DEEP PIECEWISE HASHING FOR EFFICIENT HAMMING SPACE RETRIEVAL

Jingzi Gu[1], *Dayan Wu*[1*], *Peng Fu*[1], *Bo Li*[1], *Weiping Wang*[1]

[1]Institute of Information Engineering, Chinese Academy of Sciences, Beijing, China
{gujingzi,wudayan,fupeng,libo,wangweiping}@iie.ac.cn

ABSTRACT

Hamming space retrieval can achieve constant-time image search, which is more efficient than linear scan. In Hamming space retrieval, the data points inside the Hamming ball imply retrievable while the data points outside are irretrievable. Therefore, it is crucial to explicitly characterize the Hamming ball. However, for the existing Hamming space retrieval methods, many similar points are found close to the outside of the Hamming ball while many dissimilar points are found close to the query point, leading to the decline of both retrieval accuracy and recall. In this paper, we present a novel method named Deep Piecewise Hashing (DPH), for Efficient Hamming Space Retrieval. A piecewise loss is elaborately designed to guide the learning of hash codes. Meanwhile, a piecewise probability distribution is introduced in the proposed loss function. The piecewise probability distribution pays more attention to the learning of those "marginal" similar points. It considers both discrimination and robustness for the dissimilar points inside the Hamming ball. Comprehensive experiments on two datasets, MS-COCO and NUS-WIDE, demonstrate that DPH can yield state-of-the-art Hamming space retrieval performance.

Index Terms— Image retrieval, Hamming space retrieval, Deep hashing, Piecewise hashing

1. INTRODUCTION

With the tremendous progress of deep learning, deep hashing methods have shown their superiority since they learn more meaningful semantic hash codes with deep neural networks (DNNs)[1, 2, 3, 4, 5]. Deep hashing methods [6, 7, 8, 9, 10, 11, 12] employ end-to-end deep networks to learn feature representations and binary codes, which have achieved excellent retrieval performance. However, most of the existing deep hashing methods still aim at handling the traditional linear scan scenario, and the time complexity of this scenario is still high for large-scale databases.

Recently, research on Hamming space retrieval [13, 14] has become popular due to its query efficiency. Hamming space retrieval can achieve constant-time search by hash lookups instead of linear scan only when Hamming radius $H \leq 2$. In this paper, we focus on the Hamming space retrieval scenario, where for each query, only points within the given Hamming ball are returned as relevant points while those outside are irretrievable. Deep Cauchy Hashing (DCH) [13] is the first deep hashing method for Hamming space retrieval. This work introduces Cauchy distribution into loss function which significantly penalize points inside Hamming ball, but it does not consider that the Hamming radius can also affect the retrieval accuracy. Then, the state-of-the-art Hamming space retrieval deep hashing method Maximum-Margin Hamming Hashing(MMHH) [14] is proposed. MMHH directly embodies the Hamming radius into the loss functions, as shown in Fig. 1. The probability is a constant within the Hamming ball and drops sharply outside the Hamming ball. It attracts more similar pairs into the Hamming ball and considers robustness of noise data at the same time. However, for similar pairs, the penalties on data points inside and outside Hamming ball are not discriminative enough. For dissimilar pairs, the penalties on data points inside Hamming ball should be both robust and discriminative, but MMHH only considers robustness.

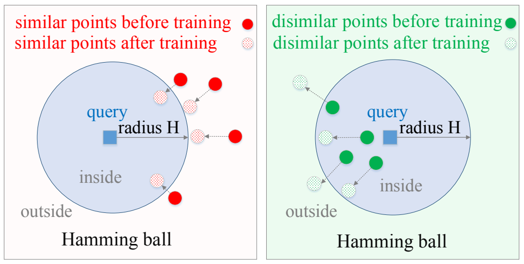
similar points before training ●
similar points after training
query
radius H
inside
outside
Hamming ball
disimilar points before training ●
disimilar points after training
query
radius H
inside
outside
Hamming ball

Fig. 1. The training procedure of MMHH.

To solve these problems, we propose a novel Deep Piecewise Hashing for Efficient Hamming Space Retrieval called DPH, where a piecewise probability distribution is introduced into the loss function and can be optimized in a unified framework. Specifically, for similar data pairs, DPH makes a significant distinction between the probability distribution for the points inside and that for the ones outside the Hamming ball. This can attract more similar pairs into the Hamming ball. Meanwhile, the penalties on dissimilar data pairs inside the Hamming ball are different according to their distances to the query point. The proposed scheme considers both robustness and discrimination for dissimilar data pairs.

*Corresponding authors.

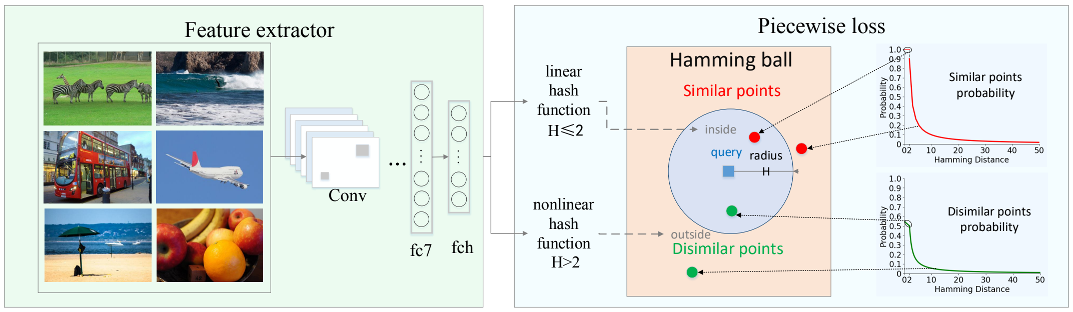

Fig. 2. The framework of DPH. The entire framework consists of two parts: feature extraction and piecewise loss.

The main contributions of our paper are outlined as follows: (1) We design a Hamming space retrieval framework with a piecewise loss function, which can be learned in a unified framework and can be optimized simultaneously. (2) We carefully design a piecewise loss function. It pays more attention to the "marginal" similar points and considers both discrimination and robustness for the dissimilar points inside the Hamming ball. (3) Experiments on two real datasets show that DPH can outperform other baselines and achieve the state-of-the-art performance.

2. DEEP PIECEWISE HASHING FOR EFFICIENT HAMMING SPACE RETRIEVAL

Assume there is an image training set of N points $\{x_i\}_{i=1}^N$. Each point is encoded into compact M-bit binary hash codes $h_i \in \{-1,1\}^M$. In addition, we have a similarity matrix $\mathbf{S}$ written as $\mathbf{S} = [s_{ij}]$. If $s_{ij} = 1$, x_i and x_j are similar, otherwise if $s_{ij} = 0$, they are dissimilar.

The overall Deep Piecewise Hashing for Efficient Hamming Space Retrieval framework is shown in Fig. 2. It mainly consists of two parts, including feature extractor and piecewise loss function.

2.1. Feature Extractor

We use AlexNet [15] as the base network to obtain representative features of images, and replace the classifier layer with a new hash layer. We can adopt equation (1) to generate the M-bit binary hash codes, but it is hard to back-propagate the gradient in the training procedure which is not continuous. Thus, we use the hyperbolic tangent (tanh) function to instead of equation (1) to transform the feature representation of each image x_i into M-dimensional continuous code z_i.

$$\operatorname{sgn}(z_i^m) = \begin{cases} +1, & z_i^m > 0 \\ -1, & z_i^m \le 0 \end{cases}, m \in [1, ..., M] \quad (1)$$

2.2. Piecewise Loss

Similar to Stochastic Neighbor Embedding (SNE) [16], we use a probability to express the similarity relationship between the query point and each image point in Equation (2). As demonstrated in [14], a weighted KL divergence can

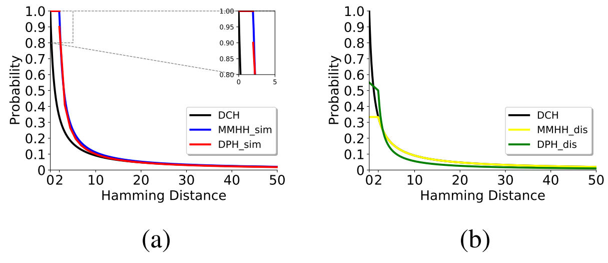

Fig. 3. Probability function on methods of DCH, MMHH and DPH for similar (a) and dissimilar (b) data pairs.

fully capture both similarity and dissimilarity relationships as shown in Equation (2).

$$L = \sum_i \sum_j w_{ij} \cdots \begin{cases} p_{ij} \log \frac{p_{ij}}{q_{ij}}, & s_{ij} = 1 \\ (1 - p_{ij}) \log \frac{1-p_{ij}}{1-q_{ij}}, & s_{ij} = 0 \end{cases} \quad (2)$$

where w_{ij} is the weight for each training data pair (x_i, x_j, s_{ij}), probability p_{ij} denotes the similarity between (x_i, x_j), and probability q_{ij} denotes the similarity between hash codes (h_i, h_j). Since p_{ij} is used to quantify the similarity information of image pairs, we can naturally set $p_{ij} = s_{ij}$ in supervised hashing and drop the constant terms. Then the Equation (2) can be rewritten as follows:

$$L = \sum_{s_{ij} \in S} w_{ij} (s_{ij} \log \frac{1}{q_{ij}} + (1 - s_{ij}) \log \frac{1}{1 - q_{ij}}). \quad (3)$$

Based on Equation (3), probability q_{ij} plays an important role in the hashing model. Fig. 3(a) shows three probability distributions for similar data pairs with Hamming distance $D_H(i,j)$. DCH [13] adopts Cauchy distribution: $q = \frac{1}{1+D_H(i,j)}$ (dark line). MMHH [14] adopts Maximum-Margin t-distribution $\frac{1}{1+max(0, D_H(i,j)-H)}$ (blue line). Our DPH adopts piecewise probability distribution (red line). Fig. 3(b) shows three probability distributions for dissimilar data pairs. DCH also adopts the Cauchy distribution (dark line). MMHH adopts Maximum-Margin t-distribution $\frac{1}{1+max(H, D_H(i,j))}$ (yellow line). Our DPH adopts piecewise probability distribution (green line).

DCH adopts a novel cross-entropy loss based on the Cauchy distribution. It takes a large penalization to similar pairs, and the points outside the Hamming ball are ignored. Then, MMHH directly embodies the Hamming radius into the loss functions to increase the penalization of the points outside the Hamming ball. However, the probability distribution of MMHH is continuous for similar data pairs. Thus, it is easy for the data points that should have been inside near the Hamming ball to be misclassified outside the Hamming ball. In addition, the value of probability is constant for dissimilar data, which has no similarity difference among the noisy data in the Hamming ball. To address these disadvantages, a novel probability distribution based on piecewise function is proposed. It not only solves the problem of data misclassification around Hamming ball, but also keeps the model's discrimination and robustness.

As mentioned above, taking Hamming radius as the boundary, the probability function is as Equation (4).

$$q_{ij} = \begin{cases} 1, & s_{ij}=1,\ H \le 2 \\ \frac{\alpha_1 r}{r+(D_H(i,j)-H)}, & s_{ij}=1,\ H>2 \\ kD_H(i,j)+b, & s_{ij}=0,\ H \le 2 \\ \frac{\alpha_2 r}{r+(D_H(i,j)-H)}, & s_{ij}=0,\ H>2 \end{cases} \tag{4}$$

where α_1, r, k, b and α_2 are hyper-parameters.

2.3. Objective Function

By combining the Equation (3) and Equation (4), we can get a piecewise loss function to differentiate the points inside and outside the Hamming ball. The piecewise loss function is designed as follows:

$$L = \begin{cases} \sum_{s_{ij}\in S} w_{ij}\big((1-s_{ij})\log\frac{1}{1-(kD_H(i,j)+b)} \\ +s_{ij}\log 1\big)\ ,\ H \le 2 \\ \\ \sum_{s_{ij}\in S} w_{ij}\big((1-s_{ij})\log\frac{r+(D_H(i,j)-H)}{((1-\alpha_2)r+(D_H(i,j)-H))} \\ +s_{ij}\log\frac{r+(D_H(i,j)-H)}{\alpha_1 r}\big)\ ,\ \ H>2 \end{cases} . \tag{5}$$

In our method, we adopt continuous representations of cosine distance to approximate the Hamming distance $D_H(i,j)$ between a pair of codes (z_i, z_j) to solve the discrete optimization problem as many previous works [13].

To control the quantization error of converting the continuous representations to binary codes, we further introduce a standard quantization loss as follows:

$$Q = \sum_{i-1}^{n} \parallel \mathrm{sgn}(z_i) - z_i \parallel_2^2 . \tag{6}$$

Combining Equations (5) and (6), we obtain our objective function as follows:

$$min_{\Theta} L + \lambda Q \tag{7}$$

where L is from Equation (5), Q is from Equation (6), Θ represents the network parameters, and λ is a hyper-parameter.

3. EXPERIMENTS

3.1. Datasets

In the experiments, we conduct image hashing on two widely-used datasets: MS-COCO [17] and NUS-WIDE [18]. For MS-COCO, we combine 82,783 training and 40,504 validation images, delete images without category information, and obtain a collection of 122,218 images. NUS-WIDE contains 269,648 images, each of which is annotated by one or more labels of 81 ground truth concepts. Following similar experimental protocols as [13, 14], we randomly sample 5,000 images as the query samples and 10,000 images as the training samples, with the remaining images used as the database.

3.2. Evaluation Protocol, Baselines and Settings

Evaluation Protocol. Following the evaluation protocol in [19, 13, 14], three standard evaluation metrics are used to measure the image retrieval performance, including Mean Average Precision within Hamming radius 2 $\mathbf{MAP@H} \le \mathbf{2}$, Precision within Hamming radius 2 $\mathbf{P@H} \le \mathbf{2}$, and Recall within Hamming radius 2 $\mathbf{R@H} \le \mathbf{2}$ [20].

Baselines. We evaluate the retrieval performance of DPH with eight state-of-the-art methods, including two supervised shallow methods and six supervised deep methods. For shallow methods KSH [21] and SDH [22], we use the 4096-dimensional DeCAF7 [23] features as input. For deep methods CNNH [24], DNNH [19], DHN [25], HashNet [19], DCH [13], and MMHH [14], the input are raw images.

Settings. The implementations are based on Pytorch framework. AlexNet [15] is adopted as the main backbone for all deep hashing methods. In our experiments, learning rate, iteration epoch, batch size are 0.0001, 1000, and 48, respectively.

3.3. Retrieval Performance

$\mathbf{MAP@H} \le \mathbf{2}$. The results of our DPH method and the eight competing methods are reported in Table 1. All methods are evaluated with three different lengths of hash codes (i.e., 32, 48, and 64). Deep hashing methods generally obtain better performance than the shallow hashing methods do. DPH consistently and substantially outperforms all the other methods for comparison over three different hash code lengths. Particularly, DPH outperforms the comparison methods MMHH by 3.1% (32 bits), 4.4% (48 bits) and 3.3% (64 bits) on database MS-COCO and 2.5% (32 bits), 2.9% (48 bits) and 1.8% (64 bits) on database NUS-WIDE.

The main reason why our model can achieve superior performance is that it not only satisfies the differentiation of probability value around Hamming ball for similar data pairs, but also introduces the penalization for noisy data within the Hamming ball for dissimilar data pairs. As expected, the performance of our DPH method is much better than that of baseline methods.

Table 1. The **MAP@H** $\leq$ **2** for different bits of DPH and baselines on MS-COCO and NUS-WIDE image datasets.

Method	MS-COCO			NUS-WIDE		
	32	48	64	32	48	64
KSH	0.513	0.244	0.025	0.563	0.422	0.063
SDH	0.663	0.526	0.513	0.671	0.592	0.453
CNNH	0.562	0.532	0.510	0.590	0.576	0.574
DNNH	0.609	0.522	0.510	0.624	0.592	0.563
DHN	0.663	0.513	0.421	0.710	0.674	0.564
HashNet	0.689	0.562	0.536	0.724	0.680	0.612
DCH	0.755	0.729	0.709	0.776	0.758	0.713
MMHH	0.765	0.797	0.809	0.793	0.801	0.799
DPH	**0.796**	**0.841**	**0.842**	**0.817**	**0.829**	**0.817**

P@H $\leq$ **2**. Fig. 4(a) and Fig. 4(b) show the precision curves with different code lengths of all the methods on datasets MS-COCO and NUS-WIDE. With hash codes length increasing the precision of the most hashing methods decreases sharply, while DPH performs stably in three different code lengths of this paper. Obviously, DPH achieves the best performance on both datasets.

The reason is that when the code length is longer, the Hamming space becomes sparse and fewer data points fall within the Hamming ball. The precision results validate the effectiveness of DPH which enables similar data pairs distribute into the Hamming ball and gets more returned results.

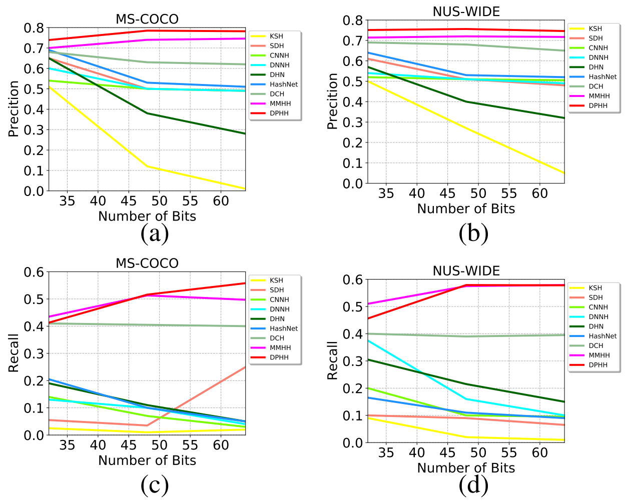

Fig. 4. The **P@H** $\leq$ **2** and **R@H** $\leq$ **2** curves of DPH and comparison methods on MS-COCO and NUS-WIDE with different bits Hash codes.

R@H $\leq$ **2**. Fig. 4(c) and Fig. 4(d) illustrate the performance of Recall in DPH and comparison methods within Hamming Radius 2 (**R@H** $\leq$ **2**) on datasets MS-COCO and NUS-WIDE. It is crucial for Hamming space retrieval, since all data points will be pruned out due to the highly sparse Hamming space. In our DPH method, with the increasing of code length, the value of recall increases sharply and achieves the highest results on MS-COCO and NUS-WIDE.

Such impressive and competitive recall performance proves that DPH can concentrate more relevant points within the Hamming ball than all the other methods.

3.4. Sensitivity to Parameters

We further analyze the effect of hyper-parameters k, b, γ, α_1. The sensitivity of hyper-parameter α_2 is not considered in our experiments, since the value of hyper-parameter α_2 can be directly got by equation $\alpha_2 = 2k+b$. In addition, experiments of parameters sensitivity are on MS-COCO dataset and the code length is 48 bits. Fig. 5 shows the MAP, Precision and Recall results on MS-COCO dataset with different values of k, b, γ, α_1. As shown in Fig. 5, we choose the parameter value $k = 0.05$,$b = 0.55$,$\gamma = 10$,$\alpha_1 = 0.95$ to ensure good performance results.

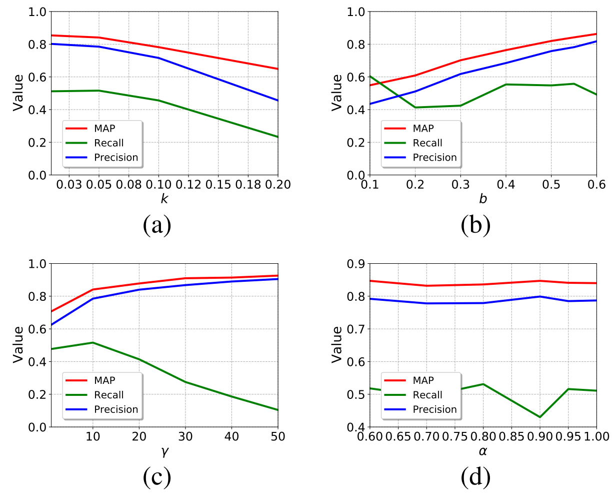

Fig. 5. The MAP, Precision and Recall curves of DPH with different values of k, b,γ, α_1.

4. CONCLUSIONS

In this paper, we propose a novel Deep Piecewise Hashing for Efficient Hamming Space Retrieval (DPH). DPH makes the wrong "marginal" similar points be allocated inside Hamming ball to improve the retrieval performance. For dissimilar data, the penalization inside the Hamming ball varies. Different dissimilar points inside the Hamming ball have different penalties. Experiments on two widely-used datasets show that our DPH can achieve the state-of-the-art Hamming space retrieval performance.

5. ACKNOWLEDGMENTS

This work was funded by National Natural Science Foundation of China (Grant No.62106258).

6. REFERENCES

[1] Zhaolong Zhang, Yuejie Zhang, Rui Feng, Tao Zhang, and Weiguo Fan, "Zero-shot sketch-based image retrieval via graph convolution network," in *AAAI*, 2020, pp. 12943–12950.

[2] Mehrdad Hosseinzadeh and Yang Wang, "Composed query image retrieval using locally bounded features," in *CVPR*, 2020, pp. 3593–3602.

[3] Yining Lang, Yuan He, Fan Yang, Jianfeng Dong, and Hui Xue, "Which is plagiarism: Fashion image retrieval based on regional representation for design protection," in *CVPR*, 2020, pp. 2592–2601.

[4] Yanzhao Xie, Yu Liu, Yangtao Wang, Lianli Gao, Peng Wang, and Ke Zhou, "Label-attended hashing for multi-label image retrieval," in *IJCAI*, 2020, pp. 955–962.

[5] Kaiyue Pang, Yongxin Yang, Timothy M. Hospedales, Tao Xiang, and Yi-Zhe Song, "Solving mixed-modal jigsaw puzzle for fine-grained sketch-based image retrieval," in *CVPR*, 2020, pp. 10344–10352.

[6] Xunguang Wang, Zheng Zhang, Baoyuan Wu, Fumin Shen, and Guangming Lu, "Prototype-supervised adversarial network for targeted attack of deep hashing," in *CVPR*, 2021, pp. 16357–16366.

[7] Yangtao Wang, Jingkuan Song, Ke Zhou, and Yu Liu, "Unsupervised deep hashing with node representation for image retrieval," *Pattern Recognit.*, vol. 112, pp. 107785, 2021.

[8] Yu-Wei Zhan, Xin Luo, Yongxin Wang, and Xin-Shun Xu, "Supervised hierarchical deep hashing for cross-modal retrieval," in *ACM MM*, pp. 3386–3394.

[9] Pedro Morgado, Yunsheng Li, José Costa Pereira, Mohammad J. Saberian, and Nuno Vasconcelos, "Deep hashing with hash-consistent large margin proxy embeddings," *IJCV*, vol. 129, no. 2, pp. 419–438, 2021.

[10] Sheng Jin, Shangchen Zhou, Yao Liu, Chao Chen, Xiaoshuai Sun, Hongxun Yao, and Xian-Sheng Hua, "SSAH: semi-supervised adversarial deep hashing with self-paced hard sample generation," in *AAAI*, 2020, pp. 11157–11164.

[11] Xunguang Wang, Zheng Zhang, Guangming Lu, and Yong Xu, "Targeted attack and defense for deep hashing," in *SIGIR*, 2021, pp. 2298–2302.

[12] Jiawang Bai, Bin Chen, Yiming Li, Dongxian Wu, Weiwei Guo, Shu-Tao Xia, and En-Hui Yang, "Targeted attack for deep hashing based retrieval," in *ECCV*, 2020, vol. 12346, pp. 618–634.

[13] Yue Cao, Mingsheng Long, Bin Liu, and Jianmin Wang, "Deep cauchy hashing for hamming space retrieval," in *CVPR*, 2018, pp. 1229–1237.

[14] Rong Kang, Yue Cao, Mingsheng Long, Jianmin Wang, and Philip S. Yu, "Maximum-margin hamming hashing," in *ICCV*, 2019, pp. 8251–8260.

[15] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton, "Imagenet classification with deep convolutional neural networks," in *NIPS*, 2012, pp. 1106–1114.

[16] Geoffrey E. Hinton and Sam T. Roweis, "Stochastic neighbor embedding," in *NIPS*, 2002, pp. 833–840.

[17] Tsung-Yi Lin, Michael Maire, Serge J. Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C. Lawrence Zitnick, "Microsoft COCO: common objects in context," in *ECCV*, 2014, pp. 740–755.

[18] Tat-Seng Chua, Jinhui Tang, Richang Hong, Haojie Li, Zhiping Luo, and Yantao Zheng, "NUS-WIDE: a real-world web image database from national university of singapore," in *CIVR*, 2009.

[19] Hanjiang Lai, Yan Pan, Ye Liu, and Shuicheng Yan, "Simultaneous feature learning and hash coding with deep neural networks," in *CVPR*, 2015, pp. 3270–3278.

[20] Xiao Bai, Cheng Yan, Haichuan Yang, Lu Bai, Jun Zhou, and Edwin Robert Hancock, "Adaptive hash retrieval with kernel based similarity," *Pattern Recognit.*, vol. 75, pp. 136–148, 2018.

[21] Wei Liu, Jun Wang, Rongrong Ji, Yu-Gang Jiang, and Shih-Fu Chang, "Supervised hashing with kernels," in *CVPR*, 2012, pp. 2074–2081.

[22] Fumin Shen, Chunhua Shen, Wei Liu, and Heng Tao Shen, "Supervised discrete hashing," in *CVPR*, 2015, pp. 37–45.

[23] Jeff Donahue, Yangqing Jia, Oriol Vinyals, Judy Hoffman, Ning Zhang, Eric Tzeng, and Trevor Darrell, "Decaf: A deep convolutional activation feature for generic visual recognition," in *ICML*, 2014, vol. 32, pp. 647–655.

[24] Rongkai Xia, Yan Pan, Hanjiang Lai, Cong Liu, and Shuicheng Yan, "Supervised hashing for image retrieval via image representation learning," in *AAAI*, 2014, pp. 2156–2162.

[25] Han Zhu, Mingsheng Long, Jianmin Wang, and Yue Cao, "Deep hashing network for efficient similarity retrieval," in *AAAI*, 2016, pp. 2415–2421.