Received 5 January 2023, accepted 7 February 2023, date of publication 14 February 2023, date of current version 23 February 2023.

Digital Object Identifier 10.1109/ACCESS.2023.3244813

RESEARCH ARTICLE

Non-Relaxation Deep Hashing Method for Fast Image Retrieval

XIAOFEI LI
College of Computer Engineering and Artificial Intelligence, Jilin University of Architecture and Technology, Changchun 130000, China
e-mail: lixiaofei2019@126.com

This work was supported by the Jilin Province Science and Technology Development Plan Project under Grant 20190201198JC.

ABSTRACT Deep hashing methods utilize an end-to-end framework to mutually learn feature representations and hash codes, thereby achieving a better retrieval performance. Traditional supervised hashing methods adopt handcrafted features for hashing function learning and then generate hash codes through classification and quantization. The lack of adaptability and independence of the quantization procedure leads to low retrieval accuracy of supervised hashing methods with handcrafted features in image retrieval. In this study, a non-relaxation deep hashing method for fast image retrieval is proposed. In this method, a differentiable host thresholding function is used to encourage hash-like codes to approach -1 or 1 nonlinearly at the output of the convolutional neural, instead of the symbol function for quantization used in the traditional method. The output of the host thresholding function is directly used to compute the network training error, and a loss function is elaborately designed with the norm to constrain each bit of the hash-like code to be as binary as possible. Finally, a symbol function is added outside the trained network model to generate binary hash codes for image storage and retrieval in a low-dimensional binary space. Extensive experiments on two large-scale public datasets show that our method can effectively learn image features, generate accurate binary hash codes, and outperform state-of-the-art methods in terms of the mean average precision.

INDEX TERMS Image retrieval, deep hash, convolutional neural network.

I. INTRODUCTION

In a machine-learning algorithm, the hashing algorithm can map similar information from a high-dimensional space to a low-dimensional space with good similarity. The main idea of the hash algorithm is to map information of any dimension into a low-dimensional space with semantic similarity and fixed dimensions. Owing to its low computational cost and high storage efficiency, it is one of the most commonly used techniques for content-based image retrieval(CBIR) [1]. Considering the computational loss and storage capacity of the algorithm, a deep hash was used for large-scale information searches [2], [3], [4], [5], [6].

In the early days, LSH [7] mapped the original data to binary codes using a random hash function. Methods [8], [9], [10] such as LSH are data-independent hash methods that learn hash functions without training data. To generate a hash function related to training data, several data-dependent hash methods [11], [12], [13], [14], [15] that achieve better retrieval performance have been proposed over the past decades. Furthermore, data-dependent hashing can be categorized into unsupervised [11], [12] and supervised [13], [14], [15] methods, based on whether supervised information is used. Traditional data-dependent hashing methods comprise hand-crafted feature representations and hash coding.

In recent years, inspired by the advanced achievements of convolutional neural networks (CNN), deep hashing methods [16], [17], [18], [19], [20], [21] have attracted increasing interest. To make better use of more widely available unlabeled data, unsupervised deep hashing methods have also been proposed. However, natural images usually contain considerable variability in trivial factors such as location, color, and pose. Pixel-wise reconstruction may degrade the

The associate editor coordinating the review of this manuscript and approving it for publication was Gianluigi Ciocca.

learned hash codes by focusing on these trivial variations. Other recent deep hashing methods learn hash codes by maximizing their representation capacity [22] or enforcing the similarity between rotated images and their corresponding original images [23]. HashNet [18] presented a novel method that solved the original non-smooth optimization problem by iteratively optimizing a similar smooth loss function. In addition, many graph-hashing methods have recently been proposed [24]. Deep hashing methods exhibit promising performance in terms of image retrieval and classification with a binary representation of data [25]. NSPH [26] proposes a simple yet effective method for fine-grained image retrieval. The model adds quantization and bit-balance losses to the hash layer. Quantization loss reduces the error caused by binarizing real-valued feature representations to hash codes, and bit balance loss weakens hash code bias. CSH [27] proposed a simple but effective approach to self-supervised hash learning based on dual pseudo-agreement. By adding a consistency constraint, this method can prevent corrupted labels and encourage generalization for effective knowledge distillation. A novel deep ordinal hashing method(DOH) [28] learns ordinal representations to generate ranking-based hash codes by leveraging the ranking structure of the feature space from both local and global views. In the hashing framework, the local spatial and global semantic nature of the images are captured in an end-to-end ranking-to-hashing manner. RODH [29] directly generates discrete hash codes from raw images by balancing the effective category-level information of discretization and discrimination of ranking information. A deep architecture that learns instance-aware image representations for multi-label image data [30] was proposed. It is organized into multiple groups, with each group containing the features for one category. Li et al. proposed a feature learning based deep supervised hashing with pairwise labels (DPSH) algorithm based on tag pairs [31], which constructed an image tag pair matrix through image category labels, and then constructed a cross-entropy loss function based on the image tag pairs. The algorithm relaxed the constraint conditions, removed the constraint conditions of the symbolic function, and solved the discrete constraint problem using the relaxation optimization method based on the Lagrange multiplier method. However, some hash bits would be excessively relaxed, resulting in incomplete semantic information between similar point pairs because the algorithm uses Lagrange multipliers.

To avoid the influence of the relaxation solution on the accuracy of the model and the influence of the inaccuracy of the similarity matrix decomposition on the subsequent quantization process, this study proposes a non-relaxation deep hashing method for fast image retrieval(NRDH). Fig.1 provides an overview of the proposed method. The main contributions of this study are summarized as follows:

(1) A deep hashing learning framework is proposed, which is an end-to-end structure. The framework generates discrete hash codes directly from raw images, and integrates image feature extraction and discrete hash learning modules into

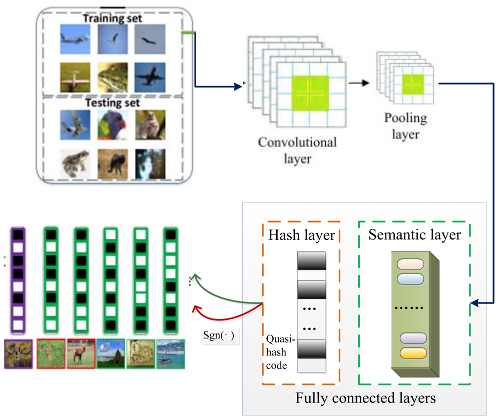


FIGURE 1. Overview of non-relaxation deep hashing method frame.

a unified framework. An improved network structure and suitable hash generation function are designed to solve the problem of non-derivable discrete space in deep hashing.

(2) A loss function is elaborately designed with the norm to constrain each bit of the hash-like code to be as binary as possible.

(3) To reduce the quantization error, a host threshold function is used at the network output to make the output quasi-hash code non-linearly close to -1 or 1. The symbolic function sgn() is used outside the model to quantize the quasi-hash code into binary value code.

II. PROPOSED METHOD

A. PROBLEM DESCRIPTION AND DEFINITION

1) PROBLEM DESCRIPTION

We define a dataset $X = \{x_i\}_{i=1}^n \in R^{d\times n}$ of n images, where $x_i \in R^d$ is each input image, and d is the size of the image. The hash code corresponding to the output image is $B \in \{-1, 1\}^{l\times n}$, where $b_i \in \{-1, 1\}^l$ is the i-th column of output data $\mathbf{B}$, which represents the binary hash code of the i-th sample data x_i, and the length is l. The purpose of perceptual hash learning is to obtain an automatically learned hash function, $H()$, by training from the training set data.

Suppose that an image is represented as $b_i = H(x_i) = [h_1(x_i), \ldots\ldots h_l(x_i)]$ using a hash function. In the linear hash coding function, the hash function $h_i()$ maps an image to a hash bit, and the l hash functions map an image to a string of l -bit binary hash codes. In the supervised hashing algorithm, each image sample data point has a label, and the label matrix is $Y = \{y_i\}_{i=1}^n \in R^{c\times n}$, where c denotes the number of categories. The sample data x_i and x_j are related by similarity matrix S_{ij}. For any two data samples (x_i and x_j), if x_i and x_j are similar, then $S_{ij} = 1$, otherwise $S_{ij} = 0$.

2) CROSS ENTROPY LOSS FUNCTION

For any two hash codes of equal length, b_i and b_j, of equal length, the similarity ϕ_{ij} of the two hash codes is defined by their inner product:

$$\phi_{ij} = \phi(b_i, b_j) = \frac{1}{2} b_i^T b_j \tag{1}$$

The larger the inner product, the greater the similarity. The similarity ϕ_{ij} is thresholded non-linearly using the sigmoid function, and its range is normalized to the interval (0,1). ϕ_{ij} can be obtained:

$$\sigma(\phi_{ij}) = \frac{1}{1 + e^{-\phi_{ij}}} \tag{2}$$

The cross-entropy loss function is used to maintain the similarity between image point pairs based on the similarity measure of hash codes. The likelihood $Q(s_{ij}|B)$ between the hash code and similarity of the image point pairs is defined as

$$Q(s_{ij}|B) = \begin{cases} \sigma(\phi_{ij}), s_{ij} = 1 \\ 1 - \sigma(\phi_{ij}), s_{ij} = 0 \end{cases} \tag{3}$$

s_{ij} represents the similarity between sample pairs in Eq.(3), where $\boldsymbol{B}$ represents the hash code corresponding to sample data. The likelihood function shows that when the hash codes b_i and b_j are more similar, that is, the larger the $\sigma(\phi_{ij})$, the larger the corresponding likelihood function $Q(s_{ij}|B)$. The negative logarithm of the likelihood is the cross-entropy loss function, which can be expressed as

$$Loss1(B) = -\sum_{s_{ij} \in s} \log Q(s_{ij}|B) = -\sum_{s_{ij} \in s} [s_{ij}\phi_{ij} - \log(1+e^{\phi_{ij}})] \tag{4}$$

The maximum likelihood estimation was converted to minimize the cross-entropy loss function, and the constrained optimization problem was established as follows:

$$\min_B -\sum_{s_{ij} \in s} [s_{ij}\phi_{ij} - \log(1 + e^{\phi_{ij}})]$$
$$b_i = sgn[W^T \varphi(x_i; \theta) + v], \forall i = 1, 2, \cdots n \tag{5}$$

W represents the neuron parameters of the fully connected layer in Eq.(5), v represents the offset; θ represents the parameter set of the network convolution layer; b_i represents the binary hash code; and $\varphi(\cdot)$ represents the image features extracted by the network. In b_i, each bit is quantized to a discrete value of -1 or 1.

B. NETWORK ARCHITECHTURE

In this study, adjustments are made based on the CNN-F network structure. The network structure is shown in Fig.2, and consists of five convolutional layers, three pooling layers, and four fully connected layers. In this network structure, a fully connected layer, FUL7, is added before the binary hash code is generated to learn the parameters before the saturated activation function is input. The data are processed using local response normalization (LRN) in each convolutional layer, oversampled to automatically extract the feature representation of the image, and then the result is output through a fully connected layer.

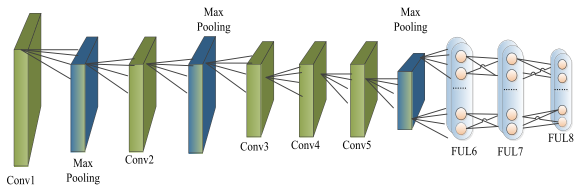


FIGURE 2. Network structure diagram.

The process of feature extraction and hash coding and quantization in this network is:

(1) Through the two convolutional layers of Conv1 and Conv2, a series of underlying visual features, such as the color, brightness, layout, and texture of the image, are extracted.

(2) The maximum pooling layer reduces the dimension of the feature value passed in from the previous layer to increase the rotational invariance of the feature and reduce the computational load of the network.

(3) The extracted image features are combined through the last three layers of Conv to generate a higher-level feature semantic representation.

(4) The high-level semantic features of the generated images are embedded in FUL6. FUL7 is the semantic layer that provides adjustable parameter learning generated by the perceptual hash function, and FUL8 is the output layer of the quasi-hash code.

C. LOSS FUNCTION

Cross-entropy was used to maintain the semantic similarity between sample pairs in the proposed method. To reduce the quantization error of the quasi-hash code output by the network, the $\ell 1$ norm is used to constrain the distribution of the quasi-hash code output by the network as follows:

$$Loss2(B) = \sum_{i=1}^{n} \left\| |b_i| - 1 \right\|_1 \tag{6}$$

The purpose of this regular term is to make each hash bit of the quasi-hash code b_i approximately equal to two discrete values -1 or 1; that is, the closer the absolute value of each bit in b_i is to 1, the smaller the loss. In the iterative process, the output of the network is directly used as quasi-hash code with a high probability of excessive deviation from -1 or 1, which increases the $Loss2(B)$ loss value. Although the sign function can quantify this well, it is not differentiable. Compared with the discrete encoding of the sign function, the hyperbolic tangent function can make each hash bit of the quasi-hash code non-linearly close to -1 or 1 and has the good property of continuous infinite derivation, and its form is

$$\tanh(x) = \frac{1 - e^{-2x}}{1 + e^{-2x}} \tag{7}$$

Based on the hyperbolic tangent function, the parameter μ is added to control the slope of the threshold function, and the optimized function $host(x)$ is

$$host(x) = \frac{1-e^{-\mu x}}{1+e^{-\mu x}} \tag{8}$$

Unlike the HashNet algorithm, the HashNet algorithm gradually increased the scale coefficient of the threshold function during the training process. With an increase in the number of iterations, the threshold function continues to approach and finally converges to the sign function. In the proposed algorithm, the new threshold coefficient μ is the model parameter. After obtaining the optimal parameters through experiments, μ remains unchanged in each iteration, which simplifies the model and causes it to converge quickly. The $host(x)$ threshold function is used for the network output in the proposed method. The discrete constraint problem is transformed into a problem of deriving the model parameters using a differentiable loss function. In the iterative process, the $host(x)$ threshold function mapped the output of the network smoothly and nonlinearly to $(-1,1)$. Most hash bits are thresholded at approximately -1 and 1.

By combining the cross-entropy loss function in Eq.(4)and the regular term in Eq.(7), and using the threshold function $host(x)$ at the output of the network structure, the objective function is established as

$$\begin{aligned} \min_B LS &= -\sum_{s_{ij}\in s}[s_{ij}\phi_{ij} - \log(1+e^{\phi_{ij}})] \\ &\quad + \beta\sum_{i=1}^{n}\||b_i|-1\|_1 \\ b_i &= host[W^T\varphi(x_i;\theta)+v], \forall i = 1,2,\cdots n \\ \phi_{ij} &= \frac{1}{2}b_i^T b_j, \forall i,j = 1,2\cdots n \end{aligned} \tag{9}$$

In Eq.(10), n represents the number of samples, $S_{ij} \in \{0, 1\}$ represents whether the sample i and sample j are similar, β represents the regular term coefficient, $host(x)$ represents the threshold function, μ represents the control parameter of the threshold function, and b_i represents the quasi-hash code output by the forward network, ϕ_{ij} represents the similarity between two hash codes.

D. PARAMETER LEARING

The variable parameters v and W are solved using back propagation(BP), and in each iteration, v and W are updated using stochastic gradient descent(SGD). During the training process, the two terms of the objective function are represented by LS_1 and LS_2 respectively:

$$\begin{aligned} LS_1 &= -\sum_{s_{ij}\in s}[s_{ij}\phi_{ij} - \log(1+e^{\phi_{ij}})] \\ LS_2 &= \beta\sum_{i=1}^{n}\Big\||b_i|-1\Big\|_1 \end{aligned} \tag{10}$$

1) PARTIAL DERIVATIVES FOR QUASI HASH CODE

Find the partial derivatives of the first item LS_1 with respect to b_i, get:

$$\frac{\partial LS_1}{\partial b_i} = \frac{1}{2}\sum_{j:s_{ij}\in S}(m_{ij}-s_{ij})b_j + \frac{1}{2}\sum_{j:s_{ij}\in S}(m_{ji}-s_{ji})b_j \tag{11}$$

there, $m_{ij} = \sigma(\frac{1}{2}b_i^T b_j)$

Find the partial derivatives of the second item LS_2 with respect to b_i, get:

$$\frac{\partial LS_2}{\partial b_i} = \beta\delta(b_i) \tag{12}$$

there, $\delta(x) = \begin{cases} 1, & -1 \le x \le 0 \quad or\ x \ge 1 \\ -1, & otherwise \end{cases}$

Combining Eq.(11) and Eq.(12), the partial derivative of the loss function LS with respect to b_i can be obtained as follows:

$$\frac{\partial LS}{\partial b_i} = \frac{1}{2}\sum_{j:s_{ij}\in S}(m_{ij}-s_{ij})b_j + \frac{1}{2}\sum_{j:s_{ij}\in S}(m_{ji}-s_{ji})b_j + \beta\delta(b_i) \tag{13}$$

2) PARTIAL DERIVATIVES FOR PARAMETER *W*

Find the partial derivatives of the loss function LS with respect to W, get:

$$\frac{\partial LS}{\partial W} = \frac{\partial b_i}{\partial W}\cdot\left[\frac{\partial LS}{\partial b_i}\right]^T \tag{14}$$

There, $\frac{\partial b_i}{\partial W}$ was got by $host'(x) = \frac{2\mu e^{-ux}}{(1+e^{-\mu x})^2}$

$$\frac{\partial b_i}{\partial W} = \varphi(x_i;\theta)\cdot\frac{2ue^{-u\alpha}}{(1+e^{-u\alpha})^2} \tag{15}$$

$$\alpha = W^T\varphi(x_i;\theta)+v \tag{16}$$

The partial derivatives of the loss function LS with respect to W can be obtained:

$$\frac{\partial LS}{\partial W} = \varphi(x_i;\theta)\cdot\frac{2ue^{-u\alpha}}{(1+e^{-u\alpha})^2}\cdot\left[\frac{\partial LS}{\partial b_i}\right]^T \tag{17}$$

3) PARTIAL DERIVATIVES FOR PARAMETER *v*

Similar to 2), the partial derivatives of the loss function LS with respect to v are obtained as follows:

$$\frac{\partial LS}{\partial v} = \frac{\partial LS}{\partial b_i}\cdot\frac{2ue^{-u\alpha}}{(1+e^{-u\alpha})^2} \tag{18}$$

It can be seen from the above solution process that the loss function is derivable, that is, the partial derivative of the backpropagation process exists, and the network model can converge after a certain iteration.

```
Algorithm 1 NRDH
Input: A dataset $X$ of $n$ images
Output: Hash code $B$ corresponding to $n$ images, network parameter.
Initialization: Gaussian distribution to initialize weights $W$ and offsets $v$.
Iteration process:
    1) Read the data from the training set and preprocess it, and input the samples into the network;
    2) Calculate the corresponding $B$ through the BP algorithm;
    3) Calculate the loss by using the loss function;
    4) Calculate the gradient layer by layer by using the SGD algorithm, adjust the network parameters, and update the parameters $W$ and $v$;
Stop: The maximum number of iterations is reached.
Return: network parameters $W$ and $v$, sgn( ), and the hash code $B$ by forward propagation.
```

E. ALGORITHM DESCRIPTION

The pseudocode for the algorithm used in this study is shown in Algorithm 1. The input data are an image in a certain format, and the feature representation of the image is extracted through convolutional and pooling layers. According to the objective function, the parameters v and W are updated using the back-propagation algorithm, and the model is completed using a specified number of iterations. Finally, a symbolic function is used outside the trained model to quantize the quasi-hash code of the image and output a binary hash code.

III. EXPERIMENTS AND ANALYSIS

A. DATASETS

We compare our proposed method with other state-of-the-art methods by using two widely used benchmark datasets.

1) *CIFAR-10*: This dataset consists of 60,000 32 × 32 color images divided into 10 classes (6000 images per class). Among these, 50,000 are the training set and 10,000 are the test set. It is a single-label dataset in which each image belongs to one of ten classes. The images are resized to 224 × 224 pixels before being inputted into the CNN-based models.

2) NUS-WIDE: This dataset contain 269,648 images gathered from Flickr. It is a multi-label dataset, where each image belongs to one or multiple class labels from 81 classes. In this experiment, 21 commonly used categories are selected, with each category containing at least 5000 images.

In the experimental stage, in the CIFAR-10 data-set, 500 images are randomly selected from each category as the training data and 100 images as the test data. A total of 5000 images are obtained from the training set and 1000 images are obtained from the test set. In the NUS-WIDE data-set, 500 images are randomly selected from each category as training data and 100 images as test data. A total of 10500 images from the training set and 2100 images from the test set are used, and each image is adjusted to 224 × 224. The size of 224 is suitable for the input of the network model. To verify the robustness of the algorithm, the norm regular term coefficients are both set to 0.05 on the CIFAR-10 and NUSWIDE datasets, and the threshold function control parameters μ are all set to 24.

TABLE 1. The map of different algorithms on CIFAR-10.

Method	CIFAR-10			
	16bits	32bits	64bits	128bits
NRDH	0.815	0.824	0.855	0.861
NSPH	0.810	0.819	0.849	0.857
RODH	0.809	0.811	0.846	0.855
FPH	0.743	0.752	0.775	0.792
CSH	0.789	0.801	0.835	0.848
DOH	0.741	0.749	0.767	0.781
HashNet	0.670	0.686	0.696	0.706

B. EXPERIMENTAL RESULTS

Because a fixed size 224 × 224 image is used as input in the network, the images in CIFAR-10 and NUS-WIDE are scaled to 224×224 before training. To eliminate the commonality of the images and facilitate computer understanding, the mean value of the entire image dataset is subtracted from each image in the experiment, which can also construct the central data distribution so that the gradient descent algorithm can operate quickly and efficiently.

After selecting a certain length of hash code during the experiment, in the test set, we select a part of the images as the samples to be retrieved, calculate the Hamming distance between the images to be retrieved and other images in the data-set, and sort and calculate the images according to the Hamming distance. The ratio of the number of images in the same category as the images to be retrieved in the sorted list to the total number of retrieved images is used as the accuracy rate.

The NRDH method is compared with several popular hash learning algorithms: FPH [16], HashNet [18], NSPH [26], CSH [27], DOH [28], and RODH [29]. Table 1 compares the MAP of our DNRH algorithm and existing hash learning algorithms on the CIFAR-10 dataset. As shown in Table 1, the average accuracy of our DHFR algorithm for the four hash code lengths is significantly higher than that of all other algorithms. By comparing the MAPs of the six deep hashing algorithms and other 6 non-deep hashing algorithms, it can be seen that the deep hashing algorithm has a higher average accuracy than the non-deep hashing algorithms, which shows that the deep hashing learning algorithm using the CNN model automatically extracts image feature representations with better performance than the traditional manual extraction of image feature representations. For a hash code of 16 bits, the retrieval accuracy of all algorithms is relatively low. As the length of the hash code increases, the retrieval accuracy of all algorithms gradually increases. When the hash code length is 64, the retrieval accuracy of all algorithms

TABLE 2. The map of different algorithms on NUS_WIDE.

Method	NUS_WIDE			
	16bits	32bits	64bits	128bits
NRDH	0.801	0.818	0.829	0.833
NSPH	0.798	0.801	0.814	0.823
RODH	0.788	0.790	0.812	0.826
FPH	0.708	0.735	0.748	0.758
CSH	0.787	0.789	0.801	0.817
DOH	0.702	0.731	0.745	0.749
HashNet	0.713	0.738	0.755	0.762

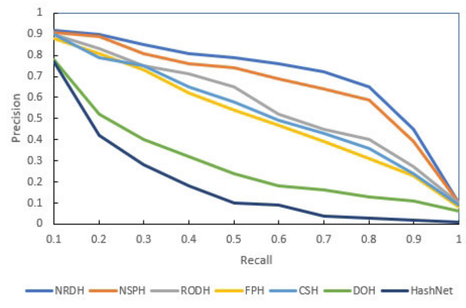
reaches compared with the hash code below 64bits, using the hash code of 48bits can store more image features, and can use more image features during retrieval to achieve higher accuracy.

Compared with the images of the CIFAR-10 dataset, the images of the NUS-WIDE dataset have higher pixels, more complete image details, and are closer to the images in practical applications. In the NUS-WIDE dataset, an image may contain multiple images during the retrieval process; as long as the retrieved image and the image to be retrieved contain the same label, it is judged as correct retrieval.

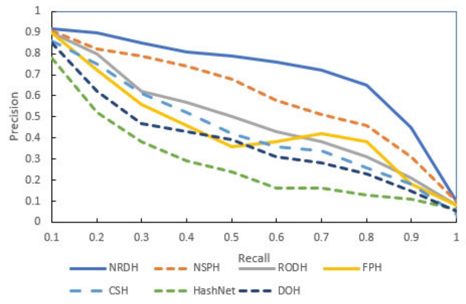
Table 2. compares the average accuracy of our NRDH algorithm with existing hash learning algorithms for hash codes of different lengths on the NUSWIDE dataset. Due to the large number of images in the NUS-WIDE dataset, on this dataset, this paper uses the first 5000 samples retrieved from each test sample to calculate the MAP. For hash codes of the same length, the average accuracy rates of the DHFR algorithm in this paper for 16bits, 32bits, 64bits, and 128bits are 0.801, 0.818, 0.829, and 0.833, respectively, which are higher than those of other hash learning algorithms. This proves the universality of the algorithm used in this study. Among them, for the RODH algorithm, we still used the Lagrange multiplier $u = 10$ experiment on the CIFAR-10 dataset and re-ran the NUS-WIDE dataset with various algorithms using the same training set and test set. We ran the code provided by the authors and calculated their average accuracy. As the length of the hash code increases, the average retrieval accuracy of almost all algorithms increases to a certain extent, particularly for the FPH algorithm, and the average accuracy of the 48-bit hash code is higher than that of the 12-bit hash code. This is nearly 7%, indicating that more hash bits can represent more image features and improve retrieval accuracy.

Owing to the large number of images in the NUS-WIDE dataset, the MAP is obtained using the first 5000 samples returned. For hash codes of the same length, the average accuracy rates of the NRDH algorithm in this study are higher than those of the other hash learning algorithms, which proves the universality of the algorithm in this study.

In addition to MAP, we evaluate our method using precision curves of 64-bit hash codes with different recall rates, as shown in Fig. 3, which not only reflects the precision of the search results but also reflects the recall rate. The larger the



(a) NUS-WIDE



(b) CIFAR-10

FIGURE 3. PR curves for algorithms in the two datasets and β.

area enclosed by the PR curve and axes, the better the retrieval performance. As shown in Fig.3(a), our work is outstanding among all methods. Fig.3(b) shows the precision rates of 64-bit hash codes for different numbers of top-returned images. According to the precision curves, as the number of returned samples increased, the precision rate of the deep hashing method decreased slightly. By contrast, our approach always returns positive samples with satisfaction.

C. ABLATION STUDY ON LOSS FUNCTION

In the NRDH algorithm proposed in this study, the role of the host threshold function is to directly threshold the results of the network output in the forward calculation of the model, and the $\ell 1$ norm is used as the regular term of the objective function to constrain the quasi-hash in the back-propagation of the model. The function of these two modules is to constrain quasi-hash code. In order to verify the constraint performance of the combined use of the $\ell 1$ norm and the host threshold function, this paper uses the CIFAR-10 data On the set, experiments are carried out on the $\ell 1$ norm regular term independent constraint, the host threshold function independent constraint, and the $\ell 1$ norm and host threshold function joint constraints.

Table 3 lists the average accuracy rates corresponding to different models on hash codes of four lengths, where ''cross entropy + host threshold'' means using the loss function of Eq.(4), and using host at the output of the network, ''cross entropy + $\ell 1$ norm'' means using the loss function of formula (10), and omitting the constraints, that is, the model that does not use the host threshold function at the output of

TABLE 3. The map of $\ell 1$ norm and host threshold function constraint on CIFAR-10.

Method	16bits	32bits	64bits	128bits
cross entropy + $\ell 1$ norm + host threshold	0.815	0.824	0.855	0.861
cross entropy + host threshold	0.702	0.737	0.784	0.792
cross entropy+ $\ell 1$ norm	0.689	0.703	0.744	0.761

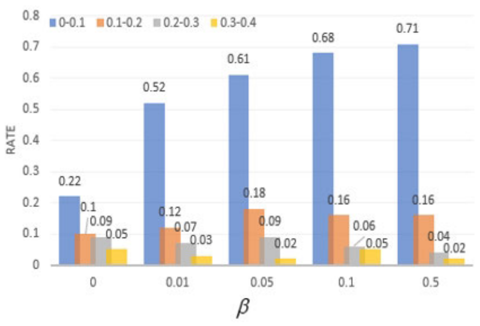


FIGURE 4. The distribution of hash code with different Regularization coefficient β.

the network, "cross entropy + $\ell 1$ norm" + host threshold" represents the NRDH algorithm model in this paper, that is, using the $\ell 1$ norm and the host threshold function jointly. Observing Table 3, it can be seen that the average accuracy of the two models "cross entropy + $\ell 1$ norm" and "cross entropy + host threshold" is lower, indicating that the effect of using the $\ell 1$ norm and the host threshold function alone is not as good as the algorithm for Lagrange multiplier relaxation solution. The combined use of the $\ell 1$ norm and the host threshold function (cross entropy + $\ell 1$ norm + host threshold) in the length of the 4-length hash code, its MAP is compared to using one of the modules alone. Both improved by nearly 10% and are higher. Therefore, it can be concluded that the combined use of the $\ell 1$ norm and the host threshold function can better constrain the hash code and improve the performance of the algorithm.

D. PARAMETER IMPACT ANALYSIS

1) INFLUENCE OF REGULAR TERM COEFFICIENT ON QUASI-HASH CODE DISTRIBUTION

To test the constraining ability of the $\ell 1$ norm regular term on the quasi-hash code output by the fully connected layer, this study presents statistics on the distribution of the output quasi-hash code in the CIFAR-10 dataset. The distance of the absolute value of one bit relative to one is distributed in the intervals [0, 0.1), [0.1, 0.2), [0.2, 0.3), and [0.3, 0.4). Fig. 4 shows the distribution of the quasi-hash code. In different cases, different colors represent different distribution intervals; the horizontal axis represents the regular term coefficient and the vertical axis represents the percentage of hash bits that fall in different intervals.

TABLE 4. The map of different β.

β	0	0.01	0.05	0.1	0.5
CIFAR-10	0.636	0.721	0.767	0.743	0.694
NUS-WIDE	0.662	0.774	0.813	0.757	0.681

It can be seen from the distribution of each hash bit of the quasi-hash code in Fig.4, that with the increase of β, the absolute value of each hash bit of the quasi-hash code is closer to 1, especially when the $\ell 1$ norm is not used (In the case of $\beta = 0$) constraint, the hash bits of the quasi-hash code are relatively evenly distributed between 0 and 0.4, so that the loss will increase in the final quantization process, resulting in inaccurate results. In the objective function, the true term is used to maintain the similarity between point pairs, and the $\ell 1$ norm regular term is used to constrain the quasi-hash code distribution. If the regular-term coefficient β is too large, the proportion of the $\ell 1$ norm regular term increases excessively, thereby reducing semantic preservation. The function of the true term affects classification. It can be seen that the appropriate $\ell 1$ norm regular term has a good constraining effect on the distribution of the hash code.

2) INFLUENCE OF REGULAR TERM COEFFICIENT β ON EXPERIMENTAL ACCURACY

The value of the regular-term coefficient β not only affects the distribution of each hash bit of the quasi-hash code output by the model but also affects the accuracy of the model trained by the NRDH algorithm in this study. Table 4 shows that when the length of the hash code is 64, the average accuracy of different values of β on the CIFAR-10 and NUS-WIDE datasets.

It can be observed from Table 4 that the value of β has the same distribution of influence on MAP in the two datasets. When $\beta = 0.05$, the retrieval effect on the test set is the best, and if the value of β is too small or too large, the retrieval will be affected. This is because the β value is too small, the constraint of the objective function aligned with the hash code distribution becomes weaker, and some of the hash bits of the quasi-hash code output by the model deviate significantly from -1 or 1, resulting in the final result. The loss increases when it is quantized into hash code. If the value of β is too large, the proportion of semantic fidelity items in the objective function will decrease. This proportion causes the distance between the same categories to increase or the distance between different categories to decrease, that is, the similarity constraints between images become weaker, which worsens the retrieval effect.

E. RESULTS VISUALIZATION

A visualization of the results is shown in Fig.5. When the number of coding bits is 64, all methods in the experiment used the Hamming distance directly for the retrieval instances of trucks in the CIFAR-10 image database. In the experiment, the first 36 images with the smallest Hamming

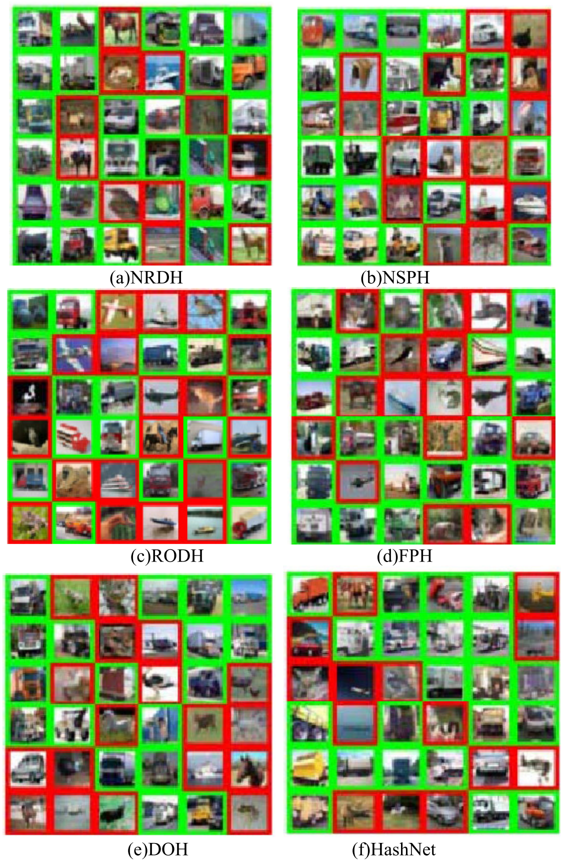


FIGURE 5. **Results visualization.**

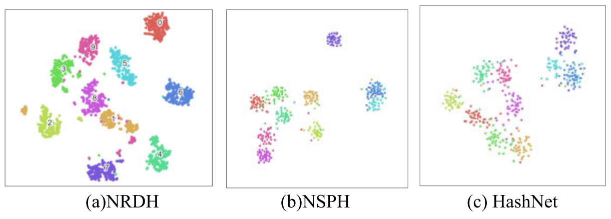


FIGURE 6. **The T-SNE of hash codes learned by our method and NSPH and HashNet.**

distance from the query image are used as returned results. The images marked in the red box are not related to the query image. It can be observed that the method in this study achieved better results than the other image-hashing algorithms.

In addition, we visualize the T-SNE of hash codes generated by NSPH, HashNet, and our method on the CIFAR-10 image database in Fig.6. For simplicity, we sample 10 categories. We observe that the hash codes generated by our method in different classes are well separated, and those in the same class are more compact. This suggests that the hash codes generated by the proposed method is more discriminative than those generated by the other two methods.

IV. CONCLUSION

To avoid the influence of the relaxation solution on the accuracy of the model and the influence of the inaccuracy of the similarity matrix decomposition on the subsequent quantization process, a non-relaxation deep hashing method was proposed to achieve effective and efficient large-scale image retrieval. To demonstrate the advantages of the proposed method, extensive experimental studies are conducted, and the results show that the proposed method significantly outperforms other hashing methods on benchmark datasets. In future work, it will be interesting and promising to develop a theoretical framework to further optimize the performance and apply the framework to other types of data (e.g., audio, video, and text).

REFERENCES

[1] L. Zheng, Y. Yang, and Q. Tian, "SIFT meets CNN: A decade survey of instance retrieval," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 40, no. 5, pp. 1224–1244, May 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. 2019.

[3] T. Li, Z. Zhang, L. Pei, and Y. Gan, "HashFormer: Vision transformer based deep hashing for image retrieval," *IEEE Signal Process. Lett.*, vol. 29, pp. 827–831, 2022.

[4] C. Qin, L. Wu, X. Zhang, and G. Feng, "Efficient non-targeted attack for deep hashing based image retrieval," *IEEE Signal Process. Lett.*, vol. 28, pp. 1893–1897, 2021.

[5] Z. Weng and Y. Zhu, "Online supervised sketching hashing for large-scale image retrieval," *IEEE Access*, vol. 7, pp. 88369–88379, 2019.

[6] S. Cheng, L. Wang, and A. Du, "An adaptive and asymmetric residual hash for fast image retrieval," *IEEE Access*, vol. 7, pp. 78942–78953, 2019.

[7] M. Datar, N. Immorlica, P. Indyk, and V. S. Mirrokni, "Locality-sensitive hashing scheme based on p-stable distributions," in *Proc. 20th Annu. Symp. Comput. Geometry*, New York, NY, USA, Jun. 2004, pp. 253–262.

[8] B. Kulis and K. Grauman, "Kernelized locality-sensitive hashing for scalable image search," in *Proc. IEEE 12th Int. Conf. Comput. Vis.*, Kyoto,Japan, Sep. 2009, pp. 2130–2137.

[9] A. Andoni and P. Indyk, "Near-optimal hashing algorithms for approximate nearest neighbor in high dimensions," in *Proc. 47th Annu. IEEE Symp. Found. Comput. Sci. (FOCS)*, Berkeley, CA, USA, Oct. 2006, pp. 459–468.

[10] Y. Weiss, A. Torralba, and R. Fergus, "Spectral hashing," in *Proc. Adv. Neural Inf. Process. Syst.*, 2008, pp. 1753–1760.

[11] 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.

[12] J.-P. Heo, Y. Lee, J. He, S.-F. Chang, and S.-E. Yoon, "Spherical hashing," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Providence, RI, USA, Jun. 2012, pp. 2957–2964.

[13] W. Liu, J. Wang, R. Ji, Y.-G. Jiang, and S.-F. Chang, "Supervised hashing with kernels," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Providence, RI, USA, Jun. 2012, pp. 2074–2081.

[14] M. Norouzi and D. M. Blei, "Minimal loss hashing for compact binary codes," in *Proc. Int. Conf. Mach. Learn.*, 2011, pp. 353–360.

[15] F. Shen, C. Shen, W. Liu, and H. T. Shen, "Supervised discrete hashing," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Boston, MA, USA, Jun. 2015, pp. 37–45.

[16] Y. Yang, L. Geng, H. Lai, Y. Pan, and J. Yin, "Feature pyramid hashing," in *Proc. Int. Conf. Multimedia Retr.*, Jun. 2019, pp. 114–122.

[17] Q. Li, Z. Sun, R. He, and T. Tan, "Deep supervised discrete hashing," in *Proc. NIPS*, 2017, pp. 2482–2491.

[18] Z. Cao, M. Long, J. Wang, and P. S. Yu, "HashNet: Deep learning to hash by continuation," in *Proc. IEEE Int. Conf. Comput. Vis. (ICCV)*, Venice, Italy, Oct. 2017, pp. 5609–5618.

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

[20] Y. Cao, M. Long, B. Liu, and J. Wang, "Deep Cauchy hashing for Hamming space retrieval," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, Salt Lake City, UT, USA, Jun. 2018, pp. 1229–1237.

[21] Y. Cao, M. Long, J. Wang, H. Zhu, and Q. Wen, "Deep quantization network for efficient image retrieval," in *Proc. AAAI*, 2016, pp. 3457–3463.

[22] K. Lin, J. Lu, C.-S. Chen, and J. Zhou, "Learning compact binary descriptors with unsupervised deep neural networks," in *Proc. CVPR*, Jun. 2016, pp. 1183–1192.

[23] J. Donahue, P. Krähenbühl, and T. Darrell, "Adversarial feature learning," in *Proc. ICLR*, 2017, pp. 1–11.

[24] X. Zhou, F. Shen, L. Liu, W. Liu, L. Nie, Y. Yang, and H. T. Shen, "Graph convolutional network hashing," *IEEE Trans. Cybern.*, vol. 50, no. 4, pp. 1460–1472, Apr. 2020.

[25] J. Li, W. W. Y. Ng, X. Tian, S. Kwong, and H. Wang, "Weighted multi-deep ranking supervised hashing for efficient image retrieval," *Int. J. Mach. Learn. Cybern.*, vol. 11, no. 4, pp. 883–897, Apr. 2020.

[26] H. Sun, Y. Fan, J. Shen, N. Liu, D. Liang, and H. Zhou, "A novel semantics-preserving hashing for fine-grained image retrieval," *IEEE Access*, vol. 8, pp. 26199–26209, 2020.

[27] Y. Li, Y. Wang, Z. Miao, J. Wang, and R. Zhang, "Contrastive self-supervised hashing with dual pseudo agreement," *IEEE Access*, vol. 8, pp. 165034–165043, 2020.

[28] L. Jin, X. Shu, K. Li, Z. Li, G.-J. Qi, and J. Tang, "Deep ordinal hashing with spatial attention," *IEEE Trans. Image Process.*, vol. 28, no. 5, pp. 2173–2186, May 2019.

[29] X. Lu, Y. Chen, and X. Li, "Discrete deep hashing with ranking optimization for image retrieval," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 31, no. 6, pp. 2052–2063, Jun. 2020.

[30] H. Lai, P. Yan, X. Shu, Y. Wei, and S. Yan, "Instance-aware hashing for multi-label image retrieval," *IEEE Trans. Image Process.*, vol. 25, no. 6, pp. 2469–2479, Jun. 2016.

[31] W. Li, S. Wang, and W. Kang, "Feature learning based deep supervised hashing with pairwise labels," in *Proc. 25th Int. Joint Conf. Artif. Intell.*, New York, NY, USA, 2016, pp. 1711–1717.

XIAOFEI LI received the master's degree in computer application technology from Jilin University, in 2012. She is currently an Associate Professor of data science and big data technology with the Jilin University of Architecture and Technology. Her research interests include image processing, image retrieval, and data analysis and mining.