Neurocomputing 120 (2013) 83–89

Contents lists available at ScienceDirect

Neurocomputing

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

Hashing with dual complementary projection learning for fast image retrieval

Peng Li, Jian Cheng *, Hanqing Lu

National Laboratory of Pattern Recognition, Institute of Automation, Chinese Academy of Sciences, No. 95, Zhongguancun East Road, Beijing 100190, China

ARTICLE INFO

Article history:
Received 25 December 2011
Received in revised form
11 June 2012
Accepted 1 July 2012
Available online 26 March 2013

Keywords:
Hashing
Complementary projection learning
Binary codes
Fast image retrieval

ABSTRACT

Due to explosive growth of visual content on the web, there is an emerging need of fast similarity search to efficiently exploit such enormous web contents from very large databases. Recently, hashing has become very popular for efficient nearest neighbor search in large scale applications. However, many traditional hashing methods learn the binary codes in a single shot or only employ a single hash table, thus they usually cannot achieve both high precision and recall simultaneously. In this paper, we propose a novel dual complementary hashing (DCH) approach to learn the codes with multiple hash tables. In our method, not only the projection for each bit inside a hash table has the property of error-correcting but also the different hash tables complement each other. Therefore, the binary codes learned by our approach are more powerful for fast similarity search. Extensive experiments on publicly available datasets demonstrate the effectiveness of our approach.

1. Introduction

Similarity search is a fundamental problem that aries commonly in many applications such as date mining [30], image classification [33], and video annotation [31,32]. The traditional content-based image retrieval (CBIR) techniques usually adopt exhaustive comparing the query image with pooled database based on all kinds of visual features [28,29], which is infeasible for large-scale applications because the linear complexity is not scalable in practical situations. Besides, most applications suffer from the curse of dimensionality since visual 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.

To avoid excessive computational and memory costs, an approximate nearest neighbor (ANN) search is more appealing than exhaustive comparing with sublinear query complexity [12] and many tree-based approaches [1–5] have been developed for ANN in large scale datasets in the past decades. However, they will encounter difficulties in practical applications when the feature dimension is high [14]. For high-dimensional cases and applications, hashing-based ANN techniques have attracted more attention. Hashing-based methods are promising in accelerating similarity search by learning compact binary codes for image description in the database. It is extremely fast to perform similarity search by computing Hamming distance based on the learned binary codes instead of the visual features [6].

Many hashing algorithms (e.g., LSH [7], SH [8]) have been proposed to address the fast retrieval issue in recent years. These hashing-based methods for fast image retrieval can be considered as embedding high-dimensional feature vectors to a low-dimensional Hamming space, while retaining as much as possible the semantic similarity structure of data. Although the traditional hashing methods have shown success in large-scale image search, there are some shortcomings of them. As they learn the hash functions or hash tables independently, the codes and tables learned by these methods do not have the complementary capability. Therefore, these methods usually cannot get the desired performance in many cases. More recently, sequential projection learning for hashing (SPLH) [10] and complementary hashing (CH) [11] are proposed to learn complementary hash functions and hash tables, respectively. However, the binary codes learned by these two methods also lack discriminative power to some extent because both of them only address one aspect of the problem.

In order to address the above-mentioned problems, we propose a novel dual complementary hashing (DCH) approach for fast similarity search in this paper. In our method, not only the multiple hash tables are learned one by one but also the hash functions inside a hash table are learned sequentially. Therefore, our hashing method has the following characteristics: (1) the different hash tables are complementary and (2) the hash functions inside each hash table are also complementary. Compared with the methods adopting a single hash table or multiple independent hash tables, multiple complementary hash tables can balance the precision and recall more effectively. Compared with the methods which learns the hash functions in a single shot, our approach can perform much more consistently in the case of long codes. Thus, the binary codes learned by our method are

* Corresponding author. Tel.: +86 1062632267.
E-mail address: jcheng@nlpr.ia.ac.cn (J. Cheng).

0925-2312/$ - see front matter © 2013 Elsevier B.V. All rights reserved.
http://dx.doi.org/10.1016/j.neucom.2012.07.053

Table 1
The comparison of four representative hashing methods with our approach.

Method	Data dependent	Complementary hash functions	Multiple hash tables	Complementary hash tables
LSH [7]	No	No	Yes	No
SH [8]	Yes	No	No	–
SPLH [10]	Yes	Yes	No	–
CH [11]	Yes	No	Yes	Yes
DCH	Yes	Yes	Yes	Yes

more effective for fast image retrieval. Table 1 gives a comparison between our approach and some representative hashing methods.

The rest of the paper is organized as follows. Section 2 reviews related work. Section 3 describes our dual complementary hashing approach in detail. In Section 4, we introduce the experiments on two publicly available datasets. Finally, conclusions are given in Section 5.

2. Related work

The goal of content-based image retrieval is to find a subset of images that are most similar to the query. Exhaustively comparing the query with each sample in a large database based on traditional visual descriptors is infeasible because of the linear time complexity and storage problem. To avoid excessive computational and memory costs, an approximate nearest neighbor (ANN) search is more appealing than exhaustive comparing with sublinear query complexity [12]. For a low-dimensional feature space, fast similarity search can be carried out efficiently by some tree-based methods (e.g., KD-tree, R-tree [13]). These methods usually partition the data space recursively to implement an exact similarity search in the low-dimension feature space while they are not suitable for the features with thousands of dimensions, which is often met in practical situations. Hashing-based methods, which learns compact binary bodes for image description and fast retrieval, are purposefully designed to approximately answer queries in virtually constant time [6]. In addition, the storage is also substantially reduced as they usually store only compact binary codes for each data point.

Many hashing algorithms have been developed in recent years. One of the most popular methods is locality sensitive hashing (LSH) [7]. Given a similarity metric S in a feature space, the LSH algorithm typically guarantees the probability for any two samples x_i and x_j falling into the same bucket to be $S(x_i, x_j)$, known as the "locality sensitive" property. One popular method in LSH is to generate a random vector h from a particular probabilistic distribution, e.g., p-stable distribution [15] for ℓ^p–metric space. However, since the random vector is data-independent, LSH may lead to quite inefficient codes in practice [8,16] as it requires multiple tables with long codes [17]. In order to overcome this problem, several recently proposed hashing techniques [9,16,18–20] attempt to apply machine learning approaches rather than random projections to find good data-aware hash functions. In [8], a new technique called spectral hashing (SH) is proposed based on spectral graph partitioning [21]. SH calculates the bits by thresholding a subset of eigenvectors of the Laplacian of the similarity graph and it has demonstrated significant improvements over many other methods in terms of the number of bits required to find good similar neighbors. Many extensions [22–25] of SH have also been proposed in recent years. However, the above methods learn the hash bits independently in a single shot, the coding errors made by one bit have no influence on the learning of other bits. In order to overcome this shortcoming, Wang et al. introduce a sequential projection learning for hashing (SPLH) method [10] to learn the hash functions so that the errors generated by the previous hash function can be corrected by the following one. This method can perform very consistently as the number of code bit grows. One problem of SPLH is that it only employs a single hash table, so it usually has to increase the radius of the Hamming ball to retrieve more points if higher recall is desired, which may drop the precision rapidly. In [11], instead of constructing a single hash table and learning hash bits one by one, Xu et al. propose a complementary hashing (CH) method by learning multiple complementary hash tables sequentially to balance the precision and recall performance. Although CH learns multiple complementary hash tables, it learns the hash bits inside a hash table independently in a single shot. Therefore, the coding errors made by one bit have no influence on the learning of other bits, which may decrease the performance as the number of bit grows. Different from SPLH and CH which either learn the hash functions sequentially or employ multiple complementary hash tables, our proposed dual complementary hashing (DCH) approach is able to learn the complementary hash functions and hash tables simultaneously for fast image retrieval.

3. Dual complementary hashing

In this section, we will introduce our dual complementary hashing (DCH) method in detail. In our approach, both the hash functions and hash tables are learned progressively so that the following ones can correct the errors generated by the previous ones. In other words, not only the hash functions inside a hash table are complementary but also the multiple hash tables are complementary. Therefore, the binary codes learned by our approach are more effective than the traditional hashing methods.

Assume the database $\mathbf{X}$ consists of N data points $\{\mathbf{x}_i\}_{i=1}^{N}$, $\mathbf{x}_i \in R^d$. A hashing method with a single hash table is to map a data point $\mathbf{x}_i$ to a K-bits hash code $H(\mathbf{x}_i) = [h_1(\mathbf{x}_i), \ldots, h_K(\mathbf{x}_i)]^T$, where each hash function maps the data point to a single bit $h_k(\mathbf{x}_i) \in \{-1, 1\}$. Let $\mathbf{X}$ to be normalized to have zero mean and given a vector $\mathbf{w}_k \in R^d$, the k-th hash function can be defined as $h_k(\mathbf{x}_i) = \mathbf{sgn}(\mathbf{w}_k^T \mathbf{x}_i)$. A hashing method with multiple hash tables is to learn L hash tables $\{H_l(\cdot)\}_{l=1}^{L}$, where each of hash table $H_l(\cdot) = [h_{l1}(\cdot), \ldots, h_{lK}(\cdot)]^T$ can map the data points to K-bits hash codes and be applied for image retrieval independently. The final results are then obtained by fusing the results of all the hash tables. Our approach employs multiple hash tables, however, different from the existing methods, our hashing method has the following characteristics: (1) the different hash tables $\{H_l(\cdot)\}_{l=1}^{L}$ are complementary and (2) the K hash functions inside each hash table $\{h_{lk}(\cdot)\}_{k=1}^{K}$ are also complementary. Therefore, we name it dual complementary hashing (DCH).

3.1. Learning complementary hash functions

In this section, we first introduce how to learn the complementary hash functions inside a certain hash table which is similar to that in [10]. Given a labeled image subset $\mathbf{X}_{label}$, we construct a semantic matrix $\mathbf{S}$, where $S_{ij} = 1$ if image $\mathbf{x}_i$ and $\mathbf{x}_j$ have the same label and $S_{ij} = -1$ otherwise. Then, without loss of generality, we learn the hash functions in the l-th hash table by maximizing the following objective function:

$$J(H_l) = \sum_{i,j=1}^{n} S_{ij} \sum_{k=1}^{K} h_{lk}(\mathbf{x}_i) h_{lk}(\mathbf{x}_j) = \sum_{i,j=1}^{n} S_{ij} H_l(\mathbf{x}_i)^T H_l(\mathbf{x}_j) \quad (1)$$

where n is the number of labeled images and $H_l(\mathbf{x}_i)^T H_l(\mathbf{x}_j) = \sum_{k=1}^{K} h_{lk}(\mathbf{x}_i) h_{lk}(\mathbf{x}_j)$ reflects the similarity of the two hash codes.

Straightforwardly, the objective function encourages a pair of images to be projected to the similar hash codes if they have the same label, and be projected to the dissimilar hash codes if they have different labels.

In order to make the learned hash codes balanced, we propose to maximize the variance of the projected data on the whole training set $\mathbf{X}$ [24]:

$$\max_{\mathbf{W}_l} \mathtt{tr}[\mathbf{W}_l^T \mathbf{X}\mathbf{X}^T \mathbf{W}_l] \tag{2}$$

where $\mathbf{W}_l = [\mathbf{w}_{l1}, \ldots, \mathbf{w}_{lK}]$ is a $d \times K$ matrix. By relaxing $\mathrm{sgn}(\mathbf{w}^T\mathbf{x})$ to the signed magnitude $\mathbf{w}^T\mathbf{x}$ in Eq. (1), and combining the regularization term Eq. (2), we can get the final objective function

$$\begin{aligned}\hat{J}(W_l) &= \mathtt{tr}[\mathbf{W}_l^T \mathbf{X}_{label}\mathbf{S}\mathbf{X}_{label}^T \mathbf{W}_l + \lambda \mathbf{W}_l^T \mathbf{X}\mathbf{X}^T \mathbf{W}_l] \\ &= \mathtt{tr}[\mathbf{W}_l^T \mathbf{M}\mathbf{W}_l]\end{aligned} \tag{3}$$

where $\mathbf{M} = \mathbf{X}_{label}\mathbf{S}\mathbf{X}_{label}^T + \lambda\mathbf{X}\mathbf{X}^T$. Parameter λ trades off the effects of the supervised data and the regularizer.

The optimal solution $\hat{\mathbf{W}}_l$ can be easily obtained by adding the orthogonal constraint $\mathbf{W}_l^T\mathbf{W}_l = \mathbf{I}$ which guarantees the bits to be uncorrelated, and the projections $\mathbf{W}_l$ corresponds to the top K eigenvectors of $\mathbf{M}$. However, in order to obtain complementary hash bits, we learn the hash projections progressively.

Let $\mathbf{G}^{lk} = \mathbf{X}_{label}^T \mathbf{w}_{lk}\mathbf{w}_{lk}^T \mathbf{X}_{label}$, which measures the sign magnitude of pairwise relationships of the k-th projection of $\mathbf{X}_{label}$ in the l-th hash table. Then, we can get the updating rules for $\mathbf{S}$ as follows:

$$S_{ij}^{l,k+1} = \begin{cases} S_{ij}^{lk} - \alpha G_{ij}^{lk} & \mathtt{if\ sgn}(S_{ij} \cdot G_{ij}^{lk}) < 0 \\ S_{ij}^{k} & \mathtt{otherwise} \end{cases} \tag{4}$$

where α is a step size parameter and $\mathbf{S}^{l1}$ is initialized to be $\mathbf{S}^l$ in each table. How to obtain and update $\mathbf{S}^l$ will be introduced in next section. $\mathtt{sgn}(S_{ij} \cdot G_{ij}^{lk}) < 0$ indicates that hash bits $h_{lk}(\mathbf{x}_i)$ and $h_{lk}(\mathbf{x}_j)$ contradict with the given pairwise label. In other words, images with the same label are assigned different bits or those with different labels are assigned same bits. For both of the cases, we increase the weights of the relevant image pairs in the weight matrix $\mathbf{S}$. With the updated weight matrix, we learn the hash projections one by one such that the errors generated by the previous projection can be decreased progressively by the following one.

3.2. Learning complementary hash tables

Although image retrieval can be conducted with a single hash table, we usually have to increase the radius of the Hamming ball and retrieve more images to get higher recall, which may drag down the precision rapidly. In this section, we will introduce how to learn multiple complementary hash tables, which can balance the precision and recall in a more effective way.

Suppose we have learned the l-th hash table $H_l(\cdot)$ with the method introduced in Section 3.1, we can compute the Hamming distance between image pairs $\mathbf{x}_i$ and $\mathbf{x}_j$ as follows:

$$d_l(\mathbf{x}_i, \mathbf{x}_j) = \tfrac{1}{4}\|H_l(\mathbf{x}_i) - H_l(\mathbf{x}_j)\|^2 \tag{5}$$

A good hash table should map the images from the same class near in the Hamming space and map the images from different classes far away. By comparing the Hamming distance $d_l(\mathbf{x}_i, \mathbf{x}_j)$ and their pairwise label on the labeled image set, we can get the following cases (β is a thresholding parameter for the Hamming distance):

1. $S_{ij} = 1$, $d_l(\mathbf{x}_i, \mathbf{x}_j) < \beta$: $\mathbf{x}_i$ has the same label with $\mathbf{x}_j$ and they are mapped to be near in the Hamming space.
2. $S_{ij} = 1$, $d_l(\mathbf{x}_i, \mathbf{x}_j) > \beta$: $\mathbf{x}_i$ has the same label with $\mathbf{x}_j$ and they are mapped to be far away in the Hamming space.
3. $S_{ij} = -1$, $d_l(\mathbf{x}_i, \mathbf{x}_j) < \beta$: $\mathbf{x}_i$ has different label from $\mathbf{x}_j$ and they are mapped to be near in the Hamming space.
4. $S_{ij} = -1$, $d_l(\mathbf{x}_i, \mathbf{x}_j) > \beta$: $\mathbf{x}_i$ has different label from $\mathbf{x}_j$ and they are mapped to be far away in the Hamming space.

Algorithm 1. Dual complementary hashing.

Input: data $\mathbf{X}$, labeled data $\mathbf{X}_{label}$, semantic matrix $\mathbf{S}$, length of hash codes K, number of hash tables L, parameters λ, α, β.
Output: hash projections $\{H_l(\cdot)\}_{l=1}^{L}$
Initialize the weight matrix $\mathbf{S}^1 = \mathbf{S}$
for $l = 1$ **to** L **do**
  $\mathbf{S}^{l1} = \mathbf{S}^l$, $\mathbf{X}_{all} = \mathbf{X}$
  for $k = 1$ **to** K **do**
    Compute $\mathbf{M}$ as follows:
      $\mathbf{M} = \mathbf{X}_{label}\mathbf{S}^{lk}\mathbf{X}_{label}^T + \lambda\mathbf{X}_{all}\mathbf{X}_{all}^T$
    Extract the first eigenvector of $\mathbf{M}$ and set it to $\mathbf{w}_{lk}$
    Update the weight matrix $\mathbf{S}^{l,k+1}$ by Eq. (4)
    Compute the residual:
      $\mathbf{X}_{all} = \mathbf{X}_{all} - \mathbf{w}_{lk}\mathbf{w}_{lk}^T\mathbf{X}_{all}$
  end for
  Obtain the hash projection $H_l(\cdot)$ (i.e., $\{\mathbf{w}_{lk}\}_{k=1}^{K}$)
  Compute the pairwise Hamming distances $d_l(\mathbf{x}_i, \mathbf{x}_j)$ on $\mathbf{X}_{label}$:
    $d_l(\mathbf{x}_i, \mathbf{x}_j) = \tfrac{1}{4}\|H_l(\mathbf{x}_i) - H_l(\mathbf{x}_j)\|^2$
  Update the weight matrix $\mathbf{S}^{l+1}$ by Eq. (6)
end for

We can see that the image pairs are wrongly mapped by the l-th hash table $H_l(\cdot)$ in cases 2 and 3. Then, we update the elements in $\mathbf{S}$ as follows:

$$S_{ij}^{l+1} = \begin{cases} S_{ij} & \mathtt{if}\ S_{ij} = 1,\ d_l(\mathbf{x}_i, \mathbf{x}_j) > \beta \quad \mathtt{or} \quad S_{ij} = -1, d_l(\mathbf{x}_i, \mathbf{x}_j) < \beta \\ 0 & \mathtt{otherwise} \end{cases} \tag{6}$$

With the updated weight matrix $\mathbf{S}^{l+1}$, we can learn the $(l+1)$−th hash table with the method introduced in Section 3.1. By removing the correctly mapped image pairs while remaining the wrongly mapped ones in the weight matrix, we hope that the following hash table put emphasis on the wrongly projected image pairs by the previous one. After learning the hash tables one by one, we can obtain multiple complementary hash tables finally.

The detail of our DCH approach is shown in Algorithm 1.

4. Experiments

In this section, we evaluate the proposed DCH approach on the USPS, MNIST and CIFAR-10 datasets. We compare its performance with five state-of-the-art binary coding methods:

1. Locality sensitive hashing (LSH) [7].
2. Spectral hashing (SH) [8].
3. Semi-supervised hashing (SSH) [24].
4. Sequential projection learning for hashing (SPLH) [10].
5. Complementary hashing (CH) [11].

4.1. Datasets and parameter settings

The USPS[1] and MNIST[2] datasets are handwritten digit images from 10 classes (0–9 digit characters). The USPS dataset contains 11 000 images and each image is 16×16 in resolution, which results in a 256-D feature vector. In our experiments, we randomly

[1] http://www.cs.nyu.edu/~roweis/data.html
[2] http://yann.lecun.com/exdb/mnist/

divide the USPS dataset into two parts: a training set that contains 5000 images and a testing set that contains 6000 images. The MNIST dataset is a much larger one which consists of 70 000 images in total. Each image has 28×28 pixels and can be represented by a 784-D vector. We randomly select 69 000 images to form the training set and the left 1000 images are used for testing.

The CIFAR-10 dataset [26] is a subset of the Tiny Images dataset and it consists of 60 000 color images in 10 classes, with 6000 images per class. There are 50 000 training images and 10 000 test images. The original Tiny images are 32×32 pixels. We represent them with GIST [27] descriptors computed at eight orientations and four different scales, resulting in a 512-D vector for each image.

For all the datasets, the training set is used to learn the hash functions of different hashing methods. The testing set is used to evaluate the performance of different approaches through the nearest neighbor retrieval based on the binary codes. About 1000 images are randomly chosen and used as labeled images in all the methods that need label information. The parameters λ, α, β are set to 0.1, 1 and 5, respectively. Hamming ranking is performed to evaluate the performance of all the hashing methods. To perform Hamming ranking for the hashing methods with multiple hash tables, we compute the Hamming distance of image $\mathbf{x}_i$ and the query $\mathbf{x}_q$ by

$$d(\mathbf{x}_i, \mathbf{x}_q) = \frac{1}{L} \sum_{l=1}^{L} \|H_l(\mathbf{x}_q) - H_l(\mathbf{x}_i)\|^2$$

where L is the number of tables and is set to 5 in the experiments.

4.2. The comparison results

Fig. 1 illustrates the performance of different algorithms for the nearest neighbor search, in terms of precision versus different numbers of bits. We can see that our DCH method achieves the best performance with different numbers of bits on all the datasets. Among the methods with a single hash table, SPLH outperforms SH and SSH by a large margin because SPLH learns complementary hash bits one by one while SH and SSH learn all the bits in a single shot. Especially when the number of bits grows, the superiority of the complementary bits becomes more obvious. For the methods with multiple hash tables, CH performs better than LSH with small hash bits (e.g., $K \leq 16$) sometimes because LSH constructs the hash tables in a random way while CH learns the hash tables in a complementary way which can reduce the redundancy across the hash tables dramatically. However, the performance of CH drops as the number of bits grows. As explained in [11], this may be attributed to the reason that CH learns all the codes at the same time by PCA composition on the adjusted data covariance matrix and the orthogonality constraints force one to pick those PCA directions that have very low variance in the case of long codes. That is also why the performance of SSH drops when the bits grows, for SSH and CH learn the hash bits inside the hash table with the same mechanism. Comparing SPLH and CH, we can see that SPLH gets better performance than CH, which shows that the complementary hash bits contributes more than the complementary hash tables in our experiments. However, our DCH approach, which learns both complementary hash bits and multiple complementary hash tables, can perform consistently better than the other methods.

The performance of all the methods, in terms of precision versus the number of retrieved images on the datasets, is illustrated in Figs. 2–5 shows the precision–recall curve on the different datasets. We can see that the DCH approach has the highest score with different evaluation methods, which means that our approach can return more relevant images in the top ranked images than the other methods for the fast image retrieval task.

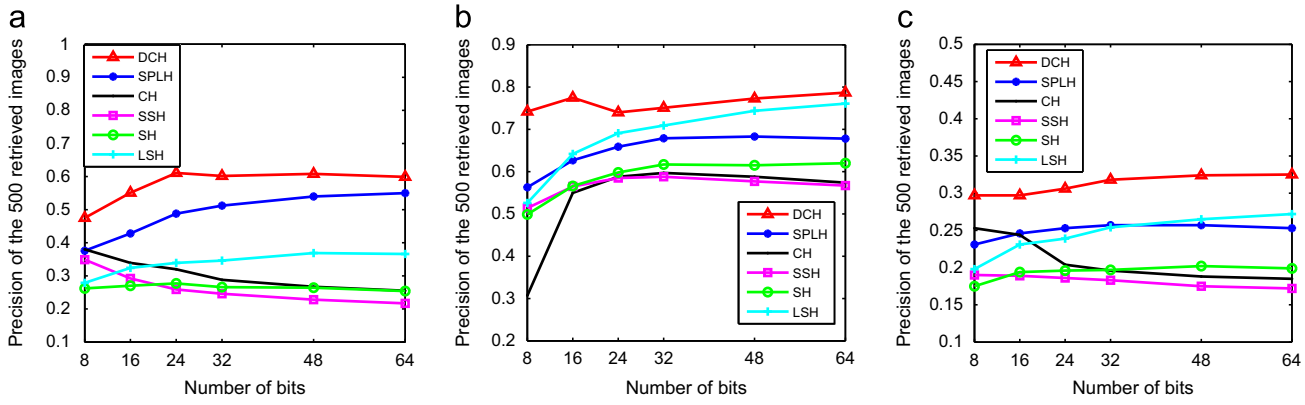

Fig. 1. Precision of image search on (a) USPS, (b) MNIST, and (c) CIFAR datasets with different numbers of bits.

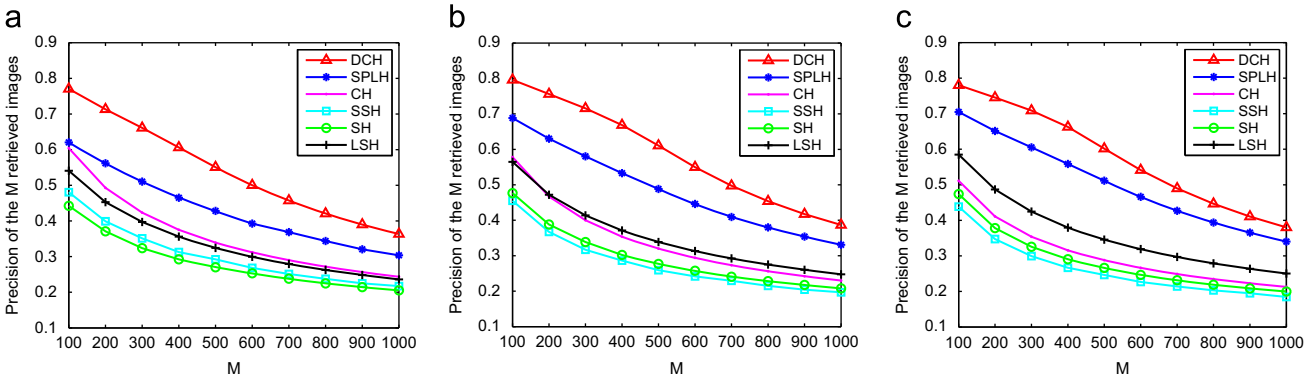

Fig. 2. Precision of the first M neighbors searched by different algorithms on the USPS dataset: (a) 16-bits, (b) 24-bits, and (c) 32-bits.

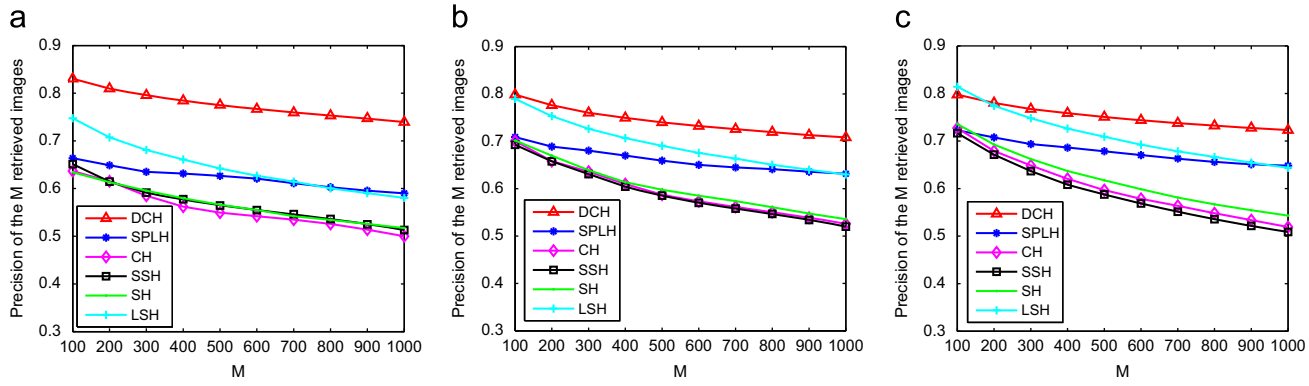

Fig. 3. Precision of the first M neighbors searched by different algorithms on the MNIST dataset: (a) 16-bits, (b) 24-bits, and (c) 32-bits.

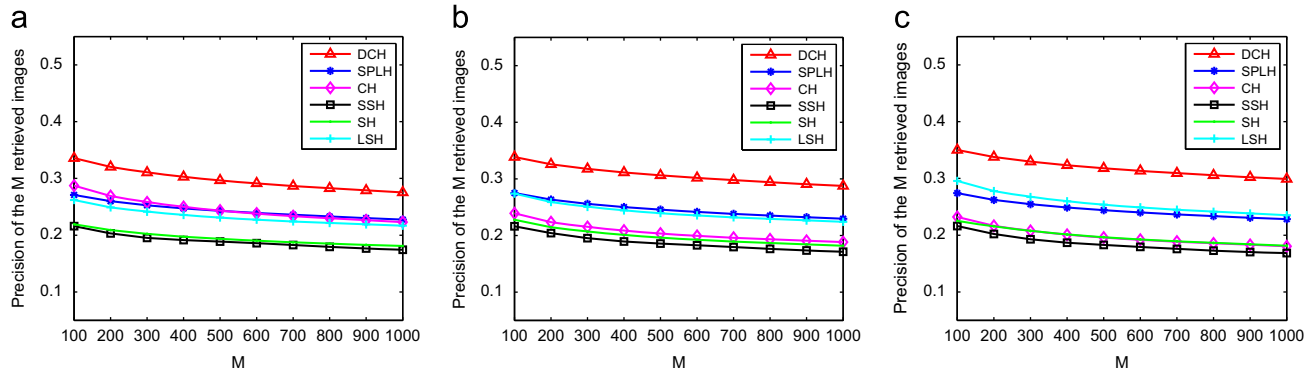

Fig. 4. Precision of the first M neighbors searched by different algorithms on the CIFAR dataset: (a) 16-bits, (b) 24-bits, and (c) 32-bits.

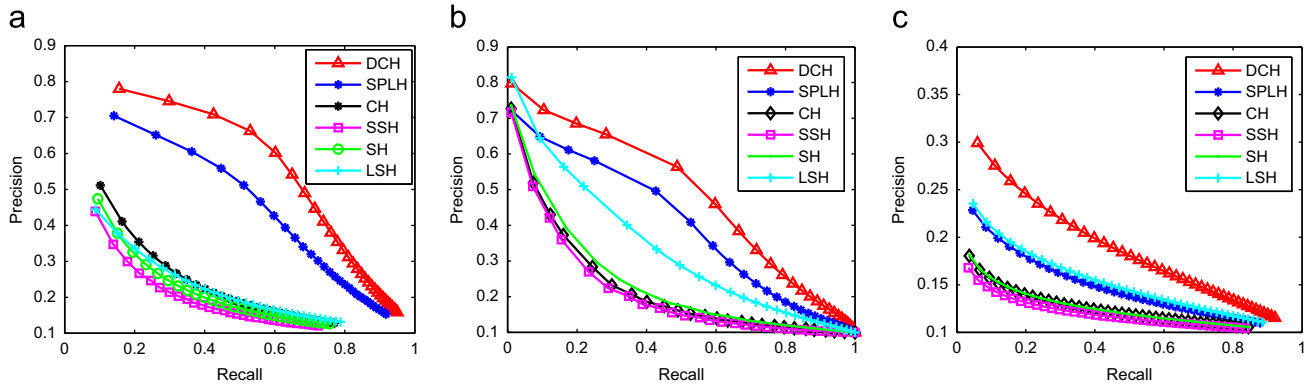

Fig. 5. The precision–recall curve of different algorithms with 32-bits on the datasets: (a) USPS, (b) MNIST, and (c) CIFAR.

We also conduct experiments by changing the number of hash tables in our experiments. Fig. 6 shows the performance of our DCH approach with different table numbers L on the three datasets. It demonstrates that the performance becomes better as the number of hash table grows and the performance trends to be converged when $L > 5$. Since more hash tables means larger memory storage and longer training time, we adopt five hash tables in our experiment to trade off the accuracy and computational cost.

The comparison of training time on the MNIST dataset is shown in Table 2. The code length of all the hashing approaches is 32-bits and five hash tables are employed for CH, LSH and DCH. We can see that SH and SSH use less time than the other methods because they only employ one hash table and learn the hash bits in a single shot by a PCA composition. SPLH learns the hash bits sequentially while CH and LSH employs multiple hash tables. They all need more training time than SH and SSH. Since our DCH learns both complementary hash bits and hash tables, its computation cost is the highest among the compared approaches. However, its computation speed is still very fast (less than 6 min on 69 000 training images) and this procedure is carried out offline. After the hash functions are learned, the compressing time for the query images in the testing set is nearly the same for all the approaches and can be finished in a constant time.

Finally, we conduct an experiment on the MNIST dataset to investigate the impact of the parameter λ, which trades off the effects of the supervised data and the regularized term in our approach. The experimental results are shown in Fig. 7, from which we can see that our approach perform quite consistently as the parameter changes. The best result can be obtained when λ is around 0.1.

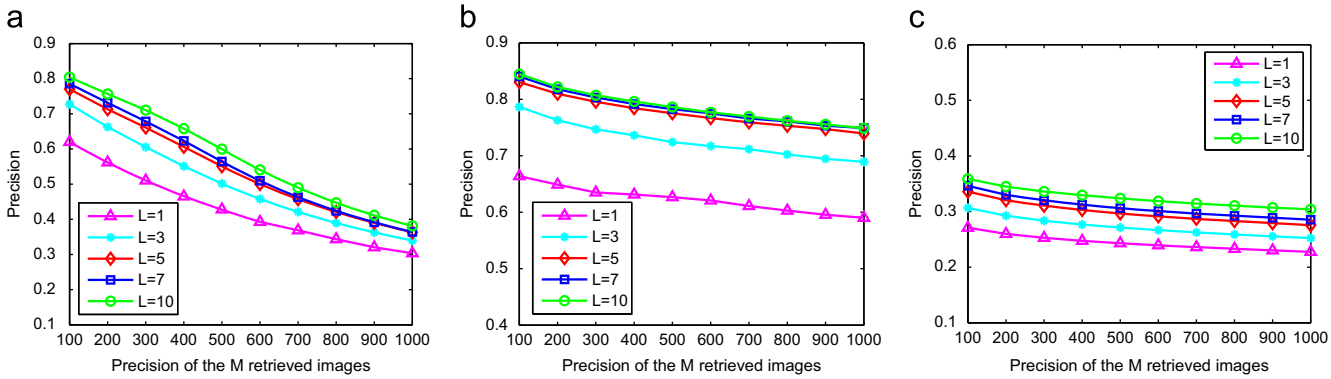

Fig. 6. The performance of our approach with different hash table numbers on the datasets: (a) USPS, (b) MNIST, and (c) CIFAR. (16-bits used).

Table 2
The comparison of training time on the MNIST dataset by different approaches with 32-bits. (69 000 training images, 2.13 GHz CPU).

Method	SH	SSH	SPLH	CH	LSH	DCH
Time (s)	5.3	3.4	95.8	18.6	293.0	570.4

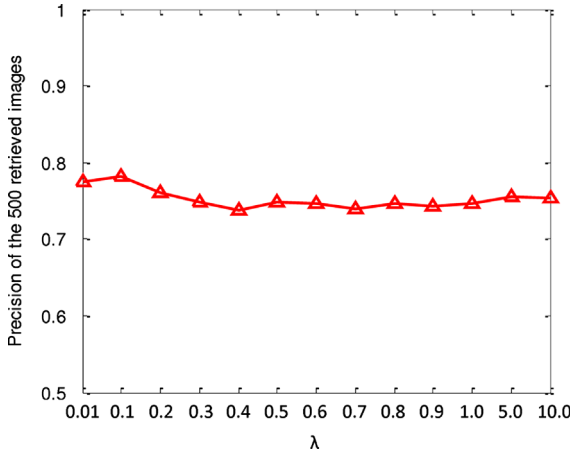

Fig. 7. The performance of our approach with respect to the parameter λ on the MNIST dataset (32-bits used).

5. Conclusion

In this paper, we propose a novel dual complementary hashing (DCH) approach to learn the binary codes for image description in fast image retrieval. Different from the traditional hashing methods that either learns the hash functions in a single shot or only employ a single hash table, our method learns both hash functions and hash tables in a complementary manner, such that the following ones can correct the errors generated by the previous ones. Therefore, the binary codes learned by our approach are more effective than the other methods. The extensive experimental results have shown that our approach can outperform the state-of-the-art hashing approaches.

Acknowledgment

This work was supported in part by the 973 Program under Project 2010CB327905, by the National Natural Science Foundation of China under Grants 61170127, 60975010, 60833006, and 61070104.

References

[1] S. Arya, D.M. Mount, N.S. Netanyahu, R. Silverman, A.Y. Wu, An optimal algorithm for approximate nearest neighbor searching, J. ACM 45 (6) (1998) 891–923.
[2] J.H. Friedman, J.L. Bentley, R.A. Finkel, An algorithm for finding best matches in logarithmic expected time, ACM Trans. Math. Software (TOMS) 3 (3) (1977) 209–226.
[3] N. Kumar, L. Zhang, S. Nayar, What is a good nearest neighbors algorithm for finding similar patches in images? in: Proceedings of the 10th European Conference on Computer Vision, 2008, pp. 364–378.
[4] M. Muja, D.G. Lowe, Fast approximate nearest neighbors with automatic algorithm configuration, in: VISAPP International Conference on Computer Vision Theory and Applications, 2009, pp. 331–340.
[5] C. Silpa-Anan, R. Hartley, Optimised KD-trees for fast image descriptor matching, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2008, pp. 1–8.
[6] B. Stein, Principles of hash-based text retrieval, in: Proceedings of the 30th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, 2007, pp. 527–534.
[7] A. Andoni, P. Indyk, Near-optimal hashing algorithms for approximate nearest neighbor in high dimensions, in: Proceedings of the 47th Annual IEEE Symposium on Foundations of Computer Science, 2006, pp. 459–468.
[8] Y. Weiss, A.B. Torralba, R. Fergus, Spectral hashing, in: Proceedings of the Conference on Advances in Neural Information Processing Systems, 2008, pp. 1753–1760.
[9] G. Hinton, R. Salakhutdinov, Reducing the dimensionality of data with neural networks, Science 313 (5786) (2006) 504–507.
[10] J. Wang, S. Kumar, S. Chang, Sequential projection learning for hashing with compact codes, in: Proceedings of the International Conference on Machine Learning, 2010.
[11] H. Xu, J. Wang, Z. Li, G. Zeng, S. Li, N. Yu, Complementary hashing for approximate nearest neighbor search, in: Proceedings of the 13th IEEE International Conference on Computer Vision, 2011.
[12] G. Shakhnarovich, T. Darrel, P. Indyk, Nearest Neighbor Methods in Learning and Vision: Theory and Practice (Neural Information Processing), The MIT Press, 2006.
[13] T.H. Cormen, C.E. Leiserson, R.L. Rivest, C. Stein, Introduction to Algorithms, 2nd ed., MIT Press, McGraw-Hill, 2001.
[14] R. Weber, H.-J. Schek, S. Blott, A quantitative analysis and performance study for similarity-search methods in high-dimensional spaces, in: Proceedings of the 24th International Conference on Very Large Data Bases (VLDB), 1998, pp. 194–205.
[15] M. Datar, N. Immorlica, P. Indyk, V. Mirrokni, Locality-sensitive hashing scheme based on p-stable distributions, in: Proceedings of the 20th Annual Symposium on Computational Geometry, 2004, pp. 253–262.
[16] R. Salakhutdinov, G. Hinton, Semantic hashing, International Journal of Approximate Reasoning 50 (7) (2009) 969–978.
[17] A. Gionis, P. Indyk, R. Motwani, Similarity search in high dimensions via hashing, in: Proceedings of the 25th International Conference on Very Large Data Bases (VLDB), 1999, pp. 518–529.
[18] G. Hinton, S. Osindero, Y.W. Teh, A fast learning algorithm for deep belief nets, Neural Computation 18 (7) (2006) 1527–1554.
[19] A.B. Torralba, R. Fergus, Y. Weiss, Small codes and large image databases for recognition, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2008, pp. 1–8.
[20] G. Shakhnarovich, P.A. Viola, T. Darrel, Fast pose estimation with parameter-sensitive hashing, in: Proceedings of the 9th IEEE International Conference on Computer Vision, 2003, pp. 750–759.
[21] F.R.K. Chung, Spectral Graph Theory, American Mathematical Society, 1997.
[22] W. Liu, J. Wang, S. Kumar, S. Chang, Hashing with graphs, in: Proceedings of the 28th International Conference on Machine Learning, 2011.
[23] D. Zhang, J. Wang, D. Cai, J. Lu, Self-taught hashing for fast similarity search, in: Proceedings of the 33rd International ACM SIGIR Conference on Research and Development in Information Retrieval, 2010.

[24] J. Wang, S. Kumar, S. Chang, Semi-supervised hashing for scalable image retrieval, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2010, pp. 3424–3431.
[25] Y. Mu, J. Shen, S. Yan, Weakly-supervised hashing in kernel space, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2010, pp. 3344–3351.
[26] A. Krizhevsky, Learning multiple layers of features from tiny images, Technical Report, University of Toronto, 2009.
[27] A. Oliva, A. Torralba, Modeling the shape of the scene: a holistic representation of spatial envelope, in: IJCV, 2001.
[28] Y. Pang, M. Shang, Y. Yuan, J. Pan, Scale invariant image matching using triplewise constraint and weighted voting, Neurocomputing 83 (9) (2012) 64–71.
[29] Y. Pang, W. Li, Y. Yuan, J. Pan, Fully affine invariant SURF for image matching, Neurocomputing 85 (3) (2012) 6–10.
[30] Y. Pang, Q. Hao, Y. Yuan, T. Hu, R. Cai, L. Zhang, Summarizing tourist destinations by mining user-generated travelogues and photos, Computer Vision and Image Understanding 115 (3) (2011) 352–363.
[31] M. Wang, X. Hua, J. Tang, R. Hong, Beyond distance measurement: constructing neighborhood similarity for video annotation, IEEE Trans. Multimedia 11 (3) (2009) 465–476.
[32] M. Wang, X. Hua, R. Hong, J. Tang, G. Qi, Y. Song, Unified video annotation via multi-graph learning, IEEE Trans. Circuits Syst. Video Technol. 19 (5) (2009) 733–746.
[33] S. Wang, Q. Huang, S. Jiang, Q. Tian, Nearest-neighbor classification using unlabeled data for real world image application, in: Proceedings of the ACM International Conference on Multimedia, 2010.

Peng Li received the B.S. degree in automation from Shandong University, in 2008. He is currently pursuing the Ph.D. degree at the National Laboratory of Pattern Recognition, Institute of Automation, Chinese Academy of Sciences, Beijing, China.

Jian cheng received the B.S. and M.S. degrees in mathematics from Wuhan University, Wuhan, China, in 1998 and 2001, respectively, and the Ph.D. degree in pattern recognition and intelligent systems from the Institute of Automation, Chinese Academy of Sciences, in 2004. He is currently an Assistant Professor with the Institute of Automation, Chinese Academy of Sciences, Beijing. His research interests include image and video retrieval, machine learning, etc.

Hanqing Lu received his B.E. degree, in 1982 and his M.E. degree, in 1985 from Harbin Institute of Technology, and Ph.D. degree from Huazhong University of Sciences and Technology, Wuhan, China, in 1992. He is a Professor with the National Laboratory of Pattern Recognition, Institute of Automation, Chinese Academy of Sciences. His current research interests include image similarity measure, video analysis, multimedia technology and system.