Communications in Information and Systems
Volume 17, Number 3, 133–146, 2017

Deep hashing using an extreme learning machine with convolutional networks

Zhiyong Zeng*, Shiqi Dai, Yunsong Li, Dunyu Chen

In this paper, we present a deep hashing approach for large scale image search. It is different from most existing deep hash learning methods which use convolutional neural networks (CNN) to execute feature extraction to learn binary codes. These methods could achieve excellent results, but they depend on an extreme huge and complex networks. We combine an extreme learning machine (ELM) with convolutional neural networks to speed up the training of deep learning methods. In contrast to existing deep hashing approaches, our method leads to faster and more accurate feature learning. Meanwhile, it improves the generalization ability of deep hashing. Experiments on large scale image datasets demonstrate that the proposed approach can achieve better results with state-of-the-art methods with much less complexity.

1. Introduction

With the growing of multimedia data, fast and effective search technique has become a hot research topic. Among existing search techniques, hashing is one of the most important retrieval techniques due to its fast query speed and low memory cost.

Hashing methods can be divided into two categories: data-independent [1–3], data-dependent [4–8]. For the first category, hash functions random generated are first used to map data into feature space and then binarization is carried out. Representative methods of this category are locality sensitive hashing (LSH) [1] and its variants [2, 3]. For the second category, hash functions are learned from training data to map samples into binary codes. Data-dependent methods can be further classified into two categories: unsupervised methods [11–13] and supervised methods[14–19]. The former only

*Research supported in part by the Key Research Funds of Fujian Province under Grant 2013H0020.

uses the feature information without utilizing supervised information during training. Representative unsupervised methods contain isotropic hashing [11], discrete graph hashing (DGH) [12], anchor graph hashing (AGH) [13] and iterative quantization (ITQ) [4]. The latter uses feature information and supervised information such as label to learn hash function. The label information is given in three forms: point-wise labels, pair-wise labels and ranking labels. Representative methods based on point-wise labels contain supervised discrete hashing (SDH) [14]. Representative methods based on pair-wise labels contain minimal loss hashing (MLH) [7], fast supervised hashing (FastH) [9], supervised hashing with kernels (KSH) [10], and convolutional neural networks hashing (CNNH) [15]. Representative methods based on ranking labels contain ranking-based supervised hashing (RSH) [16], ranking preserving hashing (RPH) [17], order preserving hashing (OPH) [18], and column generation hashing (CGH) [19].

Although many hashing methods have been proposed, most existing hashing methods utilize hand-crafted features. These hashing methods cannot well extract the nonlinear manifold structure of data points, the resulting features might not be optimally compatible with hash function. Hence, these hashing methods based on hand-crafted features might not obtain satisfactory performance in practice. Although some hashing methods based on kernel have been presented, they suffer from huge computational complex problem [10].

To solve the shortcoming of hashing methods based on the hand-crafted features, researchers have recently proposed some feature learning based deep hashing methods [20–24]. These deep hashing methods execute feature learning and hash function learning with deep neural networks, which can well learn the nonlinear manifold structure of data and have achieved better results than existing hashing methods using hand-crafted features. However, deep learning needs huge and deep neural networks, a large number of model parameters need to be trained. Hence, although stochastic gradient descent algorithm and backward propagation algorithm can be used in network model training, learning speed of deep neural networks is the main drawback of deep learning, which will affect its practical application.

Recently, extreme learning machine (ELM) has been paid more attention in computer vision community duo to its fast training speed and good generalization [25–27]. The parameters of hidden nodes of the ELM are randomly generated and need not to be trained. Hence, the structure of network of the ELM could be constructed before the samples are gained. However, the ELM will face some problems in natural scene applications because original ELM or its variants mainly focus on classification. In an ELM, feature

learning need first to be conducted before classification is carried out in various applications. Hence, it is feasible to design multilayer solutions to finish above tasks. ELM-based autoencoder is a good multilayer learning architecture [26]. However, the framework has not well utilized the advantages of ELM duo to without random feature mapping. ELM-based Deep belief network (DBN) is another good multilayer architecture [27], which can undertake pattern classification and attain better results than state-of-the-art methods.

Seen from the above analysis, the existing hashing methods cannot attain good generalization performance with fast learning velocity. Inspired by CNN and ELM theory, we propose a deep ELM framework for hash learning. The contributions of this paper are mainly as follows:

1) A deep ELM hash learning framework contains three key components. The first component is deep convolutional network to learn hierarchical image abstraction. The second component is ELM to fasten feature learning and attain good generalization. The third component is hash functions to map the learned image feature into binary codes. All components are seamlessly combined into the deep framework to map each image into hash codes. Unlike existing hash learning methods which try to find a linear projection to map each image into hash codes, we study a deep ELM network to seek multilayer nonlinear transformation to learn these binary vectors.

2) Experiments on large scale image datasets demonstrate that the proposed method can achieve better image retrieval results than state-of-the-art methods.

2. ELM-based learning algorithm

ELM was developed as a single-hidden layer feedforward neural network (SLFN) [28]. If only the activation functions of the neurons are piecewise continuous, the weights of hidden nodes in the ELM may be produced and need not to be changed. An ELM includes two stages, data projection and feature learning.

In ELM data projection stage, given input data $x \in R^d$, the output of ELM for generalized SLFN with L hidden neurons and activation function $g(x)$ is:

$$f(x) = \sum_{i=1}^{L} \beta_i g(w_i x_j + b_i) = H\beta, \quad j = 1, 2, \ldots, N \tag{1}$$

where $w_i = [w_{i1}, w_{i2}, \ldots, w_{in}]$ represents the weight vector between the ith hidden neuron and all input neurons, $\beta_i = [\beta_{i1}, \beta_{i2}, \ldots, \beta_{im}]$ is the weight vector between the ith hidden neuron and all output neurons, and b_i represents the bias of the ith hidden neuron. In an ELM, w_i and b_i are randomly generated based on a continuous probability function. H is the hidden layer output matrix of neural network; the ith column of H is the ith neuron's output vector in regard to inputs $x_1, x_2, \ldots, x_N$.

$$H = \begin{bmatrix} g(w_1 x_1 + b_1) & \cdots & g(w_L x_1 + b_L) \\ \vdots & \cdots & \vdots \\ g(w_1 x_N + b_1) & \cdots & g(w_L x_N + b_L) \end{bmatrix}, \beta = \begin{bmatrix} \beta_1^T \\ \vdots \\ \beta_L^T \end{bmatrix}_{L \times m} \tag{2}$$

H represents the ELM feature projection which maps the input data in R^d to feature space R^L.

In an ELM learning stage, denote $Y \in R^{N \times M}$ as the target matrix provided by N training data. $H = [g(x_1), g(x_2), \ldots, g(x_N)] \in R^{R \times L}$ contains N random feature projections produced in the ELM mapping stage. The objective function is the minimization of the weighted sum of the training error and the norm of output weights:

$$\arg\min(\lambda \parallel H\beta - Y \parallel_2^2 + \parallel \beta \parallel_2^2) \tag{3}$$

β can be optimized in a closed solution:

$$\beta = \begin{cases} \left(H^T H + \frac{1}{\lambda} I\right)^{-1} H^T Y, & L \leq K \\ H^T \left(H^T H + \frac{1}{\lambda} I\right)^{-1} Y, & L > K \end{cases} \tag{4}$$

where I is identity matrix with respective dimension.

3. Feature learning with ELM

Fig.1 shows the architecture of a deep ELM-based hashing learning. For the input image $f(x, y)$ of size $d \times d$, let the local receptive field be $r \times r$. Hence, the size of the feature maps is $(d - r + 1) \times (d - r + 1)$. If the deep ELM includes L layers of convolution and pooling, the lth layer takes the feature map of the $(l - 1)$th layer as input, the convolution between the randomly generated normalized convolutional kernels and input image at this layer will generates K feature maps. Each of random convolutional kernels is normalized so that their weights sum to 1. For a given layer l, its matrix of the normalized random convolution kernels is defined as:

$$W_l = [w_{l,k}]_{k=1}^{K} \subset R^{r\times r\times K},\ l = 1, 2, \ldots, L \tag{5}$$

$$w_{l,k}(i,j) = rand(0,1),\ i,j = 1, 2, \ldots, r \tag{6}$$

$$w_{l,k}(i,j) = w_{l,k}(i,j)/\Sigma_{i,j} w_{l,k}(i,j) \tag{7}$$

where W_l consists of K normalized random convolutional kernels $w_{l,k}$ of size $r \times r$, $rand(0,1)$ produces a random number in $[0,1]$.

Convolution calculation is executed over the feature maps of the previous layer. Specifically, the kth feature map at layer l can be attained as follows:

$$F_{l,k} = F_{l-1,k}(i,j) * w_{l,k} \tag{8}$$

where $*$ is convolution operating, $F_{l,k}$ is the feature map of layer l, $F_{l-1,k}$ is the feature map of layer $l-1$. For the input layer, $F_{l,k} = x$. To generate input for the next layer, then we use pooling operation to down-sample the feature map. For the lth layer, let s_l denote pooling size, the size of pooling map is $d_l/s_l \times d_l/s_l$. We attain the kth pooling map using average pooling operation over the kth feature map.

$$P_{l,k}(p,q) = \frac{1}{s_l^2}\Sigma_{i=(p-1)*s_l+1}^{p*s_l}\Sigma_{j=(p-1)*s_l+1}^{p*s_l} F_{l,k}(i,j), \quad p,q = 1, 2, \ldots, s_l \tag{9}$$

where $P_{l,k}$ represents the input feature map of the next layer: $F_{l+1,k} = P_{l,k}$.

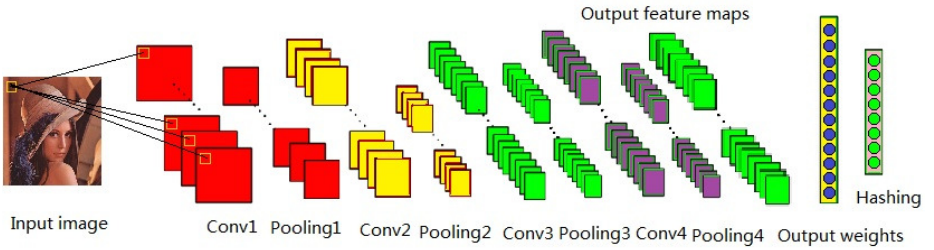

Figure 1: The architecture of a deep ELM hashing learning.

In the deep ELM feature learning phase, the feature maps of the last layer are next used to train the output weights. The last layer is in full connection with the output layer, as shown in Fig. 1. Then we concatenate the feature maps of all N training samples into N row vectors together, the full connection layer matrix H is attained, $H \in R^{N\times M}$, where $M =$

$K_L * (d_L/s_L)^2$. K_L, d_L and s_L are the number of feature maps of each image, the size of feature map, and pooling size, respectively, all for the last layer. Finally, the output weights β can be calculated as:

$$\beta = \begin{cases} \left(H^T H + \frac{1}{\lambda} I\right)^{-1} H^T Y, & N \leq M \\ H^T \left(H^T H + \frac{1}{\lambda} I\right)^{-1} Y, & N > M \end{cases} \tag{10}$$

In the testing phase, each test image goes through the whole network so that the feature map H of each test image can be attained. The weighted feature map of each test image can be calculated as follows:

$$H_{weighted} = \beta H \tag{11}$$

4. Binary code learning

After we obtain the weighted feature map of each image, we first use PCA projection to reduce the dimensionality of feature map, then we use iterative quantization strategy [4] to preserve the local structure of projected feature map so as to minimize the quantization error.

Suppose we have n images $X = \{x_i\}_{i=1}^n$, $x_i \in IR^d$, where x_i is the feature vector of image i. We assume that these images are zero-centered, i.e., $\Sigma_{i=1}^n x_i = 0$. The goal of hashing learning is to learn a binary code matrix $B = \{-1, 1\}^{n \times c}$ for all images, where c is the code length. For each bit

$$b_i = h(x_i) = [h_1(x_i), h_2(x_i), \ldots, h_c(x_i)]^T$$

, where $h(x_i)$ is a binary function to learn and

$$h(x_i) = sgn(v) = \begin{cases} 1, & \text{if } v \geq 0 \\ -1, & \text{otherwise} \end{cases} \tag{12}$$

To obtain an efficient binary code, we should maximize the variance of learned binary code and decorrelate the code bits of pairwise. We can adopt the following continuous objective function to finish the task.

$$\begin{aligned} L(W) &= \sum_k IE(\| xw_k \|_2^2) \\ &= \frac{1}{n} \sum_k w_k^T X^T X w_k \\ &= \frac{1}{n} tr(W^T X^T X W), \quad W^T W = I \end{aligned} \tag{13}$$

where the constraint $W^T W = I$ means that the hashing hyperplanes have to be orthogonal each other, which meets the requirement that the code bits is pairwise uncorrelated. The objective function is completely the same as that of PCA. Hence, we can take the top c eigenvectors of the data covariance matrix $X^T X$ to calculate W for a hash code of c bits.

Denote $v \in IR^c$ be a vector in PCA projected space, $sgn(v)$ is the vertex of the hypercube $\{-1, 1\}^c$ closest to v on the basis of euclidean distance. If W is an optimal solution of equation (13), then for any orthogonal $c \times c$ matrix R, there is $\widetilde{W} = WR$. Hence, we can rotate projected data $V = XW$ to minimize the following quantization loss:

$$\begin{aligned} Q(B, R) =& \| B - VR \|_F^2 \\ =& \| B \|_F^2 + \| V \|_F^2 - 2tr(BR^T V^T) \\ =& nc + \| V \|_F^2 - 2tr(BR^T V^T) \end{aligned} \tag{14}$$

where $\| \bullet \|_F$ is the Frobenius norm. Because the projected matrix V is constant, minimizing quantization loss is same as maxizing $tr(BR^T V^T) = \sum_{i=1}^{n} \sum_{j=1}^{c} B_{ij} \widetilde{V_{ij}}$, where $\widetilde{V_{ij}}$ is the elements of $\widetilde{V} = VR$, To maximize the expression with respect to B, we need to make $B = 1$ whenever $\widetilde{V_{ij}} \geq 0$ and $B = -1$ otherwise. For a fixed B, to minimize the expression (14), we need to try to find a rotation transform to align two point sets. In our project, the two point sets are provided by the projected matrix V and target binary code matrix B. For a fixed B, minimize the expression as follows: we first calculate the SVD of the $c \times c$ matrix $B^T V$ as $B^T V = S\Omega \widehat{S^T}$ and then let $R = \widehat{S} S^T$.

5. Experiments

We evaluate deep ELM on two datasets which are from the Tiny image dataset. The first one is CIFAR-10 dataset, it contains 60k color images from 10 classes. The size of each image is 32×32 . We randomly select 1000 images to serve as query data, 100 per class. The remaining 59000 images form the train set as well as the image database against which the queries are carried out. Each image is represented as 320 bins GIST feature vector. All experimental results reported in CIFAR-10 dataset are averaged over five random partitioned training/test datasets. The second dataset is MNIST, it contains 70k handwritten digit images from 10 object classes (labeled from 0 to 9). The size of each image is 28×28 . We randomly select 1000 images, 100 per class, to serve as query data. The remaining 69000 images are used as the

Table 1: Parameters of the proposed network.

param	Layer1	Layer2	Layer3	Layer4
FM	16×16	12×12	8×8	4×4
K_L	8	16	32	64
c_l	4	4	4	4
s_l	2	2	2	1

train set as well as the database against which the queries are carried out. Each image is represented as 784 bins feature vector by using its intensity. All experimental results reported in MNIST dataset are averaged over five random partitioned training/test datasets.

In our proposed method, we train our deep ELM network with 4 layers, where the parameters used in our implementation are described in Table 1, where FM, K_l, c_l and s_l represent feature map size, the number of convolutional kernel, normalized convolutional kernel size and pooling size of different layers used in our implementation, respectively. The regularization parameter λ are set as 0.001. We compare our proposed hashing methods against five state-of-the-art methods. The five methods include LSH [1], SKLSH [3], PCA-ITQ [4], SH [28] and PCAH [8]. For all of these compared methods, we use the code implementations of the corresponding methods provided by the original authors and use the default parameters adopted by these papers. To evaluate the performance of different hashing methods, we use the following three evaluation metrics: 1) mean average precision (mAP): which computes the area under the precision-recall curve; 2) precision curves at N samples which is the percentage of true neighbors among top N retrieved samples; and 3) recall curves at N samples which is the percentage of true neighbors among top N retrieved samples.

For CIFAR dataset, Figure 2 shows the precision vs. recall curves for different methods at 64 bits. Figure 3 is the recall vs. N retrieved samples curves for different methods at 64 bits. Figure 4 is the precision vs. N retrieved samples curves for different methods at 64 bits. Figure 5 is the comparison results of different methods at 16, 32, 48 and 64 bits, respectively. As can be seen, our method outperforms the other compared hashing methods. This is because on CIFAR dataset, the above five state-of-the-art hash methods use GIST feature, while the proposed method uses deep feature which has better discriminant power. Hence, higher semantic relevance can be obtained in the top retrieved samples. In addition, we also compare the proposed model with the CNNH [15] which applies a deep model. Our model

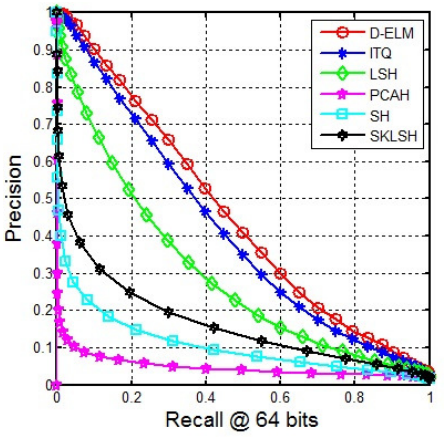

Figure 2: precision vs. recall curves on CIFAR dataset.

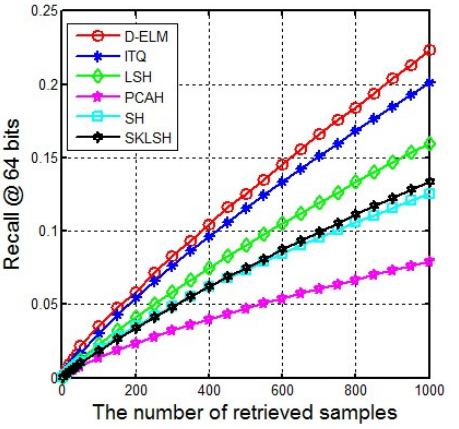

Figure 3: recall vs. N retrieved samples curves on CIFAR dataset.

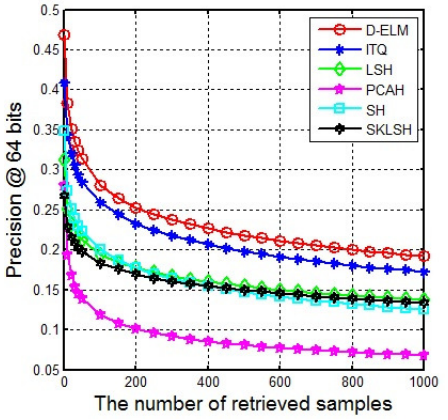

Figure 4: precision vs. N retrieved samples curves on CIFAR dataset.

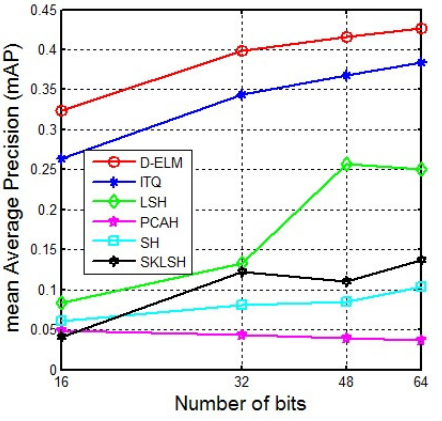

Figure 5: comparison of different hash functions on CIFAR dataset at 16, 32, 48 and 64 bits.

performs much better than the CNNH, the proposed method improves in average 2.6% image precision contrast to CNNH. Because our method not only minimize the reconstruction cost but also ensures balanced bits and independence of each transformation.

We compare the computational time of our proposed D-ELM with those state-of-the-art hashing methods. Our machine is configured with a 4.0GHz

Table 2: Computational cost of different hashing methods on CIFAR dataset.

Method	Training(seconds)	Test(microseconds)
ITQ	4.8	2.1
PCAH	0.5	0.05
LSH	0.09	0.07
SH	0.5	6.2
SKLSH	1.4	2.5
CNNH	354.6	3.8
D-ELM	23.7	3.0

Table 3: Comparison of different depth D-ELM on CIFAR dataset.

Dataset	one layer	two layer	four layer
CIFAR	41.5%	42.3%	43.6%
	13.4sec	16.3sec	23.7sec

CPU and 32.0 GB RAM. Table 2 demonstrates the training and test cost of different hashing algorithms over the CIFAR dataset when we evaluate the computational cost using 16-bit binary codes, where the test time is calculated for each query image. We see that the training time of deep learning methods are higher than state-of-the-art hashing methods, but our method is lower than CNNH because our method uses ELM to speed up feature learning process. Meanwhile, the test time of deep learning methods are comparable to the existing methods.

We also compare the performance of different layer versions of D-ELM over CIFAR dataset when 64-bits are used for evaluation. For all these structures of D-ELM, the number of filters for convolution takes 8 while the size of feature maps in the output layer is set to 8×8. From Table 3, we can see that multilayer D-ELM achieves better retrieval performance on CIFAR dataset. Because of multiple layer feature mapping, deep ELM usually takes more time to train.

For MNIST dataset, Figure 6 shows the precision vs. recall curves for different methods at 64 bits. Figure 7 is the recall vs. N retrieved samples curves for different methods at 64 bits. Figure 8 is the precision vs. N retrieved samples curves for different methods at 64 bits. Figure 9 is the

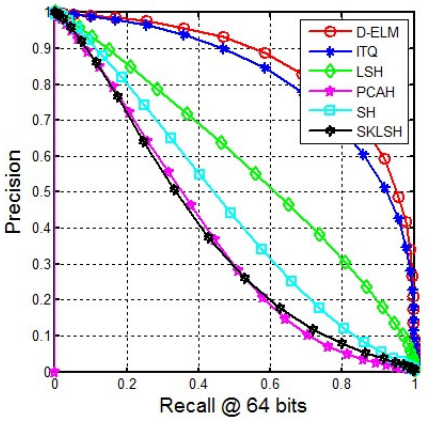

Figure 6: precision vs. recall curves on MNIST dataset.

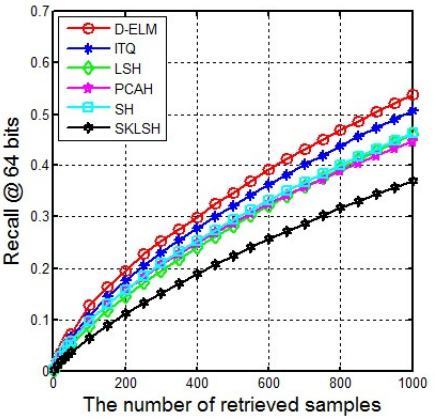

Figure 7: recall vs. N retrieved samples curves on MNIST dataset.

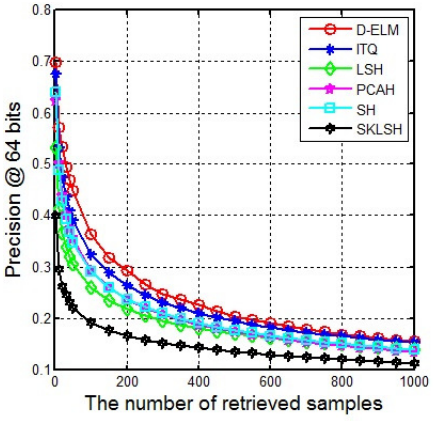

Figure 8: precision vs. N retrieved samples curves on MNIST dataset.

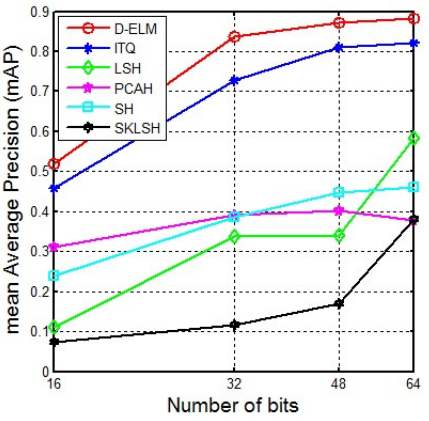

Figure 9: comparison of different hash functions on MNIST dataset at 16, 32, 48 and 64 bits.

comparison results of different methods at 16, 32, 48 and 64 bits, respectively. As can be seen, our method outperforms the other compared hashing methods.

6. Conclusion

In this paper, we have proposed a hashing method called deep ELM hashing (D-ELM) for compact binary codes learning. Experimental results on two widely used datasets showed the effectiveness of the proposed methods. How to apply our proposed methods to other vision applications such as object recognition and visual tracking seems an interesting future work.

References

[1] A. Andoni and P. Indyk, *Near-optimal hashing algorithms for approximate nearest neighbor in high dimensions*, in: Foundations of Computer Science, 2006. FOCS '06. IEEE Symposium, pages 459–468, 2006.

[2] P. Jain, B. Kulis, and K. Grauman, *Fast image search for learned metrics*, in: Computer Vision and Pattern Recognition, 2008, CVPR 2008, IEEE Conference, pages 1–8, 2009.

[3] M. Raginsky and S. Lazebnik, *Locality-sensitive binary codes from shift-invariant kernels*, in: Advances in Neural Information Processing Systems 22: Conference on Neural Information Processing Systems 2009. Vancouver, British Columbia, Canada, pages 1509–1517, 2009.

[4] Y. Gong and S. Lazebnik, *Iterative quantization: A procrustean approach to learning binary codes*, in: IEEE Conference on Computer Vision and Pattern Recognition, pages 817–824, 2011.

[5] K. He, F. Wen, and J. Sun, *K-means hashing: An affinity-preserving quantization method for learning binary compact codes*, in: IEEE Conference on Computer Vision and Pattern Recognition, pages 2938–2945, 2013.

[6] B. Kulis and T. Darrell, *Learning to hash with binary reconstructive embeddings*, in: International Conference on Neural Information Processing Systems, pages 1042–1050, 2009.

[7] M. Norouzi and D. J. Fleet, *Minimal loss hashing for compact binary codes*, in: International Conference on International Conference on Machine Learning, pages 353–360, 2011.

[8] J. Wang, S. Kumar, and S. F. Chang, *Semi-supervised hashing for large-scale search*, IEEE Transactions on Pattern Analysis & Machine Intelligence, **34** (2012), no. 12, 2393–2406.

[9] G. Lin, C. Shen, Q. Shi, A. V. D. Hengel, and D. Suter, *Fast supervised hashing with decision trees for high-dimensional data*, in: IEEE Conference on Computer Vision and Pattern Recognition, pages 1971–1978, 2014.

[10] S.-F. Chang, *Supervised hashing with kernels*, in: IEEE Conference on Computer Vision and Pattern Recognition, pages 2074–2081, 2012.

[11] W. Kong and W. J. Li, *Isotropic hashing*, in: International Conference on Neural Information Processing Systems, pages 1646–1654, 2012.

[12] W. Liu, S. Kumar, S. Kumar, and S. F. Chang, *Discrete graph hashing*, in: International Conference on Neural Information Processing Systems, pages 3419–3427, 2014.

[13] W. Liu, J. Wang, S. Kumar, and S. F. Chang, *Hashing with graphs*, 2011.

[14] F. Shen, C. Shen, W. Liu, and H. T. Shen, *Supervised discrete hashing*, pages 37–45, 2015.

[15] R. Xia, Y. Pan, H. Lai, C. Liu, and S. Yan, *Supervised hashing for image retrieval via image representation learning*, 2014.

[16] J. Wang, W. Liu, A. X. Sun, and Y. G. Jiang, *Learning hash codes with listwise supervision*, in: IEEE International Conference on Computer Vision, pages 3032–3039, 2014.

[17] Q. Wang, Z. Zhang, and L. Si, *Ranking preserving hashing for fast similarity search*, in: International Conference on Artificial Intelligence, pages 3911–3917, 2015.

[18] J. Wang, J. Wang, N. Yu, and S. Li, *Order preserving hashing for approximate nearest neighbor search*, pages 133–142, 2013.

[19] X. Li, G. Lin, C. Shen, A. V. D. Hengel, and A. Dick, *Learning hash functions using column generation*, Computer Science, pages 142–150, 2013.

[20] K. Lin, H. F. Yang, J. H. Hsiao, and C. S. Chen, *Deep learning of binary hash codes for fast image retrieval*, in: Computer Vision and Pattern Recognition Workshops, pages 27–35, 2015.

[21] H. Lai, Y. Pan, Y. Liu, and S. Yan, *Simultaneous feature learning and hash coding with deep neural networks*, pages 3270–3278, 2015.

[22] R. Zhang, L. Lin, R. Zhang, W. Zuo, and L. Zhang, *Bit-scalable deep hashing with regularized similarity learning for image retrieval and person re-identification*, IEEE Transactions on Image Processing, **24** (2015), no. 12, 4766–4779.

[23] F. Zhao, Y. Huang, L. Wang, and T. Tan, *Deep semantic ranking based hashing for multi-label image retrieval*, in: Computer Vision and Pattern Recognition, pages 1556–1564, 2015.

[24] V. E. Liong, J. Lu, G. Wang, P. Moulin, and J. Zhou, *Deep hashing for compact binary codes learning*, in: Computer Vision and Pattern Recognition, pages 2475–2483, 2015.

[25] G. B. Huang, Z. Bai, and M. V. Chi, *Local receptive fields based extreme learning machine*, IEEE Computational Intelligence Magazine, **10** (2015), no. 2, 18–29.

[26] J. Tang, C. Deng, and G. B. Huang, *Extreme learning machine for multilayer perceptron*, IEEE Transactions on Neural Networks & Learning Systems, **27** (2016), no. 4, 809.

[27] L. L. Cao, W. B. Huang, and F. C. Sun, *A deep and stable extreme learning approach for classification and regression*, 2015.

[28] Y. Weiss, A. Torralba, and R. Fergus, *Spectral hashing*, in: International Conference on Neural Information Processing Systems, pages 1753–1760, 2008.

College of Mathematics and Informatics, Fujian Normal University
Qishan Campus, Minghou District,Fuzhou,350108, China
E-mail address: zzyong@fjnu.edu.cn

College of Mathematics and Informatics, Fujian Normal University
Qishan Campus, Minghou District,Fuzhou,350108, China
E-mail address: 476712987@qq.com

School of Communication and Engineering, Xidian University
Xi'an, 710071, China
E-mail address: ysli@mail.xidian.edu.cn

College of Electrical Engineering and Computer Science, Yuan Ze University, zhongli, Taoyuan, 32003, Taiwan
E-mail address: dychen@saturn.yzu.edu.tw