Sequential Discrete Hashing for Scalable Cross-Modality Similarity Retrieval

Li Liu, Zijia Lin, *Student Member, IEEE*, Ling Shao, *Senior Member, IEEE*, Fumin Shen, Guiguang Ding, *Member, IEEE*, and Jungong Han

***Abstract*— With the dramatic development of the Internet, how to exploit large-scale retrieval techniques for multimodal web data has become one of the most popular but challenging problems in computer vision and multimedia. Recently, hashing methods are used for fast nearest neighbor search in large-scale data spaces, by embedding high-dimensional feature descriptors into a similarity preserving Hamming space with a low dimension. Inspired by this, in this paper, we introduce a novel supervised cross-modality hashing framework, which can generate unified binary codes for instances represented in different modalities. Particularly, in the learning phase, each bit of a code can be sequentially learned with a discrete optimization scheme that jointly minimizes its empirical loss based on a boosting strategy. In a bitwise manner, hash functions are then learned for each modality, mapping the corresponding representations into unified hash codes. We regard this approach as cross-modality sequential discrete hashing (CSDH), which can effectively reduce the quantization errors arisen in the oversimplified rounding-off step and thus lead to high-quality binary codes. In the test phase, a simple fusion scheme is utilized to generate a unified hash code for final retrieval by merging the predicted hashing results of an unseen instance from different modalities. The proposed CSDH has been systematically evaluated on three standard data sets: Wiki, MIRFlickr, and NUS-WIDE, and the results show that our method significantly outperforms the state-of-the-art multi-modality hashing techniques.**

***Index Terms*— Cross-modality retrieval, hashing, discrete optimization, bitwise, unified hash code.**

Manuscript received June 11, 2016; revised September 12, 2016; accepted October 9, 2016. Date of publication October 19, 2016; date of current version November 15, 2016. This work was supported in part by Northumbria University and in part by the National Natural Science Foundation of China under Grant 61528106. The associate editor coordinating the review of this manuscript and approving it for publication was Prof. Xiaochun Cao. *(Corresponding author: Ling Shao.)*

L. Liu and L. Shao are with the School of Computer and Information Science, Southwest University, Chongqing 400715, China, and also with the Department of Computer and Information Sciences, Northumbria University, Newcastle upon Tyne, NE1 8ST, U.K. (e-mail: li2.liu@northumbria.ac.uk; ling.shao@ieee.org).

Z. Lin is with the Department of Computer Science and Technology, Tsinghua University, Beijing 100084, China (e-mail: linzijia07@tsinghua.org.cn).

F. Shen is with the School of Computer Science and Engineering, University of Electronic Science and Technology of China, Chengdu 611731, China (e-mail: fumin.shen@gmail.com).

G. Ding is with the School of Software, Tsinghua University, Beijing 100084, China (e-mail: dinggg@tsinghua.edu.cn).

J. Han is with the Department of Computer and Information Sciences, Northumbria University, Newcastle upon Tyne, NE1 8ST, U.K. (e-mail: jungong.han@northumbria.ac.uk).

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.2016.2619262

I. Introduction

IN RECENT years, with the increasing number of Internet users, a large amount of multimodal web data (e.g., images, texts and audio) has been continually generated and stored. Under such circumstances, how to achieve cross-modality similarity search becomes an interesting but challenging problems attracting a lot of attention [1]–[11]. One of the most basic but crucial schemes for previous similarity search is the nearest neighbor (NN) search. Particulary, given a query instance, we aim to find an instance that is most similar to it within a large database and assign the same label of the nearest neighbor to this query instance. However, such an NN search has the linear computational complexity $O(N)$, which is intractable for large-scale retrieval tasks in realistic scenarios. To overcome this issue, tree-based schemes are proposed to partition the search space via various tree structures. Among them, KD-tree and R-tree [12] are the most successful methods which can be applied to indexing the data for fast query responses with sub-linear complexity, i.e., $O(log(N))$. Although tree-based methods have their advantages for retrieval tasks, they still suffer from the *curse of dimensionality* problems,[1] since good and informative descriptors usually have hundreds or even thousands of dimensions. To achieve more effective and fast search, hashing schemes have been proposed to embed data from a high-dimensional feature space into a similarity-preserving low-dimensional Hamming space where an approximate nearest neighbor of a given query can be found efficiently. Current hashing techniques can be roughly divided into two groups, i.e., single-modality hashing [1], [13]–[27] and cross-modality hashing [2]–[4], [6]–[9], [28]–[32].

In particular, single-modality hashing mainly focuses on generating hash codes for only one specific data domain in both training and test phases. One of the most well-known single-modality hashing techniques is the Locality-Sensitive Hashing (LSH) [20] which simply employs random linear projections (followed by random thresholding) to map data points, that are close in a Euclidean space, to similar codes. Furthermore, Spectral Hashing (SpH) [19] is another representative unsupervised hashing method, in which the Laplace-Beltrami eigenfunctions of manifolds are used to determine binary codes. Principal linear projections like PCA

[1]The effectiveness and efficiency of these methods drop exponentially as the dimensionality increases, which is commonly referred to as the curse of dimensionality.

Hashing (PCAH) [21] was introduced for better quantization rather than random projection hashing. Anchor Graph-based Hashing (AGH) [24] can automatically discover the neighborhood structural inherent in the data to learn appropriate compact codes. More recently, Spherical Hashing (SpherH) [25] was proposed to map more spatially coherent data points into a novel spherical binary space compared to using previous hyperplane-based binary space. Iterative Quantization (ITQ) [23] was developed for more compact and balanced binary coding. Besides, asymmetric hashing [33] (e.g., Asymmetric Inner-product Binary Coding [27]) was also introduced for better retrieval performance. More single-modality hashing techniques can be seen in [5] and [34]–[40].

In terms of the cross-modality hashing, it deals with similarity searching between different data domains (e.g., image, text, audio, and video). For instance, we utilize text information as a query and retrieval the similar instances in an image database. Since multimedia era arriving, cross-modality retrieval has become significantly helpful and attracted increasing attention. Recently, various hashing methods for cross-modality retrieval were exploited, including unsupervised methods [9], [31], [41] and supervised methods [1], [3], [4], [7], [28]–[30]. For unsupervised methods, preserving intrinsic data structure via matrix decomposition or reconstruction is the most common scheme utilized in the binary code learning procedure. Thus, the quality of high-dimensional data before hashing is usually the vital fact influencing the effects of binary codes. Differently, supervised cross-modality hashing can generate more discriminative binary codes for better retrieval performance, since it effectively preserves the high-level label information instead of the low-level data structures. Such supervised information used in the cross-modality hashing can be semantic category labels and also even be multiple attribute tags.

Regardless of the types of the hashing methods (either single-modality hashing or cross-modality hashing), learning binary codes via target hash functions with discrete constraints always involve a mixed binary-integer optimization problem [26] which is generally NP-hard. To become tractable, most of the previous hashing methods [19], [21], [24], [42], first simplify such an NP-hard problem into a relaxed continues embedding problem by directly discarding the discrete constraints, and then threshold (quantize) the optimized continues embedding into an approximate binary solution. However, the relaxation scheme always makes the hash function less effective and leads to low-quality binary codes due to the accumulated quantization errors, especially when learning long binary codes. To further improve hashing quality, some other works, such as Kernelized Supervise Hashing (KSH) [18], attempt to simulate discrete constraint by replacing the sign function with the sigmoid function instead. Although this kind of discrete approximation can theoretically achieve better hash codes, hash functions with the sigmoid operation still make the optimization problem suboptimal. More importantly, for large-scale training data, the sigmoid operation will also be computationally expensive for hash function optimization due to the lack of an analytic solution, and the best way to achieve the non-closed-form solution is using gradient descent methods (e.g., SGD, or Newton method). To seek more effective hashing scheme, recently, some pioneer works [26], [27], [43], [44] have devoted to directly optimizing the hashing problem using discrete constraints and obtained closed-form solution. The provided results show that the methods based on the discrete optimization significantly outperform those methods relying on the relaxed optimization.

In this paper, we mainly focus on designing a supervised hashing approach which can produce compact but high-quality binary codes for cross-modality retrieval. Inspired by recent progresses in [26] and [27], we propose a novel bitwise hash learning scheme combining the discrete bit optimization with a boosting trick, which aims to directly optimize the binary codes efficiently and effectively. In particular, discrete bit optimization is sequentially applied to obtain a least-weighted-error binary code for each bit by jointly minimizing its empirical loss with the boosting strategy [45] on the training set. The supervision information in our CSDH is leveraged in proposed discrete optimization by preserving the pairwise semantic similarity, resulting in more discriminative binary codes. As our target is to achieve cross-modality retrieval, we regard the learned hash codes during the training phase as the unified codes for different modalities of each instance. We then adopt a linear regression to learn optimal hash code projections (i.e., hash functions), which effectively bridge each different modality and the learned unified binary codes. To better capture the intrinsic data structure, the hash function for each modality is actually learned in a nonlinear kernel space rather than directly using the raw feature from different modalities. In CSDH, we discretely optimize unified code for each modality with an alternate manner. Similar to [29], for better performance during the test phase, a simple unified code generating method is also proposed to solve "out-of-sample" problems for unseen data from different modalities. Experimental results clearly demonstrate that our CSDH can achieve superior cross-modality retrieval performance on three benchmark datasets compared to previous state-of-the-art hash methods. It is worthwhile to highlight several contributions of this paper:

- We propose a novel supervised hashing algorithm for cross-modality similarity retrieval named CSDH, which can generate high-quality unified binary codes for instances represented in different modalities. In particular, each bit of a code can be sequentially learned with the proposed discrete optimization by jointly minimizing its empirical loss with a boosting strategy. With such a bitwise manner, hash functions are then learned for each modality, mapping the corresponding representations into hash codes.
- In order to achieve better discrete optimization, an alternate scheme is adopted to transform the global NP-hard problem into several tractable sub-problems which can be directly solved with closed-form solutions.

The remainder of this paper is organized as follows. A brief review of the related work is given in Section II. In Section III, we present the proposed CSDH in detail. Experiments and results are described in Section IV. In Section V, we conclude this paper and outline the possible future work.

II. Related Work

In terms of the cross-modality similarity retrieval, recently, various unsupervised hashing methods and supervised hashing methods have been proposed.

For unsupervised methods, they usually optimize the hash codes by preserving the data structure within each modality (intra-modality) and the data structure between different modalities (inter-modality). In [9], an Inter-media Hashing (IMH) was introduced to explore the correlations among different modalities by considering the consistency of inter-modality and intra-modality data distribution, and learned hashing functions with the aid of a linear regression model. Furthermore, a novel hash method terms Collective Matrix Factorization Hashing (CMFH) was proposed in [31], where latent collective matrix factorization was successfully adopted to learn unified hash codes for different mortalities of instances. Besides, Latent Semantic Sparse Hashing (LSSH) [41] was developed to learn latent semantic representations for each modality with sparse coding scheme, and then mapped these latent representations into unified hash codes in a joint abstraction space. Despite the unsupervised setting, CMFH and LSSH could still achieve high performance for cross-modality search due to the qualitative latent semantics mining used in their algorithms.

With respect to supervised methods, available semantic information can be used to further improve the retrieval performance. Cross-Modal Semi-Supervised Hashing (CMSSH) [3] considered the optimization of the hash objective functions for different modalities as a binary classification problem and solved it based on a boosting scheme. In [4], Cross-View Hashing (CVH) was introduced to extend the ordinary spectral hashing [19] into the cases of multiple modalities by minimizing similarity-weighed Hamming dis- tance of the learned codes. Moreover, to learn the hash function with good generalization for cross-modality retrieval, Co-Regularized Hashing (CRH) [7] was proposed to learn binary codes in a bitwise manner based on the boosting framework. Similarly, Cross-view Kernel-based Supervised Hashing (KSH-CV) [30] aimed to preserve the similarity between different modalities and learned kernel hash func- tions with AdaBoost. Besides, some researchers integrated the semantic labels into a new hash methods, i.e., Semantic Cor- relation Maximization (SCM) [28], in which the binary codes could be optimized by maximizing semantic correlations. In a practical implement, two different versions of SCM were proposed, which were sequential SCM and orthogonal SCM respectively. Recently, another supervised hash method termed Semantics-Preserving Hashing (SePH) [29] was developed to generate unified codes by minimizing the KL-divergence of data semantic probability from different modalities. Once the optimal codes were obtained, a kernelized logistic regression was used to learn the hashing functions for each modality, respectively. From the reported results, SePH can significantly outperform most of state-of-the-art hashing techniques on cross-modality similarity retrieval.

Similar to CMSSH, CVH and KSH-CV, the proposed CSDH also adopts the boosting scheme in hash code learning procedure. However, a vital difference between CSDH and the other boosting related methods is that these three methods apply weak classifiers in boosting scheme as the hash function for generating binary codes; while, our CSDH just use boosting scheme to better integrate the proposed discrete optimization for each bit into a unified framework and the final hash functions are learned with linear regression. Since CSDH discretely optimizes hash codes without any oversimplifying rounding-off step, it can effectively reduce the quantization errors and leads to more accurate cross-modality retrieval performance compared to previous ones mentioned above.

III. Methodology

In this section, we mainly introduce the algorithm of our supervised Cross-modality Sequential Discrete Hashing (CSDH) in detail. A sequentially discrete optimization strategy is proposed to learn the unified hash codes of each bit. Simultaneously, hash functions are also learned during the optimization for each modality, mapping the feature representations into corresponding unified hash codes. For ease of explanation, we only formulate CSDH with considering a most common two-modality (i.e., image and text) retrieval scenario in our following sections. Of course, our CSDH can be intuitively extended to multi-modality (i.e., more than two modalities) cases as shown in the later part.

A. Notations and Problem Formulation

Given $O = \{X^{(1)}, X^{(2)}\}$ is a set of multimodal data and $X^{(1)} = [x^{(1)}_1, x^{(1)}_2, \ldots, x^{(1)}_n] \in \mathbb{R}^{d_1 \times n}$, $X^{(2)} = [x^{(2)}_1, x^{(2)}_2, \ldots, x^{(2)}_n] \in \mathbb{R}^{d_2 \times n}$ indicate the two different modalities (e.g., image and text) of O, where d_1 and d_2 (usually $d_1 \neq d_2$) are the dimensions of data from the two modalities, n is the number of instances. Due to our supervised framework,
we also introduce the label vector $Y = [y_1, y_2, ..., y_n] \in \{0, 1\}^{C \times n}$ in the algorithm, where C is the number of classes of O. Each column of Y contains at least one entry[2] as "1" which indicates the class of the instance that it belongs to, otherwise as "0". As mentioned above, in this paper, we aim to learn the unified binary codes $B = [b_1, b_2, \ldots, b_n]^{M \times n}$ for both modalities, where M denotes the length of the codes and $b_i \in \{1, -1\}^M$. To make codes discriminative based on semantic information, we denote each bit's codes as the results of a binary classifier $F(\cdot)$. Thus, following [46] and [47], the m-th bit code learning problem can be formulated as minimizing the pairwise classification loss

$$\begin{aligned} L^m &= \sum_{i=1}^{n}\sum_{j=1}^{n} \mathrm{h}(S_{ij} - b^m_i b^m_j), \\ \text{s.t. } b^m_i &= \mathrm{sign}(F^m_{(1)}(x^{(1)}_i)) = \mathrm{sign}(F^m_{(2)}(x^{(2)}_i)), \\ b^m_j &= \mathrm{sign}(F^m_{(1)}(x^{(1)}_j)) = \mathrm{sign}(F^m_{(2)}(x^{(2)}_j)) \end{aligned} \tag{1}$$

where, b^m_i is binary code for the m-th bit of the i-th instance and S is a $n \times n$ semantic similarity matrix[3] determined as

$$S_{ij} = \begin{cases} 1, & \text{if } y^T_i y_j \neq 0 \\ -1, & \text{if } y^T_i y_j = 0 \end{cases} \tag{2}$$

[2]For multi-labeled data, multiple "1" would exist in each column of Y.

[3]In fact, for large-scale setting, S matrix can be easily rewritten with low-rank decomposition as $S = Y^T K Y$, where $K = \mathbf{1} - 2\mathbf{I}$, $\mathbf{1}$ is $C \times C$ all ones matrix and $\mathbf{I}$ is the identity matrix.

In our approach, any two samples can jointly form a data pair and theoretically n^2 pairs in total can be obtained for all training instances. $h(\cdot, \cdot)$ in Eq. (1) indicates the Hamming distance. The linear binary classifiers can be regarded as the hash functions $H(x^{(1)}) = \text{sign}(F^m_{(1)}(x^{(1)}))$ and $H(x^{(2)}) = \text{sign}(F^m_{(2)}(x^{(2)}))$ which can encode the data from two different modalities into the single bit unified hash code. $\text{sign}(\cdot)$ is the sign function outputting $+1$ for positive values and -1 for negative ones. The similar formulation has also been used in Collaborative Hashing (CH) [48]. Particularly, CH can effectively handle many scenarios involving nearest neighbor search on the data given in matrix form, where two different yet naturally associated entities correspond to its two dimensions or views. Different from CH, Eq. (1) of our method involves the semantic similarity S_{ij} which is calculated from label information rather than directly using an existing relationship matrix as CH. Moreover, CH is specifically designed for retrieval in recommendation systems with user-item ratings, while our target is to design a unified framework for general cross-modality retrieval.

From Eq. (1), each bit can be optimized separately, however it will lead to redundant and less discriminative hash codes. To further make the codes more compact and effective, we adopt a sequential learning framework to optimize hash codes bit by bit instead of separately learning them. In particular, we tend to minimize the weighted error for each bit with a boosting strategy [45], which has been successfully deployed in previous hash techniques [3], [4], [30], [46], so that the total empirical loss can be written as

$$\begin{aligned} L &= \sum_{m=1}^{M}\sum_{i=1}^{n}\sum_{j=1}^{n} \alpha_{ij}^m \, h(S_{ij} - b_i^m b_j^m), \\ \text{s.t. } b_i^m &= \text{sign}(F_{(1)}^m(x_i^{(1)})) = \text{sign}(F_{(2)}^m(x_i^{(2)})), \\ b_j^m &= \text{sign}(F_{(1)}(x_j^{(1)})) = \text{sign}(F_{(2)}(x_j^{(2)})) \end{aligned} \tag{3}$$

where, α_{ij}^m is the the sample weight, which controls and adjusts the pairwise data classification results corresponding to m-th bit codes. Eq. (3) reflects the accumulated pairwise loss on the binary code prediction using the bitwise hash functions. Minimizing Eq. (3) aims at reducing the Hamming distances between data from positive pairs ($S_{ij} = 1$) while increasing the Hamming distances between data from negative pairs ($S_{ij} = -1$). The optimization problem of Eq. (3) seems to be related to the standard boosting formulation. Initially, each data pair is assigned the identical weights i.e., α_{ij}^1, $i, j = 1, \ldots, n$. After a single bit optimization, the weights associated to incorrectly recognized binary code pairs, i.e., $S_{ij} \neq b_i^m b_j^m$, $i, j = 1, \ldots, n$ and $m = 1 \ldots, M$, will be incremented, i.e., $\alpha_{ij}^m \uparrow$ which will be used in the next iteration of the bit optimization. Otherwise, the weights of correctly embedded binary code pair are decreased, i.e., $\alpha_{ij}^m \downarrow$. In this way, we attempt to minimize the weighted error $\sum_{i=1}^{n}\sum_{j=1}^{n} \alpha_{ij}^m h(S_{ij} - b_i^m b_j^m)$ in each bit optimization procedure and next bit always tend to correct the errors of preceding ones.

However, the problem in Eq. (3) is still NP hard and difficult to solve due to the discrete constrain of b_i. For easier getting the solutions, many of previous works, such as PCAH [21], KSH [18], AGH [24], tackle this by simply dropping off the sign function and transfer the discrete hashing optimization problem into a continues embedding problem, i.e., $b_i = F(x_i)$. After continues embedding b_i is obtained, the binary codes can be calculated by thresholding (quantization) schemes. However, as we mentioned above, due to the accumulated quantization errors in the non-discrete learning procedure, this kind of relaxing scheme causes low-quality binary codes and make the hash function less effective. Specifically, in multimodal hashing cases, this problem will become worse since large structure gap exists in different data modalities.

To obtain better unified hash codes for our cross-view similarity retrieval, in this paper, we propose a discrete binary codes optimization scheme by adding the binary constrain in Eq. (3) as a regularization item to achieve the large-scale optimization [26], [49]. Inspired by the previous discrete hashing method [26], [27], [43], [44], an alternate scheme is exploited here for each bit learning. In the following section, we will first introduce the kernelized feature embedding in $F(\cdot)$, then discrete bit optimization and boosting joint learning will be explained in detail.

B. Kernelized Feature Embedding

For hashing methods, the quality of large-scale data feature is one of the important factor determining the retrieval accuracy. However, in practice, the large variances, redundancies and noises would be unavoidably existed in mess of data features, which all negatively affect the quality of the generated hash codes. Thus, having good feature representations becomes crucial. To achieve better performance, in this paper, we propose a simple kernelized nonlinear embedding scheme to produce more powerful and discriminative data representations beyond the raw features. Particularly, we first adopt Gaussian Mixed Model (GMM) [50], [51] for each class of data in different modalities. From the viewpoint of data probabilistic distribution, instances in the original feature space do not always follow the same distribution, but are naturally clustered into several groups. The data in the same group shares the same probabilistic distribution. Thus, GMM can find the distribution centroids as the anchor points, which helps to form our kernelized nonlinear embedding as

$$\varphi(x) = [exp(-\gamma\|x - g_1\|^2), \ldots, exp(-\gamma\|x - g_k\|^2)]^T \tag{4}$$

where, $\|\cdot\|$ denotes the 2-Norm operation, $\varphi(x) \in \mathbb{R}^{k\times 1}$ is the k-dimensional column vector computed via RBF kernel mapping, $\{g_i\}_{i=1}^k$ indicates the k learned GMM anchor points from training data and γ is radial basis. It is noteworthy that for our cross-modality scenarios, we respectively embed $x^{(1)}$ into $\varphi(x^{(1)})$ with GMM anchors learned from $X^{(1)}$, and $x^{(2)}$ into $\varphi(x^{(2)})$ with GMM anchors learned from $X^{(2)}$. To simplify our method, in fact we learn the same number of anchors k for both modalities in our CSHD algorithm.

This kind of nonlinear embedding provides an effective way to obtain better feature representations and has also been demonstrated with superior performance in previous hashing method [18], [26], [29], [30]. Once the kernelized feature

embedding is achieved, the next step is to optimize the binary codes as well as the hash functions.

C. Discrete Bit Optimization

Since our algorithm is based on a sequential learning framework, the binary codes and corresponding hash functions are learned with a bitwise manner. In this section, we mainly focus on how to obtain a high-quality single bit of the codes. Recalling the Eq. (1), we add the binary constrains as regularization items to reshape our bitwise cross-modality objective function to

$$\begin{aligned}\min\ & L^m(S_{ij}, b_i^m b_j^m, \alpha_{ij}^m) + \lambda_1 \sum_{i=1}^{n} \|b_i^m - F_{(1)}^m(x_i^{(1)})\|_2 \\ & + \lambda_2 \sum_{i=1}^{n} \|b_i^m - F_{(2)}^m(x_i^{(2)})\|_2 \\ \text{s.t.}\ & b_i^m, b_j^m \in \{-1, +1\} \text{ and } m = 1, \ldots, M \end{aligned} \tag{5}$$

We transfer the binary constrain in Eq. (1) into two regularization terms which fit the error of the single bit binary code with the continues embedding $F(\cdot)$. λ_1 and λ_2 are penalty parameters which balance the discrete binary loss L^m and regularization terms. As mentioned in [26], with the large penalty parameters, minimizing Eq. (5) is approximately equal to Eq. (1) and in the realistic applications, the small difference between b^m and $F(x)$ is tolerant. Similar formulation has also been applied in [27] and [43]. In fact, to make the our algorithm more general, the embedding function $F(x)^m$ of m-th bit here adopts a simple projection form for both modalities as

$$F_{(j)}^m(x) = P_{(1)}^m \varphi(x^{(1)}) \text{ and } F_{(2)}^m(x) = P_{(2)}^m \varphi(x^{(2)}) \tag{6}$$

where $P_{(1)}^m \in \mathbb{R}^{1\times k}$ and $P_{(2)}^m \in \mathbb{R}^{1\times k}$ are m-th bit projection vectors that map $\varphi(x^{(1)})$ and $\varphi(x^{(2)})$ into a low-dimensional feature space.

From Eq. (5) and Eq. (6), it is obvious that the bitwise optimization with discrete constrain is still a non-convex, and non-smooth NP-hard problem. To the best of our knowledge, there is no direct way to find a globally optimal solution. Thus, in our paper, we proposed an alternate approach to obtain a local optimal solution instead. Particularly, we iteratively tackle this problem by solving the sub-problem corresponding to one variable when fixing all the other variables. For each sub-problem, it is tractable and easy to solve, and in this way we can effectively optimize Eq. (5). In the following part, we will elaborate our alternate bitwise optimization algorithm.

Alternate Optimization Scheme: To make our method more efficient and better for understand,, we first rewrite Eq. (5) into matrix form and replace Hamming distance $h(\cdot,\cdot)$ with Euclidean distance due to $h(a, b) = \frac{1}{4}\|a - b\|_2$ The new-form objective function of Eq. (5) is

$$\begin{aligned}\min\ & \sum_{i=1}^{n}\sum_{j=1}^{n} \alpha_{ij}^m \|S_{i,j} - b_i^m b_j^m\|_2 + \lambda_1 \|B^m - P_{(1)}^m \varphi(X^{(1)})\|_2 \\ & + \lambda_2 \|B^m - P_{(2)}^m \varphi(X^{(2)})\|^2 \\ \text{s.t.}\ & b_i^m, b_j^m \in \{-1, +1\} \text{ and } m = 1, \ldots, M \end{aligned} \tag{7}$$

where $B^m = [b_1^m, \ldots, b_n^m]^{1\times n}$ is the m-th bit unified hash codes. In Eq. (7), we have three variables i.e., B^m, $P_{(1)}^m$ and $P_{(2)}^m$. It is noteworthy that α_{ij}^m is regarded as a given value here and its calculation method will be introduced in the next section. To solve such an NP-hard problem, thus, we further split the Eq. (7) into three sub-problems, each of which becomes tractable.

1) $P_{(1)}^m$ Sub-Problem: To compute $P_{(1)}^m$, we fixed B^m and $P_{(2)}^m$ The problem (7) will be shrunk to

$$\min_{P_{(1)}^m} \|B^m - P_{(1)}^m \varphi(X^{(1)})\|^2 \tag{8}$$

Thus, the projection vector of $X^{(1)}$ can be then easily obtained by a linear regression

$$P_{(1)}^m = B^m \varphi(X^{(1)})^T (\varphi(X^{(1)})\varphi(X^{(1)})^T)^{-1} \tag{9}$$

2) $P_{(2)}^m$ Sub-Problem: Similar to solving $P_{(1)}^{(m)}$ sub-problem, $P_{(2)}^{(m)}$ can also be computed by

$$P_{(2)}^m = B^m \varphi(X^{(2)})^T (\varphi(X^{(2)})\varphi(X^{(2)})^T)^{-1} \tag{10}$$

3) B^m Sub-Problem: It is challenging to optimize B^m in Eq. (7), since the discrete constrain makes this problem perform as NP-hard. In this paper, we find a closed-form solution for b_i^m by fixing all other bit $\{b_j^m\}_{j\neq i}^n$ during optimization. Once $\{b_j^m\}_{j\neq i}^n$ are fixed, and $\|b_i^m\|_2^2 = 1, \forall i$,we can rewrite B^m sub-problem as the following equations:

$$\begin{aligned}\min_{B^m}\ & \sum_{i,j}^{n} \alpha_{ij}^m \|S_{i,j} - b_i^m b_j^m\|_2 + \lambda_1 \|B^m - P_{(1)}^m \varphi(X^{(1)})\|_2 \\ & + \lambda_2 \|B^m - P_{(2)}^m \varphi(X^{(2)})\|^2 \\ =\ & \min_{b_i^m} \sum_{i,j}^{n} \alpha_{ij}^m \|S_{i,j} - b_i^m b_j^m\|^2 + \lambda_1 \|b_i^m - P_{(1)}^m \varphi(x_i^{(1)})\|^2 \\ & + \lambda_2 \|B^m - P_{(2)}^m \varphi(x_i^{(2)})\|^2 + const \\ =\ & \min_{b_i^m} -2 \sum_{i\neq j}^{n} \alpha_{ij}^m S_{ij} b_i^{mT} b_j^m - 2\lambda_1 b_i^{mT} P_{(1)}^m \varphi(x_i^{(1)}) \\ & - 2\lambda_2 b_i^{mT} P_{(2)}^m \varphi(x_i^{(2)}) + const \\ =\ & \min_{b_i^m} -2 b_i^{mT} \Big(\sum_{i\neq j}^{n} \alpha_{ij}^m S_{ij} b_j^m + \lambda_1 P_{(1)}^m \varphi(x_i^{(1)}) \\ & + \lambda_2 P_{(2)}^m \varphi(x_i^{(2)}) \Big) + const \\ \text{s.t.}\ & b_i^m, b_j^m \in \{-1, +1\},\ i = 1, \ldots, n \end{aligned} \tag{11}$$

where "*const*" means the constant value during the mathematical inferring. Eq. (11) can then be easily regarded as

$$\max_{b^m} b^{mT} \Big(\sum_{i\neq j}^{n} \alpha_{ij}^m S_{ij} b_j^m + \lambda_1 P_{(1)}^m \varphi(x_i^{(1)}) + \lambda_2 P_{(2)}^m \varphi(x_i^{(2)}) \Big) \tag{12}$$

Straightforwardly, the above problem has the closed-form optimal solution:

$$b_i^m = \text{sign}\Big(\sum_{j=1, j\neq i}^{n} \alpha_{ij}^m S_{ij} b_j^m + \lambda_1 P_{(1)}^m{}^i \varphi(x_i^{(1)}) + \lambda_2 P_{(2)}^m \varphi(x_i^{(2)})\Big)$$
$$= \text{sign}(\widehat{B}_i^m (\alpha_i^m \circledS S_i) + \lambda_1 P_{(1)}^m \varphi(x_i^{(1)}) + \lambda_2 P_{(2)}^m \varphi(x_i^{(2)})) \tag{13}$$

where $\widehat{B}_i^m \in \{-1, +1\}^{1\times(n-1)}$ indicates B^m excluding b_i^m, $S_i = (S_{i,1}, \ldots, S_{i,i-1}, S_{i,i+1}, \ldots, S_{i,n})^T$ and $\alpha_i^m = (\alpha_{i,1}^m, \ldots, \alpha_{i,i-1}^m, \alpha_{i,i+1}^m, \ldots, \alpha_{i,n}^m)^T$. Note that "$\circledS$" indicates the element-wise product.

We can observe that computing single bit binary codes for each data point relies on the rest of pre-learned $(n-1)$ binary codes. Thus, we need to learn b_i^m for each data one by one and update B^m for n times to obtain the final optimized B^m. Since only one bit of a certain data is updated at each time, the total procedure for discretely learning B^m in this sub-problem will be very efficient.

We have so far described our optimization of each step for Eq. (7) in detail. As mentioned above, to obtain a local optimal solution of problem 7, we adopt an alternate scheme for each bit code learning, in which we repeat t times to solve $P_{(1)}^m$ sub-problem, $P_{(2)}^m$ sub-problem and B^m sub-problem in sequence. In our experiments, $t = 3 \sim 5$ is proved to be enough for convergence.

B^m initialization: Since B^m a conjunction variable involved in all three sub-problems, in our alternate optimization procedure, a good initialization of B^m becomes very crucial. Inspired by SH [19] and KSH [18], we initialize the binary codes by thresholding spectral graph decomposition. Considering supervised setting, in this paper, we construct a weighted semantic graph on training data and minimize $\text{Trace}(V^m{}^T (A \circledS S) V^m)$ s.t. $V^m{}^T V^m = 1$, $V^m \in \mathbb{R}^{n\times 1}$, where $S \in \{-1, +1\}^{n\times n}$ as defined in Eq. (2) and $A \in \mathbb{R}^{n\times n}$ is the sample weight matrix for current bit optimization, in which $A_{ij} = \alpha_{ij}^m, \forall i, j$. The initial B^m can be easily determined as

$$B^m = \text{sign}(V^m) \tag{14}$$

where we apply a standard eigenvector-eigenvalue decomposition on $(A \circledS S)$ and V^m is the eigenvector with the largest eigenvalue.

D. Boosting Joint Learning

In the above section, we have presented the discrete learning algorithm for each bit of our CSDH. However, we haven't discussed how to calculate the sample weight α_{ij}^m, $i, j = 1, \ldots, n$ for minimizing the total loss function in Eq. (3), where α_{ij}^m is a key factor effectively linking each bit together and forming our final compact hash codes. In this paper, we adopted an AdaBoost-like scheme to updated α_{ij}^m for each bit optimization, as AdaBoost is the most practically efficient boosting algorithm used in various applications, such as Viola-Jones face detector [52].

After the discrete optimization (mentioned in Section III-C) for m-th bit is accomplished, we can calculate the embedding error of this bit by

$$Er^m = \sum_{i,j=1}^{n} \alpha_{ij}^m h(S_{i,j} - b_i^m b_j^m) = \sum_{i,j=1}^{n} \alpha_{ij}^m \frac{1}{4} \|S_{i,j} - b_i^m b_j^m\|_2 \tag{15}$$

where $m = 1, \ldots, M$. Based on AdaBoost scheme [45], for the next bit (i.e., $m+1$-th bit), the sample weight α_{ij}^{m+1} can be updated as

$$\alpha_{ij}^{m+1} = \frac{\alpha_{ij}^m \exp(-\ln(\frac{1-Er^m}{Er^m}) S_{ij} b_i^m b_j^m)}{\sum_{i,j=1}^{n} \alpha_{ij}^m \exp(-\ln(\frac{1-Er^m}{Er^m}) S_{ij} b_i^m b_j^m)}$$
$$\text{s.t.} \sum_{i,j=1}^{n} \alpha_{ij}^{m+1} = 1 \tag{16}$$

where $\ln(\cdot)$ denotes the natural logarithm. Based on above updating rules, at each bit incorrectly hashed instances (e.g., a pair data with same semantic label was mistakenly embedded into different binary values $b_i^m \neq b_j^m$) will be re-assigned to larger sample weight α_{ij}^{m+1} in next bit optimization. Otherwise, the weight α_{ij}^{m+1} will be reduced for correctly hashed instances.

Note that, for the first bit ($m = 1$) optimization, the identical sample weight $\alpha_{ij}^1 = 1/n^2$ is initially assigned and then sequentially updated via Eq. (16) for subsequent bit optimization. Besides, due to the mechanism of AdaBoost, hashing bits are not equally important in our method, and the importance of each bit can be quantitatively described by $\ln(\frac{1-Er^m}{Er^m})$ with a roughly decreased tendency from the first to the last bit.

According to the boosting joint learning, we can conclude our CSDH algorithm as follows: given training data $X^{(1)}$, $X^{(2)}$ and the corresponding label Y, CSDH can sequentially learn a bitwise binary hash function P^m, B^m codes and projections of $P_{(1)}^m$ and $P_{(2)}^m$ with updated sample weight α_{ij}^m according to Section III-C, where $m = 1, \ldots, M$. Such a sequential learning scheme tends to continually correct the hashing errors of preceding bits and jointly optimize its empirical loss with boosting strategy. Thus, the final hash projection matrices for different modalities can be computed as $P_{(1)} = [P_{(1)}^1{}^T, \ldots, P_{(1)}^M{}^T]^T \in \mathbb{R}^{M\times k}$ and $P_{(2)} = [P_{(2)}^1{}^T, \ldots, P_{(2)}^M{}^T]^T \in \mathbb{R}^{M\times k}$, where k is the dimension of kernelized feature space. The optimized hash code is $B = [b_1, b_2, \ldots, b_n]^{M\times n} = [B^1{}^T, \ldots, B^M{}^T]^T$. The overall of the proposed CSDH is depicted in Algorithm. 1.

E. Out-of-Sample Hash Codes Generating

Once we obtain hash projection matrices in each view $P_{(1)}$ and $P_{(2)}$, for an unseen instance, the predicted hash code can be computed via a sign function on a linear embedding. In particular, if the unseen instance can only be observed from one certain view, the straightforward way is to hash the data to the view-specific binary codes. For instance, given an unseen data $x_u^{(\psi)}$ from ψ-th view, we first compute kernelized feature embedding $\varphi(x_u^{(\psi)})$ by Eq. 4 and the final hash codes for this

Algorithm 1 Cross-Modality Sequential Discrete Hashing (CSDH)

Input: Multimodal training data matrices $X^{(\psi)} \in \mathbb{R}^{d_\psi \times n}$, $\psi = 1, 2$, label matrix $Y \in \{0, 1\}^{C \times n}$, parameter k, λ_1 and λ_2.
Output: Unified Hash codes $B = [B^1{}^T, \ldots, B^M{}^T]^T \in \{-1, +1\}^{M \times n}$ of the training set, projection matrices $P_{(\psi)} = [P^1_{(\psi)}{}^T, \ldots, P^M_{(\psi)}{}^T]^T \in \mathbb{R}^{M \times k}$, $\psi = 1, 2$ for different modalities.

Compute the kernel feature embedding $\phi(X^{(\psi)}) \in \mathbb{R}^{k \times n}$ for $X^{(\psi)}$, $\psi = 1, 2$ using Eq. (4), set the initial sample weight $\alpha^1_{ij} = \frac{1}{n^2}$, $i, j = 1, \ldots, n$, and construct semantic similarity matrix $S = Y^T \Lambda Y \in \{-1, +1\}^{n \times n}$, where $\Lambda = \mathbf{1} - 2\mathbf{I}$, $\mathbf{1}$ is $C \times C$ all ones matrix and $\mathbf{I}$ is the identity matrix.
for $m = 1 : M$ **do**
  Initialize $B^m = [b^m_1, \ldots, b^m_n]^{1 \times n}$ for the m-th bit using Eq. (14).
  Repeat
    P-Step: Fix B^m and update $P^m_{(\psi)}$ using Eq. 9 or Eq. (10), $\psi = 1, 2$;
    B-Step: Fix $P_{(\psi)}$, $\psi = 1, 2$ and update b^m_i using Eq. (13), $i = 1, \ldots, n$;
  Until convergency or reach maximum t iterations.
  Return B^m and $P^m_{(\psi)}$, $\psi = 1, 2$.
  Calculate the hash error of m-th bit using Eq. (15) and then update sample weight $\alpha^{m+1}_{ij} \leftarrow \alpha^m_{ij}$ using Eq. (16).
end

unseen data are defined by

$$H(x_u^{(\psi)}) = \text{sign}(P_{(\psi)}\varphi(x_u^{(\psi)})), \ \psi = 1, 2 \quad (17)$$

For the unseen instance that can be observed from multiple views (i.e., 2 views for simplification) observed, we want to generate unified hash codes by merging the predicted codes under different views. As mentioned in recent cross-view hashing methods SePH [29] and CMFH [31], unified code can achieve much better results for cross-view similarity retrieval tasks, especially when the predicted hash codes from different views are inconsistent. Inspired by the success in [29] and [31], we also want to generate the unified code for our searching task. However, our code generating in different views are not based on any probabilistic prediction like SePH which combines Bayes's theorem with probabilities outputted from kernel logistic regression to formulate a code unifying framework. Thus, we have to figure out a novel code unifying way under our framework rather than directly use previous techniques. In this paper, we carry out a weighted linear combination of the hash predictions from different views to generate the unified codes. More specifically, for m-th bit ($\forall m \in \{1, \ldots, M\}$), we aim to learn a proper balancer $W^m \in \mathbb{R}^{1 \times 2}$ and bias $K^m \in \mathbb{R}^{1 \times 1}$ to better fit

$$B^m = \text{sign}(W^m \begin{bmatrix} P^m_{(1)}\varphi(X^{(1)}) \\ P^m_{(2)}\varphi(X^{(2)}) \end{bmatrix} + K^m) \text{ s.t. } B^m \in \{-1, +1\}^n \quad (18)$$

where $X^{(1)}$ and $X^{(2)}$ are the training data from both views and B^m is optimal hash codes for m-th bit after the discrete optimization by Eq. (13). To solve problem (18), we simply apply a linear SVM[4] to find the solution of W^m and K^m for each bit, $m = 1, \ldots, M$. Once we obtained the balancers and biases from the training set, for out-of-sample extension of a unseen data from both views $o_u = \{x^{(1)}_u, x^{(2)}_u\}$, the unified codes for m-th bit can be generated as

$$H^m(x^{(1)}_u, x^{(2)}_u) = \text{sign}(W^m \begin{bmatrix} P^m_{(1)}\varphi(x^{(1)}_u) \\ P^m_{(2)}\varphi(x^{(2)}_u) \end{bmatrix} + K^m) \quad (19)$$

[4]In practical implementation, LibSVM toolbox is used to solve this binary classification problem.

Algorithm 2 Out-of-Sample Hash Code Generating

Input: Multimodal training data $\mathcal{O} = \{X^{(1)}, X^{(2)}\}$, label matrix $Y \in \{0, 1\}^{C \times n}$ and an unseen multimodal data $o_u = \{x^{(1)}_u, x^{(2)}_u\}$.
Output: W^m and Ω^m, $m = 1, \ldots M$.

Compute $P^m_{(1)}$, $P^m_{(2)}$ and B^m by running Algorithm. 1, $m = 1, \ldots M$.
for $m = 1 : M$ **do**
  Obtain W^m and Ω^m via LibSVM by fitting Eq. (18).
end
Return the unified hash code $B_u = [H^1(x^{(1)}_u, x^{(2)}_u), \ldots, H^M(x^{(1)}_u, x^{(2)}_u)]^T \in \{-1, +1\}^{M \times 1}$ for the unseen data $o_u = \{x^{(1)}_u, x^{(2)}_u\}$ via Eq. (19).

The later experiments demonstrate this unified code generating scheme can lead to good performance on cross-view retrieval tasks. Algorithm. 2 concludes our out-of-sample unified code generating procedure.

F. Multiple Modalities Extension

All the formulations above are used for the simplified version of two-modality cross-view similarity search. In this section, we will extend the current algorithm to more general cases with $\mathcal{O} = \{X^{(1)}, \ldots, X^{(l)}\}$ ($l > 2$), where l is the number of modalities. The bit optimization objective function can be simply rewritten from Eq. (7) as

$$\sum_{i=1}^{n}\sum_{j=1}^{n} \alpha^m_{ij} ||S_{i,j} - b^m_i b^m_j||_2 + \sum_{\psi=1}^{l} \lambda_\psi ||B^m - P^m_{(\psi)}\varphi(X^{(\psi)})||_2$$
$$\text{s.t. } b^m_i, b^m_j \in \{-1, +1\} \text{ and } m = 1, \ldots, M$$

Here, our alternate optimization scheme can be directly used to compute B^m and $P^m_{(\psi)}$ for each bit, where $\psi = 1, \ldots, l$ and $m = 1, \ldots, M$. For out-of-sample unified code learning of an unseen multimodal data, we can also rewrite

the Eq. (19) as

$$H^m(x_u^{(1)}, \ldots, x_u^{(l)}) = \text{sign}\left(W^m \begin{bmatrix} P_{(1)}^m \varphi(x_u^{(1)}) \\ \vdots \\ P_{(l)}^m \varphi(x_u^{(l)}) \end{bmatrix} + K^m\right)$$

where $o_u = \{x_u^{(1)}, \ldots, x_u^{(l)}\}$ is the multimodal data from l views, W^m and K^m are optimized via a linear SVM similar to two-modality case. By doing so, our CSDH can be applied on general cross-modality retrieval tasks.

IV. Experiments and Results

In this section, we first introduce three different datasets, i.e., Wiki, MIRFlickr and NUS-WIDE, used in our experiments and then carefully describe corresponding experimental settings. After that, we systematically evaluate our CSDH with several state-of-the-art methods and illustrate the relevant results.

A. Datasets

The **Wiki** dataset [53] collects samples from Wikipedia "featured articles", containing 2866 image-text pairs in 10 semantic classes. For each image, a 128-dimensional SIFT [54] feature is extracted and a 10-dimensional topic vector is used for representing each text. Wiki is announced as a single label dataset, in which each image-text pair is only attached to one of the 10 semantic labels. Following the setting in the original paper [53], we take 2173 image-text pairs as the retrieval set and the remaining 693 image-text pairs as the query set.

The **MIRFlickr** dataset [55] contains 25,000 images as well as their textural tags collected from Flickr. Unlike Wiki dataset, each instance in the MIRFlickr dataset is assigned with multiple semantic labels from some of the 24 provided categories. For each instance, the image view is represented by a 150-dimensional edge histogram and the text view is represented by a 500-dimensional feature vector which is derived from PCA on its textual tags. Following the previous setting in [29], we take 5% of the dataset as the query set and the rest as the retrieval set.

The **NUS-WIDE** dataset [56] is a real-world web image dataset containing around 270,000 web images associated with 81 ground truth concept classes. As in [29], we only use the most frequent 10 concept classes, each of which has abundant relevant images ranging from 5,000 to 30,000, and totally 186,577 instances are kept. Similar to the MIRFlickr dataset, each image in the NUS-WIDE dataset is also assigned with multiple semantic labels. Each image-text pair of one instance is represented by a 500-dimensional Bag-of-Word SIFT feature and a 1000-dimensional binary tagging vectors, respectively. Similar to [29]–[31], we further sample 1% of the dataset as the query set and the rest as the retrieval set.

Following the popular experimental setting in [29] and [31], we use the whole retrieval set of Wiki dataset as the training set due to its small size, while for larger MIRFlickr and NUS-WIDE datasets, to reduce the computational complexity, we take 5000 instances from their retrieval sets to construct training set, respectively. Specifically, for a fair comparison, we use the selection index of 5000-instance training data for these two datasets provided by [29]. Since all three datasets are with two-modality cases, i.e., Image and Text, we evaluate the performance of cross-modality retrieval for both "Image Query with Text Dataset" (i.e., I2T) and "Text Query with Image Dataset" (i.e., T2I) on each dataset. Besides, MIRFlickr and NUS-WIDE are multi-label datasets, thus we regard two instances to be in the same category only if they share at least one common tag.

B. Compared Methods and Experimental Settings

In our experiments, we systematically compare the proposed CSDH method with three unsupervised hashing methods: CMFH [31], LSSH [41] and QCH [32], and eight supervised hashing methods: IMH [9], CVH [4], CMSSH [3], CRH [7], SCM-Orth [28], SCM-Seq [28], KSH-CV [30] and SePH [29] for cross-modality retrieval tasks. It is noteworthy that IMH is originally designed as an unsupervised method, but in our experiments, we follow [29] to regard IMH as a supervised method by using annotated category tags of the training instance to compute the required similarity matrix. All the compared methods are achieved with the public codes except for self-implemented CVH. For fair comparison, the parameters of the above baselines are carefully selected via cross-validation rather than using default values for yielding better performance.

For the proposed CSDH, the parameter k in Eq. (4) is emphatically set to 500 as same as [29] and the radial basis γ is tuned with cross-validation on the training set. Furthermore, to make optimization simple, we set the balancing parameters for two modalities $\lambda_1 = \lambda_2 = \lambda$ in Eq. (7) which is selected from one of $\{0.0001, 0.001, 0.01, 0.1, 1, 10\}$ yielding the best performance by cross-validation on the training set. The maximum iteration number for each bit optimization is set to $t = 5$. It is noted that the semantic similarity matrix construction for CSDH in Eq. (2) is slightly different on the three datasets used in this paper. Particularly, for Wiki dataset, we define $S_{ij} == -1$ as if two instances are not from the same category, otherwise $S_{ij} == 1$. However, for multi-labeled MIRFlickr and NUS-WIDE datasets, $S_{ij} == -1$ only if two instances do not have at least one common label; otherwise $S_{ij} == 1$.

In the query phase, we report the mean average precision (MAP) to evaluate the retrieval performance for all three datasets. It is defined as

$$\text{MAP} = \frac{1}{|Q|} \sum_{i=1}^{|Q|} \frac{1}{r} \sum_{j=1}^{r} P(ij),$$

where $|Q|$ is the size of the query set, r is the number of the ground-truth relevant instances in the retrieval set for the i-th query and $P(ij)$ indicates the precision of the top j retrieved texts (images) of the i-th image (text). In addition, all of the methods are evaluated on four different lengths of codes $\{16, 32, 64, 128\}$. Our experiments are conducted using Matlab 2014a on a server configured with a 12-core processor and 128 GB of RAM running the Linux OS. Since relative

TABLE I

CROSS-MODAL RETRIEVAL PERFORMANCE OF THE PROPOSED CSDH AND COMPARED BASELINES ON ALL DATASETS WITH DIFFERENT HASH CODE LENGTHS, IN TERMS OF MAP

Task	Method	Wiki				MIRFlickr				NUS-WIDE			
		16 bits	32 bits	64 bits	128 bits	16 bits	32 bits	64 bits	128 bits	16 bits	32 bits	64 bits	128 bits
Image Query vs. Text Database	IMH	0.1576	0.1596	0.1632	0.1593	0.6020	0.6119	0.6082	0.5991	0.4171	0.3974	0.3792	0.3673
	CMFH	0.2150	0.2268	0.2396	0.2424	0.5849	0.5844	0.5843	0.5841	0.4309	0.4231	0.4217	0.4201
	LSSH	0.2155	0.2293	0.2235	0.2188	0.5785	0.5787	0.5811	0.5811	0.3919	0.3939	0.3951	0.3964
	CMSSH	0.1866	0.1729	0.1670	0.1562	0.5722	0.5737	0.5721	0.5704	0.4051	0.3995	0.3886	0.3805
	CVH	0.1249	0.1199	0.1212	0.1167	0.6075	0.6177	0.6156	0.6078	0.3689	0.4172	0.4605	0.4459
	CRH	0.1562	0.1576	0.1560	0.1520	0.5595	0.5647	0.5633	0.5648	0.3448	0.3454	0.3513	0.3578
	QCH	0.2058	0.2173	0.2202	0.2199	0.5505	0.5528	0.5589	0.5732	0.4897	0.4980	0.5015	0.5017
	KSH-CV	0.2005	0.1835	0.1775	0.1649	0.5782	0.5826	0.5786	0.5744	0.4049	0.4061	0.3910	0.3810
	SCM-Orth	0.1544	0.1511	0.1230	0.1290	0.5848	0.5765	0.5707	0.5639	0.3782	0.3678	0.3604	0.3528
	SCM-Seq	0.2214	0.2325	0.2443	0.2601	0.6236	0.6325	0.6467	0.6494	0.4843	0.4936	0.4948	0.4967
	SePH	0.2787	0.2956	0.3064	0.3134	0.6723	0.6771	0.6783	0.6817	0.5421	0.5499	0.5537	0.5601
	CSDH	**0.3173** ± 0.0102	**0.3377** ± 0.0084	**0.3441** ± 0.0095	**0.3567** ± 0.0062	**0.7032** ± 0.0091	**0.7143** ± 0.0076	**0.7222** ± 0.0080	**0.7274** ± 0.0068	**0.5638** ± 0.0031	**0.5742** ± 0.0026	**0.5808** ± 0.0020	**0.5869** ± 0.0033
Text Query vs. Image Database	IMH	0.1472	0.1358	0.1286	0.1285	0.5896	0.6035	0.6010	0.5937	0.4037	0.3907	0.3769	0.3621
	CMFH	0.4884	0.5139	0.5314	0.5391	0.5945	0.5932	0.5928	0.5920	0.4639	0.4561	0.4505	0.4487
	LSSH	0.4954	0.5205	0.5244	0.5291	0.5909	0.5924	0.5937	0.5944	0.4280	0.4239	0.4228	0.4177
	CMSSH	0.1620	0.1594	0.1557	0.1539	0.5694	0.5741	0.5709	0.5686	0.3896	0.3779	0.3750	0.3673
	CVH	0.1194	0.1046	0.1016	0.1003	0.6036	0.6034	0.6016	0.5966	0.3635	0.4032	0.4346	0.4261
	CRH	0.1205	0.1174	0.1221	0.1189	0.5612	0.5629	0.5615	0.5630	0.3511	0.3516	0.3583	0.3623
	QCH	0.3033	0.3173	0.3263	0.3290	0.5613	0.5674	0.5712	0.5705	0.5768	0.5810	0.5969	0.6047
	KSH-CV	0.1701	0.1721	0.1669	0.1638	0.5761	0.5789	0.5747	0.5735	0.4066	0.3941	0.3847	0.3826
	SCM-Orth	0.1546	0.1332	0.1099	0.1106	0.5870	0.5761	0.5692	0.5605	0.3762	0.3645	0.3573	0.3523
	SCM-Seq	0.2138	0.2377	0.2469	0.2574	0.6147	0.6212	0.6301	0.6327	0.4540	0.4631	0.4626	0.4652
	SePH	0.6318	0.6577	0.6646	0.6709	0.7197	0.7271	0.7309	0.7354	0.6302	0.6425	0.6506	0.6580
	CSDH	**0.6778** ± 0.0078	**0.6915** ± 0.0054	**0.6986** ± 0.0088	**0.7038** ± 0.0079	**0.7440** ± 0.0098	**0.7532** ± 0.0077	**0.7618** ± 0.0080	**0.7639** ± 0.0110	**0.6530** ± 0.0041	**0.6681** ± 0.0052	**0.6742** ± 0.0047	**0.6760** ± 0.0024

The results of CSDH are mean accuracies of 5 runs with a degree of uncertainty.

randomness exists in the selection of k centroids by GMM during our CSDH procedure, we repeated CSDH five times for all the datasets, and we report the averages as the final results.

C. Results

In Table I, we demonstrate the MAP on all three datasets, i.e, Wiki, MIRFlickr and NUS-WIDE datasets. Since we focus on the cross-modality retrieval task, we show the corresponding results on two aspects respectively: image query vs. text database (I2T) and text query vs. image database (T2I). In general, we can clearly see that the MAP of Wiki dataset is lower than those of MIRFlickr and NUS-WIDE datasets in terms of both I2T and T2I. The reason is that each instance of Wiki is assigned with only one single label, while the instances of other two datasets are assigned with multiple labels. As we mentioned above, two instances are regarded as the same category in MIRFlickr and NUS-WIDE datasets, only if they share at least one common tag.

More detail, it is easy to discover that the MAP of CVH CMSSH and IMH are always fluctuant with the increase of the code length. Specifically, the best performances of CVH and IMH are achieved with relatively short length of codes (i.e., 16 and 32 bits) for both I2T and T2I on all three datasets. Besides, we can also find that with the code length increasing, the tendencies of MAPs calculated by KSH-CV and SCM-Orth are dramatically deceasing on all datasets. Moreover, unsupervised methods QCH, LSSH and CMFH always achieve better performance than supervised IMH, CVH, CMSSH, CRH, SCM-Orth and KSH-CV for both I2T search and T2I search, since the qualitative latent semantic features are well exploited in QCH, LSSH and CMFH. Besides, the unified codes used in CMFH can also improve the retrieval accuracies. From Table I, our CSDH can produce significantly better performance than CMFH, LSSH and QCH, IMH, CVH, CMSSH, CRH, SCM-Orth, SCM-Seq and KSH-CV for both I2T and T2I on all three datasets and even slightly outperforms the recent SePH method. Beyond those, the degree of uncertainty via 5 trails of our CSDH is generally less than 1% according to the corresponding MAP for both tasks, since the selection of the GMM centroids is not that influential in achieving good performance. The precision-recall curves with the code lengths of 32 and 128 bits are also shown in Fig. 1. By measuring the area under curve (AUC), it is obviously discovered that the proposed CSDH can consistently lead to better performance than all other state-of-the-art methods.

We have also done parameter sensitivity analysis on λ. Fig. 2 shows the MAP with 32 and 128 bits of our CSDH by using different λ values on all three datasets and two different tasks (i.e., I2T and T2I). From the general perspective, the best retrieval performance always occurs when $\lambda = 0.01$ or $\lambda = 0.001$ for all datasets. Particularly, it is discovered that with the increase of λ, the MAPs on the Wiki and NUS-WIDE datasets is relative stable for I2T task, while for T2I task the MAPs on these two datastes are fluctuant. For the MIRFlickr dataset, the MAP has the similar tendency of "rise-then-down" for both I2T and T2I tasks. To conclude, when λ takes various values for all datasets, it is not sensitive for the I2T performance, but variation slightly exists for the T2I task.

Moreover, we have also illustrated the effectiveness of varying training set size on the NUS-WIDE dataset. In detail, we vary the number of samples in the training set from 500 to 5000 with the step of 500. Fig. 3 demonstrates the MAP of our proposed CSDH with 128 bits for different training set sizes. It is obviously observed that the performance significantly increases when the number of training samples grows from

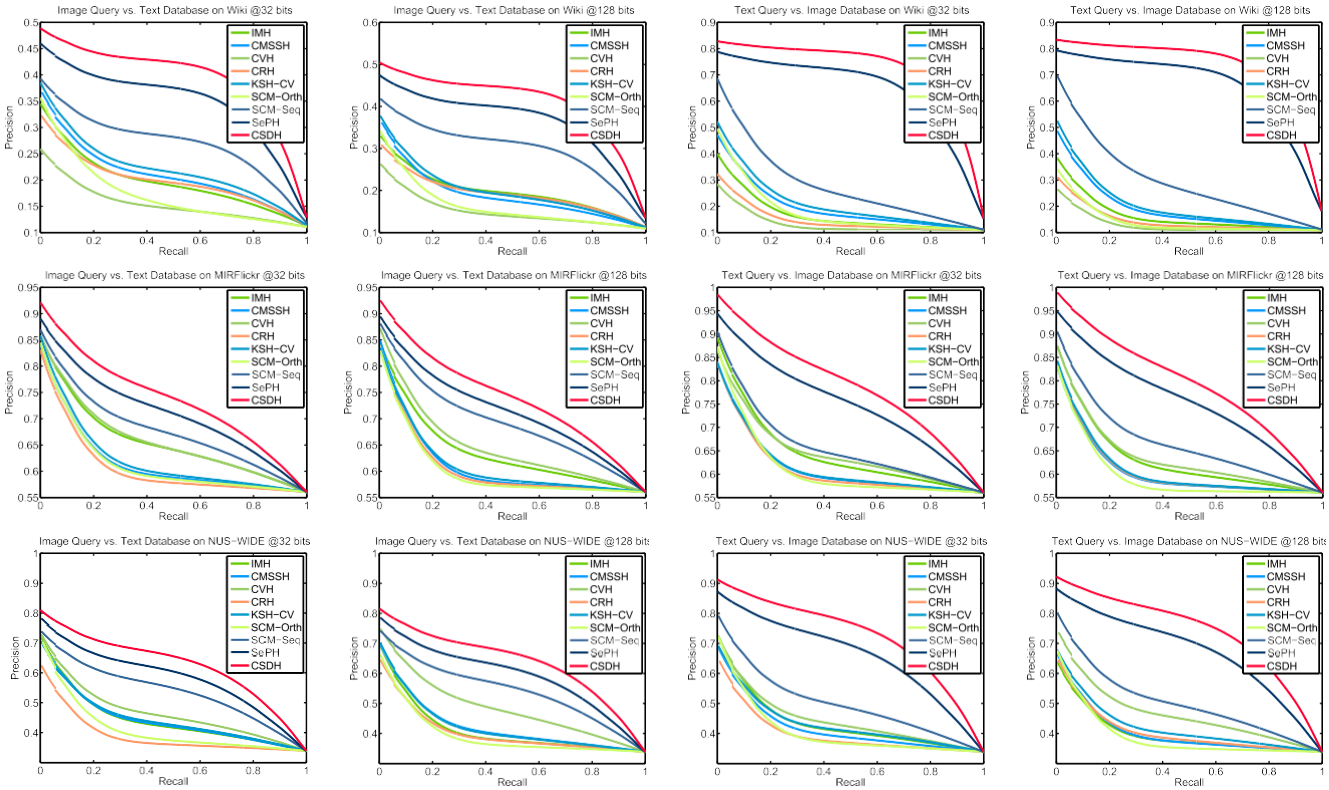


Fig. 1. Comparison of precision recall curves for cross-modal retrieval with different bits on the Wiki, MIRFlickr and NUS-WIDE datasets.

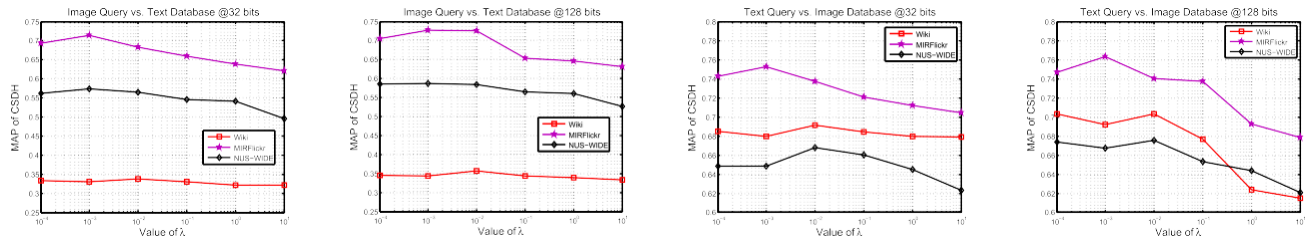


Fig. 2. Parameter sensitivity analysis on λ @32 bits and @128 bits for the Wiki, MIRFlickr and NUS-WIDE datasets.

TABLE II

COMPARISON OF MAP BY USING DIFFERENT UNIFIED CODE LEARNING SCHEMES WITH DIFFERENT HASH CODE LENGTHS ON ALL THREE DATASETS

Image Query vs. Text Database												
Method	**Wiki**				**MIRFlickr**				**NUS-WIDE**			
	16 bits	32 bits	64 bits	128 bits	16 bits	32 bits	64 bits	128 bits	16 bits	32 bits	64 bits	128 bits
CSDH+Random merge	0.2672	0.2904	0.3017	0.3131	0.6438	0.6503	0.6698	0.6787	0.5248	0.5377	0.5482	0.5501
CSDH+Average merge	0.3060	0.3251	0.3317	0.3502	0.6923	0.7104	0.7116	0.7269	0.5592	0.5634	0.5722	0.5793
CSDH+**SVM-weighted merge**	**0.3173**	**0.3377**	**0.3441**	**0.3567**	**0.7032**	**0.7143**	**0.7222**	**0.7274**	**0.5638**	**0.5742**	**0.5808**	**0.5869**
Text Query vs. Image Database												
Method	**Wiki**				**MIRFlickr**				**NUS-WIDE**			
	16 bits	32 bits	64 bits	128 bits	16 bits	32 bits	64 bits	128 bits	16 bits	32 bits	64 bits	128 bits
CSDH+Random merge	0.6016	0.6240	0.6479	0.6521	0.6771	0.6814	0.6992	0.7002	0.5792	0.5917	0.6041	0.6103
CSDH+Average merge	0.6628	0.6777	0.6896	0.7021	0.7257	0.7367	0.7572	0.7609	0.6331	0.6573	0.6653	0.6744
CSDH+**SVM-weighted merge**	**0.6778**	**0.6915**	**0.6986**	**0.7038**	**0.7440**	**0.7532**	**0.7618**	**0.7639**	**0.6530**	**0.6681**	**0.6742**	**0.6760**

"SVM-weighted merge" indicates our proposed method in Algorithm. 2. "Average merge" indicate the $W^m = [0.5, 0.5]$ and $\Omega^m = 0$ in Eq. (19), where $m = 1, \ldots, M$. While "Random merge" indicates $W^m = [a, b]$ and $\Omega^m = 0$ in Eq. (19), where $m = 1, \ldots, M$, a and b are random value, s.t. $0 \leq a \leq 1$, $0 \leq b \leq 1$ and $a + b = 1$.

500 to 4000 for both I2T and T2I tasks. With the training size growing larger than 4000, the performance becomes converged and stable. Fig. 3 reflects our CSDH can achieve satisfactory performance for scalable cross-modality retrieval with only a small set of training samples.

At last, we evaluate the different unified code generating schemes and effects of different binary code initializations for our CSDH. In terms of unified code generating schemes, we introduce our scheme as "SVM-weighted merge" which indicates the proposed method in Algorithm. 2 and

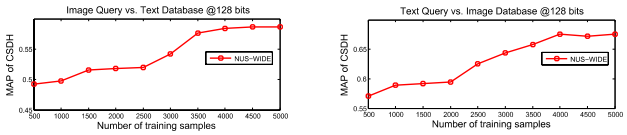


Fig. 3. Effect of training set size of proposed CSDH on NUS-WIDE datasets @128 bits.

TABLE III

EFFECT OF THE DIFFERENT BINARY CODE INITIALIZATIONS ON ALL THREE DATASETS @128 BITS

Datasets	Image Query vs. Text Database		Text Query vs. Image Database	
	Optimized **B** initialization	Random **B** initialization	Optimized **B** initialization	Random **B** initialization
Wiki	0.3567	0.3447	0.7038	0.6894
MIRFlickr	0.7274	0.7151	0.7639	0.7577
NUS-WIDE	0.5869	0.5763	0.6760	0.6701

"Optimized **B** initialization" indicates initializing binary codes bit by bit using Eq. (14) of our paper. "Random **B** initialization" indicates initializing binary codes with random values of -1 or 1.

two comparable baselines: (1) "Average merge" indicates the $W^m = [0.5, 0.5]$ and $K^m = 0$ in Eq. (19), where $m = 1, \ldots, M$; and (2) "Random merge" indicates $W^m = [a, b]$ and $K^m = 0$ in Eq. (19), where $m = 1, \ldots, M$, a and b are random values, s.t. $0 \leq a \leq 1$, $0 \leq b \leq 1$ and $a + b = 1$. Since different schemes lead to different unified codes, we evaluate the retrieval performance (MAP) of the three schemes in Table II for both I2T and T2I tasks. It is discovered that our "SVM-weighted merge" used in CSDH can consistently achieve better results than other baselines on all of datasets and tasks. Despite discarding the weights from different modalities, "Average merge" can still produce acceptable performance. However, for "Random merge", the arbitrary and unbalanced weights cause loss of the retrieval actuaries due to the distortion of the code to some extent. In terms of the binary code initialization, we mainly compare our scheme in Eq. (14) with the random binary code initialization. The results in Table III illustrate our scheme can achieve slightly better retrieval performance on all three datasets for both I2T and T2I tasks. It proves that our CSDH is not very sensitive when applying different binary code initialization.

V. CONCLUSION

In this paper, a novel supervised approach named Cross-modality Sequential Discrete Hashing (CSDH) has been introduced for large-scale similarity retrieval. In the training phase, we learn each bit of codes sequentially with discrete optimization that jointly minimizes its empirical loss based on a boosting strategy. With such a bitwise manner, hash functions are then learned for each modality, mapping the corresponding representations into hash codes. In the test phase, a simple fusion scheme is utilized to generate the unified hash codes for final retrieval by merging the predicted hashing results of an unseen instance from different modalities. Extensive results have shown that our CSDH can outperform state-of-the-art methods in terms of cross-modal retrieval accuracies. Our future work aims to generalize our approach to achieve not only ordinary cross-modularity task but also for large-scale retrieval with partial modalities missing.

REFERENCES

[1] D. Wang, X. Gao, X. Wang, and L. He, "Semantic topic multimodal hashing for cross-media retrieval," in *Proc. Int. Joint Conf. Artif. Intell.*, 2015, pp. 3890–3896.

[2] J. Song, Y. Yang, Z. Huang, H. T. Shen, and J. Luo, "Effective multiple feature hashing for large-scale near-duplicate video retrieval," *IEEE Trans. Multimedia*, vol. 15, no. 8, pp. 1997–2008, Dec. 2013.

[3] M. M. Bronstein, A. M. Bronstein, F. Michel, and N. Paragios, "Data fusion through cross-modality metric learning using similarity-sensitive hashing," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2010, pp. 3594–3601.

[4] S. Kumar and R. Udupa, "Learning hash functions for cross-view similarity search," in *Proc. Int. Joint Conf. Artif. Intell.*, 2011, pp. 1360–1365.

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

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

[7] Y. Zhen and D.-Y. Yeung, "Co-regularized hashing for multimodal data," in *Proc. Adv. Neural Inf. Process. Syst.*, 2012, pp. 1385–1393.

[8] X. Zhu, Z. Huang, H. T. Shen, and X. Zhao, "Linear cross-modal hashing for efficient multimedia search," in *Proc. 21st Int. Conf. Multimedia*, 2013, pp. 143–152.

[9] J. Song, Y. Yang, Y. Yang, Z. Huang, and H. T. Shen, "Inter-media hashing for large-scale retrieval from heterogeneous data sources," in *Proc. ACM Int. Conf. Manage. (SIGMOD)*, 2013, pp. 785–796.

[10] X. Liu, C. Deng, B. Lang, D. Tao, and X. Li, "Query-adaptive reciprocal hash tables for nearest neighbor search," *IEEE Trans. Image Process.*, vol. 25, no. 2, pp. 907–919, Feb. 2016.

[11] X. Liu, L. Huang, C. Deng, J. Lu, and B. Lang, "Multi-view complementary hash tables for nearest neighbor search," in *Proc. IEEE Int. Conf. Comput. Vis.*, Dec. 2015, pp. 1107–1115.

[12] V. Gaede and O. Günther, "Multidimensional access methods," *ACM Comput. Surv.*, vol. 30, no. 2, pp. 170–231, Jun. 1998.

[13] D. Wang, X. Gao, and X. Wang, "Semi-supervised constraints preserving hashing," *Neurocomputing*, vol. 167, pp. 230–242, Nov. 2015.

[14] L. Cao, Z. Li, Y. Mu, and S.-F. Chang, "Submodular video hashing: A unified framework towards video pooling and indexing," in *Proc. 20th ACM Int. Conf. Multimedia*, 2012, pp. 299–308.

[15] X. Liu, X. Fan, C. Deng, Z. Li, H. Su, and D. Tao, "Multilinear hyperplane hashing," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2016, pp. 5119–5127.

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

[17] Y. Liu, F. Wu, Y. Yang, Y. Zhuang, and A. G. Hauptmann, "Spline regression hashing for fast image search," *IEEE Trans. Image Process.*, vol. 21, no. 10, pp. 4480–4491, Oct. 2012.

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

[19] Y. Weiss, A. Torralba, and R. Fergus, "Spectral hashing," in *Proc. Adv. Neural Inf. Process. Syst.*, 2008, pp. 1753–1760.

[20] A. Gionis, P. Indyk, and R. Motwani, "Similarity search in high dimensions via hashing," in *Proc. VLDB*, 1999, pp. 518–529.

[21] 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.

[22] B. Kulis and K. Grauman, "Kernelized locality-sensitive hashing for scalable image search," in *Proc. IEEE 12th Int. Conf. Comput. Vis.*, Oct. 2009, pp. 2130–2137.

[23] Y. Gong, S. Lazebnik, A. Gordo, and F. Perronnin, "Iterative quantization: A procrustean approach to learning binary codes for large-scale image retrieval," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 35, no. 12, pp. 2916–2929, Dec. 2013.

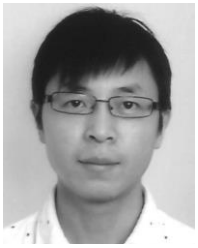
[24] W. Liu, J. Wang, S. Kumar, and S.-F. Chang, "Hashing with graphs," in *Proc. Int. Conf. Mach. Learn.*, 2011, pp. 1–8.

[25] J.-P. Heo, Y. Lee, J. He, S.-F. Chang, and S.-E. Yoon, "Spherical hashing," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2012, pp. 2957–2964.

[26] F. Shen, C. Shen, W. Liu, and H. T. Shen, "Supervised discrete hashing," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2015, pp. 37–45.
[27] F. Shen, W. Liu, S. Zhang, Y. Yang, and H. T. Shen, "Learning binary codes for maximum inner product search," in *Proc. IEEE Int. Conf. Comput. Vis.*, Dec. 2015, pp. 4148–4156.
[28] D. Zhang and W.-J. Li, "Large-scale supervised multimodal hashing with semantic correlation maximization," in *Proc. Assoc. Adv. Artif. Intell. Conf.*, 2014, pp. 2177–2183.
[29] Z. Lin, G. Ding, M. Hu, and J. Wang, "Semantics-preserving hashing for cross-view retrieval," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2015, pp. 3864–3872.
[30] J. Zhou, G. Ding, Y. Guo, Q. Liu, and X. Dong, "Kernel-based supervised hashing for cross-view similarity search," in *Proc. IEEE Int. Conf. Multimedia Expo*, Jul. 2014, pp. 1–6.
[31] G. Ding, Y. Guo, and J. Zhou, "Collective matrix factorization hashing for multimodal data," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2014, pp. 2083–2090.
[32] B. Wu, Q. Yang, W.-S. Zheng, Y. Wang, and J. Wang, "Quantized correlation hashing for fast cross-modal search," in *Proc. 24th Int. Joint Conf. Artif. Intell.*, 2015, pp. 3946–3952.
[33] A. Shrivastava and P. Li, "Asymmetric LSH (ALSH) for sublinear time maximum inner product search (MIPS)," in *Proc. Adv. Neural Inf. Process. Syst.*, 2014, pp. 2321–2329.
[34] G. Lin, C. Shen, Q. Shi, A. V. den Hengel, and D. Suter, "Fast supervised hashing with decision trees for high-dimensional data," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2014, pp. 1971–1978.
[35] L. Liu, M. Yu, and L. Shao, "Projection bank: From high-dimensional data to medium-length binary codes," in *Proc. Int. Conf. Comput. Vis.*, Dec. 2015, pp. 2821–2829.
[36] Z. Cai, L. Liu, M. Yu, and L. Shao, "Latent structure preserving hashing," in *Proc. BMVC*, 2016, pp. 1–19.
[37] L. Liu, M. Yu, and L. Shao, "Latent structure preserving hashing," *Int. J. Comput. Vis.*, 2016, doi: 10.1007/s11263-016-0931-4.
[38] J. Song, Y. Yang, X. Li, Z. Huang, and Y. Yang, "Robust hashing with local models for approximate similarity search," *IEEE Trans. Cybern.*, vol. 44, no. 7, pp. 1225–1236, Jul. 2014.
[39] Y. Gao, M. Shi, D. Tao, and C. Xu, "Database saliency for fast image retrieval," *IEEE Trans. Multimedia*, vol. 17, no. 3, pp. 359–369, Mar. 2015.
[40] J. Yu, D. Tao, M. Wang, and Y. Rui, "Learning to rank using user clicks and visual features for image retrieval," *IEEE Trans. Cybern.*, vol. 45, no. 4, pp. 767–779, Apr. 2015.
[41] J. Zhou, G. Ding, and Y. Guo, "Latent semantic sparse hashing for cross-modal similarity search," in *Proc. Int. ACM SIGIR Conf. Res. Develop. Inf. Retr.*, 2014, pp. 415–424.
[42] L. Liu, M. Yu, and L. Shao, "Unsupervised local feature hashing for image similarity search," *IEEE Trans. Cybern.*, vol. 46, no. 11, pp. 2548–2558, Nov. 2016.
[43] W. Liu, C. Mu, S. Kumar, and S.-F. Chang, "Discrete graph hashing," in *Proc. Adv. Neural Inf. Process. Syst.*, 2014, pp. 3419–3427.
[44] W.-C. Kang, W.-J. Li, and Z.-H. Zhou, "Column sampling based discrete supervised hashing," in *Proc. 30th AAAI Conf. Artif. Intell.*, 2016, pp. 3432–3438.
[45] J. Friedman, T. Hastie, and R. Tibshirani, "Additive logistic regression: A statistical view of boosting (with discussion and a rejoinder by the authors)," *Ann. Statist.*, vol. 28, no. 2, pp. 337–407, 2000.
[46] L. Liu and L. Shao, "Sequential compact code learning for unsupervised image hashing," *IEEE Trans. Neural Netw. Learn. Syst.*, to be published, doi: 10.1109/TNNLS.2015.2495345.
[47] J. Qin, L. Liu, Z. Zhang, Y. Wang, and L. Shao, "Compressive sequential learning for action similarity labeling," *IEEE Trans. Image Process.*, vol. 25, no. 2, pp. 756–769, Feb. 2016.
[48] X. Liu, J. He, C. Deng, and B. Lang, "Collaborative hashing," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2014, pp. 2147–2154.
[49] J. Malick, J. Povh, F. Rendl, and A. Wiegele, "Regularization methods for semidefinite programming," *SIAM J. Optim.*, vol. 20, no. 1, pp. 336–356, 2009.
[50] L. Liu and L. Shao, "Discriminative partition sparsity analysis," in *Proc. 22nd Int. Conf. Pattern Recognit. (ICPR)*, Aug. 2014, pp. 1597–1602.
[51] G. McLachlan and D. Peel, *Finite Mixture Models*. New York, NJ, USA: Wiley, 2004.
[52] P. Viola and M. Jones, "Rapid object detection using a boosted cascade of simple features," in *Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Dec. 2001, pp. 511–518.
[53] N. Rasiwasia *et al.*, "A new approach to cross-modal multimedia retrieval," in *Proc. Int. Conf. Multimedia*, 2010, pp. 251–260.
[54] D. G. Lowe, "Distinctive image features from scale-invariant keypoints," *Int. J. Comput. Vis.*, vol. 60, no. 2, pp. 91–110, 2004.
[55] M. J. Huiskes and M. S. Lew, "The MIR flickr retrieval evaluation," in *Proc. ACM Int. Conf. Multimedia Inf. Retr.*, 2008, pp. 39–43.
[56] T.-S. Chua, J. Tang, R. Hong, H. Li, Z. Luo, and Y. Zheng, "Nus-wide: A real-world Web image database from National Uni- versity of Singapore," in *Proc. ACM Int. Conf. Image Video Retr.*, 2009.

Li Liu received the Ph.D. degree from the Department of Electronic and Electrical Engineering, The University of Sheffield, Sheffield, U.K., in 2014. He is currently a Research Fellow with the Department of Computer and Information Sciences, Northumbria University, Newcastle upon Tyne, U.K.

Zijia Lin (S'13) received the B.Sc. degree from the School of Software, Tsinghua University, Beijing, China, in 2011, and the Ph.D. degree from the Department of Computer Science and Technology, Tsinghua University, in 2016.



Ling Shao (M'09–SM'10) is currently a Professor with the Department of Computer and Information Sciences, Northumbria University, Newcastle upon Tyne, U.K., and a Visiting Professor with Southwest University, Chongqing, China. He is an Associate Editor of the IEEE TRANSACTIONS ON IMAGE PROCESSING, the IEEE TRANSACTIONS ON NEURAL NETWORKS AND LEARNING SYSTEMS, and several other journals.

Fumin Shen received the Ph.D. degree from the Nanjing University of Science and Technology, China, in 2014. He is currently an Associate Professor with the School of Computer Science and Engineering, University of Electronic Science and Technology of China, China.

Guiguang Ding (M'13) received the Ph.D. degree in electronic engineering from the University of Xidian, Xi'an, China. He is currently an Associate Professor with the School of Software, Tsinghua University.

Jungong Han received the Ph.D. degree in telecommunication and information system from the University of Xidian, Xi'an, China, in 2004. He is currently a Senior Lecturer with the Department of Computer Science and Digital Technologies, Northumbria University, Newcastle upon Tyne, U.K.