Discrete Latent Factor Model for Cross-Modal Hashing

Qing-Yuan Jiang, Wu-Jun Li, *Member, IEEE*

***Abstract*—Due to its storage and retrieval efficiency, cross-modal hashing (CMH) has been widely used for cross-modal similarity search in many multimedia applications. According to the training strategy, existing CMH methods can be mainly divided into two categories: relaxation-based continuous methods and discrete methods. In general, the training of relaxation-based continuous methods is faster than discrete methods, but the accuracy of relaxation-based continuous methods is not satisfactory. On the contrary, the accuracy of discrete methods is typically better than relaxation-based continuous methods, but the training of discrete methods is very time-consuming. In this paper, we propose a novel CMH method, called discrete latent factor model based cross-modal hashing (DLFH), for cross modal similarity search. DLFH is a discrete method which can directly learn the binary hash codes for CMH. At the same time, the training of DLFH is efficient. Experiments show that DLFH can achieve significantly better accuracy than existing methods, and the training time of DLFH is comparable to that of relaxation-based continuous methods which are much faster than existing discrete methods.**

***Index Terms*—Approximate nearest neighbor, cross-modal retrieval, hashing, multimedia.**

I. Introduction

NEAREST neighbor (NN) search plays a fundamental role in many areas including machine learning, information retrieval, computer vision and so on. In many real applications, there is no need to return exact nearest neighbors for every given query and approximate nearest neighbor (ANN) is enough to achieve satisfactory performance [1]–[3]. Because ANN search might be much faster than exact NN search, ANN search has become an active research topic with a wide range of applications especially for large-scale problems [1]–[3].

Among existing ANN search methods, hashing methods have attracted much more attention due to their storage and retrieval efficiency in real applications [4]–[23]. The goal of hashing is to embed data points from the original space into a Hamming space where the similarity is preserved. More specifically, in the Hamming space, each data point will be represented as a binary code. Based on the binary code representation, the storage cost can be dramatically reduced, and furthermore we can achieve constant or sub-linear search speed which is much faster than the search speed in the original space [12], [13], [24], [25].

Early hashing methods are mainly proposed for uni-modal data to perform uni-modal similarity search. In recent years, with the explosive growing of multimedia data in real applications, multi-modal similarity search has attracted a lot of attention. For example, given a text query, a multi-modal similarity search system can return the nearest images or videos in the database. To achieve an efficient performance for large-scale problems, multi-modal hashing (MMH) has been proposed for multi-modal search [26]–[28].

All authors are with the National Key Laboratory for Novel Software Technology, Department of Computer Science and Technology, Nanjing University, Nanjing 210023, China. Wu-Jun Li is the corresponding author. E-mail: jiangqy@lamda.nju.edu.cn; liwujun@nju.edu.cn

Existing MMH methods can be divided into two major categories: multi-source hashing (MSH) [15], [29]–[31] and cross-modal hashing (CMH) [26], [27], [32]–[34]. MSH methods aim to learn binary hash codes by utilizing information from multiple modalities for each point. In other words, all these multiple modalities should be observed for all data points including the query points and those in database under MSH settings. Because it's usually difficult to observe all the modalities in many real applications, the application scenarios for MSH methods are limited. Unlike MSH methods, CMH methods usually require only one modality for a query point to perform search in a database with other modalities. The application scenarios for CMH are more flexible than those for MSH. For example, CMH can perform text-to-image or image-to-text retrieval tasks in real applications. Hence, CMH has gained more attention than MSH [26], [27].

A lot of CMH methods have been proposed in recent years. Based on whether supervised information is used or not during training procedure, existing CMH methods can be further divided into two categories: unsupervised CMH and supervised CMH. Unsupervised CMH methods directly explore data features without supervised information to learn binary codes (or hash functions). Representative unsupervised CMH methods include canonical correlation analysis iterative quantization (CCA-ITQ) [6], collective matrix factorization hashing (CMFH) [27], alternating co-quantization (ACQ) [35] and unsupervised generative adversarial cross-modal hashing (UGACH) [36]. Supervised CMH tries to learn the hash function by utilizing supervised information. As supervised CMH methods can incorporate semantic labels to mitigate the semantic gap, supervised CMH methods can achieve better accuracy than unsupervised CMH methods. Representative supervised CMH methods include multi-modal latent binary embedding (MLBE) [37], semantic correlation maximization (SCM) [26], semantics preserving hashing (SePH) [38], supervised matrix factorization hashing (SMFH) [39], binary set hashing (BSH) [40], deep cross-modal hashing (DCMH) [34], cross-modal hamming hashing (CMHH) [41], and generalized semantic preserving hashing (GSPH) [42] .

According to the training strategy, existing CMH methods can be divided into two categories: relaxation-based continuous methods and discrete methods. Hashing is essentially a discrete learning problem. To avoid the difficulty caused by discrete learning, relaxation-based continuous methods try to solve a relaxed continuous problem with some relaxation strategy. Representative continuous methods include CMFH [27], cross view hashing (CVH) [43], SCM [26], SMFH [39] and GSPH [42]. Discrete methods try to directly solve the discrete problem without continuous relaxation. Representative discrete methods include CCA-ITQ [6], ACQ [35], MLBE [37], predictable dual-view hashing (PDH) [44] and SePH [38]. In general, the training of relaxation-based continuous methods is faster than discrete methods, but the accuracy of relaxation-based continuous methods is not satisfactory. On the contrary, the accuracy of discrete methods is typically better than relaxation-based continuous methods, but the training of discrete methods is time-consuming.

In this paper, we propose a novel CMH method, called discrete latent factor model based cross-modal hashing (DLFH), for cross modal similarity search. The contributions of DLFH are outlined as follows:

- DLFH is a supervised CMH method, and in DLFH a novel discrete latent factor model is proposed to model the supervised information.
- DLFH is a discrete method which can directly learn the binary hash codes without continuous relaxation.
- A novel discrete learning algorithm is proposed for DLFH, which can be proved to be convergent. Furthermore, the implementation of DLFH is simple.
- The training (learning) of DLFH is still efficient although DLFH is a discrete method.
- Experiments on real datasets show that DLFH can achieve significantly better accuracy than existing methods, including both relaxation-based continuous methods and existing discrete methods. Experimental results also show that the training speed of DLFH is comparable to that of relaxation-based continuous methods, and is much faster than that of existing discrete methods.

The rest of this paper is organized as follows. In Section II, we briefly review the related works. In Section III, we describe the notations and problem definition. In Section IV, we present our DLFH in detail, including model formulation and learning algorithm. In Section V, we carry out experiments for evaluation on three widely used datasets. At last, we conclude the paper in Section VI.

II. Related Work

In this section, we briefly review the related works of cross-modal hashing, including continuous cross-modal hashing and discrete cross-modal hashing.

A. Continuous Cross-Modal Hashing

Continuous CMH methods usually adopt relaxation strategy for learning. More specifically, these methods adopt relaxation strategy to learn continuous representation at the first stage, and then utilize some rounding techniques to generate discrete binary codes at the second stage. Representative continuous CMH methods include CMFH [27], BSE [40], SCM [26], SMFH [39] and GSPH [42]. CMFH is an unsupervised CMH method which adopts collective matrix factorization to learn cross-view hash functions. BSE, SCM, SMFH and GSPH are supervised CMH methods. BSE tries to preserve the inter-modal and intra-modal similarity by learning two projections and taking the geometric structures of each modality into account. SCM learns two hash functions by integrating semantic labels into the learning procedure. To perform efficient training and fully utilize supervised information, SCM proposes an approximation method to avoid explicit computation of the pairwise similarity matrix. SMFH is the supervised version of CMFH which integrates supervised information into learning procedure to further improve retrieval performance. GSPH tries to design a generalized hashing framework to handle a wide range of scenarios.

One shortcoming of the continuous methods is that the relaxation procedure might result in a sub-optimal solution [45].

B. Discrete Cross-Modal Hashing

Discrete CMH methods try to directly learn binary codes without discarding discrete constraints. Representative discrete CMH methods include CCA-ITQ [46], ACQ [35], MLBE [37], PDH [44], SePH [38] and DCMH [34]. CCA-ITQ, ACQ and PDH are unsupervised CMH methods. CCA-ITQ and ACQ utilize the dimension reduction technologies, e.g., canonical correlation analysis (CCA) and neighborhood preserving embedding (NPE) [47], to embed multimodal data into one common subspace. Then, they minimize the quantization error to learn binary codes. PDH adopts max-margin theory to learn binary codes. By using an iterative optimization method based on block coordinate descent, PDH tries to maintain the predictability of the binary codes. MLBE, SePH and DCMH are supervised CMH methods. MLBE leverages a probabilistic latent factor model to learn binary codes which are devised as binary latent factors and designs an alternating learning algorithm to learn binary latent factors (i.e., binary codes). Although MLBE is a supervised CMH method, the complexity of MLBE is extremely high. Hence, the training of MLBE is extremely time-consuming and it cannot scale to large-scale datasets. SePH aims to transform semantic affinities information between training data into a probability distribution by minimizing the Kullback-Leibler divergence. Furthermore, SePH utilizes a kernel logistic regression method as an out-of-the-sample strategy to learn binary code for unseen data. SePH relaxes binary code as real-value continuous codes and imposes a quantization error term to learn binary codes. In the meantime, the time complexity of SePH is also high. DCMH is a deep learning based cross-modal hashing method which integrates feature learning and binary code learning simultaneously with a deep neural networks. The time complexity of DCMH is also high.

Typically, discrete methods can outperform continuous methods in terms of retrieval accuracy. However, high complexity makes the training of discrete supervised CMH time-consuming. To make the training practicable, discrete supervised CMH methods usually sample a subset from database during training procedure. Hence, existing discrete supervised CMH methods can't fully utilize supervised information and will deteriorate the accuracy. In summary, to fully utilize the supervised information, we need to design a scalable discrete CMH method to further improve retrieval accuracy and scalability for large-scale datasets.

III. Notations and Problem Definition

We briefly introduce the notations and problem definition in this section.

A. Notations

We use bold uppercase letters like $\mathbf{U}$ and bold lowercase letters like $\mathbf{u}$ to denote matrices and vectors, respectively. The element at the position (i,j) of matrix $\mathbf{U}$ is denoted as U_{ij}. The ith row of matrix $\mathbf{U}$ is denoted as $\mathbf{U}_{i*}$, and the jth column of matrix $\mathbf{U}$ is denoted as $\mathbf{U}_{*j}$. $\|\cdot\|_F$ and $\mathbf{U}^T$ denote the Frobenius norm of a matrix and the transpose of matrix $\mathbf{U}$ respectively. $\text{sign}(\cdot)$ is an element-wise sign function.

B. Problem Definition

Without loss of generality, we assume there exist only two modalities in the data although our DLFH can be easily adapted to more modalities. We use $\mathbf{X} = [\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_n]^T \in \mathbb{R}^{n\times d_x}$ and $\mathbf{Y} = [\mathbf{y}_1, \mathbf{y}_2, \dots, \mathbf{y}_n]^T \in \mathbb{R}^{n\times d_y}$ to denote the feature vectors of the two modalities (modality x and modality y), where d_x and d_y respectively denote the dimensionality of the feature spaces for two modalities and n is the number of training data. In particular, $\mathbf{x}_i$ and $\mathbf{y}_i$ denote the feature vectors of the two modalities for training point i, respectively. Without loss of generality, the data are assumed to be zero-centered which means $\sum_{i=1}^n \mathbf{x}_i = \mathbf{0}$ and $\sum_{i=1}^n \mathbf{y}_i = \mathbf{0}$. Here, we assume that both modalities are observed for all *training* points. However, we do not require that both modalities are observed for *query* (test) points. Hence, the setting is cross-modal. Actually, DLFH can be easily adapted to cases where some training points are with missing modalities, which will be left for future study. In this paper, we focus on supervised CMH which has shown better accuracy that unsupervised CMH [16], [17], [26], [38], [44]. In supervised CMH, besides the feature vectors $\mathbf{X}$ and $\mathbf{Y}$, we are also given a cross-modal supervised similarity matrix $\mathbf{S} \in \{0,1\}^{n\times n}$. If $S_{ij} = 1$, it means that point $\mathbf{x}_i$ and point $\mathbf{y}_j$ are similar. Otherwise $\mathbf{x}_i$ and $\mathbf{y}_j$ are dissimilar. Here, we assume all elements of $\mathbf{S}$ are observed. But our DLFH can also be adapted for cases with missing elements in $\mathbf{S}$. S_{ij} can be manually labeled by users, or constructed from the labels of point i and point j.

We use $\mathbf{U}, \mathbf{V} \in \{-1,+1\}^{n\times c}$ to respectively denote the binary codes for modality x and modality y, where $\mathbf{U}_{i*}$ and $\mathbf{V}_{i*}$ respectively denote the binary hash codes of two modalities for point i and c is the length of binary code. The goal of supervised CMH is to learn the binary codes $\mathbf{U}$ and $\mathbf{V}$, which try to preserve the similarity information in $\mathbf{S}$. In other words, if $S_{ij} = 1$, the Hamming distance between $\mathbf{U}_{i*}$ and $\mathbf{V}_{j*}$ should be as small as possible and vice versa. Furthermore, we also need to learn two hash functions $h_x(\mathbf{x}_q) \in \{-1,+1\}^c$ and $h_y(\mathbf{y}_q) \in \{-1,+1\}^c$ respectively for modality x and modality y, which can compute binary hash codes for any new query point ($\mathbf{x}_q$ or $\mathbf{y}_q$) unseen in the training set.

IV. Discrete Latent Factor Model based Cross-Modal Hashing

In this section, we introduce the details of DLFH, including model formulation and learning algorithm.

A. Model Formulation

Given a binary code pair $\{\mathbf{U}_{i*}, \mathbf{V}_{j*}\}$, we define Θ_{ij} as: $\Theta_{ij} = \frac{\lambda}{c}\mathbf{U}_{i*}\mathbf{V}_{j*}^T$, where c is the code length which is pre-specified, and $\lambda > 0$ is a hyper-parameter denoting a scale factor for tuning.

By using a logistic function, we define A_{ij} as: $A_{ij} = \sigma(\Theta_{ij}) = \frac{1}{1+e^{-\Theta_{ij}}}$. Based on A_{ij}, we define the likelihood of the cross-modal similarity $\mathbf{S}$ as:

$$p(\mathbf{S}|\mathbf{U},\mathbf{V}) = \prod_{i,j=1}^n p(S_{ij}|\mathbf{U},\mathbf{V}),$$

where $p(S_{ij}|\mathbf{U},\mathbf{V})$ is defined as follows:

$$p(S_{ij}|\mathbf{U},\mathbf{V}) = \begin{cases} A_{ij}, & \text{if } S_{ij}=1, \\ 1 - A_{ij}, & \text{otherwise.} \end{cases}$$

Then the log-likelihood of $\mathbf{U}$ and $\mathbf{V}$ can be derived as follows:

$$L = \log p(\mathbf{S}|\mathbf{U},\mathbf{V}) = \sum_{i,j=1}^n \left[S_{ij}\Theta_{ij} - \log(1+e^{\Theta_{ij}})\right]$$

The model of DLFH tries to maximize the log-likelihood of $\mathbf{U}$ and $\mathbf{V}$. That is, DLFH tries to solve the following problem:

$$\begin{aligned} \max_{\mathbf{U},\mathbf{V}} L &= \log p(\mathbf{S}|\mathbf{U},\mathbf{V}) \qquad (1) \\ &= \sum_{i,j=1}^n \left[S_{ij}\Theta_{ij} - \log(1+e^{\Theta_{ij}})\right] \\ &\text{s.t. } \mathbf{U},\mathbf{V} \in \{-1,+1\}^{n\times c}, \end{aligned}$$

where $\mathbf{U}$ and $\mathbf{V}$ are constrained to be in a binary space, i.e., $\{-1,+1\}^{n\times c}$, because they are binary hash codes for learning.

We can find that maximizing the objective function in (1) exactly matches the goal of hashing. More specifically, the learned binary hash codes try to preserve the similarity information in $\mathbf{S}$.

Please note that in (1), DLFH adopts the maximum likelihood loss function. Although there exist some CMH methods [34], [41] which also use maximum likelihood loss function, none of them can utilize *discrete* latent factor model to model the supervised information. On the contrary, DLFH can directly utilize discrete latent factor model to model the supervised information and learn the binary hash codes without relaxation.

B. Learning Algorithm

Problem (1) is a discrete (binary) learning problem, which is difficult to solve. One possible solution is to relax the discrete problem to a continuous problem by discarding the discrete (binary) constraints. Similar relaxation strategies have been adopted by many existing relaxation-based continuous methods like GSPH [42], CMFH [27] and SMFH [39]. However, this relaxation may cause the solution to be sub-optimal. Hence, the search accuracy might be unsatisfactory.

In this paper, we propose a novel method to directly learn the binary codes without continuous relaxation. The two parameters $\mathbf{U}$ and $\mathbf{V}$ are learned in an alternating way. Specifically, we design an iterative learning algorithm, and in each iteration we learn one parameter with the other parameter fixed.

1) Learning $\mathbf{U}$ *with* $\mathbf{V}$ *Fixed:* We try to learn $\mathbf{U}$ with $\mathbf{V}$ fixed. Even if $\mathbf{V}$ is fixed, it is still difficult to optimize (learn) the whole $\mathbf{U}$ in one time. For example, if we simply flip the signs of each element to learn $\mathbf{U}$, the total time complexity will be $\mathcal{O}(2^{n\times c})$ which is very high. Here, we adopt a column-wise learning strategy to optimize one column (corresponds to one bit for all data points) of $\mathbf{U}$ each time with other columns fixed. The time complexity to directly learn one column is $\mathcal{O}(2^n)$ which is still high. Here, we adopt a surrogate strategy [48] to learn each column, which results in a lower time complexity of $\mathcal{O}(n^2)$. More specifically, to optimize the kth column $\mathbf{U}_{*k}$, we construct a lower bound of $L(\mathbf{U}_{*k})$ and then optimize the lower bound, which can get a closed form solution and make learning procedure simple and efficient. Moreover, the lower-bound based learning strategy can guarantee the solution to converge. The following content will introduce the details about the derivation of column-wise learning strategy.

The gradient and Hessian of the objective function L with respect to $\mathbf{U}_{*k}$ can be computed as follows [1]:

$$\begin{cases} \dfrac{\partial L}{\partial \mathbf{U}_{*k}} = \dfrac{\lambda}{c}\displaystyle\sum_{j=1}^{n}(\mathbf{S}_{*j}-\mathbf{A}_{*j})V_{jk}, \\ \dfrac{\partial^2 L}{\partial \mathbf{U}_{*k}\partial \mathbf{U}_{*k}^T} = -\dfrac{\lambda^2}{c^2}\mathrm{diag}(a_1,a_2,\cdots,a_n), \end{cases}$$

where $\mathbf{A} = [A_{ij}]_{i,j=1}^n$, $a_i = \sum_{j=1}^n A_{ij}(1-A_{ij})$, and $\mathrm{diag}(a_1,a_2,\cdots,a_n)$ denotes a diagonal matrix with the ith diagonal element being a_i.

Let $\mathbf{U}_{*k}(t)$ denote the value of $\mathbf{U}_{*k}$ at the t-th iteration, $\frac{\partial L}{\partial \mathbf{U}_{*k}}(t)$ denote the gradient with respect to $\mathbf{U}_{*k}(t)$, and $\mathbf{H} = -\frac{n\lambda^2}{4c^2}\mathbf{I}$ where $\mathbf{I}$ is an identity matrix. We define $\widetilde{L}(\mathbf{U}_{*k})$ as follows:

$$\begin{aligned}\widetilde{L}(\mathbf{U}_{*k}) =& L(\mathbf{U}_{*k}(t)) + [\mathbf{U}_{*k}-\mathbf{U}_{*k}(t)]^T\frac{\partial L}{\partial \mathbf{U}_{*k}}(t) \\ &+ \frac{1}{2}[\mathbf{U}_{*k}-\mathbf{U}_{*k}(t)]^T\mathbf{H}[\mathbf{U}_{*k}-\mathbf{U}_{*k}(t)], \\ =& \mathbf{U}_{*k}^T[\frac{\partial L}{\partial \mathbf{U}_{*k}}(t) - \mathbf{H}\mathbf{U}_{*k}(t)] - [\mathbf{U}_{*k}(t)]^T\frac{\partial L}{\partial \mathbf{U}_{*k}}(t) \\ &+ L(\mathbf{U}_{*k}(t)) + \frac{[\mathbf{U}_{*k}(t)]^T\mathbf{H}\mathbf{U}_{*k}(t)}{2} + \frac{\mathbf{U}_{*k}^T\mathbf{H}\mathbf{U}_{*k}}{2} \\ =& \mathbf{U}_{*k}^T[\frac{\partial L}{\partial \mathbf{U}_{*k}}(t) - \mathbf{H}\mathbf{U}_{*k}(t)] - [\mathbf{U}_{*k}(t)]^T\frac{\partial L}{\partial \mathbf{U}_{*k}}(t) \\ &+ L(\mathbf{U}_{*k}(t)) + \frac{[\mathbf{U}_{*k}(t)]^T\mathbf{H}\mathbf{U}_{*k}(t)}{2} - \frac{\lambda^2 n^3}{8c^2} \\ =& \mathbf{U}_{*k}^T\Big[\frac{\partial L}{\partial \mathbf{U}_{*k}}(t) - \mathbf{H}\mathbf{U}_{*k}(t)\Big] + const \end{aligned} \tag{2}$$

where $const$ denotes the constant term $L(\mathbf{U}_{*k}(t)) - \frac{\lambda^2 n^3}{8c^2} + \frac{1}{2}[\mathbf{U}_{*k}(t)]^T\mathbf{H}\mathbf{U}_{*k}(t) - [\mathbf{U}_{*k}(t)]^T\frac{\partial L}{\partial \mathbf{U}_{*k}}(t)$ which is independent of the variable $\mathbf{U}_{*k}$.

Theorem 1. *$\widetilde{L}(\mathbf{U}_{*k})$ is a lower bound of $L(\mathbf{U}_{*k})$.*

To prove Theorem 1, we first introduce the following Lemma.

Lemma 1. *For a concave function $f(\mathbf{x})$ with bounded curvature, if the Hessian $\frac{\partial^2 f(\mathbf{x})}{\partial\mathbf{x}\partial\mathbf{x}^T}$ satisfies $\frac{\partial^2 f(\mathbf{x})}{\partial\mathbf{x}\partial\mathbf{x}^T} \succ \mathbf{D}$ for some matrix $\mathbf{D} \prec \mathbf{0}$, we have:*

$$f(\mathbf{y}) \geq f(\mathbf{x}) + (\mathbf{y}-\mathbf{x})^T\frac{\partial f(\mathbf{x})}{\partial \mathbf{x}} + \frac{1}{2}(\mathbf{y}-\mathbf{x})^T\mathbf{D}(\mathbf{y}-\mathbf{x}).$$

Here, $\mathbf{D} \prec \mathbf{0}$ denotes that $\mathbf{D}$ is a negative definite matrix, and $\mathbf{B} \succ \mathbf{C}$ indicates that $\mathbf{B}-\mathbf{C}$ is a positive definite matrix.

The proof for Lemma 1 can be found in [48], [49].

Based on Lemma 1, we can prove Theorem 1.

Proof of Theorem 1. It's easy to see that the $L(\mathbf{U}_{*k})$ is a concave function with bounded curvature. Furthermore, because $0 < A_{ij} < 1$, we have $0 < A_{ij}(1-A_{ij}) < \frac{1}{4}$. Then we can get: $\frac{\partial^2 L}{\partial \mathbf{U}_{*k}\partial \mathbf{U}_{*k}^T} \succ \mathbf{H}$.

According to Lemma 1, we can get: $L(\mathbf{U}_{*k}) \geq \widetilde{L}(\mathbf{U}_{*k})$. This means $\widetilde{L}(\mathbf{U}_{*k})$ is a lower bound of $L(\mathbf{U}_{*k})$. □

Then, we learn the column $\mathbf{U}_{*k}$ by maximizing the lower bound $\widetilde{L}(\mathbf{U}_{*k})$:

$$\begin{aligned}\max_{\mathbf{U}_{*k}} \ \widetilde{L}(\mathbf{U}_{*k}) =& \mathbf{U}_{*k}^T\Big[\frac{\partial L}{\partial \mathbf{U}_{*k}}(t) - \mathbf{H}\mathbf{U}_{*k}(t)\Big] + const, \\ \text{s.t. } & \mathbf{U}_{*k} \in \{-1,+1\}^n. \end{aligned} \tag{3}$$

$\forall l, U_{lk}$ is defined over $\{-1,+1\}$. Hence, to maximize $\widetilde{L}(\mathbf{U}_{*k})$, we only need to take $U_{lk}=1$ whenever the l-th element of $\big[\frac{\partial L}{\partial \mathbf{U}_{*k}}(t) - \mathbf{H}\mathbf{U}_{*k}(t)\big]$ is greater than 0, and $U_{lk}=-1$ otherwise. That is to say, the optimal solution for Problem (3) is: $\mathbf{U}_{*k} = \mathrm{sign}[\frac{\partial L}{\partial \mathbf{U}_{*k}}(t) - \mathbf{H}\mathbf{U}_{*k}(t)]$. And we use this solution to get $\mathbf{U}_{*k}(t+1)$:

$$\mathbf{U}_{*k}(t+1) = \mathrm{sign}[\frac{\partial L}{\partial \mathbf{U}_{*k}}(t) - \mathbf{H}\mathbf{U}_{*k}(t)]. \tag{4}$$

[1]Please note that the objective function L is defined on the whole real space although the variables $\mathbf{U}$ and $\mathbf{V}$ are constrained to be discrete. Hence, we can still compute the gradient and Hessian for any discrete points $\mathbf{U}$ and $\mathbf{V}$.

Algorithm 1 Learning algorithm for DLFH

```
Input: $\mathbf{S} \in \{1,0\}^{n\times n}$: supervised similarity matrix,
    $c$: code length.
Output: $\mathbf{U}$ and $\mathbf{V}$: binary codes for two modalities.
1: Procedure: initialize $\mathbf{U}$ and $\mathbf{V}$.
2: for $t = 1 \to T$ do
3:     for $k = 1 \to c$ do
4:         Update $\mathbf{U}_{*k}$ according to (4).
5:     end for
6:     for $k = 1 \to c$ do
7:         Update $\mathbf{V}_{*k}$ according to (5).
8:     end for
9: end for
```

2) Learning $\mathbf{V}$ *with* $\mathbf{U}$ *Fixed:* When $\mathbf{U}$ is fixed, we adopt a similar strategy as that for $\mathbf{U}$ to learn $\mathbf{V}$. Specifically, we can get the following closed form solution to update $\mathbf{V}_{*k}$:

$$\mathbf{V}_{*k}(t+1) = \text{sign}[\frac{\partial L}{\partial \mathbf{V}_{*k}}(t) - \mathbf{H}\mathbf{V}_{*k}(t)], \tag{5}$$

where $\frac{\partial L}{\partial \mathbf{V}_{*k}} = \frac{\lambda}{c}\sum_{i=1}^{n}(\mathbf{S}_{i*}^T - \mathbf{A}_{i*}^T)U_{ik}$ and $\mathbf{H} = -\frac{n\lambda^2}{4c^2}\mathbf{I}$.

The learning algorithm for DLFH is summarized in Algorithm 1, which can be easily implemented.

Please note that LFH [50] has adopted latent factor model for hashing. However, DLFH is different from LFH and is novel due to the following reasons. Firstly, DLFH is for cross-modal supervised hashing but LFH is for uni-modal supervised hashing. Secondly, DLFH is a discrete method which directly learns discrete (binary) codes without relaxation, but LFH is a relaxation-based continuous method which cannot directly learn the discrete codes. Last but not least, LFH learns all bits of a point each time, but DLFH learns one bit of all points each time which can lead to highly uncorrelated hash codes.

Theorem 2. *The learning algorithm for DLFH which is shown in Algorithm 1 is convergent.*

Proof of Theorem 2. According to Theorem 1, we have:

$$L(\mathbf{U}_{*k}) \geq \widetilde{L}(\mathbf{U}_{*k}). \tag{6}$$

Furthermore, since we use the optimal solution of Problem (3) to get $\mathbf{U}_{*k}(t+1)$, we have:

$$\widetilde{L}(\mathbf{U}_{*k}(t+1)) \geq \widetilde{L}(\mathbf{U}_{*k}(t)). \tag{7}$$

We can also find that $\widetilde{L}(\mathbf{U}_{*k}(t)) = L(\mathbf{U}_{*k}(t))$. Then, we have:

$$L(\mathbf{U}_{*k}(t+1)) \geq \widetilde{L}(\mathbf{U}_{*k}(t+1)) \geq \widetilde{L}(\mathbf{U}_{*k}(t)) = L(\mathbf{U}_{*k}(t)).$$

Hence, during the learning procedure, we can always guarantee that $L(\mathbf{U}_{*k}(t+1)) \geq L(\mathbf{U}_{*k}(t))$. Similarly, we can also guarantee that $L(\mathbf{V}_{*k}(t+1)) \geq L(\mathbf{V}_{*k}(t))$. This means that we can always guarantee that the objective function will not decrease during the whole learning procedure in Algorithm 1.

Together with the fact that $L(\mathbf{U},\mathbf{V})$ is upper-bounded by 0, we can guarantee that Algorithm 1 will converge. Because $L(\mathbf{U},\mathbf{V})$ is non-concave[2] in both $\mathbf{U}$ and $\mathbf{V}$, the learned solution will converge to a local optimum. □

3) Stochastic Learning Strategy: We can find that the computational cost for learning $\mathbf{U}_{*k}$ and $\mathbf{V}_{*k}$ is $\mathcal{O}(n^2)$. DLFH will become intractable when the size of training set is large. Here, we design a stochastic learning strategy to avoid high computational cost.

It is easy to see that the high computational cost mainly comes from the gradient computation for both $\mathbf{U}_{*k}$ and $\mathbf{V}_{*k}$. In our stochastic learning strategy, we randomly sample m columns (rows) of $\mathbf{S}$ to compute $\frac{\partial L}{\partial \mathbf{U}_{*k}}$ ($\frac{\partial L}{\partial \mathbf{V}_{*k}}$) during each iteration. Then, we get the following formulas for updating $\mathbf{U}_{*k}$ and $\mathbf{V}_{*k}$:

$$\mathbf{U}_{*k}(t+1) = \text{sign}\big[\frac{\lambda}{c}\sum_{q=1}^{m}(\mathbf{S}_{*j_q} - \mathbf{A}_{*j_q})V_{j_q k}(t) + \frac{m\lambda^2}{4c^2}\mathbf{U}_{*k}(t)\big], \tag{8}$$

$$\mathbf{V}_{*k}(t+1) = \text{sign}\big[\frac{\lambda}{c}\sum_{q=1}^{m}(\mathbf{S}_{i_q *}^T - \mathbf{A}_{i_q *}^T)U_{i_q k}(t) + \frac{m\lambda^2}{4c^2}\mathbf{V}_{*k}(t)\big], \tag{9}$$

where $\{j_q\}_{q=1}^m$ and $\{i_q\}_{q=1}^m$ are the m sampled column and row indices, respectively.

To get the stochastic learning algorithm for DLFH, we only need to substitute (4) and (5) in Algorithm 1 by (8) and (9), respectively. Then the computational cost will decrease from $\mathcal{O}(n^2)$ to $\mathcal{O}(nm)$, where $m \ll n$.

C. Out-of-Sample Extension

For any unseen query points $\mathbf{x}_q \notin \mathbf{X}$ or $\mathbf{y}_q \notin \mathbf{Y}$, we need to perform out-of-sample extension to generate the binary codes. Many existing hashing methods learn classifiers to predict binary codes for out-of-sample extension. These classifiers include SVM [24], boosted decision tree [8], kernel logistic regression [38], and so on. In this paper, we adopt two classifiers for out-of-sample extension, one being linear and the other being non-linear. We use the modality x as an example to demonstrate these two strategies.

For linear classifier, we minimize the following square loss:

$$L_{\text{square}}(\mathbf{W}^{(x)}) = \|\mathbf{U} - \mathbf{X}\mathbf{W}^{(x)}\|_F^2 + \gamma_x\|\mathbf{W}^{(x)}\|_F^2,$$

where γ_x is the hyper-parameter for regularization term. We can get: $\mathbf{W}^{(x)} = (\mathbf{X}^T\mathbf{X} + \gamma_x\mathbf{I})^{-1}\mathbf{X}^T\mathbf{U}$.

Then we can get the hash function for out-of-sample extension: $h_x(\mathbf{x}_q) = \text{sign}([\mathbf{W}^{(x)}]^T\mathbf{x}_q)$.

For non-linear classifier, we adopt kernel logistic regression (KLR) as that in SePH [38]. Specifically, we learn c classifiers, one classifier for a bit, to predict binary codes for $\mathbf{x}_q$. For the kth bit, we minimize the following KLR loss:

$$L_{\text{KLR}}(\mathbf{M}_{*k}^{(x)}) = \sum_{i=1}^{n}\log(1 + e^{-U_{ik}\phi(\mathbf{x}_i)^T\mathbf{M}_{*k}^{(x)}}) + \eta_x\|\mathbf{M}_{*k}^{(x)}\|_F^2,$$

where η_x is the hyper-parameter for regularization term and $\phi(\mathbf{x}_i)$ a RBF kernel feature representation for $\mathbf{x}_i$. Then we can get the hash function for out-of-sample extension:

$$h_x(\mathbf{x}_q) = \text{sign}([\mathbf{M}^{(x)}]^T\phi(\mathbf{x}_i)),$$

[2]Please note that $L(\cdot)$ is concave in $\mathbf{U}_{*k}$ or $\mathbf{V}_{*k}$, but non-concave in both $\mathbf{U}$ and $\mathbf{V}$.

TABLE I
EXAMPLE POINTS FROM THREE DATASETS.

Dataset	IAPR-TC12	MIRFLICKR-25K	NUS-WIDE
Image example			
Text information	cascading, waterfall, middle, jungle, pool, dirty, water, foreground.	explore, beach, people, sea, summer, boat, jump, playa, mar, coast, gente, shore.	sunset, night, architecture, city, building, lake, lights, colorful, buildings, tower, america, evening, downtown, work, chicago, office, great, cityscape, streets, illinois, business.
Label	sky-light, trees, waterfall, water, vegetation.	male, people, sea, sky, sunset, transport, water.	sky, buildings, lake.

where $\mathbf{M}_{*k}^{(x)}$ is the kth column of $\mathbf{M}^{(x)}$.

We can use similar ways to learn the out-of-sample classifiers for the y modality. We can also use other out-of-sample classifiers in our method, but this is not the focus of this paper. In the following content, we use "DLFH" and "KDLFH" to denote the linear and non-linear versions for out-of-sample extension, respectively.

D. Complexity Analysis

We call the DLFH version without stochastic learning strategy *DLFH-Full*, and call the stochastic version of DLFH *DLFH-Stochastic*. The time complexity of DLFH-Full is $\mathcal{O}(Tcn^2)$, where T and c are typically small constants. The time complexity of DLFH-Stochastic is $\mathcal{O}(Tcnm)$, which is much lower than that of the DLFH-Full when $m \ll n$. In practice, we suggest to adopt the DLFH-Stochastic because in our experiment we find that the accuracy of DLFH-Stochastic is comparable to that of the DLFH-Full even if m is set to be c which is typically a small constant. When $m = c$, the training of DLFH-Stochastic is very efficient.

V. EXPERIMENTS

We utilize three widely used datasets for evaluation. Our DLFH and KDLFH are non-deep methods. The experiments for non-deep baselines are performed on a workstation with Intel (R) CPU E5-2620V2@2.1G of 12 cores and 96G RAM. For this platform, the operating system is CentOS 6.5. For deep CMH baselines, we conduct our experiments on a workstation with Intel (R) CPU E5-2860V4@2.4G of 14 cores, 768G RAM and a NVIDIA M40 GPU[3]. For this platform, the operating system is Ubuntu 14.04.

A. Datasets

Three datasets, IAPR-TC12 [51], MIRFLICKR-25K [52] and NUS-WIDE [53], are used for evaluation.

The IAPR-TC12 dataset consists of 20,000 image-text pairs which are annotated using 255 labels. We use the entire dataset for our experiment. Each text is represented as a 2912-D bag-of-words (BOW) vector. Each image is represented by a 512-D GIST feature vector.

The MIRFLICKR-25K dataset contains 25,000 data points. Each point corresponds to an image associated with some textual tags. We only select those points which have at least 20 textual tags for our experiment. Each text is represented as a 1386-D BOW vector. Each image is represented by a 512-D GIST feature vector. Each point is manually annotated with some of the 24 unique labels.

The NUS-WIDE dataset contains 260,648 data points, each of which corresponds to an image associated with some textual tags. Furthermore, each point is annotated with one or multiple labels from 81 concept labels. We only select 186,577 points that belong to the 10 most frequent concepts from the original dataset. The text for each point is represented as a 1000-D BOW vector. Furthermore, each image is a 500-D bag-of-visual words (BOVW) vector.

For all three datasets, the ground-truth neighbors (similar pairs) are defined as those image-text pairs which share at least one common label. Table I illustrates some example points from three datasets.

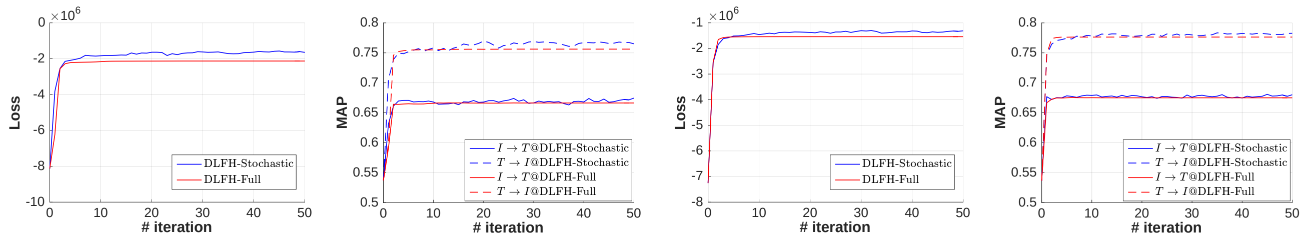
B. Baselines and Evaluation Protocol

We adopt nine state-of-the-art CMH methods as baselines for comparison. They are CMHH [41], UGACH [36], DCMH [34], GSPH [42], SePH [38], SCM [26], CMFH [27], CCA-ITQ [6] and MLBE [37]. Among these baselines, UGACH, CCA-ITQ and CMFH are unsupervised, and others are supervised. CMFH, SCM and GSPH are relaxation-based continuous methods, others are discrete methods. CMHH, UGACH and DCMH are deep learning based methods, others are non-deep methods. For DCMH, GSPH, SePH, SCM and MLBE, the source codes are kindly provided by their corresponding authors. For the other baselines, we implement them carefully by ourselves. GSPH and SePH are kernel-based methods. Following their authors' suggestion, we utilize RBF kernel and adopt two strategies to create kernel bases to learn hash functions for SePH. Specifically, one strategy is randomly taking 500 data points as kernel bases and the other strategy is to use k-means algorithm to learn 500 centers as kernel

[3]When we compare our DLFH and KDLFH with deep CMH baselines in terms of time cost, DLFH and KDLFH are also performed on the same GPU platform.

TABLE II
DETAILED STATISTICS OF THREE DATASETS.

Dataset	#Total	#Query	#Database	Image feature	Text feature
IAPR-TC12	20,000	2,000	18,000	512-D GIST	2912-D BOW
MIRFLICKR-25K	20,015	2,000	18,015	512-D GIST	1386-D BOW
NUS-WIDE	186,577	1,867	184,710	500-D BOVW	1000-D BOW



(a) Obj. value@32 bits (b) MAP@32 bits (c) Obj. value@64 bits (d) MAP@64 bits

Fig. 1. Objective function value and MAP of DLFH on subset of MIRFLICKR-25K.

bases. These two versions of SePH are denoted as SePH$_{rnd}$ and SePH$_{km}$. For GSPH, we use k-means algorithm to learn 500 centers as kernel bases. For the other baselines, we set the parameters by following suggestions of the corresponding authors. For our DLFH, we set $\lambda = 8$ and $T = 30$ which is selected based on a validation set. We adopt the stochastic version of DLFH unless otherwise stated. In stochastic DLFH, $m = c$. To initialize $\mathbf{U}$ and $\mathbf{V}$, we first use a uniform distribution to randomly generate the elements in the matrices, and then use sign$(\cdot)$ function to map them to $\{\pm 1\}$. We find that our DLFH is not sensitive to initialization. For KDLFH, we randomly take 500 data points as kernel bases like SePH$_{rnd}$. All experiments are run 5 times to remove randomness, then the average accuracy with standard deviations is reported.

For IAPR-TC12 and MIRFLICKR-25K datasets, we randomly select 2,000 data points as query (test) set and the rest as retrieval (database) set. For NUS-WIDE dataset, we randomly select 1,867 data points as a query set and the rest as retrieval set. For our DLFH and most baselines except GSPH, SePH and MLBE, we use the whole retrieval set as training set. Because GSPH, SePH and MLBE can not scale to large training set, we randomly sample 5,000 data points from retrieval set to construct training set for GSPH and SePH, and sample 1,000 data points for training MLBE due to an out-of-memory error with larger training set. Table II presents the detailed statistics of three datasets.

As existing methods [6], Hamming ranking and hash lookup are adopted as retrieval protocols for evaluation. We report Mean Average Precision (MAP) and Top-k precision for the Hamming ranking protocol. The precision-recall curve is the widely used metric to measure the accuracy of the hash lookup protocol and we report the precision-recall for evaluation.

C. Convergence Analysis

To verify the convergence property of DLFH, we conduct an experiment on a subset of MIRFLICKR-25K dataset. Specifically, we randomly select 5,000 data points from MIRFLICKR-25K dataset, with 3,000 data points for training and the rest for test (query).

Figure 1 shows the convergence of objective function value (Figure 1 (a) and (c)) and MAP (Figure 1 (b) and (d)), where “DLFH-Full” denotes the full version of DLFH and “DLFH-Stochastic” denotes the stochastic version of DLFH. In the figure, “$I \to T$” denotes image-to-text retrieval where we use image modality as queries and then retrieve text from database, and other notations are defined similarly. We can find that the objective function value of DLFH-Full will not decrease as iteration number increases, which verifies the claim about the convergence in Theorem 2. There is some vibration in DLFH-Stochastic due to the stochastic sampling procedure, but the overall trend is convergent. For MAP, we can observe the convergence for overall trend. Both DLFH-Full and DLFH-Stochastic converge very fast, and only a small number of iterations are needed.

Another interesting phenomenon is that the accuracy of DLFH-Stochastic is almost the same as that of DLFH-Full. Hence, unless otherwise stated, the DLFH in the following experiment refers to the stochastic version of DLFH.

D. Hamming Ranking Task

Table III, Table IV and Table V presents the MAP and Top100 precision on IAPR-TC12, MIRFLICKR-25K and NUS-WIDE datasets, respectively. The best accuracy is shown in boldface. Because DLFH and KDLFH are non-deep methods, here we only compare them with non-deep baselines, including GSPH, SePH, SCM, CMFH, CCA-ITQ and MLBE. The detailed comparison with deep learning-based baselines will be separately shown in Section V-G.

By comparing GSPH, SePH$_{rnd}$, SePH$_{km}$, SCM to CMFH and CCA-ITQ, we can see that supervised methods can outperform unsupervised methods in most cases. By comparing KDLFH, DLFH, SePH$_{rnd}$ and SePH$_{km}$ to other methods, we can find that in general discrete methods can outperform relaxation-based continuous methods. Please note that MLBE is a special case because the training of MLBE is too slow and we have to sample only a very small subset for training. Hence, its accuracy is low although it is supervised and discrete. We can find that our DLFH can significantly

TABLE III
MAP ($mean \pm std$, IN PERCENT) AND TOP100 PRECISION ($mean \pm std$, IN PERCENT) ON IAPR-TC12 DATASET WITH 8, 16, 32 AND 64 BITS.

Task	Method	MAP				Top100 precision			
		8 bits	16 bits	32 bits	64 bits	8 bits	16 bits	32 bits	64 bits
$I \to T$	DLFH	**43.07±1.85**	**45.58±1.33**	**52.00±0.75**	**54.84±0.57**	**54.62±1.98**	**57.94±1.93**	**66.60±0.74**	**70.05±0.96**
	SCM	37.67±0.00	38.74±0.00	39.72±0.00	41.02±0.00	45.85±0.00	49.40±0.00	50.94±0.00	52.73±0.00
	CMFH	30.92±0.81	30.91±0.62	31.70±1.16	30.76±0.53	31.57±1.10	31.27±0.61	32.51±2.04	31.31±0.94
	CCA-ITQ	35.35±0.00	34.17±0.00	33.09±0.00	32.36±0.00	44.03±0.00	42.31±0.00	39.51±0.00	37.50±0.00
	MLBE	30.39±0.01	30.52±0.16	30.36±0.04	30.46±0.09	31.16±0.00	30.79±0.52	30.30±0.72	30.50±0.24
	KDLFH	**43.26±1.42**	**47.02±1.11**	**52.61±0.65**	**56.54±1.00**	**54.57±1.66**	**60.51±2.11**	**67.04±1.52**	**71.04±1.21**
	SePH$_{rnd}$	40.30±0.07	41.47±0.22	42.27±0.11	42.89±0.10	45.95±0.39	48.20±0.41	49.97±0.24	51.52±0.30
	SePH$_{km}$	40.30±0.08	41.41±0.26	42.07±0.12	42.65±0.10	46.22±0.14	48.22±0.60	49.99±0.36	51.27±0.24
	GSPH	38.67±0.41	40.03±0.22	41.67±0.16	42.85±0.21	45.43±0.44	48.79±0.23	51.27±0.40	53.42±0.23
$T \to I$	DLFH	**43.84±1.96**	**48.47±1.30**	**57.13±0.60**	**62.43±0.41**	**55.08±1.24**	**62.22±1.56**	**71.76±0.77**	**77.65±0.62**
	SCM	37.65±0.00	38.57±0.00	39.81±0.00	40.82±0.00	44.39±0.00	47.81±0.00	51.08±0.00	53.08±0.00
	CMFH	31.02±1.17	31.67±0.94	32.42±1.15	31.71±0.36	32.33±2.19	33.24±1.27	33.67±2.56	32.79±0.94
	CCA-ITQ	35.27±0.00	34.19±0.00	33.18±0.00	32.44±0.00	44.85±0.00	42.85±0.00	40.63±0.00	38.59±0.00
	MLBE	30.39±0.00	30.44±0.09	30.35±0.11	30.49±0.12	31.16±0.00	30.30±0.93	30.09±1.08	30.55±0.15
	KDLFH	**44.40±1.39**	**50.73±1.00**	**58.10±0.59**	**63.93±1.34**	**55.85±3.62**	**65.76±1.14**	**73.50±1.52**	**79.21±1.12**
	SePH$_{rnd}$	42.70±0.12	44.31±0.18	45.62±0.14	46.66±0.21	53.20±0.26	57.04±0.41	60.60±0.22	63.38±0.70
	SePH$_{km}$	42.19±0.08	43.77±0.28	44.76±0.14	45.67±0.29	51.92±0.30	55.48±0.92	58.49±0.48	60.72±0.56
	GSPH	40.01±0.61	42.30±0.28	44.77±0.26	46.62±0.27	49.14±1.09	55.91±0.25	60.72±0.63	64.40±0.41

TABLE IV
MAP ($mean \pm std$, IN PERCENT) AND TOP100 PRECISION ($mean \pm std$, IN PERCENT) ON MIRFLICKR-25K DATASET WITH 8, 16, 32 AND 64 BITS.

Task	Method	MAP				Top100 precision			
		8 bits	16 bits	32 bits	64 bits	8 bits	16 bits	32 bits	64 bits
$I \to T$	DLFH	**72.68±1.11**	**75.84±0.46**	**78.02±0.07**	**78.92±0.03**	**78.25±2.26**	**82.94±0.93**	**85.37±0.90**	**86.13±0.46**
	SCM	62.88±0.00	63.89±0.00	65.06±0.00	65.76±0.00	67.41±0.00	69.59±0.00	70.12±0.00	71.01±0.00
	CMFH	56.98±1.02	57.38±0.59	57.32±0.22	56.87±0.22	58.72±1.44	59.76±0.91	60.35±0.60	59.43±0.74
	CCA-ITQ	58.33±0.00	57.62±0.00	57.11±0.00	56.77±0.00	62.26±0.00	60.80±0.00	59.67±0.00	58.72±0.00
	MLBE	55.75±0.05	55.81±0.14	55.83±0.15	55.83±0.15	57.34±0.71	56.20±0.85	55.79±2.45	58.04±2.22
	KDLFH	**74.28±0.67**	**77.07±0.79**	**79.27±0.24**	**80.08±0.19**	**79.02±0.24**	**83.75±1.81**	**85.99±0.57**	**86.92±0.45**
	SePH$_{rnd}$	65.16±0.16	65.60±0.17	65.96±0.09	66.23±0.08	67.05±0.41	67.64±0.13	68.70±0.36	69.40±0.39
	SePH$_{km}$	65.51±0.21	66.06±0.16	66.40±0.22	66.66±0.10	67.81±0.57	68.38±0.45	68.99±0.53	69.87±0.19
	GSPH	63.12±0.65	64.62±0.35	65.77±0.32	66.40±0.31	67.81±0.47	69.44±0.32	70.29±0.25	70.59±0.40
$T \to I$	DLFH	**77.73±0.88**	**82.39±0.29**	**85.01±0.17**	**86.05±0.21**	**82.85±2.03**	**87.95±0.69**	**90.50±0.31**	**91.28±0.19**
	SCM	61.62±0.00	62.37±0.00	63.37±0.00	64.25±0.00	65.08±0.00	67.97±0.00	69.53±0.00	71.38±0.00
	CMFH	57.07±0.72	57.52±0.46	57.71±0.20	57.69±0.07	57.09±1.74	57.49±0.57	60.20±0.24	60.18±0.27
	CCA-ITQ	58.22±0.00	57.59±0.00	57.12±0.00	56.82±0.00	62.65±0.00	60.85±0.00	59.93±0.00	59.03±0.00
	MLBE	55.73±0.08	55.79±0.12	55.79±0.10	56.04±0.22	57.01±1.25	55.54±0.87	55.57±0.70	56.05±0.59
	KDLFH	**79.08±1.40**	**82.14±0.57**	**85.02±0.08**	**85.83±0.24**	**85.15±1.91**	**88.63±1.05**	**91.37±0.49**	**92.46±0.33**
	SePH$_{rnd}$	68.11±0.22	68.66±0.27	69.42±0.28	69.79±0.42	73.98±0.55	75.47±0.37	77.00±0.33	78.15±0.34
	SePH$_{km}$	69.15±0.24	69.72±0.30	70.39±0.36	70.83±0.20	75.98±0.40	76.65±0.84	78.28±0.71	79.62±0.43
	GSPH	66.50±0.50	68.47±0.40	69.93±0.37	71.21±0.30	76.73±0.48	79.12±0.38	80.18±0.27	81.53±0.40

TABLE V
MAP ($mean \pm std$, IN PERCENT) AND TOP100 PRECISION ($mean \pm std$, IN PERCENT) ON NUS-WIDE DATASET WITH 8, 16, 32 AND 64 BITS.

Task	Method	MAP				Top100 precision			
		8 bits	16 bits	32 bits	64 bits	8 bits	16 bits	32 bits	64 bits
$I \to T$	DLFH	**63.82±0.61**	**67.00±0.45**	**69.27±0.21**	**70.33±0.25**	**74.61±1.13**	**78.91±0.88**	**82.07±0.80**	**84.03±0.82**
	SCM	51.29±0.00	52.19±0.00	54.81±0.00	55.58±0.00	53.38±0.00	57.02±0.00	60.90±0.00	61.96±0.00
	CMFH	36.31±1.53	38.12±1.08	38.62±1.36	40.03±0.81	38.23±1.36	41.23±1.13	43.08±1.42	44.03±0.92
	CCA-ITQ	40.97±0.00	39.55±0.00	38.23±0.00	37.03±0.00	52.24±0.00	51.89±0.00	50.05±0.00	46.44±0.00
	MLBE	34.56±0.00	34.51±0.00	34.46±0.00	34.86±0.08	35.07±0.00	33.84±0.00	33.84±0.00	40.77±3.65
	KDLFH	**65.24±1.03**	**68.47±0.65**	**70.19±0.15**	**71.12±0.29**	**76.84±1.18**	**79.91±0.67**	**82.49±0.63**	**84.31±0.45**
	SePH$_{rnd}$	52.67±0.47	53.96±0.87	54.85±0.25	55.24±0.42	55.00±0.30	55.89±0.31	56.52±0.24	56.81±0.24
	SePH$_{km}$	53.36±0.50	54.41±0.57	55.54±0.41	55.64±0.41	55.51±0.21	56.32±0.41	56.56±0.38	57.09±0.35
	GSPH	51.28±0.75	54.19±0.40	55.29±0.25	56.02±0.22	55.15±0.21	56.66±0.74	57.89±0.19	58.27±0.16
$T \to I$	DLFH	**72.22±2.28**	**78.50±0.50**	**82.03±0.42**	**83.78±0.09**	**80.01±2.92**	**86.37±1.75**	**90.12±0.67**	**90.89±0.52**
	SCM	48.26±0.00	49.25±0.00	51.53±0.00	52.26±0.00	53.45±0.00	57.05±0.00	61.59±0.00	62.92±0.00
	CMFH	35.67±1.56	36.87±1.09	37.48±1.44	38.24±0.90	36.22±2.85	40.72±0.76	44.57±2.92	46.91±1.92
	CCA-ITQ	40.32±0.00	38.98±0.00	37.81±0.00	36.79±0.00	51.68±0.00	52.67±0.00	50.62±0.00	46.81±0.00
	MLBE	34.53±0.00	34.53±0.00	34.53±0.00	34.53±0.08	33.93±0.00	33.93±0.00	33.93±0.00	35.52±1.24
	KDLFH	**72.37±2.19**	**78.27±0.95**	**81.78±0.51**	**83.25±0.28**	**82.77±1.90**	**88.59±0.81**	**91.03±1.03**	**92.57±0.47**
	SePH$_{rnd}$	60.71±0.47	61.79±1.17	62.75±0.29	63.40±0.50	68.85±0.41	69.79±0.92	70.36±0.81	71.44±0.53
	SePH$_{km}$	61.34±0.38	62.63±0.78	64.14±0.14	64.47±0.53	69.50±0.61	70.37±0.27	71.48±0.20	72.30±0.45
	GSPH	58.60±0.86	62.69±0.47	63.95±0.39	65.13±0.34	68.83±0.71	71.58±0.90	72.91±0.27	73.81±0.42

outperform all baselines in all cases for both image-to-text and text-to-image retrieval tasks. Furthermore, KDLFH can further improve retrieval accuracy thanks to the non-linear out-of-sample extension strategy.

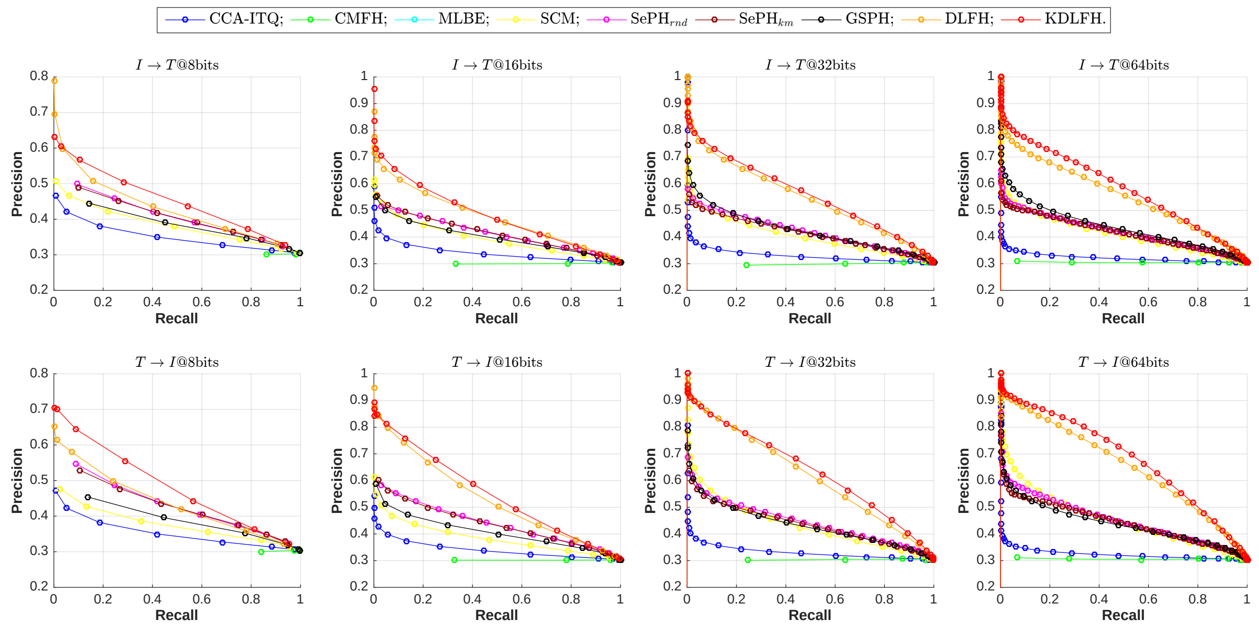


Fig. 2. Precision-recall curve on IAPR-TC12 dataset.

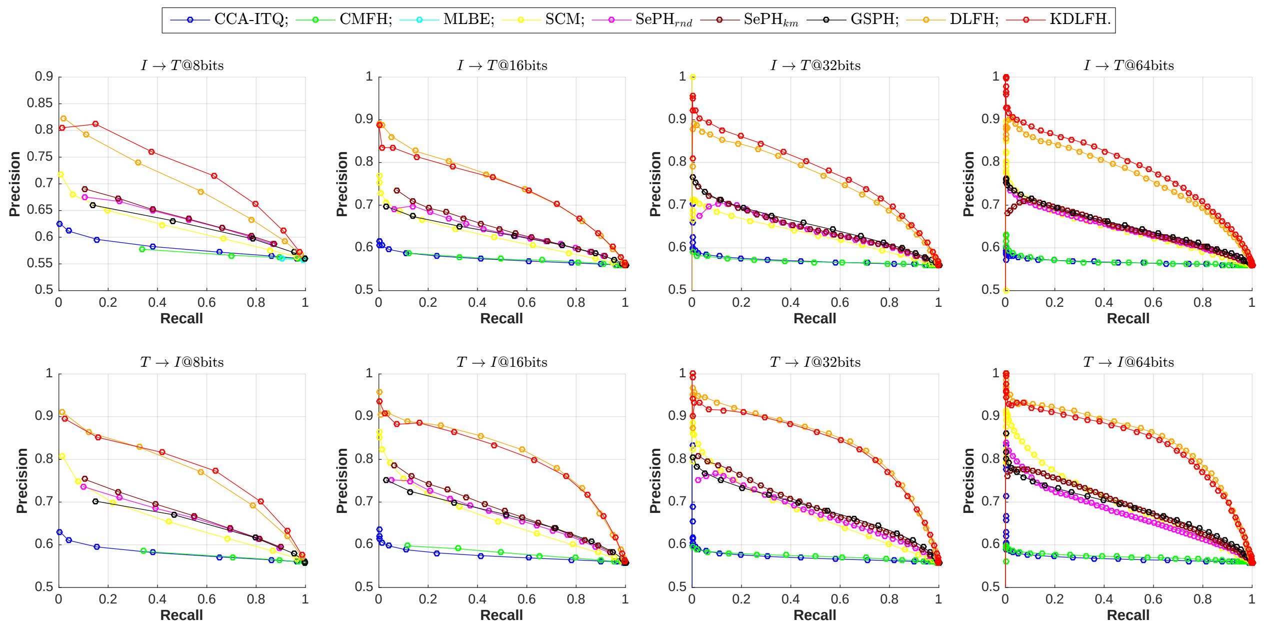


Fig. 3. Precision-recall curve on MIRFLICKR-25K dataset.

E. Hash Lookup Task

In real applications, retrieval with hash lookup can usually achieve constant or sub-linear search speed. For hash lookup protocols, we report precision-recall curves to evaluate the proposed DLFH, KDLFH and baselines on three datasets.

Figure 2, Figure 3 and Figure 4 show the precision-recall curve on IAPR-TC12, MIRFLICKR-25K and NUS-WIDE datasets, respectively. Once again, we can find that DLFH and KDLFH can significantly outperform all baselines in all cases.

F. Training Speed

To evaluate the training speed of DLFH, we adopt different number of data points from retrieval set to construct training set and then report the training time. Table VI presents the training time (in second) for our DLFH and baselines, where "–" denotes that we cannot carry out corresponding experiments due to out-of-memory errors. We can find that the unsupervised method CCA-ITQ is the fastest method because it does not use supervised information for training. Although the training of CCA-ITQ is fast, the accuracy of it is low. Hence, CCA-ITQ is not practical in real applications. By com-

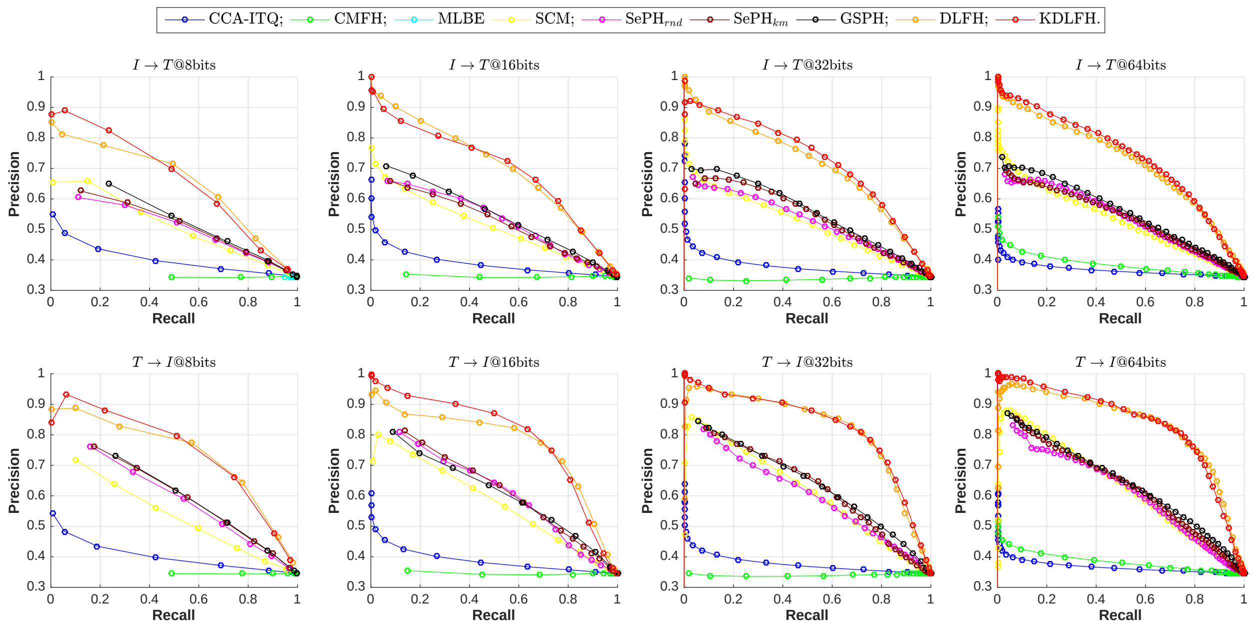


Fig. 4. Precision-recall curve on NUS-WIDE dataset.

TABLE VI
TRAINING TIME (IN SECOND) ON SUBSETS OF NUS-WIDE.

Method	1K	5K	10K	50K	~184K
DLFH	1.26	3.84	6.61	34.88	112.88
SCM	10.41	11.13	11.09	11.34	12.30
CMFH	4.20	12.01	20.65	84.00	305.51
CCA-ITQ	0.56	0.59	0.69	1.51	4.25
MLBE	998.98	–	–	–	–
KDLFH	56.58	214.65	479.46	2097.52	7341.05
SePH$_{rnd}$	80.54	606.18	–	–	–
SePH$_{km}$	83.81	705.39	–	–	–
GSPH	126.43	981.35	–	–	–

TABLE VII
COMPARISON WITH DEEP BASELINES ON IAPR-TC12 DATASET.

Method	MAP (%)		Avg.	Time
	$I \to T$	$T \to I$		
CMHH	48.81±0.46	49.46±0.24	49.14±0.11	45468.5
DCMH	45.26±0.33	51.85±0.38	48.56±0.35	33090.2
DLFH	43.77±1.27	52.67±1.52	48.22±1.33	5.85
KDLFH	50.48±0.83	49.91±1.09	**50.20±0.97**	741.75

TABLE VIII
COMPARISON WITH DEEP BASELINES ON MIRFLICKR-25K DATASET.

Method	MAP (%)		Avg.	Time
	$I \to T$	$T \to I$		
CMHH	75.37±0.13	76.84±0.22	76.10±0.32	44358.6
DCMH	74.41±0.12	78.48±0.53	76.44±0.22	33736.7
DLFH	77.94±0.98	79.00±0.68	**78.47±0.79**	4.46
KDLFH	85.91±0.72	82.03±1.28	**83.97±0.99**	777.59

paring MLBE, SePH$_{rnd}$ and SePH$_{km}$ to SCM and CMFH, we can find that existing discrete methods are much slower than relaxation-based continuous methods. Although our DLFH is a discrete method, its training speed can be comparable to relaxation-based continuous methods. Furthermore, KDLFH is also much faster than the kernel baselines SePH$_{rnd}$ and SePH$_{km}$.

Although KDLFH can achieve better accuracy than DLFH, the training speed of KDLFH is much slower than that of DLFH. Hence, in real applications, we provide users with two choices between DLFH and KDLFH based on whether they care more about training speed or accuracy.

Overall, our DLFH and KDLFH methods achieve the best accuracy with a relatively fast training speed. In particular, our methods can significantly outperform relaxation-based continuous methods in terms of accuracy, but with a comparable training speed. Furthermore, our methods can significantly outperform existing discrete methods in terms of both accuracy and training speed.

G. Comparison with Deep Baselines

There have appeared some deep learning based cross-modal hashing methods [16], [34], [36], [41]. We compare DLFH and KDLFH with some representative deep CMH methods, including CMHH, DCMH and UGACH. Please note that the main focus of this paper is on supervised hashing and UGACH is an unsupervised method. Hence, we only compare DLFH and KDLFH with UGACH on the largest dataset NUS-WIDE in terms of accuracy for demonstration. For fair comparison, we use 4096-D deep features extracted by using a pre-trained CNN model for the image modality in DLFH and KDLFH. This pre-trained CNN model is also used for initializing DCMH and CMHH. Hence, the comparison is fair. Because deep CMH methods are time-consuming, we use a subset of the whole dataset as training set by following the same strategy in [34].

The MAP and training time (in second) with 16 bits are shown in Table VII, Table VIII and Table IX on IAPR-TC12, MIRFLICKR-25K and NUS-WIDE datasets, respectively. Other cases with different number of bits have similar phenomenon. We use boldface to denote the cases

TABLE IX
COMPARISON WITH DEEP BASELINES ON NUS-WIDE DATASET.

Method	MAP (%)		Avg.	Time
	$I \rightarrow T$	$T \rightarrow I$		
UGACH	61.3	60.3	60.8	N/A
CMHH	66.82±1.02	67.35±0.84	67.08±0.91	48128.5
DCMH	64.79±0.19	68.84±0.28	66.62±0.18	35777.2
DLFH	80.67±0.77	75.76±1.68	**78.22±1.44**	36.01
KDLFH	82.32±1.23	76.04±1.99	**79.18±1.78**	6651.90

where the average accuracy is better than that of baselines. We can see that KDLFH and DLFH can outperform deep CMH methods in most cases. Furthermore, KDLFH and DLFH are more efficient than deep CMH methods in all cases, even if KDLFH and DLFH are trained on CPU while deep CMH methods are trained on GPU.

H. Comparison with LFH-extension

To verify the effectiveness of the proposed method, we extend LFH method to cross-modal hashing and adopt this method as baseline. Specifically, we utilize LFH method to learn a unified binary codes for both two modalities. Then we learn two linear functions to perform out-of-sample extension. We denote this method as LFH$_{cm}$.

We present the MAP results in Figure 5 on NUS-WIDE dataset. We can see that DLFH can outperform LFH$_{cm}$ thanks to discrete learning and bit-wise learning strategy. To verify that DLFH can learn a higher uncorrelated binary codes because of a bit-wise learning strategy, we analysis the correlation matrix $\mathbf{E}$ of the learned binary codes for DLFH and LFH. We also calculate the mean Absolute Correlation (mAC) based on the following equation: mAC $= \frac{2\sum_{i=1}^{c}\sum_{j<i}|E_{ij}|}{c(c-1)}$, where $\mathbf{E} = \{E_{ij}\}_{i,j=1}^{c} \in \mathbb{R}^{c\times c}$ is the correlation matrix.

We show the correlation matrix (absolute value) of learned binary codes $\mathbf{U}$ and $\mathbf{V}$ for DLFH in Figure 6 (a),(b). And the correlation matrix for LFH$_{cm}$ is shown in Figure 6 (c). Furthermore, we can find that DLFH can achieve lower mAC than LFH$_{cm}$. Hence, DLFH can learn more highly uncorrelated binary codes than LFH$_{cm}$.

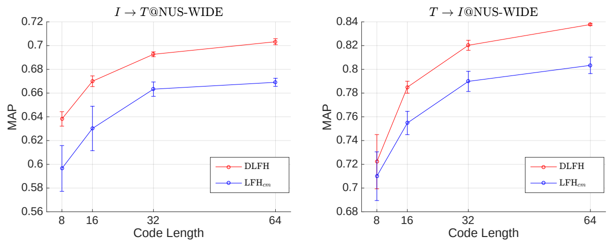
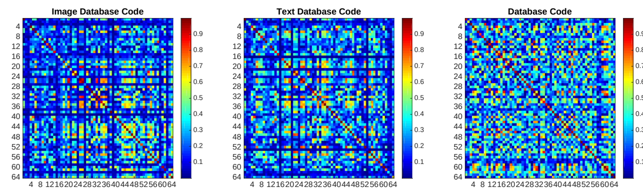


Fig. 5. Comparison with LFH$_{cm}$ on NUS-WIDE dataset.

I. Sensitivity to Hyper-Parameter

We study the influence of γ_x, η_x, λ and the number of sampled points m on IAPR-TC12, MIRFLICKR-25K and NUS-WIDE datasets.

We present the MAP values with different γ_x from the range of $[10^{-4}, 10^{3}]$ with the code length being 16 bits. The results are shown in Figure 7. We can find that DLFH is not sensitive to γ_x in a large range when $10^{-4} \leq \eta_x \leq 1$. Furthermore,



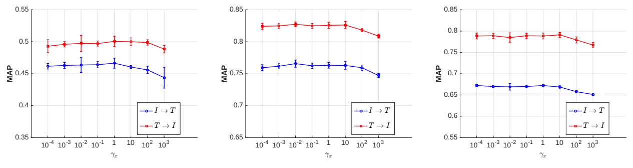
(a) DLFH, mAC= 0.223. (b) DLFH, mAC= 0.234. (c) LFH$_{cm}$, mAC= 0.296.

Fig. 6. Correlation matrix and mAC on NUS-WIDE dataset.

from Figure 8, we can find that DLFH is not sensitive to η_x in a large range when $10^{-3} \leq \eta_x \leq 1$.

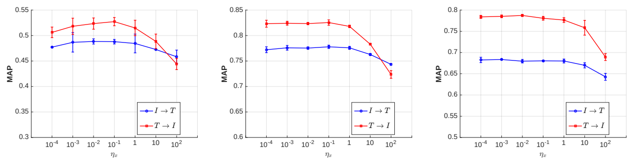
We also report the MAP values for different λ from the range of $[1, 24]$ with the code length being 16 bits. The results are shown in Figure 9. We can find that DLFH is not sensitive to λ in a large range when $4 \leq \lambda \leq 16$.

Furthermore, we present the influence of m in Figure 10. We can find that as the number of sampled points increases, the accuracy increases at the beginning and then remains unchanged. As more sampled points will require higher computation cost, we simply set $m = c$ in our experiment to get a good tradeoff between accuracy and efficiency for DLFH and KDLFH.



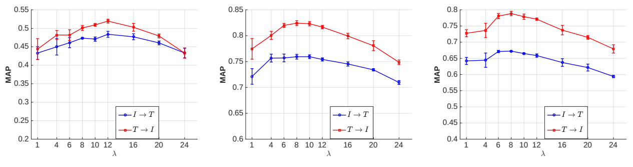
(a) IAPR-TC12. (b) MIRFLICKR-25K. (c) NUS-WIDE.

Fig. 7. MAP values with different γ_x on three datasets.



(a) IAPR-TC12. (b) MIRFLICKR-25K. (c) NUS-WIDE.

Fig. 8. MAP values with different η_x on three datasets.

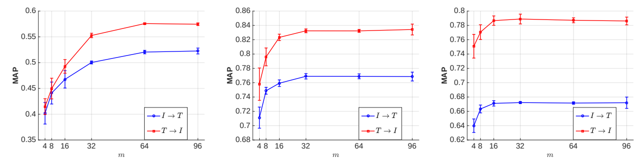


(a) IAPR-TC12. (b) MIRFLICKR-25K. (c) NUS-WIDE.

Fig. 9. MAP values with different λ on three datasets.

VI. CONCLUSION

In this paper, we propose a novel cross-modal hashing method, called discrete latent factor model based cross-modal hashing (DLFH), for cross-modal similarity search in large-scale datasets. DLFH is a discrete method which can directly learn the binary hash codes, and at the same time it is efficient. Experiments show that DLFH can significantly outperform relaxation-based continuous methods in terms of accuracy, but with a comparable training speed. Furthermore,



(a) IAPR-TC12. (b) MIRFLICKR-25K. (c) NUS-WIDE.

Fig. 10. MAP values with different number of sampled points m on three datasets.

DLFH can significantly outperform existing discrete methods in terms of both accuracy and training speed.

References

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

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

[3] A. Andoni and I. P. Razenshteyn, "Optimal data-dependent hashing for approximate near neighbors," in *STOC*, 2015, pp. 793–801.

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

[5] J. Wang, O. Kumar, and S. Chang, "Semi-supervised hashing for scalable image retrieval," in *CVPR*, 2010, pp. 3424–3431.

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

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

[8] G. Lin, C. Shen, Q. Shi, A. van den Hengel, and D. Suter, "Fast supervised hashing with decision trees for high-dimensional data," in *CVPR*, 2014, pp. 1971–1978.

[9] C. Da, S. Xu, K. Ding, G. Meng, S. Xiang, and C. Pan, "Amvh: Asymmetric multi-valued hashing," in *CVPR*, 2017, pp. 736–744.

[10] L. Liu, L. Shao, F. Shen, and M. Yu, "Discretely coding semantic rank orders for supervised image hashing," in *CVPR*, 2017, pp. 1425–1434.

[11] Z. Hu, J. Chen, H. Lu, and T. Zhang, "Bayesian supervised hashing," in *CVPR*, 2017, pp. 6348–6355.

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

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

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

[15] D. Wang, P. Cui, M. Ou, and W. Zhu, "Deep multimodal hashing with orthogonal regularization," in *IJCAI*, 2015, pp. 2291–2297.

[16] Y. Cao, M. Long, J. Wang, Q. Yang, and P. S. Yu, "Deep visual-semantic hashing for cross-modal retrieval," in *KDD*, 2016, pp. 1445–1454.

[17] Y. Cao, M. Long, J. Wang, and S. Liu, "Collective deep quantization for efficient cross-modal retrieval," in *AAAI*, 2017, pp. 3974–3980.

[18] D. Tian and D. Tao, "Global hashing system for fast image search," *TIP*, vol. 26, no. 1, pp. 79–89, 2017.

[19] J. Wang, H. T. Shen, J. Song, and J. Ji, "Hashing for similarity search: A survey," *CoRR*, vol. abs/1408.2927, 2014.

[20] J. Wang, W. Liu, S. Kumar, and S. Chang, "Learning to hash for indexing big data - A survey," *Proceedings of the IEEE*, vol. 104, no. 1, pp. 34–57, 2016.

[21] Y. Guo, G. Ding, L. Liu, J. Han, and L. Shao, "Learning to hash with optimized anchor embedding for scalable retrieval," *TIP*, vol. 26, no. 3, pp. 1344–1354, 2017.

[22] J. Wang, T. Zhang, J. Song, N. Sebe, and H. T. Shen, "A survey on learning to hash," *TPAMI*, vol. 40, no. 4, pp. 769–790, 2018.

[23] X. Lu, X. Zheng, and X. Li, "Latent semantic minimal hashing for image retrieval," *TIP*, vol. 26, no. 1, pp. 355–368, 2017.

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

[25] F. Shen, X. Zhou, Y. Yang, J. Song, H. T. Shen, and D. Tao, "A fast optimization method for general binary code learning," *TIP*, vol. 25, no. 12, pp. 5610–5621, 2016.

[26] D. Zhang and W. Li, "Large-scale supervised multimodal hashing with semantic correlation maximization," in *AAAI*, 2014, pp. 2177–2183.

[27] G. Ding, Y. Guo, and J. Zhou, "Collective matrix factorization hashing for multimodal data," in *CVPR*, 2014, pp. 2083–2090.

[28] X. Liu, L. Huang, C. Deng, B. Lang, and D. Tao, "Query-adaptive hash code ranking for large-scale multi-view visual search," *TIP*, vol. 25, no. 10, pp. 4514–4524, 2016.

[29] J. Song, Y. Yang, Z. Huang, H. T. Shen, and R. Hong, "Multiple feature hashing for real-time large scale near-duplicate video retrieval," in *MM*. ACM, 2011, pp. 423–432.

[30] D. Zhang, F. Wang, and L. Si, "Composite hashing with multiple information sources," in *SIGIR*, 2011, pp. 225–234.

[31] Y. Kang, S. Kim, and S. Choi, "Deep learning to hash with multiple representations," in *ICDM*, 2012, pp. 930–935.

[32] Y. Zhen and D. Yeung, "Co-regularized hashing for multimodal data," in *NIPS*, 2012, pp. 1385–1393.

[33] B. Dai, R. Guo, S. Kumar, N. He, and L. Song, "Stochastic generative hashing," in *ICML*, 2017, pp. 913–922.

[34] Q.-Y. Jiang and W.-J. Li, "Deep cross-modal hashing," in *CVPR*, 2017, pp. 3232–3240.

[35] G. Irie, H. Arai, and Y. Taniguchi, "Alternating co-quantization for cross-modal hashing," in *ICCV*, 2015, pp. 1886–1894.

[36] J. Zhang, Y. Peng, and M. Yuan, "Unsupervised generative adversarial cross-modal hashing," in *AAAI*, 2018, pp. 539–546.

[37] Y. Zhen and D. Yeung, "A probabilistic model for multimodal hash function learning," in *KDD*, 2012, pp. 940–948.

[38] Z. Lin, G. Ding, M. Hu, and J. Wang, "Semantics-preserving hashing for cross-view retrieval," in *CVPR*, 2015, pp. 3864–3872.

[39] H. Liu, R. Ji, Y. Wu, and G. Hua, "Supervised matrix factorization for cross-modality hashing," in *IJCAI*, 2016, pp. 1767–1773.

[40] M. Yu, L. Liu, and L. Shao, "Binary set embedding for cross-modal retrieval," *TNNLS*, 2016.

[41] Y. Cao, B. Liu, M. Long, and J. Wang, "Cross-modal hamming hashing," in *ECCV*, 2018, pp. 207–223.

[42] D. Mandal, K. N. Chaudhury, and S. Biswas, "Generalized semantic preserving hashing for n-label cross-modal retrieval," in *CVPR*, 2017, pp. 4076–4084.

[43] S. Kumar and R. Udupa, "Learning hash functions for cross-view similarity search," in *IJCAI*, 2011, pp. 1360–1365.

[44] M. Rastegari, J. Choi, S. Fakhraei, H. D. III, and L. S. Davis, "Predictable dual-view hashing," in *ICML*, 2013, pp. 1328–1336.

[45] R. Raziperchikolaei and M. Á. Carreira-Perpiñán, "Learning hashing with affinity-based loss functions using auxiliary coordinates," in *NIPS*, 2016.

[46] Y. Gong, S. Lazebnik, A. Gordo, and F. Perronnin, "Iterative quantization: A procrustean approach to learning binary codes for large-scale image retrieval," *TPAMI*, vol. 35, no. 12, pp. 2916–2929, 2013.

[47] X. He, D. Cai, S. Yan, and H. Zhang, "Neighborhood preserving embedding," in *ICCV*, 2005, pp. 1208–1213.

[48] K. Lange, D. R. Hunter, and I. Yang, "Optimization transfer using surrogate objective functions," *JCGS*, vol. 9, no. 1, pp. 1–20, 2000.

[49] D. Böhning and B. G. Lindsay, "Monotonicity of quadratic-approximation algorithms," *AISM*, vol. 40, no. 4, pp. 641–663, 1988.

[50] P. Zhang, W. Zhang, W. Li, and M. Guo, "Supervised hashing with latent factor models," in *SIGIR*, 2014, pp. 173–182.

[51] H. J. Escalante, C. A. Hernández, J. A. González, A. López-López, M. Montes-y-Gómez, E. F. Morales, L. E. Sucar, L. V. Pineda, and M. Grubinger, "The segmented and annotated IAPR TC-12 benchmark," *CVIU*, vol. 114, no. 4, pp. 419–428, 2010.

[52] M. J. Huiskes and M. S. Lew, "The MIR flickr retrieval evaluation," in *ACM SIGMM*, 2008, pp. 39–43.

[53] T. Chua, J. Tang, R. Hong, H. Li, Z. Luo, and Y. Zheng, "NUS-WIDE: a real-world web image database from national university of singapore," in *CIVR*, 2009.

Qing-Yuan Jiang received the BSc degree in computer science from Nanjing University, China, in 2014. He is currently working toward the PhD degree in the Department of Computer Science and Technology, Nanjing University. His research interests are in machine learning and learning to hash.

Wu-Jun Li received the BSc and MEng degrees in computer science from Nanjing University of China, and the PhD degree in computer science from the Hong Kong University of Science and Technology. He started his academic career as an assistant professor in the Department of Computer Science and Engineering, Shanghai Jiao Tong University. He then joined Nanjing University where he is currently an associate professor in the Department of Computer Science and Technology. His research interests are in machine learning, big data, and artificial intelligence. He is a member of the IEEE.