Deep Top Similarity Preserving Hashing for Image Retrieval

Qiang Li, Haiyan Fu, and Xiangwei Kong(✉)

School of Information and Communication Engineering,
Dalian University of Technology, Dalian 116024, China
kongxw@dlut.edu.cn

Abstract. As a kind of approximate nearest neighbor search method, hashing is widely used in large scale image retrieval. Compared to traditional hashing methods, which first encode each image through hand-crafted features and then learn hash functions, deep hashing methods have shown superior performance for image retrieval due to its learning image representations and hash functions simultaneously. However, most existing deep hashing methods mainly consider the semantic similarities among images. The information of images' positions in the ranking list to the query image has not yet been well explored, which is crucial in image retrieval. In this paper, we propose a Deep Top Similarity Preserving Hashing (DTSPH) method to improve the quality of hash codes for image retrieval. In our approach, when training the convolutional neural network, a top similarity preserving hashing loss function is designed to preserve similarities of images at the top of the ranking list. Experiments on two benchmark datasets show that our proposed method outperforms several state-of-the-art deep hashing methods and traditional hashing methods.

Keywords: Image retrieval · Deep hashing · Top similarity preserving

1 Introduction

Image retrieval [17–23] has received increasing attention in computer vision. With the rapid growth of large-scale image data, hashing has attracted more and more attention in image retrieval due to its fast search speed and low storage cost. The purpose of hashing is to learn a set of hash functions that map each image to binary codes while trying best to preserve the semantic similarities among images.

Based on whether using deep convolutional neural network to learn hash codes, hashing methods can be divided into two categories: traditional hashing and deep hashing. Traditional hashing methods first extract hand-crafted feature vectors of each image and then learn hash functions based on these feature vectors. Traditional hashing methods can be further categorized into unsupervised methods and supervised methods. In unsupervised methods, only unlabeled data is used during the training procedure. Representative unsupervised methods contain iterative quantization hashing (ITQ) [4] and topology preserving hashing (TPH) [14]. Supervised methods utilize label information to assist the learning of hash functions, which can improve the quality of hash codes. Representative supervised methods include minimal loss hashing (MLH) [10] and supervised hashing with kernels (KSH) [9].

Compared to traditional hashing methods, deep hashing [8,15,16] methods show their superior performance for image retrieval. Usually, deep hashing methods use CNNs [6] to extract discriminative image representations and learn hash functions simultaneously. For example, Xia *et al.* [13] proposed CNNH, which is a two stage learning method that first learns approximate hash codes from the pairwise similarity matrix and then utilizes CNN to learn image representations and hash functions. Lai *et al.* [8] proposed DNNH, in which the deep architecture used a triplet ranking loss function to preserve relative similarities. Zhang *et al.* [15] presented a novel bit-scalable deep hashing approach DRSCH.

Although the existing deep hashing methods have brought substantial improvements over traditional supervised hashing methods. Most of them mainly consider similarities among images. The information of images' positions in the ranking list to the query image has not yet been well explored. However, users always pay their most attention to the images ranked in the top. So whether images ranked in the top are similar to the query image or not is crucial to the quality of image retrieval.

In this paper, we propose deep top similarity preserving hashing (DTSPH) to generate high quality of hash codes for image retrieval. As shown in Fig. 1, we utilize CNN to extract discriminative image representations and learn hash functions directly from images. At the top of the CNN model, a top similarity preserving hashing loss function is designed to preserve the similarities in the top of the ranking list. Each time, a group of images are fed into the CNN model and then the stochastic gradient descent algorithm is used to train model parameters. Experimental results on two benchmark datasets show that our proposed method has superior performance over several other deep hashing methods including traditional hashing methods with CNN features. The rest of this paper is organized as follows. In Sect. 2, we introduce our deep top similarity preserving hashing method. Then experimental results are shown in Sect. 3. Finally, we give a conclusion in Sect. 4.

2 Deep Top Similarity Preserving Hashing

In this section, we first introduce our deep top similarity preserving hashing loss function. Then we give its gradients which are vital to back-propagation algorithm.

2.1 Deep Hash Model

We adopt the architecture of AlexNet [6] as our basic framework, which has five convolutional layers($conv_1 - conv_5$) with optional pooling layers, followed

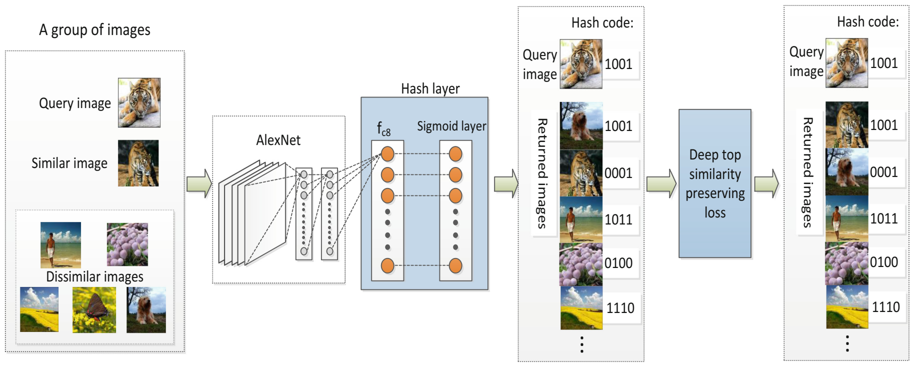

Fig. 1. Overview of DTSPH framework. During training stage, the input to the CNN model is a group of images, i.e., $\{I^q, I^s, \{I_k^d\}_{k=1}^n\}$, where I^q is the query image, I^s is similar to I^q and $\{I_k^d\}_{k=1}^n$ are dissimilar to I^q. Through the CNN model, each image is encoded into a hash code. Then these hash codes are used to calculate the deep top similarity preserving loss, which aims to optimize the parameters of the model to preserve the similarities at the top of the ranking list.

by two fully connected layers($f_{c6} - f_{c7}$) and the classification output layer. The activation function of the first seven layers of AlexNet is the rectified linear units (ReLUs), which is much faster than these saturating nonlinearities such as the logistic sigmoid and the hyperbolic tangent in terms of training time with gradient descent [3]. For more details about the configurations of the convolutional layers, pooling layers and fully connected layers, please refer to [6]. In order to adopt to learning hash functions, we replace the classification output layer in original AlexNet with a hash layer as shown in Fig. 1. The goal of hash functions is to encode feature representation into hash codes. As we can see from Fig. 1, our hash layer is a fully connected layer followed by a sigmoid layer, which transform the 4096-dimension feature of the layer f_{c7} into q-dimension output and the value is restricted to [0, 1]. Then the K-bit hash codes are obtained by quantifying the K-dimension output. Usually, the performance of deep hash method is related to two aspects. One is the deep model. Usually, the model is deeper, the performance is better. The other is the loss function, which is used to guide training the deep model. Here, we focus on the loss function, not the deep model. Please note that our DTSPH algorithm also could be easily applied to other models, such as VGG [2], and we will also test our method on VGG-F model [2].

2.2 Top Similarity Preserving

For an Information Retrieval, correctly ranking documents on the top of the result list is crucial [1]. While for an Image Retrieval, user usually pay most of their attentions to the results on the first few pages [16]. However, most of deep hash methods mainly consider similarities among images and few of them explore the information of images' positions in the ranking list. Considering it, we carefully design a top similarity preserving loss function, which mainly preserves the similarities in the top of ranking list to explore the information of positions.

Let $\mathcal{I}$ be the image space and $f_{c7}(I_i) \in \mathbb{R}^d$ denote the feature vector from the output of the layer f_{c7} by feeding the image $I_i \in \mathcal{I}$ into the CNN model. Then we can obtain the binary codes $b(I_i) \in \mathbb{H}^K \equiv \{0,1\}^K$ as follows:

$$b(I_i) = sgn(\mathbf{w}^T f_{c7}(I_i)) \tag{1}$$

where $\mathbf{w} \in \mathbb{R}^{d \times K}$ denotes the weights of hash layer. Here for the sake of conciseness, bias terms of hash layer and parameters of the first seven layers are omitted. $sgn(x)$ is a sign function that $sgn(x) = 1$ if $x > 0$ and 0 otherwise, and it performs element-wise operations.

Considering the information of images' positions in the ranking list, during training stage, each time, we feed a group of images to the CNN model, among which the query image is I^q, whose similar image is I^s and dissimilar images are $\{I_k^d\}_{k=1}^n$, where n is the number of dissimilar images. Then we get the corresponding hash codes, denoted as $b(I^q)$, $b(I^s)$, and $\{b(I_k^d)\}_{k=1}^n$ respectively. According to their Hamming distance to the query image, we can obtain a ranking list for the query image I^q. Intuitively, we expect that the similar image I^s will be in the top of the ranking list, which means that the Hamming distance between $b(I^q)$ and $b(I^s)$ is smaller than that between $b(I^k)$ and $b(I^s)$ for any $k \in \{1, n\}$. Then following [12], the "rank" of the similar image I^s with respect to the query image I^q is defined as following:

$$R(I^q, I^s) = \sum_{k=1}^{n} I\{(\|b(I^q) - b(I^s)\|_{\mathrm{H}} - \|b(I^q) - b(I_k^d)\|_{\mathrm{H}}) > 0\} \tag{2}$$

where $\|\cdot\|_H$ represents Hamming distance. $I(\mathit{condition})$ is an indicator function that $I(\mathit{condition}) = 1$ if the *condition* is true and 0 otherwise. The function $R(I^q, I^s)$ counts the number of the dissimilar images I^k, which are closer to the query image I^q than the similar image I^s in terms of Hamming distance. Obviously, $R(I^q, I^s)$ is relative small if the similar image I^q is ranked in the top of the ranking list and is relative large otherwise.

Usually, whether images ranked in the top of the ranking list are similar to the query image or not is crucial in image retrieval [16]. Considering that, we define a top similarity preserving loss function:

$$L(R(I^q, I^s)) = -\frac{\psi^2}{\psi + R(I^q, I^s)} \tag{3}$$

where the parameter ψ controls the decreasing speed of the first order derivative $L'(R)$ of $L(R)$. With the increase of R, the loss $L(R)$ increases relative quickly when R is small than that when R is large. The loss function penalizes the mistakes in the top of the ranking list more than those in the bottom, which means the similarity in top of the ranking list is more important than that in the bottom.

Then, the objective function is defined as following:

$$\mathcal{L} = \frac{1}{Z} \sum_q \sum_s - \frac{\psi^2}{\psi + R(I^q, I^s)} \tag{4}$$

where Z is the number of $\{I^q, I^s, \{I_k^d\}_{k=1}^n\}$.

Although the ranking loss in Eq. (3) is differentiable, the objective function in Eq. (4) is still not differentiable due to Eqs. (1) and (2). For ease of optimization, Eq.(1) is relaxed as:

$$f_h(I_i) = sigmoid(\mathbf{w}^T f_{c7}(I_i)) \tag{5}$$

where $sigmoid(\cdot)$ function is used to approximate $sgn(\cdot)$ function and $f_h(I_i)$ denotes the output of the hash layer for image I_i. Please note, as shown in Fig. 1, the hash layer of our model is a fully connected layer followed by a sigmoid layer essentially. Then we can get the hash codes with respect to the image I_i as following:

$$b(I_i) = sgn(f_h(I_i) - 0.5) \tag{6}$$

Next, in Eq. (2), the Hamming norm is replaced with the l_1 norm and the indicator function $I(\cdot)$ is approximated by the $sigmoid(\cdot)$ function. Accordingly, Eq. (2) can be written as:

$$\begin{aligned} \hat{R}(I^q, I^s) = &\sum_{k=1}^{n} sigmoid(\|f_h(I^q) - f_h(I^s)\|_1 - \\ &\|f_h(I^q) - f_h(I_k^d)\|_1) \end{aligned} \tag{7}$$

Finally, the overall objective function is formulated as following:

$$\begin{aligned} \mathcal{L} = &\frac{1}{Z} \sum_q \sum_s - \frac{\psi^2}{\psi + \hat{R}(I^q, I^s)} + \frac{\alpha}{2} \sum_i \|b(I_i) - f_h(I_i)\|_2 \\ &+ \frac{\beta}{2} \| \underset{i}{mean}(f_h(I_i) - 0.5)\|_2 + \frac{\lambda}{2} \|\mathbf{W}\|_2 \end{aligned} \tag{8}$$

The second term is used to minimize the quantization error between the output of the hash layer and the corresponding hash codes, which could minimize the loss of information. The third term is used to make each bit averaged over the training data [11], which in some sense is to maximize the entropy of binary codes. So the binary codes would be compact and discriminative. The fourth term is regularizer term to control the scales of the weights so as to reduce the effect of the over-fitting. α, β and λ are three parameters to control the effect of above three terms.

2.3 Learning Algorithm

Stochastic gradient descent method is used to train the network parameters. Each time, we randomly select Z (during training stage, Z is the mini-batch) group of images from training set. Each group consists of $n+2$ images. One image as the query image I^q, one as the similar image I^s, and n images as dissimilar images $\{I_k^d\}_{k=1}^n$. We feed the network with these Z groups of images to get the corresponding output. The gradients of the objective function $\mathcal{L}$ with respect to $f_h(I^q)$, $f_h(I^s)$, $\{f_h(I_k^d)\}_{k=1}^n$ are as following:[1]

$$\begin{aligned}\frac{\partial\mathcal{L}}{\partial f_h(I^q)} =& \frac{1}{Z}\frac{\psi^2}{(\psi+\hat{R}(I^q,I^s))^2}\frac{\partial\hat{R}(I^q,I^s)}{\partial f_h(I^q)} \\ &+ \alpha(f_h(I^q)-b(I^q)) + \frac{\beta}{n+2}(\underset{i}{mean}(f_h(I_i)-0.5))\end{aligned} \tag{9}$$

$$\begin{aligned}\frac{\partial\mathcal{L}}{\partial f_h(I^s)} =& \frac{1}{Z}\frac{\psi^2}{(\psi+\hat{R}(I^q,I^s))^2}\frac{\partial\hat{R}(I^q,I^s)}{\partial f_h(I^s)} \\ &+ \alpha(f_h(I^s)-b(I^s)) + \frac{\beta}{n+2}(\underset{i}{mean}(f_h(I_i)-0.5))\end{aligned} \tag{10}$$

$$\begin{aligned}\frac{\partial\mathcal{L}}{\partial f_h(I_k^d)} =& \frac{1}{Z}\frac{\psi^2}{(\psi+\hat{R}(I^q,I^s))^2}\frac{\partial\hat{R}(I^q,I^s)}{\partial f_h(I_k^d)} \\ &+ \alpha(f_h(I_k^d)-b(I_k^d)) + \frac{\beta}{n+2}(\underset{i}{mean}(f_h(I_i)-0.5))\end{aligned} \tag{11}$$

where the *mean* operation is performed on Z groups of images. Then these gradient values would be fed into the network via back-propagation algorithm to update the parameters of each layer.

3 Experiments

3.1 Datasets and Settings

We validate our algorithm on two benchmark datasets: (1) The CIFAR10 dataset contains 60,000 32×32 color images of 10 classes; (2) The NUS-WIDE dataset consists of 269,648 images. Following [24], we use the subset of images annotated with the 21 most frequently happened classes. For CIFAR10, we use 10,000 images as the query samples, and the rest as the training samples. For NUS-WIDE, we randomly select 100 images from each of the 21 class labels as the query samples, and the rest as the training samples. Whether two images are similar or not depends on whether they share at least one common label.

We compare our method with five state-of-the-art hashing methods, including three traditional hashing methods KSH [9], MLH [10] and BRE [7], and two deep hashing methods, DSRH [16] and DRSCH [15].

[1] $\frac{\partial\hat{R}(I^q,I^s)}{\partial f_h(I^q)}$, $\frac{\partial\hat{R}(I^q,I^s)}{\partial f_h(I^s)}$, and $\frac{\partial\hat{R}(I^q,I^s)}{\partial f_h(I_k^d)}$ are easy to compute. Due to space limitations, we don't give the specific expressions of above terms here.

We implement our method by the open-source Caffe [5] framework. In all the experiments, the weights of layers F_{1-7} of our network are initialized by the weights of AlexNet [6] which has been trained on the ImageNet dataset, while the weights of other layers are initialized randomly. The mini-batch is set to 64. The parameter ψ, α and β are empirically set as 20, 0.01 and 0.1 respectively.

3.2 Results

We use four evaluation metrics for the performance comparison, which are mean average precision (mAP), precision within Hamming distance 2, precision at top 500 samples of different code lengths and precision curves with 64 bits with respect to different numbers of top returned samples.

Performance on CIFAR10. For CIFAR10, we follow [15] that the query image is searched within the query set itself. Table 1 shows the mAP values of CIFAR10 with different code lengths. From Table 1, we can find that DTSPH achieves better performance than other traditional hashing and deep hashing methods. Specially, DTSPH improves the mAP values of 48 bits to 0.805 from 0.631 achieved by DRSCH [15]. In addition, DTSPH shows an improvement of 34.7% compared to KSH [9] with CNN features. Figure 2 shows the comparison results of other three evaluation metrics on CIFAR10. As we can see, DTSPH has better performance gains over the other five methods.

Table 1. Accuracy in terms of mAP w.r.t. different number of bits on two datasets. For NUS-WIDE, the mAP value is calculated within the top 50,000 returned neighbors.

Method	CIFAR10(mAP)				NUS-WIDE(mAP)			
	16bits	24bits	32bits	48bits	16bits	24bits	32bits	48bits
DTSPH	**0.783**	**0.800**	**0.803**	**0.805**	**0.773**	**0.787**	**0.788**	**0.789**
DRSCH [15]	0.615	0.622	0.629	0.631	0.618	0.622	0.623	0.628
DSRH [16]	0.608	0.611	0.617	0.618	0.609	0.618	0.621	0.631
KSH-CNN [9]	0.401	0.430	0.444	0.458	0.607	0.619	0.625	0.626
MLH-CNN [10]	0.250	0.289	0.313	0.319	0.525	0.559	0.568	0.581
BRE-CNN [7]	0.198	0.206	0.206	0.216	0.538	0.558	0.566	0.576
KSH [9]	0.322	0.352	0.365	0.383	0.546	0.556	0.562	0.567
MLH [10]	0.133	0.158	0.163	0.180	0.487	0.507	0.511	0.524
BRE [7]	0.122	0.156	0.161	0.172	0.486	0.515	0.518	0.528

Performance on NUS -WIDE. Due to that NUS-WIDE is a relative big dataset, so we calculate mAP values within the top 50,000 returned samples. At the right of Table 1, the mAP values of different methods with different code lengths are shown. DTSPH shows superiority over other five methods again.

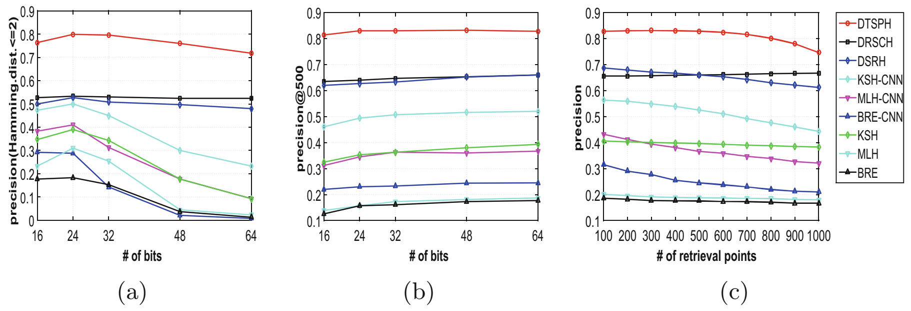

Fig. 2. The results on CIFAR10: (a) Precision curves within Hamming distance 2; (b) Precision curves within top 500 returned; (c) Precision curves with 64 bits w.r.t. different numbers of top returned samples.

Specially, DTSPH obtains a mAP of 0.789 with 48bits while the mAP value of DRSCH is 0.628 and the mAP value of KSH-CNN is 0.626. Figure 3(a) reports the precision curves within Hamming radius 2. Figure 3(b) shows the precision curves within top 500 returned. Figure 3(c) shows the precision curves of top 1000 returned samples with 64 bits. Our approach DTSPH shows better search accuracies than other five methods.

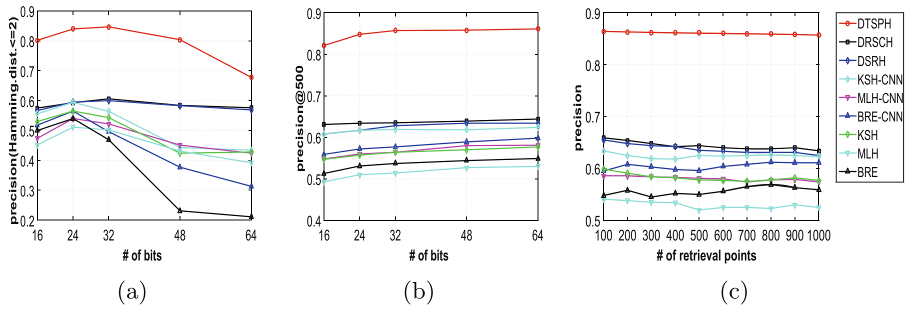

Fig. 3. The results on NUS-WIDE: (a) Precision curves within Hamming distance 2; (b) Precision curves within top 500 returned; (c) Precision curves with 64 bits w.r.t. different numbers of top returned samples.

Note that DRSCH and DSRH are deep hashing methods and they mainly consider the similarities among images. Experimental results on two benchmark datasets show that DTSPH is superior to DRSCH [15] and DSRH [16]. It demonstrates the effectiveness of DTSPH, which tries to preserve similarities of images in the top of the ranking list.

4 Conclusions

In this paper, we propose a deep top similarity preserving hashing (DTSPH) method to improve the quality of hash codes for image retrieval. We use CNN to extract discriminative image representations and learn hash functions simultaneously. At the top the CNN model, a deep top similarity preserving hashing loss function is designed to preserve the similarities at the top of the ranking list. The bigger dissimilarities at the top of the ranking list, the larger penalties are. Experimental results on two benchmark datasets show that DTSPH has superior performance against several state-of-the-art deep hashing and traditional hashing methods.

References

1. Cao, Y., Xu, J., Liu, T.Y., Li, H., Huang, Y., Hon, H.W.: Adapting ranking SVM to document retrieval. In: Proceedings of the 29th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, pp. 186–193. ACM (2006)
2. Chatfield, K., Simonyan, K., Vedaldi, A., Zisserman, A.: Return of the devil in the details: delving deep into convolutional nets. arXiv preprint arXiv:1405.3531 (2014)
3. Glorot, X., Bordes, A., Bengio, Y.: Deep sparse rectifier neural networks. In: Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics, pp. 315–323 (2011)
4. Gong, Y., Lazebnik, S., Gordo, A., Perronnin, F.: Iterative quantization: a procrustean approach to learning binary codes for large-scale image retrieval, vol. 35, pp. 2916–2929. IEEE (2013)
5. Jia, Y., Shelhamer, E., Donahue, J., Karayev, S., Long, J., Girshick, R., Guadarrama, S., Darrell, T.: Caffe: convolutional architecture for fast feature embedding. In: Proceedings of the 22nd ACM International Conference on Multimedia, pp. 675–678. ACM (2014)
6. Krizhevsky, A., Sutskever, I., Hinton, G.E.: Imagenet classification with deep convolutional neural networks. In: Advances in Neural Information Processing Systems, pp. 1097–1105 (2012)
7. Kulis, B., Darrell, T.: Learning to hash with binary reconstructive embeddings. In: Advances in Neural Information Processing Systems, pp. 1042–1050 (2009)
8. Lai, H., Pan, Y., Liu, Y., Yan, S.: Simultaneous feature learning and hash coding with deep neural networks. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 3270–3278 (2015)
9. Liu, W., Wang, J., Ji, R., Jiang, Y.G., Chang, S.F.: Supervised hashing with kernels. In: 2012 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp. 2074–2081. IEEE (2012)
10. Norouzi, M., Blei, D.M.: Minimal loss hashing for compact binary codes. In: Proceedings of the 28th International Conference on Machine Learning (ICML 2011), pp. 353–360 (2011)
11. Norouzi, M., Fleet, D.J., Salakhutdinov, R.R.: Hamming distance metric learning. In: Advances in Neural Information Processing Systems, pp. 1061–1069 (2012)

12. Song, D., Liu, W., Meyer, D.A., Tao, D., Ji, R.: Rank preserving hashing for rapid image search. In: Data Compression Conference, DCC 2015, pp. 353–362. IEEE (2015)
13. Xia, R., Pan, Y., Lai, H., Liu, C., Yan, S.: Supervised hashing for image retrieval via image representation learning. AAAI **1**, 2156–2162 (2014)
14. Zhang, L., Zhang, Y., Tang, J., Gu, X., Li, J., Tian, Q.: Topology preserving hashing for similarity search. In: Proceedings of the 21st ACM International Conference on Multimedia, pp. 123–132. ACM (2013)
15. Zhang, R., Lin, L., Zhang, R., Zuo, W., Zhang, L.: Bit-scalable deep hashing with regularized similarity learning for image retrieval and person re-identification. TIP **24**(12), 4766–4779 (2015)
16. Zhao, F., Huang, Y., Wang, L., Tan, T.: Deep semantic ranking based hashing for multi-label image retrieval. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 1556–1564 (2015)
17. Zheng, L., Wang, S., Liu, Z., Tian, Q.: Lp-norm IDF for large scale image search. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 1626–1633 (2013)
18. Zheng, L., Wang, S., Liu, Z., Tian, Q.: Packing and padding: coupled multi-index for accurate image retrieval. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 1939–1946 (2014)
19. Zheng, L., Wang, S., Tian, L., He, F., Liu, Z., Tian, Q.: Query-adaptive late fusion for image search and person re-identification. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 1741–1750 (2015)
20. Zheng, L., Wang, S., Tian, Q.: Coupled binary embedding for large-scale image retrieval. IEEE Trans. Image Process. **23**(8), 3368–3380 (2014)
21. Zheng, L., Wang, S., Wang, J., Tian, Q.: Accurate image search with multi-scale contextual evidences. Int. J. Comput. Vis. **120**(1), 1–13 (2016)
22. Zheng, L., Wang, S., Zhou, W., Tian, Q.: Bayes merging of multiple vocabularies for scalable image retrieval. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 1955–1962 (2014)
23. Zheng, L., Yang, Y., Tian, Q.: Sift meets CNN: a decade survey of instance retrieval. IEEE Trans. Pattern Anal. Mach. Intell. https://doi.org/10.1109/TPAMI.2017.2709749
24. Zhu, X., Zhang, L., Huang, Z.: A sparse embedding and least variance encoding approach to hashing. TIP **23**(9), 3737–3750 (2014)