This article has been accepted for inclusion in a future issue of this journal. Content is final as presented, with the exception of pagination.

Relaxed Asymmetric Deep Hashing Learning: Point-to-Angle Matching

Jinxing Li, Bob Zhang, *Member, IEEE*, Guangming Lu, Jane You, Yong Xu, *Senior Member, IEEE*, Feng Wu, *Fellow, IEEE*, and David Zhang, *Fellow, IEEE*

***Abstract*—Due to the powerful capability of the data representation, deep learning has achieved a remarkable performance in supervised hash function learning. However, most of the existing hashing methods focus on point-to-point matching that is too strict and unnecessary. In this article, we propose a novel deep supervised hashing method by relaxing the matching between each pair of instances to a point-to-angle way. Specifically, an inner product is introduced to asymmetrically measure the similarity and dissimilarity between the real-valued output and the binary code. Different from existing methods that strictly enforce each element in the real-valued output to be either +1 or −1, we only encourage the output to be close to its corresponding semantic-related binary code under the cross-angle. This asymmetric product not only projects both the real-valued output and the binary code into the same Hamming space but also relaxes the output with wider choices. To further exploit the semantic affinity, we propose a novel Hamming-distance-based triplet loss, efficiently making a ranking for the positive and negative pairs. An algorithm is then designed to alternatively achieve optimal deep features and binary codes. Experiments on four real-world data sets demonstrate the effectiveness and superiority of our approach to the state of the art.**

Manuscript received March 12, 2019; revised August 20, 2019 and November 18, 2019; accepted November 28, 2019. This work was supported in part by the China Postdoctoral Science Foundation under Grant 2019TQ0316 and Grant 2019M662198, in part by the NSFC fund under Grant 61906162, Grant 61773080, and Grant 61602540, in part by the Shenzhen Municipal Science and Technology Innovation Council under Grant GJHZ20180419190732022, in part by the Shenzhen Fundamental Research fund under Grant JCYJ20180306172023949 and Grant JCYJ20170811155442454, in part by the Medical Biometrics Perception and Analysis Engineering Laboratory (Shenzhen), in part by the Shenzhen Research Institute of Big Data, Shenzhen Institute of Artificial Intelligence and Robotics for Society, in part by the Fundamental Research Funds for the Central Universities under Grant 2019CDYGZD001, and in part by the Scientific Reserve Talent Programs of Chongqing University under Grant cqu2018CDHB1B04. *(Corresponding author: David Zhang.)*

J. Li is with the School of Science and Engineering, The Chinese University of Hong Kong (Shenzhen), Shenzhen 518172, China, and the University of Science and Technology of China, Hefei 230000, China (e-mail: lijinxing@cuhk.edu.cn).

B. Zhang is with the Department of Computer and Information Science, University of Macau, Macau 999078, China (e-mail: bobzhang@umac.mo).

G. Lu and Y. Xu are with the Department of Computer Science, Harbin Institute of Technology Shenzhen Graduate School, Shenzhen 518000, China (e-mail: luguangm@hit.edu.cn; yongxu@ymail.com).

J. You is with the Department of Computing, The Hong Kong Polytechnic University, Hong Kong (e-mail: csyjia@comp.polyu.edu.hk).

F. Wu is with the CAS Key Laboratory of Technology in Geo-Spatial Information Processing and Application System, University of Science and Technology of China, Hefei 230000, China (e-mail: fengwu@ustc.edu.cn).

D. Zhang is with the School of Science and Engineering, The Chinese University of Hong Kong (Shenzhen), Shenzhen 518172, China (e-mail: davidzhang@cuhk.edu.cn).

Color versions of one or more of the figures in this article are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TNNLS.2019.2958061

***Index Terms*—Asymmetric, deep learning, hashing learning, point-to-angle, triplet loss.**

I. Introduction

Due to the rapid development of the internet, multimedia data in search engines and social networks meets a great increase in recent years. Subsequently, how to store these data and allow searches to have a quick query when a test sample is given has become a fundamental problem. Fortunately, hashing techniques provide a reasonable solution, thanks to its low storage cost (a few binary bits per sample) and fast retrieval speed (low complexity of the Hamming distance computation). The main goal of hashing methods is to learn multiple hash functions to project the input to a more compact Hamming space, in which the feature is represented as binary codes. Since these codes further enjoy the semantic or structure information existing in the original space, hashing methods have been used in many applications with a large-scale data set, e.g., image retrieval [1]–[3], pattern recognition [4]–[6], and data fusion [7].

Generally, hashing methods can be classified into two categories: data-independent and data-dependent. A typical data-independent method is the locality sensitive hashing (LSH) [8]. Although LSH is quite simple and easy to be implemented, it cannot meet our requirements when the length of the binary codes is relatively small due to the random hashing functions that are too weak to capture the complex distribution of the input data. To address this problem, data-dependent methods aim to learn adaptive hashing functions based on the input data, achieving much better performances with fewer binary bits per sample compared with LSH. Anchor graph hashing (AGH) [9] and spectral hashing (SH) [10] introduce the graph-based hashing methods to automatically discover the neighborhood structure inherent in the data to learn appropriate compact codes. However, these two methods discard the discrete constraints by solving the continuous problems. Therefore, discrete graph hashing (DGH) [11] further presents a tractable alternating maximization algorithm to deal with the discrete constraints. In addition, instead of using hyperplane-based hashing functions, spherical hashing [12] proposes a novel hypersphere-based hashing function, being capable of mapping more spatially coherent data points into a binary code. Despite the fact that the aforementioned methods learn the data-driven based hashing functions, the supervised information that is valuable for the performance improvement

This article has been accepted for inclusion in a future issue of this journal. Content is final as presented, with the exception of pagination.

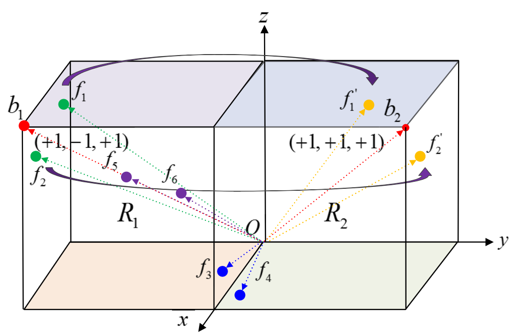


Fig. 1. Motivation of the point-to-angle matching. Assume $\mathbf{f}_1$, $\mathbf{f}_2$, $\mathbf{f}_3$, $\mathbf{f}_4$, $\mathbf{f}_5$, $\mathbf{f}_6$, and $\mathbf{b}_1$ belong to the same Hamming space $\mathbf{R}_1$. 1) $(\mathbf{f}_1^T\mathbf{f}_2 - k)^2$ makes them similar and discrete. However, if $\mathbf{f}_1$ and $\mathbf{f}_2$ are both transformed to $\mathbf{f}_1'$ and $\mathbf{f}_2'$ in $\mathbf{R}_2$, the value of $\mathbf{f}_1^T\mathbf{f}_2$ remains unchanged but their hashing codes change. 2) The asymmetric inner product $(\mathbf{f}_1^T\mathbf{b}_1 - k)^2$ can well tackle the problem in 1), but this term strongly enforces each element in $\mathbf{f}_1$ ($\mathbf{f}_2$) to approximate to be either +1 or −1. This is too strict and unnecessary since what we need is $\text{sign}(\mathbf{f}_1)$. 3) To address the problem in 2), although some works try to use the cosine distance to measure the similarity, it may be unable to process some specific cases like $(\mathbf{f}_3, \mathbf{f}_4)$.

is ignored. To tackle this problem, fast supervised hashing (FastH) [13] and supervised discrete hashing (SDH) [14] simultaneously project the original features to compact binary codes and preserve the label-based similarity. Although these traditional data-dependent hashing methods are superior to data-independent approaches, two separated steps (feature extraction and hashing learning) limit their applications to large-scale data sets, which have complex distributions. Fortunately, deep learning [15], [16] provides a reasonable way to jointly train the features and hashing functions in an end-to-end structure. For instance, deep pairwise supervised hashing (DPSH) [17] introduces the deep network and a pairwise loss to extract deep features and learn the hashing function, achieving a much better performance on image retrieval. More learning to hash algorithms including both shallow and deep architectures can be found in [18]. In this article, we aim to apply our deep hashing method to image retrieval.

However, existing studies on deep hashing methods still encounter several problems as follows.

1) The first limitation is that most deep hashing approaches aim to match each pair of samples through point-to-point. Assume that $\mathbf{f}_i \in \mathbb{R}^{k\times 1}$ and $\mathbf{f}_j \in \mathbb{R}^{k\times 1}$ are outputs corresponding to the ith and jth samples, where k is the dimension of the output. A common strategy in hashing learning is to minimize their Hamming distance through $(\mathbf{f}_i^T\mathbf{f}_j - kS_{ij})^2$, where S_{ij} is the element of the similarity matrix in the ith row and jth column. Although this term approximates $(\mathbf{f}_i, \mathbf{f}_j)$ to be close to the discrete values (either +1 or −1) and exploits their pairwise semantic information, it fails in some specific cases. As shown in Fig. 1, suppose $\mathbf{f}_1$ and $\mathbf{f}_2$ have the same ground-truth and are located in the same Hamming space $\mathbf{R}_1$. However, if both points are transformed to $\mathbf{f}_1'$ and $\mathbf{f}_2'$ in another Hamming space $\mathbf{R}_2$, there is no change for the value of $\mathbf{f}_1'^T\mathbf{f}_2'$, while their corresponding hashing codes $\text{sign}(\mathbf{f}_1')$ and $\text{sign}(\mathbf{f}_2')$ are obviously different from their ground-truth $\text{sign}(\mathbf{f}_1)$ and $\text{sign}(\mathbf{f}_2)$. To address this problem, several studies [19] try to use the asymmetric loss $(\mathbf{f}_2^T\mathbf{b}_1 - kS_{21})^2$ to alternatively update binary code $\mathbf{b}_1$ and real-valued output $\mathbf{f}_2$. Experimental results demonstrate the superiority of this asymmetric inner product. Note that what we require are the binary codes $\text{sign}(\mathbf{f}_1)$ and $\text{sign}(\mathbf{f}_2)$. Thus, encouraging $\mathbf{f}_1$, $\mathbf{f}_2$, and $\mathbf{b}_1$ to be in the same Hamming space is necessary, while the traditional asymmetric product $\mathbf{f}_1^T\mathbf{b}_1$ ($\mathbf{f}_2^T\mathbf{b}_1$) not only enforces $\mathbf{f}_1$ ($\mathbf{f}_2$) and $\mathbf{b}_1$ in the same Hamming space but also requires them to be close in Euclidean distance, which is too strict and unnecessary. Therefore, Cao *et al.* [20] and He *et al.* [21] introduced the cosine distance $(\langle\mathbf{f}_i, \mathbf{f}_j\rangle / \|\mathbf{f}_i\|\|\mathbf{f}_j\|)$ to make similar points lie in the same hypercube with high probability. However, this method also cannot meet our requirement in some specific cases. As shown in Fig. 1, assume $(\mathbf{f}_3, \mathbf{f}_4)$ and $(\mathbf{f}_5, \mathbf{f}_6)$ enjoy the same semantic information. We favor $\mathbf{f}_5$ and $\mathbf{f}_6$, but avoid $\mathbf{f}_3$ and $\mathbf{f}_4$. In fact, the cosine distance in both pairs is large, while $\mathbf{f}_3$ and $\mathbf{f}_4$ have different hashing codes. Therefore, based on the aforementioned analysis, we propose a novel relaxed asymmetric strategy to achieve the matching through point-by-angle. Particularly, as shown in Fig. 1, points $\mathbf{f}_5$ and $\mathbf{f}_6$ are encouraged to be close to the binary variable $\mathbf{b}_1$ in cosine distance. Thanks to this strategy, we can not only make samples belonging to the same class in a common hypercube without any length constraint but also efficiently avoid the case occurring between $\mathbf{f}_3$ and $\mathbf{f}_4$.
2) The second problem is how to efficiently exploit the semantic affinity existing in the original space. In recent years, several functions have been proposed to reveal the similarity and dissimilarity between each pair of samples. The two commonest ways are the pairwise loss and triplet loss. The pairwise loss enforces the samples belonging to the same class to be close while those belonging to different classes to be far. In contrast, the triplet loss encourages the distance between each pair of similar (positive) samples to be smaller than that between each pair of dissimilar (negative) samples. In fact, in the image retrieval task, what we need is to make the similar samples closer than other dissimilar samples. Thus, we focus on using the triplet loss to measure the semantic affinity.

The traditional triplet loss uses the Euclidean distance to separate the similar and dissimilar pairs. However, the Euclidean distance is not adaptive for our relaxed asymmetric method. As shown in Fig. 2, $\mathbf{f}_1$ and $\mathbf{f}_2$ belong to the same category and are located in the same Hamming space, while $\mathbf{f}_3$ is dissimilar to them and falls in another Hamming space. In fact, both $\mathbf{f}_1$ and $\mathbf{f}_2$ are in appropriate positions in the relaxed point-to-angle viewpoint since their cosine distances to $\mathbf{b}_1$ are large. However, the Euclidean distance between $\mathbf{f}_1$ and $\mathbf{f}_2$ is much larger than that between $\mathbf{f}_2$ and $\mathbf{f}_3$. Therefore, if the traditional triplet loss is directly used for semantic information exploration, there would be an influence on our proposed relaxed asymmetric strategy. To address this problem, we propose a novel triplet loss based on the Hamming distance. Particularly, we first normalize the output onto a

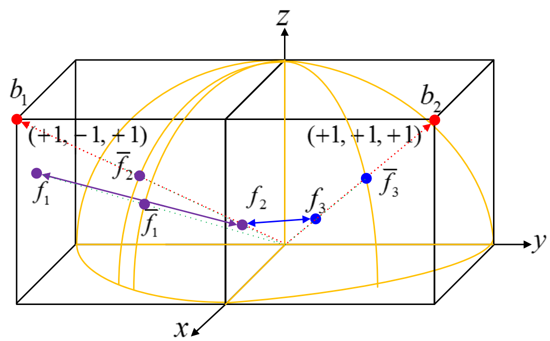


Fig. 2. Motivation of the novel triplet loss. $\mathbf{f}_1$ and $\mathbf{f}_2$ belong to the same class, while $\mathbf{f}_3$ belongs to another class. According to the analysis in Fig. 1, $\mathbf{f}_1$ and $\mathbf{f}_2$ are located in appropriate positions. However, the Euclidean distance between them is much larger than that of $\mathbf{f}_2$ and $\mathbf{f}_3$. The traditional triplet cannot be directly used. Thus, we propose a novel triplet loss. First, each output is normalized onto a unit ball to get $\bar{\mathbf{f}}_1$, $\bar{\mathbf{f}}_2$, and $\bar{\mathbf{f}}_3$ corresponding to $\mathbf{f}_1$, $\mathbf{f}_2$, and $\mathbf{f}_3$, respectively. Then $[1-(\bar{\mathbf{f}}_1^T\bar{\mathbf{f}}_3-1)_2^2+(\bar{\mathbf{f}}_1^T\bar{\mathbf{f}}_2-1)_2^2]_+$ is constructed to encourage $\bar{\mathbf{f}}_1$ to be closer to $\bar{\mathbf{f}}_2$ than to $\bar{\mathbf{f}}_3$ under the Hamming distance.

multidimensional unit ball and get $\bar{\mathbf{f}}_1$, $\bar{\mathbf{f}}_2$, and $\bar{\mathbf{f}}_3$ corresponding to $\mathbf{f}_1$, $\mathbf{f}_2$ and $\mathbf{f}_3$, respectively. Different from traditional methods that only consider the Euclidean distance in the triplet loss, we further use the Hamming distance to encourage $(\bar{\mathbf{f}}_1, \bar{\mathbf{f}}_2)$ to be closer than $(\bar{\mathbf{f}}_1, \bar{\mathbf{f}}_3)$ and $(\bar{\mathbf{f}}_2, \bar{\mathbf{f}}_3)$. Mathematically, the triplet loss among these three points can be represented as $[1-(\bar{\mathbf{f}}_1^T\bar{\mathbf{f}}_3-1)^2+(\bar{\mathbf{f}}_1^T\bar{\mathbf{f}}_2-1)^2]_+$ and $[1-(\bar{\mathbf{f}}_2^T\bar{\mathbf{f}}_3-1)^2+(\bar{\mathbf{f}}_1^T\bar{\mathbf{f}}_2-1)^2]_+$, where $[z]_+ = \max(z, 0)$. We will discuss the details of this in Section III.

The main contributions are concluded as follows.

1) A relaxed asymmetric strategy is proposed to reveal the similarity between real-valued outputs and discrete binary codes in point-to-angle matching. The real-valued features and hashing variables are encouraged to fall in the same Hamming space through an inner product without any length constraint.
2) A novel triplet loss is presented, which is quite adaptive for our proposed relaxed asymmetric method. Different from the traditional version that only ranks the positive and negative pairs by using the Euclidean distance, we normalize each output onto a multidimensional unit ball and the Hamming distance is introduced to make a ranking for different pairs.
3) An efficient algorithm is designed to alternatively update various variables in an end-to-end deep structure. Particularly, the binary codes can be obtained in a discrete way.
4) In image retrieval, experimental results on four large-scale data sets substantiate the effectiveness and superiority of our proposed method compared with some existing state-of-the-art hashing approaches.

The rest of this article is organized as follows. The related works, including both data-independent and data-dependent hashing methods, are briefly reviewed in Section II. In Section III, the proposed relaxed asymmetric deep hashing (RADH) is analyzed, followed by its optimization, inference, and implementation. In Section IV, experiments are conducted on four large-scale data sets to demonstrate the superiority of RADH. This article is finally concluded in Section V.

II. Related Works

As described in the first section, the hashing methods can be roughly divided into data-independent and data-dependent approaches.

LSH [8] is one of the most typical data-independent methods, which aims to use several randomly projections to get the hashing codes, ensuring the probability of collision is much higher for data points that are closer to each other than for those that are far apart. LSH is further extended to a kernel version (KLSH) [22] to nonlinearly represent the real-world data sets with more complex structures. In addition, various distance or similarity priors are also imposed on the basic LSH to achieve several extensions [23]–[25]. However, there is a performance limitation for LSH due to the fact that it is totally data-independent and ignores the data distribution that is valuable for the performance improvement.

To tackle this problem, researchers focus on the data-dependent methods to learn an adaptive hashing function for a specific data set. Generally, data-dependent methods can also be separated into two parts: unsupervised and supervised. Unsupervised hashing methods aim to exploit the structure information to learn compact codes for the input data. Density-sensitive hashing (DSH) [26] was proposed to replace the random projection in LSH. SH was proposed by Weiss *et al.* [10]. SH bridges the binary coding to the graph partitioning. Due to the high complexity of SH when the data set is large, Heo *et al.* proposed spherical hashing based on the hyperplane to learn a spherical Hamming distance [12]. Jiang and Li [27] proposed a scalable graph hashing (SGH) for large-scale graph hashing. Different from SH and SGH that ignore the discrete constraint, the DGH [11] was studied, which can find the inherent neighborhood structure in a discrete code space. In contrast to graph hashing, iterative quantization (ITQ) [28] and double-bit quantization (DBQ) [29] were presented to minimize the quantization error. Instead of measuring data similarity by the Euclidean distances, Hu *et al.* [30] designed a cosine similarity-based hashing learning strategy to achieve better performances. Different from unsupervised hashing methods that ignore the label information in the training set, supervised hashing learning focuses on learning the hash function to encourage the projected hashing codes to preserve the semantic information existing in the original space. Some typical supervised hashing approaches include kernel-based supervised hashing (KSH) [31], FastH [13], SDH [14], and column sample-based discrete supervised hashing (COSDISH) [32]. Both KSH and FastH achieve nonlinearity in supervised hashing, while SDH and COSDISH optimize their models discretely. Considering that there do exist potentially noisily labeled samples, the robust discrete code modeling (RDCM) [33] was presented to employ $l_{2,p}$-norm, being capable of performing code selection and noisy sample identification.

Although many data-dependent methods have been studied, they often meet a performance limitation due to the use of hand-crafted features. In recent years, deep learning with

This article has been accepted for inclusion in a future issue of this journal. Content is final as presented, with the exception of pagination.

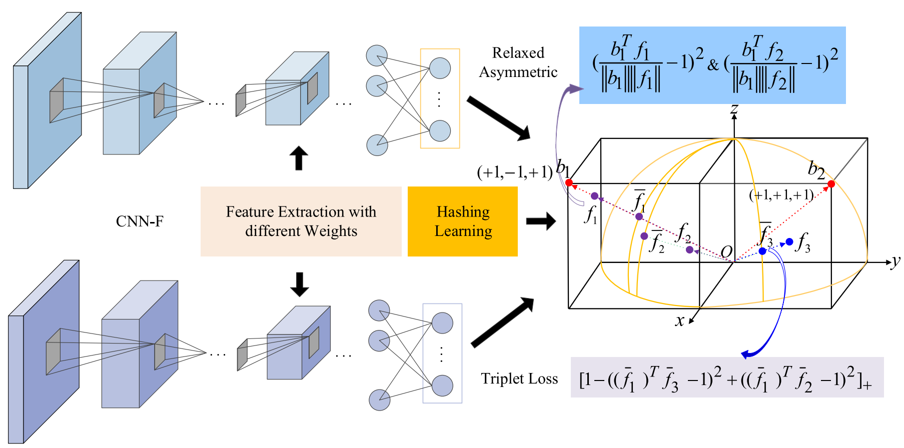


Fig. 3. Framework of the proposed method. The top and bottom streams with different weights are used to extract features from the images. A shared binary code $\mathbf{b}_i$ is then generated from the same input for both streams. Here we assume that ($\mathbf{b}_1$, $\mathbf{f}_1$, $\mathbf{f}_2$) have the same semantic information, while ($\mathbf{b}_2$, $\mathbf{f}_3$) enjoy different semantic information from $\mathbf{b}_1$. In the learning step, the relaxed asymmetric is exploited to make ($\mathbf{b}_1$, $\mathbf{f}_1$, $\mathbf{f}_2$) locate in the same Hamming space without the length constraint. We also propose a novel triplet loss to rank the positive and negative pairs.

an end-to-end network provides a reasonable and promising solution. A deep network was proposed by Liong *et al*. [34] to jointly represent the data and obtain binary codes. Due to the powerful image representation, convolutional neural networks (CNN) are widely applied to hashing learning. For instance, Zhang *et al*. [35] combined the CNN and hashing learning in a unified model and applied it to image retrieval and person re-identification. Similarly, deep hashing network (DHN) [36], deep supervised hashing (DSH-DL) [37], and CNN-based hashing (CNNH) [38] were also proposed. A pairwise loss (DPSH) was presented in [17] to preserve the semantic information between each pair of outputs. Shen *et al.* [39] deep asymmetric pairwise hashing (DAPH) and Jiang and Li asymmetric deep supervised hashing (ADSH) [19] proposed asymmetric structures and experimental results demonstrated their superiority. Note that an asymmetric hashing method, named asymmetric inner-product binary coding (AIBC), was previously proposed in [40] by revealing the inner products between raw data vectors and the discrete vectors, whose objective function is similar to that of ADSH. However, our proposed method is greatly different from these three asymmetric methods. For ADSH and AIBC, they only try to use the point-to-point based inner product to link the relationship between the real-valued output and the discrete variable. By contrast, RADH transforms the point-to-point operation to the point-to-angle way, which can provide more freedom for the estimation of the real-valued outputs. This strategy is more adaptive and reasonable for hashing learning, being beneficial to the performance improvement. Referring to DAPH, it only takes two streams of networks for training, while the Hamming distance-based measurement between the binary variable and the real-valued output is ignored. By contrast, our proposed method RADH not only takes the two-stream network into account but also introduces a novel measurement between the binary variable and the real-valued output.

To exploit the ranking information, the triplet labels are used to learn hashing models, including deep semantic ranking-based hashing (DSRH) [41], deep similarity comparison hashing (DSCH) [35], deep regularized similarity comparison hashing (DRSCH) [35], and unsupervised deep triplet hashing (UDTH) [42]. Note that, although UDTH uses the cosine similarity to construct the pseudo, it still exploits the Euclidean distance to measure the similarity and dissimilarity. Since there is not any length constraint on the learned features in RADH, these existing triplet losses are unsuitable to be directly applied to RADH. By contrast, our proposed triplet loss not only normalizes each learned feature to a unit length first but also transforms the Euclidean distance to the relaxed Hamming distance, which is more suitable for hashing learning. In addition, Heo *et al*. [12], [43] proposed a hypersphere-based hashing method, in which the pivot positions of two hyperspheres are made closer if the number of data points in a subset corresponding to two hashing functions are smaller or equal to a threshold. Otherwise, the pivots would be placed farther away. By contrast, our proposed method exploits the relaxed Hamming distance to encourage the distance between the positive pair to be smaller than that between the negative pair, which introduces the ranking information, being quite different from the measurement in [12] and [43]. Also, Gordo *et al*. [44] proposed the asymmetric schemes to binarize the database signatures but not the query. Differently, our presented approach exploits the supervised information between the binary codes and real-valued features, as well as the positive pairs and negative pairs.

III. Proposed Method

In this section, we first define the deep structure and some notations used in this article. The presented RADH is then analyzed, followed by its optimization, inference, and implementation.

A. Network Structure and Notations

In [39], it has been proved that the asymmetric deep network can preserve more similarity information. Note that in this article, we mainly focus on the efficiency of the relaxed asymmetric and novel triplet losses. Thus, we only simply apply the CNN-F [45] and the asymmetric deep network [39] to represent the input data, as shown in Fig. 3. The main advantage of this two-stream network is that we can regard the input in one stream as the query and view the other one as the database, which is beneficial for updating the weights in each stream in a supervised way. It is obvious that this deep structure can be replaced with other network structures, e.g., VGG and ResNet, whereas these different structures are not the focus of this article.

Since we prefer to obtain a k-bit binary code, the last layer in CNN-F is replaced with a k-D vector. Besides this, we initialize the weights of the first seven layers in CNN-F by using the pretrained ImageNet model, where the weights in the last layer are set to be random values. Here we denote the inputs in the first and second streams as $\mathbf{X} = \{\mathbf{x}_1, \ldots, \mathbf{x}_i, \ldots, \mathbf{x}_N\} \in \mathbb{R}^{N \times d_1 \times d_2 \times 3}$ and $\mathbf{Y} = \{\mathbf{y}_1, \ldots, \mathbf{y}_i, \ldots, \mathbf{y}_N\} \in \mathbb{R}^{N \times d_1 \times d_2 \times 3}$, where N is the number of training samples and (d_1, d_2) are the size of an image. Note that $\mathbf{X}$ and $\mathbf{Y}$ are only different in symbol notations. They both represent the same training set. Being similar to [39], the purpose of our model is to learn two hash functions $\mathcal{F}$ and $\mathcal{G}$ to map the raw data $\mathbf{X}$ and $\mathbf{Y}$ into the Hamming space. In this article, we denote the outputs associated with $\mathbf{X}$ and $\mathbf{Y}$ as $\mathbf{F} = [\mathbf{f}_1, \ldots, \mathbf{f}_i, \ldots, \mathbf{f}_N]^T \in \mathbb{R}^{N \times k}$ and $\mathbf{G} = [\mathbf{g}_1, \ldots, \mathbf{g}_i, \ldots, \mathbf{g}_N]^T \in \mathbb{R}^{N \times k}$, respectively. In addition, their corresponding shared binary codes are denoted as $\mathbf{B} = [\mathbf{b}_1, \ldots, \mathbf{b}_i, \ldots, \mathbf{b}_N]^T \in \mathbb{R}^{N \times k}$, where $\mathbf{b}_i \in \{-1, +1\}^{k \times 1}$. As our method performs supervised learning, a matrix $\mathbf{S} \in \mathbb{R}^{N \times N}$ is introduced to measure the semantic similarity between $\mathbf{X}$, $\mathbf{Y}$, and $\mathbf{B}$. S_{ij} is its element in the ith row and jth column. $S_{ij} = 1$ if $\mathbf{x}_i$ and $\mathbf{y}_j$ ($\mathbf{b}_j$) share the same semantic information or label, otherwise $S_{ij} = 0 - \varepsilon$, where ε is a slack variable, e.g., 0.11. Note that, the sizes of $\mathbf{F}$, $\mathbf{G}$, and $\mathbf{B}$ are all the same. In our two-stream network, being the same to DAPH, we input the images $\mathbf{X}$ and $\mathbf{Y}$ (both are the same training set) into the first and second streams alternatively. When the images are inputted into the first stream and get the output $\mathbf{F}$, we encourage $\mathbf{F}$ to do retrieval from $\mathbf{G}$. The second stream carries out the same operation. Then $\mathbf{B}$ is the binary code that is completely associated with $\mathbf{F}$ and $\mathbf{G}$ row by row.

B. RADH

The framework of the proposed method is shown in Fig. 3. There are two streams and both are used to extract the features from the input images. Different from DAPH [39] that learns two binary codes associated with these two streams, we aim to learn a shared hashing code associated with the outputs from the first and second streams. More specifically, a relaxed asymmetric strategy is used to encourage the real-valued outputs and the discrete hashing variables enjoying the same semantic information to be close in terms of the cosine distance. The novel triplet loss is then introduced to make an efficient ranking for each positive and negative pair.

Let $\mathbf{F}$ and $\mathbf{G}$ be the outputs in the first and second streams, respectively. Since our goal is to obtain hash functions through the deep networks, the binary code $\mathbf{B}$ is also generated by minimizing the distance between $\mathbf{B}$ and $\mathbf{F}$ / $\mathbf{G}$. As analyzed in DAPH and ADSH, the asymmetric strategy can reduce the difficulty of the discrete constraint optimization. Equation (1) is used to not only make the real-valued outputs and the hashing variables close in Hamming distance but also enjoy the semantic affinity existing in the original space

$$\min \left(\mathbf{b}_i^T \mathbf{f}_j - k S_{ij}\right)_2^2 + \left(\mathbf{b}_i^T \mathbf{g}_j - k S_{ij}\right)_2^2. \tag{1}$$

However, (1) also enforces $\mathbf{F}$ or $\mathbf{G}$ to be similar in the length, which is too strict and unnecessary. What we need is to ensure the hashing codes and real-valued outputs to have the same semantic information in a common Hamming space rather than the same length. To address this problem, we relax (1) to (2).

$$\min \left(\frac{\mathbf{b}_i^T \mathbf{f}_j}{\|\mathbf{b}_i\| \|\mathbf{f}_j\|} - S_{ij} \right)^2 + \left(\frac{\mathbf{b}_i^T \mathbf{g}_j}{\|\mathbf{b}_i\| \|\mathbf{g}_j\|} - S_{ij} \right)^2 \tag{2}$$

where $\|\mathbf{b}_i\|$, $\|\mathbf{f}_j\|$, and $\|\mathbf{g}_j\|$ represent the length of $\mathbf{b}_i$, $\mathbf{f}_j$, and $\mathbf{g}_j$, respectively. Since $\|\mathbf{b}_i\| = \sqrt{k}$, we further transform (2) to (3)

$$\min L_a = \sum_{i,j} \left(\left(\frac{\mathbf{b}_i^T \mathbf{f}_j}{\|\mathbf{f}_j\|} - \sqrt{k} S_{ij} \right)^2 + \left(\frac{\mathbf{b}_i^T \mathbf{g}_j}{\|\mathbf{g}_j\|} - \sqrt{k} S_{ij} \right)^2 \right). \tag{3}$$

From (3), it is easy to see that there is not any length constraint on the learned features, while this equation simultaneously makes the real-valued outputs and binary codes locate in the same hypercube if they belong to the same class, otherwise they will be located in different Hamming spaces. Compared with (1), this strategy provides more freedom for the estimation of $\mathbf{f}_j$ and $\mathbf{g}_j$, which is more adaptive and reasonable for hashing learning.

In the retrieval task, what we need is to ensure the distance between each positive pair to be smaller than that of the corresponding negative pair. Thus, we further introduce the triplet loss, which is quite adaptive for the retrieval task. However, directly applying the existing triplet loss to RADH is not reasonable due to the following two limitations.

1) In hashing learning, the input vectors for the existing triplet loss often follow the constraint that each element is encouraged to be $+1$ or -1, so that their lengths would be limited to a narrow range. However, there is not any length constraint on the vectors in RADH, which makes the existing triplet loss unsuitable for our proposed method, as shown in Fig. 2.
2) The adaptive measurement in hashing learning is the Hamming distance, while the existing triplet loss often adopts the Euclidean distance to measure the similarity or dissimilarity, subsequently resulting in the quantative error.

By contrast, in this article, we propose a novel triplet loss that is quite suitable for RADH. Particularly, we first normalize the real-valued features to a unit ball so that the vector length problem can be well avoided. The inner product of two outputting vectors is then introduced as the relaxed Hamming distance, which would be more adaptive and reasonable for our hashing learning.

Our proposed triplet comparison formulation is trained on a series of triplets $(\mathbf{f}_i, \mathbf{f}_j, \mathbf{f}_t)$, where $\mathbf{f}_i$ and $\mathbf{f}_j$ belong to the same class, while $\mathbf{f}_j$ and $\mathbf{f}_t$ are from different classes. The same formulation can be applied to $(\mathbf{g}_i, \mathbf{g}_j, \mathbf{g}_t)$. To make the Hamming distance between $\mathbf{f}_i$ and $\mathbf{f}_j$ smaller than that between $\mathbf{f}_j$ and $\mathbf{f}_t$, for any triplet $(\mathbf{f}_i, \mathbf{f}_j, \mathbf{f}_t)$, these three samples should satisfy the following constraint:

$$\left(\frac{\mathbf{f}_t^T\mathbf{f}_j}{\|\mathbf{f}_t\|\|\mathbf{f}_j\|}-1\right)^2-\left(\frac{\mathbf{f}_i^T\mathbf{f}_j}{\|\mathbf{f}_i\|\|\mathbf{f}_j\|}-1\right)^2 > 1-\xi_{ijt} \tag{4}$$

where ξ_{ijt} is a nonnegative slack variable. Note that since there is not any constraint on the length of the output, we first normalize $(\mathbf{f}_i, \mathbf{f}_j, \mathbf{f}_t)$ onto a k-dimensional unit ball. However, it is difficult to get the gradient of $\mathbf{f}_j$ since it may also be associated with other samples as a retrieval point. Thanks to our two-stream network structure, we can regard $\mathbf{f}_j$ as a query, while $\{\mathbf{g}_i\}_{i=1}^N$ is used as a retrieval data set. Thus, jointly taking $(\mathbf{f}_i, \mathbf{f}_j, \mathbf{f}_t)$ and $(\mathbf{g}_i, \mathbf{g}_j, \mathbf{g}_t)$ into account, the triplet loss function in RADH is

$$\begin{aligned}\min L_p &= \sum_{i,j,t}\left[1-\left(\frac{\mathbf{g}_t^T\mathbf{f}_j}{\|\mathbf{g}_t\|\|\mathbf{f}_j\|}-1\right)^2+\left(\frac{\mathbf{g}_i^T\mathbf{f}_j}{\|\mathbf{g}_i\|\|\mathbf{f}_j\|}-1\right)^2\right]_+\\ &\quad+\sum_{i,j,t}\left[1-\left(\frac{\mathbf{f}_t^T\mathbf{g}_j}{\|\mathbf{f}_t\|\|\mathbf{g}_j\|}-1\right)^2+\left(\frac{\mathbf{f}_i^T\mathbf{g}_j}{\|\mathbf{f}_i\|\|\mathbf{g}_j\|}-1\right)^2\right]_+\end{aligned} \tag{5}$$

where $[z]_+ = \max(z, 0)$. Compared with the existing triplet loss that only uses the Euclidean distance to measure the ranking between the positive and negative pairs, (5) successfully transforms it to a hashing-based function, efficiently exploiting the Hamming distance to achieve a satisfactory ranking.

As described in [39], an additional Euclidean regularization between $\mathbf{b}_j$ and $\mathbf{f}_j$ / $\mathbf{g}_j$ is beneficial to the performance improvement. Thus, we further enforce $\mathbf{b}_j$ and $\mathbf{f}_j$ / $\mathbf{g}_j$ to be close in terms of the Euclidean distance as shown in the following equation:

$$\min L_e = \sum_j\left(\left\|\sqrt{k}\frac{\mathbf{f}_j}{\|\mathbf{f}_j\|}-\mathbf{b}_j\right\|_2^2+\left\|\sqrt{k}\frac{\mathbf{g}_j}{\|\mathbf{g}_j\|}-\mathbf{b}_j\right\|_2^2\right). \tag{6}$$

Here $\sqrt{k}(\mathbf{f}_j/\|\mathbf{f}_j\|)$ and $\sqrt{k}(\mathbf{g}_j/\|\mathbf{g}_j\|)$ are applied to make their elements close to either -1 or $+1$.

In addition, as shown in (7), in order to achieve a balance for each bit in the training samples, as well as to maximize the information provided by each bit, another regularization is introduced

$$\min L_r = \|\sqrt{k}\bar{\mathbf{F}}^T\mathbf{1}\|_2^2+\|\sqrt{k}\bar{\mathbf{G}}^T\mathbf{1}\|_2^2 \tag{7}$$

where $\bar{\mathbf{F}} = [\bar{\mathbf{f}}_1, \ldots, \bar{\mathbf{f}}_N]^T = [(\mathbf{f}_1/\|\mathbf{f}_1\|), \ldots, (\mathbf{f}_N/\|\mathbf{f}_N\|)]^T$, $\bar{\mathbf{G}} = [\bar{\mathbf{g}}_1, \ldots, \bar{\mathbf{g}}_N]^T = [(\mathbf{g}_1/\|\mathbf{g}_1\|), \ldots, (\mathbf{g}_N/\|\mathbf{g}_N\|)]^T$ and $\mathbf{1}$ is a $N \times 1$ vector whose elements are all 1.

Overall, the objective function can be obtained as follows:

$$\min L_a + \tau L_p + \gamma L_e + \eta L_r, \quad \text{s.t. } \mathbf{b}_i \in \{-1, +1\} \tag{8}$$

where τ, γ, and η are the nonnegative parameters to make a tradeoff among various terms.

C. Optimization

In the objective function (8), the variables including $\mathbf{F}$, $\mathbf{G}$ and $\mathbf{B}$ should be optimized. Due to the discrete constraint on $\mathbf{B}$, it is difficult to get the optimal solution directly. In this article, an efficient algorithm is designed to alternatively and discretely update different variables.

1) Update $\mathbf{F}$ With $\mathbf{G}$ and $\mathbf{B}$ Fixed: By fixing $\mathbf{G}$ and $\mathbf{B}$, the objective function (8) is transformed to

$$\begin{aligned}\min L_f &= \sum_{i,j}\left(\mathbf{b}_i^T\bar{\mathbf{f}}_j-\sqrt{k}S_{ij}\right)^2+\gamma\sum_j\|\sqrt{k}\bar{\mathbf{f}}_j-\mathbf{b}_j\|_2^2\\ &\quad\times\tau\sum_{i,j,t}\left[1-\left(\bar{\mathbf{g}}_t^T\bar{\mathbf{f}}_j-1\right)^2+\left(\bar{\mathbf{g}}_i^T\bar{\mathbf{f}}_j-1\right)^2\right]_+\\ &\quad+\eta\left\|\sqrt{k}\bar{\mathbf{F}}^T\mathbf{1}\right\|_2^2.\end{aligned} \tag{9}$$

We obtain the following derivative of L_f with respect to $\bar{\mathbf{f}}_j$:

$$\begin{aligned}\frac{\partial L_f}{\partial\bar{\mathbf{f}}_j} &= 2\sum_i\mathbf{b}_i\left(\mathbf{b}_i^T\bar{\mathbf{f}}_j-\sqrt{k}S_{ij}\right)+2\gamma\sqrt{k}(\sqrt{k}\bar{\mathbf{f}}_j-\mathbf{b}_j)\\ &\quad+2\tau\sum_{i,t}M_{it}\left(-\bar{\mathbf{g}}_t(\bar{\mathbf{g}}_t^T\bar{\mathbf{f}}_j-1)+\bar{\mathbf{g}}_i(\bar{\mathbf{g}}_i^T\bar{\mathbf{f}}_j-1)\right)\\ &\quad+2\eta k\bar{\mathbf{F}}^T\mathbf{1}\end{aligned} \tag{10}$$

where $\mathbf{M}$ is a mask matrix and its component in the ith row and jth column is denoted as M_{it}. $M_{it} = 1$ if $[1-(\bar{\mathbf{f}}_t^T\bar{\mathbf{f}}_j-1)^2+(\bar{\mathbf{f}}_i^T\bar{\mathbf{f}}_j-1)^2] > 0$, otherwise $M_{it} = 0$.

Then the derivative of $\bar{\mathbf{f}}_j$ with respect to $\mathbf{f}_j$ is calculated as follows:

$$\frac{\partial\bar{\mathbf{f}}_j}{\partial\mathbf{f}_j} = \frac{\mathbf{I}}{\|\mathbf{f}_j\|}-\frac{\mathbf{f}_j\mathbf{f}_j^T}{\|\mathbf{f}_j\|^3}. \tag{11}$$

Combining (10) and (11) together and exploiting the chain rule, the derivative of L_f with respect to $\mathbf{f}_j$ is

$$\frac{\partial L_f}{\partial\mathbf{f}_j} = \frac{\partial\bar{\mathbf{f}}_j}{\partial\mathbf{f}_j}\frac{\partial L_f}{\partial\bar{\mathbf{f}}_j}. \tag{12}$$

After getting the gradient $(\partial L_f/\partial\mathbf{f}_j)$, the chain rule is used to obtain $(\partial L_f/\partial\mathbf{W}_f)$, where $\mathbf{W}_f$ is the weight in the first stream. $\mathbf{W}_f$ is updated by using backpropagation.

2) Update $\mathbf{G}$ With $\mathbf{F}$ and $\mathbf{B}$ Fixed: By fixing $\mathbf{F}$ and $\mathbf{B}$, the objective function (8) is transformed to

$$\begin{aligned}\min\ L_g &= \sum_{i,j}\left(\mathbf{b}_i^T\bar{\mathbf{g}}_j-\sqrt{k}S_{ij}\right)^2+\gamma\sum_j\|\sqrt{k}\bar{\mathbf{g}}_j-\mathbf{b}_j\|_2^2\\ &\quad\times\tau\sum_{i,j,t}\left[1-\left(\bar{\mathbf{f}}_t^T\bar{\mathbf{g}}_j-1\right)^2+\left(\bar{\mathbf{f}}_i^T\bar{\mathbf{g}}_j-1\right)^2\right]_+\\ &\quad+\eta\|\sqrt{k}\bar{\mathbf{G}}^T\mathbf{1}\|_2^2.\end{aligned} \tag{13}$$

Similarly, the derivative of L_g with respect to $\mathbf{g}_j$ is

$$\frac{\partial L_g}{\partial\mathbf{g}_j} = \frac{\partial\bar{\mathbf{g}}_j}{\partial\mathbf{g}_j}\frac{\partial L_g}{\partial\bar{\mathbf{g}}_j} \tag{14}$$

where

$$\frac{\partial \bar{\mathbf{g}}_j}{\partial \mathbf{g}_j} = \frac{\mathbf{I}}{\|\mathbf{g}_j\|} - \frac{\mathbf{g}_j \mathbf{g}_j^T}{\|\mathbf{g}_j\|^3} \tag{15}$$

$$\begin{aligned}\frac{\partial L_g}{\partial \bar{\mathbf{g}}_j} &= 2\sum_i \mathbf{b}_i\big(\mathbf{b}_i^T \bar{\mathbf{g}}_j - \sqrt{k}S_{ij}\big) + 2\gamma\sqrt{k}(\sqrt{k}\bar{\mathbf{g}}_j - \mathbf{b}_j) \\ &\quad + 2\tau \sum_{i,t} M_{it}\big(-\bar{\mathbf{f}}_t\big(\bar{\mathbf{f}}_t^T \bar{\mathbf{g}}_j - 1\big) + \bar{\mathbf{f}}_i\big(\bar{\mathbf{f}}_i^T \bar{\mathbf{g}}_j - 1\big)\big) \\ &\quad + 2\eta k \bar{\mathbf{G}}^T \mathbf{1}.\end{aligned} \tag{16}$$

After getting the gradient $(\partial L_g / \partial \mathbf{g}_j)$, we use the chain rule to obtain $(\partial L_g / \partial \mathbf{W}_g)$, where $\mathbf{W}_g$ is the weight in the second stream. $\mathbf{W}_g$ is updated by using backpropagation.

3) Update **B** *With* **F** *and* **G** *Fixed:* By fixing **F** and **G**, the objective function (8) is transformed to

$$\begin{aligned}\min\ L_b &= \|\bar{\mathbf{F}}\mathbf{B}^T - \sqrt{k}\mathbf{S}\|_F^2 + \|\bar{\mathbf{G}}\mathbf{B}^T - \sqrt{k}\mathbf{S}\|_F^2 \\ &\quad + \gamma\big(\|\sqrt{k}\bar{\mathbf{F}} - \mathbf{B}\|_F^2 + \|\sqrt{k}\bar{\mathbf{G}} - \mathbf{B}\|_F^2\big) \\ \text{s.t. } \mathbf{b}_i &\in \{-1, +1\}.\end{aligned} \tag{17}$$

Then (17) can be rewritten as

$$\begin{aligned}\min\ L_b &= -2\mathrm{Tr}[\mathbf{B}(\sqrt{k}(\bar{\mathbf{F}}^T + \bar{\mathbf{G}}^T)(\mathbf{S} + \gamma\mathbf{I}))] \\ &\quad + \|\mathbf{B}\bar{\mathbf{F}}^T\|_F^2 + \|\mathbf{B}\bar{\mathbf{G}}^T\|_F^2 + \text{const.} \\ \text{s.t. } \mathbf{b}_i &\in \{-1, +1\}\end{aligned} \tag{18}$$

where "const" means a constant value without any association with **B**. Here let $\mathbf{Q} = -2\sqrt{k}(\mathbf{S}^T + \gamma\mathbf{I})(\bar{\mathbf{F}} + \bar{\mathbf{G}})$. Equation (18) can then be simplified to

$$\begin{aligned}\min\ L_b &= \|\mathbf{B}\bar{\mathbf{F}}^T\|_F^2 + \|\mathbf{B}\bar{\mathbf{G}}^T\|_F^2 + \mathrm{Tr}[\mathbf{B}\mathbf{Q}^T] + \text{const} \\ \text{s.t. } \mathbf{b}_i &\in \{-1, +1\}.\end{aligned} \tag{19}$$

Since it is difficult to optimize **B** directly, we update it bit by bit. In other words, we update one column in **B** with remaining columns fixed. Denote $\mathbf{B}_{*c}$ as the cth column and $\hat{\mathbf{B}}_c$ as the remaining columns in **B**. The same can be applied to $\mathbf{F}_{*c}$, $\hat{\mathbf{F}}_c$, $\mathbf{G}_{*c}$, $\hat{\mathbf{G}}_c$, $\mathbf{Q}_{*c}$, and $\hat{\mathbf{Q}}_c$. When we optimize $\mathbf{B}_{*\mathbf{c}}$, (19) can then be rewritten as

$$\begin{aligned}&\min_{\mathbf{B}_{*\mathbf{c}}}\ \mathrm{Tr}\big(\mathbf{B}_{*\mathbf{c}}\big[2\big(\mathbf{F}_{*c}^T\hat{\mathbf{F}}_c + \mathbf{G}_{*c}^T\hat{\mathbf{G}}_c\big)\hat{\mathbf{B}}_c^T + \mathbf{Q}_{*c}^T\big] + \text{const} \\ &\text{s.t. } \mathbf{B} \in \{-1, +1\}^{n\times k}.\end{aligned} \tag{20}$$

It is easy to get the optimal solution for $\mathbf{B}_{*\mathbf{c}}$

$$\mathbf{B}_{*\mathbf{c}} = -\mathrm{sign}\big(2\hat{\mathbf{B}}_c\big(\hat{\mathbf{F}}_c^T\mathbf{F}_{*c} + \hat{\mathbf{G}}_c^T\mathbf{G}_{*c}\big) + \mathbf{Q}_{*c}\big). \tag{21}$$

After computing $\mathbf{B}_{*\mathbf{c}}$, we update **B** by replacing the cth column with $\mathbf{B}_{*\mathbf{c}}$. Then we repeat (21) until all columns are updated.

Algorithm 1 shows the details of the optimization.

Algorithm 1 RADH

Input: Training data **X/Y**; similarity matrix **S**; hash code length k; predefined parameters τ, γ and η.

Output: Hashing functions $\mathcal{F}$ and $\mathcal{G}$ for the two streams, respectively.

Initialization: Initialize weights of the first seven layers by using the pretrained ImageNet model; the last layer is initialized randomly; **B** is set to be a matrix whose elements are zero.

while not converged or the maximum iteration is not reached **do**

  Update $(\mathbf{F}, \mathbf{W}_f)$:
  Fix $(\mathbf{G}, \mathbf{W}_g)$ and **B** and update $(\mathbf{F}, \mathbf{W}_f)$ using back-propagation according to Eq.(12).

  Update $(\mathbf{G}, \mathbf{W}_g)$:
  Fix $(\mathbf{F}, \mathbf{W}_f)$ and **B** and update $(\mathbf{G}, \mathbf{W}_g)$ using back-propagation according to Eq.(14).

  Update B:
  Fix $(\mathbf{F}, \mathbf{W}_f)$ and $(\mathbf{G}, \mathbf{W}_g)$ and update **B** according to Eq.(21).

end while

D. Out-of-Sample

After performing training, we obtain the weights $\mathbf{W}_f$ and $\mathbf{W}_g$ associated with the first and second streams. Given an out-of-sample image $\mathbf{x}^*$ belonging to the query set, we can get its corresponding binary code by following either $\mathbf{b}_f^* = \mathrm{sign}(\mathcal{F}(\mathbf{x}^*, \mathbf{W}_f)$ or $\mathbf{b}_g^* = \mathrm{sign}(\mathcal{G}(\mathbf{x}^*, \mathbf{W}_g)$. The experimental results show that $\mathbf{b}_f^*$ and $\mathbf{b}_g^*$ achieve the similar performance. However, in order to obtain a more robust result, we regard their average as the final result, as shown in the following equation:

$$\mathbf{b}^* = \mathrm{sign}(\mathcal{F}(\mathbf{x}^*, \mathbf{W}_f) + \mathcal{G}(\mathbf{x}^*, \mathbf{W}_g)). \tag{22}$$

E. Implementation

We implement RADH under the deep learning toolbox MatConvNet [46] on a Titan X GPU. Specifically, the pretrained model of CNN-F on the ImageNet data set is applied to both streams to initialize their weights, which is greatly beneficial to the performance improvement. Since the output of the last layer is different from that in the original CNN-F model, we randomly initialize the weight in this layer. In the training phase, we set the maximized epoch to be 150, the learning rate to be 10^{-3}, the batch size to be 64, and the weight decay to be 5×10^{-4}. The stochastic gradient descent (SGD) is then exploited to update the weights.

In (5), the triplet loss is achieved by ranking each positive pair and negative pair. However, it is too time consuming if we take all training samples into account. In our experiments, we sort the positive pairs in descending order and the negative pairs in ascending order by following $(\mathbf{f}_i^T\mathbf{g}_j - 1)^2$ / $(\mathbf{g}_i^T\mathbf{f}_j - 1)^2$. We then select top 200 samples from each part as the positive and negative instances, respectively.

IV. Experiments

In this section, different experiments on four data sets are conducted based on different experimental settings. We first followed the settings in some existing methods to make a comparison between RADH and other deep hashing methods. In order to better demonstrate the generality of the proposed model, additional experiments based on our settings are also conducted, followed by the analysis on the sensitivity of the parameters τ, γ, and η.

A. Data Sets and Evaluation Protocol

In this article, four real-world data sets, including CIFAR-10 [47], NUS-WIDE [48], IAPR-12 [49], and MIRFLICKR-25K [50], are used to evaluate the superiority of RADH.

CIFAR-10 is composed of 60 000 images, where each sample belongs to one of ten classes.

NUS-WIDE consists of 269 648 web images. Following [17] and [19], 195 834 images are selected, which are associated with the 21 most frequent tags. Since images in this data set are multilabel, $S_{ij} = 1$ if there is at least one label between $\mathbf{x}_i$ and $\mathbf{x}_j$, otherwise $S_{ij} = 0 - \varepsilon$. Here ε is set to 0.11.

IAPR-12 is composed of 20 000 images associated with 255 classes. Note that some samples in the IAPR-12 data set have multiple labels. The definition of S_{ij} is the same to that in NUS-WIDE.

MIRFLICKR-25K is composed of 25 000 images collected from Flickr. Being similar to the IAPR-12 data set, this is also a multilabel data set. Here, we also define two images to be ground-truth neighbors if they share at least one common label. According to [51], 20 015 images annotated with at least 24 tags are selected.

To quantatively evaluate different approaches, the mean average precision (MAP) and top-k precision are adopted.

B. Existing Experimental Settings

Since the CIFAR-10 and NUS-WIDE data sets are the most common data sets used in existing deep hashing methods, we show the results conducted on these two data sets. Being similar to existing methods, MAP and Top-50 000 MAP are applied to evaluate the performance of different approaches on the CIFAR-10 and NUS-WIDE data sets, respectively. Note that all results of the comparison methods in this subsection are directly copied from the published articles [17], [19], [35]. Besides, DRSCH, DSCH, and DSRH are the triplet label-based methods.

Note that, since the training number is quite large and if we use all training samples in each epoch, the required storage of the similarity matrix $\mathbf{S}$ would be very huge, being impractical in the implementation. Thus, being similar to ADSH, we also randomly sample a part of instances from the training set in each epoch. Table I makes a comparison among RADH, DSH-DL, DHN, and ADSH conducted on the CIFAR-10 and NUS-WIDE data sets. Here 1000 samples are selected for testing (100 per class) and remaining samples are used for training in the CIFAR-10 data set. Similarly, 2100 samples (100 per class) are selected for testing and the rest of images are used for training in the NUS-WIDE data set. As we can see, RADH obtains much better performance compared with these deep hashing methods in all cases. Especially in the CIFAR-10 data set, when the code length is small, e.g., 12-bit, the gap of the performance obtained by RADH and other methods is quite large, relatively indicating the effectiveness of our method. In the NUS-WIDE data set, RADH always achieves the remarkable improvement compared with DPSH and several existing triplet label-based methods.

TABLE I

MAP AND MAP@TOP50000 SCORES OBTAINED BY DIFFERENT METHODS ON THE CIFAR-10 AND NUS-WIDE DATA SETS, RESPECTIVELY. 1000 AND 2100 SAMPLES ARE SELECTED FOR TESTING AND REMAINING SAMPLES ARE USED FOR TRAINING IN THESE TWO DATA SETS, RESPECTIVELY

Dataset	CIFAR-10			
Method	12-bit	24-bit	32-bit	48-bit
DSH-DL	63.70	74.93	78.03	80.86
DHN	67.94	72.01	73.09	74.08
DPSH	68.63	72.76	74.06	75.20
ADSH	84.66	90.62	91.75	92.63
RADH	**94.39**	**95.11**	**95.39**	**95.17**
Dataset	NUS-WIDE			
Method	16-bit	24-bit	32-bit	48-bit
DRSCH	61.8	62.2	62.3	62.8
DSCH	59.2	59.7	61.1	60.9
DSRH	60.9	61.8	62.1	63.1
DPSH	71.5	72.2	73.6	74.1
RADH	**83.3**	**84.2**	**85.0**	**85.0**

TABLE II

MAP SCORES OBTAINED BY DIFFERENT METHODS ON THE CIFAR-10 DATA SET

Method	8-bit	12-bit	16-bit	24-bit	36-bit	48-bit
LSH	14.69	14.22	17.17	15.56	16.54	16.44
SpH	18.22	19.14	18.98	21.50	21.82	23.09
ITQ	24.47	25.56	24.52	24.75	26.51	26.49
DSH	22.07	20.99	21.48	22.69	23.93	23.87
DPSH	63.48	66.97	68.83	73.45	74.66	75.02
ADSH	56.67	71.41	76.50	80.40	82.73	82.73
DAPH	59.09	61.17	68.15	69.22	70.74	70.28
RADH	**76.27**	**77.92**	**79.79**	**82.35**	**83.12**	**84.43**

C. Additional Experimental Settings

Although our method obtains satisfactory performance in Table I, it cannot comprehensively show the generality of the models due to the too large number of training samples. To address this problem, we aim to divide the data set into three parts: training, retrieval, and testing sets.

CIFAR-10: 100 samples per class are used for testing, 500 samples per class are selected for training, and the remaining images are used for retrieval.

NUS-WIDE: 2100 images are selected for testing and the remaining images are used for retrieval, in which 10 500 images are selected for training.

IAPR-12: The retrieval set consists of 18 000 images and the test set is composed of the remaining 2000 images. Furthermore, we randomly select 5000 images from the retrieval set to be the training set.

MIRFLICKR-25K 2000 are used for testing while the remaining samples are regarded as the retrieval subset. Also, 5000 images from this retrieval subset are randomly selected for training.

Some existing hashing methods including data-independent and data-dependent are compared with our proposed method to demonstrate its superiority. These are LSH [8], SpH [12], ITQ [28], DSH [26], DPSH [17], ADSH [19], and DAPH [39]. Since LSH, SpH, ITQ, and DSH are traditional

TABLE III

MAP@TOP500 AND PRECISION@TOP500 SCORES OBTAINED BY DIFFERENT METHODS ON THE CIFAR-10 DATA SET

Evaluation	MAP@Top500						Precision@Top500					
Method	8-bit	12-bit	16-bit	24-bit	36-bit	48-bit	8-bit	12-bit	16-bit	24-bit	36-bit	48-bit
LSH	17.73	24.97	29.06	28.03	30.39	32.59	14.37	18.30	22.85	22.70	25.54	27.19
SpH	31.10	32.87	33.41	38.64	41.24	43.35	24.15	27.59	28.59	34.24	36.60	38.73
ITQ	26.43	37.48	37.42	41.06	45.57	46.16	24.10	33.00	31.86	34.87	40.53	40.78
DSH	27.25	34.14	34.64	37.00	40.02	41.33	24.47	28.20	29.33	33.07	35.54	36.55
DPSH	58.58	66.85	71.48	75.74	79.69	80.62	64.13	70.80	74.71	78.34	80.59	81.55
ADSH	58.92	70.07	74.95	78.09	78.85	77.61	61.09	73.71	78.85	81.84	83.45	82.86
DAPH	50.88	66.24	72.43	77.31	79.21	80.40	54.22	68.28	74.42	77.36	78.80	79.55
RADH	**68.72**	**70.39**	**77.36**	**79.39**	**82.90**	**84.92**	**76.37**	**76.94**	**81.62**	**82.88**	**85.31**	**86.47**

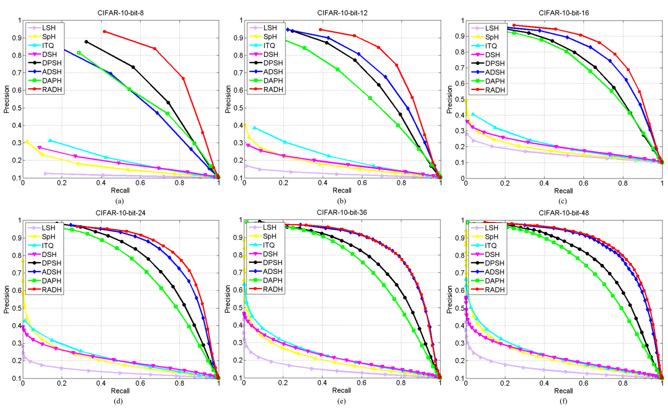


Fig. 4. PR curves computed by LSH, SpH, ITQ, SDH, SGH, DPSH, ADSH, DAPH, and RADH on the CIFAR-10 data set. (a)–(f) Figures associated with the code lengths of 8-bit, 12-bit, 16-bit, 24-bit, 36-bit, and 48-bit.

TABLE IV

MAP SCORES OBTAINED BY DIFFERENT METHODS ON THE NUS-WIDE DATA SET

Method	8-bit	12-bit	16-bit	24-bit	36-bit	48-bit
LSH	38.78	38.28	41.58	41.32	44.32	46.36
SpH	43.18	44.64	45.28	46.66	47.24	48.20
ITQ	54.40	56.16	56.04	56.75	57.34	57.56
DSH	52.80	47.66	49.24	49.85	50.51	52.24
DPSH	62.86	63.63	66.23	66.65	67.96	68.12
ADSH	58.79	61.73	63.97	65.38	63.86	63.98
DAPH	60.17	61.88	62.56	63.52	63.30	64.62
RADH	**64.90**	**68.07**	**70.30**	**71.60**	**73.04**	**73.47**

hashing methods, the features should be extracted in advance. In this article, we use the CNN feature to be the input for these methods. We implement RADH with the MatconvNet tool under the CNN-F structure. This is done to make a fair comparison since the released codes of DPSH and ADSH are also based on the MatconvNet and CNN-F. Also, since DAPH is similar to DPSH and can be easily re-implemented by using MatconvNet and CNN-F, we also make a comparison between RADH and DAPH. Specifically, for DPSH and ADSH that are all deep learning methods, we apply the CNN-F as the network for the feature extraction and parameters in them are set according to the descriptions in their publications. For DAPH, the original network is ResNet. We replace it with the CNN-F and try our best to tune the parameters in the modified DAPH. Specifically, these three deep hashing methods are implemented with MatConvNet and the raw image is their input. Note that DPSH, ADSH, and DAPH achieve the state-of-the-art performance in existing deep hashing methods. Thus, it is reasonable for us to only compare RADH with them.

Here the MAP is exploited as the evaluation protocol. Furthermore, being similar to [39], we also adopt the MAP of the top 500 retrieval samples (MAP@500) and the mean precision of the top 500 retrieval samples (Precision@500) as another two evaluation protocols.

1) Comparison With Other Methods: We make a comparison between the proposed method and other state-of-the-art approaches on four real-world data sets. Note that γ and τ are set to 500 and 0.1 for the four data sets, respectively. η is set to 0.3 for the NUS-WIDE, IAPR TC-12, and MIRFLICKR-25K data sets, while it is set to 1 for the CIFAR-10 data set. All these values are selected by cross-validation.

a) CIFAR-10: The experimental results about MAP scores obtained by different methods are tabulated in Table II. From this table, it is easy to see that RADH arrives to the highest points on the MAP score in all cases, especially when the code length is relatively small. Compared with LSH, SpH, ITQ, and DSH, which are not deep learning methods, DPSH, ADSH, DAPH, and our proposed RADH always get the better results. In contrast to these three deep comparison methods, RADH also has more or less improvement. Particularly, when the code length is only 8-bit, our method obtains the much larger enhancement, achieving more than 10% on the MAP score.

Table III further lists the MAP@Top500 and Precision@Top500 scores computed by different approaches. As we can see, RADH is still superior to other approaches. When the code length is set to be 8-bit, RADH achieves almost more than 10% enhancement on the both MAP@Top500 and Precision@Top500 scores, compared with all comparison methods. With the increase in the code length, although the performance gap meets a degradation, RADH is still superior to the other algorithms. Particularly, when the code length is set to be 48-bit, our proposed method gains about 4% improvement.

Fig. 4 plots the precision–recall (PR) curves when the code length ranges from 8-bit to 48-bit on the CIFAR-10 data set. It is obvious to observe that the presented strategy covers the largest or competitive areas. Except for the case when the

TABLE V

MAP@Top500 and Precision@Top500 Scores Obtained by Different Methods on the NUS-WIDE Data Set

Evaluation	MAP@Top500						Precision@Top500					
Method	8-bit	12-bit	16-bit	24-bit	32-bit	48-bit	8-bit	12-bit	16-bit	24-bit	32-bit	48-bit
LSH	43.49	47.46	53.39	57.58	63.95	68.40	43.82	45.83	51.81	55.28	61.48	65.82
SpH	57.78	63.70	65.69	69.11	72.01	73.34	57.07	61.95	63.62	67.07	69.78	71.06
ITQ	67.85	72.96	74.95	77.07	78.86	78.83	67.96	72.46	74.07	75.98	77.63	78.51
DSH	62.85	65.55	67.28	70.95	72.24	73.82	62.30	64.26	65.88	69.34	70.34	72.02
DPSH	**75.27**	77.42	79.62	81.39	82.27	82.90	75.04	76.88	79.25	80.82	81.79	82.32
ADSH	63.35	68.77	72.62	74.99	74.02	75.14	63.77	68.82	72.35	74.39	73.72	74.75
DAPH	74.27	78.03	80.07	81.22	82.39	83.71	74.40	77.67	79.28	80.63	81.79	82.97
RADH	75.03	**79.15**	**81.19**	**82.94**	**84.35**	**84.96**	**75.21**	**79.26**	**81.06**	**82.78**	**84.15**	**84.63**

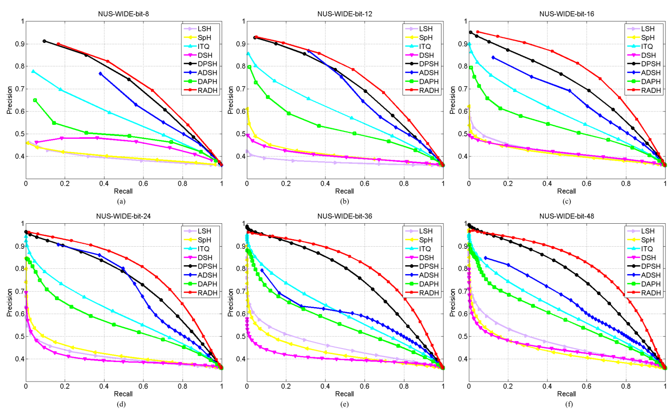


Fig. 5. PR curves computed by LSH, SpH, ITQ, SDH, SGH, DPSH, ADSH, DAPH, and RADH on the NUS-WIDE data set. (a)–(f) Figures associated with the code length of 8-bit, 12-bit, 16-bit, 24-bit, 36-bit, and 48-bit.

TABLE VI

MAP Scores Obtained by Different Methods on the IAPR TC-12 Data Set

Method	8-bit	12-bit	16-bit	24-bit	36-bit	48-bit
LSH	34.07	33.98	35.11	35.79	36.71	37.94
SpH	36.04	36.04	36.75	37.77	38.56	39.11
ITQ	41.36	42.20	42.89	43.10	43.20	43.91
DSH	40.64	40.09	39.73	41.66	42.42	42.33
DPSH	45.19	46.03	46.82	47.37	47.97	48.60
ADSH	44.69	46.98	48.25	49.06	50.24	50.59
DAPH	44.33	44.48	44.73	45.12	45.24	45.52
RADH	**47.16**	**50.59**	**52.13**	**53.71**	**54.80**	**55.38**

code length is 36-bit, RADH gains more or less improvement. In comparison to LSH, SpH, ITQ, and DSH, there are much larger areas computed by RADH. Although RADH only has a slight increase when the code length is 48-bit compared with DPSH, ADSH, and DAPH, it enlarges the gap of covered areas remarkably when the code length is 8-bit, 12-bit, 16-bit, and 24-bit, demonstrating its superiority.

b) NUS-WIDE: The MAP scores obtained by different methods are shown in Table IV. As we can see, the proposed method RADH achieves a remarkable improvement compared with other approaches. In contrast to the nondeep learning strategies, including LSH, SpH, ITQ, and DSH, RADH always gains about more than 15% enhancement. Referring to the deep hashing methods, RADH is still superior to them. Particularly, our presented approach obtains 73.47% on the MAP score, while the best performance computed by other deep hashing methods is only 68.12%.

Table V tabulates the MAP@Top500 and Precision@Top500 scores, which also demonstrate the superiority of the proposed method. Except for the case when the code length is 8-bit, RADH arrives to the highest points. In comparison to LSH, SpH, ITQ, and DSH, there is at least 5% improvement in both metrics. Compared with DPSH and DAPH, our approach still gains more or less enhancement. Note that, from Tables IV and V, we can see although ADSH obtains better results in the MAP score compared with ITQ, its MAP@Top500 and Precision@Top500 scores are much inferior to that of ITQ, relatively indicating its instability. By contrast, our proposed method RADH can get satisfactory results in MAP, MAP@Top500, and Precision@Top500 scores.

The PR curves computed by various methods are displayed in Fig. 5. It is easy to observe that RADH covers the largest areas when the code length ranges from 8-bit to 48-bit. Specifically, DPSH and RADH always gain satisfactory PR curves compared with other methods. In contrast to DPSH, RADH also achieves an obvious improvement.

c) IAPR TC-12: The experimental results on MAP scores obtained by various methods are listed in Table VI. As we can see, RADH gains better performance than all comparison methods. For instance, as listed in Table VI, the MAP scores computed by RADH are much higher than that obtained by LSH, SpH, ITQ, and DSH, which relatively demonstrate that data-independent and traditional data-dependent hashing methods are often inferior to deep hashing strategies. In contrast to other deep hashing methods, our proposed method is also superior. In most cases, RADH gets more than 4% improvement on the MAP score, indicating the effectiveness of our relaxed strategy and novel triplet loss.

Referring to MAP@Top500 and Precision@Top500 shown in Table VII, there is also a remarkable improvement for RADH compared with other approaches. In contrast to LSH, SpH, ITQ, and DSH, RADH gains more than 10% improvement on the MAP@Top500 and Precision@Top500 scores in most cases. In comparison to DPSH, ADSH, and DAPH, the proposed method also achieves more than 5% enhancement on the two evaluation protocols when the code length ranges from 12 to 48. Furthermore, in these three comparison deep hashing methods, although ADSH obtains better performance compared with DPSH and DAPH when the code length is 12, 16, 24, 36, or 48, it has much inferior results when the code

TABLE VII

MAP@TOP500 AND PRECISION@TOP500 SCORES OBTAINED BY DIFFERENT METHODS ON THE IAPR TC-12 DATA SET

Evaluation	MAP@Top500						Precision@Top500					
Method	8-bit	12-bit	16-bit	24-bit	36-bit	48-bit	8-bit	12-bit	16-bit	24-bit	36-bit	48-bit
LSH	39.87	39.83	42.72	44.80	48.07	50.78	38.14	37.90	40.41	42.05	44.45	46.85
SpH	44.38	45.43	47.73	50.41	52.45	54.30	42.33	42.80	44.69	46.94	48.49	49.95
ITQ	51.75	54.48	55.93	57.64	58.83	59.90	49.95	52.16	53.31	54.33	55.02	56.01
DSH	48.34	48.63	49.70	52.74	54.78	55.51	47.46	46.87	47.21	50.12	51.61	52.20
DPSH	57.07	58.13	59.94	61.61	63.05	64.49	55.03	55.90	57.73	58.96	60.15	61.40
ADSH	53.70	58.12	61.35	63.25	64.90	65.59	52.31	56.47	58.97	60.29	61.94	62.50
DAPH	56.26	57.98	59.48	61.27	62.57	63.94	54.32	55.44	56.52	57.92	58.95	60.03
RADH	**58.99**	**64.67**	**67.34**	**69.12**	**70.91**	**71.64**	**57.29**	**61.81**	**64.45**	**66.25**	**67.86**	**68.40**

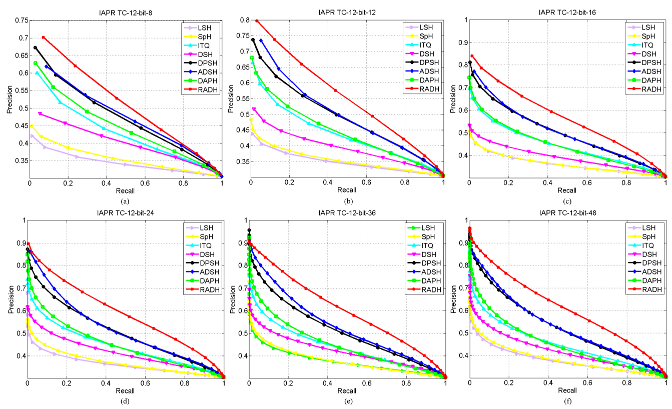


Fig. 6. PR curves computed by LSH, SpH, ITQ, SDH, SGH, DPSH, ADSH, DAPH, and RADH on the IAPR TC-12 data set. (a)–(f) Figures associated with the code length of 8-bit, 12-bit, 16-bit, 24-bit, 36-bit, and 48-bit.

TABLE VIII

MAP SCORES OBTAINED BY DIFFERENT METHODS ON THE MIRFLICKR-25K DATA SET

Method	8-bit	12-bit	16-bit	24-bit	36-bit	48-bit
LSH	57.14	57.21	58.25	59.37	59.62	59.86
SpH	59.20	60.71	60.91	60.97	61.63	62.43
ITQ	63.09	63.40	63.52	63.82	64.06	64.34
DSH	64.48	63.85	64.10	65.96	65.00	65.62
DPSH	73.48	74.68	75.58	76.01	76.09	76.05
ADSH	75.39	76.41	76.98	76.59	76.20	74.53
DAPH	72.79	74.70	74.30	74.14	73.81	73.41
RADH	**75.50**	**76.77**	**78.09**	**79.18**	**80.13**	**80.38**

length is 8. By contrast, our proposed method RADH always gets the best performance no matter how long the code length is, relatively indicating its superiority.

Fig. 6 shows the PR curves obtained by different methods when the bit length ranges from 8 to 48. It is easy to observe that PR curves computed by RADH cover larger areas compared with that obtained by different comparison approaches. In comparison to DPSH, ADSH, and DAPH, RADH can get an obvious improvement due to the relaxed asymmetric strategy and the novel triplet loss.

d) MIRFLICKR-25K: The experimental results about MAP scores on the MIRFLICKR-25K data set computed by RADH and other comparison methods are tabulated in Table VIII. Similarly, our presented strategy is superior to other methods on this evaluation protocol. Making a comparison between the deep learning-based hashing methods and data-independent and traditional data-dependent approaches, deep hashing methods achieve much better results. In contrast to DPSH, ADSH, and DAPH, although RADH obtains a slight enhancement on the MAP scores when the bit length is 8 and 12, it achieves larger improvement when the code length ranges from 16 to 48. Particularly, RADH reaches 80.13% and 80.38% when the code length is 36 and 48, respectively, while the best results gained by the three deep hashing approaches are only 76.20% and 76.05%, being far lower than that obtained by RADH.

Referring to MAP@Top500 and Precision@Top500 scores tabulated in Table IX, our method has more remarkable achievement. In contrast to LSH, SpH, ITQ, and DSH, there is more than 10% improvement on both evaluation protocols. Compared with the three deep hashing methods, RADH gains also about 2%–4% enhancement when the code length ranges from 8 to 48.

The PR curves on the MIRFLICK-25K data set computed by LSH, SpH, ITQ, DSH, DPSH, ADSH, DAPH, and RADH are shown in Fig. 7. As we can see, although RADH has a similar result compared with ADSH when the code length is 8, it covers larger areas in other situations. Particularly, with the increase in the code length, our presented approach enlarges the gap of covered areas compared with the three deep learning based hashing methods. Furthermore, in Fig. 7, despite the fact that ADSH has a competitive performance on the PR curve, it is inferior to DPSH when the code length increases to 24, 36, and 48, indicating its instability. By contrast, RADH always obtains the competitive or best performance no matter how long the code length is, which demonstrates the effectiveness and robustness of our method.

As mentioned above, to accelerate the speed, RADH samples a part of training samples in each training epoch. To make a fair comparison, we remove this sampling operation in RADH and reconduct experiments when the code length is set to 16-bit. The results of the three metrics are (78.96, 76.59, 80.91), (64.18, 72.92, 73.12), (48.55, 62.27, 59.80), and (76.83, 85.09, 84.27) for the four data sets, respectively. Obviously, there is a slight improvement for ADSH if all the training samples are used in each epoch. However, compared with our proposed method RADH, ADSH is still inferior.

Totally, RADH achieves the best performances except one case in Table V. In fact, both the distributions of different data

TABLE IX

MAP@Top500 and Precision@Top500 Scores Obtained by Different Methods on the MIRFLICKR-25K Data Set

Evaluation	MAP@Top500						Precision@Top500					
Method	8-bit	12-bit	16-bit	24-bit	36-bit	48-bit	8-bit	12-bit	16-bit	24-bit	36-bit	48-bit
LSH	60.02	60.81	63.66	65.86	67.81	68.47	59.25	59.65	62.20	64.33	65.74	66.31
SpH	65.61	68.36	69.49	70.33	71.64	73.65	64.19	66.65	67.56	68.34	69.43	71.26
ITQ	72.30	73.90	74.30	75.60	76.19	77.06	70.95	72.09	72.42	73.54	74.08	74.66
DSH	70.59	71.83	73.31	75.43	74.89	76.37	69.90	70.50	71.82	73.90	73.12	74.52
DPSH	82.88	83.84	84.34	84.84	85.77	85.64	81.85	83.01	83.58	84.11	84.97	84.80
ADSH	82.14	83.80	84.94	84.90	84.20	82.06	81.50	83.18	84.15	84.03	83.52	81.31
DAPH	81.08	84.20	83.71	84.45	84.02	84.07	80.37	83.24	82.73	83.40	82.93	82.91
RADH	**84.91**	**85.78**	**86.52**	**87.75**	**88.66**	**88.39**	**84.14**	**84.88**	**85.96**	**87.07**	**88.00**	**87.77**

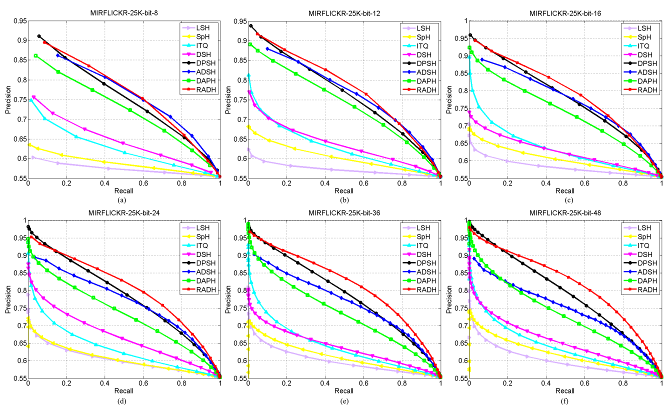


Fig. 7. PR curves computed by LSH, SpH, ITQ, SDH, SGH, DPSH, ADSH, DAPH, and RADH on the MIRFLICKR-25K data set. (a)–(f) Figures associated with the code length of 8-bit, 12-bit, 16-bit, 24-bit, 36-bit, and 48-bit.

sets and the numbers of training samples can influence the performance. To demonstrate the superiority of the proposed method, it should achieve better performances on most of data sets. In other words, it is possible that the proposed method is inferior to other comparison methods in some specific cases. Particularly, in Table I, RADH obtains much higher results on the NUS-WIDE data set when the code length is small, while it gets much more remarkable improvements in most of the remaining experiments when the code length is large. Thus, with various code lengths, the reason why RADH gains different levels of enhancements is mainly associated with different data sets and training numbers. Also, as mentioned above, except the case when the code length is set to 8-bit on the NUS-WIDE data set in Table V, RADH obtains more or less an improvement on MAP values in all remaining cases, further demonstrating the effectiveness and robustness of RADH. In addition, referring to Table V, although RADH is occasionally inferior to DPSH due to different training numbers, their gap is quite tiny, which also relatively substantiates the superiority of RADH.

2) Parameter Sensitivity Analysis: In our proposed method, three parameters, including η, γ, and τ, are introduced. Additionally, the learning rate, batch size, and inner iteration of (21) are also needed to be predefined. Here, we analyze the sensitivity of them under different values when the code length is 16-bit.

Fig. 8 displays the experimental results under the change of η. Note that the γ and τ are set to 500 and 0.1 for the four data sets, respectively. It is easy to see that with the increase of η from a small value 0.01, there is a slight performance improvement for the four data sets. Specifically, when η arrives at 1, 0.3, 0.3, and 0.1 for the four data sets, which are quite similar to our settings, RADH achieves the best performance on the MAP scores. Furthermore, from this figure we can also see that the proposed method is robust on η. When η is located in the ranges of $[0.01, 4]$, $[0.01, 5]$, $[0.01, 5]$ and $[0.01, 5]$ for these four data sets, RADH can always achieve satisfactory performance.

The influences on the experimental results with different values of γ are shown in Fig. 9. Note that the τ is set to 0.1 for the four data sets. The η is set to 0.3 for the NUS-WIDE, IAPR TC-12 and MIRFLICK-25K data sets, while it is set to 1 for the CIFAR-10 data set. From this figure, we can observe that our presented approach is also insensitive to γ. When γ is located in the range from 100 to 700, the maximum of the performance gap is quite small on the MAP score.

Fig. 10 plots the MAP, MAP@Top500, and Precision@Top500 scores when τ ranges from 0.0001 to 10. Note that in this experiment, γ is set to 500 for the four data sets. The η is also set to 1 for the first data set and 0.3 for the last three data sets. As we can see, with the increase of τ, there is about 2%–3% enhancement on the MAP scores for the four data sets, which indicates the effectiveness of our proposed triplet loss. Additionally, τ also has a wide choice from 0.01 to 1, demonstrating its robustness.

Referring to the learning rate, we set it to 10^{-1}, 10^{-2}, 10^{-3}, 10^{-4}, and 10^{-5} for all the data sets, respectively, when the code length is 16-bit. Their corresponding results are shown in Table X. As we can see, the learning rate with a large value would result in divergence or fluctuation, while the learning rate with a tiny value would make the network converge too slow, which may also influence the performance. A typical example is the experiment on the CIFAR-10, in which too large or too small learning rates have the inferior influence on the performance. Thus, a suitable learning rate is 10^{-3}.

Table XI lists the results obtained by RADH when the batch size changes from 16 to 128. Note that the code length is set to 16-bit. It is easy to observe that our proposed method is robust on the batch size, which is similar to many deep learning methods [15], [16]. Thus, we empirically set it to 64 for all of the four data sets.

This article has been accepted for inclusion in a future issue of this journal. Content is final as presented, with the exception of pagination.

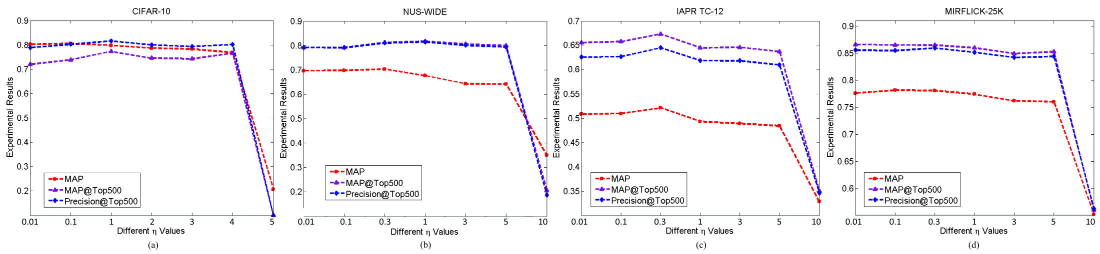


Fig. 8. MAP, MAP@Top500, and Precision@Top500 scores under the change of η. Results conducted on (a) CIFAR-10, (b) NUS-WIDE, (c) IAPR TC-12, and (d) MIRFLICK-25K.

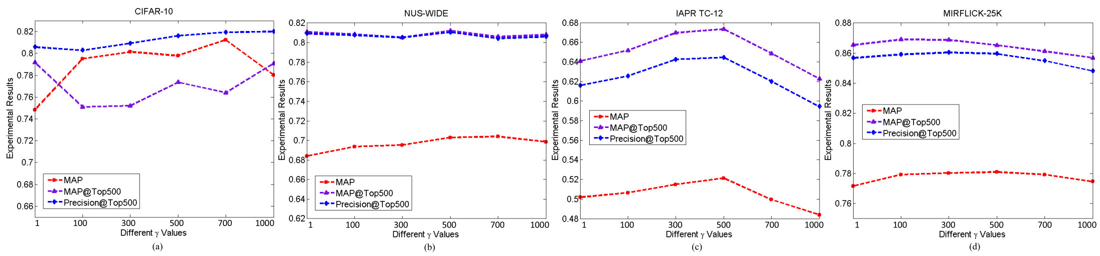


Fig. 9. MAP, MAP@Top500, and Precision@Top500 scores under the change of γ. Results conducted on (a) CIFAR-10, (b) NUS-WIDE, (c) IAPR TC-12, and (d) MIRFLICK-25K.

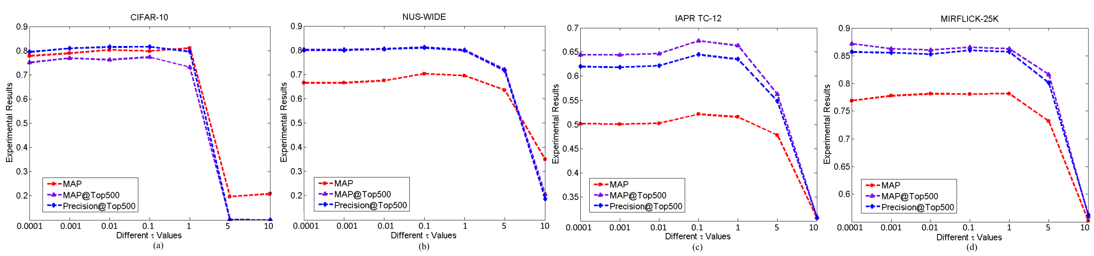


Fig. 10. MAP, MAP@Top500, and Precision@Top500 scores under the change of τ. Results conducted on (a) CIFAR-10, (b) NUS-WIDE, (c) IAPR TC-12, and (d) MIRFLICK-25K.

TABLE X

RESULTS OBTAINED BY RADH WITH DIFFERENT LEARNING RATES

Learning Rate	10^{-1}	10^{-2}	10^{-3}	10^{-4}	10^{-5}
Dataset	CIFAR-10				
MAP	20.79	18.15	79.79	79.10	60.12
MAP@Top500	10.00	10.17	77.36	73.76	69.55
Precision@Top500	10.00	10.09	81.62	79.54	69.58
Dataset	NUS-WIDE				
MAP	35.02	35.91	70.30	67.32	62.97
MAP@Top500	20.56	23.54	81.19	80.13	75.97
Precision@Top500	18.56	21.81	81.06	79.75	75.35
Dataset	IAPR TC-12				
MAP	30.65	52.29	52.13	50.06	46.83
MAP@Top500	30.79	68.30	67.34	64.51	60.03
Precision@Top500	30.71	64.80	64.45	61.95	57.33
Dataset	MIRFLICKR-25K				
MAP	55.18	55.18	78.09	77.21	76.68
MAP@Top500	55.95	55.95	86.52	86.30	85.16
Precision@Top500	56.20	56.20	85.96	85.42	84.49

TABLE XI

RESULTS OBTAINED BY RADH WITH DIFFERENT BATCH SIZES

Batchsize	16	32	48	64	128
Dataset	CIFAR-10				
MAP	79.41	77.97	80.74	79.79	79.78
MAP@Top500	76.25	78.01	77.08	77.36	73.66
Precision@Top500	80.76	81.47	81.80	81.62	79.88
Dataset	NUS-WIDE				
MAP	68.39	68.22	69.33	70.30	68.66
MAP@Top500	81.29	81.16	80.42	81.19	80.39
Precision@Top500	81.09	80.84	80.29	81.06	80.32
Dataset	IAPR TC-12				
MAP	51.95	50.18	52.27	52.13	50.56
MAP@Top500	68.34	65.24	67.73	67.34	65.74
Precision@Top500	65.20	62.47	64.86	64.45	62.98
Dataset	MIRFLICKR-25K				
MAP	78.65	77.59	78.09	78.09	77.82
MAP@Top500	88.39	86.68	86.88	86.52	86.46
Precision@Top500	86.88	86.11	85.70	85.96	85.50

Referring to updating the variable $\mathbf{B}$ in (21), in fact, there is an inner iteration when a column is updated while others are fixed. In the aforementioned experiments, we set this inner iteration to be 1. In fact, Huiskes and Lew [52] have proven that setting the iteration number to be 1 for the subproblem can also get similar solutions if the iteration for the whole problem is large enough. Here we have conducted additional experiments by setting this inner iteration number to be 10 and the results are tabulated in Table XII. Note the code length is set to 16-bit. It is easy to see that RADH obtains similar results compared with results in Tables II–IX, demonstrating the reasonability of the B updating strategy.

TABLE XII

RESULTS OBTAINED BY RADH WITH DIFFERENT INNER ITERATIONS IN UPDATING **B** COLUMN BY COLUMN. M@TOP500 AND P@TOP500 MEAN MAP@TOP500 AND PRECISION@TOP500, RESPECTIVELY. RADH-B DENOTES THE PROPOSED METHOD WITH TEN ITERATIONS IN UPDATING **B**

Metrics	MAP	M@Top500	P@Top500
Dataset	CIFAR-10		
RADH	79.79	77.36	81.62
RADH-B	80.61	76.99	80.53
Dataset	NUS-WIDE		
RADH	70.30	81.19	81.06
RADH-B	70.20	81.54	81.49
Dataset	IAPR TC-12		
RADH	52.13	67.34	64.45
RADH-B	51.64	66.47	63.61
Dataset	MIRFLICKR-25K		
RADH	78.09	86.52	85.96
RADH-B	77.56	87.63	86.51

V. CONCLUSION

In this article, an RADH method is proposed. Different from the most existing methods that achieve the matching between each pair of instances through a point-to-point way, we relax it to a point-to-angle way. Specifically, there is not any length constraint on the learned real-valued features, while the asymmetric strategy encourages the discrete hashing variables and real-valued features to be located in the same Hamming space if they share the same semantic information. In addition, to further make a good ranking for the positive and negative pairs, a novel triplet loss is proposed, which is quite adaptive for the hashing learning. Experiments conducted on four real-world data sets demonstrate the superiority of our proposed approach.

REFERENCES

[1] F. Shen, Y. Xu, L. Liu, Y. Yang, Z. Huang, and H. T. Shen, "Unsupervised deep hashing with similarity-adaptive and discrete optimization," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 40, no. 12, pp. 3034–3044, Dec. 2018.

[2] C. Ma, I. W. Tsang, F. Shen, and C. Liu, "Error correcting input and output hashing," *IEEE Trans. Cybern.*, vol. 49, no. 3, pp. 781–791, Mar. 2018.

[3] Y. Cao, M. Long, J. Wang, and S. Liu, "Collective deep quantization for efficient cross-modal retrieval," in *Proc. AAAI*, 2017, pp. 3974–3980.

[4] F. Shen *et al.*, "Classification by retrieval: Binarizing data and classifiers," in *Proc. ACM SIGIR*, 2017, pp. 595–604.

[5] D. Tao, X. Li, X. Wu, and S. J. Maybank, "General tensor discriminant analysis and Gabor features for gait recognition," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 29, no. 10, pp. 1700–1715, Oct. 2007.

[6] J. Sanchez and F. Perronnin, "High-dimensional signature compression for large-scale image classification," in *Proc. IEEE CVPR*, Jun. 2011, pp. 1665–1672.

[7] J. Li, B. Zhang, and D. Zhang, "Shared autoencoder Gaussian process latent variable model for visual classification," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 29, no. 9, pp. 4272–4286, Sep. 2017.

[8] A. Gionis, P. Indyk, and R. Motwani, "Similarity search in high dimensions via hashing," *VLDB*, vol. 99, no. 6, pp. 518–529, 1999.

[9] W. Liu, J. Wang, S. Kumar, and S.-F. Chang, "Hashing with graphs," in *Proc. ICML*, 2011, pp. 1–8.

[10] Y. Weiss, A. Torralba, and R. Fergus, "Spectral hashing," in *Proc. NIPS*, 2009, pp. 1753–1760.

[11] W. Liu, C. Mu, S. Kumar, and S.-F. Chang, "Discrete graph hashing," in *Proc. NIPS*, 2014, pp. 3419–3427.

[12] J.-P. Heo, Y. Lee, J. He, S.-F. Chang, and S.-E. Yoon, "Spherical hashing," in *Proc. IEEE CVPR*, Jun. 2012, pp. 2957–2964.

[13] G. Lin, C. Shen, Q. Shi, A. Van den Hengel, and D. Suter, "Fast supervised hashing with decision trees for high-dimensional data," in *Proc. IEEE CVPR*, Jun. 2014, pp. 1963–1970.

[14] F. Shen, C. Shen, W. Liu, and H. T. Shen, "Supervised discrete hashing," in *Proc. IEEE CVPR*, Jun. 2015, pp. 37–45.

[15] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Proc. Adv. Neural Inf. Process. Syst. (NIPS)*, 2012, pp. 1097–1105.

[16] J. Li, B. Zhang, G. Lu, and D. Zhang, "Dual asymmetric deep hashing learning," *IEEE Access*, vol. 7, pp. 113372–113384, 2019.

[17] W.-J. Li, S. Wang, and W.-C. Kang, "Feature learning based deep supervised hashing with pairwise labels," 2015, *arXiv:1511.03855*. [Online]. Available: https://arxiv.org/abs/1511.03855

[18] J. Wang, T. Zhang, J. Song, N. Sebe, and H. T. Shen, "A survey on learning to hash," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 40, no. 4, pp. 769–790, Apr. 2018.

[19] Q.-Y. Jiang and W.-J. Li, "Asymmetric deep supervised hashing," 2017, *arXiv:1707.08325*. [Online]. Available: https://www.aaai.org/ocs/index.php/AAAI/AAAI18/paper/viewPaper/17296

[20] Y. Cao, M. Long, J. Wang, and P. S. Yu, "Correlation hashing network for efficient cross-modal retrieval," 2016, *arXiv:1602.06697*. [Online]. Available: https://arxiv.org/abs/1602.06697

[21] Y. He, S. Xiang, C. Kang, J. Wang, and C. Pan, "Cross-modal retrieval via deep and bidirectional representation learning," *IEEE Trans. Multimedia*, vol. 18, no. 7, pp. 1363–1377, Jul. 2016.

[22] B. Kulis and K. Grauman, "Kernelized locality-sensitive hashing for scalable image search," in *Proc. IEEE ICCV*, Sep./Oct. 2009, pp. 2130–2137.

[23] A. Z. Broder, "On the resemblance and containment of documents," in *Proc. Compress. Complex. SEQUENCES*, Jun. 1997, pp. 21–29.

[24] M. S. Charikar, "Similarity estimation techniques from rounding algorithms," in *Proc. Annu. ACM Symp. Theory Comput.*, 2002, pp. 380–388.

[25] M. Datar, N. Immorlica, P. Indyk, and V. S. Mirrokni, "Locality-sensitive hashing scheme based on p-stable distributions," in *Proc. Annu. Symp. Comput. Geometry*, 2004, pp. 253–262.

[26] Z. Jin, C. Li, Y. Lin, and D. Cai, "Density sensitive hashing," *IEEE Trans. Cybern.*, vol. 44, no. 8, pp. 1362–1371, Aug. 2014.

[27] Q.-Y. Jiang and W.-J. Li, "Scalable graph hashing with feature transformation," in *Proc. IJCAI*, 2015, pp. 2248–2254.

[28] Y. Gong, S. Lazebnik, A. Gordo, and F. Perronnin, "Iterative quantization: A procrustean approach to learning binary codes for large-scale image retrieval," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 35, no. 12, pp. 2916–2929, Dec. 2013.

[29] W. Kong and W.-J. Li, "Double-bit quantization for hashing," in *Proc. Conf. Artif. Intell. (AAAI)*, 2012, vol. 1, no. 2, p. 5.

[30] M. Hu, Y. Yang, F. Shen, N. Xie, and H. T. Shen, "Hashing with angular reconstructive embeddings," *IEEE Trans. Image Process.*, vol. 27, no. 2, pp. 545–555, Feb. 2018.

[31] W. Liu, J. Wang, R. Ji, Y.-G. Jiang, and S.-F. Chang, "Supervised hashing with kernels," in *Proc. IEEE CVPR*, Jun. 2012, pp. 2074–2081.

[32] W.-C. Kang, W.-J. Li, and Z.-H. Zhou, "Column sampling based discrete supervised hashing," in *Proc. AAAI*, 2016, pp. 1230–1236.

[33] Y. Luo, Y. Yang, F. Shen, Z. Huang, P. Zhou, and H. T. Shen, "Robust discrete code modeling for supervised hashing," *Pattern Recognit.*, vol. 75, pp. 128–135, Mar. 2018.

[34] V. E. Liong, J. Lu, G. Wang, P. Moulin, and J. Zhou, "Deep hashing for compact binary codes learning," in *Proc. CVPR*, Jun. 2015, pp. 2475–2483.

[35] 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 Trans. Image Process.*, vol. 24, no. 12, pp. 4766–4779, Dec. 2015.

[36] H. Zhu, M. Long, J. Wang, and Y. Cao, "Deep hashing network for efficient similarity retrieval," in *Proc. AAAI*, 2016, pp. 2415–2421.

[37] H. Liu, R. Wang, S. Shan, and X. Chen, "Deep supervised hashing for fast image retrieval," in *Proc. CVPR*, Jun. 2016, pp. 2064–2072.

[38] R. Xia, Y. Pan, H. Lai, C. Liu, and S. Yan, "Supervised hashing for image retrieval via image representation learning," *Assoc. Adv. Artif. Intell.*, vol. 1, no. 1, pp. 2156–2162, 2014.

[39] F. Shen, X. Gao, L. Liu, Y. Yang, and H. T. Shen, "Deep asymmetric pairwise hashing," in *Proc. 25th ACM Int. Conf. Multimedia*, 2017, pp. 1522–1530.

[40] F. Shen, Y. Yang, L. Liu, W. Liu, D. Tao, and H. T. Shen, "Asymmetric binary coding for image search," *IEEE Trans. Multimedia*, vol. 19, no. 9, pp. 2022–2032, Sep. 2017.

[41] F. Zhao, Y. Huang, L. Wang, and T. Tan, “Deep semantic ranking based hashing for multi-label image retrieval,” in *Proc. CVPR*, Jun. 2015, pp. 1556–1564.

[42] Y. Gu, H. Zhang, Z. Zhang, and Q. Ye, “Unsupervised deep triplet hashing with pseudo triplets for scalable image retrieval,” in *Multimedia Tools and Applications*. Amsterdam, The Netherlands: Springer, 2019, pp. 1–22.

[43] J.-P. Heo, Y. Lee, J. He, S.-F. Chang, and S.-E. Yoon, “Spherical Hashing: Binary Code Embedding with Hyperspheres,” *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 37, no. 11, pp. 2304–2316, Nov. 2015.

[44] A. Gordo, F. Perronnin, Y. Gong, and S. Lazebnik, “Asymmetric distances for binary embeddings,” *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 36, no. 1, pp. 33–47, Jan. 2014.

[45] K. Chatfield, K. Simonyan, A. Vedaldi, and A. Zisserman, “Return of the devil in the details: Delving deep into convolutional nets,” 2014, *arXiv:1405.3531*. [Online]. Available: https://arxiv.org/abs/1405.3531

[46] A. Vedaldi and K. Lenc, “Matconvnet: Convolutional neural networks for MATLAB,” in *Proc. ACMMM*, 2015, pp. 689–692.

[47] A. Krizhevsky and G. Hinton, “Learning multiple layers of features from tiny images,” Univ. Toronto, Toronto, ON, USA, Tech. Rep., 2009.

[48] T.-S. Chua, J. Tang, R. Hong, H. Li, Z. Luo, and Y. Zheng, “Nus-wide: A real-world Web image database from National University of Singapore,” in *Proc. ACM Int. Conf. Image Video Retr.*, 2009, p. 48.

[49] H. J. Escalante *et al.*, “The segmented and annotated IAPR TC-12 benchmark,” *Comput. Vis. Image Understand.*, vol. 114, no. 4, pp. 419–428, 2010.

[50] M. J. Huiskes and M. S. Lew, “The mir flickr retrieval evaluation,” in *Proc. ACMMIR*, 2008, pp. 39–43.

[51] Q.-Y. Jiang and W.-J. Li, “Deep cross-modal hashing,” 2016, *arXiv:1602.02255*. [Online]. Available: http://openaccess.thecvf.com/content_cvpr_2017/html/Jiang_Deep_Cross-Modal_Hashing_CVPR_2017_paper.html

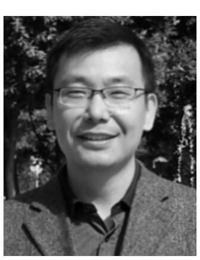
[52] E. J. Candès, X. Li, Y. Ma, and J. Wright, “Robust principal component analysis?” *J. ACM*, vol. 58, no. 3, p. 11, May 2011.

Jinxing Li received the B.Sc. degree from the Department of Automation, Hangzhou Dianzi University, Hangzhou, China, in 2012, the M.Sc. degree from the Department of Automation, Chongqing University, Chongqing, China, in 2015, and the Ph.D. degree from the Department of Computing, The Hong Kong Polytechnic University, Hong Kong, in 2018.

His is currently a Post-Doctoral with The Chinese University of Hong Kong (Shenzhen), Shenzhen, China. His research interests are pattern recognition, deep learning, medical biometrics, and machine learning.

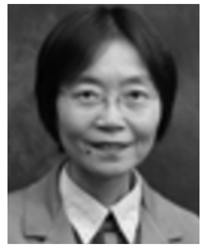
Bob Zhang (M’12) received the B.A. degree in computer science from York University, Toronto, ON, USA, in 2006, the M.A.Sc. degree in information systems security from Concordia University, Montreal, Quebec, Canada, in 2007, and the Ph.D. degree in electrical and computer engineering from the University of Waterloo, Waterloo, ON, Canada, in 2011.

He is currently an Assistant Professor with the Department of Computer and Information Science, University of Macau, Macau, China. His research interests focus on medical biometrics, pattern recognition, and image processing.



Guangming Lu received the B.S. degree in electrical engineering, the M.S. degree in control theory and control engineering, and the Ph.D. degree in computer science and engineering from the Harbin Institute of Technology (HIT), Harbin, China, in 1998, 2000, and 2005, respectively.

He is currently a Professor with the Biocomputing Research Center, Shenzhen Graduate School, HIT, Shenzhen, China. His current research interests include pattern recognition, image processing, and automated biometric technologies and applications.



Jane You received the Ph.D. degree from La Trobe University, Melbourne, VIC, Australia, in 1992.

She is currently a Professor with the Department of Computing, The Hong Kong Polytechnic University, Hong Kong. Her current research interests include image processing, medical imaging, computer-aided diagnosis, and pattern recognition.

Yong Xu (M’06–SM’15) was born in Sichuan, China, in 1972. He received the B.S. and M.S. degrees with the Air Force Institute of Meteorology, Nanjing, China, in 1994 and 1997, respectively, and the Ph.D. degree in pattern recognition and intelligence system from the Nanjing University of Science and Technology, Nanjing, in 2005.

He is currently with the Shenzhen Graduate School, Harbin Institute of Technology, Harbin, China. His current interests include pattern recognition, biometrics, machine learning, and video analysis.

Feng Wu (M’99–SM’06–F’13) received the B.S. degree in electrical engineering from Xidian University, Xi’an, China, in 1992, and the M.S. and Ph.D. degrees in computer science from the Harbin Institute of Technology, Harbin, China, in 1996 and 1999, respectively.

He was a Principle Researcher and a Research Manager with Microsoft Research Asia, Beijing, China. He is currently a Professor with the University of Science and Technology of China, Hefei, China, where he is also the Dean of the School of Information Science and Technology. He has authored or coauthored over 200 high-quality articles (including several dozens of IEEE Transaction articles) and top conference papers on MOBICOM, SIGIR, CVPR, and ACM MM. He has 77 granted U.S. patents. His 15 techniques have been adopted into international video coding standards. As a coauthor, he received the best paper award in the IEEE T-CSVT 2009, PCM 2008, and SPIE VCIP 2007. His research interests include image and video compression, media communication, and media analysis and synthesis.

Dr. Wu received the IEEE Circuits and Systems Society 2012 Best Associate Editor Award. He serves as an Associate Editor for the IEEE TRANSACTIONS ON CIRCUITS AND SYSTEM FOR VIDEO TECHNOLOGY, IEEE TRANSACTIONS ON MULTIMEDIA, and several other International journals.

David Zhang (F’08) received the Graduation degree in computer science from Peking University, Beijing, China, the M.Sc. degree in computer science and the Ph.D. degree from the Harbin Institute of Technology (HIT), Harbin, China, in 1982 and 1985, respectively, and the second Ph.D. degree in electrical and computer engineering from the University of Waterloo, Waterloo, ON, Canada, in 1994.

He is currently a Chair Professor with The Hong Kong Polytechnic University, Hong Kong, and The Chinese University of Hong Kong (Shenzhen), Shenzhen, China. He also serves as a Visiting Chair Professor with Tsinghua University, Beijing, China, and an Adjunct Professor with Peking University, Beijing, China, Shanghai Jiao Tong University, Shanghai, China, HIT, and the University of Waterloo. He has authored or coauthored about 20 monographs, over 400 international journal articles, and around 40 patents from USA/Japan/HK/China. His research interests are medical biometrics and pattern recognition.

Dr. Zhang is a fellow of IAPR. He is the Associate Editor of more than ten international journals, including the IEEE TRANSACTIONS and so on. He was selected as a Highly Cited Researcher in Engineering by Thomson Reuters in 2014, 2015, and 2016, respectively.