Computer Vision and Image Understanding 124 (2014) 12–21

Contents lists available at ScienceDirect

Computer Vision and Image Understanding

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

Semi-supervised multi-graph hashing for scalable similarity search

Jian Cheng [a,*], Cong Leng [a], Peng Li [a], Meng Wang [b], Hanqing Lu [a]

[a] National Laboratory of Pattern Recognition, Institute of Automation, Chinese Academy of Sciences, Beijing 100190, China
[b] School of Computer and Information, Hefei University of Technology, Hefei 230009, China

ARTICLE INFO

Article history:
Received 18 August 2013
Accepted 1 April 2014

Keywords:
Hashing
Multiple graph learning
Multiple modality
Semi-supervised learning

ABSTRACT

Due to the explosive growth of the multimedia contents in recent years, scalable similarity search has attracted considerable attention in many large-scale multimedia applications. Among the different similarity search approaches, hashing based approximate nearest neighbor (ANN) search has become very popular owing to its computational and storage efficiency. However, most of the existing hashing methods usually adopt a single modality or integrate multiple modalities simply without exploiting the effect of different features. To address the problem of learning compact hashing codes with multiple modality, we propose a semi-supervised Multi-Graph Hashing (MGH) framework in this paper. Different from the traditional methods, our approach can effectively integrate the multiple modalities with optimized weights in a multi-graph learning scheme. In this way, the effects of different modalities can be adaptively modulated. Besides, semi-supervised information is also incorporated into the unified framework and a sequential learning scheme is adopted to learn complementary hash functions. The proposed framework enables direct and fast handling for the query examples. Thus, the binary codes learned by our approach can be more effective for fast similarity search. Extensive experiments are conducted on two large public datasets to evaluate the performance of our approach and the results demonstrate that the proposed approach achieves promising results compared to the state-of-the-art methods.

1. Introduction

Recently, with the rapid advances of digit devices and the Internet, the amount of multimedia data (e.g. images, videos) are explosively increasing. These huge databases have posed a significant challenge in terms of scalable similarity search to many multimedia applications, such as content based multimedia retrieval (CBMR) [1], and classification.

In order to perform nearest neighbor search, many content based multimedia retrieval applications exhaustively compare the query with each sample in the database, which is infeasible for large-scale cases because the linear complexity is not scalable in practical situations. Besides, many large-scale CBMR applications suffer from the curse of dimensionality since feature descriptors usually have hundreds or even thousands of dimensions. Therefore, beyond the infeasibility of exhaustive search, storage of the original data also becomes a challenging problem. Fortunately, in CBMR and many other applications, finding approximate nearest neighbors is sufficient. To overcome these problems, a lot of research efforts have been devoted to investigate the alternative solution – approximate nearest neighbor (ANN) search, which trades off a little search accuracy to greatly speed up the search process.

Over the past decades, many advances have been made on ANN techniques for large-scale applications. In the earlier works, many tree-based methods [2–4] can perform similarity search effectively for low-dimensional data. However, for the high-dimensional cases and applications, their performance is not satisfactory and does not guarantee faster search compared to linear scan methods. Therefore, hashing techniques have been actively studied to provide efficient solutions for such high-dimensional data in recent years [5–7]. Hashing-based methods are promising in accelerating similarity search for their capability of generating compact binary codes for a large number of images in the dataset so that similar images will be mapped to close binary codes. Retrieving similar neighbors is then accomplished simply by finding the images that have codes with a small Hamming distance from the query, which is extremely fast to calculate. In addition, the binary codes representation can save much more storage space than the feature descriptors for large databases.

Many hashing approaches have been developed in recent years. A notable hashing method is Locality Sensitive Hashing (LSH) [5,6] which projects the data point to a random hyperplane and then conducts random thresholding. Inspired by spectral clustering,

* Corresponding author.
E-mail address: jcheng@nlpr.ia.ac.cn (J. Cheng).

http://dx.doi.org/10.1016/j.cviu.2014.04.001
1077-3142/

Spectral Hashing (SH) [7] and its extensions [8–10] attempt to apply machine learning approaches to find good data-aware hash functions. All of these hashing methods have shown success in scalable similarity search. However, most of the existing hashing methods only adopt single feature modality to learn the binary codes in hamming space. As we all know, the real-world images are often represented by multiple modalities, such as different kinds of visual features, and the importance of the roles that different modalities play can also be significantly different in hash function learning.

In recent years, more and more researchers are attracted to study how to fuse multiple modality to improve the performance of hashing algorithms [11]. Zhang et al. proposed a *Composite Hashing* (CHMS) [12]. In CHMS, one graph is established for each source and then combine them to learn linear hashing functions for each source. Song et al. presented a *Multiple Feature Hashing* (MFH) [13] to tackle the near-duplicate video retrieval problem. Liu et al. formulated the hashing problem as a similarity preserving hashing with linearly combined multiple kernels [14]. Xia et al. extended KLSH to Multi-Kernel Locality Sensitive Hashing (MKLSH) [15] by using multiple kernels. However, most of the multi-modal hashing methods which employ graph learning might suffer from the high complexity of large graph construction and most of them learn the hash functions in an unsupervised manner, which cannot effectively preserve the semantic similarity. There exist supervised hashing methods that can handle such semantic similarity but they are prone to overfitting when labeled data is small or noisy.

In this paper, we propose a semi-supervised Multi-Graph Hashing (MGH) approach for scalable similarity search. Different form the traditional methods, our approach can effectively integrate the multiple modalities with optimized weights in a multi-graph learning framework. In this way, the effects of different modalities can be adaptively modulated. It is worth to note that, in the existing multi-view (modality) based hashing method, [12,13] are also graph based, but they treat each graph equally. Besides, in our approach, anchor graph is employed for fast large graph construction and semi-supervised information is also incorporated into the unified learning scheme. Thus, the binary codes learned by our method can be more effective for fast similarity search. Fig. 1 illustrates the flowchart of our proposed hashing approach. The characteristics of our work can be summarized as follows:

(1) We propose a semi-supervised multi-graph hashing framework for binary code learning in scalable similarity search.
(2) The effects of different modalities can be adaptively modulated in our unified learning scheme. Therefore, our hashing approach can integrates different sources of information with optimized weights.

The remainder of this paper is organized as follows. In Section 2, we give a brief review of the related work. Section 3 describes our semi-supervised Multi-Graph Hashing (MGH) approach in detail. We conduct extensive experiments on two large publicly available datasets in Section 4. Finally, conclusion and future work are given in Section 5.

2. Related Work

There has been extensive research on fast similarity search due to its critical importance in many information retrieval and computer vision fields. For a low-dimensional feature space, similarity search can be carried out efficiently by some tree-based methods [2–4], e.g., KD-tree, M-tree. They usually partition the data space recursively to implement an exact similarity search in the low-dimension feature space. However, the high dimensionality will significantly corrupt the efficiency of tree-based methods, and make them even perform worse than the naive methods (e.g. linear scan) [16]. Thus, they will encounter difficulties in practical applications where the number of dimensions may be hundreds or even thousands. To overcome this issue, hashing-based approximate nearest neighbor (ANN) search approaches have become more and more popular recently. By encoding the high-dimensional data points into binary codes in a low-dimensional Hamming space, hashing methods provide compact data representation and efficient indexing mechanism for scalable similarity search. In general, we summarize them into two categories, single-view hashing and multi-view hashing.

2.1. Single-view hashing

Most of the existing hashing work utilize a single view to generate binary codes. They can be broadly categorized into two groups: data-independent and data-dependent schemes.

One of the most well known data-independent methods is Locality Sensitive Hashing (LSH) [5,6], which utilizes a batch of locality sensitive functions to embed the data into Hamming space. The LSH algorithm typically guarantees the high probability for any two similar samples falling into the same bucket and two dissimilar samples into the different buckets. One popular scheme in LSH is to generate a batch of random vectors that preserves inner-product similarity from a particular probabilistic distribution [17]. LSH has been extended to several similarity measures, such as, p-norm distances for $p \in (0,2]$ [18], the Mahalanobis distance [19,20] and the kernel similarity [21]. However, since the random vector is data-independent, LSH may lead to quite inefficient codes in practice as it requires multiple tables with long

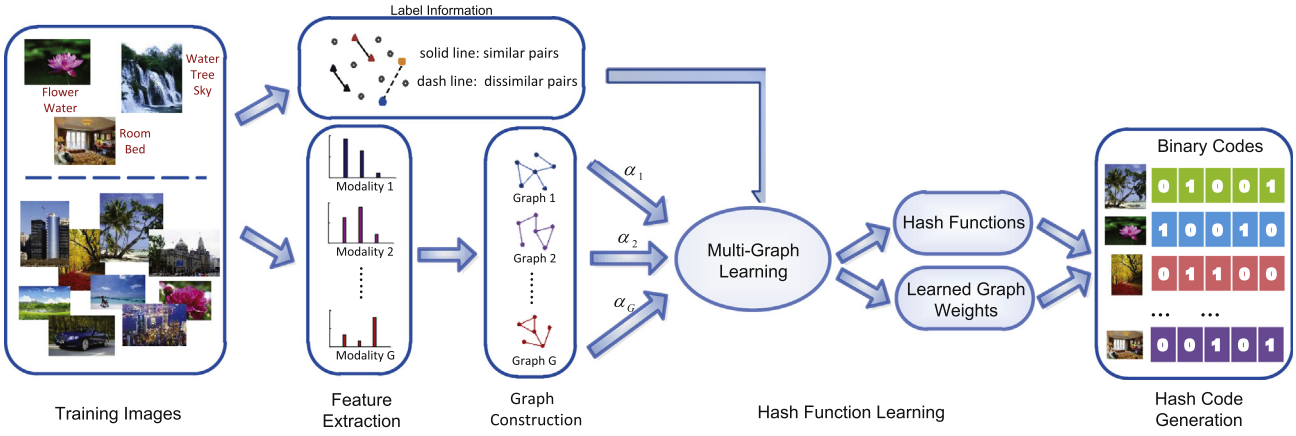

Fig. 1. The flowchart of our proposed semi-supervised Multi-Graph Hashing (MGH) approach.

codes [7] to guarantee reasonable recall, which leads to longer query time and increase in storage.

Recent research attentions have been shifted to data-dependent hashing techniques [7,22–27,9,28,29], which attempt to apply machine learning approaches rather than random projections to find better data-aware hash functions. One famous approach among these techniques is Spectral Hashing (SH) [7], which calculates the bits by thresholding a subset of eigenvectors of the Laplacian matrix of the similarity graph. SH has shown significant improvement over conventional hashing methods. Inspired by SH, He et al. [29] proposed a kernel based hashing algorithm which could be explicitly represented and optimized. Mu et al. [30] presented a kernel-based hashing method in weakly-supervised learning approach. A self-taught scheme is proposed in [31] to train SVM for unseen samples. A hypergraph approach and a semantic graph method are adopted for binary code learning in [32,8], respectively. Anchor Graph Hashing (AGH) is proposed in [9], which adopts a fast and efficient anchor graph for hash function learning. Gong and Lazebnik [27] proposed an alternating minimization scheme for finding a rotation to minimize the quantization error of binary codes and got the state-of-the-art retrieval performance. (Semi-)Supervised information is incorporated into newly proposed hashing methods, such as [10,25,33], and it is demonstrated that supervised information is conducive to preserve semantic affinity. Deep Learning framework (RBMs) is used to generate binary codes in [22,24,34]. Unlike most of the hashing methods which focus on the projection stage, [27,35–38] focus on the quantization stage of hashing and achieve superior performance.

2.2. Multi-view hashing

In many real-world applications the data is often derived from serval different sources, so they can be represented by different kinds of features. For instance, in CBIR application, the images can be described by local features (e.g. SIFT descriptor [39]) and global features (e.g. RGB color histogram). In recent years, some hashing algorithms have been proposed to encode data represented by different features. One branch is *Cross-View Hashing*, which focuses on the task of similarity search across the views. For example, given an image, one may want to find some text articles which describe the image. Bronstein et al. [40] proposed a framework for supervised similarity learning based on embedding the input data from two arbitrary spaces into the Hamming space. Kumar and Udupa [41] proposed a principled method for learning a hash function for each view based on spectral hashing. Zhen and Yeung [42] proposed a probabilistic model to learn hash functions from multimodel data automatically.

Another branch of multi-view hashing is to fuse multiple information source to get more efficient and effective hashing code. To the best of our knowledge, Zhang et al. proposed the first related model, called *Composite Hashing* (CHMS) [12]. In CHMS, one graph is established for each source and then combine them to learn linear hashing functions for each sources. Song et al. presented a *Multiple Feature Hashing* (MFH) [13] to tackle the near-duplicate video retrieval problem. Similar to CHMS, MFH established one graph for each view. MFH and CHMS differ in the output hashing functions: In CHMS the output hashing functions are the combination of these learned linear hash functions on different sources [12]. However, MFH concatenates all features into a long vector and projects it using the learned hashing functions. Liu et al. utilize kernel trick to capture the similarity affinity of different sources (MFKH) [14]. Concatenating different features in kernel space, in MFKH, the hashing problem is formulated as a similarity preserving hashing with linearly combined multiple kernels [14]. Xia et al. extended KLSH to Multi-Kernel Locality Sensitive Hashing (MKLSH) [15] by using multiple kernels and a boosting scheme is utilized to greedily find a good solution. In [11], Yang et al. effectively explored the information contained in multiple features of both labeled and unlabeled data for multimedia content analysis and propose a new semi-supervised multi-feature learning algorithm.

The closest work to ours might be CHMS [12]. CHMS is graph based algorithm and also constructs one graph for each source. However, it does not exploit the individual impact of different graphs and treats them equally. In addition, it suffers from the high time complexity on large graph construction and computing the eigenvectors of the graph Laplacian. CHMS learns the hash functions in an unsupervised manner, which may lead to undiscriminating binary codes. Our work differs from CHMS in that we treat each graph differently and an optimized weight can be learned automatically. In our work, supervised information is also incorporated and a sequential learning scheme is developed to learn the hash functions one by one.

3. Semi-supervised multi-graph hashing

The conventional hashing methods usually either adopt a single modality or integrate multiple modalities simply without exploiting the effect of different modalities. In this section, we will introduce our semi-supervised Multi-Graph Hashing (MGH) approach, in which the effects of different modalities can be adaptively modulated in a unified scheme.

3.1. Graph construction and hash function

One main step for the graph-based hashing methods, such as SH and its extensions, is to build a neighborhood graph $\Omega = \{X, W\}$ on the training dataset X, where W is the similarity matrix. However, this procedure takes $O(dn^2)$ (d is the feature dimension and n is the number of data points) time complexity, which is intractable for large-scale databases. In this paper, we adopt anchor graph [43] to speed up graph construction and computation, which can be built in $O(n)$ time and has a powerful approximation to the true k-NN graph.

The anchor graph uses a small set of $m (m \ll n)$ points called anchors to approximate the data neighborhood structure. First, we generate m anchor points $\{a_j\}_{j=1}^{m}$ from the training dataset, which can be done by running K-means on a small subsample of the database with very few iterations. Then, we compute a truncated similarity matrix $Z \in \mathbb{R}^{n\times m}$ between the training data and the anchor points as follows:

$$Z_{ij} = e^{-\|x_i - a_j\|^2/\sigma} \tag{1}$$

Only the s nearest anchors are computed for each data point x_i in Z and the sum of each row in Z is normalized to 1. With the help of Z, the anchor graph provides a powerful approximation to the original similarity matrix W as $\widehat{W} = Z\Lambda^{-1}Z^T$ where $\Lambda = \mathrm{diag}(Z^T\mathbf{1}) \in \mathbb{R}^{m\times m}$ [43].

Since we have G different modalities in our approach, we can obtain the matrix $Z^g (1 \leqslant g \leqslant G)$ for each modality respectively. Finally, we define the form of hash function from G different modalities as follows:

$$Y = \mathrm{sign}\left(\sum_{g=1}^{G}\alpha_g Z^g V\right) \tag{2}$$

$$s.t. \sum_{g=1}^{G}\alpha_g = 1 \quad \text{and } \alpha_g \geqslant 0$$

where $Y \in \mathbb{R}^{n\times K}$ is the code matrix and $V \in \mathbb{R}^{m\times K}$ is the projection mapping the data points to Hamming space. K denotes the number of hash bits and α_g is the weight parameter for the gth modality.

3.2. Problem formulation

3.2.1. Multi-graph locality preservation

A common criteria adopted by most of graph-based hashing methods is to make the similar data points map to close codes within a short Hamming distance. We deal with multiple modalities in our approach and, for the gth modality, we define

$$Y^g = sign(Z^g V)$$

Here, it is worth to note that Y^g is just an intermediate variable for convenient description, and we do not need to compute it. We also aim to preserve the local structure of the data on the graph as in [7]:

$$\min_{y^g} \sum_{i,j=1}^{n} W_{ij}^g \| y_i^g - y_j^g \|^2$$

With simple matrix transformation and substituting the Eq. (2), the above equation can be rewritten as follows:

$$\begin{aligned} &\min_{Y^g} \ \mathrm{Tr}(Y^{gT} L^g Y^g) \\ &s.t.\ V^T V = I \end{aligned} \tag{3}$$

where $L^g = D^g - W^g$ is called the graph Laplacian and D^g is a diagonal matrix whose elements are the sum of each row in W^g. However, constructing graph and solving the eigen system of L^g is quite time consuming for large dataset. Since we employ anchor graph in our approach, the graph Laplacian of anchor graph is represented as

$$L^g = I - \widehat{W}^g = I - Z^g \Lambda^{g-1} Z^{gT} \tag{4}$$

Then, after substituting Eq. (4) into Eq. (3) and relaxing the sign$(\cdot)$ symbol for Y^g, the above minimization problem can be rewritten as

$$\begin{aligned} \min_{Y^g} \mathrm{Tr}\left(Y^{gT} L^g Y^g\right) &= \min_{V} \mathrm{Tr}\left(V^T Z^{gT} (I - Z^g \Lambda^{g-1} Z^{gT}) Z^g V\right) \\ &= \min_{V} \mathrm{Tr}\left(V^T \widetilde{L}^g V\right) \end{aligned} \tag{5}$$

where $\widetilde{L}^g = Z^{gT} Z^g - Z^{gT} Z^g \Lambda^{g-1} Z^{gT} Z^g$, $(\widetilde{L}^g \in \mathbb{R}^{m \times m})$ is called the reduced graph Laplacian, which can be computed and solved very fast.

Since G modalities are used for data representation, we can build G different graphs in total. By integrating the different graphs together with their weighting parameter, we have the following minimization objective function

$$\begin{aligned} &\min_{V,\alpha} \sum_{g=1}^{G} \alpha_g \mathrm{Tr}\left(V^T \widetilde{L}^g V\right) = \min_{V,\alpha} \mathrm{Tr}\left(V^T \sum_{g=1}^{G} \alpha_g \widetilde{L}^g V\right) \\ &s.t.\ \ V^T V = I, \quad \sum_{g=1}^{G} \alpha_g = 1 \quad \text{and } \alpha_g \geqslant 0 \end{aligned} \tag{6}$$

3.2.2. Semi-supervised information

The different modalities used to construct the graphs in our paper are all from different visual features. Considering the issue of semantic gap, the graphs built purely on visuals features sometimes may not reflect the intrinsic structure of the data. In order to leverage semantic similarity of the binary codes, we incorporate a little semi-supervised information into the hash function learning process. Besides, with the help of semi-supervised information, an effective sequential learning scheme can be employed to learn complementary hash functions [10].

We randomly select p samples from the training set as labeled data and a semantic matrix $S \in \mathbb{R}^{p \times p}$ is defined on the labeled set. S_{ij} is set to 1 if two images x_i and x_j have the same label and S_{ij} is set to -1 otherwise. Code inner product [33] is used to measure the similarity of two learned codes y_i and y_j. The inner product of two codes should be large if the two images have the same label and vice versa. Therefore, we have the following maximization problem:

$$\max_{y_{i,j}} \sum_{i,j=1}^{p} S_{ij} y_i y_j^T = \max_{Y_{label}} \mathrm{Tr}\left(Y_{label}^T S Y_{label}\right) \tag{7}$$

By replacing Y_{label} with the definition in Eq. (2) and relaxing the sign$(\cdot)$ symbol, we can get

$$\begin{aligned} &\max_{V} \mathrm{Tr}\left(\left(\sum_{g=1}^{G} \alpha_g Z_{label}^g V\right)^T S \left(\sum_{g=1}^{G} \alpha_g Z_{label}^g V\right)\right) \\ &\qquad = \max_{V} \mathrm{Tr}\left(V^T \left(\sum_{i,j=1}^{G} \alpha_i \alpha_j Z_{label}^{iT} S Z_{label}^{j}\right) V\right) \\ &s.t.\ V^T V = I \end{aligned} \tag{8}$$

where $Z_{label}^g \in \mathbb{R}^{p \times m}$ is the mapping representation of the p labeled images on the m anchor points.

3.2.3. The Final Objective Function

Combining the multi-graph smoothness in Eq. (6) and semi-supervised information in Eq. (8) together, we can obtain the final objective of our approach as follows:

$$\begin{aligned} Q &= \min_{V,\alpha} \left(\mathrm{Tr}\left(V^T \sum_{g=1}^{G} \alpha_g \widetilde{L}^g V\right) - \lambda \mathrm{Tr}\left(V^T \left(\sum_{i,j=1}^{G} \alpha_i \alpha_j Z_{label}^{iT} S Z_{label}^{j}\right) V\right)\right) \\ &= \min_{V,\alpha} \mathrm{Tr}\left(V^T M V\right) \\ &s.t.\ V^T V = I, \quad \sum_{g=1}^{G} \alpha_g = 1 \quad \text{and } \alpha_g \geqslant 0 \end{aligned} \tag{9}$$

where $M = \sum_{g=1}^{G} \alpha_g \widetilde{L}^g - \lambda \sum_{i,j=1}^{G} \alpha_i \alpha_j Z_{label}^{iT} S Z_{label}^{j}$ and λ is a balanced parameter. The hash projection V and weights α for different modalities are to be learned. We adopt an alternative optimization to solve Eq. (9). More specifically, we alternatively update V and α to optimize the objective function.

3.3. Optimization

3.3.1. Update V with fixed α

Now we consider the optimization of V. If α is fixed, the solutions of V can be obtained by eigenvalue decomposition in a single shot by extracting the first K smallest eigenvectors of M in Eq. (9). However, for most real-world datasets, most of variance (information) is contained in top eigenvectors and the remain eigenvectors usually contain less information or even noise. This will force one to progressively pick those directions that have very low variance, substantially reducing the quality of lower bits, and hence the whole embedding [10]. Besides, the hash functions learned in this way do not have the error correcting ability. In this paper, we adopt a sequential learning scheme to learn the K hash functions one by one.

Suppose we have learned the kth hash function v_k, which is the kth column of V, we can compute the corresponding bit of binary code for all the p labeled images $Y_{label}(:,k)$. Then we define a $p \times p$ matrix in the form of $C = Y_{label}(:,k) \cdot Y_{label}(:,k)^T$. As the bit value is -1 or +1, $C_{ij} = 1$ if the two images are projected to have the same code by v_k and $C_{ij} = -1$ otherwise. By comparing the sign in C with the groundtruth semantic matrix S, we can find out the image pairs that are wrongly mapped by the vector v_k. We update the matrix S by increasing the weights of incorrectly mapped pairs as follows

$$S_{ij}^{new} = S_{ij}^{old} - \eta C_{ij} \ \text{if} \ S_{ij} \cdot C_{ij} < 0 \tag{10}$$

Table 1
The Hamming ranking performance of different algorithms with different code length on the two datasets. (**Mean Average Precision (MAP)** is reported.)

Methods	MNIST				NUS-WIDE			
	32-bits	64-bits	96-bits	128-bits	32-bits	64-bits	96-bits	128-bits
LSH	0.2585	0.3218	0.3624	0.3729	0.4068	0.3868	0.3983	0.4099
SH	0.2642	0.2493	0.2509	0.2507	0.3622	0.3544	0.3531	0.3567
AGH	0.3621	0.3172	0.3001	0.2898	0.4167	0.4061	0.4005	0.3964
ITQ	0.4416	0.4531	0.4677	0.4662	0.4119	0.4173	0.4200	0.4223
SPLH	0.5029	0.5517	0.5676	0.5602	0.5205	0.5176	0.5120	0.5074
CHMS	0.2974	0.3318	0.2812	0.2602	0.3846	0.3720	0.3668	0.3636
MFKH	0.1505	0.2264	0.2472	0.2807	0.3689	0.3627	0.3414	0.3419
MGH_uw	0.6259	0.6683	0.6883	0.6869	0.5260	0.5368	0.5346	0.5342
MGH	0.6619	0.7058	0.7149	0.7170	0.5318	0.5378	0.5407	0.5421
l_2 Scan	0.4170				0.3979			

where $\eta > 0$ is a control parameter. As we can see, the update of S only changes the magnitude of the elements while the semantic relationships (the sign) remain unchanged all the time. By the updating Eq. (10), we expect that more attention can be paid to the incorrectly projected image pairs when learning the next hash function v_{k+1}. With the updated S^{new}, we can compute the new matrix M^{new} in Eq. (9). After removing the component of v_k from M^{new}, we extract the first smallest eigenvector of M^{new} as v_{k+1}. The above updating process is repeated until all the K hash functions (V) are obtained.

3.3.2. Update α with fixed V

Now we introduce how to update α with fixed V. After the projection matrix V is learned, the weighting parameters for different modalities can be optimized by minimizing the following objective function:

$$\min_{\alpha} \left(\sum_{g=1}^{G} \alpha_g T_g - \lambda \sum_{i,j=1}^{G} \alpha_i \alpha_j B_{ij} \right) \quad (11)$$
$$s.t. \sum_{g=1}^{G} \alpha_g = 1 \quad \text{and } \alpha_g \geqslant 0$$

where T_g, B_{ij} can be regarded as constants, and

$$T_g = \mathrm{Tr}\left(V^T \widetilde{L}^g V\right) \quad (12)$$

$$B_{ij} = \mathrm{Tr}\left(V^T Z^{iT}_{label} S Z^{j}_{label} V\right) \quad (13)$$

This is constrained quadratic programming problem, which is easy to solve. We first relax the equality constraint. By using the KKT conditions for the inequality constraint, we can get the updating rules for α as

$$\alpha_i^* = \frac{\sum_{g=1}^{G} \alpha_g (B_{gi} + B_{ig})}{T_i} \alpha_i \quad (14)$$

After the solutions for all the α are obtained, we normalize the sum of them to 1, making the equality constraint to be satisfied.

By alternatively updating V and α for a few iterations, we can get the final solutions for our approach. We describe the whole flowchart of the proposed semi-supervised Multi-Graph Hashing (MGH) approach in Algorithm 1.

3.4. Out-of-sample extension

After the hash functions V and the weighting parameters α for different modalities are learned, we also need to know how to obtain the binary codes for query images. For a new sample, we first transform it to $z^g_{query} (g = 1, \ldots G)$ by computing its similarity to the anchor points in all the modalities. Then we can get the binary code for the query image as

$$y_{query} = \mathrm{sign}\left(\sum_{g=1}^{G} \alpha_g z^g_{query} V \right) \quad (15)$$

Algorithm 1. Semi-Supervised Multi-Graph Hashing Algorithm

Input: training data X and labeled data X_{label} in G modalities, m anchor points, semantic matrix S, number of bits K, number of iterations H, parameters λ, η.
Output: hash projections V and weights for the different modalities α_g $(g = 1, \ldots G)$
Compute matrix Z^g and Z^g_{label} $(g = 1, \ldots G)$ by Eq. (1)
Compute the reduced graph Laplacian $\widetilde{L}^g$ with Z^g
Initialize $\alpha_g = \frac{1}{G}$ and $v_0 = 0$
for *iter* $= 1$ **to** H **do**
  1. Update V with fixed α
    for $k = 1$ **to** K **do**
      Compute matrix M in Eq. (9)
      Remove the component of previous projection v_{k-1} from M : $M = M - v_{k-1} v_{k-1}^T$
      Extract the first smallest eigenvector of M as v_k
      Update the semantic matrix S^{new} by Eq. (10)
    end for
    Set $V = [v_1, v_2, \ldots, v_k]$
  2. Update α with fixed V
    Compute T and B by Eqs. (12) and (13)
    Update α by Eq. (14)
    Normalize the sum of α's to 1
end for

4. Experiments

In this section, we evaluate the performance of our semi-supervised Multi-Graph Hashing (MGH) approach by conducting extensive experiments and compare it with the state-of-art methods.

4.1. Datasets and protocols

We perform various experiments with the following two large datasets:

- **MNIST**[1]**:** A handwritten digit image dataset, including ten classes (0–9). It consists of 70,000 images in total and each image is 28×28 in resolution. In order to obtain the multimodal representations for this dataset, we employ three dimensionality reduction methods (PCA, LPP, NPE) to reduce the feature dimensions to 200, respectively, which results in three different modalities.

[1] http://yann.lecun.com/exdb/mnist/.

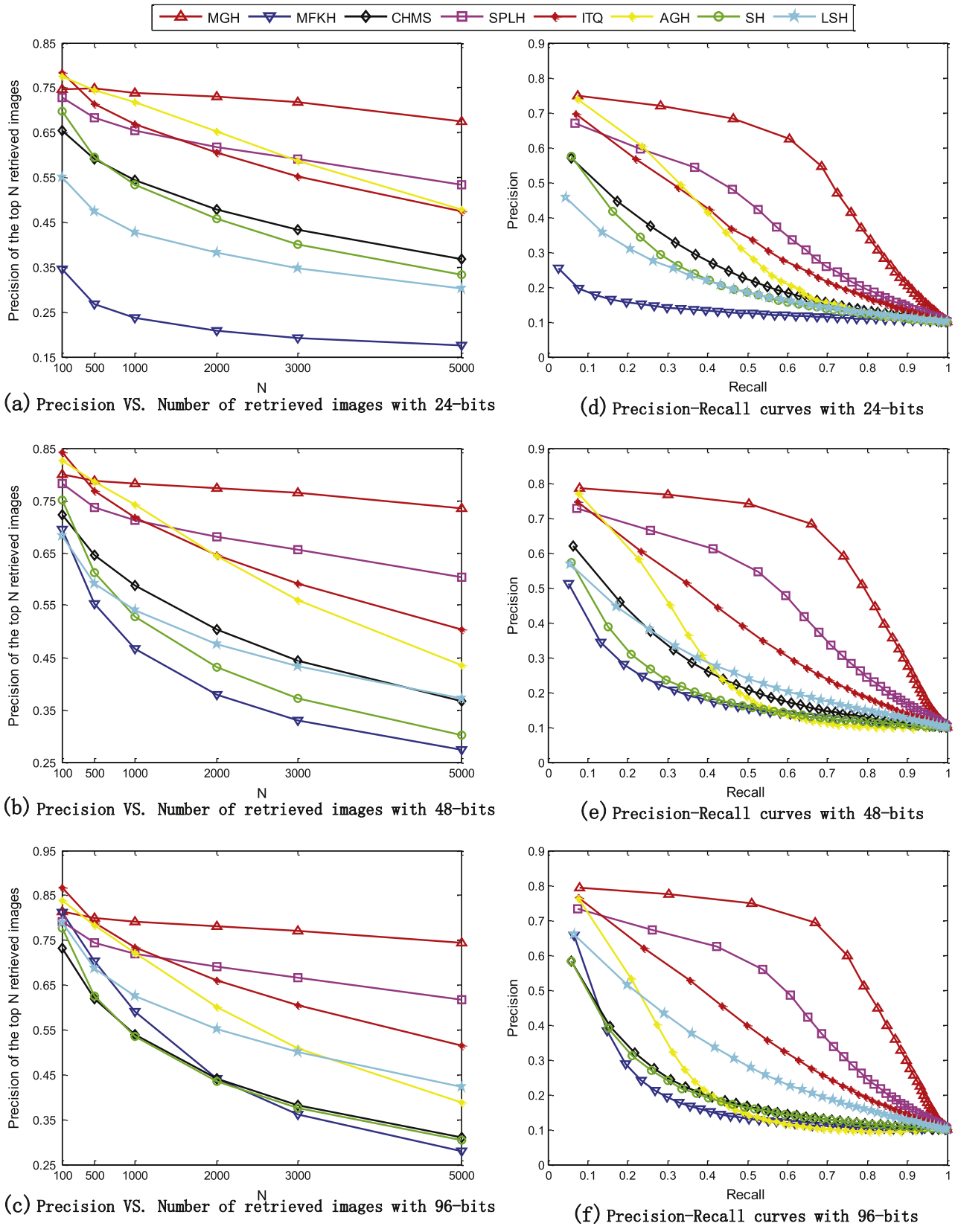

Fig. 2. Hamming ranking: comparison results of different hashing methods on the MNIST dataset. (a)–(c): Precision vs. number of retrieved images with 24, 48, 96 bits. (d)–(f): Precision–Recall curves with 24, 48, 96 bits.

We randomly select 1000 images as queries and the left 69,000 images are used for training.

- **NUSWIDE:** A large-scale web image dataset which contains 270,000 images from Flickr [44]. We use five different modalities that come from five types of visual features: 64-D color his-

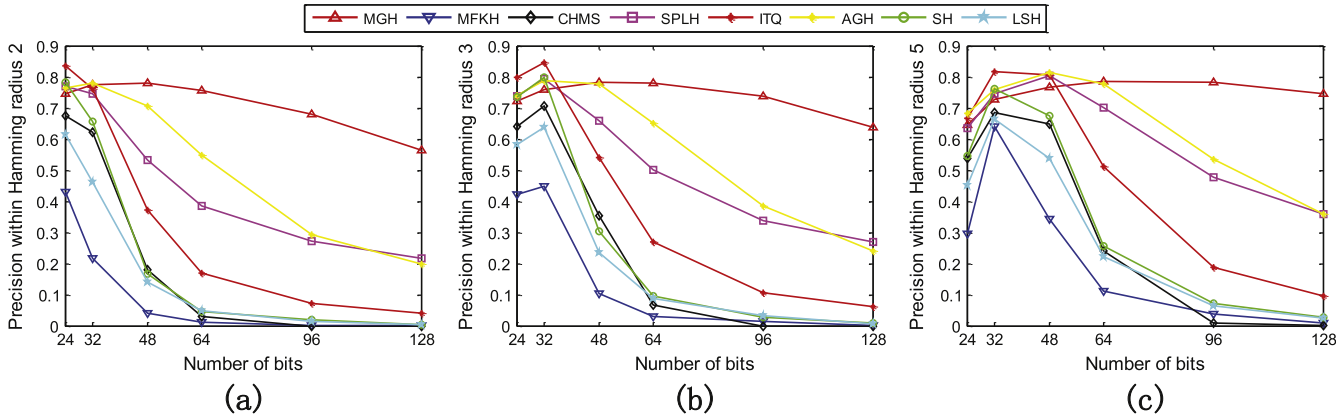

Fig. 3. Hashing lookup: comparison results of different hashing methods on the MNIST dataset. (a)–(c) Mean precision within Hamming radius 2, 3, 5 for hash lookup.

togram, 144-D color correlogram, 73-D edge direction histogram, 128-D wavelet texture, and 225-D block-wise color moments. In this dataset, 81 concepts are labeled which can be used as ground truth for evaluation. Two images are considered to be similar when they share at least 1 concept. 2000 images randomly chosen from the most frequent 20 concepts are used as queries and the remained are used to learn the hash functions.

To perform fair evaluation, we adopt two criteria commonly used in the literature: *Hamming Ranking* and *Hashing Lookup*, which focus on different characteristics of hashing techniques. Precision and recall are recorded for both of the two evaluation methods. For Hamming ranking, all the images in the training set are ranked according to their Hamming distance from the query and the desired neighbors are returned from the top of the ranked list. For hash lookup, all the images in the buckets that fall within a small Hamming radius r of the query are returned. If a query returns no neighbors inside Hamming ball with radius r, it is treated as a failed query with zero precision [10].

4.2. Compared methods and parameter setting

- Locality Sensitive Hashing (LSH) [17], which is based on the random projection.
- Spectral Hashing (SH) [7], we use the publicly available code provided by the authors.[2]
- Anchor Graph Hashing (AGH) [9], we use the publicly available code provided by the authors[3] and the number of anchors is set to 500.
- Iterative Quantization (ITQ) [27], we use the publicly available code provided by the authors.[4]
- Sequential Projection Learning for Hashing (SPLH) [10], we use the code provided by the authors. 2000 images are randomly selected as labeled images.
- Composite Hashing with Multiple Sources (CHMS) [12], we use the publicly available code provided by the authors.[5]
- Kernel Hashing with Multiple Features (MFKH) [14], we use the publicly available code provided by the authors.[6] 5000 images are randomly selected as labeled images.
- Semi-Supervised Multi-Graph Hashing (MGH), which is the work introduced in this paper. 500 anchor points are generated to build the graph for both of two datasets. 2000 images are randomly selected as labeled images. The parameters λ and η are empirically set to 100 and 10 in the experiments.

In LSH, SH, AGH, ITQ and SPLH, we concatenate the features from different sources into a long vector. It is worth to note that LSH, SH, AGH, ITQ and CHMS are unsupervised approaches. Both SPLH and MGH are semi-supervised, and for fair comparison we select the same number of labeled images (e.g. 2000) in the experiment. MFKH is a supervised method, in order to achieve comparable results we select 5000 labeled images to provide supervise information. However, even so, we find it suffers from overfitting through the results.

4.3. Results and analysis

Table 1 shows the Hamming ranking performance of different hashing methods on the two datasets. The **Mean Average Precision (MAP)** with different code length is reported. We can see that our semi-supervised MGH approaches outperform the other hashing methods in different number of bits. Both SPLH and MGH learn the hash functions sequentially in a semi-supervised manner, but our approach get higher performance due to the multi-graph learning scheme. Furthermore, our methods shows increasingly higher benefits over two other multiview based approaches CHMS and MFKH. Both CHMS and MGH construct one graph for each source, however, CHMS treats each graph equally while MGH learns an optimized weight for each graph and supervised information is incorporated. MFKH is of supervised scheme, but achieves worse result than CHMS and MGH in most cases even more images are labeled. It is likely that MFKH is easy to suffer from overfitting in experiments [10]. Moreover, we also investigate a naive approach of linear scan according to Euclidean distance (l_2 scan) as a baseline in Table 1. The results show that Hashing based ANN searching does not necessarily be inferior to linear scan with l_2 distance, which has also be observed in AGH [9]. All of these have proven the effectiveness of our proposed semi-supervised multi-graph hashing method for binary code learning.

In order to give an intuitive understanding about effect of the learned weights in our method, we further provide the MAP scores of equal weight (i.e. unweighted) scheme of MGH (MGH_uw) in Table 1. These results show that the weighted MGH is superior to the unweighted one in all the setting, which demonstrates the proposed weight learning approach is effective in our hashing method.

[2] http://www.cs.huji.ac.il/yweiss/SpectralHashing/.
[3] http://www.ee.columbia.edu/wliu.
[4] http://www.unc.edu/yunchao/.
[5] http://www.cs.purdue.edu/homes/zhang168/.
[6] http://www.nlsde.buaa.edu.cn/xlliu/.

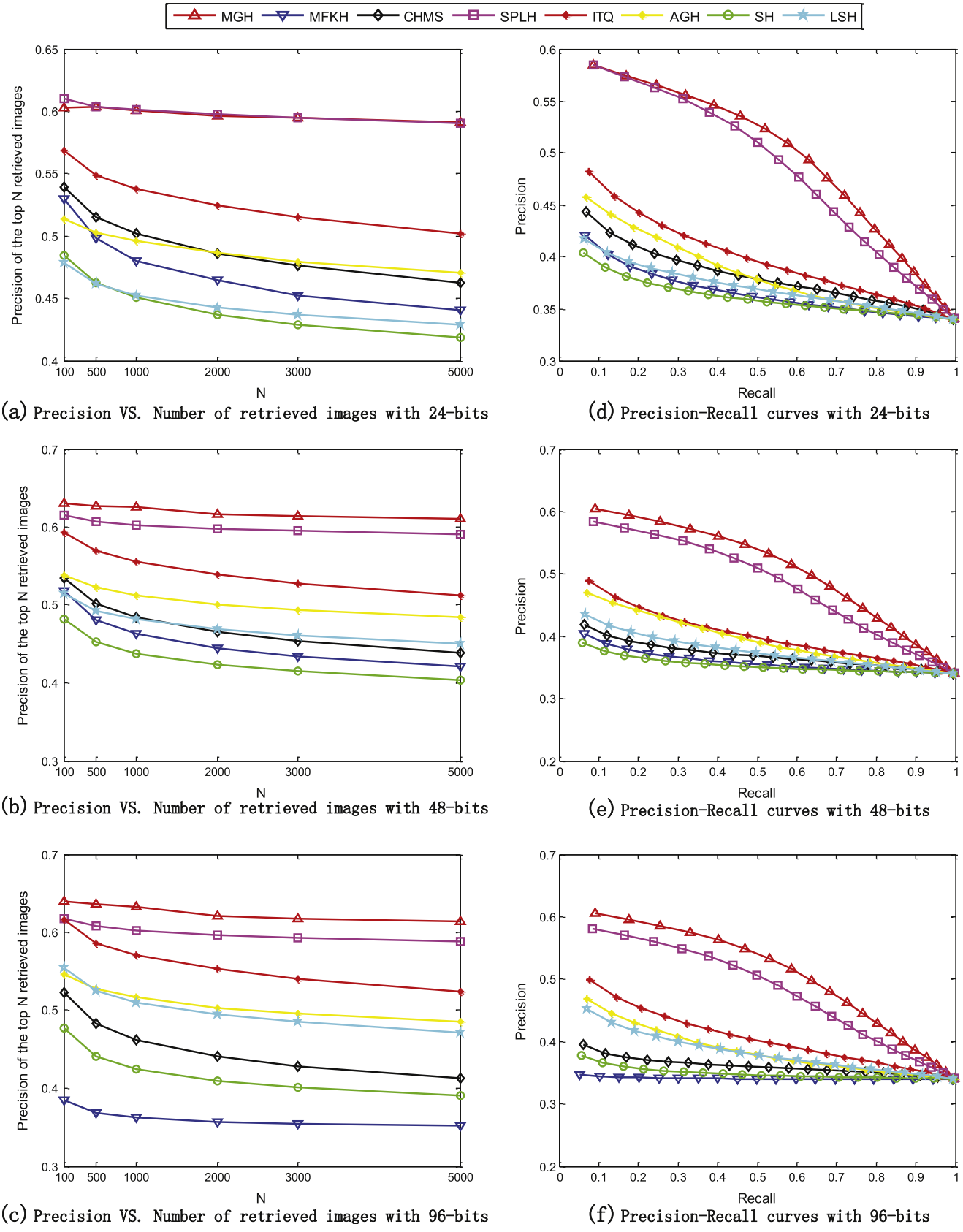

Fig. 4. Hamming ranking: comparison results of different hashing methods on the NUSWIDE dataset. (a)–(c): Precision vs. number of retrieved images with 24, 48, 96 bits. (d)–(f): Precision–Recall curves with 24, 48, 96 bits.

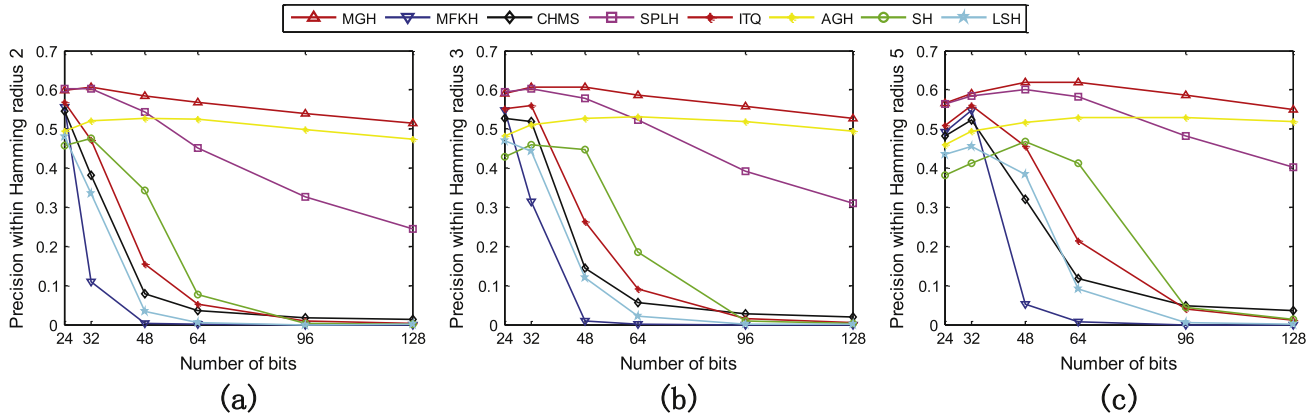

Fig. 5. Hashing lookup: comparison results of different hashing methods on the NUSWIDE dataset. (a)–(c) depict precision within Hamming radius 2, 3, 5 for hash lookup.

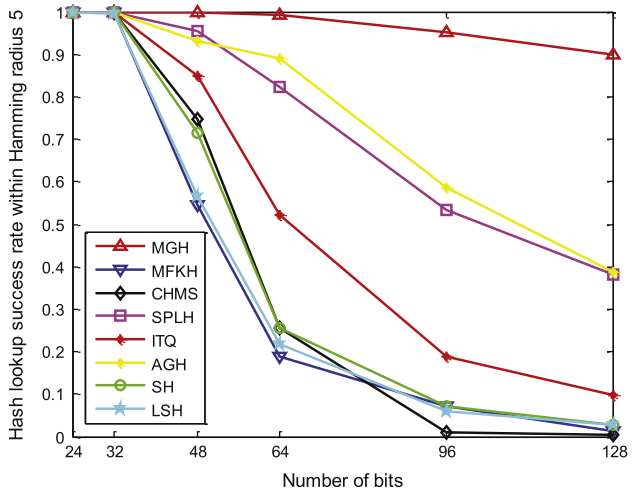

Fig. 6. Hashing lookup success rate within Hamming radius 5 on the MNIST dataset.

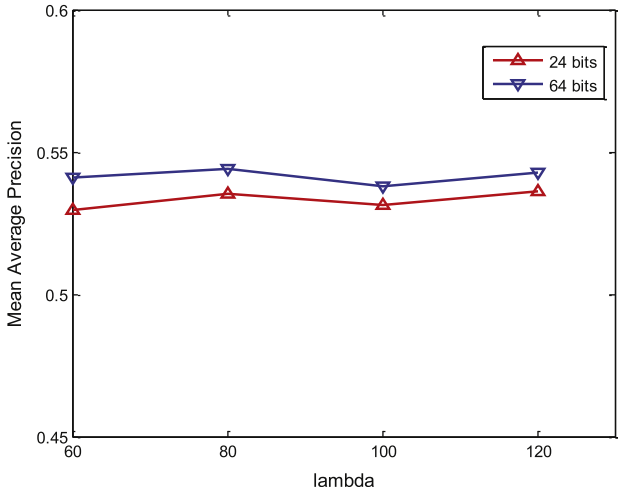

Fig. 7. Mean average precision on NUSWIDE dataset of our method with different λ.

The ranking performance in terms of different retrieved image numbers by different methods on the two datasets is demonstrated in Figs. 2(a)–(c) and 4(a)–(c). The experiment is carried out with 24, 48, 96-bits binary code respectively. From the results we can see, when the number of retrieved images varies, our MGH approaches can still get higher precision over all the queries.

We give the Precision–Recall curves of different algorithms on the two datasets in Figs. 2(d)–(f) and 4(d)–(f), respectively. The experiment is carried out with 24, 48, 96-bits binary code respectively and from the results we can see that our approaches have the highest PR score, which demonstrates the stronger overall capability than other methods for scalable similarity search.

The results of hash lookup by different hashing approaches on the two datasets are shown in Figs. 3 and 5, respectively. We carry out hash lookup within Hamming radius 2, 3, 5 respectively and report the search precision in terms of different bits. We can see that our MGH approaches perform better than the other methods except short code length on MNIST. In general, spectral analysis based hashing, such as SH and AGH, often choose the biggest eigenvectors as projection functions, which contain the most information in theory. Therefore, these spectral analysis based hashing algorithms may achieve good performance with short code length. However, it is not always true on complex dataset. In Fig. 5, our MGH is superior to these algorithms consistently on NUS-Wide dataset.

Note that we follow [10] to treat failing to find any hash bucket for a query as zero precision, different from [7] which ignored the failed queries when computing the mean precision over all queries. The performance of other hashing methods drops to some extent in high code length (e.g. >48), which is attributed to the increasing number of failed queries. To quantify the failing cases, we further conduct an experiment on the MNIST dataset and report the hash lookup success rate (the proportion of the queries resulting in successful hash lookup in all queries) in Fig. 6. We can see that the MGH approaches have higher success rate than other methods, which verify our stable performance with different code bits. We also study the sensitiveness of parameter λ. Due to the space limit, we only report the results in terms of MAP over NUSWIDE dataset. The experiment results are shown in Fig. 7. We can see that our method is not sensitive to λ.

5. Conclusion and future work

In this paper, we propose a semi-supervised Multi-Graph Hashing (MGH) approach for scalable similarity search. Different from the traditional hashing methods that usually adopt a single modality or combine the multiple modalities simply without exploiting the impact of different modalities, we propose a multi-graph learning scheme for binary code learning. In this way, the effects of different modalities can be adaptively modulated. Besides, semi-supervised information is also incorporated into the unified learning scheme to obtain more accurate hash functions. Extensive

experiments are conducted on two large-scale datasets to prove the effectiveness of our approach.

In our proposed approach, all the graphs are constructed from different modalities with Gaussian kernel. However, different kernels may be more appropriate for different modalities in graph construction. We can attempt multi-kernel and multimodal hashing in the future work. Besides, our proposed hashing approach can also be applied to other large-scale problems such as duplicate image/video detection.

Acknowledgments

This work was supported in part by 973 Program (Grant No. 2010CB327905), National Natural Science Foundation of China (Grant No. 61170127, 61332016).

References

[1] Y. Yang, F. Nie, D. Xu, J. Luo, Y. Zhuang, Y. Pan, A multimedia retrieval framework based on semi-supervised ranking and relevance feedback, IEEE Trans. Pattern Anal. Mach. Intell. 34 (2012) 723–742.
[2] S. Arya, D. Mount, N. Netanyahu, R. Silverman, A. Wu, An optimal algorithm for approximate nearest neighbor searching, J. ACM 45 (1998) 891–923.
[3] J. Friedman, J. Bentley, R. Finkel, An algorithm for finding best matches in logarithmic expected time, TOMS 3 (1977) 209–226.
[4] C. Silpa-Anan, R. Hartley, Optimised KD-trees for fast image descriptor matching, in: CVPR, 2008.
[5] P. Indyk, R. Motwani, Approximate nearest neighbors: towards removing the curse of dimensionality, in: STOC, 1998.
[6] A. Andoni, P. Indyk, Near-optimal hashing algorithms for approximate nearest neighbor in high dimensions, in: FSCS, 2006.
[7] Y. Weiss, A. Torralba, R. Fergus, Spectral hashing, in: NIPS, 2008.
[8] P. Li, M. Wang, J. Cheng, C. Xu, H. Lu, Spectral hashing with semantically consistent graph for image indexing, TMM (2012).
[9] W. Liu, J. Wang, S. Kumar, S. Chang, Hashing with graphs, in: ICML, 2011.
[10] J. Wang, S. Kumar, S. Chang, Semi-supervised hashing for large-scale search, TPAMI 34 (2012) 2393–2406.
[11] Y. Yang, J. Song, Z. Huang, Z. Ma, N. Sebe, A. Hauptmann, Multi-feature fusion via hierarchical regression for multimedia analysis, IEEE Trans. MultiMedia (2012).
[12] D. Zhang, F. Wang, L. Si, Composite hashing with multiple information sources, in: SIGIR, 2011.
[13] J. Song, Y. Yang, Z. Huang, H. Shen, R. Hong, Multiple feature hashing for real-time large scale near-duplicate video retrieval, in: ACM MM, 2011.
[14] X. Liu, J. He, D. Liu, B. Lang, Compact kernel hashing with multiple features, in: ACM MM, 2012.
[15] H. Xia, P. Wu, S.C. Hoi, R. Jin, Boosting multi-kernel locality-sensitive hashing for scalable image retrieval, in: SIGIR, 2012.
[16] R. Weber, H.-J. Schek, S. Blott, A quantitative analysis and performance study for similarity-search methods in high-dimensional spaces, in: VLDB, 1998.
[17] M. Charikar, Similarity estimation techniques from rounding algorithm, in: ACM Symposium on Theory of Computing, 2002, pp. 380–388.
[18] M. Datar, N. Immorlica, P. Indyk, V. Mirrokni, Locality-sensitive hashing scheme based on p-stable distributions, in: SCG, 2004, pp. 253–262.
[19] P. Jain, B. Kulis, K. Grauman, Fast image search for learned metrics, in: CVPR, 2008.
[20] B. Kulis, P. Jain, K. Grauman, Fast similarity search for learned metrics, TMAPI (2009) 2143–2157.
[21] B. Kulis, K. Grauman, Kernelized locality-sensitive hashing for scalable image search, in: ICCV, 2009.
[22] G. Hinton, R. Salakhutdinov, Reducing the dimentionality of data with neural networks, Science (2006) 504–507.
[23] B. Kulis, T. Darrell, Learning to hash with binary reconstructive embeddings, in: NIPS, 2009.
[24] A. Torralba, R. Fergus, Y. Weiss, Small codes and large image databases for recognition, in: CVPR, 2008.
[25] M. Norouzi, D.J. Fleet, Minimal loss hashing for compact binary codes, in: ICML, 2011.
[26] H. Xu, J. Wang, Z. Li, G. Zeng, S. Li, N. Yu, Complementary hashing for approximate nearest neighbor search, in: ICCV, 2011.
[27] Y. Gong, S. Lazebnik, Iterative quantization: a procrustean approach to learning binary codes, in: CVPR, 2011.
[28] J. Heo, Y. Lee, J. He, S. Chang, S. Yoon, Spherical hashing, in: CVPR, 2012.
[29] J. He, W. Liu, S. Chang, Scalable similarity search with optimized kernel hashing, in: KDD, 2010.
[30] Y. Mu, J. Shen, S. Yan, Weakly-supervised hashing in kernel space, in: CVPR, 2010.
[31] D. Zhang, J. Wang, D. Cai, J. Lu, Self-taught hashing for fast similarity search, in: SIGIR, 2010.
[32] Y. Zhuang, Y. Liu, F. Wu, Y. Zhang, J. Shao, Hypergraph spectral hashing for similarity search of social image, in: ACM MM, 2011.
[33] W. Liu, J. Wang, R. Ji, Y. Jiang, S. Chang, Supervised hashing with kernels, in: CVPR, 2012.
[34] R. Salakhutdinov, G. Hinton, Semantic hashing, Int. J. Approx. Reasoning (2009) 504–507.
[35] K. He, F. Wen, J. Sun, K-means hashing: an affinity-preserving quantization method for learning binary compact codes, in: CVPR, 2013.
[36] W. Kong, W. Li, Double-bit quantization for hashing, in: AAAI, 2012.
[37] W. Kong, W. Li, M. Guo, Manhattan hashing for large-scale image retrieval, in: SIGIR, 2012.
[38] S. Moran, V. Lavrenko, M. Osborne, Neighbourhood preserving quantisation for lsh, in: SIGIR, 2013.
[39] D. Lowe, Object recognition from local scale-invariant features, in: ICCV, 1999.
[40] M. Bronstein, A. Bronstein, F. Michel, N. Paragios, Data fusion through cross-modality metric learning using similarity-sensitive hashing, in: CVPR, 2010.
[41] S. Kumar, R. Udupa, Learning hash functions for cross-view similarity search, in: IJCAI, 2011.
[42] Y. Zhen, D. Yeung, A probabilistic model for multimodal hash function learning, in: KDD, 2012.
[43] W. Liu, J. He, S. Chang, Large graph construction for scalable semi-supervised learning, in: ICML, 2010.
[44] T. Chua, J. Tang, R. Hong, H. Li, Z. Luo, Y. Zheng, NUS-WIDE: a real world web image database from National University of Singapore, in: CIVR, 2009.