W}^m)^\top - \mathbf{I})\mathbf{W}^m + \lambda_3\mathbf{W}^m\end{aligned} \tag{6}$$

$$\frac{\partial J}{\partial \mathbf{c}^m} = \bar{\Delta^m} + \lambda_3\mathbf{c}^m \tag{7}$$

where

$$\Delta^M = (-(\mathbf{B} - \mathbf{H}^M) - \lambda_1\mathbf{H}^M) \odot s'(\mathbf{Z}^M) \tag{8}$$

$$\Delta^m = ((\mathbf{W}_1^{m+1})^\top\Delta^{m+1}) \odot s'(\mathbf{Z}^m) \tag{9}$$

Here $\odot$ denotes element-wise multiplication, and $\mathbf{Z}^m = \mathbf{W}^m\mathbf{H}^{m-1} + \mathbf{c}^m$, and $\bar{\Delta^m} \in \mathbb{R}^{p^m}$ represents the mean of Δ^m. The parameters are updated by using the following gradient descent algorithm until convergence.

$$\mathbf{W}^m = \mathbf{W}^m - \eta\frac{\partial J}{\partial \mathbf{W}^m} \tag{10}$$

$$\mathbf{c}^m = \mathbf{c}^m - \eta\frac{\partial J}{\partial \mathbf{c}^m} \tag{11}$$

where η is the step-size.

Algorithm 1 summarizes the detailed procedure of the proposed DH method.

C. Supervised Deep Hashing

Since DH is an unsupervised learning approach, it is desirable to further improve its performance by using the label information of training samples if such information is available in the training stage. In this subsection, we propose a supervised deep hashing (SDH) method which extends DH into a supervised version to enhance the discriminative power of DH.

For each pair of training samples $(\mathbf{x}_i, \mathbf{x}_j)$, we know whether they are from the same class or not. Hence, we can construct two sets $\mathcal{S}$ or $\mathcal{D}$ from the training set, which represents the positive samples pairs and the negative samples pairs in the training set, respectively. Then, we formulate the following

Algorithm 1 DH

```
Input: Training set $\mathbf{X}$, network layer number $M$, learning rate
    $\eta$, iterative number $R$, parameters $\lambda_1$, $\lambda_2$ and $\lambda_3$, and
    convergence error $\varepsilon$.
Output: Parameters $\{\mathbf{W}^m, \mathbf{c}^m\}_{m=1}^M$.
Step 1 (Initialization):
    Initialize $\{\mathbf{W}^m, \mathbf{c}^m\}_{m=1}^M$ by getting the top $p^1$
    eigenvectors from the covariance matrix.
Step 2 (Optimization by backpropagation):
for $r = 1, 2, \cdots, R$ do
    Set $\mathbf{H}^0 = \mathbf{X}$
    for $m = 1, 2, \cdots, M$ do
        Compute $\mathbf{H}^m$ using the deep networks from (3).
    end
    for $m = M, M-1, \cdots, 1$ do
        Obtain the gradients according to (6)-(7).
    end
    for $m = 1, 2, \cdots, M$ do
        Update $\mathbf{W}^m$ and $\mathbf{c}^m$ according to (10)-(11).
    end
    Calculate $J_t$ using (5).
    If $r > 1$ and $|J_r - J_{r-1}| < \varepsilon$, go to Return.
end
Return: $\{\mathbf{W}^m, \mathbf{c}^m\}_{m=1}^M$.
```

optimization problem for our SDH method:

$$\begin{aligned}\min_{\mathbf{W},\mathbf{c}} J &= \frac{1}{2}\|\mathbf{B} - \mathbf{H}^M\|_F^2 \\ &- \frac{\lambda_1}{2}(\mathrm{tr}(\frac{1}{N}\mathbf{H}^M(\mathbf{H}^M)^\top) + \alpha\,\mathrm{tr}(\Sigma_B - \Sigma_W)) \\ &+ \frac{\lambda_2}{2}\sum_{m=1}^{M}\|\mathbf{W}^m(\mathbf{W}^m)^\top - \mathbf{I}\|_F^2 \\ &+ \frac{\lambda_3}{2}\sum_{m=1}^{M}(\|\mathbf{W}^m\|_F^2 + \|\mathbf{c}^m\|_2^2) \qquad (12)\end{aligned}$$

where

$$\begin{aligned}\Sigma_W &= \frac{1}{N_S}\sum_{(\mathbf{x}_i,\mathbf{x}_j)\in\mathcal{S}}(\mathbf{h}_i^M - \mathbf{h}_j^M)(\mathbf{h}_i^M - \mathbf{h}_j^M)^\top \\ &= \frac{1}{N_S}(\mathbf{H}_{s1}^M - \mathbf{H}_{s2}^M)(\mathbf{H}_{s1}^M - \mathbf{H}_{s2}^M)^\top \qquad (13)\\ \Sigma_B &= \frac{1}{N_D}\sum_{(\mathbf{x}_i,\mathbf{x}_j)\in\mathcal{D}}(\mathbf{h}_i^M - \mathbf{h}_j^M)(\mathbf{h}_i^M - \mathbf{h}_j^M)^\top \\ &= \frac{1}{N_D}(\mathbf{H}_{d1}^M - \mathbf{H}_{d2}^M)(\mathbf{H}_{d1}^M - \mathbf{H}_{d2}^M)^\top \qquad (14)\end{aligned}$$

N_S and N_D are the number of neighbor and non-neighbor pairs, Σ_W and Σ_B are computed from randomly selected within-class and between-class training samples, $\{\mathbf{H}_{s1}^m, \mathbf{H}_{s2}^m\}_{m=1}^M$ are the hidden representation of sample pairs in $\mathcal{S}$ and $\{\mathbf{H}_{d1}^m, \mathbf{H}_{d2}^m\}_{m=1}^M$ in $\mathcal{D}$. The objective of J_2 is to minimize the intra-class variations and maximize the inter-class variations, α is the parameter to balance these two parts in this term. The aims of J_1, J_3, and J_4 are the same as those of the DH method.

Similar to DH, we also use the stochastic gradient descent method to learn parameters $\{\mathbf{W}^m, \mathbf{c}^m\}_{m=1}^M$ in SDH. The gradient of the objective function in (12) with respect to these

Algorithm 2 SDH

```
Input: Training set $\mathbf{X}$, pairwise sample indices, network layer
    number $M$, learning rate $\eta$, iterative number $R$,
    parameter $\lambda_1$, $\lambda_2$, $\lambda_3$ and $\alpha$, and convergence error $\epsilon$.
Output: Parameters $\{\mathbf{W}^m, \mathbf{c}^m\}_{m=1}^M$.
Step 1 (Initialization):
    Initialize $\{\mathbf{W}^m, \mathbf{c}^m\}_{m=1}^M$ by getting the top $p^1$
    eigenvectors from a semi-supervised hashing
    method in [73].
Step 2 (Optimization by backpropagation):
for $r = 1, 2, \cdots, R$ do
    Set $\mathbf{H}^0 = \mathbf{X}$, $\{\mathbf{H}_{s1}^0, \mathbf{H}_{s2}^0\}$ and $\{\mathbf{H}_{d1}^0, \mathbf{H}_{d2}^0\}$ for pairwise
    samples in $\mathcal{S}$ and $\mathcal{D}$ respectively from set $\mathbf{X}$
    for $m = 1, 2, \cdots, M$ do
        Compute $\mathbf{H}^m$, $\mathbf{H}_{s1}^m$, $\mathbf{H}_{s2}^m$, $\mathbf{H}_{d1}^m$, and $\mathbf{H}_{d2}^m$ using the
        deep networks.
    end
    for $m = M, M-1, \cdots, 1$ do
        Obtain the gradients according to (15)-(16).
    end
    for $m = 1, 2, \cdots, M$ do
        Update $\mathbf{W}^m$ and $\mathbf{c}^m$ according to (10)-(11).
    end
    Calculate $J_t$ using (12).
    If $r > 1$ and $|J_r - J_{r-1}| < \varepsilon$, go to Return.
end
Return: $\{\mathbf{W}^m, \mathbf{c}^m\}_{m=1}^M$.
```

parameters can be computed as follows:

$$\begin{aligned}\frac{\partial J}{\partial \mathbf{W}^m} &= \Delta^m\mathbf{H}^{m-1} + \alpha\lambda_1(\Delta_{s1}^m\mathbf{H}_{s1}^{m-1} \\ &\quad - \Delta_{s2}^m\mathbf{H}_{s2}^{m-1} - \Delta_{d1}^m\mathbf{H}_{d1}^{m-1} \\ &\quad + \Delta_{d2}^m\mathbf{H}_{d2}^{m-1}) \\ &\quad + \lambda_2(\mathbf{W}^m(\mathbf{W}^m)^\top - \mathbf{I})\mathbf{W}^m \\ &\quad + \lambda_3\mathbf{W}^m \qquad (15)\\ \frac{\partial J}{\partial \mathbf{c}^m} &= \bar{\Delta}^m + \alpha\lambda_1(\bar{\Delta}_{s1}^m - \bar{\Delta}_{s2}^m \\ &\quad - \bar{\Delta}_{d1}^m + \bar{\Delta}_{d2}^m) + \lambda_3\mathbf{c}^m \qquad (16)\end{aligned}$$

where the Δ terms for the top layer can be computed as follows:

$$\Delta_{s1}^M = \frac{1}{N_S}(\mathbf{H}_{s1}^M - \mathbf{H}_{s2}^M) \odot s'(\mathbf{Z}_{s1}^M) \qquad (17)$$

$$\Delta_{s2}^M = \frac{1}{N_S}(\mathbf{H}_{s1}^M - \mathbf{H}_{s2}^M) \odot s'(\mathbf{Z}_{s1}^M) \qquad (18)$$

$$\Delta_{d1}^M = \frac{1}{N_D}(\mathbf{H}_{d1}^M - \mathbf{H}_{d2}^M) \odot s'(\mathbf{Z}_{d1}^M) \qquad (19)$$

$$\Delta_{d2}^M = \frac{1}{N_D}(\mathbf{H}_{d1}^M - \mathbf{H}_{d2}^M) \odot s'(\mathbf{Z}_{d1}^M) \qquad (20)$$

For the hidden layer, they can be computed as follows:

$$\Delta_{s1}^m = ((\mathbf{W}^{m+1})^\top\Delta_{s1}^{(m+1)}) \odot s'(\mathbf{Z}_{s1}^m) \qquad (21)$$

$$\Delta_{s2}^m = ((\mathbf{W}^{m+1})^\top\Delta_{s2}^{(m+1)}) \odot s'(\mathbf{Z}_{s2}^m) \qquad (22)$$

$$\Delta_{d1}^m = ((\mathbf{W}^{m+1})^\top\Delta_{d1}^{(m+1)}) \odot s'(\mathbf{Z}_{d1}^m) \qquad (23)$$

$$\Delta_{d2}^m = ((\mathbf{W}^{m+1})^\top\Delta_{d2}^{(m+1)}) \odot s'(\mathbf{Z}_{d2}^m) \qquad (24)$$

Algorithm 2 summarizes the detailed procedure of the proposed SDH method.

D. Supervised Multi-Label Deep Hashing

In this subsection, we present a supervised multi-label deep hashing method by exploiting the discriminative information of samples in the multi-label setting. We re-formulate the between-class and within-class scatter matrix of our SDH for multi-label samples and as follows [73]:

$$\Sigma_w^{(l)} = \sum_{i=1}^{N} \delta_{il}(\mathbf{h}_i^M - \mu_l)(\mathbf{h}_i^M - \mu_l)^\top \tag{25}$$

$$\Sigma_b^{(l)} = \sum_{i=1}^{N} \delta_{il}(\mu_l - \mu)(\mu_l - \mu)^\top \tag{26}$$

$$\Sigma_w = \sum_{l=1}^{L} \Sigma_w^{(l)} \tag{27}$$

$$\Sigma_b = \sum_{l=1}^{L} \Sigma_b^{(l)} \tag{28}$$

where Σ_w and Σ_b are the within-class and between-class scatter matrix, respectively, μ_l is the mean of the output of the top layer of all training samples belonging to the lth class, μ is the mean of all the outputs at the top layer, and $\delta_{il} = 1$ if the ith sample belongs to the lth class and 0 otherwise.

Since it is difficult to perform backpropagation due to the existence of μ and μ_l, we re-write (25)-(26) as follows by using the pairwise definition of the between-class and within-class variations of LDA [18]:

$$\Sigma_w^{(l)} = \sum_{i=1}^{N}\sum_{j=1}^{N} R_w(i,j)(\mathbf{h}_i^M - \mathbf{h}_j^M)(\mathbf{h}_i^M - \mathbf{h}_j^M)^\top \tag{29}$$

$$\Sigma_b^{(l)} = \sum_{i=1}^{N}\sum_{j=1}^{N} R_b(i,j)(\mathbf{h}_i^M - \mathbf{h}_j^M)(\mathbf{h}_i^M - \mathbf{h}_j^M)^\top \tag{30}$$

where

$$R_w^{(l)}(i,j) = \begin{cases} \frac{1}{N_l} & c(\mathbf{x}_i) = c(\mathbf{x}_j) = l \\ 0 & \text{otherwise} \end{cases} \tag{31}$$

$$R_b^{(l)}(i,j) = \begin{cases} \frac{1}{N} - \frac{1}{N_l} & c(\mathbf{x}_i) = c(\mathbf{x}_j) = l \\ \frac{1}{N} & \text{otherwise} \end{cases} \tag{32}$$

and $R_w = \sum_l R_w^{(l)}$ and $R_b = \sum_l R_b^{(l)}$ are the within-class and between-class weight matrices, respectively, $c(\mathbf{x}_i)$ is the label of $\mathbf{x}_i$. To further exploit the relationship of different labels, we use an affinity matrix A to define a weight obtained for sample pairs which are semantically similar or not according to the labels. In this work, we define the affinity matrix as follows

$$A_{ij} = 1 - \cos(\mathbf{y}_i, \mathbf{y}_j) \tag{33}$$

$\mathbf{y}_i = [y_{il}]_{l=1}^{L}$ is the label information if $\mathbf{x}_i$ where y_{il} is 1 if $\mathbf{x}_i$ belongs to the l^{th} class and 0, otherwise. Then, the new within-class and between weight matrices are defined as follows:

$$\tilde{R}_w = R_w \cdot A \tag{34}$$

$$\tilde{R}_b = R_b \cdot A \tag{35}$$

Lastly, the matrix from of them can be simplified as:

$$\Sigma_w^{(l)} = \mathrm{tr}(2\mathbf{H}^M S_w^{(l)} \mathbf{H}^{M\top} - 2\mathbf{H}^M \tilde{R}_w^{(l)} (\mathbf{H}^M)^\top) \tag{36}$$

$$\Sigma_b^{(l)} = \mathrm{tr}(2\mathbf{H}^M S_b^{(l)} \mathbf{H}^{M\top} - 2\mathbf{H}^M \tilde{R}_b^{(l)} (\mathbf{H}^M)^\top) \tag{37}$$

$$S_w^{(l)} = \mathrm{diag}(\sum_{j=1}^{N} \tilde{R}_w^{(l)}(\cdot, j)) \tag{38}$$

$$S_b^{(l)} = \mathrm{diag}(\sum_{j=1}^{N} \tilde{R}_b^{(l)}(\cdot, j)) \tag{39}$$

The stochastic gradient descent method is also employed to update the parameters of the model except Δ^M, which is modified as follows:

$$\Delta^M = \Big((\mathbf{H}^M - \mathbf{B}) - \lambda_1(\mathbf{H}^M + \alpha \mathbf{G}\mathbf{H}^M)\Big) \odot s'(\mathbf{Z}^M) \tag{40}$$

where $\mathbf{G} = \sum_{l=1}^{L} 2(S_w^{(l)} - \tilde{R}_w^{(l)} - S_b^{(l)} + \tilde{R}_b^{(l)})$.

IV. Experiments

In this section, we first conduct experiments on six widely used datasets (CIFAR-10, MNIST, SIFT-1M, GIST-1M, LabelMe-22k, and SUN397) to evaluate our proposed DH and SDH methods for scalable single-label image search. Then, we conduct experiments on two datasets (MIRFLICKR and NUS-WIDE) to evaluate our proposed MSDH method for scalable multi-label image search. The following describes the details of the experiments and results.

A. Results on CIFAR-10

The CIFAR-10 dataset [29] contains 60000 color images from 10 object classes, which are from the Tiny image dataset [68]. The size of each image is 32 × 32. Following the same setting in [74], we randomly sampled 1000 samples, 100 per class, as the query samples, and used the remaining 59000 images as the gallery set. From the gallery set, we randomly select a training set of 10000 samples. Each image was represented as a 512-D GIST feature vector [50].

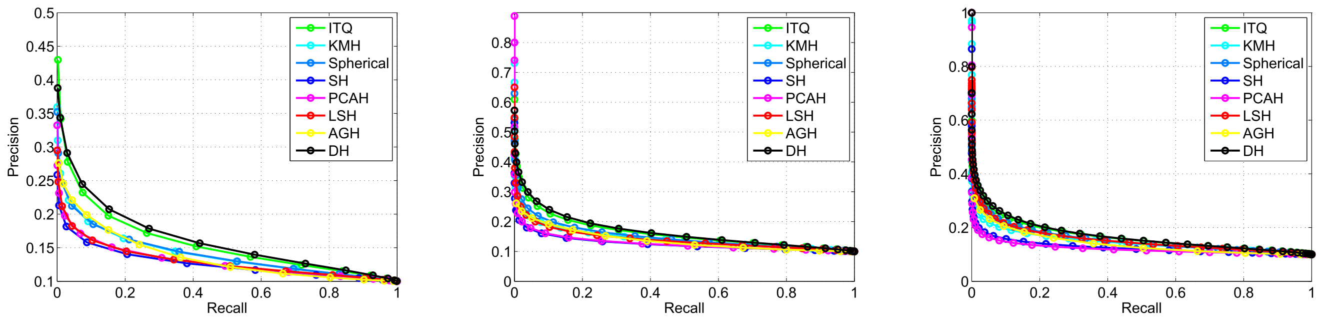
For our DH method, we trained our deep model with a 3-layer model by setting $M = 2$, where the dimensions for these layers were empirically set as [60 → 30 → 16], [80 → 50 → 32], and [100 → 80 → 64] for the 16, 32 and 64 bits experiments, respectively. For our SDH method, we trained our deep model with 4-layer model as [512 → 1000 → 1000 → K]. The parameters λ_1, λ_2 and λ_3 were empirically set as 100, 0.001 and 0.001, respectively. We used the rectified linear unit (relu) as the non-linear activation function in the hidden layer while a hyperbolic tangent function (tanh) is used at the last layer to provide a centered value ranging from {-1,1}. Initialization on the network parameters, $\{\mathbf{W}^m, \mathbf{c}^m\}_{m=1}^{M}$, are set following the Xavier initialization [11], and drop-out [62] rate of 0.5 is also implemented to prevent over-fitting. The parameter α for SDH was empirically set as 1.

1) Comparisons with State-of-the-art Hashing Methods: We compared our DH and SDH methods with fourteen state-of-the-art hashing methods, where seven of them are unsupervised and the other seven are supervised. The unsupervised methods include PCA-ITQ [13], KMH [15], Spherical [16],

TABLE I

RESULTS ON THE CIFAR-10 DATASET. RESULTS ON THE TOP SECTION ARE FROM UNSUPERVISED METHODS AND THOSE ON THE BOTTOM SECTION ARE FROM THE SUPERVISED METHODS. THE FIRST TWO COLUMNS SHOW THE HAMMING RANKING RESULTS EVALUATED BY mAP AND PRECISION@N (WHERE N=500). THE RIGHT COLUMN SHOWS THE HAMMING LOOK-UP RESULTS WHEN THE HAMMING RADIUS $r = 2$. HAMMING LOOK-UP FOR $K = 64$ IS NOT EVALUATED BECAUSE THIS EVALUATION IS IMPRACTICAL FOR LONGER CODES

Method	Hamming ranking (mAP, %)			precision (%) @ sample = 500			precision (%) @ r=2	
	16	32	64	16	32	64	16	32
PCA-ITQ [13]	15.67	16.20	16.64	22.46	25.30	27.09	22.60	14.99
KMH [15]	13.59	13.93	14.46	20.28	21.97	22.80	22.08	5.72
Spherical [16]	13.98	14.58	15.38	20.13	22.33	25.19	20.96	12.50
SH [81]	12.55	12.42	12.56	18.83	19.72	20.16	18.52	20.60
PCAH [74]	12.91	12.60	12.10	18.89	19.35	18.73	21.29	2.68
LSH [1]	12.55	13.76	15.07	16.21	19.10	22.25	16.73	7.07
AGH [41]	13.64	13.61	13.54	22.61	23.28	25.48	21.25	**24.53**
DH	**16.17**	**16.62**	**16.96**	**23.79**	**26.00**	**27.70**	**23.33**	15.77
SPLH [74]	17.61	20.20	20.98	25.32	29.43	32.22	23.05	30.47
MLH [48]	18.37	20.49	21.89	24.43	29.60	33.01	23.52	28.72
BRE [32]	14.42	15.14	15.88	20.68	22.86	25.14	20.89	20.29
KSH [40]	14.83	15.25	15.11	20.79	22.16	23.59	20.73	7.62
FastHash [37]	29.73	34.54	38.15	37.60	42.04	48.78	40.77	26.88
CCA-ITQ [13]	14.64	16.27	16.42	23.06	27.23	27.67	19.26	28.08
SDisH [59]	29.35	35.81	37.43	**39.48**	43.87	47.43	31.79	**42.77**
SDH	**31.01**	**35.88**	**38.50**	30.94	**47.32**	**50.95**	**69.18**	14.41



(a) (b) (c)

Fig. 2. Recall vs. precision curve on the CIFAR dataset. The first row shows the results of unsupervised hashing methods at 16, 32 and 64 bits, respectively. (a) 16 bits. (b) 32 bits. (c) 64 bits.

SH [81], PCAH [74], LSH [1] and AGH [41]. The supervised methods are SPLH [74], MLH [48], BRE [32],KSH [40], CCA-ITQ [13], FastHash [37] and SDisH [59]. For all these compared methods, we used the codes provided by the original authors and used the default parameters recommended by the corresponding papers.

We used the following three evaluation metrics to measure the performance of different methods: 1) mean average precision (mAP), which computes the area under the precision-recall curve and evaluates the overall performance of different hashing algorithms; 2) precision at N samples, which is the percentage of true neighbors among the top N retrieved samples. For datasets with label information, true neighbor corresponds to samples having the same label information as the query set. Otherwise, the true neighbor corresponds to the nearest neighbors computed according to the Euclidean metric in the original feature space; and 3) Hamming look-up result when the hamming radius is set as r, which measures the precision over all the points in the buckets that fall within a hamming radius of $r = 2$ provided that a failed search has zero precision.

Table I shows the search results of different hashing methods on the CIFAR-10 dataset. Fig. 2 shows the recall vs. precision curves for different methods on 16, 32 and 64 bits, respectively for the unsupervised methods. As can be seen, our DH method outperforms the other compared unsupervised hashing methods with the PCA-ITQ as its closest baseline. Our SDH outperforms some supervised hashing methods and are competitive with FastHash and SDisH. Fig. 3 presents some example query images and the retrieved neighbors on the CIFAR-10 dataset when 64 bits were used to learn binary codes for different hashing methods. We see that our DH and SDH methods show better search performance because higher semantic relevance can be obtained in the top retrieved samples.

2) Experiments Using CNN Features: Beside hand-crafted features, we also show the effectiveness of our methods when deep features which are extracted by the deep Convolutional Neural Networks (CNN) are used for scalable image search.[1] We conducted experiments on the CIFAR dataset by using the deep CNN features for various hashing methods. Except the CNN features, other settings in our experiments are the same

[1]CNN features have achieved many state-of-the-art performance in various visual analysis tasks such as image classification [30], object detection [10] and image matching [77]. Hence, we also demonstrate the performance of our methods when CNN features are used.

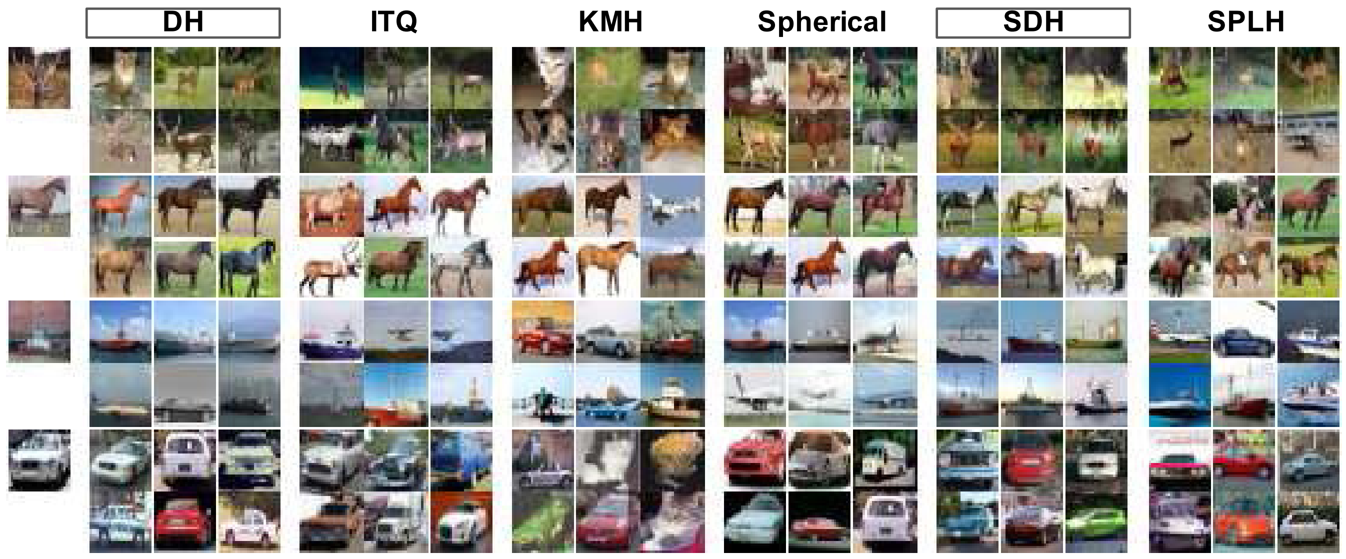
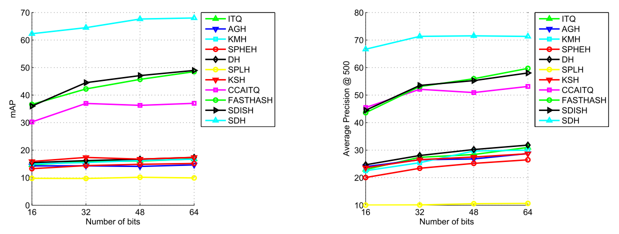


Fig. 3. Top retrieved 6 images of 4 queries returned by different hashing methods on the CIFAR dataset. Images in the first column are query samples. From left to right are the retrieved images by DH, ITQ, KMH, Spherical, SDH and SPLH when 64-bit binary codes are used for image search, respectively.



(a) (b)

Fig. 4. Ranking performance of different hashing methods using CNN features on the CIFAR-10 dataset. (a) mAP. (b) Average Precision.

as those used in the above experiments. We followed the same settings of the CNN model in [70] and used the MatConvNet toolbox to represent each image as a 4096-dimensional feature vector. We used the pre-trained model which was learned from ImageNet and employed the training set of the CIFAR dataset to finely tune the parameters of the deep model for feature extraction. In this experiment, we set the parameter of SDH and DH layers as $[4096 \rightarrow 500 \rightarrow 200 \rightarrow K]$. Fig. 4 shows the results of our DH and SDH methods compared to other hashing methods. For the KMH method, since it cannot handle high-dimensional features, we performed PCA first to reduce each CNN feature into a 512-dimensional feature vector. As can be seen, the performance of all hashing methods were improved due to the strong representation power of the CNN features. Particularly, the performance of our SDH method has an increase of approximately 0.3 in mAP when using CNN features and our SDH also outperforms all compared supervised hashing methods.

3) Comparison With Existing Deep Hashing Models: We compared our DH with semantic hashing [56], which is the first work of deep learning in hashing learning for image search. Semantic hashing learns a deep graphical model to mapping documents into compact binary codes, where semantically similar documents are mapped as close as possible. The deep generative model was pre-trained by stacked RBM and was fine-tuned by back-propagation based on the reconstruction cost. At the top layer of the network, binary codes are learned. Unlike our DH, semantic hashing does not consider other properties in binary codes learning. We followed the same settings in [69] and tuned the its parameters to obtain the best possible results on the CIFAR dataset. We also compared our SDH method with several supervised deep hashing methods. We particularly conducted experiments and compared our method with the multi-modal neural network (MM-NN) [45] method with the authors' code by using the GIST and CNN features. Results from other deep hashing methods including CNNH, DLBHC, DNNH and DSH were directly obtained from the corresponding papers. To fairly compare our method with these deep models, our SDH made use of CNN features and all gallery samples were used as training.

Table II shows the mAP performance of our DH and the semantic hashing method on the CIFAR-10 dataset. We see that our DH significantly outperforms the semantic hashing method on the CIFAR dataset. Table III shows the performance of our SDH and other compared supervised deep learning based hashing methods on the CIFAR-10 dataset. We see that our SDH achieves very competitive performance with DSH, and outperforms other compared supervised deep learning based hashing methods.

4) Comparisons of Using Different Layers: We conducted experiments to examine the performance of our DH method when different number of layers are used. We used the following layers: $[100 \rightarrow 64]$, $[100 \rightarrow 80 \rightarrow 64]$, $[200 \rightarrow 100 \rightarrow 80 \rightarrow 64]$, $[200 \rightarrow 150 \rightarrow 100 \rightarrow 80 \rightarrow 64]$ where 1-4 layers are used, respectively. Table IV shows the results obtained when different layers are used. We see that our DH achieves the best performance when the number of layer is set to 2 and 3. That is because we don't have extensive training examples to learn our model and it may be overfitted when the number of layers is large because there are more parameters to be trained in such a scenario.

TABLE II
COMPARISONS OF OUR DH AND THE SEMANTIC HASHING ON THE CIFAR-10 DATASET

Method	Hamming ranking (mAP, %)			precision (%) @ sample = 500			precision (%) @ r=2	
	16	32	64	16	32	64	16	32
DH	**16.17**	**16.62**	**16.96**	**23.79**	**26.00**	**27.70**	**23.33**	**15.77**
Semantic Hashing [56]	12.95	14.09	13.89	14.79	17.87	18.27	11.49	13.78

TABLE III
COMPARISONS OF OUR SDH AND OTHER DEEP HASHING MODELS ON THE CIFAR-10 DATASET

Method	Hamming ranking (mAP, %)			
	12	24	36	48
SDH	**63.12**	64.28	**66.50**	**69.50**
MM-NN (GIST) [45]	28.53	28.67	30.51	33.01
MM-NN (CNN) [45]	38.91	46.09	56.92	56.98
CNNH [82]	54.25	56.04	56.40	55.74
DLBHC [38]	55.03	58.03	57.78	58.85
DNNH [33]	57.08	58.75	58.99	59.04
DSH [39]	61.57	**65.12**	66.07	67.55

TABLE IV
COMPARISON OF DH WHEN DIFFERENT NUMBER OF LAYERS ARE USED ON THE CIFAR-10 DATASET

Method	Hamming ranking (mAP, %)			precision (%) @ sample = 500			precision (%) @ r=2	
	16	32	64	16	32	64	16	32
Layer 1	15.64	16.34	16.69	22.38	24.72	26.75	22.95	14.88
Layer 2	**16.17**	**16.62**	**16.96**	**23.79**	**26.00**	**27.70**	**23.33**	**15.77**
Layer 3	15.52	16.13	16.72	22.58	25.33	27.26	23.31	14.49
Layer 4	15.48	16.24	16.64	22.52	25.53	27.20	23.50	14.90

TABLE V
COMPUTATIONAL TIME OF DIFFERENT HASHING METHODS ON THE CIFAR-10 DATASET

Method	Training (seconds)			Test (seconds)		
	16	32	64	16	32	64
PCA-ITQ [13]	.34	.31	.37	.02	.03	.06
KMH [15]	458	425	1072	.92	.98	10.8
Spherical [16]	.23	.42	.94	.02	.03	.04
SH [81]	.04	.03	.04	.05	.05	.06
PCAH [74]	.04	.04	.05	.02	.03	.05
LSH [1]	.02	.02	.02	.01	.02	.02
AGH [41]	1.2	1.4	1.2	1.5	1.6	1.9
DH	.61	2.6	3.0	.69	.76	.85
SPLH [74]	3.2	7.0	14	.03	.05	.09
MLH [48]	1045	2174	2748	.04	.07	.09
BRE [32]	24	34	50	.04	.07	.11
KSH [40]	57	117	237	.49	.53	.56
CCA-ITQ [13]	.33	.28	.29	.14	.16	.21
FastHash [37]	82	159	323	9.8	18	28
SDisH [59]	3.1	10.6	10.3	2.6	2.9	3.2
SDH	100.0	101.6	109.7	3.1	3.1	3.2

5) Computational Time: We investigated the computational time of our DH and SDH methods, and compared them with those of other hashing methods. Our PC is configured with a 3.20GHz CPU and 32.0 GB RAM. Table V shows the training and test time of different hashing methods on the CIFAR-10 dataset when 16, 32 and 64-bit binary codes are used. The training time corresponds to the time of learning the hashing functions using the provided training set. This does not include the feature extraction which is similar across all hashing functions. The test time corresponds to extracting the binary codes from the query set using the learned hashing function. We see that the training time of our DH method is comparable to other previous unsupervised hashing methods, and our SDH method is one of the faster one among all the compared supervised hashing methods. Moreover, the test time of our methods are comparable to the existing hashing methods.

6) Influence of Different Constraints: We investigated the contributions of different terms in our DH model. We defined the following alternative baselines to study the importance of different terms in our feature learning model:

1) DH-1: learning the hashing function from J_1 and J_2.
2) DH-2: learning the hashing function from J_1 and J_3
3) DH-3: learning the hashing function from J_2 and J_3.

Table VI shows the mAP and precision performance of DH and the other three alternative variations on the CIFAR experiment. We see that all three terms our DH model contributes in the retrieval performance, and J_1 and J_3 seem to contribute more than J_2 which says the minimizing quantization loss and ensuring orthogonality gives larger contribution in the cost function. Moreover, the highest retrieval performance can be obtained when all the three terms are used together.

7) Evaluation Using the Class-Wise Splitting Protocol [55]: To further evaluate our hashing methods based on class-wise labels as nearest neighbor, we followed an alternative evaluation protocol from [55] which uses disjoint set of classes for training and testing. This is to show that our method is still effective in preserving the semantic information of certain

TABLE VI

RESULTS ON THE CIFAR DATASET OF OUR DH METHOD AND OTHER ALTERNATIVE BASELINES

Method	Hamming ranking (mAP, %)			precision (%) @ sample = 500			precision (%) @ r=2	
	16	32	64	16	32	64	16	32
DH-1	15.98	16.26	16.90	22.90	24.97	27.24	23.59	**17.15**
DH-2	15.69	16.15	16.85	22.61	25.11	27.48	23.40	14.12
DH-3	16.03	16.51	16.49	22.99	25.46	26.36	**23.68**	16.08
DH	**16.17**	**16.62**	**16.96**	**23.79**	**26.00**	**27.70**	23.33	15.77

TABLE VII

RESULTS ON THE CIFAR10 DATASET UNDER THE NEW PROTOCOL [55]

Method	Hamming ranking (mAP, %)			Average Precision (%) @ N = 500		
	16	32	64	16	32	64
PCA-ITQ [13]	43.79	45.39	45.87	53.66	56.70	58.88
PCAH	40.41	40.39	39.87	49.32	51.02	51.34
KMH [15]	43.70	43.55	44.23	53.59	56.00	57.33
Spherical [16]	41.52	43.26	44.44	51.60	54.74	56.87
AGH [41]	41.46	40.35	40.35	52.71	53.33	54.32
DH	**44.09**	**46.12**	**46.39**	**53.72**	**59.00**	**59.33**
KSH [40]	38.03	39.33	39.78	44.44	47.05	49.48
SPLH [74]	42.16	42.63	44.17	48.78	50.07	52.07
CCA-ITQ [13]	42.54	43.03	42.68	49.52	49.53	48.76
FastHash [37]	42.62	44.31	44.15	50.65	54.28	54.43
SDisH [59]	**44.61**	45.74	45.66	**52.07**	**54.46**	54.64
SDH	42.70	**46.22**	**48.25**	48.16	52.43	**55.09**

classes implicitly even if these class samples are not included in the training set. Specifically, we separated the samples based on class splits such that we used 70% of the classes for training, and 30% of the classes for testing. The samples that belong to the 30% of the classes was then separated into gallery and query set similar to the previous experiment. We repeated this procedure using 5 class splits and took the average of the results. Table VII shows the results for this experiment. Take note that the unsupervised hashing methods used a different set of class splits which explains the large difference between supervised and unsupervised results. We also see that the mAP performance of methods are generally higher than that of the previous protocol since there is less variation in the gallery set which consists of only 30% of the classes and fewer samples to retrieve from. However, our deep hashing methods still achieved competitive performance with the methods of comparison.

B. Results on MNIST

The MNIST dataset [35] consists of 70000 handwritten digit images from 10 classes (labeled from 0 to 9). The size of each image is 28×28. We randomly sampled 1000 samples, 100 per class, as the query data, and used the remaining 69000 images as the gallery set. Each image was represented as a 784-D gray-scale feature vector by using its intensity [50]. We followed the same settings as those used in the CIFAR-10 dataset and also used the same evaluation metrics to compare the performance of different hashing methods. Table VIII shows the search results of different hashing methods on the MNIST dataset.

1) Comparisons With State-of-the-Art Hashing Methods: Fig. 5 shows the recall vs. precision curves for different unsupervised hashing methods on 16, 32 and 64 bits, respectively. As can be seen, our DH and SDH methods are competitive with the existing unsupervised and supervised hashing methods, respectively. It is to be noted that the MNIST dataset consists of handwritten digit images which show simpler patterns and are well represented using only intensity features. Hence, it is relatively easier compared to other datasets which include complicated visual images. This leads to a lower requirement on the compared hashing methods. But, our SDH still shows better or competitive results than FastHash which shows its flexibility in capturing the nonlinearity of data samples regardless of its feature representation.

2) Influence of Different Constraints: We also investigate the influence of the different terms of the cost function of our DH model. We implement the alternative baselines similar to the CIFAR experiment. Table IX shows the performance of DH compared to the other three alternative variations on the MNIST experiment. We see that removing each term leads generally leads to a dip in performance of our DH model which shows that all three terms is important in the overall performance of our proposed method.

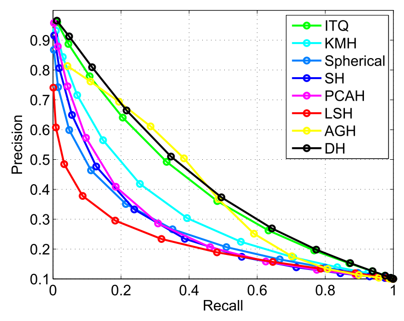
C. Results on LabelMe22k

The LabelMe dataset [69] contains 22000 object images, where each image was represented as 512-D GIST feature. Unlike the CIFAR and MNIST datasets, this dataset only provided the pairwise labels rather than the actual label, therefore cannot compare it with the fully supervised methods such as CCA-ITQ, FastHash and SDisH. For each sample, a maximum of 50 neighbors were provided as the ground truths for evaluation. Following the same evaluation protocol in [32], we randomly sampled 2000 images as the query data and used the remaining 20000 images are the gallery set. We applied

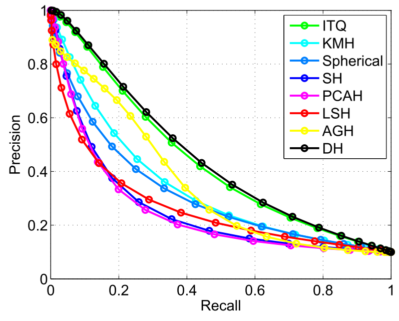
TABLE VIII

RESULTS ON THE MNIST DATASET. THE TOP SECTION ARE THE UNSUPERVISED METHODS AND THE BOTTOM SECTION ARE THE SUPERVISED METHODS. THE FIRST TWO COLUMNS SHOW THE HAMMING RANKING RESULTS EVALUATED BY mAP AND PRECISION@N WHERE N=500, AND THE LAST COLUMN SHOWS THE HAMMING LOOK-UP RESULTS WHEN THE HAMMING RADIUS $r = 2$

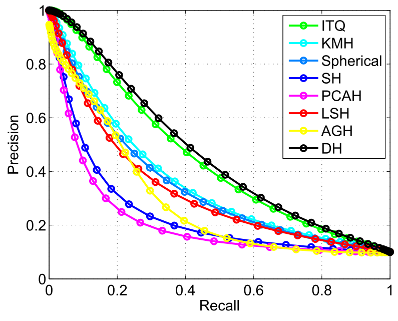
Method	Hamming ranking (mAP, %)			precision (%) @ sample = 500			precision (%) @ r=2	
	16	32	64	16	32	64	16	32
PCA-ITQ [13]	41.18	43.82	45.37	66.39	74.04	77.42	65.73	73.14
KMH [15]	32.12	33.29	35.78	60.43	67.19	72.65	61.88	68.85
Spherical [16]	25.81	30.77	34.75	49.48	61.27	69.85	51.71	64.26
SH [81]	26.64	25.72	24.10	56.29	61.29	61.98	57.52	65.31
PCAH [74]	27.33	24.85	21.47	56.56	59.99	57.97	36.36	65.54
LSH [1]	20.88	25.83	31.71	37.77	50.16	61.73	25.10	55.61
AGH [41]	39.92	33.39	28.64	70.75	73.93	74.79	64.69	**74.64**
DH	**43.14**	**44.97**	**46.74**	**67.89**	**74.72**	**78.63**	**66.10**	73.29
SPLH [74]	44.20	48.29	48.34	62.98	67.89	67.99	63.71	74.06
MLH [48]	43.12	48.04	48.12	61.51	66.98	67.10	62.14	72.14
BRE [32]	33.34	35.09	36.80	60.72	68.86	73.08	34.09	64.21
KSH [40]	36.17	35.44	34.61	56.10	63.07	66.74	57.16	53.95
CCA-ITQ [13]	64.88	68.23	71.44	75.76	78.88	81.06	68.65	77.80
FastHash [37]	**92.50**	94.18	95.19	**91.45**	93.84	94.51	78.68	**87.13**
SDisH [59]	70.25	75.17	77.72	74.44	79.85	82.45	61.68	76.28
SDH	89.79	**94.55**	**95.48**	89.00	**94.04**	**94.57**	**85.31**	68.40



(a)



(b)



(c)

Fig. 5. Recall vs. precision curve on the MNIST dataset for unsupervised hashing methods at 16, 32 and 64 bits, respectively. (a) 16 bits. (b) 32 bits. (c) 64 bits.

TABLE IX

RESULTS ON THE MNIST DATASET OF OUR DH METHOD AND OTHER ALTERNATIVE BASELINES

Method	Hamming ranking (mAP, %)			precision (%) @ sample = 500			precision (%) @ r=2	
	16	32	64	16	32	64	16	32
DH-1	36.86	39.39	40.93	63.01	70.11	72.44	56.99	**74.19**
DH-2	35.72	39.44	40.66	61.82	69.59	73.26	57.68	73.09
DH-3	36.59	40.27	40.67	62.79	70.84	72.59	58.55	73.85
DH	**40.47**	**42.70**	**44.85**	**65.78**	**73.13**	**77.00**	**65.98**	71.78

the Recall @ N samples to evaluate different methods, which is defined as the fraction of retrieved true neighbors to the total number of true neighbors. Table X shows the search performance and we see that our DH and SDH achieved very competitive results with the state-of-the-art hashing methods with a higher performance gap at 16 and 32 bits..

D. Results on SIFT-1M and GIST1M

We conducted experiments on the SIFT-1M and GIST-1M dataset to further show the effectiveness of our DH method on much larger datasets. The SIFT-1M dataset [25] contains 1 million 128-D SIFT features as the database and 1000 independent queries which were extracted from the INRIA Holiday Images [24]. The GIST-1M dataset [25] contains 1 million GIST 960-D features as the database and 1000 independent queries. Unlike other datasets which contain semantic information with the class label information of samples, we considered the groundtruth of each query as the K Euclidean nearest neighbor for both SIFT-1M and GIST-1M dataset, where K was set as 100 in our experiments and used the Hamming ranking for the search strategy. 100000 and 500000 learning samples were used for training the models for the SIFT-1M and GIST-1M dataset. Similar to the LabelMe22k experiment, we used the Recall @ N evaluation metric. Because the training set for this experiment is larger

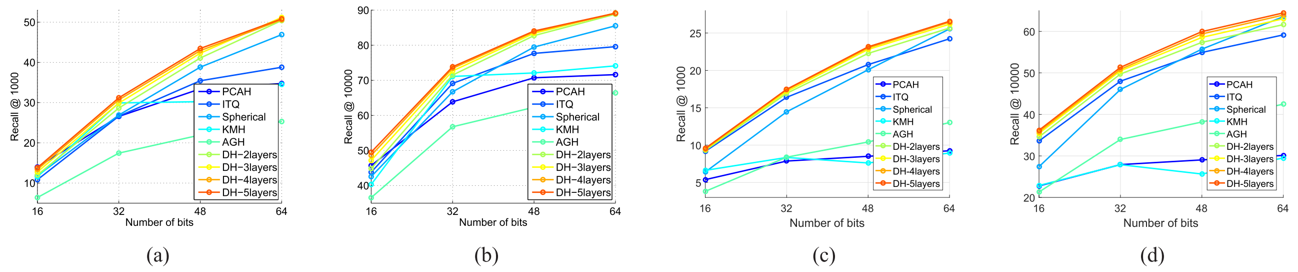


Fig. 6. ANN search performance of five unsupervised hashing methods compared to our deep hashing method on the SIFT-1M and GIST-1M. (a) SIFT-1M Recall @ 1000. (b) SIFT-1M Recall @ 10000. (c) GIST-1M Recall @ 1000. (d) GIST-1M Recall @ 10000.

TABLE X

RECALL@N RESULTS ON THE LABELME22K DATABASE WHEN N=1000. ON THE TOP SECTION ARE THE UNSUPERVISED METHODS AND ON THE BOTTOM SECTION ARE THE SUPERVISED METHODS, RESPECTIVELY

Method	Recall @ $N = 1000$		
	16	32	64
PCA-ITQ [13]	31.29	34.54	37.97
KMH [15]	30.58	32.65	34.31
Spherical [16]	26.51	29.96	35.04
SH [81]	22.04	24.09	24.28
PCAH [74]	23.11	20.06	18.49
LSH [1]	22.01	28.22	32.42
AGH [41]	30.12	29.91	29.89
DH	**32.60**	**35.51**	**38.42**
SPLH [74]	32.29	36.90	37.82
MLH [48]	32.05	37.21	36.66
BRE [32]	28.46	33.02	38.30
KSH [40]	**34.85**	37.65	40.50
SDH	33.79	**39.10**	**42.98**

compared to other datasets, we again explored the performance of our DH at different layers. We set that our DH to have 2 to 5 layers where each layer would consist of 100 nodes. Fig 6(a)-6(d) shows the Recall@1000 and Recall@10000 in varying code length compared with five unsupervised hashing methods and DH with different number of layers. In general, the performance of different unsupervised hashing methods in the SIFT1M dataset is much better since the GIST1M dataset is more challenging and the larger feature dimension. It can also be seen that for both datasets, our DH outperforms the compared methods at a lower retrieval number, $N = 10000$ and most especially at larger binary code length. Moreover, our DH with 5 layers shows the best results. This further shows that using more training samples led to better performance for our DH method.

E. Results on SUN397

The SUN397 [83] dataset consists of 108K images of 397 scene categories. This dataset is more challenging than the previous datasets because images captured in this dataset are more in the wild condition and has a larger number of classes and gallery samples. In our experiments, we randomly sampled 8000 query samples and used the remaining images as gallery set for training. Similar to one of the experiments on CIFAR, each image in this dataset is also represented as CNN feature and the MatConvNet toolbox was also used to extract the features, where the pre-trained model is learned on the ImageNet [7] dataset. In particular, we extracted the CNN features on the layer 'fc7' which results to 4096-dimensional feature vector for each image. Since this dataset is very challenging, we only obtain binary codes at the length of 48, 64 and 128, and cannot implement BRE and MLH due to their requirements of large length of binary codes.

Table XI shows the results of our methods as well as other popular hashing methods. As can be seen, the performance of all hashing methods in this dataset are relatively lower than those obtained on the CIFAR and MNIST datasets because the SUN397 dataset is more challenging. However, our DH and SDH methods still achieve very competitive performance with the other compared hashing methods for the both the unsupervised and supervised settings.

F. Results on MIRFLICKR-25K

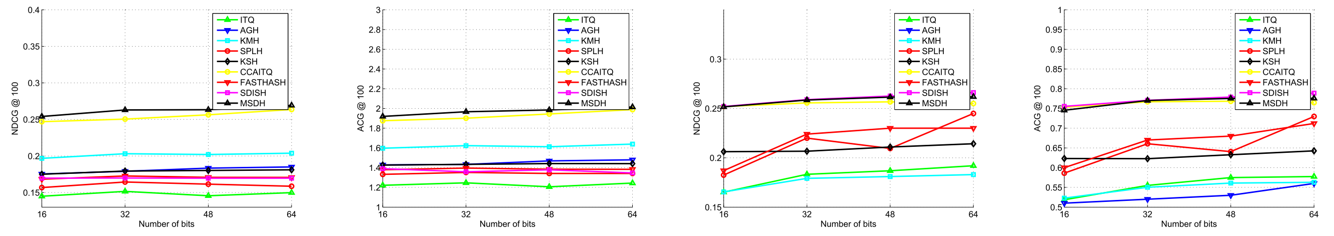
The MIRFLICKR-25K dataset [19] consists of 25k images from one million images obtained from the Flickr website as well as the associated tags. These images are annotated with 24 concepts which includes object categories (bird, tree, people) and scene categories (indoor, sky, night). An addition of 14 concepts is also added for stricter labeling in which each image is annotated with a concept only if the concept is salient. Therefore, a total of 38 labels are used to annotate each image and some of them are annotated with several labels. We randomly selected 2000 images as query samples and took the remaining as gallery and training samples. Similar to [63], each image is represented as a concatenation of Pyramid Histogram of Words (PHOW), GIST and MPEG-7 descriptors, which is finally represented as a 3857-dimensional feature vector. The common evaluation criteria for multi-label hashing [23], [86] are the Normalized Discounted Cumulative Gain (NDCG), and Average Cumulative Gain. The NDCG evaluates the ranking by penalizing errors in higher ranked items more strongly, while ACG takes the average of the similarity levels of data within the retrieved samples.

Fig. 7(a)-(b) show the NDCG and ACG results of different hashing methods with varying code length when the hand-crafted feature was used. In this experiment, we focused on hashing methods which show strong performance and omitted those which yields poor performance. Therefore, several supervised hashing methods such as SPLH, KSH, FastHash, SDisH, and CCA-ITQ and the unsupervised PCA-ITQ and KMH were

TABLE XI

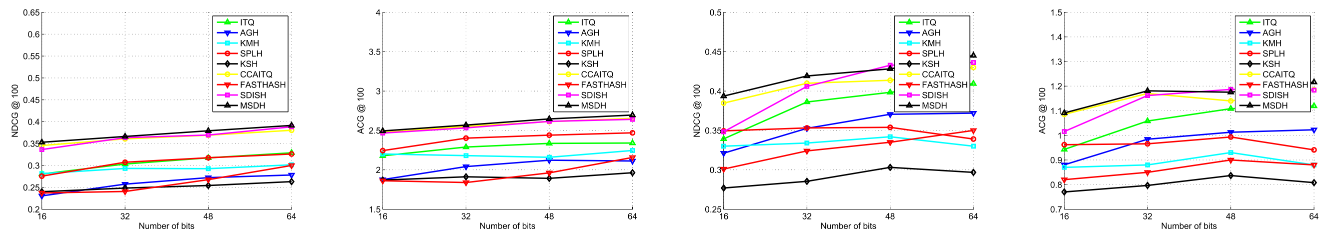
RESULTS ON THE SUN397 DATASET. RESULTS ON THE TOP SECTION ARE FROM UNSUPERVISED METHODS AND THOSE ON THE BOTTOM SECTION ARE FROM THE SEMI-SUPERVISED METHODS. THE FIRST TWO COLUMNS SHOW THE HAMMING RANKING RESULTS EVALUATED BY mAP AND PRECISION@N (WHERE N=2000)

Method	Hamming ranking (mAP, %)			Average Precision (%) @ N = 2000		
	48	64	128	48	64	128
PCA-ITQ [13]	5.18	5.67	6.60	5.83	6.31	6.81
KMH [15]	5.56	5.34	6.31	5.84	6.00	6.52
Spherical [16]	2.83	3.22	4.19	4.19	4.55	5.55
AGH [41]	2.30	2.69	3.06	4.32	4.73	4.87
DH	**5.75**	**5.92**	**7.10**	**6.34**	**6.65**	**7.18**
SPLH [74]	6.02	6.26	6.44	5.84	5.84	5.73
CCA-ITQ [13]	7.22	6.38	6.08	6.21	5.90	5.56
FastHash [37]	2.71	4.98	8.28	2.90	3.90	5.22
SDisH [59]	5.08	7.26	**11.08**	4.86	5.29	**7.18**
SDH	**8.53**	**8.01**	9.74	**7.22**	**7.26**	6.71



(a) (b) (c) (d)

Fig. 7. Ranking performance of different hashing methods on the MIRFlickr and NUS-WIDE datasets using hand-crafted features. (a) MIRFlickr - NDCG. (b) MIRFlickr - ACG. (c) NUSWIDE - NDCG. (d) NUSWIDE - ACG.



(a) (b) (c) (d)

Fig. 8. Ranking performance of different hashing methods on the MIRFlickr and NUS-WIDE datasets using CNN features. (a) MIRFlickr - NDCG. (b) MIRFlickr - ACG. (c) NUSWIDE - NDCG. (d) NUSWIDE - ACG.

selected for comparison. It can be seen that our MSDH method achieves better performance than the other hashing methods. Unlike other methods such as KSH, SPLH and FastHash which exploit only the pairwise similarity information to obtain the affinity matrices, our MSDH exploits a weighted within and between-class matrix using multi-label information. It is expected that using pairwise similarity may not fully capture the order and ranking of different samples during training. To further improve our performance, we also extracted the CNN features on the MIRFlickr dataset. We followed the same procedure which was used on the SUN397 dataset. Fig. 8(a)-(b) show the results of different hashing methods when the CNN features were used. As can be seen, the NDCG and ACG performance generally increased for all methods.

G. Results on NUS-WIDE

The NUS-WIDE dataset [4] contains 269648 images, which are also collected from the Flickr website and annotated with 81 concepts. Different from the MIRFlickr-25k, this dataset contains larger number of images and more diverse concepts which make the retrieval task more difficult. We removed the images that does not contain any of the 81 concepts, so that a total of 209347 images were left and used in our experiments. Similar to [86], We randomly selected 5000 images as query samples and took the remaining as gallery and training samples. Each image feature is represented as a concatenation of six different feature descriptors. Specifically, SIFT Bag-of-words, color histograms, color correlogram, edge direction histogram, wavelet texture and blockwise color moments were extracted and combined to represent each image as a 1134-dimensional feature vector.

Fig. 7(c)-(d) show the results of different hashing methods on the NUS-WIDE dataset when the hand-crafted feature was used. The performance is much lower than that of MIRFlickr due to the larger number of images and increased number of tags. However, our method still competitive with compared hashing methods. Particularly, our MSDH outperforms all methods at lower bit lenghts and is comparable with SDisH at 48-64 bit length. Similarly, we also investigated the performance of our method when the CNN feature is used. Fig. 8(c)-(d) show the results of different hashing methods when the CNN feature is used. As can be seen, the MDSH's performance increased by 0.15 and 0.35 for NDCG and ACG, respectively. It still follows the same trend in which our MSDH is competitive with other hashing methods.

V. Conclusion

In this paper, we have proposed a deep hashing (DH) approach for scalable image search. Instead of learning a single layer of projection in existing hashing methods, our approach employs a deep neural network to seek multiple hierarchical non-linear transformations to learn these binary codes, so that the nonlinear relationship of samples can be well exploited. We have further extended DH into supervised DH (SDH) and multi-label supervised DH (MSDH) by exploiting discriminative information in DH with the single-label and multi-label settings, respectively. Experimental results on six benchmark databases have clearly demonstrated the effectiveness of the proposed approach.

There are two interesting directions for future work:

1) It is interesting to extend our approach to scalable video search to better exploit the structure in videos to learn the hashing functions.
2) It is interesting to extend our approach to cross-modal search to better exploit the relationship of different modalities for scalable multimedia search.

References

[1] A. Andoni and P. Indyk, "Near-optimal hashing algorithms for approximate nearest neighbor in high dimensions," in *Proc. FOCS*, 2006, pp. 459–468.

[2] Y. Bengio, "Learning deep architectures for AI," *Found. Trends Mach. Learn.*, vol. 2, no. 1, pp. 1–127, 2009.

[3] J. L. Bentley, "Multidimensional binary search trees used for associative searching," *Commun. ACM*, vol. 18, no. 9, pp. 509–517, 1975.

[4] T.-S. Chua, J. Tang, R. Hong, H. Li, Z. Luo, and Y.-T. Zheng, "NUS-WIDE: A real-world Web image database from National University of Singapore," in *Proc. ACM ICIVR*, 2009, p. 48.

[5] O. Chum, J. Philbin, and A. Zisserman, "Near duplicate image detection: Min-hash and TF-IDF weighting," in *Proc. BMVC*, 2008, pp. 812–815.

[6] M. Datar, N. Immorlica, P. Indyk, and V. S. Mirrokni, "Locality-sensitive hashing scheme based on p-stable distributions," in *Proc. ACM Comput. Geometry*, 2004, pp. 253–262.

[7] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "ImageNet: A large-scale hierarchical image database," in *Proc. CVPR*, Jun. 2009, pp. 248–255.

[8] M. Douze, H. Jégou, and F. Perronnin, "Polysemous codes," in *Proc. ECCV*, 2016, pp. 785–801.

[9] W. Feng, B. Jia, and M. Zhu, "Deep hash: Semantic similarity preserved hash scheme," *Electron. Lett.*, vol. 50, no. 19, pp. 1347–1349, Sep. 2014.

[10] R. Girshick, J. Donahue, T. Darrell, and J. Malik, "Rich feature hierarchies for accurate object detection and semantic segmentation," in *Proc. CVPR*, 2014, pp. 580–587.

[11] X. Glorot and Y. Bengio, "Understanding the difficulty of training deep feedforward neural networks," in *Proc. AISTATS*, 2011, pp. 249–256.

[12] Y. Gong, S. Kumar, V. Verma, and S. Lazebnik, "Angular quantization-based binary codes for fast similarity search," in *Proc. NIPS*, 2012, pp. 1196–1204.

[13] Y. Gong and S. Lazebnik, "Iterative quantization: A procrustean approach to learning binary codes," in *Proc. CVPR*, Jun. 2011, pp. 817–824.

[14] J. He, W. Liu, and S.-F. Chang, "Scalable similarity search with optimized kernel hashing," in *Proc. KDD*, 2010, pp. 1129–1138.

[15] K. He, F. Wen, and J. Sun, "K-means hashing: An affinity-preserving quantization method for learning binary compact codes," in *Proc. CVPR*, Jun. 2013, pp. 2938–2945.

[16] J.-P. Heo, Y. Lee, J. He, S.-F. Chang, and S.-E. Yoon, "Spherical hashing," in *Proc. CVPR*, Jun. 2012, pp. 2957–2964.

[17] G. E. Hinton and R. R. Salakhutdinov, "Reducing the dimensionality of data with neural networks," *Science*, vol. 313, no. 5786, pp. 504–507, 2006.

[18] H. Hu, "Orthogonal neighborhood preserving discriminant analysis for face recognition," *Pattern Recognit.*, vol. 41, no. 6, pp. 2045–2054, Jun. 2008.

[19] M. J. Huiskes and M. S. Lew, "The MIR Flickr retrieval evaluation," in *Proc. ACM MIR*, 2008, pp. 39–43.

[20] P. Indyk and R. Motwani, "Approximate nearest neighbors: Towards removing the curse of dimensionality," in *Proc. ACM Symp. Theory Comput.*, 1998, pp. 604–613.

[21] G. Irie, Z. Li, X.-M. Wu, and S.-F. Chang, "Locally linear hashing for extracting non-linear manifolds," in *Proc. CVPR*, Jun. 2014, pp. 2123–2130.

[22] P. Jain, B. Kulis, and K. Grauman, "Fast image search for learned metrics," in *Proc. CVPR*, Jun. 2008, pp. 1–8.

[23] K. Järvelin and J. Kekäläinen, "Ir evaluation methods for retrieving highly relevant documents," in *Proc. ACM SIGIR*, 2000, pp. 41–48.

[24] H. Jegou, M. Douze, and C. Schmid, "Hamming embedding and weak geometric consistency for large scale image search," in *Proc. ECCV*, 2008, pp. 304–317.

[25] H. Jegou, M. Douze, and C. Schmid, "Product quantization for nearest neighbor search," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 33, no. 1, pp. 117–128, Jan. 2011.

[26] H. Jégou, F. Perronnin, M. Douze, J. Sánchez, P. Pérez, and C. Schmid, "Aggregating local image descriptors into compact codes," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 34, no. 9, pp. 1704–1716, Sep. 2012.

[27] J. Ji, J. Li, S. Yan, B. Zhang, and Q. Tian, "Super-bit locality-sensitive hashing," in *Proc. NIPS*, 2012, pp. 108–116.

[28] S. Ji, W. Xu, M. Yang, and K. Yu, "3D convolutional neural networks for human action recognition," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 35, no. 1, pp. 221–231, Jan. 2013.

[29] A. Krizhevsky and G. Hinton, "Learning multiple layers of features from tiny images," Univ. Toronto, Toronto, ON, Canada, Tech. Rep. 1, 2009.

[30] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet classification with deep convolutional neural networks," in *Proc. NIPS*, 2012, pp. 1097–1105.

[31] B. Kulis and K. Grauman, "Kernelized locality-sensitive hashing for scalable image search," in *Proc. ICCV*, 2009, pp. 2130–2137.

[32] B. Kulis, P. Jain, and K. Grauman, "Fast similarity search for learned metrics," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 31, no. 12, pp. 2143–2157, Dec. 2009.

[33] H. Lai, Y. Pan, Y. Liu, and S. Yan, "Simultaneous feature learning and hash coding with deep neural networks," in *Proc. CVPR*, 2015, pp. 3270–3278.

[34] Q. V. Le, W. Y. Zou, S. Y. Yeung, and A. Y. Ng, "Learning hierarchical invariant spatio-temporal features for action recognition with independent subspace analysis," in *Proc. CVPR*, Jun. 2011, pp. 3361–3368.

[35] Y. LeCun and C. Cortes, "The MNIST database of handwritten digits," New York Univ., New York, NY, USA, Tech. Rep. 1, 1998.

[36] H. Lee, R. Grosse, R. Ranganath, and A. Y. Ng, "Convolutional deep belief networks for scalable unsupervised learning of hierarchical representations," in *Proc. ICML*, 2009, pp. 609–616.

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

[38] K. Lin, H.-F. Yang, J.-H. Hsiao, and C.-S. Chen, "Deep learning of binary hash codes for fast image retrieval," in *Proc. CVPRW*, Jun. 2015, pp. 27–35.

[39] H. Liu, R. Wang, S. Shan, and X. Chen, "Deep supervised hashing for fast image retrieval," in *Proc. CVPR*, Jun. 2015, pp. 2064–2072.

[40] W. Liu, J. Wang, R. Ji, Y.-G. Jiang, and S.-F. Chang, "Supervised hashing with kernels," in *Proc. CVPR*, Jun. 2012, pp. 2074–2081.

[41] W. Liu, J. Wang, S. Kumar, and S.-F. Chang, "Hashing with graphs," in *Proc. ICML*, 2011, pp. 1–8.

[42] X. Liu, J. He, B. Lang, and S.-F. Chang, "Hash bit selection: A unified solution for selection problems in hashing," in *Proc. CVPR*, Jun. 2013, pp. 1570–1577.

[43] Y. Liu, J. Shao, J. Xiao, F. Wu, and Y. Zhuang, "Hypergraph Spectral Hashing for image retrieval with heterogeneous social contexts," *Neurocomputing*, vol. 119, pp. 49–58, Nov. 2013.

[44] J. Lu, V. E. Liong, X. Zhou, and J. Zhou, "Learning compact binary face descriptor for face recognition," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 37, no. 10, pp. 2041–2056, Oct. 2015.

[45] J. Masci, M. M. Bronstein, A. M. Bronstein, and J. Schmidhuber, "Multimodal similarity-preserving hashing," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 36, no. 4, pp. 824–830, Apr. 2014.

[46] J. Masci, M. M. Bronstein, A. M. Bronstein, and J. Schmidhuber, "Fast approximate nearest neighbors with automatic algorithm configuration," *VISAPP*, vol. 2, no. 1, pp. 331–340, 2009.

[47] M. Muja and D. G. Lowe, "Scalable nearest neighbor algorithms for high dimensional data," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 36, no. 11, pp. 2227–2240, Nov. 2014.

[48] M. Norouzi and D. M. Blei, "Minimal loss hashing for compact binary codes," in *Proc. ICML*, 2011, pp. 353–360.

[49] M. Norouzi and D. J. Fleet, "Cartesian K-means," in *Proc. CVPR*, Jun. 2013, pp. 3017–3024.

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

[51] M. Raginsky and S. Lazebnik, "Locality-sensitive binary codes from shift-invariant kernels," in *Proc. NIPS*, 2009, pp. 1509–1517.

[52] A. Rahimi and B. Recht, "Random features for large-scale kernel machines," in *Proc. NIPS*, 2007, pp. 1177–1184.

[53] M. Ranzato, F. J. Huang, Y.-L. Boureau, and Y. LeCun, "Unsupervised learning of invariant feature hierarchies with applications to object recognition," in *Proc. CVPR*, Jun. 2007, pp. 1–8.

[54] M. Rastegari, A. Farhadi, and D. Forsyth, "Attribute discovery via predictable discriminative binary codes," in *Proc. ECCV*, 2012, pp. 876–889.

[55] A. Sablayrolles, M. Douze, H. Jégou, and N. Usunier. (2016). "How should we evaluate supervised hashing?" [Online]. Available: https://arxiv.org/abs/1609.06753

[56] R. Salakhutdinov and G. Hinton, "Semantic hashing," *Int. J. Approx. Reasoning*, vol. 50, no. 7, pp. 969–978, Jul. 2009.

[57] G. Shakhnarovich, T. Darrell, and P. Indyk, "Nearest-neighbor methods in learning and vision," *IEEE Trans. Neural Netw.*, vol. 19, no. 2, p. 377, Feb. 2008.

[58] J. Shao, F. Wu, C. Ouyang, and X. Zhang, "Sparse spectral hashing," *Pattern Recognit. Lett.*, vol. 33, no. 3, pp. 271–277, Feb. 2012.

[59] F. Shen, C. Shen, W. Liu, and H. T. Shen, "Supervised discrete hashing," in *Proc. CVPR*, Jun. 2015, pp. 37–45.

[60] F. Shen, C. Shen, Q. Shi, A. van den Hengel, and Z. Tang, "Inductive hashing on manifolds," in *Proc. CVPR*, 2013, pp. 1562–1569.

[61] C. Silpa-Anan and R. Hartley, "Optimised KD-trees for fast image descriptor matching," in *Proc. CVPR*, Jun. 2008, pp. 1–8.

[62] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: A simple way to prevent neural networks from overfitting," *J. Mach. Learn. Res.*, vol. 15, no. 1, pp. 1929–1958, 2014.

[63] N. Srivastava and R. R. Salakhutdinov, "Multimodal learning with deep Boltzmann machines," in *Proc. NIPS*, 2012, pp. 2222–2230.

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

[65] C. Szegedy, A. Toshev, and D. Erhan, "Deep neural networks for object detection," in *Proc. NIPS*, 2013, pp. 2553–2561.

[66] Y. Taigman, M. Yang, M. Ranzato, and L. Wolf, "DeepFace: Closing the gap to human-level performance in face verification," in *Proc. CVPR*, Jun. 2014, pp. 1701–1708.

[67] G. W. Taylor, R. Fergus, Y. LeCun, and C. Bregler, "Convolutional learning of spatio-temporal features," in *Proc. ECCV*, 2010, pp. 140–153.

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

[69] A. Torralba, R. Fergus, and Y. Weiss, "Small codes and large image databases for recognition," in *Proc. CVPR*, Jun. 2008, pp. 1–8.

[70] A. Vedaldi, and K. Lenc, "Matconvnet: Convolutional neural networks for MATLAB," in *Proc. ACM MM*, Oct. 2015, pp. 689–692.

[71] D. Wang, X. Gao, X. Wang, and L. He, "Semantic topic multimodal hashing for cross-media retrieval," in *Proc. IJCAI*, 2015, pp. 3890–3896.

[72] D. Wang, X. Gao, X. Wang, L. He, and B. Yuan, "Multimodal discriminative binary embedding for large-scale cross-modal retrieval," *IEEE Trans. Image Process.*, vol. 25, no. 10, pp. 4540–4554, Oct. 2016.

[73] H. Wang, C. Ding, and H. Huang, "Multi-label linear discriminant analysis," in *Proc. ECCV*, 2010, pp. 126–139.

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

[75] J. Wang, W. Liu, A. X. Sun, and Y.-G. Jiang, "Learning hash codes with listwise supervision," in *Proc. CVPR*, 2013, pp. 3032–3039.

[76] J. Wang, H. T. Shen, J. Song, and J. Ji. (2014). "Hashing for similarity search: A survey." [Online]. Available: https://arxiv.org/abs/1408.2927

[77] J. Wang *et al.*, "Learning fine-grained image similarity with deep ranking," in *Proc. CVPR*, 2014, pp. 1386–1393.

[78] J. Wang, J. Wang, N. Yu, and S. Li, "Order preserving hashing for approximate nearest neighbor search," in *Proc. ACM MM*, 2013, pp. 133–142.

[79] N. Wang and D.-Y. Yeung, "Learning a deep compact image representation for visual tracking," in *Proc. NIPS*, 2013, pp. 809–817.

[80] X. Wang, T. Zhang, G.-J. Qi, J. Tang, and J. Wang, "Supervised quantization for similarity search," in *Proc. CVPR*, Jun. 2016, pp. 2018–2026.

[81] Y. Weiss, A. Torralba, and R. Fergus, "Spectral hashing," in *Proc. NIPS*, 2008, pp. 1753–1760.

[82] R. Xia, Y. Pan, H. Lai, C. Liu, and S. Yan, "Supervised hashing for image retrieval via image representation learning," in *Proc. AAAI*, 2014, pp. 2156–2162.

[83] J. Xiao, J. Hays, K. A. Ehinger, A. Oliva, and A. Torralba, "SUN database: Large-scale scene recognition from abbey to zoo," in *Proc. CVPR*, Jun. 2010, pp. 3485–3492.

[84] T. Yao, F. Long, T. Mei, and Y. Rui, "Deep semantic-preserving and ranking-based hashing for image retrieval," in *Proc. IJCAI*, 2016, pp. 3931–3937.

[85] T. Zhang, C. Du, and J. Wang, "Composite quantization for approximate nearest neighbor search," in *Proc. ICML*, 2014, pp. 838–846.

[86] F. Zhao, Y. Huang, L. Wang, and T. Tan, "Deep semantic ranking based hashing for multi-label image retrieval," in *Proc. CVPR*, 2015, pp. 1556–1564.

Jiwen Lu (S'10–M'11–SM'15) received the B.Eng. degree in mechanical engineering and the M.Eng. degree in electrical engineering from the Xi'an University of Technology, Xi'an, China, in 2003 and 2006, respectively, and the Ph.D. degree in electrical engineering from Nanyang Technological University, Singapore, in 2012. From 2011 to 2015, he was a Research Scientist with the Advanced Digital Sciences Center, Singapore. He is currently an Associate Professor with the Department of Automation, Tsinghua University, Beijing, China. He has authored or co-authored over 150 scientific papers in these areas, where 40 were the IEEE Transactions papers. His current research interests include computer vision, pattern recognition, and machine learning. He serves/has served as an Associate Editor of *Pattern Recognition Letters*, *Neurocomputing*, and the IEEE ACCESS, a Managing Guest Editor of *Pattern Recognition* and *Image and Vision Computing*, a Guest Editor of *Computer Vision and Image Understanding*, and an Elected Member of the Information Forensics and Security Technical Committee of the IEEE Signal Processing Society. He is/was a Workshop Chair/Special Session Chair/Area Chair for over 10 international conferences. He was a recipient of the National 1000 Young Talents Plan Program in 2015.

Venice Erin Liong received the B.S. degree from the University of the Philippines Diliman, Quezon City, Philippines, in 2010, and the M.S. degree from the Korea Advanced Institute of Science and Technology, Daejeon, South Korea, in 2013. She is currently pursuing the Ph.D. degree with the Rapid-Rich Object Search (ROSE) Laboratory, Interdisciplinary Graduate School, Nanyang Technological University, Singapore. Her research interests include computer vision and pattern recognition.

Jie Zhou (M'01–SM'04) received the B.S. and M.S. degrees from the Department of Mathematics, Nankai University, Tianjin, China, in 1990 and 1992, respectively, and the Ph.D. degree from the Institute of Pattern Recognition and Artificial Intelligence, Huazhong University of Science and Technology, Wuhan, China, in 1995. From 1995 to 1997, he served as a Post-Doctoral Fellow with the Department of Automation, Tsinghua University, Beijing, China. Since 2003, he has been a Full Professor with the Department of Automation, Tsinghua University. In recent years, he has authored over 100 papers in peer-reviewed journals and conferences. Among them, over 40 papers have been published in top journals and conferences, such as PAMI, TIP and CVPR. His current research interests include computer vision, pattern recognition, and image processing. He received the National Outstanding Youth Foundation of China Award. He is an Associate Editor of the *International Journal of Robotics and Automation* and two other journals.