Hindawi
Computational Intelligence and Neuroscience
Volume 2019, Article ID 8490364, 17 pages
https://doi.org/10.1155/2019/8490364

Research Article

Triplet Deep Hashing with Joint Supervised Loss Based on Deep Neural Networks

Mingyong Li, Ziye An, Qinmin Wei, Kaiyue Xiang, and Yan Ma

College of Computer and Information Science, Chongqing Normal University, Chongqing 401331, China

Correspondence should be addressed to Yan Ma; cqnu_mayan@163.com

Received 26 April 2019; Revised 8 July 2019; Accepted 14 July 2019; Published 9 October 2019

Academic Editor: Michele Migliore

In recent years, with the explosion of multimedia data from search engines, social media, and e-commerce platforms, there is an urgent need for fast retrieval methods for massive big data. Hashing is widely used in large-scale and high-dimensional data search because of its low storage cost and fast query speed. Thanks to the great success of deep learning in many fields, the deep learning method has been introduced into hashing retrieval, and it uses a deep neural network to learn image features and hash codes simultaneously. Compared with the traditional hashing methods, it has better performance. However, existing deep hashing methods have some limitations; for example, most methods consider only one kind of supervised loss, which leads to insufficient utilization of supervised information. To address this issue, we proposed a triplet deep hashing method with joint supervised loss based on the convolutional neural network (JLTDH) in this work. The proposed method JLTDH combines triplet likelihood loss and linear classification loss; moreover, the triplet supervised label is adopted, which contains richer supervised information than that of the pointwise and pairwise labels. At the same time, in order to overcome the cubic increase in the number of triplets and make triplet training more effective, we adopt a novel triplet selection method. The whole process is divided into two stages: In the first stage, taking the triplets generated by the triplet selection method as the input of the CNN, the three CNNs with shared weights are used for image feature learning, and the last layer of the network outputs a preliminary hash code. In the second stage, relying on the hash code of the first stage and the joint loss function, the neural network model is further optimized so that the generated hash code has higher query precision. We perform extensive experiments on the three public benchmark datasets CIFAR-10, NUS-WIDE, and MS-COCO. Experimental results demonstrate that the proposed method outperforms the compared methods, and the method is also superior to all previous deep hashing methods based on the triplet label.

1. Introduction

In recent years, because of the explosive growth of Internet big data, the Internet is filled with a large number of multimedia resources, including pictures, videos, and text, and there is an urgent need for fast search methods for massive big data. Approximate nearest neighbor (ANN) [1] search has been widely used in many fields such as image retrieval, computer vision, and data mining. Because of speed and low memory cost, hashing has become an important branch of ANN search, which is one of the widely used techniques in image retrieval [2–9]. Hashing techniques encode images, documents, videos, or other types of data in a short set of binary codes while keeping the original data similar. The hashing method produces binary encodings that make the nearest neighbor search of large datasets easy.

A series of different hashing methods are proposed to implement efficient ANN search using Hamming distance [3, 5, 8, 9]. More recently, deep hashing methods [10, 11, 12, 13, 14] show that image representation and hash coding can be learned more effectively using deep neural networks, resulting in state-of-the-art results on many benchmark datasets.

Recently, triplet loss has been studied for computer vision problems. The triplet labels contain richer information than pairwise labels. Each triplet label can be naturally decomposed into two pairwise labels. In particular, a triplet label ensures that the query image is close to the positive image and far away from the negative image in the learning hash code space. However, a pairwise label only ensures that one constraint is observed. The retrieval performance of triplet loss is better than that of pointwise and pairwise losses. Therefore, the triplet likelihood loss is introduced in this paper.

At the same time, the existing deep hashing methods still have some shortcomings in the utilization of classification information. The classification information only plays a role in deep neural network image representation but has no direct impact on the optimization of the hash function. Therefore, this paper proposes a linear classification loss to deal with this situation.

Therefore, combining triplet likelihood loss and linear classification loss, we propose a triplet deep hashing method with joint supervised loss based on the convolutional neural network (JLTDH). The supervised information used in this method is in the form of triplet labels. For the sake of fully utilizing the triplet information and the classification information, we propose a joint loss function, which consists of two parts: the triplet negative log-likelihood loss and the linear classification loss. Depending on this joint loss function, the hash codes can be further optimized by the linear classifier. The linear classifier indicates the relationship between the label information and the hash codes. We choose the convolutional neural network (CNN) as our deep learning model, for example, AlexNet, ResNet, and VGG, which can learn image representation and hash function at the same time. The whole process is divided into two stages: In the first stage, taking the triplets generated by the triplet selection method as the input of the CNN, the three CNNs with shared weights are used for image feature learning, and the last layer of the network outputs a preliminary hash code. In the second stage, relying on the hash code of the first stage and the joint loss function, the neural network model is further optimized so that the generated hash code has higher query precision.

This work is summarized as follows:

(1) In this paper, a triplet deep hashing method with negative log-likelihood is proposed, and the method performs both image feature representation and hash code learning in a convolutional neural network. In order to overcome the cubic increase in the number of triplets and make triplet training more effective, we adopt a novel triplet selection method.

(2) To fully utilize the supervised triplet information, JLTDH proposed a joint loss function combining the triplet negative log-likelihood loss and the linear classification loss. Relying on the hash code of the first stage and the joint loss function, the neural network model is further optimized so that the generated hash code has higher query precision.

(3) We perform extensive experiments on the three public benchmark datasets CIFAR-10, NUS-WIDE, and MS-COCO. Experimental results demonstrate that the proposed method outperforms the compared methods, and the method is also superior to all previous deep hashing methods based on the triplet label.

2. Related Works

Hashing methods are divided into data-independent methods and data-dependent methods. Locality-sensitive hashing (LSH) [1] is a typical representative of data-independent hashing methods among various hashing techniques; the hash function of this method is generated by a random map, and random projection maps data points from the original space to the Hamming space; in this process, the training data are not used to learn hash functions. One drawback of LSH is that LSH usually requires a very long bit length, which leads to huge storage overhead. Data-dependent methods [15] learn the data feature from the training data so as to learn the hash function, which are also known as learning to hashing (L2H) methods. Compared to the data-independent method, the L2H method can get better accuracy with a shorter hash code. Therefore, the L2H method is more widely used than the data-independent method in the practical application.

The L2H methods include two types: unsupervised hashing and supervised hashing [4, 16]. Unsupervised hashing does not use supervised information for hash function learning, and the purpose of unsupervised hashing is to preserve the metric structure in the training data. Typical unsupervised approaches include iterative quantization (ITQ) [3], isotropic hashing (IsoHash) [4], discrete graph hashing (DGH) [15], and scalable graph hashing (SGH) [17]. Unsupervised hashing often fails to achieve satisfactory retrieval performance in practical applications.

On the contrary, supervised hashing tries to learn the hash function by utilizing supervised information. The purpose of supervised hashing is to map the data points in the original space to the Hamming space with supervised information, and supervised information optimizes the learning of hash function so as to learn better hash code. In recent years, using supervised information for learning, supervised hashing has higher precision than unsupervised hashing, so more and more researchers have studied it deeply [5, 12, 18]. Typical supervised hashing methods include supervised hashing with kernels (KSH) [5], fast supervised hashing (FastH) [8], supervised discrete hashing (SDH) [9], latent factor hashing (LFH) [19], adaptive binary quantization (ABQ) [20], hash bit selection [21], and structure-sensitive hashing (SSH) [10]. ABQ [20] jointly pursues a set of prototypes in the original space and a subset of binary codes in the Hamming space. The prototypes and the codes are correspondingly associated and together define the hash function for small hash codes. Hash bit selection [21] presented two related selection methods via dynamic programming and quadratic programming, incorporating bit reliability and complementarity. SSH [10] simultaneously captures the two types of structures among data in an alternative way. It learns discriminative hash functions that quantize data into the cluster prototypes associated with unique binary codes.

However, most traditional supervised hashing methods cannot extract features very well. In recent years, researchers have proposed deep learning hashing methods, which can effectively extract image features to identify similar images, and their performance is better than that of the traditional hashing method. Representative deep hashing methods including convolutional neural network-based hashing (CNNH) [18] adopt a two-stage strategy: learning binary hash codes in the first stage and learning a deep-network-based hash function to fit the codes in the second stage. DNNH [12] improved CNNH with a simultaneous feature learning and hash coding pipeline such that deep representations and hash codes are optimized by the triplet loss. The deep hashing network (DHN) [14] improves DNNH by jointly preserving the pairwise semantic similarity and controlling the quantization error by simultaneously optimizing the pairwise cross-entropy loss and quantization loss via a multitask approach. Other typical deep hashing methods include deep pairwise supervised hashing (DPSH) [13] and deep supervised hashing (DSH) [22].

Recently, some new deep hashing methods emerged, such as cross-modal hashing and hashing-based generative adversarial network. Representative methods including progressive generative hashing (PGH) [23] learn a discriminative hashing network in an unsupervised way, which exploits the power of hash-conditioned GANs and progressive learning. Triplet-based deep hashing (TDH) [24] is used for cross-modal retrieval, and triplet labels are exploited as supervised information to capture relative semantic correlation between heterogeneous data from different modalities. UCH [25] is a cross-modal retrieval method, where the outer-cycle network is used to learn powerful common representation and the inner-cycle network is explained to generate reliable hash codes. Deep learning hashing methods can significantly outperform nondeep supervised hashing in many applications, so we focused on deep hashing.

3. The Framework of JLTDH

3.1. Problem Definition. In our proposed method JLTDH, the input of the convolutional neural network is triplets. We denote the image triplet set as $T = \{t_i\}_{i=1}^{m} = \{(t_i^a, t_i^p, t_i^n)\}_{i=1}^{m}$; in each triplet of the image $t_i = (t_i^a, t_i^p, t_i^n)$, t_i^a, t_i^p, and t_i^n indicate, respectively, the anchor point, the positive point, and the negative point; the positive image t_i^p is defined as $s_{ap} = 1$ (t_i^p and t_i^a are similar and belong to the same category), and the negative image t_i^n is defined as $s_{an} = 0$ (t_i^n and t_i^a are dissimilar and belong to different categories). The distance between t_i^a and t_i^p is smaller than the distance between t_i^a and t_i^n.

Our goal is to learn the hash codes $B = \{b_i\}_{i=1}^{n} \in \{-1, 1\}^{L\times n}$ for the image X; the similarity between the two hash codes is calculated using the Hamming distance. The hash codes B should satisfy all the triplet labels T in the Hamming space as much as possible; for triplet labels, $\mathrm{dis}H(b_{t_i^a}, b_{t_i^p})$ should be as small as possible as $\mathrm{dis}H(b_{t_i^a}, b_{t_i^n})$, where $\mathrm{dis}H()$ represents the Hamming distance between two hash codes.

3.2. Framework. We introduce the proposed framework of JLTDH; this is an end-to-end hash learning framework based on the convolutional neural network.

As shown in Figure 1, we proposed a triplet deep hashing with joint supervised loss, which is a deep learning framework capable of both automatic feature learning and hash coding, and it joins triplet deep learning and linear classification quantization. This is an end-to-end approach and supervised by triplet labels, which contains three main components: image feature learning part, hash code learning part, and joint loss function part. It integrates these three components into the same end-to-end framework.

We generally generate triplets based on the category information of the sample, select the anchor image and positive image from the same category, and select the negative image from different categories. However, as the dataset increases, the number of all possible triplets is very large; using all triplets is computationally difficult and not optimal, and at the same time, it is not helpful for training and will lead to slow convergence of training. The existing triplet hashing method does not solve the problem of triplet selection very well. Therefore, the mining and selection of triplets is an urgent problem to be solved. We adopt a novel triplet selection method, which will be discussed in detail in Section 4.

3.2.1. Image Feature Learning Part. In this part, we use three CNNs with shared weights to extract the appropriate feature representation for binary hash code learning. We use the AlexNet [26] network architecture for this part, and VGG [27] and ResNet [28] can also apply to this part as well. Each CNN contains five convolutional layers and three fully connected layers. The last layer of AlexNet is replaced with the FC (fully connected) layer, and the output of the last layer is projected as a hash code. The configuration of AlexNet is shown in Table 1.

3.2.2. Hash Code Learning Part. This part generates the hash code of the image according to the image features of the previous part. The FC layer uses the sign functions as activation functions. Binary code is obtained by using the sign function. The length of the hash code c is determined by the number of FC-layer neurons in the last layer.

3.2.3. Joint Loss Function Part. The joint loss function combines two kinds of supervised loss functions: the triplet negative log-likelihood loss and the linear classification loss, and is designed to further optimize the hash code so that the hash code and classification information can maintain the semantic relationship between points. The joint loss function is the weighted combination of triplet label likelihood loss and linear classification loss.

4. Triplet Selection Method

In large datasets like NUS-WIDE and MS-COCO, the number of all possible triplets is very large. Thus, using all triplets is computationally difficult and not optimal.

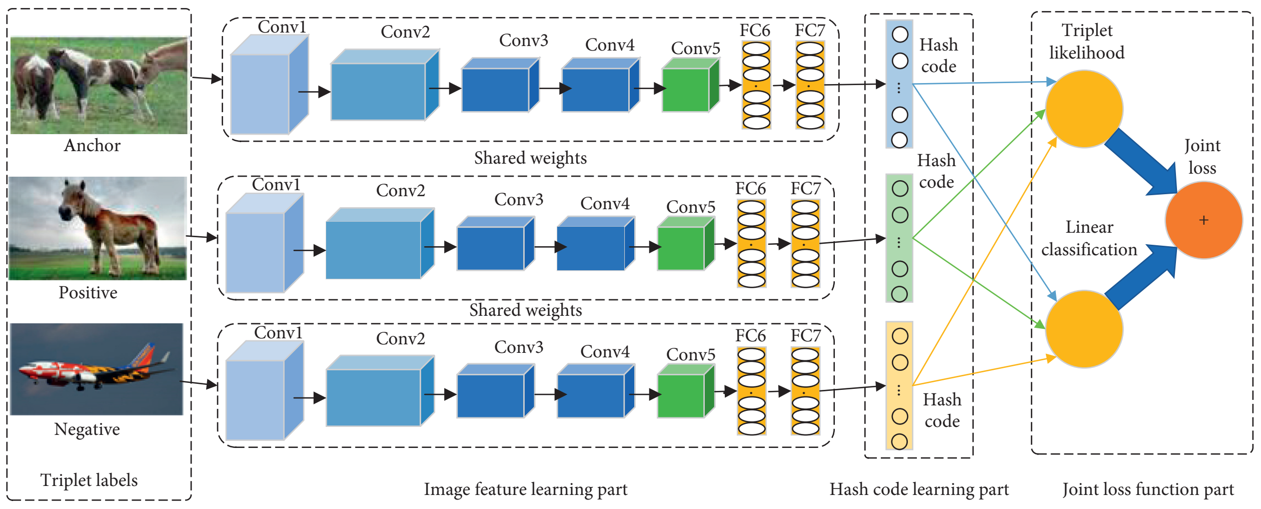


Figure 1: Framework of JLTDH.

Table 1: Configuration of AlexNet in our method.

Layer	Configuration				
	#Filter	Filter size	Stride	Padding	Pooling
Conv1	64	11×11	4×4	2×2	3×3
Conv2	192	5×5	1×1	2×2	3×3
Conv3	384	3×3	1×1	1×1	—
Conv4	256	3×3	1×1	1×1	—
Conv5	256	3×3	1×1	1×1	3×3
Full6	9216				
Full7	4096				
Full8	Hash code length c				

Table 2: Number of triplets using the proposed triplet selection method.

Dataset	Training dataset	Group	Category	Selected triplets	Time
CIFAR-10	5000	20	10	124,000	86 s
NUS-WIDE	10,500	20	21	1,372,000	316 s
MS-COCO	10,000	20	80	1,719,000	492 s

Specifically, it is not helpful for training and will lead to slow convergence of training. For example, the dataset MS-COCO is used in this paper, whose training dataset contains 10,000 images, and the number of all possible triplets is approximately $(1.0\times10^4)^3 = 1.0\times10^{12}$. This is a very large number which is very difficult to calculate.

Inspired by the study in [29], we adopt a novel triplet selection method to reduce the computational cost. Randomly splitting the training data into several groups $\{G_i\}_{i=1}^m$, the triplets are selected only within groups $G_i^T = \{G_i^a, G_i^p, G_i^n\}$, where G_i^a, G_i^p, and G_i^n, respectively, represent the anchor points, positive points, and negative points in the i-th group. $G_i^p = \{p \in G_i\colon\ p \neq a, s_{pa} = 1\}$ is the group of positive samples consisting of the samples similar to the anchor point G_i^a in the i-th group. We randomly chose negative samples from the group of negative samples $G_i^n = \{n \in G_i\colon\ \alpha - \operatorname{dist}(a,n) + \operatorname{dist}(a,p) > 0, s_{an} = 0\}$, and we found that negative points far away from the anchor point were not helpful for training, so we excluded these negative points.

Using the proposed triplet selection method, we find that the number of triplets is much smaller than the number of possible triplets in our dataset. The specific results are shown in Table 2. The CPU running the code of the triplet selection method is Intel Xeon CPU E5-2687W @ 3.0 GHz with 12 cores, and the RAM is 32 GB. The time cost of the triplet selection method on three datasets is low and acceptable.

5. Joint Loss Function

The joint loss function combines two kinds of supervised loss functions: the triplet negative log-likelihood loss and the linear classification loss. We introduce them in the following.

5.1. *Triplet Negative Log-Likelihood Loss.*

The Hamming distance between two strings of equal length is the number of positions at which the corresponding symbols are different. It is often used to calculate the similarity between two hash codes. In other words, the smaller the Hamming distance between two hash codes, the more similar they are, and vice versa. $\mathrm{dist}H(b_i, b_j)$ indicates the Hamming distance between b_i and b_j, which can be calculated by inner product $\langle b_i, b_j\rangle$: $\mathrm{dist}H(b_i, b_j) = 1/2(L - \langle b_i, b_j\rangle)$, where L is generally the length of the hash code. From the above equation, it can be concluded that the larger the inner product of the two hash codes, the smaller the Hamming distance and the more similar they are. Let Ω_{ij} represent half of the inner product between two binary codes $b_i, b_j \in \{-1, 1\}^L$: $\Omega_{ij} = 1/2 b_i^T b_j$. DPSH [13] is a method for simultaneously learning feature representation and hash codes using pairwise label likelihood function as the loss function, and the likelihood function is formulated as

$$p(S \mid B) = \prod_{s_{ij}\in S} p\left(s_{ij} | B\right), \tag{1}$$

where $p(s_{ij} \mid B) = \begin{cases} \varphi(\Omega_{ij}), & s_{ij} = 1, \\ 1 - \varphi(\Omega_{ij}), & s_{ij} = 0, \end{cases}$ B is the set of all hash codes and $\varphi(x)$ is the sigmoid function: $\varphi(x) = 1/1 + e^{-x}$. The two hash codes with the large inner product should have high similarity.

The supervised information used by DPSH [13] is a pairwise label. Inspired by DPSH, this paper proposes a hashing method using triplet label likelihood function, which is a deep hashing method using a deep neural network. A set of triplet labels $T = \{t_i\}_{i=1}^m = \left\{(t_i^a, t_i^p, t_i^n)\right\}_{i=1}^m$ is given. Suppose the conditions are independent of each other, according to naive Bayes' theorem, with some prior $p(B)$, then the posterior probability of B can be computed as follows:

$$p(B \mid T) \sim p(T \mid B) p(B) = \prod_{t_i\in T} p(t_i \mid B) p(B). \tag{2}$$

We can define the triplet label likelihood function $p(T \mid B)$ as follows:

$$p(T \mid B) = \prod_{t_i\in T} p(t_i \mid B) = \prod_{t_i\in T} p\left(\left(t_i^a, t_i^p, t_i^n\right) \mid B\right). \tag{3}$$

We can learn the optimal B through maximum a posteriori estimation.

In order to solve $p((t_i^a, t_i^p, t_i^n) \mid B)$, according to $\mathrm{dist}H(b_i, b_j) = 1/2(L - 2\Omega_{ij})$, we can get

$$\mathrm{dis}H\left(b_{t_i^a}, b_{t_i^p}\right) - \mathrm{dis}H\left(b_{t_i^a}, b_{t_i^n}\right) = -\left(\Omega_{t_i^a t_i^p} - \Omega_{t_i^a t_i^n}\right). \tag{4}$$

From equation (4), we can conclude that the smaller the distance between the anchor point and the positive point, and the greater the distance between the anchor point and the negative point, the larger the value of $\Omega_{t_i^a t_i^p} - \Omega_{t_i^a t_i^n}$.

According to equation (4), we can make the following definition:

$$p\left(\left(t_i^a, t_i^p, t_i^n\right) \mid B\right) = \varphi\left(\Omega_{t_i^a t_i^p} - \Omega_{t_i^a t_i^n} - \beta\right), \tag{5}$$

where β is a superparameter whose role is to adjust the gap between $\mathrm{dis}H(b_{t_i^a}, b_{t_i^p})$ and $\mathrm{dis}H(b_{t_i^a}, b_{t_i^n})$. When $\mathrm{dis}H(b_{t_i^a}, b_{t_i^n}) \geq \mathrm{dis}H(b_{t_i^a}, b_{t_i^p}) + \beta$, we can conclude that $\Omega_{t_i^a t_i^p} - \Omega_{t_i^a t_i^n} - \beta \geq 0$ and $p((t_i^a, t_i^p, t_i^n) \mid B) \geq 0.5$. The smaller the distance between the anchor point and the positive point, and the greater the distance between the anchor point and the negative point, the larger the value of the triplet likelihood function, and vice versa. This is consistent with the intention of our objective function.

By maximizing the triplet likelihood $p(T \mid B)$, we can enforce the Hamming distance between the anchor image and the positive image to be smaller than that between the anchor image and the negative image. Taking the negative log-likelihood of the triplet label, the following optimization problem can be obtained:

$$\min_B J_1 = -\log p(T \mid B) = -\sum_{t_i\in T} \log p\left(\left(t_i^a, t_i^p, t_i^n\right) \mid B\right). \tag{6}$$

The above optimization problem can minimize the Hamming distance between the anchor point and the positive point and simultaneously maximize the Hamming distance between the anchor point and the negative point. This exactly matches the goal of supervised hashing with the triplet label.

Substituting formula (5) into formula (6), the following formula can be obtained:

$$\begin{aligned}\min_B J_1 &= -\log p(T \mid B) = -\sum_{t_i\in T} \log \varphi\left(\Omega_{t_i^a t_i^p} - \Omega_{t_i^a t_i^n} - \beta\right) \\ &= -\sum_{t_i\in T}\left(\Omega_{t_i^a t_i^p} - \Omega_{t_i^a t_i^n} - \beta - \log\left(1 + e^{\Omega_{t_i^a t_i^p} - \Omega_{t_i^a t_i^n} - \beta}\right)\right).\end{aligned} \tag{7}$$

A common problem with the deep hashing methods is how to train the neural network output to be binary codes [11, 20, 25, 26]. Minimizing the loss of equation (7) is a very difficult discrete optimization problem [30]. The usual method is to relax $\{b_i\}$ from discrete to continuous. In the last layer of the neural network, we use the sign function as the activation function to obtain the binary code. However, the sign function is nondifferentiable because the gradient of the sign function always equals zero, and the back-propagation of the loss function is difficult to proceed. A good method is to relax the binary codes and add a quantization error term in the objective function during training. This method is also utilized in [30, 25].

This method is described below: we use $\{u_i\}$ to denote the continuous output of the last layer before the sign function for the image. $\{b_i\}$ can be obtained by $b_i = \mathrm{sgn}(u_i)$. We relax $\{b_i\}$ to continuous vectors $\{u_i\}$, and we redefine Ω_{ij} as follows:

$$\Phi_{ij} = \frac{1}{2} u_i^T u_j. \tag{8}$$

Then, we approximate equation (7) as

$$\begin{aligned} l_{\mathrm{triplet}} = \min_B J_2 = -\sum_{t_i\in T}\Bigg(&\Phi_{t_i^a t_i^p} - \Phi_{t_i^a t_i^n} - \beta - \log \\ &\cdot\left(1 + e^{\Phi_{t_i^a t_i^p} - \Phi_{t_i^a t_i^n} - \beta}\right)\Bigg) + \eta \sum_{i=1}^N \left\|b_i - u_i\right\|_2^2,\end{aligned} \tag{9}$$

where $\sum_{i=1}^N \|b_i - u_i\|_2^2$ represents the quantization error term and η is the superparameter, which is used to balance the original objective function and quantization error.

To integrate the above image feature representation part and the loss function part into a deep neural network framework, we set

$$u_i = W^T \Theta(x_i;\ \phi) + v, \tag{10}$$

where ϕ stands for all the parameters of the neural network except for the last layer, $\Theta(x_i;\ \phi)$ represents the output of the neural network, W represents a weight matrix, and v is a bias term.

5.2. Joint Learning with Linear Classification Quantization Loss. Triplet label information was used to learn hash codes by equation (9), but the label information was not fully utilized. We hope to fully utilize the label information so that we use the joint learning linear classifier to further optimize the hash code, making the learned hash code optimal. Inspired by the study in [31], we use the following classifier, which can represent the relationship between the learned hash code B and label information Y:

$$Y = W^T B, \tag{11}$$

where $W_{L\times c} = \{w_i\}_{i=1}^{c}$ is the classifier weight and $Y_{c\times N} = \{y_i\}_{i=1}^{N}$ is the ground-truth label vector, in which$c$ is the number of categories in the dataset. We usually choose $l2$ loss for the linear classifier, and we define the loss function as follows:

$$l_{\text{linear}} = L\left(Y, W^T B\right) + \mu\|W\|_F^2 = \sum_{i=1}^{N}\left\|y_i - W^T b_i\right\|_2^2 + \mu\|W\|_F^2, \tag{12}$$

where l_{linear} is the linear classifier loss function, μ is the regularization parameter, and $\|\|_F^2$ is the F norm of a matrix. Equations (9) and (12) are combined by weight parameters, and the following formula can be obtained:

$$\begin{aligned} l_{\text{total}} = l_{\text{triplet}} + \lambda l_{\text{linear}} = &-\sum_{t_i\in T}\left(\Phi_{t_i^a t_i^p} - \Phi_{t_i^a t_i^n} - \beta - \log\right.\\ &\left.\cdot\left(1 + e^{\Phi_{t_i^a t_i^p} - \Phi_{t_i^a t_i^n} - \beta}\right)\right)\\ &+ \eta\sum_{i=1}^{N}\left\|b_i - u_i\right\|_2^2 + \lambda\left(\sum_{i=1}^{N}\left\|y_i - W^T b_i\right\|_2^2 + \mu\|W\|_F^2\right), \end{aligned} \tag{13}$$

where $\|\|_2^2$ is the $l2$ norm of a vector and λ is the trade-off parameter used to balance the triplet likelihood term and the linear classification loss.

5.3. Optimization. Equation (13) is a joint loss optimization problem, which is still nonconvex, and it is difficult to solve. Here, we adopt the discrete cyclic coordinate descent (DCC) optimization method. Equation (13) can be decomposed into two suboptimization problems, and the linear classification loss can be solved iteratively by alternating minimization. For equation (12), the linear classification loss can be rewritten as

$$l_{\text{linear}} = \min_{B,W}\left\|Y - W^T B\right\|_2^2 + \mu\|W\|_F^2. \tag{14}$$

By fixing B, using the matrix trace function tr(), the following formula is obtained:

$$\begin{aligned} \min_W \left\|Y - W^T B\right\|_2^2 + \mu\|W\|_F^2 = \min_W \operatorname{tr}\Big(\left(Y - W^T B\right)^T\\ \cdot\left(Y - W^T B\right)\Big) + \mu\operatorname{tr}\left(W^T W\right). \end{aligned} \tag{15}$$

Taking the derivative of W and setting $dJ(W)/dW = 0$, we get the minimum value of W:

$$W = \left(BB^T + \mu I\right)^{-1} BY^T. \tag{16}$$

So once we solve W, we assume W as a constant matrix. By fixing W, equation (14) becomes

$$\begin{aligned} l_{\text{linear}} = \min_B \left\|Y - W^T B\right\|_2^2 + \mu\|W\|_F^2 = \min_B \|Y\|^2\\ - 2\operatorname{tr}\left(Y^T W^T B\right) + \left\|W^T B\right\|^2. \end{aligned} \tag{17}$$

We can get a closed solution to a row of B by fixing the other rows. We use the discrete cyclic coordinate descent method to iteratively solve B row by row. Let z^T be the $k-$th row of B, $k = 1, 2, \ldots, K$ (K is the length of the hash code), and B' the matrix of B excluding z^T. Let v^T be the $k-$th row of W and W' the matrix of $W_{12\times 10}$ excluding v^T. The third term in equation (17) can be solved as follows:

$$\begin{aligned} \left\|W^T B\right\|^2 &= \left\|\begin{bmatrix} W' \\ v^T \end{bmatrix}^T \begin{bmatrix} B' \\ z^T \end{bmatrix}\right\|^2 = \left\|W'^T B' + vz^T\right\|^2\\ &= \left\|W'^T B'\right\|^2 + \left\|vz^T\right\|^2 + 2tr\\ &\quad\cdot\left(\left(W'^T B'\right)^T vz^T\right)\\ &= \text{const} + 2v^T W'^T B' z. \end{aligned} \tag{18}$$

Similarly, let $Q = WY$, and let q^T be the k^{th} row of Q and Q' the matrix of Q excluding q^T. The second term in equation (17) can be solved as follows:

$$\begin{aligned} \operatorname{tr}\left(Y^T W^T B\right) &= \operatorname{tr}\left(B^T Q\right) = \operatorname{tr}\left(\begin{bmatrix} B' \\ z^T \end{bmatrix}^T \begin{bmatrix} Q' \\ q^T \end{bmatrix}\right)\\ &= \operatorname{tr}\left(B'Q' + zq^T\right) = \operatorname{tr}\left(B'Q'\right) + \operatorname{tr}\left(q^T z\right)\\ &= \text{const} + q^T z. \end{aligned} \tag{19}$$

Putting equations (17), (18), and (19) altogether, there is an optimal solution to this problem:

```
Input: training images $X = \{x_i\}_{i=1}^n$; code length $L$; epochs = 150; superparameters $\beta, \eta, \mu$, and $\lambda$; and minibatch = 128
Initialization: initialize neural network parameters $\phi$, $W$, and $v$ with the AlexNet model and iteration number $t$
  Generate triplet training set: $T = \left\{(t_i^a, t_i^p, t_i^n)\right\}_{i=1}^m$;
for $i = 0$; $i \le t$; $i$++ do
  Randomly sample a minibatch of points from $T$, and for each sampled point $x_i$:
  (i) Calculate $\Theta(x_i;\phi)$ by forward propagation;
  (ii) Compute $u_i = W^T\Theta(x_i;\phi) + v$;
  (iii) Compute the binary code of $x_i$ with $b_i = \mathrm{sgn}(u_i)$;
  (iv) Compute derivatives for point $x_i$;
  (v) Update the parameters $W, v$, and $\phi$ by utilizing BP;
  (vi) Compute batch$B = \{b_i\}_{i=1}^N$;
  for $k = 0$; $k \le L$; $k$++ do
    Discrete cyclic coordinate descent (DCC) optimization:
    (i) Compute $W$ according to (16);
    (ii) Iteratively optimization update $B$ bit by bit using the DCC method according to (21) in the minibatch;
  End for
  Update $b_i = \mathrm{sgn}(u_i)$ in the minibatch according to joint loss function in (13);
End for
Output: $u_i = W^T\Theta(x_i;\phi) + v$, with the parameters $W, v$, and $\phi$;
  Hash codes $B = \{b_i\}_{i=1}^N$.
```

Algorithm 1: The procedure of JLTDH.

$$\begin{aligned} l_{\text{linear}} &= \min_B \left\|W^T B\right\|^2 - 2\mathrm{tr}\left(Y^T W^T B\right) \\ &= \min_z \left(v^T W'^T B' - q^T\right) z \\ \text{s.t.}\quad & z \in \{-1,1\}^n, \end{aligned} \tag{20}$$

$$z = \mathrm{sgn}\left(q^T - v^T W'^T B'\right)^T = \mathrm{sgn}\left(q - B'^T W' v\right). \tag{21}$$

According to equation (21), each bit hash code z is calculated according to the $k-1$ bit B' that has been learned. We can iteratively update each bit until the program converges to a better set of hash codes B.

The proposed method JLTDH is briefly summarized in Algorithm 1.

6. Experiment and Analysis

In this section, we will describe our experiments and results. Three commonly used datasets are used to verify the effectiveness of our algorithm. We calculated the precision and mean average precision (MAP) of the retrieval results and showed the performance of our method on CIFAR-10, NUS-WIDE, and MS-COCO. Specifically, given an anchor x_q, we can calculate its average precision (AP) using the following equation:

$$\mathrm{AP}\left(x_q\right) = \frac{1}{R_k}\sum_{k=1}^{n} P(k)\Delta r(k), \tag{22}$$

where R_k is the number of relevant samples, $P(k)$ is the precision at cutoff k in the returned sample list, and $\Delta r(k)$ is an indicator function which equals 1 if the k-th returned sample is a ground-truth neighbor of x_q. Otherwise, $\Delta r(k)$ is 0. Given Q queries, MAP is the AP of all query results sorted; we can compute the MAP as follows:

$$\mathrm{MAP} = \frac{1}{Q}\sum_{q=1}^{Q} \mathrm{AP}\left(x_q\right). \tag{23}$$

6.1. Experimental Settings. Our server configuration is as follows: the CPU is Intel Xeon CPU E5-2687W @ 3.0 GHz with 12 cores, the GPU is NVIDIA GTX 1080 8 GB, and the RAM is 32 GB. The Linux operating system is Ubuntu 16.04, and the deep learning framework is PyTorch [32].

We use three widely used benchmark datasets of different scales; they are CIFAR-10 [33], NUS-WIDE [34], and MS-COCO [35]. The CIFAR-10 dataset contains 60,000 images and 10,000 test images, belonging to 10 categories. The size of each image is 32×32 pixels. We randomly selected 5,000 images (500 for each class) as the training set and 1,000 images (100 for each class) as the test query set.

The NUS-WIDE dataset contains 269,648 images in 81 categories. We used the 21 most commonly used categories. We randomly selected 2,100 images (100 images per class) as the query point and randomly selected 10,500 images (500 images per class) as the training set.

MS-COCO is an image dataset widely used for image recognition, segmentation, and captioning. It contains 82,783 training images and 40,504 validation images, in which each image is labeled by some of the 80 semantic concepts. We randomly selected 5,000 images as the query point and the rest images as the database and randomly sample 10,000 images from the database for training. Table 3

Table 3: Examples of the datasets.

Dataset	Example	Label
CIFAR-10		"airplane"
		"cat"
		"horse"
NUS-WIDE		"clouds," "ocean," "sky," "water"
		"person," "ocean," "water"
MS-COCO		"person," "sheep," "dog"
		"dog," "bottle," "barrel"

Table 4: Dataset settings used in our experiment.

Dataset	Total	Training (randomly)	Testing (randomly)	Labels
CIFAR-10	60,000	5,000 (500 ∗ 10)	1,000 (100 ∗ 10)	10
NUS-WIDE	269,648	10,500 (500 ∗ 21)	2,100 (100 ∗ 21)	21
MS-COCO	82,783	10,000	5,000	80

Table 5: MAP results for different numbers of bits (12, 24, 32, and 48 bits) on the two benchmark image datasets (CIFAR-10 and NUS-WIDE).

Methods		CIFAR-10				NUS-WIDE			
		12 bits	24 bits	32 bits	48 bits	12 bits	24 bits	32 bits	48 bits
Deep methods	Ours	**0.770**	**0.774**	**0.788**	**0.783**	**0.791**	**0.828**	**0.832**	**0.836**
	DTSH	0.710	0.750	0.765	0.774	0.773	0.808	0.812	0.824
	DPSH	0.713	0.727	0.744	0.757	0.752	0.790	0.794	0.812
	DQN	0.554	0.558	0.564	0.580	0.768	0.776	0.783	0.792
	DHN	0.555	0.594	0.603	0.621	0.708	0.735	0.748	0.758
	NINH	0.552	0.566	0.558	0.581	0.674	0.697	0.713	0.715
	CNNH	0.439	0.511	0.509	0.522	0.611	0.618	0.625	0.608
Nondeep methods	FastH	0.305	0.349	0.369	0.384	0.621	0.650	0.665	0.687
	SDH	0.285	0.329	0.341	0.356	0.568	0.600	0.608	0.637
	KSH	0.303	0.337	0.346	0.356	0.556	0.572	0.581	0.588
	LFH	0.176	0.231	0.211	0.253	0.571	0.568	0.568	0.585
	SPLH	0.171	0.173	0.178	0.184	0.568	0.589	0.597	0.601
	ITQ	0.162	0.169	0.172	0.175	0.452	0.468	0.472	0.477
	SH	0.127	0.128	0.126	0.129	0.454	0.406	0.405	0.400

shows some sample points from three datasets. Table 4 shows the dataset settings used in our experiment.

We compared our method with several representative hashing methods for MAP; the comparison methods we selected are divided into two groups: traditional hashing methods and deep hashing methods. Traditional unsupervised hashing methods include SH [36] and ITQ [3]. Traditional supervised hashing methods include KSH [5], FastH [8], SDH [9], SPLH [37], and LFH [19]. The deep hashing methods include DSRH [30], DSCH [38], DRCSH [38], CNNH [18], NINH [12], DPSH [13], DHN [14], DQN [39], DTSH [40], VDSH [41], and DSDH [42]. In this paper, we also emphasize on the comparison of the deep hashing methods with triplet labels, including DSRH [37], DSCH [38], DRSCH [38], and NINH [12]. To be fair, some test results are directly evaluated in previously published papers. Following [41], CNN features on CIFAR-10 were extracted using the pretrained CNN-F model. The hyperparameters of this method are set according to the standard cross-validation procedure. In equation (12), u is set to 0.1, λ is 1, η is 55, and β is half the length of the hash code.

6.2. Empirical Analysis

6.2.1. Comparison to Other Deep Methods and Nondeep Methods. As shown in Table 5 and Figure 2, the MAP is calculated based on the top 5,000 returned neighbors. NINH, CNNH, KSH, and ITQ results were obtained from the study in [11, 18]. Results of other methods were obtained from the study in [42]. Our method performs better in these three datasets than the existing hashing methods; compared with nondeep learning methods, our method has been significantly improved. Our method further improved performance by 2–6% compared to the current best deep learning methods. At the same time, we found that our method performed better in shorter hash codes (≤32 bits). Most deep hashing methods have significant performance advantages.

MAP results for different numbers of bits on the MS-COCO dataset are shown in Table 6; except for our method, the other results were obtained from the study in [29]. In order to be consistent with the settings in [29], we set the hash code length as 8 bits, 16 bits, 24 bits, and 32 bits. The image pixel of MS-COCO is more complex, which will lead to more difficult classification, which may lead to inaccurate results of feature extraction and inaccurate classification results. As can be seen from Table 5, the performance of the MS-COCO dataset decreased to a certain extent compared with the results of MAP of the NUS-WIDE dataset. In spite of this, our method is still much better than the comparison methods.

Our method can achieve excellent performance under shorter hash codes. At the same time, we also discussed the performance change under the long hash code. Since the comparison method does not provide the results of long hash codes, we only discuss our own method here. As can be seen from Figure 3, CIFAR-10 gets a good MAP value at 32 bits, while NUS-WIDE gets a good MAP value between 48 and 64 bits. With the growth of hash code length, the MAP of CIFAR-10 is decreasing, while the MAP of NUS-WIDE is not decreasing, but there is no obvious increase.

6.2.2. Comparison to Nondeep Hashing Methods Using Deep Features. One might say that the performance improvements come from the neural network, not our approach. In order to further verify our method, we compared our method with the traditional hashing method using the CNN-F network [43] to extract the depth features. As shown in Figure 4, we can see that our method is significantly superior to the traditional method. It should be

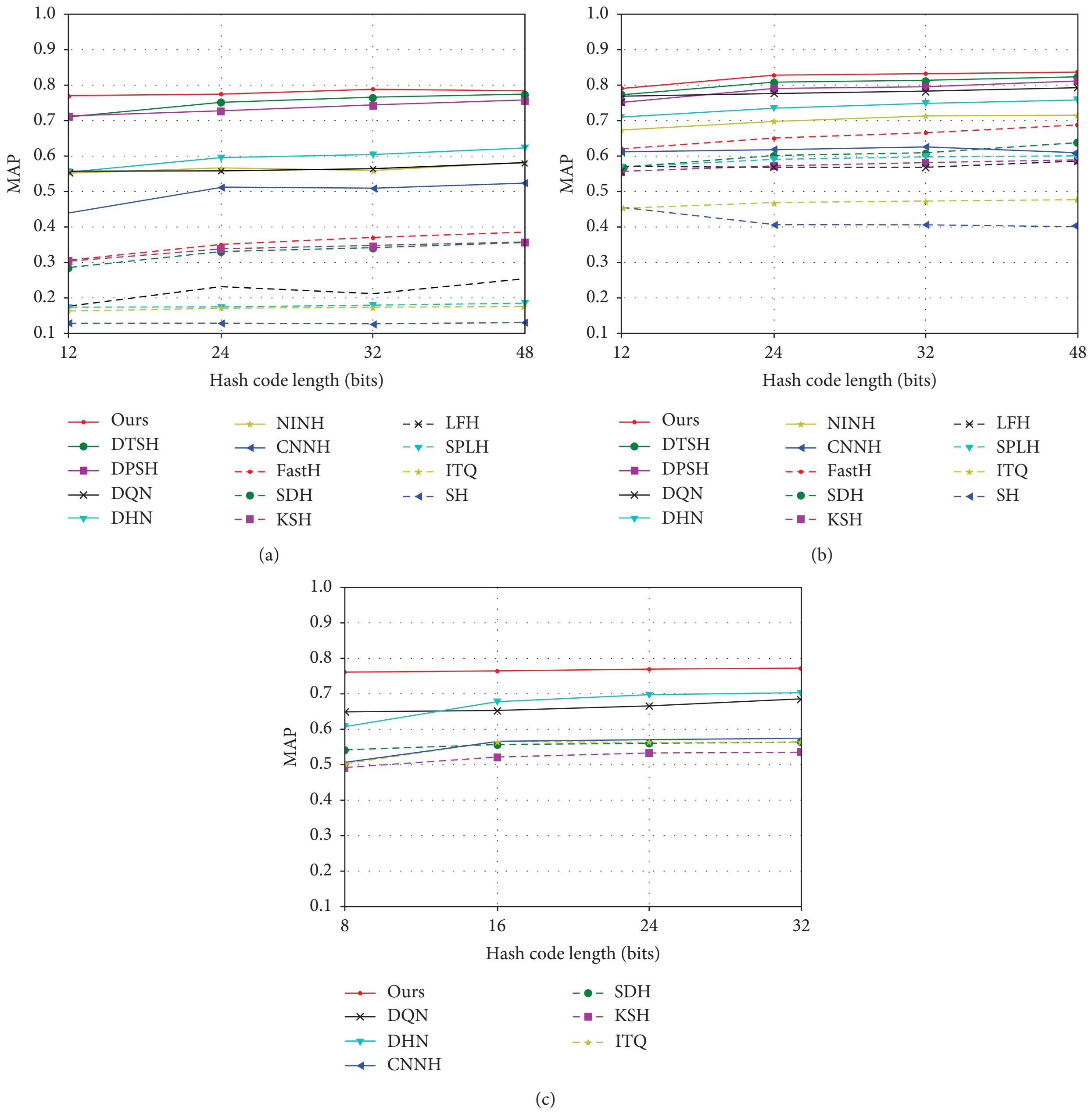


FIGURE 2: MAP results for different numbers of bits on the three benchmark image datasets. (a) CIFAR-10. (b) NUS-WIDE. (c) MS-COCO.

TABLE 6: MAP results for different numbers of bits on the MS-COCO dataset.

Methods	MS-COCO			
	8 bits	16 bits	24 bits	32 bits
Ours	**0.761**	**0.765**	**0.768**	**0.772**
DQN	0.649	0.653	0.666	0.685
DHN	0.607	0.677	0.697	0.701
CNNH	0.505	0.564	0.569	0.574
SDH	0.541	0.555	0.560	0.564
KSH	0.492	0.521	0.533	0.534
ITQ-CCA	0.501	0.566	0.563	0.562

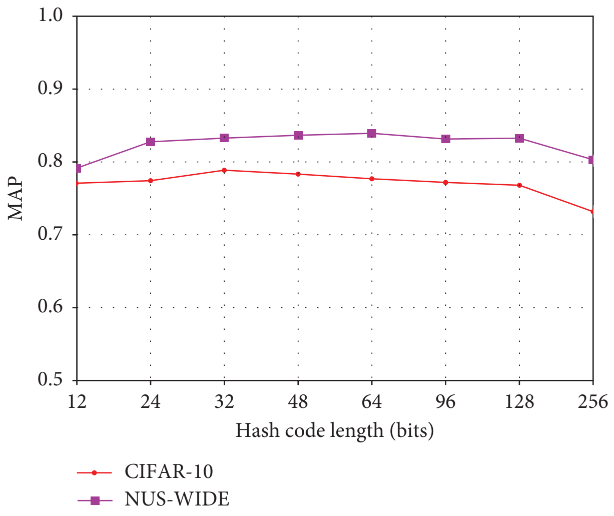


FIGURE 3: Effects of longer hash codes.

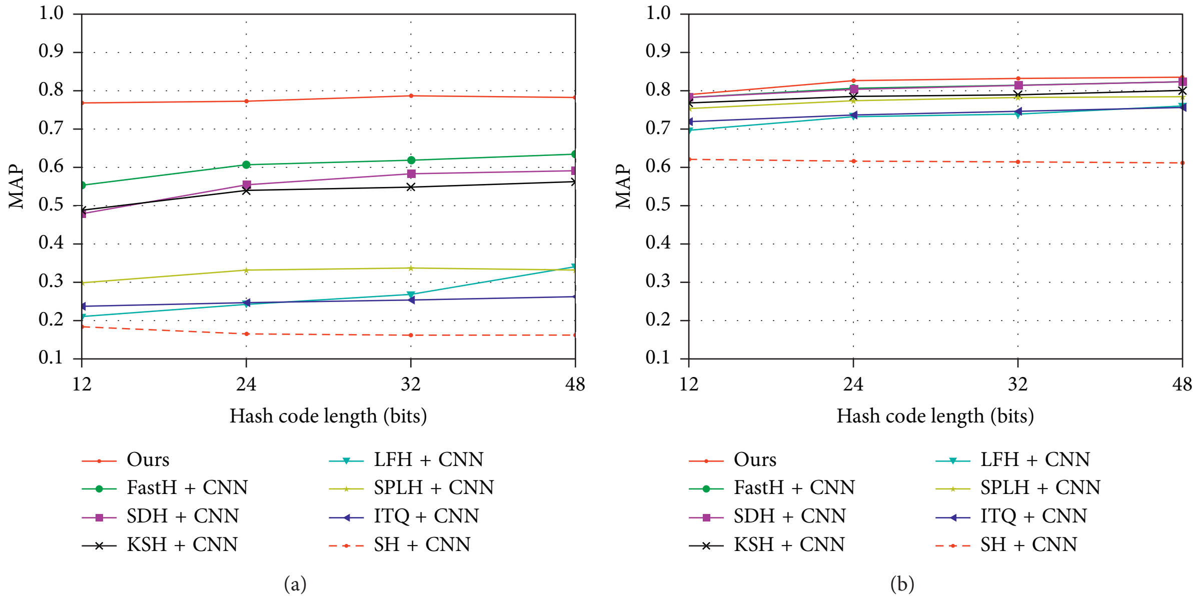


FIGURE 4: MAP results for different numbers of bits on the two benchmark image datasets (a) CIFAR-10 and (b) NUS-WIDE, compared to different traditional hashing methods using the CNN.

noted that most of the traditional methods are not tested based on MS-COCO, and we cannot get the corresponding data, so we did not do the comparative test of MS-COCO in this part.

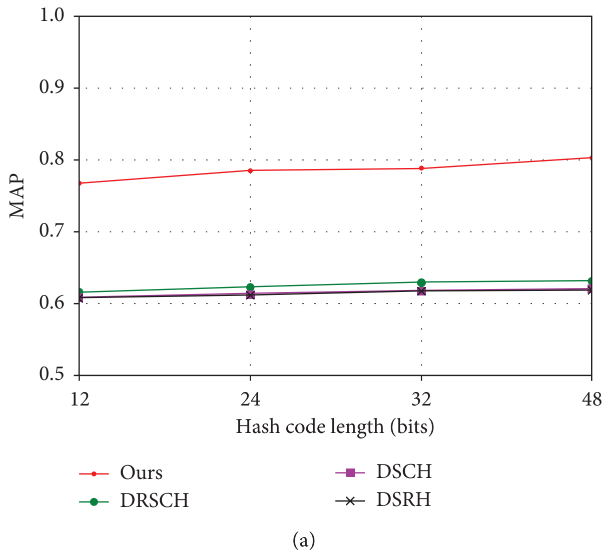
6.2.3. *Comparison to Hashing Methods with Triplet Labels.*

This paper adopts the method of triplet labels; therefore, we focus on the comparison of the deep hashing methods with the triplet label, and we will compare it further with other deep hashing methods using the triplet label. These methods include DSRH [37], DSCH [38], DRSCH [38], and NINH [12]. The results of DSRH, DSCH, and DRSCH were adopted from the study in [38]. As shown in Figure 5, compared with previous deep hashing methods based on the triplet label, our method is obviously better and leads the comparative method by a wide margin.

6.3. *Sensitivity to Hyperparameters.*

The most important hyperparameters for JLTDH are λ and η. λ is the trade-off parameter used to balance the triplet likelihood term and the linear classification loss. η is used to balance the likelihood term and the quantization error. We explore the influence of these two hyperparameters.

We report the MAP values for different λ from the range of [0.1, 200] on two datasets with the code length being 12 bits and 32 bits. We can find that JLTDH is not sensitive to λ in a large range when $10 < \lambda < 50$. According to Figure 6(a), we found that when $\lambda = 10$, CIFAR-10 can



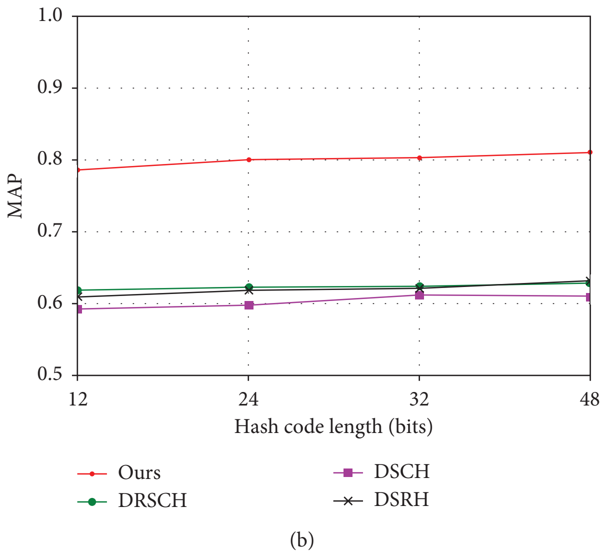
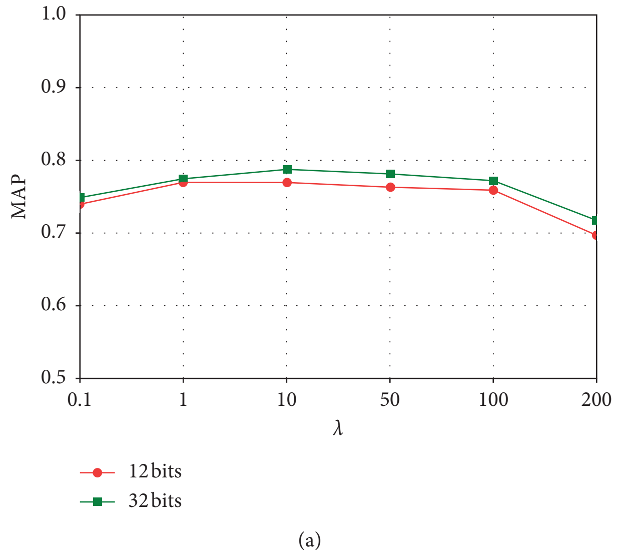


FIGURE 5: MAP results for different numbers of bits on the two benchmark image datasets (a) CIFAR-10 and (b) NUS-WIDE, compared with the triplet label methods.



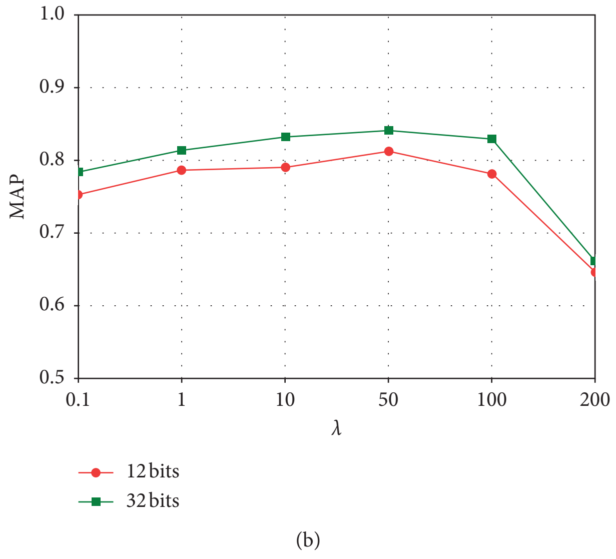


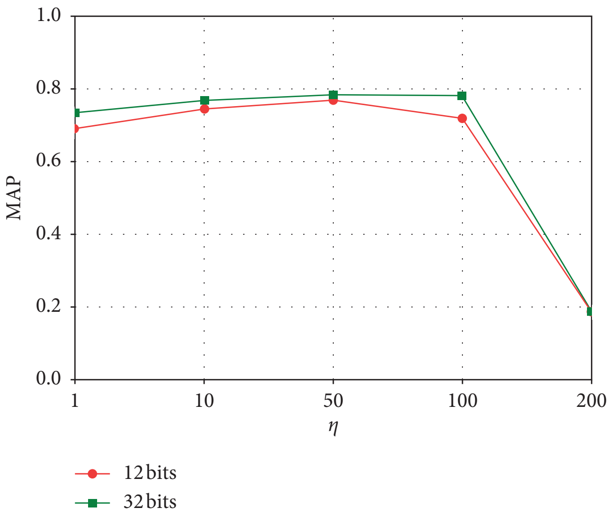
FIGURE 6: Effects of λ on (a) CIFAR-10 and (b) NUS-WIDE. Hash codes = 12 and 32 bits.

obtain the best MAP performance. As shown in Figure 6(b), when $10 < \lambda < 50$, NUS-WIDE can achieve better MAP performance.

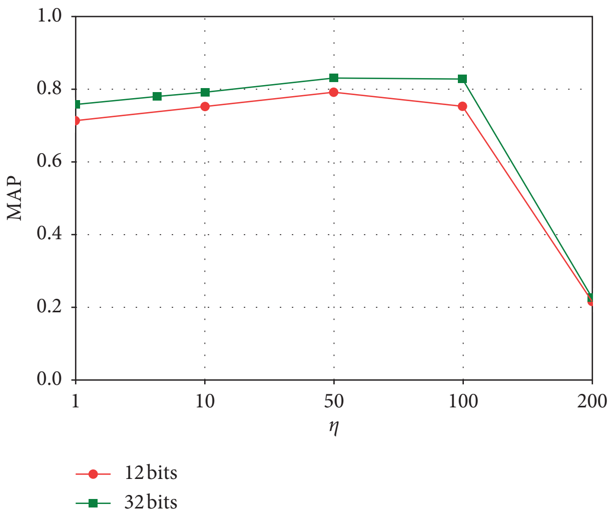
Furthermore, we present the influence of η in Figures 7(a) and 6(b). As can be seen, there is a wide range of η in that our method performs well. When hash code = 12 bits, MAP accuracy is better on both datasets with $\eta = 50$, and when hash code = 32 bits, MAP accuracy is better on both datasets with $50 \leq \eta \leq 100$. Other super-parameter settings are obtained through cross-validation: u is set to 0.1, and β is half the length of the hash code.

6.4. Analysis of Three Loss Functions. By observing the convergence of the loss function, we can judge whether the selected loss function is reasonable or not. Figure 8 shows the change of three kinds of losses for different lengths of hash codes during training. We only take CIFAR-10 as an example; the results for the other two datasets are similar. It is reasonable to conclude that the joint loss combining triplet likelihood loss and linear classification quantization loss successfully optimizes the loss during training. Figures 8(a) and 8(b) are similar: they show that the joint loss and triplet likelihood loss converge rapidly and are kept at a low level for different bits, that our method is reasonable and effective, and that the optimization process only needs a few iterations.

In order to further reveal the respective influence of the two parts of the joint loss function, we have shown the loss curves of triplet likelihood loss and linear classification loss, respectively, in Figures 8(b) and 8(c). The magnitude of the

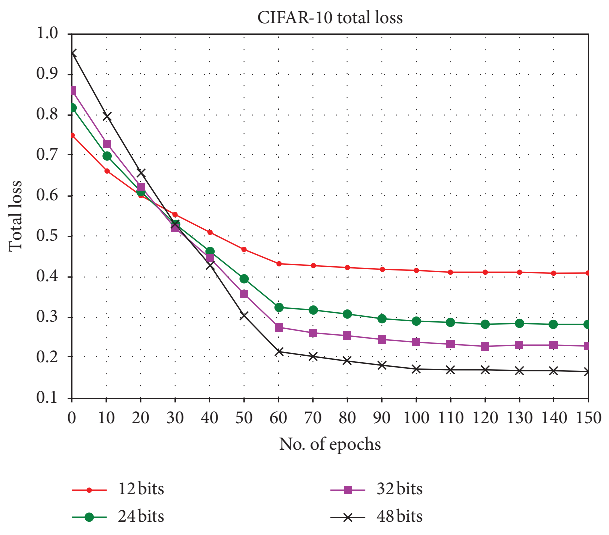


(a)

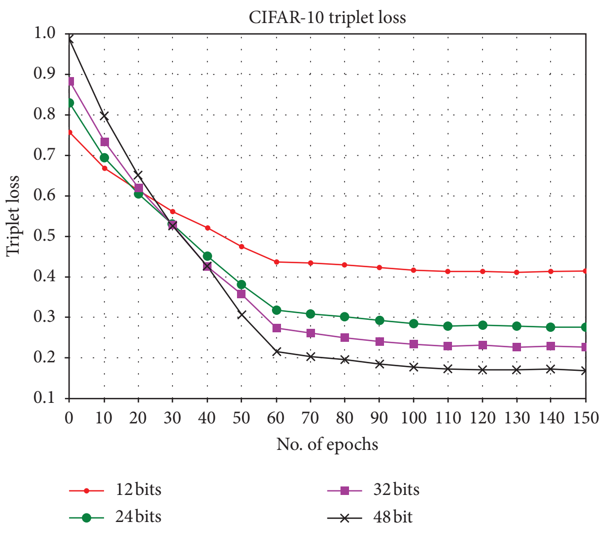


(b)

Figure 7: Effects of η on (a) CIFAR-10 and (b) NUS-WIDE. Hash codes = 12 and 32 bits.

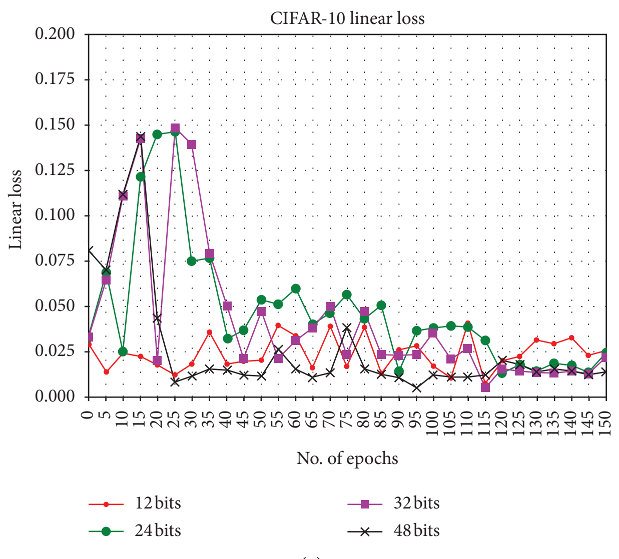


(a)



(b)

Figure 8: Continued.



(c)

Figure 8: Convergence of three kinds of losses for different lengths of hash codes (epoch = 150). (a) Joint loss. (b) Triplet likelihood loss. (c) Linear classification loss.

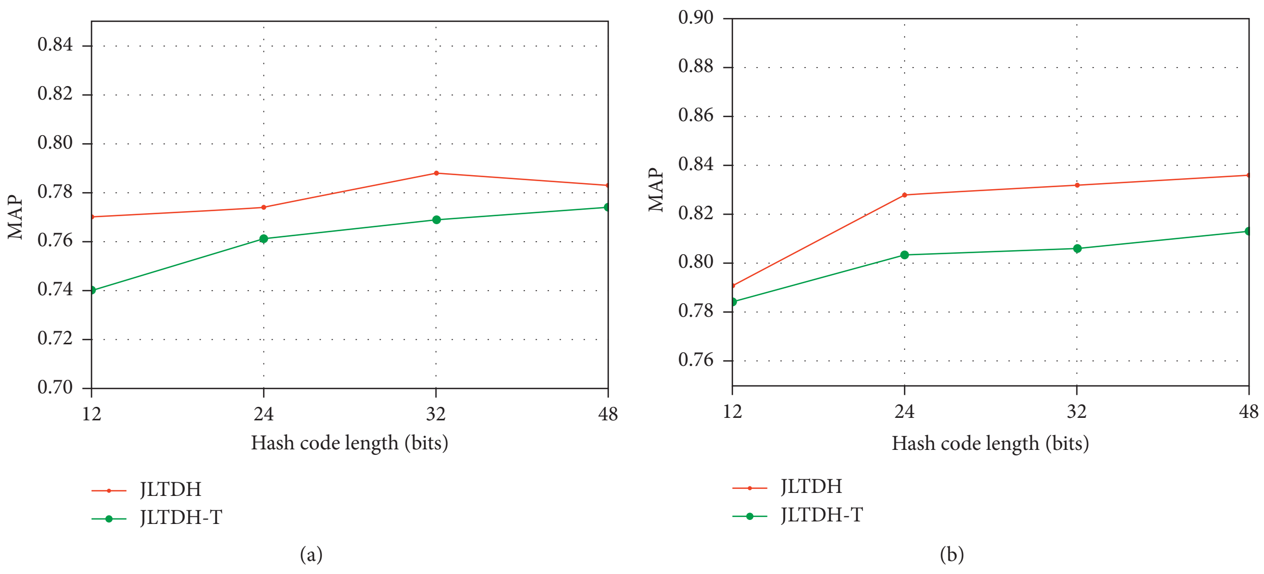


Figure 9: (a) Ablation effects of loss on CIFAR-10. (b) Ablation effects of loss on NUS-WIDE.

loss value in Figure 8(c) is about 10^{-1} of that in Figure 7(b), which is because triplet likelihood loss is used to optimize the first stage and plays a major role in training, while linear classification loss is used to optimize the second stage, which is further optimization based on the first stage and fine-tuning optimization.

6.4.1. Ablation Study about Loss Function. In order to confirm the contribution of different losses to final performance, we selected a variant of JLTDH for comparison: JLTDH-T is a variant of JLTDH, whose loss function contains only triplet loss and no linear classification loss. As can be seen from Figure 9, on the CIFAR and NUS-WIDE datasets, JLTDH-T can achieve good MAP performance with only triplet loss. Based on further optimization of linear classification loss, JLTDH achieves better MAP performance by using the joint loss. JLTDH is about 2% ahead of JLTDH-T on average.

6.5. Visualization of Query Results. We show the illustration of top 12 returned images for better understanding of the impressive performance improvement of the proposed method. Figure 10 illustrates the top 12 returned images of the proposed method for three query images on the three datasets CIFAR-10, NUS-WIDE, and MS-COCO, and the length of the hash code is 32. It shows that the method we proposed can truly preserve the features of an image and

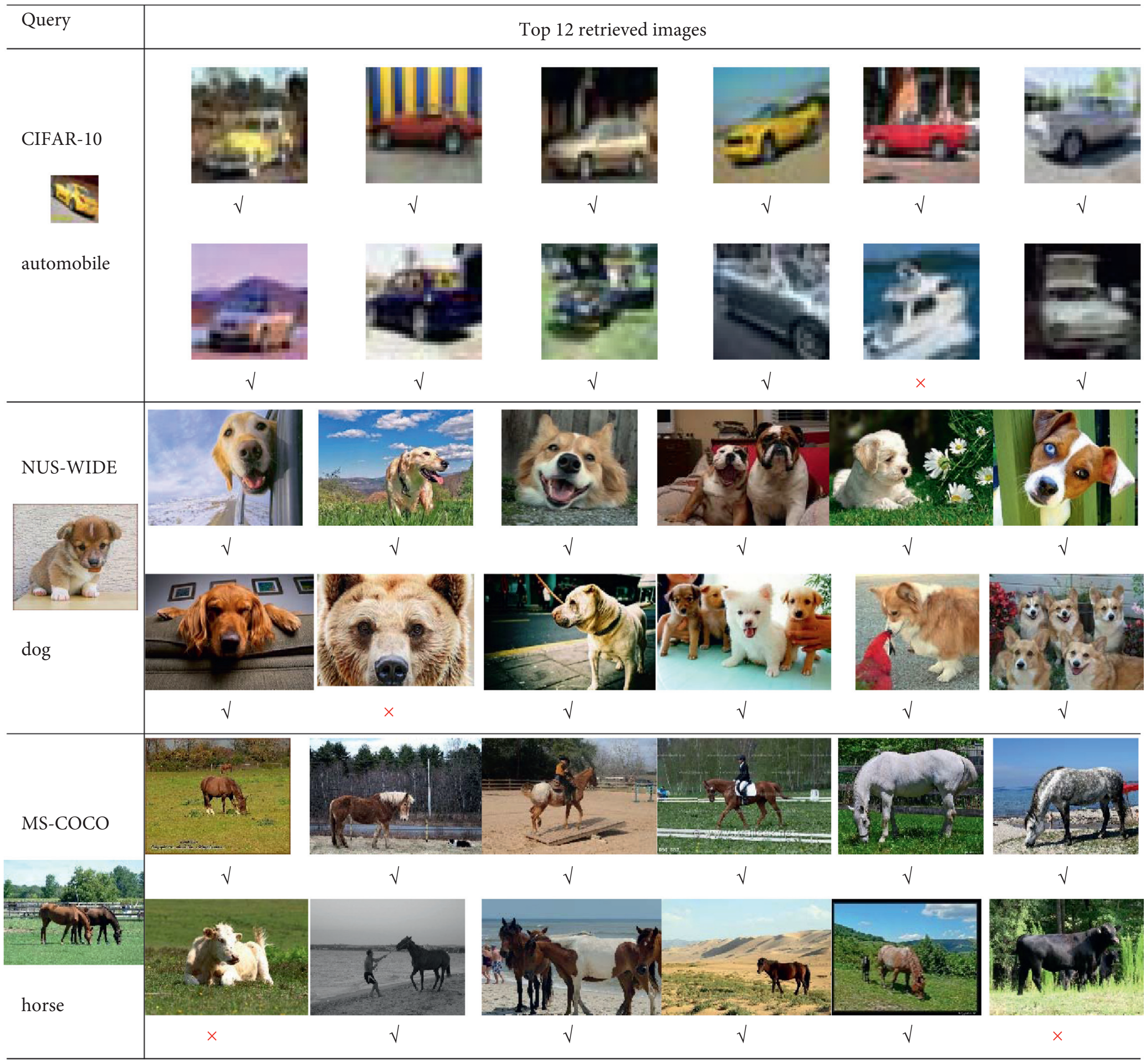


Figure 10: The top 12 images returned by the proposed method.

save them to the hash code. Regarding the query image in CIFAR-10, only one of the returned results is wrong, and the wrong image is only at the bottom of the sorted image. In contrast, for the query image in NUS-WIDE, only 1 out of the top 12 images is incorrect, and the top 12 images have a similar shape or similar color to the query image. And for the query image in MS-COCO, ten of the top 12 images are correct; compared with the previous two datasets, the query accuracy is slightly reduced. The possible reason is that MS-COCO is a multiobjective dataset. This shows that our method can provide the desired search results.

7. Conclusion

In this work, we propose a triplet deep hashing method with joint supervised loss based on the convolutional neural network (JLTDH). To fully utilize the supervised triplet information, this paper proposed a joint loss function combining two kinds of supervised loss functions: the triplet negative log-likelihood loss function and the linear classification loss function. At the same time, in order to overcome the cubic increase in the number of triplets and make triplet training more effective, we designed a triple selection method. The whole process is divided into two stages: Firstly, the last layer of the network outputs a preliminary hash code. Secondly, relying on the joint loss function and backpropagation algorithm, the parameters of the neural network are further updated so that the generated hash code has higher query precision. We perform extensive experiments on the three public benchmark datasets CIFAR-10, NUS-WIDE, and MS-COCO. Experimental results demonstrate that the proposed method outperforms the compared methods and is also superior to all previous deep hashing methods based on the triplet label.

Data Availability

Data are owned by a third party: the experimental part of this paper uses three widely used public datasets (CIFAR-10, MS-

COCO, and NUS-WIDE), which can be publicly accessed. CIFAR-10 can be accessed at http://www.cs.toronto.edu/kriz/cifar.html, MS-COCO at http://mscoco.org, and NUS-WIDE at http://lms.comp.nus.edu.sg/research/NUS-WIDE.htm The implementation code of the algorithm in this paper is written by PyTorch. The code can be obtained from Mingyong Li upon request at limingyong@cqnu.edu.cn.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

Acknowledgments

This work was supported by the science and technology project of Chongqing Education Commission of China (No. KJ1600332), the humanities and social science project of Ministry of Education (18YJA880061), the humanities and social science project of Chongqing Education Commission of China (No. 19SKGH035), Chongqing Education Scientific Planning Project (No. 2017-GX-116), and Chongqing Graduate Education Reform Project (No. yjg193093).

References

[1] Y. Guo, G. Ding, L. Liu, J. Han, and L. Shao, "Learning to hash with optimized anchor embedding for scalable retrieval," *IEEE Transactions on Image Processing*, vol. 26, no. 3, pp. 1344–1354, 2017.

[2] B. Kulisand and K. Grauman, "Ker-nelized locality-sensitive hashing for scalable image search," in *Proceedings of the 2009 IEEE 12th International Conference on Computer Vision*, pp. 2130–2137, Kyoto, Japan, September-October 2009.

[3] Y. Gong and S. Lazebnik, "Iterative quantization: a procrustean approach to learning binary codes," in *Proceedings of the CVPR*, pp. 817–824, Colorado Springs, CO, USA, August 2011.

[4] W. Kong and W.-J. Li, "Isotropic hashing," in *Proceedings of the Advances in Neural Information Processing Systems*, pp. 1646–1654, December 2012.

[5] W. Liu, J. Wang, R. Ji, Y.-G. Jiang, and S.-F. Chang, "Supervised hashing with kernels," in *Proceedings of the 2012 IEEE Conference on Computer Vision and Pattern Recognition*, pp. 2074–2081, Providence, RI, USA, June 2012.

[6] M. Rastegari, J. Choi, S. Fakhraei, D. Hal, and L. S. Davis, "Predictable dual-view hashing," in *Proceedings of the ICML*, pp. 1328–1336, Atlanta, GA, USA, June 2013.

[7] K. He, F. Wen, and J. Sun, "K-means hashing: an affinity-preserving quantization method for learning binary compact codes," in *Proceedings of the CVPR*, pp. 2938–2945, Portland, OR, USA, June 2013.

[8] G. Lin, C. Shen, Q. Shi, H. Anton van den, and D. Suter, "Fast supervised hashing with decision trees for high-dimensional data," in *Proceedings of the CVPR*, pp. 1971–1978, Columbus, OH, USA, June 2014.

[9] F. Shen, X. Zhou, Y. Yang, J. Song, H. T. Shen, and D. Tao, "A fast optimization method for general binary code learning," *IEEE Transactions on Image Processing*, vol. 25, no. 12, pp. 5610–5621, 2016.

[10] X. Liu, B. Du, C. Deng, M. Liu, and B. Lang, "Structure sensitive hashing with adaptive product quantization," *IEEE Transactions on Cybernetics*, vol. 46, no. 10, pp. 2252–2264, 2016.

[11] K. Lin, H.-F. Yang, J.-H. Hsiao, and C.-S. Chen, "Deep learning of binary hash codes for fast image retrieval," in *Proceedings of the CVPR Workshops*, pp. 27–35, Boston, MA, USA, June 2015.

[12] H. Lai, Y. Pan, Y. Liu, and S. Yan, "Simultaneous feature learning and hash coding with deep neural networks," in *Proceedings of the 2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 3270–3278, Boston, MA, USA, June 2015.

[13] W.-J. Li, S. Wang, and W.-C. Kang, "Feature learning based deep supervised hashing with pairwise labels," in *Proceedings of the IJCAI*, pp. 1711–1717, New York, NY, USA, July 2016.

[14] H. Zhu, M. Long, J. Wang, and Y. Cao, "Deep hashing network for efficient similarity retrieval," in *Proceedings of the AAAI*, Phoenix, AZ USA, February 2016.

[15] W. Liu, C. Mu, S. Kumar, and S.-F. Chang, "Discrete graph hashing," in *Proceedings of the Advances in Neural Information Processing Systems*, pp. 3419–3427, Montreal, Canada, December 2014.

[16] Y. Zheng, J. Zhu, W. Fang, and L.-H. Chi, "Deep learning hash for wireless multimedia image content security," *Security and Communication Networks*, vol. 2018, Article ID 8172725, 13 pages, 2018.

[17] Q.-Y. Jiang and W.-J. Li, "Scalable graph hashing with feature transformation," in *Proceedings of the IJCAI*, pp. 2248–2254, Buenos Aires, Argentina, July 2015.

[18] R. Xia, Y. Pan, H. Lai, C. Liu, and S. Yan, "Supervised hashing for image retrieval via image representation learning," in *Proceedings of the AAAI*, pp. 2156–2162, Québec City, Canada, July 2014.

[19] P. Zhang, W. Zhang, W.-J. Li, and M. Guo, "Supervised hashing with latent factor models," in *Proceedings of the SIGIR*, pp. 173–182, Gold Coast, Australia, July 2014.

[20] X. Liu, Z. Li, C. Deng, and D. Tao, "Distributed adaptive binary quantization for fast nearest neighbor search," *IEEE Transactions on Image Processing*, vol. 26, no. 11, pp. 5324–5336, 2017.

[21] X. Liu, J. He, and S.-F. Chang, "Hash bit selection for nearest neighbor search," *IEEE Transactions on Image Processing*, vol. 26, no. 11, pp. 5367–5380, 2017.

[22] H. Liu, R. Wang, S. Shan, and X. Chen, "Deep supervised hashing for fast image retrieval," in *Proceedings of the 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Las Vegas, NV, USA, June 2016.

[23] Y. Ma, H. Yue, F. Ding, S. Hu, J. Li, and X. Liu, "Progressive generative hashing for image retrieval," in *Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence*, Stockholm, Sweden, July 2018.

[24] C. Deng, Z. Chen, X. Liu, X. Gao, and D. Tao, "Triplet-based deep hashing network for cross-modal retrieval," *IEEE Transactions on Image Processing*, vol. 27, no. 8, pp. 3893–3903, 2018.

[25] C. Li, C. Deng, L. Wang, De Xie, and X. Liu, "Coupled CycleGAN: unsupervised hashing network for cross-modal retrieval," in *Proceedings of the AAAI Conference on Artificial Intelligence*, Honolulu, HI, USA, February 2019.

[26] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Proceedings of the Advances in Neural Information Processing Systems*, pp. 1097–1105, Lake Tahoe, NV, USA, December 2012.

[27] C. Szegedy, W. Liu, Y. Jia et al., "Going deeper with convolutions," in *Proceedings of the 2015 IEEE Conference on*

Computer Vision and Pattern Recognition (CVPR), pp. 1–9, Boston, MA, USA, June 2015.

[28] K. He, X. Zhang, S. Ren, and J. Sun, “Deep residual learning for image recognition,” in *Proceedings of the 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Las Vegas, NV, USA, June 2016.

[29] B. Liu, Y. Cao, M. Long, J. Wang, and J. Wang, “Deep triplet quantization,” in *Proceedings of the ACMMM*, Seoul, Republic of Korea, October 2018.

[30] F. Zhao, Y. Huang, L. Wang, and T. Tan, “Deep semantic ranking based hashing for multi-label image retrieval,” in *Proceedings of the 2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 1556–1564, Boston, MA, USA, June 2015.

[31] F. Shen, C. Shen, W. Liu, and H. T. Shen, “Supervised discrete hashing,” in *Proceedings of the 2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Boston, MA, USA, June 2015.

[32] P. Adam, S. Gross, S. Chintala, Gregory Chanan, and E. Yang, “Automatic differentiation in PyTorch,” in *Proceedings of the Advances in Neural Information Processing Systems*, Long Beach, CA, USA, December 2017.

[33] A. Krizhevsky, I. Sutskever, and G. E. Hinton, “Imagenet classification with deep convolutional neural networks,” in *Proceedings of the Annual Conference on Neural Information Processing Systems*, pp. 1106–1114, Lake Tahoe, NV, USA, December 2012.

[34] 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 *Proceedings of the ACM International Conference on Image and Video Retrieval*, Santorini Island, Greece, July 2009.

[35] T.-Y. Lin, M. Maire, S. Belongie et al., “Microsoft coco: common objects in context,” in *ECCV*, pp. 740–755, Springer, Berlin, Germany, 2014.

[36] Y. Weiss, A. Torralba, and R. Fergus, “Spectral hashing,” in *Proceedings of the 22nd Annual Conference on Neural Information Processing Systems, NIPS 2008*, pp. 1753–1760, Vancouver, BC, Canada, December 2008.

[37] J. Wang, S. Kumar, and S. F. Chang, “Sequential projection learning for hashing with compact codes,” in *Proceedings of the 27th International Conference on Machine Learning (ICML-10)*, pp. 1127–1134, Haifa, Israel, June 2010.

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

[39] Y. Cao, M. Long, J. Wang, H. Zhu, and Q. Wen, “Deep quantization network for efficient image retrieval,” in *Proceedings of the AAAI*, pp. 3457–3463, Phoenix, Arizona, February 2016.

[40] X. Wang, Y. Shi, and K. M. Kitani, “Deep supervised hashing with triplet labels,” in *ACCV*, pp. 70–84, Springer, Berlin, Germany, 2016.

[41] Z. Zhang, Y. Chen, and V. Saligrama, “Efficient training of very deep neural networks for supervised hashing,” in *Proceedings of the 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 1487–1495, Las Vegas, NV, USA, June 2016.

[42] L. Qi, Z. Sun, R. He, and T. Tan, “Deep supervised discrete hashing,” in *Proceedings of the Advances in Neural Information Processing Systems*, Long Beach, CA, USA, December 2017.

[43] K. Chatfield, K. Simonyan, A. Vedaldi, and A. Zisserman, “Return of the devil in the details: delving deep into convolutional nets,” 2014, http://arxiv.org/abs/1405.3531.